**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 Бахмутский М.В.

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 к-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: 1

| **Вариант** | **1** |
|---|---|
| Вид классов | blobs |
| Random_state | 34 |
| cluster_std | 1.5 |
| noise | - |
| Centers | 2 |

Ход работы

1. 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

```
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.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

Рисунок 2 - Необходимые зависимости

2. Загрузка данных

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

| Вид класса | random_state | cluster_std | noise | centers |
|---|---|---|---|---|
| blobs | 34 | 1.5 | - | 2 |

```
X, y = make_blobs(centers=2, random_state=34, cluster_std=1.5)
```

```
print("Координаты точек:\n", X[:15])
print("Метки класса: ", y[:15])
```

```
Координаты точек:
 [[ -9.4383967    2.85736287]
 [ -8.51188005   6.27823641]
 [ -9.13949465   3.63954101]
 [-12.37089043   7.15594338]
 [ -8.91150743   5.93048649]
 [ -9.54177266   3.84185898]
 [-10.90214655   2.04849828]
 [ -9.50825238   7.74690079]
 [ -7.00867273   3.01451428]
 [ -9.70678044   7.57851126]
 [-10.37004706   3.06762603]
 [ -7.36974408   2.34416931]
 [ -9.27756446   3.32067295]
 [ -6.7484923    3.58902355]
 [-10.17439185   2.344658  ]]
Метки класса:  [1 0 0 0 0 1 1 0 1 0 1 1 1 1 1]
```

Рисунок 3 - Процесс загрузки данных

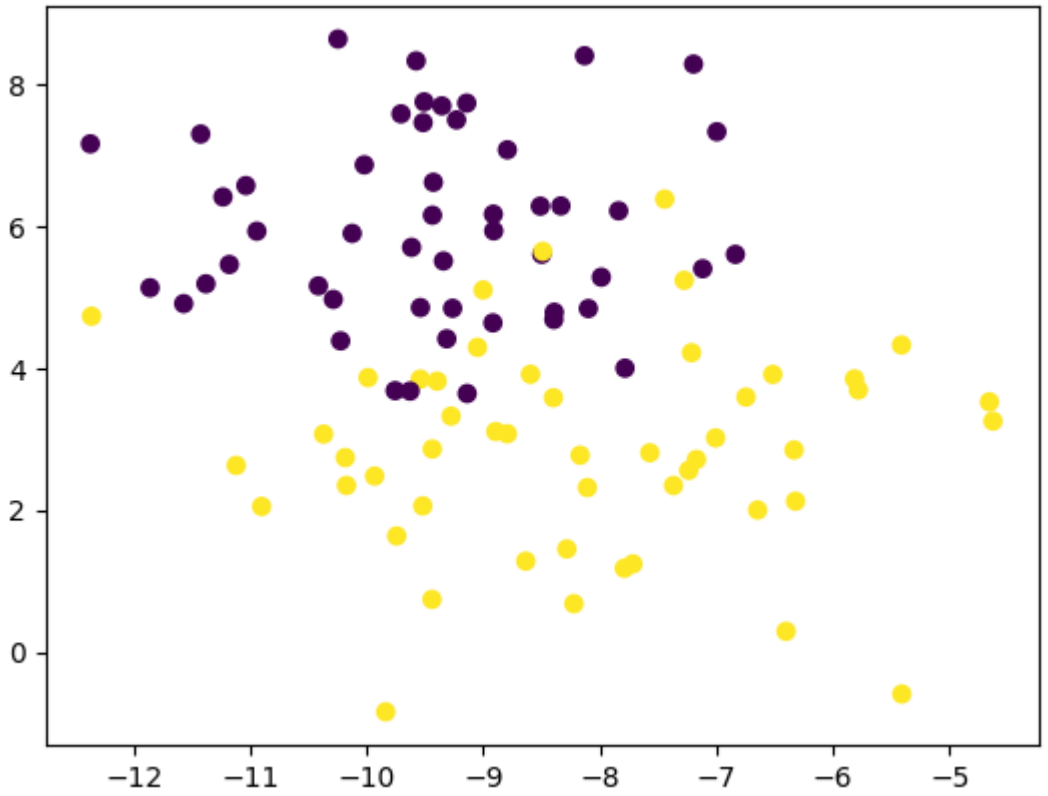


Рисунок 4 - Визуализация данных

```
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(())
```

Рисунок 5 - Функция для визуализации данных

3. 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.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state=1)
```

Рисунок 5 - Разбиение данных

```
plt.scatter(X_train[:,0], X_train[:,1], c=y_train)
```

```
<matplotlib.collections.PathCollection at 0x23585503950>
```

