Лабораторная работа №1

Задание

Вариант №8

Вид классов: `moons`

Random state: `15`

noise: `0.2`

In [23]:
```
# Модуль numpy (сокращение от "Numerical Python") предоставляет функциональность для эффективной работы
# с массивами и математическими операциями на ними.
import numpy as np

# Модуль matplotlib.pyplot используется для создания графиков и визуализации данных.
# Он предоставляет множество функций для построения различных типов графиков.
import matplotlib.pyplot as plt

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

In [24]:
```
# А для отображения на графике области принятия решения - готовую функцию plot_2d_separator,
# которой нужно передать на вход объект classifier — модель классификатора и X — массив входных 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(())
```

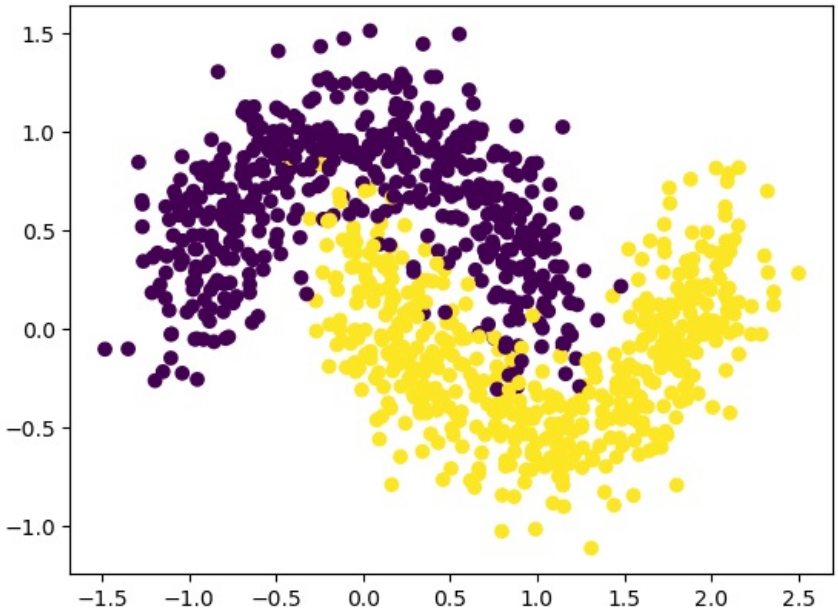
Генерация выборки

In [25]:
```
X, y = make_moons(n_samples=1000, shuffle=True, noise=0.2, random_state=15)
```

In [26]:
```
print('Координаты точек: ')
print(X[:15])
print('Метки класса: ')
print(y[:15])
```

```
Координаты точек:
[[ 1.7271961  -0.39285757]
 [-0.91801735  0.81910014]
 [-0.91532959 -0.05460812]
 [ 0.14537408  0.2064726 ]
 [ 0.95552152  0.20921022]
 [ 1.85825106 -0.35738814]
 [ 0.0761107   0.90867532]
 [-0.66311624  1.08115035]
 [ 0.13798809  0.98723143]
 [ 1.85704117  0.34111441]
 [ 0.17994761  0.15344022]
 [ 0.91666297  0.49152481]
 [ 1.25585707 -0.50035284]
 [ 1.11412853 -0.36151518]
 [-0.41601705  0.82276341]]
Метки класса:
[1 0 0 1 0 1 0 0 0 1 1 0 1 1 0]
```

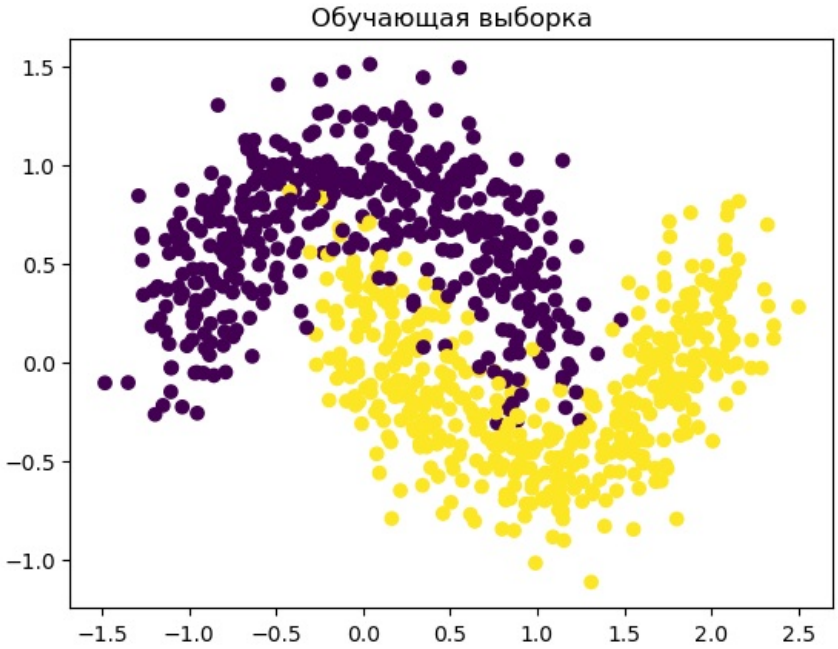
In [27]:
```
plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```



