Applying Linear Regression To Pairs Trading

Pramod Mitikiri, Aneesh Durai, Raymond Xia

ABSTRACT

In the following research, we will analyze the effects of pairs trading (multiple companies across multiple industries) excluding the profitability of such strategies. Rather, we will analyze various risk measures across all different pairings of stocks within their own respective industry across multiple industries. With this methodology, we will develop a pairs trading strategy that performs better than the index.

INTRODUCTION

Pairs trading (or statistical arbitrage) is a popular trading strategy used by investors to make profits by simultaneously buying and selling a set number of highly correlated securities. For our purposes, we will simplify pairs trading by simultaneously buying and selling two highly correlated stocks (Stock A and B) within the same industry to ensure both stocks have similar fundamental characteristics. Pairs are created with the understanding that Stock A would perform well and Stock B would not perform well from the same sequence of events in the industry, which means to profit from this strategy, you would have a long position for Stock A and to have a short position for Stock B, hence taking advantage of both companies' relative price movements.

Investors calculate the mean ratio between two stocks (the measure doesn't fluctuate as much as individual stock prices move), and when there is a significant deviation from a stock's price to the mean ratio, investors will take advantage of the opportunity and make profit when the stock prices eventually fall back to the mean ratio. However, investors will not always make a profit with pairs trading as the strategy has its set of limitations. If the strategy isn't handled properly, then an unexpected change of the relationship between both stocks within the pair would cause significant losses. If there was a dynamic market-wide fluctuation that caused both stocks in the pair to move simultaneously (and not simply return to their mean ratio after a period of time), then it may be difficult to interpret the relative change and cause investors to miss their profit.

With all this in mind, we will analyze the effects of pairs trading within three industries: healthcare, tech, and energy through statistical analysis, specifically comparing correlation measures, cointegration measures, betas and Sharpe ratios. In doing so, we can learn more information about the distinctive natures of the three industries, and then incorporate this information into our pairs trading strategy of a given portfolio
(which in theory should outperform the index as we are using more information to make well-educated decisions).

REVIEW OF EXISTING LITERATURE

The literature, "Research on Modern Implications of **Pairs Trading**", analyzes modern implications of using a pairs trading strategy to find relationships between profit and different variables associated with stock selections in pairs trading. They explain statistical arbitrage takes advantage of mis-pricings in the financial market by analyzing stock relative values rather than true

stock values. Using the distance method (non-parametric) for pairs trading, Zhang tracks the distance between two normalized stock prices and stock volatility (at certain thresholds), and when the stocks are misplaced, we buy the underpriced stock and sell the overpriced stock. However, she mentions that pairs trading has limitations with price-level divergence and its lack of its forecasting ability (by expected holding period) but has an advantage in being immune to macroeconomic changes.

In her analysis, she simulates multiple trades with the pairs-trading distance and plots the historical prices and historical ratios (analyzing the number of standard deviations from the mean) first with the first pair of stocks (from 1980s): Nordstrom (JWN) and GAP (GPS), and indicates when the ratio surpasses a multiple of a standard deviation to start a trade (ends when it reaches the mean). She calculates the total profit from the total trade, incorporating a $0.001 transactional cost (but still provide analysis for either no transactional cost or $0.005 transactional cost). She repeats this example with a pair of stocks (from the present, 2000s): Coca-Cola (KO) and Pepsi (PEP), and deduces that the 1980s pair's price ratio is more volatile while the 2000s pair's price is closer to the mean (so more instances of possible trading opportunities with smaller thresholds).

With a closer analysis, Coca-Cola and Pepsi have a divergence trend (Pepsi increases steadily while Coca-Cola gradually decreases), but the mean ratio is still reverted from 2000-2006. Zhang notes that different results may occur depending on the data referenced, number of years referenced, and the future directions of future companies.

