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 (ИУ5)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

__Прогнозирование сердечной недостаточности

_____

_____

_____

_____

_____

_____

Студент _____ _____ **_С.В. Гришин____**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **_Ю.Е. Гапанюк ___**
(Подпись, дата) (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_____прогнозирование сердечной недостаточности _____

_____

_____

Студент группы _ИУ5-61Б_____

_____Гришин Станислав Васильевич _____
(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 *решить задачу бинарной классификации на основе материалов дисциплины по выбранной предметной области* _____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на_27____листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

_____

Дата выдачи задания « _15___» _февраля_____2022 г.

Руководитель НИР _____ ___**Ю.Е. Гапанюк** ____
(Подпись, дата) (И.О.Фамилия)

Студент _____ ___**С.В. Гришин** _____
(Подпись, дата) (И.О.Фамилия)

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

СОДЕРЖАНИЕ

1 Введение

В качестве предметной области был выбран датасет с информацией о сердечной недостаточности. В исследовании будет решаться задача бинарной классификации.

Сердечно-сосудистые заболевания (ССЗ) являются причиной смерти номер 1 во всем мире, унося примерно 17,9 миллиона жизней ежегодно, что составляет 31% всех смертей в мире. Четыре из 5 смертей от сердечно-сосудистых заболеваний связаны с сердечными приступами и инсультами, и одна треть этих смертей происходит преждевременно среди людей в возрасте до 70 лет. Сердечная недостаточность является распространенным явлением, вызванным сердечно-сосудистыми заболеваниями, и этот набор данных содержит 11 признаков, которые можно использовать для прогнозирования возможного заболевания сердца.

Люди с сердечно-сосудистыми заболеваниями или с высоким сердечно-сосудистым риском (из-за наличия одного или нескольких факторов риска, таких как гипертония, диабет, гиперлипидемия или уже установленное заболевание) нуждаются в раннем выявлении и лечении, в чем большую помощь может оказать модель машинного обучения.

2. Описание датасета

В качестве набора данных мы будем использовать набор данных прогнозирования инсульта: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

- Age: возраст пациента [лет]
- Sex: пол пациента [M: Мужской, F: Женский]
- ChestPainType: тип боли в груди [TA: типичная стенокардия, ATA: атипичная стенокардия, NAP: неангинальная боль, ASY: бессимптомная]
- RestingBP: артериальное давление в состоянии покоя [мм рт.ст.]
- Cholesterol: холестерин сыворотки [мм/дл]
- FastingBS: уровень сахара в крови натощак [1: если FastingBS > 120 мг/дл, 0: иначе]
- RestingECG: результаты электрокардиограммы в покое [Normal: нормальная, ST: анома- лия ST-T (инверсия T и/или элевация или депрессия ST > 0,05 мВ), LVH: вероятная или определенная гипертрофия левого желудочка по критериям Эстеса]
- MaxHR: максимальная достигнутая частота сердечных сокращений [Числовое значение от 60 до 202]
- ExerciseAngina: стенокардия, вызванная физической нагрузкой [Y: Да, N: Нет]
- Oldpeak: oldpeak: ST [Числовое значение, измеренное в депрессии]
- ST_Slope: наклон сегмента ST пикового упражнения [Up: восходящий, Flat: плоский, Down: нисходящий]
- HeartDisease: выходной класс [1: болезнь сердца, 0: нормальный]

3. Импорт библиотек

[68]:
```
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
from  sklearn.metrics  import  mean_absolute_error,  mean_squared_error,
 ↪mean_squared_log_error,  median_absolute_error,  r2_score
from sklearn.metrics  import  roc_curve,  roc_auc_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
from sklearn.preprocessing import LabelEncoder
%matplotlib inline
sns.set(style=DticksD)
import warnings
warnings.filterwarnings('ignore')
```

4. Загрузка данных

[2]:
```
#first_data    =    pd.read_csv('healthcare-dataset-stroke-data.csv')
first_data = pd.read_csv('heart.csv')
```

[3]:
```
# Удалим дубликаты записей, если они присутствуют
data = first_data.drop_duplicates()
```

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

Основные характеристики датасета

[4]:
```
data.head()
```

[4]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	\
0	40	M	ATA	140	289	0	Normal	172	
1	49	F	NAP	160	180	0	Normal	156	
2	37	M	ATA	130	283	0	ST	98	
3	48	F	ASY	138	214	0	Normal	108	

```
4   54   M           NAP        150          195          0      Normal     122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N      0.0       Up             0
1              N      1.0     Flat             1
2              N      0.0       Up             0
3              Y      1.5     Flat             1
4              N      0.0       Up             0
```

[5]:
```
data.shape
```

[5]:
```
(918, 12)
```

[6]:
```
# Список колонок
data.columns
```

[6]:
```
Index(['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol', 'FastingBS',
       'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope',
       'HeartDisease'],
      dtype='object')
```

[7]:
```
# Список колонок с типами данных
data.dtypes
```

[7]:
```
: Age              int64
Sex               object
ChestPainType     object
RestingBP          int64
Cholesterol        int64
FastingBS          int64
RestingECG        object
MaxHR              int64
ExerciseAngina    object
Oldpeak          float64
ST_Slope          object
HeartDisease       int64
dtype: object
```

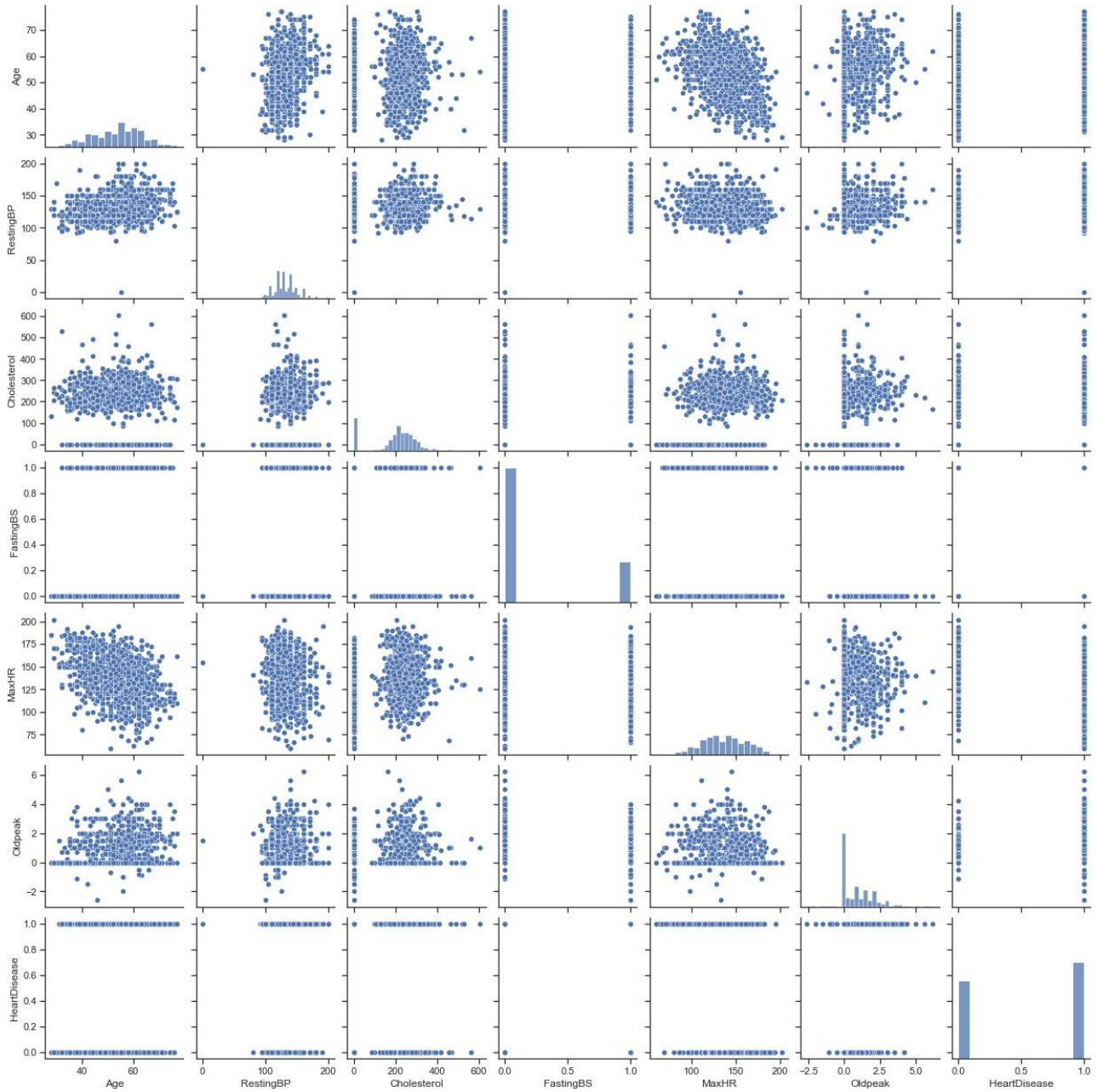
[8]:
```
# Проверим наличие пустых значений
data.isnull().sum()
```

[8]:
```
Age              0
Sex              0
ChestPainType    0
RestingBP        0
Cholesterol      0
FastingBS        0
RestingECG       0
MaxHR            0
ExerciseAngina   0
Oldpeak          0
ST_Slope         0
HeartDisease     0
dtype: int64
```

6. Построение графиков для понимания структуры данных

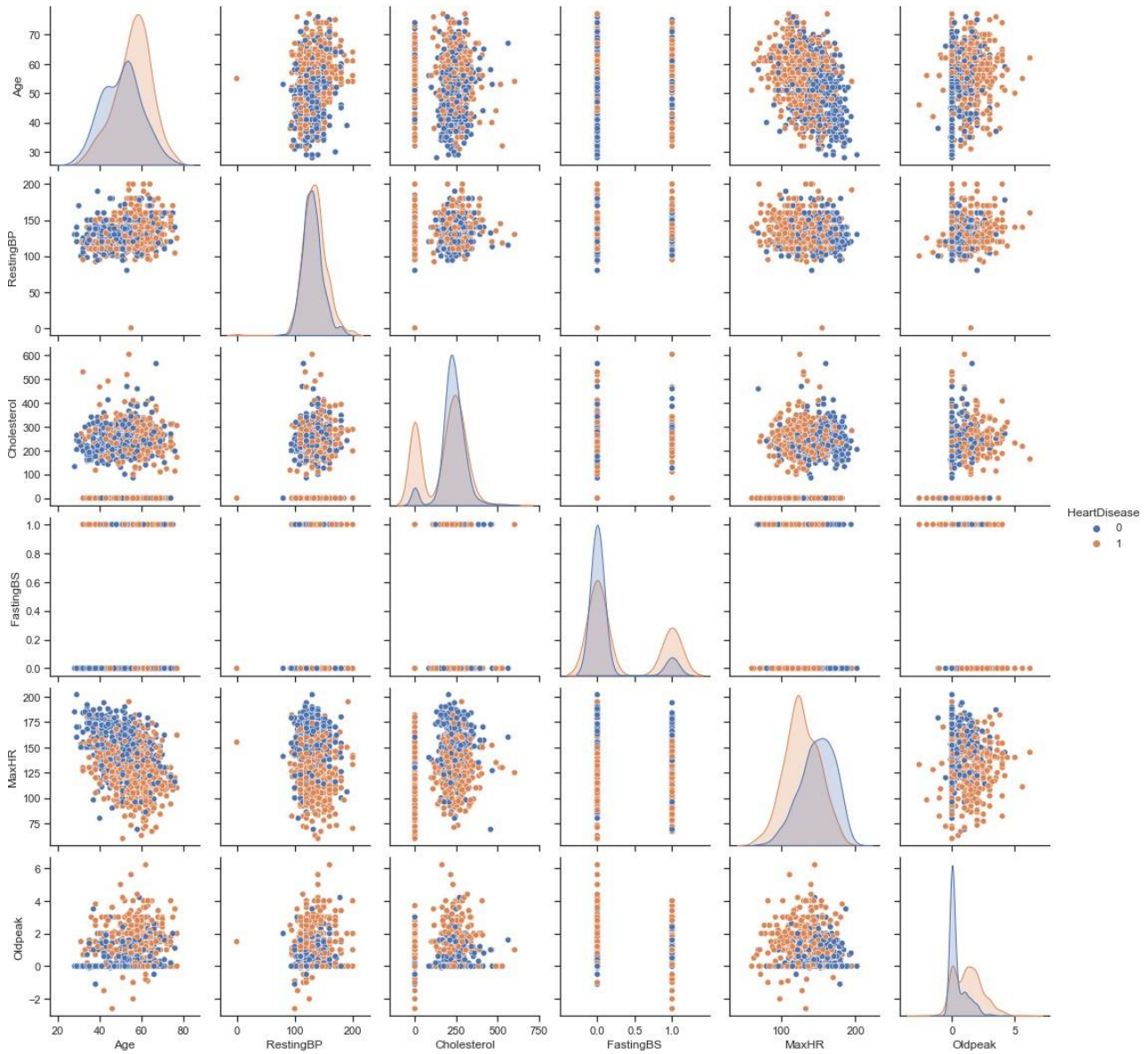
[9]:
```
# Парные диаграммы
sns.pairplot(data)
```

[9]: <seaborn.axisgrid.PairGrid at 0x1dff0c31120>



[10]:
```
sns.pairplot(data, hue=DHeartDiseaseD)
```

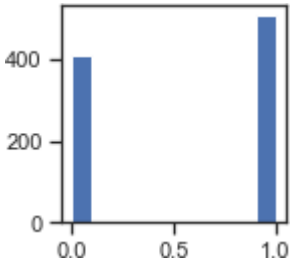
[10]: <seaborn.axisgrid.PairGrid at 0x1dff5523dc0>



[11]:
```
# Убедимся, что целевой признак
# для задачи бинарной классификации содержит только 0 и 1
data['HeartDisease'].unique()
```

[11] :
```
array([0, 1], dtype=int64)
```

[12]:
```
# Оценим дисбаланс классов для stroke
fig, ax = plt.subplots(figsize=(2,2))
plt.hist(data['HeartDisease'])
plt.show()
```



[13]:
```
data['HeartDisease'].value_counts()
```

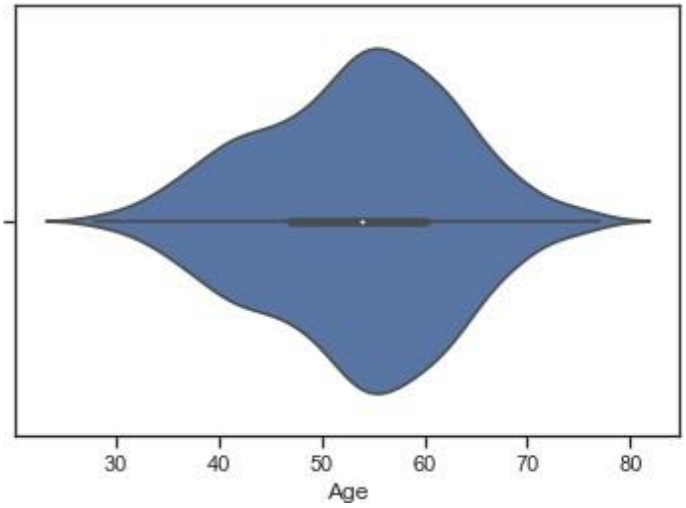
[13]:
```
1    508
0    410
Name: HeartDisease, dtype: int64
```

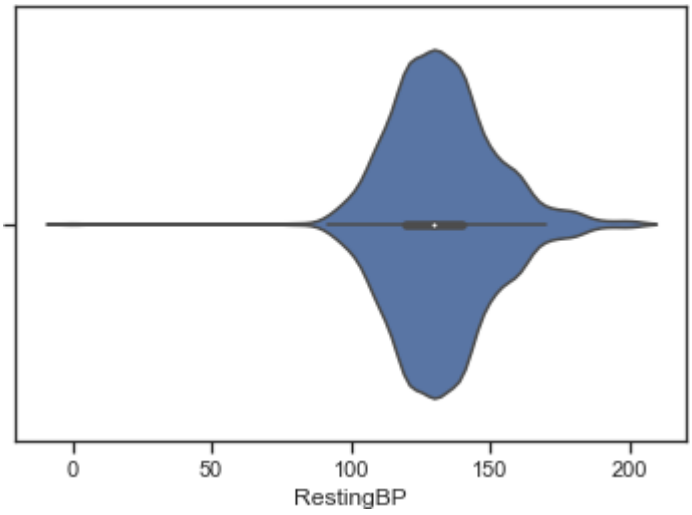
[14]:
```
# посчитаем дисбаланс классов
total = data.shape[0]
class_0, class_1 = data['HeartDisease'].value_counts()
print('Класс 0 составляет {}%, а класс 1 составляет {}%.'
      .format(round(class_0 / total, 4)*100, round(class_1 / total, 4)*100))
```

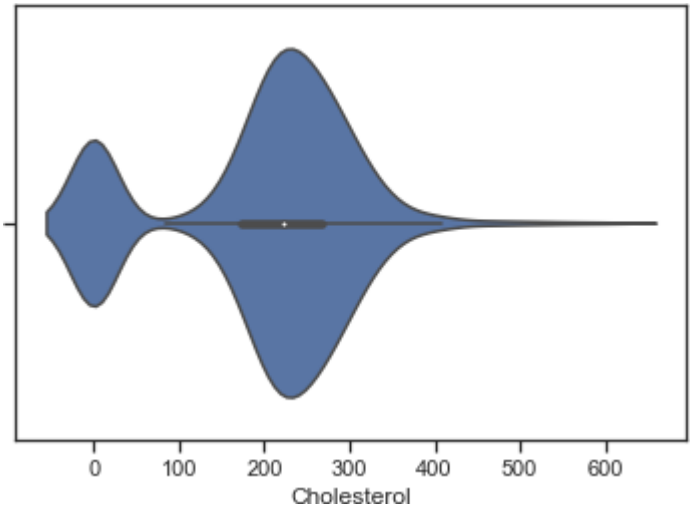
```
Класс 0 составляет 55.34%, а класс 1 составляет 44.66%.
```

