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 ___ИУ5 «Системы обработки информации и управления»_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Решение задачи машинного обучения

Студент группы _ИУ5-61Б__ _____ _____Сукиасян В.М.
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____Гапанюк Ю.Е.
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2020 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 курсового проекта

по дисциплине «Технологии машинного обучения» _____

Студент группы ИУ5-61Б_____

_____Сукиасян Владимир Мартунович_____
(Фамилия, имя, отчество)

Тема курсового проекта _____
_____
_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_____
Источник тематики (кафедра, предприятие, НИР) _____

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание: решение задачи машинного обучения на основе материалов дисциплины. Выполняется студентом единолично.
_____
_____
_____
_____

Оформление курсового проекта:

Расчетно-пояснительная записка на __27___ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____
_____
_____
_____

Дата выдачи задания « 12 » февраля 2020 г.

Руководитель курсового проекта _____ _____Гапанюк Ю.Е.
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____Сукиасян В.М.
(Подпись, дата) (И.О.Фамилия)

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

Оглавление

1. Задание 3
2. Введение 5
3. Основная часть 6
   Постановка задачи 6
   Описание набора данных 6
   Ход работы 7
4. Метрики качества модели 23
5. Выводы 25
6. Список использованных источников 27

Задание

Схема типового исследования, проводимого студентом в рамках курсовой работы, содержит выполнение следующих шагов:

- 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.
- Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
- Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
- 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.
- Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
- Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
- Формирование обучающей и тестовой выборок на основе исходного набора данных.
- Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производятся обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.
- 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.

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

Введение

Курсовой проект – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного

обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

Курсовой проект опирается на знания, умения и владения, полученные студентом в рамках лекций и лабораторных работ по дисциплине.

В рамках данной курсовой работы необходимо применить навыки, полученные в течение курса «Технологии машинного обучения», и обосновать полученные результаты.

Основная часть

Постановка задачи

В данной курсовой работе ставится задача определения пригодности гриба в употребление по внешним параметрам с помощью методов машинного обучения «Stochastic gradient descent», «Support vector machine», «Метод ближайших соседей», «Gradient boosting» и «Random forest».

Описание набора данных

В качестве набора данных мы будем использовать набор данных по обнаружению сердечного заболевания у пациента.
Файл содержит следующие колонки:
1. age - возраст пациента;
2. sex - пол пациента (1 = мужчина; 0 = женщина);
3. chest pain type (4 values) - тип боли в груди пациента;
4. resting blood pressure - кровяное давление в покое;
5. serum cholestoral in mg/dl - содержание холестирина;
6. fasting blood sugar > 120 mg/dl - уровень сахара в крови натощак;
7. resting electrocardiographic results (values 0,1,2) - результаты электрокардиографии в покое;
8. maximum heart rate achieved - максимальная частота сердечных сокращений;
9. exercise induced angina - стенокардия, вызванная физической нагрузкой;
10.oldpeak = ST depression induced by exercise relative to rest - показание на

электрокардиограмме;
11.the slope of the peak exercise ST segment - показание на электрокардиограмме;
12.number of major vessels (0-3) colored by flourosopy - количество крупных сосудов;
13.thal: 3 = normal; 6 = fixed defect; 7 = reversable defect - анализ из крови;
14.target - наличие или отсутствие сердечного заболевания у пациента

Будем решать задачу классификации. В качестве целевого признака возьмем колонку sex. Поскольку она содержит только значения 0 или 1, то это задача бинарной классификации.

Ход работы

Импортируем необходимые для работы библиотеки:

```
import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, precision_score, recall_score, accuracy_score, plot_
confusion_matrix, roc_curve
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier


%matplotlib inline

# Установим тип графиков
sns.set(style="ticks")

# Для лучшего качество графиков
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")

# Установим ширину экрана для отчета
pd.set_option("display.width", 80)
```

```
data = pd.read_csv('heart.csv', sep=",")
```

Считываем набор данных:Размер датасета:

Размер датасета:

```
data.shape
```

```
(303, 14)
```

Первые пять строк датасета:

Типы колонок:

```
# Первые 5 строк датасета
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
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

Типы колонок:

```
# Поймем какими типами данных заполнены колонки
data.dtypes
```

```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

Названия колонок:

```
#Увидим, из каких колонок состоит датасет
data.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
# Проверим наличие пустых значений
data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Проверка набора данных на пропуски:

Пропущенных значений в наборе данных нет.

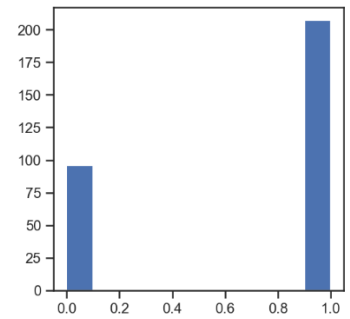
Убедимся, что целевой признак для задачи бинарной классификации содержит только 0 и 1

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

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

Оценим дисбаланс классов для Heart

