**Project 6**: Stock Price Prediction

**Project Title:** Stock Price Prediction

**Problem Statement:** To Build a predictive model to forecast stock prices based on historical market data, assisting investors in making well-informed decisions and optimizing their investment strategies.

**Problem Definition:** The problem is to build a predictive model that forecasts stock prices based on historical market data. The goal is to create a tool that assists investors in making well-informed decisions and optimizing their investment strategies. This project involves data collection, data preprocessing, feature engineering, model selection, training, and evaluation.

**Design Thinking:**

**Data Collection:** Collect historical stock market data, including features like date, open price, close price, volume, and other relevant indicators.

**Data Preprocessing:** Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.

**Feature Engineering:** Create additional features that could enhance the predictive power of the model, such as moving averages, technical indicators, and lagged variables.

**Model Selection:** Choose suitable algorithms for time series forecasting (e.g., ARIMA, LSTM) to predict stock prices.

**Model Training:** Train the selected model using the preprocessed data.

**Evaluation:** Evaluate the model's performance using appropriate time series forecasting metrics (e.g., Mean Absolute Error, Root Mean Squared Error).

### PHASE OF DEVELOPMENT:

***Given Dataset:***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Date** | **Open** | **High** | **Low** | **Close** | **Volume** |
| 1 | 01-03-2012 | 325.25 | 332.83 | 324.97 | 663.59 | 73,80,500 |
| 2 | 01-04-2012 | 331.27 | 333.87 | 329.08 | 666.45 | 57,49,400 |
| 3 | 01-05-2012 | 329.83 | 330.75 | 326.89 | 657.21 | 65,90,300 |
| 4 | 01-06-2012 | 328.34 | 328.77 | 323.68 | 648.24 | 54,05,900 |
| 5 | 01-09-2012 | 322.04 | 322.29 | 309.46 | 620.76 | 1,16,88,800 |
| … | … | … | … | … | … | … |
| 1254 | 12-23-2016 | 790.9 | 792.74 | 787.28 | 789.91 | 6,23,400 |
| 1255 | 12-27-2016 | 790.68 | 797.86 | 787.66 | 791.55 | 7,89,100 |
| 1256 | 12-28-2016 | 793.7 | 794.23 | 783.2 | 785.05 | 11,53,800 |
| 1257 | 12-29-2016 | 783.33 | 785.93 | 778.92 | 782.79 | 7,44,300 |
| 1258 | 12-30-2016 | 782.75 | 782.78 | 770.41 | 771.82 | 17,70,000 |

**1. Import Libraries**

***Program:***

#Import libraries

import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

**2. Importing the Training set**

***Program:***

dataset\_train = pd.read\_csv (“Google\_Stock\_Price\_Train.csv”)

training\_set = dataset\_train.ilot [:, 1:2].values

**3. Feature Scaling**

***Program:***

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxscaler (feature\_range = (0,1))

training\_set\_scaled = sc.fit\_transform (training\_set)

**4. Creating a data structure with 60 time steps and 1 output**

***Program:***

x\_train = []

y\_train = []

for i in range (60, 1258):

x\_train.append (training\_set\_scaled [i-60: i, 0])

y\_train.append (training\_set\_scaled [I, 0])

x\_train, y\_train = np.array (x\_train), np.array (y\_train)

**5. Reshaping**

***Program:***

x\_train = np.reshape (x\_train, (x\_train.shape [0],

x\_train.shape [1], 1))

**6.Constructing the RNN**

***Program:***

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers import Dropout

**7. Initialising the RNN**

***Program:***

regressor = Sequential()

**8.Adding the first LSTM layer and some Dropout regularisation**

***Program:***

regressor.add(LSTM(units = 50, return\_sequences = True, input\_shape = (X\_train.shape[1], 1)))

regressor.add(Dropout(0.2))

**9.Adding a second LSTM layer and some Dropout regularisation**

***Program:***

regressor.add(LSTM(units = 50, return\_sequences = True))

regressor.add(Dropout(0.2))

**10.Adding a third LSTM layer and some Dropout regularisation**

***Program:***

regressor.add(LSTM(units = 50, return\_sequences = True))

regressor.add(Dropout(0.2))

**11. Adding a fourth LSTM layer and some Dropout regularization**

***Program:***

regressor.add(LSTM(units = 50))

regressor.add(Dropout(0.2))

**12. Adding the output layer**

***Program:***

regressor.add(Dense(units = 1))

**13. Compiling the RNN**

***Program:***

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

**14.Fitting the RNN to the Training set**

regressor.fit(X\_train, y\_train, epochs = 100, batch\_size = 32)

**Making the predictions and visualising the results**

**15. Getting the real stock price of 2017**

***Program:***

dataset\_test = pd.read\_csv('Google\_Stock\_Price\_Test.csv')

real\_stock\_price = dataset\_test.iloc[:, 1:2].value

**16.Getting the predicted stock price of 2017**

***Program:***

dataset\_total = pd.concat((dataset\_train['Open'], dataset\_test['Open']), axis = 0)

inputs = dataset\_total[len(dataset\_total) - len(dataset\_test) - 60:].values

inputs = inputs.reshape(-1,1)

inputs = sc.transform(inputs)

X\_test = []

for i in range(60, 80):

X\_test.append(inputs[i-60:i, 0])

X\_test = np.array(X\_test)

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

predicted\_stock\_price = regressor.predict(X\_test)

predicted\_stock\_price = sc.inverse\_transform(predicted\_stock\_price)

plt.plot(real\_stock\_price, color = 'red', label = 'Real Google Stock Price')

plt.plot(predicted\_stock\_price, color = 'blue', label = 'Predicted Google Stock Price')

plt.title('Google Stock Price Prediction')

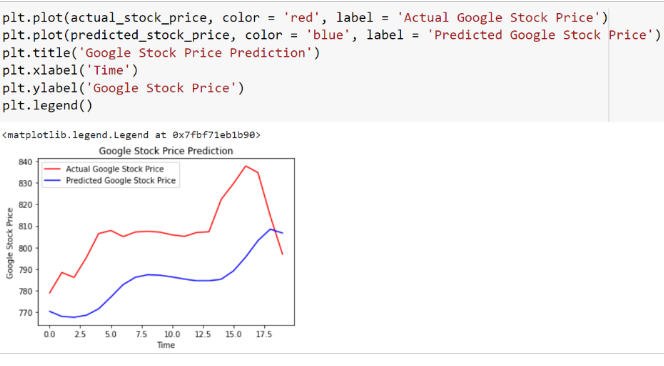
plt.xlabel('Time')

plt.ylabel('Google Stock Price')

plt.legend()

plt.show()

**17. Plotting the Actual and Predicted Prices for Google Stocks.**

****

**FLOWCHART:**

Data Collection

Data Preprocessing

Feature Selection/Engineering

Model Selection

### 

Model Training

Model Evaluation

Monitoring and Maintenance

Visualization

Testing

Hyperparameter Tuning

**Outcomes:**

Stock Price Prediction using machine learning helps to discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits.

**CHALLENGES:**

* **Data quality:**The accuracy and completeness of historical data are critical for making accurate predictions. Stock price data may contain errors, missing values, or outliers that can affect the model’s performance.
* **Non-linearity:** Stock prices are influenced by a complex set of factors, including economic indicators, market sentiment, news, and events. These factors are often non-linear and can interact in unexpected ways, making it difficult to model the relationships between them.
* **Data volume:** Financial data can be vast, and analyzing large datasets can be challenging. This is particularly true when dealing with high-frequency data, where the volume of data can quickly become overwhelming.
* **Overfitting:** Models can become too complex and overfit to the training data, making them less accurate when applied to new data. It’s essential to balance the model’s complexity with its ability to generalize to new data.
* **Dynamic nature of markets:**Financial markets are constantly changing, and models that work well in one market condition may not work in another. As a result, it’s important to continuously evaluate and update models to ensure they remain relevant.
* **Lack of transparency:**Some models, such as deep learning models, can be challenging to interpret. It’s important to ensure that models are transparent and can be easily understood by stakeholders.
* **Limited predictability:**Finally, it’s important to recognize that stock prices are inherently unpredictable, and even the best models can’t predict with 100% accuracy. It’s essential to communicate the limitations of the model and provide stakeholders with a range of possible outcomes.

**SOLUTIONS:**

* To address data quality issues, it’s essential to ensure that the data used for stock price prediction is accurate, complete, and up-to-date. This can be achieved by using data from reputable sources, performing data cleaning and validation, and implementing quality control measures to identify and correct errors.
* To address non-linear relationships between stock prices and other factors, advanced modeling techniques such as machine learning algorithms can be used. These techniques can identify complex patterns and relationships that may not be apparent using traditional statistical methods.
* Financial data can be vast, and analyzing large datasets can be challenging. This is particularly true when dealing with high-frequency data, where the volume of data can quickly become overwhelming.
* Models can become too complex and overfit to the training data, making them less accurate when applied to new data. It’s essential to balance the model’s complexity with its ability to generalize to new data.
* Financial markets are constantly changing, and models that work well in one market condition may not work in another. As a result, it’s important to continuously evaluate and update models to ensure they remain relevant.
* Some models, such as deep learning models, can be challenging to interpret. It’s important to ensure that models are transparent and can be easily understood by stakeholders.
* Finally, it’s important to recognize that stock prices are inherently unpredictable, and even the best models can’t predict with 100% accuracy. It’s essential to communicate the limitations of the model and provide stakeholders with a range of possible outcomes.

**CONCLUSION:**

The stock market plays a remarkable role in our daily lives. It is a significant factor in a country's GDP growth.