МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Информационно-вычислительные системы»

Курсовая работа

по дисциплине: «Математические и инструментальные методы поддержки принятия решений»

на тему «Разработка программного продукт бота для классификации грибов»

ПГУ 1.090403.04

Направление подготовки — 09.04.03 Прикладная информатика

Профиль — Прикладная информатика в экономике

Выполнил магистрант: Карамышева К.В.

Группа: 20ВЭм1

Руководитель

к.т.н., доцент Кузнецова О.Ю.

Работа защищена с оценкой отлично

Преподаватели _____

_____

_____

Дата защиты 23.12.2021

2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра "Информационно-вычислительные системы"

"УТВЕРЖДАЮ"
Зав. кафедрой _____
"_2_" "02" _____2021 г.

ЗАДАНИЕ

на курсовое проектирование по дисциплине «Математическое и инструментальные методы поддержки принятия решений»

Тема: «Разработка программного продукт бота для классификации грибов»

Данные представлены в одном из форматов: txt, pdf, doc, docx, xls, xslx, xml, csv. Необходимо преобразовать исходный набор данных к виду, пригодному для загрузки. Далее необходимо провести предобработку данных, например, разбить сложные атрибуты, очистить от технической информации, удалить спецсимволы. В обработанном наборе данных выделяются значимые атрибуты для дальнейшего анализа и построения модели машинного обучения.

На основе предобработанных данных предстоит построить модель машинного обучения. Как правило это модель кластеризации, классификации или регрессии. Необходимо определить точность работы модели и предпринять какие-либо действия для повышения точности.

На основе обученной модели необходимо разработать программный продукт бота, который будет автоматически определять значение целевой переменной по входному набору данных. Бот может быть реализован с командным или графическим интерфейсом. Для бота определяются формальные команды и возможность взаимодействовать с пользователем на естественном языке.

Исходные данные загрузить из репозитория Kaggle:
https://www.kaggle.com/uciml/mushroom-classification

Руководитель работы _____ Кузнецова О.Ю.

Реферат

Пояснительная записка 62 листа, 35 рисунков.

Объектом исследования являются грибы.

Цель работы – разработка продукт бота для классификации грибов с использованием методов машинного обучения: деревья решений, логистическая регрессия, К-ближайших соседей, опорных векторов, нейронная сеть (персептрон), многослойная нейронная сеть.

В результате проделанной работы был создан продукт бот для классификации грибов.

Анализ данных проводился с использованием программного средства Jupyter Notebook.

					ПГУ 1.09.04.03		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Карамышева К.В.		22.12.21	Разработка программного продукт бота для классификации грибов Пояснительная записка	Лит.	Лист 3 / Листов 62
Провер.		Кузнецова О.Ю.		23.12.21			
Реценз.						Гр. 20ВЭм1	
Н. Контр.							
Утв.							

Содержание

1 Постановка задачи

В данной курсовой работе необходимо классифицировать грибы с помощью методов машинного обучения и создать продукт бот.

Исходные данные были загружены из Репозитория Kaggle. Это онлайн-сообщество Data Scientist'ов и специалистов по машинному обучению (machine learning). Kaggle позволяет пользователям находить или публиковать датасеты, строить модели в специальной среде Kernel, работать с другими ML-специалистами и участвовать в соревнованиях в области Data Science.

Набор данных включает описания гипотетических образцов, соответствующих 23 видам грибов с жабрами в семействе грибов Agaricus и Lepiota, взятых из Полевого руководства Общества Одюбона по североамериканским грибам (1981).

Входные данные включают признаки грибов (например цвет шляпки, запах и др.), а выходные данные идентифицирует гриб как определенно съедобный или определенно ядовитый.

Данная выборка содержит 8124 экземпляров. Количество атрибутов – 22 + выходной атрибут.

Для решения поставленной задачи будут использоваться методы машинного о бучения: дерево решений, логистическая регрессия, К-ближайших соседей, опорных векторов, нейронная сеть (персептрон), многослойная нейронная сеть.

Входные поля:

1. cap-shape – форма шляпки (строковый тип);
2. cap-surface – поверхность шляпки (строковый тип);
3. cap-color – цвет шляпки (строковый тип);
4. bruises – синяки (логический тип);
5. odor – запах (строковый тип);
6. gill-attachment – прикрепление жабр (логический тип);
7. gill-spacing – расстояние между жабрами (строковый тип);
8. gill-size – размер жабр (строковый тип);
9. gill-color – цвет жабр (строковый тип);
10. stalk-shape – форма стебля (строковый тип);

11. stalk-root – корень стебля (строковый тип);
12. stalk-surface-above-ring – поверхность стебля над кольцом (строковый тип);
13. stalk-surface-below-ring – поверхность стебля под кольцом (строковый тип);
14. stalk-color-above-ring – цвет стебля над кольцом (строковый тип);
15. stalk-color-below-ring - цвет стебля под кольцом (строковый тип);
16. veil-type – тип вуали (строковый тип);
17. veil-color – цвет вуали (строковый тип);
18. ring-number – количество колец (строковый тип);
19. ring-type – тип кольца (строковый тип);
20. spore-print-color – цвет спор (строковый тип);
21. population – популяция (строковый тип);
22. habitat – среда обитания (строковый тип).

Выходное поле – class - класс (строковый тип).

2 Решение задачи

2.1 Подготовка данных к анализу

Для решения поставленной задачи будет использоваться программное средство Jupyter Notebook. В начале, необходимо импортировать исходный файл с данными «mushrooms.csv». Результат импорта файла представлен на рисунке 1.

```
f = pd.read_csv('mushrooms.csv', delimiter=r'\s*,\s*', engine='python')
f
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	populatio
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8119	e	k	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o	p	b	
8120	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	n	o	p	b	
8121	e	f	s	n	f	n	a	c	b	n	...	s	o	o	p	o	o	p	b	
8122	p	k	y	n	f	y	f	c	n	b	...	k	w	w	p	w	o	e	w	
8123	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o	p	o	

8124 rows × 23 columns

Рисунок 1 - Результат импорта файла mushrooms.csv

После импорта на экран было выведен размер, тип данных и статистические данные. Результаты статистических данных представлены на рисунке 2.

```
# выведем размер и тип данных
print(f.shape)
print(f.dtypes)
```

```
(8124, 23)
class                       object
cap-shape                   object
cap-surface                 object
cap-color                   object
bruises                     object
odor                        object
gill-attachment             object
gill-spacing                object
gill-size                   object
gill-color                  object
stalk-shape                 object
stalk-root                  object
stalk-surface-above-ring    object
stalk-surface-below-ring    object
stalk-color-above-ring      object
stalk-color-below-ring      object
veil-type                   object
veil-color                  object
ring-number                 object
ring-type                   object
spore-print-color           object
population                  object
habitat                     object
dtype: object
```

```
# выведем статистику
f.describe()
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	popula
count	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	...	8124	8124	8124	8124	8124	8124	8124	8124	
unique	2	6	4	10	2	9	2	2	2	12	...	4	9	9	1	4	3	5	9	
top	e	x	y	n	f	n	f	c	b	b	...	s	w	w	p	w	o	p	w	
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	...	4936	4464	4384	8124	7924	7488	3968	2388	

4 rows × 23 columns

Рисунок 2 – Статистические данные

Для того чтобы использовать в дальнейшем статистические методы, необходимо преобразовать нашу выборку из буквенных значений в числовые. Для этого закодируем выборку с помощью метода Ordinal Encoder модуля preprocessing библиотеки sklearn. Затем трансформируем буквенные значение с помощью метода transform. Для удобства закодированные данные загрузим в новый файл «mushrooms1.csv» и импортируем в программу. Результаты кодирования представлены на рисунке 3.

```
a=np.array(f)
print(a)
# separate the data from the target attributes
enc = preprocessing.OrdinalEncoder()
enc.fit(a)
v=enc.transform(a)
v
```

```
[['p' 'x' 's' ... 'k' 's' 'u']
 ['e' 'x' 's' ... 'n' 'n' 'g']
 ['e' 'b' 's' ... 'n' 'n' 'm']
 ...
 ['e' 'f' 's' ... 'b' 'c' 'l']
 ['p' 'k' 'y' ... 'w' 'v' 'l']
 ['e' 'x' 's' ... 'o' 'c' 'l']]

array([[1., 5., 2., ..., 2., 3., 5.],
       [0., 5., 2., ..., 3., 2., 1.],
       [0., 0., 2., ..., 3., 2., 3.],
       ...,
       [0., 2., 2., ..., 0., 1., 2.],
       [1., 3., 3., ..., 7., 4., 2.],
       [0., 5., 2., ..., 4., 1., 2.]])
```

```
import pandas as pd
pd.DataFrame(v).to_csv("mushrooms1.csv")
```

```
df = pd.read_csv('mushrooms1.csv', delimiter=r'\s*,\s*', engine='python')
del df['Unnamed: 0']
df
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	populatio
0	1.0	5.0	2.0	4.0	1.0	6.0	1.0	0.0	1.0	4.0	...	2.0	7.0	7.0	0.0	2.0	1.0	4.0	2.0	3.
1	0.0	5.0	2.0	9.0	1.0	0.0	1.0	0.0	0.0	4.0	...	2.0	7.0	7.0	0.0	2.0	1.0	4.0	3.0	2.
2	0.0	0.0	2.0	8.0	1.0	3.0	1.0	0.0	0.0	5.0	...	2.0	7.0	7.0	0.0	2.0	1.0	4.0	3.0	2.
3	1.0	5.0	3.0	8.0	1.0	6.0	1.0	0.0	1.0	5.0	...	2.0	7.0	7.0	0.0	2.0	1.0	4.0	2.0	3.
4	0.0	5.0	2.0	3.0	0.0	5.0	1.0	1.0	0.0	4.0	...	2.0	7.0	7.0	0.0	2.0	1.0	0.0	3.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
8119	0.0	3.0	2.0	4.0	0.0	5.0	0.0	0.0	0.0	11.0	...	2.0	5.0	5.0	0.0	1.0	1.0	4.0	0.0	1.
8120	0.0	5.0	2.0	4.0	0.0	5.0	0.0	0.0	0.0	11.0	...	2.0	5.0	5.0	0.0	0.0	1.0	4.0	0.0	4.
8121	0.0	2.0	2.0	4.0	0.0	5.0	0.0	0.0	0.0	5.0	...	2.0	5.0	5.0	0.0	1.0	1.0	4.0	0.0	1.
8122	1.0	3.0	3.0	4.0	0.0	8.0	1.0	0.0	1.0	0.0	...	1.0	7.0	7.0	0.0	2.0	1.0	0.0	7.0	4.
8123	0.0	5.0	2.0	4.0	0.0	5.0	0.0	0.0	0.0	11.0	...	2.0	5.0	5.0	0.0	1.0	1.0	4.0	4.0	1.

8124 rows × 23 columns

Рисунок 3 – Закодированные данные

Теперь необходимо провести подготовку данных: проверить выборку на пропущенные значение, выбросы, дубликаты, неинформативные показатели, определить закон распределения показателей.

Для начала необходимо исследовать выходную переменную. Для этого выведем диаграмму рассеяния, чтобы увидеть, есть ли в данных выбросы. Диаграмма представлена на рисунке 4.

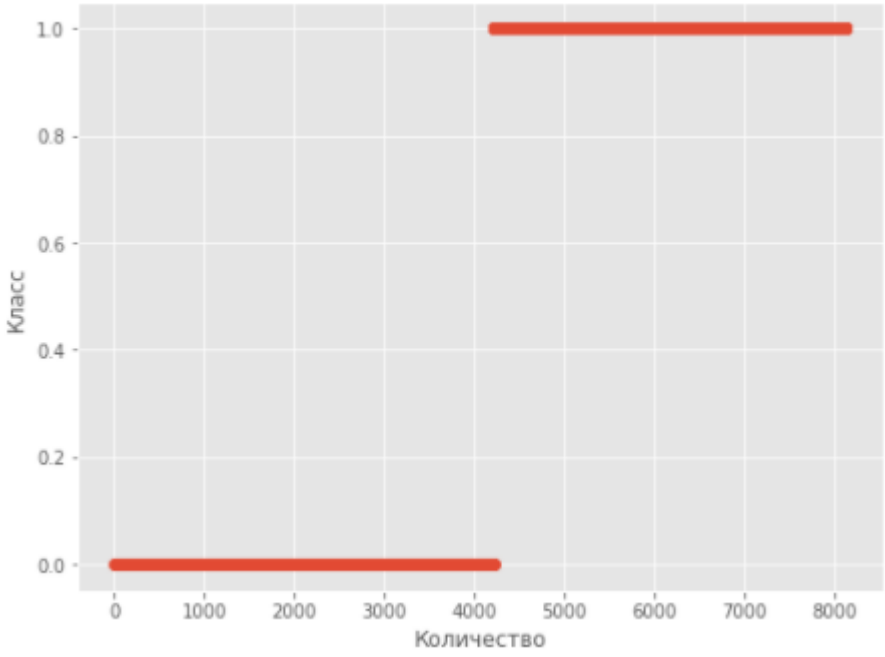


Рисунок 4 – Диаграмма рассеяния выходной переменной class

График выглядит нормально. Так как у нас выходная переменная может принимать значения либо 0, либо 1.

Далее необходимо проверить выборку на отсутствующие значения. Для этого вернемся к исходной выборке с буквенными показателями. Из представленной выше статистики (рисунок 1) мы можем заметить, что нулевых значений или значений NaN не обнаружено, количество строк совпадает с количеством экземпляров. Выведем выборку со всеми столбцами (рисунок 5). Теперь можно заметить столбец stalk-root, который имеет значение знак вопроса (?).

