**[Introduction]**

"Hello everyone! In this video, I’ll be walking you through a Python script designed to predict stock prices using a machine learning approach. Let’s break it down step by step to understand how the code works and what each part does.

**[Downloading Data]**

We begin with downloading historical stock data. In our download\_data function, we use the Yahoo Finance API to fetch stock data for a given company ticker—in this case, we use Apple, with the ticker 'AAPL'. This function collects the stock prices between 2020 and 2024, which will be used to train our machine learning model.

**[Preprocessing Data]**

Next, in the preprocess\_data function, we clean the data. Any missing values are filled using forward fill. Then, we normalize the features such as 'Open', 'High', 'Low', 'Close', and 'Volume' using MinMaxScaler. This scaling ensures that the values are on the same scale, which is essential for our model to perform well.

**[Feature Engineering]**

After that, we add technical indicators through the feature\_engineering function. This step enhances our model’s ability to predict stock prices. We compute a 20-day moving average and the Relative Strength Index, or RSI, which are key features in predicting trends in stock prices. The moving average smooths out fluctuations, and the RSI helps identify whether a stock is overbought or oversold.

**[Training the Model]**

Now, let’s train the model. In the train\_model function, we use the Random Forest Regressor, a powerful ensemble learning method, to train on the stock features like opening price, RSI, moving average, and others. This model will help us predict future stock prices.

**[Evaluating the Model]**

Once the model is trained, we evaluate its performance. The evaluate\_model function tests the model by comparing predicted prices with actual prices. We calculate two metrics: Mean Squared Error and Mean Absolute Error. These values help us understand how accurate the model is.

To make it easier to visualize, the function also plots two separate graphs. The first graph shows the actual stock prices, while the second displays the predicted stock prices. This allows us to clearly see how well the model’s predictions align with the real market data.

**[Merge Sort Demonstration]**

We’ve also included a classic algorithm in the code for demonstration—merge\_sort. This function sorts an array of numbers by breaking it into smaller pieces and merging them back together in the correct order. Although this isn't related to stock prediction, it's an example of how algorithms work in Python.

**[Conclusion]**

So, in summary, this Python script downloads stock data, preprocesses it, adds important features, trains a Random Forest model, and then evaluates how well it predicts future stock prices. The final output is a clean and easy-to-read graph that helps us visualize both actual and predicted prices side by side.

That’s it for this explanation! I hope you found it helpful. Thank you for watching, and feel free to try this code out on your own!"

**[End of Script]**