Machine Learning Project

Timeseries Forecasting

GOLD (XAUUSD)

Estrada, Leoranele Grace Q.
MAN 206 – Predictive Modelling and Machine Learning

Presentation Outline

Here's what we'll cover:

- Business Understanding
- Data Understanding
- Data Preparation
- Modelling and Evaluation – ETS Model
- Additional Preparation for ARIMA Models
- Modelling and Evaluation – ARIMAX and SARIMAX
- Recommendation and Business Context

Business Understanding

Statement of the Business Problem

The gold market offers high liquidity and excellent opportunities to profit. But like all financial assets, investing in and trading gold comes with the risks of losing capital.

Objectives

To create a model that would forecast the closing price of GOLD (XAUUSD) to maximize trading gains and minimize risks of losing capital.

Data Understanding

Description of the variables used in the model

ETS Model

- XAUUSD Close [target]

ARIMAX and SARIMAX

- XAUUSD Close [target]
- XAUUSD Open
- XAUUSD Low
- XAUUSD High
- S&P Close
- SLV Close
- OIL Close
- EURUSD Close

Based on my research, the above-mentioned variables/values can be used as predictors to be able to forecast GOLD (XAUUSD) price

Timeframe for Data Gathering

The historical dataset that I gathered ranges from January 5, 2000, to October 12, 2022

Data Preparation

FOR ETS MODEL

Discussion of data sources and processing

Discussion of data issues and remedies implemented

Discussion of new features created

[will be discussed further on the Additional Data Preparation section]

Data Preparation

1. Gathered the historical prices of my variables https://ph.investing.com/currencies/xau-usd-historical-data and saved it on a csv file

2. Loaded the data and parsed the **Date column** as a date variable

```
# Load data
xau = pd.read_csv(r"C:\Users\GRACE ESTRADA\OneDrive\Desktop\XAUUSD.csv", parse_dates = ['Date'])
```

3. Set the **Date column** as index to speed up data manipulations and modeling

```
# Set index to date
xau = xau.sort_values('Date', ascending = True)
xau = xau.set_index('Date')
```

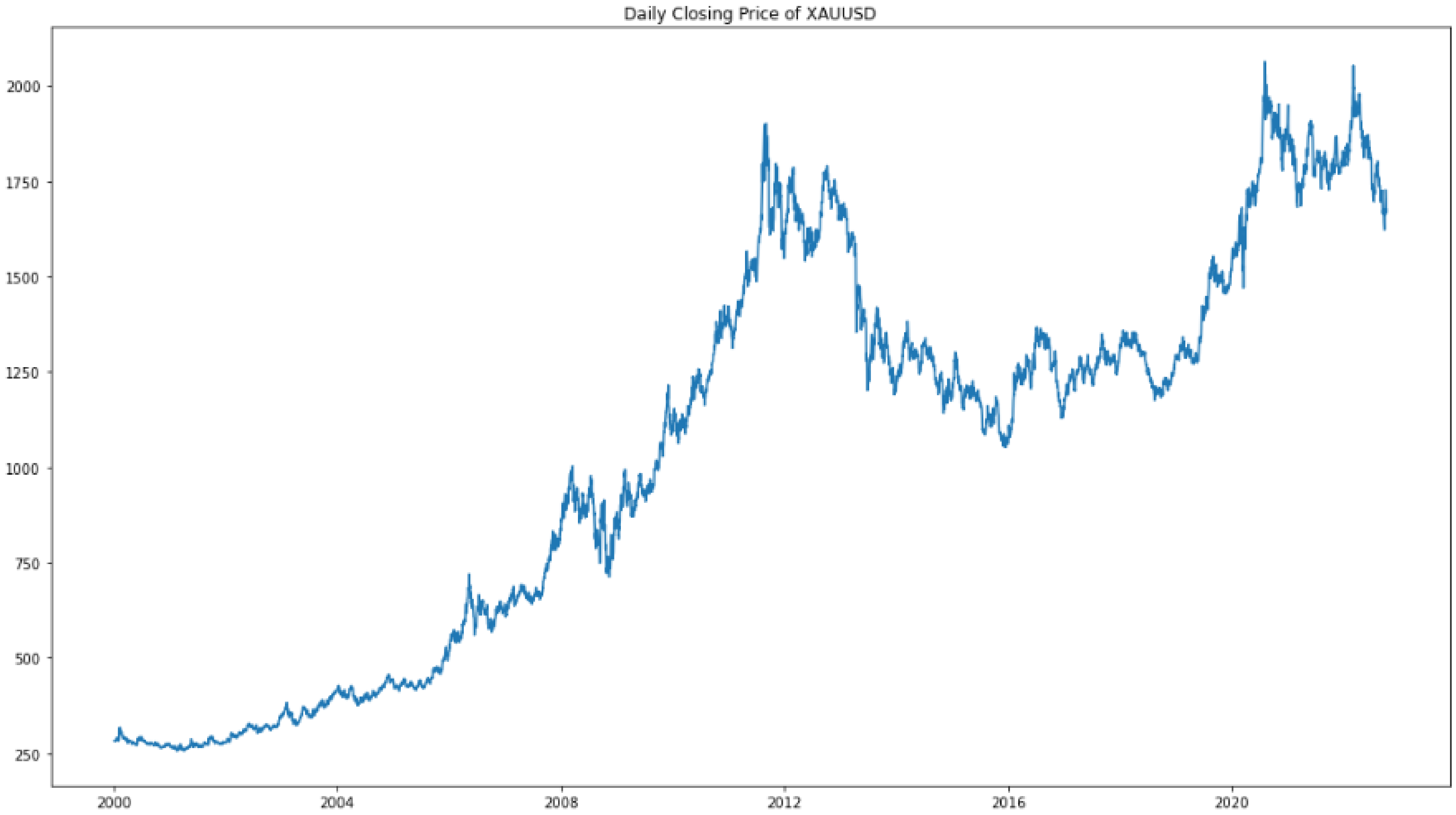
4. Sed the date frequency to business days to ensure that there are no missing dates

```
# Ensure that the frequency is set to business days
xau = xau.asfreq('B')
```

5. As there are some missing values for the days with suspended trading days, I decided to fill missing values using forward fill method

```
# Fill in missing values with forward fill
xau = xau[1:].ffill()

# Ensure that there are no more missing values
assert xau.isna().sum().sum() == 0
```

Data Visualization

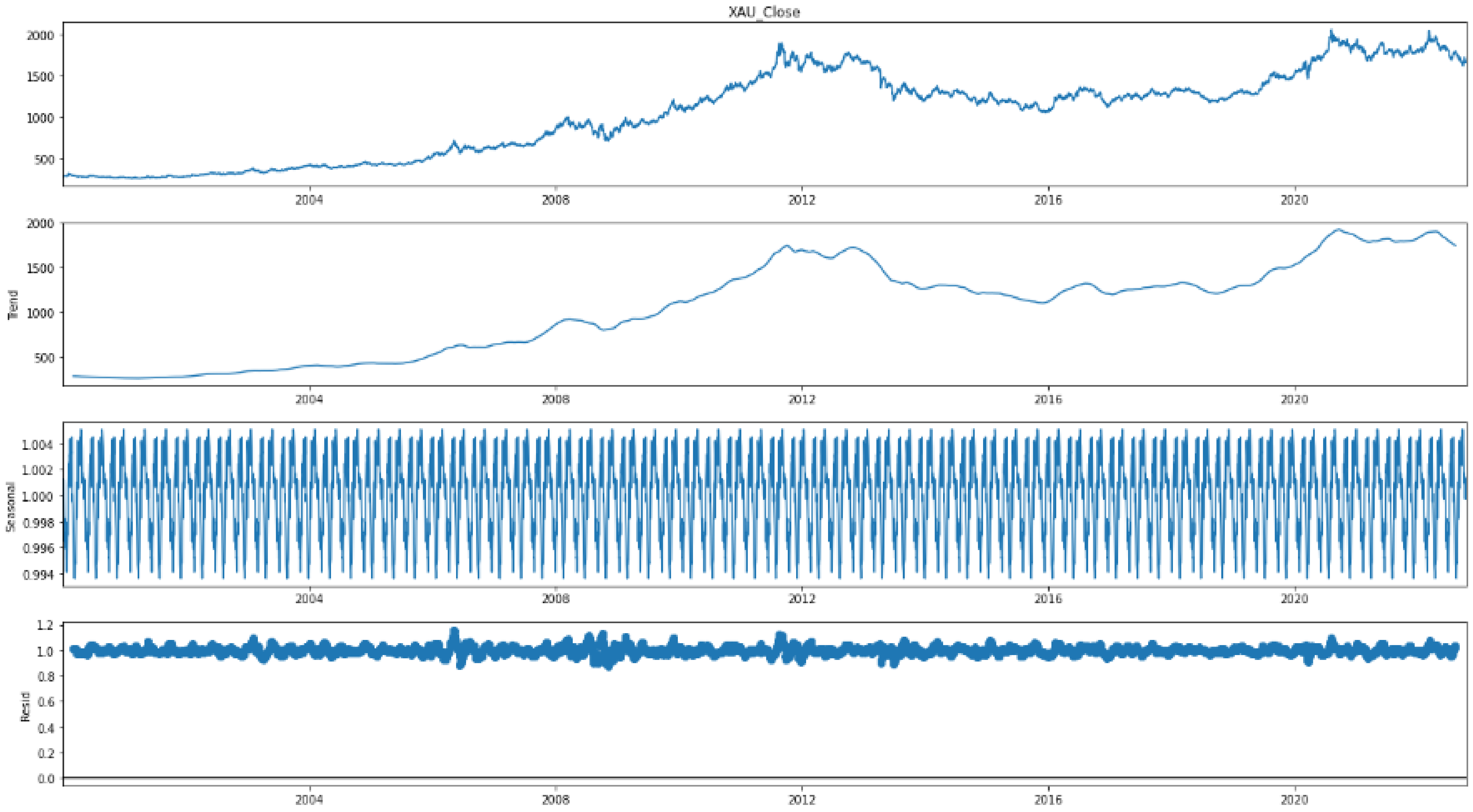


The closing prices has an increasing trend and has some seasonality component as observed from the data

Decomposition Plot

Decomposition plot shows seasonality using the 90-day cycle length.
The trend is increasing, and the residual seems to have equal variances can be considered as stable

```
# Decompose
seasonal_decompose(xau['XAU_Close'], model='multiplicable', period=90).plot()
plt.show()
```



Modelling and Evaluation

FOR ETS MODEL

Presentation of various models that were considered

- <u>ETS</u>
- ARIMA
- SARIMAX

Presentation of final model selected

Train-Test Split and Optimization

1

Split the data into training and test sets to have an out-of-sample forecast and to be able to evaluate the model accurately

```
# Define Endog Variables

endog = xau['XAU_Close']
endog_train = xau['XAU_Close']['2000-01-01':'2021-12-31']
endog_test = xau[['XAU_Close']]['2022-01-01':]
```

Trainset contains data ranging from January 1, 2000 to December 31, 2021
Test set contains data ranging from January 1, 2022 to October 12, 2022

2

Created a function to fit a range of possible combination of parameters into the model and returns the best model based on MSE and AIC

```
def optimize_ETS(endog, param_list):
    """
        Return dataframe with parameters and corresponding MSE and AIC

        order_list - list with error, trend, and seasonality parameters
        endog - the observed variable
    """

    results = []

    for order in tqdm_notebook(param_list):
        model = ETSModel(endog,
                         error = order[0],
                         trend = order[1],
                         seasonal = order[2],
                         damped_trend = True,
                         seasonal_periods = order[3]).fit()

        aic = model.aic
        mse = model.mse
        results.append([order, aic, mse])

    result_df = pd.DataFrame(results)
    result_df.columns = ["(e, t, s)", "AIC", "MSE"]
    #Sort in ascending order, lower MSE is better
    result_df = result_df.sort_values(by='MSE', ascending=True).reset_index(drop=True)

    return result_df
```

Model Tuning

1

Created a list of 40 possible combination for the error, trend, seasonality, and period parameter

```
e = ['add', 'mul']
t = ['add', 'mul']
s = ['add', 'mul']
p = [5, 12, 20, 30, 90]

param_list = list(product(e,t,s,p))
```

2

After running the custom model tuning function, it returned the following results:

```
result_df = optimize_ETS(endog_train, param_list)
result_df
```

	(e, t, s)	AIC	MSE
0	(mul, mul, mul, 90)	41736.552753	0.000112
1	(mul, add, mul, 90)	41737.183074	0.000112
2	(mul, add, add, 90)	41743.642183	0.000112
3	(mul, mul, add, 90)	41752.463402	0.000112
4	(mul, mul, mul, 20)	41645.463423	0.000113
5	(mul, add, mul, 20)	41646.109726	0.000113
6	(mul, mul, add, 20)	41647.604017	0.000113
7	(mul, add, add, 20)	41648.062295	0.000113
8	(mul, add, add, 30)	41676.007259	0.000113
9	(mul, mul, add, 30)	41676.015447	0.000113
10	(mul, mul, mul, 30)	41678.928041	0.000113

NOTE: Snippet of the Top 10 results with the lowest MSE

The best model parameters for the dataset are as follows:

- ERROR- Multiplicative
- TREND – Multiplicative
- SEASONALITY – Multiplicative
- PERIOD – 90 business days

I can say that the results do make sense as the multiplicative method is usually preferred when seasonal variations are changing proportional to the level of the series which what is observed based on the data

Model Fitting

1

Fitted the model using the best parameters

```python
# Fit the model using the best parameters
model = ETSModel(
    endog_train,
    error="mul",
    trend="mul",
    seasonal="mul",
    damped_trend=True,
    seasonal_periods=90)

res = model.fit()
print(res.summary())
```

```
                                 ETS Results
==============================================================================
Dep. Variable:                XAU_Close   No. Observations:                 5739
Model:                        ETS(MMdM)   Log Likelihood              -20771.276
Date:                  Thu, 20 Oct 2022   AIC                          41736.553
Time:                          00:29:24   BIC                          42382.092
Sample:                      01-04-2000   HQIC                         41961.232
                           - 12-31-2021   Scale                            0.000
Covariance Type:                 approx
=======================================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------------
smoothing_level         0.9999      0.014     70.520      0.000       0.972       1.028
smoothing_trend      9.999e-05      0.007      0.014      0.988      -0.013       0.014
smoothing_seasonal   3.506e-05        nan        nan        nan         nan         nan
damping_trend           0.8000        nan        nan        nan         nan         nan
initial_level         287.3764        nan        nan        nan         nan         nan
initial_trend           0.9935        nan        nan        nan         nan         nan

===================================================================================
Ljung-Box (Q):                      231.53   Jarque-Bera (JB):              10730.99
Prob(Q):                              0.01   Prob(JB):                          0.00
Heteroskedasticity (H):               0.73   Skew:                             -0.06
Prob(H) (two-sided):                  0.00   Kurtosis:                          9.70
===================================================================================

Warnings:
[1] Covariance matrix calculated using numerical (complex-step) differentiation.
```

Forecasting (in-sample)

The predicted values by the model follows closely the actual values from the train set. This is expected since the data used for forecasting is the data used to fit the model.

```python
#In-sample ETS forecast

fig, ax = plt.subplots()

ax.plot(xau['XAU_Close'][:'2021'], 'steelblue', label = 'Train set', linewidth = 3)
ax.plot(xau['Predicted'][:'2021'], 'black', linestyle = 'dashed', label = 'In sample prediction', linewidth = 1)

ax.legend(loc = 'upper left')

plt.show()

```

In-sample forecasts has a very high **coefficient of determination (R^2) value of 99%** and a low **RMSE of $11.74**

```python
# In-sample ETS evaluation

r2 = r2(xau['XAU_Close'][:'2021'], xau['Predicted'][:'2021'])
mse = mse(xau['XAU_Close'][:'2021'], xau['Predicted'][:'2021'])
rmse = np.sqrt(mse)

print(f"R2:{r2}\nMSE: {mse}\nRMSE: {rmse}")
```

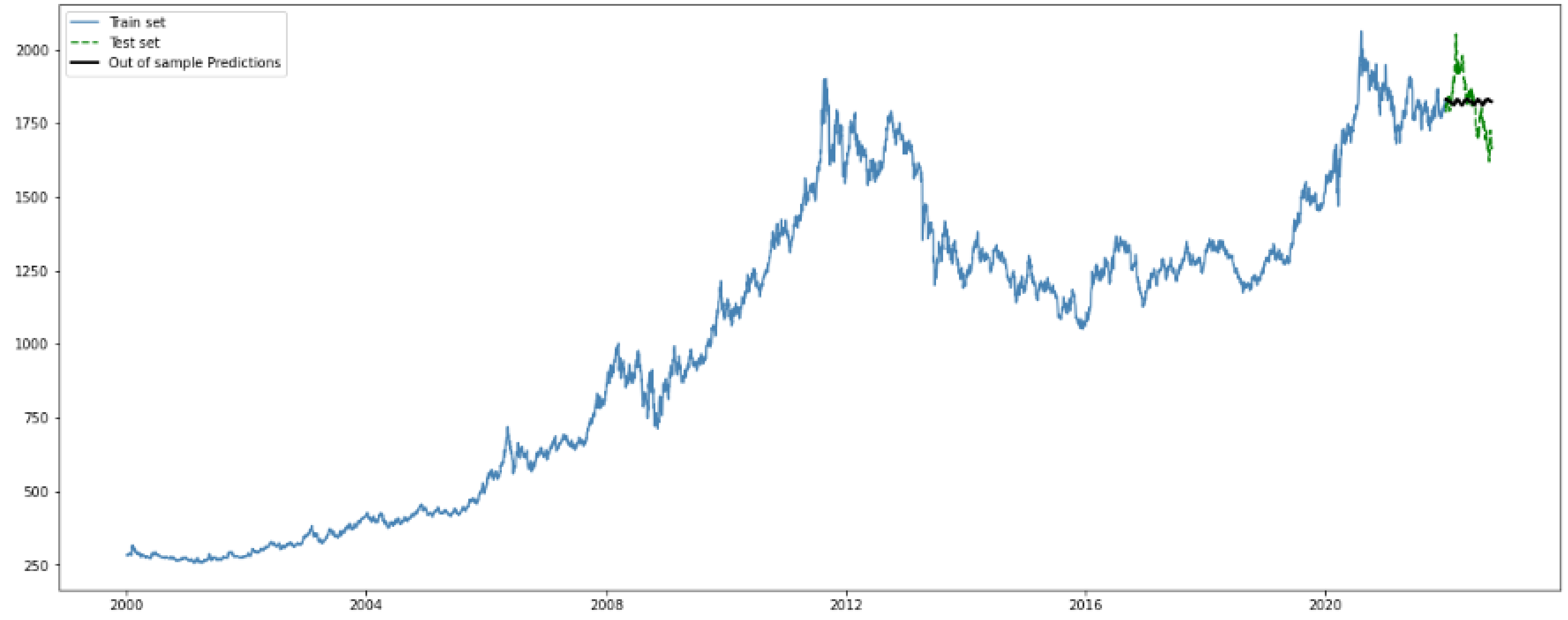
```
R2:0.9994623393436289
MSE: 138.05720427928765
RMSE: 11.749774648021452
```

Forecasting & Evaluation (out-of-sample)

The forecast using out-of-sample or unseen data did not match the actual values based on the test set. To put it simply, the ETS model did not perform well on forecasting future values

Out-of-sample forecasts obtained a **coefficient of determination (R^2) value of -4%** and **RMSE of $92.42**

```python
# Out-of-sample ETS forecast (zoomed-out)
plt.rcParams['figure.figsize'] = [20, 8]
fig, ax = plt.subplots()
ax.plot(endog_train, 'steelblue', label = 'Train set')
ax.plot(endog_test, 'green', linestyle = 'dashed', label = 'Test set')
ax.plot(forecast['Predicted'], 'black', linewidth = 2,label = 'Out of sample Predictions')
ax.legend(loc = 'upper left')
plt.show()
```




```python
# Out-of-sample forecast (zoomed-in)
plt.rcParams['figure.figsize'] = [20, 8]
fig, ax = plt.subplots()
ax.plot(endog_train['2020':], 'steelblue', label = 'Train set')
ax.plot(endog_test, 'green', label = 'Test set')
ax.plot(forecast['Predicted'], 'black', linestyle = 'dashed', linewidth = 2,label = 'Out of sample Predictions')
ax.legend(loc = 'upper left')
plt.show()
```

Additional Data Preparation

FOR ARIMAX and SARIMAX MODEL

Discussion of data sources and processing

Discussion of data issues and remedies implemented

Discussion of new features created

Data Transformation

1

Performed Augmented Dickey Fuller ADF Test on the dataset to determine if it meets the stationarity requirement for ARIMA models

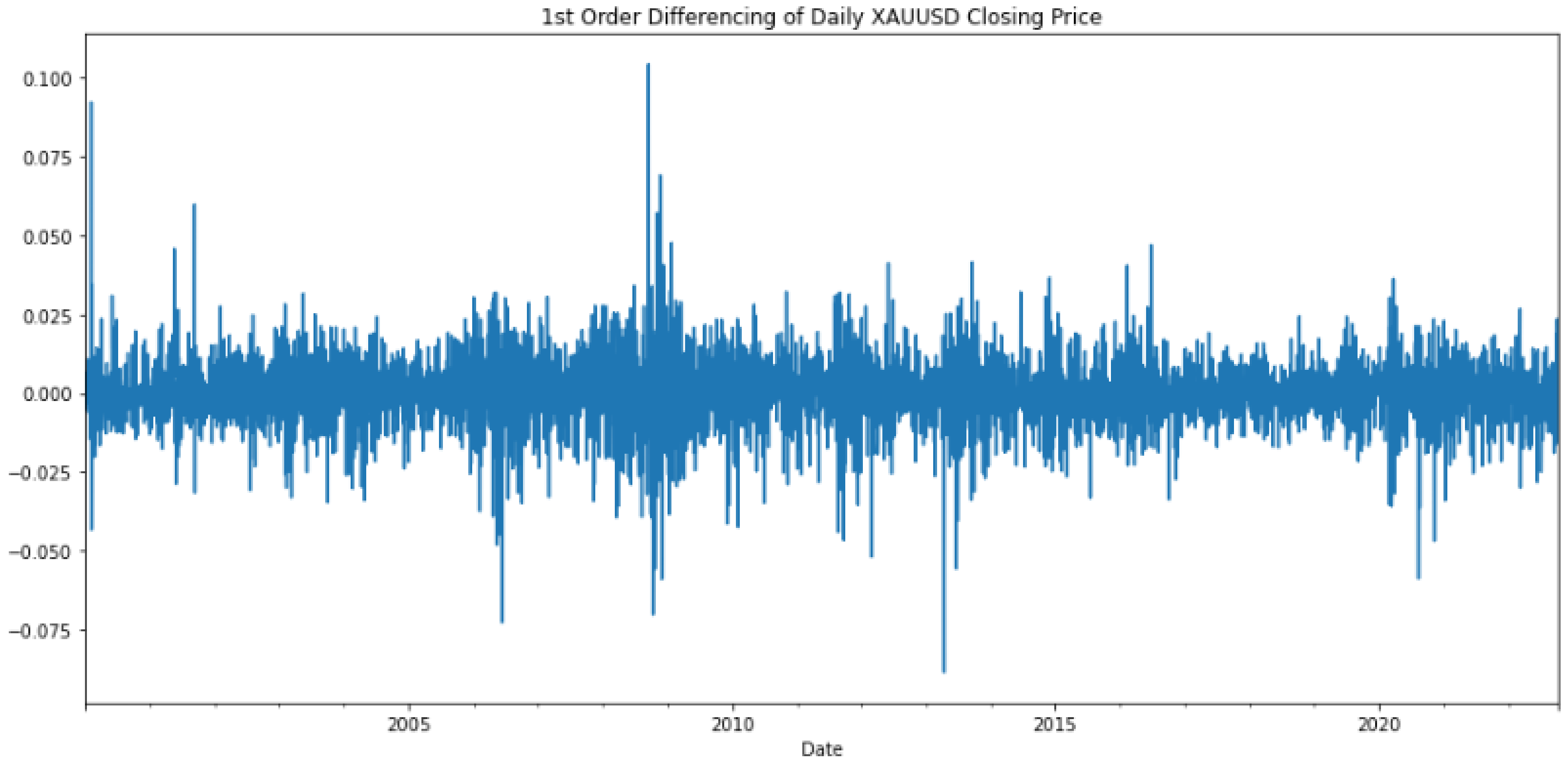
```
# AdFuller Test prior to Transformation

ad_fuller_result = adfuller(xau['XAU_Close'])

print(f'ADF Statistic: {ad_fuller_result[0]}')
print(f'p-value: {ad_fuller_result[1]}')
```

```
ADF Statistic: -1.0706089836469215
p-value: 0.7266492896560852
```

2

It can be inferred based on the p-value of .73 that the data is not stationary, therefore further transformation is required such as log differencing

```
# 1st order of differencing
XAU_diff = np.log(xau['XAU_Close']).diff()

XAU_diff.plot()
plt.title('1st Order Differencing of Daily XAUUSD Closing Price')
plt.show()
```



3

After performing the first order of differencing, the data is already stationary as evidenced by the p-value of almost equal to zero.

```
# AdFuller Test after transformation

ad_fuller_result_D1 = adfuller(XAU_diff[1:])

print(f'ADF Statistic: {ad_fuller_result_D1[0]}')
print(f'p-value: {ad_fuller_result_D1[1]}')
```

```
ADF Statistic: -77.81468158349843
p-value: 0.0
```

Based on the results of the statistical test and transformation, it seems fit to use first order difference for the differencing (d) parameter in the ARIMAX and SARIMAX model

Additional Features

1

Created lag 1 values for the XAU High, XAU Low, S&P500, SLV, OIL, and EURUSD variables to be fitted to the model.
These features will enable the model to predict the value of XAU Close for the next day based on the previous value of the mentioned exogenous variables.

```
xau_lag = xau[['XAU_High', 'XAU_Low', 'S&P500', 'SLV', 'OIL', 'EURUSD']].shift(1)
xau_lag.insert(0, 'XAU_Close', xau['XAU_Close'])
xau_lag.insert(1, 'XAU_Open', xau['XAU_Open'])
xau_lag = xau_lag[1:]
xau_lag.rename(columns = {'XAU_High': 'XAU_High_lag1',
                          'XAU_Low': 'XAU_Low_lag1',
                          'S&P500': 'S&P500_lag1',
                          'SLV' : 'SLV_lag1',
                          'OIL': 'OIL_lag1',
                          'EURUSD': 'EURUSD_lag'})

xau = xau[1:]

assert len(xau) == len(xau_lag)
```

Date	XAU_Close	XAU_Open	XAU_High	XAU_Low	S&P500	SLV	OIL	EURUSD
2000-01-05	281.00	281.50	281.00	281.00	1402.10	5.210	24.91	1.0316
2000-01-06	281.23	280.12	281.23	281.23	1403.50	5.167	24.78	1.0324
2000-01-07	281.75	281.15	281.75	281.75	1441.50	5.195	24.22	1.0292
2000-01-10	281.48	281.88	281.48	281.48	1457.60	5.190	24.67	1.0257
2000-01-11	283.38	281.48	283.38	283.38	1438.60	5.195	25.77	1.0335
...	...	...	...	...	...	...	...	...
2022-10-06	1710.85	1716.06	1726.09	1706.53	3744.52	20.660	88.45	0.9788
2022-10-07	1694.52	1711.10	1715.40	1690.35	3639.66	20.255	92.64	0.9741
2022-10-10	1667.96	1696.80	1700.33	1665.30	3612.39	19.615	91.13	0.9700
2022-10-11	1665.31	1668.40	1684.44	1660.45	3588.84	19.487	89.35	0.9703
2022-10-12	1674.60	1664.16	1678.24	1661.49	3577.03	19.032	87.27	0.9705

Date	XAU_Close	XAU_Open	XAU_High_lag1	XAU_Low_lag1	S&P500_lag1	SLV_lag1	OIL_lag1	EURUSD_lag
2000-01-05	281.00	281.50	282.45	282.45	1399.40	5.375	25.55	1.0312
2000-01-06	281.23	280.12	281.00	281.00	1402.10	5.210	24.91	1.0316
2000-01-07	281.75	281.15	281.23	281.23	1403.50	5.167	24.78	1.0324
2000-01-10	281.48	281.88	281.75	281.75	1441.50	5.195	24.22	1.0292
2000-01-11	283.38	281.48	281.48	281.48	1457.60	5.190	24.67	1.0257
...	...	...	...	...	...	...	...	...
2022-10-06	1710.85	1716.06	1728.10	1700.15	3783.28	20.544	87.76	0.9882
2022-10-07	1694.52	1711.10	1726.09	1706.53	3744.52	20.660	88.45	0.9788
2022-10-10	1667.96	1696.80	1715.40	1690.35	3639.66	20.255	92.64	0.9741
2022-10-11	1665.31	1668.40	1700.33	1665.30	3612.39	19.615	91.13	0.9700
2022-10-12	1674.60	1664.16	1684.44	1660.45	3588.84	19.487	89.35	0.9703

Modelling and Evaluation

FOR ARIMAX MODEL

Presentation of various models that were considered

- ETS
- ARIMA
- SARIMAX

Presentation of final model selected

Train-Test Split and Optimization

1

Split the data into training and test sets to have an out-of-sample forecast and to be able to evaluate the model accurately

```python
# Define Endog and Exog Variables

endog = xau['XAU_Close']
exog = xau_lag.drop(columns = ['XAU_Close'])

endog_train = xau[['XAU_Close']]['2000-01-01':'2021-12-31']
endog_test = xau[['XAU_Close']]['2022-01-01':]

exog_train = xau_lag.drop(columns = ['XAU_Close'])['2000-01-01':'2021-12-31']
exog_test = xau_lag.drop(columns = ['XAU_Close'])['2022-01-01':]
```

Train set contains data ranging from January 1, 2000 to December 31, 2021
Test set contains data ranging from January 1, 2022 to October 12, 2022

2

Created a function to fit a range of possible combination of parameters into the model and returns the best model based on MSE and AIC

```python
def optimize_ARIMA(endog,exog, order_list):
    """
        Return dataframe with parameters and corresponding AIC

        order_list - list with (p, d, q) tuples
        endog - the observed variable
    """

    results = []

    for order in tqdm_notebook(order_list):
        try:
            model = ARIMA(endog, exog, order=order).fit()
        except:
            continue

        aic = model.aic
        mse = model.mse
        results.append([order, aic, mse])

    result_df = pd.DataFrame(results)
    result_df.columns = ["(p, d, q)", "AIC", "MSE"]
    #Sort in ascending order, lower MSE is better
    result_df = result_df.sort_values(by='MSE', ascending=True).reset_index(drop=True)

    return result_df
```

Model Tuning

1

Created a list of 64 possible combination for the p, d, and q, period parameter. Parameter d is fixed at the first order based on the results of the differencing and ADF Test performed during the data transformation

```
# Create a range of all possible parameters
ps = range(0, 8, 1)
d = 1
qs = range(0, 8, 1)


# Create a list with all possible combination of parameters
parameters = product(ps, qs)
parameters_list = list(parameters)

order_list = []

for each in parameters_list:
    each = list(each)
    each.insert(1, 1)
    each = tuple(each)
    order_list.append(each)

# Number of possible parameter combinations
len(order_list)
```

```
64
```

2

After running the custom model tuning function, it returned the following results:

```
# Search for the combination of p,d, and q values with the lowest MSE

result_df = optimize_ARIMA(endog_train, exog_train, order_list)
result_df
```

	(p, d, q)	AIC	MSE
0	(3, 1, 5)	44611.742722	139.814425
1	(4, 1, 6)	44613.178631	139.920162
2	(5, 1, 6)	44616.196737	139.986715
3	(4, 1, 5)	44613.927283	139.998791
4	(4, 1, 4)	44610.300378	140.011916
...	...	...	...
59	(3, 1, 0)	44611.261524	142.757178
60	(2, 1, 0)	44609.874849	143.377880
61	(2, 1, 1)	44612.558302	143.771440
62	(1, 1, 0)	44611.281986	145.479937
63	(0, 1, 0)	44620.001859	149.515216

The best parameters to fit the model are the following:

- Order of autoregressive model - 3
- Order of differencing – 1
- Order of moving average model - 5

NOTE: Snippet of the Top 10 results with the lowest MSE

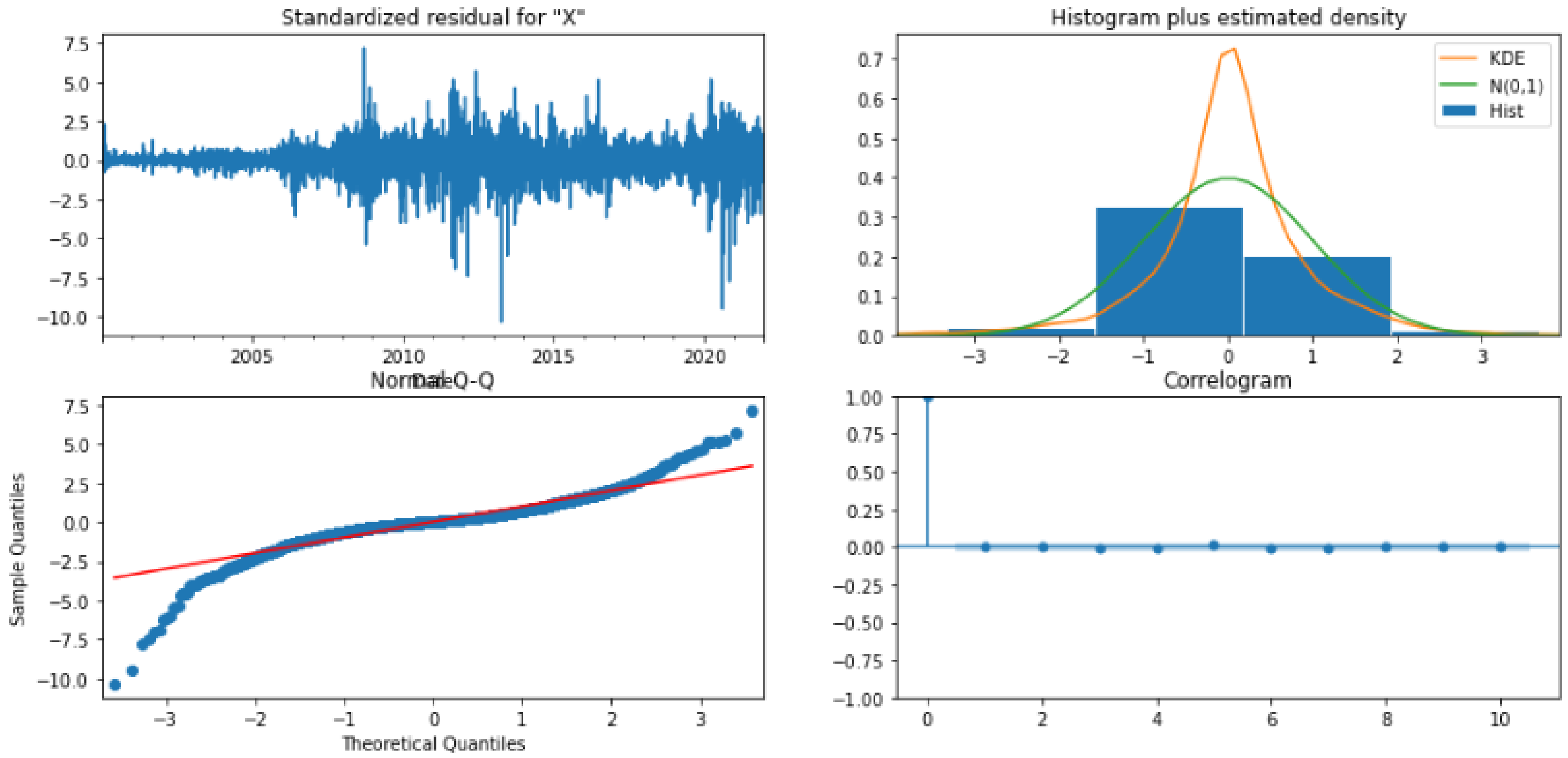
Model Fitting

1 Fitted the model using the best parameters

```
# Fit the model using the best parameters
best_model = ARIMA(endog_train, exog_train, order=(3,1,5))

res = best_model.fit()
print(res.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:              XAU_Close   No. Observations:                 5738
Model:                 ARIMA(3, 1, 5)   Log Likelihood              -22289.871
Date:                Wed, 19 Oct 2022   AIC                          44611.743
Time:                        02:37:40   BIC                          44718.218
Sample:                    01-05-2000   HQIC                         44648.802
                         - 12-31-2021
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
XAU_Open       0.7006      0.059     11.840      0.000       0.585       0.817
XAU_High      -0.0740      0.022     -3.422      0.001      -0.116      -0.032
XAU_Low       -0.0201      0.018     -1.103      0.270      -0.056       0.016
S&P500         0.0128      0.004      2.937      0.003       0.004       0.021
SLV           -0.9067      0.244     -3.712      0.000      -1.385      -0.428
OIL           -0.1186      0.077     -1.544      0.123      -0.269       0.032
EURUSD        20.4214     16.637      1.228      0.220     -12.186      53.028
ar.L1         -0.0393      1.838     -0.021      0.983      -3.642       3.564
ar.L2         -0.2975      1.252     -0.238      0.812      -2.751       2.156
ar.L3         -0.4812      1.369     -0.352      0.725      -3.164       2.202
ma.L1         -0.5842      1.831     -0.319      0.750      -4.174       3.005
ma.L2          0.3030      2.394      0.127      0.899      -4.390       4.996
ma.L3          0.3054      2.199      0.139      0.890      -4.005       4.616
ma.L4         -0.3019      0.881     -0.343      0.732      -2.028       1.425
ma.L5         -0.0017      0.016     -0.102      0.919      -0.034       0.030
sigma2       138.9213      1.202    115.530      0.000     136.565     141.278
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):             23576.69
Prob(Q):                              0.96   Prob(JB):                         0.00
Heteroskedasticity (H):               7.45   Skew:                            -0.74
Prob(H) (two-sided):                  0.00   Kurtosis:                        12.82
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

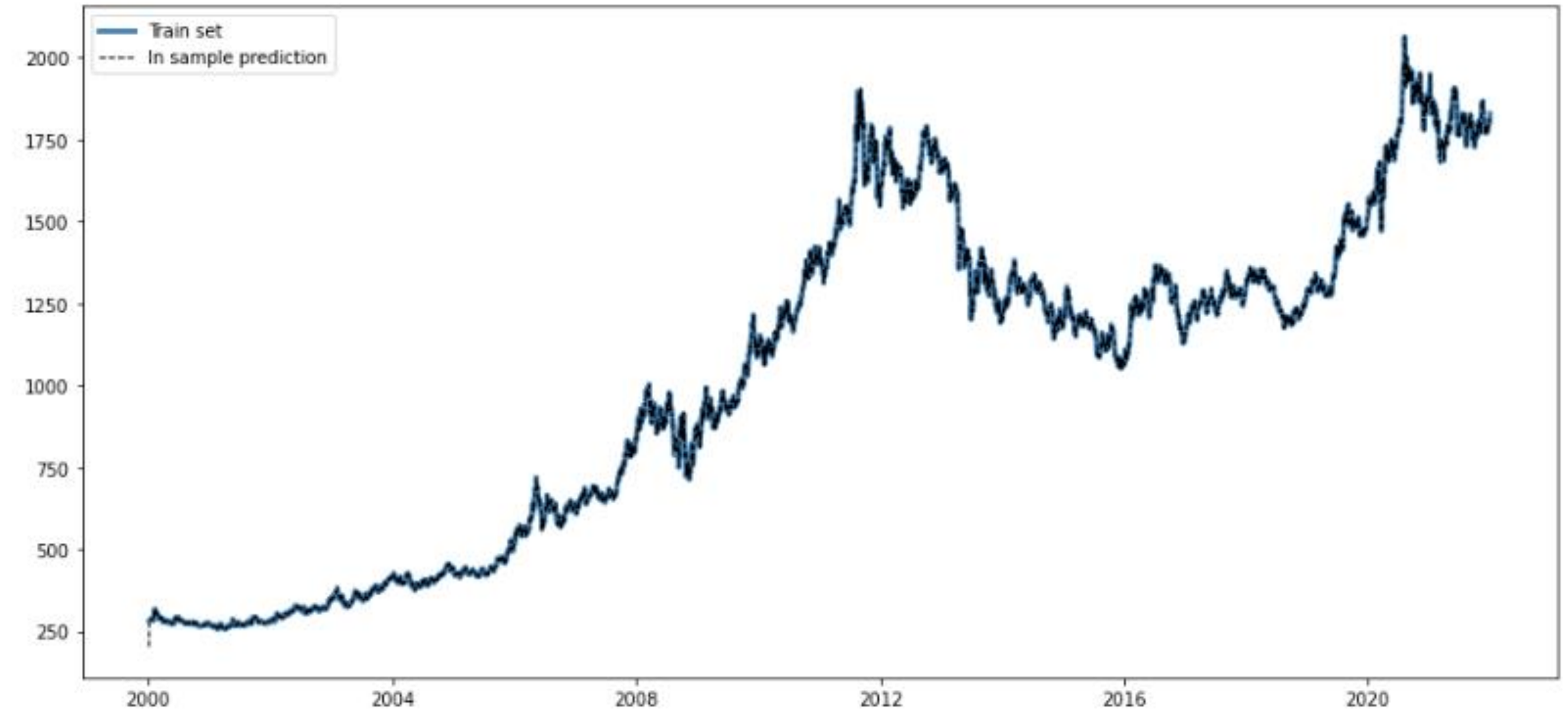
```
res.plot_diagnostics();
```



Forecasting (in-sample)

The predicted values by the ARIMAX model also follows closely the actual values from the train set. This is expected since the data used for forecasting is also the data used to fit the model.

```
#In-sample ARIMAX forecast

fig, ax = plt.subplots()

ax.plot(xau['XAU_Close'][:'2021'], 'steelblue', label = 'Train set', linewidth = 3)
ax.plot(xau_lag['Predicted'][:'2021'], 'black', linestyle = 'dashed', label = 'In sample prediction', linewidth = 1)

ax.legend(loc = 'upper left')

plt.show()

```



In-sample forecasts has a very high **coefficient of determination (R^2) value of 99%** and a low **RMSE of $11.82**

```
# In-sample ARIMAX evaluation

r2 = r2(xau['XAU_Close'][:'2021'], xau_lag['Predicted'][:'2021'])
mse = mse(xau['XAU_Close'][:'2021'], xau_lag['Predicted'][:'2021'])
rmse = np.sqrt(mse)

