Name: Ganesh Sunil Injinware

Roll No: 2337005

Batch: A

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
```

In [2]:
```
df=pd.read_csv('diamonds.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [4]:
```
df.shape
```

Out[4]:
```
(53940, 11)
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Unnamed: 0    0
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
price         0
x             0
y             0
z             0
dtype: int64
```

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Unnamed: 0    0
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
price         0
x             0
y             0
z             0
dtype: int64
```

In [7]:
```
df.dropna(inplace=True)
```

In [8]:
```
df.head()
```

Out[8]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [9]:
```
df.fillna(999,inplace=True)
```

In [10]:
```
df.head()
```

Out[10]:

	Unnamed: 0	carat	cut	color	clarity	depth	table	price	x	y	z
0	1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

In [11]:
```
df.dtypes
```

Out[11]:
```
Unnamed: 0      int64
carat         float64
cut            object
color          object
clarity        object
depth         float64
table         float64
price           int64
x             float64
y             float64
z             float64
dtype: object
```

In [12]:
```
df1=df.select_dtypes(['int','float'])
```

In [13]:
```
df1
```

Out[13]:

	Unnamed: 0	carat	depth	table	price	x	y	z
0	1	0.23	61.5	55.0	326	3.95	3.98	2.43
1	2	0.21	59.8	61.0	326	3.89	3.84	2.31
2	3	0.23	56.9	65.0	327	4.05	4.07	2.31
3	4	0.29	62.4	58.0	334	4.20	4.23	2.63
4	5	0.31	63.3	58.0	335	4.34	4.35	2.75
...	...	...	...	...	...	...	...	...
53935	53936	0.72	60.8	57.0	2757	5.75	5.76	3.50
53936	53937	0.72	63.1	55.0	2757	5.69	5.75	3.61
53937	53938	0.70	62.8	60.0	2757	5.66	5.68	3.56
53938	53939	0.86	61.0	58.0	2757	6.15	6.12	3.74
53939	53940	0.75	62.2	55.0	2757	5.83	5.87	3.64

53940 rows × 8 columns

In [14]:
```
cor=df1.corr()
```

