DOCUMENTATION

ABSTRACT

Stock markets help companies to raise capital.It helps generate personal wealth.Stock markets serve as an indicator of the state of the economy.It is a widely used source for people to invest money in companies with high growth potential. Traders in the stock market buy or sell shares on one or more of the stock exchanges that are part of the overall stock market.

Stock Price Prediction using machine learning helps you 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. Predicting how the stock market will perform is a hard task to do.

Stock market prediction and analysis are some of the most difficult jobs to complete. There are numerous causes for this, including market volatility and a variety of other dependent and independent variables that influence the value of a certain stock in the market. These variables make it extremely difficult for any stock market expert to anticipate the rise and fall of the market with great precision.

This project helps in analysis the trend of stock prices with minimum offset using machine learning.

TABLE OF CONTENTS

INTRODUCTION

Stock market can be something hard to guess, today we might see a price, check out the price of the same stock few minutes later, it will be a different story. We might often hear many investors saying that stock prices are so unpredictable and they have lost their money.
With help of right data and machine learning variations in stock prices can be predicted which can help a lot of investors by saving money and time and safeguard their investment.

Stock market prediction is a web application which is implemented by using machine learning to analyse stock prices .

The dataset which consists of details such as name of the stock , stock code ,opening price, closing price, high and low of price, volume of stock traded in a trading session over the past few years are considered.

Certain algorithms of ML are performed on dataset to convert the data into machine understandable language.
This project aims to predict stock trends to help the investors to decide whether to buy, sell or hold the stock by forecasting the price using previous data.

METHODOLOGY

Modules used in this process are

- Yfinance - library for obtaining data from yahoo finance API

  The Yahoo Finance API is a RESTful API that provides access to financial data. This data includes stock quotes, historical prices, and company information. The API is free to use and does not require an API key.

- Pandas-Open-source Python library designed to deal with data analysis and data manipulation.

  The main data structure in Pandas is a 2-dimensional table called **Data Frame**. To create a Data Frame, you can import data in several formats, such as *CSV*, *XLSX*, *JSON*, *SQL*. With some lines of code, you can add, delete, or edit data in your rows/columns, check your set's statistics, identify and handle missing entries, etc.

- Pandas_datareader - library for obtaining financial & economic data from online sources such as yahoo finance

  This module is specifically designed to interface with some of the world's most popular financial data APIs, and import their data into an easily digestible pandas DataFrame

- Numpy-used for working with arrays

  It also has functions for working in domain of linear algebra, fourier transform, and matrices. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

- Matplotlib - a comprehensive library for creating visualizations

  Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB.

- Sklearn - contains tools for ML, including regression and statistical modeling

  As a high-level library, it lets you define a predictive data model in just a few lines of code, and then use that model to fit your data. It's versatile and integrates well with other Python libraries, such as matplotlib for plotting**,** numpy for array vectorization**,** and pandas for dataframes**.**

- Streamlit-open source library for building and deploying ML and data science web applications

  Streamlit allows you to re-use any Python code you have already written. This can save considerable amounts of time compared to non-Python based tools where all code to create visualizations needs to be re-written.

- Keras library provides a collection of functions for creating and training machine learning models.

  keras.models is a module in keras library. We used ‘Sequential’ model which creates a simple linear stack of layers that define the model.’add’ method is used to add layers to the model.

