Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине «Прикладные интеллектуальные системы и экспертные системы»

Бинарная классификация фактографических данных

Студент Коровайцева А.В.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.

Доцент

Липецк 2023 г.

Цель работы

Получить практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook. Научиться загружать данные, обучать классификаторы и проводить классификацию. Научиться оценивать точность полученных моделей.

Задание кафедры

1) В среде Jupiter Notebook создать новый ноутбук (Notebook)

2) Импортировать необходимые для работы библиотеки и модули

3) Загрузить данные в соответствие с вариантом

4) Вывести первые 15 элементов выборки (координаты точек и метки класса)

5) Отобразить на графике сгенерированную выборку. Объекты разных классов должны иметь разные цвета.

6) Разбить данные на обучающую (train) и тестовую (test) выборки в пропорции 75% - 25% соответственно.

7) Отобразить на графике обучающую и тестовую выборки. Объекты разных классов должны иметь разные цвета.

8) Реализовать модели классификаторов, обучить их на обучающем множестве. Применить модели на тестовой выборке, вывести результаты классификации:

9) Истинные и предсказанные метки классов

10) Матрицу ошибок (confusion matrix)

11) Значения полноты, точности, f1-меры и аккуратности

12) Значение площади под кривой ошибок (AUC ROC)

13) Отобразить на графике область принятия решений по каждому классу

14) В качестве методов классификации использовать:

15) Метод к-ближайших соседей (n_neighbors = {1, 3, 5, 9})

16) Наивный байесовский метод

17) Случайный лес (n_estimators = {5, 10, 15, 20, 50})

18) По каждому пункту работы занести в отчет программный код и результат вывода.

19) По результатам п.8 занести в отчет таблицу с результатами классификации всеми методами и выводы о наиболее подходящем методе классификации ваших данных.

20) Изучить, как изменится качество классификации, если на тестовую часть выделить 10% выборки, 35% выборки. Для этого повторить п.п. 6 – 10.

Вариант №8

Вид классов: moons

Random_state: 15

noise: 0.2

Ход работы

Подготовка данных

Для генерации данных воспользуемся функцией make_moons из пакета sklearn.datasets. Результат генерации данных и вывод первых 15 значений представлен на рисунке 1.

```
X, y = make_moons(n_samples=1000, shuffle=True, noise=0.2, random_state=15)
```

```
print('Координаты точек: ')
print(X[:15])
print('Метки класса: ')
print(y[:15])
```

```
Координаты точек:
[[ 1.7271961  -0.39285757]
 [-0.91801735  0.81910014]
 [-0.91532959 -0.05460812]
 [ 0.14537408  0.2064726 ]
 [ 0.95552152  0.20921022]
 [ 1.85825106 -0.35738814]
 [ 0.0761107   0.90867532]
 [-0.66311624  1.08115035]
 [ 0.13798809  0.98723143]
 [ 1.85704117  0.34111441]
 [ 0.17994761  0.15344022]
 [ 0.91666297  0.49152481]
 [ 1.25585707 -0.50035284]
 [ 1.11412853 -0.36151518]
 [-0.41601705  0.82276341]]
Метки класса:
[1 0 0 1 0 1 0 0 0 1 1 0 1 1 0]
```

Рисунок 1 – Генерация данных и вывод первых 15-ти значений

Отобразим на графике сгенерированную выборку с выделением классов разными цветами. Для этого воспользуемся функцией scatter из библиотеки matplotlib.pyplot. Результат визуализации представлен на рисунке 2.

```
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```

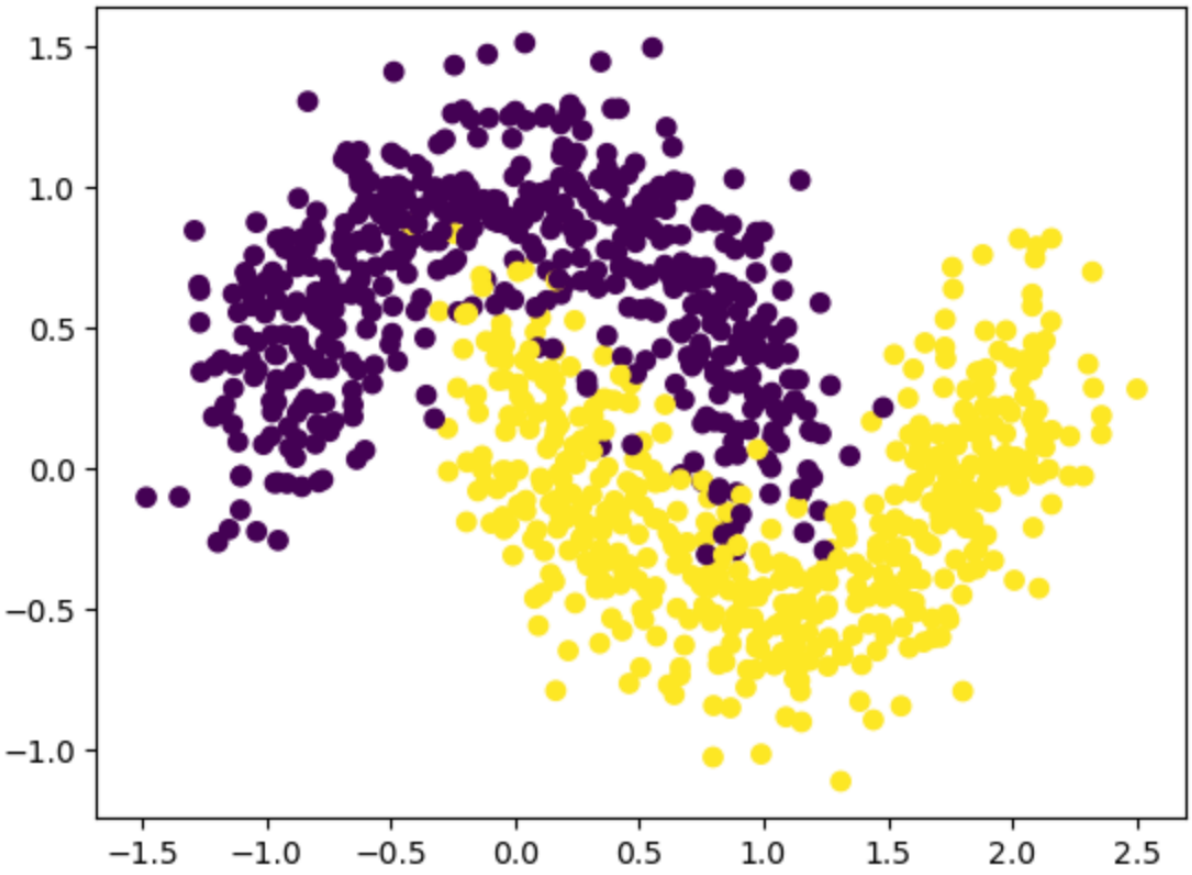

Рисунок 2 – Визуализация выборки

Разделим данных на обучающую и тестовую выборку. Для этого воспользуемся функцией train_test_split из пакета sklearn.model_selection. Скрипт для разделения данных представлен на рисунке 3. Результат разбиения выборки на тестовую и обучающую с последующей их визуализацией представлены на рисунке 4 и 5.

Разбитие выборки на обучающее и тестовое множество (75/25)

```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=77)
```

Рисунок 3 – Разделение выборки на обучающее и тестовое множество

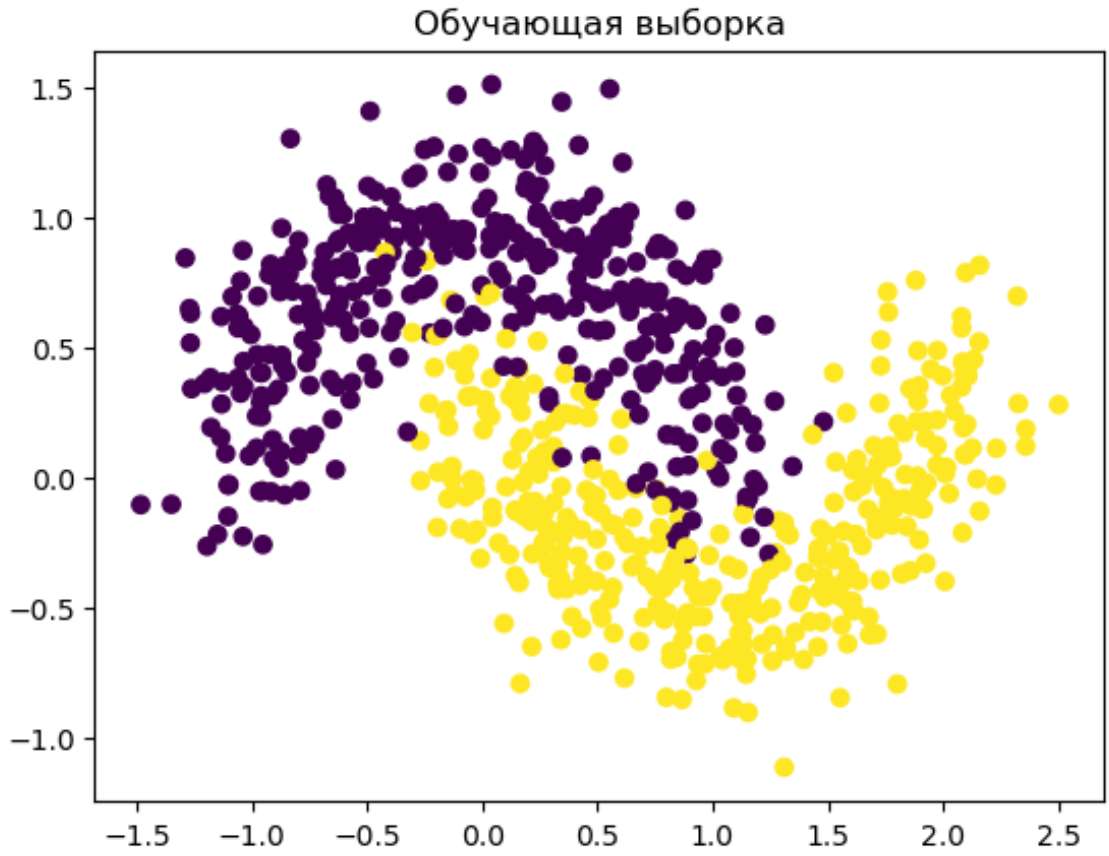

Рисунок 4 – Обучающая выборка

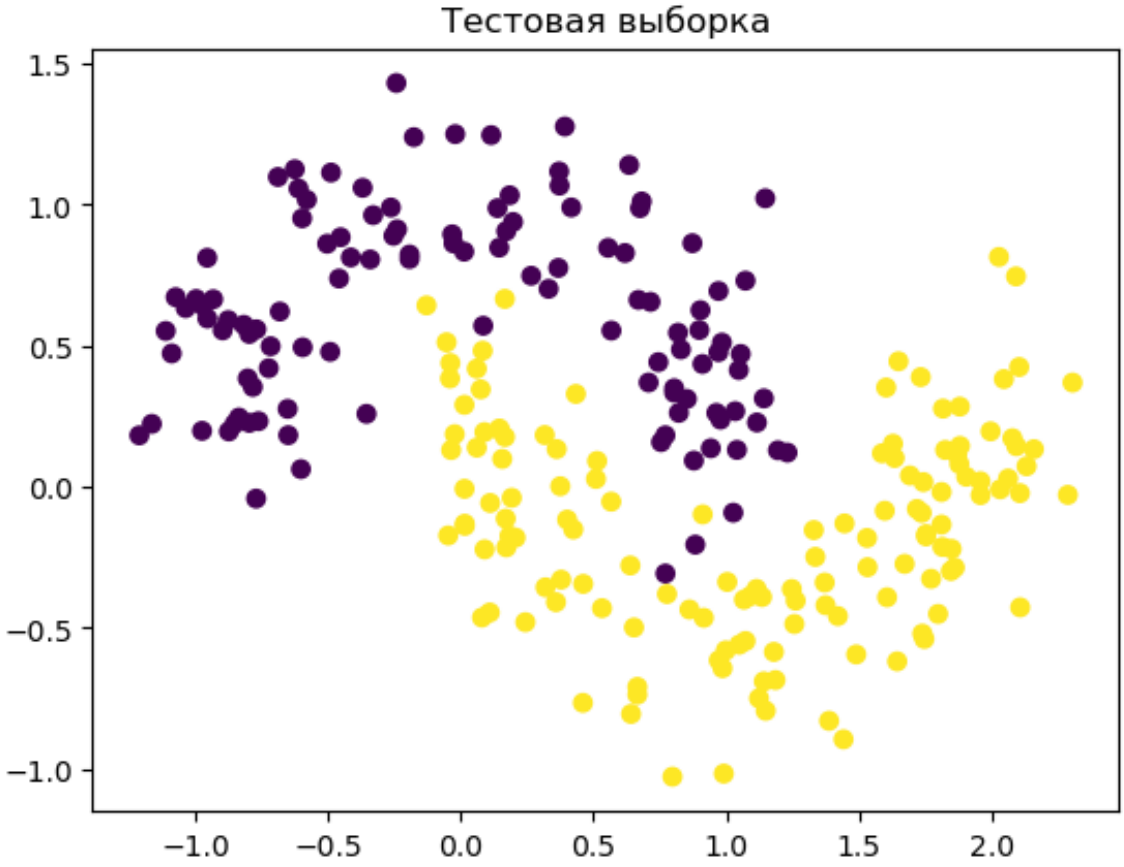

Рисунок 5 – Тестовая выборка

Классификация с помощью метода к-ближайших соседей

Использование параметра n_neighbors = 1. Составленный код для использования данного классификатора с данным параметром представлен на рисунке 6. Полученная информация о точности классификации при использовании данного метода представлена на рисунке 7.

```
knn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (1)', y_test, prediction)
```

Рисунок 6 – Код для классификатора с помощью метода к-ближайших соседей с параметром n_neighbors = 1

```
Метод классификации: ближайшие соседи (1)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1 1 1 1 0
 1 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[111   3]
 [  6 130]]

Точность классификации: 0.964

Полнота:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       114
           1       0.98      0.96      0.97       136

    accuracy                           0.96       250
   macro avg       0.96      0.96      0.96       250
weighted avg       0.96      0.96      0.96       250


Площадь под кривой: 0.9647832817337462
```

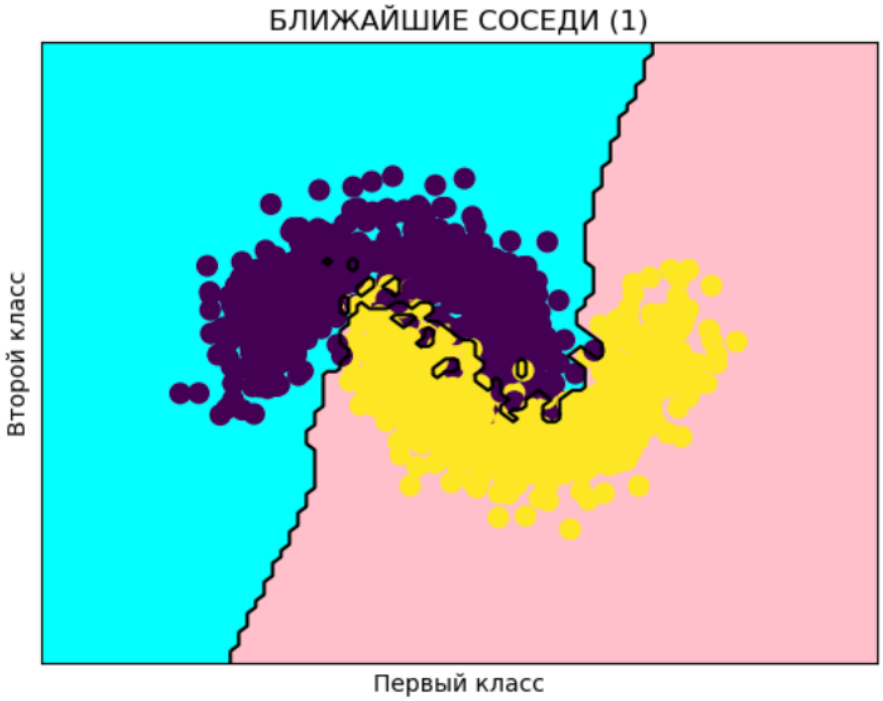

Рисунок 7 – Результат классификации с помощью метода к-ближайших соседей с параметром n_neighbors = 1

Запустим классификацию с использованием параметра n_neighbors = 3, 5 и 9. Результаты классификации представлены на рисунках 8, 9 и 10 соответственно.

```
Метод классификации: ближайшие соседи (3)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 1 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[111   3]
 [  5 131]]

Точность классификации: 0.968

Полнота:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       114
           1       0.98      0.96      0.97       136

    accuracy                           0.97       250
   macro avg       0.97      0.97      0.97       250
weighted avg       0.97      0.97      0.97       250


Площадь под кривой: 0.9684597523219816
```

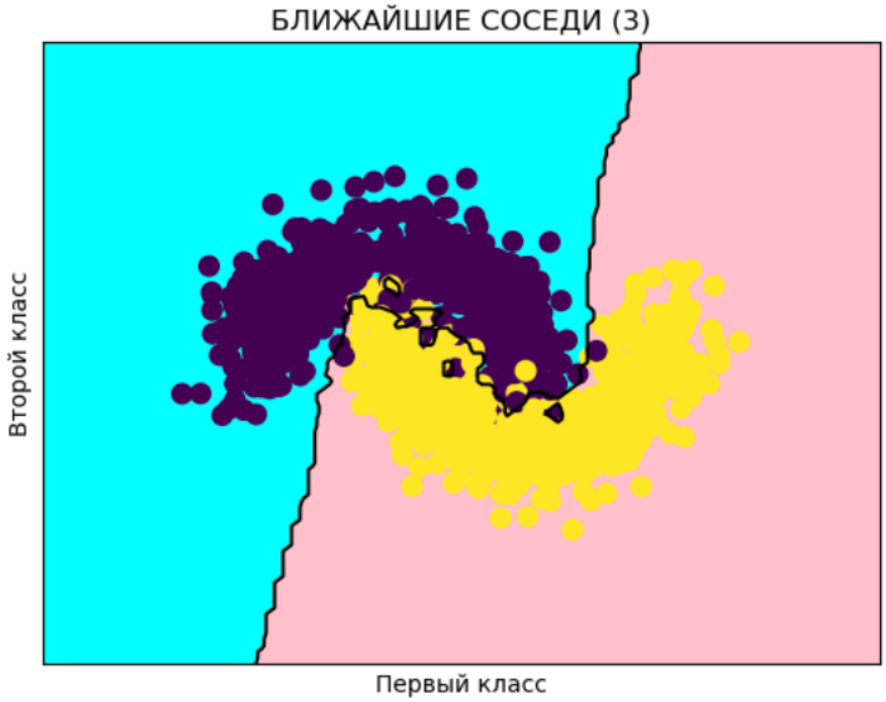

Рисунок 8 – Результат классификации с помощью метода к-ближайших соседей с параметром n_neighbors = 3

```
Метод классификации: ближайшие соседи (5)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[112   2]
 [  3 133]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       114
           1       0.99      0.98      0.98       136

    accuracy                           0.98       250
   macro avg       0.98      0.98      0.98       250
weighted avg       0.98      0.98      0.98       250


Площадь под кривой: 0.9801986584107326
```

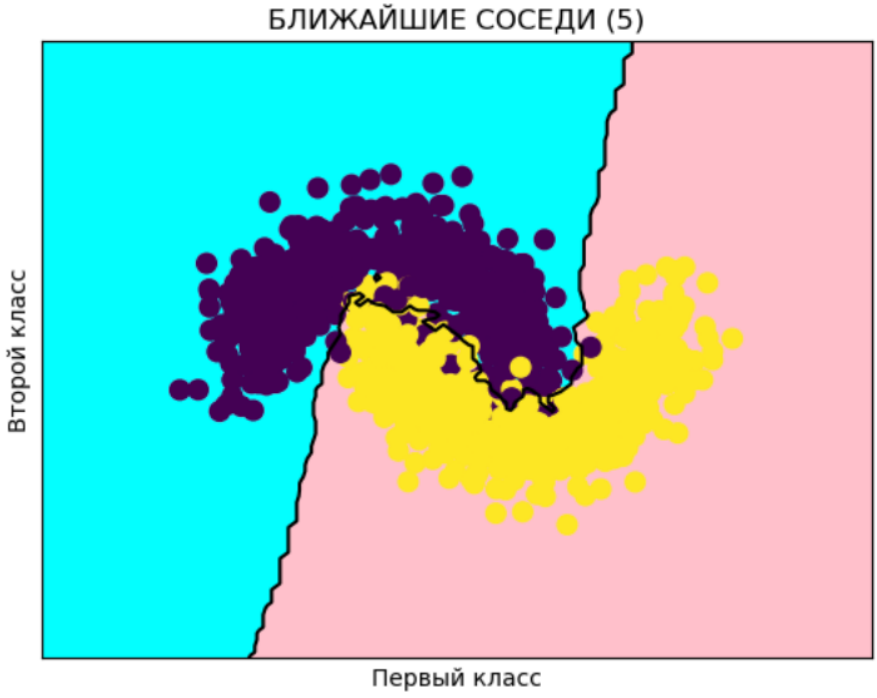

Рисунок 9 – Результат классификации с помощью метода к-ближайших соседей с параметром n_neighbors = 5

```
Метод классификации: ближайшие соседи (9)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1 1
 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 1 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[111   3]
 [  3 133]]

Точность классификации: 0.976

Полнота:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       114
           1       0.98      0.98      0.98       136

    accuracy                           0.98       250
   macro avg       0.98      0.98      0.98       250
weighted avg       0.98      0.98      0.98       250


Площадь под кривой: 0.975812693498452
```

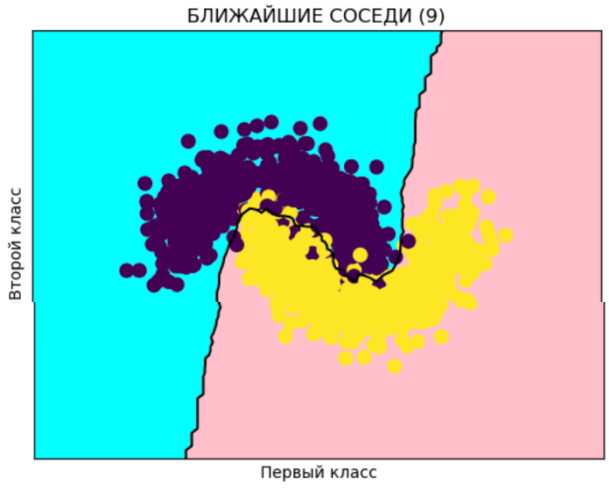

Рисунок 10 – Результат классификации с помощью метода к-ближайших соседей с параметром n_neighbors = 9

Классификация с помощью наивного байесовского классификатора

Составленный код для использования данного классификатора представлен на рисунке 11. Полученная информация о точности классификации при использовании данного метода представлена на рисунке 12.

```
:
1  from sklearn.naive_bayes import GaussianNB
2
3  nb = GaussianNB()
4
5  # Обучаем модель данных
6  nb.fit(X_train, y_train)
7
8  # Оцениваем качество модели
9  prediction = nb.predict(X_test)
10
11 # Выводим сводную информацию
12 show_info(nb, 'Наивный байесовский классификатор', y_test, prediction)
```

Рисунок 11 – Код для классификатора с помощью наивного байесовского классификатора

```
Метод классификации: Наивный байесовский классификатор

Предсказанные и реальные значения:
[1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 0
 0 0 0 0 0 1 1 1 0 1 0 1 1 1 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1
 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 0 1 1 1 1 0 0 1 0 1 1
 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 0 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 0
 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0 1
 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 0 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[ 95  19]
 [ 19 117]]

Точность классификации: 0.848

Полнота:
              precision    recall  f1-score   support

           0       0.83      0.83      0.83       114
           1       0.86      0.86      0.86       136

    accuracy                           0.85       250
   macro avg       0.85      0.85      0.85       250
weighted avg       0.85      0.85      0.85       250


Площадь под кривой: 0.8468137254901961
```

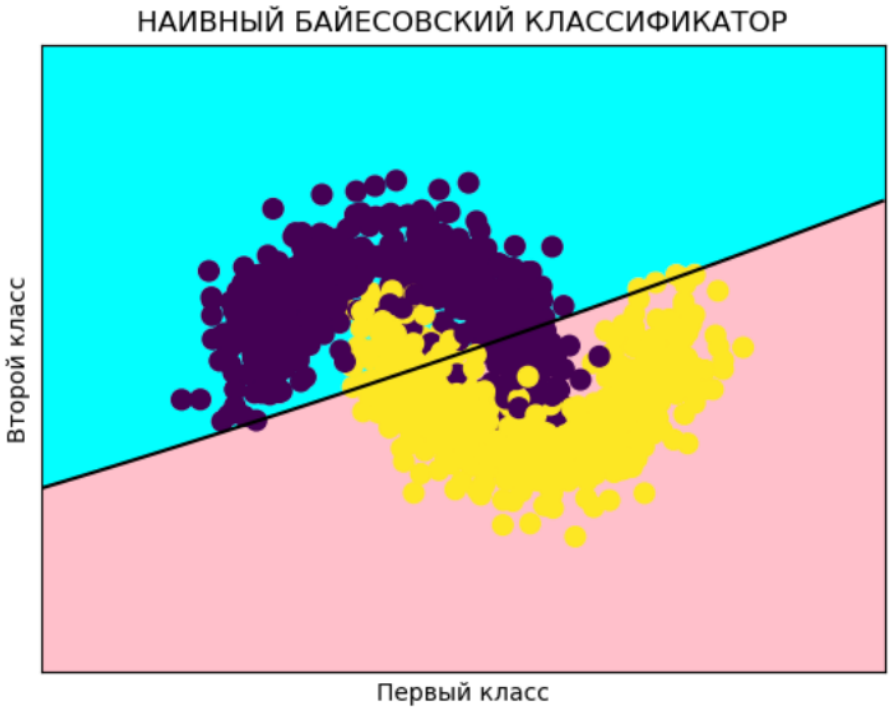

Рисунок 12 – Результат классификации с помощью наивного байесовского классификатора

Классификация с помощью случайного леса

Использование параметра n_estimators = 5. Составленный код для использования данного классификатора с данным параметром представлен на рисунке 13. Полученная информация о точности классификации при использовании данного метода представлена на рисунке 14.

```
rfc = RandomForestClassifier(n_estimators=5)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (5)', y_test, prediction)
```

Рисунок 13 – Код для классификатора с помощью случайного леса с параметром n_ estimators = 5

```
Метод классификации: случайный лес (5)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[113   1]
 [  5 131]]

Точность классификации: 0.976

Полнота:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       114
           1       0.99      0.96      0.98       136

    accuracy                           0.98       250
   macro avg       0.98      0.98      0.98       250
weighted avg       0.98      0.98      0.98       250


Площадь под кривой: 0.9772316821465429
```

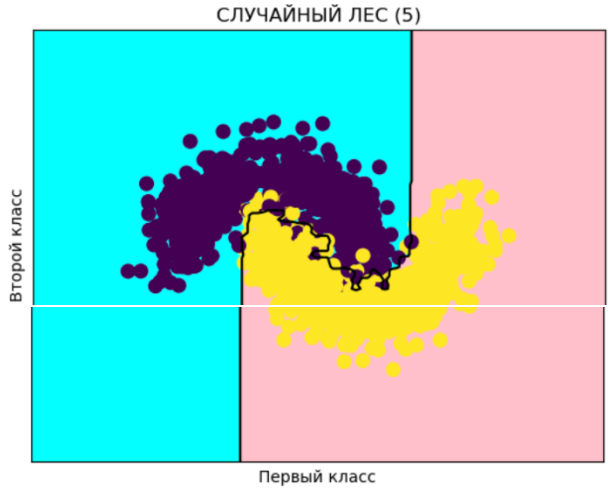

Рисунок 14 – Результат классификации с помощью случайного леса с параметром n_ estimators = 5

Запустим классификацию с использованием параметра n_ estimators = 10, 15, 20 и 50. Результаты классификации представлены на рисунках 15, 16, 17 и 18 соответственно.

```
Метод классификации: случайный лес (10)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[113   1]
 [  4 132]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       114
           1       0.99      0.97      0.98       136

    accuracy                           0.98       250
   macro avg       0.98      0.98      0.98       250
weighted avg       0.98      0.98      0.98       250


Площадь под кривой: 0.9809081527347782
```

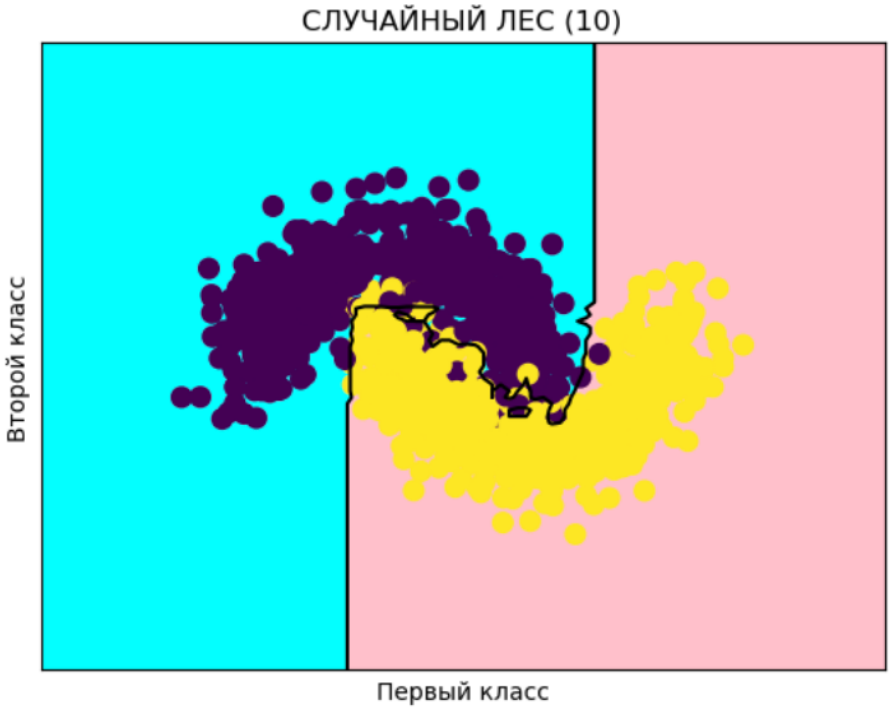

Рисунок 15 – Результат классификации с помощью случайного леса с параметром n_ estimators = 10

```
Метод классификации: случайный лес (15)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[113   1]
 [  4 132]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       114
           1       0.99      0.97      0.98       136

    accuracy                           0.98       250
   macro avg       0.98      0.98      0.98       250
weighted avg       0.98      0.98      0.98       250


Площадь под кривой: 0.9809081527347782
```

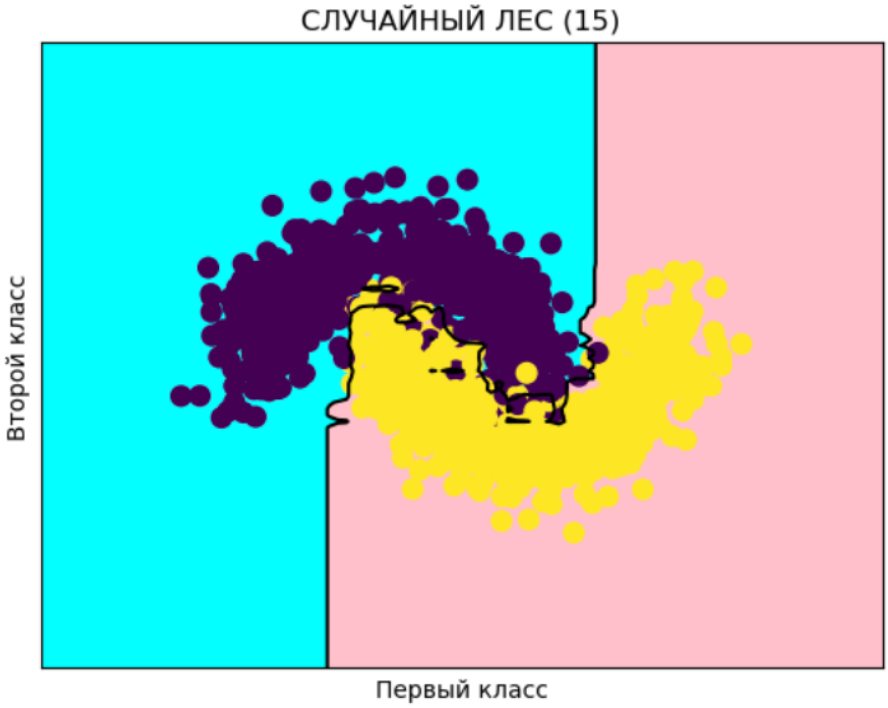

Рисунок 16 – Результат классификации с помощью случайного леса с параметром n_ estimators = 15

```
Метод классификации: случайный лес (20)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[113   1]
 [  5 131]]

Точность классификации: 0.976

Полнота:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       114
           1       0.99      0.96      0.98       136

    accuracy                           0.98       250
   macro avg       0.98      0.98      0.98       250
weighted avg       0.98      0.98      0.98       250


Площадь под кривой: 0.9772316821465429
```

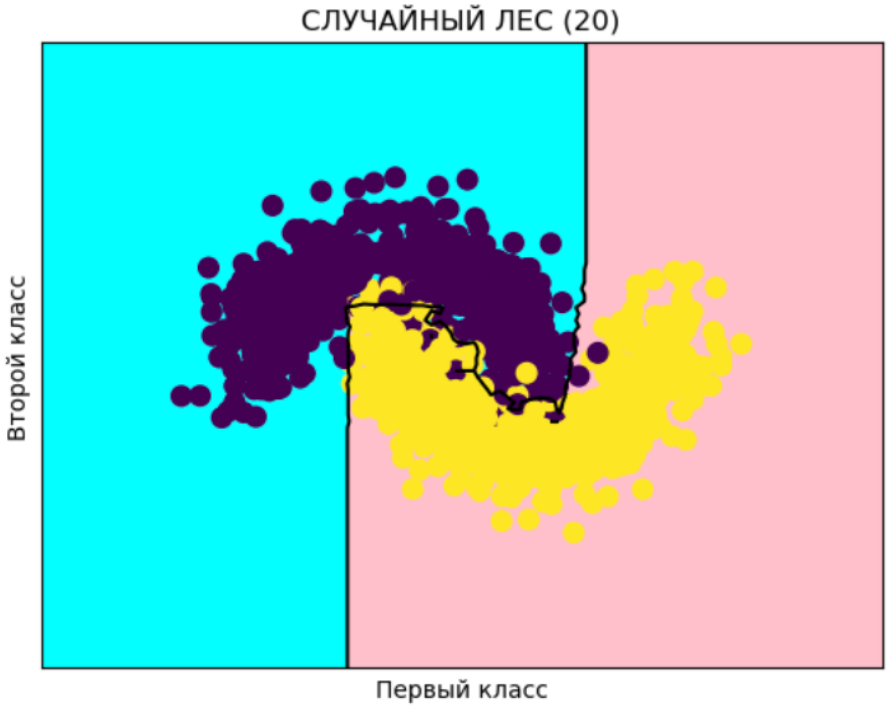

Рисунок 17 – Результат классификации с помощью случайного леса с параметром n_ estimators = 20

```
Метод классификации: случайный лес (50)

Предсказанные и реальные значения:
[0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]
[0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0
 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 1 1
 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 1 0 1 1
 1 0 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 1 0 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0
 1 1 1 0 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1 1 0 1
 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 1 1 0 0 0 1 0 1 1 0]

Матрица неточностей
[[113   1]
 [  4 132]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       114
           1       0.99      0.97      0.98       136

    accuracy                           0.98       250
   macro avg       0.98      0.98      0.98       250
weighted avg       0.98      0.98      0.98       250


Площадь под кривой: 0.9809081527347782
```

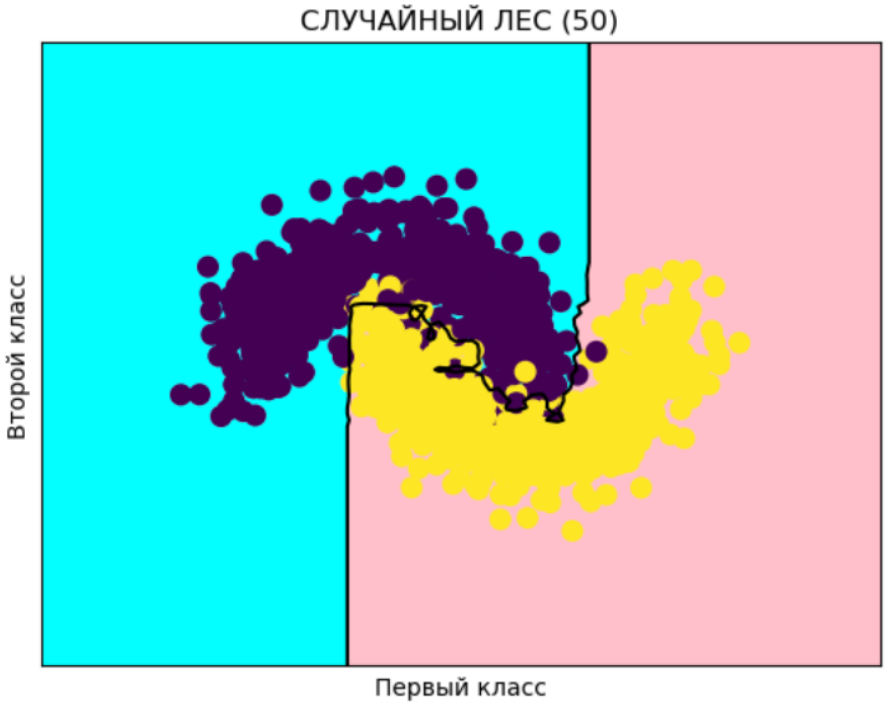

Рисунок 18 – Результат классификации с помощью случайного леса с параметром n_ estimators = 50

Анализ результатов

Сведем полученные данные в таблицу 1 и сделаем вывод.

Таблица 1 – Результат классификации по методам про 75% обучающей выборки

Метод (параметры)	Точность	Площадь под кривой
Метод к-ближайших соседей (n_neighbors = 1)	0,964	0,965
Метод к-ближайших соседей (n_neighbors = 3)	0,968	0,968
Метод к-ближайших соседей (n_neighbors = 5)	0,98	0,98
Метод к-ближайших соседей (n_neighbors = 9)	0,976	0,976
Наивный байесовский классификатор	0,848	0,847
Случайный лес (n_estimators = 5)	0,976	0,977
Случайный лес (n_estimators = 10)	0,98	0,981
Случайный лес (n_estimators = 15)	0,98	0,981
Случайный лес (n_estimators = 20)	0,966	0,977
Случайный лес (n_estimators = 50)	0,98	0,981

Исходя из таблицы 1, можно сделать вывод о том, что лучше всего себя показал метод к-ближайших соседей (n_neighbors = 5) и случайный лес (n_estimators = 10, n_estimators = 15 и n_estimators = 50), хуже всего – наивный байесовский классификатор.

Рассмотрим случай уменьшения тестовой выборки. Установим, что тестовая выборка составляет 10% и построим графики визуализации обучающей и тестовой выборки, данные графики представлены на рисунках 19 и 20 соответственно.

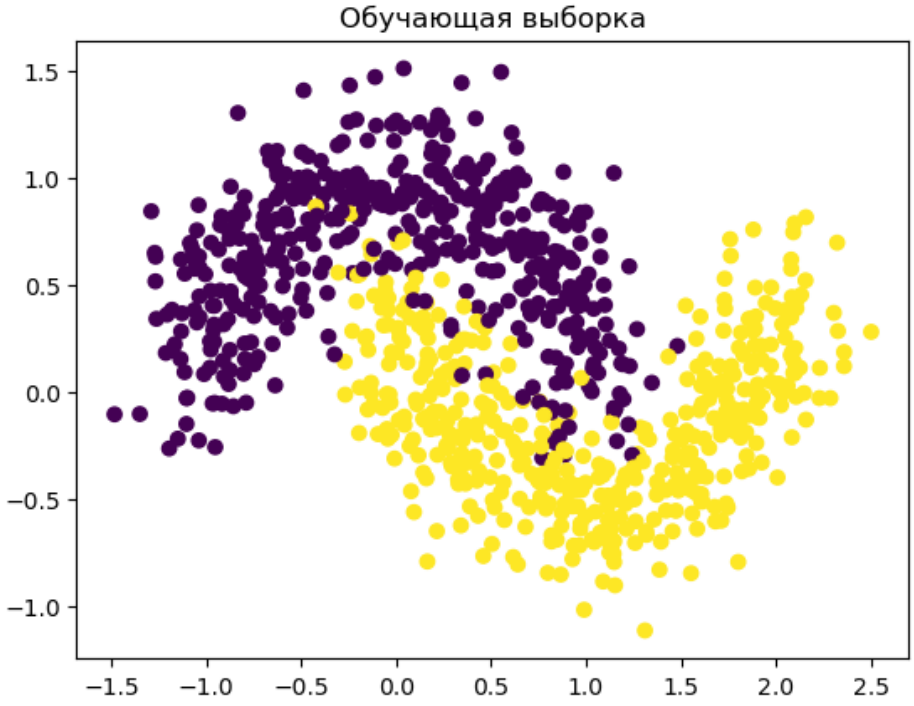

Рисунок 19 – Обучающая выборка при 90% от общего размера

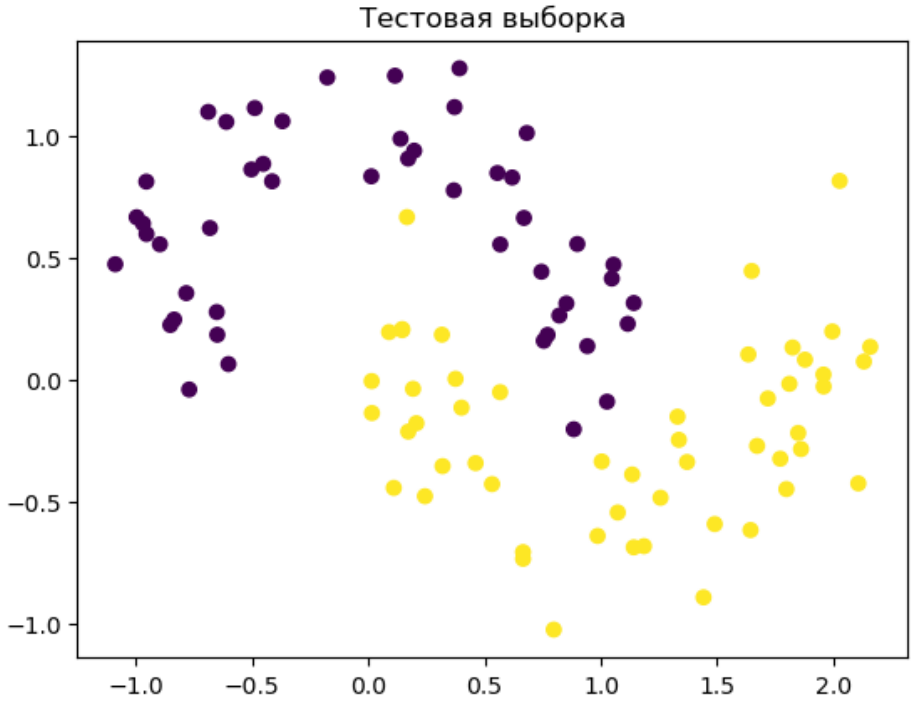

Рисунок 20 – Тестовая выборка при 10% от общего размера

Сведем полученные данные в таблицу 2 и сделаем вывод.

Таблица 2 – Результат классификации по методам про 90% обучающей выборки

Метод (параметры)	Точность	Площадь под кривой
Метод к-ближайших соседей (n_neighbors = 1)	0,99	0,99
Метод к-ближайших соседей (n_neighbors = 3)	0,98	0,98
Метод к-ближайших соседей (n_neighbors = 5)	0,98	0,98
Метод к-ближайших соседей (n_neighbors = 9)	0,98	0,98
Наивный байесовский классификатор	0,84	0,838
Случайный лес (n_estimators = 5)	0,98	0,98
Случайный лес (n_estimators = 10)	0,99	0,99
Случайный лес (n_estimators = 15)	0,99	0,99
Случайный лес (n_estimators = 20)	0,99	0,99
Случайный лес (n_estimators = 50)	0,98	0,98

Исходя из таблицы 2, можно сделать вывод о том, что лучше всего себя показал метод к-ближайших соседей (n_neighbors = 1) и случайный лес (n_estimators = 10, n_estimators = 15 и n_estimators = 50), хуже всего – наивный байесовский классификатор.

Рассмотрим случай уменьшения тестовой выборки. Установим, что тестовая выборка составляет 35% и построим графики визуализации обучающей и тестовой выборки, данные графики представлены на рисунках 21 и 22 соответственно.

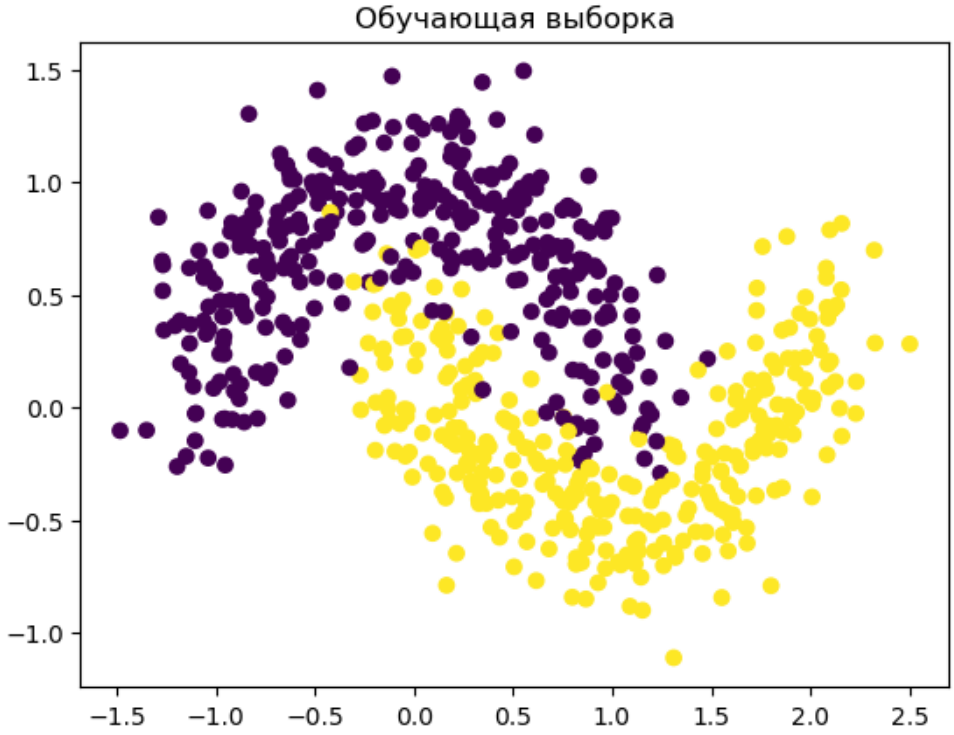

Рисунок 21 – Обучающая выборка при 65% от общего размера

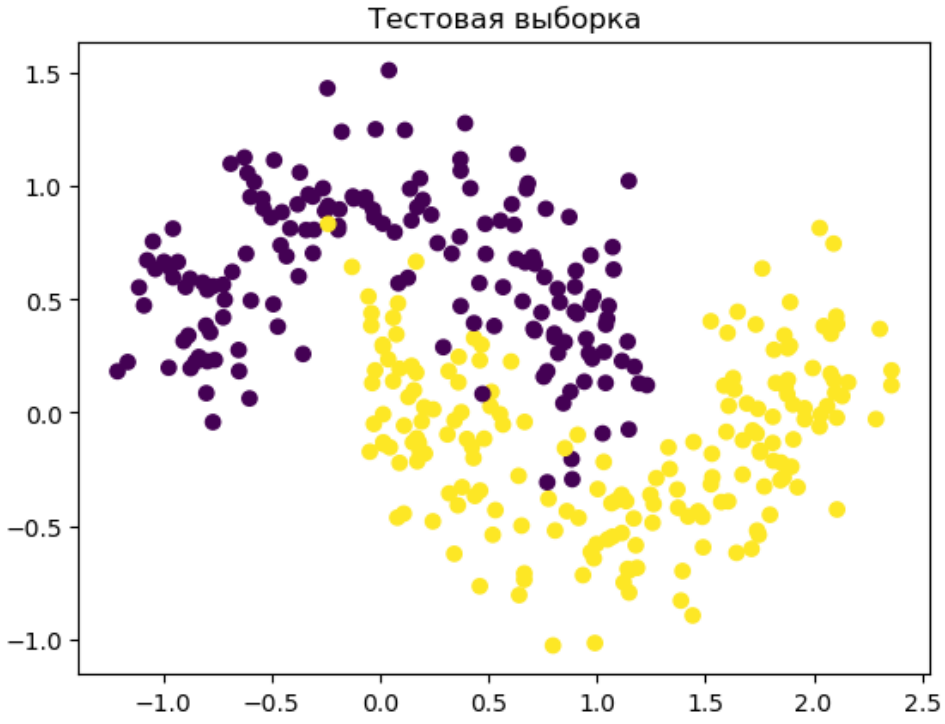

Рисунок 22 – Тестовая выборка при 35% от общего размера

Сведем полученные данные в таблицу 3 и сделаем вывод.

Таблица 3 – Результат классификации по методам про 65% обучающей выборки

Метод (параметры)	Точность	Площадь под кривой
Метод к-ближайших соседей (n_neighbors = 1)	0,946	0,95
Метод к-ближайших соседей (n_neighbors = 3)	0,951	0,951
Метод к-ближайших соседей (n_neighbors = 5)	0,96	0,96
Метод к-ближайших соседей (n_neighbors = 9)	0,963	0,963
Наивный байесовский классификатор	0,854	0,853
Случайный лес (n_estimators = 5)	0,951	0,95
Случайный лес (n_estimators = 10)	0,957	0,958
Случайный лес (n_estimators = 15)	0,96	0,961
Случайный лес (n_estimators = 20)	0,966	0,966
Случайный лес (n_estimators = 50)	0,96	0,961

Исходя из таблицы 3, можно сделать вывод о том, что лучше всего себя показал случайны лес (n_estimators = 9), хуже всего – наивный байесовский классификатор.

Вывод

В ходе выполнения данной лабораторной работы мною были получены практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook. Также я научилась загружать данные, обучать классификаторы и проводить классификацию и получила опыт в оценивании точности полученных моделей.

В результате анализа полученных результатов выяснилось, что наивысшая точность классификации 0,99 (и наибольшая площадь под кривой) достигается, когда размер обучающей выборки равен 90% и в качестве метода выбран метод к-ближайших соседей (n_neighbors = 1) и случайный лес (n_estimators = 10, n_estimators = 15 и n_estimators = 50), хуже всего – наивный байесовский классификатор.

Приложение А

Исходный код при 25% тестовой выборки

```
#!/usr/bin/env python
# coding: utf-8

# ## Лабораторная работа №1
# ### Задание
# ### Вариант №8
# ### Вид классов: `moons`
# ### Random state: `15`
# ### noise: `0.2`

# In[1]:


# Модуль numpy (сокращение от "Numerical Python") предоставляет
функциональность для эффективной работы
# с массивами и математическими операциями на ними.
import numpy as np

# Модуль matplotlib.pyplot используется для создания графиков и визуализации
данных.
# Он предоставляет множество функций для построения различных типов графиков.
import matplotlib.pyplot as plt

from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split


# In[2]:


# А для отображения на графике области принятия решения - готовую функцию
plot_2d_separator,
# которой нужно передать на вход объект classifier - модель классификатора и
X - массив входных данных:
def plot_2d_separator(classifier, X, fill=False, line=True, ax=None,
eps=None):
    if eps is None:
        eps = 1.0
    x_min, x_max = X[:, 0].min() - eps, X[:, 0].max() + eps
    y_min, y_max = X[:, 1].min() - eps, X[:, 1].max() + eps
    xx = np.linspace(x_min, x_max, 100)
    yy = np.linspace(y_min, y_max, 100)
    x1, x2 = np.meshgrid(xx, yy)
    X_grid = np.c_[x1.ravel(), x2.ravel()]
    try:
        decision_values = classifier.decision_function(X_grid)
        levels = [0]
        fill_levels = [decision_values.min(), 0, decision_values.max()]
    except AttributeError:
        decision_values = classifier.predict_proba(X_grid)[:, 1]
        levels = [.5]
        fill_levels = [0, .5, 1]

    if ax is None:
        ax = plt.gca()
    if fill:
        ax.contourf(x1,
                    x2,
                    decision_values.reshape(x1.shape),
```

```
                    levels=fill_levels,
                    colors=['cyan', 'pink', 'yellow'])
    if line:
        ax.contour(x1,
                   x2,
                   decision_values.reshape(x1.shape),
                   levels=levels,
                   colors='black')

    ax.set_xlim(x_min, x_max)
    ax.set_ylim(y_min, y_max)
    ax.set_xticks(())
    ax.set_yticks(())


# ### Генерация выборки

# In[3]:


X, y = make_moons(n_samples=1000, shuffle=True, noise=0.2, random_state=15)


# In[4]:


print('Координаты точек: ')
print(X[:15])
print('Метки класса: ')
print(y[:15])


# In[5]:


plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()


# ### Разбитие выборки на обучающее и тестовое множество (75/25)

# In[6]:


X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=15)


# In[7]:


plt.title('Обучающая выборка')
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
plt.show()


# In[8]:


plt.title('Тестовая выборка')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test)
```

```
plt.show()


# ### Классификация

# In[9]:


from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score


# In[10]:


def show_info(classifier, classifier_name, real_values, prediction_values):
    print(f'Метод классификации: {classifier_name}\n')

    # Выводим предсказанное и реальное значение
    print('Предсказанные и реальные значения:')
    print(prediction_values)
    print(real_values)

    # Выводим матрицу неточностей
    print('\nМатрица неточностей')
    print(confusion_matrix(real_values, prediction_values))

    # Выводим точность классификации
    print(f'\nТочность классификации: {accuracy_score(prediction_values,
real_values)}')

    # Выводим полноту
    print('\nПолнота: ')
    print(classification_report(real_values, prediction_values))

    # AUC ROC
    print(f'\nПлощадь под кривой: {roc_auc_score(real_values,
prediction_values)}')

    plt.xlabel('Первый класс')
    plt.ylabel('Второй класс')
    plt.title(classifier_name.upper())
    plot_2d_separator(classifier, X, fill=True)
    plt.scatter(X[:, 0], X[:, 1], c=y, s=70)


# ### Метод k-ближайших соседей (1)

# In[11]:


from sklearn.neighbors import KNeighborsClassifier


# In[12]:


knn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)
```

```
# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (1)', y_test, prediction)


# ### Метод k-ближайших соседей (3)

# In[13]:


knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (3)', y_test, prediction)


# ### Метод k-ближайших соседей (5)

# In[14]:


knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (5)', y_test, prediction)


# ### Метод k-ближайших соседей (9)

# In[15]:


knn = KNeighborsClassifier(n_neighbors=9, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (9)', y_test, prediction)


# ## Наивный байесовский классификатор

# In[16]:
```

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

# Обучаем модель данных
nb.fit(X_train, y_train)

# Оцениваем качество модели
prediction = nb.predict(X_test)

# Выводим сводную информацию
show_info(nb, 'Наивный байесовский классификатор', y_test, prediction)


# ### Случайный лес (5)

# In[17]:


from sklearn.ensemble import RandomForestClassifier


# In[18]:


rfc = RandomForestClassifier(n_estimators=5)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (5)', y_test, prediction)


# ### Случайный лес (10)

# In[19]:


rfc = RandomForestClassifier(n_estimators=10)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (10)', y_test, prediction)


# ### Случайный лес (15)

# In[20]:


rfc = RandomForestClassifier(n_estimators=15)

# Обучаем модель данных
```

```
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (15)', y_test, prediction)


# ### Случайный лес (20)

# In[21]:


rfc = RandomForestClassifier(n_estimators=20)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (20)', y_test, prediction)


# ### Случайный лес (50)

# In[22]:


rfc = RandomForestClassifier(n_estimators=50)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (50)', y_test, prediction)
```

Приложение Б

Исходный код при 10% тестовой выборки

```
#!/usr/bin/env python
# coding: utf-8

# ## Лабораторная работа №1
# ### Задание
# ### Вариант №8
# ### Вид классов: `moons`
# ### Random state: `15`
# ### noise: `0.2`

# In[23]:


# Модуль numpy (сокращение от "Numerical Python") предоставляет
функциональность для эффективной работы
# с массивами и математическими операциями на ними.
import numpy as np

# Модуль matplotlib.pyplot используется для создания графиков и визуализации
данных.
# Он предоставляет множество функций для построения различных типов графиков.
import matplotlib.pyplot as plt

from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split


# In[24]:


# А для отображения на графике области принятия решения - готовую функцию
plot_2d_separator,
# которой нужно передать на вход объект classifier - модель классификатора и
X - массив входных данных:
def plot_2d_separator(classifier, X, fill=False, line=True, ax=None,
eps=None):
    if eps is None:
        eps = 1.0
    x_min, x_max = X[:, 0].min() - eps, X[:, 0].max() + eps
    y_min, y_max = X[:, 1].min() - eps, X[:, 1].max() + eps
    xx = np.linspace(x_min, x_max, 100)
    yy = np.linspace(y_min, y_max, 100)
    x1, x2 = np.meshgrid(xx, yy)
    X_grid = np.c_[x1.ravel(), x2.ravel()]
    try:
        decision_values = classifier.decision_function(X_grid)
        levels = [0]
        fill_levels = [decision_values.min(), 0, decision_values.max()]
    except AttributeError:
        decision_values = classifier.predict_proba(X_grid)[:, 1]
        levels = [.5]
        fill_levels = [0, .5, 1]

    if ax is None:
        ax = plt.gca()
    if fill:
        ax.contourf(x1,
                    x2,
                    decision_values.reshape(x1.shape),
```

```
                    levels=fill_levels,
                    colors=['cyan', 'pink', 'yellow'])
    if line:
        ax.contour(x1,
                   x2,
                   decision_values.reshape(x1.shape),
                   levels=levels,
                   colors='black')

    ax.set_xlim(x_min, x_max)
    ax.set_ylim(y_min, y_max)
    ax.set_xticks(())
    ax.set_yticks(())


# ### Генерация выборки

# In[25]:


X, y = make_moons(n_samples=1000, shuffle=True, noise=0.2, random_state=15)


# In[26]:


print('Координаты точек: ')
print(X[:15])
print('Метки класса: ')
print(y[:15])


# In[27]:


plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()


# ### Разбитие выборки на обучающее и тестовое множество (90/10)

# In[28]:


X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.10,
                                                    random_state=15)


# In[29]:


plt.title('Обучающая выборка')
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
plt.show()


# In[30]:


plt.title('Тестовая выборка')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test)
```

```
plt.show()


# ### Классификация

# In[31]:


from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score


# In[32]:


def show_info(classifier, classifier_name, real_values, prediction_values):
    print(f'Метод классификации: {classifier_name}\n')

    # Выводим предсказанное и реальное значение
    print('Предсказанные и реальные значения:')
    print(prediction_values)
    print(real_values)

    # Выводим матрицу неточностей
    print('\nМатрица неточностей')
    print(confusion_matrix(real_values, prediction_values))

    # Выводим точность классификации
    print(f'\nТочность классификации: {accuracy_score(prediction_values,
real_values)}')

    # Выводим полноту
    print('\nПолнота: ')
    print(classification_report(real_values, prediction_values))

    # AUC ROC
    print(f'\nПлощадь под кривой: {roc_auc_score(real_values,
prediction_values)}')

    plt.xlabel('Первый класс')
    plt.ylabel('Второй класс')
    plt.title(classifier_name.upper())
    plot_2d_separator(classifier, X, fill=True)
    plt.scatter(X[:, 0], X[:, 1], c=y, s=70)


# ### Метод k-ближайших соседей (1)

# In[33]:


from sklearn.neighbors import KNeighborsClassifier


# In[34]:


knn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)
```

```
# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (1)', y_test, prediction)


# ### Метод k-ближайших соседей (3)

# In[35]:


knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (3)', y_test, prediction)


# ### Метод k-ближайших соседей (5)

# In[36]:


knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (5)', y_test, prediction)


# ### Метод k-ближайших соседей (9)

# In[37]:


knn = KNeighborsClassifier(n_neighbors=9, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (9)', y_test, prediction)


# ## Наивный байесовский классификатор

# In[38]:
```

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

# Обучаем модель данных
nb.fit(X_train, y_train)

# Оцениваем качество модели
prediction = nb.predict(X_test)

# Выводим сводную информацию
show_info(nb, 'Наивный байесовский классификатор', y_test, prediction)


# ### Случайный лес (5)

# In[39]:


from sklearn.ensemble import RandomForestClassifier


# In[40]:


rfc = RandomForestClassifier(n_estimators=5)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (5)', y_test, prediction)


# ### Случайный лес (10)

# In[41]:


rfc = RandomForestClassifier(n_estimators=10)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (10)', y_test, prediction)


# ### Случайный лес (15)

# In[42]:


rfc = RandomForestClassifier(n_estimators=15)

# Обучаем модель данных
```

```
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (15)', y_test, prediction)


# ### Случайный лес (20)

# In[43]:


rfc = RandomForestClassifier(n_estimators=20)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (20)', y_test, prediction)


# ### Случайный лес (50)

# In[44]:


rfc = RandomForestClassifier(n_estimators=50)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (50)', y_test, prediction)
```

Приложение В

Исходный код при 35% тестовой выборки

```
#!/usr/bin/env python
# coding: utf-8

# ## Лабораторная работа №1
# ### Задание
# ### Вариант №8
# ### Вид классов: `moons`
# ### Random state: `15`
# ### noise: `0.2`

# In[1]:


# Модуль numpy (сокращение от "Numerical Python") предоставляет
функциональность для эффективной работы
# с массивами и математическими операциями на ними.
import numpy as np

# Модуль matplotlib.pyplot используется для создания графиков и визуализации
данных.
# Он предоставляет множество функций для построения различных типов графиков.
import matplotlib.pyplot as plt

from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split


# In[2]:


# А для отображения на графике области принятия решения - готовую функцию
plot_2d_separator,
# которой нужно передать на вход объект classifier - модель классификатора и
X - массив входных данных:
def plot_2d_separator(classifier, X, fill=False, line=True, ax=None,
eps=None):
    if eps is None:
        eps = 1.0
    x_min, x_max = X[:, 0].min() - eps, X[:, 0].max() + eps
    y_min, y_max = X[:, 1].min() - eps, X[:, 1].max() + eps
    xx = np.linspace(x_min, x_max, 100)
    yy = np.linspace(y_min, y_max, 100)
    x1, x2 = np.meshgrid(xx, yy)
    X_grid = np.c_[x1.ravel(), x2.ravel()]
    try:
        decision_values = classifier.decision_function(X_grid)
        levels = [0]
        fill_levels = [decision_values.min(), 0, decision_values.max()]
    except AttributeError:
        decision_values = classifier.predict_proba(X_grid)[:, 1]
        levels = [.5]
        fill_levels = [0, .5, 1]

    if ax is None:
        ax = plt.gca()
    if fill:
        ax.contourf(x1,
                    x2,
                    decision_values.reshape(x1.shape),
```

```
                    levels=fill_levels,
                    colors=['cyan', 'pink', 'yellow'])
    if line:
        ax.contour(x1,
                   x2,
                   decision_values.reshape(x1.shape),
                   levels=levels,
                   colors='black')

    ax.set_xlim(x_min, x_max)
    ax.set_ylim(y_min, y_max)
    ax.set_xticks(())
    ax.set_yticks(())


# ### Генерация выборки

# In[3]:


X, y = make_moons(n_samples=1000, shuffle=True, noise=0.2, random_state=15)


# In[4]:


print('Координаты точек: ')
print(X[:15])
print('Метки класса: ')
print(y[:15])


# In[5]:


plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()


# ### Разбитие выборки на обучающее и тестовое множество (65/35)

# In[6]:


X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.35,
                                                    random_state=15)


# In[7]:


plt.title('Обучающая выборка')
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
plt.show()


# In[8]:


plt.title('Тестовая выборка')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test)
```

```
plt.show()


# ### Классификация

# In[9]:


from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score


# In[10]:


def show_info(classifier, classifier_name, real_values, prediction_values):
    print(f'Метод классификации: {classifier_name}\n')

    # Выводим предсказанное и реальное значение
    print('Предсказанные и реальные значения:')
    print(prediction_values)
    print(real_values)

    # Выводим матрицу неточностей
    print('\nМатрица неточностей')
    print(confusion_matrix(real_values, prediction_values))

    # Выводим точность классификации
    print(f'\nТочность классификации: {accuracy_score(prediction_values,
real_values)}')

    # Выводим полноту
    print('\nПолнота: ')
    print(classification_report(real_values, prediction_values))

    # AUC ROC
    print(f'\nПлощадь под кривой: {roc_auc_score(real_values,
prediction_values)}')

    plt.xlabel('Первый класс')
    plt.ylabel('Второй класс')
    plt.title(classifier_name.upper())
    plot_2d_separator(classifier, X, fill=True)
    plt.scatter(X[:, 0], X[:, 1], c=y, s=70)


# ### Метод k-ближайших соседей (1)

# In[11]:


from sklearn.neighbors import KNeighborsClassifier


# In[12]:


knn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)
```

```
# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (1)', y_test, prediction)


# ### Метод k-ближайших соседей (3)

# In[13]:


knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (3)', y_test, prediction)


# ### Метод k-ближайших соседей (5)

# In[14]:


knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (5)', y_test, prediction)


# ### Метод k-ближайших соседей (9)

# In[15]:


knn = KNeighborsClassifier(n_neighbors=9, metric='euclidean')

# Обучаем модель данных
knn.fit(X_train, y_train)

# Оцениваем качество модели
prediction = knn.predict(X_test)

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (9)', y_test, prediction)


# ## Наивный байесовский классификатор

# In[16]:
```

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

# Обучаем модель данных
nb.fit(X_train, y_train)

# Оцениваем качество модели
prediction = nb.predict(X_test)

# Выводим сводную информацию
show_info(nb, 'Наивный байесовский классификатор', y_test, prediction)


# ### Случайный лес (5)

# In[17]:


from sklearn.ensemble import RandomForestClassifier


# In[18]:


rfc = RandomForestClassifier(n_estimators=5)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (5)', y_test, prediction)


# ### Случайный лес (10)

# In[19]:


rfc = RandomForestClassifier(n_estimators=10)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (10)', y_test, prediction)


# ### Случайный лес (15)

# In[20]:


rfc = RandomForestClassifier(n_estimators=15)

# Обучаем модель данных
```

```
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (15)', y_test, prediction)


# ### Случайный лес (20)

# In[21]:


rfc = RandomForestClassifier(n_estimators=20)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (20)', y_test, prediction)


# ### Случайный лес (50)

# In[22]:


rfc = RandomForestClassifier(n_estimators=50)

# Обучаем модель данных
rfc.fit(X_train, y_train)

# Оцениваем качество модели
prediction = rfc.predict(X_test)

# Выводим сводную информацию
show_info(rfc, 'случайный лес (50)', y_test, prediction)
```