Final_Code

May 13, 2025

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

EXPLORATORY DATA ANALYSIS BEGIN

[2]:
```
#Apple stock dataset
file_apple = "apple_stock.csv"
apple_df = pd.read_csv(file_apple)
apple_df
```

[2]:

	Unnamed: 0	Adj Close	Close	High	Low	Open	Volume
0	1980-12-12	0.098834	0.128348	0.128906	0.128348	0.128348	469033600
1	1980-12-15	0.093678	0.121652	0.122210	0.121652	0.122210	175884800
2	1980-12-16	0.086802	0.112723	0.113281	0.112723	0.113281	105728000
3	1980-12-17	0.088951	0.115513	0.116071	0.115513	0.115513	86441600
4	1980-12-18	0.091530	0.118862	0.119420	0.118862	0.118862	73449600
…	…	…	…	…	…	…	…
11102	2024-12-27	255.589996	255.589996	258.700012	253.059998	257.829987	42355300
11103	2024-12-30	252.199997	252.199997	253.500000	250.750000	252.229996	35557500
11104	2024-12-31	250.419998	250.419998	253.279999	249.429993	252.440002	39480700
11105	2025-01-02	243.850006	243.850006	249.100006	241.820007	248.929993	55558000
11106	2025-01-03	243.860001	243.860001	244.179993	241.889999	243.369995	15135053

```
[11107 rows x 7 columns]
```

[3]:
```
#Microsoft stock dataset
file_msft = "MSFT_stock.csv"
msft_df = pd.read_csv(file_msft)
msft_df
```

[3]:

	Date	Open	High	Low	Close	Adj Close	\
0	1986-03-13	0.088542	0.101563	0.088542	0.097222	0.059827	
1	1986-03-14	0.097222	0.102431	0.097222	0.100694	0.061963	
2	1986-03-17	0.100694	0.103299	0.100694	0.102431	0.063032	
3	1986-03-18	0.102431	0.103299	0.098958	0.099826	0.061429	
4	1986-03-19	0.099826	0.100694	0.097222	0.098090	0.060361	
…	…	…	…	…	…	…	
9795	2025-01-28	434.600006	448.380005	431.380005	447.200012	447.200012	
9796	2025-01-29	446.690002	446.880005	440.399994	442.329987	442.329987	
9797	2025-01-30	418.769989	422.859985	413.160004	414.989990	414.989990	
9798	2025-01-31	418.980011	420.690002	414.910004	415.059998	415.059998	
9799	2025-02-03	411.600006	415.410004	408.660004	410.920013	410.920013	

	Volume
0	1031788800
1	308160000
2	133171200
3	67766400
4	47894400
…	…
9795	23491700
9796	23581400
9797	54586300
9798	34223400
9799	25580600

```
[9800 rows x 7 columns]
```

[4]:
```
#rename date columnn make sure it is Datetime formayt
apple_df.rename(columns={"Unnamed: 0": "date"}, inplace=True)
apple_df['date'] = pd.to_datetime(apple_df['date'])
print("Apple Table: \n\n", apple_df.head())

msft_df.rename(columns={"Date": "date"}, inplace=True)
msft_df['date'] = pd.to_datetime(msft_df['date'])
print("\n\nMicrosoft Table: \n\n", msft_df.head())
```

```
Apple Table:

         date  Adj Close     Close      High       Low      Open     Volume
0 1980-12-12   0.098834  0.128348  0.128906  0.128348  0.128348  469033600
```

```
1 1980-12-15   0.093678  0.121652  0.122210  0.121652  0.122210  175884800
2 1980-12-16   0.086802  0.112723  0.113281  0.112723  0.113281  105728000
3 1980-12-17   0.088951  0.115513  0.116071  0.115513  0.115513   86441600
4 1980-12-18   0.091530  0.118862  0.119420  0.118862  0.118862   73449600
```

Microsoft Table:

```
         date      Open      High       Low     Close  Adj Close      Volume
0 1986-03-13  0.088542  0.101563  0.088542  0.097222   0.059827  1031788800
1 1986-03-14  0.097222  0.102431  0.097222  0.100694   0.061963   308160000
2 1986-03-17  0.100694  0.103299  0.100694  0.102431   0.063032   133171200
3 1986-03-18  0.102431  0.103299  0.098958  0.099826   0.061429    67766400
4 1986-03-19  0.099826  0.100694  0.097222  0.098090   0.060361    47894400
```

[5]:
```
#creating new columns daily return (how much adj_close % changes) and
 ↪Volatility (5 day/one trading week std of daily return)
apple_df['Daily Return'] = apple_df['Adj Close'].pct_change()*100
msft_df['Daily Return'] = msft_df['Adj Close'].pct_change()*100

apple_df['Volatility_5'] = apple_df['Daily Return'].rolling(window=5).std()
msft_df['Volatility_5'] = msft_df['Daily Return'].rolling(window=5).std()

print("Apple Table: \n\n", apple_df.tail())
print("\n\nMicrosoft Table: \n\n", msft_df.tail())
```

Apple Table:

```
             date   Adj Close       Close        High         Low        Open  \
11102 2024-12-27  255.589996  255.589996  258.700012  253.059998  257.829987
11103 2024-12-30  252.199997  252.199997  253.500000  250.750000  252.229996
11104 2024-12-31  250.419998  250.419998  253.279999  249.429993  252.440002
11105 2025-01-02  243.850006  243.850006  249.100006  241.820007  248.929993
11106 2025-01-03  243.860001  243.860001  244.179993  241.889999  243.369995

         Volume  Daily Return  Volatility_5
11102  42355300     -1.324219      1.195941
11103  35557500     -1.326343      1.103469
11104  39480700     -0.705789      1.085592
11105  55558000     -2.623589      1.070336
11106  15135053      0.004099      0.968502
```

Microsoft Table:

```
             date        Open        High         Low       Close   Adj Close  \
9795 2025-01-28  434.600006  448.380005  431.380005  447.200012  447.200012
9796 2025-01-29  446.690002  446.880005  440.399994  442.329987  442.329987
```

```
9797 2025-01-30  418.769989  422.859985  413.160004  414.989990  414.989990
9798 2025-01-31  418.980011  420.690002  414.910004  415.059998  415.059998
9799 2025-02-03  411.600006  415.410004  408.660004  410.920013  410.920013

        Volume  Daily Return  Volatility_5
9795  23491700      2.908693      2.576610
9796  23581400     -1.089004      1.901359
9797  54586300     -6.180905      3.267812
9798  34223400      0.016870      3.317358
9799  25580600     -0.997442      3.283986
```

[6]:
```
#Statistical breakdown
print("Apple Summary Stats: \n\n", apple_df[['Open', 'High', 'Low', 'Close',␣
  ↪'Adj Close', 'Volume', 'Daily Return', 'Volatility_5']].describe())
print("\n\nMicrosoft Summary Stats: \n\n", msft_df[['Open', 'High', 'Low',␣
  ↪'Close', 'Adj Close', 'Volume', 'Daily Return', 'Volatility_5']].describe())
```

```
Apple Summary Stats:

                Open          High           Low         Close     Adj Close  \
count  11107.000000  11107.000000  11107.000000  11107.000000  11107.000000
mean      24.339076     24.598169     24.092608     24.357607     23.522229
std       50.166818     50.691902     49.682631     50.217498     49.767881
min        0.049665      0.049665      0.049107      0.049107      0.037815
25%        0.300090      0.306362      0.292411      0.300290      0.243402
50%        0.542679      0.553393      0.534598      0.542411      0.446682
75%       21.367679     21.569285     21.115715     21.397143     18.260086
max      258.190002    260.100006    257.630005    259.019989    259.019989

             Volume  Daily Return  Volatility_5
count  1.110700e+04  11106.000000  11102.000000
mean   3.154341e+08      0.109604      2.304649
std    3.348735e+08      2.778277      1.562694
min    0.000000e+00    -51.869200      0.075523
25%    1.111164e+08     -1.260233      1.266991
50%    2.036944e+08      0.000000      1.962168
75%    3.960418e+08      1.437862      2.946773
max    7.421641e+09     33.228009     24.495311


Microsoft Summary Stats:

                Open          High           Low         Close     Adj Close  \
count  9800.000000  9800.000000  9800.000000  9800.000000  9800.000000
mean     63.173728    63.812493    62.513818    63.187157    57.451679
std      98.682750    99.575073    97.725918    98.697881    98.714771
min       0.088542     0.092014     0.088542     0.090278     0.055554
25%       5.898438     5.976563     5.794922     5.880860     3.618860
```

```
50%      27.436250    27.770000    27.200001    27.490000    19.266710
75%      47.597813    48.145000    47.062500    47.592500    40.066001
max     467.000000   468.350006   464.459991   467.559998   465.786438

              Volume  Daily Return  Volatility_5
count   9.800000e+03   9799.000000   9795.000000
mean    5.630897e+07      0.112386      1.756862
std     3.812127e+07      2.102810      1.206792
min     2.304000e+06    -30.115887      0.068665
25%     3.141062e+07     -0.910068      0.964287
50%     4.946235e+07      0.037737      1.479103
75%     7.027870e+07      1.119693      2.207315
max     1.031789e+09     19.565212     18.018816
```

[7]:
```
#Notice NaNs
apple_df.head()
```

[7]:
```
         date  Adj Close     Close      High       Low      Open     Volume  \
0 1980-12-12   0.098834  0.128348  0.128906  0.128348  0.128348  469033600
1 1980-12-15   0.093678  0.121652  0.122210  0.121652  0.122210  175884800
2 1980-12-16   0.086802  0.112723  0.113281  0.112723  0.113281  105728000
3 1980-12-17   0.088951  0.115513  0.116071  0.115513  0.115513   86441600
4 1980-12-18   0.091530  0.118862  0.119420  0.118862  0.118862   73449600

   Daily Return  Volatility_5
0           NaN           NaN
1     -5.217063           NaN
2     -7.339813           NaN
3      2.475121           NaN
4      2.899232           NaN
```

[8]:
```
#Notice NaNs
msft_df.head()
```

[8]:
```
         date      Open      High       Low     Close  Adj Close      Volume  \
0 1986-03-13  0.088542  0.101563  0.088542  0.097222   0.059827  1031788800
1 1986-03-14  0.097222  0.102431  0.097222  0.100694   0.061963   308160000
2 1986-03-17  0.100694  0.103299  0.100694  0.102431   0.063032   133171200
3 1986-03-18  0.102431  0.103299  0.098958  0.099826   0.061429    67766400
4 1986-03-19  0.099826  0.100694  0.097222  0.098090   0.060361    47894400

   Daily Return  Volatility_5
0           NaN           NaN
1      3.571207           NaN
2      1.725111           NaN
3     -2.543234           NaN
4     -1.738989           NaN
```

[9]:
```
#only Nans are in the first five rows cause it uses previous 5 days before that␣
 ↪day to calculate, so drop those
apple_df.dropna(inplace=True)
msft_df.dropna(inplace=True)
```

[10]:
```
apple_df.head()
```

[10]:

	date	Adj Close	Close	High	Low	Open	Volume	\
5	1980-12-19	0.097116	0.126116	0.126674	0.126116	0.126116	48630400	
6	1980-12-22	0.101842	0.132254	0.132813	0.132254	0.132254	37363200	
7	1980-12-23	0.106140	0.137835	0.138393	0.137835	0.137835	46950400	
8	1980-12-24	0.111726	0.145089	0.145647	0.145089	0.145089	48003200	
9	1980-12-26	0.122039	0.158482	0.159040	0.158482	0.158482	55574400	

	Daily Return	Volatility_5
5	6.102858	5.758344
6	4.867013	5.317928
7	4.219907	1.474397
8	5.262747	1.202232
9	9.230927	1.963757

[11]:
```
msft_df.head()
```

[11]:

	date	Open	High	Low	Close	Adj Close	Volume	\
5	1986-03-20	0.098090	0.098090	0.094618	0.095486	0.058758	58435200	
6	1986-03-21	0.095486	0.097222	0.091146	0.092882	0.057156	59990400	
7	1986-03-24	0.092882	0.092882	0.089410	0.090278	0.055554	65289600	
8	1986-03-25	0.090278	0.092014	0.089410	0.092014	0.056622	32083200	
9	1986-03-26	0.092014	0.095486	0.091146	0.094618	0.058224	22752000	

	Daily Return	Volatility_5
5	-2.654749	2.816513
6	-2.727116	1.893905
7	-2.803508	0.432563
8	1.922941	2.016293
9	2.830012	2.814910

[12]:
```
#breakdown of cleaned up datasets now
print("Apple description:")
print(apple_df.describe())
print("\nApple data types:")
print(apple_df.dtypes)
print("\nApple missing values:")
print(apple_df.isna().sum())

print("\nMicrosoft description:")
print(msft_df.describe())
```

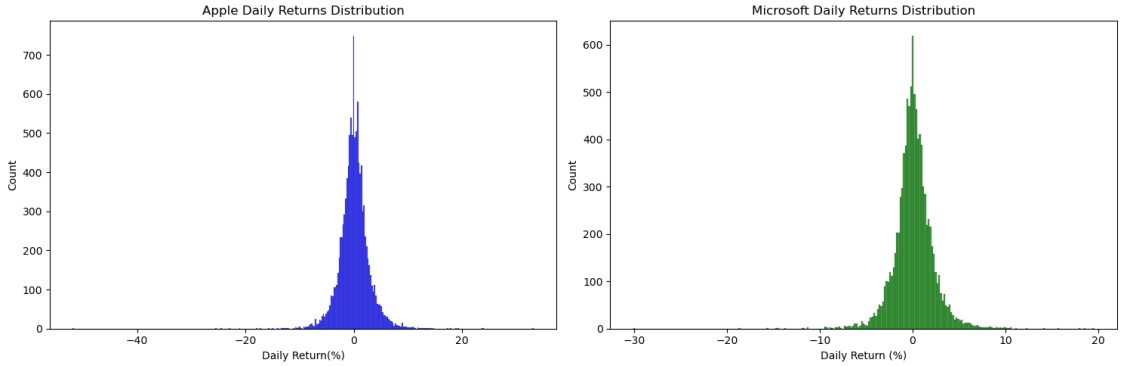
```
print("\nMicrosoft data types:")
print(msft_df.dtypes)
print("\nMicrosoft missing values:")
print(msft_df.isna().sum())
```

```
Apple description:
                                date     Adj Close         Close  \
count                          11102  11102.000000  11102.000000
mean   2002-12-18 06:22:14.678436352     23.532781     24.368523
min              1980-12-19 00:00:00      0.037815      0.049107
25%              1991-12-11 06:00:00      0.243700      0.300781
50%              2002-12-12 12:00:00      0.447646      0.542411
75%              2013-12-22 06:00:00     18.272139     21.406875
max              2025-01-03 00:00:00    259.019989    259.019989
std                              NaN     49.776603     50.226171

                High           Low          Open        Volume  Daily Return  \
count  11102.000000  11102.000000  11102.000000  1.110200e+04  11102.000000
mean      24.609193     24.103404     24.349983  3.154942e+08      0.110290
min        0.049665      0.049107      0.049665  0.000000e+00    -51.869200
25%        0.306920      0.293292      0.300223  1.111544e+08     -1.259590
50%        0.553571      0.535435      0.544464  2.037560e+08      0.000000
75%       21.588483     21.131250     21.400893  3.960593e+08      1.437291
max      260.100006    257.630005    258.190002  7.421641e+09     33.228009
std       50.700654     49.691213     50.175480  3.349223e+08      2.777201

       Volatility_5
count  11102.000000
mean       2.304649
min        0.075523
25%        1.266991
50%        1.962168
75%        2.946773
max       24.495311
std        1.562694

Apple data types:
date            datetime64[ns]
Adj Close              float64
Close                  float64
High                   float64
Low                    float64
Open                   float64
Volume                   int64
Daily Return           float64
Volatility_5           float64
dtype: object
```

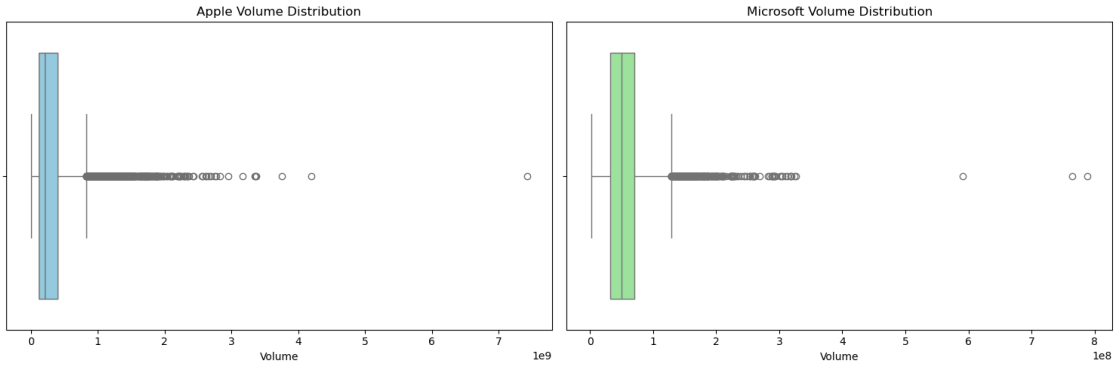
```
Apple missing values:
date            0
Adj Close       0
Close           0
High            0
Low             0
Open            0
Volume          0
Daily Return    0
Volatility_5    0
dtype: int64

Microsoft description:
                                date          Open          High           Low  \
count                           9795  9795.000000  9795.000000  9795.000000
mean   2005-08-17 20:16:23.522205056    63.205926    63.845015    62.545680
min              1986-03-20 00:00:00     0.090278     0.092014     0.089410
25%              1995-11-23 00:00:00     5.921875     6.015625     5.812500
50%              2005-08-16 00:00:00    27.450001    27.775000    27.200001
75%              2015-05-09 12:00:00    47.617500    48.173750    47.076250
max              2025-02-03 00:00:00   467.000000   468.350006   464.459991
std                              NaN    98.697642    99.590079    97.740681

             Close    Adj Close        Volume  Daily Return  Volatility_5
count  9795.000000  9795.000000  9.795000e+03   9795.000000   9795.000000
mean     63.219361    57.480975  5.617551e+07      0.112329      1.756862
min       0.090278     0.055554  2.304000e+06    -30.115887      0.068665
25%       5.906250     3.634482  3.140820e+07     -0.909028      0.964287
50%      27.500000    19.271568  4.945560e+07      0.037737      1.479103
75%      47.605000    40.090405  7.025070e+07      1.119149      2.207315
max     467.559998   465.786438  7.886880e+08     19.565212     18.018816
std      98.712775    98.731446  3.673823e+07      2.102631      1.206792

Microsoft data types:
date            datetime64[ns]
Open                   float64
High                   float64
Low                    float64
Close                  float64
Adj Close              float64
Volume                   int64
Daily Return           float64
Volatility_5           float64
dtype: object

Microsoft missing values:
date            0
Open            0
```

```
High            0
Low             0
Close           0
Adj Close       0
Volume          0
Daily Return    0
Volatility_5    0
dtype: int64
```

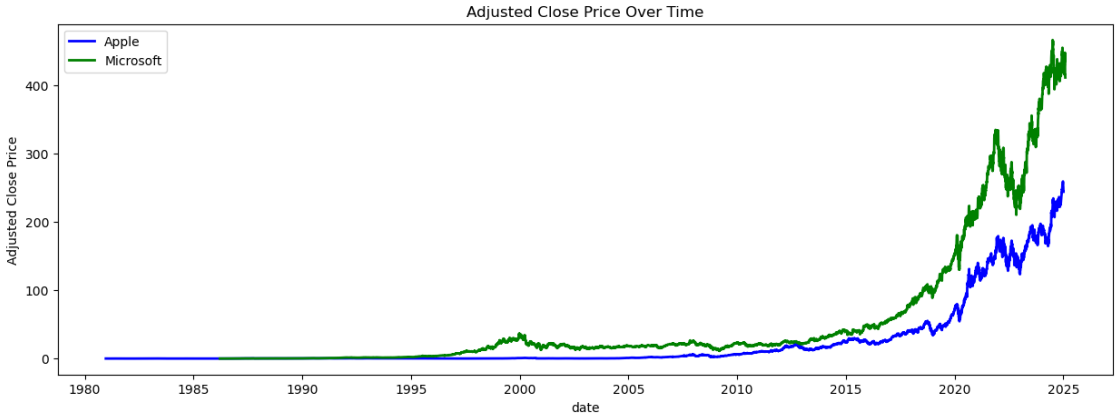
[13]:
```
#Visualization - Histograms of daily returns
plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.histplot(apple_df['Daily Return'], color='blue')
plt.title('Apple Daily Returns Distribution')
plt.xlabel('Daily Return(%)')

plt.subplot(1, 2, 2)
sns.histplot(msft_df['Daily Return'], color='green')
plt.title('Microsoft Daily Returns Distribution')
plt.xlabel('Daily Return (%)')
plt.tight_layout()
plt.show()
```



[14]:
```
#Visualization - boxplots of volume traded
plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.boxplot(x=apple_df['Volume'], color='skyblue')
plt.title('Apple Volume Distribution')

plt.subplot(1, 2, 2)
sns.boxplot(x=msft_df['Volume'], color='lightgreen')
plt.title('Microsoft Volume Distribution')
plt.tight_layout()
plt.show()
```

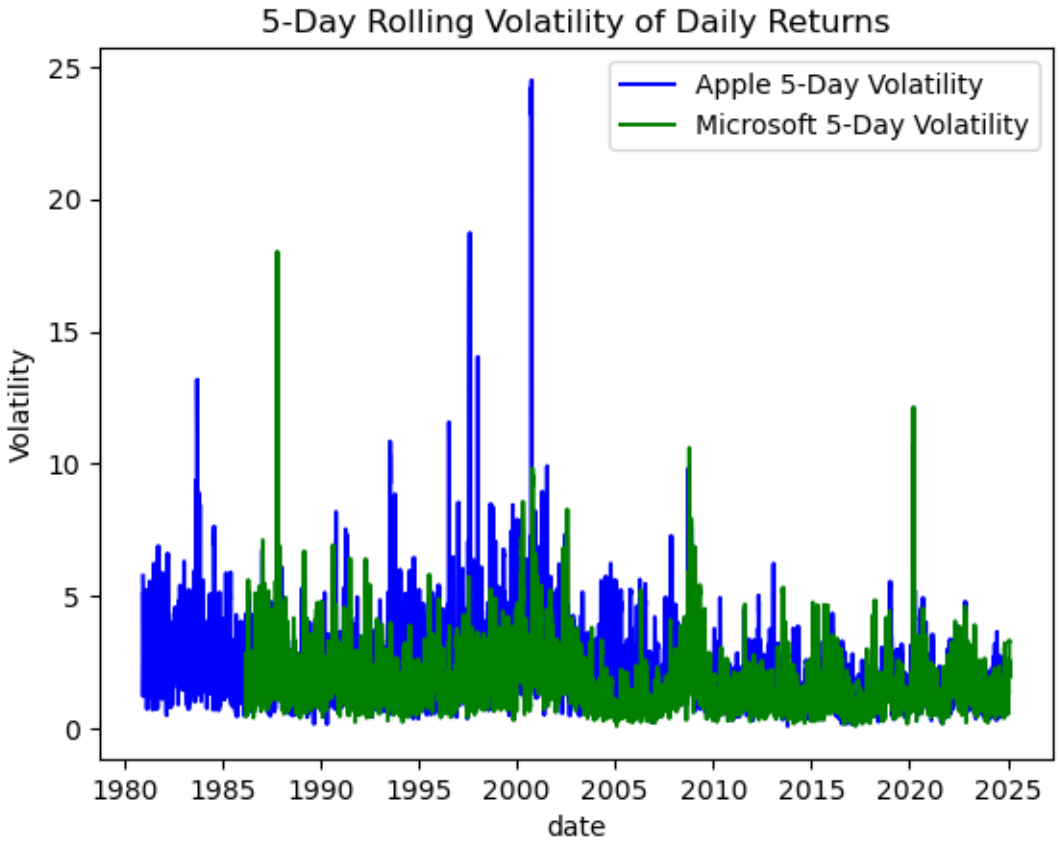


[15]:
```
#Visualization - line charts of adj closing price over time
plt.figure(figsize=(15, 5))
plt.plot(apple_df['date'], apple_df['Adj Close'], label='Apple', color='blue',␣
 ↪linewidth=2)
plt.plot(msft_df['date'], msft_df['Adj Close'], label='Microsoft',␣
 ↪color='green', linewidth=2)

plt.title('Adjusted Close Price Over Time')
plt.xlabel('date')
plt.ylabel('Adjusted Close Price')
plt.legend()
plt.show()
```



[16]:
```
#Visualization - Volatility of daily returns over a 5 day period (1 trading␣
 ↪week)
plt.plot(apple_df['date'], apple_df['Volatility_5'], label='Apple 5-Day␣
 ↪Volatility', color='blue')
```

```
plt.plot(msft_df['date'], msft_df['Volatility_5'], label='Microsoft 5-Day␣
 ↪Volatility', color='green')
plt.title('5-Day Rolling Volatility of Daily Returns')
plt.xlabel('date')
plt.ylabel('Volatility')
plt.legend()
plt.show()
```



EXPLORATORY DATA ANALYSI END

Logistic regression model START:First have to choose events that we want to look at and put them into two seperate "event types".

1 - tecnological events 2 - Government policy events

apple tech events - announcemnt of iphone, announcemnt of ipad, announcemnt of M1 chip apple policy events - US-china tariffs trumps first time in office, US-EU antitrust probe

microsoft tech events - release of Windows 95, announcemnt of Azure, announcemnt of Xbox microsoft policy events - antitrust lawsuit filed, antitrust lawsuit settled

[17]:
```
#Mark events in dataset first: 0 = normal day, 1 = post-technological␣
 ↪innovation event, 2 = post-givernment policy event
```

```
apple_events = {
    '2007-01-09': 1,  #iPhone
    '2010-01-27': 1,  #iPad
    '2020-11-10': 1,  #M1 chip
    '2018-03-22': 2,  #US-China tariffs
    '2020-06-16': 2   #US-EU antitrust probe
}

msft_events = {
    '1995-08-24': 1,  # Windows 95
    '2010-02-01': 1,  # Azure
    '2000-03-10': 1,  # Xbox
    '1998-05-18': 2,  # US antitrust lawsuit filed
    '2001-11-02': 2   # Final antitrust settled
}

def mark_event_window(df, event_dates, column='date', window=5):
    df['event_type'] = 0  #every normal day is 0

    for event_date_str, event_code in event_dates.items():
        event_date = pd.to_datetime(event_date_str)

        #first day after the event, event may have been on weekend so next␣
 ↪available "trading day"
        future_dates = df[column][df[column] >= event_date]
        if len(future_dates) == 0:
            continue

        start_idx = future_dates.index[0]
        end_idx = start_idx + window

        #if event is near end of df
        if end_idx < len(df):
            df.loc[start_idx:end_idx, 'event_type'] = event_code

    return df

apple_df = mark_event_window(apple_df, apple_events)
msft_df = mark_event_window(msft_df, msft_events)
```

[18]:
```
#want to make sure the events dates were correctly edited in each dataset
#Apple
print("Apple Event Type Count:")
print(apple_df['event_type'].value_counts())
print("Apple - Technology Events (1)")
display(apple_df[apple_df['event_type'] == 1][['date', 'event_type']])
```

```
print("Apple - Government Policy Events (2)")
display(apple_df[apple_df['event_type'] == 2][['date', 'event_type']])
```

```
Apple Event Type Count:
event_type
0    11072
1       18
2       12
Name: count, dtype: int64
Apple - Technology Events (1)
```

	date	event_type
6579	2007-01-09	1
6580	2007-01-10	1
6581	2007-01-11	1
6582	2007-01-12	1
6583	2007-01-16	1
6584	2007-01-17	1
7347	2010-01-27	1
7348	2010-01-28	1
7349	2010-01-29	1
7350	2010-02-01	1
7351	2010-02-02	1
7352	2010-02-03	1
10064	2020-11-10	1
10065	2020-11-11	1
10066	2020-11-12	1
10067	2020-11-13	1
10068	2020-11-16	1
10069	2020-11-17	1

```
Apple - Government Policy Events (2)
```

	date	event_type
9399	2018-03-22	2
9400	2018-03-23	2
9401	2018-03-26	2
9402	2018-03-27	2
9403	2018-03-28	2
9404	2018-03-29	2
9961	2020-06-16	2
9962	2020-06-17	2
9963	2020-06-18	2
9964	2020-06-19	2
9965	2020-06-22	2
9966	2020-06-23	2

