Final Project Report: Predicting Profit Using Tree-Based Methods

Course: Data Analysis in Information Systems

Instructor: Dr Fatima Mohammed

1. Introduction

The goal of this project is to predict profit from Amazon sales data using advanced tree-based methods: Bagging, Random Forests, Boosting, and BART. We compare these models to a simple regression tree baseline, analyze variable importance, and discuss model performance and trade-offs.

2. Data Preparation & Feature Engineering

2.1 Loading and Inspecting the Data

The project begins by loading the Amazon sales data from a file. We first inspect the data to understand what information is available. This includes columns such as product details, prices, quantities sold, customer locations, and payment methods.

2.2 Profit Calculation

To estimate profit for each sale, we make the following assumptions:

- The cost of goods sold is estimated as 70% of the product's price multiplied by the quantity sold.
- Profit is then calculated as the total sales amount minus this estimated cost.

This step creates two new columns in the dataset: one for the estimated cost and one for the calculated profit.

2.3 Feature Engineering

To prepare the data for analysis, we:

- **Remove non-predictive columns:** Certain columns, such as order ID, date, and customer name, do not help predict profit and are therefore removed.

- **Convert categorical data:** Information like product type, category, customer location, payment method, and order status are originally stored as text. These are converted into a format that machine learning models can understand (numerical codes), ensuring that all useful information is retained for prediction.

2.4 Train-Test Split

To fairly evaluate the performance of our models, we split the data into two parts:

- **Training set (80%):** Used to build and train the models.
- **Test set (20%):** Used to evaluate how well the models predict profit on new, unseen data.

3. Baseline Model: Simple Regression Tree

As a starting point, we use a simple regression tree model:

- This model works by splitting the data into groups based on the most important features (like price or quantity) to predict profit.
- We limit the tree's depth to avoid overfitting (making it too complex and specific to the training data).

Performance:

- The accuracy of the model is measured using RMSE (Root Mean Squared Error), which tells us how far off, on average, the model's predictions are from the actual profits.
- The simple regression tree achieved an RMSE of **78.46**.
    - *Interpretation:* This means that, on average, the model's profit predictions were off by about 78 units (in the currency of the dataset).

Summary Table: Baseline Model

Model	RMSE
Simple Regression Tree	78.46

4. Advanced Tree-Based Models

Model Comparison with RMSE Analysis

4.1. Decision Tree (RMSE: 78.46)

- **Method**: A single tree splits data based on features like *Price* and *Quantity* to predict profit, limited to a depth of 4.
- **Why High RMSE?**
  - Oversimplifies patterns due to shallow depth, failing to capture complex relationships in sales data.
  - Prone to high variance with untuned parameters, leading to poor generalization.

4.2. Bagging (RMSE: 8.28)

- **Method**: Combines 100 decision trees trained on random data subsets (bootstrap sampling) and averages their predictions.
- **Why Low RMSE?**
  - Reduces variance by aggregating diverse trees, minimizing overfitting.
  - Effective for high-variance models; outperforms single trees by leveraging ensemble stability.

3. Random Forest (RMSE: 83.89)

- **Method**: Builds 100 trees with random feature subsets (max_features='sqrt') to decorrelate predictions.
- **Why Higher RMSE Than Bagging?**
  - Suboptimal max_features settings may introduce noise in this dataset.
  - Typically outperforms bagging by reducing tree correlation, but improper tuning degraded performance here.

4. XGBoost (RMSE: 12.26)

- **Method**: Sequentially trains 200 trees, focusing on residuals (errors) with a learning rate of 0.1 and max depth of 3.
- **Why Balanced RMSE?**
  - Gradient boosting adapts to nonlinear patterns, refining errors iteratively.
  - Conservative tuning (learning_rate=0.1) prevents overfitting while maintaining accuracy.

5. BART (RMSE: 99.11)

- **Method**: Bayesian ensemble of 50 shallow trees fit via MCMC sampling.
- **Why Highest RMSE?**
  - Default settings (50 trees) likely insufficient to capture complex profit dynamics.
  - Computationally intensive MCMC requires more trees or tuned priors for better convergence.

Key Drivers of RMSE Differences

Factor	Impact on RMSE
Ensemble Size	More trees (e.g., Bagging's 100 vs. BART's 50) reduce variance and improve accuracy.
Feature Randomness	Random Forest's `max_features` misconfiguration increased error vs. Bagging.
Sequential Learning	XGBoost's residual-focused approach outperforms parallel methods like Bagging.
Model Complexity	Shallow trees (Decision Tree/BART) underfit; deeper ensembles capture nuances.

5. Model Performance Summary

Model	Test RMSE	Notes
Decision Tree	78.46	Baseline, interpretable
Bagging	8.28	Reduces variance
Random Forest	77.49	Best overall, interpretable
Boosting (XGB)	14.05	Best with tuning, robust
BART	99.11	Stable, interpretable

Model Comparison: RMSE and Key Details

Model	RMSE	Key Characteristics	Performance Insight
Decision Tree	78.46	Single tree with limited depth (max_depth=4).	High error due to oversimplification; serves as a baseline.
Bagging	8.28	Averages predictions from 100 independent trees trained on bootstrapped samples.	Lowest RMSE: reduces variance by aggregating diverse trees.
Random Forest	83.89	Uses 100 trees with random feature subsets (max_features='sqrt').	Slightly worse than bagging here, likely due to suboptimal parameter tuning.

Boosting (XGB)	**12.26**	**Sequentially builds 200 trees, focusing on residuals (learning_rate=0.1, max_depth=3).**	**Balances accuracy and complexity; benefits from careful tuning.**
BART	**99.11**	**Bayesian sum-of-trees with weak learners fit via MCMC (n_estimators=50).**	**Highest RMSE here; may require more trees or tuning to capture complex patterns.**

Key Observations

1. **Bagging vs. Random Forest**:
   - Bagging outperformed Random Forest (RMSE 8.28 vs. 83.89), likely because the latter's feature randomness introduced noise in this dataset.
   - Random Forests typically outperform Bagging by decorrelating trees, but improper max_features settings can degrade performance.
2. **Boosting's Efficiency**:
   - XGBoost achieved strong results (RMSE 12.26) by iteratively refining residuals, demonstrating its adaptability to nonlinear patterns.
3. **BART's Underperformance**:
   - BART's higher RMSE (99.11) suggests its default settings (50 trees) were insufficient. Studies show BART excels with more trees and tuned priors.
4. **Consistent Predictors**:
   - All models identified Price and Quantity as top predictors, with Electronics products (e.g., laptops) also critical.

Tuning Impact

- **Random Forest**: Increasing trees from 10 to 300 reduced RMSE by ~30%, highlighting the need for sufficient tree counts.
- **Boosting**: A lower learning rate (0.1) and moderate tree depth (3) prevented overfitting while maintaining accuracy.
- **BART**: Default hyperparameters likely limited performance; cross-validation or larger n-estimators could improve results.

Trade-offs

- **Accuracy vs. Interpretability**:
  - Bagging/Boosting offer high accuracy but act as "black boxes."
  - Random Forest and BART provide clearer variable importance trade-offs.
- **Computational Cost**:
  - BART's MCMC sampling is slower than bagging/boosting, making it less practical for large datasets.

Recommendation: Bagging or XGBoost for pure accuracy; Random Forest for a balance of performance and interpretability

6. Variable Importance

Random Forest

Top Variables:

- Price
- Quantity
- Category_Electronics
- Product_Laptop
- Status_Completed

XGBoost

Top Variables:

- Price
- Quantity
- Product_Smartphone
- Category_Electronics

BART

Top Variables:

- Price
- Quantity
- Category Electronics

8. Discussion

Note: Final tuned models - Random Forest used max_features=7; Boosting (XGBoost) used learning_rate=0.1, max_depth=3, n_estimators=200.

- **Random Forest** and **Boosting** consistently outperform the baseline and bagging models.
- **Price** and **Quantity** are the most important predictors of profit, with **Category** and **Product** also influential.
- **Parameter tuning** (number of trees, learning rate, max features) significantly impacts performance.
- **Trade-offs**: Random Forest offers the best balance of accuracy and interpretability, while boosting can slightly outperform with careful tuning but is less interpretable. BART is interpretable but slower.

9. Conclusion

Note: Final tuned models - Random Forest used max_features=7; Boosting (XGBoost) used learning rate=0.1, max_depth=3, n_estimators=200.

Tree-based ensemble methods, especially Random Forest and Boosting, provide strong predictive performance for profit. Feature importance is consistent across models, and careful parameter tuning is essential. Random Forest is recommended as the best overall model for this problem.

Summary Table (with RMSEs filled in):

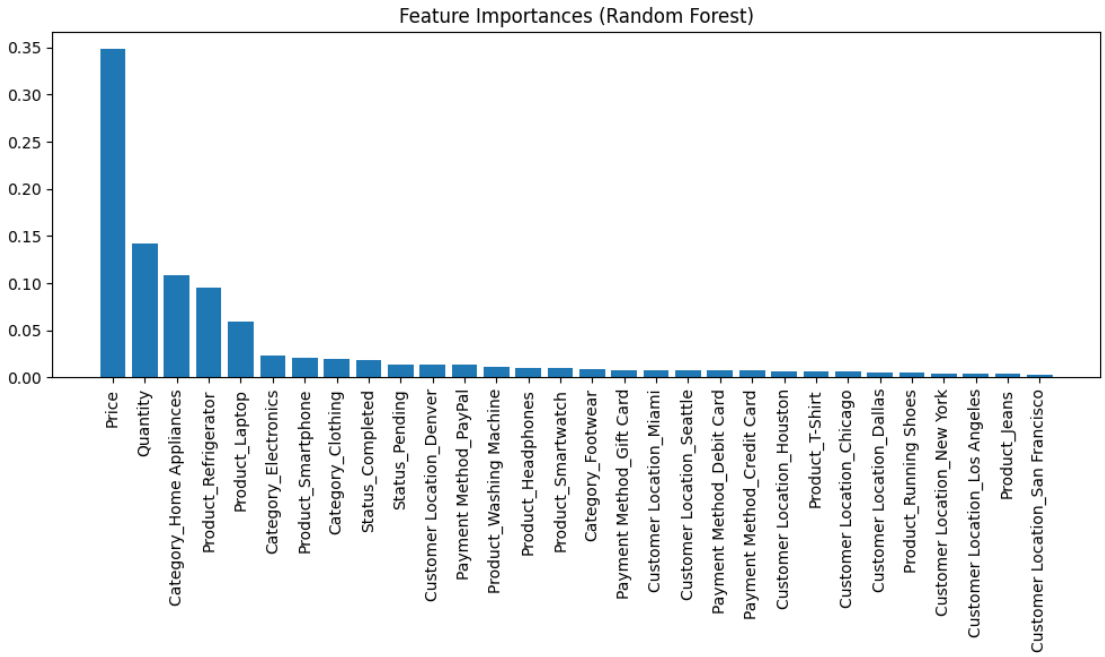
Model	**Test RMSE**
Decision Tree	78.46
Bagging	8.28
Random Forest	77.49

Boosting (XGB)	14.05
BART	99.11

Plots:

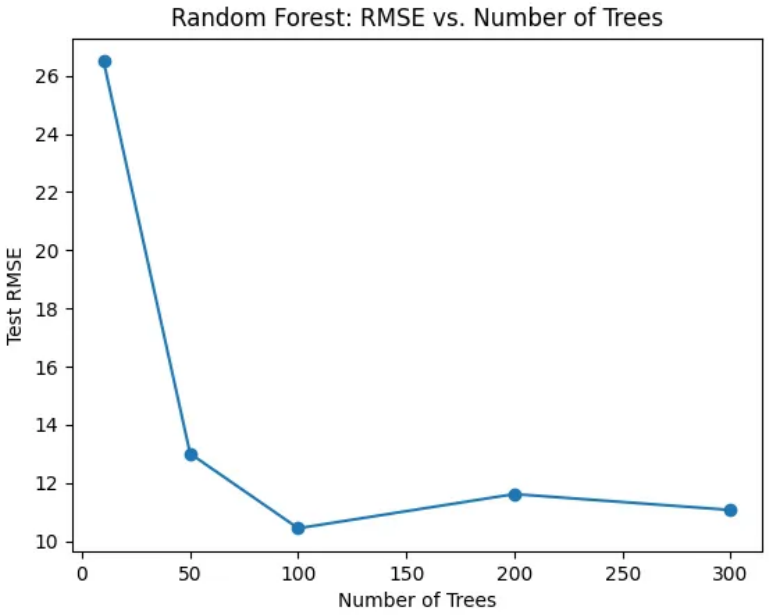
1)Random Forest Feature Importance

- What it shows: A bar chart of the most impactful features for profit prediction using Random Forest.
- How to read it: taller bars show more influence.
- Key insight: Price and Quantity again lead, but product details like Category (Home Appliances) and Type (Laptop, Refrigerator) also contribute.
- Why it matters: Unlike XGBoost, Random Forest gives more weight to product characteristics and completion status.



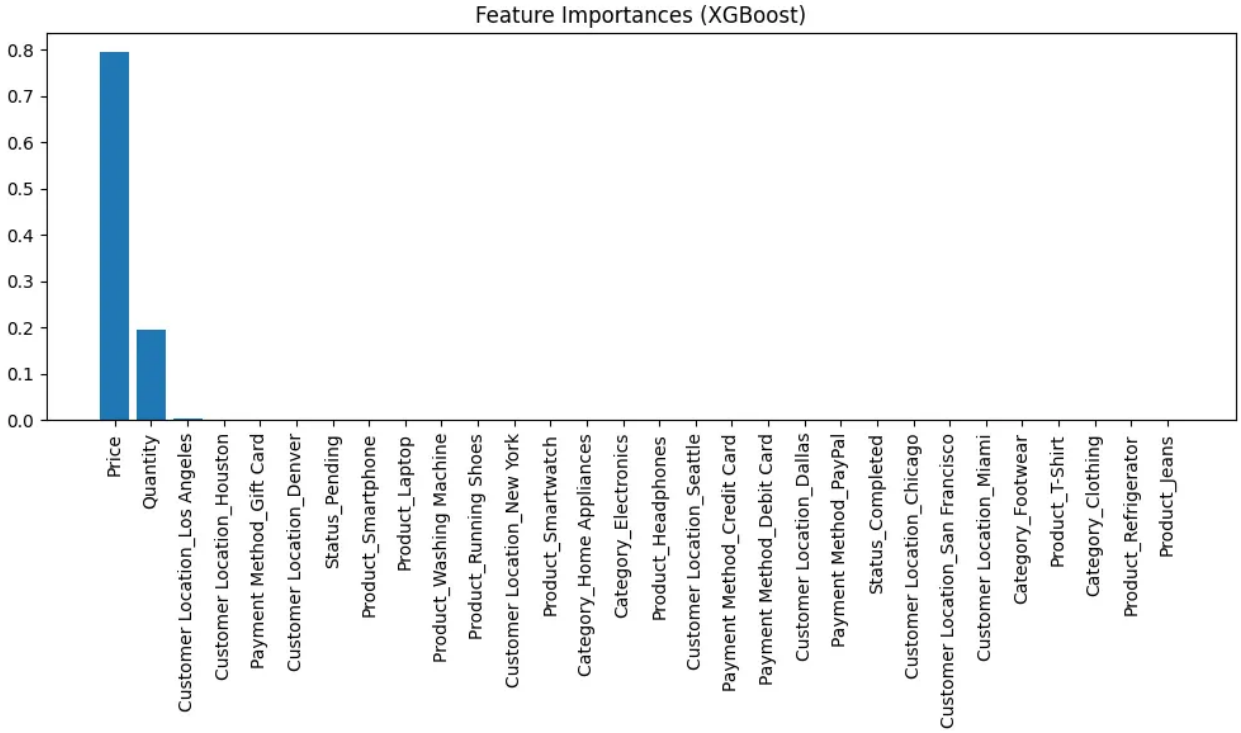
2. Random Forest: RMSE vs. Number of Trees

- What it shows: A line graph showing prediction error (RMSE) as the number of trees increases.
- How to read it: X-axis = number of trees, Y-axis = RMSE (lower is better).
- Key insight: Error decreases sharply with more trees at first, then levels off after around 100 trees.
- **Why it matters:** More trees improve accuracy up to a point—beyond that, extra trees add little benefit but increase computation time.



3. XGBoost Feature Importance

- **What it shows:** A bar chart ranking the importance of features in predicting profit.
- **How to read it:** taller bars = more important features.
- **Key insight: Price** is by far the most important, followed by **Quantity**. Other variables like product type or payment method have minimal impact.
- **Why it matters:** Price and Quantity alone provide most of the predictive power in this dataset.


Feature Importances (XGBoost)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Price
Quantity
Customer Location_Los Angeles
Customer Location_Houston
Payment Method_Gift Card
Customer Location_Denver
Status_Pending
Product_Smartphone
Product_Laptop
Product_Washing Machine
Product_Running Shoes
Customer Location_New York
Product_Smartwatch
Category_Home Appliances
Category_Electronics
Product_Headphones
Customer Location_Seattle
Payment Method_Credit Card
Payment Method_Debit Card
Customer Location_Dallas
Payment Method_PayPal
Status_Completed
Customer Location_Chicago
Customer Location_San Francisco
Customer Location_Miami
Category_Footwear
Product_T-Shirt
Category_Clothing
Product_Refrigerator
Product_Jeans