|  |  |
| --- | --- |
|  | **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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОМУ ПРОЕКТУ***

***НА ТЕМУ:***

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

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

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

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

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

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

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

Руководитель курсового проекта **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2020 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

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

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсового проекта**

по дисциплине \_\_\_\_\_\_Технологии машинного обучения\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_\_ИУ5-62\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Морозенков Олег Николаевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

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

График выполнения проекта: 25% к \_4\_ нед., 50% к \_8\_ нед., 75% к 12 нед., 100% к 16 нед.

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

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

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

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

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

***Оформление курсового проекта:***

Расчетно-пояснительная записка на \_35\_\_ листах формата А4.

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

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

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

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

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

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

**Руководитель курсового проекта**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

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

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ**](#введение) **5**

[**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](#первый)**6**

[**2.** Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных](#второй)**7**

[**3.** Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей](#третий)**11**

[**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](#четвертый)**16**

[**5.** Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор](#пятый)**19**

[**6.** Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми](#шестой)**21**

[**7.** Формирование обучающей и тестовой выборок на основе исходного набора данных](#седьмой)**21**

[**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](#восьмой)**21**

[**9.** Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы](#девятый) **28**

[**10.** Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей](#десятый)**30**

[**11.** Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.](#одиннадцатый)**31**

[**ЗАКЛЮЧЕНИЕ**](#Заключение) **34**

[**ЛИТЕРАТУРА**](#Литература) **35**

**ВВЕДЕНИЕ**

В данном курсовом проекте предстоит выполнить типовую задачу машинного обучения - провести анализ данных, провести операции с датасетом, подобрать модели, а также подобрать наиболее подходящие гиперпараметры выбранных моделей.

Попробуем различные модели и подберем наилучшую из них на основе выбранных метрик. Также построим вспомогательные графики, которые помогут нам визуально взглянуть на все необходимые показатели.

# ЗАДАНИЕ

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 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. 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 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. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

## 1) Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.

В качестве набора данных мы будем использовать набор данных [https://archive.ics.uci.edu/ml/datasets/Air+quality#](https://archive.ics.uci.edu/ml/datasets/Air+quality)

Содержит ответы газового мультисенсорного устройства, развернутого на месторождении в итальянском городе. Среднечасовые значения откликов регистрируются вместе с эталонами концентрации газа на сертифицированном анализаторе.

Датасет состоит из 1 файла:

* AirQualityUCI.csv - датасет

Файл содержит следующие колонки:

1. Дата (ДД / ММ/ гггг).
2. Время (чч. мм. СС).
3. Истинная часовая усредненная концентрация CO в мг / м^3 (эталонный анализатор).
4. ПТ08.S1 (оксид олова) часовой усредненный отклик датчика (номинально Co-таргетированный).
5. Истинная почасовая усредненная общая концентрация Неметановых углеводородов в мкг / м^3 (эталонный анализатор).
6. Истинная почасовая усредненная концентрация бензола в мкг / м^3 (эталонный анализатор).
7. ПТ08.S2 (titania) часовой усредненный отклик датчика (номинально nmhc-таргетинг).
8. Истинная почасовая усредненная концентрация NOx в ppb (эталонный анализатор).
9. ПТ08.S3 (оксид вольфрама) ежечасно усредненный отклик датчика (номинально направленный на NOx).
10. Истинная часовая усредненная концентрация NO2 в микрог / М^3 (эталонный анализатор).
11. ПТ08.S4 (оксид вольфрама) часовой усредненный отклик датчика (номинально No2)).
12. ПТ08.S5 (оксид индия) часовой усредненный отклик датчика (номинально направленный на O3).
13. Температура в °C.
14. Относительная Влажность Воздуха (%).
15. Абсолютная влажность воздуха.

В рассматриваемом примере будем решать обе задачи - и задачу классификации, и задачу регрессии:

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

### Импорт библиотек

Импортируем библиотеки с помощью команды import. Как правило, все команды import размещают в первых ячейках ноутбука.

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  
import datetime  
import time  
%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

Загрузим файлы датасета в помощью библиотеки Pandas.

