

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

Сухоруких А.О.

Группа М-ИАП-22

Руководитель

Кургасов В.В.

доцент

Липецк 2022 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:
 - Истинные и предсказанные метки классов
 - Матрицу ошибок (confusion matrix)
 - Значения полноты, точности, f1-меры и аккуратности
 - Значение площади под кривой ошибок (AUC ROC)
 - Отобразить на графике область принятия решений по каждому классу
- В качестве методов классификации использовать:
- a) Метод k-ближайших соседей ($n_neighbors = \{1, 3, 5, 9\}$)
 - b) Наивный байесовский метод
 - c) Случайный лес ($n_estimators = \{5, 10, 15, 20, 50\}$)
- 9) По каждому пункту работы занести в отчет программный код и результат вывода.
 - 10) По результатам п.8 занести в отчет таблицу с результатами классификации всеми методами и выводы о наиболее подходящем методе классификации ваших данных.

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

Вариант: 2

Вид класса: blobs

Random_state: 28

cluster_std: 4.5

noise: -

Centers: 2

Ход работы

1. Создание нового Notebook в среде Jupiter Notebook

Первым шагом создаем виртуальное окружение. Виртуальные окружения — мощный и удобный инструмент изоляции программ друг от друга и от системы. Для этого в терминале необходимо перейти в директорию где предположительно будет проект и ввести команду:

```
virtualenv lab1_env
```

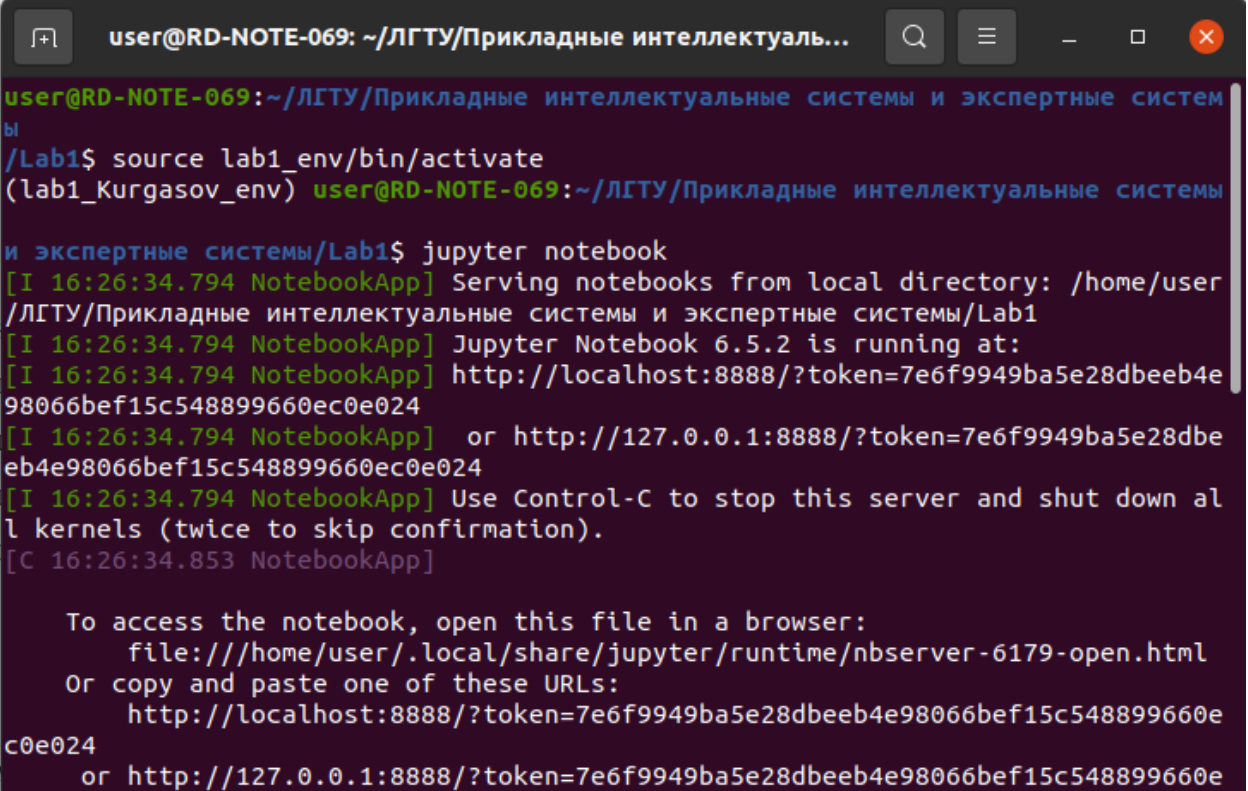
Затем нужно перейти в виртуальное окружение используя команду:

```
source lab1_env/bin/activate
```

После вводим команду для запуска среды Jupiter Notebook:

```
jupyter notebook
```

Результат выполнения команд показан на рисунке 1.



```
user@RD-NOTE-069: ~/ЛГТУ/Прикладные интеллектуаль...
user@RD-NOTE-069:~/ЛГТУ/Прикладные интеллектуальные системы и экспертные системы/Lab1$ source lab1_env/bin/activate
(lab1_Kurgasov_env) user@RD-NOTE-069:~/ЛГТУ/Прикладные интеллектуальные системы и экспертные системы/Lab1$ jupyter notebook
[I 16:26:34.794 NotebookApp] Serving notebooks from local directory: /home/user/ЛГТУ/Прикладные интеллектуальные системы и экспертные системы/Lab1
[I 16:26:34.794 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 16:26:34.794 NotebookApp] http://localhost:8888/?token=7e6f9949ba5e28dbeeb4e98066bef15c548899660ec0e024
[I 16:26:34.794 NotebookApp] or http://127.0.0.1:8888/?token=7e6f9949ba5e28dbeeb4e98066bef15c548899660ec0e024
[I 16:26:34.794 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 16:26:34.853 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/user/.local/share/jupyter/runtime/nbserver-6179-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=7e6f9949ba5e28dbeeb4e98066bef15c548899660ec0e024
    or http://127.0.0.1:8888/?token=7e6f9949ba5e28dbeeb4e98066bef15c548899660ec0e024
```

Рисунок 1 - Включение среды Jupiter Notebook

В процессе использования последней команды будет открыт браузер, в котором будет наша директория, для создания нового Notebook необходимо навести курсор на кнопку new, и выбрать тип файла в данном случае python3 и назвать файл. Процесс создания нового Notebook представлен на рисунке 2.



Рисунок 2 - Создание нового Notebook

2. Импорт необходимых для работы библиотек и модулей

Для данной лабораторной работы нам необходимы такие библиотеки как: numpy, matplotlib, sklearn.

NumPy — это расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами.

Команда для установки:

```
pip install numpy
```

Matplotlib — библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой. Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях.

Команда для установки:

```
pip install matplotlib
```

Scikit-learn (sklearn) — это один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он содержит функции и алгоритмы для машинного обучения: классификации, прогнозирования или разбивки данных на группы. Sklearn написана на языках Python, C, C++ и Cython.

Команда для установки:

```
pip install scikit-learn
```

sklearn.datasets - необходим для создания данных для использования

sklearn.metrics - содержит метрики для оценивания полученных моделей и данных классификации.

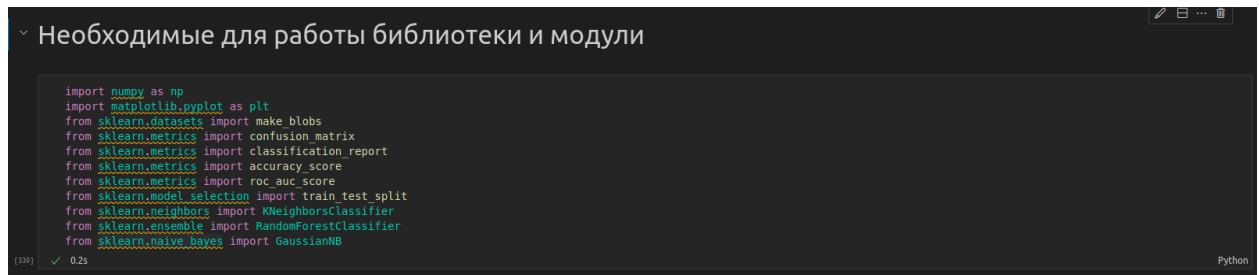
`sklearn.model_selection` - служит для разбиения данных на тестовые и обучающие.

`sklearn.neighbors` - содержит метод классификации к-ближайших соседей.

`sklearn.naive_bayes` - содержит наивный байесовский метод.

`sklearn.ensemble` - содержит метод случайный лес.

