Shining Bright: Diamond Competition (Kaggel)

Share Artificial Intelligence

SHAI
Share Artificial Intelligence CLUB

Team Member:

1) Ashwaq

2) Rawan

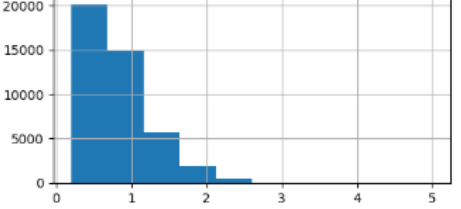
3)Maram

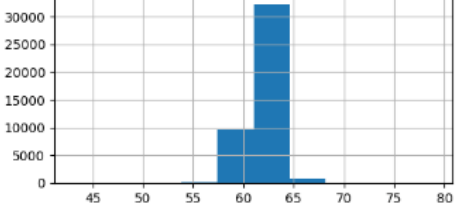
cut	color	clarity	depth	table	price	x	y	z
Ideal	I	SI2	61.8	57.0	4270	6.57	6.60	4.07
Premium	G	VVS2	60.9	58.0	15164	7.38	7.42	4.51
Ideal	F	VS2	61.3	56.0	828	4.43	4.41	2.71
Ideal	G	VS2	61.2	56.0	1577	5.19	5.22	3.19
Premium	H	VVS2	61.0	57.0	2596	5.76	5.72	3.50

