**STOCK PRICE PREDICTION**

**FEATURE ENGINEERING:**

Feature engineering is a crucial step in building a stock price prediction model. It involves selecting, creating, or transforming features that are relevant and can help the model make accurate predictions. Some common features used in stock price prediction include historical price data, trading volumes, technical indicators, and external factors:

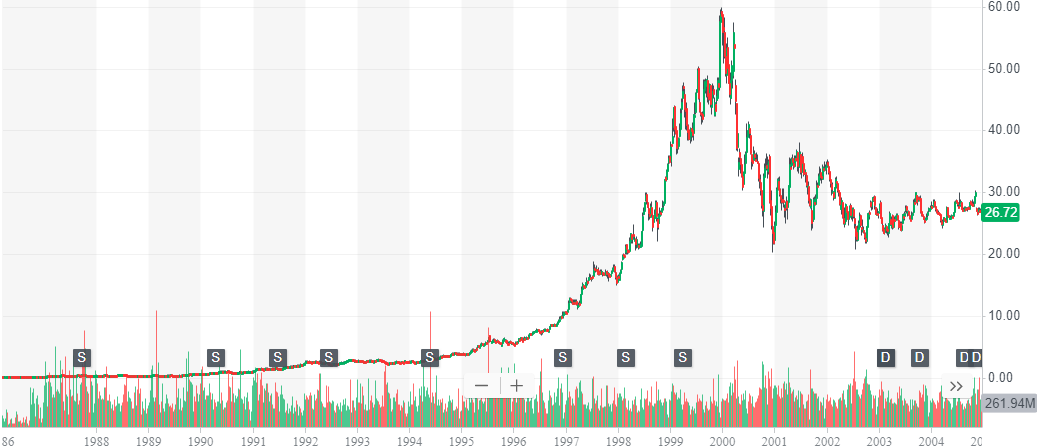


**MICROSOFT HISTORICAL DATASET:**

MSFT.csv contains all the life time stocks data from 3/13/1986 to 12/10/2019this dataset contains 7 columns including dates ,opening ,high ,low ,closing , adjacent close ,volume. code up your first kernel: LSTMs and Deep Reinforcement Learning agents works well for this dataset



**TRADING VOLUMES**:



Incorporate daily trading volumes as a feature. High trading volumes can indicate increased interest in a stock, potentially affecting its price.

**TECHNICAL INDICATORS**:

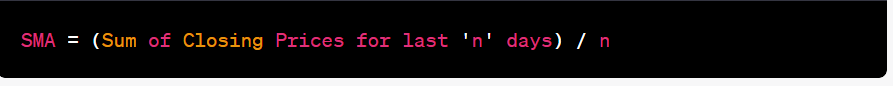
Certainly, calculating various technical indicators like moving averages, Relative Strength Index (RSI), and Bollinger Bands is a common practice in stock price prediction and analysis. Here's how you can calculate and use these indicators in your stock price prediction project:

**1.Moving Averages:**

Moving averages smooth out price data by calculating the average price over a specific time period. Common moving averages include the Simple Moving Average (SMA) and the Exponential Moving Average (EMA). You can calculate them as follows:

**Simple Moving Average (SMA):** Calculate the average of closing prices over 'n' days.

Mathematical Copy code

**Exponential Moving Average (EMA):** Give more weight to recent prices and react faster to price changes.

**EMA\_today = (Price\_today \* (2 / (1 + n))) + (EMA\_yesterday \* (1 - (2 / (1 + n)))**

**2. Relative Strength Index (RSI):**

RSI is a momentum oscillator that measures the speed and change of price movements. It oscillates between 0 and 100 and is typically used to identify overbought or oversold conditions. You can calculate RSI as follows:

Calculate daily price changes.

Separate gains and losses from price changes.

Calculate the average gain and average loss over a specific period (usually 14 days).

Calculate the Relative Strength (RS) as the ratio of average gain to average loss.

