Unsupervised Learning: Trade & Ahead

Problem Statement

Context

The stock market has consistently proven to be a good place to invest in and save for the future. There are a lot of compelling reasons to invest in stocks. It can help in fighting inflation, create wealth, and also provides some tax benefits. Good steady returns on investments over a long period of time can also grow a lot more than seems possible. Also, thanks to the power of compound interest, the earlier one starts investing, the larger the corpus one can have for retirement. Overall, investing in stocks can help meet life's financial aspirations.

It is important to maintain a diversified portfolio when investing in stocks in order to maximise earnings under any market condition. Having a diversified portfolio tends to yield higher returns and face lower risk by tempering potential losses when the market is down. It is often easy to get lost in a sea of financial metrics to analyze while determining the worth of a stock, and doing the same for a multitude of stocks to identify the right picks for an individual can be a tedious task. By doing a cluster analysis, one can identify stocks that exhibit similar characteristics and ones which exhibit minimum correlation. This will help investors better analyze stocks across different market segments and help protect against risks that could make the portfolio vulnerable to losses.

Objective

Trade&Ahead is a financial consultancy firm who provide their customers with personalized investment strategies. They have hired you as a Data Scientist and provided you with data comprising stock price and some financial indicators for a few companies listed under the New York Stock Exchange. They have assigned you the tasks of analyzing the data, grouping the stocks based on the attributes provided, and sharing insights about the characteristics of each group.

Data Dictionary

- Ticker Symbol: An abbreviation used to uniquely identify publicly traded shares of a particular stock on a particular stock market
- Company: Name of the company
- GICS Sector: The specific economic sector assigned to a company by

the Global Industry Classification Standard (GICS) that best defines its business operations
- GICS Sub Industry: The specific sub-industry group assigned to a company by the Global Industry Classification Standard (GICS) that best defines its business operations
- Current Price: Current stock price in dollars
- Price Change: Percentage change in the stock price in 13 weeks
- Volatility: Standard deviation of the stock price over the past 13 weeks
- ROE: A measure of financial performance calculated by dividing net income by shareholders' equity (shareholders' equity is equal to a company's assets minus its debt)
- Cash Ratio: The ratio of a company's total reserves of cash and cash equivalents to its total current liabilities
- Net Cash Flow: The difference between a company's cash inflows and outflows (in dollars)
- Net Income: Revenues minus expenses, interest, and taxes (in dollars)
- Earnings Per Share: Company's net profit divided by the number of common shares it has outstanding (in dollars)
- Estimated Shares Outstanding: Company's stock currently held by all its shareholders
- P/E Ratio: Ratio of the company's current stock price to the earnings per share
- P/B Ratio: Ratio of the company's stock price per share by its book value per share (book value of a company is the net difference between that company's total assets and total liabilities)

Importing necessary libraries and data

In [647…
```
# Installing the libraries with the specified version.
# uncomment and run the following line if Google Colab is being used
# !pip install scikit-learn==1.2.2 seaborn==0.13.1 matplotlib==3.7.1 numpy==1.
```

In [648…
```
# Installing the libraries with the specified version.
# uncomment and run the following lines if Jupyter Notebook is being used
# !pip install scikit-learn==1.2.2 seaborn==0.13.1 matplotlib==3.7.1 numpy==1.
# !pip install --upgrade -q jinja2
```

Note: *After running the above cell, kindly restart the notebook kernel and run all cells sequentially from the start again.*

In [650…
```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from scipy.stats import zscore
```

In [651…
```
## loading the dataset
data = pd.read_csv("stock_data.csv")
data.head()
```

Out[651…

	Ticker Symbol	Security	GICS Sector	GICS Sub Industry	Current Price	Price Change	Volatili
0	AAL	American Airlines Group	Industrials	Airlines	42.349998	9.999995	1.6871
1	ABBV	AbbVie	Health Care	Pharmaceuticals	59.240002	8.339433	2.1978
2	ABT	Abbott Laboratories	Health Care	Health Care Equipment	44.910000	11.301121	1.2736
3	ADBE	Adobe Systems Inc	Information Technology	Application Software	93.940002	13.977195	1.3576
4	ADI	Analog Devices, Inc.	Information Technology	Semiconductors	55.320000	-1.827858	1.7011

Data Overview

- Observations
- Sanity checks

Observations

In [654…
```
## looking at the shape of the dataset
data.shape
```

Out[654…
```
(340, 15)
```

In [655…
```
## looking at the statistical summary of the data
data.describe()
```

Out[655…

	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cas Flo
count	340.000000	340.000000	340.000000	340.000000	340.000000	3.400000e+(
mean	80.862345	4.078194	1.525976	39.597059	70.023529	5.553762e+(
std	98.055086	12.006338	0.591798	96.547538	90.421331	1.946365e+(
min	4.500000	-47.129693	0.733163	1.000000	0.000000	-1.120800e+
25%	38.555000	-0.939484	1.134878	9.750000	18.000000	-1.939065e+(
50%	59.705000	4.819505	1.385593	15.000000	47.000000	2.098000e+(
75%	92.880001	10.695493	1.695549	27.000000	99.000000	1.698108e+(
max	1274.949951	55.051683	4.580042	917.000000	958.000000	2.076400e+

Observation

- it can be seen that the Current Price has outlier because the min is 4.500 while the max is 1274.9499 with a mean of 80.8
- not just the current price but majority of the columns have alot of outliers

In [657…

```
## looking at the different datatypes of each column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 340 entries, 0 to 339
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Ticker Symbol                340 non-null    object 
 1   Security                     340 non-null    object 
 2   GICS Sector                  340 non-null    object 
 3   GICS Sub Industry            340 non-null    object 
 4   Current Price                340 non-null    float64
 5   Price Change                 340 non-null    float64
 6   Volatility                   340 non-null    float64
 7   ROE                          340 non-null    int64  
 8   Cash Ratio                   340 non-null    int64  
 9   Net Cash Flow                340 non-null    int64  
 10  Net Income                   340 non-null    int64  
 11  Earnings Per Share           340 non-null    float64
 12  Estimated Shares Outstanding  340 non-null    float64
 13  P/E Ratio                    340 non-null    float64
 14  P/B Ratio                    340 non-null    float64
dtypes: float64(7), int64(4), object(4)
memory usage: 40.0+ KB
```

Observation

- There are no null values
- There are 7 decimal columns, 4 whole number columns and 4 string columns

Sanity checks

In [660…
```
## checking for null values
data.isna().sum()
```

Out[660…
```
Ticker Symbol                 0
Security                      0
GICS Sector                   0
GICS Sub Industry             0
Current Price                 0
Price Change                  0
Volatility                    0
ROE                           0
Cash Ratio                    0
Net Cash Flow                 0
Net Income                    0
Earnings Per Share            0
Estimated Shares Outstanding  0
P/E Ratio                     0
P/B Ratio                     0
dtype: int64
```

The dataset has no null values

In [662…
```
## checking for duplicates
data.duplicated().sum()
```

Out[662…
```
0
```

In [ ]:

In [ ]:

Exploratory Data Analysis (EDA)

- EDA is an important part of any project involving data.
- It is important to investigate and understand the data better before building a model with it.
- A few questions have been mentioned below which will help you

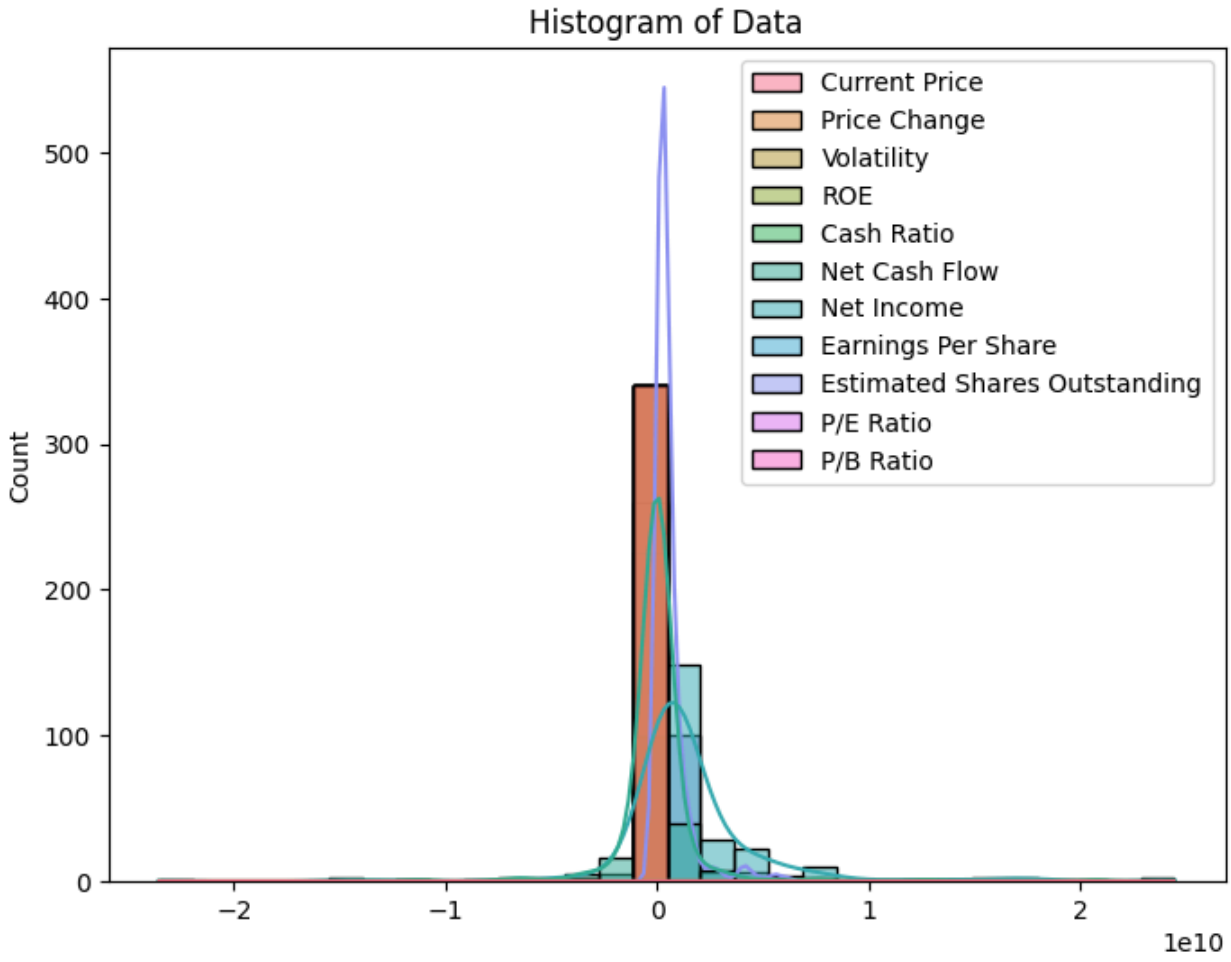
approach the analysis in the right manner and generate insights from the data.
- A thorough analysis of the data, in addition to the questions mentioned below, should be done.

Questions:

1. What does the distribution of stock prices look like?
2. The stocks of which economic sector have seen the maximum price increase on average?
3. How are the different variables correlated with each other?
4. Cash ratio provides a measure of a company's ability to cover its short-term obligations using only cash and cash equivalents. How does the average cash ratio vary across economic sectors?
5. P/E ratios can help determine the relative value of a company's shares as they signify the amount of money an investor is willing to invest in a single share of a company per dollar of its earnings. How does the P/E ratio vary, on average, across economic sectors?

In [665…

```
# Create a histogram with histogram plot
plt.figure(figsize=(8, 6))
sns.histplot(data, bins = 30, kde=True)
plt.title('Histogram of Data')
plt.show()
```



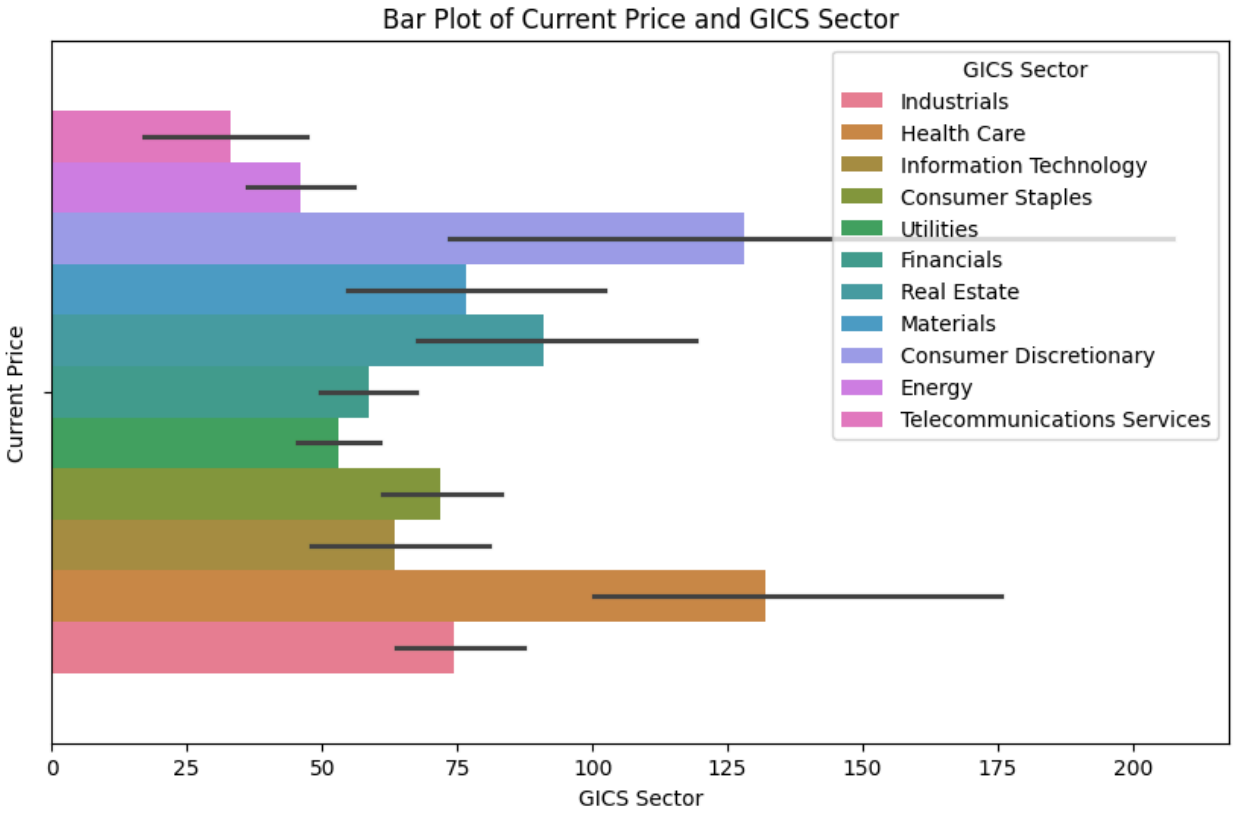
The dataset is almost normally distributed

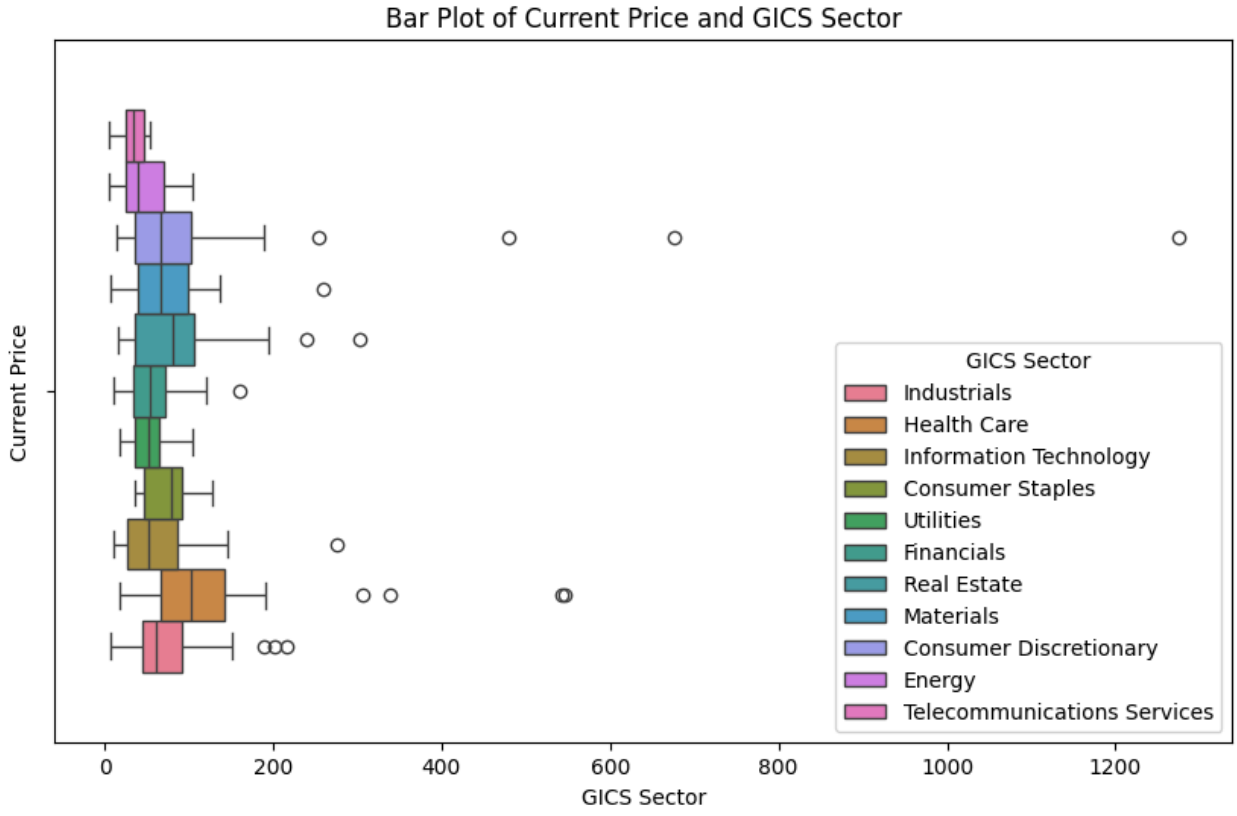
In [667…
```
stock_max_price_change = data.groupby(["GICS Sector", "Price Change"])["Currer
stock_max_price_change = stock_max_price_change.sort_values(by="Price Change",
print(stock_max_price_change)
```

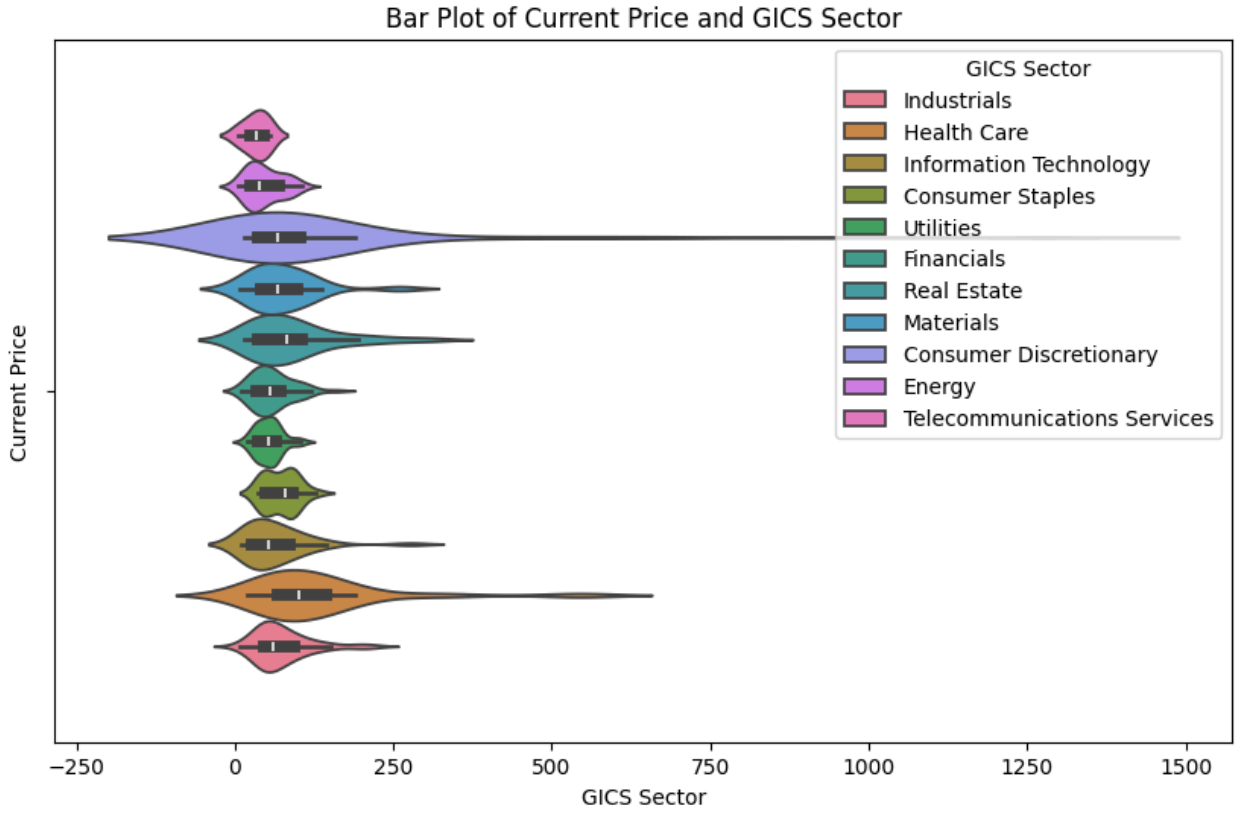
```
                GICS Sector  Price Change         sum        mean
263  Information Technology     55.051683   65.989998   65.989998
283               Materials     37.489677   66.599998   66.599998
39   Consumer Discretionary     34.803917   85.250000   85.250000
177              Health Care     33.177346   54.070000   54.070000
38   Consumer Discretionary     32.268105  675.890015  675.890015
..                      ...           ...         ...         ...
264               Materials    -31.685167    6.770000    6.770000
0    Consumer Discretionary    -33.131268  479.850006  479.850006
61                   Energy    -38.101788    4.500000    4.500000
60                   Energy    -44.798137    7.110000    7.110000
59                   Energy    -47.129693   14.920000   14.920000

[340 rows x 4 columns]
```

In [668…
```
plt.figure(figsize=(10,6))
```

```
sns.barplot(data=data, x='Current Price',hue = 'GICS Sector')
plt.title('Bar Plot of Current Price and GICS Sector')
plt.xlabel('GICS Sector')
plt.ylabel('Current Price')
plt.gca().invert_yaxis()
plt.show()

plt.figure(figsize=(10,6))
sns.boxplot(data=data, x='Current Price',hue = 'GICS Sector')
plt.title('Bar Plot of Current Price and GICS Sector')
plt.xlabel('GICS Sector')
plt.ylabel('Current Price')
plt.gca().invert_yaxis()
plt.show()

plt.figure(figsize=(10,6))
sns.violinplot(data=data, x='Current Price',hue = 'GICS Sector')
plt.title('Bar Plot of Current Price and GICS Sector')
plt.xlabel('GICS Sector')
plt.ylabel('Current Price')
plt.gca().invert_yaxis()
plt.show()
```



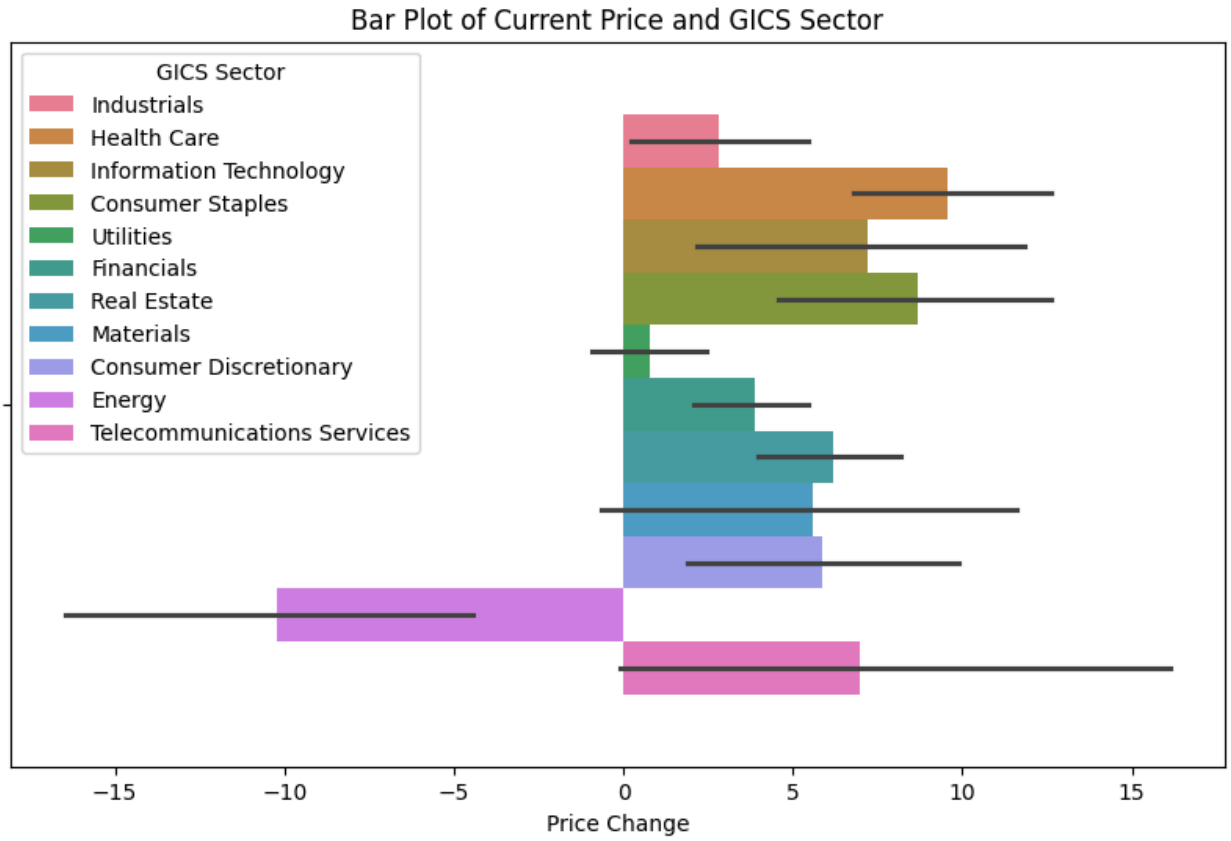


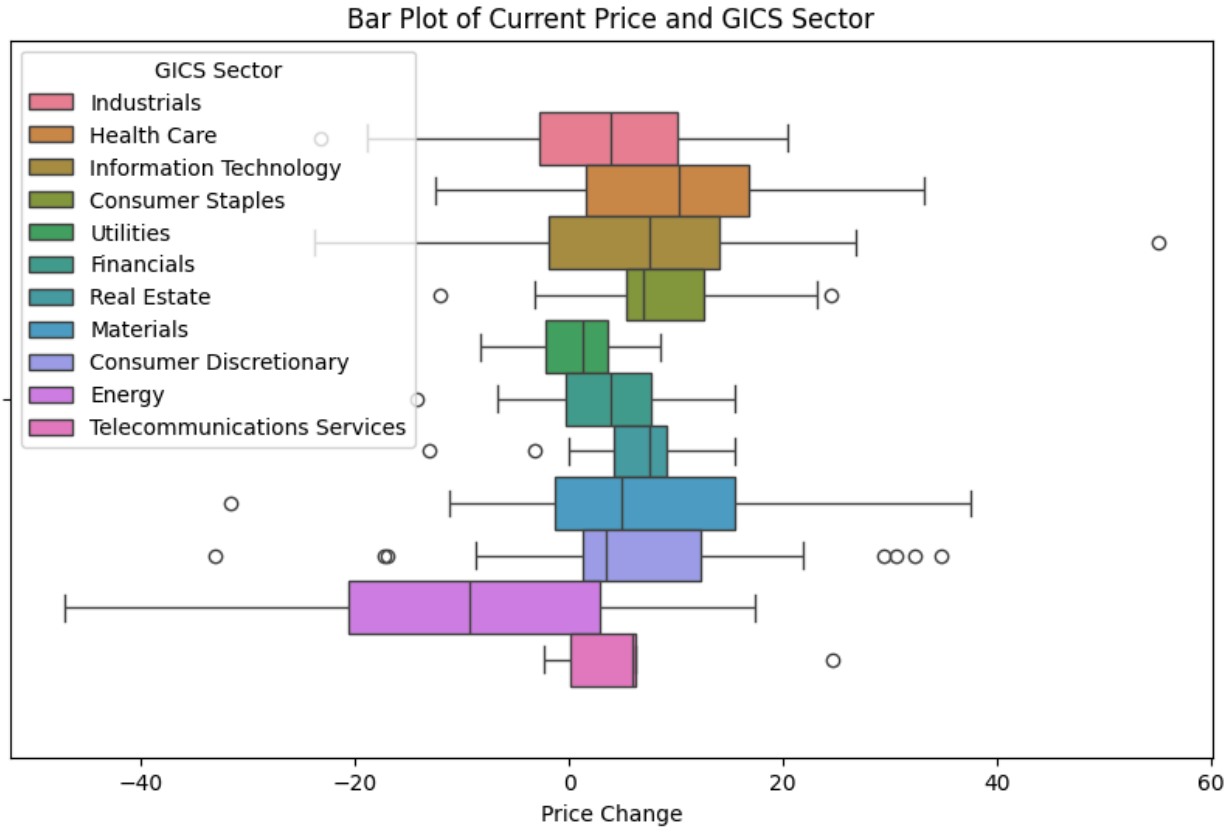


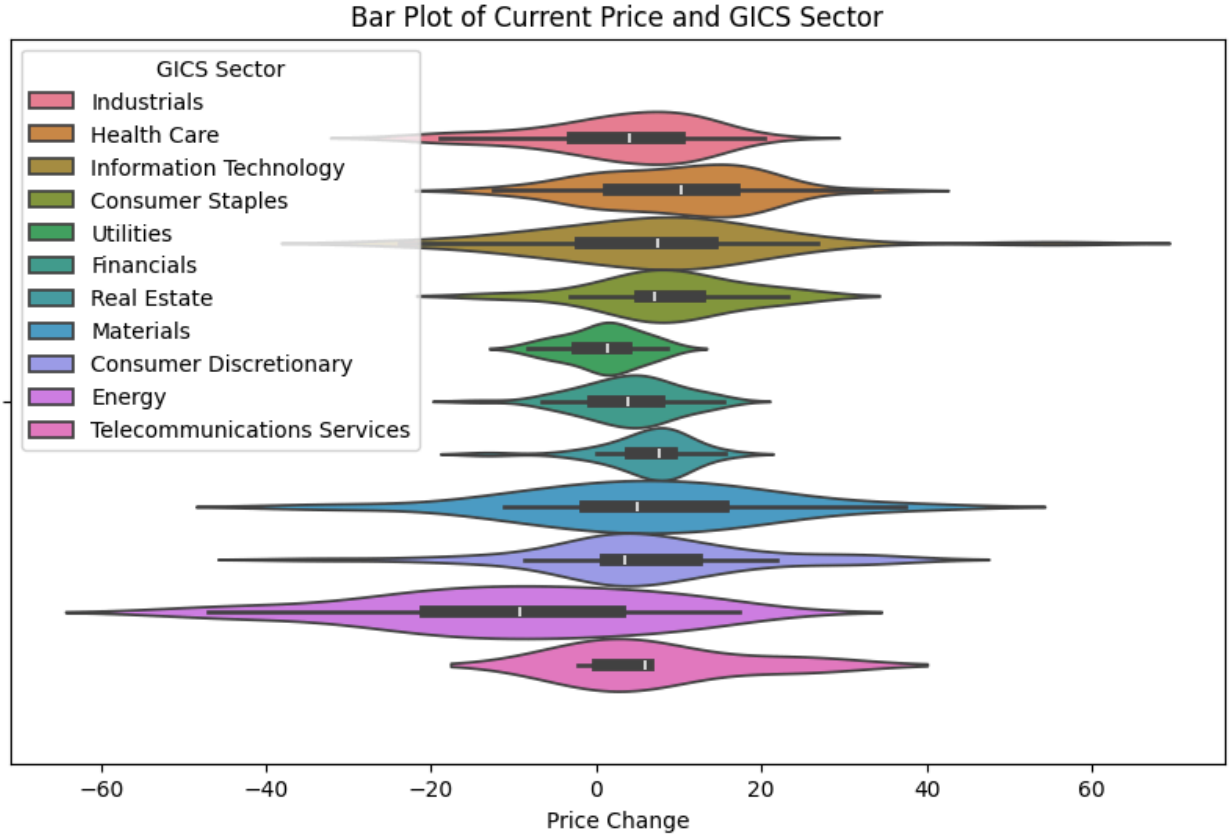
In [669…

```
plt.figure(figsize=(10,6))
sns.barplot(data=data, x='Price Change',hue = 'GICS Sector')
plt.title('Bar Plot of Current Price and GICS Sector')
```

```
plt.show()

plt.figure(figsize=(10,6))
sns.boxplot(data=data, x='Price Change',hue = 'GICS Sector')
plt.title('Bar Plot of Current Price and GICS Sector')
plt.show()

plt.figure(figsize=(10,6))
sns.violinplot(data=data, x='Price Change',hue = 'GICS Sector')
plt.title('Bar Plot of Current Price and GICS Sector')
plt.show()
```







In [670…

```
correlation = data.corr()
correlation
```

```
C:\Users\NYB  COMPUTER\AppData\Local\Temp\ipykernel_19664\3106392754.py:1: Futu
reWarning: The default value of numeric_only in DataFrame.corr is deprecated. I
n a future version, it will default to False. Select only valid columns or spec
ify the value of numeric_only to silence this warning.
  correlation = data.corr()
```

Out[670…

	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	In
Current Price	1.000000	0.134982	-0.124257	-0.000549	0.127816	-0.021961	0.0
Price Change	0.134982	1.000000	-0.408281	-0.043310	0.168586	0.026746	0.1
Volatility	-0.124257	-0.408281	1.000000	0.162532	0.020605	-0.014172	-0.3
ROE	-0.000549	-0.043310	0.162532	1.000000	-0.069122	-0.052205	-0.2
Cash Ratio	0.127816	0.168586	0.020605	-0.069122	1.000000	0.113819	0.0
Net Cash Flow	-0.021961	0.026746	-0.014172	-0.052205	0.113819	1.000000	0.0
Net Income	0.036625	0.179298	-0.383433	-0.288524	0.028589	0.044229	1.0
Earnings Per Share	0.479604	0.175401	-0.379309	-0.405291	0.024759	0.019348	0.5
Estimated Shares Outstanding	-0.145306	0.033656	-0.095704	-0.030288	0.135869	-0.051392	0.5
P/E Ratio	0.260273	-0.002491	0.263600	0.023341	0.089483	0.026876	-0.2
P/B Ratio	0.124634	0.034329	0.046528	-0.063218	0.229672	0.057039	-0.0

In [671…

```
data.head()
```

Out[671…

	Ticker Symbol	Security	GICS Sector	GICS Sub Industry	Current Price	Price Change	Volatili
0	AAL	American Airlines Group	Industrials	Airlines	42.349998	9.999995	1.6871
1	ABBV	AbbVie	Health Care	Pharmaceuticals	59.240002	8.339433	2.1978
2	ABT	Abbott Laboratories	Health Care	Health Care Equipment	44.910000	11.301121	1.2736
3	ADBE	Adobe Systems Inc	Information Technology	Application Software	93.940002	13.977195	1.3576
4	ADI	Analog Devices, Inc.	Information Technology	Semiconductors	55.320000	-1.827858	1.7011

Data Preprocessing

- Duplicate value check
- Missing value treatment
- Outlier check
- Feature engineering (if needed)
- Any other preprocessing steps (if needed)

In [673…

```
## Separating numerical columns from categorical columns
data_num = data.iloc[:,4:]
data_num.head()
```

Out[673…

	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	Net Income	Earnings Pe Share
0	42.349998	9.999995	1.687151	135	51	-604000000	7610000000	11.39
1	59.240002	8.339433	2.197887	130	77	51000000	5144000000	3.15
2	44.910000	11.301121	1.273646	21	67	938000000	4423000000	2.94
3	93.940002	13.977195	1.357679	9	180	-240840000	629551000	1.26
4	55.320000	-1.827858	1.701169	14	272	315120000	696878000	0.31

In [674…

```
## scaling the numerical columns of the dataset
datscaled = data_num.apply(zscore)
datscaled
```

Out[674…

	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	Net Income	E
0	-0.393341	0.493950	0.272749	0.989601	-0.210698	-0.339355	1.554415	1
1	-0.220837	0.355439	1.137045	0.937737	0.077269	-0.002335	0.927628	0
2	-0.367195	0.602479	-0.427007	-0.192905	-0.033488	0.454058	0.744371	0
3	0.133567	0.825696	-0.284802	-0.317379	1.218059	-0.152497	-0.219816	-0
4	-0.260874	-0.492636	0.296470	-0.265515	2.237018	0.133564	-0.202703	-0
...	...	...	...	...	...	...	...	
335	-0.486181	0.901646	0.540121	-0.255142	4.308162	-0.559673	-1.487784	-1
336	-0.289510	-1.065766	-0.079703	1.062211	-0.476513	0.053235	-0.051186	0
337	0.221913	0.439539	-0.206067	-0.400362	0.332009	0.164889	-0.342467	-0
338	-0.547053	-0.436811	-0.097813	-0.369243	0.320933	-0.051022	-0.301171	-0
339	-0.336453	1.051046	0.142671	-0.078803	-0.055639	0.111378	-0.293666	-0

340 rows × 11 columns

In [675…

```
## Encoding categorical variables
data_cat = data.iloc[:, :4]
data_cat
```

Out[675…

	Ticker Symbol	Security	GICS Sector	GICS Sub Industry
0	AAL	American Airlines Group	Industrials	Airlines
1	ABBV	AbbVie	Health Care	Pharmaceuticals
2	ABT	Abbott Laboratories	Health Care	Health Care Equipment
3	ADBE	Adobe Systems Inc	Information Technology	Application Software
4	ADI	Analog Devices, Inc.	Information Technology	Semiconductors
...	...	...	...	...
335	YHOO	Yahoo Inc.	Information Technology	Internet Software & Services
336	YUM	Yum! Brands Inc	Consumer Discretionary	Restaurants
337	ZBH	Zimmer Biomet Holdings	Health Care	Health Care Equipment
338	ZION	Zions Bancorp	Financials	Regional Banks
339	ZTS	Zoetis	Health Care	Pharmaceuticals

340 rows × 4 columns

In [676…
```
data_cat.columns
```

Out[676…
```
Index(['Ticker Symbol', 'Security', 'GICS Sector', 'GICS Sub Industry'], dtype='object')
```

In [677…
```
data_cat["price change"] = datscaled["Price Change"]
```

In [678…
```
## Encoding categorical variables
def mean_encode(df, col, mean_col):
    df_dict = df.groupby([col])[mean_col].mean().to_dict()
    df[col] = df[col].map(df_dict)
    return df
for col in data_cat.columns[0:4]:
    data_cat = mean_encode(data_cat, col, "price change")
```

In [679…
```
print(data_cat)
```

```
     Ticker Symbol  Security  GICS Sector  GICS Sub Industry  price change
0         0.493950  0.493950    -0.103854           0.452443      0.493950
1         0.355439  0.355439     0.459389           0.621119      0.355439
2         0.602479  0.602479     0.459389           0.438189      0.602479
3         0.825696  0.825696     0.261854          -0.125655      0.825696
4        -0.492636 -0.492636     0.261854           0.864016     -0.492636
..             ...       ...          ...                ...           ...
335       0.901646  0.901646     0.261854           0.159706      0.901646
336      -1.065766 -1.065766     0.147464          -0.948832     -1.065766
337       0.439539  0.439539     0.459389           0.438189      0.439539
338      -0.436811 -0.436811    -0.017749          -0.033903     -0.436811
339       1.051046  1.051046     0.459389           0.621119      1.051046

[340 rows x 5 columns]
```

In [680…
```
data_cat.drop(["price change"], axis = 1, inplace = True)
```

In [681…
```
num_data = datscaled
cat_data = data_cat
dataframe = pd.concat([num_data, cat_data], axis = 1)
```

In [682…
```
dataframe.head(10)
```

Out[682…

	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	Net Income	Ear
0	-0.393341	0.493950	0.272749	0.989601	-0.210698	-0.339355	1.554415	1.3
1	-0.220837	0.355439	1.137045	0.937737	0.077269	-0.002335	0.927628	0.0
2	-0.367195	0.602479	-0.427007	-0.192905	-0.033488	0.454058	0.744371	0.0
3	0.133567	0.825696	-0.284802	-0.317379	1.218059	-0.152497	-0.219816	-0.2
4	-0.260874	-0.492636	0.296470	-0.265515	2.237018	0.133564	-0.202703	-0.3
5	-0.451251	-1.342556	-0.016049	-0.307006	-0.232849	-0.125823	0.090133	0.0
6	1.998837	0.176091	-0.692132	-0.099549	-0.498664	0.018187	-0.228206	0.9
7	-0.384353	-0.158797	-0.679931	-0.317379	-0.620496	0.119096	-0.218177	-0.0
8	-0.230744	-0.142338	-0.774192	-0.296633	-0.675874	-0.021424	0.141806	0.0
9	-0.214096	-0.087667	-0.808359	-0.265515	0.320933	-0.187053	0.263986	0.4

Handling outliers

In [684…
```
dataframe.describe()
```

Out[684…

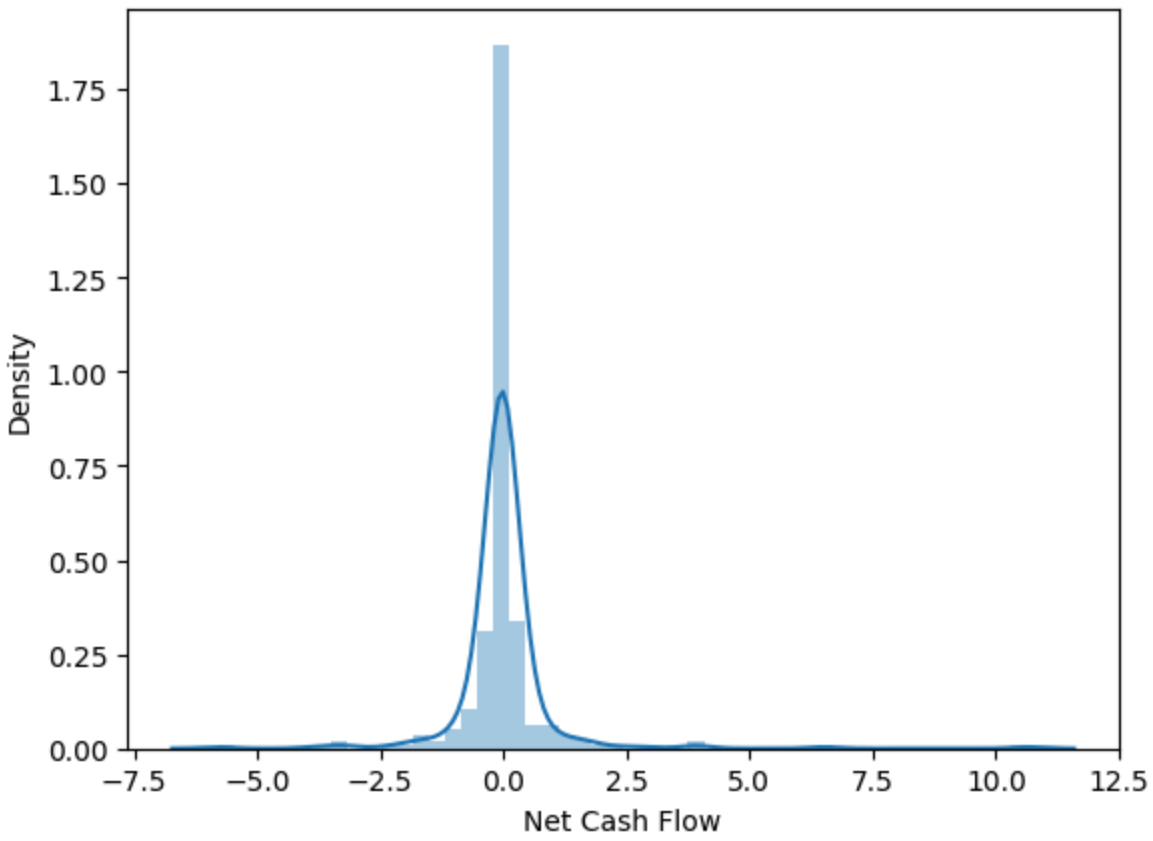
	Current Price	Price Change	Volatility	ROE	Cash Ratio
count	3.400000e+02	3.400000e+02	3.400000e+02	3.400000e+02	3.400000e+02
mean	7.836868e-18	-2.873518e-17	-7.967483e-17	2.873518e-17	-8.098097e-17
std	1.001474e+00	1.001474e+00	1.001474e+00	1.001474e+00	1.001474e+00
min	-7.799176e-01	-4.271357e+00	-1.341642e+00	-4.003618e-01	-7.755552e-01
25%	-4.321010e-01	-4.185350e-01	-6.618376e-01	-3.095993e-01	-5.761937e-01
50%	-2.160880e-01	6.183430e-02	-2.375644e-01	-2.551418e-01	-2.550003e-01
75%	1.227409e-01	5.519628e-01	2.869602e-01	-1.306675e-01	3.209329e-01
max	1.219567e+01	4.251806e+00	5.168258e+00	9.101176e+00	9.834905e+00

In [685…

```
## using seaborn librarie
sns.distplot(dataframe["Net Cash Flow"])
```

```
C:\Users\NYB  COMPUTER\AppData\Local\Temp\ipykernel_19664\855308096.py:2: UserW
arning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataframe["Net Cash Flow"])
```

Out[685… <Axes: xlabel='Net Cash Flow', ylabel='Density'>

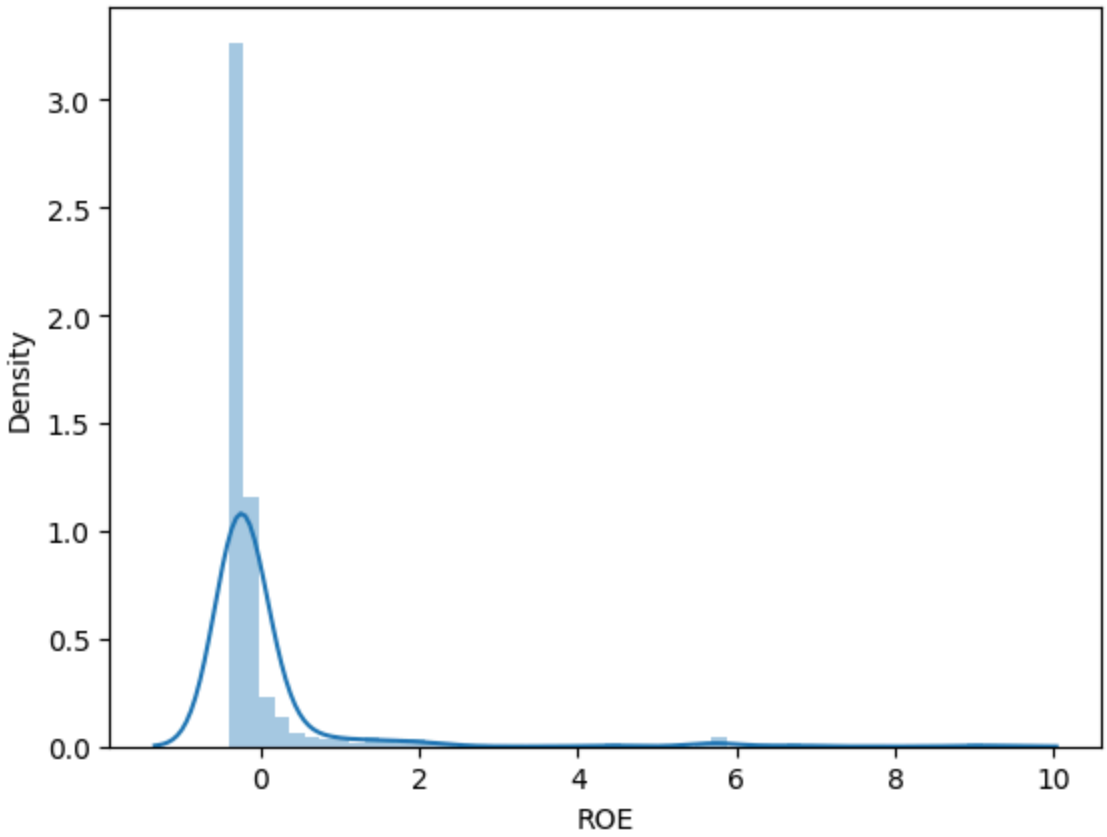


In [686…
```
def handle_outlier(col):
    dataframe[col] = np.log1p(dataframe[col])
```

In [687…
```
## using seaborn librarie
sns.distplot(dataframe["ROE"])
```

```
C:\Users\NYB  COMPUTER\AppData\Local\Temp\ipykernel_19664\2226377044.py:2: User
Warning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataframe["ROE"])
```

Out[687… <Axes: xlabel='ROE', ylabel='Density'>

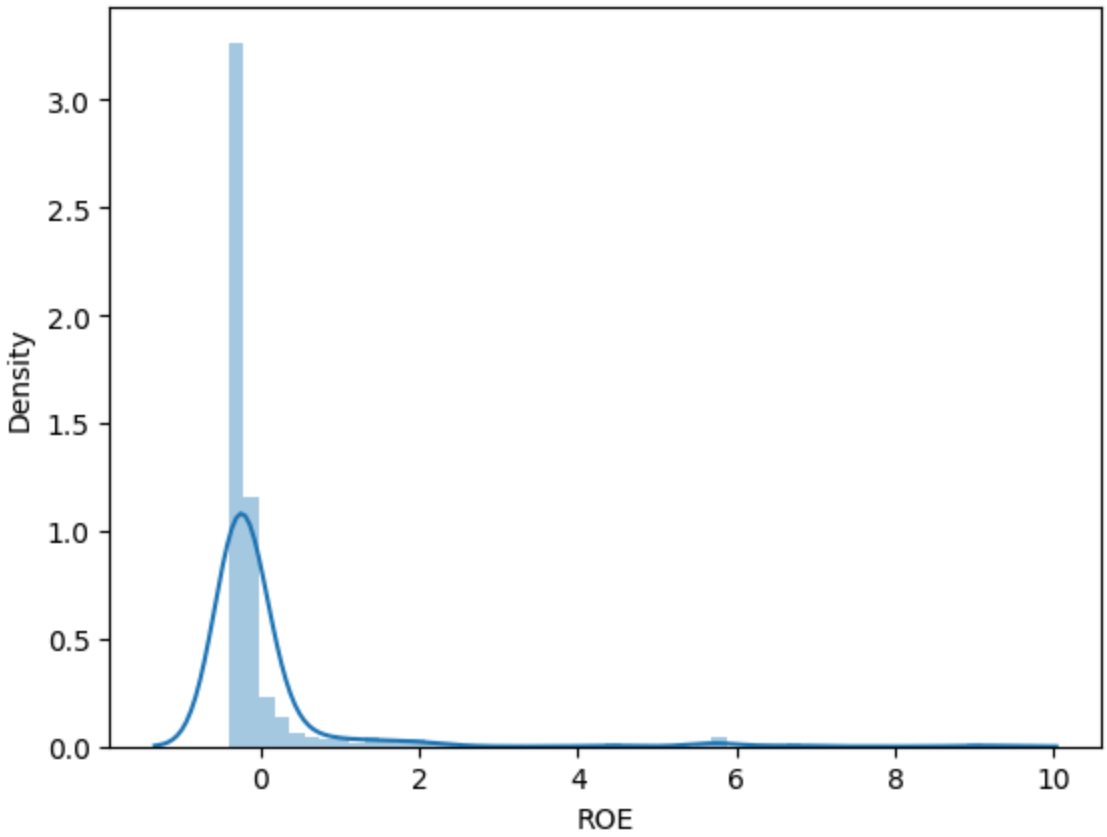


In [688…
```
def handle_outlier(col):
    dataframe[col] = np.log1p(dataframe[col])
```

In [689…
```
sns.distplot(dataframe["ROE"])
```

```
C:\Users\NYB  COMPUTER\AppData\Local\Temp\ipykernel_19664\3600362287.py:1: User
Warning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataframe["ROE"])
```

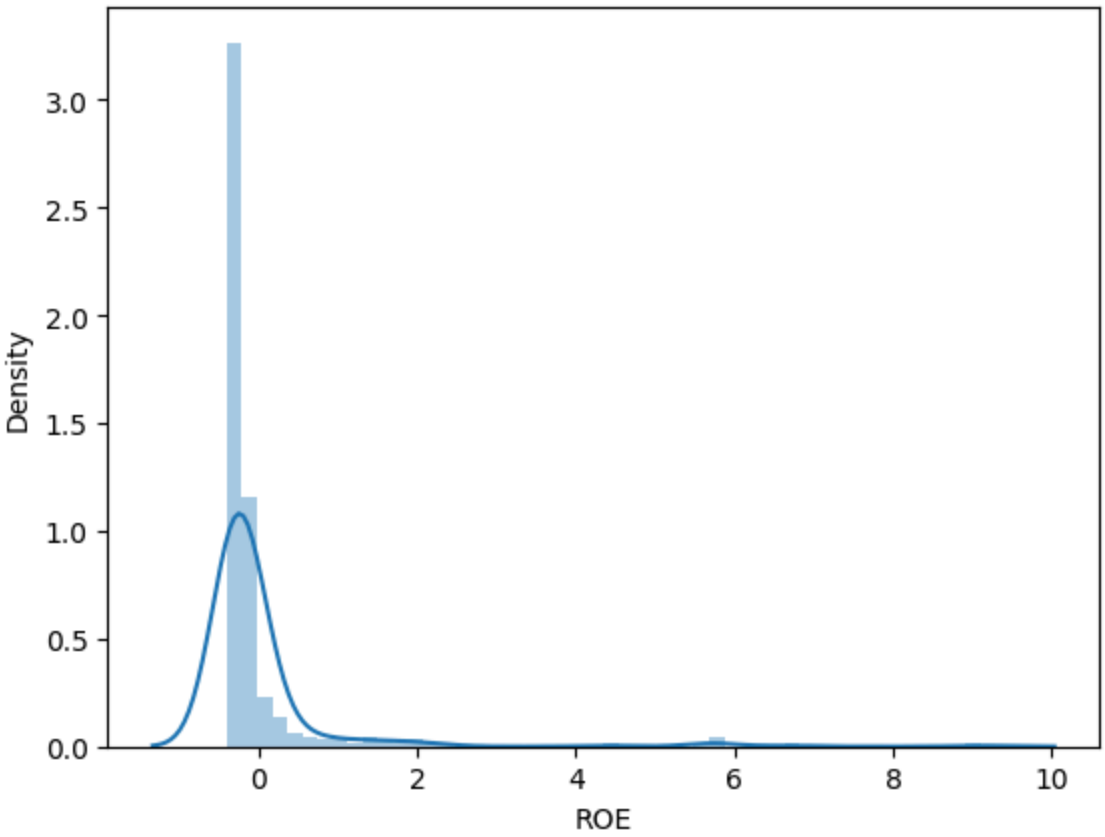
Out[689… <Axes: xlabel='ROE', ylabel='Density'>



In [690…
```
## using seaborn librarie
sns.distplot(dataframe["ROE"].dropna())
```

```
C:\Users\NYB  COMPUTER\AppData\Local\Temp\ipykernel_19664\3647616435.py:2: User
Warning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(dataframe["ROE"].dropna())
```

Out[690… <Axes: xlabel='ROE', ylabel='Density'>



In [691…
```
dataframe.isna().sum()
```

Out[691…
```
Current Price                   0
Price Change                    0
Volatility                      0
ROE                             0
Cash Ratio                      0
Net Cash Flow                   0
Net Income                      0
Earnings Per Share              0
Estimated Shares Outstanding    0
P/E Ratio                       0
P/B Ratio                       0
Ticker Symbol                   0
Security                        0
GICS Sector                     0
GICS Sub Industry               0
dtype: int64
```

In [692…
```
dataframe.isnull().sum()
```

Out[692…
```
Current Price                   0
Price Change                    0
Volatility                      0
ROE                             0
Cash Ratio                      0
Net Cash Flow                   0
Net Income                      0
Earnings Per Share              0
Estimated Shares Outstanding    0
P/E Ratio                       0
P/B Ratio                       0
Ticker Symbol                   0
Security                        0
GICS Sector                     0
GICS Sub Industry               0
dtype: int64
```

EDA

- It is a good idea to explore the data once again after manipulating it.

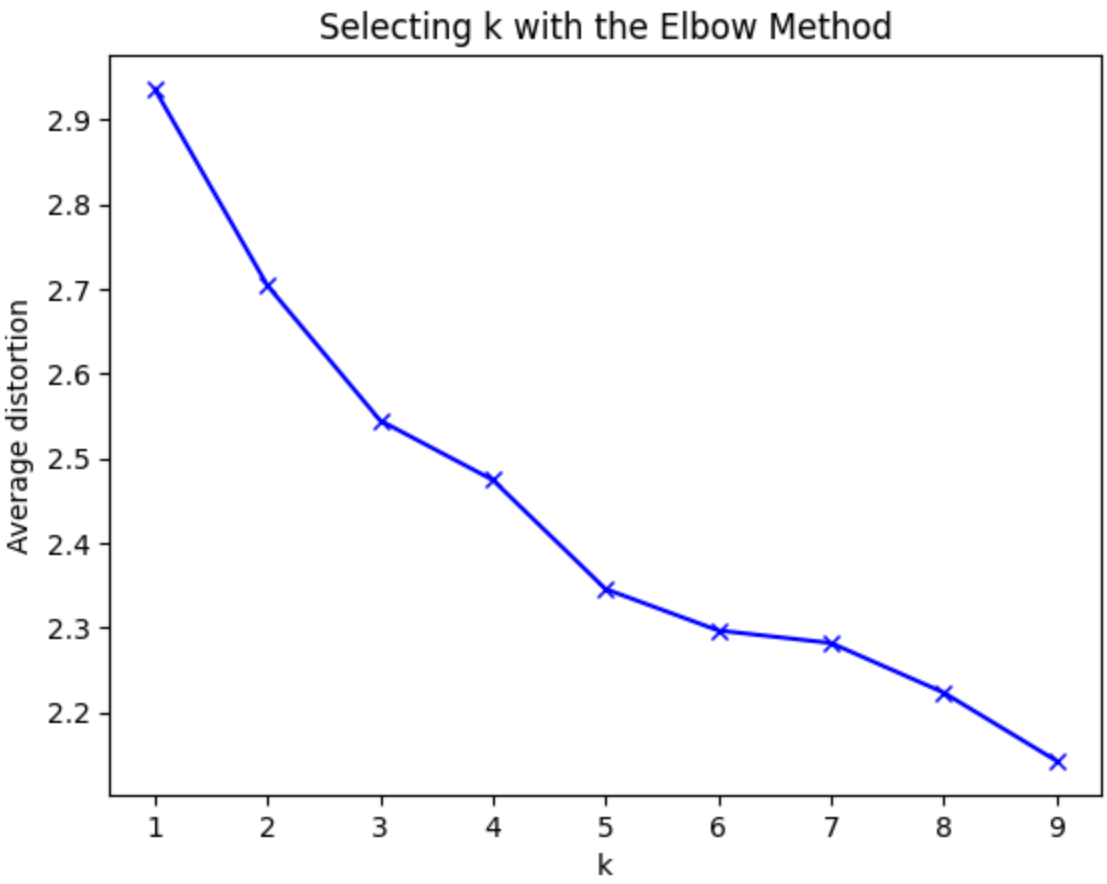
In [ ]:
```
```

K-means Clustering

In [695…
```
#Finding optimal no. of clusters
from scipy.spatial.distance import cdist
clusters=range(1,10)
meanDistortions=[]

for k in clusters:
    model=KMeans(n_clusters=k)
    model.fit(dataframe)
    prediction=model.predict(dataframe)
    meanDistortions.append(sum(np.min(cdist(dataframe, model.cluster_centers_,


plt.plot(clusters, meanDistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

```
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the
warning
  warnings.warn(
```

Out[695… Text(0.5, 1.0, 'Selecting k with the Elbow Method')



From the figure above, the elbow shows that the minimum number of clusters is 3 as the is a sharp bend at that point with an average distortion of 2.53

In [697…

```
# Let us first start with K = 3
final_model=KMeans(3)
final_model.fit(dataframe)
prediction=final_model.predict(dataframe)

#Append the prediction
data["GROUP"] = prediction
dataframe["GROUP"] = prediction
print("Groups Assigned : \n")
data.head(10)
```

```
Groups Assigned :
```

```
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

Out[697…

	Ticker Symbol	Security	GICS Sector	GICS Sub Industry	Current Price	Price Change	Volat
0	AAL	American Airlines Group	Industrials	Airlines	42.349998	9.999995	1.68
1	ABBV	AbbVie	Health Care	Pharmaceuticals	59.240002	8.339433	2.19
2	ABT	Abbott Laboratories	Health Care	Health Care Equipment	44.910000	11.301121	1.27
3	ADBE	Adobe Systems Inc	Information Technology	Application Software	93.940002	13.977195	1.35
4	ADI	Analog Devices, Inc.	Information Technology	Semiconductors	55.320000	-1.827858	1.70
5	ADM	Archer-Daniels-Midland Co	Consumer Staples	Agricultural Products	36.680000	-12.017268	1.51
6	ADS	Alliance Data Systems	Information Technology	Data Processing & Outsourced Services	276.570007	6.189286	1.11
7	AEE	Ameren Corp	Utilities	MultiUtilities	43.230000	2.174424	1.12
8	AEP	American Electric Power	Utilities	Electric Utilities	58.270000	2.371753	1.06
9	AFL	AFLAC Inc	Financials	Life & Health Insurance	59.900002	3.027181	1.04

In [774…

```
dataClust = dataframe.groupby(['GROUP'])
dataClust.mean()
```

Out[774…

GROUP	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	Net Income
0	0.171612	0.564665	-0.260037	6.068844	-0.382370	-0.088245	-1.230970
1	0.594812	1.208406	0.116382	-0.110926	0.757349	-0.021019	-0.135470
2	-0.280760	-1.766661	1.833283	0.174240	-0.132877	-0.067309	-0.887208
3	-0.106861	-0.056985	-0.304738	-0.157786	-0.180848	0.048312	0.054733
4	-0.309926	0.139248	-0.669418	-0.088233	0.065186	-0.580298	3.390319

In [ ]:

In [699…

```
#Let us first start with K = 5
final_model=KMeans(5)
final_model.fit(dataframe)
prediction=final_model.predict(dataframe)

#Append the prediction
data["GROUP"] = prediction
dataframe["GROUP"] = prediction
print("Groups Assigned : \n")
data.head()
```

```
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
Groups Assigned :
```

Out[699…

	Ticker Symbol	Security	GICS Sector	GICS Sub Industry	Current Price	Price Change	Volatili
0	AAL	American Airlines Group	Industrials	Airlines	42.349998	9.999995	1.6871
1	ABBV	AbbVie	Health Care	Pharmaceuticals	59.240002	8.339433	2.1978
2	ABT	Abbott Laboratories	Health Care	Health Care Equipment	44.910000	11.301121	1.2736
3	ADBE	Adobe Systems Inc	Information Technology	Application Software	93.940002	13.977195	1.3576
4	ADI	Analog Devices, Inc.	Information Technology	Semiconductors	55.320000	-1.827858	1.7011

In [772…

```
dataClust = dataframe.groupby(['GROUP'])
dataClust.mean()
```

Out[772…

GROUP	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	Net Income
0	0.171612	0.564665	-0.260037	6.068844	-0.382370	-0.088245	-1.230970
1	0.594812	1.208406	0.116382	-0.110926	0.757349	-0.021019	-0.135470
2	-0.280760	-1.766661	1.833283	0.174240	-0.132877	-0.067309	-0.887208
3	-0.106861	-0.056985	-0.304738	-0.157786	-0.180848	0.048312	0.054733
4	-0.309926	0.139248	-0.669418	-0.088233	0.065186	-0.580298	3.390319

Observation

- Grooup "2" is the most popular in current price followed by group "0"
- Group 1 is the most popular in terms of ROE with 603

In [ ]:

```
```

Hierarchical Clustering

In [703…

```
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters = 3, affinity = "euclidean", linkag
model.fit(dataframe)
```

```
C:\Users\NYB  COMPUTER\AppData\Roaming\Python\Python311\site-packages\sklearn\c
luster\_agglomerative.py:983: FutureWarning: Attribute `affinity` was deprecate
d in version 1.2 and will be removed in 1.4. Use `metric` instead
  warnings.warn(
```

Out[703…

```
AgglomerativeClustering(affinity='euclidean', linkage='average', n_cl
usters=3)
```

In [704…

```
dataframe["labels"] = model.labels_
dataframe.head(10)
```

Out[704…

	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	Net Income	Ear
0	-0.393341	0.493950	0.272749	0.989601	-0.210698	-0.339355	1.554415	1.3
1	-0.220837	0.355439	1.137045	0.937737	0.077269	-0.002335	0.927628	0.0
2	-0.367195	0.602479	-0.427007	-0.192905	-0.033488	0.454058	0.744371	0.0
3	0.133567	0.825696	-0.284802	-0.317379	1.218059	-0.152497	-0.219816	-0.2
4	-0.260874	-0.492636	0.296470	-0.265515	2.237018	0.133564	-0.202703	-0.3
5	-0.451251	-1.342556	-0.016049	-0.307006	-0.232849	-0.125823	0.090133	0.0
6	1.998837	0.176091	-0.692132	-0.099549	-0.498664	0.018187	-0.228206	0.9
7	-0.384353	-0.158797	-0.679931	-0.317379	-0.620496	0.119096	-0.218177	-0.0
8	-0.230744	-0.142338	-0.774192	-0.296633	-0.675874	-0.021424	0.141806	0.0
9	-0.214096	-0.087667	-0.808359	-0.265515	0.320933	-0.187053	0.263986	0.4

In [705…

```
datagrp = dataframe.groupby(["labels"])
datagrp.mean()
```

Out[705…

labels	Current Price	Price Change	Volatility	ROE	Cash Ratio	Net Cash Flow	Net Income
0	-0.034982	-0.001587	-0.003113	-0.026601	-0.004062	0.001651	0.018022
1	-0.371689	0.610544	1.488223	9.101176	0.110496	0.330570	-6.359977
2	12.195670	-0.074042	-0.435986	-0.109922	1.262362	-0.888563	0.268653

In [706…

```
from scipy.cluster.hierarchy import cophenet, dendrogram, linkage
from scipy.spatial.distance import pdist ## pairwise distribution between data
```

In [707…

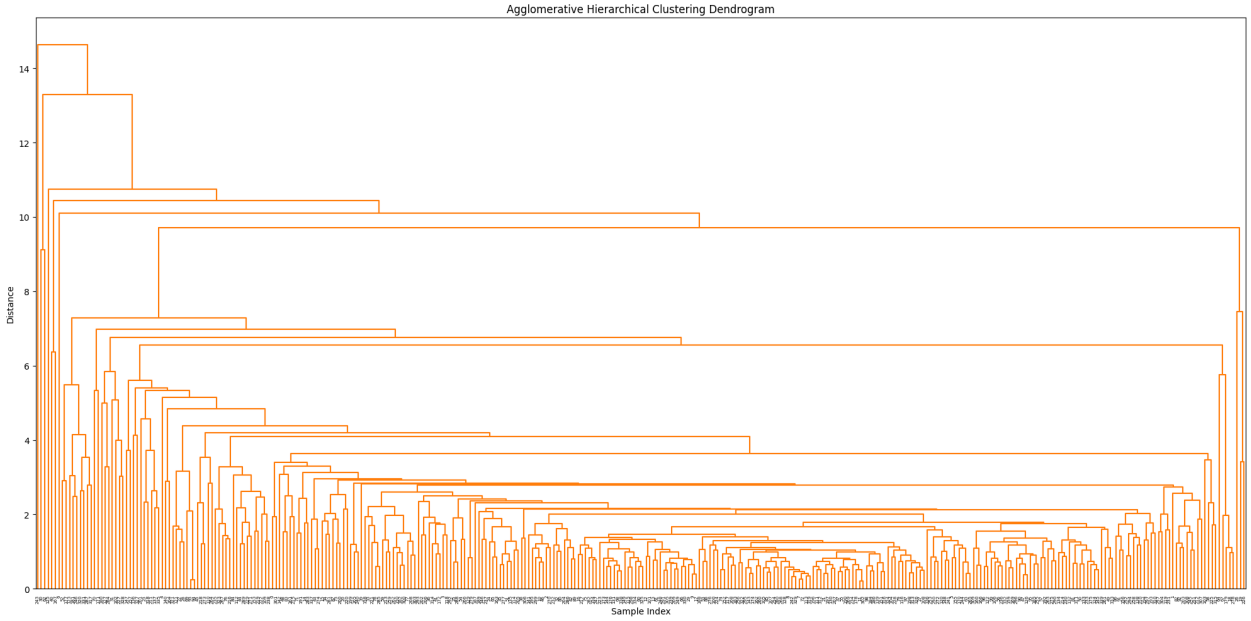
```
z = linkage(datscaled, metric = "euclidean", method = "average")
c, coph_dists = cophenet(z, pdist(datscaled))
c
```

Out[707…

```
0.9422540609560814
```

In [708…

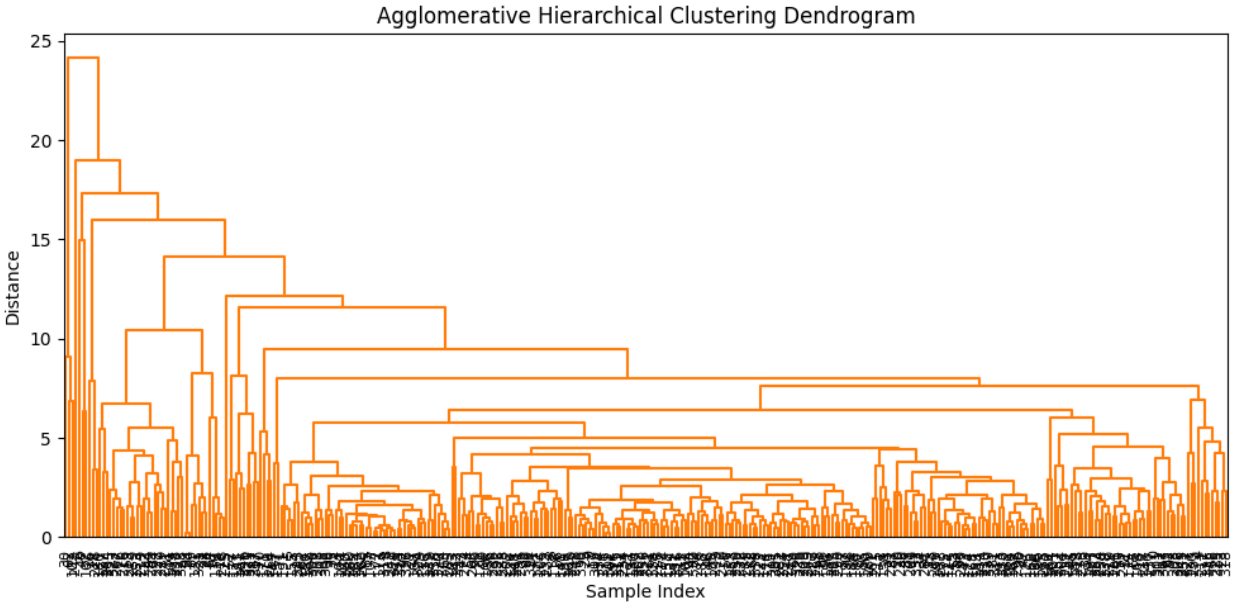
```
plt.figure(figsize = (20,10))
plt.title("Agglomerative Hierarchical Clustering Dendrogram")
plt.xlabel("Sample Index")
plt.ylabel("Distance")
dendrogram(z, leaf_rotation = 90, color_threshold = 42, leaf_font_size = 5)
plt.tight_layout()
```



In [709…
```
z = linkage(datscaled, metric = "euclidean", method = "complete")
c, coph_dists = cophenet(z, pdist(datscaled))
c
```

Out[709…
```
0.7873280186580672
```

In [710…
```
plt.figure(figsize = (10,5))
plt.title("Agglomerative Hierarchical Clustering Dendrogram")
plt.xlabel("Sample Index")
plt.ylabel("Distance")
dendrogram(z, leaf_rotation = 90, color_threshold = 40, leaf_font_size = 8)
plt.tight_layout()
```



In [ ]:

In [ ]:

In [ ]:

K-means vs Hierarchical Clustering

You compare several things, like:

- Which clustering technique took less time for execution?
- Which clustering technique gave you more distinct clusters, or are they the same?
- How many observations are there in the similar clusters of both algorithms?
- How many clusters are obtained as the appropriate number of clusters from both algorithms?

You can also mention any differences or similarities you obtained in the cluster profiles from both the clustering techniques.

Observations

- The kmeans clustering took the least time to be executed
- The Hierarchical clustering gives the most distint clusters
- From the two algorithms, it shows that we can only have a minimum of 3 clusters
- the number of clusters that both algoriyhms predict is 4

In [ ]:

Actionable Insights and Recommendations

- For group "0", security and price change contribute ~ 0.56
- For group "1", Price change, Security and Ticker Symbolare the maincontributorsof this group. They contribute greatly to the positive increase in price
- For group 2, only volatility contributes 1.83 to the change in price change of the stocks in the stock market
- For group 4, Estimated shares outstanding, Net Income, contribute greatly to influence price change
- I will recommend that one can invest in group 4, group 0, group 2, because they total a positive increase in the price change.

In [ ]: