ARTIFICIAL INTELLIGENCE

STOCK MARKET PREDICTION SYSTEM

**Submitted to:**

**Ma’am M. Salas Akbar**

**Ma’am Aniqa Naeem**

**Group members**

* Ibrahim Abdul Khaliq (02-134181-162)
* Ameer Hamza (02-134181-032)
* M. Farrukh Ali (02-134181-161)

Contents

[ABSTRACT 3](#_Toc60549595)

[Introduction 3](#_Toc60549596)

[Prediction Concept 4](#_Toc60549597)

[False prediction 4](#_Toc60549598)

[Conditions because of which predictions can be false 4](#_Toc60549599)

[Libraries 5](#_Toc60549600)

[For Dataset analysis & Graph plotting 5](#_Toc60549601)

[For Linear Regression 5](#_Toc60549602)

[For LSTM Model 6](#_Toc60549603)

[Dataset 6](#_Toc60549604)

[Methodology 7](#_Toc60549605)

[Dataset Analysis: 7](#_Toc60549606)

[Linear Regression 7](#_Toc60549607)

[Steps of using linear Regression model on our datasets 7](#_Toc60549608)

[LSTM (Long short-term memory) 10](#_Toc60549609)

[Steps of using LSTM model on the datasets 10](#_Toc60549610)

[CODE 13](#_Toc60549611)

[Data Exploration 13](#_Toc60549612)

[Lowest close of all the stocks ? 14](#_Toc60549613)

[Google Open and Closing Shares 17](#_Toc60549614)

[Facebook Opening and Closing Shares 18](#_Toc60549615)

[Amazon Opening and Closing Shares 19](#_Toc60549616)

[Total Traded 21](#_Toc60549617)

[Correlation of Close stocks 23](#_Toc60549618)

[Daily Percentage Change 24](#_Toc60549619)

[Cumulative Return 29](#_Toc60549620)

[Linear Regression 30](#_Toc60549621)

[Using LSTM For prediction 36](#_Toc60549622)

[Predict 39](#_Toc60549623)

[Conclusion 44](#_Toc60549624)

# ABSTRACT

Artificial Intelligence has advanced impressively on account that inventors started out tampering with its capability. Many consider that the use for AI is best in the field of financial market hypothesis. Technology may be used both to make our lives higher or make cash. The stock alternate marketplace is the most volatile and most dynamic of all. unique care must be exercised in shopping for and selling of shares from different businesses or companies. The chance of losing the stocks and acquiring blessings via the shares are less, if not careful. Artificial Intelligence has the capability to predict the company’s profit margin and financial issue. In this project we are building an AI program that help in predicting the share price of the major company’s such as Amazon, Google, and Facebook. And we will be training and testing our model using linear Regression as well as the LSTM (Long Short-Term Memory) model.

# Introduction

There are many factors to keep in mind when predicting the stock price, because stock price market is not only influenced by the performance of the company but also by the sentimental and the support of the major shareholders of the company. Stock market prediction is certainly a difficult undertaking due to the fact do’s and don’ts are very applicable issue on this connection. Pulling out profit from stock marketplace prediction at the moment are genuinely possible with the advent of artificial intelligence thru which researchers may additionally strive with different strategies In reference to economic records which substantially performs a essential position for generating an awesome selection on the idea of the available benchmarks. AS an investor it’s a bold assignment to totally depend upon a brand-new technology like Artificial Intelligence (AI) for the prediction of stock market. Stock trading ability can be computed with economic commercial enterprise out of the available assets that is primarily based on monetary records. Fear, greed, threats, and all different human emotions shall never play any sort of role at the same time as predicting stock market trends within the case of Artificial Intelligence (AI). Algorithmic technique for growing A complete package deal based at the predictions as well as preceding facts evaluation at the moment are trending Only due to the fact the system dependency of humans. Virtually infinite opportunities for acquiring increasingly profits out of the invested cash are the high concerns of an individual’s and that is what AI now are doing with algorithmic trading.

Stock Market is a place where buying and selling of a share take place for listed companies. Stock exchange is basically a bridge between buyers and sellers of the shares*.*It helps companies to improve their financial condition. Help individuals to create personal wealth. It serves as a symbol of states economic status. Increase investment. It helps to determine the future value of any company stock and other financial issues.

# Prediction Concept

The possible market prediction goal may be the future stock rate or the volatility of the expenses or market trend. The usage of this prediction developer holds two exceptional varieties of predictions like Dummy and actual time predictions i.e., used in stock market system. In Dummy prediction we outline some policies and are predict the future of stocks by way of calculating common fee. The prediction method that we are going to use is the AI recurrent neural network called Long Short-term Memory (LSTM) to predict the closing percentage of the organization using their previously recorded data.

# False prediction

It is possible for a program to predict wrong because it is not working on real time it is based on previous records so it might be possible that on the time of prediction share price of a xyz company is high as per system prediction according to previous record but on ground the reality might be different.

# Conditions because of which predictions can be false

* Country’s financial condition at the time of record gathering and at the time of prediction.
* Market environment at the time of records collection and at the time of prediction making.
* Country’s overall situation of peace.
* The condition of a company financial status is most important for accuracy because if the financial condition of a company was good at the time of record gathering and now at the time of prediction it becomes poor than our program is working on the basis of records so it will predict company’s share prices will go up but according to ground realities it will go down. So, the marketing sense of a trader is also important because our system can only predict according to previous facts and figures it does not work on real time.
* Many other reasons.

# Libraries

## For Dataset analysis & Graph plotting

|  |  |
| --- | --- |
| **Import** | **Purpose** |
| Plotly | For implementing linear regression graphs |
| Pandas | For data reading and analysis |
| matplotlib.pyplot | Allow the implementation of MATLAB functionalities in the code |
| datetime | For converting the dataset values into datetime format |
| scatter\_matrix | For implementing dataset analysis graph |
| seaborn | It gives an elevated level interface to drawing appealing and educational factual designs. |
| numpy | adding support for large, multi-dimensional arrays |
| plotly.graph\_objs | For designing the layout of a graph and display it beautifully |
|  |  |

## For Linear Regression

|  |  |  |
| --- | --- | --- |
| **Import** | **From** | **Purpose** |
| Train\_test\_split | Sklearn.model\_selection | For building a linear regression model |
| MinMaxScaler | Sklearn.preprocessing | For preprocessing the dataset analysis |
| StandardScaler | Sklearn.preprocessing | For preprocessing the scaling of datasets |
| R2\_score | Sklearn.metrics | Used for model evaluation and for finding the accuracy |
| Mean\_square\_error | Sklearn.metrics | Used for model evaluation and for finding the accuracy |
| LinearRegression | Sklearn.linear\_model | For apply linear Regression formula on a dataset |

## For LSTM Model

|  |  |  |
| --- | --- | --- |
| **Import** | **From** | **Purpose** |
| Sequential | tensorflow.keras.models | For building a LSTM model |
| Dense | tensorflow.keras.layers | For adding Dense layers for the training model |
| LSTM | tensorflow.keras.layers | For adding LSTM layers for the training model |
| tensorflow |  | To implement the machine learning. |

# Dataset

For data collection we used yahoo finance database. Yahoo database contains stock prices for various companies. The companies that we chose are Google, Amazon, and Facebook. we collected data from January 2016 to January 2021. Although training our model using this data would cause our model to be less accurate because of a lack of trend during the crisis period.

The dataset contains stock prices of the following companies:

* Google
* Amazon
* Facebook

For that sector we tried to predict using stock prices. These are all listed on the same index. The stock data obtained from yahoo are as follows:

* Date
* Open
* High
* Low
* Close
* Adj Close
* Volume

Other parameters calculated for input dataset are as listed below.

* Month
* Total traded
* Return
* Cumulative Return

# Methodology

## Dataset Analysis:

## Linear Regression

In this project linear regression is used as a supervised learning algorithm to predict the results of a contribution. Regression lines gives the predict results such as shares sold based on a given features. It is a very powerful statistical technique used for solving machine learning problems. It can be used to predict the total revenue of the organization and also predict the total unit expected to be sold for a certain product. The regression line gives the predicted result i.e., unit sold based on an input features the formula is as following.

Y=MX+C

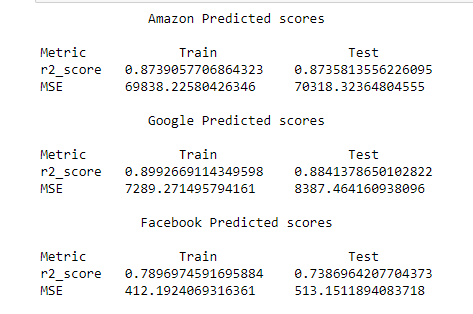
Where m is slope of the line regression and c is the intercept of the regression line. Linear Regression allow nus to predict the total revenue of the company. Major features of linear regression are whether prediction and Stock price prediction.