She then does a simulation study, examining relationships with profit and stock selections through vector autoregression (which helps generate stock picks with specific properties) that uses the following variables: psi, beta_1, rho, beta_0, sigma^2, p; correlation with each other, slope of price (change in stock price over time), correlation with time, intercept, distribution spread, and trading cost respectively. Plotting only psi and beta_1 first to determine the relationship results in the conclusion that when there isn't any psi or beta_1, the data is heavily scattered. For this particular pair of stocks, having a high psi and no clear trend results in the pair being the most correlated.

A trading strategy was implemented with this methodology: 36 possible combinations of different psi, beta_1, and k (number of standard deviations that set the trading threshold) with 1000 simulations per combination for an average product. Zhang concludes that a closely-correlated pair of stocks, stocks with a decreasing (price) trend, trading with lower thresholds make more profit on average.

Limitations to the study include not including mergers and acquisitions, technological innovations, and infrastructure changes apart from theoretical assumptions not respective to reality as well as no limit to the initial budget for our portfolio. Zhang encourages to repeat her study with cointegration measures (analyzing linear combinations of two time series with a long-term equilibrium) instead of using just the distance method knowing the price ratio is static (content with little variance) over time but notes cointegration measures have their own limitations too (real stocks have non stationary factor spreads).

DATA

We analyze four stocks within three industries (tech, health, and energy). We have for tech: Apple (AAPL), Microsoft (MFST), Meta (META), and Google (GOOG), for health:

Merck & Co. (MRK), Johnson & Johnson (JNJ), Pfizer Inc. (PFE), and Abbott Laboratories (ABT), and for energy: NextEra Energy Inc (NEE), Chevron Corporation (CVX), Exxon Mobil Corp (XOM), and Shell PLC (SHEL). There are 6 possible pairings for each industry, and for each pairing with each industry, we find the corresponding correlations, price ratios, betas, and the cointegration test through each pairings' conjoined time series of the closing price over the last three months. This information helps us understand the relationship between the given stocks and if there are optimal pairings within each industry.

MODELING

As for the trading strategies, we first do Ordinary Least Squares (OLS) regression (in line with the distance method) to find the best possible fit for each of the three industries. With that in mind, we take these coefficients and we create the best possible linear diophantine equation for each industry (this would simulate the constant price mean), or in the form aX + bY = C. If the sum increases due to a movement in either X or Y and it surpasses C, we would need to adjust the strategy with coefficients -a, -b to simulate returning to the constant C (and that being considered as a trade similar to the literature). We repeat this process for all three industries, and the given pair highlighted in each industry would be the most profitable pair by our trading strategy.

RESULTS

Looking at the results from each industry, we find that we can make a profit in the tech industry with AAPL and GOOG (~ +$0.08), while we make a loss in the Health Industry with MRK and PFE (~ -$0.17) and the Energy Industry with NEE and SHEL (~ -$0.62). Was the strategy successful in comparison to our initial thoughts? No, the strategy would need to be refined even more to consider variables outside analyzing closing prices. For instance, we can use a GB Model rather than a linear model to simulate the market better and to consider standard deviations of the total amount of the respective pair increasing or decreasing about the mean (due to limiting our data to three months, such changes would be harder to implement), and the strategy would need to be altered to consider the lag time in placing a trade itself (you wouldn't be able to instantaneously place an order at a given period of time, there would be some lag which can cause a shift in profit as a result). In addition, we are aware of the effects of noise on our data analysis. For example, noise itself may contribute to a certain percentage of cointegration even in the absence of actual relationship between equities. Therefore, a possible modification is to set a higher threshold for cointegration test.

CONCLUSION

Overall, the strategy did not work as we have intended so, but it has been a deep analysis in how different pairings from different industries interact with respect to pairs trading. If we were to conduct this strategy again, we would try randomizing values to create a training and test set to validate our model. We would also attempt to use various other features that are relevant to pairs trading.