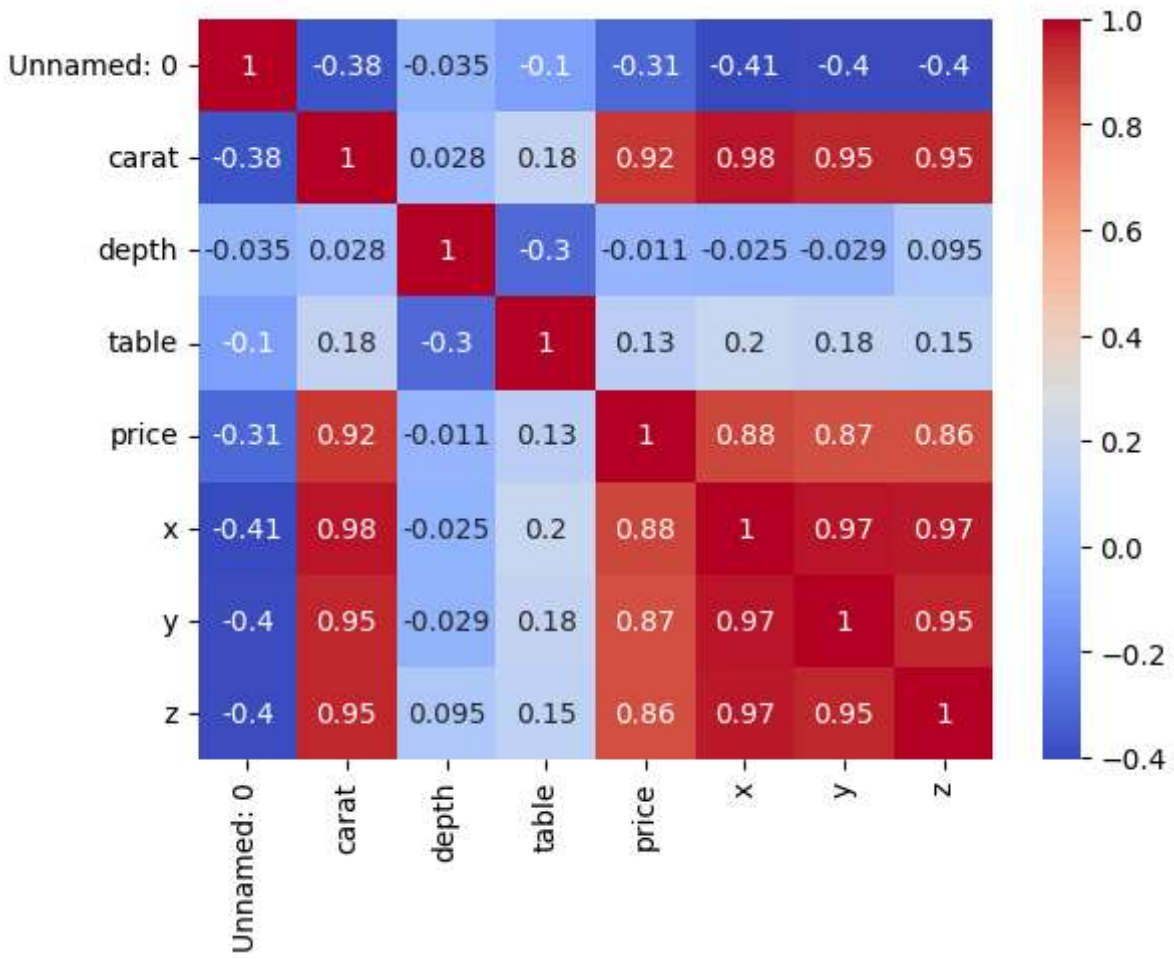
In [15]:
```
cor
```

Out[15]:

	Unnamed: 0	carat	depth	table	price	x	y	
Unnamed: 0	1.000000	-0.377983	-0.034800	-0.100830	-0.306873	-0.405440	-0.395843	-0.3
carat	-0.377983	1.000000	0.028224	0.181618	0.921591	0.975094	0.951722	0.9
depth	-0.034800	0.028224	1.000000	-0.295779	-0.010647	-0.025289	-0.029341	0.0
table	-0.100830	0.181618	-0.295779	1.000000	0.127134	0.195344	0.183760	0.1
price	-0.306873	0.921591	-0.010647	0.127134	1.000000	0.884435	0.865421	0.8
x	-0.405440	0.975094	-0.025289	0.195344	0.884435	1.000000	0.974701	0.9
y	-0.395843	0.951722	-0.029341	0.183760	0.865421	0.974701	1.000000	0.9
z	-0.399208	0.953387	0.094924	0.150929	0.861249	0.970772	0.952006	1.0

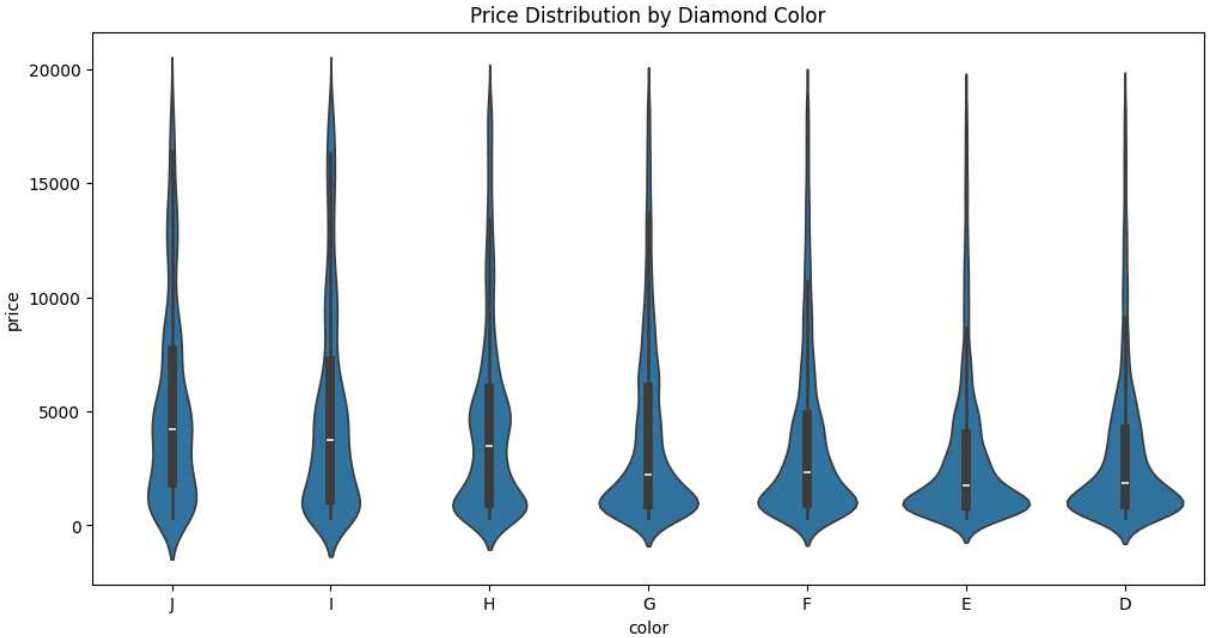
In [16]:
```
sns.heatmap(cor,cmap='coolwarm',annot=True)
```

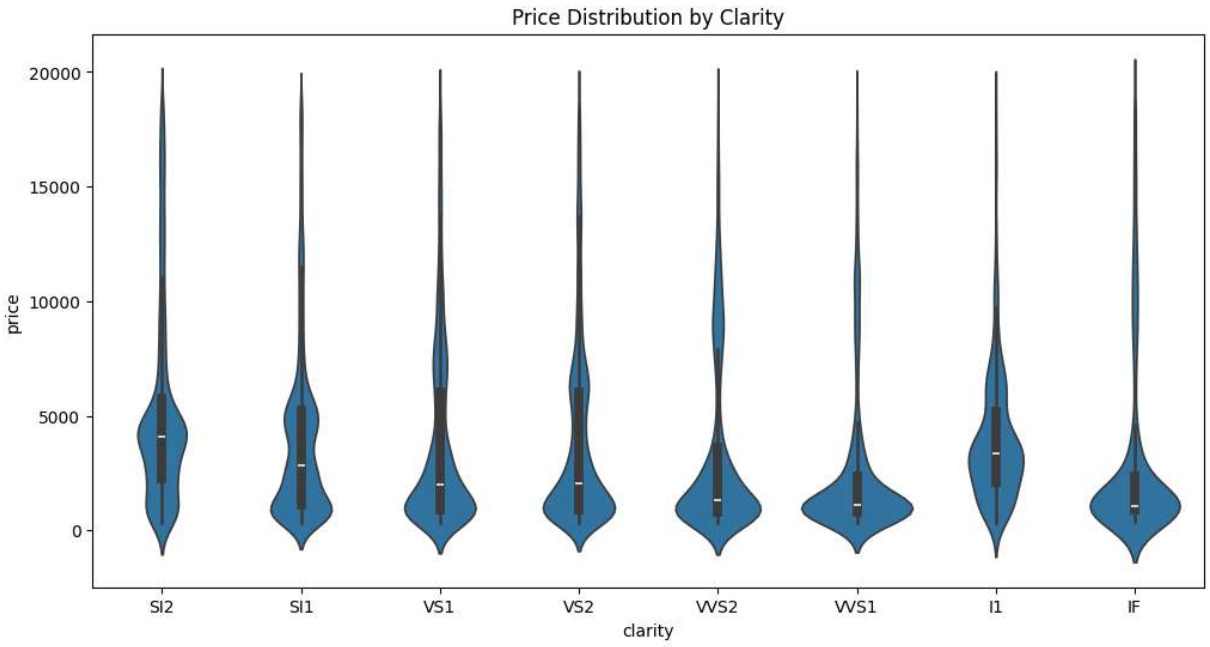
Out[16]: <Axes: >



In [17]:
```
plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x='cut', y='price')
```

