**index**

объясни код import streamlit as st

PAGE\_LAYOUT = "wide"

PAGE\_TITLE = "# Расчетно-графическая работа"

SIDEBAR\_INFO = "Выберите страницу"

PAGE\_TEXT = """

Это главная страница РГР по машинному обучению.

В боковом меню вы можете перейти на три страницы:

\* Информация о датасете

\* Визуализации

\* Модели и предсказания

"""

st.set\_page\_config(layout=PAGE\_LAYOUT)

st.write(PAGE\_TITLE)

st.sidebar.info(SIDEBAR\_INFO)

st.markdown(PAGE\_TEXT)

Этот код написан на языке Python и использует библиотеку Streamlit для создания веб-приложения. В коде определяются несколько переменных, таких как PAGE\_LAYOUT, PAGE\_TITLE, SIDEBAR\_INFO и PAGE\_TEXT, которые содержат информацию о конфигурации страницы, заголовке, информации в боковой панели и тексте на странице соответственно.

Далее вызывается функция set\_page\_config из библиотеки Streamlit, чтобы установить конфигурацию страницы с использованием значения переменной PAGE\_LAYOUT. Затем вызывается функция write для вывода заголовка страницы, используя значение переменной PAGE\_TITLE. Функция info из боковой панели (sidebar) используется для вывода информации в боковой панели с использованием значения переменной SIDEBAR\_INFO. Наконец, функция markdown используется для вывода текста на странице с использованием значения переменной PAGE\_TEXT.

В общем, этот код создает простое веб-приложение с использованием библиотеки Streamlit, которое содержит заголовок, информацию в боковой панели и текст на главной странице.

**dataset**

объясни код import pandas as pd

import streamlit as st

PAGE\_LAYOUT = "wide"

PAGE\_TITLE = "# Информация о датасете"

SIDEBAR\_HEADER = "Информация о датасете"

DATA\_FILE = "data\\regression\\diamonds\_prepared.csv"

COLUMN\_TO\_DROP = "Unnamed: 0"

COLUMN\_DROP\_AXIS = 1

COLUMN1\_TEXT = """

Изначально этот классический набор данных содержит атрибуты 53940 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)

"""

PAGE\_TEXT = """

В процессе предобработки пришлось закодировать категориальные признаки (цвет, качество и чистоту) с помощью one-hot кодирования, поэтому признаковое пространство расширилось до 27 столбцов.

"""

st.set\_page\_config(layout=PAGE\_LAYOUT)

st.write(PAGE\_TITLE)

st.sidebar.header(SIDEBAR\_HEADER)

df = pd.read\_csv(DATA\_FILE)

df.drop([COLUMN\_TO\_DROP], axis=COLUMN\_DROP\_AXIS, inplace=True)

col1, col2 = st.columns(2)

col1.dataframe(df)

col2.markdown(COLUMN1\_TEXT)

st.markdown(PAGE\_TEXT)

Этот код также написан на языке Python и использует библиотеки Streamlit и Pandas. В коде импортируются эти две библиотеки и определяются несколько переменных, таких как PAGE\_LAYOUT, PAGE\_TITLE, SIDEBAR\_HEADER, DATA\_FILE, COLUMN\_TO\_DROP, COLUMN\_DROP\_AXIS, COLUMN1\_TEXT и PAGE\_TEXT. Они содержат информацию о конфигурации страницы, заголовке, заголовке боковой панели, файле данных, столбце для удаления, оси удаления столбца, тексте в первой колонке и тексте на странице соответственно.

Далее вызывается функция set\_page\_config из библиотеки Streamlit, чтобы установить конфигурацию страницы с использованием значения переменной PAGE\_LAYOUT. Затем вызывается функция write для вывода заголовка страницы, используя значение переменной PAGE\_TITLE. Функция header из боковой панели (sidebar) используется для вывода заголовка в боковой панели с использованием значения переменной SIDEBAR\_HEADER.

Затем создается объект DataFrame из библиотеки Pandas с использованием функции read\_csv и файла данных, указанного в переменной DATA\_FILE. Вызывается метод drop у объекта DataFrame для удаления столбца, указанного в переменной COLUMN\_TO\_DROP, по оси, указанной в переменной COLUMN\_DROP\_AXIS.