In [1]:
```
import numpy as np
import pandas as pd
import yfinance as yf
import statsmodels.api as sm
from statsmodels.tsa.stattools import coint
from sklearn import linear_model
```

Cointegration test

In [2]:
```
def coInt(df, stock1, stock2):
    # run regression of AAPL on GOOG
    res1 = sm.OLS(df[stock1], sm.add_constant(df[stock2])).fit()
    resid = res1.resid
    _, pvalue, _ = coint(df_tech['AAPL'], df_tech['GOOG'])
    return pvalue
```

Tech industry data is downloaded from yfinance and then stored in a dataframe. Similar treatment is replicated for health and energy industries in later sections.

In [3]:
```
# Tech Industries

df_aapl = yf.download("AAPL", start="2023-01-01", end="2023-04-30")
df_msft = yf.download("MSFT", start="2023-01-01", end="2023-04-30")
df_meta = yf.download("META", start="2023-01-01", end="2023-04-30")
df_goog = yf.download("GOOG", start="2023-01-01", end="2023-04-30")
df_tech = pd.concat([df_aapl["Close"], df_msft["Close"], df_meta["Clo
df_tech = df_tech.diff()
df_tech.replace([np.inf, -np.inf], np.nan, inplace=True)
df_tech = df_tech.dropna()
df_tech.columns = ['AAPL', 'MSFT', 'META', 'GOOG']

df_tech
```

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

Out[3]:

Date	AAPL	MSFT	META	GOOG
2023-01-04	1.290001	-10.479996	2.630005	-0.989998
2023-01-05	-1.340004	-6.790009	-0.430000	-1.940002
2023-01-06	4.599998	2.619995	3.080002	1.390007
2023-01-09	0.529999	2.190002	-0.550003	0.639999
2023-01-10	0.580002	1.730011	3.520004	0.439995
...	...	...	...	...
2023-04-24	0.309998	-3.990021	-0.100006	0.869995
2023-04-25	-1.559998	-6.349976	-5.239990	-2.169998
2023-04-26	-0.010010	19.949982	1.849991	-0.160004
2023-04-27	4.650009	9.459991	29.160004	3.920006
2023-04-28	1.269989	2.430023	1.760010	-0.150002

80 rows × 4 columns

In [4]:

```
# Correlations of All Possible Pairings

df_tech['r_AAPL_MSFT'] = df_tech['AAPL'].corr(df_tech['MSFT'])
df_tech['r_AAPL_META'] = df_tech['AAPL'].corr(df_tech['META'])
df_tech['r_AAPL_GOOG'] = df_tech['AAPL'].corr(df_tech['GOOG'])
df_tech['r_MSFT_META'] = df_tech['MSFT'].corr(df_tech['META'])
df_tech['r_MSFT_GOOG'] = df_tech['MSFT'].corr(df_tech['GOOG'])
df_tech['r_META_GOOG'] = df_tech['META'].corr(df_tech['GOOG'])

# Price Ratios of All Possible Pairings

df_tech['ratio_AAPL_MSFT'] = df_tech['AAPL']/(df_tech['MSFT'])
df_tech['ratio_AAPL_META'] = df_tech['AAPL']/(df_tech['META'])
df_tech['ratio_AAPL_GOOG'] = df_tech['AAPL']/(df_tech['GOOG'])
df_tech['ratio_MSFT_META'] = df_tech['MSFT']/(df_tech['META'])
df_tech['ratio_MSFT_GOOG'] = df_tech['MSFT']/(df_tech['GOOG'])
df_tech['ratio_META_GOOG'] = df_tech['META']/(df_tech['GOOG'])

# Betas of all Possible Pairings

df_tech['b_AAPL_MSFT'] = np.cov(df_tech['AAPL'], df_tech['MSFT'])[0][
df_tech['b_AAPL_META'] = np.cov(df_tech['AAPL'], df_tech['META'])[0][
df_tech['b_AAPL_GOOG'] = np.cov(df_tech['AAPL'], df_tech['GOOG'])[0][
df_tech['b_MSFT_META'] = np.cov(df_tech['MSFT'], df_tech['META'])[0][
df_tech['b_MSFT_GOOG'] = np.cov(df_tech['MSFT'], df_tech['GOOG'])[0][
df_tech['b_META_GOOG'] = np.cov(df_tech['META'], df_tech['GOOG'])[0][

# CoIngration Test of all Possible Pairings

df_tech['coInt_AAPL_MSFT'] = coInt(df_tech, 'AAPL', 'MSFT')
df_tech['coInt_AAPL_META'] = coInt(df_tech, 'AAPL', 'META')
df_tech['coInt_AAPL_GOOG'] = coInt(df_tech, 'AAPL', 'GOOG')
df_tech['coInt_MSFT_META'] = coInt(df_tech, 'MSFT', 'META')
df_tech['coInt_MSFT_GOOG'] = coInt(df_tech, 'MSFT', 'GOOG')
df_tech['coInt_META_GOOG'] = coInt(df_tech, 'META', 'GOOG')

df_tech
```

```
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[4]:

Date	AAPL	MSFT	META	GOOG	r_AAPL_MSFT	r_AAPL_META	r_AAPL_GOOG
2023-01-04	1.290001	-10.479996	2.630005	-0.989998	0.565244	0.632555	0.661596
2023-01-05	-1.340004	-6.790009	-0.430000	-1.940002	0.565244	0.632555	0.661596
2023-01-06	4.599998	2.619995	3.080002	1.390007	0.565244	0.632555	0.661596
2023-01-09	0.529999	2.190002	-0.550003	0.639999	0.565244	0.632555	0.661596
2023-01-10	0.580002	1.730011	3.520004	0.439995	0.565244	0.632555	0.661596
...	...	...	...	...	...	...	...
2023-04-24	0.309998	-3.990021	-0.100006	0.869995	0.565244	0.632555	0.661596
2023-04-25	-1.559998	-6.349976	-5.239990	-2.169998	0.565244	0.632555	0.661596
2023-04-26	-0.010010	19.949982	1.849991	-0.160004	0.565244	0.632555	0.661596
2023-04-27	4.650009	9.459991	29.160004	3.920006	0.565244	0.632555	0.661596
2023-04-28	1.269989	2.430023	1.760010	-0.150002	0.565244	0.632555	0.661596

80 rows × 28 columns

OLS Regression - Tech Industry

In [5]:

```
x_tech = df_tech[['AAPL', 'b_AAPL_MSFT', 'r_AAPL_MSFT']]
y_tech = (df_tech['MSFT'])

regr_tech = linear_model.LinearRegression()
regr_tech.fit(x_tech, y_tech)

print('Intercept: \n', regr_tech.intercept_)
print('Coefficients: \n', regr_tech.coef_)

# with statsmodels
x_tech = sm.add_constant(x_tech) # adding a constant

model_tech = sm.OLS(y_tech, x_tech).fit()
predictions_tech = model_tech.predict(x_tech)

print_model_tech = model_tech.summary()
print(print_model_tech)
```

```
Intercept:
 0.06471377264417377
Coefficients:
 [ 1.40109652e+00 -4.62415730e-34  1.84966292e-33]
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    MSFT   R-squared:
0.320
Model:                             OLS   Adj. R-squared:
0.311
Method:                  Least Squares   F-statistic:
36.62
Date:                Fri, 05 May 2023   Prob (F-statistic):
4.70e-08
Time:                        21:08:34   Log-Likelihood:
-229.63
No. Observations:                  80   AIC:
463.3
Df Residuals:                      78   BIC:
468.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
AAPL            1.4011      0.232      6.052      0.000       0.940
1.862
b_AAPL_MSFT     0.0401      0.310      0.129      0.897      -0.577
0.657
r_AAPL_MSFT     0.0981      0.759      0.129      0.897      -1.412
1.608
==============================================================================
=======
Omnibus:                       22.863   Durbin-Watson:
1.792
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
103.038
Skew:                           0.620   Prob(JB):
4.22e-23
Kurtosis:                       8.420   Cond. No.
9.62e+16
==============================================================================
=======