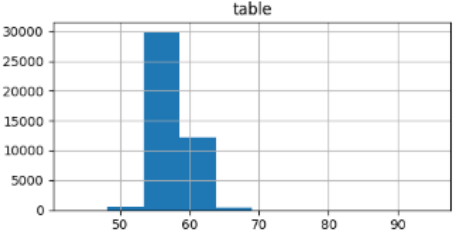
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43152 entries, 0 to 43151
Data columns (total 11 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Id       43152 non-null  int64
 1   carat    43152 non-null  float64
 2   cut      43152 non-null  object
 3   color    43152 non-null  object
 4   clarity  43152 non-null  object
 5   depth    43152 non-null  float64
 6   table    43152 non-null  float64
 7   price    43152 non-null  int64
 8   x        43152 non-null  float64
 9   y        43152 non-null  float64
 10  z        43152 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 3.6+ MB
```

	Id	carat	depth	table	price	x	y	z
count	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000	43152.000000
mean	21576.500000	0.797855	61.747177	57.458347	3929.491912	5.731568	5.735018	3.538568
std	12457.053745	0.473594	1.435454	2.233904	3985.527795	1.121279	1.148809	0.708238
min	1.000000	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	10788.750000	0.400000	61.000000	56.000000	947.750000	4.710000	4.720000	2.910000
50%	21576.500000	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	32364.250000	1.040000	62.500000	59.000000	5312.000000	6.540000	6.540000	4.040000
max	43152.000000	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

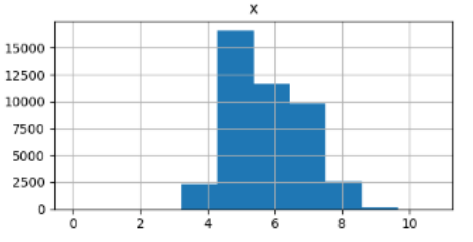
Data Analisys and Visualization

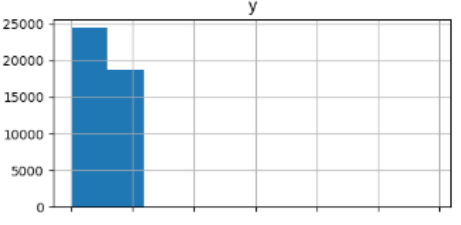


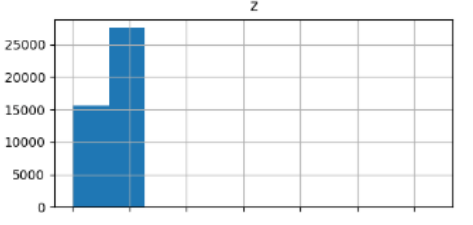


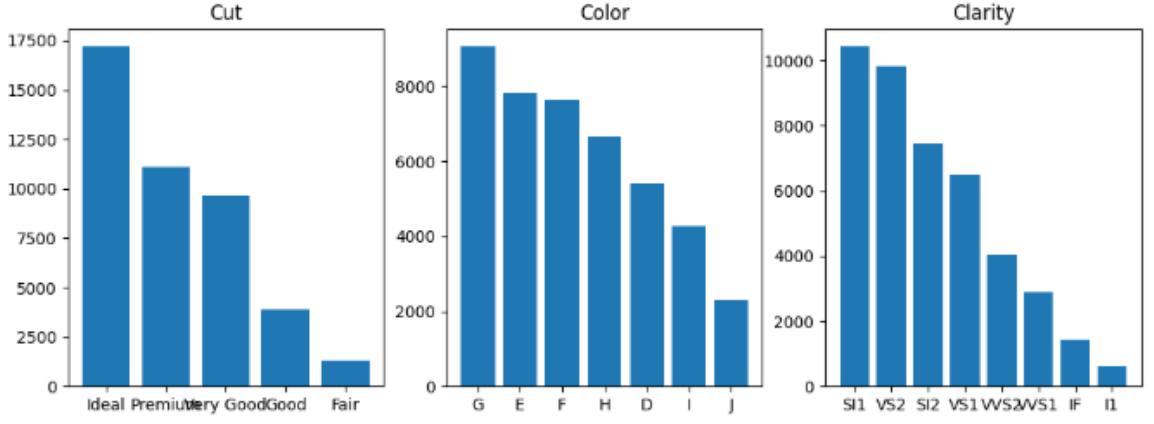


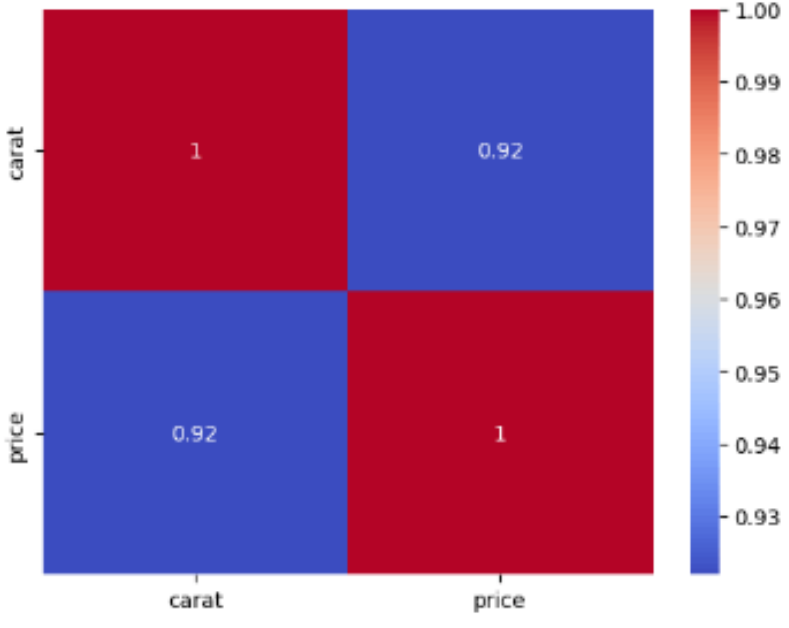
table

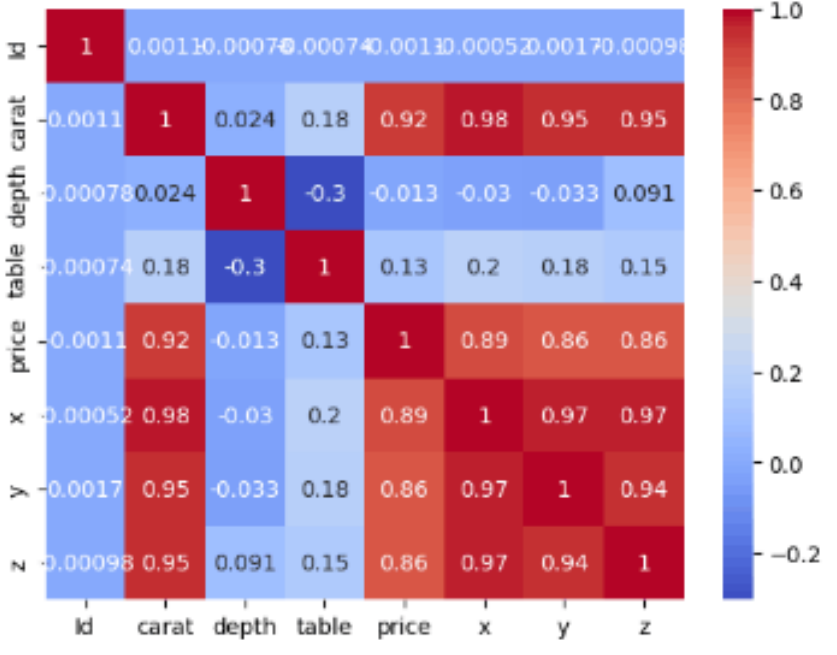

x

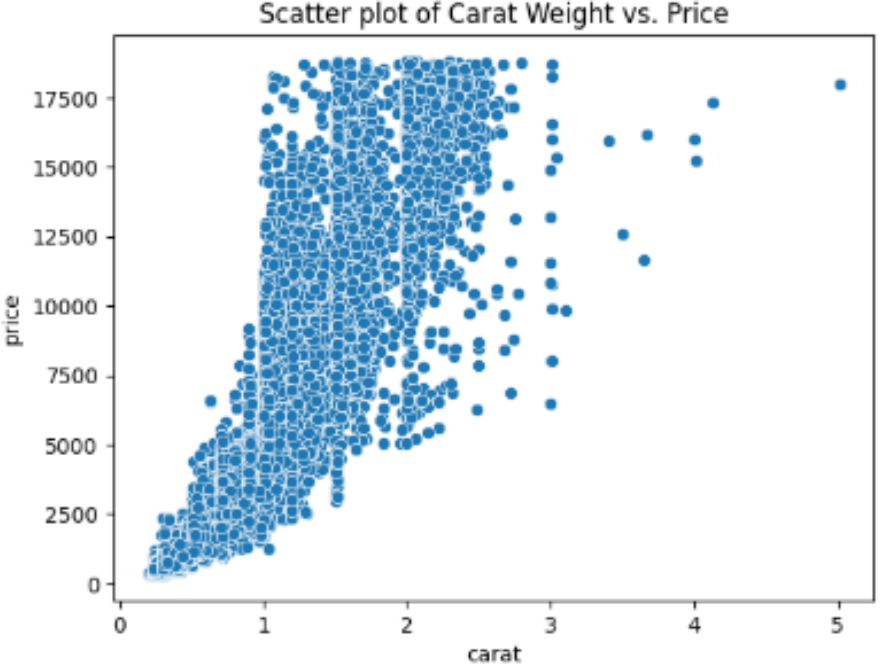

y

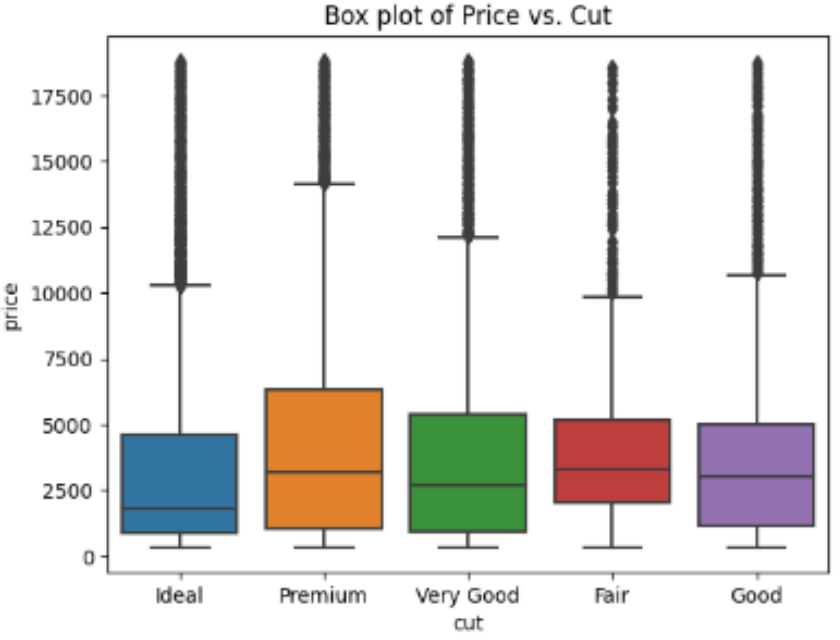

z


Cut
17500
15000
12500
10000
7500
5000
2500
0
Ideal
Premium
Very Good
Good
Fair
Color
