

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

Кириянов Д. И.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

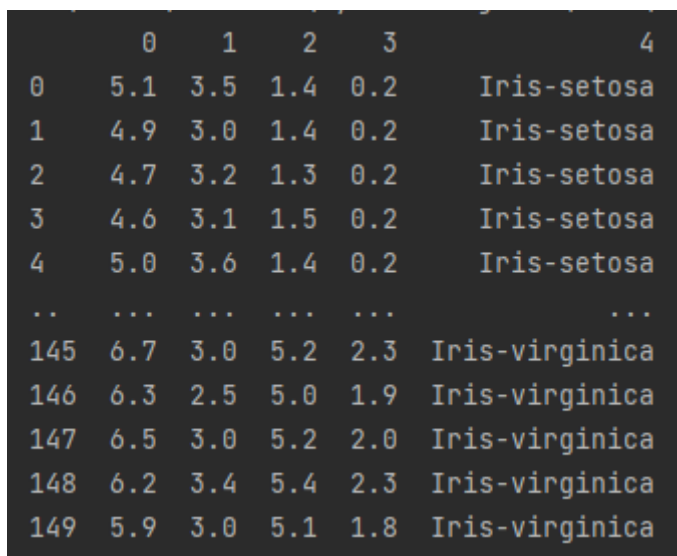
Цель работы.

Ознакомиться с методами классификации модуля Sklearn.

Ход работы.

1. Загрузка данных

1.1. Произведена загрузка данных. Создан Python скрипт. Данные загружены в датафрейм.



```
0      5.1  3.5  1.4  0.2  Iris-setosa
1      4.9  3.0  1.4  0.2  Iris-setosa
2      4.7  3.2  1.3  0.2  Iris-setosa
3      4.6  3.1  1.5  0.2  Iris-setosa
4      5.0  3.6  1.4  0.2  Iris-setosa
..      ...   ...   ...   ...   ...
145     6.7  3.0  5.2  2.3  Iris-virginica
146     6.3  2.5  5.0  1.9  Iris-virginica
147     6.5  3.0  5.2  2.0  Iris-virginica
148     6.2  3.4  5.4  2.3  Iris-virginica
149     5.9  3.0  5.1  1.8  Iris-virginica
```

