

**FINAL REPORT**

**Prepared by:**

**Rawan Asaad Hemdan**

**Akram Gamal Mohamed**

**Mohamed Abdelhamid Wagdy**

**Tarek Mohamed Adel**

**Alaa Eid Mohamed**

**Yosef Ahmed Shaban**

# Abstract

Forecasting sales accurately has become more important than ever for businesses looking to avoid overstocking, reduce losses, and better meet customer demand. This project, titled **Sales Forecasting and Demand Prediction**, takes a practical, hands-on approach to that challenge. Using real-world retail data, we built a complete system that helps estimate future sales by analyzing factors like seasonality, store types, product categories, and promotional campaigns.

The project was divided into milestones to make the process more structured. We started by exploring and cleaning multiple datasets, then moved on to engineering features that could better capture patterns in the data. For modeling, we focused on machine learning techniques that are well-suited for tabular data — especially tree-based algorithms like **Random Forest** and **XGBoost**. These models were trained and evaluated using key metrics like **MAE** and **RMSE** to make sure our predictions were accurate and reliable.

We also experimented with a basic dynamic pricing concept. Although simple, it demonstrated how sales predictions could influence pricing decisions in a real-world business setting.

Overall, the final result is a forecasting framework that can help with inventory management, campaign planning, and operational efficiency. This report walks through the entire process step by step, highlighting how data can be turned into actionable insights that make a tangible impact.

Table of Contents

[Abstract 2](#_Toc197865077)

[Introduction 5](#_Toc197865078)

[Milestone 1: Data Collection, Exploration, and Preprocessing 6](#_Toc197865079)

[1. Introduction 6](#_Toc197865080)

[1.1 Business Problem 6](#_Toc197865081)

[1.2 Project Goal 6](#_Toc197865082)

[1.3 Dataset Overview 7](#_Toc197865083)

[2. Exploratory Data Analysis (EDA) 7](#_Toc197865084)

[2.1 What We Found in the Datasets 7](#_Toc197865085)

[2.2 Initial Observations & Stats 8](#_Toc197865086)

[Milestone 2: Advanced Data Analysis and Feature Engineering 8](#_Toc197865087)

[1. Objective 8](#_Toc197865088)

[2. Time Series Behavior and Stationarity Check 9](#_Toc197865089)

[3. Feature Engineering 9](#_Toc197865090)

[3.1 Time-Based Features 9](#_Toc197865091)

[3.2 Lag Features 10](#_Toc197865092)

[3.3 Rolling Averages and Trends 10](#_Toc197865093)

[3.4 Promotions and Holidays 10](#_Toc197865094)

[3.5 Price Simulation 11](#_Toc197865095)

[4. Transforming Features for Modeling 11](#_Toc197865096)

[5. Visual Analysis and Correlations 11](#_Toc197865097)

[6. Feature Testing and Importance 12](#_Toc197865098)

[7. What We Learned 12](#_Toc197865099)

[8. Conclusion 13](#_Toc197865100)

[Milestone 3: Machine Learning Model Development and Evaluation 13](#_Toc197865101)

[1. Objective 13](#_Toc197865102)

[2. Setting Up the Problem 13](#_Toc197865103)

[3. Models We Tried 14](#_Toc197865104)

[4. Feature Selection and Encoding 14](#_Toc197865105)

[5. Evaluation Metrics 15](#_Toc197865106)

[6. Results and Observations 15](#_Toc197865107)

[7. Feature Importance Insights 16](#_Toc197865108)

[8. Error Behavior 16](#_Toc197865109)

[9. Conclusion 17](#_Toc197865110)

[Milestone 4: Website Integration and Model Deployment 17](#_Toc197865111)

[Reinforcement Learning-Based Price Optimization 19](#_Toc197865112)

[Challenges Faced 20](#_Toc197865113)

[Future Work 21](#_Toc197865114)

[Conclusion 21](#_Toc197865115)

[References 22](#_Toc197865116)

# Introduction

Forecasting future sales might seem straightforward at first, but once you consider how many things can affect demand — from holidays and promotions to location and product type — it becomes clear that it's anything but simple. For retailers, guessing wrong can be costly. If they stock too much, they risk waste and storage problems. If they stock too little, they lose out on sales and frustrate customers. This project was born out of that challenge: how can we use real data to make these decisions smarter?