Необходимые зависимости показаны на рисунке 2



```
Необходимые для работы библиотеки и модули

import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

[139] ✓ 0.2s Python

Рисунок 2 - Необходимые зависимости

3. Загрузка данных

Процесс загрузки данных показан на рисунке 3. График полученных данных показан на рисунке 4.

✓ Загрузка данных

```
[341] X, y = make_blobs(centers=2, random_state=28, cluster_std=4.5) ✓ 0.1s
```

```
[342] print("Координаты точек: X0|X1\n", X[:15])  
      print("Метки класса: ", y[:15]) ✓ 0.8s
```

```
... Координаты точек: X0|X1  
[[ -2.52467218  5.509991  ]  
 [ -13.4336193  -1.69809935]  
 [ -3.42914564  0.20484467]  
 [  1.86920736  3.3012942  ]  
 [ -11.68128476 -6.35740982]  
 [ -11.21403158  4.70320389]  
 [  4.26436057  9.22904112]  
 [ -4.7448784  -0.15398861]  
 [  2.05099142 -1.90331879]  
 [  4.38832243  1.43395274]  
 [  1.431567    1.4632393  ]  
 [  2.67789298  0.9976655  ]  
 [ -11.3693878 -4.65613344]  
 [  5.63456107  6.80209141]  
 [  4.42550879 -2.60609209]]  
Метки класса: [0 1 1 0 1 1 0 1 0 0 0 0 1 0 0]
```

Рисунок 3 - Процесс загрузки данных

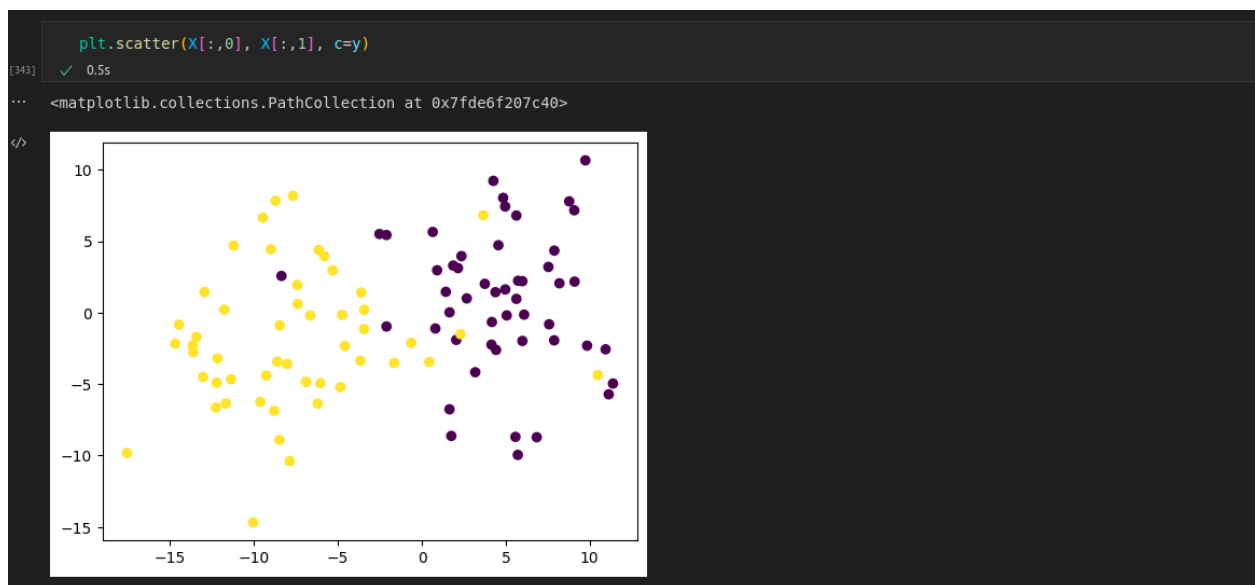


Рисунок 4 - Визуализация данных

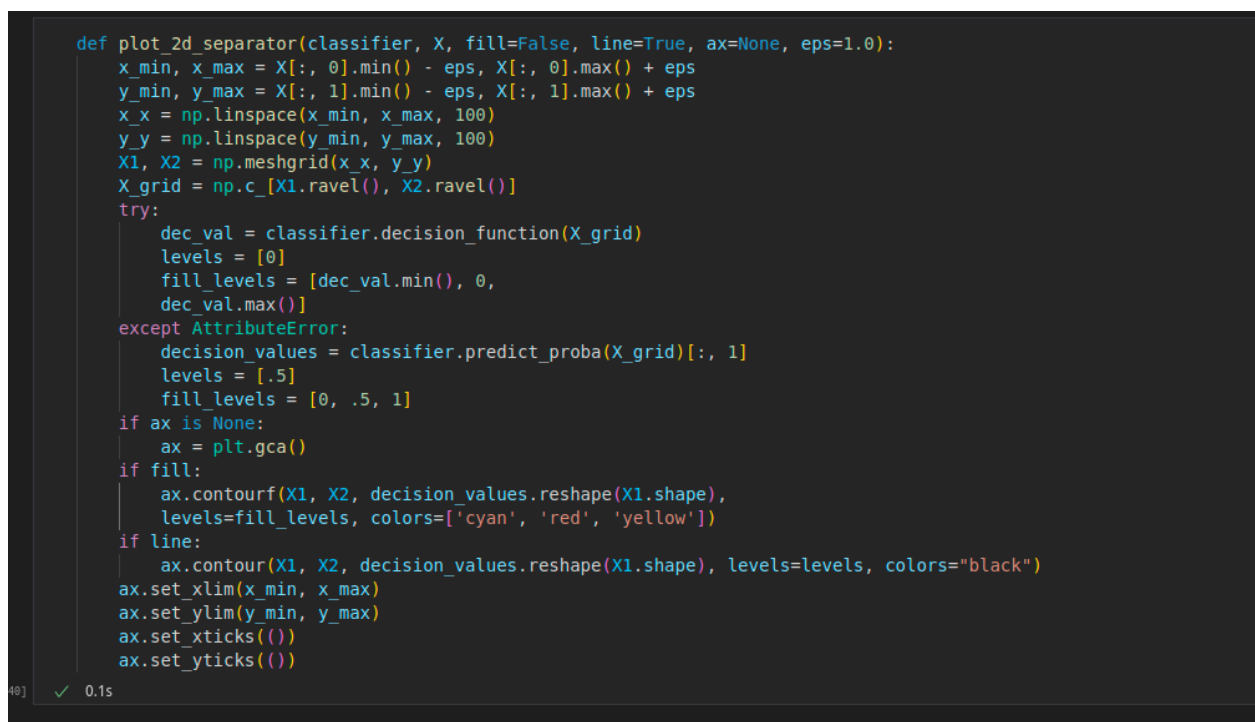


Рисунок 5 - Функция для визуализации данных

4. Разбиение данных на обучающую (train) и тестовую (test) выборки в пропорции 75% - 25%

Для этого воспользуемся функцией `train_test_split`. Процесс разбиения данных и результаты разбиения показаны на рисунках ниже.

```
train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size = 0.25, random_state=1)
```

[344] ✓ 0.6s

+ Code + Markdown

Рисунок 5 - Разбиение данных

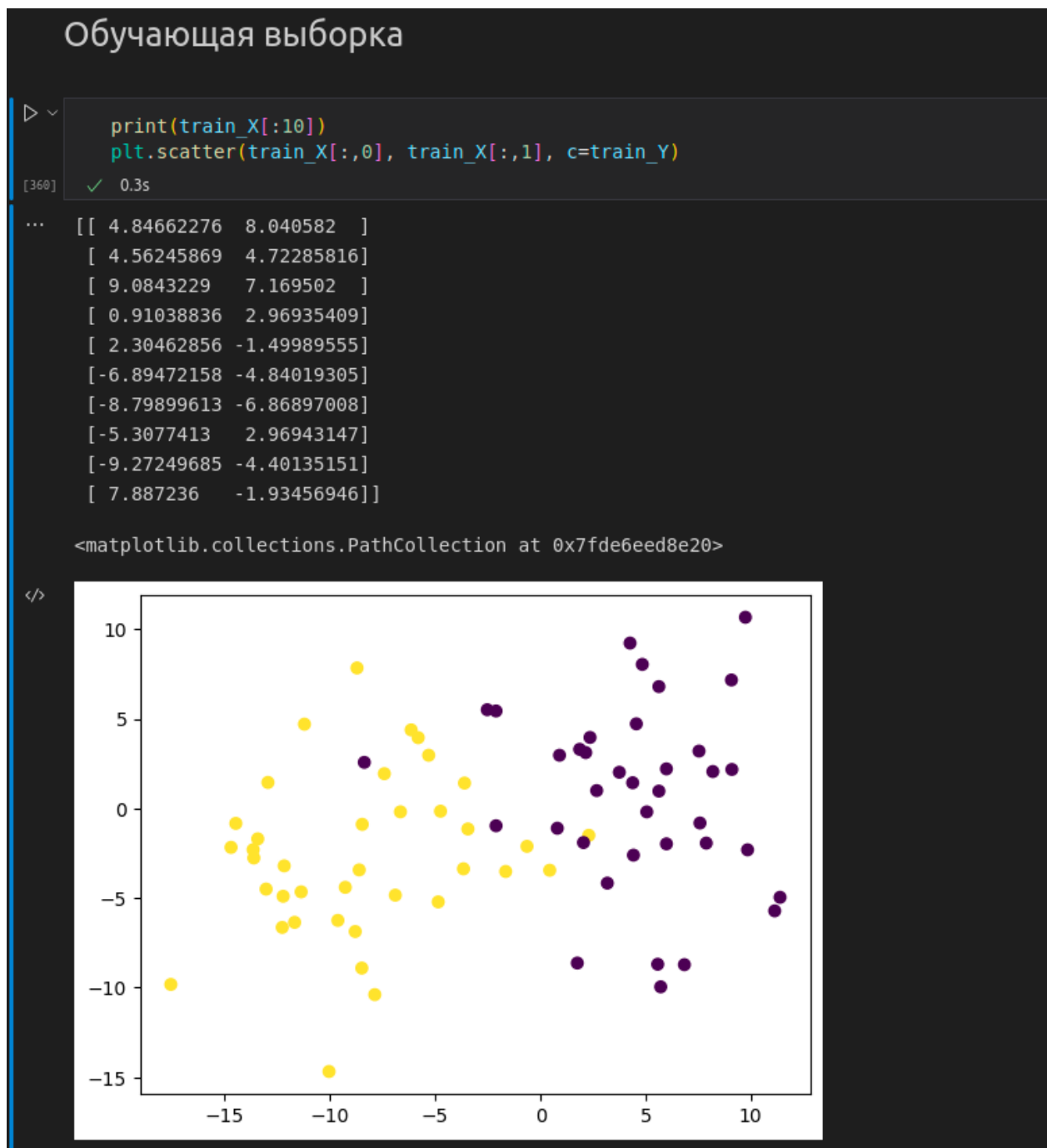


Рисунок 6 - Обучающая выборка

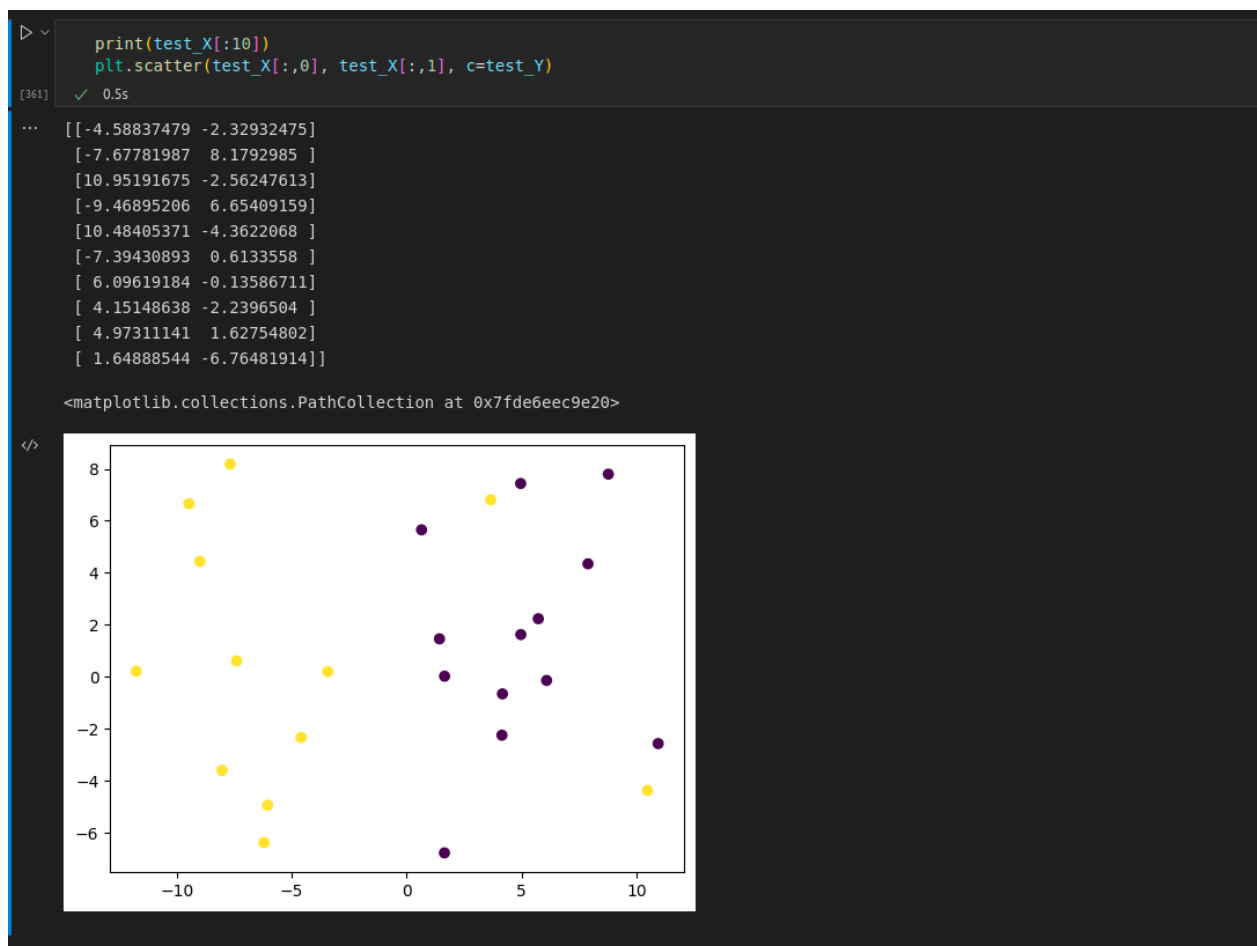


Рисунок 7 - Тестовая выборка

5. Классификация данных

Создадим универсальную функцию, которая будет выводить характеристики классификации на основе данных для теста и полученных в результате предсказания. Реализация такой функции показана на рисунке 8.

```
def print_classification_metrics(classifier, X, y, prediction, test_Y):
    print("Значения предсказанные правдивые тестовые")
    print(prediction)
    print("Значения правдивые тестовые")
    print(test_Y)

    print("Матрица классификации")
    print(confusion_matrix(test_Y, prediction))
    print("Полученная точность классификации: ", accuracy_score(prediction, test_Y))
    print("Значения полноты, точности, f1-меры и аккуратности")
    print(classification_report(test_Y, prediction))
    print("Значение ошибок (AUC ROC)")
    print(roc_auc_score(test_Y, prediction))
    print("Область принятия решений")
    plt.xlabel("first")
    plt.ylabel("second")
    plot_2d_separator(classifier, X, fill=True)
    plt.scatter(X[:, 0], X[:, 1], c=y, s=70)
    plt.show()
```

7] ✓ 0.1s

Рисунок 8 - Реализация функции для оценки классификации

5.1 Метод k-ближайших соседей

Метод ближайших соседей — простейший метрический классификатор, основанный на оценивании сходства объектов. Классифицируемый объект относится к тому классу, которому принадлежат ближайшие к нему объекты обучающей выборки.

5.1.1 Метод k-ближайших соседей $n_neighbors = 1$

Реализация классификации методом k-ближайших соседей с параметром $n_neighbors = 1$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 9 - 10.

Метод k-ближайших соседей n_neighbors = 1

```
classifier = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
classifier.fit(train_X, train_Y)
prediction = classifier.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

✓ 0.1s

Значения предсказанные правдивые тестовые

```
[1 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0]
```

Значения правдивые тестовые

```
[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]
```

Матрица классификации

```
[[13  0]
 [ 3  9]]
```

Полученная точность классификации: 0.88

Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.81	1.00	0.90	13
1	1.00	0.75	0.86	12
accuracy			0.88	25
macro avg	0.91	0.88	0.88	25
weighted avg	0.90	0.88	0.88	25

Значение ошибок (AUC ROC)