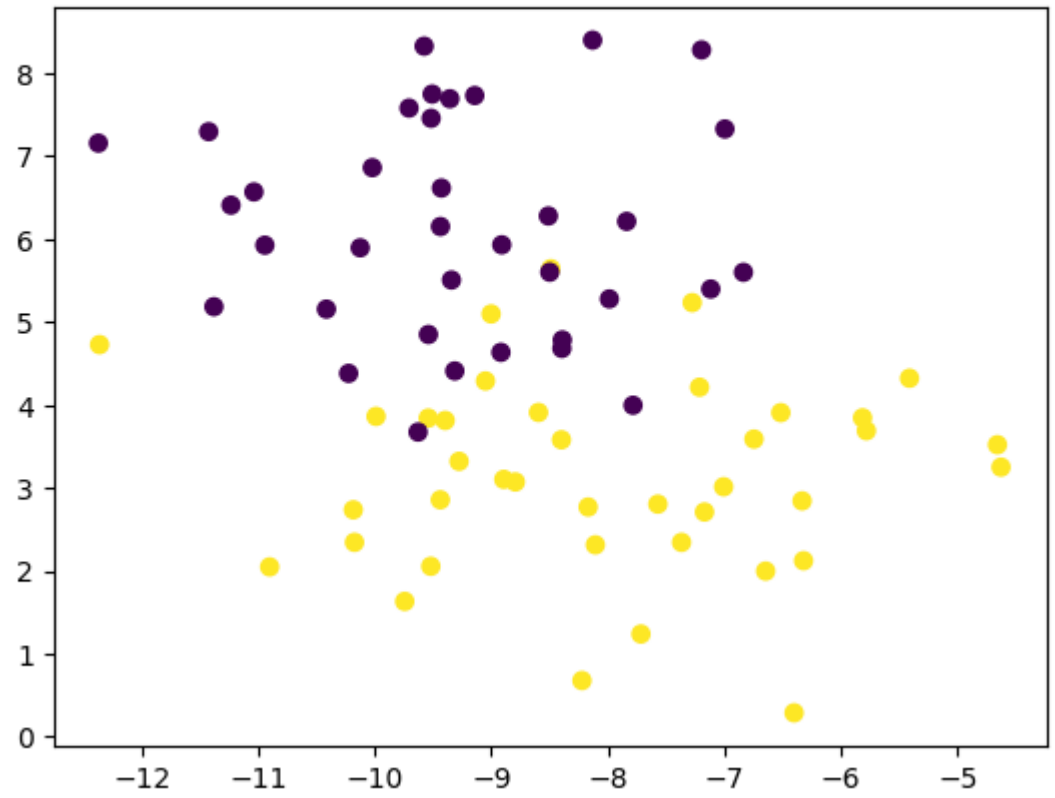


Рисунок 6 - Обучающая выборка

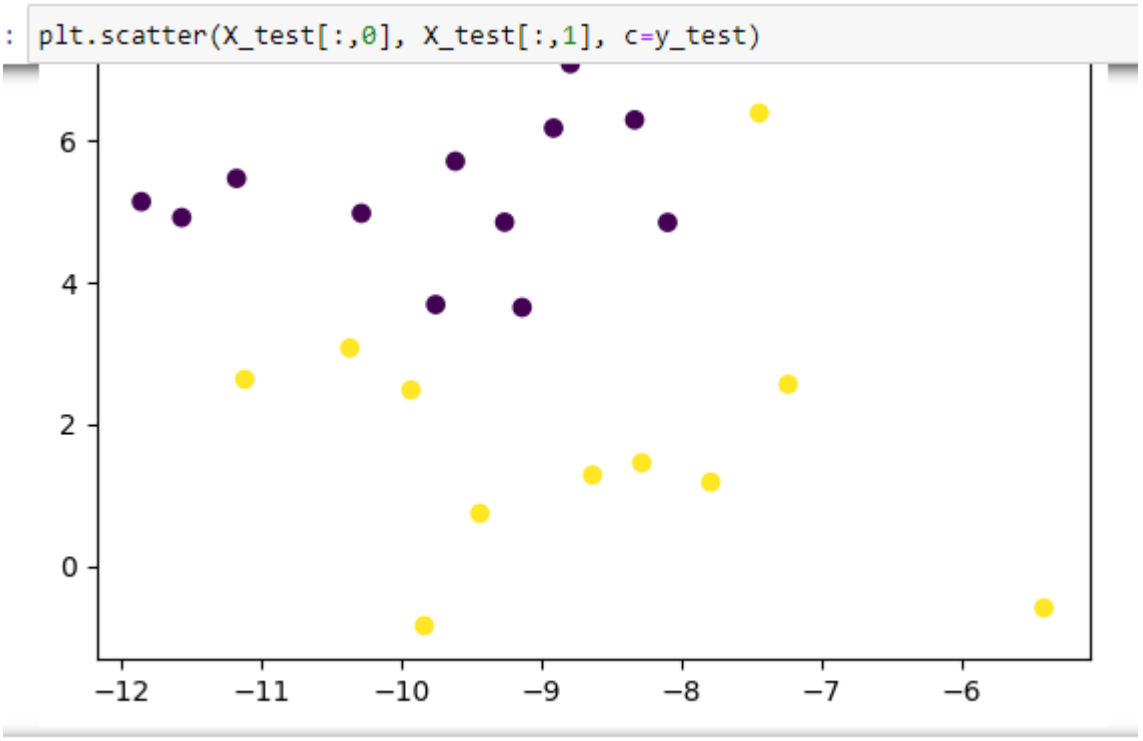


Рисунок 7 - Тестовая выборка

4. Классификация данных

Создадим универсальную функция, 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, y_test):
    print("Предстказанные и истинные значения")
    print(prediction)
    print(y_test)

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

Рисунок 8 - Реализация функции для оценки классификации

4.1 Метод к-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.

4.1.1 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.

```
for i in [1, 3, 5, 9]:
    knn = KNeighborsClassifier(n_neighbors=i, metric='euclidean')
    knn.fit(X_train, y_train)
    prediction = knn.predict(X_test)
    print("n_neightbors = ", i)
    print_classification_metrics(knn, X, y, prediction, y_test)
```

```
n_neightbors =  1
Предстказанные и истинные значения
[0 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[13  1]
 [ 1 10]]
Точность классификации:  0.92
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.91      0.91      0.91        11

    accuracy                           0.92        25
   macro avg       0.92      0.92      0.92        25
weighted avg       0.92      0.92      0.92        25
```

Рисунок 9 - Метод k-ближайших соседей n_neighbors = 1

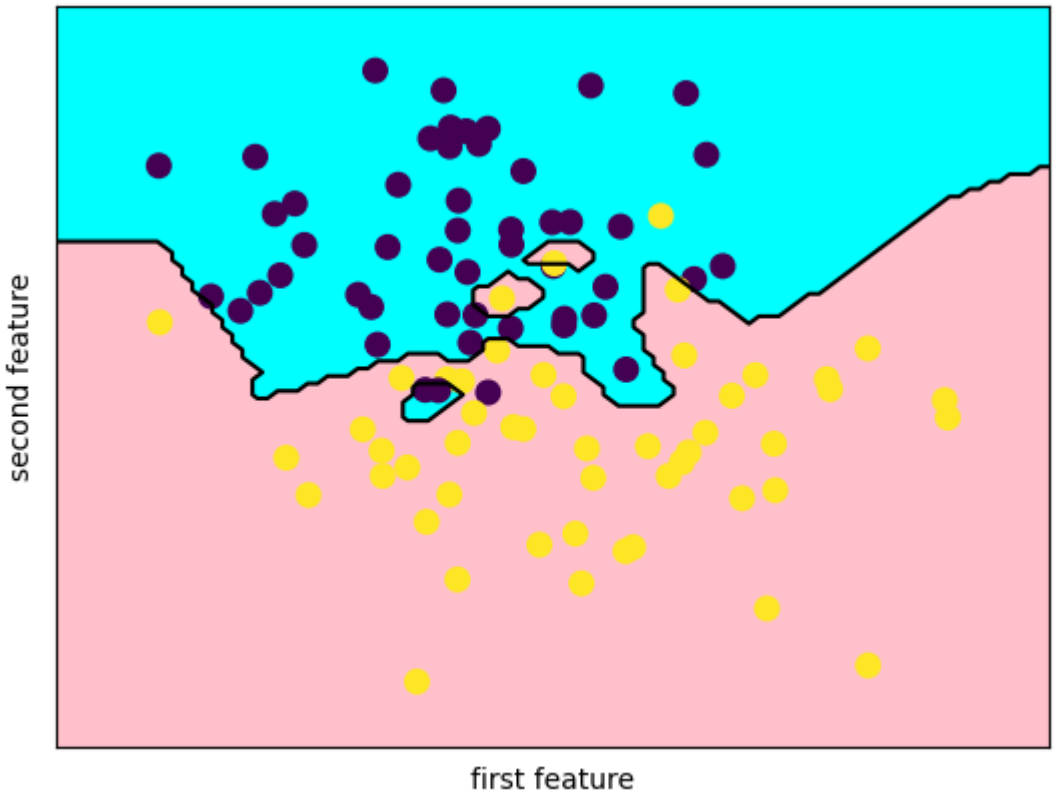


Рисунок 10 - Визуализация данных полученных при классификации

4.1.2 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.

```
n_neighbors =  3
Предстказанные и истинные значения
[1 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[12  2]
 [ 1 10]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        14
           1       0.83      0.91      0.87        11

    accuracy                           0.88        25
   macro avg       0.88      0.88      0.88        25
weighted avg       0.88      0.88      0.88        25

Значение площади под кривой ошибок (AUC ROC)
0.8831168831168832
```

Рисунок 11 - Метод k-ближайших соседей n_neighbors = 3

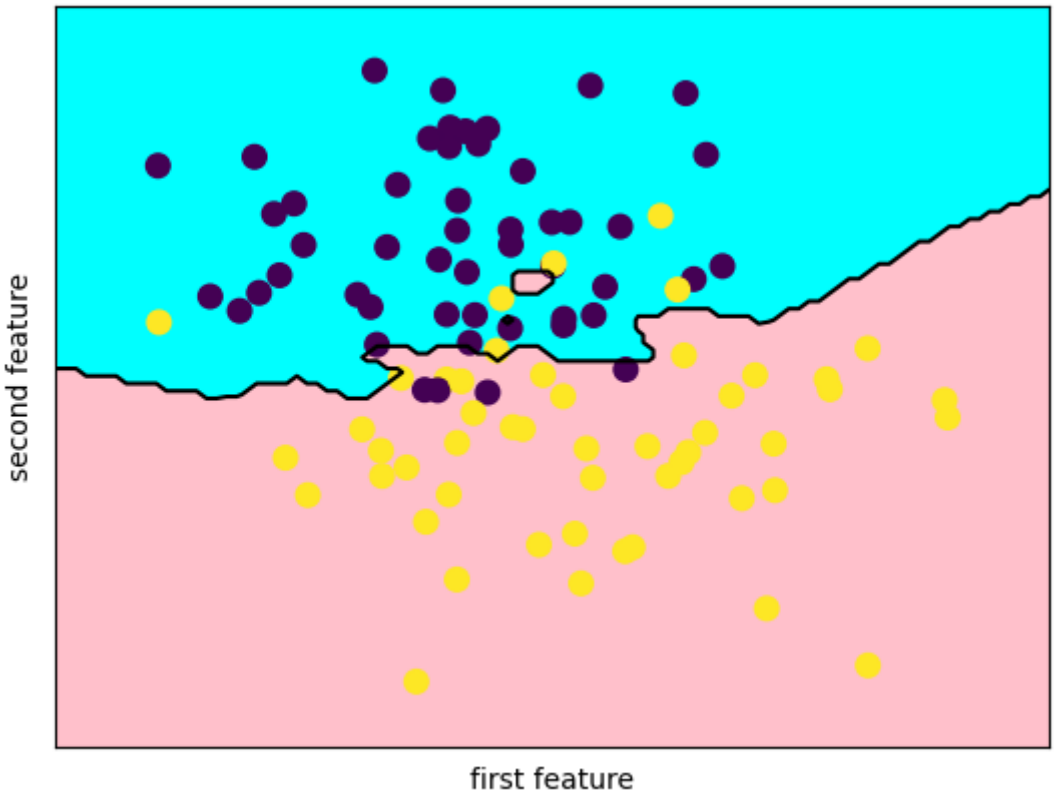


Рисунок 12 - Визуализация данных полученных при классификации

4.1.3 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.

```
n_neightbors =  5
Предстказанные и истинные значения
[1 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[12  2]
 [ 1 10]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        14
           1       0.83      0.91      0.87        11

    accuracy                           0.88        25
   macro avg       0.88      0.88      0.88        25
weighted avg       0.88      0.88      0.88        25

Значение площади под кривой ошибок (AUC ROC)
0.8831168831168832
```

