|  |  |
| --- | --- |
| Изображение выглядит как эмблема, герб, нашивка, символ  Автоматически созданное описание | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_\_\_\_\_\_\_\_\_\_\_Разработка и оценка моделей\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_машинного обучения\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_ИУ5-63Б\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Лобанов И.И.\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_****Ю.Е. Гапанюк\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2024 г.***Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ5\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_В.И. Терехов\_\_

(И.О.Фамилия)

«\_07\_» \_\_\_\_февраля\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Предсказание ядовитости гриба\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_ИУ5-63Б\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Лобанов Иван Игоревич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИССЛЕДОВАТЕЛЬСКАЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_КАФЕДРА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_\_\_\_ нед., 50% к \_\_\_\_ нед., 75% к \_\_\_ нед., 100% к \_\_\_\_ нед.

***Техническое задание*** \_\_\_\_*Решить задачу классификацию по определению ядовитый гриб или нет с использованием материалов дисциплины «Технологии машинного обучения»*\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_23\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_07\_» \_\_\_\_февраля\_\_\_\_\_\_\_ 2024 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Ю.Е. Гапанюк\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_**И. И. Лобанов**\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc168506849)

[Загрузка датасета 6](#_Toc168506850)

[Разведочный анализ датасета 6](#_Toc168506851)

[Основная информация о датасете 6](#_Toc168506852)

[Для понимания датасета построим графики 7](#_Toc168506853)

[Посмотрим на целевой признак 7](#_Toc168506854)

[Выбор признаков, подходящих для построения моделей. 8](#_Toc168506855)

[Выбираем признаки 8](#_Toc168506856)

[Выполним масштабирование данных 8](#_Toc168506857)

[Проведение корреляционного анализа данных. Выбор метрик. Выбор моделей. 10](#_Toc168506858)

[Корреляционный анализ 10](#_Toc168506859)

[Выбор метрик для последующей оценки качества моделей 11](#_Toc168506860)

[Выбор моделей 12](#_Toc168506861)

[Разделение датасета на обучающую и тестовую выборки 12](#_Toc168506862)

[Начальное обучение моделей 13](#_Toc168506863)

[Подбор гиперпараметров 16](#_Toc168506864)

[Сравнение качества полученных моделей с качеством baseline-моделей 17](#_Toc168506865)

[Выводы 18](#_Toc168506866)

[Заключение 22](#_Toc168506867)

[Литература 23](#_Toc168506872)

## Введение

В данной научно-исследовательской работе предстоит выполнить типовую задачу машинного обучения - провести анализ данных, некоторые операции с датасетом, подобрать модели, а также подобрать наиболее подходящие гиперпараметры выбранных моделей.

Машинное обучение очень актуально в современном мире, оно используется практически во всех сферах. Программист должен подбирать подходящие технологии машинного обучения для достижения наилучших результатов. Чему мы и научимся в этом курсовом проекте. Попробуем не менее пяти видов различных моделей и подберем наилучшую из них на основе выбранных метрик. Также построим вспомогательные графики, которые помогут нам визуально взглянуть на все необходимые показатели.

**Задание**

1. Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.
2. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
4. Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.
5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
6. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
7. Формирование обучающей и тестовой выборок на основе исходного набора данных.
8. Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.
9. Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс- валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.
11. Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

Приведенная схема исследования является рекомендуемой. В зависимости от решаемой задачи возможны модификации.

## ****Загрузка датасета****

In [90]:

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.linear\_model import LinearRegression, LogisticRegression

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier, GradientBoostingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, ConfusionMatrixDisplay, roc\_curve, roc\_auc\_score

%matplotlib inline

sns.set(style="ticks")

In [3]:

data = pd.read\_csv('mushroom.csv')

## ****Разведочный анализ датасета****

### Основная информация о датасете

In [4]:

data.head()

Out[4]:

|  | **cap-diameter** | **cap-shape** | **gill-attachment** | **gill-color** | **stem-height** | **stem-width** | **stem-color** | **season** | **class** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1372 | 2 | 2 | 10 | 3.807467 | 1545 | 11 | 1.804273 | 1 |
| **1** | 1461 | 2 | 2 | 10 | 3.807467 | 1557 | 11 | 1.804273 | 1 |
| **2** | 1371 | 2 | 2 | 10 | 3.612496 | 1566 | 11 | 1.804273 | 1 |
| **3** | 1261 | 6 | 2 | 10 | 3.787572 | 1566 | 11 | 1.804273 | 1 |
| **4** | 1305 | 6 | 2 | 10 | 3.711971 | 1464 | 11 | 0.943195 | 1 |

In [5]:

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 54035 entries, 0 to 54034

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 cap-diameter 54035 non-null int64

1 cap-shape 54035 non-null int64

2 gill-attachment 54035 non-null int64

3 gill-color 54035 non-null int64

4 stem-height 54035 non-null float64

5 stem-width 54035 non-null int64

6 stem-color 54035 non-null int64

7 season 54035 non-null float64

8 class 54035 non-null int64

dtypes: float64(2), int64(7)

memory usage: 3.7 MB

In [6]:

data.shape

Out[6]:

(54035, 9)

