Stock Prediction using ARIMA and Facebook Prophet

Danny Huang

Introduction

The stock market is a complex and highly dynamic environment influenced by numerous factors, making accurate stock analysis difficult. However, stock predictors still have values as it can be used to make informed decisions and maximize returns. To reduce risk and increase the accuracy of predictions, the focus will be on stocks with a history of low volatility and variability, often found in ETFs (Exchange-Traded Funds).

For this project, historical stock price data will be collected from financial databases like Yahoo Finance. Python's Yahoo Finance Library will be used to obtain this data. This data will include daily open, close, high, low prices, and trading volumes for a range of stocks. As a retail trader I would like to see the actual vs predicted results of this project. I wouldn't be really using this for my trades but I think it'd be fun to see and compare.

Stakeholders

1. Retail traders
2. Investors
3. Financial Analyst
4. Researchers

Datasource

Dataset was retrieved from Yahoo Finance, using the yfinance library on python. 7 years of data were pulled for the SPLV: Invesco S&P 500 Low Volatility ETF . As mentioned, the stock market is a highly dynamic environment so to decrease risk a safe low risk ETF was chosen to test our models.

Data Wrangling

For this project, the select stock is SPLV: Invesco S&P 500 Low Volatility ETF.The dataset obtained from Yahoo Finance(yfinance) is formatted as followed:

Date	**Open**	**High**	**Low**	**Close**	**Adj. Close**	**Volume**
2024-07-18	67.2399	67.8799	67.1299	67.1699	67.0672	900500
2024-07-19	67.2600	67.4100	66.5899	66.6600	66.5580	1688100
2024-07-20	66.7200	67.0000	66.5400	66.9800	66.9800	884000

2024-07-21	66.9800	67.0400	66.5999	66.6399	66.6399	1015300

The data analysis and modeling only requires date and close prices, so a new DataFrame called ‘df_close’ is created containing just ‘Data’ and ‘Close’ columns. The index is then reset so that the date serves as the index.

Exploratory Data Analysis

SPLV Closing Price throughout the years

This graph illustrates seven years of closing prices of SPLV from mid-2017 to 2024. It highlights a rising trend in 2019, which continued until 2020 when the COVID-19 pandemic caused a significant decline in the global economy and negatively impacted the stock market. As the virus subsided and the market began to recover, the price has been on an upward trend. The orange line represents the monthly moving average. A monthly moving average smooths out fluctuations in data to reveal trends over time. This helps to see the overall trend by filtering out short-term volatility and providing a clearer picture of the stock's performance over a longer period.(Figure 1)

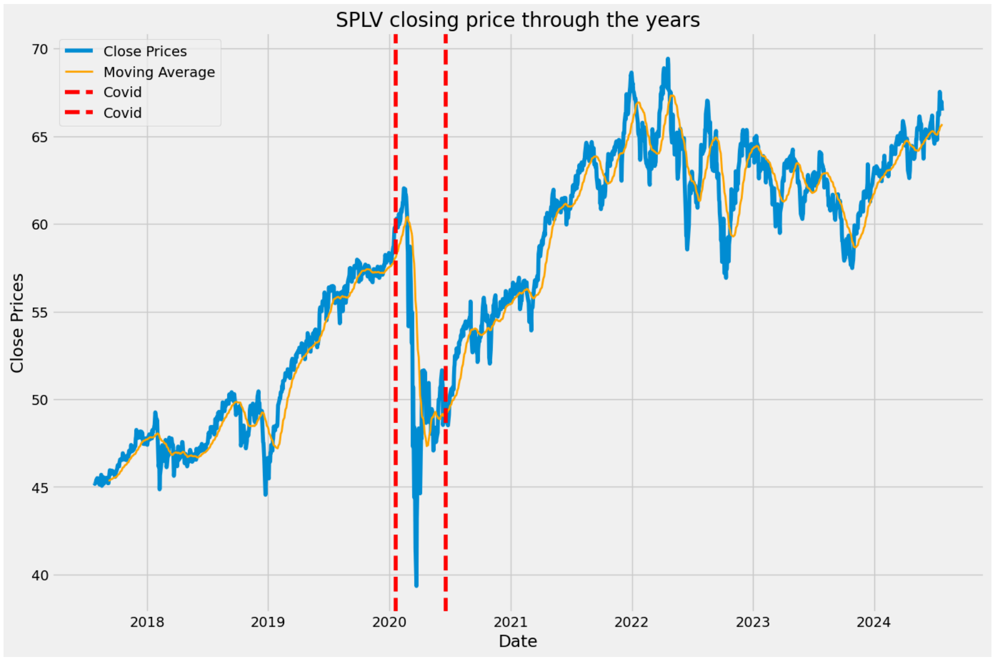


Figure 1. SPLV Closing Price

Kernel Density Estimate: Distribution of Closing Prices

This KDE graph illustrates the distribution of prices within our dataset. There is a higher concentration of data points around $63, with a secondary concentration around $48. (Figure 2.)

