**stock analysis preprocessing, data visualization and feature engineering**

# Install required libraries

!pip install yfinance # Installs the Yahoo Finance API package

!pip install tensorflow # Installs TensorFlow for LSTM modeling

# Import necessary libraries

import pandas as pd # For data manipulation

import numpy as np # For numerical operations

import matplotlib.pyplot as plt # For plotting

from sklearn.preprocessing import MinMaxScaler # For normalizing data

import yfinance as yf # To fetch financial data

# Define functions for preprocessing and analysis

def load\_stock\_data(ticker, start\_date, end\_date):

"""Fetch historical stock data using Yahoo Finance API."""

data = yf.download(ticker, start=start\_date, end=end\_date) # Download stock data

return data

def preprocess\_data(data):

"""Preprocess stock data."""

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

# Scale 'Close' prices and 'Volume'

scaler = MinMaxScaler() # Initialize scaler

data['Scaled\_Close'] = scaler.fit\_transform(data[['Close']]) # Scale Close price

data['Scaled\_Volume'] = scaler.fit\_transform(data[['Volume']]) # Scale Volume

# Create binary label for price movement

data['Price\_Up'] = (data['Close'].shift(-1) > data['Close']).astype(int) # 1 if price goes up next day

# Calculate short-term (40-day) and long-term (100-day) SMAs

data['SMA\_40'] = data['Close'].rolling(window=40).mean() # 40-day SMA

data['SMA\_100'] = data['Close'].rolling(window=100).mean() # 100-day SMA

# Generate trading signals based on SMA crossovers

data['Signal'] = 0 # Initialize signals as 0

data.loc[data['SMA\_40'] > data['SMA\_100'], 'Signal'] = 1 # Buy signal

data.loc[data['SMA\_40'] < data['SMA\_100'], 'Signal'] = -1 # Sell signal

# Calculate 52-week high and low

data['52\_Week\_High'] = data['Close'].rolling(window=252).max() # Highest close in 1 year

data['52\_Week\_Low'] = data['Close'].rolling(window=252).min() # Lowest close in 1 year

return data

def visualize\_data(data):

"""Perform Exploratory Data Analysis (EDA) with visualizations."""

# Time Series Plot of closing prices

plt.figure(figsize=(12, 6))

plt.plot(data.index, data['Close'], label='Stock Price')

plt.title('Stock Price Over Time')

plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.show()

# Histogram of stock prices

plt.figure(figsize=(8, 6))

plt.hist(data['Close'], bins=50, color='blue', alpha=0.7)

plt.title('Distribution of Stock Prices')

plt.xlabel('Price')

plt.ylabel('Frequency')

plt.show()

# Bar plot of binary price movement

plt.figure(figsize=(6, 4))

data['Price\_Up'].value\_counts().plot(kind='bar', color=['red', 'green'])

plt.title('Price Movement Distribution')

plt.xlabel('Price Up (1) or Down (0)')

plt.ylabel('Count')

plt.show()

# Bar plot of buy/sell/hold signals

plt.figure(figsize=(6, 4))

data['Signal'].value\_counts().plot(kind='bar', color=['red', 'blue', 'green'])

plt.title('Buy/Sell/Hold Signal Distribution')

plt.xlabel('Signal')

plt.ylabel('Count')

plt.show()

# Plot 52-week high and low vs Close price

plt.figure(figsize=(12, 6))

plt.plot(data.index, data['Close'], label='Close Price', alpha=0.7)

plt.plot(data.index, data['52\_Week\_High'], label='52-Week High', linestyle='--')

plt.plot(data.index, data['52\_Week\_Low'], label='52-Week Low', linestyle='--')

plt.title('52-Week High and Low with Close Price')

plt.xlabel('Date')

plt.ylabel('Price')

plt.legend()

plt.show()

# Plot normalized volume

plt.figure(figsize=(12, 6))

plt.plot(data.index, data['Scaled\_Volume'], label='Normalized Volume', color='purple')

plt.title('Normalized Volume Over Time')

plt.xlabel('Date')

plt.ylabel('Normalized Volume')

plt.legend()

plt.show()

# Main script

if \_\_name\_\_ == "\_\_main\_\_":

ticker = "TATAMOTORS.BO" # Specify the stock ticker

start\_date = "2020-01-01" # Start date for data

end\_date = "2025-01-01" # End date for data

try:

stock\_data = load\_stock\_data(ticker, start\_date, end\_date) # Fetch data

stock\_data = preprocess\_data(stock\_data) # Preprocess it

# Check essential columns exist

if 'Close' not in stock\_data.columns or 'Price\_Up' not in stock\_data.columns:

raise ValueError("Critical columns are missing after processing.")

visualize\_data(stock\_data) # Run visual analysis

except Exception as e:

print(f"An error occurred: {e}") # Error handling