8000
6000
4000
2000
0
G
E
F
H
D
I
J
Clarity
10000
8000
6000
4000
2000
0
SI1
VS2
SI2
VS1
VVS2
VVS1
IF
I1


carat
price
1
0.92
0.92
1
carat
price
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93


id
carat
depth
table
price
x
y
z
1
0.0011
0.00078
0.00074
0.0011
0.00052
0.0017
0.00098
0.0011
1
0.024
0.18
0.92
0.98
0.95
0.95
0.00078
0.024
1
-0.3
-0.013
-0.03
-0.033
0.091
0.00074
0.18
-0.3
1
0.13
0.2
0.18
0.15
0.0011
0.92
-0.013
0.13
1
0.89
0.86
0.86
0.00052
0.98
-0.03
0.2
0.89
1
0.97
0.97
0.0017
0.95
-0.033
0.18
0.86
0.97
1
0.94
0.00098
0.95
0.091
0.15
0.86
0.97
0.94
1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2


Scatter plot of Carat Weight vs. Price
price
carat
0
2500
5000
7500
10000
12500
15000
17500
0
1
2
3
4
5


Box plot of Price vs. Cut
price
17500
15000
12500
10000
7500
5000
2500
0
Ideal
Premium
Very Good
Fair
Good
cut

Build and Evaluation the model

```
# import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
```

```
# Select the numeric columns
num_cols = ['carat', 'depth', 'table', 'x', 'y', 'z']

# Scale the numeric columns
scaler = StandardScaler()
df[num_cols] = scaler.fit_transform(df[num_cols])

# Select the features and target variable
X = df[['carat', 'depth', 'table', 'x', 'y', 'z']]
y = df['price']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
# create a linear regression model
model = LinearRegression()
# train the model using the training data
model.fit(X_train, y_train)
```

▾ LinearRegression

LinearRegression()

```python
# Train the model 25 times on different subsets of the data and calculate accuracy and error rate
accuracy_scores = []
error_rates = []
for i in range(100):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
    model = LinearRegression()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = model.score(X_test, y_test)
    mse = mean_squared_error(y_test, y_pred)
    error_rate = mse / y_test.mean()
    accuracy_scores.append(accuracy)
    error_rates.append(error_rate)

print("Average accuracy over 100 iterations:", sum(accuracy_scores) / len(accuracy_scores))
print("Average error rate over 100 iterations:", sum(error_rates) / len(error_rates))
```

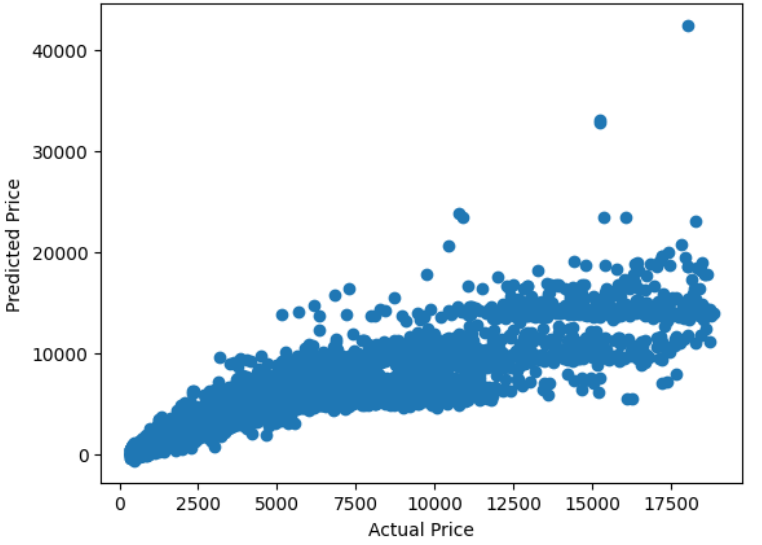
```
Average accuracy over 100 iterations: 0.8555573822134214
Average error rate over 100 iterations: 582.8276192994317
```

```
# make predictions on the test data
y_pred = model.predict(X_test)

# evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
print("Mean squared error:", mse)
```

```
Mean squared error: 2377222.0352184195
```

```
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.show()
```



predict target for new data

```
test_df= pd.read_csv("/content/test.csv")
```

```
num_cols_test = ['carat', 'depth', 'table', 'x', 'y', 'z']
# Scale the numeric columns
scaler_new = StandardScaler()
# Scale the numeric columns in test data using the same scaler object as before
test_df[num_cols_test] = scaler_new.fit_transform(test_df[num_cols_test])
```

```
# Select the features from the test data
X_new_test = test_df[['carat', 'depth', 'table', 'x', 'y', 'z']]

# Predict prices using the trained model
y_pred_test = model.predict(X_new_test)

# Create a new column in the test data for the predicted prices
test_df['predicted_price'] = y_pred_test

# Save the test data with the predicted prices to a CSV file
test_df.to_csv('predicted_prices.csv', index=False)
```

```
y_pred_test.shape
```

```
y_pred_test
```

```
array([  720.53932139,  2693.33893044,  1146.38386622, ...,
        3169.65760764,  5959.22706199, 13861.48144473])
```

```
import pandas as pd
```

```
f=pd.read_csv("/content/predicted_prices 22.csv")
```

```
f.shape
```

```
(10788, 11)
```

```
f.columns
```

```
Index(['Id', 'carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y',
       'z', 'predicted_price'],
      dtype='object')
```

```
f=f.drop(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z'],axis=1)
```

```
f.shape
```

```
f.to_csv('price.csv', index=False)

# display a message to indicate the file has been saved
print("The updated dataframe has been saved to updated_dataframe.csv")
```

```
The updated dataframe has been saved to updated_dataframe.csv
```

```
f.shape
```

```
(10788, 2)
```

The 4Cs of Diamond Quality

The **4Cs** of diamond quality are carat weight, cut, clarity, and color. These factors determine the value and quality of a diamond. A high-quality diamond will have a high carat weight, excellent cut, high clarity, and minimal color.

Conclusion

Diamonds are a fascinating and beloved gemstone that have captivated people for centuries. Whether you admire them for their beauty, rarity, or symbolism, there's no denying that diamonds are truly special.

Thanks!

Do you have any questions?
addyouremail@freepik.com
+91 620 421 838
yourcompany.com