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 _____ РТ (Радиотехнический)_____

КАФЕДРА _____ИУ5 (Системы обработки информации и управления)_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Создание веб-приложения для анализа и визуализации данных с использованием методов машинного обучения

Студентка ИУ5-61 _____ **Алехин С.С.**
(Группа) (Подпись, дата) (Фамилия И.О.)

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

Москва, 2021 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине _____

Студентка группы ИУ5-61 Алехин Сергей Сергеевич
(Фамилия, имя, отчество)

Тема курсового проекта Создание веб-приложения для анализа и визуализации данных
_____с использованием методов машинного обучения _____
_____

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
_____учебная_____
Источник тематики (кафедра, предприятие, НИР) _____кафедра_____

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

Задание _____провести предобработку выбранного набора данных, обучить модели машинного обучения, оценить метрики качества для каждой модели, подобрать соответствующие гиперпараметры, выбрать лучшие алгоритмы для данного датасета; обернуть всё в интерактивное веб-приложение _____
_____
_____
_____
_____

Оформление курсовой работы:

Расчетно-пояснительная записка на _____ листах формата А4.

_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсовой работы _____ _____
(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 4
Задание 4
Средства реализации 5
Выполнение 5
Листинг 5
Список источников информации 13

Введение

В рамках курса «Технологии машинного обучения» текущего семестра мы постепенно наращивали свои знания и проходили все стадии разработки продукта, связанного с анализом данных: разведочный анализ данных, предобработку признаков, обучение моделей, применений метрик качества. Были рассмотрены различные алгоритмы машинного обучения, в том числе и ансамблевые модели. Шаг за шагом мы собирали наработки в рамках лабораторных работ, и вот пришло время аккумулировать всё сделанное за семестр в курсовой работе.

Задание

На выбранном наборе данных:

- произвести разведочный анализ данных, визуализировать признаки (все или некоторые) таким образом, чтобы прояснить структуру данных;
- разделить признаки на независимые фичи и целевую переменную;
- провести всю необходимую предобработку данных (масштабирование числовых признаков, заполнение пропущенных значений, кодирование категориальных признаков);
- провести корреляционный анализ (корреляционная матрица и/или тепловая карта), сделать соответствующие выводы о независимых признаках, сильно коррелирующих между собой или с целевой переменной;
- выбрать на основании предыдущих пунктов те признаки, которые войдут в модели;
- разделить набор данных на обучающую и тестовую выборки;
- определиться с моделями машинного обучения и метриками оценки качества, подходящими для конкретной задачи (классификации или регрессии);

- обучить моделей на данных тестовой выборки без подбора гиперпараметров;
- обучить моделей на всех данных с использованием стратегий кросс-валидации с подбором гиперпараметров и выявлением оптимальных значений;
- сравнить значений метрик качества для двух последних пунктов;
- в веб-приложении сделать возможным задавать гиперпараметры для каждого алгоритма и наблюдать за изменением метрик (в том числе в виде графиков);
- сравнить полученные вручную результаты с pipeline'ом библиотеки AutoML;
- сформулировать выводы о качестве обученных моделей.

Средства реализации

Приложение реализовано на языке программирования Python с использованием веб-фреймворка для задач машинного обучения Streamlit, а также библиотек для работы с данными Pandas, Numpy, Scipy, sklearn.

Выполнение

Будем использовать датасет из 15 колонок про отчеты.

Листинг

```
import streamlit as st
import seaborn as sns
import pandas as pd
import numpy as np
import plotly.figure_factory as ff
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler, MinMaxScaler, StandardScaler, Normalizer
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn import tree
import re

def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv('data/report.csv')
    return data


@st.cache
def preprocess_data(data_in):
    '''
    Масштабирование признаков, функция возвращает X и y для кросс-валидации
    '''
    data_out = data_in.copy()
    # Числовые колонки для масштабирования
    scale_cols = ['population', 'homicides','assaults','robberies']
    new_cols = []
    sc1 = MinMaxScaler()
    sc1_data = sc1.fit_transform(data_out[scale_cols])
    for i in range(len(scale_cols)):
        col = scale_cols[i]
        new_col_name = col + '_scaled'
        new_cols.append(new_col_name)
        data_out[new_col_name] = sc1_data[:, i]
    X = data_out[new_cols]
    Y = data_out['violent_crimes'].astype(int)
    # Чтобы в тесте получилось низкое качество используем только 0,5% данных для обучения
    X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size=0.8, test_size=0.2, random_state=1)
    return X_train, X_test, y_train, y_test, X, Y


data = load_data()

st.sidebar.header('Случайный лес')
n_estimators_1 = st.sidebar.slider('Количество фолдов:', min_value=3, max_value=10, value=3, step=1)

st.sidebar.header('Градиентный бустинг')
n_estimators_2 = st.sidebar.slider('Количество:', min_value=3, max_value=10, value=3, step=1)
random_state_2 = st.sidebar.slider('random_state:', min_value=3, max_value=15, value=3, step=1)

st.sidebar.header('Модель ближайших соседей')
n_estimators_3 = st.sidebar.slider('Количество K:', min_value=3, max_value=10, value=3, step=1)

# Первые пять строк датасета
st.subheader('Первые 5 значений')
st.write(data.head())

st.subheader('Размер датасета')
st.write(data.shape)

# Заполняем отсутствующие значения
data['homicides'] = data['homicides'].replace(0,np.nan)
data['homicides'] = data['homicides'].fillna(data['homicides'].mean())
# Заполняем отсутствующие значения
data['population'] = data['population'].replace(0,np.nan)
data['population'] = data['population'].fillna(data['population'].mean())
```

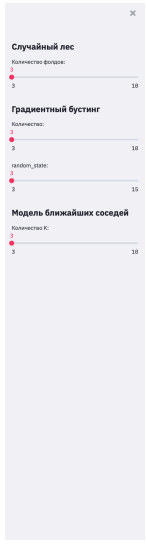
```
data['assaults'] = data['assaults'].replace(0,np.nan)
data['assaults'] = data['assaults'].fillna(data['assaults'].mean())

data['robberies'] = data['robberies'].replace(0,np.nan)
data['robberies'] = data['robberies'].fillna(data['robberies'].mean())

data['violent_crimes'] = data['violent_crimes'].replace(0,np.nan)
data['violent_crimes'] = data['violent_crimes'].fillna(data['violent_crimes'].mean())

st.subheader('Количество нулевых элементов')
st.write(data.isnull().sum())

st.write(data['report_year'].value_counts())

st.subheader('Колонки и их типы данных')
st.write(data.dtypes)

st.subheader('Статистические данные')
st.write(data.describe())

fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x=data['report_year'], y=data['violent_crimes'])
plt.xlabel("report_year")
plt.ylabel("violent_crimes")
st.pyplot(fig)

f1, ax = plt.subplots()
sns.boxplot(x=data['report_year'])
st.pyplot(f1)

st.subheader('Масштабирование данных')
f, ax = plt.subplots()
plt.hist(data['report_year'], 50)
plt.show()
st.pyplot(f)

st.subheader('Показать корреляционную матрицу')
fig1, ax = plt.subplots(figsize=(10, 5))
sns.heatmap(data.corr(), annot=True, fmt='.2f')
st.pyplot(fig1)

X_train, X_test, Y_train, Y_test, X, Y = preprocess_data(data)
forest_1 = RandomForestRegressor(n_estimators=n_estimators_1, oob_score=True, random_state=10)
forest_1.fit(X, Y)
Y_predict = forest_1.predict(X_test)


st.subheader('RandomForestRegressor')
st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_predict))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_predict))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_predict))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_predict))

fig1 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['homicides_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['homicides_scaled'], Y_predict, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('homicides_scaled')
```

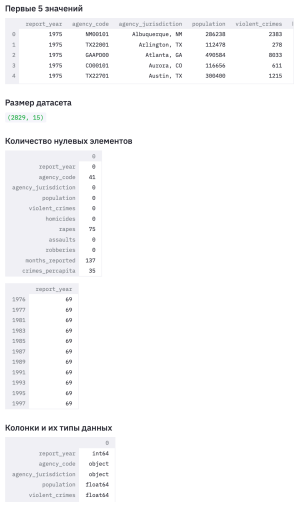
```
plt.ylabel('strength')
plt.plot(n_estimators_1)
st.pyplot(fig1)

st.subheader('Нахождение лучшего случайного леса')

params2 = {
   'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50, 75, 100],
}

grid_2 = GridSearchCV(estimator=RandomForestRegressor(oob_score=True, random_state=10),
                      param_grid=params2,
                      scoring='neg_mean_squared_error',
                      cv=3,
                      n_jobs=-1)
grid_2.fit(X, Y)

st.write(grid_2.best_params_)

forest_3 = RandomForestRegressor(n_estimators=6, oob_score=True, random_state=5)
forest_3.fit(X, Y)
Y_predict3 = forest_3.predict(X_test)
st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_predict3))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_predict3))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_predict3))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_predict3))

fig1 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['homicides_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['homicides_scaled'], Y_predict3, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('homicides_scaled')
plt.ylabel('strength')
plt.plot(n_estimators_1)
st.pyplot(fig1)

st.subheader('Градиентный бустинг')

grad = GradientBoostingRegressor(n_estimators=n_estimators_2, random_state=random_state_2)
grad.fit(X_train, Y_train)
Y_grad_pred = grad.predict(X_test)
st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_grad_pred))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_grad_pred))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_grad_pred))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_grad_pred))

fig2 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['homicides_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['homicides_scaled'], Y_grad_pred, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('homicides_scaled')
plt.ylabel('strength')
plt.plot(random_state_2)
st.pyplot(fig2)
```

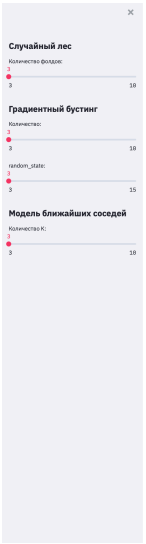
```
st.subheader('Нахождение лучшего////')

params = {
   'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50, 75, 100],
   'max_features': [0.2, 0.3, 0.4, 0.6, 0.8, 0.9, 1.0],
   'min_samples_leaf': [0.01, 0.04, 0.06, 0.08, 0.1]
}

grid_gr = GridSearchCV(estimator=GradientBoostingRegressor(random_state=10),
                       param_grid=params,
                       scoring='neg_mean_squared_error',
                       cv=3,
                       n_jobs=-1)
grid_gr.fit(X_train, Y_train)
st.write(grid_gr.best_params_)

grad1 = GradientBoostingRegressor(n_estimators=100, max_features=0.8, min_samples_leaf=0.01,
random_state=10)
grad1.fit(X_train, Y_train)
Y_grad_pred1 = grad1.predict(X_test)

st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_grad_pred1))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_grad_pred1))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_grad_pred1))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_grad_pred1))

fig1 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['homicides_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['homicides_scaled'], Y_grad_pred1, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('homicides_scaled')
plt.ylabel('strength')
plt.plot(n_estimators_1)
st.pyplot(fig1)

st.subheader('Построение линейной регрессии')

Lin_Reg = LinearRegression().fit(X_train, Y_train)

lr_y_pred = Lin_Reg.predict(X_test)

fig3 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['homicides_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['homicides_scaled'], lr_y_pred, marker='o', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('homicides_scaled')
plt.ylabel('strength')
plt.show()
st.pyplot(fig3)

st.subheader('Tree')

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, Y_train)

lr_y_pred = Lin_Reg.predict(X_test)
```

