# Discount Management

|  |  |
| --- | --- |
| **Discount Management** | 1) INVENTORY BASED DISCOUNT |
| 2) SEASONAL DISCOUNT (Based on seasons/ weathers and special events) |
| 3) PRODUCT DEMAND DISCOUNT |
| 4) BUNDLE DISCOUNT |
| 5) PRODUCT CATEGORY SPECIFIC DISCOUNT |
| 6) PRICE VELOCITY DISCOUNT |

**INVENTORY BASED DISCOUNT**

**Why Use RL for Inventory-Based Discount Prediction?**

Reinforcement Learning (RL) is a great choice **if your goal is to optimize discounting dynamically** based on real-time data. **Proximal Policy Optimization (PPO)** is suitable because:

* It **learns** from interactions (historical & real-time sales data).
* It **adjusts discounts** dynamically based on stock, sales trends, and seasonality.
* It can **balance exploration and exploitation**, ensuring better revenue.

**However, RL isn’t the only option.** Here are some alternatives:

| **Approach** | **Pros** | **Cons** |
| --- | --- | --- |
| **Supervised Learning (Regression/Classification)** | Simple to implement, works well with historical data | Needs **labelled data**, may not adapt to real-time changes |
| **Bayesian Optimization** | Efficient in uncertain environments | Can be complex to fine-tune |
| **Rule-Based Approach** | Easy to implement, interpretable | Hard to scale, lacks adaptability |
| **Reinforcement Learning (PPO, Q-Learning, etc.)** | Dynamic, learns optimal pricing, adapts to new trends | Needs simulations or real-time feedback loop |

👉 **have limited data and need a quick model, go with Regression.**  
👉 **need a dynamic system that adjusts in real-time, RL is best.**

**Features for R&D**

| **Feature** | **Description** | **Why It’s Needed?** | **Real-Time Availability?** |
| --- | --- | --- | --- |
| **Product\_ID** | Unique identifier for the product | To track inventory per product | ✅ Available |
| **Product\_Name** | Actual product name (e.g., "Nike Shoes") | Helps in product-specific trends | ✅ Available |
| **Store\_Admin\_ID** | Store admin managing the product | To personalize discount strategies per admin | ✅ Available |
| **Product\_Category** | Electronics, Clothing, etc. | Discounts may vary by category | ✅ Available |
| **Product\_Price** | Current price of the product | Essential for calculating discount | ✅ Available |
| **Discount\_Offered** | Discount currently applied | Helps analyse past discount effects | ✅ Available |
| **Stock\_Availability** | Current stock quantity | Inventory-based discounts depend on stock levels | ✅ Available |
| **Reorder\_Threshold** | Minimum stock before reordering | Helps in stock replenishment | ✅ Available |
| **Average\_Daily\_Sales** | Average units sold per day | High sales = lower discount, low sales = higher discount | ✅ Available |
| **Days\_of\_Stock\_Left** | Estimated days before stock runs out | Crucial for inventory planning | ✅ Available |
| **Competitor\_Price** | Price of a similar product from competitors | If competitors drop price, you may need to adjust discounts | ❌ Needs Web Scraping |
| **Customer\_Interest\_Score** | Based on clicks, views, and wishlists | Higher interest = lower discount needed | ❌ Needs Analytics |
| **Historical\_Discount\_Impact** | Impact of past discounts on sales | Helps in predicting future discount effectiveness | ✅ Available |
| **Seasonality\_Flag** | 0 = Normal, 1 = Seasonal demand | Discounts vary in seasonal periods | ✅ Available |
| **Day\_of\_Week** | Current day (Monday–Sunday) | Shopping trends change by day | ✅ Available |
| **Festive\_Season** | 0 = No, 1 = Yes | Discounts often increase during festivals | ✅ Available |
| **Recommended\_Discount (Target)** | Suggested discount percentage | This is what we predict! | ❌ Needs for RL Model |

**📌 Can We Get All This Data in Real Time?**

✅ Most data are available in e-commerce platforms.  
❌ Some data (Competitor Price, Customer Interest) **needs external sources** (Scraping, Analytics).

The dataset now contains the following columns:

**Product\_ID** → Unique identifier for each product  
**Store\_Admin\_ID** → The store admin managing the product  
**Product\_Category** → Category of the product (e.g., Electronics, Clothing)  
**Product\_Name** → Name of the product  
**Product\_Price** → Price of the product  
**Discount\_Offered** → Current discount applied (can be used as an input feature)  
**Stock\_Availability** → Number of units available in stock  
**Reorder\_Threshold** → Minimum stock level before reordering  
**Average\_Daily\_Sales** → Average units sold per day  
**Days\_of\_Stock\_Left** → Estimated days before stock runs out  
**Competitor\_Price** → Price of similar products from competitors  
**Customer\_Interest\_Score** → Score indicating customer demand  
**Historical\_Discount\_Impact** → Effect of past discounts on sales  
**Seasonality\_Flag** → Indicates if the product is affected by seasonal trends  
**Day\_of\_Week** → The weekday (useful for demand fluctuations)  
**Festive\_Season** → Indicates if a festive event is happening (e.g., Christmas, Black Friday)

**How Is the RL Model Choosing Discounts?**

RL model (trained using **PPO - Proximal Policy Optimization**) is learning to pick discount values based on:

1. **Product Demand** – Higher customer interest encourages offering discounts.
2. **Stock Availability** – If stock is low, it may reduce discounts to maximize profit per item.
3. **Competitor Pricing** – If competitors have lower prices, it may increase discounts to compete.
4. **Sales Impact** – The model balances between revenue loss (due to discounts) and demand increase.

**BUNDLE DISCOUNT:**

**🔍 Bundle Discount = Buying More Items Together → Extra Discount**

A **bundle discount** is applied when:

1. **Buying multiple units of the same product** (e.g., **2 laptops instead of 1**).
2. **Buying a group of different products together** (e.g., **Laptop + Mouse + Keyboard** as a bundle).

**📌 Examples of Bundle Discounts**

| **Scenario** | **Discount** |
| --- | --- |
| Buying **2 or more** of the same item (e.g., **2 Laptops**) | ✅ 5% Discount |
| Buying a **predefined bundle** (e.g., **Laptop + Mouse + Keyboard**) | ✅ 10% Discount |
| Buying a **fashion combo** (e.g., **Shirt + Jeans + Shoes**) | ✅ 15% Discount |
| Buying an **office setup** (e.g., **Desk + Chair + Lamp**) | ✅ 12% Discount |

**📊 How Does the Model Learn Bundle Discounts?**

1. **Identifies purchase patterns**
   * Checks if people **usually buy these items together**.
   * Example: Many users buy **"Laptop + Mouse + Keyboard"** together → It should get a discount.
2. **Analyses past sales data**
   * If the bundle was sold many times, **give a bigger discount** to increase sales.
   * If the bundle is **not selling well**, increase the discount further.
3. **Considers stock availability**
   * If there is **high stock**, apply a bigger discount.
   * If stock is **low**, keep the discount small.
4. **Looks at competitor pricing**
   * If a competitor offers **10% off on a laptop bundle**, your system can suggest **11%** to stay competitive.

**Overview:**

This script enhances bundle discount recommendations using **Reinforcement Learning (RL)** with **Proximal Policy Optimization (PPO)** from **Stable-Baselines3**.

* **Data Preprocessing:** Loads and prepares discount dataset.
* **Custom Gym Environment:** Defines a simulation where the AI learns to predict discounts.
* **Training with PPO:** Uses reinforcement learning to optimize discount predictions.
* **Prediction:** Tests the trained model on a random sample.

**Step-by-Step Explanation:**

**1️ Data Loading & Preprocessing**

✅ Loads the dataset from CSV.  
✅ Normalizes column names (to lowercase and removes spaces).  
✅ Ensures necessary columns exist (e.g., seller\_id, product\_id, bundle\_id).  
✅ Fills missing suggested\_discount values using historical\_discount \* 0.9.  
✅ Converts categorical features (seller\_id, bundle\_id, etc.) into numerical values using LabelEncoder().  
✅ Scales numeric columns between **0 and 1** using MinMaxScaler().  
✅ Removes any NaN or infinite values.

**2️ Creating the Custom Gym Environment**

✅ Defines a **BundleDiscountEnv** class that simulates the RL environment.  
✅ Observation space includes all feature columns except suggested\_discount.  
✅ Action space: AI decides discount values between **0% and 50%**.  
✅ **Reward Calculation:**

* **Closer the prediction to the actual discount → Higher reward.**
* **Larger deviation → Lower reward (penalty).**

**3️ Training with PPO Algorithm**

✅ Creates a **DummyVecEnv** wrapper for training stability.  
✅ Defines **PPO hyperparameters** (learning rate, batch size, discount factor, etc.).  
✅ Trains the model for **50,000 timesteps** using **PPO ("MlpPolicy")**.  
✅ Saves the trained model as "ppo\_bundle\_discount\_enhancement".

**4️ Making Predictions**

✅ Selects a **random sample** from the dataset.  
✅ Removes the target column (suggested\_discount).  
✅ **Feeds the sample into the trained PPO model** for prediction.  
✅ The model outputs the **predicted discount** (in %).

**Price velocity discount:**