Our main focus in this project was to predict product-level sales on a daily basis using historical data. The data came from a real-world setting — a large grocery chain in Ecuador called Corporación Favorita. They provided several datasets, including daily sales records, transaction counts, store metadata, holiday schedules, and even oil prices (which can affect spending patterns in oil-exporting economies like Ecuador). This gave us a lot to work with, but it also meant we had to be thoughtful about cleaning, merging, and preparing everything before jumping into modeling.

We approached the problem step by step, breaking it down into clear milestones. The first phase was all about data preparation: understanding each file, handling missing values, merging datasets carefully, and checking for any obvious errors or inconsistencies. From there, we moved into more advanced analysis — testing patterns over time, creating rolling statistics and lag features, and experimenting with which variables actually influenced sales.

For modeling, we focused on machine learning approaches that work well with structured data. We tested multiple regressors, but Random Forest and XGBoost stood out for their ability to handle non-linear relationships and rank feature importance. We didn’t stop at just predictions, though — we also explored a basic dynamic pricing strategy based on those forecasts, tying our insights back to real business decisions.

In the end, this project isn’t just about building a model. It’s about creating something that could actually help a retailer make smarter choices day to day. Whether it's predicting how much milk to stock next Tuesday or figuring out if a promotion really makes sense this weekend, our system provides the kind of support businesses need to act confidently — with data on their side.

# Milestone 1: Data Collection, Exploration, and Preprocessing

## 1. Introduction

### 1.1 Business Problem

Retail businesses constantly struggle with two big problems: how much to stock and how to price their products. If you overstock, you waste money. If you understock, you lose sales and frustrate customers. To make things harder, prices don’t always reflect demand — especially when factors like holidays, paydays, or sudden market events (like oil price changes) come into play.

In this project, we tried to solve both challenges. Our goal was to build a system that uses historical sales data to predict future demand. With those predictions, we can also make smarter pricing decisions. The entire pipeline is based on data from Corporación Favorita, a large grocery retailer in Ecuador.

### 1.2 Project Goal

This project centers on **time series forecasting of unit sales** for **Corporación Favorita** using historical transaction data and external signals such as holidays and oil prices. The ultimate goal is to:

**1- Predict daily item-level sales** for each store to support inventory management, staffing, and supply chain operations.

**2- Design a dynamic pricing framework** that adjusts product prices based on demand patterns, product attributes, seasonal trends, and macroeconomic indicators.

Milestone 1 was all about building a clean and usable dataset for future machine learning models. We collected several raw datasets and did the hard work of merging, cleaning, and understanding them. Some of the key tasks included:

* Figuring out how each dataset fits together (sales, stores, holidays, etc.)
* Cleaning missing values and fixing weird data types
* Identifying trends in sales and checking if things like holidays or promotions really affect demand
* Creating a final merged dataset with no duplicates and minimal noise

### 1.3 Dataset Overview

Here’s what we worked with:

**train.csv**: Over 3 million rows of daily sales for different items in stores

**test.csv**: Similar to train but without the target column (sales)

**stores.csv**: Info about each store (location, type, cluster

**transactions.csv**: Daily transaction counts per store

**oil.csv**: Daily oil prices (interesting macroeconomic context)

**holidays\_events.csv**: Local and national holidays that might impact customer behavior

**sample\_submission.csv**: Template for making submissions in the original competition

**Contextual Note:**

In Ecuador, government salaries are usually paid on the 15th and last day of the month — this seemed like something that could influence shopping patterns. Also, a major earthquake hit the region in April 2016, which probably caused an unusual spike in essential goods. We kept this in mind when looking at outliers in the data.

## 2. Exploratory Data Analysis (EDA)

### 2.1 What We Found in the Datasets

Here’s a summary of the datasets we examined:

The sales data was huge and mostly fine, but clearly skewed — a lot of entries with low values, but a few with extremely high sales (over 100,000 units). Promotions were irregular and unpredictable — some days had hundreds of items promoted, others had none

The oil price data had a few missing values (~43), which isn’t ideal but not unmanageable.

Transaction counts varied a lot from store to store — some had 100 transactions a day, others had thousands.

