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 | Есакова Кирилла Николаевича | | |
| Курс | 2 | Группа | ФИТ-232 |
| Направление | 02.03.02 Фундаментальная информатика и информационные технологии | | |
| Руководитель | Старший преподаватель | | |
|  | Шарун Иван Владимирович | | |
| Выполнил |  | | |
| Проверил |  | | |

Омск 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc200215907)

[1 Описание выбранных моделей 4](#_Toc200215908)

[1.1 Деревья решений 4](#_Toc200215909)

[1.2 Сериализация 4](#_Toc200215910)

[1.3 Bagging Regressor, Gradient Boosting Regressor, Stacking Regressor 5](#_Toc200215911)

[2 Процесс разработки дашборда 5](#_Toc200215912)

[2.1 Разработка страницы с информацией об авторе 6](#_Toc200215913)

[2.2 Разработка страницы с информацией о датасете 6](#_Toc200215914)

[2.3 Разработка страницы с визуализацией данных 7](#_Toc200215915)

[2.4 Разработка страницы для получения инференса 8](#_Toc200215916)

[2.5 Разработка страницы со вступительным экраном и содержанием 10](#_Toc200215917)

[ЗАКЛЮЧЕНИЕ 12](#_Toc200215918)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 13](#_Toc200215919)

[ПРИЛОЖЕНИЕ А 14](#_Toc200215920)

[ПРИЛОЖЕНИЕ Б (обязательное) 19](#_Toc200215921)

ВВЕДЕНИЕ

Актуальность темы данной РГР заключается в применении приобретённых навыков машинного обучения к решению практических задач. Работа представляет собой пример такой задачи — разработку веб-приложения для получения инференса моделей.

В процессе выполнения были изучены и использованы следующие инструменты: библиотека *scikit-learn* — для обучения ансамблевых моделей (включая стекинг, бустинг и бэггинг), *pickle* — для сохранения обученных моделей *scikit-learn*, а также *streamlit* — для создания веб-приложения на языке программирования *Python*.

Целью РГР является разработка *Web*-приложения для получения инференса моделями машинного обучения. Для достижения этой цели были поставлены следующие задачи:

1. Выбрать модели.
2. Сохранить обученные модели.
3. Реализовать *Web*-приложение с использованием сохранённых моделей.

# **1 Описание выбранных моделей**

## **1.1 Деревья решений**

В качестве базовой модели был выбран алгоритм DecisionTreeRegressor. В ходе лабораторных исследований проводилось тестирование различных конфигураций дерева: варьировались параметры максимальной глубины (max\_depth от 3 до 15), минимального числа образцов для разделения узлов (min\_samples\_split от 2 до 30) и минимального числа образцов в листе (min\_samples\_leaf от 1 до 20). Особое внимание уделялось регуляризационным параметрам - коэффициенту минимального уменьшения примеси (min\_impurity\_decrease) и параметру обрезки ccp\_alpha. Для оценки эффективности модель сравнивалась с другими регрессорами, включая линейную регрессию, метод опорных векторов (SVR) и KNeighborsRegressor. Наилучшие результаты продемонстрировала конфигурация с max\_depth=9, min\_samples\_split=27, min\_samples\_leaf=7 и ccp\_alpha=0.05, что обеспечило оптимальный баланс между точностью предсказаний и устойчивостью к переобучению. Именно эти параметры были выбраны для финальной версии модели.

## **1.2 Сериализация**

Для сериализации моделей, обученных с использованием библиотеки *scikit*-*learn*, применялась библиотека *pickle* и её метод *dump*, позволяющий сохранять модель в файл с расширением .*pkl*. С помощью той же библиотеки осуществлялась десериализация — извлечение моделей из файлов и интеграция их в работу дашборда.  
На рисунке 1 представлена реализация процесса сохранения всех моделей [1].

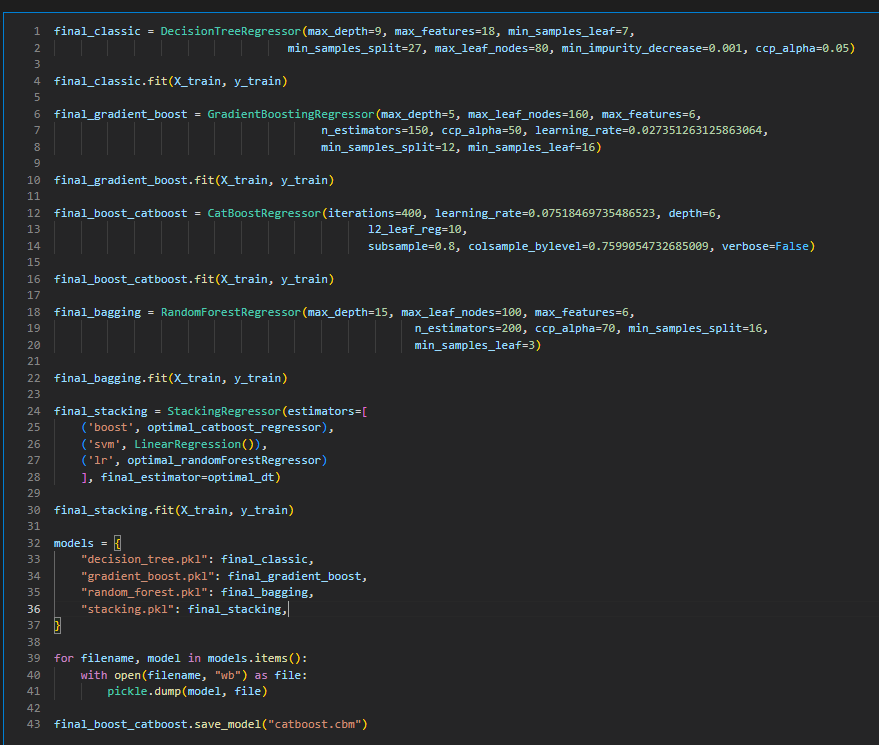


Рисунок 1 – Сохранение модели через библиотеку pickle

## **1.3 Bagging Regressor, Gradient Boosting Regressor, Stacking Regressor**

Дополнительно были выбраны модели с использованием *Bagging* *Regressor*, *Gradient* *Boosting* *Regressor* и *Stacking* *Regressor* из библиотеки *scikit*-*learn*. Их выбор был обоснован результатами практических тестов, проведённых в ходе выполнения лабораторной работы [2].

Также рассматривались альтернативные варианты моделей на основе *CatBoost*, *XGBoost* и *LightGBM*. Однако решающим фактором при выборе послужило меньшее время обучения и кросс-валидации у моделей из *scikit*-*learn*, поскольку достигнутые значения метрик качества были сопоставимы.

# **2 Процесс разработки дашборда**

В ходе работы было принято решение реализовывать дашборд поэтапно. На первом этапе была разработана его архитектура, определяющая структуру и основные компоненты интерфейса. На следующем этапе была выполнена интеграция обученных моделей в веб-приложение, что позволило обеспечить функциональность инференса непосредственно через дашборд [3].

## **2.1 Разработка страницы с информацией об авторе**

После создания главной страницы была реализована страница с информацией об авторе, размещённая в файле *Об\_авторе.py*. На данной странице представлены ФИО, учебная группа, фотография и тема РГР. Также размещены четыре ссылки для перехода на остальные страницы дашборда.

Для отображения информации использовались функции *title*, *markdown* и *image*, предоставляемые библиотекой *Streamlit*.

Пример кода, реализующего страницу с данными об авторе, показан на рисунке 2.

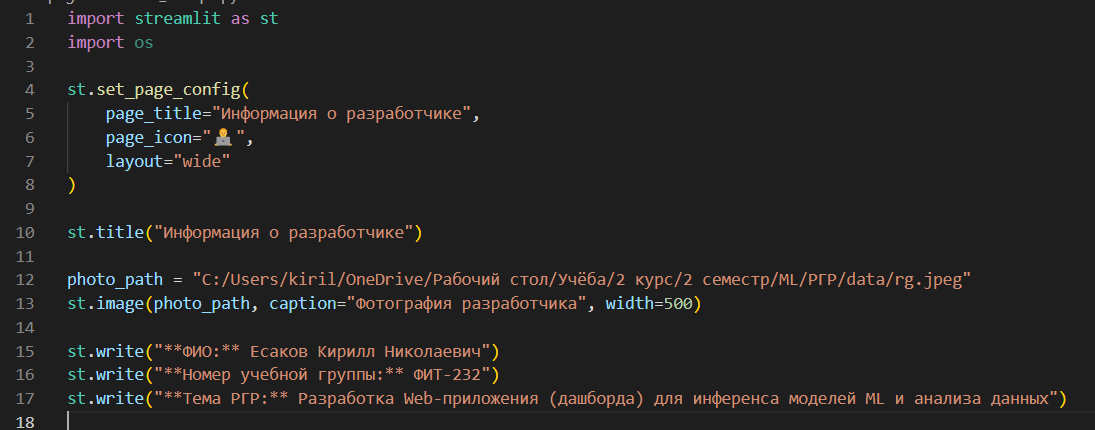


Рисунок 2 – Код для создания страницы об авторе

## **2.2 Разработка страницы с информацией о датасете**

Файл *Описание\_данных\_датасета.py* содержит код, реализующий страницу с описанием датасета и особенностями его предобработки. Для структурирования информации и удобства взаимодействия с пользователем были использованы методы *cache\_data* и *markdown*

На рисунке 3 показан фрагмент кода, демонстрирующий создание страницы об описании датасета.

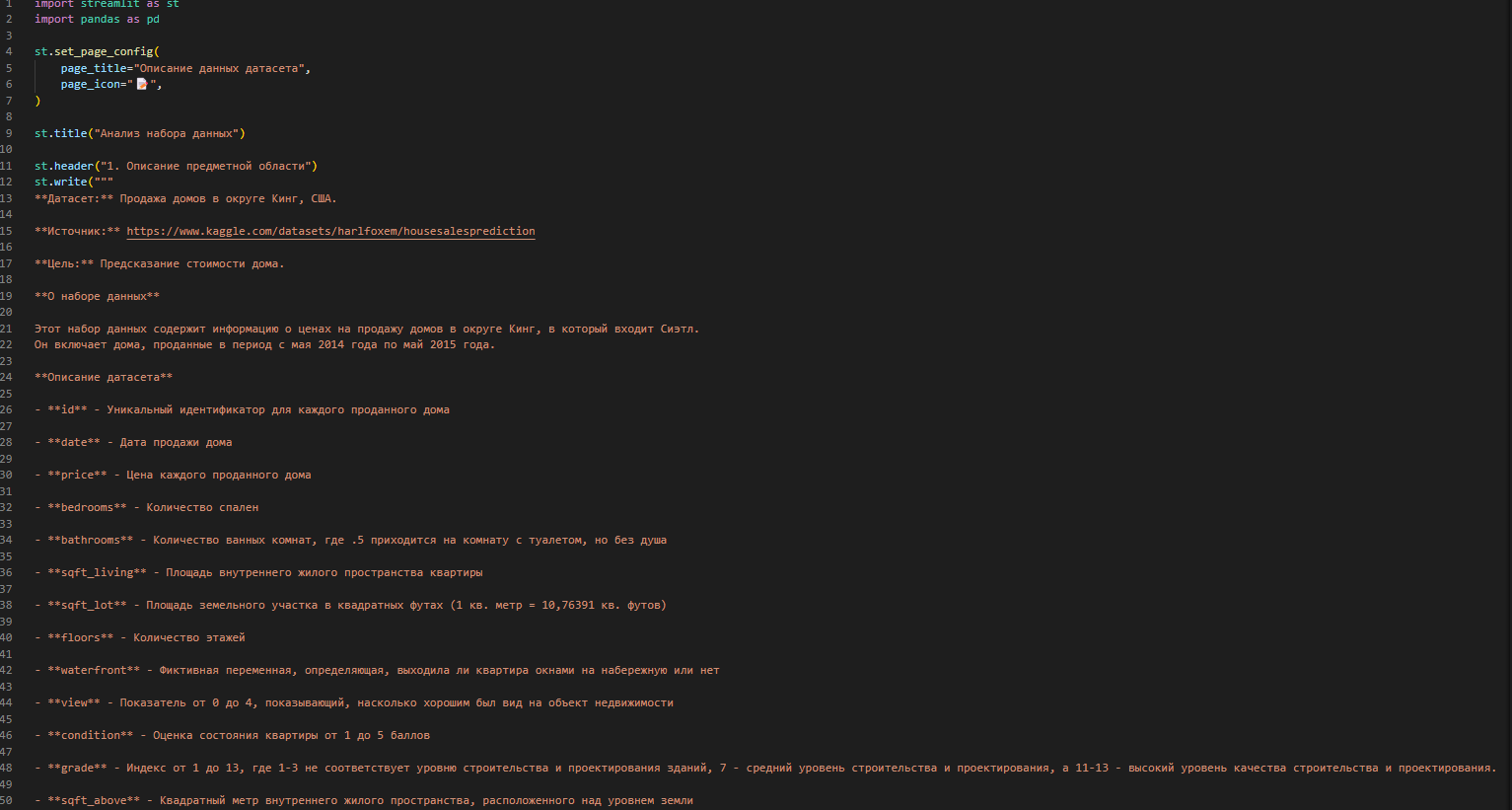


Рисунок 3 – Код для создания страницы о датасете

## **2.3 Разработка страницы с визуализацией данных**

На данной странице пользователь может увидеть графики распределения таргета, непрерывных и категориальных признаков. Для этого использовались такие виды графиков как *histplot*, *heatmap*, *boxplot*, *scatterplot.* На рисунке 4 представлена часть реализации разработки страницы с графиками.

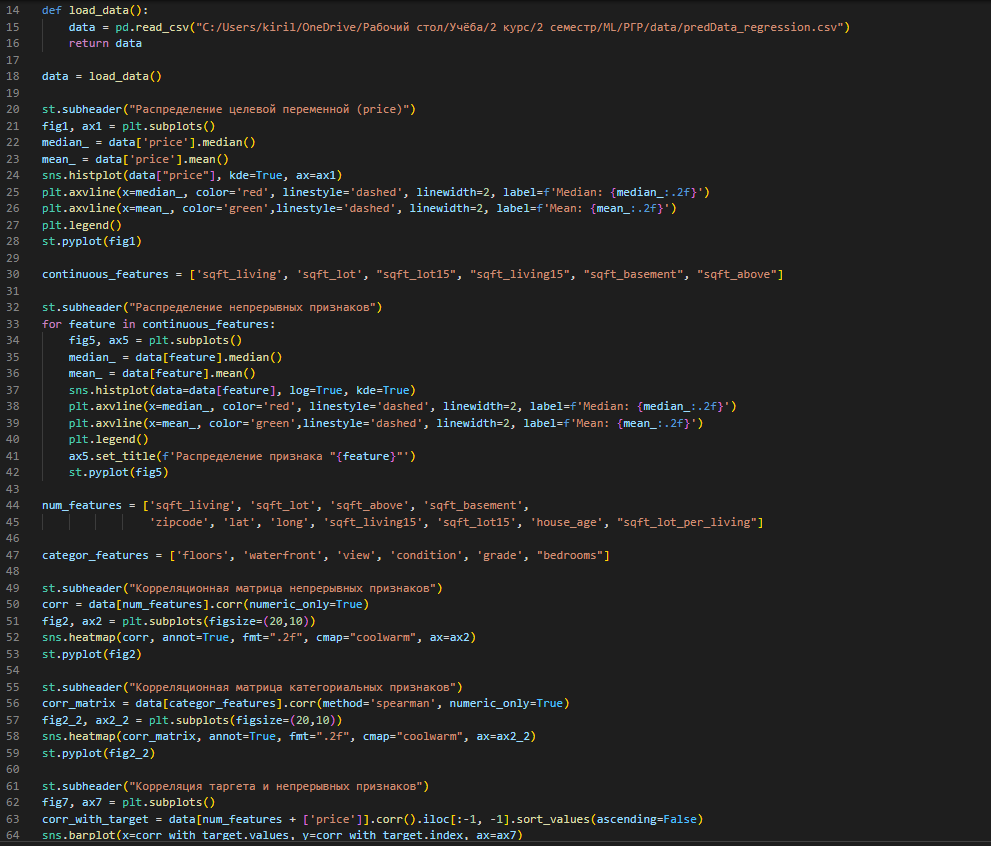


Рисунок 4 – Составление графиков

## **2.4 Разработка страницы для получения инференса**

На данной странице пользователь может получить предсказание, сформированное на основе работы ранее описанных моделей, используя введённые данные. Для этого предлагается загрузить собственный датасет.

На рисунке 5 приведён пример кода, отвечающего за загрузку *CSV*-файла.



Рисунок 5 – Код для загрузки датасета

Затем был написан код, отвечающий за загрузку моделей, а также отображение полей для ввода данных пользователем. Пример такого кода приведен на рисунке 6.



Рисунок 6 – Код для записи данных пользователем

Далее пользователь может получить предсказание от моделей, нажав на кнопку “Получить прогноз срабатывания датчика”. После ее нажатия выдаётся ответ. Ниже на рисунке 7 представлен код для получения предсказания.

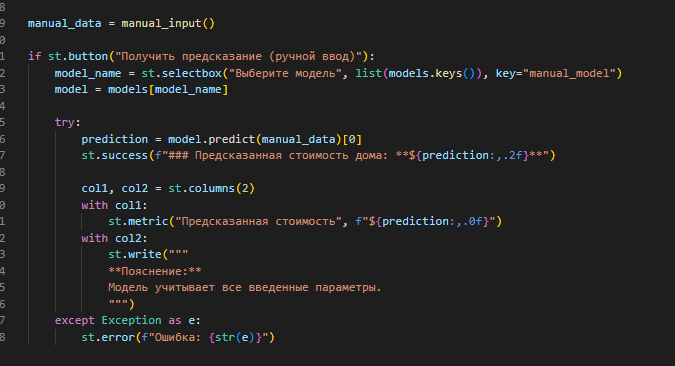


Рисунок 7 – Код для получения предсказания

## **2.5 Разработка страницы со вступительным экраном и содержанием**

Чтобы пользователю было удобно переключаться между вкладками, в корневой папке проекта был создать файл *Добро\_пожаловать.py.* Благодаря этому streamlit автоматически создал навигацию по всем страницам. Реализация этой страницы представлена на рисунке 8.

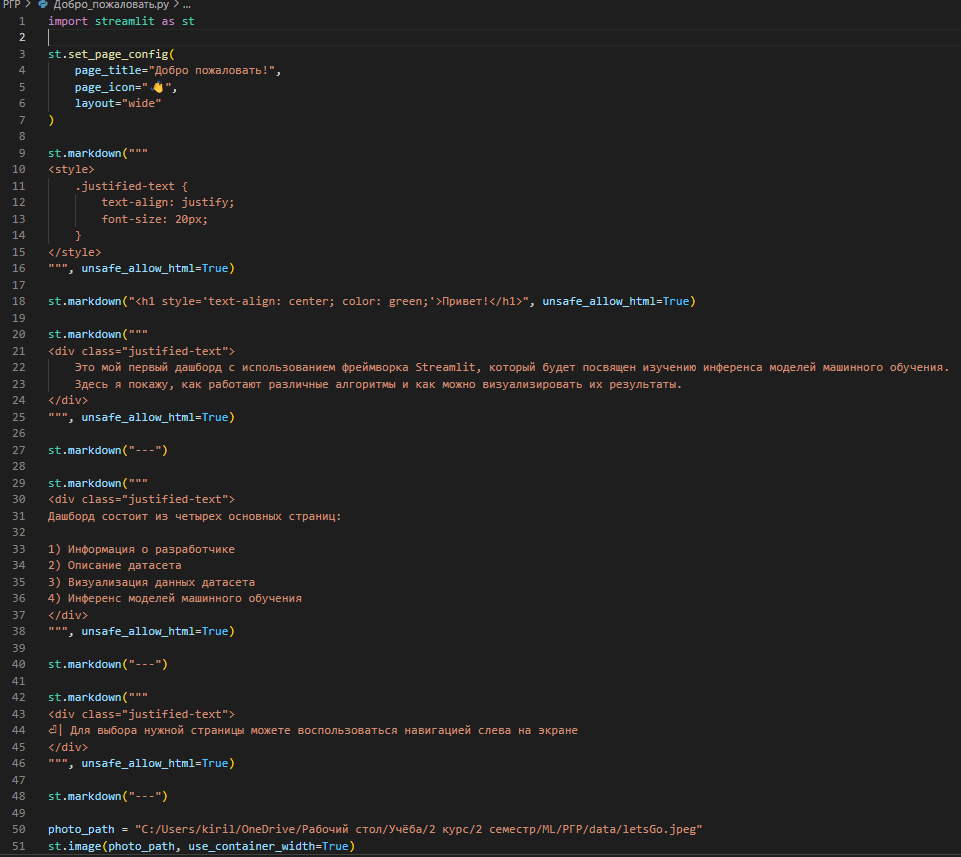


Рисунок 8 – Код главной страницы

ЗАКЛЮЧЕНИЕ

В рамках работы была изучена библиотека *Streamlit*, предназначенная для разработки веб-приложений на языке программирования *Python*.

Был реализован дашборд, состоящий из пяти отдельных страниц: главная страница, страница с информацией об авторе, раздел с описанием датасета, страница для получения инференса, а также страница для визуализации данных.

Кроме того, в процессе разработки был приобретён практический опыт в решении конфликтов зависимостей между библиотеками, возникающих из-за несовместимых требований к версиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Сериализация в Python. URL: <https://pythongeeks.org/python-serialization/> (дата обращения 06.06.2025).

2) Официальная документация библиотеки streamlit. URL: <https://docs.streamlit.io/> (дата обращения 06.06.2025).

3) Официальная документация библиотеки scikit**-**learn. URL: <https://scikit-learn.ru/user_guide/> (дата обращения 06.06.2025).

ПРИЛОЖЕНИЕ А

(обязательное)

**Скриншоты дашборда**

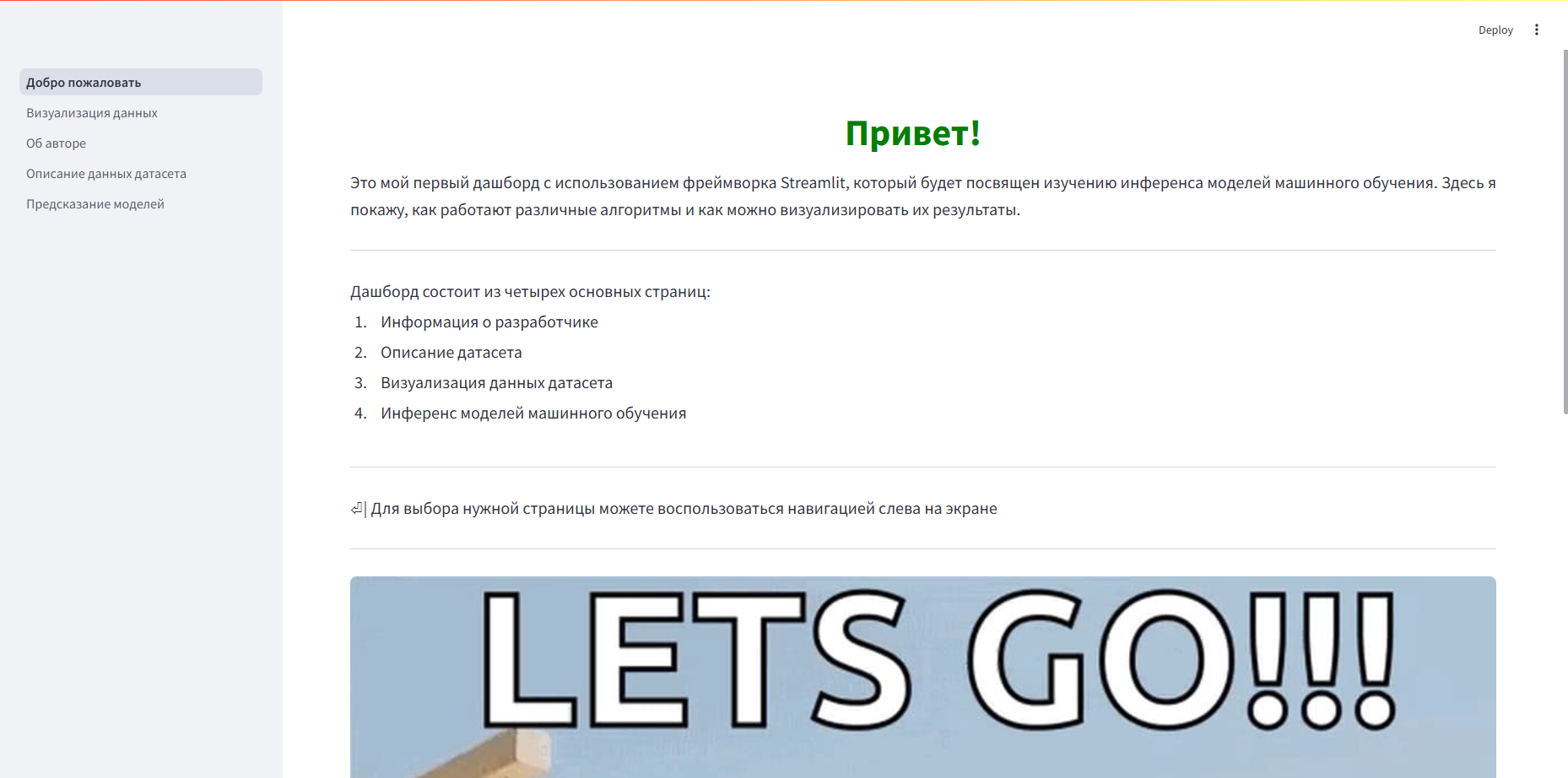


Рисунок А.1 – Главная страница

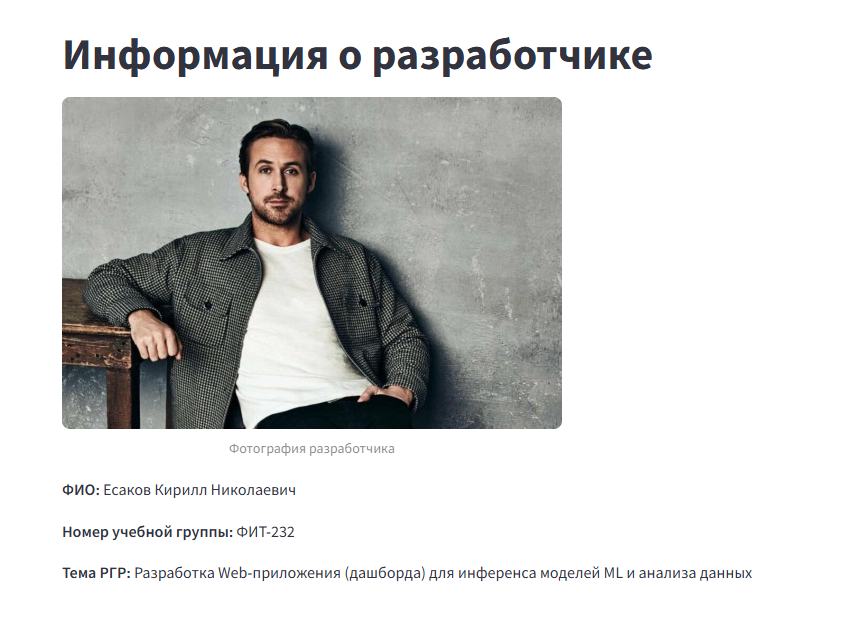


Рисунок А.2 – Страница с информацией об авторе



Рисунок А.3 – Страница с информацией о датасете



Рисунок А.4 – Страница с визуализацией признаков

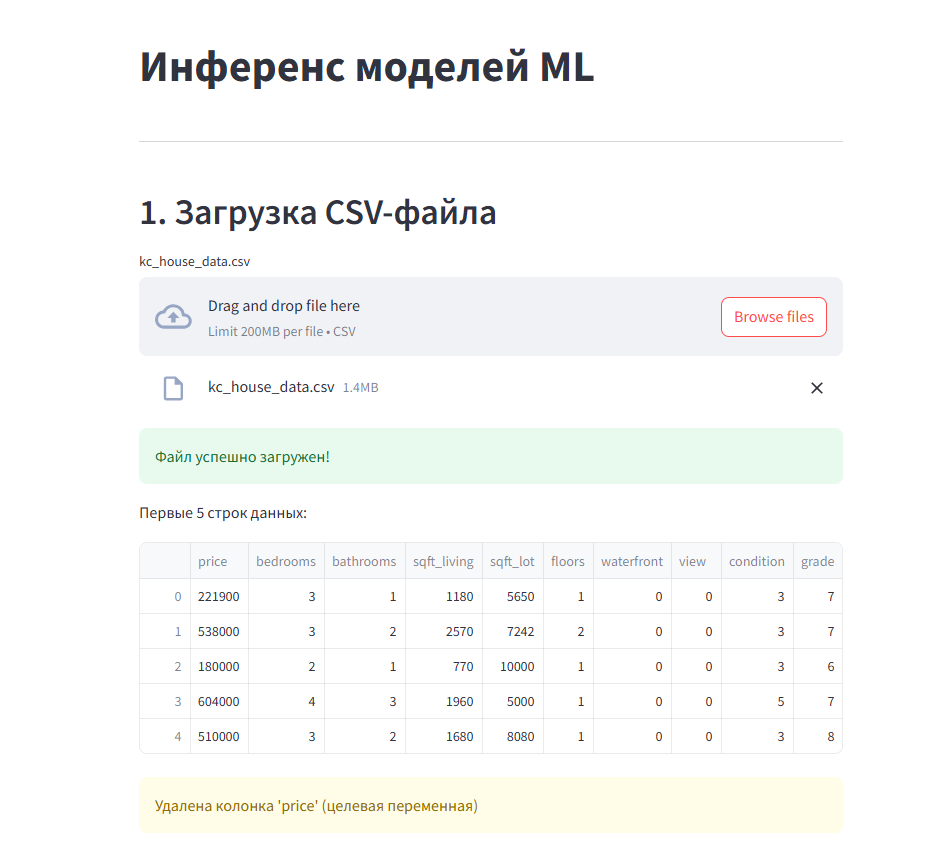


Рисунок А.5 – Инференс моделей. Загрузка датасета пользователем

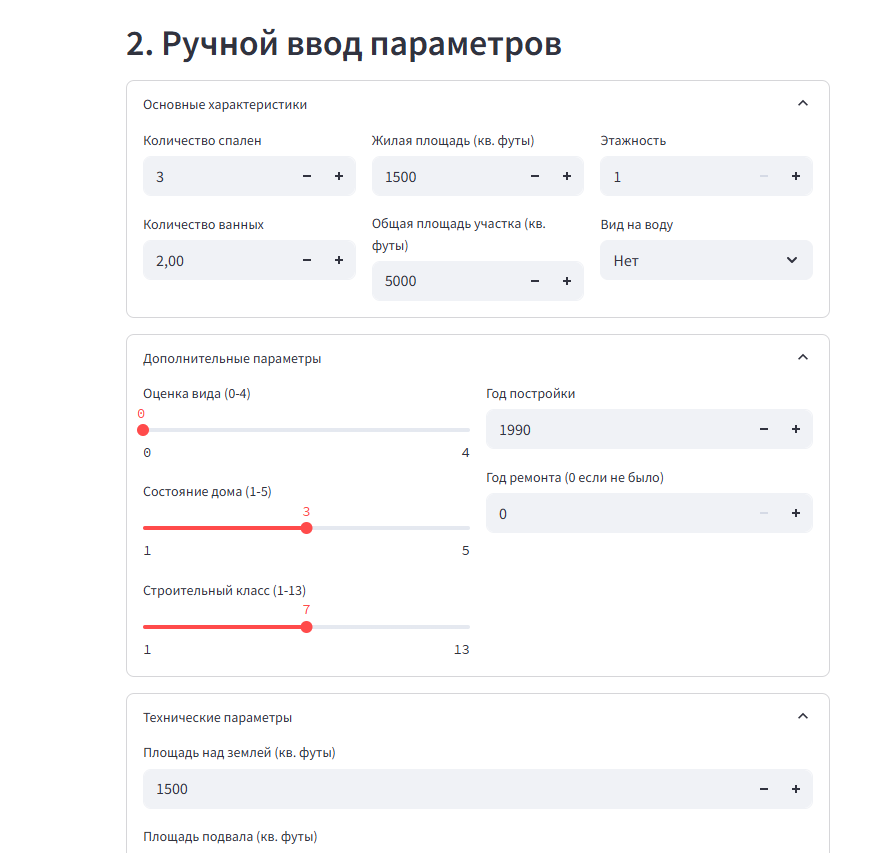


Рисунок А.6 – Инференс моделей. Ручной ввод данных всех призанков

ПРИЛОЖЕНИЕ Б  
(обязательное)

**Исходный код дашборда**

Код разработанного приложения

*Добро\_пожаловать.py*

import streamlit as st

st.set\_page\_config(

    page\_title="Добро пожаловать!",

    page\_icon="👋",

    layout="wide"

)

st.markdown("""

<style>

    .justified-text {

        text-align: justify;

        font-size: 20px;

    }

</style>

""", unsafe\_allow\_html=True)

st.markdown("<h1 style='text-align: center; color: green;'>Привет!</h1>", unsafe\_allow\_html=True)

st.markdown("""

<div class="justified-text">

    Это мой первый дашборд с использованием фреймворка Streamlit, который будет посвящен изучению инференса моделей машинного обучения.

    Здесь я покажу, как работают различные алгоритмы и как можно визуализировать их результаты.

</div>

""", unsafe\_allow\_html=True)

st.markdown("---")

st.markdown("""

<div class="justified-text">

Дашборд состоит из четырех основных страниц:

1) Информация о разработчике

2) Описание датасета

3) Визуализация данных датасета

4) Инференс моделей машинного обучения

</div>

""", unsafe\_allow\_html=True)

st.markdown("---")

st.markdown("""

<div class="justified-text">

⏎| Для выбора нужной страницы можете воспользоваться навигацией слева на экране

</div>

""", unsafe\_allow\_html=True)

st.markdown("---")

photo\_path = "C:/Users/kiril/OneDrive/Рабочий стол/Учёба/2 курс/2 семестр/ML/РГР/data/letsGo.jpeg"

st.image(photo\_path, use\_container\_width=True)

*Об\_авторе.py*

import streamlit as st

import os

st.set\_page\_config(

    page\_title="Информация о разработчике",

    page\_icon="👨‍💻",

    layout="wide"

)

st.title("Информация о разработчике")

photo\_path = "C:/Users/kiril/OneDrive/Рабочий стол/Учёба/2 курс/2 семестр/ML/РГР/data/rg.jpeg"

st.image(photo\_path, caption="Фотография разработчика", width=500)

st.write("\*\*ФИО:\*\* Есаков Кирилл Николаевич")

st.write("\*\*Номер учебной группы:\*\* ФИТ-232")

st.write("\*\*Тема РГР:\*\* Разработка Web-приложения (дашборда) для инференса моделей ML и анализа данных")

*Описание\_данных\_датасета.py*

import streamlit as st

import pandas as pd

st.set\_page\_config(

    page\_title="Описание данных датасета",

    page\_icon="📝",

)

st.title("Анализ набора данных")

st.header("1. Описание предметной области")

st.write("""

\*\*Датасет:\*\* Продажа домов в округе Кинг, США.

\*\*Источник:\*\* https://www.kaggle.com/datasets/harlfoxem/housesalesprediction

\*\*Цель:\*\* Предсказание стоимости дома.

\*\*О наборе данных\*\*

Этот набор данных содержит информацию о ценах на продажу домов в округе Кинг, в который входит Сиэтл.

Он включает дома, проданные в период с мая 2014 года по май 2015 года.

\*\*Описание датасета\*\*

- \*\*id\*\* - Уникальный идентификатор для каждого проданного дома

- \*\*date\*\* - Дата продажи дома

- \*\*price\*\* - Цена каждого проданного дома

- \*\*bedrooms\*\* - Количество спален

- \*\*bathrooms\*\* - Количество ванных комнат, где .5 приходится на комнату с туалетом, но без душа

- \*\*sqft\_living\*\* - Площадь внутреннего жилого пространства квартиры

- \*\*sqft\_lot\*\* - Площадь земельного участка в квадратных футах (1 кв. метр = 10,76391 кв. футов)

- \*\*floors\*\* - Количество этажей

- \*\*waterfront\*\* - Фиктивная переменная, определяющая, выходила ли квартира окнами на набережную или нет

- \*\*view\*\* - Показатель от 0 до 4, показывающий, насколько хорошим был вид на объект недвижимости

- \*\*condition\*\* - Оценка состояния квартиры от 1 до 5 баллов

- \*\*grade\*\* - Индекс от 1 до 13, где 1-3 не соответствует уровню строительства и проектирования зданий, 7 - средний уровень строительства и проектирования, а 11-13 - высокий уровень качества строительства и проектирования.

- \*\*sqft\_above\*\* - Квадратный метр внутреннего жилого пространства, расположенного над уровнем земли

- \*\*sqft\_basement\*\* - Квадратный метр внутреннего жилого пространства, находящегося ниже уровня земли

- \*\*yr\_built\*\* - Году, когда дом был впервые построен

- \*\*yr\_renovated\*\* - Год последней реконструкции дома

- \*\*zipcode\*\* - В каком районе с почтовым индексом находится дом

- \*\*lat\*\* - Широта

- \*\*long\*\* - Долгота

- \*\*sqft\_living15\*\* - Квадратный метр внутренней жилой площади для 15 ближайших соседей

- \*\*sqft\_lot15\*\* - Площадь земельных участков ближайших 15 соседей

""")

@st.cache\_data

def load\_data():

    data = pd.read\_csv("C:/Users/kiril/OneDrive/Рабочий стол/Учёба/2 курс/2 семестр/ML/РГР/data/predData\_regression.csv")

    return data

data = load\_data()

st.header("2. Предобработка данных")

st.write("""

Перед анализом были выполнены следующие шаги:

- Удалены дубликаты.

- Заполнены пропуски (если были).

- Добавлены дополнительные признаки.

""")

st.subheader("Первые 10 строк предобработанного датасета")

st.dataframe(data.head(10))

st.subheader("Основные статистики")

st.write(data.describe())

*Визуализация\_данных.py*

import streamlit as st

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

st.set\_page\_config(

    page\_title="Визуализация данных",

    page\_icon="📊",

)

st.title('Визуализация данных')

@st.cache\_data

def load\_data():

    data = pd.read\_csv("C:/Users/kiril/OneDrive/Рабочий стол/Учёба/2 курс/2 семестр/ML/РГР/data/predData\_regression.csv")

    return data

data = load\_data()

st.subheader("Распределение целевой переменной (price)")

fig1, ax1 = plt.subplots()

median\_ = data['price'].median()

mean\_ = data['price'].mean()

sns.histplot(data["price"], kde=True, ax=ax1)

plt.axvline(x=median\_, color='red', linestyle='dashed', linewidth=2, label=f'Median: {median\_:.2f}')

plt.axvline(x=mean\_, color='green',linestyle='dashed', linewidth=2, label=f'Mean: {mean\_:.2f}')

plt.legend()

st.pyplot(fig1)

continuous\_features = ['sqft\_living', 'sqft\_lot', "sqft\_lot15", "sqft\_living15", "sqft\_basement", "sqft\_above"]

st.subheader("Распределение непрерывных признаков")

for feature in continuous\_features:

    fig5, ax5 = plt.subplots()

    median\_ = data[feature].median()

    mean\_ = data[feature].mean()

    sns.histplot(data=data[feature], log=True, kde=True)

    plt.axvline(x=median\_, color='red', linestyle='dashed', linewidth=2, label=f'Median: {median\_:.2f}')

    plt.axvline(x=mean\_, color='green',linestyle='dashed', linewidth=2, label=f'Mean: {mean\_:.2f}')

    plt.legend()

    ax5.set\_title(f'Распределение признака "{feature}"')

    st.pyplot(fig5)

num\_features = ['sqft\_living', 'sqft\_lot', 'sqft\_above', 'sqft\_basement',

                'zipcode', 'lat', 'long', 'sqft\_living15', 'sqft\_lot15', 'house\_age', "sqft\_lot\_per\_living"]

categor\_features = ['floors', 'waterfront', 'view', 'condition', 'grade', "bedrooms"]

st.subheader("Корреляционная матрица непрерывных признаков")

corr = data[num\_features].corr(numeric\_only=True)

fig2, ax2 = plt.subplots(figsize=(20,10))

sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm", ax=ax2)

st.pyplot(fig2)

st.subheader("Корреляционная матрица категориальных признаков")

corr\_matrix = data[categor\_features].corr(method='spearman', numeric\_only=True)

fig2\_2, ax2\_2 = plt.subplots(figsize=(20,10))

sns.heatmap(corr\_matrix, annot=True, fmt=".2f", cmap="coolwarm", ax=ax2\_2)

st.pyplot(fig2\_2)

st.subheader("Корреляция таргета и непрерывных признаков")

fig7, ax7 = plt.subplots()

corr\_with\_target = data[num\_features + ['price']].corr().iloc[:-1, -1].sort\_values(ascending=False)

sns.barplot(x=corr\_with\_target.values, y=corr\_with\_target.index, ax=ax7)

st.pyplot(fig7)

st.subheader("Корреляция таргета и категориальных признаков")

fig8, ax8 = plt.subplots()

corr\_with\_target = data[categor\_features + ['price']].corr(method='spearman').iloc[:-1, -1].sort\_values(ascending=False)

sns.barplot(x=corr\_with\_target.values, y=corr\_with\_target.index, ax=ax8)

st.pyplot(fig8)

st.subheader("Boxplot: Цена в разрезе категориальных признаков")

for feature in categor\_features:

    fig6, ax6 = plt.subplots()

    sns.boxplot(x=feature, y="price", data=data, ax=ax6)

    st.pyplot(fig6)

important\_continuous\_features = ['sqft\_living', 'sqft\_lot', 'lat', 'long', 'sqft\_living15', "sqft\_lot\_per\_living"]

st.subheader("Зависимость цены от наиболее важных непрерывных признаков")

for feature in important\_continuous\_features:

    fig4, ax4 = plt.subplots()

    sns.scatterplot(x=feature, y="price", data=data, ax=ax4)

    st.pyplot(fig4)

st.header('Выводы по EDA')

st.write("""

Факторы, влияющие на цену:

- \*\*Жилая площадь:\*\*

Знак корреляции положительный, значит если растет жилая площадь, то растет и цена.

- \*\*Индекс качества строительства:\*\*

Данный индекс влияет на цену дома больше всего. Знак корреляции положительный, значит если растет индекс качества строительства, то растет и цена.

- \*\*Жилая площадь соседей:\*\*

Знак корреляции положительный, значит если растет жилая площадь соседей, то растет и цена.

- \*\*Расположение по широте:\*\*

Знак корреляции положительный, значит с ростом широты растет и цена.

- \*\*Количество спален:\*\*

Количество спален незначительно влияет на цену дома. Знак корреляции положительный, значит с ростом количества спален растет и цена.

- \*\*Количество ванных комнат:\*\*

Количество вынных комнат также незначительно влияет на цену дома. Знак корреляции положительный, значит с ростом количества ванных комнат растет и цена.

- \*\*Доля общей площади к жилой:\*\*

Знак корреляции отрицательный, значит если растет доля общей площади к жилой, то цена падает

""")

*Предсказание\_моделей.py*

import streamlit as st

import pandas as pd

import pickle

from catboost import CatBoostRegressor

st.set\_page\_config(

    page\_title="Предсказание модели",

    page\_icon="🔍"

)

st.title("Инференс моделей ML")

st.markdown("---")

@st.cache\_resource

def load\_models():

    models = {

        "Дерево решений": pickle.load(open("models/decision\_tree.pkl", "rb")),

        "Градиентный бустинг": pickle.load(open("models/gradient\_boost.pkl", "rb")),

        "CatBoost": CatBoostRegressor().load\_model("models/catboost.cbm"),

        "Случайный лес": pickle.load(open("models/random\_forest.pkl", "rb")),

        "Стекинг": pickle.load(open("models/stacking.pkl", "rb")),

    }

    return models

models = load\_models()

st.header("1. Загрузка CSV-файла")

uploaded\_file = st.file\_uploader("kc\_house\_data.csv", type=["csv"])

if uploaded\_file:

    data = pd.read\_csv(uploaded\_file)

    st.success("Файл успешно загружен!")

    st.write("Первые 5 строк данных:")

    st.dataframe(data.head())

    if "price" in data.columns:

        st.warning("Удалена колонка 'price' (целевая переменная)")

        new\_data = data.drop("price", axis=1)

    model\_name = st.selectbox("Выберите модель", list(models.keys()))

    if st.button("Получить предсказания"):

        model = models[model\_name]

        predictions = model.predict(new\_data)

        predictions\_df = pd.DataFrame({'Действительное значение (в долларах США $)': data['price'], 'Предсказанное значение (в долларах США $)': predictions})

        st.subheader("Результаты")

        st.write("""

        Результаты представлены в виде Таблицы, где в левом столбце находятся реальные значения стоимости дома,

        а в правом - значения, предсказанные алгоритмом машинного обучения.

        """)

        st.dataframe(predictions\_df)

        st.download\_button(

            label="Скачать предсказания (CSV)",

            data=data.to\_csv(index=False).encode("utf-8"),

            file\_name="predictions.csv",

            mime="text/csv",

        )

def manual\_input():

    st.header("2. Ручной ввод параметров")

    with st.expander("Основные характеристики", expanded=True):

        col1, col2, col3 = st.columns(3)

        with col1:

            bedrooms = st.number\_input("Количество спален", min\_value=1, max\_value=10, value=3)

            bathrooms = st.number\_input("Количество ванных", min\_value=1.0, max\_value=5.0, value=2.0, step=0.5)

        with col2:

            sqft\_living = st.number\_input("Жилая площадь (кв. футы)", min\_value=500, max\_value=10000, value=1500)

            sqft\_lot = st.number\_input("Общая площадь участка (кв. футы)", min\_value=500, max\_value=50000, value=5000)

        with col3:

            floors = st.number\_input("Этажность", min\_value=1, max\_value=3, value=1)

            waterfront = st.selectbox("Вид на воду", [0, 1], format\_func=lambda x: "Да" if x == 1 else "Нет")

    with st.expander("Дополнительные параметры"):

        col1, col2 = st.columns(2)

        with col1:

            view = st.slider("Оценка вида (0-4)", 0, 4, 0)

            condition = st.slider("Состояние дома (1-5)", 1, 5, 3)

            grade = st.slider("Строительный класс (1-13)", 1, 13, 7)

        with col2:

            yr\_built = st.number\_input("Год постройки", min\_value=1900, max\_value=2023, value=1990)

            yr\_renovated = st.number\_input("Год ремонта (0 если не было)", min\_value=0, max\_value=2023, value=0)

    with st.expander("Технические параметры", expanded=False):

        sqft\_above = st.number\_input("Площадь над землей (кв. футы)", min\_value=500, value=1500)

        sqft\_basement = st.number\_input("Площадь подвала (кв. футы)", min\_value=0, value=0)

        zipcode = st.number\_input("Почтовый индекс", min\_value=98001, max\_value=98199, value=98178)

        lat = st.number\_input("Широта", min\_value=47.0, max\_value=48.0, value=47.5)

        long = st.number\_input("Долгота", min\_value=-123.0, max\_value=-121.0, value=-122.0)

        sqft\_living15 = st.number\_input("Средняя жилая площадь соседей (кв. футы)", min\_value=500, value=1500)

        sqft\_lot15 = st.number\_input("Средняя площадь участка соседей (кв. футы)", min\_value=500, value=5000)

    input\_data = {

        'bedrooms': bedrooms,

        'bathrooms': bathrooms,

        'sqft\_living': sqft\_living,

        'sqft\_lot': sqft\_lot,

        'floors': floors,

        'waterfront': waterfront,

        'view': view,

        'condition': condition,

        'grade': grade,

        'sqft\_above': sqft\_above,

        'sqft\_basement': sqft\_basement,

        'yr\_built': yr\_built,

        'yr\_renovated': yr\_renovated,

        'zipcode': zipcode,

        'lat': lat,

        'long': long,

        'sqft\_living15': sqft\_living15,

        'sqft\_lot15': sqft\_lot15

    }

    return pd.DataFrame([input\_data])

manual\_data = manual\_input()

if st.button("Получить предсказание (ручной ввод)"):

    model\_name = st.selectbox("Выберите модель", list(models.keys()), key="manual\_model")

    model = models[model\_name]

    try:

        prediction = model.predict(manual\_data)[0]

        st.success(f"### Предсказанная стоимость дома: \*\*${prediction:,.2f}\*\*")

        col1, col2 = st.columns(2)

        with col1:

            st.metric("Предсказанная стоимость", f"${prediction:,.0f}")

        with col2:

            st.write("""

            \*\*Пояснение:\*\*

            Модель учитывает все введенные параметры.

            """)

    except Exception as e:

        st.error(f"Ошибка: {str(e)}")