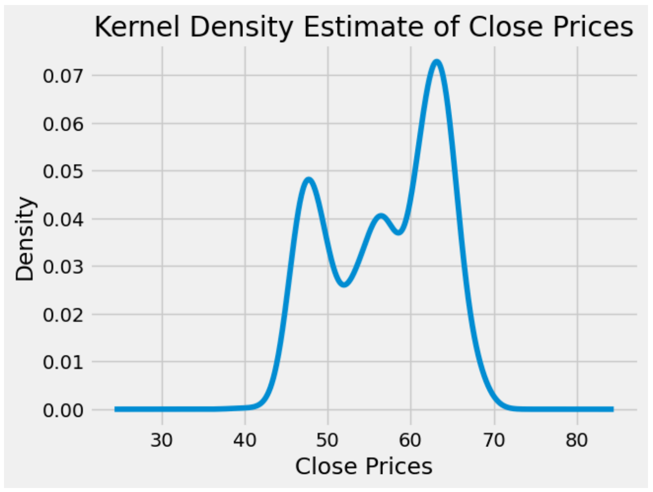


Figure 2. KDE of Close Prices

Autocorrelation plot

The initial values of the blue line are close to 1, indicating that there is a strong positive autocorrelation at small lags. This suggests that closing prices are positively correlated with their previous values(lags). As the lag increases, autocorrelation decreases indicating that the influence diminishes over time. At lag 700, it crosses 0 and thus the autocorrelation becomes statistically insignificant. The strong positive autocorrelation at small lags suggests that the series can predict ahead of time on a short term basis. (Figure 3.)

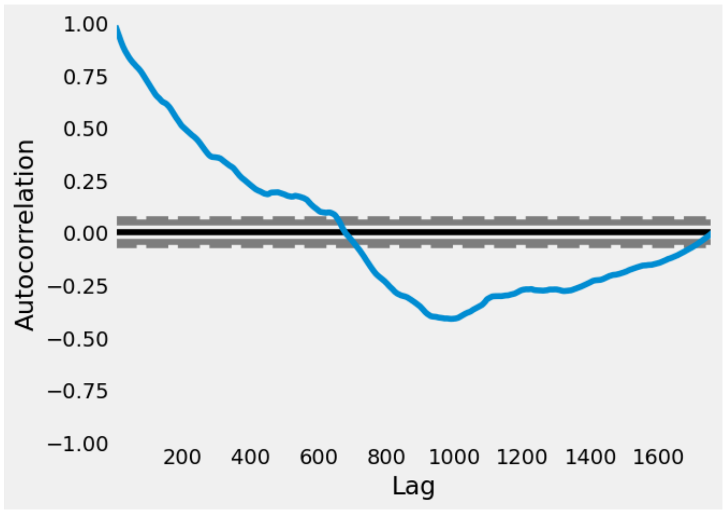


Figure 3. Autocorrelation plot

Partial Autocorrelation Plot

In the partial autocorrelation plot, after the first lag coefficient drops and remains close to zero, suggesting that there is little to no partial autocorrelation at higher lags after accounting for the lag 1 effect. In other words, this time series has a strong short-term dependency and the current value is highly correlated with the previous value.(Figure 4.)

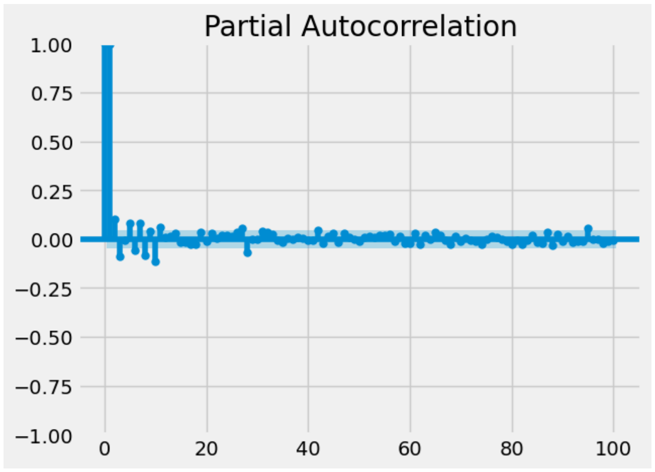


Figure 4. Partial Autocorrelation Plot

Seasonal Decompose

As expected, seasonality exists in a stock time series. In order to perform a time series analysis, we may need to separate seasonality and trend from our series. It is also worth noting that while there is a slight rising trend in the data, it is not significant enough to influence the model's performance.(Figure 5.)

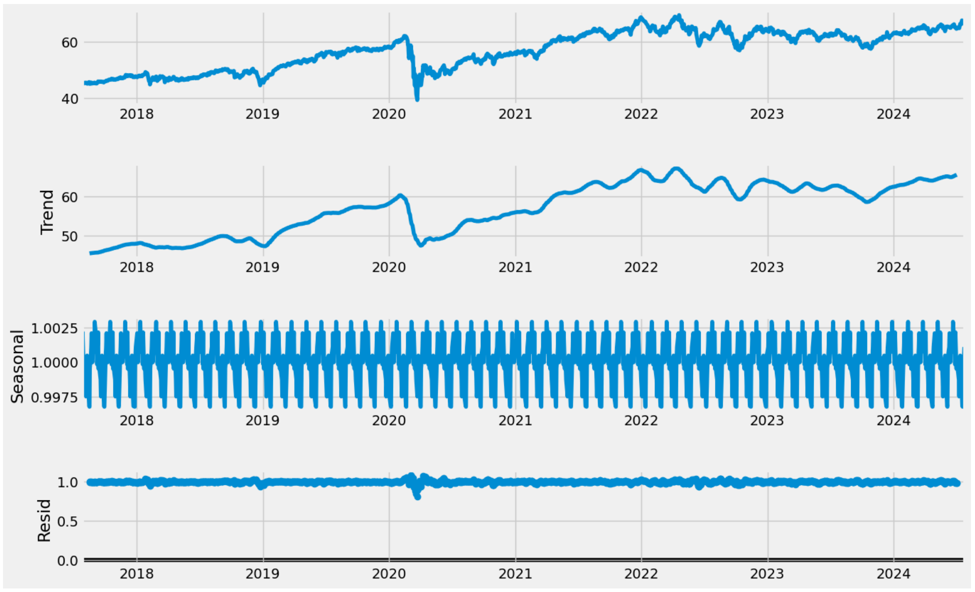
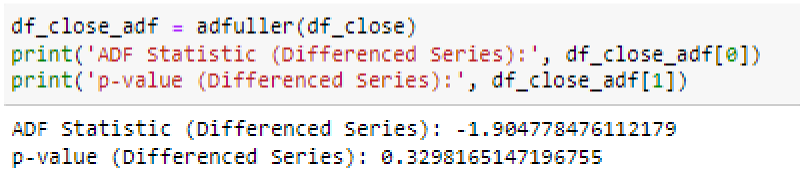


Figure 5. Seasonal Decompose plot

Augmented Dickey-Fuller (ADF) Test

The ADF test can be used to determine the presence of a unit root in a time series. If a unit root is present, the series is not stationary. The test revealed that Trends and Seasonal effects make our time series non-stationary. (Figure 6.)



```
df_close_adf = adfuller(df_close)
print('ADF Statistic (Differenced Series):', df_close_adf[0])
print('p-value (Differenced Series):', df_close_adf[1])
```

```
ADF Statistic (Differenced Series): -1.904778476112179
p-value (Differenced Series): 0.3298165147196755
```

Figure 6. ADF Code + Results

p-value is more than 0.05 so we fail to reject the Null Hypothesis (The series has a unit root (value of a =1), in other words our time series is non-stationary.

Preprocessing

The ADF test revealed that our time series was non-stationary, indicating a need for transformation before modeling. This transformation is essential because in many time series models, such as ARIMA, the assumption is that past patterns in the data can help predict future values. Stationarity ensures that these patterns are consistent over time. If the statistical properties of the series are changing, the model's predictions may become less reliable.

Train/Test Split

The data was divided into training and testing datasets with an 80/20 split. (Figure 7.)

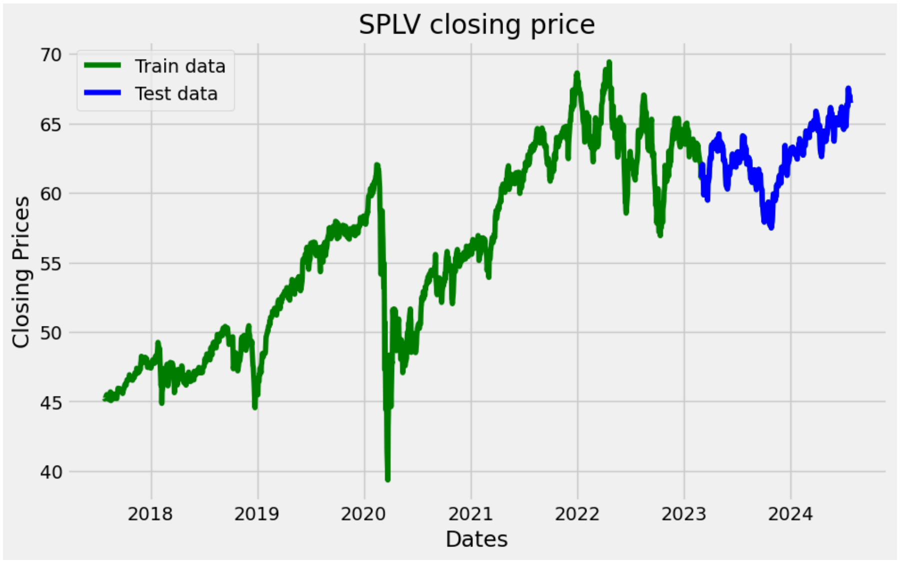
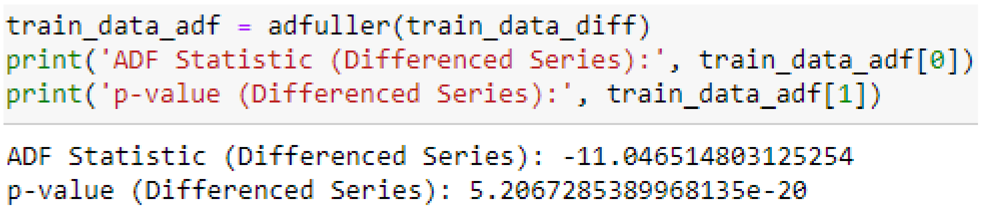


Figure 7. Train and Test dataset

Transformation

As previously mentioned, our data is not stationary. To make the series stationary, a differencing needs to be applied. Differencing which is subtracting previous values from the current value, this is to remove stationarity. (Figure 8.)



```
train_data_adf = adfuller(train_data_diff)
print('ADF Statistic (Differenced Series):', train_data_adf[0])
print('p-value (Differenced Series):', train_data_adf[1])
```

```
ADF Statistic (Differenced Series): -11.046514803125254
p-value (Differenced Series): 5.2067285389968135e-20
```

Figure 8. Differencing ADF Code and Results

Now that our p-value is less than 0.05, the data is considered stationary. This can be visualized to observe the effects of the transformation. (Figure 9)

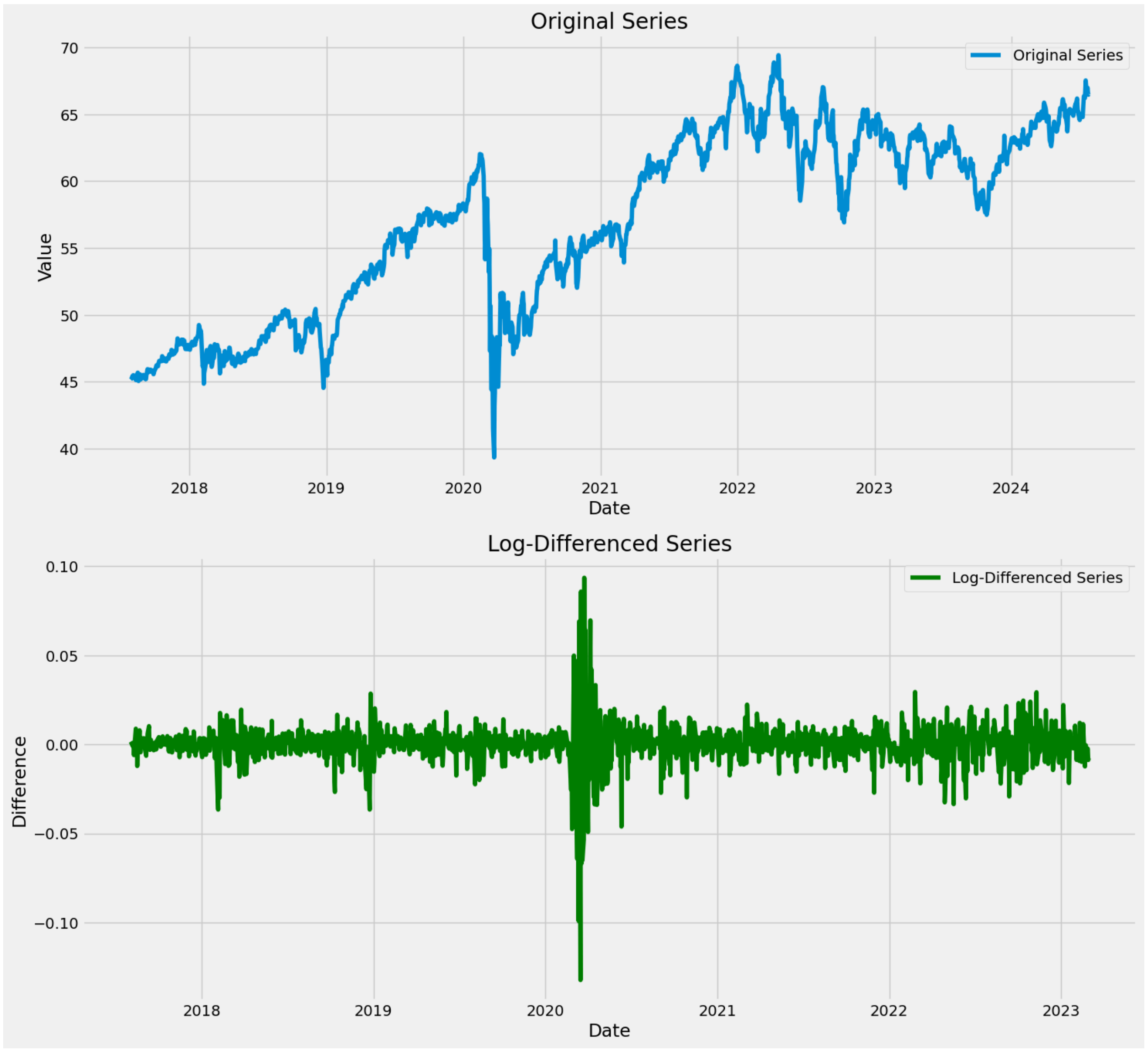


Figure 9. Series transformation graphs

Modeling

ARIMA

ARIMA is a popular statistical method used for analyzing and forecasting time series data. It combines three main components: AutoRegressive (AR), Integrated (I), and Moving Average (MA). The Pmdarima library's ‘auto_arima’ function was used to determine the AR (p), I (d), and MA (q) parameters. The best ARIMA model order (p,d,q) was determined to be (1,0,5). After fitting the model, we ran our model. To obtain the final results, we had to reverse the differencing. (Figure 10.)

Arima Results

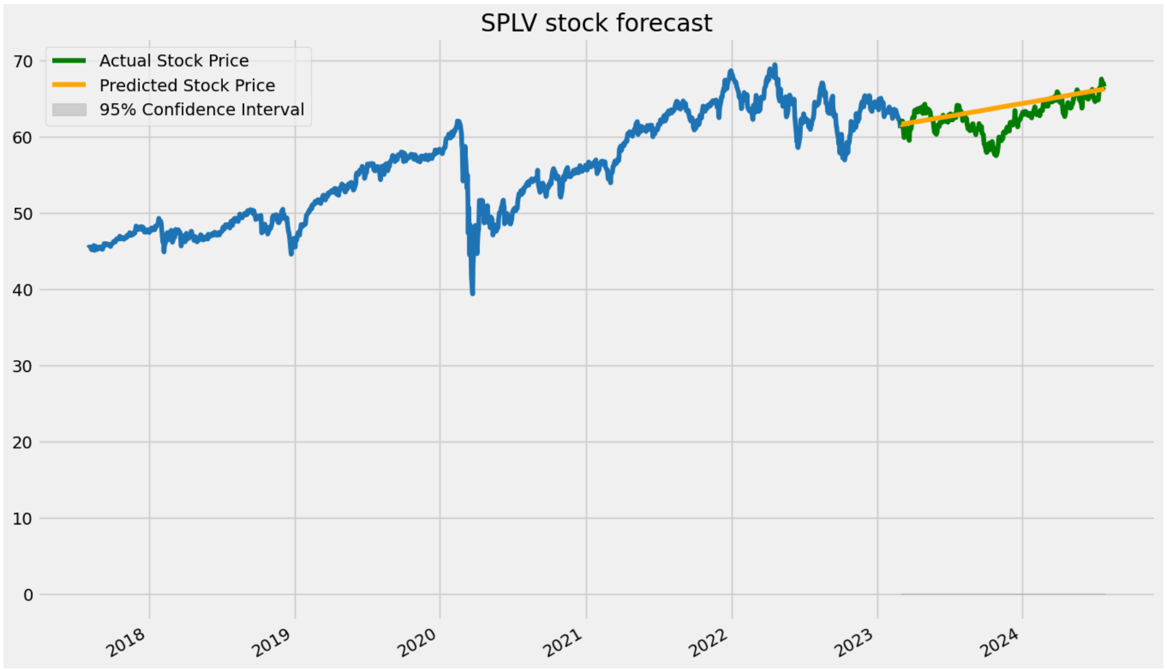
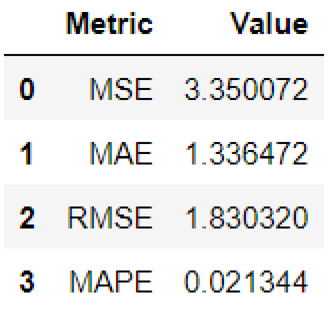


