Step 1: Feature Engineering (Lag Features)

Goal: Transform the time-series data into a format your supervised learning models (XGBoost, KNN, LightGBM) can understand.

- **Create Lag Features:** You must create "past price" features. For example, to predict today's price (t), use the prices from t-1, t-2, up to t-7.
- **Define Target (y):** Your target variable is the "Close Price" (or Next_Day_Close if you shifted the data).
- **Define Features (X):** Your input features include the Lag Features, plus "Open," "High," "Low," and "Volume".

Step 2: Time-Series Cross-Validation (Critical)

Goal: Implement the validation logic. **Do not use standard K-Fold**, as it shuffles data and destroys the time order.

- **Method:** Use TimeSeriesSplit from scikit-learn. This creates "rolling windows" where the training set grows, and the test set is always "in the future" relative to the training set.
- **Split Logic:**
    - *Fold 1:* Train on Jan-Mar, Test on Apr.
    - *Fold 2:* Train on Jan-Apr, Test on May.
    - *Fold 3:* Train on Jan-May, Test on June.

Step 3: Implement the 5 Models

You must implement the following five algorithms.

A. The Regressors (KNN, XGBoost, LightGBM)

These models treat the problem as standard regression using the Lag Features created in Step 2.

- **Input:** 2D Arrays (Rows, Features).
- **Implementation:** Use XGBRegressor, LGBMRegressor, and KNeighborsRegressor.
- **Tip:** Tree-based models (XGBoost, LightGBM) are excellent at handling non-linear relationships.

B. The Statistical Model (ARIMA)

- **Library:** Use statsmodels.tsa.arima.model.ARIMA.
- **Requirement:** ARIMA requires stationarity. Check if the data needs differencing (Role 1 might have handled this, but verify).
- **Looping:** You may need to re-fit the ARIMA model for each step in your cross-validation loop.

C. The Deep Learning Model (LSTM)

- **Library:** Use Keras (TensorFlow) or PyTorch.
- **Input Shape:** LSTM requires a 3D input shape: (Samples, Time Steps, Features). You must reshape your DataFrame before feeding it into the model.
- **Structure:** A standard architecture includes an LSTM layer followed by a Dense (output) layer.

Step 4: Train, Predict, and Save

Goal: Generate the deliverable file.

- **Execution:** Loop through your Time-Series Split. For every fold, train all 5 models on the *Train* portion and predict on the *Test* portion.
- **Aggregation:** Collect all predictions. You generally want to save the "out-of-sample" predictions (the test fold results) to compare against actual values.
- **Deliverable:** Save a file named predictions.csv. This file should likely contain columns for Date, Actual_Price, Pred_ARIMA, Pred_KNN, Pred_LSTM, Pred_XGB, and Pred_LGBM.