Далее создается две колонки с помощью функции columns из библиотеки Streamlit. В первой колонке отображается DataFrame с помощью метода dataframe. Во второй колонке отображается текст с помощью метода markdown и значения переменной COLUMN1\_TEXT.

Наконец, вызывается функция markdown для вывода текста на странице с использованием значения переменной PAGE\_TEXT.

В общем, этот код создает страницу веб-приложения с использованием библиотек Streamlit и Pandas, которая содержит заголовок, информацию в боковой панели, данные из файла CSV и текст на странице.

**visualizations**

объясни код import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

import streamlit as st

PAGE\_LAYOUT = "wide"

PAGE\_TITLE = "# Визуализации"

SIDEBAR\_HEADER = "Визуализации"

DATA\_FILE = "data\\regression\\diamonds.csv"

DATA\_FILE\_PREPARED = "data\\regression\\diamonds\_prepared.csv"

COLUMN\_TO\_DROP = "Unnamed: 0"

COLUMN\_DROP\_AXIS = 1

STYLE = 'seaborn-v0\_8-deep'

FIG\_SIZE = (15, 15)

FIG\_FACECOLOR = 'paleturquoise'

PIE\_TITLE = 'Круговая диаграмма качества огранки'

HIST\_TITLE = 'Распределение цен бриллиантов'

HIST\_DATA\_COLUMN = 'price'

HEATMAP\_TITLE = 'Корреляция признаков'

BOXPLOT\_TITLE = 'Аномалии в глубине'

BOXPLOT\_DATA\_COLUMN = 'depth'

st.set\_page\_config(layout=PAGE\_LAYOUT)

st.write(PAGE\_TITLE)

st.sidebar.header(SIDEBAR\_HEADER)

df = pd.read\_csv(DATA\_FILE)

dfp = pd.read\_csv(DATA\_FILE\_PREPARED)

df.drop([COLUMN\_TO\_DROP], axis=COLUMN\_DROP\_AXIS, inplace=True)

plt.style.use(STYLE)

fig, axes = plt.subplots(2, 2, figsize=FIG\_SIZE)

fig.set\_facecolor(FIG\_FACECOLOR)

labels, counts = np.unique(df['cut'], return\_counts=True)

axes[0][0].set\_title(PIE\_TITLE)

axes[0][0].pie(counts, labels=labels)

axes[1][0].set\_title(HEATMAP\_TITLE, fontsize=14)

axes[0][1].set\_title(HIST\_TITLE)

sns.histplot(ax=axes[0][1], data=df[HIST\_DATA\_COLUMN])

corr = dfp.corr()

sns.heatmap(corr, ax=axes[1][0], xticklabels=corr.columns, yticklabels=corr.columns, annot\_kws={"size":10})

axes[1][1].boxplot(x=df[BOXPLOT\_DATA\_COLUMN])

axes[1][0].set\_title(HEATMAP\_TITLE, fontsize=14)

axes[1][1].set\_title(BOXPLOT\_TITLE)

st.pyplot(fig)

Этот код использует библиотеки matplotlib.pyplot, numpy, pandas, seaborn и streamlit для создания визуализаций и отображения их в веб-приложении Streamlit.

В первых пяти строках кода импортируются необходимые библиотеки. Затем определяются константы, которые будут использоваться в коде, такие как заголовки страницы и боковой панели, пути к файлам с данными, стиль графиков и размеры рисунков.

В строке с вызовом функции st.set\_page\_config устанавливается конфигурация страницы Streamlit с использованием значения константы PAGE\_LAYOUT. В следующих двух строках выводятся заголовки страницы и боковой панели с использованием значений констант PAGE\_TITLE и SIDEBAR\_HEADER.

Затем с помощью метода read\_csv класса pandas загружаются данные из файлов, указанных в константах DATA\_FILE и DATA\_FILE\_PREPARED, и сохраняются в переменных df и dfp соответственно. Из DataFrame df удаляется столбец, указанный в константе COLUMN\_TO\_DROP.

Далее устанавливается стиль графиков с использованием значения константы STYLE и создается рисунок с четырьмя подграфиками с использованием значений констант FIG\_SIZE и FIG\_FACECOLOR. На первом подграфике создается круговая диаграмма, которая отображает распределение значений столбца ‘cut’ DataFrame df. На втором подграфике создается гистограмма, которая отображает распределение значений столбца, указанного в константе HIST\_DATA\_COLUMN, DataFrame df. На третьем подграфике создается тепловая карта, которая отображает корреляцию между признаками DataFrame dfp. На четвертом подграфике создается ящик с усами, который отображает распределение значений столбца, указанного в константе BOXPLOT\_DATA\_COLUMN, DataFrame df.

В последней строке кода вызывается метод pyplot объекта st с аргументом fig, чтобы отобразить созданный рисунок в веб-приложении Streamlit.

Таким образом, этот код создает ряд визуализаций на основе данных из файлов и отображает их в веб-приложении Streamlit.

**predicions**

объясни код import pickle

import random

import numpy as np

import pandas as pd

from sklearn import \*

import streamlit as st

import tensorflow as tf

PAGE\_LAYOUT = "wide"

PAGE\_TITLE = "# Предсказания"

SIDEBAR\_HEADER = "Предсказания"

INFO\_TEXT = """

Заполните характеристики, чтобы предсказать цену бриллианта.

"""

CUT\_LABEL = "Выберите качество"

CUT\_OPTIONS = ("Fair", "Good", "Very Good", "Premium", "Ideal")

CUT\_DEFAULT = "Very Good"

CARAT\_LABEL = "Выберите размер в каратах"

CARAT\_MIN = 0.2

CARAT\_MAX = 5.02

CARAT\_DEFAULT = 0.5

CLARITY\_LABEL = "Выберите чистоту"

CLARITY\_OPTIONS = ("I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF")

CLARITY\_DEFAULT = "VS2"

COLOR\_LABEL = "Выберите цвет"

COLOR\_OPTIONS = ("J", "I", "H", "G", "F", "E", "D")

COLOR\_DEFAULT = "G"

X\_LABEL = "Выберите размер по оси x"

X\_MIN = 0.0

X\_MAX = 10.0

X\_DEFAULT = 4.5

Y\_LABEL = "Выберите размер по оси y"

Y\_MIN = 0.0

Y\_MAX = 10.0

Y\_DEFAULT = 4.5

Z\_LABEL = "Выберите размер по оси z"

Z\_MIN = 0.0

Z\_MAX = 10.0

Z\_DEFAULT = 4.5

DEPTH\_LABEL = "Выберите глубину бриллианта"

DEPTH\_MIN = 40

DEPTH\_MAX = 80

DEPTH\_DEFAULT = 55

TABLE\_LABEL = "Выберите отношение высота/ширина"

TABLE\_MIN = 40

TABLE\_MAX = 100

TABLE\_DEFAULT = 65

REGRESSOR\_LABEL = "Выберите регрессор"

REGRESSOR\_OPTIONS = ("DecisionTree", "Bagging", "TensorFlow")

REGRESSOR\_DEFAULT\_INDEX=0

st.set\_page\_config(layout=PAGE\_LAYOUT)

st.write(PAGE\_TITLE)

st.sidebar.header(SIDEBAR\_HEADER)

st.info(INFO\_TEXT)

cut=st.select\_slider(CUT\_LABEL, CUT\_OPTIONS, CUT\_DEFAULT)

carat=st.slider(CARAT\_LABEL, CARAT\_MIN, CARAT\_MAX, CARAT\_DEFAULT)

clarity=st.select\_slider(CLARITY\_LABEL, CLARITY\_OPTIONS, CLARITY\_DEFAULT)

color=st.select\_slider(COLOR\_LABEL, COLOR\_OPTIONS, COLOR\_DEFAULT)

x=st.slider(X\_LABEL, X\_MIN, X\_MAX, X\_DEFAULT)

y=st.slider(Y\_LABEL, Y\_MIN, Y\_MAX, Y\_DEFAULT)

z=st.slider(Z\_LABEL, Z\_MIN, Z\_MAX, Z\_DEFAULT)

depth=st.slider(DEPTH\_LABEL, DEPTH\_MIN, DEPTH\_MAX, DEPTH\_DEFAULT)

table=st.slider(TABLE\_LABEL, TABLE\_MIN, TABLE\_MAX, TABLE\_DEFAULT)

regressor = st.selectbox(REGRESSOR\_LABEL, REGRESSOR\_OPTIONS)

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:

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))

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)}\$")

Сообщение отправлено. объясни код import pickle import random import numpy as np import pandas as pd from sklearn import \* import streamlit as st import tensorflow as tf PAGE\_LAYOUT = "wide" PAGE\_TITLE = "# Предсказания" SIDEBAR\_HEADER = "Предсказания" INFO\_TEXT = """ Заполните характеристики, чтобы предсказать цену бриллианта. """ CUT\_LABEL = "Выберите качество" CUT\_OPTIONS = ("Fair", "Good", "Very Good", "Premium", "Ideal") CUT\_DEFAULT = "Very Good" CARAT\_LABEL = "Выберите размер в каратах" CARAT\_MIN = 0.2 CARAT\_MAX = 5.02 CARAT\_DEFAULT = 0.5 CLARITY\_LABEL = "Выберите чистоту" CLARITY\_OPTIONS = ("I1", "SI2", "SI1", "VS2", "VS1", "VVS2", "VVS1", "IF") CLARITY\_DEFAULT = "VS2" COLOR\_LABEL = "Выберите цвет" COLOR\_OPTIONS = ("J", "I", "H", "G", "F", "E", "D") COLOR\_DEFAULT = "G" X\_LABEL = "Выберите размер по оси x" X\_MIN = 0.0 X\_MAX = 10.0 X\_DEFAULT = 4.5 Y\_LABEL = "Выберите размер по оси y" Y\_MIN = 0.0 Y\_MAX = 10.0 Y\_DEFAULT = 4.5 Z\_LABEL = "Выберите размер по оси z" Z\_MIN = 0.0 Z\_MAX = 10.0 Z\_DEFAULT = 4.5 DEPTH\_LABEL = "Выберите глубину бриллианта" DEPTH\_MIN = 40 DEPTH\_MAX = 80 DEPTH\_DEFAULT = 55 TABLE\_LABEL = "Выберите отношение высота/ширина" TABLE\_MIN = 40 TABLE\_MAX = 100 TABLE\_DEFAULT = 65 REGRESSOR\_LABEL = "Выберите регрессор" REGRESSOR\_OPTIONS = ("DecisionTree", "Bagging", "TensorFlow") REGRESSOR\_DEFAULT\_INDEX=0 st.set\_page\_config(layout=PAGE\_LAYOUT) st.write(PAGE\_TITLE) st.sidebar.header(SIDEBAR\_HEADER) st.info(INFO\_TEXT) cut=st.select\_slider(CUT\_LABEL, CUT\_OPTIONS, CUT\_DEFAULT) carat=st.slider(CARAT\_LABEL, CARAT\_MIN, CARAT\_MAX, CARAT\_DEFAULT) clarity=st.select\_slider(CLARITY\_LABEL, CLARITY\_OPTIONS, CLARITY\_DEFAULT) color=st.select\_slider(COLOR\_LABEL, COLOR\_OPTIONS, COLOR\_DEFAULT) x=st.slider(X\_LABEL, X\_MIN, X\_MAX, X\_DEFAULT) y=st.slider(Y\_LABEL, Y\_MIN, Y\_MAX, Y\_DEFAULT) z=st.slider(Z\_LABEL, Z\_MIN, Z\_MAX, Z\_DEFAULT) depth=st.slider(DEPTH\_LABEL, DEPTH\_MIN, DEPTH\_MAX, DEPTH\_DEFAULT) table=st.slider(TABLE\_LABEL, TABLE\_MIN, TABLE\_MAX, TABLE\_DEFAULT) regressor = st.selectbox(REGRESSOR\_LABEL, REGRESSOR\_OPTIONS) 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: 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)) 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)}\$")

Этот код написан на языке Python и использует библиотеки Pickle, Random, Numpy, Pandas, Scikit-learn, Streamlit и TensorFlow. В коде импортируются эти библиотеки и определяются несколько переменных, таких как PAGE\_LAYOUT, PAGE\_TITLE, SIDEBAR\_HEADER, INFO\_TEXT, CUT\_LABEL, CUT\_OPTIONS, CUT\_DEFAULT, CARAT\_LABEL, CARAT\_MIN, CARAT\_MAX, CARAT\_DEFAULT, CLARITY\_LABEL, CLARITY\_OPTIONS, CLARITY\_DEFAULT, COLOR\_LABEL, COLOR\_OPTIONS, COLOR\_DEFAULT, X\_LABEL, X\_MIN, X\_MAX, X\_DEFAULT, Y\_LABEL, Y\_MIN, Y\_MAX, Y\_DEFAULT, Z\_LABEL, Z\_MIN, Z\_MAX, Z\_DEFAULT, DEPTH\_LABEL, DEPTH\_MIN, DEPTH\_MAX, DEPTH\_DEFAULT,TABLE\_LABEL,TABLE\_MIN,TABLE\_MAX,TABLE\_DEFAULT,REGRESSOR\_LABEL,REGRESSOR\_OPTIONS иREGRESSOR\_DEFAULT\_INDEX. Они содержат информацию о конфигурации страницы, заголовке, заголовке боковой панели, информационном тексте, метках и параметрах для выбора качества огранки, размера в каратах, чистоты, цвета, размеров по осям x, y и z, глубины бриллианта и отношения высота/ширина соответственно.

Далее вызывается функцияset\_page\_config из библиотеки Streamlit, чтобы установить конфигурацию страницы с использованием значения переменнойPAGE\_LAYOUT. Затем вызывается функцияwrite для вывода заголовка страницы, используя значение переменнойPAGE\_TITLE. Функцияheader из боковой панели (sidebar) используется для вывода заголовка в боковой панели с использованием значения переменнойSIDEBAR\_HEADER.

Затем вызывается функцияinfo из библиотеки Streamlit для вывода информационного текста с использованием значения переменнойINFO\_TEXT.

Далее создается несколько виджетов с помощью функцийselect\_slider,slider иselectbox из библиотеки Streamlit для выбора значений качества огранки, размера в каратах, чистоты, цвета, размеров по осям x, y и z, глубины бриллианта и отношения высота/ширина соответственно.

Если пользователь нажимает кнопку “Предсказать!”, то создается объект DataFrame из библиотеки Pandas с использованием выбранных пользователем значений. Затем создается словарь с моделями регрессии и выбирается модель в зависимости от выбранного пользователем регрессора. Если регрессор не TensorFlow, то модель загружается с помощью библиотеки Pickle. Иначе модель загружается с помощью методаload\_model из библиотеки TensorFlow.

Затем вызывается методpredict у выбранной модели для предсказания цены бриллианта на основе выбранных пользователем значений. Вызывается функцияmarkdown из библиотеки Streamlit для вывода предсказанной цены.

В общем, этот код создает страницу веб-приложения с использованием библиотек Streamlit и других библиотек машинного обучения, которая содержит заголовок, информацию в боковой панели и виджеты для выбора значений характеристик бриллианта и предсказания его цены.

объясни код 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:

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))

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)}\$")

Сообщение отправлено. объясни код 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: 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)) 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)}\$")

Этот фрагмент кода написан на языке Python и использует библиотеки Pickle, Random, Numpy, Streamlit и TensorFlow. В коде создается словарь models, который содержит соответствие между именами моделей регрессии и путями к файлам с сохраненными моделями. Затем создается переменная prediction со случайным значением.

Далее проверяется, равно ли значение переменной regressor строке “TensorFlow”. Если нет, то выполняется блок кода внутри условного оператора. В этом блоке кода открывается файл с сохраненной моделью с помощью функции open и метода get у словаря models. Затем модель загружается с помощью библиотеки Pickle и метода load. Вызывается атрибут feature\_names\_in\_ у загруженной модели для получения списка имен признаков. Затем устанавливаются имена столбцов у объекта DataFrame df с использованием полученного списка имен признаков.

Если значение переменной regressor равно строке “TensorFlow”, то выполняется блок кода внутри ветви else. В этом блоке кода загружается модель с помощью метода load\_model из библиотеки TensorFlow и метода get у словаря models.

Затем вызывается метод predict у загруженной модели для предсказания цены бриллианта на основе данных из объекта DataFrame df. Вызывается функция markdown из библиотеки Streamlit для вывода предсказанной цены и выбранных пользователем значений характеристик бриллианта.

В общем, этот фрагмент кода загружает сохраненную модель регрессии в зависимости от выбранного пользователем регрессора и использует ее для предсказания цены бриллианта на основе выбранных пользователем значений характеристик.