Problem Statement: Bitcoin is a price volatile, which indicates the need for studying the underlying price model. Here the data between the year 2014 - 2022. It proposed this as Long short-term memory problem. Here an attempt is made to predict increase/decrease in price. It is performed to predict increase/decrease price for next 30 days. In this case we have attempted to predict next day using Machine learning and deep learning algorithms.

Steps involved:

1. Importing Libraries
2. Loading datasets
3. EDA
4. Building model
5. Prediction
6. Evaluation
7. Conclusion

```
#Importing necessary libraries
import os
#os - functions for creating and removing a
# directory (folder), fetching its contents, changing and identifying the current director
import pandas as pd
import numpy as np
import math
import datetime as dt
import matplotlib.pyplot as plt

# For Evalution we will use these library

from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_sc
from sklearn.metrics import mean_poisson_deviance, mean_gamma_deviance, accuracy_score
from sklearn.preprocessing import MinMaxScaler

# For model building we will use these library

import tensorflow as tf
from tensorflow.keras.models import Sequential
#A Sequential model is appropriate for a plain stack
# of layers where each layer has exactly one input tensor and one output tensor.
from tensorflow.keras.layers import Dense, Dropout
#Dense layer is the regular deeply connected neural network layer. It is most common and f
# operation on the input and return the output. output = activation(dot(input, kernel) + b
from tensorflow.keras.layers import LSTM
# Long Short-Term Memory Network or LSTM, is a variation of a recurrent neural network (RN
# in predicting the long sequences of data like sentences and stock prices over a period o

# For PLotting we will use these library

import matplotlib.pyplot as plt
```

```
from itertools import cycle
import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots
```

```
maindf=pd.read_csv('BTC-USD.csv')
print('Total number of days present in the dataset: ',maindf.shape[0])
print('Total number of fields present in the dataset: ',maindf.shape[1])
```

```
Total number of days present in the dataset:  2713
Total number of fields present in the dataset:  7
```

```
maindf.shape
```

```
(2713, 7)
```

```
maindf.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	465.864014	468.174011	452.421997	457.334015	457.334015	21056800
1	2014-09-18	456.859985	456.859985	413.104004	424.440002	424.440002	34483200
2	2014-09-19	424.102997	427.834991	384.532013	394.795990	394.795990	37919700
3	2014-09-20	394.673004	423.295990	389.882996	408.903992	408.903992	36863600
4	2014-09-21	408.084991	412.425995	393.181000	398.821014	398.821014	26580100

```
maindf.tail()
```

	Date	Open	High	Low	Close	Adj Close	
2708	2022-02-15	42586.464844	44667.218750	42491.035156	44575.203125	44575.203125	227
2709	2022-02-16	44578.277344	44578.277344	43456.691406	43961.859375	43961.859375	197
2710	2022-02-17	43937.070313	44132.972656	40249.371094	40538.011719	40538.011719	262

```
maindf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2713 entries, 0 to 2712
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       2713 non-null   object 
 1   Open       2713 non-null   float64
 2   High       2713 non-null   float64
 3   Low        2713 non-null   float64
```

```
 4   Close      2713 non-null   float64
 5   Adj Close  2713 non-null   float64
 6   Volume     2713 non-null   int64
dtypes: float64(5), int64(1), object(1)
memory usage: 148.5+ KB
```

```
maindf.describe()
```

	Open	High	Low	Close	Adj Close	Vol
count	2713.000000	2713.000000	2713.000000	2713.000000	2713.000000	2.713000e
mean	11311.041069	11614.292482	10975.555057	11323.914637	11323.914637	1.470462e
std	16106.428891	16537.390649	15608.572560	16110.365010	16110.365010	2.001627e
min	176.897003	211.731003	171.509995	178.102997	178.102997	5.914570e
25%	606.396973	609.260986	604.109985	606.718994	606.718994	7.991080e
50%	6301.569824	6434.617676	6214.220215	6317.609863	6317.609863	5.098183e
75%	10452.399414	10762.644531	10202.387695	10462.259766	10462.259766	2.456992e
max	67549.734375	68789.625000	66382.062500	67566.828125	67566.828125	3.509679e

```
#checking for null values
print('NULL VALUES:',maindf.isnull().values.sum())
```

```
NULL VALUES: 0
```

```
print('NA values:',maindf.isnull().values.any())
```

```
NA values: False
```

```
maindf.shape
```

```
(2713, 7)
```

Exploratory Data Analysis

```
# Printing the start date and End date of the dataset
sd=maindf.iloc[0][0]
ed=maindf.iloc[-1][0]
print('Starting Date',sd)
print('Ending Date',ed)
```

```
Starting Date 2014-09-17
Ending Date 2022-02-19
```

```
#stockprice analysis from start
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')
```

```
y_2014 = maindf.loc[(maindf['Date'] >= '2014-09-17')& (maindf['Date'] < '2014-12-31')]
y_2014.drop(y_2014[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
0	2014-09-17	465.864014	468.174011	452.421997	457.334015
1	2014-09-18	456.859985	456.859985	413.104004	424.440002
2	2014-09-19	424.102997	427.834991	384.532013	394.795990
3	2014-09-20	394.673004	423.295990	389.882996	408.903992
4	2014-09-21	408.084991	412.425995	393.181000	398.821014
...	...	...	...	...	...
100	2014-12-26	319.152008	331.424011	316.627014	327.924011
101	2014-12-27	327.583008	328.911011	312.630005	315.863007
102	2014-12-28	316.160004	320.028015	311.078003	317.239014
103	2014-12-29	317.700989	320.266998	312.307007	312.670013
104	2014-12-30	312.718994	314.808990	309.372986	310.737000

105 rows × 5 columns

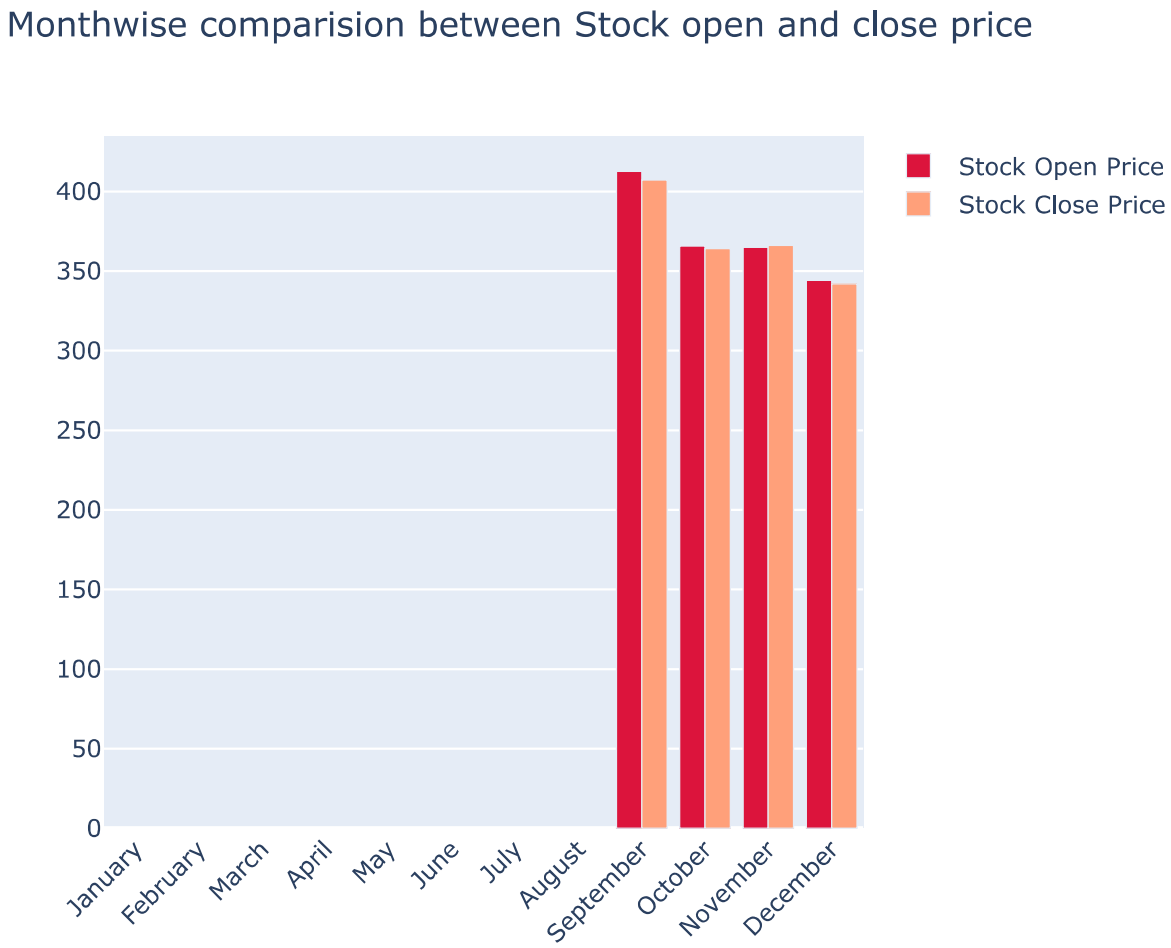
```
y_2014.shape
```

```
(105, 7)
```

```
monthvise= y_2014.groupby(y_2014['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August','Sep
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
#most of the values herer are NaN
```

Date	Open	Close
January	NaN	NaN

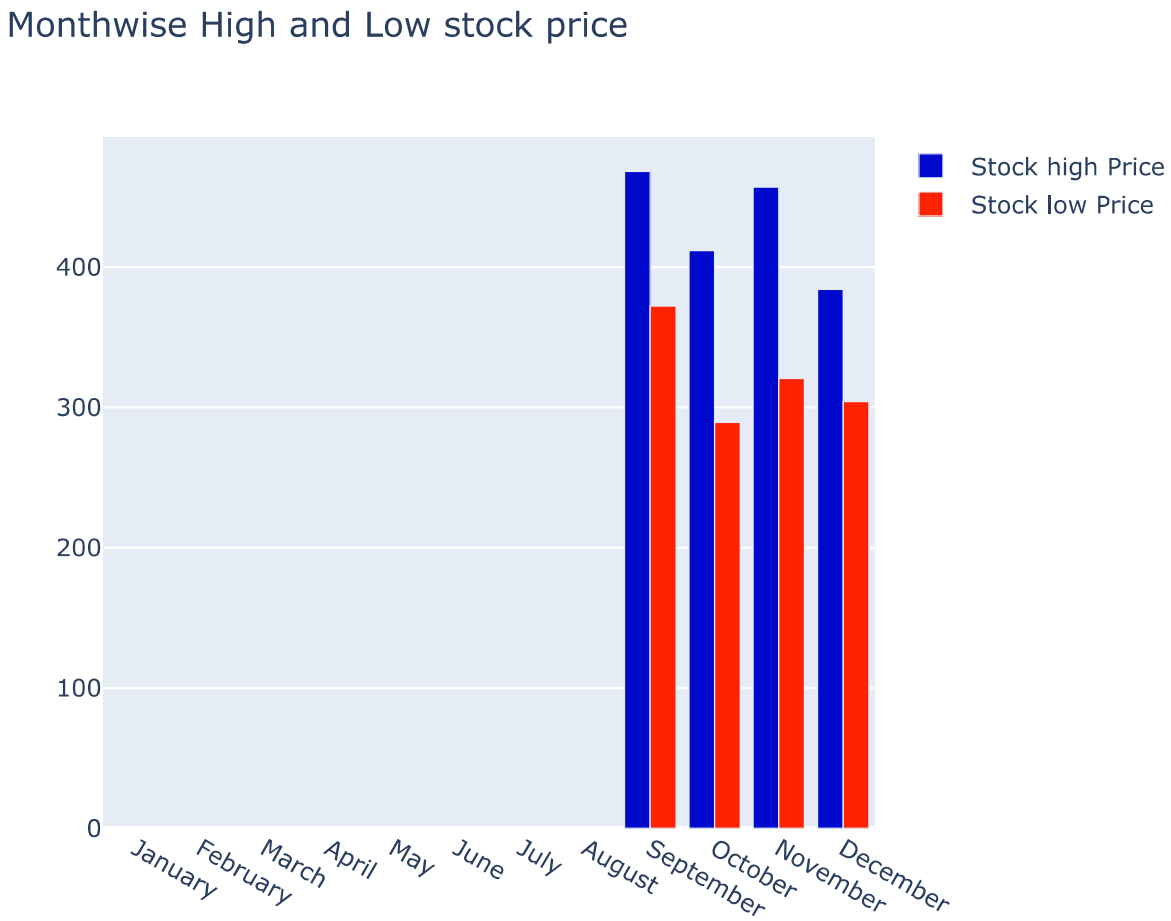
```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))
fig.update_layout(barmode='group', xaxis_tickangle=-45, title='Monthwise comparision betwe
fig.show()
```



only few points are plotted in the above graph as rest of the data has no values in it

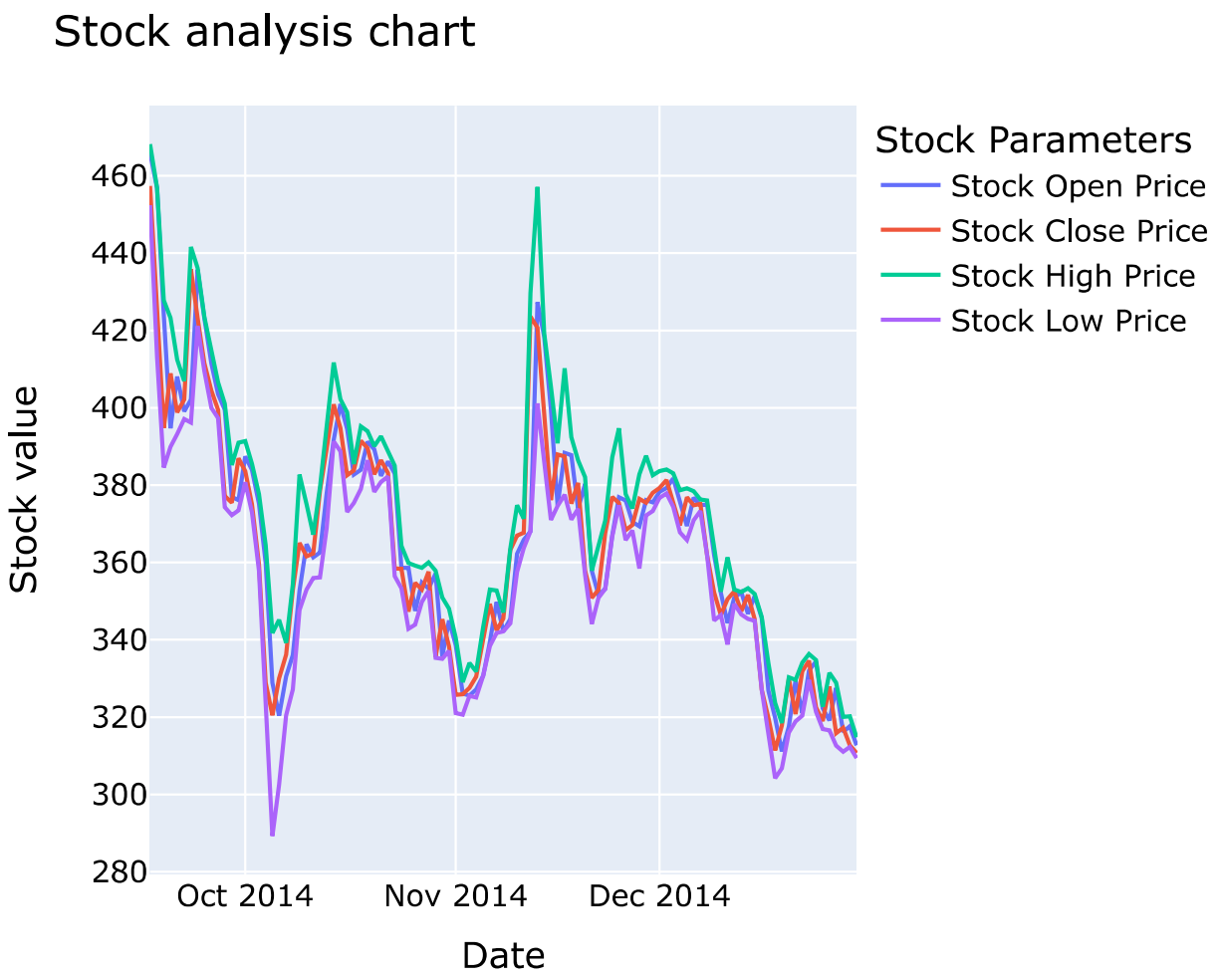
```
y_2014.groupby(y_2014['Date'].dt.strftime('%B'))['Low'].min()
```

```
monthvise_high = y_2014.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2014.groupby(y_2014['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 10, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 34, 0)'
))

fig.update_layout(barmode='group', title=' Monthwise High and Low stock price')
fig.show()
```




```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'
fig = px.line(y_2014, x=y_2014.Date, y=[y_2014['Open'], y_2014['Close'], y_2014['High'], y
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
```

```
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=True)
fig.update_yaxes(showgrid=True)
fig.show()
```



Analysis of Year - 2015

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')
y_2015 = maindf.loc[(maindf['Date'] >= '2015-01-01') & (maindf['Date'] < '2016-01-01')]
y_2015.drop(y_2015[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
106	2015-01-01	320.434998	320.434998	314.002991	314.248993
107	2015-01-02	314.079010	315.838989	313.565002	315.032013
108	2015-01-03	314.846008	315.149994	281.082001	281.082001
109	2015-01-04	281.145996	287.230011	257.612000	264.195007

```
y_2015.shape
```

```
(365, 7)
```

```
monthvise= y_2015.groupby(y_2015['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'Se
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

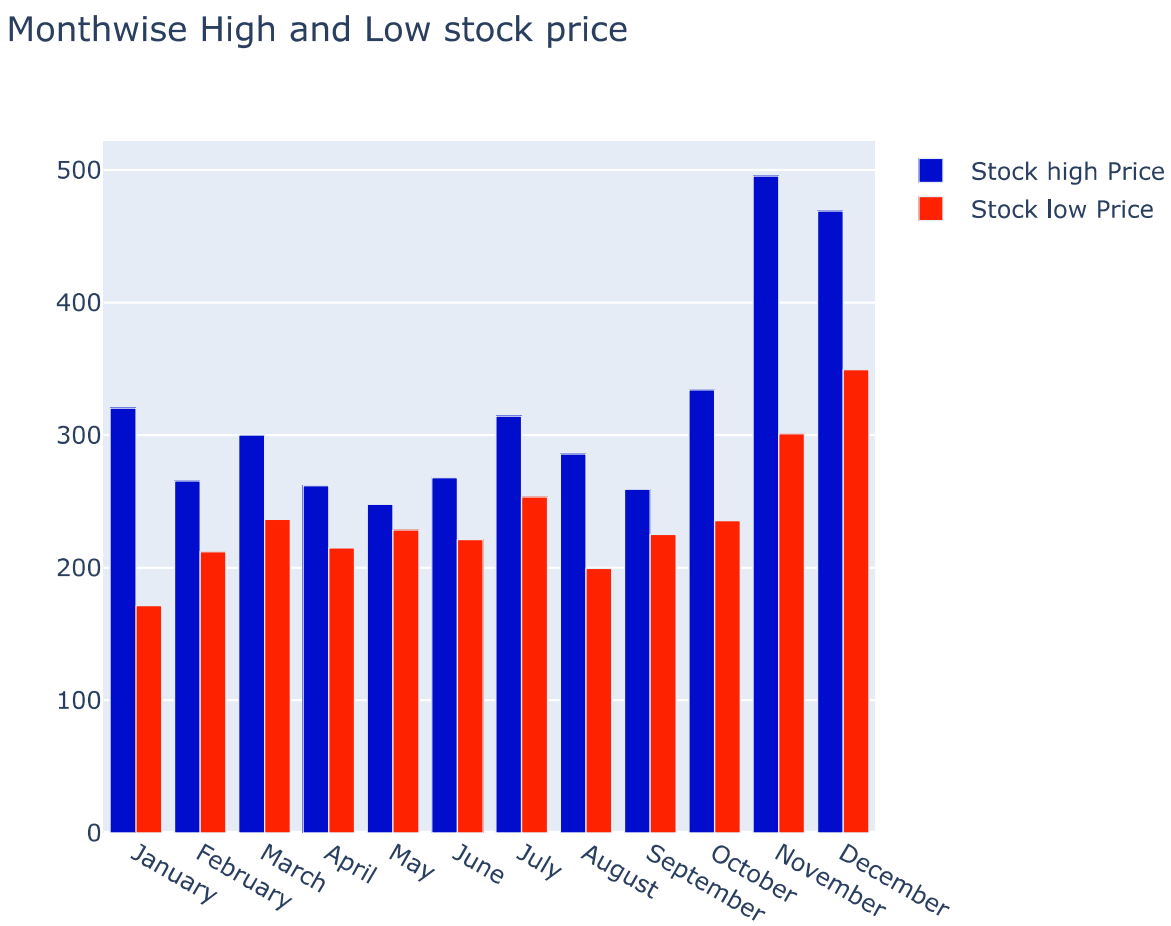
Date	Open	Close
January	251.799905	248.782547
February	232.821856	234.153645
March	269.278419	269.042259
April	235.708667	235.491534
May	237.161806	236.997001
June	236.941433	238.081766
July	278.857679	279.563740
August	252.986774	250.733805
September	233.486733	233.595533
October	262.306000	264.855356
November	346.866833	348.883332
December	422.618033	424.464547

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
```

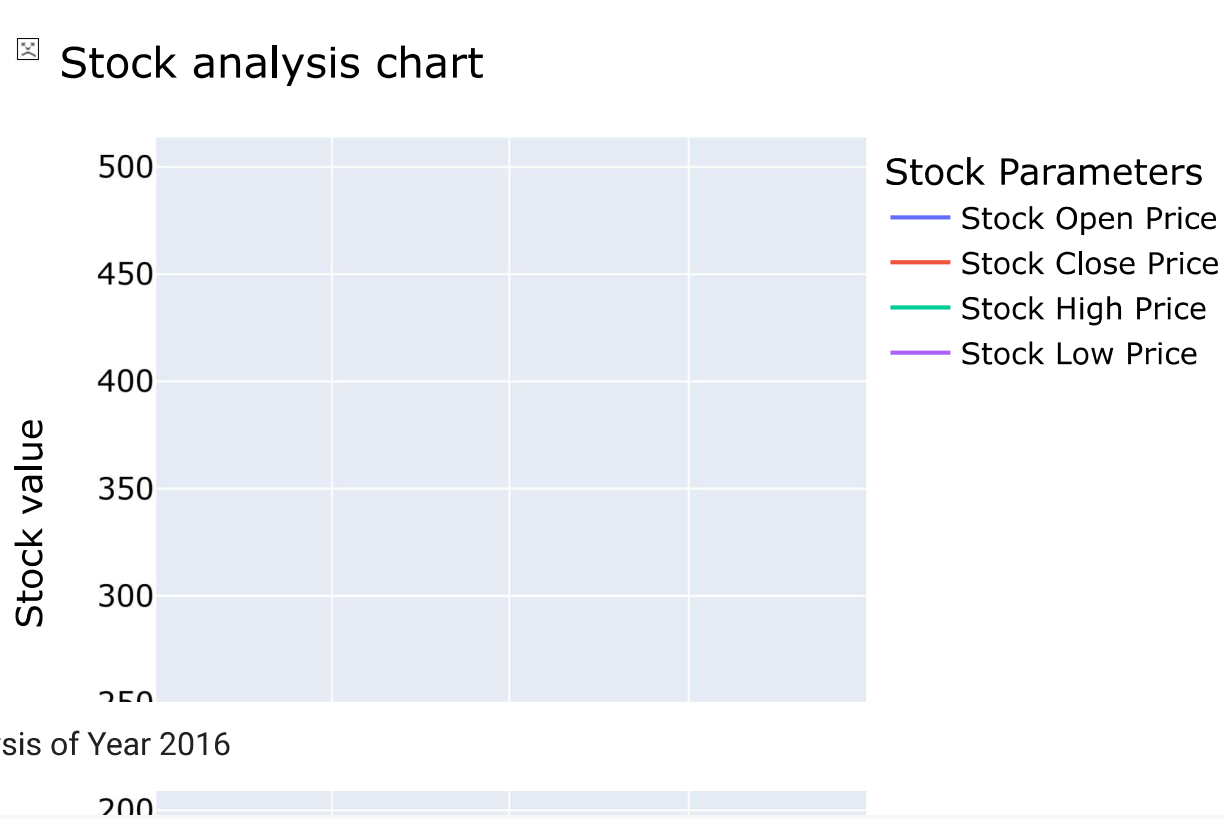
```
    marker_color='lightsalmon'
))
fig.update_layout(barmode='group', xaxis_tickangle=-45, title='Monthwise comparision betwe
fig.show()
```


```
y_2015.groupby(y_2015['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2015.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2015.groupby(y_2015['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 13, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 34, 0)'
))
```

```
fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```




```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'
fig = px.line(y_2015, x=y_2015.Date, y=[y_2015['Open'], y_2015['Close'],
                                        y_2015['High'], y_2015['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=True)
fig.update_yaxes(showgrid=True)
fig.show()
```




```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2016 = maindf.loc[(maindf['Date'] >= '2016-01-01')
                    & (maindf['Date'] < '2017-01-01')]

y_2016.drop(y_2016[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
471	2016-01-01	430.721008	436.246002	427.515015	434.334015
472	2016-01-02	434.622009	436.062012	431.869995	433.437988
473	2016-01-03	433.578003	433.743011	424.705994	430.010986
474	2016-01-04	430.061005	434.516998	429.084015	433.091003
475	2016-01-05	433.069000	434.182007	429.675995	431.959991
...	...	...	...	...	...
832	2016-12-27	908.354004	940.047974	904.255005	933.197998
833	2016-12-28	934.830994	975.921021	934.830994	975.921021
834	2016-12-29	975.125000	979.396973	954.502991	973.497009
835	2016-12-30	972.534973	972.534973	934.833008	961.237976
836	2016-12-31	960.627014	963.742981	947.236023	963.742981

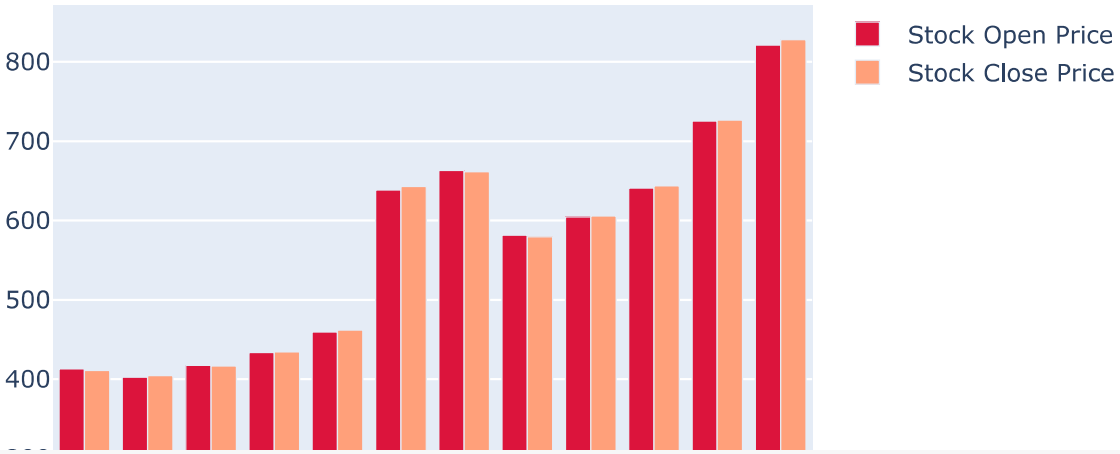
366 rows × 5 columns

```
monthvise= y_2016.groupby(y_2016['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'Se
```

```
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

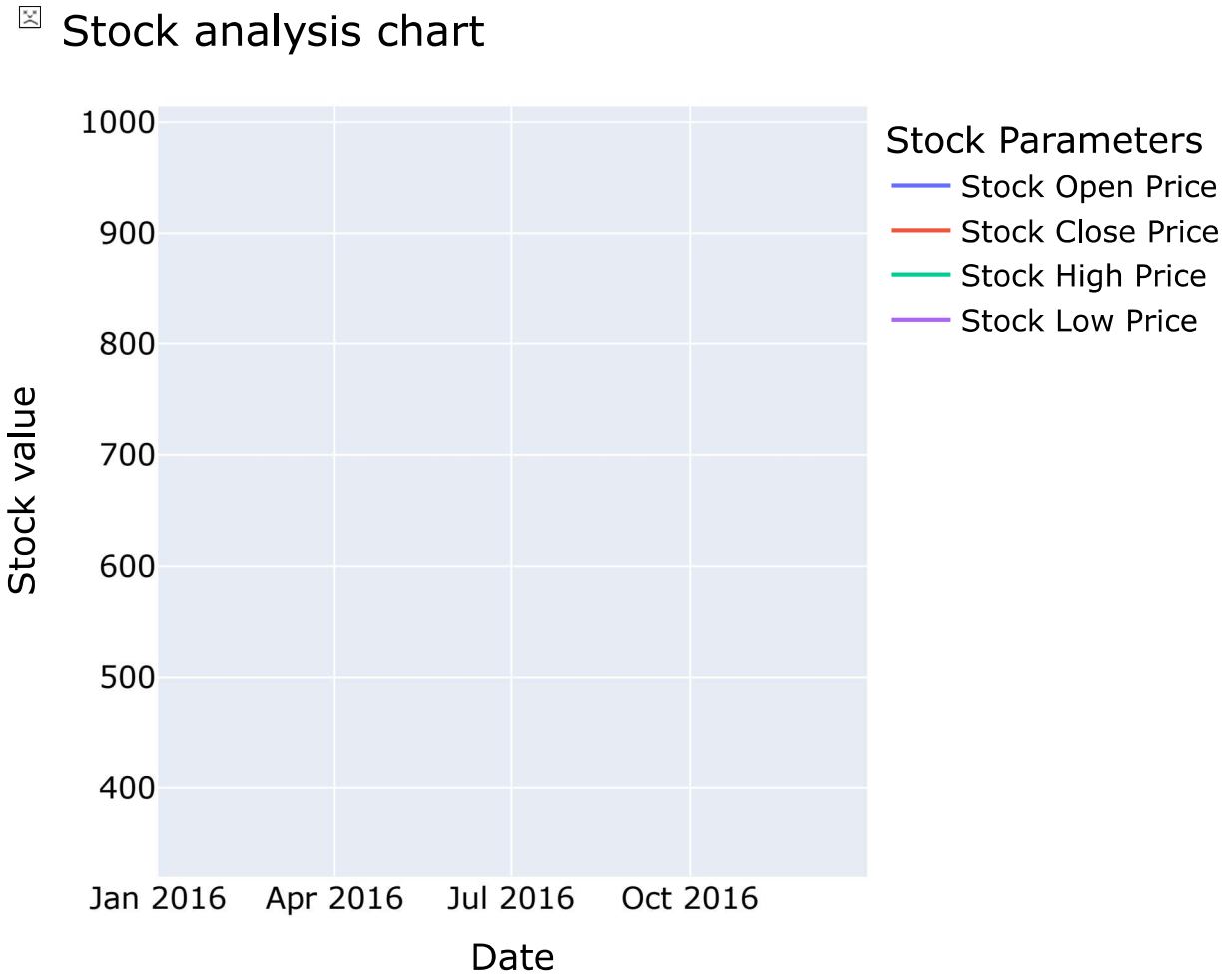
Date	Open	Close
January	412.805902	410.844485
February	402.304692	404.408274
March	417.262033	416.525774
April	433.487433	434.339398
May	459.237547	461.954415
June	638.544834	642.869061
July	662.977779	661.356103
August	581.238966	579.585197
September	604.614034	605.848633
October	640.702546	643.550935
November	725.073804	726.349101
December	821.108255	828.060356

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))
fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```




```
y_2016.groupby(y_2016['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2016.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2016.groupby(y_2016['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 143, 205)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(235, 108, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```


```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'
fig = px.line(y_2016, x=y_2016.Date, y=[y_2016['Open'], y_2016['Close'],  y_2016['High'],
      labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=True)
fig.update_yaxes(showgrid=True)
fig.show()
```



Analysis of Year 2017

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2017 = maindf.loc[(maindf['Date'] >= '2017-01-01')
                    & (maindf['Date'] < '2018-01-01')]

y_2017.drop(y_2017[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
837	2017-01-01	963.658020	1003.080017	958.698975	998.325012
838	2017-01-02	998.617004	1031.390015	996.702026	1021.750000
839	2017-01-03	1021.599976	1044.079956	1021.599976	1043.839966
840	2017-01-04	1044.400024	1159.420044	1044.400024	1154.729980
841	2017-01-05	1156.729980	1191.099976	910.416992	1013.380005
...	...	...	...	...	...
1197	2017-12-27	16163.500000	16930.900391	15114.299805	15838.500000
1198	2017-12-28	15864.099609	15888.400391	13937.299805	14606.500000
1199	2017-12-29	14695.799805	15279.000000	14307.000000	14656.200195
1200	2017-12-30	14681.900391	14681.900391	12350.099609	12952.200195
1201	2017-12-31	12897.700195	14377.400391	12755.599609	14156.400391

365 rows × 5 columns

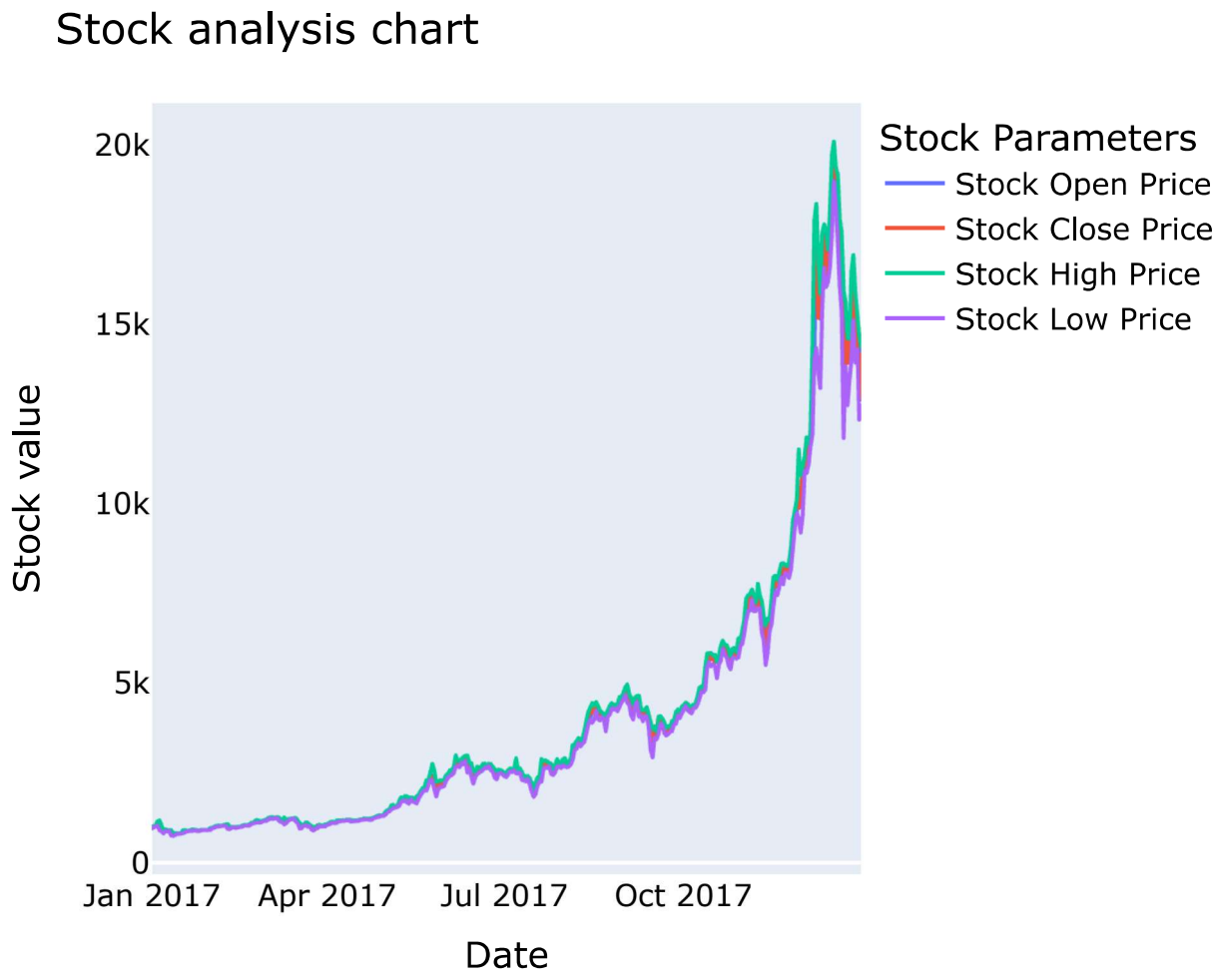
```
monthvise= y_2017.groupby(y_2017['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Date	Open	Close
January	914.680971	914.916159
February	1055.620071	1062.533672
March	1133.212576	1129.365228

```
fig = go.Figure()

fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```

```
y_2017.groupby(y_2017['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2017.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2017.groupby(y_2017['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```

```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'

fig = px.line(y_2017, x=y_2017.Date, y=[y_2017['Open'], y_2017['Close'],
                                        y_2017['High'], y_2017['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```



Analysis of Year 2018

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2018 = maindf.loc[(maindf['Date'] >= '2018-01-01')
                    & (maindf['Date'] < '2019-01-01')]

y_2018.drop(y_2018[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
1202	2018-01-01	14112.200195	14112.200195	13154.700195	13657.200195
1203	2018-01-02	13625.000000	15444.599609	13163.599609	14982.099609
1204	2018-01-03	14978.200195	15572.799805	14844.500000	15201.000000
1205	2018-01-04	15270.700195	15739.700195	14522.200195	15599.200195
1206	2018-01-05	15477.200195	17705.199219	15202.799805	17429.500000
...	...	...	...	...	...
1562	2018-12-27	3854.688477	3874.416992	3645.448486	3654.833496
1563	2018-12-28	3653.131836	3956.135986	3642.632080	3923.918701
1564	2018-12-29	3932.491699	3963.758789	3820.408691	3820.408691
1565	2018-12-30	3822.384766	3901.908936	3797.219238	3865.952637
1566	2018-12-31	3866.839111	3868.742920	3725.867432	3742.700439

365 rows x 5 columns

```
monthvise= y_2018.groupby(y_2018['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

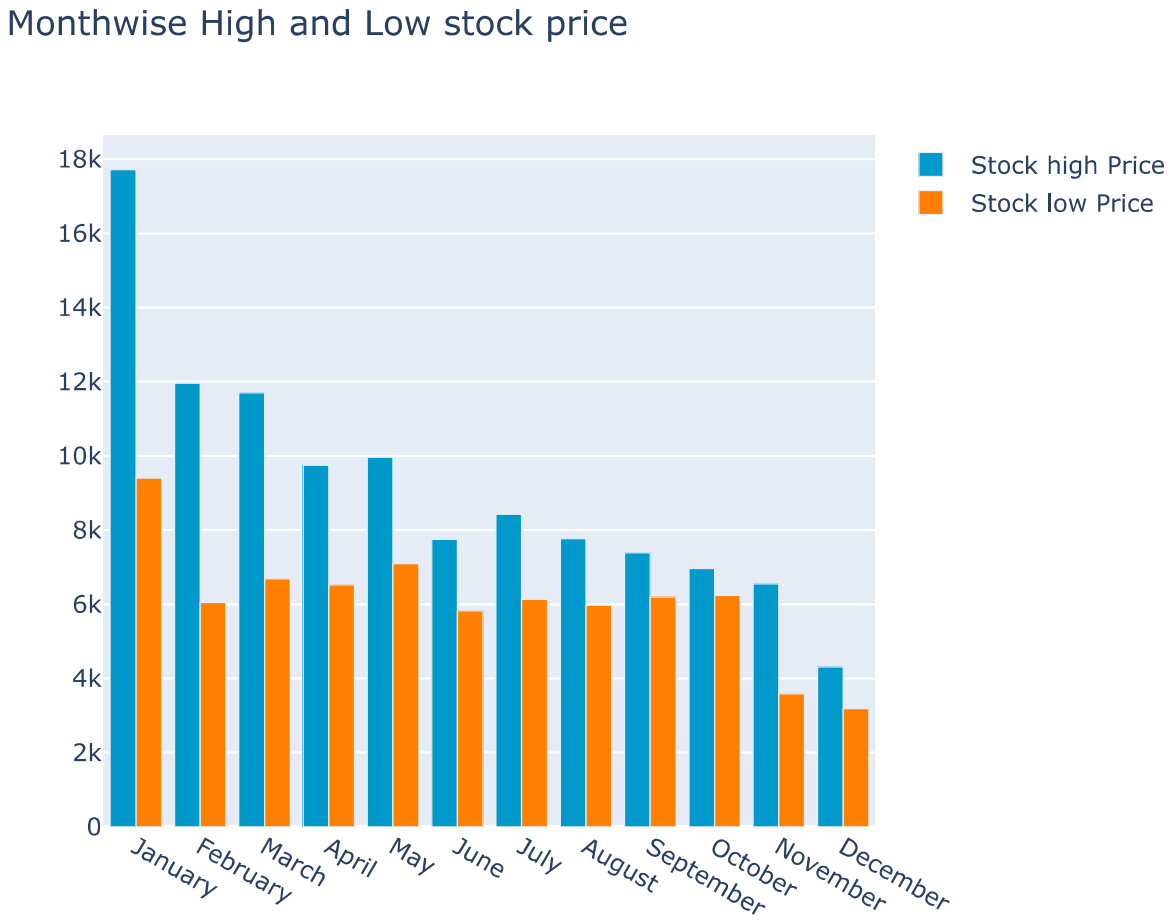
Date	Open	Close
January	13212.074219	13085.558090
February	9462.242920	9472.001151
March	9156.591718	9040.557097
April	7963.618311	8033.596631
May	8505.240675	8450.997732
June	6829.257975	6793.507666
July	7101.466450	7146.349987
August	6723.800955	6700.129946
September	6622.821338	6610.675033
October	6494.016491	6485.118747
November	5481.615120	5404.250171
December	3726.475106	3717.488344