### Steps of using linear Regression model on our datasets

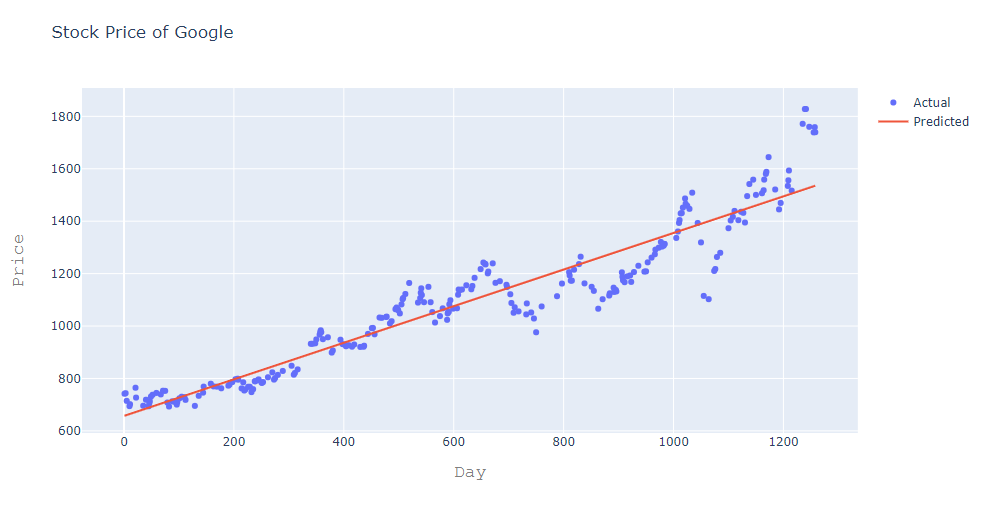
After the complete dataset analysis, we are going to import all the above library that are mentioned in the regression table. Then we are going to split the data into two parts i.e., train and test sets. Keeping in mind that the X variable is the independent variable, and the Y variable is the dependent variable or the target variable i.e., the ‘Close’ column.

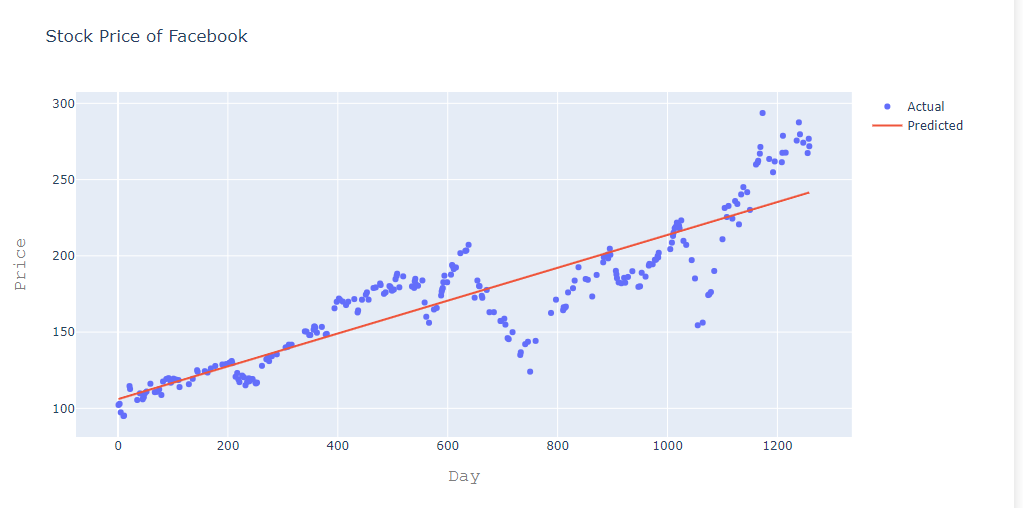
After splitting the dataset, we are going to perform the scaling feature on the X variable’s train datasets using the standard Scaler to fit the X variable train data. The standardize scaler is mandatory for almost all the machine learning algorithm, if the split data is not scaled and fit then they will behave badly that is if they are not distributed properly.

Now create a linear regression model and pass x train variable along with the y train variable and perform the linear regression on the training data. After applying the linear regression, we are going to calculate the scores for model evaluation in other word we are going to find the accuracy of the train and test models by using the r-square error and the mean squared error.









## 

## LSTM (Long short-term memory)

The second methodology which is used in this project is LSTM. LSTM stands for Long Short-Term memory. It is AI recurrent neural network. LSTM blocks are used to build a recurrent neural community. An RNN is a sort of neural network wherein the output of a block is fed as enter to the subsequent iteration. An LSTM block consists of 4 main additives: a cell, an enter gate, an output gate and a forget gate. The cell is accountable for "remembering" values over arbitrary time durations; consequently, the phrase "memory" in LSTM. Each of the three gates can be idea of as a "conventional" synthetic neuron, as in a multi-layer (or feedforward) neural community: that is, they compute an activation (the usage of an activation feature) of a weighted sum. Intuitively, they can be concept as regulators of the float of values that goes thru the connections of the LSTM; therefore, the denotation "gate". There are connections between those gates and the cell. Some of the connections are recurrent, a few them are not.

### Steps of using LSTM model on the datasets

LSTM model can be achieved by going through the following three steps:

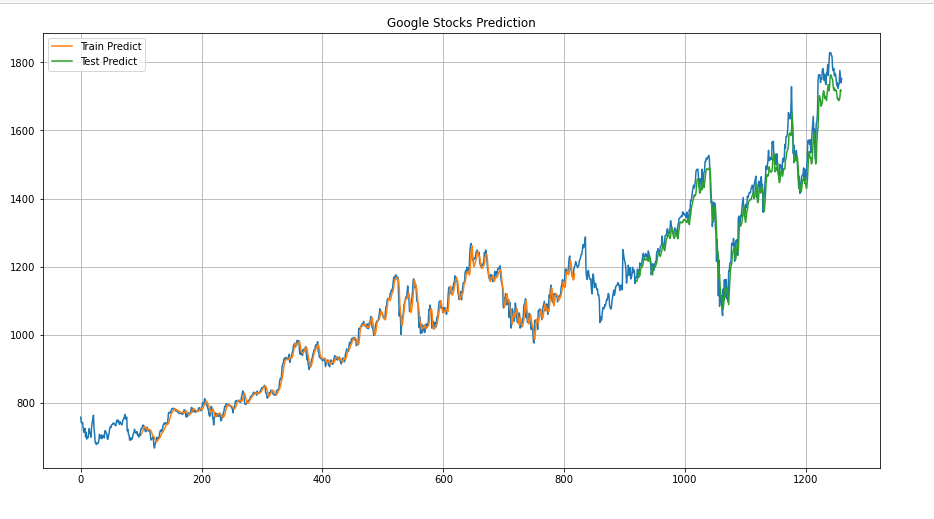
The initial step is to choose which data is to be discarded from the cell in that time step. It is decided by the help of sigmoid function. It looks at the previous state that is Xt-1 and the current state that is Xt and compute the function accordingly.

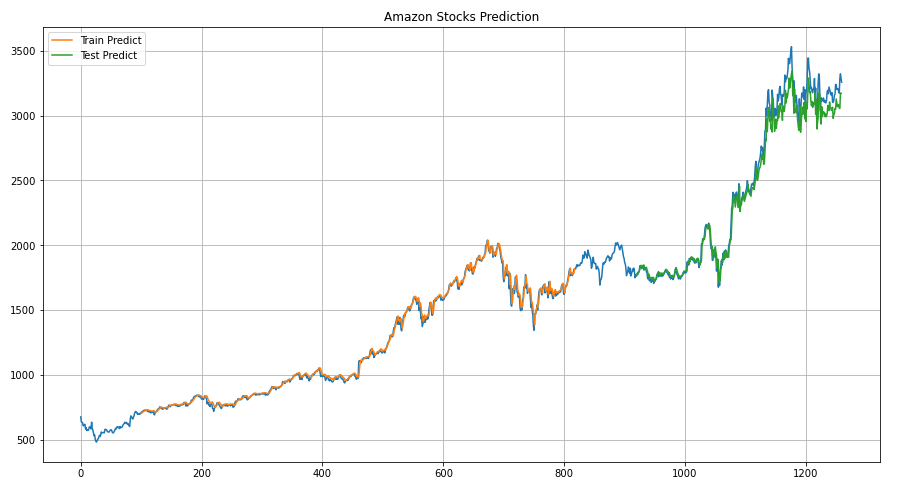
Sigmoid layer contains two parts.