Рисунок 13 - Метод k-ближайших соседей n_neighbors = 5

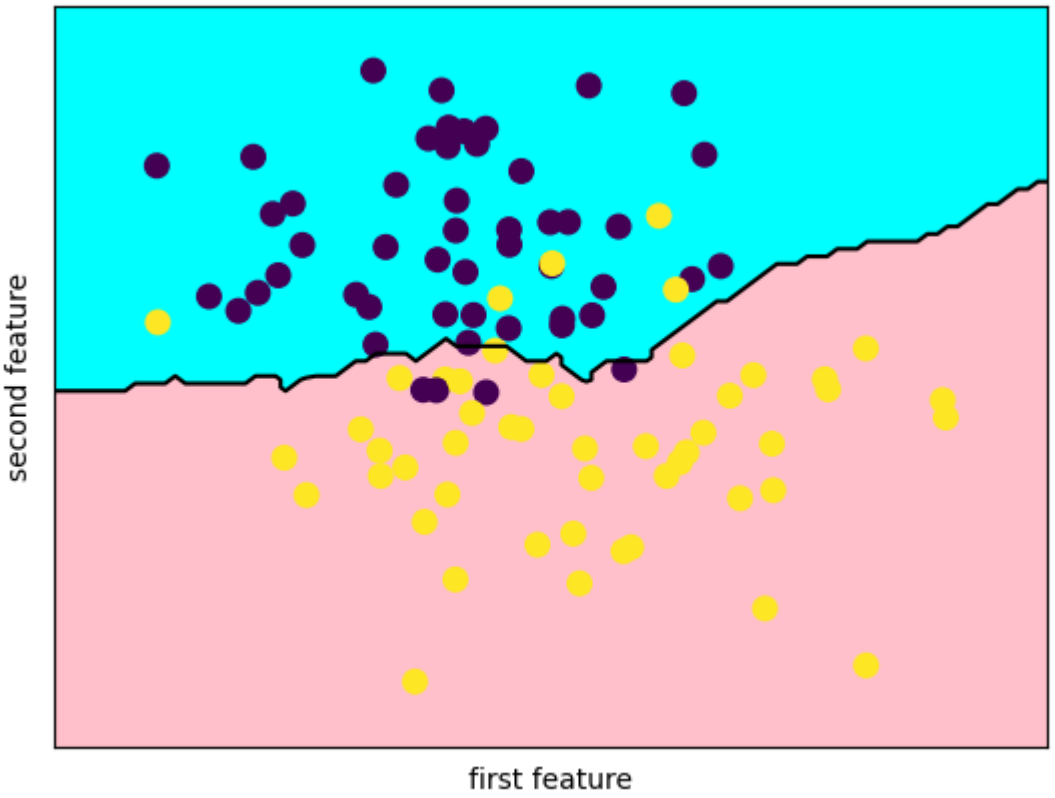


Рисунок 14 - Визуализация данных полученных при классификации

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.

```
Предстказанные и истинные значения
[1 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[12  2]
 [ 1 10]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        14
           1       0.83      0.91      0.87        11

    accuracy                           0.88        25
   macro avg       0.88      0.88      0.88        25
weighted avg       0.88      0.88      0.88        25

Значение площади под кривой ошибок (AUC ROC)
0.8831168831168832
```

Рисунок 15 - Метод k-ближайших соседей n_neighbors = 9

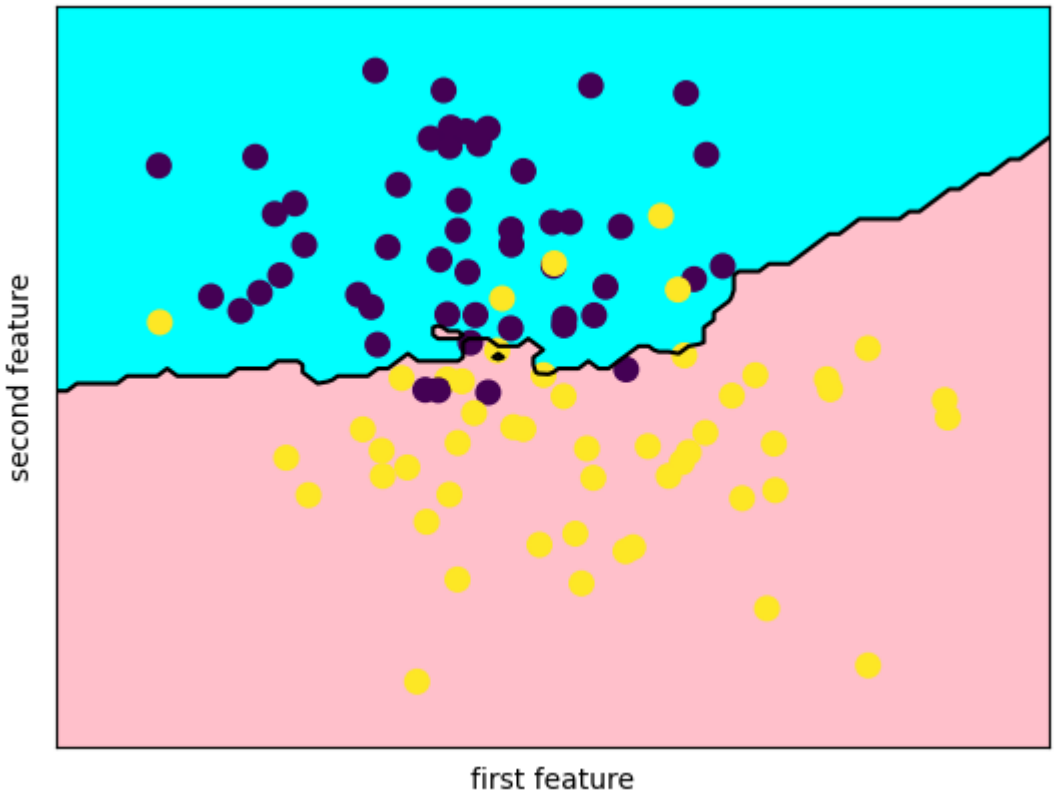


Рисунок 16 - Визуализация данных полученных при классификации

4.2 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(X_train, y_train)
predict = naive.predict(X_test)
print_classification_metrics(naive, X, y, predict, y_test)
```

```
Предстказанные и истинные значения
[1 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[12  2]
 [ 1 10]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        14
           1       0.83      0.91      0.87        11

    accuracy                           0.88        25
   macro avg       0.88      0.88      0.88        25
weighted avg       0.88      0.88      0.88        25

Значение площади под кривой ошибок (AUC ROC)
0.8831168831168832
```

Рисунок 17 - Наивный байесовский метод

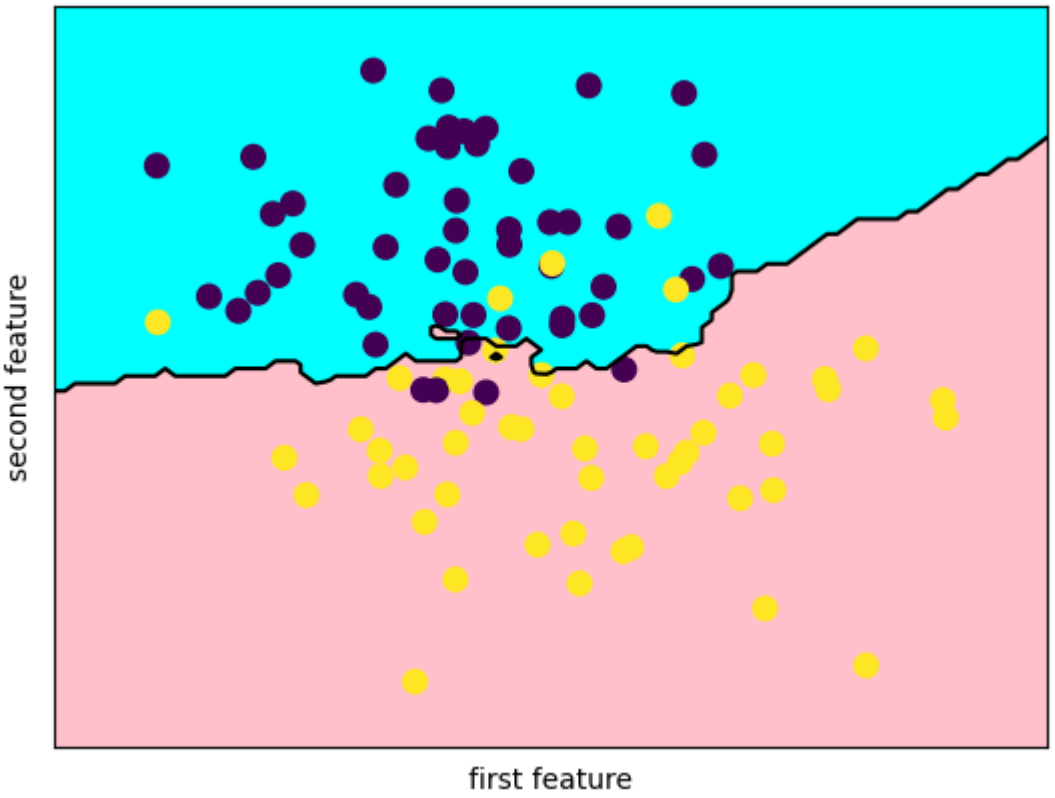


Рисунок 18 - Визуализация данных полученных при классификации

4.3 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.

4.3.1 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.

```
for i in [5, 10, 15, 20, 50]:
    rand_forest = RandomForestClassifier(n_estimators=i)
    rand_forest.fit(X_train, y_train)
    prediction = rand_forest.predict(X_test)
    print("n_estimators = ", i)
    print_classification_metrics(knn, X, y, prediction, y_test)
```

```
Предстказанные и истинные значения
[1 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[12  2]
 [ 1 10]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        14
           1       0.83      0.91      0.87        11

    accuracy                           0.88        25
   macro avg       0.88      0.88      0.88        25
weighted avg       0.88      0.88      0.88        25

Значение площади под кривой ошибок (AUC ROC)
0.8831168831168832
```

Рисунок 19 - Метод случайного леса n_estimators 5

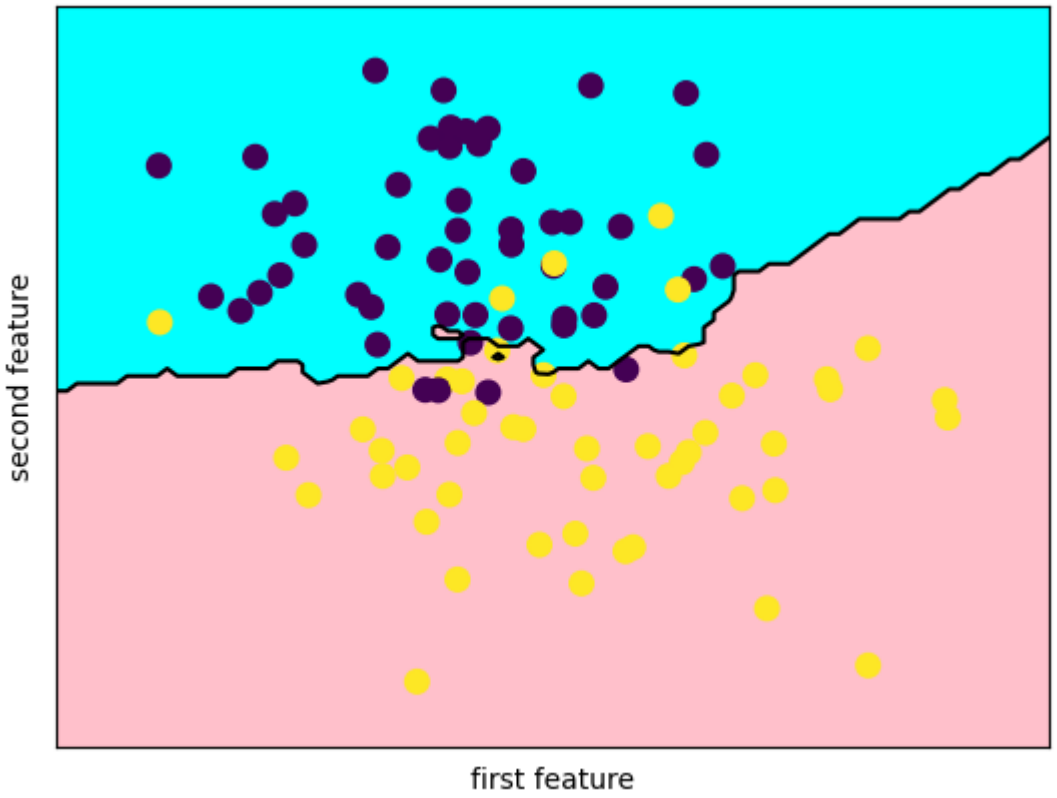


Рисунок 20 - Визуализация данных полученных при классификации

4.3.2 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.

```
Предстказанные и истинные значения
[1 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[12  2]
 [ 1 10]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        14
           1       0.83      0.91      0.87        11

    accuracy                           0.88        25
   macro avg       0.88      0.88      0.88        25
weighted avg       0.88      0.88      0.88        25

Значение площади под кривой ошибок (AUC ROC)
0.8831168831168832
```

Рисунок 21 - Метод случайного леса n_estimators 10

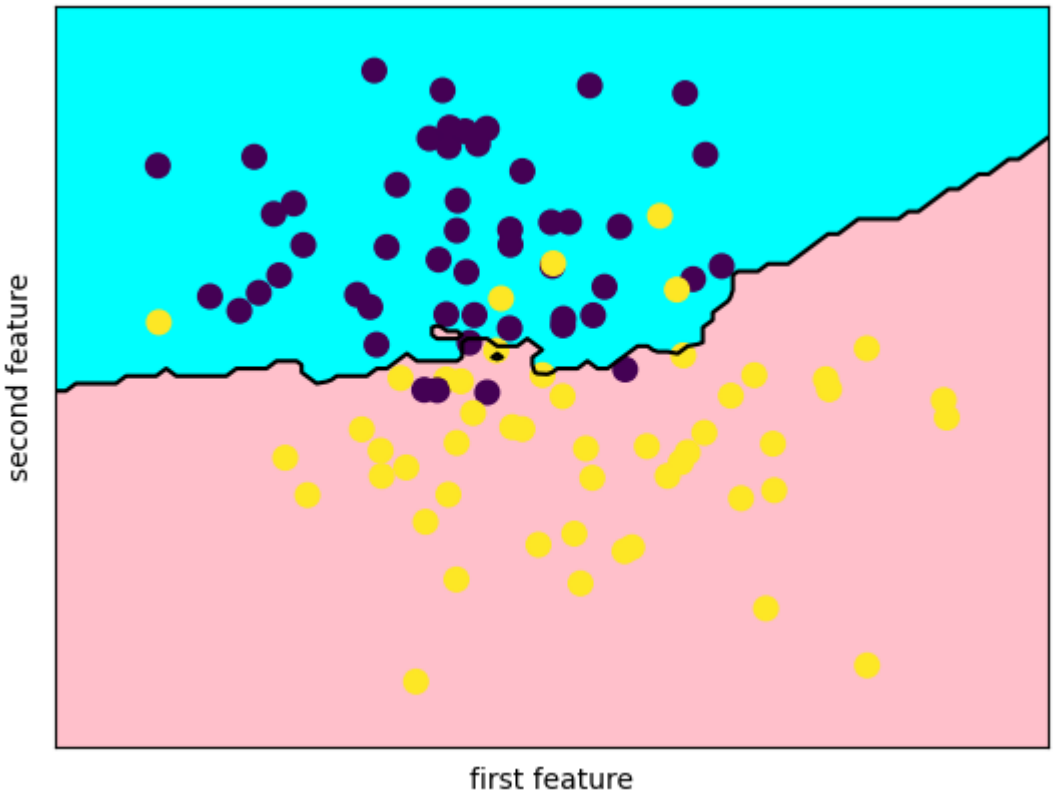


Рисунок 22 - Визуализация данных полученных при классификации

4.3.3 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.

```
Предстказанные и истинные значения
[0 0 1 0 0 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1]
[0 0 1 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1]
Матрица ошибок
[[13  1]
 [ 1 10]]
Точность классификации:  0.92
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.91      0.91      0.91        11

    accuracy                           0.92        25
   macro avg       0.92      0.92      0.92        25
weighted avg       0.92      0.92      0.92        25

Значение площади под кривой ошибок (AUC ROC)
0.9188311688311689
```

Рисунок 23 - Метод случайного леса n_estimators 15

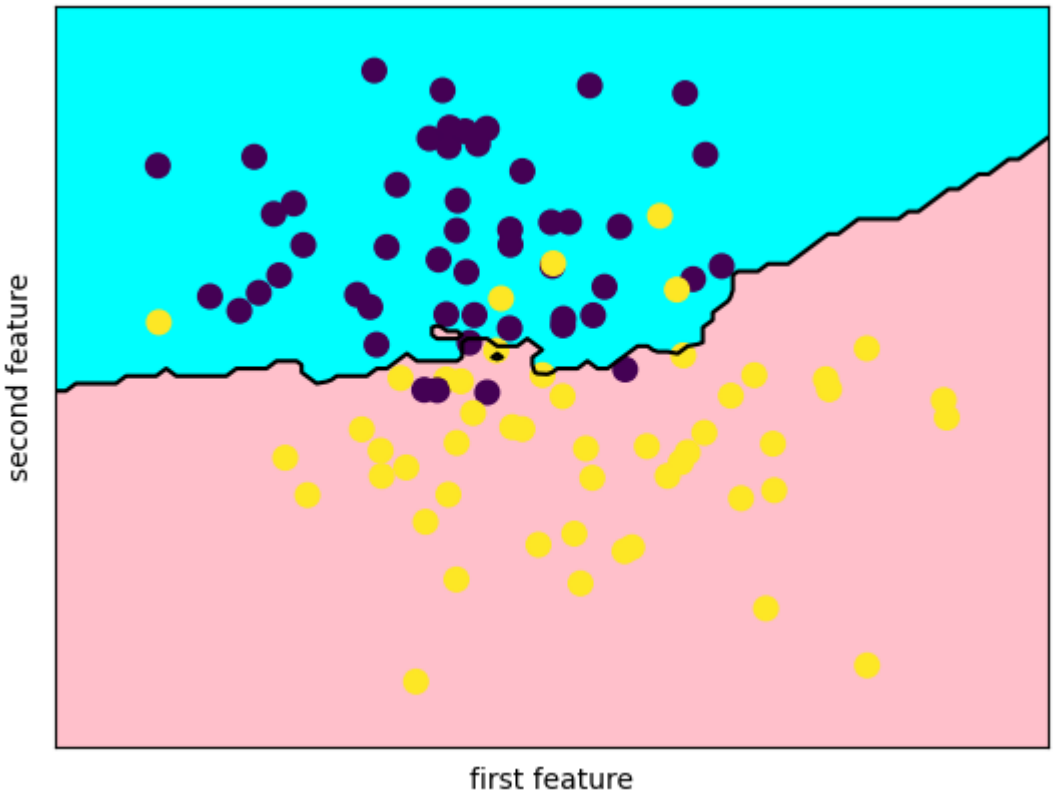


Рисунок 24 - Визуализация данных полученных при классификации

4.3.4 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.

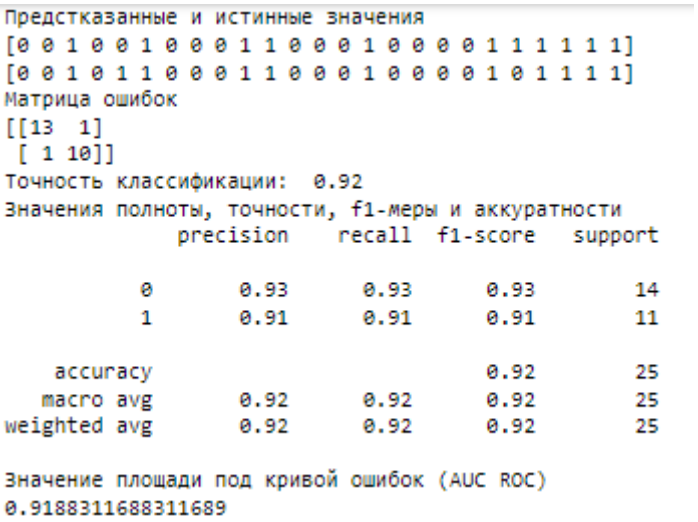


Рисунок 25 - Метод случайного леса n_estimators 20

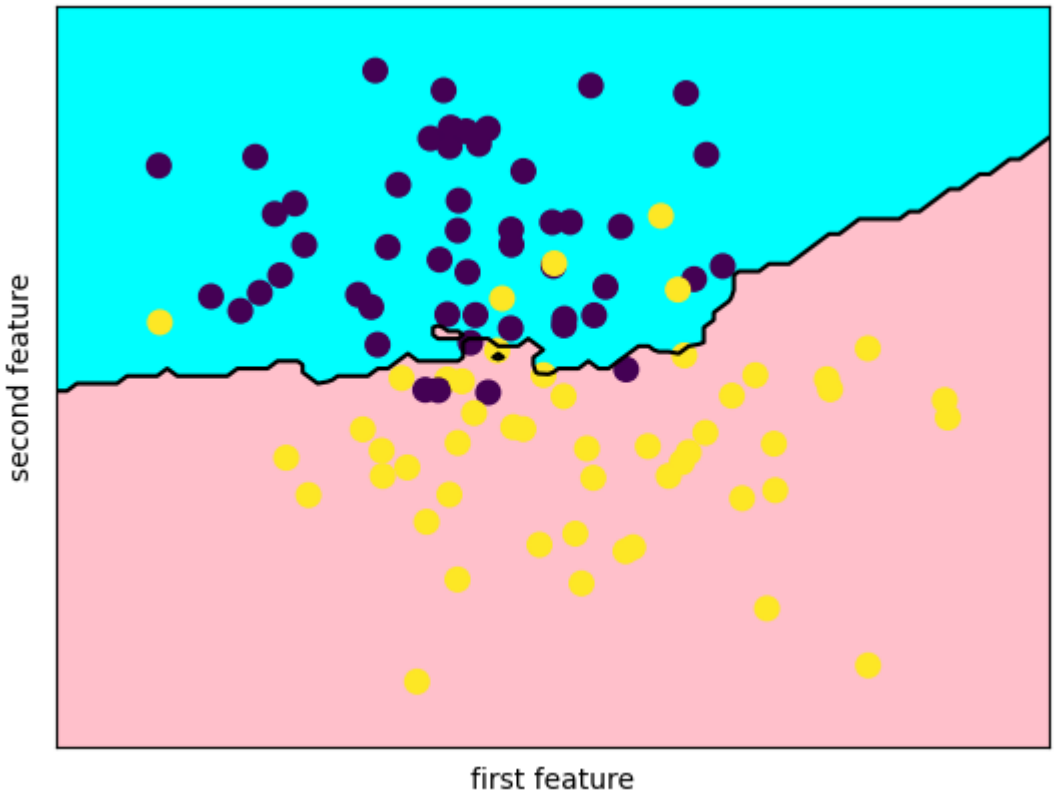


Рисунок 26 - Визуализация данных полученных при классификации

4.3.5 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.

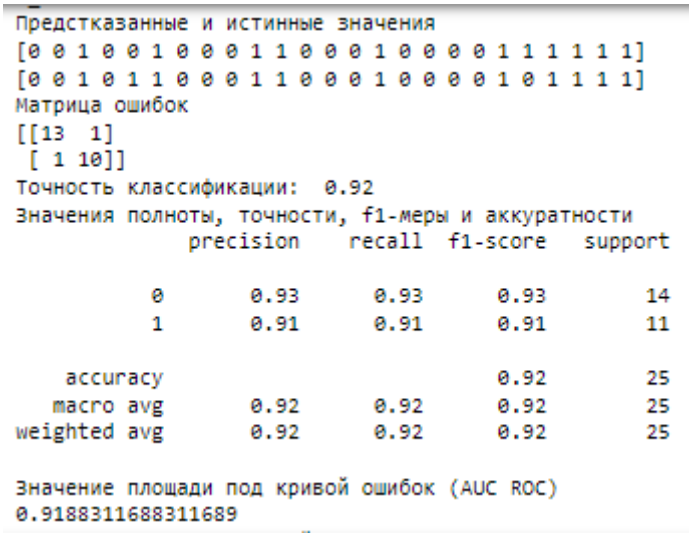


Рисунок 27 - Метод случайного леса n_estimators 50

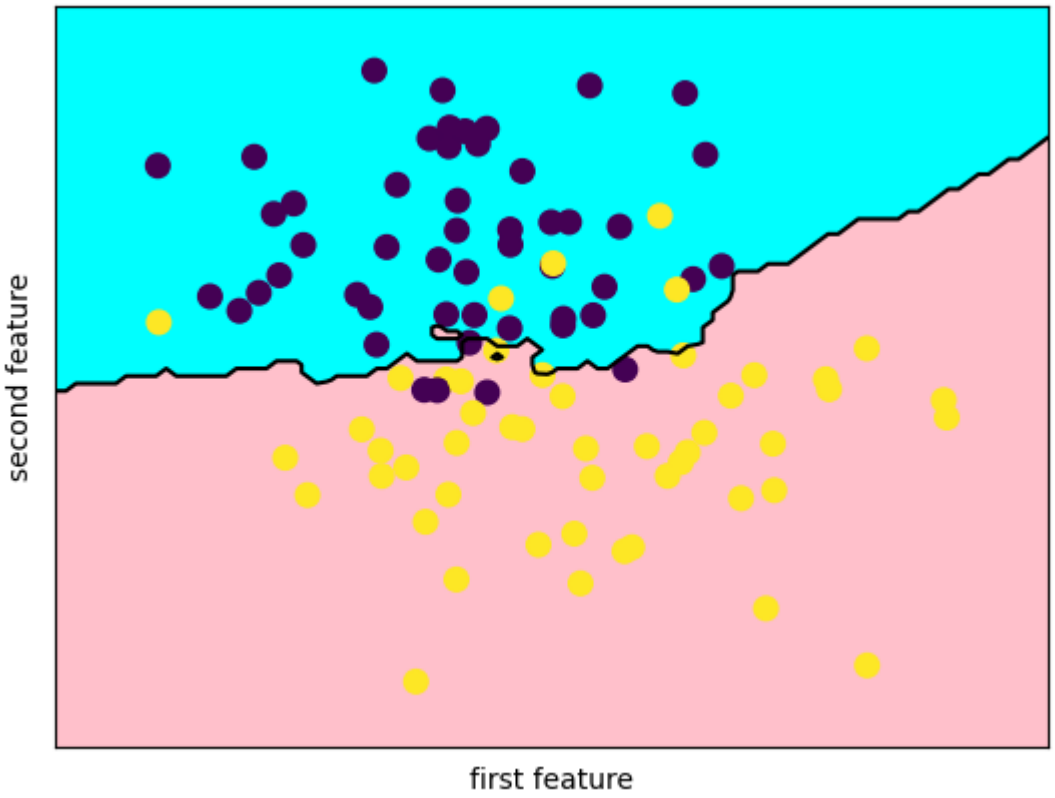


Рисунок 28 - Визуализация данных полученных при классификации

4.4 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 | 0,92 | 0,918 | 0,92<br>0,92 | 0,92<br>0,92 | 0,92<br>0,92 |
| Метод k-ближайших {3,5,9} | 0,88 | 0,88 | 0,88<br>0,88 | 0,88<br>0,88 | 0,88<br>0,88 |
| Наивный байесовский метод | 0,88 | 0,88 | 0,88<br>0,88 | 0,88<br>0,88 | 0,88<br>0,88 |
| Случайный лес n_estimators 5 | 0,88 | 0,88 | 0,88<br>0,88 | 0,88<br>0,88 | 0,88<br>0,88 |
| Случайный лес n_estimators = {10,15,20,50} | 0,92 | 0,91 | 0,92<br>0,92 | 0,92<br>0,92 | 0,92<br>0,92 |

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

— Метод k-ближайших соседей при n_neighbors 1;

— Случайный лес n_estimators при 10,15,20,50.

Все данные методы показали одинаковую точность и остальные характеристики. Методы k-ближайших соседей n_neighbors = 3,5,9, наивный байесовский метод и случайный лес n_estimators 5 показали самые низкие результаты.

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})