Dates were originally stored as strings, so we converted everything to proper datetime formats to help with time-based operations.

### 2.2 Initial Observations & Stats

Some quick stats and observations:

* Sales had a mean of ~358 units but a median of only 11, which shows the data is very skewed. Most days, most items sell a small amount.
* Oil prices ranged from around $26 to $110, with an average of about $68. The dips matched global economic trends we looked up.
* Some stores ran massive promotions, especially around local events or holidays. Others rarely had items on sale.
* Transactions seemed like a strong predictor of sales, but weren’t perfectly linear.

These early findings helped us decide which features would be useful later — like keeping track of holidays, transaction volume, and promotions when building forecasting models.

# Milestone 2: Advanced Data Analysis and Feature Engineering

## 1. Objective

After completing data cleaning and merging in Milestone 1, our next step was to dig deeper into the dataset and make it more useful for prediction. In this phase, we focused on two main goals: first, understanding the behavior of the data over time, and second, creating new features that could help the machine learning model make better predictions. We wanted to capture trends, repetitive patterns, special events, and other signals that might influence how much a product sells on a given day.

We used a mix of time series analysis, basic statistics, and domain knowledge to engineer features we believed would be useful for learning demand behavior. By the end of this milestone, we had built a dataset with many new variables that gave the model much more context than just raw sales numbers.

## 2. Time Series Behavior and Stationarity Check

We started with a basic check: whether the sales data was stationary or not. In time series forecasting, a stationary dataset has a consistent average and variance over time, which makes it easier for some models to work well. We used the **Augmented Dickey-Fuller (ADF) test** to check this.

Most of the results showed **p-values greater than 0.05**, meaning the data was **not stationary** — which is common in sales datasets. This confirmed that we needed to include time-aware features like lag variables and rolling averages, rather than relying on raw time steps alone.

## 3. Feature Engineering

Once we understood the nature of the data, we focused on building new features that would give the model better visibility into trends and cycles. Here's a breakdown of what we added:

### 3.1 Time-Based Features

We extracted several components from the date column, such as:

* day, month, year, and day\_of\_week
* A week variable to catch seasonal patterns
* Flags like is\_weekend
* A quarter variable to indicate calendar quarter (1-4)
* A season variable categorized as:  
    • 0 for late winter or early spring (February–March)  
    • 1 for spring (April–June)  
    • 2 for summer (July–August)  
    • 3 for fall (September–November)  
    • 4 for winter (December–January)
* Days\_to\_Thanksgiving and Days\_to\_Christmas to track proximity to major holidays

These features help the model understand repeating calendar cycles and business behaviors that align with specific days.

### 3.2 Lag Features

To give the model access to historical context, we created lag variables:

* lag\_1: Yesterday’s sales
* lag\_7: Sales a week ago
* lag\_14: Sales 2 weeks ago
* lag\_promo\_1: Whether the item was on promotion the previous day

These are extremely helpful in sales forecasting since past behavior is often the best clue to future performance.

### 3.3 Rolling Averages and Trends

We wanted to capture short- and medium-term trends by using rolling statistics:

* 7-day **rolling median** for onpromotion and 7-day and 14-day **rolling medians** for sales

These features helped smooth out noisy data and showed whether a product was currently in a growth or decline trend.

### 3.4 Promotions and Holidays

We added new indicators for:

* Whether the product was on promotion
* Whether the date was a holiday (local, regional, or national)
* The type of holiday (encoded numerically)

This gave the model the ability to understand temporary demand spikes during special events.

### 3.5 Price Simulation

To enrich the dataset with price-related information, we simulated realistic product prices:

* We created a base\_price for each product family using domain-informed estimates.
* To emulate dynamic pricing, we added a daily\_variation factor — a random value between 0.9 and 1.1 — to reflect day-to-day fluctuations.
* The final price feature was calculated as the product of base\_price and daily\_variation, rounded to two decimal places.

## 4. Transforming Features for Modeling

After engineering the features, we had to prepare them for model input. That included:

* Log-transforming the sales variable using log(sales + 1) to reduce skewness and avoid negative predictions.
* For categorical features like family, city, holiday\_type, and store\_type, we used **binary encoding** instead of one-hot encoding to avoid an explosion in dimensionality.

