QTSProject_Stein_Version

February 23, 2023

[ ]:
```python
from matplotlib import pyplot as plt

import pandas as pd
import numpy as np
import functools
import statistics
import math
import os
from datetime import datetime, timedelta
import random
import scipy as sp
import warnings
import gzip
from collections import Counter
from sklearn.linear_model import LinearRegression
from sklearn import linear_model
warnings.filterwarnings("ignore", category=UserWarning, module="pandas")
pd.options.mode.chained_assignment = None
import statsmodels
from statsmodels.regression.rolling import RollingOLS
import requests
import matplotlib as mpl
plt.style.use('seaborn')
mpl.rcParams['font.family'] = 'serif'
```

[ ]:
```python

```

---

1 Motivation

American legislators, such members of Congress, have access to priviledged information regarding governmental affairs, the economic landscape, and the regulatory future of the US and they are able to directly impact and influence policy and firms. Despite this, it is incredibly common for legislators to conduct strategic open-market activities that allow them to benefit directly from policy decisions that they have influence over or inside information on. As a result, public perception of these trades, made public knowledge due to the STOCK Act of 2012, which forces members of government to disclose open-market activity, suggests that these trades contain material information on either the value of the firm or some future event.

Contrary to popular belief, findings by Abdurankhmonov et al. (2022) suggest that positive abnormal returns resulting from trading disclosure materialize the *day of* disclosure and in the time period immediately after disclosure negative abnormal returns are more likely. While surprising, this is in line with literature, with Bellmont et al. (2022) and Hall et al. (2021) also discovering that, on average, Congress members performed only slightly better than the market, and that members of Congress had higher and more robust excess returns in the pre-STOCK Act period, before 2012. For the purposes of this project, the implication is that a viable trading strategy may in fact be found in betting against, rather than with, the trades of Congresspeople and legislators.

In this project, we will create a systematic trading strategy that incorporates various signals stemming from publicly available data sources in order to invest in a selection of different assets in order to generate excess returns that are largely uncorrelated with the market and other common market factors in the Fama-French 5 factor model.

1.1 Strategy

1.2 Leverage

1.3 Risk management

1.4 Evidence for excess returns

[ ]:
```
data_senate = pd.read_csv('C:/Users/dcste/OneDrive/Economics_Research/
↪Economics_Research/trade_transactions.csv')
```

[ ]:
```
data_senate['disclosure_date'] = pd.to_datetime(data_senate['disclosure_date'])
data_senate = data_senate.set_index(data_senate['disclosure_date'])
data_senate = data_senate.sort_index()
data_senate = data_senate.dropna(subset=['ticker'])
data_senate
```

[ ]:
```
                transaction_date disclosure_date   owner ticker  \
disclosure_date
2014-01-31             1/24/2014      2014-01-31  Spouse     GE
2014-01-31             1/24/2014      2014-01-31  Spouse    CRM
2014-01-31             1/24/2014      2014-01-31  Spouse     FB
2014-01-31             1/28/2014      2014-01-31  Spouse   EBAY
2014-01-31             1/29/2014      2014-01-31  Spouse      C
…                            …             …     …      …
2023-02-14              1/5/2023      2023-02-14   Joint      X
2023-02-14              1/5/2023      2023-02-14   Joint      X
2023-02-14              1/5/2023      2023-02-14   Joint      X
2023-02-14              1/5/2023      2023-02-14   Joint    CLF
2023-02-14             1/11/2023      2023-02-14   Joint    TXN

                                                    asset_description  \
disclosure_date
```

```
2014-01-31                      General Electric Company (NYSE)
2014-01-31                           Salesforce.com, Inc (NYSE)
2014-01-31                              Facebook, Inc. (NASDAQ)
2014-01-31                                   eBay Inc. (NASDAQ)
2014-01-31                               Citigroup, Inc. (NYSE)
…                                                           …
2023-02-14         United States Steel Corporation Common Stock
2023-02-14         United States Steel Corporation Common Stock
2023-02-14         United States Steel Corporation Common Stock
2023-02-14    Cleveland-Cliffs Inc. Common Stock <div class=…
2023-02-14    Texas Instruments Incorporated - Common Stock …

                  asset_type            type             amount comment  \
disclosure_date
2014-01-31               NaN  Sale (Partial)    $1,001 - $15,000      --
2014-01-31               NaN        Purchase    $1,001 - $15,000      --
2014-01-31               NaN        Purchase    $1,001 - $15,000      --
2014-01-31               NaN  Sale (Partial)    $1,001 - $15,000      --
2014-01-31               NaN  Sale (Partial)    $1,001 - $15,000      --
…                          …               …                   …     …
2023-02-14             Stock  Sale (Partial)    $1,001 - $15,000      --
2023-02-14             Stock  Sale (Partial)   $15,001 - $50,000      --
2023-02-14             Stock  Sale (Partial)  $50,001 - $100,000      --
2023-02-14      Stock Option  Sale (Partial)   $15,001 - $50,000      --
2023-02-14      Stock Option  Sale (Partial)    $1,001 - $15,000      --

                       senator  \
disclosure_date
2014-01-31      Susan M. Collins
2014-01-31      Susan M. Collins
2014-01-31      Susan M. Collins
2014-01-31      Susan M. Collins
2014-01-31      Susan M. Collins
…                            …
2023-02-14      Tommy Tuberville
2023-02-14      Tommy Tuberville
2023-02-14      Tommy Tuberville
2023-02-14      Tommy Tuberville
2023-02-14      Tommy Tuberville

                                                        ptr_link  \
disclosure_date
2014-01-31      https://efdsearch.senate.gov/search/view/ptr/5…
2014-01-31      https://efdsearch.senate.gov/search/view/ptr/5…
2014-01-31      https://efdsearch.senate.gov/search/view/ptr/5…
2014-01-31      https://efdsearch.senate.gov/search/view/ptr/5…
2014-01-31      https://efdsearch.senate.gov/search/view/ptr/5…
```

```
…                                                         …
2023-02-14      https://efdsearch.senate.gov/search/view/ptr/9…
2023-02-14      https://efdsearch.senate.gov/search/view/ptr/9…
2023-02-14      https://efdsearch.senate.gov/search/view/ptr/9…
2023-02-14      https://efdsearch.senate.gov/search/view/ptr/9…
2023-02-14      https://efdsearch.senate.gov/search/view/ptr/9…

                     party state  \
disclosure_date
2014-01-31      Republican    ME
2014-01-31      Republican    ME
2014-01-31      Republican    ME
2014-01-31      Republican    ME
2014-01-31      Republican    ME
…                        …    …
2023-02-14      Republican    AL
2023-02-14      Republican    AL
2023-02-14      Republican    AL
2023-02-14      Republican    AL
2023-02-14      Republican    AL

                                                  industry  \
disclosure_date
2014-01-31                Consumer Electronics/Appliances
2014-01-31            Computer Software: Prepackaged Software
2014-01-31      Computer Software: Programming, Data Processing
2014-01-31                                Business Services
2014-01-31                                      Major Banks
…                                                       …
2023-02-14                                  Steel/Iron Ore
2023-02-14                                  Steel/Iron Ore
2023-02-14                                  Steel/Iron Ore
2023-02-14                                 Precious Metals
2023-02-14                                  Semiconductors

                          sector  Unnamed: 15  Unnamed: 16  Unnamed: 17  \
disclosure_date
2014-01-31                Energy          NaN          NaN          NaN
2014-01-31            Technology          NaN          NaN          NaN
2014-01-31            Technology          NaN          NaN          NaN
2014-01-31         Miscellaneous          NaN          NaN          NaN
2014-01-31               Finance          NaN          NaN          NaN
…                            …           …           …           …
2023-02-14           Industrials          NaN          NaN          NaN
2023-02-14           Industrials          NaN          NaN          NaN
2023-02-14           Industrials          NaN          NaN          NaN
2023-02-14      Basic Industries          NaN          NaN          NaN
```

```
2023-02-14              Technology          NaN           NaN           NaN

                 Unnamed: 18  Unnamed: 19
disclosure_date
2014-01-31               NaN          NaN
2014-01-31               NaN          NaN
2014-01-31               NaN          NaN
2014-01-31               NaN          NaN
2014-01-31               NaN          NaN
…                       …           …
2023-02-14               NaN          NaN
2023-02-14               NaN          NaN
2023-02-14               NaN          NaN
2023-02-14               NaN          NaN
2023-02-14               NaN          NaN

[7463 rows x 20 columns]
```