```
pd.options.display.max_columns = None
f
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	rin typ
0	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	
1	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	
2	e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	
3	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	
4	e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8119	e	k	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	o	o	
8120	e	x	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	n	o	
8121	e	f	s	n	f	n	a	c	b	n	e	?	s	s	o	o	p	o	o	
8122	p	k	y	n	f	y	f	c	n	b	t	?	s	k	w	w	p	w	o	
8123	e	x	s	n	f	n	a	c	b	y	e	?	s	s	o	o	p	o	o	

8124 rows × 23 columns

```
f['stalk-root']
```

```
0       e
1       c
2       c
3       e
4       e
       ..
8119    ?
8120    ?
8121    ?
8122    ?
8123    ?
Name: stalk-root, Length: 8124, dtype: object
```

Рисунок 5 – Выборка со всеми столбцами

Можно легко пометить значения как NaN с помощью Pandas DataFrame, используя функцию replace() на подмножестве интересующих нас столбцов (рисунок 6).

```
n = f
n = n.replace('?', np.NaN)
n
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	rin typ
0	p	x	s	n	t	p	f	c	n	k	e	e	s	s	w	w	p	w	o	
1	e	x	s	y	t	a	f	c	b	k	e	c	s	s	w	w	p	w	o	
2	e	b	s	w	t	l	f	c	b	n	e	c	s	s	w	w	p	w	o	
3	p	x	y	w	t	p	f	c	n	n	e	e	s	s	w	w	p	w	o	
4	e	x	s	g	f	n	f	w	b	k	t	e	s	s	w	w	p	w	o	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8119	e	k	s	n	f	n	a	c	b	y	e	NaN	s	s	o	o	p	o	o	
8120	e	x	s	n	f	n	a	c	b	y	e	NaN	s	s	o	o	p	n	o	
8121	e	f	s	n	f	n	a	c	b	n	e	NaN	s	s	o	o	p	o	o	
8122	p	k	y	n	f	y	f	c	n	b	t	NaN	s	k	w	w	p	w	o	
8123	e	x	s	n	f	n	a	c	b	y	e	NaN	s	s	o	o	p	o	o	

8124 rows × 23 columns

Рисунок 6 – Измененные значения в переменной stalk-root

Построим процентное соотношение пропущенных значений по каждому столбцу (рисунок 7).

```
class - 0%
cap-shape - 0%
cap-surface - 0%
cap-color - 0%
bruises - 0%
odor - 0%
gill-attachment - 0%
gill-spacing - 0%
gill-size - 0%
gill-color - 0%
stalk-shape - 0%
stalk-root - 31%
stalk-surface-above-ring - 0%
stalk-surface-below-ring - 0%
stalk-color-above-ring - 0%
stalk-color-below-ring - 0%
veil-type - 0%
veil-color - 0%
ring-number - 0%
ring-type - 0%
spore-print-color - 0%
population - 0%
habitat - 0%
```

Рисунок 7 – Процентное соотношение пропущенных значений

Таким образом, отсутствующие значения находятся только в столбце stalk-root. Проверим зависимость параметра stalk-root и выходного параметра - class с помощью матрицы корреляции (рисунок 8).

```
sns.heatmap(df.corr(),annot=True,cmap='RdYlGn',linewidths=0.2)
fig=plt.gcf()
fig.set_size_inches(12,12)
plt.show()
```

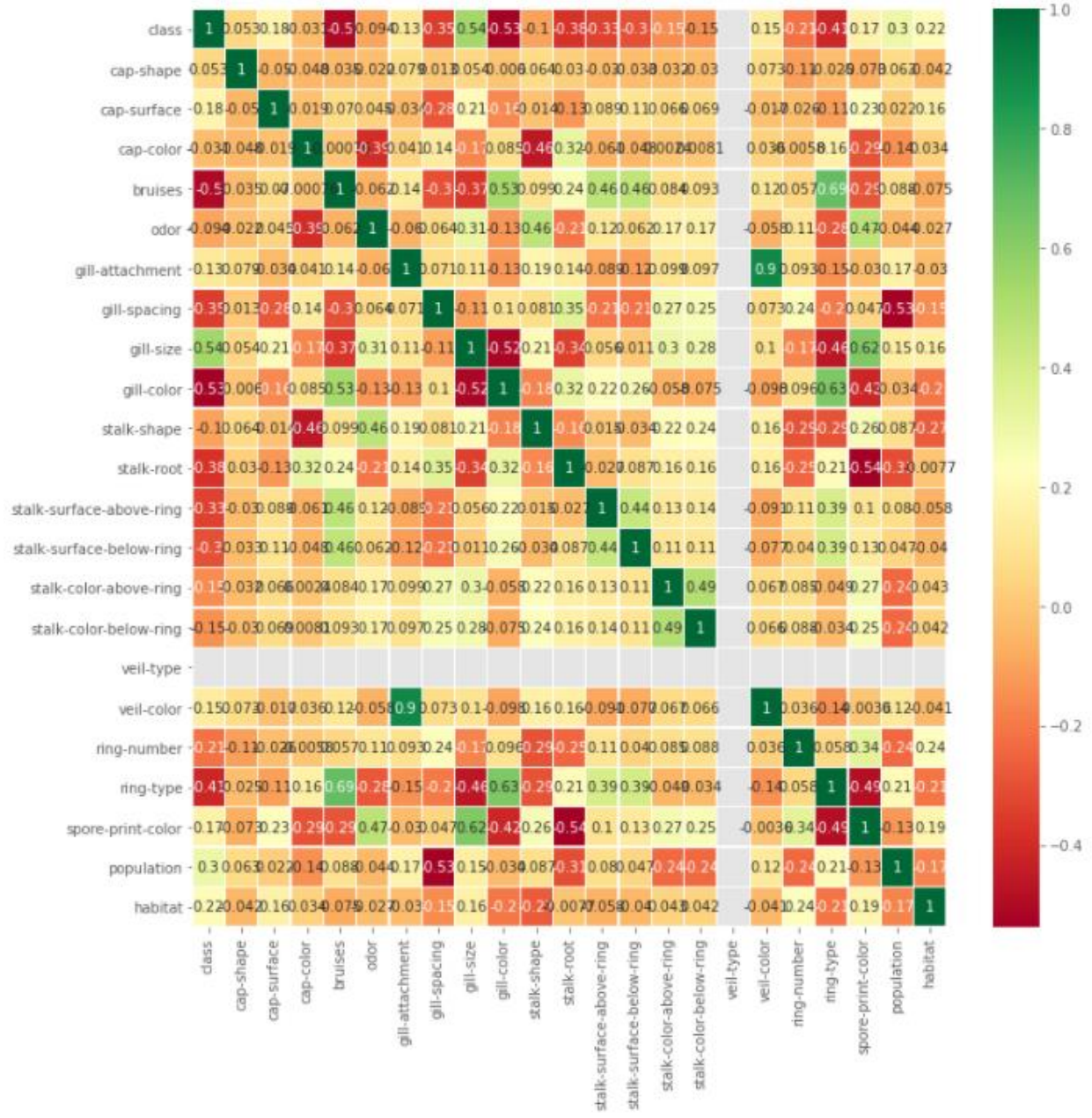


Рисунок 8 – Матрица корреляции

Так как показатель зависимости низкий и равен - 0,38, мы можем удалить параметр stalk-root. Также обнаруживается сильная зависимость между показателями:

- veil-color и gill-attachment;
- bruises и ring-type.

На основе этого, мы можем удалить один из параметров в каждой паре. В итоге мы удалили параметры stalk-root, veil-color и bruises.

Отсутствующих значений больше нет. Следующим шагом будет выявление нетипичных данных. Выбросы – это данные, которые существенно отличаются от других наблюдений. Они могут соответствовать реальным отклонениям, но могут быть и просто ошибками.

Так как данные выборки дискретные, мы не можем верно определить выбросы. Поэтому переходим к следующему шагу.

Если значения признаков (всех или большинства) в двух разных записях совпадают, эти записи называются дубликатами. Из-за специфики предметной области мы не сможем удалить дубликаты записей, так как потеряем большое количество данных.

Следующим шагом будет определение закона распределения показателей. При использовании описательной статистики важно учитывать тип данных и параметры распределения, характеризующиеся показателями асимметрии и эксцесса. Асимметрия и эксцесс представляют меру отклонения формы распределения от нормального вида. Если значения эксцесса и асимметрии превышают 0, т.е. данные не подчиняются нормальному закону распределения, и имеют более острые вершины, чем у нормального закона распределения. Проверим распределение с помощью гистограмм и показателей асимметрии и эксцесса по всем показателям (рисунок 9-10).

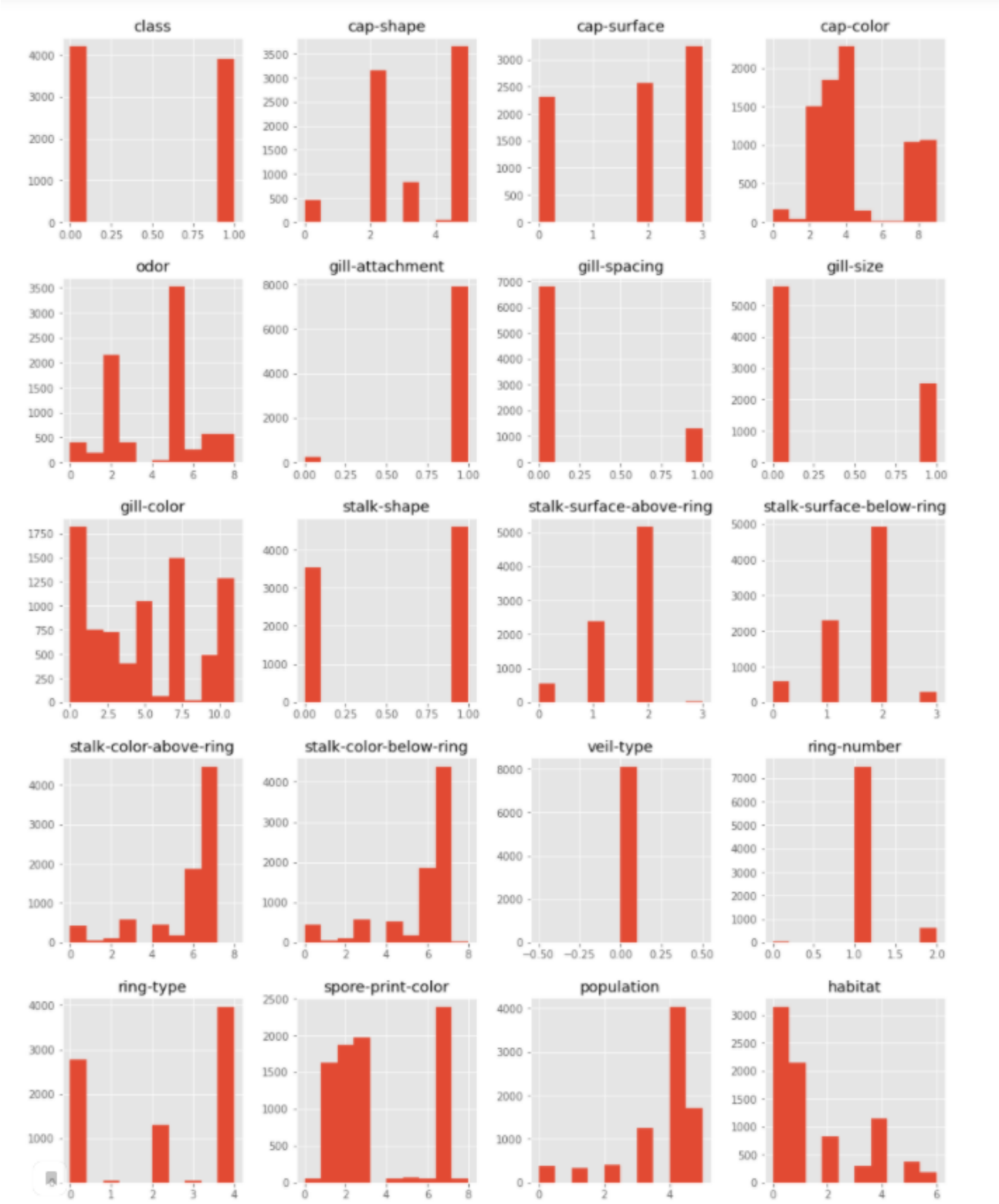


Рисунок 9 – Гистограммы всех показателей

```
columns = list(df)
for col in columns:
    kurt=sts.kurtosis(df[col], axis=0, fisher=True, bias=True)
    sk= sts.skew(df[col], axis=0, bias=True)
    print('Показатели асимметрия и эксцесс для: {}'.format(col))
    print(kurt,sk)
```

```
Показатели асимметрия и эксцесс для: class
-1.9948257514263408 0.07193224988597952
Показатели асимметрия и эксцесс для: cap-shape
-1.2423869516263428 -0.24700637928120617
Показатели асимметрия и эксцесс для: cap-surface
-1.2769525649055313 -0.5907497451740007
Показатели асимметрия и эксцесс для: cap-color
-0.8364851781178197 0.7068342913350331
Показатели асимметрия и эксцесс для: odor
-0.7768659647557654 -0.08077478619378756
Показатели асимметрия и эксцесс для: gill-attachment
33.71224953969458 -5.975972685654995
Показатели асимметрия и эксцесс для: gill-spacing
1.384674462569639 1.83974847807238
Показатели асимметрия и эксцесс для: gill-size
-1.3183113074349329 0.8256444104849662
Показатели асимметрия и эксцесс для: gill-color
-1.286056382005263 0.061398853411354266
Показатели асимметрия и эксцесс для: stalk-shape
-1.9263989618885098 -0.27129511258312494
Показатели асимметрия и эксцесс для: stalk-surface-above-ring
0.23952037027040562 -1.0985365054493224
Показатели асимметрия и эксцесс для: stalk-surface-below-ring
0.22881765071125182 -0.7575632718648749
Показатели асимметрия и эксцесс для: stalk-color-above-ring
2.4998856910837386 -1.8350954640494637
Показатели асимметрия и эксцесс для: stalk-color-below-ring
2.362455489279915 -1.7912626088095116
Показатели асимметрия и эксцесс для: veil-type
-3.0 0.0
Показатели асимметрия и эксцесс для: ring-number
8.339208763693595 2.701158403504732
Показатели асимметрия и эксцесс для: ring-type
-1.7084543184642098 -0.28996465324345516
Показатели асимметрия и эксцесс для: spore-print-color
-1.3385083249457046 0.5483245789238169
Показатели асимметрия и эксцесс для: population
1.674787779261619 -1.4128347513176136
Показатели асимметрия и эксцесс для: habitat
-0.25904041032408953 0.9853656933222494
```

Рисунок 10 – Показатели асимметрии и эксцесса по всем показателям

Как видно результаты показывают, что значения эксцесса по многим показателям превышают 0, т.е. данные не подчиняются нормальному закону распределения, и имеют более острые вершины, чем у нормального закона распределения. По значениям асимметрии по многим показателям смещены относительно среднего значения. Распределение большинства показателей смещено относительно среднего значения.

Из этого следует, что для обработки данных доступны не все методы статистического анализа. Будем применять непараметрические методы, которые позволяют исследовать данные без каких-либо допущений о характере

распределения переменных, в том числе при нарушении требования нормальности распределения.

Далее проверим выборку на неинформативные показатели. Если признак имеет слишком много строк с одинаковыми значениями, он не несет полезной информации для проекта.

Составим список признаков, у которых более 95% строк содержат одно и то же значение (рисунок 11).

```
51.7971442639094
45.00246184145741
39.931068439192515
28.114229443623827
43.42688330871492
97.41506646971935
gill-attachment: 97.41507%
1.0    7914
0.0     210
Name: gill-attachment, dtype: int64

83.85032003938946
69.0792712949286
21.270310192023633
56.72082717872969
63.712456917774496
60.75824716888233
54.94830132939439
53.96356474643033
100.0
veil-type: 100.00000%
0.0    8124
Name: veil-type, dtype: int64

92.17134416543574
48.842934515017234
29.394387001477106
49.72919743968489
38.74938453963564
```

Рисунок 11 – Список признаков с процентным соотношением количества одинаковых значений

На основе этого списка мы можем заметить два показателя, у которых более 95% строк содержат одно и то же значение: gill-attachment, veil-type. Пока мы можем удалить только параметр veil-type, т.к. он имеет только одно значение и является неинформативным признаком.

Для выбора информативных показателей так же можно использовать методы feature selection. Feature selection – это методы, при которых мы выбираем те переменные в наших данных, которые больше всего влияют на целевую переменную. Другими словами, мы выбираем лучшие предикторы для целевой переменной.

Для начала необходимо разделить переменные на выходную - y и входные – X. Затем применить класс SelectKBest, чтобы извлечь лучшие показатели, и объединить два фрейма данных для лучшей визуализации (рисунок 12). Графическое представление информативности показателей представлено на рисунке 13

	Показатель	Счет
7	gill-color	5957.764469
14	ring-type	1950.610146
6	gill-size	1636.606833
5	gill-spacing	826.795274
17	habitat	751.309489
15	spore-print-color	379.132729
16	population	311.766736
9	stalk-surface-above-ring	222.982400
1	cap-surface	214.068544
10	stalk-surface-below-ring	206.648180
11	stalk-color-above-ring	119.792216
12	stalk-color-below-ring	109.789410
3	odor	75.910163
8	stalk-shape	36.594105
13	ring-number	25.646335
0	cap-shape	17.508364
2	cap-color	11.511382
4	gill-attachment	3.505447

Рисунок 12 – Таблица информативности показателей

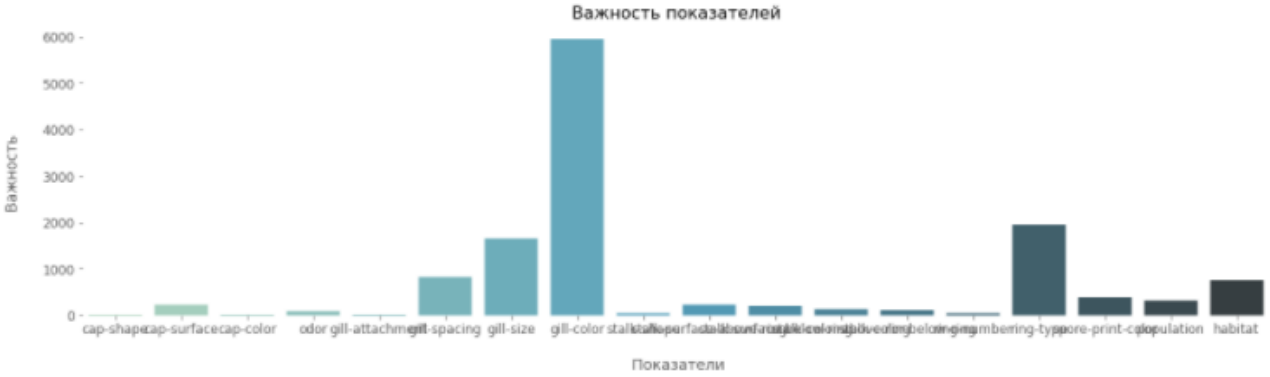


Рисунок 13 – Графическое представление информативности

Используя метод фильтрации Хи2 для отбора показателей мы получили следующий результат. Наиболее информативными показателями являются:

– gill-color;
– ring-type;
– gill-size;
– gill-spacing;
– habitat;

– spore-print-color;

– population;

– stalk-surface-above-ring;

– cap-surface;

– stalk-surface-below-ring;

– stalk-color-above-ring;

– stalk-color-below-ring;

– odor.

Наиболее неинформативными показателями являются:

- stalk-shape
- ring-number
- cap-shape
- cap-color
- gill-attachment

Поэтому мы можем их удалить.

Мы подготовили выборку к использованию методов машинного обучения. Итоговая выборка состоит из 13 входных переменных и одной выходной, количество экземпляров 8124. Результат представлен на рисунке 14.

```
df
```

	class	cap-surface	odor	gill-spacing	gill-size	gill-color	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	ring-type	spore-print-color	population	habitat
0	1.0	2.0	6.0	0.0	1.0	4.0	2.0	2.0	7.0	7.0	4.0	2.0	3.0	5.0
1	0.0	2.0	0.0	0.0	0.0	4.0	2.0	2.0	7.0	7.0	4.0	3.0	2.0	1.0
2	0.0	2.0	3.0	0.0	0.0	5.0	2.0	2.0	7.0	7.0	4.0	3.0	2.0	3.0
3	1.0	3.0	6.0	0.0	1.0	5.0	2.0	2.0	7.0	7.0	4.0	2.0	3.0	5.0
4	0.0	2.0	5.0	1.0	0.0	4.0	2.0	2.0	7.0	7.0	0.0	3.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8119	0.0	2.0	5.0	0.0	0.0	11.0	2.0	2.0	5.0	5.0	4.0	0.0	1.0	2.0
8120	0.0	2.0	5.0	0.0	0.0	11.0	2.0	2.0	5.0	5.0	4.0	0.0	4.0	2.0
8121	0.0	2.0	5.0	0.0	0.0	5.0	2.0	2.0	5.0	5.0	4.0	0.0	1.0	2.0
8122	1.0	3.0	8.0	0.0	1.0	0.0	2.0	1.0	7.0	7.0	0.0	7.0	4.0	2.0
8123	0.0	2.0	5.0	0.0	0.0	11.0	2.0	2.0	5.0	5.0	4.0	4.0	1.0	2.0

8124 rows × 14 columns

Рисунок 14 – Преобразованная выборка

2.2 Использование метода машинного обучения «Деревья решений»

Classification and Regression Trees (CART) часто используются в задачах, в которых объекты имеют категориальные признаки и используется для задач регрессии и классификации. Очень хорошо деревья подходят для многоклассовой классификации

Дерево решений - по сути, разбиение пространства на многомерные параллелепипеды, для каждой области соответствует простая модель - решающее правило.

Прежде всего, необходимо разбить нашу выборку на обучающие и тестовые данные.

Наблюдения в обучающей выборке (training set) содержат опыт, который алгоритм использует для обучения. В задачах обучения с учителем каждое наблюдение состоит из наблюдаемой (зависимой) переменной и одной или нескольких независимых переменных.

Тестовое множество, или тестовая выборка, представляет из себя аналогичный набор наблюдений, который используется для оценки качества модели, используя некоторые показатели.

Важно, чтобы никакие наблюдения из обучающей выборки не были включены в тестовую выборку. Если тестовые данные действительно содержат примеры из обучающей выборки, то будет трудно оценить, научился ли алгоритм обобщать, используя обучающую выборку или же просто запомнил данные. Программа, которая хорошо обобщает, будет в состоянии эффективно выполнять задачи с новыми данными. И наоборот, программа, которая запоминит обучающие данные, создав чрезмерно сложную модель, может точно предсказывать значения зависимой переменной для обучающего множества, но не сможет предсказать значение зависимой переменной для новых примеров.

Как правило, единая выборка наблюдений, используемых для обучения, разделяется на обучающее, тестовое и проверочное множества. Не существует каких-то особенных требований к размерам таких множеств, и они могут изменяться в соответствии с количеством имеющихся данных.

Разделенная выборка в соотношении 80% к 20% представлена на рисунке 15.

```
# определяем x и y
y = df['class']
x = df.drop(['class'], axis = 1)

# деление на обучающую и тестовую выборки: 80 % - 20 %
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=29)
```

```
x_train
x_test
```

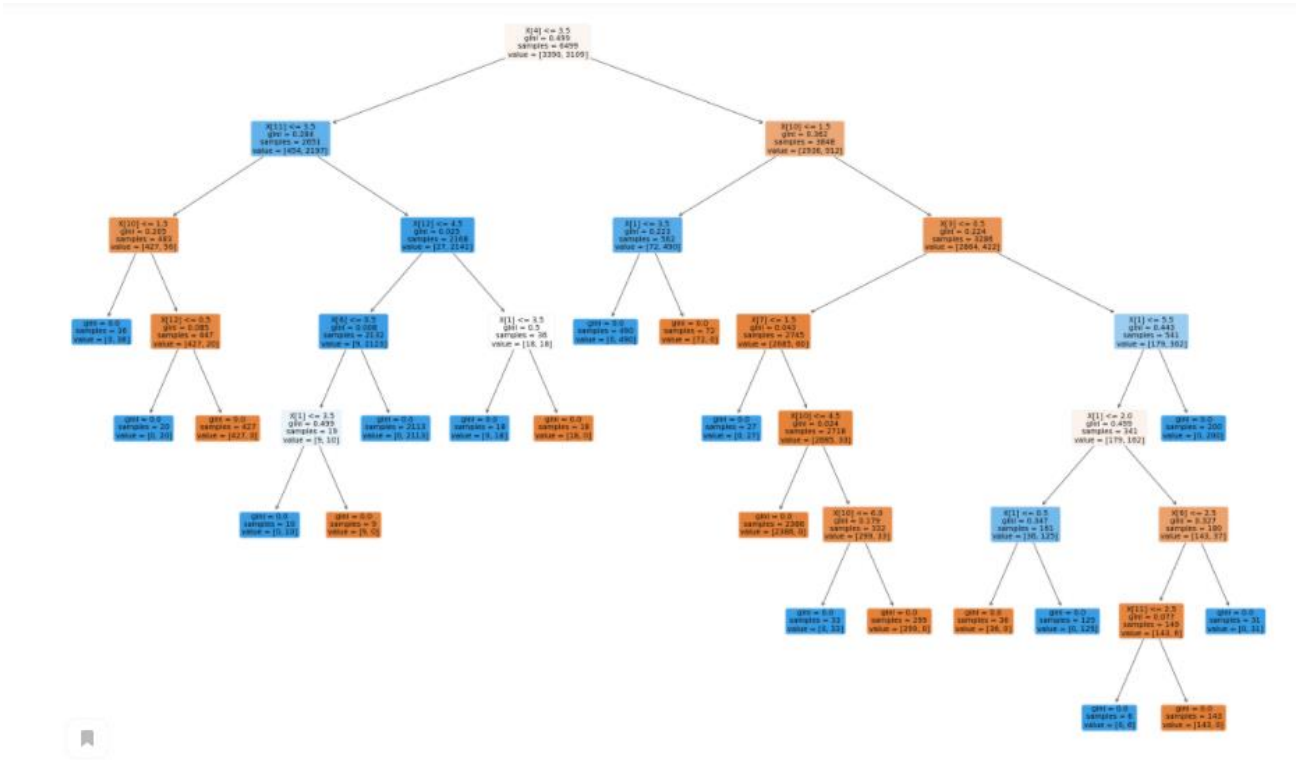
	cap-surface	odor	gill-spacing	gill-size	gill-color	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	ring-type	spore-print-color	population	habitat
3842	0.0	5.0	0.0	0.0	5.0	2.0	2.0	3.0	6.0	4.0	2.0	5.0	0.0
5806	2.0	2.0	0.0	0.0	3.0	0.0	0.0	7.0	7.0	4.0	1.0	4.0	5.0
721	2.0	0.0	0.0	0.0	4.0	2.0	2.0	7.0	7.0	4.0	2.0	2.0	1.0
833	3.0	3.0	0.0	0.0	10.0	2.0	3.0	7.0	7.0	4.0	3.0	3.0	1.0
508	0.0	0.0	1.0	1.0	5.0	2.0	2.0	7.0	7.0	4.0	3.0	4.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
7679	3.0	7.0	0.0	1.0	0.0	1.0	1.0	7.0	7.0	0.0	7.0	4.0	4.0
5741	2.0	5.0	0.0	0.0	10.0	2.0	2.0	7.0	7.0	0.0	7.0	1.0	6.0
640	3.0	3.0	0.0	0.0	10.0	2.0	2.0	7.0	7.0	4.0	2.0	2.0	3.0
5226	3.0	2.0	0.0	1.0	0.0	1.0	2.0	6.0	7.0	0.0	7.0	4.0	4.0
4763	3.0	2.0	0.0	0.0	2.0	1.0	1.0	0.0	6.0	2.0	1.0	4.0	0.0

1625 rows × 13 columns

Рисунок 15 – Деление выборки на обучающую и тестовую

Scikit-Learn также позволяет нам визуализировать наше дерево. Для этого есть несколько настриваемых опций, которые помогут визуализировать узлы принятия решений и разбить изученную модель, что очень полезно для понимания того, как она работает.

Обучим предсказывать каждый класс по сравнению с другим и выведем на экран дерево решений и текстовый отчет, показывающий основные показатели классификации. Для этого подаем тренировочные данные на вход классификатору DecisionTreeClassifier, предсказываем данные с помощью функции predict и выводим показатели оценки классификатора (рисунок 16).



DecisionTreeClassifier()

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	818
1.0	1.00	1.00	1.00	807
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
weighted avg	1.00	1.00	1.00	1625

Рисунок 16 – Дерево решений и текстовый отчет

Общий показатель успеха (Overall Success Rate, OSR), или просто точность (Accuracy), — это число правильно классифицированных наблюдений, отнесенное к общему числу наблюдений. В нашем случае точность равна 1, что является максимальным значением.

Ошибки классификатора удобно представлять в графической форме в виде матрицы несоответствий (Confusion Matrix), таблицы сопряженности — матрицы, в которой для каждого класса наблюдений приводятся результаты отнесения наблюдений к тому или иному классу. Английское название (confusion — путаница) возникло потому, что матрица позволяет видеть, путает ли классификатор классы. Столбцы матрицы соответствуют предсказанным классам, а строки — фактическим классам.

В бинарной классификации (когда все объекты разделяются на два класса) каждое отдельное предсказание может иметь четыре исхода:

– истинноположительный (True Positive, Tp);

– истинноотрицательный (True Negative, Tn);

– ложноположительный (False Positive, Fp);

– ложноотрицательный (False Negative, Fn).

Пусть в качестве положительного исхода выбрано значение Да, а в качестве отрицательного — Нет. Истинноположительным исход будет, когда фактический класс данного примера Да и модель на выходе выдаст Да. Истинноотрицательным исход будет, когда фактический класс наблюдения Нет и модель выдаст Нет. Ложноположительное значение имеет место, когда класс наблюдения Нет, а модель для него сформирует выход Да. При ложноотрицательном выходе целевая переменная принимает значение Да, а на выходе модель выдаст Нет.

Выведем на экран матрицу несоответствий (рисунок 17).

