|  |  |
| --- | --- |
|  | **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 КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

***НИР по обработке и анализу набора данных химического исследования для установления происхождения вина\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

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

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

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

Студент \_\_\_\_\_ИУ5-32М\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ \_\_Петрова И.В.\_\_\_\_**

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

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ \_ \_\_Гапанюк Ю.Е.\_ \_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ \_ \_\_Гапанюк Ю.Е.\_ \_**

(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.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине Технологии машинного обучения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы ИУ5-32М\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Петрова Ирина Викторовна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсового проекта НИР по обработке и анализу набора данных химического \_\_\_\_\_\_\_\_\_

исследования для установления происхождения вина \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Направленность КП (учебный, исследовательский, практический, производственный, др.)

учебный \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

***Задание\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

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

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

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

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

***Оформление курсового проекта:***

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

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

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

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

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

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

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Руководитель курсового проекта**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_ \_ \_\_Гапанюк Ю.Е.\_ \_**

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ \_\_Петрова И.В.\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

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

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc57057769)

[Постановка задачи 4](#_Toc57057770)

[Текстовое описание набора данных 5](#_Toc57057771)

[ОСНОВНАЯ ЧАСТЬ 7](#_Toc57057772)

[Импорт библиотек 7](#_Toc57057773)

[Загрузка данных 7](#_Toc57057774)

[Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных 8](#_Toc57057775)

[Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей 14](#_Toc57057776)

[Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения 17](#_Toc57057777)

[Выбор метрик для последующей оценки качества моделей 19](#_Toc57057778)

[Выбор наиболее подходящих моделей для решения задачи классификации или регрессии 21](#_Toc57057779)

[Формирование обучающей и тестовой выборок на основе исходного набора данных 22](#_Toc57057780)

[Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров 22](#_Toc57057781)

[Решение задачи классификации 22](#_Toc57057782)

[Решение задачи регрессии 24](#_Toc57057783)

[Подбор гиперпараметров для выбранных моделей 25](#_Toc57057784)

[Решение задачи классификации 25](#_Toc57057785)

[Решение задачи регрессии 26](#_Toc57057786)

[Сравнение качества полученных моделей с качеством baseline-моделей 27](#_Toc57057787)

[Решение задачи классификации 27](#_Toc57057788)

[Решение задачи регрессии 27](#_Toc57057789)

[ЗАКЛЮЧЕНИЕ 28](#_Toc57057790)

[Формирование выводов о качестве построенных моделей на основе выбранных метрик 28](#_Toc57057791)

[Решение задачи классификации 28](#_Toc57057792)

[Решение задачи регрессии 29](#_Toc57057793)

ВВЕДЕНИЕ

Постановка задачи

**Цель работы:** курсовая работа – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсовой работы является отчет.

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

**Необходимые шаги для выполнения КР:** предметная часть курсовой работы содержит выполнение следующих шагов:

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.
3. Анализ и заполнение пропусков в данных. 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.
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. Оценка качества выбранных моделей на основе кросс-валидации. Необходимо использовать не менее трех различных схем кросс-валидации и обосновать их выбор.
9. Оценка качества моделей на основе тестовой выборки.
10. Формирование выводов о качестве построенных моделей. Необходимо сравнить качество моделей на основании выбранных метрик для различных схем кросс-валидации и тестовой выборки. Результаты сравнения качества нужно отобразить в виде графиков и сделать выводы в форме текстового описания.

**Порядок и время проведения работы:** работа выполняется самостоятельно в часы внеаудиторных занятий. Выполняется студентом индивидуально. По итогам выполнения КР составляется и защищается отчет в бумажном и электронном виде.

Отчет по курсовой работе – документ, содержащий описание решения комплексной задачи машинного обучения с обоснованием принятых решений.

Обязательными структурными элементами отчета являются:

1. Титульный лист установленного образца.
2. Задание установленного образца.
3. Содержание.
4. Введение.
5. Основная часть, содержащая описание постановки задачи и последовательности действий студента по решению поставленной задачи.
6. Заключение (формулировка выводов по выполненной работе).
7. Список использованных источников информации (бумажных и электронных).

**Задание:** выполнить поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора построить модели для решения задачи классификации и регрессии. Провести разведочный анализ данных, построить графики. Выполнить анализ и заполнение пропусков в данных. Выбрать признаки, подходящие для выполнения моделей. Провести корреляционный анализ данных и сформировать промежуточные выводы о возможности построения моделей машинного обучения. Выбрать метрики для последующей оценки качества моделей. Выбрать наиболее подходящие модели для решения задач классификации и регрессии. Сформировать обучающую и тестовую выборки на основе исходного набора данных. Оценить качество выбранных моделей на основе кросс-валидации и на основе тестовой выборки. Сформировать выводы о качестве построенных моделей.

Текстовое описание набора данных

В рамках курсовой работы используется набор данных, хранящий в себе информацию о химическом анализе для установления происхождения вина: <https://archive.ics.uci.edu/ml/datasets/Wine>.

Эти данные являются результатами химического анализа вин, выращенных в одном регионе Италии, но полученных из трех различных сортов. В результате анализа было определено 13 компонентов, содержащихся в каждом из трех видов вин.

Набор данных содержит следующие колонки:

* Алкоголь
* Яблочная кислота
* Зола
* Щелочность золы
* Магний
* Всего фенолов
* Флаваноиды
* Нефлаваноидные фенолы
* Проантоцианы
* Интенсивность цвета
* Оттенок
* OD280 / OD315 (разбавленность вина)
* Пролин

Выбранный набор данных в рамках текущей курсовой работы будет использован как для решения задачи классификации, так и для решения задачи регрессии.