[ ]:
```
data_house = pd.read_csv('C:/Users/dcste/OneDrive/Economics_Research/
  ↪Economics_Research/all_transactions_house (2).csv')
data_house['disclosure_date'] = pd.to_datetime(data_house['disclosure_date'])
data_house = data_house.set_index(data_house['disclosure_date'])
data_house = data_house.sort_index()
data_house = data_house.dropna(subset=['ticker'])
data_house = data_house[data_house['ticker'] != "--"]
data_house = data_house[['disclosure_date', 'transaction_date', 'ticker',␣
  ↪'asset_description', 'type', 'amount', 'representative', 'party','state',␣
  ↪'industry' , 'sector' ]]
```

[ ]:
```
data_senate = data_senate[['disclosure_date', 'transaction_date', 'ticker',␣
  ↪'asset_description', 'type', 'amount', 'senator', 'party','state',␣
  ↪'industry' , 'sector' ]]
```

[ ]:
```
data = pd.concat([data_house, data_senate]).sort_index()
data['representative'] = data['representative'].fillna(data['senator'])
del data['senator']
```

[ ]:
```
data.index[-1]
```

[ ]:
```
Timestamp('2023-02-20 00:00:00')
```

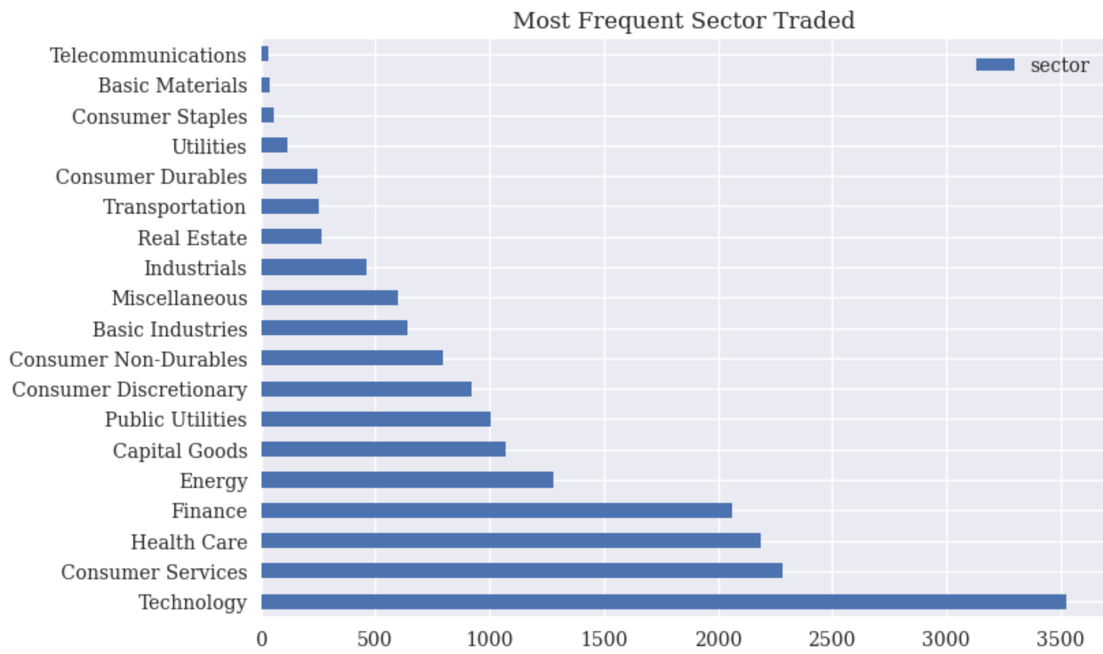
[ ]:
```
prices = pd.read_csv('C:/Users/dcste/OneDrive/Economics_Research/
  ↪Economics_Research/project_price_df.csv')
```

[ ]:
```
most_frequent_tickers = pd.DataFrame(data['ticker'].value_counts()).iloc[:20,:]
```

[ ]:
```
most_frequent_sector = pd.DataFrame(data['sector'].value_counts())
```

[ ]:
```
most_frequent_sector.plot.barh(stacked = True, title = 'Most Frequent Sector Traded')
```

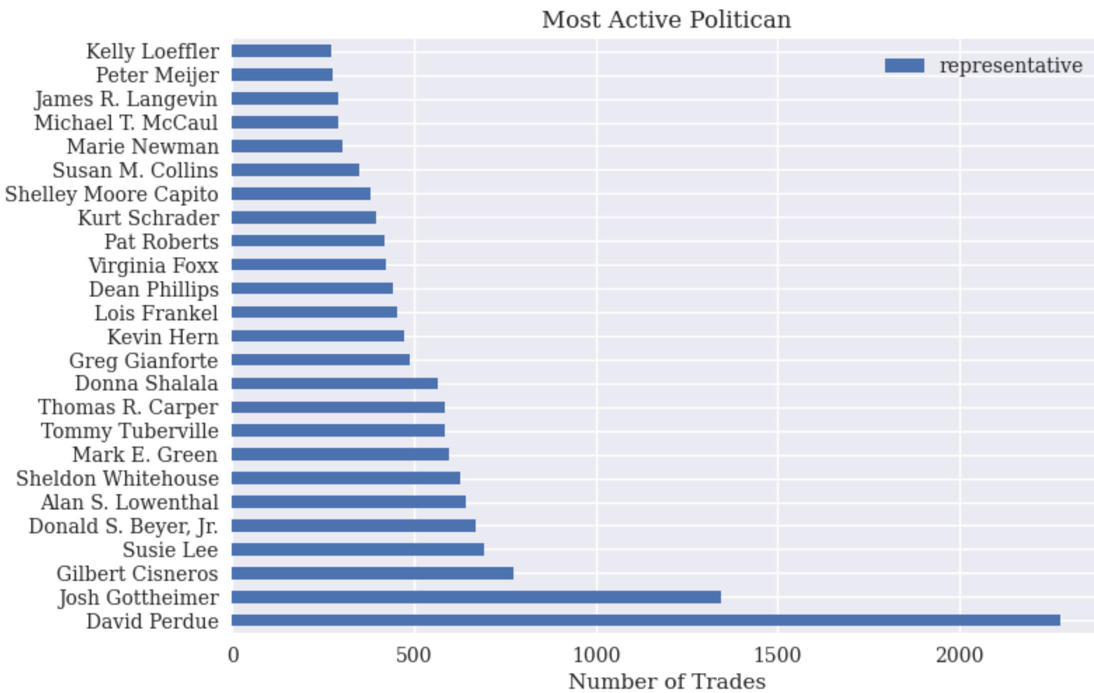
[ ]: <AxesSubplot:title={'center':'Most Frequent Sector Traded'}>



[ ]:
```
most_frequent_politician = pd.DataFrame(data['representative'].value_counts()).iloc[:25,:]
```

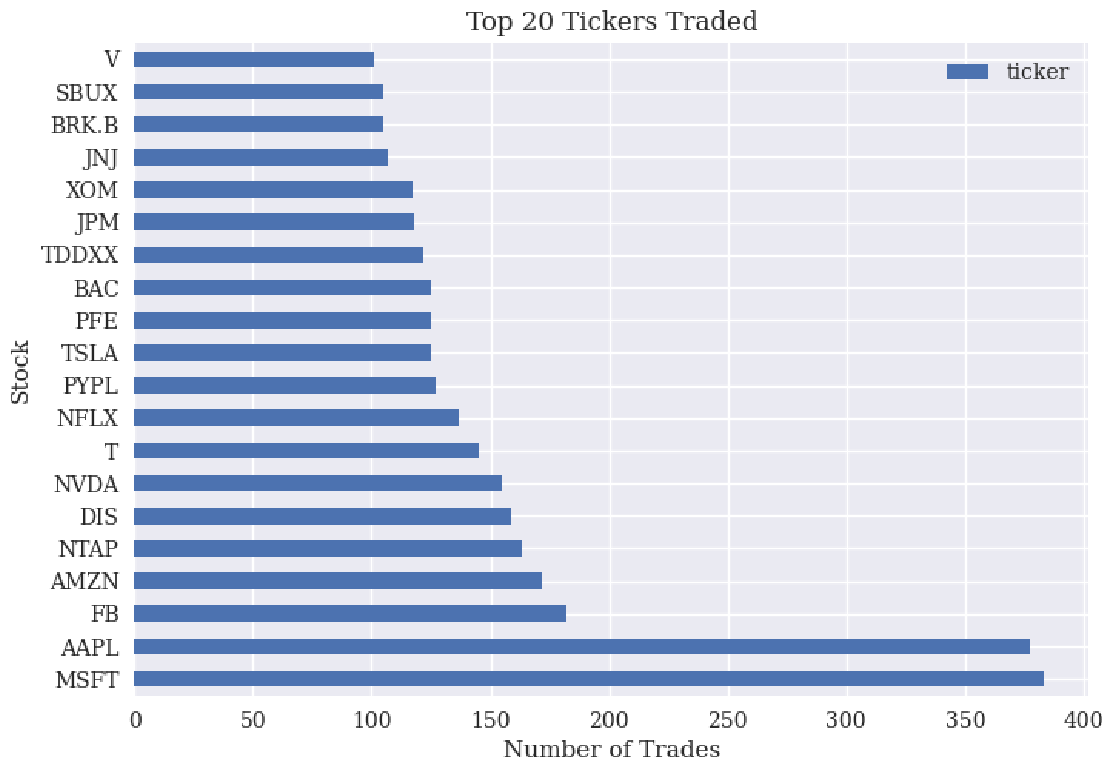
[ ]:
```
most_frequent_politician.plot.barh(stacked = 'True',title = 'Most Active Politican')
plt.xlabel('Number of Trades')
plt.legend()
```

[ ]: <matplotlib.legend.Legend at 0x2516f1b4040>



[ ]:

```
most_frequent_tickers.plot.barh(stacked = True, title = 'Top 20 Tickers Traded')
plt.xlabel('Number of Trades')
plt.ylabel('Stock')
plt.savefig('top_20_tickers_trade.png')
```



2 Downloading Quandl Data

[ ]:
```
import quandl

apikey = 'J_fXGeVW_zC6RaDeJSQv'
quandl.ApiConfig.api_key = apikey
```

[ ]:
```
import fredapi
```

[ ]:
```
# FRED API
api_fred = 'caf2a437b55be8f56406870c1bed3521'
fred = fredapi.Fred(api_key= api_fred)
```

[ ]:
```
period_begin = '2004-01-01'
end_date = data.index[-1]
```

[ ]:
```
end_date
```

[ ]:
```
Timestamp('2023-02-20 00:00:00')
```

```
[ ]: market = quandl.get_table('QUOTEMEDIA/PRICES', ticker = 'SPY',qopts =␣
     ↪{'columns' : ['adj_close','date']}, date = {'gte':period_begin,'lte':
     ↪end_date}).set_index('date').sort_index()
     MARKET_RETURNS = market.resample('M').first().pct_change()
     MARKET_RETURNS.columns = ['MKT_RETS']

     interest_rates = quandl.get('YC/USA', start_date = period_begin,end_date =␣
     ↪end_date)[['1-Month','3-Month','10-Year']]*(1/100)
     interest_rates['term_spread'] =␣
     ↪interest_rates['10-Year']-interest_rates['3-Month']
     interest_rates = interest_rates.resample('M').first()
     spy_earnings_yield = quandl.get('MULTPL/SP500_EARNINGS_YIELD_MONTH', start_date␣
     ↪= period_begin, end_date = end_date).rename(columns={'Value':'MKT_EPS'})
     consumer_sentiment = quandl.get('UMICH/SOC1',start_date = period_begin,␣
     ↪end_date= end_date).rename(columns={'Value':'Industrial_Production'}).
     ↪pct_change().fillna(0)
```

```
[ ]: # Downloading AAA and BAA corporate bond yield from FRED WEBSIDE
     AAA = pd.DataFrame(fred.get_series('DAAA'), columns = ['AAA_Yield'])*(1/100)
     BAA = pd.DataFrame(fred.get_series('DBAA'), columns=['BAA_Yield'])*(1/100)
     corporate_bond_yields = BAA.join(AAA, how = 'inner')
     corporate_bond_yields = corporate_bond_yields.fillna(corporate_bond_yields.
     ↪mean())
```

```
[ ]: corporate_bond_yields['dsspread'] = corporate_bond_yields.BAA_Yield -␣
     ↪corporate_bond_yields.AAA_Yield
```

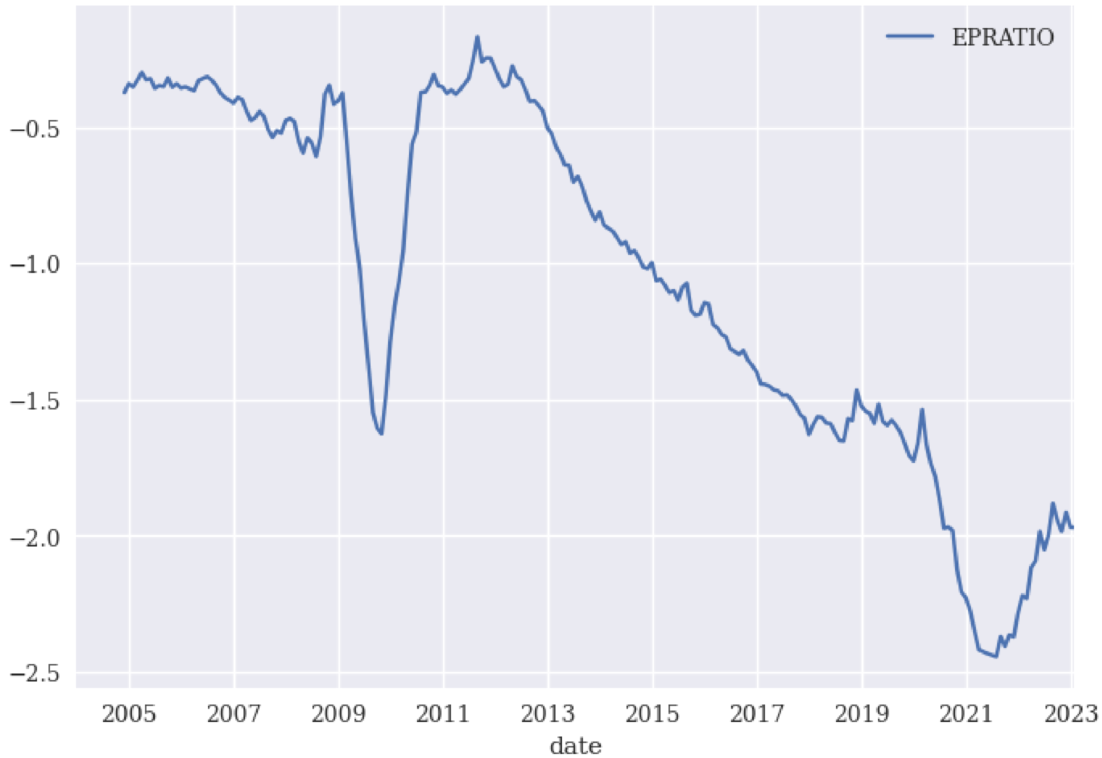
```
[ ]: mprices = market.resample('M').last()
     spy_mend_eps = spy_earnings_yield.resample('M').first()
     spy_mend_eps = spy_mend_eps.rolling(12).sum()
```

```
[ ]: EPRATIO = pd.DataFrame(spy_mend_eps.values/mprices.values, columns=['EPRATIO'],␣
     ↪index = mprices.index)
```

```
[ ]: EPRATIO = pd.DataFrame(np.log(spy_mend_eps).values - np.log(mprices).values,␣
     ↪columns=['EPRATIO'], index = mprices.index)
```

```
[ ]: EPRATIO.plot()
```

```
[ ]: <AxesSubplot:xlabel='date'>
```



[ ]:
```
interest_rates
```

[ ]:
```
            1-Month  3-Month  10-Year  term_spread
Date
2004-01-31   0.0088   0.0093   0.0438       0.0345
2004-02-29   0.0087   0.0094   0.0418       0.0324
2004-03-31   0.0097   0.0097   0.0400       0.0303
2004-04-30   0.0095   0.0093   0.0391       0.0298
2004-05-31   0.0083   0.0100   0.0453       0.0353
…               …        …        …             …
2022-10-31   0.0287   0.0346   0.0367       0.0021
2022-11-30   0.0372   0.0423   0.0407      -0.0016
2022-12-31   0.0404   0.0433   0.0353      -0.0080
2023-01-31   0.0417   0.0453   0.0379      -0.0074
2023-02-28   0.0459   0.0466   0.0339      -0.0127

[230 rows x 4 columns]
```

[ ]:
```
spy_earnings_yield.resample('M').first()
```

[ ]:
```
            MKT_EPS
Date
```

```
2004-01-31   0.0440
2004-02-29   0.0445
2004-03-31   0.0463
2004-04-30   0.0471
2004-05-31   0.0497
…               …
2022-10-31   0.0502
2022-11-30   0.0478
2022-12-31   0.0478
2023-01-31   0.0474
2023-02-28   0.0454

[230 rows x 1 columns]
```

[ ]:
```python
market.resample('M').last()
```

[ ]:
```
              adj_close
date
2004-01-31    78.531486
2004-02-29    79.597207
2004-03-31    78.546884
2004-04-30    77.060675
2004-05-31    78.380209
…                   …
2022-10-31   384.423639
2022-11-30   405.794333
2022-12-31   382.430000
2023-01-31   406.480000
2023-02-28   407.260000

[230 rows x 1 columns]
```

[ ]:
```python
spy_earnings_yield.resample('M').first()
```

[ ]:
```
            MKT_EPS
Date
2004-01-31   0.0440
2004-02-29   0.0445
2004-03-31   0.0463
2004-04-30   0.0471
2004-05-31   0.0497
…               …
2022-10-31   0.0502
2022-11-30   0.0478
2022-12-31   0.0478
2023-01-31   0.0474
2023-02-28   0.0454
```

```
[230 rows x 1 columns]
```

[ ]:
```python
data_copy = data.copy()
```

[ ]:
```python
def filter_trade_type(trade_type:str):
    if trade_type == 'Purchase':
        trade_type = 'Buy'
    elif trade_type == 'purchase':
        trade_type = 'Buy'
    elif trade_type == 'sale':
        trade_type = 'Sell'
    elif trade_type == 'Sale (Partial)':
        trade_type = 'Sell'
    elif trade_type == 'sale_partial':
        trade_type = 'Sell'
    elif trade_type == 'Sale (Full)':
        trade_type = 'Sell'
    elif trade_type == 'sale_full':
        trade_type = 'Sell'
    return trade_type
```

[ ]:
```python
data_copy['type']  = data_copy['type'].apply(filter_trade_type)
```

[ ]:
```python
data_copy = data_copy[(data_copy.type == "Buy")|(data_copy.type == "Sell")]
```

[ ]:
```python
data_copy.head(5)
```

[ ]:
```
                disclosure_date transaction_date ticker  \
disclosure_date
2014-01-31           2014-01-31        1/24/2014     GE
2014-01-31           2014-01-31        1/24/2014    CRM
2014-01-31           2014-01-31        1/24/2014     FB
2014-01-31           2014-01-31        1/28/2014   EBAY
2014-01-31           2014-01-31        1/29/2014      C

                              asset_description  type            amount  \
disclosure_date
2014-01-31       General Electric Company (NYSE)  Sell  $1,001 - $15,000
2014-01-31            Salesforce.com, Inc (NYSE)   Buy  $1,001 - $15,000
2014-01-31               Facebook, Inc. (NASDAQ)   Buy  $1,001 - $15,000
2014-01-31                    eBay Inc. (NASDAQ)  Sell  $1,001 - $15,000
2014-01-31                Citigroup, Inc. (NYSE)  Sell  $1,001 - $15,000

                   representative       party state  \
disclosure_date
2014-01-31       Susan M. Collins  Republican    ME
```

```
2014-01-31       Susan M. Collins  Republican    ME
2014-01-31       Susan M. Collins  Republican    ME
2014-01-31       Susan M. Collins  Republican    ME
2014-01-31       Susan M. Collins  Republican    ME

                                                                  industry  \
disclosure_date
2014-01-31                        Consumer Electronics/Appliances
2014-01-31                    Computer Software: Prepackaged Software
2014-01-31       Computer Software: Programming, Data Processing
2014-01-31                                       Business Services
2014-01-31                                             Major Banks

                        sector
disclosure_date
2014-01-31              Energy
2014-01-31          Technology
2014-01-31          Technology
2014-01-31       Miscellaneous
2014-01-31             Finance
```

- Constructing PTI **Person-Based-Trading Index**

[ ]:
```
PTI_df = data_copy[['transaction_date','ticker','type']]
```

[ ]:
```
p = data_copy[['ticker','type','representative']]
p['Date'] = p.index
```

[ ]:
```
p.head(3)
```

[ ]:
```
                ticker  type    representative       Date
disclosure_date
2014-01-31          GE  Sell  Susan M. Collins 2014-01-31
2014-01-31         CRM   Buy  Susan M. Collins 2014-01-31
2014-01-31          FB   Buy  Susan M. Collins 2014-01-31
```

[ ]:
```
trades_grouped = p.groupby([pd.
 ↪Grouper(key='Date'),'representative','ticker','type']).nunique().
 ↪reset_index()
```

[ ]:
```
trades_grouped
```

[ ]:
```
            Date      representative ticker  type
0     2014-01-31    Susan M. Collins      C  Sell
1     2014-01-31    Susan M. Collins    CRM   Buy
2     2014-01-31    Susan M. Collins   EBAY  Sell
3     2014-01-31    Susan M. Collins     FB   Buy
4     2014-01-31    Susan M. Collins     GE  Sell
```

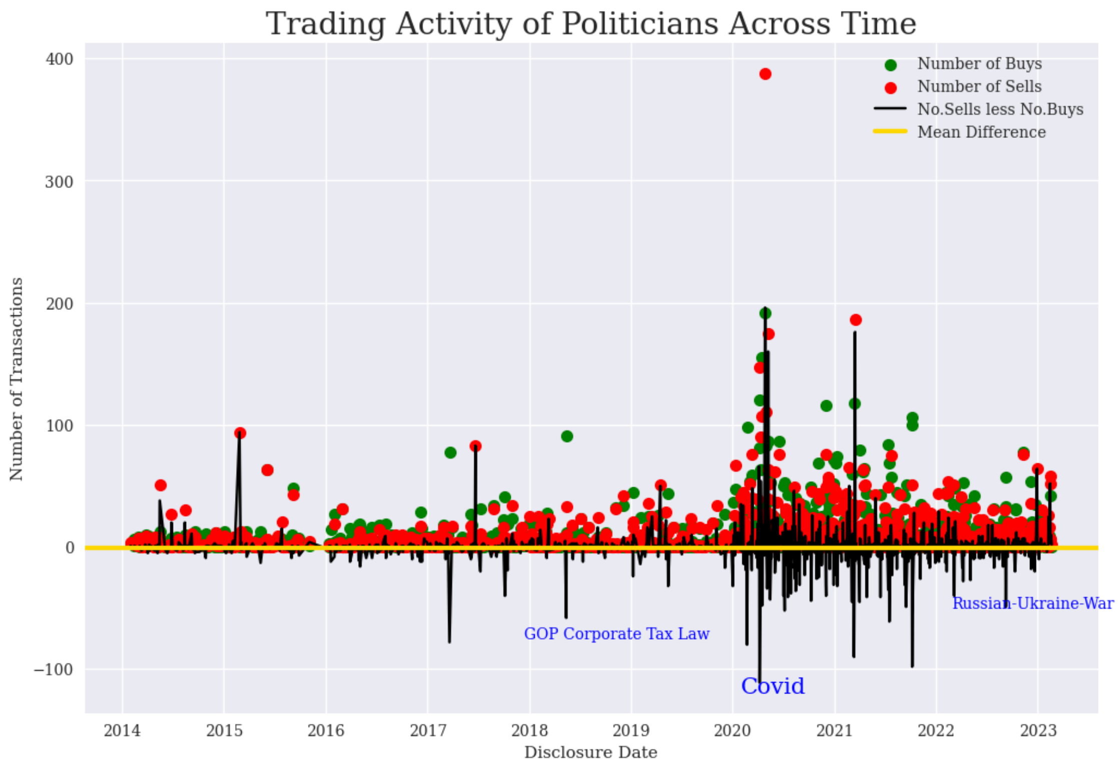
```
…          …                        …     …    …
17187  2023-02-17              Neal P. Dunn  KEY$J  Sell
17188  2023-02-17              Neal P. Dunn   RF$A  Sell
17189  2023-02-17              Neal P. Dunn     SO  Sell
17190  2023-02-17              Seth Moulton   ATVI  Sell
17191  2023-02-20  Debbie Wasserman Schultz    AGI  Sell

[17192 rows x 4 columns]
```

[ ]:
```
trades_grouped = trades_grouped.pivot_table(index = 'Date',columns='type',␣
↪values='ticker',aggfunc='count').fillna(0)
```

[ ]:
```
# Use Trades Grouped to Calculate Trading Index
trades_grouped['Difference'] = trades_grouped['Buy']-trades_grouped['Sell']
trades_grouped['No_Trades'] = trades_grouped['Buy'] + trades_grouped['Sell']
```

[ ]:
```
buys_sells = PTI_df.pivot_table(index = PTI_df.index,␣
↪columns='type',values='ticker',aggfunc='count')
buys_sells = buys_sells.fillna(0)
buys_sells['Difference'] = buys_sells['Buy'] - buys_sells['Sell']
buys_sells['No_Trades'] = buys_sells['Buy'] + buys_sells['Sell']
```

[ ]:
```
plt.figure(figsize=(12,8))
plt.scatter(buys_sells.index,buys_sells.Buy, c = 'green', label = 'Number of␣
↪Buys', linewidth = 1)
plt.scatter(buys_sells.index,buys_sells.Sell, c = 'red', label = 'Number of␣
↪Sells', linewidth = 1)
plt.plot(buys_sells.Sell- buys_sells.Buy,c='black',label= 'No.Sells less No.
↪Buys')
plt.axhline((buys_sells.Sell-buys_sells.Buy).mean(), label = 'Mean Difference',␣
↪linewidth = 3,color = 'gold')
plt.text(pd.to_datetime('2020-01-30'), -120,'Covid', c ='blue', fontsize = 15)
plt.text(pd.to_datetime('2022-02-24'),-50,'Russian-Ukraine-War', c = 'blue',␣
↪fontsize = 10)
plt.text(pd.to_datetime('2017-12-15'),-75,'GOP Corporate Tax Law', c = 'blue',␣
↪fontsize = 10)
plt.legend(loc =0)
plt.xlabel('Disclosure Date')
plt.ylabel('Number of Transactions')
plt.title('Trading Activity of Politicians Across Time', fontsize = 20)
plt.savefig('Trading_Activity.png')
```



- As you can see from the chart above there are spikes in the **Number of Buys** and **Number of Sells** on any given disclosure date. It begs the question of why? It is human nature that when it comes to money - generally speaking- we are always motiviated in our self interest. Yes, the **STOCK ACT** is *supposed* to prohibit members of Congress and employees of Congress from using priviate information derived from their official positions for their personal benefit. In the court of law, proving such insider trading is probably impossible and not a top priority for the Department of Justice. With that being said, I believe we can find a predictive signal from aggregate *buying* and *selling* activity of United States Politicians.
- Since politicians are privy to sensitive economic, geopolitical, and other important information before others know, we can get a better understanding of their psychological mindset. *Buying* and *Selling* relate to fear and greed. If politicians know senstive macroeconomic information, they will without a doubt react emotionally through buying and selling out of greed or fear.
- For example, even though many investors *knew* about Covid-19, **many investors did not know** just how bad it would affect the global economy. However, being that polticians are surrouded by top scientists and the most up-to-date information, they have a better perspective on the gravity of the situation. It would make sense they would trade on this knowledge by selling off assets and raising cash to protect their money.
- In the following week, month, or even quarter markets will begin to *price-in* this negative sentiment.

[ ]:
```
buy_sell = buys_sells.groupby(pd.Grouper(freq = '3M')).sum()
```

[ ]:
```
pti_index = trades_grouped.groupby(pd.Grouper(freq = '1m')).sum()
```

[ ]:
```
pti_index
```

[ ]:
```
type           Buy   Sell  Difference  No_Trades
Date
2014-01-31     3.0    3.0         0.0        6.0
2014-02-28    20.0   16.0         4.0       36.0
2014-03-31    23.0   16.0         7.0       39.0
2014-04-30    29.0   24.0         5.0       53.0
2014-05-31    29.0   65.0       -36.0       94.0
…             …      …            …          …
2022-10-31    92.0   87.0         5.0      179.0
2022-11-30   132.0  142.0       -10.0      274.0
2022-12-31   148.0  187.0       -39.0      335.0
2023-01-31    80.0   81.0        -1.0      161.0
2023-02-28    73.0  145.0       -72.0      218.0

[110 rows x 4 columns]
```

[ ]:
```
pti_index['PTI_Index'] = pti_index.Difference/pti_index['No_Trades']
```

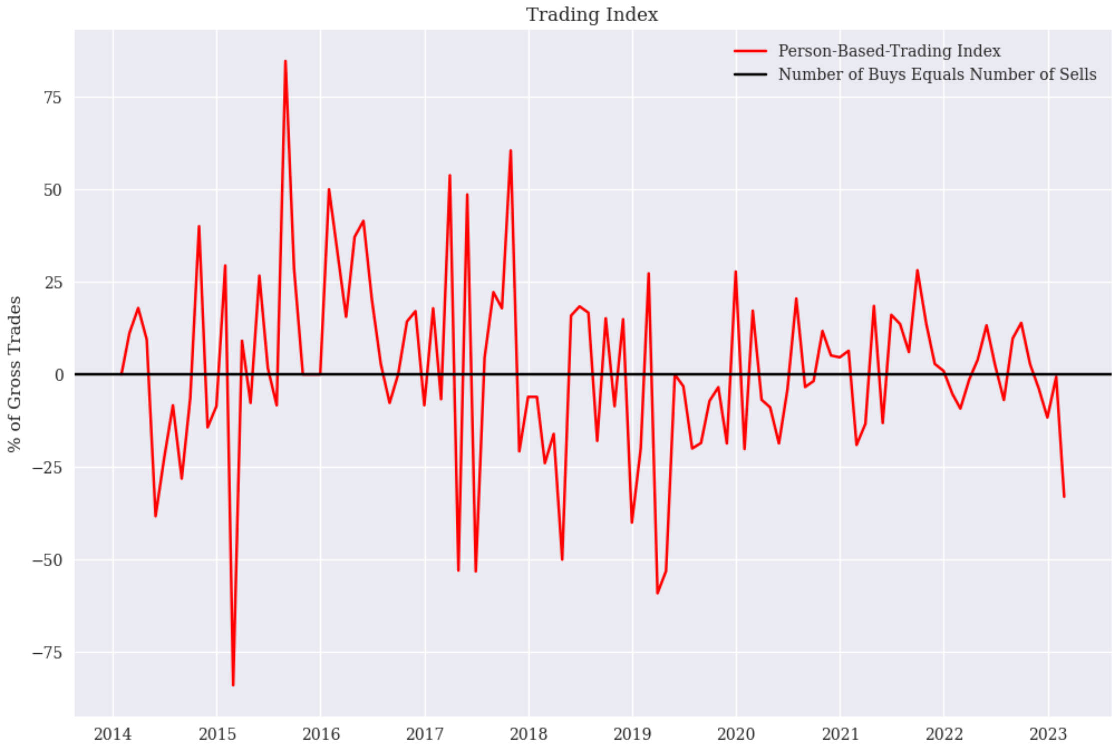
[ ]:
```
pti_index = pti_index.fillna(0)
```

[ ]:
```
pti_index['PTI_Index'].idxmax()
```

[ ]:
```
Timestamp('2015-08-31 00:00:00', freq='M')
```

[ ]:
```
t = 'If the index level is 75%, this means 75% of all trades in the time period␣
↪were buys.'
```

[ ]:
```
plt.figure(figsize=(12,8))
plt.plot(pti_index['PTI_Index']*100,label = 'Person-Based-Trading Index', c =␣
↪'red')
plt.title('Trading Index')
plt.axhline(0,label= 'Number of Buys Equals Number of Sells', c = 'black')
plt.ylabel('% of Gross Trades')
plt.figtext(x = .5,y = 0.01, s = t, fontsize = '12', wrap = True,␣
↪horizontalalignment = 'center')
plt.legend(loc = 0)
plt.savefig('PTI_Index.png')
```



If the index level is 75%, this means 75% of all trades in the time period were buys.

[ ]:
```
pti_index = pti_index.replace(0,.001)
```

[ ]:
```
pti_index['PTI_Index'].std()
```

[ ]: 0.25063021081274484

[ ]:
```
# Calculating Z-score PTI_Index
pti_index['Z_PTI_Index'] = (pti_index['PTI_Index']-pti_index['PTI_Index'].
 ↪mean())/(pti_index['PTI_Index'].std())
```

[ ]:
```

```

[ ]:

type	Buy	Sell	Difference	No_Trades	PTI_Index	\
count	110.000000	110.000000	110.000000	110.000000	110.000000	
mean	77.400000	78.890909	-1.490909	156.290909	0.012192	
std	96.077014	106.455258	36.027193	199.572219	0.250630	
min	0.000000	0.000000	-162.000000	0.000000	-0.839286	
25%	18.250000	16.000000	-10.000000	35.000000	-0.088022	
50%	30.000000	32.000000	0.000000	66.000000	0.000000	
75%	117.000000	123.000000	10.000000	243.250000	0.157937	
max	626.000000	748.000000	89.000000	1374.000000	0.846154	

```
type     Z_PTI_Index
count  1.100000e+02
mean  -1.614870e-17
std    1.000000e+00
min   -3.397348e+00
25%   -3.998513e-01
50%   -4.864675e-02
75%    5.815107e-01
max    3.327458e+00
```

[ ]:
```
trades_grouped
```

[ ]:
```
type         Buy  Sell  Difference  No_Trades  PTI_Index
Date
2014-01-31   3.0   3.0         0.0        6.0   0.000000
2014-02-05   3.0   3.0         0.0        6.0   0.000000
2014-02-11   6.0   3.0         3.0        9.0   0.333333
2014-02-14   1.0   2.0        -1.0        3.0  -0.333333
2014-02-25   2.0   1.0         1.0        3.0   0.333333
…             …    …          …         …         …
2023-02-13   9.0  15.0        -6.0       24.0  -0.250000
2023-02-14   6.0  47.0       -41.0       53.0  -0.773585
2023-02-15   2.0   6.0        -4.0        8.0  -0.500000
2023-02-17  33.0  38.0        -5.0       71.0  -0.070423
2023-02-20   0.0   1.0        -1.0        1.0  -1.000000

[1250 rows x 5 columns]
```

[ ]:
```
traded_days = market.loc['2014-01-01':'2023-02-21',:].index
```

[ ]:
```
buy_sell['PTI_Index'] = buy_sell['Difference']/buy_sell['No_Trades']
```

[ ]:
```
buy_sell['PTI_Index'].idxmin()
```

[ ]:
```
Timestamp('2015-04-30 00:00:00', freq='3M')
```

[ ]:
```
buys_sells.reset_index()
```

[ ]:
```
type disclosure_date   Buy  Sell  Difference  No_Trades Disclosure_Date
0         2014-01-31   3.0   4.0        -1.0        7.0      2014-01-31
1         2014-02-05   3.0   3.0         0.0        6.0      2014-02-05
2         2014-02-11   6.0   4.0         2.0       10.0      2014-02-11
3         2014-02-14   1.0   2.0        -1.0        3.0      2014-02-14
4         2014-02-25   2.0   1.0         1.0        3.0      2014-02-25
…                …    …    …          …         …              …
1245      2023-02-13   9.0  16.0        -7.0       25.0      2023-02-13
1246      2023-02-14   6.0  58.0       -52.0       64.0      2023-02-14
```

```
1247      2023-02-15   3.0   6.0         -3.0        9.0      2023-02-15
1248      2023-02-17  42.0  52.0        -10.0       94.0      2023-02-17
1249      2023-02-20   0.0   2.0         -2.0        2.0      2023-02-20

[1250 rows x 6 columns]
```

[ ]:
```
buys_sells = buys_sells.merge(dts, how = 'left', left_on=['Disclosure_Date'])#
 ↪right_on=['Month_Start'])
buys_sells.index = buys_sells['Disclosure_Date']
```

[ ]:
```
MARKET_RETURNS
```

[ ]:
```
            MKT_RETS
date
2004-01-31       NaN
2004-02-29  0.024634
2004-03-31  0.019216
2004-04-30 -0.017005
2004-05-31 -0.014326
…                …
2022-10-31 -0.071370
2022-11-30  0.048853
2022-12-31  0.059451
2023-01-31 -0.060853
2023-02-28  0.078725

[230 rows x 1 columns]
```

[ ]:
```
pti_index
```

[ ]:
```
type          Buy   Sell  Difference  No_Trades  PTI_Index
Date
2014-01-31    3.0    3.0         0.0        6.0   0.000000
2014-02-28   20.0   16.0         4.0       36.0   0.111111
2014-03-31   23.0   16.0         7.0       39.0   0.179487
2014-04-30   29.0   24.0         5.0       53.0   0.094340
2014-05-31   29.0   65.0       -36.0       94.0  -0.382979
…             …     …          …         …          …
2022-10-31   92.0   87.0         5.0      179.0   0.027933
2022-11-30  132.0  142.0       -10.0      274.0  -0.036496
2022-12-31  148.0  187.0       -39.0      335.0  -0.116418
2023-01-31   80.0   81.0        -1.0      161.0  -0.006211
2023-02-28   73.0  145.0       -72.0      218.0  -0.330275

[110 rows x 5 columns]
```

[ ]:
```
market.resample('M').last()
```

[ ]:
```
            adj_close
date
2004-01-31   78.531486
2004-02-29   79.597207
2004-03-31   78.546884
2004-04-30   77.060675
2004-05-31   78.380209
…                  …
2022-10-31  384.423639
2022-11-30  405.794333
2022-12-31  382.430000
2023-01-31  406.480000
2023-02-28  407.260000

[230 rows x 1 columns]
```

[ ]:
```
market
```

[ ]:
```
            adj_close
date
2004-01-02   76.974420
2004-01-05   77.811775
2004-01-06   77.887899
2004-01-07   78.150868
2004-01-08   78.462279
…                  …
2023-02-13  412.830000
2023-02-14  412.640000
2023-02-15  413.980000
2023-02-16  408.280000
2023-02-17  407.260000

[4816 rows x 1 columns]
```

3 Building Predictive Regression

[ ]:
```
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

[ ]:
```
PTI_lag1 = pti_index[['Z_PTI_Index']].shift(3)
MARKET_lag1 = MARKET_RETURNS.shift(1)
MARKET_lag1.columns = ['MKT_Rets_Lag1']
MARKET_lag3 = MARKET_RETURNS.shift(3)
MARKET_lag3.columns = ['MKT_Rets_Lag3']
MARKET_lag6 = MARKET_RETURNS.shift(6)
MARKET_lag6.columns = ['MKT_Rets_Lag6']
```

```
MARKET_lag9 = MARKET_RETURNS.shift(9)
MARKET_lag9.columns = ['MKT_Rets_Lag9']
MARKET_lag12 = MARKET_RETURNS.shift(12)
MARKET_lag12.columns = ['MKT_Rets_Lag12']
MARKET_Lag24 = MARKET_RETURNS.shift(24)
MARKET_Lag24.columns = ['MKT_Rets_Lag24']
```

[ ]:
```
corporate_bond_yields.head(3)
```

[ ]:
```
            BAA_Yield  AAA_Yield  dsspread
1986-01-02     0.1138     0.0992    0.0146
1986-01-03     0.1135     0.0992    0.0143
1986-01-06     0.1136     0.0994    0.0142
```

[ ]:
```
corporate_bonds = corporate_bond_yields.resample('M').last()
```

[ ]:
```
consumer_sentiment.loc['2014-01-31']
```

[ ]:
```
Index   -0.015758
Name: 2014-01-31 00:00:00, dtype: float64
```

[ ]:
```
EPRATIO.loc['2014-01-31']
```

[ ]:
```
EPRATIO   -0.811184
Name: 2014-01-31 00:00:00, dtype: float64
```

[ ]:
```
TB3_Rate_Lag1 = interest_rates[['3-Month']].shift(1)
TB3_Rate_Lag1.columns = ['Three_M_TBill']
tsspread_lag1 = interest_rates[['term_spread']].shift(1)
dsspread_lag1 = corporate_bonds[['dsspread']].shift(1)
epratio_lag1 = EPRATIO.shift(1)
consumer_sentiment_lag1 = consumer_sentiment.shift(1)
consumer_sentiment_lag1.columns = ['UMICH_Consumer_Sentiment']
```

[ ]:
```
DATA_MATRIX = PTI_lag1
```

[ ]:
```
DATA_MATRIX = DATA_MATRIX.merge(MARKET_RETURNS,left_index=True,right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(MARKET_lag1, left_index=True, right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(MARKET_lag3, left_index=True, right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(MARKET_lag6, left_index=True, right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(MARKET_lag12, left_index=True, right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(MARKET_Lag24, left_index=True, right_index=True)
# Adding Macroeconomic and other Important Financial Variables

DATA_MATRIX = DATA_MATRIX.merge(TB3_Rate_Lag1, left_index=True,␣
 ↪right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(tsspread_lag1, left_index=True,␣
 ↪right_index=True)
```

```
DATA_MATRIX = DATA_MATRIX.merge(dsspread_lag1, left_index=True,␣
↪right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(epratio_lag1, left_index=True, right_index=True)
DATA_MATRIX = DATA_MATRIX.merge(consumer_sentiment_lag1, left_index=True,␣
↪right_index=True)
```

[ ]:
```
DATA_MATRIX.corr()
```

[ ]:
```
                          Z_PTI_Index  MKT_RETS  MKT_Rets_Lag1  MKT_Rets_Lag3  \
Z_PTI_Index                  1.000000  0.139415       0.050648      -0.005509
MKT_RETS                     0.139415  1.000000      -0.180238      -0.029286
MKT_Rets_Lag1                0.050648 -0.180238       1.000000      -0.108588
MKT_Rets_Lag3               -0.005509 -0.029286      -0.108588       1.000000
MKT_Rets_Lag6               -0.081762 -0.059030       0.001751      -0.059068
MKT_Rets_Lag12               0.019241 -0.144078      -0.011287       0.005561
MKT_Rets_Lag24               0.085606  0.104204       0.062601      -0.104718
Three_M_TBill               -0.164159 -0.062949      -0.065323      -0.088905
term_spread                  0.163823 -0.053243      -0.010548       0.017508
dsspread                     0.049316 -0.132057      -0.247170      -0.199573
EPRATIO                     -0.000801 -0.068937      -0.048703      -0.035818
UMICH_Consumer_Sentiment    -0.060761 -0.048513       0.422483      -0.203663

                          MKT_Rets_Lag6  MKT_Rets_Lag12  MKT_Rets_Lag24  \
Z_PTI_Index                   -0.081762        0.019241        0.085606
MKT_RETS                      -0.059030       -0.144078        0.104204
MKT_Rets_Lag1                  0.001751       -0.011287        0.062601
MKT_Rets_Lag3                 -0.059068        0.005561       -0.104718
MKT_Rets_Lag6                  1.000000        0.009945       -0.094145
MKT_Rets_Lag12                 0.009945        1.000000       -0.166849
MKT_Rets_Lag24                -0.094145       -0.166849        1.000000
Three_M_TBill                 -0.132695       -0.050563        0.057120
term_spread                    0.068473        0.075232        0.045518
dsspread                      -0.135702       -0.038325        0.056868
EPRATIO                       -0.052872       -0.048206        0.025546
UMICH_Consumer_Sentiment      -0.193535       -0.199642        0.145631

                          Three_M_TBill  term_spread  dsspread   EPRATIO  \
Z_PTI_Index                   -0.164159     0.163823  0.049316 -0.000801
MKT_RETS                      -0.062949    -0.053243 -0.132057 -0.068937
MKT_Rets_Lag1                 -0.065323    -0.010548 -0.247170 -0.048703
MKT_Rets_Lag3                 -0.088905     0.017508 -0.199573 -0.035818
MKT_Rets_Lag6                 -0.132695     0.068473 -0.135702 -0.052872
MKT_Rets_Lag12                -0.050563     0.075232 -0.038325 -0.048206
MKT_Rets_Lag24                 0.057120     0.045518  0.056868  0.025546
Three_M_TBill                  1.000000    -0.698964  0.162009 -0.141539
term_spread                   -0.698964     1.000000 -0.285480  0.475260
dsspread                       0.162009    -0.285480  1.000000  0.050718
```

```
EPRATIO                           -0.141539     0.475260  0.050718  1.000000
UMICH_Consumer_Sentiment           0.016458     0.060068 -0.160190  0.131685

                          UMICH_Consumer_Sentiment
Z_PTI_Index                              -0.060761
MKT_RETS                                 -0.048513
MKT_Rets_Lag1                             0.422483
MKT_Rets_Lag3                            -0.203663
MKT_Rets_Lag6                            -0.193535
MKT_Rets_Lag12                           -0.199642
MKT_Rets_Lag24                            0.145631
Three_M_TBill                             0.016458
term_spread                               0.060068
dsspread                                 -0.160190
EPRATIO                                   0.131685
UMICH_Consumer_Sentiment                  1.000000
```

	Z_PTI_Index	MKT_RETS	MKT_Rets_Lag1	MKT_Rets_Lag3	MKT_Rets_Lag6	MKT_Rets_Lag12	MKT_Rets_Lag24	Three_M_TBill	term_spread	dsspread	EPRATIO	UMICH_Consumer_Sentiment
count	107	107	107	107	107	107	107	107	107	107	107	107
mean	0.0167669	0.0101577	0.00977690	0.0105956	0.0113315	0.0131050	0.0125490	0.00825327	0.0126953	0.00922699	-1.58044	-0.00218209
std	0.251608	0.0471054	0.0468672	0.0461555	0.0454044	0.0432229	0.0443014	0.00961234	0.00799	0.00239073	0.416783	0.0504526
min	-0.839286	-0.198738	-0.198738	-0.198738	-0.198738	-0.198738	-0.198738	0	-0.0051	0.0054	-2.44461	-0.194164
25%	-0.0857143	-0.0143630	-0.0143631	-0.0140764	-0.0128349	-0.0023404	-0.0052643	0.0005	0.00685	0.0071	-1.91194	-0.0237461
50%	0	0.0179656	0.0179656	0.0187079	0.0196079	0.0197749	0.0197749	0.0032	0.013	0.009	-1.56317	0.00103199
75%	0.159825	0.0344625	0.0332733	0.0332733	0.0343045	0.0343045	0.0353163	0.01555	0.0189	0.0106	-1.22999	0.0286577
max	0.846154	0.148852	0.148852	0.148852	0.148852	0.148852	0.148852	0.0423	0.0293	0.019	-0.811184	0.130097

[ ]:
```
DATA_MATRIX.columns
```

[ ]:
```
Index(['Z_PTI_Index', 'MKT_RETS', 'MKT_Rets_Lag1', 'MKT_Rets_Lag3',
       'MKT_Rets_Lag6', 'MKT_Rets_Lag12', 'MKT_Rets_Lag24', 'Three_M_TBill',
       'term_spread', 'dsspread', 'EPRATIO', 'UMICH_Consumer_Sentiment'],
      dtype='object')
```

[ ]:
```
multiple_ols = smf.ols(formula='MKT_RETS ~ Z_PTI_Index + MKT_Rets_Lag1 +␣
↪MKT_Rets_Lag3 + MKT_Rets_Lag6 + MKT_Rets_Lag12 + MKT_Rets_Lag24 +␣
↪Three_M_TBill +\
    term_spread + dsspread + EPRATIO + UMICH_Consumer_Sentiment', data =␣
↪DATA_MATRIX)
```

[ ]:
```python
results = multiple_ols.fit()
results.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:               MKT_RETS   R-squared:                       0.202
Model:                            OLS   Adj. R-squared:                  0.108
Method:                 Least Squares   F-statistic:                     2.144
Date:                Thu, 23 Feb 2023   Prob (F-statistic):             0.0243
Time:                        16:36:05   Log-Likelihood:                 184.56
No. Observations:                 105   AIC:                            -345.1
Df Residuals:                      93   BIC:                            -313.3
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
============
                              coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------------
Intercept                   0.1384      0.044      3.170      0.002       0.052
0.225
Z_PTI_Index                 0.0077      0.004      1.724      0.088      -0.001
0.017
MKT_Rets_Lag1              -0.2938      0.108     -2.712      0.008      -0.509
-0.079
MKT_Rets_Lag3              -0.1554      0.102     -1.526      0.130      -0.358
0.047
MKT_Rets_Lag6              -0.0956      0.102     -0.936      0.352      -0.298
0.107
MKT_Rets_Lag12             -0.1433      0.105     -1.366      0.175      -0.352
0.065
MKT_Rets_Lag24              0.1353      0.102      1.324      0.189      -0.068
0.338
Three_M_TBill              -1.5960      0.691     -2.310      0.023      -2.968
-0.224
term_spread                -2.5181      0.986     -2.553      0.012      -4.477
-0.560
dsspread                   -6.9740      2.179     -3.200      0.002     -11.301
-2.646
EPRATIO                     0.0083      0.012      0.681      0.498      -0.016
0.032
UMICH_Consumer_Sentiment   -0.0553      0.104     -0.529      0.598      -0.263
0.152
==============================================================================
```

```
Omnibus:                       12.325   Durbin-Watson:                   1.902
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               18.180
Skew:                          -0.545   Prob(JB):                     0.000113
Kurtosis:                       4.722   Cond. No.                         988.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[ ]:
```python
table = pd.DataFrame({'Beta_Coefficients': results.params,'SE':results.
 ↪bse,'t_stat':results.tvalues,'pval':results.pvalues})
table = np.round(table, 5)
```

[ ]:
```python
predictions = pd.DataFrame(results.fittedvalues,␣
 ↪columns=['Forecasted_Market_Return'])
observed_values  = DATA_MATRIX[['MKT_RETS']]
dependendent_variables = predictions.join(observed_values, how = 'inner')
errors = pd.DataFrame(results.resid,columns=['Residuals'])
```

[ ]:
```python
dependendent_variables.corr()
```

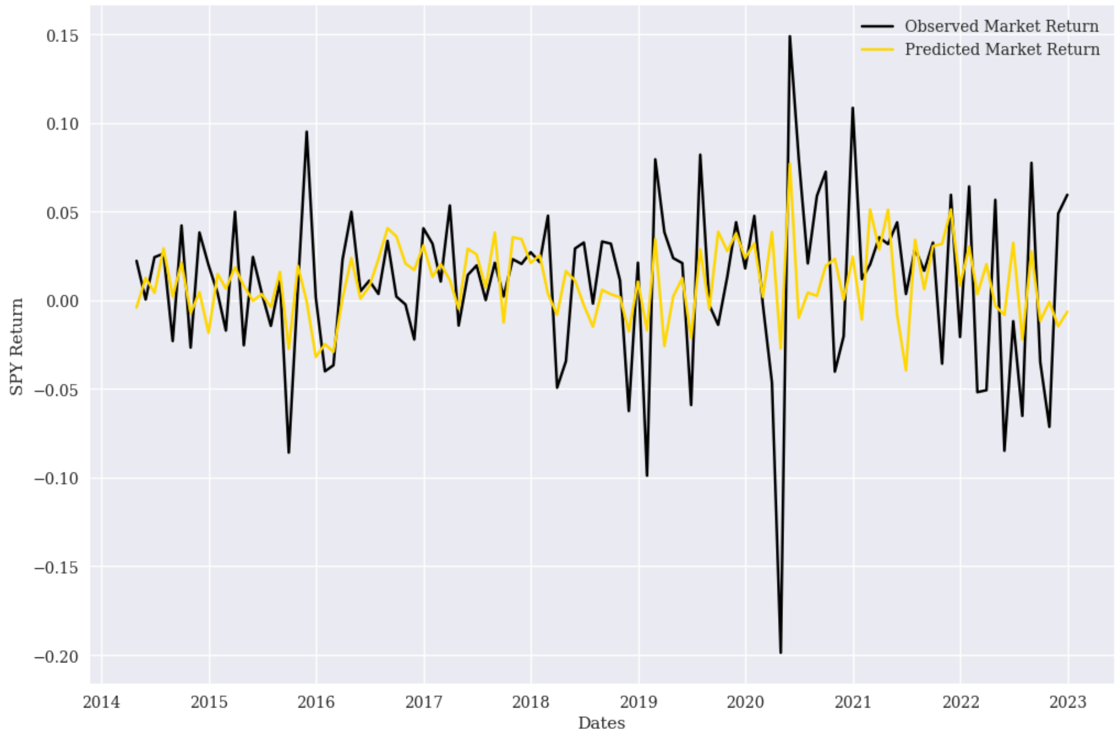
[ ]:

	Forecasted_Market_Return	MKT_RETS
Forecasted_Market_Return	1.000000	0.449751
MKT_RETS	0.449751	1.000000

[ ]:
```python
txt = 'Correlation Coefficient Between Predicted returns and Observed Returns␣
 ↪is equal to 44.9% on a monthly basis'
```

[ ]:
```python
plt.figure(figsize=(12,8))

plt.plot(dependendent_variables.MKT_RETS, color = 'black',label = 'Observed␣
 ↪Market Return')
plt.plot(dependendent_variables['Forecasted_Market_Return'], color =␣
 ↪'gold',label ='Predicted Market Return')
plt.figtext(x = .5,y = 0.01, s = txt, fontsize = '12', wrap = True,␣
 ↪horizontalalignment = 'center')
plt.ylabel('SPY Return')
plt.xlabel('Dates')
plt.legend(loc = 0)
```

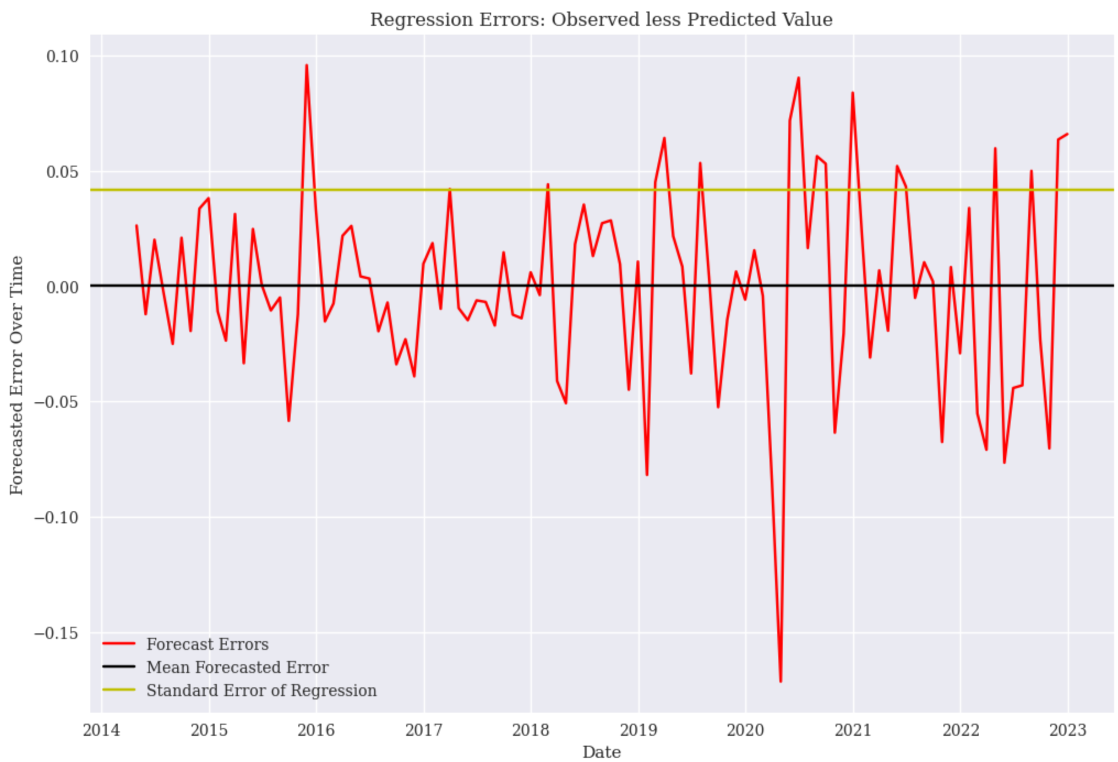
[ ]:
```
<matplotlib.legend.Legend at 0x25107606ac0>
```



Correlation Coefficient Between Predicted returns and Observed Returns is equal to 44.9%

[ ]:
```
plt.figure(figsize=(12,8))
plt.plot(errors, label = 'Forecast Errors', c = 'red')
plt.axhline(0,label = 'Mean Forecasted Error', c= 'black')
plt.axhline(errors.std().values, label = 'Standard Error of Regression', c =␣
 ↪'y')
plt.title('Regression Errors: Observed less Predicted Value')
plt.xlabel('Date')
plt.ylabel('Forecasted Error Over Time')

plt.legend(loc = 0)
```

[ ]: <matplotlib.legend.Legend at 0x2510569f940>



4 Interpretation of Multiple Linear Regression Equation

	Beta_Coefficients	SE	t_stat	pval
Intercept	0.13837	0.04365	3.17004	0.00206
Z_PTI_Index	0.00769	0.00446	1.72353	0.08812
MKT_Rets_Lag1	-0.29384	0.10833	-2.71235	0.00796
MKT_Rets_Lag3	-0.15536	0.1018	-1.52605	0.13039
MKT_Rets_Lag6	-0.09559	0.10216	-0.93573	0.35184
MKT_Rets_Lag12	-0.14332	0.10492	-1.3659	0.17526
MKT_Rets_Lag24	0.13525	0.10218	1.32368	0.18885
Three_M_TBill	-1.59604	0.691	-2.30975	0.02311
term_spread	-2.51809	0.98628	-2.55312	0.0123
dsspread	-6.97395	2.17922	-3.20021	0.00188
EPRATIO	0.00826	0.01212	0.68095	0.4976
UMICH_Consumer_Sentiment	-0.05526	0.10448	-0.52889	0.59814

- With a *p-value* of 0.08812 we have fairly strong evidence against the null hypothesis that a **Z_score Coefficient on the lagged 3-month PTI Index** is statistically significant. I am happy to have found economic signficance in this regression while controlling for many different variables. Holding other factors fixed, we can say that a one standard deviation increase in the lagged 3-month **Politican Trading Index** will predict $.00769 \cdot 0.25 = 0.0019225$ or about 20 basis point increase in the 1-month market returns.

- This provides solid evidence that elevated levels of politicians trading stocks does indicate causal effects on future market returns.