Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине «Прикладные интеллектуальные системы и экспертные системы»

Бинарная классификация

Студент Осипов А.А.

Группа М-ИАП-22

Руководитель Кургасов В.В.

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.

Ход работы

Вариант 10.

Вариант	**10**
Вид классов	classification
Random_state	58
class_sep	0.7

Для всех вариантов, использующих для генерации make_classification, дополнительные параметры: n_features=2, n_redundant=0, n_informative=1, n_clusters_per_class=1.

Импортируем необходимые для работы библиотеки и модули.

```
from sklearn.datasets import make_classification
import numpy as np
import matplotlib.pyplot as plt
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
```

Загрузим данные в соответствии с вариантом

Загрузка данных

```
coordinates, labels = make_classification(random_state=58, class_sep=0.7,  n_features=2,n_redundant=0,n_informative=1, n_clusters
```

Рисунок 1 – Загрузка данных

Выведем первые 15 элементов выборки (координаты точек и метки класса).

Первые 15 элементов выборки

```
print("X координаты точек:", coordinates[:15,0])
print("Y координаты точек:", coordinates[:15,0])
print("Метки класса:", labels[:15])
```

```
X координаты точек: [-0.52997741  0.36661531  1.10271671 -1.16536177  0.83331961  0.98304946
  0.40771148 -0.20917139 -0.75337982 -1.27401981 -0.66115944 -0.47119838
 -0.80756863 -0.16455133 -0.55561578]
Y координаты точек: [-0.52997741  0.36661531  1.10271671 -1.16536177  0.83331961  0.98304946
  0.40771148 -0.20917139 -0.75337982 -1.27401981 -0.66115944 -0.47119838
 -0.80756863 -0.16455133 -0.55561578]
Метки класса: [1 1 0 1 0 0 0 0 0 1 0 1 0 0 0]
```

Рисунок 2 – Элементы выборки

Отобразим на графике сгенерированную выборку. Объекты разных классов должны иметь разные цвета.

Отобразим на графике сгенерированную выборку

```
plt.scatter(coordinates[:,0], coordinates[:,1], c=labels)
```

```
<matplotlib.collections.PathCollection at 0x1c49f0a22e0>
```

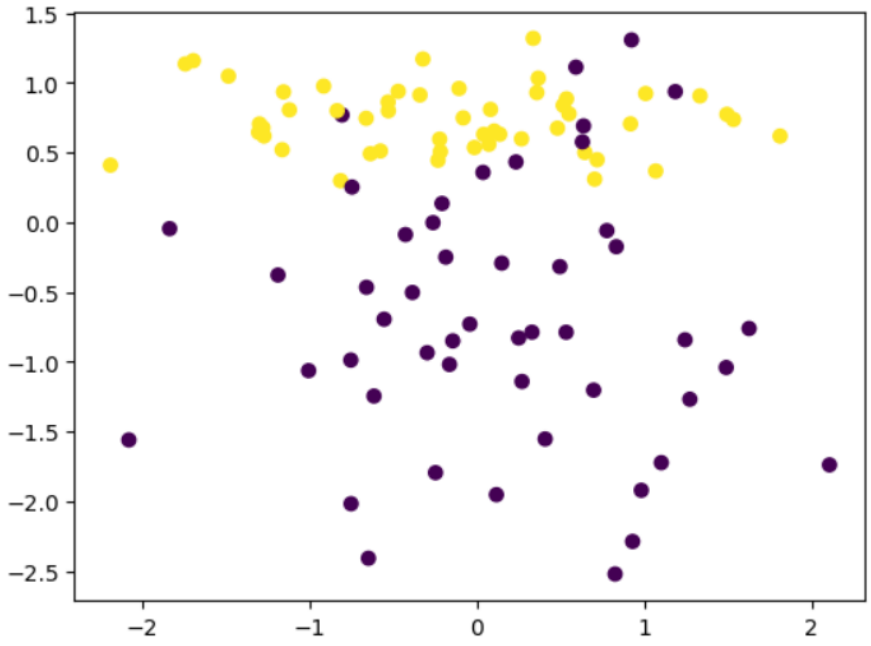


Рисунок 3 – Элементы выборки

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

Обучающущая и тестовая выборки

```
coordinates_train, coordinates_test, labels_train, labels_test = train_test_split(coordinates, labels, test_size = 0.25, random_
```

Рисунок 4 – Разбиение на тестовое и обучающее множество

Отобразим на графике обучающую и тестовую выборки. Объекты разных классов должны иметь разные цвета.

Выборки на графиках

- обучающая

```
plt.scatter(coordinates_train[:,0], coordinates_train[:,1], c=labels_train)
```

```
<matplotlib.collections.PathCollection at 0x1c49f1c9820>
```

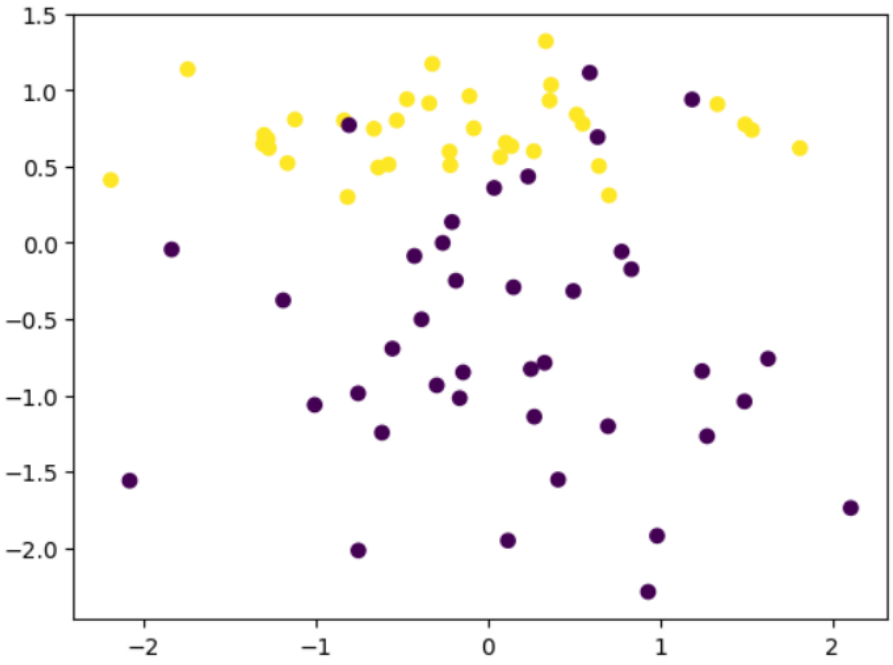


Рисунок 5 – График обучающей выборки

- тестовая

```
plt.scatter(coordinates_test[:,0], coordinates_test[:,1], c=labels_test)
```

```
<matplotlib.collections.PathCollection at 0x1c49f2f7190>
```

