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 _____ИУ_____

КАФЕДРА _____ИУ5_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Анализ данных

Студент ИУ5-63 _____ **Гаврилов Л.Я.**
(Группа) (Подпись, дата) (И.О.Фамилия)

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

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 ___ИУ5____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 2021___ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине __Технологии машинного обучения_____

Студент группы ИУ5-63Б Гаврилов Леонид Янович
(Фамилия, имя, отчество)

Тема курсового проекта __анализ данных_____
_____
_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_____
Источник тематики (кафедра, предприятие, НИР) _____

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

Задание_____
_____
_____
_____
_____

Оформление курсового проекта:

Расчетно-пояснительная записка на 31__ листе формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 20__ г.

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

Студент _____ Гаврилов Л.Я.
(Подпись, дата) (И.О.Фамилия)

Оглавление

1. Введение *4*

2. Основная часть *4*

2.1 Схема типового исследования **4**

2.2 Выполнение задачи **5**

2.3 Файил KR.py **5**

2.4 Скрины **11**

3. Заключение *21*

4. Список использованной литературы *21*

1. Введение

Курсовой проект – самостоятельная часть учебной дисциплины «Технологии машинного обучения» – учебная и практическая исследовательская студенческая работа, направленная на решение комплексной задачи машинного обучения. Результатом курсового проекта является отчет, содержащий описания моделей, тексты программ и результаты экспериментов.

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

В рамках курсового проекта возможно проведение типового или нетипового исследования.

- Типовое исследование - решение задачи машинного обучения на основе материалов дисциплины. Выполняется студентом единолично.
- Нетиповое исследование - решение нестандартной задачи. Тема должна быть согласована с преподавателем. Как правило, такая работа выполняется группой студентов.

2. Основная часть

2.1 Схема типового исследования

Схема типового исследования, проводимого студентом в рамках курсовой работы, содержит выполнение следующих шагов:

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

2. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. 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.

Приведенная схема исследования является рекомендуемой. В зависимости от решаемой задачи возможны модификации.

2.2 Выполнение задачи

2.3 Файил KR.py

```
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.preprocessing import LabelEncoder
from sklearn import tree
import re

def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv('data/cars_dataset.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 = ['year', 'transmission', 'Make']
    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['price']
    # 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()

le = LabelEncoder()
le.fit(data.Make.drop_duplicates())
data.Make = le.transform(data.Make)


le.fit(data.model.drop_duplicates())
data.model = le.transform(data.model)


le.fit(data.transmission.drop_duplicates())
data.transmission = le.transform(data.transmission)

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, ste
p=1)

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

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

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

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

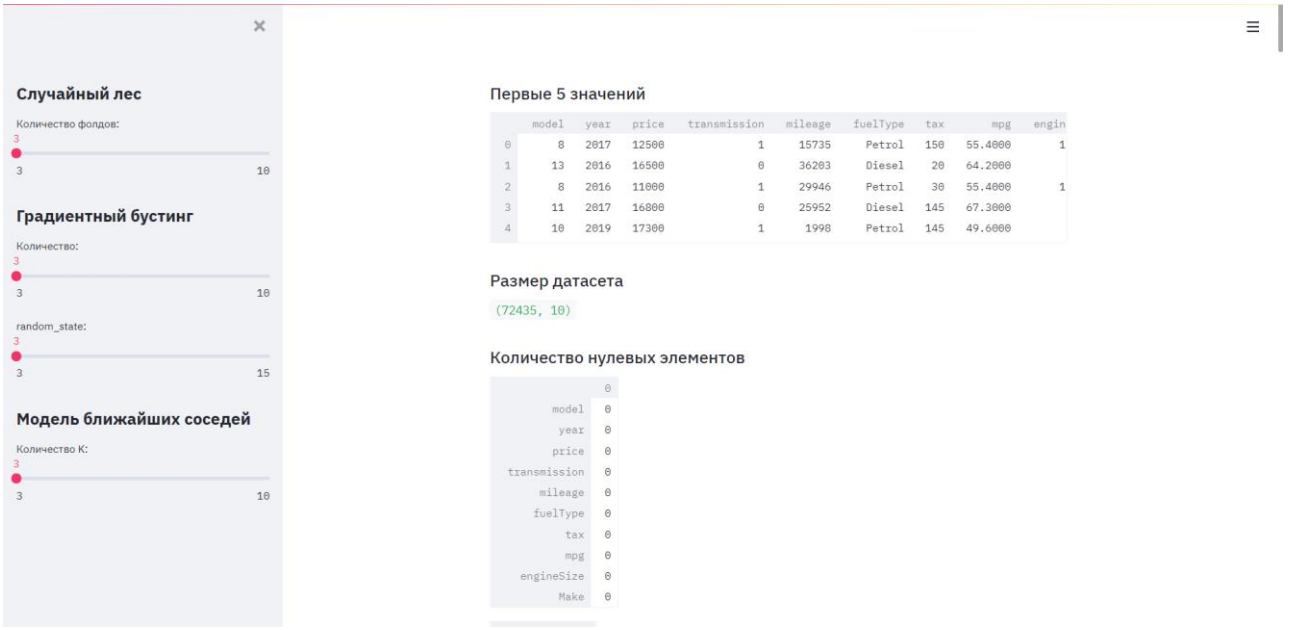
```
st.write(data['year'].value_counts())

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['year'], y=data['price'])
plt.xlabel("year")
plt.ylabel("price")
st.pyplot(fig)