```
fig = go.Figure()
```

```
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```

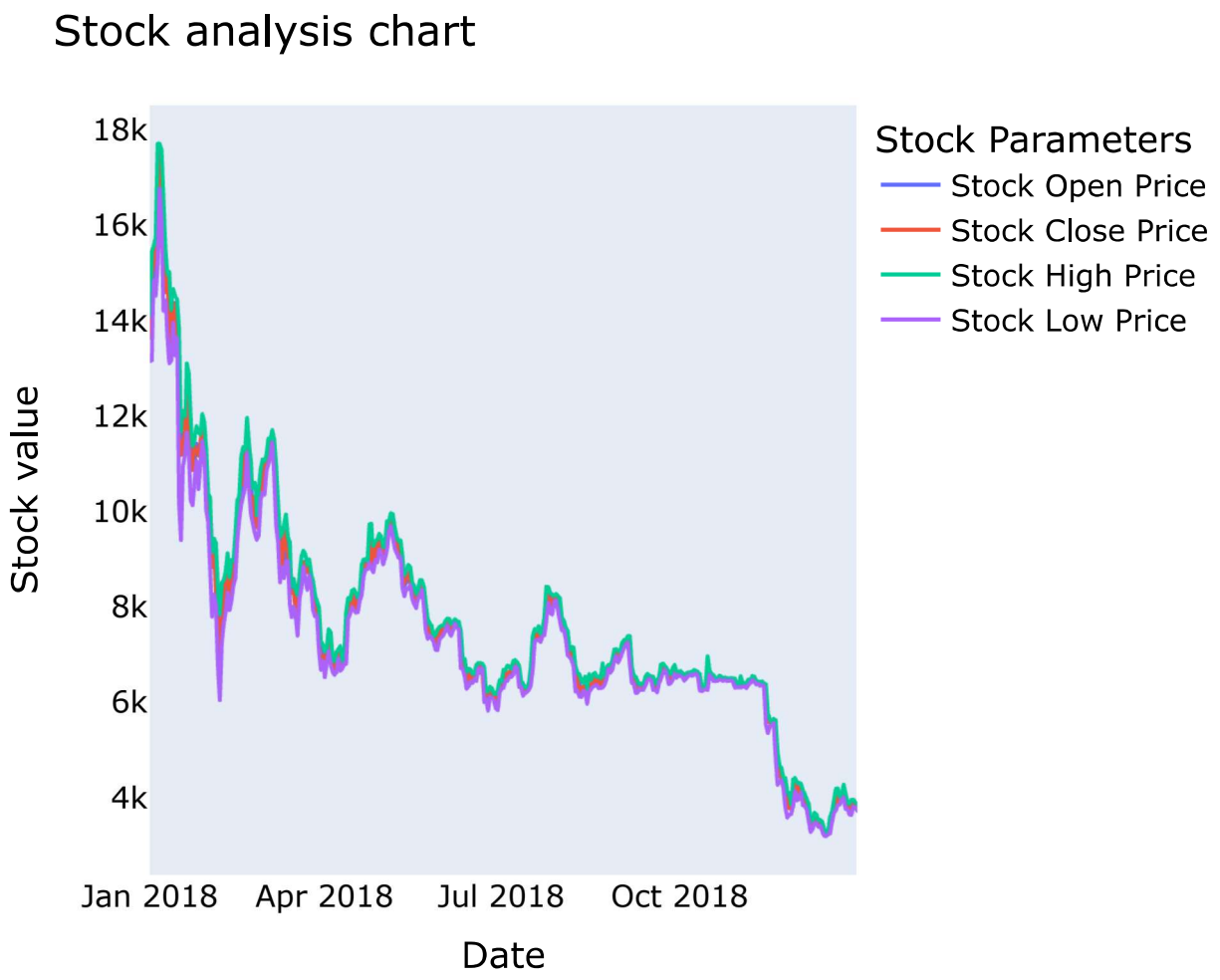

```
y_2018.groupby(y_2018['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2018.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2018.groupby(y_2018['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
```

```
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```




```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'

fig = px.line(y_2018, x=y_2018.Date, y=[y_2018['Open'], y_2018['Close'],
                                        y_2018['High'], y_2018['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
```

```
fig.show()
```



Analysis of Year 2019

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2019 = maindf.loc[(maindf['Date'] >= '2019-01-01')
                    & (maindf['Date'] < '2020-01-01')]

y_2019.drop(y_2019[['Adj Close','Volume']],axis=1)
```

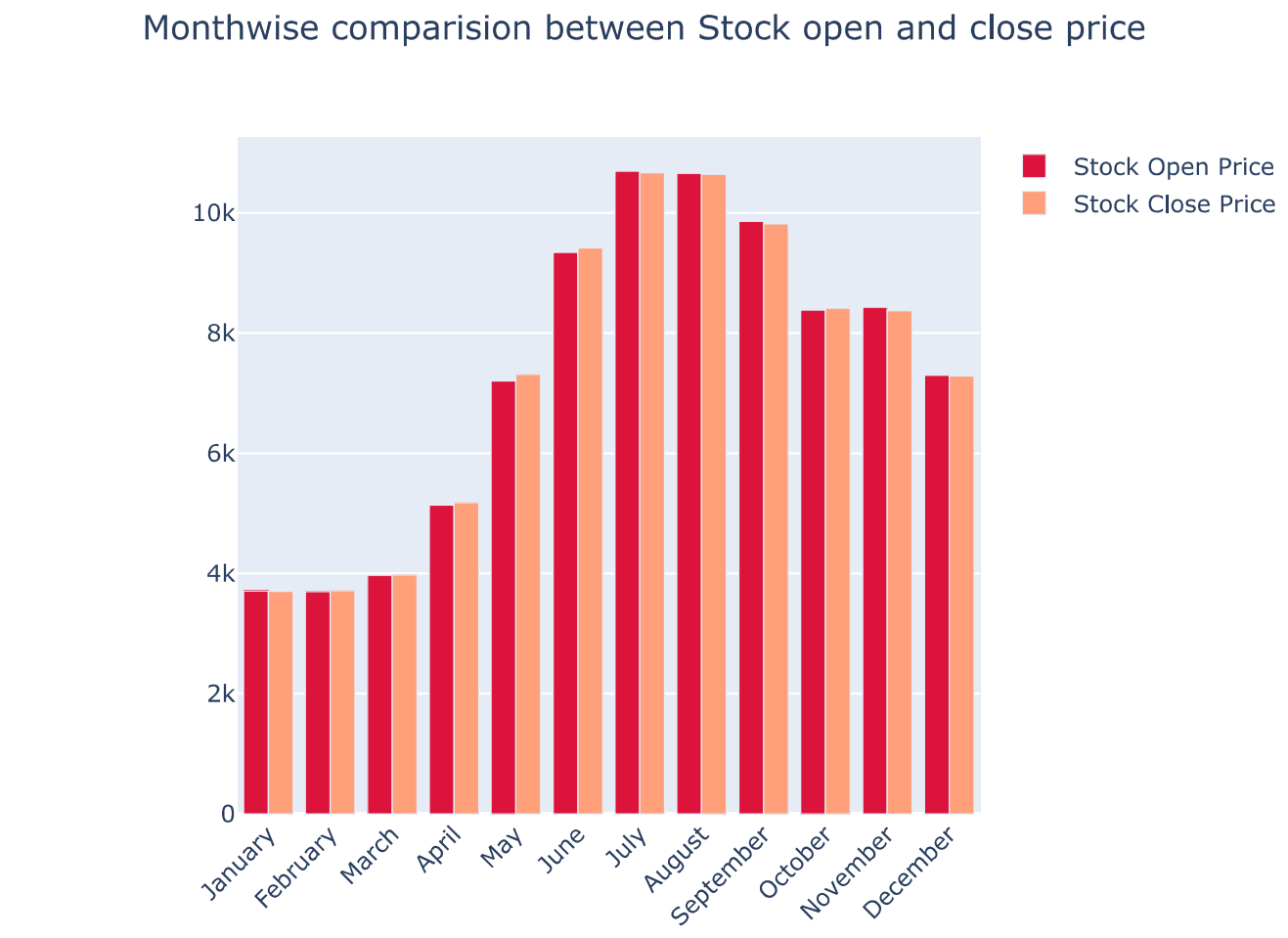
	Date	Open	High	Low	Close
1567	2019-01-01	3746.713379	3850.913818	3707.231201	3843.520020
1568	2019-01-02	3849.216309	3947.981201	3817.409424	3943.409424
1569	2019-01-03	3931.048584	3935.685059	3826.222900	3836.741211
1570	2019-01-04	3832.040039	3865.934570	3783.853760	3857.717529
1571	2019-01-05	3851.973877	3904.903076	3836.900146	3845.194580

```
monthvise= y_2019.groupby(y_2019['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Date	Open	Close
January	3709.705645	3701.554963
February	3697.178327	3711.907261
March	3967.740400	3976.069100
April	5136.813314	5178.469434
May	7205.208024	7309.694131
June	9339.480322	9415.900179
July	10691.706055	10669.336158
August	10657.745621	10643.248362
September	9858.141813	9814.067871
October	8382.432129	8411.929168
November	8427.103516	8373.572412
December	7296.351625	7284.013042

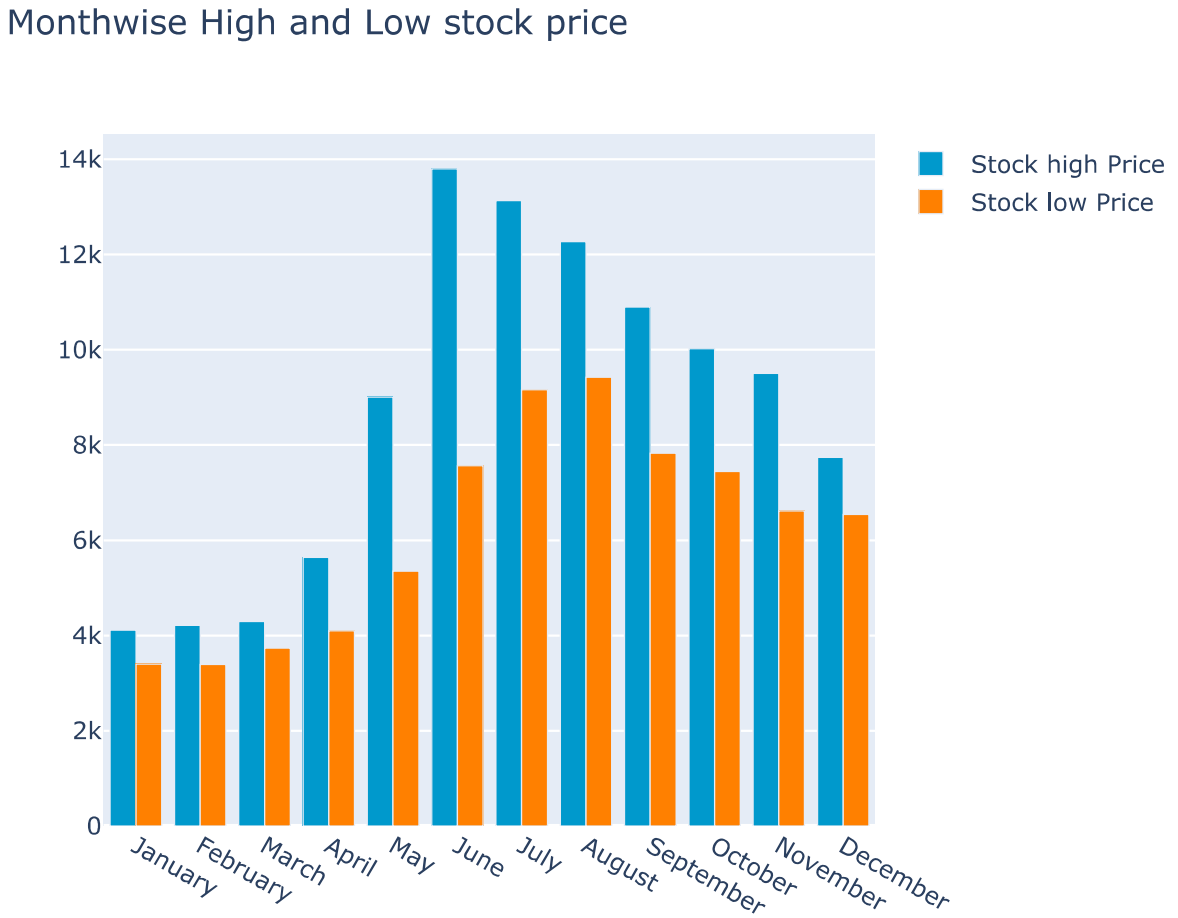
```
fig = go.Figure()

fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
```

```
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```

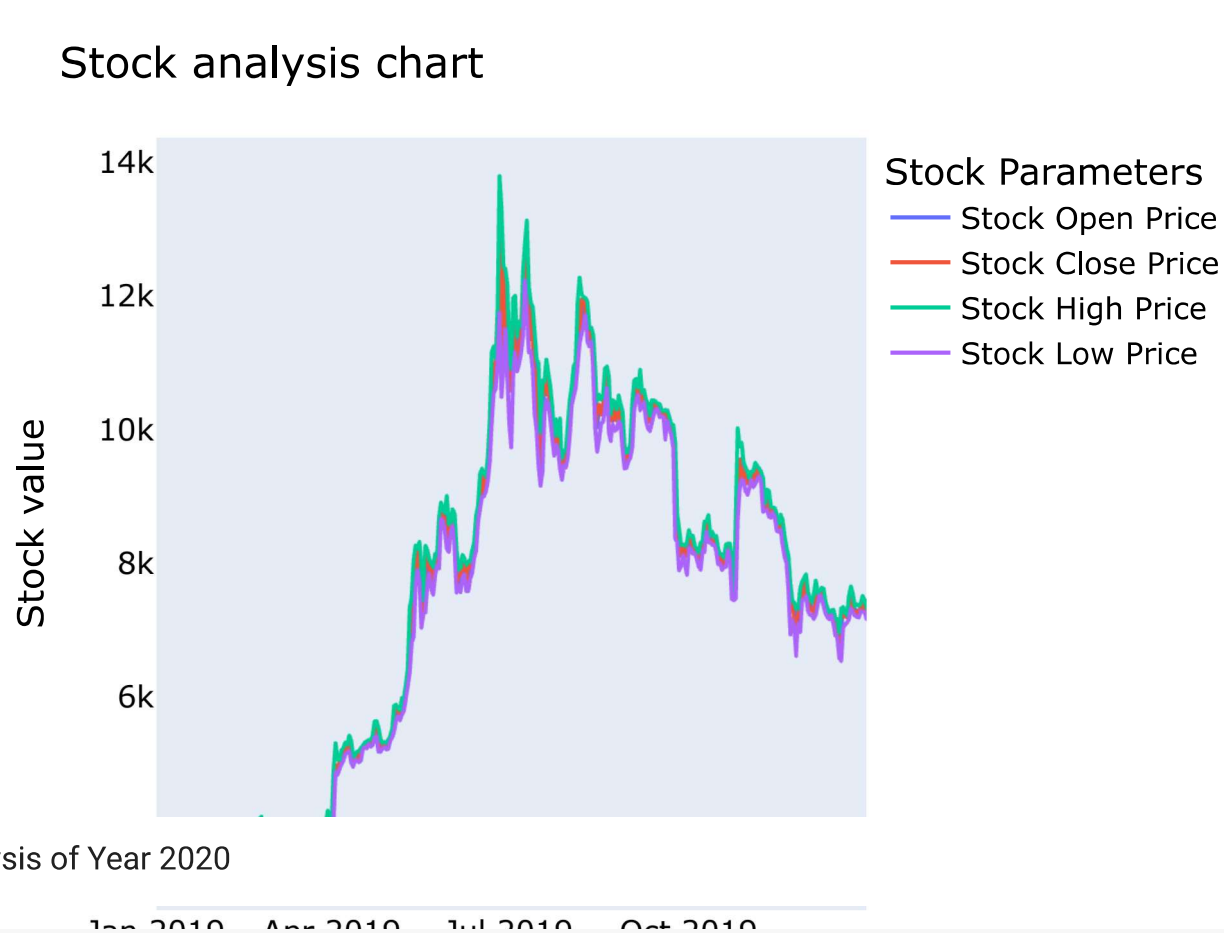



```
y_2019.groupby(y_2019['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2019.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2019.groupby(y_2019['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
```

```
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```




```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'

fig = px.line(y_2019, x=y_2019.Date, y=[y_2019['Open'], y_2019['Close'],
                                        y_2019['High'], y_2019['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```




```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2020 = maindf.loc[(maindf['Date'] >= '2020-01-01')
                    & (maindf['Date'] < '2021-01-01')]

y_2020.drop(y_2020[['Adj Close','Volume']],axis=1)
```