Не смотря на то, что файлы имеют расширение txt они представляют собой данные в формате CSV (<https://ru.wikipedia.org/wiki/CSV>). Часто в файлах такого формата в качестве разделителей используются символы ",", ";" или табуляция. Поэтому вызывая метод read\_csv всегда стоит явно указывать разделитель данных с помощью параметра sep. Чтобы узнать какой разделитель используется в файле его рекомендуется предварительно посмотреть в любом текстовом редакторе.

data = pd.read\_csv('AirQualityUCI.csv', sep=";")

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

#### Основные характеристики датасетов

# Первые 5 строк датасета  
data.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Date | Time | CO(GT) | PT08.S1(CO) | NMHC(GT) | C6H6(GT) | PT08.S2(NMHC) | NOx(GT) | PT08.S3(NOx) | NO2(GT) | PT08.S4(NO2) | PT08.S5(O3) | T | RH | AH | Unnamed: 15 | Unnamed: 16 |
| 0 | 10/03/2004 | 18.00.00 | 2,6 | 1360.0 | 150.0 | 11,9 | 1046.0 | 166.0 | 1056.0 | 113.0 | 1692.0 | 1268.0 | 13,6 | 48,9 | 0,7578 | NaN | NaN |
| 1 | 10/03/2004 | 19.00.00 | 2 | 1292.0 | 112.0 | 9,4 | 955.0 | 103.0 | 1174.0 | 92.0 | 1559.0 | 972.0 | 13,3 | 47,7 | 0,7255 | NaN | NaN |
| 2 | 10/03/2004 | 20.00.00 | 2,2 | 1402.0 | 88.0 | 9,0 | 939.0 | 131.0 | 1140.0 | 114.0 | 1555.0 | 1074.0 | 11,9 | 54,0 | 0,7502 | NaN | NaN |
| 3 | 10/03/2004 | 21.00.00 | 2,2 | 1376.0 | 80.0 | 9,2 | 948.0 | 172.0 | 1092.0 | 122.0 | 1584.0 | 1203.0 | 11,0 | 60,0 | 0,7867 | NaN | NaN |
| 4 | 10/03/2004 | 22.00.00 | 1,6 | 1272.0 | 51.0 | 6,5 | 836.0 | 131.0 | 1205.0 | 116.0 | 1490.0 | 1110.0 | 11,2 | 59,6 | 0,7888 | NaN | NaN |

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

Date 114  
Time 114  
CO(GT) 114  
PT08.S1(CO) 114  
NMHC(GT) 114  
C6H6(GT) 114  
PT08.S2(NMHC) 114  
NOx(GT) 114  
PT08.S3(NOx) 114  
NO2(GT) 114  
PT08.S4(NO2) 114  
PT08.S5(O3) 114  
T 114  
RH 114  
AH 114  
Unnamed: 15 9471  
Unnamed: 16 9471  
dtype: int64

# Обработаем данные  
data = data.drop(columns=['Unnamed: 15', 'Unnamed: 16'])  
data = data.dropna(how='any', axis=0)   
data = data.drop(data[(data == -200).sum(axis=1) > 0].index)  
data['Date'] = data['Date'].apply(lambda it: datetime.datetime.strptime(it, '%d/%m/%Y'))  
data['Time'] = data['Time'].apply(lambda it: datetime.datetime.strptime(it, '%H.%M.%S'))  
data['CO(GT)'] = data['CO(GT)'].apply(lambda it: float(it.replace(',', '.')))  
data['C6H6(GT)'] = data['C6H6(GT)'].apply(lambda it: float(it.replace(',', '.')))  
data['T'] = data['T'].apply(lambda it: float(it.replace(',', '.')))  
data['RH'] = data['RH'].apply(lambda it: float(it.replace(',', '.')))  
data['AH'] = data['AH'].apply(lambda it: float(it.replace(',', '.')))

data.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Date | Time | CO(GT) | PT08.S1(CO) | NMHC(GT) | C6H6(GT) | PT08.S2(NMHC) | NOx(GT) | PT08.S3(NOx) | NO2(GT) | PT08.S4(NO2) | PT08.S5(O3) | T | RH | AH |
| 0 | 2004-03-10 | 1900-01-01 18:00:00 | 2.6 | 1360.0 | 150.0 | 11.9 | 1046.0 | 166.0 | 1056.0 | 113.0 | 1692.0 | 1268.0 | 13.6 | 48.9 | 0.7578 |
| 1 | 2004-03-10 | 1900-01-01 19:00:00 | 2.0 | 1292.0 | 112.0 | 9.4 | 955.0 | 103.0 | 1174.0 | 92.0 | 1559.0 | 972.0 | 13.3 | 47.7 | 0.7255 |
| 2 | 2004-03-10 | 1900-01-01 20:00:00 | 2.2 | 1402.0 | 88.0 | 9.0 | 939.0 | 131.0 | 1140.0 | 114.0 | 1555.0 | 1074.0 | 11.9 | 54.0 | 0.7502 |
| 3 | 2004-03-10 | 1900-01-01 21:00:00 | 2.2 | 1376.0 | 80.0 | 9.2 | 948.0 | 172.0 | 1092.0 | 122.0 | 1584.0 | 1203.0 | 11.0 | 60.0 | 0.7867 |
| 4 | 2004-03-10 | 1900-01-01 22:00:00 | 1.6 | 1272.0 | 51.0 | 6.5 | 836.0 | 131.0 | 1205.0 | 116.0 | 1490.0 | 1110.0 | 11.2 | 59.6 | 0.7888 |

