**Part 1: EDA, Preprocessing, Feature Engineering for Stock Return Prediction**

## 📅 1. Install and Import Required Libraries

Before any data analysis or modeling begins, we must import the right set of tools.

!pip install yfinance --quiet  
  
import yfinance as yf  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from datetime import datetime

### 🔍 Why these libraries?

| Library | Use Case |
| --- | --- |
| yfinance | To download historical stock data (Open, High, Low, Close, Volume) |
| pandas | DataFrame operations, cleaning, transformation |
| numpy | Numerical operations like % change, array handling |
| matplotlib | Plotting graphs (line, scatter, bar) |
| seaborn | Advanced visualization for trends & correlations |
| datetime | Handle today’s date automatically for end\_date of stock download |

## 🔎 2. Download Stock Price Data (TSLA)

ticker = "TSLA"  
start\_date = "2018-01-01"  
end\_date = datetime.today().strftime('%Y-%m-%d')  
  
# Download data  
df = yf.download(ticker, start=start\_date, end=end\_date)  
df.head()

### 🤔 What is OHLCV data?

| Column | Description |
| --- | --- |
| Open | Opening price of the stock for that day |
| High | Highest price of the day |
| Low | Lowest price of the day |
| Close | Closing price of the day |
| Volume | Number of shares traded during the day |

### 🚧 Warning about auto\_adjust=True

* New versions of yfinance default to adjusting prices for **splits/dividends**.
* This is **good for ML** because raw stock splits/dividends can look like artificial crashes or spikes.

Example: Tesla splits 5-for-1, price drops 80% overnight — but it’s not a crash. Adjusted prices make this smooth.

## 🌎 3. Visualize Stock Price & Trends

plt.figure(figsize=(14, 6))  
plt.plot(df['Close'], label='TSLA Closing Price', color='purple')  
plt.title('Tesla Stock Price (Closing)', fontsize=16)  
plt.xlabel('Date')  
plt.ylabel('Price ($)')  
plt.legend()  
plt.grid(True)  
plt.show()

This shows the full price trend since 2018.

## 📈 4. Add Moving Averages (MA)

df['MA\_20'] = df['Close'].rolling(window=20).mean()  
df['MA\_50'] = df['Close'].rolling(window=50).mean()

### 📉 Why use MA\_20 and MA\_50?

| MA | Meaning | Purpose |
| --- | --- | --- |
| 20-day MA | Short-term trend (~1 month) | Detect short-term momentum |
| 50-day MA | Medium-term trend (~2.5 months) | Smoother, useful for crossovers |

plt.figure(figsize=(14, 6))  
plt.plot(df['Close'], label='Close Price', alpha=0.5)  
plt.plot(df['MA\_20'], label='20-Day MA', linestyle='--')  
plt.plot(df['MA\_50'], label='50-Day MA', linestyle='--')  
plt.title('Moving Averages on TSLA Stock', fontsize=16)  
plt.xlabel('Date')  
plt.ylabel('Price ($)')  
plt.legend()  
plt.grid(True)  
plt.show()

## 🔢 5. Create Features for ML Model

df['Return'] = df['Close'].pct\_change()  
df['Lag\_1'] = df['Return'].shift(1)  
df['Lag\_2'] = df['Return'].shift(2)  
df['Lag\_3'] = df['Return'].shift(3)  
  
# Drop rows with NaNs from pct\_change and shift  
df.dropna(inplace=True)  
  
features = ['Lag\_1', 'Lag\_2', 'Lag\_3']  
target = 'Return'  
  
df\_ml = df[features + [target]].copy()

### 🧮 Why does pct\_change() create NaN?

It calculates % change from the previous day:

| Day | Close | Return |
| --- | --- | --- |
| 1 | 100 | NaN |
| 2 | 105 | 0.05 |
| 3 | 104 | -0.0095 |
| 4 | 105.04 | 0.01 |

* Day 1 has no prior value → NaN

### 🛠️ Why does shift() create NaNs?

df['Lag\_1'] = df['Return'].shift(1)

This moves the return column **down by 1 row**, so top rows will be NaN:

| Day | Return | Lag\_1 |
| --- | --- | --- |
| 1 | NaN | NaN |
| 2 | 0.05 | NaN |
| 3 | -0.0095 | 0.05 |
| 4 | 0.01 | -0.0095 |

### 🔎 What does y = Return mean?

You’re trying to **predict today’s return** using **Lag\_1, Lag\_2, Lag\_3**:

| Lag\_3 | Lag\_2 | Lag\_1 | Return (y) |
| --- | --- | --- | --- |
| 0.05 | -0.0095 | 0.01 | 0.007 |

So:

* X = inputs = [0.05, -0.0095, 0.01] (past 3 days)
* y = output = 0.007 (today’s return)

This is a **regression problem** (predicting a continuous value).

## 🔺 Visual Timeline Example

| Date | Close | Return | Lag\_1 | Lag\_2 | Lag\_3 |
| --- | --- | --- | --- | --- | --- |
| Mar 10 | 100 | NaN |  |  |  |
| Mar 11 | 105 | 0.050 | NaN |  |  |
| Mar 12 | 104 | -0.0095 | 0.050 | NaN |  |
| Mar 13 | 105.04 | 0.01 | -0.0095 | 0.050 | NaN |
| Mar 14 | ??? | ??? | 0.01 | -0.0095 | 0.050 |

Model input (X): [0.050, -0.0095, 0.01]  
Model output (y): Return on Mar 14

## 📃 6. Save Preprocessed CSV to Drive (Optional)

from google.colab import drive  
drive.mount('/content/drive')  
  
output\_path = '/content/drive/MyDrive/Colab Notebooks/Finance Projects/stock-price-prediction-ml/data/tsla\_preprocessed.csv'  
df\_ml.to\_csv(output\_path, index=False)  
print("Saved to:", output\_path)

This completes Part 1: Download, Visualize, Feature Engineer, and Prepare the dataset for ML-based stock return prediction.