In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from dataprep.eda import plot, plot_missing, plot_correlation, create_report

from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```python
import tensorflow as tf
tf.__version__
```

Out[2]:
```
'2.12.0'
```

In [3]:
```python
df = pd.read_csv("all_stocks_5yr.csv")
df
```

Out[3]:

	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
...	...	...	...	...	...	...	...
619035	2018-02-01	76.84	78.27	76.69	77.82	2982259	ZTS
619036	2018-02-02	77.53	78.12	76.73	76.78	2595187	ZTS
619037	2018-02-05	76.64	76.92	73.18	73.83	2962031	ZTS
619038	2018-02-06	72.74	74.56	72.13	73.27	4924323	ZTS
619039	2018-02-07	72.70	75.00	72.69	73.86	4534912	ZTS

619040 rows × 7 columns

In [4]:
```python
print(len(df[df.Name == "GOOGL"]))
print(len(df[df.Name == "AAPL"]))
print(len(df[df.Name == "ZTS"]))
```

```
1259
1259
1259
```

From a quick search, it seems like all companies have the same number of entries. For this project, we have two approaches that I can think of. One is to include all the companies (or a big subset of them) and use them to train the LSTM. The second is to use one company to train the LSTM on. The first approach will likely (or hopefully) learn the patterns of the market as a whole, at least as it purtains to the S&P 500. It will be imprecise per company and per prediction as it's designed to be this way by not only focusing on one company and learning it's patterns and behaviors. The second option which is the one we're going to go with is to train the model on one specific company. We want to be precise in our forecasting as that seems more fun to me :) I really like Google so I will move forward with it for this assignment. It's also fitting as Google is an AI-focused company and they're the developers of tensorflow.

In [5]:
```python
df = df[df.Name == "GOOGL"]
df
```

Out[5]:

	date	open	high	low	close	volume	Name
250308	2013-02-08	390.4551	393.7283	390.1698	393.0777	6031199	GOOGL
250309	2013-02-11	389.5892	391.8915	387.2619	391.6012	4330781	GOOGL
250310	2013-02-12	391.2659	394.3440	390.0747	390.7403	3714176	GOOGL
250311	2013-02-13	390.4551	393.0677	390.3750	391.8214	2393946	GOOGL
250312	2013-02-14	390.2549	394.7644	389.2739	394.3039	3466971	GOOGL
...	...	...	...	...	...	...	...
251562	2018-02-01	1175.9900	1187.4500	1169.3600	1181.5900	3675709	GOOGL
251563	2018-02-02	1127.4200	1131.3000	1111.1700	1119.2000	5892122	GOOGL
251564	2018-02-05	1100.6100	1114.9900	1056.7400	1062.3900	4177469	GOOGL
251565	2018-02-06	1033.9800	1087.3800	1030.0100	1084.4300	3831524	GOOGL
251566	2018-02-07	1084.9700	1086.5300	1054.6200	1055.4100	2597094	GOOGL

1259 rows × 7 columns

In [6]:

```
df.dtypes
```

Out[6]:

```
date       object
open      float64
high      float64
low       float64
close     float64
volume      int64
Name       object
dtype: object
```

Timestamp is `str` type. let's change it to datetime

In [7]:

```
df.date = pd.to_datetime(df.date)
```

```
C:\Users\yousi\AppData\Local\Temp\ipykernel_22580\3331931521.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  df.date = pd.to_datetime(df.date)
```

In [8]:

```
df.describe(include="all")
```

```
C:\Users\yousi\AppData\Local\Temp\ipykernel_22580\1985922364.py:1: FutureWarning: Treating datetime data as categorical rather than n
umeric in `.describe` is deprecated and will be removed in a future version of pandas. Specify `datetime_is_numeric=True` to silence
this warning and adopt the future behavior now.
  df.describe(include="all")
```

Out[8]:

	date	open	high	low	close	volume	Name
count	1259	1259.000000	1259.000000	1259.000000	1259.000000	1.259000e+03	1259
unique	1259	NaN	NaN	NaN	NaN	NaN	1
top	2013-02-08 00:00:00	NaN	NaN	NaN	NaN	NaN	GOOGL
freq	1	NaN	NaN	NaN	NaN	NaN	1259
first	2013-02-08 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN
last	2018-02-07 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	682.357041	687.362776	676.691790	682.233847	2.457501e+06	NaN
std	NaN	187.409986	188.531563	186.265742	187.573892	1.591432e+06	NaN
min	NaN	384.964600	390.164800	381.010700	383.340000	5.211410e+05	NaN
25%	NaN	543.660000	547.585000	539.200000	543.022500	1.456867e+06	NaN
50%	NaN	651.570000	658.255500	642.165000	652.470000	1.938260e+06	NaN
75%	NaN	805.960000	810.739500	801.565000	806.400000	3.031624e+06	NaN
max	NaN	1188.000000	1198.000000	1184.060000	1187.560000	2.314537e+07	NaN

We see no null values which is good. We see a big range with the minimum `open` at about $385 and maximum at 1188. The fluctuation is reasonable as we're pulling in the data from February of 2013 to February of 2018, 5 full years worth of data.

In [9]:

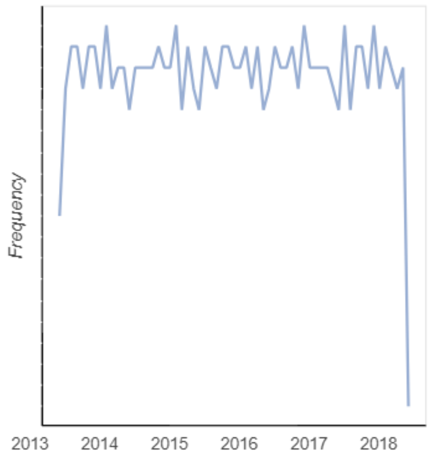
```
plot(df)
```

```
  0%|                                                                    | 0/326 [00:00<…
```

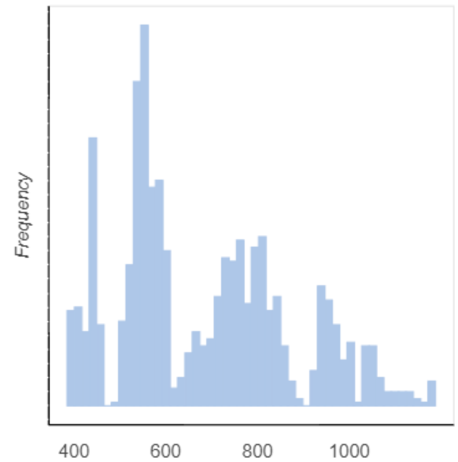
Out[9]:

Show Stats and Insights

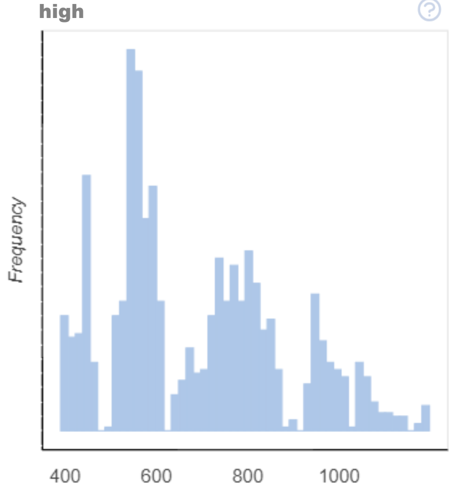
date



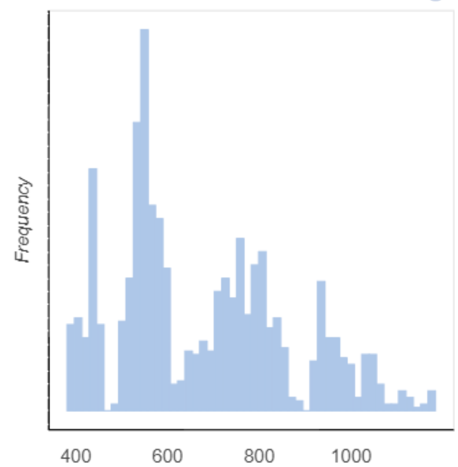
open



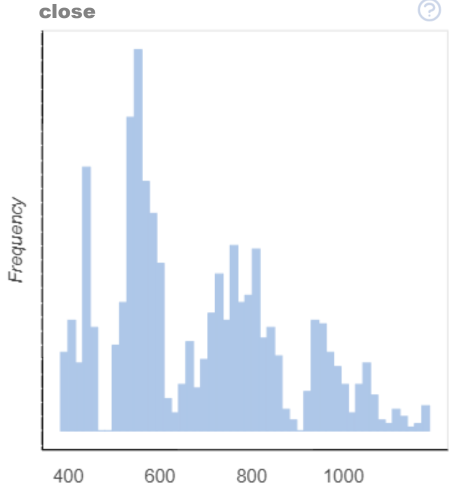
high



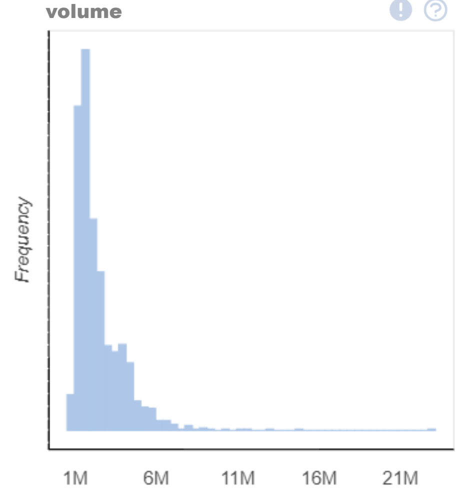
low



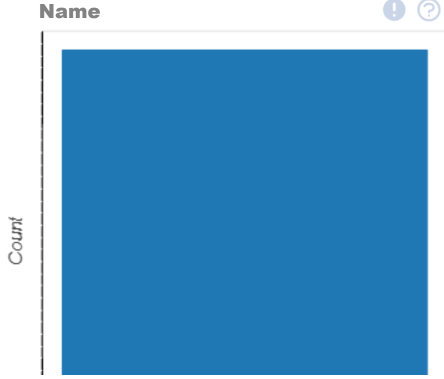
close

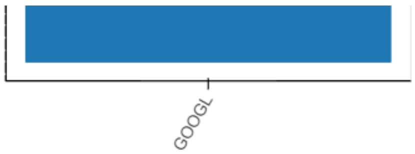


volume



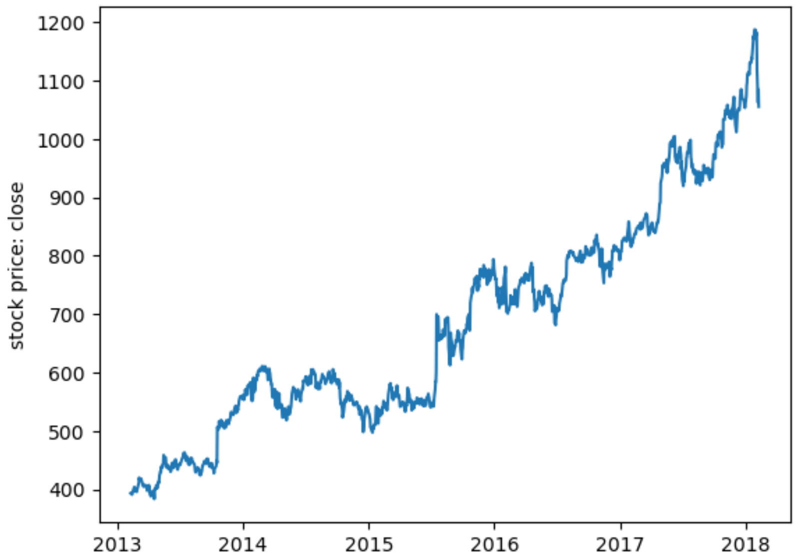
Name





We can see that the volume is skewed right which means it has a lot of outliers. This makes sense if we think of times where a company goes through big changes such as a high executive changes positions, the company releases a new product and so on.

In [10]:
```
plt.plot(df.date, df.close)
plt.ylabel("stock price: close")
plt.show()
```



We can see from the plot above that Google has been doing a solid job growing their stock value!

In [11]:
```
plot_correlation(df)
```

Out[11]:

Stats Pearson Spearman KendallTau

	Pearson	Spearman	KendallTau
Highest Positive Correlation	1.0	0.999	0.981
Highest Negative Correlation	-0.43	-0.552	-0.387
Lowest Correlation	0.418	0.536	0.374
Mean Correlation	0.344	0.306	0.348

`open`, `high`, `low`, and `close` are completely positively correlation which makes perfect sense given their relationship. What's interesting is that volume is negatively correlation to the rest of the columns. Meaning, the more trading that happens the lower the stock price drops. A big volume is typically a sign of volatility. It's usually linked to uncertain events. Traders typically like stability and thus it makes a lot of sense to see this negative correlation.

Now, I would like to add some features from our `date` column, namely, day of week, week of year, month, and even day of year, cause why not! We are spanning a lot of years so it makes sense to take advantage of what we can and hopefully find relationships in these engineered features.

In [12]:
```
df["date_dayofweek"] = df["date"].dt.dayofweek
df["date_weekofyear"] = df["date"].dt.isocalendar().week
df["date_month"] = df["date"].dt.month
df["date_dayofyear"] = df["date"].dt.dayofyear
```

```
C:\Users\yousi\AppData\Local\Temp\ipykernel_22580\3755081549.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  df["date_dayofweek"] = df["date"].dt.dayofweek
C:\Users\yousi\AppData\Local\Temp\ipykernel_22580\3755081549.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  df["date_weekofyear"] = df["date"].dt.isocalendar().week
C:\Users\yousi\AppData\Local\Temp\ipykernel_22580\3755081549.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  df["date_month"] = df["date"].dt.month
C:\Users\yousi\AppData\Local\Temp\ipykernel_22580\3755081549.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
-copy
  df["date_dayofyear"] = df["date"].dt.dayofyear
```

In [13]:
```python
plot_correlation(df)
```

Out[13]:

Stats | Pearson | Spearman | KendallTau

	Pearson	Spearman	KendallTau
Highest Positive Correlation	1.0	0.999	0.981
Highest Negative Correlation	-0.43	-0.552	-0.387
Lowest Correlation	0.008	0.008	0.006
Mean Correlation	0.196	0.182	0.191

We don't see a lot of correlation between the date columns and the stock price columns. Perhaps they're not actually adding much value, or, more palusibly, their affect isn't linear and that's why we're not seeing a big correlation. There are always more eda to do to find these relationships but we won't waste a lot more time on this for now and we'll jump into the modeling

I was thinking about including perhaps `open`, `high`, and `low`, and even `close` to predict `close` but because of the high correlation, they're likely to dominate over other features in the prediction. According to the message by professor Esmaeili in the course Slack channel saying that `close` is usually used in these types or predictions, I'll go ahead and use it.

As the data is sequential in nature, We shouldn't split it up randomly, thus we will pick the first 80% of the data for training and the remaining 20% for testing.

In [14]:
```python
features_to_use = ["close", "volume", "date_dayofweek", "date_dayofyear",
                   "date_month", "date_dayofyear"]

features_to_use_indices = [df.columns.get_loc(feature) for feature in features_to_use]

train_size = int(len(df) * 0.7)
test_size = len(df) - train_size

train_data = df.iloc[0:train_size,features_to_use_indices]
test_data = df.iloc[train_size:len(df),features_to_use_indices]

target = "close"
target_index = [train_data.columns.get_loc(target)]
```

Now, we will rescale our data using MinMaxScaler as NNs are sensitive to magnitude

In [15]:
```python
sc = MinMaxScaler()
sc_target = MinMaxScaler().fit(train_data[["close"]]) # this will make it easy to inverse_transform the target

train_data = pd.DataFrame(sc.fit_transform(train_data), columns=train_data.columns)
test_data = pd.DataFrame(sc.transform(test_data), columns=test_data.columns)
```

We will try a window size of 10 for 2 weeks as stock data is typically recorded Mon-Fri

In [16]:
```python
window_size = 10
ph = 1 # predictive horizon is how far in advance we want to do the prediction
```

```
       # in this case it's set to 1 meaning we're predicting the very next day
       # or first day next week if we're ending on a Sunday or a holiday
       # after the 10  day window or whatever window_size we choose

X_train, y_train = [], []
for i in range(window_size, len(train_data)-ph+1):
    X_train.append(train_data.iloc[i - window_size:i, :].values)
    y_train.append(train_data.iloc[i + ph - 1, target_index].values)
X_train, y_train = np.array(X_train), np.array(y_train)

# Create the testing data

X_test, y_test = [], []
for i in range(window_size, len(test_data)-ph+1):
    X_test.append(test_data.iloc[i - window_size:i, :].values)
    y_test.append(test_data.iloc[i + ph - 1, target_index].values)
X_test, y_test = np.array(X_test), np.array(y_test)
```

In [17]:
```
print(X_train.shape)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], len(features_to_use)))
```

```
(871, 10, 6)
```

In [18]:
```
# Build the LSTM model
model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(150, input_shape=(window_size, len(features_to_use))),
    tf.keras.layers.Dense(1)
])

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Train the model
model.fit(X_train, y_train, epochs=300, batch_size=64)
```

```
Epoch 1/300
14/14 [==============================] - 5s 26ms/step - loss: 0.0789
Epoch 2/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0112
Epoch 3/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0042
Epoch 4/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0024
Epoch 5/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0019
Epoch 6/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0019
Epoch 7/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0018
Epoch 8/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0017
Epoch 9/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0016
Epoch 10/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0016
Epoch 11/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0016
Epoch 12/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0016
Epoch 13/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0015
Epoch 14/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0015
Epoch 15/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0015
Epoch 16/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0015
Epoch 17/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0015
Epoch 18/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0014
Epoch 19/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0014
Epoch 20/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0015
Epoch 21/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0015
Epoch 22/300
14/14 [==============================] - 0s 25ms/step - loss: 0.0014
Epoch 23/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0014
Epoch 24/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0014
Epoch 25/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0013
Epoch 26/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0013
Epoch 27/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0013
Epoch 28/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0013
Epoch 29/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0013
Epoch 30/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0013
Epoch 31/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0013
Epoch 32/300
14/14 [==============================] - 0s 26ms/step - loss: 0.0012
Epoch 33/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0013
Epoch 34/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0012
Epoch 35/300
14/14 [==============================] - 0s 24ms/step - loss: 0.0013
Epoch 36/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0012
Epoch 37/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0012
Epoch 38/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0012
Epoch 39/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0012
Epoch 40/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 41/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 42/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 43/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
```

```
Epoch 44/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 45/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 46/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0012
Epoch 47/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 48/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 49/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0013
Epoch 50/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 51/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0010
Epoch 52/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0010
Epoch 53/300
14/14 [==============================] - 0s 23ms/step - loss: 9.9365e-04
Epoch 54/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 55/300
14/14 [==============================] - 0s 23ms/step - loss: 0.0011
Epoch 56/300
14/14 [==============================] - 0s 23ms/step - loss: 9.6681e-04
Epoch 57/300
14/14 [==============================] - 0s 23ms/step - loss: 9.5076e-04
Epoch 58/300
14/14 [==============================] - 0s 23ms/step - loss: 9.4567e-04
Epoch 59/300
14/14 [==============================] - 0s 24ms/step - loss: 9.4155e-04
Epoch 60/300
14/14 [==============================] - 0s 26ms/step - loss: 9.8296e-04
Epoch 61/300
14/14 [==============================] - 0s 23ms/step - loss: 9.7864e-04
Epoch 62/300
14/14 [==============================] - 0s 23ms/step - loss: 9.3018e-04
Epoch 63/300
14/14 [==============================] - 0s 23ms/step - loss: 9.2482e-04
Epoch 64/300
14/14 [==============================] - 0s 23ms/step - loss: 9.3702e-04
Epoch 65/300
14/14 [==============================] - 0s 24ms/step - loss: 8.8938e-04
Epoch 66/300
14/14 [==============================] - 0s 23ms/step - loss: 9.7829e-04
Epoch 67/300
14/14 [==============================] - 0s 23ms/step - loss: 9.3594e-04
Epoch 68/300
14/14 [==============================] - 0s 23ms/step - loss: 8.9332e-04
Epoch 69/300
14/14 [==============================] - 0s 24ms/step - loss: 8.9273e-04
Epoch 70/300
14/14 [==============================] - 0s 23ms/step - loss: 8.9623e-04
Epoch 71/300
14/14 [==============================] - 0s 23ms/step - loss: 8.9366e-04
Epoch 72/300
14/14 [==============================] - 0s 23ms/step - loss: 8.4103e-04
Epoch 73/300
14/14 [==============================] - 0s 23ms/step - loss: 8.5917e-04
Epoch 74/300
14/14 [==============================] - 0s 23ms/step - loss: 8.5974e-04
Epoch 75/300
14/14 [==============================] - 0s 22ms/step - loss: 8.9023e-04
Epoch 76/300
14/14 [==============================] - 0s 23ms/step - loss: 8.4378e-04
Epoch 77/300
14/14 [==============================] - 0s 23ms/step - loss: 8.2800e-04
Epoch 78/300
14/14 [==============================] - 0s 23ms/step - loss: 9.3694e-04
Epoch 79/300
14/14 [==============================] - 0s 24ms/step - loss: 8.2567e-04
Epoch 80/300
14/14 [==============================] - 0s 23ms/step - loss: 8.4897e-04
Epoch 81/300
14/14 [==============================] - 0s 23ms/step - loss: 8.5739e-04
Epoch 82/300
14/14 [==============================] - 0s 24ms/step - loss: 8.2213e-04
Epoch 83/300
14/14 [==============================] - 0s 23ms/step - loss: 8.2633e-04
Epoch 84/300
14/14 [==============================] - 0s 23ms/step - loss: 8.4244e-04
Epoch 85/300
14/14 [==============================] - 0s 23ms/step - loss: 9.1392e-04
Epoch 86/300
14/14 [==============================] - 0s 23ms/step - loss: 8.2813e-04
```