## 5. Visual Analysis and Correlations

We built a set of visualizations to better understand our engineered features and their impact:

* **Heatmaps** showed that rolling averages, lagged sales, and onpromotion were highly correlated with the target variable.
* **Bar plots** grouped by day of the week showed lower sales on weekends, especially Sundays.
* **Box plots** across store types and holiday types revealed how different contexts affected average sales.
* **Line graphs** of sales vs. rolling averages helped validate that our smoothing features were capturing trends effectively.

## 6. Feature Testing and Importance

To see how helpful our new features were, we ran a basic **Random Forest Regressor** and plotted feature importances.

Key observations:

* price, family, and onpromotion consistently ranked among the top features.
* Date-based features like day and week were less dominant but still added helpful cyclic signals.

## 7. What We Learned

* **Time-driven demand** is crucial: Sales patterns follow clear weekly, monthly, and seasonal cycles. Capturing these through time-based features significantly improved the model's ability to forecast trends.
* **Promotional flags and holiday effects** made a big difference — but only for certain product categories.
* **Simulated price dynamics** added meaningful context: Even approximate price information introduced additional variance and helped the model distinguish between high- and low-value items.
* **Lag and rolling features** provided historical memory and trend detection, both of which are essential in forecasting problems.
* **Efficient encoding** (using binary encoding) allowed us to incorporate categorical variables without bloating the feature space, improving computational efficiency.

## 8. Conclusion

This milestone turned a basic historical sales dataset into a much richer, feature-packed version that gives the forecasting model more context. We engineered a wide range of variables, from lagged sales to smoothed oil prices, and tested their usefulness through visualization and early model feedback.

By the end of this phase, we were confident that the dataset was well-prepared for machine learning — especially for models like Random Forest and XGBoost that benefit from strong, structured inputs. These features laid the groundwork for the training and evaluation work that followed in Milestone 3.

# **Milestone 3: Machine Learning Model Development and Evaluation**

## 1. Objective

Now that the dataset was fully cleaned and packed with meaningful features, the next logical step was building a machine learning model that could accurately predict daily product sales. Our main goal in this milestone was to choose the right algorithms, train them on our enriched data, and evaluate how well they performed.

We tried to approach the modeling phase with both accuracy and practicality in mind. We wanted models that not only predicted well but also made sense in a business context — where interpretability, speed, and robustness matter.

## 2. Setting Up the Problem

We approached this as a **regression problem** since we were predicting a continuous variable — sales — for specific item-store-date combinations.

Instead of randomly splitting the data (which would create leakage in time series), we made sure to split it **chronologically**. The training set included older data, while the test set included more recent dates. This way, we simulated real-world forecasting: training the model on the past and using it to predict the future.

## 3. Models We Tried

**1. Linear Regression (Baseline)**

We started with a basic linear regression model just to get a sense of how the features behave. It was fast and easy to interpret, but not great at capturing the complex, non-linear relationships in our data.

**2. Random Forest Regressor**

Next, we tried a **Random Forest**, which is an ensemble of decision trees. This model handles non-linearity and interactions between features very well and gives us access to **feature importance** plots. It worked significantly better than linear regression.

**3. XGBoost Regressor**

We also trained an **XGBoost** model, which is a more advanced tree-based method using gradient boosting. It’s known for being highly effective with structured/tabular data and tends to perform better than Random Forest in many real-world cases.

4. **LightGBM Regressor**

Finally, we used LightGBM, a fast, distributed, and efficient implementation of gradient boosting developed by Microsoft. It supports histogram-based learning and categorical features, making it especially powerful for large datasets.

## 4. Feature Selection and Encoding

Before training the models, we made sure the features were formatted correctly:

* Categorical variables like family, city, store\_type, and holiday\_type were **binary-encoded**.
* Numerical features like rolling means, oil price, and lags were already scaled and cleaned in the previous milestone.

We didn’t use one-hot encoding since it would have resulted in too many sparse columns due to the number of unique products and stores.

## 5. Evaluation Metrics

To measure performance, we used several regression metrics:

|  |  |
| --- | --- |
| **Metric** | **Why It Matters** |
| **MAE (Mean Absolute Error)** | Easy to interpret — tells us how far off, on average, our predictions were |
| **MSE (Mean Squared Error)** | Penalizes bigger mistakes more heavily |
| **RMSE (Root Mean Squared Error)** | More sensitive to large errors, but still in the same units as the target |
| **R² Score** | Shows how much of the variation in sales is explained by the model |

These metrics gave us a well-rounded picture of how well each model was performing.

