ECDA

SC1015 PROJECT

Diamond Price Calculator

carat Clarity Color Cut

By

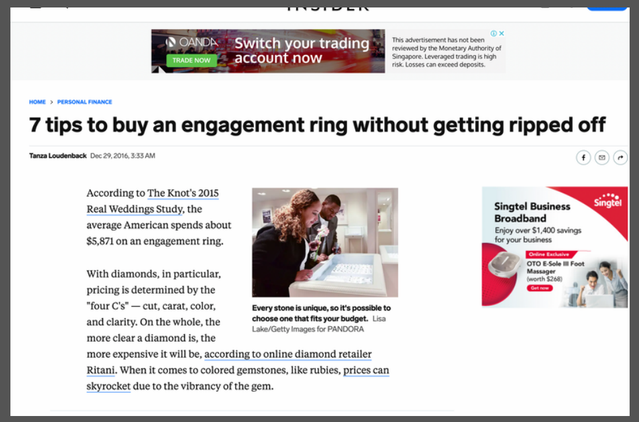
- Dylan Lim Zhuo Yang
- Lim Sin Pei

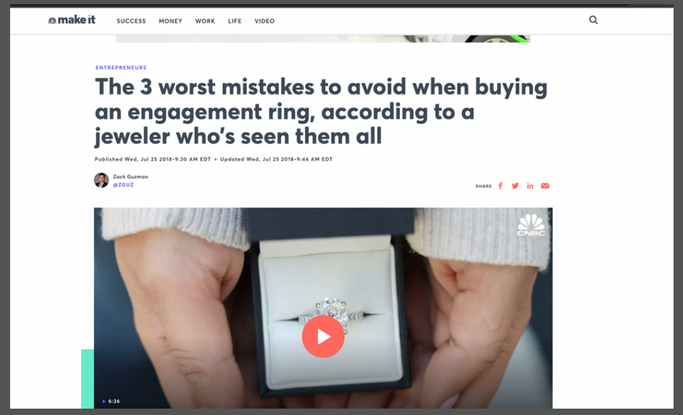
PRACTICAL MOTIVATION

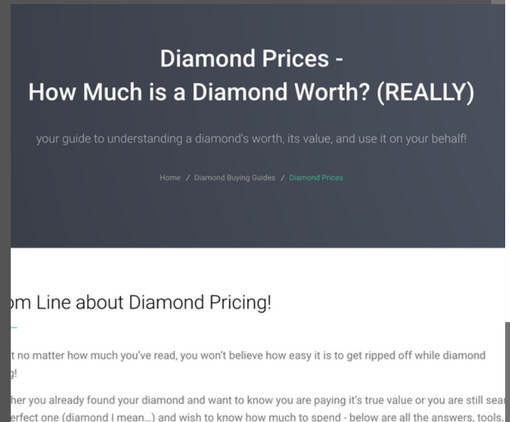
REAL PROBLEM FACED:

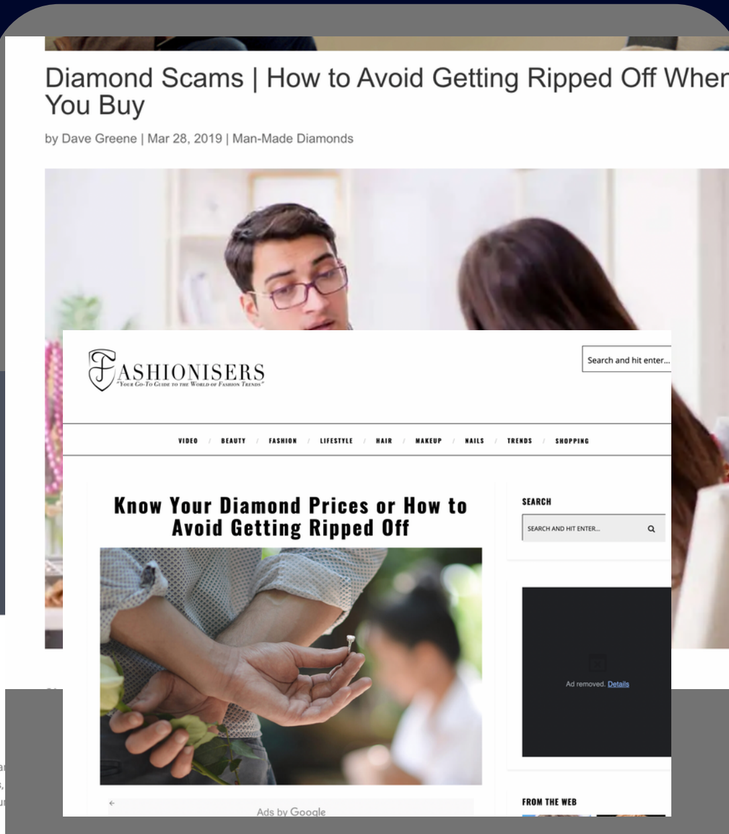
-> Average consumers(couples) overcharged for diamonds scams by certain retailers as they are unaware of diamond's true market value

->Unlike other commodities, quality of diamonds can be reliably measured by specific parameters



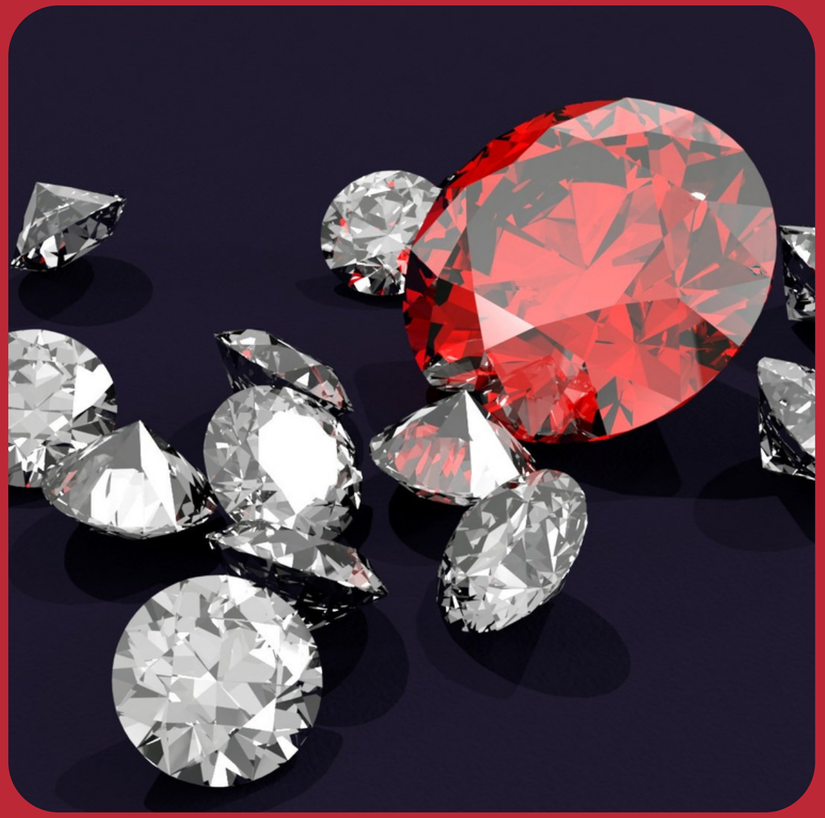






PROBLEM DEFINITION

How do different variables determine the prices of diamonds in the market



SAMPLE COLLECTION

KAGGLE. https://www.kaggle.com/datasets/shivam2503/diamonds?datasetId=1312&sortBy=voteCount

DATA SIZE: 53, 940

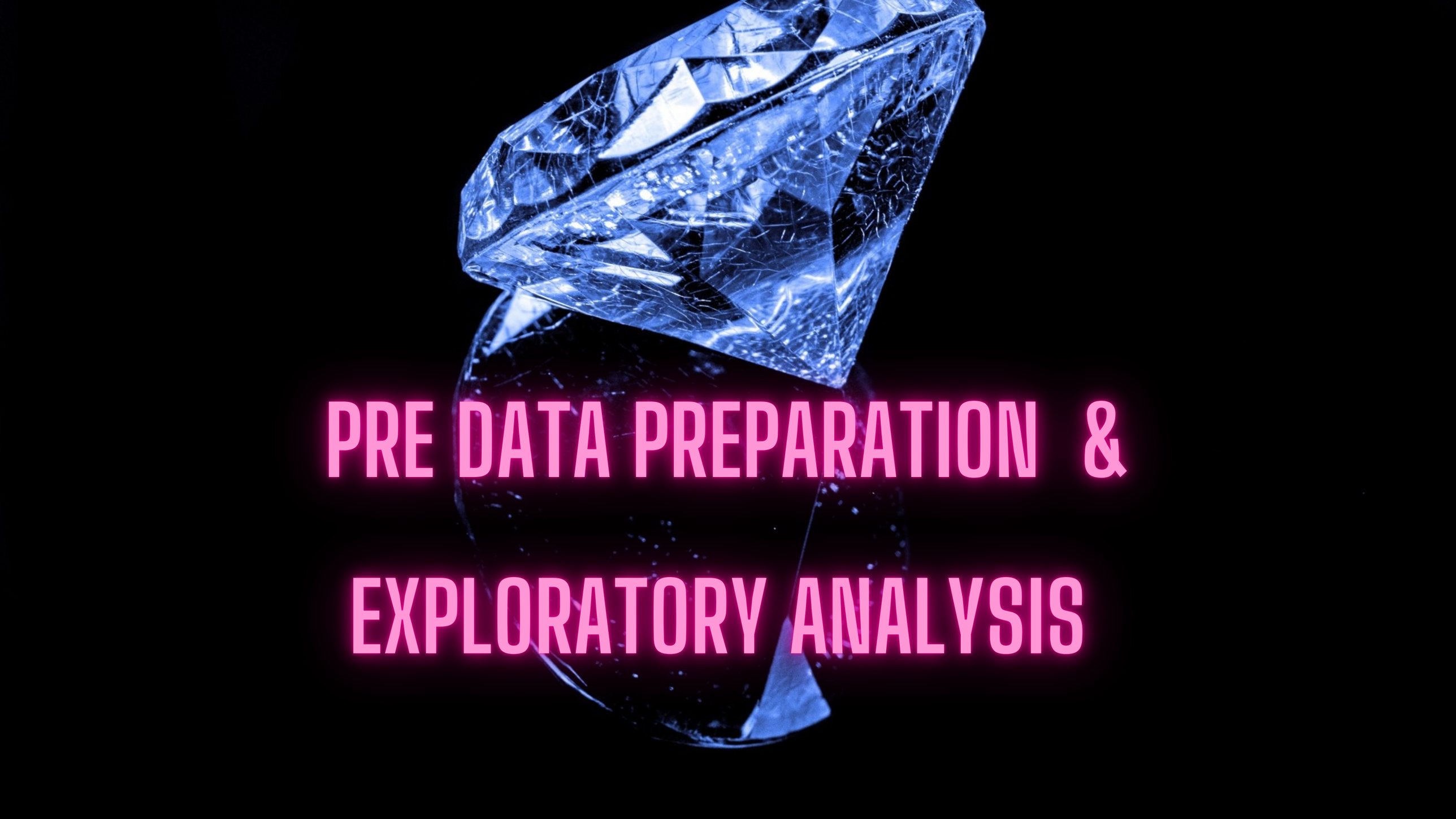
diamonds.csv (3.19 MB)

Detail | Compact | Column — 10 of 11 columns

#	carat	cut	color	clarity	depth	table	price	x	y
26	0.23	Very Good	G	VVS2	60.4	58	354	3.97	4.01
27	0.24	Premium	I	VS1	62.5	57	355	3.97	3.94
28	0.3	Very Good	J	VS2	62.2	57	357	4.28	4.3
29	0.23	Very Good	D	VS2	60.5	61	357	3.96	3.97
30	0.23	Very Good	F	VS1	60.9	57	357	3.96	3.99
31	0.23	Very Good	F	VS1	60	57	402	4	4.03
32	0.23	Very Good	F	VS1	59.8	57	402	4.04	4.06
33	0.23	Very Good	E	VS1	60.7	59	402	3.97	4.01
34	0.23	Very Good	E	VS1	59.5	58	402	4.01	4.06
35	0.23	Very Good	D	VS1	61.9	58	402	3.92	3.96
36	0.23	Good	F	VS1	58.2	59	402	4.06	4.08
37	0.23	Good	E	VS1	64.1	59	402	3.83	3.85
38	0.31	Good	H	SI1	64	54	402	4.29	4.31
39	0.26	Very Good	D	VS2	60.8	59	403	4.13	4.16
40	0.33	Ideal	I	SI2	61.8	55	403	4.49	4.51
41	0.33	Ideal	I	SI2	61.2	56	403	4.49	4.5
42	0.33	Ideal	J	SI1	61.1	56	403	4.49	4.55
43	0.26	Good	D	VS2	65.2	56	403	3.99	4.02
44	0.26	Good	D	VS1	58.4	63	403	4.19	4.24
45	0.32	Good	H	SI2	63.1	56	403	4.34	4.37
46	0.29	Premium	F	SI1	62.4	58	403	4.24	4.26
47	0.32	Very Good	H	SI2	61.8	55	403	4.35	4.42
48	0.32	Good	H	SI2	63.8	56	403	4.36	4.38
49	0.25	Very Good	E	VS2	63.3	60	404	4	4.03

```
diamondData.describe()
```

	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	[illegible]	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.[illegible]1761	1.142135	0.705[illegible]
min	0.200000	43.000000	43.000000	326.000000	[illegible]000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.7[illegible]0000	4.720000	2.910[illegible]0
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	[illegible]	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


PRE DATA PREPARATION &
EXPLORATORY ANALYSIS

DATA CLEANING & EXTRACTION

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53940 entries, 0 to 53939
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   53940 non-null  int64
 1   carat        53940 non-null  float64
 2   cut          53940 non-null  object
 3   color        53940 non-null  object
 4   clarity      53940 non-null  object
 5   depth        53940 non-null  float64
 6   table        53940 non-null  float64
 7   price        53940 non-null  int64
 8   x            53940 non-null  float64
 9   y            53940 non-null  float64
 10  z            53940 non-null  float64
dtypes: float64(6), int64(2), object(3)
memory usage: 4.5+ MB
```

Drop Unnamed column

```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
price      0
x          0
y          0
z          0
dtype: int64
```

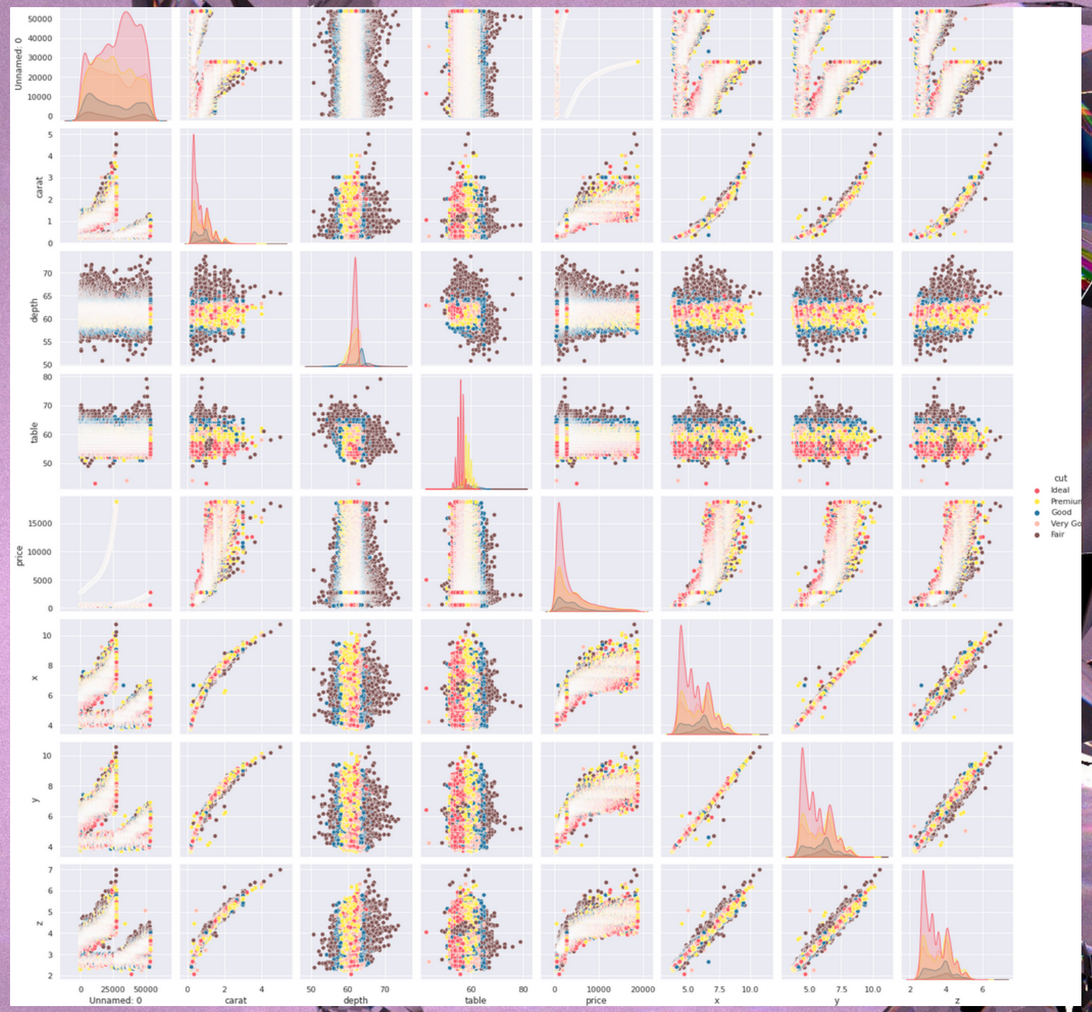
Check no NULL Values

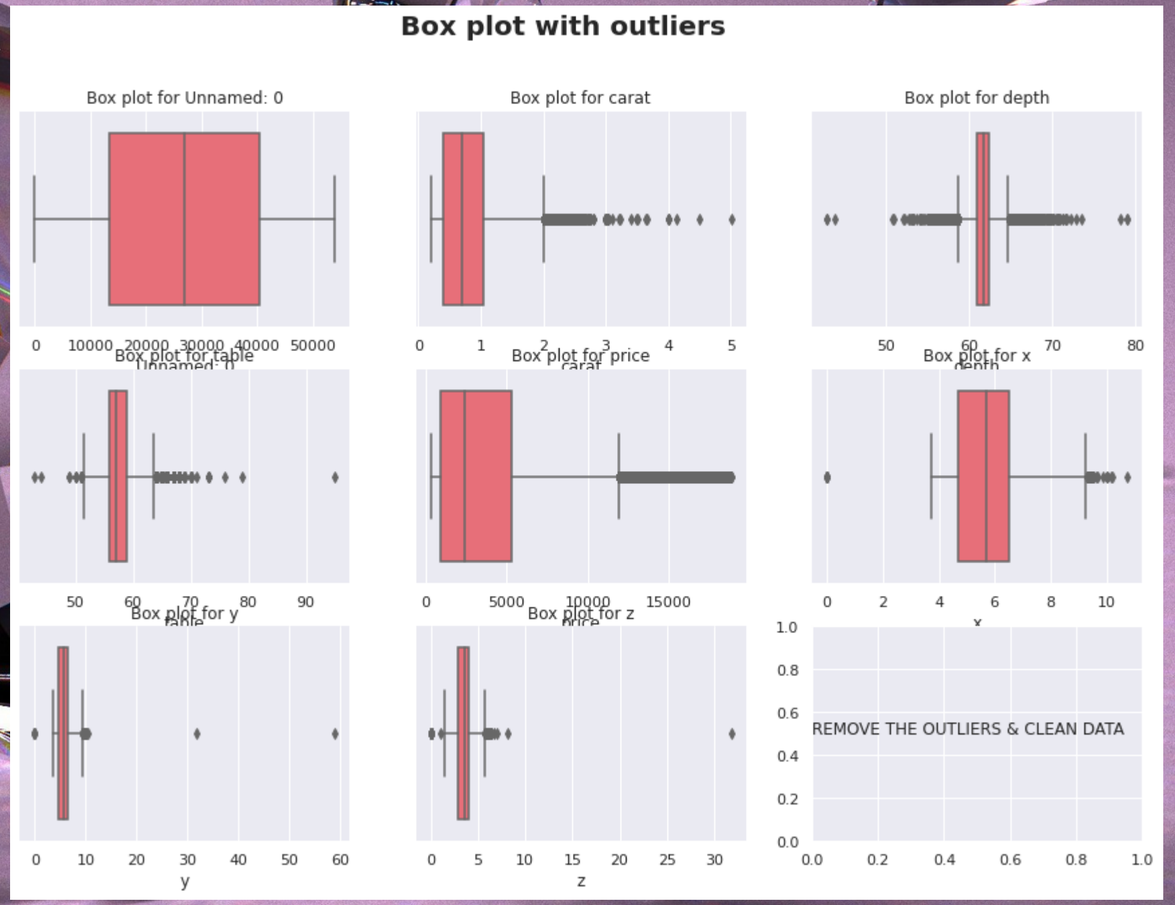
	carat	cut	color	clarity	depth	table	pric	x	y	z
2207	1.00	Premium	G	SI2	59.1	59.0	3 2	6.55	6.48	0.0
2314	1.01	Premium	H	I1	58.1	59.0	167	6.66	6.60	0.0
4791	1.10	Premium	G	SI2	63.0	59.0	3696	6.50	6.47	0.0
5471	1.01	Premium	F	SI2	59.2	58.0	3837	6.50	6.47	0.0
10167	1.50	Good	G	I1	64.0	61.0	4731	7.15	7.04	0.0
11182	1.07	Ideal	F	SI2	61.6	56.0	4954	0.00	6.62	0.0
11963	1.00	Very Good	H	VS2	63.3	53.0	5139	0.00	0.00	0.0
13601	1.15	Ideal	G	VS2	59.2	56.0	5564	6.88	6.83	0.0
15951	1.14	Fair	G	VS1	57.5	67.0	6381	0.00	0.00	0.0
24394	2.18	Premium	H	SI2	59.4	61.0	1 631	8.49	8.45	0.0
24520	1.56	Ideal	G	VS2	62.2	54.0	12 0	0.00	0.00	0.0
26123	2.25	Premium	I	SI1	61.3	58.0	1539	8.52	8.42	0.0
26243	1.20	Premium	D	VVS1	62.1	59.0	15686	0.00	0.00	0.0

X, Y & Z has Minimum values 0 = Defective dataset
Remove faulty data

DATA CLEANING & EXTRACTION

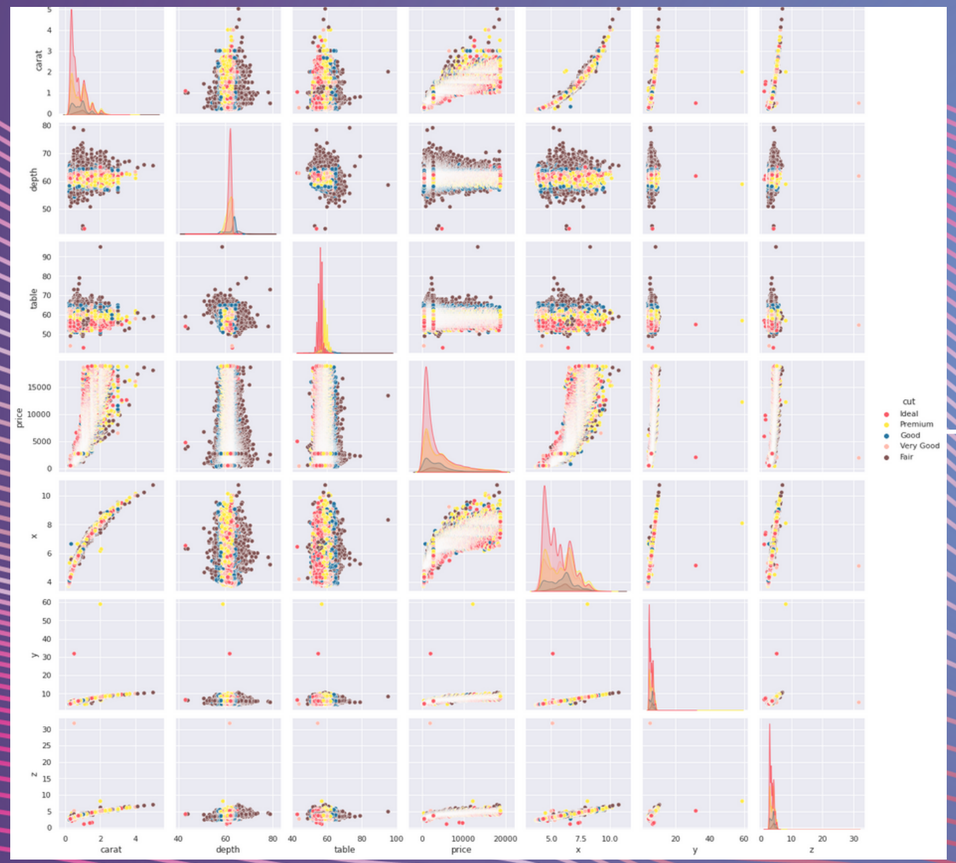
REMOVING OUTLIERS

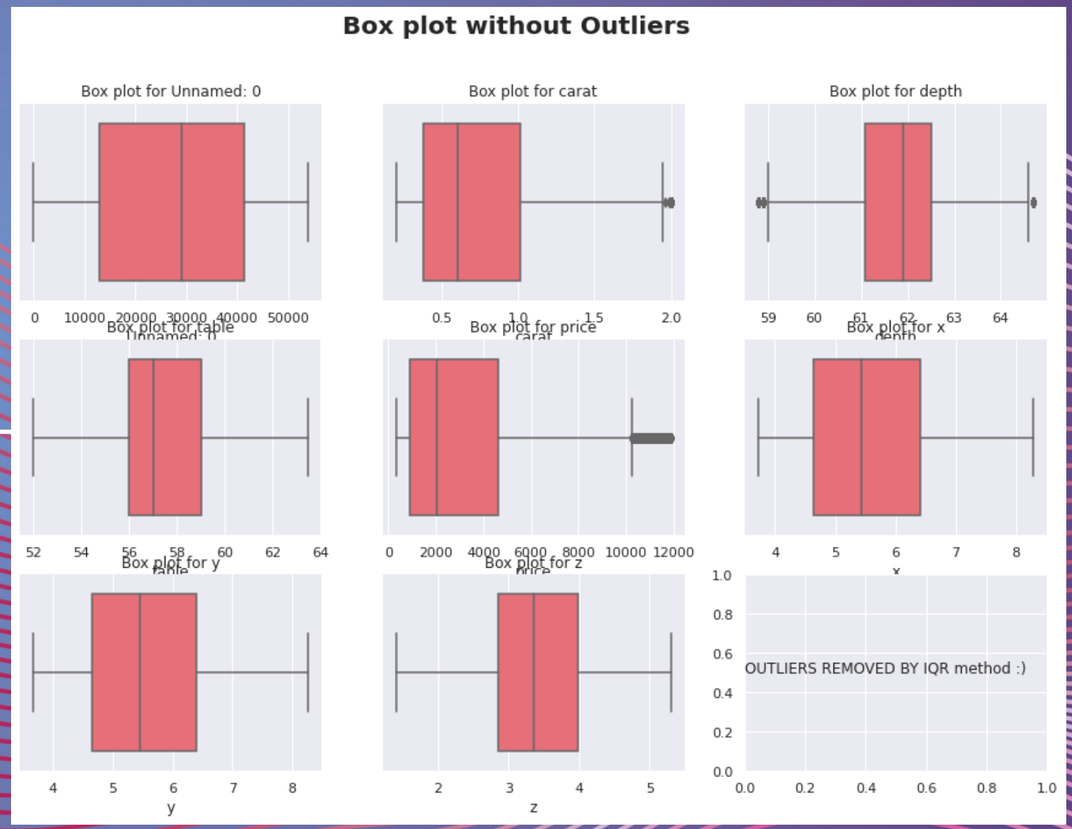




DATA CLEANING & EXTRACTION

```
Q1 = diamondData.quantile(0.25)
Q3 = diamondData.quantile(0.75)
IQR = Q3-Q1
diamondData_clean=diamondData[~((diamondData<(Q1-1.5*IQR))|(diamondData>(Q3+1.5*IQR))).any(axis=1)]
```





OUTLIERS REMOVED :)

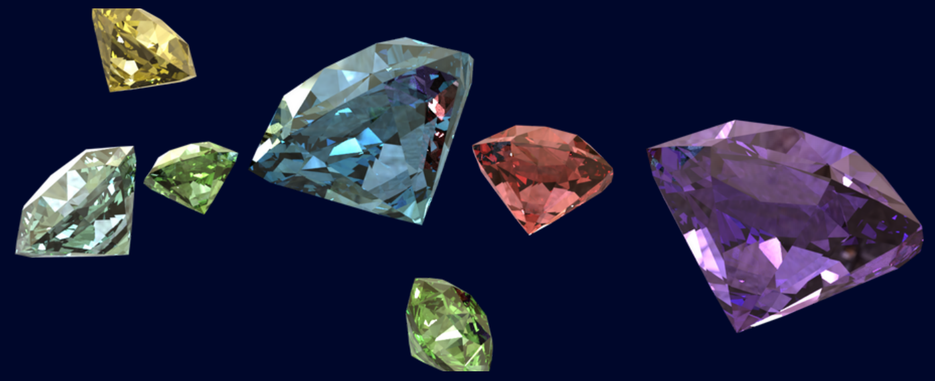
EXPLORATORY DATA ANALYSIS

- HEAT MAP
- BOX PLOT
- HISTO

	carat	depth	table	price	x	y	z
carat	1	0.028	0.18	0.92	0.98	0.95	0.96
depth	0.028	1	-0.3	-0.011	-0.025	-0.029	0.095
table	0.18	-0.3	1	0.13	0.2	0.18	0.15
price	0.92	-0.011	0.13	1	0.89	0.87	0.87
x	0.98	-0.025	0.2	0.89	1	0.97	0.98
y	0.95	-0.029	0.18	0.87	0.97	1	0.96
z	0.96	0.095	0.15	0.87	0.98	0.96	1

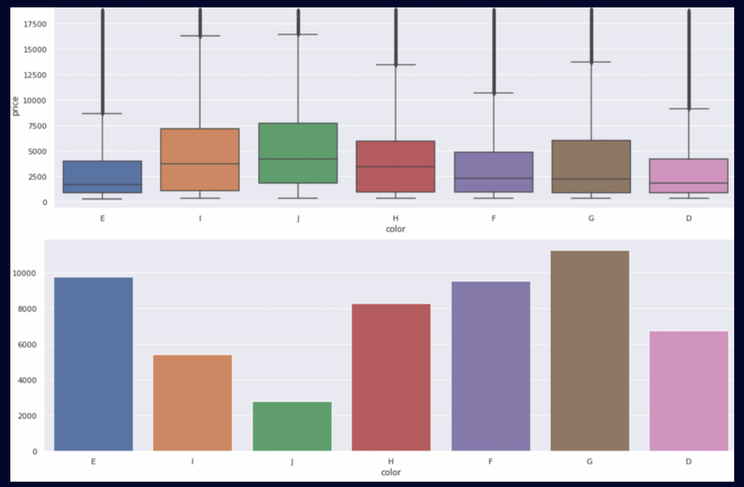
Checking for with weak correlation Univariate and Bivariate Data Exploration

- The carat(weight) & the dimensions are highly related to price
- Table & price doesn't have reliable relation
- X,Y, & Z seems to have a strong relation with price HOWEVER depth which constitutes x,y, and z doesn't has a significant relation with price

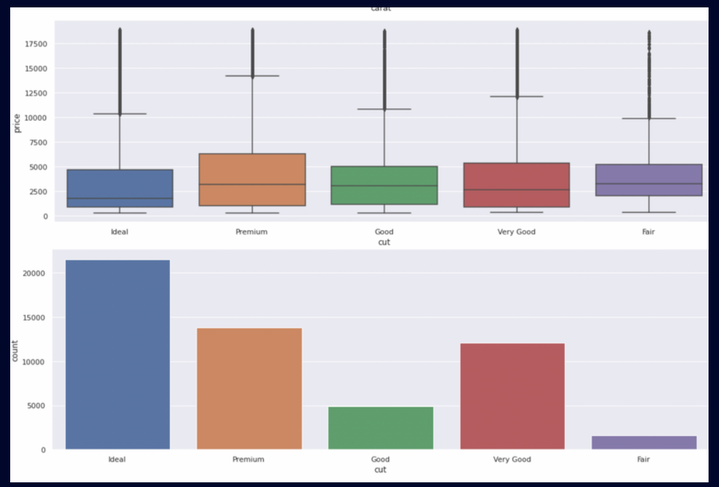


DATA CLEANING & EXTRACTION

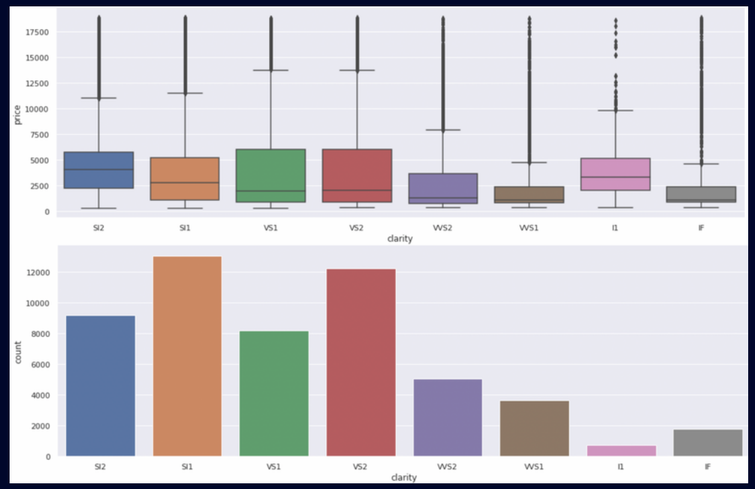
COLOR



CUT



CLARITY



- Since the larger stone = Rarer it is = More Expensive than the Total cost of Carat (weight) Diamonds of the Same Quality.
- A Higher Cut Quality => Diamond's Cost per Carat (weight) Increases.
- Although the Carat Weight of a Diamond has the Strongest Effect on Prices, the Cut can still Drastically Increase or Decrease its value With a Higher Cut Quality.
- Therefore, cut is an essential variable that would be extremely useful as it has capabilities of fluctuating price points.

SAMPLE COLLECTION

NARROWED VARIABLES

- Cut
- Color
- Clarity
- Carat -Carat refers to the Weight of the Stone, not the Size.

WHY? => Because the 4C's are parameters which are most accessible & essential information given to customers which can be reliably measured with specific values by consumers.

- Colorless Diamonds are Rarer and more Valuable because they appear Whiter and Brighter.
- Clarity refers to the absence of the Inclusions and Blemishes.

DETAILED REVIEWED QUESTION

HOW DOES THE DIFFERENT VARIABLES OF EACH SPECIFIC DIAMOND'S CLARITY, CUT, COLOUR AND CARAT CONTRIBUTE IN DETERMINING THEIR TRUE MARKET VALUE & RARITY

MACHINE LEARNING

Goal: Diamond price calculator

- Input: 4Cs, x, y, z
- Output: Price

Our approach:

- to handle categorical data:
  - Label Encoding
  - One Hot Encoding
- Machine Learning model: Regression
  - Linear Regression
  - Random Forest Regression

Compare and select technique that minimises error

MACHINE LEARNING

Goal: Diamond price calculator

- Input: 4Cs, x, y, z
- Output: Price

- dropped 'depth' and 'table' columns as they have very low correlation with Price
- LEdf: dataframe to be used for Label Encoding
- OHEdf: dataframe to be used for One Hot Encoding

```
diamondData_clean.drop(['depth','table'], axis=1, inplace= True)

LEdf = diamondData_clean.copy()
OHEdf = diamondData_clean.copy()
```

HANDLING CATEGORICAL DATA

Method 1: Label Encoding

```
#label encoding
LEdf['cut'].replace(['Ideal','Premium','Very Good','Good','Fair'],
                            [0,1,2,3,4], inplace=True)
LEdf['color'].replace(['D','E','F','G','H','I','J'],
                            [0,1,2,3,4,5,6], inplace=True)
LEdf['clarity'].replace(['IF','VVS1','VVS2','VS1','VS2','SI1','SI2','I1'],
                            [0,1,2,3,4,5,6,7], inplace=True)
```

manually replaced catagorical data such that 0 is the best instead of using alphabetical order

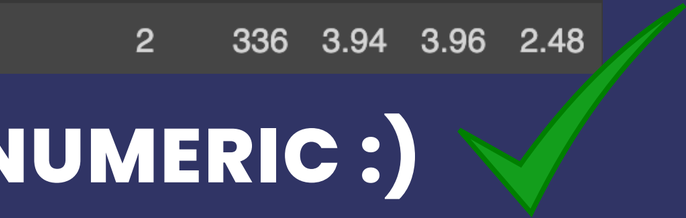
Convert categorical data into integers

	carat	cut	color	clarity	price	x	y	z
0	0.23	Ideal	E	SI2	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	326	3.89	3.84	2.31
3	0.29	Premium	I	VS2	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	336	3.94	3.96	2.48

→

	carat	cut	color	clarity	price	x	y	z
0	0.23	0	1	6	326	3.95	3.98	2.43
1	0.21	1	1	5	326	3.89	3.84	2.31
3	0.29	1	5	4	334	4.20	4.23	2.63
4	0.31	3	6	6	335	4.34	4.35	2.75
5	0.24	2	6	2	336	3.94	3.96	2.48

ALL NUMERIC :)



HANDLING CATEGORICAL DATA

Method 2: One Hot Encoding

1. Create a column for each category in columns with categorical data:
   - 1 indicates true
   - 0 indicates false
2. Drop columns with categorical data

```
for var in ['Ideal','Premium','Very Good','Good','Fair']:
    OHEdf[var] = np.where(OHEdf['cut']== var, 1, 0)
for var in ['D','E','F','G','H','I','J']:
    OHEdf[var] = np.where(OHEdf['color']== var, 1, 0)
for var in ['IF','VVS1','VVS2','VS1','VS2','SI1','SI2','I1']:
    OHEdf[var] = np.where(OHEdf['clarity']== var, 1, 0)
OHEdf.drop(['cut','color','clarity'], axis=1, inplace= True)
```

ALL NUMERIC :)

	carat	price	x	y	z
0	0.23	326	3.95	3.98	2.43
1	0.21	326	3.89	3.84	2.31
3	0.29	334	4.20	4.23	2.63
4	0.31	335	4.34	4.35	2.75
5	0.24	336	3.94	3.96	2.48

Ideal	Premium	Very Good	Good	Fair	...	I	J	IF	VVS1	VVS2	VS1	VS2	SI1	SI2	I1
1	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
0	1	0	0	0	...	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	...	1	0	0	0	0	0	1	0	0	0
0	0	0	1	0	...	0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	...	0	1	0	0	1	0	0	0	0	0

LABEL ENCODING VS ONE HOT ENCODING

CON: RANKS CATEGORICAL DATA

ranks using integers 0, 1, 2...,
which may not reflect the
true value of the data

prevents dummy variable trap

prevents ranking of categorical data

CON: DUMMY VARIABLE TRAP

leads to multicollinearity, where there is dependency between independent variable

MACHINE LEARNING MODELS

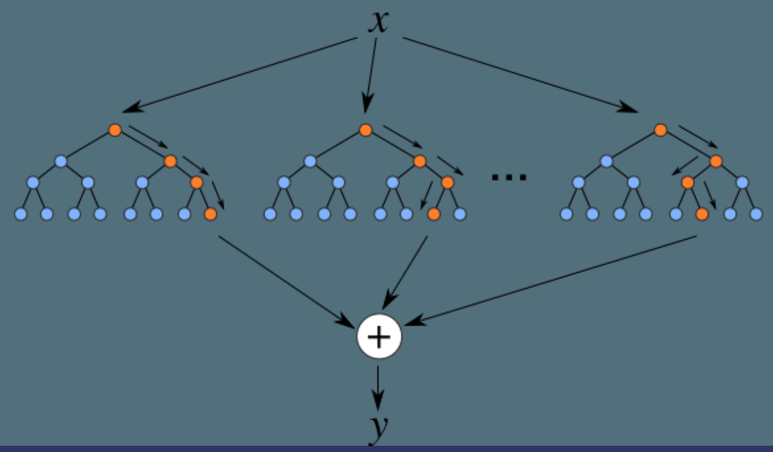
LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
```

- Price=sum of coefficients*predictors + y-intercept
- fits a linear model with coefficients that minimise the errors

RANDOM FOREST REGRESSOR

```
from sklearn.ensemble import RandomForestRegressor
```



- for both: split train and test data set in 8:2 ratio

```
from sklearn.model_selection import train_test_split
```

```
#split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, random_state=7)
```

COMPARING MODELS

RMSE measures the standard deviation of prediction errors

Label Encoding:

	Model	RMSE
0	Linear Regression	816.983817
1	Random Forest Regression	373.049901

One Hot Encoding:

	Model	RMSE
0	Linear Regression	767.769815
1	Random Forest Regression	378.390993

Linear Regression:

- RMSE lower with one hot encoding

Random Forest Regression:

- RMSE slightly lower with label encoding

Lowest RMSE value: Label Encoding + Random Forest Regression

PRICE CALCULATOR

1. train data set of Random Forest Regressor + Label Encoding
2. test model on inputs

```
y_pred = rf.predict([[Carat,Cut,Color,Clarity,x,y,z]])
print('price: ', y_pred)
```

SAMPLE:

```
from random import randint
```

```
print(diamondData_clean.iloc[[randint(0,47523)]])
```

	carat	cut	color	clarity	price	x	y	z
185	0.71	Ideal	G	SI1	2776	5.8	5.76	3.5

actual price: 2776

predicted price: 2656.6 .....

Diamond Price Calculator

CARAT:

Carat: 0.71

CUT: 0: Ideal // 1: Premium // 2: Very Good // 3: Good // 4: Fair

Cut: 0

COLOR: 0: D // 1: E // 2: F // 3: G // 4: H // 5: I // 6: J

Color: 3

CLARITY: 0: IF // 1: VVS1 // 2: VVS2 // 3: VS1 // 4: VS2 // 5: SI1 // 6: SI2 // 7: I1

Clarity: 5

DIMENSIONS (in mm): x: length // y: width // z: depth

x: 5.8

y: 5.76

z: 3.5

price: [2656.60333333]

CONCLUDING...

- Decent prediction
  - RMSE value: 373
  - Average value of 'Price': 3932

- Possible improvement for usability of the calculator:
  - remove dimensions of diamonds from the calculator as such information is not readily available

0.30ct Round Cut diamond

Cut: Good | Colour: H | Clarity: SI1

description of diamond from online site

THANK YOU!