```
# Визуализация матрицы неточности
def tree_matrix():
    dtc_matrix = confusion_matrix(y_test, dtc_predicted)

    ax= plt.subplot()
    sns.heatmap(pd.DataFrame(dtc_matrix), annot=True,cmap="Reds" , fmt='g')
    plt.title('Confusion matrix Decision Tree', y=1.1)
    ax.set_xlabel('Predicted ');ax.set_ylabel('True');
tree_matrix()
```

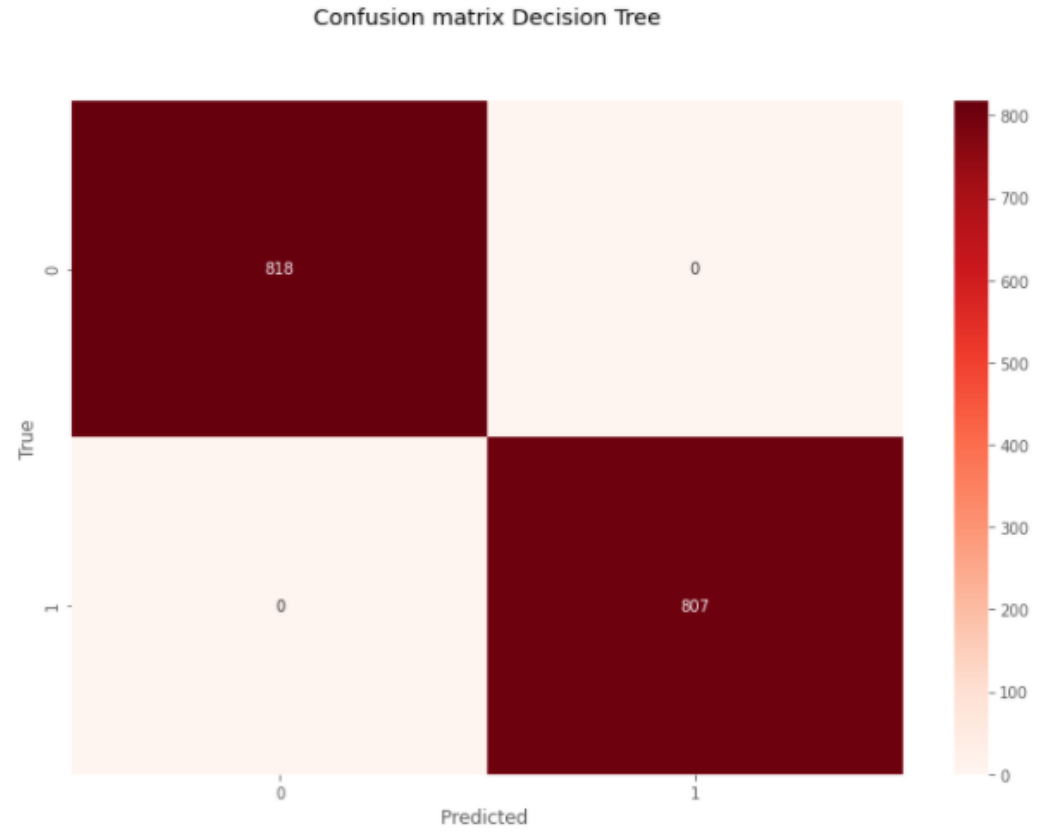


Рисунок 17 – Матрица несоответствий

Матрица несоответствий показывает, что метод машинного обучения определил истинноположительные и истинноотрицательные значения в количестве 818 и 807 соответственно.

Для оценки качества классификатора используется ROC-анализ. Каждая точка ROC-кривой соответствует определенному значению порога отсечения. При этом оптимальным будет значение, соответствующее точке ROC-кривой с координатами, максимально близкими к (0; 100), для которой и чувствительность, и специфичность равны 100 %, то есть как положительные, так и отрицательные примеры распознаны правильно.

Площадь под ROC кривой (AUC — Area Under ROC Curve) говорит о прогностической силе модели, причем AUC=1 соответствует идеальному классификатору. Для реальных классификаторов площадь под ROC-кривой больше 0,7–0,8, соответствует достаточно высокой точности классификации.

ROC-анализ представлен на рисунке 18.

```
def tree_roc():
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(y_test, dtc_predicted)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
tree_roc()
```

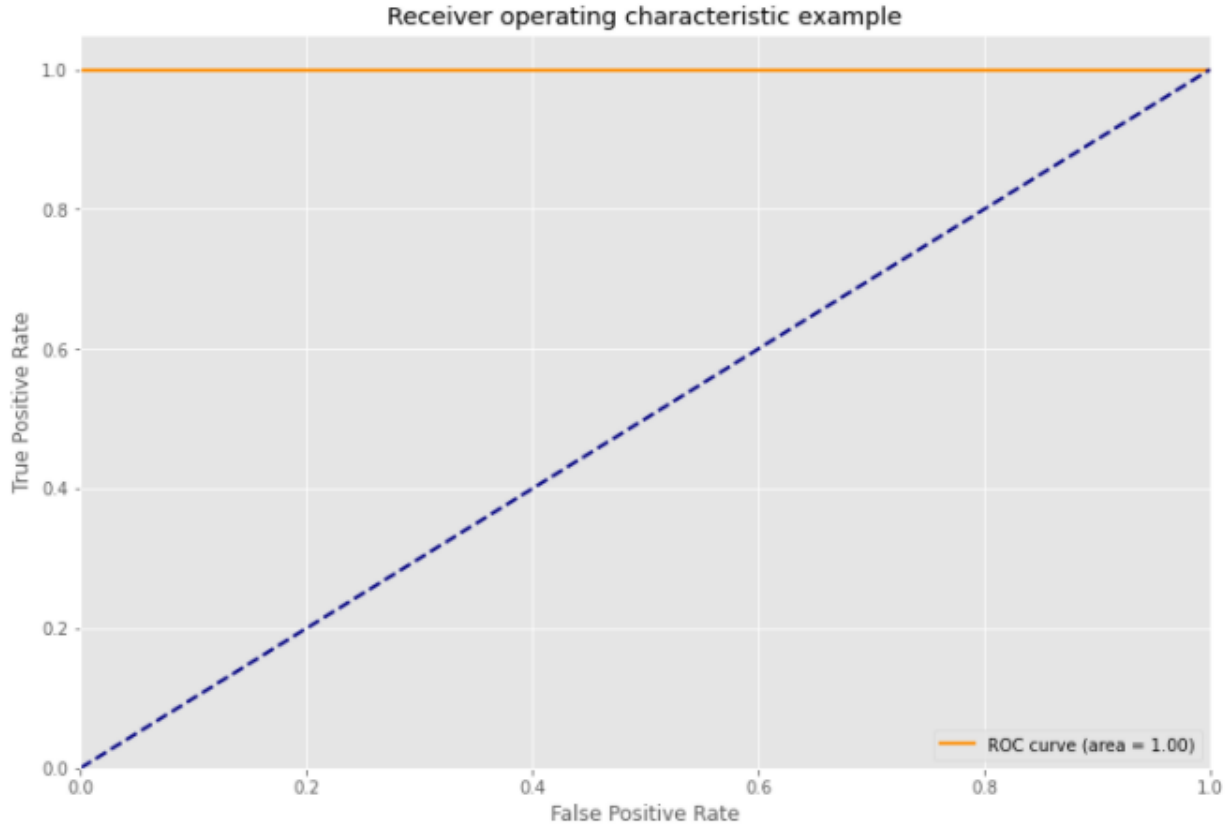


Рисунок 18 – ROC-анализ

На основе проведенного выше исследования можно сделать вывод, что точность метода Деревья решений составляет 100%, это говорит о том, что данный метод хорошо определяет классы. Для оценки качества классификатора использовался ROC-анализ. Площадь под ROC кривой (AUC — Area Under ROC Curve) говорит о прогностической силе модели, причем AUC=1 соответствует идеальному классификатору. Для реальных классификаторов площадь под ROC-

кривой больше 0,7–0,8, соответствует достаточно высокой точности классификации. В данном примере AUC=1, это говорит о том, что данный алгоритм имеет очень высокую точность.

2.3 Использование метода машинного обучения «Логистическая регрессия»

Логистическая регрессия - это разновидность множественной регрессии, общее назначение которой состоит в анализе связи между несколькими независимыми переменными (называемыми также регрессорами или предикторами) и зависимой переменной. Бинарная логистическая регрессия применяется, если зависимая переменная является бинарной (т.е. может принимать только два значения). Иными словами, с помощью логистической регрессии можно оценивать вероятность того, что событие наступит для конкретного испытуемого (больной/здоровый, возврат кредита/дефолт и т.д.).

Перед применением метода «Деревья решений» мы уже поделили выборку на тестовую и обучающую. Поэтому еще раз этого делать не будем, перейдем сразу к применению метода «Логистическая регрессия».

Подаем тренировочные данные классификатору LogisticRegression, предсказываем данные с помощью функции predict и выводим показатели оценки классификатора (рисунок 19).

```
def log_reg():
    # logistic regression

    lg_model = LogisticRegression().fit(x_train, y_train)

    lg_predicted = lg_model.predict(x_test)

    # доля правильных ответов алгоритма: Точность = (истинное положительное + истинно отрицательное значение) / всего
    acc_lg = accuracy_score(y_test, lg_predicted)

    # Оценка f1: оценку F1 можно интерпретировать как средневзвешенное значение точности и запоминания, где оценка F1 достигает
    f1_lg = f1_score(y_test, lg_predicted)

    # долю объектов, названных классификатором положительными и при этом действительно являющимися положительными: если прогноз
    precision_lg = precision_score(y_test, lg_predicted)

    # показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм: истинно полож
    recall_lg = recall_score(y_test, lg_predicted)

    print(f"The accuracy score for LogReg is: {round(acc_lg,3)*100}%")
    print(f"The f1 score for LogReg is: {round(f1_lg,3)*100}%")
    print(f"The precision score for LogReg is: {round(precision_lg,3)*100}%")
    print(f"The recall score for LogReg is: {round(recall_lg,3)*100}%")

    return lg_predicted, acc_lg, f1_lg, precision_lg, recall_lg

lg_predicted, acc_lg, f1_lg, precision_lg, recall_lg = log_reg()
```

```
The accuracy score for LogReg is: 93.89999999999999%
The f1 score for LogReg is: 93.8%
The precision score for LogReg is: 95.0%
The recall score for LogReg is: 92.60000000000001%
```

Рисунок 19 – Показатели оценки классификатора «Логистическая регрессия»

Мы можем увидеть, что точность равна 93,9%, средневзвешенное значение точности и запоминания равна 93,8%, доля объектов, названных классификатором положительными и при этом действительно являющимися положительными равна 95%, recall, который показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм равен 92,6%.

Выведем на экран матрицу несоответствий (рисунок 20).

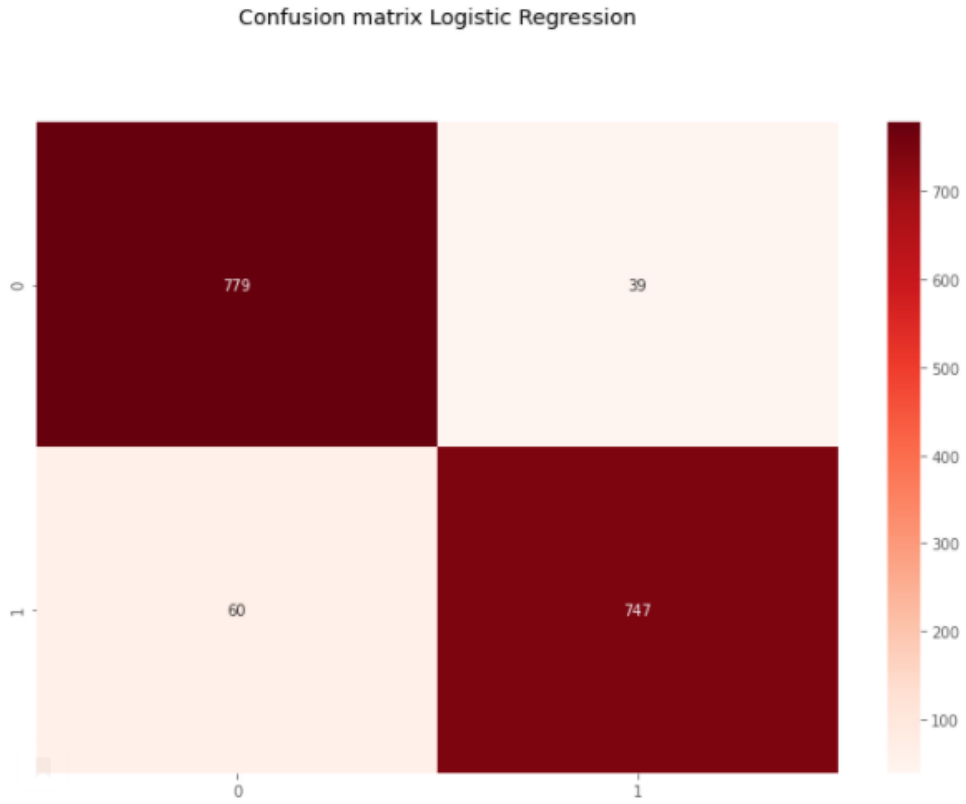


Рисунок 20 – Матрица несоответствий

Матрица несоответствий показывает, что метод машинного обучения определил ложноположительные и ложноотрицательные значения в количестве 39 и 60 соответственно.

ROC-анализ представлен на рисунке 21.

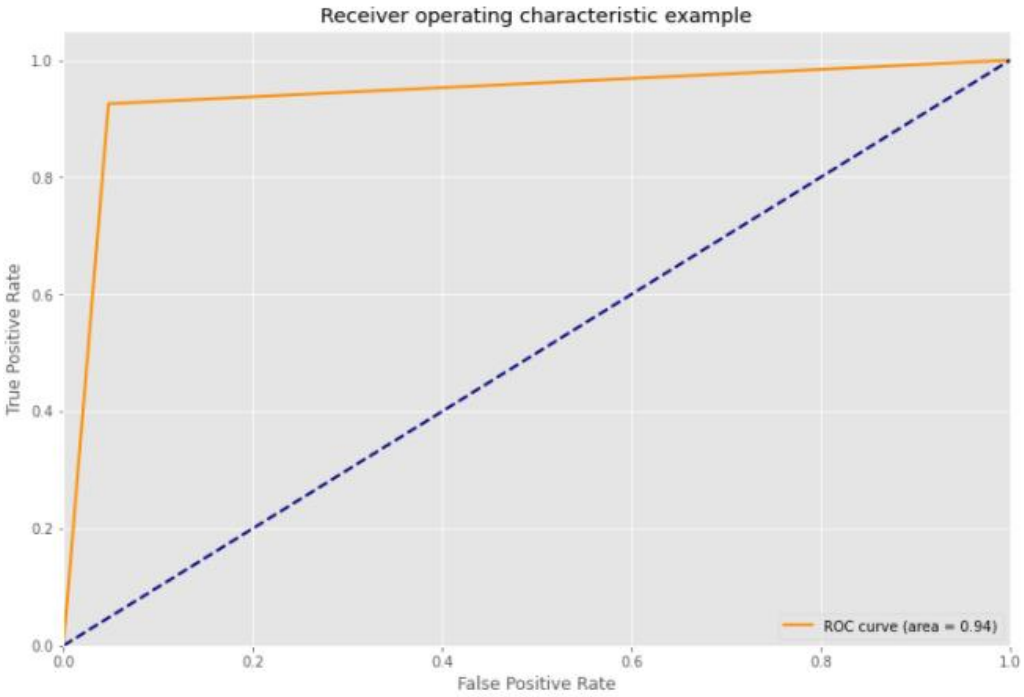


Рисунок 21– ROC-анализ

На основе проведенного выше исследования можно сделать вывод, что точность метода Логистическая регрессия составляет 93,9%, это говорит о том, что данный метод неплохо определяет классы. Для оценки качества классификатора использовался ROC- анализ. Площадь под ROC кривой (AUC — Area Under ROC Curve) говорит о прогностической силе модели, причем AUC=1 соответствует идеальному классификатору. Для реальных классификаторов площадь под ROC-кривой больше 0,7–0,8, соответствует достаточно высокой точности классификации. В данном примере AUC=0,94, это говорит о том, что данный алгоритм имеет высокую точность. Но в сравнении с деревом решений этот метод проигрывает, так как выявил ложноположительные и ложноотрицательные значения.

2.4 Использование метода машинного обучения «К-ближайших соседей»

Метод kNN (k-Nearest Neighbors) часто используется как составная часть более сложного алгоритма классификации. Например, его оценку можно использовать как признак для объекта. А иногда, простой kNN на хорошо подобранных признаках дает отличное качество. При грамотной настройке параметров (в основном - метрики) алгоритм дает зачастую хорошее качество в задачах регрессии.

Подаем тренировочные данные классификатору KNeighborsClassifier, предсказываем данные с помощью функции predict и выводим показатели оценки классификатора (рисунок 22).

