Time series forecasting model using ARIMA or LSTM.

1. Understanding the Problem and Data Preparation

- **Define the Objective**: Clearly define what you want to forecast (e.g., monthly sales, daily temperature).
- **Collect Data**: Time series data should have a timestamp and a target variable (e.g., date and sales).
- **Explore the Data**:
  1. **Visualize**: Plot the data to spot any trends, seasonality, or irregularities.
  2. **Check for Stationarity**: Stationary data has a constant mean and variance over time. Use the Augmented Dickey-Fuller (ADF) test to check this.
     - **Tip: If data isn't stationary, try *differencing* (subtracting previous observations) to make it stationary.**
  3. **Missing Values**: Handle missing values through forward-fill, backward-fill, or interpolation.
  4. **Outliers**: Identify and handle outliers using Z-score or IQR methods, as they can skew results

2. Choosing ARIMA or LSTM Model

- **When to Choose ARIMA:**
  - If the data has a linear trend and is univariate.
  - Best for short-term forecasting and stationary data.
- **When to Choose LSTM:**
  - If the data has a non-linear trend or is multivariate.
  - Works well for long-term dependencies.

3. Building ARIMA Model

- **Step 1**: **Parameter Selection (p, d, q)**
  - **p**: Order of the Auto-Regressive (AR) term.

  - **d**: Differencing needed to make the data stationary.
  - **q**: Order of the Moving Average (MA) term.
  - **Tip: Use the ACF (Auto-Correlation Function) and PACF (Partial Auto-Correlation Function) plots to identify p and q.**
- **Step 2**: **Train the Model**
  - Use Python's `statsmodels` library with `ARIMA` class.
  - Fit the model on training data.
  - **Tip: If results are unsatisfactory, try adjusting p, d, q values. Use auto-ARIMA (`pmdarima` library) to automate this selection.**
- **Step 3**: **Model Diagnostics**
  - Check residuals to ensure they're normally distributed and uncorrelated.
  - Plot residuals and perform ADF tests.
  - **Tip: If residuals aren't normally distributed, consider transforming the data with a log or square root transformation.**

4. Building LSTM Model

- **Step 1**: **Data Transformation**
  - **Normalization**: Scale data between 0 and 1 using MinMaxScaler.
  - **Windowing**: Convert time series into supervised format. Use a sliding window to create sequences for LSTM input.
    - **Tip: Start with a window size of 3-5 for smaller datasets and 10-20 for larger ones**.
- **Step 2**: **Define Model Architecture**
  - Use `Sequential` model from `tensorflow.keras`.
  - Define an LSTM layer with units (number of neurons) and optionally add Dropout to avoid overfitting.
  - Add a Dense layer for output.
    - **Tip: Start with one LSTM layer with 50-100 units; more layers might improve performance but increase training time.**
- **Step 3**: **Train the Model**
  - Split data into training and testing sets.
  - Compile the model with a loss function (`mean_squared_error`) and an optimizer (`adam`).
  - Fit the model with an appropriate number of epochs (10-50 to start with) and a batch size.
    - **Tip: Start with a lower number of epochs (e.g., 10) and increase gradually. Too many epochs can lead to overfitting.**

5. Evaluating the Model

For Both Models (ARIMA & LSTM):

- **Metrics**: Use RMSE (Root Mean Square Error) or MAE (Mean Absolute Error) to evaluate.
- **Train-Test Split**: If you're forecasting a long period, it's best to have a large test set to assess performance on unseen data.
- **Tip: To avoid information leakage, avoid shuffling data in time series models.**

  Cross-Validation: Use time series cross-validation (`TimeSeriesSplit` in `sklearn`) for a more reliable error estimation.

- **Tip: Standard K-Fold cross-validation doesn't work for time series, as it would disrupt the temporal order.**

6. Fine-Tuning and Model Selection

For ARIMA:

- Tune the p, d, q values iteratively and compare results based on RMSE or MAE.
- **Tip: Try seasonal ARIMA (SARIMA) if there's seasonality in the data.**

For LSTM:

- Experiment with the number of layers, neurons, and dropout rate.
- **Tip: Adding a few dense layers or adjusting the learning rate can improve performance but monitor for overfitting.**

7. Forecasting and Visualization

Generate Forecasts:

- Use `predict` for ARIMA and LSTM to get future predictions.
- **Tip: In LSTM, remember to inverse the scaling transformation to return data to its original form.**

Visualize Results:

- Plot predictions against actuals to assess fit.

- Plot error metrics like RMSE over time.
- **Tip: Use shaded areas in plots to indicate forecast uncertainty.**

8. Deployment Tips

Save the Model:

- For ARIMA: Save model parameters (p, d, q).
- For LSTM: Use `model.save()` in Keras.

Monitoring:

- Set up a system to monitor model performance over time. If the model drifts, consider retraining.
- **Tip: Periodically retrain with the latest data to keep forecasts accurate.**