0.875

Рисунок 9 - Метод k-ближайших соседей n_neighbors = 1

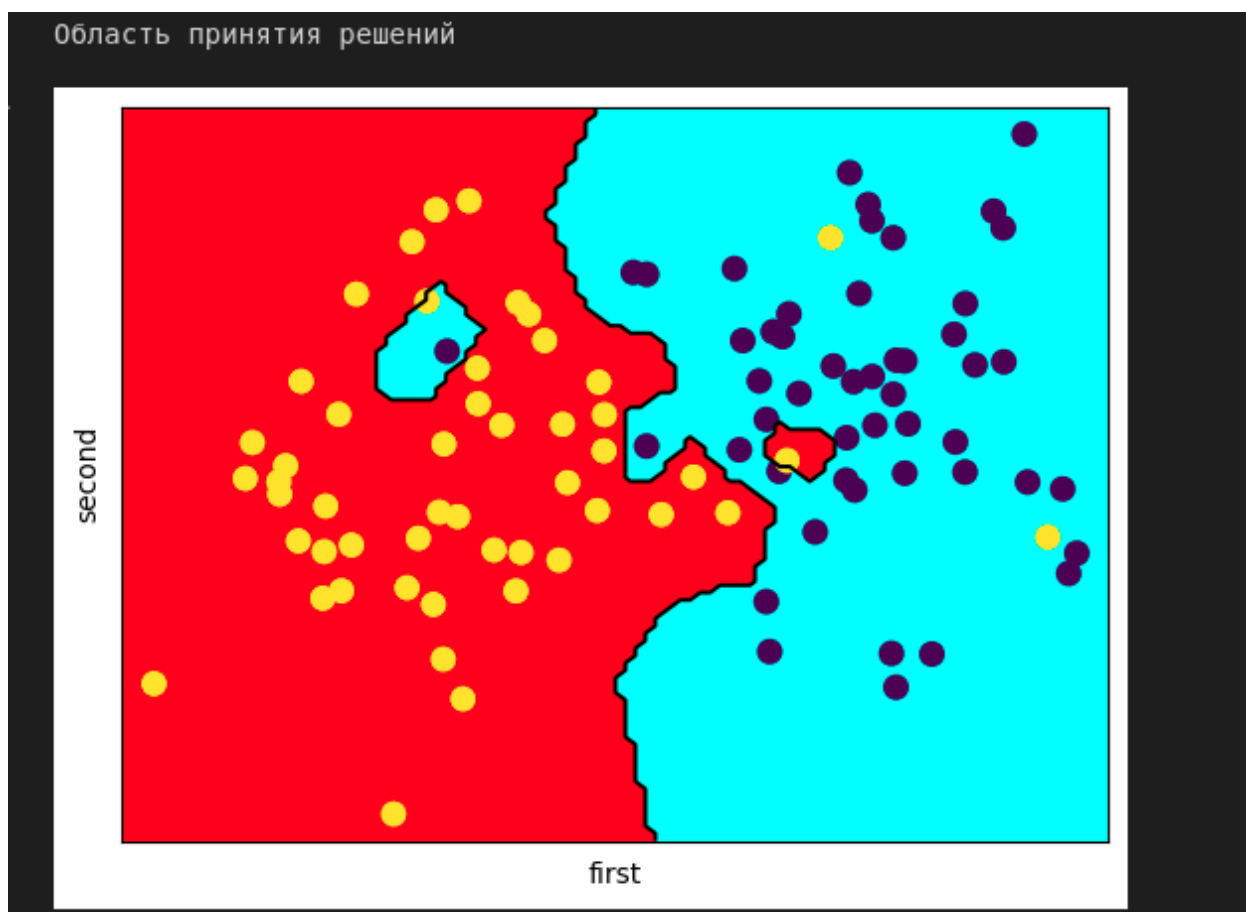


Рисунок 10 - Визуализация данных полученных при классификации

5.1.2 Метод k-ближайших соседей $n_neighbors = 3$

Реализация классификации методом k-ближайших соседей с параметром $n_neighbors = 3$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 11 - 12.

Метод k-ближайших соседей n_neighbors = 3

```
classifier = KNeighborsClassifier(n_neighbors=3, metric='euclidean')
classifier.fit(train_X, train_Y)
prediction = classifier.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

✓ 0.1s

Значения предсказанные правдивые тестовые

[1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]

Значения правдивые тестовые

[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]

Матрица классификации

```
[[13  0]
 [ 2 10]]
```

Полученная точность классификации: 0.92

Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)

0.9166666666666667

Область принятия решений

Рисунок 11 - Метод k-ближайших соседей n_neighbors = 3

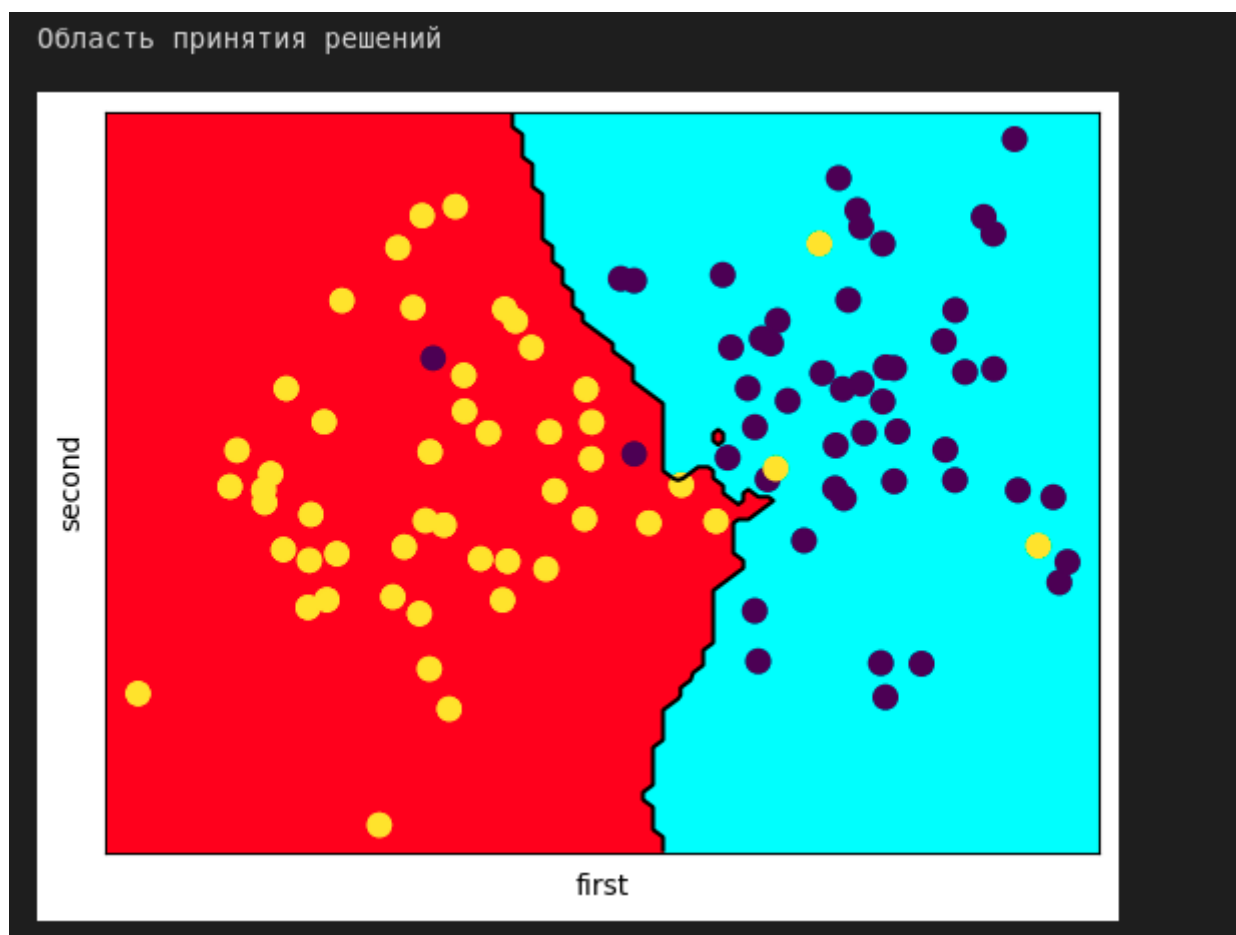


Рисунок 12 - Визуализация данных полученных при классификации

5.1.3 Метод k-ближайших соседей $n_neighbors = 5$

Реализация классификации методом k-ближайших соседей с параметром $n_neighbors = 5$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 13 - 14.

Метод k-ближайших соседей n_neighbors = 5

```
classifier = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
classifier.fit(train_X, train_Y)
prediction = classifier.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

[350] ✓ 0.2s

... Значения предсказанные правдивые тестовые
[1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]
Значения правдивые тестовые
[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]
Матрица классификации
[[13 0]
 [2 10]]
Полученная точность классификации: 0.92
Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)
0.9166666666666667

Рисунок 13 - Метод k-ближайших соседей n_neighbors = 5

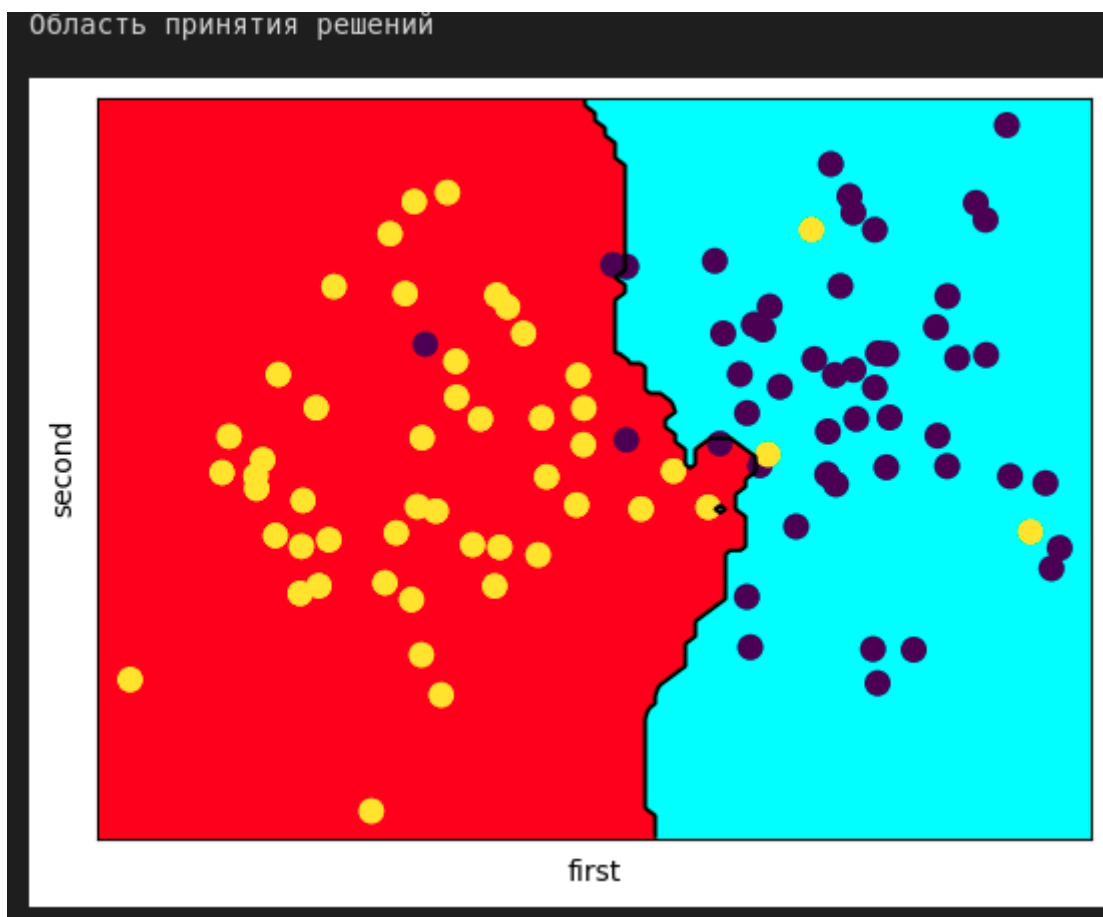


Рисунок 14 - Визуализация данных полученных при классификации

5.1.3 Метод k-ближайших соседей $n_neighbors = 9$

Реализация классификации методом k-ближайших соседей с параметром $n_neighbors = 9$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 15 - 16.

Метод k-ближайших соседей n_neighbors = 9

```
classifier = KNeighborsClassifier(n_neighbors=9, metric='euclidean')
classifier.fit(train_X, train_Y)
prediction = classifier.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

351] ✓ 0.2s

.. Значения предсказанные правдивые тестовые
[1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]
Значения правдивые тестовые
[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]
Матрица классификации
[[13 0]
 [2 10]]
Полученная точность классификации: 0.92
Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)
0.9166666666666667

Рисунок 15 - Метод k-ближайших соседей n_neighbors = 9

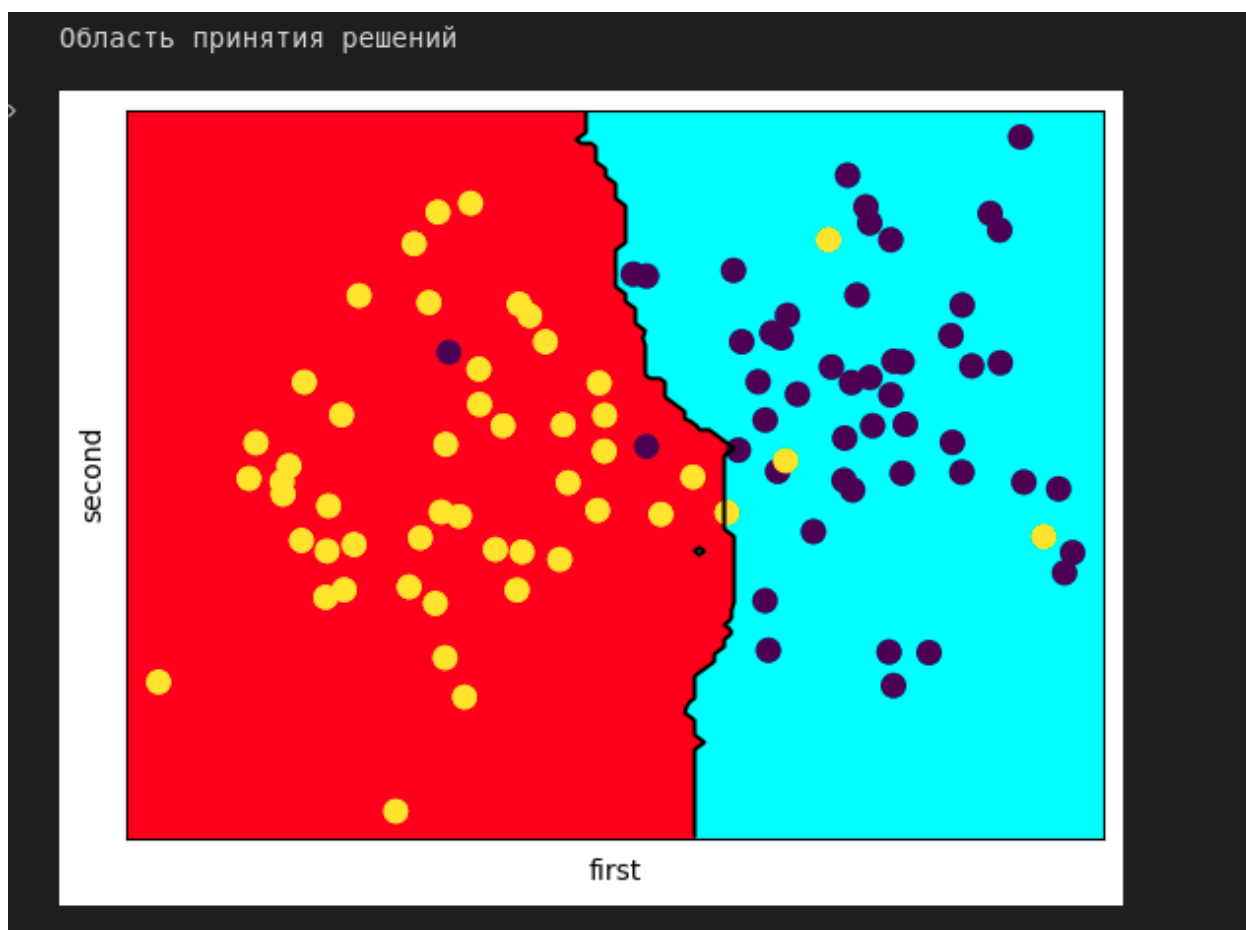


Рисунок 16 - Визуализация данных полученных при классификации

5.2 Наивный байесовский метод

Наивный байесовский классификатор — простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими (наивными) предположениями о независимости.

В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы могут обучаться очень эффективно. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия; другими словами, можно работать с наивной байесовской моделью, не веря в байесовскую вероятность и не используя байесовские методы.

Несмотря на наивный вид и, несомненно, очень упрощенные условия, наивные байесовские классификаторы часто работают намного лучше нейронных сетей во многих сложных жизненных ситуациях.

Достоинством наивного байесовского классификатора является малое количество данных, необходимых для обучения, оценки параметров и классификации.

Классификация данным методом, полученные характеристики оценки классификации и визуализация данных показаны на рисунках 17-18

```
naive = GaussianNB()
naive.fit(train_X, train_Y)
predict = naive.predict(test_X)
print_classification_metrics(naive, X, y, predict, test_Y)
```

✓ 0.2s

Значения предсказанные правдивые тестовые
 [1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]

Значения правдивые тестовые
 [1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]

Матрица классификации
 [[13 0]
 [2 10]]

Полученная точность классификации: 0.92

Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)
 0.9166666666666667

Рисунок 17 - Наивный байесовский метод

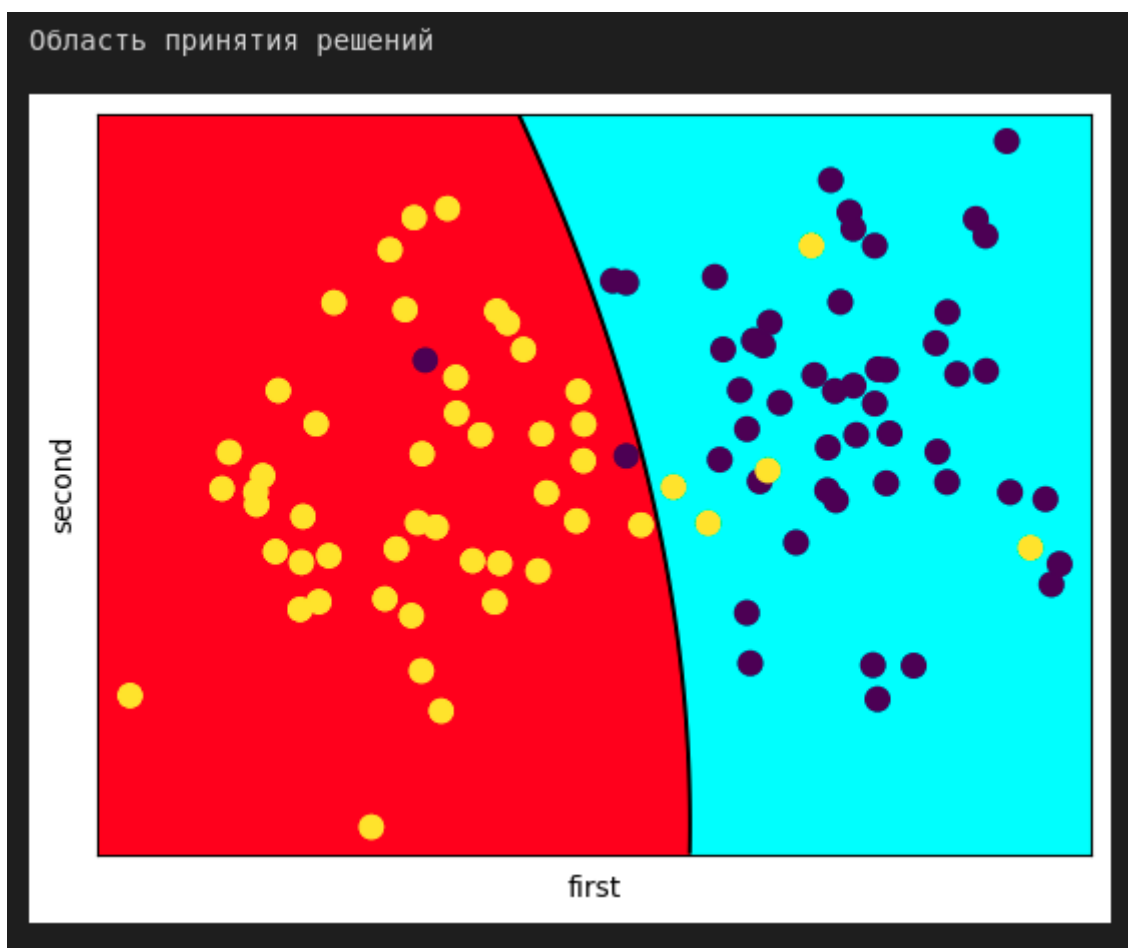


Рисунок 18 - Визуализация данных полученных при классификации

5.3 Случайный лес

Алгоритм случайного леса (Random Forest) — универсальный алгоритм машинного обучения, суть которого состоит в использовании ансамбля решающих деревьев. Само по себе решающее дерево предоставляет крайне невысокое качество классификации, но из-за большого их количества результат значительно улучшается. Также это один из немногих алгоритмов, который можно использовать в абсолютном большинстве задач.

5.3.1 Случайный лес $n_estimators = 5$

Реализация классификации методом случайного леса с параметром $n_estimators = 5$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 19 - 20.

✓ Случайный лес n_estimators 5

