COMP 642 Final Project - Part 2

Data: Prepare a data analysis report that includes:

- Data definition for each parameter including measurement units
- Plan for missing data for each parameter (if needed)
- Plan for additional parameters or data (if needed)
- Any transformations necessary
- Plan for separating
- Visualization of data if possible

Data Definitions

Summary

We will explore some features and data which will be considered for use in our project.

It may be best to redefine our goal. We are attempting to determine, based on daily or weekly market close prices of TQQQ, whether the following day or week will yield a gain or a loss. Depending on that, we will decide to long (buy) or short (sell) TQQQ for the upcoming week/day. There are many factors that could influence this. TQQQ is a 3x leveraged ETF of the Nasdaq Composite. Major movements in large cap stocks, mostly centered toward technology stocks, are the major influencers of the price movements in this product. These large cap companies are also heavily influenced by outside economic factors that are not entirely dependent on company performance/financials. Also, there are technical techniques lightly used in the investment industry that may allow more insight into possible future movements of secuirty prices. These techniques are related to historical price movements and trade volumes. Armed with this observation, we can make some general assumptions for data required for our analysis and our model.

---

1. Macro market factors that influence stock prices
    - Interest rates
    - Inflation

- Currency
- Market Volitility (Fear Gauge)
- Commodities

2. Technical Indicators

- 35 day price exponential moving average
- 200 day price moving average
- Volume
- Moving Average Convergence Divergence (MACD)
- Relative Strength Index (RSI)

Macro Market Data

We will be using some readily available ETF data which tracks each of these macro economic features.

TQQQ ETF* = Nasdaq 3x leveraged ETF (stock price & daily return: dtype - float for both features)

Interest Rates = ProShares Short 20+ Year Treasury ETF: Ticker TBF (stock price & daily return: dtype - float)

Inflation = iShares TIPS Bond ETF: Ticker TIP (stock price & fractional return: dtype - float)

Currency = Invesco DB US Dollar Index Bullish Fund: Ticker UUP (stock price & fractional return: dtype - float for both features)

Market Volitility = ProShares VIX Short-Term Futures ETF: Ticker VIXY (stock price & fractional return: dtype - float for both features)

Commodities = United States Oil Fund: Ticker USO (stock price & fractional return: dtype - float for both features)

Commodities = SPDR Gold Shares: Ticker GLD (stock price & fractional return: dtype - float for both features)

Technical Indicator Data

We will be conducting feature engineering of the daily close price data in order to construct features that make 4 of the indicators;

35 day price exponential moving average: ema_35 (average price: dtype - float)

200 day simple moving average: sma_200 (average price: dtype - float)

Moving Average Convergence Divergence (MACD): macd (price fraction: dtype - float)

Relative Strength Index (RSI): rsi (average price: dtype - float)

We will also track purchase volume (vol) and we may need to convert the price data into percentages gained or loss from the previous day close depending on how we would like to visualize the data and/or make predictions.

Finally we will feature engineer our dependent variable. This depending variable will be the next day/week closing price. We will also create a binary categorical variable to indicate long or short depending on if the next day/week closing price is greater than the current day/week closing price

In [ ]:
```
# import required packages
# !pip3 install yfinance
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Upload Data

We will start by uploading the required data as defined by the above information.

In [ ]:
```
# fetching daily adjusted close price, volume, and percent changes
tqqq_price = yf.download('TQQQ', start='2010-12-31', end='2024-02-29', group_by='ticker')[['Adj Close', 'Volume']]
tqqq_price['tqqq_ret'] = tqqq_price['Adj Close'].pct_change()
tqqq_price.rename(columns={'Adj Close': 'tqqq_close', 'Volume': 'tqqq_volume'}, inplace=True)

tbf_price = pd.DataFrame(yf.download('TBF', start='2010-12-31', end='2024-02-29')[['Adj Close', 'Volume']])
tbf_price['tbf_ret'] = tbf_price['Adj Close'].pct_change()
tbf_price.rename(columns={'Adj Close': 'tbf_close', 'Volume': 'tbf_volume'}, inplace=True)

tip_price = pd.DataFrame(yf.download('TIP', start='2010-12-31', end='2024-02-29')[['Adj Close', 'Volume']])
tip_price['tip_ret'] = tip_price['Adj Close'].pct_change()
tip_price.rename(columns={'Adj Close': 'tip_close', 'Volume': 'tip_volume'}, inplace=True)

uup_price = pd.DataFrame(yf.download('UUP', start='2010-12-31', end='2024-02-29')[['Adj Close', 'Volume']])
uup_price['uup_ret'] = uup_price['Adj Close'].pct_change()
uup_price.rename(columns={'Adj Close': 'uup_close', 'Volume': 'uup_volume'}, inplace=True)

vixy_price = pd.DataFrame(yf.download('VIXY', start='2010-12-31', end='2024-02-29')[['Adj Close', 'Volume']])
vixy_price['vixy_ret'] = vixy_price['Adj Close'].pct_change()
vixy_price.rename(columns={'Adj Close': 'vixy_close', 'Volume': 'vixy_volume'}, inplace=True)

uso_price = pd.DataFrame(yf.download('USO', start='2010-12-31', end='2024-02-29')[['Adj Close', 'Volume']])
uso_price['uso_ret'] = uso_price['Adj Close'].pct_change()
uso_price.rename(columns={'Adj Close': 'uso_close', 'Volume': 'uso_volume'}, inplace=True)

gld_price = pd.DataFrame(yf.download('GLD', start='2010-12-31', end='2024-02-29')[['Adj Close', 'Volume']])
```

```
gld_price['gld_ret'] = gld_price['Adj Close'].pct_change()
gld_price.rename(columns={'Adj Close': 'gld_close', 'Volume': 'gld_volume'}, inplace=True)
```

