

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

Кафедра математического обеспечения и применения ЭВМ

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

Студент гр. 1310

Комаров Д. Е.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2025

Постановка задачи

Цель работы: ознакомление с методами классификации модуля Sklearn.

Выполнение лабораторной работы

Загрузка данных

Для выполнения лабораторной работы используем набор данных «Iris».

Данный набор данных содержит 150 записей о трех классах цветов. Загрузим данные в датафрейм. Фрагмент результата представлен на рисунке 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

Рисунок 1 – Данные для классификации

Выделим из данных метки классов, преобразуем тексты меток к числам.

Разобьём выборку на обучающую и тестовую. Получилось 75 записей в обучающей выборке и 75 записей в тестовой выборке.

Байесовские методы

Байесовский классификатор использует теорему Байеса для расчета вероятности попадания наблюдения в какой-либо класс и относит наблюдение к тому классу, вероятность которого больше. Наивный байесовский классификатор является модификацией байесовского классификатора, которая считает, что все признаки являются независимыми случайными величинами.

Проведем классификацию наблюдений наивным байесовским методом при помощи *GaussianNB*. Данный классификатор предполагает, что все признаки данных имеют нормальное распределение. Правильно было классифицировано 71 наблюдение, неправильно 4. Данный классификатор имеет такие атрибуты, как *class_count_*, *class_prior_*, *classes_*, *epsilon_*,

n_features_in_, features_name_in_, var_, theta_. Атрибут *classes_* хранит метки всех классов, известных классификатору. Атрибут *class_count_* показывает, сколько наблюдений содержится в каждом классе. Атрибут *class_prior_* хранит априорные вероятности классов. Атрибут *n_features_in_* хранит размерность пространства признаков классифицируемых данных. Атрибут *features_name_in_* хранит имена признаков. Атрибут *theta_* хранит среднее значение каждого признака каждого класса. Атрибут *var_* хранит дисперсию каждого признака каждого класса. Атрибут *epsilon_* хранит абсолютное добавочное значение к дисперсии.

Воспользуемся методом *score* чтобы вычислить точность классификатора. Точность классификации равна 95%.

Построим графики зависимости количества неправильно классифицированных наблюдений и точности классификации от соотношения обучающей и тестовой выборок. График зависимости количества неправильно классифицированных наблюдений от соотношения обучающей и тестовой выборок представлен на рисунке 2.

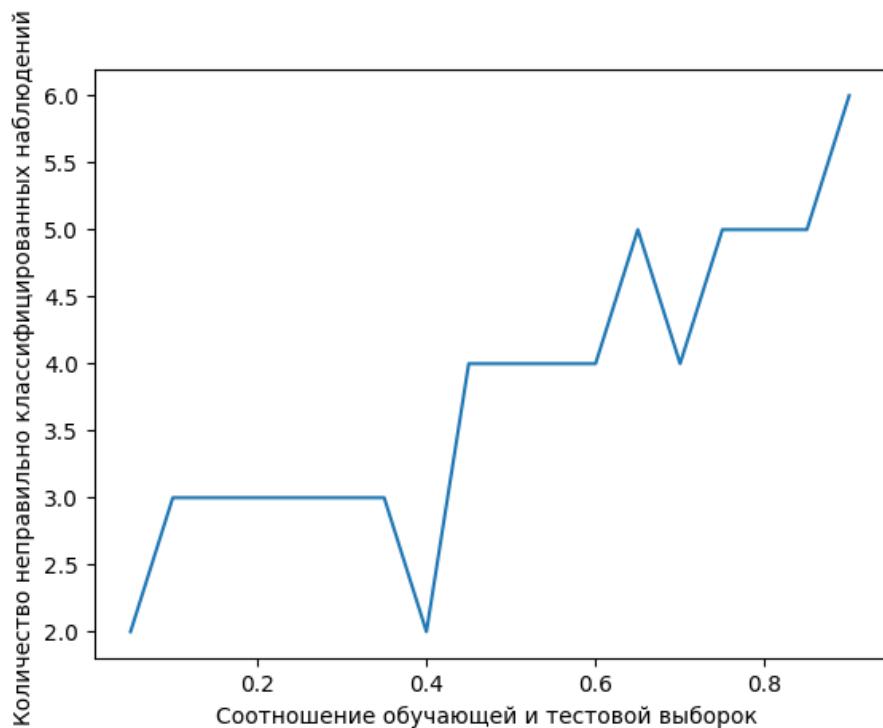


Рисунок 2 – График зависимости количества неправильно классифицированных наблюдений от соотношения обучающей и тестовой выборок для *GaussianNB*

Как можно увидеть из рисунка 2, при увеличении размера тестовой выборки количество неправильно классифицированных наблюдений также растет, что логично, так как в большей выборке находится больше наблюдений, которые трудно классифицировать. На рисунке 3 представлен график зависимости точности классификации от соотношения обучающей и тестовой выборок.

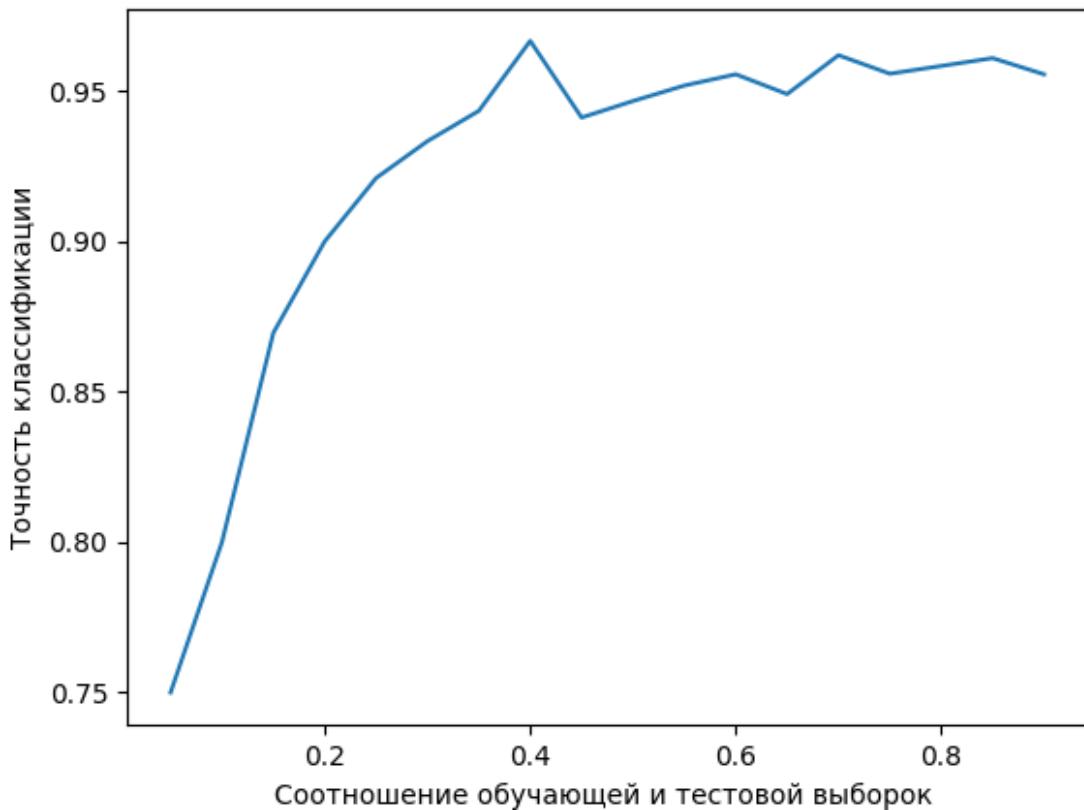


Рисунок 3 – График зависимости точности классификации от соотношения обучающей и тестовой выборок для *GaussianNB*

Из рисунка 3 можно заметить, что при увеличении соотношения обучающей и тестовой выборки точность также растет. Объясняется это тем, что для данного набора данных достаточно небольшой обучающей выборки для обучения классификатора, а при слишком большой обучающей выборке классификатор оказывается слишком точно подогнан под обучающие данные и плохо классифицирует тестовые.

Рассмотрим остальные классификаторы, использующие наивный байесовский метод. Классификатор *MultinomialNB* считает, что признаки имеют мультиномиальное распределение. Данный классификатор оптимальен

для классификации дискретных признаков, например счетчиков. Также он может работать и с дробными значениями, хотя и не оптимально. Данный классификатор ошибочно классифицировал 14 наблюдений, правильно 61. Точность классификации составила 81%.

Классификатор *ComplementNB* является модификацией *MultinomialNB*, заточенный под несбалансированные классы. Данный классификатор ошибочно классифицировал 21 наблюдений, правильно 54. Точность классификации составила 72%.

Классификатор *BernoulliNB* считает, что признаки имеют распределение Бернулли. Данный классификатор оптимален для классификации бинарных признаков. Поэтому для непрерывных данных он показал плохой результат: ошибочно было классифицировано 54 наблюдения, правильно 21. Точность классификации составила 28%.

Классифицирующие деревья

Классифицирующие деревья делят пространство признаков на подпространство. Деление происходит по линиям, параллельным осям координат признаков. Деление пространства заканчивается, когда в каждом подпространстве достигнуто достаточное значение некоторого критерия, при котором данное подпространство можно однозначно отнести к какому-либо классу.

Классифицируем те же данные при помощи классифицирующих деревьев. Ошибочно было классифицировано 3 наблюдения, правильно 72. Точность классификации составила 96%. Количество листьев классифицирующего дерева равно 5, глубина равна 4. Классифицирующее дерево представлено на рисунке 4.

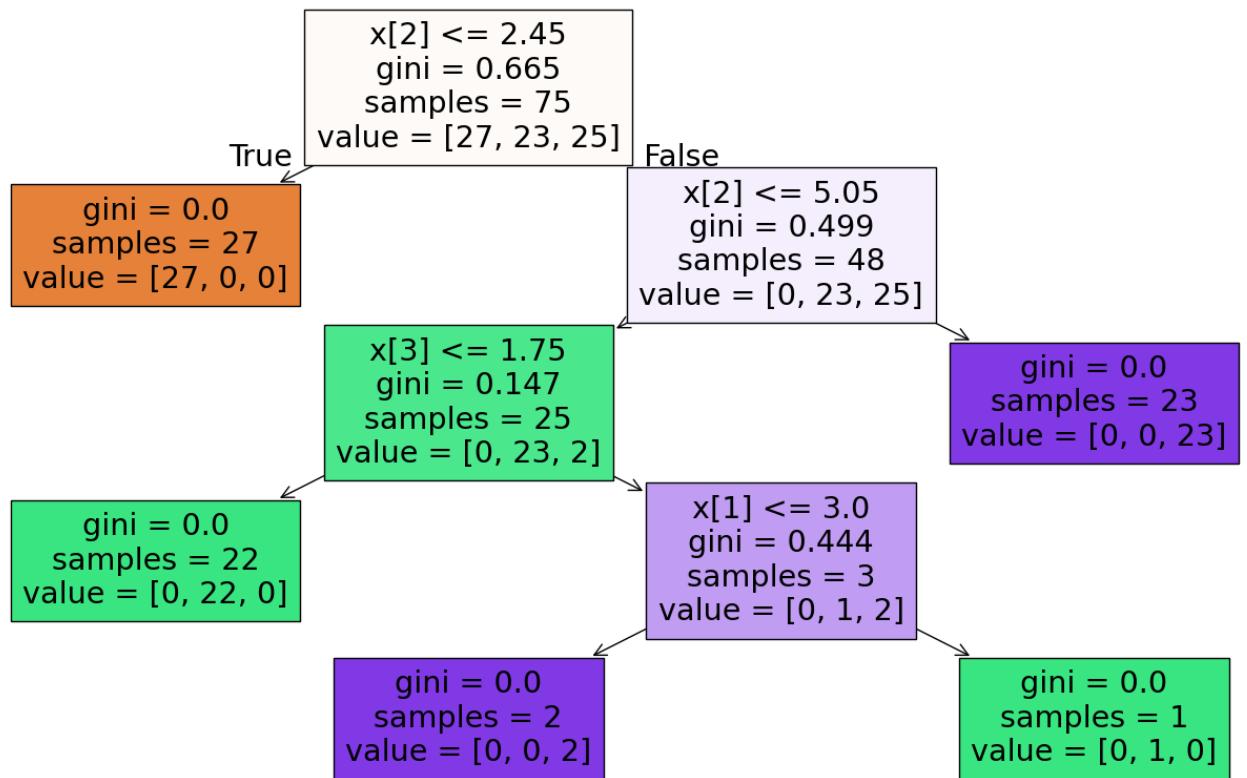


Рисунок 4 – Классифицирующее дерево

Для всех узлов дерева, не являющихся листьями указано условие. В случае, если это условие истинно, классификатор спускается по левой ветви, иначе – по правой. Атрибут *samples* за количество наблюдений, которой попало в данную ветвь при обучении. Атрибут *value* отвечает за то, к каким классам данные наблюдения относятся. Как можно заметить, в каждом листе все наблюдения относятся к одному классу. Атрибут *gini* отвечает за значение критерия, используемого для разделения пространства признаков на подпространства.

Построим графики зависимости количества неправильно классифицированных наблюдений и точности классификации от соотношения обучающей и тестовой выборок График зависимости количества неправильно классифицированных наблюдений от соотношения обучающей и тестовой выборок представлен на рисунке 5.

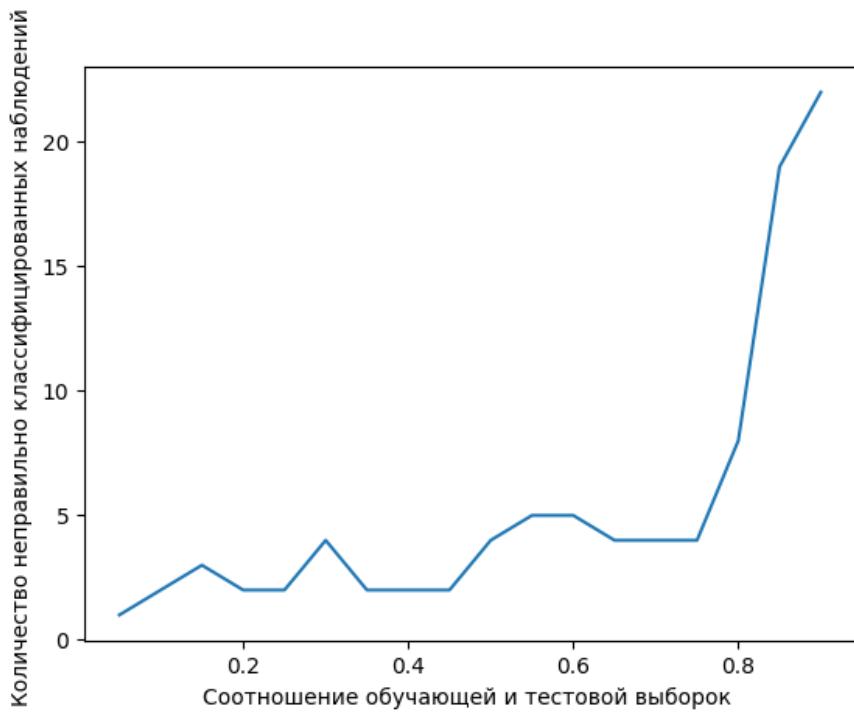


Рисунок 5 – График зависимости количества неправильно классифицированных наблюдений от соотношения обучающей и тестовой выборок

На рисунке 6 представлен график зависимости точности классификации от соотношения обучающей и тестовой выборок.

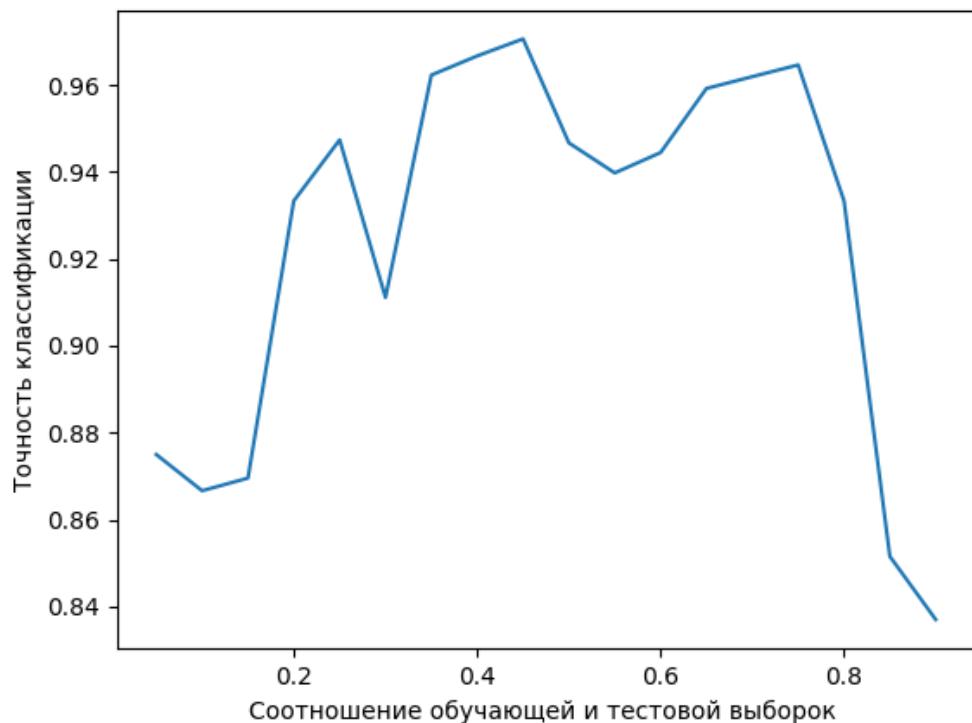


Рисунок 6 – График зависимости точности классификации классифицированных наблюдений от соотношения обучающей и тестовой выборок

Как можно заметить из рисунков 5 и 6, при слишком маленькой обучающей выборке количество неправильно классифицированных наблюдений резко возрастает, а точность классификации резко падает, что указывает на то, что такой выборки недостаточно для обучения дерева.

Параметр *criterion* отвечает за выбор критерия для разделения пространства признаков на подпространства. Его значения могут быть *entropy*, *gini* (по умолчанию) и *log_loss*. Изменим критерий на *entropy*. Полученное классифицирующее дерево представлено на рисунке 7.

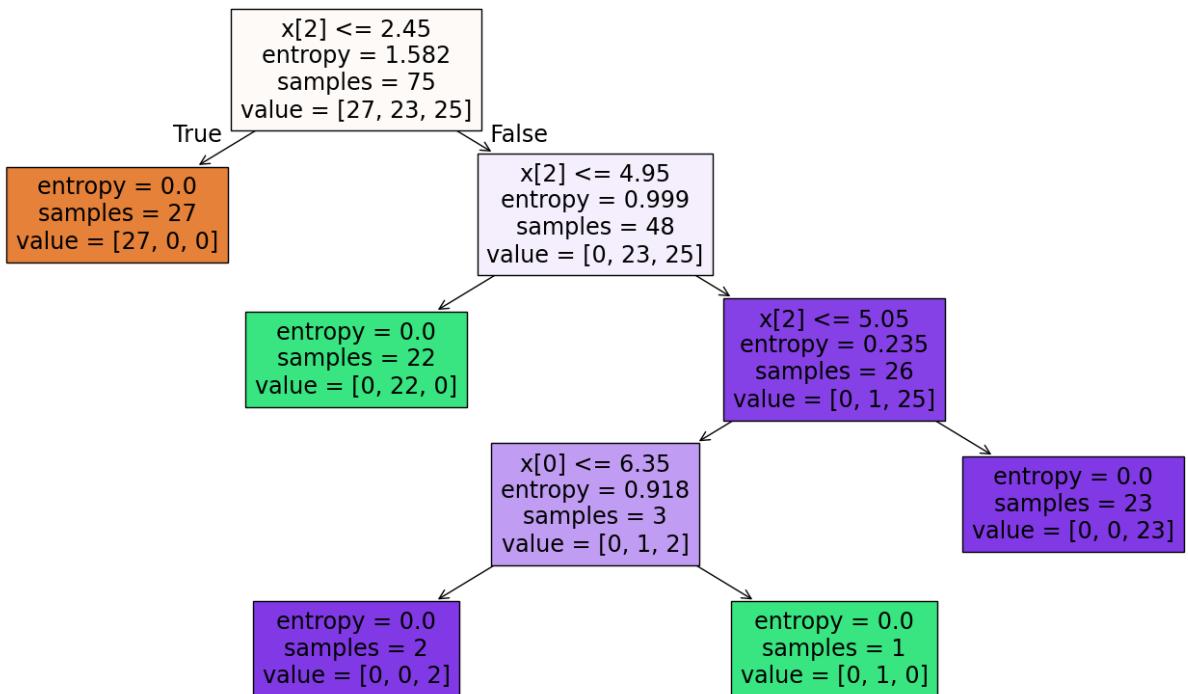


Рисунок 7 – Классифицирующее дерево при *criterion='entropy'*

Параметр *splitter* отвечает за то, как будет происходить разделение пространства признаков на подпространства. При значении *best* (по умолчанию) будет выбираться наилучшее разделения при помощи выбранного критерия. При значении *random* разделение будет иметь случайный характер. Изменим значение на *random*. Полученное классифицирующее дерево представлено на рисунке 8. Как можно заметить, поскольку разделение не являлось оптимальным, полученное дерево оказалось больше.

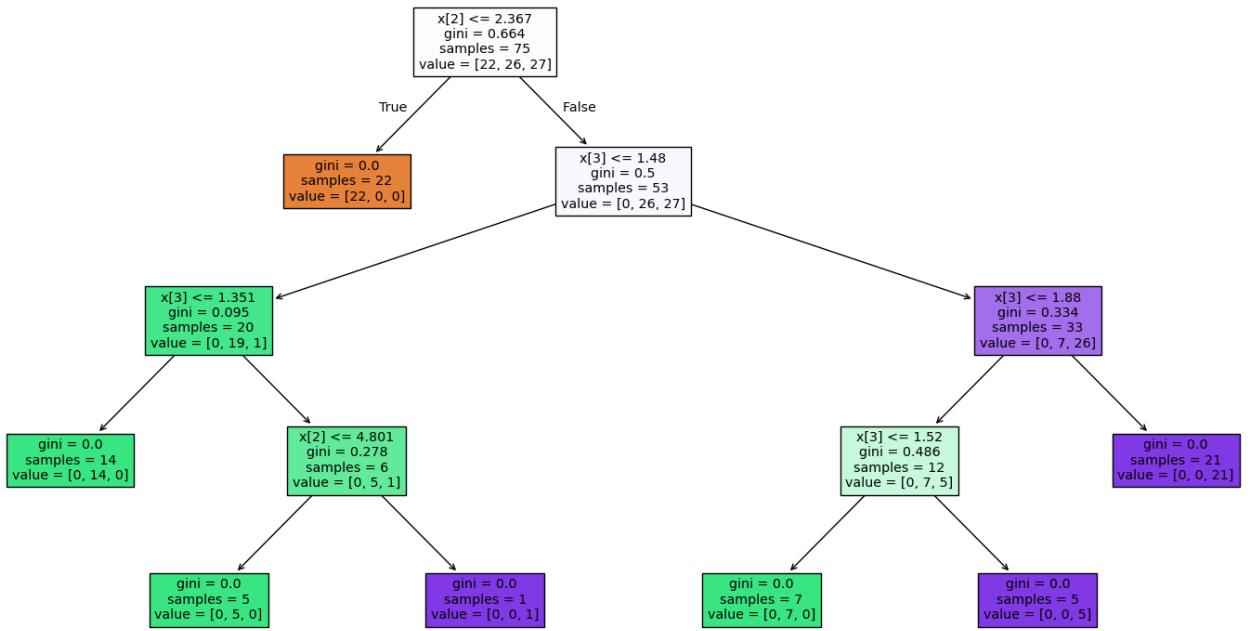


Рисунок 8 – Классифицирующее дерево при $\text{splitter}=\text{'random'}$

Параметр max_depth отвечает за максимальную глубину классифицирующего дерева. Изменим значение данного параметра на 2. Полученное классифицирующее дерево представлено на рисунке 9. Как можно заметить, построение дерева остановилось на глубине 2, хотя значение критерия не равно 0.

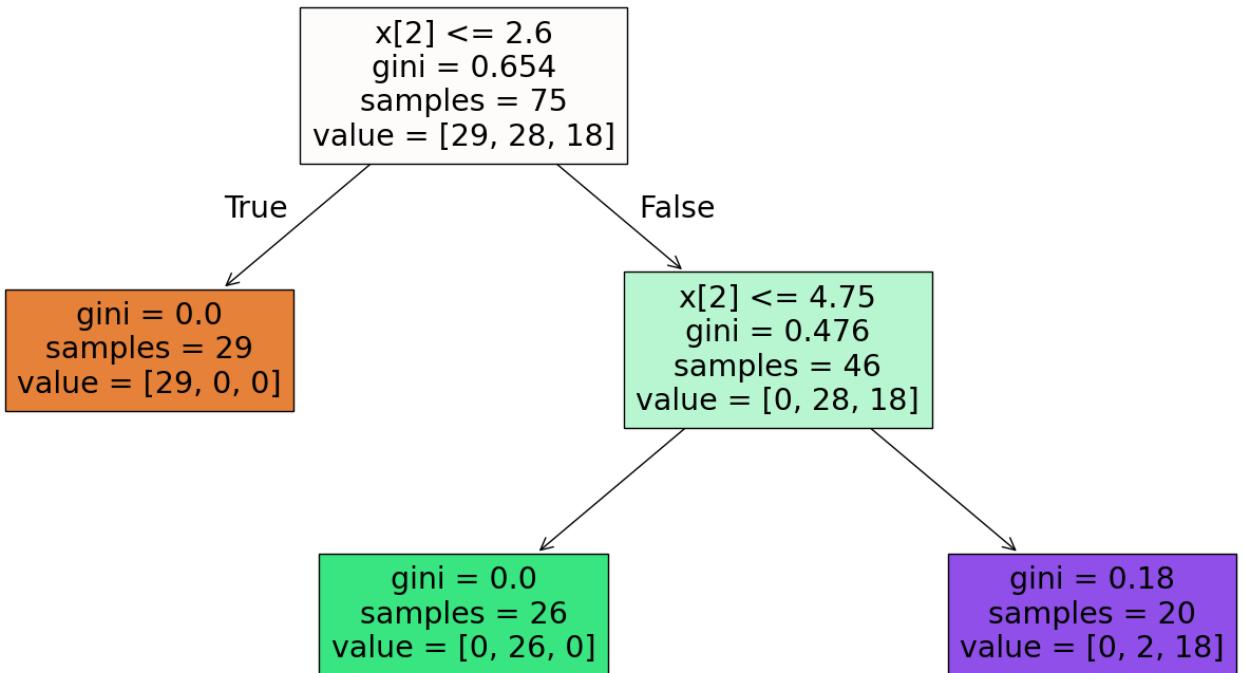


Рисунок 9 – Классифицирующее дерево при $\text{max_depth} = 2$

Параметр `min_samples_split` отвечает за минимальное количество наблюдений, при котором возможно дальнейшее разделение пространства. Изменим значение данного параметра на 30. Полученное классифицирующее дерево представлено на рисунке 10. Как можно заметить, построение дерева остановилось так как количество наблюдений стало меньше 30, хотя значение критерия не равно 0.

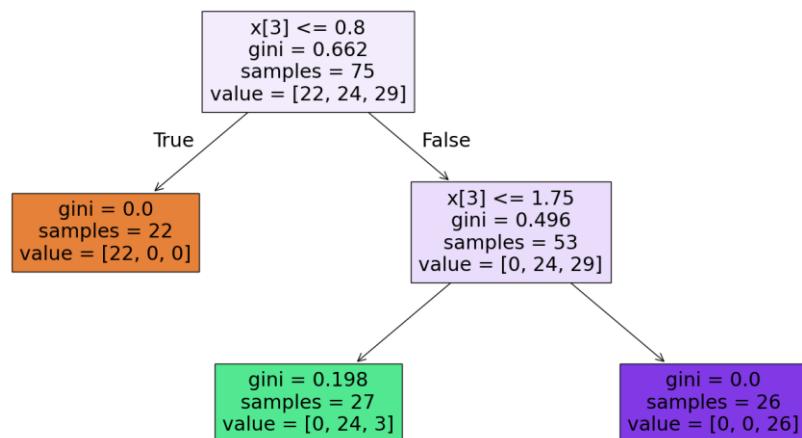


Рисунок 10 – Классифицирующее дерево при $min_samples_split = 30$

Параметр `min_samples_leaf` отвечает за минимальное количество наблюдений, при котором пространство может стать листом дерева. Изменим значение данного параметра на 30. Полученное классифицирующее дерево представлено на рисунке 11. Как можно заметить, построение дерева остановилось, поскольку при дальнейшем разделении количество наблюдений в листах стало бы меньше 30.

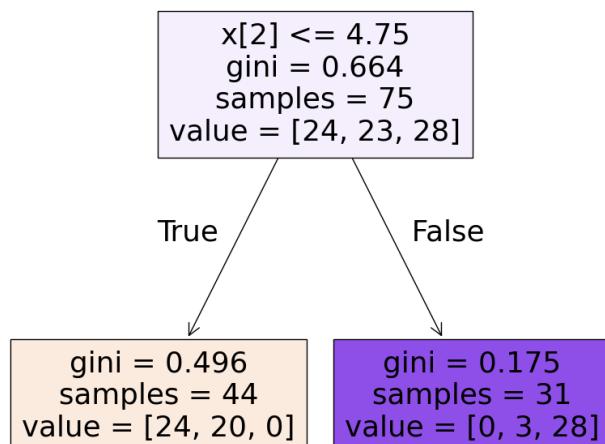


Рисунок 11 – Классифицирующее дерево при $min_samples_leaf = 30$

Выводы

В ходе выполнения лабораторной работы было проведено ознакомление с методами классификации модуля Sklearn.

Был изучен байесовский классификатор. Данный классификатор использует теорему Байеса для поиска вероятностей принадлежности наблюдения к классам и выбирает класс с наибольшей вероятностью. Наивный байесовский классификатор отличается от обычного тем, что предполагает независимость всех признаков. Была проведена классификация различными версиями наивного байесовского классификатора для нормально распределенных, мультиномиально распределенных, распределенных по Бернули данных. Также были построены графики зависимости неправильно классифицированных наблюдений и точности классификации от соотношения обучающей и тестовой выборок.

Был изучены классифицирующие деревья. Классифицирующие деревья делят пространство признаков на подпространства. Линии разделения параллельны осям координат признаком. Разделение происходит исходя из некоторого критерия и заканчивается при достижении определенного его значения. Данный классификатор был протестирован при различных его параметрах. Также были построены графики зависимости неправильно классифицированных наблюдений и точности классификации от соотношения обучающей и тестовой выборок.

Код программы, написанной для выполнения лабораторной работы представлен в приложении А.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn import model_selection
from sklearn import naive_bayes
from sklearn import tree
import matplotlib.pyplot as plt
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 = model_selection.train_test_split(X, Y,
test_size=0.5)

gnb = naive_bayes.GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print("GaussianNB:", (y_test != y_pred).sum(), gnb.score(X_test, y_test))

test_size=[s/100 for s in range(5, 95, 5)]
nc=[]
sc=[]
for ts in test_size:
    gnb = naive_bayes.GaussianNB()
    X_train, X_test, y_train, y_test = model_selection.train_test_split(X, Y,
test_size=ts, random_state=5)
    y_pred = gnb.fit(X_train, y_train).predict(X_test)
    nc.append((y_test != y_pred).sum())
    sc.append(gnb.score(X_test, y_test))
plt.plot(test_size,nc)
plt.xlabel("Соотношение обучающей и тестовой выборок ")
plt.ylabel("Количество неправильно классифицированных наблюдений ")
plt.show()
plt.plot(test_size,sc)
plt.xlabel("Соотношение обучающей и тестовой выборок ")
plt.ylabel("Точность классификации")
plt.show()

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, Y,
test_size=0.5)
mnb = naive_bayes.MultinomialNB()
y_pred = mnb.fit(X_train, y_train).predict(X_test)
print("MultinomialNB:", (y_test != y_pred).sum(), mnb.score(X_test, y_test))

bnb = naive_bayes.BernoulliNB()
y_pred = bnb.fit(X_train, y_train).predict(X_test)
print("BernoulliNB:", (y_test != y_pred).sum(), bnb.score(X_test, y_test))

gnb = naive_bayes.ComplementNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print("ComplementNB:", (y_test != y_pred).sum(), gnb.score(X_test, y_test))

clf = tree.DecisionTreeClassifier()
y_pred = clf.fit(X_train, y_train).predict(X_test)
print('wrong:', (y_test != y_pred).sum())
print('score:', clf.score(X_test, y_test))
print('n 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()

test_size=[s/100 for s in range(5,95,5)]
nc=[]
sc=[]
for ts in test_size:
    X_train, X_test, y_train, y_test = model_selection.train_test_split(X, Y,
test_size=ts,random_state=5)
    clf = tree.DecisionTreeClassifier()
    y_pred = clf.fit(X_train, y_train).predict(X_test)
    nc.append((y_test != y_pred).sum())
    sc.append(clf.score(X_test,y_test))
plt.plot(test_size,nc)
plt.xlabel("Соотношение обучающей и тестовой выборок ")
plt.ylabel("Количество неправильно классифицированных наблюдений ")
plt.show()
plt.plot(test_size,sc)
plt.xlabel("Соотношение обучающей и тестовой выборок ")
plt.ylabel("Точность классификации")
plt.show()

clf = tree.DecisionTreeClassifier(criterion='entropy')
y_pred = clf.fit(X_train, y_train).predict(X_test)
plt.subplots(1,1,figsize = (10,10))
tree.plot_tree(clf, filled = True)
plt.show()

clf = tree.DecisionTreeClassifier(splitter='random')
y_pred = clf.fit(X_train, y_train).predict(X_test)
plt.subplots(1,1,figsize = (10,10))
tree.plot_tree(clf, filled = True)
plt.show()

clf = tree.DecisionTreeClassifier(max_depth=2)
y_pred = clf.fit(X_train, y_train).predict(X_test)
plt.subplots(1,1,figsize = (10,10))
tree.plot_tree(clf, filled = True)
plt.show()

clf = tree.DecisionTreeClassifier(min_samples_split=30)
y_pred = clf.fit(X_train, y_train).predict(X_test)
plt.subplots(1,1,figsize = (10,10))
tree.plot_tree(clf, filled = True)
plt.show()

clf = tree.DecisionTreeClassifier(min_samples_leaf=30)
y_pred = clf.fit(X_train, y_train).predict(X_test)
plt.subplots(1,1,figsize = (10,10))
tree.plot_tree(clf, filled = True)
plt.show()

```