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

st.subheader('Масштабирование данных')
f, ax = plt.subplots()
plt.hist(data['price'], 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['year_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['year_scaled'], Y_predict, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('year_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=15, 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['year_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['year_scaled'], Y_predict3, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('year_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['year_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['year_scaled'], Y_grad_pred, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('year_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=1, 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['year_scaled'], Y_test, marker='o', label='Тестовая выборка')
plt.scatter(X_test['year_scaled'], Y_grad_pred1, marker='.', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('year_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['year_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['year_scaled'], lr_y_pred, marker='o', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('year_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 = clf.predict(X_test)

fig5 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['year_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['year_scaled'], lr_y_pred, marker='o', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('year_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['year_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['year_scaled'], lr_y_pred, marker='o', label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('year_scaled')
plt.ylabel('strength')
plt.show()
st.pyplot(fig6)
```

2.4 Скрины



Первые 5 значений

	model	year	price	transmission	mileage	fuelType	tax	mpg	engin
0	8	2017	12500	1	15735	Petrol	150	55.4000	1
1	13	2016	16500	0	36203	Diesel	20	64.2000	
2	8	2016	11000	1	29946	Petrol	30	55.4000	1
3	11	2017	16800	0	25952	Diesel	145	67.3000	
4	10	2019	17300	1	1998	Petrol	145	49.6000	

Размер датасета

(72435, 10)

Количество нулевых элементов

	0
model	0
year	0
price	0
transmission	0
mileage	0
fuelType	0
tax	0

mpg	0
engineSize	0
Make	0

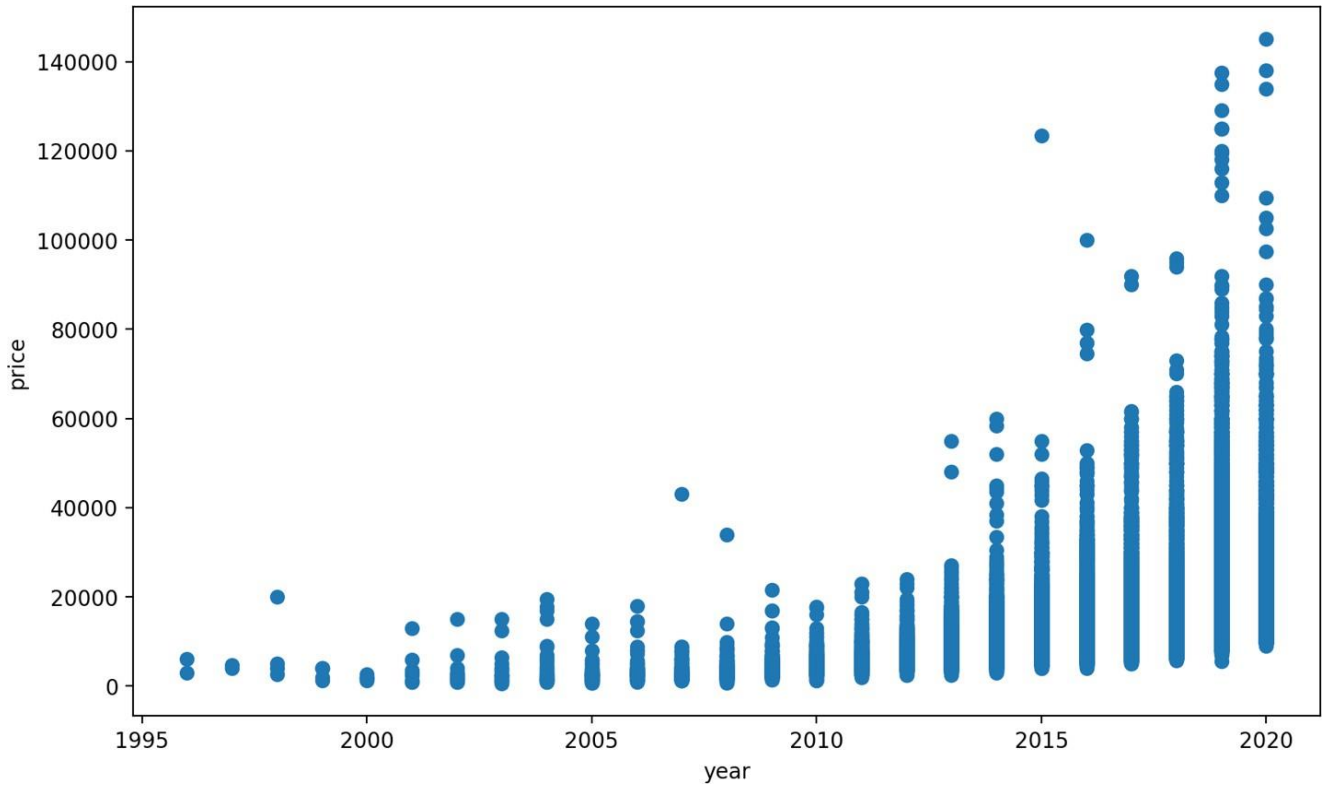
	year
2019	19031
2017	16227
2016	11377
2018	10111
2015	5584
2020	3316
2014	3053
2013	1989
2012	480
2011	322
2010	234

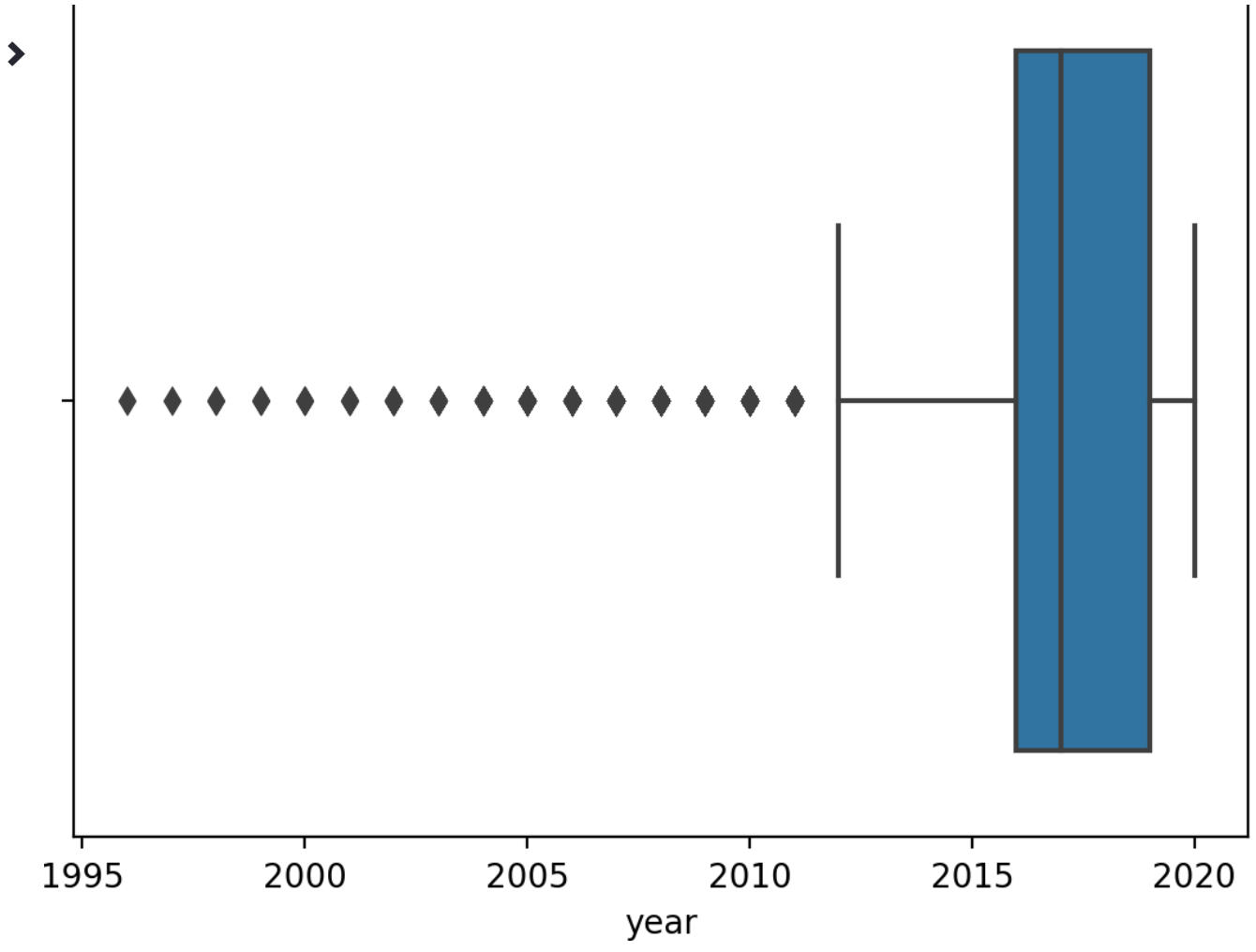
Колонки и их типы данных

	0
model	int32
year	int64
price	int64
transmission	int32
mileage	int64
fuelType	object
tax	float64
mpg	float64
engineSize	float64
Make	int32

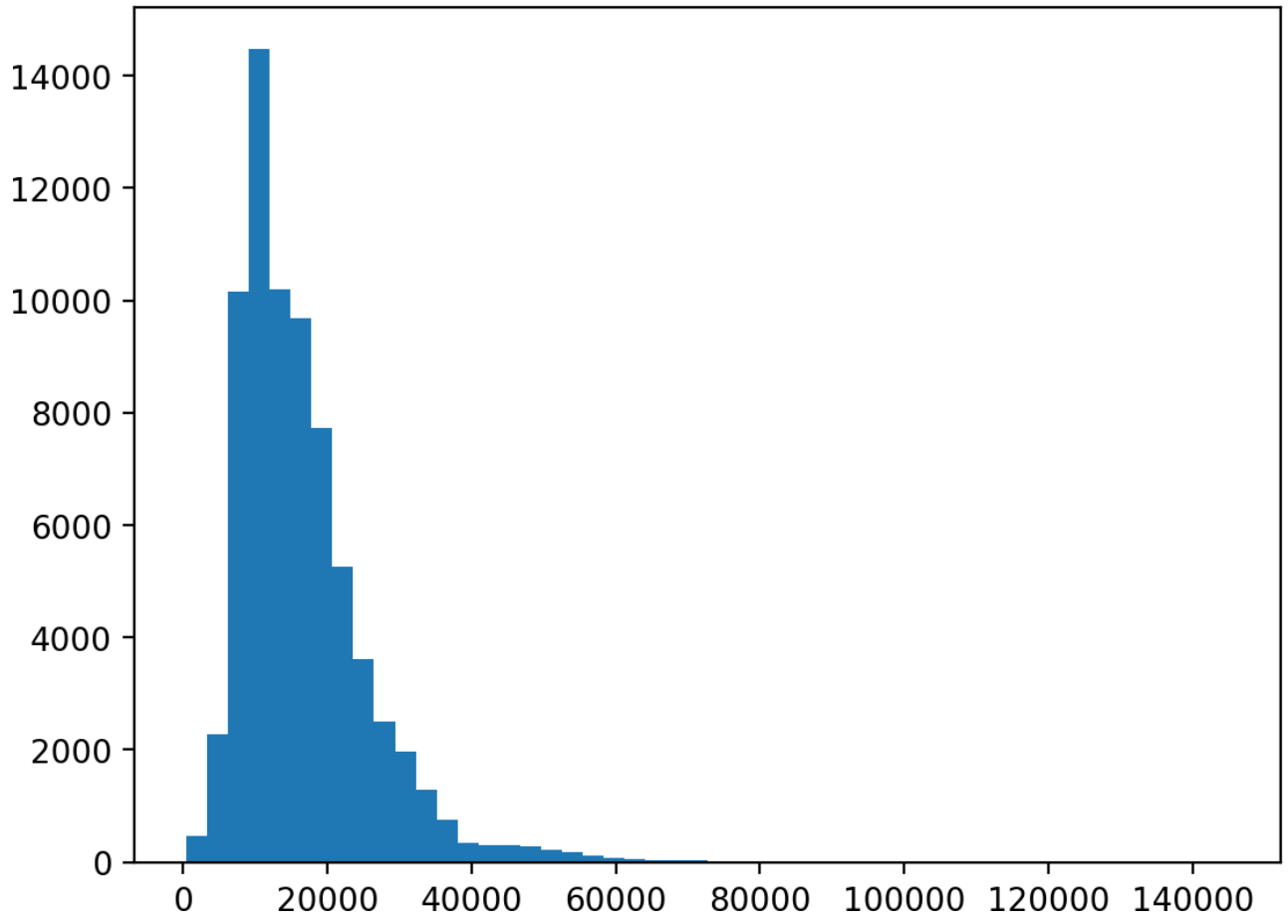
Статистические данные

	model	year	price	transmission	mileage	tax
count	72435	72435	72435	72435	72435	72435
mean	58.4539	2,017.0737	16,580.1587	1.2304	23,176.5171	116.9534
std	41.2293	2.1013	9,299.0288	0.9947	21,331.5156	64.0455
min	0	1996	495	0	1	0
25%	25	2016	10175	1	7,202.5000	30
50%	49	2017	14495	1	17531	145
75%	85	2019	20361	1	32449	145
max	145	2020	145000	3	323000	580

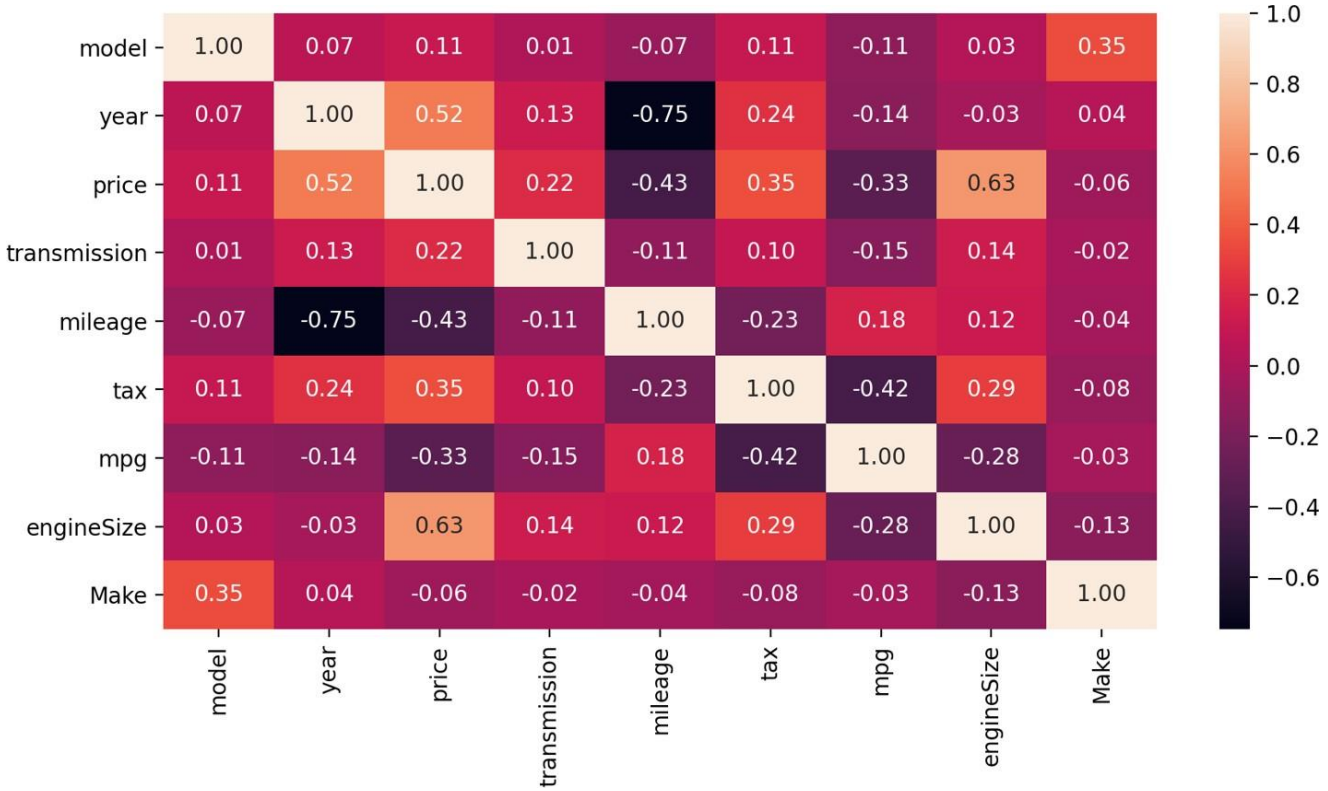




Масштабирование данных



Показать корреляционную матрицу



RandomForestRegressor Средняя абсолютная

ошибка:

3644.9779982834893

Средняя квадратичная ошибка:

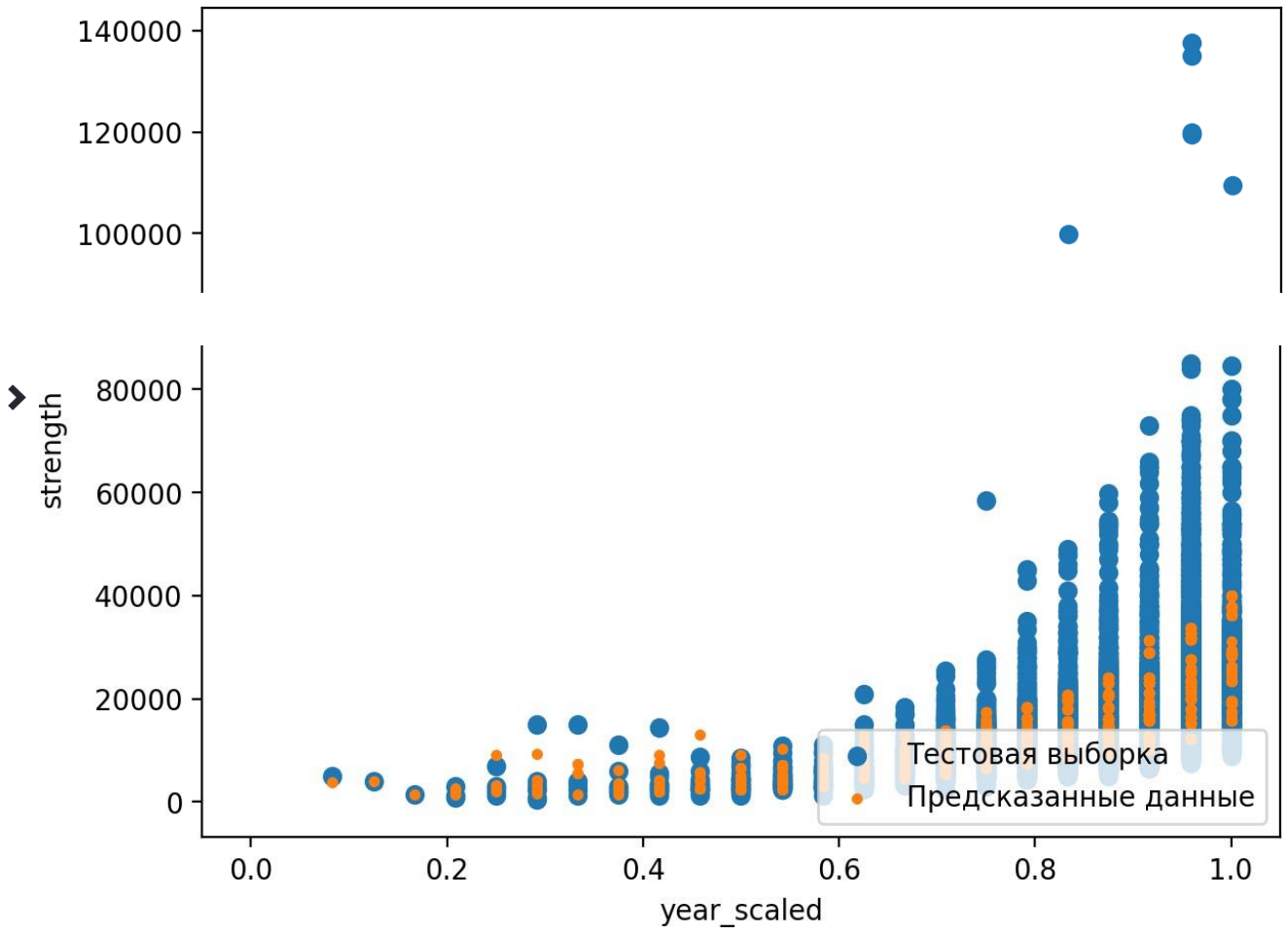
33155029.433419578

Median absolute error:

2523.724206532821

Коэффициент детерминации:

0.630569623172001



Нахождение лучшего случайного леса


```
{
  "n_estimators" : 15
}
```


Средняя абсолютная ошибка:

3643.9411244774246

Средняя квадратичная ошибка:

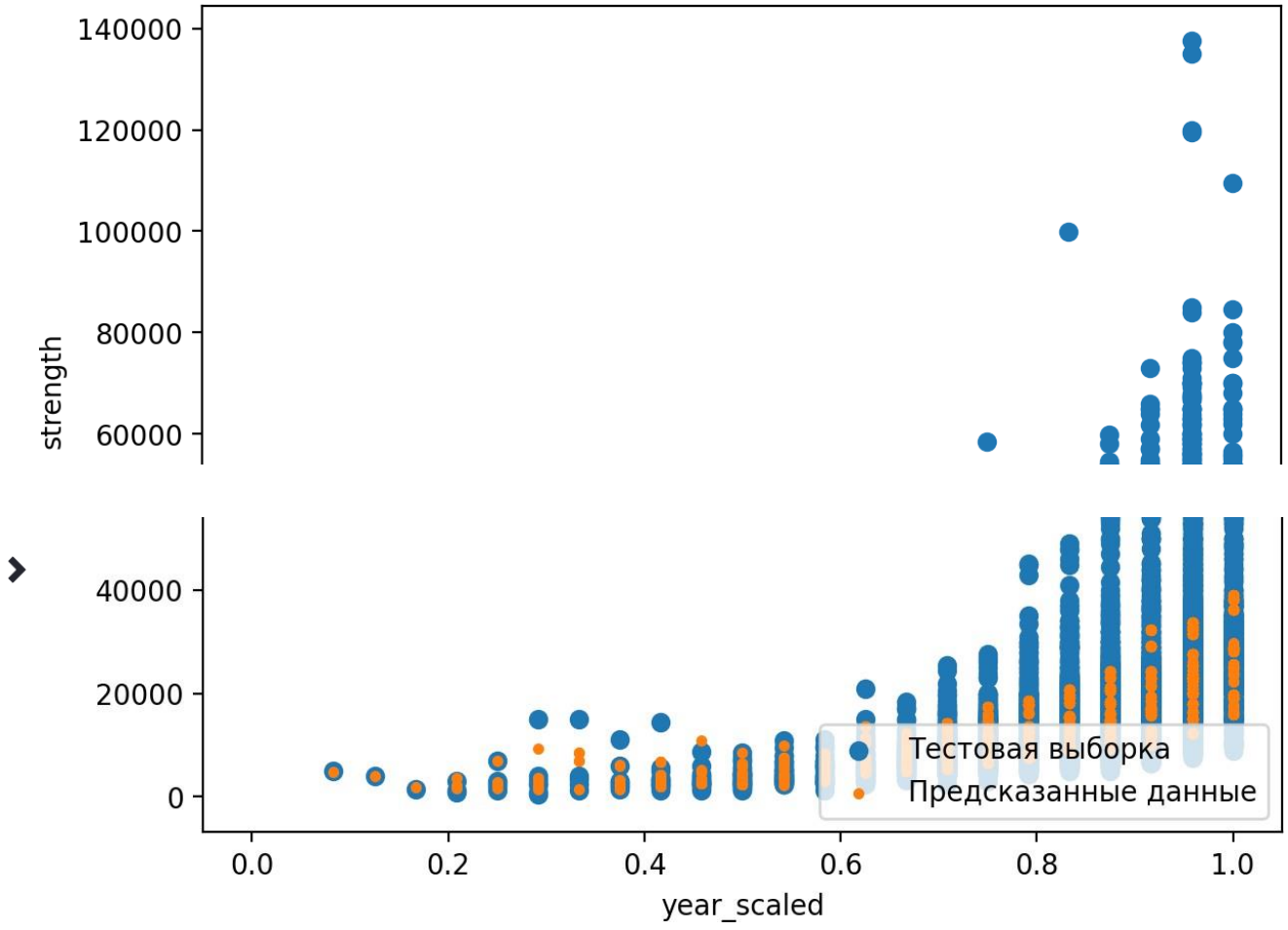
33108463.9034

Median absolute error:

2510.0026922767374

Коэффициент детерминации:

0.6310884802382231



Градиентный бустинг

Средняя абсолютная ошибка:

5773.454058146618

Средняя квадратичная ошибка:

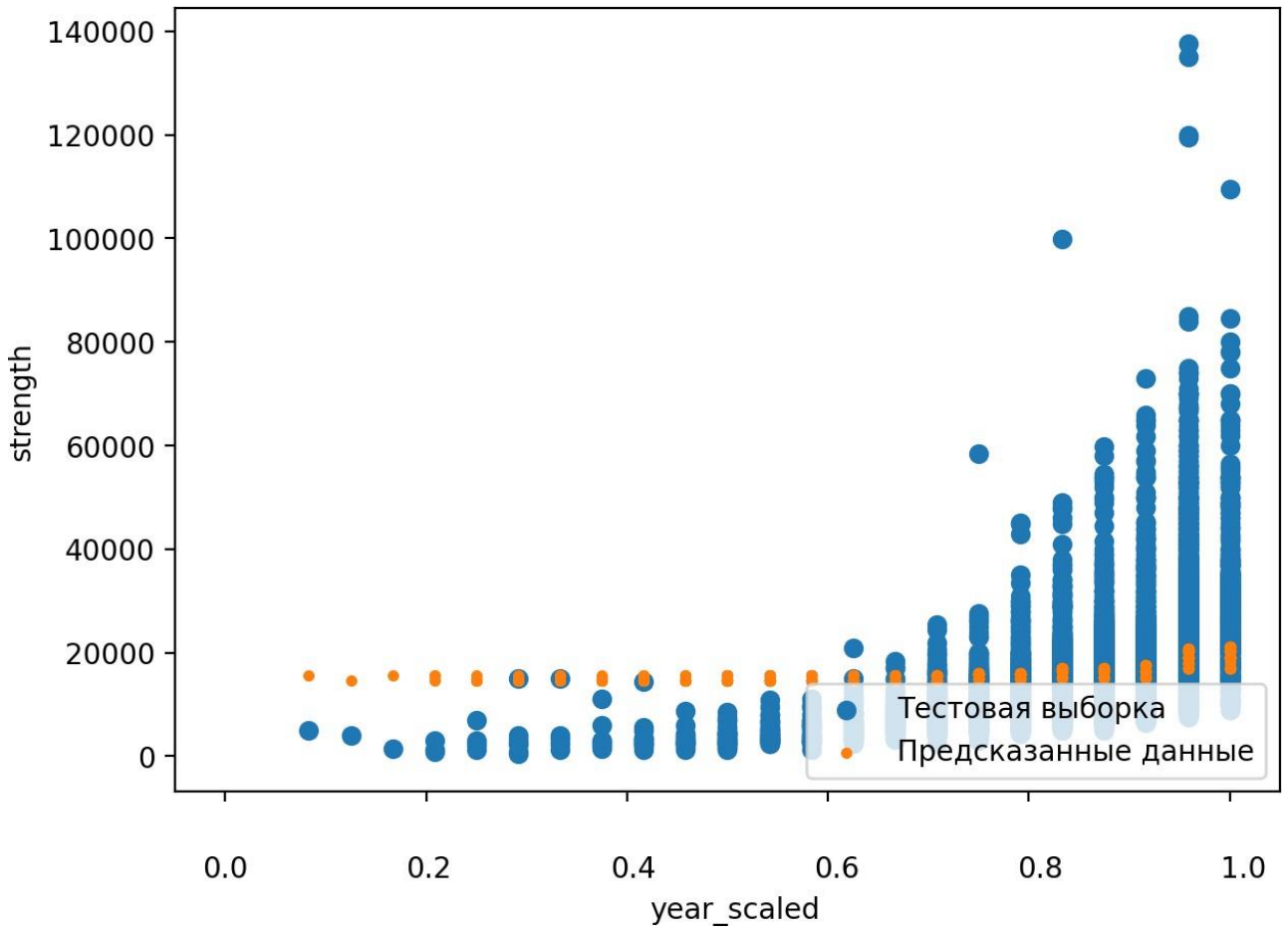
69353265.79167804

Median absolute error:

4746.169167051456

Коэффициент детерминации:

0.22723027083648561



Нахождение лучшего////


