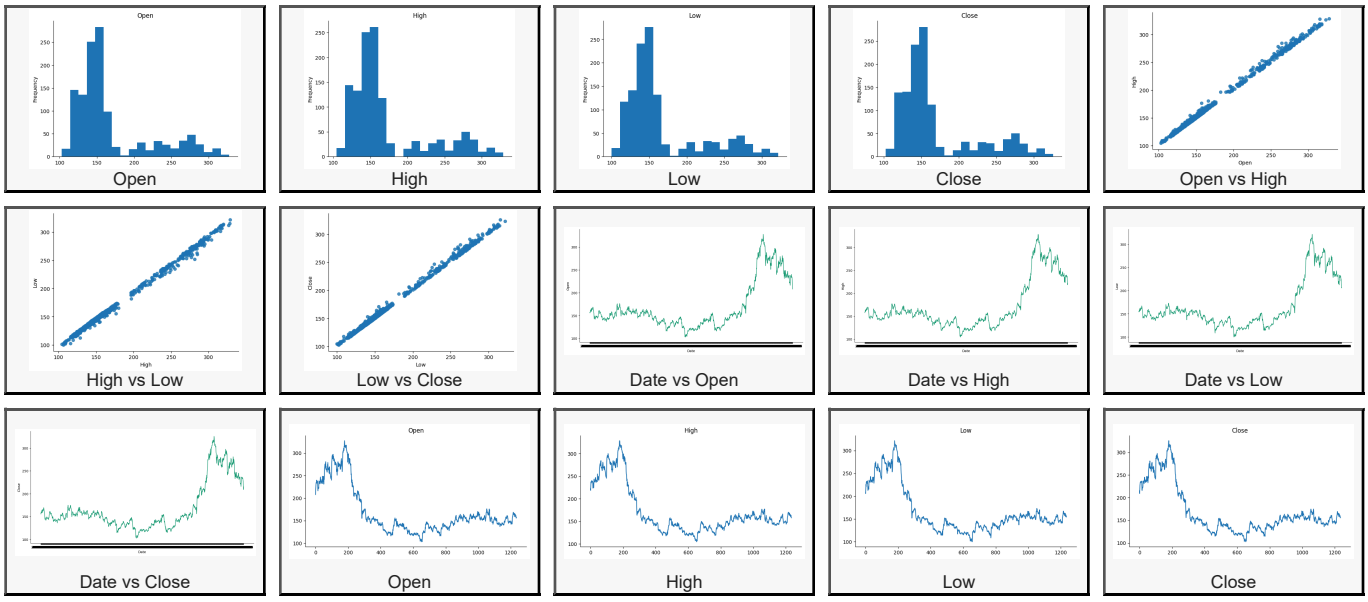
```
import pandas as pd
import datetime
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
from matplotlib import pyplot as plt
from sklearn import model_selection
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
data = pd.read_csv('NSE-Tata_Global_Beverages_Limited.csv')
dataset_train=data.iloc[0:930,1:2]
dataset_test=data.iloc[930:,1:2]
training_set = data.iloc[0:930, 1:2].values
testing_set=data.iloc[930:,1:2].values
data.head()
```

	Date	Open	High	Low	Last	Close	Total Trade Quantity	Turnover (Lacs)
0	2018-10-08	208.00	222.25	206.85	216.00	215.15	4642146.0	10062.83
1	2018-10-05	217.00	218.60	205.90	210.25	209.20	3519515.0	7407.06
2	2018-10-04	223.50	227.80	216.15	217.25	218.20	1728786.0	3815.79

Next steps: View recommended plots

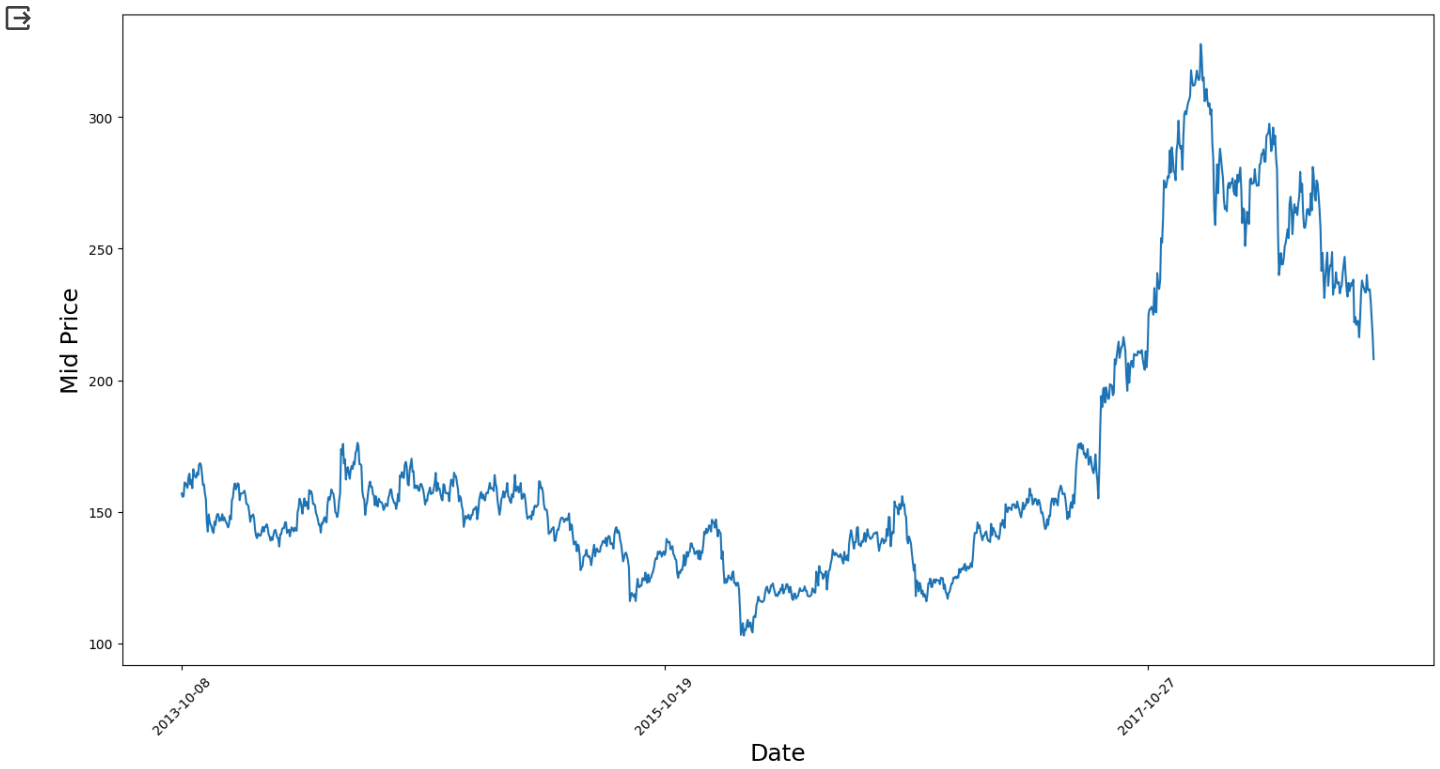



```
data.drop('Last', axis=1, inplace=True)
data.drop('Total Trade Quantity', axis=1, inplace=True)
data.drop('Turnover (Lacs)', axis=1, inplace=True)
print(data.head())
data.to_csv('tata_preprocessed.csv',index= False)
data = data.iloc[::-1]
```

```
         Date    Open    High     Low   Close
0  2018-10-08  208.00  222.25  206.85  215.15
1  2018-10-05  217.00  218.60  205.90  209.20
2  2018-10-04  223.50  227.80  216.15  218.20
3  2018-10-03  230.00  237.50  225.75  227.60
4  2018-10-01  234.55  234.60  221.05  230.90
```

```
import math
import pandas as pd
import numpy as np
from IPython.display import display
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import TimeSeriesSplit
```

```
plt.figure(figsize = (18,9))
plt.plot(range(data.shape[0]),(data['Open']))
plt.xticks(range(0,data.shape[0],500),data['Date'].loc[::500],rotation=45)
plt.xlabel('Date',fontsize=18)
plt.ylabel('Mid Price',fontsize=18)
plt.show()
```




```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
```

```
X_train = []
y_train = []
for i in range(10,930):
    X_train.append(training_set_scaled[i-10:i, 0])
    y_train.append(training_set_scaled[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))
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
regressor = Sequential()

regressor.add(LSTM(units = 75, return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.1))

regressor.add(LSTM(units = 75))
regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

regressor.fit(X_train, y_train, epochs = 100, batch_size = 64)
```

```
Epoch 1/100
15/15 [==============================] - 12s 34ms/step - loss: 0.0785
Epoch 2/100
15/15 [==============================] - 0s 31ms/step - loss: 0.0114
Epoch 3/100
15/15 [==============================] - 0s 31ms/step - loss: 0.0046
Epoch 4/100
15/15 [==============================] - 1s 35ms/step - loss: 0.0039
Epoch 5/100
15/15 [==============================] - 1s 55ms/step - loss: 0.0032
Epoch 6/100
15/15 [==============================] - 1s 52ms/step - loss: 0.0033
Epoch 7/100
15/15 [==============================] - 1s 54ms/step - loss: 0.0037
Epoch 8/100
15/15 [==============================] - 1s 34ms/step - loss: 0.0036
Epoch 9/100
15/15 [==============================] - 0s 32ms/step - loss: 0.0032
Epoch 10/100
15/15 [==============================] - 0s 33ms/step - loss: 0.0029
Epoch 11/100
15/15 [==============================] - 0s 32ms/step - loss: 0.0032
Epoch 12/100
15/15 [==============================] - 1s 34ms/step - loss: 0.0028
Epoch 13/100
15/15 [==============================] - 0s 33ms/step - loss: 0.0034
Epoch 14/100
15/15 [==============================] - 0s 33ms/step - loss: 0.0035
Epoch 15/100
15/15 [==============================] - 0s 32ms/step - loss: 0.0031
Epoch 16/100
15/15 [==============================] - 0s 33ms/step - loss: 0.0037
Epoch 17/100
15/15 [==============================] - 0s 31ms/step - loss: 0.0030
Epoch 18/100
15/15 [==============================] - 1s 33ms/step - loss: 0.0027
Epoch 19/100
15/15 [==============================] - 0s 32ms/step - loss: 0.0030
Epoch 20/100
15/15 [==============================] - 1s 34ms/step - loss: 0.0030
Epoch 21/100
15/15 [==============================] - 0s 33ms/step - loss: 0.0033
Epoch 22/100
15/15 [==============================] - 1s 33ms/step - loss: 0.0030
Epoch 23/100
15/15 [==============================] - 0s 33ms/step - loss: 0.0026
Epoch 24/100
15/15 [==============================] - 0s 32ms/step - loss: 0.0027
Epoch 25/100
15/15 [==============================] - 0s 32ms/step - loss: 0.0026
Epoch 26/100
15/15 [==============================] - 1s 36ms/step - loss: 0.0024
Epoch 27/100
15/15 [==============================] - 0s 31ms/step - loss: 0.0028
Epoch 28/100
15/15 [==============================] - 1s 54ms/step - loss: 0.0029
```

```
Epoch 29/100
15/15 [==============================] - 1s 53ms/step - loss: 0.0027
```

```
real_stock_price = testing_set
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 10:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(10,305):
    X_test.append(inputs[i-10:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

```
10/10 [==============================] - 2s 8ms/step
```

```
plt.plot(real_stock_price, color = 'black', label = 'TATA Stock Price')
plt.plot(predicted_stock_price, color = 'green', label = 'Predicted TATA Stock Price')
plt.title('TATA Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('TATA Stock Price')
plt.legend()
plt.show()
```