Mini-Project Report

---

1. Introduction

As part of my internship project, I was tasked with analyzing customer behavior in an e-commerce platform to understand and predict cart abandonment. This project aimed to extract behavioral patterns using Association Rule Mining and build classification models that could differentiate between sessions that led to purchases and those that resulted in cart abandonment.

Through this project, I experienced the full cycle of a project: from data understanding, cleaning, and feature engineering to model building, evaluation, and reflection.

---

2. Data Understanding & Merging

Files I Got:

- customer_table.csv
- device_table.csv
- product_table.csv
- date_table.csv
- fact_table.csv

Each file represented a part of the overall story:

- Customers: age, gender, city
- Devices: type (Mobile/Tablet), OS
- Products: category, price
- Dates: day, month, weekend info

- Fact table: session-level data (quantity, timestamps, etc.)

Steps I Took:

- Merged all five tables using their foreign keys
- Checked null values after merging
- Did a few sanity checks to ensure joins were correct

Final dataset was session-wise — each row represented a user session along with device, product, and date-related info.

---

3. Preprocessing & Feature Engineering

Goal: Build a target column where abandoned = 1 if the session ended in cart abandonment.

Features I Created:

- Temporal: day_of_week, month, is_weekend
- Price bins → price_quartile
- cart_total = price * quantity
- age_x_price (interaction term)
- high_price_flag → 1 if price above median
- weekend_high_price → combines weekend + high price

Rule-based Features (from association rules):

- rule_apparel = 1 if category was Apparel
- rule_android = 1 if OS was Android

- rule_price_vhigh = 1 if price in top 25%
- rule_weekend = 1 if weekend session
- rule_tablet = 1 if device type was Tablet

Encoding + Selection:

- Applied pd.get_dummies() on categorical vars
- Used SelectFromModel with a Random Forest to reduce dimensionality

> Honestly, feature engineering took the most time, but it was worth it — most models rely on these to make sense of the patterns.

---

4. Association Rule Mining

I converted each session into a **basket of items**:

Then I ran a simple 2-itemset scan to find rules like:

Rule	Confidence
Category_Apparel ⇒ Abandoned	52.5%
OS_Android ⇒ Abandoned	51.6%
Price_Very High ⇒ Abandoned	51.3%
Weekend ⇒ Abandoned	51.2%
Device_Tablet ⇒ Abandoned	50.4%

What I Got from This:

- Apparel and Electronics categories had slightly higher abandonment
- Android and Tablet users were more likely to abandon
- High prices and weekends correlated weakly with abandonment

These weren't strong rules, but gave me **explainable features** — which I used as inputs in the final model.

---

5. Modeling: Classification

Models I Tried:

- Logistic Regression
- Decision Tree
- Random Forest
- XGBoost
- **Stacking** (LR + DT + XGB → RandomForest)

All were evaluated using:

- **Accuracy**
- **ROC AUC**
- For XGBoost, I also checked precision/recall at threshold = 0.4

Model Results:

Model	Accuracy	ROC AUC
Logistic	49.5%	0.492
DecisionTree	48.8%	0.489
RandomForest	50.5%	0.511
XGBoost	49.3%	0.486
Stacking	**51.0%**	**0.508**

XGBoost (with threshold = 0.4):

- Precision: 51%
- Recall: 66%
- F1 Score: 57%

> TLDR: Models weren't great. Almost all predictions hovered around random guessing. Still, the stacking model gave a slight lift.

---

6. Feature Importance (Random Forest)

I checked which features had the most importance:

- Age → 15.5%
- Cart Total → 13.7%

- Price → 10.3%
- Quantity → 6.4%
- Gender_Male → 3.4%
- Some **rule-based flags** were also selected by the model (rule_android, rule_weekend, etc.)

Surprisingly, demographic features had higher weight than I thought. Product category and city were lower down.

---

7. Final Thoughts

What Went Well:

- Clean dataset merge
- Good mix of raw and engineered features
- Rule mining was fun and added some explainability
- Tried model stacking (good learning)

What Didn't:

- Accuracy stayed low (~50%) despite models
- No behavioral signals in the dataset
- No clickstream / time spent / browsing patterns

---

8. Conclusion

This project taught me that:

Great ML models need great data.
Algorithms can only learn what the data allows.

Despite low predictive power, this project helped me:

- Understand how to structure end-to-end ML pipelines
- Gain hands-on experience with rule mining and model stacking
- Appreciate the importance of feature design over model choice
- Learn to interpret limitations and communicate them effectively