[19]:
```
apple_df
```

[19]:
```
            date   Adj Close       Close        High         Low        Open  \
5     1980-12-19    0.097116    0.126116    0.126674    0.126116    0.126116
6     1980-12-22    0.101842    0.132254    0.132813    0.132254    0.132254
7     1980-12-23    0.106140    0.137835    0.138393    0.137835    0.137835
8     1980-12-24    0.111726    0.145089    0.145647    0.145089    0.145089
9     1980-12-26    0.122039    0.158482    0.159040    0.158482    0.158482
…            …           …           …           …           …           …
11102 2024-12-27  255.589996  255.589996  258.700012  253.059998  257.829987
11103 2024-12-30  252.199997  252.199997  253.500000  250.750000  252.229996
11104 2024-12-31  250.419998  250.419998  253.279999  249.429993  252.440002
11105 2025-01-02  243.850006  243.850006  249.100006  241.820007  248.929993
11106 2025-01-03  243.860001  243.860001  244.179993  241.889999  243.369995

          Volume  Daily Return  Volatility_5  event_type
5       48630400      6.102858      5.758344           0
6       37363200      4.867013      5.317928           0
7       46950400      4.219907      1.474397           0
8       48003200      5.262747      1.202232           0
9       55574400      9.230927      1.963757           0
…            …           …           …           …
11102   42355300     -1.324219      1.195941           0
11103   35557500     -1.326343      1.103469           0
11104   39480700     -0.705789      1.085592           0
11105   55558000     -2.623589      1.070336           0
11106   15135053      0.004099      0.968502           0

[11102 rows x 10 columns]
```

[20]:
```
#Microsoft
print("\nMicrosoft Event Type Counts:")
print(msft_df['event_type'].value_counts())
print("Microsoft - Technology Events (1)")
display(msft_df[msft_df['event_type'] == 1][['date', 'event_type']])
print("Microsoft - Government Policy Events (2)")
display(msft_df[msft_df['event_type'] == 2][['date', 'event_type']])
```

```
Microsoft Event Type Counts:
event_type
0    9765
1      18
2      12
Name: count, dtype: int64
Microsoft - Technology Events (1)
```

```
           date  event_type
2390 1995-08-24           1
2391 1995-08-25           1
```

```
2392 1995-08-28           1
2393 1995-08-29           1
2394 1995-08-30           1
2395 1995-08-31           1
3537 2000-03-10           1
3538 2000-03-13           1
3539 2000-03-14           1
3540 2000-03-15           1
3541 2000-03-16           1
3542 2000-03-17           1
6024 2010-02-01           1
6025 2010-02-02           1
6026 2010-02-03           1
6027 2010-02-04           1
6028 2010-02-05           1
6029 2010-02-08           1
```

Microsoft - Government Policy Events (2)

```
           date  event_type
3079 1998-05-18           2
3080 1998-05-19           2
3081 1998-05-20           2
3082 1998-05-21           2
3083 1998-05-22           2
3084 1998-05-26           2
3950 2001-11-02           2
3951 2001-11-05           2
3952 2001-11-06           2
3953 2001-11-07           2
3954 2001-11-08           2
3955 2001-11-09           2
```

msft_df

[21]:
```python
#functions for logistic regression
def sigmoid(z):
    z = np.clip(z, -500, 500)  # prevents overflow
    return 1 / (1 + np.exp(-z))

#train model utilizng gradient descent
def fit_logistic(X, y, lr=0.01, n_iter=10000):
    weights = np.zeros(X.shape[1])
    bias = 0

    for _ in range(n_iter):
        linear = np.dot(X, weights) + bias
        probs = sigmoid(linear)
```

```
        error = probs - y
        dweigh = np.dot(X.T, error) / len(y)
        dbias = np.mean(error)

        weights -= lr * dweigh
        bias -= lr * dbias

    return weights, bias

#predict prob of stock daily return being positive
def predict_proba(X, weights, bias):
    return sigmoid(np.dot(X, weights) + bias)

#predict up or down based off of predict_proba. >= 0.5 is up <= 0.5 is down on␣
 ↪the day
def predict(X, weights, bias, threshold=0.5):
    return (predict_proba(X, weights, bias) >= threshold).astype(int)
```

[22]:
```
def train_model(df):
    train_data = df[df['event_type'] == 0]
    X_train = train_data[features].values
    y_train = train_data['Target'].values

    weights, bias = fit_logistic(X_train, y_train)
    return weights, bias

#make predictions during blocked out time period for event types
def evaluate_model(df, weights, bias, event_type):
    test_data = df[df['event_type'] == event_type]
    X_test = test_data[features].values
    y_test = test_data['Target'].values

    if len(y_test) == 0:
        return None, 0

    y_pred = predict(X_test, weights, bias)
    accuracy = np.mean(y_pred == y_test)
    return accuracy
```

