CV Prep

Sales Forecasting:

At CIPLA, I led a sales forecasting project to predict 3- and 6-month demand for pharmaceutical products, addressing export planning challenges across pre- and post-COVID periods. I developed time-series models using FbProphet and XGBoost, capturing complex seasonality and pandemic-induced shifts. The final solution achieved 82% forecast accuracy (MAPE), enabling more reliable export logistics and data-driven inventory decisions.

What metrics did you use to evaluate your models and why?

Ans.

For this sales forecasting project, I primarily used **Mean Absolute Percentage Error (MAPE)** as the evaluation metric. Since we were forecasting **monthly sales volumes**, and the scale varied significantly across different products, MAPE was a good fit because it expresses the error as a **percentage**, making it easier to interpret and compare across product categories.

In addition to MAPE, I also tracked:

- **RMSE (Root Mean Squared Error)** to penalize larger errors more heavily,
- And **MAE (Mean Absolute Error)** for a more robust view that isn't overly sensitive to outliers.

Ultimately, MAPE was the most business-aligned metric, as the stakeholders at CIPLA were interested in percentage-level forecast accuracy to guide export planning.

If They Ask “Why not use accuracy or R^2?”

You can say:

> "Traditional classification metrics like accuracy don't apply here since it's a regression/forecasting task. R^2 can be useful but is harder to interpret from a business perspective in terms of actual forecast deviation, especially with high variance products."

Why did you choose MAPE over RMSE or MAE to evaluate your forecasting model?"

I chose MAPE (Mean Absolute Percentage Error) as the primary evaluation metric because it provides a percentage-based error, which makes it easy for business stakeholders to interpret and compare across different products — especially in our case, where product sales volumes varied significantly.

For example, predicting 1,000 units vs. 10,000 units can lead to very different absolute errors, but MAPE normalizes this and gives a consistent view of model accuracy as a percentage.

I also calculated RMSE and MAE for internal validation:

- RMSE was useful to penalize larger errors more heavily,
- MAE helped with overall stability and robustness of the model.

But since the business team (CIPLA) needed an intuitive and actionable metric for sales and export planning, MAPE was the most suitable and communicable choice.

That said, I'm aware of MAPE's limitations — it can become undefined or unstable when actual values are close to zero. So I made sure to handle zero or near-zero values with conditional logic during evaluation.

Did you perform cross-validation? How?

Since this was a time-series problem, I couldn't use standard k-fold cross-validation because that would randomly mix past and future data, which causes data leakage.

Instead, I used a time-based or rolling-window cross-validation approach. I trained the model on a past window of data and validated it on the next immediate period — like training on months 1–12 and validating on month 13. Then I rolled the window forward, for example, training on months 2–13 and validating on month 14.

This mimics how the model would perform in a real-world scenario, where we always forecast future values using only past data.

How did you decide between FbProphet and XGBoost?

I chose both **FbProphet** and **XGBoost** based on their complementary strengths, and I evaluated them against each other to pick the best performer for different product categories.

FbProphet was my first choice because it's designed specifically for **time series with strong seasonality and trends**, which was ideal for pharmaceutical products that show recurring patterns — like cold & flu medicine peaking in winters. It also allowed us to easily include **holiday effects** and **COVID-period change points**.

On the other hand, I used **XGBoost** when I wanted to go beyond univariate forecasting and incorporate **external or engineered features**, such as:

- **Lag features** (previous month's sales),
- **Rolling averages**,
- **COVID lockdown flags**,
- **Export region**, and even **product types**.

XGBoost gave me more control over **feature engineering and interactions**, while Prophet was quick and interpretable for seasonal trends.

I evaluated both using **MAPE and RMSE**, and in many cases, **XGBoost outperformed Prophet** due to the extra information it could leverage — especially post-COVID, where external factors played a bigger role.

Add on->

In fact, for some products, I even tried **combining both** — using Prophet to model the baseline trend and seasonality, and feeding its predictions as a feature into XGBoost for residual learning. That hybrid approach worked especially well during uncertain post-COVID fluctuations.

Why didn't you use ARIMA or SARIMA instead of FbProphet or XGBoost?

I considered ARIMA and SARIMA early on, but they had a few limitations for this specific project. Our data involved:

- **Multiple products** with different seasonal behaviors,
- A clear **pre- and post-COVID structural change**,
- And the need to forecast for **3 to 6 months ahead**.

ARIMA assumes the time series is **stationary**, which required heavy differencing and transformation. It also doesn't handle **multiple seasonalities** or sudden trend shifts (like COVID) very well without manual tuning.

SARIMA adds seasonal components, but still requires manual parameter tuning (p, d, q, P, D, Q), and isn't very flexible when external variables or business events (**like export region or holidays**) need to be included.

In contrast:

- **FbProphet** handled seasonality, holidays, and change points with ease.
- **XGBoost** allowed me to add exogenous(external) features like **lockdown flags, lag features, and region-wise variations** — which ARIMA/SARIMA can't do natively.

So while ARIMA/SARIMA are strong statistical models, they weren't the best fit for the complexity and dynamic nature of our data. Prophet and XGBoost gave better performance and interpretability for the business case.

1. What features did you use for XGBoost?

✅ **Best Answer:**

For XGBoost, I engineered a mix of **time-based, lag-based, and external features** to help the model learn sales patterns and seasonal dependencies. Specifically, I used:

- **Lag features**: previous 1, 2, and 3-month sales to help the model capture autocorrelation.
- **Rolling statistics**: 3-month and 6-month moving averages and standard deviations.
- **Date features**: month, quarter, and year to represent seasonality.

- **Holiday flags**: to account for spikes or drops during national festivals or events.
- **COVID indicators**: binary flags for lockdown periods and a "post-COVID" flag for behavioral shift.

These features helped the model understand both **temporal trends** and **external shocks** like COVID disruptions.

2. How did you handle missing data or anomalies (like COVID impact)?

✅ **Best Answer:**

For missing data, I used a combination of:

- **Forward-filling** for missing sales in short gaps,

- And **median imputation** grouped by product and month for longer or early-period gaps.

To handle anomalies — especially due to **COVID disruptions** — I took a few steps:

- **Created binary flags** for lockdown months and post-COVID periods,
- **Treated extreme spikes/drops** using IQR-based outlier capping,
- And in some cases, I **excluded data points** from the training set if they were too distorted and could mislead the model.

I also used visual analysis (e.g., time series plots) to manually verify if an anomaly was legitimate or a data error.

3. Did you do any hyperparameter tuning?

✅ **Best Answer:**

Yes, I performed **extensive hyperparameter tuning** using a **randomized grid search** combined with **TimeSeriesSplit cross-validation**.

The key parameters I tuned were:

- `n_estimators` (number of trees),
- `max_depth` (tree complexity),
- `learning_rate` (how fast the model learns),
- `subsample` and `colsample_bytree` (to reduce overfitting),
- And `gamma` (to control split sensitivity).

Since it was a time-series problem, I made sure to use **time-aware validation** (rolling or expanding window CV), not standard K-fold, to avoid data leakage from future values.

This helped improve generalization and stability of the model on unseen forecast horizons.

8. How did you handle seasonality in your models?

✅ **Best Answer:**

I handled seasonality differently for each model.

- In **FbProphet**, seasonality is built-in — it models yearly, weekly, and even daily seasonality automatically.
- For **XGBoost**, I manually encoded seasonality by creating features like:
    - **Month**, **quarter**, and **day of the year**,
    - Binary flags for **festivals or public holidays**,
    - And **lag-based patterns** to capture repeated behavior over time.

This helped XGBoost learn recurring trends even though it doesn't natively model seasonality like Prophet does.

---

9. Did you do differencing or smoothing?

✅ **Best Answer:**

I didn't need to apply differencing or smoothing directly because I wasn't using ARIMA-based models, which require stationary data.
However, I did test for **stationarity** to better understand the structure of the data.
In the case of **XGBoost**, since it's a tree-based model, it can handle non-stationary data quite well — especially when we provide it with **lag features** and **trend-based inputs**.

---

10. Was the data stationary? Did you check ACF/PACF?

✅ **Best Answer:**

I did check for stationarity using the **Augmented Dickey-Fuller (ADF) test**, and found that the data was **non-stationary**, which was expected due to strong seasonality and post-COVID trends.
I also explored **ACF and PACF plots** to understand the presence of autocorrelation and the lag structure, mainly to guide feature engineering for XGBoost.
Since XGBoost can learn from lagged patterns even in non-stationary data, I focused on crafting meaningful lag features rather than forcing the series to be stationary.

Let me know if you want visuals like ACF/PACF examples or ADF test explanation to bring into your interview or portfolio!

---

11. Why was 3- and 6-month prediction important for CIPLA?

✅ **Answer:**
CIPLA needed 3- and 6-month forecasts to plan exports in advance. This helped with **production, inventory, and international shipping**, which all have long lead times.

---

12. How did your model help improve decision-making?

✅ **Answer:**
The model gave accurate sales forecasts, which helped the business plan **stock levels**, **avoid overproduction**, and make **better export and procurement decisions**.

---

13. Did the business team use your outputs? How were results delivered?

✅ **Answer:**
Yes, they used the outputs. I shared the results through **interactive dashboards and Excel reports**, so the business and supply chain teams could make informed decisions easily.

---

14. What was the size of the dataset? What was the granularity?

✅ **Answer:**
The dataset had **monthly sales data** for different medicines over **3–4 years**. Each row had a **medicine name**, **sales value**, and **date**, so the granularity was **monthly per product**.

---

15. Did you face any data quality issues? How did you resolve them?

✅ **Answer:**
Yes, some months had **missing sales values** and **inconsistent product names**. I handled this by:

- Using **forward fill or median imputation** for missing data,
- And cleaning the product names using **standard mapping and normalization**.

---

16. How did you deploy or operationalize your models?

✅ **Answer:**
I shared results via **automated dashboards** and **Excel reports**. The model was set up to **retrain monthly**, and forecasts were updated and sent to the business team for planning.

---

17. How would you improve the model further?

✅ **Answer:**
I'd improve it by adding **external factors** like weather, flu season indicators, or **market demand signals**.
I could also try **deep learning models** like **LSTM** for better long-term sequence understanding, especially for more complex patterns.

---

18. If product demand changed suddenly, how would your model adapt?

✅ **Answer:**
I'd add a **model retraining pipeline** to update predictions regularly.
I'd also track **model drift** and use **change-point detection** to catch sudden shifts like new regulations or health events.

---

19. Did you consider ensemble or hybrid models?

✅ **Answer:**
Yes, I tested combining **Prophet's trend and seasonality** with **XGBoost's feature-based learning**.
A **hybrid or weighted average** approach gave better results for some product categories, especially post-COVID where patterns were less stable.
