Machine Learning Pipeline for Analysing Fine Food Product Reviews

Group 10

Presented by

Lakshmi Priya Diwakar, Yi-Hsuan Kuo, Ritik Mahajan

TABLE OF CONTENT

PROBLEM STATEMENT

Our goal is to analyze product reviews, extract insights, and build a model to predict the helpfulness of a review.

Objectives:

- Build a model to classify reviews based on their ratings.
- Predicting whether a review is helpful and Identifying the most representative review
- Content-Based Recommendation System for Amazon Reviews

PLAN

- We aim to build a machine learning pipeline using AWS SageMaker to analyze over 500,000 Amazon Fine Food Reviews. The project focuses on classifying reviews based on customer ratings, predicting review helpfulness to identify the most representative reviews, and building a content-based recommendation system to suggest similar products based on review content.

- Online reviews significantly influence purchasing decisions. With AWS SageMaker, we preprocess data, conduct EDA, and train scalable models to automate this analysis and aid both businesses and consumers.

PROJECT IMPLEMENTATION PROCESS

Research

Identifying a problem statement and gathering data.

Data Preprocessing

Cleaning Data and understanding it.

Model Buidling

Using multiple ML Models for NLP application.

ML Pipeline

Building a ML Pipeline using AWS.

DATASET OVERVIEW

We are using the Amazon Fine Food Reviews dataset, available publicly on Kaggle. This dataset contains over 500,000 customer reviews of fine food products sold on Amazon.

This dataset is ideal for:

- Natural Language Processing (NLP) tasks such as sentiment analysis and keyword extraction.
- Trend analysis across users, products, and time.
- Building machine learning models that predict how helpful a review will be based on its content and classify reviews based on its ratings and predict similar products by analyzing reviews

DATA PREPROCESSING

Data Type Handling

- Verified data types using .info() and .dtypes.
- Converted Time to datetime format for time-based analysis.
- Ensured numerical fields like HelpfulnessNumerator and HelpfulnessDenominator were integers.

Handling Missing and Duplicate Values

- Used .isnull().sum() and .describe() to explore null values.
- Replaced null values in review text with placeholder values.
- Dropped duplicate rows to ensure uniqueness and prevent bias.

DATA PREPROCESSING

Text Cleaning

Applied a robust NLP preprocessing pipeline:

- Lowercased all text
- Removed punctuation and special characters
- Tokenized text
- Removed stopwords using NLTK
- Applied lemmatization using WordNetLemmatizer

1.Automated Rating Prediction from Food Reviews using Machine Learning

Problem Statement:

- Build an ML-based classification system to predict customer review ratings from Amazon Fine Food Reviews, addressing challenges of data imbalance, feature extraction, and scalable model deployment.

Data Handling:

- The raw review dataset was uploaded to an AWS S3 bucket, serving as a centralized storage location from which the data was retrieved for preprocessing, feature engineering, and model training steps.

Input Features:

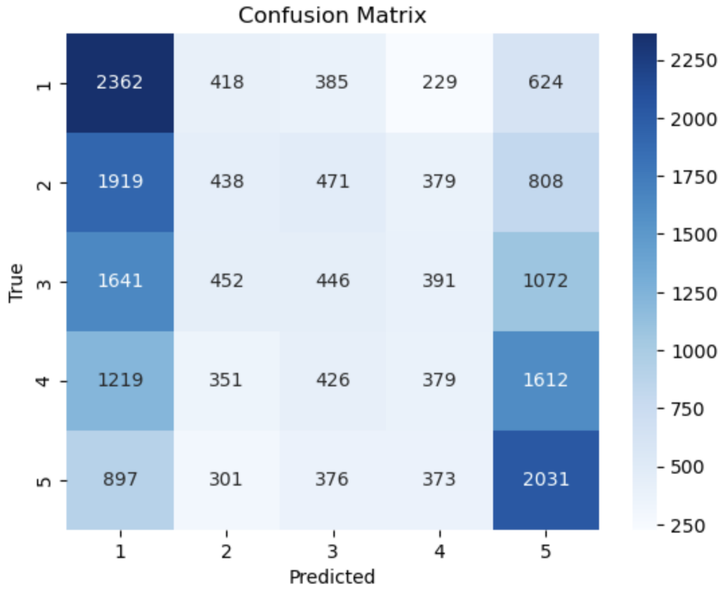
- TF-IDF vectorized text (CleanedText) with top 5000 features.
- Helpfulness Ratio: Calculated as HelpfulnessNumerator / HelpfulnessDenominator.
- Sentiment Score: Derived using TextBlob polarity.
- Subjectivity Score: Derived using TextBlob subjectivity.
- Text Length: Total number of words in each review.

Target Variable:

- Review Score

Model 1: Support Vector Classifier Model

- A **5-class classification model.**
- Applied **TF-IDF vectorization** (top 5000 features) and combined with **Sentiment**, **Helpfulness Ratio**, **Text Length**, and **Subjectivity** features.
- Used **Truncated SVD** to reduce TF-IDF dimensions, retaining **over 95% variance** for efficiency.
- Trained **SVC with linear kernel**, tuned hyperparameters using **GridSearchCV**; achieved **28% accuracy**, with difficulties handling minority classes.
- The model could **only reliably classify very clear positive or negative reviews**, but **often confused middle/neutral cases**, meaning it's useful for basic separation but not yet ready for fine-grained real-world decision-making.




```
Confusion Matrix:
[[2362  418  385  229  624]
 [1919  438  471  379  808]
 [1641  452  446  391 1072]
 [1219  351  426  379 1612]
 [ 897  301  376  373 2031]]
Classification Report:
              precision    recall  f1-score   support

           1       0.29      0.59      0.39      4018
           2       0.22      0.11      0.15      4015
           3       0.21      0.11      0.15      4002
           4       0.22      0.10      0.13      3987
           5       0.33      0.51      0.40      3978

    accuracy                           0.28     20000
   macro avg       0.26      0.28      0.24     20000
weighted avg       0.26      0.28      0.24     20000
```

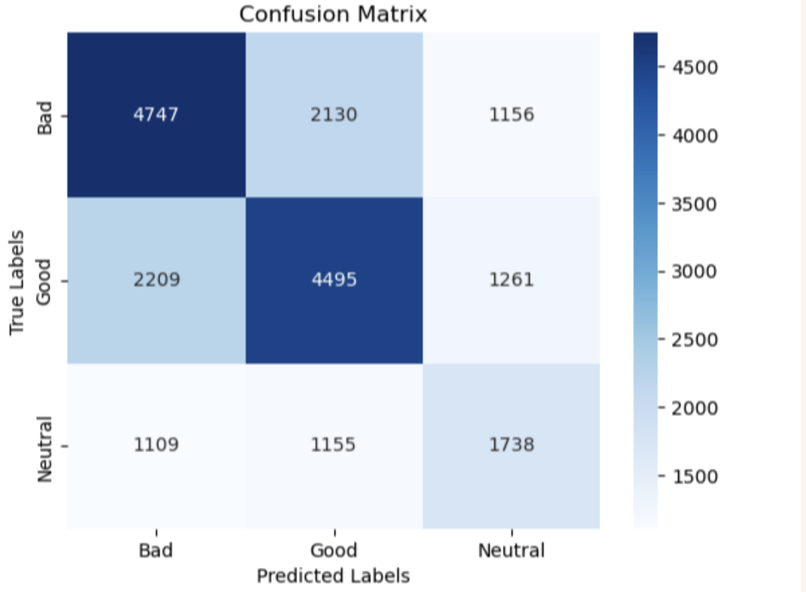
Model 2: Random Forest Classifier Model

- A **3- class classification model**. Predicting 5 individual ratings (1–5) led to confusion between adjacent scores; simplifying into 3 categories (*Bad*, *Neutral*, *Good*) made the classification task more realistic and better suited for practical sentiment analysis.
- **TF-IDF vectorization of text** (top 5000 terms) combined with Sentiment, Helpfulness Ratio, Text Length, and Subjectivity features, scaled uniformly.
- **Truncated SVD** applied to compress the TF-IDF features while keeping over 95% of variance, improving efficiency without losing critical information.
- Random Forest Classifier trained using RandomizedSearchCV with **class balancing (class_weight='balanced'); achieved 54.9% accuracy, showing much better precision and recall than SVC across all classes.**
- The model could **reliably classify clear positive and negative reviews, and handled neutral reviews moderately well**, making it a good candidate for real-world review filtering systems like customer feedback analysis and online review monitoring.

```
Accuracy: 0.549
Confusion Matrix:
 [[4747 2130 1156]
 [2209 4495 1261]
 [1109 1155 1738]]
Classification Report:
              precision    recall  f1-score   support

         Bad       0.59      0.59      0.59      8033
        Good       0.58      0.56      0.57      7965
     Neutral       0.42      0.43      0.43      4002

    accuracy                           0.55     20000
   macro avg       0.53      0.53      0.53     20000
weighted avg       0.55      0.55      0.55     20000
```



Model 3: XGBoost Classifier Model

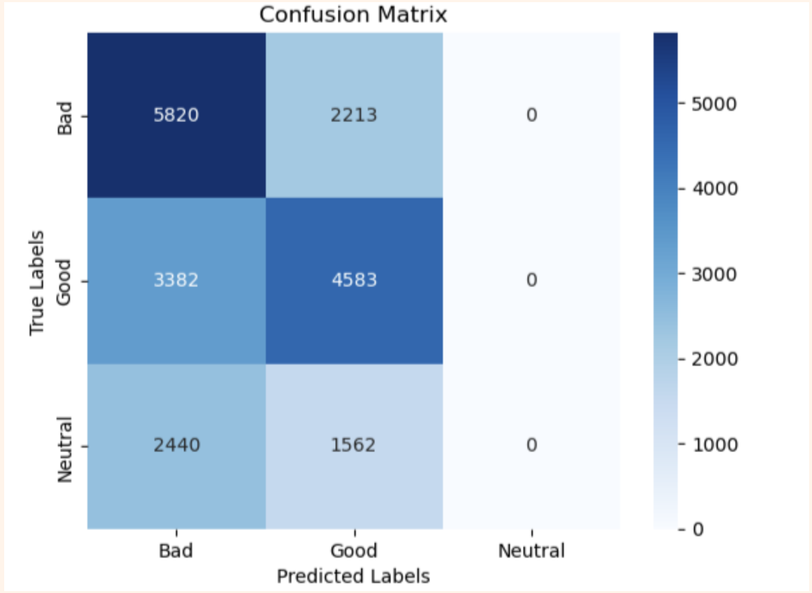
- A **3- class classification model**.
- Applied **TF-IDF vectorization** (top 5000 terms) on cleaned text and combined with scaled numeric features — **Sentiment**, **Helpfulness Ratio**, **Text Length**, and **Subjectivity**.
- **Truncated SVD** was used to compress feature space while keeping **over 95% of information**, optimizing model training efficiency.
- **XGBoost Classifier** trained with **RandomizedSearchCV**; handled imbalance using **sample weighting** and **label encoding**; achieved **52% accuracy** and an **RMSE of approximately 0.91**, performing well for *Bad* and *Good* classes but struggled with *Neutral* predictions.
- The model can **reliably detect clear positive and negative reviews**, but **neutral reviews remain harder to catch**, making it a strong option for **large-scale review classification** where catching extremes matters more than subtle middle cases.

```
Accuracy: 0.52015
Confusion Matrix:
 [[5820 2213    0]
 [3382 4583    0]
 [2440 1562    0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.72      0.59      8033
           1       0.55      0.58      0.56      7965
           2       0.00      0.00      0.00      4002

    accuracy                           0.52     20000
   macro avg       0.35      0.43      0.38     20000
weighted avg       0.42      0.52      0.46     20000

Root Mean Squared Error (RMSE): 0.919701038381495
```



2. Identifying Helpful and Representative Reviews Using Machine Learning

Motivation: Review data often lacks feedback (many reviews not voted helpful or not)
Goal: Predict if a review is helpful
Bonus: Verify if 'representative' reviews are actually more helpful

Plan:
•Predicting whether a review is helpful (HelpfulBinary) using XGBClassifier.
•Identifying the most representative review per product using BERT-based semantic similarity.
•Evaluating whether these representative reviews tend to be helpful

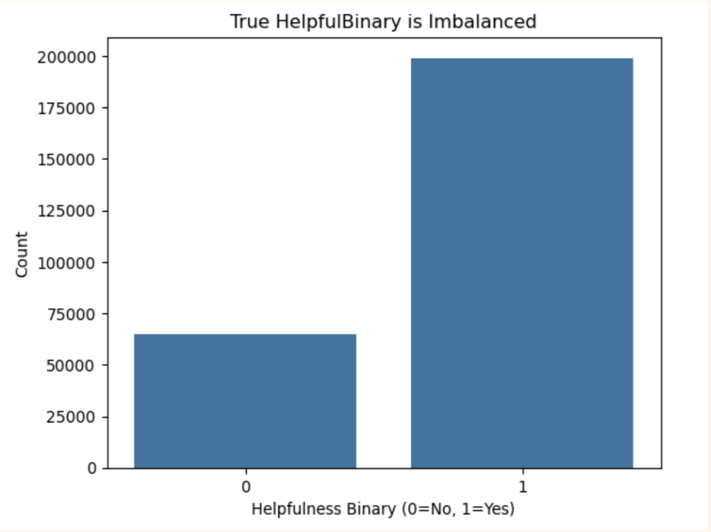
Data Handling:
- Clean text
- Calculate helpfulness ratio and Label HelpfulBinary
- Calculate recency weight -- Derived using Timestamp
- Generate semantic similarity scores (cosine similarity) -- Derived using Text and a pre-trained BERT model
- Split the data into: Reviews **with** HelpfulBinary label and Reviews **without** HelpfulBinary label

Model design & Evaluation

- XGBoost classifier:
  - **Input Features:**
    1. **TF-IDF vectorized text** (CleanedText) Doing SVD or not
    2. **Recency_weight**: represent weight of time
    3. **cosine_avg_sim**: represent the representative score
  - **Target Variable: Helpfulness Binary**
  - **Parameters: n_estimators=1000, max_depth= 7 or 8, learning_rate=0.1**

, scale_pos_weight **= (HelpfulBinary = 0) / (HelpfulBinary = 1)**

- Training/test split: 80/20



Raw TF-IDF features

	precision	recall	f1-score	support
0	0.55	0.71	0.62	12887
1	0.90	0.81	0.85	39912
accuracy			0.79	52799
macro avg	0.72	0.76	0.73	52799
weighted avg	0.81	0.79	0.79	52799

SVD-reduced TF-IDF features

	precision	recall	f1-score	support
0	0.73	0.60	0.66	12887
1	0.88	0.93	0.90	39912
accuracy			0.85	52799
macro avg	0.81	0.77	0.78	52799
weighted avg	0.84	0.85	0.84	52799

Representative Review Analysis

For all data with helpfulBinary(ground truth)

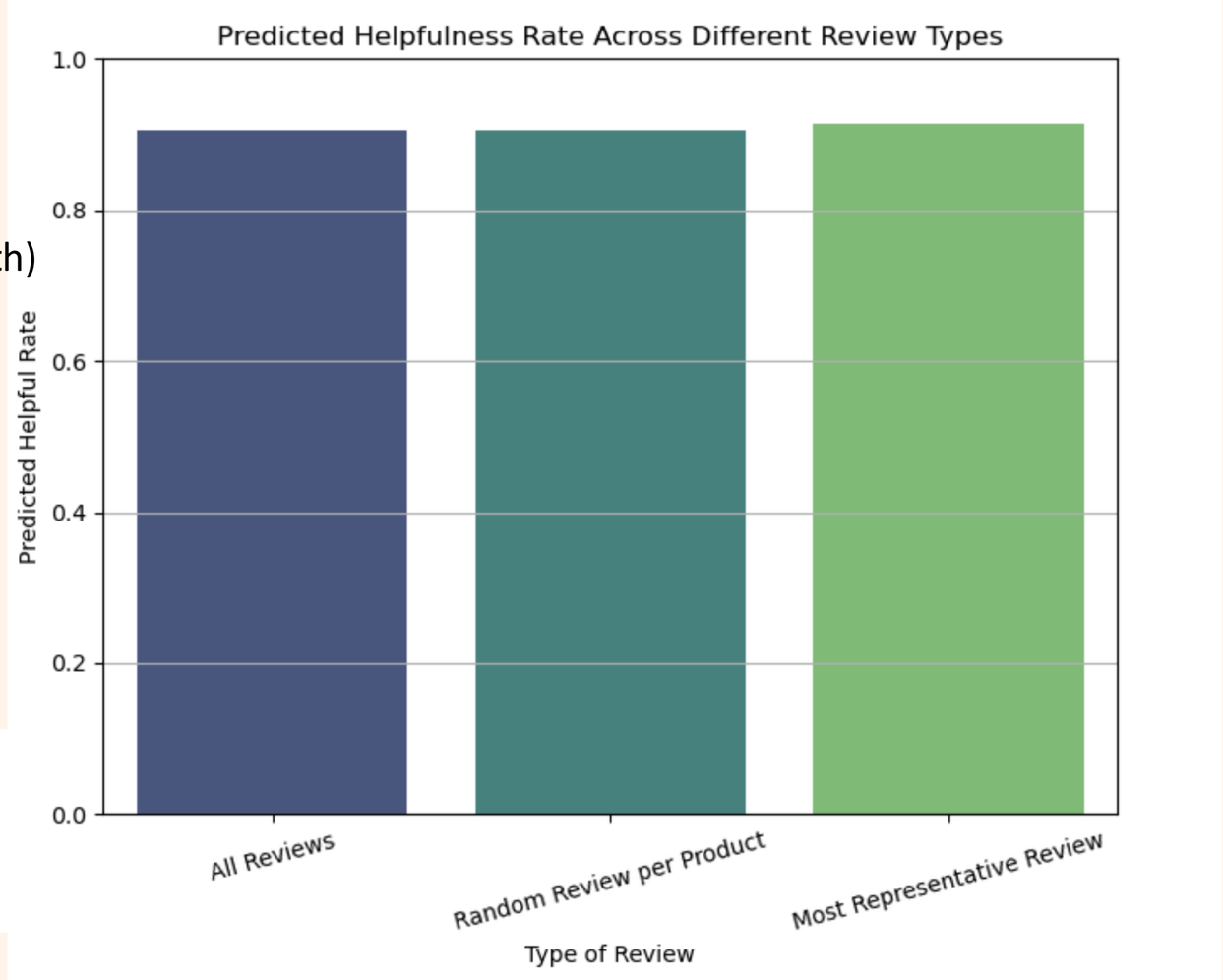
```
HelpfulBinary = 1: 0.7540389328504923
HelpfulBinary = 0: 0.24596106714950774
```

