Technical Report: Statistical Analysis of a Superstore Dataset

Purpose and Perspective

This report explains how I analyzed a retail transaction dataset (Superstore, Kaggle). My goal was to understand profit margins and the factors linked to losses. I did not aim to maximize prediction scores. I focused on models that I can explain and defend.

I followed a simple rule throughout the project: start simple, check assumptions, and add complexity only when the data demands it. This report shows the reasoning behind each choice, including what I chose not to do.

1) Data Overview and Cleaning

The dataset contains transaction-level records with order details, customer and product information, geographic data, shipping variables, sales, and profit.

After cleaning, the data includes:

- 9,994 rows
- 41 variables
- Transactions from 2014-01-01 to 2017-12-01

I removed the variable Row.ID at the start. It works only as an index and adds no analytical value. The following image presents a glimpse of the data

a) Identifier Structure and Redundancy

i) Order ID

The variable Order.ID stores several pieces of information in one string. I checked its structure and then split it into components.

One component repeated the order year, which already exists in Order.Date. I removed it to avoid duplicated information. This keeps the dataset easier to read and reason about.

ii) Customer and Product IDs

I applied the same process to customer and product identifiers. Some parts duplicated existing variables such as product category and sub-category.

I removed these redundant parts. This avoids giving extra weight to the same information in later models.

b) Time Variables

I split order and shipping dates into year, month, day, year–month, and quarter.

I ordered year–month values in time order to avoid errors in plots and models. I treated quarters as categories to study seasonality without fitting time-series models.

I did not model time dependence directly. The data has many transactions per day, and adding time structure without aggregation would add complexity with little benefit.

c) Variable Types

I converted all text variables to factors. I prefer to set types by hand so model inputs stay clear.

Two variables needed judgment:

- Discount: numeric in form, but it reflects pricing rules. I treated it as categorical.
- Postal.Code: treated as a label, not a number.

These choices match how the data is created in practice.

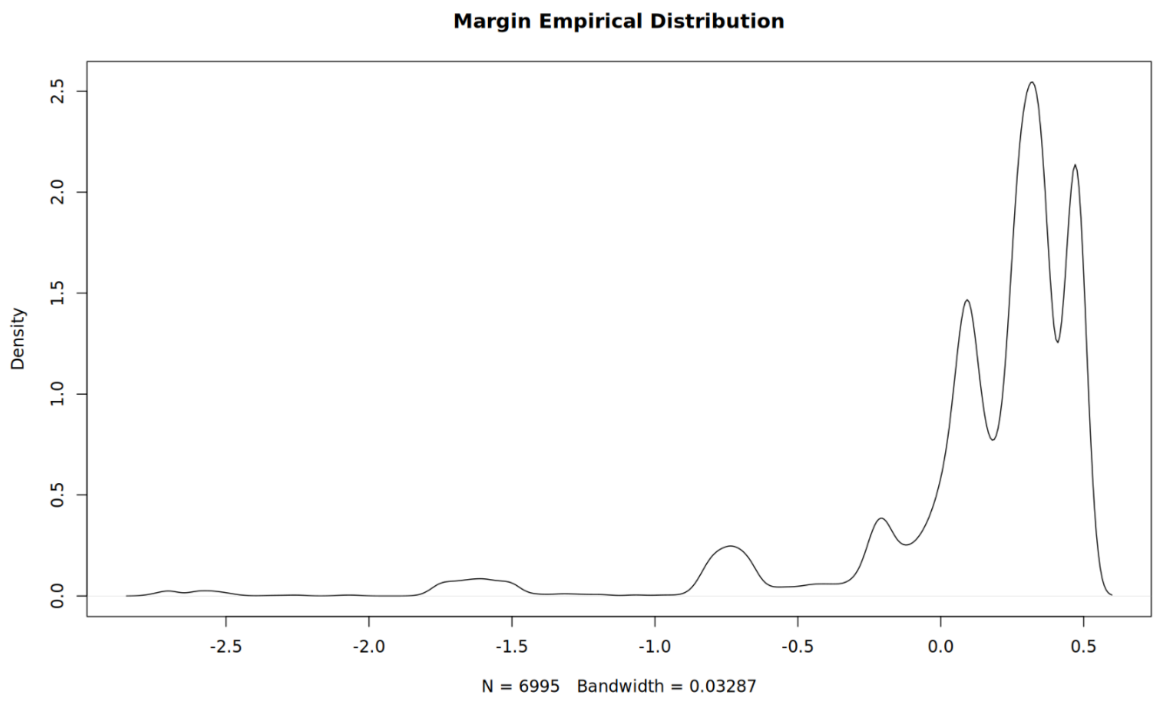
d) Target Variable

I defined the response variable as:

$$\text{Margin} = \frac{\text{Profit}}{\text{Sales}}$$

chose margin instead of profit to compare orders of different sizes and to attain better distribution qualities, like “lighter” tails.

The margin distribution shows skewness, heavy tails, and negative values. It also shows signs of more than one mode. These features guided later modeling choices.



e) Handling Rare Categories

i) Geographic Grouping

State-level data includes many states with few observations. I grouped states into larger geographic divisions similar to U.S. census regions.

This removes some detail but produces more stable and easier-to-interpret variables.

ii) Discount and Quantity Groups

I grouped discounts into 10% bands. I grouped quantities into ordered bins, with a separate group for orders of eight units or more.

The raw distributions are skewed, with sparse high values. Grouping reduces sensitivity to extreme cases and makes comparisons clearer.

f) Train, Validation, and Test Sets

I split the data at random into:

- 70% training
- 15% validation
- 15% test

This split lets me tune models without touching the final test set. After splitting, I checked that variable distributions stayed similar across all subsets.

2) Modelling

a) Baseline Linear Models

I started with lasso and ridge regression. I built the design matrix using model.matrix to control how categorical variables enter the model.

I use regularized linear models as a baseline because they are easy to inspect and handle collinearity well. At this stage, I wanted a reference point, not a final answer.

b) Linear and Semi-Parametric Models

i) Gaussian Linear Model

I fitted a linear model with Gaussian errors and selected variables using AIC. Residual plots showed heavy tails. This tells me the Gaussian assumption does not hold.

I kept this model as a diagnostic step.

ii) Interaction Model

I added selected pairwise interactions. These improved fit on the training data but gave mixed results on validation data. This suggests higher variance.

I chose not to keep a large interaction structure.

iii) Student-t Additive Model

I fitted a Student-t additive model using GAMLSS to handle heavy tails. Residuals improved, but the distribution still showed more than one mode.

This pointed to latent structure in the data.

c) Finite Mixture Regression

I fitted a mixture of two Gaussian linear regressions. This model assumes that transactions come from two hidden regimes, each with its own linear pattern.

The model combines predictions using posterior probabilities. Standardized residuals are close to normal, which shows a better fit than single-regime models.

The two components differ in:

- average margin
- margin spread

- chance of negative margin

This makes the mixture model useful for risk analysis.

d) Nonlinear Benchmarks

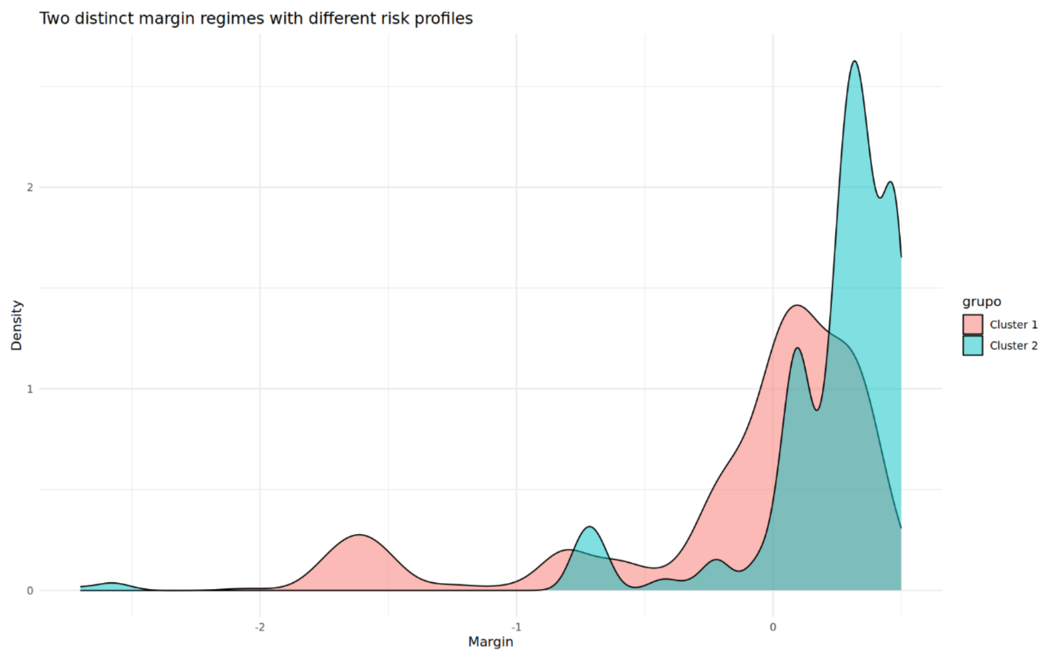
I also tested tree-based models: regression trees, random forests, and boosting methods.

Boosting models gave the best prediction scores. I did not select them as final models because they are harder to explain and offer limited insight into loss drivers.

e) Final Model Choice

I chose the two-component Gaussian mixture regression as the final model.

It balances accuracy, clarity, and structure. It captures different margin regimes while keeping a clear link between inputs and outcomes.



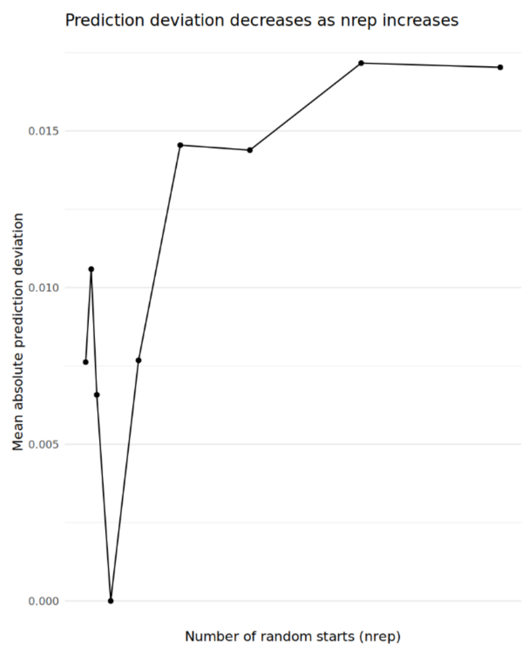
f) Stability and Uncertainty Checks

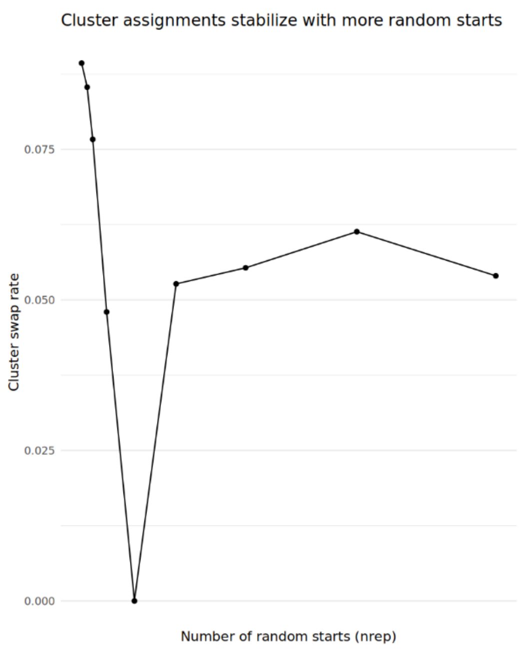
Mixture models can depend on starting values. I tested stability by varying the number of random starts.

I also ran a nonparametric bootstrap to study:

- prediction spread
- stability of regime assignment
- uncertainty in parameters

I removed failed fits from the analysis. These checks show that the results do not depend on chance initialization.





Accuracy improves with higher nrep, but the gains beyond 10 are marginal relative to the added computational cost

3) Technical Summary

This project reflects how I work with applied data:

- I clean data until its limits are clear.
- I test assumptions before trusting results.
- I treat simple models as tools, not defaults.
- I add complexity only when diagnostics support it.
- I value stable results over small score gains.
- The report focuses on reasoning and trade-offs rather than polished outcomes.