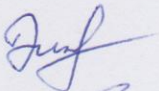



Министерство науки и высшего образования РФ
федеральное государственное автономное
образовательное учреждение высшего образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем
Кафедра «Прикладная математика и фундаментальная информатика»

Расчетно-графическая работа
по дисциплине **Машинное обучение и большие данные**

Тема: Разработка Web-приложения (дашборда)
для инференса моделей ML и анализа данных

Студента	Финк Эдуарда Валентиновича	
Курс	<u>2</u>	Группа <u>ФИТ-222</u>
Направление	<u>02.03.02 Фундаментальная информатика и информационные технологии</u>	
Руководитель	<u>ст. преп.</u> <u>Шарун И.В.</u>	
Выполнил	12.01.2024	
Проверил	12.01.2024	

Содержание

Введение	3
Основная часть	5
Заключение.....	10
Список использованных источников	11
Приложение А	12
Приложение Б	20

Введение

Тема машинного обучения (ML) является крайне актуальной и востребованной в современном мире. Она находит применение во многих областях, включая медицину, финансы, транспорт, маркетинг и другие. Развитие решений на основе ML позволяет автоматизировать процессы, улучшать прогнозирование, оптимизировать принятие решений и создавать инновационные продукты и услуги.

Для реализации задач ML используется стек технологий, включающий различные инструменты и компоненты. В данной расчётно-графической работе мы изучили такие инструменты, как Sklearn, TensorFlow, matplotlib и pandas. Каждая из этих библиотек выполняет свою роль в ML.

1) Scikit-learn (Sklearn): Это одна из наиболее популярных библиотек машинного обучения для языка Python. Она предоставляет широкий спектр алгоритмов машинного обучения, обработки данных, предобработки, валидации моделей и метрик для оценки результатов. Sklearn обладает простым и интуитивно понятным API, и его можно использовать для классификации, регрессии, кластеризации и других задач.

2) TensorFlow: Это открытая платформа глубокого обучения, которая позволяет строить и обучать нейронные сети. TensorFlow предлагает гибкое программное средство для разработки ML-моделей на разных уровнях сложности, включая создание нейронных сетей глубокого обучения. Он также обеспечивает эффективное использование аппаратных ресурсов и развертывание моделей на различных платформах.

3) Matplotlib: Это библиотека визуализации данных для языка Python. Она позволяет создавать простые и сложные графики, включая гистограммы, диаграммы рассеяния, линейные и логарифмические графики и др. Matplotlib играет важную роль в анализе и визуализации результатов ML-моделей, представлении данных и понимании моделирования.

4) Pandas: Это библиотека для обработки и анализа данных в Python. Она предоставляет удобные структуры данных, такие как DataFrames, для

эффективной работы с табличными данными. pandas позволяет проводить предобработку данных, выполнение операций фильтрации, агрегации и трансформации, а также интеграцию данных с моделями машинного обучения.

Основываясь на полученных знаниях и использованных инструментах, мы разработали web-приложение (дашборд) с использованием библиотеки Streamlit. Главная цель приложения состоит в том, чтобы представить модели машинного обучения и результаты анализа данных с возможностью их визуализации в различных форматах. Это позволяет пользователям легко взаимодействовать с моделями и получать наглядные представления результатов анализа данных.

Streamlit: Это библиотека для создания интерактивных веб-приложений для машинного обучения и анализа данных на языке Python. Streamlit позволяет разработчикам быстро и просто создавать красивые дашборды, включая визуализацию данных, интерактивные элементы управления и интеграцию моделей машинного обучения. Он может быть полезным инструментом для демонстрации результатов и предоставления доступа к моделям в режиме реального времени.

Основная часть

1) Описание выбранных моделей.

1.1) Простая линейная регрессия: Простая линейная регрессия - это метод моделирования, использующий линейную функцию для предсказания зависимой переменной на основе одной независимой переменной. Она строит линейную связь между независимой и зависимой переменными, представляя это как уравнение прямой линии (вида $y = mx + b$), где m - это наклон линии, а b - это смещение.

1.2) Множественная линейная регрессия: Множественная линейная регрессия расширяет простую линейную регрессию на случай, когда зависимость предсказываемой переменной от нескольких независимых переменных. В данной модели мы строим линейную комбинацию нескольких независимых переменных, чтобы предсказывать значение зависимой переменной.

1.3) Бэггинг (Bagging): Бэггинг - это метод ансамблевого обучения, в котором создается несколько независимых моделей на основе подвыборки обучающих данных. Затем, для получения окончательного прогноза, результаты этих моделей усредняются (в случае регрессии) или используется голосование (в случае классификации). Бэггинг помогает уменьшить дисперсию модели и повысить её стабильность.

1.4) Градиентный бустинг (Gradient Boosting): Градиентный бустинг - это метод ансамблевого обучения, который обучает последовательность слабых моделей (например, деревьев решений) в итеративном режиме. Каждая новая модель приближает остатки предыдущей модели. Затем результаты этих моделей суммируются для получения окончательного прогноза. Градиентный бустинг является мощным методом, позволяющим получить лучшую модель, чем только одиночная модель.

1.5) Стекинг (Stacking): Стекинг - это метод ансамблевого обучения, который комбинирует результаты нескольких базовых моделей, обученных на одних и тех же данных. Базовые модели предоставляют свои предсказания, а

затем метамодель используется для объединения этих предсказаний и получения окончательного прогноза.

1.6) Глубокая полносвязная нейронная сеть: Глубокая полносвязная нейронная сеть - это модель машинного обучения, состоящая из нескольких слоев нейронов, где каждый нейрон из предыдущего слоя соединен с каждым нейроном в следующем слое. Эта модель может иметь множество слоев и обучается на больших объемах данных. Глубокая полносвязная нейронная сеть широко используется для выполнения сложных задач, таких как распознавание образов и обработка естественного языка.

2) Процесс сериализации моделей.

2.1) Обученные модели простой линейной регрессии, множественной регрессии, бэггинга, бустинга и стэкинга были сериализованы с использованием библиотеки Joblib. Модели сохраняются в файлы типа .pkl и могут быть восстановлены для дальнейшей работы.

2.2) Обученная глубокая нейронная сеть была сериализована с использованием инструментов, предоставленных библиотекой TensorFlow. Модель сохраняется в файл типа .h5 и может быть восстановлена для дальнейшего использования.

3) Процесс разработки дашборда.

3.1) Web-страница 1: Информация о разработчике. На рисунке 1 приложения А показано как выглядит разработанная web-страничка. Ниже представлен код реализации данной web-странички

```
def show_author_page():  
    st.title("Разработка Web-приложения (дашборда) для инференса (вывода) моделей ML и анализа данных")  
  
    st.header("Автор")  
    st.write("ФИО: Финк Эдуард Валентинович")  
    st.write("Группа: ФИТ-222")  
    image = "Project/This is for RGR.jpg"  
    st.image(image, caption='Моё фото', use_column_width=True)
```

3.2) Web-страница 2: Информация о наборе данных. На рисунках 2-4 приложения А показано как выглядит разработанная web-страничка. Ниже

представлена часть кода реализации данной web-странички

```
def show_dataset_page():
    st.title("Информация о наборе данных")

    st.header("Тематика набора данных")
    st.write("Этот классический набор данных содержит цены и другие атрибуты п

    st.header("Описание признаков")
    st.write("price: цена в долларах США (\$326--\$18,823)")
    st.write("carat: вес бриллианта (0.2--5.01)")
    st.write("cut: качество среза (Fair, Good, Very Good, Premium, Ideal)")
    st.write("color: цвет бриллианта, от J (worst) до D (best)")
```