For most representative review with helpfulBinary(ground truth)

```
HelpfulBinary = 1: 0.85193889541715 63
HelpfulBinary = 0: 0.14806110458284372
```

we compare the helpfulness of:
- Most representative reviews
- All reviews
- Random reviews (1 per product)

Type of Review	Predicted Helpful Rate
All Reviews	0.905235
Random Review per Product	0.906284
Most Representative Review	0.914828



Conclusion

1. XGBoost classifier performed well in predicting helpfulness with 85% accuracy

2. We use the trained XGBClassifier model to predict whether reviews without any helpfulness votes are likely to be helpful

3. Dimensionality reduction by SVD helped with model efficiency.

4. "representative reviews" lead to higher perceived helpfulness, validating our feature engineering and match our common sense

3.Content-Based Recommendation System for Amazon Reviews

Problem Statement:

- Build an ML-based recommendation system to suggest similar products from Amazon Fine Food Reviews.

Data Handling:

- The raw review dataset was uploaded to an AWS S3 bucket, serving as a centralized storage location from which the data was retrieved for preprocessing, feature engineering, and model training steps.

Input Features:

- ProductId (to identify items)
- CleanedText (the preprocessed review text used for embedding generation)

Target Variable:

- ProductId

Methodology-

- Approach: Leveraged natural language processing to build a content-based product recommendation system using semantic similarity between product reviews

- Algorithm Selection:
  - Used sentence-transformers library with pre-trained all-MiniLM-L6-v2 model to generate dense vector representations (embeddings) of product reviews
  - Generated a single embedding vector per product by averaging embeddings of all its associated reviews
  - Applied cosine similarity to measure semantic relatedness between product embeddings

- Implementation Details:
  - Grouped cleaned review texts by ProductId
  - Generated embeddings for each review using the pre-trained model
  - Calculated product-level embeddings through averaging
  - Computed full similarity matrix for efficient recommendation retrieval

Results & Evaluation-

Qualitative Evaluation Results

- Evaluation Method: Performed qualitative spot-checking to assess recommendation quality
    - Randomly sampled products and examined their top 5 recommendations
    - Cross-referenced product IDs with original dataset to verify relevance
    - Assessed contextual appropriateness based on similarity scores

- Key Findings:
    - Most recommendations were relevant and contextually appropriate
    - Only occasional ambiguous recommendations (typically 1 out of 5)
    - System successfully captured meaningful similarities between products based on review content
    - Higher similarity scores generally corresponded to more intuitive recommendations

CHALLENGES FACED

- **Kernel Crashes**: Kernel frequently died due to large dataset size and limited memory-efficient instances.
- **Dataset Imbalance**: Significant class imbalance required undersampling and class-weight adjustments to balance the data.
- **Model Selection**: Identifying the right ML model and tuning it for better accuracy and stability was challenging.
- **AWS SageMaker Access**: Access denied errors restricted the use of pre-built NLP models in SageMaker, limiting experimentation.

FUTURE DIRECTIONS

1. Deploy Models in Production
Integrate the trained models into a live application using AWS SageMaker endpoints or AWS Lambda for real-time review classification, helpfulness prediction, and recommendations.

2. Improve Model Performance
Fine-tune advanced deep learning models like **BERT**, **DistilBERT**, or **GPT-based architectures** for better text understanding and improved prediction accuracy across tasks.

3. Expand Dataset and Features
Incorporate additional data such as **user demographics**, **product name**, **category** to enrich feature sets and build more personalized and context-aware systems.


THANK
YOU