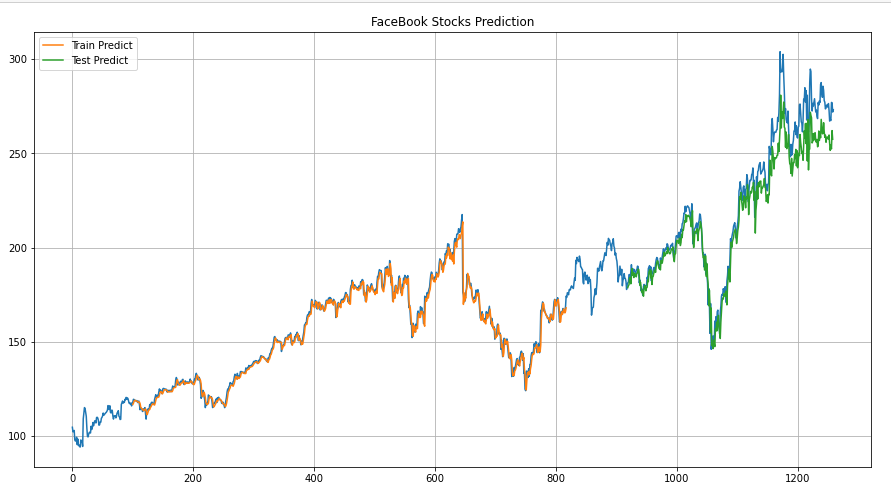
1. Sigmoid function
2. Hyperbolic tangent function

In sigmoid function it decides which value to let through that is 0 or 1. The hyperbolic tangent function layer is used to give weightage to the values which are the values of past and decides the importance of the value from -1 and 1

At the third step, it decides which will be the final output and to achieve that it will first run the sigmoid function layer and decide which path of the cell stayed and made it to the output then we put the cell state though the hyperbolic tangent function layer to push the values between -1 and +1 and multiply it by the sigmoid state value and will then achieve the output.







# 

# CODE

## Data Exploration

In [1]:

**import** **pandas\_datareader.data** **as** **web**

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**import** **datetime**

%**matplotlib** inline

**from** **pandas.plotting** **import** scatter\_matrix

**import** **seaborn** **as** **sns**

In [2]:

start=datetime.datetime(2016,1,1)

end=datetime.datetime(2021,1,1)

In [3]:

amazon=web.DataReader("AMZN","yahoo",start,end)

google=web.DataReader("GOOG","yahoo",start,end)

facebook=web.DataReader("fb","yahoo",start,end)

In [4]:

amazon.head()

google.head()

facebook.head()

In [5]:

amazon.isna().sum()

Out[5]:

High 0

Low 0

Open 0

Close 0

Volume 0

Adj Close 0

dtype: int64

In [6]:

google.isna().sum()

In [7]:

facebook.isna().sum()

In [9]:

facebook.to\_csv("fb\_stocks.csv")

amazon.to\_csv("amazon\_stocks.csv")

google.to\_csv("google\_stocks.csv")

## Lowest close of all the stocks ?

In [10]:

facebook[facebook['Close']==facebook['Close'].min()]

In [11]:

amazon[amazon['Close']==amazon['Close'].min()]

In [12]:

google[google['Close']==google['Close'].min()]

In [13]:

facebook=facebook.reset\_index()

amazon=amazon.reset\_index()

google=google.reset\_index()

facebook.head()

In [14]:

facebook['month'] = pd.DatetimeIndex(facebook['Date']).month

google['month'] = pd.DatetimeIndex(google['Date']).month

amazon['month'] = pd.DatetimeIndex(amazon['Date']).month

**Amazon Monthly Closing**

In [15]:

amazon\_monthly\_closing=amazon.groupby("month").sum()['Close'].reset\_index()

plt.title("Amazon Monthly Closing")

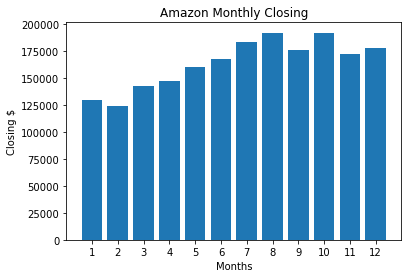
plt.bar(amazon\_monthly\_closing['month'],amazon\_monthly\_closing['Close'])

plt.xticks(amazon\_monthly\_closing['month'])

plt.xlabel("Months")

plt.ylabel("Closing $")

plt.show()



**Facebook Monthly Closing**

In [16]:

facebook\_monthly\_closing=facebook.groupby("month").sum()['Close'].reset\_index()

plt.title("Facebook Monthly Closing")

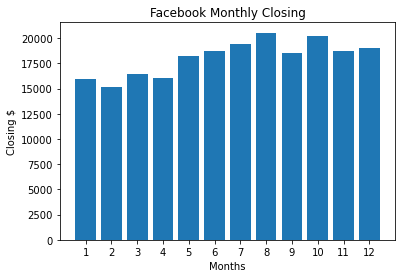
plt.bar(facebook\_monthly\_closing['month'],facebook\_monthly\_closing['Close'])

plt.xticks(facebook\_monthly\_closing['month'])

plt.xlabel("Months")

plt.ylabel("Closing $")

plt.show()



**Google Monthly Closing**

In [17]:

google\_monthly\_closing=google.groupby("month").sum()['Close'].reset\_index()

plt.title("Google Monthly Closing")

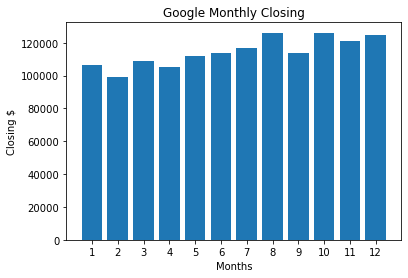
plt.bar(google\_monthly\_closing['month'],google\_monthly\_closing['Close'])

plt.xticks(google\_monthly\_closing['month'])

plt.xlabel("Months")

plt.ylabel("Closing $")

plt.show()



## Google Open and Closing Shares

In [18]:

google.set\_index("Date")

plt.plot(google['Date'],google['Close'],label ="Closing")

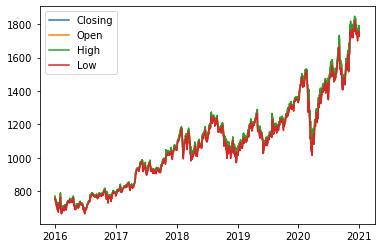
plt.plot(google['Date'],google['Open'],label ="Open")

plt.plot(google['Date'],google['High'],label ="High")

plt.plot(google['Date'],google['Low'],label ="Low")

plt.legend()

plt.show()



In [19]:

google['Open'].iloc[1000:1400].plot

Out[19]:

<pandas.plotting.\_core.PlotAccessor object at 0x00000174CEBDC850>

In [20]:

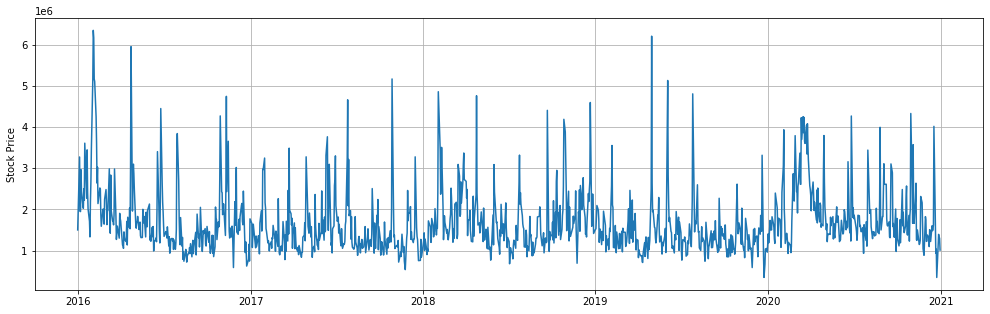
plt.figure(figsize=(17,5))

plt.plot(google['Date'],google['Volume'])

plt.ylabel("Stock Price")

plt.grid()

plt.show()



### Facebook Opening and Closing Shares

In [21]:

plt.plot(facebook['Date'],facebook['Close'],label ="Closing")

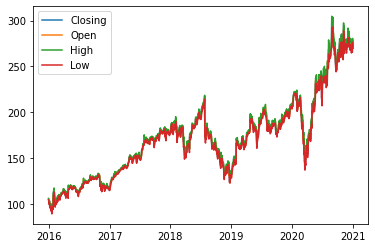
plt.plot(facebook['Date'],facebook['Open'],label ="Open")

plt.plot(facebook['Date'],facebook['High'],label ="High")

plt.plot(facebook['Date'],facebook['Low'],label ="Low")

plt.legend()

plt.show()



In [22]:

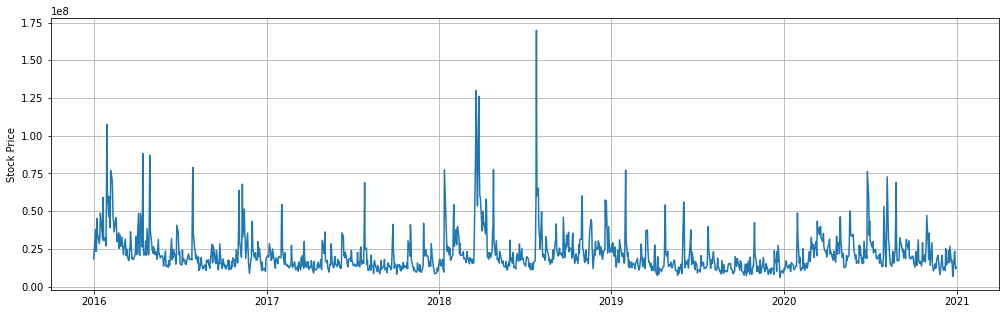
plt.figure(figsize=(17,5))

plt.plot(facebook['Date'],facebook['Volume'])

plt.ylabel("Stock Price")

plt.grid()

plt.show()



### Amazon Opening and Closing Shares

In [23]:

plt.plot(amazon['Date'],amazon['Close'],label ="Closing")

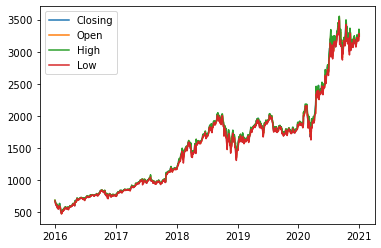
plt.plot(amazon['Date'],amazon['Open'],label ="Open")

plt.plot(amazon['Date'],amazon['High'],label ="High")

plt.plot(amazon['Date'],amazon['Low'],label ="Low")

plt.legend()

plt.show()



In [24]:

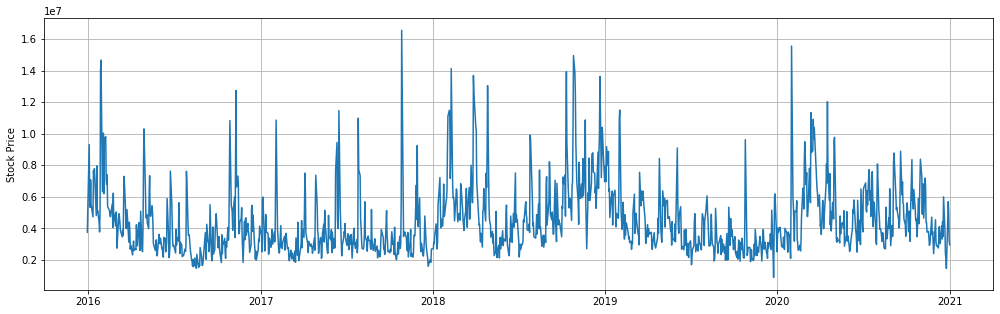
plt.figure(figsize=(17,5))

plt.plot(amazon['Date'],amazon['Volume'])

plt.ylabel("Stock Price")

plt.grid()

plt.show()



In [25]:

google['Total Traded']=google['Open']\*google['Volume']

facebook['Total Traded']=facebook['Open']\*facebook['Volume']

amazon['Total Traded']=amazon['Open']\*amazon['Volume']

In [26]:

google.set\_index("Date",inplace=**True**)

facebook.set\_index("Date",inplace=**True**)

amazon.set\_index("Date",inplace=**True**)

## Total Traded

In [27]:

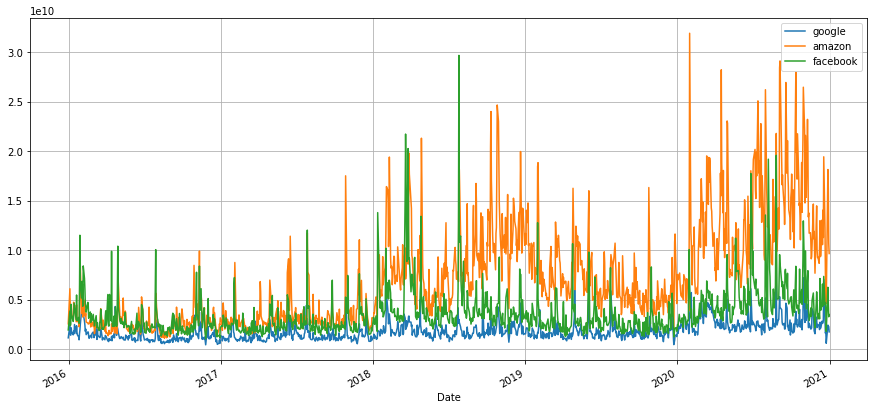
google['Total Traded'].plot(label='google',figsize=(15,7))

amazon['Total Traded'].plot(label='amazon',figsize=(15,7))

facebook['Total Traded'].plot(label='facebook',figsize=(15,7))

plt.legend()

plt.grid()



In [28]:

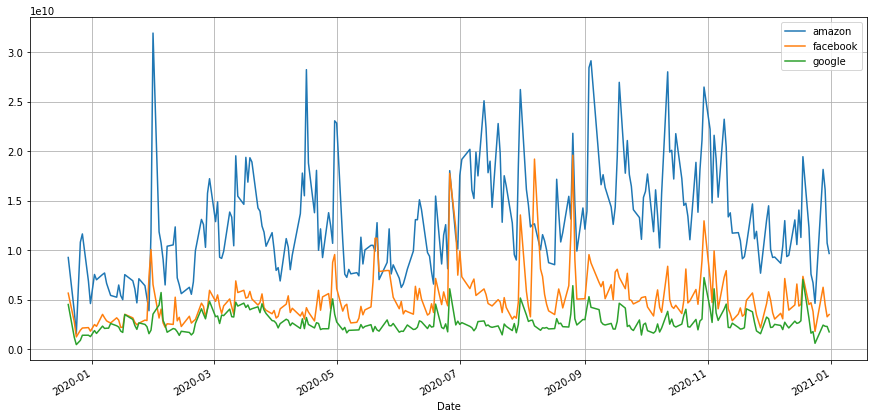
amazon['Total Traded'].iloc[1000:1300].plot(label="amazon",figsize=(15,7))

facebook['Total Traded'].iloc[1000:1300].plot(label="facebook")

google['Total Traded'].iloc[1000:1300].plot(label="google")

plt.legend()

plt.grid()



Amazon is the most traded share during 2020, because in during lockdown most of people start their own buisness using Amazon FBA and Amazon PL

**Now Check The Relation by using correlation and scatter matrix**

In [29]:

open\_share=pd.concat([google['Open'],amazon['Open'],facebook['Open']],axis=1)

open\_share.columns=['Google Open', 'Amazon Open','Facebook Open']

In [30]:

scatter\_matrix(open\_share,figsize=(8,8),hist\_kwds={'bins':50})

Out[30]:

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CEF446A0>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CEF72130>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CEF9B580>],

[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CEFC6A00>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CEFF3E80>,

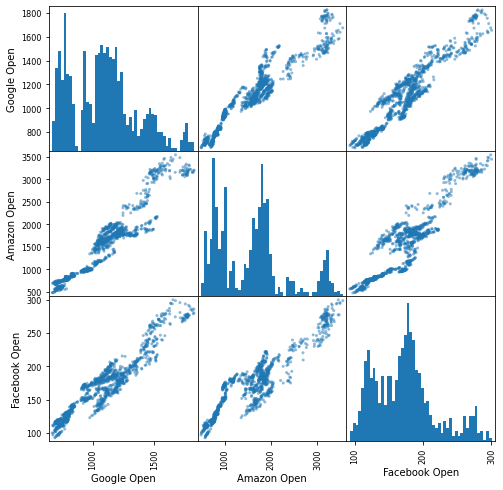
<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF2AC280>],

[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF2AC370>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF2D8820>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF3310D0>]],

dtype=object)



As you see, facebook and google show a possitive good relation between each other

## Correlation of Close stocks

In [31]:

close\_share=pd.concat([google['Close'],amazon['Close'],facebook['Close']],axis=1)

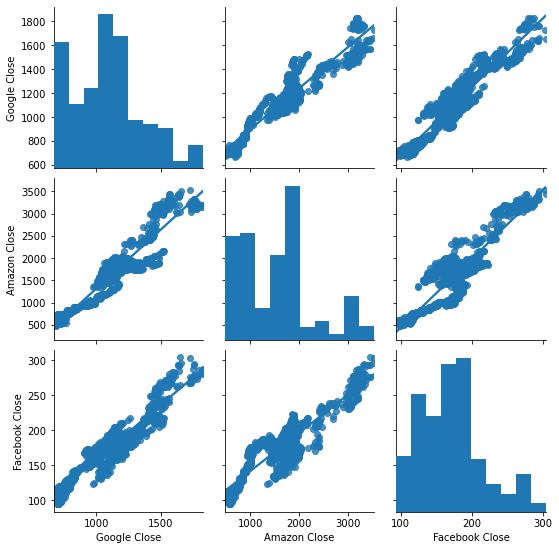
close\_share.columns=['Google Close', 'Amazon Close','Facebook Close']

In [32]:

sns.pairplot(close\_share, kind='reg')

Out[32]:

<seaborn.axisgrid.PairGrid at 0x174cfa20700>



## Daily Percentage Change

In [33]:

*## Or check the volatility of any stock*

facebook['return']=(facebook['Close']/facebook['Close'].shift(1))-1

amazon['return']=(amazon['Close']/amazon['Close'].shift(1))-1

google['return']=(google['Close']/google['Close'].shift(1))-1

In [34]:

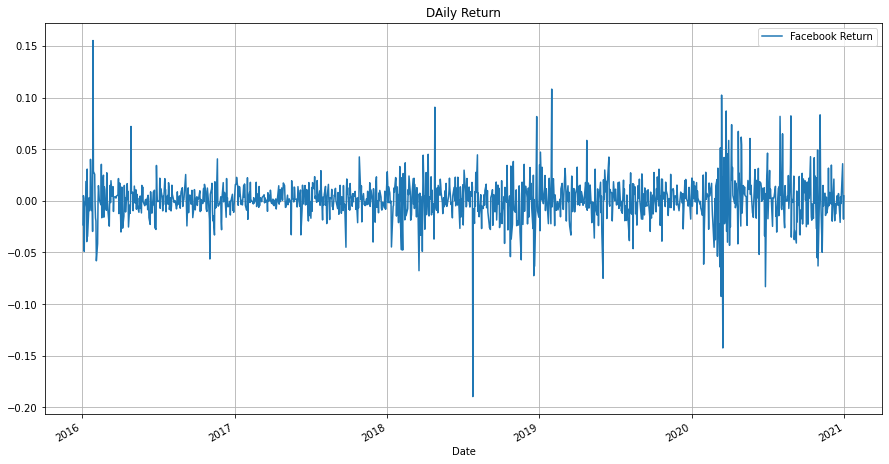
plt.title("DAily Return")

facebook['return'].plot(label='Facebook Return',figsize=(15,8))

plt.legend()

plt.grid()

plt.show()



In [35]:

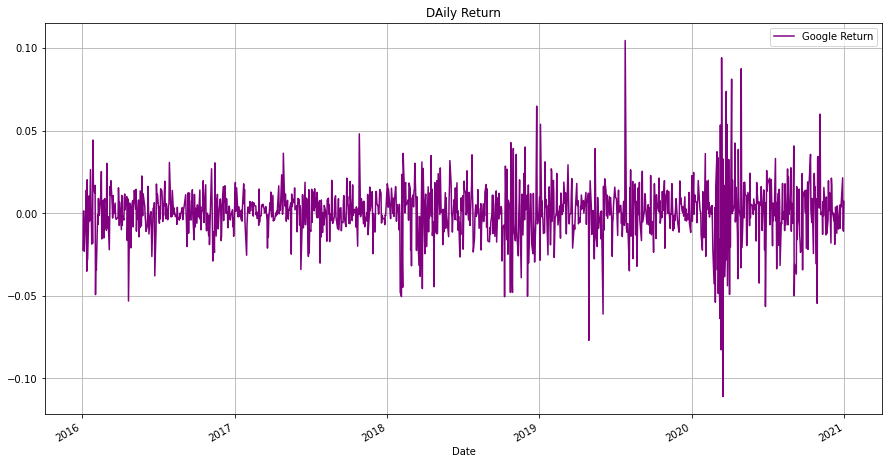
plt.title("DAily Return")

google['return'].plot(label='Google Return',figsize=(15,8),color='purple')

plt.legend()

plt.grid()

plt.show()



In [36]:

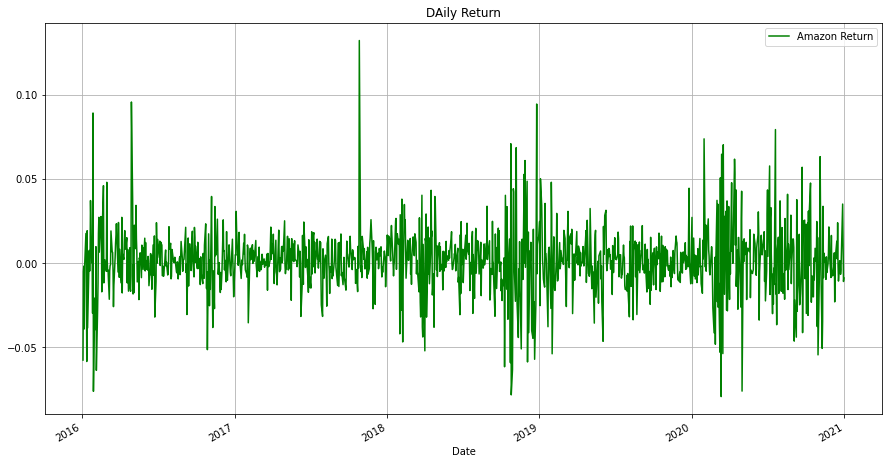
plt.title("DAily Return")

amazon['return'].plot(label='Amazon Return',figsize=(15,8),color="green")

plt.legend()

plt.grid()

plt.show()



In [37]:

facebook['return'].hist(bins=100, label="facebook",alpha=0.5,figsize=(15,8))

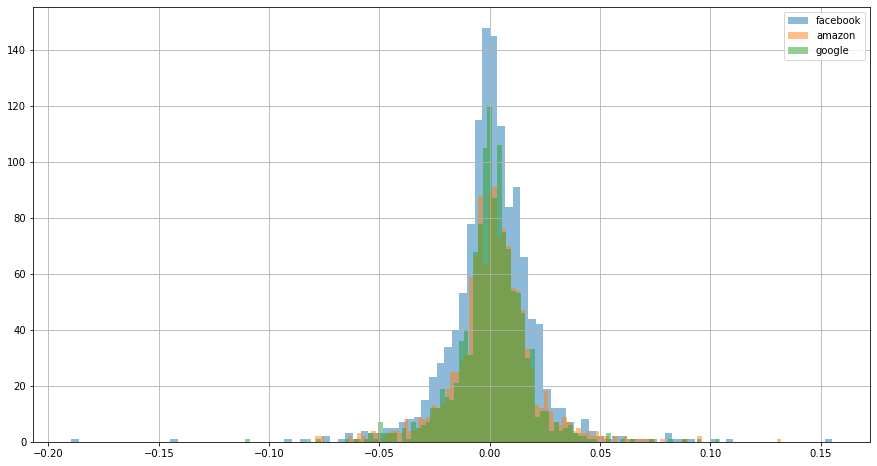
amazon['return'].hist(bins=100, label="amazon",alpha=0.5)

google['return'].hist(bins=100, label="google",alpha=0.5)

plt.legend()

Out[37]:

<matplotlib.legend.Legend at 0x174d0b47490>



In [38]:

*## TO normalize the data you should use KDE = kernal distribution estimation*

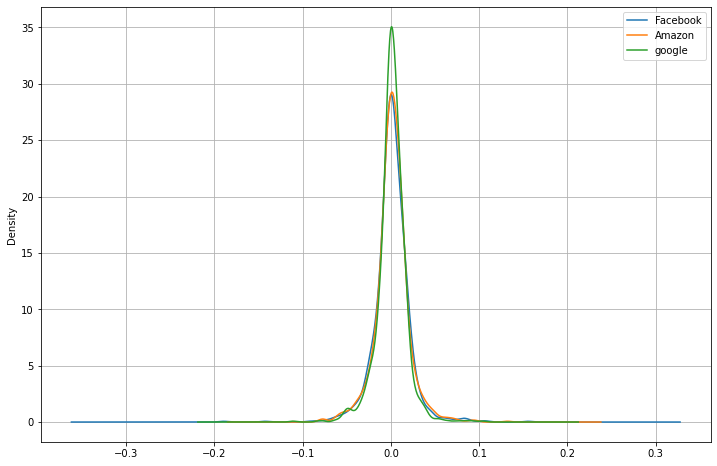
facebook['return'].plot(kind='kde',label='Facebook',figsize=(12,8))

amazon['return'].plot(kind='kde',label='Amazon',figsize=(12,8))

google['return'].plot(kind='kde',label='google',figsize=(12,8))

plt.legend()

plt.grid()



Dont be wonder about that spikes of facebook is high in histogram. and here it is down. It shows density and KDE is normalized. So the area of all three stocks are constant

In [39]:

box=pd.concat([facebook['return'],google['return'],amazon['return']],axis=1)

box.columns=['Facebook Return',"Google Return", "Amazon Return"]

box.head()

In [40]:

scatter\_matrix(box,figsize=(8,8),hist\_kwds={'bins':50})

Out[40]:

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF26DD30>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF626550>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF654940>],

[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF680D90>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF6BC220>,

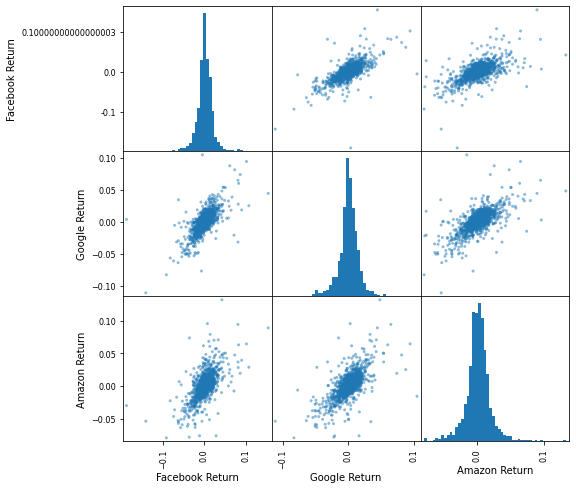
<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF7165E0>],

[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF7166D0>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174CF742B80>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000174D10C83D0>]],

dtype=object)



## Cumulative Return

we use cumulative return for capture the long term investment. if the cumulative return is greater than 1 you gain profit otherwise you lose it.

In [41]:

facebook['Cumulative Return']=(1+facebook['return']).cumprod()

google['Cumulative Return']=(1+google['return']).cumprod()

amazon['Cumulative Return']=(1+amazon['return']).cumprod()

In [42]:

plt.title("Cumulative Return Vs Time")

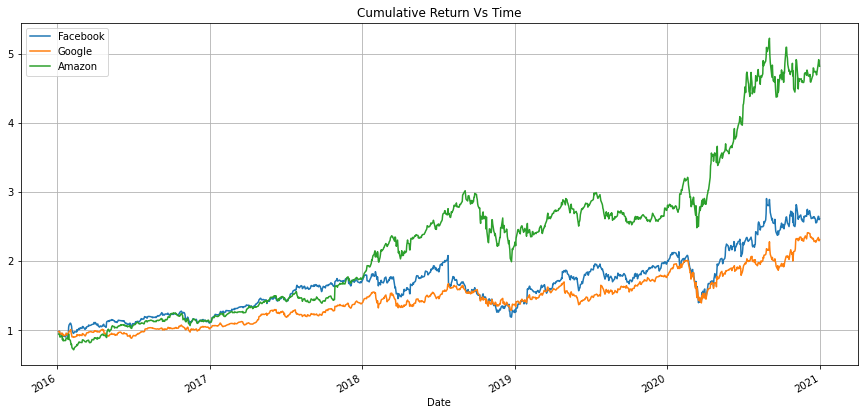
facebook['Cumulative Return'].plot(label="Facebook",figsize=(15,7))

google['Cumulative Return'].plot(label="Google")

amazon['Cumulative Return'].plot(label="Amazon")

plt.legend()

plt.grid()



## Linear Regression

In [47]:

**from** **plotly** **import** \_\_version\_\_

**from** **plotly.offline** **import** download\_plotlyjs, init\_notebook\_mode, plot, iplot

print(\_\_version\_\_)

4.14.1

In [48]:

*#libraries for ploting Graph*

**import** **chart\_studio.plotly** **as** **py**

**import** **plotly.graph\_objs** **as** **go**

**from** **plotly.offline** **import** plot

**import** **numpy** **as** **np**

*#for offline plotting*

init\_notebook\_mode(connected=**True**)

In [49]:

*#Resettin g all the dataset index from date to numeric form*

amazon=amazon.reset\_index()

google=google.reset\_index()

facebook=facebook.reset\_index()

*#Setting the layout of all the company for ploting*

amazon\_layout=go.Layout(

title='Stock Price of Amazon',

xaxis=dict(

title='Date',

titlefont=dict(

family='Courier New, monospace',

size=18,

color='#7f7f7f'

)

),

yaxis=dict(

title='Price',

titlefont=dict(

family='Courier New, monospace',

size=18,

color='#7f7f7f'

)

)

)

google\_layout=go.Layout(

title='Stock Price of Google',

xaxis=dict(

title='Date',

titlefont=dict(

family='Courier New, monospace',

size=18,

color='#7f7f7f'

)

),

yaxis=dict(

title='Price',

titlefont=dict(

family='Courier New, monospace',

size=18,

color='#7f7f7f'

)

)

)

facebook\_layout=go.Layout(

title='Stock Price of Facebook',

xaxis=dict(

title='Date',

titlefont=dict(

family='Courier New, monospace',

size=18,

color='#7f7f7f'

)

),

yaxis=dict(

title='Price',

titlefont=dict(

family='Courier New, monospace',

size=18,

color='#7f7f7f'

)

)

)

In [50]:

*#ploting amazon data using the layout*

amazon\_data=[{'x':amazon['Date'],'y':amazon['Close']}]

amazon\_plot=go.Figure(data=amazon\_data,layout=amazon\_layout)

iplot(amazon\_plot)

In [51]:

*#ploting google data using the layout*

google\_data=[{'x':google['Date'],'y':google['Close']}]

google\_plot=go.Figure(data=google\_data,layout=google\_layout)

iplot(google\_plot)

In [52]:

*#ploting facebook data using the layout*

facebook\_data=[{'x':facebook['Date'],'y':facebook['Close']}]

facebook\_plot=go.Figure(data=facebook\_data,layout=facebook\_layout)

iplot(facebook\_plot)

In [53]:

*#Building the regression model*

**from** **sklearn.model\_selection** **import** train\_test\_split

*#for prepocessing*

**from** **sklearn.preprocessing** **import** MinMaxScaler

**from** **sklearn.preprocessing** **import** StandardScaler

*#For model evaluation*

**from** **sklearn.metrics** **import** mean\_squared\_error **as** mse

**from** **sklearn.metrics** **import** r2\_score

In [54]:

*#Splitting the dataset into train and test sets for linear regression*

*#for amazon*

amazon\_X =np.array(amazon.index).reshape(-1,1)

amazon\_Y =amazon['Close']

*#for training we are taking 80%(0.8) of the whole dataset and assigning a random state of 101*

amazon\_X\_train,amazon\_X\_test,amazon\_Y\_train,amazon\_Y\_test=train\_test\_split(amazon\_X,amazon\_Y,test\_size=0.8,random\_state=101)

In [55]:

*#Splitting the dataset into train and test sets for linear regression*

*#for google*

google\_X =np.array(google.index).reshape(-1,1)

google\_Y =google['Close']

*#for training we are taking 80%(0.8) of the whole dataset and assigning a random state of 101*

google\_X\_train,google\_X\_test,google\_Y\_train,google\_Y\_test=train\_test\_split(google\_X,google\_Y,test\_size=0.8,random\_state=101)

In [56]:

*#Splitting the dataset into train and test sets for linear regression*

*#for facebook*

facebook\_X =np.array(facebook.index).reshape(-1,1)

facebook\_Y =facebook['Close']

*#for training we are taking 80%(0.8) of the whole dataset and assigning a random state of 101*

facebook\_X\_train,facebook\_X\_test,facebook\_Y\_train,facebook\_Y\_test=train\_test\_split(facebook\_X,facebook\_Y,test\_size=0.8,random\_state=101)

In [57]:

*#feature scaling*

amazon\_scaler=StandardScaler().fit(amazon\_X\_train)

google\_scaler=StandardScaler().fit(google\_X\_train)

facebook\_scaler=StandardScaler().fit(facebook\_X\_train)

*#standardize scaler is mandatory for any machine learning*

*#they tend to behave badly if the dataset are not distributed properly*

In [58]:

*#importing linear Regression library to perform modeling*

**from** **sklearn.linear\_model** **import** LinearRegression

In [59]:

*#create a linear model*

*#provide x and y train value for the linear regression model*

amazon\_lm = LinearRegression()

amazon\_lm.fit(amazon\_X\_train,amazon\_Y\_train)

google\_lm = LinearRegression()

google\_lm.fit(google\_X\_train,google\_Y\_train)

facebook\_lm = LinearRegression()

facebook\_lm.fit(facebook\_X\_train,facebook\_Y\_train)

Out[59]:

LinearRegression()

In [60]:

*#plot actual and predicted values for train dataset using scatter plot*

*#actual values*

amazon\_trace0 = go.Scatter(

x=amazon\_X\_train.T[0],

y=amazon\_Y\_train,

mode='markers',

name='Actual'

)

google\_trace0 = go.Scatter(

x=google\_X\_train.T[0],

y=google\_Y\_train,

mode='markers',

name='Actual'

)

facebook\_trace0 = go.Scatter(

x=facebook\_X\_train.T[0],

y=facebook\_Y\_train,

mode='markers',

name='Actual'

)

*#predicted Values*

amazon\_trace1 = go.Scatter(

x=amazon\_X\_train.T[0],

y=amazon\_lm.predict(amazon\_X\_train).T,

mode='lines',

name='Predicted'

)

google\_trace1 = go.Scatter(

x=google\_X\_train.T[0],

y=google\_lm.predict(google\_X\_train).T,

mode='lines',

name='Predicted'

)

facebook\_trace1 = go.Scatter(

x=facebook\_X\_train.T[0],

y=facebook\_lm.predict(facebook\_X\_train).T,

mode='lines',

name='Predicted'

)

*#combining the actual and predicted value in a variable*

*#for amazon*

amazon\_data=[amazon\_trace0,amazon\_trace1]

amazon\_layout.xaxis.title.text='Day'

amazon\_plot2=go.Figure(data=amazon\_data,layout=amazon\_layout)

*#for google*

google\_data=[google\_trace0,google\_trace1]

google\_layout.xaxis.title.text='Day'

google\_plot2=go.Figure(data=google\_data,layout=google\_layout)

*#for facebook*

facebook\_data=[facebook\_trace0,facebook\_trace1]

facebook\_layout.xaxis.title.text='Day'

facebook\_plot2=go.Figure(data=facebook\_data,layout=facebook\_layout)

In [61]:

iplot(amazon\_plot2)

iplot(google\_plot2)

iplot(facebook\_plot2)

020040060080010001200500100015002000250030003500

ActualPredictedStock Price of AmazonDayPrice

02004006008001000120060080010001200140016001800

ActualPredictedStock Price of GoogleDayPrice

020040060080010001200100150200250300

ActualPredictedStock Price of FacebookDayPrice

In [62]:

*#Calculate scores for the model evaluation*

*# we will find the error using r2 error and the mean squar error on our trained model*

print(' Amazon Predicted scores'.center(50))

amazon\_scores=f'''

**{**'Metric'.ljust(10)**}** **{**'Train'.center(20)**}** **{**'Test'.center(20)**}**

**{**'r2\_score'.ljust(10)**}** **{**r2\_score(amazon\_Y\_train,amazon\_lm.predict(amazon\_X\_train))**}** **\t** **{**r2\_score(amazon\_Y\_test,amazon\_lm.predict(amazon\_X\_test))**}**

**{**'MSE'.ljust(10)**}** **{**mse(amazon\_Y\_train,amazon\_lm.predict(amazon\_X\_train))**}** **\t** **{**mse(amazon\_Y\_test,amazon\_lm.predict(amazon\_X\_test))**}**

'''

print(amazon\_scores)

print(' Google Predicted scores'.center(50))

google\_scores=f'''

**{**'Metric'.ljust(10)**}** **{**'Train'.center(20)**}** **{**'Test'.center(20)**}**

**{**'r2\_score'.ljust(10)**}** **{**r2\_score(google\_Y\_train,google\_lm.predict(google\_X\_train))**}** **\t** **{**r2\_score(google\_Y\_test,google\_lm.predict(google\_X\_test))**}**

**{**'MSE'.ljust(10)**}** **{**mse(google\_Y\_train,google\_lm.predict(google\_X\_train))**}** **\t** **{**mse(google\_Y\_test,google\_lm.predict(google\_X\_test))**}**

'''

print(google\_scores)

print(' Facebook Predicted scores'.center(50))

facebook\_scores=f'''

**{**'Metric'.ljust(10)**}** **{**'Train'.center(20)**}** **{**'Test'.center(20)**}**

**{**'r2\_score'.ljust(10)**}** **{**r2\_score(facebook\_Y\_train,facebook\_lm.predict(facebook\_X\_train))**}** **\t** **{**r2\_score(facebook\_Y\_test,facebook\_lm.predict(facebook\_X\_test))**}**

**{**'MSE'.ljust(10)**}** **{**mse(facebook\_Y\_train,facebook\_lm.predict(facebook\_X\_train))**}** **\t** **{**mse(facebook\_Y\_test,facebook\_lm.predict(facebook\_X\_test))**}**

'''

print(facebook\_scores)

Amazon Predicted scores

Metric Train Test

r2\_score 0.8739057706864323 0.8735813556226095

MSE 69838.22580426346 70318.32364804555

Google Predicted scores

Metric Train Test

r2\_score 0.8992669114349598 0.8841378650102822

MSE 7289.271495794161 8387.464160938096

Facebook Predicted scores

Metric Train Test

r2\_score 0.7896974591695884 0.7386964207704373

MSE 412.1924069316361 513.1511894083718

## Using LSTM For prediction

In [63]:

gogle\_close=google.reset\_index()['Close']

amzn\_close=amazon.reset\_index()['Close']

fb\_close=facebook.reset\_index()['Close']

**Google Closing Stocks**

In [64]:

gogle\_close.isna().sum()

Out[64]:

0

In [65]:

gogle\_closing=MinMaxScaler(feature\_range=(0,1))

gogle\_close=gogle\_closing.fit\_transform(np.array(gogle\_close).reshape(-1,1))

In [66]:

gogle\_close *# you see the differnce before and after transformation. You can see that the values are now*

*# in between 0 and 1. It is very necessary step to normalize your data to applying any model.*

Out[66]:

array([[0.07813887],

[0.06344582],

[0.06408389],

...,

[0.94027056],

[0.92371503],

[0.93437267]])

In [67]:

gogle\_training\_size=int(len(gogle\_close)\*0.65) *# if you take 65% of data into training and remaining 35% into testing*

gogle\_test\_size=len(gogle\_close)-gogle\_training\_size

gogle\_train\_data,gogle\_test\_data=gogle\_close[0:gogle\_training\_size,:],gogle\_close[gogle\_training\_size:len(gogle\_close),:1]

In [68]:

**def** create\_dataset(dataset,time\_step=1):

X\_data\_gogle,Y\_data\_gogle=[],[]

**for** i **in** range(len(dataset)-time\_step-1):

a= dataset[i:(i+time\_step),0]

X\_data\_gogle.append(a)

Y\_data\_gogle.append(dataset[i+time\_step,0])

**return** np.array(X\_data\_gogle),np.array(Y\_data\_gogle)

In [69]:

time\_step=100

gogle\_X\_train,gogle\_Y\_train=create\_dataset(gogle\_train\_data,time\_step)

gogle\_X\_test,gogle\_Y\_test=create\_dataset(gogle\_test\_data,time\_step)

In [70]:

print(gogle\_X\_test.shape)

print(gogle\_X\_train.shape)

(340, 100)

(718, 100)

In [71]:

*# Now reshape your data into 3 dimensional because it is neccessary to make your data 3 dimensional*

gogle\_X\_train=gogle\_X\_train.reshape(gogle\_X\_train.shape[0],gogle\_X\_train.shape[1],1)

gogle\_X\_test=gogle\_X\_test.reshape(gogle\_X\_test.shape[0],gogle\_X\_test.shape[1],1)

In [72]:

**from** **tensorflow.keras.models** **import** Sequential

**from** **tensorflow.keras.layers** **import** Dense

**from** **tensorflow.keras.layers** **import** LSTM

In [73]:

model=Sequential()

model.add(LSTM(50,return\_sequences=**True**,input\_shape=(100,1)))

model.add(LSTM(50,return\_sequences=**True**))

model.add(LSTM(50))

model.add(Dense(1))

model.compile(loss="mean\_squared\_error",optimizer='adam')

In [74]:

model.fit(gogle\_X\_train,gogle\_Y\_train,validation\_data=(gogle\_X\_test,gogle\_Y\_test),epochs=100,batch\_size=64,verbose=1)

In [75]:

**import** **tensorflow** **as** **tf**

In [76]:

gogle\_train\_predict=model.predict(gogle\_X\_train)

gogle\_test\_predict=model.predict(gogle\_X\_test)

In [77]:

gogle\_train\_predict=gogle\_closing.inverse\_transform(gogle\_train\_predict)

gogle\_test\_predict=gogle\_closing.inverse\_transform(gogle\_test\_predict)

## Predict

In [78]:

**import** **math**

math.sqrt(mse(gogle\_Y\_train,gogle\_train\_predict))

Out[78]:

980.699826960617

In [79]:

math.sqrt(mse(gogle\_Y\_test,gogle\_test\_predict))

Out[79]:

1409.3942483374544

In [80]:

look\_back=100

trainPredictPlot=np.empty\_like(gogle\_close)

trainPredictPlot[:, :]=np.nan

trainPredictPlot[look\_back:len(gogle\_train\_predict)+look\_back,:]=gogle\_train\_predict

testPredictPlot=np.empty\_like(gogle\_close)

testPredictPlot[:, :]=np.nan

testPredictPlot[len(gogle\_train\_predict)+(look\_back\*2)+1:len(gogle\_close)-1,:]=gogle\_test\_predict

plt.figure(figsize=(15,8))

plt.title("Google Stocks Prediction")

plt.plot(gogle\_closing.inverse\_transform(gogle\_close))

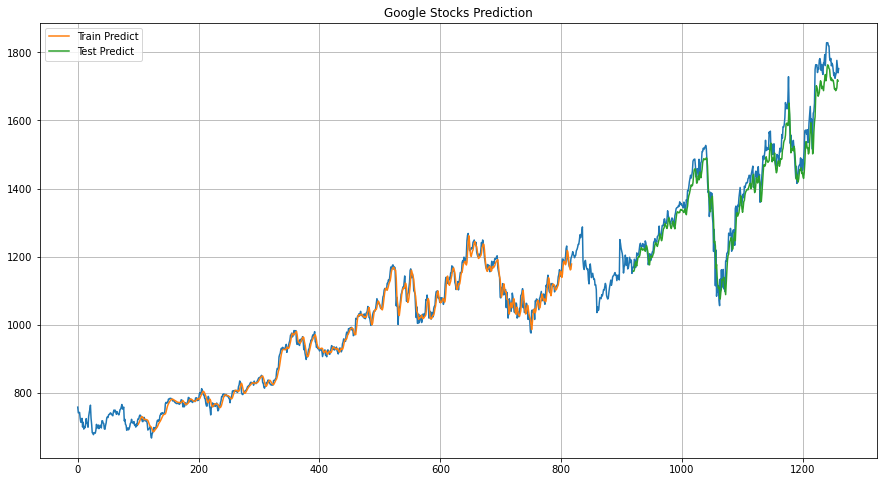
plt.plot(trainPredictPlot,label="Train Predict", )

plt.plot(testPredictPlot,label="Test Predict")

plt.legend()

plt.grid()

plt.show()



**Amazon Stocks Prediction**

In [81]:

amzn\_closing=MinMaxScaler(feature\_range=(0,1))

amzn\_close=amzn\_closing.fit\_transform(np.array(amzn\_close).reshape(-1,1))

In [82]:

amzn\_training\_size=int(len(amzn\_close)\*0.65) *# if you take 65% of data into training and remaining 35% into testing*

amzn\_test\_size=len(amzn\_close)-amzn\_training\_size

amzn\_train\_data,amzn\_test\_data=amzn\_close[0:amzn\_training\_size,:],amzn\_close[amzn\_training\_size:len(amzn\_close),:1]

In [83]:

**def** amzn\_create\_dataset(dataset,time\_step=1):

X\_data\_amzn,Y\_data\_amzn=[],[]

**for** i **in** range(len(dataset)-time\_step-1):

a= dataset[i:(i+time\_step),0]

X\_data\_amzn.append(a)

Y\_data\_amzn.append(dataset[i+time\_step,0])

**return** np.array(X\_data\_amzn),np.array(Y\_data\_amzn)

In [84]:

time\_step=100

amzn\_X\_train,amzn\_Y\_train=amzn\_create\_dataset(amzn\_train\_data,time\_step)

amzn\_X\_test,amzn\_Y\_test=amzn\_create\_dataset(amzn\_test\_data,time\_step)

In [85]:

amzn\_X\_train=amzn\_X\_train.reshape(amzn\_X\_train.shape[0],amzn\_X\_train.shape[1],1)

amzn\_X\_test=amzn\_X\_test.reshape(amzn\_X\_test.shape[0],amzn\_X\_test.shape[1],1)

In [86]:

model.fit(amzn\_X\_train,amzn\_Y\_train,validation\_data=(amzn\_X\_test,amzn\_Y\_test),epochs=100,batch\_size=64,verbose=1)

In [87]:

amzn\_train\_predict=model.predict(amzn\_X\_train)

amzn\_test\_predict=model.predict(amzn\_X\_test)

In [88]:

amzn\_train\_predict=amzn\_closing.inverse\_transform(amzn\_train\_predict)

amzn\_test\_predict=amzn\_closing.inverse\_transform(amzn\_test\_predict)

In [89]:

**import** **math**

math.sqrt(mse(amzn\_Y\_train,amzn\_train\_predict))

Out[89]:

1294.8729367632075

In [90]:

math.sqrt(mse(amzn\_Y\_test,amzn\_test\_predict))

Out[90]:

2458.8373717143113

In [91]:

look\_back=100

trainPredictPlot=np.empty\_like(amzn\_close)

trainPredictPlot[:, :]=np.nan

trainPredictPlot[look\_back:len(amzn\_train\_predict)+look\_back,:]=amzn\_train\_predict

testPredictPlot=np.empty\_like(amzn\_close)

testPredictPlot[:, :]=np.nan

testPredictPlot[len(amzn\_train\_predict)+(look\_back\*2)+1:len(amzn\_close)-1,:]=amzn\_test\_predict

plt.figure(figsize=(15,8))

plt.title("Amazon Stocks Prediction")

plt.plot(amzn\_closing.inverse\_transform(amzn\_close))

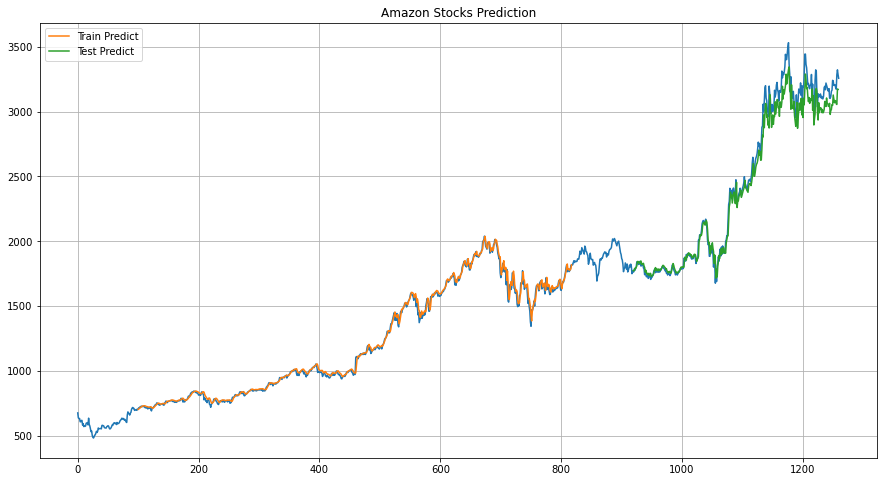
plt.plot(trainPredictPlot,label="Train Predict", )

plt.plot(testPredictPlot,label="Test Predict")

plt.legend()

plt.grid()

plt.show()



Facebook Stocks Prediction

In [92]:

fb\_closing=MinMaxScaler(feature\_range=(0,1))

fb\_close=fb\_closing.fit\_transform(np.array(fb\_close).reshape(-1,1))

In [93]:

fb\_training\_size=int(len(fb\_close)\*0.65) *# if you take 65% of data into training and remaining 35% into testing*

fb\_test\_size=len(fb\_close)-fb\_training\_size

fb\_train\_data,fb\_test\_data=fb\_close[0:fb\_training\_size,:],fb\_close[fb\_training\_size:len(fb\_close),:1]

In [94]:

**def** fb\_create\_dataset(dataset,time\_step=1):

X\_data\_fb,Y\_data\_fb=[],[]

**for** i **in** range(len(dataset)-time\_step-1):

a= dataset[i:(i+time\_step),0]

X\_data\_fb.append(a)

Y\_data\_fb.append(dataset[i+time\_step,0])

**return** np.array(X\_data\_fb),np.array(Y\_data\_fb)

In [95]:

time\_step=100

fb\_X\_train,fb\_Y\_train=fb\_create\_dataset(fb\_train\_data,time\_step)

fb\_X\_test,fb\_Y\_test=fb\_create\_dataset(fb\_test\_data,time\_step)

In [96]:

fb\_X\_train=fb\_X\_train.reshape(fb\_X\_train.shape[0],fb\_X\_train.shape[1],1)

fb\_X\_test=fb\_X\_test.reshape(fb\_X\_test.shape[0],fb\_X\_test.shape[1],1)

In [97]:

model.fit(fb\_X\_train,fb\_Y\_train,validation\_data=(fb\_X\_test,fb\_Y\_test),epochs=100,batch\_size=64,verbose=1)

In [98]:

fb\_train\_predict=model.predict(fb\_X\_train)

fb\_test\_predict=model.predict(fb\_X\_test)

In [99]:

fb\_train\_predict=fb\_closing.inverse\_transform(fb\_train\_predict)

fb\_test\_predict=fb\_closing.inverse\_transform(fb\_test\_predict)

In [100]:

math.sqrt(mse(fb\_Y\_train,fb\_train\_predict))

Out[100]:

155.44888603526775

In [101]:

math.sqrt(mse(fb\_Y\_test,fb\_test\_predict))

Out[101]:

217.56567833207023

In [102]:

look\_back=100

trainPredictPlot=np.empty\_like(fb\_close)

trainPredictPlot[:, :]=np.nan

trainPredictPlot[look\_back:len(fb\_train\_predict)+look\_back,:]=fb\_train\_predict

testPredictPlot=np.empty\_like(fb\_close)

testPredictPlot[:, :]=np.nan

testPredictPlot[len(fb\_train\_predict)+(look\_back\*2)+1:len(fb\_close)-1,:]=fb\_test\_predict

plt.figure(figsize=(15,8))

plt.title("FaceBook Stocks Prediction")

plt.plot(fb\_closing.inverse\_transform(fb\_close))

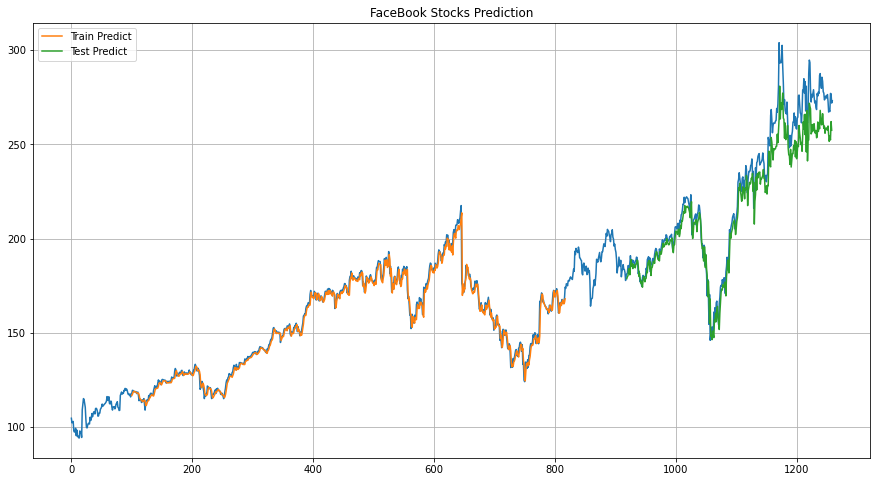
plt.plot(trainPredictPlot,label="Train Predict", )

plt.plot(testPredictPlot,label="Test Predict")

plt.legend()

plt.grid()

plt.show()



# Conclusion

Our project is based on artificial intelligence and programed on python programming language the goal of this project is to predict about ups and downs of stock market with respect to previous records it is working on multiple datasets. This program can be predicted false because it is not working on real time. It is working on a history-based records so it cannot be 100 percent accurate all the time sometimes shares prices may vary according to country’s economic conditions, environment of the market and other circumstances also influence on the stock market.