3.3) Web-страница 3: Визуализация зависимостей в наборе данных. На рисунках 5-8 приложения А показано как выглядит разработанная web-страничка. Ниже представлена часть кода реализации данной web-странички

```
def show_visualizations_page():
    st.title("Визуализации данных")

    data = pd.read_csv("Project/diamonds_redux.csv")
    st.subheader("Примеры визуализаций:")

    st.write("Столбчатая диаграмма:")
    plt.figure(figsize=(8, 6))
    sns.countplot(x="color", data=data)
    st.pyplot(plt)

    st.write("Ящик с усами:")
    plt.figure(figsize=(8, 6))
    sns.boxplot(x="color", y="price", data=data)
    st.pyplot(plt)
```

3.4) Web-страница 4: Предсказания моделей на наборе данных. На рисунках 9-10 приложения А показано как выглядит разработанная web-

страничка. Ниже представлена часть кода реализации данной web-странички

```
uploaded_file = st.file_uploader("Загрузите файл данных (CSV)", type="csv")

if uploaded_file:
    df = pd.read_csv(uploaded_file)
    df.drop(df.filter(regex="Unnamed"),axis=1, inplace=True)
    st.write("Полученный набор данных:", df)

    models_mapping = {
        'LinearRegression(множественная)': model1,
        'LinearRegression(простая)': model2,
        'bagging': model3,
        'boosting': model4,
        'stacking': model5,
        'NeuralNetwork': model6
    }
    cut_model = st.selectbox("Выберите тип модели", list(models_mapping.keys()))
    model = models_mapping[cut_model]
    y = df["price"]
    X = df
    X.drop(['price'], axis=1, inplace=True)
```

3.5) Web-страница 5: Предсказания моделей на введённых пользователем данных. На рисунках 11-14 приложения А показано как выглядит разработанная web-страничка. Ниже представлена часть кода реализации данной web-странички

```
clarity_text = st.selectbox("Clarity", list(clarity_mapping.keys()))
clarity = clarity_mapping[clarity_text]
x = st.slider("Length (mm)", 0.0, 10.74, 5.0)
y = st.slider("Width (mm)", 0.0, 58.9, 3.0)
z = st.slider("Depth (mm)", 0.0, 31.8, 2.0)
depth = st.slider("Depth Percentage (%)", 43, 79, 60)
table = st.slider("Table Width (%)", 43, 95, 60)

features = np.array([[carat, cut, color, clarity, x, y, z, depth, table]])
features1 = np.array(carat).reshape(1, -1)
price_prediction1 = model1.predict(features)
price_prediction2 = model2.predict(features1)
price_prediction3 = model3.predict(features)
price_prediction4 = model4.predict(features)
price_prediction5 = model5.predict(features)
price_prediction6 = model6.predict(features)
```


3.6) GitHub-репозиторий и размещение исходных файлов. Все файлы размещены в открытом репозитории GitHub. Подробное описание проекта находится в файле README. Ссылка на репозиторий: https://github.com/Sovanno/RGR_ML

3.7) Веб-сервис, разработанный с использованием библиотеки Streamlit был размещён на Streamlit Cloud и доступен по следующей ссылке: <https://rgrmlfink.streamlit.app/>

Заключение

В рамках данной работы было создано web-приложение (dashboard) с использованием библиотеки Streamlit. Основная цель приложения заключалась в выводе результатов инференса моделей машинного обучения и их визуализации.

Был выбран один из двух наборов данных, соответствующих заданию. Среди построенных моделей машинного обучения было выбрано шесть лучших моделей на основе значения коэффициента детерминации для регрессии или F1-меры для классификации.

Выбранные модели были сериализованы и сохранены на жесткий диск, используя соответствующие инструменты (TensorFlow, Joblib).

С помощью библиотеки Streamlit было реализовано web-приложение (dashboard), содержащее несколько страниц:

- 1) Страница с информацией о разработчике моделей машинного обучения.
- 2) Страница с информацией о выбранном наборе данных.
- 3) Страница с визуализациями зависимостей в наборе данных, используя библиотеки Matplotlib и Seaborn.
- 4) Страница, позволяющая получить предсказание соответствующей модели машинного обучения для набора данных.
- 5) Страница, позволяющая получить предсказание соответствующей модели машинного обучения для данных вводимых пользователем.

Список использованных источников

- 1) user guide / [Электронный ресурс] // scikit-learn : [сайт]. — URL: https://scikit-learn.ru/user_guide/ (дата обращения: 10.12.2023).
- 2) API Documentation / [Электронный ресурс] // TensorFlow : [сайт]. — URL: https://www.tensorflow.org/api_docs (дата обращения: 10.12.2023).
- 3) Streamlit documentation / [Электронный ресурс] // Streamlit : [сайт]. — URL: <https://docs.streamlit.io> (дата обращения: 20.12.2023).

Приложение А

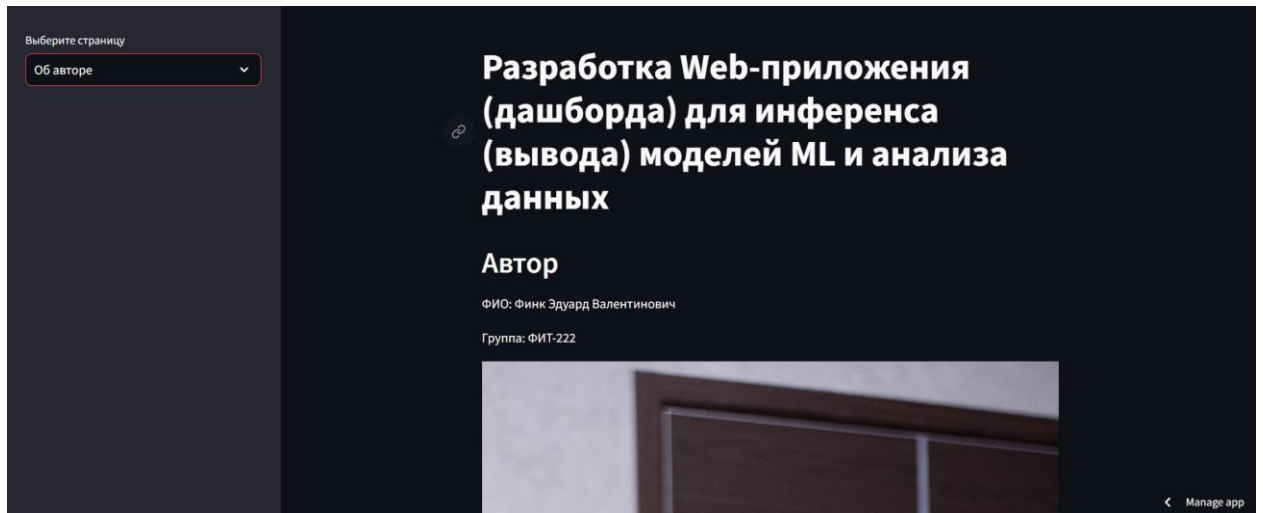


Рисунок 1 – Информация о разработчике моделей ML

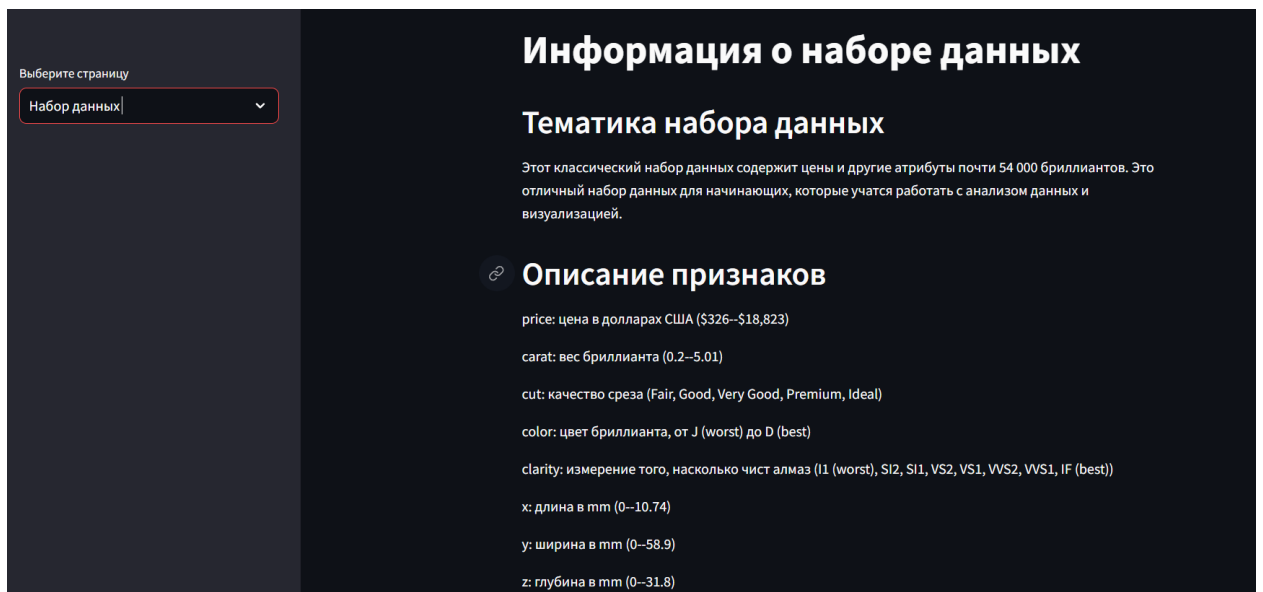


Рисунок 2 – Информация о наборе данных

Выберите страницу

Набор данных

depth: общая глубина в процентах = $z / \text{mean}(x, y) = 2 * z / (x + y)$ (43--79)

table: ширина вершины ромба относительно самой широкой точки (43--95)

Особенности предобработки данных

В данном наборе данных нужно было предугадывать цену бриллианта в долларах США. Сама цена находится в столбике price.

В наборе данных присутствовал столбик 'Unpamed: 0', который не является нужным, он был удалён.

Также в датасете присутствовали дубликаты, которые в последствии были удалены.

Столбики cut, color, clarity были представлены в виде чисел.

Для cut замена выглядит так:

Ideal: 5

Premium: 4

Very Good: 3

Good: 2

Fair: 1

Рисунок 3 – Информация о наборе данных

Выберите страницу

Набор данных

F: 5

G: 4

H: 3

I: 2

J: 1

Для clarity замена выглядит так:

IF: 8

VS1: 7

VS2: 6

VSI: 5

VS2: 4

S11: 3

SI2: 2

I1: 1

Был проведен EDA и удалены выбросы.