ОСНОВНАЯ ЧАСТЬ

Импорт библиотек

**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** **gmdhpy** **import** gmdh

%matplotlib inline

sns.set(style="ticks")

*# 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()

Загрузка данных

**from** **sklearn.datasets** **import** \*

wine = load\_wine()

train = pd.DataFrame(data= np.c\_[wine['data'], wine['target']],

columns= wine['feature\_names'] + ['target'])

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

*# Первые 5 строк датасета*

train.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | |  |  | |  |  | |  | |
|  | **alcohol** | **malic\_acid** | **ash** | **alcalinity\_of\_ash** | | **magnesium** | **total\_phenols** | | **flavanoids** | **nonflavanoid\_phenols** | | **proanthocyanins** | |
| **0** | 14.23 | 1.71 | 2.43 | 15.6 | | 127.0 | 2.80 | | 3.06 | 0.28 | | 2.29 | |
| **1** | 13.20 | 1.78 | 2.14 | 11.2 | | 100.0 | 2.65 | | 2.76 | 0.26 | | 1.28 | |
| **2** | 13.16 | 2.36 | 2.67 | 18.6 | | 101.0 | 2.80 | | 3.24 | 0.30 | | 2.81 | |
| **3** | 14.37 | 1.95 | 2.50 | 16.8 | | 113.0 | 3.85 | | 3.49 | 0.24 | | 2.18 | |
| **4** | 13.24 | 2.59 | 2.87 | 21.0 | | 118.0 | 2.80 | | 2.69 | 0.39 | | 1.82 | |
|  |  |  |  |  | |  |  | |  |  | |  | |
|  | **color\_intensity** | | | | **hue** | | | **od280/od315\_of\_diluted\_wines** | | | **proline** | | **target** |
| **0** | 5.64 | | | | 1.04 | | | 3.92 | | | 1065.0 | | 0.0 |
| **1** | 4.38 | | | | 1.05 | | | 3.40 | | | 1050.0 | | 0.0 |
| **2** | 5.68 | | | | 1.03 | | | 3.17 | | | 1185.0 | | 0.0 |
| **3** | 7.80 | | | | 0.86 | | | 3.45 | | | 1480.0 | | 0.0 |
| **4** | 4.32 | | | | 1.04 | | | 2.93 | | | 735.0 | | 0.0 |

*# Размер обучающего датасета - 178 строк, 14 колонок*

train.shape

(178, 14)

*# Список колонок*

train.columns

Index(['alcohol', 'malic\_acid', 'ash', 'alcalinity\_of\_ash', 'magnesium',

'total\_phenols', 'flavanoids', 'nonflavanoid\_phenols',

'proanthocyanins', 'color\_intensity', 'hue',

'od280/od315\_of\_diluted\_wines', 'proline', 'target'],

dtype='object')

*# Список колонок с типами данных*

train.dtypes

alcohol float64

malic\_acid float64

ash float64

alcalinity\_of\_ash float64

magnesium float64

total\_phenols float64

flavanoids float64

nonflavanoid\_phenols float64

proanthocyanins float64

color\_intensity float64

hue float64

od280/od315\_of\_diluted\_wines float64

proline float64

target float64

dtype: object

*# Проверим наличие пустых значений*

train.isnull().sum()

alcohol 0

malic\_acid 0

ash 0

alcalinity\_of\_ash 0

magnesium 0

total\_phenols 0

flavanoids 0

nonflavanoid\_phenols 0

proanthocyanins 0

color\_intensity 0

hue 0

od280/od315\_of\_diluted\_wines 0

proline 0

target 0

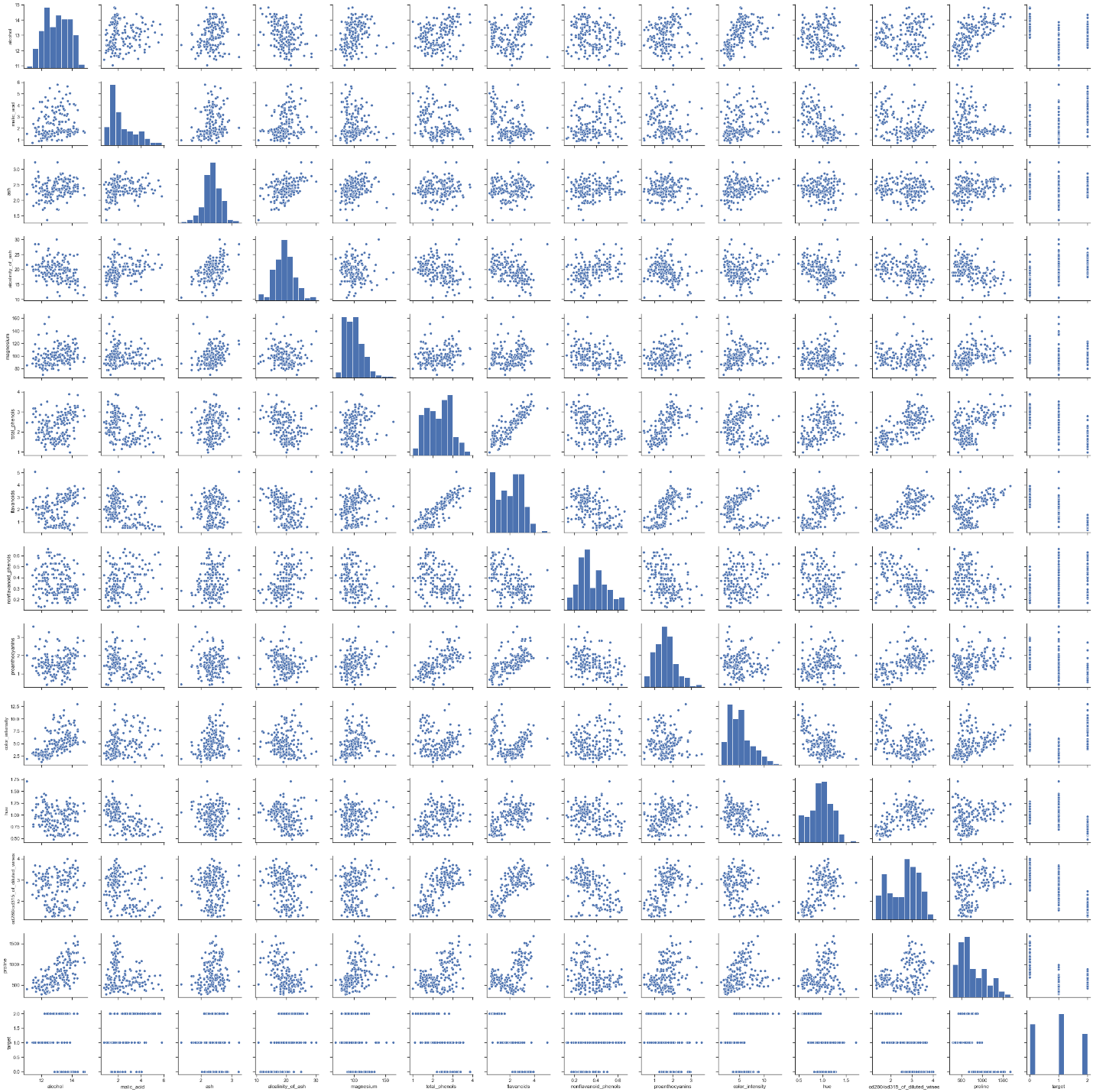
dtype: int64

**Вывод:** выбранный набор данных не содержит пропусков.

*# Парные диаграммы*

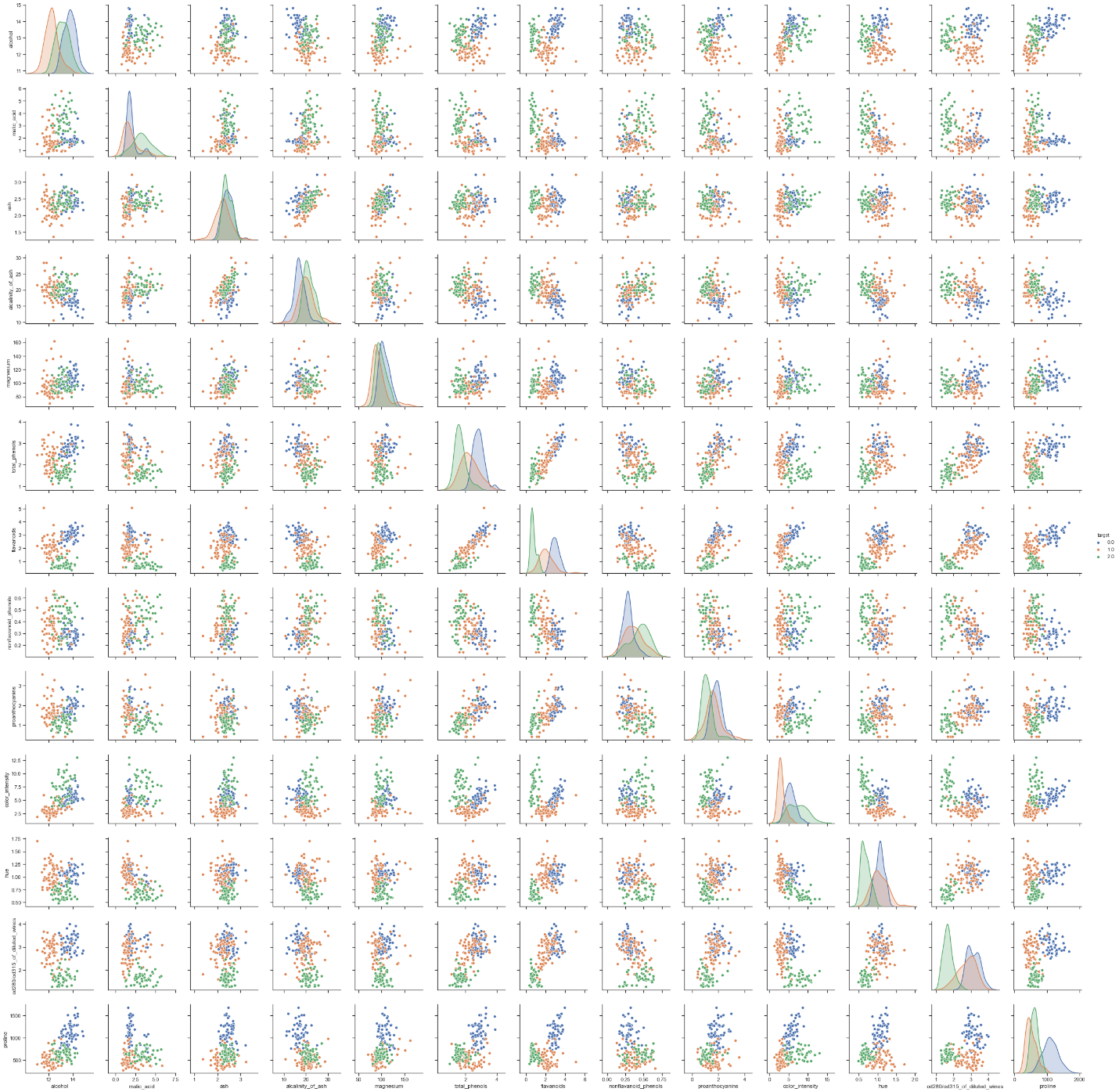
sns.pairplot(train)

<seaborn.axisgrid.PairGrid at 0x25b3fef78d0>



sns.pairplot(train, hue="target")

<seaborn.axisgrid.PairGrid at 0x25b49547048>



*# Убедимся, что целевой признак*

*# для задачи классификации содержит только 0, 1 и 2*

train['target'].unique()

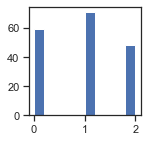
array([0., 1., 2.])