## 6. Results and Observations

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **MAE** | **RMSE** | **R² Score** | **RMSLE** | **MSE** |
| Linear Regression | 1.57 | 1.88 | 0.5 | 0.65 | 3.54 |
| Random Forest | 0.43 | 0.64 | 0.94 | 0.26 | 0.42 |
| LGBM | 0.42 | 0.57 | 0.95 | 0.25 | 0.42 |
| **XGBoost** | **0.39** | **0.23** | **0.96** | **0.23** | **0.28** |

**XGBoost clearly performed the best** in our testing. It handled complex patterns like weekly seasonality, promotional effects, and holiday spikes much more accurately than the other models. The **Random Forest** wasn’t far behind and still gave us useful insights, especially through its feature importance rankings.

## 7. Feature Importance Insights

We used built-in feature importance tools from both Random Forest and XGBoost to understand which features had the most impact. The top features were:

* price
* family
* onpromotion
* dcoilwtico
* store\_nbr and store\_type (to a lesser extent)

This confirmed what we’d seen earlier in the EDA: recent performance and short-term trends are strong predictors of future demand.

## 8. Error Behavior

We also looked at how the models behaved when they made mistakes. We found:

* Good accuracy for steady products (things that sell in stable amounts day-to-day)
* Underprediction during **holidays and major promotions**, likely because those events are harder to model unless they appear frequently in the training data
* The residuals are roughly centered around zero, indicating no systematic over- or underestimation. However, the increasing spread of residuals at higher predicted values suggests heteroscedasticity, meaning the model's prediction error increases with the magnitude of predictions.

## 9. Conclusion

In this milestone, we successfully trained and tested several machine learning models on our feature-rich dataset. The results showed that **XGBoost was the best-performing model**, balancing accuracy and flexibility.

The models captured important signals like lag effects, short-term trends, and promotional boosts. More importantly, they provided a solid foundation for making actual sales forecasts that businesses could use for planning inventory, scheduling staff, and timing promotions.

The next step was to make these predictions more accessible — which we tackled in the deployment and web integration phase.

# Milestone 4: Website Integration and Model Deployment

To make the system accessible to users, we built a fully functional website that integrates our trained machine learning models for sales forecasting and demand prediction.

**Technologies Used:**

* **Backend:** Django (Python Web Framework)
* **Frontend:** HTML, CSS, JavaScript, Bootstrap
* **Model Integration:** pickle module to load trained models into Django views
* **Database:** PostgreSQL (for prototyping and local testing)

**Integration Process:**

1. After training and saving the best-performing model (XGBoost), we used the pickle library to serialize it.
2. In the Django project:
   * The model was loaded in the views.py file when the server started.
   * User inputs (e.g., item type, date, store) were collected via HTML forms.
   * These inputs were preprocessed and passed to the model to generate forecasts.
   * The predicted values were displayed directly on the website.

**User Interface:**

* The website includes:
  + A homepage describing the system
  + A form-based interface to input forecasting parameters
  + A results page that displays predicted sales
* The site is responsive and user-friendly, allowing stakeholders to easily interact with the forecasting engine.

**Challenges Faced:**

* Ensuring model compatibility across environments (especially between training in Jupyter Notebook and serving in Django)
* Managing feature scaling and encoding consistently between training and deployment
* Handling categorical variables properly for the prediction form

This phase made our solution more practical and closer to a real-world tool used by businesses or retail stores.

# Reinforcement Learning-Based Price Optimization

To extend the functionality of our demand forecasting system, we introduced a reinforcement learning (RL) model for dynamic pricing. While traditional models offer predictive insights based on historical data, the RL-based component actively **learns optimal pricing strategies** over time by maximizing long-term rewards — such as profit, customer retention, or inventory efficiency.

We used a simplified Q-learning framework to simulate pricing decisions under different demand and seasonal scenarios. The agent observes the current demand level, price sensitivity, and seasonality, then selects actions such as **increasing, decreasing, or maintaining the price**. It receives a reward based on how well that action performs in driving sales and maintaining competitiveness.

