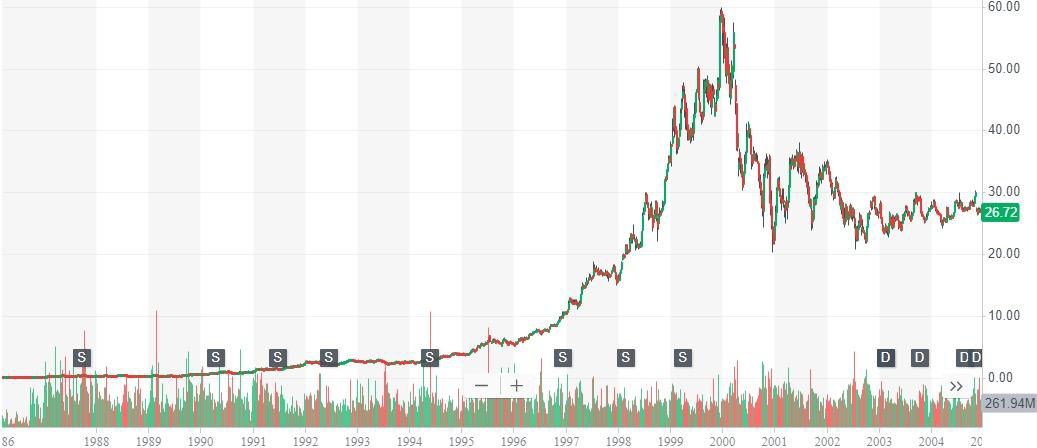
**Stock price prediction**

**312621104025:Sakthivel**

The provided code is a simple example of a stock price prediction model using Python and scikit-learn. It follows the basic steps of building a machine learning model for stock price prediction, but please keep in mind that this is a simplified illustration and may not produce accurate predictions for actual trading. Here's an explanation of each part of the code:



**1. Import Required Libraries:**

```python

import numpy as np

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

import yfinance as yf

```

- NumPy and Pandas are used for data manipulation and analysis.

- `train\_test\_split` from scikit-learn is used to split the data into training and testing sets.

- `LinearRegression` from scikit-learn is used to create a linear regression model.

- `yfinance` is used to fetch historical stock price data from Yahoo Finance.

**2. Define Stock Symbol and Date Range:**

```python

stock\_symbol = 'AAPL'

start\_date = '2020-01-01'

end\_date = '2021-12-31'

```

These variables specify the stock symbol (in this case, Apple Inc.) and the date range for which historical stock price data will be downloaded.

**3. Download Historical Stock Data:**

```python

data = yf.download(stock\_symbol, start=start\_date, end=end\_date)

```

This code snippet uses the Yahoo Finance API to download historical stock price data for the specified stock symbol and date range.

**4. Calculate Daily Returns:**

```python

data['Daily Return'] = data['Adj Close'].pct\_change()

data = data.dropna()

```

It calculates daily returns by taking the percent change in the adjusted closing price. The `dropna` method is used to remove any rows with missing data.

**5. Feature Engineering:**

```python

num\_days = 10

for i in range(1, num\_days + 1):

data[f'Day-{i}'] = data['Daily Return'].shift(i)

data = data.dropna()

```

This section creates input features for the machine learning model. It adds columns for the daily returns of the previous `num\_days` days as input features.

**6. Split Data into Training and Testing Sets:**

```python

X = data.iloc[:, 6:]

y = data['Daily Return']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

```

The data is split into input features (`X`) and the target variable (`y`). Then, the data is further divided into training and testing sets for model evaluation.

**7. Create and Train a Linear Regression Model:**

```python

model = LinearRegression()

model.fit(X\_train, y\_train)

```

A simple linear regression model is created and trained using the training data.

**8. Make Predictions:**

```python

predictions = model.predict(X\_test)

```

The model is used to make predictions on the testing data.

**9. Calculate Model Performance:**

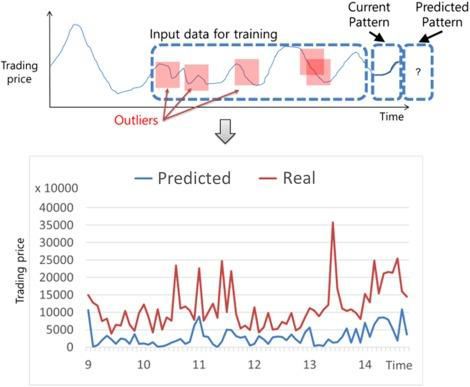
```python

from sklearn.metrics import mean\_squared\_error, r2\_score

mse = mean\_squared\_error(y\_test, predictions)

r2 = r2\_score(y\_test, predictions)

```



The code calculates the Mean Squared Error (MSE) and R-squared (R2) to evaluate the model's performance on the test data. These metrics are common for regression models and provide insights into how well the model fits the data.

Keep in mind that this is a simplified example. Real-world stock price prediction models are far more complex, involving a wide range of features and often using advanced machine learning techniques. Additionally, stock market predictions are subject to many uncertainties and should be used cautiously for financial decision-making.

**Conclusion**

The provided code is a basic example of a stock price prediction model using historical stock price data and a linear regression model. Here are the key takeaways from the code:

1. Data Retrieval:he code uses the `yfinance` library to fetch historical stock price data for a specified stock symbol and date range from Yahoo Finance.

2. Feature Engineering:It calculates daily returns and creates input features based on the previous days' returns. These features are used to predict future returns.

3. Data Splitting: The data is divided into training and testing sets. This is a crucial step to evaluate the model's performance and avoid overfitting.

4. Model Building: A basic linear regression model is created and trained using the training data. Linear regression is a simple model for predicting numeric values based on input features.

5. Prediction: The trained model is used to make predictions on the testing data.

6. Performance Evaluation: The Mean Squared Error (MSE) and R-squared (R2) metrics are calculated to assess the model's performance. MSE measures the model's prediction error, while R2 measures how well the model fits the data.

It's important to note that this code is a simplified illustration and may not be suitable for real-world trading decisions. Stock price prediction is a complex task and often requires more sophisticated models, a wider range of features (including sentiment analysis of news and other external factors), and rigorous testing to assess its accuracy. Additionally, investing in the stock market carries inherent risks, and predictions should be used cautiously.