```
def k_means():
    km_model = KNeighborsClassifier()
    a_km=km_model.fit(x_train,y_train)
    print(a_km)
    km_predicted = km_model.predict(x_test)
    km_metr=metrics.classification_report(y_test,km_predicted)
    print(km_metr)
    print(metrics.confusion_matrix(y_test, km_predicted))
    return km_model, a_km, km_predicted, km_metr
km_model, a_km, km_predicted, km_metr =k_means()
```

```
KNeighborsClassifier()
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       818
         1.0       1.00      1.00      1.00       807

    accuracy                           1.00      1625
   macro avg       1.00      1.00      1.00      1625
weighted avg       1.00      1.00      1.00      1625

[[814   4]
 [  0 807]]
```

Рисунок 22 – Показатели оценки классификатора К-ближайших соседей

Мы можем увидеть, что точность, средневзвешенное значение точности и запоминания, доля объектов, названных классификатором положительными и при этом действительно являющимися положительными, recall, который показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм, равны 1 или 100%.

Выведем на экран матрицу несоответствий (рисунок 23).

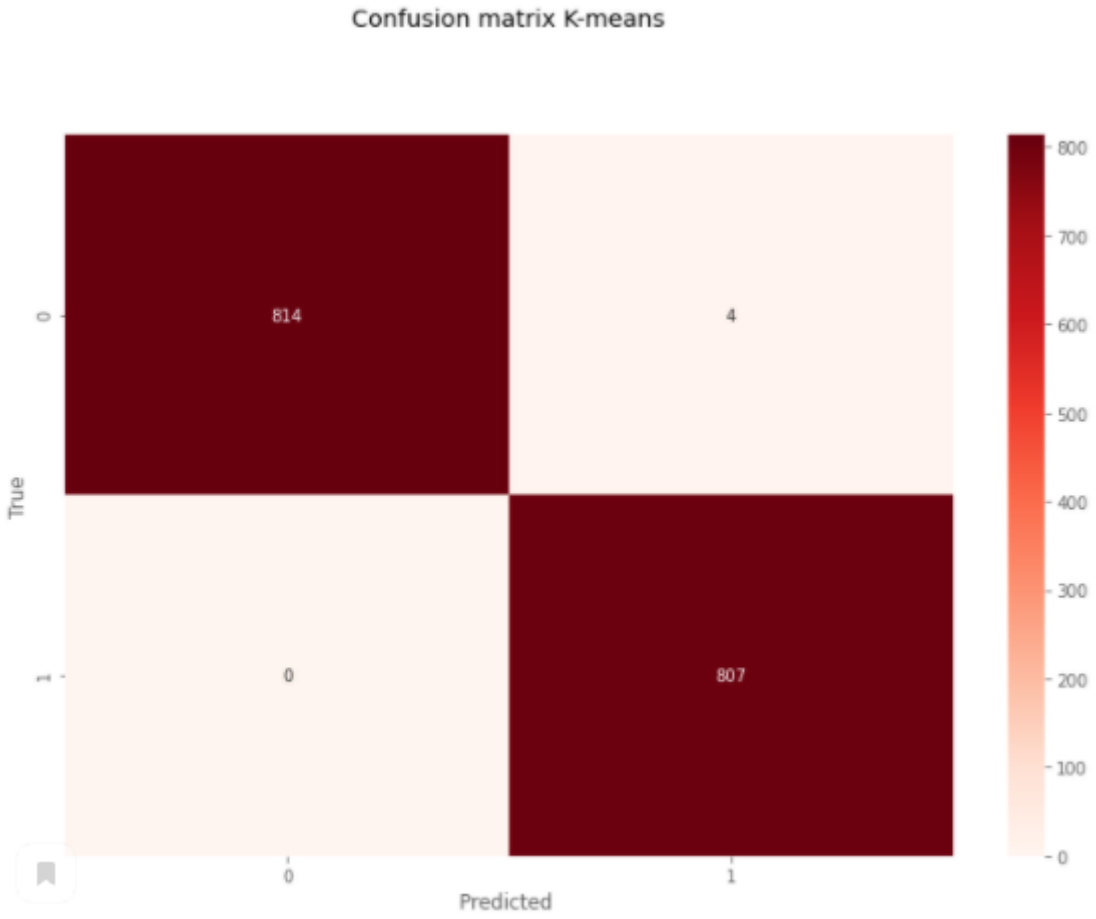


Рисунок 23 – Матрица несоответствий

Матрица несоответствий показывает, что метод машинного обучения определил ложноположительные значения в количестве 4.

ROC-анализ представлен на рисунке 24.

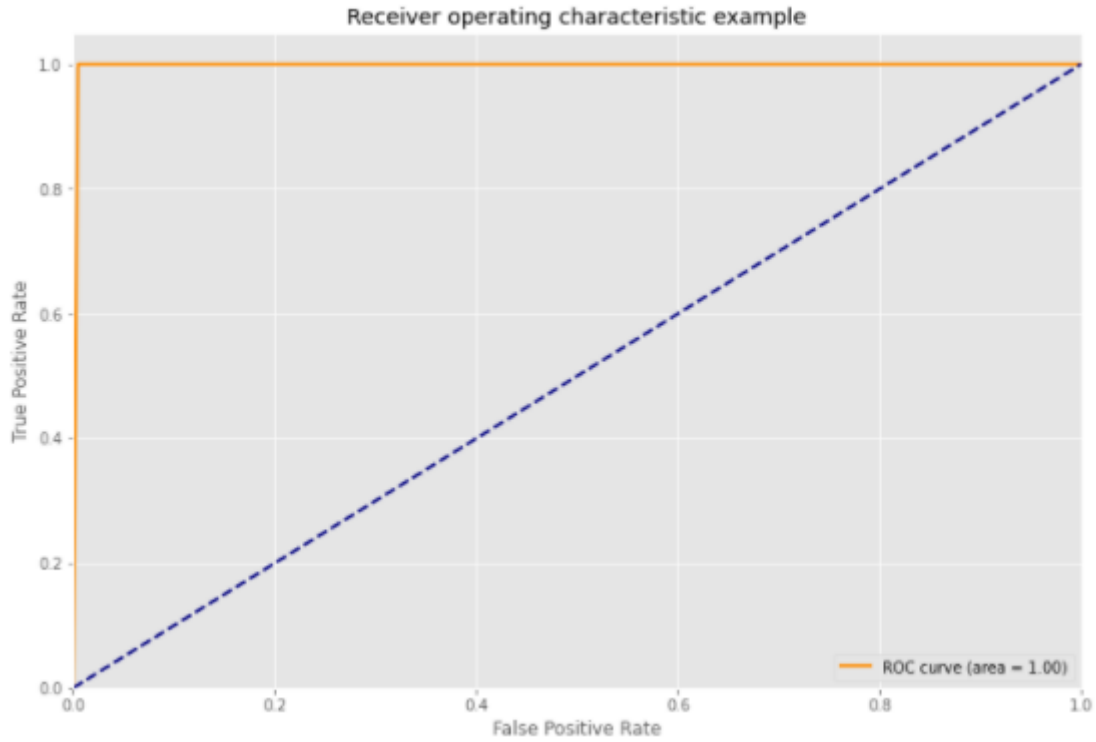


Рисунок 24 - ROC-анализ

На основе проведенного выше исследования можно сделать вывод, что точность метода К-ближайших соседей составляет 100%, это говорит о том, что данный метод очень хорошо определяет классы. Для оценки качества классификатора использовался ROC-анализ. Площадь под ROC кривой (AUC — Area Under ROC Curve) говорит о прогностической силе модели, причем AUC=1 соответствует идеальному классификатору. Для реальных классификаторов площадь под ROC-кривой больше 0,7–0,8, соответствует достаточно высокой точности классификации. В данном примере AUC=1,00, это говорит о том, что данный алгоритм очень точен. Но в сравнении с деревом решений этот метод проигрывает, так как выявил 4 ложноположительных значения.

2.5 Использование метода машинного обучения «Метод опорных векторов»

Метод SVM (Support Vector Machines) является одним из самых известных алгоритмов машинного обучения, применяемых в основном для задачи классификации. Так же как и логистическая регрессия SVM допускает многоклассовую классификацию методом one-vs-all.

Подаем тренировочные данные классификатору SVC, предсказываем данные с помощью функции predict и выводим показатели оценки классификатора (рисунок 25).

```
def vectors():
    svc_model = SVC()
    a_svc=svc_model.fit(x_train,y_train)
    print(a_svc)
    svc_expected = y_test
    svc_predicted = svc_model.predict(x_test)
    svc_metr=metrics.classification_report(svc_expected,svc_predicted)
    print(svc_metr)
    print(metrics.confusion_matrix(svc_expected,svc_predicted))
    return svc_model, a_svc, svc_predicted,svc_expected, svc_metr
svc_model, a_svc, svc_predicted,svc_expected, svc_metr=vectors()
```

```
SVC()
              precision    recall  f1-score   support

         0.0       0.96      0.99      0.97       818
         1.0       0.99      0.96      0.97       807

    accuracy                           0.97      1625
   macro avg       0.97      0.97      0.97      1625
weighted avg       0.97      0.97      0.97      1625

