Simple Linear Regression Project (Gold Price Prediction)

Of all the precious metals, gold is the most popular as an investment. Investors generally buy gold as a way of diversifying risk, especially through the use of futures contracts and derivatives. The gold market is subject to speculation and volatility as are other markets. Compared to other precious metals used for investment, gold has been the most effective safe haven across a number of countries.

The Dataset contain gold prices (in USD) from 2001 to 2019. Our goal is to predict where the gold prices will be in the coming years

Import the necessary libraries

In [151…

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Read 'gold_price_usd.csv' & store it in a variable

In [152…

```
df=pd.read_csv("gold_price_usd.csv")
```

View the first 5 rows

In [153…

```
df.head()
```

Out[153]:

	Date	USD (AM)
0	2001-01-02	272.80
1	2001-01-03	269.00
2	2001-01-04	268.75
3	2001-01-05	268.00
4	2001-01-08	268.60

Check the information

In [154…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4718 entries, 0 to 4717
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Date      4718 non-null   object 
 1   USD (AM)  4718 non-null   float64
dtypes: float64(1), object(1)
memory usage: 73.8+ KB
```

Find the columns

In [155…

```
df.columns
```

Out[155]:

```
Index(['Date', 'USD (AM)'], dtype='object')
```

Rename USD (AM) to Price

In [156…

```
df.rename(columns={"USD (AM)":"price"},inplace="true")
df
```

Out[156]:

	Date	price
0	2001-01-02	272.80
1	2001-01-03	269.00
2	2001-01-04	268.75
3	2001-01-05	268.00
4	2001-01-08	268.60
...	...	...
4713	2019-08-27	1531.85
4714	2019-08-28	1541.75
4715	2019-08-29	1536.65
4716	2019-08-30	1526.55
4717	2019-09-02	1523.35

4718 rows × 2 columns

Check if there are any missing values in the dataset

In [157…

```
df.isna().sum()
```

Out[157]:

```
Date     0
price    0
dtype: int64
```

Gather the basic statistical information about the dataset

In [158…

```
df.describe()
```

Out[158]:

	price
count	4718.000000
mean	959.990812
std	449.456217
min	256.700000
25%	449.112500
50%	1113.125000
75%	1293.750000
max	1896.500000

Convert Date column from object to datetime format

In [159…

```
df["year"]=pd.DatetimeIndex(df["Date"]).year
df["month"]=pd.DatetimeIndex(df["Date"]).month
df
```

Out[159]:

	Date	price	year	month
0	2001-01-02	272.80	2001	1
1	2001-01-03	269.00	2001	1
2	2001-01-04	268.75	2001	1
3	2001-01-05	268.00	2001	1
4	2001-01-08	268.60	2001	1
...	...	...	...	...
4713	2019-08-27	1531.85	2019	8
4714	2019-08-28	1541.75	2019	8
4715	2019-08-29	1536.65	2019	8
4716	2019-08-30	1526.55	2019	8
4717	2019-09-02	1523.35	2019	9

4718 rows × 4 columns

In [160…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4718 entries, 0 to 4717
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Date    4718 non-null   object 
 1   price   4718 non-null   float64
 2   year    4718 non-null   int64  
 3   month   4718 non-null   int64  
dtypes: float64(1), int64(2), object(1)
memory usage: 147.6+ KB
```

Create a new column with Year

In [163…

```
df["year"]=pd.DatetimeIndex(df["Date"]).year
df
```

Out[163]:

	Date	price	year	month
0	2001-01-02	272.80	2001	1
1	2001-01-03	269.00	2001	1
2	2001-01-04	268.75	2001	1
3	2001-01-05	268.00	2001	1
4	2001-01-08	268.60	2001	1
...	...	...	...	...
4713	2019-08-27	1531.85	2019	8
4714	2019-08-28	1541.75	2019	8
4715	2019-08-29	1536.65	2019	8
4716	2019-08-30	1526.55	2019	8
4717	2019-09-02	1523.35	2019	9

4718 rows × 4 columns

Create a new column with Months

In [26]:

```
df["month"]=pd.DatetimeIndex(df["Date"]).month
```

In [164…

```
df
```

Out[164]:

	Date	price	year	month
0	2001-01-02	272.80	2001	1
1	2001-01-03	269.00	2001	1
2	2001-01-04	268.75	2001	1
3	2001-01-05	268.00	2001	1
4	2001-01-08	268.60	2001	1
...	...	...	...	...
4713	2019-08-27	1531.85	2019	8
4714	2019-08-28	1541.75	2019	8
4715	2019-08-29	1536.65	2019	8
4716	2019-08-30	1526.55	2019	8
4717	2019-09-02	1523.35	2019	9

4718 rows × 4 columns

See all the years and Months in our dataset

In [37]:

```
df[["year"]]
```

Out[37]:

	year
0	2001
1	2001
2	2001
3	2001
4	2001
...	...
4713	2019
4714	2019
4715	2019
4716	2019
4717	2019

4718 rows × 1 columns

In [38]:

```
df[["month"]]
```

Out[38]:

	month
0	1
1	1
2	1
3	1
4	1
...	...
4713	8
4714	8
4715	8
4716	8
4717	9

4718 rows × 1 columns

Visualization

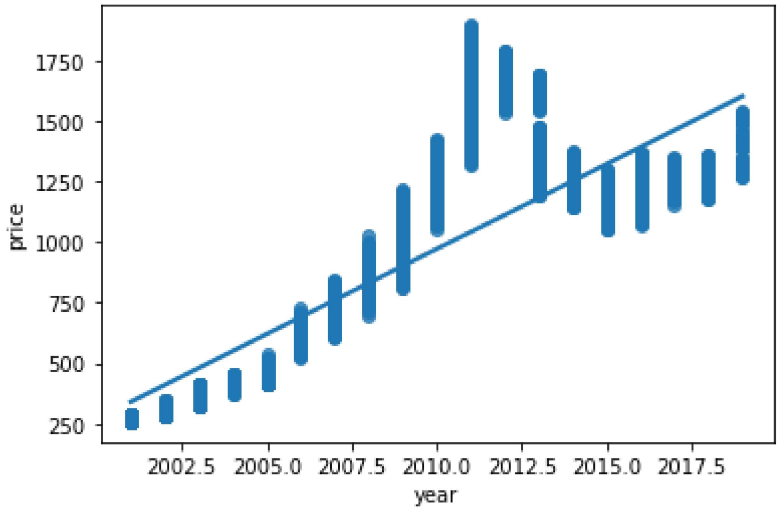
Create a regression plot with x-axis as years and y-axis as Price

In [33]:

```
sns.regplot(x=df["year"],y=df["price"])
```

Out[33]:

```
<AxesSubplot:xlabel='year', ylabel='price'>
```



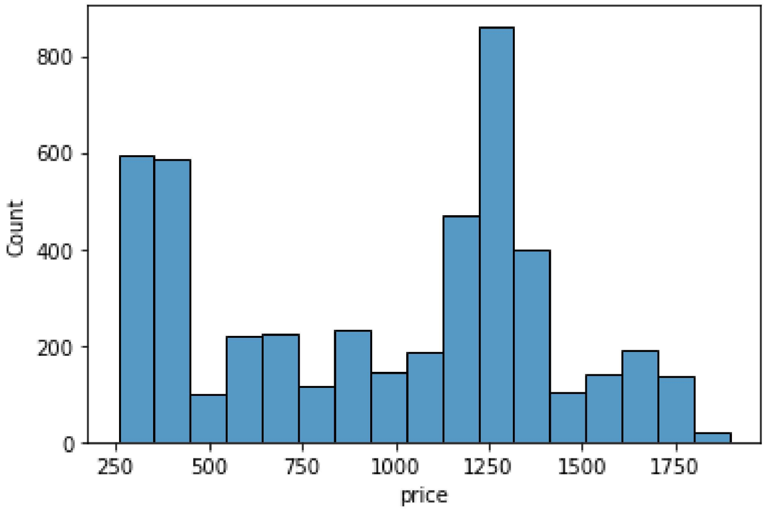
Plot a histplot to find the variation in price

In [32]:

```
sns.histplot(x=df["price"])
```

Out[32]:

```
<AxesSubplot:xlabel='price', ylabel='Count'>
```



Assign year and price in x and y variables

In [119…

```
X = df[['year']]
y = df['price']
```

Split the data into traning and testin set

We will train our model on the training set and then use the test set to evaluate the model

In [120…

```
# import train_test split
from sklearn.model_selection import train_test_split
```

In [121…

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_s
```

Train Data

In [122…

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

Create Linear Regression Model

In [123…

```
model=LinearRegression()
```

Train the model

In [124…

```
model.fit(X_train,y_train)
```

Out[124]:

```
LinearRegression()
```

Check the score of our model

In [125…

```
model.score(X_train,y_train)
```

Out[125]:

```
0.7048691960223057
```

Check the coefficient and Intercept

In [128…

```
#print the intercept
print(model.intercept_)
```

```
-140074.32374779382
```

In [129…

```
#print the coefficent
print(model.coef_)
```

```
[70.17366927]
```

Make Prediction with Test data

In [134…

```
# Also store the predicted values in a variable
y_pred=model.predict(X_test)
len(y_pred)
```

Out[134]:

```
1416
```

Create a new dataframe with actual and predicted values with year(X_test) as index

In [ ]:

```

```

Check the mean absolute error, mean square error

In [137…

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [138…

```
# Mean absolute error
mean_absolute_error(y_test,y_pred)
```

Out[138]:

```
186.2427389387351
```

In [139…

```
# Mean squared error
mean_squared_error(y_test,y_pred)
```

Out[139]:

```
58032.97376893088
```

Predict the prices for the following years

- 2025, 2026, 2027, 2028, 2030

In [144…

```
data=[2025,2026,2027,2028,2030]
df=pd.DataFrame(data,columns=["year"])
df
```

Out[144]:

	year
0	2025
1	2026
2	2027
3	2028
4	2030

Great Job!