```
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
[**********************100%%**********************]  1 of 1 completed
```

In [ ]:
```
# joining all data into a single dataframe according to the index (date)
df = pd.merge(tqqq_price, tbf_price, left_index=True, right_index=True)
df = pd.merge(df, tip_price, left_index=True, right_index=True)
df = pd.merge(df, uup_price, left_index=True, right_index=True)
df = pd.merge(df, vixy_price, left_index=True, right_index=True)
df = pd.merge(df, uso_price, left_index=True, right_index=True)
df = pd.merge(df, gld_price, left_index=True, right_index=True)
df.reset_index(inplace=True)
df.head()
```

Out[ ]:

	Date	tqqq_close	tqqq_volume	tqqq_ret	tbf_close	tbf_volume	tbf_ret	tip_close	tip_volume	tip_ret	...	uup_ret	vixy_close	vi
0	2011-01-04	0.784261	65760000	-0.001553	40.865879	272600	-0.000226	77.868210	1049900	0.002521	...	0.002629	158460.0	
1	2011-01-05	0.803726	61440000	0.024819	41.751659	402500	0.021675	77.570801	869400	-0.003819	...	0.010053	155100.0	
2	2011-01-06	0.810993	52262400	0.009042	41.576344	463800	-0.004199	77.795692	827900	0.002899	...	0.007789	155760.0	
3	2011-01-07	0.809113	76204800	-0.002319	41.373363	599800	-0.004882	78.042297	979300	0.003170	...	0.003006	156080.0	
4	2011-01-10	0.818210	43027200	0.011243	41.133461	654800	-0.005798	78.267204	639600	0.002882	...	-0.003425	155760.0	

5 rows × 22 columns

In [ ]:
```
# reviewing feature data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3309 entries, 0 to 3308
Data columns (total 22 columns):
 #   Column       Non-Null Count  Dtype         
---  ------       --------------  -----         
 0   Date         3309 non-null   datetime64[ns]
 1   tqqq_close   3309 non-null   float64       
 2   tqqq_volume  3309 non-null   int64         
 3   tqqq_ret     3309 non-null   float64       
 4   tbf_close    3309 non-null   float64       
 5   tbf_volume   3309 non-null   int64         
 6   tbf_ret      3309 non-null   float64       
 7   tip_close    3309 non-null   float64       
 8   tip_volume   3309 non-null   int64         
 9   tip_ret      3309 non-null   float64       
 10  uup_close    3309 non-null   float64       
 11  uup_volume   3309 non-null   int64         
 12  uup_ret      3309 non-null   float64       
 13  vixy_close   3309 non-null   float64       
 14  vixy_volume  3309 non-null   int64         
 15  vixy_ret     3308 non-null   float64       
 16  uso_close    3309 non-null   float64       
 17  uso_volume   3309 non-null   int64         
 18  uso_ret      3309 non-null   float64       
 19  gld_close    3309 non-null   float64       
 20  gld_volume   3309 non-null   int64         
 21  gld_ret      3309 non-null   float64       
dtypes: datetime64[ns](1), float64(14), int64(7)
memory usage: 568.9 KB
```

Observations

All data types are accurate and do not need to be changed moving forward.

In [ ]:
```
# reviewing summary statistics of our data
df.describe()
```

Out[ ]:

	Date	tqqq_close	tqqq_volume	tqqq_ret	tbf_close	tbf_volume	tbf_ret	tip_close	tip_volume	
count	3309	3309.000000	3.309000e+03	3309.000000	3309.000000	3.309000e+03	3309.000000	3309.000000	3.309000e+03	3309
mean	2017-07-31 13:03:19.093381632	16.489174	8.893862e+07	0.002058	22.830024	8.431116e+05	-0.000122	96.663663	1.801949e+06	
min	2011-01-04 00:00:00	0.532294	1.360320e+07	-0.344652	13.947239	3.700000e+04	-0.075558	76.221649	2.008000e+05	-(
25%	2014-04-21 00:00:00	2.634940	5.132160e+07	-0.014286	19.304272	3.479000e+05	-0.005660	89.900841	6.784000e+05	-(
50%	2017-08-01 00:00:00	8.815700	7.406220e+07	0.003444	21.503807	5.815000e+05	-0.000446	93.443398	1.121900e+06	
75%	2020-11-11 00:00:00	23.848450	1.095624e+08	0.021589	26.490389	9.918000e+05	0.005472	104.720581	2.276900e+06	
max	2024-02-28 00:00:00	86.638939	5.471712e+08	0.269884	43.006519	1.035650e+07	0.062972	118.636902	1.878590e+07	
std	NaN	18.936697	5.592784e+07	0.038521	5.802448	8.810263e+05	0.009435	9.295056	1.838814e+06	

8 rows × 22 columns

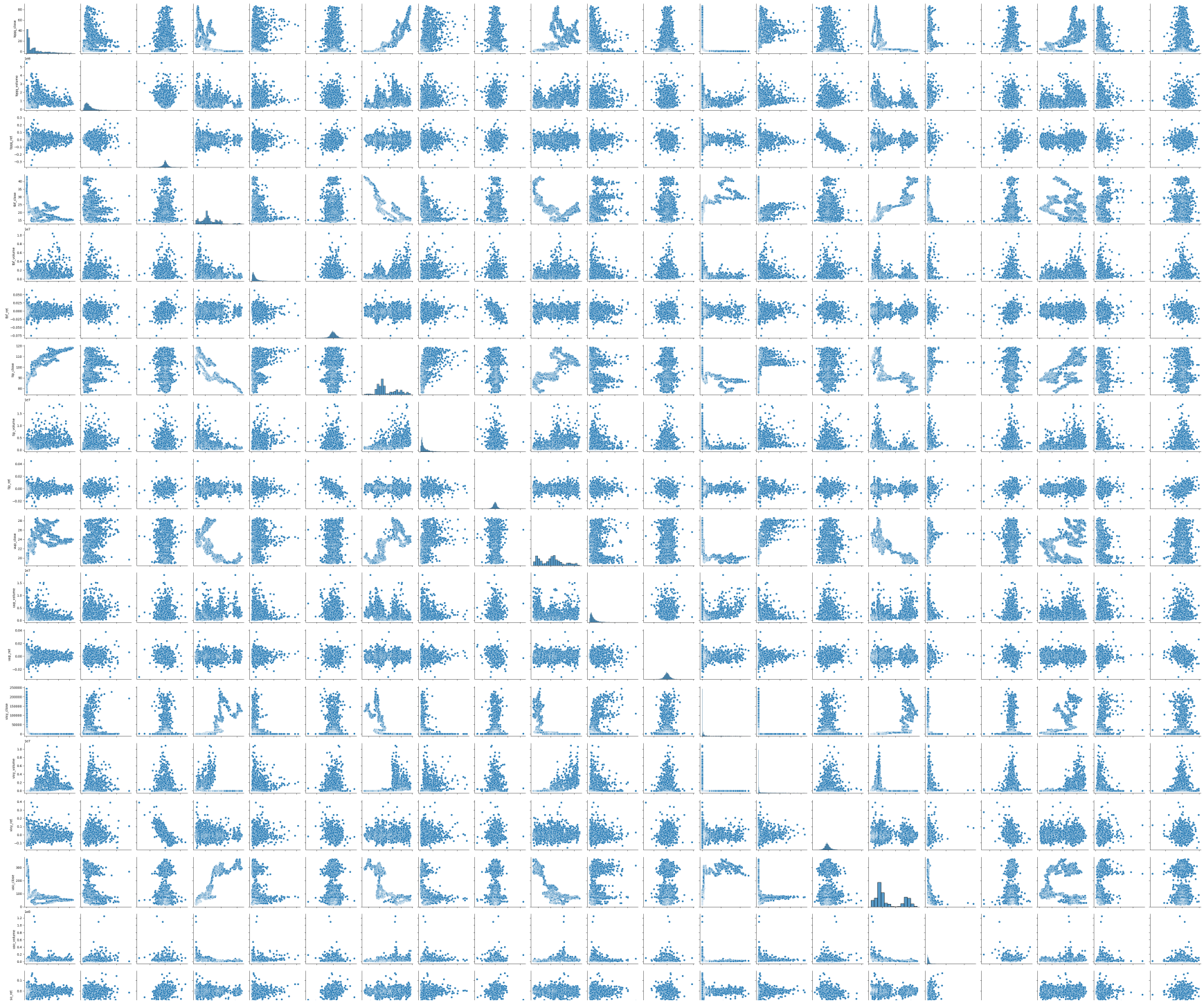
Observations

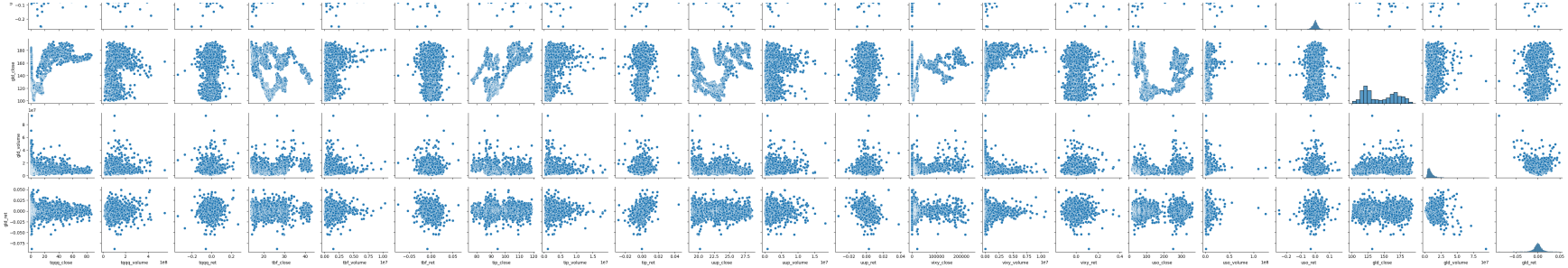
There is not too much information to gather from the summary statistics. Although it gives us a broader understanding of our data, the timeseries nature of the data lends itself fewer conclusions on this information. We can see that there are some features that have decently large std and might lead us to handle the preprocessing of the data differently.

Plotting Behavior

As part of understanding our data, we will review some of the price, volume and daily return behavior for each of the secirities under investigation.

In [ ]:
```
sns.pairplot(df)
plt.show()
```

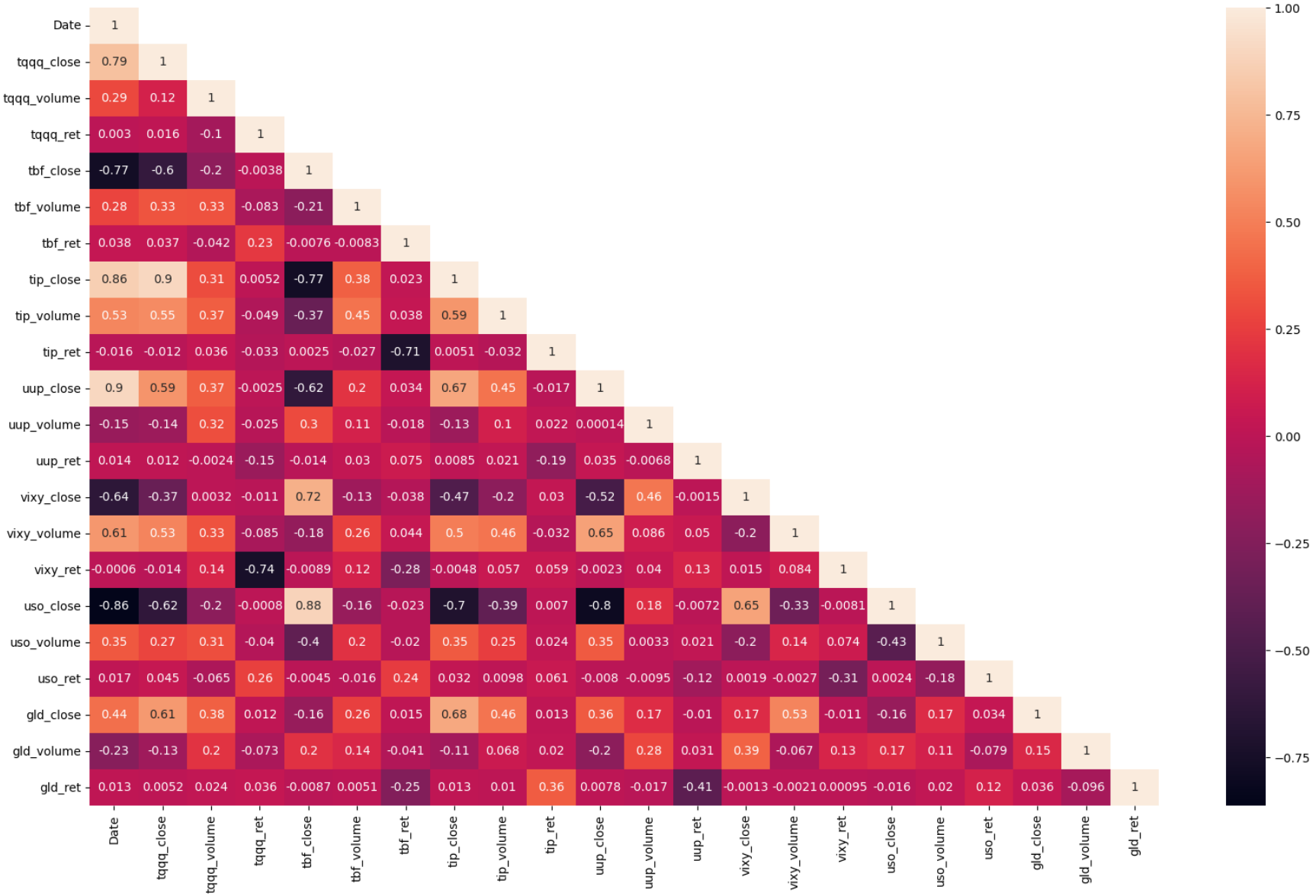




Observations

We will need to download this chart output for a pairplot to see more of the details of the output. However, we can see some features that are correlated and others that show an interesting relationship. There is definitely too much information to gather meaningful results from this chart. A better approach might be to review a correlation heatmap.

In [ ]:
```
df_corr = df.corr()
mask = np.tril(np.ones(df_corr.shape), k=0).astype(bool)
df_corr_tril = df_corr.where(mask)

plt.figure(figsize=(20, 12))
sns.heatmap(df_corr_tril, annot=True)
plt.show()
```

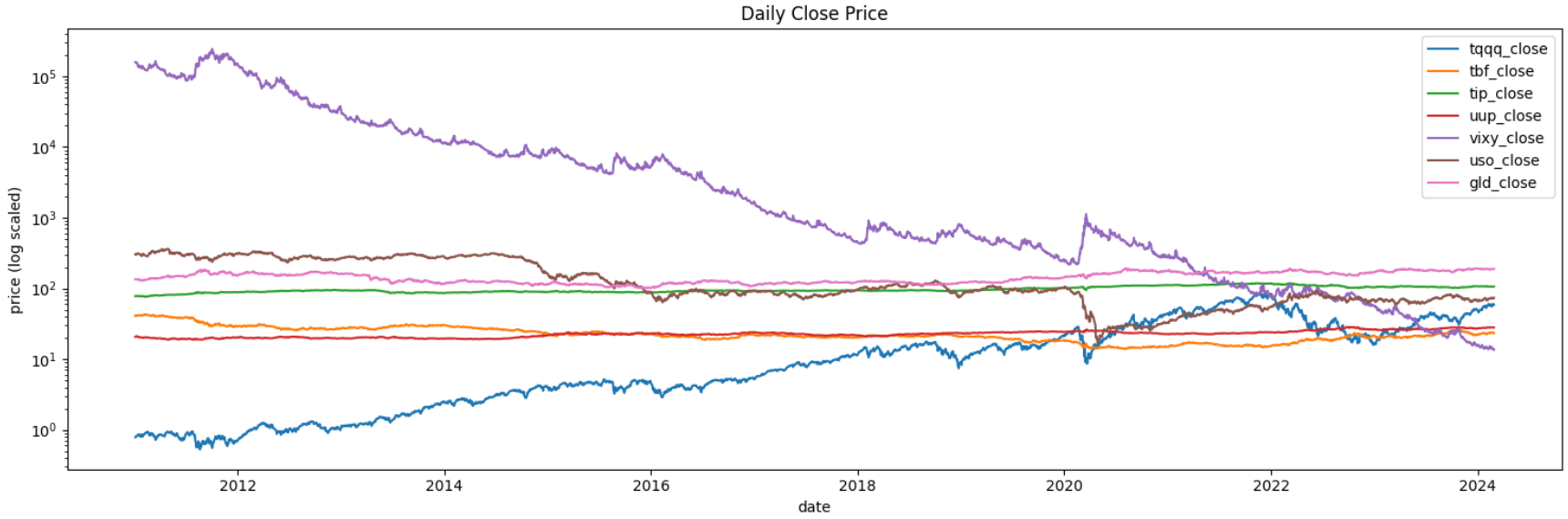


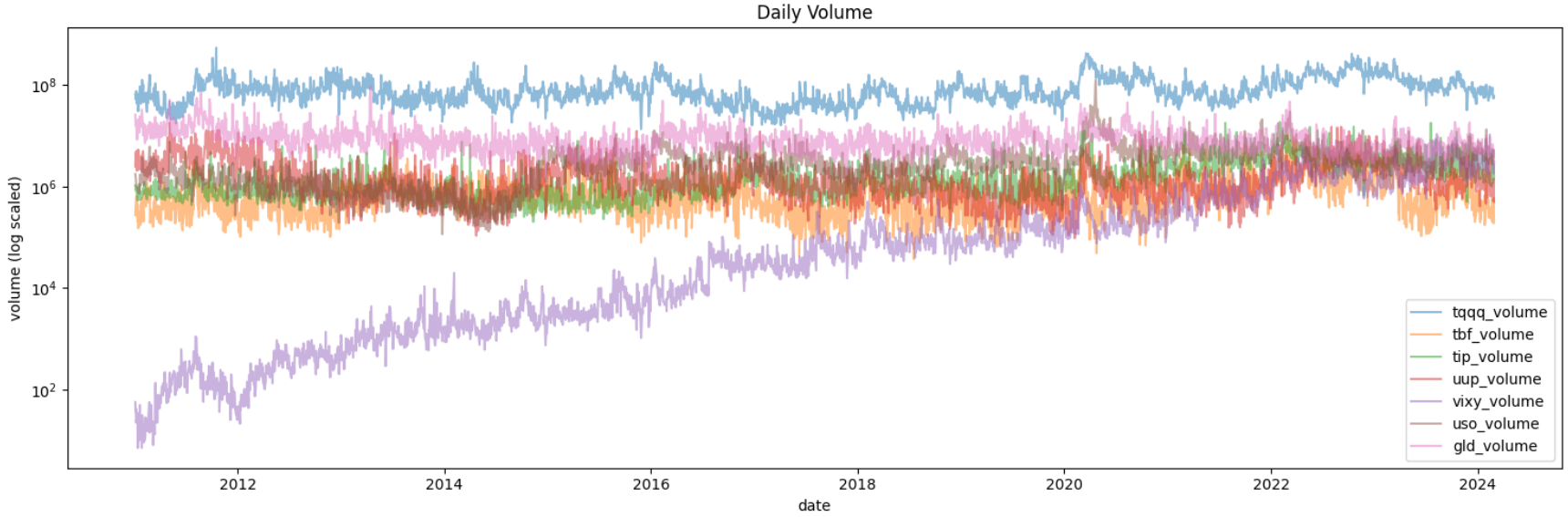
Observations

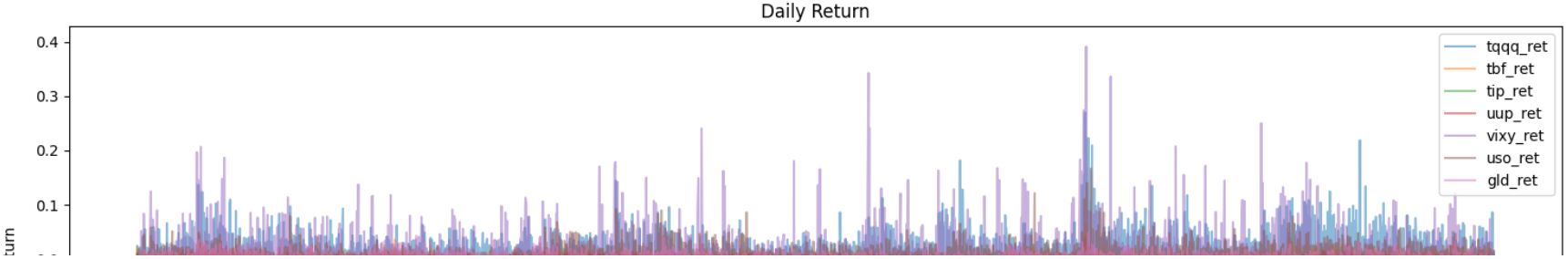
There are some highly positively and negatively correlated features. Before creating a plan of action to handle these, lets look further into the data.

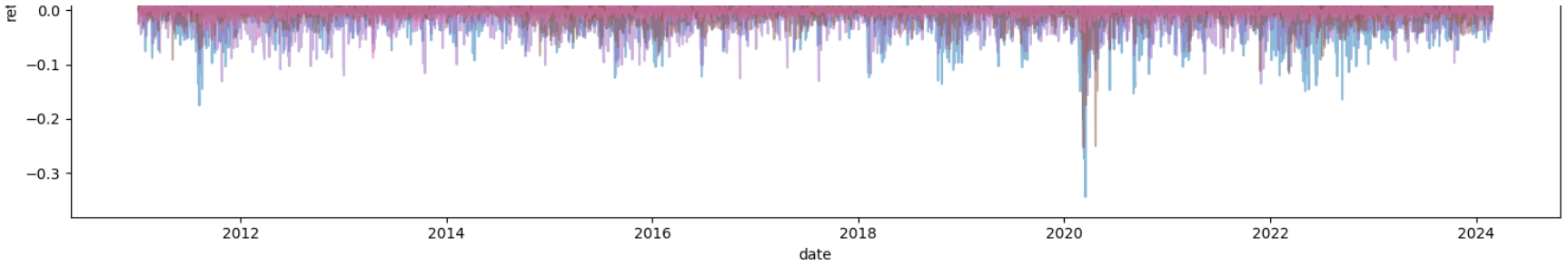
In [ ]:
```
# plotting the daily price movement features
fig, axes = plt.subplots(3, 1, figsize=(15,15))
```

```
axes[0].plot(df.Date, df.tqqq_close, label=df.tqqq_close.name)
axes[0].plot(df.Date, df.tbf_close, label=df.tbf_close.name)
axes[0].plot(df.Date, df.tip_close, label=df.tip_close.name)
axes[0].plot(df.Date, df.uup_close, label=df.uup_close.name)
axes[0].plot(df.Date, df.vixy_close, label=df.vixy_close.name)
axes[0].plot(df.Date, df.uso_close, label=df.uso_close.name)
axes[0].plot(df.Date, df.gld_close, label=df.gld_close.name)
axes[0].legend()
axes[0].set_yscale('log')
axes[0].set_title('Daily Close Price')
axes[0].set_ylabel('price (log scaled)')
axes[0].set_xlabel('date')

axes[1].plot(df.Date, df.tqqq_volume, label=df.tqqq_volume.name, alpha=0.5)
axes[1].plot(df.Date, df.tbf_volume, label=df.tbf_volume.name, alpha=0.5)
axes[1].plot(df.Date, df.tip_volume, label=df.tip_volume.name, alpha=0.5)
axes[1].plot(df.Date, df.uup_volume, label=df.uup_volume.name, alpha=0.5)
axes[1].plot(df.Date, df.vixy_volume, label=df.vixy_volume.name, alpha=0.5)
axes[1].plot(df.Date, df.uso_volume, label=df.uso_volume.name, alpha=0.5)
axes[1].plot(df.Date, df.gld_volume, label=df.gld_volume.name, alpha=0.5)
axes[1].legend()
axes[1].set_yscale('log')
axes[1].set_title('Daily Volume')
axes[1].set_ylabel('volume (log scaled)')
axes[1].set_xlabel('date')

axes[2].plot(df.Date, df.tqqq_ret, label=df.tqqq_ret.name, alpha=0.5)
axes[2].plot(df.Date, df.tbf_ret, label=df.tbf_ret.name, alpha=0.5)
axes[2].plot(df.Date, df.tip_ret, label=df.tip_ret.name, alpha=0.5)
axes[2].plot(df.Date, df.uup_ret, label=df.uup_ret.name, alpha=0.5)
axes[2].plot(df.Date, df.vixy_ret, label=df.vixy_ret.name, alpha=0.5)
axes[2].plot(df.Date, df.uso_ret, label=df.uso_ret.name, alpha=0.5)
axes[2].plot(df.Date, df.gld_ret, label=df.gld_ret.name, alpha=0.5)
axes[2].legend()
axes[2].set_title('Daily Return')
axes[2].set_ylabel('return')
axes[2].set_xlabel('date')

plt.tight_layout()
plt.show()
```


Daily Close Price
price (log scaled)
date
tqqq_close
tbf_close
tip_close
uup_close
vixy_close
uso_close
gld_close


Daily Volume
volume (log scaled)
date
tqqq_volume
tbf_volume
tip_volume
uup_volume
vixy_volume
uso_volume
gld_volume


Daily Return
tqqq_ret
tbf_ret
tip_ret
uup_ret
vixy_ret
uso_ret
gld_ret



Observations

It seems that TQQQ have inverse price behavior to VIXY, while the other securities seem to be considerably stable, at least in comparison to TQQQ, with the exception of USO following similar behavior to TQQQ but with a small delay/ reaction to movements in TQQQ.

Volume for VIXY has steadily increased over time while the volume for all other securities has remained consistent.

It is difficult to find any significant value from the daily returns chart, there is a lot of noise and movement within each security.

Feature Engineering

Engineering additional technical indicators that are used as indication tools of momentum in the market. These tools/indicators usually aid the investor determining if securities are overbought or oversold. Although, in practice they are not 100% accurate, they can give some sign or signal to the investor if price action could possibly increase or decrease.

In [ ]:
```
# feature engineering tqqq technical indicator data
# 35-day EMA
df['tqqq_35_day_ema'] = df['tqqq_close'].ewm(span=35, adjust=False).mean()

# Calculate 200-day SMA
df['tqqq_200_day_sma'] = df['tqqq_close'].rolling(window=200).mean()

# Calculate MACD and MACD signal
df['tqqq_macd'] = df['tqqq_close'].ewm(span=12, adjust=False).mean() - df['tqqq_close'].ewm(span=26, adjust=False).mean()
df['tqqq_macd_signal'] = df['tqqq_macd'].ewm(span=9, adjust=False).mean()

# Calculate RSI
delta = df['tqqq_close'].diff()
gain = (delta.where(delta > 0, 0)).rolling(window=14).mean()
loss = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
rs = gain / loss
df['tqqq_rsi'] = 100 - (100 / (1 + rs))
df.fillna(0, inplace=True)
df.head()
```

Out[ ]:

	Date	tqqq_close	tqqq_volume	tqqq_ret	tbf_close	tbf_volume	tbf_ret	tip_close	tip_volume	tip_ret	...	uso_volume	uso_ret	
0	2011-01-04	0.784261	65760000	-0.001553	40.865879	272600	-0.000226	77.868210	1049900	0.002521	...	1758150	-0.024840	
1	2011-01-05	0.803726	61440000	0.024819	41.751659	402500	0.021675	77.570801	869400	-0.003819	...	1582125	0.011555	
2	2011-01-06	0.810993	52262400	0.009042	41.576344	463800	-0.004199	77.795692	827900	0.002899	...	1399413	-0.021807	
3	2011-01-07	0.809113	76204800	-0.002319	41.373363	599800	-0.004882	78.042297	979300	0.003170	...	1025713	-0.000265	
4	2011-01-10	0.818210	43027200	0.011243	41.133461	654800	-0.005798	78.267204	639600	0.002882	...	980450	0.013273	

5 rows × 27 columns

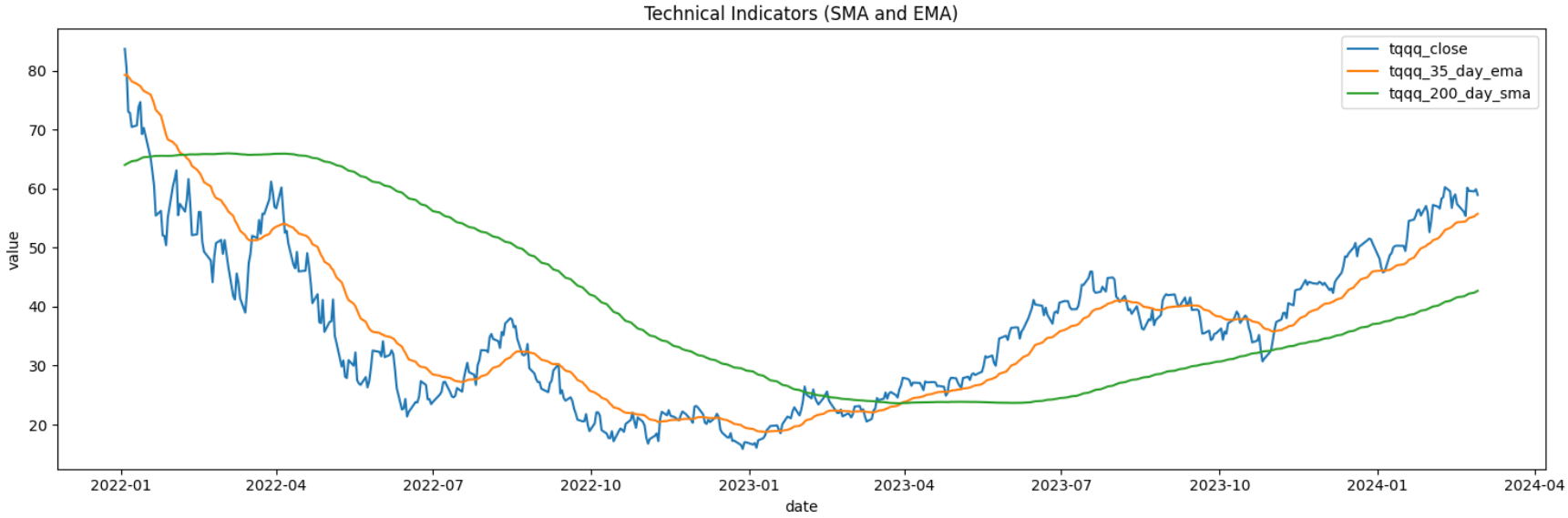
Plotting New Features

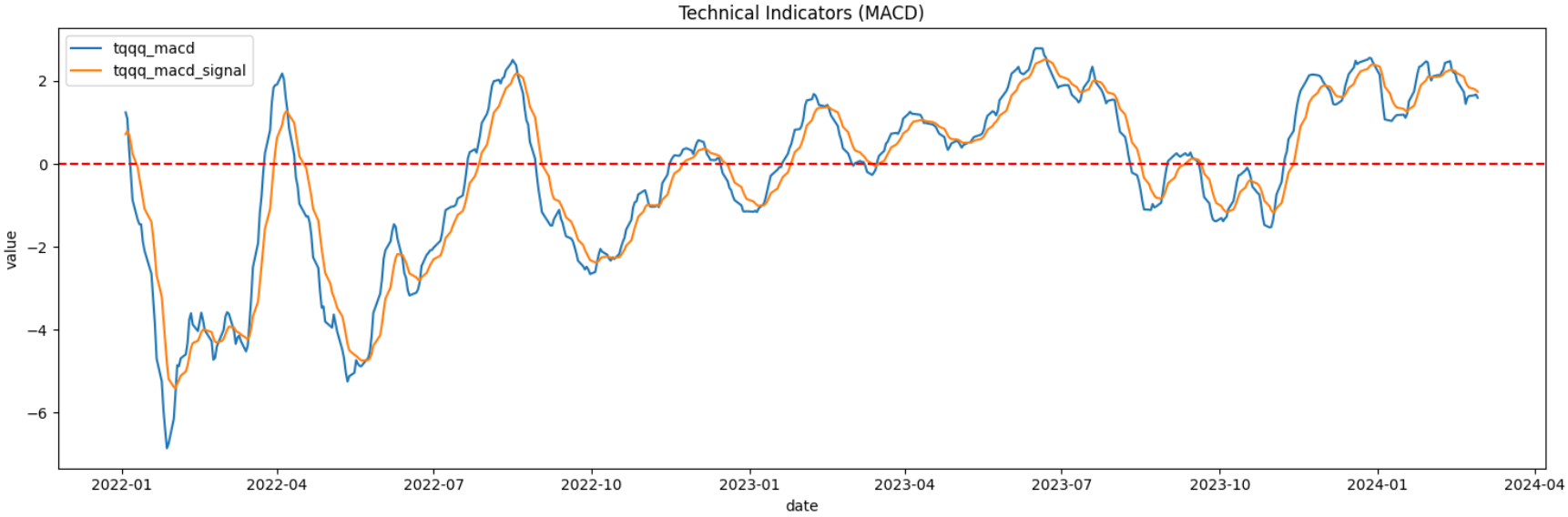
Technical Indicators

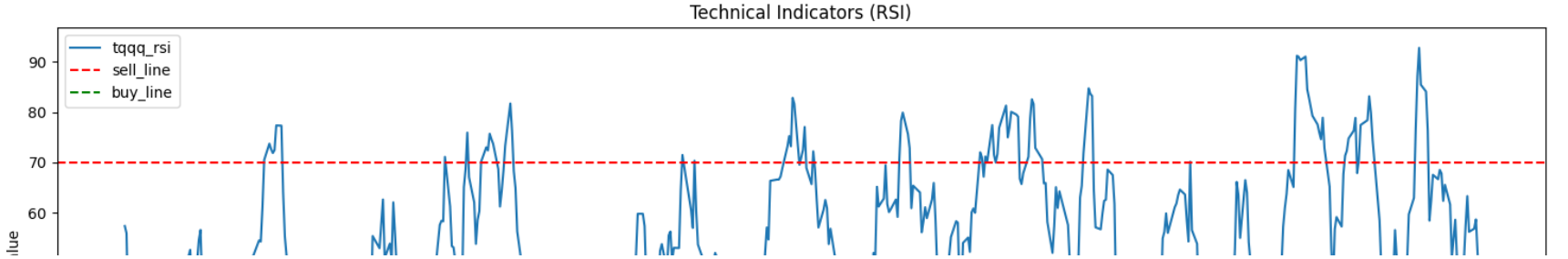
Here, we will be plotting the behvaior of these features to determine the understanding of their behavior with the price movement of TQQQ.

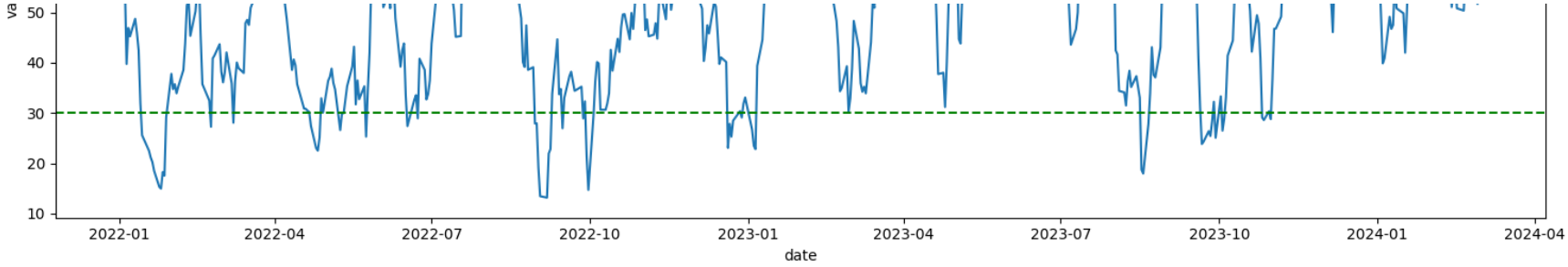
In [ ]:
```
# plotting technical indicators
fig, axes = plt.subplots(3, 1, figsize=(15, 15))

axes[0].plot(df[df.Date > '2022-01-01'].Date, df[df.Date > '2022-01-01'].tqqq_close, label=df.tqqq_close.name)
axes[0].plot(df[df.Date > '2022-01-01'].Date, df[df.Date > '2022-01-01'].tqqq_35_day_ema, label=df.tqqq_35_day_ema.name)
axes[0].plot(df[df.Date > '2022-01-01'].Date, df[df.Date > '2022-01-01'].tqqq_200_day_sma, label=df.tqqq_200_day_sma.name)
axes[0].legend()
axes[0].set_title('Technical Indicators (SMA and EMA)')
axes[0].set_ylabel('value')
axes[0].set_xlabel('date')

axes[1].plot(df[df.Date > '2022-01-01'].Date, df[df.Date > '2022-01-01'].tqqq_macd, label=df.tqqq_macd.name)
axes[1].plot(df[df.Date > '2022-01-01'].Date, df[df.Date > '2022-01-01'].tqqq_macd_signal, label=df.tqqq_macd_signal.name)
axes[1].axhline(y=0, color='r', linestyle='--')
axes[1].legend()
axes[1].set_title('Technical Indicators (MACD)')
axes[1].set_ylabel('value')
axes[1].set_xlabel('date')
```

```
axes[2].plot(df[df.Date > '2022-01-01'].Date, df[df.Date > '2022-01-01'].tqqq_rsi, label=df.tqqq_rsi.name)
axes[2].axhline(y=70, color='r', linestyle='--', label='sell_line')
axes[2].axhline(y=30, color='g', linestyle='--', label='buy_line')
axes[2].legend()
axes[2].set_title('Technical Indicators (RSI)')
axes[2].set_ylabel('value')
axes[2].set_xlabel('date')

plt.tight_layout()
plt.show()
```


Technical Indicators (SMA and EMA)
tqqq_close
tqqq_35_day_ema
tqqq_200_day_sma
value
date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
2024-04


Technical Indicators (MACD)
tqqq_macd
tqqq_macd_signal
value
date
2022-01
2022-04
2022-07
2022-10
2023-01
2023-04
2023-07
2023-10
2024-01
2024-04


Technical Indicators (RSI)
tqqq_rsi
sell_line
buy_line



Observations

According to industry, we can make some assumptions about how these indicators work to signal an investor on shifts in momentum.

The first chart shows the 35 day exponential moving average (EMA) and the 200 day simple moving average (SMA). According to experts, when the the EMA is above the SMA, then we can expect that the market will be trending upwards and visa-versa. Industry experts also determine that the EMA acts as either a support, where the price movement does not go below the EMA line and continues to trend upwards, or a resistor where the price movement will not go above the EMA line and trend downwards. Where as the SMA is identified as the primary trend setter, which means that the direction of the SMA, and whether it converges the EMA, will determine overall price movements.

https://www.investopedia.com/ask/answers/difference-between-simple-exponential-moving-average/#:~:text=Traders%20use%20many%20technical%20indicators,higher%20weighting%20to%20recent%20prices.

The second chart displays the Moving Average Convergence Divergence. This indicator uses two measures, the MACD and the MACD Signal, to determine when prices are essentially too high or too low using points that osccilate between positive and negative values. When the MACD is greater than the MACD Signal and also above 0, it signals to the investor towards a buying opportunity and visa-versa for a selling signal.

https://www.investopedia.com/terms/m/macd.asp

The last chart, Relative Strength Index (RSI), indicates momentum in the markets. This is used to determine at what point a security might be over bought or oversold. This is done by measuring the speed and magnitued of the changes in security pricing. When the momentum indicator is above 70, it indicates over buying and usually leads to a following decrease in price. When it is below 30 it indicates overselling and leads to a following increase in price.

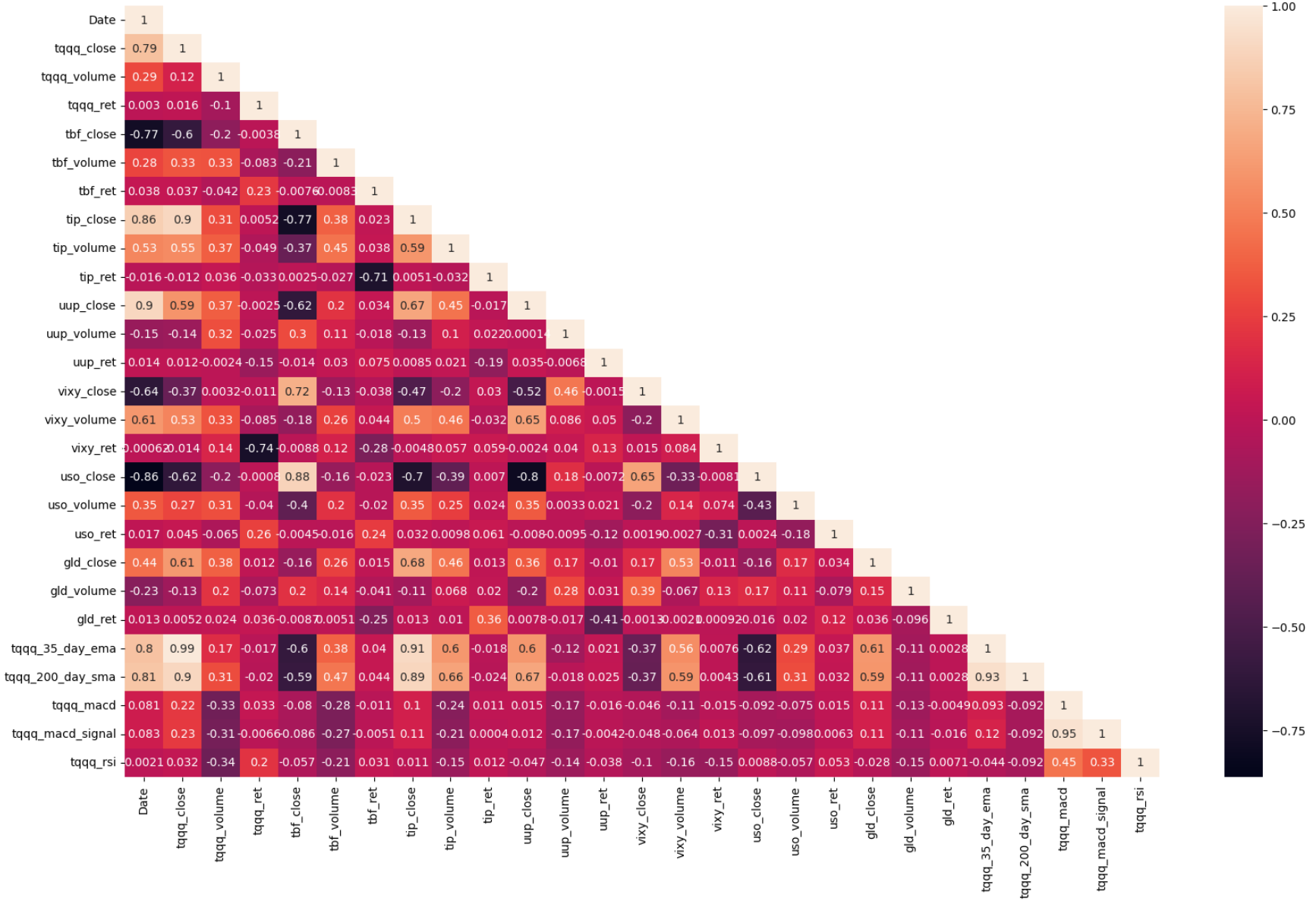
https://www.investopedia.com/terms/r/rsi.asp

Additional Exploratory Data Analysis

Conducting more exploratory analysis on the current data

In [ ]:
```
df_corr = df.corr()
mask = np.tril(np.ones(df_corr.shape), k=0).astype(bool)
```

```
df_corr_tril = df_corr.where(mask)

plt.figure(figsize=(20, 12))
sns.heatmap(df_corr_tril, annot=True)
plt.show()
```



Observations

Although we do have correlations between some features of our data, each feature does maintain some importance due to the methodologies underlying their purpose. We will retain these highly correlated features for now. However, if the model performance is poor, we may attempt to remove the highly correlated features to research the impact on performance.

Buy / Sell Simulation

In the case where we conduct a supervised vs unsupervised model, we will simulate an environment of buy and sell decisions knowing what the future return is. We will conduct a simulation of buy and sell holding decisions for 1, 3, 5, 10, and 20 day buy and sell decisions. These decisions will essentially allow us to determine which holding strategy might work best. For example if we receive a greater return with the 1 day buy/sell decision vs the 3, 5, 10, or 20, then we will execute this strategy for that day. However, if the 10 day buy/sell decision leads to greater return then the others, we will execute this strategy until the 10 days have passed and on the 10th day we will re-evaluate the decisions again. One the day of decision we will label it as a buy, if we decision on longer than 1 day buy/sell then we will label as hold for those days in between. If each buy/sell decision (1, 3, 5, 10, and 20) yields a negative return, then we will decision a sell (short) and use the most negative buy/sell hold decision to execute.

Implementing this strategy will yield the greatest returns and allow us to have a dependent variable: **"decision" dtype: float**

In [ ]:
```
# setting the new features

# buy, sell, hold decision variable
df['decision'] = ''

# simulating cash position set to the starting price of tqqq
df['cash'] = df['tqqq_close'][0]

# daily return on cash, significant for shorts and if we need to add cumulative return at any point
df['cash_ret'] = df['tqqq_ret'][0]

# the initial number of days that we are set to hold a security, could be used as categorical at any point
df['initial_hold_days'] = 1

# countdown check to be sure code is executing properly
df['hold_countdown'] = 1

# variable we will use as a countdown indicator for updating the iterative algorithm
days_to_hold = 0

# indicating how we will change the return value, either maintaining positive for buy or inverse for shorts
hold_type = 'long'

for idx in range(len(df)):
    if idx == 0:
        # decision day 0 as long as entry into the market
        df['decision'][idx] = 'long'
```

```
    elif (idx < len(df)-20) & (idx != 0):

        # reaching the end of the days_to_hold, trigger new hold_dict with future returns
        if days_to_hold <= 1:

            # determine how many days to hold for greatest return
            hold_dict = {
                1: (df['tqqq_close'][idx+1]/df['tqqq_close'][idx])-1,
                3: (df['tqqq_close'][idx+3]/df['tqqq_close'][idx])-1,
                5: (df['tqqq_close'][idx+5]/df['tqqq_close'][idx])-1,
                10: (df['tqqq_close'][idx+10]/df['tqqq_close'][idx])-1,
                20: (df['tqqq_close'][idx+20]/df['tqqq_close'][idx])-1
            }

            # if all future values are negative, short the stock
            if all(value < 0 for value in hold_dict.values()):

                # when the initial position is a short, finish the current day as short
                if hold_type == 'short':

                    df['decision'][idx] = 'short'
                    days_to_hold = min(hold_dict, key=hold_dict.get)
                    df['initial_hold_days'][idx] = days_to_hold
                    df['hold_countdown'][idx] = days_to_hold
                    df['cash'][idx] = df['cash'][idx-1]*(1+(df['tqqq_ret'][idx]*-1))
                    df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
                    hold_type = 'short'

                # when the initial position is a long, finish the current day as long
                elif hold_type == 'long':

                    df['decision'][idx] = 'short'
                    days_to_hold = min(hold_dict, key=hold_dict.get)
                    df['initial_hold_days'][idx] = days_to_hold
                    df['hold_countdown'][idx] = days_to_hold
                    df['cash'][idx] = df['cash'][idx-1]*(1+(df['tqqq_ret'][idx]))
                    df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
                    hold_type = 'short'

            else:

                # go long otherwise
                if hold_type == 'short':
                    # when the initial position is a short, finish the current day as short
                    df['decision'][idx] = 'long'
                    days_to_hold = max(hold_dict, key=hold_dict.get)
                    df['initial_hold_days'][idx] = days_to_hold
                    df['hold_countdown'][idx] = days_to_hold
                    df['cash'][idx] = df['cash'][idx-1]*(1+(df['tqqq_ret'][idx]*-1))
                    df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
```

```
                    hold_type = 'long'

                elif hold_type == 'long':
                    # when the initial position is a long, finish the current day as long
                    df['decision'][idx] = 'long'
                    days_to_hold = max(hold_dict, key=hold_dict.get)
                    df['initial_hold_days'][idx] = days_to_hold
                    df['hold_countdown'][idx] = days_to_hold
                    df['cash'][idx] = df['cash'][idx-1]*(1+(df['tqqq_ret'][idx]))
                    df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
                    hold_type = 'long'

        elif days_to_hold > 1:

            # when a days_to_hold is decided - continue position until end of the hold days
            if hold_type == 'long':
                df['decision'][idx] = 'hold'
                df['cash'][idx] = df['cash'][idx-1]*(1+df['tqqq_ret'][idx])
                df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
                days_to_hold -= 1
                df['hold_countdown'][idx] = days_to_hold
                df['initial_hold_days'][idx] = df['initial_hold_days'][idx-1]

            elif hold_type == 'short':
                df['decision'][idx] = 'hold'
                df['cash'][idx] = df['cash'][idx-1]*(1+(df['tqqq_ret'][idx]*-1))
                df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
                days_to_hold -= 1
                df['hold_countdown'][idx] = days_to_hold
                df['initial_hold_days'][idx] = df['initial_hold_days'][idx-1]

    else:
        # when we get to the end of our dataframe hold our current position (long/short)
        if hold_type == 'long':
            df['decision'][idx] = 'hold'
            df['cash'][idx] = df['cash'][idx-1]*(1+df['tqqq_ret'][idx])
            df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
            df['hold_countdown'][idx] = 0
            df['initial_hold_days'][idx] = df['initial_hold_days'][idx-1]

        elif hold_type == 'short':
            df['decision'][idx] = 'hold'
            df['cash'][idx] = df['cash'][idx-1]*(1+(df['tqqq_ret'][idx]*-1))
            df['cash_ret'][idx] = (df['cash'][idx]/df['cash'][idx-1])-1
            df['hold_countdown'][idx] = 0
            df['initial_hold_days'][idx] = df['initial_hold_days'][idx-1]
```

In [ ]:
```
# 60 day snapshot to confirm the investment decisions are working properly
df[['tqqq_ret', 'tqqq_close', 'decision', 'cash', 'cash_ret', 'initial_hold_days', 'hold_countdown']].iloc[0:60, :]
```

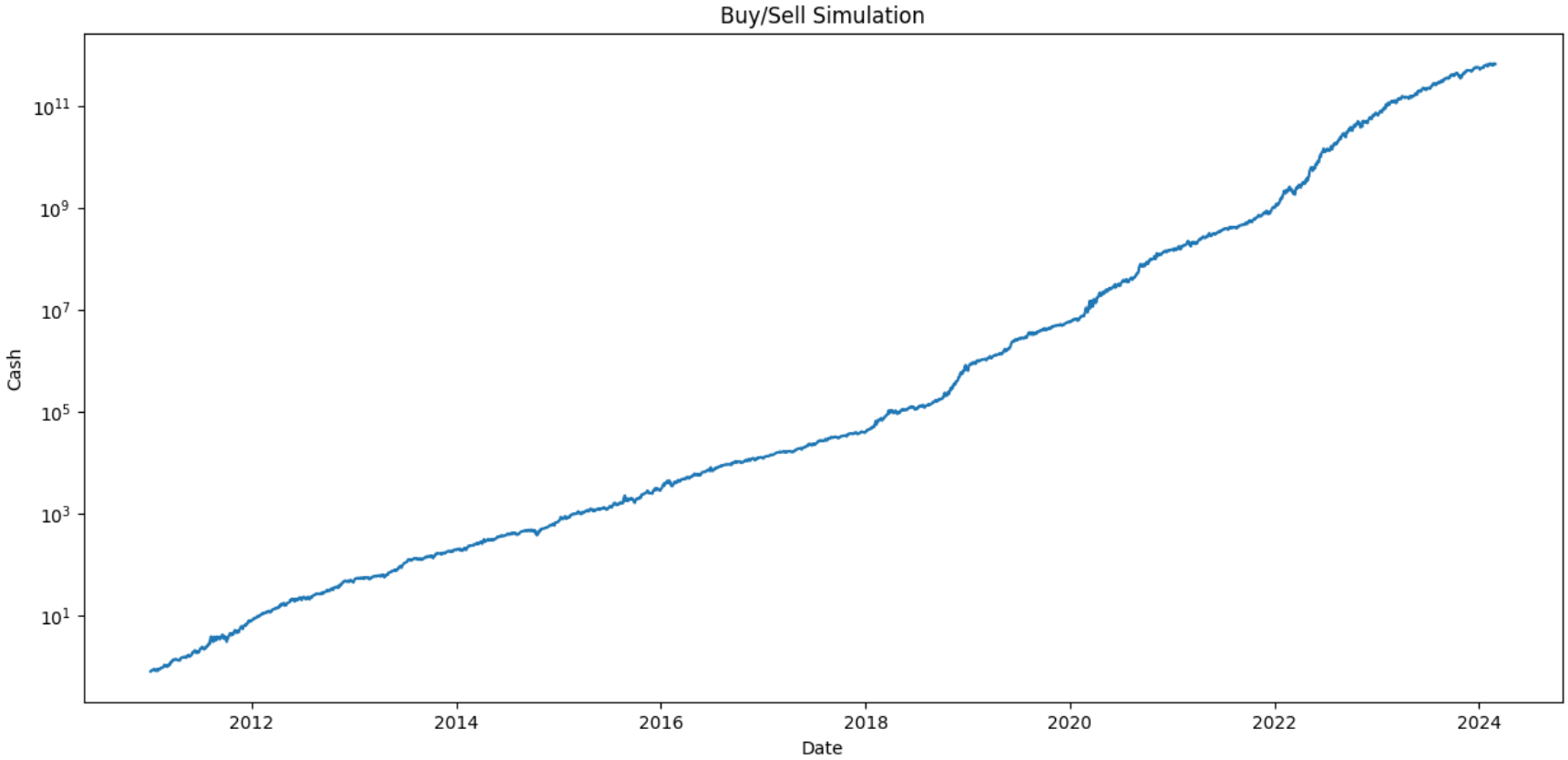
Out[ ]:

	tqqq_ret	tqqq_close	decision	cash	cash_ret	initial_hold_days	hold_countdown
0	-0.001553	0.784261	long	0.784261	-0.001553	1	1
1	0.024819	0.803726	long	0.803726	0.024819	20	20
2	0.009042	0.810993	hold	0.810993	0.009042	20	19
3	-0.002319	0.809113	hold	0.809113	-0.002319	20	18
4	0.011243	0.818210	hold	0.818210	0.011243	20	17
5	0.004596	0.821970	hold	0.821970	0.004596	20	16
6	0.020279	0.838639	hold	0.838639	0.020279	20	15
7	0.003152	0.841282	hold	0.841282	0.003152	20	14
8	0.021686	0.859526	hold	0.859526	0.021686	20	13
9	0.007983	0.866388	hold	0.866388	0.007983	20	12
10	-0.033494	0.837369	hold	0.837369	-0.033494	20	11
11	-0.022456	0.818565	hold	0.818565	-0.022456	20	10
12	-0.022846	0.799864	hold	0.799864	-0.022846	20	9
13	0.041298	0.832897	hold	0.832897	0.041298	20	8
14	0.003478	0.835794	hold	0.835794	0.003478	20	7
15	0.015505	0.848753	hold	0.848753	0.015505	20	6
16	0.019280	0.865117	hold	0.865117	0.019280	20	5
17	-0.075427	0.799864	hold	0.799864	-0.075427	20	4
18	0.014804	0.811705	hold	0.811705	0.014804	20	3
19	0.055535	0.856782	hold	0.856782	0.055535	20	2
20	-0.004508	0.852920	hold	0.852920	-0.004508	20	1
21	0.003872	0.856223	long	0.856223	0.003872	10	10
22	0.018043	0.871672	hold	0.871672	0.018043	10	9
23	0.014518	0.884327	hold	0.884327	0.014518	10	8
24	0.021321	0.903181	hold	0.903181	0.021321	10	7
25	-0.006077	0.897693	hold	0.897693	-0.006077	10	6
26	0.004868	0.902063	hold	0.902063	0.004868	10	5
27	0.020508	0.920562	hold	0.920562	0.020508	10	4

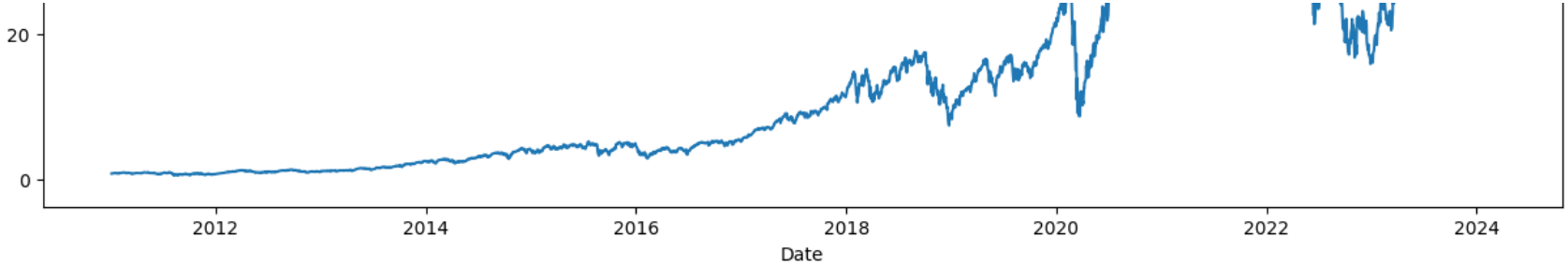
	tqqq_ret	tqqq_close	decision	cash	cash_ret	initial_hold_days	hold_countdown
28	0.006293	0.926355	hold	0.926355	0.006293	10	3
29	-0.002579	0.923967	hold	0.923967	-0.002579	10	2
30	0.018481	0.941043	hold	0.941043	0.018481	10	1
31	-0.000864	0.940230	short	0.940230	-0.000864	20	20
32	-0.005676	0.934893	hold	0.945566	0.005676	20	19
33	-0.088063	0.852564	hold	1.028835	0.088063	20	18
34	-0.024678	0.831525	hold	1.054224	0.024678	20	17
35	0.014240	0.843366	hold	1.039212	-0.014240	20	16
36	0.042844	0.879499	hold	0.994688	-0.042844	20	15
37	0.007281	0.885903	hold	0.987446	-0.007281	20	14
38	-0.048073	0.843315	hold	1.034915	0.048073	20	13
39	0.014101	0.855206	hold	1.020322	-0.014101	20	12
40	0.059663	0.906231	hold	0.959446	-0.059663	20	11
41	-0.014805	0.892814	hold	0.973651	0.014805	20	10
42	-0.039845	0.857240	hold	1.012446	0.039845	20	9
43	0.010433	0.866184	hold	1.001883	-0.010433	20	8
44	-0.020182	0.848702	hold	1.022103	0.020182	20	7
45	-0.045749	0.809875	hold	1.068863	0.045749	20	6
46	0.016692	0.823393	hold	1.051022	-0.016692	20	5
47	-0.009752	0.815364	hold	1.061271	0.009752	20	4
48	-0.040888	0.782025	hold	1.104665	0.040888	20	3
49	-0.073434	0.724598	hold	1.185784	0.073434	20	2
50	0.026091	0.743503	hold	1.154846	-0.026091	20	1
51	-0.005467	0.739438	long	1.161160	0.005467	10	10
52	0.055807	0.780704	hold	1.225961	0.055807	10	9
53	-0.002734	0.778570	hold	1.222609	-0.002734	10	8
54	0.015143	0.790360	hold	1.241124	0.015143	10	7
55	0.054399	0.833354	hold	1.308639	0.054399	10	6

	tqqq_ret	tqqq_close	decision	cash	cash_ret	initial_hold_days	hold_countdown
56	0.007318	0.839453	hold	1.318216	0.007318	10	5
57	-0.016952	0.825223	hold	1.295869	-0.016952	10	4
58	0.028698	0.848905	hold	1.333059	0.028698	10	3
59	0.014728	0.861407	hold	1.352691	0.014728	10	2

In [ ]:
```
fig, axes = plt.subplots(2, 1, figsize=(15, 15))
axes[0].plot(
    df['Date'],
    df['cash'],
    label='cash'
)
axes[0].set_yscale('log')
axes[0].set_title('Buy/Sell Simulation')
axes[0].set_ylabel('Cash')
axes[0].set_xlabel('Date')

axes[1].plot(
    df['Date'],
    df['tqqq_close'],
    label='tqqq'
)
axes[1].set_title('TQQQ Close Price')
axes[1].set_ylabel('Price')
axes[1].set_xlabel('Date')

plt.show()
```


Buy/Sell Simulation
Cash
10^{11}
10^{9}
10^{7}
10^{5}
10^{3}
10^{1}
2012
2014
2016
2018
2020
2022
2024
Date


TQQQ Close Price
Price
80
60
40



Observation

We can see that this is a perfectly executed strategy with outragious returns. We have essentially created a scenario where we have timed the market perfectly. These observations and this simulation will be used as the perfect investor and we can attempt to invest similarly by using the features provided and apply an ML model that can, hopefully, learn similarly.

Handling Preprocessing

In [ ]:
```
# calculate skewness to see if we can use any standardization/normalization that relies on gaussian data
df.loc[:, ~df.columns.isin(['Date', 'decision'])].skew()
```

Out[ ]:
```
tqqq_close            1.488809
tqqq_volume           1.956621
tqqq_ret             -0.383987
tbf_close             1.084706
tbf_volume            3.616385
tbf_ret              -0.098085
tip_close             0.553996
tip_volume            3.049369
tip_ret               0.239377
uup_close             0.381702
uup_volume            2.788063
uup_ret               0.083408
vixy_close            2.754565
vixy_volume           3.632189
vixy_ret              1.590344
uso_close             0.736118
uso_volume           12.507782
uso_ret              -0.919704
gld_close             0.204616
gld_volume            3.594638
gld_ret              -0.400264
tqqq_35_day_ema       1.467674
tqqq_200_day_sma      1.418377
tqqq_macd            -1.241017
tqqq_macd_signal     -1.259827
tqqq_rsi             -0.079500
cash                  4.243408
cash_ret              0.448630
initial_hold_days    -1.603035
hold_countdown        0.305622
dtype: float64
```

Observation

majority of the featres are not normally distributed.

In [ ]:
```
# determining outliars by reviewing values that are greater then 3 or less than -3 standard deviations away from the mean
for col in df.loc[:, ~df.columns.isin(['Date', 'decision'])].columns.values:
    z_scores = (df[col] - df[col].mean()) / df[col].std()
    outliers = np.where((z_scores < -3) | (z_scores > 3))
    print('feature:', col, '\nnumber of outlier data points:', len(outliers[0]), '\n')
```

```
feature: tqqq_close
number of outlier data points: 54

feature: tqqq_volume
number of outlier data points: 63

feature: tqqq_ret
number of outlier data points: 53

feature: tbf_close
number of outlier data points: 76

feature: tbf_volume
number of outlier data points: 63

feature: tbf_ret
number of outlier data points: 30

feature: tip_close
number of outlier data points: 0

feature: tip_volume
number of outlier data points: 67

feature: tip_ret
number of outlier data points: 49

feature: uup_close
number of outlier data points: 0

feature: uup_volume
number of outlier data points: 78

feature: uup_ret
number of outlier data points: 33

feature: vixy_close
number of outlier data points: 109

feature: vixy_volume
number of outlier data points: 83

feature: vixy_ret
number of outlier data points: 51

feature: uso_close
number of outlier data points: 0

feature: uso_volume
number of outlier data points: 38
```

```
feature: uso_ret
number of outlier data points: 38

feature: gld_close
number of outlier data points: 0

feature: gld_volume
number of outlier data points: 57

feature: gld_ret
number of outlier data points: 43

feature: tqqq_35_day_ema
number of outlier data points: 52

feature: tqqq_200_day_sma
number of outlier data points: 0

feature: tqqq_macd
number of outlier data points: 89

feature: tqqq_macd_signal
number of outlier data points: 87

feature: tqqq_rsi
number of outlier data points: 13

feature: cash
number of outlier data points: 124

feature: cash_ret
number of outlier data points: 50

feature: initial_hold_days
number of outlier data points: 28

feature: hold_countdown
number of outlier data points: 0
```

Observation

Many of the features contain outliers

Preprocessing Approach

Our data does not have many features that are normally distributed. Also, the timeseries data lends itself to many values within the data that might show as some outliars or exponentially increasing. We will most likely be using decision trees / random forest at some points to determine feature importance so we will not need to preprocess or transform data for that. However, majority of other ML models would require us to use **RobustScaler**, which is similar to StandardScalar, to ensure that outliers do not skew the data. This is done by useing the median and interquartile ranges for calculations. Also, we might try using **log transformation** which is more ideal for time series financial data due to the log normal distribution nature of the data.

Moving forward in preprocessing we would remove the Date feature, as it is not important to our ML model. We would also remove the cash, cash_ret, initial_hold_days, hold_countdown features as well and use the decision feature as our dependent variable in the case where we use a supervised learning approach. Otherwise, we could look into unsupervised classification methods as well and remove the decision feature.