[[809   9]
 [ 34 773]]
```

Рисунок 25 – Показатели оценки метода опорных векторов

Мы можем увидеть, что точность равна 0,97, средневзвешенное значение точности и запоминания равно 0,97.

Выведем на экран матрицу несоответствий (рисунок 26).

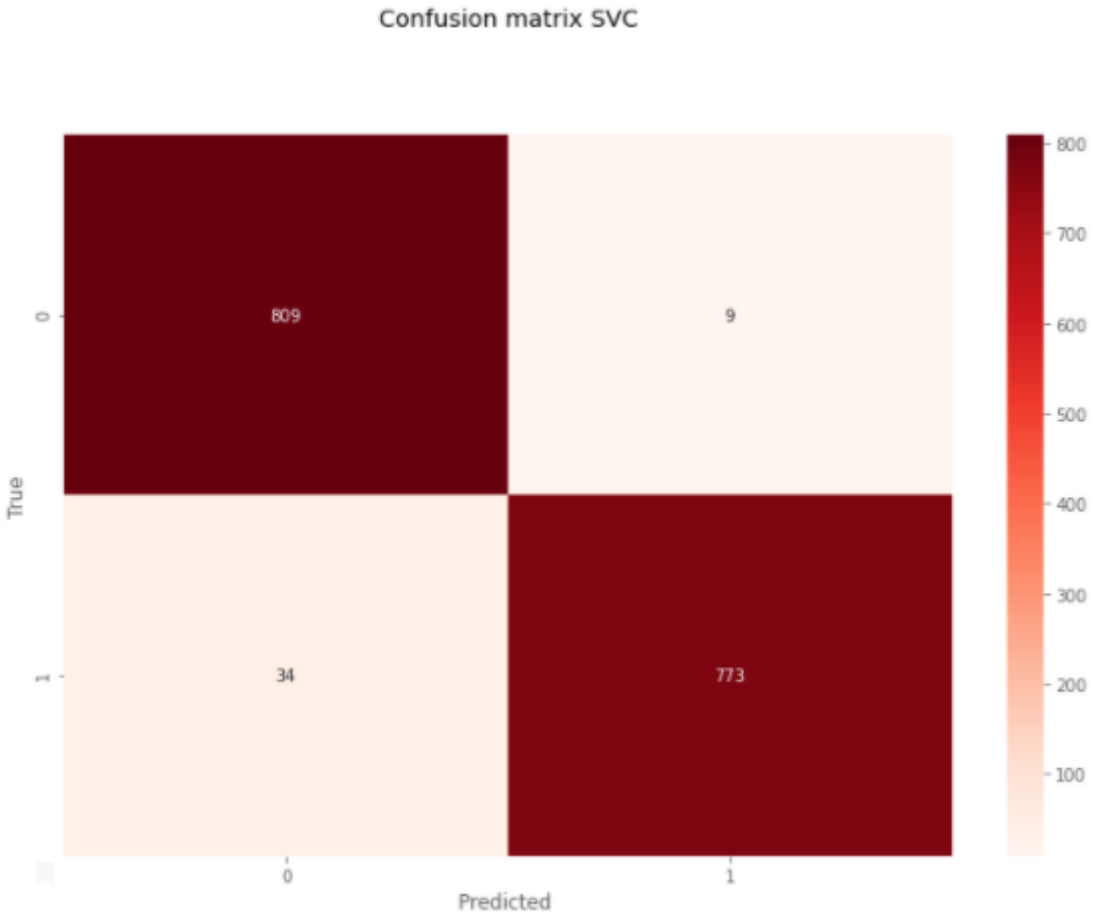


Рисунок 26 – Матрица несоответствий

Матрица несоответствий показывает, что метод машинного обучения определил ложноположительные и ложноотрицательные значения в количестве 9 и 34 соответственно.

ROC-анализ представлен на рисунке 27.

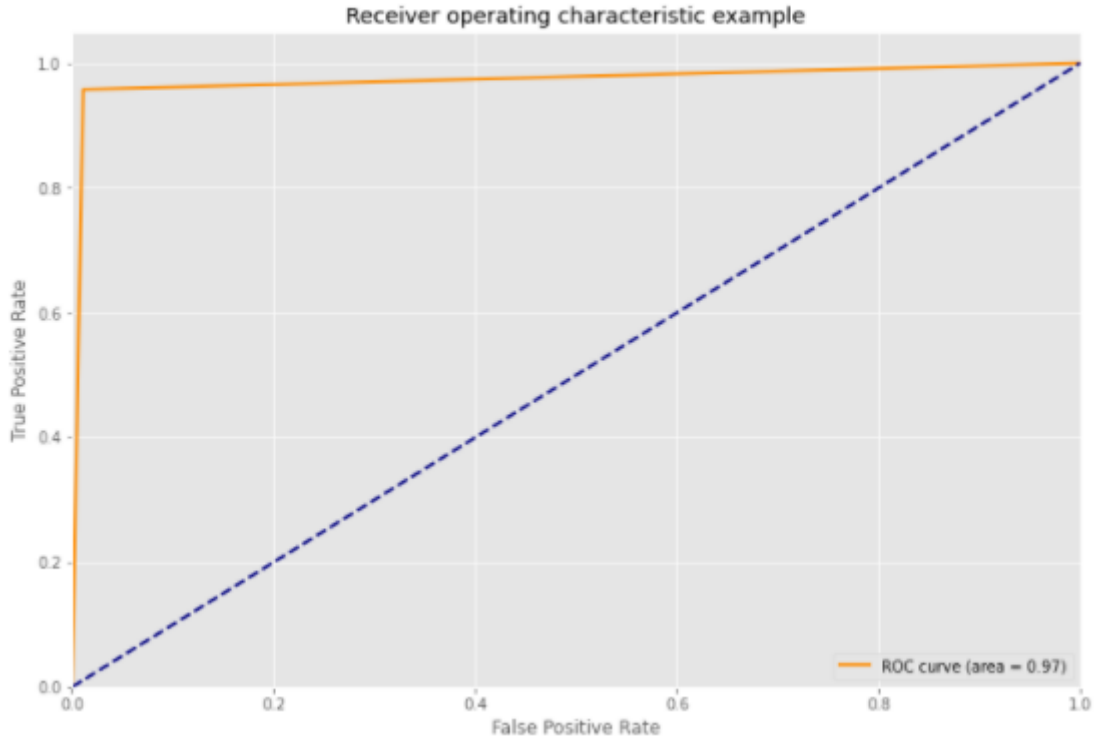


Рисунок 27 - ROC-анализ

На основе проведенного выше исследования можно сделать вывод, что точность метода опорных векторов составляет 97%, это говорит о том что данный метод не плохо определяет классы. Для оценки качества классификатора использовался ROC- анализ. Площадь под ROC кривой (AUC — Area Under ROC Curve) говорит о прогностической силе модели, причем AUC=1 соответствует идеальному классификатору. Для реальных классификаторов площадь под ROC-кривой больше 0,7–0,8, соответствует достаточно высокой точности классификации. В данном примере AUC=0,97, это говорит о том, что данный алгоритм имеет высокую точность. Но этот метод выявил ложноположительные и ложноотрицательные значения. В сравнении с деревом решений этот метод проигрывает.

2.6 Использование метода машинного обучения «Нейронная сеть (Персептрон)»

Искусственные нейронные сети появились на основе знаний о функционировании мозга живых существ. Хотя практически любая книга по

искусственным нейронным сетям начинается с краткого описания работы мозга, наши знания о работе мозга столь ограничены, что можно говорить только об очень грубой аналогии.

Персептрон (англ. perceptron от лат. perceptio — восприятие) — устройство МАРК-1, а также соответствующая ему математическая модель, созданная Фрэнком Розенблаттом с целью построения модели мозга. Под «моделью мозга» понимается любая теоретическая система, которая стремится объяснить физиологические функции мозга с помощью известных законов физики и математики, а также известных фактов нейроанатомии и нейрофизиологии.

Классический метод обучения персептрона — это обучение с коррекцией ошибки. Представляет собой такой метод обучения, при котором вес связи не изменяется до тех пор, пока текущая реакция персептрона остается правильной. При появлении неправильной реакции вес изменяется на единицу, а знак (+/-) определяется противоположным от знака ошибки.

Настраиваем классификатор Perceptron с параметрами max_iter=1000, eta0=0.1, random_state=0. Подаем тренировочные данные классификатору, предсказываем данные с помощью функции predict и выводим показатели оценки классификатора (рисунок 28).

```
def neuro():
    #инициализация
    ppn = Perceptron(max_iter=1000, eta0=0.1, random_state=0)

    #настройка
    a_ppn = ppn.fit(x_train, y_train)

    ppn_predicted = ppn.predict(x_test)

    print('Неверно классифицированные образцы: %d' % (y_test != ppn_predicted).s
    print('Точность: %.2f' % accuracy_score(y_test, ppn_predicted))
    return ppn, a_ppn, ppn_predicted

ppn, a_ppn, ppn_predicted = neuro()
```

```
Неверно классифицированные образцы: 227
Точность: 0.86
```

Рисунок 28 – Показатели оценки метода «Нейронные сети (Персептрон)»

Мы можем увидеть, что точность равна 0,86, классификатор неверно определил 227 образцов.

Выведем на экран матрицу несоответствий (рисунок 29).

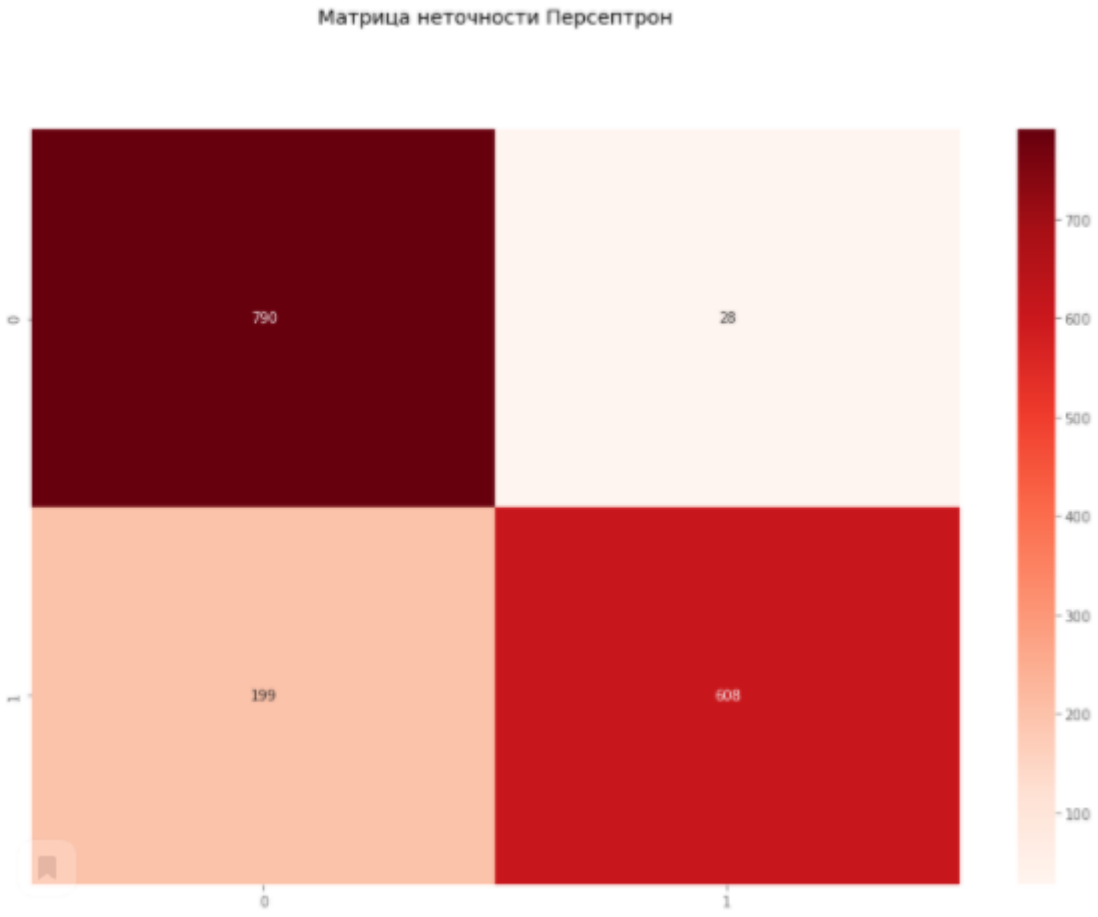


Рисунок 29 – Матрица несоответствий

Матрица несоответствий показывает, что метод машинного обучения определил ложноположительные и ложноотрицательные значения в количестве 28 и 199 соответственно.

ROC-анализ представлен на рисунке 30.

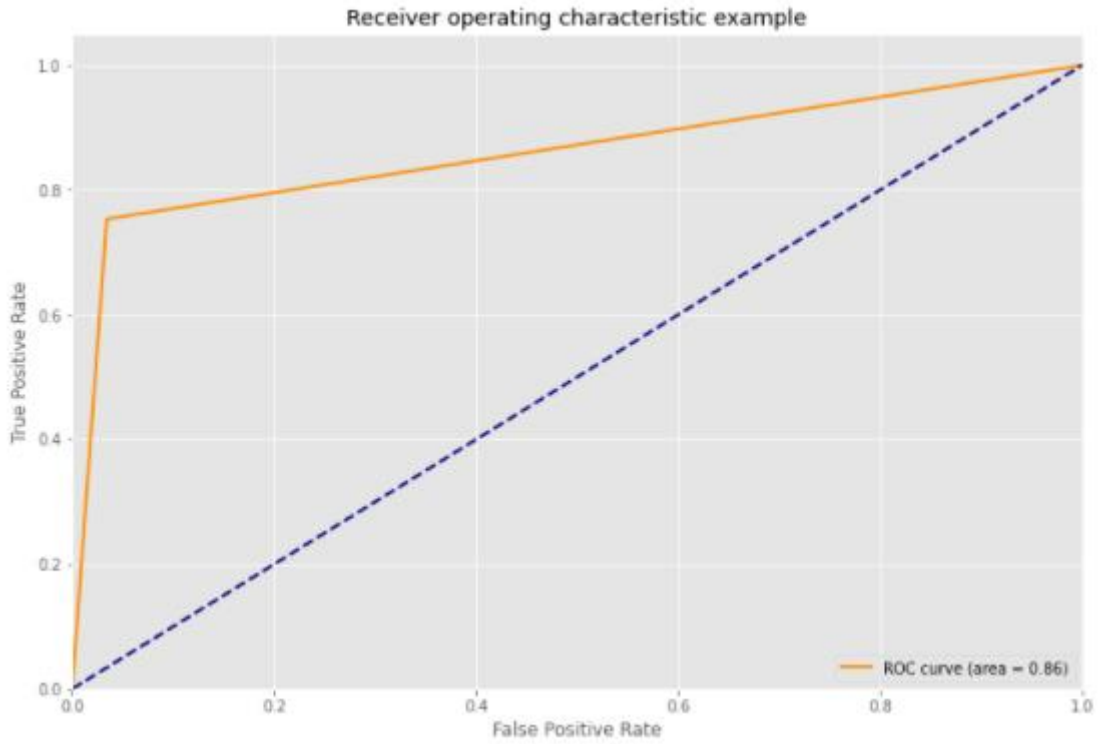


Рисунок 30 - ROC-анализ

На основе проведенного выше исследования можно сделать вывод, что точность метода Нейронная сеть (Персептрон) составляет 86%, это говорит о том что данный метод хорошо определяет классы. Для оценки качества классификатора использовался ROC- анализ. Площадь под ROC кривой (AUC — Area Under ROC Curve) говорит о прогностической силе модели, причем AUC=1 соответствует идеальному классификатору. Для реальных классификаторов площадь под ROC-кривой больше 0,7–0,8, соответствует достаточно высокой точности классификации. В данном примере AUC=0,86 ,это говорит о том, что данный алгоритм точен. По сравнению с деревом решений, данный метод уступает в точности и количестве выявленных ложноположительных и ложноотрицательных значений.

2.7 Использование метода машинного обучения «Многослойная нейронная сеть»

Многослойная нейронная сеть (англ. Multilayer neural network) - нейронная сеть, состоящая из входного, выходного и расположенного (ых) между ними одного (нескольких) скрытых слоев нейронов.

Помимо входного и выходного слоев эти нейронные сети содержат промежуточные, *скрытые слои*. Такие сети обладают гораздо большими возможностями, чем однослойные нейронные сети, однако методы обучения нейронов скрытого слоя были разработаны относительно недавно.

Работу скрытых слоев нейронов можно сравнить с работой большого завода. Продукт (выходной сигнал) на заводе собирается по стадиям на станках. После каждого станка получается какой-то промежуточный результат. Скрытые слои тоже преобразуют входные сигналы в некоторые промежуточные результаты.

Настраиваем классификатор MLPClassifier с параметрами activation='relu', solver='lbfgs', alpha=1e-5, max_iter=1000, hidden_layer_sizes=(5,), random_state=1. Данные параметры были выбраны, экспериментальны путем, с целью выявления наилучших результатов. Количество нейронов в 1 слое посчитали как квадратный корень из произведения количества входных параметров и количества выходов ($\sqrt{(13 * 2)}$=5) Подаем тренировочные данные классификатору, предсказываем данные с помощью функции predict и выводим показатели оценки классификатора (рисунок 31).

