Machine Learning Practical Assignment

Restaurant Product Recommendation System

Fall 2025

Tanasa Ionut-Eduard Cojocarescu Rebeca Daria

January 18, 2025

Abstract

This report presents a comprehensive machine learning solution for restaurant product recommendations, focusing on sauce prediction and product ranking for upsell opportunities. We implement and evaluate multiple classification algorithms including custom Logistic Regression variants (Gradient Descent, Newton's Method, Mini-batch GD), Naive Bayes, k-Nearest Neighbors, Decision Trees (ID3), and AdaBoost. Our experiments demonstrate that while popularity-based baselines remain competitive, learned models can capture meaningful purchase patterns, particularly at moderate recommendation depths (K=3). We provide detailed analysis of feature importance, hyperparameter sensitivity, and algorithmic trade-offs across three distinct prediction tasks.

Contents

1 Introduction

1.1 Problem Description

This project addresses the challenge of product recommendation in a restaurant setting, using transaction data from fiscal receipts. The primary goal is to predict customer purchasing behavior and recommend relevant products (particularly sauces and complementary items) based on cart composition and temporal context.

The project encompasses three main tasks:

1. **Task 2.1:** Binary classification to predict whether a customer who purchases "Crazy Schnitzel" will also purchase "Crazy Sauce"
2. **Task 2.2:** Multi-sauce recommendation system with individual models for each of 8 standalone sauces
3. **Task 3:** Product ranking for upsell recommendations using multiple machine learning algorithms

1.2 Dataset Overview

The dataset consists of transaction records from a restaurant, containing the following key statistics:

Table 1: Dataset Statistics

Attribute	**Value**
Time Period	September 5, 2025 – December 3, 2025
Total Receipts	∼7,869
Total Transaction Lines	28,039
Unique Products	59
Average Cart Size	3.56 products/receipt
Maximum Cart Size	26 products

The standalone sauces analyzed in this project are:

- Crazy Sauce, Cheddar Sauce, Extra Cheddar Sauce, Garlic Sauce
- Tomato Sauce, Blueberry Sauce, Spicy Sauce, Pink Sauce

1.3 Key Columns

- `id_bon`: Receipt identifier (groups products into transactions)
- `data_bon`: Timestamp of the transaction
- `retail_product_name`: Product name
- `SalePriceWithVAT`: Price including VAT per line item

2 Theoretical Background

2.1 Logistic Regression

Logistic Regression is a fundamental classification algorithm that models the probability of a binary outcome using the logistic (sigmoid) function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

For a feature vector $x \in \mathbb{R}^n$ and weights $\theta \in \mathbb{R}^n$, the predicted probability is:

$$P(y = 1 \mid x) = \sigma(x^T \theta) = \frac{1}{1 + e^{-x^T \theta}} \tag{2}$$

The model is trained by minimizing the binary cross-entropy loss with optional L2 regularization:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] + \frac{\lambda}{2} \|\theta\|^2 \tag{3}$$

2.2 Naive Bayes

The Naive Bayes classifier applies Bayes' theorem with the "naive" assumption of conditional independence between features:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{4}$$

For classification, we select the class with maximum posterior probability:

$$\hat{y} = \arg\max_c P(y = c) \prod_{i=1}^{n} P(x_i \mid y = c) \tag{5}$$

2.3 k-Nearest Neighbors

k-NN is a non-parametric algorithm that classifies instances based on the majority vote of their k nearest neighbors in feature space. For distance-weighted voting:

$$\hat{y} = \arg\max_c \sum_{i \in N_k(x)} w_i \cdot \mathbb{1}[y_i = c], \quad w_i = \frac{1}{d(x, x_i) + \epsilon} \tag{6}$$

where $d(x, x_i)$ is the Euclidean distance and ϵ prevents division by zero.

2.4 Decision Trees (ID3)

The ID3 algorithm builds decision trees by recursively selecting the feature that maximizes information gain:

$$\text{IG}(S, A) = H(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} H(S_v) \tag{7}$$

where entropy $H(S) = -\sum_c p_c \log_2 p_c$ measures the impurity of set S.

2.5 AdaBoost

AdaBoost (Adaptive Boosting) creates an ensemble of weak learners, iteratively focusing on misclassified samples:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x) \right) \tag{8}$$

where $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$ weights each weak learner h_t based on its error rate ϵ_t.

2.6 Evaluation Metrics

We employ several metrics to evaluate model performance:

Classification Metrics:

- **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision:** $\frac{TP}{TP+FP}$ (proportion of positive predictions that are correct)
- **Recall:** $\frac{TP}{TP+FN}$ (proportion of actual positives correctly identified)
- **F1 Score:** $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
- **ROC-AUC:** Area under the Receiver Operating Characteristic curve

Ranking Metrics:

- **Hit@K:** Proportion of queries where the relevant item appears in top-K results
- **Precision@K:** $\frac{\text{Relevant items in top-K}}{K}$
- **MRR (Mean Reciprocal Rank):** $\frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$
- **NDCG@K:** Normalized Discounted Cumulative Gain, accounting for position-based relevance

3 Data Preprocessing

3.1 Feature Engineering Pipeline

We developed a comprehensive feature engineering pipeline to transform raw transaction data into machine learning-ready features. The preprocessing steps include:

3.1.1 Temporal Features

From the `data_bon` timestamp, we extracted:

- `hour`: Hour of the day (0–23)
- `day_of_week`: Day of the week (1–7, where 1 = Monday)
- `is_weekend`: Binary indicator (1 if Saturday or Sunday)
- `date`: Date component for temporal splitting

3.1.2 Receipt-Level Aggregations

For each receipt (`id_bon`), we computed:

- `cart_size`: Total number of products in the cart
- `distinct_products`: Number of unique products
- `total_value`: Sum of `SalePriceWithVAT`

3.1.3 Product Indicator Features

We created binary features for product presence:

$$\text{has\_product}_p = \begin{cases} 1 & \text{if product } p \text{ is in the receipt} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Important: To avoid data leakage, the target sauce is always excluded from the feature set when predicting that sauce.

3.2 Train-Test Split Strategy

We employed a **temporal split** strategy to simulate real-world deployment conditions:

- **Training set (80%):** All receipts before a temporal cutoff date
- **Test set (20%):** All receipts after the cutoff date

This approach ensures that the model is evaluated on future transactions it has not seen during training, providing a realistic assessment of generalization performance.

For Task 2.1 (Crazy Sauce prediction), we first filtered the dataset to include only receipts containing "Crazy Schnitzel", resulting in:

- Training receipts: ∼5,140
- Test receipts: ∼1,291

4 Task 2.1: Crazy Sauce Prediction

4.1 Problem Formulation

Given a receipt containing "Crazy Schnitzel", predict whether the customer will also purchase "Crazy Sauce".

Target variable:

$$y = \begin{cases} 1 & \text{if Crazy Sauce is in the receipt} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Class distribution:

- Without Crazy Sauce: 56.2%
- With Crazy Sauce: 43.8%

4.2 Custom Logistic Regression Implementations

We implemented three variants of logistic regression from scratch:

4.2.1 Gradient Descent (GD)

The standard gradient descent optimization:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla J(\theta^{(t)}) \tag{11}$$

where the gradient of the cross-entropy loss with L2 regularization is:

$$\nabla J(\theta) = \frac{1}{m} X^T (\sigma(X\theta) - y) + \lambda\theta \tag{12}$$

Hyperparameters:

- Learning rate $\alpha = 0.1$
- Maximum iterations: 2000
- L2 regularization $\lambda = 0.01$

4.2.2 Newton's Method

Second-order optimization using the Hessian matrix:

$$\theta^{(t+1)} = \theta^{(t)} - H^{-1}\nabla J(\theta^{(t)}) \tag{13}$$

where the Hessian is:

$$H = \frac{1}{m}X^T SX + \lambda I \tag{14}$$

with $S = \text{diag}(\sigma(X\theta)(1 - \sigma(X\theta)))$.

Hyperparameters:

- Maximum iterations: 100
- L2 regularization $\lambda = 0.01$

4.2.3 Mini-Batch Gradient Descent

Stochastic optimization with mini-batches:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha\nabla J_{\text{batch}}(\theta^{(t)}) \tag{15}$$

Hyperparameters:

- Learning rate $\alpha = 0.1$
- Batch size: 32
- Maximum epochs: 500
- L2 regularization $\lambda = 0.01$

4.3 Model Comparison Results

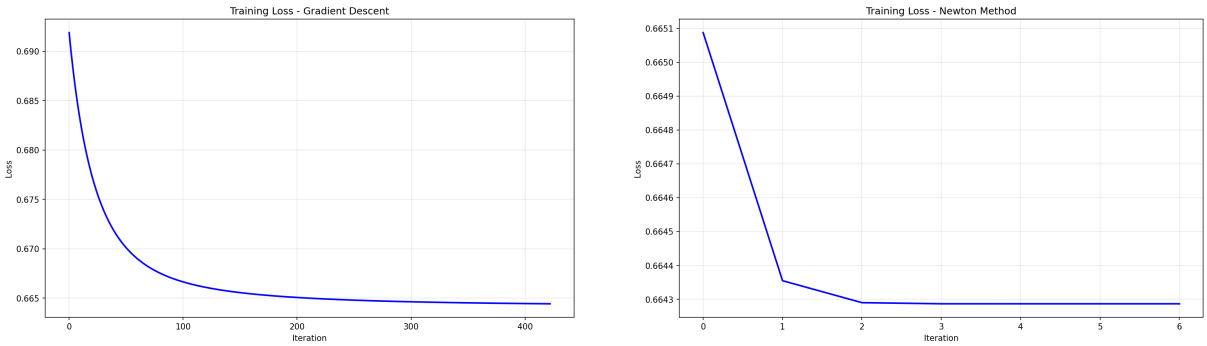
Table 2: Task 2.1: Model Performance Comparison

Model	Accuracy	Precision	Recall	F1	ROC-AUC
Custom LR (GD)	54.78%	0.602	0.575	0.588	0.549
Custom LR (Newton)	**55.62%**	**0.611**	0.580	0.595	**0.548**
Custom LR (Mini-batch)	54.78%	0.598	**0.595**	**0.596**	0.536
Sklearn LR	55.06%	0.606	0.570	0.588	0.544
Majority Baseline	56.18%	0.562	1.000	0.719	0.500

Key observations:

- All models perform similarly, achieving approximately 55% accuracy
- The majority class baseline achieves 56.18% accuracy, indicating that the prediction task is inherently challenging
- Custom implementations achieve competitive performance with sklearn, validating our algorithm implementations
- Newton's Method achieves the best performance among custom implementations

4.4 Convergence Analysis



(a) Gradient Descent convergence (b) Newton's Method convergence

Figure 1: Training loss curves for custom Logistic Regression implementations

Newton's Method demonstrates faster convergence (fewer iterations) compared to Gradient Descent, as expected from second-order optimization.

4.5 ROC Curve Comparison

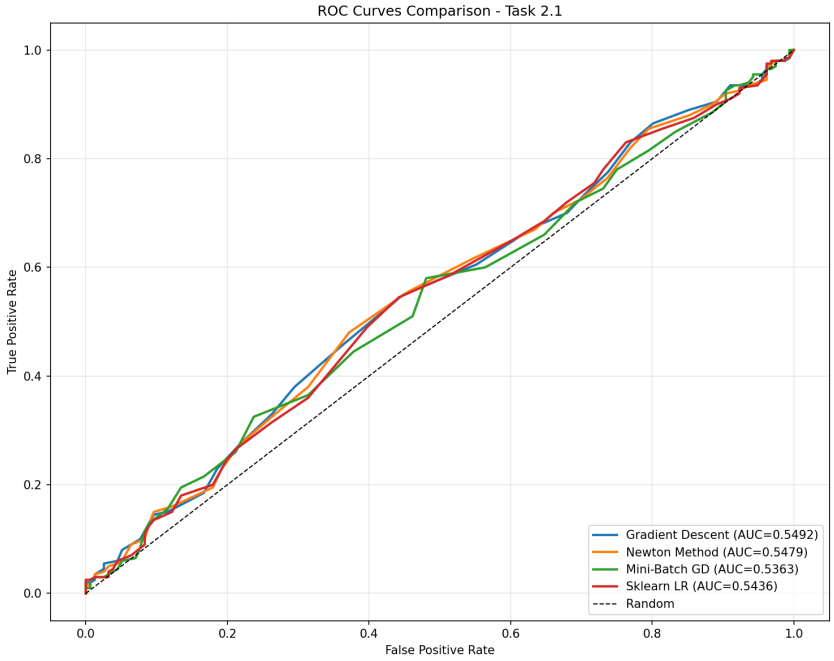


Figure 2: ROC curves comparison for Task 2.1 models

4.6 Feature Importance Analysis

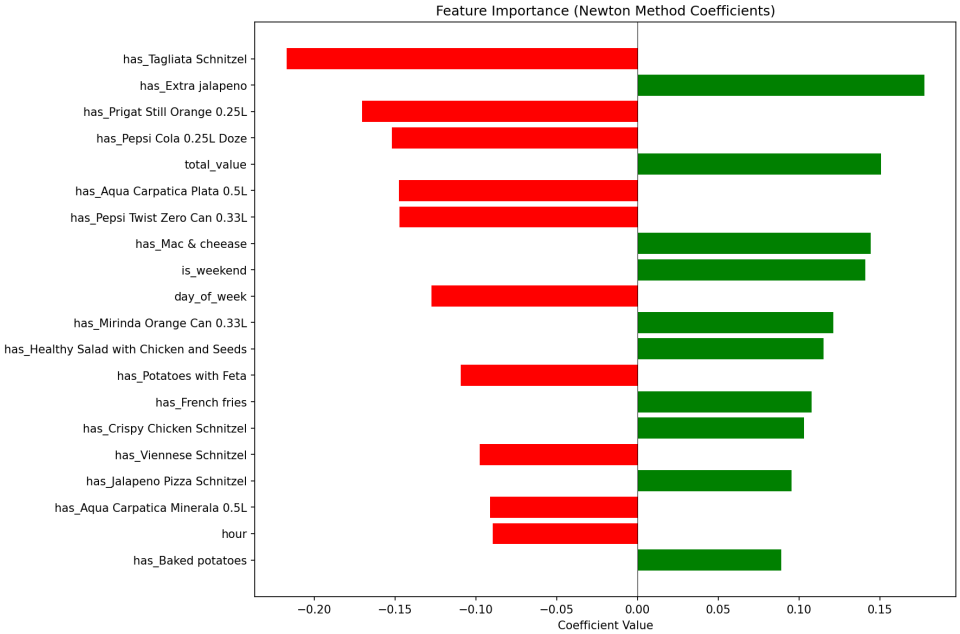


Figure 3: Top 20 feature coefficients (Newton's Method)

Interpretation of coefficients:

- **Top positive predictors** (increase probability of Crazy Sauce):
  - Extra jalapeño ($\beta = 0.177$)
  - Total cart value ($\beta = 0.151$)
  - Mac & Cheese ($\beta = 0.144$)
  - Weekend purchases ($\beta = 0.141$)
  - French fries ($\beta = 0.108$)
- **Top negative predictors** (decrease probability of Crazy Sauce):
  - Tagliata Schnitzel ($\beta = -0.217$)
  - Prigat Still Orange ($\beta = -0.171$)
  - Pepsi Cola 0.25L ($\beta = -0.152$)
  - Aqua Carpatica Plata ($\beta = -0.148$)
  - Day of week ($\beta = -0.127$)

4.7 Confusion Matrix

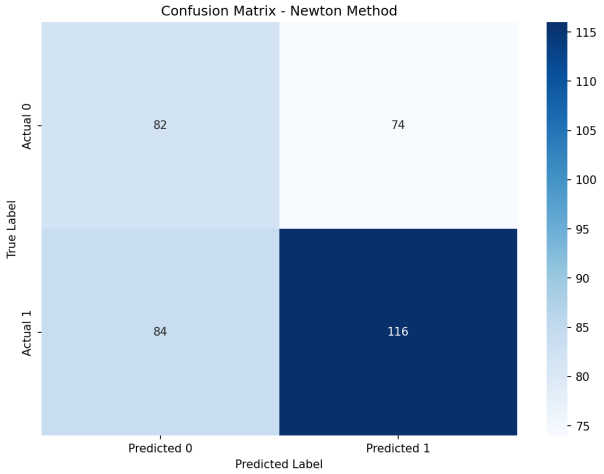


Figure 4: Confusion matrix for Newton's Method

5 Task 2.2: Multi-Sauce Recommendation

5.1 Problem Formulation

Build a personalized sauce recommendation system by training one Logistic Regression model per sauce:

$$y_s = \begin{cases} 1 & \text{if sauce } s \text{ is in the receipt} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

5.2 Architecture

The `SauceRecommender` class maintains a dictionary of models:

```
models = {
    "Crazy␣Sauce": LogisticRegressionNewton(),
    "Cheddar␣Sauce": LogisticRegressionNewton(),
    ...  # One model per sauce
}
```

Each model uses:

- Newton's Method optimization
- L2 regularization ($\lambda = 0.01$)
- Feature standardization (z-score normalization)

5.3 Individual Model Performance

Table 3: Task 2.2: Per-Sauce Model Performance

Sauce	Accuracy	ROC-AUC	F1	Positive Rate	N Samples
Crazy Sauce	78.30%	0.486	0.0	20.62%	6,431
Cheddar Sauce	87.56%	0.502	0.0	12.94%	6,431
Extra Cheddar Sauce	99.84%	0.498	0.0	0.31%	6,431
Garlic Sauce	90.05%	0.452	0.0	9.27%	6,431
Tomato Sauce	97.20%	0.592	0.0	2.66%	6,431
Blueberry Sauce	90.90%	0.486	0.0	9.02%	6,431
Spicy Sauce	93.93%	0.604	0.0	4.87%	6,431
Pink Sauce	97.82%	0.516	0.0	1.80%	6,431

Note: High accuracy for rare sauces (e.g., Extra Cheddar Sauce at 99.8%) reflects class imbalance – the model learns to predict the majority class. The F1 score of 0.0 indicates that the models predict almost exclusively the negative class.