Рисунок 4 – Информация о наборе данных

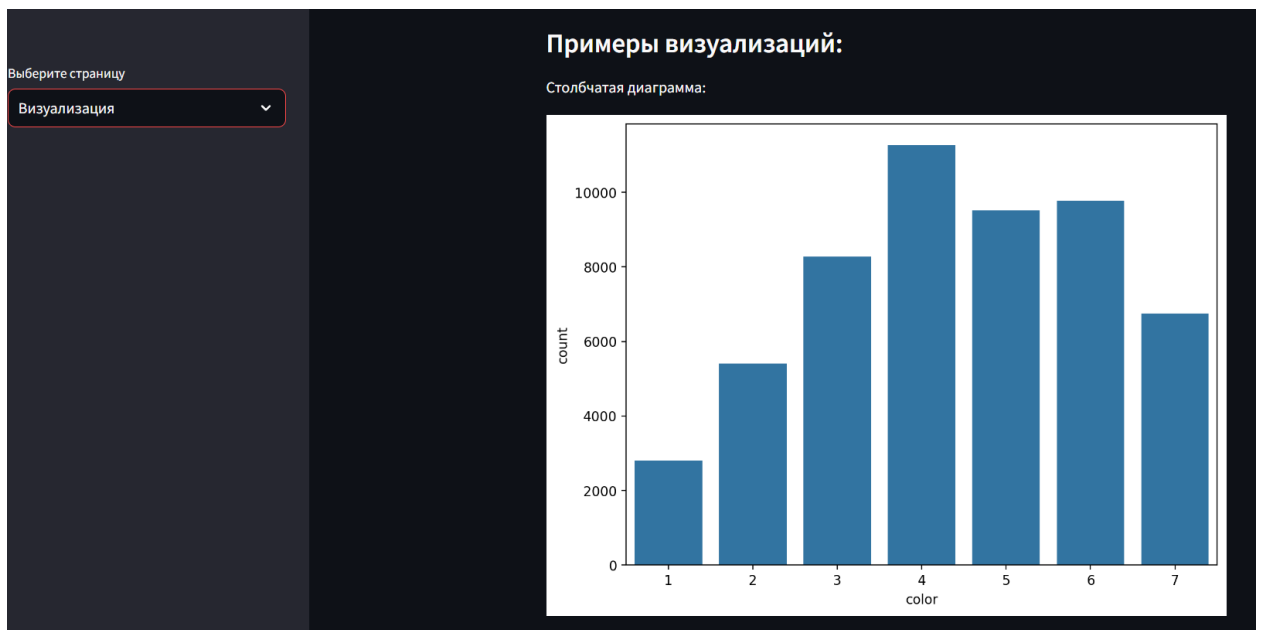


Рисунок 5 – Визуализация зависимостей в наборе данных

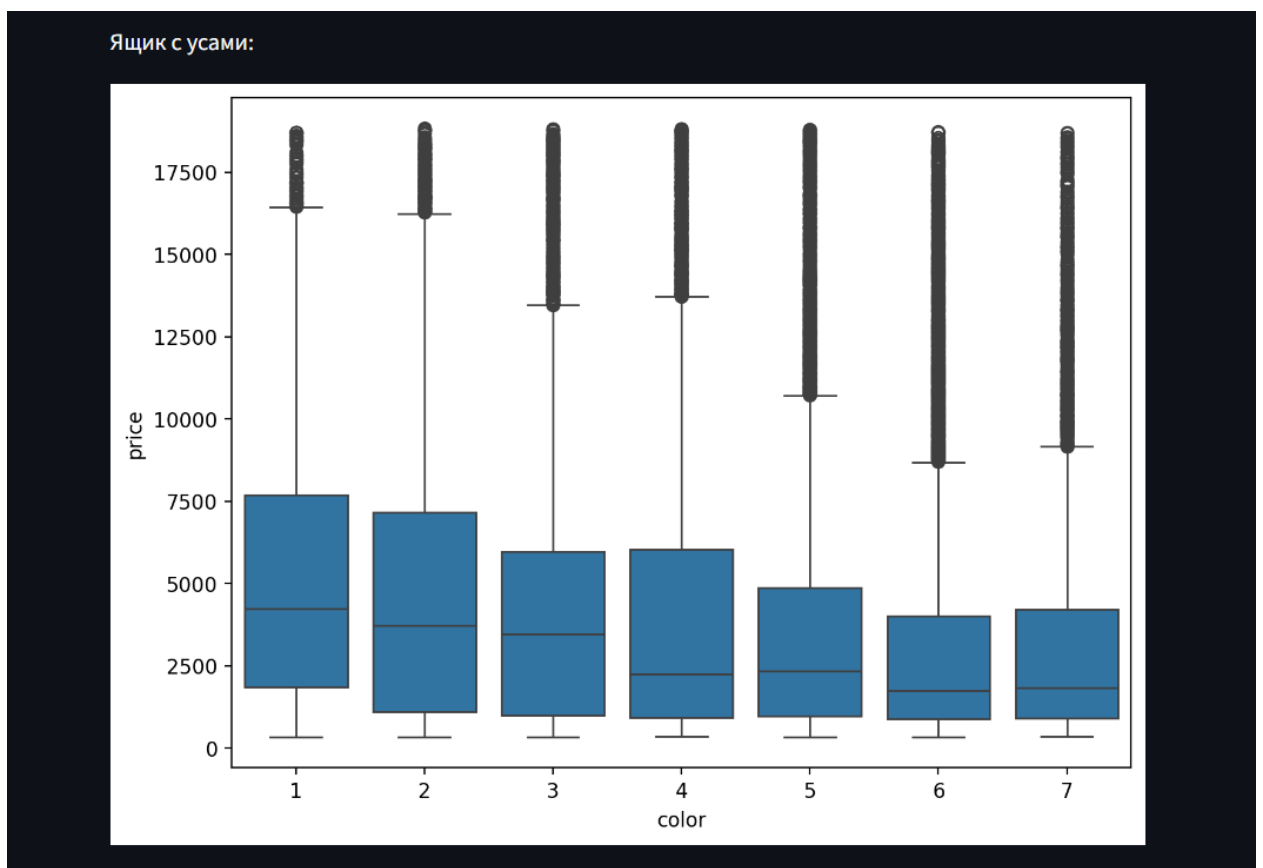


Рисунок 6 – Визуализация зависимостей в наборе данных

Тепловая карта:

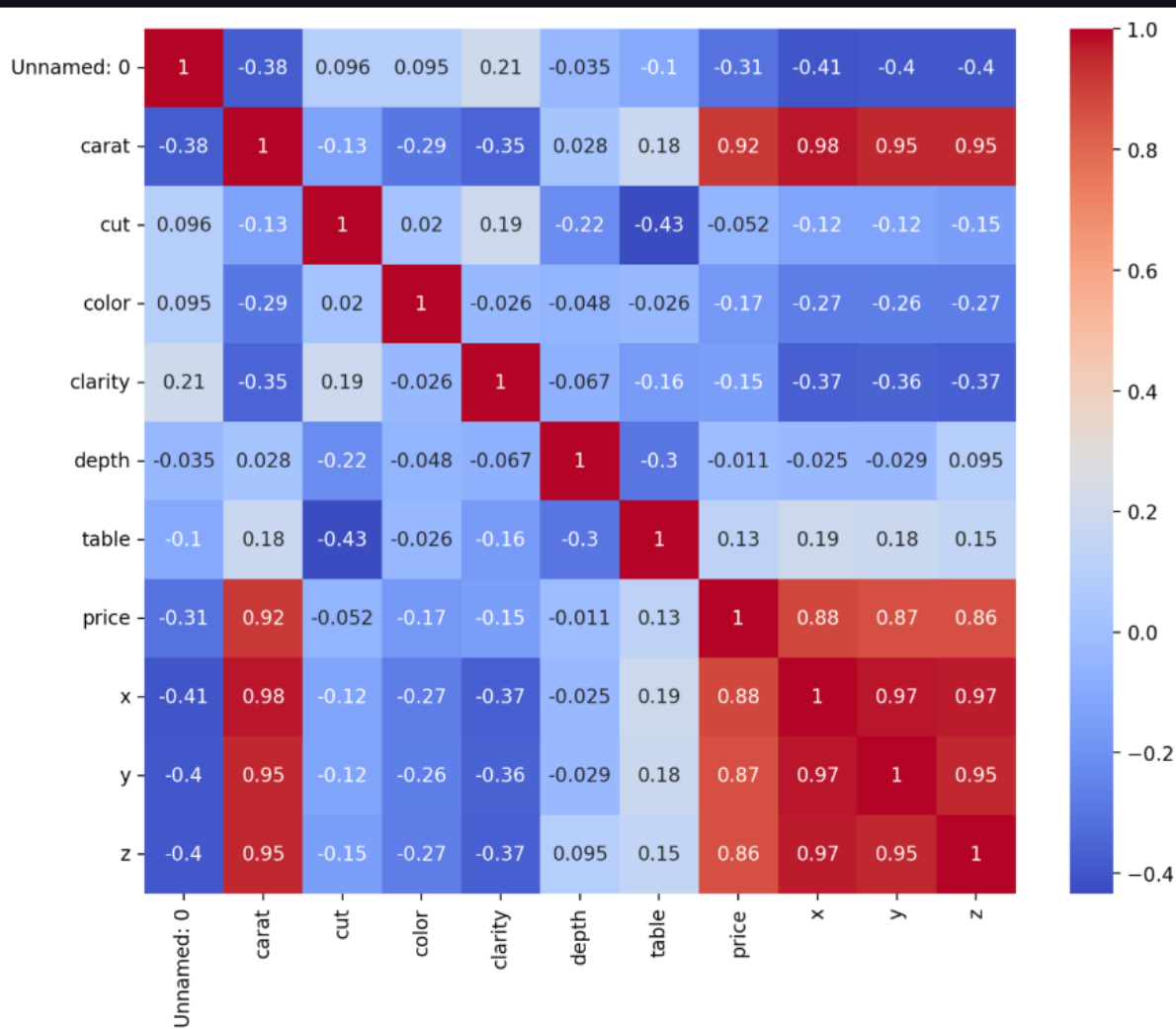


Рисунок 7 – Визуализация зависимостей в наборе данных

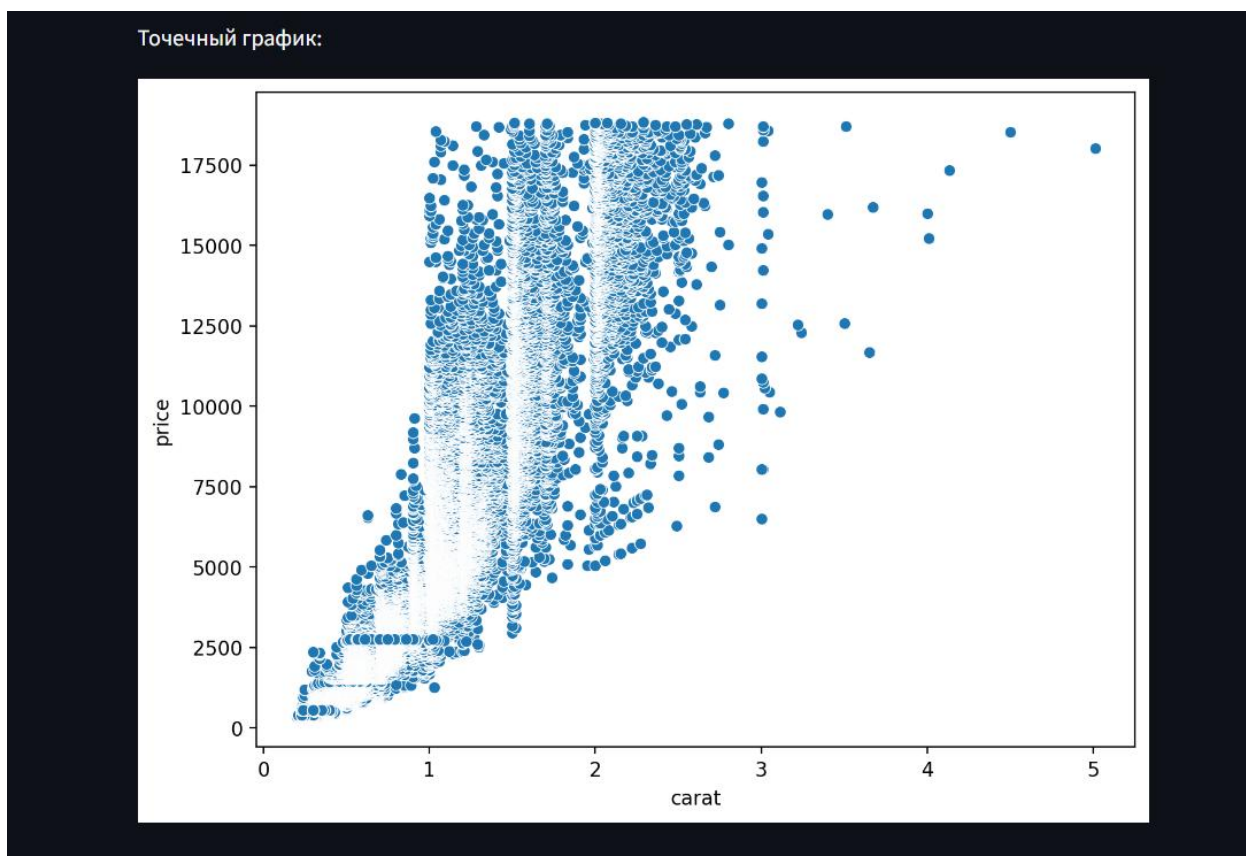


Рисунок 8 – Визуализация зависимостей в наборе данных

Выберите страницу

Предсказания для набора дан...

Загрузите файл данных (CSV)

Drag and drop file here
Limit 200MB per file • CSV

Browse files

diamonds_redux.csv 2.4MB

Полученный набор данных:

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	5	6	2	61.5	55	326	3.95	3.98	2.43
1	0.21	4	6	3	59.8	61	326	3.89	3.84	2.31
2	0.23	2	6	5	56.9	65	327	4.05	4.07	2.31
3	0.29	4	2	4	62.4	58	334	4.2	4.23	2.63
4	0.31	2	1	2	63.3	58	335	4.34	4.35	2.75
5	0.24	3	1	6	62.8	57	336	3.94	3.96	2.48
6	0.24	3	2	7	62.3	57	336	3.95	3.98	2.47
7	0.26	3	3	3	61.9	55	337	4.07	4.11	2.53
8	0.22	1	6	4	65.1	61	337	3.87	3.78	2.49
9	0.23	3	3	5	59.4	61	338	4	4.05	2.39

Рисунок 9 – Предсказания моделей на наборе данных

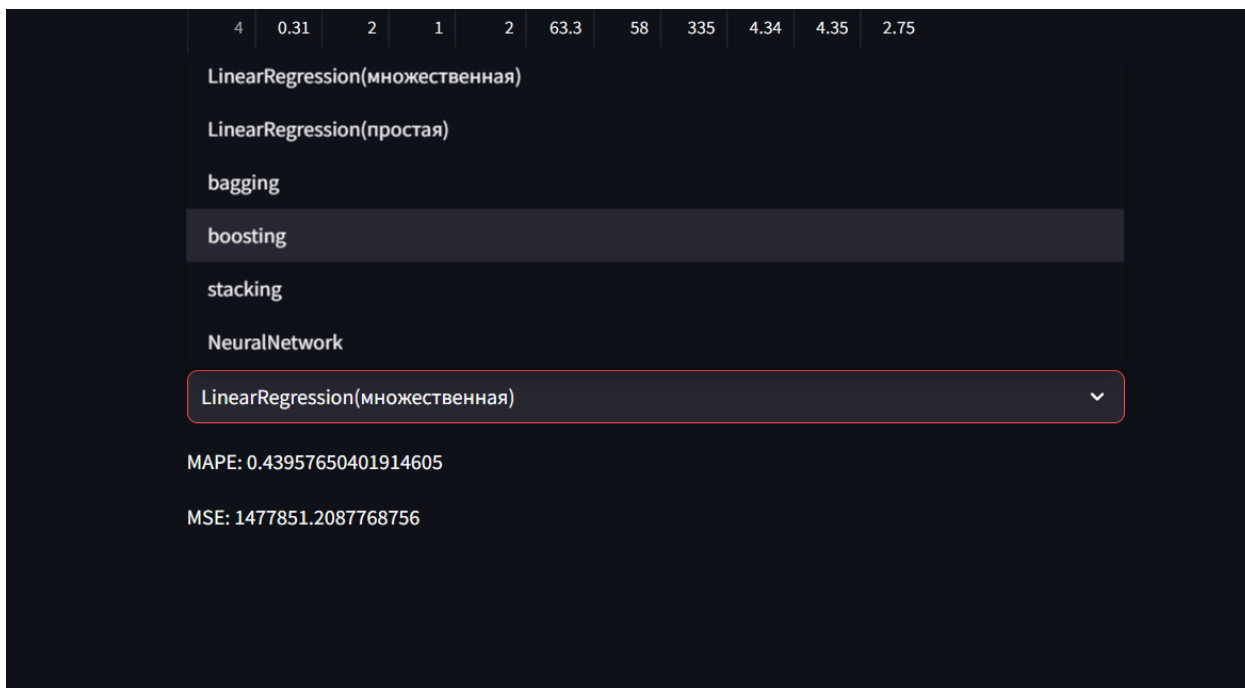


Рисунок 10 – Предсказания моделей на наборе данных

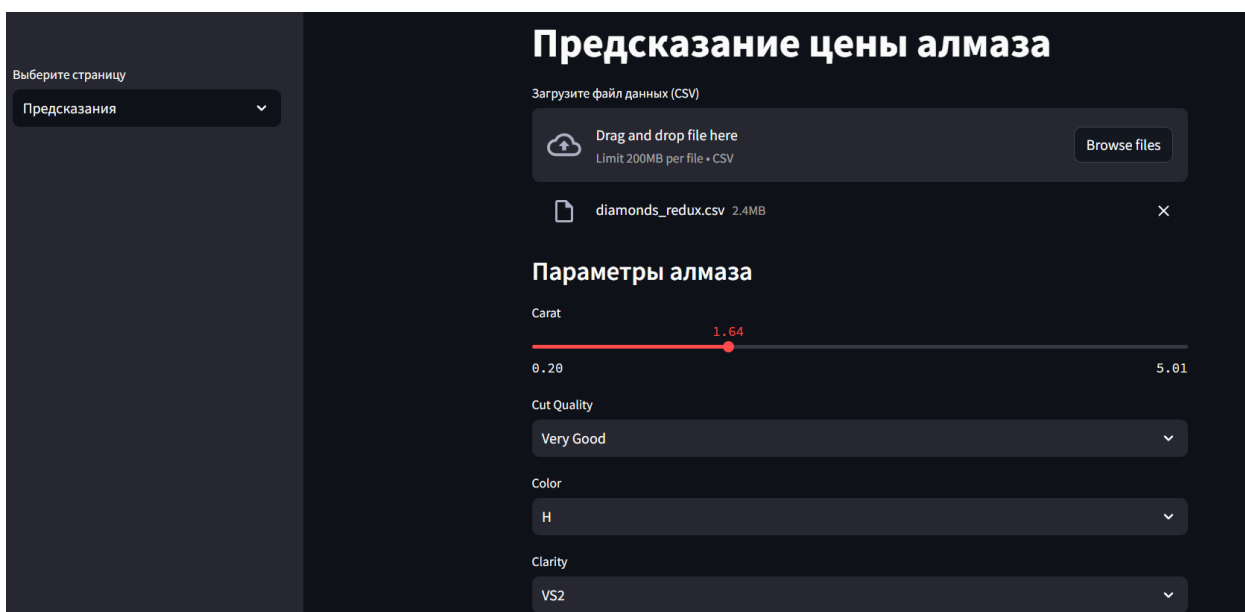


Рисунок 11 – Предсказания моделей на введённых пользователем данных

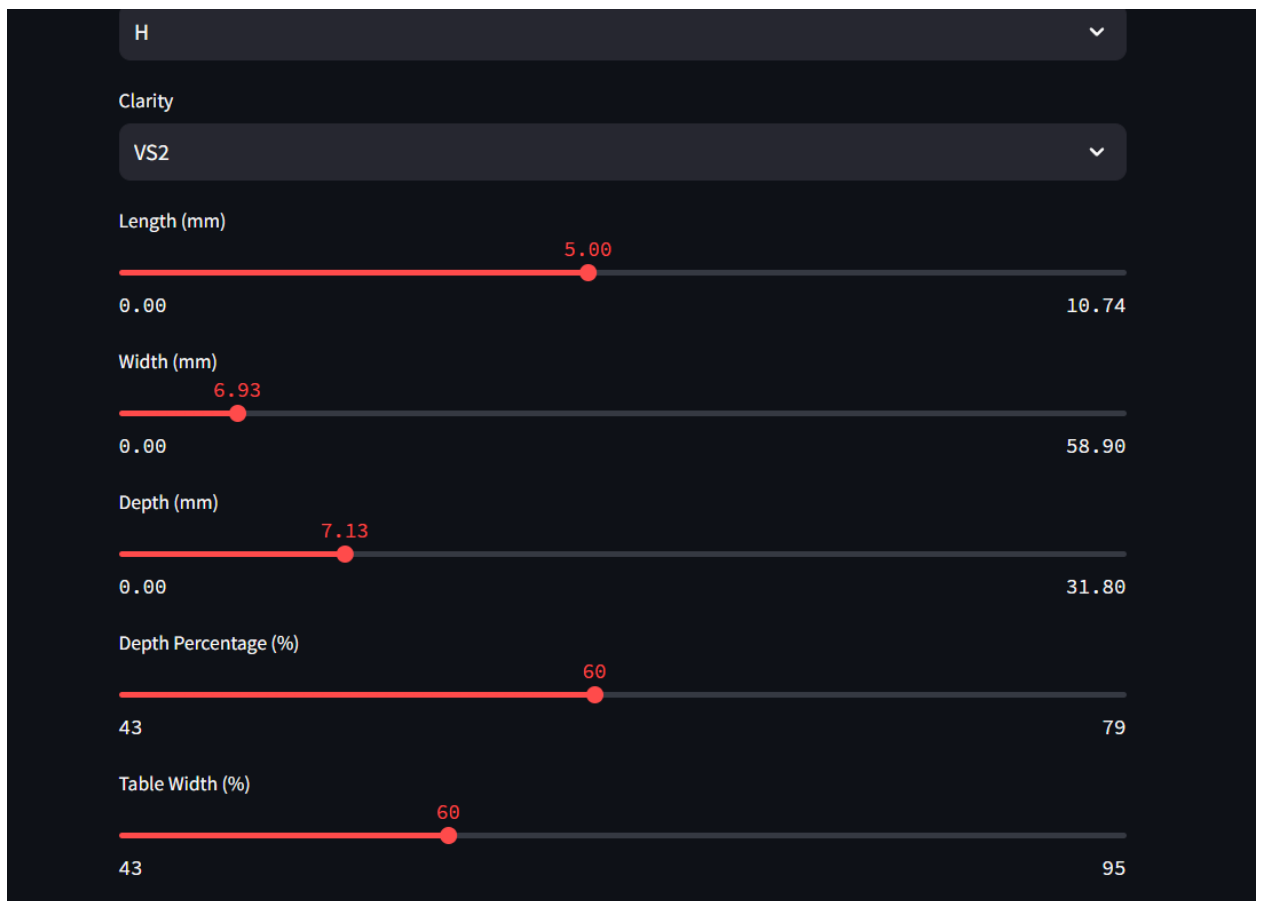


Рисунок 12 – Предсказания моделей на введённых пользователем данных

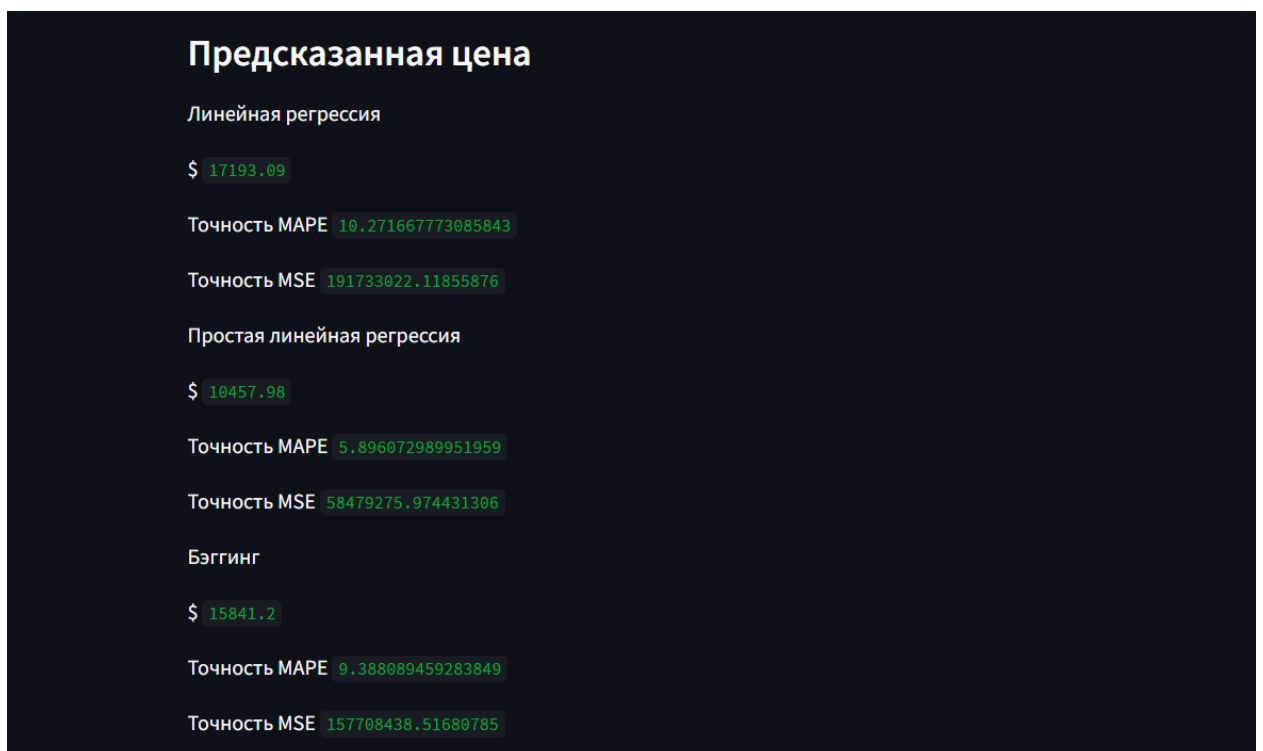


Рисунок 13 – Предсказания моделей на введённых пользователем данных

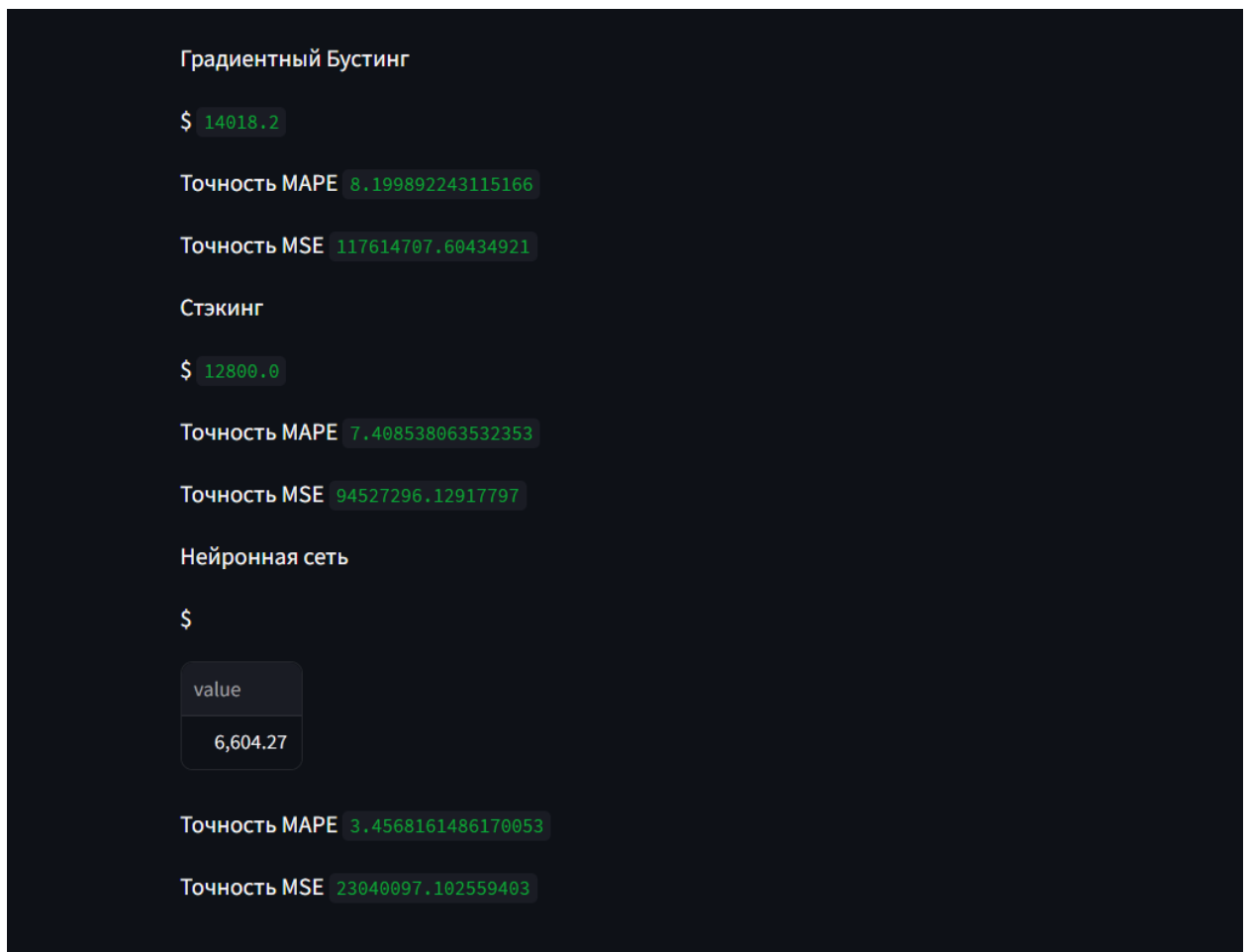


Рисунок 14 – Предсказания моделей на введенных пользователем данных

Приложение Б

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import joblib
import tensorflow as tf
import keras
import bias

def mean_absolute_percentage_error(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))
    return mape

def mean_squared_error(y_true, y_pred):
    mse = np.mean((y_true - y_pred)**2)
    return mse

def predict_for_dataset(clf, X_test, y_test):
    y_pred = clf.predict(X_test)

    st.write(f'MAPE: {mean_absolute_percentage_error(y_test, y_pred)}')
    st.write(f'MSE: {mean_squared_error(y_test, y_pred)}')

def show_author_page():
    st.title("Разработка Web-приложения (дашборда) для инференса (вывода) моделей ML и анализа данных")

    st.header("Автор")
    st.write("ФИО: Финк Эдуард Валентинович")
    st.write("Группа: ФИТ-222")
    image = "Project/This is for RGR.jpg"
    st.image(image, caption='Моё фото', use_column_width=True)

def show_dataset_page():
    st.title("Информация о наборе данных")

    st.header("Тематика набора данных")
    st.write("Этот классический набор данных содержит цены и другие атрибуты почти 54 000 бриллиантов. Это отличный набор данных для начинающих, которые учатся работать с анализом данных и визуализацией.")

    st.header("Описание признаков")
    st.write("price: цена в долларах США (\$326--\$18,823)")
    st.write("carat: вес бриллианта (0.2--5.01)")
    st.write("cut: качество среза (Fair, Good, Very Good, Premium, Ideal)")
    st.write("color: цвет бриллианта, от J (worst) до D (best)")
    st.write("clarity: измерение того, насколько чист алмаз (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))")
    st.write("x: длина в мм (0--10.74)")
    st.write("y: ширина в мм (0--58.9)")
    st.write("z: глубина в мм (0--31.8)")
    st.write("depth: общая глубина в процентах = z / mean(x, y) = 2 * z / (x + y) (43--79)")
    st.write("table: ширина вершины ромба относительно самой широкой точки (43--95)")

    st.header("Особенности предобработки данных")
    st.write("В данном наборе данных нужно было предугадывать цену бриллианта в долларах США. Сама цена находится в столбике price.")
```

```

    st.write("В наборе данных присутствовал столбик 'Unnamed: 0', который не
является нужным, он был удалён.")
    st.write("Также в датасете присутствовали дубликаты, которые в
последствии были удалены.")
    st.write("Столбики cut, color, clarity были представлены в виде чисел.")
    st.subheader("Для cut замена выглядит так:")
    ratings = {
        'Ideal': 5,
        'Premium': 4,
        'Very Good': 3,
        'Good': 2,
        'Fair': 1
    }
    for rating, value in ratings.items():
        st.write(f"{rating}: {value}")
    st.subheader("Для color замена выглядит так:")
    ratings = {
        'D': 7,
        'E': 6,
        'F': 5,
        'G': 4,
        'H': 3,
        'I': 2,
        'J': 1
    }
    for rating, value in ratings.items():
        st.write(f"{rating}: {value}")
    st.subheader("Для clarity замена выглядит так:")
    ratings = {
        'IF': 8,
        'VVS1': 7,
        'VVS2': 6,
        'VS1': 5,
        'VS2': 4,
        'SI1': 3,
        'SI2': 2,
        'I1': 1
    }
    for rating, value in ratings.items():
        st.write(f"{rating}: {value}")
    st.write("Был проведен EDA и удалены выбросы.")

def show_visualizations_page():
    st.title("Визуализации данных")

    data = pd.read_csv("Project/diamonds_redux.csv")
    st.subheader("Примеры визуализаций:")

    st.write("Столбчатая диаграмма:")
    plt.figure(figsize=(8, 6))
    sns.countplot(x="color", data=data)
    st.pyplot(plt)

    st.write("Ящик с усами:")
    plt.figure(figsize=(8, 6))
    sns.boxplot(x="color", y="price", data=data)
    st.pyplot(plt)

    st.write("Тепловая карта:")
    plt.figure(figsize=(10, 8))
    sns.heatmap(data.corr(), annot=True, cmap="coolwarm")
    st.pyplot(plt)

```

```

st.write("Точечный график:")
plt.figure(figsize=(8, 6))
sns.scatterplot(x="carat", y="price", data=data)
st.pyplot(plt)

def show_predict_dataset_page():
    model1 = joblib.load("Project/LinearRegression_model.pkl")
    model2 = joblib.load("Project/Linear-one_model.pkl")
    model3 = joblib.load("Project/bagging-one_model.pkl")
    model4 = joblib.load("Project/gradient_boosting_model.pkl")
    model5 = joblib.load("Project/stacking_model.pkl")
    model6 = keras.models.load_model("Project/NeuralNetwork.h5")
    uploaded_file = st.file_uploader("Загрузите файл данных (CSV)",
type="csv")

    if uploaded_file:
        df = pd.read_csv(uploaded_file)
        df.drop(df.filter(regex="Unname"),axis=1, inplace=True)
        st.write("Полученный набор данных:", df)

        models_mapping = {
            'LinearRegression(множественная)': model1,
            'LinearRegression(простая)': model2,
            'bagging': model3,
            'boosting': model4,
            'stacking': model5,
            'NeuralNetwork': model6
        }
        cut_model = st.selectbox("Выберите тип модели",
list(models_mapping.keys()))
        model = models_mapping[cut_model]
        y = df["price"]
        X = df
        X.drop(['price'], axis=1, inplace=True)
        if model is model2:
            X = df["carat"]
            X = X.values.reshape(-1, 1)
        elif model is model6:
            sample = df.sample(n=1000)
            X = np.array(sample)
            y = np.array(y)
        predict_for_dataset(model, X, y)

def show_prediction_page():
    st.title("Предсказание цены алмаза")

    model1 = joblib.load("Project/LinearRegression_model.pkl")
    model2 = joblib.load("Project/Linear-one_model.pkl")
    model3 = joblib.load("Project/bagging-one_model.pkl")
    model4 = joblib.load("Project/gradient_boosting_model.pkl")
    model5 = joblib.load("Project/stacking_model.pkl")
    model6 = keras.models.load_model("Project/NeuralNetwork.h5")

    uploaded_file = st.file_uploader("Загрузите файл данных (CSV)",
type="csv")
    if uploaded_file:
        df = pd.read_csv(uploaded_file)
        y_df = df["price"]

        st.subheader("Параметры алмаза")
        carat = st.slider("Carat", 0.2, 5.01, 1.0)
        cut_mapping = {
            'Fair': 1,

```

```

        'Good': 2,
        'Very Good': 3,
        'Premium': 4,
        'Ideal': 5
    }
    cut_text = st.selectbox("Cut Quality", list(cut_mapping.keys()))
    cut = cut_mapping[cut_text]
    color_mapping = {
        'D': 7,
        'E': 6,
        'F': 5,
        'G': 4,
        'H': 3,
        'I': 2,
        'J': 1
    }
    color_text = st.selectbox("Color", list(color_mapping.keys()))
    color = color_mapping[color_text]
    clarity_mapping = {
        'IF': 8,
        'VVS1': 7,
        'VVS2': 6,
        'VS1': 5,
        'VS2': 4,
        'SI1': 3,
        'SI2': 2,
        'I1': 1
    }
    clarity_text = st.selectbox("Clarity", list(clarity_mapping.keys()))
    clarity = clarity_mapping[clarity_text]
    x = st.slider("Length (mm)", 0.0, 10.74, 5.0)
    y = st.slider("Width (mm)", 0.0, 58.9, 3.0)
    z = st.slider("Depth (mm)", 0.0, 31.8, 2.0)
    depth = st.slider("Depth Percentage (%)", 43, 79, 60)
    table = st.slider("Table Width (%)", 43, 95, 60)

    features = np.array([[carat, cut, color, clarity, x, y, z, depth,
table]])
    features1 = np.array(carat).reshape(1, -1)
    price_prediction1 = model1.predict(features)
    price_prediction2 = model2.predict(features1)
    price_prediction3 = model3.predict(features)
    price_prediction4 = model4.predict(features)
    price_prediction5 = model5.predict(features)
    price_prediction6 = model6.predict(features)

    st.subheader("Предсказанная цена")
    st.write("Линейная регрессия")
    st.write("$", np.round(price_prediction1[0], 2))
    st.write("Точность MAPE", mean_absolute_percentage_error(y_df,
np.round(price_prediction1[0], 2)))
    st.write("Точность MSE", mean_squared_error(y_df,
np.round(price_prediction1[0], 2)))
    st.write("Простая линейная регрессия")
    st.write("$", np.round(price_prediction2[0], 2))
    st.write("Точность MAPE", mean_absolute_percentage_error(y_df,
np.round(price_prediction2[0], 2)))
    st.write("Точность MSE", mean_squared_error(y_df,
np.round(price_prediction2[0], 2)))
    st.write("Бэггинг")
    st.write("$", np.round(price_prediction3[0], 2))
    st.write("Точность MAPE", mean_absolute_percentage_error(y_df,
np.round(price_prediction3[0], 2)))

```

```

        st.write("Точность MSE", mean_squared_error(y_df,
np.round(price_prediction3[0], 2)))
        st.write("Градиентный Бустинг")
        st.write("$", np.round(price_prediction4[0], 2))
        st.write("Точность MAPE", mean_absolute_percentage_error(y_df,
np.round(price_prediction4[0], 2)))
        st.write("Точность MSE", mean_squared_error(y_df,
np.round(price_prediction4[0], 2)))
        st.write("Стэкинг")
        st.write("$", np.round(price_prediction5[0], 2))
        st.write("Точность MAPE", mean_absolute_percentage_error(y_df,
np.round(price_prediction5[0], 2)))
        st.write("Точность MSE", mean_squared_error(y_df,
np.round(price_prediction5[0], 2)))
        st.write("Нейронная сеть")
        st.write("$", np.round(price_prediction6[0], 2))
        st.write("Точность MAPE", mean_absolute_percentage_error(y_df,
np.round(price_prediction6[0], 2)))
        st.write("Точность MSE", mean_squared_error(y_df,
np.round(price_prediction6[0], 2)))

pages = {
    "Об авторе": show_author_page,
    "Набор данных": show_dataset_page,
    "Визуализация": show_visualizations_page,
    "Предсказания для набора данных": show_predict_dataset_page,
    "Предсказания": show_prediction_page
}

page = st.sidebar.selectbox("Выберите страницу", tuple(pages.keys()))

pages[page]()

```