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»
на тему «Деревья и леса решений»

Студенты гр. 1301 _____ Семейкин С.А.

_____ Гальченко М.А.

Преподаватель _____ Боброва Ю.О.

Санкт-Петербург
2025

Цель:

Реализация классификатора на основе дерева принятия решений и исследование его свойств.

Ход работы:

1. Создадим переменные, распределение по нормальному закону с незначительно различными средними и дисперсиями.
2. Создадим переменные, соответствующие классам.
3. Обучим классификаторы (дерево решений и случайный лес) на обучающем наборе данных.
4. Создадим гистограммы распределения вероятности на выходе классификатора и таблицы с данными об эффективности для 3х выборок (нормальное распределение: хорошо и плохо разделимы, а также нелинейно-пересекаемые).
5. Для каждой выборки — таблицы с результатами оценки качества классификации (точность, чувствительность, специфичность).

mu0 = [0, 2] mu1 = [3, 5] sigma0 = [1, 1] sigma1 = [1, 2]

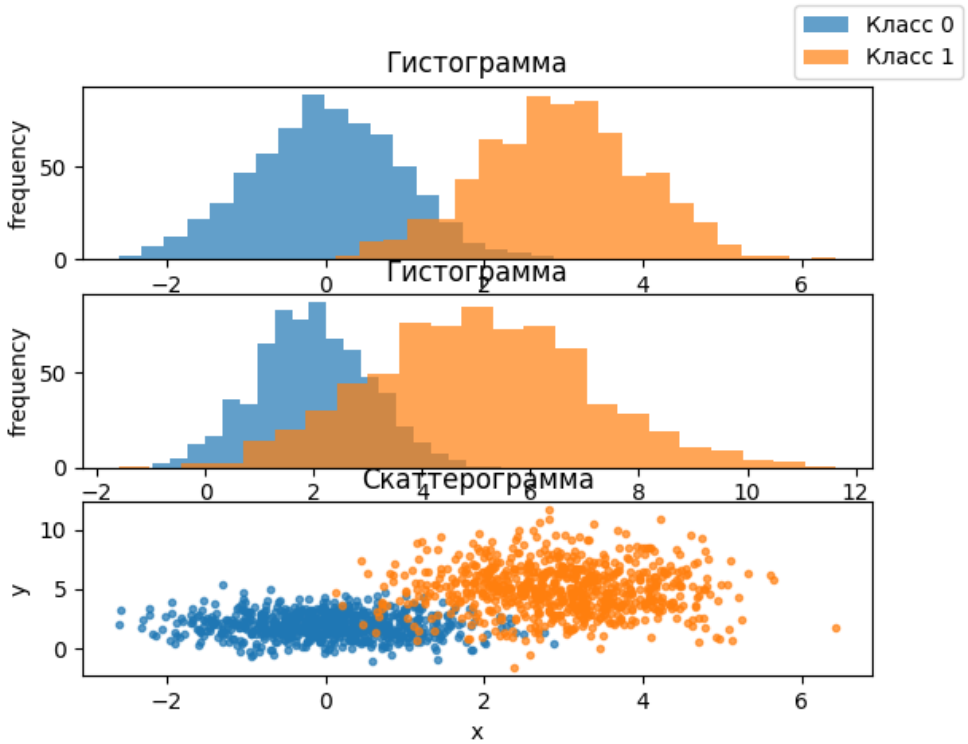

Рисунок 1 Гистограммы и скаттерограммы средне разделимой выборки

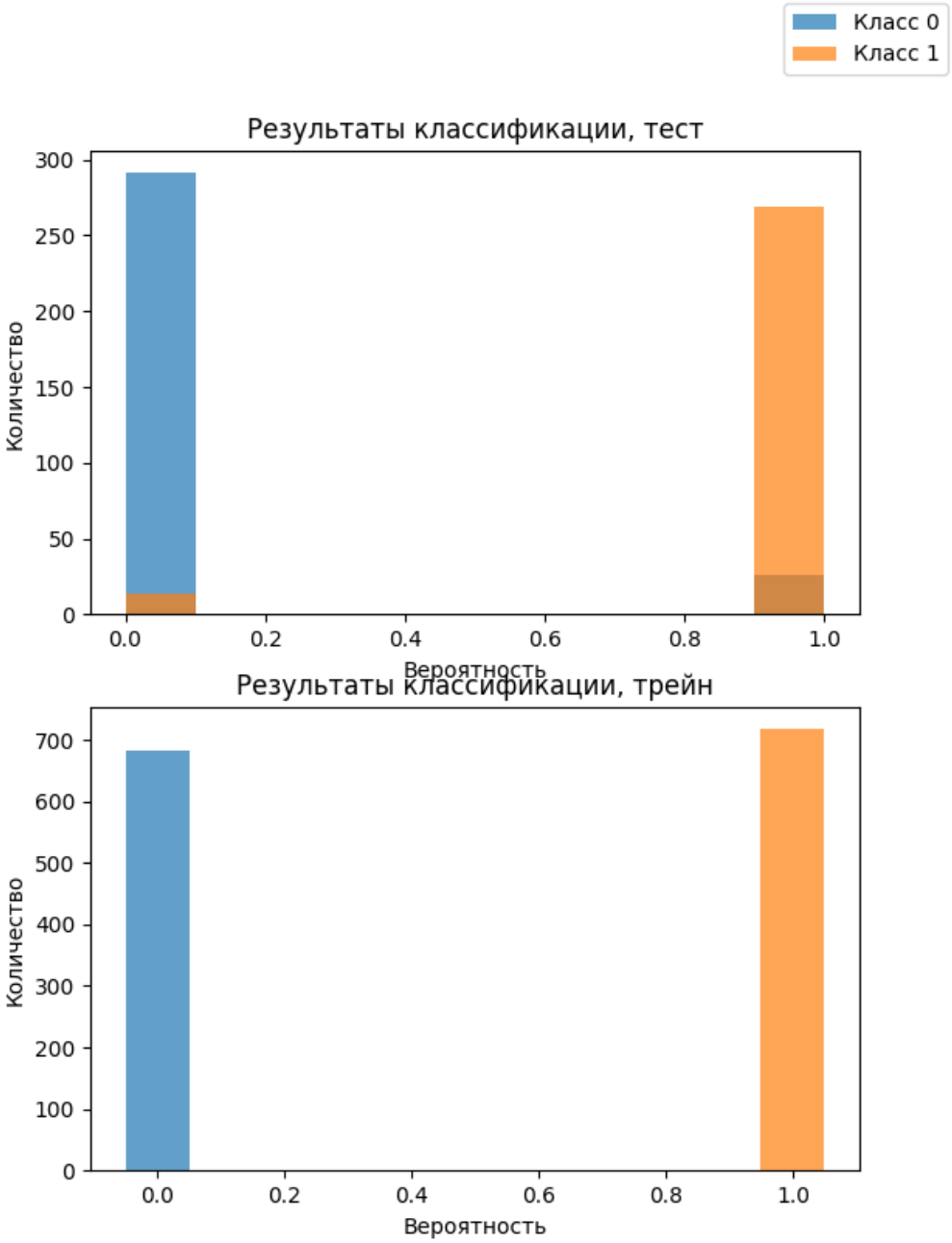

Рисунок 2 Распределение вероятностей для средне разделимой выборки при использовании дерева

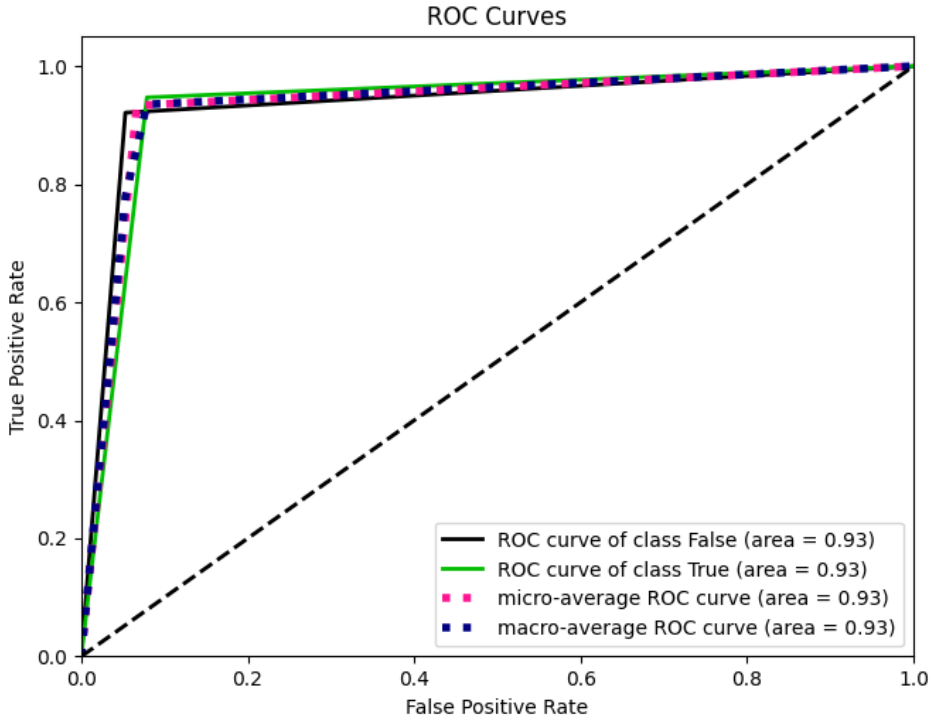

Рисунок 3 ROC-кривая для средне разделимой выборки при использовании дерева

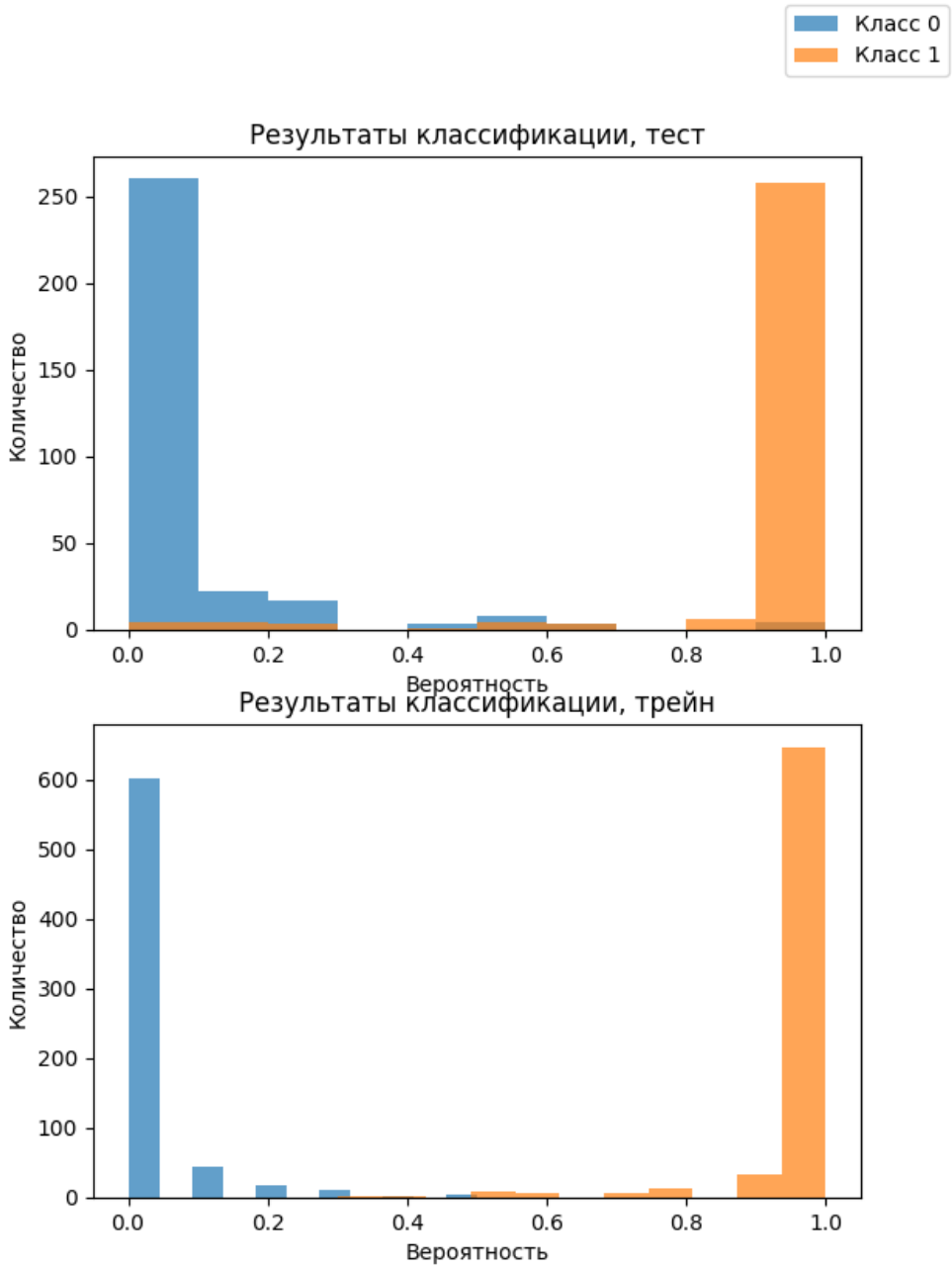

Рисунок 4 Распределение вероятностей для средне разделимой выборки при использовании леса

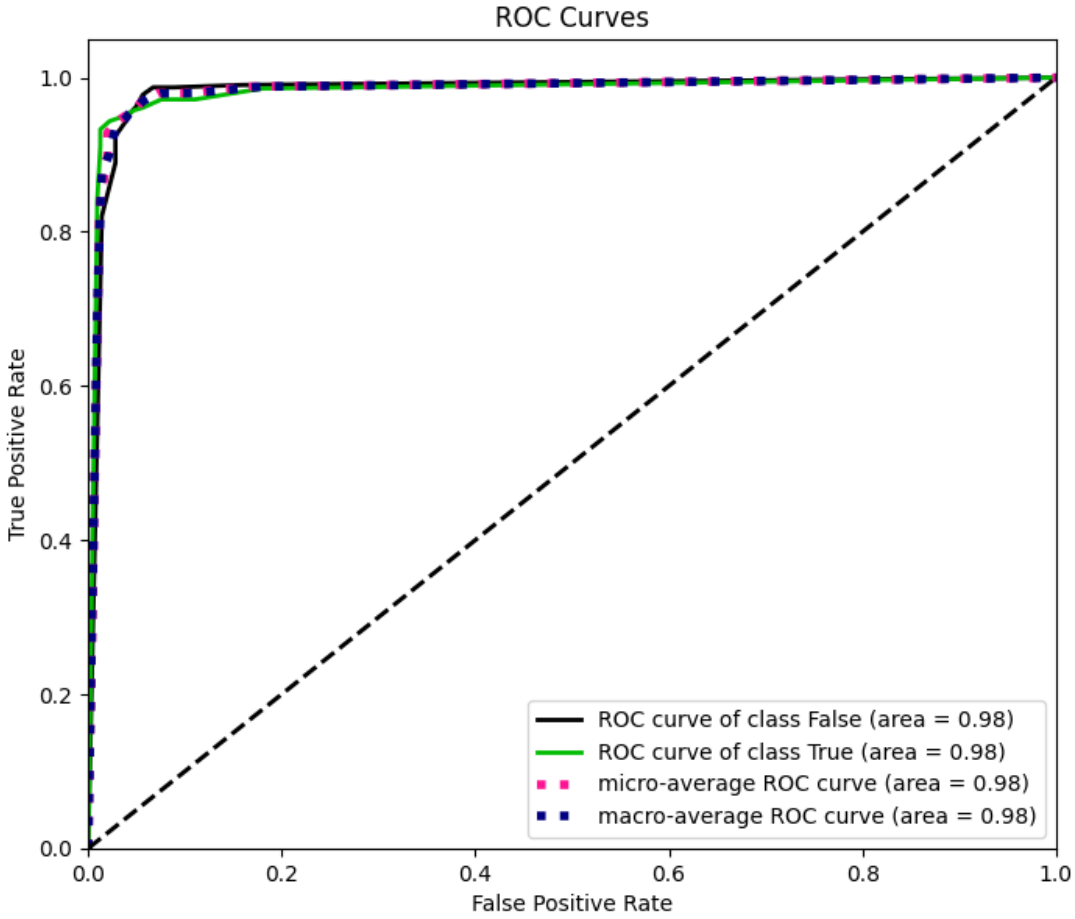

Рисунок 5 ROC-кривая для средне разделимой выборки при использовании леса

		Число объектов	Точность	Чувствительность	Специфичность
Дерево	Train	1400	100	100	100
	Test	600	92.66	94.34	91.16
Лес	Train	1400	99.14	98.32	100
	Test	600	95.66	95.05	96.21

Плохо разделимое нормальное распределение:
mu0 = [0, 1] mu1 = [1, 3] sigma0 = [1, 1] sigma1 = [1, 2]

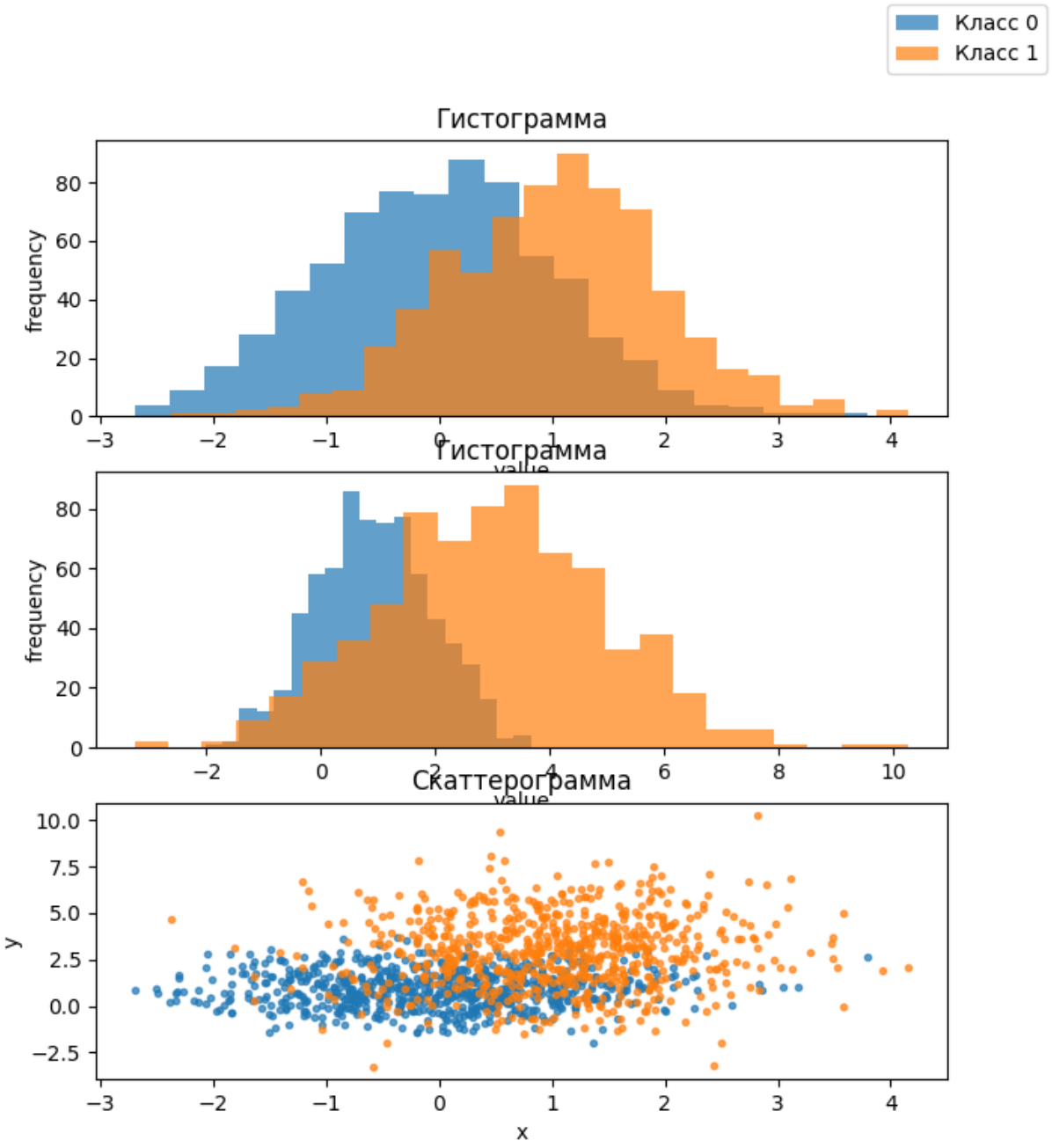

Рисунок 6 Гистограммы и скаттерограммы плохо разделимой выборки

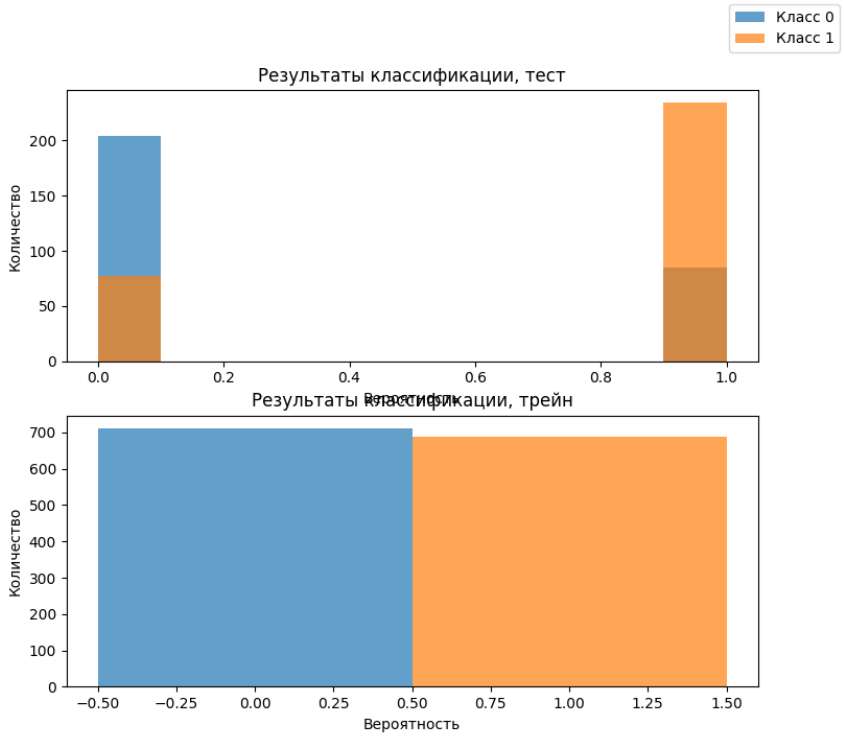

Рисунок 7 Распределение вероятностей для плохо разделимой выборки при использовании дерева

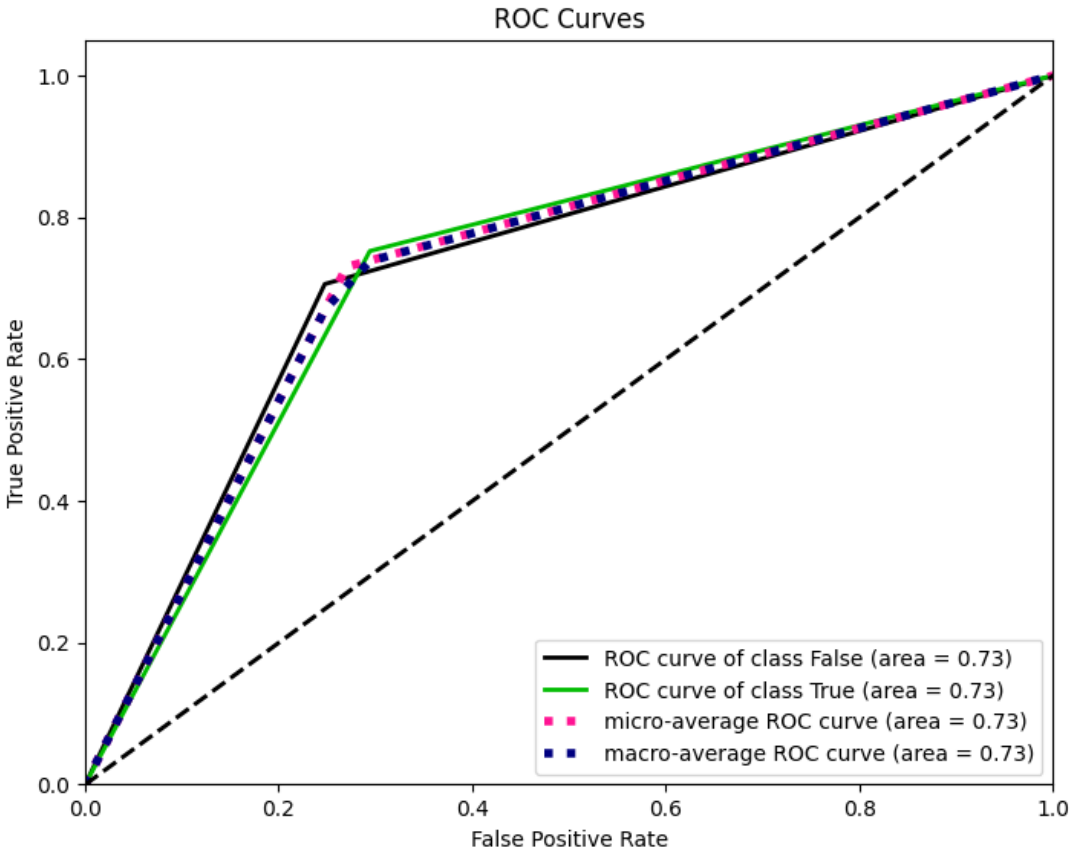

Рисунок 8 ROC-кривая для плохо разделимой выборки при использовании дерева

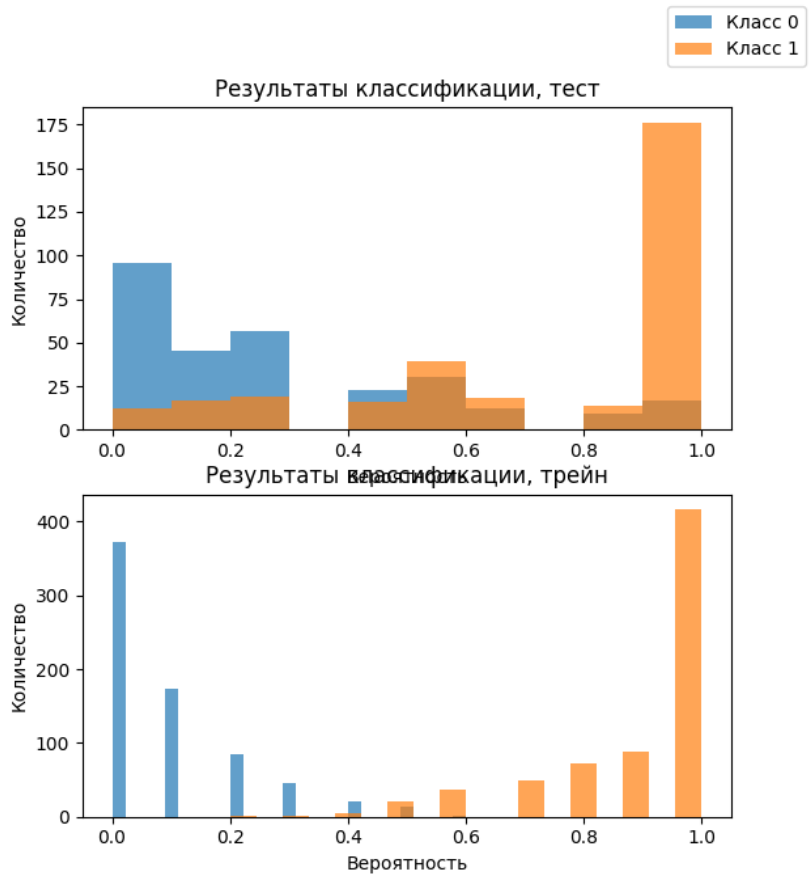

Рисунок 9 Распределение вероятностей для плохо разделимой выборки при использовании леса

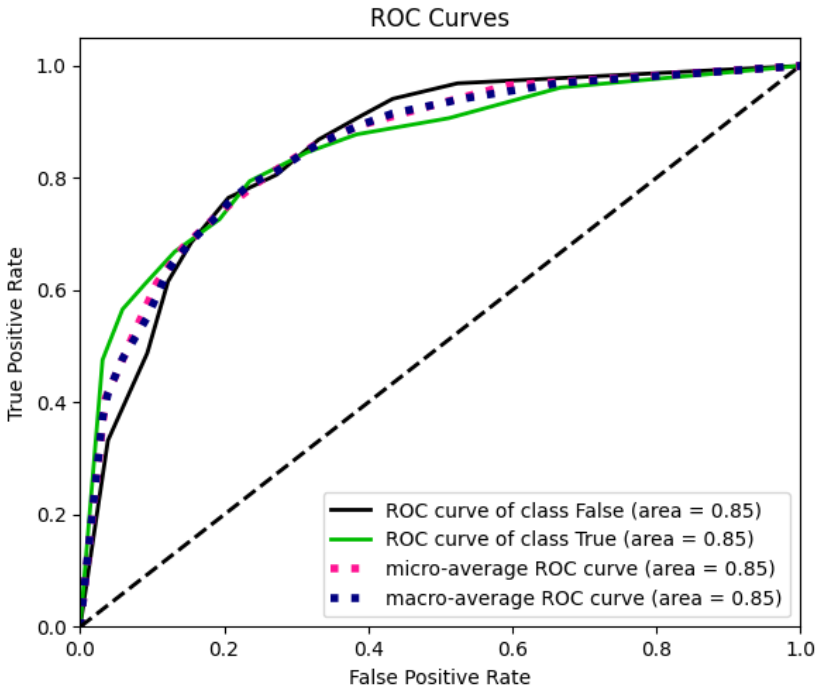

Рисунок 10 ROC-кривая для плохо разделимой выборки при использовании леса

		Число объектов	Точность	Чувствительность	Специфичность
Дерево	Train	1400	1	1	1
	Test	600	73	75.24	70.58
Лес	Train	1400	1	1	1
	Test	600	0.6417	0.7061	0.6202

Нелинейно-пересекаемые выборки

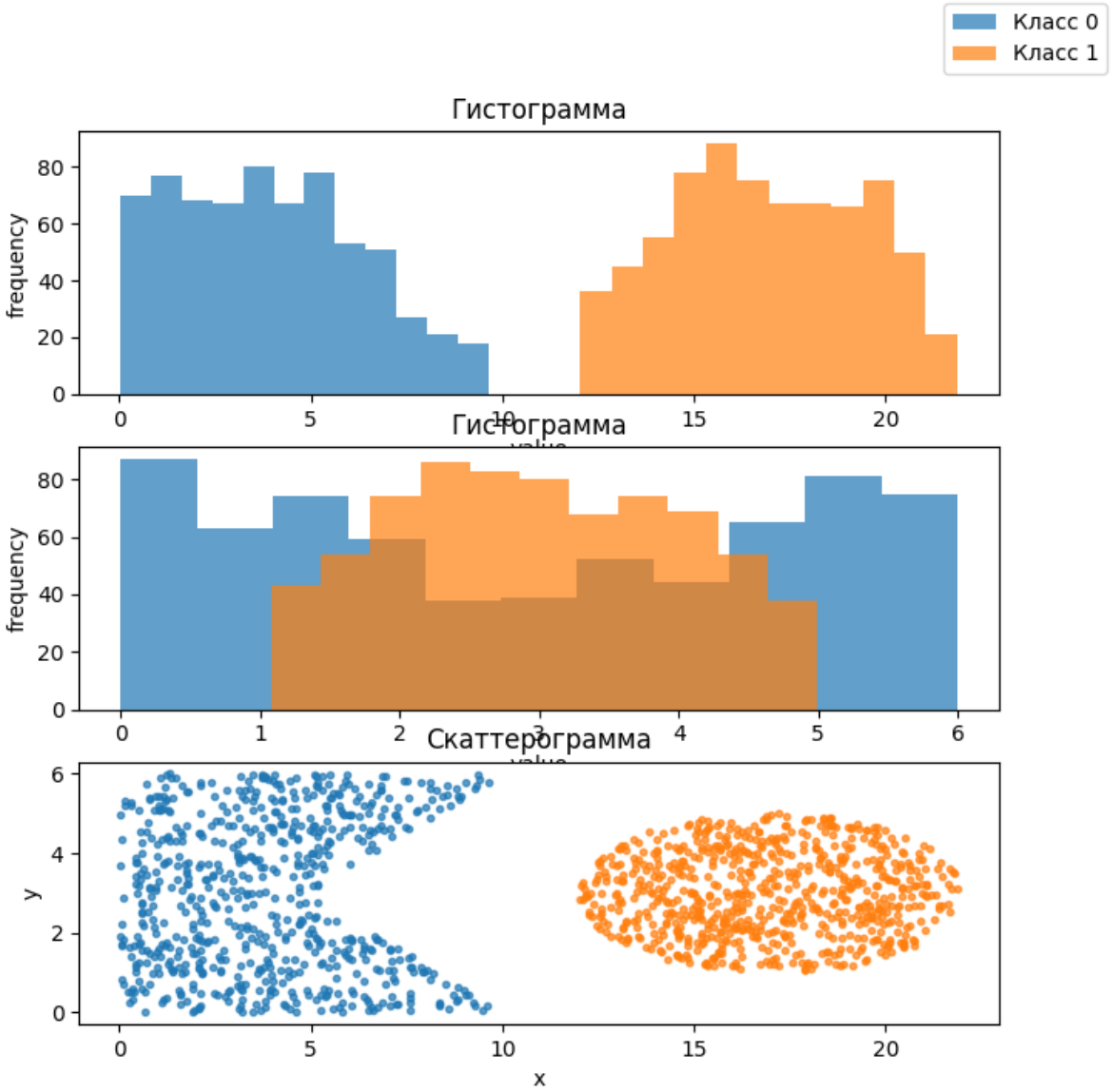

Рисунок 11 Гистограммы и скаттерограммы линейно-непересекаемой выборки

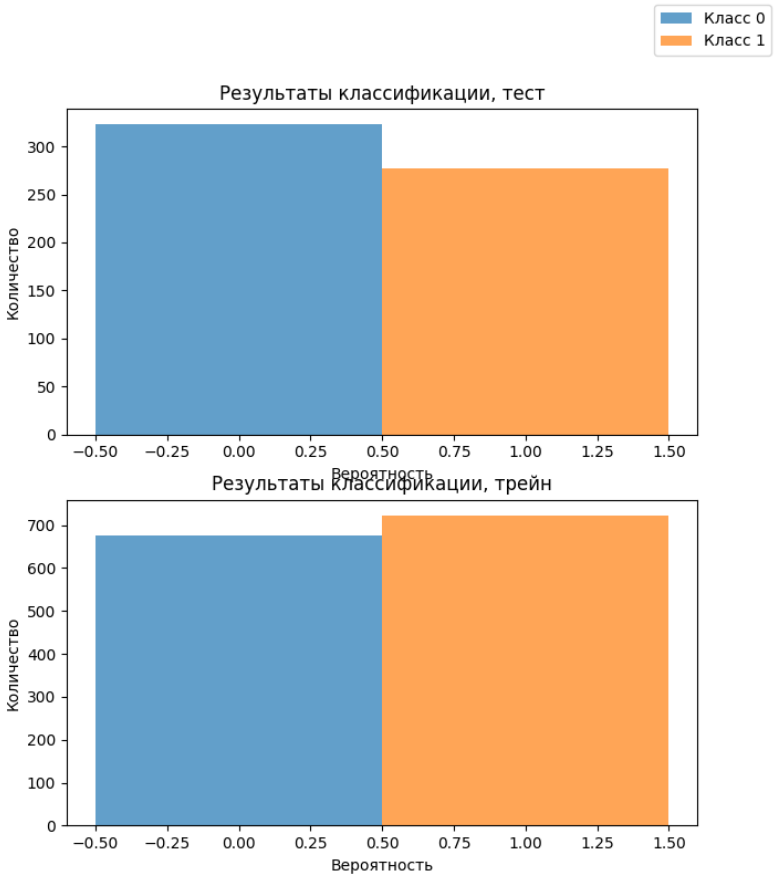

Рисунок 12 Распределение вероятностей для линейно-непересекаемой выборки при использовании дерева

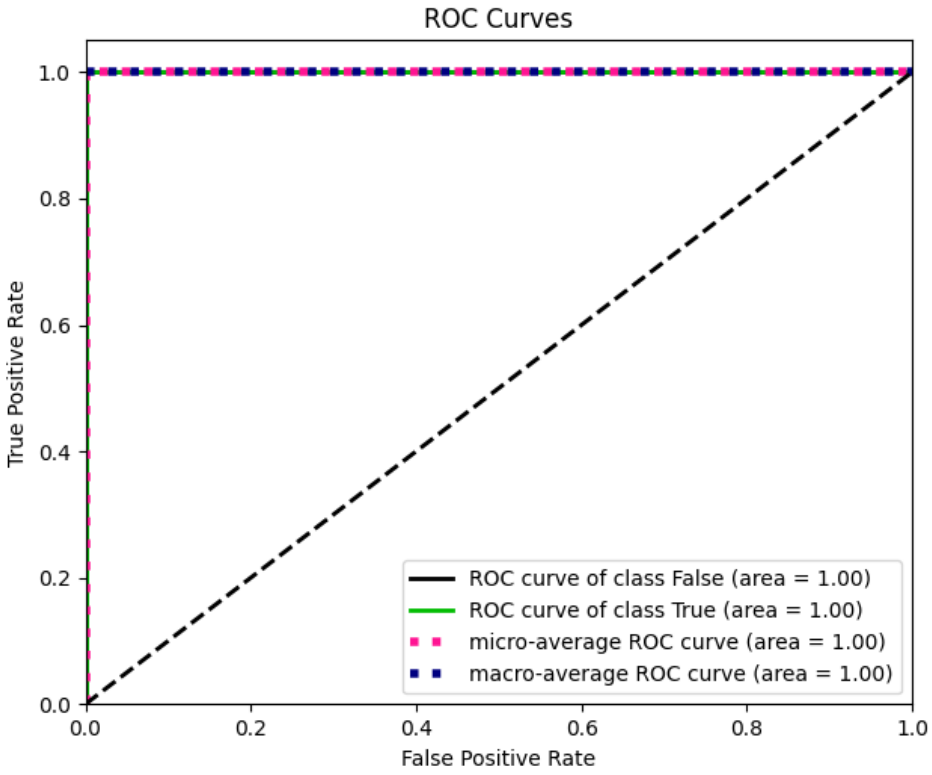

Рисунок 13 ROC-кривая для линейно-непересекаемой выборки при использовании дерева

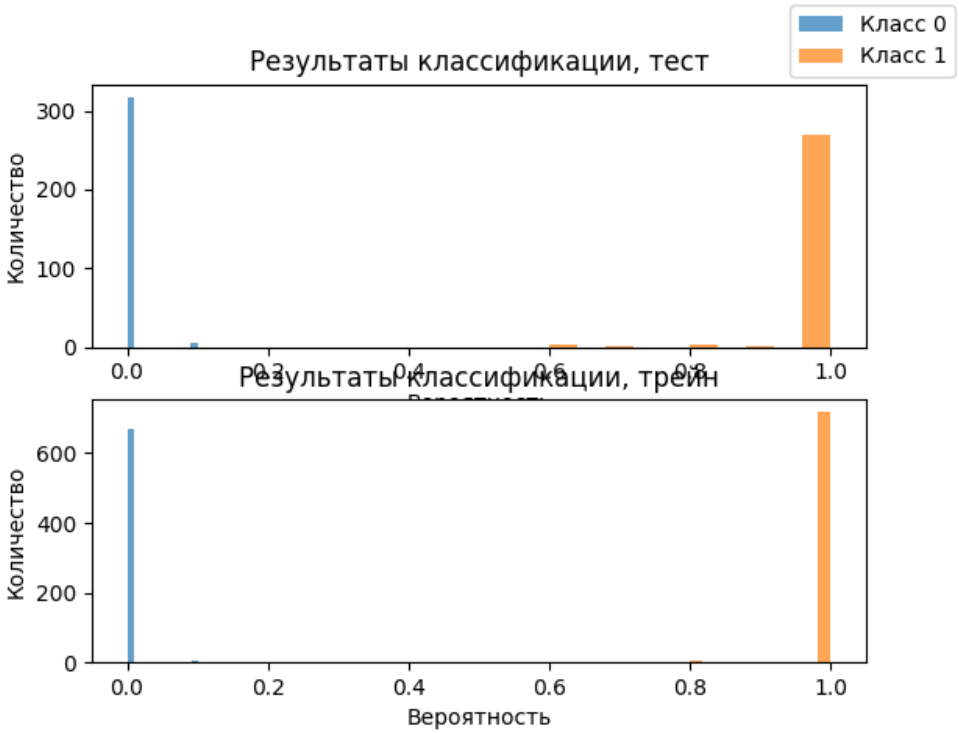

Рисунок 14 Распределение вероятностей для линейно-непересекаемой выборки при использовании леса

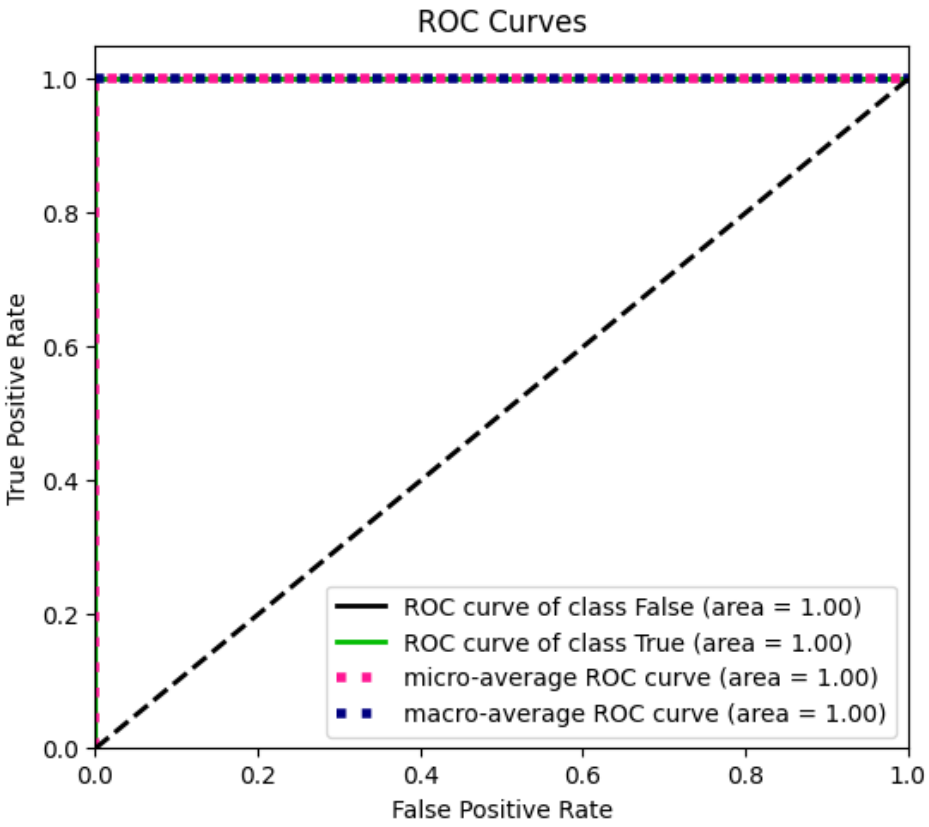

Рисунок 15 ROC-кривая для линейно-непересекаемой выборки при использовании леса

		Число объектов	Точность	Чувствительность	Специфичность
Дерево	Train	1400	1	1	1
	Test	600	1	1	1
Лес	Train	1400	1	1	1
	Test	600	1	1	1

Если модель переобучена, она может показывать высокую точность на обучающей выборке, но низкую точность на тестовой выборке. Исходя из полученных данных, оптимальная глубина дерева, чтобы избежать переобучение модели — около 20. При удалении параметра max_depth, библиотека получает глубину 20-21, при значении 24 результаты находятся на пике, при 25 и выше – результаты снижаются.

Используя цикл for и перебор, было получено такое значение количества деревьев в лесе (в диапазоне от 10 до 300 с шагом в 10), которое даст наибольшее значение площади под кривой на тестовой выборке — 100. Значение площади под кривой - 0.8801502288990621

Вывод:

В ходе лабораторной работы был разработан классификатор на основе дерева принятия решений и исследованы его свойства. В результате подбора гиперпараметров, таких как глубина дерева для модели дерева решений и количество деревьев для случайного леса, удалось выявить оптимальные значения, обеспечивающие наибольшее значение AUC на тестовой выборке. Соответственно, оптимизация гиперпараметров может значительно повысить качество моделей.

Кроме того, анализ результатов по метрикам точности, чувствительности и специфичности позволил оценить эффективность классификации для каждой модели на обучающей и тестовой выборках. Случайный лес продемонстрировал более высокое качество классификации на различных наборах гиперпараметров по сравнению с моделью дерева решений.

Листинг:

```
import numpy as np
from pathlib import Path
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_auc_score
import scikitplot as skplt
from sklearn.ensemble import RandomForestClassifier
from lab1.data_generator import DataGenerator
import matplotlib.pyplot as plt


def calculate_metrics(prediction, answers):
    true_positive_test = np.sum((prediction == 1) & (answers == 1))
    true_negative_test = np.sum((prediction == 0) & (answers == 0))
    false_positive_test = np.sum((prediction == 1) & (answers == 0))
    false_negative_test = np.sum((prediction == 0) & (answers == 1))
    sensivity = true_positive_test / (true_positive_test +
false_negative_test)
    specifity = true_negative_test / (true_negative_test +
false_positive_test)
    accuracy = sum(prediction == answers) / len(answers)

    return sensivity, specifity, accuracy


with_graphs = True
x_train_path = Path("x_train.npy")
x_test_path = Path("x_test.npy")
y_train_path = Path("y_train.npy")
y_test_path = Path("y_test.npy")

path_to_files = [x_train_path,
                 x_test_path,
                 y_train_path,
                 y_test_path
                 ]
mu0 = [0, 1]
mu1 = [1, 3]
sigma0 = [1, 1]
sigma1 = [1, 2]
col = len(mu0)
N = 1000

if not all([e.exists() for e in path_to_files]):

    X, Y, *_ = DataGenerator.norm_dataset((mu0, mu1), (sigma0, sigma1), N)
    # X, Y, *_ = DataGenerator.nonlinear_dataset_N(N)

    train_size = 0.7
    trainCount = round(train_size * N * 2)  # *2 потому что было 2 класса
    Xtrain = X[0:trainCount]
    Xtest = X[trainCount:N * 2 + 1]
    Ytrain = Y[0:trainCount]
    Ytest = Y[trainCount:N * 2 + 1]
    np.save(x_train_path, Xtrain)
    np.save(x_test_path, Xtest)
    np.save(y_train_path, Ytrain)
    np.save(y_test_path, Ytest)
else:
    Xtrain = np.load(x_train_path)
    Xtest = np.load(x_test_path)
    Ytrain = np.load(y_train_path)
```

```
    Ytest = np.load(y_test_path)
max_i = 0
max_auc = 0
for i in range(10, 300, 10):
    """
    Опытным путем было получено что глубина дерева 24 - топ вариант
    """
    # if with_graphs:
    #     X = Xtrain
    #     figure, axis = plt.subplots(3)
    #     axis[0].set_title(f"Гистограмма")
    #     axis[0].hist(X[Ytrain == 0][:, 0], bins='auto', alpha=0.7)
    #     axis[0].hist(X[Ytrain == 1][:, 0], bins='auto', alpha=0.7)
    #     axis[0].set_xlabel("value")
    #     axis[0].set_ylabel("frequency")
    #     axis[1].set_title(f"Гистограмма")
    #     axis[1].hist(X[Ytrain == 0][:, 1], bins='auto', alpha=0.7)
    #     axis[1].hist(X[Ytrain == 1][:, 1], bins='auto', alpha=0.7)
    #     axis[1].set_xlabel("value")
    #     axis[1].set_ylabel("frequency")
    #     axis[2].set_xlabel("x")
    #     axis[2].set_ylabel("y")
    #     axis[2].set_title(f"Скаттерограмма")
    #     axis[2].scatter(X[Ytrain == 0][:, 0], X[Ytrain == 0][:, 1],
marker=".", alpha=0.7)
    #     axis[2].scatter(X[Ytrain == 1][:, 0], X[Ytrain == 1][:, 1],
marker=".", alpha=0.7)
    #
    #     figure.legend(['Класс 0', 'Класс 1'])
    #     plt.show()
    # clf = DecisionTreeClassifier(max_depth=3).fit(Xtrain, Ytrain)
    # print(clf.get_depth())
    #
    clf = RandomForestClassifier(random_state=0, n_estimators=i).fit(Xtrain,
Ytrain)
    pred_test = clf.predict(Xtest)
    # print(pred_test)
    pred_test_proba = clf.predict_proba(Xtest)

    pred_train = clf.predict(Xtrain)
    pred_train_proba = clf.predict_proba(Xtrain)
    # print(pred test proba)

    acc_train = clf.score(Xtrain, Ytrain)
    # print(acc_train)
    acc_test = clf.score(Xtest, Ytest)
    # print(acc test)
    # acc_test = sum(pred_test == Ytest) / len(Ytest)
    # print(acc_test)
    # from sklearn.calibration import calibration_curve
    # print(Ytest.shape)
    # y_means, proba_means = calibration_curve(Ytest, pred_test_proba,
n_bins=10)

    figure, axis = plt.subplots(2)
    # print(pred train proba)
    # print(Ytrain)
    axis[0].hist(pred_test_proba[~Ytest, 1], bins='auto', alpha=0.7)

    axis[0].hist(pred_test_proba[Ytest, 1], bins='auto', alpha=0.7)
    axis[0].set_xlabel("Вероятность")
    axis[0].set_ylabel("Количество")
    axis[0].set_title("Результаты классификации, тест")
```

```
    # print(pred_train_proba[Ytrain, 1])
    # print(pred train proba[~Ytrain, 1])
    axis[1].hist(pred_train_proba[~Ytrain, 1], bins='auto', alpha=0.7)
    axis[1].hist(pred_train_proba[Ytrain, 1], bins='auto', alpha=0.7)
    axis[1].set_xlabel("Вероятность")
    axis[1].set_ylabel("Количество")
    axis[1].set_title("Результаты классификации, трейн")
    figure.legend(['Класс 0', 'Класс 1'])
    # plt.show()
    # print(Ytest.size)
    # print(calculate metrics(pred test, Ytest))
    # print(Ytrain.size)
    # print(calculate_metrics(pred_train, Ytrain))
    #
    # fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=800)
    # tree.plot tree(clf.estimators [0],
    #                filled = True)
    # fig.savefig('rf_individualtree.png')
    # fig.show()
    skplt.metrics.plot_roc_curve(Ytest, pred_test_proba, figsize=(10, 10))
    # plt.show()

    # Расчет площади под кривой
    AUC = roc_auc_score(Ytest, pred_test_proba[:, 1])
    print(i)
    print("AUC tree:" + str(AUC))
    if AUC > max_auc:
        max_auc = AUC
        max_i = i
print(max_i)
print(max_auc)
"""
600
(np.float64(0.7185430463576159), np.float64(0.7181208053691275),
np.float64(0.7183333333333334))
1400
(np.float64(1.0), np.float64(1.0), np.float64(1.0))

Дерево идеально классифицирует данные, на которых было обучено, при этом на
тестовых данных метрики падают до 71%


600
(np.float64(0.7384105960264901), np.float64(0.8288590604026845),
np.float64(0.7833333333333333))
1400
(np.float64(1.0), np.float64(1.0), np.float64(1.0))

Для леса результаты работы на трейн выборке не изменились. А вот для тестовой
выборки стали немного лучше чем у дерева.


AUC tree:0.852221209831548
"""
```