```
# Оценим дисбаланс классов для Heart
fig, ax = plt.subplots(figsize=(4,4))
plt.hist(data['sex'])
plt.show()
```




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

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

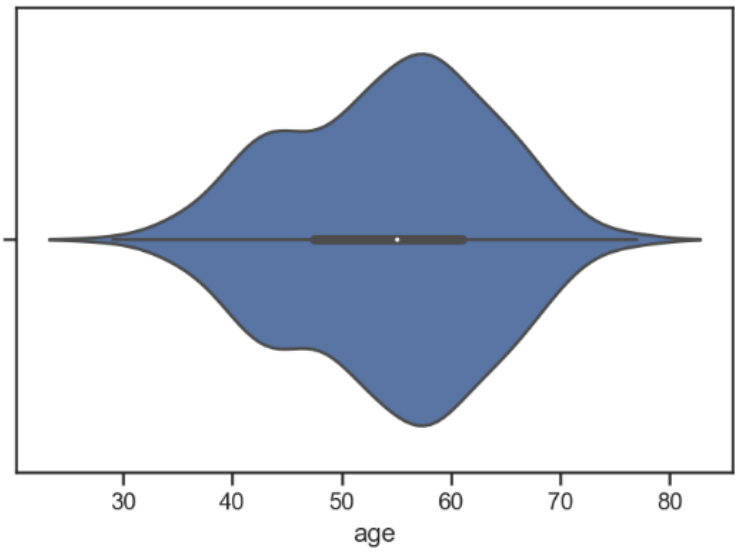
Посчитаем дисбаланс классов:

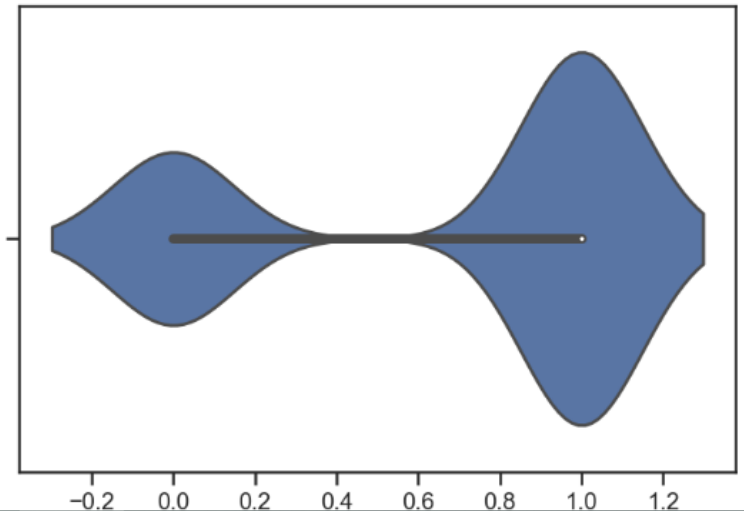
```
# посчитаем дисбаланс классов
total = data.shape[0]
class_1, class_0 = data['exang'].value_counts()
print('Класс 0 - женщина составляет {}%, а класс 1 - мужчина составляет {}%.'
      .format(round(class_0 / total, 2)*100, round(class_1 / total, 2)*100))
```

```
Класс 0 - женщина составляет 33.0%, а класс 1 - мужчина составляет 67.0%.
```

Скрипичные диаграммы для некоторых колонок

```
# Скрипичные диаграммы для некоторых колонок
for col in ['age', 'sex', 'chol', 'thalach', 'thal']:
    sns.violinplot(x=data[col])
    plt.show()
```





Проведем операции, связанные с масштабированием данных

```
# Числовые колонки для масштабирования
scale_cols = ['age', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope',
'ca', 'thal', 'target']
```

```
sc = MinMaxScaler()
sc_data = sc.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] = sc_data[:,i]
```

```
#Выведем первые 5 строк и убедимся, что масштабированные данные были успешно добавлены в набор
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	...	chol_scaled	fbs_scaled	restecg_scaled	thalach_
0	63	1	3	145	233	1	0	150	0	2.3	...	0.244292	1.0	0.0	0.603053
1	37	1	2	130	250	0	1	187	0	3.5	...	0.283105	0.0	0.5	0.885496
2	41	0	1	130	204	0	0	172	0	1.4	...	0.178082	0.0	0.0	0.770992
3	56	1	1	120	236	0	1	178	0	0.8	...	0.251142	0.0	0.5	0.816794
4	57	0	0	120	354	0	1	163	1	0.6	...	0.520548	0.0	0.5	0.702290

5 rows × 28 columns

Выведем последние 5 строк:

```
#Выведем последние 5 строк
data.tail()
```

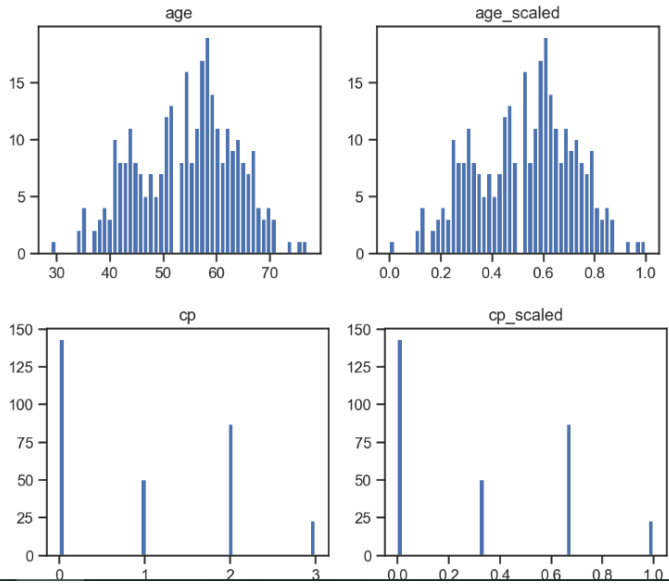
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	...	chol_scaled	fbs_scaled	restecg_scaled	thalac
298	57	0	0	140	241	0	1	123	1	0.2	...	0.262557	0.0	0.5	0.3969
299	45	1	3	110	264	0	1	132	0	1.2	...	0.315068	0.0	0.5	0.4656
300	68	1	0	144	193	1	1	141	0	3.4	...	0.152968	1.0	0.5	0.5343
301	57	1	0	130	131	0	1	115	1	1.2	...	0.011416	0.0	0.5	0.3358
302	57	0	1	130	236	0	0	174	0	0.0	...	0.251142	0.0	0.0	0.7862

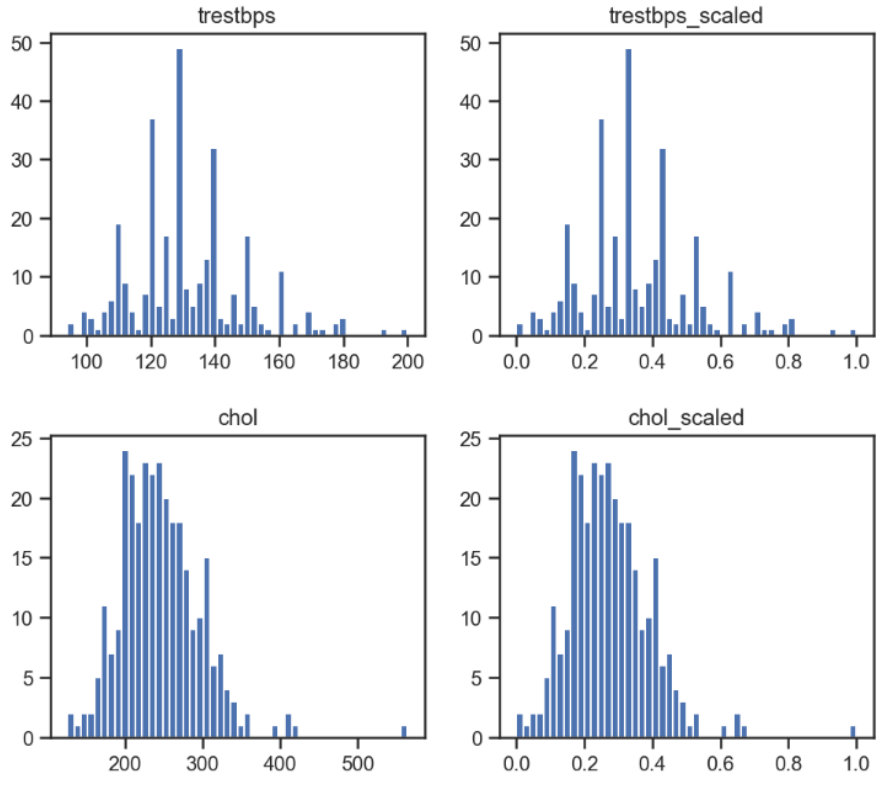
5 rows × 28 columns

Убедимся, что масштабирование не повлияло на распределение данных

In [40]:
```
# Убедимся, 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()
```





```
corr_cols_1 = scale_cols + ['sex']
corr_cols_1
```

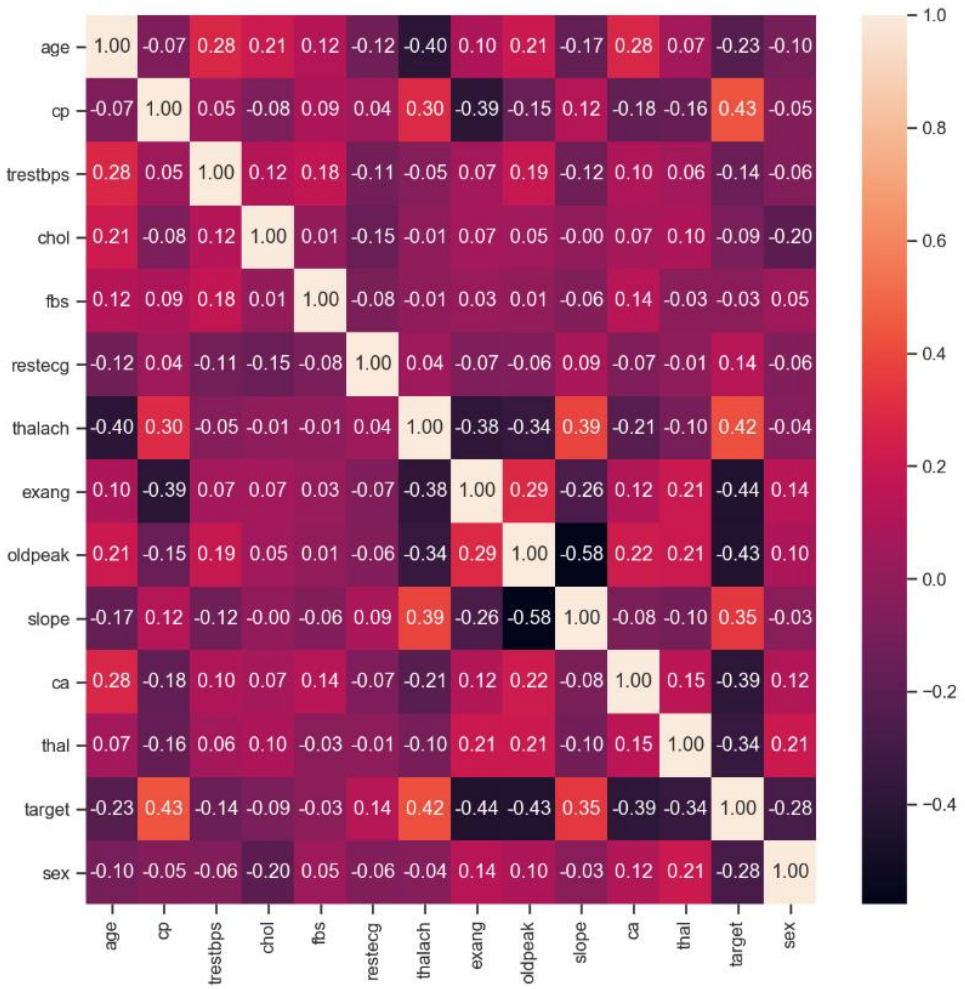
```
['age',
 'cp',
 'trestbps',
 'chol',
 'fbs',
 'restecg',
 'thalach',
 'exang',
 'oldpeak',
 'slope',
 'ca',
 'thal',
 'target',
 'sex']
```

