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 «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №6

Выполнила:

студент группы ИУ5-63Б

Латыпова К.Н.

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2021 г.

Задание:

Разработайте макет веб-приложения, предназначенного для анализа данных.

Вариант 1. Макет должен быть реализован для одной модели машинного обучения. Макет должен позволять:

задавать гиперпараметры алгоритма,
производить обучение,
осуществлять просмотр результатов обучения, в том числе в виде графиков.
Вариант 2. Макет должен быть реализован для нескольких моделей машинного обучения. Макет должен позволять:

выбирать модели для обучения,
производить обучение,
осуществлять просмотр результатов обучения, в том числе в виде графиков.
Для разработки рекомендуется использовать следующие (или аналогичные) фреймворки:

streamlit
gradio
dash

Текст программы и экранные формы:

```
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
from sklearn.preprocessing import LabelEncoder
import re

def load_data():
    '''
```

```
    Загрузка данных
    '''
    data = pd.read_csv('data/nics-firearm-background-checks.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 = ['private_sale_long_gun', 'private_sale_other']
    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['private_sale_handgun'].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()

data['private_sale_long_gun'] =
data['private_sale_long_gun'].replace(0,np.nan)
data['private_sale_long_gun'] =
data['private_sale_long_gun'].fillna(data['private_sale_long_gun'].mean())
data['private_sale_handgun'] = data['private_sale_handgun'].replace(0,np.nan)
data['private_sale_handgun'] =
data['private_sale_handgun'].fillna(data['private_sale_handgun'].mean())
data['private_sale_other'] = data['private_sale_other'].replace(0,np.nan)
data['private_sale_other'] =
data['private_sale_other'].fillna(data['private_sale_other'].mean())

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)
```

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

st.write(data['state'].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['permit'], y=data['permit_recheck'])
plt.xlabel("permit")
plt.ylabel("permit_recheck")
st.pyplot(fig)

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

st.subheader('Масштабирование данных')
f, ax = plt.subplots()
plt.hist(data['permit'], 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['private_sale_long_gun_scaled'], Y_test, marker='o',
label='Тестовая выборка')
plt.scatter(X_test['private_sale_long_gun_scaled'], Y_predict, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('private_sale_long_gun_scaled')
plt.ylabel('suicides_no')
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=75, 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['private_sale_long_gun_scaled'], Y_test, marker='o',
label='Тестовая выборка')
plt.scatter(X_test['private_sale_long_gun_scaled'], Y_predict3, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('private_sale_long_gun_scaled')
plt.ylabel('suicides_no')
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['private_sale_long_gun_scaled'], Y_test, marker='o',
label='Тестовая выборка')
plt.scatter(X_test['private_sale_long_gun_scaled'], Y_grad_pred, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('private_sale_long_gun_scaled')
plt.ylabel('suicides_no')
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['private_sale_long_gun_scaled'], Y_test, marker='o',
label='Тестовая выборка')
plt.scatter(X_test['private_sale_long_gun_scaled'], Y_grad_pred1, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('private_sale_long_gun_scaled')
plt.ylabel('suicides_no')
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)

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

fig3 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['private_sale_long_gun_scaled'], Y_test, marker='s',
label='Тестовая выборка')
plt.scatter(X_test['private_sale_long_gun_scaled'], lr_y_pred, marker='o',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('private_sale_long_gun_scaled')
plt.ylabel('suicides_no')
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['private_sale_long_gun_scaled'], Y_test, marker='s',
label='Тестовая выборка')
plt.scatter(X_test['private_sale_long_gun_scaled'], lr_y_pred, marker='o',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('private_sale_long_gun_scaled')
plt.ylabel('suicides_no')
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['private_sale_long_gun_scaled'], Y_test, marker='s',
label='Тестовая выборка')
plt.scatter(X_test['private_sale_long_gun_scaled'], lr_y_pred, marker='o',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('private_sale_long_gun_scaled')
plt.ylabel('suicides_no')
plt.show()
st.pyplot(fig6)
```

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

Первые 5 значений

	month	state	permit	permit_recheck	handgun	long_gun	other	mu
0	2021-05	Alabama	28248	317	21664	12423	1334	
1	2021-05	Alaska	307	7	3368	2701	323	
2	2021-05	Arizona	21767	695	20984	9259	1676	
3	2021-05	Arkansas	7697	1171	8501	5072	422	
4	2021-05	California	20742	11514	40160	25824	5576	

Размер датасета