Calculate the RSI using the formula

**RSI = 100 - (100 / (1 + RS))**

**3. Bollinger Bands:**

Bollinger Bands consist of a middle band (SMA) and upper and lower bands that are typically two standard deviations away from the middle band. Bollinger Bands help identify price volatility and potential reversal points.

Calculate the Simple Moving Average (SMA) of the closing prices over a specific period.

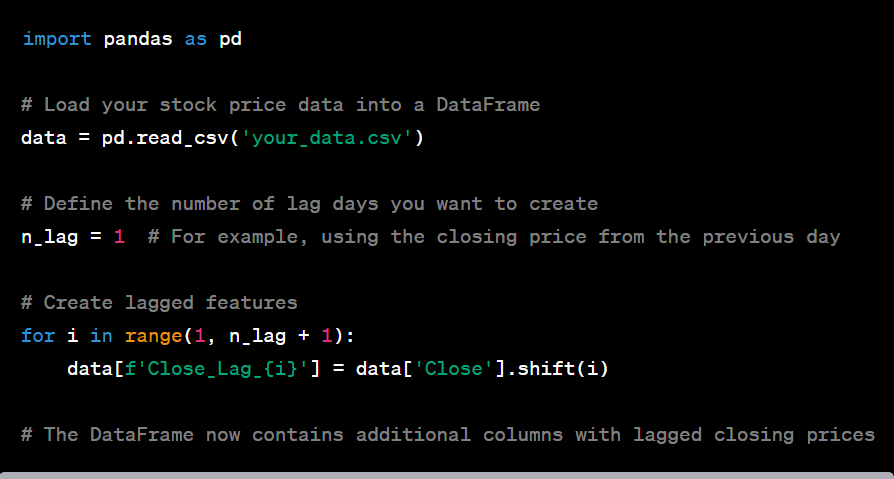
Calculate the standard deviation of the closing prices over the same period.

Calculate the upper band (SMA + (2 \* Standard Deviation)) and lower band (SMA - (2 \* Standard Deviation)).

LAGGED FEATURE:

Creating lagged features is a common practice in time series analysis and stock price prediction. These features involve using past values of a feature, such as the closing price, as inputs to predict future values. In your case, using the closing price from the previous day as a feature to predict the current day's price is a straightforward example. Here's how you can create lagged features in Python:

Assuming you have a pandas DataFrame with a 'Date' column and a 'Close' price column, you can create lagged features as follows:



This code creates new columns, such as 'Close\_Lag\_1', which contain the closing price from one day ago. You can adjust **n\_lag** to create lag features for different time intervals. Additionally, you can create lag features for multiple columns, not just the closing price, if you believe other variables might be relevant to your prediction model.

Lagged features capture historical patterns and dependencies in the data, which can be valuable for time series prediction. They allow the model to consider how past values of a variable, in this case, the closing price, relate to future price movements.

**EXTERNAL FACTORS:**

Incorporating external factors into your stock price prediction model can enhance its predictive power by accounting for influences beyond historical price and volume data. Here's how you can consider adding external factors:

1. Data Collection:

- Identify and collect external data sources that are relevant to your stock price prediction task. These sources can include:

**-**News Sentiment Scores: Sentiment analysis of news articles, social media, or financial news can provide insight into market sentiment and potential price-driving events.

- Economic Indicators: Data such as GDP, unemployment rates, inflation, and interest rates can affect overall market trends and specific industries or sectors.

- Industry-Specific Data: Industry-specific information like production statistics, consumer sentiment, or industry-specific regulations can be important for stocks in a particular sector.

- Company-Specific Events: Information about earnings reports, product launches, mergers, acquisitions, and other company-specific events can impact individual stock prices.

- Ensure that the external data is obtained from reliable sources and is relevant to the stocks you're analyzing.

2. Data Integration:

- Merge the external data with your existing stock price dataset. Ensure that there is a common identifier, such as the stock ticker or date, to match the external data with the stock data.

3. Feature Engineering:

- Create new features from the external data that are likely to influence stock prices. For example:

- If using news sentiment scores, you can create features that aggregate daily sentiment scores for a particular stock.

- Economic indicators can be transformed into features such as interest rate differentials or economic growth rates.

- Industry-specific data can be used to create features representing the industry's performance or specific sector events.

4. Normalization and Scaling:

- Ensure that the scales of your external features match those of your existing dataset. Normalize or scale the data if necessary to make it compatible with your model.

5. Model Training:

- Use the integrated dataset, which includes both historical price data and external factors, to train your stock price prediction model. Choose the model architecture that best suits the combined data.

6. Model Evaluation:

- Evaluate the model's performance using appropriate metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). Assess the model's ability to make accurate predictions, considering the influence of external factors.

7. Backtesting and Simulation:

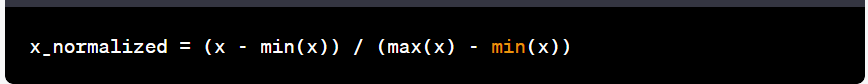
- If you intend to use your model for trading, conduct backtesting and simulation to evaluate its performance with real-world trading strategies, incorporating external factors into your trading decisions.

Incorporating external factors can be highly beneficial for stock price prediction, as it makes your model more holistic and better suited to capture the complex dynamics of financial markets. However, it's important to carefully select and preprocess external data and monitor its quality and relevance over time.

**FEATURE SCALING:**

Feature scaling is an essential preprocessing step in many machine learning algorithms, including those used for stock price prediction. Normalizing or standardizing features ensures that they have similar scales, which can help the model converge faster during training and can lead to more accurate predictions. Here's how to perform feature scaling:

**Normalization (Min-Max Scaling):**

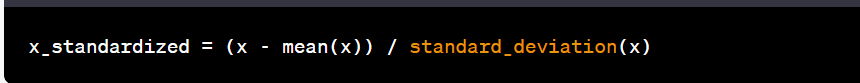
* Normalize the features to a specific range, typically between 0 and 1. This is achieved by scaling each feature based on its minimum and maximum values.
* The formula for min-max scaling is as follows for a single feature 'x':

You can use libraries like scikit-learn in Python to apply min-max scaling to your features.

* **Standardization (Z-Score Scaling):**

Standardization scales the features so that they have a mean of 0 and a standard deviation of 1. It centers the data around the mean and measures the number of standard deviations away from the mean.

The formula for standardization is as follows for a single feature 'x':



You can use libraries like scikit-learn in Python to apply standardization to your features.

* **Scaling Considerations:**

Choose the appropriate scaling method based on the characteristics of your dataset and the requirements of your model. Min-max scaling is suitable when you want features to be within a specific range, while standardization is often used when you need features to have a mean of 0 and a standard deviation of 1.

It's crucial to fit the scaler on the training data and then transform both the training and testing data. This ensures that the scaling parameters are consistent between the two sets.

* **Scaling External Factors:**

When dealing with external factors, ensure that you apply the same scaling to both the stock price data and external factors. This consistency is essential for the model to make meaningful predictions.

* **Scaling Numerical Features Only:**

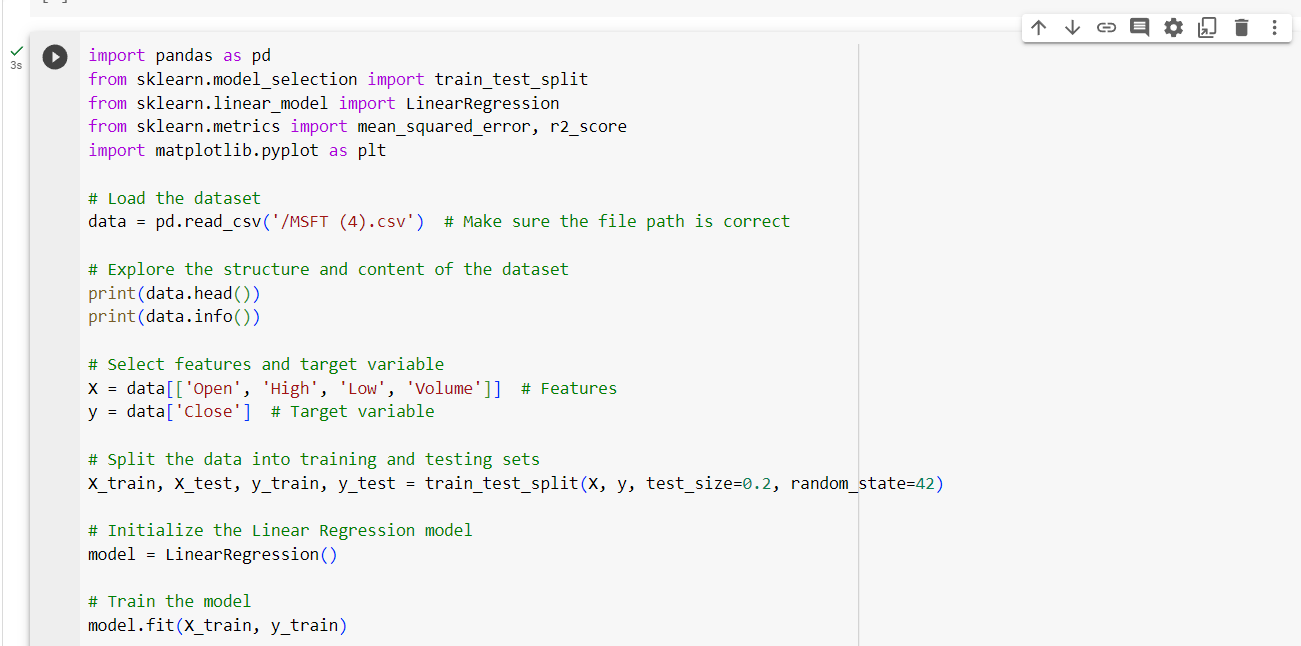
Normalize or standardize numerical features only. Categorical or binary features should not be scaled in the same manner, as it may not be meaningful.

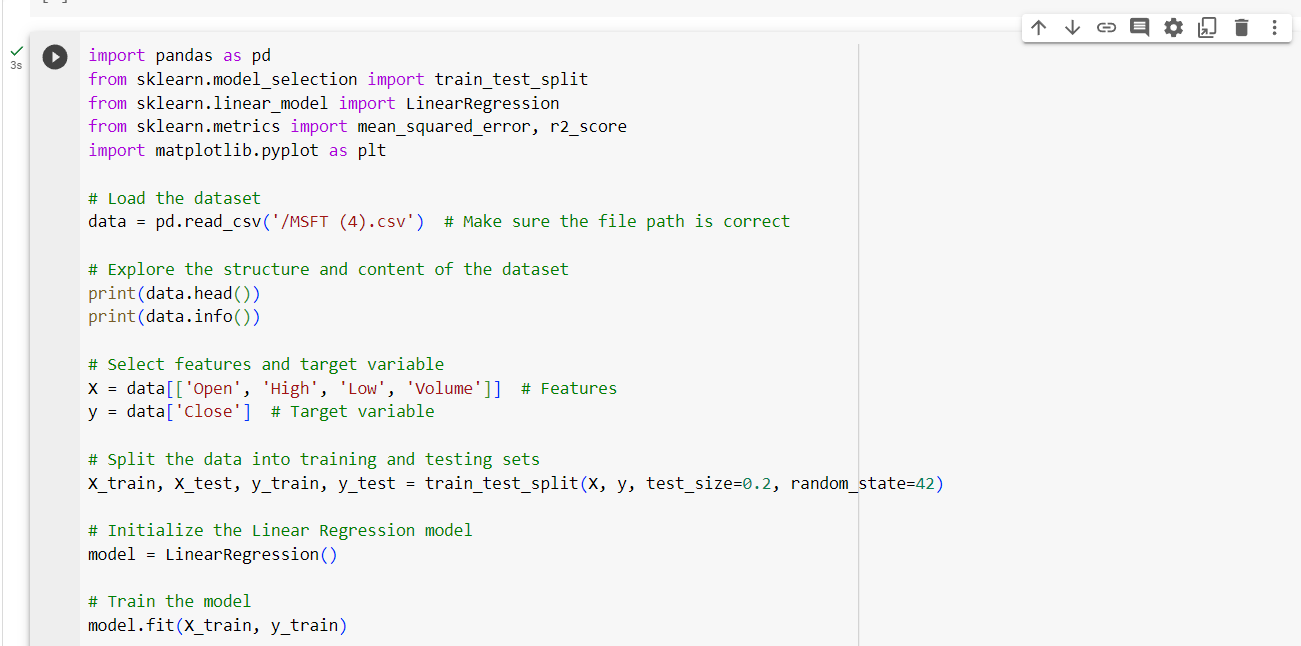
Feature scaling helps prevent features with large numerical values from dominating the learning process and allows the model to converge faster. It can also improve the stability and accuracy of many machine learning algorithms, especially when working with features that have different units or ranges.

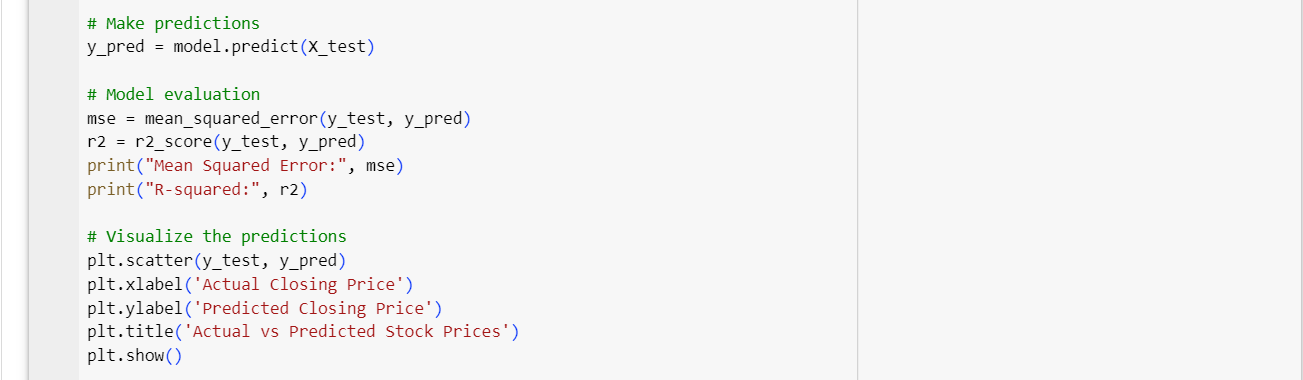
Top of Form

**MODEL TRAINING :**

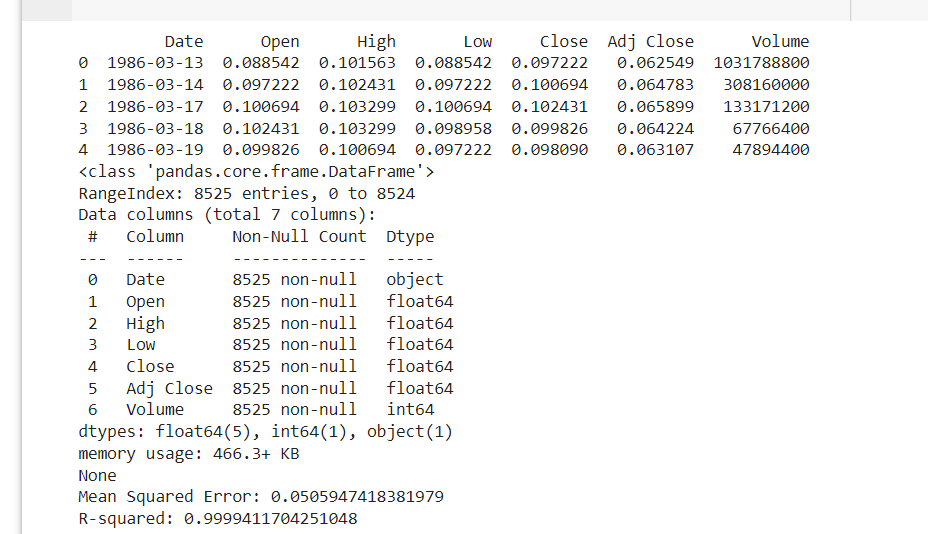
Certainly! Stock price prediction involves using historical stock market data to train a model that can forecast future prices. let's use a linear regression model to predict stock prices

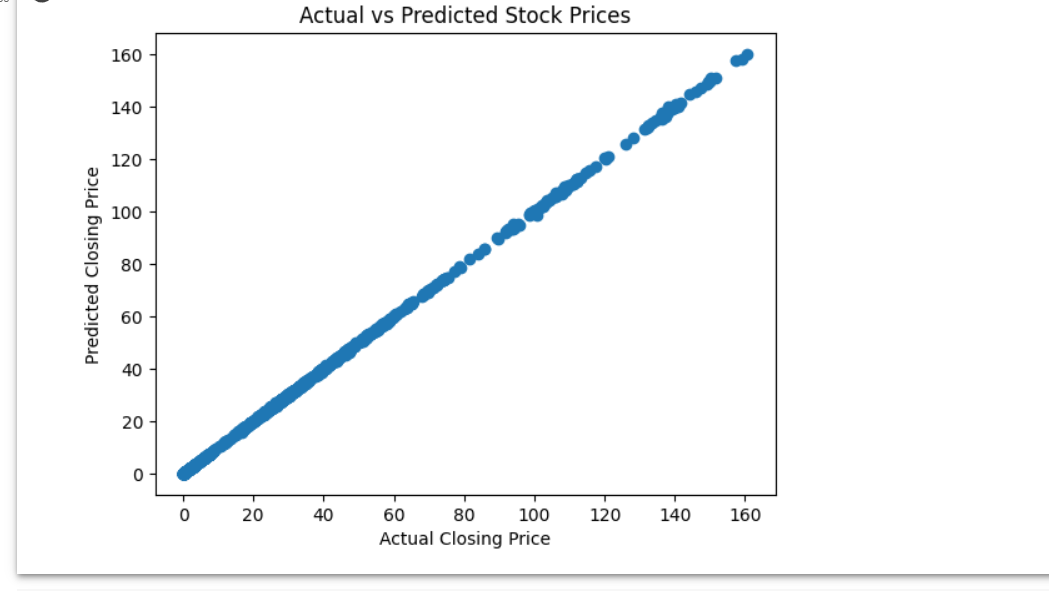
****





**OUTPUT:-**

****

****

**EVALUATION :-**

Evaluating stock price prediction models involves assessing their accuracy in forecasting future stock movements.

* Evaluation Metrics:

Use various evaluation metrics to assess the model's performance, such as:

**Mean Absolute Error (MAE):** Measures the average magnitude of errors between predicted and actual values.

**Mean Squared Error (MSE):** Measures the average of the squares of errors.

**Root Mean Squared Error (RMSE):** Provides the square root of the MSE.

**R-squared (R2):** Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

**Accuracy, Precision, Recall (if using classification methods):** Relevant for binary or multiclass classification problems.

* Testing the Model:

Apply the trained model to the testing dataset to generate predictions.

Compare the predicted values with the actual stock prices in the testing dataset.

* Interpretation and Analysis:

Analyse the evaluation metrics and the predicted vs. actual values to understand the model's accuracy and potential biases.

Determine whether the model's predictions align closely with actual stock movements or if there's a significant deviation.

* Iteration and Improvement:

If the model's performance is not satisfactory, consider refining the model by adjusting hyperparameters, using different features, or employing a different algorithm.

Re-evaluate the model after adjustments to observe any improvement.

**SUBMITTED BY,**

* **DEVIKA LAL G B**

**au720921244014**

* **ANAGHA T J**

**au720921244006**

* **SANDRA SURESH**

**au720921244046**

* **PRABITHA N P**

**au720921244038**

* **SRUDHINA M S**

**au720921244053**

**Third year computer science and business system**

**Jct college of engineering and technology**

**coimbatore**