**💡 Product Ideas & Core Features**

Your primary goal is to create a tool that provides **actionable insights**, not just raw numbers. The best products empower business leaders to make decisions.

1. **Interactive Forecasting Dashboard:** This is the core of your product. It should allow users to visualize historical sales and future forecasts. **Key features** include:
   * **Hierarchical Drill-Down:** Users should be able to see forecasts at the national level, then click down to **REGION**, **AREA**, and finally **TERRITORY**.
   * **Adjustable Forecast Horizon:** Let users select whether they want to see a forecast for the next 3, 6, or 12 months.
   * **Confidence Intervals:** Never show just a single prediction. Always show a probable range (e.g., 95% confidence interval). This helps managers understand the level of uncertainty.
2. **"What-If" Scenario Analysis:** This feature turns a simple forecast into a strategic tool. Allow users to model scenarios like:
   * "What if we run a 10% discount campaign in the Dhaka region next month?"
   * "What impact will a new public holiday have on sales?"
3. **Anomaly Detection:** Automatically flag regions or territories where sales are significantly above or below the forecasted amount. This helps managers quickly identify over-performing areas to learn from or under-performing areas that need attention.

**⚙️ Best Practices: The Technical Roadmap**

Here's a step-by-step approach to building a robust forecasting system.

**1. Data Preparation and Feature Engineering**

This is the most critical step. Your model is only as good as your data.

* **Data Cleaning:** Handle missing values (e.g., using interpolation or forward-fill) and identify any outliers that might be data entry errors.
* **Feature Creation (Crucial for Accuracy):**
  + **Time-Based Features:** Create features from the date, such as week\_of\_year, day\_of\_week, quarter, and is\_month\_start/end.
  + **Lag Features:** Include sales from the previous period (e.g., last month's sales, sales from the same month last year). This captures momentum and seasonality.
  + **Rolling Statistics:** Calculate rolling averages or standard deviations (e.g., 3-month rolling average sales).
* **Incorporate External Data:** This is a key differentiator. Include data that influences sales:
  + **Holidays:** Especially important in Bangladesh (e.g., Eid-ul-Fitr, Eid-ul-Azha, Pohela Boishakh).
  + **Economic Indicators:** Inflation rates, GDP growth.
  + **Marketing Data:** Information on promotional campaigns and marketing spend per region.

**2. Modeling Strategy**

Don't jump straight to the most complex model. Start simple and build up.

* **Start with a Baseline:** Use simpler models like **SARIMA** (Seasonal Auto-Regressive Integrated Moving Average) or **Exponential Smoothing**. This gives you a benchmark to beat.
* **Embrace the Hierarchy:** Your data is hierarchical. Your model should be too. **Hierarchical Time Series Forecasting** ensures that the sum of the territory forecasts logically adds up to the area forecast, which adds up to the region forecast. Libraries like scikit-hts can help with this. This prevents the common problem where individual forecasts don't make sense when combined.
* **Advanced Models:** For higher accuracy, use tree-based models which are excellent with tabular data:
  + **LightGBM** or **XGBoost:** These are often the best-performing models for this type of problem. They are fast, scalable, and can capture complex relationships between your features and sales.
* **Model Validation:** Always hold back the last 6-12 months of your data as a test set. **Never train your model on this data.** You will use it only once at the end to evaluate the real-world performance of your final model using metrics like **MAPE** (Mean Absolute Percentage Error) and **RMSE** (Root Mean Squared Error).

**3. MLOps and Deployment**

A model is useless if it's not in production and being monitored.

* **Automated Retraining Pipeline:** Sales patterns change. You need to build an automated pipeline (using a tool like **Apache Airflow**) that periodically retrains your model on new data (e.g., at the end of every month).
* **Model Serving:** Deploy your trained model as a REST API using a framework like **FastAPI**. Your dashboard will call this API to get forecasts on demand.
* **Monitoring for Drift:** The world changes, and your model's accuracy can degrade over time (this is called "model drift"). Implement a monitoring system to track your model's performance on new data and trigger alerts if its accuracy drops below a certain threshold, indicating a need for retraining or redesign.