```
Epoch 87/300
14/14 [==============================] - 0s 23ms/step - loss: 7.9814e-04
Epoch 88/300
14/14 [==============================] - 0s 23ms/step - loss: 7.8154e-04
Epoch 89/300
14/14 [==============================] - 0s 23ms/step - loss: 7.9162e-04
Epoch 90/300
14/14 [==============================] - 0s 26ms/step - loss: 8.1996e-04
Epoch 91/300
14/14 [==============================] - 0s 23ms/step - loss: 7.7998e-04
Epoch 92/300
14/14 [==============================] - 0s 23ms/step - loss: 7.5161e-04
Epoch 93/300
14/14 [==============================] - 0s 22ms/step - loss: 7.8195e-04
Epoch 94/300
14/14 [==============================] - 0s 23ms/step - loss: 9.1170e-04
Epoch 95/300
14/14 [==============================] - 0s 23ms/step - loss: 8.5259e-04
Epoch 96/300
14/14 [==============================] - 0s 23ms/step - loss: 7.2254e-04
Epoch 97/300
14/14 [==============================] - 0s 24ms/step - loss: 7.3986e-04
Epoch 98/300
14/14 [==============================] - 0s 23ms/step - loss: 7.1569e-04
Epoch 99/300
14/14 [==============================] - 0s 24ms/step - loss: 7.4931e-04
Epoch 100/300
14/14 [==============================] - 0s 24ms/step - loss: 7.8186e-04
Epoch 101/300
14/14 [==============================] - 0s 24ms/step - loss: 7.6072e-04
Epoch 102/300
14/14 [==============================] - 0s 23ms/step - loss: 7.6099e-04
Epoch 103/300
14/14 [==============================] - 0s 23ms/step - loss: 7.0799e-04
Epoch 104/300
14/14 [==============================] - 0s 23ms/step - loss: 7.2319e-04
Epoch 105/300
14/14 [==============================] - 0s 24ms/step - loss: 7.3496e-04
Epoch 106/300
14/14 [==============================] - 0s 24ms/step - loss: 6.7568e-04
Epoch 107/300
14/14 [==============================] - 0s 23ms/step - loss: 7.2958e-04
Epoch 108/300
14/14 [==============================] - 0s 24ms/step - loss: 6.7816e-04
Epoch 109/300
14/14 [==============================] - 0s 24ms/step - loss: 6.7217e-04
Epoch 110/300
14/14 [==============================] - 0s 24ms/step - loss: 7.0442e-04
Epoch 111/300
14/14 [==============================] - 0s 24ms/step - loss: 7.1422e-04
Epoch 112/300
14/14 [==============================] - 0s 23ms/step - loss: 6.7239e-04
Epoch 113/300
14/14 [==============================] - 0s 22ms/step - loss: 6.7460e-04
Epoch 114/300
14/14 [==============================] - 0s 23ms/step - loss: 6.5976e-04
Epoch 115/300
14/14 [==============================] - 0s 23ms/step - loss: 7.7965e-04
Epoch 116/300
14/14 [==============================] - 0s 23ms/step - loss: 7.1658e-04
Epoch 117/300
14/14 [==============================] - 0s 24ms/step - loss: 7.1352e-04
Epoch 118/300
14/14 [==============================] - 0s 23ms/step - loss: 6.4620e-04
Epoch 119/300
14/14 [==============================] - 0s 24ms/step - loss: 6.8564e-04
Epoch 120/300
14/14 [==============================] - 0s 24ms/step - loss: 6.4139e-04
Epoch 121/300
14/14 [==============================] - 0s 24ms/step - loss: 6.1449e-04
Epoch 122/300
14/14 [==============================] - 0s 24ms/step - loss: 6.2637e-04
Epoch 123/300
14/14 [==============================] - 0s 23ms/step - loss: 6.3421e-04
Epoch 124/300
14/14 [==============================] - 0s 24ms/step - loss: 6.4454e-04
Epoch 125/300
14/14 [==============================] - 0s 23ms/step - loss: 6.4158e-04
Epoch 126/300
14/14 [==============================] - 0s 23ms/step - loss: 6.2649e-04
Epoch 127/300
14/14 [==============================] - 0s 23ms/step - loss: 6.2466e-04
Epoch 128/300
14/14 [==============================] - 0s 23ms/step - loss: 6.7777e-04
Epoch 129/300
14/14 [==============================] - 0s 25ms/step - loss: 6.0870e-04
```

```
Epoch 130/300
14/14 [==============================] - 0s 24ms/step - loss: 6.0799e-04
Epoch 131/300
14/14 [==============================] - 0s 23ms/step - loss: 6.1573e-04
Epoch 132/300
14/14 [==============================] - 0s 23ms/step - loss: 6.1923e-04
Epoch 133/300
14/14 [==============================] - 0s 24ms/step - loss: 5.9954e-04
Epoch 134/300
14/14 [==============================] - 0s 23ms/step - loss: 5.9925e-04
Epoch 135/300
14/14 [==============================] - 0s 24ms/step - loss: 6.7758e-04
Epoch 136/300
14/14 [==============================] - 0s 23ms/step - loss: 5.7562e-04
Epoch 137/300
14/14 [==============================] - 0s 23ms/step - loss: 5.7728e-04
Epoch 138/300
14/14 [==============================] - 0s 23ms/step - loss: 5.8220e-04
Epoch 139/300
14/14 [==============================] - 0s 23ms/step - loss: 5.7397e-04
Epoch 140/300
14/14 [==============================] - 0s 24ms/step - loss: 6.0526e-04
Epoch 141/300
14/14 [==============================] - 0s 24ms/step - loss: 5.6318e-04
Epoch 142/300
14/14 [==============================] - 0s 23ms/step - loss: 5.6670e-04
Epoch 143/300
14/14 [==============================] - 0s 23ms/step - loss: 5.9221e-04
Epoch 144/300
14/14 [==============================] - 0s 23ms/step - loss: 6.1823e-04
Epoch 145/300
14/14 [==============================] - 0s 24ms/step - loss: 5.6097e-04
Epoch 146/300
14/14 [==============================] - 0s 23ms/step - loss: 5.7495e-04
Epoch 147/300
14/14 [==============================] - 0s 23ms/step - loss: 6.1349e-04
Epoch 148/300
14/14 [==============================] - 0s 24ms/step - loss: 5.7443e-04
Epoch 149/300
14/14 [==============================] - 0s 23ms/step - loss: 5.8431e-04
Epoch 150/300
14/14 [==============================] - 0s 26ms/step - loss: 5.4800e-04
Epoch 151/300
14/14 [==============================] - 0s 24ms/step - loss: 6.1157e-04
Epoch 152/300
14/14 [==============================] - 0s 24ms/step - loss: 5.4162e-04
Epoch 153/300
14/14 [==============================] - 0s 23ms/step - loss: 5.3967e-04
Epoch 154/300
14/14 [==============================] - 0s 23ms/step - loss: 5.5962e-04
Epoch 155/300
14/14 [==============================] - 0s 23ms/step - loss: 5.3188e-04
Epoch 156/300
14/14 [==============================] - 0s 24ms/step - loss: 5.4897e-04
Epoch 157/300
14/14 [==============================] - 0s 23ms/step - loss: 6.4046e-04
Epoch 158/300
14/14 [==============================] - 0s 24ms/step - loss: 5.4882e-04
Epoch 159/300
14/14 [==============================] - 0s 23ms/step - loss: 5.4194e-04
Epoch 160/300
14/14 [==============================] - 0s 24ms/step - loss: 5.4554e-04
Epoch 161/300
14/14 [==============================] - 0s 23ms/step - loss: 5.8557e-04
Epoch 162/300
14/14 [==============================] - 0s 23ms/step - loss: 5.5163e-04
Epoch 163/300
14/14 [==============================] - 0s 23ms/step - loss: 5.8836e-04
Epoch 164/300
14/14 [==============================] - 0s 20ms/step - loss: 5.6192e-04
Epoch 165/300
14/14 [==============================] - 0s 22ms/step - loss: 5.2611e-04
Epoch 166/300
14/14 [==============================] - 0s 21ms/step - loss: 5.4773e-04
Epoch 167/300
14/14 [==============================] - 0s 22ms/step - loss: 6.0590e-04
Epoch 168/300
14/14 [==============================] - 0s 22ms/step - loss: 5.2351e-04
Epoch 169/300
14/14 [==============================] - 0s 23ms/step - loss: 5.2099e-04
Epoch 170/300
14/14 [==============================] - 0s 22ms/step - loss: 5.1299e-04
Epoch 171/300
14/14 [==============================] - 0s 21ms/step - loss: 5.0779e-04
Epoch 172/300
14/14 [==============================] - 0s 21ms/step - loss: 4.9227e-04
```

```
Epoch 173/300
14/14 [==============================] - 0s 21ms/step - loss: 4.9439e-04
Epoch 174/300
14/14 [==============================] - 0s 23ms/step - loss: 5.0154e-04
Epoch 175/300
14/14 [==============================] - 0s 21ms/step - loss: 5.3541e-04
Epoch 176/300
14/14 [==============================] - 0s 22ms/step - loss: 5.0675e-04
Epoch 177/300
14/14 [==============================] - 0s 22ms/step - loss: 5.1835e-04
Epoch 178/300
14/14 [==============================] - 0s 22ms/step - loss: 5.6018e-04
Epoch 179/300
14/14 [==============================] - 0s 23ms/step - loss: 5.4456e-04
Epoch 180/300
14/14 [==============================] - 0s 26ms/step - loss: 4.9363e-04
Epoch 181/300
14/14 [==============================] - 0s 23ms/step - loss: 5.0519e-04
Epoch 182/300
14/14 [==============================] - 0s 23ms/step - loss: 4.9973e-04
Epoch 183/300
14/14 [==============================] - 0s 23ms/step - loss: 5.2408e-04
Epoch 184/300
14/14 [==============================] - 0s 23ms/step - loss: 5.7637e-04
Epoch 185/300
14/14 [==============================] - 0s 23ms/step - loss: 5.8932e-04
Epoch 186/300
14/14 [==============================] - 0s 21ms/step - loss: 5.7023e-04
Epoch 187/300
14/14 [==============================] - 0s 21ms/step - loss: 5.0093e-04
Epoch 188/300
14/14 [==============================] - 0s 23ms/step - loss: 5.1530e-04
Epoch 189/300
14/14 [==============================] - 0s 23ms/step - loss: 5.3695e-04
Epoch 190/300
14/14 [==============================] - 0s 25ms/step - loss: 5.1078e-04
Epoch 191/300
14/14 [==============================] - 0s 25ms/step - loss: 4.8529e-04
Epoch 192/300
14/14 [==============================] - 0s 24ms/step - loss: 4.9210e-04
Epoch 193/300
14/14 [==============================] - 0s 25ms/step - loss: 4.9105e-04
Epoch 194/300
14/14 [==============================] - 0s 25ms/step - loss: 5.5192e-04
Epoch 195/300
14/14 [==============================] - 0s 25ms/step - loss: 5.1400e-04
Epoch 196/300
14/14 [==============================] - 0s 28ms/step - loss: 5.4468e-04
Epoch 197/300
14/14 [==============================] - 0s 27ms/step - loss: 5.3565e-04
Epoch 198/300
14/14 [==============================] - 0s 27ms/step - loss: 5.0352e-04
Epoch 199/300
14/14 [==============================] - 0s 26ms/step - loss: 4.9377e-04
Epoch 200/300
14/14 [==============================] - 0s 27ms/step - loss: 4.8568e-04
Epoch 201/300
14/14 [==============================] - 0s 27ms/step - loss: 5.2577e-04
Epoch 202/300
14/14 [==============================] - 0s 25ms/step - loss: 5.5830e-04
Epoch 203/300
14/14 [==============================] - 0s 29ms/step - loss: 5.1859e-04
Epoch 204/300
14/14 [==============================] - 0s 25ms/step - loss: 4.8171e-04
Epoch 205/300
14/14 [==============================] - 0s 26ms/step - loss: 4.7253e-04
Epoch 206/300
14/14 [==============================] - 0s 26ms/step - loss: 4.7081e-04
Epoch 207/300
14/14 [==============================] - 0s 26ms/step - loss: 4.7438e-04
Epoch 208/300
14/14 [==============================] - 0s 25ms/step - loss: 4.8101e-04
Epoch 209/300
14/14 [==============================] - 0s 26ms/step - loss: 5.1923e-04
Epoch 210/300
14/14 [==============================] - 0s 27ms/step - loss: 5.0824e-04
Epoch 211/300
14/14 [==============================] - 0s 26ms/step - loss: 4.8827e-04
Epoch 212/300
14/14 [==============================] - 0s 24ms/step - loss: 4.8450e-04
Epoch 213/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5726e-04
Epoch 214/300
14/14 [==============================] - 0s 25ms/step - loss: 5.5220e-04
Epoch 215/300
14/14 [==============================] - 0s 24ms/step - loss: 5.2586e-04
```

```
Epoch 216/300
14/14 [==============================] - 0s 24ms/step - loss: 4.7186e-04
Epoch 217/300
14/14 [==============================] - 0s 25ms/step - loss: 4.8185e-04
Epoch 218/300
14/14 [==============================] - 0s 24ms/step - loss: 5.0193e-04
Epoch 219/300
14/14 [==============================] - 0s 24ms/step - loss: 5.0588e-04
Epoch 220/300
14/14 [==============================] - 0s 24ms/step - loss: 5.1275e-04
Epoch 221/300
14/14 [==============================] - 0s 25ms/step - loss: 4.7830e-04
Epoch 222/300
14/14 [==============================] - 0s 28ms/step - loss: 4.6383e-04
Epoch 223/300
14/14 [==============================] - 0s 27ms/step - loss: 4.9111e-04
Epoch 224/300
14/14 [==============================] - 0s 28ms/step - loss: 4.7991e-04
Epoch 225/300
14/14 [==============================] - 0s 25ms/step - loss: 5.5616e-04
Epoch 226/300
14/14 [==============================] - 0s 31ms/step - loss: 5.5741e-04
Epoch 227/300
14/14 [==============================] - 0s 32ms/step - loss: 4.6235e-04
Epoch 228/300
14/14 [==============================] - 0s 29ms/step - loss: 5.1036e-04
Epoch 229/300
14/14 [==============================] - 0s 29ms/step - loss: 4.6491e-04
Epoch 230/300
14/14 [==============================] - 1s 41ms/step - loss: 4.6543e-04
Epoch 231/300
14/14 [==============================] - 1s 36ms/step - loss: 4.5911e-04
Epoch 232/300
14/14 [==============================] - 0s 27ms/step - loss: 4.7959e-04
Epoch 233/300
14/14 [==============================] - 0s 26ms/step - loss: 4.6158e-04
Epoch 234/300
14/14 [==============================] - 0s 26ms/step - loss: 4.7594e-04
Epoch 235/300
14/14 [==============================] - 0s 25ms/step - loss: 4.7871e-04
Epoch 236/300
14/14 [==============================] - 0s 25ms/step - loss: 4.5858e-04
Epoch 237/300
14/14 [==============================] - 0s 25ms/step - loss: 4.6031e-04
Epoch 238/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5750e-04
Epoch 239/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5046e-04
Epoch 240/300
14/14 [==============================] - 0s 27ms/step - loss: 4.6603e-04
Epoch 241/300
14/14 [==============================] - 0s 28ms/step - loss: 5.5713e-04
Epoch 242/300
14/14 [==============================] - 0s 25ms/step - loss: 5.0969e-04
Epoch 243/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5542e-04
Epoch 244/300
14/14 [==============================] - 0s 28ms/step - loss: 4.5658e-04
Epoch 245/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5432e-04
Epoch 246/300
14/14 [==============================] - 0s 24ms/step - loss: 4.8842e-04
Epoch 247/300
14/14 [==============================] - 0s 24ms/step - loss: 4.6948e-04
Epoch 248/300
14/14 [==============================] - 0s 23ms/step - loss: 4.4847e-04
Epoch 249/300
14/14 [==============================] - 0s 24ms/step - loss: 4.4738e-04
Epoch 250/300
14/14 [==============================] - 0s 23ms/step - loss: 4.6831e-04
Epoch 251/300
14/14 [==============================] - 0s 23ms/step - loss: 4.8098e-04
Epoch 252/300
14/14 [==============================] - 0s 23ms/step - loss: 5.0114e-04
Epoch 253/300
14/14 [==============================] - 0s 23ms/step - loss: 5.2787e-04
Epoch 254/300
14/14 [==============================] - 0s 23ms/step - loss: 5.5242e-04
Epoch 255/300
14/14 [==============================] - 0s 24ms/step - loss: 5.3969e-04
Epoch 256/300
14/14 [==============================] - 0s 22ms/step - loss: 4.7631e-04
Epoch 257/300
14/14 [==============================] - 0s 23ms/step - loss: 4.5884e-04
Epoch 258/300
14/14 [==============================] - 0s 22ms/step - loss: 4.8499e-04
```

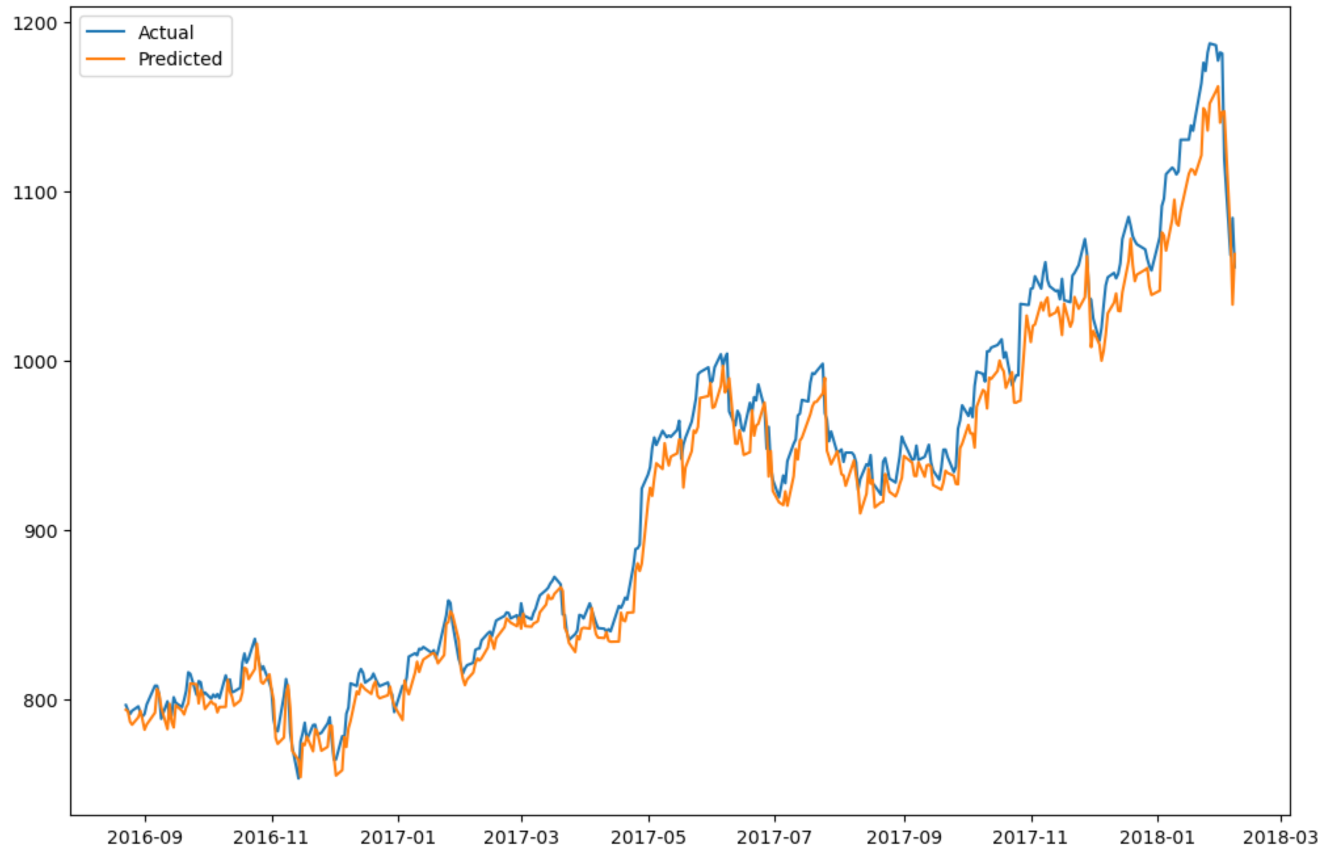
```
Epoch 259/300
14/14 [==============================] - 0s 23ms/step - loss: 4.7721e-04
Epoch 260/300
14/14 [==============================] - 0s 22ms/step - loss: 4.6364e-04
Epoch 261/300
14/14 [==============================] - 0s 23ms/step - loss: 4.8706e-04
Epoch 262/300
14/14 [==============================] - 0s 23ms/step - loss: 4.5652e-04
Epoch 263/300
14/14 [==============================] - 0s 23ms/step - loss: 5.7140e-04
Epoch 264/300
14/14 [==============================] - 0s 23ms/step - loss: 4.9467e-04
Epoch 265/300
14/14 [==============================] - 0s 23ms/step - loss: 4.5435e-04
Epoch 266/300
14/14 [==============================] - 0s 23ms/step - loss: 4.6931e-04
Epoch 267/300
14/14 [==============================] - 0s 23ms/step - loss: 4.6818e-04
Epoch 268/300
14/14 [==============================] - 0s 26ms/step - loss: 4.3983e-04
Epoch 269/300
14/14 [==============================] - 0s 24ms/step - loss: 4.4524e-04
Epoch 270/300
14/14 [==============================] - 0s 24ms/step - loss: 5.0288e-04
Epoch 271/300
14/14 [==============================] - 0s 23ms/step - loss: 4.7425e-04
Epoch 272/300
14/14 [==============================] - 0s 23ms/step - loss: 5.0024e-04
Epoch 273/300
14/14 [==============================] - 0s 24ms/step - loss: 4.4882e-04
Epoch 274/300
14/14 [==============================] - 0s 23ms/step - loss: 4.7681e-04
Epoch 275/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5696e-04
Epoch 276/300
14/14 [==============================] - 0s 24ms/step - loss: 4.4276e-04
Epoch 277/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5795e-04
Epoch 278/300
14/14 [==============================] - 0s 24ms/step - loss: 4.6309e-04
Epoch 279/300
14/14 [==============================] - 0s 23ms/step - loss: 4.6090e-04
Epoch 280/300
14/14 [==============================] - 0s 24ms/step - loss: 4.7937e-04
Epoch 281/300
14/14 [==============================] - 0s 24ms/step - loss: 4.4395e-04
Epoch 282/300
14/14 [==============================] - 0s 24ms/step - loss: 4.7639e-04
Epoch 283/300
14/14 [==============================] - 0s 23ms/step - loss: 4.5623e-04
Epoch 284/300
14/14 [==============================] - 0s 23ms/step - loss: 4.5162e-04
Epoch 285/300
14/14 [==============================] - 0s 24ms/step - loss: 4.5218e-04
Epoch 286/300
14/14 [==============================] - 0s 26ms/step - loss: 4.5399e-04
Epoch 287/300
14/14 [==============================] - 0s 29ms/step - loss: 4.4071e-04
Epoch 288/300
14/14 [==============================] - 0s 28ms/step - loss: 4.7249e-04
Epoch 289/300
14/14 [==============================] - 0s 29ms/step - loss: 4.9164e-04
Epoch 290/300
14/14 [==============================] - 0s 28ms/step - loss: 4.7003e-04
Epoch 291/300
14/14 [==============================] - 0s 26ms/step - loss: 4.9818e-04
Epoch 292/300
14/14 [==============================] - 0s 26ms/step - loss: 5.8096e-04
Epoch 293/300
14/14 [==============================] - 0s 31ms/step - loss: 4.8870e-04
Epoch 294/300
14/14 [==============================] - 0s 29ms/step - loss: 4.5937e-04
Epoch 295/300
14/14 [==============================] - 0s 27ms/step - loss: 5.1846e-04
Epoch 296/300
14/14 [==============================] - 0s 25ms/step - loss: 5.2225e-04
Epoch 297/300
14/14 [==============================] - 0s 24ms/step - loss: 5.0101e-04
Epoch 298/300
14/14 [==============================] - 0s 24ms/step - loss: 4.9822e-04
Epoch 299/300
14/14 [==============================] - 0s 24ms/step - loss: 4.8815e-04
Epoch 300/300
14/14 [==============================] - 0s 25ms/step - loss: 4.9928e-04
```

```
Out[18]: <keras.callbacks.History at 0x157f5babc70>
```

In [20]:
```
predictions = model.predict(X_test)
predictions = sc_target.inverse_transform(predictions)

plt.figure(figsize=(12, 8))
plt.plot(df.iloc[-test_size+window_size:,0], sc_target.inverse_transform(y_test), label='Actual')
plt.plot(df.iloc[-test_size+window_size:,0], predictions, label='Predicted')
plt.legend()
plt.show()
```

```
12/12 [==============================] - 0s 9ms/step
```



In [21]:
```
mse = model.evaluate(X_test, y_test)
print("RMSE:", np.sqrt(sc_target.inverse_transform([[mse]]))[0][0])
```

```
12/12 [==============================] - 1s 11ms/step - loss: 0.0015
RMSE: 19.59482877158062
```

The model behaved incredibly well with a RMSE of about $\$19.50$ (after inverse transforming the result) which means that on average, we're off by less than $\$20$. I think that this result is pretty good for the simple LSTM that we created. Let's change the architecture to make it more complex and see if we can beat that. Let's also keep in mind that the predictive horizon is just 1 meaning we're only predicting 1 day in advance so if we were to change it to a greater number, we'll very likely see worse results.
by the way, I increased the number of neurons in the LSTM layer from 50 to 150 and, while it looks better in the visualization, the RMSE pretty much stayed the same

In [46]:
```
# Build the LSTM model
model2 = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(units=150, input_shape=(window_size, len(features_to_use)),
                        return_sequences=True),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.LSTM(units=20, return_sequences=False),
    tf.keras.layers.Dense(1)
])

# Compile the model
model2.compile(loss='mean_squared_error', optimizer="adam", metrics=["mse"])

# Train the model
num_epochs = 300
batch_size = 8
model_path = "model.h5"

history = model2.fit(X_train, y_train, epochs=num_epochs, batch_size=batch_size, validation_split=0.05, verbose=2,
          callbacks = [tf.keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=0, mode='min'),
                       tf.keras.callbacks.ModelCheckpoint(model_path,monitor='val_loss', save_best_only=True, mode='min', verbose=0)]
          )
```

```
Epoch 1/300
104/104 - 12s - loss: 0.0105 - mse: 0.0105 - val_loss: 0.0030 - val_mse: 0.0030 - 12s/epoch - 117ms/step
Epoch 2/300
104/104 - 2s - loss: 0.0029 - mse: 0.0029 - val_loss: 0.0025 - val_mse: 0.0025 - 2s/epoch - 23ms/step
Epoch 3/300
104/104 - 2s - loss: 0.0021 - mse: 0.0021 - val_loss: 0.0022 - val_mse: 0.0022 - 2s/epoch - 23ms/step
Epoch 4/300
104/104 - 2s - loss: 0.0026 - mse: 0.0026 - val_loss: 0.0022 - val_mse: 0.0022 - 2s/epoch - 24ms/step
Epoch 5/300
104/104 - 2s - loss: 0.0024 - mse: 0.0024 - val_loss: 0.0054 - val_mse: 0.0054 - 2s/epoch - 23ms/step
Epoch 6/300
104/104 - 2s - loss: 0.0024 - mse: 0.0024 - val_loss: 0.0020 - val_mse: 0.0020 - 2s/epoch - 23ms/step
Epoch 7/300
104/104 - 2s - loss: 0.0018 - mse: 0.0018 - val_loss: 0.0034 - val_mse: 0.0034 - 2s/epoch - 23ms/step
Epoch 8/300
104/104 - 2s - loss: 0.0019 - mse: 0.0019 - val_loss: 0.0018 - val_mse: 0.0018 - 2s/epoch - 24ms/step
Epoch 9/300
104/104 - 2s - loss: 0.0020 - mse: 0.0020 - val_loss: 0.0017 - val_mse: 0.0017 - 2s/epoch - 24ms/step
Epoch 10/300
104/104 - 2s - loss: 0.0019 - mse: 0.0019 - val_loss: 0.0040 - val_mse: 0.0040 - 2s/epoch - 23ms/step
Epoch 11/300
104/104 - 2s - loss: 0.0018 - mse: 0.0018 - val_loss: 0.0018 - val_mse: 0.0018 - 2s/epoch - 23ms/step
Epoch 12/300
104/104 - 2s - loss: 0.0014 - mse: 0.0014 - val_loss: 0.0014 - val_mse: 0.0014 - 2s/epoch - 23ms/step
Epoch 13/300
104/104 - 2s - loss: 0.0015 - mse: 0.0015 - val_loss: 0.0021 - val_mse: 0.0021 - 2s/epoch - 23ms/step
Epoch 14/300
104/104 - 2s - loss: 0.0017 - mse: 0.0017 - val_loss: 0.0022 - val_mse: 0.0022 - 2s/epoch - 24ms/step
Epoch 15/300
104/104 - 2s - loss: 0.0014 - mse: 0.0014 - val_loss: 0.0011 - val_mse: 0.0011 - 2s/epoch - 24ms/step
Epoch 16/300
104/104 - 2s - loss: 0.0014 - mse: 0.0014 - val_loss: 0.0025 - val_mse: 0.0025 - 2s/epoch - 23ms/step
Epoch 17/300
104/104 - 2s - loss: 0.0013 - mse: 0.0013 - val_loss: 0.0012 - val_mse: 0.0012 - 2s/epoch - 23ms/step
Epoch 18/300
104/104 - 3s - loss: 0.0012 - mse: 0.0012 - val_loss: 0.0010 - val_mse: 0.0010 - 3s/epoch - 24ms/step
Epoch 19/300
104/104 - 2s - loss: 0.0013 - mse: 0.0013 - val_loss: 0.0016 - val_mse: 0.0016 - 2s/epoch - 24ms/step
Epoch 20/300
104/104 - 2s - loss: 0.0013 - mse: 0.0013 - val_loss: 8.9159e-04 - val_mse: 8.9159e-04 - 2s/epoch - 24ms/step
Epoch 21/300
104/104 - 2s - loss: 0.0011 - mse: 0.0011 - val_loss: 0.0022 - val_mse: 0.0022 - 2s/epoch - 24ms/step
Epoch 22/300
104/104 - 2s - loss: 0.0013 - mse: 0.0013 - val_loss: 0.0012 - val_mse: 0.0012 - 2s/epoch - 23ms/step
Epoch 23/300
104/104 - 2s - loss: 0.0011 - mse: 0.0011 - val_loss: 0.0030 - val_mse: 0.0030 - 2s/epoch - 23ms/step
Epoch 24/300
104/104 - 2s - loss: 0.0011 - mse: 0.0011 - val_loss: 9.6658e-04 - val_mse: 9.6658e-04 - 2s/epoch - 24ms/step
Epoch 25/300
104/104 - 2s - loss: 0.0011 - mse: 0.0011 - val_loss: 0.0011 - val_mse: 0.0011 - 2s/epoch - 23ms/step
Epoch 26/300
104/104 - 3s - loss: 0.0011 - mse: 0.0011 - val_loss: 8.8038e-04 - val_mse: 8.8038e-04 - 3s/epoch - 24ms/step
Epoch 27/300
104/104 - 2s - loss: 0.0010 - mse: 0.0010 - val_loss: 8.9873e-04 - val_mse: 8.9873e-04 - 2s/epoch - 23ms/step
Epoch 28/300
104/104 - 2s - loss: 9.3174e-04 - mse: 9.3174e-04 - val_loss: 0.0011 - val_mse: 0.0011 - 2s/epoch - 23ms/step
Epoch 29/300
104/104 - 2s - loss: 0.0010 - mse: 0.0010 - val_loss: 0.0018 - val_mse: 0.0018 - 2s/epoch - 23ms/step
Epoch 30/300
104/104 - 2s - loss: 0.0010 - mse: 0.0010 - val_loss: 8.4222e-04 - val_mse: 8.4222e-04 - 2s/epoch - 24ms/step
Epoch 31/300
104/104 - 2s - loss: 9.5604e-04 - mse: 9.5604e-04 - val_loss: 8.2997e-04 - val_mse: 8.2997e-04 - 2s/epoch - 24ms/step
Epoch 32/300
104/104 - 2s - loss: 9.1656e-04 - mse: 9.1656e-04 - val_loss: 0.0019 - val_mse: 0.0019 - 2s/epoch - 23ms/step
Epoch 33/300
104/104 - 2s - loss: 9.0548e-04 - mse: 9.0548e-04 - val_loss: 0.0011 - val_mse: 0.0011 - 2s/epoch - 23ms/step
Epoch 34/300
104/104 - 2s - loss: 0.0010 - mse: 0.0010 - val_loss: 0.0011 - val_mse: 0.0011 - 2s/epoch - 23ms/step
Epoch 35/300
104/104 - 2s - loss: 9.2540e-04 - mse: 9.2540e-04 - val_loss: 0.0020 - val_mse: 0.0020 - 2s/epoch - 23ms/step
Epoch 36/300
104/104 - 2s - loss: 9.6684e-04 - mse: 9.6684e-04 - val_loss: 9.7062e-04 - val_mse: 9.7062e-04 - 2s/epoch - 23ms/step
Epoch 37/300
104/104 - 2s - loss: 8.8073e-04 - mse: 8.8073e-04 - val_loss: 9.8882e-04 - val_mse: 9.8882e-04 - 2s/epoch - 23ms/step
Epoch 38/300
104/104 - 2s - loss: 9.0769e-04 - mse: 9.0769e-04 - val_loss: 0.0013 - val_mse: 0.0013 - 2s/epoch - 23ms/step
Epoch 39/300
104/104 - 2s - loss: 9.2730e-04 - mse: 9.2730e-04 - val_loss: 7.3426e-04 - val_mse: 7.3426e-04 - 2s/epoch - 23ms/step
Epoch 40/300
104/104 - 2s - loss: 8.3041e-04 - mse: 8.3041e-04 - val_loss: 6.8989e-04 - val_mse: 6.8989e-04 - 2s/epoch - 23ms/step
Epoch 41/300
104/104 - 2s - loss: 8.6387e-04 - mse: 8.6387e-04 - val_loss: 7.4269e-04 - val_mse: 7.4269e-04 - 2s/epoch - 23ms/step
Epoch 42/300
104/104 - 2s - loss: 8.8992e-04 - mse: 8.8992e-04 - val_loss: 7.1199e-04 - val_mse: 7.1199e-04 - 2s/epoch - 23ms/step
Epoch 43/300
104/104 - 2s - loss: 7.7368e-04 - mse: 7.7368e-04 - val_loss: 0.0013 - val_mse: 0.0013 - 2s/epoch - 23ms/step
```

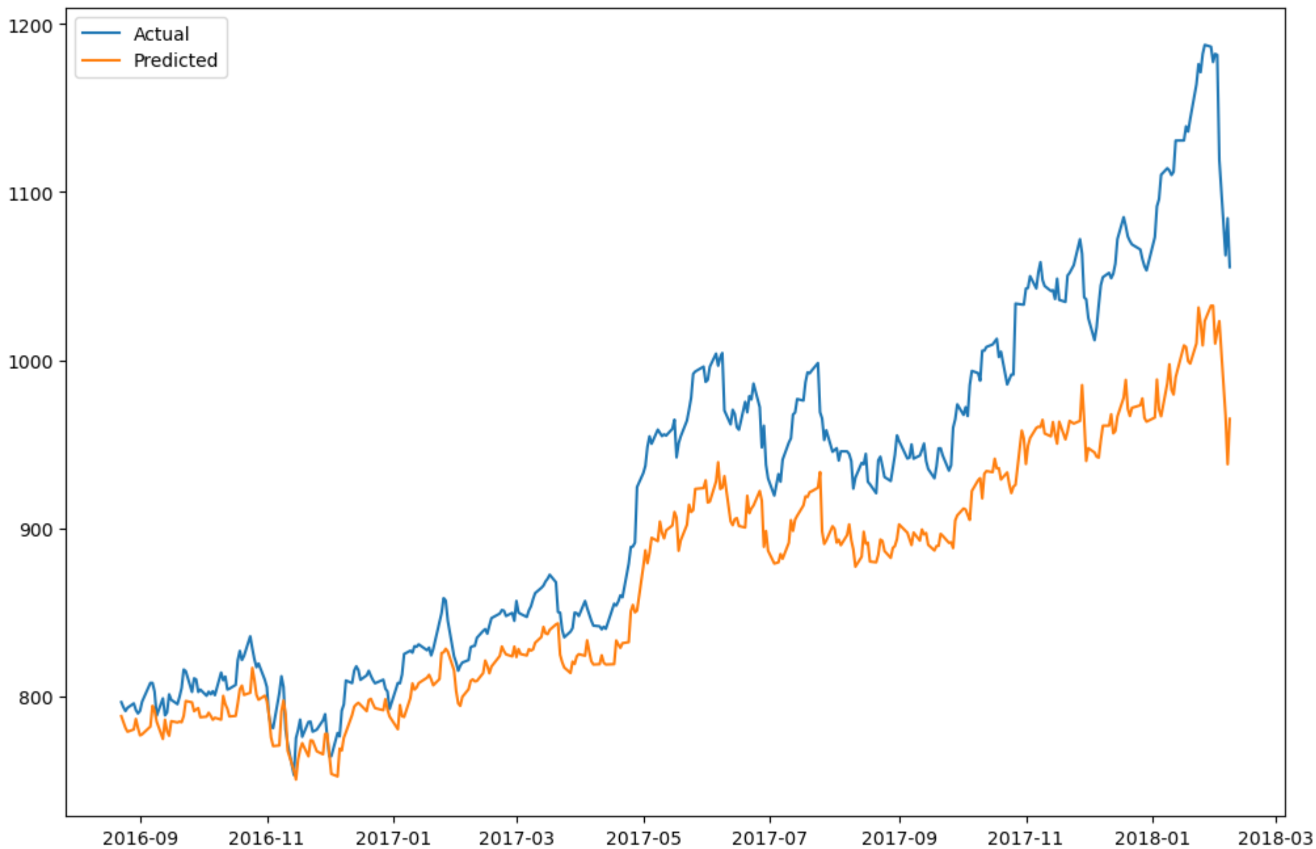
```
Epoch 44/300
104/104 - 2s - loss: 8.3761e-04 - mse: 8.3761e-04 - val_loss: 7.2644e-04 - val_mse: 7.2644e-04 - 2s/epoch - 23ms/step
Epoch 45/300
104/104 - 2s - loss: 8.0233e-04 - mse: 8.0233e-04 - val_loss: 0.0012 - val_mse: 0.0012 - 2s/epoch - 23ms/step
Epoch 46/300
104/104 - 2s - loss: 7.6365e-04 - mse: 7.6365e-04 - val_loss: 7.0396e-04 - val_mse: 7.0396e-04 - 2s/epoch - 23ms/step
Epoch 47/300
104/104 - 2s - loss: 8.4252e-04 - mse: 8.4252e-04 - val_loss: 9.0775e-04 - val_mse: 9.0775e-04 - 2s/epoch - 22ms/step
Epoch 48/300
104/104 - 2s - loss: 7.4311e-04 - mse: 7.4311e-04 - val_loss: 6.9127e-04 - val_mse: 6.9127e-04 - 2s/epoch - 23ms/step
Epoch 49/300
104/104 - 2s - loss: 7.7776e-04 - mse: 7.7776e-04 - val_loss: 0.0025 - val_mse: 0.0025 - 2s/epoch - 23ms/step
Epoch 50/300
104/104 - 2s - loss: 7.4193e-04 - mse: 7.4193e-04 - val_loss: 0.0012 - val_mse: 0.0012 - 2s/epoch - 23ms/step
```

In [47]:

```
predictions = model2.predict(X_test)
predictions = sc_target.inverse_transform(predictions)

plt.figure(figsize=(12, 8))
plt.plot(df.iloc[-test_size+window_size:,0], sc_target.inverse_transform(y_test), label='Actual')
plt.plot(df.iloc[-test_size+window_size:,0], predictions, label='Predicted')
plt.legend()
plt.show()
```

```
12/12 [==============================] - 3s 14ms/step
```



In [48]:

```
mse = model2.evaluate(X_test, y_test)
print("RMSE:", np.sqrt(sc_target.inverse_transform([[mse]])[0][0]))
```

```
12/12 [==============================] - 0s 10ms/step - loss: 0.0208 - mse: 0.0208
RMSE: 19.802622867702112
```

I tried a bunch of architectures with and without dropout layers. I noticed that if sometimes if I go up to three LSTM layers with a lot of neurons, the performance drops significantly.. I've repeated it multiple times with different number of neurons, two or three layers, adding/taking out the Dropout layers, tweaking the batch_size and I noticed thgat having one layer consistently behaves better! I'm not really sure why that is but I suspect that it's hard to train so many parameters and so having a small number of training data isn't really doing it justice. I think it might behave better and learn and optimize way better if we have a training set that is much larger.

In [ ]: