МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
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

на тему Разработка дашборда для инференса моделей и анализа данных

Студента Сафронова Александра Александровича
фамилия, имя, отчество полностью

Курс 2 Группа МО-211

Направление 02.03.03. Математическое обеспечение и администрирование информационных систем
код, наименование

Руководитель ассистент
должность, ученая степень, звание

Гуненков М. Ю.
фамилия, инициалы, дата, подпись

Выполнил
дата, подпись студента

Омск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Машинное обучение – это наука о разработке алгоритмов и статистических моделей, которые компьютерные системы используют для выполнения задач без явных инструкций, полагаясь вместо этого на шаблоны и логические выводы. Компьютерные системы используют алгоритмы машинного обучения для обработки больших объемов статистических данных и выявления шаблонов данных. Таким образом, системы могут более точно прогнозировать результаты на основе заданного набора входных данных. Например, специалисты по работе с данными могут обучить медицинское приложение диагностировать рак по рентгеновским изображениям, сохраняя миллионы отсканированных изображений и соответствующие диагнозы. Машинное обучение помогает компаниям стимулировать рост, открывать новые источники дохода и решать сложные проблемы. Данные являются важной движущей силой принятия бизнес-решений, но традиционно компании использовали данные из различных источников, таких как отзывы клиентов, сотрудников и финансов. Исследования в области машинного обучения автоматизируют и оптимизируют этот процесс. Используя ПО, которое анализирует очень большие объемы данных на высокой скорости, компании могут быстрее достигать результатов. Машинное обучение применяется в здравоохранении и научно-исследовательских проектах, в финансовой сфере, на производствах и т.д., что говорит о высокой степени его актуальности [1].

Для решения задач машинного обучения в рамках курса практикума по программированию использовались следующие инструменты:

– *scikit-learn* - библиотека, предназначенная для машинного обучения, написанная на языке программирования *Python* и распространяемая в виде свободного программного обеспечения [2]. В её состав входят различные алгоритмы, в том числе предназначенные для задач классификации, регрессионного и кластерного анализа данных, включая метод опорных векторов, метод случайного леса, алгоритм усиления градиента, метод k-средних и

DBSCAN.

– *TensorFlow* - открытая программная библиотека для машинного обучения, разработанная компанией *Google* для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия [3].

– *CatBoost* — открытая программная библиотека, разработанная компанией Яндекс и реализующая уникальный патентованный алгоритм построения моделей машинного обучения, использующий одну из оригинальных схем градиентного бустинга [4].

– *XGBoost* – открытая программная библиотека, в основе которой лежит алгоритм градиентного бустинга деревьев решений — техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. [5].

– *Streamlit* – фреймворк для создания информационных панелей, часто применяющийся при работе с машинным обучением или большими данными для удобной визуализации.

1 Подготовка моделей

1.1 Классическая модель

В качестве примера классической модели машинного обучения было использовано дерево принятия решений *DecisionTreeRegressor* из пакета *scikit-learn*. Это дерево, в листьях которого стоят значения целевой функции, а в остальных узлах — условия перехода, определяющие по какому из ребер идти. Если для данного наблюдения условие истина, то осуществляется переход по левому ребру, если же ложь — по правому [6]. У выбранного экземпляра модели значение коэффициента детерминации приблизительно равняется 0.973, очень близкое к единице, при этом средняя величина ошибки составляет 30%. Модель была сериализована с помощью библиотеки *pickle*.

1.2 Ансамблевая модель

Из ансамблевых моделей был выбран *BaggingRegressor* из пакета *scikit-learn*. В ней используются деревья принятия решений небольшой глубины, ответы которых усредняются, что снижает дисперсию и предотвращает переобучение. Коэффициент детерминации у данной модели равен 0.98, а ошибка 26%, заметно лучше, чем у одного дерева. Для сериализации была применена библиотека *pickle*.

1.3 Глубокая полносвязная нейронная сеть

Нейронная сеть была создана с помощью *TensorFlow*. Коэффициент детерминации 0.917, а ошибка всего 10%. Сериализация была выполнена с помощью встроенных методов библиотеки. Структура дерева представлена на рисунке 1.