# Размер обучающего датасета  
data.shape

(851, 15)

# Список колонок  
data.columns

Index(['Date', 'Time', 'CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(GT)',  
 'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO2)',  
 'PT08.S5(O3)', 'T', 'RH', 'AH'],  
 dtype='object')

# Список колонок с типами данных   
data.dtypes

Date datetime64[ns]  
Time datetime64[ns]  
CO(GT) float64  
PT08.S1(CO) float64  
NMHC(GT) float64  
C6H6(GT) float64  
PT08.S2(NMHC) float64  
NOx(GT) float64  
PT08.S3(NOx) float64  
NO2(GT) float64  
PT08.S4(NO2) float64  
PT08.S5(O3) float64  
T float64  
RH float64  
AH float64  
dtype: object

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

Date 0  
Time 0  
CO(GT) 0  
PT08.S1(CO) 0  
NMHC(GT) 0  
C6H6(GT) 0  
PT08.S2(NMHC) 0  
NOx(GT) 0  
PT08.S3(NOx) 0  
NO2(GT) 0  
PT08.S4(NO2) 0  
PT08.S5(O3) 0  
T 0  
RH 0  
AH 0  
dtype: int64

#### Вывод. Представленный набор данных содержит пропуски. Удаляем строки с пропусками.

Построим некоторые графики для понимания структуры данных.

# Парные диаграммы  
sns.pairplot(data)

<seaborn.axisgrid.PairGrid at 0x110d79610>

num\_cols = ['CO(GT)', 'PT08.S1(CO)', 'NMHC(GT)', 'C6H6(GT)', 'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO2)', 'PT08.S5(O3)', 'T', 'RH', 'AH']

# Скрипичные диаграммы для числовых колонок  
for col in num\_cols:  
 sns.violinplot(x=data[col])  
 plt.show()

/usr/local/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.  
 return np.add.reduce(sorted[indexer] \* weights, axis=axis) / sumval

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

data.dtypes

Date datetime64[ns]  
Time datetime64[ns]  
CO(GT) float64  
PT08.S1(CO) float64  
NMHC(GT) float64  
C6H6(GT) float64  
PT08.S2(NMHC) float64  
NOx(GT) float64  
PT08.S3(NOx) float64  
NO2(GT) float64  
PT08.S4(NO2) float64  
PT08.S5(O3) float64  
T float64  
RH float64  
AH float64  
dtype: object

Для построения моделей будем использовать все признаки кроме признаков date time, потому что мы не рассматриваем наши данные как временной ряд.

Категориальные признаки отсутствуют, их кодирования не требуется.

Вспомогательные признаки для улучшения качества моделей в данном примере мы строить не будем.

data.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Date | Time | CO(GT) | PT08.S1(CO) | NMHC(GT) | C6H6(GT) | PT08.S2(NMHC) | NOx(GT) | PT08.S3(NOx) | NO2(GT) | PT08.S4(NO2) | PT08.S5(O3) | T | RH | AH |
| 0 | 2004-03-10 | 1900-01-01 18:00:00 | 2.6 | 1360.0 | 150.0 | 11.9 | 1046.0 | 166.0 | 1056.0 | 113.0 | 1692.0 | 1268.0 | 13.6 | 48.9 | 0.7578 |
| 1 | 2004-03-10 | 1900-01-01 19:00:00 | 2.0 | 1292.0 | 112.0 | 9.4 | 955.0 | 103.0 | 1174.0 | 92.0 | 1559.0 | 972.0 | 13.3 | 47.7 | 0.7255 |
| 2 | 2004-03-10 | 1900-01-01 20:00:00 | 2.2 | 1402.0 | 88.0 | 9.0 | 939.0 | 131.0 | 1140.0 | 114.0 | 1555.0 | 1074.0 | 11.9 | 54.0 | 0.7502 |
| 3 | 2004-03-10 | 1900-01-01 21:00:00 | 2.2 | 1376.0 | 80.0 | 9.2 | 948.0 | 172.0 | 1092.0 | 122.0 | 1584.0 | 1203.0 | 11.0 | 60.0 | 0.7867 |
| 4 | 2004-03-10 | 1900-01-01 22:00:00 | 1.6 | 1272.0 | 51.0 | 6.5 | 836.0 | 131.0 | 1205.0 | 116.0 | 1490.0 | 1110.0 | 11.2 | 59.6 | 0.7888 |

