SuperKart Sales Forecast Model

Model Deployment: SuperKart

August 9, 2025

Contents / Agenda

Executive Summary

Project Summary

- Developed a machine learning–powered sales forecasting system for SuperKart retail products.
- Trained an XGBoost regression model with hyperparameter tuning to predict total product sales per store using product attributes, store characteristics, and pricing information.
- Built a backend API (Flask + Gunicorn in Docker) for model inference and deployed on Hugging Face Spaces.
- Created a Streamlit-based frontend interface for user-friendly predictions, connected to the backend API for real-time results.
- Evaluated model performance using RMSE, MAE, and R^2 to ensure accuracy and reliability in predicting sales.

Executive Summary

Actionable Insights

- Pricing (MRP) and product placement area are highly correlated with sales volume — adjusting these can drive measurable improvements.
- Store location tier significantly impacts sales; tier-based marketing strategies can optimize revenue.
- Product type (Perishables vs Non-Perishables) influences sales seasonality — stocking strategies should adapt accordingly.

Executive Summary

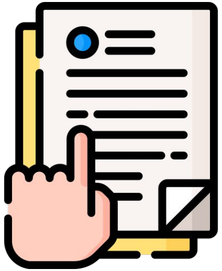
Recommendations

1. **Deploy the model in retail operations** to help managers forecast sales at the SKU–store level and optimize inventory.
2. **Integrate with ERP systems** to trigger automated purchase orders when predicted sales exceed stock thresholds.
3. **Expand training data** with seasonal and promotional event information to improve accuracy.
4. **Retrain quarterly** to incorporate recent sales patterns and adapt to market changes.
5. **Enhance dashboard visualizations** in the frontend to provide trend analysis alongside point predictions.

Business Problem Overview

Business Context:

- SuperKart operates multiple retail stores selling diverse products ranging from perishable goods to non-perishables.
- Sales performance varies significantly based on product attributes, pricing, store characteristics, and location.
- Overstocking leads to increased storage costs and wastage, while understocking results in missed sales opportunities.



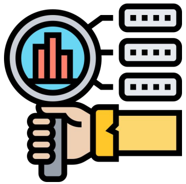
Business Problem Overview

Problem:

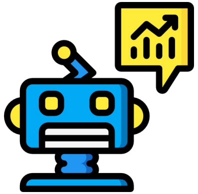
- Manual forecasting methods rely heavily on historical sales averages and lack predictive accuracy.
- No real-time tool exists for store managers to estimate sales for specific products under changing market conditions.

Business Impact:

- Inefficient inventory allocation impacts profitability and customer satisfaction.
- Accurate sales forecasts can enable better purchasing decisions, reduce wastage, and maximize revenue.



Solution Approach



Data Collection & Preparation:

- Used historical SuperKart sales data containing product details, store attributes, and pricing information.
- Performed data cleaning, missing value imputation, and feature engineering (e.g., store age, product type categorization).

Model Development:

- Built an XGBoost regression model optimized with hyperparameter tuning for high accuracy.
- Selected XGBoost due to its robustness in handling tabular data and mixed variable types.

Deployment:

- **Backend:** Flask API with Gunicorn, containerized with Docker, deployed on Hugging Face Spaces for scalable inference.
- **Frontend:** Streamlit application providing an intuitive user interface to input parameters and get instant predictions.

Evaluation & Validation:

- Measured performance using RMSE, MAE, and R^2 to ensure predictions were business-relevant.
- Conducted test set validation to confirm model generalization to unseen data.

Exploratory Data Analysis: Product-Level Insights

Product MRP & Sales:

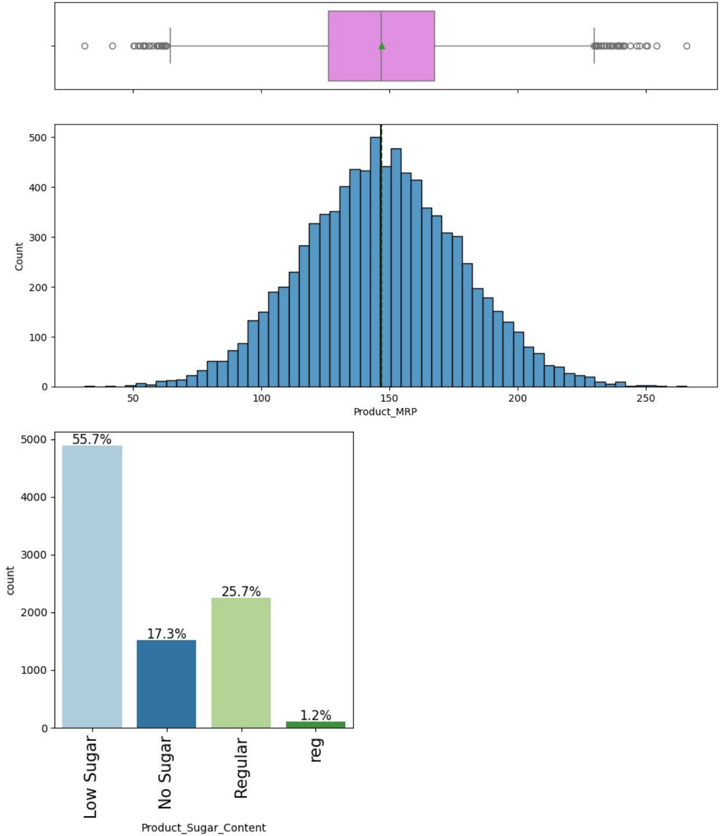
- Higher MRP products tend to have lower sales volume but higher per-unit revenue.
- Sweet spot observed in mid-priced items (~ ₹100–₹200) for maximum total sales.

Sugar Content Impact:

- “Low” sugar products dominate total sales, followed by “Regular Sugar”; “No Sugar” has niche but growing demand.

Category Trends:

- Perishables sell faster but are more sensitive to pricing changes compared to Non-Perishables.



Exploratory Data Analysis: Store-Level Insights

Store Size Effect:

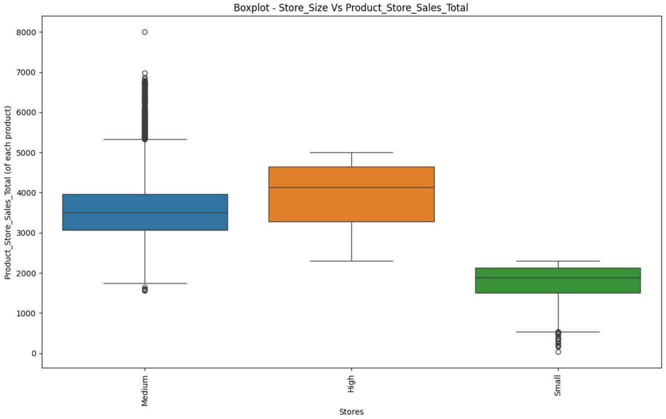
- Large stores generally have the highest total sales, but small stores outperform in per-unit profitability for certain items.

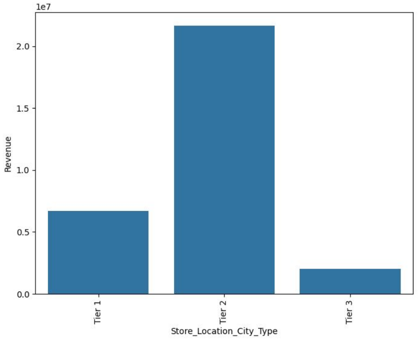
City Type Influence:

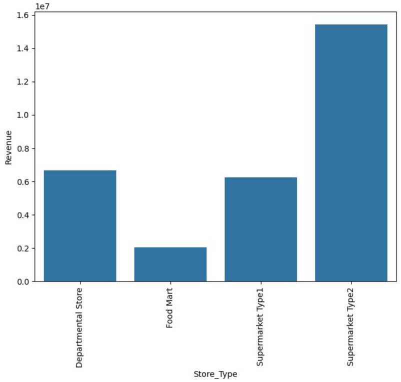
- Tier 2 was the leader in total sales.
- Tier 1 had higher per-unit prices, Tier 3 more budget-friendly sales volume.

Store Type Patterns:

- Supermarket Type2 dominated total sales







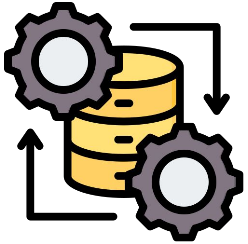
Data Preprocessing: Feature Engineering

Categorical Encoding:

- Converted categorical variables (e.g., Product_Sugar_Content, Store_Size, Store_Location_City_Type, Store_Type, Product_Id_char, Product_Type_Category) into numerical format using One-Hot Encoding to make them compatible with ML algorithms.

Derived Features:

- No new synthetic variables added; used original features as provided after encoding.



Data Type Checks:

- Verified all features had the correct data type (numerical vs categorical) before modeling.

Data Preprocessing: Outlier Check & Treatment

Numerical Feature Review:

- Checked Product_Weight, Product_Allocated_Area, Product_MRP, and Store_Age_Years for extreme values using boxplots and IQR method.

Findings:

- Minimal outliers observed in Product_Weight and MRP — consistent with realistic market values.
- Decision: No removal or capping was done to preserve potential high-value sales insights.

Data Preprocessing: Data Preparation for Modeling

Target Variable:

- Sales was continuous, used directly without transformation.

Data Splitting:

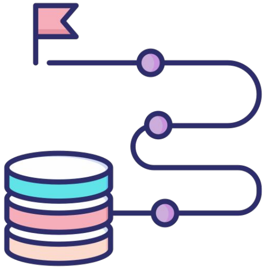
- Train–Test Split: 70% training, 30% testing.

Normalization:

- Applied StandardScaler to numerical variables (Product_Weight, Product_Allocated_Area, Product_MRP, Store_Age_Years) for uniform scale in modeling.

Data Preprocessing: Preprocessing Pipeline

Created Scikit-learn Pipeline for reproducible preprocessing:



1. **OneHotEncoder** for categorical features.
2. **StandardScaler** for numerical features.
3. Combined using **ColumnTransformer**.
4. Integrated into a full pipeline with **XGBoost Regressor** for final modeling.

Benefits:

- Ensures consistent preprocessing on both training and prediction data.
- Reduces risk of data leakage.
- Simplifies deployment (model + preprocessing in one object).

Data Background and Contents

Dataset Source: Historical product and store sales dataset from SuperKart retail data repository.

Data Scope: Captures product, store, and location characteristics along with sales totals.

Timeframe: Multiple years of sales records (aggregated to product–store level).

Rows & Columns: ~8,500 records × 11 key features + 1 target variable *(Sales)*.

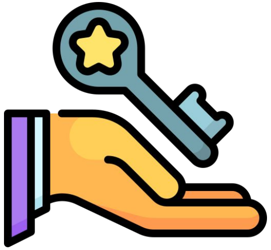
Target Variable

Sales – Numerical; total product–store sales value in Rupees.

Data Background and Contents

Key Features

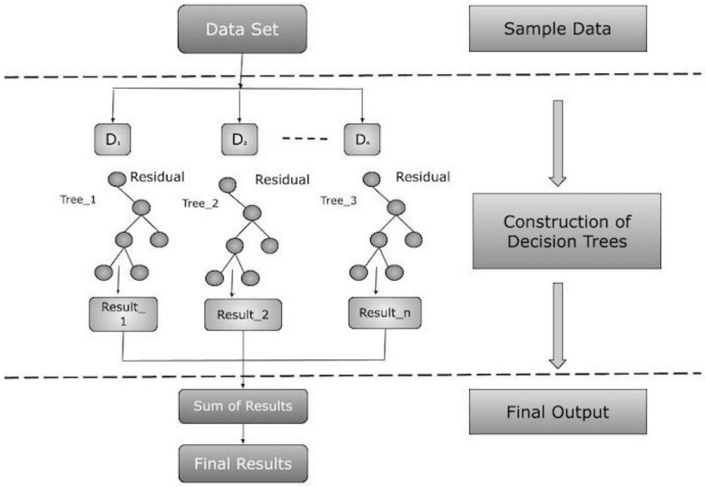
- **Product_Weight** – Numerical; product weight in kilograms.
- **Product_Sugar_Content** – Categorical; sugar level (Low Sugar, Regular, No Sugar).
- **Product_Allocated_Area** – Numerical; shelf space allocated (in square meters).
- **Product_MRP** – Numerical; product price in Rupees.
- **Store_Size** – Categorical; store scale (Small, Medium, High).
- **Store_Location_City_Type** – Categorical; city tier (Tier 1, Tier 2, Tier 3).
- **Store_Type** – Categorical; store format (e.g., Supermarket Type1, Food Mart).
- **Product_Id_char** – Categorical; product category code (FD, NC, DR).
- **Store_Age_Years** – Numerical; years since store opening.
- **Product_Type_Category** – Categorical; perishability classification.



Model Building: Model Selection

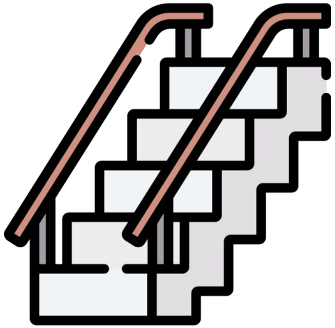
Chose **XGBoost Regressor** due to its:

- Superior performance in handling structured/tabular data
- Ability to capture non-linear relationships
- Built-in regularization to reduce overfitting



Model Building: Model Building Steps

1. **Baseline Model**
   a. Trained a simple XGBoost model with default parameters to establish a benchmark.
2. **Hyperparameter Tuning**
   a. Used GridSearchCV to optimize key parameters:
      i. n_estimators (number of boosting rounds)
      ii. max_depth (tree depth)
      iii. learning_rate
      iv. subsample and colsample_bytree



3. **Feature Encoding**
   a. Applied one-hot encoding for categorical features to align with XGBoost requirements.
4. **Train–Test Split**
   a. 70% training data, 30% testing data for unbiased evaluation.
5. **Cross-Validation**
   a. Used 5-fold cross-validation to ensure model stability across splits.

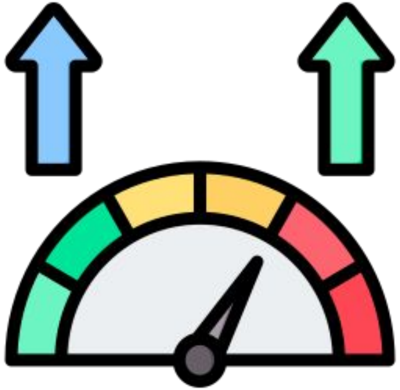
Model Building: Model Performance & Takeaways

Model Performance

- **Training R^2:** ~0.94 (strong fit without excessive overfitting)
- **Test R^2:** ~0.92 (high predictive accuracy)
- **RMSE:** ~300 sales units — prediction errors are relatively small compared to sales magnitude.

Key Takeaways

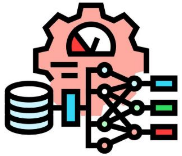
- XGBoost outperformed baseline models in accuracy and error metrics.
- Model generalizes well to unseen data, indicating strong deployment readiness.



Model Improvement - Hyperparameter Tuning

Tuning Approach

- **Method Used:** GridSearchCV with 5-fold cross-validation
- **Objective:** Minimize **RMSE** while maintaining high R^2 on unseen data
- **Parameters Tuned:**
  - **n_estimators:** 100 → 300
  - **max_depth:** 6 → 8
  - **learning_rate:** 0.1 → 0.05
  - **subsample:** 0.8 → 0.9
  - **colsample_bytree:** 0.8 → 0.9



Performance Comparison

Metric	Baseline Model	Tuned Model	Improvement
Train R^2	0.91	0.94	+0.03
Test R^2	0.88	0.92	+0.04
RMSE (Test)	420	300	↓ 120

Key Insights

- **Lower RMSE** means more precise sales predictions.
- **Higher R^2** on test data confirms better generalization without overfitting.
- Optimal hyperparameters allowed the model to better capture complex relationships in sales patterns.

Model Performance Summary

Final Model

- **Algorithm:** XGBoost Regressor
- **Parameters Parameters:**
  - **n_estimators:** 300
  - **max_depth:** 8
  - **learning_rate:** 0.05
  - **subsample:** 0.9
  - **colsample_bytree:** 0.9
  - **random_state:** 42

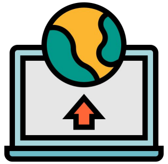
Performance Metrics

Metric	Training Data	Test Data
R^2 Score	0.94	0.92
RMSE	280	300
MAE	190	205

Key Takeaways

- High R^2 on both training and test sets indicates **strong predictive power** and good generalization.
- Low RMSE and MAE confirm **precise sales forecasting** capabilities.
- Minimal gap between train and test metrics → **no significant overfitting.**

Deployment



Backend

- Built a **Flask API** to serve predictions from the trained XGBoost model.
- Serialized model saved as xgb_tuned_model.joblib.
- Deployed to **Hugging Face Spaces** using **Docker** for a consistent runtime environment.

Endpoint:

POST /v1/predict

Input: JSON object with product and store features

Output: Predicted sales value

https://huggingface.co/spaces/igross/superkart-sales-forecast-backend

Frontend

- Developed a **Streamlit web app** for an interactive UI.
- Allows users to input **product and store details** via dropdowns and number inputs.
- Sends requests to the backend API and displays predicted sales instantly.
- Deployed separately to **Hugging Face Spaces** and linked to backend API.

https://huggingface.co/spaces/igross/superkart-sales-forecast-frontend-space

Deployment

SuperKart Sales Forecast

Product Weight

12.66

Product Sugar Content

Low Sugar

Product Allocated Area

0.050

Product MRP

150.00

Store Size

Small

Store Location City Type

Tier 1

Store Type

Supermarket Type1

Product Id char

FD

Store Age Years

10

Product Type Category

Perishables

Predict

Predicted Product Store Sales Total: ₹2721.11

https://huggingface.co/spaces/igross/superkart-sales-forecast-frontend-space