A Predictive Approach With PyTorch

New York Stock Exchange ANALYSIS

DATA

The project is completed on a dataset of New Stock Exchange. The Data is divided into 4 Excel sheets where

- Prices.csv: raw, as-is daily prices. Most of data spans from 2010 to the end 2016, for companies new on stock market date range is shorter. There have been approx. 140 stock splits in that time, this set doesn't account for that.
- Prices-split-adjusted.csv: same as prices, but there have been added adjustments for splits.
- Securities.csv: general description of each company with division on sectors
- Fundamentals.csv: metrics extracted from annual SEC 10K fillings (2012-2016), should be enough to derive most of popular fundamental indicators.

```python
#data manipulation
import pandas as pd
import numpy as np

 #data visualization
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
from tabulate import tabulate
from sklearn.inspection import DecisionBoundaryDisplay

#Model Building
# for splititng of data
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree, export_text
from sklearn.preprocessing import LabelEncoder

#for building linear regression model
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error

#for building logistic regression model
from sklearn.linear_model import LogisticRegression

#for ignoring warnings
import warnings
warnings.filterwarnings('ignore')

# for decisions tree
from sklearn.tree import DecisionTreeClassifier

# for random forest algorithm
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier

# for naive-bayes algorithm
from sklearn.naive_bayes import GaussianNB

#for K-Nearest Neighbour algorithm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_classification
from sklearn.preprocessing import StandardScaler
```

```python
#for support-vector mechanism
from sklearn.svm import SVC

#for Extreme Gradient Boosting
import xgboost as xgb

#Pytorch for extensive deep learning
import torch
import torch.nn as nn
import numpy as np
from torch.utils.data import Dataset, DataLoader

funda = pd.read_csv('/content/fundamentals.csv')
prsp = pd.read_csv('/content/prices-split-adjusted.csv')
prc = pd.read_csv('/content/prices.csv')
sec = pd.read_csv('/content/securities.csv')

funda.isnull().sum()
```

```
Unnamed: 0                          0
Ticker Symbol                       0
Period Ending                       0
Accounts Payable                    0
Accounts Receivable                 0
                                ...
Total Revenue                       0
Treasury Stock                      0
For Year                          173
Earnings Per Share                219
Estimated Shares Outstanding      219
Length: 79, dtype: int64
```

```python
funda.shape
```

```
(1781, 79)
```

```python
prsp.isnull().sum()
```

```
date      0
symbol    0
open      0
close     0
low       0
high      0
volume    0
dtype: int64
```

```python
prsp.shape
```

```
(851264, 7)
```

```python
prc.isnull().sum()
```

```
date      0
symbol    0
open      0
close     0
low       0
high      0
volume    0
dtype: int64
```

```
prc.shape
```

```
(851264, 7)
```

```
sec.isnull().sum()
```

```
Ticker symbol                0
Security                     0
SEC filings                  0
GICS Sector                  0
GICS Sub Industry            0
Address of Headquarters      0
Date first added           198
CIK                          0
dtype: int64
```

```
sec.shape
```

```
(505, 8)
```

```
funda['For Year'] = pd.to_datetime(funda['For
Year']).fillna(method='ffill')
funda['Earnings Per Share'] = pd.to_datetime(funda['Earnings Per
Share']).fillna(method='ffill')
funda['Estimated Shares Outstanding'] =
pd.to_datetime(funda['Estimated Shares
Outstanding']).fillna(method='ffill')
```

```
sec['Date first added'] = pd.to_datetime(sec['Date first
added']).fillna(method='ffill')
```

```
funda.columns
```

```
Index(['Unnamed: 0', 'Ticker Symbol', 'Period Ending', 'Accounts
Payable',
       'Accounts Receivable', 'Add'l income/expense items', 'After Tax
ROE',
       'Capital Expenditures', 'Capital Surplus', 'Cash Ratio',
       'Cash and Cash Equivalents', 'Changes in Inventories', 'Common
Stocks',
       'Cost of Revenue', 'Current Ratio', 'Deferred Asset Charges',
       'Deferred Liability Charges', 'Depreciation',
```

```
       'Earnings Before Interest and Tax', 'Earnings Before Tax',
       'Effect of Exchange Rate',
       'Equity Earnings/Loss Unconsolidated Subsidiary', 'Fixed
Assets',
       'Goodwill', 'Gross Margin', 'Gross Profit', 'Income Tax',
       'Intangible Assets', 'Interest Expense', 'Inventory',
'Investments',
       'Liabilities', 'Long-Term Debt', 'Long-Term Investments',
       'Minority Interest', 'Misc. Stocks', 'Net Borrowings', 'Net
Cash Flow',
       'Net Cash Flow-Operating', 'Net Cash Flows-Financing',
       'Net Cash Flows-Investing', 'Net Income', 'Net Income
Adjustments',
       'Net Income Applicable to Common Shareholders',
       'Net Income-Cont. Operations', 'Net Receivables', 'Non-
Recurring Items',
       'Operating Income', 'Operating Margin', 'Other Assets',
       'Other Current Assets', 'Other Current Liabilities', 'Other
Equity',
       'Other Financing Activities', 'Other Investing Activities',
       'Other Liabilities', 'Other Operating Activities',
       'Other Operating Items', 'Pre-Tax Margin', 'Pre-Tax ROE',
       'Profit Margin', 'Quick Ratio', 'Research and Development',
       'Retained Earnings', 'Sale and Purchase of Stock',
       'Sales, General and Admin.',
       'Short-Term Debt / Current Portion of Long-Term Debt',
       'Short-Term Investments', 'Total Assets', 'Total Current
Assets',
       'Total Current Liabilities', 'Total Equity', 'Total
Liabilities',
       'Total Liabilities & Equity', 'Total Revenue', 'Treasury