IMPLEMENTATION

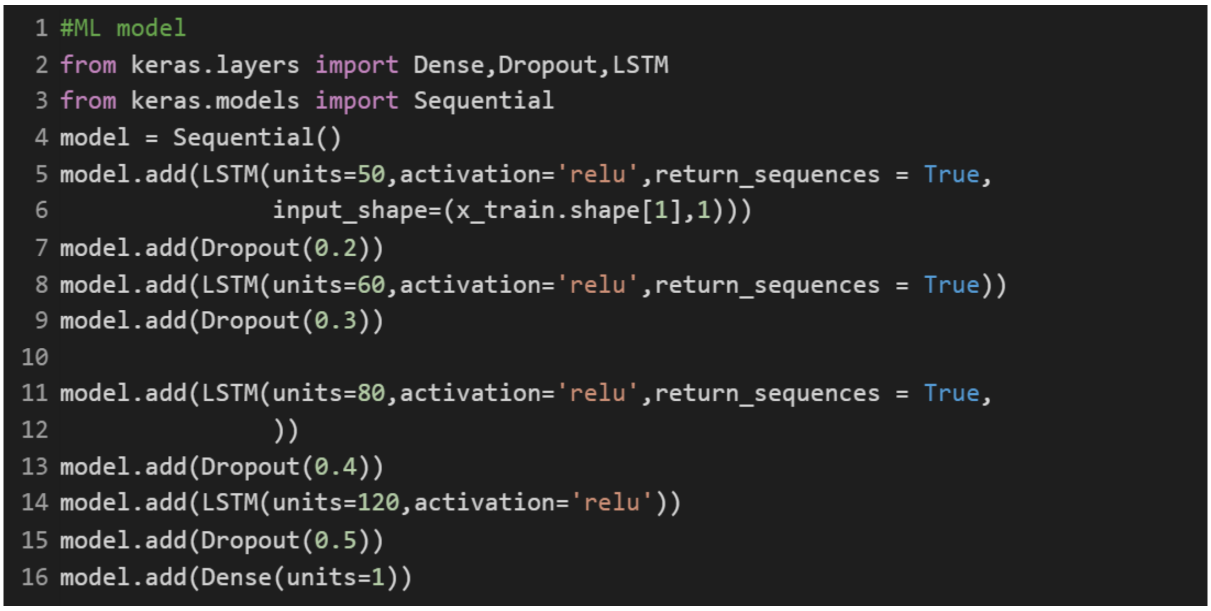
- In sequential model and the data flows from one layer to another layer in the given order until the data finally reaches the output layer.

- We have trained each layer of our model to extract features and pass them to next layers to extract complex features.

- LSTM layer stands for Long Short-Term Memory, and it is a type of Recurrent Neural Network (RNN) architecture used for processing sequential data like time series or text.

- LSTM unlike traditional RNN's have memory units which can store important information for longer period this makes them ideal to handle sequential data which have long term dependencies like time series data.

- This helps the network in classifying, processing and make more accurate predictions on time series data.



```python
#ML model
from keras.layers import Dense,Dropout,LSTM
from keras.models import Sequential
model = Sequential()
model.add(LSTM(units=50,activation='relu',return_sequences = True,
               input_shape=(x_train.shape[1],1)))
model.add(Dropout(0.2))
model.add(LSTM(units=60,activation='relu',return_sequences = True))
model.add(Dropout(0.3))

model.add(LSTM(units=80,activation='relu',return_sequences = True,
               ))
model.add(Dropout(0.4))
model.add(LSTM(units=120,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=1))
```

This code creates a sequential model using the Keras library and adds several layers to the model.

- The first line creates an instance of the Sequential model.

- The next line uses the model.add method to add an LSTM layer with 50 units and the ReLU (rectified linear unit) activation function to the model.

- The input_shape argument specifies the shape of the input data, where x_train.Shape is the number of time steps and 1 is the number of features. The return_sequences argument is set to True, which means that the output from this layer will be passed to the next layer in the sequence.

- The next line adds a Dropout layer with a rate of 0.2 to the model. This layer randomly drops out 20% of the neurons during training to prevent overfitting.

- The next two lines repeat the process of adding an LSTM layer followed by a Dropout layer with increased units and dropout rate.

- The last two lines add another LSTM layer with 120 units and a ReLU activation function and a Dropout layer with a rate of 0.5 to the model.

- The final line adds a Dense layer with 1 unit to the model, which is used to make the final prediction.

- In summary, this code creates a sequential model with several LSTM and Dropout layers that can be used for sequence modeling tasks.

ALGORITHM

Here is a step-by-step algorithm for the code:

- Import the necessary libraries: yfinance, numpy, pandas,matplotlib, pandas_datareader, keras, and streamlit.