```
rand_forest = RandomForestClassifier(n_estimators=5)
rand_forest.fit(train_X, train_Y)
prediction = rand_forest.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

[434] ✓ 0.2s

... Значения предсказанные правдивые тестовые
[1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0]
Значения правдивые тестовые
[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]
Матрица классификации
[[13 0]
 [5 7]]
Полученная точность классификации: 0.8
Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.72	1.00	0.84	13
1	1.00	0.58	0.74	12
accuracy			0.80	25
macro avg	0.86	0.79	0.79	25
weighted avg	0.86	0.80	0.79	25

Значение ошибок (AUC ROC)
0.7916666666666667

Рисунок 19 - Метод случайного леса n_estimators 5

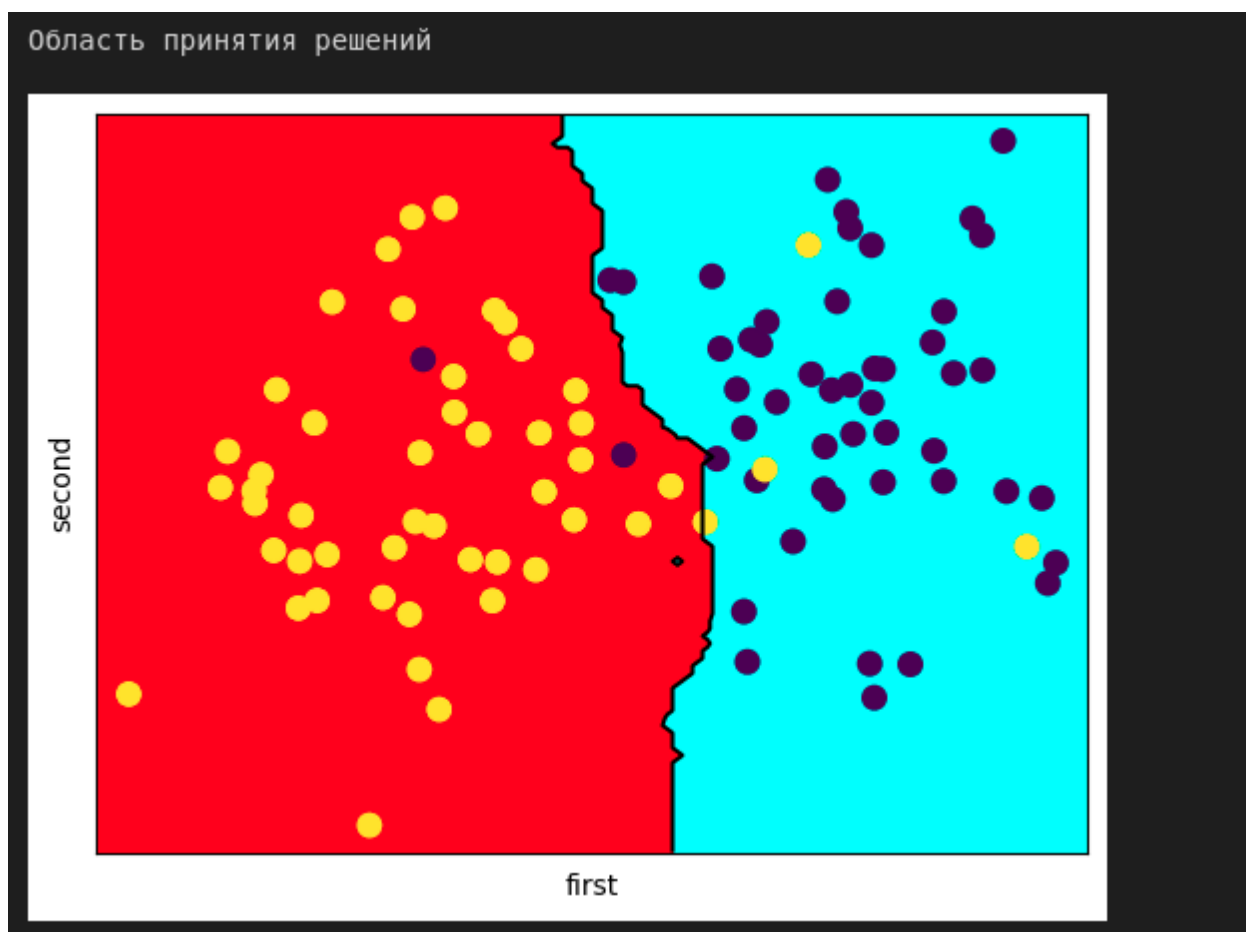


Рисунок 20 - Визуализация данных полученных при классификации

5.3.2 Случайный лес $n_estimators$ 10

Реализация классификации методом случайного леса с параметром $n_estimators = 10$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 21 - 22.

Случайный лес n_estimators 10

```
rand_forest = RandomForestClassifier(n_estimators=10)
rand_forest.fit(train_X, train_Y)
prediction = rand_forest.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

[435] ✓ 0.2s

... Значения предсказанные правдивые тестовые

[1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]

Значения правдивые тестовые

[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]

Матрица классификации

[[13 0]

[2 10]]

Полученная точность классификации: 0.92

Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)

0.9166666666666667

Рисунок 21 - Метод случайного леса n_estimators 10

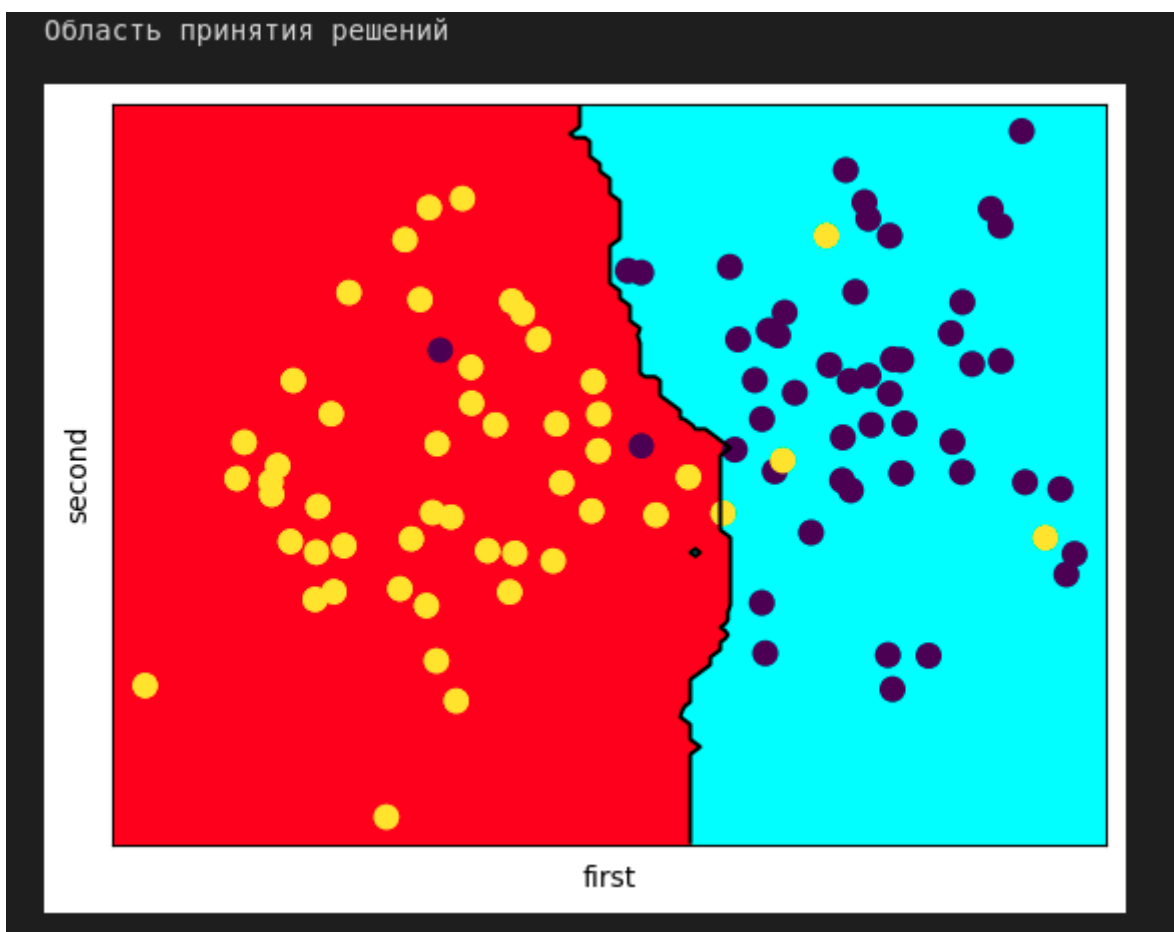


Рисунок 22 - Визуализация данных полученных при классификации

5.3.3 Случайный лес $n_estimators$ 15

Реализация классификации методом случайного леса с параметром $n_estimators = 15$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 23 - 24.

Случайный лес n_estimators 15

```
rand_forest = RandomForestClassifier(n_estimators=15)
rand_forest.fit(train_X, train_Y)
prediction = rand_forest.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

✓ 0.2s

Значения предсказанные правдивые тестовые

[1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]

Значения правдивые тестовые

[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]

Матрица классификации

[[13 0]

[2 10]]

Полученная точность классификации: 0.92

Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)

0.9166666666666667

Рисунок 23 - Метод случайного леса n_estimators 15

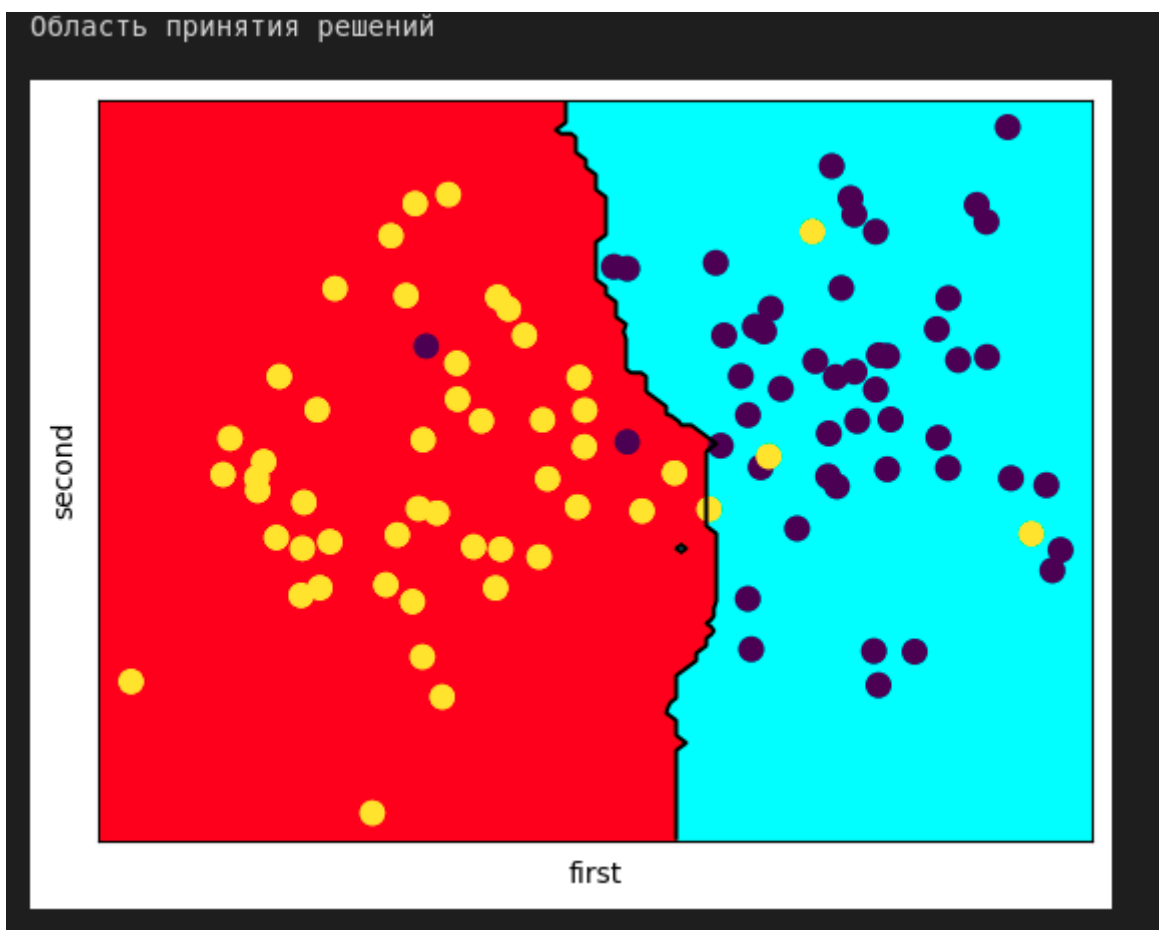


Рисунок 24 - Визуализация данных полученных при классификации

5.3.4 Случайный лес $n_estimators$ 20

Реализация классификации методом случайного леса с параметром $n_estimators = 20$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 25 - 26.

Случайный лес n_estimators 20

```
rand_forest = RandomForestClassifier(n_estimators=20)
rand_forest.fit(train_X, train_Y)
prediction = rand_forest.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

✓ 0.2s

Значения предсказанные правдивые тестовые

[1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]

Значения правдивые тестовые

[1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]

Матрица классификации

[[13 0]

[2 10]]

Полученная точность классификации: 0.92

Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)

0.9166666666666667

Рисунок 25 - Метод случайного леса n_estimators 20

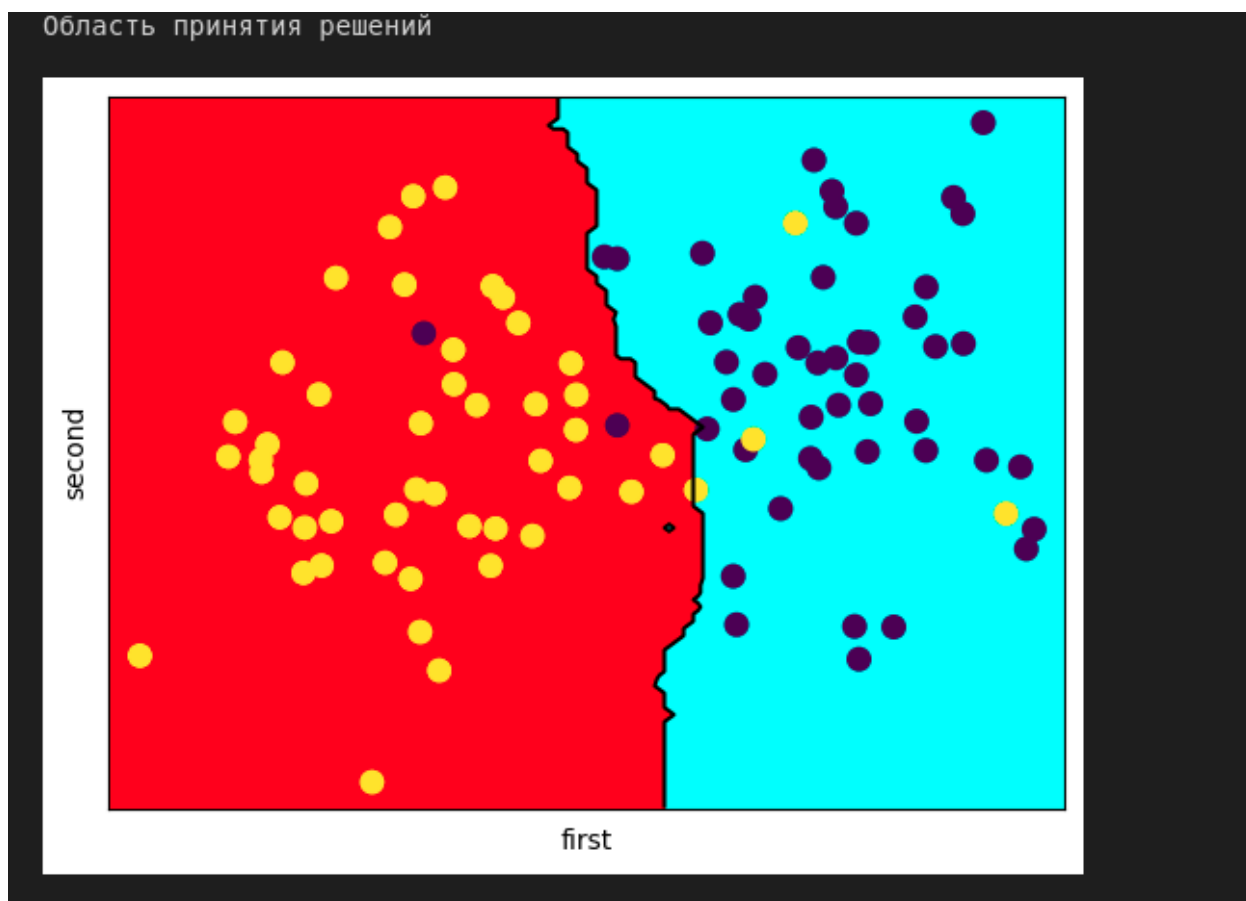


Рисунок 26 - Визуализация данных полученных при классификации

5.3.5 Случайный лес $n_estimators$ 50

Реализация классификации методом случайного леса с параметром $n_estimators = 50$, полученные характеристики классификации и визуальное представление данных представлены на рисунках 27 - 28.

Случайный лес n_estimators 50

```
rand_forest = RandomForestClassifier(n_estimators=50)
rand_forest.fit(train_X, train_Y)
prediction = rand_forest.predict(test_X)
print_classification_metrics(classifier, X, y, prediction, test_Y)
```

✓ 0.2s

Значения предсказанные правдивые тестовые

```
[1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]
```

Значения правдивые тестовые

```
[1 1 0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]
```

Матрица классификации

```
[[13  0]
 [ 2 10]]
```

Полученная точность классификации: 0.92

Значения полноты, точности, f1-меры и аккуратности

	precision	recall	f1-score	support
0	0.87	1.00	0.93	13
1	1.00	0.83	0.91	12
accuracy			0.92	25
macro avg	0.93	0.92	0.92	25
weighted avg	0.93	0.92	0.92	25

Значение ошибок (AUC ROC)

0.9166666666666667

Рисунок 27 - Метод случайного леса n_estimators 50

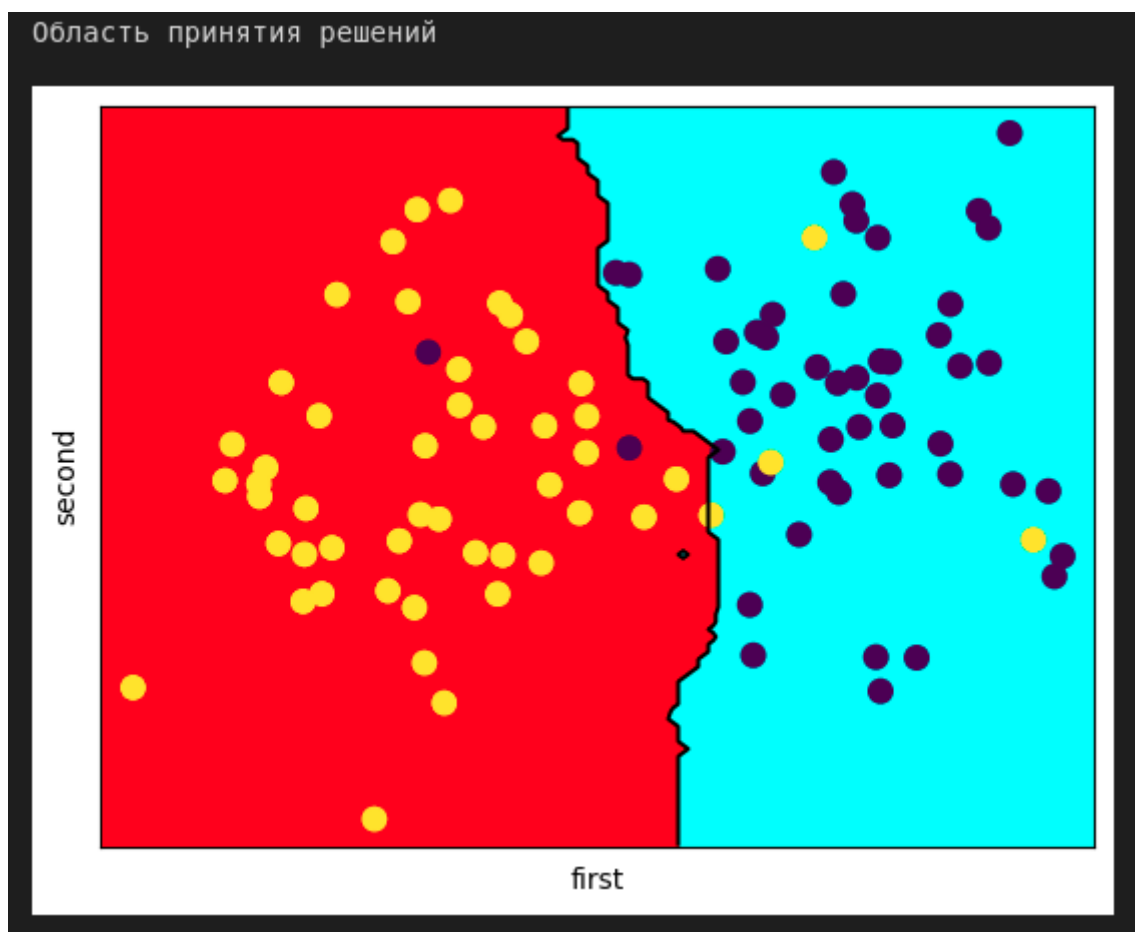


Рисунок 28 - Визуализация данных полученных при классификации

5.4 Результаты классификации

Сведем полученные оценки классификации в общую таблицу, для анализа лучшего классификатора по заданным данным.

Таблица 1 - Полученные результаты классификации

Метод	Точность	Значение ошибки	Точность классификации класса	Чувствительность	Специфичность
Метод k-ближайших соседей n_neighbors = 1	0,88	0,875	0,81 1,00	1,00 0,75	0,75 1,00
Метод k-ближайших соседей n_neighbors = {3,5,9}	0,92	0,917	0,87 1,00	1,00 0,83	0,83 1,00
Наивный байесовский метод	0,92	0,917	0,87 1,00	1,00 0,83	0,83 1,00
Случайный лес n_estimators 5	0,8	0,792	0,72 1,00	1,00 0,58	0,58 1,00
Случайный лес n_estimators = {10,15,20,50}	0,92	0,917	0,87 1,00	1,00 0,83	0,83 1,00

По данным таблицы 1 можно сделать вывод, что для классификации данного набора данных хорошо подходят методы:

- Метод k-ближайших соседей при `n_neighbors` 3,5,9;
- Наивный байесовский метод;
- Случайный лес `n_estimators` при 10,15,20,50.

Все данные методы показали одинаковую точность и остальные характеристики. Методы k-ближайших соседей `n_neighbors` = 1 и случайный лес `n_estimators` 5 показали самые низкие результаты.

5.5 Результаты при выделении на тестовую выборку 10% выборки
Проделаем пункты 5.1-5.4, но изменим тестовую выборку, взяв 10% от изначальной.

Результаты представлены в таблице 2.

Таблица 2 - Результаты классификации при 10% тестовой выборке

Метод	Точность	Значение ошибки	Точность классификации класса	Чувствительность	Специфичность
Метод k-ближайших соседей	0,90	0,90	0,83 1,00	1,00 0,80	0,80 1,00
Наивный байесовский метод	0,90	0,90	0,83 1,00	1,00 0,80	0,80 1,00
Случайный лес	0,90	0,90	0,83 1,00	1,00 0,80	0,80 1,00

Мы видим, что при увеличении обучающей выборки все классификаторы показали одинаковый результат классификации, из-за чего нельзя выбрать наилучшего представителя.

5.6 Результаты при выделении на тестовую выборку 35% выборки

По аналогии с пунктом 5.5 сделаем тоже самое, но объем тестовой выборки сделаем 35%. Результаты занесем в таблицу 3.

Таблица 3 - Результаты классификации при 35% тестовой выборке

Метод	Точность	Значение ошибки	Точность классификации класса	Чувствительность	Специфичность
Метод k-ближайших соседей n_neighbors = 1	0,89	0,88	0,82 1,00	1,00 0,76	0,76 1,00
Метод k-ближайших соседей n_neighbors = {3,5,9}	0,91	0,91	0,86 1,00	1,00 0,82	1,00 0,82
Наивный байесовский метод	0,91	0,91	0,86 1,00	1,00 0,82	1,00 0,82
Случайный лес	0,91	0,91	0,86 1,00	1,00 0,82	1,00 0,82

Из полученных результатов видно, что общие показатели уменьшаются, что говорит об ухудшении классификации при уменьшении обучающей выборки.

Вывод

В ходе выполнения данной лабораторной работы были получены практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook, загрузки данных, обучения классификаторов и проведение классификации.

Мы узнали, что кол-во данных в обучающей выборке влияет на показатели классификации данных. Были изучены и применены на практике такие классификаторы, как:

- Метод к-ближайших соседей ($n_neighbors = \{1, 3, 5, 9\}$)
- Наивный байесовский метод
- Случайный лес ($n_estimators = \{5, 10, 15, 20, 50\}$)