Московский государственный технический университет имени Н.Э.Баумана

Кафедра «Системы обработки информации и управления»

ОТЧЕТ

по домашнему заданию
по курсу «Методы машинного обучения»

Исполнитель: Гунькин М.А.
группа ИУ5-21М

Проверил: Гапанюк Ю.Е.

Москва, 2020

Задание

Домашнее задание по дисциплине направлено на решение комплексной задачи машинного обучения.
Домашнее задание включает 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. Анализ и заполнение пропусков в данных.
3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
4. Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.
5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее двух метрик и обосновать выбор.
6. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее трех моделей, хотя бы одна из которых должна быть ансамблевой.
7. Формирование обучающей и тестовой выборок на основе исходного набора данных.
8. Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.
9. Подбор гиперпараметров для выбранных моделей. Рекомендуется подбирать не более 1-2 гиперпараметров. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.
11. Формирование выводов о качестве построенных моделей на основе выбранных метрик.

Решение

In [2]:

```
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, classificat ion_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_square d_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, LinearSV R
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_g raphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor from sklearn.ensemble import
ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegress or
from sklearn.utils import shuffle
#   !pip install gmdhpy
from gmdhpy import gmdh
%matplotlib inline
sns.set(style="ticks")
```

1. Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных построение модели машинного обучения для решения или задачи регрессии.

В качестве набора данных возьмем набор с данными о песнях и их характеристиках.

Набор содержит такие колонки как:

- song_name - название песни
- song_popularity - индекс популярности песни
- song_duration_ms - длительность в мс
- acousticness - индекс акустики
- danceability - индекс танцевальности
- energy - индекс энергичности
- instrumentalness - индекс инструментальности
- key - ключ
- liveness - индекс живости
- loudness - индекс громкости
- audio_mode - режим аудио
- speechiness - индекс разговорности
- tempo - темп
- time_signature - временная метка
- audio_valence

Поставим задачу предсказания популярности песни по данным характеристикам. Построим модель машинного обучения для данного набора и решим задачу регрессии.

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

In [4]:

```
data = pd.read_csv('data/song_data.csv', sep=',')
data.head()
```

Out[4]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy
0	Boulevard of Broken Dreams	73	262333	0.005520	0.496	0.682
1	In The End	66	216933	0.010300	0.542	0.853
2	Seven Nation Army	76	231733	0.008170	0.737	0.463
3	By The Way	74	216933	0.026400	0.451	0.970
4	How You Remind Me	56	223826	0.000954	0.447	0.766

In [5]:

```
data.shape
```

Out[5]:

```
(18835, 15)
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['song_name', 'song_popularity', 'song_duration_ms', 'acoustic ness',
                                                'danceability', 'energy', 'instrumentalness', 'key', 'livenes
s',
                                                'loudness', 'audio_mode', 'speechiness', 'tempo', 'time_signa
ture',
         'audio_valence'],
        dtype='object')
```

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
song_name           0
song_popularity     0
song_duration_ms    0
acousticness        0
danceability        0
energy              0
instrumentalness    0
key                 0
liveness            0
loudness            0
audio_mode          0
speechiness         0
tempo               0
time_signature      0
audio_valence       0
dtype: int64
```

In [9]:

```
data.dtypes
```

Out[9]:

```
song_name           object
song_popularity      int64
song_duration_ms     int64
acousticness       float64
danceability       float64
energy             float64
instrumentalness   float64
key                  int64
liveness           float64
loudness           float64
audio_mode           int64
speechiness        float64
tempo              float64
time_signature       int64
audio_valence      float64
dtype: object
```

In [10]:

```
data.describe()
```

Out[10]:

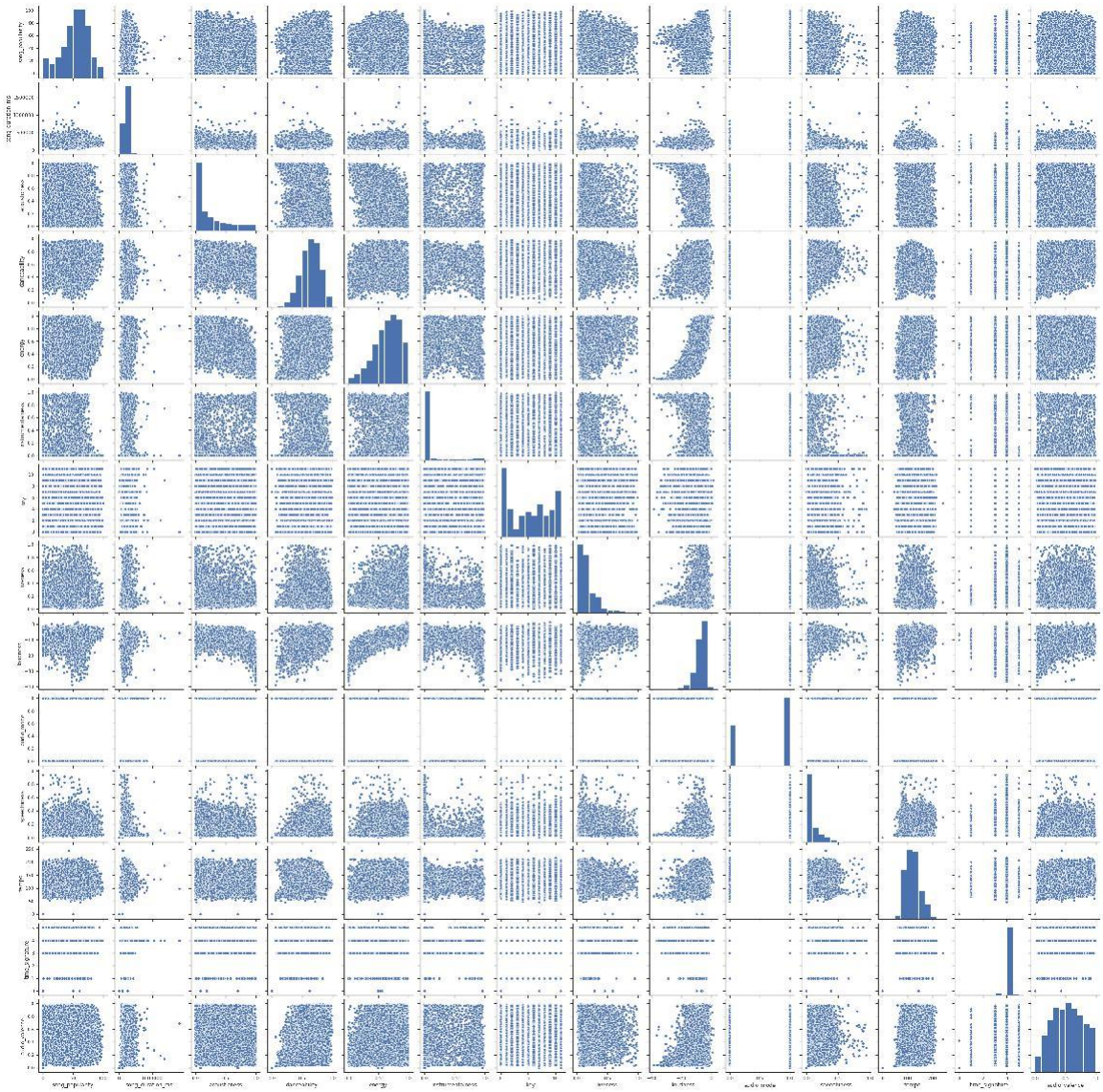
	song_popularity	song_duration_ms	acousticness	danceability	energy
count	18835.000000	1.883500e+04	18835.000000	18835.000000	18835.000000
mean	52.991877	2.182116e+05	0.258539	0.633348	0.644995
std	21.905654	5.988754e+04	0.288719	0.156723	0.214101
min	0.000000	1.200000e+04	0.000001	0.000000	0.001070
25%	40.000000	1.843395e+05	0.024100	0.533000	0.510000
50%	56.000000	2.113060e+05	0.132000	0.645000	0.674000
75%	69.000000	2.428440e+05	0.424000	0.748000	0.815000
max	100.000000	1.799346e+06	0.996000	0.987000	0.999000

In [14]:

```
sns.pairplot(data)
```

Out[14]:

<seaborn.axisgrid.PairGrid at 0x7fbd3ff7a7d0>



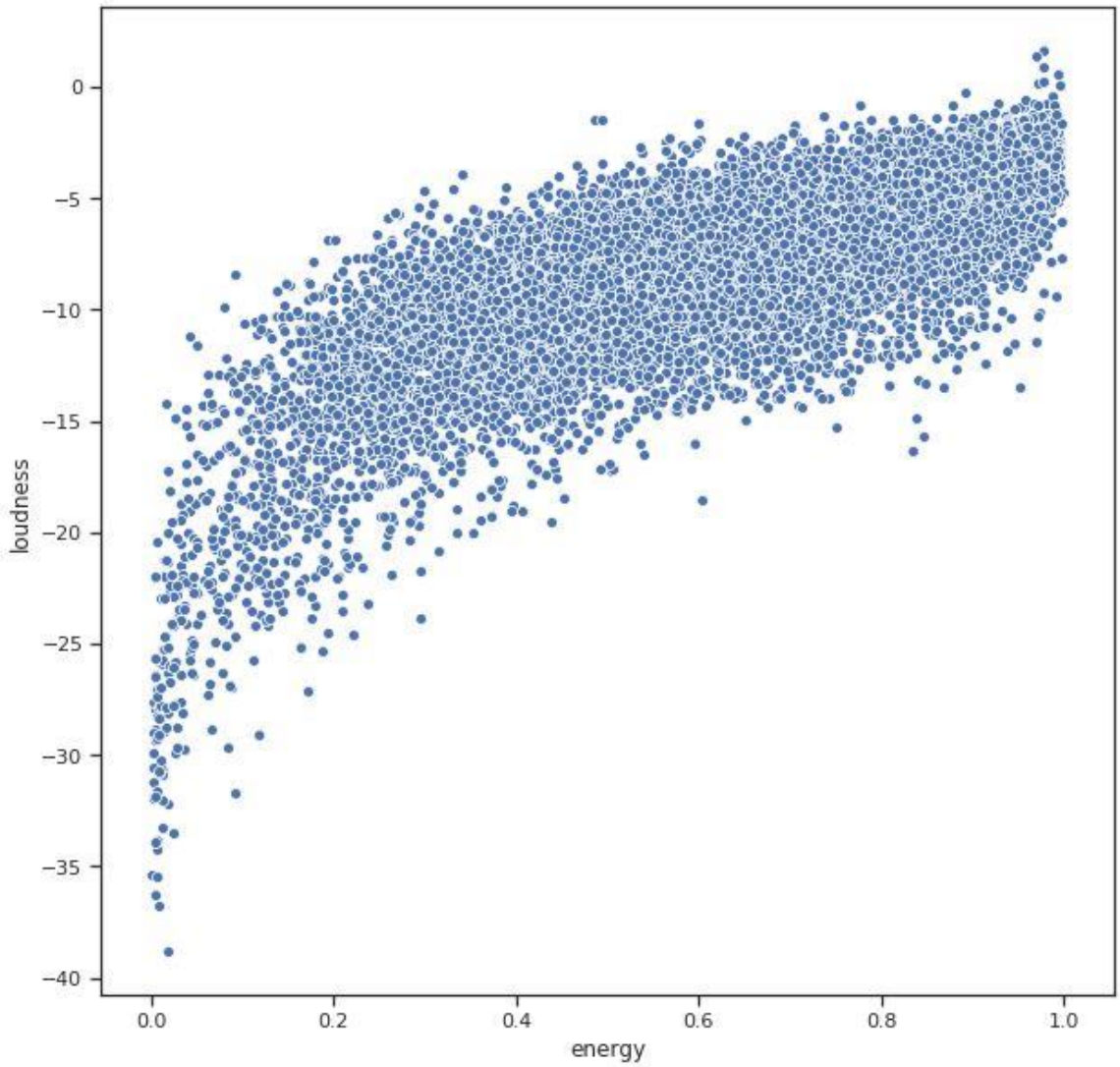
Видим, что наиболее заметна корреляция таких характеристик как громкость и энергичность. В остальных случаях зависимости не такие очевидные.

In [15]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='energy', y='loudness', data=data)
```

Out[15]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fbd3467b410>

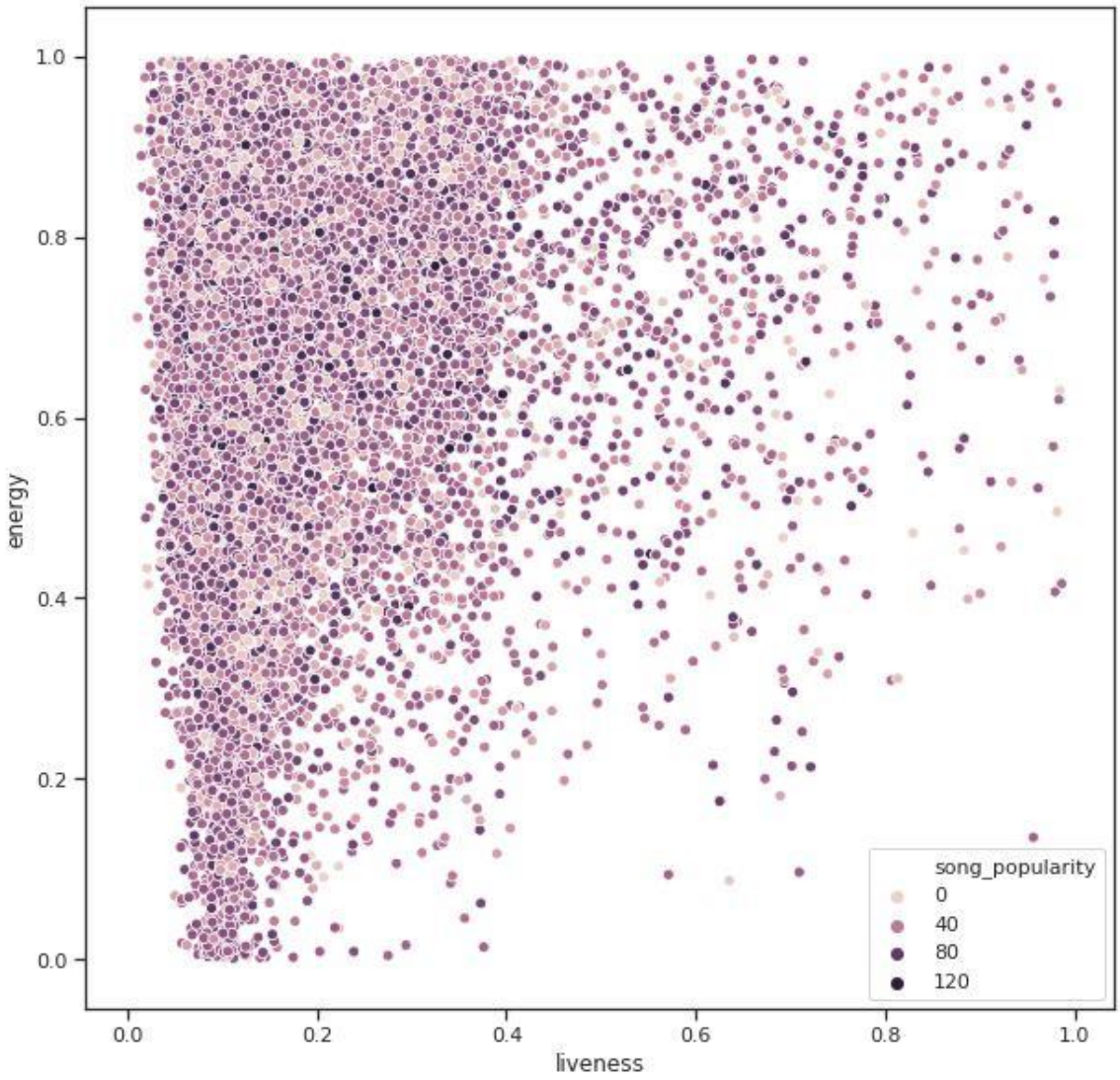


In [22]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='liveness', y='energy', data=data, hue='song_popularit y')
```

Out[22]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fbd32a99d90>

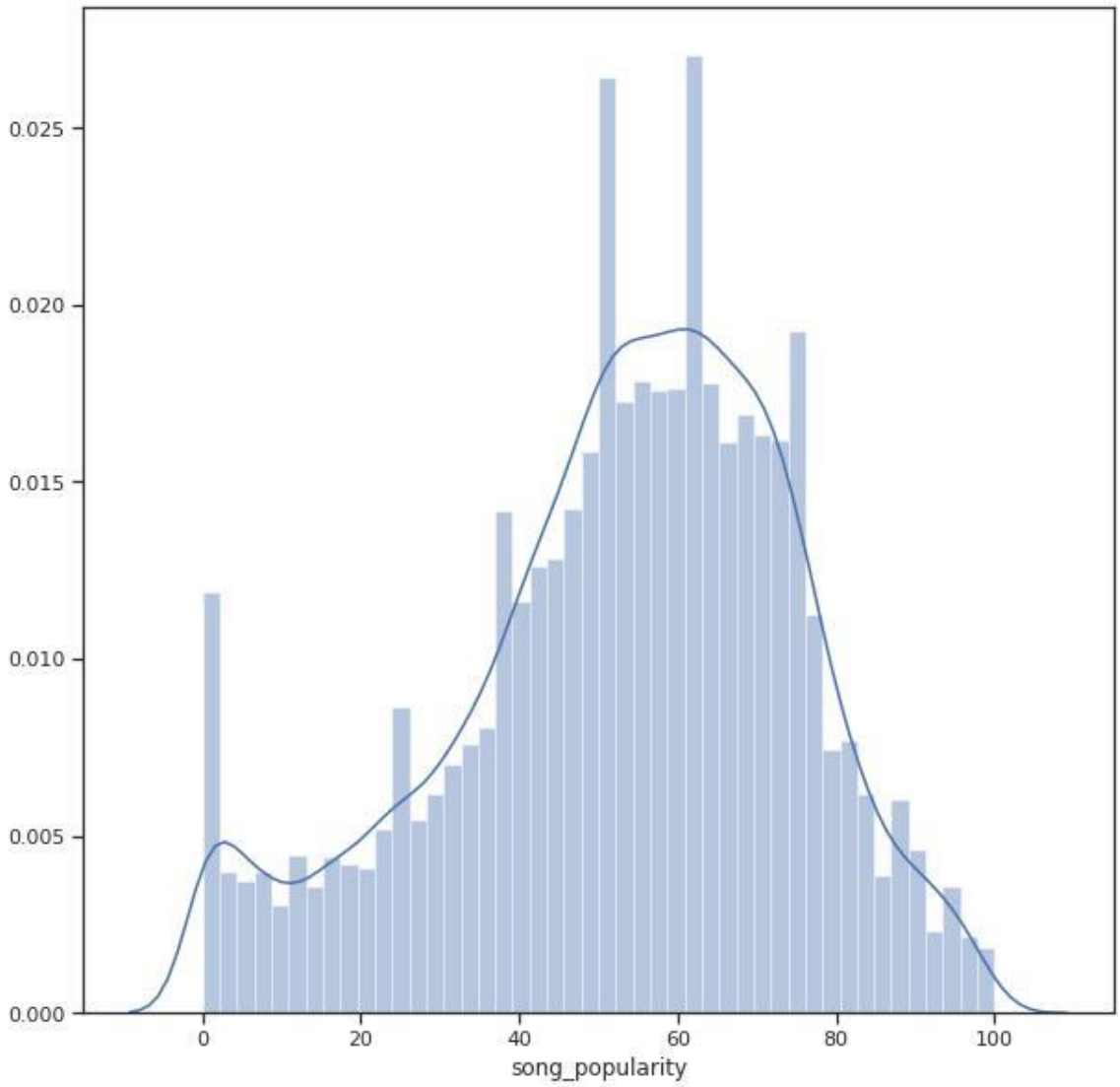


In [23]:

```
fig, ax = plt.subplots(figsize=(10,10))
sns.distplot(data['song_popularity'])
```

Out[23]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fbd32a23a50>



In [24]:

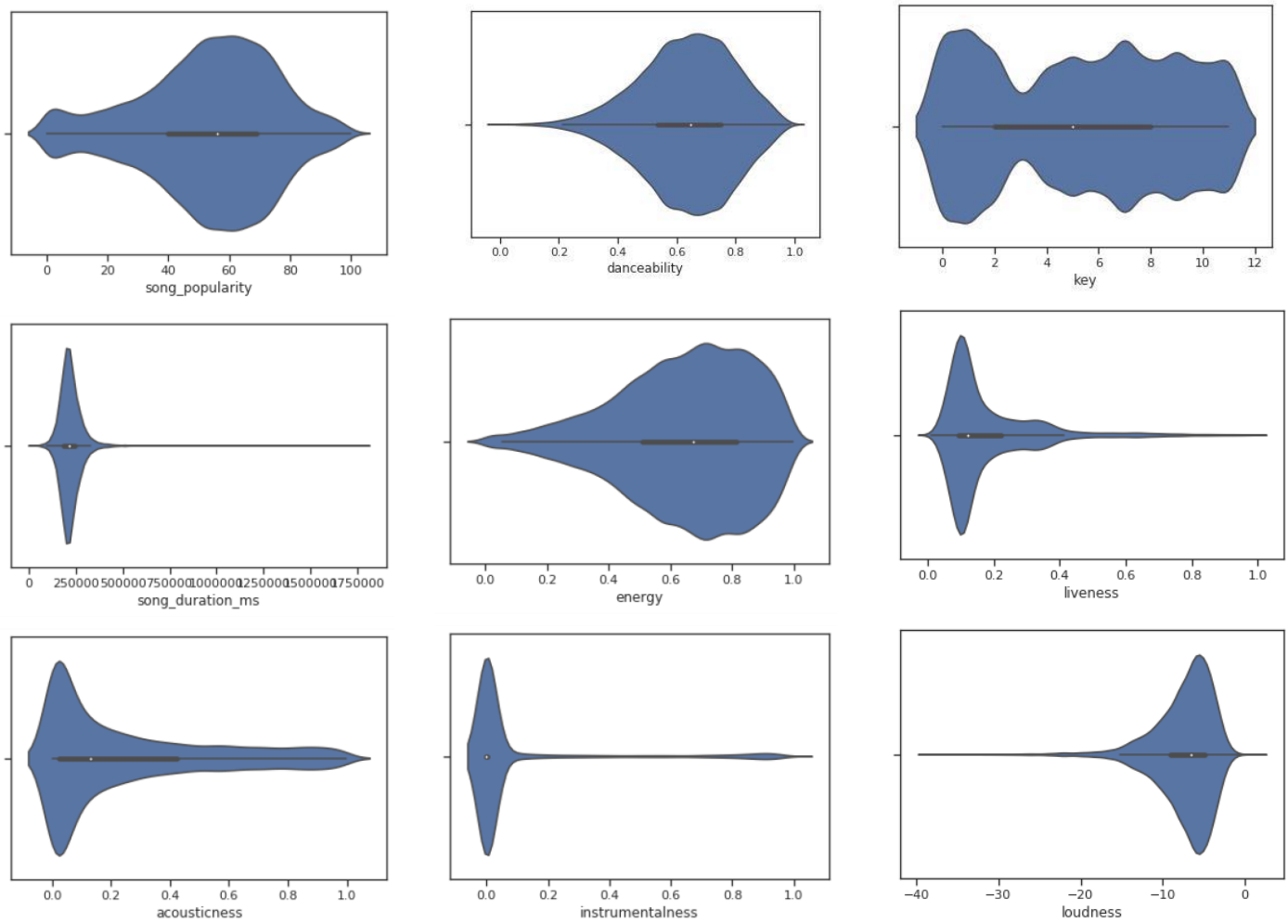
```
data.columns
```

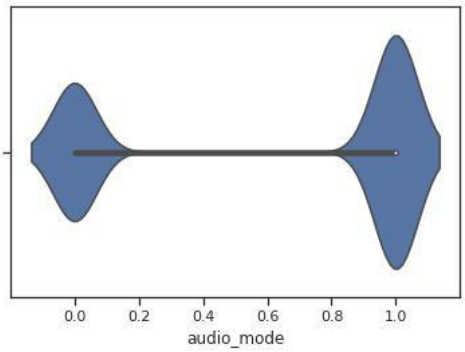
Out[24]:

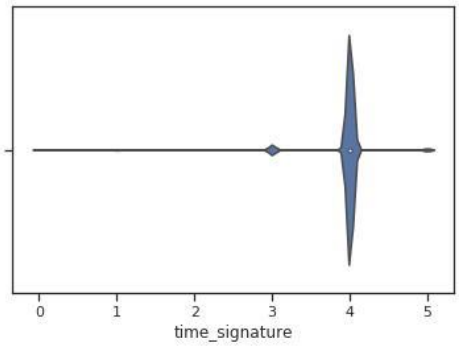
```
Index(['song_name', 'song_popularity', 'song_duration_ms', 'acoustic ness',
                                                'danceability', 'energy', 'instrumentalness', 'key', 'livenes
s',
                                               'loudness', 'audio_mode', 'speechiness', 'tempo', 'time_signa
ture',
         'audio_valence'],
        dtype='object')
```

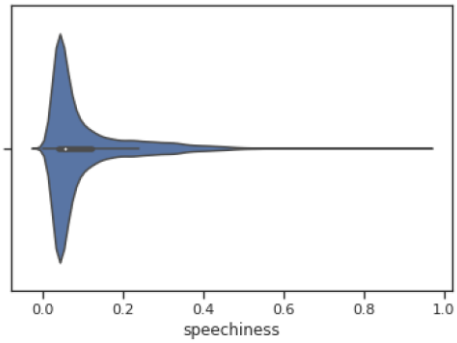
In [26]:

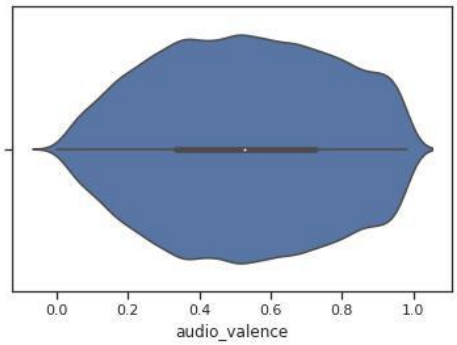
```
#  Скрипичные диаграммы для числовых колонок
for col in [ 'song_popularity', 'song_duration_ms', 'acousticness', 'danceability', 'energy', 'instrumentalness',
             'key', 'liveness', 'loudness', 'audio_mode', 'speechiness', 'tempo', 'time_signature', 'audio_valence']:

    sns.violinplot(x=data[col])
    plt.show()
```

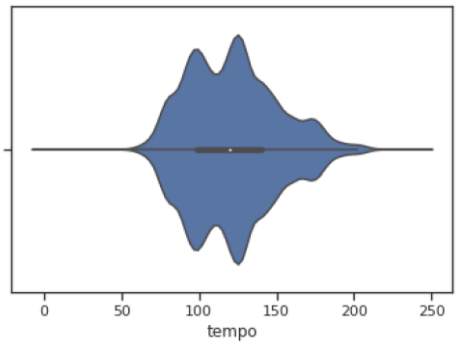












Анализ и заполнение пропусков в данных.

Поскольку в данном наборе пустых значений нет, пропустим данный пункт.

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

Кодирование категориальных признаков числовыми

In [28]:

```
from sklearn.preprocessing import LabelEncoder le =
LabelEncoder()
data['song_name'] = le.fit_transform(data['song_name'])
data.dtypes
```

Out[28]:

```
song_name             int64
song_popularity       int64
song_duration_ms      int64
acousticness        float64
danceability        float64
energy              float64
instrumentalness    float64
key                   int64
liveness            float64
loudness            float64
audio_mode            int64
speechiness         float64
tempo               float64
time_signature        int64
audio_valence       float64
dtype: object
```

In [29]:

```
data.head()
```

Out[29]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy
0	1561	73	262333	0.005520	0.496	0.682
1	5541	66	216933	0.010300	0.542	0.853
2	9638	76	231733	0.008170	0.737	0.463
3	1760	74	216933	0.026400	0.451	0.970
4	4988	56	223826	0.000954	0.447	0.766

Масштабирование данных.

In [30]:

```
scale_cols = ['song_popularity', 'song_duration_ms', 'acousticness', 'danceability', 'energy', 'instrumentalness',
              'key', 'liveness', 'loudness', 'audio_mode', 'speechiness', 'tempo', 'time_signature', 'audio_valence']
```

In [31]:

```
data.columns
```

Out[31]:

```
Index(['song_name', 'song_popularity', 'song_duration_ms', 'acoustic ness',
       'danceability', 'energy', 'instrumentalness', 'key', 'livenes
s',
       'loudness', 'audio_mode', 'speechiness', 'tempo', 'time_signa
ture',
       'audio_valence'],
      dtype='object')
```

In [32]:

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

In [33]:

```
#  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]
```

In [34]:

```
data.head()
```

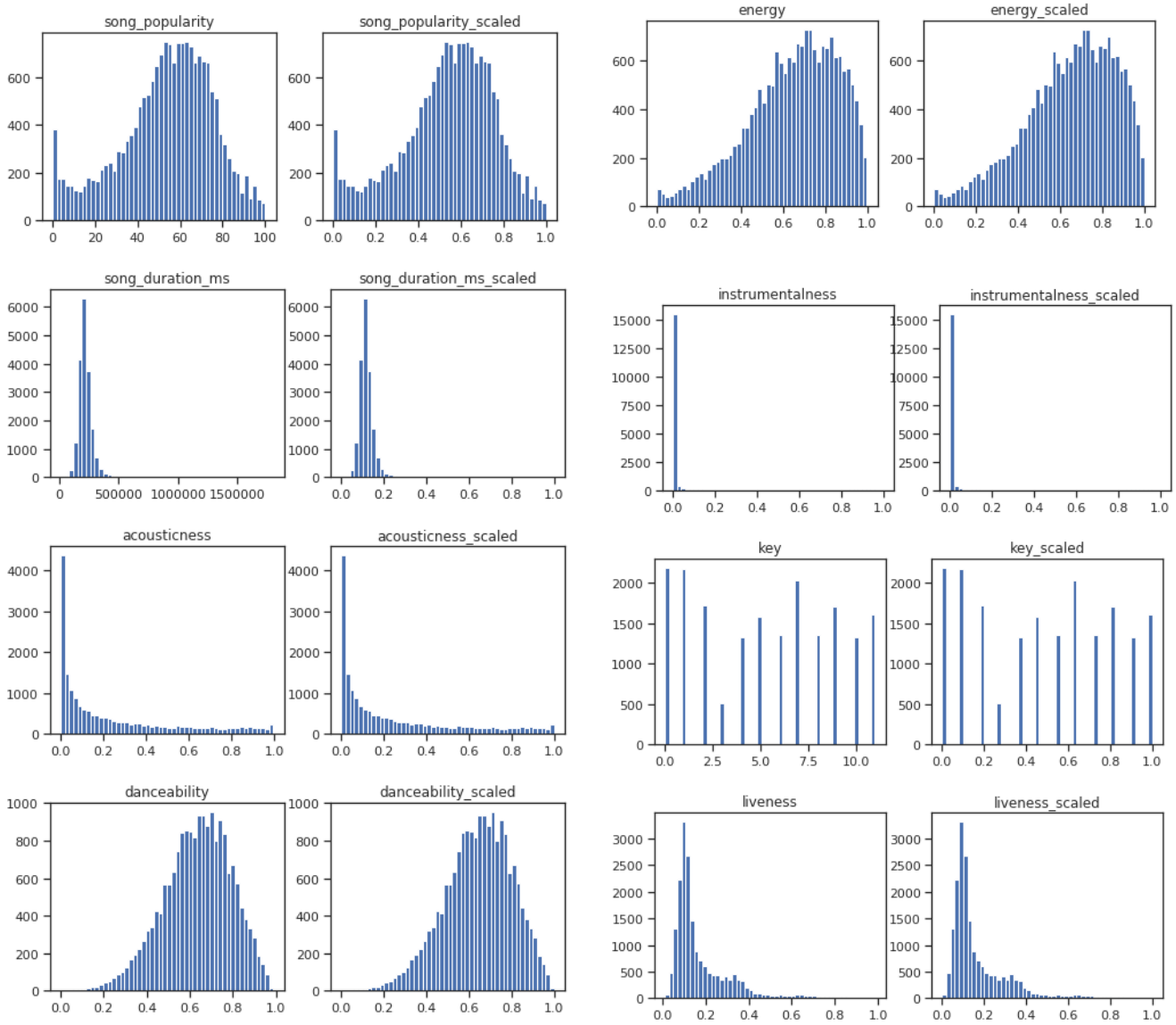
Out[34]:

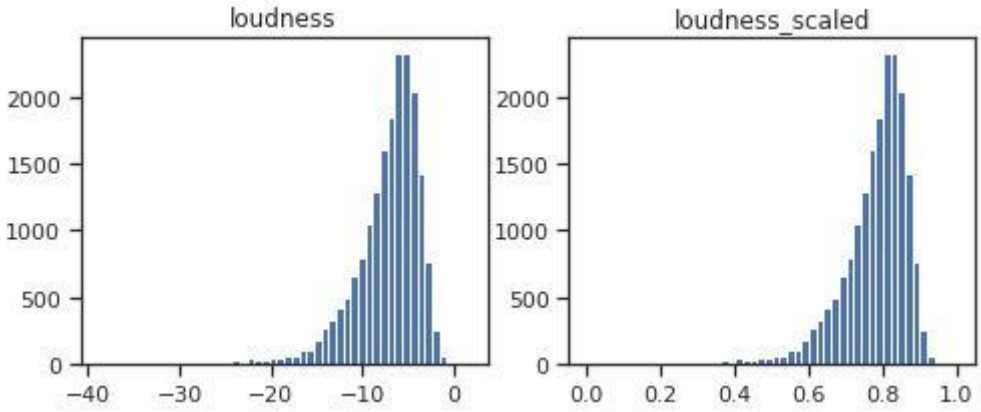
	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy
0	1561	73	262333	0.005520	0.496	0.682
1	5541	66	216933	0.010300	0.542	0.853
2	9638	76	231733	0.008170	0.737	0.463
3	1760	74	216933	0.026400	0.451	0.970
4	4988	56	223826	0.000954	0.447	0.766

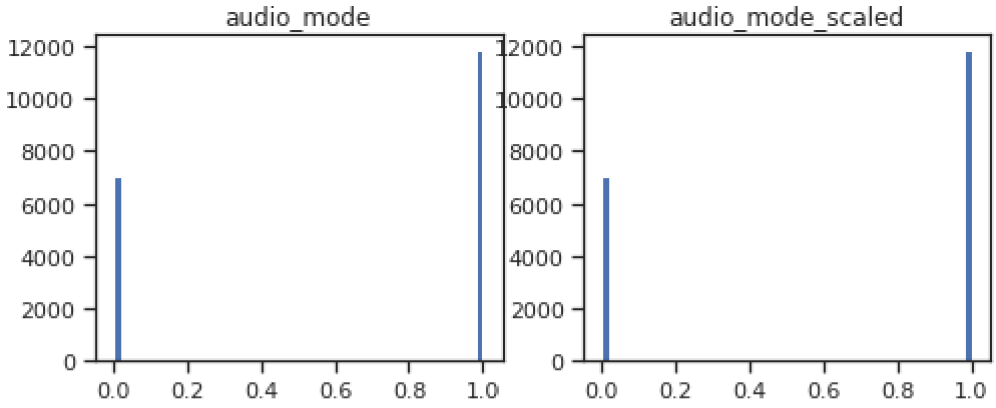
5 rows × 29 columns

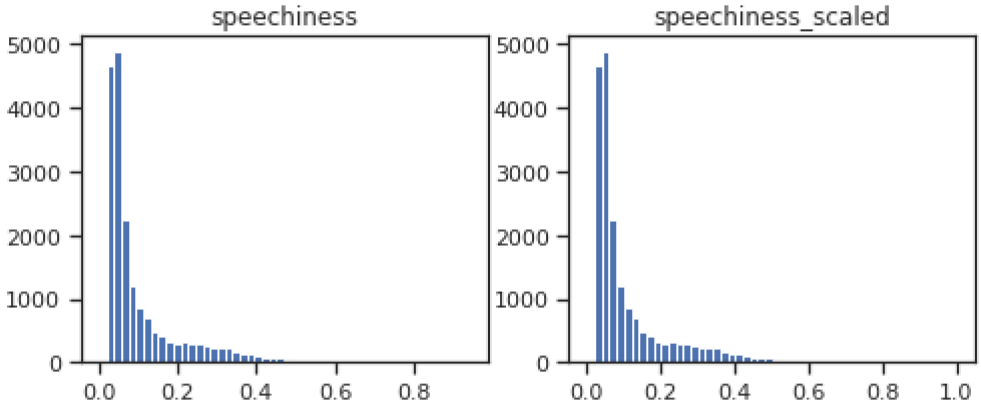
In [35]:

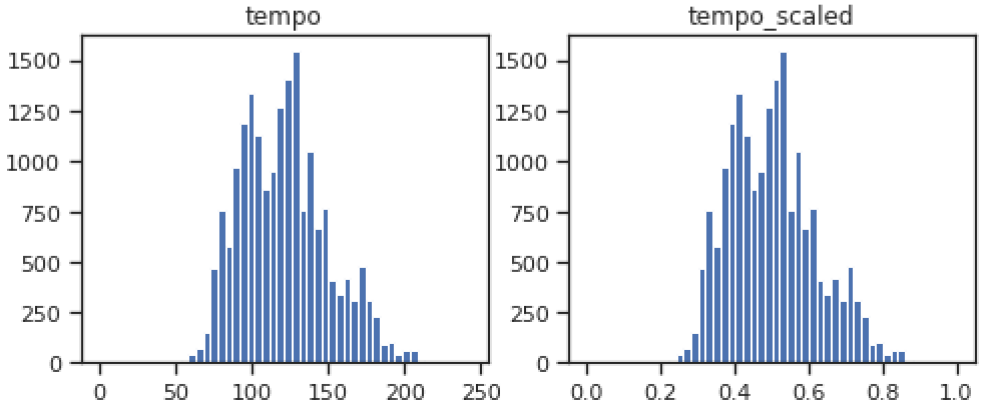
```
#  Проверим, что масштабирование не повлияло на распределение данных for col in
scale_cols:
    col_scaled = col + '_scaled'

    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()
```

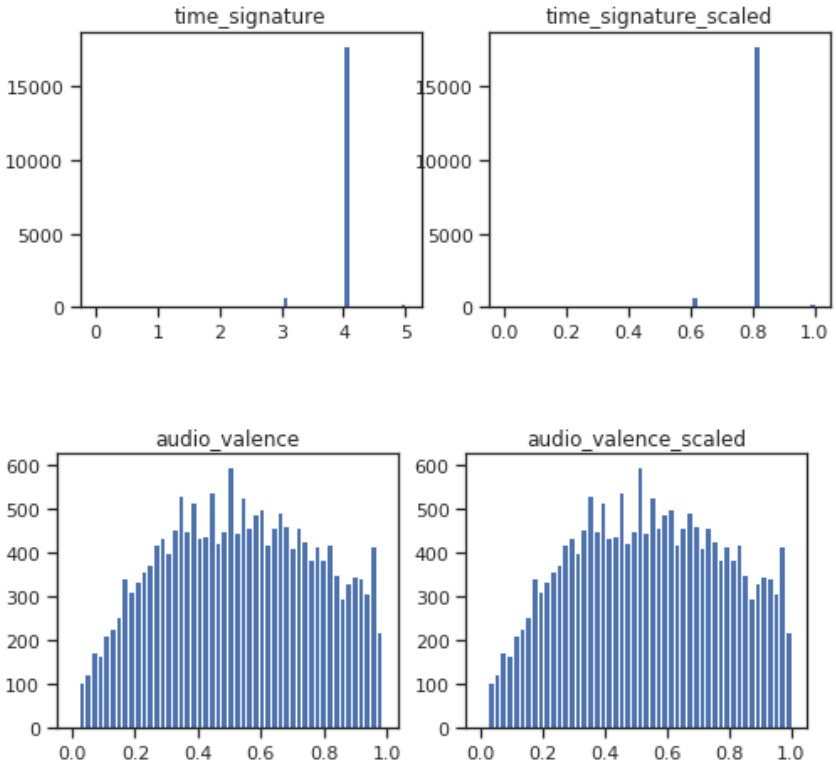



loudness
loudness_scaled


audio_mode
audio_mode_scaled


speechiness
speechiness_scaled


tempo
tempo_scaled



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

In [36]:

```
#  Воспользуемся наличием тестовых выборок,
#  включив их в корреляционную матрицу
corr_cols_1 = scale_cols
corr_cols_1
```

Out[36]:

```
['song_popularity',
  'song_duration_ms',
  'acousticness',
  'danceability',
  'energy',
  'instrumentalness',
  'key',
  'liveness',
  'loudness',
  'audio_mode',
  'speechiness',
  'tempo',
  'time_signature',
  'audio_valence']
```

In [37]:

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

Out[37]:

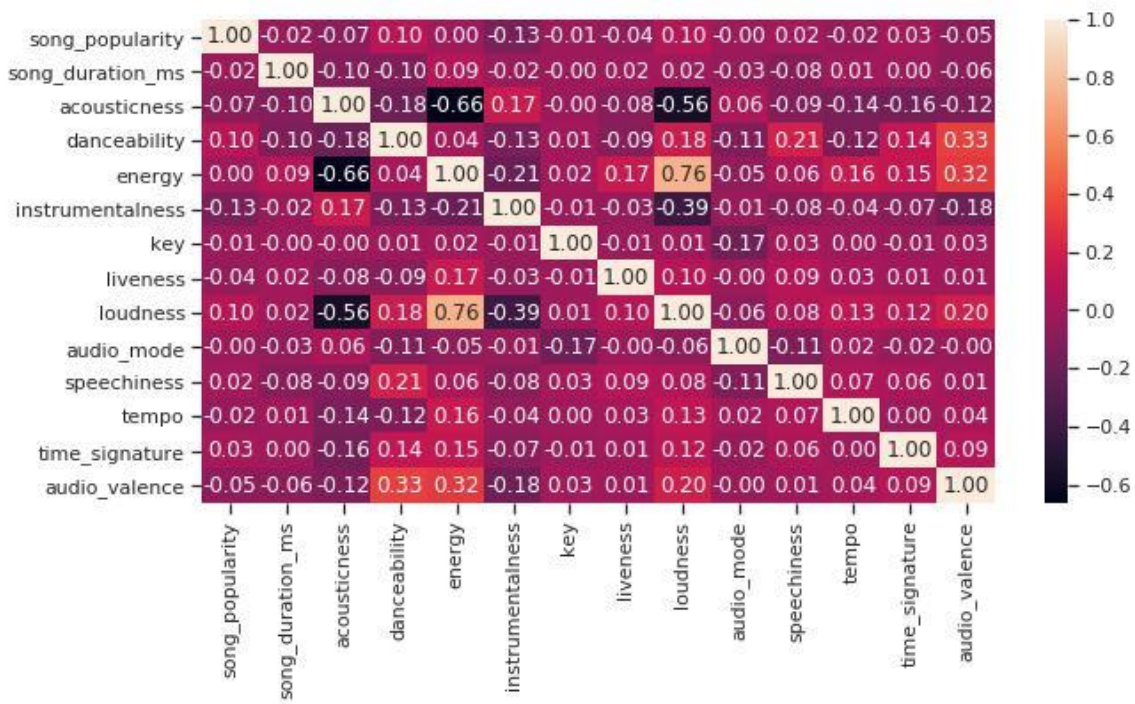
```
['song_popularity_scaled',
 'song_duration_ms_scaled',
 'acousticness_scaled',
 'danceability_scaled',
 'energy_scaled',
 'instrumentalness_scaled',
 'key_scaled',
 'liveness_scaled',
 'loudness_scaled',
 'audio_mode_scaled',
 'speechiness_scaled',
 'tempo_scaled',
 'time_signature_scaled',
 'audio_valence_scaled']
```

In [38]:

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

Out[38]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fbd31befc90>

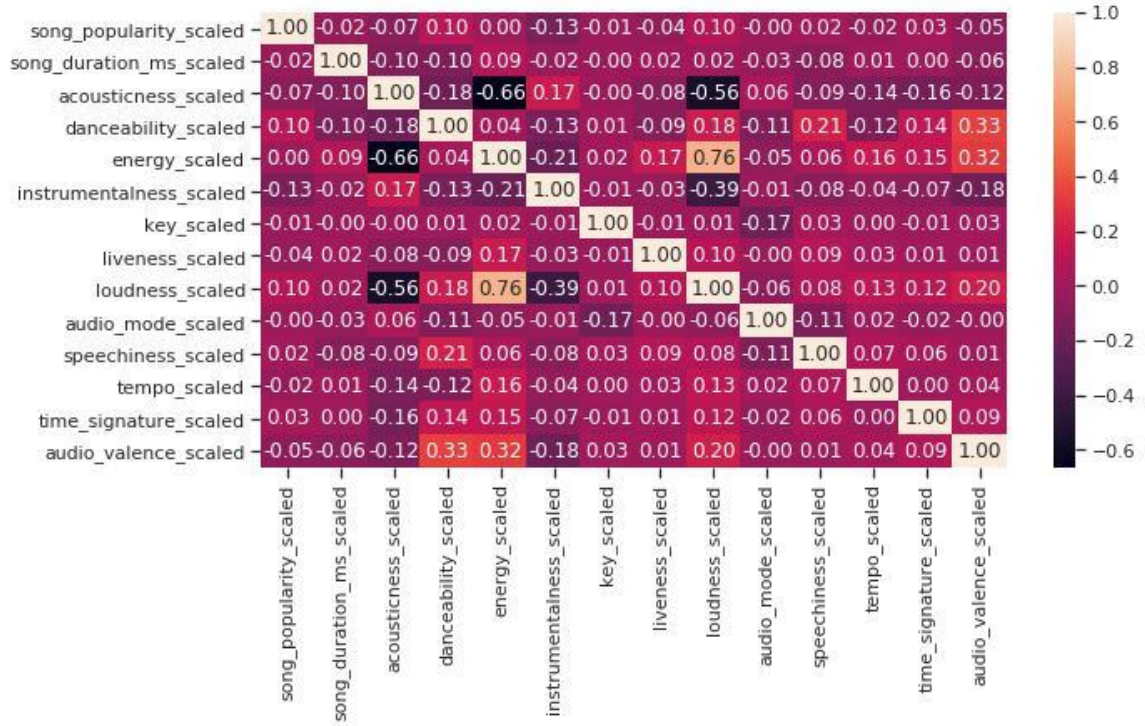


In [39]:

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

Out[39]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fbd31da5fd0>



- Видим, что популярность песни не сильно коррелирует с данными характеристиками. Наибольшее влияние на популярность оказывают такие признаки как танцевальность трека и громкость.
- Наибольшую корреляцию видим между громкостью и энергичностью трека, как и во 2 пункте.

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

Возьмем метрики MAE, Median Absolute Error и R^2.

- MAE (Mean Absolute Error) — это среднее абсолютное значение ошибки(среднее модуля ошибки). Данная метрика удобна, так как показывает среднюю ошибку, но при этом не так чувствительна к выбросам, как, например, MSE.
- Медиана абсолютного отклонения(Median Absolute Error) - это альтернатива стандартного отклонения, но она менее чувствительна к воздействию промахов, чем среднее отклонение.
- Коэффициент детерминации, или R^2 покажет насколько модель соответствует или не соответствует данным.

In [41]:

```
class MetricLogger:

    def __init__(self):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
            'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        #   Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)]. index, inplace = True)
        #   Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}] self.df =
        self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def plot(self, str_header, metric, ascending=True, figsize=(5, 5)):
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric, ascending)
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                                    align='center',
                                    height=0.5,
                                    tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
        plt.show()
```

6. Выбор наиболее подходящих моделей для решения задачи регрессии.

- Возьмем модели случайный лес и дерево решений, поскольку в проведенных экспериментах в лабораторных работах случайный лес показал себя наилучшим образом. Результаты, которые удалось получить при помощи данной модели были соспоставимы с результатами самых сильных среди протестированных ансамблевых моделей. Дерево решений так же дает хорошие результаты по сравнению с, например, линейными моделями.
- В качестве ансамблевой модели возьмем лучшую модель, полученную при выполнении 6 лабораторной работы: 'TREE+RF=>LR', то есть на первом уровне у нас будут две модели: дерево и случайный лес, а на втором уровне - линейная регрессия.

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

In [42]:

```
data1 = shuffle(data)
data1
```

Out[42]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	en
5764	12054	50	204213	0.8570	0.543	0
16725	8262	33	160693	0.0027	0.700	0
5593	4559	100	214289	0.1910	0.687	0
2612	12841	63	246186	0.0133	0.546	0
2528	4667	7	231800	0.0193	0.646	0
...	...	...	...	...	...	
15148	45	54	228240	0.3800	0.745	0
7191	11719	72	233028	0.2000	0.667	0
12766	1204	47	119360	0.0387	0.647	0
3779	4837	66	307200	0.2250	0.932	0
3085	12698	24	161751	0.5160	0.738	0

18835 rows × 29 columns

In [43]:

```
len(data1)
```

Out[43]:

18835

In [46]:

```
#  На основе масштабированных данных выделим
#  обучающую и тестовую выборки
train_data_all = data1[:13000]
test_data_all = data1[13001:]
train_data_all.shape, test_data_all.shape
```

Out[46]:

((13000, 29), (5834, 29))

In [47]:

```
data.columns
```

Out[47]:

```
Index(['song_name', 'song_popularity', 'song_duration_ms', 'acoustic ness',
       'danceability', 'energy', 'instrumentalness', 'key', 'livenes
s',
       'loudness', 'audio_mode', 'speechiness', 'tempo', 'time_signa
ture',
       'audio_valence', 'song_popularity_scaled', 'song_duration_ms_ scaled',
       'acousticness_scaled', 'danceability_scaled', 'energy_scale
d',
       'instrumentalness_scaled', 'key_scaled', 'liveness_scaled', 'loudness_scaled',
       'audio_mode_scaled', 'speechiness_scaled', 'tempo_scaled', 'time_signature_scaled',
       'audio_valence_scale
d'],
      dtype='object')
```

In [159]:

```
#  Признаки для задачи регресии (опустим название) task_regr_cols =
['song_duration_ms', 'acousticness',
       'danceability', 'energy', 'instrumentalness', 'key', 'liveness', 'loudness', 'audio_mode', 'speechiness',
       'tempo', 'time_signature', 'audio_valence', 'song_duration_ms_scaled', 'acousticness_scaled',
       'danceability_scaled', 'energy_scaled', 'instrumentalness_scaled', 'key_scaled', 'liveness_scaled',
       'loudness_scaled', 'audio_mode_scaled', 'speechiness_scaled', 'tempo_scaled',
       'time_signature_scaled', 'audio_valence_scaled']
```

In [160]:

```
#  Выборки для задачи регресии
regr_X_train = train_data_all[task_regr_cols]
regr_X_test = test_data_all[task_regr_cols]
regr_Y_train = train_data_all['song_popularity']
regr_Y_test = test_data_all['song_popularity']
regr_X_train.shape, regr_X_test.shape, regr_Y_train.shape, regr_Y_test.shape
```

Out[160]:

```
((13000, 26), (5834, 26), (13000,), (5834,))
```

8. Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

In [284]:

```
#  Модели
regr_models = {'Tree':DecisionTreeRegressor(max_depth=10),
                       'RF':RandomForestRegressor(max_depth=10, n_estimators=30),
                      }
```

In [285]:

```
#  Сохранение метрик regrMetricLogger =
MetricLogger()
```

In [286]:

```
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)
    medae = median_absolute_error(regr_Y_test, Y_pred)
    r2 = r2_score(regr_Y_test, Y_pred)

    regrMetricLogger.add('MAE', model_name, mae)
    regrMetricLogger.add('MedAE', model_name, medae)
    regrMetricLogger.add('R2', model_name, r2)

    print('**********************************************************')
    print(model)
    print()
    print('MAE={}, MedAE={}, R2={}'.format(
        round(mae, 3), round(medae, 3), round(r2, 3)))
    print('**********************************************************')
```

In [287]:

```
for model_name, model in regr_models.items():
    regr_train_model(model_name, model, regrMetricLogger)
```

```
**********************************************************
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=10,
                      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='depreca
ted',
                      random_state=None, splitter='best')

MAE=16.609, MedAE=13.63, R2=0.034
**********************************************************
**********************************************************
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='ms e',
                                                  max_depth=10, max_features='auto', max_leaf_no
des=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=30, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=Fals
e)

MAE=15.211, MedAE=12.932, R2=0.222
**********************************************************
```

Ансамблевая модель

In [335]:

```
from heamy.estimator import Regressor
from heamy.pipeline import ModelsPipeline
from heamy.dataset import Dataset
#  набор данных
dataset = Dataset(regr_X_train, regr_Y_train, regr_X_test)
#  Возьмем лучшую модель: 'TREE+RF=>LR'
#  модели первого уровня
model_tree = Regressor(dataset=dataset, estimator=DecisionTreeRegressor, paramet ers={'max_depth':10},name='tree')
model_lr = Regressor(dataset=dataset, estimator=LinearRegression, name='lr') model_rf = Regressor(dataset=dataset,
estimator=RandomForestRegressor, parameter s={'max_depth':10},name='rf')

#  Первый уровень - две модели: дерево и случайный лес
#  Второй уровень: линейная регрессия
pipeline = ModelsPipeline(model_tree, model_rf)
stack_ds = pipeline.stack(k=10, seed=1)
#  модель второго уровня
stacker = Regressor(dataset=stack_ds, estimator=LinearRegression)
results = stacker.validate(k=10,scorer=mean_absolute_error)
print()

results = stacker.validate(k=10,scorer=median_absolute_error)
```

```
Metric: mean_absolute_error
Folds accuracy: [14.89643993242869, 15.616762658202559, 14.972788489 493295,
15.222334189473978, 15.080414157836218, 14.75906546781935, 1 5.05850462402728,
15.044001090351392, 14.8316297901197, 15.548169842 390033]
Mean accuracy: 15.103011024214249
Standard Deviation: 0.2709048887355547
Variance: 0.07338945874082325

Metric: median_absolute_error
Folds accuracy: [12.277784575440783, 13.328390040154055, 12.16766382 3524233,
12.946816156161724, 13.11487250948803, 12.111798154068914, 12.78946051322334,
13.216714632887722, 12.85310829872179, 12.9660593 07565189]
Mean accuracy: 12.777266801123577
Standard Deviation: 0.4177398537009336
Variance: 0.1745065853700774
```

9. Подбор гиперпараметров для выбранных моделей.

Случайный лес

In [173]:

```
RandomForestRegressor().get_params()
```

Out[173]:

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'mse',
 '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}
```

In [297]:

```
n_range = np.array(range(0,50,5))
tuned_parameters = [{'max_depth': n_range}] tuned_parameters
```

Out[297]:

```
[{'max_depth': array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45])}]
```

In [298]:

```
%%time
rf_gs = GridSearchCV(RandomForestRegressor(), tuned_parameters, cv=5, scoring='n eg_mean_squared_error')
rf_gs.fit(regr_X_train, regr_Y_train)
```

/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est imator fit failed. The score on this train-test partition for these parameters will be set to nan. Details: ValueError: max_depth must be greater than zero.

    FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

    FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

    FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

    FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

    FitFailedWarning)

CPU times: user 15min 41s, sys: 1.08 s, total: 15min 42s
Wall time: 15min 44s

Out[298]:

```
GridSearchCV(cv=5, error_score=nan, estimator=RandomForestRegressor(bootstrap=True, ccp_alp
ha=0.0,
                                                                    criterion='mse', max_de
pth=None,
                                                                    max_features='auto',
                                                                    max_leaf_nodes=None,
                                                                    max_samples=None,
                                                                    min_impurity_decrease=
0.0,
                                                                    min_impurity_split=Non
e,
                                                                    min_samples_leaf=1,
                                                                    min_samples_split=2,
                                                                    min_weight_fraction_lea
f=0.0,
                                                                     n_estimators=100, n_job
s=None,
                                                                    oob_score=False, random
_state=None,
                                                                    verbose=0, warm_start=F
alse),
                   iid='deprecated', n_jobs=None,
                   param_grid=[{'max_depth': array([ 0, 5, 10, 15, 20, 2 5, 30, 35, 40, 45])}],
                              pre_dispatch='2*n_jobs', refit=True, return_train_score
=False,
                   scoring='neg_mean_squared_error', verbose=0)
```

In [299]:

```
# Лучшая модель
rf_gs.best_estimator_
```

Out[299]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='ms e',
                                                    max_depth=40, max_features='auto', max_leaf_no
des=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=Fals
e,
                                 random_state=None, verbose=0, warm_start=Fals
e)
```

In [300]:

```
# Лучшее значение параметров
rf_gs.best_params_
```

Out[300]:

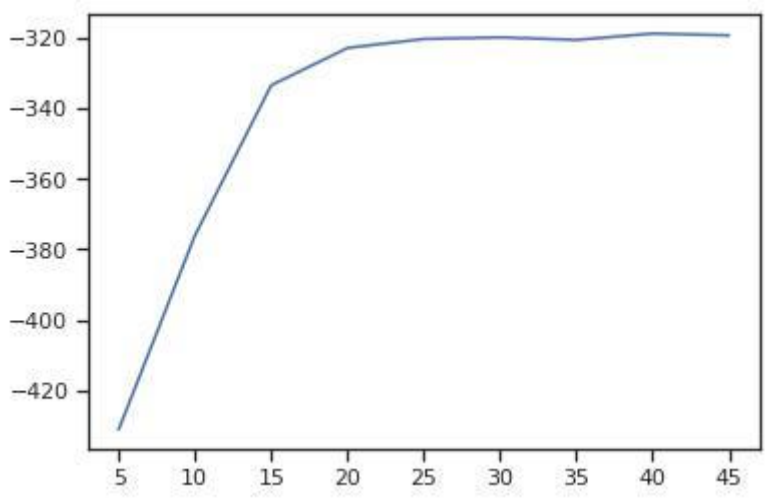
```
{'max_depth': 40}
```

In [301]:

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

Out[301]:

[<matplotlib.lines.Line2D at 0x7fbd32307e10>]



Дерево

In [204]:

```
DecisionTreeRegressor().get_params()
```

Out[204]:

```
{'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'}
```

In [292]:

```
n_range = np.array(range(0,50,5))

tuned_parameters = [{'max_depth': n_range}] tuned_parameters
```

Out[292]:

```
[{'max_depth': array([ 0,  5, 10, 15, 20, 25, 30, 35, 40, 45])}]
```

In [293]:

```
%%time
dt_gs = GridSearchCV(DecisionTreeRegressor(), tuned_parameters, cv=5, scoring='n eg_mean_squared_error')
dt_gs.fit(regr_X_train, regr_Y_train)
```

/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est imator fit failed. The score on this train-test partition for these parameters will be set to nan. Details: ValueError: max_depth must be greater than zero.

  FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

  FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

  FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

  FitFailedWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/model_selection/_validation.py:536: FitFailedWarning: Est
imator fit failed. The score on this train-test partition for these
parameters will be set to nan. Details:
ValueError: max_depth must be greater than zero.

  FitFailedWarning)

CPU times: user 15.1 s, sys: 4.24 ms, total: 15.1 s
Wall time: 15.2 s

Out[293]:

```
GridSearchCV(cv=5, error_score=nan, estimator=DecisionTreeRegressor(ccp_alpha=0.0, criterio
n='mse',
                                                                    max_depth=None, max_fea
tures=None,
                                                                    max_leaf_nodes=None,
                                                                    min_impurity_decrease=
0.0,
                                                                    min_impurity_split=Non
e,
                                                                    min_samples_leaf=1,
                                                                    min_samples_split=2,
                                                                    min_weight_fraction_lea
f=0.0,
                                                                    presort='deprecated',
                                                                    random_state=None,
                                                                    splitter='best'),
                   iid='deprecated', n_jobs=None,
                   param_grid=[{'max_depth': array([ 0, 5, 10, 15, 20, 2 5, 30, 35, 40, 45])}],
                              pre_dispatch='2*n_jobs', refit=True, return_train_score
=False,
                   scoring='neg_mean_squared_error', verbose=0)
```

In [294]:

```
# Лучшая модель
dt_gs.best_estimator_
```

Out[294]:

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=5,
                                  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='depreca
ted',
                                              random_state=None, splitter='best')
```

In [295]:

```
# Лучшее значение параметров
dt_gs.best_params_
```

Out[295]:

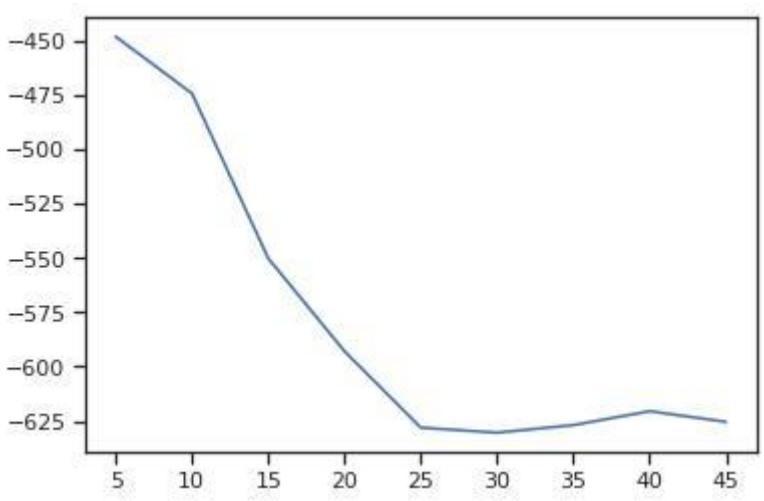
```
{'max_depth': 5}
```

In [296]:

```
#  Изменение качества на тестовой выборке
plt.plot(n_range, dt_gs.cv_results_['mean_test_score'])
```

Out[296]:

[<matplotlib.lines.Line2D at 0x7fbd31789410>]



Ансамблевая модель

Поскольку параметры для случайного леса и дерева уже подобрали, то воспользуемся ими, а так же попробуем подобрать еще 2 параметра для данных моделей.

Decision tree

In [310]:

```
n_range = [0, 0.5, 1, 1.5, 2, 2.5, 3]
tuned_parameters = [{'min_impurity_split': n_range}] tuned_parameters
```

Out[310]:

[{'min_impurity_split': [0, 0.5, 1, 1.5, 2, 2.5, 3]}]

In [320]:

```
%%time
ens_dt_gs = GridSearchCV(DecisionTreeRegressor(), tuned_parameters, cv=5, scorin g='neg_mean_squared_error')
ens_dt_gs.fit(regr_X_train, regr_Y_train)
```

/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl it parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_im purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag

es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl it parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_im purity_decrease parameter instead.

```
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
```

it parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_im purity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7 to 0 in version 0.23, and it will be removed in 0.25. Use the min_impurity_decrease parameter instead.
    FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packages/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_split parameter is deprecated. Its default value will change from 1e-7

```
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)
/home/lisobol/tensorflow_env/my_tensorflow/lib/python3.7/site-packag
es/sklearn/tree/_classes.py:301: FutureWarning: The min_impurity_spl
it parameter is deprecated. Its default value will change from 1e-7
to 0 in version 0.23, and it will be removed in 0.25. Use the min_im
purity_decrease parameter instead.
  FutureWarning)


CPU times: user 14 s, sys: 4.05 ms, total: 14 s
Wall time: 14 s
```

Out[320]:

```
GridSearchCV(cv=5, error_score=nan, estimator=DecisionTreeRegressor(ccp_alpha=0.0, criterio
n='mse',
                                                                    max_depth=None, max_fea
tures=None,
                                                                    max_leaf_nodes=None,
                                                                    min_impurity_decrease=
0.0,
                                                                    min_impurity_split=Non
e,
                                                                    min_samples_leaf=1,
                                                                    min_samples_split=2,
                                                                    min_weight_fraction_lea
f=0.0,
                                                                    presort='deprecated',
                                                                    random_state=None,
                                                                    splitter='best'),
                   iid='deprecated', n_jobs=None,
                   param_grid=[{'min_impurity_split': [0, 0.5, 1, 1.5, 2,
2.5, 3]}],
                               pre_dispatch='2*n_jobs', refit=True, return_train_score
=False,
                   scoring='neg_mean_squared_error', verbose=0)
```

In [321]:

```
# Лучшая модель
ens_dt_gs.best_estimator_
```

Out[321]:

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=Non e,
                                  max_features=None, max_leaf_nodes=None,
                                  min_impurity_decrease=0.0, min_impurity_split=
1.5,
                                  min_samples_leaf=1, min_samples_split=2,
                                  min_weight_fraction_leaf=0.0, presort='depreca
ted',
                                          random_state=None, splitter='best')
```

In [322]:

```
# Лучшее значение параметров
ens_dt_gs.best_params_
```

Out[322]:

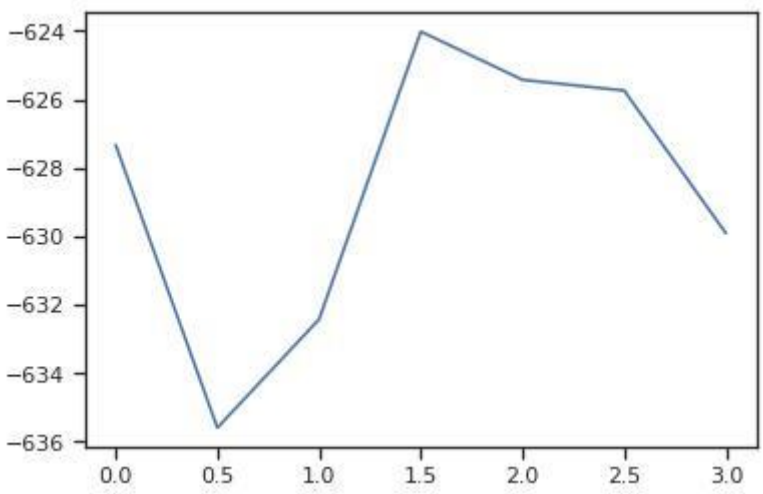
```
{'min_impurity_split': 1.5}
```

In [323]:

```
#  Изменение качества на тестовой выборке
plt.plot(n_range, ens_dt_gs.cv_results_['mean_test_score'])
```

Out[323]:

[<matplotlib.lines.Line2D at 0x7fbd313be850>]



Random Forest

In [324]:

```
n_range = [1, 5, 10, 20, 30, 40 ,50 ,60]
tuned_parameters = [{'n_estimators': n_range}] tuned_parameters
```

Out[324]:

[{'n_estimators': [1, 5, 10, 20, 30, 40, 50, 60]}]

In [326]:

```
%%time
ens_rf_gs = GridSearchCV(RandomForestRegressor(), tuned_parameters, cv=5, scorin g='neg_mean_squared_error')
ens_rf_gs.fit(regr_X_train, regr_Y_train)
```

CPU times: user 4min 33s, sys: 95.4 ms, total: 4min 33s
Wall time: 4min 33s

Out[326]:

```
GridSearchCV(cv=5, error_score=nan, estimator=RandomForestRegressor(bootstrap=True, ccp_alp
ha=0.0,
                                                                    criterion='mse', max_de
pth=None,
                                                                    max_features='auto',
                                                                    max_leaf_nodes=None,
                                                                    max_samples=None,
                                                                    min_impurity_decrease=
0.0,
                                                                    min_impurity_split=Non
e,
                                                                    min_samples_leaf=1,
                                                                    min_samples_split=2,
                                                                    min_weight_fraction_lea
f=0.0,
                                                                    n_estimators=100, n_job
s=None,
                                                                    oob_score=False, random
_state=None,
                                                                    verbose=0, warm_start=F
alse),
                   iid='deprecated', n_jobs=None,
                   param_grid=[{'n_estimators': [1, 5, 10, 20, 30, 40, 50,
60]}],
                              pre_dispatch='2*n_jobs', refit=True, return_train_score
=False,
                   scoring='neg_mean_squared_error', verbose=0)
```

In [328]:

```
# Лучшая модель
ens_rf_gs.best_estimator_
```

Out[328]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='ms e',
                      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=60, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=Fals
e)
```

In [329]:

```
# Лучшее значение параметров
ens_rf_gs.best_params_
```

Out[329]:

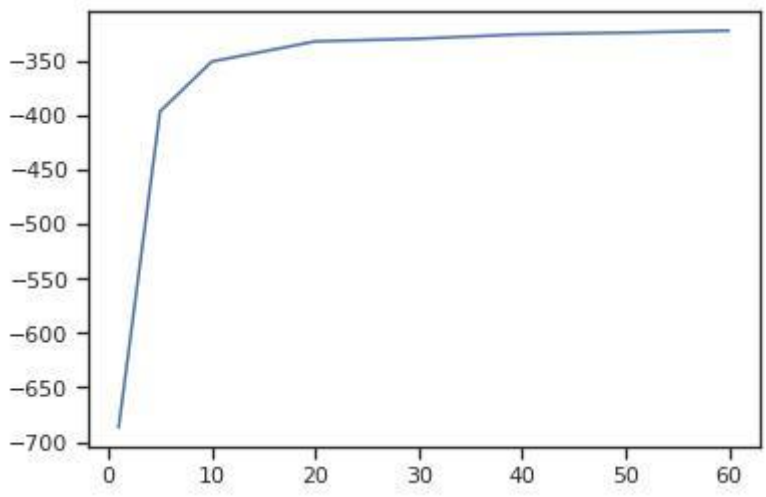
{'n_estimators': 60}

In [330]:

```
# Изменение качества на тестовой выборке
plt.plot(n_range, ens_rf_gs.cv_results_['mean_test_score'])
```

Out[330]:

[<matplotlib.lines.Line2D at 0x7fbd31511f10>]



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

In [289]:

```
regr_models_grid = {'Tree':dt_gs.best_estimator_, 'RF':
                                rf_gs.best_estimator_
                                }
```

In [290]:

```
for model_name, model in regr_models_grid.items():
    regr_train_model(model_name, model, regrMetricLogger)
```

```
**********************************************************
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=Non e,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=
None,
                      min_samples_leaf=14, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='depreca
ted',
                      random_state=None, splitter='best')

MAE=16.303, MedAE=13.375, R2=0.076
**********************************************************
**********************************************************
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='ms e',
                      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=61, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=Fals
e)

MAE=12.462, MedAE=8.665, R2=0.367
**********************************************************
```

Удалось немного улучшить модель дерева решений и достаточно неплохо улучшить модель случайный лес

Ансамблевый метод

In [333]:

```
#  # # Возьмем лучшую модель: 'TREE+RF=>LR'
#  # # модели первого уровня
model_tree = Regressor(dataset=dataset,
                                    estimator=DecisionTreeRegressor,
                                    parameters={'min_impurity_split':1.5,
                                                'max_depth':5},name='tree')
model_lr = Regressor(dataset=dataset,
                                estimator=LinearRegression,
                                name='lr')
model_rf = Regressor(dataset=dataset,
                                estimator=RandomForestRegressor,
                                parameters={'n_estimators': 60,
                                            'max_depth': 40},name='rf')

#  Первый уровень - две модели: дерево и случайный лес
#  Второй уровень: линейная регрессия
pipeline = ModelsPipeline(model_tree, model_rf)
stack_ds = pipeline.stack(k=10, seed=1)
#  модель второго уровня
stacker = Regressor(dataset=stack_ds, estimator=LinearRegression)
```

In [334]:

```
results = stacker.validate(k=10,scorer=mean_absolute_error)
print()

results = stacker.validate(k=10,scorer=median_absolute_error)
```

Metric: mean_absolute_error
Folds accuracy: [12.80562367994027, 13.255704475940108, 13.052168995 181646, 13.014414170510065, 12.848374721738873, 13.016161548132631, 12.75272305242177, 13.073417650626174, 12.619076791682607, 13.289157 092900757]
Mean accuracy: 12.97268221790749
Standard Deviation: 0.2042389455719696
Variance: 0.04171354688834995

Metric: median_absolute_error
Folds accuracy: [8.914441360926233, 9.787841717837992, 9.14755896948 7442, 9.635733648259986, 8.944208849177393, 9.359388721509454, 9.130 147663254334, 9.756749380193888, 9.731591020643553, 9.7452828035110 2]
Mean accuracy: 9.41529441348013
Standard Deviation: 0.3379059186286033
Variance: 0.11418040984424027

Удалось неплохо улучшить модель.

11. Формирование выводов о качестве построенных моделей на основе выбранных метрик.

In [218]:

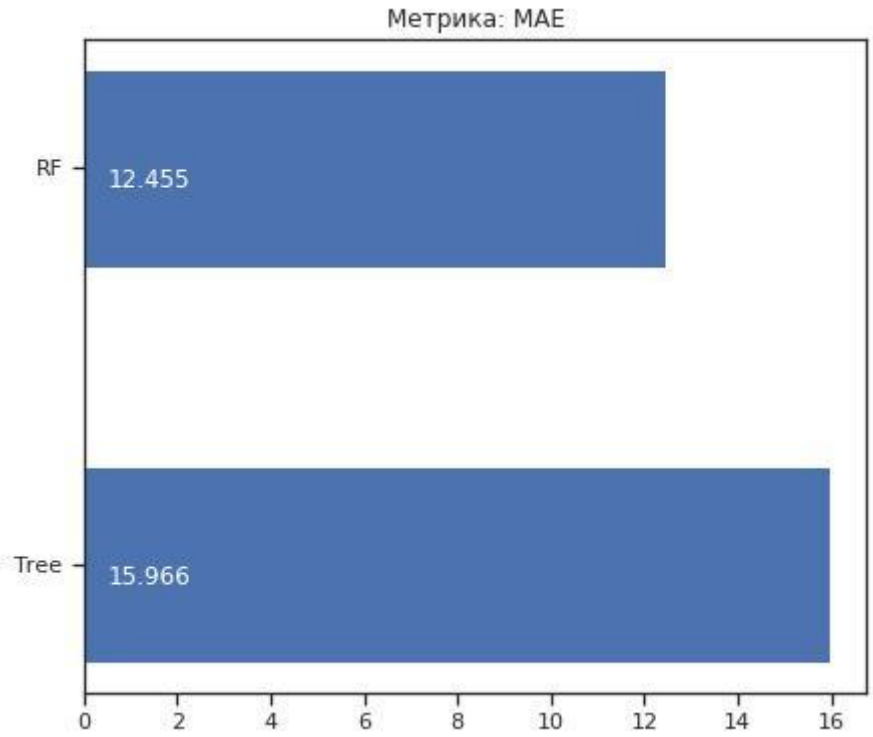
```
#  Метрики качества модели
regr_metrics = regrMetricLogger.df['metric'].unique()
regr_metrics
```

Out[218]:

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

In [219]:

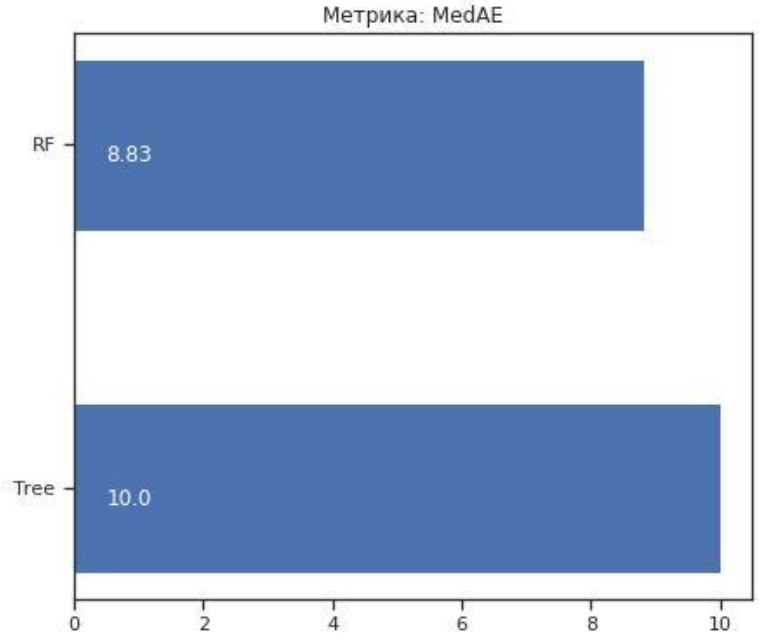
```
regrMetricLogger.plot('Метрика: ' + 'MAE', 'MAE', ascending=False, figsize=(7,
                                                                            6))
```



Ансамбль - 12.973

In [220]:

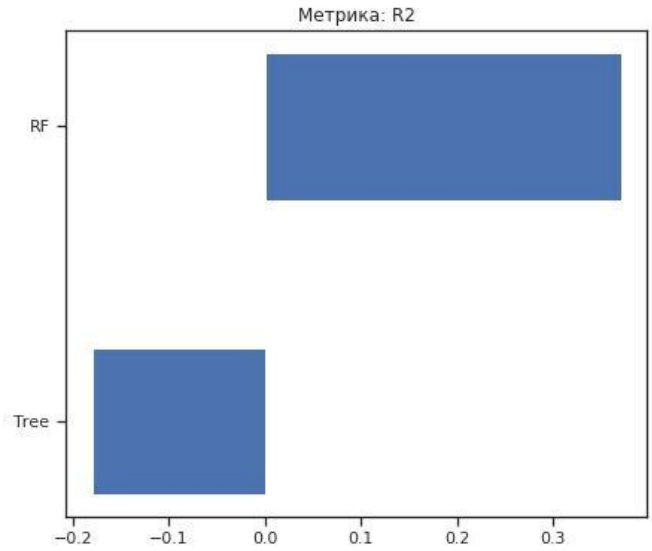
```
regrMetricLogger.plot('Метрика: ' + 'MedAE', 'MedAE', ascending=False,
                                        figsize=(7, 6))
```



Ансамбль - 9.415

In [221]:

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



Вывод:

Лучше всего показала себя модель случайный лес, на втором месте - ансамблевая модель, на третьем - дерево решений. Однако в другой задаче в лабораторной работе лучше показала себя ансамблевая модель, так что в дальнейшем можно использовать обе эти модели и проверять, какая будет работать лучше для конкретной задачи.