Katherine Davis

Project 3-UofM FinTECH Bootcamp

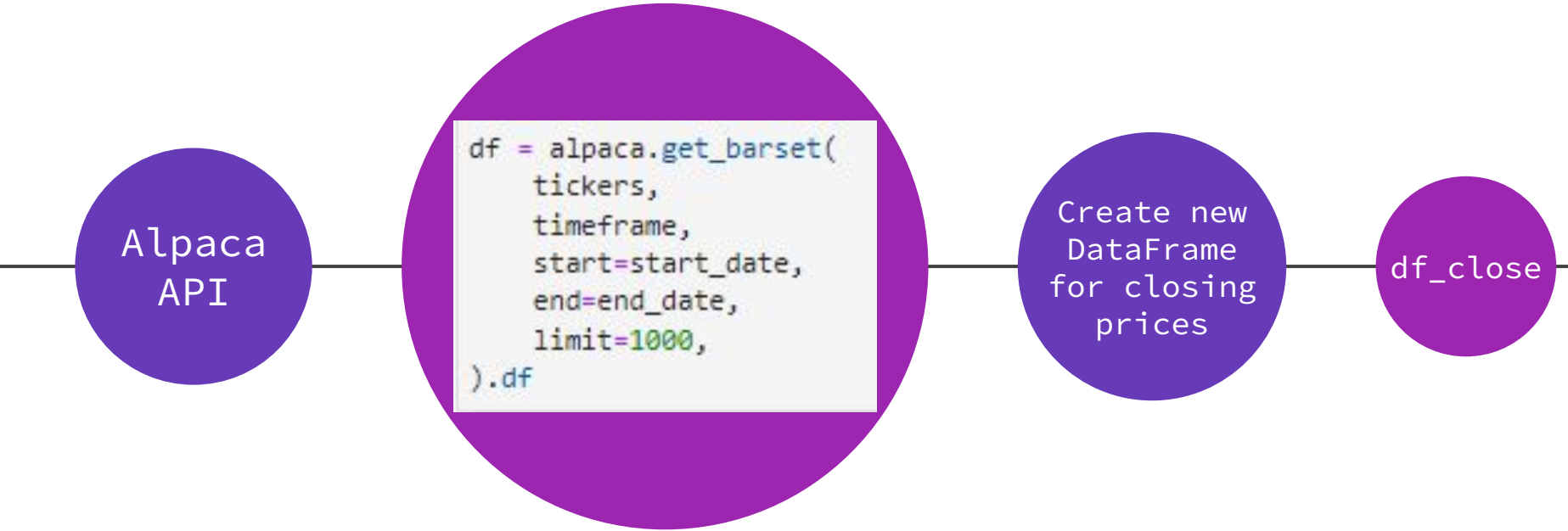
Executive Summary

Attempted to create an algorithmic trading model to evaluate the success of the DMAC trading strategy. This was applied to MGM Resorts International stock.

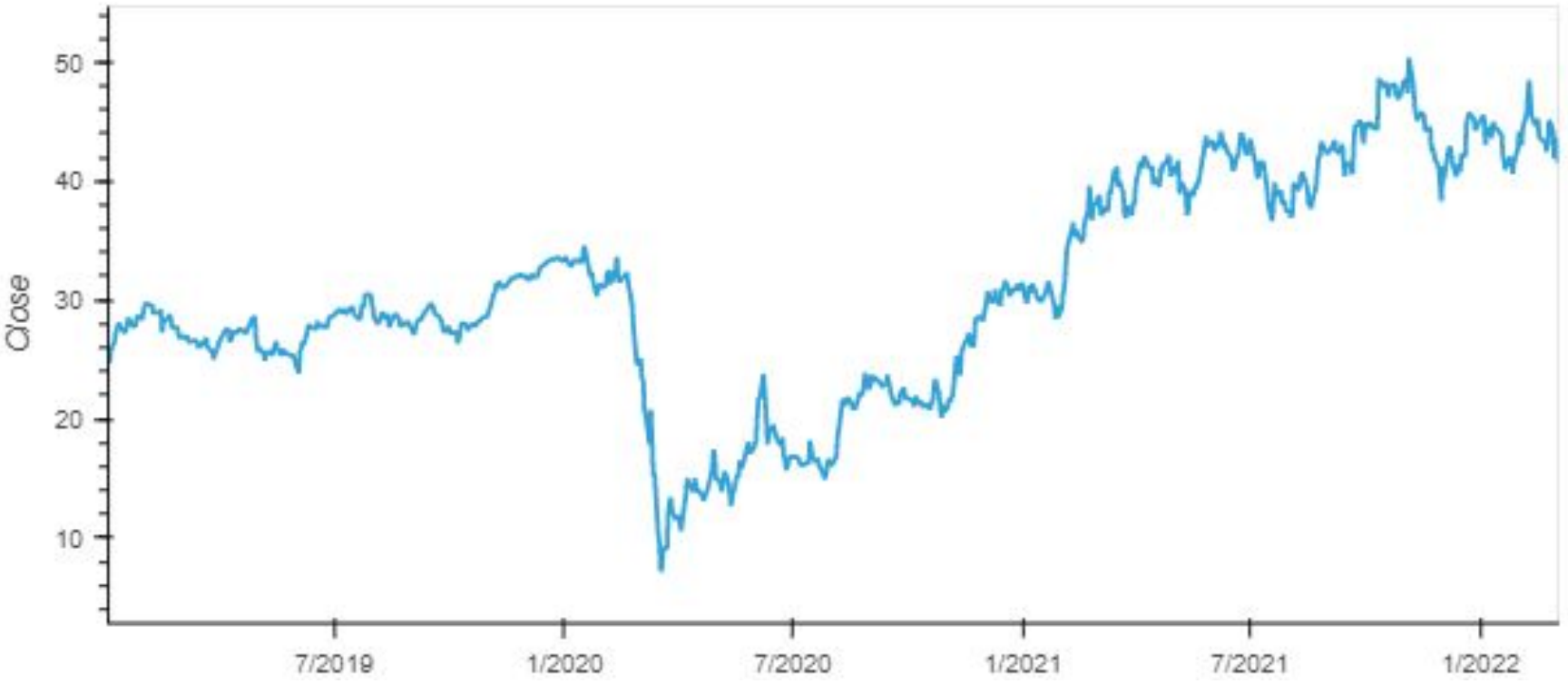
I evaluated the daily close prices of the date range from January 1st, 2019 through March 4th, 2022.

The purpose of the study was to evaluate a company that has implemented a sportsbook app with which BetMGM is a part of the MGM Resorts International.

Data Pull & DataFrame Creation



MGM Close Price- 2019 through Current Day



Strategy 1

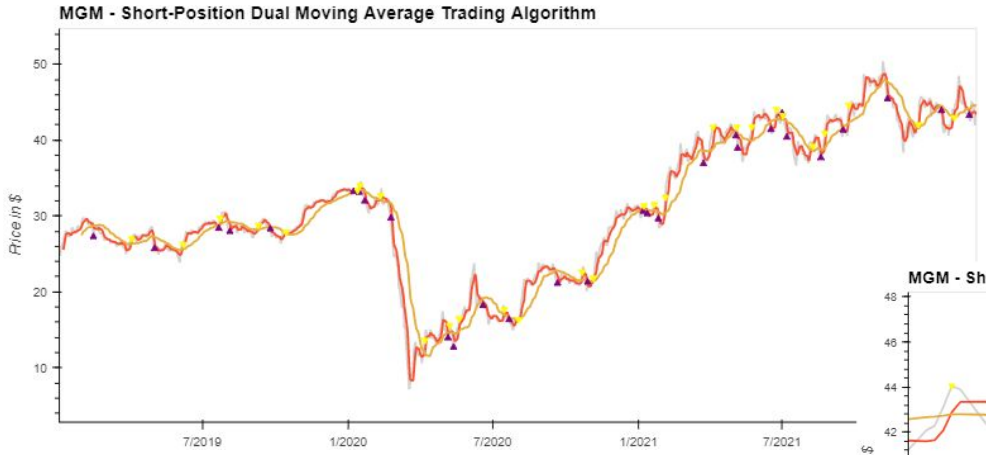
Set the Windows and Signals

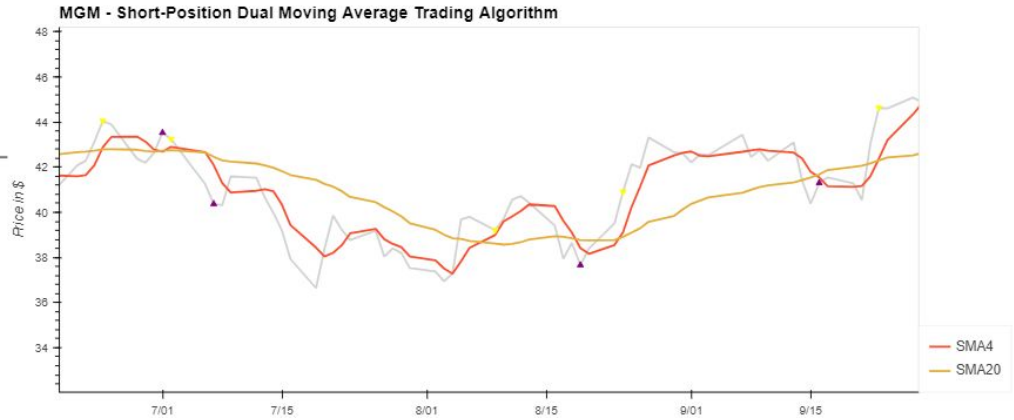
Used a 4 day SMA window for the short window and a 20 day SMA for the long window.

This was to assist with a visualization of when to buy and sell the stock.


```
# Set the short_window and long window variables
short_window = 4
long_window = 20

# Generate the short and long moving averages
df_close['SMA4'] = df_close['Close'].rolling(window=short_window).mean()
df_close['SMA20'] = df_close['Close'].rolling(window=long_window).mean()
```

```
# Initialize the new Signal column to hold the trading signal
df_close['Signal'] = 0.0
# Generate the trading signal 0 or 1,
df_close["Signal"][short_window:] = np.where(
    df_close["SMA4"][short_window:] < df_close["SMA20"][short_window:], 1.0, 0.0
)

# Calculate the points in time at which a position should be taken, 1 or -1
df_close['Entry/Exit'] = df_close['Signal'].diff()
```



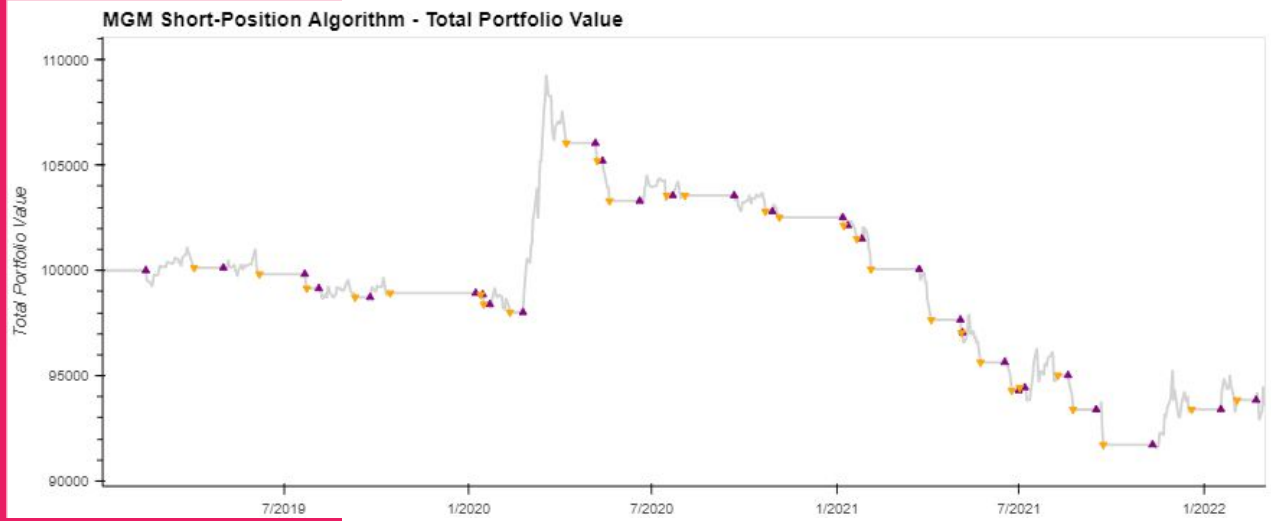


Combine the SMA and Signals

Combining the SMA and entry exit signals allows for the visualization of the DMAC technique to be utilized.

Dual Moving Average Crossover

The DMAC strategy signals buy when the short-term moving average crosses below the long-term moving average



Portfolio Value

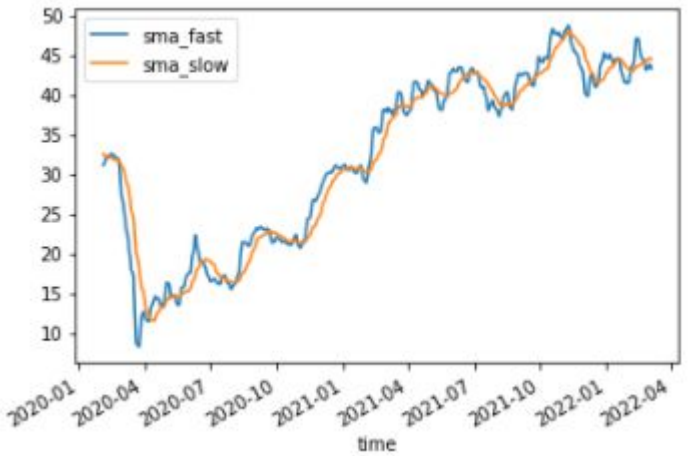
Using the DMAC technique would not work with this certain stock

Strategy 2

```python
# Create a new column calling the signal value to zero.
df_close["signal"] = 0.0

# Create the signal to buy
df_close.loc[(df_close["actual_returns"] >= 0), "signal"] = 1

# Create the signal to sell
df_close.loc[(df_close["actual_returns"] < 0), "signal"] = -1

# Copy the new signal column to a new Series called y.
y = df_close["signal"].copy()
y
```

```
time
2019-01-31 00:00:00-05:00    1.0
2019-02-01 00:00:00-05:00    1.0
2019-02-04 00:00:00-05:00   -1.0
2019-02-05 00:00:00-05:00   -1.0
2019-02-06 00:00:00-05:00    1.0
                            ...
2022-02-28 00:00:00-05:00   -1.0
2022-03-01 00:00:00-05:00   -1.0
2022-03-02 00:00:00-05:00    1.0
2022-03-03 00:00:00-05:00   -1.0
2022-03-04 00:00:00-05:00   -1.0
Name: signal, Length: 780, dtype: float64
```

- Signaled BUY when returns were greater than 0
- Signaled SELL when returns were less than 0
- Plotted the training data with the fast and slow SMA



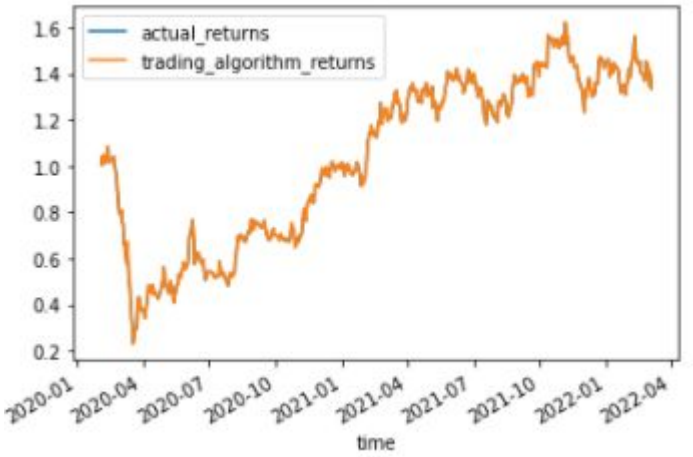
Results

Inconclusive

From the code I generated, this strategy produced inconclusive results.

```
# Create a new empty predictions DataFrame
predictions_df = pd.DataFrame(index=X_test.index)
predictions_df["predicted_signal"] = testing_signal_predictions
predictions_df["actual_returns"] = df_close["actual_returns"]
predictions_df["trading_algorithm_returns"] = (
    predictions_df["actual_returns"] * predictions_df["predicted_signal"]
)
predictions_df.head()
```

time	predicted_signal	actual_returns	trading_algorithm_returns
2020-02-03 00:00:00-05:00	1.0	0.004507	0.004507
2020-02-04 00:00:00-05:00	1.0	0.033654	0.033654
2020-02-05 00:00:00-05:00	1.0	0.003411	0.003411
2020-02-06 00:00:00-05:00	1.0	0.003708	0.003708
2020-02-07 00:00:00-05:00	1.0	-0.034483	-0.034483



Activity Results

In comparison to the class Activity code results you can see my code resulted in inconclusive results. More digging and code variations needed.