```
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")
```

```
import pandas as pd

data = pd.read_csv('/content/all_stocks_5yr.csv', delimiter=',', on_bad_lines='skip')
print(data.shape)
print(data.sample(7))
```

```
(152910, 7)
              date    open    high       low   close   volume  Name
50493   2014-06-02  108.56  108.95  107.8100  108.35  1270180  ANTM
74656   2015-03-16   67.76   68.48   67.7500   68.43  2966222   BAX
149681  2016-09-22   27.08   27.72   26.9500   27.61  5094979   CTL
72524   2016-09-23   46.43   46.87   46.3100   46.61  1957032     A
27624   2017-10-23  100.04  100.04   99.1200   99.49   463664   AIZ
42411   2017-04-26  165.61  165.61  164.3397  164.61  4175536  AMGN
76393   2017-02-06  162.42  164.08  162.3800  163.98  3110494    BA
```

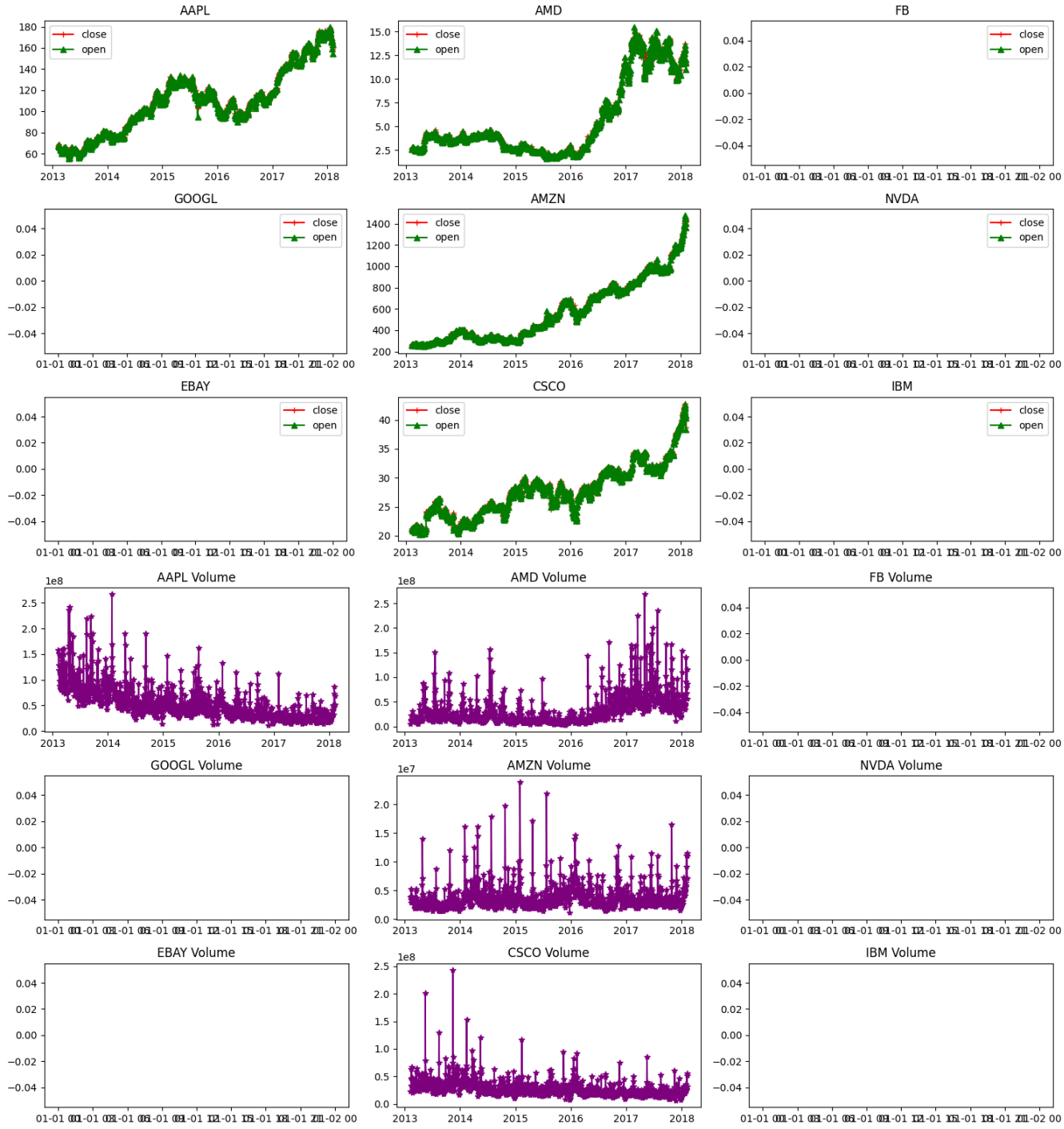
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 152910 entries, 0 to 152909
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    152910 non-null  object
 1   open    152909 non-null  float64
 2   high    152909 non-null  float64
 3   low     152909 non-null  float64
 4   close   152910 non-null  float64
 5   volume  152910 non-null  int64
 6   Name    152909 non-null  object
dtypes: float64(4), int64(1), object(2)
memory usage: 8.2+ MB
```

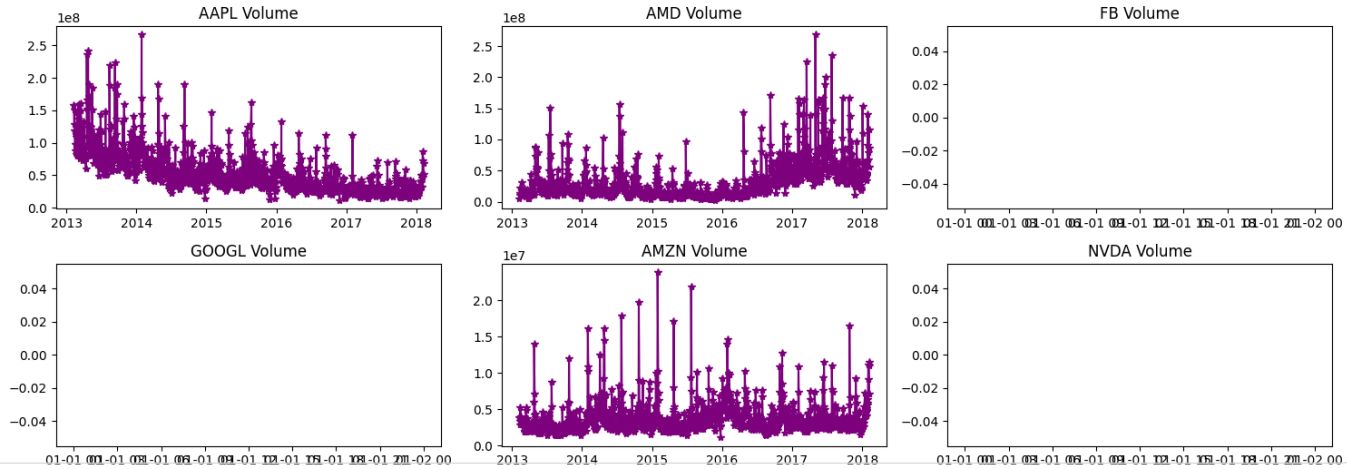
```
data['date'] = pd.to_datetime(data['date'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 152910 entries, 0 to 152909
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    152910 non-null  datetime64[ns]
 1   open    152909 non-null  float64
 2   high    152909 non-null  float64
 3   low     152909 non-null  float64
 4   close   152910 non-null  float64
 5   volume  152910 non-null  int64
 6   Name    152909 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 8.2+ MB
```

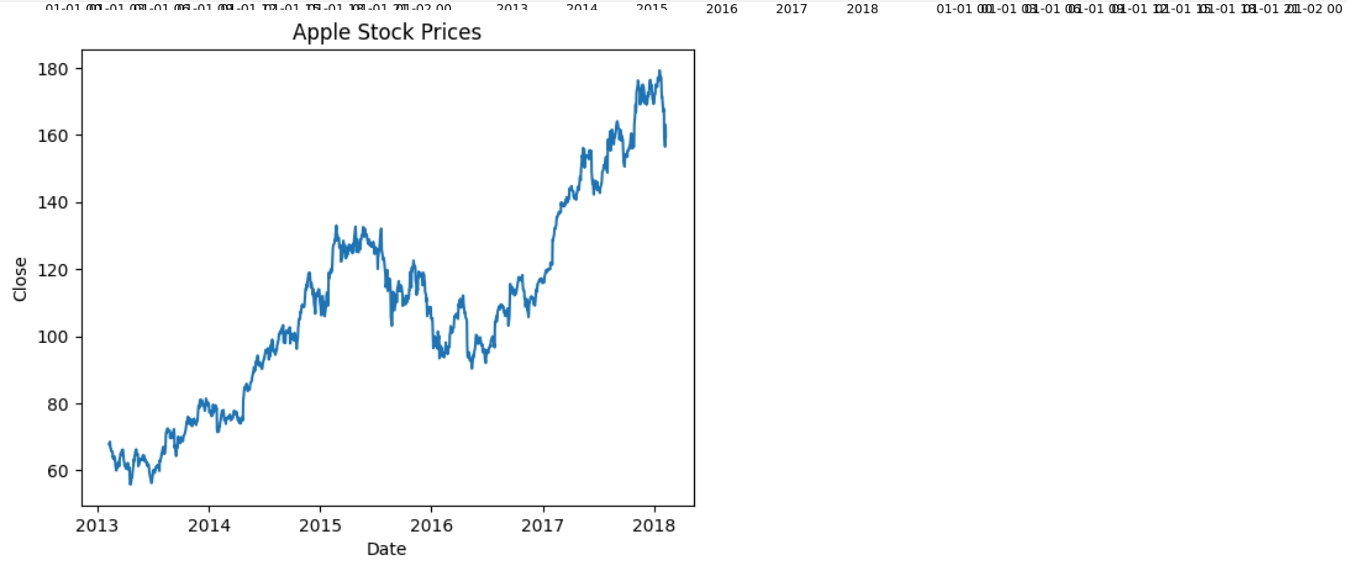
```
companies = ['AAPL', 'AMD', 'FB', 'GOOGL', 'AMZN', 'NVDA', 'EBAY', 'CSCO', 'IBM']

