МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе № 7
по дисциплине «Машинное обучение»
Тема: Классификация (Байесовские методы, деревья)

Студенты гр. 6304 _____ Григорьев И.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы

Ознакомиться с методами классификации модуля *Sklearn*.

Ход работы

Загрузка данных

Датасет загружен в датафрейм. Вид данных представлен на рис. 1.

	0	1	2	3	4
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

150 rows × 5 columns

Рисунок 1 – Исходные данные

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

Байесовские методы

1. Проведена классификация наблюдений наивным байесовским методом. Выявлено 4 неправильно классифицированных наблюдения. Атрибуты классификатора представлены в табл. 1.

Таблица 1 – Атрибуты *GaussianNB*

Атрибут	Описание
class_count_	Количество обучающих выборок, наблюдаемых в каждом классе
class_prior_	Вероятность каждого класса
classes_	Метки классов, известные классификатору
epsilon_	Абсолютная аддитивная величина дисперсий
sigma_	Дисперсия каждого признака по классу
theta_	Среднее каждого признака по классу

2. Точность классификации получена с помощью функции score() и составляет 97%.
3. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки. График представлен на рис. 2.

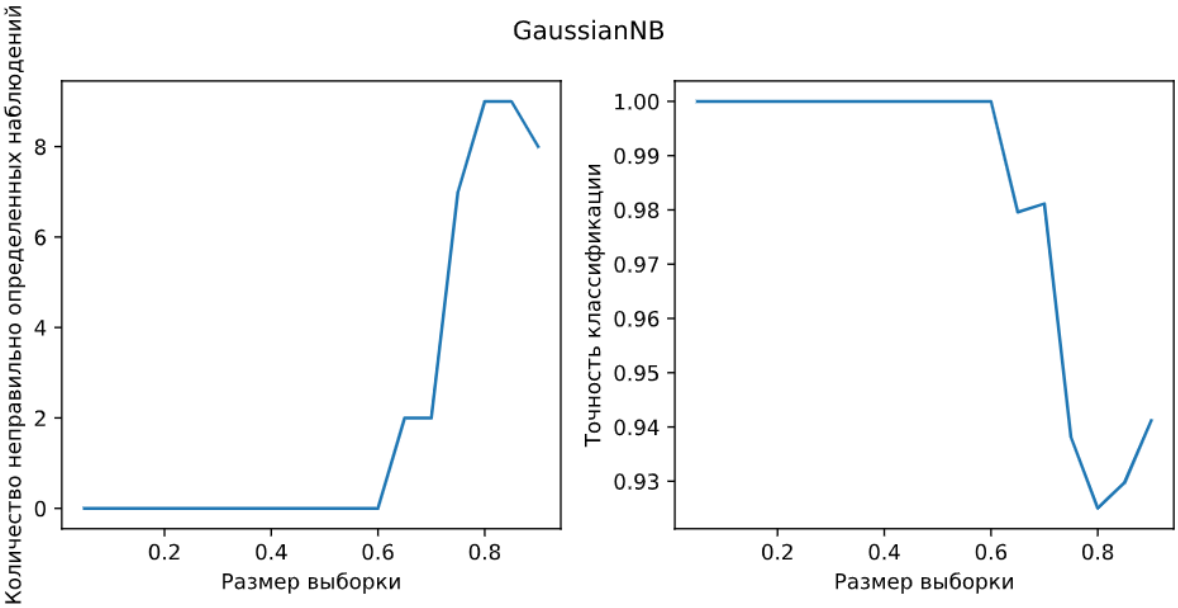

Рисунок 2 – Классификация *GaussianNB*

4. Классификация проведена с помощью *MultinominalNB, ComplementNB, BernoulluNB*. Результат представлен на рис. 3-5.

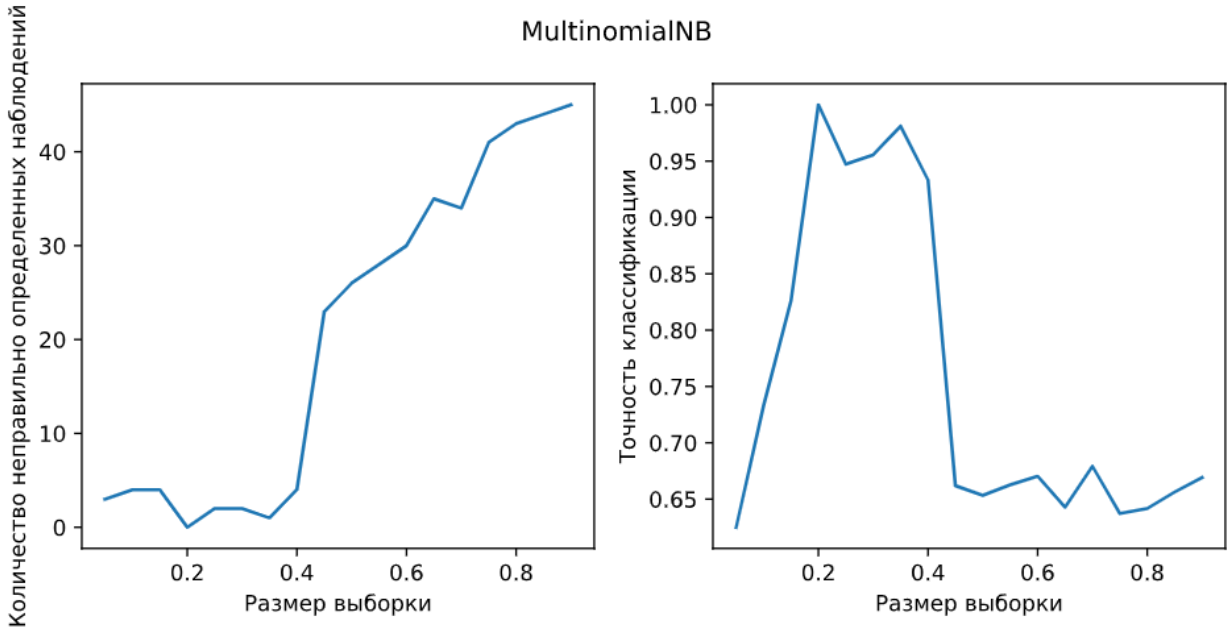

Рисунок 3 – Классификация *MultinominalNB*

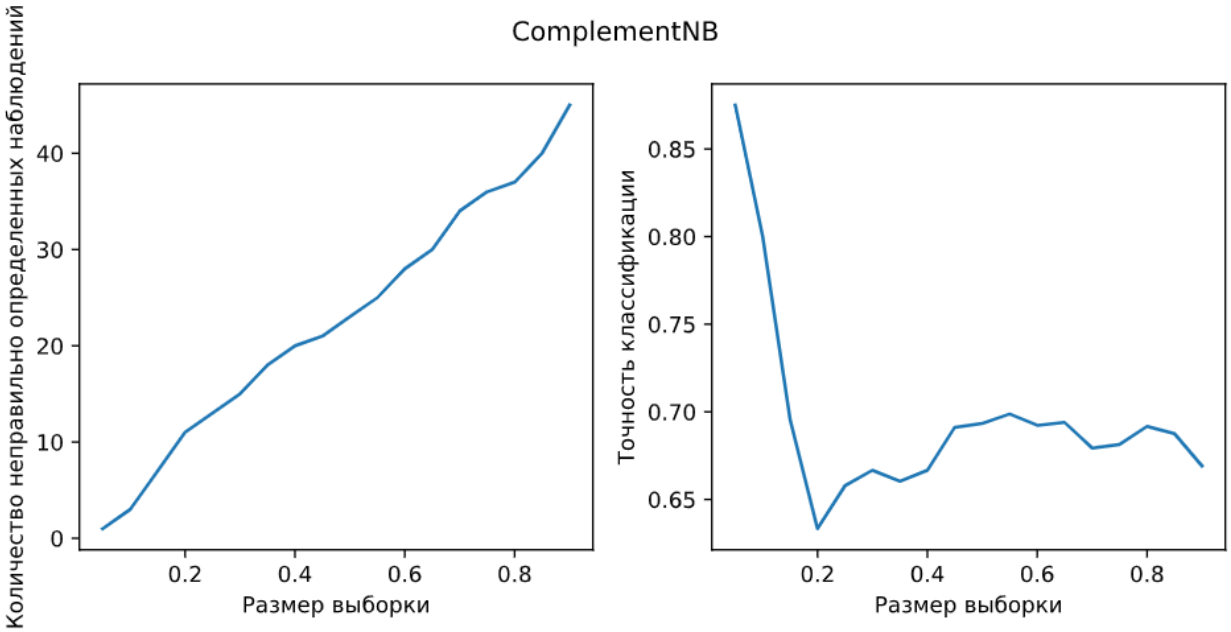

Рисунок 4 – Классификация *ComplementNB*

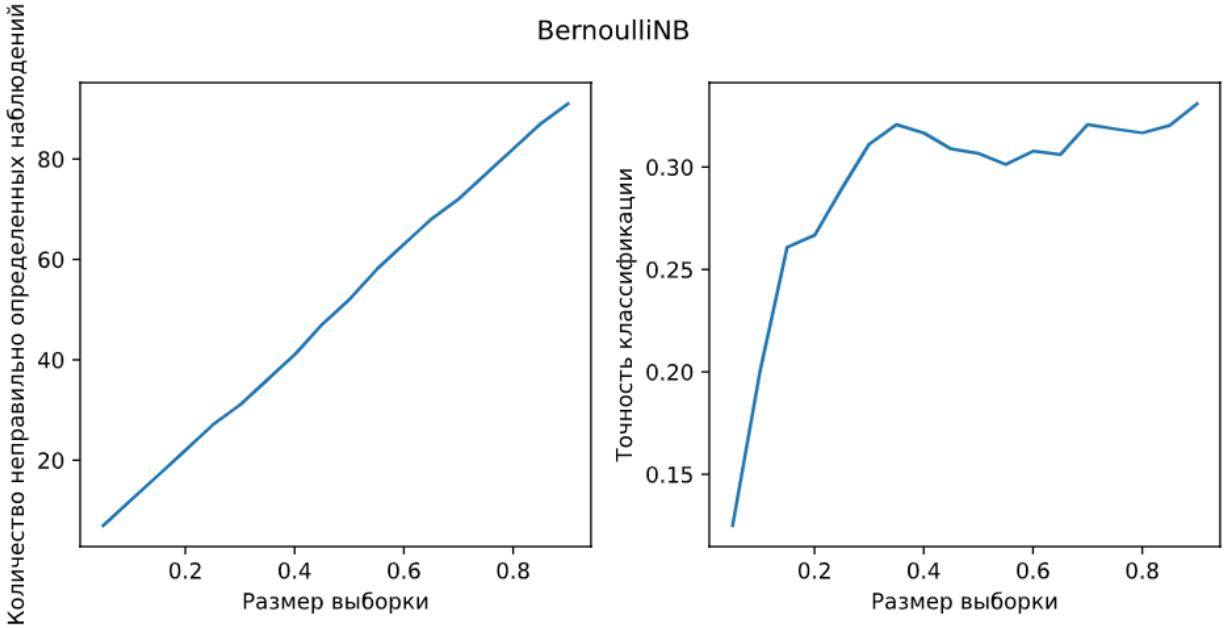

Рисунок 5 – Классификация *BernoulluNB*

MultinominalNB – полиномиальный наивный байесовский классификатор, подходит для классификации с дискретными признаками (например, подсчет слов для классификации текста). *MultinominalNB* реализует наивный алгоритм Байеса для полиномиально распределенных данных. Распределение для каждого класса параметризируется векторами, содержащими вероятности вхождения признаков в элемент выборки, соответствующий данному классу.

ComplementNB – адаптация *MultinominalNB*, подходит для несбалансированных наборов данных. В частности, CNB использует статистику из дополнения каждого класса для вычисления весов модели. *ComplementNB* часто превосходит *MultinominalNB* в задачах классификации текста.

BernoulluNB – как и *MultinominalNB*, этот классификатор подходит для дискретных данных. Разница в том, что в то время, как *MultinominalNB* работает с подсчетом вхождений, *BernoulluNB* предназначен для двоичных/логических признаков.

Классифицирующие деревья

1. Проведена классификация наблюдений с помощью деревьев решений на тех же данных. Выявлено 4 неправильно классифицированных наблюдения.
2. Точность классификации получена с помощью функции score() и составляет 100%.
3. Получившееся дерево имеет глубину, равную 3, и 4 листа.
4. Дерево продемонстрировано на рис. 6.

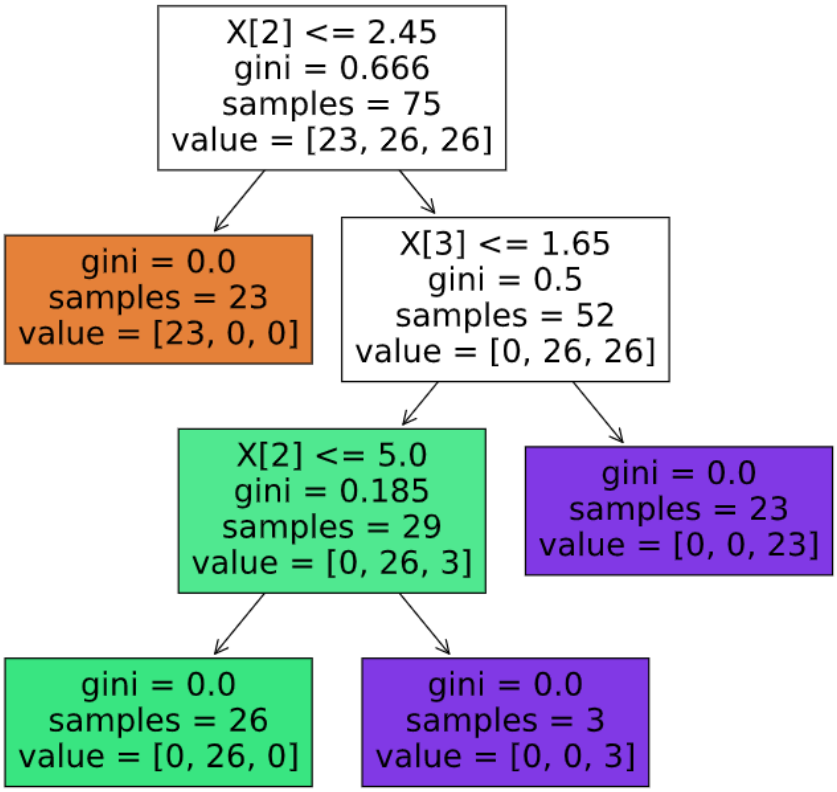

Рисунок 6 – Дерево решений для классификации

5. Построен график зависимости неправильно классифицированных наблюдений и точности классификации от размера тестовой выборки. График представлен на рис. 7.

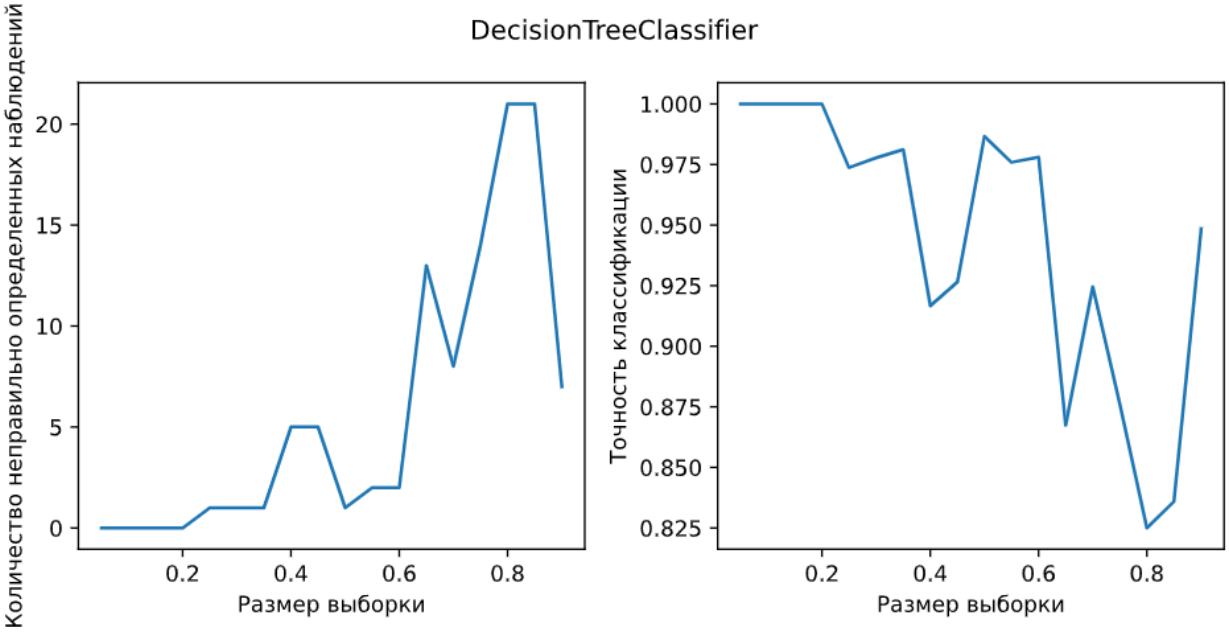

Рисунок 7 – Классификация *DecisionTreeClassifier*

6. Исследованы параметры *DecisionTreeClassifier*, результаты представлены в табл. 2. и на рис. 8.

Таблица 2 – Парметры *DecisionTreeClassifier*

Параметр	Описание
criterion	Функция измерения качества разбиения. Поддерживается индекс Джини и энтропия.
splitter	Стратегия, используемая для выбора разбиения на каждом узле. Поддерживается выбор наилучшего разбиения и случайный выбор.
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, которое требуется для конечного узла. Точка разделения на любой глубине будет учитываться только в том случае, если она оставляет не менее min_samples_leaf обучающих выборок в каждой из левой и правой ветвей.

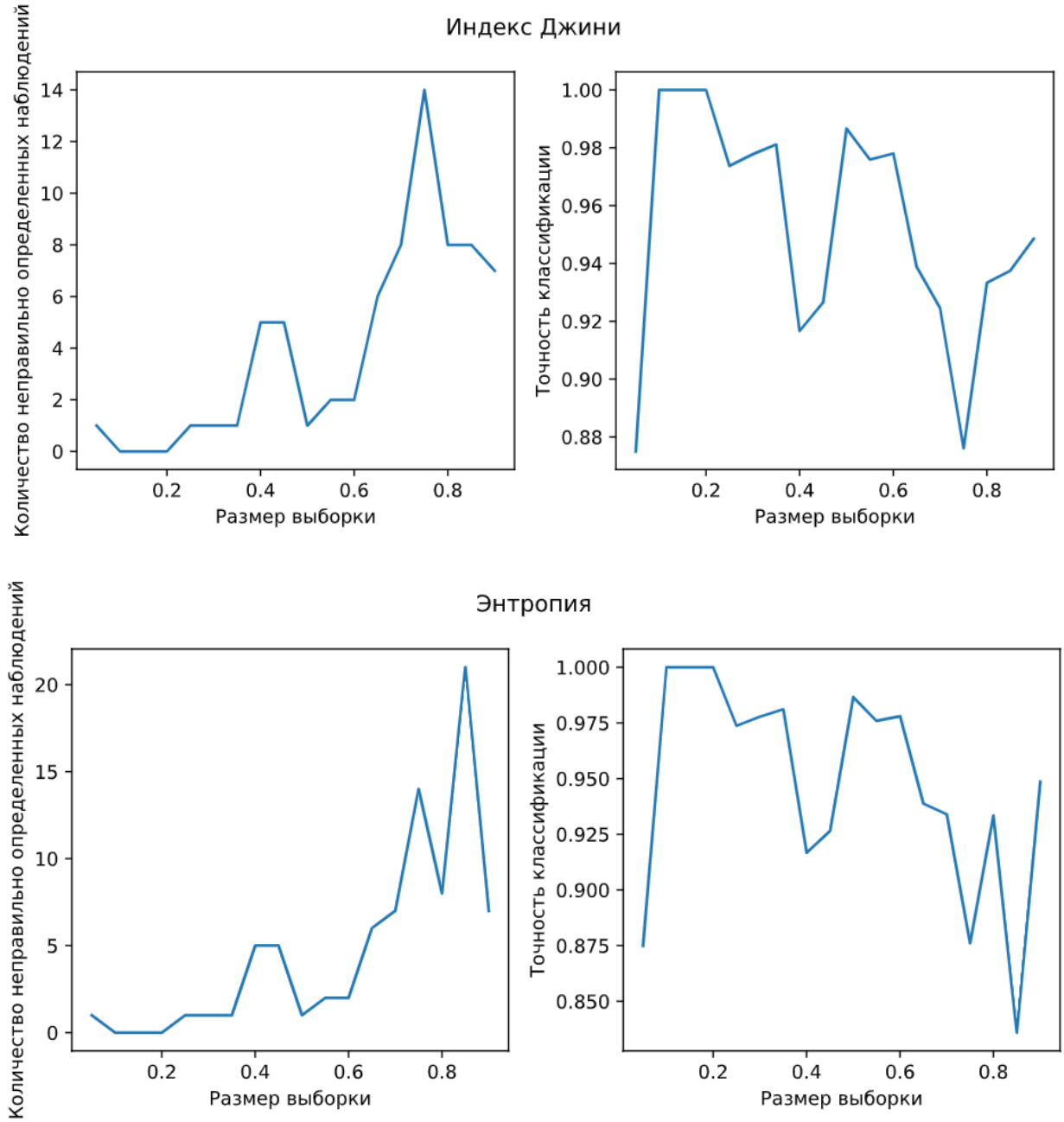

Рисунок 8 – Разные функции измерения качества разбиения

Выводы

В ходе лабораторной работы рассмотрены такие методы классификации модуля *Sklearn*, как *GaussianNB, MultinominalNB, ComplementNB, BernoulluNB* и *DecisionTreeClassifier*.

Приложение А

Код программы на python

```
# To add a new cell, type '# %%'
# To add a new markdown cell, type '# %% [markdown]'
# %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB, ComplementNB, BernoulliNB
from sklearn import tree

# %%
data = pd.read_csv('iris.data',header=None)
data

# %%
X = data.iloc[:,:4].to_numpy()
labels = data.iloc[:,4].to_numpy()

# %%
le = preprocessing.LabelEncoder()
Y = le.fit_transform(labels)

# %%
Y

# %%
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()) # количество наблюдений, который были неправильно опр
еделены

# %%
gnb.score(X_train, y_train)

# %%
def plot_clf(clf, title=""):
    test_sizes = np.arange(0.05, 0.95, 0.05)
    wrong_results = []
    scores = []

    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=630405)
        y_pred = clf.fit(X_train, y_train).predict(X_test)
        wrong_results.append((y_test != y_pred).sum())
        scores.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, scores)
    axs[0].set_ylabel('Количество неправильно определенных наблюдений')
    axs[1].set_ylabel('Точность классификации')
    axs[0].set_xlabel('Размер выборки')
    axs[1].set_xlabel('Размер выборки')
```

```
    fig.suptitle(title)
    plt.tight_layout()
    plt.show()

# %%
plot_clf(GaussianNB(), 'GaussianNB')
plot_clf(MultinomialNB(), 'MultinomialNB')
plot_clf(ComplementNB(), 'ComplementNB')
plot_clf(BernoulliNB(), 'BernoulliNB')

# %%
clf = tree.DecisionTreeClassifier()
y_pred = clf.fit(X_train, y_train).predict(X_test)
print((y_test != y_pred).sum())

# %%
clf.score(X_train, y_train)

# %%
print('leaves: ', clf.get_n_leaves())
print('depth: ', clf.get_depth())

# %%
plt.subplots(1,1,figsize = (10,10))
tree.plot_tree(clf, filled = True)
plt.show()

# %%
plot_clf(tree.DecisionTreeClassifier(), 'DecisionTreeClassifier')

# %%
plot_clf(tree.DecisionTreeClassifier(criterion='entropy'), 'DecisionTreeClassifier')

# %%
test_sizes = np.arange(0.05, 0.95, 0.05)
wrong_results_1 = []
scores_1 = []
wrong_results_2 = []
scores_2 = []

for test_size in test_sizes:
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=test_size, ra
ndom_state=630405)

    clf_1 = tree.DecisionTreeClassifier()
    y_pred = clf_1.fit(X_train, y_train).predict(X_test)
    wrong_results_1.append((y_test != y_pred).sum())
    scores_1.append(clf_1.score(X_test, y_test))

    clf_2 = tree.DecisionTreeClassifier(criterion='entropy')
    y_pred = clf_2.fit(X_train, y_train).predict(X_test)
    wrong_results_2.append((y_test != y_pred).sum())
    scores_2.append(clf_2.score(X_test, y_test))

fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(test_sizes, wrong_results_1)
axs[1].plot(test_sizes, scores_1)
axs[0].set_ylabel('Количество неправильно определенных наблюдений')
axs[1].set_ylabel('Точность классификации')
axs[0].set_xlabel('Размер выборки')
axs[1].set_xlabel('Размер выборки')
fig.suptitle('Индекс Джини')
```

```
plt.tight_layout()
plt.show()

fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(test_sizes, wrong_results_2)
axs[1].plot(test_sizes, scores_2)
axs[0].set_ylabel('Количество неправильно определенных наблюдений')
axs[1].set_ylabel('Точность классификации')
axs[0].set_xlabel('Размер выборки')
axs[1].set_xlabel('Размер выборки')
fig.suptitle('Энтропия')
plt.tight_layout()
plt.show()

# %%
```