```
def ms_neuro():
    mlp = MLPClassifier(activation='relu', solver='lbfgs', alpha=1e-5, max_iter=
    a_mlp=mlp.fit(x_train,y_train)
    mlp_pred = mlp.predict(x_test)
    print('Неверно классифицированные образцы: %d' % (y_test != mlp_pred).sum())
    print('Точность: %.2f' % accuracy_score(y_test, mlp_pred))
    return mlp, a_mlp, mlp_pred
mlp, a_mlp, mlp_pred = ms_neuro()
```

```
Неверно классифицированные образцы: 2
Точность: 1.00
```

Рисунок 31 – Показатели оценки метода «Многослойные нейронные сети»

Мы можем увидеть, что точность равна 1, классификатор неверно определил 2 образца.

Выведем на экран матрицу несоответствий (рисунок 32).

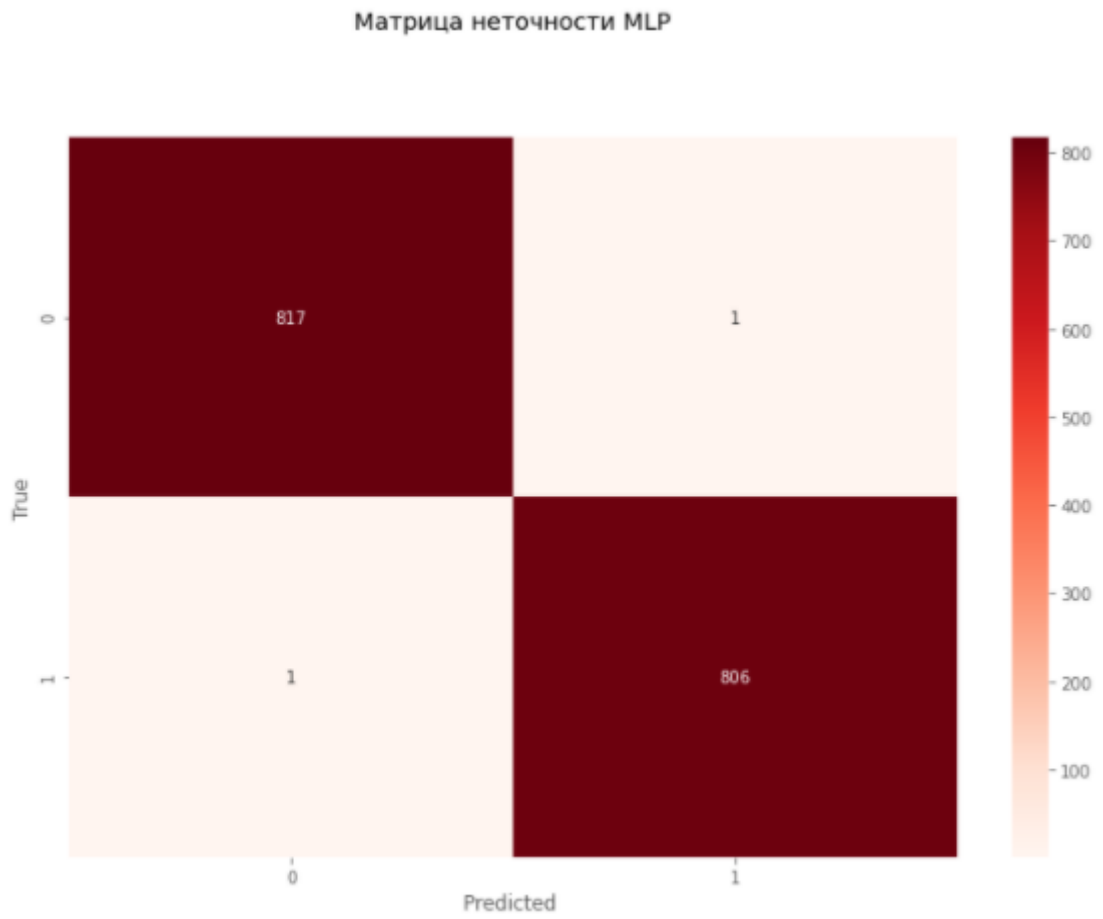


Рисунок 31 – Матрица несоответствий

Матрица несоответствий показывает, что метод машинного обучения определил ложноположительные и ложноотрицательные значения в количестве 1 и 1 соответственно.

ROC-анализ представлен на рисунке 32.

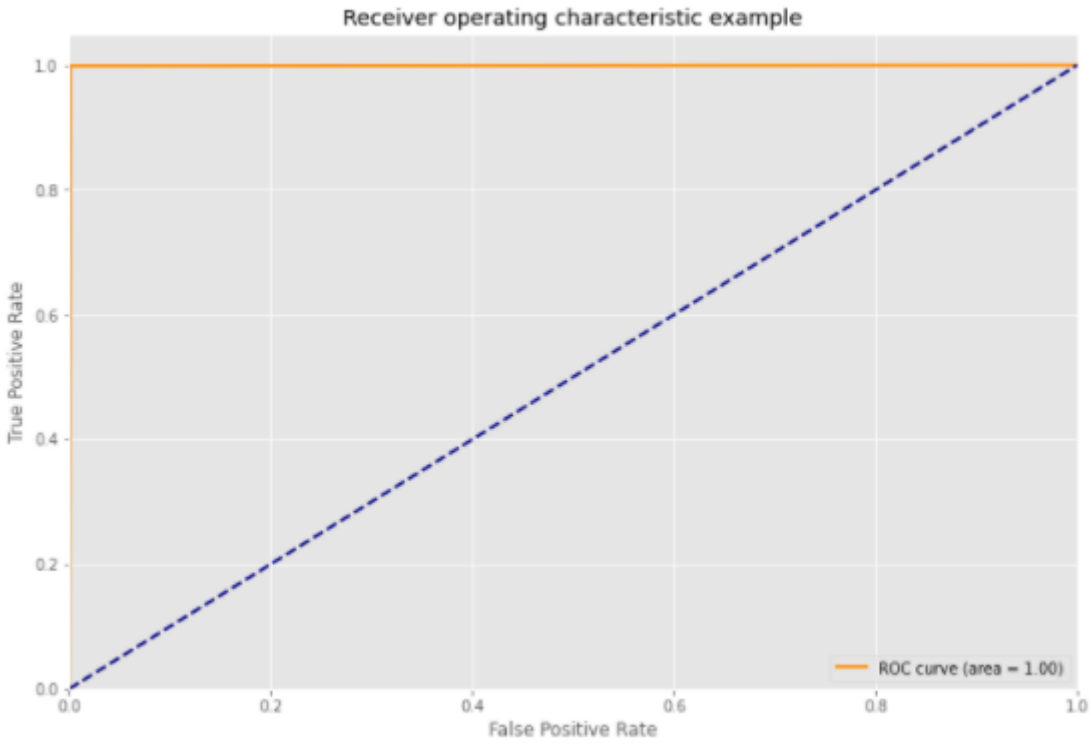


Рисунок 32 - ROC-анализ

На основе проведенного выше исследования можно сделать вывод, что точность метода Многослойная нейроная сеть с одним скрытым слоем и пятью нейронами в скрытом слое, функцией активации logistic, оптимизацией веса lbfgs, составляет 100%, это говорит о том, что данный метод достаточно хорошо определяет классы. Для оценки качества классификатора использовался ROC-анализ. Площадь под ROC кривой (AUC — Area Under ROC Curve) говорит о прогностической силе модели, причем AUC=1 соответствует идеальному классификатору. Для реальных классификаторов площадь под ROC-кривой больше 0.7–0.8, соответствует достаточно высокой точности классификации. В данном примере AUC=1.00, это говорит о том, что данный алгоритм достаточно точен. В сравнении с деревом решений этот метод проигрывает в количестве выявленных ложноположительных и ложноотрицательных значений.

2.8 Продукт бот

После использования различных методов машинного обучения был создан продукт бот, который выводит метод анализа, выбранный пользователем. На экран выводится оценка метода, матрица несоответствий и ROC-анализ.

Для того, чтобы начать работу с продукт ботом, необходимо его запустить нажав комбинацию клавиш Ctrl+Enter или кнопку Run на панели. После запуска пользователю необходимо написать цифру метода, который он бы хотел увидеть и нажать Enter. При необходимости он может завершить работу, выбрав цифру 7. Список методов с цифрами:

1. Дерево решений
2. Логистическая регрессия
3. К-ближайших соседей
4. Метод опорных векторов
5. Нейронная сеть (персептрон)
6. Многослойная нейронная сеть
7. Выход

После выбора, бот выводит на экран метод, показатели точности, матрицу несоответствий и ROC-анализ. Далее пользователь может выбрать другой метод, введя нужную цифру, или завершить работу (цифра 7).

Для примера был выбран метод «Деревья решений». Результат работы продукт бота приведен на рисунках 33-35.

```
Выберите метод для анализа:
1. Дерево решений
2. Логистическая регрессия
3. К-ближайших соседей
4. Метод опорных векторов
5. Нейронная сеть (персептрон)
6. Многослойная нейронная сеть
7. Выход

Ответ: 1
Дерево решений
```

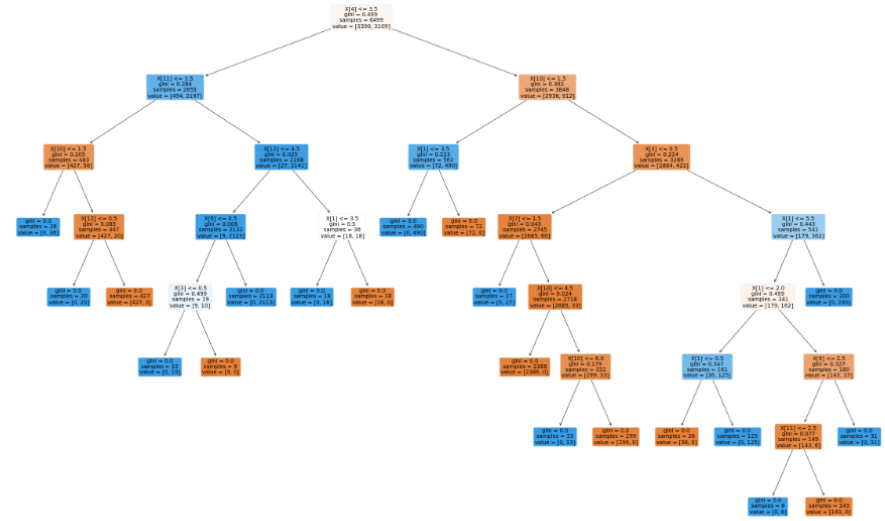


Рисунок 33 – Пример работы продукт бота

```
DecisionTreeClassifier()
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       818
         1.0       1.00      1.00      1.00       807

    accuracy                           1.00      1625
   macro avg       1.00      1.00      1.00      1625
weighted avg       1.00      1.00      1.00      1625

[[818   0]
 [  0 807]]
```

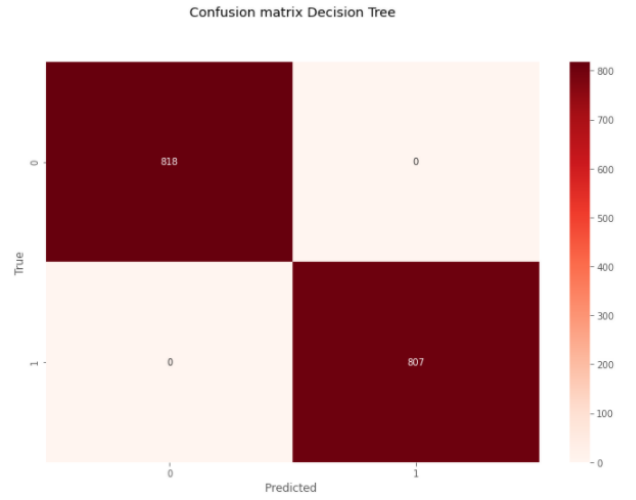
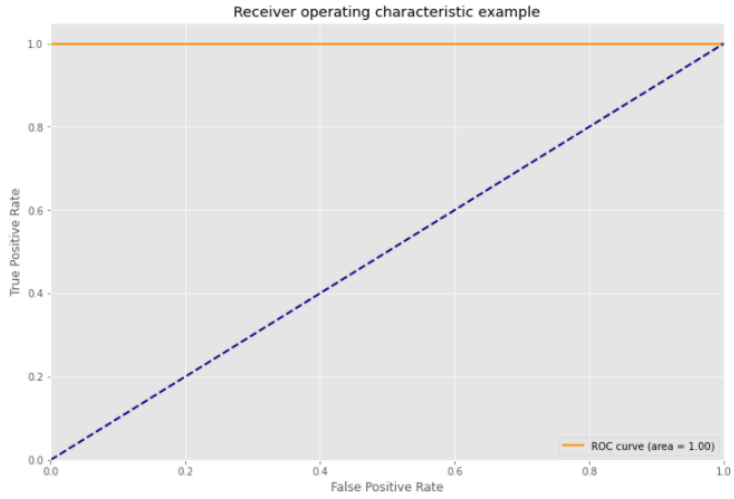


Рисунок 34 – Пример работы продукт бота (продолжение)




```
Выберите метод для анализа:
1. Дерево решений
2. Логистическая регрессия
3. K-ближайших соседей
4. Метод опорных векторов
5. Нейронная сеть (персептрон)
6. Многослойная нейронная сеть
7. Выход

Ответ: 7
```

Рисунок 35 – Пример работы продукт бота (продолжение)

Заключение

В результате проделанной работы был создан продукт бот для классификации гри бов. Были изучены и использованы такие методы машинного обучения как: деревья решений, логистическая регрессия, К-ближайших соседей, опорных векторов, нейро нная сеть (персептрон), многослойная нейронная сеть.

Анализ данных и создание продукт бота проводились с использованием программного средства Jupyter Notebook.

При анализе данных методов было выявлено, что наиболее точную классификацию дал метод «Дерево решений».

ТЕКСТ ПРОГРАММЫ

Приложение А

```
#!/usr/bin/env python
# coding: utf-8
# In[1]:
import pandas as pd

import numpy as np
import seaborn as sns
import scipy.stats as sts
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.svm import SVC
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier

from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc

from sklearn import preprocessing
from sklearn.preprocessing import label_binarize
```

```
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import matplotlib
plt.style.use('ggplot')
from matplotlib.pyplot import figure

get_ipython().run_line_magic('matplotlib', 'inline')
matplotlib.rcParams['figure.figsize'] = (12,8)

pd.options.mode.chained_assignment = None
# In[2]:
f = pd.read_csv('mushrooms.csv', delimiter=r'\s*,\s*', engine='python')
f
# In[3]:
# выведем размер и тип данных
print(f.shape)
print(f.dtypes)
# In[4]:
# выведем статистику
f.describe()
# In[5]:
a=np.array(f)
print(a)
# separate the data from the target attributes
enc = preprocessing.OrdinalEncoder()
enc.fit(a)
v=enc.transform(a)
v
# In[ ]:
import pandas as pd
pd.DataFrame(v).to_csv("mushrooms1.csv")
# In[6]:
df = pd.read_csv('mushrooms1.csv', delimiter=r'\s*,\s*', engine='python')
```

```
del df['Unnamed: 0']
df
# In[7]:
plt.figure(figsize=(8,6))
plt.scatter(range(df.shape[0]), np.sort(df['class'].values))
plt.xlabel('Количество', fontsize=12)
plt.ylabel('Класс', fontsize=12)
plt.show()
# In[8]:
pd.options.display.max_columns = None
f
# In[9]:
f['stalk-root']
# In[10]:
n = f
n = n.replace('?', np.NaN)
n
# In[11]:
for col in n.columns:
    pct_missing = np.mean(n[col].isnull())
    k=round(pct_missing*100)
    print('{} - {}%'.format(col, k))
# In[12]:
sns.heatmap(df.corr(),annot=True,cmap='RdYlGn',linewidths=0.2)
fig=plt.gcf()
fig.set_size_inches(12,12)
plt.show()
# In[13]:
del df['stalk-root']
del df['bruises']
del df['veil-color']
# In[14]:
# проверка распределений с помощью гистограмм
```

```
fig = plt.figure(figsize = (15,20))
ax = fig.gca()
df.hist(ax = ax)
# In[15]:
columns = list(df)
for col in columns:
    kurt=sts.kurtosis(df[col], axis=0, fisher=True, bias=True)
    sk= sts.skew(df[col], axis=0, bias=True)
    print('Показатели асимметрия и эксцесс для: {}'.format(col))
    print(kurt,sk)
# In[16]:
num_rows = len(df.index)
low_information_cols = []

for col in df.columns:
    cnts = df[col].value_counts(dropna=False)
    top_pct = (cnts/num_rows).iloc[0]
    print(top_pct*100)

    if top_pct > 0.95:
        low_information_cols.append(col)
        print('{0}: {1:.5f}%'.format(col, top_pct*100))
        print(cnts)
        print()
# In[17]:
del df['veil-type']
# In[19]:
# Определяем показатели, имеющие наибольшее значение для выходной переменной Класс
# разделяем независимые и зависимые переменные
X=df.iloc[:,1:19] # независимые
y=df.iloc[:,0] # цель
print(X)
# In[21]:
```

```
# примените класс SelectKBest, чтобы извлечь лучшие показатели
bestfeatures = SelectKBest(score_func=chi2, k='all')
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

#объединим два фрейма данных для лучшей визуализации
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Показатель','Счет']
print(featureScores.nlargest(19,'Счет'))
# In[22]:
# визуализация отбора показателей
plt.figure(figsize=(20,5))
sns.barplot(x='Показатель', y='Счет', data=featureScores, palette = "GnBu_d")
plt.box(False)
plt.title('Важность показателей', fontsize=16)
plt.xlabel('\n Показатели', fontsize=14)
plt.ylabel('Важность \n', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
# In[23]:
del df['stalk-shape']
del df['ring-number']
del df['cap-shape']
del df['cap-color']
del df['gill-attachment']
# In[24]:
df
# In[26]:
# определяем x и y
y = df['class']
x = df.drop(['class'], axis = 1)
```

```
# деление на обучающую и тестовую выборки: 80 % - 20 %
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=29)
# In[27]:
x_train
x_test
# In[28]:
# Проверка баланса выходной переменной
target_count = df['class'].value_counts()
print('Class 0:', target_count[0])
print('Class 1:', target_count[1])
print('Пропорция:', round(target_count[0] / target_count[1], 2), ': 1')

sns.countplot(df['class'], palette="OrRd")
plt.box(False)
plt.xlabel('Class 1/0',fontsize=11)
plt.ylabel('Количество наблюдений',fontsize=11)
plt.show()
# In[29]:
def tree():
    # Обучаем предсказывать каждый класс по сравнению с другим
    plt.figure(figsize=((20,13)))
    clf = DecisionTreeClassifier()
    clf = clf.fit(x_train, y_train)
    plot_tree(clf,filled=True,rounded=True)
    plt.show()

    # fit a CART model to the data
    model_dtc = DecisionTreeClassifier()
    a_dtc = model_dtc.fit(x_train, y_train)
    print(model_dtc)
    # make predictions
    dtc_predicted = a_dtc.predict(x_test) #прогнозированный данные
    # summarize the fit of the model
```

```
    dtc_metr=metrics.classification_report(y_test, dtc_predicted)
    print(dtc_metr)
    print(metrics.confusion_matrix(y_test, dtc_predicted))
    return model_dtc, a_dtc, dtc_predicted, dtc_metr
model_dtc, a_dtc, dtc_predicted, dtc_metr =tree()
# In[30]:
# Визуализация матрицы неточности
def tree_matrix():
    dtc_matrix = confusion_matrix(y_test, dtc_predicted)

    ax= plt.subplot()
    sns.heatmap(pd.DataFrame(dtc_matrix), annot=True,cmap="Reds" , fmt='g')
    plt.title('Confusion matrix Decision Tree', y=1.1)
    ax.set_xlabel('Predicted ');ax.set_ylabel('True');
tree_matrix()
# In[31]:
def tree_roc():
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(y_test, dtc_predicted)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
```

```
    plt.legend(loc="lower right")
    plt.show()
tree_roc()
# In[32]:
def log_reg():
    # logistic regression
    lg_model = LogisticRegression().fit(x_train, y_train)
    lg_predicted = lg_model.predict(x_test)
    # доля правильных ответов алгоритма: Точность = (истинное положительное + истинно отрицательное значение) / всего
    acc_lg = accuracy_score(y_test, lg_predicted)


    # Оценка f1: оценку F1 можно интерпретировать как средневзвешенное значение точности и запоминания, где оценка F1 достигает своего лучшего значения при 1 и худшего значения при 0.
    f1_lg = f1_score(y_test, lg_predicted)
        # долю объектов, названных классификатором положительными и при этом действительно являющимися положительными: если прогноз «да», как часто он оказывается правильным? Точность = истинно положительный / предсказанный да
    precision_lg = precision_score(y_test, lg_predicted)
    # показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм: истинно положительный показатель (чувствительность): когда на самом деле да, как часто он дает предсказание «да»? Истинно положительный процент = истинно положительный / фактический да
    recall_lg = recall_score(y_test, lg_predicted)
        print(f"The accuracy score for LogReg is: {round(acc_lg,3)*100}%")
    print(f"The f1 score for LogReg is: {round(f1_lg,3)*100}%")
    print(f"The precision score for LogReg is: {round(precision_lg,3)*100}%")
    print(f"The recall score for LogReg is: {round(recall_lg,3)*100}%")


    return lg_predicted, acc_lg, f1_lg, precision_lg, recall_lg


lg_predicted, acc_lg, f1_lg, precision_lg, recall_lg = log_reg()
# In[33]:
def log_reg_matrix():
    # матрица неточности
```

```
    matrix_log = confusion_matrix(y_test, lg_predicted)
    ax= plt.subplot()
    sns.heatmap(pd.DataFrame(matrix_log), annot=True,cmap="Reds" , fmt='g')
    plt.title('Confusion matrix Logistic Regression\n', y=1.1)
log_reg_matrix()
# In[34]:
def log_reg_roc():
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(y_test, lg_predicted)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
log_reg_roc()
# In[35]:
def k_means():
    km_model = KNeighborsClassifier()
    a_km=km_model.fit(x_train,y_train)
    print(a_km)
    km_predicted = km_model.predict(x_test)
    km_metr=metrics.classification_report(y_test,km_predicted)
```

```
    print(km_metr)
    print(metrics.confusion_matrix(y_test, km_predicted))
    return km_model, a_km, km_predicted, km_metr
km_model, a_km, km_predicted, km_metr =k_means()
# In[36]:
def k_means_matrix():
    # матрица неточности
    matrix_km = confusion_matrix(y_test,km_predicted)

    ax= plt.subplot()
    sns.heatmap(pd.DataFrame(matrix_km), annot=True,cmap="Reds" , fmt='g')
    plt.title('Confusion matrix K-means\n', y=1.1)
    ax.set_xlabel('Predicted ');ax.set_ylabel('True');
k_means_matrix()
# In[37]:
def k_means_roc():
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(y_test, km_predicted)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
```

```
    plt.show()
k_means_roc()
# In[38]:
def vectors():
    svc_model = SVC()
    a_svc=svc_model.fit(x_train,y_train)
    print(a_svc)
    svc_expected = y_test
    svc_predicted = svc_model.predict(x_test)
    svc_metr=metrics.classification_report(svc_expected,svc_predicted)
    print(svc_metr)
    print(metrics.confusion_matrix(svc_expected,svc_predicted))
    return svc_model, a_svc, svc_predicted,svc_expected, svc_metr
svc_model, a_svc, svc_predicted,svc_expected, svc_metr=vectors()
# In[39]:
def vectors_matrix():
    # матрица неточности
    matrix_svc = confusion_matrix(svc_expected,svc_predicted)


    ax= plt.subplot()
    sns.heatmap(pd.DataFrame(matrix_svc), annot=True,cmap="Reds" , fmt='g')
    plt.title('Confusion matrix SVC\n', y=1.1)
    ax.set_xlabel('Predicted ');ax.set_ylabel('True');
vectors_matrix()
# In[40]:
def vectors_roc():
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(svc_expected,svc_predicted)
    roc_auc = auc(fpr, tpr)
```

```
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
vectors_roc()
# In[41]:
def neuro():
    #инициализация
    ppn = Perceptron(max_iter=1000, eta0=0.1, random_state=0)

    #настройка
    a_ppn = ppn.fit(x_train, y_train)

    ppn_predicted = ppn.predict(x_test)

    print('Неверно классифицированные образцы: %d' % (y_test != ppn_predicted).sum())
    print('Точность: %.2f' % accuracy_score(y_test, ppn_predicted))
    return ppn, a_ppn, ppn_predicted

ppn, a_ppn, ppn_predicted = neuro()
# In[42]:
def neuro_matrix():
    # матрица неточности

    ppn_matrix = confusion_matrix(y_test, ppn_predicted)
```

```
    sns.heatmap(pd.DataFrame(ppn_matrix), annot=True,cmap="Reds" , fmt='g')
    plt.tight_layout()
    plt.title('Матрица неточности Персептрон\n', y=1.1)
neuro_matrix()
# In[43]:
def neuro_roc():
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(y_test, ppn_predicted)
    roc_auc = auc(fpr, tpr)


    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
neuro_roc()
# In[44]:
def ms_neuro():
    mlp = MLPClassifier(activation='relu', solver='lbfgs', alpha=1e-5, max_iter=1000, hidden_layer_sizes=(5,),
random_state=1)
    a_mlp=mlp.fit(x_train,y_train)
    mlp_pred = mlp.predict(x_test)
    print('Неверно классифицированные образцы: %d' % (y_test != mlp_pred).sum())
    print('Точность: %.2f' % accuracy_score(y_test, mlp_pred))
```

```
    return mlp, a_mlp, mlp_pred
mlp, a_mlp, mlp_pred = ms_neuro()
# In[45]:
def ms_neuro_matrix():
    # матица неточности

    mlp_matrix = confusion_matrix(y_test, mlp_pred)

    ax= plt.subplot()
    sns.heatmap(pd.DataFrame(mlp_matrix), annot=True,cmap="Reds" , fmt='g')
    plt.title('Матрица неточности MLP\n', y=1.1)
    ax.set_xlabel('Predicted ');ax.set_ylabel('True');
ms_neuro_matrix()
# In[46]:
def ms_neuro_roc():
    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    fpr, tpr, thresholds = roc_curve(y_test, mlp_pred)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

```
ms_neuro_roc()


# In[47]:
get_ipython().run_cell_magic('javascript', '', 'IPython.OutputArea.auto_scroll_threshold = 9999;')
# In[48]:
while True:
    print ("""
Выберите метод для анализа:
1. Дерево решений
2. Логистическая регрессия
3. К-ближайших соседей
4. Метод опорных векторов
5. Нейронная сеть (персептрон)
6. Многослойная нейронная сеть
7. Выход
    """)
    method=input("Ответ: ")
    if method=="1":
        print("Дерево решений")
        model_dtc, a_dtc, dtc_predicted, dtc_metr =tree()
        tree_matrix()
        tree_roc()
    elif method=="2":
        print("Логистическая регрессия")
        lg_predicted, acc_lg, f1_lg, precision_lg, recall_lg = log_reg()
        log_reg_matrix()
        log_reg_roc()
    elif method=="3":
        print("К-ближайших соседей")
        km_model, a_km, km_predicted, km_metr =k_means()
        k_means_matrix()
        k_means_roc()
    elif method=="4":
```

```
    print("Метод опорных векторов")
    svc_model, a_svc, svc_predicted,svc_expected, svc_metr=vectors()
    vectors_matrix()
    vectors_roc()
elif method=="5":
    print("Нейронная сеть (персептрон)")
    ppn, a_ppn, ppn_predicted = neuro()
    neuro_matrix()
    neuro_roc()
elif method=="6":
    print("Многослойная нейронная сеть")
    mlp, a_mlp, mlp_pred = ms_neuro()
    ms_neuro_matrix()
    ms_neuro_roc()
elif method=="7":
    break;
else:
    print("Неверный ввод")
```