Видим, что датасет не содержит пропусков данных

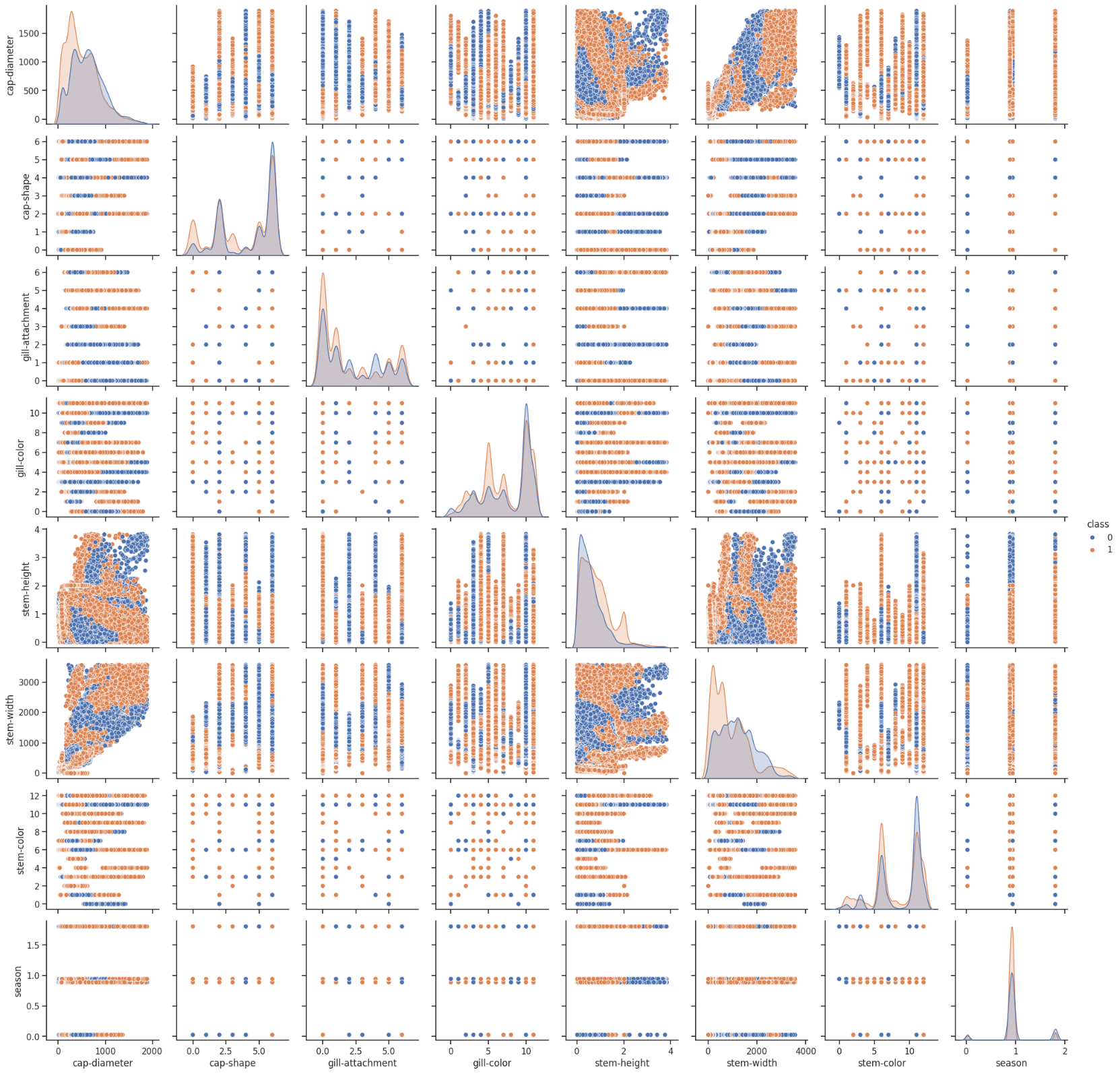
### Для понимания датасета построим графики

In [ ]:

sns.pairplot(data, hue="class")

Out[ ]:

<seaborn.axisgrid.PairGrid at 0x79809a6b5f90>

****

### Посмотрим на целевой признак

In [7]:

data['class'].unique()

Out[7]:

array([1, 0])

In [8]:

data['class'].value\_counts()

Out[8]:

class

1 29675

0 24360

Name: count, dtype: int64

In [9]:

total = data.shape[0]

class\_1, class\_0 = data['class'].value\_counts()

print('Класс 0 составляет {}%, а класс 1 составляет {}%.'.format(round(class\_0 / total, 4)\*100, round(class\_1 / total, 4)\*100))

Класс 0 составляет 45.08%, а класс 1 составляет 54.92%.

Дисбаланс классов практически отсутствует.

## ****Выбор признаков, подходящих для построения моделей.****

### Выбираем признаки

In [10]:

data.dtypes

Out[10]:

cap-diameter int64

cap-shape int64

gill-attachment int64

gill-color int64

stem-height float64

stem-width int64

stem-color int64

season float64

class int64

dtype: object

Для построения моделей будем использовать все признаки.

Категориальные признаки уже закодированы с помощью подхода LabelEncoding.

### Выполним масштабирование данных

In [11]:

scale\_cols = ['cap-diameter', 'stem-height', 'stem-width']

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(data[scale\_cols])

*# Добавим масштабированные данные в набор данных*

for i in range(len(scale\_cols)):

col = scale\_cols[i]

new\_col\_name = col + '\_scaled'

data[new\_col\_name] = scaled\_data[:,i]

In [12]:

data.head()

Out[12]:

|  | **cap-diameter** | **cap-shape** | **gill-attachment** | **gill-color** | **stem-height** | **stem-width** | **stem-color** | **season** | **class** | **cap-diameter\_scaled** | **stem-height\_scaled** | **stem-width\_scaled** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1372 | 2 | 2 | 10 | 3.807467 | 1545 | 11 | 1.804273 | 1 | 0.725542 | 0.992737 | 0.432894 |
| **1** | 1461 | 2 | 2 | 10 | 3.807467 | 1557 | 11 | 1.804273 | 1 | 0.772607 | 0.992737 | 0.436257 |
| **2** | 1371 | 2 | 2 | 10 | 3.612496 | 1566 | 11 | 1.804273 | 1 | 0.725013 | 0.941896 | 0.438778 |
| **3** | 1261 | 6 | 2 | 10 | 3.787572 | 1566 | 11 | 1.804273 | 1 | 0.666843 | 0.987549 | 0.438778 |
| **4** | 1305 | 6 | 2 | 10 | 3.711971 | 1464 | 11 | 0.943195 | 1 | 0.690111 | 0.967835 | 0.410199 |

In [13]:

for col in scale\_cols:

col\_scaled = col + '\_scaled'

fig, ax = plt.subplots(1, 2, figsize=(8,3))

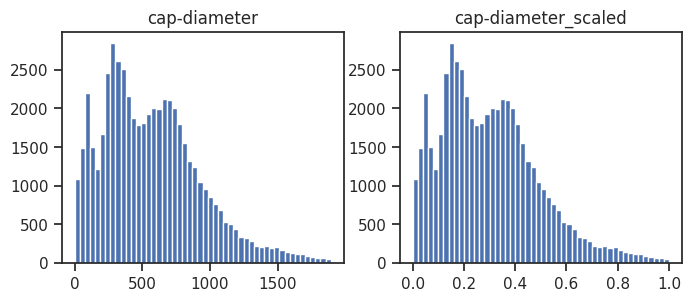
ax[0].hist(data[col], 50)

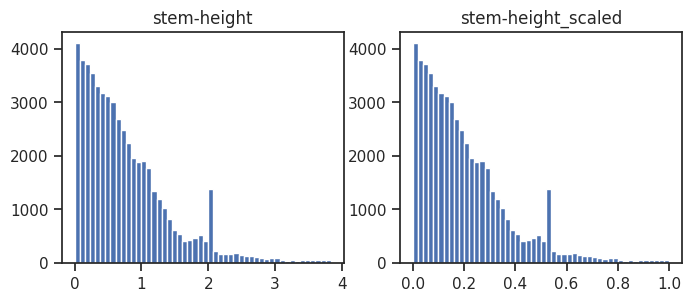
ax[1].hist(data[col\_scaled], 50)

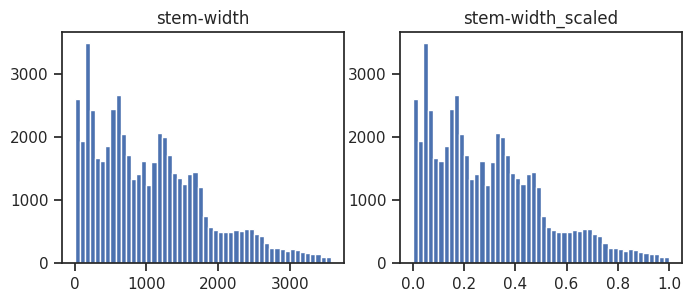
ax[0].title.set\_text(col)

ax[1].title.set\_text(col\_scaled)

plt.show()

****

****

****

Видим, что распределение не поменялось, значит масштабирование данных прошло успешно.

## ****Проведение корреляционного анализа данных. Выбор метрик. Выбор моделей.****

### Корреляционный анализ

In [14]:

train\_cols = ['cap-diameter\_scaled', 'cap-shape', 'gill-attachment', 'gill-color', 'stem-height\_scaled', 'stem-width\_scaled', 'stem-color', 'season', 'class']

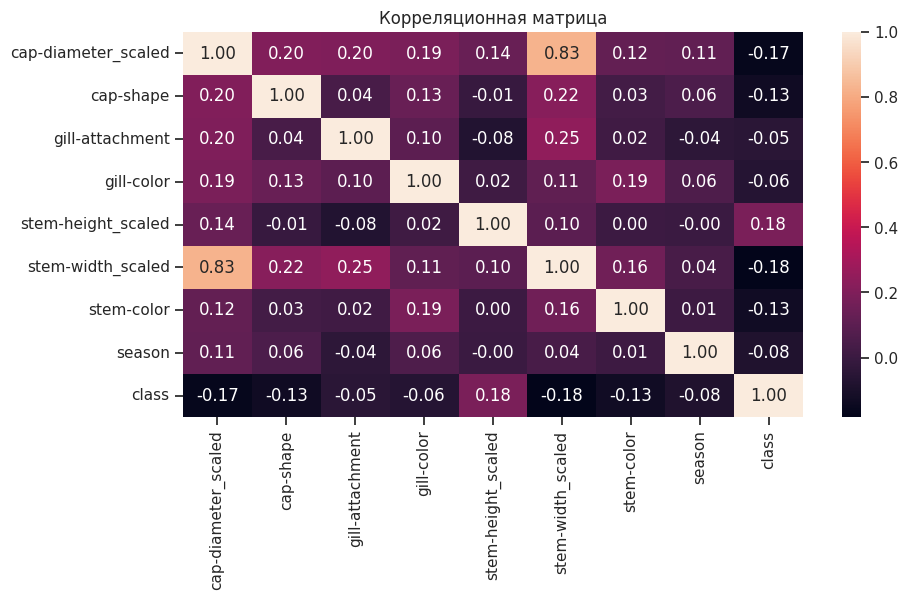
In [15]:

fig, ax = plt.subplots(figsize=(10,5))

sns.heatmap(data[train\_cols].corr(), annot=True, fmt='.2f')

ax.set\_title('Корреляционная матрица')

plt.show()

****

Делаем вывод, что целевой признак напрямую не зависит ни от одного другого.

