**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №7**

**по дисциплине «Машинное обучение»**

Тема: **Классификация (Байесовские методы, деревья)**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Кирьянов Д. И. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

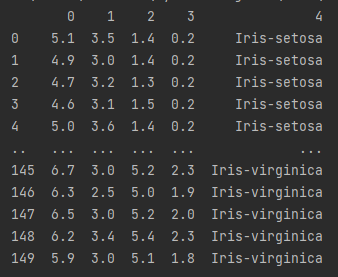
2021

**Цель работы.**

Ознакомиться с методами классификации модуля Sklearn.

**Ход работы.**

1. **Загрузка данных**
   1. Произведена загрузка данных. Создан Python скрипт. Данные загружены в датафрейм.



* 1. Выделены данные и их метки, преобразованы тексты меток к числам, разбита выборка на обучающую и тестовую.

1. **Байесовские методы**
   1. Проведена классификация наблюдений наивным байесовским методом. Количество найденных неправильно классифицированных наблюдений = 4.

Атрибуты:

class\_count\_ - количество обучаемых выборок, наблюдаемых в каждом классе.

class\_prior\_ - вероятность каждого класса.

classes\_ - лейблы класса известные классификатору.

epsilon\_ - абсолютная аддитивная величина отклонений.

n\_features\_in\_ - количество видимых деталей во время посадки.

feature\_names\_in\_ - названия видимых особенностей во время посадки.

var\_ - дисперсия каждого объекта каждого класса.

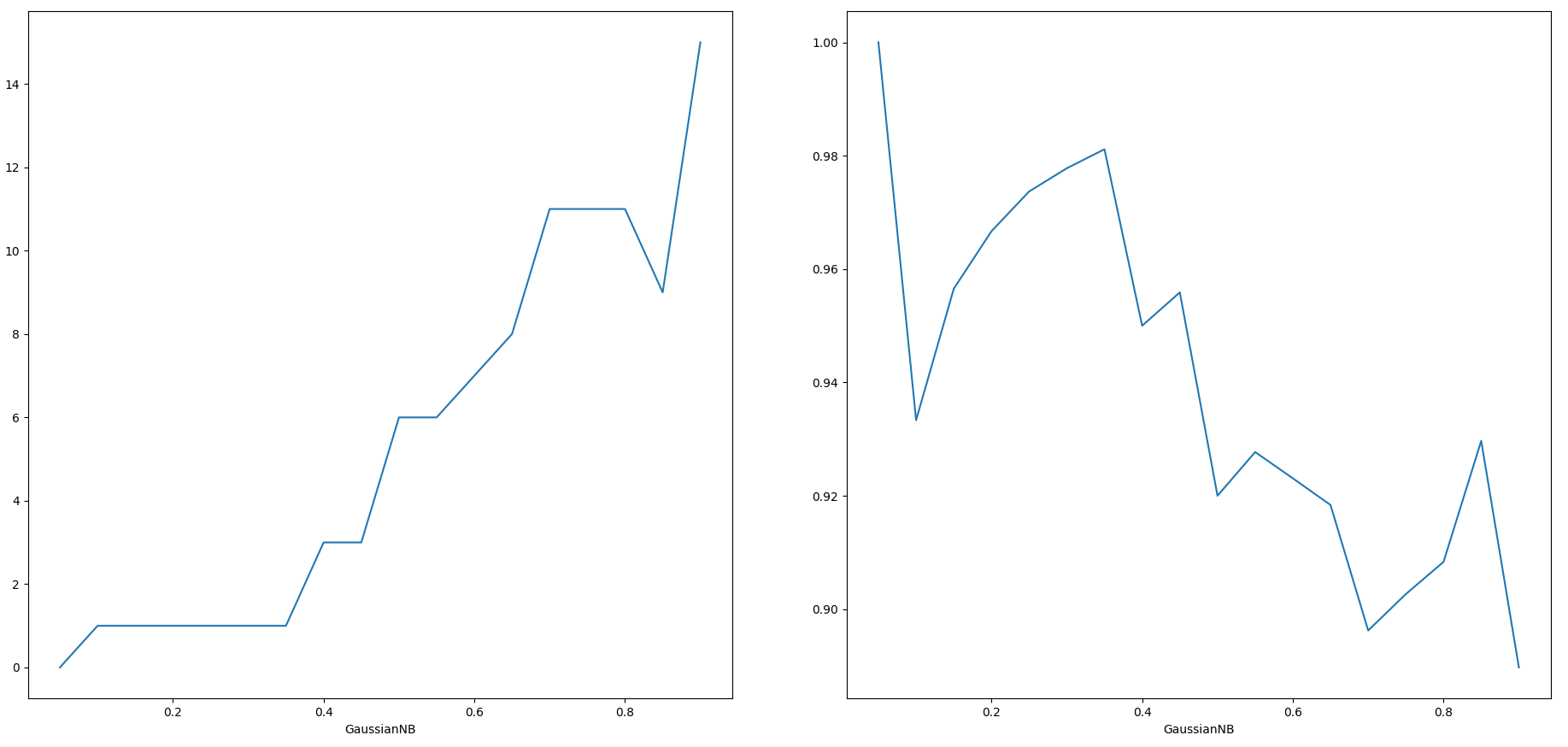
theta\_ - среднее значение каждого объекта каждого класса.