*# Оценим дисбаланс классов для target*

fig, ax = plt.subplots(figsize=(2,2))

plt.hist(train['target'])

plt.show()



train['target'].value\_counts()

1.0 71

0.0 59

2.0 48

Name: target, dtype: int64

*# посчитаем дисбаланс классов*

total = train.shape[0]

class\_0, class\_1, class\_3 = train['target'].value\_counts()

print('Класс 0 составляет **{}**%, класс 1 составляет **{}**%, класс 2 составляет **{}**%.'

.format(round(class\_0 / total, 4)\*100, round(class\_1 / total, 4)\*100, round(class\_3 / total, 4)\*100))

Класс 0 составляет 39.89%, класс 1 составляет 33.15%, класс 2 составляет 26.97%.

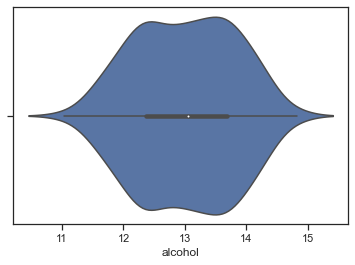
**Вывод:** дисбаланс классов практически отсутствует.

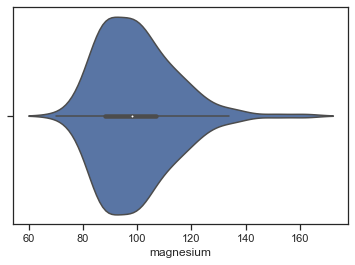
*# Скрипичные диаграммы для числовых колонок*

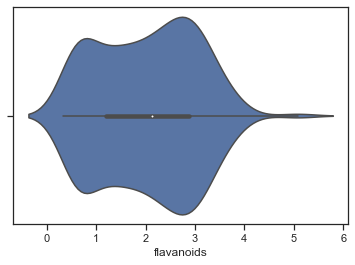
**for** col **in** ['alcohol', 'magnesium', 'flavanoids', 'hue', 'proline']:

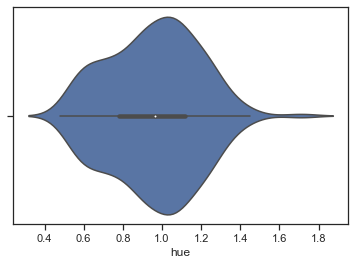
sns.violinplot(x=train[col])

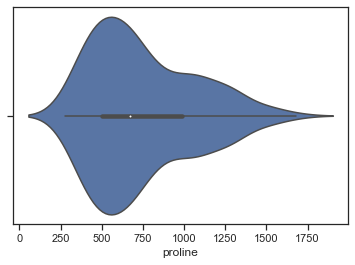
plt.show()











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

train.dtypes

alcohol float64

malic\_acid float64

ash float64

alcalinity\_of\_ash float64

magnesium float64

total\_phenols float64

flavanoids float64

nonflavanoid\_phenols float64

proanthocyanins float64

color\_intensity float64

hue float64

od280/od315\_of\_diluted\_wines float64

proline float64

target float64

dtype: object

Для построения моделей будут использоваться все признаки. Категориальные признаки отсутствуют, их кодирования не требуется.

*# Числовые колонки для масштабирования*

scale\_cols = ['alcohol', 'malic\_acid', 'ash', 'alcalinity\_of\_ash', 'magnesium',

'total\_phenols', 'flavanoids', 'nonflavanoid\_phenols',

'proanthocyanins', 'color\_intensity', 'hue',

'od280/od315\_of\_diluted\_wines', 'proline']

sc1 = MinMaxScaler()

sc1\_data = sc1.fit\_transform(train[scale\_cols])

*# Добавим масштабированные данные в набор данных*

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

col = scale\_cols[i]

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

train[new\_col\_name] = sc1\_data[:,i] *# 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))

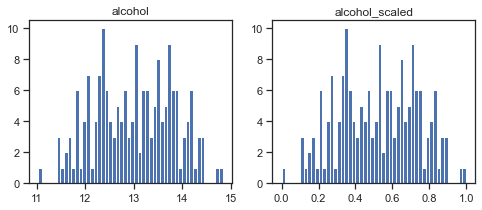
ax[0].hist(train[col], 50)

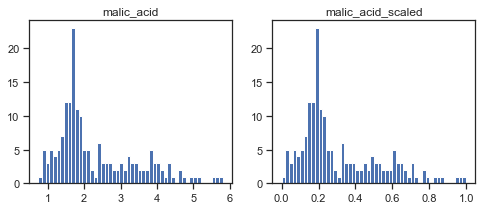
ax[1].hist(train[col\_scaled], 50)

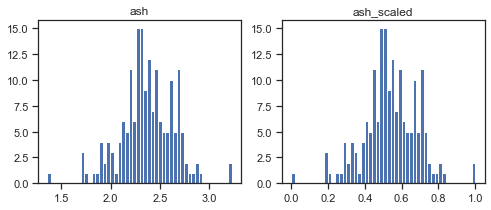
ax[0].title.set\_text(col)

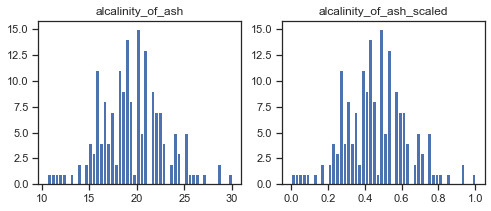
ax[1].title.set\_text(col\_scaled)

