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 «Автоматизированные системы обработки информации и управления»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Решение комплексной задачи машинного обучения

Студент ИУ5ц-82Б _____ Акимкин М.Г.
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы _____ Гапанюк Ю.Е.
(Подпись, дата) (И.О.Фамилия)

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

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине ____ Технологии машинного обучения _____

Студент группы _____ИУ5ц-82Б_____

____Акимкин Максим Григорьевич_____
(Фамилия, имя, отчество)

Тема курсовой работы ____ решение комплексной задачи машинного обучения _____
_____
_____

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

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

Задание ____ решение задачи машинного обучения на основе материалов дисциплины. Выполняется студентом единолично. _____
_____
_____
_____

Оформление курсовой работы:

Расчетно-пояснительная записка на 12 листах формата А4.

_____
_____
_____
_____

Дата выдачи задания « 10» мая 2021 г.

Руководитель курсовой работы _____ Гапанюк Ю.Е.
(Подпись, дата) (И.О.Фамилия)

Студент _____ Акимкин М.Г.
(Подпись, дата) (И.О.Фамилия)

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

Запуск курсового проекта:

1. Открыть в терминале директорию проекта.

2. Прописать в терминале команду: streamlit run 6lab.py.

    Прим.: Название "6lab" осталось исторически, исходя из хода разработки проекта.

3. Откроется браузер с запущенным проектом.

    Прим.: Если браузер не открылся автоматически, необходимо перейти по адресу, который был выведен в терминал; адреса в терминале два:

    Пример вывода в терминале:

    "You can now view your Streamlit app in your browser.

    Local URL: http://localhost:8501

    Network URL: http://172.16.93.210:8501"

    Можно перейти, как по первому (локальный – более предпочтителен), так и по второму (сетевой).

Работа в курсовом проекте:

1. Браузер откроется на "основном окне проекта" (*см. Рис.1*):

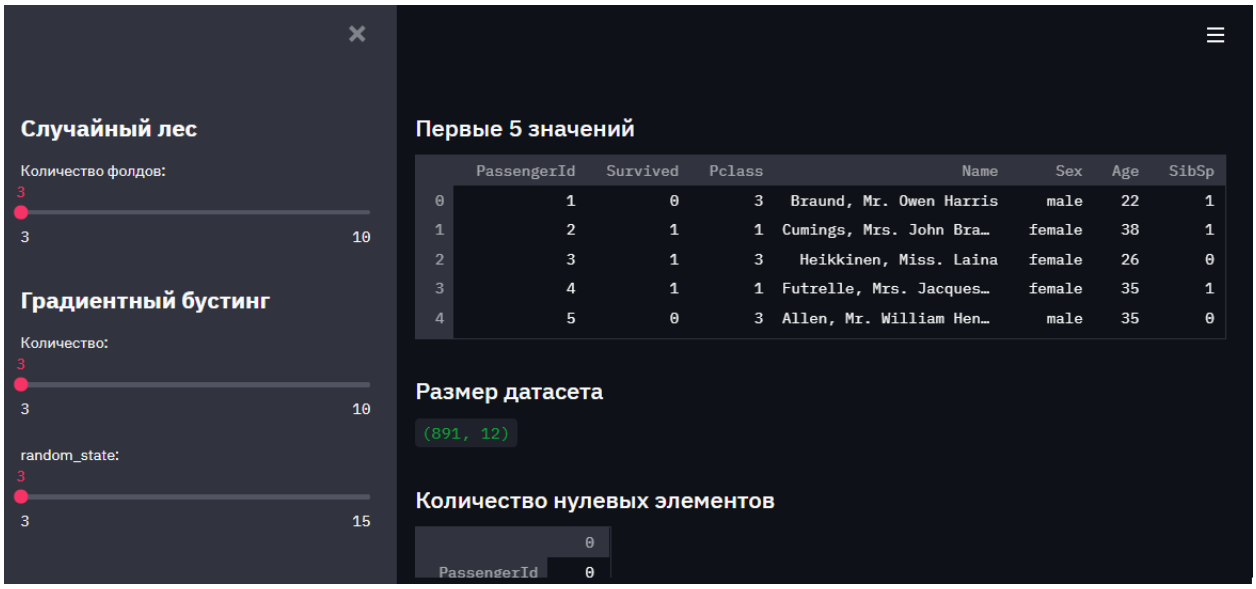

Рис.1 – "*Основное окно проекта*"

Слева – меню ручной настройки двух из пяти моделей: Случайного леса и Градиентного бустинга (*см. Рис.1*).

Справа – информация об использующимся датасете: Первые 5 значений (*см. Рис.1*), Размер (*см. Рис.1*), Количество нулевых элементов(*см. Рис.2,3*), Колонки и их типы данных (*см. Рис.4*).

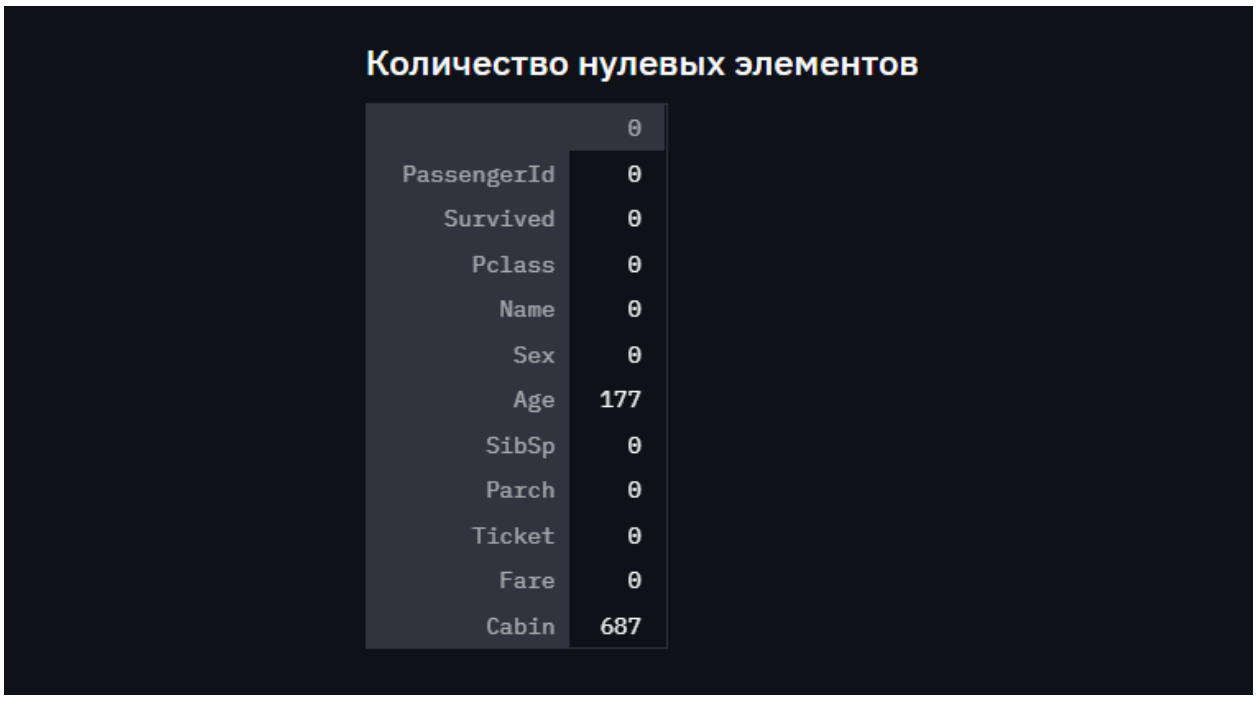

Рис.2 – "*Количество нулевых элементов (1)*"

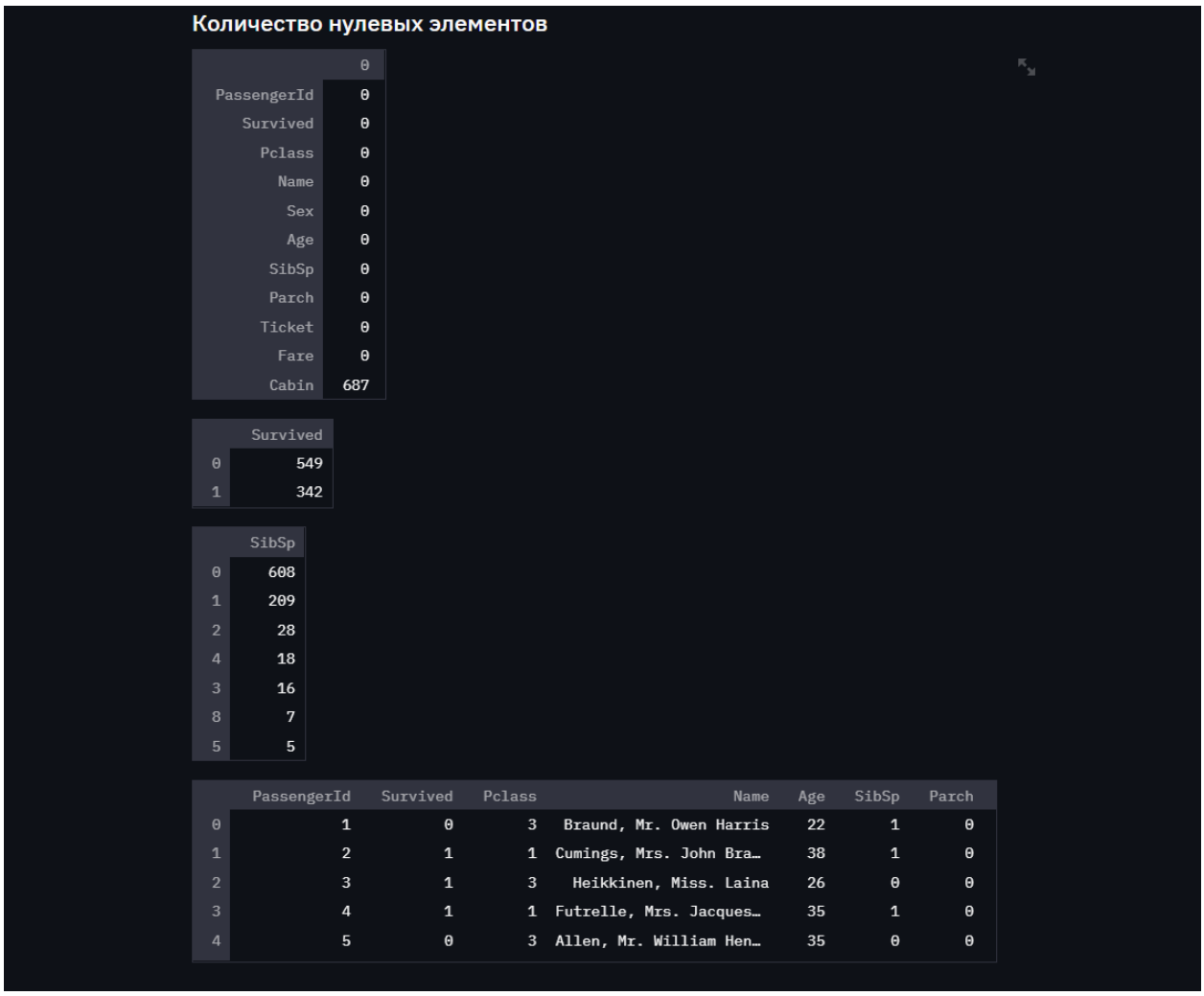

Рис.3 – "*Количество нулевых элементов (2)*"

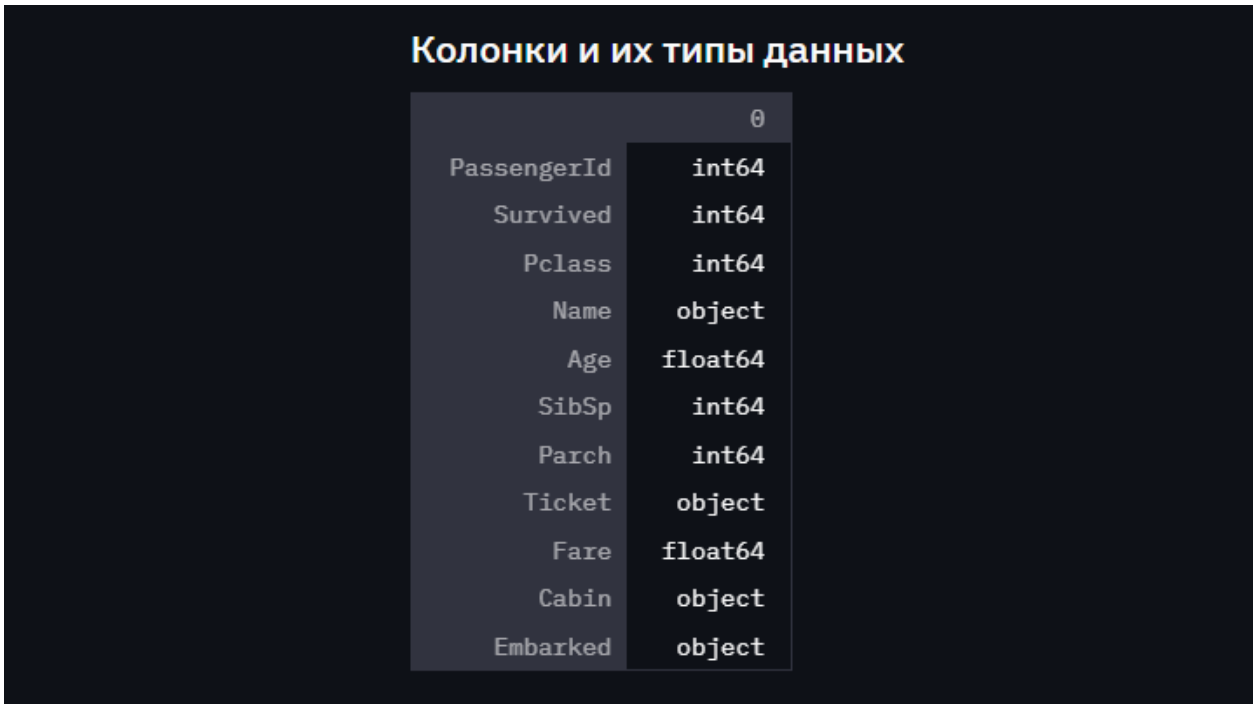

Рис.4 – "*Колонки и их типы данных*"

2. Далее, вниз по странице, – статистические данные (см. Рис.5,6,7).