```
plt.title('Price Distribution by Cut Quality')
plt.show()

plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x='color', y='price', order=['J', 'I', 'H', 'G', 'F', 'E', 'D
plt.title('Price Distribution by Diamond Color')
plt.show()

plt.figure(figsize=(12, 6))
sns.violinplot(data=df, x='clarity', y='price')
plt.title('Price Distribution by Clarity')
plt.show()
```





In [18]:
```
df.columns
```

Out[18]:
```
Index(['Unnamed: 0', 'carat', 'cut', 'color', 'clarity', 'depth', 'table',
       'price', 'x', 'y', 'z'],
      dtype='object')
```

In [19]:
```
data1 = df.copy()

# Applying label encoder to columns with categorical data
columns = ['cut','color','clarity']
label_encoder = LabelEncoder()
for col in columns:
    data1[col] = label_encoder.fit_transform(df[col])
data1.describe()
```

Out[19]:

	Unnamed: 0	carat	cut	color	clarity	depth
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	26970.500000	0.797940	2.553003	2.594197	3.835150	61.749405
std	15571.281097	0.474011	1.027708	1.701105	1.724591	1.432621
min	1.000000	0.200000	0.000000	0.000000	0.000000	43.000000
25%	13485.750000	0.400000	2.000000	1.000000	2.000000	61.000000
50%	26970.500000	0.700000	2.000000	3.000000	4.000000	61.800000
75%	40455.250000	1.040000	3.000000	4.000000	5.000000	62.500000
max	53940.000000	5.010000	4.000000	6.000000	7.000000	79.000000

In [20]:
```
X= data1.drop(["price"],axis =1)
y= data1["price"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, random_state=
```

LinearRegression

In [21]:
```
lr=LinearRegression()
```

In [22]:
```
lr.fit(X_train, y_train)
```

Out[22]:
```
▾ LinearRegression
LinearRegression()
```

In [23]:
```
y_pred=lr.predict(X_test)
```

In [24]:
```
lr.score(X_train,y_train)
```

Out[24]: 0.8856402556340337

In [25]:
```
lr.score(X_test,y_test)
```

Out[25]: 0.8860134363272643

In [47]:
```
lr_acc=lr.score(X_test,y_pred)
```

DecisionTreeRegressor

In [27]:
```
dt=DecisionTreeRegressor()
```

In [28]:
```
dt.fit(X_train, y_train)
```

Out[28]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [29]:
```
y_pred=dt.predict(X_test)
```

In [30]:
```
y_pred
```

Out[30]: array([559., 2200., 1238., ..., 761., 9837., 3742.])

In [31]:
```
dt.score(X_train,y_train)
```

Out[31]: 1.0

In [32]:
```
dt.score(X_test,y_test)
```

Out[32]: 0.9999663216826308

In [46]:
```
DT_acc=dt.score(X_test,y_pred)
```

RandomForestRegressor

In [34]:
```
df=RandomForestRegressor()
```

In [35]:
```
df.fit(X_train, y_train)
```

Out[35]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [36]:
```
y_pred=df.predict(X_test)
```

In [37]:
```
df.score(X_train,y_train)
```

Out[37]:
```
0.9999888821807392
```

In [38]:
```
df.score(X_test,y_test)
```

Out[38]:
```
0.9999621920484028
```

In [45]:
```
RFR_acc=df.score(X_test,y_pred)
```

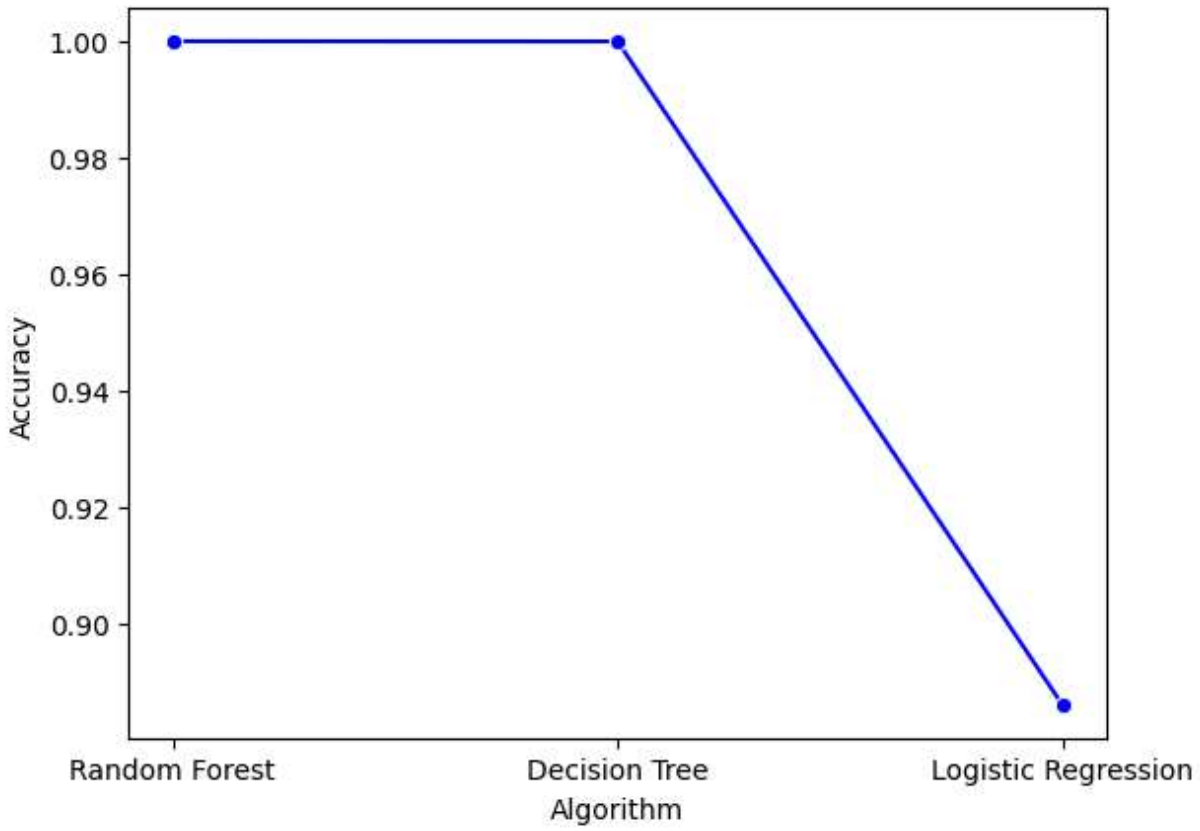
In [50]:
```
 accuracy_scores = {
    "Random Forest": RFR_acc,
    "Decision Tree": DT_acc,
    "Logistic Regression": lr_acc
}
```

In [51]:
```
results_df = pd.DataFrame(list(accuracy_scores.items()), columns=['Algorithm', 'Accur

results_df = results_df.sort_values(by='Accuracy', ascending=False).reset_index(drop=

print(results_df)
```

```
              Algorithm  Accuracy
0         Random Forest  1.000000
1         Decision Tree  0.999959
2   Logistic Regression  0.886095
```

In [53]:
```
sns.lineplot(data=results_df, x='Algorithm', y='Accuracy', marker='o', color='b')
```

Out[53]:
```
<Axes: xlabel='Algorithm', ylabel='Accuracy'>
```


1.00
0.98
0.96
0.94
0.92
0.90
Accuracy
Random Forest
Decision Tree
Logistic Regression
Algorithm

In [ ]: