📘 XGBoost-Based 8-Week Demand Forecasting Documentation

File Name: predict_8weeks_XGB.py
Purpose: Predict next 8 weeks of medicine demand using a pre-trained XGBoost regression model.
Primary Use Case: Backend inference integration for hospital or pharmaceutical inventory demand forecasting.

⚙️ 1️⃣ Overview

This script is designed to forecast medicine demand for the next **8 weeks** (~2 months) using a **trained XGBoost model**.
It takes the historical weekly demand data as input, generates lag and time features, and performs iterative predictions week by week.

The script outputs both **visual plots** and a **saved Excel file** containing the forecasted demand quantities.

📁 2️⃣ File Path Dependencies

Type	Description	Example Path
Input Dataset	Historical weekly demand data	../data/demand_prediction_weekly.xlsx
Saved Models	Trained XGBoost models for each medicine	../saved models/xgboost_<**medicine_name**>.json
Output File	8-week forecast result (Excel)	../outputs/forecast_<**medicine_name**>_8weeks_XGB.xlsx

---

📥 3️⃣ Input Details

User Input

The script requests the medicine name from the user:

medicine_name = input("Enter the medicine name: ").strip()

This value is used to filter the dataset and select the corresponding model.

Expected Columns in Dataset

The input Excel file (../data/demand_prediction_weekly.xlsx) must include the following columns:

Column	Description
Product_Name	Name of the medicine/product
Week	Week formatted as YYYY-W## (e.g., 2025-W43)
Year	Year of the record
Week_Number	Week number (1–52)
Total_Quantity	Actual weekly demand quantity

🧠 4️⃣ Inference Workflow

Step 1: Load and Filter Data

- Loads the Excel dataset from ../data/demand_prediction_weekly.xlsx.
- Filters the data for the selected medicine.
- Converts Week strings like 2025-W43 into actual datetime objects.

Step 2: Feature Engineering

The script builds several features to help the model understand patterns and seasonality:

- **Temporal features:**
  - Month (derived from week number)
  - Quarter
  - Is_Year_Start
  - Is_Year_End
- **Cyclic encodings (seasonality):**
  - Sin_Week
  - Cos_Week

- **Lag-based features:**
  - Past 12 weeks of demand (lag_1 to lag_12)
- **Rolling statistics:**
  - rolling_mean_3, rolling_mean_5, rolling_mean_6
  - rolling_std_4, rolling_std_6
  - rolling_mean_8

These features simulate historical patterns during training.

Step 3: Load Trained Model

- Loads the medicine-specific model file from:
- ../saved models/xgboost_<**medicine_name**>.json
- Each medicine has its own trained XGBoost model.

Example:

model_path = os.path.join("../saved models", f"xgboost_{**medicine_name**}.json")

model.load_model(model_path)

Step 4: StandardScaler Reconstruction

The model expects normalized input data.
A new StandardScaler is recreated using the filtered medicine dataset:

scaler.fit(X_train)

This ensures feature scaling matches what was used during training.

Step 5: Iterative Forecasting Logic

The model predicts demand **one week at a time** for the next 8 weeks.
Each prediction is then fed back as an input for the next week's prediction (auto-regressive loop).

Loop process:

1. Compute next week's temporal features (Week, Month, Quarter, etc.)
2. Use previous 12 demand values (actual or predicted) for lag features.

3. Calculate rolling averages and standard deviations.
4. Build a single feature row, apply scaling, and predict using the model.
5. Append the predicted value and move to the next week.

This process repeats until 8 future weeks are predicted.

Step 6: Create Forecast Output

After predictions, the script builds a result DataFrame:

Week	**Predicted_Quantity**
2025-W43	418
2025-W44	405
2025-W45	392
...	...

This DataFrame is displayed in the console and used for plotting.

5️⃣ Visualization

Two plots are generated:

1. **Actual vs Forecast (Last 12 Weeks + Next 8 Weeks)**
   - Blue line: Historical actual demand
   - Red dashed line: Model forecast
2. **Simple Forecast Overview**
   - Displays weekly predicted quantities over future dates.

These plots help visually verify model accuracy and future demand trends.

6️⃣ Output Details

After forecasting, results are saved as an Excel file for integration:

output_file = f"../outputs/forecast_{medicine_name.replace(' ', '_')}_8weeks_XGB.xlsx"

```
future_df.to_excel(output_file, index=False)
```

Example Saved File

../outputs/forecast_PARACETAMOL_500MG_8weeks_XGB.xlsx

Output Columns

Column	**Description**
Week	Forecasted week (e.g., 2025-W44)
Predicted_Quantity	Forecasted weekly demand (integer value)

7️⃣ Integration Notes for Backend Developers

Parameter	**Description**
Input (User/Backend)	**medicine_name** (string)
Model Path	../saved models/xgboost_<**medicine_name**>.json
Output Path	../outputs/forecast_<**medicine_name**>_8weeks_XGB.xlsx
Dependencies	pandas, numpy, matplotlib, scikit-learn, xgboost
Functionality	Predicts future 8 weeks of demand using lag-based inference
Return Format	DataFrame containing Week and Predicted_Quantity
Usage in Backend	Replace the input() call with a backend variable for automation

---

8️⃣ Summary Workflow

Step	**Process**
1️⃣	User enters medicine name
2️⃣	Data loaded and filtered
3️⃣	Model loaded from ../saved models/
4️⃣	Lag and time-based features generated
5️⃣	Predicts 8 future weeks iteratively

Step	Process
6️⃣	Plots and displays forecast results
7️⃣	Saves forecast in ../outputs/ folder

✅ Example Console Output

```
Enter the medicine name: Paracetamol 500mg


✅ Forecast complete for 'Paracetamol 500mg'!

📊 Next 8 Weeks Prediction:


   Week  Predicted_Quantity
2025-W43           412
2025-W44           398
2025-W45           384
2025-W46           376
2025-W47           365
2025-W48           359
2025-W49           348
2025-W50           341


💾 Forecast saved to '../outputs/forecast_Paracetamol_500mg_8weeks_XGB.xlsx'
```

🧾 End of Documentation

Developer Note:
This script is fully reusable for inference.
For backend automation, simply pass the medicine name programmatically (instead of using input()), and ensure the trained model file and dataset follow the same folder structure and naming convention.