BITCOIN PRICE PREDICTION USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

DIVYA SREE CHIGURAKULA (18HR1A0515)

Under the Guidance of

Prof. Dr.PRABHU KUMAR P.C

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

TABLE OF CONTENTS

1. INTRODUCTION

After the boom and bust of cryptocurrencies' prices in recent years, Bitcoin has been increasingly regarded as an investment asset. Because of its highly volatile nature, there is a need for good predictions on which to base investment decisions. Although existing studies have leveraged machine learning for more accurate Bitcoin price prediction, few have focused on the feasibility of applying different modeling techniques to samples with different data structures and dimensional features. To predict Bitcoin price at different frequencies using machine learning techniques, we first classify Bitcoin price by daily price and high-frequency price. A set of high-dimension features including property and network, trading and market, attention and gold spot price are used for Bitcoin daily price prediction, while the basic trading features acquired from a cryptocurrency exchange are used for 5-minute interval price prediction. Statistical methods including Logistic Regression and Linear Discriminant Analysis for Bitcoin daily price prediction with high-dimensional features achieve an accuracy of 66%, outperforming more complicated machine learning algorithms. Machine learning models including Random Forest, XGBoost, Quadratic Discriminant Analysis, Support Vector Machine and Long Short-term Memory for Bitcoin 5-minute interval price prediction are superior to statistical methods. Our investigation of Bitcoin price prediction can be considered a pilot study of the importance of the sample dimension in machine learning techniques.

PROBLEM STATEMENT

To develop a model which can help us to predict the price of the cryptocurrency used like Bitcoin, with low error rate and a high precision of accuracy. The model will not tell the future, but it might forecast the general trend and the direction to expect the prices to move

2. LITIRATURE SURVEY

SL. NO	AUTHO RS	JOURNA LNAME	METHODOLOGY	GAPS
01	Alvin Ho[1], Ramesh Vatambeti[1*], Sathish Kumar Ravichandr an[1]	Bitcoin Price Prediction Using Machine Learning and Artificial Neural Network Model	With the help of python libraries, the data filtration process was done. Python has provided with a best feature for data analysis and visualization. After the understanding of the data, we trimthe data and use the features or attributes best suited for the model. Implementation of the model is done and the result is recorded.	It was discovered that the linear regression model's accuracy rate is very highwhen compared to other Machine Learning modelsfrom related works; it wasfound to be 99.87 percent accurate. The LSTM model,shows a mini error rate of 0.08 percent. This, in turn, demonstrates that the neuralnetwork model is more optimized than the machine learning model.
02	Ana Lucia Lima	Bitcoin Price Prediction Using Recurrent Neural	It is defined as the procedure of collecting, measuring, and analyzing accurate insights for research using standard validationtechniques It was discovered that the linear regression model's accuracy rate is very high when	adaptation of the block chaintechnology, causes the biggest concern i.e., scalability. It is still dwarfedby the number of transactions that, VISA, processes each day.

		Networks andLSTM	compared to other Machine Learning models from related works; it was found to be 99.87 percent accurate. The LSTM model, on the other hand, shows amini error rate of 0.08 percent. This, in turn, demonstrates that theneural network model is more optimized than the machine learning model.	Cryptocurrencies have not been around for long enoughto provide sufficient information regarding the resistance and key support compared to stock market, currencies and commodities.This makes it difficult to predict and practice.
03	Seçkin Karasu; Aytaç Altan; Zehra Saraç; Rifat Hacioğlu	Prediction of Bitcoin Prices with Machine Learning Methods usingTime Series Data	Bitcoin prediction is performed with Linear Regression (LR) andSupport Vector Machine (SVM) from machine learning methods by using time series consisting of daily Bitcoin closing prices between 2012-2018.	In the study, the A/D oscillator is also used as amodel input. used. 2192 pieces of databetween 2012-2018 The training and testing process of models for the 10-fold crossover point made using the verificationmethod.
04	A. Demir, B. N. Akılotu, Z. Kadiroğlu and A. Şengür	Bitcoin Price Prediction Using Machine Learning Methods	Bitcoin price estimation was madeby using machine learning methods using KAGGLE Bitcoin Dataset 2010-2019 data set. The methods used are long-short term memory networks, support vector machines, artificial neural networks, Naive Bayes, decision trees and the nearest neighbor algorithm	Obtained accuracy rates are 97.2%, 91.8%, 86.6%, 85%, 81.2% respectively.not obtained exact results
05	L. J. Paraband P. P. Nitnaware	Evaluation of Cryptocurrency coins with Machine Learning algorithms and Blockchain Technology	Two Machine Learning algorithmmodels ARIMA (auto-regressive integrated moving average model)and LSTM (long-short-term memory networks) where the database is protected by Blockchain technology	Getting accurate and improved results by including additional modelin future work

06	A. Tanwar and V. Kumar	Prediction of Cryptocurren c y prices using Transformers and Long Short term Neural Networks	The method uses Transformers and Long-short term Neural networks(LSTM) to forecast the prices of various cryptocurrencies	Although using LSTM alongwith transformers leads to longer computational times, but the predictive accuracy is better as compared to traditional regression neural networks and kNN forecasting models
07	M. Fernandes, S. Khanna, L. Monteiro, A. Thomas and G. Tripathi	Bitcoin Price Prediction	Bitcoin prices and design-integration of price prediction of different cryptocurrencies using RNN (Recurrent Neural Network),LSTM (Long Short-Term Memory) and GRU (Gated recurrent units)	Due to the data mapping issue faced while developingthe sentiment analysis model, it had to be dropped and use only historical Bitcoin transactions data for building the model.
08	Q. Guo, S. Lei, Q. Ye and Z. Fang	MRC-LSTM: A Hybrid Approach of Multi-scale Residual CNNand LSTM to Predict BitcoinPrice,	MRC-LSTM, which combines aMulti-scale Residual Convolutional neural network (MRC) and a Long Short-Term Memory (LSTM)	More future work could befocused on comprehensive metrics which measure the investor's attention to moretimely detection of bitcoin market volatility and thus more accurate price prediction.
9	Zidi Gao; YiwenHe; Ercan Engin Kuruoglu	A Hybrid model integrating LSTM and Garch for Bitcoin Price Prediction	A hybrid approach which combines models such as Generalized Autoregressive Conditional Heteroskedasticity (GARCH) with the nonlinear modelling potential of Long-ShortTerm Memory (LSTM) neural networks.	This parametric models like GARCH with deep neural network may come up with better results in cryptocurrency price forecasting when short data sequences are available.
10	D. R. Pant, P. Neupane, A. Poudel, A. K. Pokhrel and B. K. Lam a,	"Recurrent Neural Network Based BitcoinPrice Prediction by Twitter Sentiment Analysis,	This research is concerned with predicting the volatile price of Bitcoin by analyzing the sentimentin Twitter and to find the relation between them.	The accuracy for sentiment classification of tweets in two class positive and negative is found to be 81.39 % and the overall price prediction accuracy using RNN is found to be 77.62%.

11	H. Kavitha, U. K. Sinhaand S. S. Jain	Performance Evaluation of Machine Learning Algorithms forBitcoin Price Prediction	predict the price of Bitcoin using Recurrent Neural Network(RNN),Long Short Term Memory (LSTM) and Linear Regression(LR) to predict the price of Bitcoin	One limitation in training both the models is the significant computation required. If the size of the dataset is small then the RNN model does not train well and results in bad set ofpredictions
12	A. Mittal, V. Dhiman, A. Singh and C. Prakash	Short-Term Bitcoin Price Fluctuation Prediction Using Social Media and Web Search Data,	Linear regression, polynomial regression, Recurrent Neural Network, and Long Short Term Memory based analysis	Among tweet volume, Google trends and tweet sentiments, tweet sentiment analysis has shown the worstresults. After applying the algorithms - LSTM, RNN, Polynomial regression is predicted with accuracy 77.01% and 66.66% of polynomial regression
13	S. Velankar, S. Valecha and S. Maji	Bitcoin price prediction using machine learning	Bayesian Regression GLM/Random Forest 1.	The price of Bitcoin does not depend on the business events or intervening government unlike stock market. thus we feel it is necessary to leverage machine learning technologyto predict the price of Bitcoin.
14	M. Samaddar, R. Roy, S.De and R. Karmakar	A Comparative Study of Different Machine Learning Algorithms onBitcoin Value Prediction,	Neural network algorithms, such as artificial neural network (ANN), recurrent neural network (RNN) and convolutional neural network (CNN), as well as some famous supervised learning algorithms such as Random Forest(RF) and k-nearest neighbors (k-NN), to form the analysis.	The data set size is a problem. Many studies don'thave long, and detailed data sets and the values predictedfrom them become very inaccurate. So, studies have to be done to maximize accuracy in small datasets and the overall accuracy of prediction can be made a great by improving the
15	E. Jakubowiczand E. Abdelfattah	The Rise andFall of Bitcoin: Predicting Market Direction Using Machine Learnin Models	Logistic Regression, Support Vector Machine (SVM), RandomForest (RF), KNN, and Decision Tree (DT),	The amount of data has any effect on the overall scoring,or if there are other factors in play, this study would need to be performed multiple times with other datasets.

16	ZheshiChen Chunhong Li WenjunSun	Bitcoin price prediction using machine learning: An approach to sample dimension engineering	Logistic Regression and Linear Discriminant Analysis for Bitcoin daily price prediction withhigh-dimensional features achieve an accuracy of 66%.	More future work could befocused on comprehensivemetrics which measure the investor's attention to moretimely detection of bitcoin market volatility and thus more accurate price prediction.
17	M. Mittal and G. Geetha,	Predicting Bitcoin Price using Machine Learning	Machine learning regression- based algorithms to build a prediction model for analysing future bitcoin prices. based on aneural network model named GRU (Gated Recurrent Unit). Root Mean Square Error and Mean Absolute Percent Error arethe key performance indicators tomeasure forecast accuracy.	The prediction shown is confined to previously observed data and shows that the GRU model is a potent learner on the trainingdataset, smart enough to recognise deep-rooted vulnerabilities and similarities.
18	Luisanna cocco,roberto tonelli and michele marchesi	Prediction of bitcoin prices through machine learning based frameworks	K-fold cross validation,bayesianneural network,ffnn,LSTMNN	Future work aims to performa more exhaustive optimization of all proposed frameworks in the work to obtain even higher performance
19	S M Raju, Ali MohammadTarif	Real-Time Prediction of BITCOIN Price usingML Techniquesand Public Sentiment Analysis	Predictable bitcoin price directionof Bitcoin in USD by machine learning techniques and sentimentanalysis.they have applied sentiment analysis and supervised machine learning principles to theextracted tweets from Twitter and Reddit posts, and analyzed the correlation between bitcoin price movements and sentiments in tweets..	Due to the difficulty of evaluating the exact natureof a Time Series(ARIMA)model, it is often very difficult to produce appropriate forecasts. whereas the ARIMA model RMSE is 209.263 which shows thatLSTM with multi feature shows the more accurate result.

20	Lekkala Sreekant hReddy, Dr.P. Sriramya	A Research OnBitcoin Price Prediction Using Machine Learning Algorithms	Used an algorithm linked to artificial intelligence named LASSO(least absolute shrinkage selection operator. In LASSO finding of the results from a largerdatabase is quick and fast.we will predict the sign of the daily price change with highest possible accuracy.	Using different algorithms like SVM(support vector machine),coinmarkupcap, Quandl, GLM, CNN(Convolutional NeuralNetworks)and RNN(which do not have a great time management

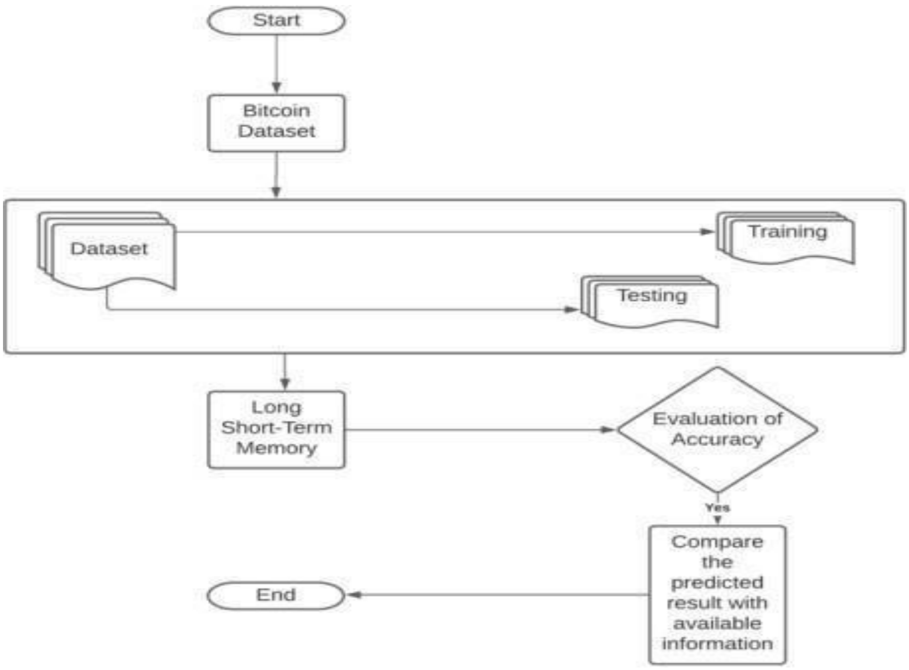
3. SYSTEM DESIGN

FRAMEWORK

- To develop a model which can help us to predict the price of the bitcoin used , with low error rate and a high precision of accuracy.
- While using this model, first, the dataset of the bitcoin from online source is collected which represent in USD over the years.
- Next,involves filtering and cleaning the data where it will remove all the incomplete data and also filters unnecessary features in data.
- Later,we do training using algorithms to predict the future price followed by testing to measure the accuracy of the algorithm.
- Finally after processing the training with the help of data set features and testing,we compare the predicted price of bitcoin ata a given time period withthe real world bitcoin price at particular period of time and evalueate the accuracy and efficiency of our model.

DATASET

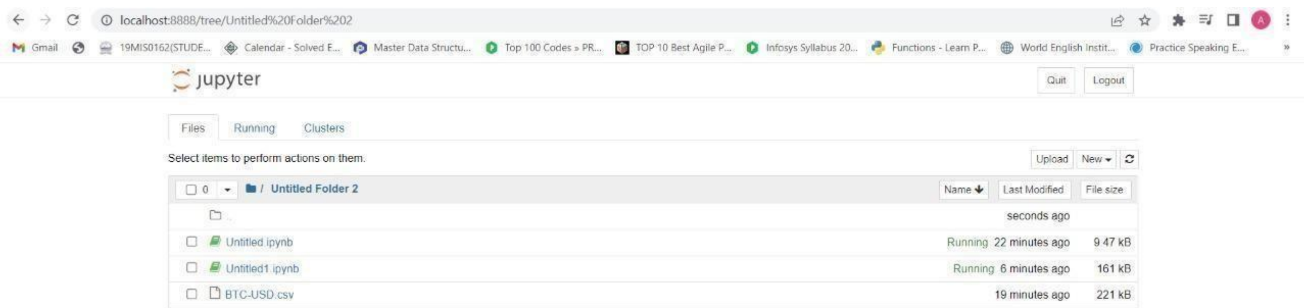
https://www.kaggle.com/datasets/meetnagadia/bitcoin-stock-data-sept-17-2014-august-24-2021



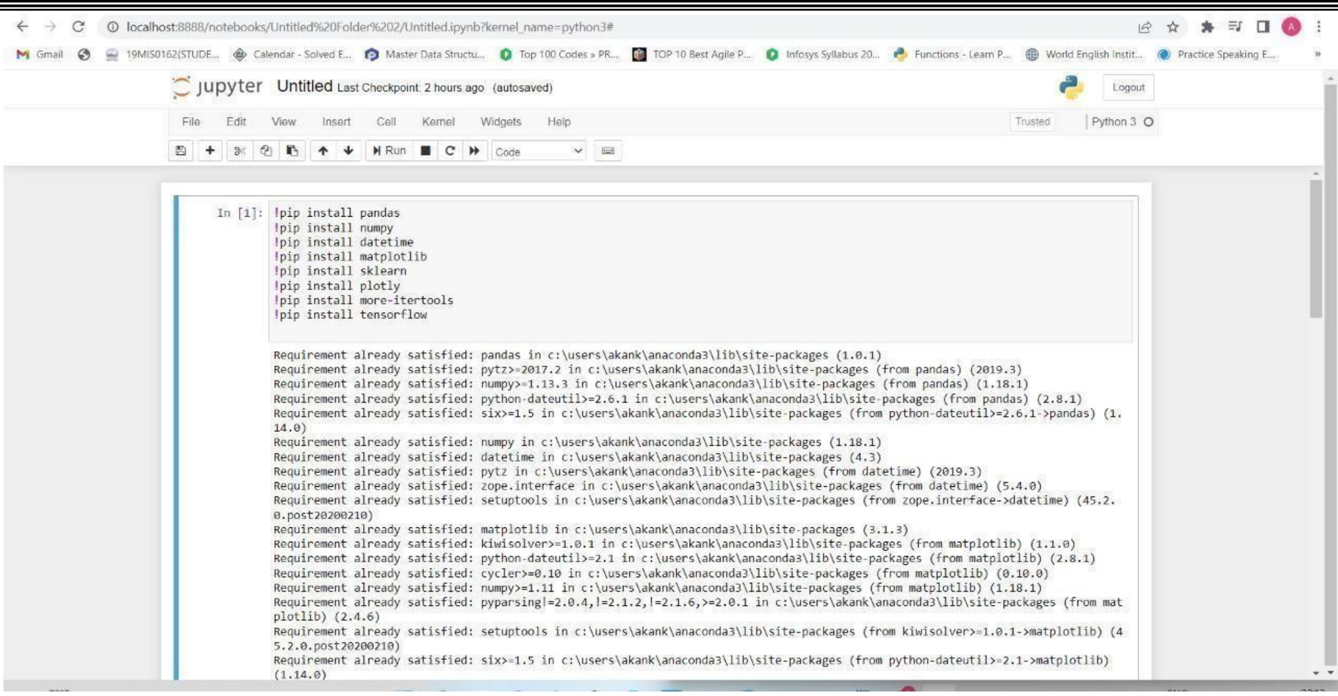
4. SYSTEM IMPLEMENTATION

CODE

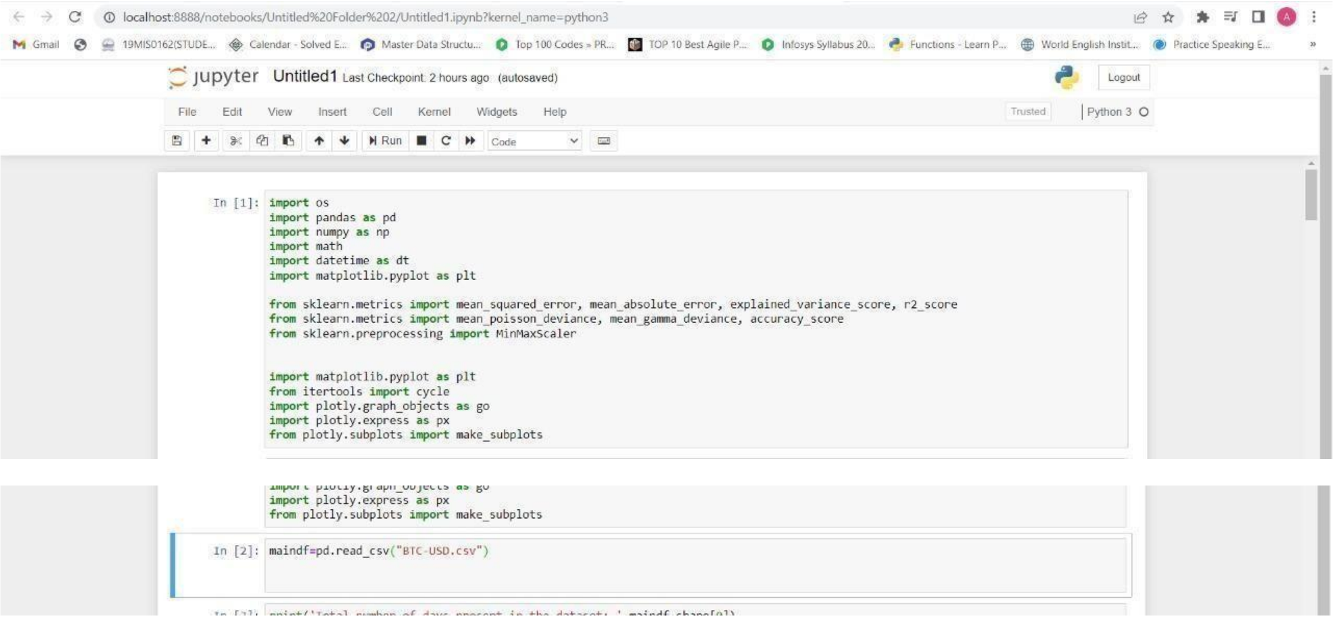
Creating files



Installing of commands



Importing libraries



Loading the dataset

Analyzing the data

In [3]:
```
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
```

In [4]:
```
maindf.shape
```

Out[4]:
```
(2713, 7)
```

In [5]:
```
maindf.head()
```

Out[5]:

	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	0.000000	413.104004	424.440002	424.440002	0
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

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

In [6]:
```
maindf.tail()
```

Out[6]:

	Date	Open	High	Low	Close	Adj Close	Volume
2708	2022-02-15	42586.464844	44667.218750	42491.035156	44575.203125	44575.203125	22721659051
2709	2022-02-16	44578.277344	44578.277344	43456.691406	43961.859375	43961.859375	19792547657
2710	2022-02-17	43937.070313	44132.972656	40249.371094	40538.011719	40538.011719	26246662813
2711	2022-02-18	40552.132813	40929.152344	39637.617188	40030.976563	40030.976563	23310007704
2712	2022-02-19	40022.132813	40246.027344	40010.867188	40126.429688	40126.429688	22263900160

In [7]:
```
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
```

Checking for Null values

In [8]:
```
maindf.describe()
```

Out[8]:

	Open	High	Low	Close	Adj Close	Volume
count	2713.000000	2713.000000	2713.000000	2713.000000	2713.000000	2.713000e+03
mean	11311.041069	11614.124086	10975.555057	11323.914637	11323.914637	1.470461e+10
std	16106.428891	16537.506630	15608.572560	16110.365010	16110.365010	2.001628e+10
min	176.897003	0.000000	171.509995	178.102997	178.102997	0.000000e+00
25%	606.396973	609.260986	604.109985	606.718994	606.718994	7.991080e+07
50%	6301.569824	6434.617676	6214.220215	6317.609863	6317.609863	5.098183e+09
75%	10452.399414	10762.644531	10202.387695	10462.259766	10462.259766	2.456992e+10
max	67549.734375	68789.625000	66382.062500	67566.828125	67566.828125	3.509679e+11

In [9]:
```
print('Null Values:',maindf.isnull().values.sum())
```

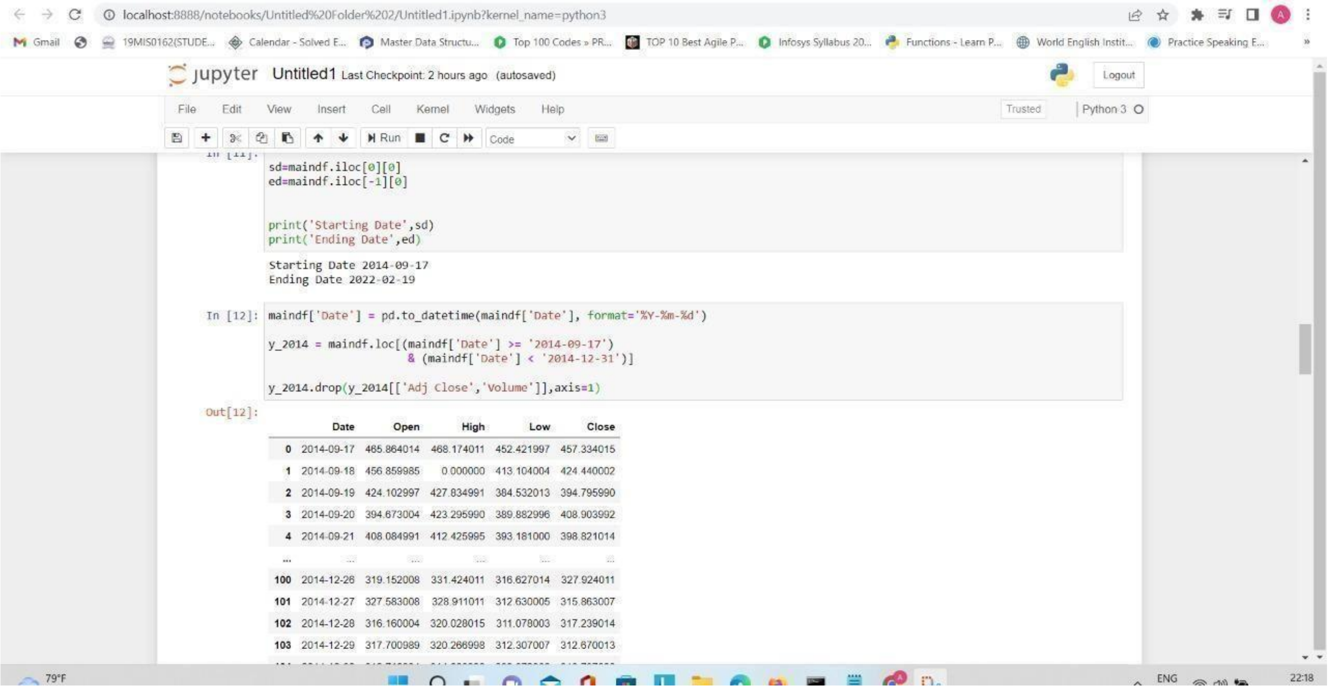
```
Null Values: 0
```

In [10]:
```
print('NA values:',maindf.isnull().values.any())
```

```
NA values: False
```

TEST RESULTS

Analyzing the data of Year 2014



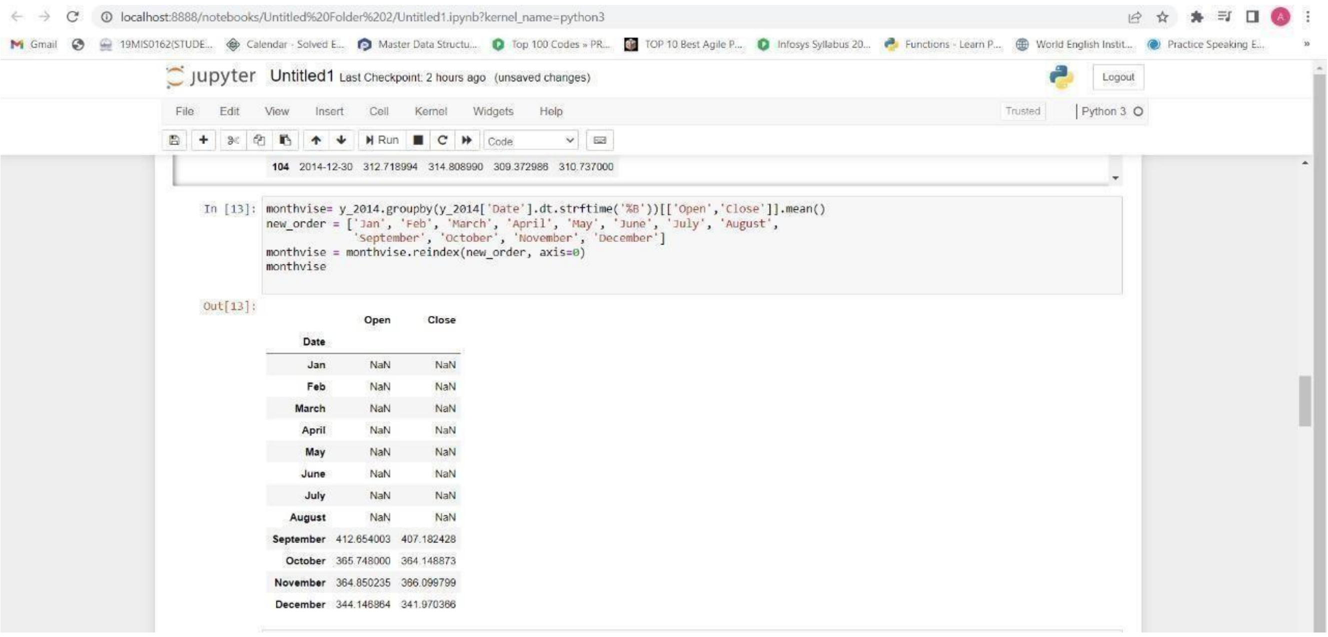
```
In [11]:
sd=maindf.iloc[0][0]
ed=maindf.iloc[-1][0]


print('Starting Date',sd)
print('Ending Date',ed)
```

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

```
In [12]: 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)
```

Out[12]:

	Date	Open	High	Low	Close
0	2014-09-17	465.864014	468.174011	452.421997	457.334015
1	2014-09-18	456.859985	0.000000	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



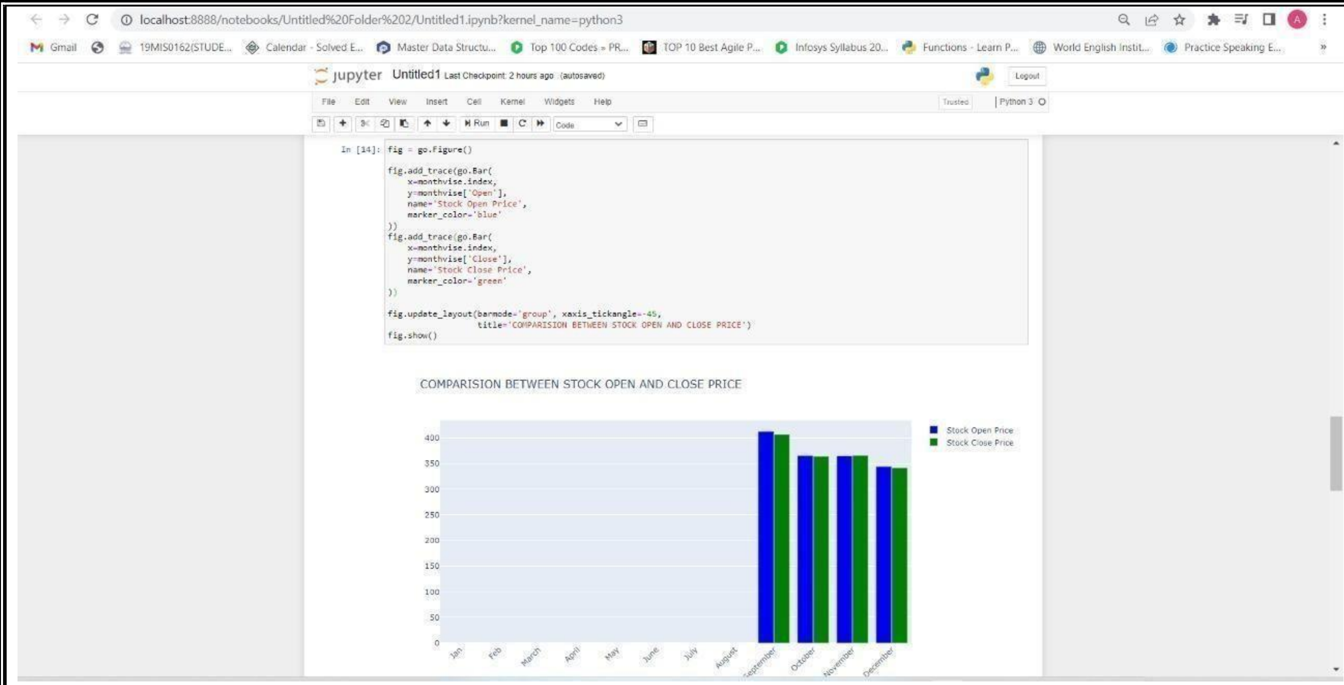
104	2014-12-30	312.718994	314.808990	309.372986	310.737000

```
In [13]: monthvise= y_2014.groupby(y_2014['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['Jan', 'Feb', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

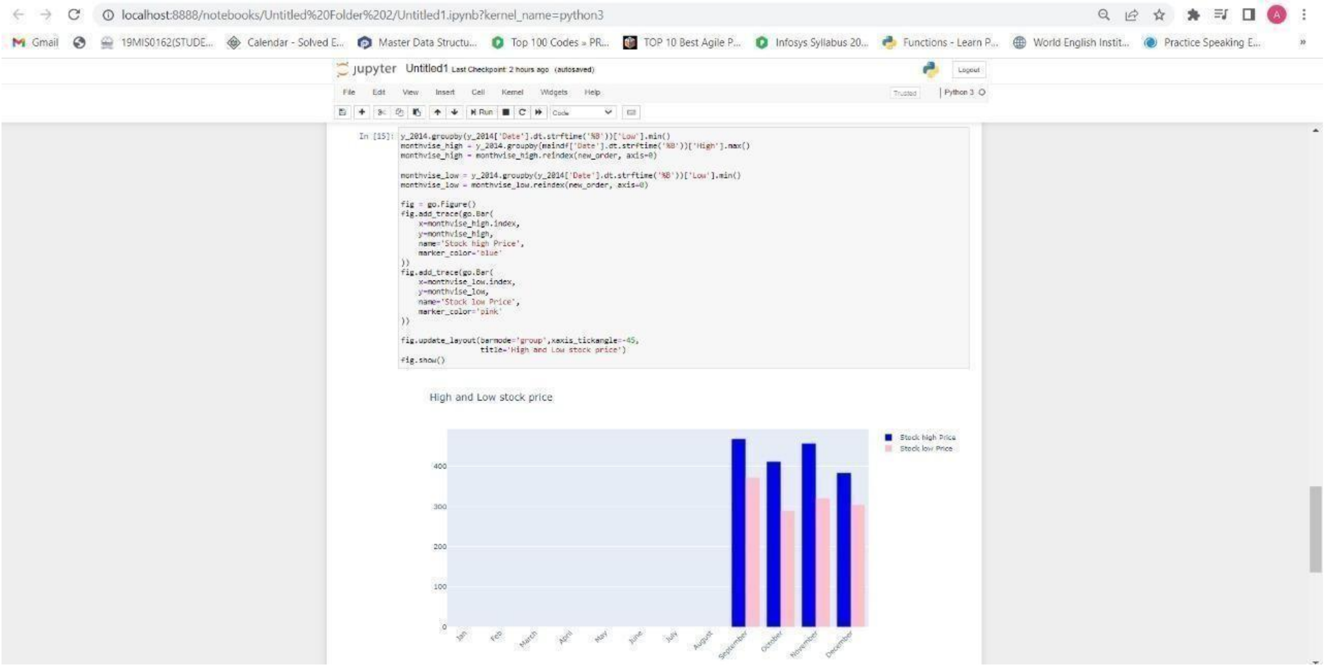
Out[13]:

Date	Open	Close
Jan	NaN	NaN
Feb	NaN	NaN
March	NaN	NaN
April	NaN	NaN
May	NaN	NaN
June	NaN	NaN
July	NaN	NaN
August	NaN	NaN
September	412.654003	407.182428
October	365.748000	364.148873
November	364.850235	366.099799
December	344.146864	341.970366

Monthwise comparision between Stock open price and stock close price for the year 2014



Month wise High and Low stock price for 2014 year



Stock analysis chart

```
In [16]: 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_2014['Low']],
              labels={'Date': 'Date','value':'Stock value'})
fig.update_layout(title_text='Stock analysis chart', font_size=15, font_color='blue',legend_title_text='Stock Parameters')
fig.for_each_trace(lambda t:  t.update(name = next(names)))
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)

fig.show()
```


```
In [ ]:
```

Analyzing the data of year 2015

```
In [17]: 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)
```

Out[17]:

	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
110	2015-01-05	265.084015	278.341003	265.084015	274.473999
...	...	...	...	...	...
466	2015-12-27	416.514008	424.006989	408.882996	422.822998
467	2015-12-28	423.342987	429.769012	418.480988	422.278992
468	2015-12-29	422.097992	432.983002	420.627014	432.983002
469	2015-12-30	433.299988	434.386993	422.084015	426.619995
470	2015-12-31	425.875000	432.920990	418.734985	430.566986

365 rows × 5 columns

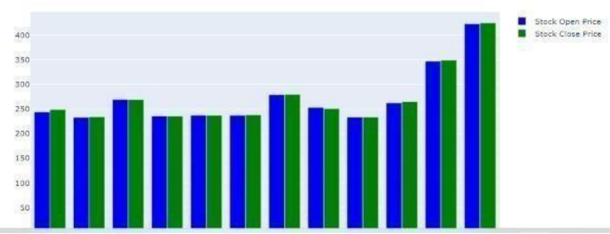
```
In [18]: monthvise= y_2015.groupby(y_2015['Date'].dt.strftime('%B'))[['Open','Close']].mean()
new_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
             'September', 'October', 'November', 'December']
monthvise = monthvise.reindex(new_order, axis=0)
monthvise
```

Out[18]:

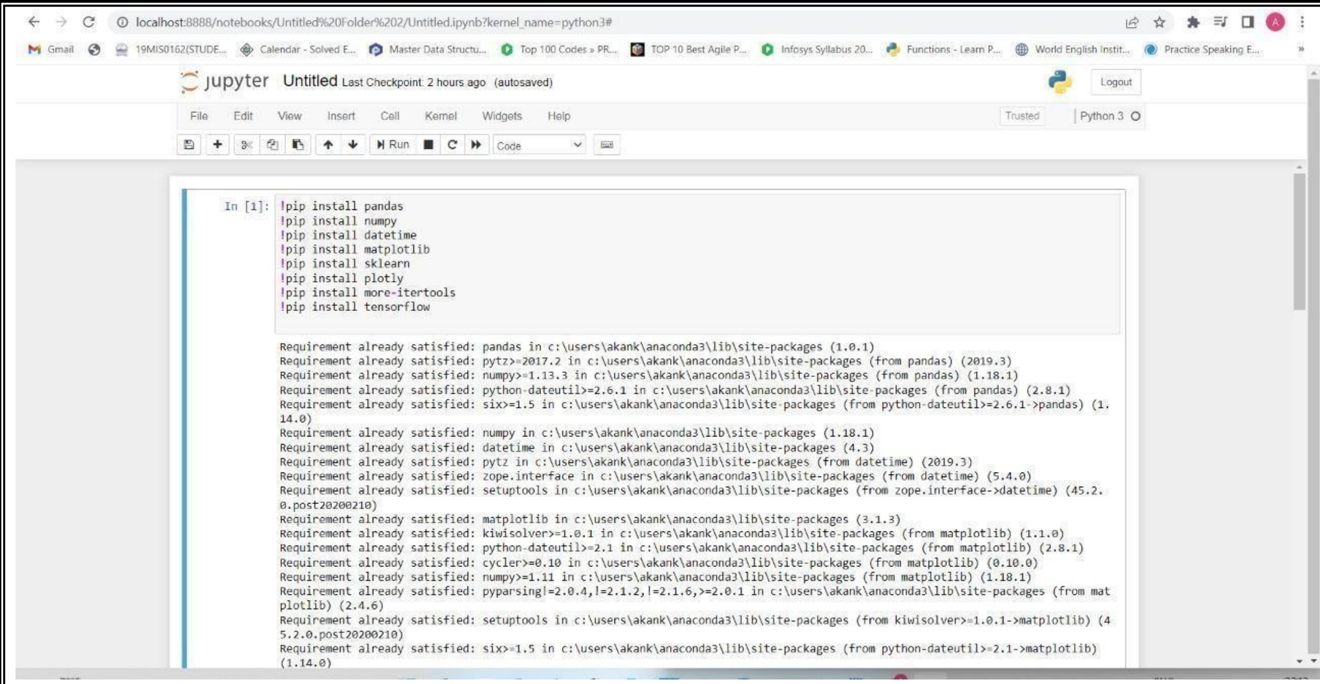
Date	Open	Close
January	243.820808	248.782547
February	232.821856	234.153645
March	269.278419	269.042259
April	235.708667	235.491534
May	237.161806	236.997001
June	236.941433	238.081766
July	278.857679	279.563740
August	252.986774	250.733805
September	233.486733	233.595533
October	262.306000	264.855356
November	346.866833	348.883332
December	422.618033	424.464547

Month wise comparision between Stock open price and stock close price for the year 2015

```
In [20]: fig = go.Figure()

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

fig.update_layout(barmode='group', xaxis_tickangle=-45,
                  title='comparision between Stock open and close price in 2015')
fig.show()
```

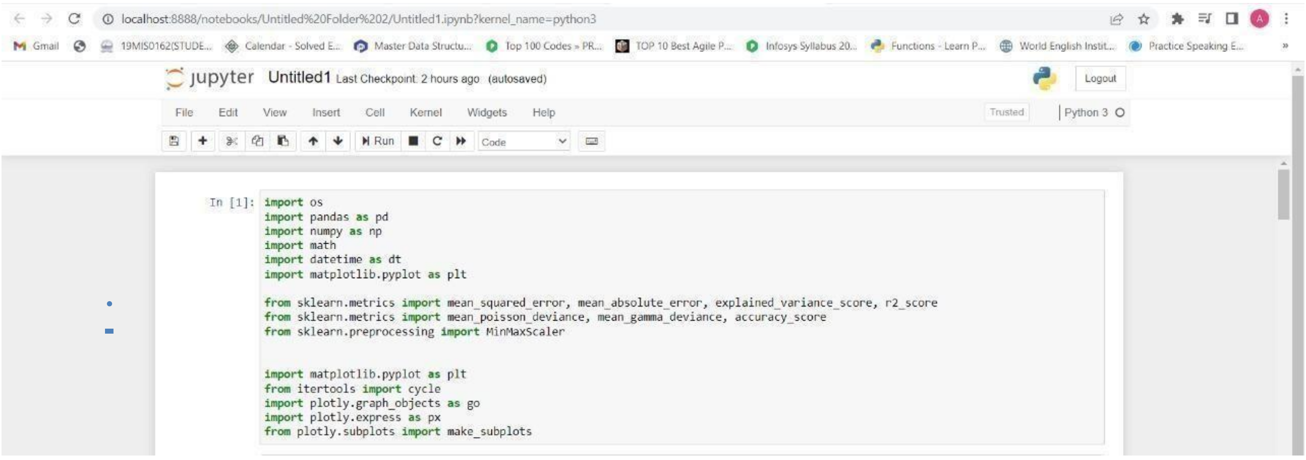



```
In [22]: y_2015.groupby(y_2015['Date'].dt.strftime('%B'))['Low'].min()
monthvise_high = y_2015.groupby(maindf['Date'].dt.strftime('%B'))['High'].max()
monthvise_high = monthvise_high.reindex(new_order, axis=0)

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='pink'
))
fig.add_trace(go.Bar(
    x=monthvise_low.index,
    y=monthvise_low,
    name='Stock low Price',
    marker_color='green'
))

fig.update_layout(barmode='group',xaxis_tickangle=-45,
                  title=' Monthwise High and Low stock price')
fig.show()
```



```
localhost:8888/notebooks/Untitled%20Folder%202/Untitled.ipynb?kernel_name=python3#
jupyter Untitled Last Checkpoint: 2 hours ago (autosaved)

In [1]: !pip install pandas
        !pip install numpy
        !pip install datetime
        !pip install matplotlib
        !pip install sklearn
        !pip install plotly
        !pip install more-itertools
        !pip install tensorflow

Requirement already satisfied: pandas in c:\users\akank\anaconda3\lib\site-packages (1.0.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\akank\anaconda3\lib\site-packages (from pandas) (2019.3)
Requirement already satisfied: numpy>=1.13.3 in c:\users\akank\anaconda3\lib\site-packages (from pandas) (1.18.1)
Requirement already satisfied: python-dateutil>=2.6.1 in c:\users\akank\anaconda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: six>=1.5 in c:\users\akank\anaconda3\lib\site-packages (from python-dateutil>=2.6.1->pandas) (1.
14.0)
Requirement already satisfied: numpy in c:\users\akank\anaconda3\lib\site-packages (1.18.1)
Requirement already satisfied: datetime in c:\users\akank\anaconda3\lib\site-packages (4.3)
Requirement already satisfied: pytz in c:\users\akank\anaconda3\lib\site-packages (from datetime) (2019.3)
Requirement already satisfied: zope.interface in c:\users\akank\anaconda3\lib\site-packages (from datetime) (5.4.0)
Requirement already satisfied: setuptools in c:\users\akank\anaconda3\lib\site-packages (from zope.interface->datetime) (45.2.
0.post20200210)
Requirement already satisfied: matplotlib in c:\users\akank\anaconda3\lib\site-packages (3.1.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\akank\anaconda3\lib\site-packages (from matplotlib) (1.1.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\akank\anaconda3\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\akank\anaconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: numpy>=1.11 in c:\users\akank\anaconda3\lib\site-packages (from matplotlib) (1.18.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\akank\anaconda3\lib\site-packages (from mat
plotlib) (2.4.6)
Requirement already satisfied: setuptools in c:\users\akank\anaconda3\lib\site-packages (from kiwisolver>=1.0.1->matplotlib) (4
5.2.0.post20200210)
Requirement already satisfied: six>=1.5 in c:\users\akank\anaconda3\lib\site-packages (from python-dateutil>=2.1->matplotlib)
(1.14.0)
```

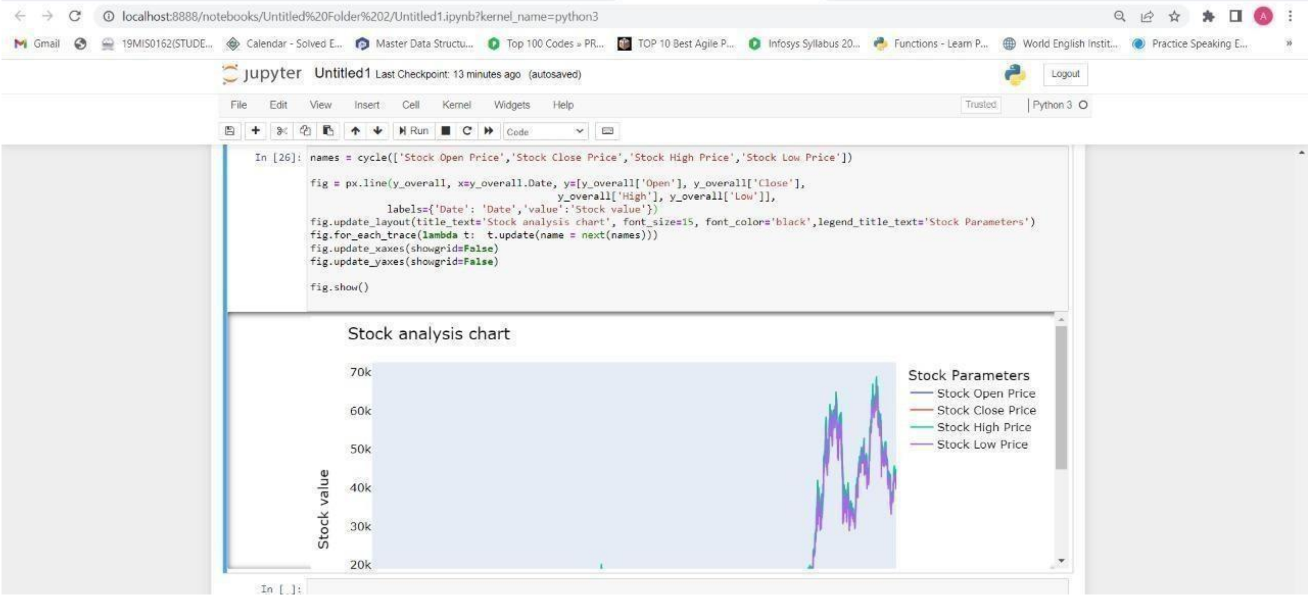
Like this we will do for all the remaining years and produce the stock analysis chart for everyyear.

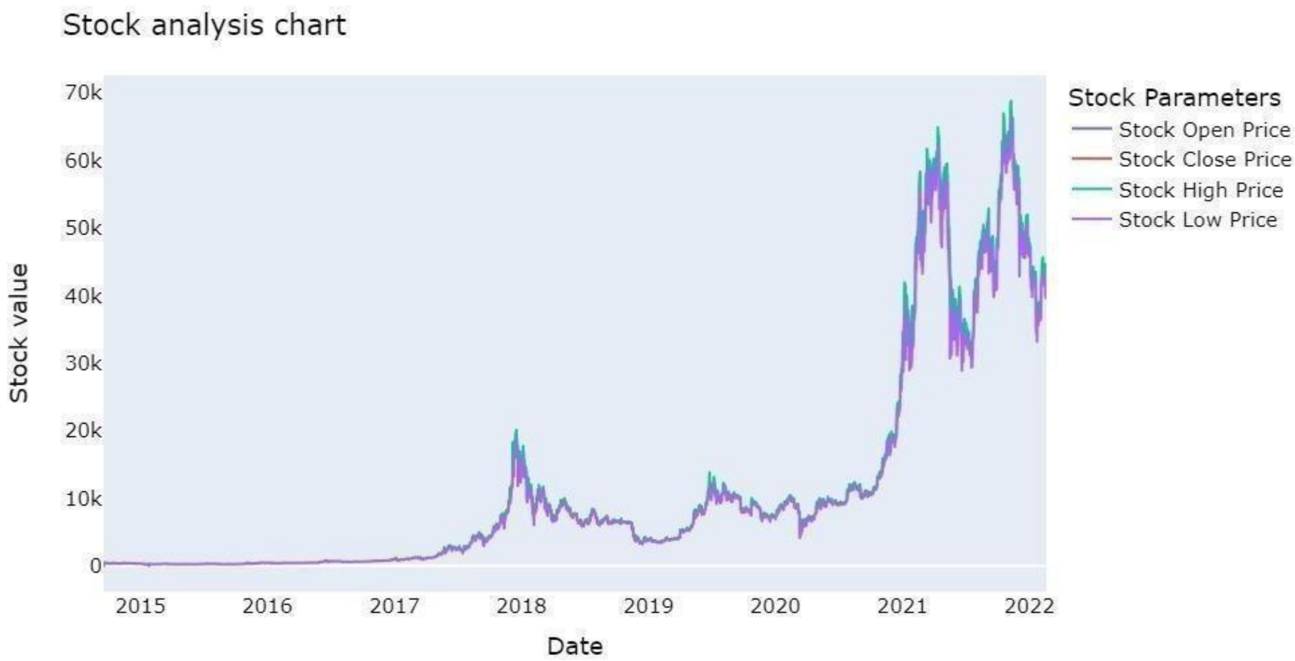


```
localhost:8888/notebooks/Untitled%20Folder%202/Untitled1.ipynb?kernel_name=python3
jupyter Untitled1 Last Checkpoint: 2 hours ago (autosaved)

In [1]: import os
        import pandas as pd
        import numpy as np
        import math
        import datetime as dt
        import matplotlib.pyplot as plt

        from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score, r2_score
        from sklearn.metrics import mean_poisson_deviance, mean_gamma_deviance, accuracy_score
        from sklearn.preprocessing import MinMaxScaler


        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
```

```
        import plotly.express as px
        from plotly.subplots import make_subplots

In [2]: maindf=pd.read_csv("BTC-USD.csv")
```


```
        import plotly.express as px
        from plotly.subplots import make_subplots

In [2]: maindf=pd.read_csv("BTC-USD.csv")
```





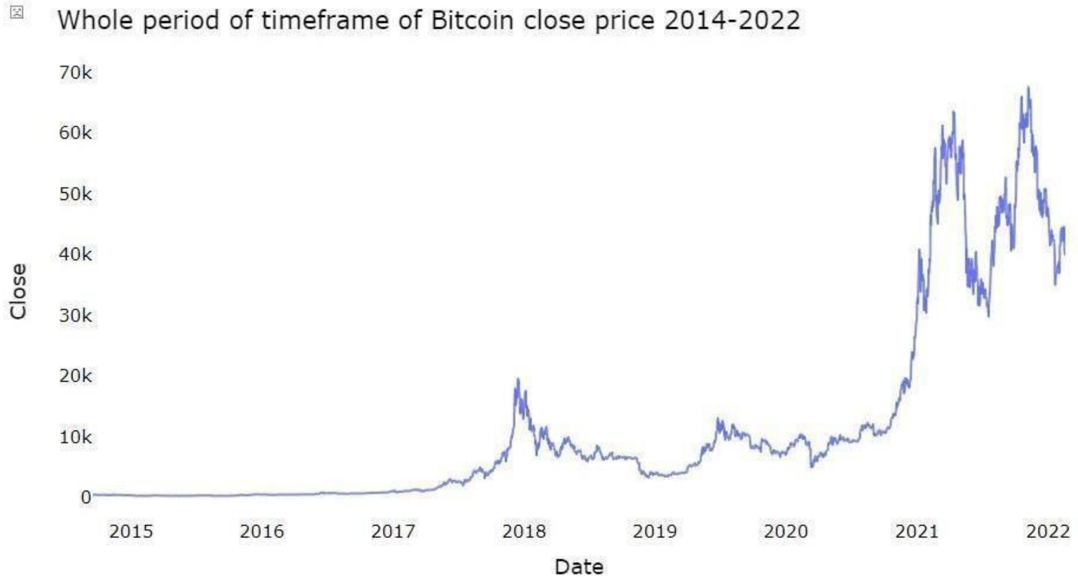
5. RESULTS AND DISCUSSION

Whole period of time frame of bitcoin close prediction 2014 to 2022

In [60]:
```
closedf = maindf[['Date','Close']]
print("Shape of close dataframe:", closedf.shape)
```

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

In [61]:
```
fig = px.line(closedf, x=closedf.Date, y=closedf.Close,labels={'date':'Date','close':'Close Stock'})
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', plot_bgcolor='white',
                  font_size=15, font_color='black')
fig.update_xaxes(showgrid=False)
fig.update_yaxes(showgrid=False)
fig.show()
```



In [62]:
```
closedf = closedf[closedf['Date'] > '2021-02-19']
close_stock = closedf.copy()
print("Total data for prediction: ",closedf.shape[0])
```

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

In [63]:
```
closedf
```

Out[63]:

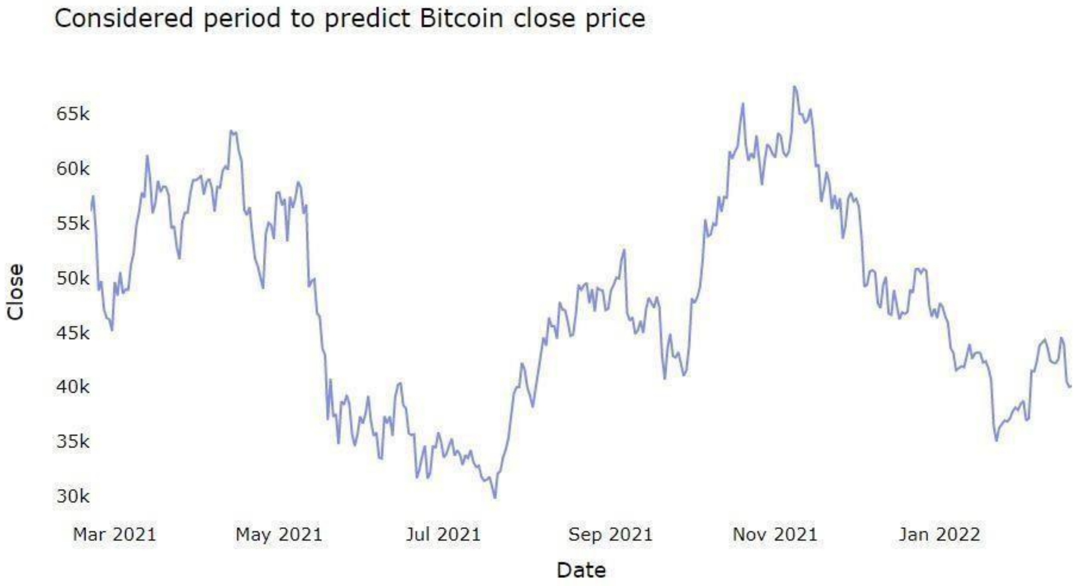
	Date	Close
2348	2021-02-20	56099.519531
2349	2021-02-21	57539.945313
2350	2021-02-22	54207.320313
2351	2021-02-23	48824.425781
2352	2021-02-24	49705.332031
...	...	...
2708	2022-02-15	44575.203125
2709	2022-02-16	43961.859375
2710	2022-02-17	40538.011719
2711	2022-02-18	40030.976563
2712	2022-02-19	40126.429688

365 rows × 2 columns

In [64]:
```
fig = px.line(closedf, x=closedf.Date, y=closedf.Close,labels={'date':'Date','close':'Close Stock'})
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()
```

Showing the price prediction between 2021 to 2022

In [64]:
```
fig = px.line(closedf, x=closedf.Date, y=closedf.Close,labels={'date':'Date','close':'Close Stock'})
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()
```



In [65]:
```
del closedf['Date']
scaler=MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))
print(closedf.shape)
```

```
(365, 1)
```

In [66]:
```
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)
```

In [67]:
```
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)
```

In [68]:
```
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,)
```

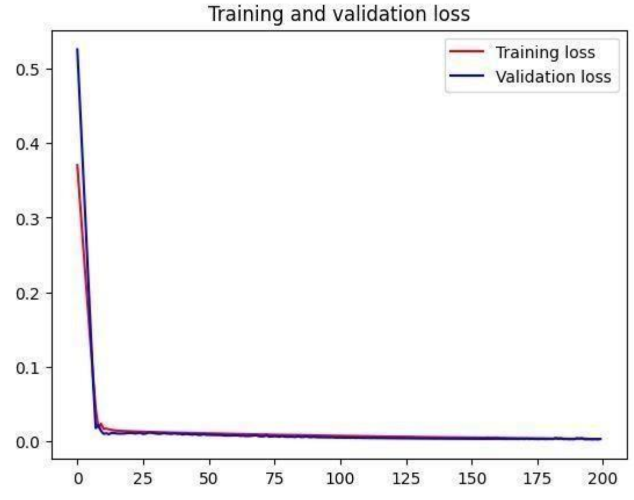
```
In [69]: X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
         X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

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

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

```
In [70]: model=Sequential()

         model.add(LSTM(10,input_shape=(None,1),activation="relu"))

         model.add(Dense(1))

         model.compile(loss="mean_squared_error",optimizer="adam")
```

```
In [71]: history = model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=200,batch_size=32,verbose=1)

Epoch 1/200
7/7 [==============================] - 5s 156ms/step - loss: 0.3703 - val_loss: 0.5255
Epoch 2/200
7/7 [==============================] - 0s 28ms/step - loss: 0.3212 - val_loss: 0.4514
Epoch 3/200
7/7 [==============================] - 0s 29ms/step - loss: 0.2734 - val_loss: 0.3782
Epoch 4/200
7/7 [==============================] - 0s 28ms/step - loss: 0.2258 - val_loss: 0.3054
Epoch 5/200
7/7 [==============================] - 0s 29ms/step - loss: 0.1798 - val_loss: 0.2311
Epoch 6/200
7/7 [==============================] - 0s 29ms/step - loss: 0.1315 - val_loss: 0.1575
Epoch 7/200
7/7 [==============================] - 0s 29ms/step - loss: 0.0863 - val_loss: 0.0809
Epoch 8/200
7/7 [==============================] - 0s 30ms/step - loss: 0.0425 - val_loss: 0.0177
Epoch 9/200
7/7 [==============================] - 0s 29ms/step - loss: 0.0183 - val_loss: 0.0219
Epoch 10/200
```

Plotting Loss vs Validation loss

```
In [72]: import matplotlib.pyplot as plt

         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 640x480 with 0 Axes>

```
In [73]: train_predict=model.predict(X_train)
         test_predict=model.predict(X_test)
         train_predict.shape, test_predict.shape

         7/7 [==============================] - 1s 7ms/step
         5/5 [==============================] - 0s 6ms/step

Out[73]: ((203, 1), (130, 1))
```

```
In [74]: 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))
```

```
In [75]: 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:  2166.9721052534032
         Train data MSE:  4695768.104946367
         Train data MAE:  1707.9846155732755
         -------------------------------------------------------------------------------------
         Test data RMSE:  2101.41386834123
         Test data MSE:  4415940.246056854
         Test data MAE:  1596.6689603999996
```

```
In [76]: 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: 0.9487940297392466
         Test data explained variance regression score: 0.9548027995355342
```

```
In [77]: 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: 0.9460307188939452
         Test data R2 score: 0.9458723812953516
```

```
         ----------------------------------------------------------------------
         Train data MPD:  103.21640664928002
         Test data MPD:  85.37983524854297
```

```
In [79]:
         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 price'])


         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_close'],
                                                   plotdf['test_predicted_close']],
                       labels={'value':'Stock price','date': 'Date'})
         fig.update_layout(title_text='Comparision between original close price vs predicted close price',
                           plot_bgcolor='white', font_size=15, font_color='black', legend_title_text='Close Price')
         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)
```

Comparision of original bitcoin price vs predicted price

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

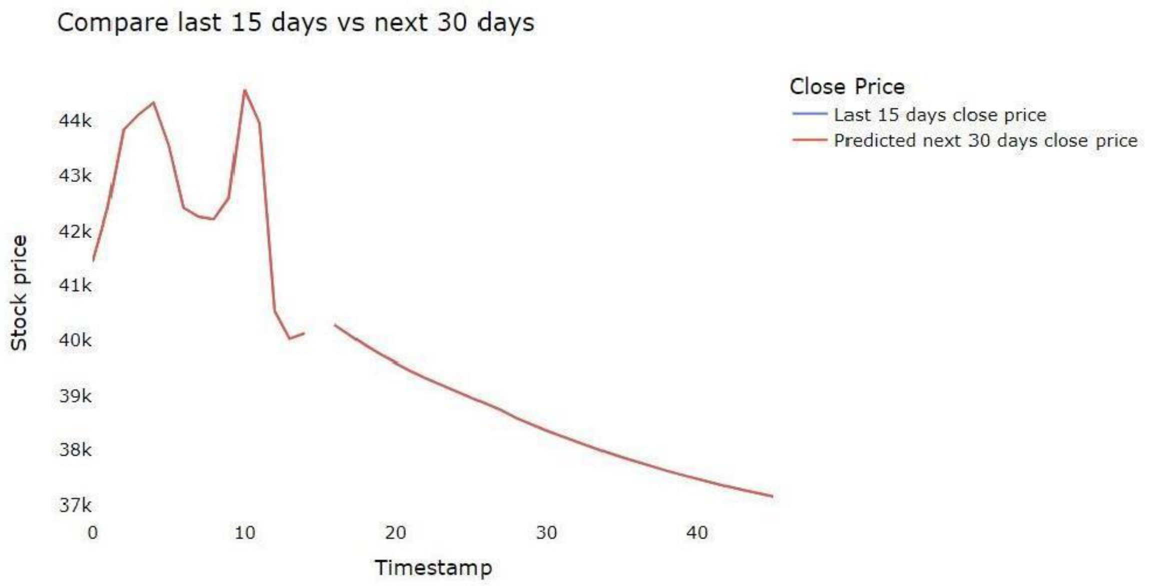
In [80]:
```
x_input=test_data[len(test_data)-time_step:].reshape(1,-1)
temp_input=list(x_input)
temp_input=temp_input[0].tolist()

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
```

```
In [81]: 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]
```

```
In [82]: 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)-time_step:]).reshape(1,-1).tolist()[0]
         next_predicted_days_value[time_step+1:] = scaler.inverse_transform(np.array(lst_output).reshape(-1,1)).reshape(1,-1).tolist()[0]

         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_value'],
                                                               new_pred_plot['next_predicted_days_value']],
                       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='Close Price')

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


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

Plotting whole closing price with prediction

In [83]:
```
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='Stock')

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

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

Here we have trained LSTM model on the taken dataset for generating the predictions ofbitcoin prices.From the dataset,we have seen that the highest price of bitcoin is in betweenoct20-nov1,2021.In this we are going to find or predict the dates after that.Finally,we are generating graphs for the entire prediction of data.

6.CONCLUSION

LSTM are excellent technologies and have great architectures that can be used to analyze and predict time-series information.The LSTM model,which is implementedhere for the purpouse of bitcoin price prediction.Here we have taken only few features that affect price.So,to increase effeciency,we have to take more features.

7.REFERENCES

[1] T. Phaladisailoed, and T. Numnoda, “Machine Learning Models Comparison for Bitcoin Price Prediction,”10thInternational Conference on Information Technology and Electrical Engineering, 2018.

[2] Neha Mangla, Akshay Bhat, Ganesh Avarbratha, and Narayana Bhat, “Bitcoin Price Prediction Using MachineLearning,” International Journal of Information and Computer Science, Volume 6, Issue 5, May 2019.

[3] Q. Guo, S. Lei, Q. Ye, Z. Fang “MRC-LSTM: A Hybrid Approach of Multi-scale Residual CNN and LSTM toPredict Bitcoin Price,” MDPI, May 2021.

[4] T. Awoke, M. Rout, L. Mohanty, S. C. Satapathy, “Bitcoin Price Prediction and Analysis Using Deep LearningModels,” ResearchGate.

[5] A. Rana, R. Kachchhi, J. Baradia, V. Shelke “Stock Market Prediction Using Deep Learning” InternationalResearch Journal of Engineering and Technology, Volume 8, Issue 4, April 2021.