Статистические данные

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	IsM
count	891	891	891	891	891	891	891	
mean	446	0.3838	2.3086	29.6991	0.5230	0.3816	32.2042	0.6
std	257.3538	0.4866	0.8361	13.0020	1.1027	0.8061	49.6934	0.4
min	1	0	1	0.4200	0	0	0	
25%	223.5000	0	2	22	0	0	7.9104	
50%	446	0	3	29.6991	0	0	14.4542	
75%	668.5000	1	3	35	1	0	31	
max	891	1	3	80	8	6	512.3292	

Рис.5 – "*Статистические данные в форме таблицы*"

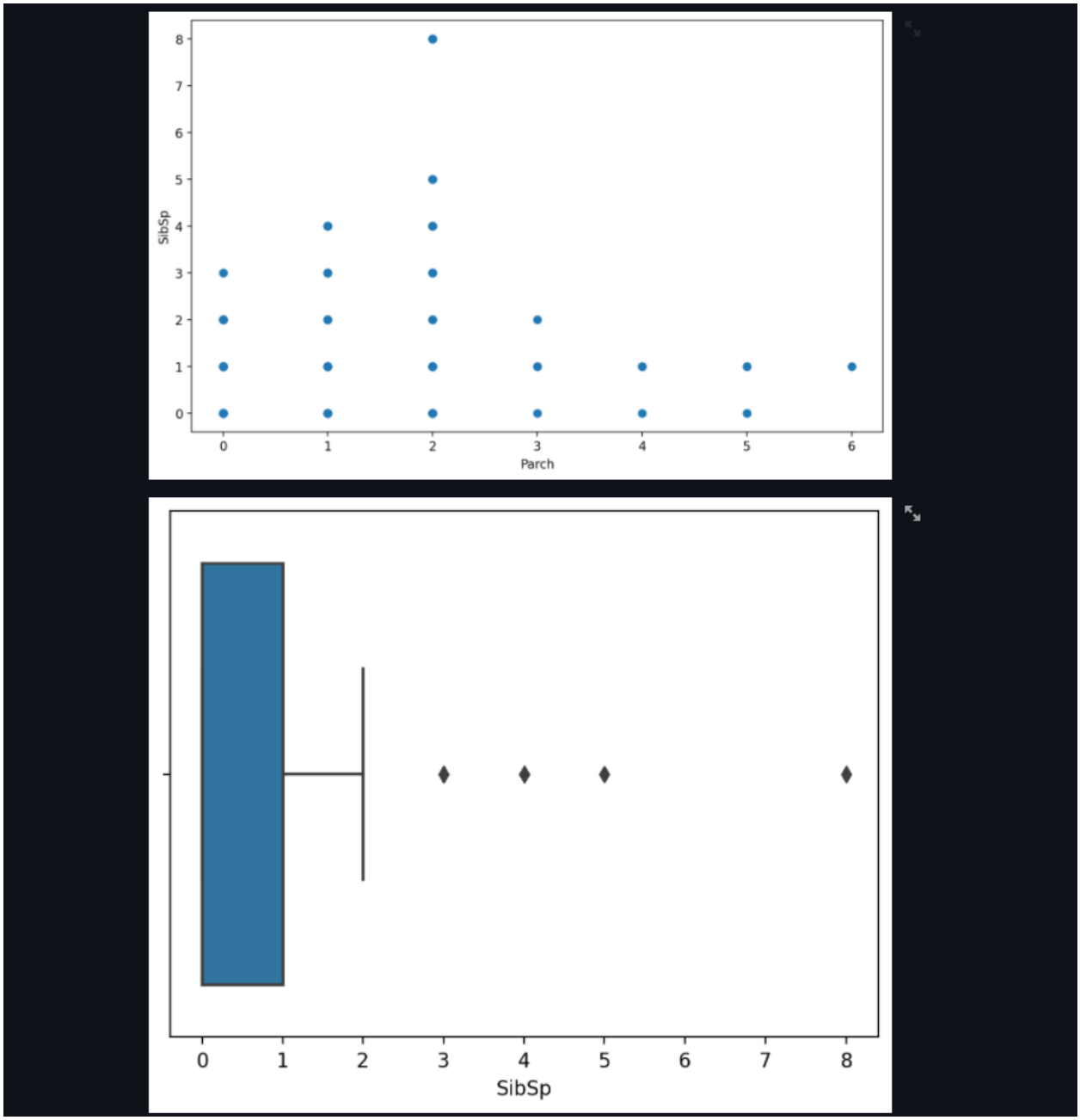

Рис.6 – "*Статистические данные в графическом виде (1)*"

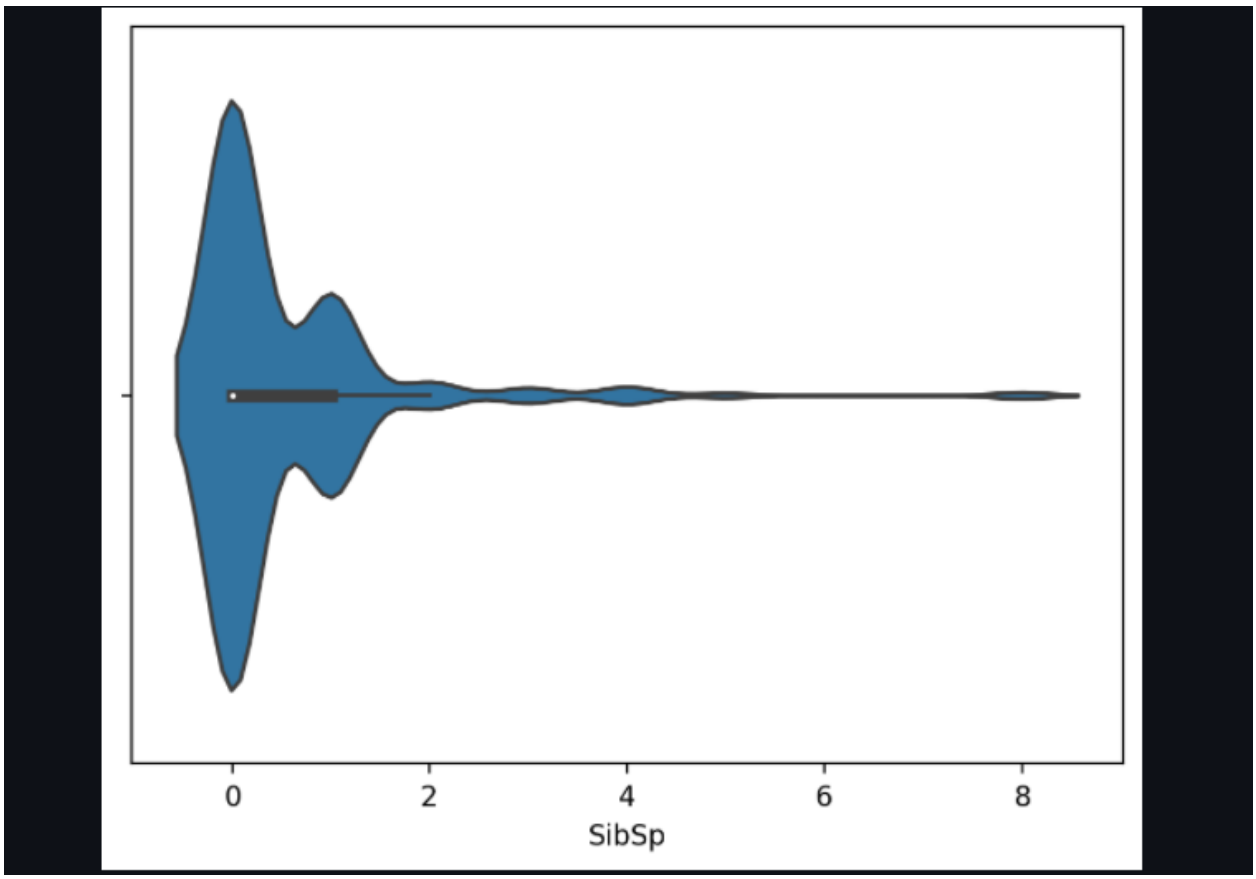

Рис.7 – "*Статистические данные в графическом виде (2)*"

3. Далее – идёт масштабирование данных (см. *Рис.8,9*).

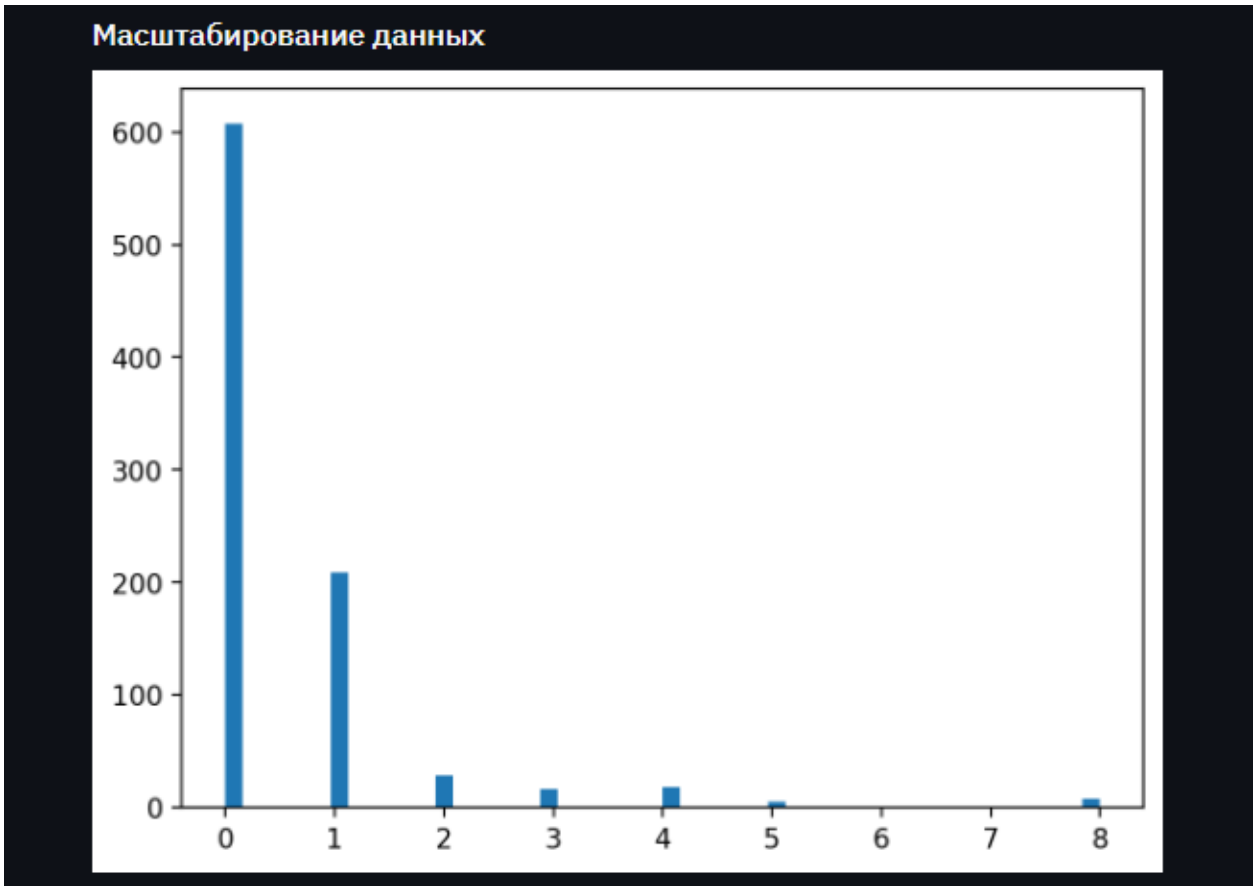

Рис.8 – "*Масштабирование данных (1)*"

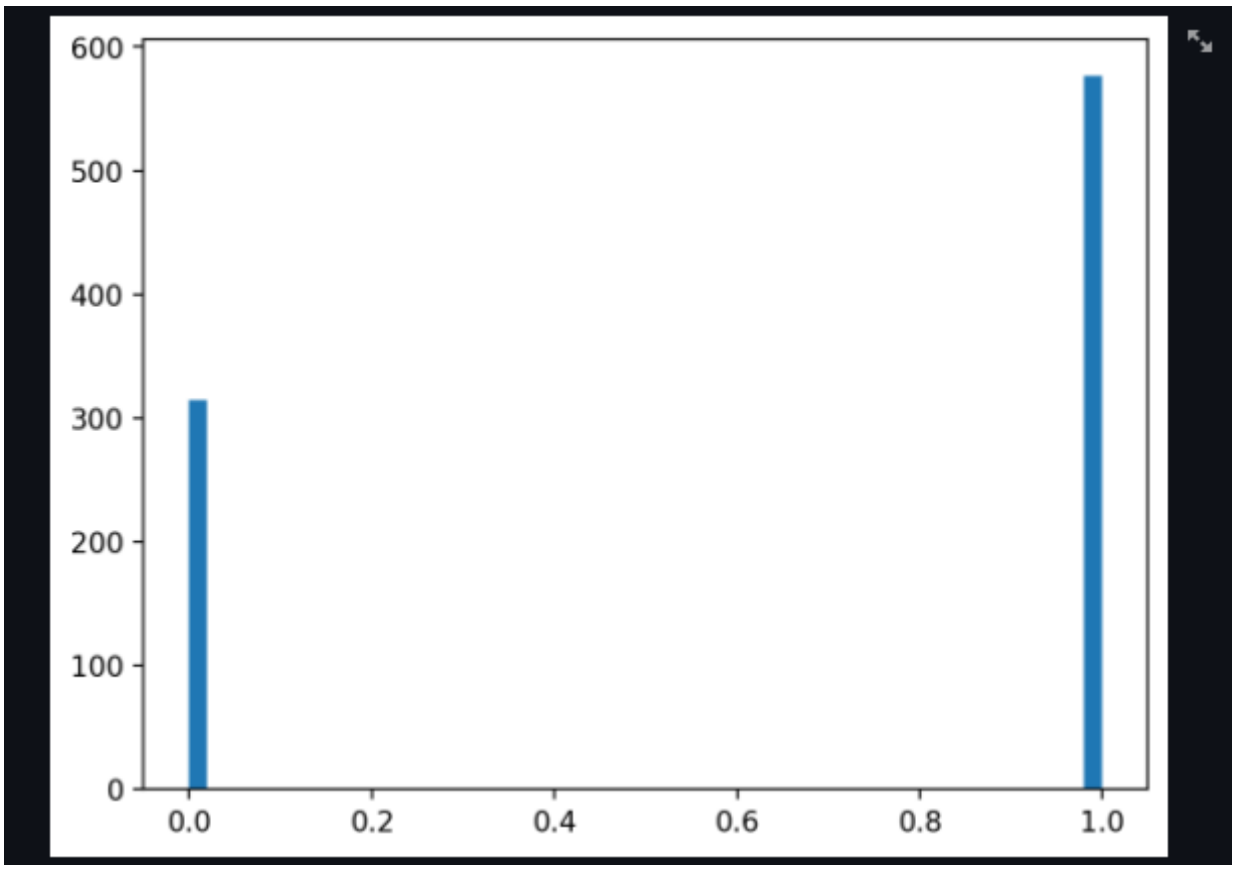

Рис.9 – "*Масштабирование данных (2)*"

4. Далее – Корреляционная матрица (см. *Рис.10*).

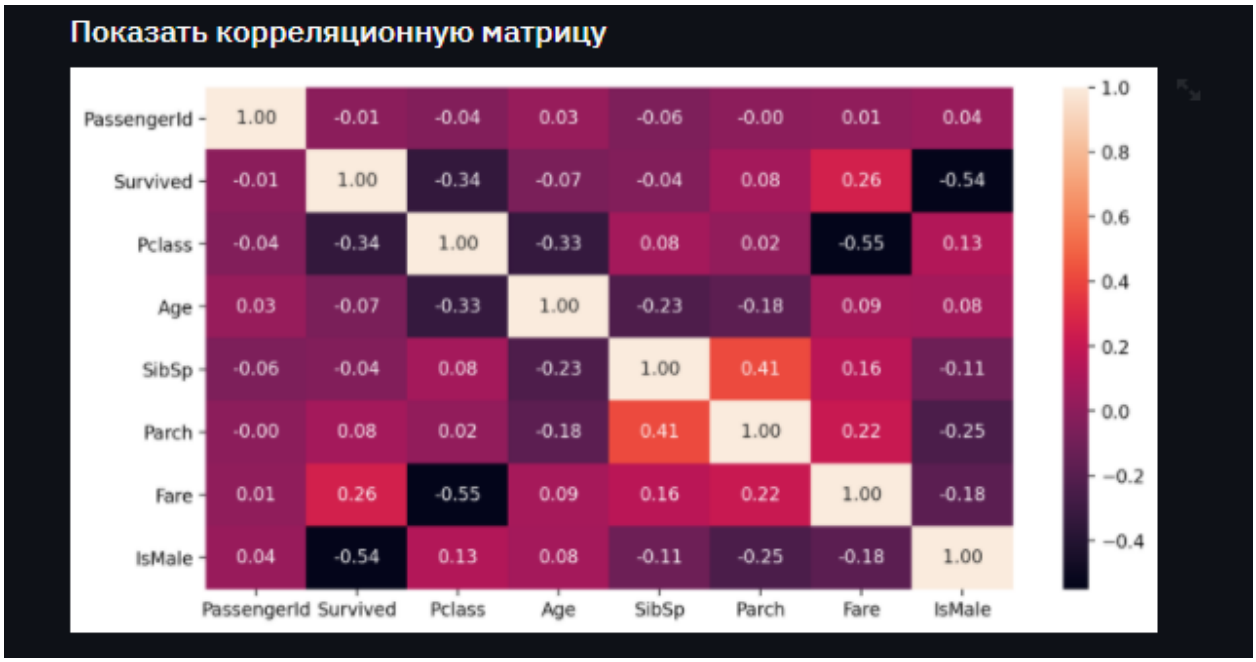

Рис.10 – "*Корреляционная матрица*"

5. Далее – идёт ансамблевая модель "Случайный лес" (см. Рис.11,12).

RandomForestRegressor

Средняя абсолютная ошибка:

0.18364435980275612

Средняя квадратичная ошибка:

0.1851745541442286

Median absolute error:

0.0

Коэффициент детерминации:

0.9062066791657145

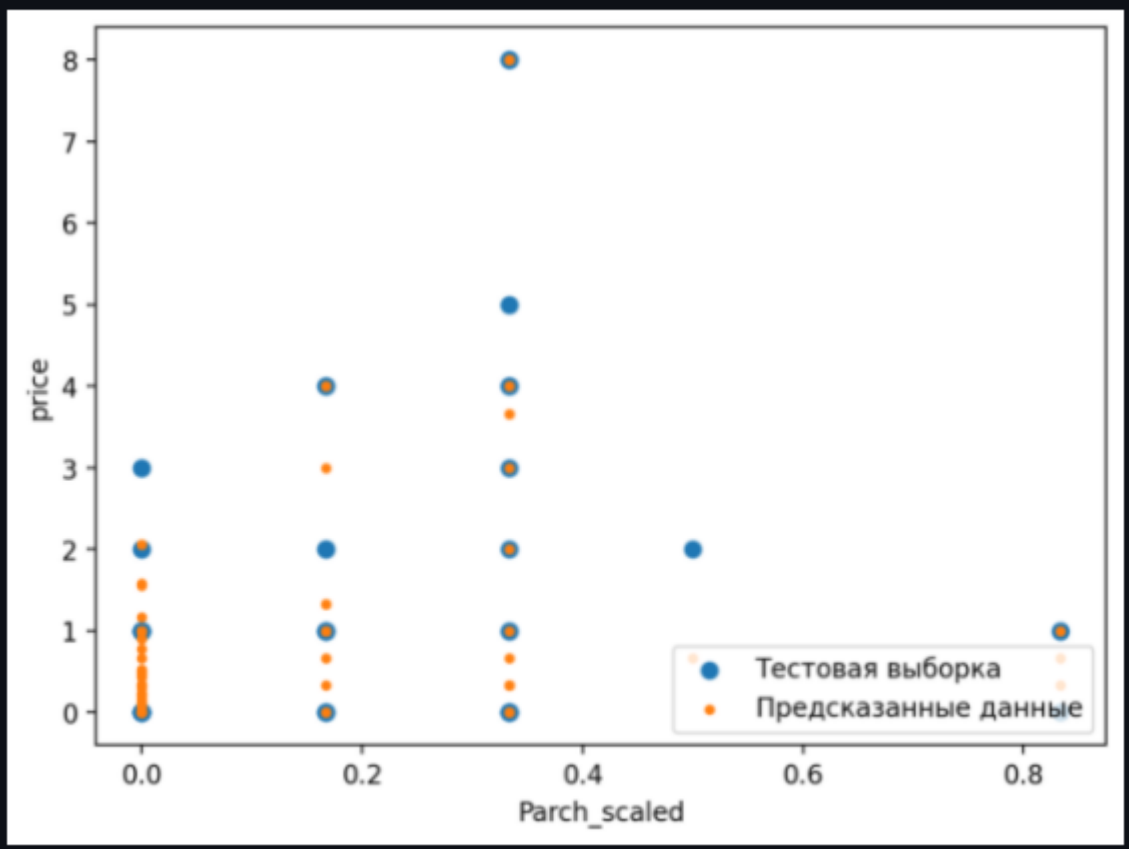

Рис.11 – "*Случайный лес: построение модели по "Ручным данным"*"

На рисунке 11 можно видеть модель, построенную по вручную настраиваемым данным, посредством "меню слева".

Нахождение лучшего случайного леса


```
{
  "n_estimators" : 8
}
```


Средняя абсолютная ошибка:


```
0.14940579010815708
```


Средняя квадратичная ошибка:


```
0.10652925835160501
```


Median absolute error:


```
0.0
```


Коэффициент детерминации:


```
0.946041544676661
```


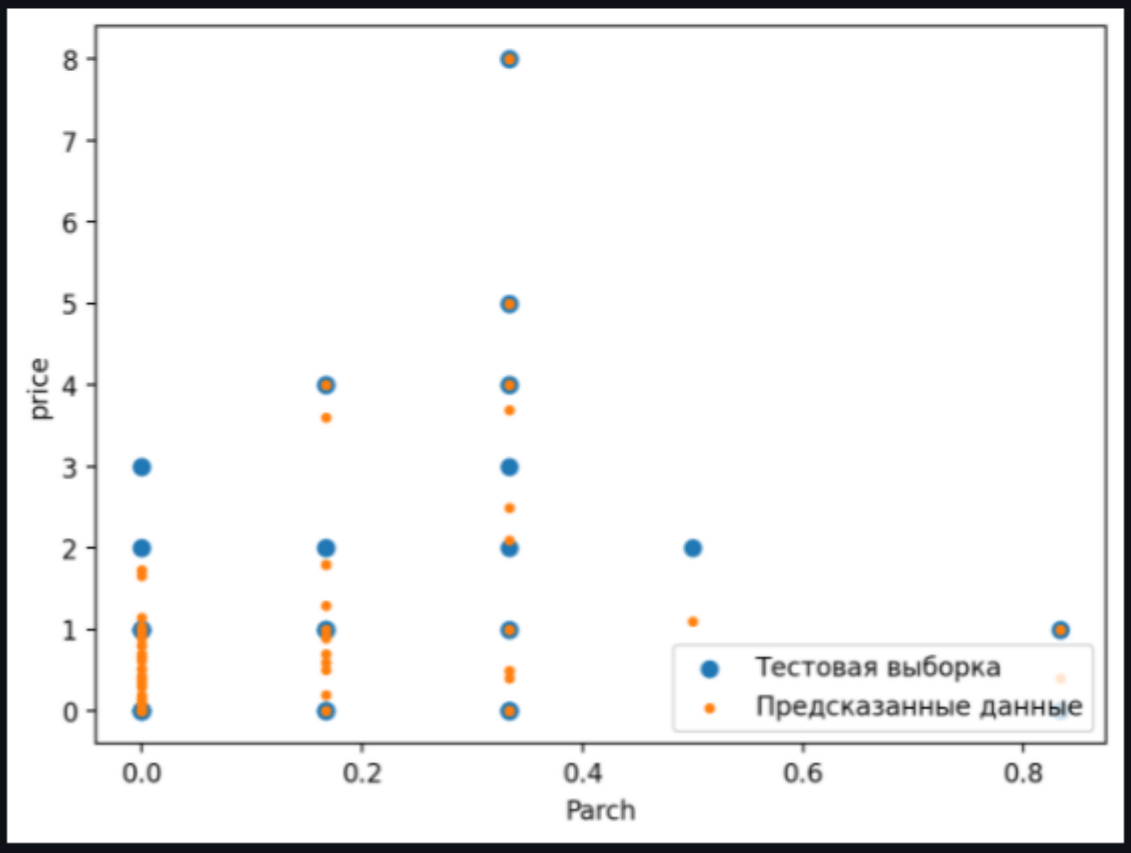

Рис.12 – "*Случайный лес: нахождение лучшей модели*"

На рисунке 12 можно наблюдать поиск лучших параметров для модели, что, в результате, можно заметить по увеличившемуся коэффициенту детерминации.

6. Далее – идёт ансамблевая модель "Градиентный бустинг" (см. Рис.13,14).

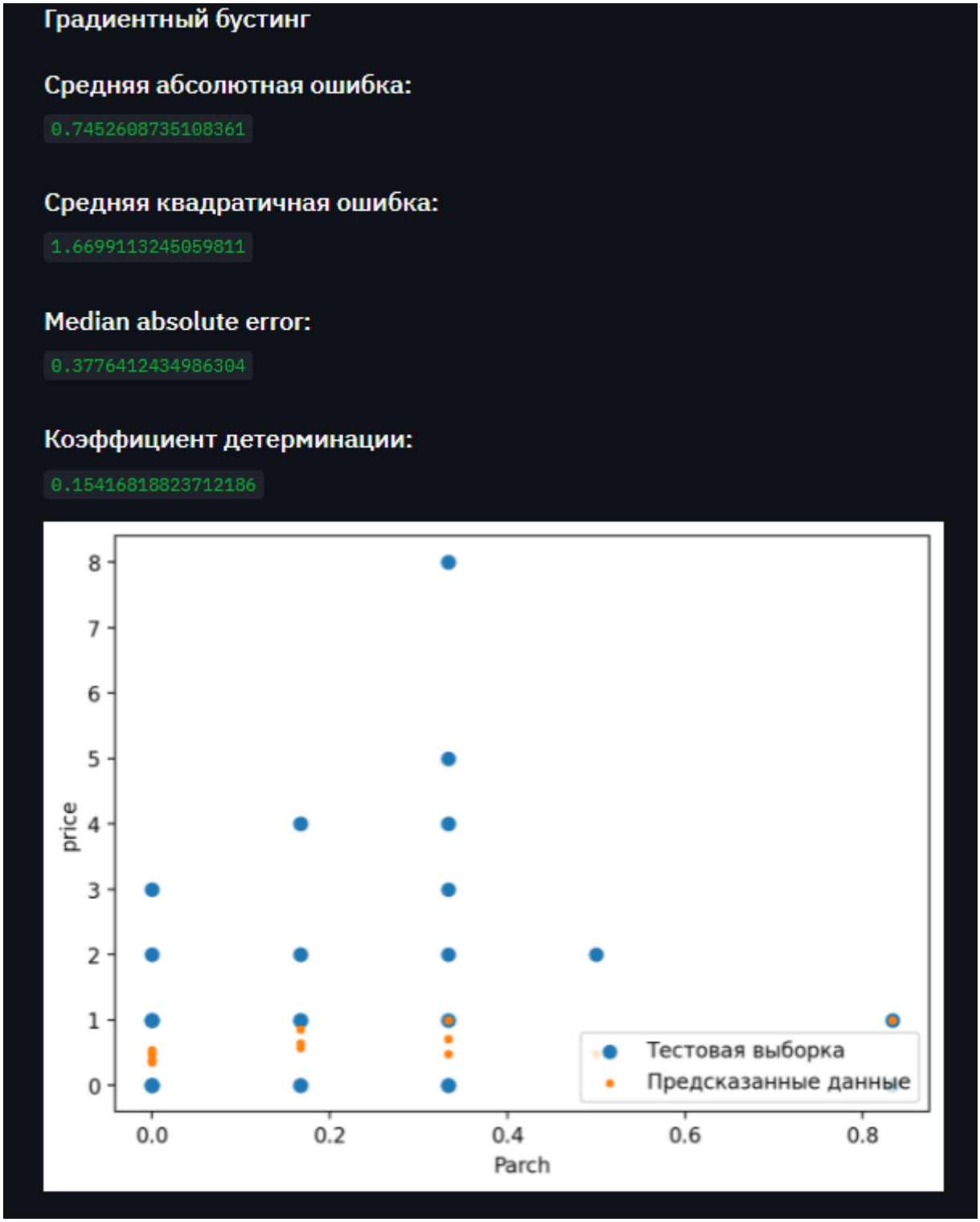

Рис.13 – *"Градиентный бустинг: построение модели по "Ручным данным""*

На рисунке 13 можно видеть модель, построенную по вручную настраиваемым данным, посредством "меню слева".

Нахождение лучшего////


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


Средняя абсолютная ошибка:

0.49086937429135014

Средняя квадратичная ошибка:

0.6555305844764013

Median absolute error:

0.1860326413267218

Коэффициент детерминации:

0.6679652303707297

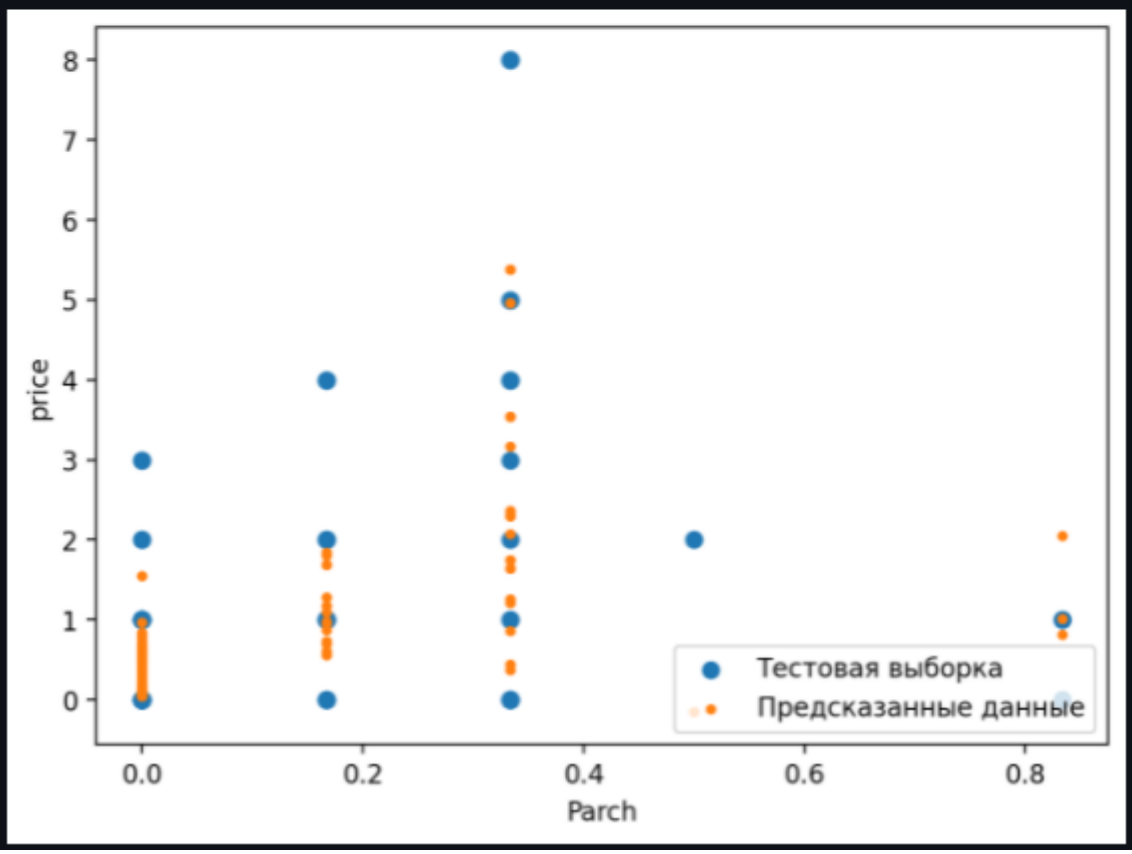

Рис.14 – "*Градиентный бустинг: нахождение лучшей модели*"

На рисунке 14 можно наблюдать поиск лучших параметров для модели, что, в результате, можно заметить по увеличившемуся коэффициенту детерминации.

