```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
import seaborn as sns
import os
from datetime import datetime

import warnings
warnings.filterwarnings("ignore")
```

```
data = pd.read_csv('/content/all_stocks_5yr.csv')
data.head()
```

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

```
data.shape
```

```
(619040, 7)
```

```
data.describe()
```

	open	high	low	close	volume
count	619029.000000	619032.000000	619032.000000	619040.000000	6.190400e+05
mean	83.023334	83.778311	82.256096	83.043763	4.321823e+06
std	97.378769	98.207519	96.507421	97.389748	8.693610e+06
min	1.620000	1.690000	1.500000	1.590000	0.000000e+00
25%	40.220000	40.620000	39.830000	40.245000	1.070320e+06
50%	62.590000	63.150000	62.020000	62.620000	2.082094e+06
75%	94.370000	95.180000	93.540000	94.410000	4.284509e+06
max	2044.000000	2067.990000	2035.110000	2049.000000	6.182376e+08

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619040 entries, 0 to 619039
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    619040 non-null  object
 1   open    619029 non-null  float64
 2   high    619032 non-null  float64
 3   low     619032 non-null  float64
 4   close   619040 non-null  float64
 5   volume  619040 non-null  int64
 6   Name    619040 non-null  object
dtypes: float64(4), int64(1), object(2)
memory usage: 33.1+ MB
```

```
data['date'] = pd.to_datetime(data['date'])
data.info()
```

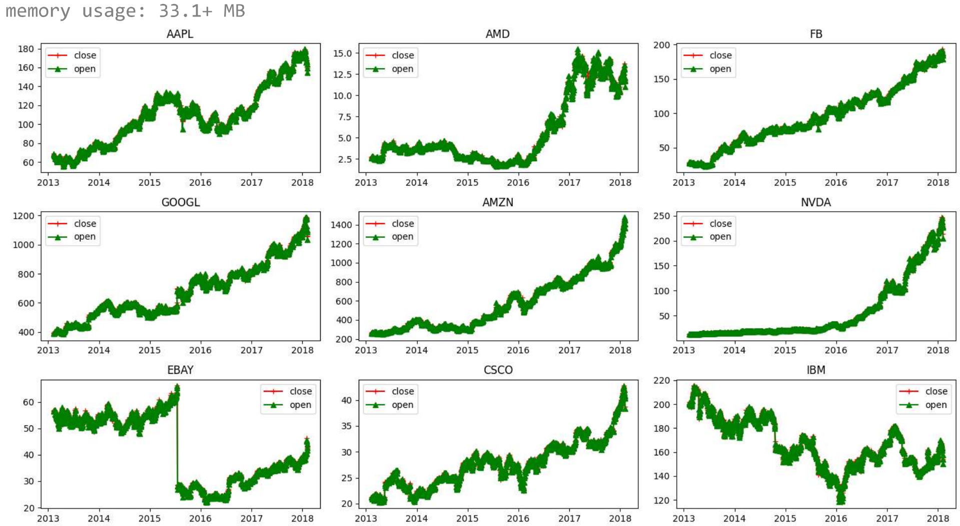
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619040 entries, 0 to 619039
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    619040 non-null  datetime64[ns]
 1   open    619029 non-null  float64
 2   high    619032 non-null  float64
 3   low     619032 non-null  float64
 4   close   619040 non-null  float64
 5   volume  619040 non-null  int64
 6   Name    619040 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 33.1+ MB
```

```
data['Name'].unique()
```

```
array(['AAL', 'AAPL', 'AAP', 'ABBV', 'ABC', 'ABT', 'ACN', 'ADBE', 'ADI',
       'ADM', 'ADP', 'ADSK', 'ADS', 'AEE', 'AEP', 'AES', 'AET', 'AFL',
       'AGN', 'AIG', 'AIV', 'AIZ', 'AJG', 'AKAM', 'ALB', 'ALGN', 'ALK',
       'ALLE', 'ALL', 'ALXN', 'AMAT', 'AMD', 'AME', 'AMGN', 'AMG', 'AMP',
       'AMT', 'AMZN', 'ANDV', 'ANSS', 'ANTM', 'AON', 'AOS', 'APA', 'APC',
       'APD', 'APH', 'APTV', 'ARE', 'ARNC', 'ATVI', 'AVB', 'AVGO', 'AVY',
       'AWK', 'AXP', 'AYI', 'AZO', 'A', 'BAC', 'BAX', 'BA', 'BBT', 'BBY',
       'BDX', 'BEN', 'BF.B', 'BHF', 'BHGE', 'BIIB', 'BK', 'BLK', 'BLL',
       'BMY', 'BRK.B', 'BSX', 'BWA', 'BXP', 'CAG', 'CAH', 'CAT', 'CA',
       'CBG', 'CBOE', 'CBS', 'CB', 'CCI', 'CCL', 'CDNS', 'CELG', 'CERN',
       'CFG', 'CF', 'CHD', 'CHK', 'CHRW', 'CHTR', 'CINF', 'CI', 'CLX',
       'CL', 'CMA', 'CMCSA', 'CME', 'CMG', 'CMI', 'CMS', 'CNC', 'CNP',
       'COF', 'COG', 'COL', 'COO', 'COP', 'COST', 'COTY', 'CPB', 'CRM',
       'CSCO', 'CSRA', 'CSX', 'CTAS', 'CTL', 'CTSH', 'CTXS', 'CVS', 'CVX',
       'CXO', 'C', 'DAL', 'DE', 'DFS', 'DGX', 'DG', 'DHI', 'DHR', 'DISCA',
       'DISCK', 'DISH', 'DIS', 'DLR', 'DLTR', 'DOV', 'DPS', 'DRE', 'DRI',
       'DTE', 'DUK', 'DVA', 'DVN', 'DWDP', 'DXC', 'D', 'EA', 'EBAY',
       'ECL', 'ED', 'EFX', 'EIX', 'EL', 'EMN', 'EMR', 'EOG', 'EQIX',
       'EQR', 'EQT', 'ESRX', 'ESS', 'ES', 'ETFC', 'ETN', 'ETR', 'EVHC',
       'EW', 'EXC', 'EXPD', 'EXPE', 'EXR', 'FAST', 'FBHS', 'FB', 'FCX',
       'FDX', 'FE', 'FFIV', 'FISV', 'FIS', 'FITB', 'FLIR', 'FLR', 'FLS',
       'FL', 'FMC', 'FOXA', 'FOX', 'FRT', 'FTI', 'FTV', 'F', 'GD', 'GE',
       'GGP', 'GILD', 'GIS', 'GLW', 'GM', 'GOOGL', 'GOOG', 'GPC', 'GPN',
       'GPS', 'GRMN', 'GS', 'GT', 'GWW', 'HAL', 'HAS', 'HBAN', 'HBI',
       'HCA', 'HCN', 'HCP', 'HD', 'HES', 'HIG', 'HII', 'HLT', 'HOG',
       'HOLX', 'HON', 'HPE', 'HPQ', 'HP', 'HRB', 'HRL', 'HRS', 'HSIC',
       'HST', 'HSY', 'HUM', 'IBM', 'ICE', 'IDXX', 'IFF', 'ILMN', 'INCY',
       'INFO', 'INTC', 'INTU', 'IPG', 'IP', 'IQV', 'IRM', 'IR', 'ISRG',
       'ITW', 'IT', 'IVZ', 'JBHT', 'JCI', 'JEC', 'JNJ', 'JNPR', 'JPM',
       'JWN', 'KEY', 'KHC', 'KIM', 'KLAC', 'KMB', 'KMI', 'KMX', 'KORS',
       'KO', 'KR', 'KSS', 'KSU', 'K', 'LB', 'LEG', 'LEN', 'LH', 'LKQ',
       'LLL', 'LLY', 'LMT', 'LNC', 'LNT', 'LOW', 'LRCX', 'LUK', 'LUV',
       'LYB', 'L', 'MAA', 'MAC', 'MAR', 'MAS', 'MAT', 'MA', 'MCD', 'MCHP',
       'MCK', 'MCO', 'MDLZ', 'MDT', 'MET', 'MGM', 'MHK', 'MKC', 'MLM',
       'MMC', 'MMM', 'MNST', 'MON', 'MOS', 'MO', 'MPC', 'MRK', 'MRO',
       'MSFT', 'MSI', 'MS', 'MTB', 'MTD', 'MU', 'MYL', 'M', 'NAVI', 'NBL',
       'NCLH', 'NDAQ', 'NEE', 'NEM', 'NFLX', 'NFX', 'NI', 'NKE', 'NLSN',
       'NOC', 'NOV', 'NRG', 'NSC', 'NTAP', 'NTRS', 'NUE', 'NVDA', 'NWL',
       'NWSA', 'NWS', 'OKE', 'OMC', 'ORCL', 'ORLY', 'OXY', 'O', 'PAYX',
       'PBCT', 'PCAR', 'PCG', 'PCLN', 'PDCO', 'PEG', 'PEP', 'PFE', 'PFG',
       'PGR', 'PG', 'PHM', 'PH', 'PKG', 'PKI', 'PLD', 'PM', 'PNC', 'PNR',
       'PNW', 'PPG', 'PPL', 'PRGO', 'PRU', 'PSA', 'PSX', 'PVH', 'PWR',
       'PXD', 'PX', 'PYPL', 'QCOM', 'QRVO', 'RCL', 'REGN', 'REG', 'RE',
       'RF', 'RHI', 'RHT', 'RJF', 'RL', 'RMD', 'ROK', 'ROP', 'ROST',
       'RRC', 'RSG', 'RTN', 'SBAC', 'SBUX', 'SCG', 'SCHW', 'SEE', 'SHW',
       'SIG', 'SJM', 'SLB', 'SLG', 'SNA', 'SNI', 'SNPS', 'SO', 'SPGI',
       'SPG', 'SRCL', 'SRE', 'STI', 'STT', 'STX', 'STZ', 'SWKS', 'SWK',
       'SYF', 'SYK', 'SYMC', 'SYY', 'TAP', 'TDG', 'TEL', 'TGT', 'TIF',
       'TJX', 'TMK', 'TMO', 'TPR', 'TRIP', 'TROW', 'TRV', 'TSCO', 'TSN',
       'TSS', 'TWX', 'TXN', 'TXT', 'T', 'UAA', 'UAL', 'UA', 'UDR', 'UHS',
       'ULTA', 'UNH', 'UNM', 'UNP', 'UPS', 'URI', 'USB', 'UTX', 'VAR',
       'VFC', 'VIAB', 'VLO', 'VMC', 'VNO', 'VRSK', 'VRSN', 'VRTX', 'VTR',
       'VZ', 'V', 'WAT', 'WBA', 'WDC', 'WEC', 'WFC', 'WHR', 'WLTW', 'WMB',
       'WMT', 'WM', 'WRK', 'WU', 'WYNN', 'WYN', 'WY', 'XEC', 'XEL',
       'XLNX', 'XL', 'XOM', 'XRAY', 'XRX', 'XYL', 'YUM', 'ZBH', 'ZION',
       'ZTS'], dtype=object)
```

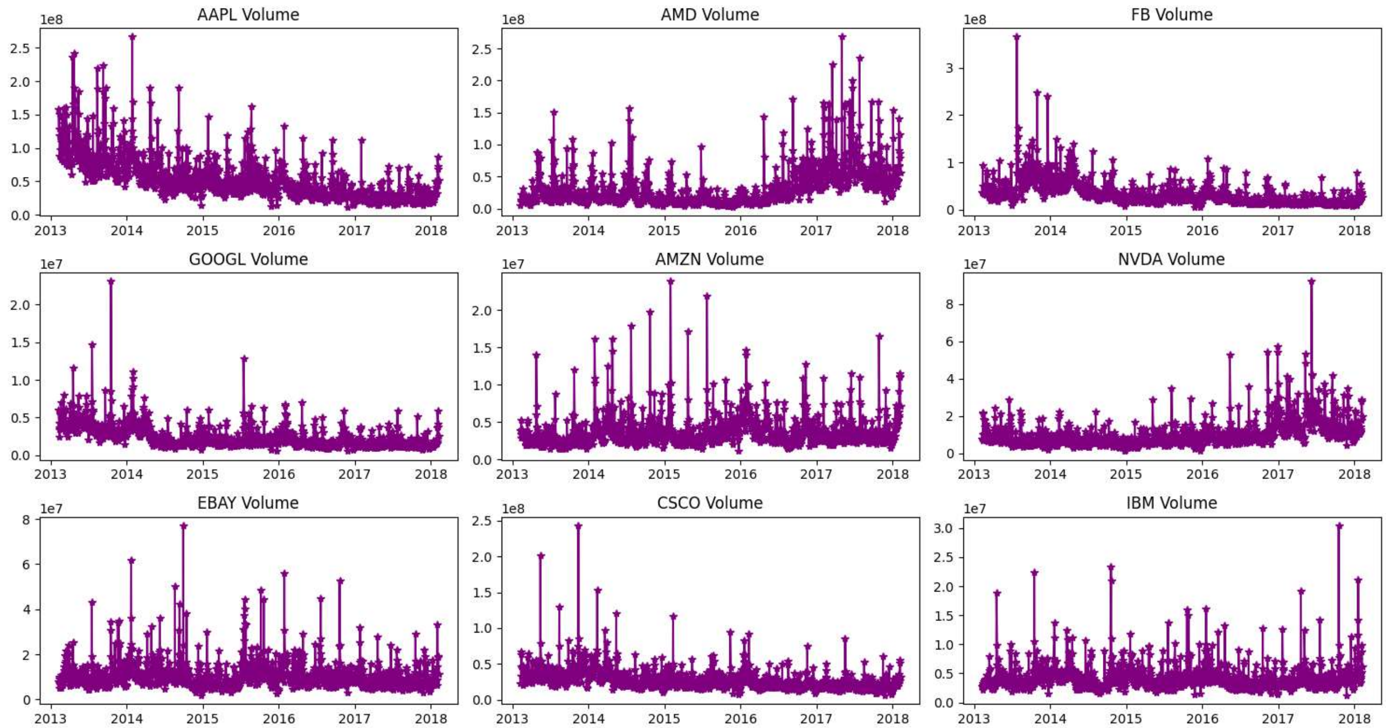
```
# Convert the 'date' column to DateTime data type
data['date'] = pd.to_datetime(data['date'])
data.info()

# List of companies you want to visualize
companies = ['AAPL', 'AMD', 'FB', 'GOOGL', 'AMZN', 'NVDA', 'EBAY', 'CSCO', 'IBM']

# Visualize open and close stock prices for the selected companies
plt.figure(figsize=(15, 8))
for index, company in enumerate(companies, 1):
    plt.subplot(3, 3, index)
    c = data[data['Name'] == company]
    plt.plot(c['date'], c['close'], c="r", label="close", marker="+")
    plt.plot(c['date'], c['open'], c="g", label="open", marker="^")
    plt.title(company)
    plt.legend()
    plt.tight_layout()

plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619040 entries, 0 to 619039
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    619040 non-null  datetime64[ns]
 1   open    619029 non-null  float64
 2   high    619032 non-null  float64
 3   low     619032 non-null  float64
 4   close   619040 non-null  float64
 5   volume  619040 non-null  int64
 6   Name    619040 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 33.1+ MB
```

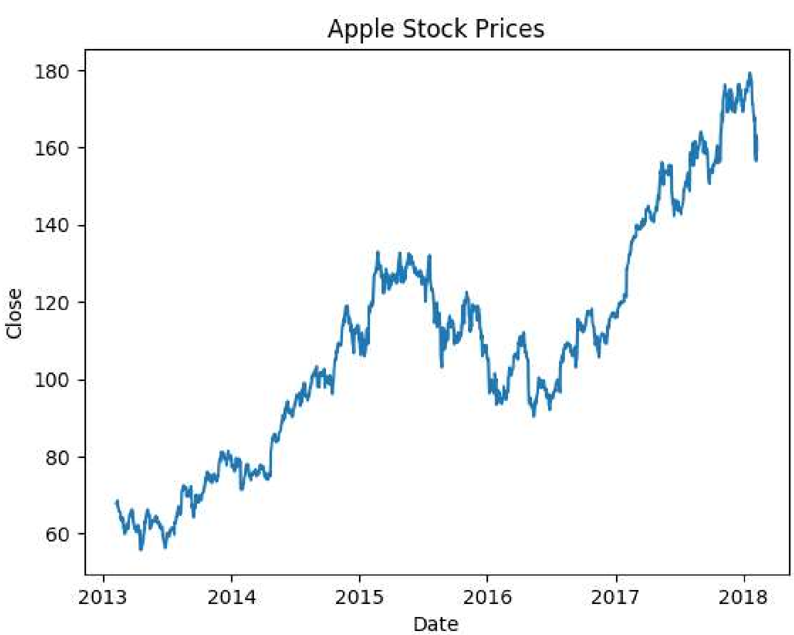



```
plt.figure(figsize=(15, 8))
for index, company in enumerate(companies, 1):
    plt.subplot(3, 3, index)
    c = data[data['Name'] == company]
    plt.plot(c['date'], c['volume'], c='purple', marker='*')
    plt.title(f"{company} Volume")
```

```
    plt.tight_layout()
```




```
apple = data[data['Name'] == 'AAPL']
prediction_range = apple.loc[(apple['date'] > datetime(2013,1,1))
& (apple['date']<datetime(2018,1,1))]
plt.plot(apple['date'],apple['close'])
plt.xlabel("Date")
plt.ylabel("Close")
plt.title("Apple Stock Prices")
plt.show()
```




```
close_data = apple.filter(['close'])
dataset = close_data.values
```

```
training = int(np.ceil(len(dataset) * .95))
print(training)
```

```
1197
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

train_data = scaled_data[0:int(training), :]
# prepare feature and labels
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