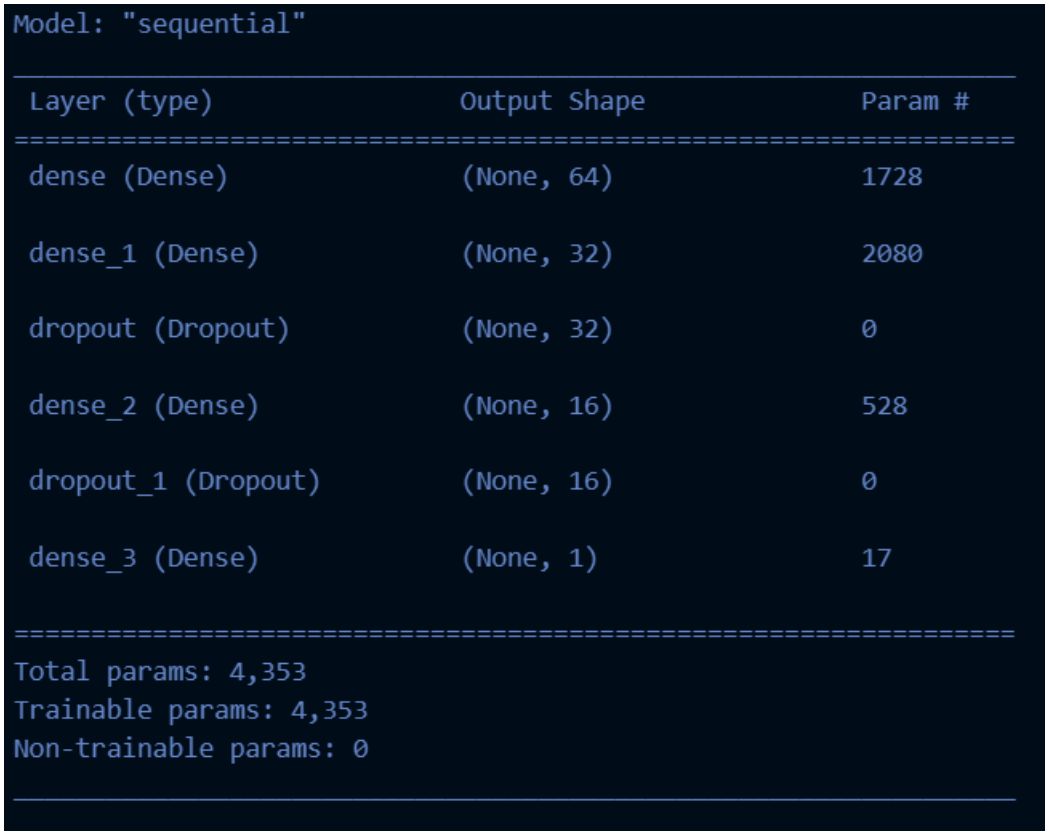

Рисунок 1 – Слои нейронной сети

2 Разработка дашборда

Для создания дашборда была использована библиотека *streamlit*. Чтобы создать многостраничное приложение, необходимо было создать стартовую страницу, и папку «*pages*» для трех страниц, необходимых для выполнения задания, которые подключились автоматически. На главной странице показаны основная информация о расчетно-графической работе и меню программы. На странице «Датасет» происходит загрузка заранее подготовленного датасета и его отображение, а также основная информация о признаках в датасете, целевом признаке и процессе предобработки. Страница «Визуализация» демонстрирует 4 вида графиков на основе датасета: ящик с усами, гистограмма, круговая диаграмма и тепловая карта. Страница “Предсказания” представляет собой набор ползунков, с помощью которых происходит выбор числового или категориального значения каждого признака. Также в этом файле подгружаются заранее заготовленные модели из пункта 1. При нажатии на кнопку значения

собираются в датафрейм, который подается моделям, и возвращается предсказанный ответ с перечислением всех выбранных признаков. Структура проекта представлена на рисунке 2.