[23]:
```
features = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume', 'Daily␣
 ↪Return', 'Volatility_5']

#target, if = 1 then daily return is positive and vice versa for 0
apple_df['Target'] = (apple_df['Daily Return'].shift(-1) > 0).astype(int)
msft_df['Target'] = (msft_df['Daily Return'].shift(-1) > 0).astype(int)

#model is trained on normal days when event_typ = 0
```

```
apple_weights, apple_bias = train_model(apple_df)
msft_weights, msft_bias = train_model(msft_df)

#then makes predictions based off the time period we blocked out for each event␣
 ↪type
apple_tech_acc = evaluate_model(apple_df, apple_weights, apple_bias, 1)
apple_policy_acc = evaluate_model(apple_df, apple_weights, apple_bias, 2)

msft_tech_acc = evaluate_model(msft_df, msft_weights, msft_bias, 1)
msft_policy_acc = evaluate_model(msft_df, msft_weights, msft_bias, 2)

#accuracy results for each event type
print("Apple Tech Events - Accuracy:", round(apple_tech_acc, 3))
print("Apple Policy Events - Accuracy:", round(apple_policy_acc, 3))

print("Microsoft Tech Events - Accuracy:", round(msft_tech_acc, 3))
print("Microsoft Policy Events - Accuracy:", round(msft_policy_acc, 3))
```

```
Apple Tech Events - Accuracy: 0.444
Apple Policy Events - Accuracy: 0.417
Microsoft Tech Events - Accuracy: 0.444
Microsoft Policy Events - Accuracy: 0.667
```

[24]:
```
#EXTRA VERIFICATION FOR PAPER
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, confusion_matrix

#Apple Tech events
apple_tech_data = apple_df[apple_df['event_type'] == 1]
X_apple_tech = apple_tech_data[features].values
y_apple_tech = apple_tech_data['Target'].values
y_pred_apple_tech = predict(X_apple_tech, apple_weights, apple_bias)

print("\nApple Tech Event Verification:")
print("Precision:", round(precision_score(y_apple_tech, y_pred_apple_tech), 3))
print("F1 Score:", round(f1_score(y_apple_tech, y_pred_apple_tech), 3))
print("Confusion Matrix:\n", confusion_matrix(y_apple_tech, y_pred_apple_tech))

#Apple Policy Events
apple_policy_data = apple_df[apple_df['event_type'] == 2]
X_apple_policy = apple_policy_data[features].values
y_apple_policy = apple_policy_data['Target'].values
y_pred_apple_policy = predict(X_apple_policy, apple_weights, apple_bias)

print("\nApple Policy Event Verification:")
print("Precision:", round(precision_score(y_apple_policy, y_pred_apple_policy),␣
 ↪3))
```

```
print("F1 Score:", round(f1_score(y_apple_policy, y_pred_apple_policy), 3))
print("Confusion Matrix:\n", confusion_matrix(y_apple_policy,␣
 ↪y_pred_apple_policy))

#Microsoft Tech Events
msft_tech_data = msft_df[msft_df['event_type'] == 1]
X_msft_tech = msft_tech_data[features].values
y_msft_tech = msft_tech_data['Target'].values
y_pred_msft_tech = predict(X_msft_tech, msft_weights, msft_bias)

print("\nMicrosoft Tech Event Verification:")
print("Precision:", round(precision_score(y_msft_tech, y_pred_msft_tech), 3))
print("F1 Score:", round(f1_score(y_msft_tech, y_pred_msft_tech), 3))
print("Confusion Matrix:\n", confusion_matrix(y_msft_tech, y_pred_msft_tech))

# Microsoft Policy Events
msft_policy_data = msft_df[msft_df['event_type'] == 2]
X_msft_policy = msft_policy_data[features].values
y_msft_policy = msft_policy_data['Target'].values
y_pred_msft_policy = predict(X_msft_policy, msft_weights, msft_bias)

print("\nMicrosoft Policy Event Verification:")
print("Precision:", round(precision_score(y_msft_policy, y_pred_msft_policy),␣
 ↪3))
print("F1 Score:", round(f1_score(y_msft_policy, y_pred_msft_policy), 3))
print("Confusion Matrix:\n", confusion_matrix(y_msft_policy,␣
 ↪y_pred_msft_policy))
```

```
Apple Tech Event Verification:
Precision: 0.444
F1 Score: 0.615
Confusion Matrix:
 [[ 0 10]
 [ 0  8]]

Apple Policy Event Verification:
Precision: 0.417
F1 Score: 0.588
Confusion Matrix:
 [[0 7]
 [0 5]]

Microsoft Tech Event Verification:
Precision: 0.444
F1 Score: 0.615
Confusion Matrix:
```

```
 [[ 0 10]
 [ 0  8]]

Microsoft Policy Event Verification:
Precision: 0.667
F1 Score: 0.8
Confusion Matrix:
 [[0 4]
 [0 8]]
```

[ ]: