**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

«Прикладные интеллектуальные системы и экспертные системы»

Студент Бубырь Д. А.

Группа М-ИАП-23-1

Руководитель Кургасов В. В.

доцент, канд. пед. наук

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:

• 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.

Ход работы

На рисунке 1 представлены импортируемые необходимые библиотеки и модули для начальной работы программы. Остальные будут добавляться по мере выполнения работы.



Рисунок 1 - Импортируемые библиотеки и модули

Загружаем данные в соответствие с вариантом (вариант 2) и выводим первые 15 элементов выборки (рисунки 2-3).



Рисунок 2 – Код для загрузки и вывода данных

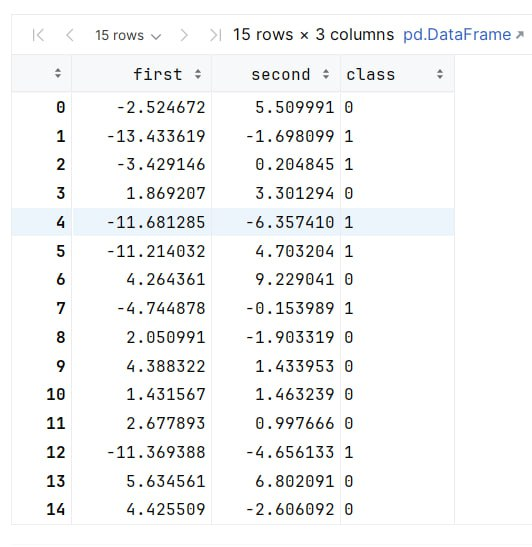


Рисунок 3 – Первые 15 элементов выборки

Отобразим на графике сгенерированную выборку (рисунок 4).

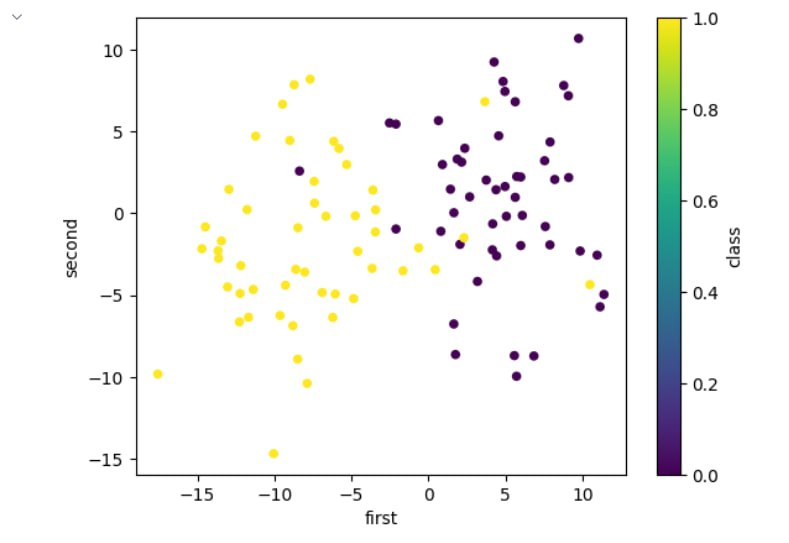


Рисунок 4 – Сгенерированная выборка

Разобьем данные на обучающую и тестовую выборки (рисунок 5).

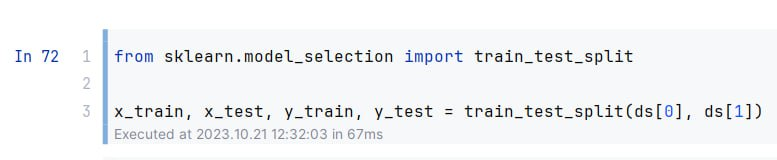


Рисунок 5 – Код для разбивки данных

Отобразим на графике обучающую выборку (рисунок 6).

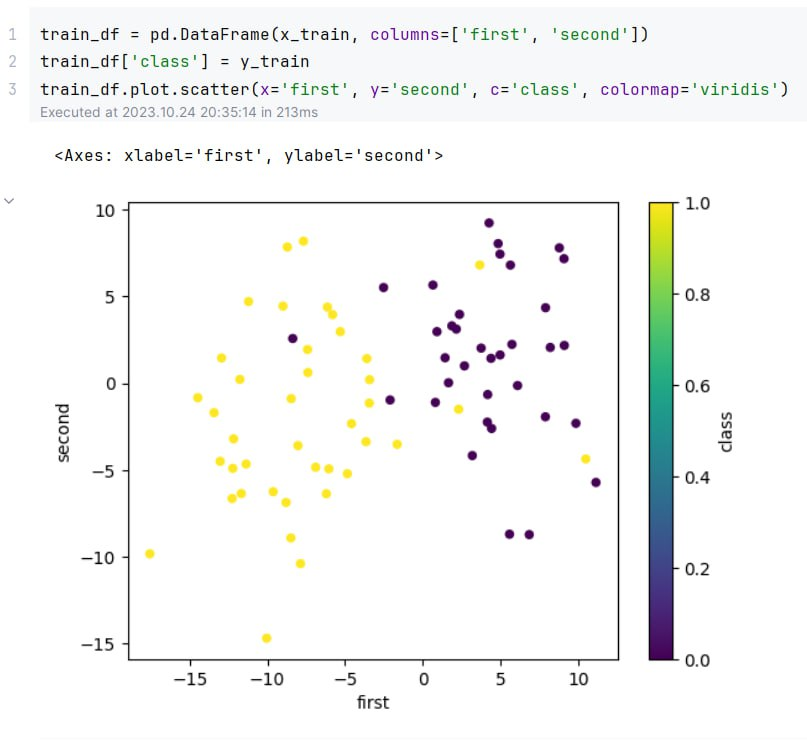


Рисунок 6 – Обучающая выборка (код и график)

Отобразим на графике тестовую выборку (рисунок 7).

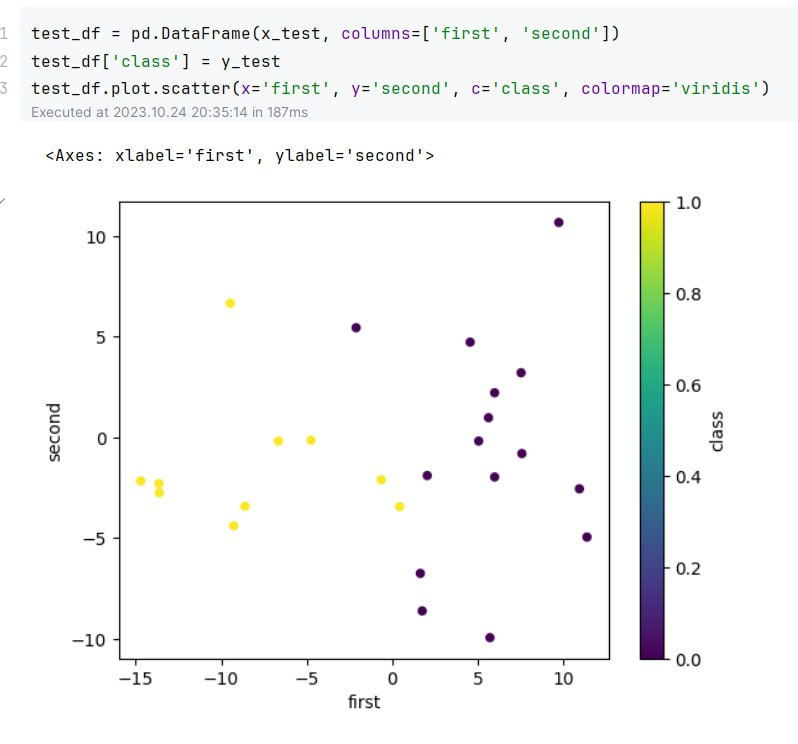


Рисунок 7 – Тестовая выборка (код и график)



Рисунок 8 – Функция для вывода результатов классификации

Напишем функцию (рисунок 9), реализующую метод k-ближних соседей с принимаемыми параметрами.



Рисунок 9 – Функция для метода k-ближних соседей

Выполним данную функцию с параметрами n\_neighbors = 1, 3, 5 и 9 и выведем результаты. Результаты представлены на рисунках 10-21.

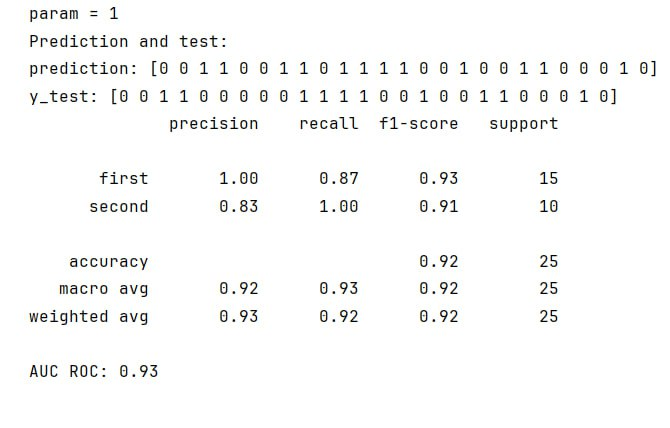


Рисунок 10 – Статистика по методу k-ближних соседей (n\_neighbors = 1)

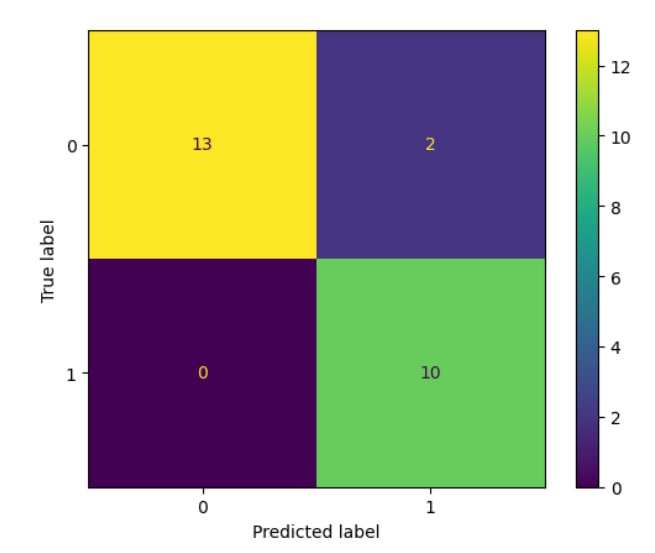


Рисунок 11 – Матрица ошибок по методу k-ближних соседей (n\_neighbors = 1)

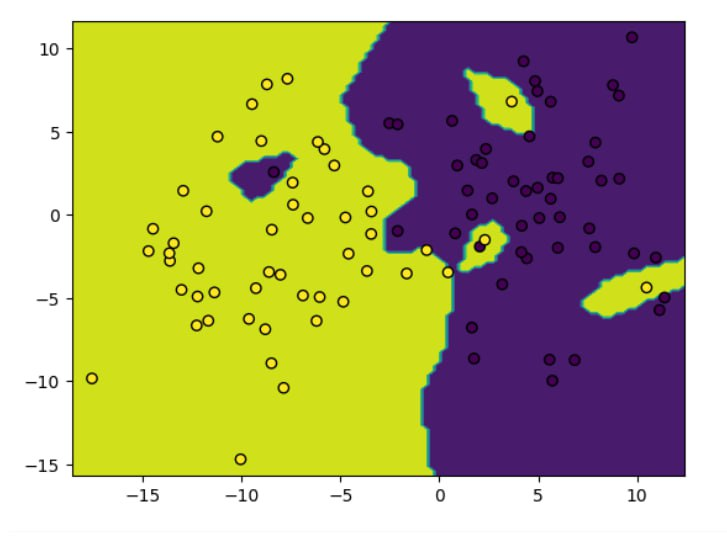


Рисунок 12 – Области принятия решений (метод k-ближних соседей (n\_neighbors = 1))

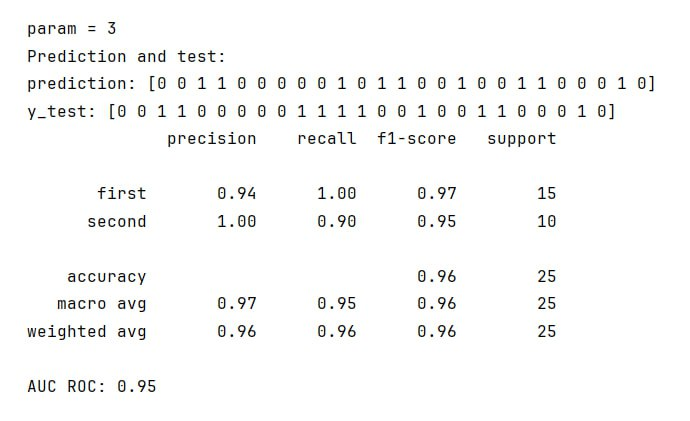


Рисунок 13 – Статистика по методу k-ближних соседей (n\_neighbors = 3)

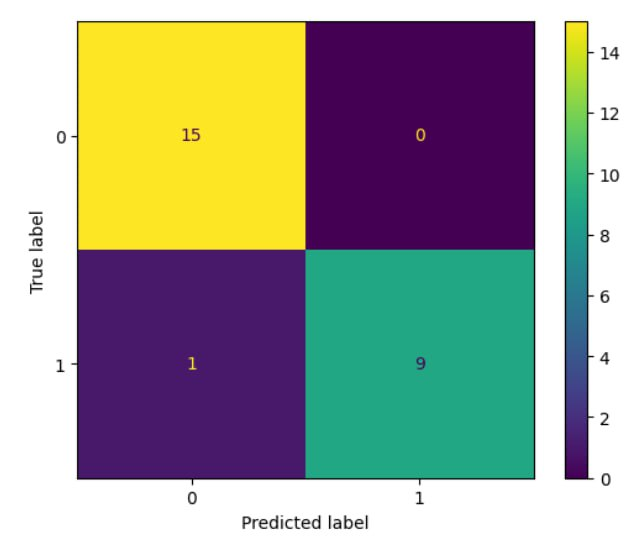


Рисунок 14 – Матрица ошибок по методу k-ближних соседей (n\_neighbors = 3)

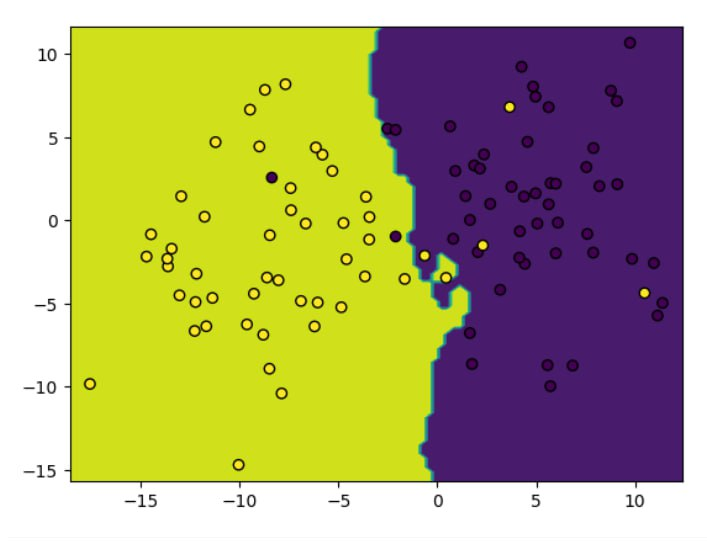


Рисунок 15 – Области принятия решений (метод k-ближних соседей (n\_neighbors = 3))

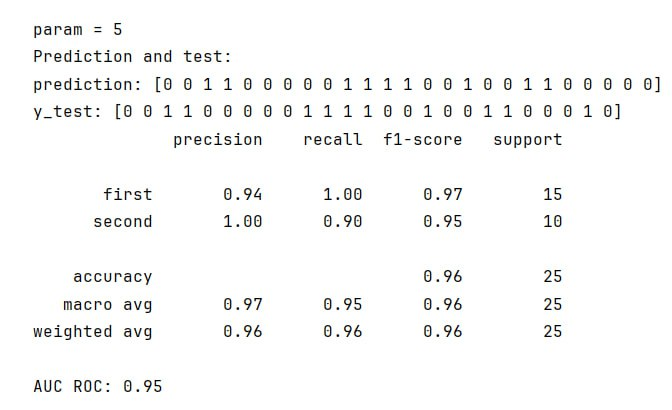


Рисунок 16 – Статистика по методу k-ближних соседей (n\_neighbors = 5)

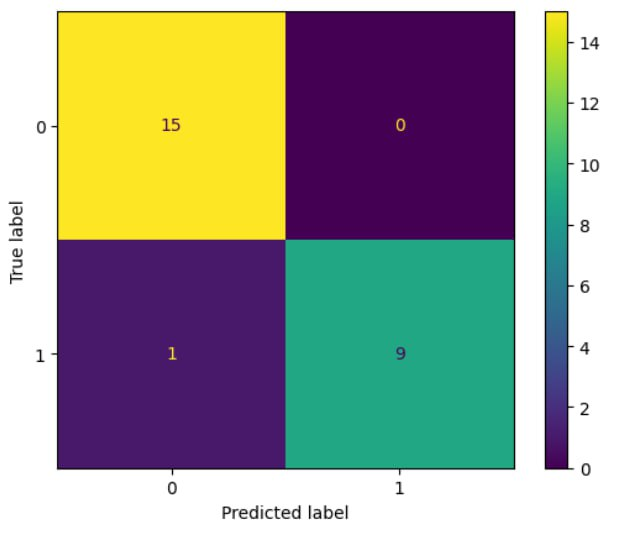


Рисунок 17 – Матрица ошибок по методу k-ближних соседей (n\_neighbors = 5)

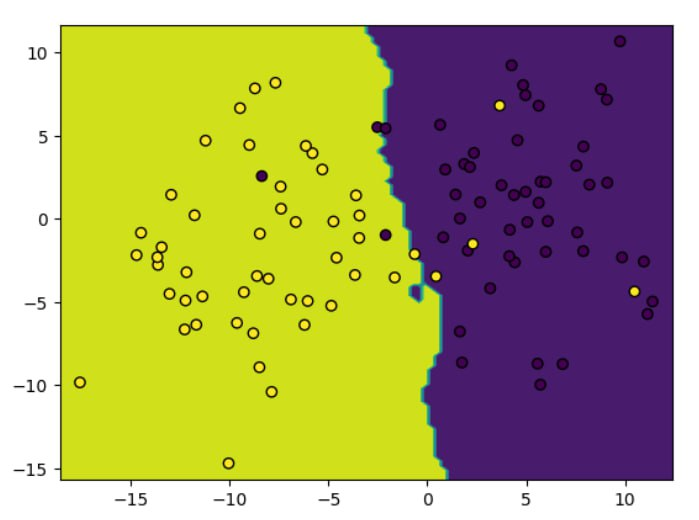


Рисунок 18 – Области принятия решений (метод k-ближних соседей (n\_neighbors = 5))

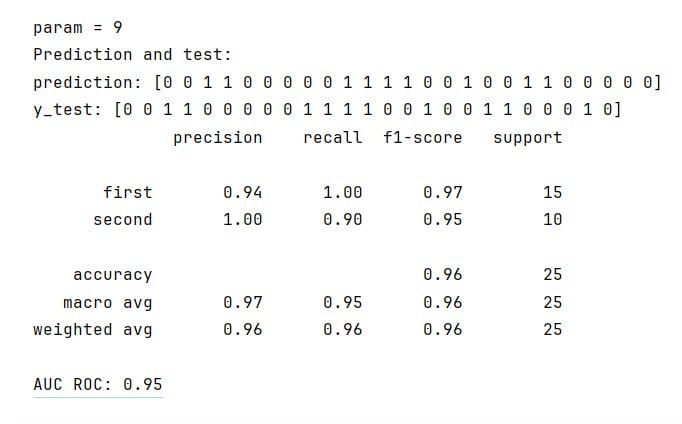


Рисунок 19 – Статистика по методу k-ближних соседей (n\_neighbors = 9)

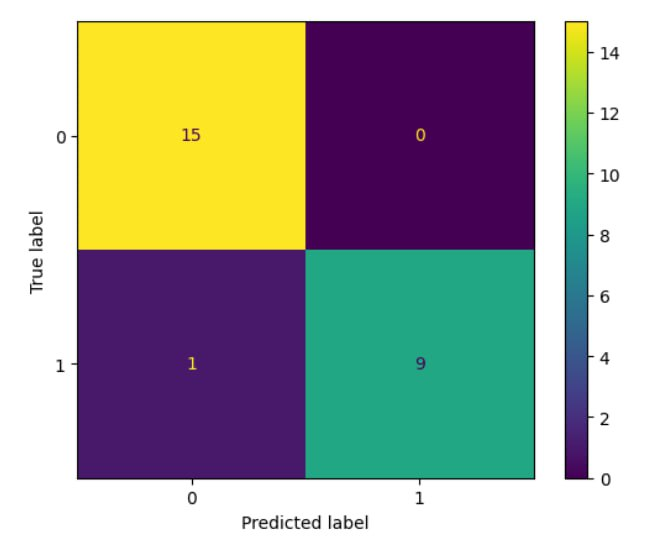


Рисунок 20 – Матрица ошибок по методу k-ближних соседей (n\_neighbors = 9)

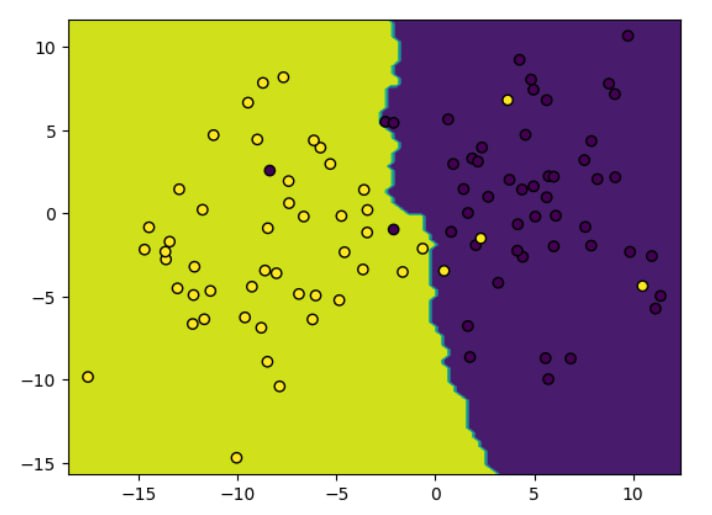


Рисунок 21 – Области принятия решений (метод k-ближних соседей (n\_neighbors = 9))

Проведём классификацию наивным байесовским методом с параметрами по умолчанию. Код представлен на рисунке 22.

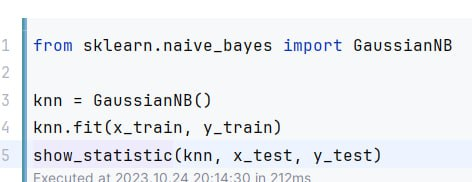


Рисунок 22 – Код для классификации наивным байесовским методом

Результаты работы представлены на рисунках 23-25.

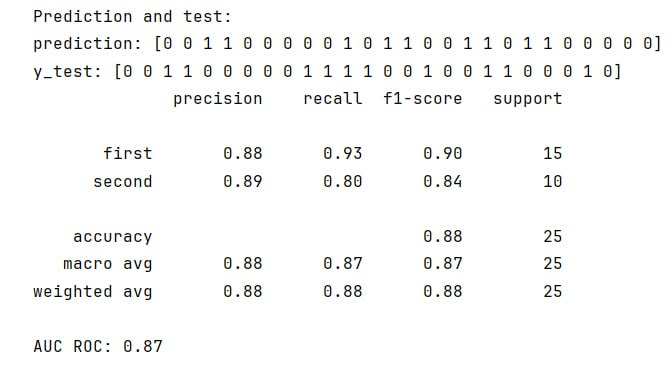


Рисунок 23 – Статистика по наивному байесовскому методу

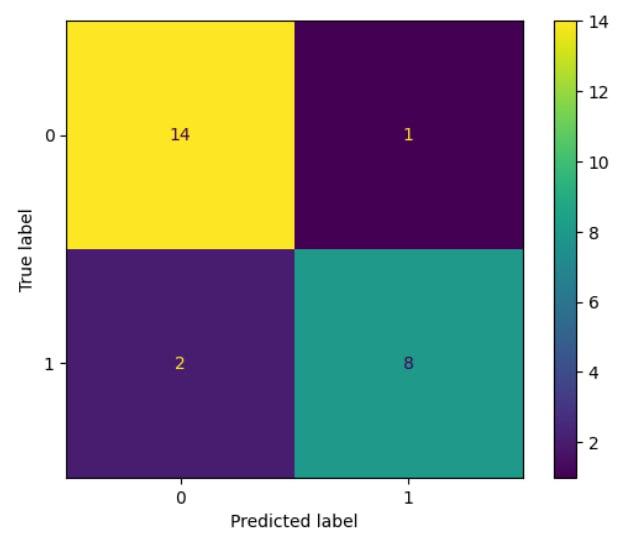


Рисунок 24 – Матрица ошибок по наивному байесовскому методу

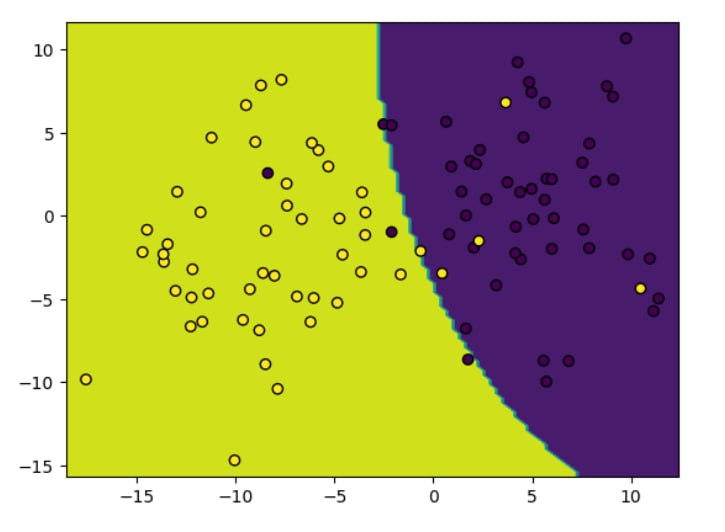


Рисунок 25 – Области принятия решения по наивному байесовскому методу

Проведём классификацию методом случайного леса поочередно используя параметр n\_estimators = 5, 10, 15, 20, 50. Функция представлена на рисунке 26.

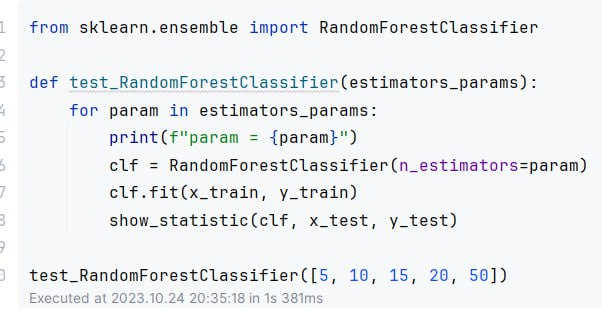


Рисунок 26 – Функция для метода случайного леса

Результаты работы представлены на рисунках 27-41.

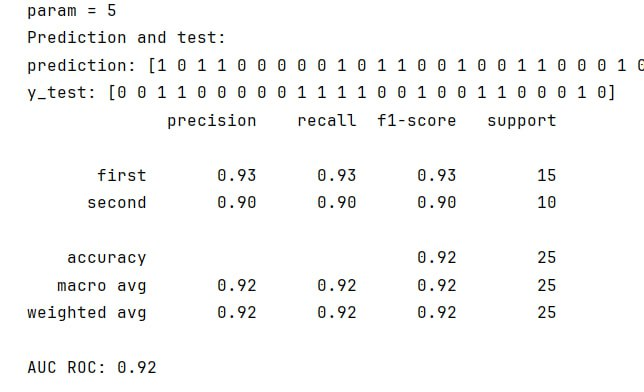


Рисунок 27 – Статистика по методу случайного леса (n\_estimators = 5)

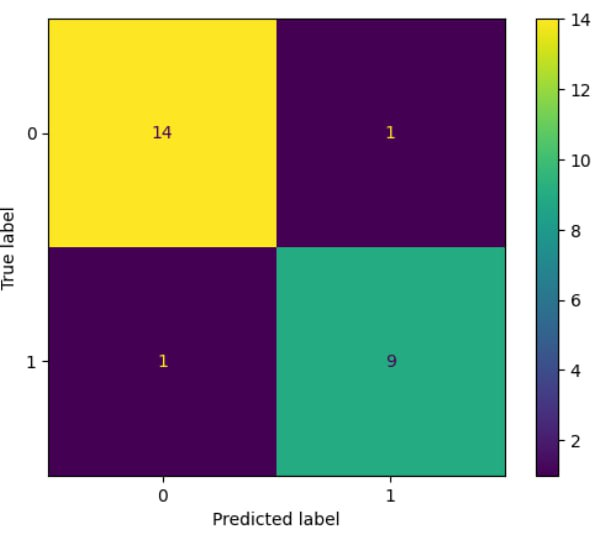


Рисунок 28 – Матрица ошибок по методу случайного леса (n\_estimators = 5)

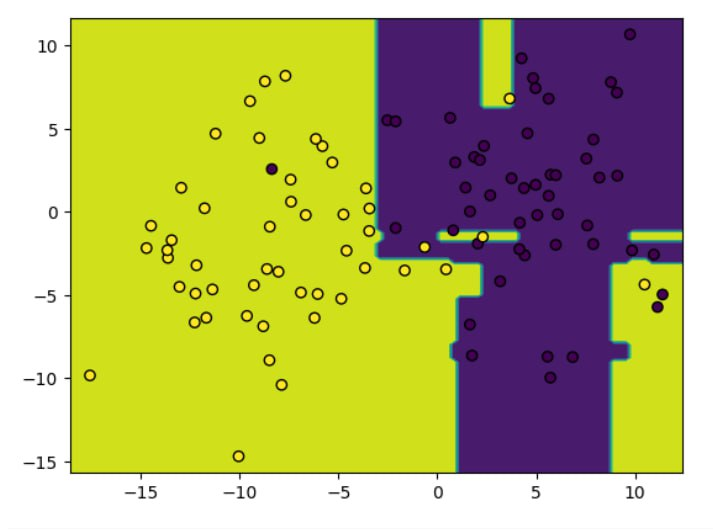


Рисунок 29 – Области принятия решений по методу случайного леса (n\_estimators = 5)

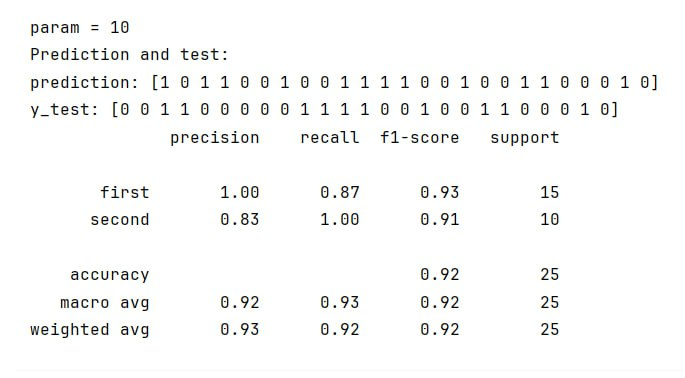


Рисунок 30 – Статистика по методу случайного леса (n\_estimators = 10)

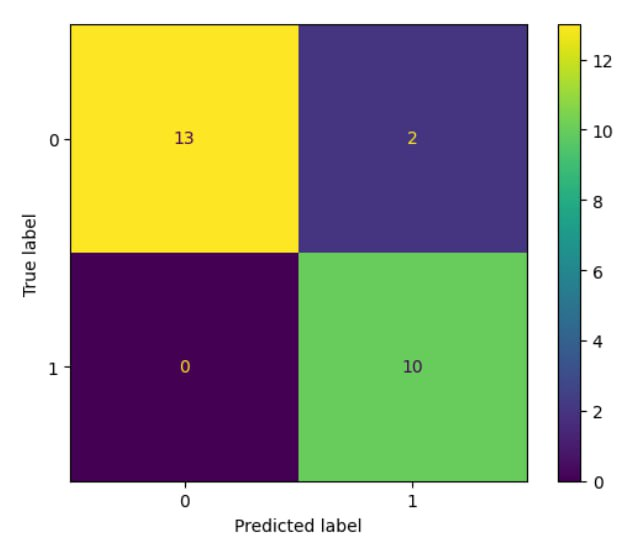


Рисунок 31 – Матрица ошибок по методу случайного леса (n\_estimators = 10)

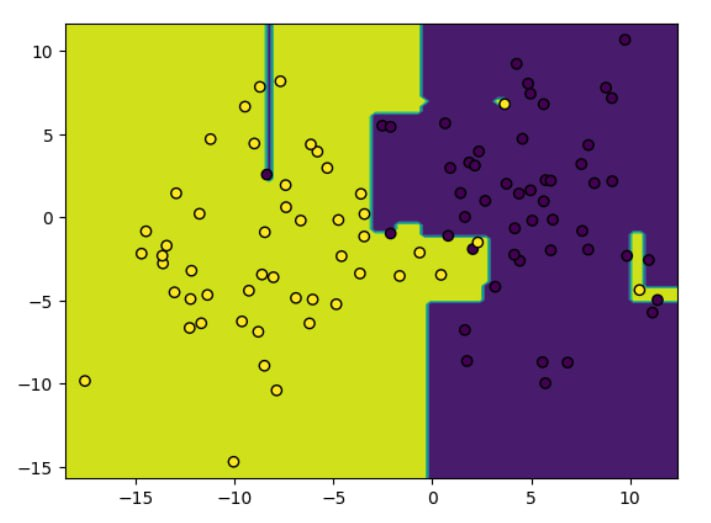


Рисунок 32 – Области принятия решений по методу случайного леса (n\_estimators = 10)

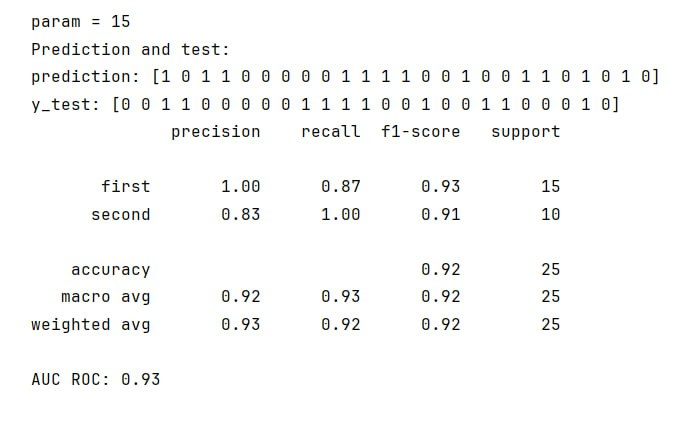


Рисунок 33 – Статистика по методу случайного леса (n\_estimators = 15)

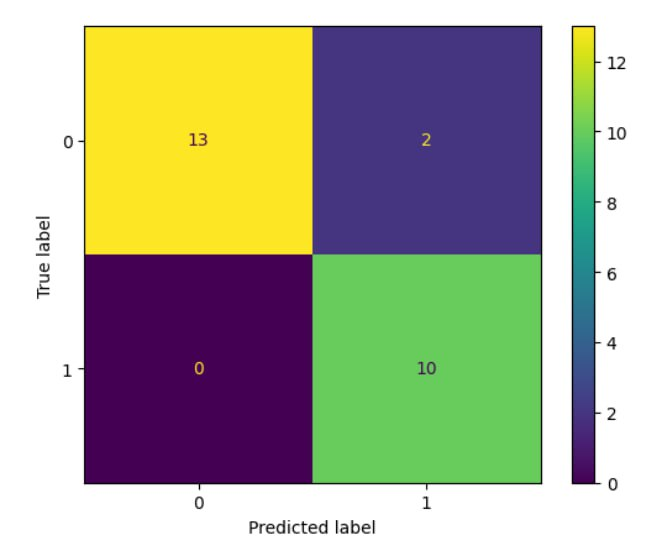


Рисунок 34 – Матрица ошибок по методу случайного леса (n\_estimators = 15)

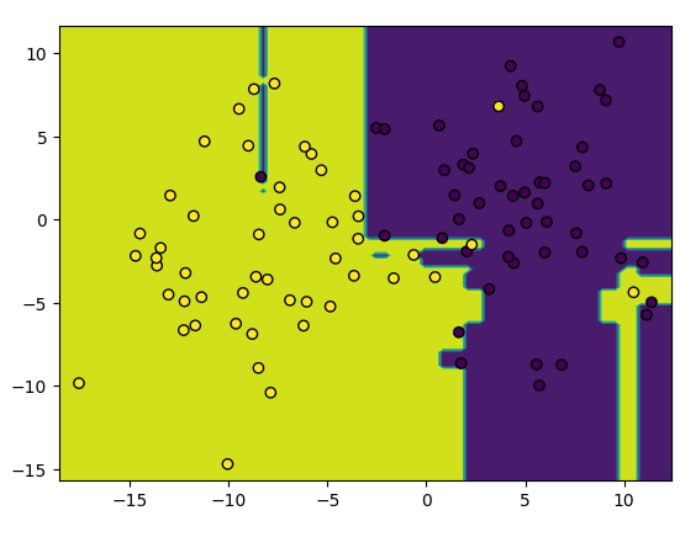


Рисунок 35 – Области принятия решений по методу случайного леса (n\_estimators = 15)

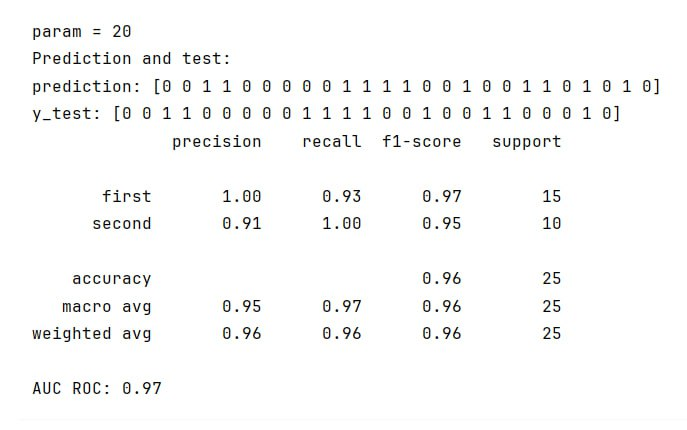


Рисунок 36 – Статистика по методу случайного леса (n\_estimators = 20)

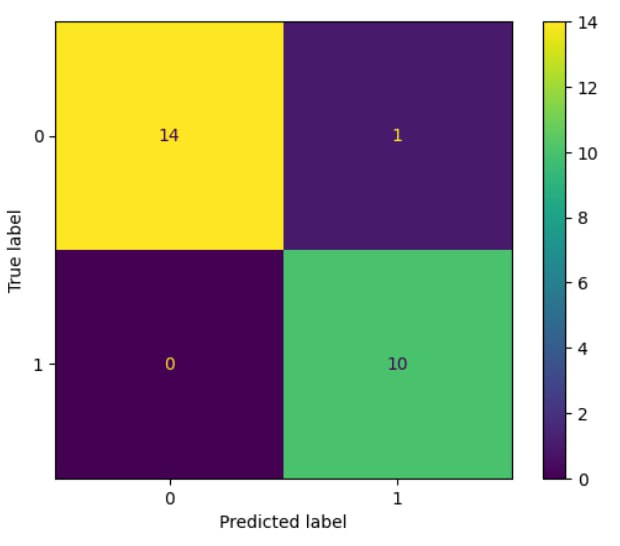


Рисунок 37 – Матрица ошибок по методу случайного леса (n\_estimators = 20)

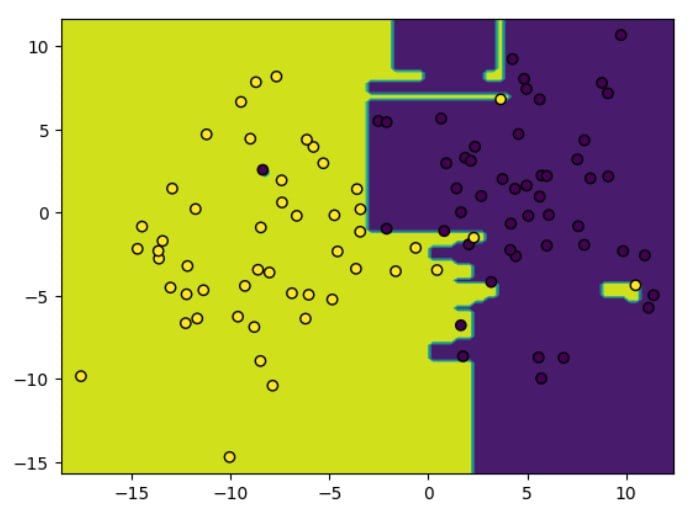


Рисунок 38 – Области принятия решений по методу случайного леса (n\_estimators = 20)

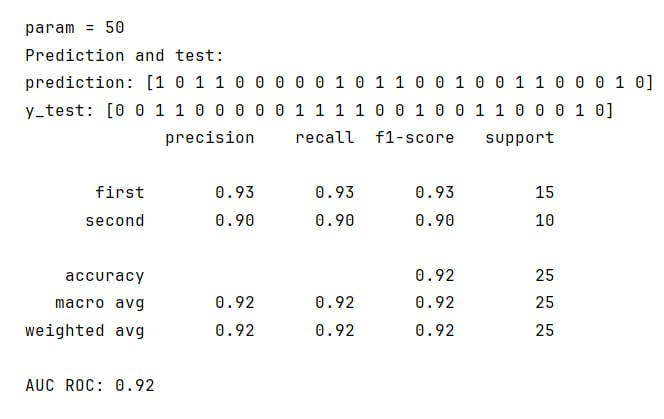


Рисунок 39 – Статистика по методу случайного леса (n\_estimators = 50)

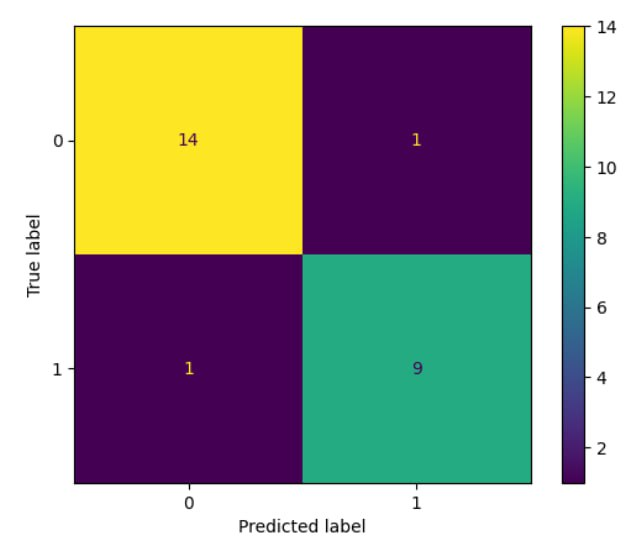


Рисунок 40 – Матрица ошибок по методу случайного леса (n\_estimators = 50)

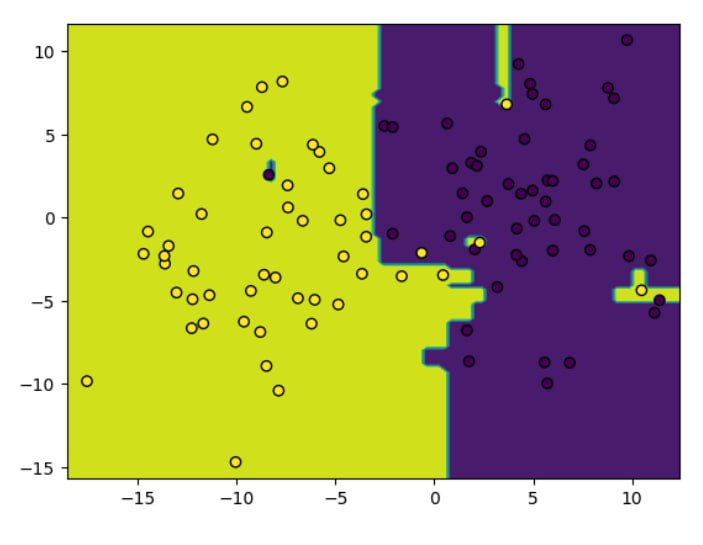


Рисунок 41 – Области принятия решений по методу случайного леса (n\_estimators = 50)

По итогам классификации при размере тестововой выборки 25% лучшие результаты показал метод случайного леса (n\_estimators = 20) и метод k-ближних соседей (n\_neighbors = 3). Увеличение параметров в этих методах не ведёт к улучшению результатов, а в случае с методом случайного леса вызывает ухудшение классификации. Увеличение тестовой выборки особо не ведёт к улучшению показателей (кроме наивного байесовского). Заполним сводную таблицу (таблица 1) для каждого метода и размера тестовой выборки 10%, 25% и 35%.

Таблица 1 – Сводная таблица

|  |  |  |
| --- | --- | --- |
| Метод | Размер тестовой выборки | Результат |
| k-ближних | 10 % | Confusion matrix:  [ 6 0 ]  [ 0 4 ]  Accuracy score: 1,00  AUC ROC:1,00 |
| Наивный байесовский | Confusion matrix:  [ 6 0 ]  [ 0 4]  Accuracy score: 1,00  AUC ROC: 1,00 |
| Рандомный лес | Confusion matrix:  [ 6 0 ]  [ 0 4 ]  Accuracy score: 1,00  AUC ROC: 1,00 |
| k-ближних | 25% | Confusion matrix:  [ 15 0 ]  [ 1 9]  Accuracy score: 0,96  AUC ROC:0,95 |
| Наивный байесовский | Confusion matrix:  [ 14 1]  [ 2 8]  Accuracy score: 0,88  AUC ROC: 0,87 |
| Рандомный лес | Confusion matrix:  [ 14 1]  [ 0 10]  Accuracy score: 0,96  AUC ROC: 0,97 |

Окончание таблицы 1

|  |  |  |
| --- | --- | --- |
| Метод | Размер тестовой выборки | Результат |
| k-ближних | 35% | Confusion matrix:  [ 16 2 ]  [ 0 17 ]  Accuracy score: 0,94  AUC ROC:0,94 |
| Наивный байесовский | Confusion matrix:  [ 16 2]  [ 0 17]  Accuracy score: 0,94  AUC ROC: 0,94 |
| Рандомный лес | Confusion matrix:  [ 16 2]  [ 0 17]  Accuracy score: 0,94  AUC ROC: 0,94 |

Вывод

Мы получили практические навыки решения задачи бинарной классификации данных.

Приложение A (рекомендованное)

Исходный код

import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.datasets import make\_blobs

ds = make\_blobs(random\_state=28, centers=2, cluster\_std=4.5, shuffle=1)  
df = pd.DataFrame(ds[0], columns=['first', 'second'])  
df['class'] = ds[1]  
df.head(15)

df.plot.scatter(x='first', y='second', c='class', colormap='viridis')

from sklearn.model\_selection import train\_test\_split  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(ds[0], ds[1])

train\_df = pd.DataFrame(x\_train, columns=['first', 'second'])  
train\_df['class'] = y\_train  
train\_df.plot.scatter(x='first', y='second', c='class', colormap='viridis')

test\_df = pd.DataFrame(x\_test, columns=['first', 'second'])  
test\_df['class'] = y\_test  
test\_df.plot.scatter(x='first', y='second', c='class', colormap='viridis')

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay, classification\_report, roc\_auc\_score  
from sklearn.inspection import DecisionBoundaryDisplay  
  
def show\_statistic(knn, x\_test, y\_test):  
 knn.fit(x\_train, y\_train)  
 prediction = knn.predict(x\_test)  
   
 print('Prediction and test: ')  
 print(f"prediction: {prediction}")  
 print(f"y\_test: {y\_test}")  
   
 cm = confusion\_matrix(y\_test, prediction, labels=knn.classes\_)  
 disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=knn.classes\_)  
 disp.plot()  
 print(classification\_report(y\_test, prediction, target\_names=['first', 'second']))  
 print("AUC ROC: {:.2f}".format(roc\_auc\_score(y\_test, prediction)))  
 disp = DecisionBoundaryDisplay.from\_estimator(knn, ds[0], response\_method="predict")  
 disp.ax\_.scatter(ds[0][:,0], ds[0][:,1], c=ds[1], edgecolor="k")  
 plt.show()

from sklearn.neighbors import KNeighborsClassifier  
  
def test\_KNeighthboursClassifier(neighbors\_params):  
 for param in neighbors\_params:  
 print(f"param = {param}")  
 clf = KNeighborsClassifier(n\_neighbors=param)  
 show\_statistic(clf, x\_test, y\_test)  
  
test\_KNeighthboursClassifier([1, 3, 5, 9])

from sklearn.naive\_bayes import GaussianNB  
  
knn = GaussianNB()  
knn.fit(x\_train, y\_train)  
show\_statistic(knn, x\_test, y\_test)

from sklearn.ensemble import RandomForestClassifier  
  
def test\_RandomForestClassifier(estimators\_params):  
 for param in estimators\_params:  
 print(f"param = {param}")  
 clf = RandomForestClassifier(n\_estimators=param)  
 clf.fit(x\_train, y\_train)  
 show\_statistic(clf, x\_test, y\_test)  
  
test\_RandomForestClassifier([5, 10, 15, 20, 50])

x\_train, x\_test, y\_train, y\_test = train\_test\_split(ds[0], ds[1], test\_size=0.1)  
  
train\_df = pd.DataFrame(x\_train, columns=['first', 'second'])  
train\_df['class'] = y\_train  
train\_df.plot.scatter(x='first', y='second', c='class', colormap='viridis')

test\_df = pd.DataFrame(x\_test, columns=['first', 'second'])  
test\_df['class'] = y\_test  
test\_df.plot.scatter(x='first', y='second', c='class', colormap='viridis')

test\_KNeighthboursClassifier([1, 3, 5, 9])  
  
knn = GaussianNB()  
knn.fit(x\_train, y\_train)  
show\_statistic(knn, x\_test, y\_test)  
  
test\_RandomForestClassifier([5, 10, 15, 20, 50])

x\_train, x\_test, y\_train, y\_test = train\_test\_split(ds[0], ds[1], test\_size=0.35)  
  
train\_df = pd.DataFrame(x\_train, columns=['first', 'second'])  
train\_df['class'] = y\_train  
train\_df.plot.scatter(x='first', y='second', c='class', colormap='viridis')

test\_df = pd.DataFrame(x\_test, columns=['first', 'second'])  
test\_df['class'] = y\_test  
test\_df.plot.scatter(x='first', y='second', c='class', colormap='viridis')

test\_KNeighthboursClassifier([1, 3, 5, 9])  
  
knn = GaussianNB()  
knn.fit(x\_train, y\_train)  
show\_statistic(knn, x\_test, y\_test)  
  
test\_RandomForestClassifier([5, 10, 15, 20, 50])