AI VIETNAM
All-in-One Course

REVIEW LINEAR REGRESSION

Diamond Price Prediction

AIO2022

Outline

Introduction

Data

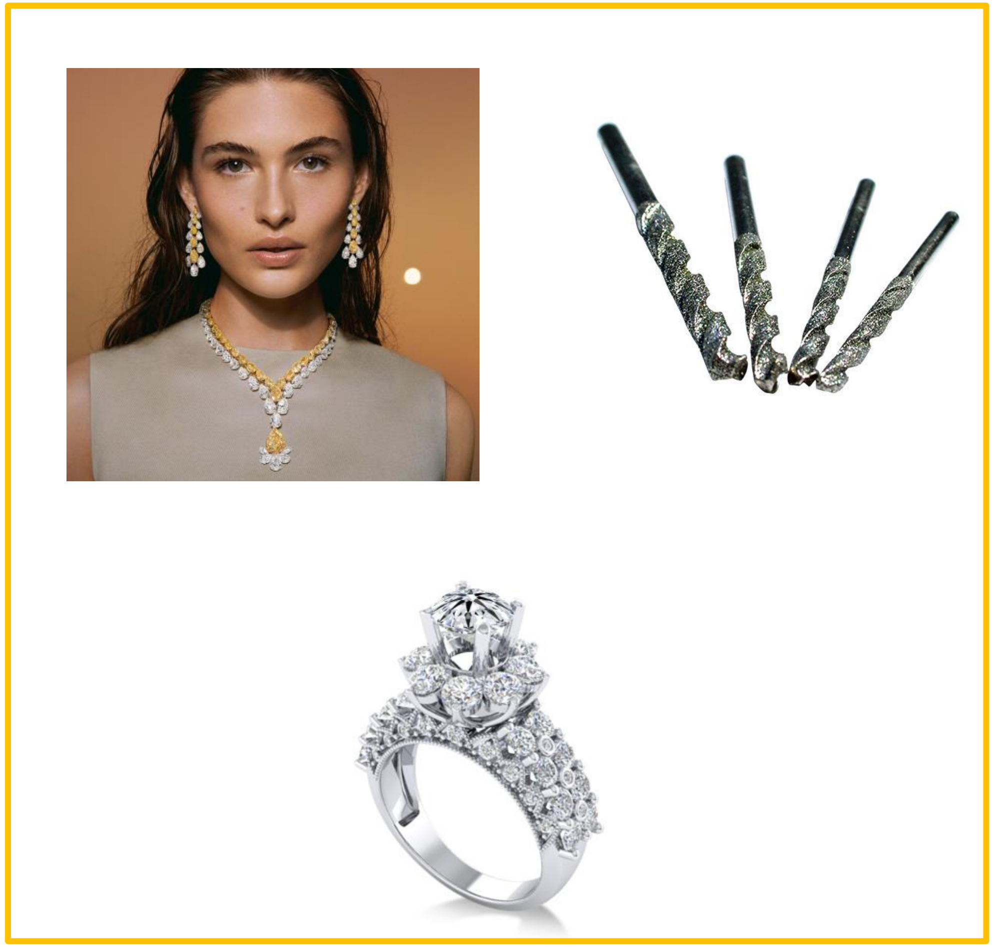
Modeling

Deployment

INTRODUCTION

Introduction

❖Diamond application



Ứng Dụng:

- Trang sức
- Công nghệ mài xén
- Công nghiệp
- Công nghệ điện tử….

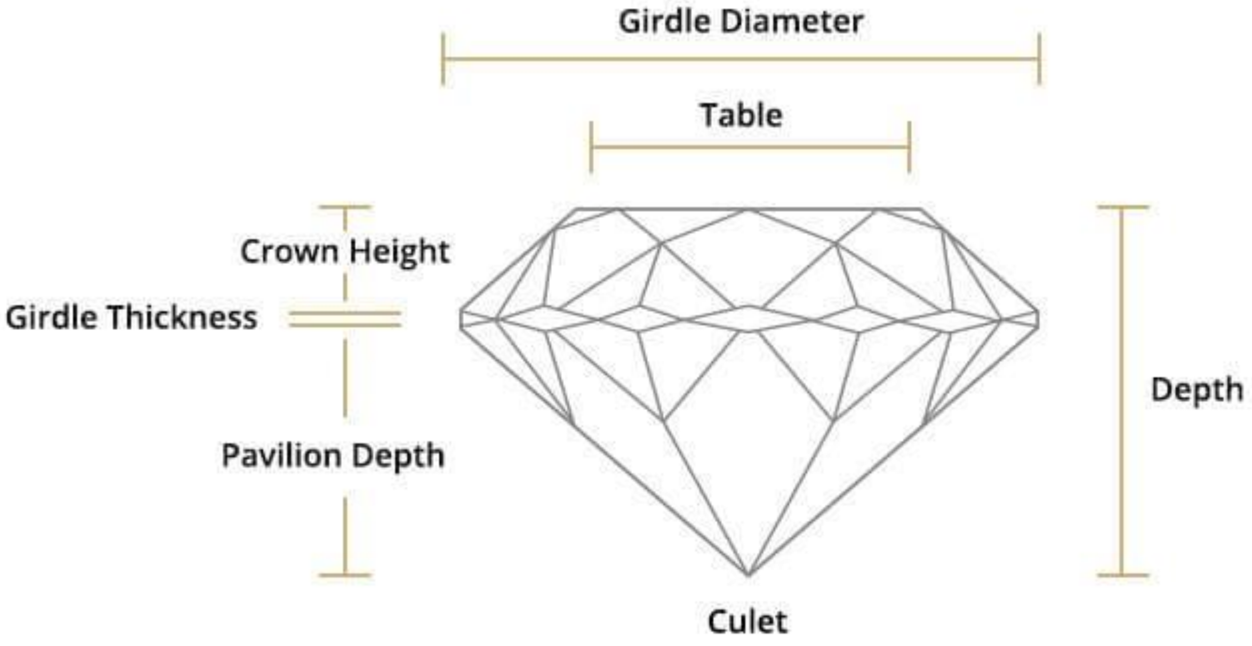
→ Giá trị cao

Introduction

❖How we value a diamond?

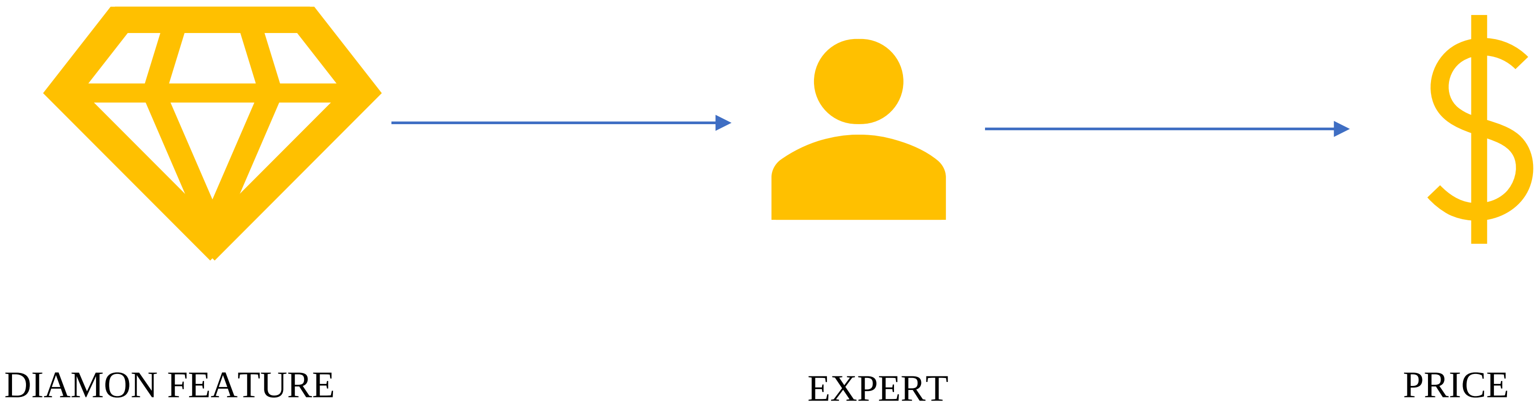
A diamond’s value is determined by its 4Cs:

- Color: how colorless the diamond is
- Carat: the weight of the diamond
- Cut: Quality of the angles, facets
- Clarity: how clean is the diamond
- Other features about the shape: depth, table, x, y, z



Introduction

❖Diamond Pricing Process



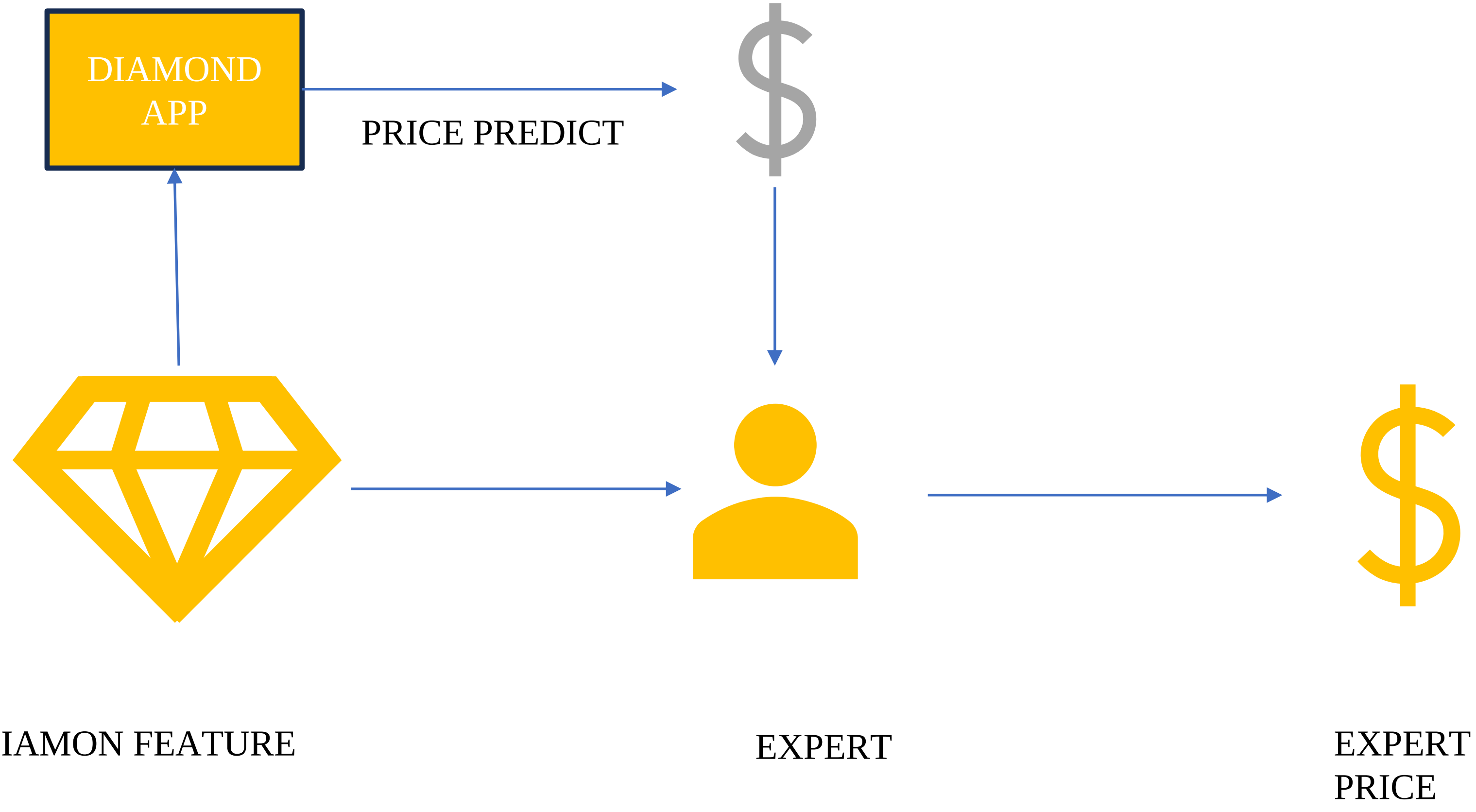
DIAMON FEATURE

EXPERT

PRICE

Introduction

❖Diamond Valuation Support Process



DIAMOND APP

PRICE PREDICT

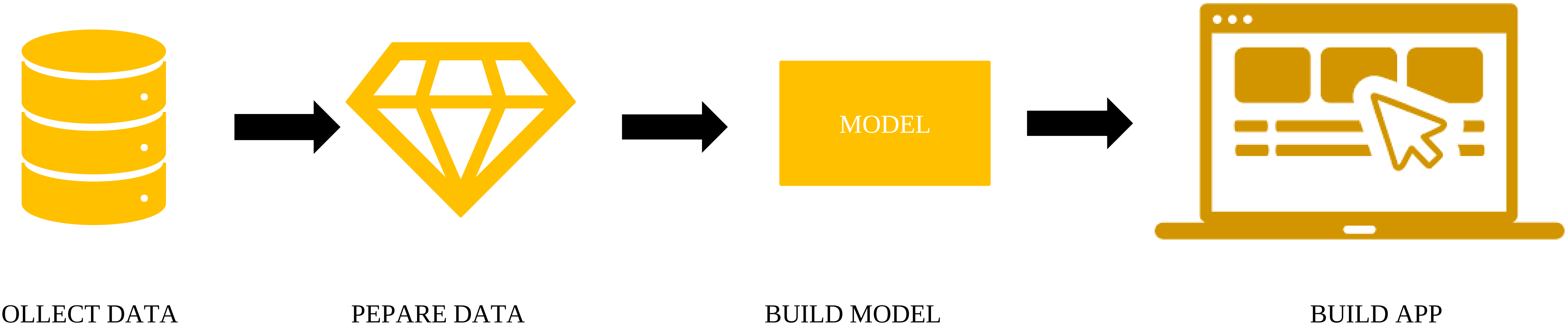
DIAMON FEATURE

EXPERT

EXPERT PRICE

Introduction

❖PIPELINE



MODEL

COLLECT DATA

PEPARE DATA

BUILD MODEL

BUILD APP

DATA

- ❑ DATA COLLECTION
- ❑ EXPLORE DATA
- ❑ DATA PREPARATION

DATA

❖DATA COLLECTION

This is a dataset that includes 9 observations about the characteristics of each unique diamond, as well as the price.

- Carat- Carat weight of the diamond
- Cut - The cut rating of the diamond
- Color - The color rating of the diamond
- Clarity - The clarity rating of the diamond
- Table - The table width of the diamond
- Depth- The percentage of depth of the diamond
- Price - The price (in USD) of the diamond
- X- X dimension of the diamond
- Y- Y dimension of the diamond
- Z- Z dimension of the diamond