```
(14905, 27)
```

Количество нулевых элементов

	0
month	0
state	0
permit	24
permit_recheck	11385
handgun	20
long_gun	19
other	6985
multiple	0
admin	23
prepawn_handgun	1943
prepawn_long_gun	1945

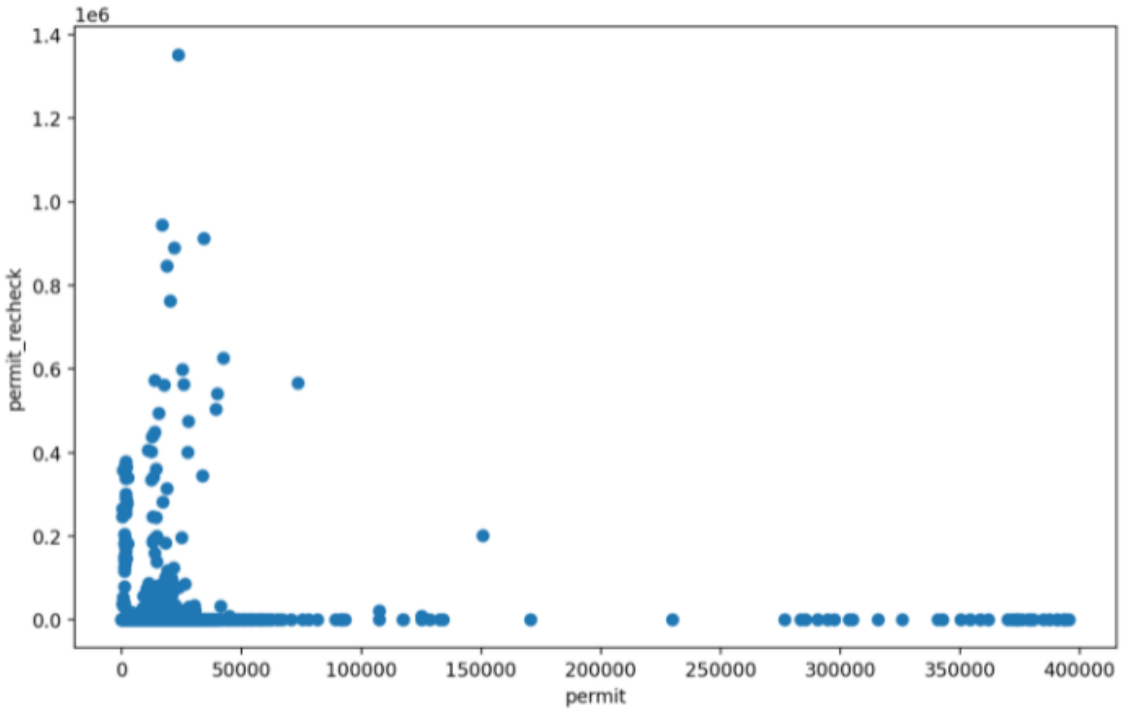
	state
Idaho	271
North Dakota	271
Colorado	271
Missouri	271
California	271
Oregon	271
New Hampshire	271
Ohio	271
Oklahoma	271
Mississippi	271
Hawaii	271

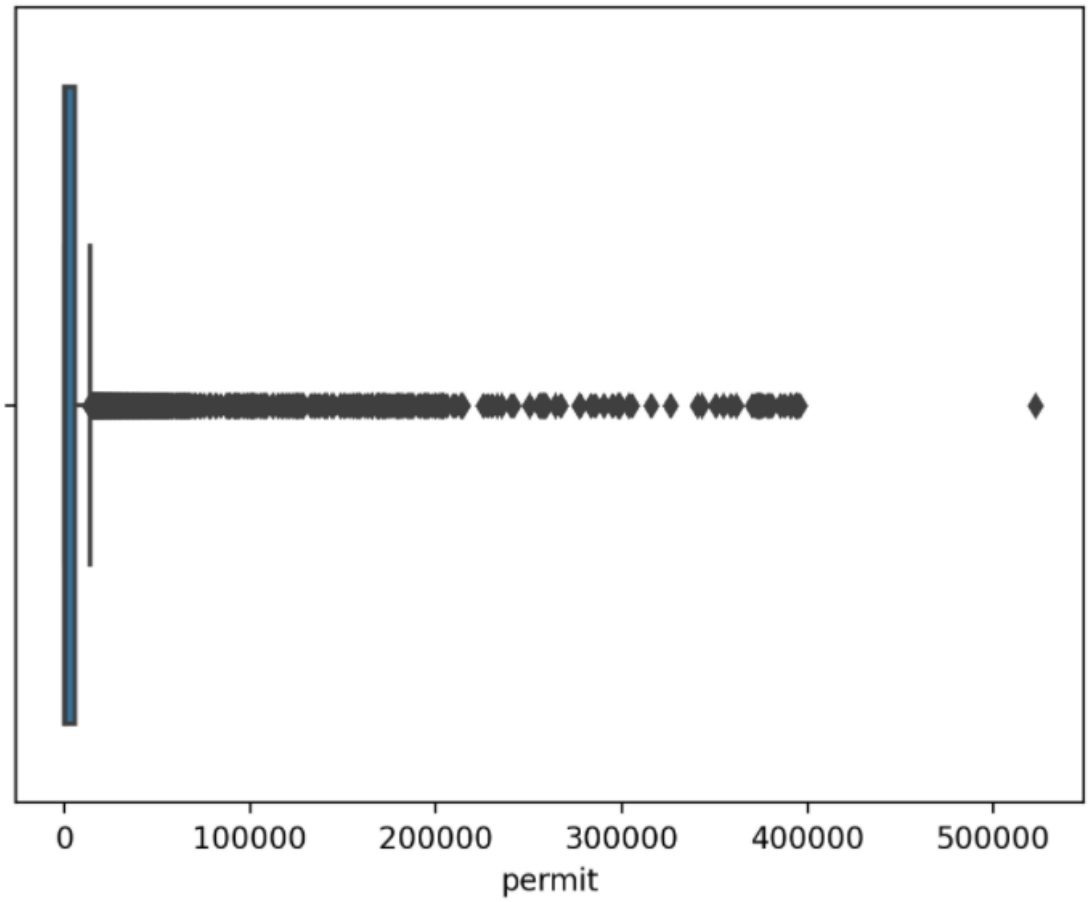
Колонки и их типы данных

	0
month	object
state	object
permit	float64
permit_recheck	float64
handgun	float64
long_gun	float64
other	float64
multiple	int64
admin	float64
prepawn_handgun	float64
prepawn_long_gun	float64

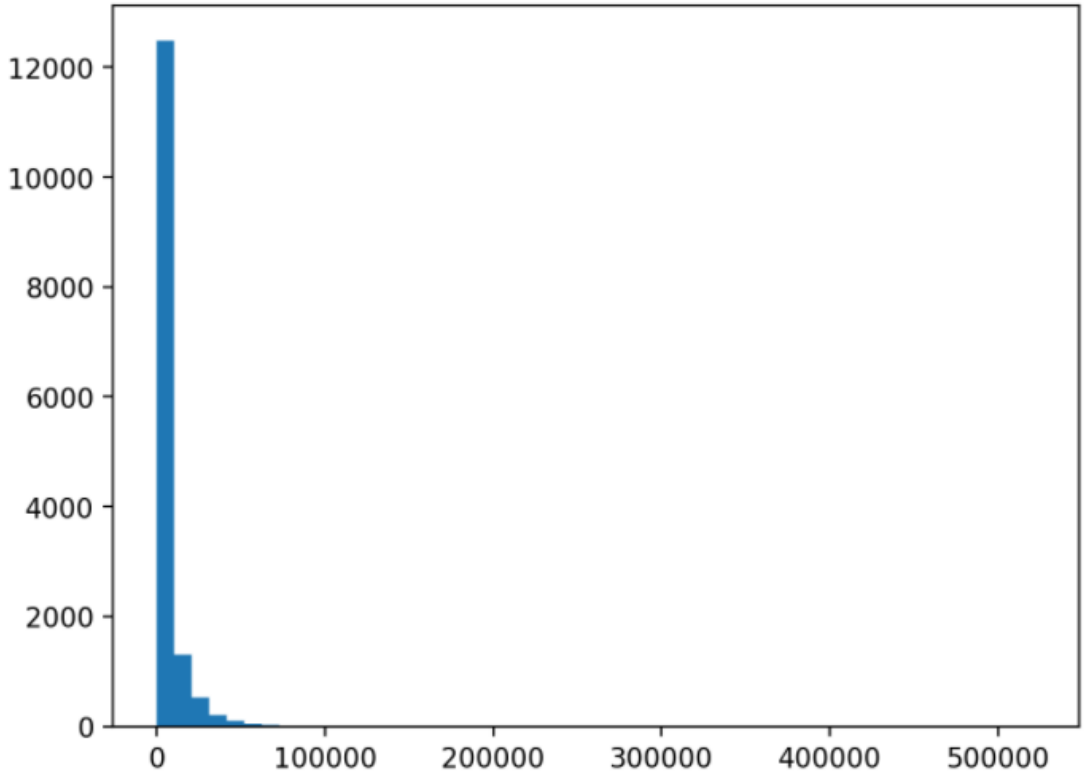
Статистические данные

	permit	permit_recheck	handgun	long_gun	other	mult
count	14881	3520	14885	14886	7920	1
mean	7,262.4230	9,121.7455	7,126.2406	7,979.9966	550.7981	300.
std	25,979.4154	61,210.8606	10,625.2507	9,223.3996	1,381.4198	780.
min	0	0	0	0	0	
25%	0	0	1039	2,176.2500	30	
50%	815	0	3529	5270	179.5000	
75%	5620	76.2500	8654	10,754.7500	565.2500	
max	522188	1350676	147714	108058	77929	3

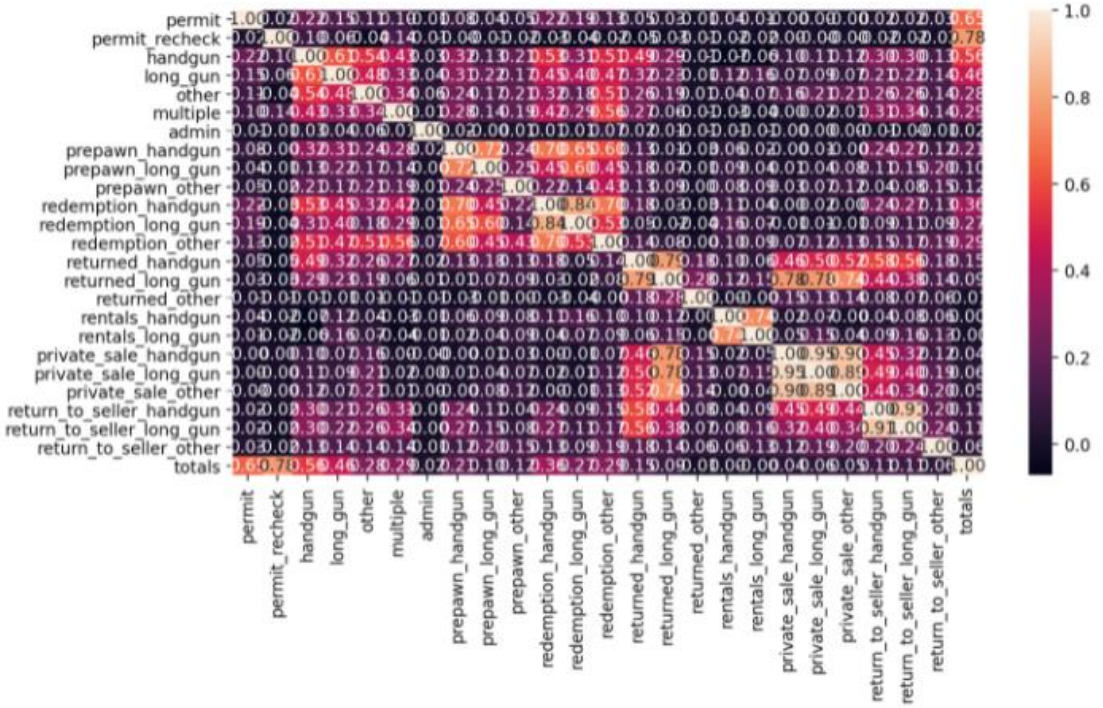

1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
permit_recheck
0
50000
100000
150000
200000
250000
300000
350000
400000
permit


0
100000
200000
300000
400000
500000
permit

Масштабирование данных



Показать корреляционную матрицу



RandomForestRegressor

Средняя абсолютная ошибка:

```
4.0313922236345565
```

Средняя квадратичная ошибка:

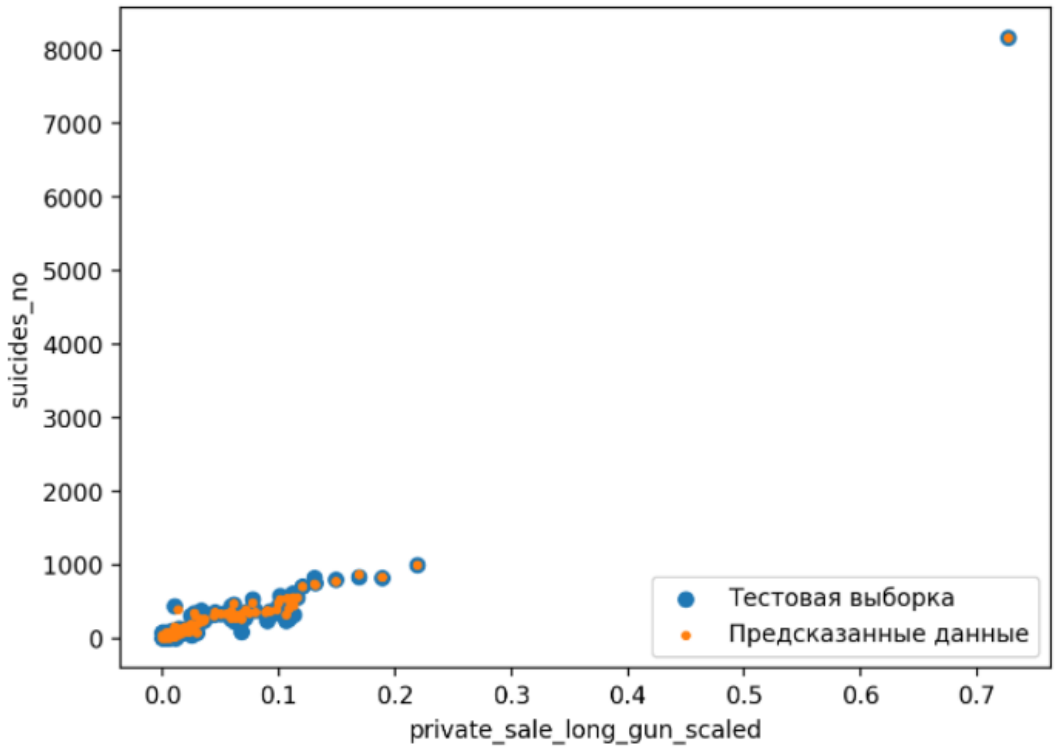
```
234.63550227096212
```

Median absolute error:

```
0.2238378370170011
```

Коэффициент детерминации:

```
0.9906120077035164
```



Нахождение лучшего случайного леса


```
{
  "n_estimators" : 75
}
```


Средняя абсолютная ошибка:

4.0017446483961665

Средняя квадратичная ошибка:

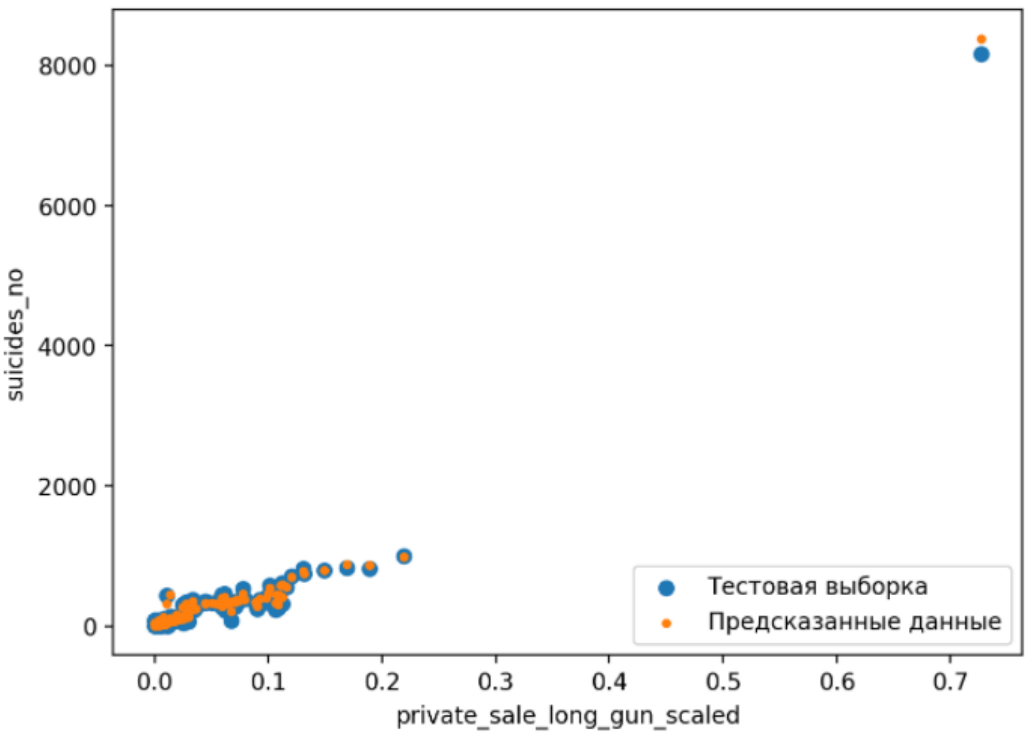
200.4414527863315

Median absolute error:

0.26450401476691354

Коэффициент детерминации:

0.9919801445372023



Градиентный бустинг

Средняя абсолютная ошибка:

```
13.203970653620845
```

Средняя квадратичная ошибка:

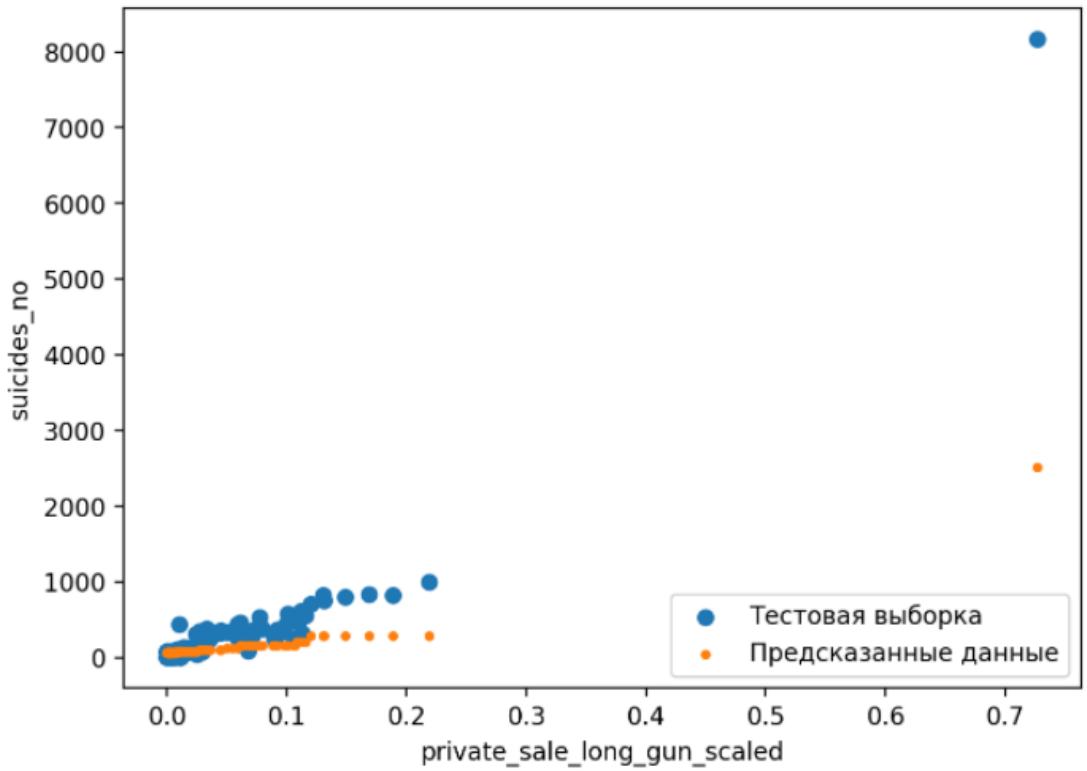
```
12538.276465301355
```

Median absolute error:

```
0.11381861612755984
```

Коэффициент детерминации:

```
0.4983314897866662
```



Нахождение лучшего////


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


Средняя абсолютная ошибка:

`11.58125003881188`

Средняя квадратичная ошибка:

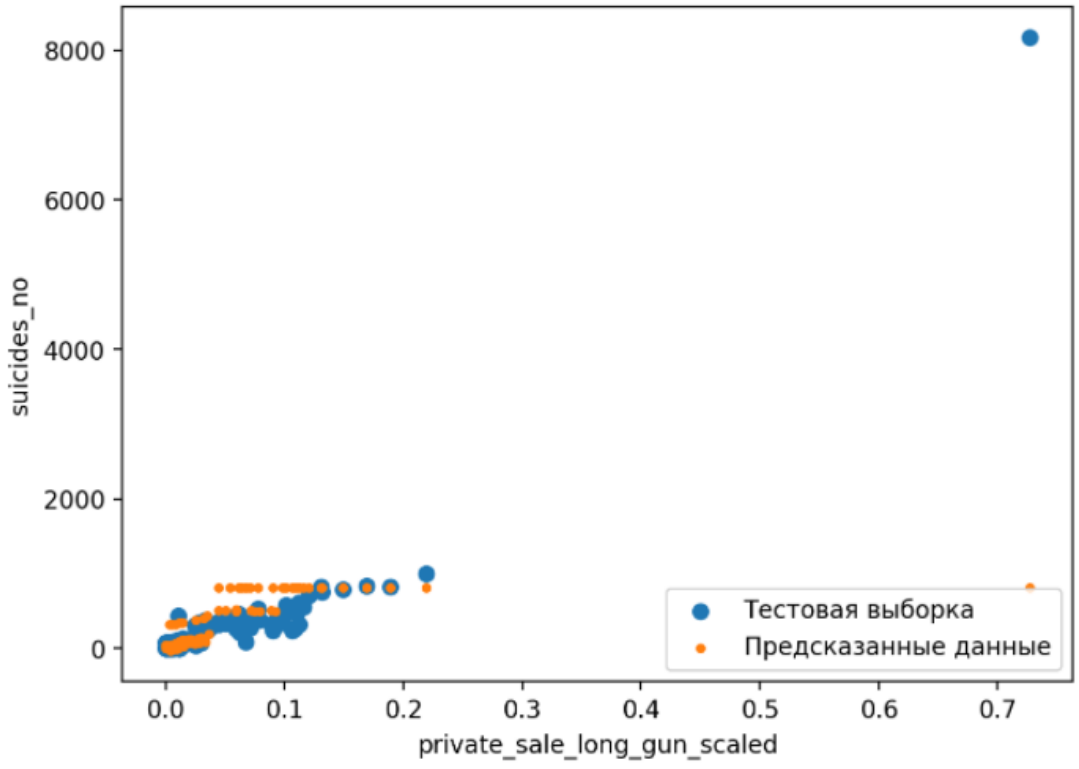
`20249.10319419001`

Median absolute error:

`0.22738303232266333`

Коэффициент детерминации:

`0.18981389023461703`



Построение линейной регрессии

Средняя абсолютная ошибка:

```
9.783830443571587
```

Средняя квадратичная ошибка:

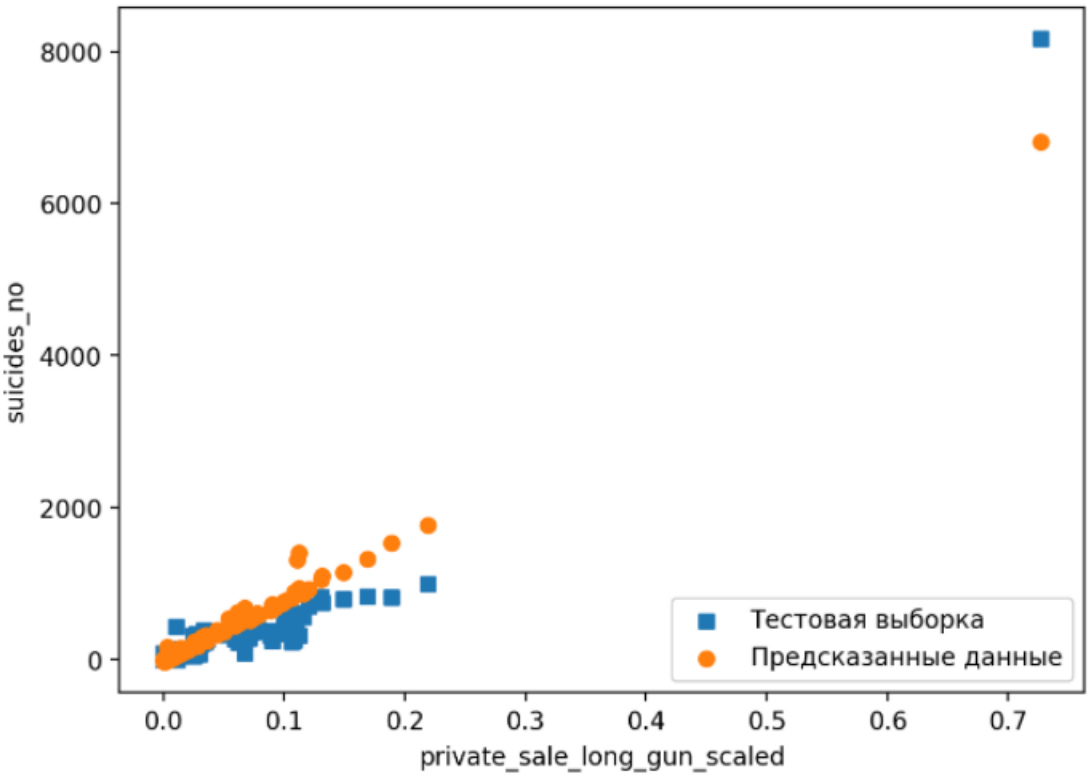
```
2886.1396550885124
```

Median absolute error:

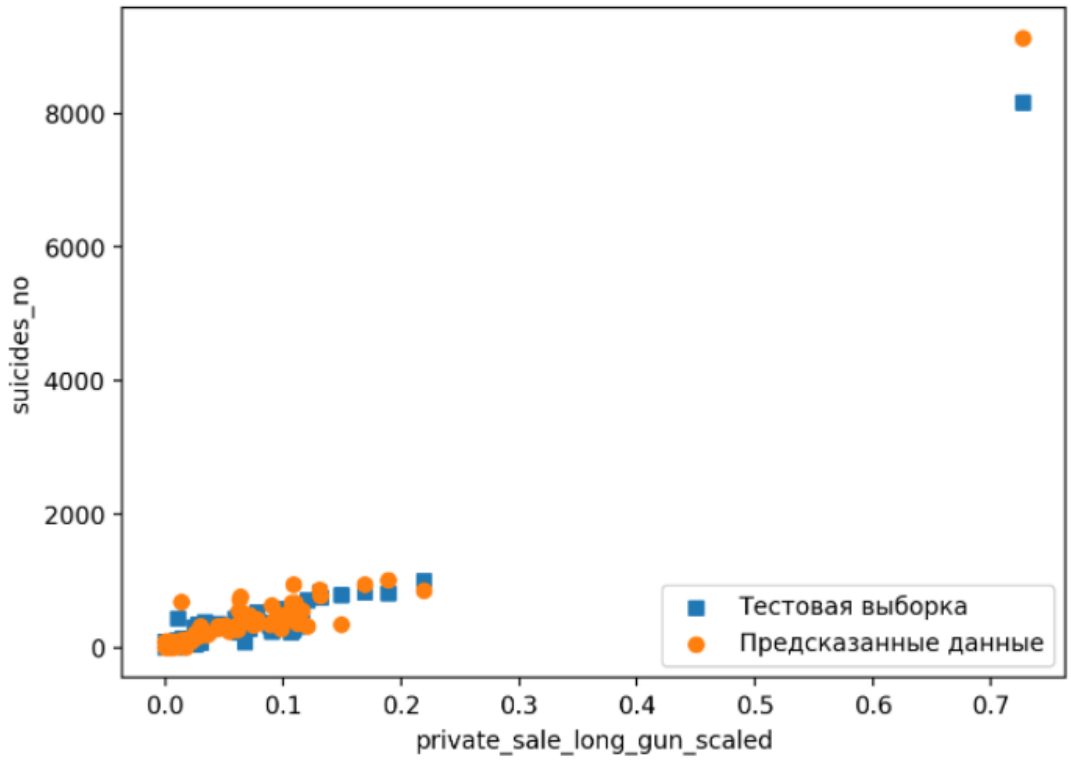
```
0.3982726118590847
```

Коэффициент детерминации:

```
0.88452277431888
```



Tree



Модель ближайших соседей для произвольного гиперпараметра K