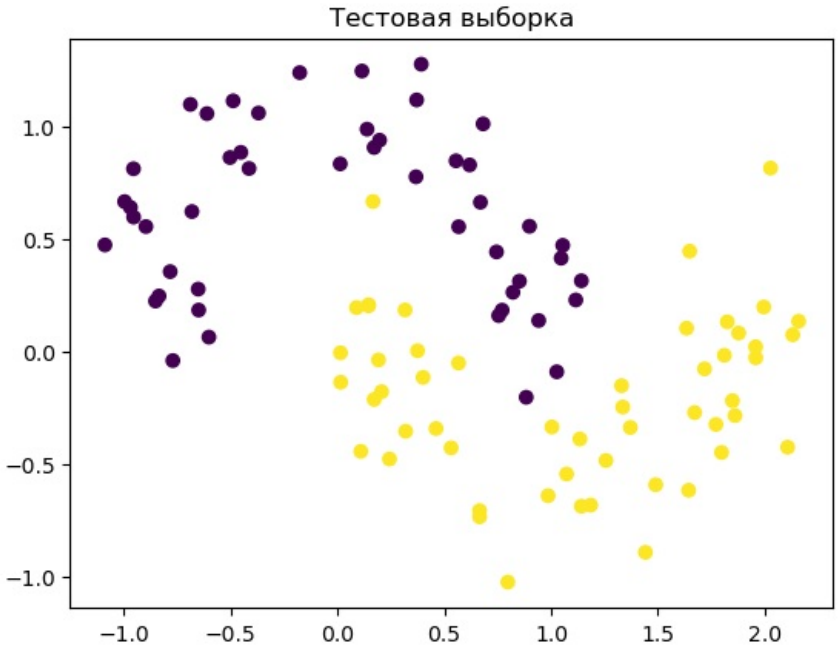
Разбитие выборки на обучающее и тестовое множество (90/10)

In [28]:
```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.10,
                                                    random_state=15)
```

In [29]:
```
plt.title('Обучающая выборка')
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
plt.show()
```



In [30]:
```
plt.title('Тестовая выборка')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test)
plt.show()
```



Классификация

In [31]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
```

In [32]:
```
def show_info(classifier, classifier_name, real_values, prediction_values):
    print(f'Метод классификации: {classifier_name}\n')

    # Выводим предсказанное и реальное значение
    print('Предсказанные и реальные значения:')
    print(prediction_values)
    print(real_values)

    # Выводим матрицу неточностей
    print('\nМатрица неточностей')
    print(confusion_matrix(real_values, prediction_values))

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

    # Выводим полноту
    print('\nПолнота: ')
    print(classification_report(real_values, prediction_values))

    # AUC ROC
    print(f'\nПлощадь под кривой: {roc_auc_score(real_values, prediction_values)}')

    plt.xlabel('Первый класс')
    plt.ylabel('Второй класс')
    plt.title(classifier_name.upper())
    plot_2d_separator(classifier, X, fill=True)
    plt.scatter(X[:, 0], X[:, 1], c=y, s=70)
```

Метод k-ближайших соседей (1)

In [33]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [34]:
```
knn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')

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

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

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (1)', y_test, prediction)
```

```
Метод классификации: ближайшие соседи (1)

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

Матрица неточностей
[[48  0]
 [ 1 51]]

Точность классификации: 0.99

Полнота:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        48
           1       1.00      0.98      0.99        52

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100


Площадь под кривой: 0.9903846153846154
```

Метод k-ближайших соседей (3)

In [35]:
```
knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')

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

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

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (3)', y_test, prediction)
```

```
Метод классификации: ближайшие соседи (3)

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

Матрица неточностей
[[47  1]
 [ 1 51]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        48
           1       0.98      0.98      0.98        52

    accuracy                           0.98       100
   macro avg       0.98      0.98      0.98       100
weighted avg       0.98      0.98      0.98       100


Площадь под кривой: 0.9799679487179487
```

Метод k-ближайших соседей (5)

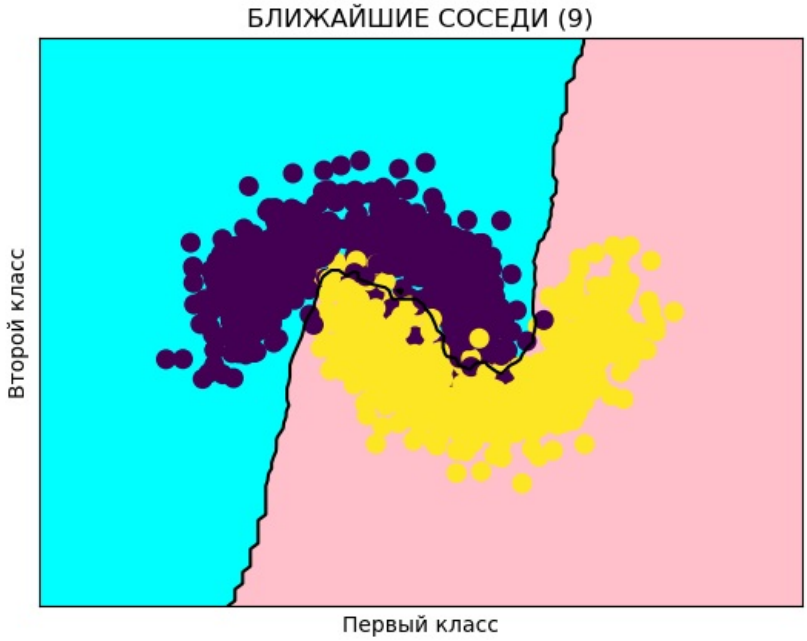
In [36]:
```
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

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

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

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (5)', y_test, prediction)
```

```
Метод классификации: ближайшие соседи (5)

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

Матрица неточностей
[[47  1]
 [ 1 51]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        48
           1       0.98      0.98      0.98        52

    accuracy                           0.98       100
   macro avg       0.98      0.98      0.98       100
weighted avg       0.98      0.98      0.98       100


Площадь под кривой: 0.9799679487179487
```

Метод k-ближайших соседей (9)

In [37]:
```
knn = KNeighborsClassifier(n_neighbors=9, metric='euclidean')

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

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

# Выводим сводную информацию
show_info(knn, 'ближайшие соседи (9)', y_test, prediction)
```

```
Метод классификации: ближайшие соседи (9)

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

Матрица неточностей
[[47  1]
 [ 1 51]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        48
           1       0.98      0.98      0.98        52

    accuracy                           0.98       100
   macro avg       0.98      0.98      0.98       100
weighted avg       0.98      0.98      0.98       100


Площадь под кривой: 0.9799679487179487
```



Наивный байесовский классификатор

In [38]:

```python
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()

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

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

# Выводим сводную информацию
show_info(nb, 'Наивный байесовский классификатор', y_test, prediction)
```

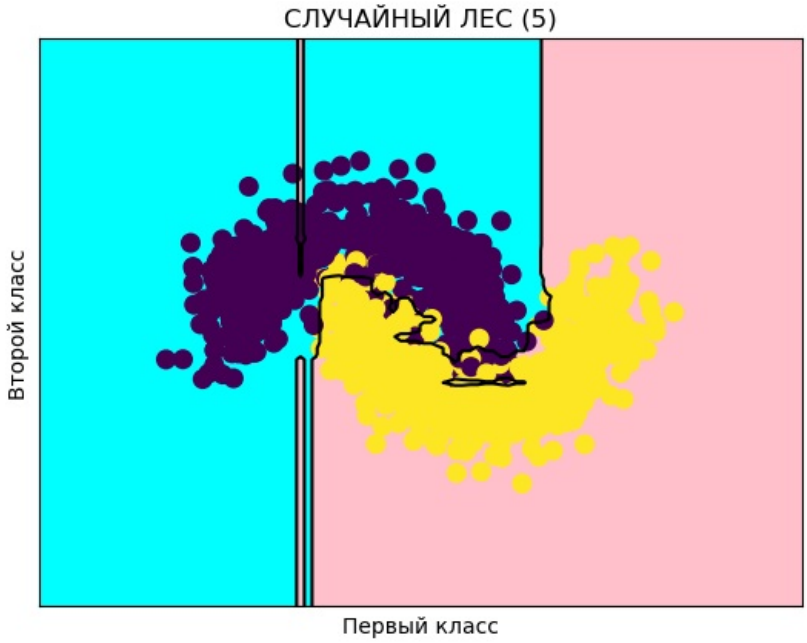
```
Метод классификации: Наивный байесовский классификатор

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

Матрица неточностей
[[38 10]
 [ 6 46]]

Точность классификации: 0.84

Полнота:
              precision    recall  f1-score   support

           0       0.86      0.79      0.83        48
           1       0.82      0.88      0.85        52

    accuracy                           0.84       100
   macro avg       0.84      0.84      0.84       100
weighted avg       0.84      0.84      0.84       100


Площадь под кривой: 0.8381410256410255
```

Случайный лес (5)

In [39]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [40]:
```
rfc = RandomForestClassifier(n_estimators=5)

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

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

# Выводим сводную информацию
show_info(rfc, 'случайный лес (5)', y_test, prediction)
```

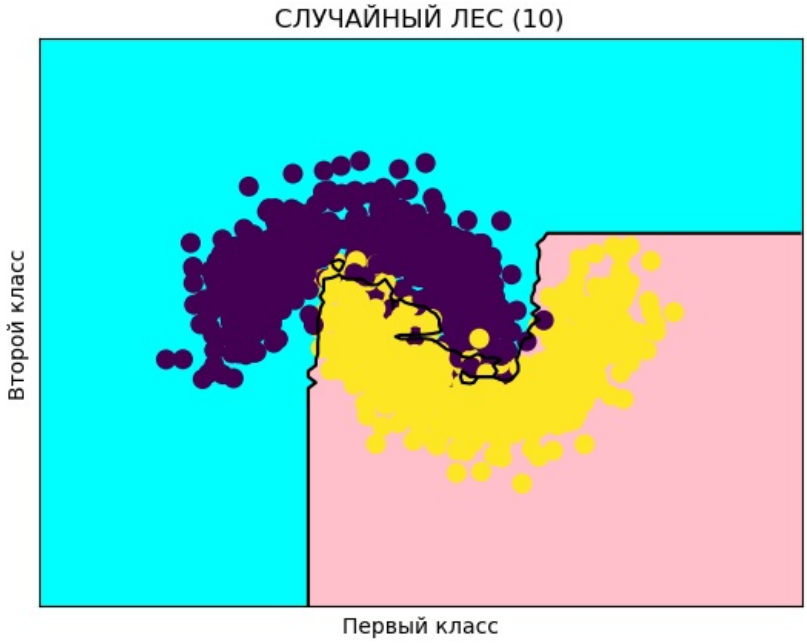
```
Метод классификации: случайный лес (5)

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

Матрица неточностей
[[47  1]
 [ 1 51]]

Точность классификации: 0.98

Полнота:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        48
           1       0.98      0.98      0.98        52

    accuracy                           0.98       100
   macro avg       0.98      0.98      0.98       100
weighted avg       0.98      0.98      0.98       100


Площадь под кривой: 0.9799679487179487
```



Случайный лес (10)

In [41]:
```
rfc = RandomForestClassifier(n_estimators=10)

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

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

# Выводим сводную информацию
show_info(rfc, 'случайный лес (10)', y_test, prediction)
```

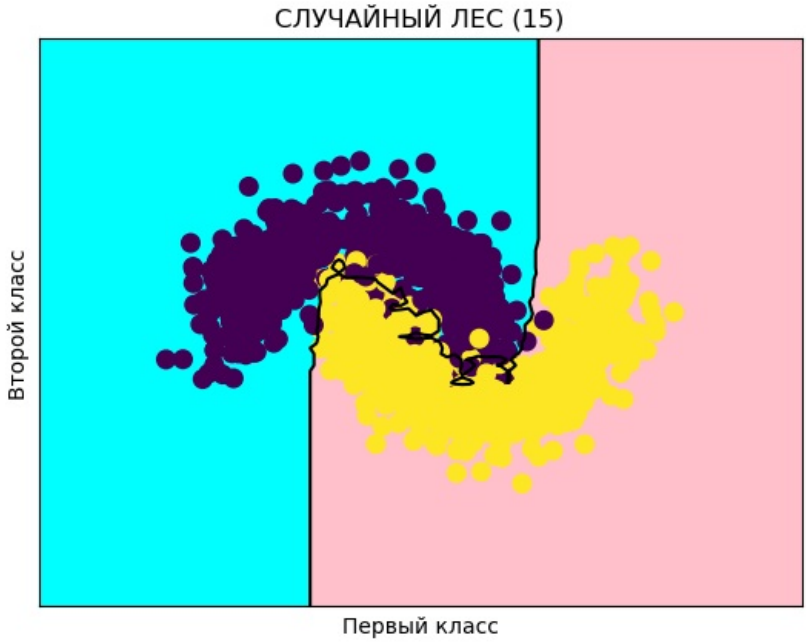
```
Метод классификации: случайный лес (10)

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

Матрица неточностей
[[48  0]
 [ 1 51]]

Точность классификации: 0.99

Полнота:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        48
           1       1.00      0.98      0.99        52

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100


Площадь под кривой: 0.9903846153846154
```



Случайный лес (15)

In [42]:
```
rfc = RandomForestClassifier(n_estimators=15)

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

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

# Выводим сводную информацию
show_info(rfc, 'случайный лес (15)', y_test, prediction)
```

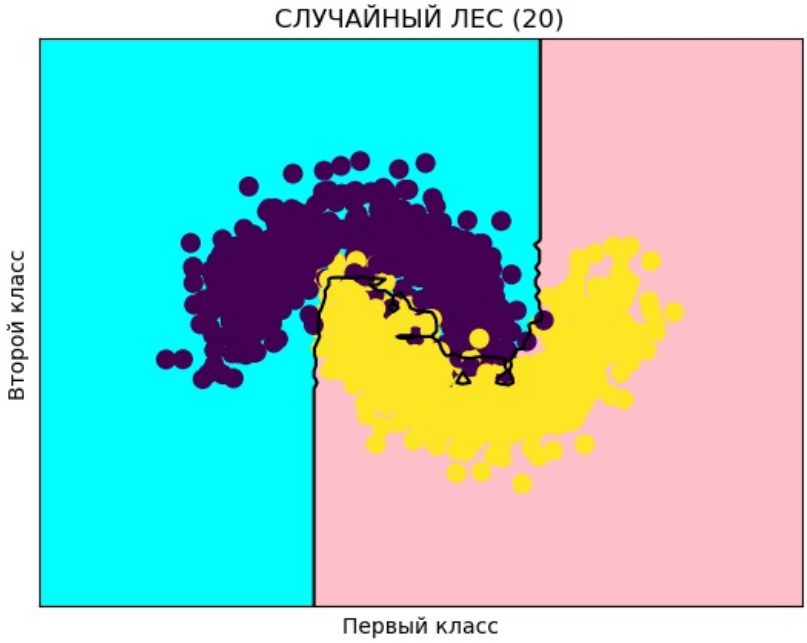
```
Метод классификации: случайный лес (15)

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

Матрица неточностей
[[48  0]
 [ 1 51]]

Точность классификации: 0.99

Полнота:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        48
           1       1.00      0.98      0.99        52

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100


Площадь под кривой: 0.9903846153846154
```



Случайный лес (20)

In [43]:
```
rfc = RandomForestClassifier(n_estimators=20)

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

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

# Выводим сводную информацию
show_info(rfc, 'случайный лес (20)', y_test, prediction)
```

```
Метод классификации: случайный лес (20)

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

Матрица неточностей
[[48  0]
 [ 1 51]]

Точность классификации: 0.99

Полнота:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        48
           1       1.00      0.98      0.99        52

    accuracy                           0.99       100
   macro avg       0.99      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100


Площадь под кривой: 0.9903846153846154
```



Случайный лес (50)

In [44]:
```python
rfc = RandomForestClassifier(n_estimators=50)

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

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

# Выводим сводную информацию
show_info(rfc, 'случайный лес (50)', y_test, prediction)
```

Метод классификации: случайный лес (50)

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

Матрица неточностей
```
[[47  1]
 [ 1 51]]
```

Точность классификации: 0.98

Полнота:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	48
1	0.98	0.98	0.98	52
accuracy			0.98	100
macro avg	0.98	0.98	0.98	100
weighted avg	0.98	0.98	0.98	100

Площадь под кривой: 0.9799679487179487