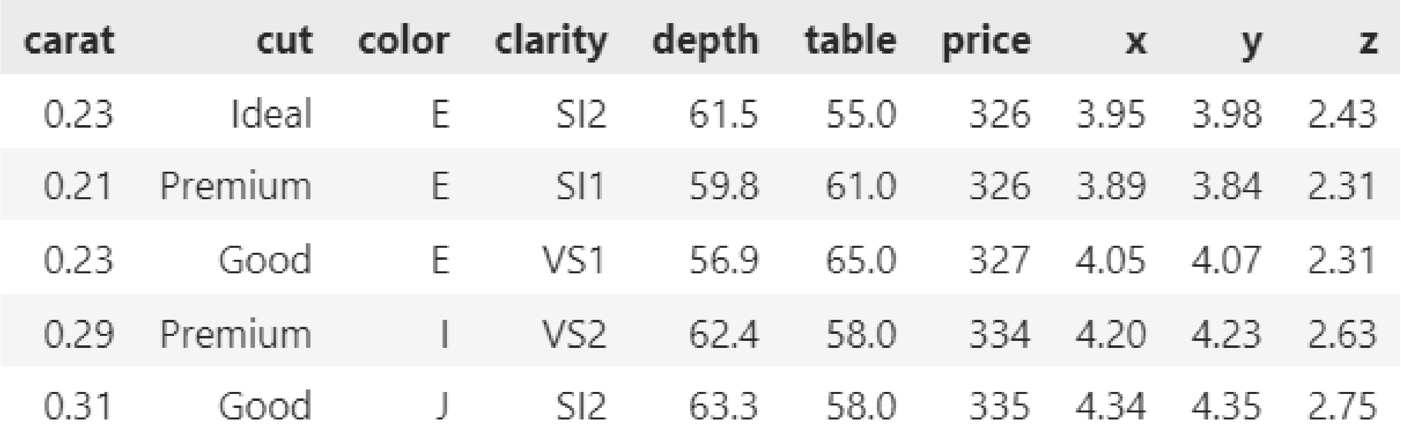
Dowload Data from diamonds | Kaggle

DATA

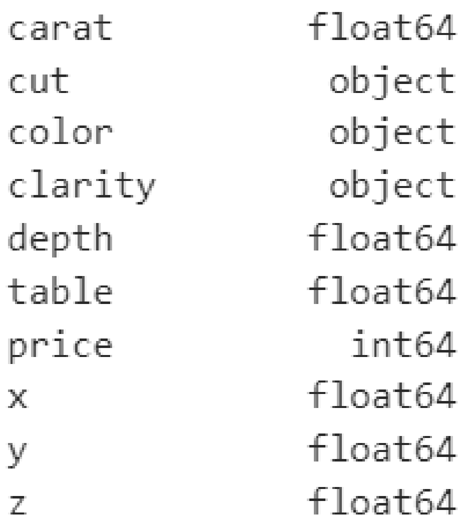
❖EXPLORE DATA

53940
samples

10
Column



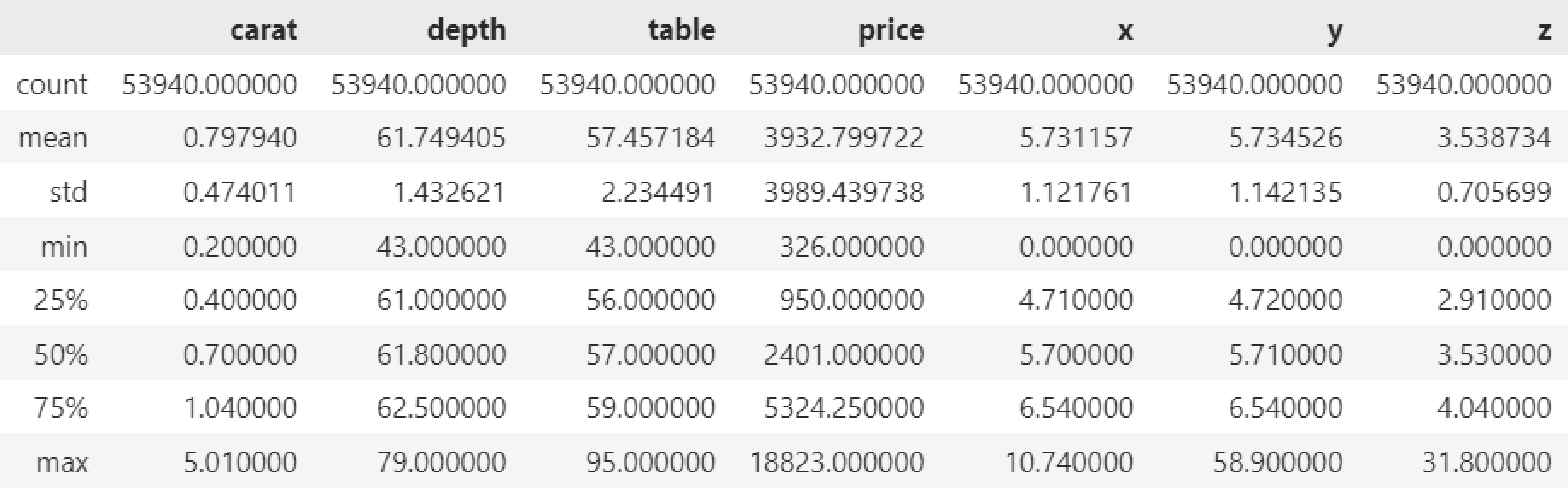
carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75



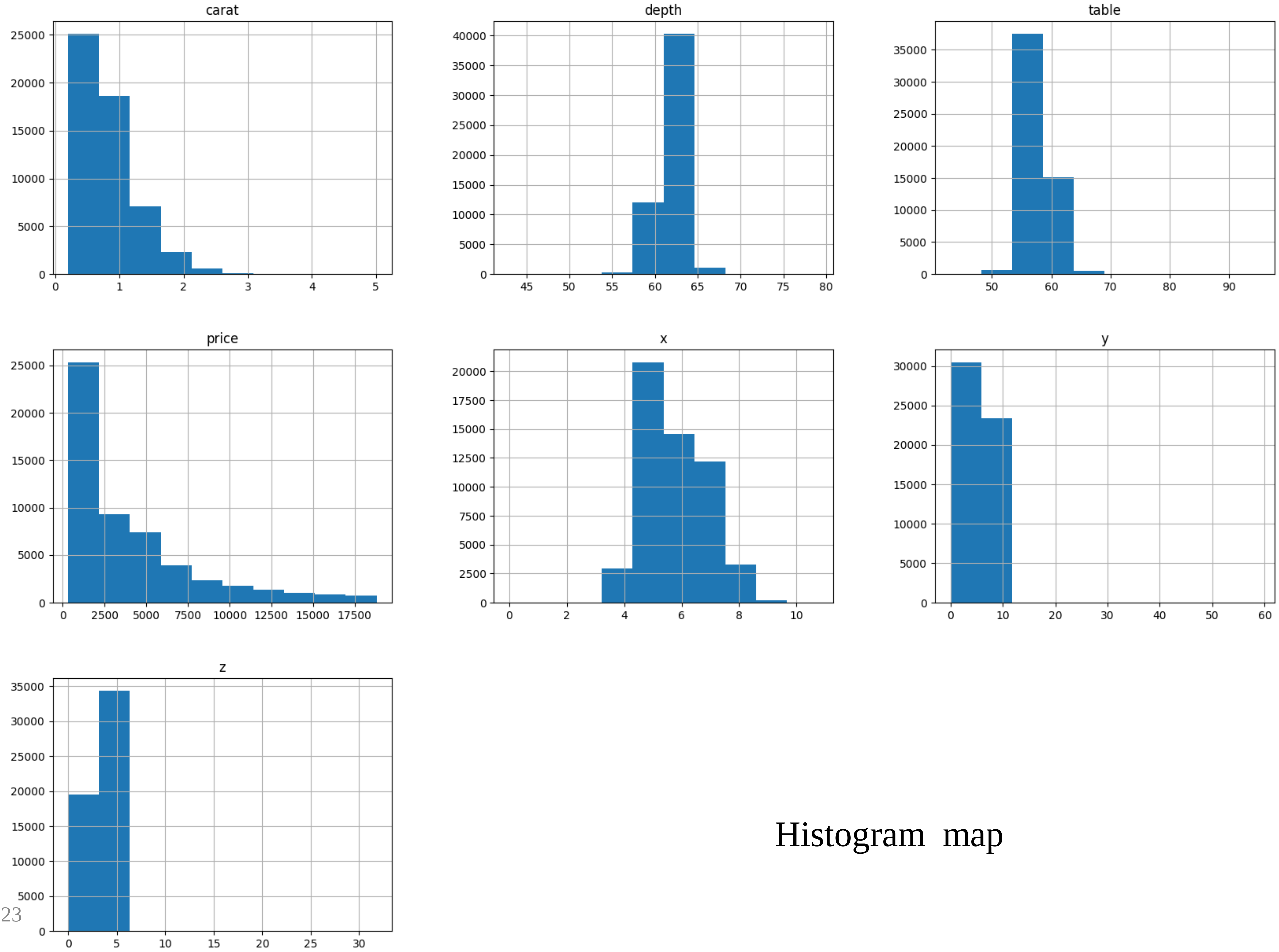
```
carat          float64
cut             object
color           object
clarity         object
depth          float64
table          float64
price            int64
x              float64
y              float64
z              float64
```

DATA

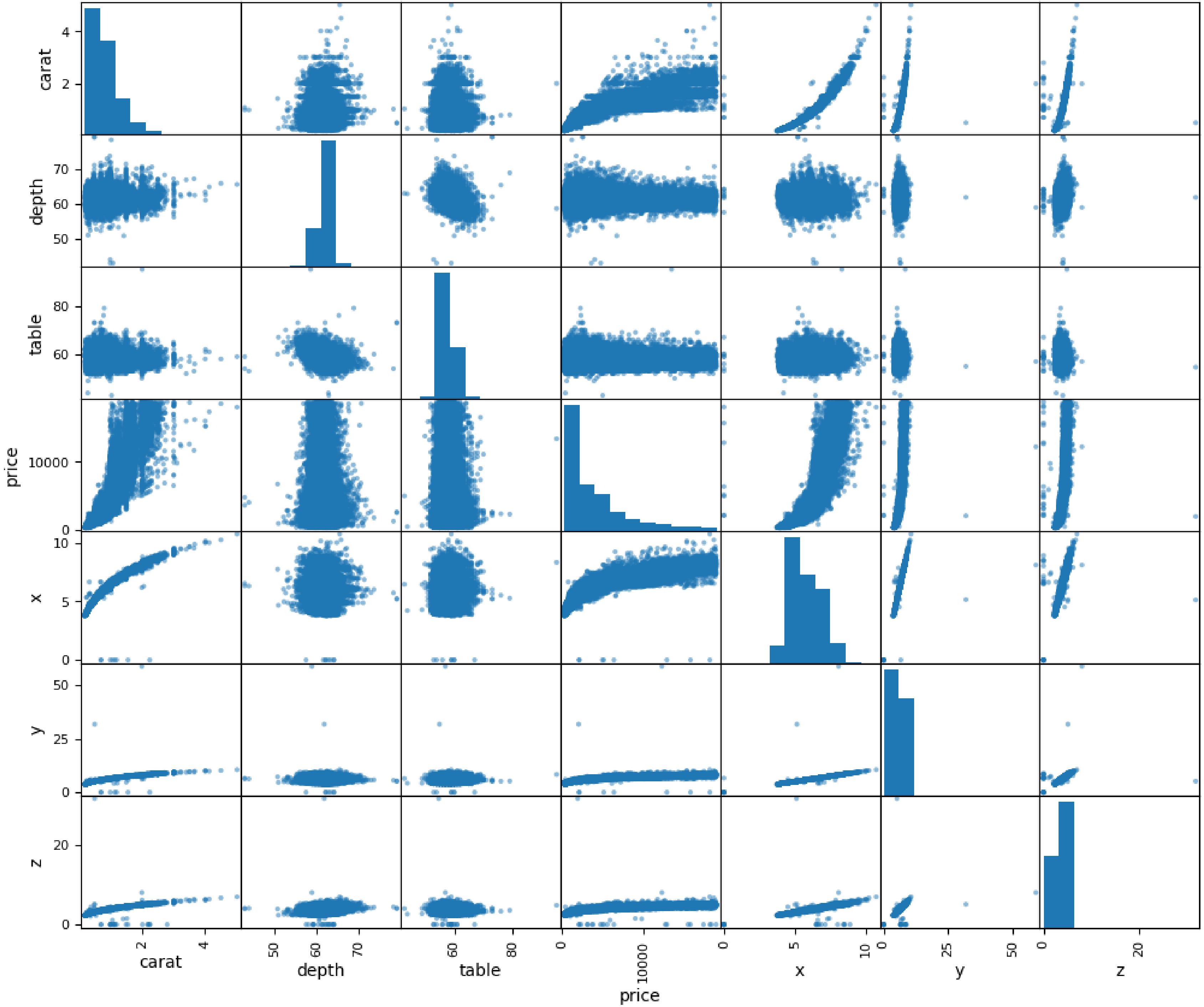
❖EXPLORE DATA



	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

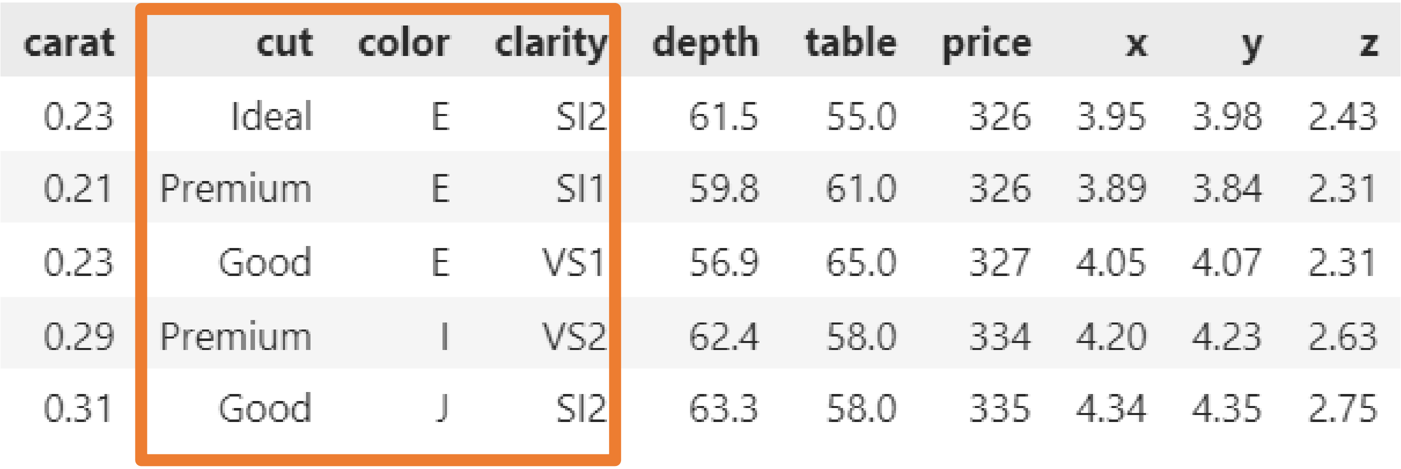


Histogram map

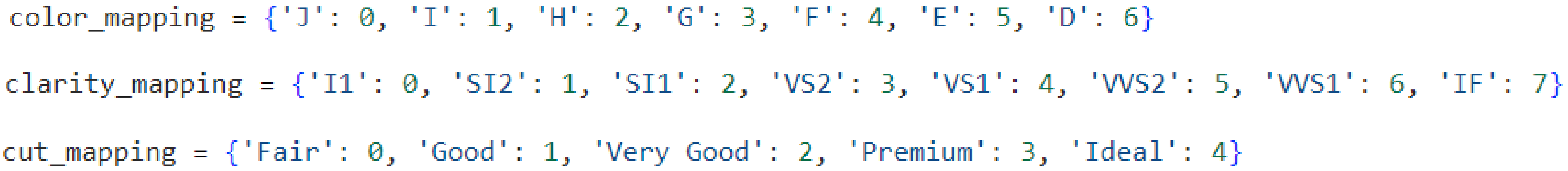

carat
4
2
depth
70
60
50
table
80
60
price
10000
0
x
10
5
0
y
50
25
0
z
20
0
2
4
carat
50
60
70
depth
60
80
table
0
10000
price
0
5
10
x
0
25
50
y
0
20
z