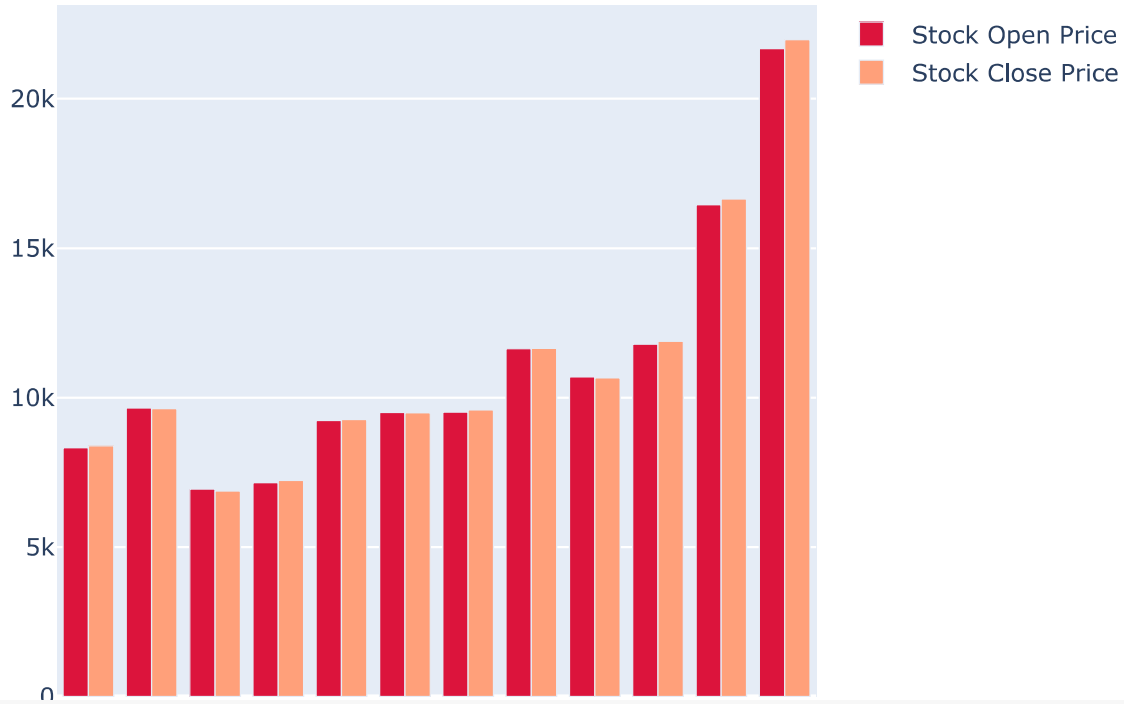
	Date	Open	High	Low	Close
1932	2020-01-01	7194.892090	7254.330566	7174.944336	7200.174316
1933	2020-01-02	7202.551270	7212.155273	6935.270020	6985.470215
1934	2020-01-03	6984.428711	7413.715332	6914.996094	7344.884277
1935	2020-01-04	7345.375488	7427.385742	7309.514160	7410.656738
1936	2020-01-05	7410.451660	7544.497070	7400.535645	7411.317383
...	...	...	...	...	...
2293	2020-12-27	26439.373047	28288.839844	25922.769531	26272.294922
2294	2020-12-28	26280.822266	27389.111328	26207.640625	27084.808594
2295	2020-12-29	27081.810547	27370.720703	25987.298828	27362.437500
2296	2020-12-30	27360.089844	28937.740234	27360.089844	28840.953125
2297	2020-12-31	28841.574219	29244.876953	28201.992188	29001.720703

366 rows × 5 columns

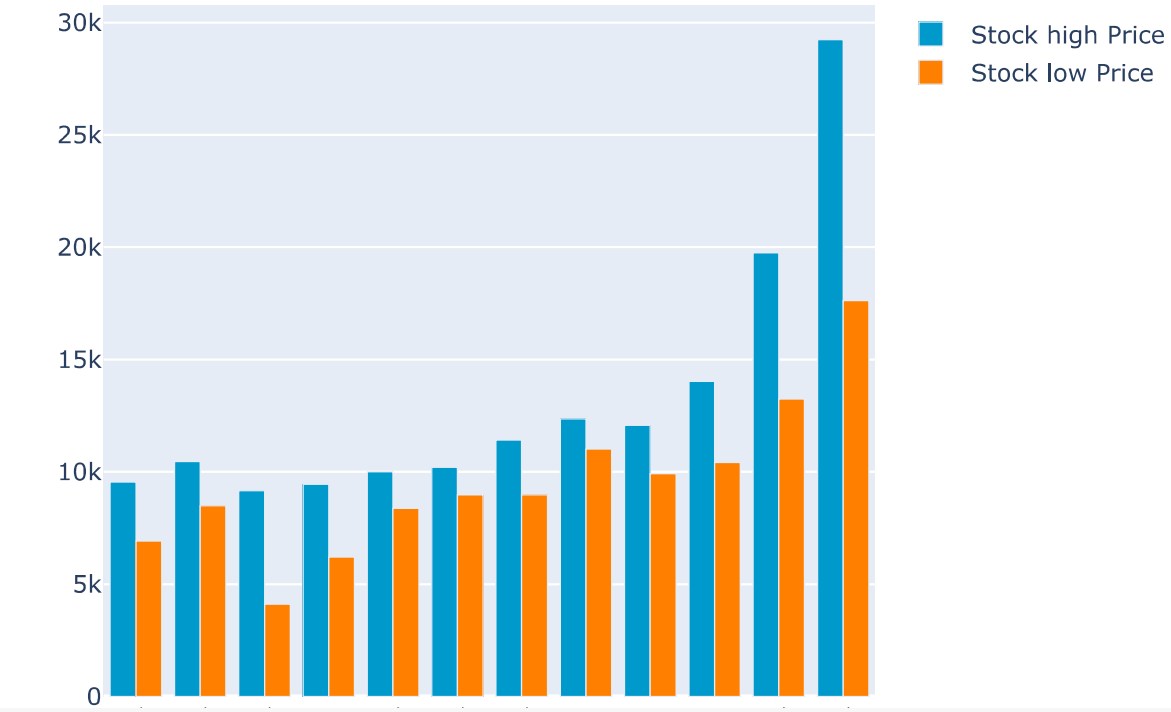
```
monthvise= y_2020.groupby(y_2020['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Date	Open	Close
January	8318.949597	8389.270476
February	9656.215113	9630.722185
March	6943.507009	6871.016113
April	7150.611328	7224.477328
May	9237.761530	9263.151745
June	9499.797005	9489.227214
July	9519.383852	9589.899729
August	11639.097215	11652.394185
September	10689.700163	10660.276856
October	11791.307491	11886.978201
November	16450.121647	16645.757422
December	21680.540827	21983.137097

```
fig = go.Figure()

fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```




```
y_2020.groupby(y_2020['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2020.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2020.groupby(y_2020['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```




```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'

fig = px.line(y_2020, x=y_2020.Date, y=[y_2020['Open'], y_2020['Close'],
                                        y_2020['High'], y_2020['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```


```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2021 = maindf.loc[(maindf['Date'] >= '2021-01-01')
                    & (maindf['Date'] < '2021-12-31')]

y_2021.drop(y_2021[['Adj Close','Volume']],axis=1)
```

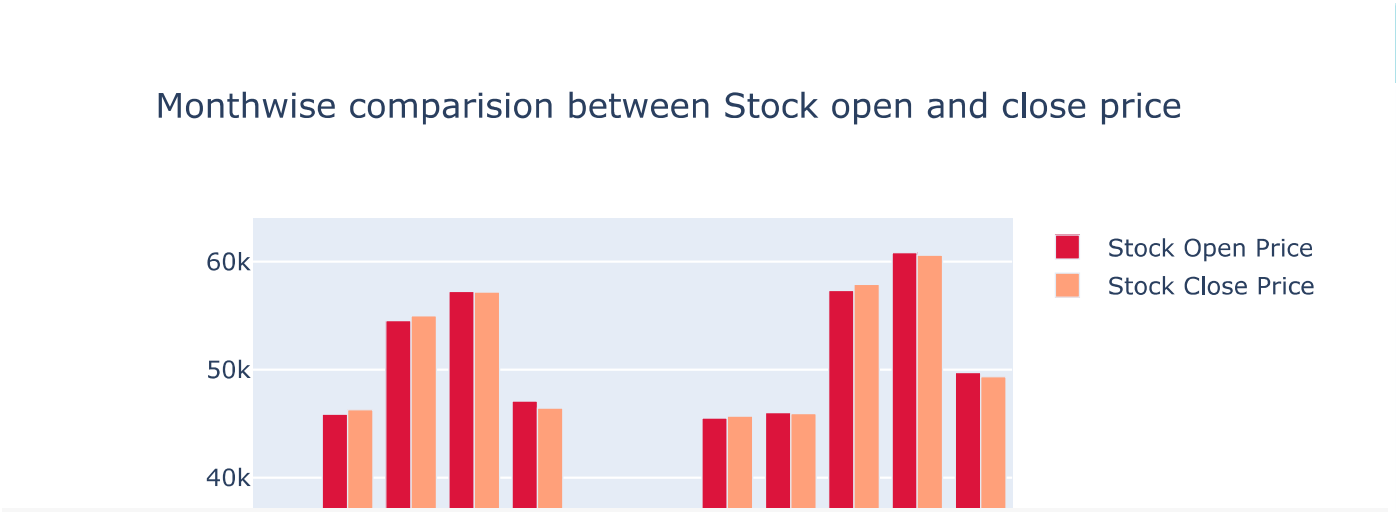
	Date	Open	High	Low	Close
2298	2021-01-01	28994.009766	29600.626953	28803.585938	29374.152344
2299	2021-01-02	29376.455078	33155.117188	29091.181641	32127.267578
2300	2021-01-03	32129.408203	34608.558594	32052.316406	32782.023438
2301	2021-01-04	32810.949219	33440.218750	28722.755859	31971.914063
2302	2021-01-05	31977.041016	34437.589844	30221.187500	33992.429688
...	...	...	...	...	...
2657	2021-12-26	50428.691406	51196.378906	49623.105469	50809.515625
2658	2021-12-27	50802.609375	51956.328125	50499.468750	50640.417969
2659	2021-12-28	50679.859375	50679.859375	47414.210938	47588.855469
2660	2021-12-29	47623.871094	48119.742188	46201.496094	46444.710938
2661	2021-12-30	46490.605469	47879.964844	46060.312500	47178.125000

364 rows × 5 columns

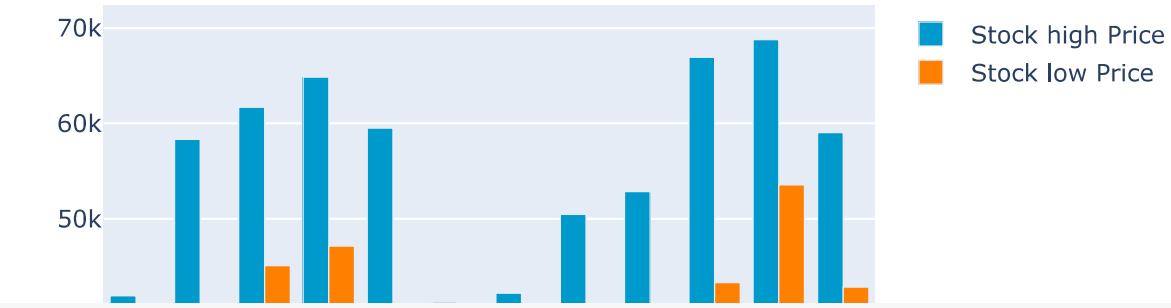
```
monthvise= y_2021.groupby(y_2021['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Date	Open	Close
January	34652.961694	34761.649950
February	45874.967216	46306.798968
March	54544.678176	54998.008695
April	57251.256250	57206.720052
May	47105.828503	46443.286668
June	35920.546940	35845.154688
July	34234.212450	34444.973790
August	45516.119834	45709.022682
September	46041.859375	45939.771484

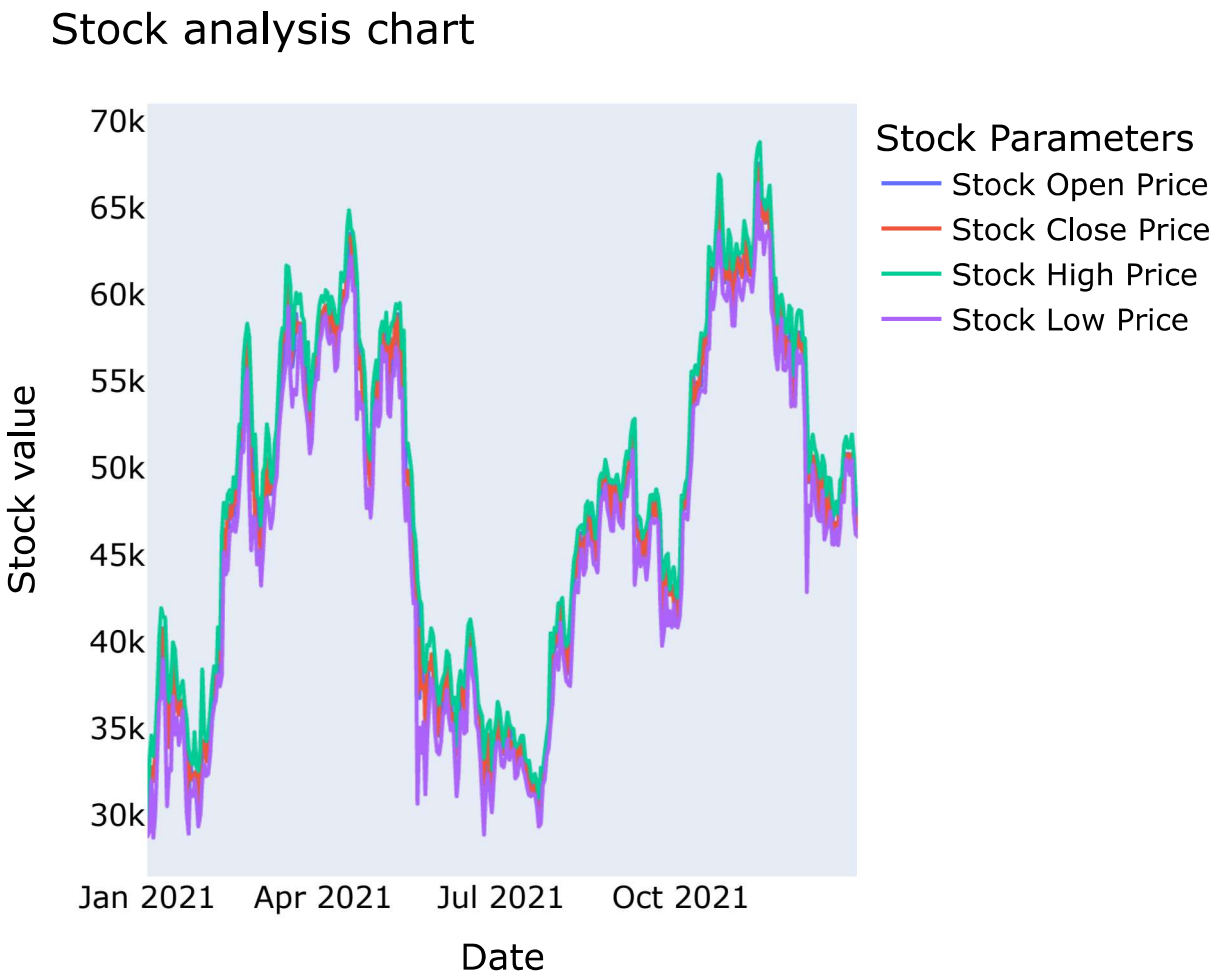
```
fig = go.Figure()

fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```




```
y_2021.groupby(y_2021['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2021.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2021.groupby(y_2021['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```




```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'

fig = px.line(y_2021, x=y_2021.Date, y=[y_2021['Open'], y_2021['Close'],
                                        y_2021['High'], y_2021['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```



```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_2022 = maindf.loc[(maindf['Date'] >= '2022-01-01')
                    & (maindf['Date'] < '2022-02-19')]

y_2022.drop(y_2015[['Adj Close','Volume']],axis=1)
```

2680	2022-01-18	42250.074219	42534.402344	41392.214844	42375.632813
2681	2022-01-19	42374.039063	42478.304688	41242.914063	41744.328125
2682	2022-01-20	41744.027344	43413.023438	40672.824219	40680.417969
2683	2022-01-21	40699.605469	41060.527344	35791.425781	36457.316406
2684	2022-01-22	36471.589844	36688.812500	34349.250000	35030.250000
2685	2022-01-23	35047.359375	36433.312500	34784.968750	36276.804688
2686	2022-01-24	36275.734375	37247.519531	33184.058594	36654.328125
2687	2022-01-25	36654.804688	37444.570313	35779.429688	36954.003906
2688	2022-01-26	36950.515625	38825.410156	36374.906250	36852.121094
2689	2022-01-27	36841.878906	37148.324219	35629.281250	37138.234375
2690	2022-01-28	37128.445313	37952.878906	36211.109375	37784.332031
2691	2022-01-29	37780.714844	38576.261719	37406.472656	38138.179688
2692	2022-01-30	38151.917969	38266.339844	37437.710938	37917.601563
2693	2022-01-31	37920.281250	38647.261719	36733.574219	38483.125000
2694	2022-02-01	38481.765625	39115.132813	38113.664063	38743.273438
2695	2022-02-02	38743.714844	38834.617188	36832.730469	36952.984375
2696	2022-02-03	36944.804688	37154.601563	36375.539063	37154.601563
2697	2022-02-04	37149.265625	41527.785156	37093.628906	41500.875000
2698	2022-02-05	41501.480469	41847.164063	41038.097656	41441.164063
2699	2022-02-06	41441.121094	42500.785156	41244.906250	42412.433594
2700	2022-02-07	42406.781250	44401.863281	41748.156250	43840.285156
2701	2022-02-08	43854.652344	45293.867188	42807.835938	44118.445313

```
monthvise= y_2022.groupby(y_2022['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Date	Open	Close
January	41368.073463	41114.422379
February	41722.190538	41811.714627
March	NaN	NaN
April	NaN	NaN
May	NaN	NaN

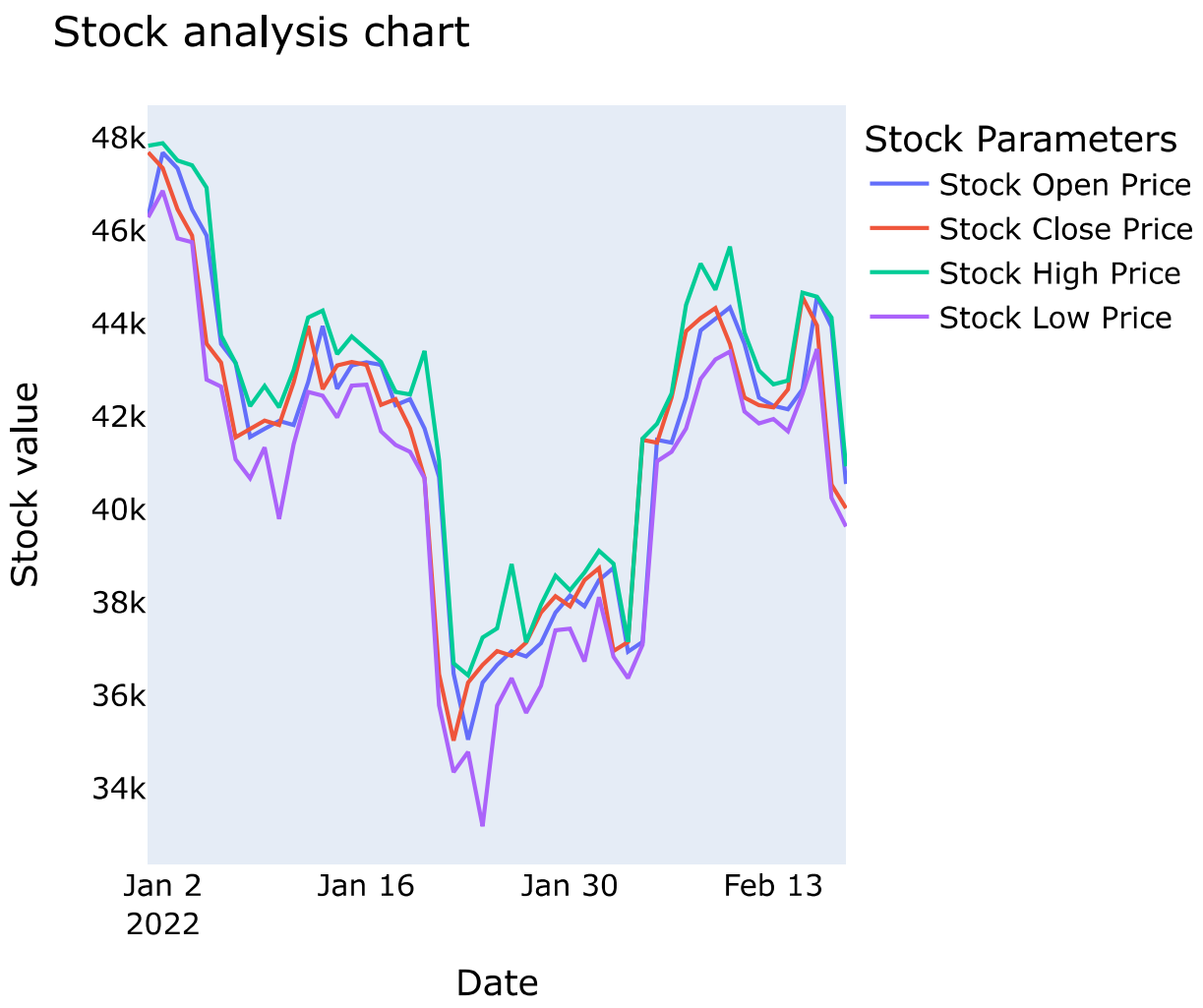
```
fig = go.Figure()

fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Open'],
    name='Stock Open Price',
    marker_color='crimson'
))
fig.add_trace(go.Bar(
    x=monthvise.index,
    y=monthvise['Close'],
    name='Stock Close Price',
    marker_color='lightsalmon'
))

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='Monthwise comparision between Stock open and close price')
fig.show()
```

Monthwise comparision between Stock open and close price

```
y_2022.groupby(y_2022['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2022.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

monthvise_low = y_2022.groupby(y_2022['Date'].dt.strftime('%B'))['Low'].min()
monthvise_low = monthvise_low.reindex(new_order, axis=0)

fig = go.Figure()
fig.add_trace(go.Bar(
    x=monthvise_high.index,
    y=monthvise_high,
    name='Stock high Price',
    marker_color='rgb(0, 153, 204)'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='rgb(255, 128, 0)'
))

fig.update_layout(barmode='group',
                  title=' Monthwise High and Low stock price')
fig.show()
```

Monthwise High and Low stock price

```
names = cycle(['Stock Open Price','Stock Close Price','Stock High Price','Stock Low Price'

fig = px.line(y_2022, x=y_2022.Date, y=[y_2022['Open'], y_2022['Close'],
                                        y_2022['High'], y_2022['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```



Overall analysis from 2014 - 2022

```
maindf['Date'] = pd.to_datetime(maindf['Date'], format='%Y-%m-%d')

y_overall = maindf.loc[(maindf['Date'] >= '2014-09-17')
                       & (maindf['Date'] <= '2022-02-19')]

y_overall.drop(y_overall[['Adj Close','Volume']],axis=1)
```

	Date	Open	High	Low	Close
0	2014-09-17	465.864014	468.174011	452.421997	457.334015
1	2014-09-18	456.859985	456.859985	413.104004	424.440002
2	2014-09-19	424.102997	427.834991	384.532013	394.795990
3	2014-09-20	394.673004	423.295990	389.882996	408.903992
4	2014-09-21	408.084991	412.425995	393.181000	398.821014
...	...	...	...	...	...
2708	2022-02-15	42586.464844	44667.218750	42491.035156	44575.203125
2709	2022-02-16	44578.277344	44578.277344	43456.691406	43961.859375
2710	2022-02-17	43937.070313	44132.972656	40249.371094	40538.011719
2711	2022-02-18	40552.132813	40929.152344	39637.617188	40030.976563
2712	2022-02-19	40022.132813	40246.027344	40010.867188	40126.429688

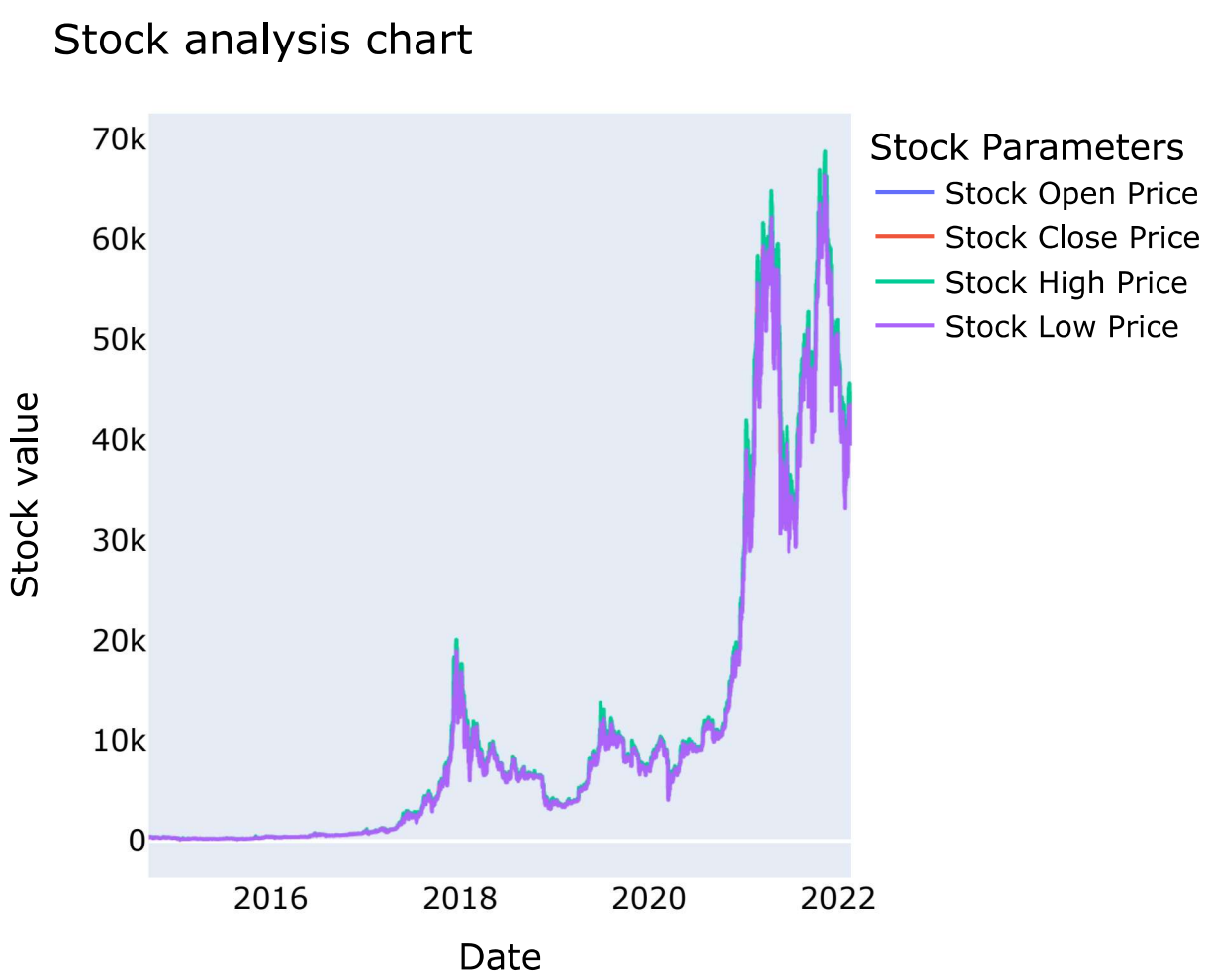
2713 rows × 5 columns

```
monthvise= y_overall.groupby(y_overall['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Date	Open	Close
January	12855.131425	12828.374881
February	12773.077824	12837.802432
March	10918.895761	10957.226324
April	11338.448900	11359.962198
May	10659.455257	10580.209317
June	9299.305977	9294.420703
July	9285.402500	9330.128271
August	11312.971706	11345.157739
September	10489.365578	10462.378150
October	11321.578327	11416.077925
November	12542.362183	12537.441752
December	12391.975010	12391.988926

```
names = cycle(['Stock Open Price', 'Stock Close Price', 'Stock High Price', 'Stock Low Price'
```

```
names = cycle([ Stock Open Price , Stock Close Price , Stock High Price , Stock Low Price

fig = px.line(y_overall, x=y_overall.Date, y=[y_overall['Open'], y_overall['Close'],
                                              y_overall['High'], y_overall['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='black',lege
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```



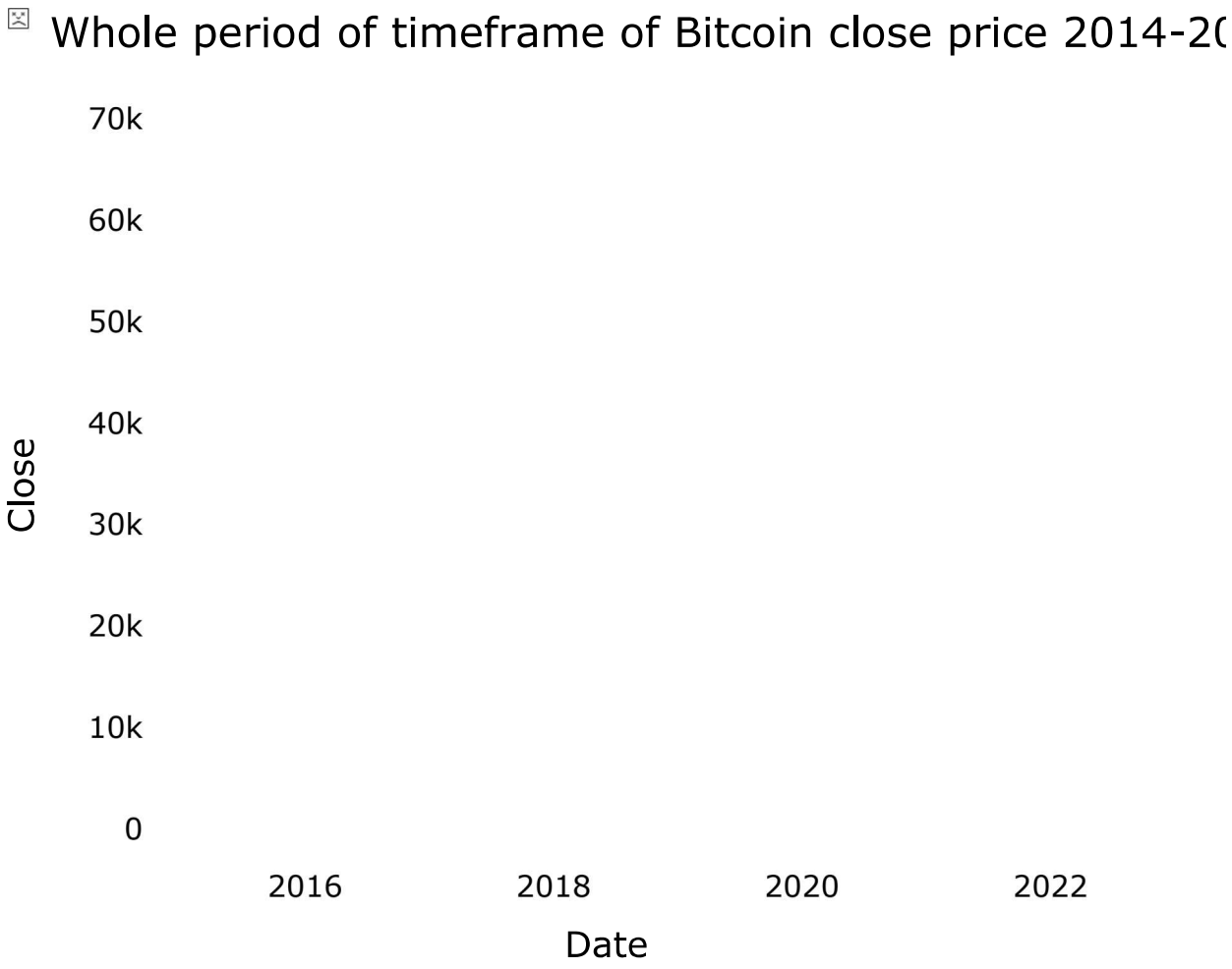
Building LSTM Model

```
#take all the close price
closedf = maindf[['Date','Close']]
print("Shape of close dataframe:", closedf.shape)
```

```
Shape of close dataframe: (2713, 2)
```

```
fig = px.line(closedf, x=closedf.Date, y=closedf.Close,labels={'date':'Date','close':'Clos
fig.update_traces(marker_line_width=2, opacity=0.8, marker_line_color='orange')
fig.update_layout(title_text='Whole period of timeframe of Bitcoin close price 2014-2022',
                  font_size=15, font_color='black')
```

```
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



Taking the data of just one year

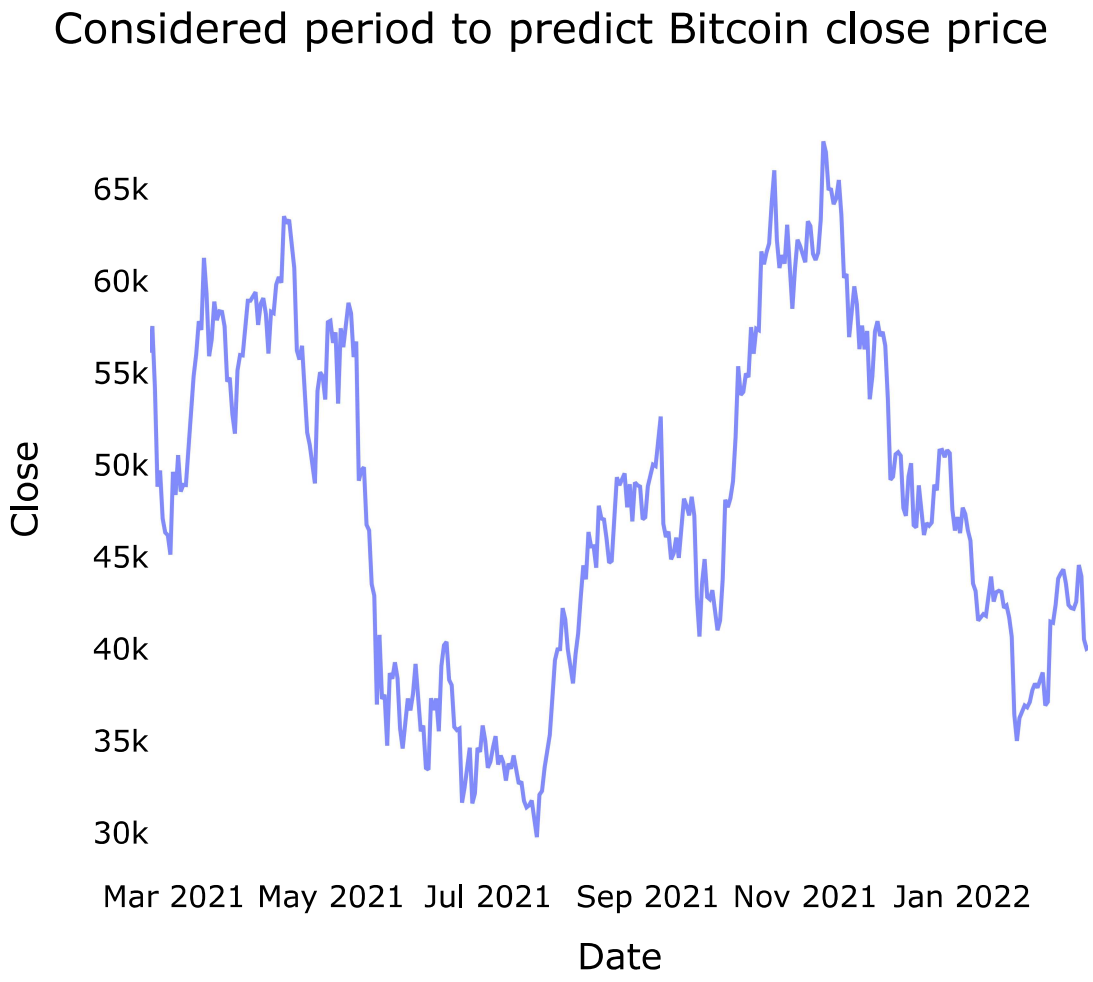
```
closedf = closedf[closedf['Date'] > '2021-02-19']
close_stock = closedf.copy()
print("Total data for prediction: ",closedf.shape[0])
```

```
Total data for prediction:  365
```

```
closedf
```

	Date	Close
2348	2021-02-20	56099.519531
2349	2021-02-21	57539.945313
2350	2021-02-22	54207.320313
2351	2021-02-23	48824.425781
2352	2021-02-24	49705.332031
...	...	...

```
fig = px.line(closedf, x=closedf.Date, y=closedf.Close,labels={'date':'Date','close':'Clos
fig.update_traces(marker_line_width=2, opacity=0.8, marker_line_color='orange')
fig.update_layout(title_text='Considered period to predict Bitcoin close price',
                  plot_bgcolor='white', font_size=15, font_color='black')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



Normalizing the data

```
# deleting date column and normalizing using MinMax Scaler

del closedf['Date']
```

```
scaler=MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))
print(closedf.shape)
```

```
(365, 1)
```

```
#training data: 60% and testing data: 40%
training_size=int(len(closedf)*0.60)
test_size=len(closedf)-training_size
train_data,test_data=closedf[0:training_size,:],closedf[training_size:len(closedf),:1]
print("train_data: ", train_data.shape)
print("test_data: ", test_data.shape)
```

```
train_data:  (219, 1)
test_data:  (146, 1)
```

```
# converting an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)
```

```
time_step = 15
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)

print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print("X_test: ", X_test.shape)
print("y_test", y_test.shape)
```

```
X_train:  (203, 15)
y_train:  (203,)
X_test:  (130, 15)
y_test (130,)
```

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
```

```
X_train:  (203, 15, 1)
X_test:  (130, 15, 1)
```

Actual Model Building

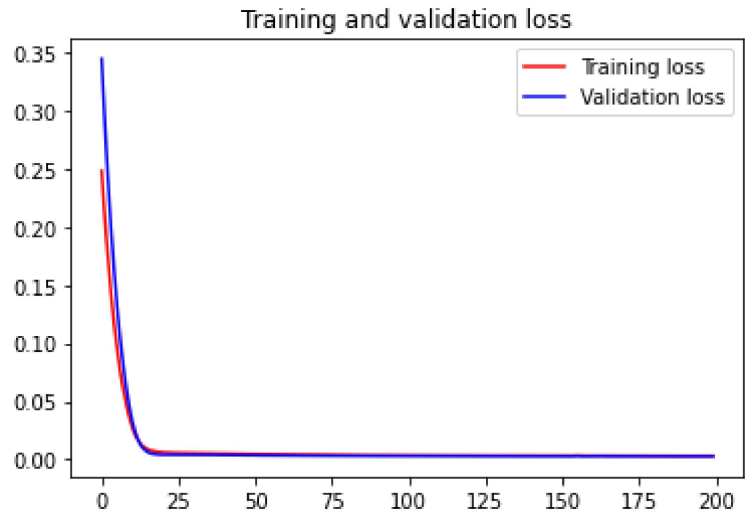
```
model=Sequential()
# A Sequential model is appropriate for a
# plain stack of layers where each layer has exactly one input tensor and one output tensc
model.add(LSTM(10,input_shape=(None,1),activation="relu"))
model.add(Dense(1))
# Model fitting is a measure of how well a machine learning
# model generalizes to similar data to that on which it was trained. A model that is well-
#ReLU activation function is an activation function defined as the positive part of its ar
model.compile(loss="mean_squared_error",optimizer="adam")
#Compile defines the loss function, the optimizer and the metrics.
```

```
history = model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=200,batch_size=
# Model fitting is a measure of how well a machine learning model generalizes to
# similar data to that on which it was trained. A model that is well-fitted produces more
```

```
7/7 [==============================] - 0s 13ms/step - loss: 0.0032 - val_loss: 0.
Epoch 173/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0032 - val_loss: 0.
Epoch 174/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 175/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0033 - val_loss: 0.
Epoch 176/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0033 - val_loss: 0.
Epoch 177/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 178/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 179/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 180/200
7/7 [==============================] - 0s 14ms/step - loss: 0.0032 - val_loss: 0.
Epoch 181/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 182/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 183/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 184/200
7/7 [==============================] - 0s 13ms/step - loss: 0.0032 - val_loss: 0.
Epoch 185/200
7/7 [==============================] - 0s 14ms/step - loss: 0.0032 - val_loss: 0.
Epoch 186/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0032 - val_loss: 0.
Epoch 187/200
7/7 [==============================] - 0s 13ms/step - loss: 0.0032 - val_loss: 0.
Epoch 188/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 189/200
7/7 [==============================] - 0s 13ms/step - loss: 0.0032 - val_loss: 0.
Epoch 190/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0032 - val_loss: 0.
Epoch 191/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0032 - val_loss: 0.
Epoch 192/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0031 - val_loss: 0.
Epoch 193/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
```

```
Epoch 194/200
7/7 [==============================] - 0s 13ms/step - loss: 0.0032 - val_loss: 0.
Epoch 195/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss: 0.
Epoch 196/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0032 - val_loss: 0.
Epoch 197/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0031 - val_loss: 0.
Epoch 198/200
7/7 [==============================] - 0s 12ms/step - loss: 0.0032 - val_loss: 0.
Epoch 199/200
7/7 [==============================] - 0s 11ms/step - loss: 0.0032 - val_loss: 0.
Epoch 200/200
7/7 [==============================] - 0s 13ms/step - loss: 0.0031 - val_loss: 0.
```

```
#plotting plot v/s validation loss
import matplotlib.pyplot as plt
#Validation loss: a metric used to assess the performance of a deep learning model on the
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(loss))
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend(loc=0)
plt.figure()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

Prediction and performance metrics

```
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
train_predict.shape, test_predict.shape
```

```
((203, 1), (130, 1))
```

Model Evaluation

```
# Transform back to original form

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-1,1))
original_ytest = scaler.inverse_transform(y_test.reshape(-1,1))
```

Evaluation metrices RMSE, MSE and MAE

```
# Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Train data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-------------------------------------------------------------------------------------")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

```
Train data RMSE:  2129.3152568478795
Train data MSE:  4533983.46304515
Train data MAE:  1665.3674472863304
-------------------------------------------------------------------------------------
Test data RMSE:  1960.3255531903724
Test data MSE:  3842876.2744911397
Test data MAE:  1506.9978064923075
```

Variance Regression score

```
print("Train data explained variance regression score:",
      explained_variance_score(original_ytrain, train_predict))
print("Test data explained variance regression score:",
      explained_variance_score(original_ytest, test_predict))
```

```
Train data explained variance regression score: 0.948974546996319
Test data explained variance regression score: 0.9534407390645407
```

R square score for regression

```
print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

```
Train data R2 score: 0.9478901379756098
Test data R2 score: 0.9528966131503859
```

Regression Loss Mean Gamma deviance regression loss (MGD) and Mean Poisson deviance regression loss (MPD)

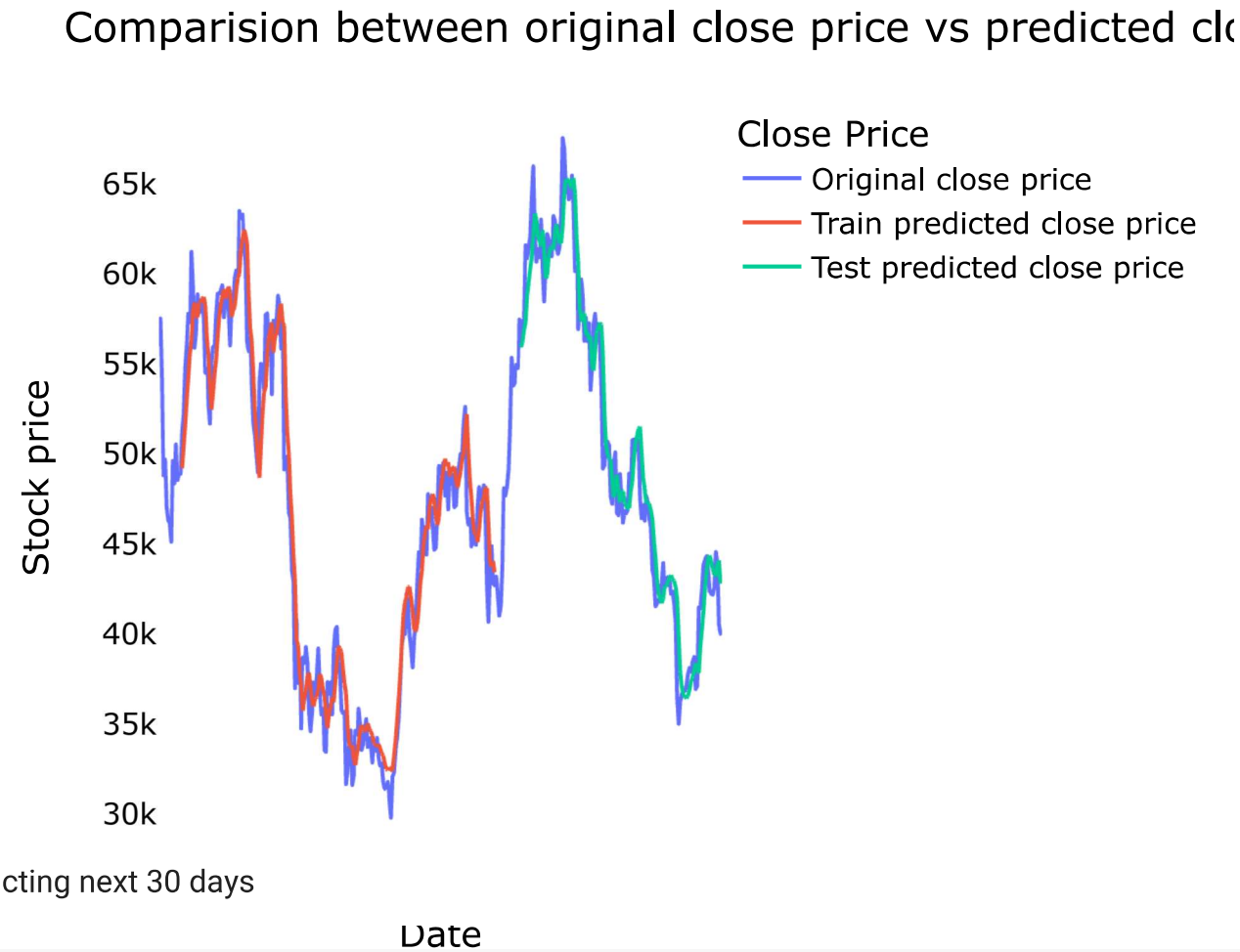
```
print("Train data MGD: ", mean_gamma_deviance(original_ytrain, train_predict))
print("Test data MGD: ", mean_gamma_deviance(original_ytest, test_predict))
print("----------------------------------------------------------------------")
print("Train data MPD: ", mean_poisson_deviance(original_ytrain, train_predict))
print("Test data MPD: ", mean_poisson_deviance(original_ytest, test_predict))
```

```
Train data MGD:  0.0021756797375580453
Test data MGD:  0.0015761373129938724
----------------------------------------------------------------------
Train data MPD:  97.60791421672019
Test data MPD:  76.59152924679354
```

Comparision of original stock close price and predicted close price

```
# shift train predictions for plotting

look_back=time_step
trainPredictPlot = np.empty_like(closedf)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
print("Train predicted data: ", trainPredictPlot.shape)

# shift test predictions for plotting
testPredictPlot = np.empty_like(closedf)
testPredictPlot[:, :] = np.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(closedf)-1, :] = test_predict
print("Test predicted data: ", testPredictPlot.shape)

names = cycle(['Original close price','Train predicted close price','Test predicted close


plotdf = pd.DataFrame({'date': close_stock['Date'],
                       'original_close': close_stock['Close'],
                      'train_predicted_close': trainPredictPlot.reshape(1,-1)[0].tolist(),
                      'test_predicted_close': testPredictPlot.reshape(1,-1)[0].tolist()})

fig = px.line(plotdf,x=plotdf['date'], y=[plotdf['original_close'],plotdf['train_predicted
                                          plotdf['test_predicted_close']],
              labels={'value':'Stock price','date': 'Date'})
fig.update_layout(title_text='Comparision between original close price vs predicted close
                  plot_bgcolor='white', font_size=15, font_color='black', legend_title_te
fig.for_each_trace(lambda t:  t.update(name = next(names)))

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```

```
Train predicted data:  (365, 1)
Test predicted data:  (365, 1)
```




```
x_input=test_data[len(test_data)-time_step:].reshape(1,-1)
temp_input=list(x_input)
temp_input=temp_input[0].tolist()

from numpy import array

lst_output=[]
n_steps=time_step
i=0
pred_days = 30
while(i<pred_days):

    if(len(temp_input)>time_step):

        x_input=np.array(temp_input[1:])
        #print("{} day input {}".format(i,x_input))
        x_input = x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))

        yhat = model.predict(x_input, verbose=0)
        #print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)

        lst_output.extend(yhat.tolist())
```

```
        i=i+1

    else:

        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        temp_input.extend(yhat[0].tolist())

        lst_output.extend(yhat.tolist())
        i=i+1

print("Output of predicted next days: ", len(lst_output))
```

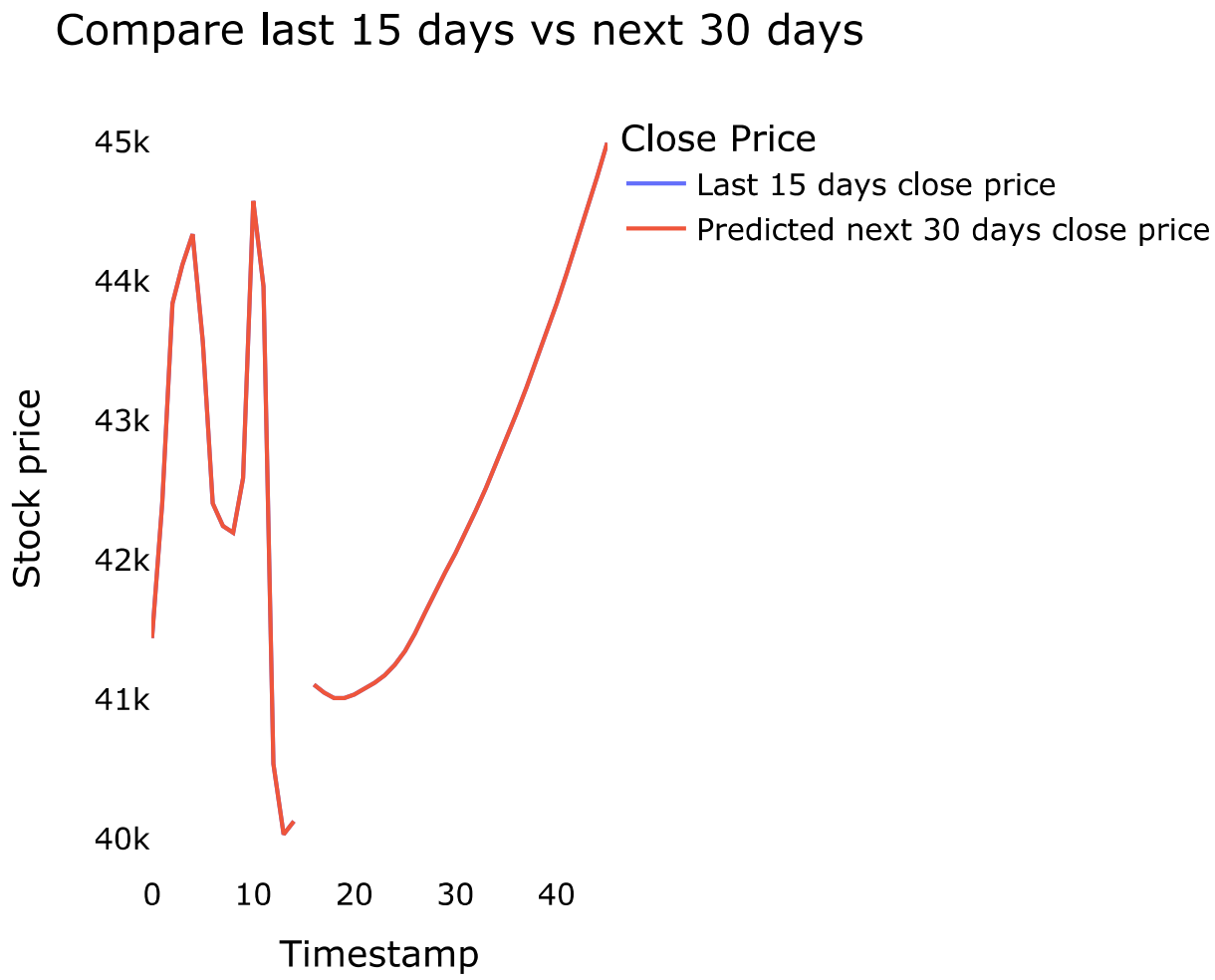
```
Output of predicted next days:  30
```

Plotting last 15 days dataset with next 30 predicted days dataset

```
last_days=np.arange(1,time_step+1)
day_pred=np.arange(time_step+1,time_step+pred_days+1)
print(last_days)
print(day_pred)
```

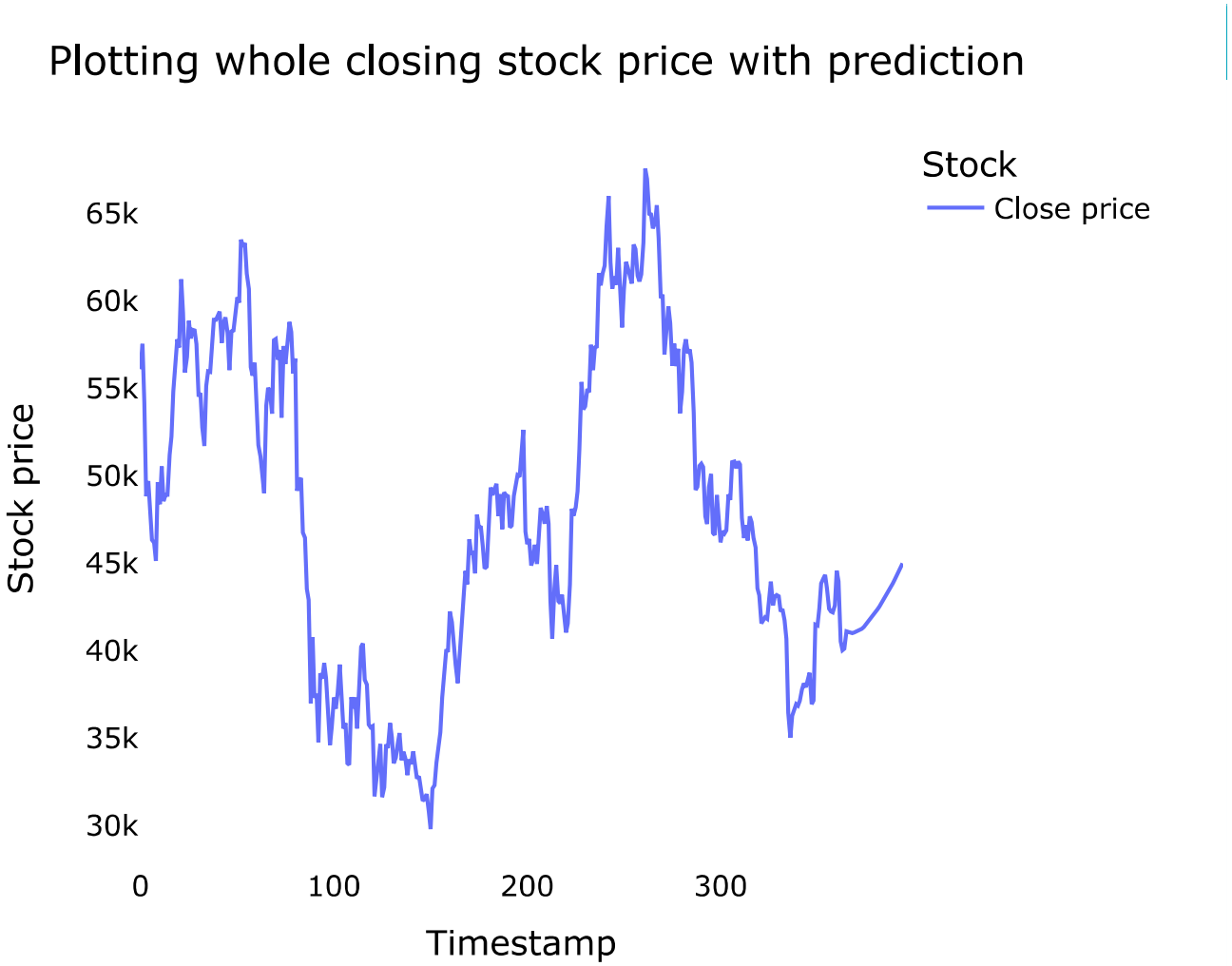
```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
[16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
 40 41 42 43 44 45]
```

```
temp_mat = np.empty((len(last_days)+pred_days+1,1))
temp_mat[:] = np.nan
temp_mat = temp_mat.reshape(1,-1).tolist()[0]

last_original_days_value = temp_mat
next_predicted_days_value = temp_mat

last_original_days_value[0:time_step+1] = scaler.inverse_transform(closedf[len(closedf)-ti
next_predicted_days_value[time_step+1:] = scaler.inverse_transform(np.array(lst_output).re

new_pred_plot = pd.DataFrame({
    'last_original_days_value':last_original_days_value,
    'next_predicted_days_value':next_predicted_days_value
})

names = cycle(['Last 15 days close price','Predicted next 30 days close price'])

fig = px.line(new_pred_plot,x=new_pred_plot.index, y=[new_pred_plot['last_original_days_va
                                                      new_pred_plot['next_predicted_days_v
              labels={'value': 'Stock price','index': 'Timestamp'})
fig.update_layout(title_text='Compare last 15 days vs next 30 days',
                  plot_bgcolor='white', font_size=15, font_color='black',legend_title_text

fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



Plotting entire Closing Stock Price with next 30 days period of prediction

```
lstmdf=closedf.tolist()
lstmdf.extend((np.array(lst_output).reshape(-1,1)).tolist())
lstmdf=scaler.inverse_transform(lstmdf).reshape(1,-1).tolist()[0]

names = cycle(['Close price'])

fig = px.line(lstmdf,labels={'value': 'Stock price','index': 'Timestamp'})
fig.update_layout(title_text='Plotting whole closing stock price with prediction',
                  plot_bgcolor='white', font_size=15, font_color='black',legend_title_text

fig.for_each_trace(lambda t:  t.update(name = next(names)))

fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



✓ 0s completed at 10:29

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.