This approach offers several advantages:

* **Adaptivity:** The RL model learns from feedback, adapting to changing market behavior and demand trends.
* **Strategic Planning:** It goes beyond short-term prediction, making decisions that optimize cumulative performance over time.
* **Business Simulation:** Owners can simulate the impact of different pricing policies in advance before applying them in real settings.

The RL model is still in its early stages and currently operates in a simulated environment, but it demonstrates how intelligent pricing agents can assist businesses in navigating complex market dynamics.

# Challenges Faced

Throughout our project journey, we encountered several challenges—both technical and strategic—that shaped our decision-making process.

Initially, we worked extensively with the M5 Forecasting dataset for several weeks, completing both Milestone 1 and 2 using that data. However, when we reached the model development phase, we discovered that the M5 dataset wasn’t suitable for our specific goals. It lacked the necessary structure and features required by both our demand prediction model and reinforcement learning-based price suggestion model. Moreover, the dataset’s size—over 7GB—made it difficult to train efficiently on our local machines, even after attempting batching techniques and optimization strategies.

Due to these limitations, we shifted to the Favorita Grocery Sales Forecasting dataset. Our first trial used an older version of Favorita, but it didn’t contain all the necessary fields, especially those tied to pricing and external context. We then explored a newer, cleaned, and richer version of the dataset, which turned out to be a better fit. It included detailed information on item-store combinations, promotional activities, holidays, and other relevant indicators—essential for both of our models.

However, the Favorita dataset wasn't organized as a time-series by default, which limited our ability to apply deep learning models like LSTM or Transformers. We considered engineering the dataset to a time-series format but decided to proceed with XGBoost, which ultimately became our best-performing model, outperforming others like Random Forest and Linear Regression.

In addition to data-related challenges, we faced significant difficulties during deployment. We intended to host our system on Render, but even the free tier required a valid credit card with USD capabilities, which none of our team members had access to. This blocked us from deploying the website there. Furthermore, we initially used PostgreSQL for the backend, but after researching common hosting platforms, we found that MySQL was more widely supported and offered better compatibility—prompting us to switch the database configuration mid-development.

# Future Work

Although the system we built is functional and delivers actionable insights, there are several directions we plan to explore in future iterations:

* **Adopt advanced forecasting models** like LSTM and Transformers if more consistent time-series data becomes available.
* **Integrate real-time data streaming** to generate continuous predictions and pricing updates.
* **Expand the reinforcement learning module** beyond simulation into live environments, allowing for on-policy learning and personalized pricing.
* **Scale deployment** to a cloud platform (like AWS, or Azure) for improved availability and scalability.

# ****Conclusion****

This project aimed to bridge the gap between intelligent forecasting and customer-driven decision-making in retail environments. By leveraging real-world data from Corporación Favorita, we built a system that forecasts demand accurately and suggests data-driven pricing strategies.

Our **XGBoost-based model** delivered strong results and offered a reliable foundation for inventory planning and business operations. On top of that, we introduced a **reinforcement learning pricing agent** that simulates optimal price adjustments, helping business owners plan for both regular periods and high-demand seasons.

Importantly, the platform includes a **gamified challenge system**, where everyday customers can predict prices and earn reward points. This creates a more interactive shopping experience while also collecting valuable user insights.

In the end, our solution is more than just a forecasting tool. It's a smart retail companion designed to support both strategic business decisions and customer engagement — combining data science with real impact.

# References

[1] Kaggle, “Corporación Favorita Grocery Sales Forecasting,” [Online]. Available: https://www.kaggle.com/competitions/favorita-grocery-sales-forecasting. [Accessed: March 2025].

[2] Kaggle, “M5 Forecasting - Accuracy,” [Online]. Available: https://www.kaggle.com/competitions/m5-forecasting-accuracy. [Accessed: Dec 2024].

[3] Scikit-learn: Machine Learning in Python. [Online]. Available: <https://scikit-learn.org/>. [Accessed: May 2025].

[4] D. Silver et al., “Reinforcement Learning: A Survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[5] Ministry of Communications and Information Technology (MCIT) – Egypt, “DEPI Track: AI and Data Science Training Program,” 2024/2025. [Internal Materials Provided by DEPI].