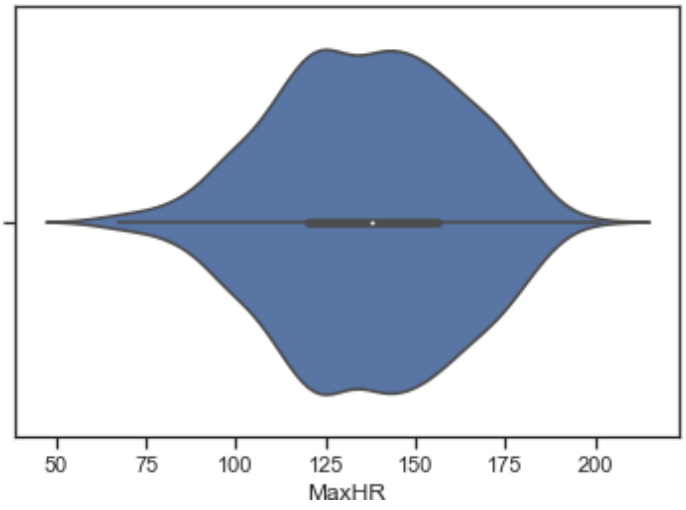
Вывод. Классы практически сбалансированы.

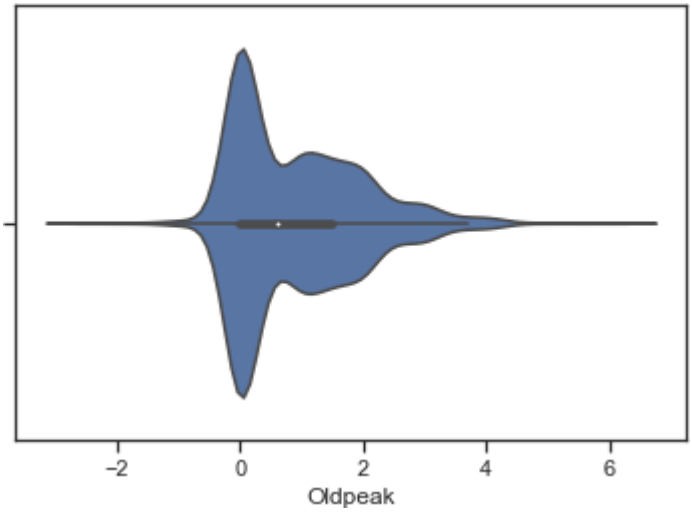
[15]:
```
# Скрипичные диаграммы для числовых колонок
for col in ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']:
    sns.violinplot(x=data[col])
    plt.show()
```




0
50
100
150
200
RestingBP


0
100
200
300
400
500
600
Cholesterol


50
75
100
125
150
175
200
MaxHR



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

[16]:
```
data.dtypes
```

[16]:
```
Age                int64
Sex               object
ChestPainType     object
RestingBP          int64
Cholesterol        int64
FastingBS          int64
RestingECG        object
MaxHR              int64
ExerciseAngina    object
Oldpeak          float64
ST_Slope          object
HeartDisease       int64
dtype: object
```

Категориальные признаки присутствуют, закодируем их.

[17]:
```
data['Sex'].unique()
```

[17]:
```
array(['M', 'F'], dtype=object)
```

[18]:
```

```

[18]:
```
array([0, 1])
```

[19]:
```
data['ChestPainType'].unique()
```

[19]: array(['ATA', 'NAP', 'ASY', 'TA'], dtype=object)

[20]:

[20]: array([0, 1, 2, 3])

[21]:
```
data['RestingECG'].unique()
```

[21]: array(['Normal', 'ST', 'LVH'], dtype=object)

[22]:

[22]: array([0, 1, 2])

[23]:
```
data['ExerciseAngina'].unique()
```

[23]: array(['N', 'Y'], dtype=object)

[24]:

[24]: array([0, 1])

[25]:
```
data['ST_Slope'].unique()
```

[25]: array(['Up', 'Flat', 'Down'], dtype=object)

[26]:

[26]: array([0, 1, 2])

[27]:
```
# Числовые колонки для масштабирования
scale_cols = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
```

[28]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(data[scale_cols])
```

[29]:
```
# 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]
```

[30]:
```
data.head()
```

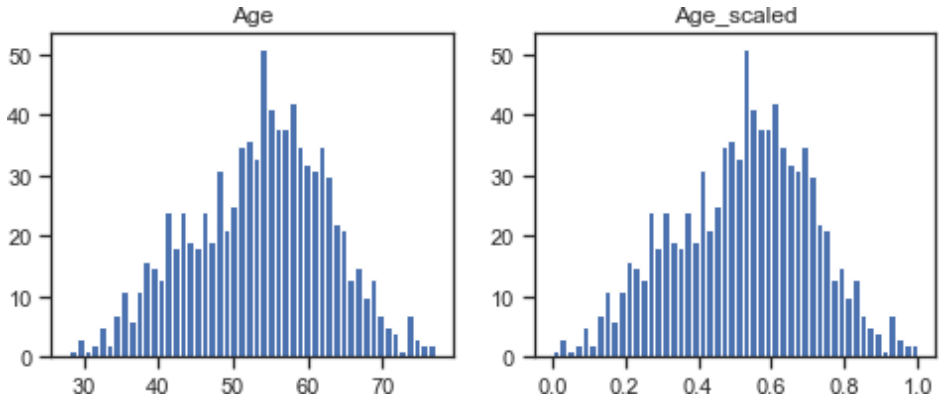
[30]:

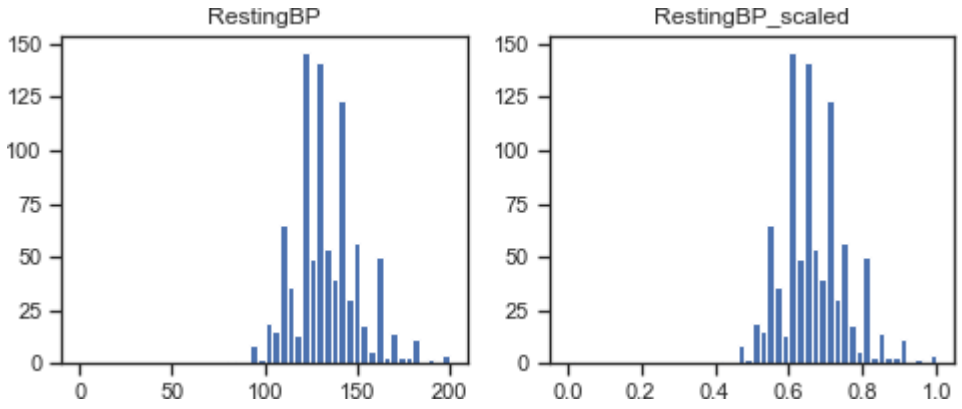
	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	\
0	40	1	1	140	289	0	1	
1	49	0	2	160	180	0	1	
2	37	1	1	130	283	0	2	
3	48	0	0	138	214	0	1	
4	54	1	2	150	195	0	1	

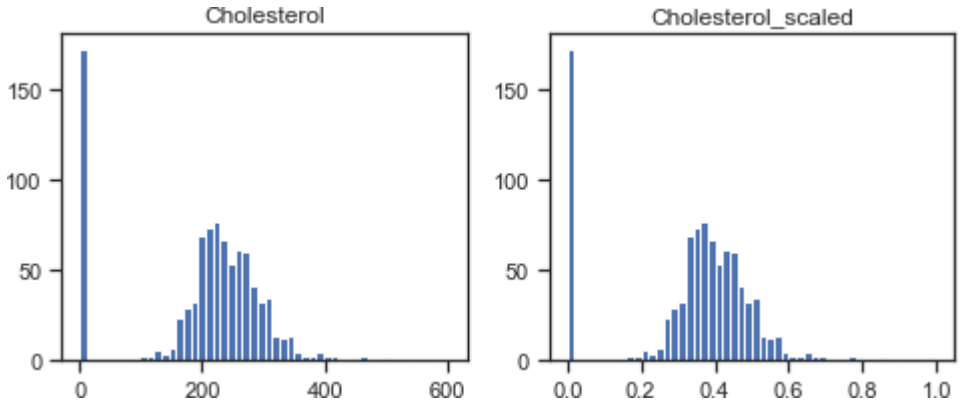
	MaxHR	ExerciseAngina	Ol peak	ST_Slope	HeartDisease	Age_scaled	\
0	172	0	0.0	2	0	0.244898	
1	156	0	1.0	1	1	0.428571	
2	98	0	0.0	2	0	0.183673	
3	108	1	1.5	1	1	0.408163	
4	122	0	0.0	2	0	0.530612	

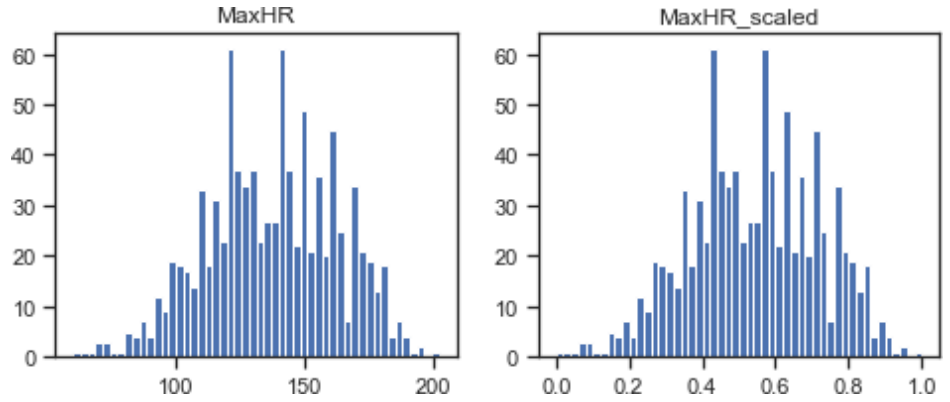
	RestingBP_scaled	Cholesterol_scaled	MaxHR_scaled	Oldpeak_scaled
0	0.70	0.479270	0.788732	0.295455
1	0.80	0.298507	0.676056	0.409091
2	0.65	0.469320	0.267606	0.295455
3	0.69	0.354892	0.338028	0.465909
4	0.75	0.323383	0.436620	0.295455

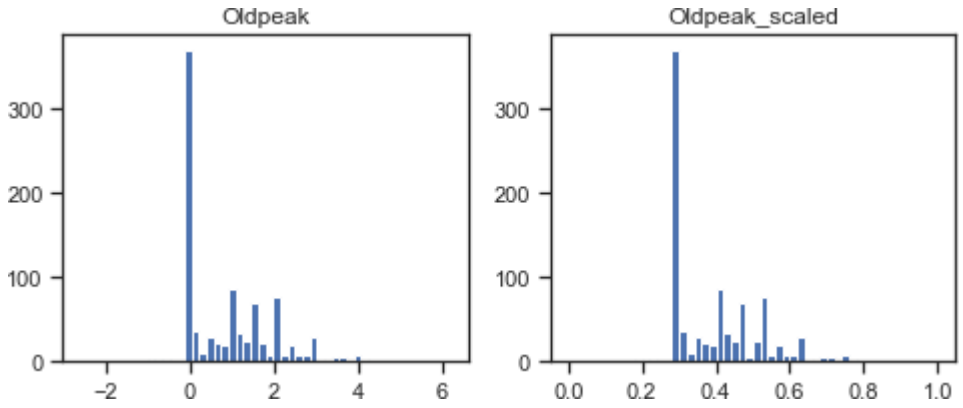
[31]:
```
# Проверим, что масштабирование не повлияло на распределение данных
for col in scale_cols:
    col_scaled = col + '_scaled'

