LỜI NÓI ĐẦU

Trong lịch sử, vàng đã được sử dụng như một hình thức tiền tệ ở nhiều nơi trên thế giới bao gồm Việt Nam. Hiện tại, kim loại quý như vàng được giữ tại ngân hàng trung ương của tất cả các quốc gia để đảm bảo thanh toán nợ nước ngoài, đồng thời kiểm soát lạm phát, phản ánh sức mạnh tài chính của đất nước. Gần đây, các nền kinh tế mới nổi trên thế giới, chẳng hạn như Trung Quốc, Nga và Ấn Độ là những nước mua vàng lớn, trong khi Hoa Kỳ, Nam Phi và Úc là những quốc gia bán vàng nhiều nhất.

Dự báo tăng và giảm tỷ giá vàng hàng ngày, có thể giúp các nhà đầu tư quyết định khi nào nên mua (hoặc bán) vàng. Nhưng giá vàng còn phụ thuộc vào nhiều yếu tố như giá các kim loại quý khác, giá của dầu thô, giao dịch chứng khoán, giá trái phiếu, tỷ giá hối đoái, …

Trong dự án này sẽ dự báo tỷ giá vàng bằng cách sử dụng bộ tính năng toàn diện nhất và sẽ áp dụng các thuật toán khác nhau để dự báo và so sánh kết quả của chúng.

TỔNG QUAN

Thách thức trong dự án này là dự đoán chính xác giá đóng cửa được điều chỉnh trong tương lai. Vấn đề là một vấn đề hồi quy, bởi vì giá trị đầu ra là giá đóng cửa được điều chỉnh trong dự án này là giá trị liên tục. Nhiều nghiên cứu khác nhau đã được các nhà nghiên cứu thực hiện để dự báo tỷ giá vàng bằng cách sử dụng các máy học khác nhau các thuật toán với mức độ thành công khác nhau nhưng cho đến gần đây khả năng xây dựng các mô hình này đã được giới hạn trong giới học thuật. Giờ đây, với các thư viện như Scikit-learning, bất kỳ ai cũng có thể xây dựng các mô hình dự đoán mạnh mẽ. Đối với dự án này, tôi sẽ sử dụng các mô hình học máy tuyến tính, tập hợp và tăng cường khác nhau để dự đoán giá đóng cửa đã điều chỉnh của Nasdaq sử dụng bộ dữ liệu về giá vàng theo thời gian thực tính bằng USD từ năm 2012 đến năm 2022 với đơn vị giá vàng được tính với đơn vị là 1 ounce (là đơn vị lượng vàng hay còn được gọi là cây vàng) .

DỰ ĐOÁN GIÁ VÀNG SỬ DỤNG HỒI QUY TUYẾN TÍNH

<u>Bước 1</u>: Nhập các thư viện cần thiết

Sử dụng Pandas để nhập dữ liệu, Matplotlib và Seaborn để trực quan hóa dữ liệu, sklearn cho các thuật toán, tran_test_split để chia tập dữ liệu trong tập kiểm tra và traing set, sử dụng mean squared error, r2 score để đánh giá mô hình.

In [1]:
```
#For general data manipulation and vistualisation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime


#For Preporcession and cleaning step
from sklearn.preprocessing import MinMaxScaler

#For training setp
from sklearn.model_selection import train_test_split
import tensorflow.keras
from keras.models import Sequential
from keras.layers import Dense

#For Evaluation
from sklearn.metrics import accuracy_score
```

<u>Bước 2</u>: Tải bộ dữ liệu

Lấy dữ liệu từ trang:

https://www.nasdaq.com/marketactivity/commodities/gc%3Acmx

In [2]:
```
dateparse = lambda x: datetime.strptime(x, '%m/%d/%Y')
file = 'C:/Users/doian/Desktop/LuyenCode/predict-gold-price/gold.csv'
df = pd.read_csv(file, parse_dates=['Date'], date_parser=dateparse)
```

Bộ dữ liệu này bao gồm giá vàng theo thời gian thực tính bằng USD (từ năm 2012 đến năm 2022). Dựa trên bộ dữ liệu giá vàng được tính với đơn vị là 1 ounce (là đơn vị lượng vàng hay còn được gọi là cây vàng).

Date - Ngày ghi giá
Close - Ngày đóng giá vàng tính bằng USD
Volume - Tổng lượng mua và bán của hàng hoá vàng
Open - Giá mở của vàng vào ngày cụ thể đó
High - Giá vàng cao nhất của ngày cụ thể đó
Low - Giá vàng thấp nhất vào ngày cụ thể đó.

```
In [3]: df.head(5)
```

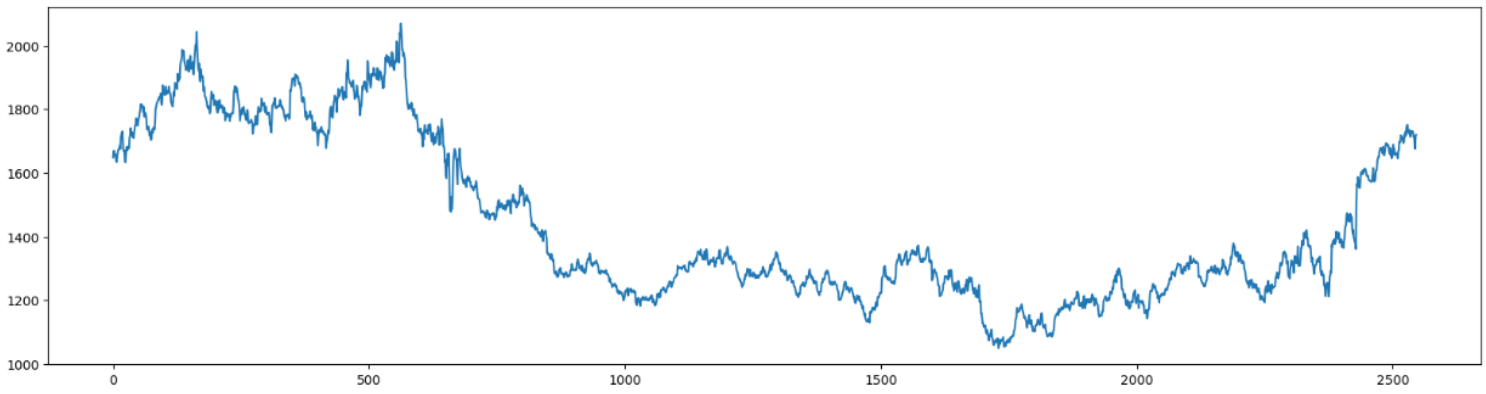
Out[3]:

	Date	Close/Last	Volume	Open	High	Low
0	2022-10-28	1648.3	186519.0	1667.2	1670.9	1640.7
1	2022-10-27	1668.8	180599.0	1668.8	1674.8	1658.5
2	2022-10-26	1669.2	183453.0	1657.7	1679.4	1653.8
3	2022-10-25	1658.0	178706.0	1654.5	1666.8	1641.2
4	2022-10-24	1654.1	167448.0	1662.9	1675.5	1648.0

```
In [4]: df.shape
Out[4]: (2547, 6)

In [5]: v_df = df[['Close/Last']]
        plt.figure(figsize=(20,5))
        plt.plot(v_df)
Out[5]: [<matplotlib.lines.Line2D at 0x2977a2744c0>]
```

Đọc dữ liệu vàng hàng ngày trong 10 năm qua và lưu trữ nó trong **df**. Xóa các cột không liên quan.



Bước 3: Đánh giá thống kê thông tin

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2547 entries, 0 to 2546
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        2547 non-null   datetime64[ns]
 1   Close/Last  2547 non-null   float64
 2   Volume      2508 non-null   float64
 3   Open        2547 non-null   float64
 4   High        2547 non-null   float64
 5   Low         2547 non-null   float64
dtypes: datetime64[ns](1), float64(5)
memory usage: 119.5 KB
```

Các thuộc tính có kiểu dữ liệu phù hợp để phân tích, cột Date đã có kiểu Datetime

```
In [7]: df.describe()
```

Out[7]:

	Close/Last	Volume	Open	High	Low
count	2547.000000	2508.000000	2547.000000	2547.000000	2547.000000
mean	1437.557008	182067.668660	1437.743031	1447.083235	1427.891991
std	255.898467	97589.342619	256.239938	257.924158	253.641116
min	1049.600000	1.000000	1051.500000	1062.700000	1045.400000
25%	1243.450000	120901.000000	1243.000000	1251.000000	1235.250000
50%	1318.500000	168425.500000	1319.000000	1326.300000	1310.900000
75%	1698.100000	231754.000000	1701.450000	1715.300000	1684.000000
max	2069.400000	787217.000000	2076.400000	2082.100000	2049.000000

Kích thước của dữ liệu

```
In [8]: # Verified shape
        df.shape
```

```
Out[8]: (2547, 6)
```

```
In [9]: # Verified the occurent of  missing in dataframe
        df.isna().any().any()
```

```
Out[9]: True
```

Dữ liệu có missing value nên phải xử lý lại dữ liệu

```
In [10]: # bỏ các hàng có có missing value
         df.dropna(inplace=True)
         df.reset_index(drop=True, inplace=True)
         # xem lại kích thước của dữ liệu
         df.shape
Out[10]: (2508, 6)
```

```
In [11]: df.describe()
```

Out[11]:

	Close/Last	Volume	Open	High	Low
count	2508.000000	2508.000000	2508.000000	2508.000000	2508.000000
mean	1438.936164	182067.668660	1439.127552	1448.610526	1429.120853
std	256.487316	97589.342619	256.831389	258.485303	254.250367
min	1049.600000	1.000000	1051.500000	1062.700000	1045.400000
25%	1244.075000	120901.000000	1243.900000	1251.500000	1235.800000
50%	1318.950000	168425.500000	1319.700000	1327.300000	1311.150000
75%	1705.200000	231754.000000	1707.125000	1718.100000	1687.525000
max	2069.400000	787217.000000	2076.400000	2082.100000	2049.000000

Sau khi xử lý missing value chúng ta loại bỏ đi 39 hàng trong bộ dữ liệu.

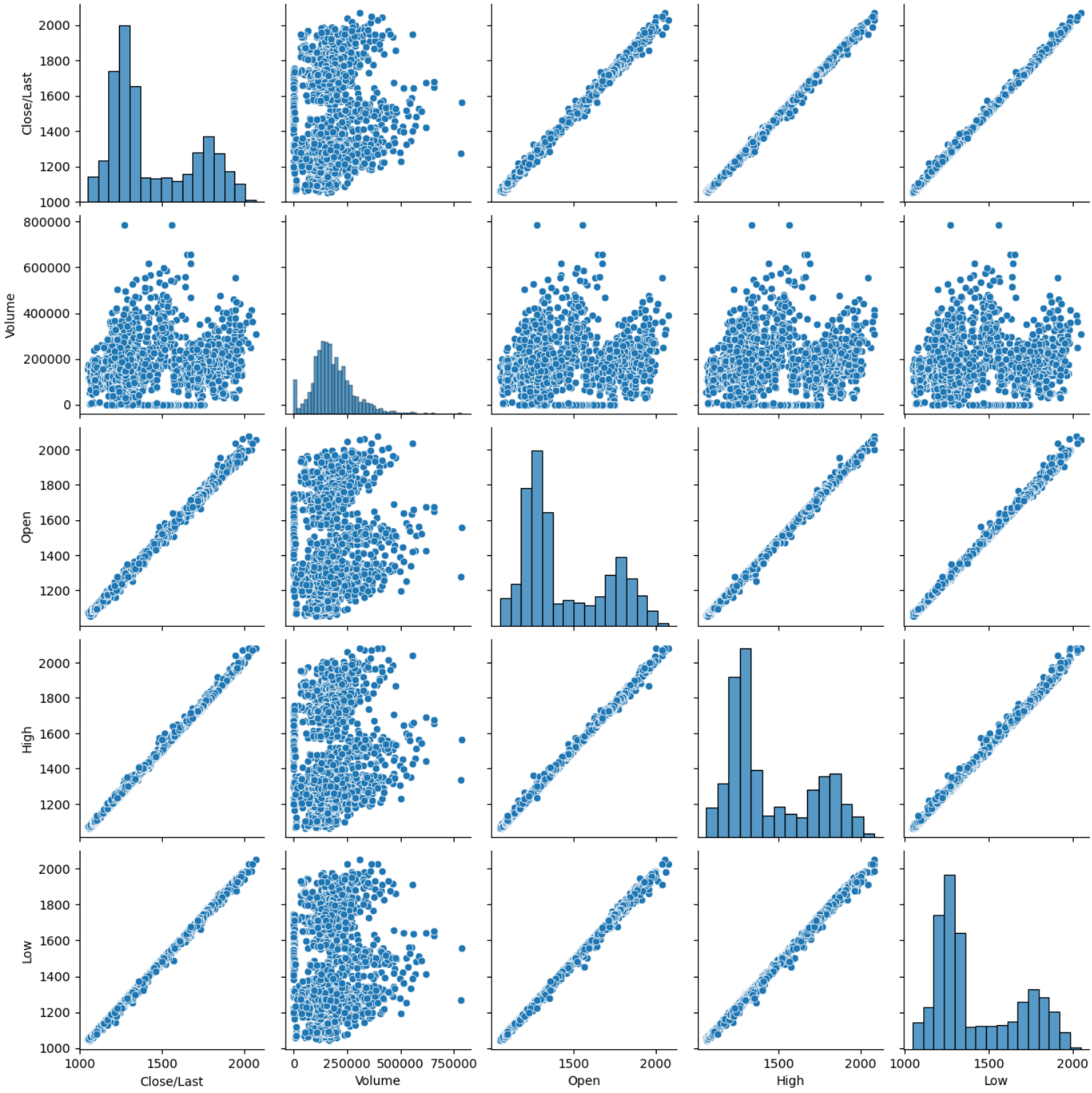
Do 39 dòng nhỏ so với số lượng hơn 2000 dòng trong bộ dữ liệu nên sau khi loại bỏ 39 dòng, bộ dữ liệu vẫn còn đủ tốt để phân tích và train model.

Bước 4: Trực quan hóa dữ liệu

Sử dụng hàm pairplot, heatmap để tìm kiếm các biến có tương quan mạnh với giá vàng

```
In [12]: sns.pairplot(df)
```

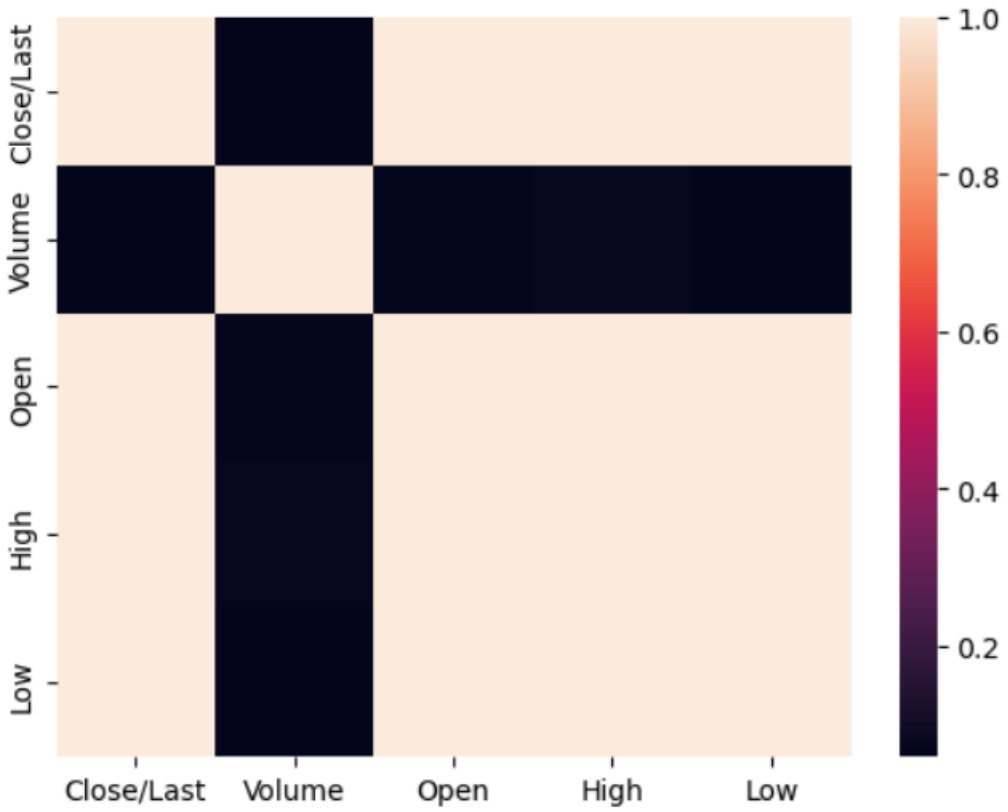
```
Out[12]: <seaborn.axisgrid.PairGrid at 0x2977ae38730>
```



Nhìn vào biểu đồ trên ta thấy các biến Close/Last, Open, High, Low có tương quan với nhau

Ngoài ra có thể sử dụng heatmap để tìm tương quan giữa các biến

```
In [13]: sns.heatmap(df.corr())
Out[13]: <AxesSubplot:>
```



Qua biểu đồ trên cho thấy các biến **Open, High, Low có tương quan mạnh với Close/Last.**

Bước 5: Features engineering and selection

Sử dụng tính năng tương quan để train model

```
In [14]: #Use strong correlation feature to train model
         features = ['Open','High','Low']
         label = ['Close/Last']
```

```
In [15]: X_df = df[features]
         y_df = df[label]
```

Sử dụng MinMaxScaler của sklearn để chuẩn hóa lại dữ liệu trong khoảng từ 0 đến 1

```
In [17]: scaler = MinMaxScaler()
         X_df_scaled = scaler.fit_transform(X_df)
         y_df_scaled = scaler.fit_transform(y_df)
```

Phía dưới là dữ liệu sau khi biến đổi:

```
In [18]: X_df_scaled
```

```
Out[18]: array([[0.60074154, 0.59662547, 0.59316461],
                [0.60230266, 0.60045125, 0.61090076],
                [0.59147234, 0.6049637 , 0.60621762],
                ...,
                [0.64796566, 0.64204434, 0.62714229],
                [0.65264904, 0.65214832, 0.66729773],
                [0.64279442, 0.65126545, 0.66201674]])
```

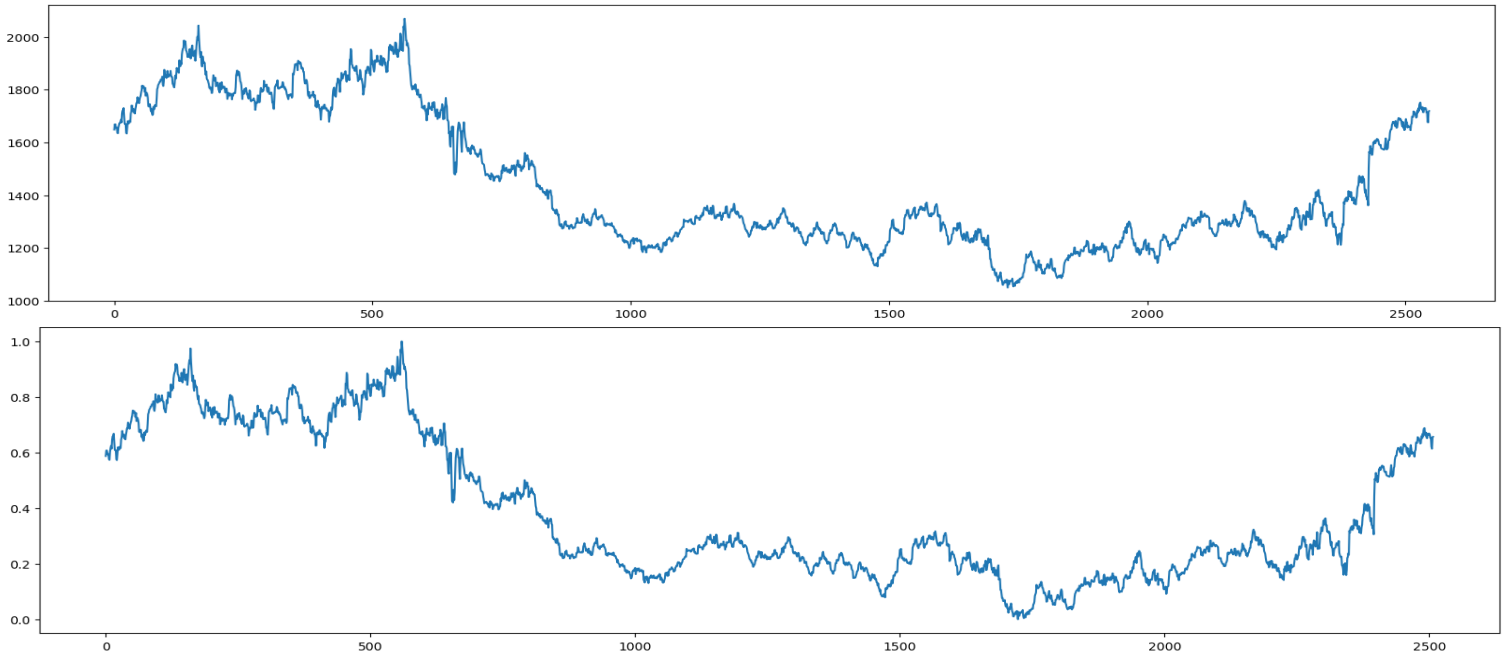
```
In [19]: y_df_scaled
```

```
Out[19]: array([[0.5870759 ],
                [0.60717788],
                [0.60757011],
                ...,
                [0.61345362],
                [0.65297117],
                [0.65650127]])
```

So sánh dữ liệu trước và sau khi biến đổi

```
In [20]: plt.figure(figsize=(20,5))
         plt.plot(v_df)
         plt.figure(figsize=(20,5))
         plt.plot(y_df_scaled)
```

```
Out[20]: [<matplotlib.lines.Line2D at 0x2977c0668b0>]
```



Có thể thấy dữ liệu trước và sau khi biến đổi giống nhau nên chúng ta sử dụng dữ liệu sau khi biến đổi để trainning model tốt hơn.

Bước 6: Model creation and training

Tiếp theo tách dữ liệu để train model theo kích thước 8:2

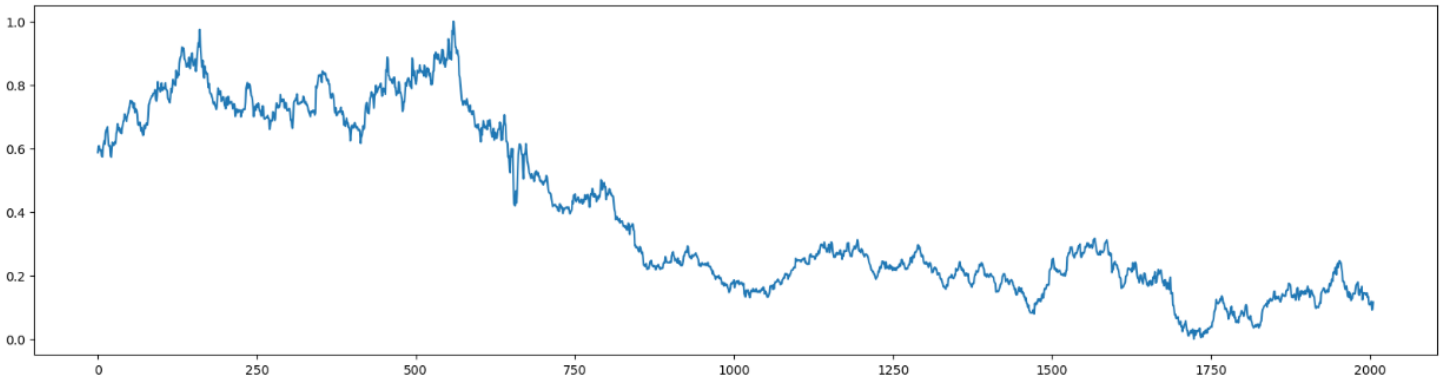
In [21]:
```
# Split training and tesing set with portion of 80% : 20% respectively.
X_train, X_test, y_train, y_test = train_test_split(X_df_scaled,y_df_scaled,shuffle = False, test_size = 0.2)
```

In [22]:
```
# xem dữ liệu sau khi tách
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2006, 3)
(502, 3)
(2006, 1)
(502, 1)
```

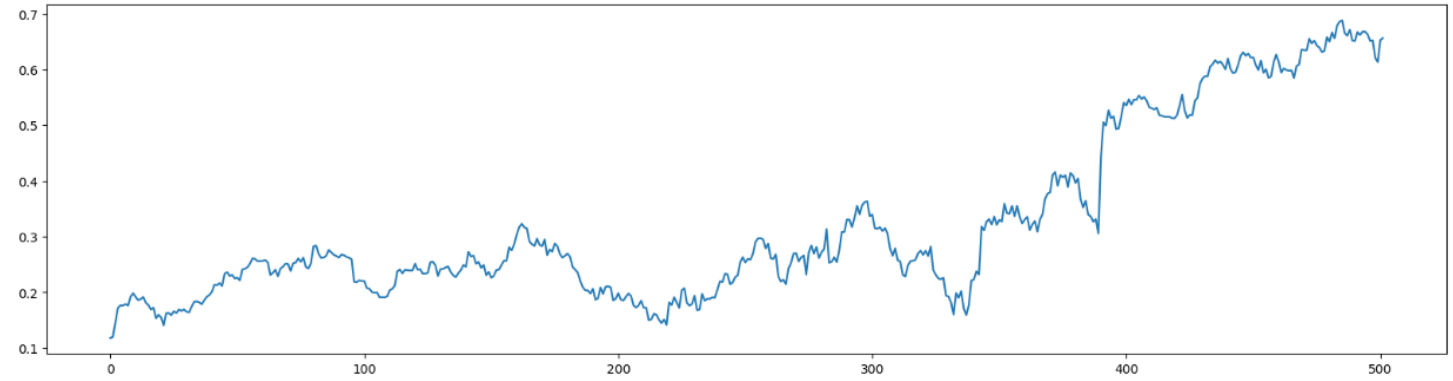
In [23]:
```
plt.figure(figsize=(20,5))
plt.plot(y_train)
```

Out[23]: [<matplotlib.lines.Line2D at 0x2977c02e2e0>]



In [23]:
```
plt.figure(figsize=(20,5))
plt.plot(y_train)
```

Out[23]: [<matplotlib.lines.Line2D at 0x2977c02e2e0>]



ĐÁNH GIÁ MÔ HÌNH

Hàm validate_result để đánh giá mô hình thông qua R2 , trung bình bình phương sai số (RMSE)

In [1]:
```
import matplotlib.dates as mdates
# Hàm để đánh giá mô hình
def validate_result(model, model_name):
    predicted = model.predict(X_test)
    RSME_score = np.sqrt(mean_squared_error(y_test, predicted))
    print('RMSE: ', RSME_score)

    R2_score = r2_score(y_test, predicted)
    print('R2 score: ', R2_score)

    plt.figure(figsize=(20,5))
    plt.plot(predicted,'r', label='Predict')
    plt.plot(y_test,'b', label='Actual')
    plt.ylabel('Price')
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
    plt.title(model_name + ' Predict vs Actual')
    plt.legend(loc='upper right')
    plt.show()
```

Mô hình ANN:

Xây dựng mô hình ANN

In [187]:
```
model = Sequential()
model.add(Dense(40, input_dim = 3, activation = 'relu'))
model.add(Dense(40, activation = 'relu'))
model.add(Dense(40, activation = 'relu'))
model.add(Dense(1, activation = 'linear'))
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 40)                160

 dense_17 (Dense)            (None, 40)                1640

 dense_18 (Dense)            (None, 40)                1640

 dense_19 (Dense)            (None, 1)                 41

=================================================================
Total params: 3,481
Trainable params: 3,481
Non-trainable params: 0
```

In [188]:
```
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
model.fit(X_train,y_train,epochs=100,batch_size = 50,verbose = 0, validation_split = 0.25)
```

Out[188]:
```
<keras.callbacks.History at 0x297056c25e0>
```

In [195]:
```
y_pred = model.predict(X_test)
pd.DataFrame({'y_pre:': y_pred.ravel(), 'y:': y_test.ravel()})
```

```
16/16 [==============================] - 0s 3ms/step
```

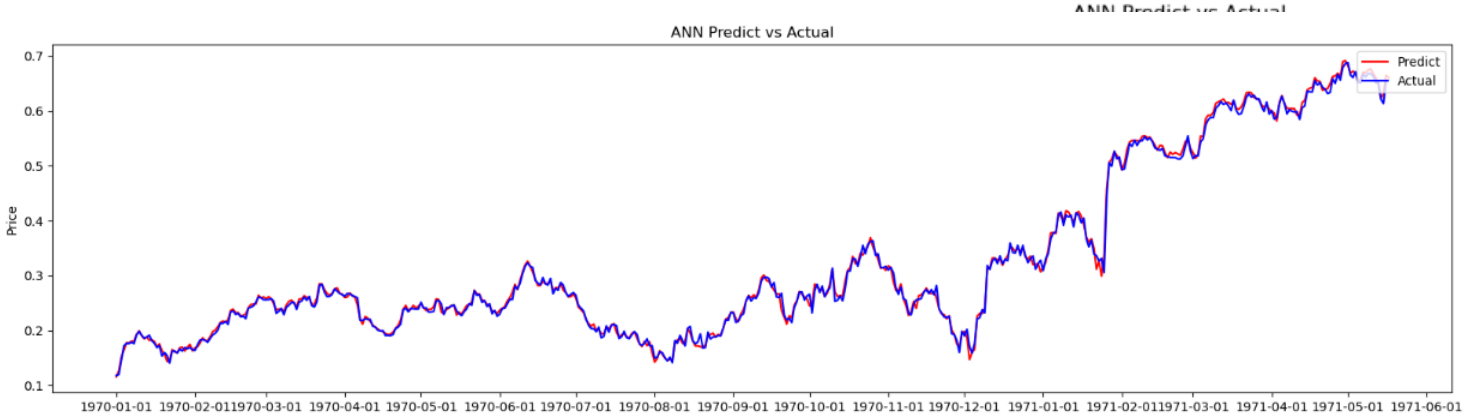
Out[195]:

	y_pre:	y:
0	0.115628	0.117866
1	0.126093	0.119631
2	0.151414	0.146107
3	0.163554	0.171896
4	0.177446	0.176309
...	...	...
497	0.647072	0.652481
498	0.623443	0.621298
499	0.630892	0.613454
500	0.664748	0.652971
501	0.661742	0.656501

502 rows × 2 columns

In [190]:
```
validate_result(model, 'ANN')
```

```
16/16 [==============================] - 0s 3ms/step
RMSE:  0.007009869468134474
R2 score:  0.9979481978378579
```



DecisionTreeRegressor:

Sử dụng mô hình DecisionTree Regressor để dự đoán với tham số mặc định

In [196]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
model = DecisionTreeRegressor(random_state = 1)
```

In [197]:
```
DT_model = model.fit(X_train, y_train)
```

In [198]:
```
y_pred = DT_model.predict(X_test)
pd.DataFrame({'y_pre:': y_pred, 'y:': y_test.ravel()})
```

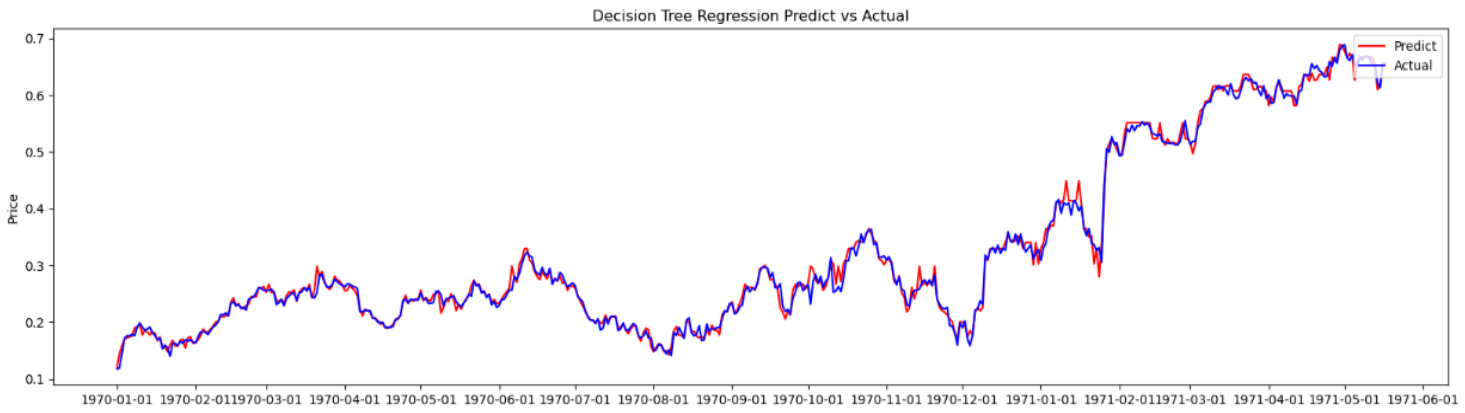
Out[198]:

	y_pre:	y:
0	0.121200	0.117866
1	0.145421	0.119631
2	0.158659	0.146107
3	0.170524	0.171896
4	0.172387	0.176309
...	...	...
497	0.647774	0.652481
498	0.609433	0.621298
499	0.616199	0.613454
500	0.657188	0.652971
501	0.653854	0.656501

502 rows × 2 columns

In [176]:
```
validate_result(DT_model, 'Decision Tree Regression')
```

```
RMSE:  0.011163564329622044
R2 score:  0.9947961906172589
```



Từ kết quả trên, chúng ta có thể thấy rằng hiệu suất của mô hình thật sự tốt trên bộ xác thực, mô hình phần nào dự đoán giá tốt.

RandomForest:

Random Forest là một thuật toán học có giám sát. Nó tạo ra một khu rừng và biến nó thành ngẫu nhiên theo cách nào đó. Các rừng mà nó xây dựng, là một tập hợp các Cây quyết định, phần lớn thời gian được huấn luyện bằng phương pháp "đóng gói". Các ý tưởng chung của phương pháp đóng gói là sự kết hợp của các mô hình học tập sẽ làm tăng kết quả tổng thể. Các khu rừng quyết định ngẫu nhiên sửa lỗi cho các cây quyết định có thói quen khớp quá mức với tập huấn luyện của chúng. Em đã sử dụng hai tham số n_estimators=50 (giá trị mặc định =10), số lượng cây trong rừng. và random_state=0, random_state là hạt giống được sử dụng bởi trình tạo số ngẫu nhiên.

Trong phần này sẽ chỉnh 3 thông số của RamdomForest là n_estimators = 100, max_features ='auto', max_depth = 10

In [177]:
```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100,max_features = 'auto',max_depth = 10 ,random_state=0)
RF_model= model.fit(X_train, y_train.ravel())
```

In [199]:
```
y_pred = RFmodel.predict(X_test)
pd.DataFrame({'y_pre:': y_pred, 'y:': y_test.ravel()})
```

Out[199]:

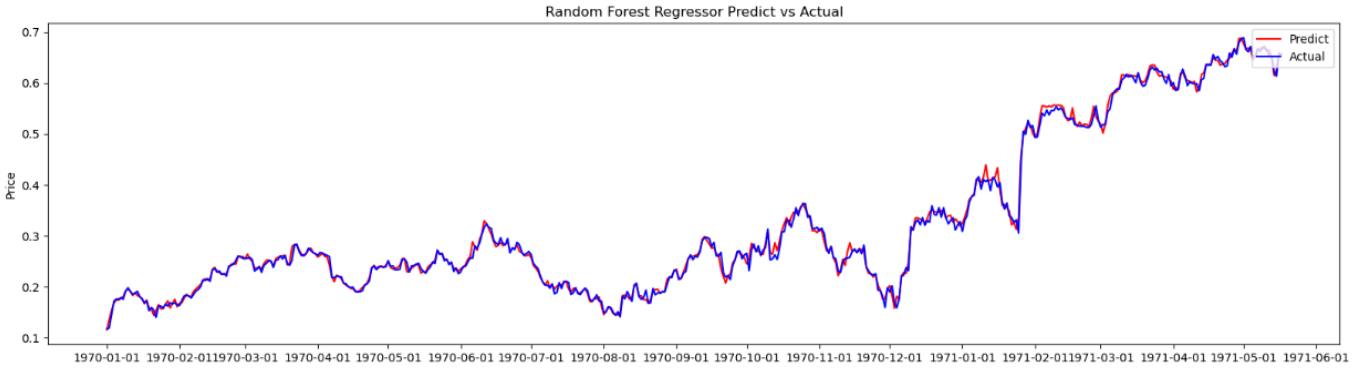
	y_pre:	y:
0	0.115861	0.117866
1	0.136633	0.119631
2	0.154081	0.146107
3	0.167850	0.171896
4	0.174844	0.176309
...	...	...
497	0.647076	0.652481
498	0.614408	0.621298
499	0.625055	0.613454
500	0.658717	0.652971
501	0.655658	0.656501

502 rows × 2 columns

Sau khi áp dụng random forest với n_estimators = 100, đạt được kết quả như sau:

```
In [180]: validate_result(RF_model, 'Random Forest Regressor')
```

```
RMSE:  0.008297463834112308
R2 score:  0.9971252084645066
```



SVR:

Mô hình được tạo ra bởi Support Vector Regression chỉ phụ thuộc vào một tập con của dữ liệu huấn luyện, bởi vì hàm chi phí để xây dựng mô hình bỏ qua mọi dữ liệu huấn luyện gần với dự đoán của mô hình.

Sử dụng SVR với (kernel=’linear’)

```
In [185]: from sklearn.svm import SVR
          # Save all soultion models
          solution_models = {}
          # SVR with  linear Kernel
          svr_lin = SVR(kernel='linear')
          linear_svr_clf_feat = svr_lin.fit(X_train,y_train.ravel())
```

In [200]:
```
y_pred = linear_svr_clf_feat.predict(X_test)
pd.DataFrame({'y_pre:': y_pred, 'y:': y_test.ravel()})
```

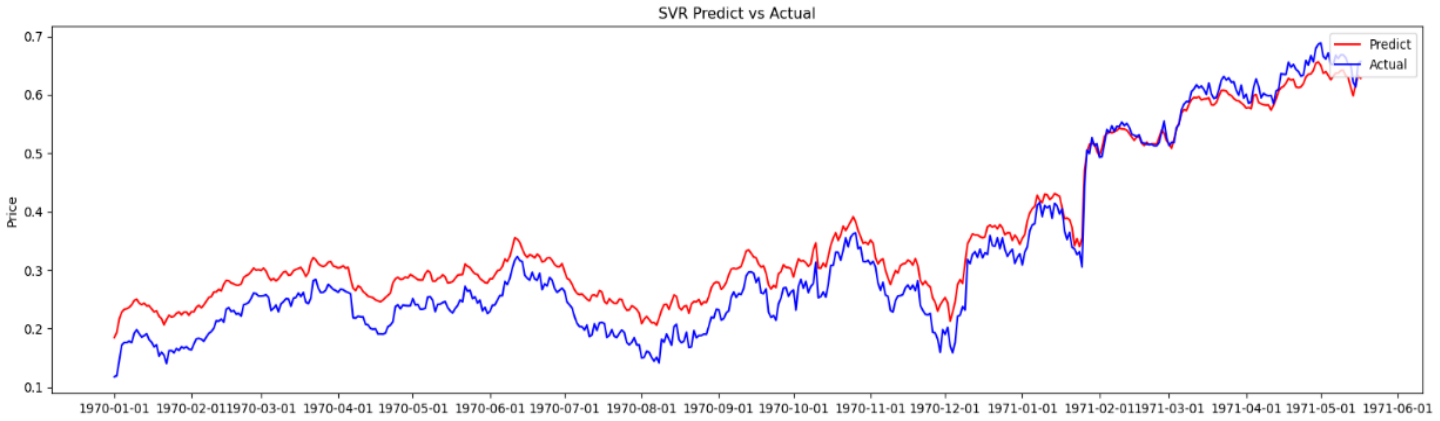
Out[200]:

	y_pre:	y:
0	0.184811	0.117866
1	0.193487	0.119631
2	0.216592	0.146107
3	0.228227	0.171896
4	0.232949	0.176309
...	...	...
497	0.612882	0.652481
498	0.598026	0.621298
499	0.617337	0.613454
500	0.632466	0.652971
501	0.628138	0.656501

502 rows × 2 columns

In [186]:
```
validate_result(linear_svr_clf_feat, 'SVR')
```

```
RMSE:  0.04282827588862639
R2 score:  0.9234091649720787
```



KẾT LUẬN

Trong dự án trên, ta xây dựng mô hình dự đoán giá vàng bằng các mô hình hồi quy tuyến tính, thông squa 3 yếu tố giá mở vàng, giá cao nhất, giá thấp nhất có tương quan mạnh đến giá vàng. Theo tính tính, mô hình đưa giá dự đoán trên tất cả các biến có độ chính xác và trung bình phương sai số.

Models	**RMSE**	**R2 score**
ANN	0.007	0.9979
DecisionTreeRegressor	0.011	0.9947
Random Forest	0.008	0.9971
SVR	0.042	0.923

Mô hình ANN với tất cả các tính năng cho thấy kết quả tốt nhất (với RMSE 0,007 và điểm R2 là 0,9979) khi so sánh với các mô hình giải pháp khác.

Vẫn còn nhiều chỉ số kỹ thuật và biến tính năng mà chúng em chưa đưa vào dự án của mình, có thể có một số chỉ số khác mà chúng em chưa khám phá sẽ hoạt động tốt hơn.

Hầu như không thể có được một mô hình có thể dự đoán 100% giá mà không có bất kỳ lỗi nào, có quá nhiều yếu tố mà giá vàng phụ thuộc vào. Em đã cố nắm bắt càng nhiều yếu tố càng tốt nhưng vẫn có những yếu tố khác như CPI, yếu tố chính trị, thảm họa, sự cố tài chính có thể ảnh hưởng đến thị trường. Nhưng mô hình giải pháp này đã thực hiện rất tốt trong dự án này, điều này sẽ giúp nhà giao dịch thực hiện quyết định tốt hơn. Xu hướng chung của giá dự đoán phù hợp với dữ liệu thực tế, vì vậy nhà giao dịch có thể có tham khảo và tự đưa ra quyết định giao dịch.

Bài dự đoán này, nhóm chúng em/mình dự đoán giá vàng dựa trên bộ dữ liệu có sẵn (được cập nhập liên tục). Chúng em đưua ra kết quả dự đoán và so sánh với giá tiếp theo nên vì thế mô hình này trong thực tế chưa khả thi về dự đoán chính xác cho giá vàng. Chúng em/mình sẽ cố gắng tìm hiểu thêm và cải thện để đưa ra mô hình tốt nhất. Hiện tại chúng em/mình dự đoán dựa trên những gì đã được học và tìm hiểu nên kết quả đưa ra chỉ là so sánh với giá tiếp theo trong dữ liệu và chỉ mang tính chất học thuật. Nếu có gì thiếu sót mong thầy và các bạn góp ý thêm để đạt hiệu quả tốt nhất. Cảm ơn thầy và các bạn đã bỏ thời gian ra để đọc và tìm hiểu bài báo cáo của nhóm.

TÀI LIỆU THAM KHẢO

https://www.nasdaq.com/market-activity/commodities/gc%3Acmx

https://scikit-learn.org/stable/ (Tham khảo các thuật toán có trong đồ án)

https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz (Tham khảo mô hình random forests)

https://viblo.asia/p/decision-tree-Do754bbBZM6 (Tham khảo mô hình decision tree)