Figure 10. Arima results with differencing

Metrics

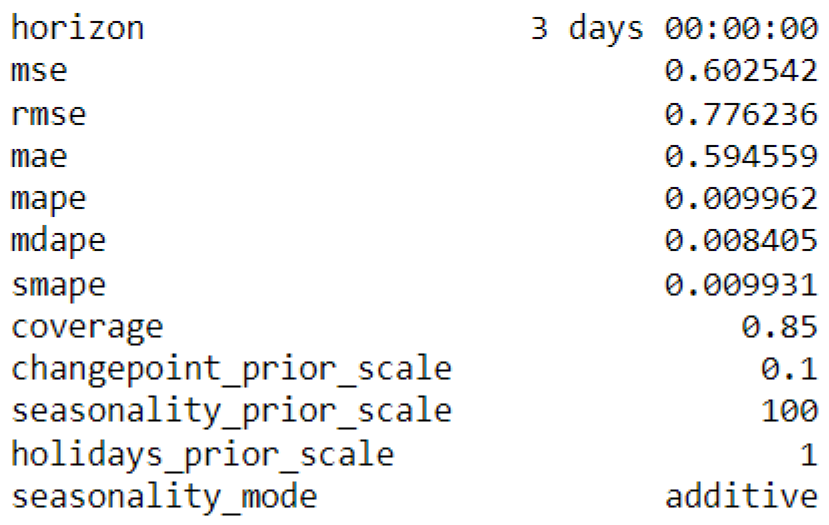


	Metric	Value
0	MSE	3.350072
1	MAE	1.336472
2	RMSE	1.830320
3	MAPE	0.021344

Facebook Prophet

Facebook Prophet is an open-source tool developed for time series forecasting. It is designed to handle data with daily, weekly, and yearly seasonal patterns, as well as holiday effects and missing data. Prophet uses an additive model to fit non-linear trends with seasonal effects and is particularly effective for time series with strong seasonal patterns and multiple seasons of historical data. It is robust to missing data, trend shifts, and outliers.

To find the best parameters, the model needs to be reiterated using different parameter combinations. Functions will automate the process of evaluating these combinations and selecting the optimal ones for our model. After completing the iterations, RMSE is chosen as the target score. The best parameters identified are:



```
horizon                     3 days 00:00:00
mse                                0.602542
rmse                               0.776236
mae                                0.594559
mape                               0.009962
mdape                              0.008405
smape                              0.009931
coverage                               0.85
changepoint_prior_scale                 0.1
seasonality_prior_scale                 100
holidays_prior_scale                      1
seasonality_mode                   additive
```

The best RMSE score was chosen as the evaluation metric because RMSE balances the units of the original data, making it easier to interpret. Additionally, RMSE penalizes larger errors more heavily, which is particularly important in stock prediction where large deviations can have significant financial implications.