    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()
```




RestingBP
RestingBP_scaled


Cholesterol
Cholesterol_scaled


MaxHR
MaxHR_scaled



8. Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделеймашинного обучения.

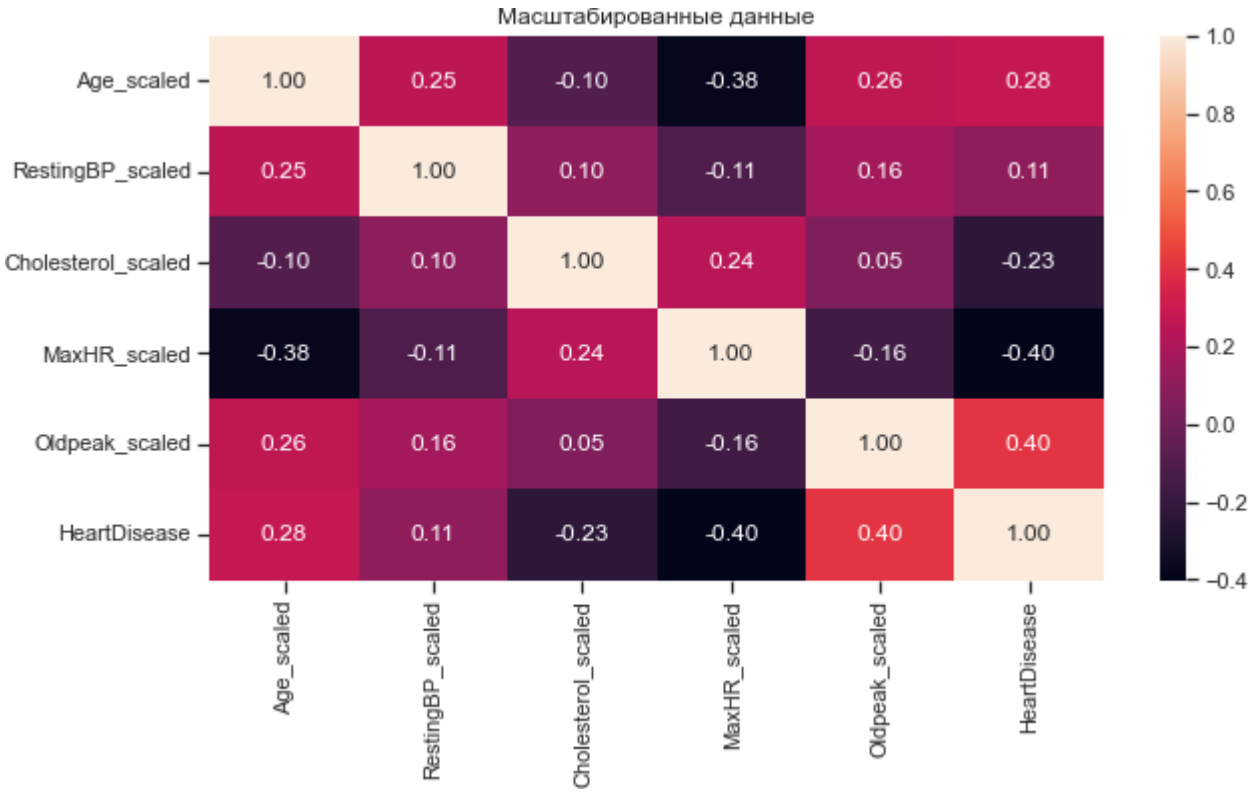
[32]:
```
# Воспользуемся наличием тестовых выборок,
# включив их в корреляционную матрицу
corr_cols_1 = scale_cols + ['HeartDisease']
corr_cols_1
```

[32]: ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease'][

]:
```
scale_cols_postfix = [x+'_scaled' for x in scale_cols]
corr_cols_2 = scale_cols_postfix + ['HeartDisease']
corr_cols_2
```

[34]:
```
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()
```

[35]:
```
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 “HeartDisease” наиболее сильно коррелирует с Oldpeak (0.4) и MaxHR (-0.4). Эти признаки обязательно следует оставить в модели классификации.

9. Выбор метрик для последующей оценки качества моделей.

В качестве метрик для решения задачи классификации будем использовать:Метрики, формируемые на основе матрицы ошибок:

Метрика precision: Доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

Метрика recall (полнота): Доля верно предсказанных классификатором положительных объ- ектов, из всех действительно положительных объектов.

Метрика F1-мера: Для того, чтобы объединить precision и recall в единую метрику исполь- зуется Fβ-мера, которая вычисляется как среднее гармоническое от precision и recall:

Метрика ROC AUC:

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 клас- сификации.

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.

Для получения ROC AUC используется функция roc_auc_score.

10. Сохранение и визуализация метрик

Разработаем класс, который позволит сохранять метрики качества построенных моделей и реализует визуализацию метрик качества.

[37]:
```
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()
```

11. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии.

Для задачи классификации будем использовать следующие модели:

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

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

[53]:
```
X_train, X_test, y_train, y_test = train_test_split(data, data.HeartDisease,
↪random_state=1)
```

[54]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

[54]: ((688, 17), (688,), (230, 17), (230,))

13. 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.

[55]:
```
# Модели
clas_models = {'LogR': LogisticRegression(),
               'KNN_5':KNeighborsClassifier(n_neighbors=5),
               'SVC':SVC(probability=True),
               'Tree':DecisionTreeClassifier(),
               'RF':RandomForestClassifier(),
               'GB':GradientBoostingClassifier()}
```

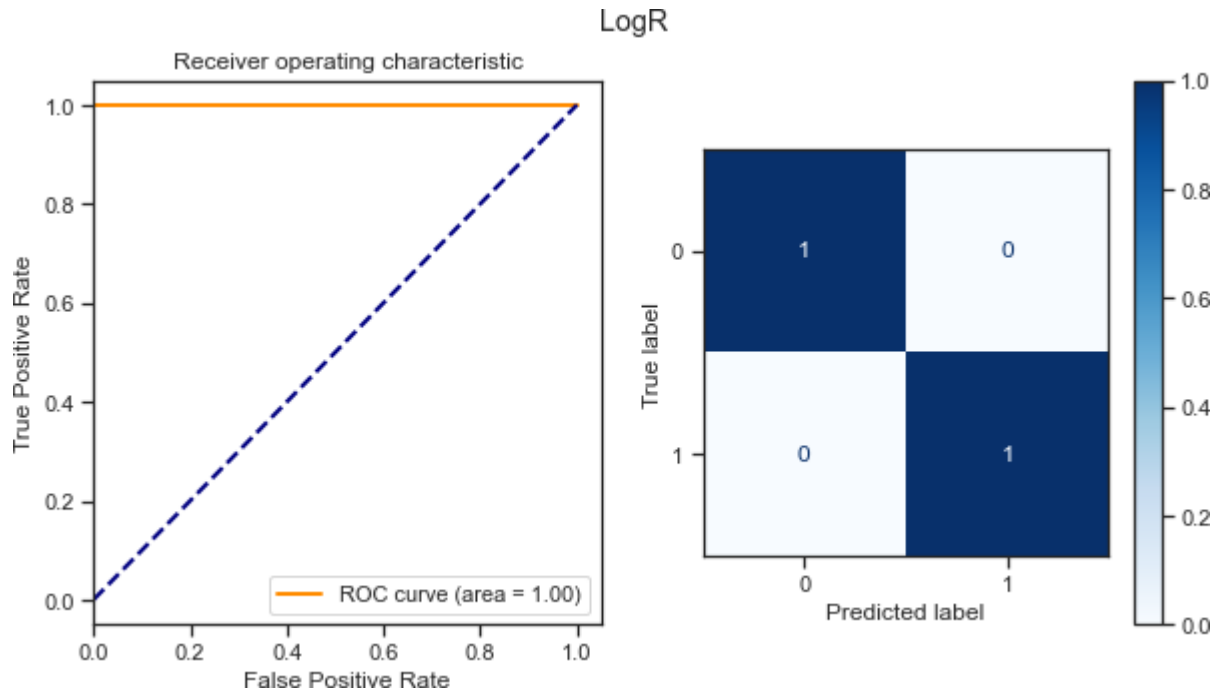
[56]:
```
# Сохранение метрик
clasMetricLogger = MetricLogger()
```

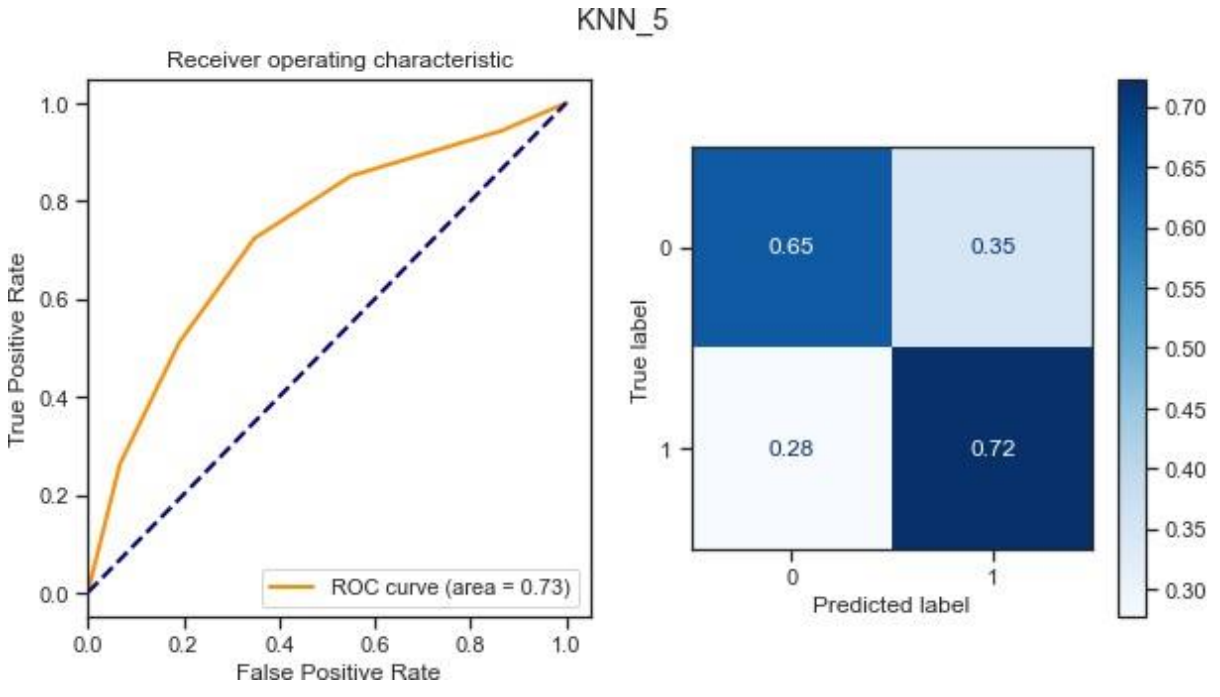
[60]:
```
# Отрисовка ROC-кривой
def draw_roc_curve(y_true, y_score, ax, pos_label=1, average='micro'):
    fpr, tpr, thresholds = roc_curve(y_true, y_score,
                                     pos_label=pos_label)
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    #plt.figure()
    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=Dlower rightD)
```

[66]:
```
def clas_train_model(model_name, model, clasMetricLogger):
    model.fit(X_train, y_train)
    # Предсказание значений
    Y_pred = model.predict(X_test)
    # Предсказание вероятности класса "1" для roc auc
    Y_pred_proba_temp = model.predict_proba(X_test)
    Y_pred_proba = Y_pred_proba_temp[:,1]

    precision = precision_score(y_test.values, Y_pred)
    recall = recall_score(y_test.values, Y_pred)
    f1 = f1_score(y_test.values, Y_pred)
    roc_auc = roc_auc_score(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))
```

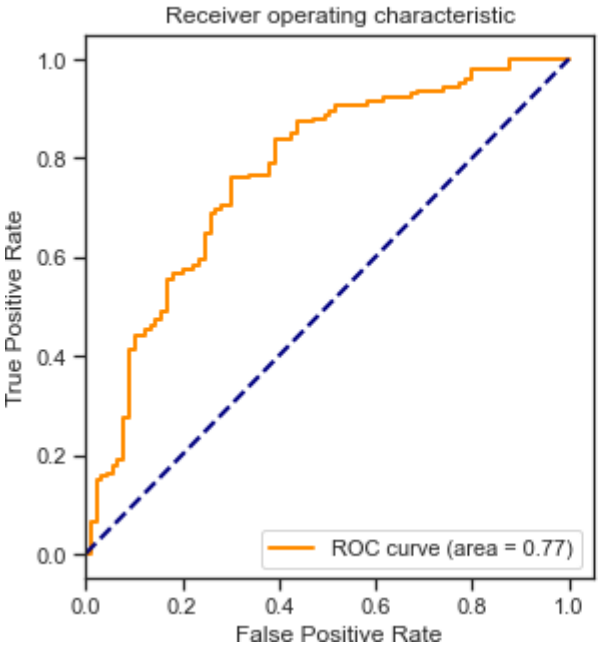
```
    draw_roc_curve(y_test.values, Y_pred_proba, ax[0])
    plot_confusion_matrix(model, X_test, y_test.values, ax=ax[1],
                          display_labels=['0','1'],
                          cmap=plt.cm.Blues, normalize='true')
    fig.suptitle(model_name)
    plt.show()
```

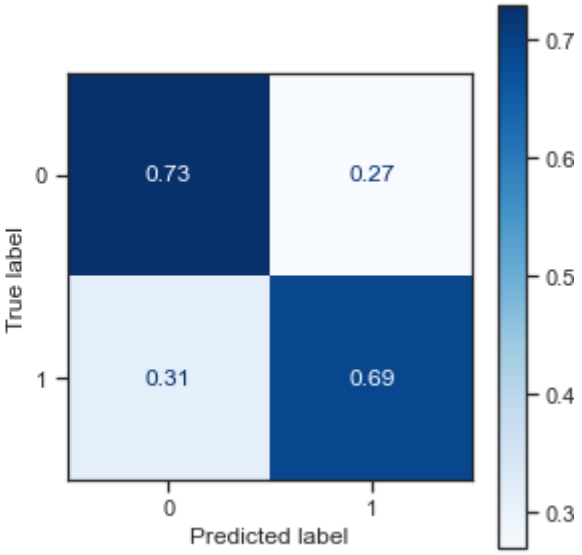
[69]:
```
for model_name, model in clas_models.items():
    clas_train_model(model_name, model, clasMetricLogger)
```



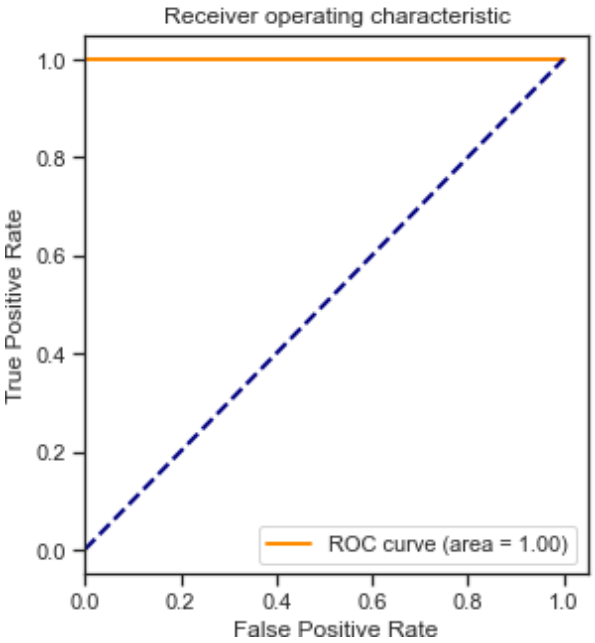


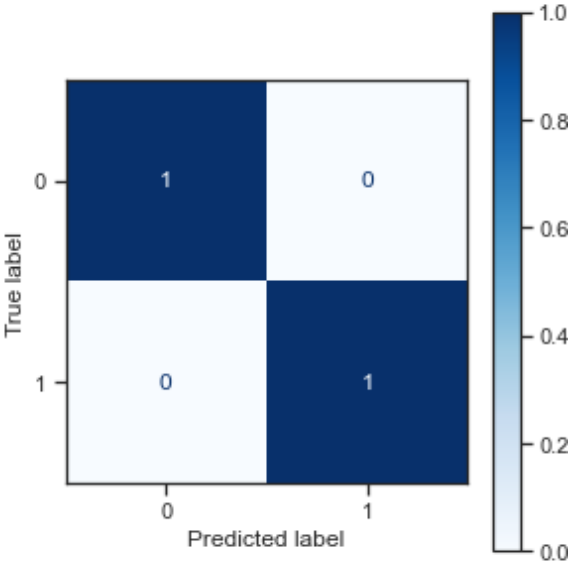
SVC


Receiver operating characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 0.77)
0.0
0.2
0.4
0.6
0.8
1.0

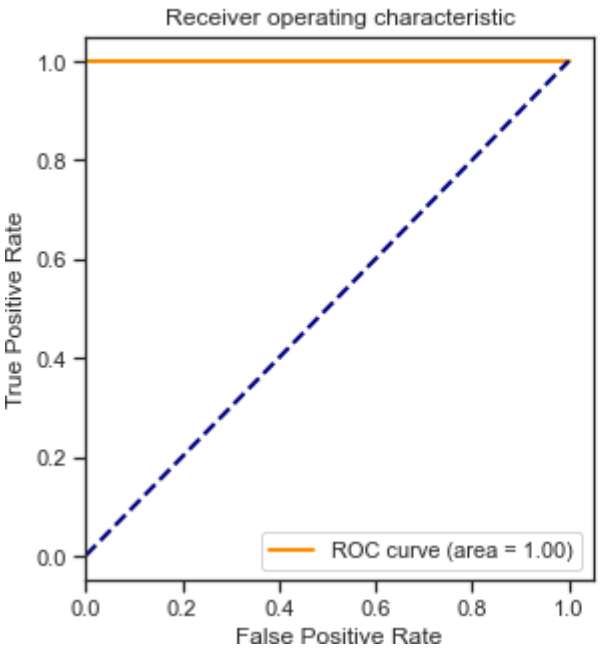

0.73
0.27
0.31
0.69
True label
Predicted label
0
1
0.7
0.6
0.5
0.4
0.3

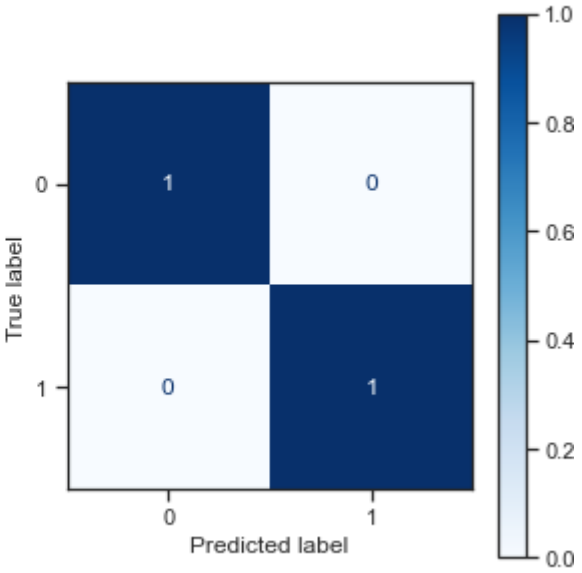
Tree


Receiver operating characteristic
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

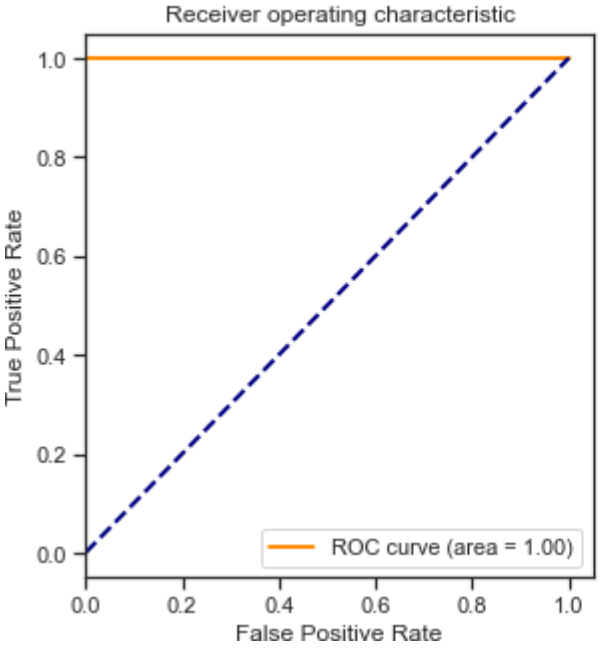

1
0
0
1
True label
Predicted label
0
1
1.0
0.8
0.6
0.4
0.2
0.0

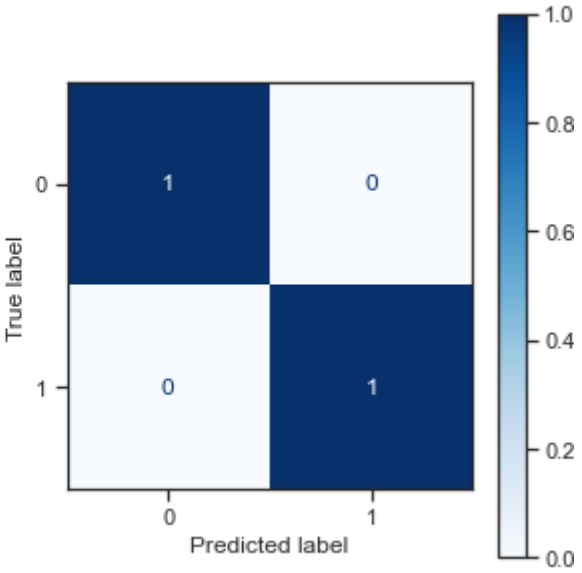
RF





GB





Логистическая регрессия

14. 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.

[70]:
```
X_train.shape
```

[70]: (688, 17)

[71]:
```
n_range_list = list(range(0,1250,50))
n_range_list[0] = 1
```

[72]:
```
n_range = np.array(n_range_list)
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

[72]:
```
[{'n_neighbors': array([   1,   50,  100,  150,  200,  250,  300,  350,  400,
450,  500,
         550,  600,  650,  700,  750,  800,  850,  900,  950, 1000, 1050,
        1100, 1150, 1200])}]
```

[74]:
```
%%time
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5,
 ↪scoring='roc_auc')
clf_gs.fit(X_train, y_train)
```

```
Wall time: 2.24 s
```

[74]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid=[{'n_neighbors': array([   1,   50,  100,  150,  200,
250,  300,  350,  400,  450,  500,
        550,  600,  650,  700,  750,  800,  850,  900,  950, 1000, 1050,
       1100, 1150, 1200])}],
             scoring='roc_auc')
```

[75]:
```
# Лучшая модель
clf_gs.best_estimator_
```

[75]:
```
KNeighborsClassifier(n_neighbors=100)
```

[76]:
```
# Лучшее значение параметров
clf_gs.best_params_
```

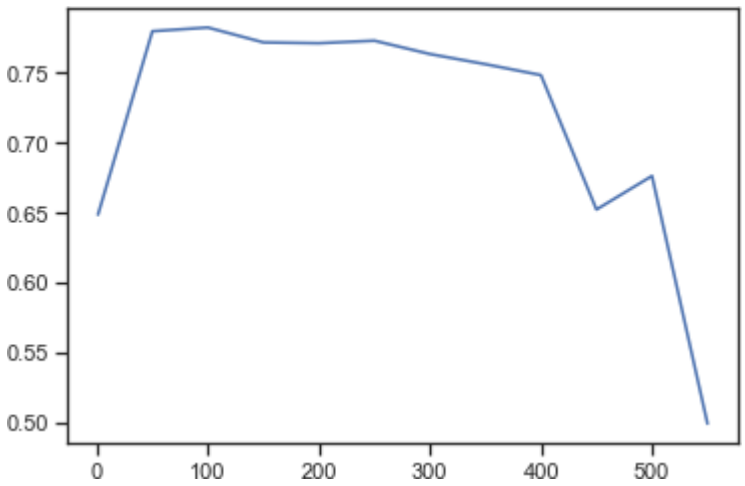
[76]:
```
{'n_neighbors': 100}
```

[77]:
```
clf_gs_best_params_txt = str(clf_gs.best_params_['n_neighbors'])
clf_gs_best_params_txt
```

[77]:
```
'100'
```

[78]:
```
# Изменение качества на тестовой выборке в зависимости от K-соседей
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

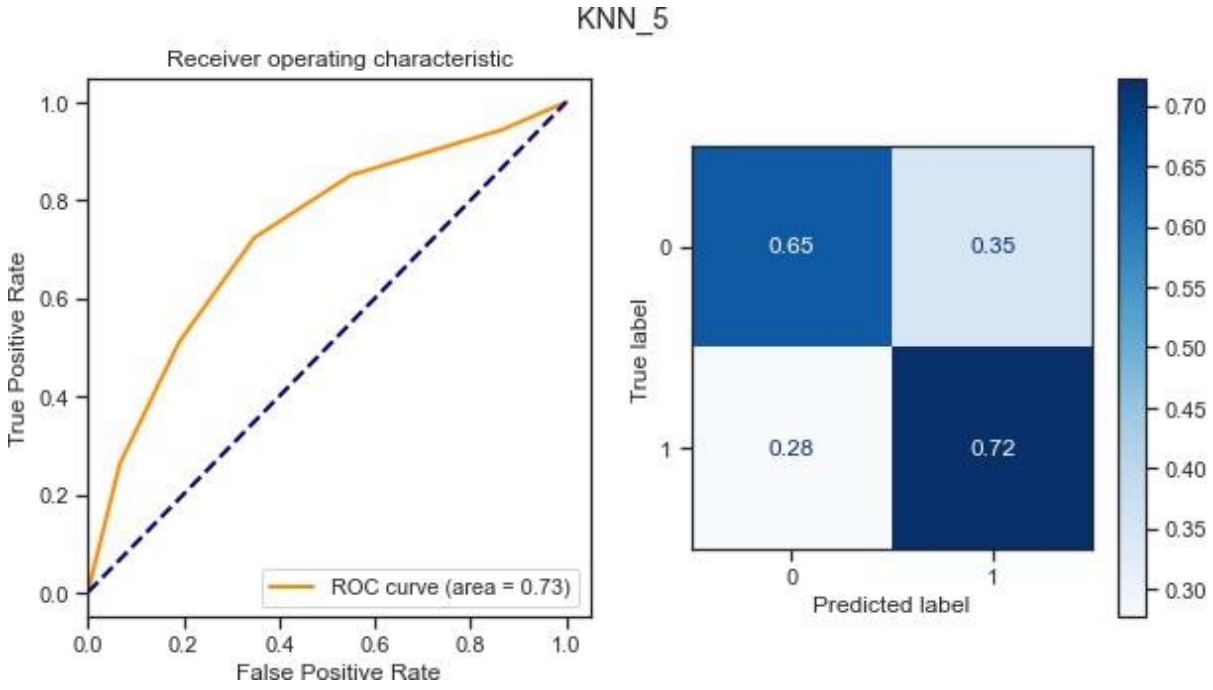
[78]:
```
[<matplotlib.lines.Line2D at 0x1dffa355420>]
```

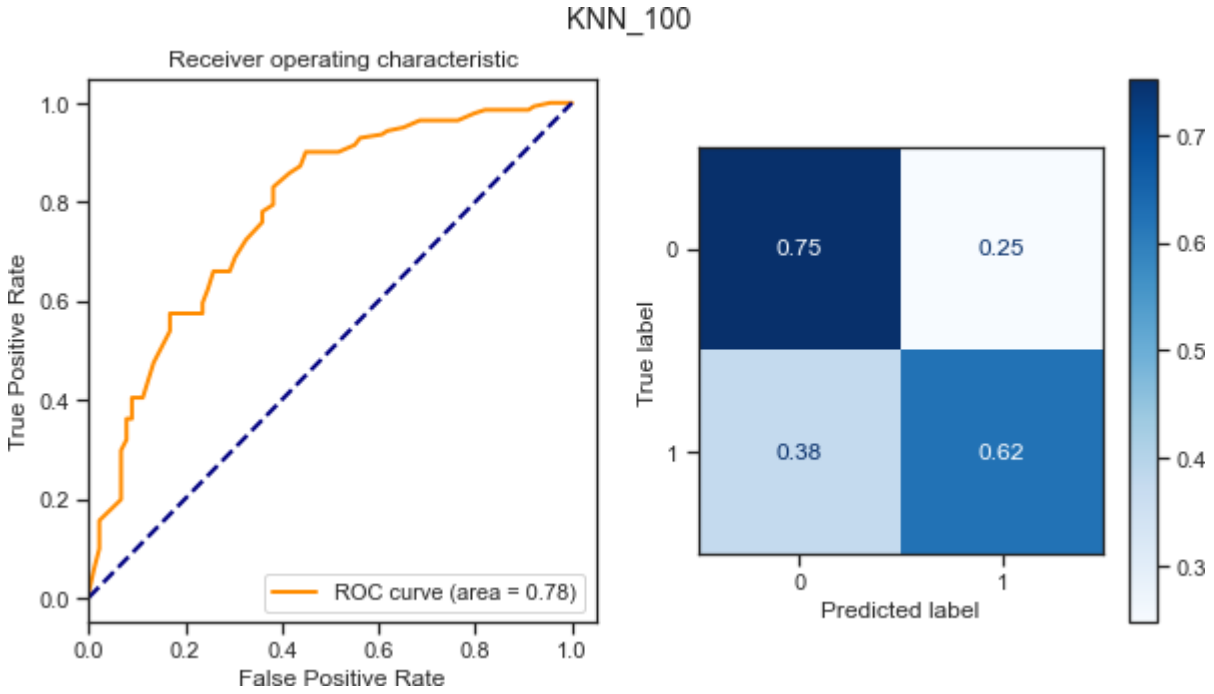


15. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.

[79]:
```
clas_models_grid = {'KNN_5':KNeighborsClassifier(n_neighbors=5),
                    str('KNN_' + clf_gs_best_params_txt):clf_gs.best_estimator_}
```

[80]:
```
for model_name, model in clas_models_grid.items():
    clas_train_model(model_name, model, clasMetricLogger)
```



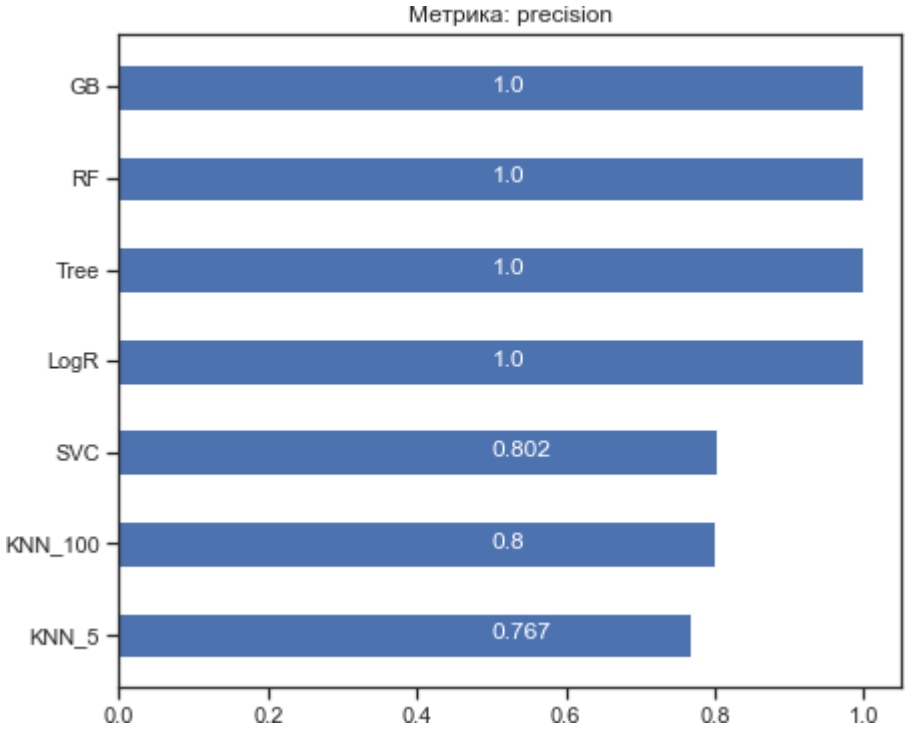


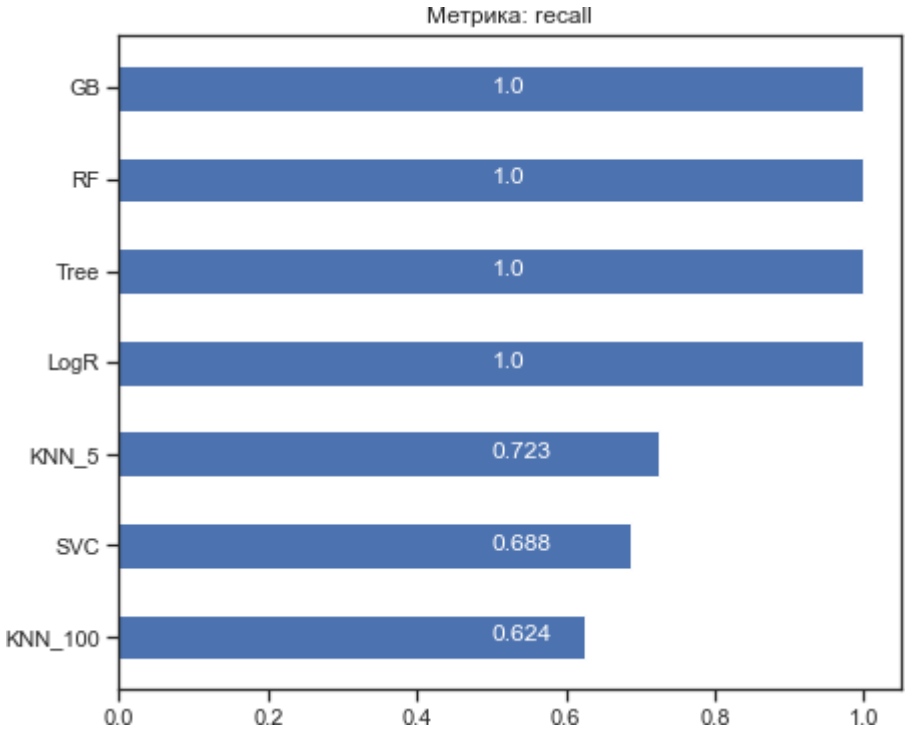
16. Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводыв форме текстового описания.

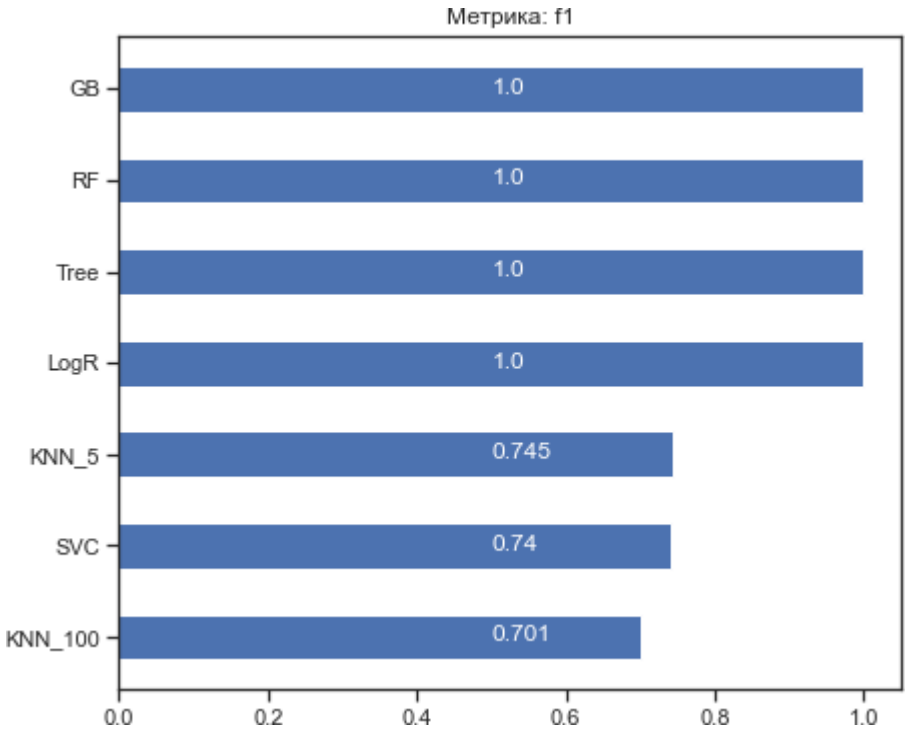
[82]:
```
# Метрики качества модели
clas_metrics = clasMetricLogger.df['metric'].unique()
clas_metrics
```

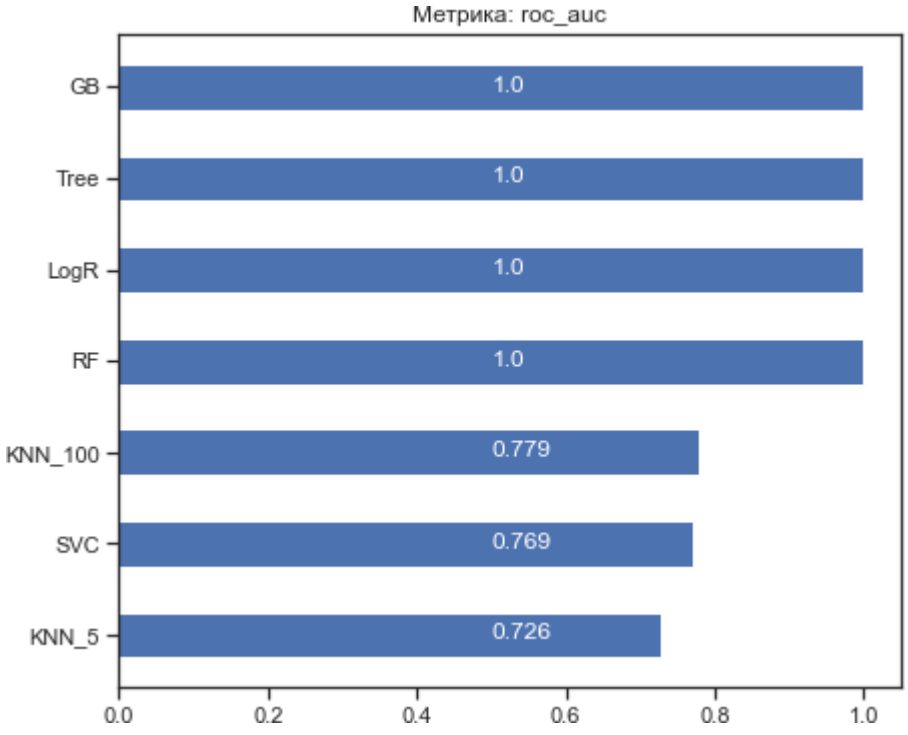
[82]: array(['precision', 'recall', 'f1', 'roc_auc'], dtype=object)[83]:

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


Метрика: precision
GB
RF
Tree
LogR
SVC
KNN_100
KNN_5
1.0
1.0
1.0
1.0
0.802
0.8
0.767
0.0
0.2
0.4
0.6
0.8
1.0


Метрика: recall
GB
RF
Tree
LogR
KNN_5
SVC
KNN_100
1.0
1.0
1.0
1.0
0.723
0.688
0.624
0.0
0.2
0.4
0.6
0.8
1.0





Вывод: 4 модели: градиентный бустинг, дерево, логистическая регрессия и случайный лес показали одинаково высокий результат

17. Заключение

Таким образом, было проведено исследование датасета для прогноза сердечной недостаточности. Для задачи классификации использовалось несколько моделей, из которых градиентный бустинг, дерево, логистическая регрессия и случайный лес показали одинаково высокий результат.

18. Список использованных источников информации

1. Методические указания по программной библиотеке Pandas на языке Python.
2. https://scikit-learn.org/stable/index.html
3. https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html
4. https://github.com/ugapanyuk/ml_course_2022/wiki/COURSE_TMO