Лабораторная работа № 5   
Построение деревьев решений для задачи классификации

1. Задание на лабораторную работу

А. Выбрать среду программирования для языка Python.

Б. Получить набор данных из 200 наблюдений с параметрами, согласно номеру варианта (табл. 1.1). Построить деревья решений и выполнить их анализ. При этом необходимо:

* разделить исходную выборку на две равные части: для обучения и для тестирования;
* обучить дерево решений и вывести его на экран;
* используя функцию рисования решающих поверхностей из лабораторной работы № 3, вывести на экран решающие поверхности и данные для построенного дерева решений для каждой пары признаков;
* оценить точность модели на тестирующей выборке;
* в случае, если точность модели менее 0,9, изменяя значение параметров criterion, max\_depth, min\_samples\_split, min\_samples\_leaf попытаться ее повысить.

В. Выполнить анализ полученных на каждом этапе результатов и оформить отчет по лабораторной работе.

***Таблица 1.1***

**Варианты индивидуальных заданий**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варианта** | **Количество  кластеров** | **Количество признаков** | **Стандартное отклонение** |
| 1 | 2 | 5 | 3 |
| 2 | 2 | 5 | 4 |
| 3 | 2 | 5 | 5 |
| 4 | 2 | 5 | 6 |
| 5 | 2 | 6 | 3 |
| 6 | 2 | 6 | 4 |
| 7 | 2 | 6 | 5 |
| 8 | 2 | 6 | 6 |
| 9 | 3 | 4 | 3 |
| 10 | 3 | 4 | 4 |
| 11 | 3 | 4 | 5 |
| 12 | 3 | 4 | 6 |
| 13 | 3 | 5 | 3 |
| 14 | 3 | 5 | 4 |
| 15 | 3 | 5 | 5 |

2. Пример выполнения

Подключим необходимые библиотеки и модули:

import numpy as np

import pandas as pd

import seaborn as sns

from sklearn import tree

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.model\_selection import train\_test\_split

Получим исходные данные для числа классов N=3:

N=3, D=5

X, y = make\_blobs(n\_samples=100, centers=N, n\_features=D, random\_state=None, cluster\_std=2)

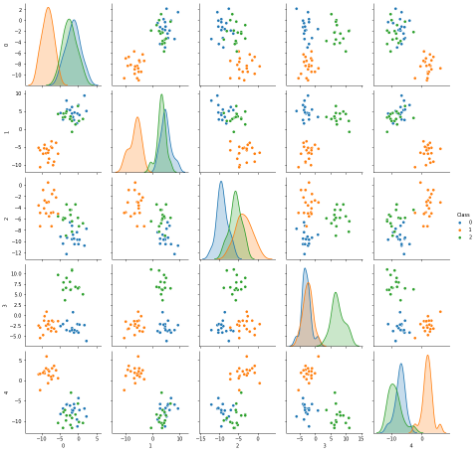
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5)

df=pd.DataFrame(X\_train)

df['Class']=y\_train

Построим диаграммы рассеяния:

sns.pairplot(df, hue='Class')



Обучим дерево решений:

model=tree.DecisionTreeClassifier(criterion="gini", min\_samples\_split=10, max\_depth=5)

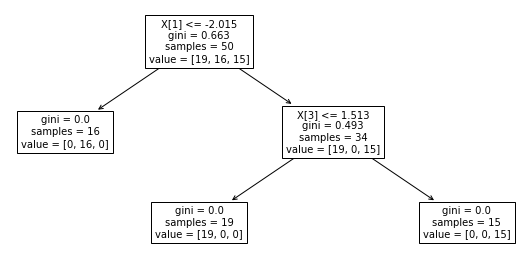
model.fit(X\_train,y\_train)

Выведем дерево решений:

plt.figure(figsize=(12,12))

tree.plot\_tree(model, fontsize=10)

plt.show()



Немного модифицируем и используем функцию отрисовки решающих поверхностей из л.р. 3, построим и выведем на экран деревья решений и решающие поверхности для каждой пары признаков:

def DT(field1, field2, k): #процедура рисования "решающих поверхностей"

splot=plt.subplot(D,D,k)

x=df[[field1, field2]].values #определим данные

df['Class']=y\_train #определим результирующий признак

model\_pair=tree.DecisionTreeClassifier(criterion="gini", min\_samples\_split=10, max\_depth=5)

model\_pair.fit(x,y\_train)

#определим минимальные и максимальные значения в данных для большей информативности диаграмм

min1, max1 = df[field1].min()-1, df[field1].max()+1

min2, max2 = df[field2].min()-1, df[field2].max()+1

#определим все строки и столбцы области отрисовки

X = np.linspace(min1, max1, 200)

Y = np.linspace(min2, max2, 200)

X, Y = np.meshgrid(X, Y)

# #определим значения "решающих поверхностей"

ZZ = np.array([model\_pair.predict(np.array([[xx, yy]])) for xx, yy in zip(np.ravel(X), np.ravel(Y))])

Z = ZZ.reshape(X.shape) #преобразуем значения в вектор

#отрисуем поверхности и данные

plt.pcolormesh(X, Y, Z,cmap='Pastel1')

#Выведем значения

plt.scatter(df[field1],df[field2],c=y\_train)

#оценка точности модели

plt.title(model\_pair.score(x,y\_train))

return splot

plt.figure(figsize=(10, 10), facecolor='white')

plt.suptitle('Decision tree areas ',y=1, fontsize=15)

k=0

for i in range(D):

for j in range (D):

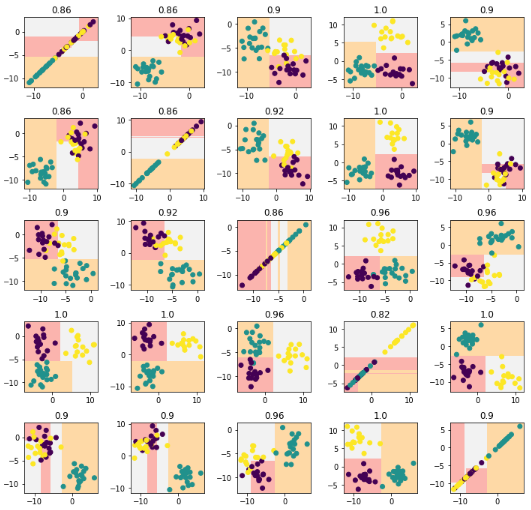
k=k+1

splot=DT(i, j, k)

plt.tight\_layout()

plt.subplots\_adjust(top=0.92)

plt.show()



Оценим точность модели на тестирующей выборке:

model.score(X\_test,y\_test)



3. Темы для самостоятельной подготовки

1. Назначение деревьев решений.
2. Основное отличие алгоритмов построения деревьев решений.
3. Алгоритм C4.5.
4. Алгоритм CART.
5. Способы борьбы с переобучением модели.