**Федеральное государственное бюджетное образовательное учреждение   
высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Отчет по курсовой работе

По дисциплине «Технологии машинного обучения.»

Выполнил:

студент группы ИУ5-61

Белых А.А.

Проверил:

Гапанюк Ю. Е.

**г. Москва, 2020**

Типовое исследование - решение задачи машинного обучения на основе материалов дисциплины. Выполняется студентом единолично.

# **Введение**

Машинное обучение изучает автоматические методы обучения, позволяющие делать точные прогнозы на основе прошлых наблюдений. Например, предположим, что мы хотели бы создать фильтр электронной почты, который может отличать спам (нежелательную почту) от нежелательной почты. Подход машинного обучения к этой проблеме будет следующим: начните со сбора как можно большего количества примеров как спамовых, так и неспамовых писем. Затем загрузите эти примеры вместе с метками, указывающими, являются ли они спамом, в свой любимый алгоритм машинного обучения, который автоматически создаст правило классификации или прогнозирования. Учитывая новое электронное письмо без метки, такое правило пытается предсказать, является оно спамом или нет. Цель, конечно же, состоит в том, чтобы сгенерировать правило, позволяющее делать наиболее точные прогнозы на новых тестовых примерах.

Построение высокоточного правила прогнозирования, безусловно, является сложной задачей. С другой стороны, совсем нетрудно придумать очень грубые практические правила, которые будут лишь умеренно точными. Пример такого правила может быть примерно таким: «Если в электронном письме встречается фраза «купи сейчас», значит, это спам». Такое правило даже близко не покрывает все спам-сообщения; например, в нем ничего не говорится о том, что предсказывать, если в сообщении не встречается фраза «купите сейчас». С другой стороны, это правило будет делать прогнозы, которые значительно лучше, чем случайные предположения.

Boosting, метод машинного обучения, основан на наблюдении, что найти множество приблизительных практических правил может быть намного проще, чем найти одно, очень точное правило прогнозирования. Чтобы применить метод повышения, мы начнем с метода или алгоритма для поиска приблизительных практических правил. Алгоритм повышения уровня многократно вызывает этот «слабый» или «базовый» алгоритм обучения, каждый раз вводя в него другое подмножество обучающих примеров (или, если быть более точным, другое распределение или взвешивание по обучающим примерам). Каждый раз, когда он вызывается, базовый алгоритм обучения генерирует новое правило слабого прогнозирования, и после многих раундов алгоритм повышения должен объединить эти слабые правила в одно правило прогнозирования, которое, будем надеяться, будет намного точнее любого из слабых правил.

# **Основная часть**

В данной работе была поставлена задача спрогнозировать количество пострадавших в дорожно-транспортном происшествии, на основе данных с сайта ГИБДД [1].

1. Колонки, которые содержатся в выгруженном датасете, представлены в таблице 1.

***Таблица 1 столбцы и их значения в датасете.***

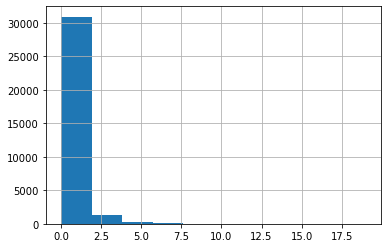
|  |  |
| --- | --- |
| **id** | Уникальный идентификатор названия ребра |
| **geom** | Полигон линии дороги(ребра) |
| **obj\_cod** | Идентификатор ребра |
| **typ\_cod** | Тип ребра  7700 – дорога с физической возможностью движения автомобилей 7701 – дорога без физической возможности движения автомобилей (пешеходная дорога, проход, тропа, пешеходный переход) 7702 – внутриквартальный проезд  7703 – парковочная зона  7704 – проход внутри здания 7705 – зимник  7706 – велодорожка 7707 – канатная дорога  7730 – железная дорога 7740 – трамвайный путь 7750 – метро 7751 – пересадка метро 7760 – монорельс 7770 – автоматически построенный отрезок для связи элементов графа между собой |
| **nlevel** | Категория ребра  1 и 2 – Федеральные трассы (М, Р, А) 3 – основные дороги субъектов РФ и главные улицы населенных пунктов 4 – значимые дороги субъектов РФ и основные улицы населенных пунктов 5 – прочие дороги субъектов РФ 6 – второстепенные дороги субъектов РФ и прочие дороги населенных пунктов 7 – внутриквартальные проезды и подъезды к POI 8 – непроезжие проселочные дороги |
| **btf** | Код конструктивного элемента ребра  0 – нет 1 – мост 2 – понтонный мост 3 – туннель 4 – паромная переправа 5 – жд переправа 6 – надземный пешеходный переход 7 – наземный пешеходный переход 8 – подземный пешеходный переход 9 – лестница 10 – эскалатор 11 – траволатор |
| **oneway** | Признак одностороннего движения  0 – нет 1 – да 2 – обратное направление (только для общественного транспорта) |
| **surface** | Тип поверхности ребра  0 – нет данных 1 – нет 2 – асфальт 3 – рельсы |
| **splitter** | Разделитель прямого и обратного направления ребра  0 – нет 1 – логический 2 – физический |
| **speedlim** | Разрешённая скорость движения по ребру, км/ч  Минимальное значение: 0 Максимальное значение: 130 |
| **f\_rspeed** | Условная расчетная среднесуточная скорость движения по ребру в прямом направлении, км/ч  Минимальное значение: 0 Максимальное значение: 130 |
| **t\_rspeed** | Условная расчетная среднесуточная скорость движения по ребру в обратном направлении, км/ч  Минимальное значение: 0 Максимальное значение: 130 |
| **lanewidth** | Ширина полосы проезжей части, м  Минимальное значение: 0 Максимальное значение: 10 |
| **f\_lanes** | Количество полос в прямом направлении  Минимальное значение: 0 Максимальное значение: 10 |
| **t\_lanes** | Количество полос в обратном направлении  Минимальное значение: 0 Максимальное значение: 10 |
| **f\_sidewalk** | Характеристики тротуара в прямом направлении  0 – нет данных 1 – нет 2 – узкий 3 – средний 4 – широкий 5 – существует возможность прохода 7 – специально оборудованная пешеходная зона 8 – технический тротуар |
| **t\_sidewalk** | Характеристики тротуара в обратном направлении  0 – нет данных 1 – нет 2 – узкий 3 – средний 4 – широкий 5 – существует возможность прохода 7 – специально оборудованная пешеходная зона 8 – технический тротуар |
| **f\_buslanes** | Наличие выделенной полосы для движения общественного транспорта в прямом направлении  0 – нет 1 – да |
| **t\_buslanes** | Наличие выделенной полосы для движения общественного транспорта в обратном направлении  0 – нет 1 – да |
| **multidigit** | Признак дороги, представленной отдельными ребрами для каждого направления движения  0 – нет 1 – да |
| **rd\_name** | Полное название (с типом) улицы/дороги |
| **rd\_e\_name** | Код наименования трассы E |
| **rd\_l\_name** | Название трассы М, А, Р |
| **f\_parking** | Наличие парковки по направлению справа от ребра  0 – нет данных 1 – парковка запрещена 2 – парковка запрещена по четным 3 – парковка запрещена по нечетным 4 – парковка разрешена в крайнем правом ряду 5 – парковка разрешена в кармане 6 – парковка разрешена на тротуаре |
| **t\_parking** | Наличие парковки по направлению слева от ребра.  0 – нет данных 1 – парковка запрещена 2 – парковка запрещена по четным 3 – парковка запрещена по нечетным 4 – парковка разрешена в крайнем правом ряду 5 – парковка разрешена в кармане 6 – парковка разрешена на тротуаре |
| **bicyclanes** | Наличие велодорожки  0 – нет 1 – справа относительно направления геометрии дороги 2 – слева относительно направления геометрии дороги 3 – на самом ребре |

**Прогнозируемая величина содержится в столбце victims.**

2. Для проведения разведочного анализа данных, было сделано следующее:

размер dtp\_train: (32516, 31).

распределение прогнозируемой величины имеет вид:



**Рисунок 1 столбец victims.**

количество 0 составляет 83.5%, а количество значений > 0 составляет 16.5%.

Количество пропущенных значений:

id - 0.0%

victims - 0.0%

geom\_buffer - 0.0%

typ\_cod - 0.0%

nlevel - 0.0%

btf - 0.0%

oneway - 0.0%

surface - 0.0%

splitter - 0.0%

speedlim - 0.0%

f\_rspeed - 0.0%

t\_rspeed - 0.0%

lanewidth - 0.0%

f\_lanes - 0.0%

t\_lanes - 0.0%

f\_sidewalk - 0.0%

t\_sidewalk - 0.0%

f\_buslanes - 0.0%

t\_buslanes - 0.0%

multidigit - 0.0%

rd\_name - 17.0%

rd\_e\_name - 100.0%

rd\_l\_name - 95.0%

f\_parking - 0.0%

t\_parking - 0.0%

bicyclanes - 0.0%

geom\_buffer\_length - 0.0%

EDGEID - 0.0%

TYP\_COD - 0.0%

avg\_check\_speed - 0.0%

cnt\_fix - 0.0%

3. Имя дороги было закодировано с помощью LabelEncoder:

dtp\_train['rd\_name'] = LabelEncoder().fit\_transform(dtp\_train['rd\_name']).

4. Тепловая карта корреляций между столбцами представлена на рисунке 2.

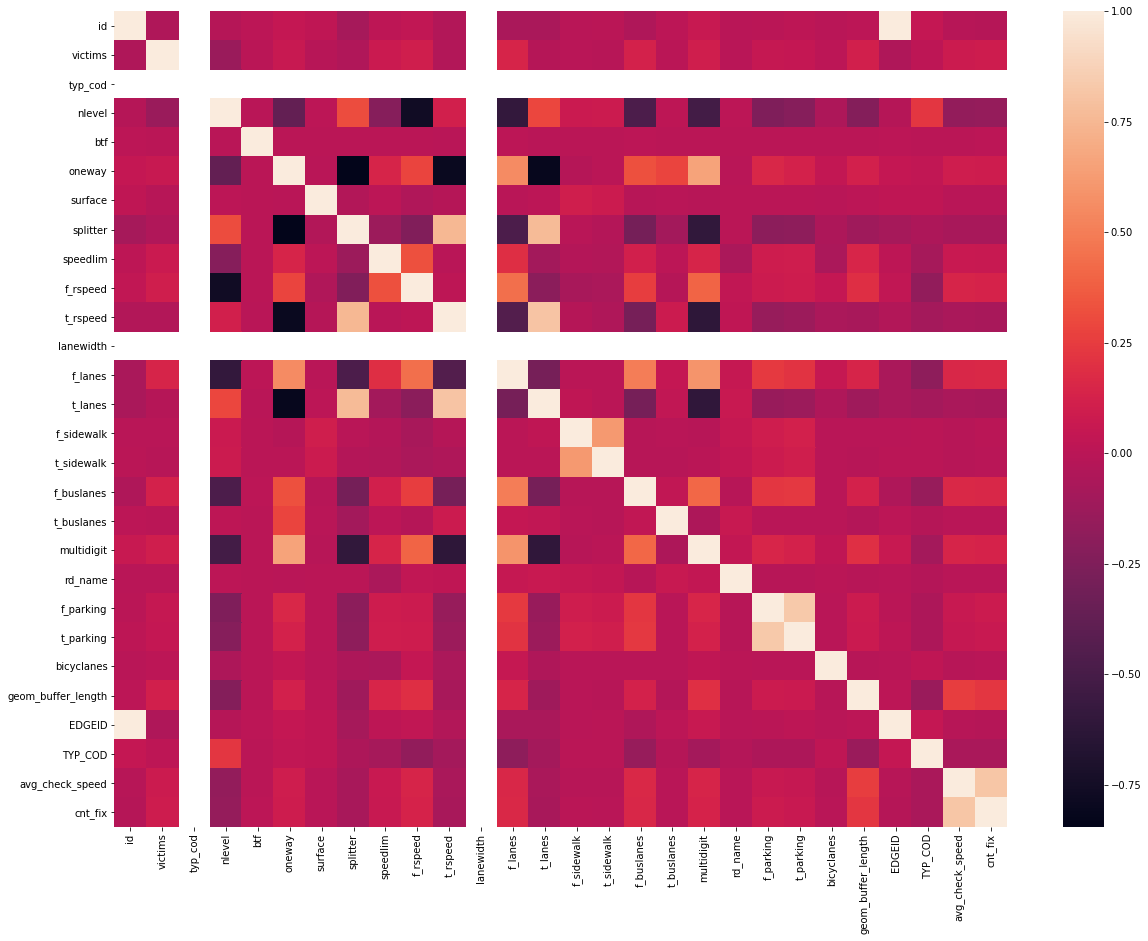


Рисунок 2 тепловая карта для столбцов датасета.

5. Метрики. Выбор метрик связан во-первых с соответствием прогноза тестовому множеству (mean\_absolute\_error, mean\_squared\_error), а также объяснемой долей дисперсии (r2).

6. Модели:

regr\_models = {'LR': LinearRegression(),

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

'SVR':SVR(),

'Tree':DecisionTreeRegressor(),

'RF':RandomForestRegressor(),

'GB':GradientBoostingRegressor()}

7.

* final\_dtp = dtp\_train.drop(['cnt\_fix', 'typ\_cod', 'lanewidth', 'f\_parking', 'f\_sidewalk', 'speedlim'], axis=1)
* X = final\_dtp.drop(['id', 'victims', 'geom\_buffer', 'EDGEID'],axis=1)
* y = final\_dtp['victims']
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42)

8.

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

LinearRegression()

MAE=0.417, MSE=0.489, R2=0.026

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

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

KNeighborsRegressor()

MAE=0.399, MSE=0.569, R2=-0.134

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

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

SVR()

MAE=0.319, MSE=0.525, R2=-0.046

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

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

DecisionTreeRegressor()

MAE=0.42, MSE=0.991, R2=-0.974

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

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

RandomForestRegressor()

MAE=0.406, MSE=0.526, R2=-0.048

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

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

GradientBoostingRegressor()

MAE=0.412, MSE=0.479, R2=0.047

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

9.

Wall time: 2min 25s

GridSearchCV(estimator=GradientBoostingRegressor(random\_state=42),

param\_grid={'max\_depth': [1, 3, 5, 7],

'min\_samples\_leaf': [1, 2, 3],

'n\_estimators': [100, 50]},

scoring='r2')

GradientBoostingRegressor(max\_depth=5, min\_samples\_leaf=3, random\_state=42)

10.

GradientBoostingRegressor(max\_depth=5, min\_samples\_leaf=3, random\_state=42)

MAE=0.408, MSE=0.486, R2=0.033

11. Сравнение результатов для теста и предсказания модели на рисунке 3.

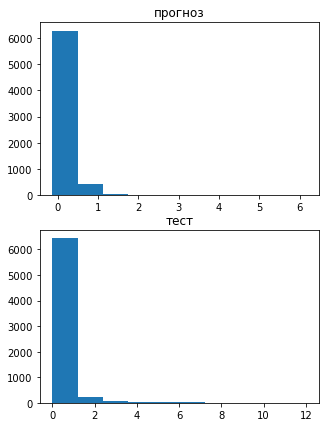


Рисунок 3 Тест и предсказания модели.

Модель предсказывает нули практически в таком же соотношении, однако есть некоторое занижение количества пострадавших, так как модель предсказывает максимальное значение 4, а в тесте максимальное значение 7.

Наиболее важные признаки, которые оказывают влияние на количество пострадавших, представлены на рисунке 4.

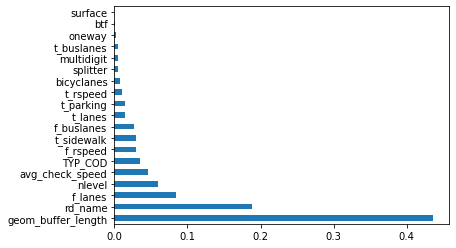


Рисунок 4 Важные столбцы для модели.

# **Заключение**

В данной работе были проанализированы ДТП в Москве за 2019 год. По результатам работы можно сделать вывод, что важным фактором является средняя скорость, кроме того, важным фактором является количество полос, категория самой дороги, а также длина дороги.

Были применены 6 моделей, для каждой из них были рассчитаны 3 метрики. Подбор параметров происходил для модели бустинга.

# Список использованных источников информации:

1. <http://stat.gibdd.ru/>

# **Приложение**

Код программы

#!/usr/bin/env python

# coding: utf-8

# # импорт библиотек

# In[38]:

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

import datetime

from sklearn.preprocessing import MinMaxScaler

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsRegressor

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 DecisionTreeRegressor, export\_graphviz

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import ExtraTreesRegressor

from sklearn.ensemble import GradientBoostingRegressor

# # http://stat.gibdd.ru/ Москва 2019

# In[39]:

dtp\_train = pd.read\_csv('train.csv')

# In[40]:

print('размер dtp\_train: ', dtp\_train.shape)

# In[41]:

dtp\_train.head(2)

# # Будем решать задачу регрессии для прогноза пострадавших (столбец victims)

# In[42]:

dtp\_train['victims'].hist()

# # посчитаем дисбаланс между нулевыми значениями и значениями больше 0

# In[43]:

# посчитаем дисбаланс классов

total = dtp\_train['victims'].value\_counts()

count\_0 = total[0]

count\_more\_0 = sum(total)-total[0]

print('количество 0 составляет {}%, а количество значений > 0 составляет {}%.'

.format(round(count\_0 \* 100 / sum(total),1), round(count\_more\_0 \* 100 / sum(total),1)))

# # Проверим наличие пустых значений и посмотрим тип данных в колонках

# In[44]:

dtp\_train.info()

# # печать информации по пропущенным значениям в таблице

# In[45]:

# печать информации по пропущенным значениям в таблице

def check\_na\_labels(df):

total = {}

for col in df.columns:

pct\_missing = np.mean(df[col].isnull())

total[col] = round(pct\_missing \* 100)

print('{} - {}%'.format(col, round(pct\_missing\*100)))

# In[46]:

check\_na\_labels(dtp\_train)

# # Много пропущенных значений для названий дорог

# In[47]:

dtp\_train = dtp\_train.drop(['rd\_e\_name', 'rd\_l\_name'], axis=1)

# In[48]:

dtp\_train = dtp\_train.dropna()

# # работа с rd\_name

# In[49]:

dtp\_train['rd\_name'] = LabelEncoder().fit\_transform(dtp\_train['rd\_name'])

# # corrmatrix

# In[50]:

fig, ax = plt.subplots(figsize=(20,15))

sns.heatmap(dtp\_train.corr(), fmt='.2f')

# # уберем один из признаков с корреляцией 1, а также те, что коррелируют со всеми остальными.

# In[51]:

final\_dtp = dtp\_train.drop(['cnt\_fix', 'typ\_cod', 'lanewidth', 'f\_parking', 'f\_sidewalk', 'speedlim'], axis=1)

# In[52]:

X = final\_dtp.drop(['id', 'victims', 'geom\_buffer', 'EDGEID'],axis=1)

y = final\_dtp['victims']

# In[53]:

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42)

# # метрики. Выбор метрик связан, во-первых, с соответствием прогноза тестовому множеству (mean\_absolute\_error, mean\_squared\_error), а также объяснимой долей дисперсии (r2)

# In[54]:

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()

# In[55]:

# Модели

regr\_models = {'LR': LinearRegression(),

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

'SVR':SVR(),

'Tree':DecisionTreeRegressor(),

'RF':RandomForestRegressor(),

'GB':GradientBoostingRegressor()}

# In[56]:

# Сохранение метрик

regrMetricLogger = MetricLogger()

# In[79]:

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

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(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('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

# In[80]:

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

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

# In[63]:

get\_ipython().run\_cell\_magic('time', '', "tuned\_parameters = {'max\_depth': [1,3,5,7],\n 'n\_estimators': [100,50],\n 'min\_samples\_leaf': [1,2,3]}\ngs = GridSearchCV(RandomForestRegressor(random\_state=42), tuned\_parameters, scoring='r2')\ngs.fit(X\_train,y\_train)")

# In[64]:

gs.best\_estimator\_

# In[65]:

regr\_models\_grid = {'GB': gs.best\_estimator\_}

# In[81]:

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

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

# # Проверка на тесте и сравнение прогноза и теста

# In[75]:

best\_GB = gs.best\_estimator\_

predict = best\_GB.predict(X\_test)

fig, (ax1, ax2) = plt.subplots(nrows=2, figsize=(5,7))

ax1.set\_title('прогноз')

ax1.hist(predict)

ax2.set\_title('тест')

ax2.hist(y\_test)

# In[84]:

feat\_importances\_best\_GB = pd.Series(best\_GB.feature\_importances\_, index=X.columns)

feat\_importances\_best\_GB.nlargest(20).plot(kind='barh')