*class\_count\_ : ndarray of shape (n\_classes,)  
 number of training samples observed in each class.  
  
class\_prior\_ : ndarray of shape (n\_classes,)  
 probability of each class.  
  
classes\_ : ndarray of shape (n\_classes,)  
 class labels known to the classifier.  
  
epsilon\_ : float  
 absolute additive value to variances.  
  
n\_features\_in\_ : int  
 Number of features seen during :term:`fit`.  
  
 .. versionadded:: 0.24  
  
feature\_names\_in\_ : ndarray of shape (`n\_features\_in\_`,)  
 Names of features seen during :term:`fit`. Defined only when `X`  
 has feature names that are all strings.  
  
 .. versionadded:: 1.0  
  
sigma\_ : ndarray of shape (n\_classes, n\_features)  
 Variance of each feature per class.  
  
 .. deprecated:: 1.0  
 `sigma\_` is deprecated in 1.0 and will be removed in 1.2.  
 Use `var\_` instead.  
  
var\_ : ndarray of shape (n\_classes, n\_features)  
 Variance of each feature per class.  
  
 .. versionadded:: 1.0  
  
theta\_ : ndarray of shape (n\_classes, n\_features)  
 mean of each feature per class.*

* 1. Используя функцию score() выведена точность классификации

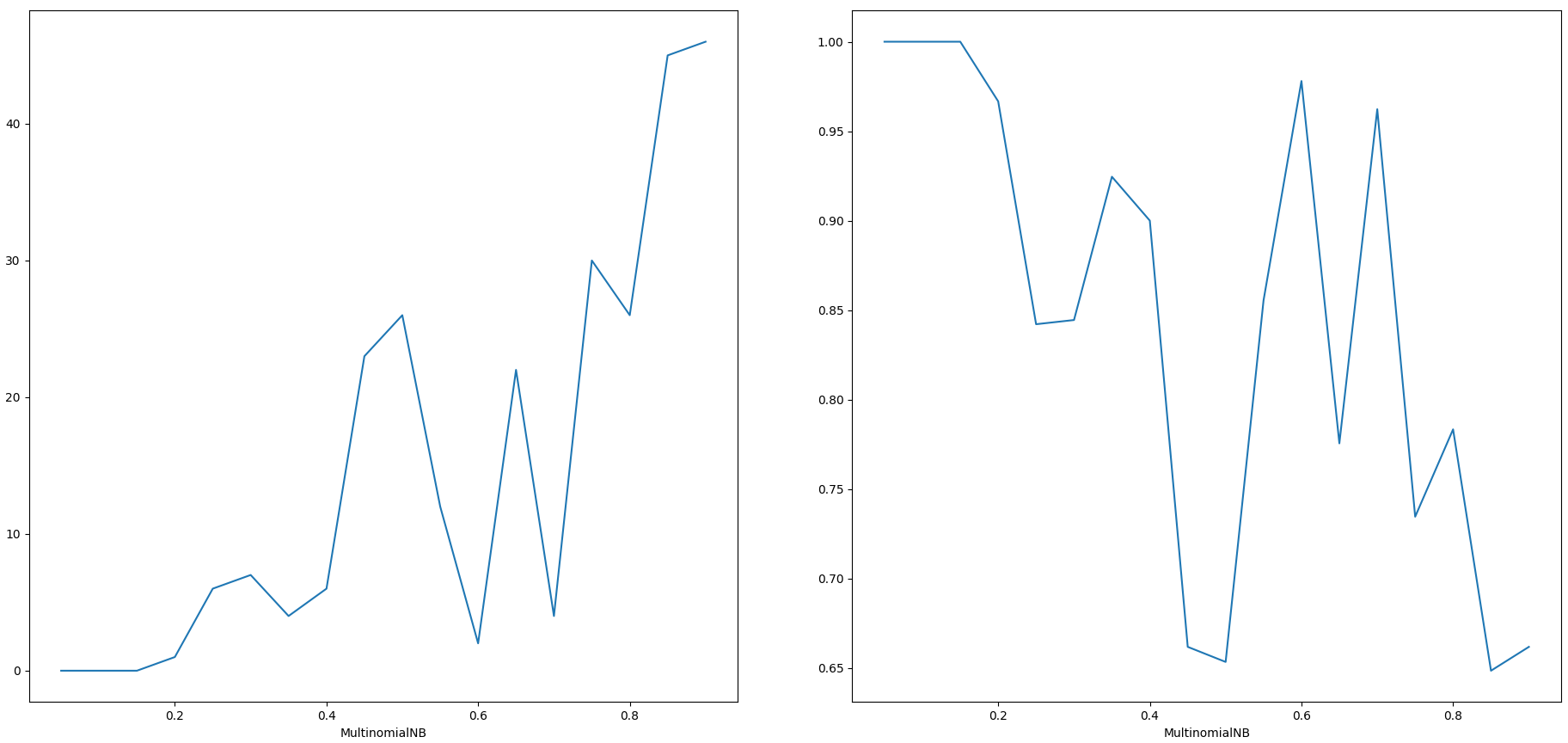
Score = 0.9466666666666667

* 1. Построены графики зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки.

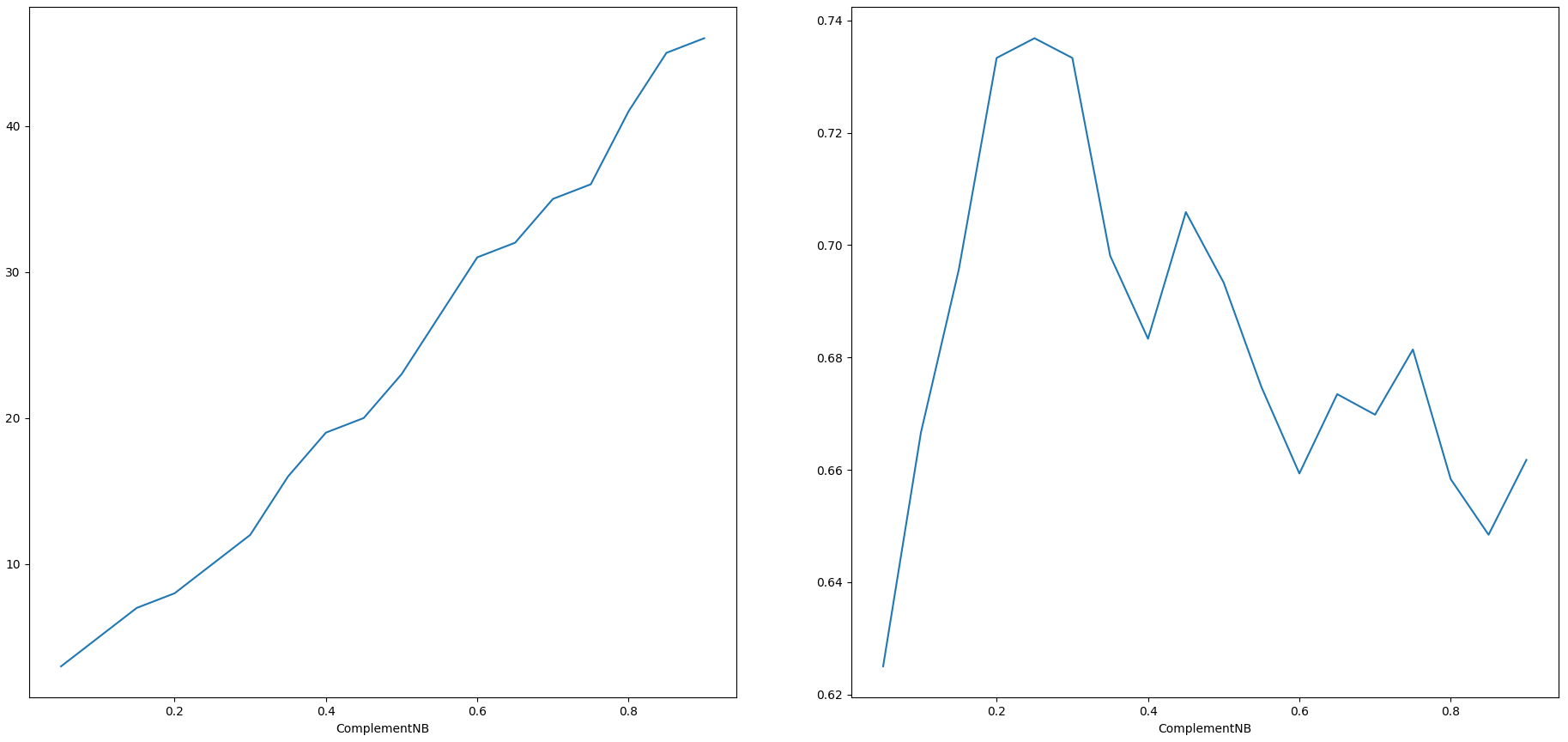


GaussianNB

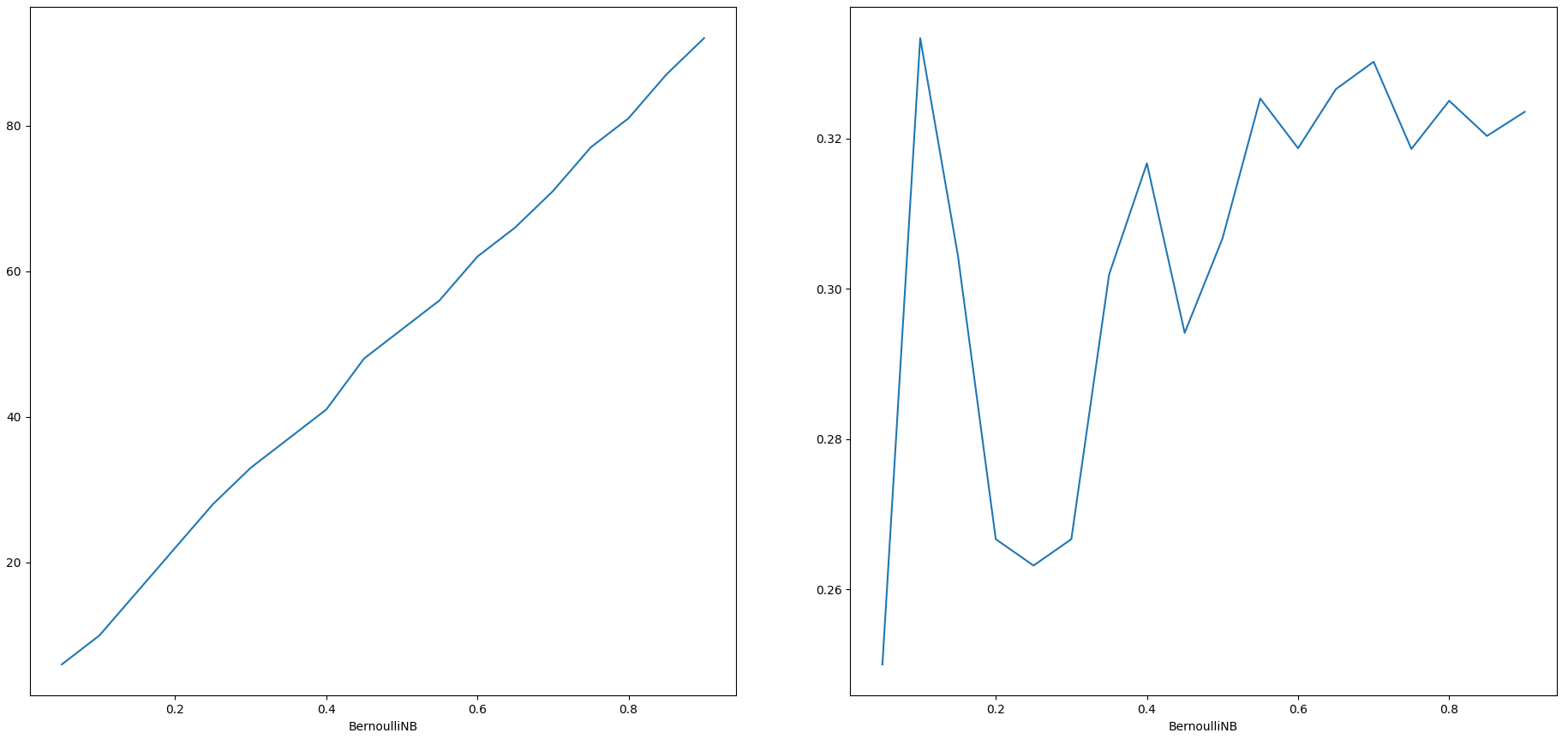
* 1. Проведена классификация, используя MultinomialNB, ComplementNB, BernoulliNB.



MultinomialNB



ComplementNB



BernoulliNB

В методе MultinominalNB распределение для каждого класса параметризуется векторами, содержащими вероятности вхождения признаков в элемент выборки, соответствующей данному классу.

Метод ComplementNB – это адаптация стандартного полиномиального наивного байесовского алгоритма MultinominalNB, который особенно подходит для несбалансированных наборов данных.

BernouliNB реализует наивные байесовские алгоритмы для данных, которые распределяются согласно многомерному распределению Бернулли.

Предполагается, что каждый признак является двоичной/логической переменной.

1. **Классифицирующие деревья**
   1. Проведена классификация при помощи деревьев на тех же данных методом. Количество найденных неправильно классифицированных наблюдений = 3.
   2. Используя функцию score() выведена точность классификации

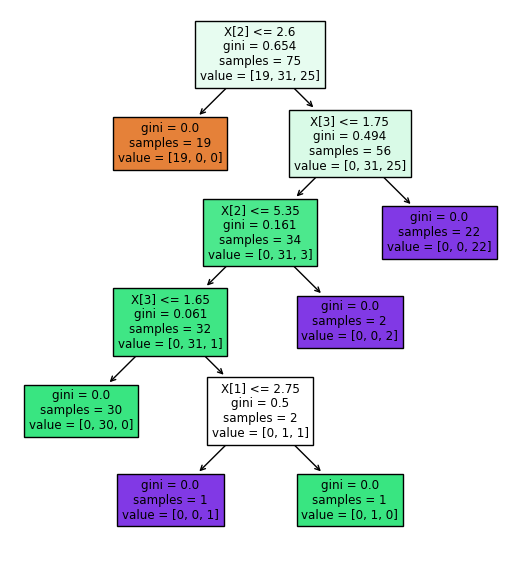
Score = 0.96

* 1. Выведены характеристики дерева, количество листьев и глубину, используя функции get\_n\_leaves и get\_depth. Полученные результаты:

get\_n\_leaves = 6

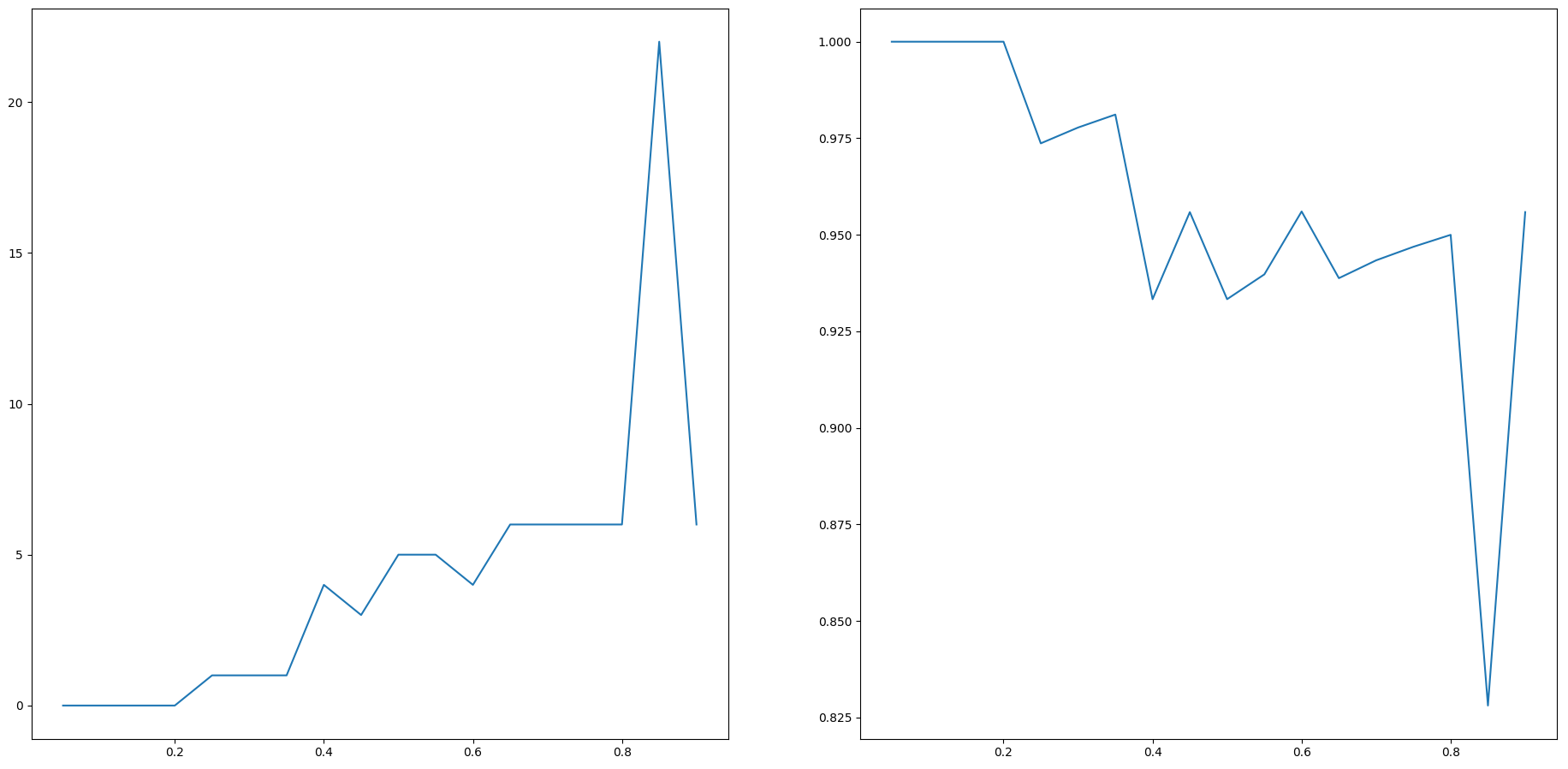
get\_depth = 5

* 1. Было выведено изображение полученного дерева



Для каждого узла указывается условие для разбиения, значение примеси Джини и кол-во наблюдений в узле. Каждый класс определён своим цветом.

* 1. Построены графики зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки.



* 1. Была исследована работа классифицирующего дерева при различных параметрах criterion, splitter, max\_depth, min\_samples\_split, min\_samples\_leaf

Criterion – отвечает за функцию измерения качества разбиения. Поддерживаемые критерии: «Джини» для примеси Джини и «энтропия». Для обоих значений получились идентичные результаты классификации.

Splitter - cтратегия, используемая для выбора разделения на каждом узле. Поддерживаемые стратегии являются «лучшими» для выбора наилучшего разбиения и «случайными» для выбора наилучшего случайного разбиения.

Max\_depth - Максимальная глубина дерева. Если None, то узлы расширяются до тех пор, пока все листья не станут чистыми или пока все листья не будут содержать менее min\_samples\_split выборок.

Min\_samples\_split - Минимальное количество выборок, необходимых для разделения внутреннего узла

Min\_samples\_leaf - Минимальное количество выборок, необходимых для работы на листовом узле

1. **Выводы**

Ознакомились с методами классификации модуля Sklearn.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

import pandas as pd  
import numpy as np  
from sklearn import preprocessing  
from sklearn.model\_selection import train\_test\_split  
from sklearn.naive\_bayes import GaussianNB, MultinomialNB, ComplementNB, BernoulliNB  
import matplotlib.pyplot as plt  
from sklearn import tree  
  
  
data = pd.read\_csv('iris.data',header=None)  
print(data)  
  
X = data.iloc[:,:4].to\_numpy()  
labels = data.iloc[:,4].to\_numpy()  
le = preprocessing.LabelEncoder()  
Y = le.fit\_transform(labels)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.5)  
  
gnb = GaussianNB()  
y\_pred = gnb.fit(X\_train, y\_train).predict(X\_test)  
print((y\_test != y\_pred).sum())  
print(gnb.score(X\_test, y\_test))  
  
def grafics(clf, title=""):  
 test\_sizes = np.arange(0.05, 0.95, 0.05)  
 wrong\_results = []  
 accuracies = []  
  
 for test\_size in test\_sizes:  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=test\_size, random\_state=830406)  
 y\_pred = clf.fit(X\_train, y\_train).predict(X\_test)  
 wrong\_results.append((y\_test != y\_pred).sum())  
 accuracies.append(clf.score(X\_test, y\_test))  
  
  
 fig, axs = plt.subplots(1, 2, figsize=(8, 4))  
 axs[0].plot(test\_sizes, wrong\_results)  
 axs[1].plot(test\_sizes, accuracies)  
 axs[0].set\_xlabel(title)  
 axs[1].set\_xlabel(title)  
 plt.tight\_layout()  
 plt.show()  
  
  
  
gnb = GaussianNB()  
mnb = MultinomialNB()  
cnb = ComplementNB()  
bnb = BernoulliNB()  
grafics(gnb, 'GaussianNB')  
grafics(mnb, 'MultinomialNB')  
grafics(cnb, 'ComplementNB')  
grafics(bnb, 'BernoulliNB')  
  
clf = tree.DecisionTreeClassifier()  
y\_pred = clf.fit(X\_train, y\_train).predict(X\_test)  
print((y\_test != y\_pred).sum())  
print(clf.score(X\_test, y\_test))  
print(clf.get\_n\_leaves())  
print(clf.get\_depth())  
plt.subplots(1,1,figsize = (5,5))  
tree.plot\_tree(clf, filled = True)  
plt.show()  
grafics(clf)