



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:
Анализ данных

Студент ИУ5-63Б
(Группа)

(Подпись, дата) **Балабас А.Г.**
(И.О.Фамилия)

Руководитель

(Подпись, дата) **Гапанюк Ю.Е.**
(И.О.Фамилия)

Консультант

(Подпись, дата) _____
(И.О.Фамилия)

2022 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 _____
(Индекс)

(И.О.Фамилия)

« ____ » _____ 20 ____ г.

**З А Д А Н И Е
на выполнение научно-исследовательской работы**

по теме Анализ данных

Студент группы ИУ5-63Б

Балабас Анна Григорьевна

(Фамилия, имя, отчество)

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

Учебная

Источник тематики (кафедра, предприятие, НИР) _____ кафедра _____

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

Техническое задание Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных построить модели машинного обучения для решения задачи классификации.

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.

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

Дата выдачи задания « ____ » _____ 20 ____ г.

Руководитель НИР

(Подпись, дата)

Гапанюк Ю.Е.

(И.О.Фамилия)

Студент

(Подпись, дата)

Балабас А.Г.

(И.О.Фамилия)

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

Оглавление

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.....	4
2. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.....	5
3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.	12
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.	14
5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.	16
6. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.....	16
7. Формирование обучающей и тестовой выборок на основе исходного набора данных.	17
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.	18
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.....	22
10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.	23
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 гиперпараметров на качество моделей и т.д.	24
Источники:	27

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.

В качестве набора данных мы будем использовать набор данных для анализа и прогнозирования сердечного приступа.

<https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset?select=heart.csv>

Датасет состоит из файла: heart.csv

Файл содержит следующие колонки:

- age: Возраст пациента
- sex: Пол пациента
- exng: стенокардия, вызванная физической нагрузкой (1 = да; 0 = нет)
- caa: количество крупных сосудов (0-3)
- cp : тип боли в груди
 - Значение 1: типичная стенокардия
 - Значение 2: атипичная стенокардия
 - Значение 3: неангинозная боль
 - Значение 4: бессимптомный
- trtbps: артериальное давление в покое (в мм рт. ст.)
- chol: холестераль в мг/дл, полученный с помощью датчика ИМТ
- fbs: (уровень сахара в крови натощак > 120 мг/дл) (1 = верно; 0 = неверно)
- rest_ecg: результаты электрокардиографии в покое
 - Значение 0: нормальный
 - Значение 1: наличие аномалии ST-T (инверсия зубца Т и/или элевация или депрессия ST > 0,05 мВ)
 - Значение 2: указание на возможную или определенную гипертрофию левого желудочка по критериям Эстеса.
- thalach: максимальная частота сердечных сокращений

Для решения задачи классификации в качестве целевого признака будем использовать "exng". Поскольку признак содержит только значения 0 и 1, то это задача бинарной классификации.

2. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.

Импорт библиотек и загрузка данных

```
[1] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

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
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import GridSearchCV, LeaveOneOut
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
%matplotlib inline
sns.set(style="ticks")

[4] data = pd.read_csv(filepath_or_buffer="./drive/MyDrive/heart.csv")
```

Основные характеристики датасетов

```
[ ] total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))

Всего строк: 303

[ ] data.columns

Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',
       'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output'],
      dtype='object')

# Список колонок с типами данных
data.dtypes

age          int64
sex          int64
cp           int64
trtbps       int64
chol         int64
fbs          int64
restecg      int64
thalachh     int64
exng         int64
oldpeak      float64
slp          int64
caa          int64
thall        int64
output       int64
dtype: object
```

0

сек.

▶

data.head()

↗

age

sex

cp

trtbps

chol

fbs

restecg

thalachh

exng

oldpeak

slp

caa

thall

output

0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

[]

проверим есть ли пропущенные значения

data.isnull().sum()

age

0

sex

0

cp

0

trtbps

0

chol

0

fbs

0

restecg

0

thalachh

0

exng

0

oldpeak

0

slp

0

caa

0

thall

0

output

0

dtype: int64

Пропусков нет, что очень хорошо.

[]

Основные статистические характеристики набора данных

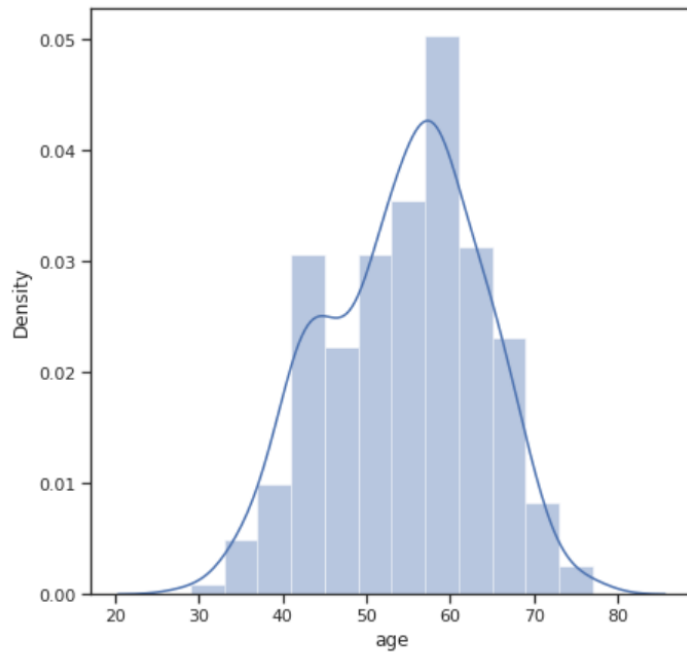
data.describe()

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.468011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

Построение графиков для понимания структуры данных

```
fig, ax = plt.subplots(figsize=(7,7))  
sns.distplot(data['age'])
```

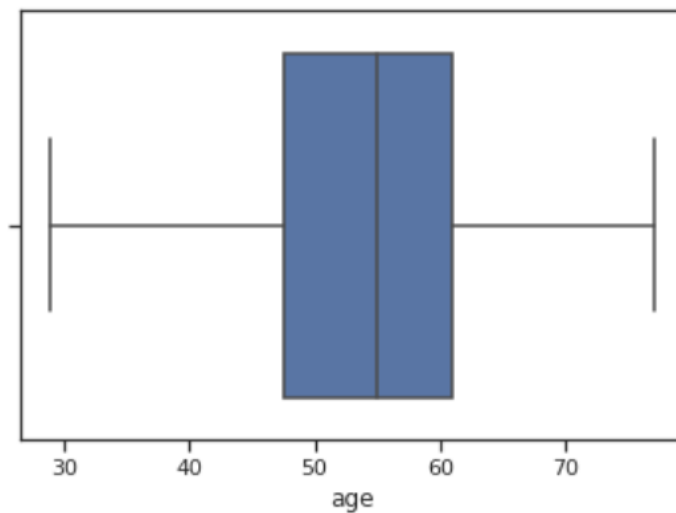
```
[>] /usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a d  
warnings.warn(msg, FutureWarning)  
<matplotlib.axes._subplots.AxesSubplot at 0x7f89e00325d0>
```

