**Липецкий государственный технический университет**

Факультет Автоматизации и Информатики

Кафедра Автоматизированных систем управления

Лабораторная работа №1

по прикладным интеллектуальным системам и экспертным системам

Бинарная классификация фактографических данных

Студент Ледовских Д.В.

Группа М-ИАП-22

Проверил

Доцент Кургасов В.В.

Липецк 2022г.

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 | Вид классов | Random\_state | noise |
| 6 | moons | 23 | 0,3 |

Ход работы

Для данной лабораторной работы нам необходимы библиотеки: numpy, matplotlib, sklearn.

NumPy нужен для работы с большими многомерными массивами и матрицами, в данной библиотеке реализованы различные математические функции и операции с массивами данных.

Matplotlib разработан для визуализации данных в понятном пользователю виде за счет обширного функционала настроек итогового изображение, представленного двумерной или трехмерной графикой.

Scikit-learn – 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.

Команда для установки необходимых библиотек:

pip install numpy

pip install matplotlib

pip install scikit-learn

Для выполнения лабораторной работы в среде Jupiter Notebook были импортированы следующие модули:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_moons

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_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

from sklearn.metrics import roc\_auc\_score

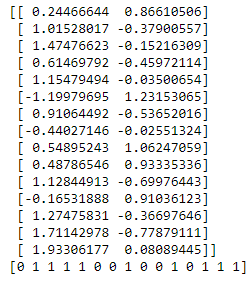


Рисунок 1 - первые 15 элементов выборки

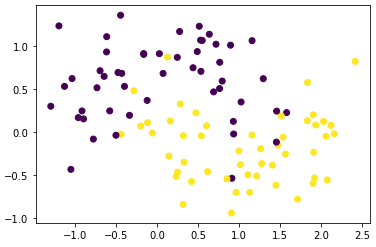


Рисунок 2 – сгенерированная выборка на графике

Функция для визуализация данных выполнена следующим образом:

def plot\_2d\_separator(classifier, X, fill=False, line=True, ax=None, eps=None):

if eps is None:

eps = 1.0

X.std() / 2

x\_min, x\_max = 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:

# no decision\_function

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(())

Разбиение данных на обучающую (train) и тестовую (test) выборки в пропорции 75% - 25% выполнено следующим способом:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.35, random\_state=1)

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

plt.show()

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

plt.show()

Где X\_train и y\_train данные для обучения, X\_test и y\_test данные для тестирования.

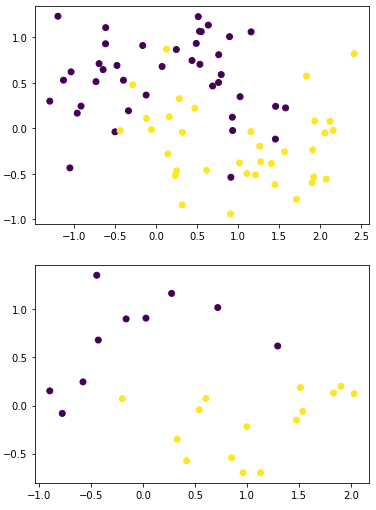


Рисунок 3 – обучающая и тестовая выборки на графике

Классификация данных

Для начала реализуем универсальную функцию, которая будет выводить характеристики классификации на основе данных для теста и полученных в результате предсказания.

Код функции «show\_metrics»:

def show\_metrics(prediction, y\_test, classifier, X, y):

prediction = knn.predict(X\_test)

print("Prediction: ", prediction)

print("True: ", y\_test)

print("Confusion matrix: \n", confusion\_matrix(y\_test, prediction))

print('Accuracy score: ', accuracy\_score(prediction, y\_test))

print(classification\_report(y\_test, prediction))

roc\_auc\_score(y\_test, prediction)

plt.xlabel("first feature")

plt.ylabel("second feature")

print("AUC ROC: ", roc\_auc\_score(y\_test, prediction))

plot\_2d\_separator(knn, X, fill=True)

plt.scatter(X[:, 0], X[:, 1], c=y, s=70)

plt.show()

Программный код для методов классификации:

1) Метод к-ближайших соседей:

neighbors = [1, 3, 5, 9]

for n in neighbors:

knn = KNeighborsClassifier(n\_neighbors=n, metric='euclidean')

knn.fit(X\_train, y\_train)

prediction = knn.predict(X\_test)

print("n\_neightbors = ", n)

show\_metrics(prediction, y\_test, knn, X, y)

2) Наивный байесовский метод:

gnb = GaussianNB()

gnb.fit(X\_train, y\_train)

prediction = gnb.predict(X\_test)

show\_metrics(prediction, y\_test, gnb, X, y)

3) Случайный лес:

estimators = [5, 10, 15, 20, 50]

for e in estimators:

rf = RandomForestClassifier(n\_estimators=e)

rf.fit(X\_train, y\_train)

prediction = rf.predict(X\_test)

show\_metrics(prediction, y\_test, rf, X, y)

Истинные и предсказанные метки классов, матрица ошибок (confusion matrix), значения полноты, точности, f1-меры и аккуратности, значение площади под кривой ошибок (AUC ROC), область принятия решений по каждому классу на графике для методов к-ближайших соседей, наивного Байеса и случайного леса представлены на рисунках 4, 5 и 6 соответственно.

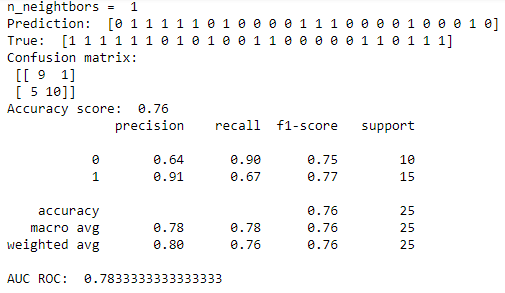


Рисунок 4 – результаты классификации для метода к-ближайших соседей при n\_neightbors = 1

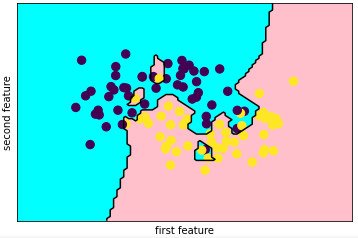


Рисунок 5 – область принятия решений для метода к-ближайших соседей при n\_neightbors = 1

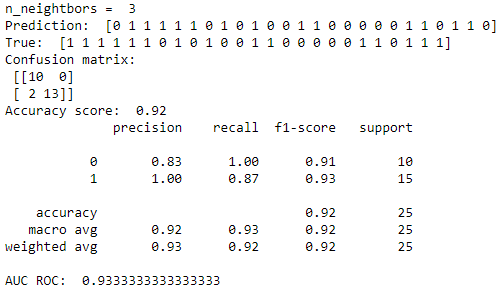


Рисунок 6 – результаты классификации для метода к-ближайших соседей при n\_neightbors = 3

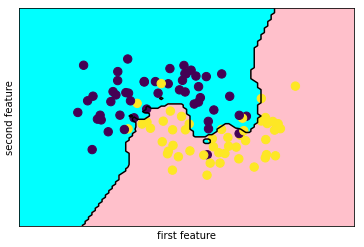


Рисунок 7 – область принятия решений для метода к-ближайших соседей при n\_neightbors = 3

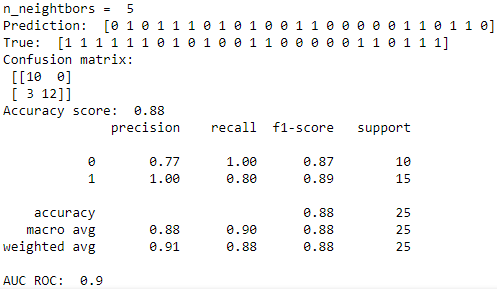


Рисунок 8 – результаты классификации для метода к-ближайших соседей при n\_neightbors = 5

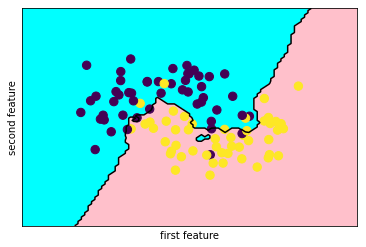


Рисунок 9 – область принятия решений для метода к-ближайших соседей при n\_neightbors = 5

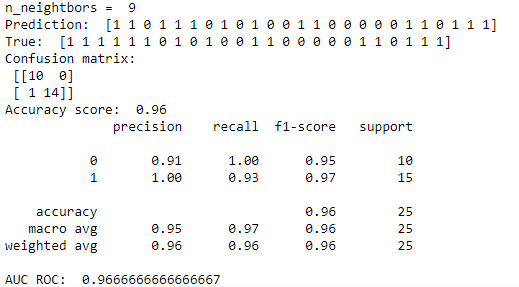


Рисунок 10 – результаты классификации для метода к-ближайших соседей при n\_neightbors = 9

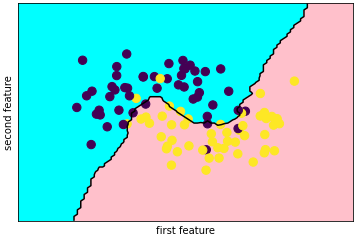


Рисунок 11 – область принятия решений для метода к-ближайших соседей при n\_neightbors = 9

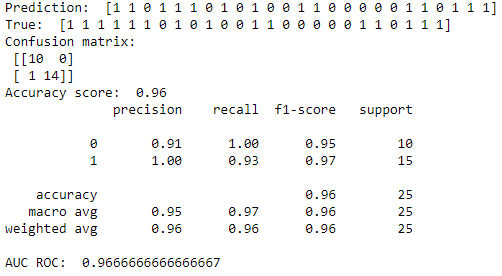


Рисунок 12 – результаты классификации для метода наивного Байеса

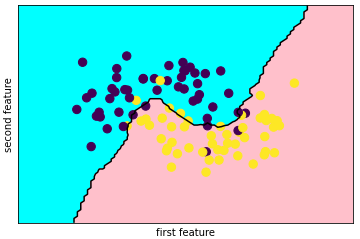


Рисунок 13 – область принятия решений для метода наивного Байеса

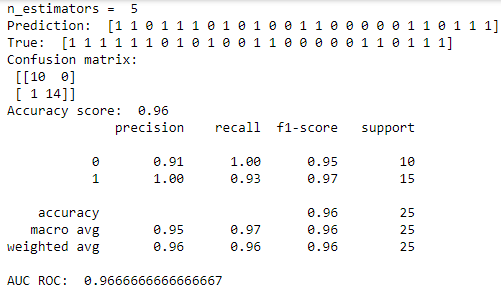


Рисунок 14 – результаты классификации для метода случайного леса при n\_estimators = 5

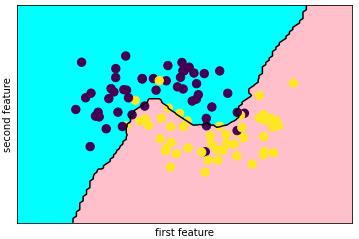


Рисунок 15 – область принятия решений для метода случайного леса при n\_estimators = 5

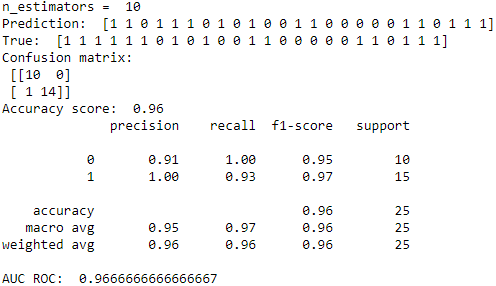


Рисунок 16 – результаты классификации для метода случайного леса при n\_estimators = 10

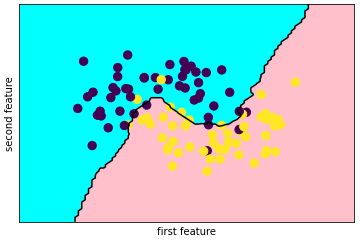


Рисунок 17 – область принятия решений для метода случайного леса при n\_estimators = 10

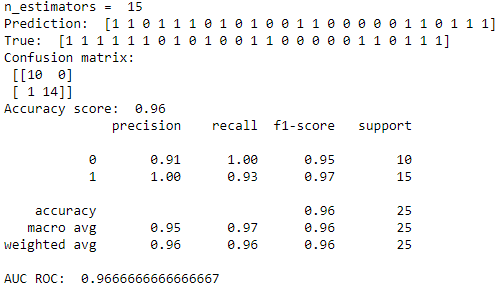


Рисунок 18 – результаты классификации для метода случайного леса при n\_estimators = 15

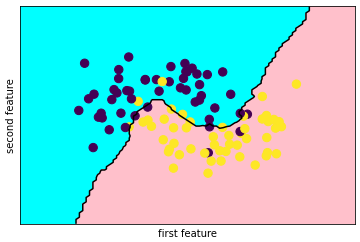


Рисунок 19 – область принятия решений для метода случайного леса при n\_estimators = 15

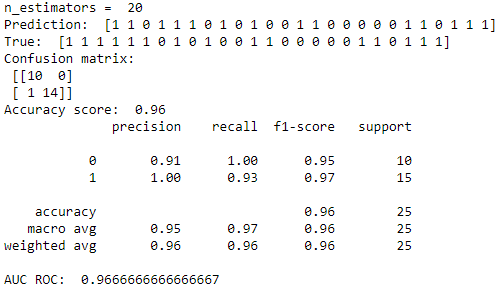


Рисунок 20 – результаты классификации для метода случайного леса при n\_estimators = 20

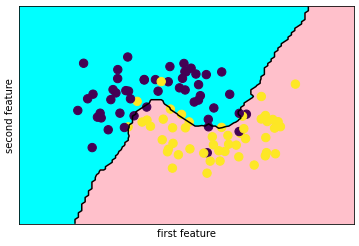


Рисунок 21 – область принятия решений для метода случайного леса при n\_estimators = 20

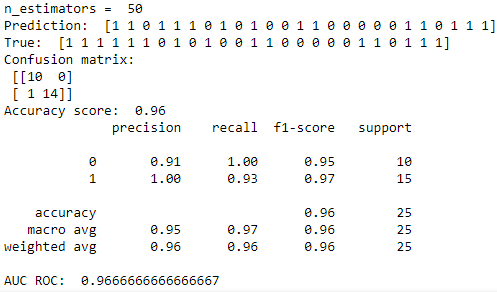


Рисунок 22 – результаты классификации для метода случайного леса при n\_estimators = 50

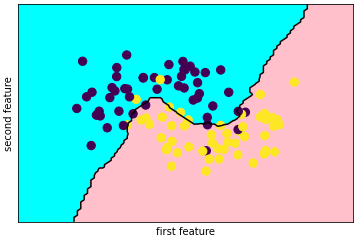


Рисунок 23 – область принятия решений для метода случайного леса при n\_estimators = 50

Таблица 1 – Результаты классификации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод | Точность | Значение ошибки | Точность классификации класса | Чувствительность | Специфичность |
| Метод к-ближайших соседей n\_neightbors = 1 | 0.76 | 0,783 | 0,78  0,8 | 0,78  0,76 | 0,76  0,76 |
| Метод к-ближайших соседей n\_neightbors = 3 | 0,92 | 0,9333 | 0,92  0,93 | 0,93  0,92 | 0,92  0,92 |
| Метод к-ближайших соседей n\_neightbors = 5 | 0,88 | 0,9 | 0,88  0,91 | 0,9  0,88 | 0,88  0,88 |
| Метод к-ближайших соседей n\_neightbors = 9 | 0,96 | 0,9666 | 0,95  0,96 | 0,97  0,96 | 0,96  0,96 |
| Наивный Байесовский метод | 0,96 | 0,9666 | 0,95  0,96 | 0,97  0,96 | 0,96  0,96 |
| Метод случайного леса n\_estimators=5 | 0,96 | 0,9666 | 0,95  0,96 | 0,97  0,96 | 0,96  0,96 |
| Метод случайного леса n\_estimators=10 | 0,96 | 0,9666 | 0,95  0,96 | 0,97  0,96 | 0,96  0,96 |
| Метод случайного леса n\_estimators = 15 | 0,96 | 0,9666 | 0,95  0,96 | 0,97  0,96 | 0,96  0,96 |
| Метод случайного леса n\_estimators = 20 | 0,96 | 0,9666 | 0,95  0,96 | 0,97  0,96 | 0,96  0,96 |
| Метод случайного леса n\_estimators = 50 | 0,96 | 0,9666 | 0,95  0,96 | 0,97  0,96 | 0,96  0,96 |

Наиболее подходящие методы для классификации данных по результатам тестов оказались метод к-ближайших соседей при n\_neightbors = 9, Наивный Байесовский метод и метод случайного леса при n\_estimators = 5, 10, 15, 20 и 50.

Худшие результаты показал метод к-ближайших соседей при n\_neightbors = 1.

Вывод

В ходе лабораторной работы были получены практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook, загрузки данные, обучения классификаторов и проведения классификации, оценки точности полученных моделей.