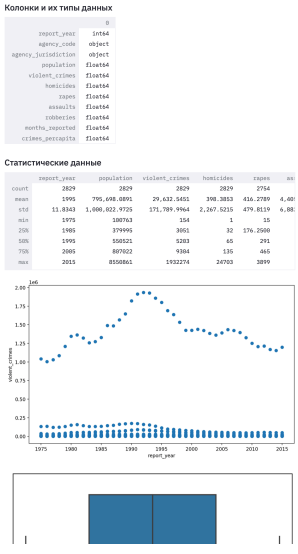
```
fig5 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['homicides_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['homicides_scaled'], lr_y_pred, marker='o', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('homicides_scaled')
plt.ylabel('strength')
plt.show()
st.pyplot(fig5)

st.subheader('Модель ближайших соседей для произвольного гиперпараметра K')

Regressor_5NN = KNeighborsRegressor(n_neighbors = n_estimators_3)
Regressor_5NN.fit(X_train, Y_train)

lr_y_pred = Regressor_5NN.predict(X_test)

fig6 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['homicides_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['homicides_scaled'], lr_y_pred, marker='o', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('homicides_scaled')
plt.ylabel('strength')
plt.show()
st.pyplot(fig6)
```

Результат выполнения









Случайный лес

Количество фолдов:

3

3 10

Градиентный бустинг

Количество:

3

3 10

random_state:

3

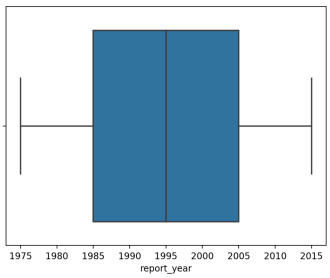
3 15

Модель ближайших соседей

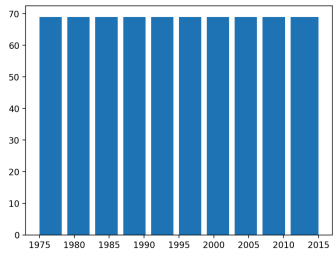
Количество K:

3

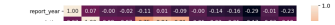
3 10



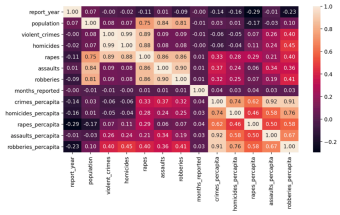
Масштабирование данных



Показать корреляционную матрицу



Показать корреляционную матрицу



RandomForestRegressor

Средняя абсолютная ошибка:

`1489.0700824499941`

Средняя квадратичная ошибка:

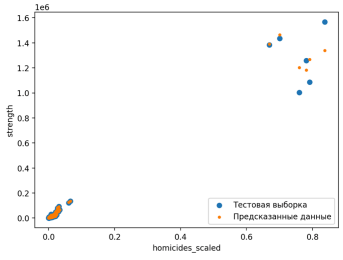
`230667264.0747939`

Median absolute error:

`35.833333333333485`

Коэффициент детерминации:

`0.9878074111256686`



Нахождение лучшего случайного леса


```
{
  "n_estimators": 6
}
```


Средняя абсолютная ошибка:

`487.67932862190855`

Средняя квадратичная ошибка:

`22963052.09790932`

Median absolute error:

`38.91666666666697`

Коэффициент детерминации:

`0.9987065806827654`

Случайный лес

Количество фолдов:

3

3 10

Градиентный бустинг

Количество:

3

3 10

random_state:

3

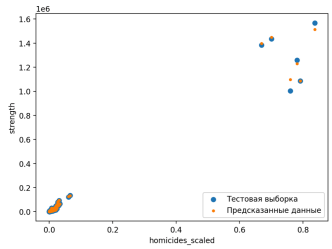
3 15

Модель ближайших соседей

Количество K:

3

3 10



Градиентный бустинг

Средняя абсолютная ошибка:

`27620.992955132027`

Средняя квадратичная ошибка:

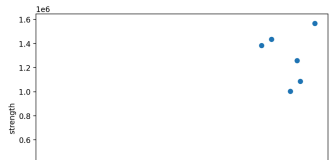
`9205745245.62499`

Median absolute error:

`19277.52266320489`

Коэффициент детерминации:

`0.48147621320269496`



Случайный лес

Количество фолдов:

3

3 10

Градиентный бустинг

Количество:

3

3 10

random_state:

3

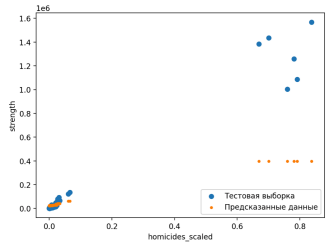
3 15

Модель ближайших соседей

Количество K:

3

3 10



Нахождение лучшего////


```
{
  "max_features" : 0.8
  "min_samples_leaf" : 0.01
  "n_estimators" : 100
}
```


Средняя абсолютная ошибка:

`3208.892020906144`

Средняя квадратичная ошибка:

`936869288.5068679`

Median absolute error:

`339.5114145301977`

Коэффициент детерминации:

`0.9472298007115124`

Случайный лес

Количество фолдов:

3

3 10

Градиентный бустинг

Количество:

3

3 10

random_state:

3

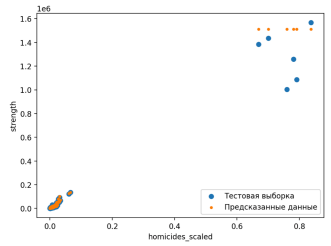
3 15

Модель ближайших соседей

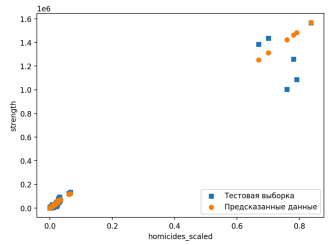
Количество K:

3

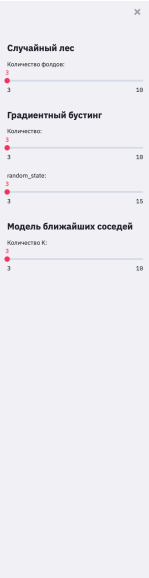
3 10

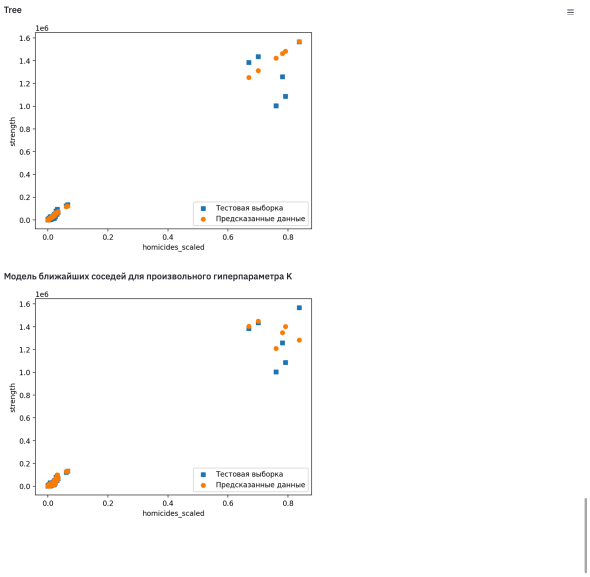


Построение линейной регрессии



Tree





Список источников информации

1. Репозиторий курса по Технологиям машинного обучения [Электронный ресурс] https://github.com/ugapanyuk/ml_course_2021/wiki/COURSE_TMO
2. Документация библиотеки AutoML TPOT [Электронный ресурс] https://github.com/EpistasisLab/tpot
3. Блог Александра Дьяконова. Ансамбли в машинном обучении [Электронный ресурс] https://dyakonov.org/2019/04/19/
4. Блог Александра Дьяконова. Градиентный бустинг [Электронный ресурс] https://dyakonov.org/2017/06/09/