Stock',
       'For Year', 'Earnings Per Share', 'Estimated Shares
Outstanding'],
      dtype='object')
```

```
columns_of_interest = ['Accounts Payable', 'Accounts Receivable',
'Cash Ratio',
                       'Earnings Before Tax', 'Net Income', 'Operating
Margin',
                       'Total Revenue', 'Total Assets', 'Total
Liabilities']

funda_subset = funda[columns_of_interest].dropna()

z_scores = (funda_subset - funda_subset.mean()) / funda_subset.std()

threshold = 3
```

```
outliers = z_scores[(z_scores > threshold).any(axis=1)]

for ticker_symbol in outliers.index:
    outlier_categories = outliers.loc[ticker_symbol]
[outliers.loc[ticker_symbol] > threshold].index
    print("Ticker symbol:", funda.loc[ticker_symbol, 'Ticker Symbol'])
    print("Outlier categories:", list(outlier_categories))
    print("Average amount for each category:")
    for category in outlier_categories:
        average_amount = funda_subset[category].mean()
        outlier_count = (outliers.loc[ticker_symbol] >
threshold).sum()
        print(f"- {category}: Average = {average_amount:.2f}, Number
of outliers = {outlier_count}")
    print()
```

```
Ticker symbol: AAPL
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 5
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 5
- Net Income: Average = 1632918735.49, Number of outliers = 5
- Total Revenue: Average = 20772269312.42, Number of outliers = 5
- Total Assets: Average = 26595844854.93, Number of outliers = 5

Ticker symbol: AAPL
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: AAPL
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: AAPL
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
```

```
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: ADI
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: ADI
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: ADI
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: ADM
Outlier categories: ['Accounts Receivable']
Average amount for each category:
- Accounts Receivable: Average = -74812071.52, Number of outliers = 1

Ticker symbol: AMGN
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: AMZN
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: AMZN
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: AMZN
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: APA
```

```
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: APC
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: BA
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: BA
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: BA
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: BA
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: BHI
Outlier categories: ['Accounts Receivable']
Average amount for each category:
- Accounts Receivable: Average = -74812071.52, Number of outliers = 1

Ticker symbol: CHK
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: CHTR
Outlier categories: ['Total Liabilities']
Average amount for each category:
- Total Liabilities: Average = 17137870717.27, Number of outliers = 1

Ticker symbol: COTY
Outlier categories: ['Cash Ratio', 'Operating Margin']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 2
- Operating Margin: Average = 18.61, Number of outliers = 2
```

```
Ticker symbol: CVS
Outlier categories: ['Accounts Receivable', 'Total Revenue']
Average amount for each category:
- Accounts Receivable: Average = -74812071.52, Number of outliers = 2
- Total Revenue: Average = 20772269312.42, Number of outliers = 2

Ticker symbol: CVS
Outlier categories: ['Total Revenue']
Average amount for each category:
- Total Revenue: Average = 20772269312.42, Number of outliers = 1

Ticker symbol: CVX
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 5
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 5
- Net Income: Average = 1632918735.49, Number of outliers = 5
- Total Revenue: Average = 20772269312.42, Number of outliers = 5
- Total Assets: Average = 26595844854.93, Number of outliers = 5

Ticker symbol: CVX
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: CVX
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: CVX
Outlier categories: ['Total Assets', 'Total Liabilities']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: DVN
```

```
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: F
Outlier categories: ['Total Assets', 'Total Liabilities']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: F
Outlier categories: ['Total Assets', 'Total Liabilities']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: F
Outlier categories: ['Total Assets', 'Total Liabilities']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: F
Outlier categories: ['Total Revenue', 'Total Assets', 'Total
Liabilities']
Average amount for each category:
- Total Revenue: Average = 20772269312.42, Number of outliers = 3
- Total Assets: Average = 26595844854.93, Number of outliers = 3
- Total Liabilities: Average = 17137870717.27, Number of outliers = 3

Ticker symbol: FB
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: FB
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: FB
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: FB
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1
```

```
Ticker symbol: FCX
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: FTR
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: GILD
Outlier categories: ['Earnings Before Tax', 'Net Income']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 2
- Net Income: Average = 1632918735.49, Number of outliers = 2

Ticker symbol: GM
Outlier categories: ['Accounts Payable', 'Total Revenue', 'Total
Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 4
- Total Revenue: Average = 20772269312.42, Number of outliers = 4
- Total Assets: Average = 26595844854.93, Number of outliers = 4
- Total Liabilities: Average = 17137870717.27, Number of outliers = 4

Ticker symbol: GM
Outlier categories: ['Accounts Payable', 'Total Revenue', 'Total
Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 4
- Total Revenue: Average = 20772269312.42, Number of outliers = 4
- Total Assets: Average = 26595844854.93, Number of outliers = 4
- Total Liabilities: Average = 17137870717.27, Number of outliers = 4

Ticker symbol: GM
Outlier categories: ['Accounts Payable', 'Total Revenue', 'Total
Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 4
- Total Revenue: Average = 20772269312.42, Number of outliers = 4
- Total Assets: Average = 26595844854.93, Number of outliers = 4
- Total Liabilities: Average = 17137870717.27, Number of outliers = 4

Ticker symbol: GM
Outlier categories: ['Accounts Payable', 'Total Revenue', 'Total
Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 4
- Total Revenue: Average = 20772269312.42, Number of outliers = 4
- Total Assets: Average = 26595844854.93, Number of outliers = 4
```

```
- Total Liabilities: Average = 17137870717.27, Number of outliers = 4

Ticker symbol: HPQ
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: HPQ
Outlier categories: ['Accounts Payable', 'Accounts Receivable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 2
- Accounts Receivable: Average = -74812071.52, Number of outliers = 2

Ticker symbol: IBM
Outlier categories: ['Earnings Before Tax', 'Net Income']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 2
- Net Income: Average = 1632918735.49, Number of outliers = 2

Ticker symbol: IBM
Outlier categories: ['Earnings Before Tax', 'Net Income', 'Total
Liabilities']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 3
- Net Income: Average = 1632918735.49, Number of outliers = 3
- Total Liabilities: Average = 17137870717.27, Number of outliers = 3

Ticker symbol: IBM
Outlier categories: ['Earnings Before Tax', 'Total Liabilities']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: ISRG
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: LLTC
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: LLTC
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: LLTC
Outlier categories: ['Cash Ratio']
```

```
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: LUK
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: MCHP
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: MCHP
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: MCHP
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: MCHP
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: MCK
Outlier categories: ['Accounts Payable']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 1

Ticker symbol: MCK
Outlier categories: ['Accounts Payable', 'Total Revenue']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 2
- Total Revenue: Average = 20772269312.42, Number of outliers = 2

Ticker symbol: MCK
Outlier categories: ['Accounts Payable', 'Total Revenue']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 2
- Total Revenue: Average = 20772269312.42, Number of outliers = 2

Ticker symbol: MNST
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1
```

```
Ticker symbol: MPC
Outlier categories: ['Accounts Receivable']
Average amount for each category:
- Accounts Receivable: Average = -74812071.52, Number of outliers = 1

Ticker symbol: MSFT
Outlier categories: ['Earnings Before Tax', 'Net Income']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 2
- Net Income: Average = 1632918735.49, Number of outliers = 2

Ticker symbol: MSFT
Outlier categories: ['Earnings Before Tax', 'Net Income', 'Total
Assets']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 3
- Net Income: Average = 1632918735.49, Number of outliers = 3
- Total Assets: Average = 26595844854.93, Number of outliers = 3

Ticker symbol: MSFT
Outlier categories: ['Total Assets']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 1

Ticker symbol: MSFT
Outlier categories: ['Earnings Before Tax', 'Net Income', 'Total
Assets', 'Total Liabilities']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 4
- Net Income: Average = 1632918735.49, Number of outliers = 4
- Total Assets: Average = 26595844854.93, Number of outliers = 4
- Total Liabilities: Average = 17137870717.27, Number of outliers = 4

Ticker symbol: MUR
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: NFX
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: NFX
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: NOV
Outlier categories: ['Accounts Receivable']
```

```
Average amount for each category:
- Accounts Receivable: Average = -74812071.52, Number of outliers = 1

Ticker symbol: NVDA
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: NVDA
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: NVDA
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: PCLN
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: PFE
Outlier categories: ['Net Income', 'Total Assets', 'Total
Liabilities']
Average amount for each category:
- Net Income: Average = 1632918735.49, Number of outliers = 3
- Total Assets: Average = 26595844854.93, Number of outliers = 3
- Total Liabilities: Average = 17137870717.27, Number of outliers = 3

Ticker symbol: PFE
Outlier categories: ['Net Income', 'Total Assets']
Average amount for each category:
- Net Income: Average = 1632918735.49, Number of outliers = 2
- Total Assets: Average = 26595844854.93, Number of outliers = 2

Ticker symbol: PFE
Outlier categories: ['Total Assets']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 1

Ticker symbol: PFE
Outlier categories: ['Total Assets', 'Total Liabilities']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: PSX
Outlier categories: ['Total Revenue']
```

```
Average amount for each category:
- Total Revenue: Average = 20772269312.42, Number of outliers = 1

Ticker symbol: PSX
Outlier categories: ['Total Revenue']
Average amount for each category:
- Total Revenue: Average = 20772269312.42, Number of outliers = 1

Ticker symbol: PSX
Outlier categories: ['Accounts Receivable', 'Total Revenue']
Average amount for each category:
- Accounts Receivable: Average = -74812071.52, Number of outliers = 2
- Total Revenue: Average = 20772269312.42, Number of outliers = 2

Ticker symbol: PSX
Outlier categories: ['Accounts Receivable']
Average amount for each category:
- Accounts Receivable: Average = -74812071.52, Number of outliers = 1

Ticker symbol: SWKS
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: SWN
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: T
Outlier categories: ['Accounts Payable', 'Total Assets', 'Total
Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 3
- Total Assets: Average = 26595844854.93, Number of outliers = 3
- Total Liabilities: Average = 17137870717.27, Number of outliers = 3

Ticker symbol: T
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 5
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 5
- Net Income: Average = 1632918735.49, Number of outliers = 5
- Total Assets: Average = 26595844854.93, Number of outliers = 5
- Total Liabilities: Average = 17137870717.27, Number of outliers = 5

Ticker symbol: T
Outlier categories: ['Accounts Payable', 'Total Assets', 'Total
Liabilities']
```

```
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 3
- Total Assets: Average = 26595844854.93, Number of outliers = 3
- Total Liabilities: Average = 17137870717.27, Number of outliers = 3

Ticker symbol: T
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Total
Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 4
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 4
- Total Assets: Average = 26595844854.93, Number of outliers = 4
- Total Liabilities: Average = 17137870717.27, Number of outliers = 4

Ticker symbol: VRTX
Outlier categories: ['Operating Margin']
Average amount for each category:
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: VZ
Outlier categories: ['Total Assets', 'Total Liabilities']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: VZ
Outlier categories: ['Earnings Before Tax', 'Total Assets', 'Total
Liabilities']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 3
- Total Assets: Average = 26595844854.93, Number of outliers = 3
- Total Liabilities: Average = 17137870717.27, Number of outliers = 3

Ticker symbol: VZ
Outlier categories: ['Total Assets', 'Total Liabilities']
Average amount for each category:
- Total Assets: Average = 26595844854.93, Number of outliers = 2
- Total Liabilities: Average = 17137870717.27, Number of outliers = 2

Ticker symbol: VZ
Outlier categories: ['Earnings Before Tax', 'Net Income', 'Total
Assets', 'Total Liabilities']
Average amount for each category:
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 4
- Net Income: Average = 1632918735.49, Number of outliers = 4
- Total Assets: Average = 26595844854.93, Number of outliers = 4
- Total Liabilities: Average = 17137870717.27, Number of outliers = 4

Ticker symbol: WAT
Outlier categories: ['Cash Ratio']
```

```
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: WMT
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: WMT
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: WMT
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: WMT
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: XEC
Outlier categories: ['Operating Margin']
Average amount for each category:
```

```
- Operating Margin: Average = 18.61, Number of outliers = 1

Ticker symbol: XLNX
Outlier categories: ['Cash Ratio']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 1

Ticker symbol: XOM
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: XOM
Outlier categories: ['Accounts Payable', 'Earnings Before Tax', 'Net
Income', 'Total Revenue', 'Total Assets', 'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 6
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 6
- Net Income: Average = 1632918735.49, Number of outliers = 6
- Total Revenue: Average = 20772269312.42, Number of outliers = 6
- Total Assets: Average = 26595844854.93, Number of outliers = 6
- Total Liabilities: Average = 17137870717.27, Number of outliers = 6

Ticker symbol: XOM
Outlier categories: ['Accounts Payable', 'Accounts Receivable',
'Earnings Before Tax', 'Net Income', 'Total Revenue', 'Total Assets',
'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 7
- Accounts Receivable: Average = -74812071.52, Number of outliers = 7
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 7
- Net Income: Average = 1632918735.49, Number of outliers = 7
- Total Revenue: Average = 20772269312.42, Number of outliers = 7
- Total Assets: Average = 26595844854.93, Number of outliers = 7
- Total Liabilities: Average = 17137870717.27, Number of outliers = 7

Ticker symbol: XOM
Outlier categories: ['Accounts Payable', 'Accounts Receivable',
'Earnings Before Tax', 'Net Income', 'Total Revenue', 'Total Assets',
'Total Liabilities']
Average amount for each category:
- Accounts Payable: Average = 3680650927.13, Number of outliers = 7
- Accounts Receivable: Average = -74812071.52, Number of outliers = 7
- Earnings Before Tax: Average = 2277350867.75, Number of outliers = 7
```

```
- Net Income: Average = 1632918735.49, Number of outliers = 7
- Total Revenue: Average = 20772269312.42, Number of outliers = 7
- Total Assets: Average = 26595844854.93, Number of outliers = 7
- Total Liabilities: Average = 17137870717.27, Number of outliers = 7

Ticker symbol: YHOO
Outlier categories: ['Cash Ratio', 'Operating Margin']
Average amount for each category:
- Cash Ratio: Average = 74.46, Number of outliers = 2
- Operating Margin: Average = 18.61, Number of outliers = 2
```

sec

{"summary":"{\n \"name\": \"sec\",\n \"rows\": 505,\n \"fields\": [\n {\n \"column\": \"Ticker symbol\",\n \"properties\": {\n \"dtype\": \"string\",\n \"num_unique_values\": 505,\n \"samples\": [\n \"ES\",\n \"LVLT\",\n \"WFM\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Security\",\n \"properties\": {\n \"dtype\": \"string\",\n \"num_unique_values\": 504,\n \"samples\": [\n \"Eversource Energy\",\n \"Level 3 Communications\",\n \"Whole Foods Market\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SEC filings\",\n \"properties\": {\n \"dtype\": \"category\",\n \"num_unique_values\": 1,\n \"samples\": [\n \"reports\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"GICS Sector\",\n \"properties\": {\n \"dtype\": \"category\",\n \"num_unique_values\": 11,\n \"samples\": [\n \"Financials\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"GICS Sub Industry\",\n \"properties\": {\n \"dtype\": \"category\",\n \"num_unique_values\": 124,\n \"samples\": [\n \"Data Processing & Outsourced Services\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Address of Headquarters\",\n \"properties\": {\n \"dtype\": \"string\",\n \"num_unique_values\": 264,\n \"samples\": [\n \"Dublin, Ohio\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Date first added\",\n \"properties\": {\n \"dtype\": \"date\",\n \"min\": \"1964-03-31 00:00:00\",\n \"max\": \"2017-01-05 00:00:00\",\n \"num_unique_values\": 252,\n \"samples\": [\n \"2007-09-07 00:00:00\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"CIK\",\n \"properties\": {\n

\"dtype\": \"number\",\n   \"std\": 514437,\n   \"min\": 1800,\n   \"max\": 1659166,\n   \"num_unique_values\": 500,\n   \"samples\": [\n   79879\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"sec"}

```
prc.columns
```

```
Index(['date', 'symbol', 'open', 'close', 'low', 'high', 'volume'],
dtype='object')
```

```
merged_data = pd.merge(sec, prc, left_on='Ticker symbol',
right_on='symbol')

grouped_data = merged_data.groupby('GICS Sector')

data = merged_data

data.head(3)
```

{"type":"dataframe","variable_name":"data"}

```
# Calculate the maximum closing price for each GICS Sub Industry
subindustry_max_close = merged_data.groupby('GICS Sub Industry')
['close'].max()

# Plotting the bar diagram
plt.figure(figsize=(17,19))
subindustry_max_close.plot(kind='bar', color='skyblue')
plt.xlabel('GICS Sub Industry')
plt.ylabel('Maximum Close Price')
plt.title('Maximum Close Price for Each GICS Sub Industry')
plt.xticks(rotation=90, ha='center')  # Rotate x-axis labels for
better readability
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

Sub-industries with high maximum closing prices: Based on the graph, it appears that Gold, Copper, and Finance are among the sub-industries with the highest maximum closing prices. This could be due to a number of factors, such as investor demand for these assets during times of economic uncertainty, or strong financial performance by companies in these sectors.

Sub-industries with zero maximum closing price: Some sub-industries, such as Multi-Sector Holdings and Real Estate Services, have a maximum closing price of zero. This could indicate that they are either new entrants to the stock market or haven't had any significant trading

activity yet. It's also possible that these sub-industries are not currently represented by any publicly traded companies.

Other sub-industries with zero maximum closing price: Other sub-industries with a maximum closing price of zero include Diversified Fixed Income and Timber & Forest Products. It's possible that these sub-industries are not currently represented by any publicly traded companies, or that the companies within these sub-industries haven't had any major price spikes during the time period covered by the graph.

Sub-industries potentially impacted by economic factors: Sub-industries like Casinos & Gaming and Hotels, Resorts & Cruise Lines might have been impacted by economic downturns or travel restrictions, leading to lower share prices and potentially causing their maximum closing prices to be lower than expected. The travel and hospitality industries are often sensitive to economic conditions, and companies in these sectors may see their stock prices decline during recessions or periods of high uncertainty.

```
sns.set_style("whitegrid")

plt.figure(figsize=(12, 8))
sns.violinplot(data=merged_data, x='GICS Sector', y='close',
inner=None)
sns.pointplot(data=merged_data.groupby('GICS Sector')
['close'].max().reset_index(),
              x='GICS Sector', y='close', color='black', markers='o')
for index, row in merged_data.loc[merged_data.groupby('GICS Sector')
['close'].idxmax()].iterrows():
    plt.text(row['GICS Sector'], row['close'], row['Ticker symbol'],
color='red', ha="left", va="top")
plt.xlabel('GICS Sector')
plt.ylabel('Close Price')
plt.title('Securities with highest closing in their sector')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
sum_volume = merged_data.groupby('GICS Sector')
['volume'].sum().reset_index()
sns.set_style("whitegrid")
plt.figure(figsize=(12, 8))
sns.barplot(data=sum_volume, x='GICS Sector', y='volume',
palette='viridis')
plt.xlabel('GICS Sector')
plt.ylabel('Total Volume')
plt.title('Securities by their volumne')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
data.columns
```

```
Index(['Ticker symbol', 'Security', 'SEC filings', 'GICS Sector',
       'GICS Sub Industry', 'Address of Headquarters', 'Date first
added',
       'CIK', 'date', 'symbol', 'open', 'close', 'low', 'high',
'volume'],
      dtype='object')
```

```
prc.columns
```

```
Index(['date', 'symbol', 'open', 'close', 'low', 'high', 'volume'],
dtype='object')
```

```
import pandas as pd
import torch
import torch.nn as nn
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split


merged_data = pd.merge(data[['date', 'symbol', 'GICS Sector']],
                       prc[['date', 'symbol', 'open', 'close', 'low',
'high', 'volume']],
```

```
                        on=['date', 'symbol'])


X = merged_data[['open', 'close', 'low', 'high', 'volume']]
y = merged_data['close']

scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1))


X_tensor = torch.tensor(X_scaled, dtype=torch.float32)
y_tensor = torch.tensor(y_scaled, dtype=torch.float32)


X_train, X_test, y_train, y_test = train_test_split(X_tensor,
y_tensor, test_size=0.2, random_state=42)


class Model(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(Model, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out


input_size = X_train.shape[1]
hidden_size = 64
output_size = 1
model = Model(input_size, hidden_size, output_size)


criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)


num_epochs = 100
for epoch in range(num_epochs):
    optimizer.zero_grad()
    outputs = model(X_train)
    loss = criterion(outputs, y_train)
    loss.backward()
    optimizer.step()
```

```
    if (epoch+1) % 10 == 0:
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss:
{loss.item():.4f}')

with torch.no_grad():
    predicted = model(X_test)
    mse = criterion(predicted, y_test)
print(f'Mean Squared Error: {mse:.4f}')
```

```
Epoch [10/100], Loss: 0.0027
Epoch [20/100], Loss: 0.0022
Epoch [30/100], Loss: 0.0013
Epoch [40/100], Loss: 0.0008
Epoch [50/100], Loss: 0.0005
Epoch [60/100], Loss: 0.0002
Epoch [70/100], Loss: 0.0001
Epoch [80/100], Loss: 0.0000
Epoch [90/100], Loss: 0.0000
Epoch [100/100], Loss: 0.0000
Mean Squared Error: 0.0000
```

```
# Inference
with torch.no_grad():
    predicted = model(X_test)
    predicted_unscaled = scaler_y.inverse_transform(predicted.numpy())
# Convert back to original scale
    actual_unscaled =
scaler_y.inverse_transform(y_test.numpy().reshape(-1, 1))  # Convert
back to original scale
    mse = criterion(predicted, y_test)
    print(f'Mean Squared Error: {mse:.4f}')

# Convert Torch tensors to numpy arrays
predicted_np = predicted_unscaled.flatten()
actual_np = actual_unscaled.flatten()

# Create a DataFrame to compare predicted and actual closing prices
comparison_df = pd.DataFrame({'Actual Close': actual_np, 'Predicted
Close': predicted_np})
print(comparison_df)
```

```
Mean Squared Error: 0.0000
        Actual Close  Predicted Close
0          41.590000        44.761215
1          15.270000        24.551680
2          22.629999        25.004986
3          69.690002        70.264297
4          51.259998        54.262478
...              ...              ...
170248     22.670000        29.137793
```

```
170249     32.790005        29.026440
170250     61.740002        58.931358
170251     51.039997        53.374519
170252     72.579994        72.440399

[170253 rows x 2 columns]
```

```
y_test_unscaled = actual_unscaled # Plot actual vs predicted values
plt.figure(figsize=(15, 8))
plt.scatter(y_test_unscaled, predicted_unscaled, color='Green',
label='Actual vs Predicted')
plt.plot([y_test_unscaled.min(), y_test_unscaled.max()],
[y_test_unscaled.min(), y_test_unscaled.max()], 'k--', lw=2,
label='Absolute Prediction')
plt.xlabel('Actual Close Price')
plt.ylabel('Predicted Close Price')
plt.title('Actual vs Predicted Close Price')
plt.legend()
plt.show()
```

```
pip install torchviz
```

```
Requirement already satisfied: torchviz in
/usr/local/lib/python3.10/dist-packages (0.0.2)
Requirement already satisfied: torch in
/usr/local/lib/python3.10/dist-packages (from torchviz) (2.2.1+cu121)
Requirement already satisfied: graphviz in
/usr/local/lib/python3.10/dist-packages (from torchviz) (0.20.3)
```

```
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(3.14.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(4.11.0)
Requirement already satisfied: sympy in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz) (1.12)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz) (3.3)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz) (3.1.4)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(2023.6.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105
in /usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(12.1.105)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(12.1.105)
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(8.9.2.26)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(10.3.2.106)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(11.4.5.107)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(12.1.0.106)
Requirement already satisfied: nvidia-nccl-cu12==2.19.3 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(2.19.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch->torchviz)
(12.1.105)
Requirement already satisfied: triton==2.2.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from torch->torchviz) (2.2.0)
Requirement already satisfied: nvidia-nvjitlink-cu12 in
/usr/local/lib/python3.10/dist-packages (from nvidia-cusolver-
cu12==11.4.5.107->torch->torchviz) (12.4.127)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch->torchviz)
(2.1.5)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch->torchviz)
(1.3.0)
```

```
pip install torchsummary
```

```
Requirement already satisfied: torchsummary in
/usr/local/lib/python3.10/dist-packages (1.5.1)
```

```
import torch
import torch.nn as nn
from torchsummary import summary

class ComplexModel(nn.Module):
    def __init__(self, input_size, hidden_sizes, output_size):
        super(ComplexModel, self).__init__()
        self.input_layer = nn.Linear(input_size, hidden_sizes[0])
        self.hidden_layers = nn.ModuleList([
            nn.Linear(hidden_sizes[i], hidden_sizes[i+1]) for i in
range(len(hidden_sizes) - 1)
        ])
        self.output_layer = nn.Linear(hidden_sizes[-1], output_size)

    def forward(self, x):
        x = torch.relu(self.input_layer(x))
        for hidden_layer in self.hidden_layers:
            x = torch.relu(hidden_layer(x))
        x = self.output_layer(x)
        return x

# Define model parameters
input_size = X_train.shape[1]
hidden_sizes = [128, 64, 32]
output_size = 1

# Initialize model
complex_model = ComplexModel(input_size, hidden_sizes, output_size)

# Display model summary
summary(complex_model, input_size=(input_size,))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
```

```
            Linear-1                   [-1, 128]             768
            Linear-2                    [-1, 64]           8,256
            Linear-3                    [-1, 32]           2,080
            Linear-4                     [-1, 1]              33
================================================================
Total params: 11,137
Trainable params: 11,137
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.04
Estimated Total Size (MB): 0.04
----------------------------------------------------------------
```

```python
import torch
import torch.nn as nn
import torch.optim as optim
import torch.utils.data as data
import torchvision.models as models
import torchvision.datasets as dset
import torchvision.transforms as transforms
from torch.autograd import Variable
from torchviz import make_dot

batch_size = 3
learning_rate = 0.0002
epoch = 50

# Load the pre-trained ResNet-50 model
resnet = models.resnet50(pretrained=True)

# Create a dummy input tensor
dummy_input = torch.randn(1, 3, 224, 224)  # Assuming input size is
224x224

# Visualize the model
make_dot(resnet(dummy_input), params=dict(resnet.named_parameters()))
```

```python
import pandas as pd

# Inference
with torch.no_grad():
    predicted = model(X_test)
    predicted_unscaled = scaler_y.inverse_transform(predicted.numpy())
# Convert back to original scale
    actual_unscaled =
scaler_y.inverse_transform(y_test.numpy().reshape(-1, 1))  # Convert
back to original scale
    mse = criterion(predicted, y_test)
    print(f'Mean Squared Error: {mse:.4f}')

# Calculate the difference between actual and predicted closing prices
difference = actual_unscaled.flatten() - predicted_unscaled.flatten()

# Create a DataFrame to compare actual and predicted closing prices
along with the difference
comparison_df = pd.DataFrame({'Actual Close':
actual_unscaled.flatten(), 'Predicted Close':
predicted_unscaled.flatten(), 'Difference': difference})
print(comparison_df)
```

```
Mean Squared Error: 0.0000
        Actual Close  Predicted Close  Difference
0          41.590000        44.761215   -3.171215
1          15.270000        24.551680   -9.281680
2          22.629999        25.004986   -2.374987
3          69.690002        70.264297   -0.574295
4          51.259998        54.262478   -3.002480
...              ...              ...         ...
170248     22.670000        29.137793   -6.467793
170249     32.790005        29.026440    3.763565
170250     61.740002        58.931358    2.808643
170251     51.039997        53.374519   -2.334522
170252     72.579994        72.440399    0.139595

[170253 rows x 3 columns]
```

```python
# Find the ticker symbol with the minimum absolute difference
min_diff_row = comparison_df.loc[comparison_df['Difference'].idxmin()]
print("Ticker Symbol with the Least Difference:")
print(min_diff_row)
```

```
Ticker Symbol with the Least Difference:
Actual Close       1536.780029
Predicted Close    1637.802124
Difference         -101.022095
Name: 67135, dtype: float32
```