
🧩 1. Forecasting Workflow

The **forecasting feature** in the GoGrow Market Insights module predicts the future **demand and supply** of agricultural crops for upcoming seasons (e.g., Maha 2026).
It combines:

- **User-selected historical data** (2022–2025, etc.) from the dataset
- A **backend machine learning regression model** to learn trends
- **Interactive visualizations** (pie charts) for demand, supply, and demand–supply gap

This allows farmers, traders, and policy planners to anticipate potential shortages or surpluses in specific crops for a given region and season.

⚙️ 2. Frontend Interaction Process

a. User Selection

Users interact with a clean, mobile-friendly HTML + Chart.js dashboard.
They can:

- Toggle **regions** (Ratnapura, Kegalle)
- Choose the **season** (Maha / Yala)
- Select **training years** (e.g., 2022 → 2025) to define the historical data window

The system automatically assumes the **forecast target year** as:

```
max(selected years) + 1
```
For example, if 2022–2025 are selected → forecast = 2026.

b. Request Generation

When the user clicks **"Load Forecast Pies"**, the frontend JavaScript collects the filter inputs and sends a JSON request:

```
{
  "regions": ["Ratnapura", "Kegalle"],
  "season": "Maha",
  "years": [2022, 2023, 2024, 2025]
}
```

This request is sent to the backend route:

```
POST /api/forecast
```

c. Visualization

Once the server responds, three **Chart.js pie charts** are rendered:

1. **Demand Share Pie** – proportion of forecasted demand per crop
2. **Supply Share Pie** – proportion of forecasted supply per crop
3. **Positive Gap Pie** – crops with excess demand over supply

This provides an immediate visual comparison of which crops are likely to experience shortages or oversupply in the forecasted season.

🧠 3. Backend Forecasting Logic

a. Route Definition

The endpoint `/api/forecast` handles incoming requests:

```
app.post('/api/forecast', (req, res) => {
  const regions = req.body.regions;
  const season  = req.body.season;
  const years   = req.body.years.map(Number);
  const f = forecastByCrop(regions, season, years);
  ...
});
```

It passes the filtered dataset and year range to the function `forecastByCrop()`.

---

b. Data Preparation

Inside `forecastByCrop()`:

1. The system filters all dataset rows matching the selected **region, season, and years**.
2. Data are grouped **by crop and by year**, summing total supply and demand for each crop across the selected period.

This produces a compact structure such as:

```
{
  "Paddy": { "2022": {"supply":1200,"demand":1300}, "2023": {...}, ... },
  "Tea":   { "2022": {"supply":800, "demand":750},  ... }
}
```

c. Machine Learning Forecast

Each crop has its own small regression model trained on the yearly totals.

A **Multivariate Linear Regression (MLR)** model from `ml-regression-multivariate-linear` is used to fit a simple linear trend:

```
const MLR = require('ml-regression-multivariate-linear');
const xs = [[2022],[2023],[2024],[2025]]; // input features (years)
const ys = [[1200],[1280],[1330],[1380]]; // supply values
const model = new MLR(xs, ys);
const prediction = model.predict([[2026]]); // → forecast for next year
```

This process repeats for **both supply and demand** for every crop.

---

d. Computation of the Forecast Gap

After prediction:

```
Forecast_Positive_Gap_t = max(0, Forecast_Demand_t - Forecast_Supply_t);
```

This shows the expected shortage (if demand > supply) for each crop.

The backend then returns:

```
{
  "labels": ["Paddy","Tea","Rubber"],
  "supply": [1400, 880, 920],
  "demand": [1550, 900, 910],
  "positive_gap": [150, 20, 0],
  "meta": { "forecast_year": 2026, "season_forecasted": "Maha" }
}
```

📊 4. Visualization and Interpretation

- **Chart.js** receives the data arrays and renders three pie charts.
- The white-text legends show crop names, and the chart titles update dynamically (e.g., *"Forecasted Demand Share (Maha 2026)"*).
- These results visually communicate the **forecasted market balance**.

---

5. Advantages of This Approach

Feature	Description
Automatic forecasting	No manual year entry; system intelligently predicts next year.
Lightweight ML model	Linear regression keeps computation fast for large datasets.
Interactive UI	Year/region toggles allow scenario testing without reloading.
Visual analytics	Pie charts quickly show relative crop importance and imbalances.

Methodology / System Implementation

The demand and supply forecasting module was developed using a client–server architecture. The frontend, implemented in HTML, JavaScript, and Chart.js, allows users to select regions, seasons, and historical years to define the training window. These parameters are sent to the backend via a REST API (`/api/forecast`). The backend, built with Node.js and Express, employs a Multivariate Linear Regression model from the `ml-regression-multivariate-linear` library to learn annual trends of supply and demand per crop. The model predicts the subsequent year's values (max year + 1). The system computes the expected demand–supply gap and returns the results to the client, which visualizes them through three interactive pie charts for demand, supply, and positive gap."