Также видим, что толщина ножки гриба сильно зависит диаметра шляпки, что в целом логично.

### Выбор метрик для последующей оценки качества моделей

В качестве метрик будем использовать те, которые формируются на основе confusion-матрицы: accuracy, precision, recall, f1-меру. Эти метрики очень хорошо подходят для оценки решения задачи классификации.

Также будем использовать метрику roc\_auc, которая основана на вычислении следующих характеристик:

TPR=TP/(TP+FN)

* True Positive Rate, откладывается по оси ординат. Совпадает с recall.

FPR=FP/(FP+TN)

* False Positive Rate, откладывается по оси абсцисс. Показывает какую долю из объектов отрицательного класса алгоритм предсказал неверно.

Идеальная ROC-кривая проходит через точки (0,0)-(0,1)-(1,1), то есть через верхний левый угол графика.

Чем сильнее отклоняется кривая от верхнего левого угла графика, тем хуже качество классификации.

В качестве количественной метрики используется площадь под кривой - ROC AUC (Area Under the Receiver Operating Characteristic Curve). Чем ниже проходит кривая тем меньше ее площадь и тем хуже качество классификатора.

In [78]:

class MetricLogger:

def \_\_init\_\_(self):

self.df = pd.DataFrame(

{'metric': pd.Series([], dtype='str'),

'alg': pd.Series([], dtype='str'),

'value': pd.Series([], dtype='float')})

def add(self, metric, alg, value):

"""

Добавление значения

"""

*# Удаление значения, если оно уже было ранее добавлено*

self.df.drop(self.df[(self.df['metric'] == metric) & (self.df['alg'] == alg)].index, inplace=True)

*# Добавление нового значения*

temp = pd.DataFrame([{'metric': metric, 'alg': alg, 'value': value}])

self.df = pd.concat([self.df, temp], ignore\_index=True)

def get\_data\_for\_metric(self, metric, ascending=True):

"""

Формирование данных с фильтром по метрике

"""

temp\_data = self.df[self.df['metric'] == metric]

temp\_data\_2 = temp\_data.sort\_values(by='value', ascending=ascending)

return temp\_data\_2['alg'].values, temp\_data\_2['value'].values

def plot(self, str\_header, metric, ascending=True, figsize=(5, 5)):

"""

Вывод графика

"""

array\_labels, array\_metric = self.get\_data\_for\_metric(metric, ascending)

fig, ax1 = plt.subplots(figsize=figsize)

pos = np.arange(len(array\_metric))

rects = ax1.barh(pos, array\_metric,

align='center',

height=0.5,

tick\_label=array\_labels)

ax1.set\_title(str\_header)

for a, b in zip(pos, array\_metric):

plt.text(0.5, a - 0.05, str(round(b, 3)), color='white')

plt.show()

In [91]:

def draw\_roc\_curve(y\_true, y\_score, ax, pos\_label=1, average='micro'):

fpr, tpr, thresholds = roc\_curve(y\_true, y\_score,

pos\_label=pos\_label)

roc\_auc\_value = roc\_auc\_score(y\_true, y\_score, average=average)

*#plt.figure()*

lw = 2

ax.plot(fpr, tpr, color='darkorange',

lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc\_value)

ax.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')

ax.set\_xlim([0.0, 1.0])

ax.set\_xlim([0.0, 1.05])

ax.set\_xlabel('False Positive Rate')

ax.set\_ylabel('True Positive Rate')

ax.set\_title('Receiver operating characteristic')

ax.legend(loc="lower right")

### Выбор моделей

Решаем задачу классификации, поэтому будем использовать соедующие модели:

* Логистическая регрессия
* Метод ближайших соседей
* Решающее дерево
* Случайный лес
* Градиентный бустинг

## ****Разделение датасета на обучающую и тестовую выборки****

In [16]:

train, test = train\_test\_split(data[train\_cols], test\_size=0.2, random\_state=42)

print("Train size:", len(train))

print("Test size:", len(test))

Train size: 43228

Test size: 10807

In [17]:

X\_train = train.drop('class', axis=1)

y\_train = train['class']

X\_test = test.drop('class', axis=1)

y\_test = test['class']

## ****Начальное обучение моделей****

In [49]:

models = {

'LogR': LogisticRegression(max\_iter=1000000),

'KNN\_50': KNeighborsClassifier(n\_neighbors=50),

'Tree': DecisionTreeClassifier(),

'RF': RandomForestClassifier(),

'GB': GradientBoostingClassifier()

}

In [79]:

*# Сохранение метрик*

logger = MetricLogger()

In [96]:

def train\_model(model\_name, model, logger):

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

y\_score = model.predict\_proba(X\_test)[:, 1] *# Предполагаем, что используется метод predict\_proba*

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

roc\_auc = roc\_auc\_score(y\_test, y\_score)

logger.add('accuracy', model\_name, accuracy)

logger.add('precision', model\_name, precision)

logger.add('recall', model\_name, recall)

logger.add('f1', model\_name, f1)

logger.add('roc\_auc', model\_name, roc\_auc)

print(f'{model\_name} accuracy: {accuracy}')

print(f'{model\_name} precision: {precision}')

print(f'{model\_name} recall: {recall}')

print(f'{model\_name} f1: {f1}')

print()

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

draw\_roc\_curve(y\_test, y\_score, axes[0])

axes[0].set\_title(f'ROC Curve for {model\_name}')

disp = ConfusionMatrixDisplay.from\_predictions(y\_test, y\_pred, xticks\_rotation='vertical', ax=axes[1])

disp.ax\_.set\_title(f'Confusion Matrix for {model\_name}')

plt.tight\_layout()

plt.show()