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The smallest eigenvalue is 4.06e-32. This might indicate that there
are
strong multicollinearity problems or that the design matrix is singula
r.
```

```
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Trading Strategy - Tech Industry

In [6]:
```
###Trading Strategy

def price_calc(df, stock1, stock2, regr):
    b_str = 'b_' + stock1 + '_' + stock2
    r_str = 'r_' + stock1 + '_' + stock2
    coef = regr.coef_
    inte = regr.intercept_
    price = regr.intercept_ + df[stock1] * (coef[0] ** 1) + df[b_str]
    return price - df[stock2]

price_calc(df_tech, 'AAPL', 'META', regr_tech)
```

Out[6]:
```
Date
2023-01-04    -0.757875
2023-01-05    -1.382761
2023-01-06     3.429754
2023-01-09     1.357296
2023-01-10    -2.642652
                 ...
2023-04-24     0.599056
2023-04-25     3.118997
2023-04-26    -1.799302
2023-04-27   -22.580178
2023-04-28     0.084081
Length: 80, dtype: float64
```

Healthcare Industry

In [7]:
```
# Healthcare Industries
df_mrk = yf.download("MRK", start="2023-01-01", end="2023-04-30")
df_jnj = yf.download("JNJ", start="2023-01-01", end="2023-04-30")
df_pfe = yf.download("PFE", start="2023-01-01", end="2023-04-30")
df_abt = yf.download("ABT", start="2023-01-01", end="2023-04-30")
df_health = pd.concat([df_mrk["Close"], df_jnj["Close"], df_pfe["Clos
df_health = df_health.diff()
df_health.replace([np.inf, -np.inf], np.nan, inplace=True)
df_health = df_health.dropna()
df_health.columns = ['MRK', 'JNJ', 'PFE', 'ABT']
```

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

In [8]:

```
# Correlations for All Possible Pairings

df_health['r_MRK_JNJ'] = df_health['MRK'].corr(df_health['JNJ'])
df_health['r_MRK_PFE'] = df_health['MRK'].corr(df_health['PFE'])
df_health['r_MRK_ABT'] = df_health['MRK'].corr(df_health['ABT'])
df_health['r_JNJ_PFE'] = df_health['JNJ'].corr(df_health['PFE'])
df_health['r_JNJ_ABT'] = df_health['JNJ'].corr(df_health['ABT'])
df_health['r_PFE_ABT'] = df_health['PFE'].corr(df_health['ABT'])

# Price Ratios for All Possible Pairings

df_health['ratio_MRK_JNJ'] = df_health['MRK']/(df_health['JNJ'])
df_health['ratio_MRK_JNJ'] = df_health['MRK']/(df_health['JNJ'])
df_health['ratio_MRK_ABT'] = df_health['MRK']/(df_health['ABT'])
df_health['ratio_JNJ_PFE'] = df_health['JNJ']/(df_health['PFE'])
df_health['ratio_JNJ_ABT'] = df_health['JNJ']/(df_health['ABT'])
df_health['ratio_PFE_ABT'] = df_health['PFE']/(df_health['ABT'])

# Betas for All Possible Pairings

df_health['b_MRK_JNJ'] = np.cov(df_health['MRK'], df_health['JNJ'])[0
df_health['b_MRK_PFE'] = np.cov(df_health['MRK'], df_health['PFE'])[0
df_health['b_MRK_ABT'] = np.cov(df_health['MRK'], df_health['ABT'])[0
df_health['b_JNJ_PFE'] = np.cov(df_health['JNJ'], df_health['PFE'])[0
df_health['b_JNJ_ABT'] = np.cov(df_health['JNJ'], df_health['ABT'])[0
df_health['b_PFE_ABT'] = np.cov(df_health['PFE'], df_health['ABT'])[0

# CoIngration Test of all Possible Pairings

df_health['coInt_MRK_JNJ'] = coInt(df_health, 'MRK', 'JNJ')
df_health['coInt_MRK_PFE'] = coInt(df_health, 'MRK', 'PFE')
df_health['coInt_MRK_ABT'] = coInt(df_health, 'MRK', 'ABT')
df_health['coInt_JNJ_PFE'] = coInt(df_health, 'JNJ', 'PFE')
df_health['coInt_JNJ_ABT'] = coInt(df_health, 'JNJ', 'ABT')
df_health['coInt_PFE_ABT'] = coInt(df_health, 'PFE', 'ABT')

df_health
```

```
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[8]:

Date	MRK	JNJ	PFE	ABT	r_MRK_JNJ	r_MRK_PFE	r_MRK_ABT	r_JNJ_PF
2023-01-04	0.940002	1.940002	-1.129997	1.629997	0.509545	0.482223	0.121208	0.54937
2023-01-05	1.559998	-1.330002	-0.470001	-0.409996	0.509545	0.482223	0.121208	0.54937
2023-01-06	1.199997	1.449997	1.259998	1.529999	0.509545	0.482223	0.121208	0.54937
2023-01-09	-4.459999	-4.669998	-2.529999	-0.180000	0.509545	0.482223	0.121208	0.54937
2023-01-10	0.430000	-0.419998	-0.770000	1.699997	0.509545	0.482223	0.121208	0.54937
...	...	...	...	...	...	...	...	
2023-04-24	0.570000	0.989990	-0.299999	-1.260002	0.509545	0.482223	0.121208	0.54937
2023-04-25	0.589996	1.500000	-0.579998	-0.430000	0.509545	0.482223	0.121208	0.54937
2023-04-26	-3.099998	-2.559998	-0.700001	-1.220001	0.509545	0.482223	0.121208	0.54937
2023-04-27	1.730003	0.380005	0.110001	0.750000	0.509545	0.482223	0.121208	0.54937
2023-04-28	0.309998	0.699997	0.149998	0.970001	0.509545	0.482223	0.121208	0.54937

80 rows × 27 columns

OLS Regression - Healthcare Industry

In [9]:

```
x_health = df_health[['MRK', 'b_MRK_JNJ', 'r_MRK_JNJ']]
y_health = (df_health['JNJ'])

regr_health = linear_model.LinearRegression()
regr_health.fit(x_health, y_health)

print('Intercept: \n', regr_health.intercept_)
print('Coefficients: \n', regr_health.coef_)

# with statsmodels
x_health = sm.add_constant(x_health) # adding a constant

model_health = sm.OLS(y_health, x_health).fit()
predictions_health = model_health.predict(x_health)

print_model_health = model_health.summary()
print(print_model_health)
```

```
Intercept:
 -0.21381007473382674
Coefficients:
 [6.03879764e-01 0.00000000e+00 5.51528396e-34]
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    JNJ   R-squared:
0.260
Model:                            OLS   Adj. R-squared:
0.250
Method:                 Least Squares   F-statistic:
27.35
Date:                Fri, 05 May 2023   Prob (F-statistic):
1.38e-06
Time:                        21:08:36   Log-Likelihood:
-147.99
No. Observations:                  80   AIC:
300.0
Df Residuals:                      78   BIC:
304.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
MRK            0.6039      0.115      5.230      0.000       0.374
0.834
b_MRK_JNJ     -0.2072      0.169     -1.226      0.224      -0.544
0.129
r_MRK_JNJ     -0.2425      0.198     -1.226      0.224      -0.636
0.151
==============================================================================
=======
Omnibus:                       18.211   Durbin-Watson:
2.047
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
68.923
Skew:                          -0.466   Prob(JB):
1.08e-15
Kurtosis:                       7.451   Cond. No.
5.41e+16
==============================================================================
=======

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The smallest eigenvalue is 6.24e-32. This might indicate that there
are
strong multicollinearity problems or that the design matrix is singula
r.
```

```
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Trading Strategy - Healthcare Industry

In [10]:
```
price_calc(df_health, 'MRK', 'PFE', regr_health)
```

Out[10]:
```
Date
2023-01-04    1.483836
2023-01-05    1.198242
2023-01-06   -0.749155
2023-01-09   -0.377114
2023-01-10    0.815859
                ...
2023-04-24    0.430400
2023-04-25    0.722475
2023-04-26   -1.385836
2023-04-27    0.720903
2023-04-28   -0.176607
Length: 80, dtype: float64
```

Energy Industry

In [11]:
```
# Energy Industries

df_nee = yf.download("NEE", start="2023-01-01", end="2023-04-30")
df_cvx = yf.download("CVX", start="2023-01-01", end="2023-04-30")
df_xom = yf.download("XOM", start="2023-01-01", end="2023-04-30")
df_shel = yf.download("SHEL", start="2023-01-01", end="2023-04-30")
df_energy = pd.concat([df_nee["Close"], df_cvx["Close"], df_xom["Clos
df_energy = df_energy.diff()
df_energy.replace([np.inf, -np.inf], np.nan, inplace=True)
df_energy = df_energy.dropna()
df_energy.columns = ['NEE', 'CVX', 'XOM', 'SHEL']
```

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

In [12]:

```
# Correlation of All Possible Pairings

df_energy['r_NEE_CVX'] = df_energy['NEE'].corr(df_energy['CVX'])
df_energy['r_NEE_XOM'] = df_energy['NEE'].corr(df_energy['XOM'])
df_energy['r_NEE_SHEL'] = df_energy['NEE'].corr(df_energy['SHEL'])
df_energy['r_CVX_XOM'] = df_energy['CVX'].corr(df_energy['XOM'])
df_energy['r_CVX_SHEL'] = df_energy['CVX'].corr(df_energy['SHEL'])
df_energy['r_XOM_SHEL'] = df_energy['XOM'].corr(df_energy['SHEL'])

# Price Ratio of All Possible Pairings

df_energy['ratio_NEE_CVX'] = df_energy['NEE']/(df_energy['CVX'])
df_energy['ratio_NEE_XOM'] = df_energy['NEE']/(df_energy['XOM'])
df_energy['ratio_NEE_SHEL'] = df_energy['NEE']/(df_energy['SHEL'])
df_energy['ratio_CVX_XOM'] = df_energy['CVX']/(df_energy['XOM'])
df_energy['ratio_CVX_SHEL'] = df_energy['CVX']/(df_energy['SHEL'])
df_energy['ratio_XOM_SHEL'] = df_energy['XOM']/(df_energy['SHEL'])

# Betas of All Possible Pairings

df_energy['b_NEE_CVX'] = np.cov(df_energy['NEE'], df_energy['CVX'])[0
df_energy['b_NEE_XOM'] = np.cov(df_energy['NEE'], df_energy['XOM'])[0
df_energy['b_NEE_SHEL'] = np.cov(df_energy['NEE'], df_energy['SHEL'])
df_energy['b_CVX_XOM'] = np.cov(df_energy['CVX'], df_energy['XOM'])[0
df_energy['b_CVX_SHEL'] = np.cov(df_energy['CVX'], df_energy['SHEL'])
df_energy['b_XOM_SHEL'] = np.cov(df_energy['XOM'], df_energy['SHEL'])

# CoIngrations of All Possible Pairings

df_energy['coInt_NEE_CVX'] = coInt(df_energy, 'NEE', 'CVX')
df_energy['coInt_NEE_XOM'] = coInt(df_energy, 'NEE', 'XOM')
df_energy['coInt_NEE_SHEL'] = coInt(df_energy, 'NEE', 'SHEL')
df_energy['coInt_CVX_XOM'] = coInt(df_energy, 'CVX', 'XOM')
df_energy['coInt_CVX_SHEL'] = coInt(df_energy, 'CVX', 'SHEL')
df_energy['coInt_XOM_SHEL'] = coInt(df_energy, 'XOM', 'SHEL')

df_energy
```

```
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Out[12]:

Date	NEE	CVX	XOM	SHEL	r_NEE_CVX	r_NEE_XOM	r_NEE_SHEL	r_CVX_X
2023-01-04	0.659996	-1.850006	0.309998	-0.549999	0.215053	0.083095	0.030785	0.826
2023-01-05	-1.860001	3.100006	2.389999	0.049999	0.215053	0.083095	0.030785	0.826
2023-01-06	1.020004	1.319992	1.320000	1.770000	0.215053	0.083095	0.030785	0.826
2023-01-09	0.419998	-1.380005	-2.059998	0.820000	0.215053	0.083095	0.030785	0.826
2023-01-10	0.470001	0.860001	1.619995	0.299999	0.215053	0.083095	0.030785	0.826
...	...	...	...	...	...	...	...	
2023-04-24	0.050003	2.360001	2.189995	0.540001	0.215053	0.083095	0.030785	0.826
2023-04-25	-1.220001	-2.489990	-1.680000	-1.290001	0.215053	0.083095	0.030785	0.826
2023-04-26	-3.750000	-3.010010	-1.070000	-0.219997	0.215053	0.083095	0.030785	0.826
2023-04-27	1.779999	0.970001	1.380005	0.189999	0.215053	0.083095	0.030785	0.826
2023-04-28	0.779999	1.630005	1.509995	1.299999	0.215053	0.083095	0.030785	0.826

80 rows × 28 columns

OLS - Energy Industry

In [13]:

```
x_energy = df_energy[['CVX', 'b_CVX_XOM', 'r_CVX_XOM']]
y_energy = (df_energy['XOM'])

regr_energy = linear_model.LinearRegression()
regr_energy.fit(x_energy, y_energy)

print('Intercept: \n', regr_energy.intercept_)
print('Coefficients: \n', regr_energy.coef_)

# with statsmodels
x_energy = sm.add_constant(x_energy) # adding a constant

model_energy = sm.OLS(y_energy, x_energy).fit()
predictions_energy = model_energy.predict(x_energy)

print_model_energy = model_energy.summary()
print(print_model_energy)
```

```
Intercept:
 0.18954224271463535
Coefficients:
 [6.16152118e-01 0.00000000e+00 2.58477542e-33]
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    XOM   R-squared:
0.683
Model:                            OLS   Adj. R-squared:
0.679
Method:                 Least Squares   F-statistic:
168.1
Date:                Fri, 05 May 2023   Prob (F-statistic):
3.74e-21
Time:                        21:08:37   Log-Likelihood:
-121.36
No. Observations:                  80   AIC:
246.7
Df Residuals:                      78   BIC:
251.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
CVX            0.6162      0.048     12.965      0.000       0.522
0.711
b_CVX_XOM      0.1095      0.072      1.517      0.133      -0.034
0.253
r_CVX_XOM      0.0806      0.053      1.517      0.133      -0.025
0.186
==============================================================================
=======
Omnibus:                        0.964   Durbin-Watson:
1.977
Prob(Omnibus):                  0.618   Jarque-Bera (JB):
0.790
Skew:                           0.243   Prob(JB):
0.674
Kurtosis:                       2.960   Cond. No.
3.21e+16
==============================================================================
=======

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The smallest eigenvalue is 5.38e-31. This might indicate that there
are
strong multicollinearity problems or that the design matrix is singula
r.
```

```
/Users/aneesh/opt/anaconda3/lib/python3.9/site-packages/statsmodels/ts
a/tsatools.py:142: FutureWarning: In a future version of pandas all arg
uments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Trading Strategy - Energy Industry

In [14]:
```
price_calc(df_energy, 'NEE', 'SHEL', regr_energy)
```

Out[14]:
```
Date
2023-01-04    1.146199
2023-01-05   -1.006500
2023-01-06   -0.951980
2023-01-09   -0.371675
2023-01-10    0.179135
                ...
2023-04-24   -0.319649
2023-04-25    0.727837
2023-04-26   -1.901031
2023-04-27    1.096294
2023-04-28   -0.629859
Length: 80, dtype: float64
```