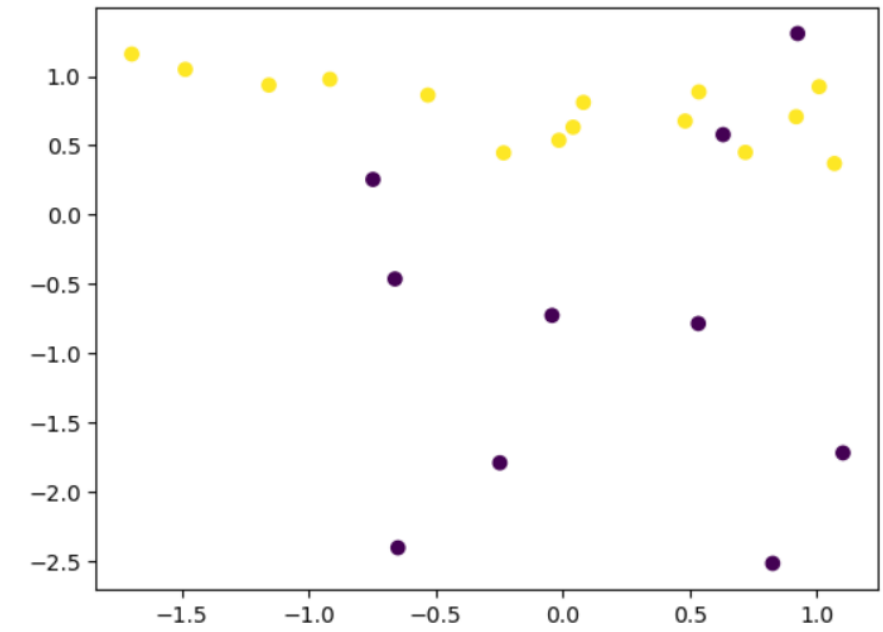


Рисунок 6 – График тестовой выборки

Реализуем модели классификаторов, обучим их на обучающем множестве. Применим модели на тестовой выборке, выведем 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:

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

k-ближайших соседей

```
: n = [1, 3, 5, 9]
for i in n:
    kNN = KNeighborsClassifier(n_neighbors=i, metric='euclidean')
    kNN.fit(coordinates_train, labels_train)
    prediction = kNN.predict(coordinates_test)
    print("для n_neightbors = ", i)
    print_data(kNN, coordinates, labels, prediction, labels_test)
для n_neightbors =  1
Предстказанные и истинные значения
[0 0 0 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 8  2]
 [ 2 13]]
Точность классификации:  0.84
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.80      0.80      0.80        10
           1       0.87      0.87      0.87        15

    accuracy                           0.84        25
   macro avg       0.83      0.83      0.83        25
weighted avg       0.84      0.84      0.84        25

Значение площади под кривой ошибок (AUC ROC)
0.8333333333333334
```

Рисунок 7 – Метод k-ближайших соседей (n=1)

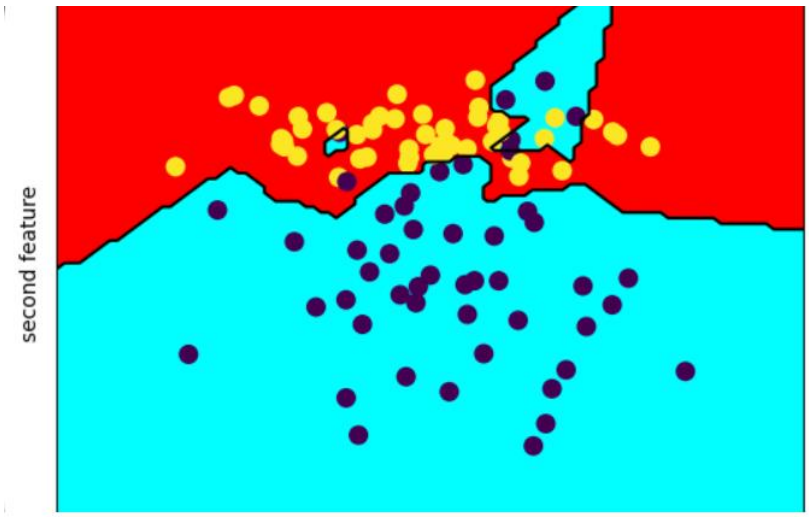


Рисунок 8 – Метод k-ближайших соседей (n=1)

```
для n_neightbors =  3
Предстказанные и истинные значения
[0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 8  2]
 [ 2 13]]
Точность классификации:  0.84
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.80      0.80      0.80        10
           1       0.87      0.87      0.87        15

    accuracy                           0.84        25
   macro avg       0.83      0.83      0.83        25
weighted avg       0.84      0.84      0.84        25

Значение площади под кривой ошибок (AUC ROC)
0.8333333333333334
```

Рисунок 9 – Метод k-ближайших соседей (n=3)

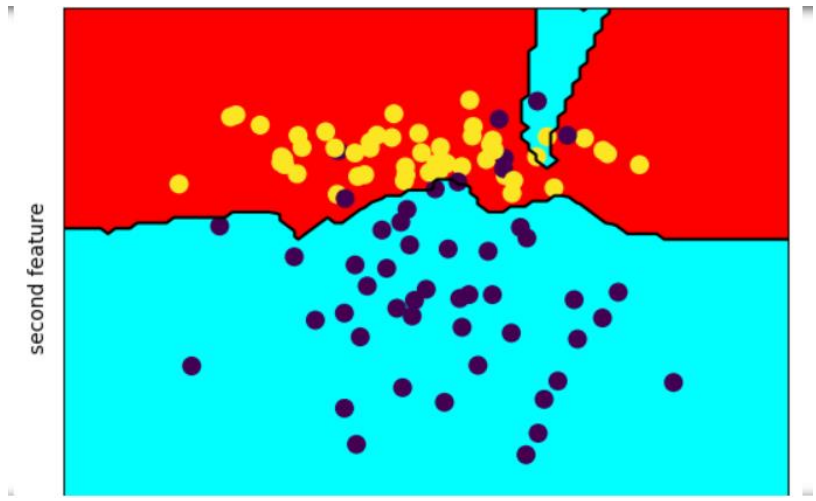


Рисунок 10 – Метод k-ближайших соседей (n=3)

```
для n_neightbors =  5
Предстказанные и истинные значения
[1 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 7  3]
 [ 2 13]]
Точность классификации:  0.8
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.78      0.70      0.74        10
           1       0.81      0.87      0.84        15

    accuracy                           0.80        25
   macro avg       0.80      0.78      0.79        25
weighted avg       0.80      0.80      0.80        25

Значение площади под кривой ошибок (AUC ROC)
0.7833333333333333
```

Рисунок 11 – Метод k-ближайших соседей (n=5)

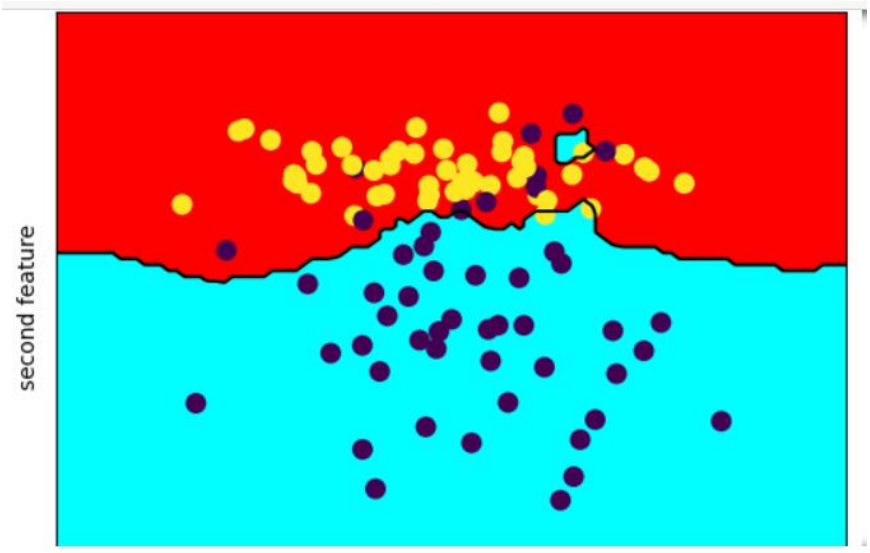


Рисунок 12 – Метод k-ближайших соседей (n=5)

```
для n_neightbors =  9
Предстказанные и истинные значения
[1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 7  3]
 [ 0 15]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       1.00      0.70      0.82        10
           1       0.83      1.00      0.91        15

    accuracy                           0.88        25
   macro avg       0.92      0.85      0.87        25
weighted avg       0.90      0.88      0.87        25

Значение площади под кривой ошибок (AUC ROC)
0.85
```

Рисунок 13 – Метод k-ближайших соседей (n=9)

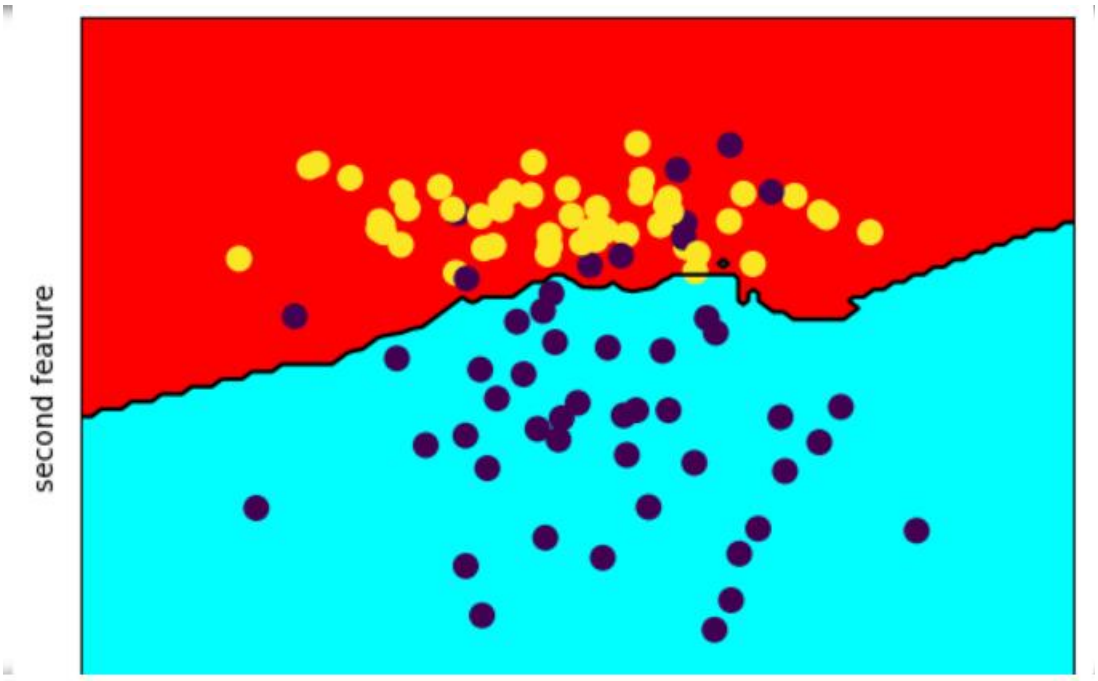


Рисунок 14 – Метод k-ближайших соседей (n=9)

b) Наивный байесовский метод

- Наивный байесовский метод

```
naive = GaussianNB()
naive.fit(coordinates_train, labels_train)
predict = naive.predict(coordinates_test)
print_data(naive, coordinates, labels, predict, labels_test)
```

```
Предстказанные и истинные значения
[1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 8  2]
 [ 0 15]]
Точность классификации:  0.92
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       1.00      0.80      0.89        10
           1       0.88      1.00      0.94        15

    accuracy                           0.92        25
   macro avg       0.94      0.90      0.91        25
weighted avg       0.93      0.92      0.92        25

Значение площади под кривой ошибок (AUC ROC)
0.9
Область принятия решений
```

Рисунок 15 – Наивный байесовский метод

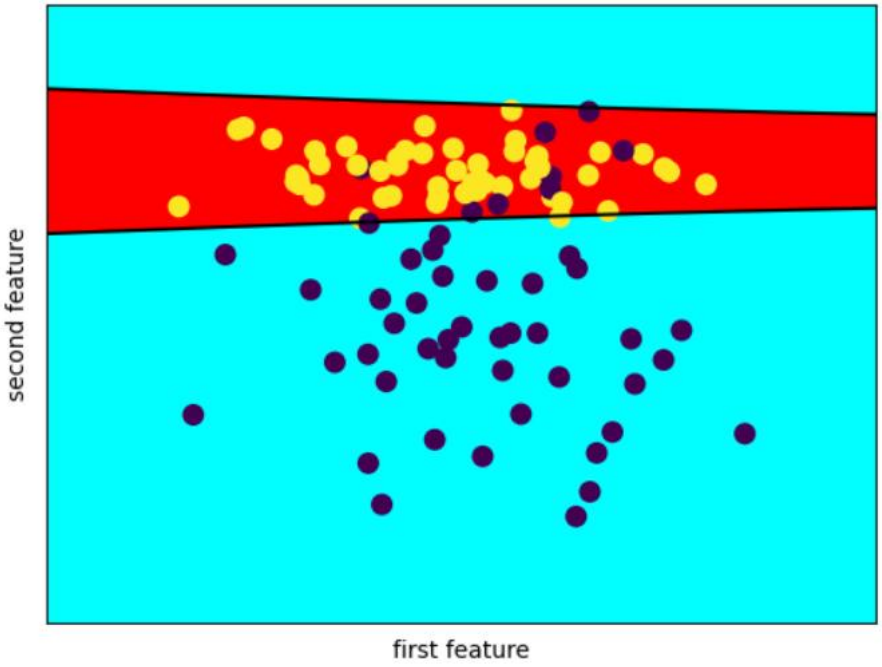


Рисунок 16 – Наивный байесовский метод

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

- Случайный лес

```
In [16]: n = [5, 10, 15, 20, 50]
         for i in n:
             rand_forest = RandomForestClassifier(n_estimators=i)
             rand_forest.fit(coordinates_train, labels_train)
             prediction = rand_forest.predict(coordinates_test)
             print("для n_estimators = ", i)
             print_data(kNN, coordinates, labels, prediction, labels_test)
```

```
для n_estimators =  5
Предстказанные и истинные значения
[1 0 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[10  0]
 [ 2 13]]
Точность классификации:  0.92
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.83      1.00      0.91        10
           1       1.00      0.87      0.93        15

    accuracy                           0.92        25
   macro avg       0.92      0.93      0.92        25
weighted avg       0.93      0.92      0.92        25

Значение площади под кривой ошибок (AUC ROC)
0.9333333333333333
```

Рисунок 17 – Случайный лес n = 5

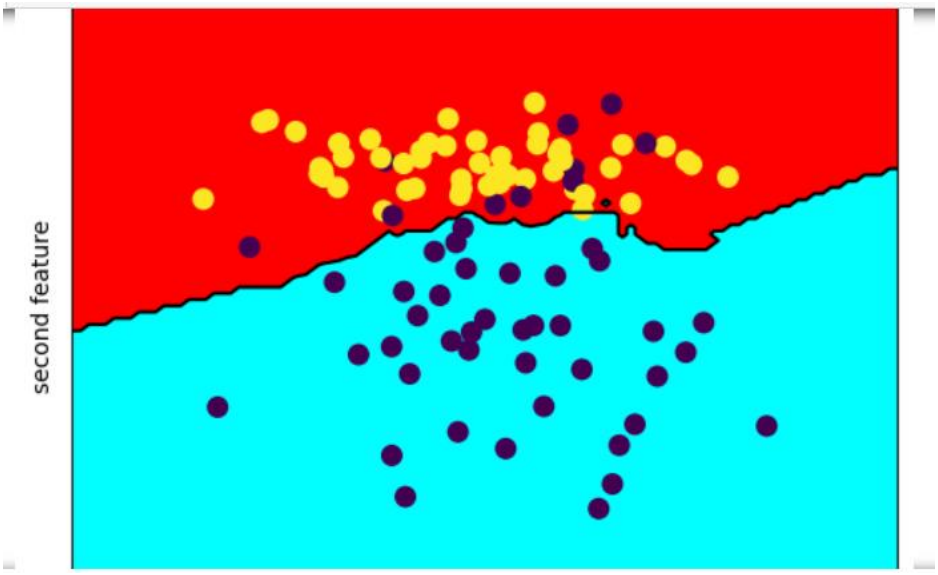


Рисунок 18 – Случайный лес n = 5

```
для n_estimators =  10
Предстказанные и истинные значения
[1 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 9  1]
 [ 3 12]]
Точность классификации:  0.84
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.75      0.90      0.82        10
           1       0.92      0.80      0.86        15

    accuracy                           0.84        25
   macro avg       0.84      0.85      0.84        25
weighted avg       0.85      0.84      0.84        25

Значение площади под кривой ошибок (AUC ROC)
0.8500000000000001
```

Рисунок 19 – Случайный лес n = 10

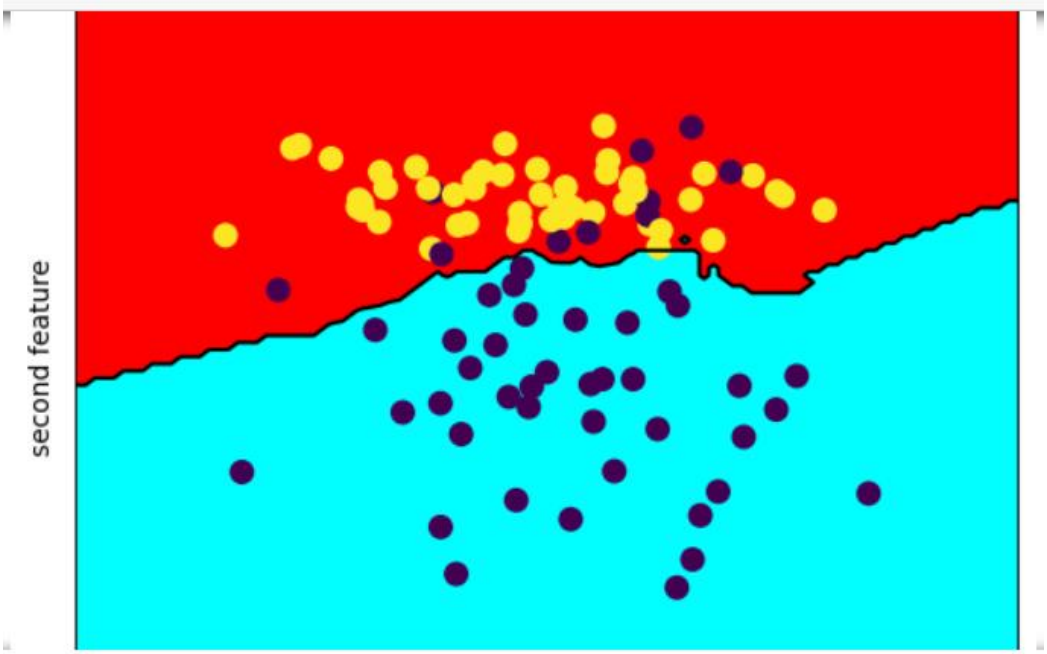


Рисунок 20 – Случайный лес n = 10

```
для n_estimators =  15
Предстказанные и истинные значения
[1 0 0 1 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 7  3]
 [ 4 11]]
Точность классификации:  0.72
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.64      0.70      0.67        10
           1       0.79      0.73      0.76        15

    accuracy                           0.72        25
   macro avg       0.71      0.72      0.71        25
weighted avg       0.73      0.72      0.72        25

Значение площади под кривой ошибок (AUC ROC)
0.7166666666666667
```

Рисунок 21 – Случайный лес n = 15

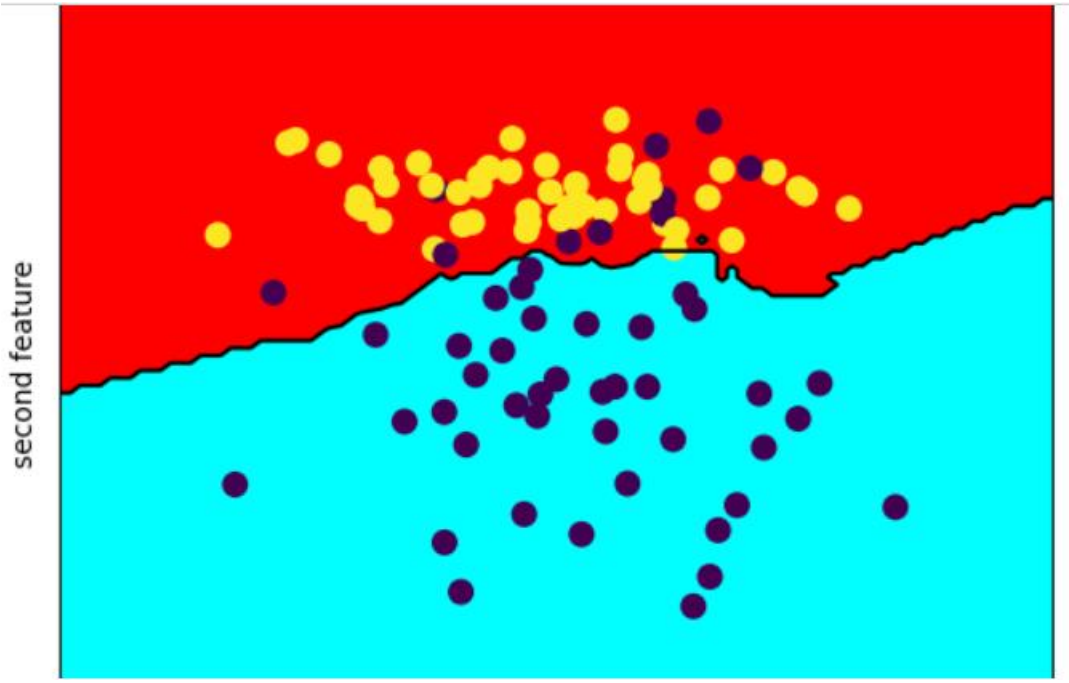


Рисунок 22 – Случайный лес n = 15

```
для n_estimators =  20
Предстказанные и истинные значения
[0 0 0 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 9  1]
 [ 3 12]]
Точность классификации:  0.84
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.75      0.90      0.82        10
           1       0.92      0.80      0.86        15

    accuracy                           0.84        25
   macro avg       0.84      0.85      0.84        25
weighted avg       0.85      0.84      0.84        25

Значение площади под кривой ошибок (AUC ROC)
0.8500000000000001
```

Рисунок 23 – Случайный лес n = 20

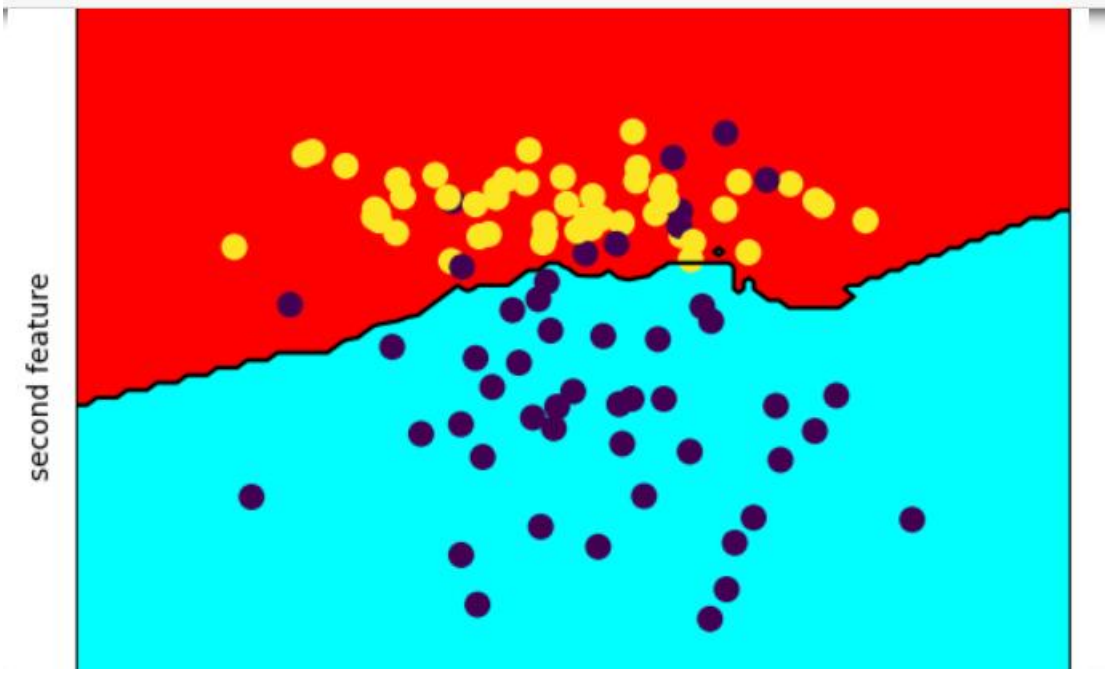


Рисунок 24 – Случайный лес n = 20

```
для n_estimators =  50
Предстказанные и истинные значения
[1 0 0 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1]
[1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]
Матрица ошибок
[[ 8  2]
 [ 1 14]]
Точность классификации:  0.88
Значения полноты, точности, f1-меры и аккуратности
              precision    recall  f1-score   support

           0       0.89      0.80      0.84        10
           1       0.88      0.93      0.90        15

    accuracy                           0.88        25
   macro avg       0.88      0.87      0.87        25
weighted avg       0.88      0.88      0.88        25

Значение площади под кривой ошибок (AUC ROC)
0.8666666666666668
```

Рисунок 25 – Случайный лес n = 50

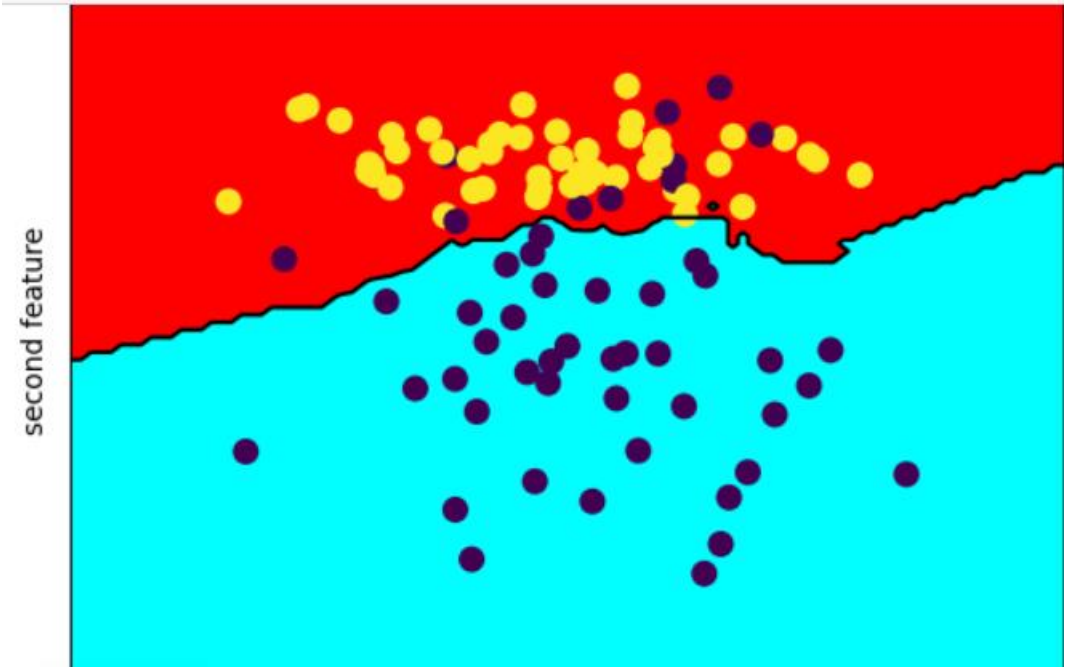


Рисунок 26 – Случайный лес n = 50

По результатам п.8 занесем в отчет таблицу 1 с результатами классификации всеми методами и выводы о наиболее подходящем методе классификации ваших данных.

Таблица 1 – Результаты работы программы

Метод	Истинные и предсказанные метки классов	Матрица ошибок	Значения полноты, точности, f1-меры и аккуратности	Значение площади под кривой ошибок
k-ближайших соседей n = 1	Предстказанные и истинные значения [0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[8 2] [2 13]	Precision(0) = 0.8 Precision(1) = 0.87 recall(0)= 0.8 recall(1)= 0.87 f1-score(0)= 0.8 f1-score(1)= 0.87 accuracy = 0.84	0.83
k-ближайших соседей n = 3	Предстказанные и истинные значения [0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[8 2] [2 13]	Precision(0) = 0.8 Precision(1) = 0.87 recall(0)= 0.8 recall(1)= 0.87 f1-score(0)= 0.8 f1-score(1)= 0.87 accuracy = 0.84	0.83
k-ближайших соседей n = 5	Предстказанные и истинные значения [1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[7 3] [2 13]	Precision(0) = 0.78 Precision(1) = 0.81 recall(0)= 0.7 recall(1)= 0.87 f1-score(0)= 0.74 f1-score(1)= 0.84 accuracy = 0.8	0.783

k-ближайших соседей n = 9	Предстказанные и истинные значения [1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[7 3] [0 15]	Precision(0) = 1 Precision(1) = 0.83 recall(0)= 0.7 recall(1)= 1 f1-score(0)= 0.82 f1-score(1)= 0.91 accuracy = 0.88	0.85
Наивный байесовский метод	Предстказанные и истинные значения [1 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[8 2] [0 15]	Precision(0) = 1 Precision(1) = 0.88 recall(0)= 0.8 recall(1)= 1 f1-score(0)= 0.89 f1-score(1)= 0.94 accuracy = 0.92	0.9
Случайный лес n = 5	Предстказанные и истинные значения [1 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[10 0] [2 13]	Precision(0) = 0.83 Precision(1) = 1 recall(0)= 1 recall(1)= 0.87 f1-score(0)= 0.91 f1-score(1)= 0.93 accuracy = 0.92	0.93
Случайный лес n = 10	Предстказанные и истинные значения [1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[9 1] [3 12]	Precision(0) = 0.75 Precision(1) = 0.92 recall(0)= 0.9 recall(1)= 0.8 f1-score(0)= 0.82 f1-score(1)= 0.86 accuracy = 0.84	0.85
Случайный лес n = 15	Предстказанные и истинные значения [1 0 0 1 0 1 0 1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[7 3] [4 11]	Precision(0) = 0.64 Precision(1) = 0.79	0.716

			recall(0)= 0 .7 recall(1)= 0 .73 f1-score(0)= 0.67 f1-score(1)= 0.76 accuracy = 0 .72	
Случайный лес n = 20	Предстказанные и истинные значения [0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[9 1] [3 12]	Precision(0) = 0.75 Precision(1) = 0.92 recall(0)= 0 .9 recall(1)= 0 .8 f1-score(0)= 0.82 f1-score(1)= 0.86 accuracy = 0 .84	0.85
Случайный лес n = 50	Предстказанные и истинные значения [1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 1] [1 1 0 1 1 1 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 1 0 1 1]	[8 2] [1 14]	Precision(0) = 0.89 Precision(1) = 0.88 recall(0)= 0 .8 recall(1)= 0 .93 f1-score(0)= 0.84 f1-score(1)= 0.9 accuracy = 0 .88	0.866

Наиболее подходящие методы для классификации данных - случайный лес, где n = 5 и наивный байесовский метод, так как значение аккуратности равно 0,92, что превышает значения относящихся к остальным методам.

Код программы

```
## Импорт необходимых библиотек
from sklearn.datasets import make_classification
import numpy as np
import matplotlib.pyplot as plt
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
## Функция вывода данных
def print_data(classifier, coordinates, labels, prediction, labels_test):
    print("Предстказанные и истинные значения")
    print(prediction)
    print(labels_test)
    print("Матрица ошибок")
    print(confusion_matrix(labels_test, prediction))
    print("Точность классификации: ", accuracy_score(prediction,
labels_test))
    print("Значения полноты, точности, f1-меры и аккуратности")
    print(classification_report(labels_test, prediction))
    print("Значение площади под кривой ошибок (AUC ROC)")
    print(roc_auc_score(labels_test, prediction))
    print("Область принятия решений")
    plt.xlabel("first feature")
    plt.ylabel("second feature")
    eps = 1.0
    x_min, x_max = coordinates[:, 0].min() - eps, coordinates[:, 0].max() + eps
    y_min, y_max = coordinates[:, 1].min() - eps, coordinates[:, 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]
            ax = plt.gca()
            ax.contourf(X1, X2, decision_values.reshape(X1.shape),
            levels=fill_levels, colors=['cyan', 'red', 'blue'])
            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(())

        plt.scatter(coordinates[:, 0], coordinates[:, 1], c=labels, s=70)
        plt.show()
    ## Загрузка данных
    coordinates, labels = make_classification(random_state=58, class_sep=0.7,
n_features=2,n_redundant=0,n_informative=1, n_clusters_per_class=1)
#генератор данных кластеризации
    ## Первые 15 элементов выборки
    print("X координаты точек:", coordinates[:15,0])
    print("Y координаты точек:", coordinates[:15,0])
    print("Метки класса:", labels[:15])
    ## Отобразим на графике сгенерированную выборку
    plt.scatter(coordinates[:,0], coordinates[:,1], c=labels)
```

```
## Обучающущая и тестовая выборки
coordinates_train, coordinates_test, labels_train, labels_test = train_test_split(coordinates, labels, test_size = 0.25, random_state=1)
##  Выборки на графиках
#### - обучающая
plt.scatter(coordinates_train[:,0], coordinates_train[:,1], c=labels_train)
#### - тестовая
plt.scatter(coordinates_test[:,0], coordinates_test[:,1], c=labels_test)
##  Модели классификаторов
n = [1, 3, 5, 9]
for i in n:
    kNN = KNeighborsClassifier(n_neighbors=i, metric='euclidean')
    kNN.fit(coordinates_train, labels_train)
    prediction = kNN.predict(coordinates_test)
    print("для n_neightbors = ", i)
    print_data(kNN, coordinates, labels, prediction, labels_test)
#### - Наивный байесовский метод
naive = GaussianNB()
naive.fit(coordinates_train, labels_train)
predict = naive.predict(coordinates_test)
print_data(naive, coordinates, labels, predict, labels_test)
#### - Случайный лес
n = [5, 10, 15, 20, 50]
for i in n:
    rand_forest = RandomForestClassifier(n_estimators=i)
    rand_forest.fit(coordinates_train, labels_train)
    prediction = rand_forest.predict(coordinates_test)
    print("для n_estimators = ", i)
    print_data(kNN, coordinates, labels, prediction, labels_test)
```

Вывод

В ходе выполнения данной лабораторной работы мы получили базовые навыки работы с языком python и набором функций для анализа и обработки данных. Получили практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook. Научились загружать данные, обучать классификаторы и проводить классификацию. Научились оценивать точность полученных моделей.

Контрольные вопросы

1) Постановка задачи классификации данных. Что такое бинарная классификация?

Задача классифика́ции — задача, в которой имеется множество объектов (ситуаций), разделённых, некоторым образом, на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать (см. ниже) произвольный объект из исходного множества.

Классифици́ровать объект — значит, указать номер (или наименование) класса, к которому относится данный объект.

Классифика́ция объекта — номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

Бинарная классификация – это один из типов задач классификации в машинном обучении, когда мы должны классифицировать два взаимоисключающих класса. Например, классифицировать сообщения как спам или не спам, классифицировать новости как фальшивые или настоящие.

2) Общий алгоритм решения задачи классификации данных.

1. Конструирование модели: описание множества предопределенных классов.

- Каждый пример набора данных относится к одному предопределенному классу.

- На этом этапе используется обучающее множество, на нем происходит конструирование модели.

- Полученная модель представлена классификационными правилами, деревом решений или математической формулой.

2. Использование модели: классификация новых или неизвестных значений.

- Оценка правильности (точности) модели.

1. Известные значения из тестового примера сравниваются с результатами использования полученной модели.

2. Уровень точности - процент правильно классифицированных примеров в тестовом множестве.

3. Тестовое множество, т.е. множество, на котором тестируется построенная модель, не должно зависеть от обучающего множества.

- Если точность модели допустима, возможно использование модели для классификации новых примеров, класс которых неизвестен

3) Чем отличаются обучающая и тестовая выборки? Какие существуют способы формирования обучающей и тестовой выборок?

Обучающая выборка - это набор, который подается на вход модели в процессе обучения вместе с ответами, с целью научить модель видеть связь между этими признаками и правильным ответом

Тестовая выборка используется для проверки модели. Модель не получает целевой признак на вход и, более того, должна предсказать его величину используя значения остальных признаков. Эти предсказания потом сравниваются с реальными ответами.

Способы формирования выборок:

- метод удерживания
- метод k-кратной перекрёстной проверки
- скользящий экзамен
- стратификация
- самонастройка

4) Как рассчитываются значения полноты и точности классификации?

Точность (precision) и полнота (recall) являются метриками которые используются при оценке большей части алгоритмов извлечения информации. Иногда они используются сами по себе, иногда в качестве базиса для производных метрик, таких как F-мера или R-Precision.

TP — истино-положительное решение;

TN — истино-отрицательное решение;

FP — ложно-положительное решение;

FN — ложно-отрицательное решение.

Тогда, точность и полнота определяются следующим образом:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

5) Как рассчитывается значение площади под кривой ошибок?

AUC-ROC (или ROC AUC) — площадь (Area Under Curve) под кривой ошибок (Receiver Operating Characteristic curve). Данная кривая представляет из себя линию от (0,0) до (1,1) в координатах True Positive Rate (TPR) и False Positive Rate (FPR):

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

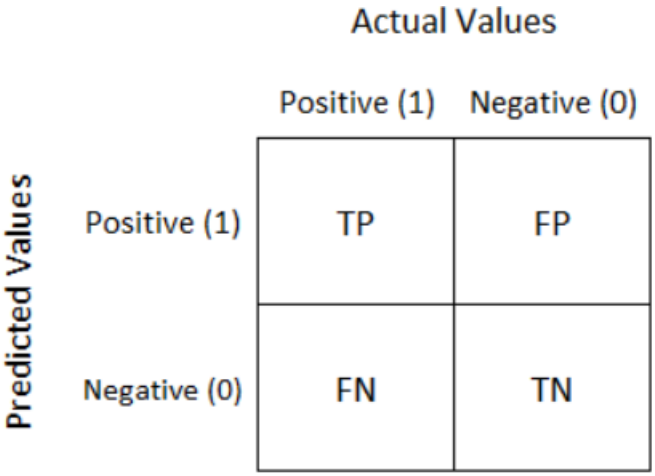
6) Что показывает и как рассчитывается матрица ошибок?

На практике значения точности и полноты гораздо более удобней рассчитывать с использованием матрицы неточностей (confusion matrix). В случае если количество классов относительно невелико (не более 100-150 классов), этот подход позволяет довольно наглядно представить результаты работы классификатора.

Матрица неточностей – это матрица размера N на N, где N — это количество классов. Столбцы этой матрицы резервируются за экспертными решениями, а строки за решениями классификатора.

Матрица ошибок позволяет оценить эффективность прогноза не только в качественном, но и в количественном выражении

Это таблица с 4 различными комбинациями прогнозируемых и фактических значений.



7) Алгоритм и особенности метода к-ближайших соседей.

Шаг 1 − Загружаем обучающий и тестовый dataset.

Шаг 2 − Выбираем значение K, то есть ближайшие точки данных. Оно может быть любым целым числом.

Шаг 3 - Вычисляем расстояние между тестовыми данными и каждой строкой обучающих данных с помощью любого из методов. Наиболее часто используемый метод вычисления расстояния - евклидов.

Шаг 4− Отсортировываем в порядке возрастания, основываясь на значении расстояния.

Шаг 5 − Алгоритм выбирает верхние K строк из отсортированного массива.

Шаг 6 − Назначаем класс контрольной точке на основе наиболее частого класса этих строк.

Особенности:

- Алгоритм прост и легко реализуем.

- Не чувствителен к выбросам.

- Нет необходимости строить модель, настраивать несколько параметров или делать дополнительные допущения.

- Алгоритм универсален. Его можно использовать для обоих типов задач: классификации и регрессии.

8) Алгоритм и особенности метода случайного леса.

Порядок действий в алгоритме

- Загрузите ваши данные.

- В заданном наборе данных определите случайную выборку.

- Далее алгоритм построит по выборке дерево решений.

- Дерево строится, пока в каждом листе не более n объектов, или пока не будет достигнута определенная высота.

- Затем будет получен результат прогнозирования из каждого дерева решений.

На этом этапе голосование будет проводиться для каждого прогнозируемого результата: мы выбираем лучший признак, делаем разбиение в дереве по нему и повторяем этот пункт до исчерпания выборки.

В конце выбирается результат прогноза с наибольшим количеством голосов. Это и есть окончательный результат прогнозирования.

Особенности:

— имеет высокую точность предсказания, на большинстве задач будет лучше линейных алгоритмов; точность сравнима с точностью бустинга

— практически не чувствителен к выбросам в данных из-за случайного сэмлирования

— не чувствителен к масштабированию значений признаков, связано с выбором случайных подпространств

— способен эффективно обрабатывать данные с большим числом признаков и классов

— одинаково хорошо обрабатывет как непрерывные, так и дискретные признаки

— редко переобучается, на практике добавление деревьев почти всегда только улучшает композицию, но на валидации, после достижения определенного количества деревьев, кривая обучения выходит на асимптоту

— для случайного леса существуют методы оценивания значимости отдельных признаков в модели

— хорошо работает с пропущенными данными; сохраняет хорошую точность, если большая часть данных пропущенна

— предполагает возможность сбалансировать вес каждого класса на всей выборке, либо на подвыборке каждого дерева