```
{
    {
      "max_features" : 1
      "min_samples_leaf" : 0.01
      "n_estimators" : 100
}   }
```


Средняя абсолютная ошибка:

3693.379653802684

Средняя квадратичная ошибка:

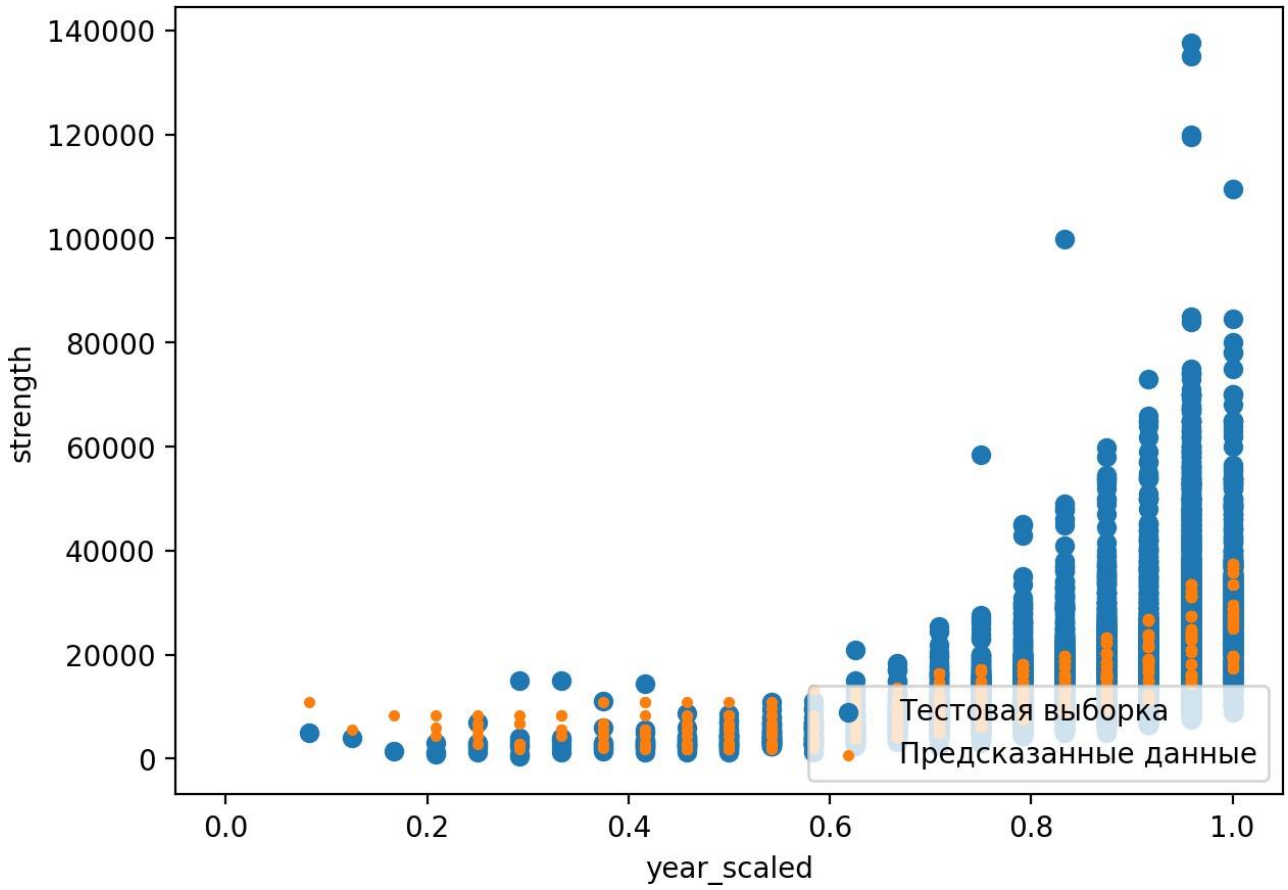
33994667.37794219

Median absolute error:

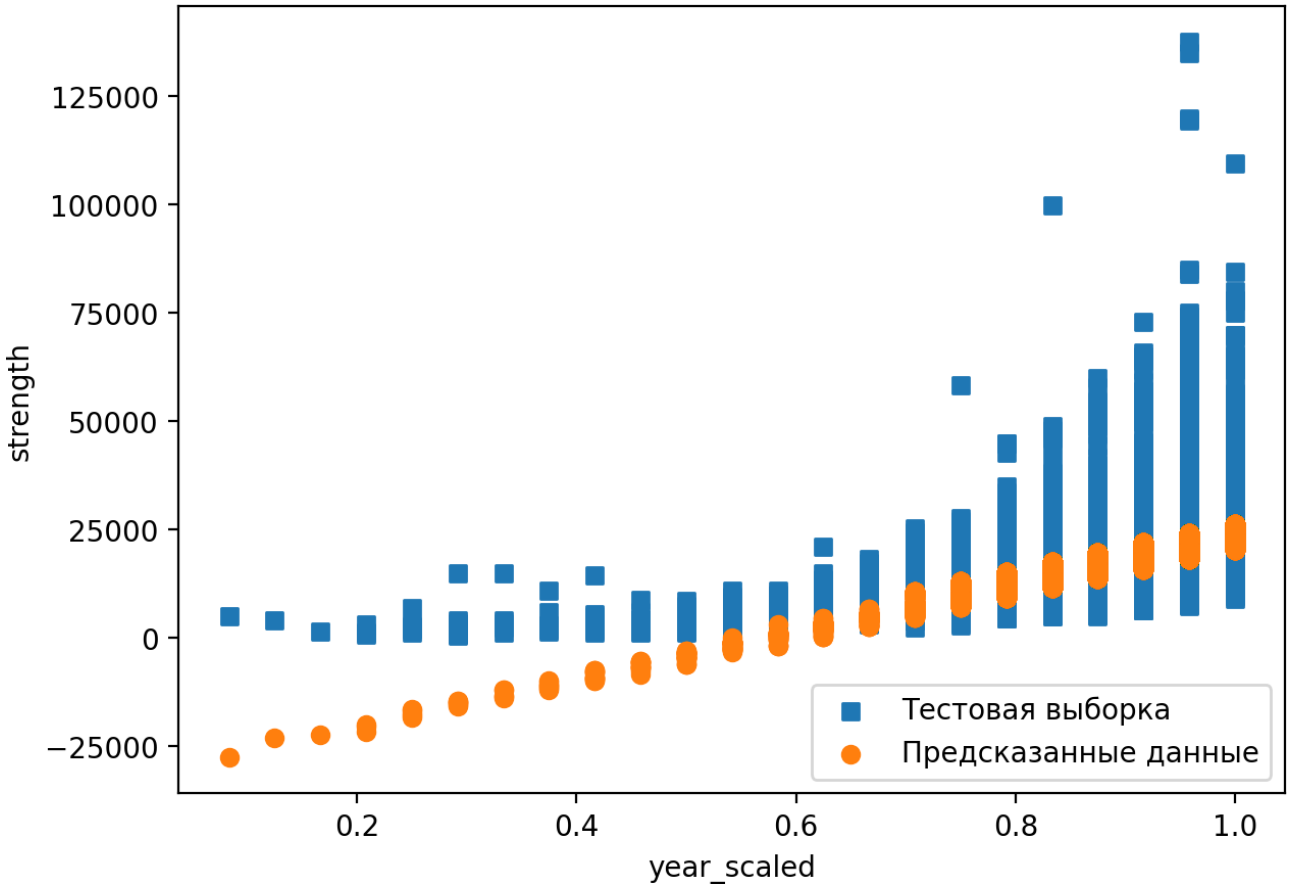
2548.5597665523874

Коэффициент детерминации:

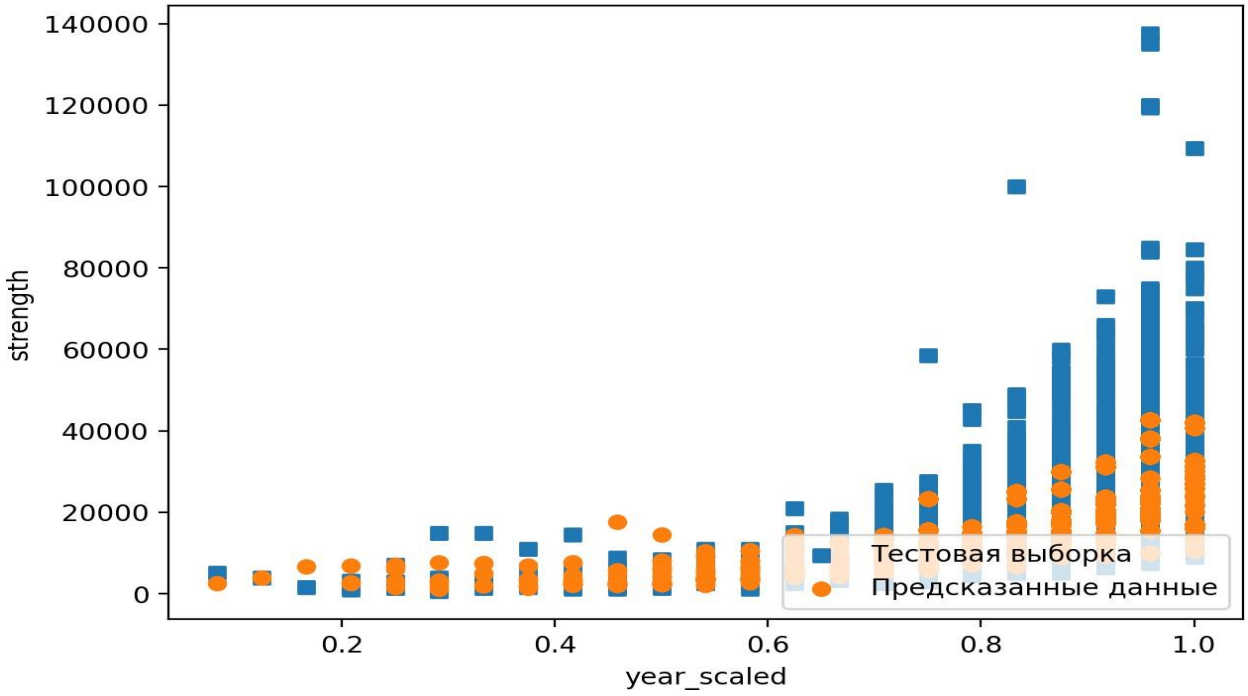
```
0.621213945691244
```



Построение линейной регрессии



Модель ближайших соседей для произвольного гиперпараметра K



3. Заключение

В данной работе был произведен поиск и выбор набора данных для построения моделей машинного обучения. Проведен разведочный анализ данных. Построены графики, необходимые для понимания структуры. Были найдены вспомогательные признаки, улучшающие качество моделей. Проведен корреляционный анализ данных. Были сформированы обучающая и тестовая выборки. Произведен подбор гиперпараметров и найдены оптимальные значения гепермараметров. А также, сформированы выводы о качестве построенных моделей.

4. Список использованной литературы

- https://github.com/ugapanyuk/ml_course_2021/wiki/COURSE_TMO
- https://streamlit.io
- конспекты лекций Гапанюка Ю.Е.