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

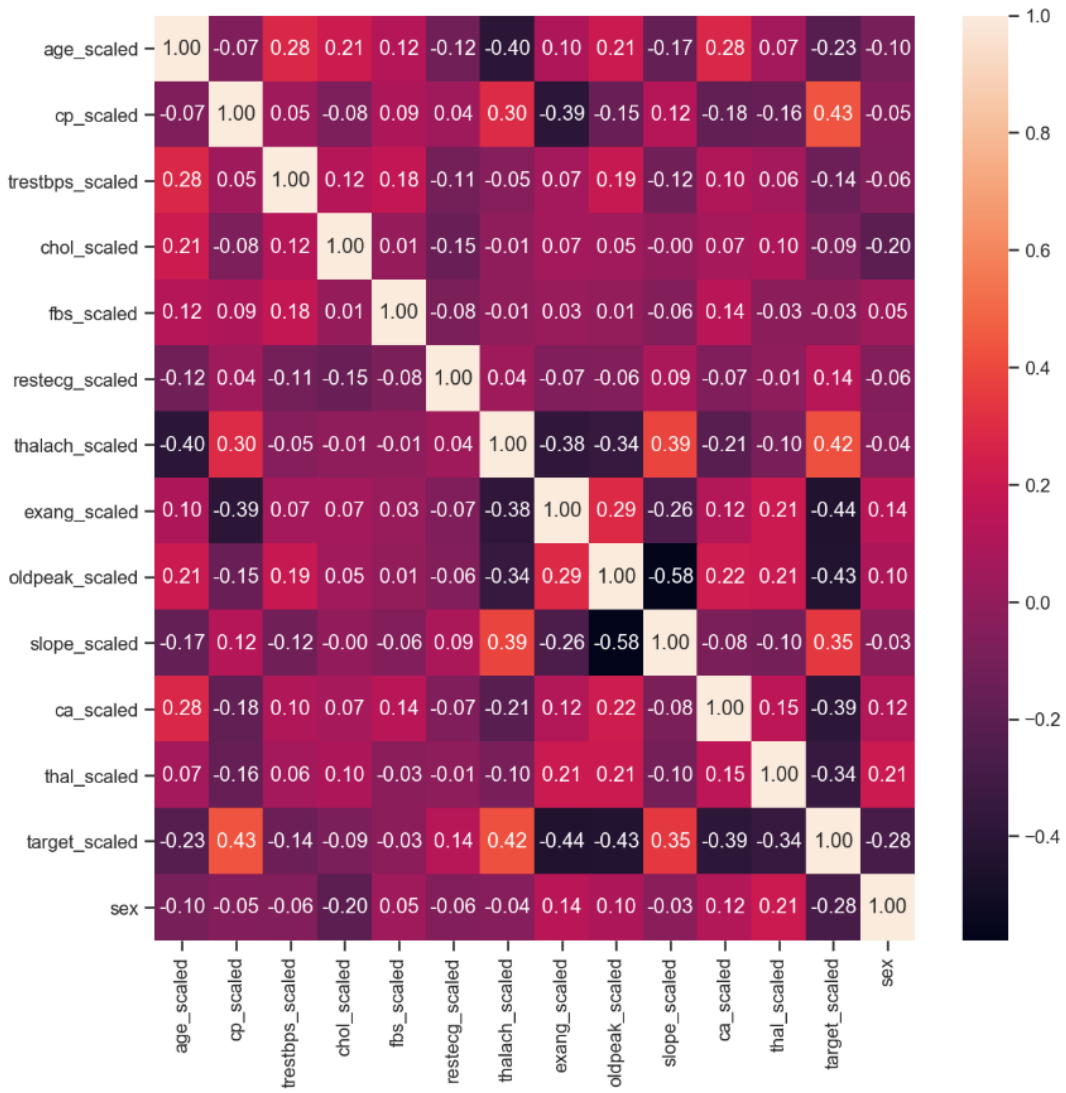
```
['age_scaled',
 'cp_scaled',
 'trestbps_scaled',
 'chol_scaled',
 'fbs_scaled',
 'restecg_scaled',
 'thalach_scaled',
 'exang_scaled',
 'oldpeak_scaled',
 'slope_scaled',
 'ca_scaled',
 'thal_scaled',
 'target_scaled',
 'sex']
```

Построим корреляционную матрицу:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(data[corr_cols_1].corr(), annot=True, fmt='.2f')
```




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



Отрисовка ROC-кривой:

```
# Отрисовка ROC-кривой
def draw_roc_curve(y_true, y_score, 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
    plt.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    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')
    plt.legend(loc="lower right")
    plt.show()
```

Подготовим данные для разделения на обучающую и тестовую выборки:

```
# С использованием метода train_test_split разделим выборку на обучающую и тестовую
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=1)
print("X_train:", X_train.shape)
print("X_test:", X_test.shape)
print("y_train:", y_train.shape)
print("y_test:", y_test.shape)
```

```
X_train: (227, 10)
X_test: (76, 10)
y_train: (227,)
y_test: (76,)
```

Выберем подходящие для нашей задачи метрики:

1. Confusion matrix

Количество верно и ошибочно классифицированных данных, представленное в виде матрицы.

2. ROC-кривая

Используется для оценки качества бинарной классификации. Показывает, какую долю классов алгоритм предсказал неверно. Чем сильнее отклоняется кривая от верхнего левого угла графика, тем хуже качество классификации.

3. Accuracy

Метрика вычисляет процент (долю в диапазоне от 0 до 1) правильно определенных классов. Главная проблема метрики accuracy в том, что она показывает точность по всем классам, но для каждого класса точность может быть разная. Поэтому более предпочтительной является метрика balanced_accuracy.

Random Forest

Ансамблевый метод, заключается в построении алгоритма машинного обучения на базе нескольких, в данном случае решающих деревьев. это множество решающих деревьев. В задаче регрессии их ответы усредняются, в задаче классификации принимается решение голосованием по большинству. Все деревья строятся независимо по следующей схеме:

- Выбирается подвыборка обучающей выборки размера samplesize (м.б. с возвращением) – по ней строится дерево (для каждого дерева — своя подвыборка).

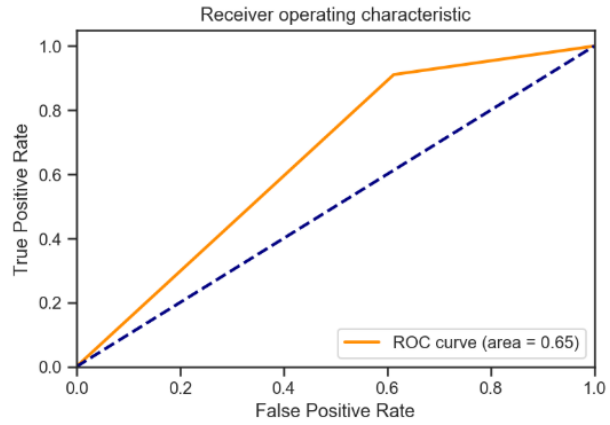
- Для построения каждого расщепления в дереве просматриваем max_features случайных признаков (для каждого нового расщепления — свои случайные признаки).
- Выбираем наилучшие признак и расщепление по нему (по заранее заданному критерию). Дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса), но есть параметры, которые ограничивают высоту дерева, число объектов в листьях и число объектов в подвыборке, при котором проводится расщепление.

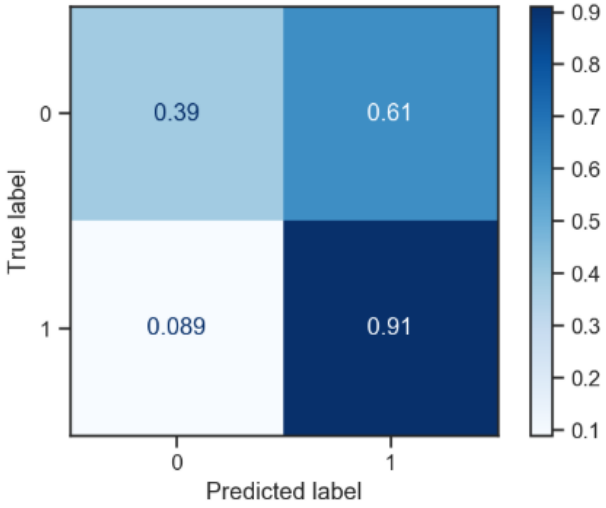
Обучим модель:

```
models = {'KNN_3':KNeighborsClassifier(n_neighbors=3),
          'SVC':SVC(),
          'Tree':DecisionTreeClassifier(),
          'RF':RandomForestClassifier(),
          'GB':GradientBoostingClassifier()}
```

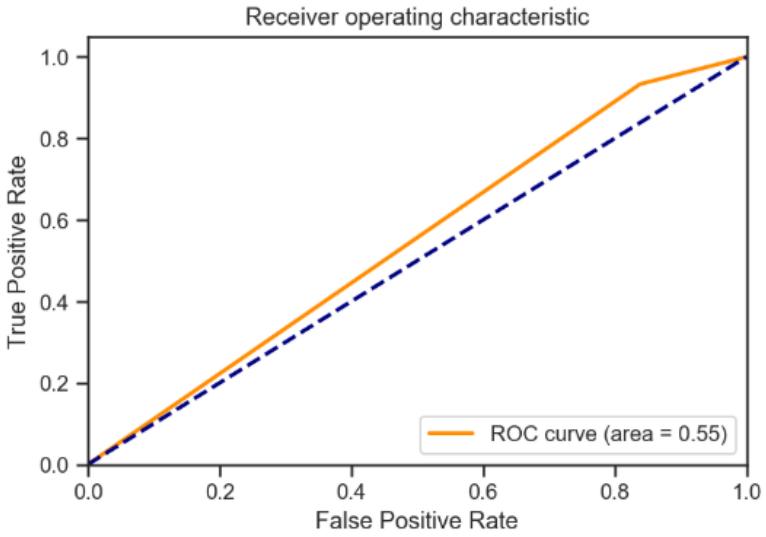
```
for model_name, model in models.items():
    test_model(model_name, model, metricLogger)
```

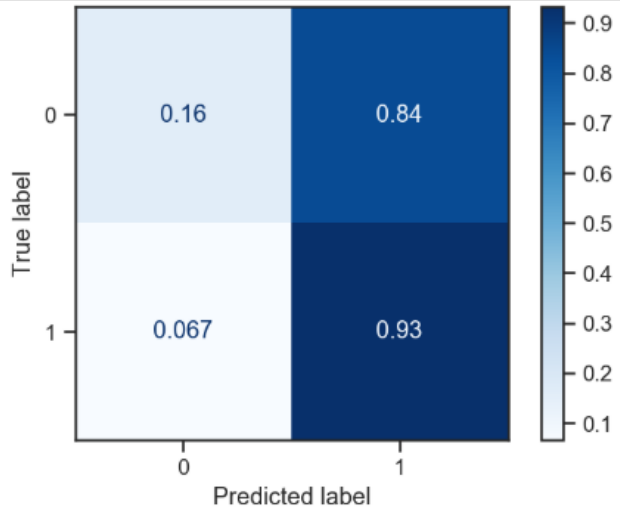
```
**************************************************
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
**************************************************
```



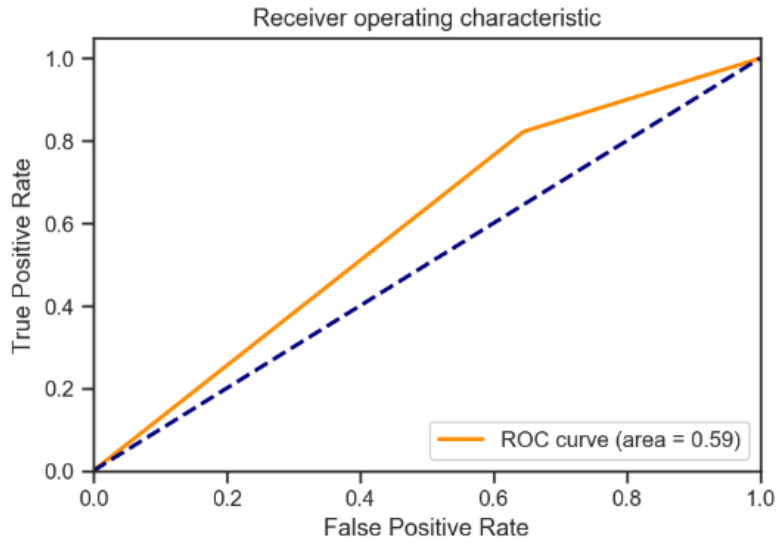


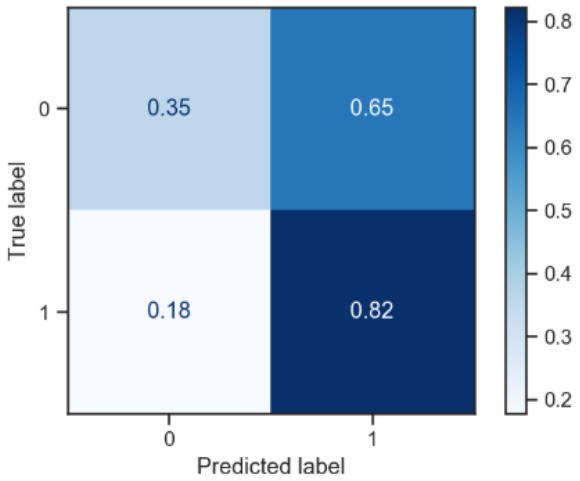

```
**************************************************
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
**************************************************
```





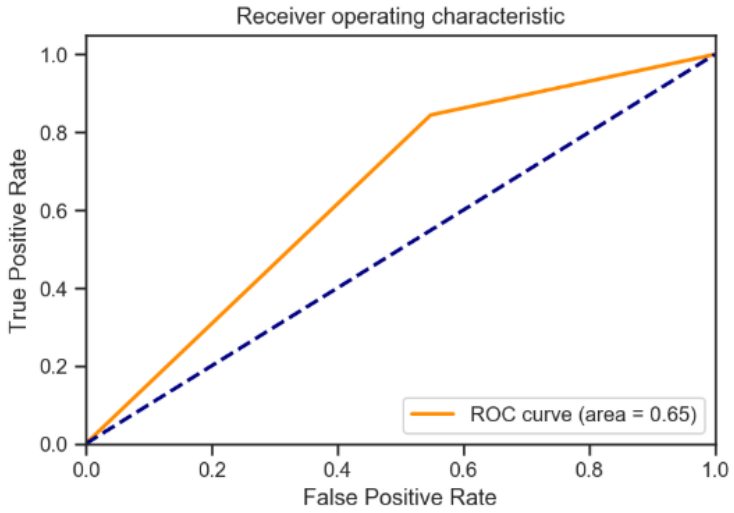

```
**************************************************
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
**************************************************
```

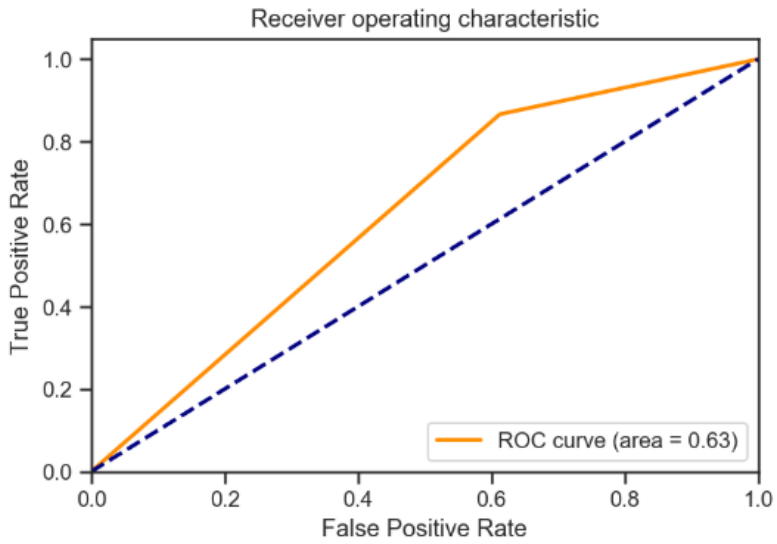


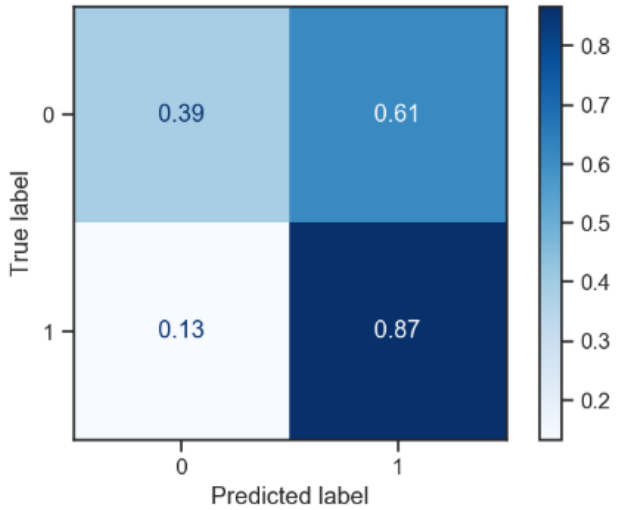



```
**************************************************
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
**************************************************
```

```
**************************************************
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
**************************************************
```







Попробуем улучшить качество модели с помощью подбора гиперпараметров при помощи метода GridSearchCV:

```
n_range = np.array(range(0,30,1))
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

```
[{'n_neighbors': array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29])}]
```

```
%%time
clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='accuracy', n_jobs = -1)
clf_gs.fit(X, y)
```

```
Wall time: 3.58 s
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=-1,
             param_grid=[{'n_neighbors': array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 1
4, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29])}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

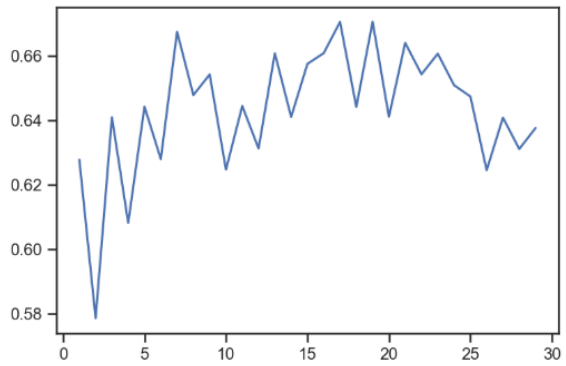
```
# Лучшая модель
clf_gs.best_estimator_
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=19, p=2,
                     weights='uniform')
```

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

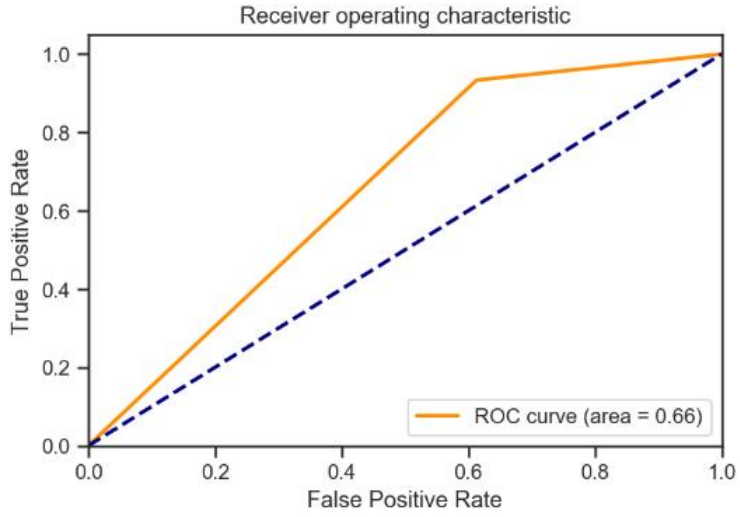
```
{'n_neighbors': 19}
```

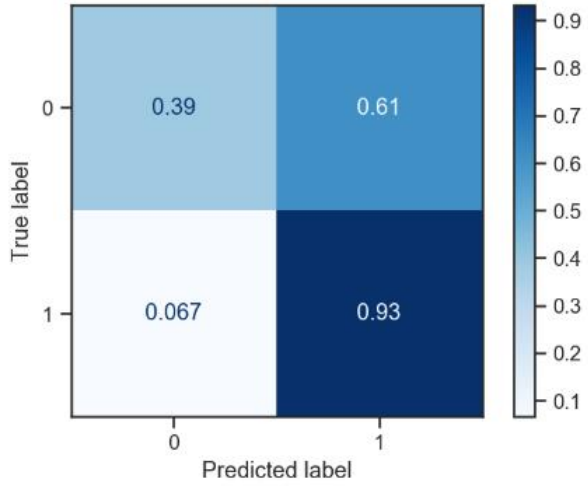
```
# Изменение качества на тестовой выборке в зависимости от K-соседей
plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```




```
test_model('KNN_5', KNeighborsClassifier(n_neighbors=5), metricLogger)
```

```
**************************************************
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
**************************************************
```






```
# Метрики качества модели
metrics = metricLogger.df['metric'].unique()
metrics
```

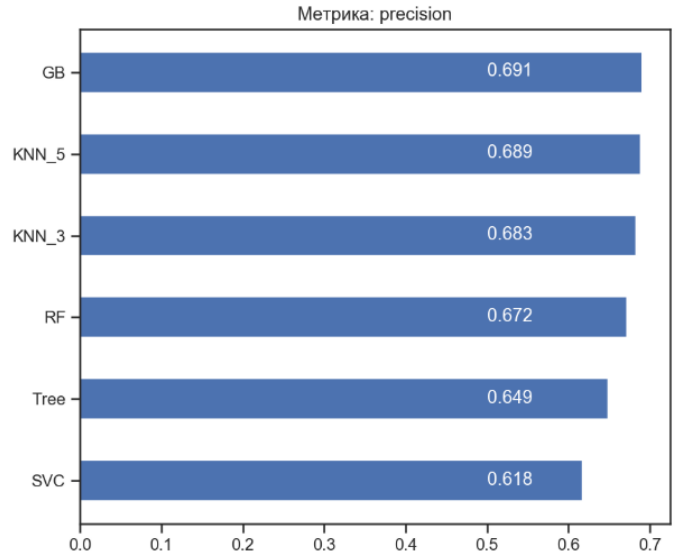
```
array(['precision', 'recall', 'accuracy', 'roc_auc'], dtype=object)
```

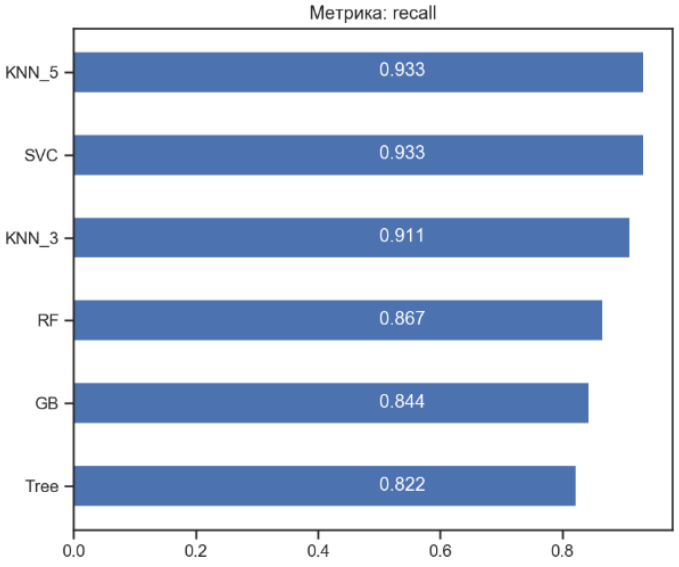
Метрики качества модели

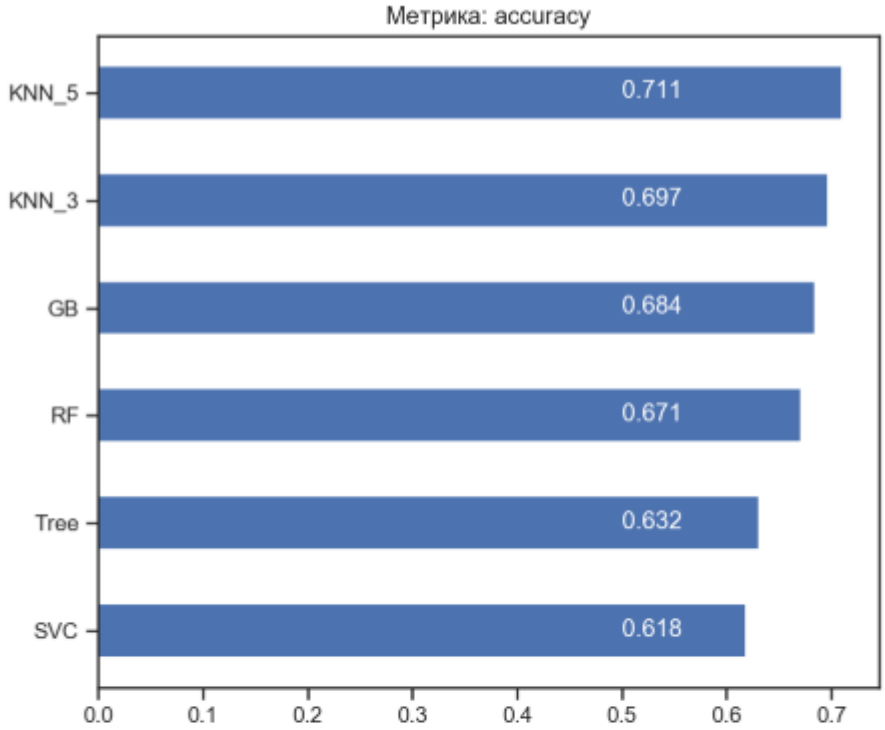
```
# Метрики качества модели
metrics = metricLogger.df['metric'].unique()
metrics
```

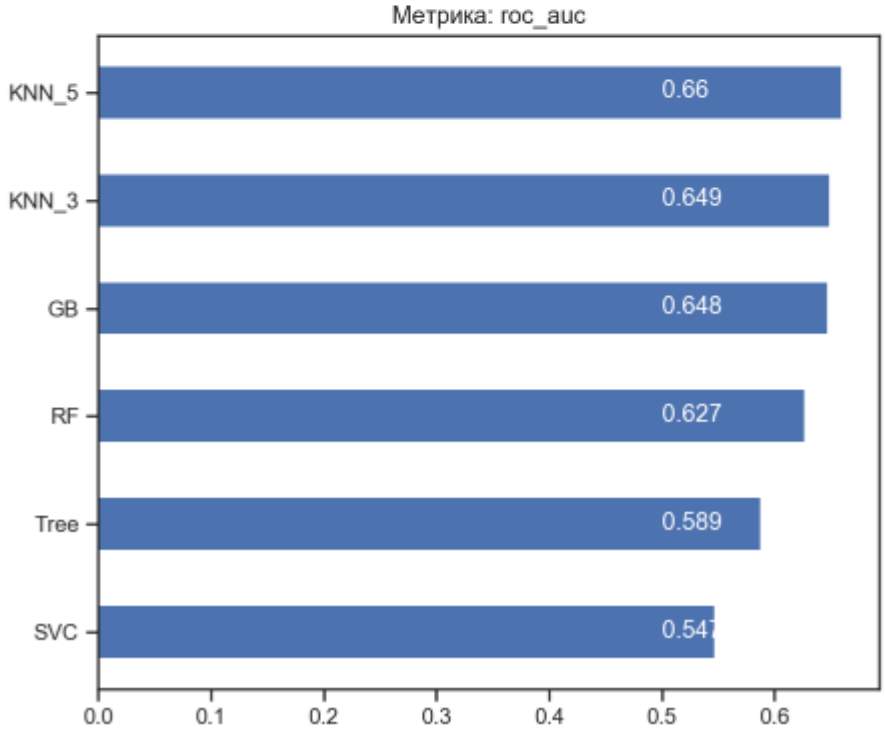
```
array(['precision', 'recall', 'accuracy', 'roc_auc'], dtype=object)
```

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









Выводы

В ходе курсовой работы были закреплены полученные в течение курса знания и навыки. Для исследования использовались следующие модели: стохастический градиентный спуск, случайный лес, градиентный бустинг, метод ближайших

соседей, метод опорных векторов. Для оценки качества использовались три метрики: ROC-кривая, confusion matrix и balanced_accuracy.

Список использованных источников

1. Конспект лекций по дисциплине «Технологии машинного обучения». 2020: https://github.com/ugapanyuk/ml_course_2020/wiki/COURSE_TMO
2. Документация scikit-learn: https://scikit-learn.org/stable/index.html
3. Метрики в задачах машинного обучения: https://habr.com/ru/company/ods/blog/328372/