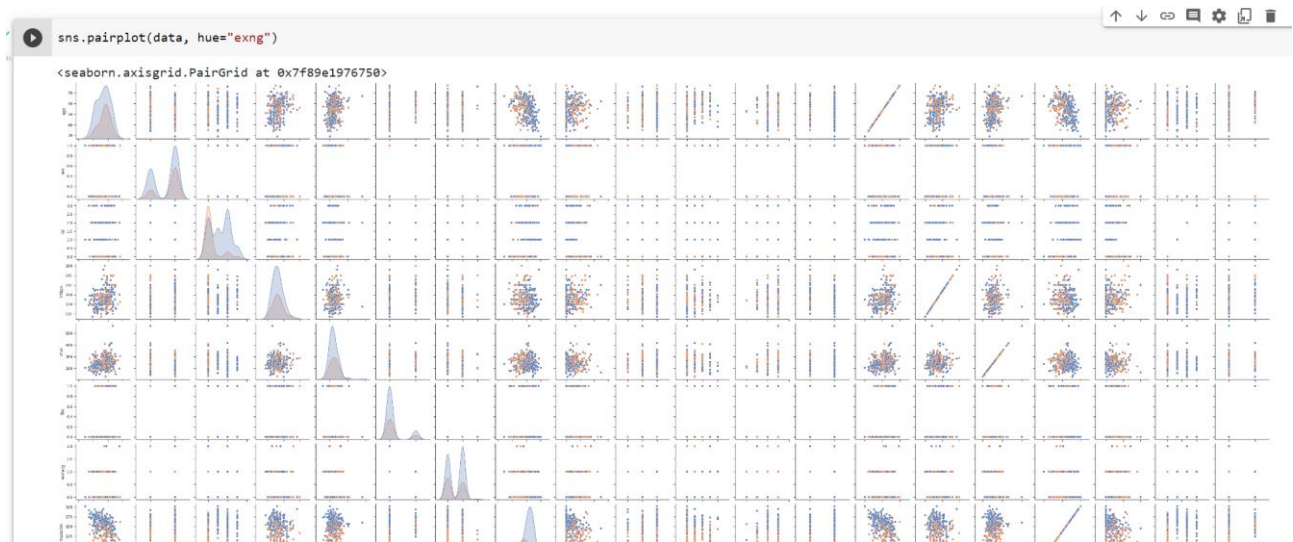


Ящик с усами

```
sns.boxplot(x=data['age'])
```

```
[>] <matplotlib.axes._subplots.AxesSubplot at 0x7f37ae9e8850>
```

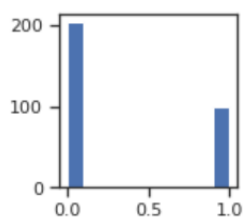




```
# Убедимся, что целевой признак  
# для задачи бинарной классификации содержит только 0 и 1  
data['exng'].unique()
```

```
array([0, 1])
```

```
[ ] # Оценим дисбаланс классов для exng  
fig, ax = plt.subplots(figsize=(2,2))  
plt.hist(data['exng'])  
plt.show()
```



```
[ ] data['exng'].value_counts()
```

```
0    204  
1     99  
Name: exng, dtype: int64
```

```
[ ] # посчитаем дисбаланс классов  
total = data.shape[0]  
class_0, class_1 = data['exng'].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 67.33%, а класс 1 составляет 32.67%.

Достаточно приемлемо

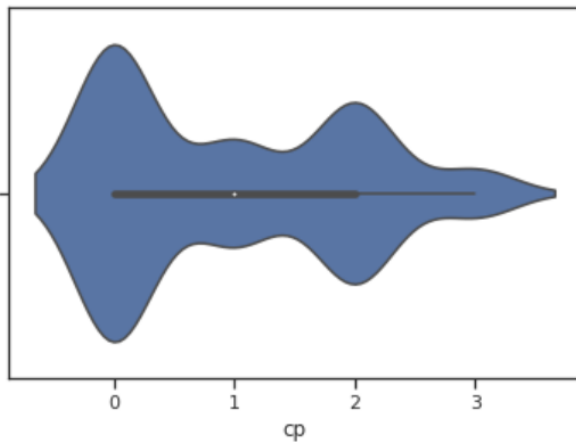
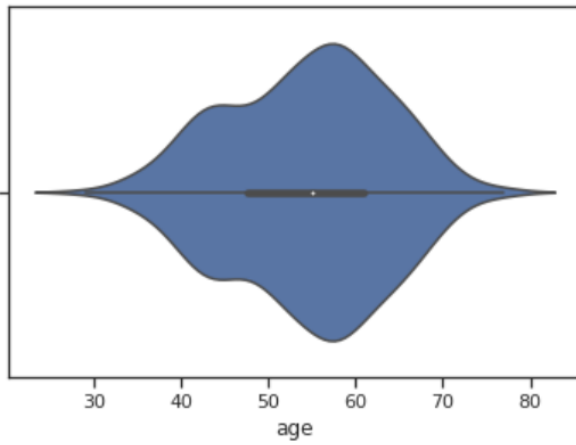
```
[ ] data.columns
```

```
Index(['age', 'sex', 'cp', 'trtbps', 'chol', 'fbs', 'restecg', 'thalachh',  
      'exng', 'oldpeak', 'slp', 'caa', 'thall', 'output'],  
      dtype='object')
```

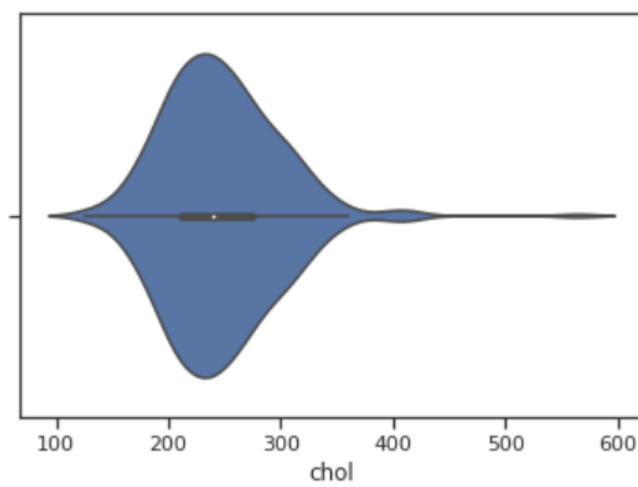
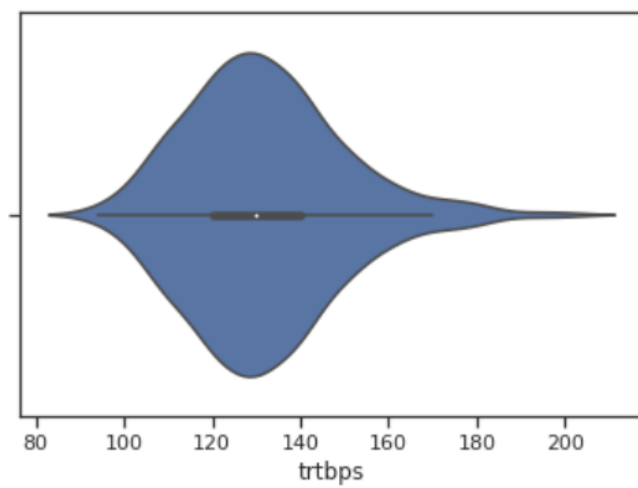


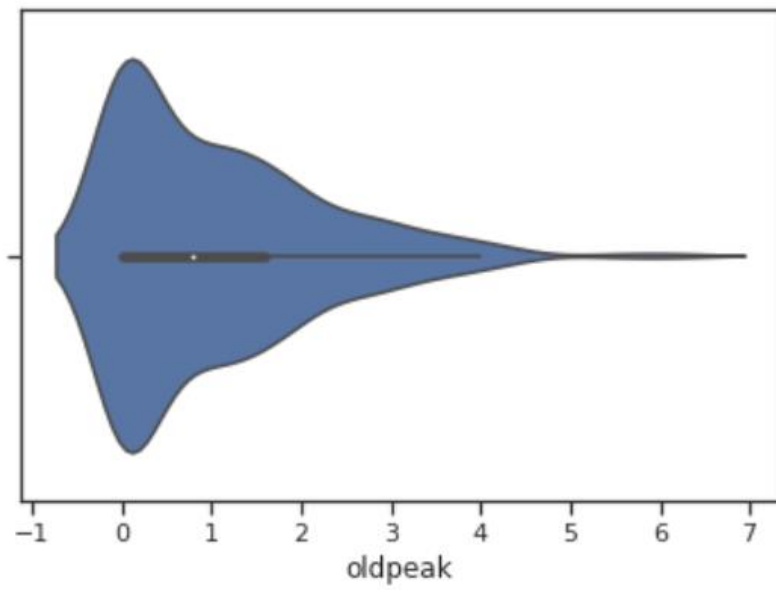
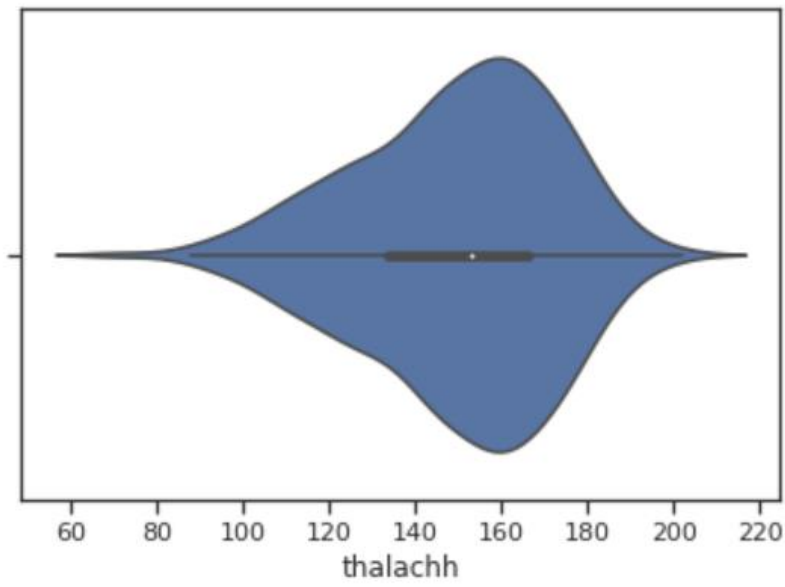

Скрипичные диаграммы для числовых колонок

```
for col in ['age', 'cp', 'trtbps', 'chol', 'thalachh', 'oldpeak']:
    sns.violinplot(x=data[col])
    plt.show()
```



cp





3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.

```
[ ] data.dtypes
```

```
age          int64
sex          int64
cp          int64
trtbps      int64
chol        int64
fbs         int64
restecg     int64
thalachh    int64
exng        int64
oldpeak     float64
slp         int64
caa         int64
thall       int64
output      int64
dtype: object
```

```
[7] # Числовые колонки для масштабирования
scale_cols = ['age', 'trtbps', 'chol', 'thalachh', 'oldpeak', 'cp', 'fbs']
```

```
[8] sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[scale_cols])
```

```
[9] # 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] = sc1_data[:,i]
```

```
[10] data.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	...	caa	thall	output	age_scaled	trtbps_scaled	chol_scaled	thalachh_scaled	oldpeak_scaled	cp_sca
0	63	1	3	145	233	1	0	150	0	2.3	...	0	1	1	0.708333	0.481132	0.244292	0.603053	0.370968	1.000
1	37	1	2	130	250	0	1	187	0	3.5	...	0	2	1	0.166667	0.339623	0.283105	0.885496	0.564516	0.666
2	41	0	1	130	204	0	0	172	0	1.4	...	0	2	1	0.250000	0.339623	0.178082	0.770992	0.225806	0.333
3	56	1	1	120	236	0	1	178	0	0.8	...	0	2	1	0.562500	0.245283	0.251142	0.816794	0.129032	0.333
4	57	0	0	120	354	0	1	163	1	0.6	...	0	2	1	0.583333	0.245283	0.520548	0.702290	0.096774	0.000

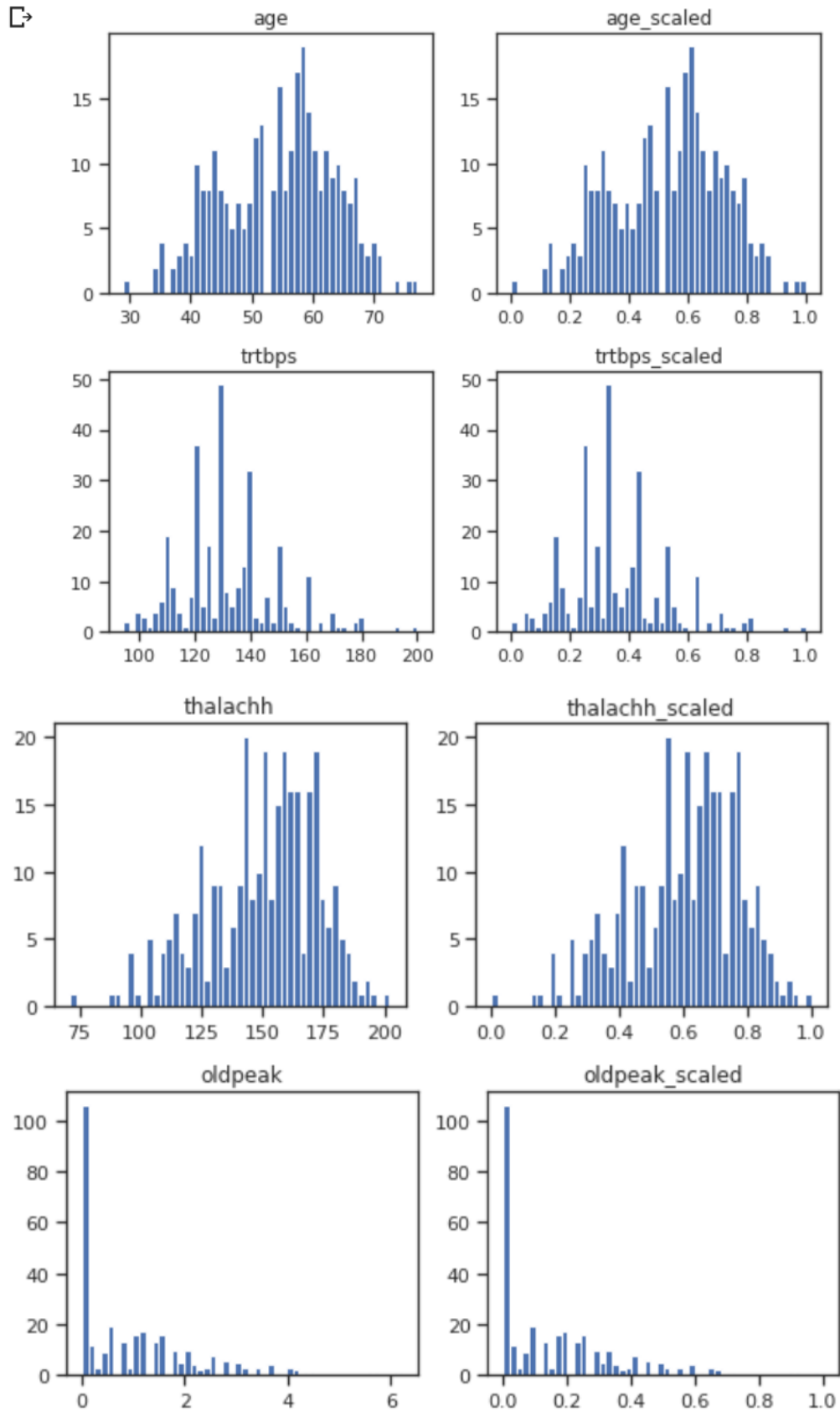
5 rows x 21 columns



```
# Проверим, что масштабирование не повлияло на распределение данных
```

```
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()
```



Всё хорошо

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.

```
# Воспользуемся наличием тестовых выборок,  
# включив их в корреляционную матрицу  
corr_cols_1 = scale_cols + ['exng']  
corr_cols_1
```

```
['age', 'trtbps', 'chol', 'thalachh', 'oldpeak', 'cp', 'fbs', 'exng']
```

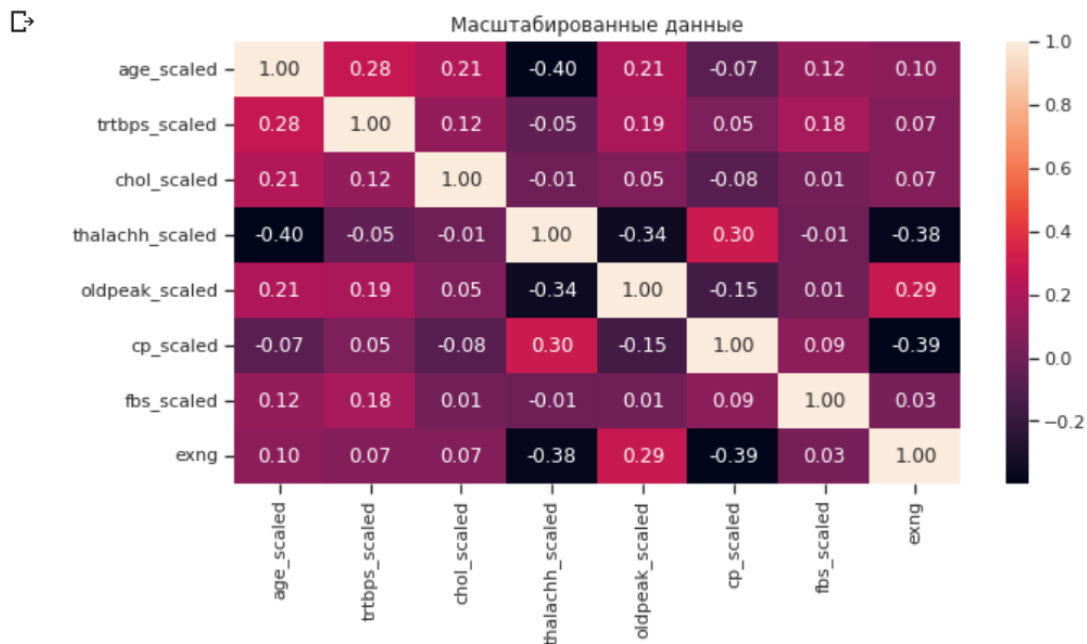
```
[13] scale_cols_postfix = [x+'_scaled' for x in scale_cols]  
corr_cols_2 = scale_cols_postfix + ['exng']  
corr_cols_2
```

```
['age_scaled',  
 'trtbps_scaled',  
 'chol_scaled',  
 'thalachh_scaled',  
 'oldpeak_scaled',  
 'cp_scaled',  
 'fbs_scaled',  
 'exng']
```

```
fig, ax = plt.subplots(figsize=(10,5))  
sns.heatmap(data[corr_cols_1].corr(), annot=True, fmt='.2f')  
ax.set_title('Исходные данные (до масштабирования)')  
plt.show()
```



```
fig, ax = plt.subplots(figsize=(10,5))
sns.heatmap(data[corr_cols_2].corr(), annot=True, fmt='.2f')
ax.set_title('Масштабированные данные')
plt.show()
```



На основе корреляционной матрицы можно сделать следующие выводы:

- Корреляционные матрицы для исходных и масштабированных данных совпадают.
- Целевой признак классификации "exng" наиболее сильно коррелирует с age, trtbps, chol, oldpeak. Эти признаки обязательно следует оставить в модели классификации.

5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.

```
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 = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(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()
```

6. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.

Для задачи классификации будем использовать следующие модели:

- Логистическая регрессия
- Метод ближайших соседей
- Машина опорных векторов
- Решающее дерево
- Случайный лес
- Градиентный бустинг

7. Формирование обучающей и тестовой выборок на основе исходного набора данных.



Разделение выборки на обучающую и тестовую

```
wine_X_train, wine_X_test, wine_y_train, wine_y_test = train_test_split(  
    data, data.exng, test_size=0.2, random_state=1)
```

[] # Признаки для задачи классификации

```
task_clas_cols = ['age_scaled', 'trtbps_scaled',  
                  'chol_scaled', 'oldpeak_scaled']
```

[] # Выборки для задачи классификации

```
clas_X_train = wine_X_train[task_clas_cols]  
clas_X_test = wine_X_test[task_clas_cols]  
clas_Y_train = wine_y_train  
clas_Y_test = wine_y_test  
clas_X_train.shape, clas_X_test.shape, clas_Y_train.shape, clas_Y_test.shape  
  
((242, 4), (61, 4), (242,), (61,))
```

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.



Отрисовка ROC-кривой

```
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")
```

```
[ ] cl1_1 = KNeighborsClassifier(n_neighbors=84)
    cl1_1.fit(clas_X_train, clas_Y_train)
    target1_0 = cl1_1.predict(clas_X_train)
    target1_1 = cl1_1.predict(clas_X_test)
    accuracy_score(clas_Y_train, target1_0), accuracy_score(clas_Y_test, target1_1)

(0.6818181818181818, 0.639344262295082)
```

```
[ ] # Модели
    clas_models = {'LogR': LogisticRegression(),
                  'KNN_5': KNeighborsClassifier(n_neighbors=5),
                  'SVC': SVC(probability=True),
                  'Tree': DecisionTreeClassifier(),
                  'RF': RandomForestClassifier(),
                  'GB': GradientBoostingClassifier()}
```

```
[ ] # Сохранение метрик
    clasMetricLogger = MetricLogger()
```

```

def clas_train_model(model_name, model, clasMetricLogger):
    model.fit(clas_X_train, clas_Y_train)
    # Предсказание значений
    Y_pred = model.predict(clas_X_test)
    # Предсказание вероятности класса "1" для roc auc
    Y_pred_proba_temp = model.predict_proba(clas_X_test)
    Y_pred_proba = Y_pred_proba_temp[:,1]

    precision = precision_score(clas_Y_test.values, Y_pred)
    recall = recall_score(clas_Y_test.values, Y_pred)
    f1 = f1_score(clas_Y_test.values, Y_pred)
    roc_auc = roc_auc_score(clas_Y_test.values, Y_pred_proba)

    clasMetricLogger.add('precision', model_name, precision)
    clasMetricLogger.add('recall', model_name, recall)
    clasMetricLogger.add('f1', model_name, f1)
    clasMetricLogger.add('roc_auc', model_name, roc_auc)


    fig, ax = plt.subplots(ncols=2, figsize=(10,5))
    draw_roc_curve(clas_Y_test.values, Y_pred_proba, ax[0])
    plot_confusion_matrix(model, clas_X_test, clas_Y_test.values, ax=ax[1],
                          display_labels=['0', '1'],
                          cmap=plt.cm.Blues, normalize='true')
    fig.suptitle(model_name)
    plt.show()

```

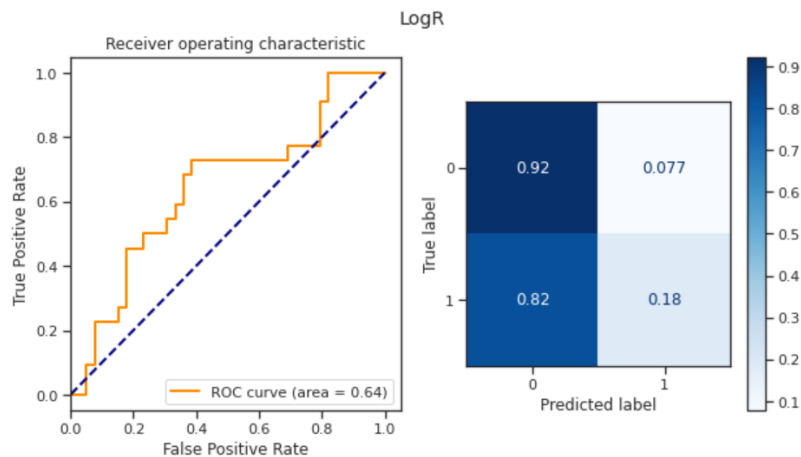
```

for model_name, model in clas_models.items():
    clas_train_model(model_name, model, clasMetricLogger)

```

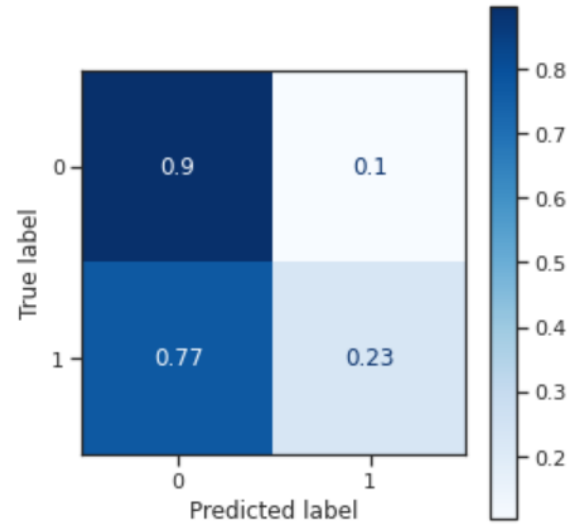
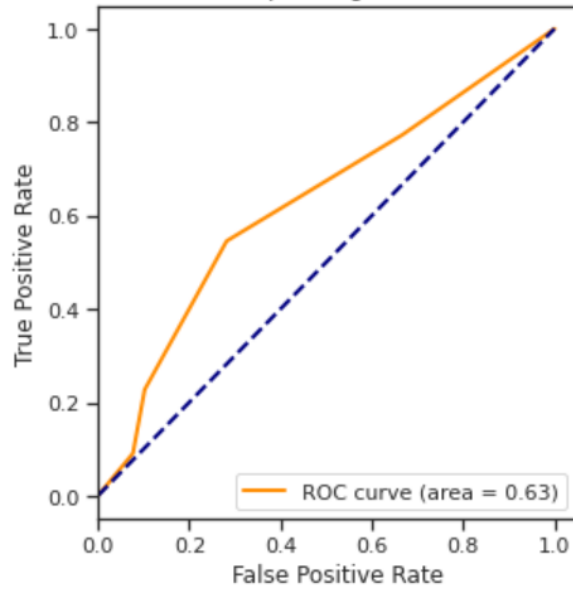
 /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_

warnings.warn(msg, category=FutureWarning)



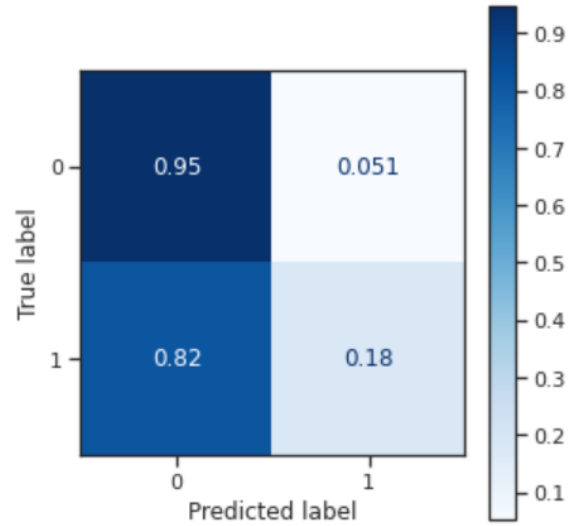
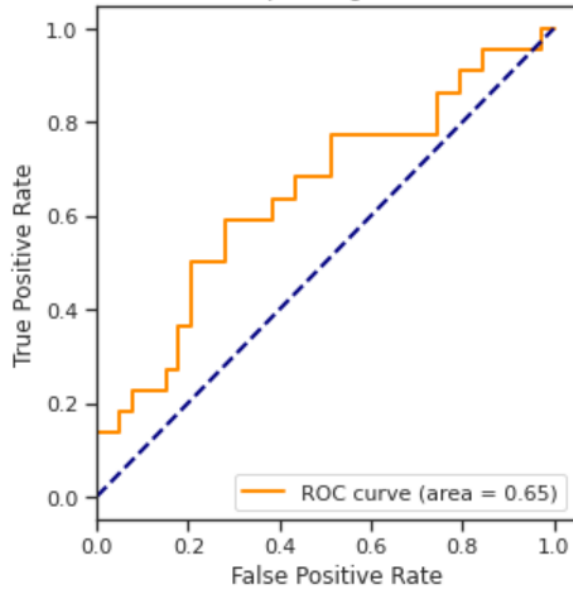
KNN_5

Receiver operating characteristic



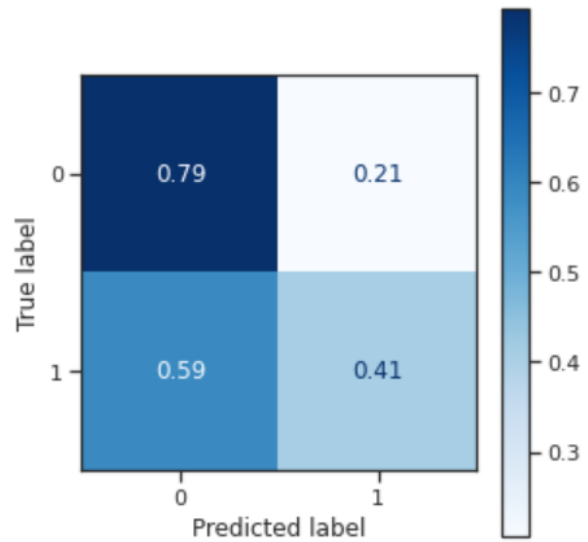
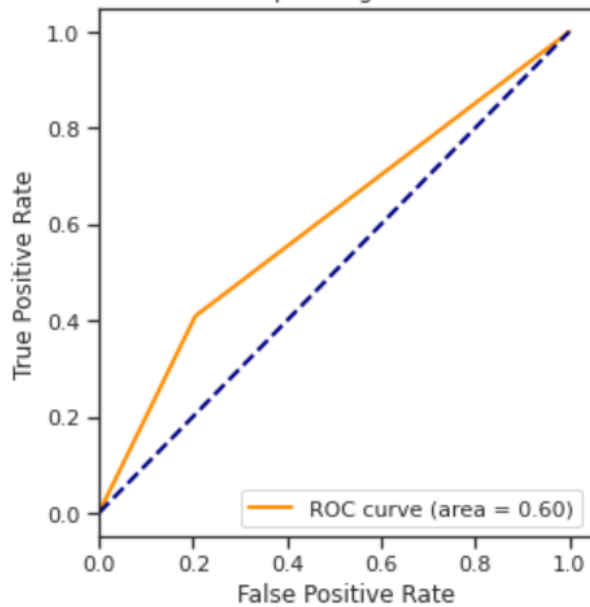
SVC

Receiver operating characteristic

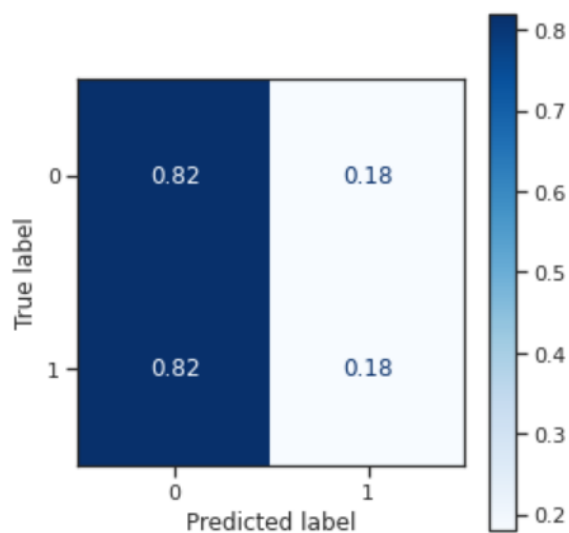
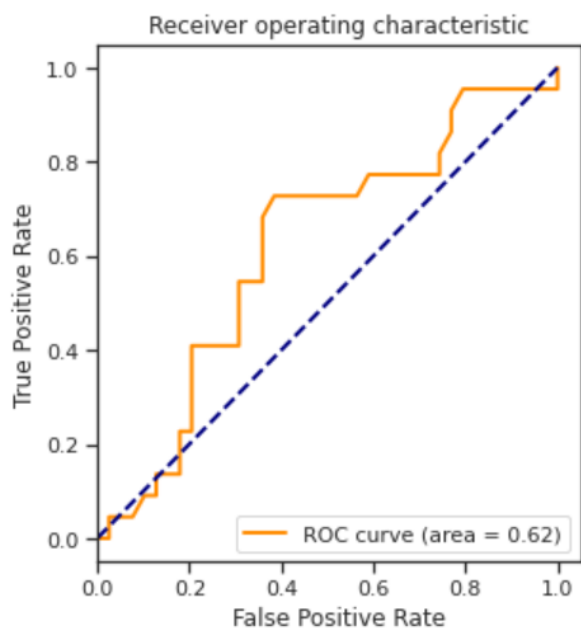


Tree

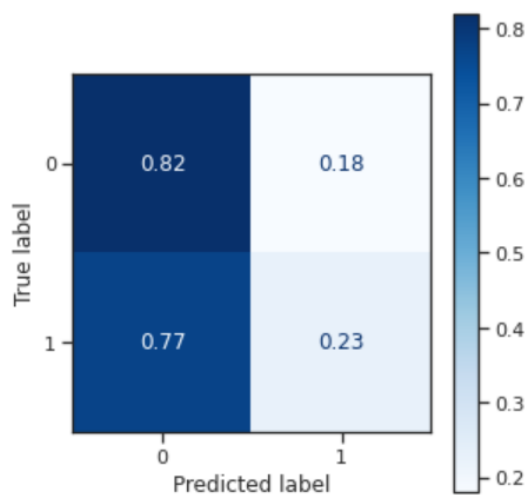
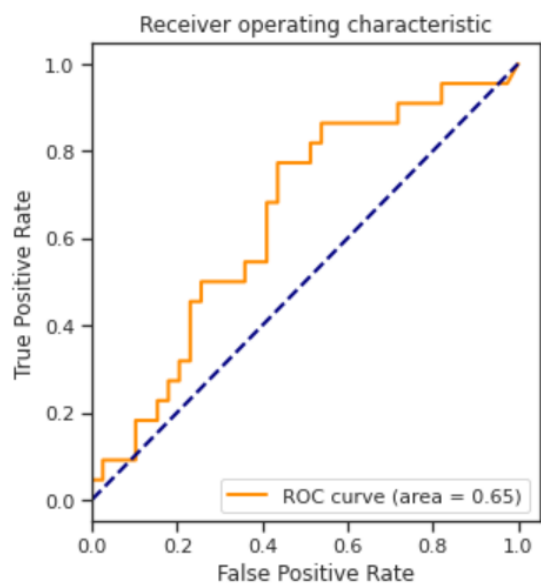
Receiver operating characteristic



RF



GB



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.

```
[ ] # Лучшее значение параметров
    clf_gs.best_params_
```

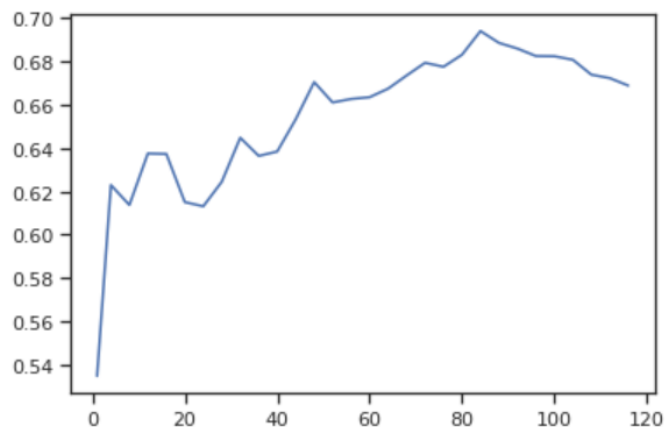
```
{'n_neighbors': 84}
```

```
[ ] clf_gs_best_params_txt = str(clf_gs.best_params_['n_neighbors'])
    clf_gs_best_params_txt
```

```
'84'
```

```
▶ # Изменение качества на тестовой выборке в зависимости от K-соседей
  plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

```
↳ [<matplotlib.lines.Line2D at 0x7f37a4162fd0>]
```



```
▶ clas_X_train.shape
```

```
↳ (242, 4)
```

```
[ ] n_range_list = list(range(0,120,4))
    n_range_list[0] = 1
```

```
[ ] n_range = np.array(n_range_list)
    tuned_parameters = [{'n_neighbors': n_range}]
    tuned_parameters
```

```
[{'n_neighbors': array([ 1,  4,  8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48,
                        52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100,
                        104, 108, 112, 116])}]
```

```
[ ] clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=15, scoring='roc_auc')
    clf_gs.fit(clas_X_train, clas_Y_train)
```

```
GridSearchCV(cv=15, estimator=KNeighborsClassifier(),
             param_grid=[{'n_neighbors': array([ 1,  4,  8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48,
                                                52, 56, 60, 64, 68, 72, 76, 80, 84, 88, 92, 96, 100,
                                                104, 108, 112, 116])}],
             scoring='roc_auc')
```