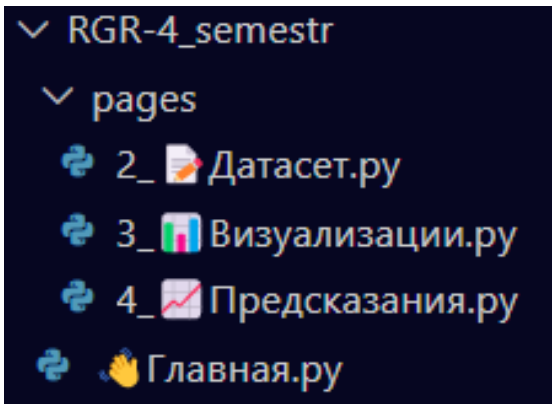

Рисунок 2 – Страницы дашборда

ЗАКЛЮЧЕНИЕ

В рамках задания был выбран набор данных для регрессии, заранее предобработанный и сохраненный в процессе лабораторных работ. Далее для этого набора были выбраны на основе значений коэффициента детерминации три модели разных видов, которые были разными средствами сериализованы. Был разработан дашборд, включающий в себя 4 страницы. В дашборде продемонстрированы навыки работы с графиками и схемами, с датафреймами и наборами данных, а также подключены сохраненные модели для предсказания значений по введенным пользователем данным. На протяжении всего процесса разработки расчетно-графической работы изучались новые библиотеки и их методы, а также научные работы или веб-сайты с информацией о моделях машинного обучения и глубоких нейронных сетях.

.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Что такое машинное обучение? Описание руководства по корпоративному машинному обучению URL: https://aws.amazon.com/ru/what-is/machine-learning/ (дата обращения: 07.05.23).

2 Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort [и др.]. scikit-learn: Machine Learning in Python // *Journal of Machine Learning Research*. 2011. № 12. С. 2825-2830.

3 tensorflow/tensorflow: An Open Source Machine Learning Framework for Everyone URL: https://github.com/tensorflow/tensorflow (дата обращения: 07.05.23).

4 Catboost URL: https://catboost.ai/en/docs/concepts/python-quickstart (дата обращения: 07.05.23).

5 XGBoost – Викиконспекты URL: https://neerc.ifmo.ru/wiki/index.php?title=XGBoostart (дата обращения: 07.05.23).

6 Классификация и регрессия с помощью деревьев принятия решений URL: https://habr.com/ru/articles/116385/ (дата обращения: 08.05.23).

ПРИЛОЖЕНИЕ А
(обязательное)

Скриншоты программы

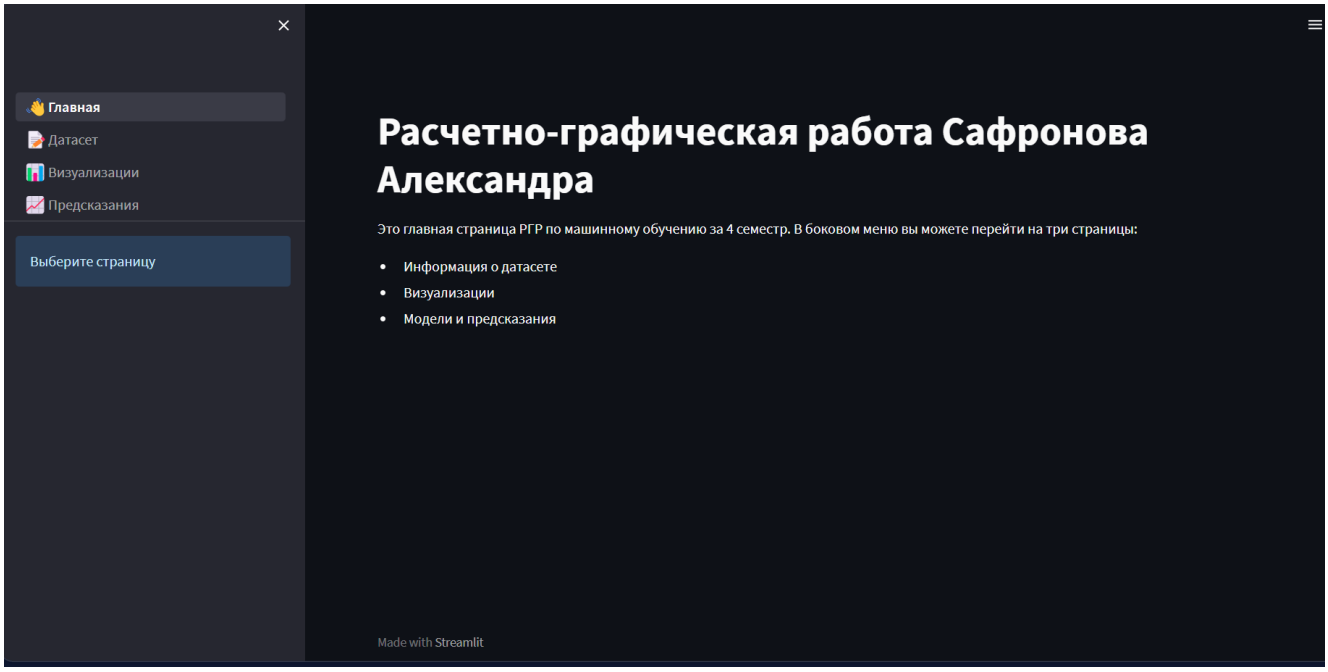

Рисунок 3 – Главная страница и меню

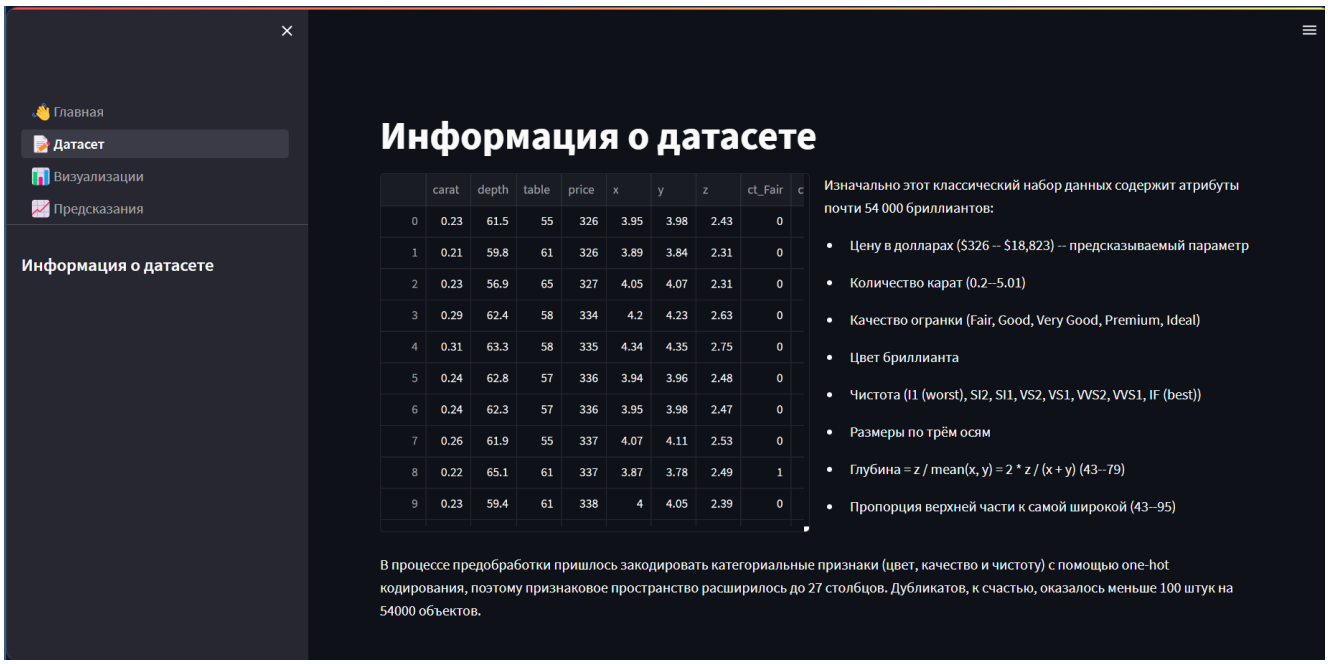

Рисунок 4 – Страница с датасетом

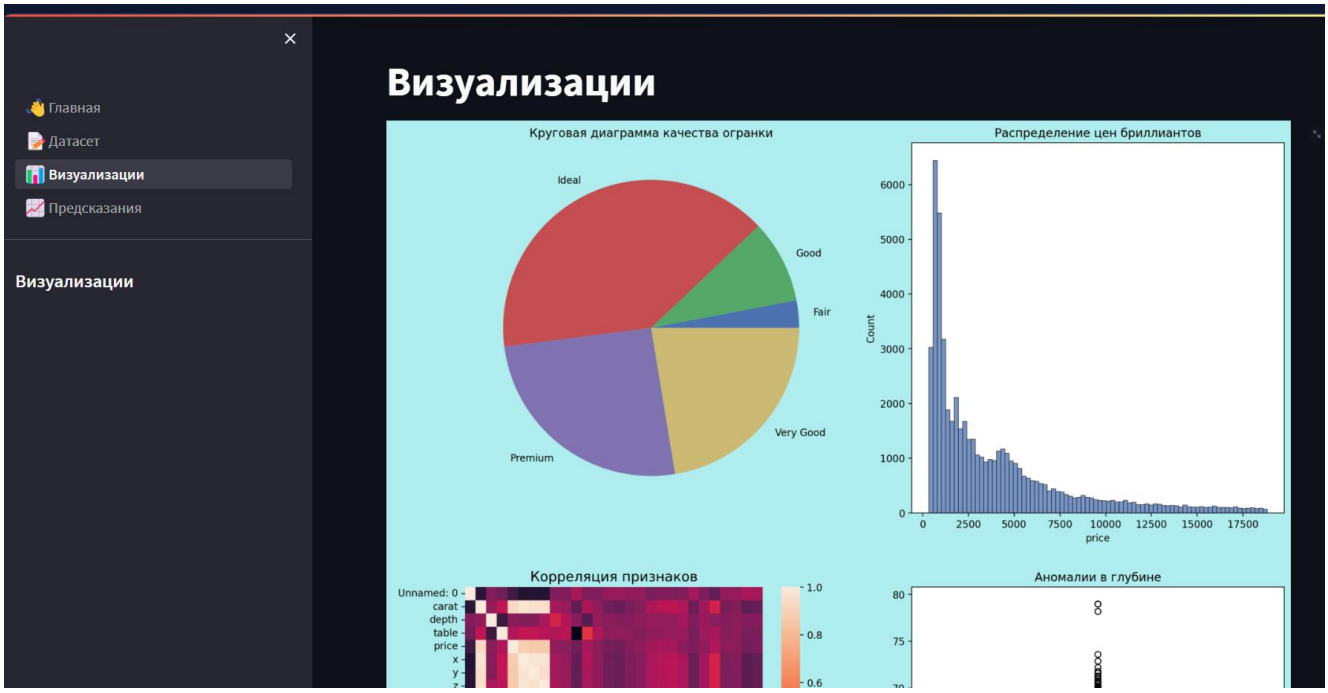

Рисунок 5 – страница с графиками

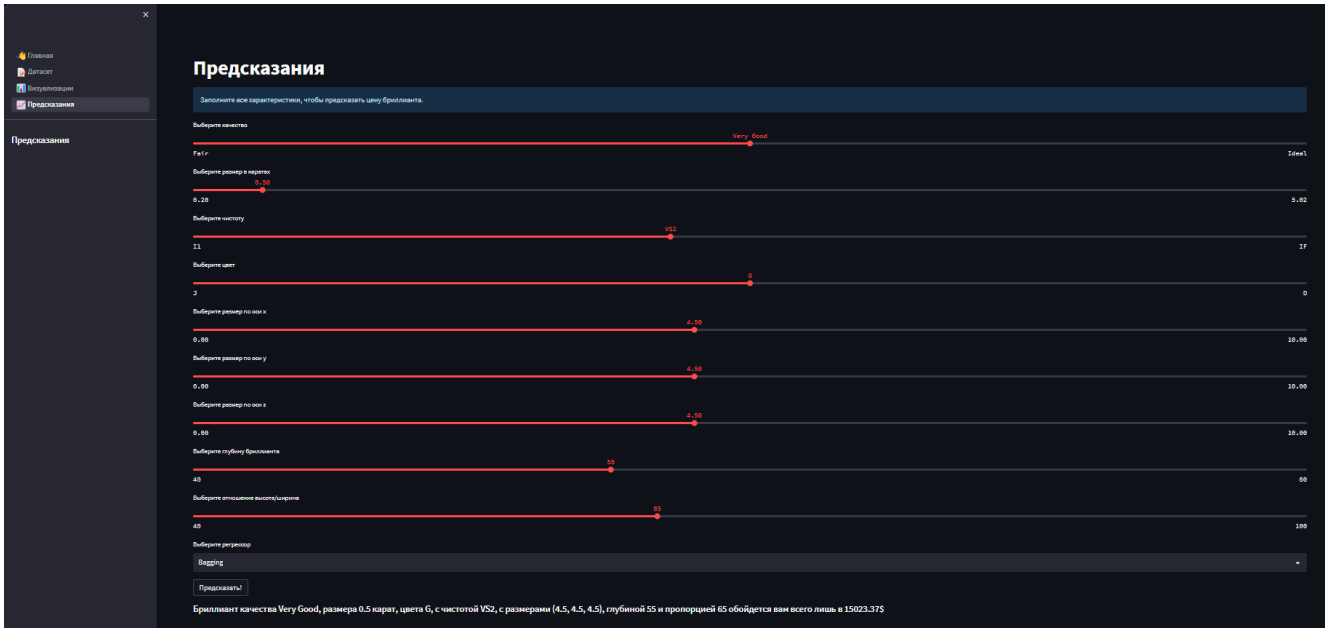

Рисунок 6 – страница с предсказанием

ПРИЛОЖЕНИЕ Б
(обязательное)

Листинг программы

Файл «Главная.ру».

```
import streamlit as st

st.set_page_config(layout="wide")

st.write("# Расчетно-графическая работа Сафронова Александра")

st.sidebar.info("Выберите страницу")

st.markdown(
    """
    Это главная страница РГР по машинному обучению за 4 семестр.
    В боковом меню вы можете перейти на три страницы:
    * Информация о датасете
    * Визуализации
    * Модели и предсказания
    """
)
```

Файл «Датасет.ру».

```
import streamlit as st
import pandas as pd

st.set_page_config(layout="wide")

st.write("# Информация о датасете")

st.sidebar.header("Информация о датасете")

df = pd.read_csv("data\\regression\\diamonds_prepared.csv")
df.drop(["Unnamed: 0"], axis=1, inplace=True)

col1, col2 = st.columns(2)

col1.dataframe(df)

col2.markdown(
    """
    Изначально этот классический набор данных содержит атрибуты почти 54 000
бриллиантов:
    * Цену в долларах (\$326 -- \$18,823) -- предсказываемый параметр

    * Количество карат (0.2--5.01)

    * Качество огранки (Fair, Good, Very Good, Premium, Ideal)

    * Цвет бриллианта
```

```
    * Чистота (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

    * Размеры по трём осям

    * Глубина = z / mean(x, y) = 2 * z / (x + y) (43--79)

    * Пропорция верхней части к самой широкой (43--95)
    """
)

st.markdown(
    """
    В процессе предобработки пришлось закодировать категориальные признаки (цвет, качество и чистоту) с помощью one-hot кодирования, поэтому признаковое пространство расширилось до 27 столбцов. Дубликатов, к счастью, оказалось меньше 100 штук на 54000 объектов.
    """
)
```

Файл «Визуализации.py».

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

st.set_page_config(layout="wide")

st.write("# Визуализации")

st.sidebar.header("Визуализации")

df = pd.read_csv("data\\regression\\diamonds.csv")
dfp = pd.read_csv("data\\regression\\diamonds_prepared.csv")
df.drop(["Unnamed: 0"], axis=1, inplace=True)

plt.style.use('seaborn-v0_8-deep')
fig, axes = plt.subplots(2, 2, figsize=(15, 15))
fig.set_facecolor('paleturquoise')
labels, counts = np.unique(df['cut'], return_counts=True)
axes[0][0].set_title('Круговая диаграмма качества огранки')
axes[0][0].pie(counts, labels=labels)
axes[1][0].set_title('Корреляция признаков', fontsize=14)
axes[0][1].set_title('Распределение цен бриллиантов')
sns.histplot(ax=axes[0][1], data=df['price'])
corr = dfp.corr()
sns.heatmap(corr, ax=axes[1][0], xticklabels=corr.columns,
yticklabels=corr.columns, annot_kws={"size":10}) # type: ignore
axes[1][1].boxplot(x=df['depth'])
axes[1][0].set_title('Корреляция признаков', fontsize=14)
axes[1][1].set_title('Аномалии в глубине')
st.pyplot(fig)
```

Файл «Предсказания.py».

```
import pickle
import random

import numpy as np
import pandas as pd
from sklearn import model_selection
import streamlit as st
import tensorflow as tf

st.set_page_config(layout="wide")

st.write("# Предсказания")

st.sidebar.header("Предсказания")

st.info(
    """
    Заполните _все_ характеристики, чтобы предсказать цену бриллианта.
    """
)

cut = st.select_slider(
    "Выберите качество",
    ("Fair", "Good", "Very Good", "Premium", "Ideal"),
    "Very Good",
)

carat = st.slider("Выберите размер в каратах", 0.2, 5.02, 0.5)

clarity = st.select_slider(
    "Выберите чистоту",
    ("I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF"),
    "VS2",
)

color = st.select_slider("Выберите цвет", ("J", "I", "H", "G", "F", "E", "D"),
"G")

x = st.slider("Выберите размер по оси x", 0.0, 10.0, 4.5)

y = st.slider("Выберите размер по оси y", 0.0, 10.0, 4.5)

z = st.slider("Выберите размер по оси z", 0.0, 10.0, 4.5)

depth = st.slider("Выберите глубину бриллианта", 40, 80, 55)

table = st.slider("Выберите отношение высота/ширина", 40, 100, 65)

regressor = st.selectbox(
    "Выберите регрессор", ("DecisionTree", "Bagging", "TensorFlow"), 0
)

if st.button("Предсказать!"):
    columns = [
        "carat",
```

```
        "depth",
        "table",
        "x",
        "y",
        "z",
        "ct_Fair",
        "ct_Good",
        "ct_Ideal",
        "ct_Premium",
        "ct_Very Good",
        "clr_D",
        "clr_E",
        "clr_F",
        "clr_G",
        "clr_H",
        "clr_I",
        "clr_J",
        "clrty_I1",
        "clrty_IF",
        "clrty_SI1",
        "clrty_SI2",
        "clrty_VS1",
        "clrty_VS2",
        "clrty_VVS1",
        "clrty_VVS2",
    ]
    df = pd.DataFrame(
        [[carat, depth, table, x, y, z, *[0 for i in range(20)]]],
columns=columns
    )
    for a in df.columns[7:11]:
        if cut in a:
            df[a] = 1
    for a in df.columns[12:18]:
        if color in a:
            df[a] = 1
    for a in df.columns[19:26]:
        if clarity == a[6:]:
            df[a] = 1
    models = {
        "DecisionTree": "models/decisiontreeregressor.pickle",
        "Bagging": "models/baggingregressor.pickle",
        "TensorFlow": "models/tensorflowregressor.h5",
    }
    prediction = random.random() * 100 + random.random() * 10 + random.random()
    if regressor != "TensorFlow":
        with open(models.get(regressor), "rb") as f:  # type: ignore
            model = pickle.load(f)
            model_columns = model.feature_names_in_
            df.columns = model_columns
    else:
        model = tf.keras.models.load_model(models.get(regressor))
    prediction = float(model.predict(df))  # type: ignore

    st.markdown(
        f"##### Бриллиант качества {cut}, размера {carat} карат, цвета {color}, с
```

```
чистотой {clarity}, с размерами {x, y, z}, глубиной {depth} и пропорцией {table}
обойдется вам всего лишь в {np.around(prediction, 2)}\$")
```