print(f"R2:{r2}\nMSE: {mse}\nRMSE: {rmse}")
```

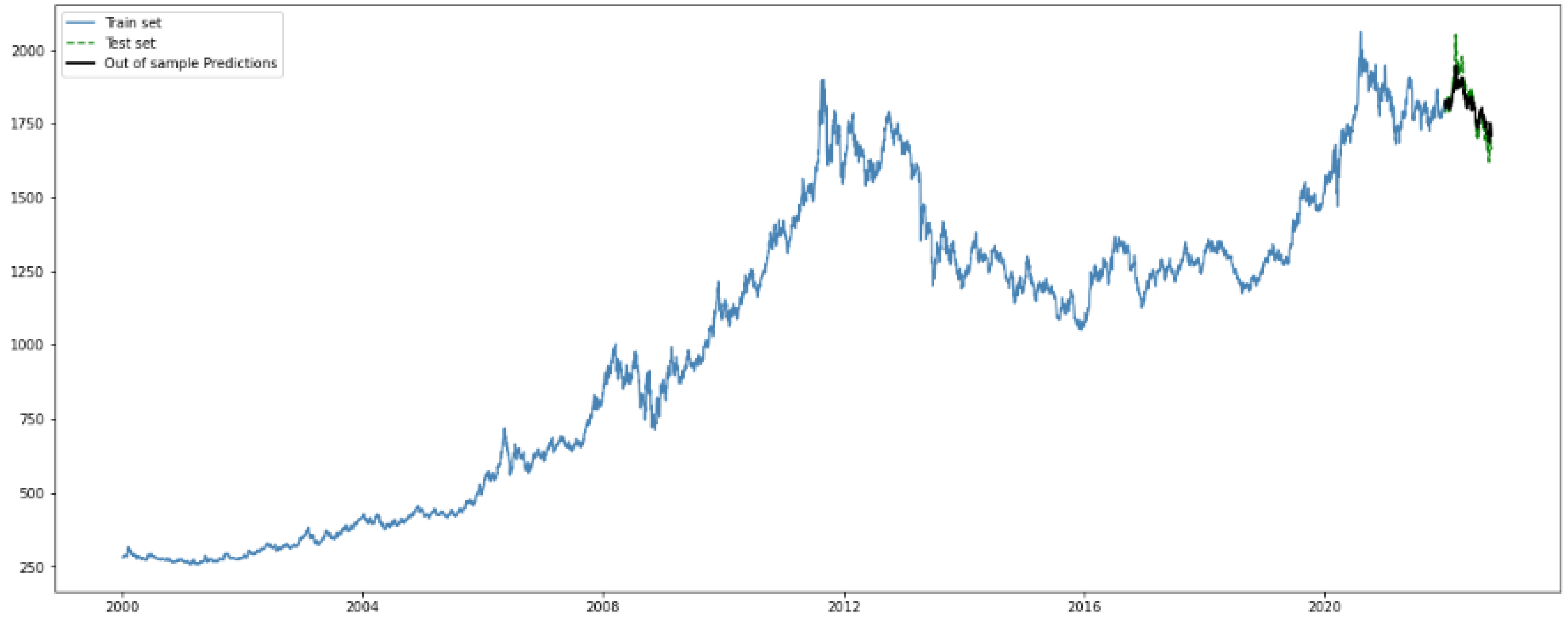
```
R2:0.999455393661319
MSE: 139.81442461109026
RMSE: 11.824314974284567
```

Forecasting & Evaluation (out-of-sample)

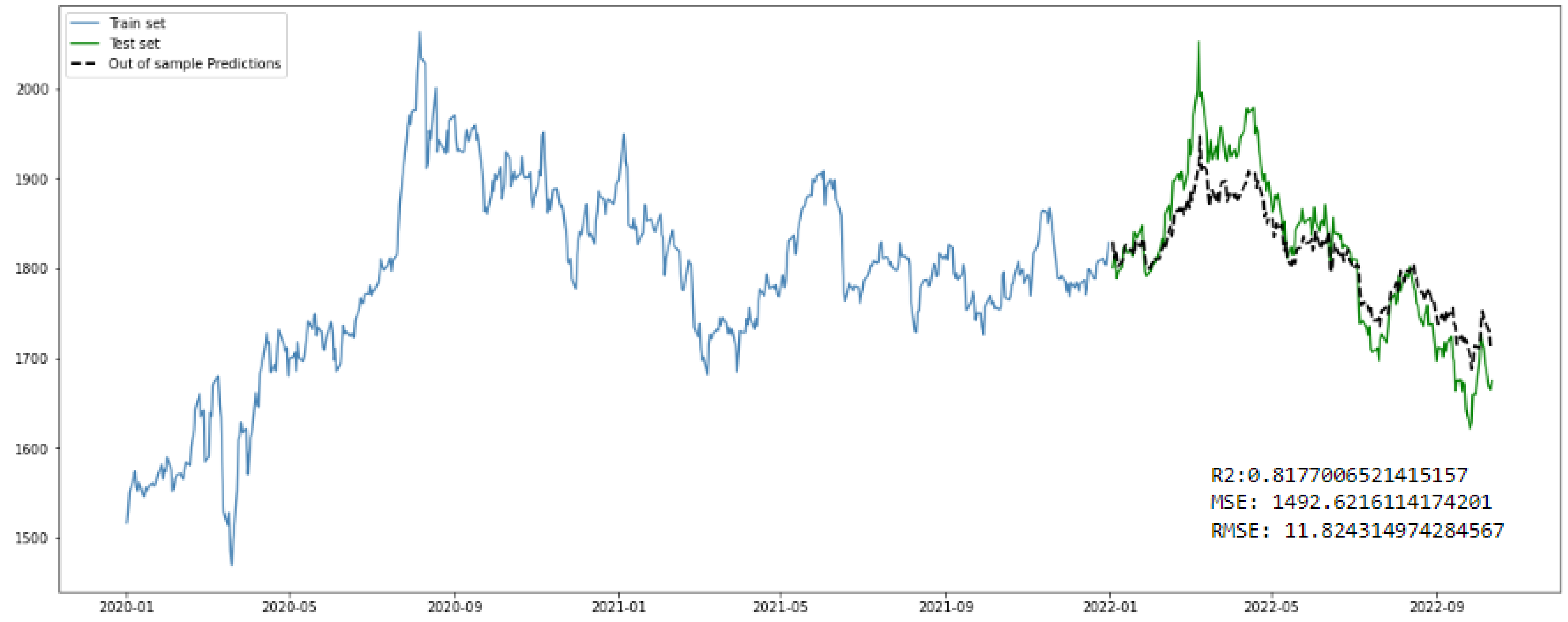
The ARIMAX forecast using out-of-sample or unseen data performed much better than that of the forecasts using the ETS model. The predicted values were close enough to the actual data

Out-of-sample ARIMAX forecasts obtained a **coefficient of determination (R^2) value of 81%** and **RMSE of $38.63**

```
#Out-of-sample ARIMAX forecast (zoomed-out)
plt.rcParams['figure.figsize'] = [20, 8]
fig, ax = plt.subplots()


ax.plot(endog_train, 'steelblue', label = 'Train set')
ax.plot(endog_test, 'green', linestyle = 'dashed', label = 'Test set')
ax.plot(forecast['Predicted'], 'black', linewidth = 2,label = 'Out of sample Predictions')

ax.legend(loc = 'upper left')

plt.show()

```




```
#Out-of-sample ARIMAX forecast (zoomed-in)
plt.rcParams['figure.figsize'] = [20, 8]
fig, ax = plt.subplots()


ax.plot(endog_train['2020':], 'steelblue', label = 'Train set')
ax.plot(endog_test, 'green', label = 'Test set')
ax.plot(forecast['Predicted'], 'black', linestyle = 'dashed', linewidth = 2,label = 'Out of sample Predictions')

ax.legend(loc = 'upper left')

plt.show()
```



Modelling and Evaluation

FOR SARIMAX MODEL

Presentation of various models that were considered

- ETS
- ARIMA
- SARIMAX

Presentation of final model selected

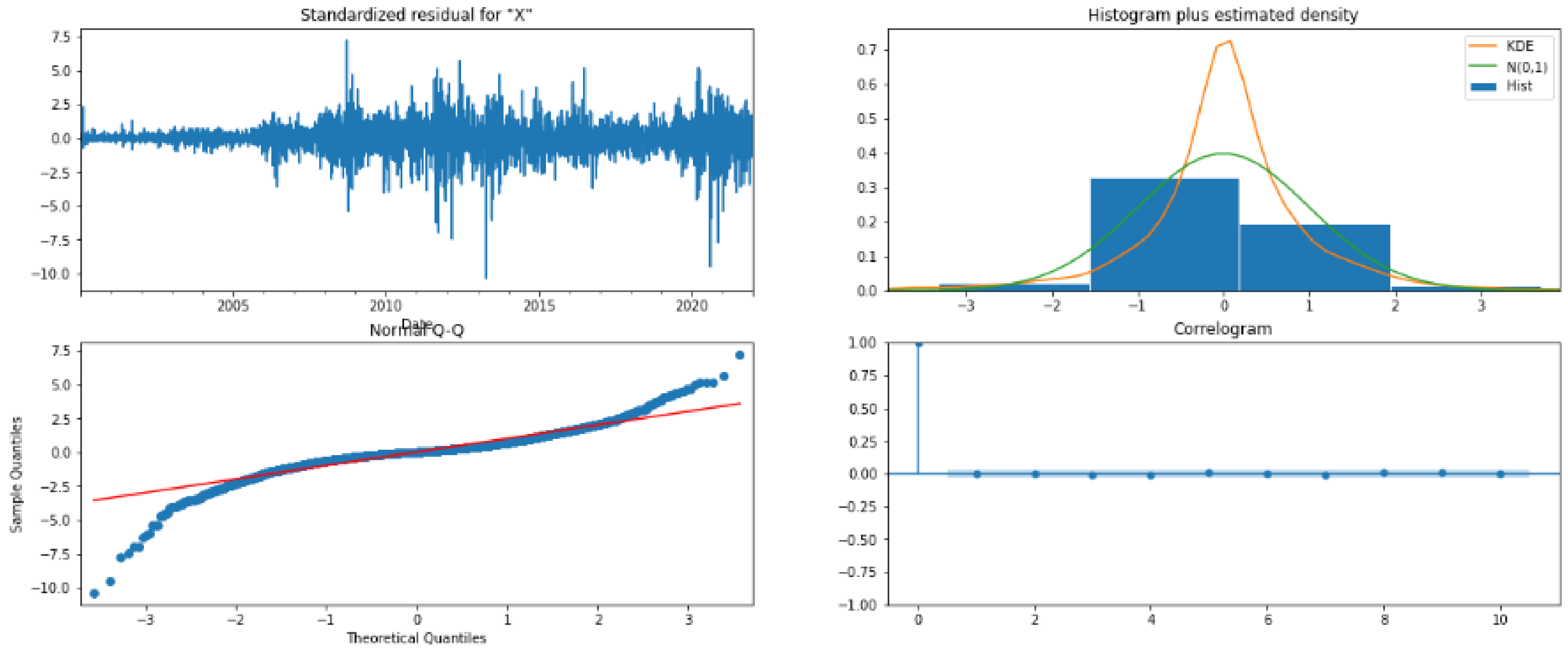
Model Fitting

1 Fitted the model using the best parameters such as the season period in the ETS model and p,d,q orders in ARIMA model

```
best_model = SARIMAX(endog_train,
                     exog_train,
                     order=(3,1,5),
                     seasonal_order=(1,0,1,90),
                     simple_differencing=False)
res = best_model.fit(dis=False)

print(res.summary())
```

```
                                 SARIMAX Results
==========================================================================================
Dep. Variable:                           XAU_Close   No. Observations:                5738
Model:             SARIMAX(3, 1, 5)x(1, 0, [1], 90)   Log Likelihood              -22289.053
Date:                             Wed, 19 Oct 2022   AIC                          44614.106
Time:                                     09:44:10   BIC                          44733.890
Sample:                                 01-05-2000   HQIC                         44655.797
                                      - 12-31-2021
Covariance Type:                               opg
==================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
XAU_Open       0.6521      0.061     10.766      0.000       0.533       0.771
XAU_High_lag1 -0.0762      0.021     -3.565      0.000      -0.118      -0.034
XAU_Low_lag1  -0.0219      0.018     -1.213      0.225      -0.057       0.014
S&P500_lag1    0.0127      0.005      2.796      0.005       0.004       0.022
SLV_lag1      -0.8715      0.251     -3.466      0.001      -1.364      -0.379
OIL_lag1      -0.0954      0.080     -1.195      0.232      -0.252       0.061
EURUSD_lag    20.4211     17.465      1.169      0.242     -13.810      54.652
ar.L1         -0.0319      1.411     -0.023      0.982      -2.797       2.733
ar.L2         -0.3029      0.977     -0.310      0.757      -2.219       1.613
ar.L3         -0.4713      1.060     -0.445      0.657      -2.549       1.606
ma.L1         -0.5420      1.404     -0.386      0.699      -3.294       2.210
ma.L2          0.3168      1.782      0.178      0.859      -3.175       3.809
ma.L3          0.3070      1.659      0.185      0.853      -2.944       3.558
ma.L4         -0.2701      0.633     -0.427      0.669      -1.511       0.970
ma.L5         -0.0014      0.017     -0.079      0.937      -0.035       0.033
ar.S.L90      -0.2214      0.718     -0.308      0.758      -1.629       1.186
ma.S.L90       0.2359      0.716      0.330      0.742      -1.167       1.639
sigma2       138.9895      1.212    114.718      0.000     136.615     141.364
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):             23671.37
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               7.46   Skew:                            -0.75
Prob(H) (two-sided):                  0.00   Kurtosis:                        12.84
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
res.plot_diagnostics();
```



Forecasting (in-sample)

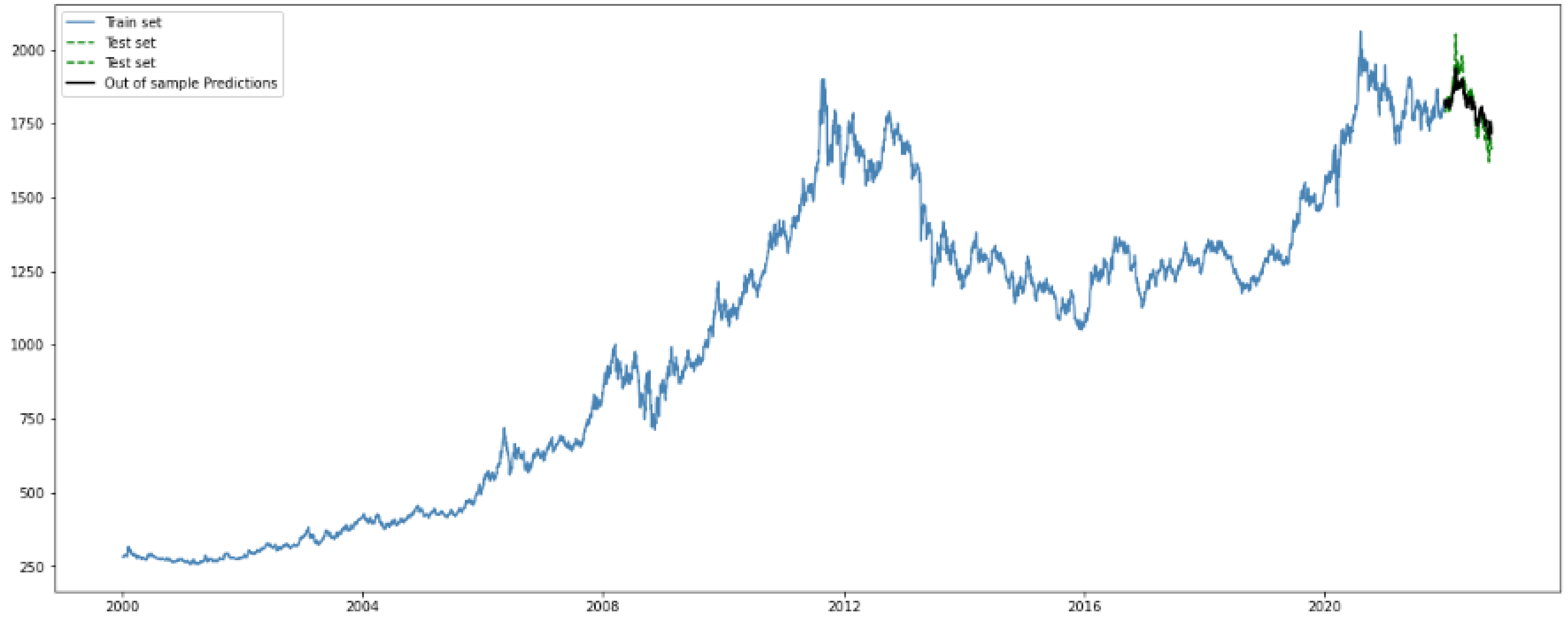
The predicted values by the SARIMAX model is fairly close to the actual values from the train set, just like the results from the ETS and ARIMA model.

```
#In-sample SARIMAX forecast

fig, ax = plt.subplots()

ax.plot(xau['XAU_Close'][:'2021'], 'steelblue', label = 'Train set', linewidth = 3)
ax.plot(xau_lag['Predicted'][:'2021'], 'black', linestyle = 'dashed', label = 'In sample prediction', linewidth = 1)

ax.legend(loc = 'upper left')

plt.show()

```

Forecasting & Evaluation (out-of-sample)

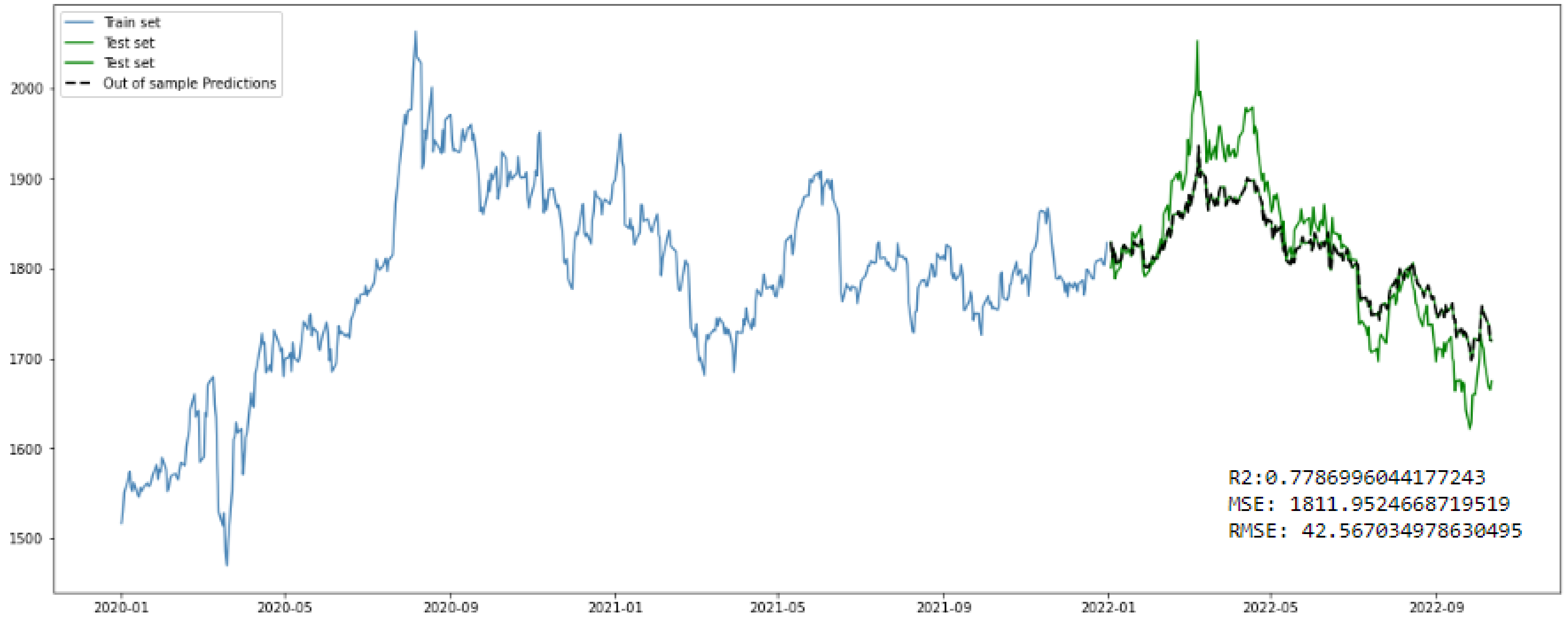
The SARIMAX forecast using out-of-sample or unseen data performed much better than that of the forecasts using the ETS model, but slightly lower than that of the ARIMAX model. Nevertheless, the predicted values were still close enough to the actual data.

```
#Out-of-sample SARIMAX forecast (zoomed-out)
plt.rcParams['figure.figsize'] = [20, 8]
fig, ax = plt.subplots()


ax.plot(endog_train, 'steelblue', label = 'Train set')
ax.plot(endog_test, 'green', linestyle = 'dashed', label = 'Test set')
ax.plot(forecast['Predicted'], 'black', linewidth = 2,label = 'Out of sample Predictions')

ax.legend(loc = 'upper left')

plt.show()

```



Out-of-sample SARIMAX forecasts obtained a **coefficient of determination (R^2) value of 78%** and **RMSE of $42.57**

```
#Out-of-sample SARIMAX forecast (zoomed-in)
plt.rcParams['figure.figsize'] = [20, 8]
fig, ax = plt.subplots()


ax.plot(endog_train['2020':], 'steelblue', label = 'Train set')
ax.plot(endog_test, 'green', label = 'Test set')
ax.plot(forecast['Predicted'], 'black', linestyle = 'dashed', linewidth = 2,label = 'Out of sample Predictions')

ax.legend(loc = 'upper left')

plt.show()
```



Model Comparison

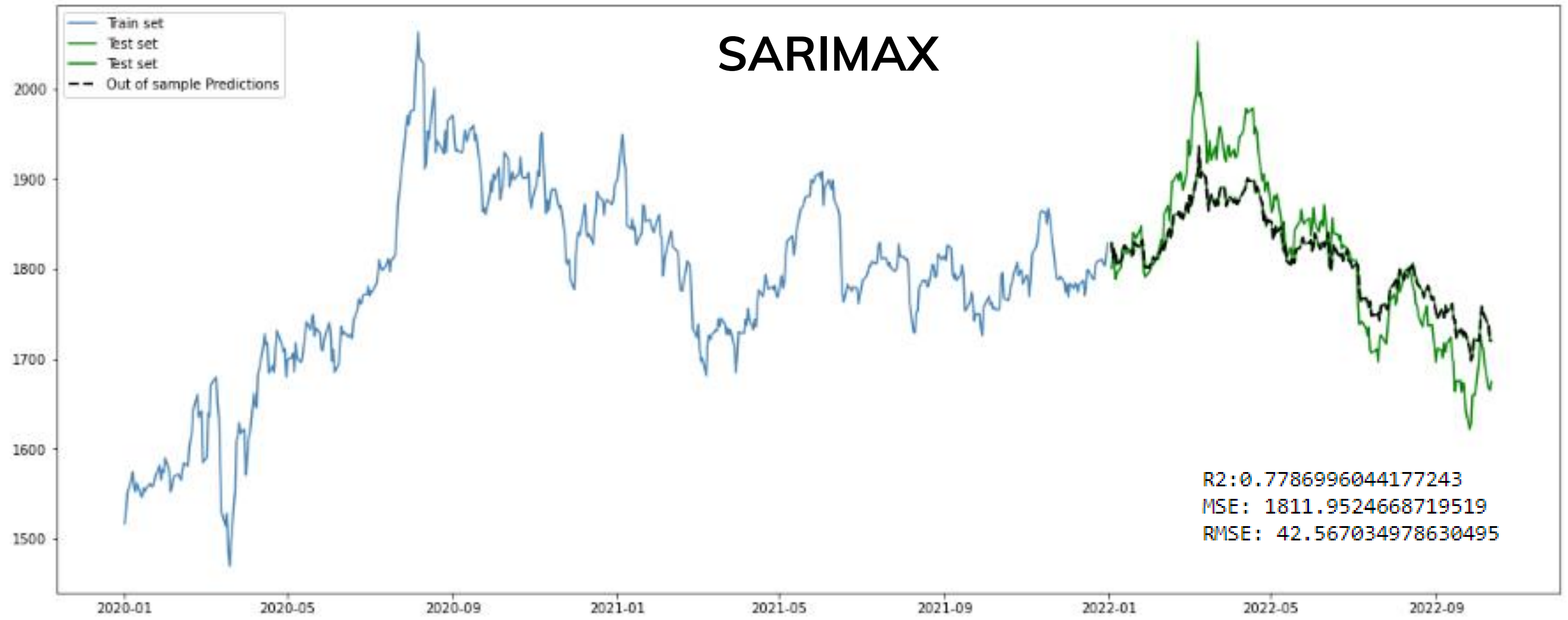
- ERROR-TREND-SEASONALITY
- ARIMAX
- SARIMAX

Model Evaluation (VISUALIZATION)

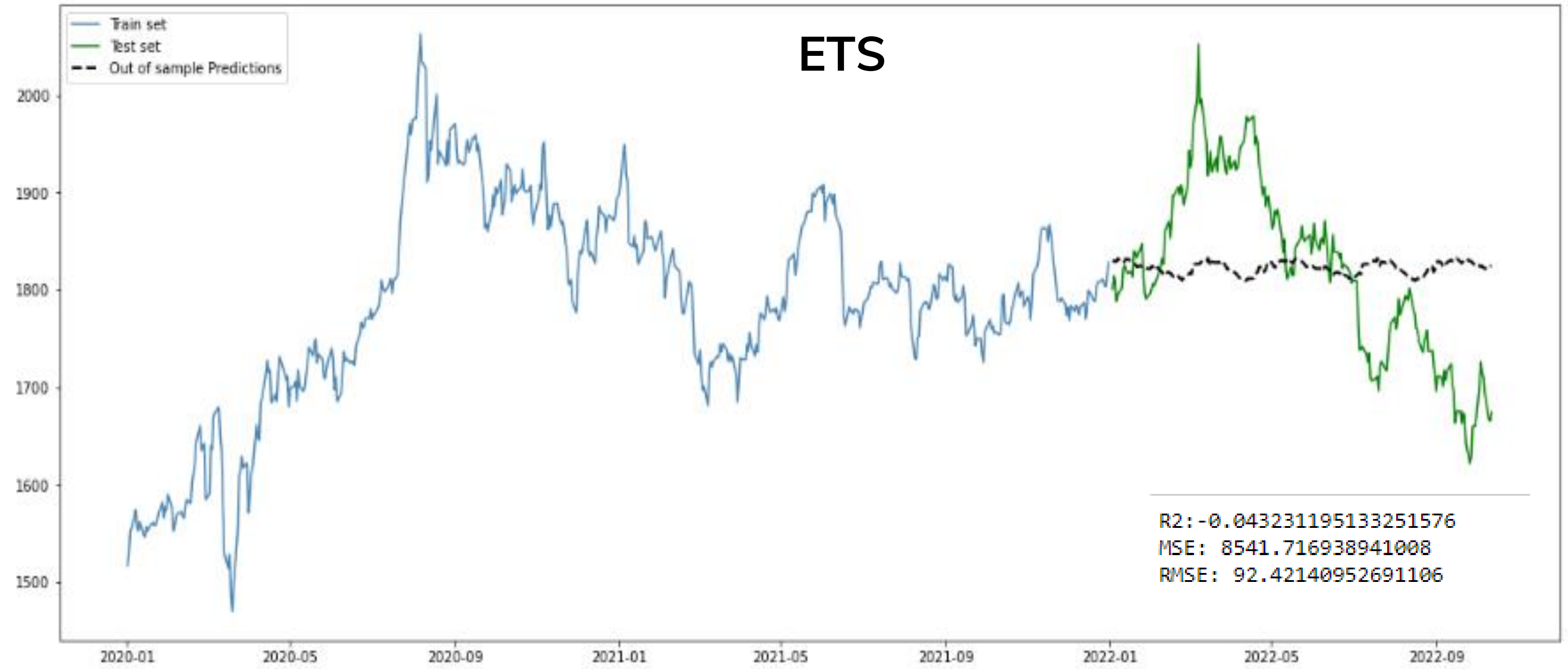
ARIMAX (best model)

```python
#Out-of-sample ARIMAX forecast (zoomed-in)
plt.rcParams['figure.figsize'] = [20, 8]
fig, ax = plt.subplots()


ax.plot(endog_train['2020':], 'steelblue', label = 'Train set')
ax.plot(endog_test, 'green', label = 'Test set')
ax.plot(forecast['Predicted'], 'black', linestyle = 'dashed', linewidth = 2,label = 'Out of sample Predictions')

ax.legend(loc = 'upper left')

plt.show()
```

SARIMAX



ETS



Model Evaluation (METRICS)

MODEL	AIC	BIC	R2	MSE	RMSE
ETS	41,736.553	42,382.092	-0.0432	8,541.7169	92.4214
ARIMAX	44,611.743	44,718.218	0.8177	1,492.6216	38.6345
SARIMAX	44,614.106	44,733.890	0.7787	1,811.9525	42.5670

The final model that will be used will be the ARIMAX model with the following exogenous variables:

- XAU_Open
- XAU_High_LAG1
- XAU_Low_LAG1
- S&P500_LAG1
- SLV_LAG1
- OIL_LAG1
- EURUSD_LAG1

With an autocorrelation (p) order of 3, differencing order (d) of 1, and moving average order (q) of 5.

This model is chosen as it yielded the highest R2 and lowest RMSE despite it not having the lowest AIC and BIC.
Its predictions are closest to the actual values compared to the other models and it is easier to model with lower run time compared to SARIMAX.
Timeseries forecasting, especially those that are related to stocks and commodities are inherently hard to model so having a coefficient of determination equal to 81% can already be considered as a feat.

Recommendation

The business objective of this project is to create a model that could forecast future Gold (XAUUSD) prices in order to maximize trading gains.

With the ARIMAX model, we were able to create a model with an R2 score of 81%.

Since the model is not 100% accurate in its forecasts, this model is recommended to be used to supplement the fundamental analysis and technical analysis being conducted by the traders to be able to devise a trading strategy and entry and exit plans.

Improvements can also be made to the model by including variables such as the sentiment score of a certain news that will affect the gold price. Also, more advanced deep learning models like LSTM, CNN, and RNN may be used to further increase the model's accuracy.

Thank you!