DATA

❖DATA PREPARATION– Xử lí các cột có thuộc tính văn bản, hạng mục



carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

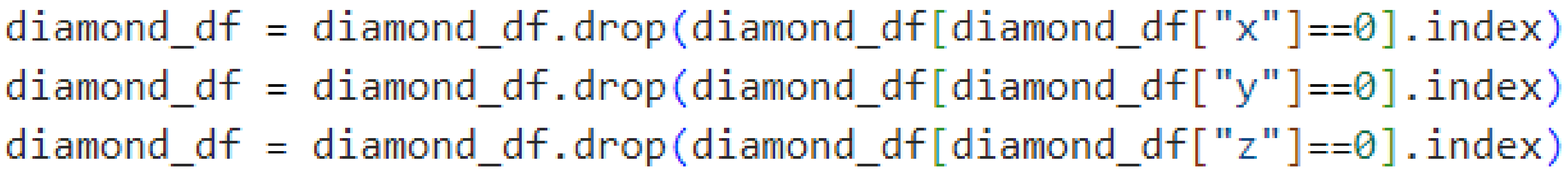


```
color_mapping = {'J': 0, 'I': 1, 'H': 2, 'G': 3, 'F': 4, 'E': 5, 'D': 6}

clarity_mapping = {'I1': 0, 'SI2': 1, 'SI1': 2, 'VS2': 3, 'VS1': 4, 'VVS2': 5, 'VVS1': 6, 'IF': 7}

cut_mapping = {'Fair': 0, 'Good': 1, 'Very Good': 2, 'Premium': 3, 'Ideal': 4}
```

DATA

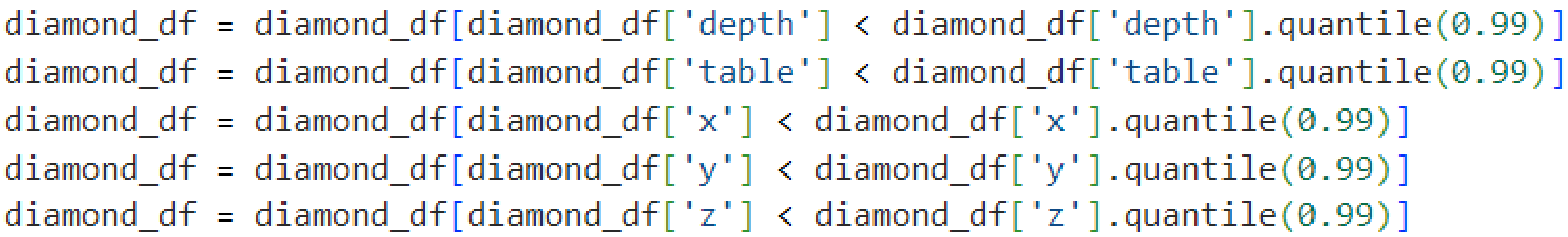
❖DATA PREPARATION– Xử lí các giá trị có thê gây nhiễu

Loại bỏ các sample có giá trị x, y, z = 0



```
diamond_df = diamond_df.drop(diamond_df[diamond_df["x"]==0].index)
diamond_df = diamond_df.drop(diamond_df[diamond_df["y"]==0].index)
diamond_df = diamond_df.drop(diamond_df[diamond_df["z"]==0].index)
```

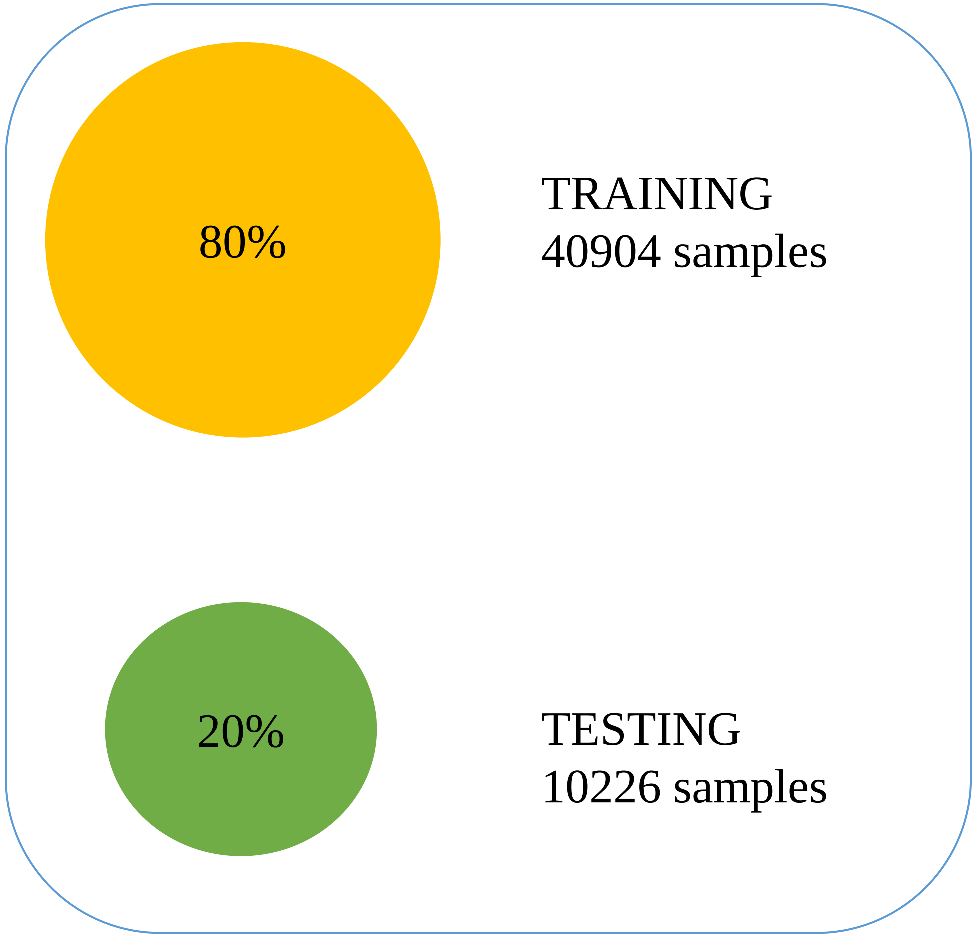
Loại bỏ các sample có giá trị lớn hơn 99% giá trị còn lại



```
diamond_df = diamond_df[diamond_df['depth'] < diamond_df['depth'].quantile(0.99)]
diamond_df = diamond_df[diamond_df['table'] < diamond_df['table'].quantile(0.99)]
diamond_df = diamond_df[diamond_df['x'] < diamond_df['x'].quantile(0.99)]
diamond_df = diamond_df[diamond_df['y'] < diamond_df['y'].quantile(0.99)]
diamond_df = diamond_df[diamond_df['z'] < diamond_df['z'].quantile(0.99)]
```

DATA

❖DATA PREPARATION– Training, Testing

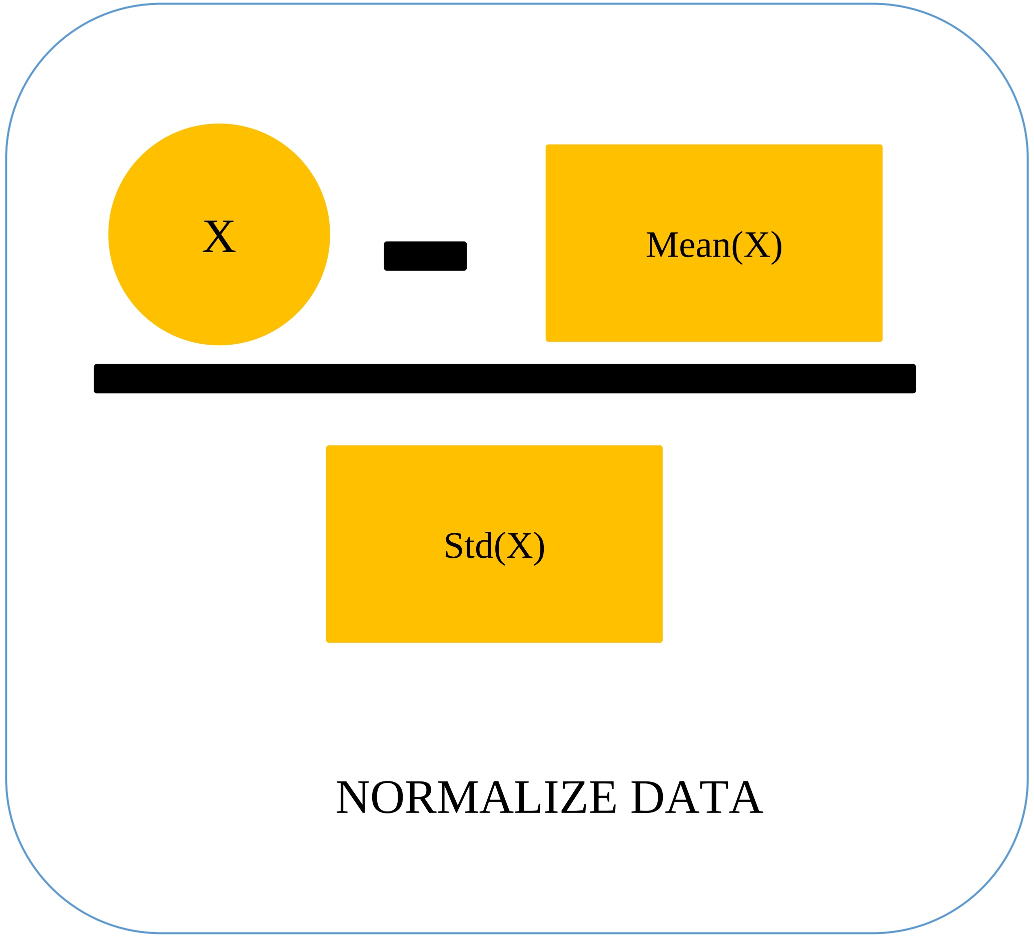


80%

TRAINING
40904 samples

20%

TESTING
10226 samples



X

Mean(X)

Std(X)

NORMALIZE DATA

MODELING

MODELING

❖Linear Regression model

1) Pick m samples $(\mathbf{x}^{(i)}, y^{(i)})$ from training data

2) Compute output $\hat{y}^{(i)}$

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} = \left(\boldsymbol{x}^{(i)}\right)^T \boldsymbol{\theta} \quad \text{for } 0 \le i < m$$

3) Compute loss

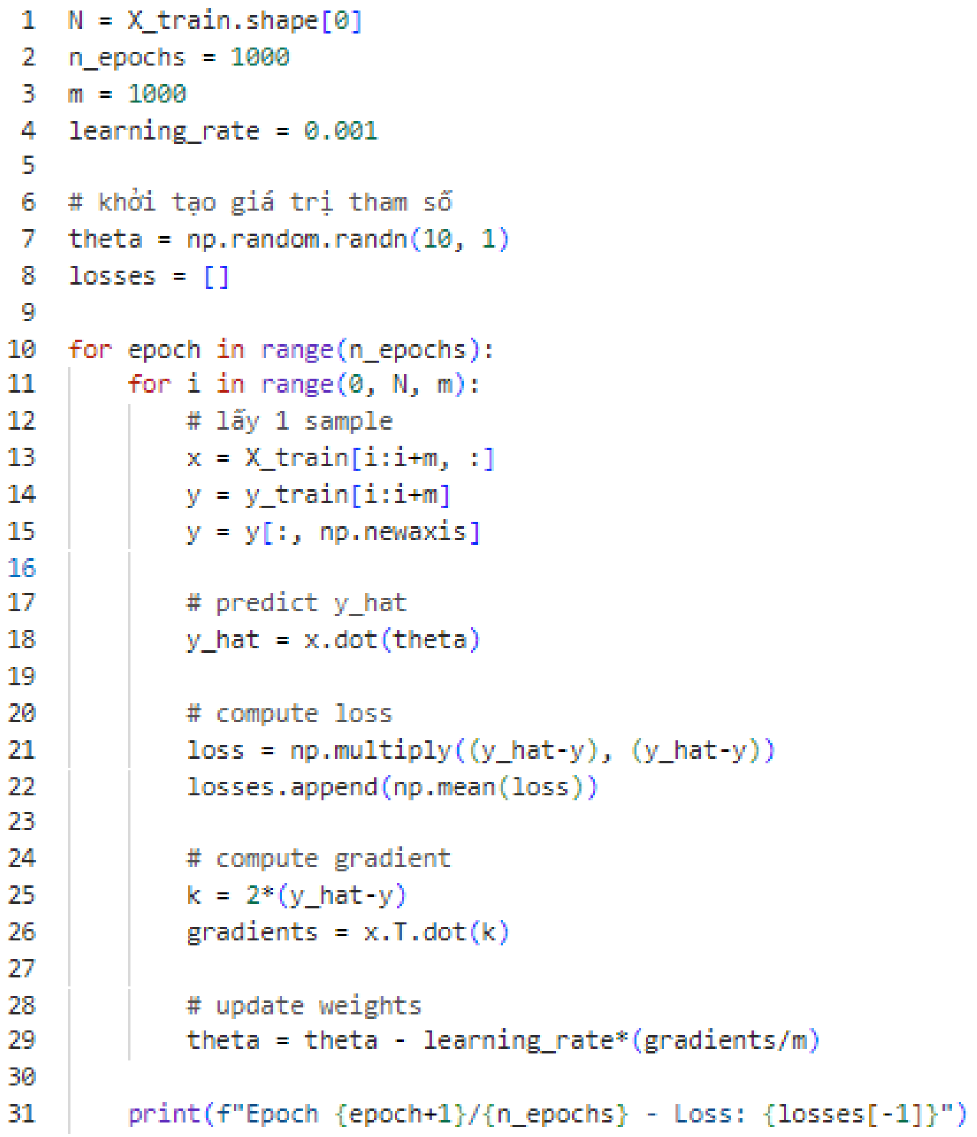
$$L^{(i)} = (\hat{y}^{(i)} - y^{(i)})^2 \quad \text{for } 0 \le i < m$$

4) Compute derivative

$$L'^{(i)}_{\boldsymbol{\theta}} = 2\boldsymbol{x}^{(i)}(\hat{y}^{(i)} - y^{(i)}) \text{ for } 0 \le i < m$$

5) Update parameters

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \frac{\sum_i L'^{(i)}_{\boldsymbol{\theta}}}{m} \qquad \eta \text{ is learning rate}$$



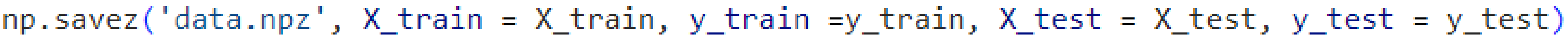
```python
N = X_train.shape[0]
n_epochs = 1000
m = 1000
learning_rate = 0.001

# khởi tạo giá trị tham số
theta = np.random.randn(10, 1)
losses = []

for epoch in range(n_epochs):
    for i in range(0, N, m):
        # lấy 1 sample
        x = X_train[i:i+m, :]
        y = y_train[i:i+m]
        y = y[:, np.newaxis]

        # predict y_hat
        y_hat = x.dot(theta)

        # compute loss
        loss = np.multiply((y_hat-y), (y_hat-y))
        losses.append(np.mean(loss))

        # compute gradient
        k = 2*(y_hat-y)
        gradients = x.T.dot(k)

        # update weights
        theta = theta - learning_rate*(gradients/m)

    print(f"Epoch {epoch+1}/{n_epochs} - Loss: {losses[-1]}")
```

MODELING

❖Training

DATA	MSE	MAE
Train	670565	
Test	1537479	955

SAVE WEIGHTS



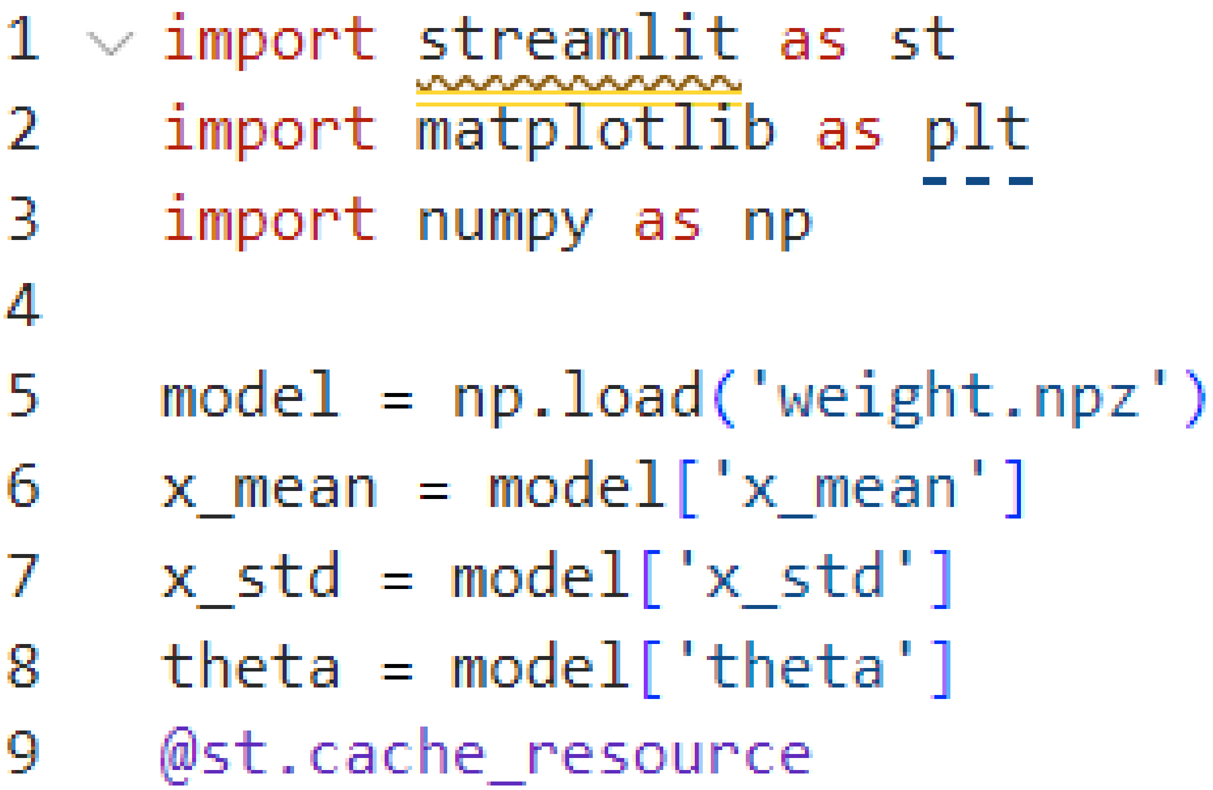
```
np.savez('data.npz', X_train = X_train, y_train =y_train, X_test = X_test, y_test = y_test)
```

Deployment

Deployment

❖Build App

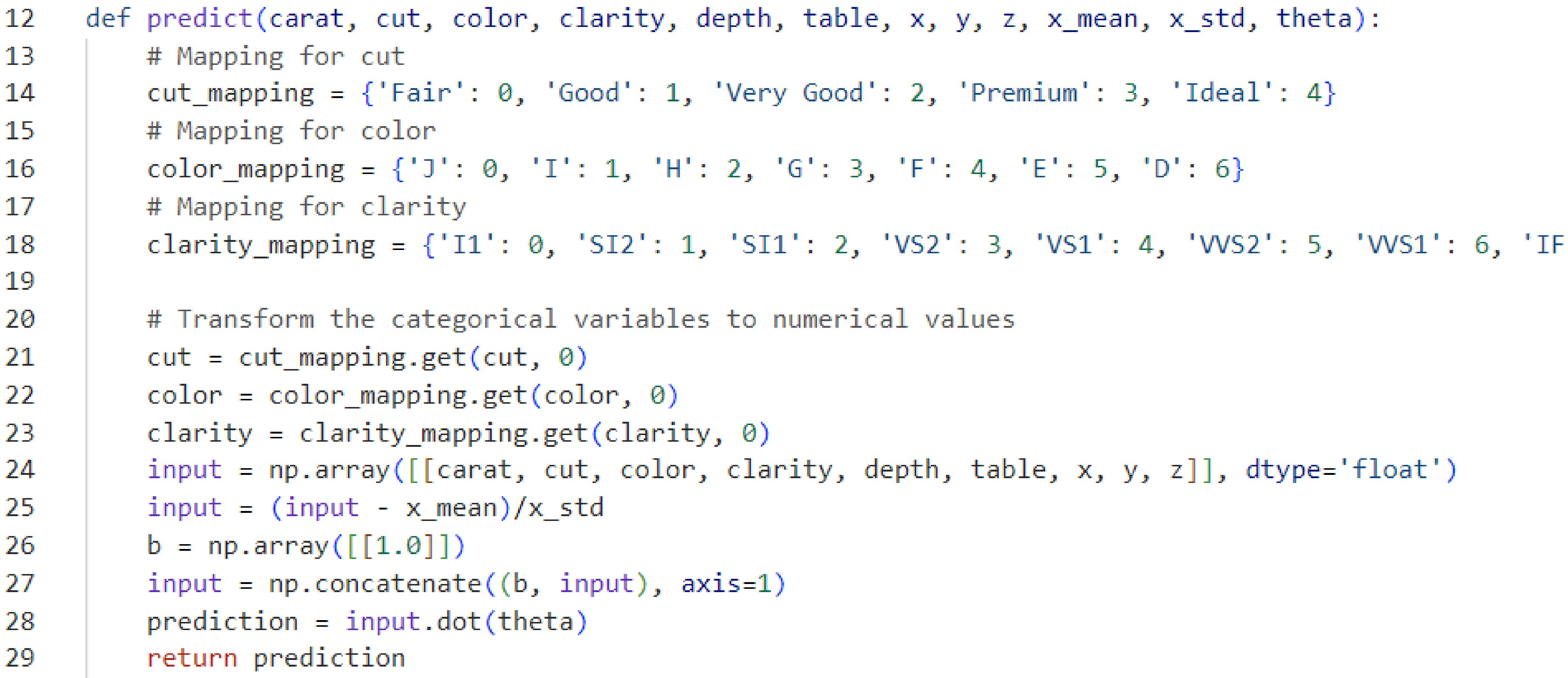
Step1: Load Weights



```python
import streamlit as st
import matplotlib as plt
import numpy as np

model = np.load('weight.npz')
x_mean = model['x_mean']
x_std = model['x_std']
theta = model['theta']
@st.cache_resource
```

Deployment

❖Build App

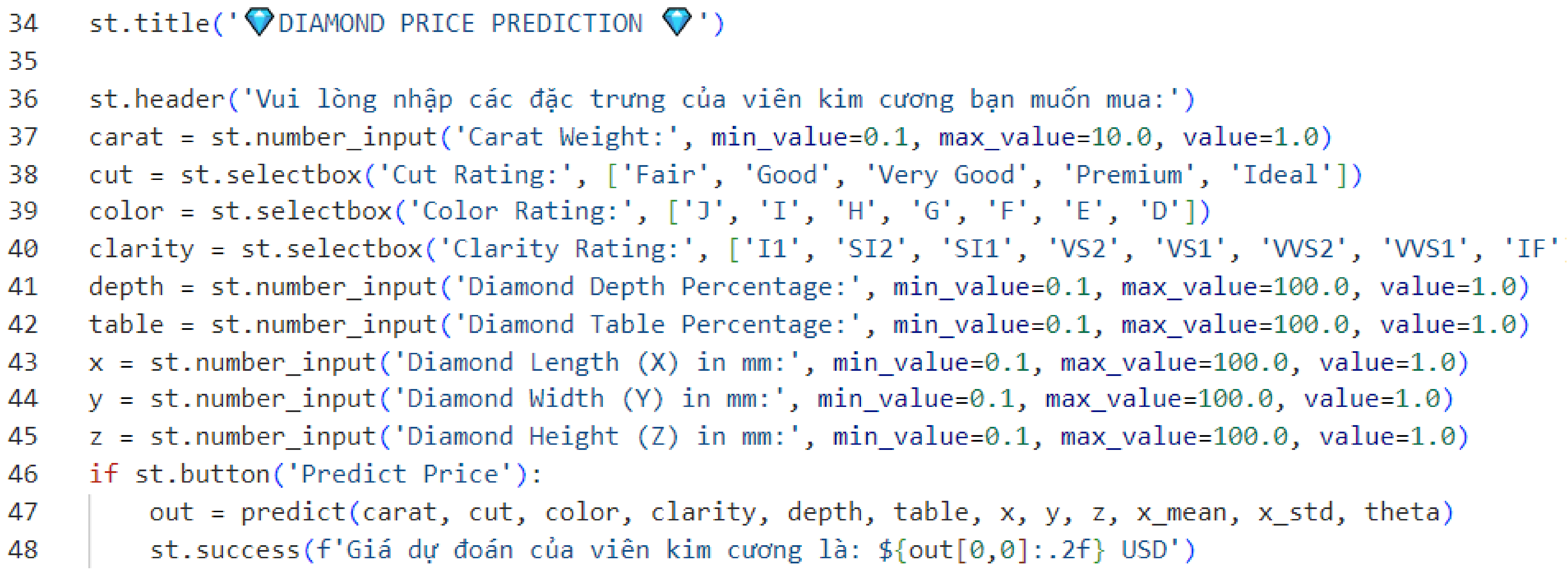


```
def predict(carat, cut, color, clarity, depth, table, x, y, z, x_mean, x_std, theta):
    # Mapping for cut
    cut_mapping = {'Fair': 0, 'Good': 1, 'Very Good': 2, 'Premium': 3, 'Ideal': 4}
    # Mapping for color
    color_mapping = {'J': 0, 'I': 1, 'H': 2, 'G': 3, 'F': 4, 'E': 5, 'D': 6}
    # Mapping for clarity
    clarity_mapping = {'I1': 0, 'SI2': 1, 'SI1': 2, 'VS2': 3, 'VS1': 4, 'VVS2': 5, 'VVS1': 6, 'IF

    # Transform the categorical variables to numerical values
    cut = cut_mapping.get(cut, 0)
    color = color_mapping.get(color, 0)
    clarity = clarity_mapping.get(clarity, 0)
    input = np.array([[carat, cut, color, clarity, depth, table, x, y, z]], dtype='float')
    input = (input - x_mean)/x_std
    b = np.array([[1.0]])
    input = np.concatenate((b, input), axis=1)
    prediction = input.dot(theta)
    return prediction
```

Step2: Define Predict Function

Deployment

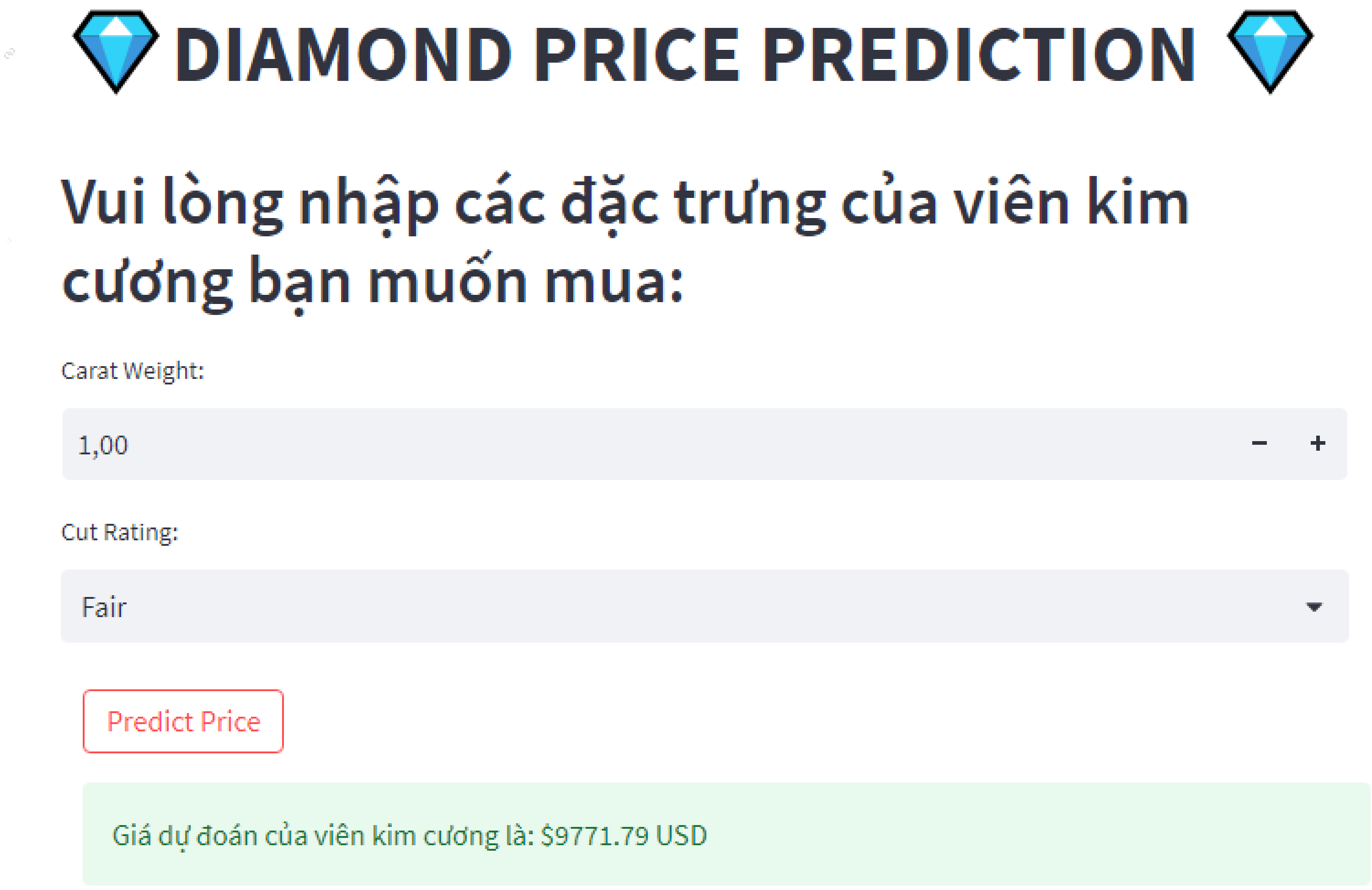
❖Build App



```
st.title('💎DIAMOND PRICE PREDICTION 💎')

st.header('Vui lòng nhập các đặc trưng của viên kim cương bạn muốn mua:')
carat = st.number_input('Carat Weight:', min_value=0.1, max_value=10.0, value=1.0)
cut = st.selectbox('Cut Rating:', ['Fair', 'Good', 'Very Good', 'Premium', 'Ideal'])
color = st.selectbox('Color Rating:', ['J', 'I', 'H', 'G', 'F', 'E', 'D'])
clarity = st.selectbox('Clarity Rating:', ['I1', 'SI2', 'SI1', 'VS2', 'VS1', 'VVS2', 'VVS1', 'IF'
depth = st.number_input('Diamond Depth Percentage:', min_value=0.1, max_value=100.0, value=1.0)
table = st.number_input('Diamond Table Percentage:', min_value=0.1, max_value=100.0, value=1.0)
x = st.number_input('Diamond Length (X) in mm:', min_value=0.1, max_value=100.0, value=1.0)
y = st.number_input('Diamond Width (Y) in mm:', min_value=0.1, max_value=100.0, value=1.0)
z = st.number_input('Diamond Height (Z) in mm:', min_value=0.1, max_value=100.0, value=1.0)
if st.button('Predict Price'):
    out = predict(carat, cut, color, clarity, depth, table, x, y, z, x_mean, x_std, theta)
    st.success(f'Giá dự đoán của viên kim cương là: ${out[0,0]:.2f} USD')
```

Deployment

❖DEMO



💎 DIAMOND PRICE PREDICTION 💎

Vui lòng nhập các đặc trưng của viên kim cương bạn muốn mua:

Carat Weight:

1,00

Cut Rating:

Fair

Giá dự đoán của viên kim cương là: $9771.79 USD