# Install necessary libraries

!pip install -q yfinance

!pip install pandas-datareader

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from pandas\_datareader.data import DataReader

import yfinance as yf

from pandas\_datareader import data as pdr

from datetime import datetime

yf.pdr\_override()

# Set plotting styles

sns.set\_style('whitegrid')

plt.style.use("fivethirtyeight")

%matplotlib inline

# # Download stock data for META, TSLA, NVDA, and NFLX

# META = yf.download('META', start='2023-01-01', end='2025-01-01')

# AAPL = yf.download('AAPL', start='2023-01-01', end='2025-01-01')

# NVDA = yf.download('NVDA', start='2023-01-01', end='2025-01-01')

# NFLX = yf.download('NFLX', start='2023-01-01', end='2025-01-01')

import pandas as pd

from datetime import datetime

#tech\_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']

tech\_list = ['META', 'AAPL', 'NVDA', 'NFLX']

end = datetime.now()

start = datetime(end.year - 1, end.month, end.day)

# Download stock data

META = yf.download('META', start=start, end=end)

AAPL = yf.download('AAPL', start=start, end=end)

NVDA = yf.download('NVDA', start=start, end=end)

NFLX = yf.download('NFLX', start=start, end=end)

company\_list = [META, AAPL, NVDA, NFLX]

company\_name = ["META", "APPLE", "NVIDIA", "NETFLIX"]

# for stock in tech\_list:

#     globals()[stock] = yf.download(stock, start=start, end=end)

# Fill missing values using forward fill

for company in company\_list:

    company.ffill(inplace=True)  # Forward fill to handle missing data

# Ensure consistent date format

for company in company\_list:

    company.reset\_index(inplace=True)  # Reset index to ensure consistent date format

    company['Date'] = pd.to\_datetime(company['Date'])  # Ensure 'Date' column is in datetime format

    company.set\_index('Date', inplace=True)  # Set 'Date' as index again

# Add company name column to each dataframe

for company, com\_name in zip(company\_list, company\_name):

    company["company\_name"] = com\_name

# Concatenate individual stock data into a single DataFrame

df = pd.concat(company\_list, axis=0)

# Shuffle the data and get a random sample of the last 10 rows

df = df.sample(frac=1).reset\_index(drop=True)  # Shuffle the data

print(df.tail(10))  # Display the last 10 rows of the shuffled dataframe

df = df.reset\_index()

df = df.fillna(method='ffill')

import matplotlib.pyplot as plt

import seaborn as sns

sns.set\_style('whitegrid')

plt.style.use("fivethirtyeight")

plt.figure(figsize=(15, 10))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company\_list, 1):

    plt.subplot(2, 2, i)

    company['Adj Close'].plot()

    plt.ylabel('Adj Close')

    plt.title(f"Closing Price of {tech\_list[i - 1]}")

plt.tight\_layout()

plt.figure(figsize=(15, 10))

plt.subplots\_adjust(top=1.25, bottom=1.2)

for i, company in enumerate(company\_list, 1):

    plt.subplot(2, 2, i)

    company['Volume'].plot()

    plt.ylabel('Volume')

    plt.title(f"Sales Volume for {tech\_list[i - 1]}")

plt.tight\_layout()

ma\_day = [10, 20, 50]

for ma in ma\_day:

    for company in company\_list:

        column\_name = f"MA for {ma} days"

        company[column\_name] = company['Adj Close'].rolling(ma).mean()

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

META[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])

axes[0,0].set\_title('META')

AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,1])

axes[0,1].set\_title('APPLE')

NVDA[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])

axes[1,0].set\_title('NVIDIA')

NFLX[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,1])

axes[1,1].set\_title('NETFLIX')

fig.tight\_layout()

for company in company\_list:

    company['Daily Return'] = company['Adj Close'].pct\_change()

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

META['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')

axes[0,0].set\_title('META')

AAPL['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')

axes[0,1].set\_title('APPLE')

NVDA['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')

axes[1,0].set\_title('NVIDIA')

NFLX['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')

axes[1,1].set\_title('NETFLIX')

fig.tight\_layout()

plt.figure(figsize=(12, 9))

for i, company in enumerate(company\_list, 1):

    plt.subplot(2, 2, i)

    company['Daily Return'].hist(bins=50)

    plt.xlabel('Daily Return')

    plt.title(f'{company\_name[i - 1]}')

plt.tight\_layout()

import yfinance as yf

import pandas as pd

import matplotlib.pyplot as plt

# Define company tickers

tech\_list = ['META', 'AAPL', 'NVDA', 'NFLX']

# Download stock data for the past year

end = pd.to\_datetime('2024-05-27')

start = end - pd.DateOffset(years=1)

company\_list = []

for stock in tech\_list:

    company\_list.append(yf.download(stock, start=start, end=end))

# Ensure consistent date format

for company in company\_list:

    company.reset\_index(inplace=True)  # Reset index to ensure consistent date format

    company['Date'] = pd.to\_datetime(company['Date'])  # Ensure 'Date' column is in datetime format

    company.set\_index('Date', inplace=True)  # Set 'Date' as index again

# Calculate daily percentage change

for company in company\_list:

    company['Daily Change'] = company['Adj Close'].pct\_change() \* 100

# Create histogram (one plot for all companies)

plt.figure(figsize=(12, 6))

for i, company in enumerate(company\_list):

    # Overlay histograms on the same plot with different transparency levels

    plt.hist(company['Daily Change'], bins=20, alpha=0.7, label=tech\_list[i])

plt.xlabel('Daily Change (%)')

plt.ylabel('Frequency')

plt.title('Distribution of Daily Change for Tech Stocks (Past Year)')

plt.legend()  # Add legend to identify each company's distribution

plt.tight\_layout()

# ... (Rest of the code for identifying best/worst performers and displaying additional info)

plt.show()

closing\_df = pdr.get\_data\_yahoo(tech\_list, start=start, end=end)['Adj Close']

tech\_rets = closing\_df.pct\_change()

sns.pairplot(tech\_rets, kind='reg')

plt.figure(figsize=(12, 10))

plt.subplot(2, 2, 1)

sns.heatmap(tech\_rets.corr(), annot=True, cmap='summer')

plt.title('Correlation of stock return')

plt.subplot(2, 2, 2)

sns.heatmap(closing\_df.corr(), annot=True, cmap='summer')

plt.title('Correlation of stock closing price')

from sklearn.model\_selection import train\_test\_split

# Drop missing values from both tech\_rets and closing\_df

tech\_rets\_cleaned = tech\_rets.dropna()

closing\_df\_cleaned = closing\_df.dropna()

# Align the number of rows in tech\_rets\_cleaned and closing\_df\_cleaned

min\_rows = min(tech\_rets\_cleaned.shape[0], closing\_df\_cleaned.shape[0])

tech\_rets\_cleaned = tech\_rets\_cleaned[:min\_rows]

closing\_df\_cleaned = closing\_df\_cleaned[:min\_rows]

# Create X and y after handling missing values

X = tech\_rets\_cleaned.values

y = closing\_df\_cleaned.values

# Print the shapes of X and y for verification

print("Shape of X:", X.shape)

print("Shape of y:", y.shape)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

import numpy as np

import matplotlib.pyplot as plt

# Define colors for the histograms

train\_colors = ['blue', 'green', 'orange', 'purple']  # Adjust as needed

test\_colors = ['red', 'yellow', 'cyan', 'magenta']    # Adjust as needed

# Plot the distribution of y\_train

plt.figure(figsize=(12, 6))

for i, dataset in enumerate(y\_train.T):

    stock\_name = tech\_list[i]  # Assuming tech\_list contains the names of the stocks

    plt.hist(dataset, bins=30, color=train\_colors[i], alpha=0.7, label=stock\_name)

plt.title('Distribution of y\_train')

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.legend()

plt.show()

# Plot the distribution of y\_test

plt.figure(figsize=(12, 6))

for i, dataset in enumerate(y\_test.T):

    stock\_name = tech\_list[i]  # Assuming tech\_list contains the names of the stocks

    plt.hist(dataset, bins=30, color=test\_colors[i], alpha=0.7, label=stock\_name)

plt.title('Distribution of y\_test')

plt.xlabel('Values')

plt.ylabel('Frequency')

plt.legend()

plt.show()