WQD7005 Data Mining

Video 4: https://www.loom.com/share/5a35913897ad420e88c097c3a57fcad0

Milestone 4: Interpretation of data & Communication of Insight of data

The module needed are imported & the dataset is read.

```
# Import module
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import seaborn
import yfinance as yf

# Read Data
df = pd.read_csv("C:\\Users\\User\\Desktop\\data_mining\\data_mining\\goldpric
e_12-Mar-2020_22-26-27.csv")
print(df.head(10))
```

Output:

The top 10 rows of dataset are printed out.

	Date	Price	Open	High	Low	Volume	Change %
0	Mar 12 2020	1580.7	1642.9	1650.0	1574.45	-	-3.75%
1	Mar 11 2020	1642.3	1649.3	1671.8	1632.40	404.35K	-1.08%
2	Mar 10 2020	1660.3	1679.6	1681.3	1641.10	385.48K	-0.92%
3	Mar 09 2020	1675.7	1692.6	1704.3	1658.00	504.16K	0.20%
4	Mar 06 2020	1672.4	1673.1	1692.8	1642.40	659.63K	0.26%
5	Mar 05 2020	1668.0	1638.2	1675.5	1635.60	363.00K	1.52%
6	Mar 04 2020	1643.0	1640.1	1654.3	1632.60	313.34K	-0.09%
7	Mar 03 2020	1644.4	1586.0	1650.5	1585.90	466.53K	3.11%
8	Mar 02 2020	1594.8	1592.8	1612.1	1576.30	443.53K	1.79%
9	Feb 28 2020	1566.7	1646.1	1651.0	1564.00	745.84K	-4.61%

The date format is changed & the dataset is under pre-processing.

```
# Change the format of "Date"
df["Date"] = pd.to_datetime(df["Date"]).dt.strftime('%Y%m%d')
# Remove the "," in the "Price", "Open', 'Low', and 'High' column and change t
he "string" type to "float"
df['Price']=df['Price'].astype(str).str.replace(',', '').astype(float)
df['Open']=df['High'].astype(str).str.replace(',', '').astype(float)
df['Low']=df['Low'].astype(str).str.replace(',', '').astype(float)
df['High']=df['High'].astype(str).str.replace(',', '').astype(float)
df['Volume']=df['Volume'].replace({'K': '*1e3', '-
': '1'}, regex=True).map(pd.eval)
df['Change %']=df['Change %'].replace({'%': '*1e-2'}, regex=True).map(pd.eval)
```

```
df['Volume'] = df['Volume'].replace(1.0,np.NaN)

# Select Date and Price only
df_price = df[['Date', 'Price']]
df_price.dropna()
print(df_price.head(10))

# Check if there contain null in the attributes
print(df_price.isnull().any())

# Set up the function of selecting range of date and commodity type
def extract_data(start_date,end_date,commodity_type):
  if commodity_type == "Gold":
    return df_price[(df_price.Date>=start_date)&(df_price.Date<=end_date)]
  elif commodity_type == "Silver":
    return df_price[(df_price.Date>=start_date)&(df_price.Date<=end_date)]

# Specify users inputs, you can change the start date and end data to extract
data from G-2019
# In this case, I select the data from 2019-01-01 to 2019-11-
12 and gold as my input
start_date = "20190101"
end_date = "20191112"
commodity_type = "Trends of Gold Price"
df_gold= extract_data(start_date, end_date, commodity_type)

# Change "Date" as index and sort the data
df_price['Date'] =pd.to_datetime(df_price.Date)
df_sort = df_price.sort_values('Date')
```

Output:

Only ‘Date’ & ‘Price’ are selected. The data is checked if there contain null in the attributes.

```
         Date   Price
0  2020-03-12  1580.7
1  2020-03-11  1642.3
2  2020-03-10  1660.3
3  2020-03-09  1675.7
4  2020-03-06  1672.4
5  2020-03-05  1668.0
6  2020-03-04  1643.0
7  2020-03-03  1644.4
8  2020-03-02  1594.8
9  2020-02-28  1566.7
Date     False
Price    False
dtype: bool
```

An explanatory variable is a variable that is manipulated to determine the value of the Gold Price the next day. Simply, they are the features which we want to use to predict the Gold Price. The explanatory variables in this strategy are the moving averages for past 3 days and 9 days.

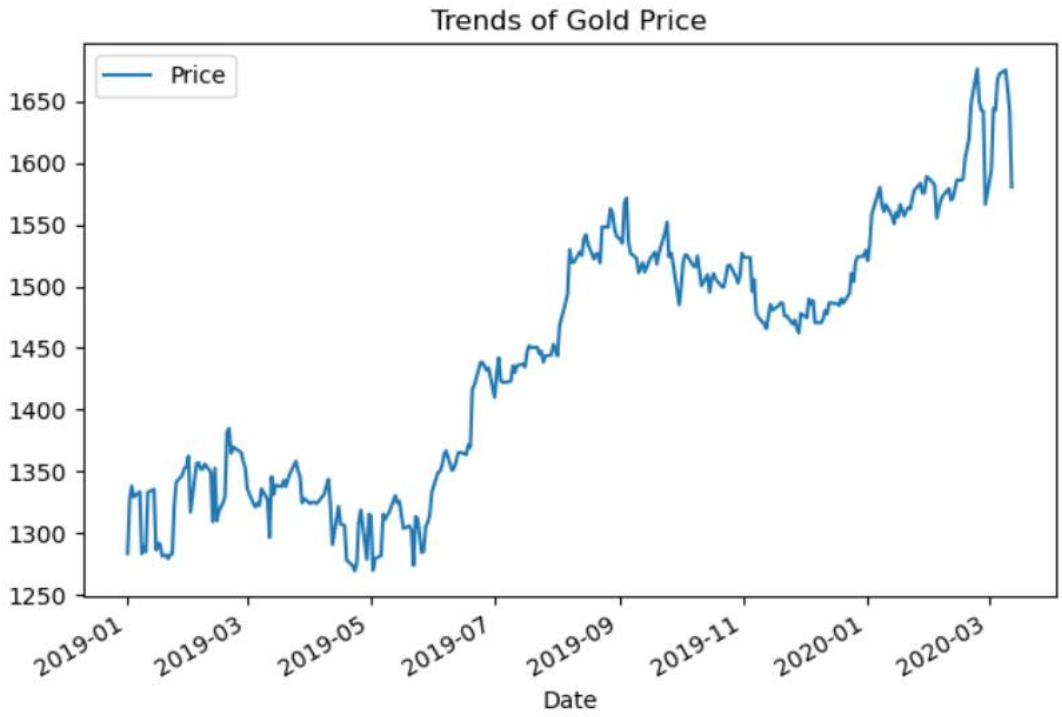
The data is visualized and split into training set & test set.

```
# Visualize the Data
df_sort.plot(x='Date',y='Price')
plt.title(commodity_type)
plt.show()

df_sort['S_3'] = df_sort['Price'].shift(1).rolling(window=3).mean()
df_sort['S_9']= df_sort['Price'].shift(1).rolling(window=9).mean()
df_sort= df_sort.dropna()
X = df_sort[['S_3','S_9']]
print(X.head())

y = df_sort['Price']
print(y.head())

t=.8
t = int(t*len(df_sort))
# Train dataset
X_train = X[:t]
y_train = y[:t]
# Test dataset
X_test = X[t:]
y_test = y[t:]
```

Output:

```
            S_3         S_9
309  1302.400000  1311.283333
308  1317.733333  1317.055556
307  1318.233333  1312.500000
306  1304.366667  1307.344444
305  1289.600000  1303.100000
```

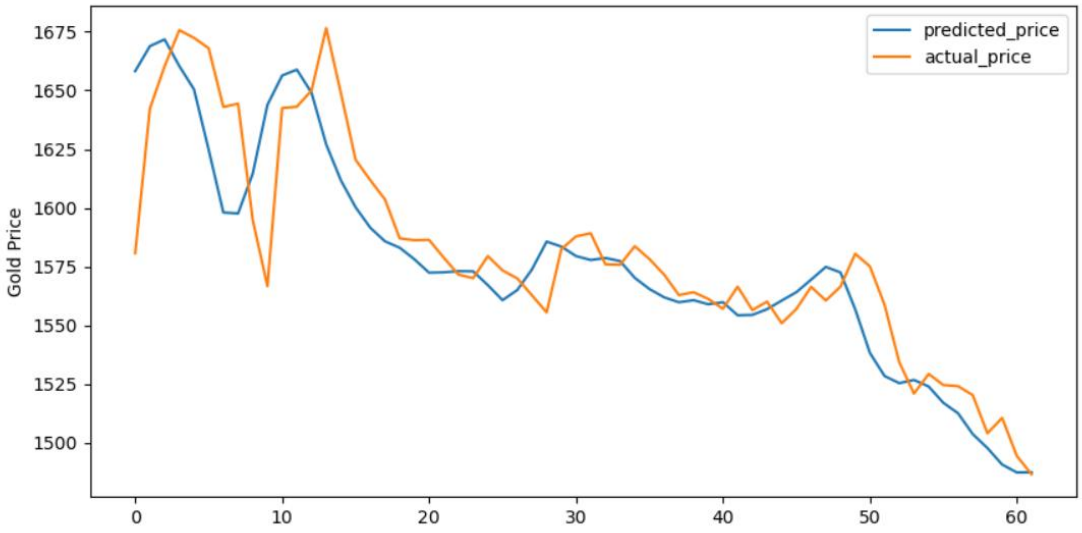
Output:



Linear regression algorithm is used to predict the future gold price.

```
linear = LinearRegression().fit(X_train,y_train)
print ("Gold Price =", round(linear.coef_[0],2), "* 3 Days Moving Average", ro
und(linear.coef_[1],2), "* 9 Days Moving Average +", round(linear.intercept_,2
))

predicted_price = linear.predict(X_test)
predicted_price = pd.DataFrame(predicted_price,index=y_test.index,columns = ['
price'])
predicted_price.plot(figsize=(10,5))
y_test.plot()
plt.legend(['predicted_price','actual_price'])
plt.ylabel("Gold ETF Price")
plt.show()

r2_score = linear.score(X[t:],y[t:])*100
float("{0:.2f}".format(r2_score))
```

Output:



The equation for linear regression:

```
Gold Price = 1.06 * 3 Days Moving Average -0.07 * 9 Days Moving Average + 23.31
```