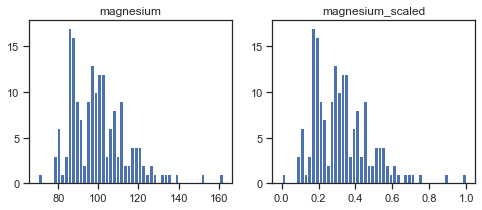
plt.show()

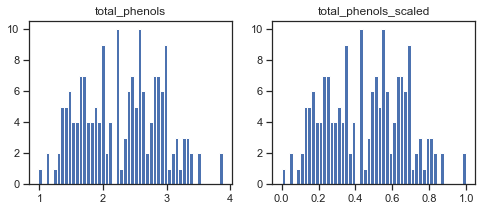


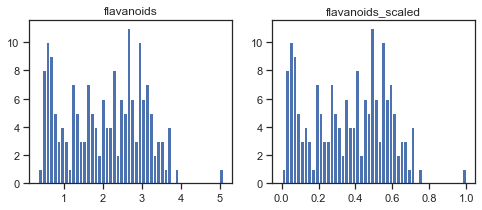


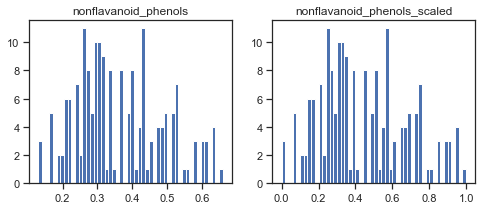


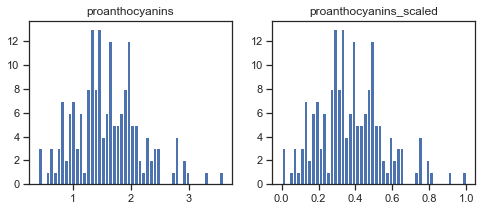


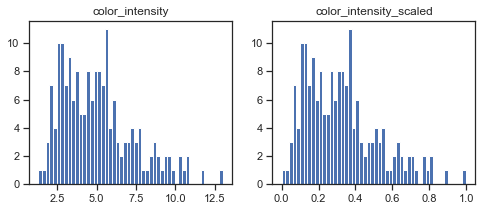


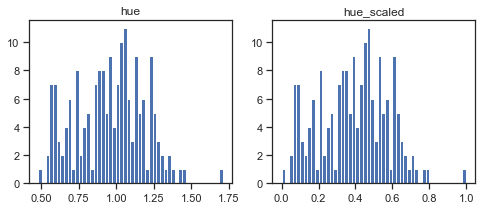


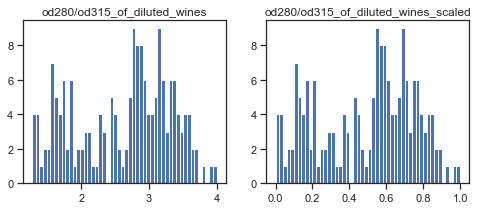


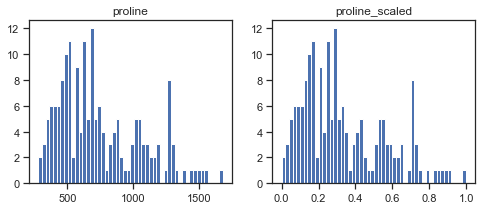












Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения

*# Воспользуемся наличием тестовых выборок,*

*# включив их в корреляционную матрицу*

corr\_cols\_1 = scale\_cols + ['target']

corr\_cols\_1

['alcohol',

'malic\_acid',

'ash',

'alcalinity\_of\_ash',

'magnesium',

'total\_phenols',

'flavanoids',

'nonflavanoid\_phenols',

'proanthocyanins',

'color\_intensity',

'hue',

'od280/od315\_of\_diluted\_wines',

'proline',

'target']

scale\_cols\_postfix = [x+'\_scaled' **for** x **in** scale\_cols]

corr\_cols\_2 = scale\_cols\_postfix + ['target']

corr\_cols\_2

['alcohol\_scaled',

'malic\_acid\_scaled',

'ash\_scaled',

'alcalinity\_of\_ash\_scaled',

'magnesium\_scaled',

'total\_phenols\_scaled',

'flavanoids\_scaled',

'nonflavanoid\_phenols\_scaled',

'proanthocyanins\_scaled',

'color\_intensity\_scaled',

'hue\_scaled',

'od280/od315\_of\_diluted\_wines\_scaled',

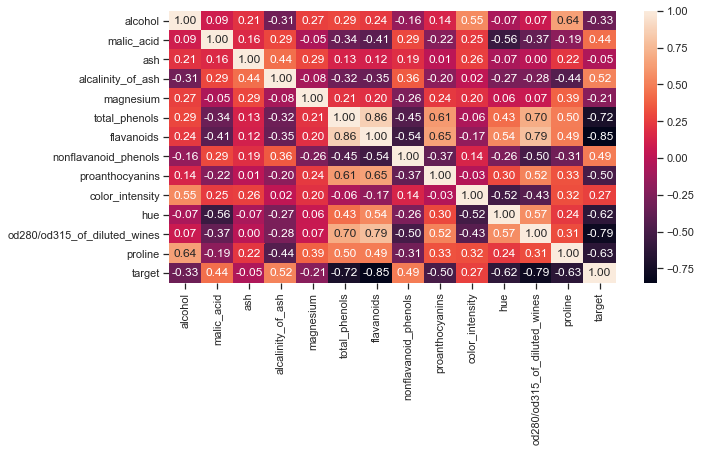
'proline\_scaled',

'target']

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

sns.heatmap(train[corr\_cols\_1].corr(), annot=**True**, fmt='.2f')

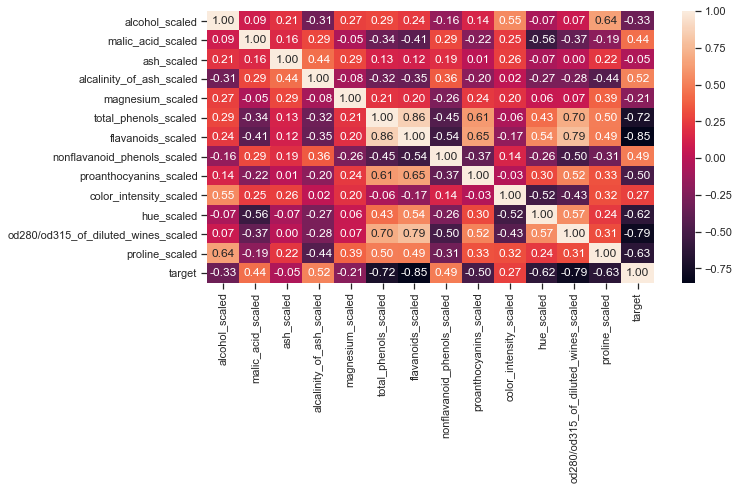
<matplotlib.axes.\_subplots.AxesSubplot at 0x25b54a564a8>



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

sns.heatmap(train[corr\_cols\_2].corr(), annot=**True**, fmt='.2f')

<matplotlib.axes.\_subplots.AxesSubplot at 0x25b52dbcc88>



На основе корреляционной матрицы можно сделать следующие выводы:

* Корреляционные матрицы для исходных и масштабированных данных совпадают.
* Целевой признак классификации "target" наиболее сильно коррелирует со щелочностью золы (0.52), нефлаваноидными фенолами (0.49) и яблочной кислотой (0.44). Эти признаки обязательно следует оставить в модели классификации.
* Целевой признак регрессии "flavanoids" наиболее сильно коррелирует с "total\_phenols" (0.86) и OD280 / OD315 (разбавленностью вина) (0.79). Эти признаки обязательно следует оставить в модели регрессии.
* Большие по модулю значения коэффициентов корреляции свидетельствуют о значимой корреляции между исходными признаками и целевым признаком. На основании корреляционной матрицы можно сделать вывод о том, что данные позволяют построить модель машинного обучения.

Выбор метрик для последующей оценки качества моделей

В качестве метрик для решения задачи классификации будут использоваться:

Метрики, формируемые на основе матрицы ошибок:

1. Метрика precision: можно переводить как точность, но такой перевод совпадает с переводом метрики "accuracy".



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

Используется функция precision\_score.

1. Метрика recall (полнота):



Доля верно предсказанных классификатором положительных объектов, из всех действительно положительных объектов.

Используется функция recall\_score.

В качестве метрик для решения задачи регрессии будут использоваться:

1. Mean absolute error - средняя абсолютная ошибка



Вычисляется с помощью функции mean\_absolute\_error.

1. Mean squared error - средняя квадратичная ошибка



Вычисляется с помощью функции mean\_squared\_error.

1. Метрика R2 или коэффициент детерминации



Вычисляется с помощью функции r2\_score.

**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()

Выбор наиболее подходящих моделей для решения задачи классификации или регрессии

Для решения задачи классификации будут использованы следующие модели:

* Логистическая регрессия
* Метод ближайших соседей
* Случайный лес

Для решения задачи регрессии будут использованы следующие модели:

* Машина опорных векторов
* Метод ближайших соседей
* Решающее дерево
* Градиентный бустинг

Формирование обучающей и тестовой выборок на основе исходного набора данных

*# Признаки для задачи классификации*

task\_clas\_cols = ['alcalinity\_of\_ash\_scaled', 'nonflavanoid\_phenols\_scaled',

'malic\_acid\_scaled', 'color\_intensity\_scaled'] *# Разделение выборки на обучающую и тестовую* clas\_X\_train, clas\_X\_test, clas\_Y\_train, clas\_Y\_test = train\_test\_split( train[task\_clas\_cols], train['target'], test\_size=0.5, random\_state=1) clas\_X\_train.shape, clas\_X\_test.shape, clas\_Y\_train.shape, clas\_Y\_test.shape

((89, 4), (89, 4), (89,), (89,))

*# Признаки для задачи регресии*

task\_regr\_cols = ['total\_phenols\_scaled', 'od280/od315\_of\_diluted\_wines\_scaled',

'proanthocyanins\_scaled', 'proline\_scaled']

*# Разделение выборки на обучающую и тестовую*

regr\_X\_train, regr\_X\_test, regr\_Y\_train, regr\_Y\_test = train\_test\_split(

train[task\_regr\_cols], train['flavanoids'], test\_size=0.5, random\_state=1)

regr\_X\_train.shape, regr\_X\_test.shape, regr\_Y\_train.shape, regr\_Y\_test.shape

((89, 4), (89, 4), (89,), (89,))

Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров

Решение задачи классификации

*# Модели*

clas\_models = {'LogR': LogisticRegression(),

'KNN\_5':KNeighborsClassifier(n\_neighbors=5),

'RF':RandomForestClassifier()} *# Сохранение метрик* clasMetricLogger = MetricLogger()

**def** clas\_train\_model(model\_name, model, clasMetricLogger):

model.fit(clas\_X\_train, clas\_Y\_train)

Y\_pred = model.predict(clas\_X\_test)

precision = precision\_score(clas\_Y\_test.values, Y\_pred, average = 'weighted')

recall = recall\_score(clas\_Y\_test.values, Y\_pred, average = 'weighted')

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

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

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(model)

