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 фактографических данных

Студент Александрук А.М.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.
к.т.н. доцент

Липецк 2023г.

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

Вид классов: blobs;

Random_state: 34;

Cluster std: 1.5;

noise: -;

Centers: 2.

Ход работы:

Подключаем все возможные библиотеки, которые потребуется для выполнения лабораторной работы. Все данные лабораторной работы и ее выполнение рассматриваются на рисунках

```
[1]: #Лабораторная работа 2
     # Вариант 1
     # Подключение библиотек
     from sklearn.datasets import make_blobs
     import numpy as np
     from sklearn.model_selection import train_test_split
     import matplotlib.pyplot as plt
     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
     from sklearn.naive_bayes import GaussianNB
     from sklearn.ensemble import RandomForestClassifier
```

Рисунок 1 – Импорт библиотек

Добавляем в код готовую функцию принятия решений.

```
[2]: # Для отображения на графике области принятия решения - готовую функцию, который на вход передают объект classifier
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 2 – Вставка в код готовой функции

[3]:
```
# Генерация выборки

X, y = make_blobs( centers= 2, cluster_std= 1.5,random_state=34)
# Массивы X и y
print('Координаты точек: ')
print(X[:15])
print('Метки класса: ')
print(y[:15])

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

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.25,
                                                    random_state=34)

# Разбиение выборки 75% обучающегося и  25% тестового множества
plt.title('Обучающая выборка')
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train)
plt.show()

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

# Обучение модели и классификация
# Импорт метода ближайших соседей
from sklearn.neighbors import KNeighborsClassifier
# Создание переменной  - модель классификатора
knn = KNeighborsClassifier(n_neighbors=1, metric='euclidean')
# Обучение модели
knn.fit(X_train, y_train)
# Оценка качества
prediction = knn.predict(X_test)
# Вывод результатов
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

Рисунок 3 – Генерация выборки 75% обучающегося и 25% тестового множества

Выводим результаты модели методом ближайших соседей.