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.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()
```




```
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
```

```
                            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
```

keras.src.models.model.Model.summary
def summary(line_length=None, positions=None, print_fn=None, expand_nested=False, show_trainable=False, layer_range=None)

/usr/local/lib/python3.12/dist-packages/keras/src/models/model.py
Prints a string summary of the network.

Args:
    line_length: Total length of printed lines
        (e.g. set this to adapt the display to different

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

```
Epoch 1/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 6s 58ms/step - loss: 0.0533
Epoch 2/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - loss: 0.0087
Epoch 3/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 92ms/step - loss: 0.0090
Epoch 4/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 99ms/step - loss: 0.0090
Epoch 5/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 58ms/step - loss: 0.0083
Epoch 6/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - loss: 0.0075
Epoch 7/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 70ms/step - loss: 0.0071
Epoch 8/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 85ms/step - loss: 0.0065
Epoch 9/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - loss: 0.0061
Epoch 10/10
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 59ms/step - loss: 0.0066
```

```
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))

predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

mse = np.mean(((predictions - y_test) ** 2))
rmse = np.sqrt(mse)

print("MSE", mse)
print("RMSE", np.sqrt(mse))
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 425ms/step
MSE 45.63328004905825
RMSE 6.755240931977056
```

```
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 0x7dc867a727e0>
```