Facebook Prophet Results

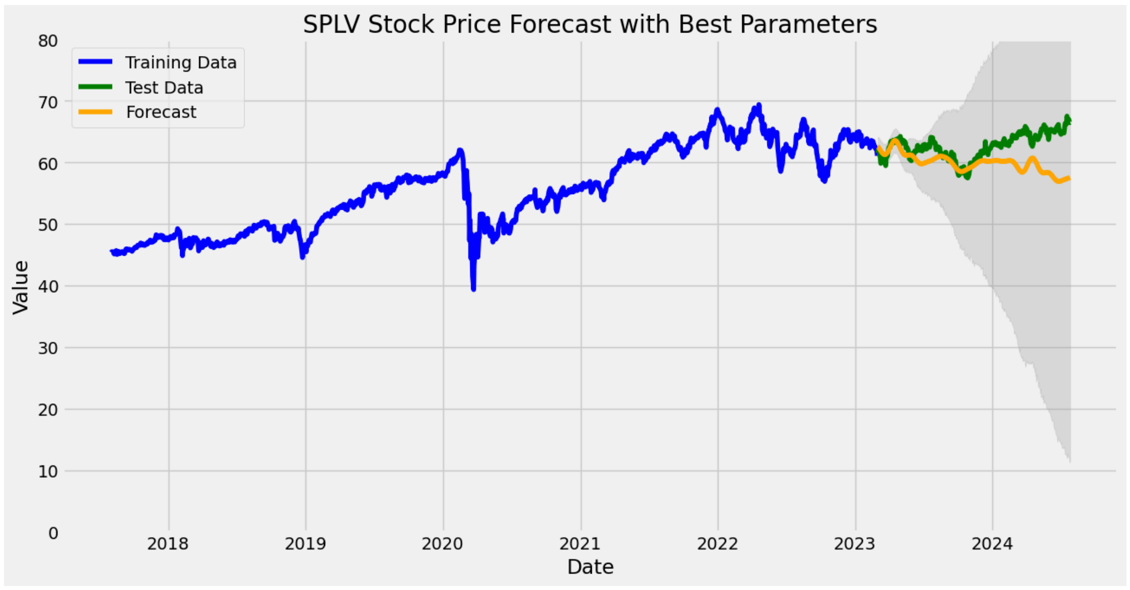
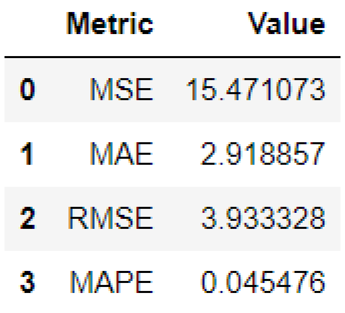


Figure 11. Prophet results

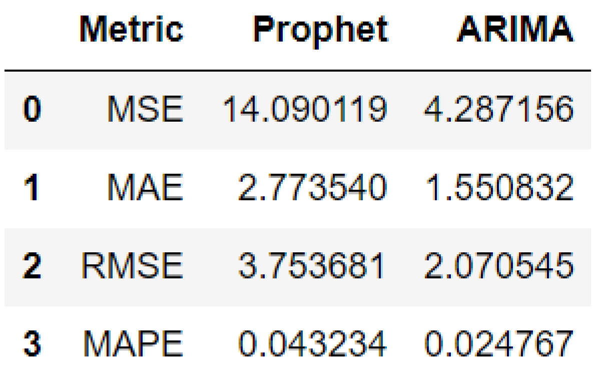
Metrics



	Metric	Value
0	MSE	15.471073
1	MAE	2.918857
2	RMSE	3.933328
3	MAPE	0.045476

Model Evaluation

Each error metric—MSE, MAE, RMSE, and MAPE—offers distinct advantages and limitations. MSE and RMSE are beneficial for emphasizing larger errors, with RMSE providing easier interpretability. MAE offers a straightforward approach less sensitive to outliers, while MAPE provides percentage-based insights, enhancing communication of results despite its sensitivity to small values.



	Metric	Prophet	ARIMA
0	MSE	14.090119	4.287156
1	MAE	2.773540	1.550832
2	RMSE	3.753681	2.070545
3	MAPE	0.043234	0.024767

Mean Squared Error (MSE)

SARIMA has a lower MSE, indicating it has better overall accuracy in capturing the variance in the data.

Mean Absolute Error (MAE)

SARIMA has a lower MAE, indicating it has better accuracy in terms of absolute differences between the predicted and actual values.

Root Mean Squared Error (RMSE)

SARIMA has a lower RMSE, indicating it has better accuracy and lower prediction error magnitude.