7. Далее – идёт построение модели линейной регрессии (см. Рис.15).

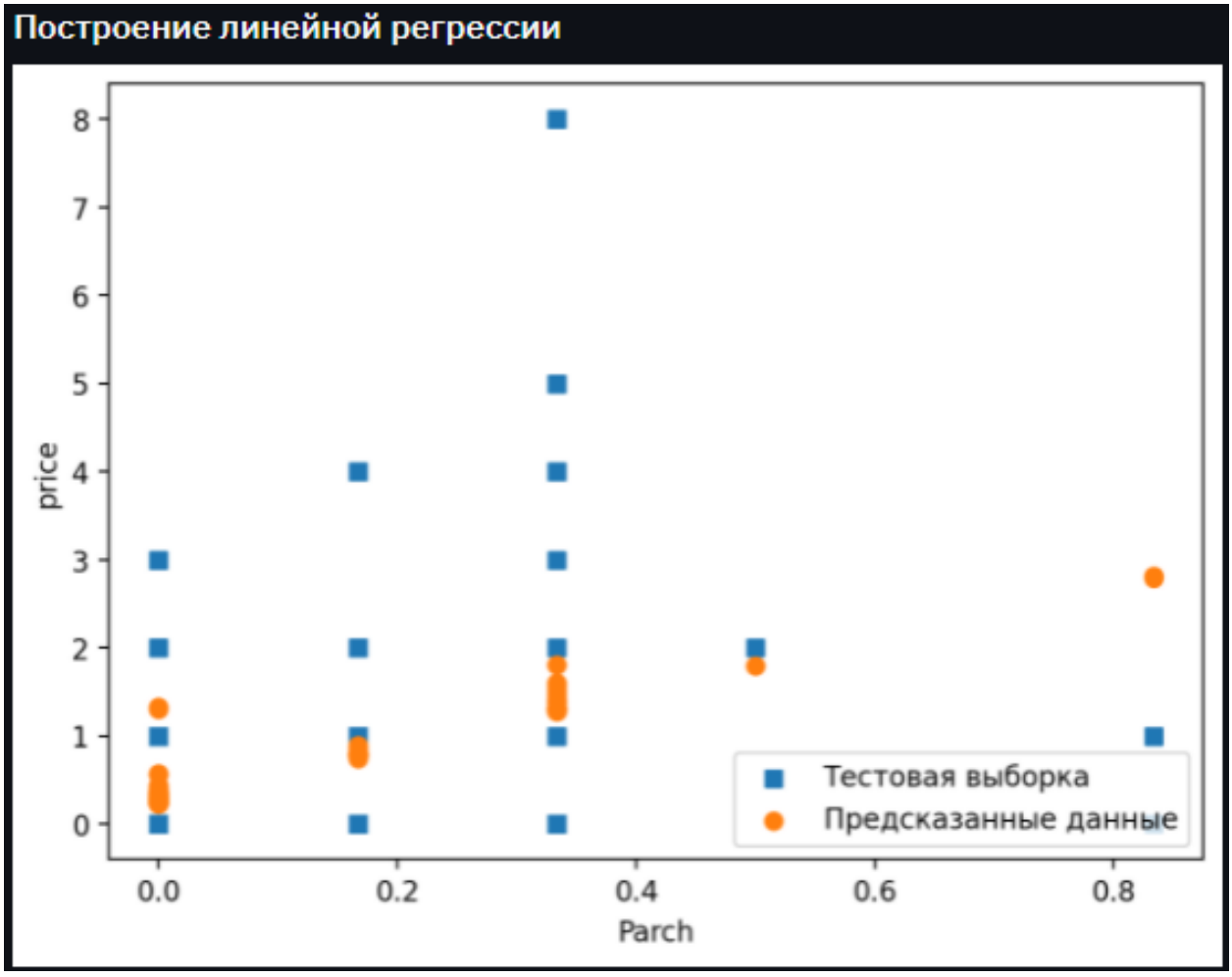

Рис.15 – "*Линейная регрессия*"

8. Далее – идёт построение модели дерева (см. Рис.16).

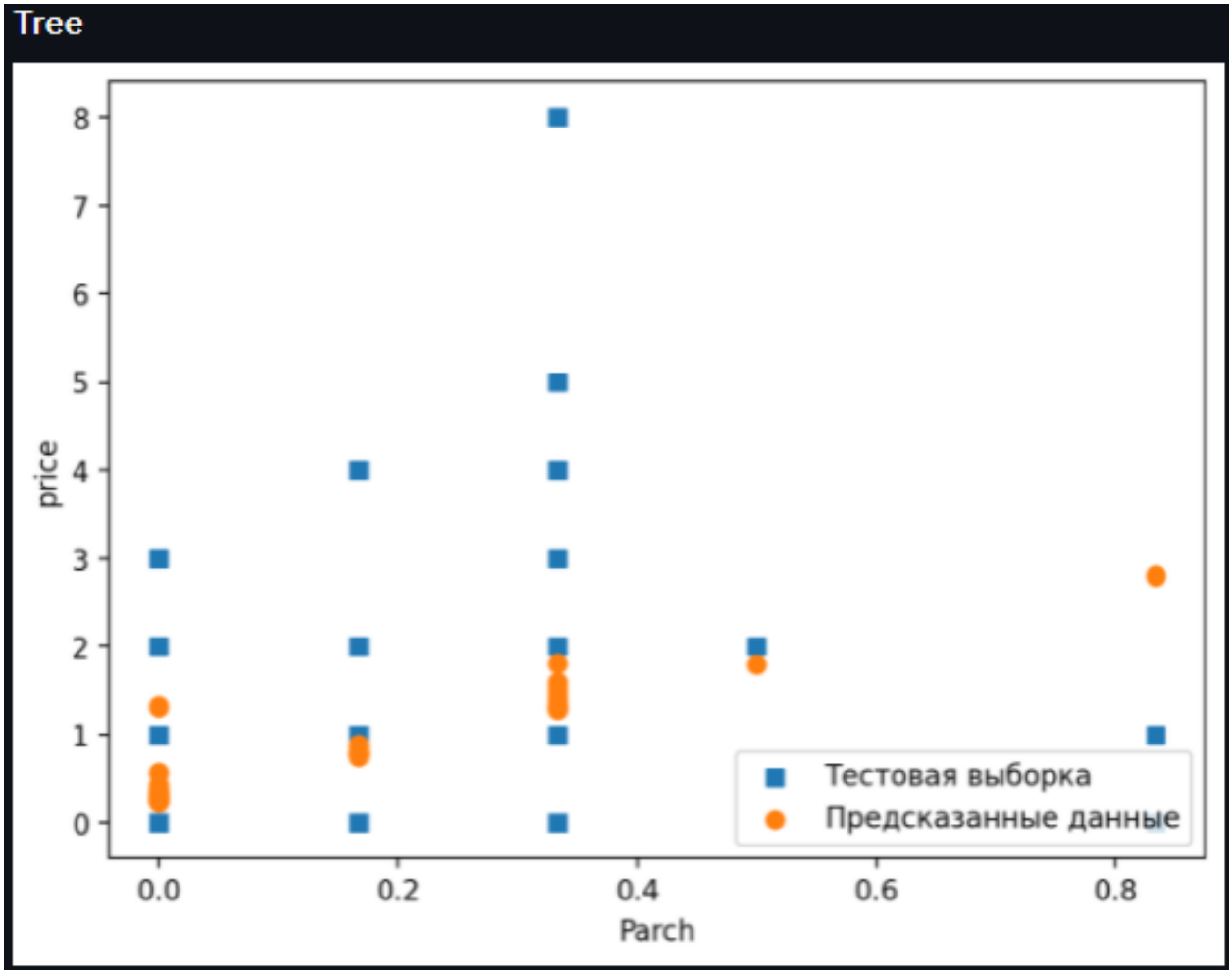

Рис.16 – "*Дерево*"

9. Далее – идёт построение модели ближайших соседей для произвольного гиперпараметра K (см. Рис.17).

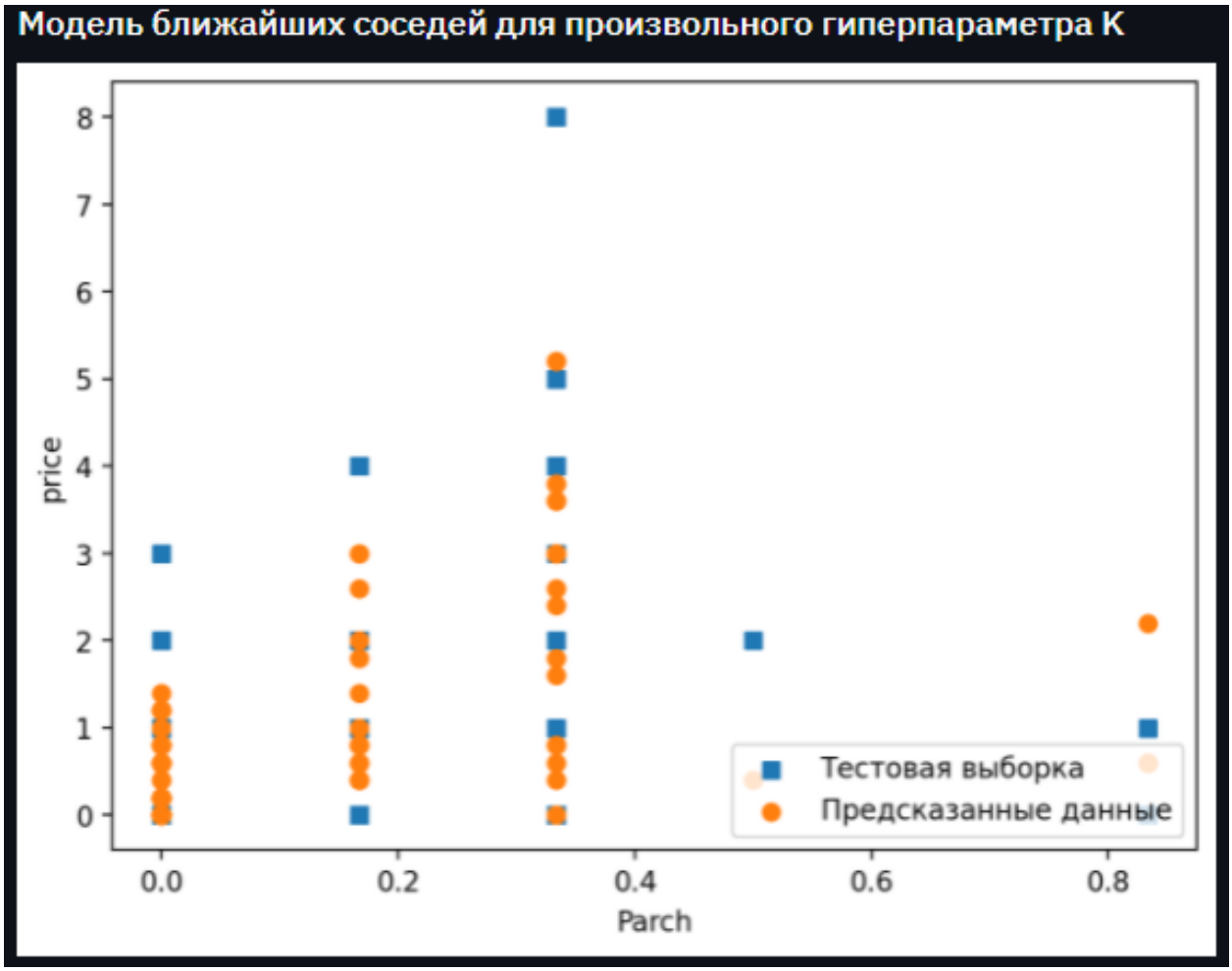

Рис.17 – *"Модель ближайших соседей для произвольного гиперпараметра"*

Код курсового проекта:

```
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, N
ormalizer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absol
ute_error, r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn import tree
import re

def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv('data/train.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 = ['Parch', 'Fare', 'IsMale']
    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['SibSp']
```

```
    # 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, tes
t_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=1
0, value=3, step=1)

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

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

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

st.write(data['Survived'].value_counts())
st.write(data['SibSp'].value_counts())

#  кодируем в 1/0
data['IsMale']=data.Sex.replace({'female':0,'male':1})
data.drop('Sex', axis = 1, inplace = True)
st.write(data.head())

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

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

f, ax = plt.subplots()
sns.violinplot(x=data['SibSp'])
st.pyplot(f)

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

f, ax = plt.subplots()
plt.hist(data['IsMale'], 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, ran
dom_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['Parch_scaled'], Y_test, marker='o', label='Тестовая выбо
рка')
```

```
plt.scatter(X_test['Parch_scaled'], Y_predict, marker='.', label='Предсказанные д
анные')
plt.legend(loc='lower right')
plt.xlabel('Parch_scaled')
plt.ylabel('price')
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_stat
e=5),
                      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=10, 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['Parch_scaled'], Y_test, marker='o', label='Тестовая выбо
рка')
plt.scatter(X_test['Parch_scaled'], Y_predict3, marker='.', label='Предсказанные
данные')
plt.legend(loc='lower right')
plt.xlabel('Parch')
plt.ylabel('price')
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['Parch_scaled'], Y_test, marker='o', label='Тестовая выбо
рка')
plt.scatter(X_test['Parch_scaled'], Y_grad_pred, marker='.', label='Предсказанные
 данные')
plt.legend(loc='lower right')
plt.xlabel('Parch')
plt.ylabel('price')
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=0.6, min_samples
_leaf=0.01, random_state=1)
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['Parch_scaled'], Y_test, marker='o', label='Тестовая выбо
рка')
plt.scatter(X_test['Parch_scaled'], Y_grad_pred1, marker='.', label='Предсказанны
е данные')
plt.legend(loc='lower right')
plt.xlabel('Parch')
plt.ylabel('price')
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['Parch_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['Parch_scaled'], lr_y_pred, marker='o', label='Предсказанные д
анные')
plt.legend(loc='lower right')
plt.xlabel('Parch')
plt.ylabel('price')
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['Parch_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['Parch_scaled'], lr_y_pred, marker='o', label='Предсказанные д
анные')
plt.legend(loc='lower right')
plt.xlabel('Parch')
plt.ylabel('price')
plt.show()
st.pyplot(fig5)

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

```
Regressor_5NN = KNeighborsRegressor(n_neighbors = 5)
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['Parch_scaled'], Y_test, marker='s', label='Тестовая выборка')
plt.scatter(X_test['Parch_scaled'], lr_y_pred, marker='o', label='Предсказанные д
анные')
plt.legend(loc='lower right')
plt.xlabel('Parch')
plt.ylabel('price')
plt.show()
st.pyplot(fig6)
```