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

2. Байесовские методы

2.1. 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

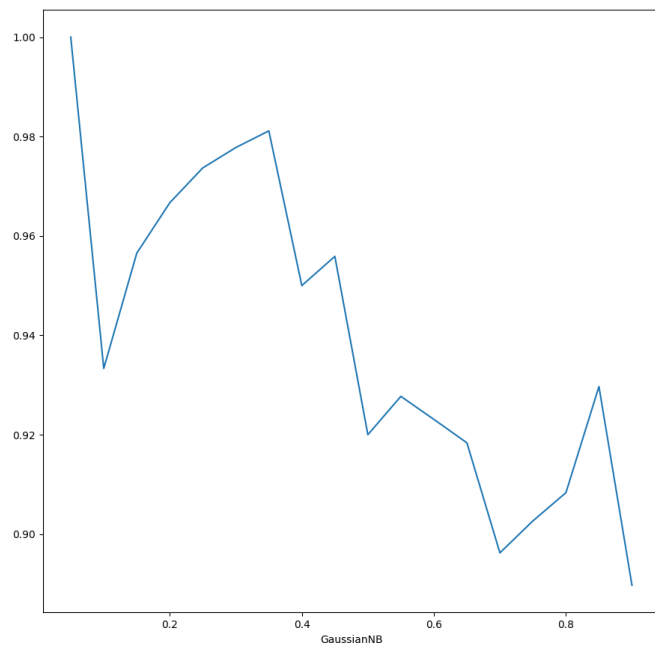
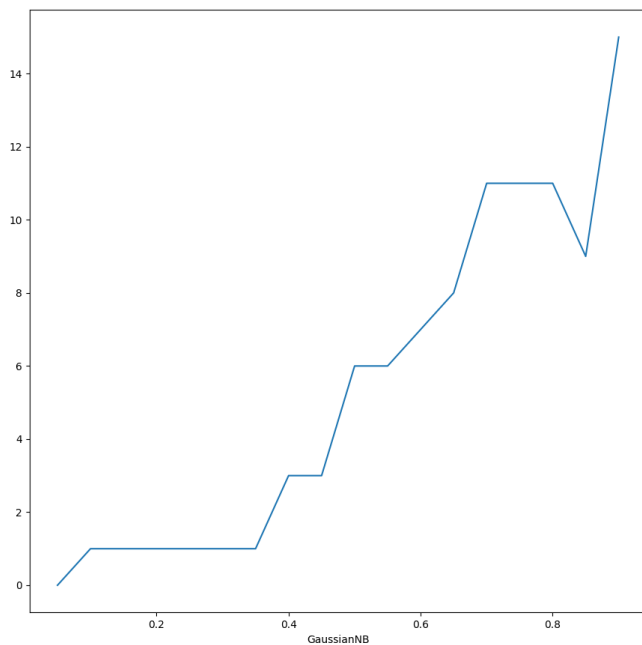
theta_ : ndarray of shape (n_classes, n_features)
    mean of each feature per class.

```

2.2. Используя функцию score() выведена точность классификации

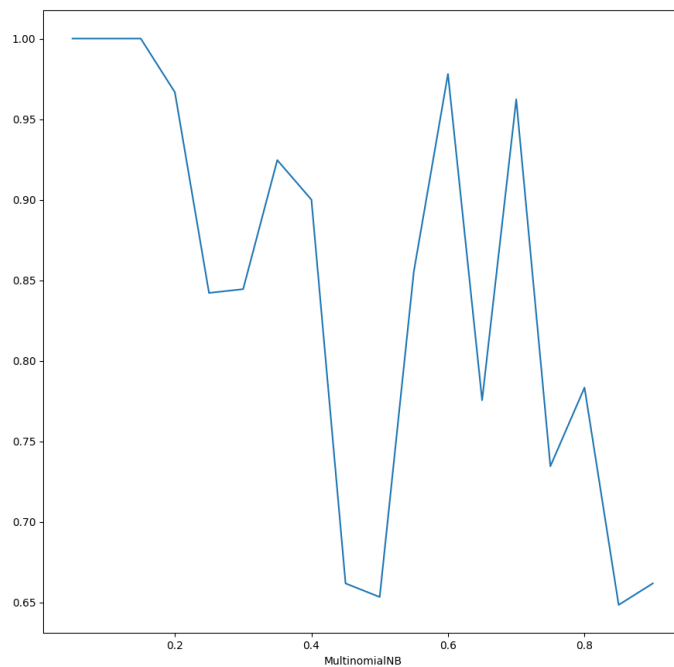
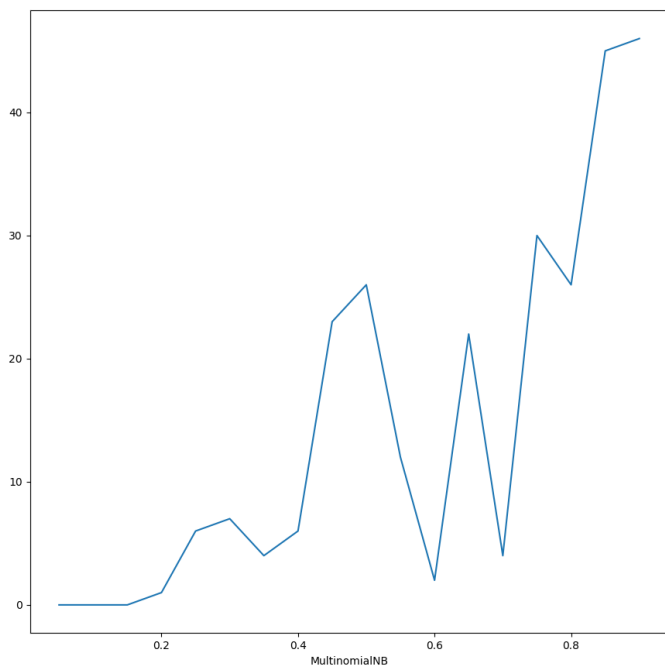
Score = 0.9466666666666667

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

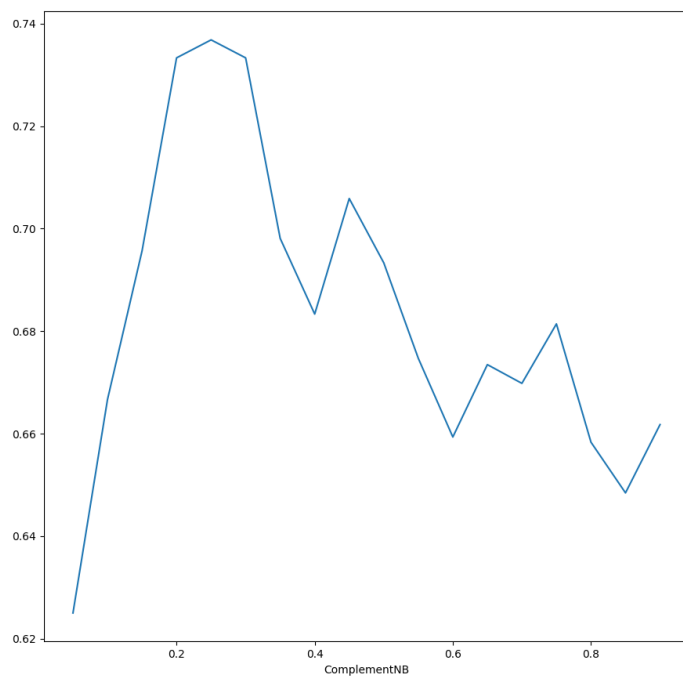
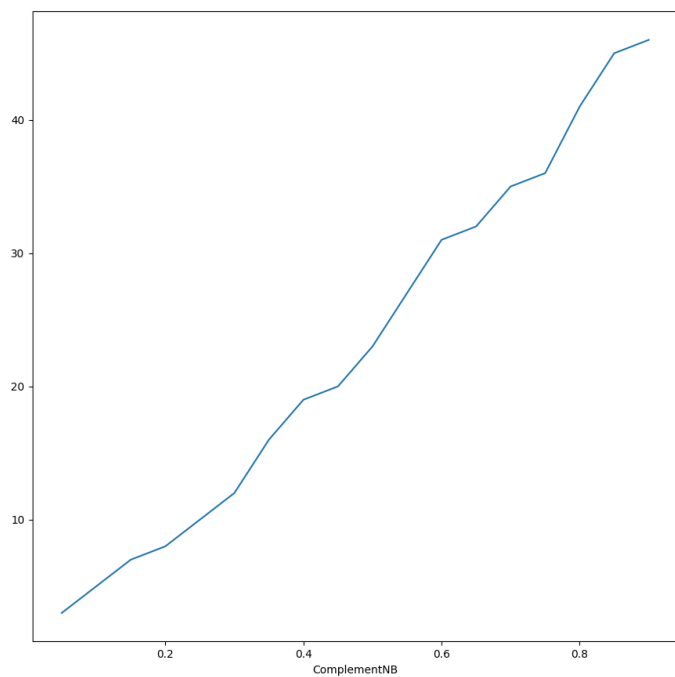


GaussianNB

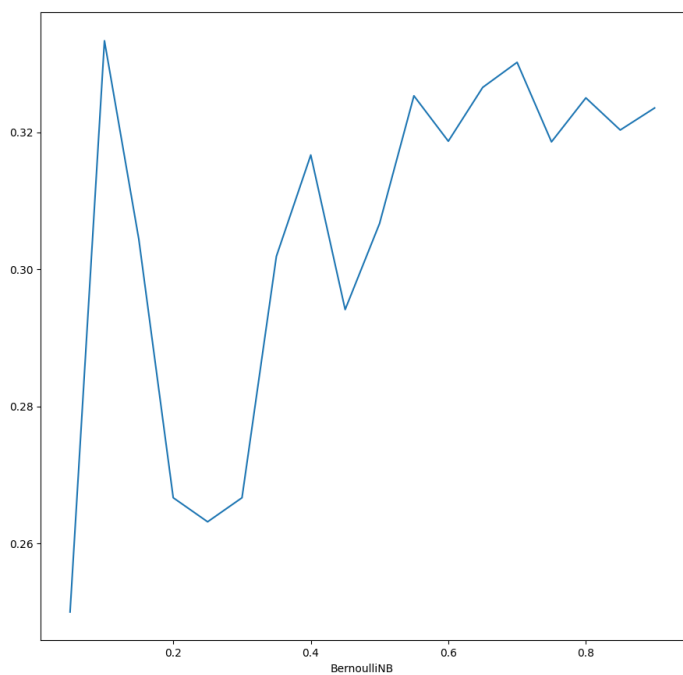
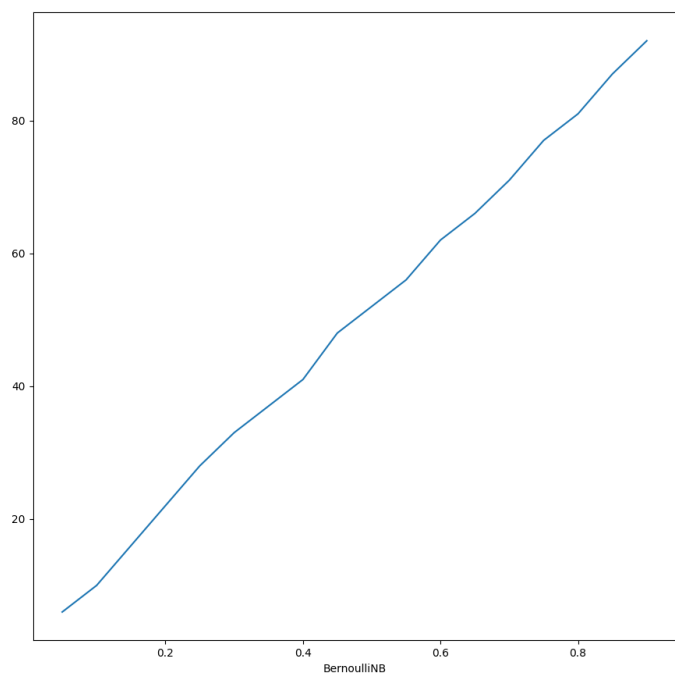
2.4. Проведена классификация, используя MultinomialNB, ComplementNB, BernoulliNB.



MultinomialNB



ComplementNB



BernoulliNB

В методе MultinomialNB 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 MultinomialNB, 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.

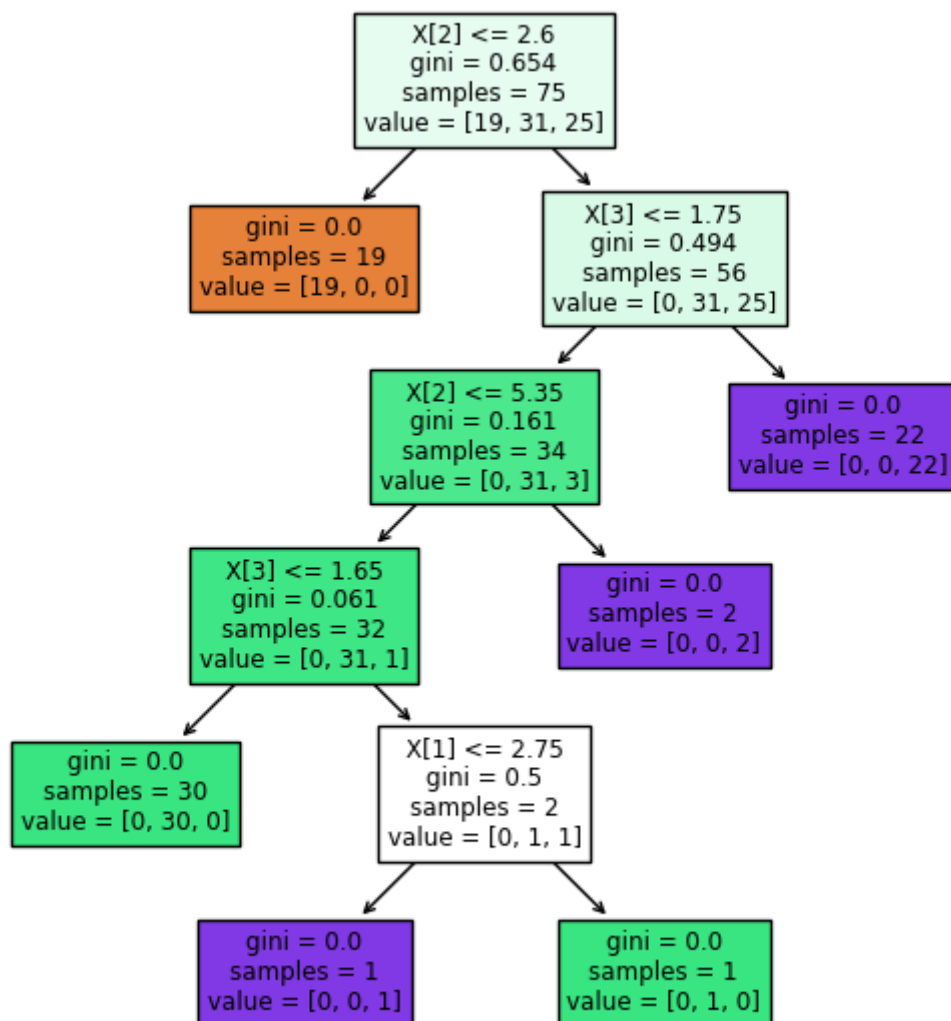
3. Классифицирующие деревья

3.1. Проведена классификация при помощи деревьев на тех же данных методом. Количество найденных неправильно классифицированных наблюдений = 3.

3.2. Используя функцию score() выведена точность классификации
Score = 0.96

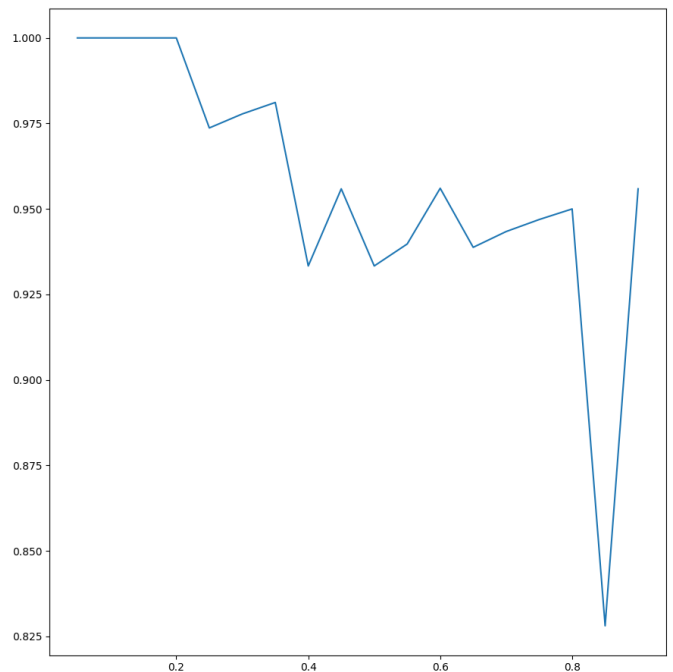
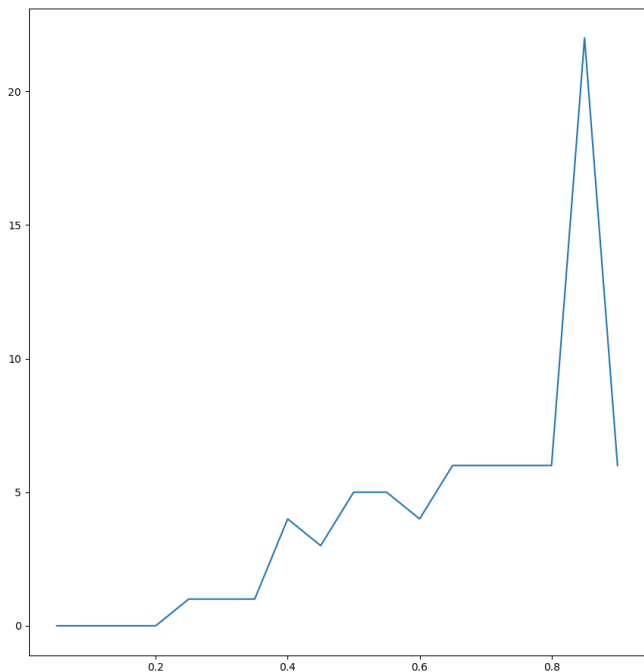
3.3. Выведены характеристики дерева, количество листьев и глубину, используя функции get_n_leaves и get_depth. Полученные результаты:
get_n_leaves = 6
get_depth = 5

3.4. 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.

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



3.6. 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` - стратегия, 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

4. 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)
```