```
model = keras.models.Sequential()
model.add(keras.layers.LSTM(units=64,
                            return_sequences=True,
                            input_shape=(x_train.shape[1], 1)))
model.add(keras.layers.LSTM(units=64))
model.add(keras.layers.Dense(32))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(1))
model.summary
```

```
<bound method Model.summary of <keras.src.engine.sequential.Sequential object at 0x7f64b5da65f0>>
```

```
model.compile(optimizer='adam',
            loss='mean_squared_error')
history = model.fit(x_train,
                    y_train,
                    epochs=10)
```

```
Epoch 1/10
36/36 [==============================] - 9s 112ms/step - loss: 0.0313
Epoch 2/10
36/36 [==============================] - 4s 101ms/step - loss: 0.0101
Epoch 3/10
36/36 [==============================] - 2s 65ms/step - loss: 0.0101
Epoch 4/10
36/36 [==============================] - 2s 60ms/step - loss: 0.0093
Epoch 5/10
36/36 [==============================] - 2s 60ms/step - loss: 0.0079
Epoch 6/10
36/36 [==============================] - 3s 90ms/step - loss: 0.0072
Epoch 7/10
36/36 [==============================] - 2s 63ms/step - loss: 0.0070
Epoch 8/10
36/36 [==============================] - 2s 59ms/step - loss: 0.0071
Epoch 9/10
36/36 [==============================] - 2s 60ms/step - loss: 0.0063
Epoch 10/10
36/36 [==============================] - 2s 60ms/step - loss: 0.0063
```

```
test_data = scaled_data[training - 60:, :]
x_test = []
y_test = dataset[training:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])
```

```
x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

# predict the testing data
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# evaluation metrics
mse = np.mean(((predictions - y_test) ** 2))
print("MSE", mse)
print("RMSE", np.sqrt(mse))
```

```
2/2 [==============================] - 1s 27ms/step
MSE 42.1455311689615
RMSE 6.491958962359628
```

```
train = apple[:training]
test = apple[training:]
test['Predictions'] = predictions

plt.figure(figsize=(10, 8))
plt.plot(train['date'], train['close'])
plt.plot(test['date'], test[['close', 'Predictions']])
plt.title('Apple Stock Close Price')
plt.xlabel('Date')
plt.ylabel("Close")
plt.legend(['Train', 'Test', 'Predictions'])
```

```
<matplotlib.legend.Legend at 0x7f64b642e1d0>
```