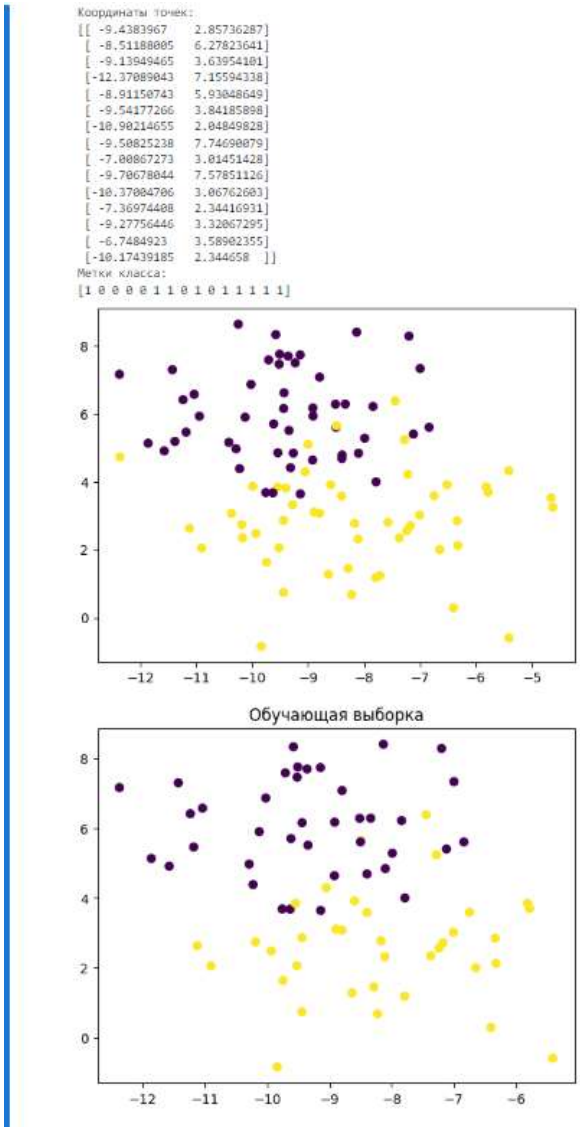


Рисунок 4 – Вывод результатов методом ближайших соседей 1

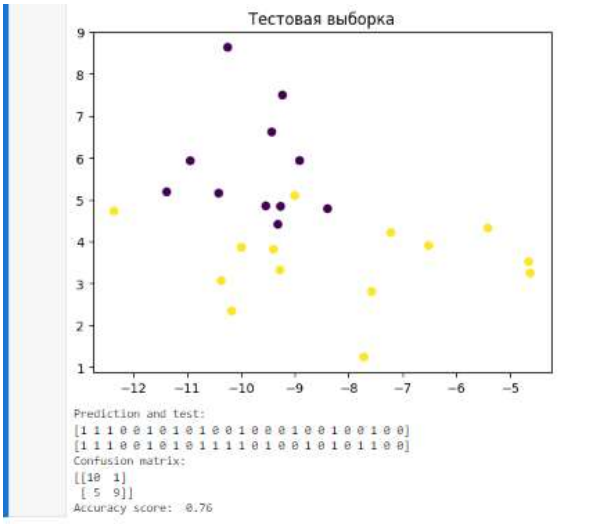


Рисунок 5 – Вывод результатов методом ближайших соседей 2

Создаем новую модель методом ближайших соседей (3,5,9) и выводим результаты

```
[16]: # Метод ближайших соседей (3)
knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')
# Обучение модели
knn.fit(X_train, y_train)
# Оценка качества
prediction = knn.predict(X_test)
# Вывод результатов
roc_auc_score(y_test, prediction)
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0]
Confusion matrix:
[[10  1]
 [ 4 10]]
Accuracy score:  0.8
```

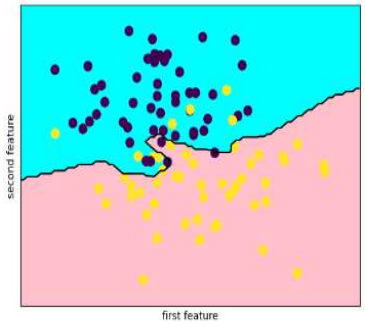


Рисунок 6 – Вывод результатов методом ближайших соседей (3)

```
[17]: # Метод ближайших соседей (5)
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
# Обучение модели
knn.fit(X_train, y_train)
# Оценка качества
prediction = knn.predict(X_test)
# Вывод результатов
roc_auc_score(y_test, prediction)
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 5  9]]
Accuracy score:  0.8
```

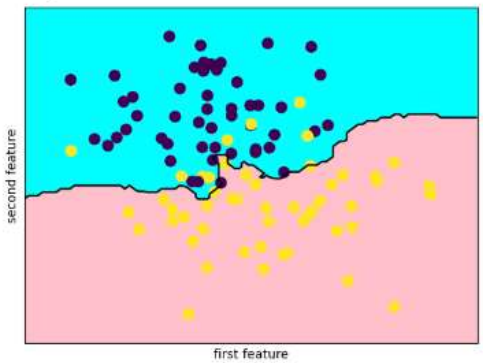


Рисунок 7 – Вывод результатов методом ближайших соседей (5)

```
[18]: # Метод ближайших соседей (9)
knn = KNeighborsClassifier(n_neighbors=9, metric='euclidean')
# Обучение модели
knn.fit(X_train, y_train)
# Оценка качества
prediction = knn.predict(X_test)
# Вывод результатов
roc_auc_score(y_test, prediction)
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 4 10]]
Accuracy score:  0.84
```

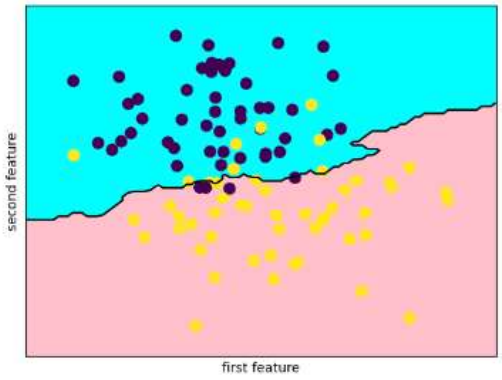


Рисунок 8 – Вывод результатов методом ближайших соседей (9)

Создаем модель наивного байесовского классификатора и выводим результаты.

```
[19]: # Метод наивного байесовского классификатора
nb = GaussianNB()
# Обучение модели
nb.fit(X_train, y_train)
# Оценка качества
prediction = nb.predict(X_test)
# Вывод результатов
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 2 12]]
Accuracy score:  0.92
```

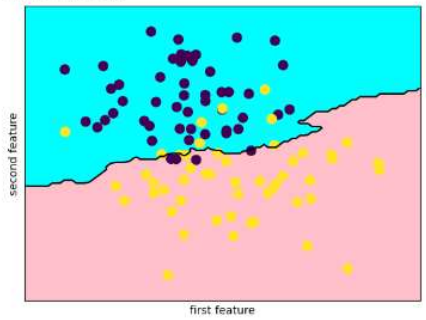


Рисунок 9 – Модель наивного байесовского классификатора

Создание модели случайных лесов (5,10,15,20,50) и вывод результатов.

```
[20]: # Метод случайного леса (5)
rdf = RandomForestClassifier(n_estimators=5)
# Обучаем модель данных
rdf.fit(X_train, y_train)
# Оцениваем качество модели
prediction = rdf.predict(X_test)
# Выводим сводную информацию
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 7  7]]
Accuracy score:  0.72
```

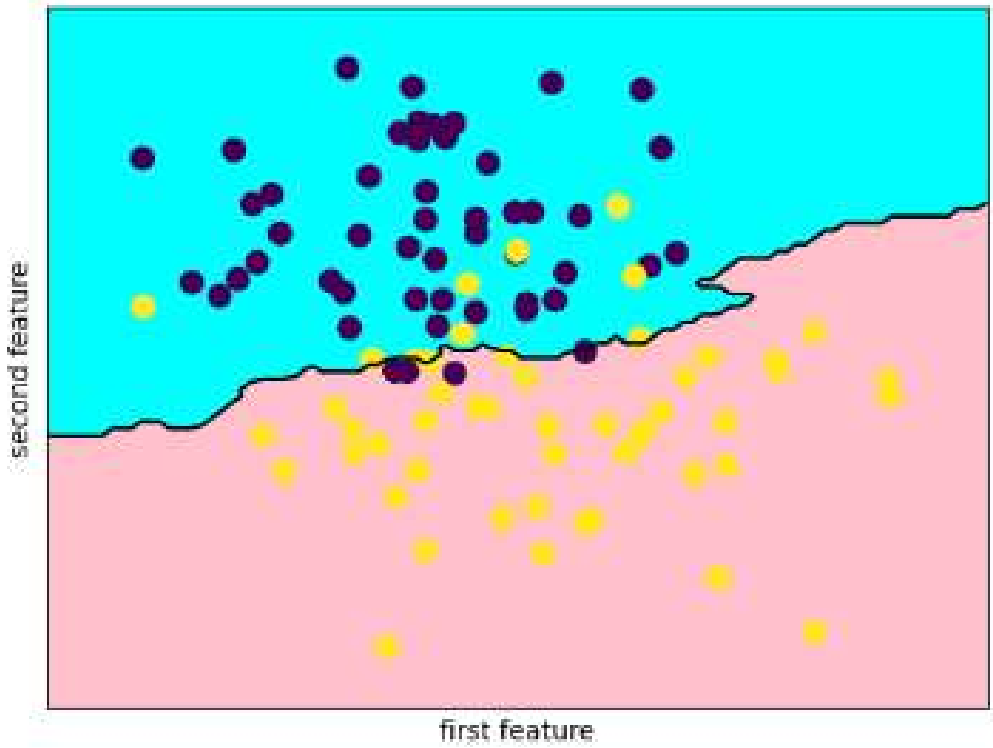


Рисунок 10 – Модель случайного леса (5)

[21]:
```
# Метод случайного леса (10)
rdf = RandomForestClassifier(n_estimators=10)
# Обучаем модель данных
rdf.fit(X_train, y_train)
# Оцениваем качество модели
prediction = rdf.predict(X_test)
# Выводим сводную информацию
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 1 1 0 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 4 10]]
Accuracy score:  0.84
```

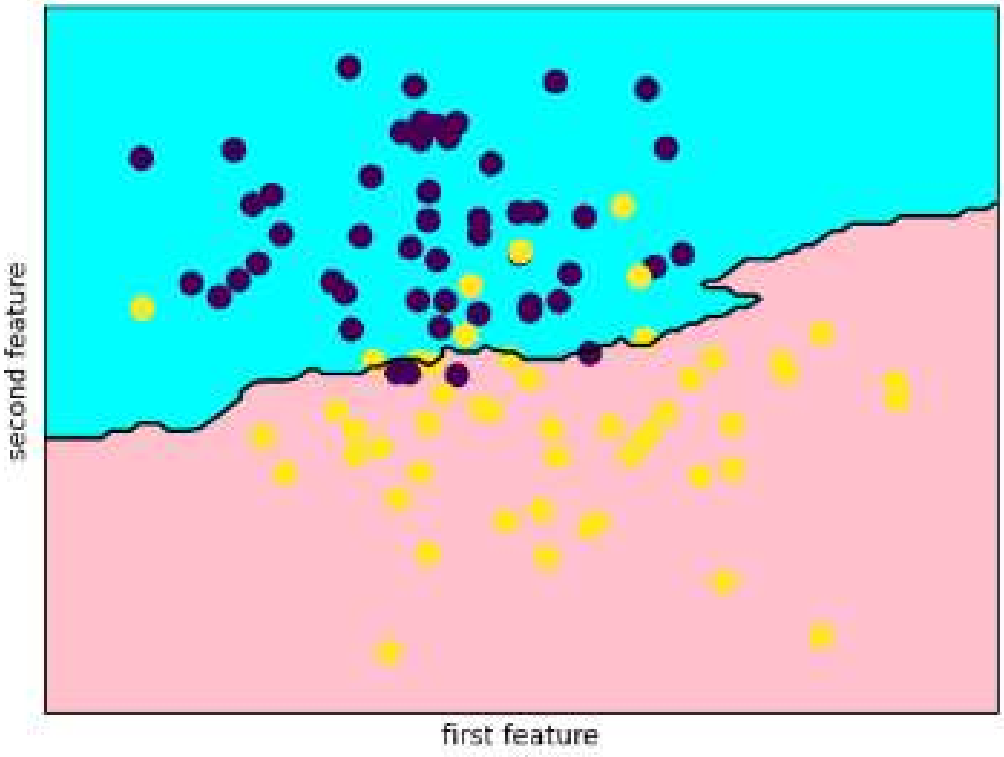


Рисунок 11 – Модель случайного леса (10)

[22]:
```
# Метод случайного леса (15)
rdf = RandomForestClassifier(n_estimators=15)
# Обучаем модель данных
rdf.fit(X_train, y_train)
# Оцениваем качество модели
prediction = rdf.predict(X_test)
# Выводим сводную информацию
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 0 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 4 10]]
Accuracy score:  0.84
```

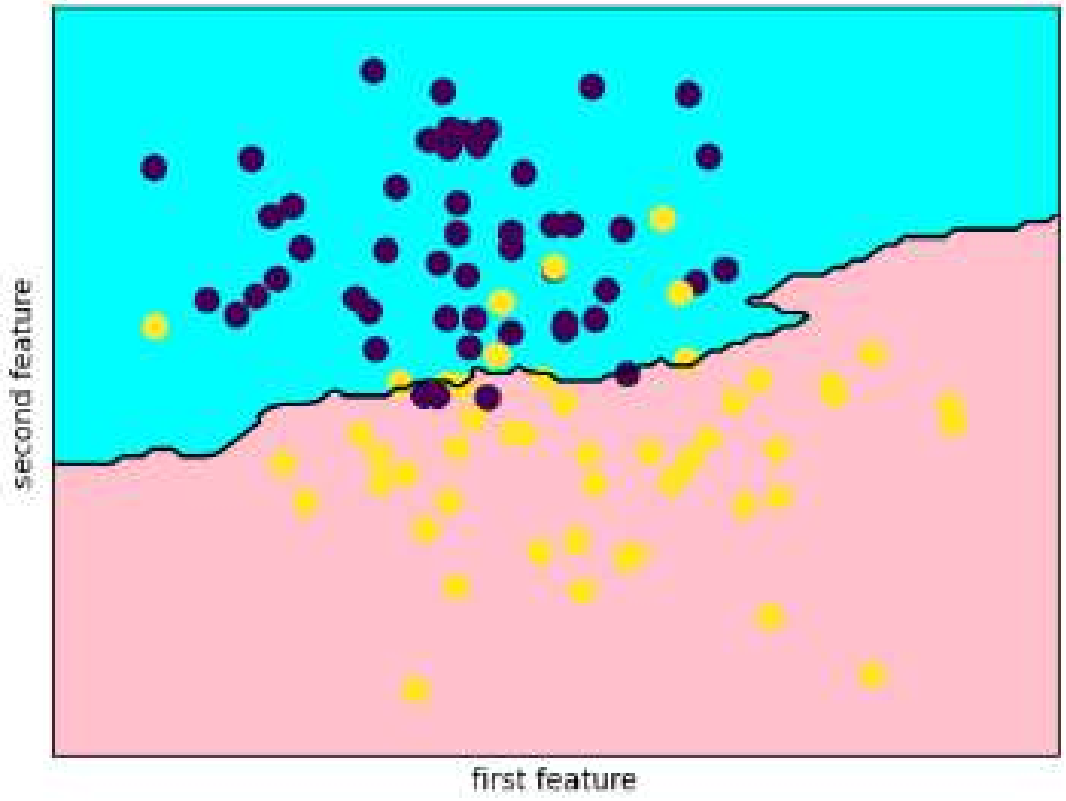


Рисунок 12 – Модель случайного леса (15)

[23]:
```
# Метод случайного леса (20)
rdf = RandomForestClassifier(n_estimators=20)
# Обучаем модель данных
rdf.fit(X_train, y_train)
# Оцениваем качество модели
prediction = rdf.predict(X_test)
# Выводим сводную информацию
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 3 11]]
Accuracy score:  0.88
```

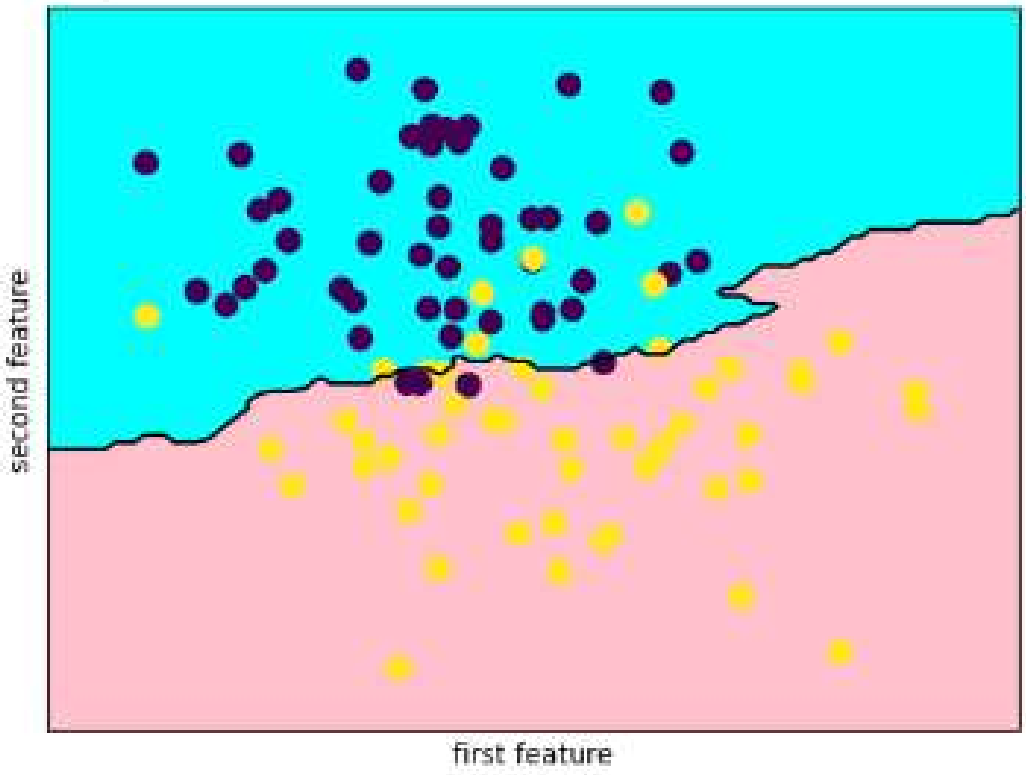


Рисунок 13 – Модель случайного леса (20)

[24]:
```
# Метод случайного леса (50)
rdf = RandomForestClassifier(n_estimators=50)
# Обучаем модель данных
rdf.fit(X_train, y_train)
# Оцениваем качество модели
prediction = rdf.predict(X_test)
# Выводим сводную информацию
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)
print ('Prediction and test: ')
print (prediction)
print (y_test)
print ('Confusion matrix: ')
print (confusion_matrix(y_test, prediction))
print ('Accuracy score: ', accuracy_score(prediction, y_test))
```

```
Prediction and test:
[1 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0]
Confusion matrix:
[[11  0]
 [ 3 11]]
Accuracy score:  0.88
```

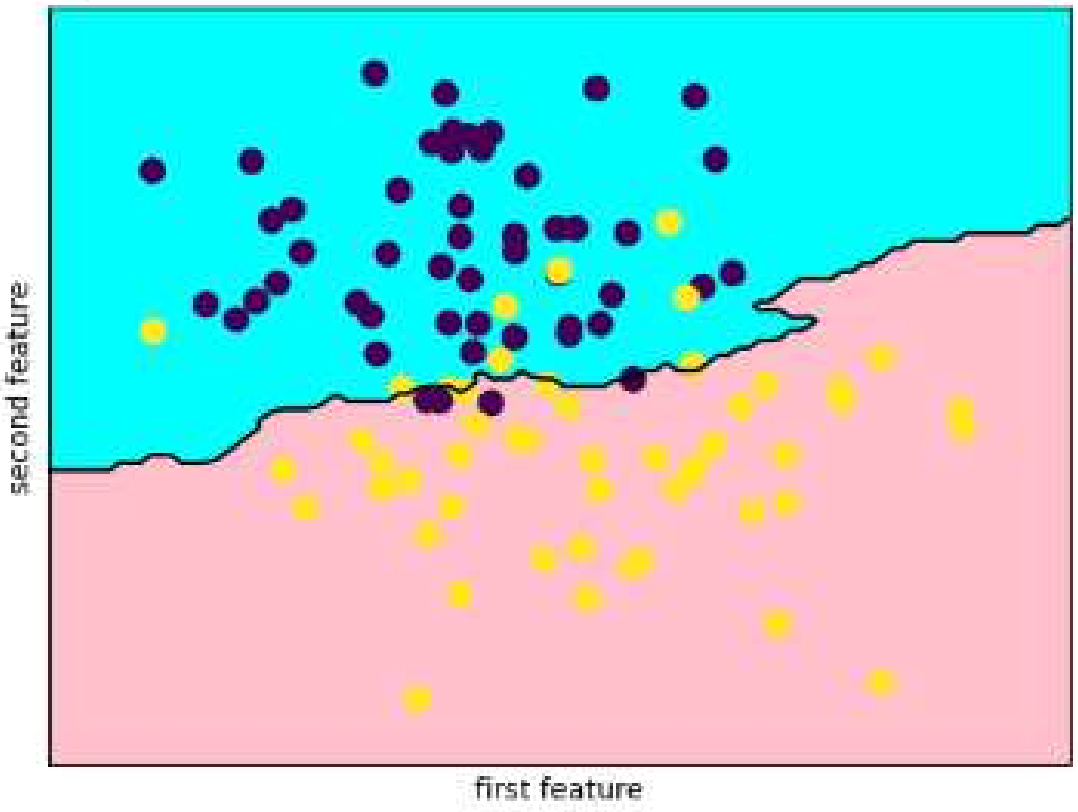
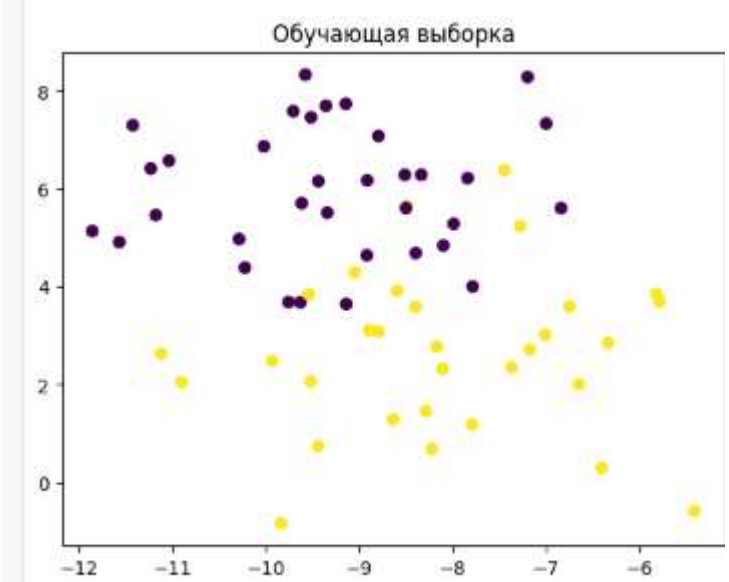
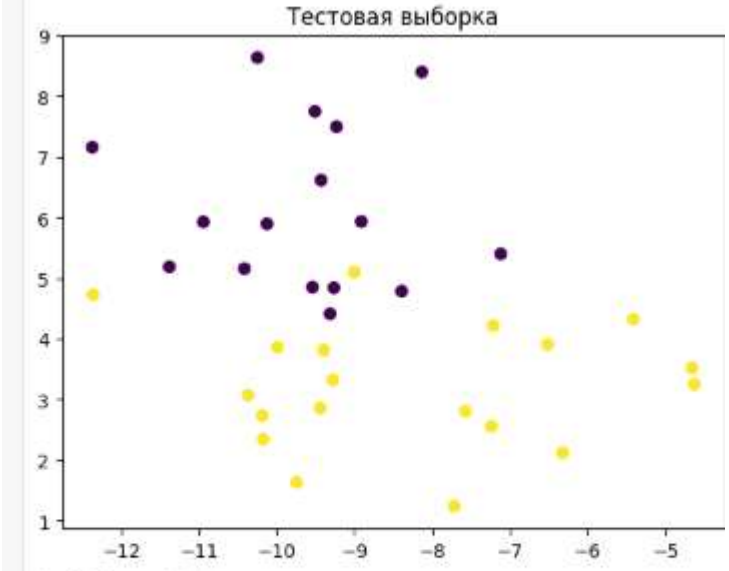


Рисунок 14 – Модель случайного леса (50)

Тестовая и обучающаяся выборка при 35% и 65%.






```
Prediction and test:
[1 1 1 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 1 1]
[1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 1 0 1 0 1 1 0 0 0 0 0 0 1 1 0 1 1 0 1]
Confusion matrix:
[[14  2]
 [ 5 14]]
Accuracy score:  0.8
```

Рисунок 15 – Выборка при 35% тестового и 65% обучающегося множества

Выведем все результаты в таблицы.

Таблица 1. Вывод результатов при обучающем множестве 75%

Методы (параметры)	**Характеристика результатов**
Метод k-ближайших соседей (1)	Точность: 0,76 Площадь по кривой: 0,76
Метод k-ближайших соседей (3)	Точность: 0.8 Площадь по кривой:0.8
Метод k-ближайших соседей (5)	Точность: 0.8 Площадь по кривой:0.8
Метод k-ближайших соседей (9)	Точность: 0.84 Площадь по кривой: 0.84
Наивный байесовский классификатор	Точность:0.92 Площадь по кривой:0.92
Случайный лес (5)	Точность:0.72 Площадь по кривой:0.72
Случайный лес (10)	Точность:0.84 Площадь по кривой:0.84
Случайный лес (15)	Точность:0.84 Площадь по кривой:0.84
Случайный лес (20)	Точность:0.88 Площадь по кривой:0.88
Случайный лес (50)	Точность:0.88 Площадь по кривой:0.88

Таблица 2. Вывод результатов при обучающем множестве 65%

Методы (параметры)	**Характеристика результатов**
Метод k-ближайших соседей (1)	Точность: 0,8 Площадь по кривой: 0,8
Метод k-ближайших соседей (3)	Точность: 0.82 Площадь по кривой:0.82
Метод k-ближайших соседей (5)	Точность: 0.88 Площадь по кривой:0.88
Метод k-ближайших соседей (9)	Точность: 0.91 Площадь по кривой: 0.91
Наивный байесовский классификатор	Точность:0.91 Площадь по кривой:0.91
Случайный лес (5)	Точность:0.91 Площадь по кривой:0.91
Случайный лес (10)	Точность:0.91 Площадь по кривой:0.91
Случайный лес (15)	Точность:0.88 Площадь по кривой:0.88
Случайный лес (20)	Точность:0.85 Площадь по кривой:0.85
Случайный лес (50)	Точность:0.91 Площадь по кривой:0.91

Код программы: https://github.com/lipadirka/Prikladnie-intelektualn-systems.git

Вывод

В ходе выполнения лабораторной работы в результате мы определили максимальную точность классификации 0,92, наибольшая площадь кривой 0,92. При условии того, что процент обучающейся выборки был 75% при использовании метода наивного байесовского классификатора.