```
[ ] # Лучшая модель
    clf_gs.best_estimator_
```

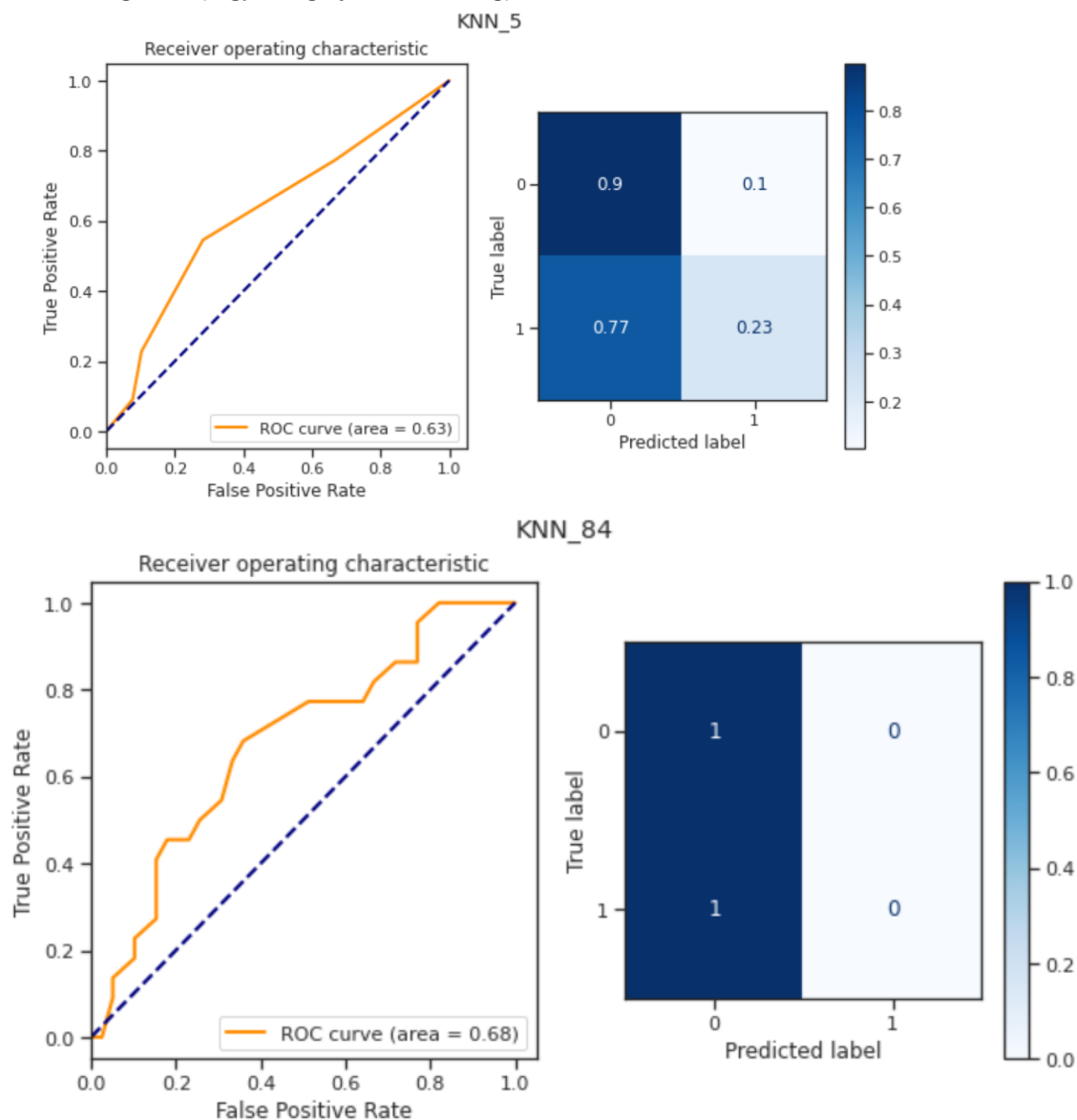
```
KNeighborsClassifier(n_neighbors=84)
```

10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.

```
[ ] clas_models_grid = {'KNN_5': KNeighborsClassifier(n_neighbors=5),  
                        str('KNN_' + clf_gs_best_params_txt): clf_gs.best_estimator_}
```

```
for model_name, model in clas_models_grid.items():  
    clas_train_model(model_name, model, clasMetricLogger)
```

/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated.
warnings.warn(msg, category=FutureWarning)

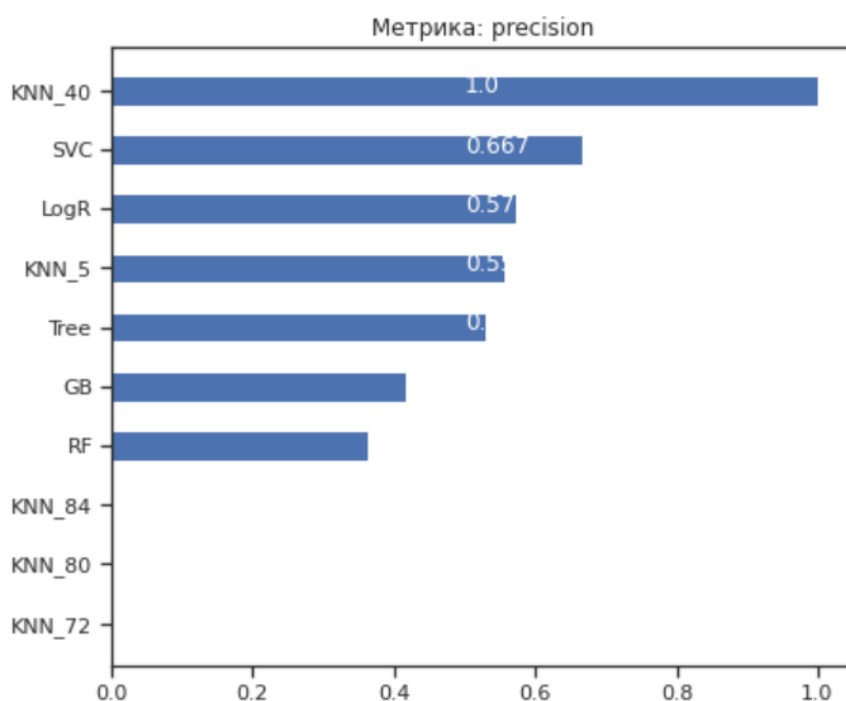


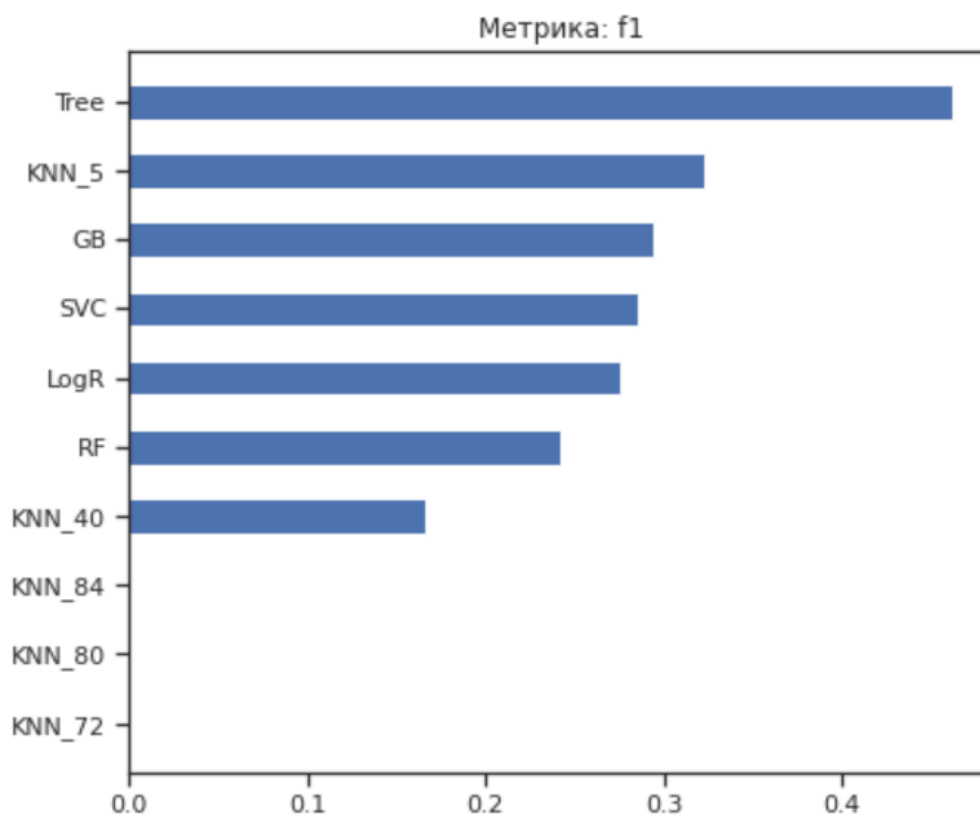
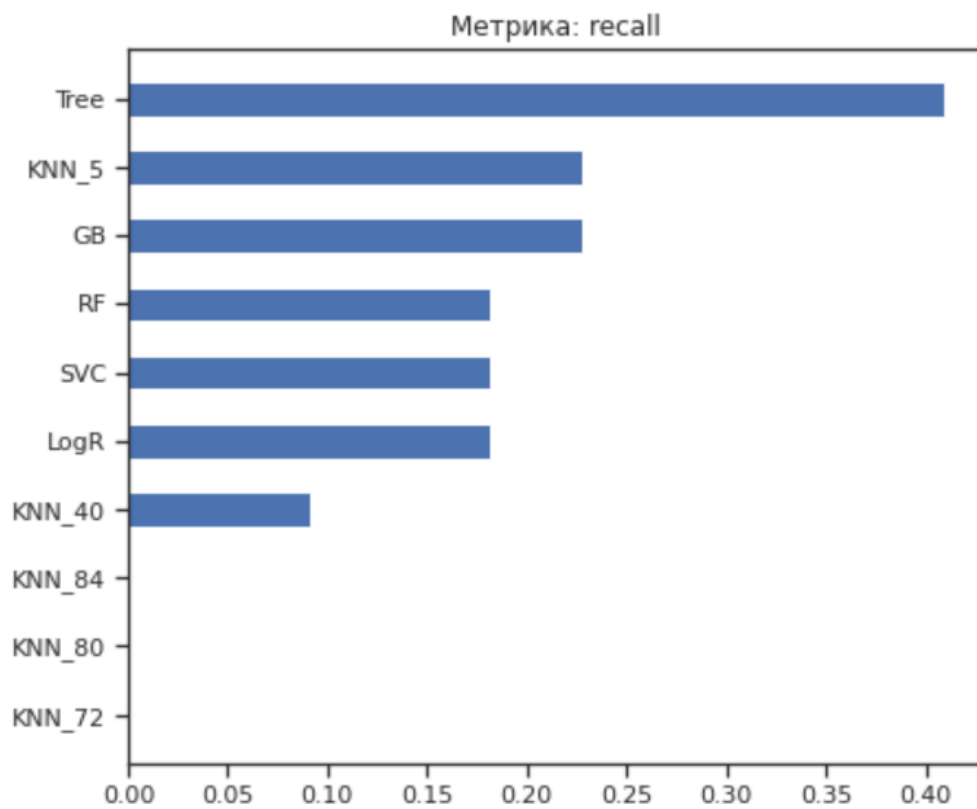
11. Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпараметров на качество моделей и т.д.

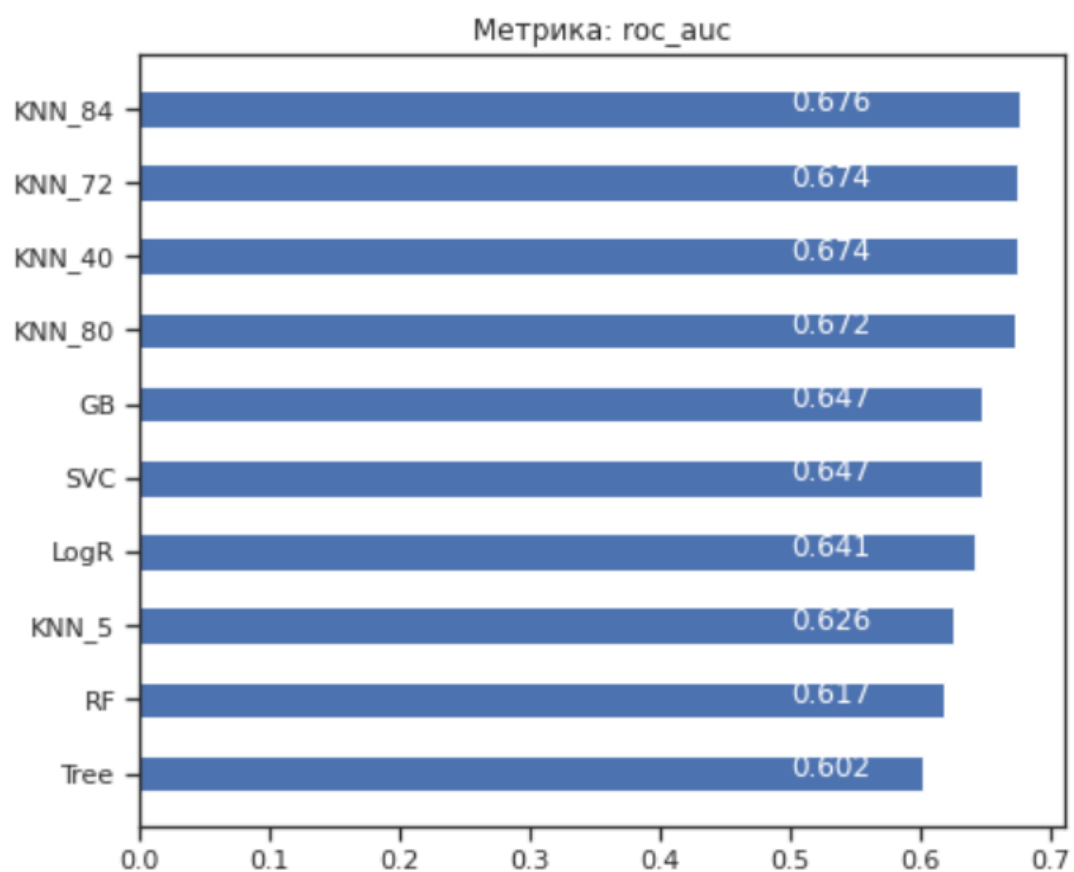
```
[ ] # Метрики качества модели
clas_metrics = clasMetricLogger.df['metric'].unique()
clas_metrics

array(['precision', 'recall', 'f1', 'roc_auc'], dtype=object)
```

```
[ ] # Построим графики метрик качества модели
for metric in clas_metrics:
    clasMetricLogger.plot('Метрика: ' + metric, metric, figsize=(7, 6))
```







Источники:

1. Конспекты лекций с репозитория курса "Технологии машинного обучения", бакалавриат, 6 семестр.
2. Видеотрансляции лекций
<https://youtube.com/playlist?list=PL9vFTJYocFHomDo2q7Cdl6KfsheYJQUdh>