Capstone Project 2 - Gold ETFs Price Forecast, Trends, & 2 Year Predictions Milestone Report

Project Scope

The scope of this project is focused on GOLD ETFs price forecasting, trends and 2 year predictions. The goal of this project is to understand and apply time-series models like ARIMA, SARIMA and Facebook Prophet in forecasting the price of gold ETFs.

What is The Problem?

Many investors buy gold as a safe haven to protect themselves against a possible catastrophe, profit from these tremendous increases in the price of gold, diversify their portfolio and protect themselves against inflation.

With recent fears of an economic recession looming in the distance, investors are looking to recession-proof their portfolios. Gold has historically been touted as an asset that booms in a recession because unlike fiat currencies such as the Dollar, Yen and the Pound, it has inherent value as a commodity currency.

However, with the rapidly changing economic landscape, does this still hold true? Is gold a good commodity to buy now for later profits?

Where Is The Data From?

SPDR® Gold Shares (NYSE Arca : GLD) is a cost-effective and convenient way to invest in gold without buying the real gold.

The historical prices of SPDR® Gold Shares (NYSE Arca : GLD) was downloaded from Yahoo. Data spans from the inception of this share from 11/18/2004 to the date of download, 11/22/2019.

Exploratory Data Analysis

There are 3780 rows of data with 7 columns. With each data, there's the Opening price, High Price, Low Price, Close Price, Adjusted Close Price and Volume.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-11-18	44.430000	44.490002	44.070000	44.380001	44.380001	5992000
1	2004-11-19	44.490002	44.919998	44.470001	44.779999	44.779999	11655300
2	2004-11-22	44.750000	44.970001	44.740002	44.950001	44.950001	11996000
3	2004-11-23	44.880001	44.919998	44.720001	44.750000	44.750000	3169200
4	2004-11-24	44.930000	45.049999	44.790001	45.049999	45.049999	6105100
5	2004-11-26	45.250000	45.599998	45.060001	45.290001	45.290001	3097700
6	2004-11-29	45.099998	45.500000	45.080002	45.400002	45.400002	3759000
7	2004-11-30	45.369999	45.410000	44.820000	45.119999	45.119999	3857200
8	2004-12-01	45.279999	45.590000	45.259998	45.380001	45.380001	2037500

Any Missing Data?

Each of the column, there's no missing data.

```
data_gld.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3780 entries, 0 to 3779
Data columns (total 7 columns):
Date          3780 non-null object
Open          3780 non-null float64
High          3780 non-null float64
Low           3780 non-null float64
Close         3780 non-null float64
Adj Close     3780 non-null float64
Volume        3780 non-null int64
dtypes: float64(5), int64(1), object(1)
memory usage: 206.8+ KB
```

However, if you look at the dataframe with the dates, we do not have prices for every date. That's because the stock market is closed on weekends and holidays. We need to fill in the missing days to make this data set a truely daily time series data.

Missing 2004-11-20
2004-11-21 →

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-11-18	44.430000	44.490002	44.070000	44.380001	44.380001	5992000
1	2004-11-19	44.490002	44.919998	44.470001	44.779999	44.779999	11655300
2	2004-11-22	44.750000	44.970001	44.740002	44.950001	44.950001	11996000
3	2004-11-23	44.880001	44.919998	44.720001	44.750000	44.750000	3169200
4	2004-11-24	44.930000	45.049999	44.790001	45.049999	45.049999	6105100
5	2004-11-26	45.250000	45.599998	45.060001	45.290001	45.290001	3097700
6	2004-11-29	45.099998	45.500000	45.080002	45.400002	45.400002	3759000
7	2004-11-30	45.369999	45.410000	44.820000	45.119999	45.119999	3857200
8	2004-12-01	45.279999	45.590000	45.259998	45.380001	45.380001	2037500

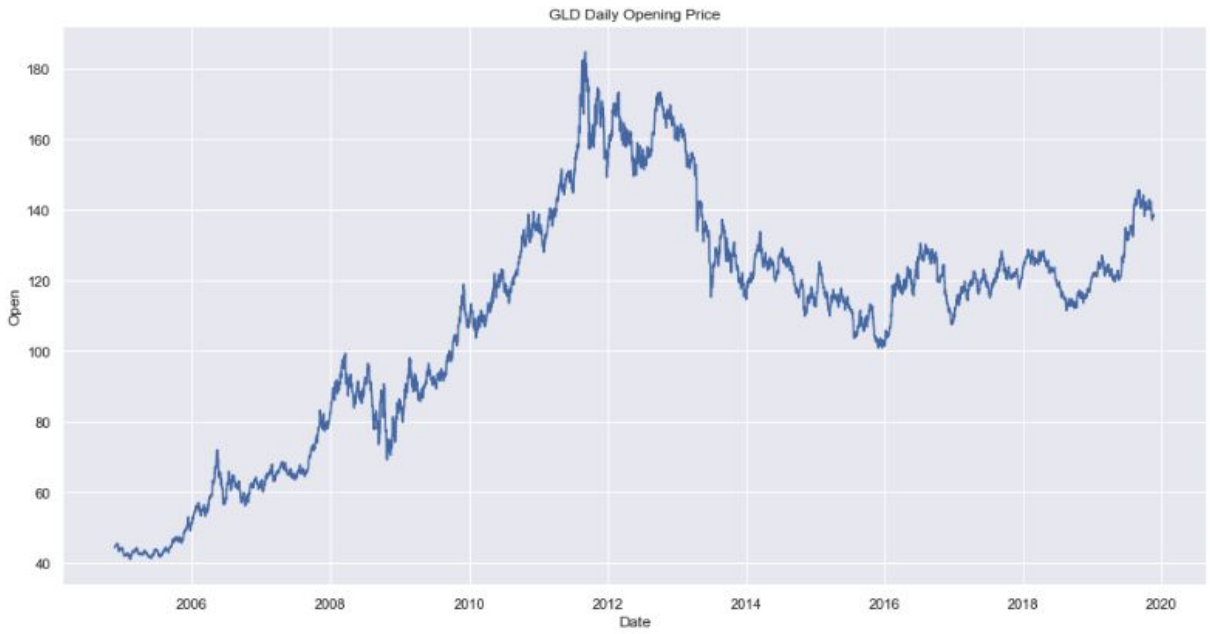
Forward Fill Missing Dates

For days where this is no pricing information, we re-sample the Day and fill in the missing values from the previous day.

these 2 missing
dates are added to
the dataframe →

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-11-18	44.430000	44.490002	44.070000	44.380001	44.380001	5992000
1	2004-11-19	44.490002	44.919998	44.470001	44.779999	44.779999	11655300
2	2004-11-20	44.490002	44.919998	44.470001	44.779999	44.779999	11655300
3	2004-11-21	44.490002	44.919998	44.470001	44.779999	44.779999	11655300
4	2004-11-22	44.750000	44.970001	44.740002	44.950001	44.950001	11996000
5	2004-11-23	44.880001	44.919998	44.720001	44.750000	44.750000	3169200

After this transformation, the dataframe now has 5483 rows of data and here's the plot of the Open Price for all 5483 rows.



Is The Time Series Data Stationary?

Dicky-Fuller Test

The time series data has to be stationary before any modeling. We use the Dicky-Fuller test on the Open Price.

Dicky-Fuller Results
ADF Statistic: -1.8025174981713052
p-value: 0.3792125439071432

Based on the results above, the data is not stationary because the p-value is greater than 0.05.

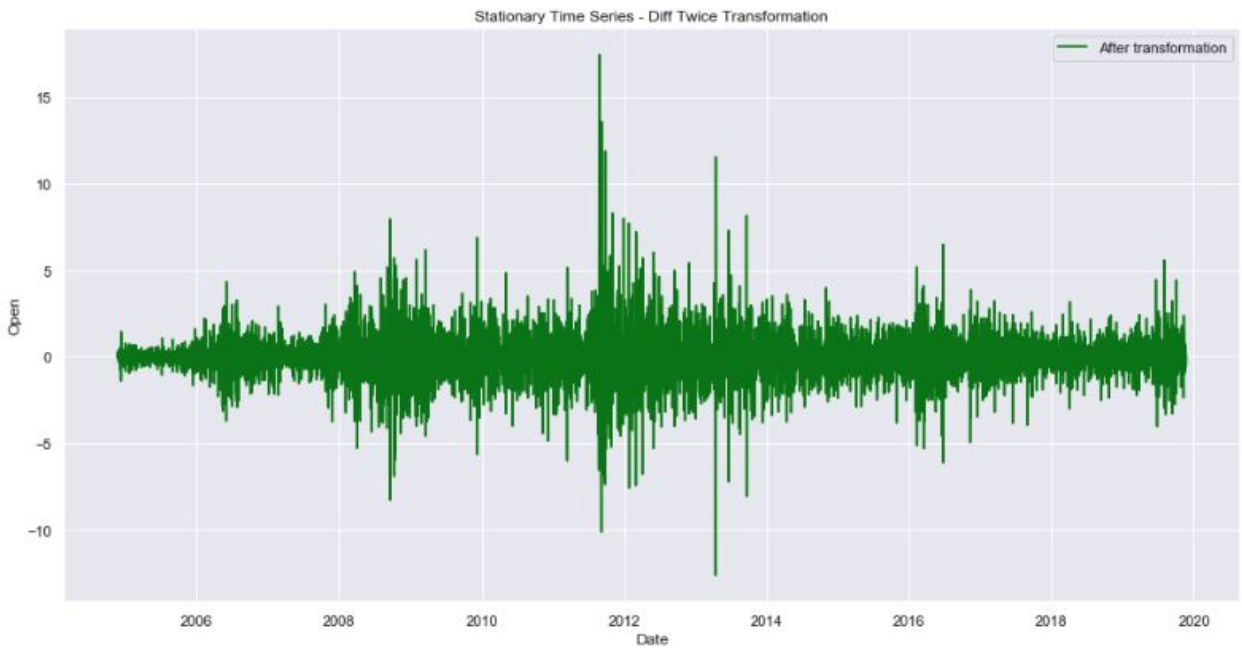
Make The Time Series Data Stationary

There are different ways to make time series data stationary. A few methods include the difference once method, difference twice and square root method. We are going to use all 3 and pick the best method. After we apply each of the methods, we apply the Dicky-Fuller test and the results are below:

Method	**ADF Statistic**	**P-Value**
Difference Once	-16.738250093306753	1.3650617859601889e-29
Difference Twice	-21.13672510562556	0.0
Square Root	-2.0279705213801678	0.27444523490828154

The Square Root methods didn't produce a p-value less than 0.05. So we should eliminate it. Both Differencing once and twice methods produced a p-value less than 0.05 but Differencing Twice produced a much more negative ADF Statistic. That's what we want, the more negative the better.

Let’s compare the time series data before differencing twice and after.



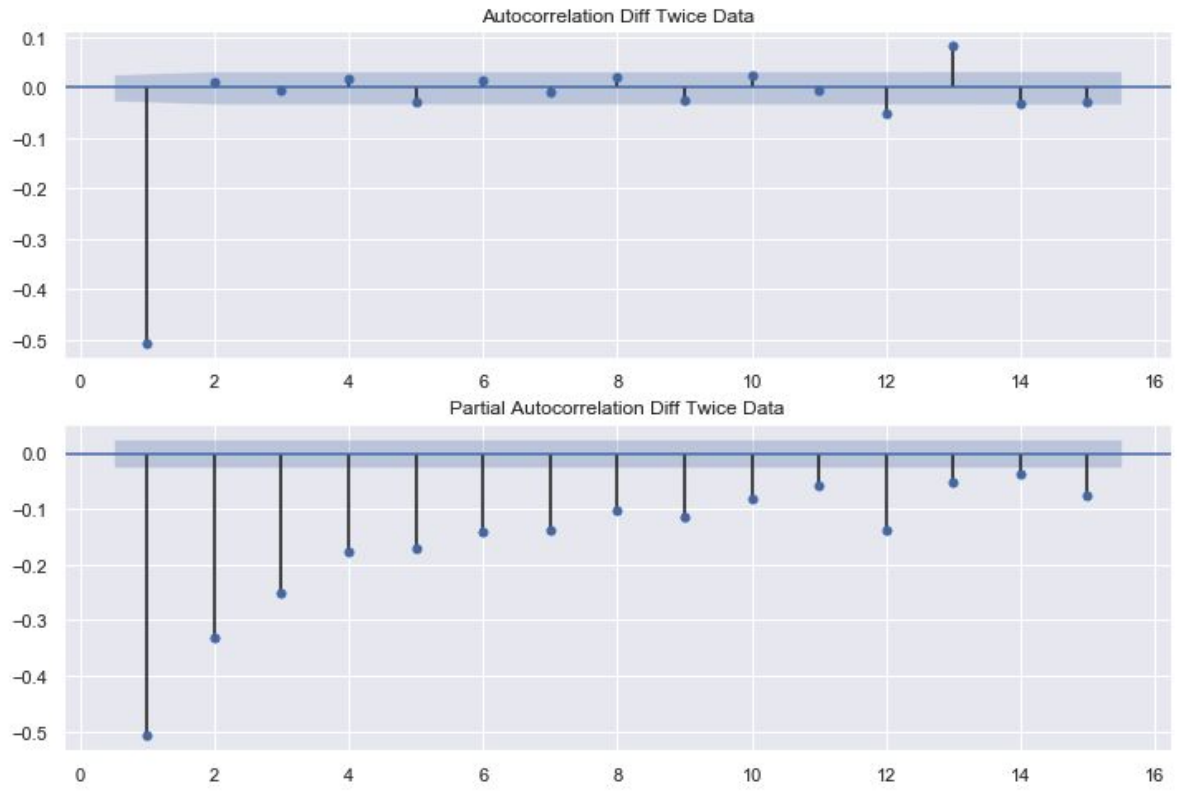
How To Choose the Right Model Order?

ACF and PACF

Autocorrelation Function (ACF) is the correlation between the time series and the same time series lag by X number of steps.

Partial Autocorrelation (PACF) is the correlation between the time series and the lag version of itself after we subtract the effect of correlation at smaller lags.So, it's just the correlation associated with just that particular lag.

By plotting both the ACF and PACF charts, it can help us select the right model order.



The initial observation shows that the ACF cuts off after lag 1 and PACF tails off over time. This may indicate that it's a Moving Average model with an order of 1, that's MA(1).

However, there are limitations to the ACF and PACF method. We are making the judgement based on how the ACF and PACF graphs look. Sometimes it may not be as clear to make a conclusion. We are going to use AIC and BIC results next to select the right model order.

AIC and BIC

Akaike Information Criterion (AIC)

- Lower AIC indicates a better model
- AIC is ideal for simple models with lower order
- Better at choosing predictive models (use this if that's our goal)

Bayesian Information Criterion (BIC)

- Lower BIC indicates a better model
- BIC penalizes complex models
- Better at choosing good explanatory model (use this if that's our goal)

Most of the times, AIC and BIC will select the same model order. However, when they don't, we have to make a choice which to pick.

We wrote a recursive loop to iterate through values of p and q to find the best combination with the lowest AIC result.

Results sorted by AIC

	p	q	aic	bic
1	0	1	15870.084038	15883.302124
2	0	2	15871.949100	15891.776229
4	1	1	15871.977921	15891.805050
7	2	1	15873.328543	15899.764715
5	1	2	15873.984951	15900.421122
8	2	2	15875.843240	15908.888454
6	2	0	17419.185255	17439.012384
3	1	0	18053.677637	18066.895723
0	0	0	19683.644070	19690.253113

Results sorted by BIC

	p	q	aic	bic
1	0	1	15870.084038	15883.302124
2	0	2	15871.949100	15891.776229
4	1	1	15871.977921	15891.805050
7	2	1	15873.328543	15899.764715
5	1	2	15873.984951	15900.421122
8	2	2	15875.843240	15908.888454
6	2	0	17419.185255	17439.012384
3	1	0	18053.677637	18066.895723
0	0	0	19683.644070	19690.253113

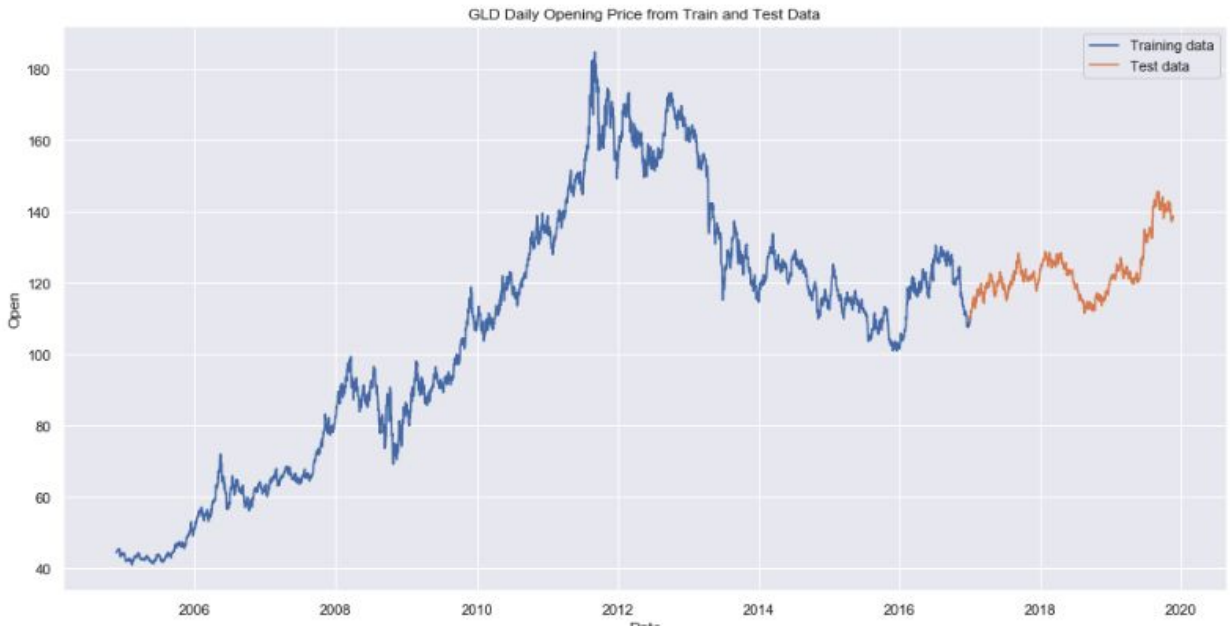
Both AIC and BIC selected the same model order. This is the same results as the ACF and PACF analysis where we determined it was a MA(1) model order.

Train-Test Data Split

Because we are dealing with time series data, we cannot split the data randomly. The earlier data should always be the training data and the later data should be in the test set.

There are 15 years of data. We are going to use the first 12 years (2004 - 2016) as training data and the last 3 years (2017 - 2019) as test data.

The train data set has 4427 rows of data and the test data set had 1056 rows of data. This is how they look when plotted together.



Modeling Using ARIMA

Method 1: ARIMA Model with No Seasonality & One-Step Ahead Forecast

When we forecast differenced time series, we end up with forecasted data of the difference. We have to use the difference to reverse engineer in order to derive the forecasted price. However, we can use ARIMA model to avoid all the work! It takes care of the reverse engineering part automatically!