- Define the start and end dates for retrieving stock data: start='2012-01-01', end='2022-12-31'.

- Use streamlit to create a user interface and define a title for the app: st.title('Stock Trend Prediction').

- Ask the user to input a stock ticker using the text_input function and store the input in a variable user_input.

- Retrieve the stock data using yfinance.download and pass the user_input and the start and end dates as arguments. Store the data in a variable df.

- Use streamlit to display the summary statistics of the data using st.write(df.describe()).

- Plot the closing price of the stock over time using matplotlib and display it using st.pyplot.

- Plot the 100-day moving average of the closing price and display it using st.pyplot.

- Plot the 100-day and 200-day moving average of the closing price and display it using st.pyplot.

- Split the data into training and testing sets: data_training and data_testing.

- Scale the data using the MinMaxScaler function and store it in data_training_array.

- Load a pre-trained model using load_model and store it in a variable model.

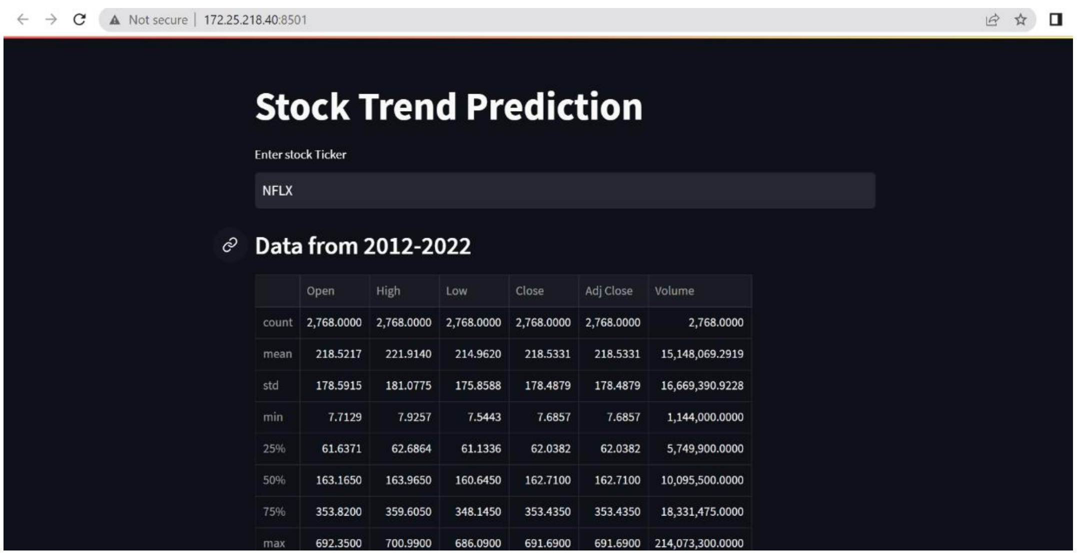
- Prepare the testing data by using the last 100 days of the training data and appending the testing data to it. Scale the testing data using the MinMaxScaler function.
- Use a loop to split the testing data into windows of 100 days and store the windows in x_test. Store the next day's closing price in y_test.

- Use the model to predict the next day's closing price using model.predict and store the result in y_predicted.

- Scale y_predicted and y_test back to their original scale.

- Plot the original and predicted closing prices using matplotlib and display the plot using st.pyplot.