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

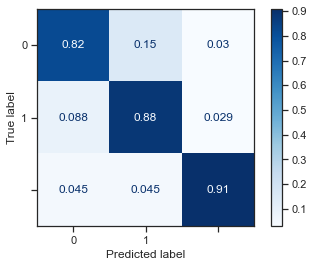
plot\_confusion\_matrix(model, clas\_X\_test, clas\_Y\_test.values,

display\_labels=['0','1'],

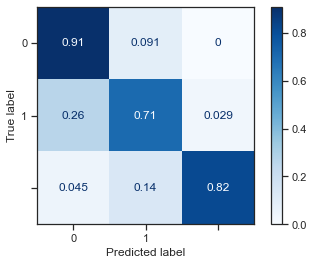
cmap=plt.cm.Blues, normalize='true')

plt.show() **for** model\_name, model **in** clas\_models.items(): clas\_train\_model(model\_name, model, clasMetricLogger)

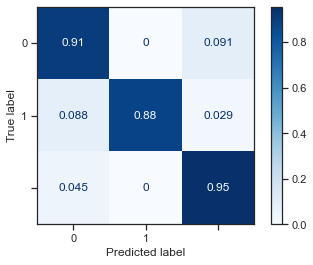
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True, intercept\_scaling=1, l1\_ratio=None, max\_iter=100, multi\_class='auto', n\_jobs=None, penalty='l2', random\_state=None, solver='lbfgs', tol=0.0001, verbose=0, warm\_start=False) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski', metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2, weights='uniform') \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Решение задачи регрессии

*# Модели*

regr\_models = {'KNN\_5':KNeighborsRegressor(n\_neighbors=5),

'SVR':SVR(),

'Tree':DecisionTreeRegressor(),

'GB':GradientBoostingRegressor()} *# Сохранение метрик* regrMetricLogger = MetricLogger()

**def** regr\_train\_model(model\_name, model, regrMetricLogger):

model.fit(regr\_X\_train, regr\_Y\_train)

Y\_pred = model.predict(regr\_X\_test)

mae = mean\_absolute\_error(regr\_Y\_test, Y\_pred)

mse = mean\_squared\_error(regr\_Y\_test, Y\_pred)

r2 = r2\_score(regr\_Y\_test, Y\_pred)

regrMetricLogger.add('MAE', model\_name, mae)

regrMetricLogger.add('MSE', model\_name, mse)

regrMetricLogger.add('R2', model\_name, r2)

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(model)

print()

print('MAE=**{}**, MSE=**{}**, R2=**{}**'.format(

round(mae, 3), round(mse, 3), round(r2, 3)))

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

**for** model\_name, model **in** regr\_models.items():

regr\_train\_model(model\_name, model, regrMetricLogger)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* KNeighborsRegressor(algorithm='auto', leaf\_size=30, metric='minkowski', metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2, weights='uniform') MAE=0.261, MSE=0.112, R2=0.888 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* SVR(C=1.0, cache\_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale', kernel='rbf', max\_iter=-1, shrinking=True, tol=0.001, verbose=False) MAE=0.249, MSE=0.112, R2=0.887 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DecisionTreeRegressor(ccp\_alpha=0.0, criterion='mse', 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') MAE=0.315, MSE=0.156, R2=0.844 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GradientBoostingRegressor(alpha=0.9, ccp\_alpha=0.0, criterion='friedman\_mse', init=None, learning\_rate=0.1, loss='ls', 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) MAE=0.259, MSE=0.113, R2=0.886 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Подбор гиперпараметров для выбранных моделей

Решение задачи классификации

clas\_X\_train.shape

(89, 4)

n\_range = np.array(range(1,70,10))

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

tuned\_parameters

[{'n\_neighbors': array([ 1, 11, 21, 31, 41, 51, 61])}]

%%time

clf\_gs = GridSearchCV(KNeighborsClassifier(), tuned\_parameters, cv=5)

clf\_gs.fit(clas\_X\_train, clas\_Y\_train)

Wall time: 252 ms

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=None,

param\_grid=[{'n\_neighbors': array([ 1, 11, 21, 31, 41, 51, 61])}],

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score=False,

scoring=None, verbose=0)

*# Лучшая модель*

clf\_gs.best\_estimator\_

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=1, p=2,

weights='uniform')

*# Лучшее значение параметров*

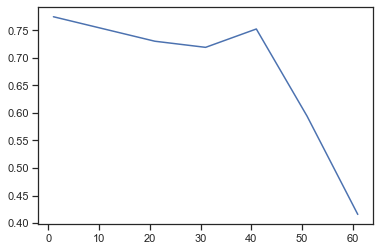
clf\_gs.best\_params\_

{'n\_neighbors': 1}

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

plt.plot(n\_range, clf\_gs.cv\_results\_['mean\_test\_score'])

[<matplotlib.lines.Line2D at 0x25b56f47be0>]



Решение задачи регрессии

n\_range = np.array(range(1,70,10))

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

tuned\_parameters

[{'n\_neighbors': array([ 1, 11, 21, 31, 41, 51, 61])}]

%%time

regr\_gs = GridSearchCV(KNeighborsRegressor(), tuned\_parameters, cv=5, scoring='neg\_mean\_squared\_error')

regr\_gs.fit(regr\_X\_train, regr\_Y\_train)

Wall time: 208 ms

GridSearchCV(cv=5, error\_score=nan,

estimator=KNeighborsRegressor(algorithm='auto', leaf\_size=30,

metric='minkowski',

metric\_params=None, n\_jobs=None,

n\_neighbors=5, p=2,

weights='uniform'),

iid='deprecated', n\_jobs=None,

param\_grid=[{'n\_neighbors': array([ 1, 11, 21, 31, 41, 51, 61])}],

pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score=False,

scoring='neg\_mean\_squared\_error', verbose=0)