# Числовые колонки для масштабирования  
scale\_cols = num\_cols

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

# Добавим масштабированные данные в набор данных  
for i in range(len(scale\_cols)):  
 col = scale\_cols[i]  
 new\_col\_name = col + '\_scaled'  
 data[new\_col\_name] = sc1\_data[:,i]

data.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Date | Time | CO(GT) | PT08.S1(CO) | NMHC(GT) | C6H6(GT) | PT08.S2(NMHC) | NOx(GT) | PT08.S3(NOx) | NO2(GT) | ... | C6H6(GT)\_scaled | PT08.S2(NMHC)\_scaled | NOx(GT)\_scaled | PT08.S3(NOx)\_scaled | NO2(GT)\_scaled | PT08.S4(NO2)\_scaled | PT08.S5(O3)\_scaled | T\_scaled | RH\_scaled | AH\_scaled |
| 0 | 2004-03-10 | 1900-01-01 18:00:00 | 2.6 | 1360.0 | 150.0 | 11.9 | 1046.0 | 166.0 | 1056.0 | 113.0 | ... | 0.294574 | 0.457887 | 0.333333 | 0.403664 | 0.536313 | 0.427494 | 0.479485 | 0.313808 | 0.497804 | 0.328285 |
| 1 | 2004-03-10 | 1900-01-01 19:00:00 | 2.0 | 1292.0 | 112.0 | 9.4 | 955.0 | 103.0 | 1174.0 | 92.0 | ... | 0.229974 | 0.388208 | 0.198718 | 0.483718 | 0.418994 | 0.350348 | 0.338263 | 0.301255 | 0.480234 | 0.298458 |
| 2 | 2004-03-10 | 1900-01-01 20:00:00 | 2.2 | 1402.0 | 88.0 | 9.0 | 939.0 | 131.0 | 1140.0 | 114.0 | ... | 0.219638 | 0.375957 | 0.258547 | 0.460651 | 0.541899 | 0.348028 | 0.386927 | 0.242678 | 0.572474 | 0.321267 |
| 3 | 2004-03-10 | 1900-01-01 21:00:00 | 2.2 | 1376.0 | 80.0 | 9.2 | 948.0 | 172.0 | 1092.0 | 122.0 | ... | 0.224806 | 0.382848 | 0.346154 | 0.428087 | 0.586592 | 0.364849 | 0.448473 | 0.205021 | 0.660322 | 0.354973 |
| 4 | 2004-03-10 | 1900-01-01 22:00:00 | 1.6 | 1272.0 | 51.0 | 6.5 | 836.0 | 131.0 | 1205.0 | 116.0 | ... | 0.155039 | 0.297090 | 0.258547 | 0.504749 | 0.553073 | 0.310325 | 0.404103 | 0.213389 | 0.654466 | 0.356912 |

5 rows × 28 columns

# Проверим, что масштабирование не повлияло на распределение данных  
for col in scale\_cols:  
 col\_scaled = col + '\_scaled'  
  
 fig, ax = plt.subplots(1, 2, figsize=(8,3))  
 ax[0].hist(data[col], 50)  
 ax[1].hist(data[col\_scaled], 50)  
 ax[0].title.set\_text(col)  
 ax[1].title.set\_text(col\_scaled)  
 plt.show()

![Изображение выглядит как снимок экрана

Автоматически созданное описание](data:image/png;base64;base64,)

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

Целевой признак регрессии - поле AH.

# Воспользуемся наличием тестовых выборок,   
# включив их в корреляционную матрицу  
corr\_cols\_1 = scale\_cols  
corr\_cols\_1

['CO(GT)',  
 'PT08.S1(CO)',  
 'NMHC(GT)',  
 'C6H6(GT)',  
 'PT08.S2(NMHC)',  
 'NOx(GT)',  
 'PT08.S3(NOx)',  
 'NO2(GT)',  
 'PT08.S4(NO2)',  
 'PT08.S5(O3)',  
 'T',  
 'RH',  
 'AH']

scale\_cols\_postfix = [x+'\_scaled' for x in scale\_cols]  
corr\_cols\_2 = scale\_cols\_postfix  
corr\_cols\_2

['CO(GT)\_scaled',  
 'PT08.S1(CO)\_scaled',  
 'NMHC(GT)\_scaled',  
 'C6H6(GT)\_scaled',  
 'PT08.S2(NMHC)\_scaled',  
 'NOx(GT)\_scaled',  
 'PT08.S3(NOx)\_scaled',  
 'NO2(GT)\_scaled',  
 'PT08.S4(NO2)\_scaled',  
 'PT08.S5(O3)\_scaled',  
 'T\_scaled',  
 'RH\_scaled',  
 'AH\_scaled']

fig, ax = plt.subplots(figsize=(10,5))  
sns.heatmap(data[corr\_cols\_1].corr(), annot=True, fmt='.2f')

<matplotlib.axes.\_subplots.AxesSubplot at 0x11a19c510>

fig, ax = plt.subplots(figsize=(10,5))  
sns.heatmap(data[corr\_cols\_2].corr(), annot=True, fmt='.2f')

<matplotlib.axes.\_subplots.AxesSubplot at 0x11a311f90>

На основе корреляционной матрицы можно сделать следующие выводы:

* Корреляционные матрицы для исходных и масштабированных данных совпадают.
* Целевой признак регрессии "AH" наиболее сильно коррелирует с PT08.S1(CO), PT08.S3(NOx), PT08.S4(NO2), PT08.S5(O3), RH. Эти признаки обязательно следует оставить в модели классификации.
* 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.

## 5) Выбор метрик для последующей оценки качества моделей.

### 5.1) В качестве метрик для решения задачи классификации будем использовать:

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.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score)

#### 2. Метрика recall (полнота):

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

Используется функция [recall\_score.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score)

#### 3. Метрика -мера

Для того, чтобы объединить precision и recall в единую метрику используется -мера, которая вычисляется как среднее гармоническое от precision и recall:

где определяет вес точности в метрике.

На практике чаще всего используют вариант F1-меры (которую часто называют F-мерой) при :

Для вычисления используется функция [f1\_score.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score)

#### 4. Метрика ROC AUC

Основана на вычислении следующих характеристик:

- True Positive Rate, откладывается по оси ординат. Совпадает с recall.

- False Positive Rate, откладывается по оси абсцисс. Показывает какую долю из объектов отрицательного класса алгоритм предсказал неверно.

Идеальная ROC-кривая проходит через точки (0,0)-(0,1)-(1,1), то есть через верхний левый угол графика.

Чем сильнее отклоняется кривая от верхнего левого угла графика, тем хуже качество классификации.

В качестве количественной метрики используется площадь под кривой - ROC AUC (Area Under the Receiver Operating Characteristic Curve). Чем ниже проходит кривая тем меньше ее площадь и тем хуже качество классификатора.

Для получения ROC AUC используется функция [roc\_auc\_score.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#sklearn.metrics.roc_auc_score)

### 5.2) В качестве метрик для решения задачи регрессии будем использовать:

#### 1. [Mean absolute error](https://en.wikipedia.org/wiki/Mean_absolute_error) - средняя абсолютная ошибка

$MAE(y,\hat{y}) = \frac{1}{N} \cdot \sum\limits\_{i=1}^N \lvert y\_i - \hat{y\_i} \rvert $

где:

* - истинное значение целевого признака
* - предсказанное значение целевого признака
* - размер тестовой выборки

Чем ближе значение к нулю, тем лучше качество регрессии.

Основная проблема метрики состоит в том, что она не нормирована.

Вычисляется с помощью функции [mean\_absolute\_error.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html#sklearn.metrics.mean_absolute_error)

#### 2. [Mean squared error](https://en.wikipedia.org/wiki/Mean_squared_error) - средняя квадратичная ошибка

$MSE(y,\hat{y}) = \frac{1}{N} \cdot \sum\limits\_{i=1}^N ( y\_i - \hat{y\_i} )^2 $

где:

* - истинное значение целевого признака
* - предсказанное значение целевого признака
* - размер тестовой выборки

Вычисляется с помощью функции [mean\_squared\_error.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html#sklearn.metrics.mean_squared_error)

#### 3. [Метрика или коэффициент детерминации](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%8D%D1%84%D1%84%D0%B8%D1%86%D0%B8%D0%B5%D0%BD%D1%82_%D0%B4%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B0%D1%86%D0%B8%D0%B8)

$R^2(y,\hat{y}) = 1 - \frac{\sum\limits\_{i=1}^N ( y\_i - \hat{y\_i} )^2}{\sum\limits\_{i=1}^N ( y\_i - \overline{y\_i} )^2} $

где:

* - истинное значение целевого признака
* - предсказанное значение целевого признака
* - размер тестовой выборки
* $\overline{y\_i} = \frac{1}{N} \cdot \sum\limits\_{i=1}^N y\_i $

Вычисляется с помощью функции [r2\_score.](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score)

#### 5.3) Сохранение и визуализация метрик

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

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.

Для задачи регрессии будем использовать следующие модели:

* Линейная регрессия
* Метод ближайших соседей
* Машина опорных векторов
* Решающее дерево
* Случайный лес
* Градиентный бустинг

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

# На основе масштабированных данных выделим   
# обучающую и тестовую выборки с помощью фильтра  
train\_data\_all, test\_data\_all = train\_test\_split(data, test\_size=0.25, random\_state=1)  
train\_data\_all.shape, test\_data\_all.shape

((638, 28), (213, 28))

# Признаки для задачи регресии  
task\_regr\_cols = ['PT08.S1(CO)\_scaled', 'PT08.S3(NOx)\_scaled', 'PT08.S4(NO2)\_scaled', 'PT08.S5(O3)\_scaled', 'RH\_scaled']

# 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['AH']  
regr\_Y\_test = test\_data\_all['AH']  
regr\_X\_train.shape, regr\_X\_test.shape, regr\_Y\_train.shape, regr\_Y\_test.shape

((638, 5), (213, 5), (638,), (213,))

## 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.

### 8.1) Решение задачи регрессии

# Модели  
regr\_models = {'LR': LinearRegression(),   
 'KNN\_5':KNeighborsRegressor(n\_neighbors=5),  
 'SVR':SVR(),  
 'Tree':DecisionTreeRegressor(),  
 'RF':RandomForestRegressor(),  
 '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)

/usr/local/lib/python3.7/site-packages/sklearn/linear\_model/\_base.py:547: RuntimeWarning: internal gelsd driver lwork query error, required iwork dimension not returned. This is likely the result of LAPACK bug 0038, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling back to 'gelss' driver.  
 linalg.lstsq(X, y)

## 9) Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

### 9.1) Пример для задачи регрессии

n\_range = np.array(range(1,500,50))  
tuned\_parameters = [{'n\_neighbors': n\_range}]  
tuned\_parameters

[{'n\_neighbors': array([ 1, 51, 101, 151, 201, 251, 301, 351, 401, 451])}]

%%time  
regr\_gs = GridSearchCV(KNeighborsRegressor(), tuned\_parameters, cv=5, scoring='neg\_mean\_squared\_error')  
regr\_gs.fit(regr\_X\_train, regr\_Y\_train)

CPU times: user 425 ms, sys: 21.3 ms, total: 446 ms  
Wall time: 312 ms

# Лучшая модель  
regr\_gs.best\_estimator\_

KNeighborsRegressor(n\_neighbors=51)

# Лучшее значение параметров  
regr\_gs.best\_params\_

{'n\_neighbors': 51}

# 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 0x11aabc650>]

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

### 10.1) Решение задачи регрессии

regr\_models\_grid = {'KNN\_801':regr\_gs.best\_estimator\_}

for model\_name, model in regr\_models\_grid.items():  
 regr\_train\_model(model\_name, model, regrMetricLogger)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
KNeighborsRegressor(n\_neighbors=51)  
  
MAE=0.108, MSE=0.021, R2=0.448  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

### 11.1) 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))

![](data:image/png;base64;base64,)

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

![](data:image/png;base64;base64,)

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

![Изображение выглядит как снимок экрана

Автоматически созданное описание](data:image/png;base64;base64,)

### Вывод: лучшей оказалась модель на основе метода опорных векторов.

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте мы выполнили типовую задачу машинного обучения. Провели анализ данных, провели некоторые операции с датасетом, подобрали модели, а также подобрали наиболее подходящие гиперпараметры.

В нашем случае регрессия на основе метода опорных векторов показала лучшие результаты.

В данном проекте были закреплены все знания, полученные в данном курсе.

**ЛИТЕРАТУРА**

1. Рукописные лекции за 2020 год по дисциплине «Технологии машинного обучения»
2. <https://scikit-learn.org/stable/index.html>
3. <https://www.kaggle.com/datasets>
4. <http://www.machinelearning.ru/>