5.4 Sauce Popularity Analysis

Understanding sauce popularity is crucial for establishing a meaningful baseline:

Table 4: Sauce Popularity in Training Data

Sauce	Receipts	Popularity
Crazy Sauce	1,326	20.62%
Cheddar Sauce	832	12.94%
Garlic Sauce	596	9.27%
Blueberry Sauce	580	9.02%
Spicy Sauce	313	4.87%
Tomato Sauce	171	2.66%
Pink Sauce	116	1.80%
Extra Cheddar Sauce	20	0.31%

The high class imbalance (Crazy Sauce at 20.6% vs Extra Cheddar Sauce at 0.3%) presents a significant challenge for classification models.

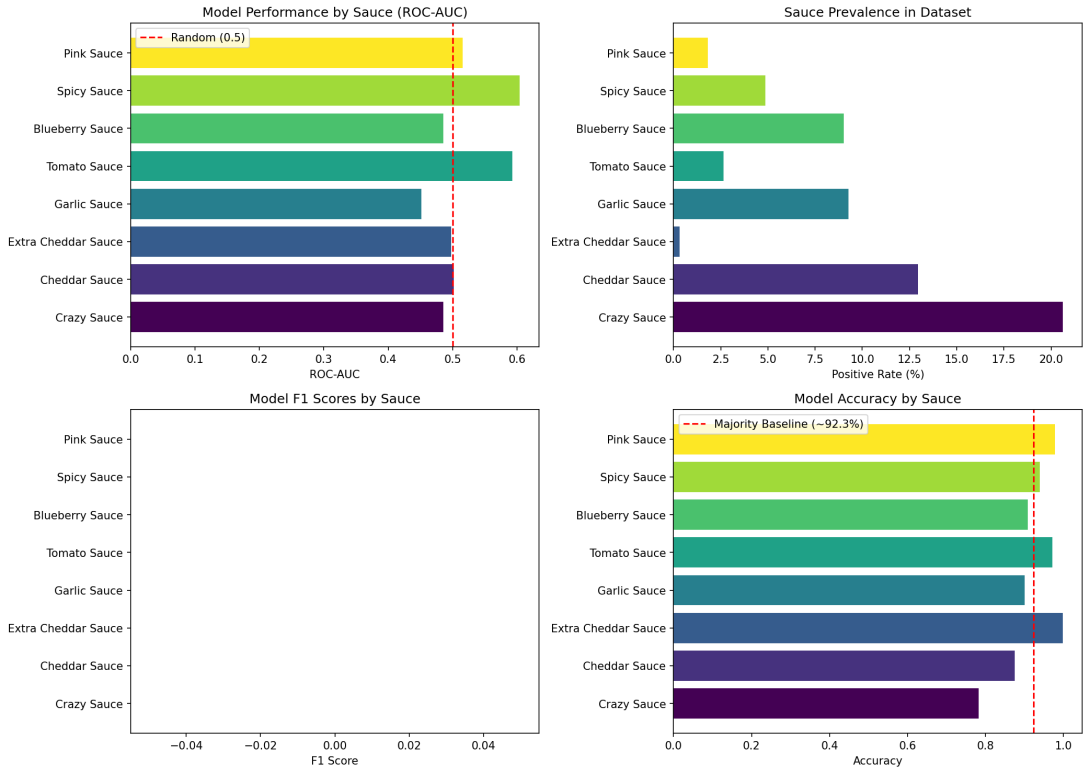


Figure 5: Per-sauce model performance metrics

5.5 Pseudo-Recommendation Mechanism

For a given cart (without sauces), we compute:

$$P(s \mid \text{cart}) = \sigma(x^T \theta_s) \tag{17}$$

for each sauce s, then recommend the Top-K sauces with highest probability.

5.6 Recommendation Evaluation

We compare the Logistic Regression-based recommender against a popularity baseline (recommending sauces by global frequency).

Table 5: Task 2.2: Recommendation Performance (LR vs. Popularity Baseline)

K	Model	Hit Rate	Precision	Recall	MRR	NDCG
1	LR	35.96%	0.360	0.313	0.571	0.360
	Baseline	35.83%	0.358	0.310	0.572	0.358
3	LR	**75.13%**	**0.276**	0.711	0.571	**0.561**
	Baseline	73.66%	0.271	0.698	0.572	0.554
5	LR	93.45%	0.214	0.918	0.571	0.650
	Baseline	94.25%	0.216	0.925	0.572	0.652

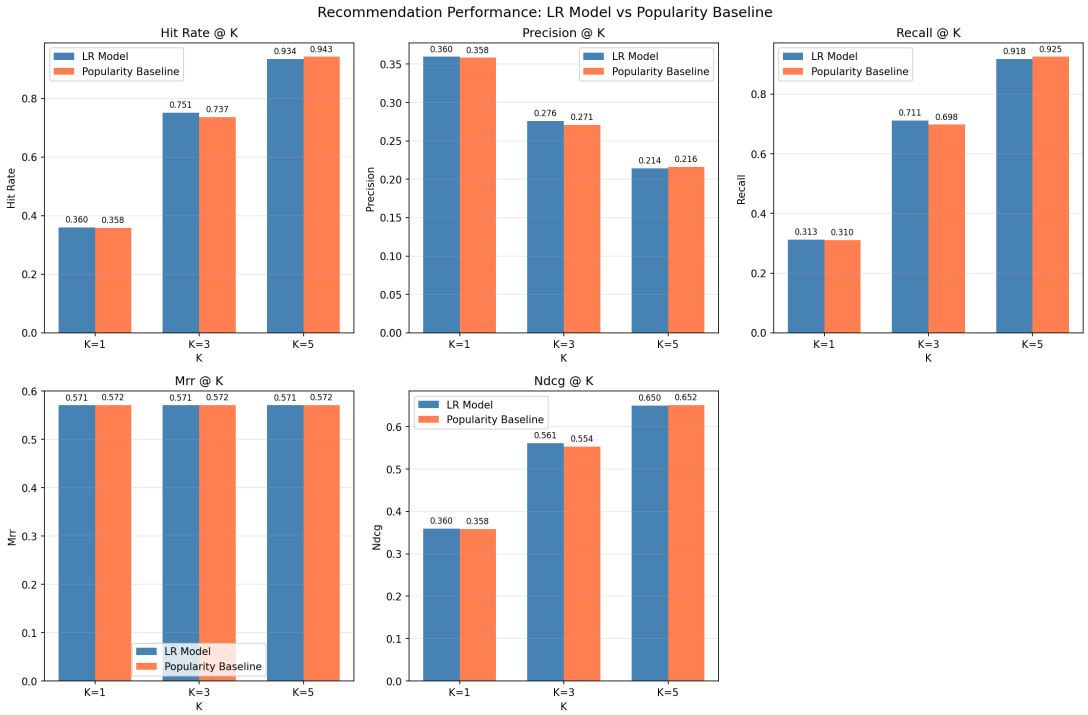


Figure 6: Recommendation metrics: LR vs. Popularity Baseline

Key findings:

- LR outperforms the popularity baseline at K=3 (+1.47% Hit Rate)
- At K=5, the popularity baseline slightly outperforms LR
- This suggests that learned models capture meaningful patterns but popularity remains a strong baseline

5.7 Coefficient Heatmap

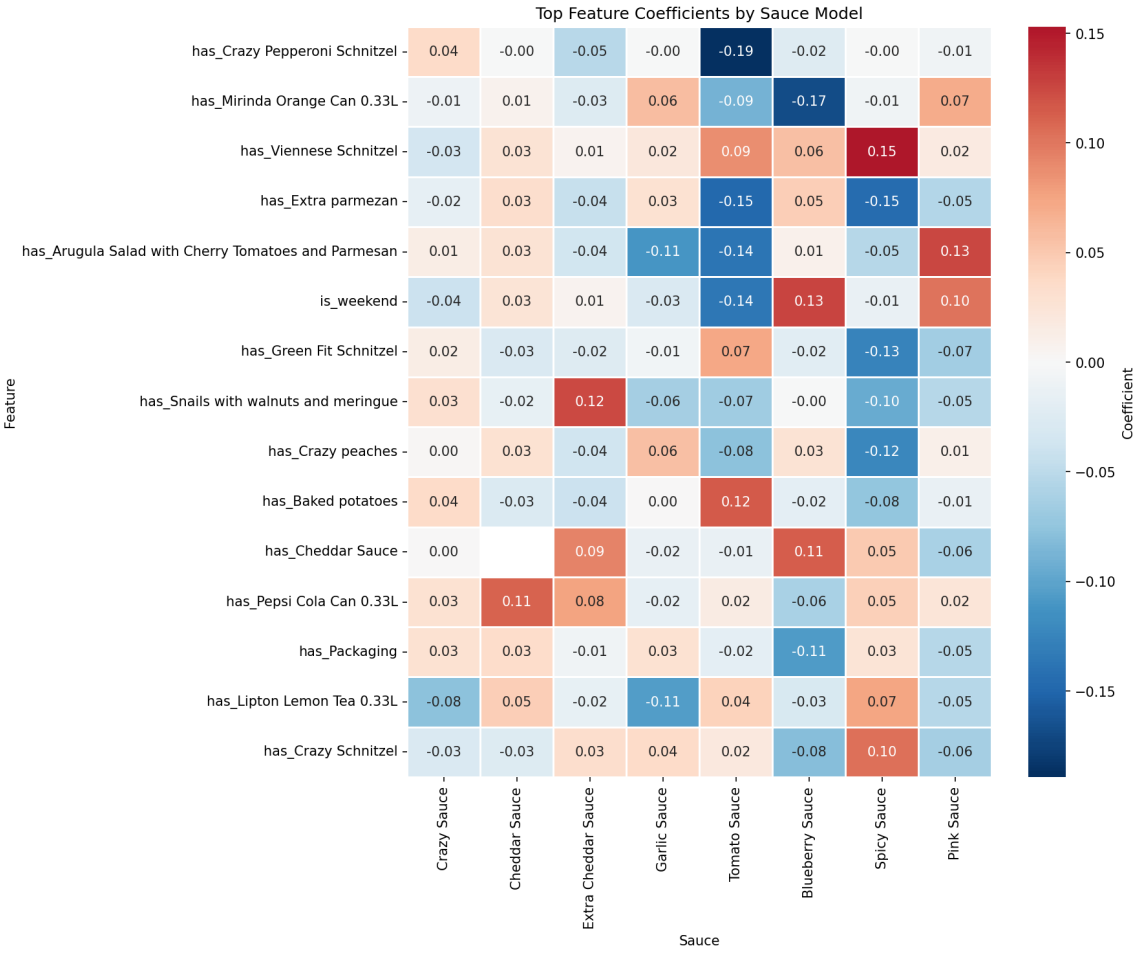


Figure 7: Feature coefficient heatmap across all sauce models

The heatmap reveals which products are predictive of each sauce, enabling interpretable recommendations.

5.8 Top Predictors per Sauce

Analysis of model coefficients reveals meaningful patterns:

Table 6: Top Positive Predictors for Selected Sauces

Sauce	**Top Predictor**	**Coefficient**
Crazy Sauce	Crazy Fries with Parmesan	+0.063
Cheddar Sauce	Pepsi Cola Can 0.33L	+0.111
Garlic Sauce	Red Fit Schnitzel	+0.100
Tomato Sauce	Baked potatoes	+0.116
Blueberry Sauce	is_weekend	+0.128
Spicy Sauce	Viennese Schnitzel	+0.153
Pink Sauce	Arugula Salad	+0.126

5.9 Example Recommendations

To illustrate the recommendation system in action, here are sample predictions:

Table 7: Example Recommendation Outputs

Cart Contents	**Actual Sauce**	**Top-3 LR Recommendations**
Crazy Schnitzel, Pepsi Cola, Baked potatoes	Cheddar Sauce	Crazy Sauce (0.23), Cheddar Sauce (0.15), Garlic Sauce (0.09)
Healthy Green Fit Schnitzel, Crazy peaches	None	Crazy Sauce (0.21), Cheddar Sauce (0.20), Blueberry Sauce (0.18)
Margherita Pizza Schnitzel	None	Crazy Sauce (0.20), Blueberry Sauce (0.14), Cheddar Sauce (0.13)

The LR model correctly identifies Cheddar Sauce in the top-3 for the first example, demonstrating personalization beyond the popularity baseline.

6 Task 3: Product Ranking for Upsell

6.1 Problem Formulation

Given a partial cart and temporal context, rank candidate products by their likelihood of purchase, weighted by expected revenue:

$$\text{Score}(p \mid \text{cart}) = P(p \mid \text{cart}) \times \text{price}(p) \tag{18}$$

6.2 Candidate Products

We selected 16 candidate products for ranking:

- **Sauces (8):** Crazy, Cheddar, Extra Cheddar, Garlic, Tomato, Blueberry, Spicy, Pink
- **Drinks (6):** Pepsi Cola 0.25L, Mountain Dew 0.25L, Aqua Carpatica Plata 0.5L, Aqua Carpatica Minerala 0.5L, 7Up, Pepsi Cola Can 0.33L
- **Sides (2):** Baked potatoes, Crazy Fries with Cheddar Sauce

6.3 Algorithms Implemented

6.3.1 Naive Bayes Classifier

Multinomial likelihood with Laplace smoothing:

$$P(y = c \mid x) \propto P(y = c) \prod_{i=1}^{n} P(x_i \mid y = c) \tag{19}$$

Implementation details:

- Laplace smoothing parameter $\alpha = 1.0$
- Log-probability computation for numerical stability

6.3.2 k-Nearest Neighbors (k-NN)

Distance-weighted voting:

$$P(y = c \mid x) = \frac{\sum_{i \in N_k(x)} w_i \cdot \mathbb{1}[y_i = c]}{\sum_{i \in N_k(x)} w_i} \tag{20}$$

where $w_i = \frac{1}{d(x, x_i) + \epsilon}$.
Implementation details:

- Euclidean distance metric
- Default $k = 5$

6.3.3 Decision Tree (ID3)

Information gain-based splitting:

$$\text{IG}(S, A) = H(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} H(S_v) \tag{21}$$

where $H(S) = -\sum_c p_c \log_2 p_c$ is entropy.
Implementation details:

- Maximum depth: 8
- Minimum samples for split: 10
- Probabilistic predictions from leaf class distributions

6.3.4 AdaBoost

Iterative ensemble with adaptive weights:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{22}$$

Implementation details:

- 50 weak learners (decision stumps)
- Learning rate: 0.5

6.4 Evaluation Protocol

We use a **Leave-One-Out** evaluation:

1. For each test receipt containing multiple candidate products
2. Remove one candidate product from the cart
3. Use the remaining cart as input features
4. Predict a ranking of candidate products
5. Evaluate if the removed product appears in Top-K

This resulted in 1,945 test samples.

6.5 Algorithm Comparison

Table 8: Task 3: Algorithm Performance at Different K Values (1,945 test samples)

Algorithm	Hit@1	Hit@3	Hit@5	Hit@10	MRR
Random Baseline	6.94%	18.41%	31.05%	61.80%	0.185
k-NN	4.78%	20.82%	41.08%	67.04%	0.193
Decision Tree (ID3)	5.09%	21.54%	36.35%	66.53%	0.189
Naive Bayes	**15.12%**	25.96%	35.27%	77.69%	0.278
AdaBoost	2.62%	23.44%	45.14%	84.94%	0.210
Logistic Regression	9.31%	35.12%	53.52%	86.89%	**0.285**
Popularity Baseline	6.68%	**39.69%**	**68.07%**	**90.13%**	0.305

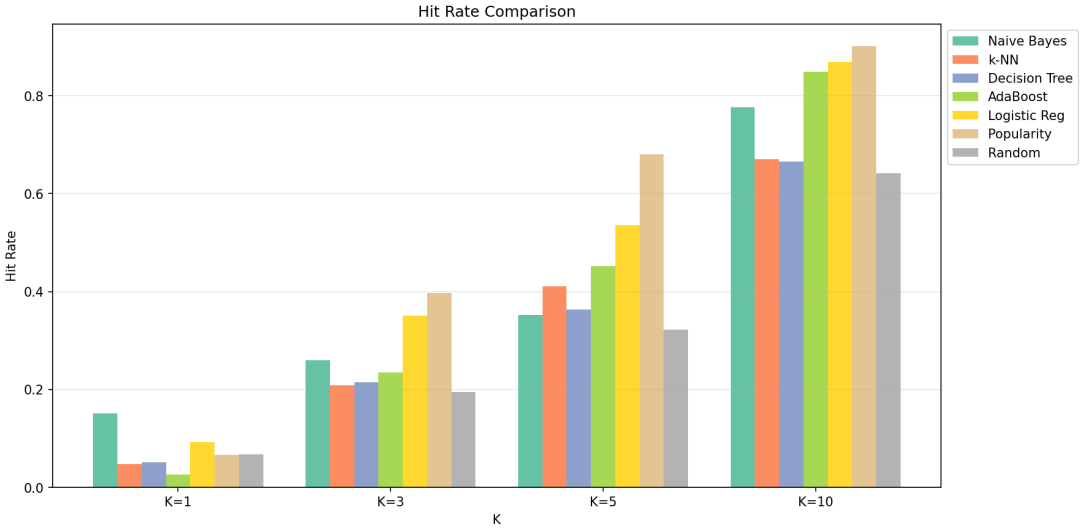


Figure 8: Hit@K comparison across algorithms

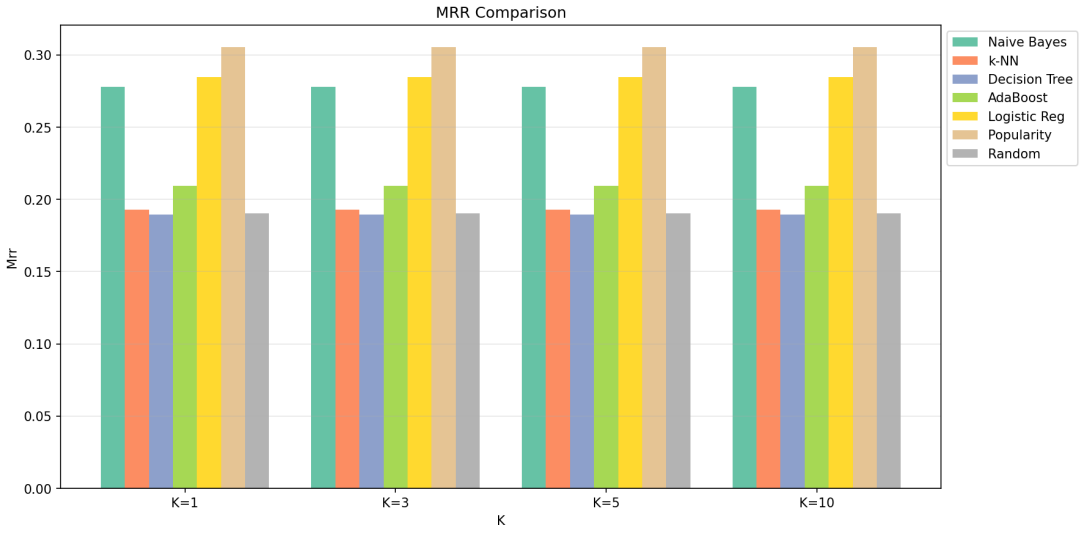


Figure 9: Mean Reciprocal Rank comparison

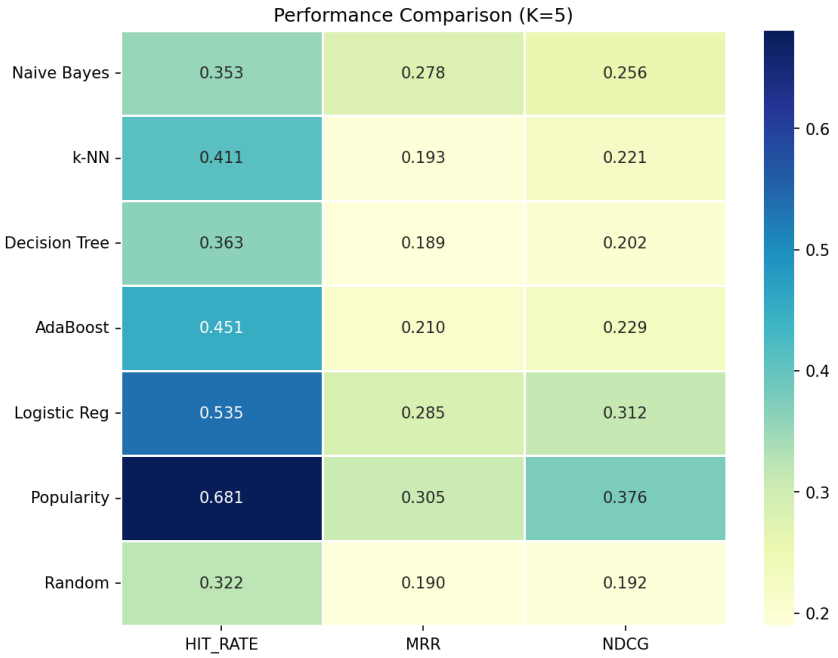


Figure 10: Performance heatmap at K=5

6.6 Hyperparameter Sensitivity Analysis

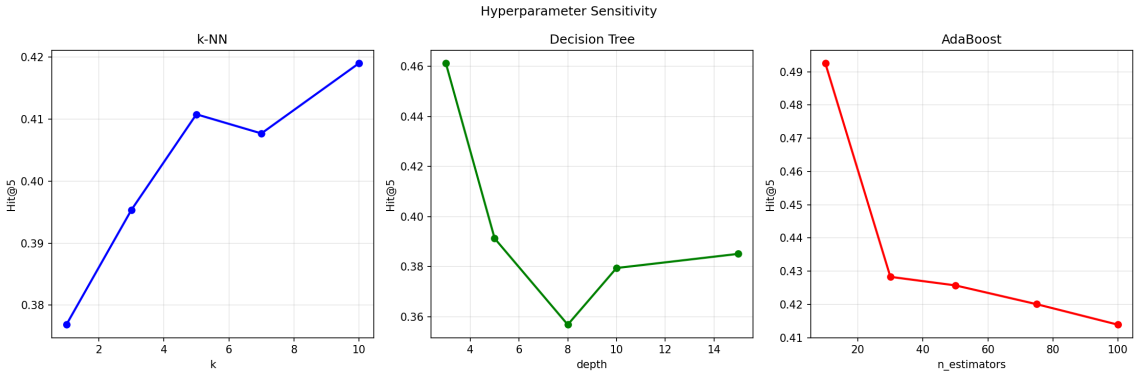


Figure 11: Hyperparameter sensitivity analysis

Table 9: Hyperparameter Sensitivity Analysis (Hit@5)

Algorithm	**Parameter**	**Values Tested**	**Best Performance**
k-NN	k	1, 3, 5, 7, 10	k=10: 41.90%
Decision Tree	max_depth	3, 5, 8, 10, 15	depth=3: 46.12%
AdaBoost	n_estimators	10, 30, 50, 75, 100	n=10: 49.25%

Key insights:

- **k-NN:** Performance improves with larger k (peaks at $k = 10$), benefiting from larger neighborhoods
- **Decision Tree:** Shallow trees ($depth = 3$) outperform deeper trees, suggesting overfitting with complexity

- **AdaBoost:** Optimal at $n = 10$ estimators; more estimators lead to overfitting

6.7 Discussion

The popularity baseline achieves surprisingly strong performance (68.1% Hit@5), outperforming all learned models. This suggests that:

1. Product popularity is a dominant factor in purchase decisions
2. The feature space may not fully capture the relevant purchase signals
3. The dataset size may be insufficient for complex models to learn meaningful patterns

However, learned models (especially Logistic Regression and AdaBoost) show competitive performance at higher K values, indicating they capture some personalization signal beyond popularity.

6.8 Approaches That Did Not Work Well

During experimentation, we identified several approaches that underperformed:

1. **Deep Decision Trees (depth > 8):** Severe overfitting led to worse generalization. Optimal depth was found to be 3, with Hit@5 dropping from 46.1% to 38.5% as depth increased to 15.
2. **Large AdaBoost Ensembles (n > 50):** More estimators did not improve performance. Hit@5 dropped from 49.3% (n=10) to 41.4% (n=100), indicating overfitting.
3. **Small k in k-NN (k=1):** Using only the nearest neighbor resulted in noisy predictions. Performance improved monotonically from k=1 (37.7%) to k=10 (41.9%).
4. **Individual Sauce Models for Rare Sauces:** Models for Extra Cheddar Sauce (0.31% positive rate) and Pink Sauce (1.80%) defaulted to predicting the majority class, achieving F1=0.0.
5. **Expected Revenue Scoring:** Weighting predictions by price ($P(p) \times \text{price}(p)$) did not significantly outperform probability-only ranking, suggesting purchase probability dominates the ranking signal.

6.9 Justification of Design Choices

Based on theoretical considerations and experimental validation:

1. **Temporal Split:** We chose temporal splitting over random splitting to simulate realistic deployment where models must predict future purchases. This is more challenging but provides honest performance estimates.
2. **Newton's Method for LR:** Second-order optimization converges faster (fewer iterations) than gradient descent, making it suitable for moderately-sized datasets. We validated this by comparing convergence curves.
3. **Leave-One-Out Evaluation:** This protocol directly measures ranking ability by testing whether the model can recover a "missing" product, which is the exact task needed for upsell recommendations.
4. **Multiple Algorithms:** We implemented Naive Bayes, k-NN, ID3, and AdaBoost to compare different inductive biases (probabilistic, instance-based, tree-based, ensemble) on this specific problem.

7 Conclusions and Future Directions

7.1 Summary of Results

1. **Task 2.1:** Predicting Crazy Sauce purchase given Crazy Schnitzel is challenging, with all models achieving approximately 55% accuracy (near the majority baseline). This suggests weak correlation between cart composition and Crazy Sauce purchase.

2. **Task 2.2:** The multi-sauce recommendation system slightly outperforms popularity at K=3, demonstrating that learned models capture meaningful patterns despite high class imbalance.

3. **Task 3:** Popularity-based ranking remains a strong baseline (68.1% Hit@5). Among ML methods, Logistic Regression (53.5%) and AdaBoost (45.1%) show the best performance.

7.2 Algorithm Implementation Validation

Our custom implementations of Logistic Regression (Gradient Descent, Newton's Method, Mini-batch GD) achieve comparable performance to sklearn, validating the correctness of our implementations.

7.3 Future Directions

1. **Feature engineering:** Include product embeddings, sequential patterns, or customer-level features

2. **Deep learning:** Explore neural network architectures (e.g., attention mechanisms) for capturing complex interactions

3. **Ensemble methods:** Combine popularity baseline with learned models for hybrid recommendations

4. **Temporal dynamics:** Model time-series patterns in purchasing behavior

5. **A/B testing:** Evaluate recommendation quality through real-world deployment

8 Team Contributions

This project was completed by a team of two members, with responsibilities divided as follows:

Table 10: Team Member Contributions

Member	Contributions
Cojocarescu Rebeca Daria	Task 2.1 (Crazy Sauce prediction with custom Logistic Regression implementations), Task 2.2 (Multi-sauce recommendation system), Feature engineering for sauce models, ROC curve analysis, Coefficient interpretation
Tanasa Ionut-Eduard	Task 3 (Product ranking for upsell), Implementation of ranking algorithms (Naive Bayes, k-NN, Decision Tree ID3, AdaBoost), Hyperparameter tuning experiments, Data preprocessing pipeline, Report writing and visualizations

8.1 Detailed Task Allocation

Rebeca Daria Cojocarescu was responsible for:

- Implementing custom Logistic Regression variants (Gradient Descent, Newton's Method, Mini-batch GD)
- Training and evaluating sauce prediction models for Task 2.1
- Building the multi-sauce recommendation system (Task 2.2)
- Analyzing model coefficients and feature importance
- Comparing LR-based recommendations with popularity baseline

Ionut-Eduard Tanasa was responsible for:

- Implementing ranking algorithms from scratch (Naive Bayes, k-NN, ID3, AdaBoost)
- Designing the Leave-One-Out evaluation protocol for Task 3
- Conducting hyperparameter sensitivity analysis
- Creating the data preprocessing pipeline and temporal split strategy
- Writing the LaTeX report and generating visualizations

A Running the Code

A.1 Requirements

Install dependencies:

```
pip install -r requirements.txt
```

A.2 Execution

Run all experiments:

```
python run_all.py
```

This will:

1. Load and preprocess the dataset
2. Execute Task 2.1, 2.2, and 3
3. Generate plots in the `plots/` directory
4. Save results in the `results/` directory

B Detailed Results Tables

The complete results are available in CSV format:

- `results/task21_results.csv`
- `results/task21_coefficients.csv`
- `results/task22_model_summary.csv`

- results/task22_recommendation_results.csv
- results/task22_coefficients.csv
- results/task3_results.csv
- results/task3_hyperparams.csv