Stock Price Prediction

Development Part-1

Datset and its details:

Stock price prediction is a crucial aspect of financial analysis and investment decision-making, and it relies heavily on historical market data. The dataset you've mentioned, which typically includes columns for Open, High, Low, Close, Adjusted Close, and Volume, is fundamental to this process. Each of these columns provides essential information for analyzing and predicting stock prices.

Open Price: The Open column represents the price of a stock when the market opens for a particular trading session. It's the first price at which the stock is traded for that day. This value is essential because it can indicate market sentiment and investor expectations at the beginning of a trading day.

High Price: The High column records the highest price at which the stock traded during a specific time period, typically within a trading day. This information is valuable for identifying intraday trends, volatility, and potential resistance levels.

Low Price: The Low column, in contrast to the High, captures the lowest price at which the stock traded during the same time period. It helps investors understand the lowest points the stock reached during the trading session and can be useful for identifying support levels.

Close Price: The Close column reflects the final price at which the stock was traded during a given trading session. It's a critical data point for calculating daily returns and for assessing whether the stock ended the day on a positive or negative note.

Adjusted Close Price: The Adjusted Close column considers corporate actions, such as stock splits, dividends, and other adjustments that can affect stock prices. It's crucial for accurate historical data analysis, as it provides a more accurate representation of the stock's performance over time.

Volume: The Volume column shows the total number of shares traded during a specific trading session. It's an indicator of market activity and liquidity. High volume can indicate increased interest in a stock, and low volume may suggest a lack of interest or potential price manipulation.

The combination of these columns in a dataset enables analysts and machine learning algorithms to make predictions about a stock's future performance. Various techniques and models can be employed for stock price prediction, including time series analysis, machine learning, and deep learning.

Loading the dataset:

Loading a dataset for stock price prediction in Python typically involves using a library like Pandas to read data from a CSV file or an online source. Below, An example of how to load a dataset of historical stock prices using Python. For this example, use the pandas library to load data from a CSV file,

Here are the steps to load a dataset for stock price prediction:

Install Necessary Libraries:

Make sure you have the required libraries installed. You can install them using pip if you haven't already:

pip install pandas

Import Libraries:

Import the necessary libraries at the beginning of your Python:

import pandas as pd

A stock price dataset from a CSV file or from an online source. Here, I'll provide an example of loading data from a CSV file.

a. From a CSV File:

Assuming you have a CSV file containing historical stock price data, you can load it like this:

Replace 'your_dataset.csv' with the actual filename

file_path = 'stock_dataset.csv'

Read the CSV file into a Pandas DataFrame

df = pd.read_csv(file_path)

Code:

```
import pandas as pd

# Load the dataset (replace 'stock_data.csv' with your dataset file)

data = pd.read_csv('stock_data.csv')

# Display the first few rows of the dataset

print(data.head())
```

Output:

	Date	Open	High	Low	Close	Volume
0	2023-01-02	140.000000	142.000000	139.000000	141.000000	500000
1	2023-01-03	141.500000	143.000000	140.000000	142.500000	600000
2	2023-01-04	143.000000	144.500000	141.500000	143.500000	700000
3	2023-01-05	144.000000	145.500000	142.500000	144.000000	800000
4	2023-01-06	143.500000	144.000000	141.500000	142.000000	900000

Explore the Data:

Once you've loaded the dataset, you can explore it by using various Pandas functions. For example, you can check the first few rows of the dataset using df.head() to ensure that the data is loaded correctly.

Code:

```
import pandas as pd

import matplotlib.pyplot as plt
```

```
import seaborn as sns
# Load the dataset (replace 'stock_data.csv' with your dataset file)
data = pd.read_csv('stock_data.csv')
# Display basic summary statistics of the dataset
summary_stats = data.describe()
print("Summary Statistics:\n", summary_stats)
# Display the first few rows of the dataset
print("\nFirst Few Rows:")
print(data.head())
# Check for missing values
missing_values = data.isnull().sum()
print("\nMissing Values:\n", missing_values)
# Visualize the 'Close' price over time
plt.figure(figsize=(12, 6))
plt.plot(data['Date'], data['Close'])
plt.title('Stock Price Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.grid(True)
plt.show()
# Visualize a correlation heatmap of numerical columns
corr_matrix = data.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.title('Correlation Heatmap')
plt.show()
```

Output:

	Date	Open	High	Low	Close	Volume
0	2023-01-02	140.000000	142.000000	139.000000	141.000000	500000
1	2023-01-03	141.500000	143.000000	140.000000	142.500000	600000
2	2023-01-04	143.000000	144.500000	141.500000	143.500000	700000
3	2023-01-05	144.000000	145.500000	142.500000	144.000000	800000
4	2023-01-06	143.500000	144.000000	141.500000	142.000000	900000

Preprocess Dataset:

Preprocessing a dataset for stock price prediction involves several steps to prepare the data for use in a machine learning model. These steps include data loading, cleaning, normalization, feature engineering, and splitting the data into training and testing sets. Below, I'll provide an example using Python and the popular libraries NumPy, Pandas, and Scikit-Learn.

Code:

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import MinMaxScaler

# Load your stock price data into a Pandas DataFrame

data = pd.read_csv("stock_price_data.csv")  # Replace with the actual file path

# Data Cleaning

data.dropna(inplace=True)  # Remove rows with missing data

# Feature Engineering

# You can create new features like moving averages, technical indicators, or use historical data as features

# For simplicity, we'll use only 'Close' price as a feature

# Normalize the data
```

```
scaler = MinMaxScaler()
data['Close'] = scaler.fit_transform(data['Close'].values.reshape(-1, 1))
# Define the number of previous days' prices to use as features
look_back = 10
# Create a new DataFrame with lag features
for i in range(1, look_back + 1):
    data[f'Close_Lag_{i}'] = data['Close'].shift(i)
# Remove rows with NaN values introduced by the lag
data.dropna(inplace=True)
# Split the data into training and testing sets
X = data.drop('Close', axis=1)  # Features
y = data['Close']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)
# At this point, you have preprocessed data and can use it for training your
stock price prediction model.
```

Output:

1986-03-14

0.097222

0.102431

0.097222

0.100694

0.064783

308160000

Performing analysis in Dataset:

Performing stock price prediction typically involves using historical stock price data and various data analysis techniques and models.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Load the dataset (replace 'stock_data.csv' with your dataset file)
data = pd.read_csv('stock_data.csv')
# Display basic statistics
statistics = data.describe()
print("Basic Statistics:\n", statistics)
# Calculate daily returns
data['Daily_Return'] = data['Close'].pct_change()
# Display daily returns statistics
returns_statistics = data['Daily_Return'].describe()
print("\nDaily Returns Statistics:\n", returns_statistics)
# Visualize the 'Close' price over time
plt.figure(figsize=(12, 6))
plt.plot(data['Date'], data['Close'])
plt.title('Stock Price Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.grid(True)
plt.show()

# Visualize daily returns distribution
plt.figure(figsize=(8, 4))
sns.histplot(data['Daily_Return'].dropna(), kde=True, color='blue')
```

```
plt.title('Daily Returns Distribution')
plt.xlabel('Daily Return')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
# Calculate and visualize rolling mean (e.g., 7-day moving average)
data['7D_MA'] = data['Close'].rolling(window=7).mean()
# Visualize the 7-day moving average
plt.figure(figsize=(12, 6))
plt.plot(data['Date'], data['Close'], label='Close Price')
plt.plot(data['Date'], data['7D_MA'], label='7-Day MA', linestyle='--',
color='orange')
plt.title('Stock Price and 7-Day Moving Average')
plt.xlabel('Date')
plt.grid(True)
plt.legend()
plt.show()
```

Output:

	Open	High	Low	Close	Volume
count	100.000000	100.000000	100.000000	100.000000	1.000000e+02
mean	150.300000	151.200000	149.200000	150.200000	8.000000e+06
std	6.943497	7.287662	6.521416	6.401613	2.041241e+06
min	140.000000	141.000000	139.000000	140.000000	5.000000e+06
25%	144.750000	145.750000	142.750000	144.750000	6.500000e+06
50%	150.000000	151.000000	149.000000	150.000000	7.750000e+06
75%	155.250000	156.000000	153.250000	154.500000	9.000000e+06

max	160.000000	162.000000	158.000000	161.000000	1.200000e+07