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 Цыганов Н.А.

Группы М-ИАП-23

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.

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:

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

Наивный байесовский метод

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.

Ход работы

Вариант по журналу 17, вариантов 12, следовательно: вариант 5 представлен на рисунке 1.

5
moons
41
-
0.25
-

Рисунок 1 - Вариант для выполнения

На рисунке 2 изображен импорт библиотек.

```
# Вариант 5 random state = 41, noize 0,25
import numpy as np


import matplotlib.pyplot as plt

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

Рисунок 2 - Импорт необходимых библиотек

```
In [3]: X, y = make_moons(n_samples=1000, shuffle=True, noise=0.25, random_state=41)
```

```
In [4]: print('Координаты точек: ')
        print(X[:15])
        print('Метки класса: ')
        print(y[:15])
```

```
Координаты точек:
[[ 2.43925858e-01  8.64335729e-01]
 [ 2.16565685e-01  7.74035450e-02]
 [ 1.86240848e+00 -3.74002484e-01]
 [-9.75022520e-01  9.62516559e-02]
 [ 2.45733769e-01 -1.20800098e-01]
 [ 2.26162888e+00  8.10738624e-01]
 [ 1.08713469e-01  2.16517085e-01]
 [-3.61582169e-01  9.36581439e-01]
 [-7.04683882e-01  4.85704886e-01]
 [ 1.48133028e+00 -1.31269733e-03]
 [-3.99966999e-01  1.34289554e+00]
 [-4.68160123e-01  1.24150659e+00]
 [ 1.41350790e+00 -1.51963621e-01]
 [ 5.99243888e-01  7.13687210e-01]
 [ 1.91390078e+00  5.42106080e-01]]
Метки класса:
[0 1 1 0 1 1 1 0 0 1 0 0 1 0 1]
```

Рисунок 3 - Генерация данных

График сгенерированной выборки на рисунке 4.

```
colors = np.where(y == 1, 'r', 'b')

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

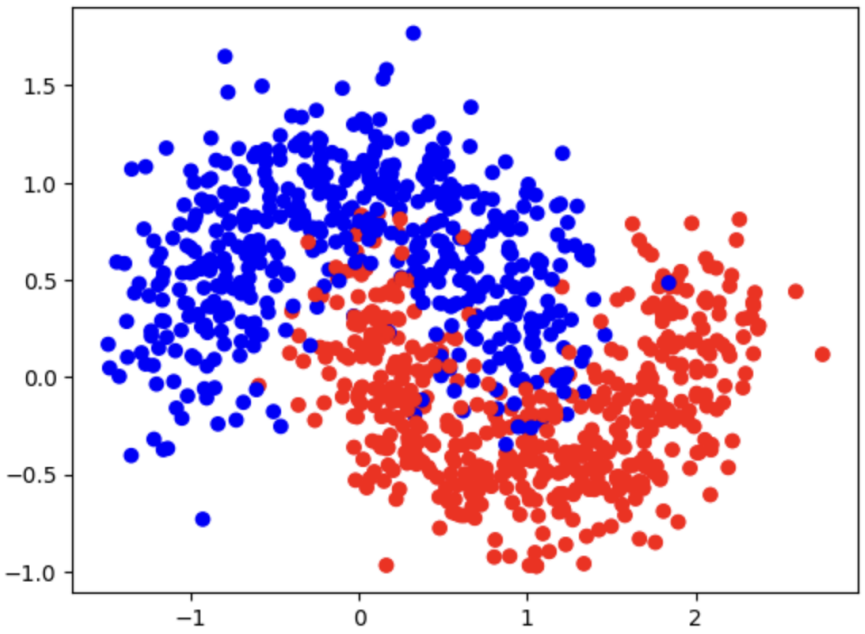


Рисунок 4 - Сгенерированная выборка

Разделим данных на обучающую и тестовую выборку. Для этого воспользуемся функцией train_test_split из пакета sklearn.model_selection. Представлено на рисунке 5.

Обучающее и тестовое множество (75/25)

[6]:
```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=41)
```

[7]:
```
colors_train = np.where(y_train == 1, 'r', 'b')
plt.title('Обучающая выборка')
plt.scatter(X_train[:, 0], X_train[:, 1], c=colors_train)
plt.show()
```

Рисунок 5 - Обучающая выборка

Рисунок тестового множества представлен на рисунке 6.

[24]:
```
colors_test = np.where(y_test == 1, 'r', 'b')
plt.title('Тестовая выборка')
plt.scatter(X_test[:, 0], X_test[:, 1], c=colors_test)
plt.show()
```

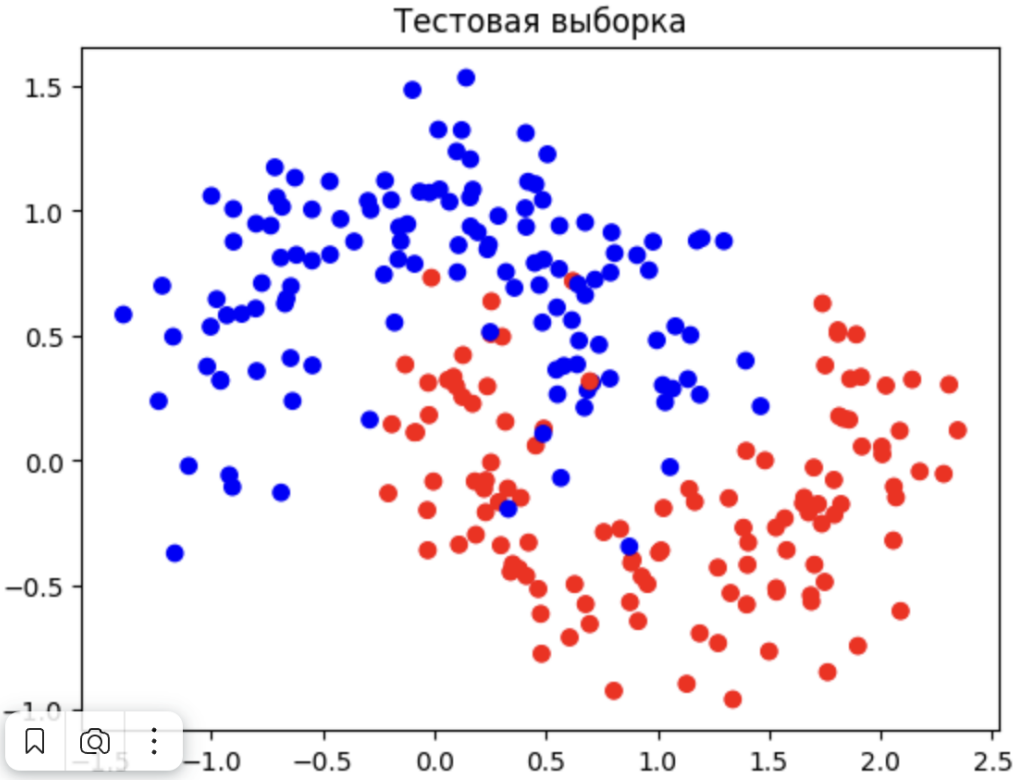


Рисунок 6 - Тестовая выборка

Перейдем к классификации, код представлен на рисунке 7.

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

    print('Предсказанные и реальные значения:')
    print(prediction_values)
    print(real_values)

    print('\nМатрица неточностей')
    print(confusion_matrix(real_values, prediction_values))

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

    print('\nПолнота: ')
    print(classification_report(real_values, prediction_values))

    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())

    colors = np.where(y == 1, 'r', 'b')

    # Рисуем разделяющую плоскость
    plot_2d_separator(classifier, X, fill=True)

    plt.scatter(X[:, 0], X[:, 1], c=colors, s=70)

    plt.show()
```

Рисунок 7 - Отображение для классификации

Метод ближайших соседей k=1 представлен на рисунке 8-10.

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

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

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

show_info(knn, 'ближайшие соседи (1)', y_test, prediction, X_test, y_test)
```

Рисунок 8 - Код для выполнения

```
Метод классификации: ближайшие соседи (1)

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

Матрица неточностей
[[110  19]
 [  9 112]]

Точность классификации: 0.888

Полнота:
              precision    recall  f1-score   support

           0       0.92      0.85      0.89       129
           1       0.85      0.93      0.89       121

    accuracy                           0.89       250
   macro avg       0.89      0.89      0.89       250
weighted avg       0.89      0.89      0.89       250


Площадь под кривой: 0.8891665065026586
```

Рисунок 9 - Данные

```
Площадь под кривой: 0.8891665065026586
```

Рисунок 10 - График

Также необходимо сделать для k = 3,5,9.

```
Точность классификации: 0.916

Полнота:
              precision    recall  f1-score   support

           0       0.93      0.91      0.92       129
           1       0.90      0.93      0.91       121

    accuracy                           0.92       250
   macro avg       0.92      0.92      0.92       250
weighted avg       0.92      0.92      0.92       250

Площадь под кривой: 0.9162982894483952
```


```
Точность классификации: 0.924

Полнота:
              precision    recall  f1-score   support

           0       0.94      0.91      0.93       129
           1       0.91      0.93      0.92       121

    accuracy                           0.92       250
   macro avg       0.92      0.92      0.92       250
weighted avg       0.92      0.92      0.92       250

Площадь под кривой: 0.9243064898456018
```


```
Точность классификации: 0.94

Полнота:
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       129
           1       0.93      0.94      0.94       121

    accuracy                           0.94       250
   macro avg       0.94      0.94      0.94       250
weighted avg       0.94      0.94      0.94       250

Площадь под кривой: 0.9400666282273047
```

Рисунок 11 - Графики

Наивный байесовский классификатор и его выполнение представлен на рисунках 12-13.

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

nb.fit(X_train, y_train)

prediction = nb.predict(X_test)

show_info(nb, 'Наивный байес', y_test, prediction,  X_test, y_test)
```

Рисунок 12 - Импорт необходимой библиотеки и код

```
Метод классификации: Наивный байес

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

Матрица неточностей
[[112  17]
 [ 18 103]]

Точность классификации: 0.86

Полнота:
              precision    recall  f1-score   support

           0       0.86      0.87      0.86       129
           1       0.86      0.85      0.85       121

    accuracy                           0.86       250
   macro avg       0.86      0.86      0.86       250
weighted avg       0.86      0.86      0.86       250


Площадь под кривой: 0.8597283618425268
```

Рисунок 13 - Результат работы

Теперь используем метод классификации “Случайный лес” с параметрами (n_estimators = {5, 10, 15, 20, 50}). Данный метод показан на рисунках 14-15.

Случайный лес n=5

```
7]: from sklearn.ensemble import RandomForestClassifier

8]: rfc = RandomForestClassifier(n_estimators=5)

rfc.fit(X_train, y_train)

prediction = rfc.predict(X_test)

show_info(rfc, 'Случайный лес n=5', y_test, prediction,  X_test, y_test)
```

```
Метод классификации: Случайный лес n=5

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

Матрица неточностей
[[117  12]
 [  8 113]]

Точность классификации: 0.92

Полнота:
              precision    recall  f1-score   support

           0       0.94      0.91      0.92       129
           1       0.90      0.93      0.92       121

    accuracy                           0.92       250
   macro avg       0.92      0.92      0.92       250
weighted avg       0.92      0.92      0.92       250


Площадь под кривой: 0.9204305208533539
```

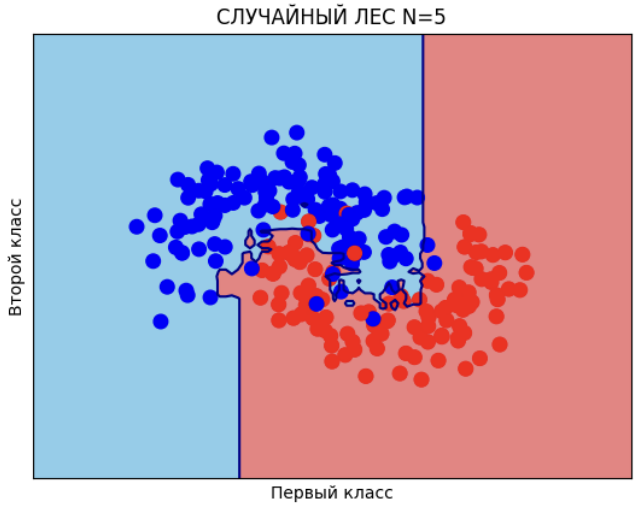


Рисунок 14 - Импорт библиотеки и запуск

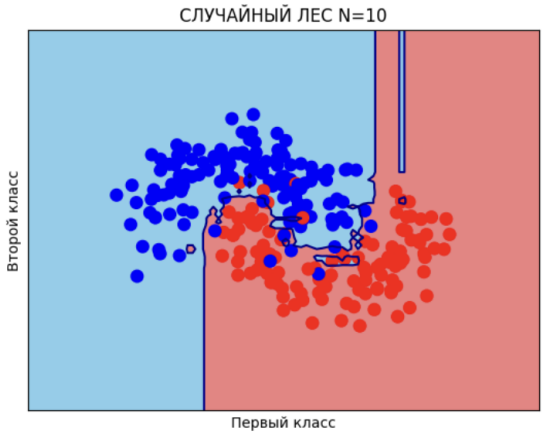
Матрица неточностей
[[119 10]
 [9 112]]

Точность классификации: 0.924

Полнота:

	precision	recall	f1-score	support
0	0.93	0.92	0.93	129
1	0.92	0.93	0.92	121
accuracy			0.92	250
macro avg	0.92	0.92	0.92	250
weighted avg	0.92	0.92	0.92	250

Площадь под кривой: 0.9240502274328912



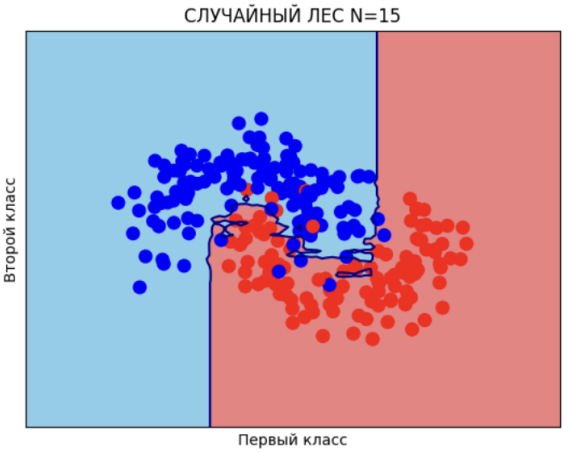
Матрица неточностей
[[120   9]
 [8 113]]

Точность классификации: 0.932

Полнота:

	precision	recall	f1-score	support
0	0.94	0.93	0.93	129
1	0.93	0.93	0.93	121
accuracy			0.93	250
macro avg	0.93	0.93	0.93	250
weighted avg	0.93	0.93	0.93	250

Площадь под кривой: 0.9320584278300981



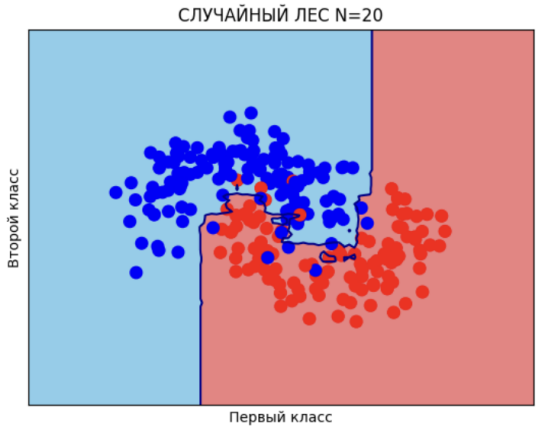
Матрица неточностей
[[119 10]
 [10 111]]

Точность классификации: 0.92

Полнота:

	precision	recall	f1-score	support
0	0.92	0.92	0.92	129
1	0.92	0.92	0.92	121
accuracy			0.92	250
macro avg	0.92	0.92	0.92	250
weighted avg	0.92	0.92	0.92	250

Площадь под кривой: 0.9199179960279327



Матрица неточностей
[[118 11]
 [8 113]]

Точность классификации: 0.924

Полнота:

	precision	recall	f1-score	support
0	0.94	0.91	0.93	129
1	0.91	0.93	0.92	121
accuracy			0.92	250
macro avg	0.92	0.92	0.92	250
weighted avg	0.92	0.92	0.92	250

Площадь под кривой: 0.9243064898456018

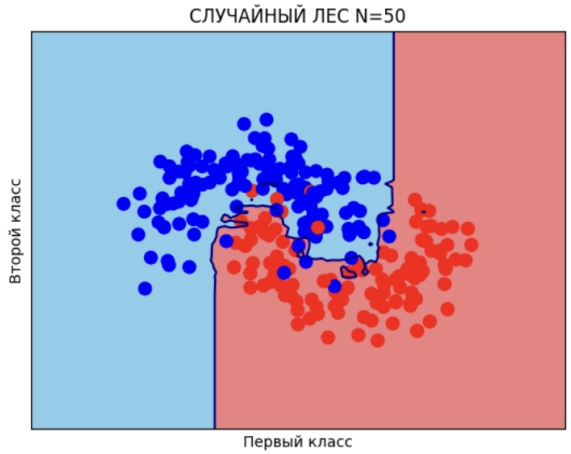


Рисунок 15 - Случайный лес

Теперь необходимо зафиксировать данные, изменить размер обучающей выборки, показано на рисунке 16, повторить тоже самое с разными размерами обучающей выборки, для этого составим таблицу.

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=41)
```

Рисунок 16 - Нужно изменить значение test_size на необходимое

Таблица 1 - Результаты

Метод	Размер выборки 75%		Размер выборки 90%		Размер выборки 65%	
	Точность	Площадь	Точность	Площадь	Точность	Площадь
Метод к-ближайших соседей (n_neighbors = 1)	0.888	0.889	0.89	0.890	0.9028	0.9031
Метод к-ближайших соседей (n_neighbors = 3)	0.916	0.916	0.91	0.910	0.9257	0.9257
Метод к-ближайших соседей (n_neighbors = 5)	0.924	0.924	0.93	0.929	0.93	0.93
Метод к-ближайших соседей (n_neighbors = 9)	0.94	0.94	0.98	0.9799	0.942857	0.942784
Наивный байесовский классификатор	0.86	0.8597	0.85	0.8509	0.8542	0.85417
Случайный лес (n_estimators = 5)	0.92	0.92	0.93	0.929	0.94	0.9399
Случайный лес (n_estimators = 10)	0.924	0.924	0.93	0.9294	0.9257	0.9254
Случайный лес (n_estimators = 15)	0.932	0.932	0.93	0.9310	0.937	0.937
Случайный лес (n_estimators = 20)	0.92	0.9199	0.93	0.9302	0.9314	0.9312
Случайный лес (n_estimators = 50)	0.924	0.924	0.93	0.9302	0.9314	0.93128

Вывод: исходя из сравнения полученных данных размер обучающей и тестовой выборки не сильно влияет на точность предсказания, лучше всего показал себя метод ближайших соседей с n = 9, он показал точность 0.98 при соотношении тестового множества к обучающему в пропорции 1 к 9. Хуже всего показал себя метод наивного байесовского классификатора, который ни разу не показал точность выше 90% на всех соотношениях тестового множества к обучающему. В ходе выполнения работы были получены практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook с загрузкой данных, обучением классификаторов и классификацией.