*# Лучшая модель*

regr\_gs.best\_estimator\_

KNeighborsRegressor(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=11, p=2,

weights='uniform')

*# Лучшее значение параметров*

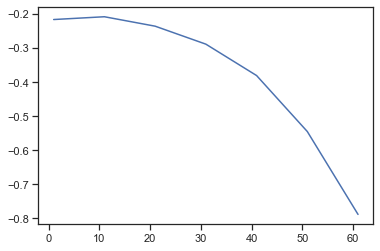
regr\_gs.best\_params\_

{'n\_neighbors': 11}

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

plt.plot(n\_range, regr\_gs.cv\_results\_['mean\_test\_score'])

[<matplotlib.lines.Line2D at 0x25b56fabf98>]



Сравнение качества полученных моделей с качеством baseline-моделей

Решение задачи классификации

clas\_models\_grid = {'KNN\_1':clf\_gs.best\_estimator\_}

**for** model\_name, model **in** clas\_models\_grid.items():

clas\_train\_model(model\_name, model, clasMetricLogger)

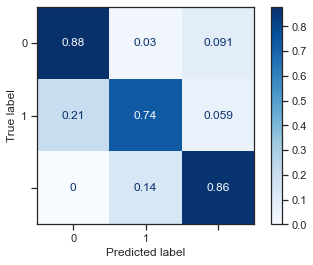
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=1, p=2,

weights='uniform')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



Решение задачи регрессии

regr\_models\_grid = {'KNN\_11':regr\_gs.best\_estimator\_}

**for** model\_name, model **in** regr\_models\_grid.items():

regr\_train\_model(model\_name, model, regrMetricLogger)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

KNeighborsRegressor(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=11, p=2,

weights='uniform')

MAE=0.267, MSE=0.116, R2=0.884

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ЗАКЛЮЧЕНИЕ

Формирование выводов о качестве построенных моделей на основе выбранных метрик

Решение задачи классификации

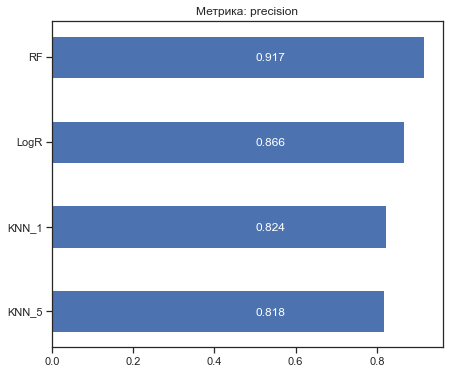
*# Метрики качества модели* clas\_metrics = clasMetricLogger.df['metric'].unique() clas\_metrics

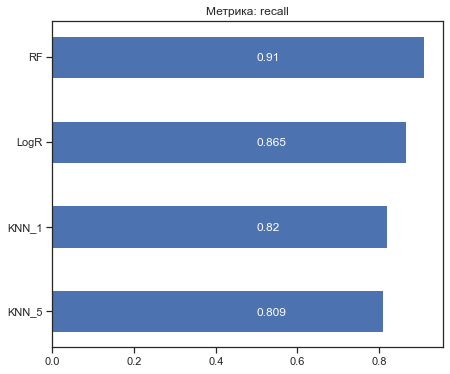
array(['precision', 'recall'], dtype=object)

*# Построим графики метрик качества модели*

**for** metric **in** clas\_metrics:

clasMetricLogger.plot('Метрика: ' + metric, metric, figsize=(7, 6))





**Вывод:** на основании двух метрик из двух используемых, лучшей оказалась модель "случайный лес"

Решение задачи регрессии

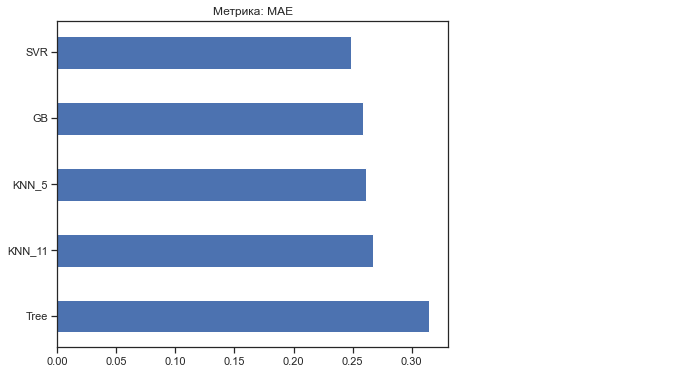
*# Метрики качества модели*

regr\_metrics = regrMetricLogger.df['metric'].unique()

regr\_metrics

array(['MAE', 'MSE', 'R2'], dtype=object)

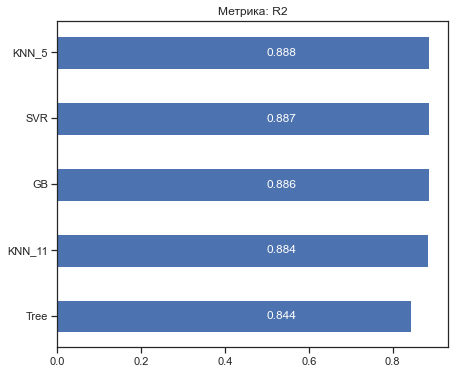
regrMetricLogger.plot('Метрика: ' + 'MAE', 'MAE', ascending=**False**, figsize=(7, 6))



regrMetricLogger.plot('Метрика: ' + 'MSE', 'MSE', ascending=**False**, figsize=(7, 6))



regrMetricLogger.plot('Метрика: ' + 'R2', 'R2', ascending=**True**, figsize=(7, 6))



**Вывод:** на основании двух метрик из трех используемых, лучшей оказалась модель 5 ближайших соседей.