We use SARIMAX model here as it accounts for seasonality parameters if seasonalty exists.

model = SARIMAX(df, order=(p,d,q), seasonal_order=(P,D,Q,S))

Non-seasonal order

- p: autoregressive order
- d: differencing order
- q: moving average order

Seasonal order (leave it blank if there's no seasonality)

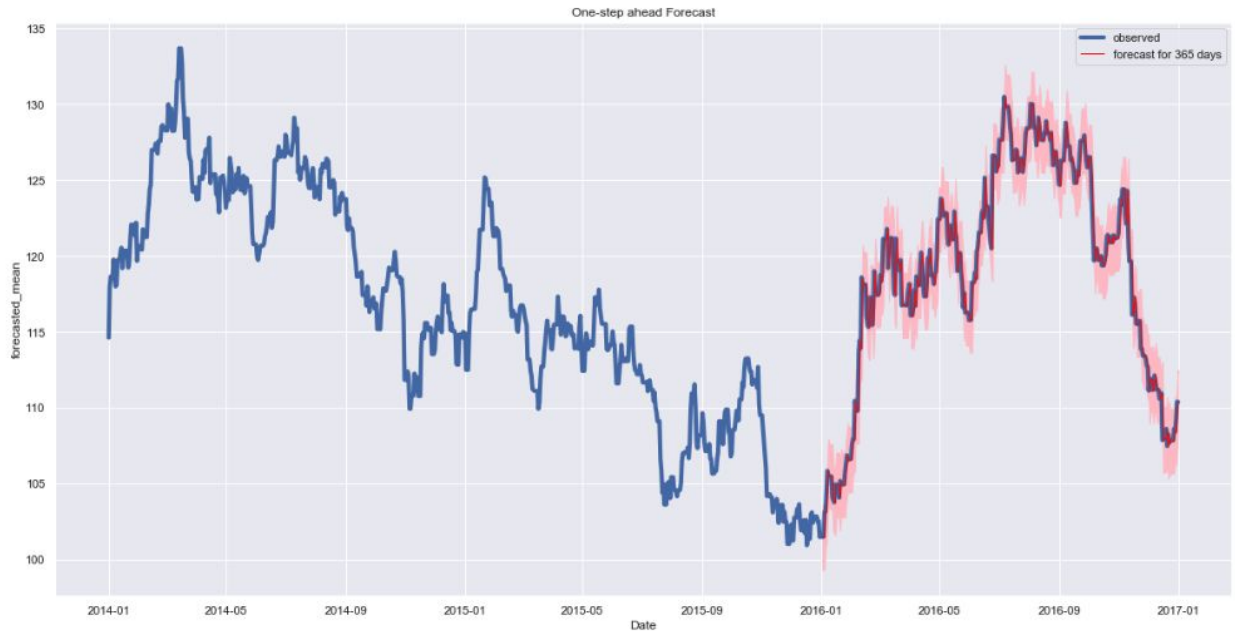
- P: seasonal autoregressive order

- D: seasonal differencing order
- Q: seasonal moving average order
- S: number of time steps per cycle

Based on our previous findings, difference order = 2 and the moving average order is 1. We are assuming there's **no seasonality** here, that's why P,D,Q are not provided in the following model:

model = SARIMAX(train_data['Open'], order=(0,2,1), trend= 'c')

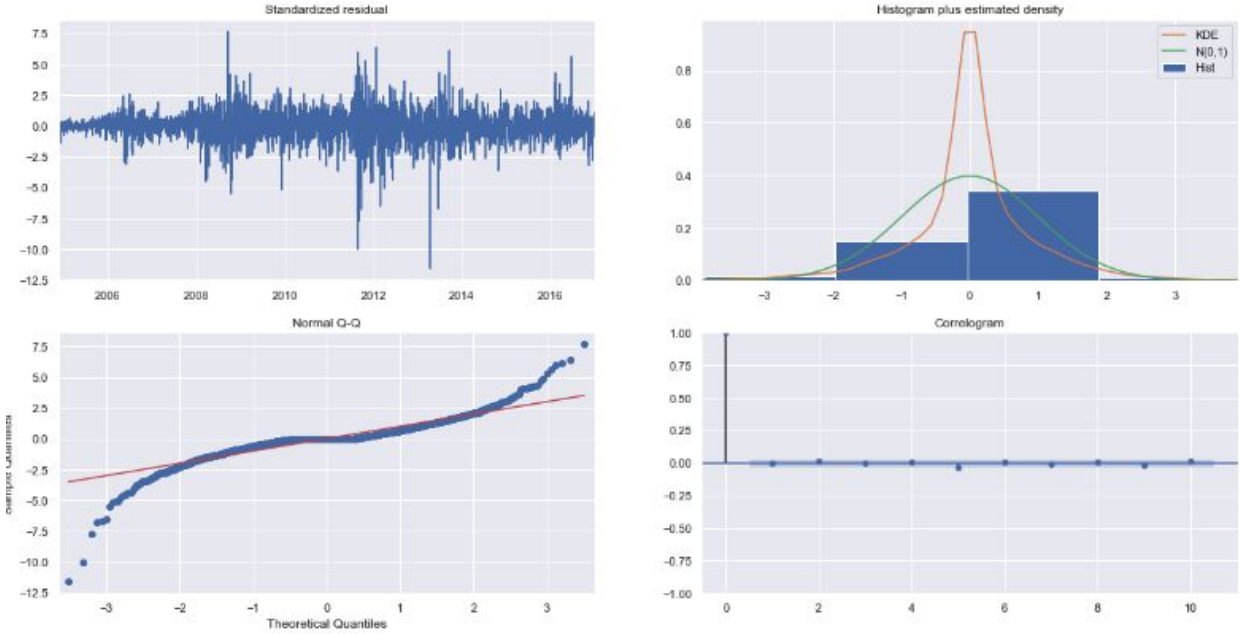
We created a fit based on the model above and used it to predict the open price of gold ETF for the last 365 days of the training data. We then compared the prediction against the real prices for the last 365 days if the training set.



Overall, the above forecasts aligns very well with the true values for the last 365 days of the training data and it falls within the confidence intervals. Let's look at the model diagnostics results next.

Model Diagnostics Results

Here's the plot diagnostic results:



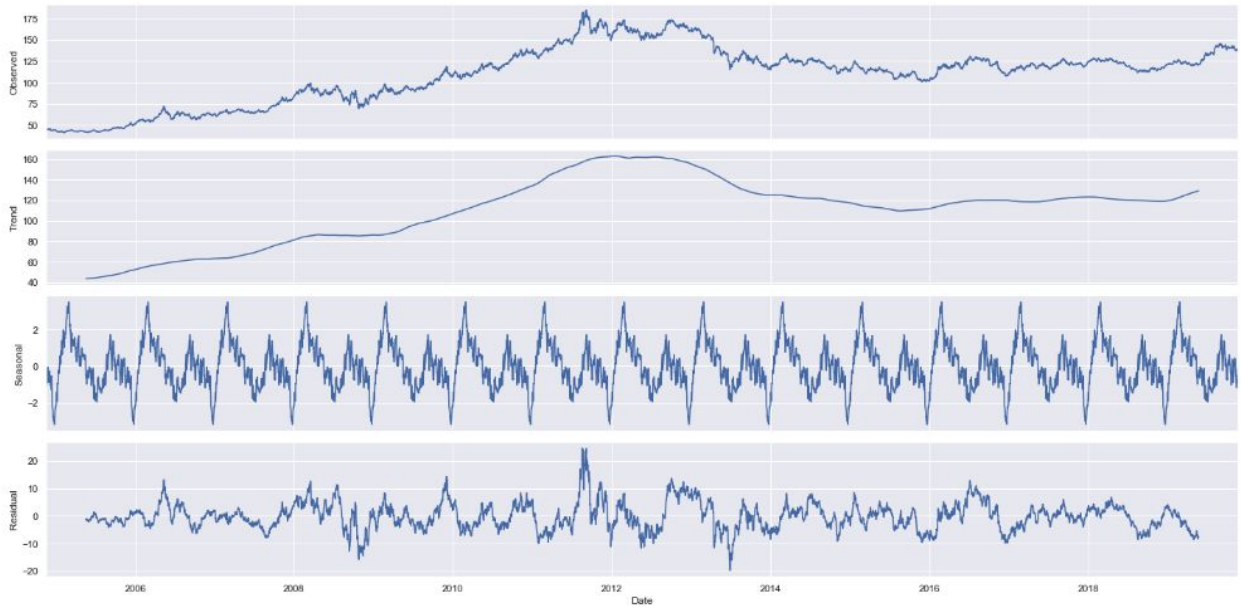
- **Standardized Residual Plot** - The graph doesn't seem to show a trend. That's what we want.
- **Histogram Plus Estimated Density** - This shows the distribution of the residuals. The green line shows a normal distribution and the orange line needs to be as close to the green line. The 2 lines are very different in this case. This model might need tweaking.
- **Normal Q-Q** - This shows how the distribution of the residuals compared to a normal distribution. Most of the residuals are on the line except the ends.
- **Correlogram** - ACF plot of the residuals. 95% of the data where lag > 0 should not be significant. That means, they need to be within the blue shaded area. Based on the graph, it looks OK as 95% of the data is not significant, they are within the blue shaded area.

MAE, MSE and RMSE Scores

- **Mean Absolute Error (MAE):** 0.64
- **Mean Squared Error (MSE):** 1.04
- **Root Mean Squared Error (RMSE):** 1.02. Our model forecasted the average daily open price in the training set is within $1.02 of the real open prices.

Method 2: Auto ARIMA Model with Seasonality & One-Step Ahead Forecast

The previous method doesn't take into consideration any seasonality. Let's use seasonal_decompose to check for any seasonality. Here's the results:



The plots above shows that the trend in prices of gold is not consistent but there is some obvious seasonality. Instead of guessing, we are going to use Auto Arima to perform an automatic grid search to discover the optimal order for an ARIMA model. It will also highlight if there's any seasonality in the data.

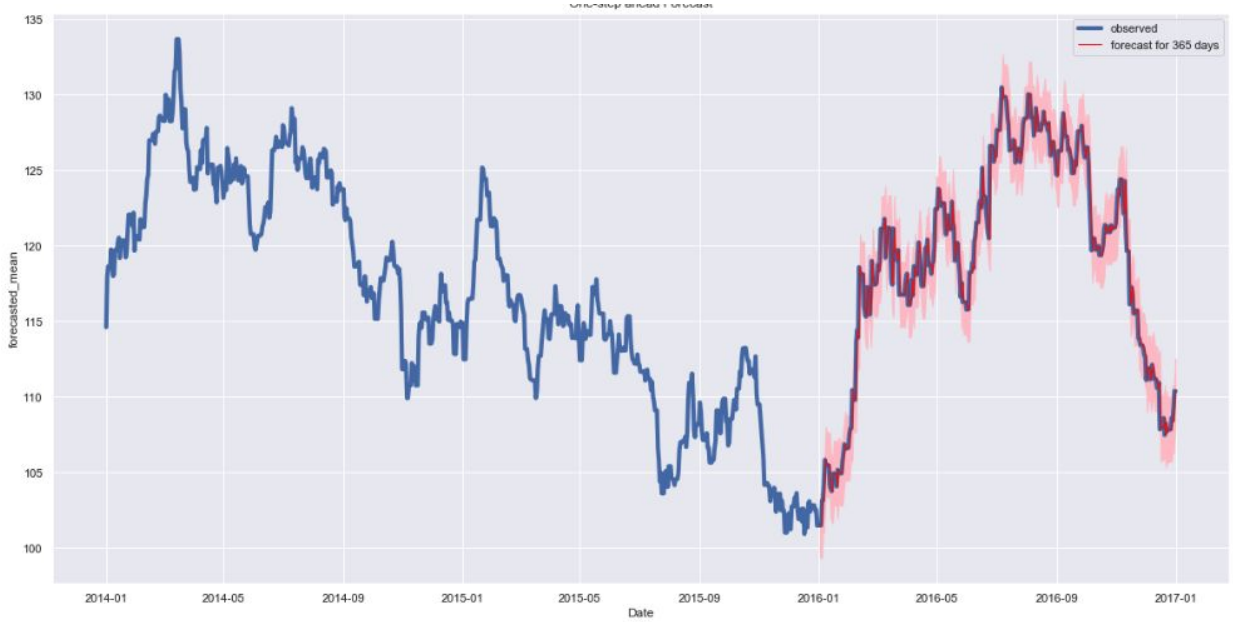
For Auto Arima to work, we need to create a univariate dataset. We need to drop all other columns other than Open Price. This is the final result of the dataset.

	Open
Date	
2004-11-18	44.430000
2004-11-19	44.490002
2004-11-20	44.490002
2004-11-21	44.490002
2004-11-22	44.750000

Based on the results generated by auto_arima that produced the lowest AIC score, the **Best Fit ARIMA is:** order=(0, 1, 0) seasonal_order=(0, 0, 0, 7).

Auto Arima didn't detect any seasonality and suggested the differencing is ONLY ONCE and not TWICE which we concluded earlier using the Dicky-Fuller test.

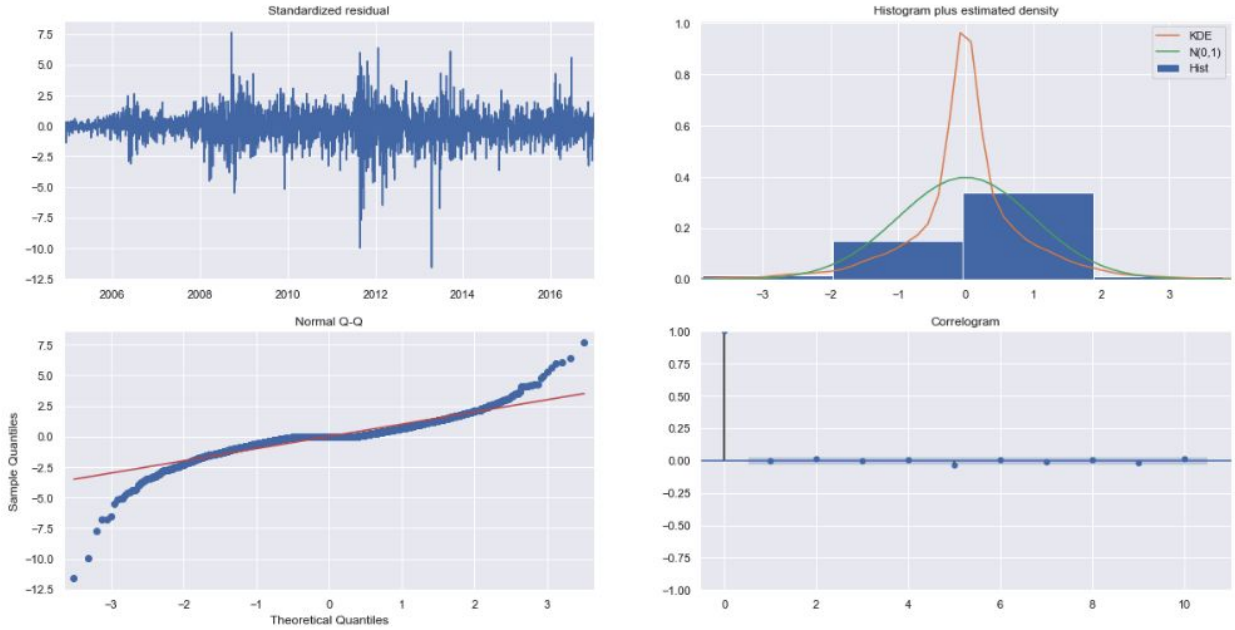
We created a model using the Best Fit ARIMA order and used it to predict the open price of gold ETF for the last 365 days of the training data (just like Method 1). We then compared the prediction against the real prices for the last 365 days if the training set.



Just like method 1, the forecast aligns very well with the true values for the last 365 days of the training data and it falls within the confidence intervals. Let’s look at the model diagnostics results next.

Model Diagnostics Results

Here's the plot diagnostic results:



These 4 diagnostic plots shows no alarming difference from Method 1 diagnostic plots.

MAE, MSE and RMSE Scores

- **Mean Absolute Error (MAE):** 0.63
- **Mean Squared Error (MSE):** 1.04
- **Root Mean Squared Error (RMSE):** 1.02. Our model forecasted the average daily open price in the training set is within $1.02 of the real open prices.

Conclusion: Method 2 of MAE performed slightly better than Method 1. However, there's no improvement in MSE and RSME.

Method 3: ARIMA Model with Seasonality & One-Step Ahead Forecast - Manual Grid Search

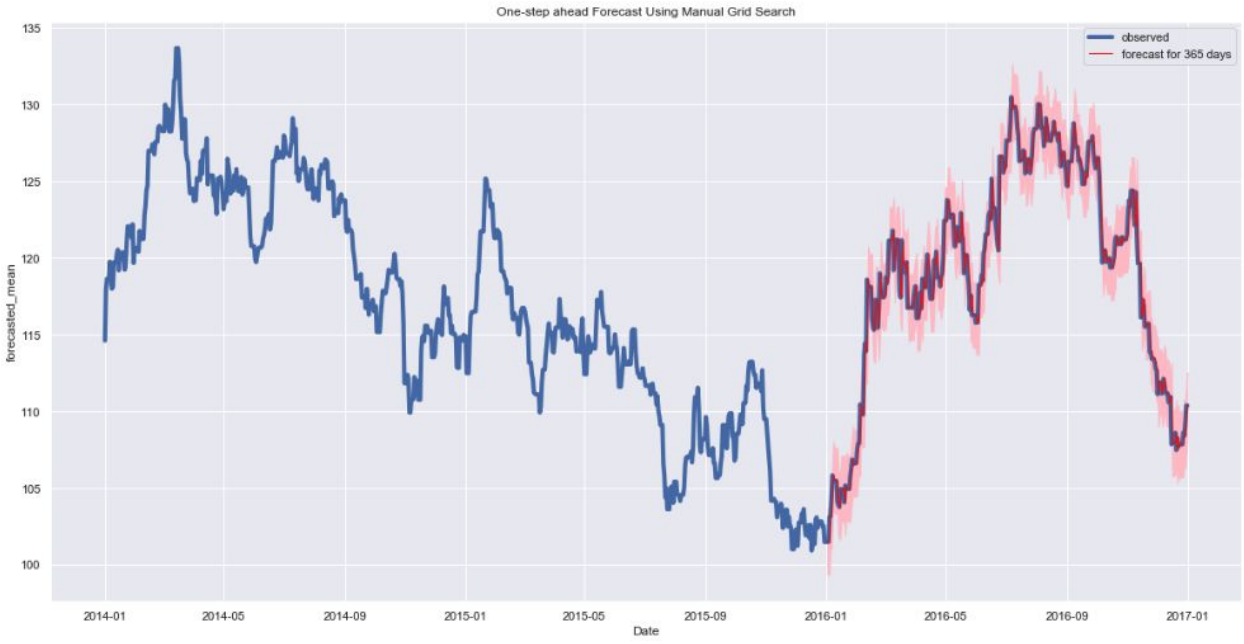
In Method 2, we used an automatic grid search method called auto_arima. Here, we created a manual grid search to see if a manual grid search can outperform auto_arima. The downside of this method is that it's time consuming. It took almost 2 hours to perform the manual grid search.

The manual grid search results were stored in a dataframe so that we can easily sort the results by AIC results in ascending order. That would tell us what's the optimal model order to use.

The optimal parameters found by the manual grid search: ARIMA(2, 1, 2)x(0, 0, 2, 7)

This is different from auto_arima which recommended order=(0, 1, 0) seasonal_order=(0, 0, 0, 7). In this case, the non-seasonal parameters are different and there is seasonality detected.

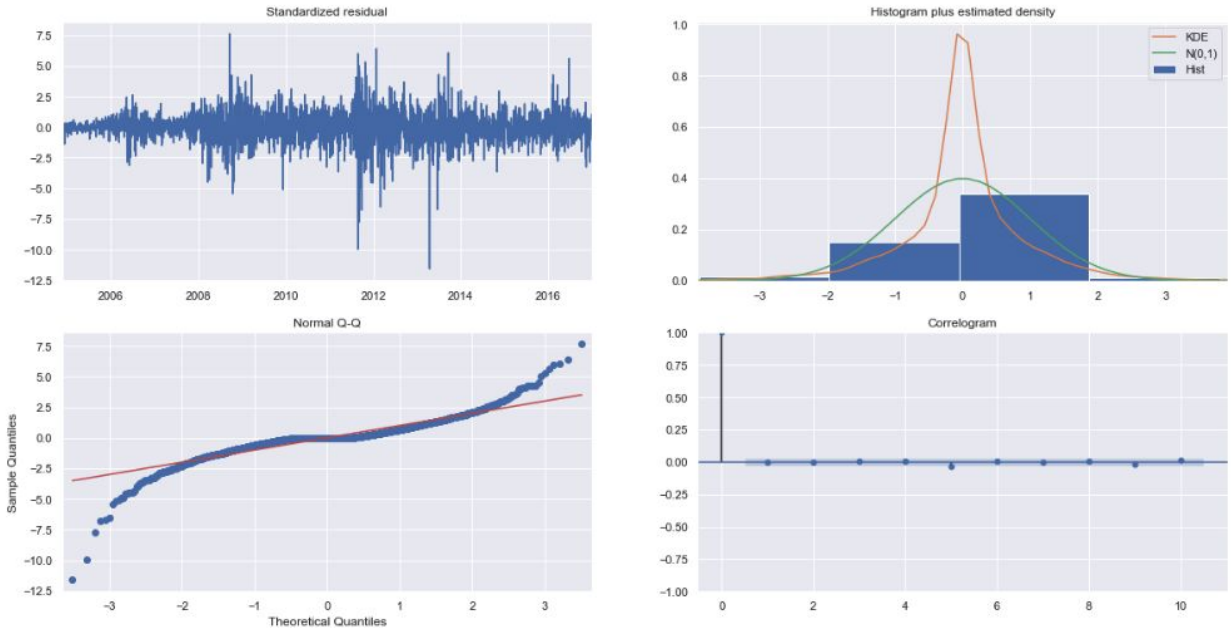
We created a model using the Best Fit ARIMA order created by the manual grid search and used it to predict the open price of gold ETF for the last 365 days of the training data (just like Method 1). We then compared the prediction against the real prices for the last 365 days if the training set.



Just like methods 1 and 2, the forecasts align very well with the true values for the last 365 days of the training data and it falls within the confidence intervals. Let's look at the model diagnostics results next.

Model Diagnostics Results

Here's the plot diagnostic results:



These 4 diagnostic plots shows no alarming difference from Method 1 and 2 diagnostic plots.

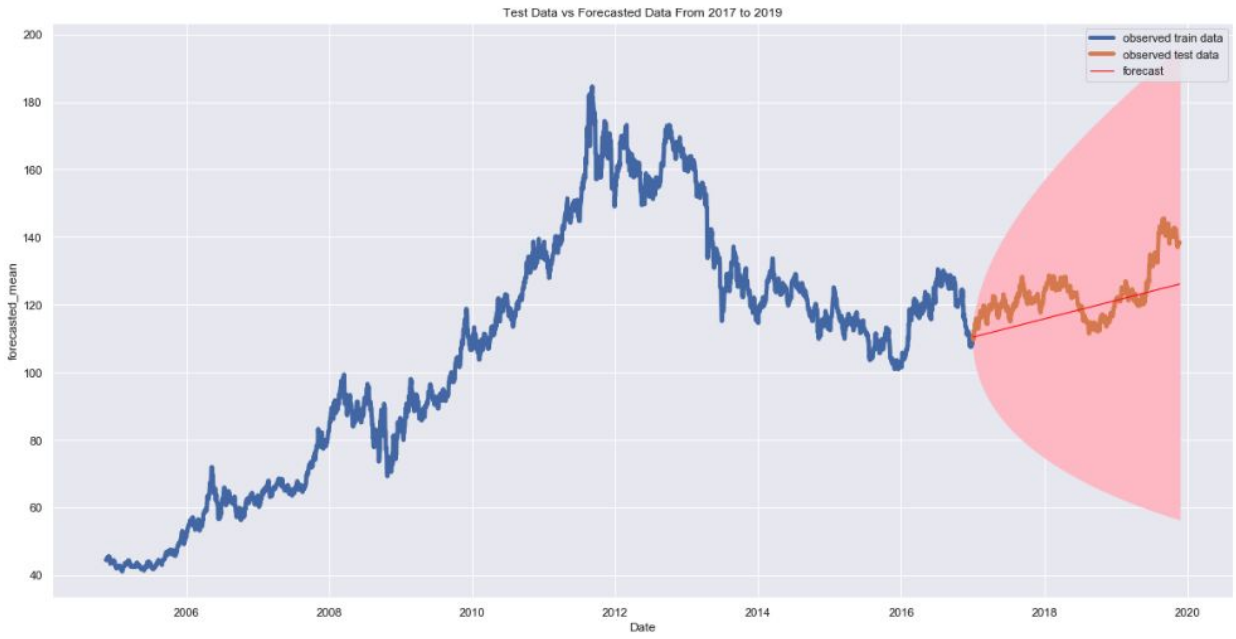
MAE, MSE and RMSE Scores

- **Mean Absolute Error (MAE):** 0.63
- **Mean Squared Error (MSE):** 1.04
- **Root Mean Squared Error (RMSE):** 1.02. Our model forecasted the average daily open price in the training set is within $1.02 of the real open prices.

Conclusion: There no difference in the results between Auto Arima and Manual Grid Search methods. MAE for Method 2 and 3 is slightly lower than Method 1. We are going to use the Auto Arima method for forecasting of future data.

Use Auto Arima (Method 2) to Forecast Opening Price and Compare With Test Data

Based on previous findings, we are going to use the fitted model using auto_arima to forecast the opening prices from 1/1/2017 - 11/22/2019 and compare the forecasted results against the test data set.



MAE, MSE and RMSE Scores

- **Mean Absolute Error (MAE):** 6.70
- **Mean Squared Error (MSE):** 64.36
- **Root Mean Squared Error (RMSE):** 8.02

Conclusion

By looking at the plot, the forecasted data showed an upward trend which is aligned with the test data. It correctly predicted that Gold Prices will go up from 2017 - 2019. It is also within the confidence interval. However, the interval is large. This shows that it’s hard to predict the prices of gold day-to-day but it's able to predict a general trend over time.

Modeling Using Facebook Prophet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data.

Like auto_arima, Prophet requires the dataset to be univariate. So we created a new data set with only date and open price and then split into training data (any data up till 2016) and test data (any data from 2017 onwards).

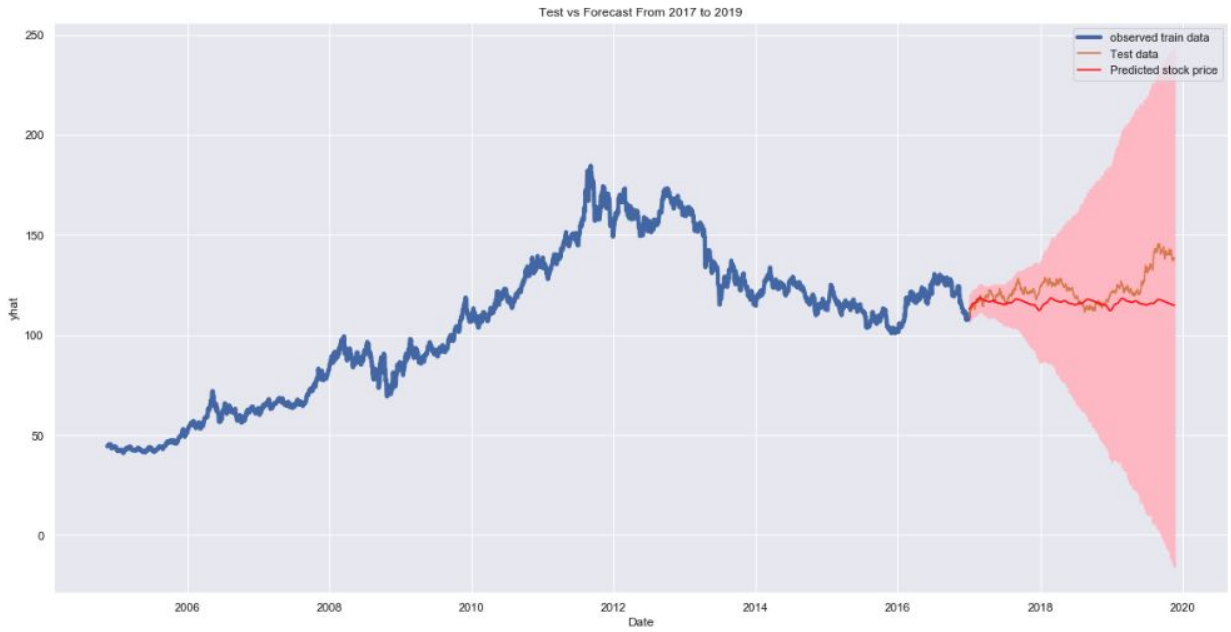
We then use the training data to create a model which will predict prices from 2017 to 2019 and compare the predicted prices with the test data.

Prophet also stores the predicted prices in a dataframe.

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	daily	...	weekly
4427	2017-01-01	116.281474	106.301631	119.217568	116.281474	116.281474	-3.432708	-3.432708	-3.432708	0.028582	...	-0.040755
4428	2017-01-02	116.281102	106.573841	119.803269	116.281102	116.281102	-3.211777	-3.211777	-3.211777	0.028582	...	0.048201
4429	2017-01-03	116.280730	106.704463	119.121396	116.280730	116.280730	-3.086714	-3.086714	-3.086714	0.028582	...	0.026828
4430	2017-01-04	116.280358	106.796941	119.216670	116.280358	116.280358	-2.924143	-2.924143	-2.924143	0.028582	...	0.030743
4431	2017-01-05	116.279986	107.251551	119.932681	116.279986	116.279986	-2.809356	-2.809356	-2.809356	0.028582	...	-0.022942
...	...	...	...	...	...	...	...	...	...	...	...	...
5478	2019-11-18	115.890425	-14.356712	240.095814	-12.700636	241.623191	-0.882751	-0.882751	-0.882751	0.028582	...	0.048201
5479	2019-11-19	115.890053	-14.611624	239.696444	-12.881180	241.745162	-0.910403	-0.910403	-0.910403	0.028582	...	0.026828
5480	2019-11-20	115.889681	-12.554075	243.246615	-13.061725	241.844236	-0.909258	-0.909258	-0.909258	0.028582	...	0.030743
5481	2019-11-21	115.889309	-12.468481	238.640514	-13.242270	241.913341	-0.962865	-0.962865	-0.962865	0.028582	...	-0.022942
5482	2019-11-22	115.888937	-15.776876	240.693316	-13.422815	241.980965	-0.951134	-0.951134	-0.951134	0.028582	...	-0.013313

1056 rows × 22 columns

Results



	Facebook Prophet	**Auto Arima**
MAE	7.71	6.70
MSE	108.88	64.36
RMSE	10.43	8.02

By looking at the above plot produced by Prophet, the forecasted data shows a flat line. The forecasted data is not aligned with the test data when the test data shows an upward trend.

Prophet does not seem to be as accurate as ARIMA model. The MAE, MSE and RMSE of Prophet is also higher than the results of ARIMA model.

Predicting The Price of Gold ETFs For The Next 2 Years

Auto Arima Method

Having validated our forecast results with our test data, we are going to perform an out of sample forecasting using the auto_arima method.. We have data of gold prices up till 11/22/2019. We are going to forecast the price of gold for the next 2 years from 11/23/2019 - 11/21/2021.

Based on previous best fit order model recommended by auto_arima, our model should have these parameters:

```
all_auto_arima_model = SARIMAX(arima_data,
                               seasonal=True,
                               order=(0,1,0),
                               seasonal_order=(0,0,0,7),
                               trend='c')
```

We use the above model to forecast 2 years out and also used the model to predict the price for the entire period from 11/18/2004 - 11/21/2021.

The red line is the prediction results from 11/18/2004 - 11/21/2021. You can see that predicted prices are very well aligned to the actual prices as shown in the black line.

The 2 year forecast does indicate an upward trend in gold price ETFs in the next 2 years.

Facebook Prophet Method

Though Prophet didn't produce predicted results as accurate as ARIMA, we are going to forecast out 2 years just to compare the results from the auto arima method above.

Based on the plot above, when you compare Prophet's prediction (thin blue line) as compared to the actual data (black line), the prediction does align very well with the actual data up till somewhere between Jul 2018 - Dec 2018. The actual prices dipped whereas Prophet predicted an increase in price. As you can see, it's not as well-aligned as the above Auto ARIMA method.

Prophet forecasted a **very slight** upward trend in gold price ETFs in the next 2 years.

Conclusion

Based on the 2 plots, ARIMA shows a better fit between actual data and predicted data.

The ARIMA model shows an upward trend in gold prices for the next 2 years, forecasting the price of gold to be at $150.80 by 11/21/2021. That's 9.28% increase from today's price of $138.

Prophet predicted the price of gold 2 years from now to be at $140.14, that's 1.55% increase. It shows a very slight upward trend.

Both models do show an upward trend in gold prices in the next 2 years. In conclusion, GLD ETF is a safe asset to buy now.

Disclaimer: This report is for educational purposes. It’s not meant to be any form of financial advice.