Mean Absolute Percentage Error (MAPE)

SARIMA has a lower MAPE, indicating it has better accuracy in terms of percentage error.

Forecast

ARIMA will be used to forecast future stock prices since it performed the best in our evaluations and successfully captured the observed upward trend. (Figure 12.)

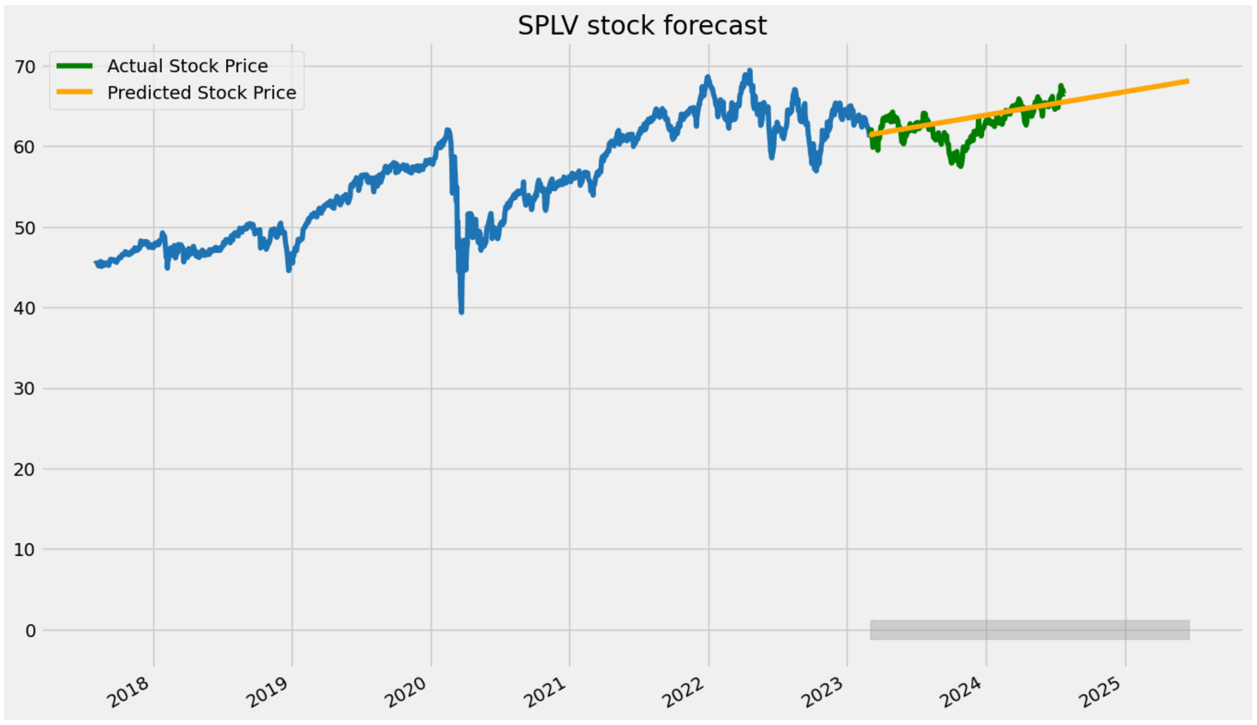


Figure 12. Arima SPLV Forecast

Conclusion

This project focused on predicting future stock prices using ARIMA and Facebook Prophet, with a particular emphasis on ETFs. Given the complex and dynamic nature of the stock market, accurate stock analysis can be challenging, yet valuable for making informed decisions and maximizing returns. By targeting stocks with low volatility and variability, often found in ETFs, we chose SPLV: Invesco S&P 500 Low Volatility

The dataset, spanning seven years, was obtained from Yahoo Finance using the yfinance library. The data included daily open, close, high, low prices, and trading volumes. For analysis, only the date and close prices were retained, forming a new DataFrame. Exploratory data analysis revealed significant trends and patterns, such as the impact of the COVID-19 pandemic on stock prices and the subsequent recovery.

Ultimately, the ARIMA model demonstrated superior performance, accurately capturing the observed upward trend and achieving better accuracy across the chosen metrics. This project's findings underscore the importance of selecting appropriate models and evaluation metrics tailored to the specific characteristics of the stock market data.

In summary, leveraging ARIMA for stock prediction proved effective for SPLV, providing a robust forecasting tool that can inform future trading decisions. The project highlighted the value of thorough exploratory data analysis, appropriate data transformations, and careful model evaluation in developing reliable stock prediction models.