Gold rate prediction using SVR

In [1]:
```
#Importing the Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
```

In [2]:
```
#Reading the dataset
gold_dataset = pd.read_csv("gold_rate_history.csv", index_col='Date', header='infer', parse_dates=True, infer_da
```

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

Out[3]:

Date	Country	State	Location	Pure Gold (24 k)	Standard Gold (22 K)
2006-01-02	India	Tamilnadu	Chennai	768.0	711.0
2006-01-03	India	Tamilnadu	Chennai	770.5	713.0
2006-01-04	India	Tamilnadu	Chennai	784.5	726.0
2006-01-05	India	Tamilnadu	Chennai	782.5	725.0
2006-01-06	India	Tamilnadu	Chennai	776.0	719.0

Data Pre-Processing

1. Checking for missing values

In [4]:
```
gold_dataset.isna().sum()
```

Out[4]:
```
Country                 0
State                   0
Location                0
Pure Gold (24 k)        0
Standard Gold (22 K)    0
dtype: int64
```

Conclusion: There are no missing values in the dataset

2. Dropping the unnecessary columns

In [5]:
```
# Dropping Unwanted Columns
unwanted_cols = ['Country','State','Location']
gold_dataset.drop(unwanted_cols, axis=1, inplace=True)
gold_dataset.head()
```

Out[5]:

Date	Pure Gold (24 k)	Standard Gold (22 K)
2006-01-02	768.0	711.0
2006-01-03	770.5	713.0
2006-01-04	784.5	726.0
2006-01-05	782.5	725.0
2006-01-06	776.0	719.0

3. Renaming the columns

In [6]:
```
# Renaming Columns
gold_dataset.rename(columns={"Pure Gold (24 k)": "Pure_Gold_24k", "Standard Gold (22 K)": "Std_Gold_22k",},inpla
gold_dataset.head()
```

Out[6]:

Date	Pure_Gold_24k	Std_Gold_22k
2006-01-02	768.0	711.0
2006-01-03	770.5	713.0
2006-01-04	784.5	726.0
2006-01-05	782.5	725.0
2006-01-06	776.0	719.0

4. Data Scaling

In [7]:
```
#For Nnormalizing real-valued input and output variables
from sklearn.preprocessing import MinMaxScaler
cols = gold_dataset.columns
idx = gold_dataset.index
scaler = MinMaxScaler(feature_range=(0,1))

df_scaled = pd.DataFrame(scaler.fit_transform(gold_dataset), columns=cols, index=idx)
```

Exploratory Data Analysis

1. Dataset Shape

In [8]:
```
gold_dataset.shape
```

Out[8]:
```
(4971, 2)
```

Conclusion: There are 4971 rows and 2 columns in the dataset

2. Dataset Info

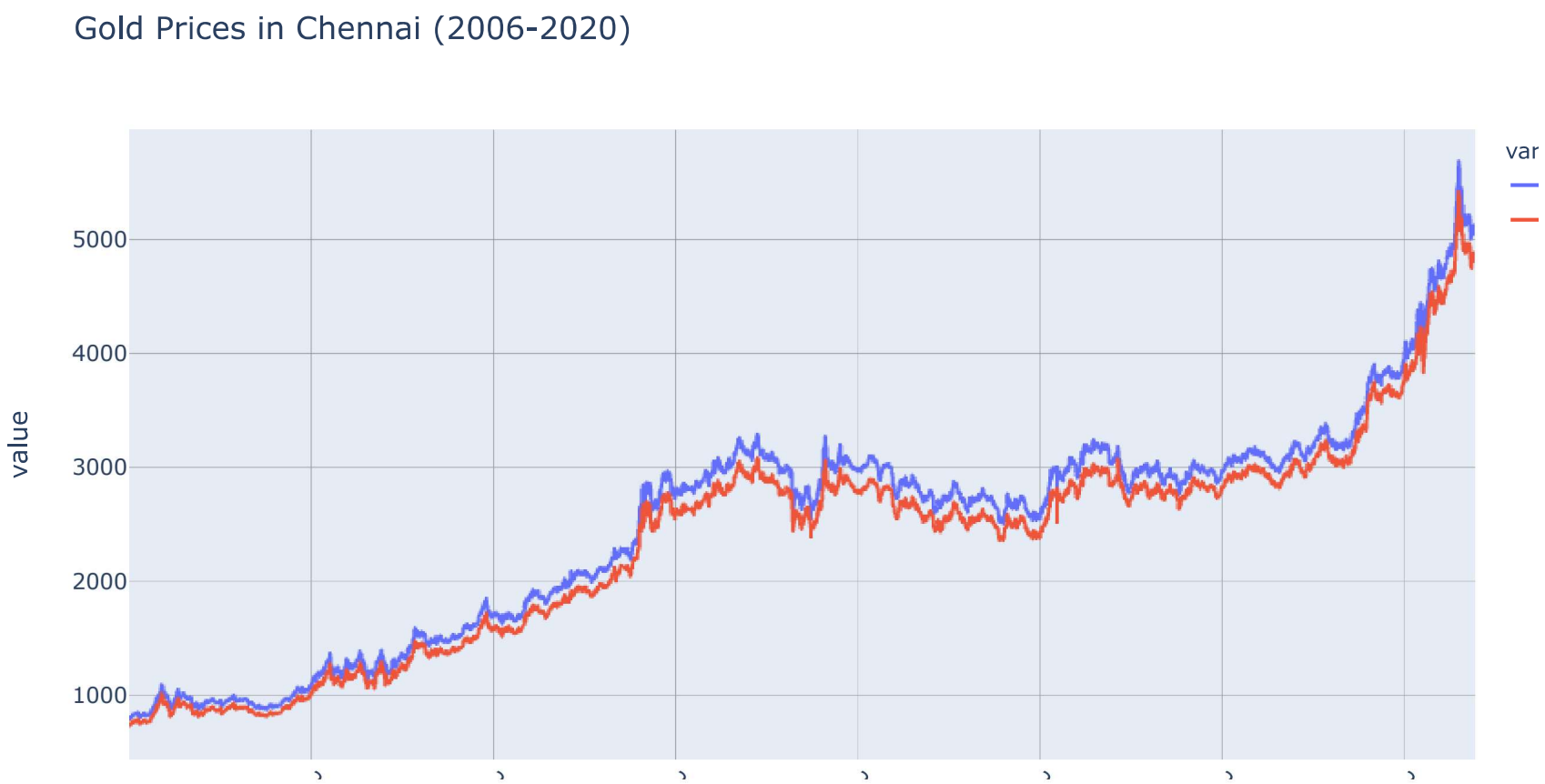
In [9]:
```
gold_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 4971 entries, 2006-01-02 to 2020-10-10
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Pure_Gold_24k  4971 non-null   float64
 1   Std_Gold_22k   4971 non-null   float64
dtypes: float64(2)
memory usage: 116.5 KB
```

3. Gold price in chennai from 2006 to 2020

In [10]:
```
fig = px.line(gold_dataset, x=gold_dataset.index, y=gold_dataset.columns, title='Gold Prices in Chennai (2006-20

fig.update_xaxes(tickangle=45)
fig.show()
```



Conclusion: The prive of gold foe 24k as well as 22k has increased every year. There is a positive relation.

Feature Extraction

In [11]:
```
df_svr = df_scaled.copy()

# Converting Date Index to Column for Feature Extraction
df_svr.reset_index(level=0, inplace=True)

# Time Feature Extraction
df_svr['year']=df_svr['Date'].dt.year
df_svr['month']=df_svr['Date'].dt.month
df_svr['day']=df_svr['Date'].dt.day
df_svr['quarter']=df_svr['Date'].dt.quarter
df_svr['weekofyear']=df_svr['Date'].dt.weekofyear
df_svr['weekday']=df_svr['Date'].dt.weekday

# Dropping Date Column
df_svr.drop('Date',axis=1,inplace=True)

df_svr
```

Out[11]:

	Pure_Gold_24k	Std_Gold_22k	year	month	day	quarter	weekofyear	weekday
0	0.000000	0.000000	2006	1	2	1	1	0
1	0.000508	0.000425	2006	1	3	1	1	1
2	0.003354	0.003188	2006	1	4	1	1	2
3	0.002948	0.002976	2006	1	5	1	1	3
4	0.001626	0.001700	2006	1	6	1	1	4
...	...	...	...	...	...	...	...	...
4966	0.881277	0.881828	2020	10	6	4	41	1
4967	0.869689	0.870351	2020	10	7	4	41	2
4968	0.870705	0.871201	2020	10	8	4	41	3
4969	0.879244	0.879702	2020	10	9	4	41	4
4970	0.885546	0.886291	2020	10	10	4	41	5

4971 rows × 8 columns

Feature Engineering

In [12]:
```
# Feature Engineering
features = ['year','month','day','quarter','weekofyear','weekday']

target_24k = ['Pure_Gold_24k']
target_22k = ['Std_Gold_22k']

X = df_svr[features]
y_24k = df_svr[target_24k]
y_22k = df_svr[target_22k]

#validation size
size = 0.1

#Splitting the dataset
from sklearn.model_selection import train_test_split
X_train_24k, X_val_24k, y_train_24k, y_val_24k = train_test_split(X, y_24k, test_size=size, random_state=42)
X_train_22k, X_val_22k, y_train_22k, y_val_22k = train_test_split(X, y_22k, test_size=size, random_state=42)
```

Model Building using SVR

In [13]:
```
from sklearn.svm import SVR
regressor = SVR(kernel = 'linear')

#For 24k gold
regressor.fit(X_train_24k, np.ravel(y_train_24k))
y_pred_24k = regressor.predict(X_val_24k)

#For 22k gold
regressor.fit(X_train_22k, np.ravel(y_train_22k))
y_pred_22k = regressor.predict(X_val_22k)
```

Model Evaluation

In [14]:
```
from sklearn import metrics as mt

#For 24k gold
rmse = mt.mean_squared_error(y_val_24k,y_pred_24k)
r2_score = mt.r2_score(y_val_24k,y_pred_24k)

print(" Model evaluation for 24k gold")
print("Root Mean Square Error: ", rmse)
print("R Squared value: ", r2_score)


#For 22k gold
rmse = mt.mean_squared_error(y_val_22k,y_pred_22k)
r2_score = mt.r2_score(y_val_22k,y_pred_22k)

print("\n")
print("Model evaluation for 22k gold")
print("Root Mean Square Error: ", rmse)
print("R Squared value: ", r2_score)
```

```
 Model evaluation for 24k gold
Root Mean Square Error:  0.0069129179790862255
R Squared value:  0.8203559306350131


Model evaluation for 22k gold
Root Mean Square Error:  0.0066595747378093454
R Squared value:  0.8275928201607174
```

Conclusion:

For 24k gold the root mean square error is 0.006 which indicated as better fit and the R squared value is 0.8203 which indicates that 82.03% of the data fits the regression model.

For 22k gold the root mean square error is 0.006 which indicated as better fit and the R squared value is 0.8275 which indicates that 82.75% of the data fits the regression model.