In [97]:

for model\_name, model in models.items():

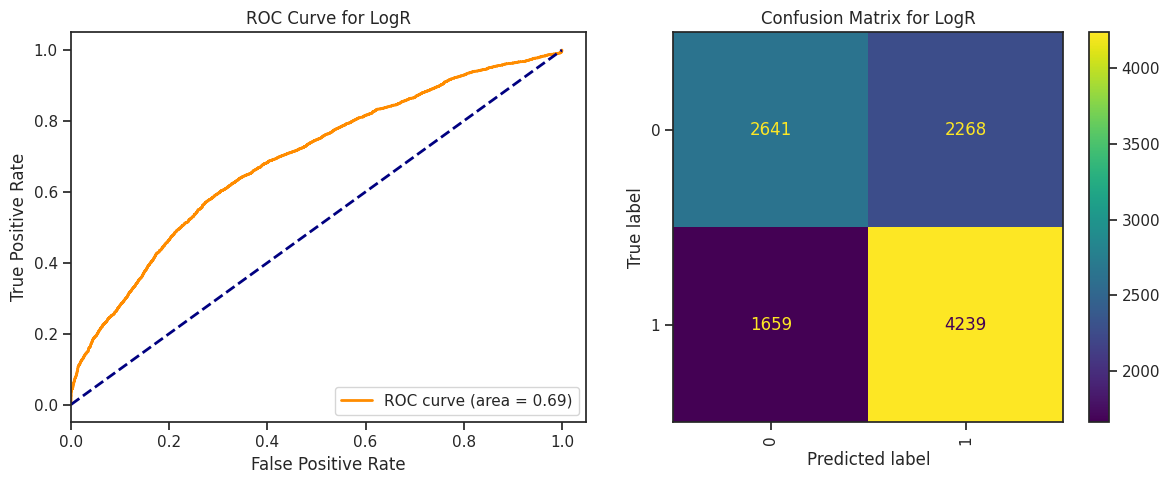
train\_model(model\_name, model, logger)

LogR accuracy: 0.6366244101045618

LogR precision: 0.6514522821576764

LogR recall: 0.7187182095625636

LogR f1: 0.6834340991535671

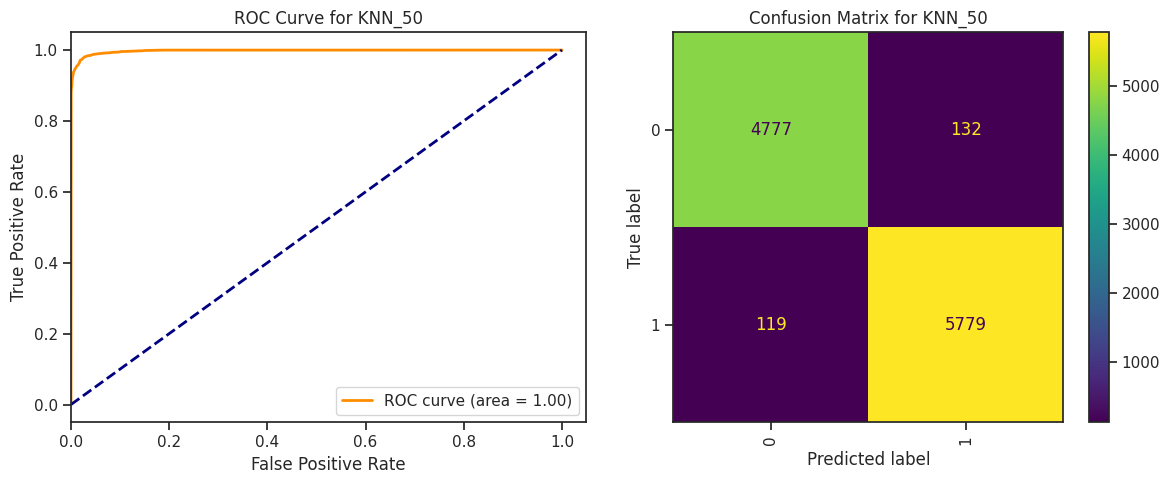
****

KNN\_50 accuracy: 0.9767743129453133

KNN\_50 precision: 0.9776687531720522

KNN\_50 recall: 0.9798236690403527

KNN\_50 f1: 0.9787450249809467

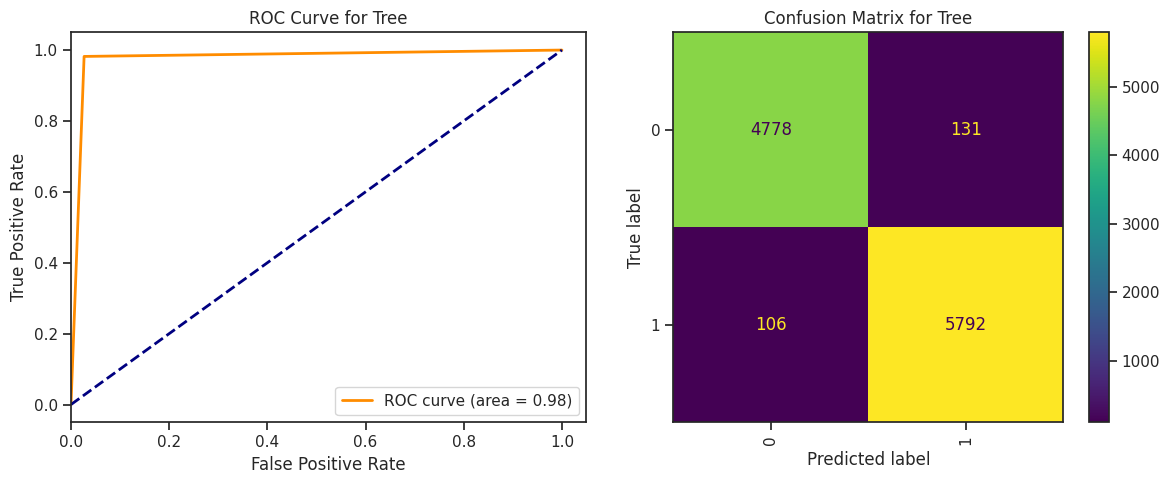
****

Tree accuracy: 0.9780697695937818

Tree precision: 0.9778828296471382

Tree recall: 0.9820278060359444

Tree f1: 0.9799509347770916

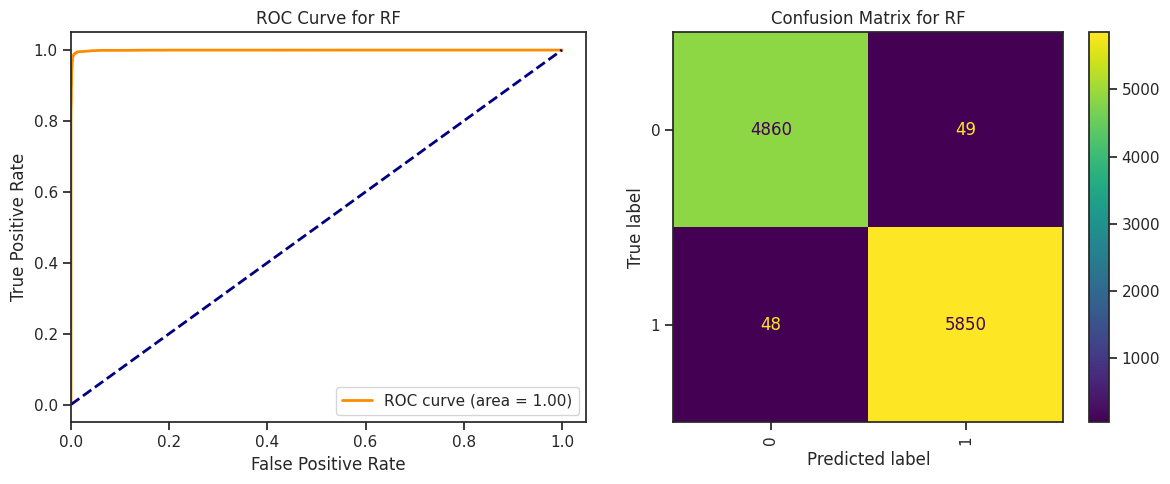
****

RF accuracy: 0.9910243360784676

RF precision: 0.9916935073741312

RF recall: 0.9918616480162767

RF f1: 0.9917775705687887

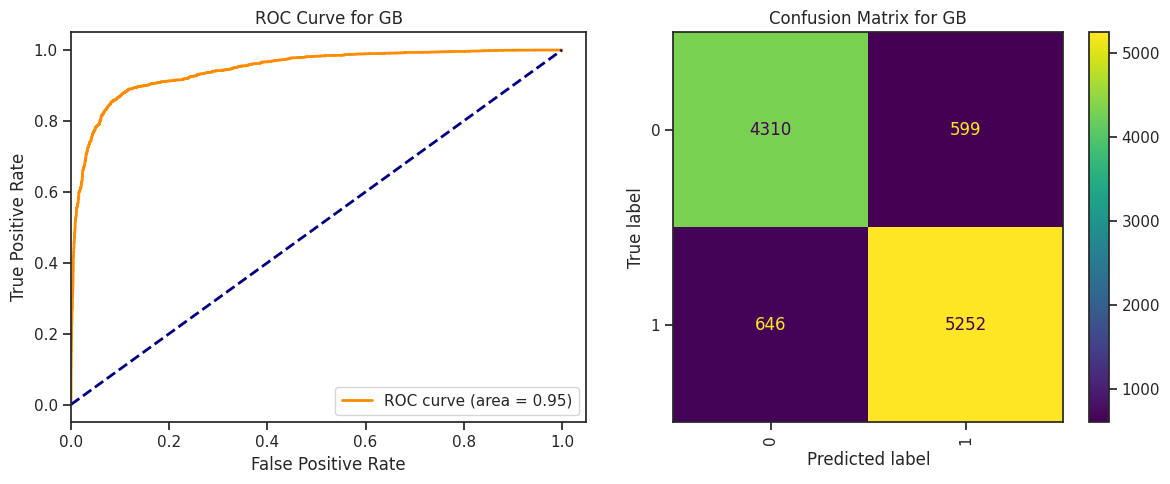
****

GB accuracy: 0.8847968909040437

GB precision: 0.8976243377200479

GB recall: 0.8904713462190573

GB f1: 0.8940335347689166

****

Видим, что из коробки лучше всего себя показал RandomForestClassifier

## ****Подбор гиперпараметров****

In [52]:

X\_train.shape

Out[52]:

(43228, 8)

In [65]:

n\_range\_list = list(range(0,51,5))

n\_range\_list[0] = 1

n\_range = np.array(n\_range\_list)

tuned\_parameters = [{'n\_neighbors': n\_range}]

tuned\_parameters

Out[65]:

[{'n\_neighbors': array([ 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}]

In [66]:

%%time

grid\_search\_cv = GridSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5, scoring='accuracy')

grid\_search\_cv.fit(X\_train, y\_train)

CPU times: user 53.6 s, sys: 764 ms, total: 54.3 s

Wall time: 1min

Out[66]:

GridSearchCV(cv=5, estimator=KNeighborsClassifier(),

param\_grid=[{'n\_neighbors': array([ 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}],

scoring='accuracy')

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [67]:

*# Лучшая модель*

grid\_search\_cv.best\_estimator\_

Out[67]:

KNeighborsClassifier()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [68]:

*# Лучшее значение параметров*

grid\_search\_cv.best\_params\_

Out[68]:

{'n\_neighbors': 5}

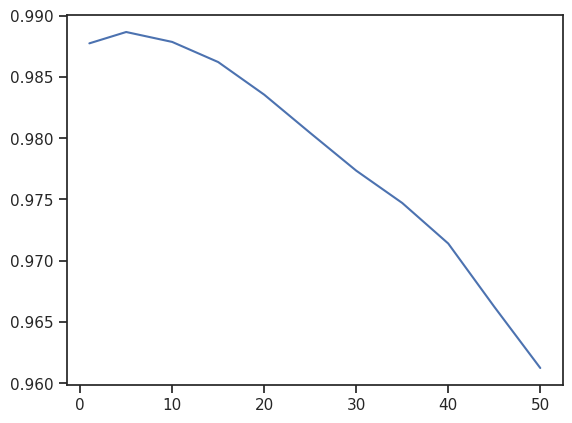
In [69]:

*# Изменение качества на тестовой выборке в зависимости от К-соседей*

plt.plot(n\_range, grid\_search\_cv.cv\_results\_['mean\_test\_score'])

Out[69]:

[<matplotlib.lines.Line2D at 0x7eeeb81f6b60>]

****

## ****Сравнение качества полученных моделей с качеством baseline-моделей****

In [70]:

models\_grid = {

'KNN\_50': KNeighborsClassifier(n\_neighbors=50),

'KNN\_5': grid\_search\_cv.best\_estimator\_

}

In [98]:

for model\_name, model in models\_grid.items():

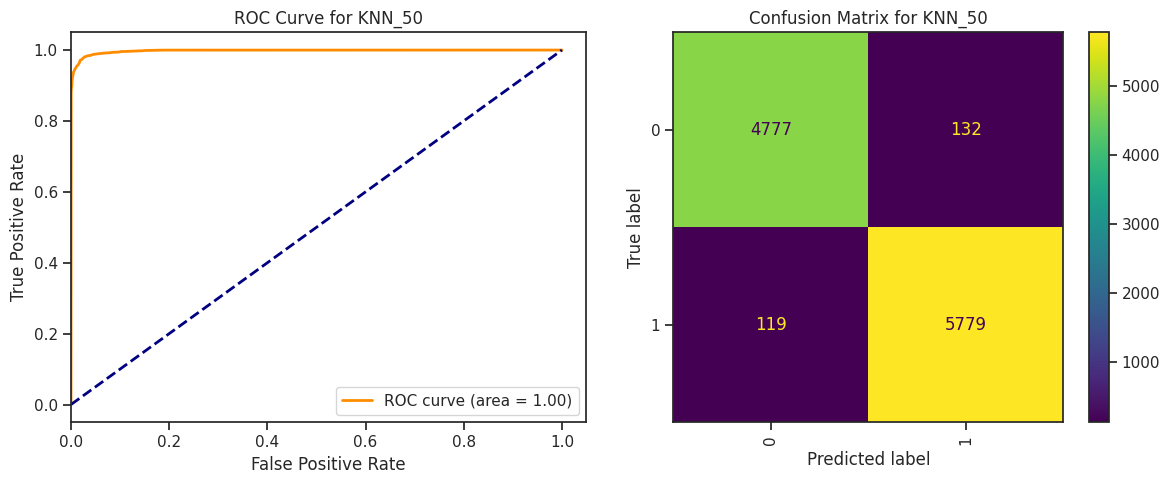
train\_model(model\_name, model, logger)

KNN\_50 accuracy: 0.9767743129453133

KNN\_50 precision: 0.9776687531720522

KNN\_50 recall: 0.9798236690403527

KNN\_50 f1: 0.9787450249809467

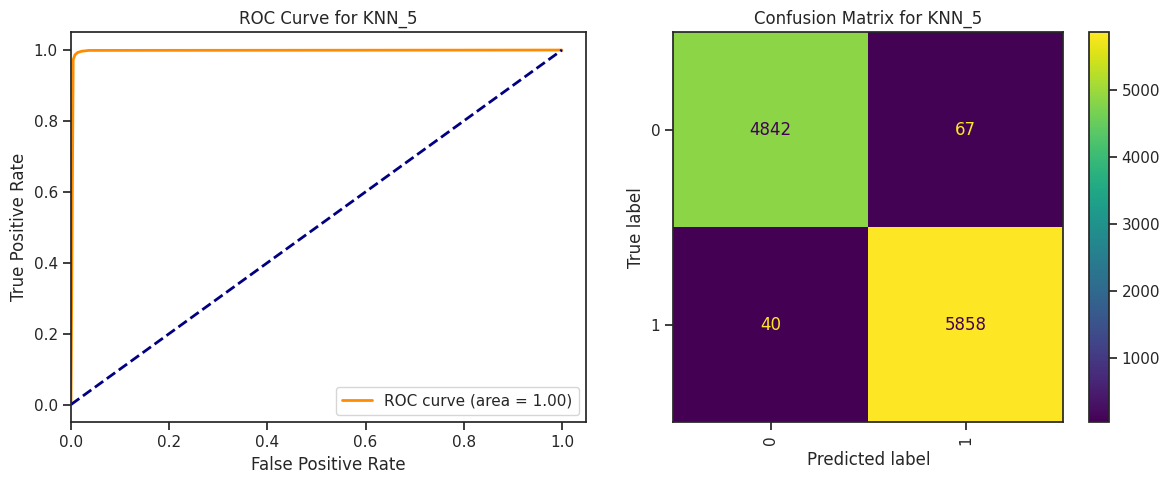
****

KNN\_5 accuracy: 0.9900990099009901

KNN\_5 precision: 0.9886919831223628

KNN\_5 recall: 0.993218040013564

KNN\_5 f1: 0.990949843525332

****

Видим, что KNN(5) действительно по всем метрикам лучше, чем KNN(50), которая была выбрана изначально.

## ****Выводы****

In [99]:

*# Метрики качества модели*

metrics = logger.df['metric'].unique()

metrics

Out[99]:

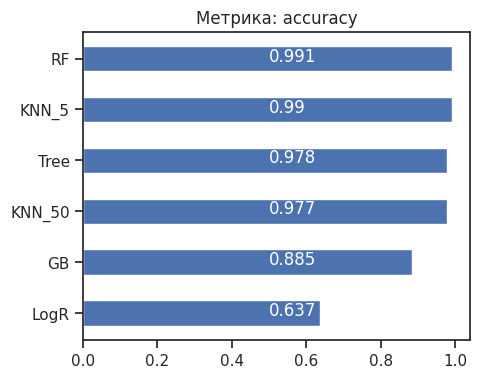
array(['accuracy', 'precision', 'recall', 'f1', 'roc\_auc'], dtype=object)

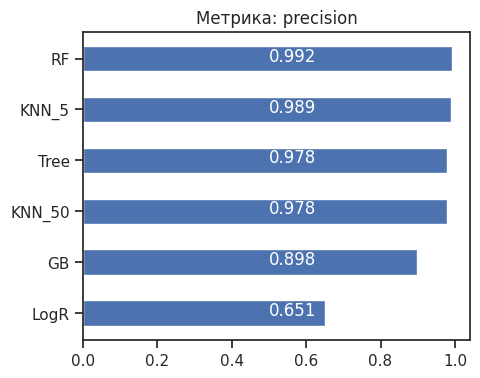
In [100]:

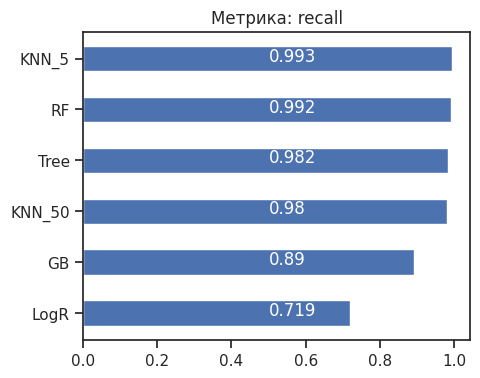
*# Построим графики метрик качества модели*

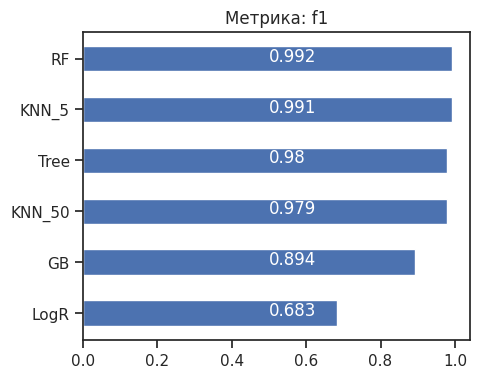
for metric in metrics:

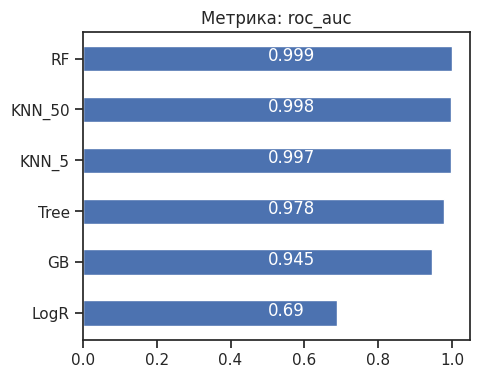
logger.plot('Метрика: ' + metric, metric, figsize=(5, 4))

****

****

****

****

****

Видим, что RandomForestClassifier показал себя лучше всех остальных моделей по 4-м метрикам из 5-и, поэтому лучше всего подходит для решения этой задачи.

## Заключение

### В данной научно-исследовательской работе мы выполнили типовую задачу машинного обучения. Провели анализ данных, преобразовали готовый датасет под наши потребности, подобрали модели, а также подобрали наиболее подходящие гиперпараметры.

### В нашем случае классификатор на основе случайного дерева показал лучший результат, однако для данной модели изменение результатов в результате подбора гиперпараметров было незначительным.

### В данном проекте были закреплены все знания, полученные в курсе лекций и на лабораторных работах. Часть информации была найдена в различных открытых источниках в интернете.

### Проделанная работа вызвала интерес к предмету и дальнейшей работе в этой сфере, которая является одной из самых перспективных и актуальных в современном мире.

## Литература

1. Рукописные лекции за 2024 год по дисциплине «Технологии машинного обучения»
2. https://scikit-learn.org/stable/index.html
3. https:/[/www.ka](http://www.kaggle.com/datasets)g[gle.com/datasets](http://www.kaggle.com/datasets)
4. <http://www.machinelearning.ru/>