INTERFACE OF WEB APPLICATION


Not secure | 172.25.218.40:8501

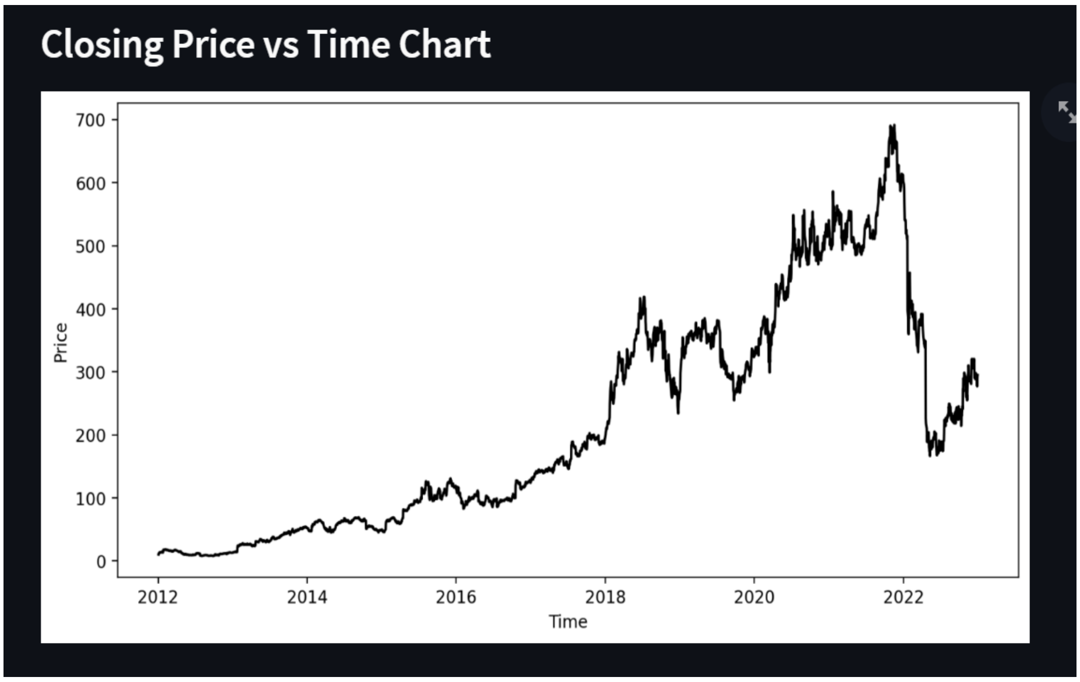
Stock Trend Prediction

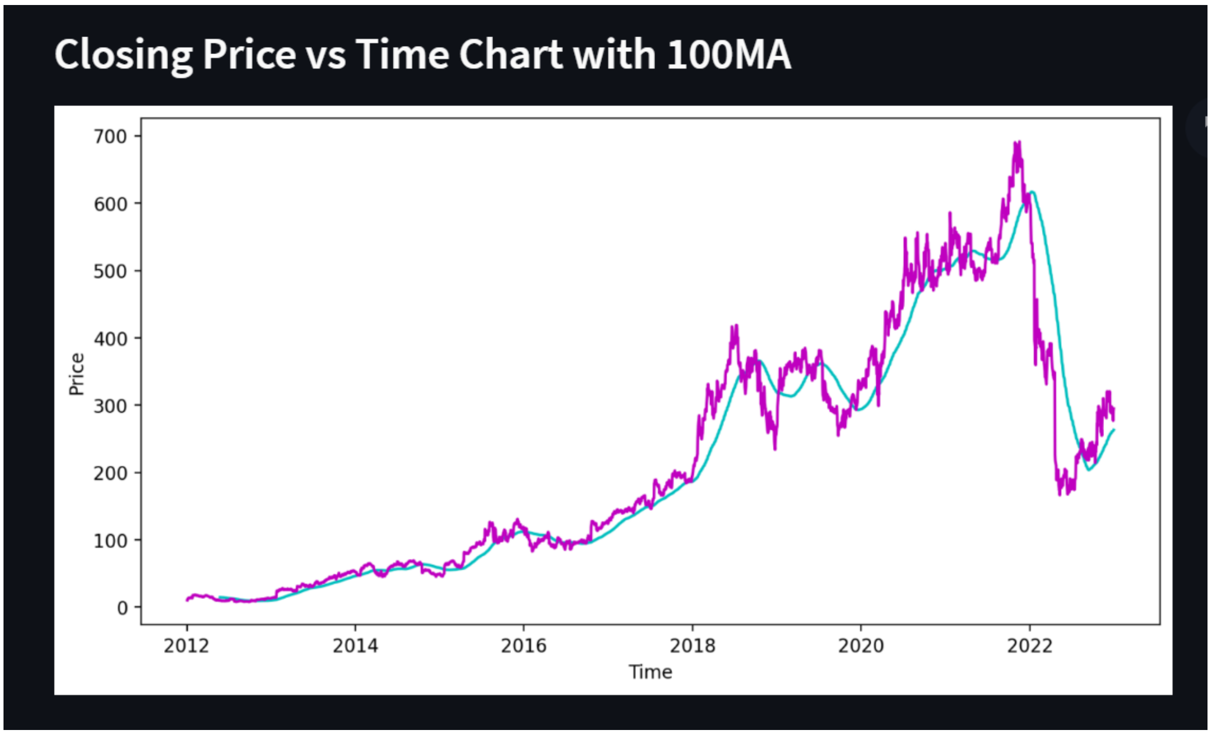
Enter stock Ticker

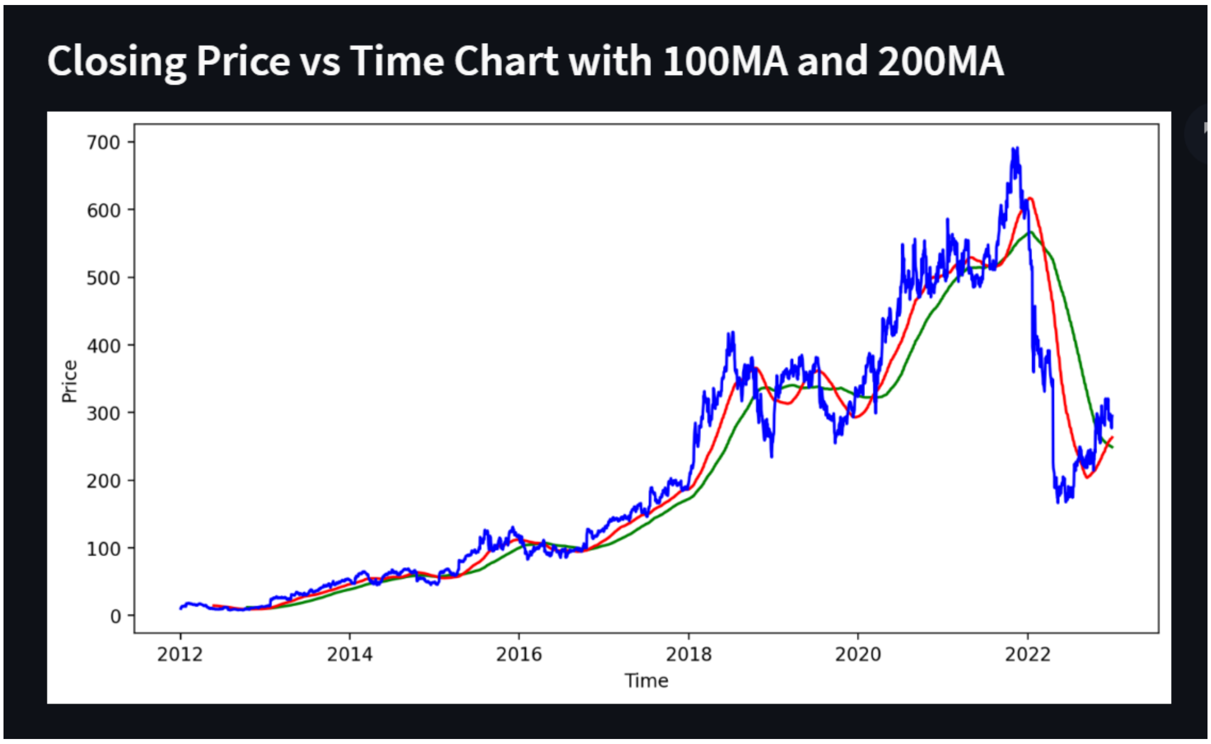
NFLX

Data from 2012-2022

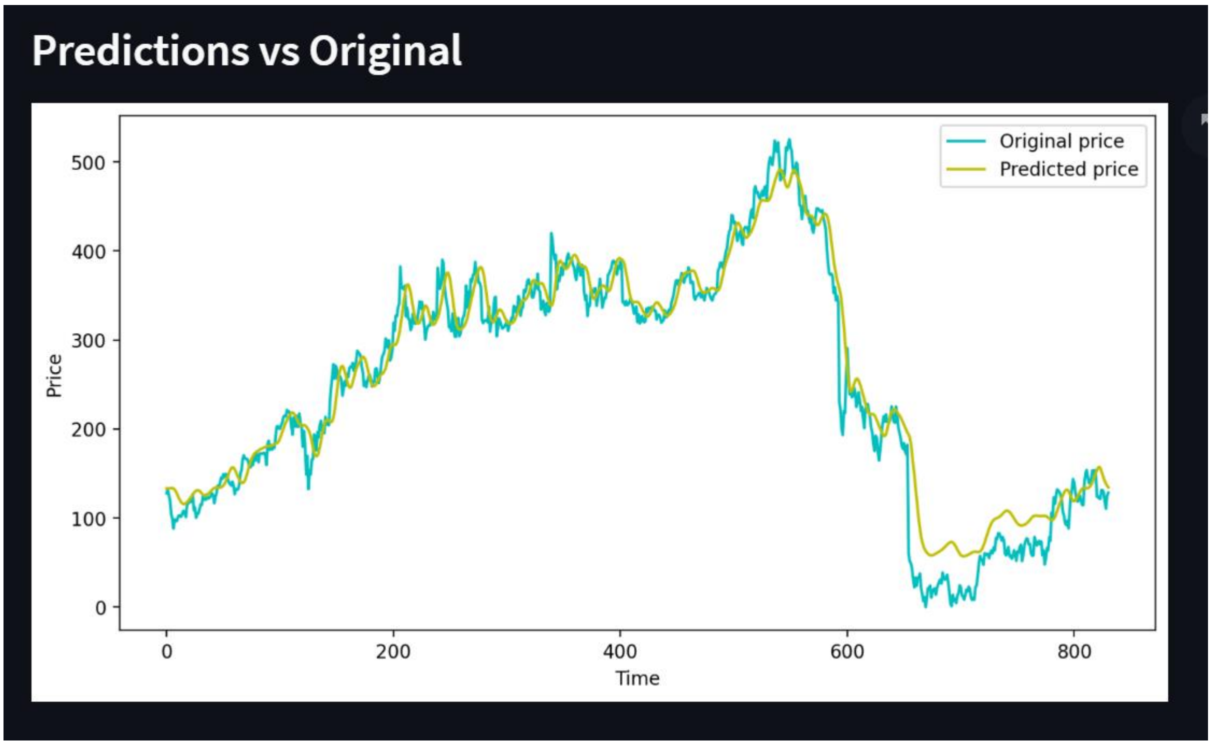
	Open	High	Low	Close	Adj Close	Volume
count	2,768.0000	2,768.0000	2,768.0000	2,768.0000	2,768.0000	2,768.0000
mean	218.5217	221.9140	214.9620	218.5331	218.5331	15,148,069.2919
std	178.5915	181.0775	175.8588	178.4879	178.4879	16,669,390.9228
min	7.7129	7.9257	7.5443	7.6857	7.6857	1,144,000.0000
25%	61.6371	62.6864	61.1336	62.0382	62.0382	5,749,900.0000
50%	163.1650	163.9650	160.6450	162.7100	162.7100	10,095,500.0000
75%	353.8200	359.6050	348.1450	353.4350	353.4350	18,331,475.0000
max	692.3500	700.9900	686.0900	691.6900	691.6900	214,073,300.0000




Closing Price vs Time Chart with 100MA
700
600
500
400
300
200
100
0
Price
2012
2014
2016
2018
2020
2022
Time


Closing Price vs Time Chart with 100MA and 200MA
700
600
500
400
300
200
100
0
Price
2012
2014
2016
2018
2020
2022
Time

Final result prediction vs original prices


Predictions vs Original
Original price
Predicted price
Price
Time