Amazon Fashion

COMP 262

Group 1:

Alireza Farkhondeh Vishkasoghi

Leor Gerber

Artem Kamov

Arash Karimi

Harsh Kumar

Table of Content

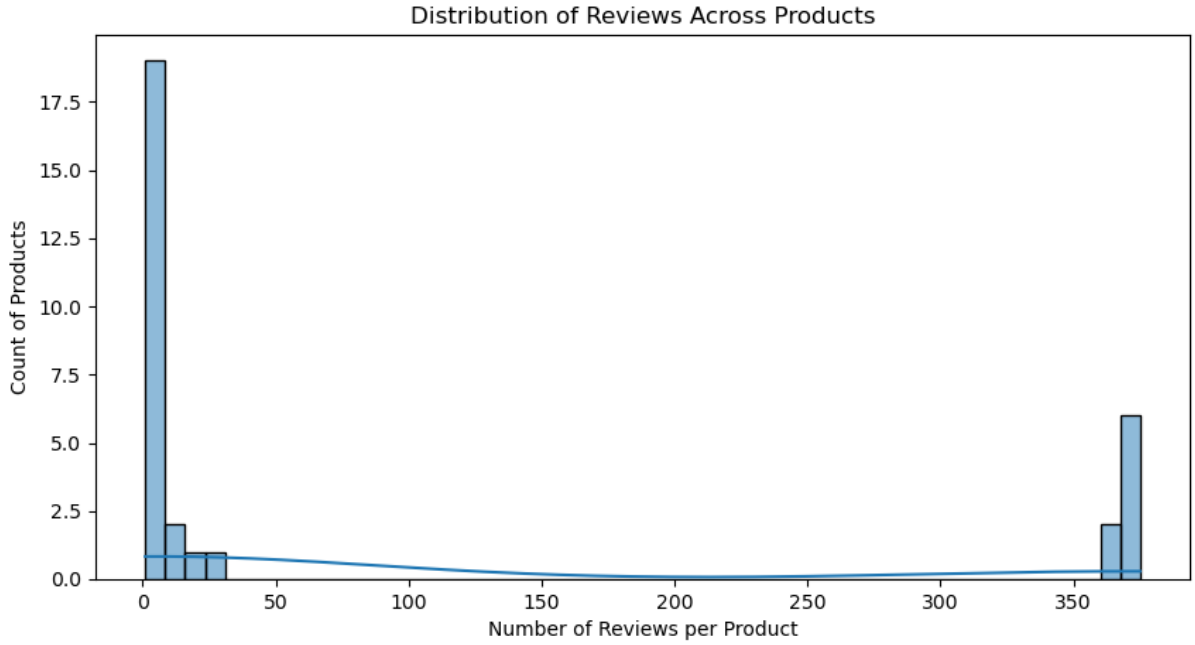
1. Detailed Results of Dataset Exploration & Conclusions for Each Phase

Phase 1: Data Exploration

Distribution of Reviews Across Products:

With a lengthy tail of products with just one to five ratings, the majority of products have extremely few evaluations. Although they make up a small portion of the entire dataset, a few goods have more than 300 ratings. This implies that certain well-liked products have significantly higher levels of engagement than others.

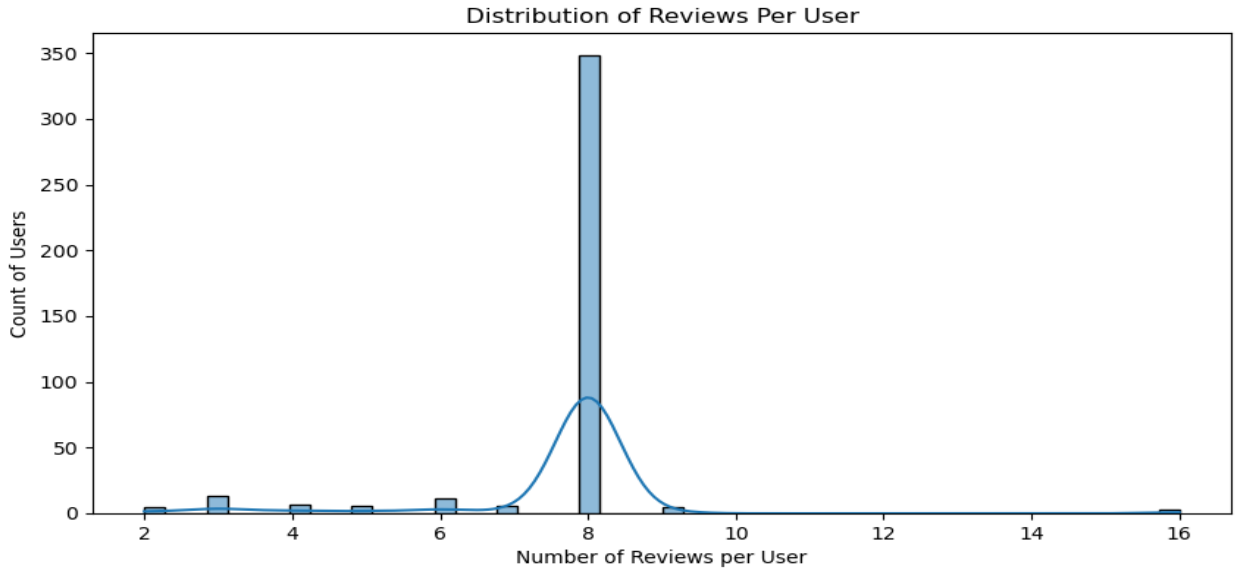
The distribution of reviews across products is significantly skewed, which may have an impact on sentiment analysis. These imbalances may need to be addressed in later phases of the dataset.



Distribution of Reviews Per User:

Merely a small percentage of users have more than ten reviews, while many merely have eight.

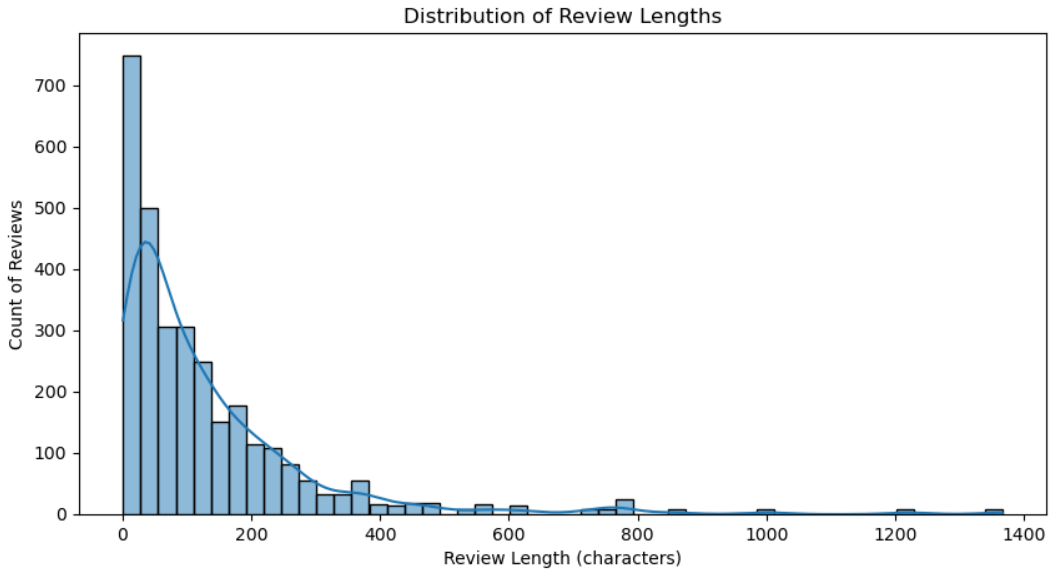
Conclusion: While some people may have been more active, the majority of users are mediocre reviewers with few reviews. Since the evaluations most likely represent a consistent user experience, this uniformity can aid in sentiment analysis.



Distribution of Review Lengths:

Few reviews are lengthy (more than 600 characters), while most are rather brief (less than 100 characters).

Conclusion: The distribution of review lengths shows that, as is common for product reviews, the majority of reviews are brief. Outliers include lengthy reviews, so it's critical to look for any biases they might introduce.



For Phase 2:

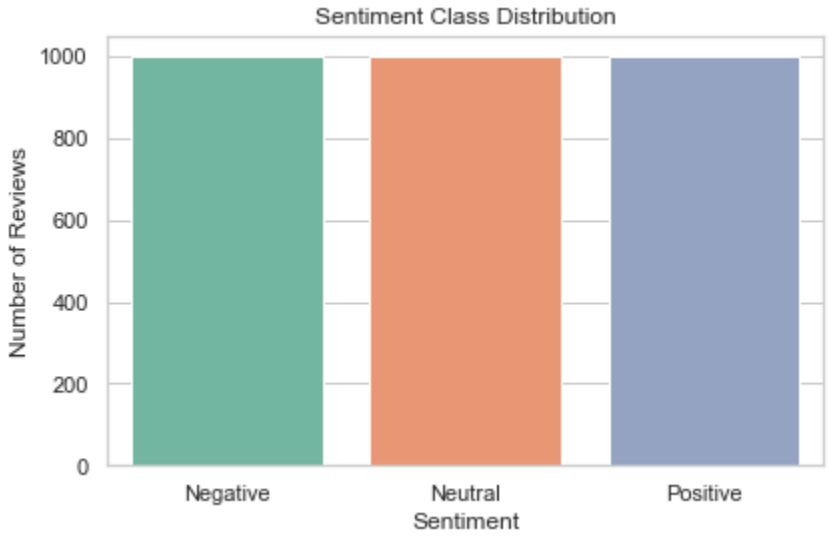
1. Class Distribution

To enable supervised learning, we labeled sentiments as:

- **Positive**: Ratings of 4 or 5
- **Neutral**: Rating of 3
- **Negative**: Ratings of 1 or 2

2. A **balanced subset** of 3,000 reviews (1,000 per class) was selected:

- Positive: 1000
- Neutral: 1000
- Negative: 1000

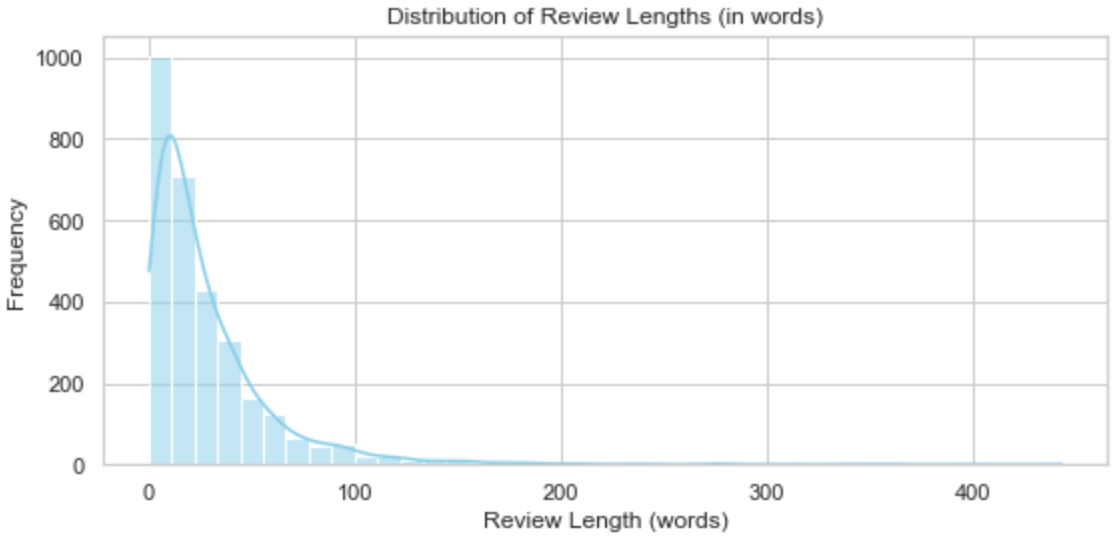


3. Missing Values & Duplicates

- Missing 'text' or 'rating' fields were removed entirely.
- 68 duplicate reviews were removed using hashable columns only.

4. User Behavior Observations

- Many users submitted only 1–2 reviews, while a small number contributed over a dozen.
- Outliers exist (e.g., extremely high or low rating paired with very short or very long text).



2. Dataset Pre-processing Steps with Explanation and Justification of Choices

In order to prepare the datasets for the lexicon-based method (Phase 1) and the machine learning models (Phase 2), the preprocessing step was essential. Despite working with Amazon Fashion reviews, the preprocessing pipelines needed for each phase varied based on the modeling technique.

Phase 1: Lexicon-Based Models (TextBlob and VADER)

1. Managing Missing Values
   - Action: Removed rows with missing values in the reviewText column.
   - Justification: Since reviewText was the core input for sentiment analysis, reviews without this field could not be processed by either TextBlob or VADER.
2. Removing Duplicates
   - Action: Converted list/dictionary-type fields to strings and removed exact duplicates.
   - Justification: Prevented repeated reviews from skewing sentiment scores or inflating performance metrics.

3. Basic Text Preprocessing
    - TextBlob only: Although the cleaned text was stored, only the raw reviewText was used for sentiment scoring by TextBlob and VADER.
    - Steps included:
        - Lowercasing
        - Punctuation removal
        - Tokenization
        - Stopword removal *(TextBlob only)*
4. Sentiment Labeling
    - Action: Used product ratings to label sentiment:
        - 4–5 → Positive
        - 3 → Neutral
        - 1–2 → Negative
    - Justification: This rule-based labeling aligns with standard sentiment definitions and allowed us to evaluate lexicon predictions.
5. Data Selection for Sentiment Models
    - Action: Selected relevant columns (cleaned_review, summary, sentiment) and saved the processed data to processed_reviews.csv.
    - Justification: This allowed consistent input for both TextBlob scoring and result comparison.

Phase 2: Machine Learning Models

1. Managing Missing Values
    - Action: Removed rows with missing text or rating fields.
    - Justification: Both fields were required — text for input features and rating for sentiment labeling.
2. Removing Duplicates
    - Action: Removed 68 duplicate rows using only hashable columns (excluding lists/dictionaries).
    - Justification: Reduced redundancy and avoided misleading model evaluation.
3. Text Cleaning
    - Action: Applied a custom function to:
        - Convert text to lowercase
        - Remove punctuation and non-alphanumeric characters
    - Justification: This standardized the input text and reduced noise for TF-IDF vectorization.
4. Sentiment Labeling
    - Same strategy as Phase 1:
        - 4–5 → Positive
        - 3 → Neutral
        - 1–2 → Negative
    - Justification: Maintained consistency across both phases for accurate model comparisons.
5. Balanced Sampling
    - Action: Sampled 1,000 reviews per sentiment class for a total of 3,000 reviews.
    - Justification: Ensured class balance to prevent training bias and support fair evaluation.
6. TF-IDF Vectorization

- Tool: TfidfVectorizer with max_features=5000.
- Justification: This technique emphasizes unique terms while controlling dimensionality. It is widely used for transforming textual data into usable features for machine learning.

Most reviews had between 20–100 non-zero TF-IDF features, indicating moderate text richness.

7. Train-Test Split
   - Action: Applied a 70/30 stratified split using train_test_split.
   - Justification: Stratification ensured proportional representation of all sentiment classes in both training and test sets.
8. Exporting Processed Data
   - Files generated:
     - phase2_processed_data.csv
     - X_train_cleaned_ml.csv, X_test_cleaned_ml.csv
     - y_train.csv, y_test.csv
   - Justification: Exporting data ensured reproducibility and made the modeling phase modular and efficient.

3. Text representation model with explanation and justification (Only in phase #2)

We employed the TF-IDF (Term Frequency–Inverse Document Frequency) technique to transform unprocessed review text into numerical characteristics appropriate for machine learning models.

Tool and Implementation

Using TfidfVectorizer from the scikit-learn library, we carried out TF-IDF vectorization with the following setup:

- max_features=5000 to select the top 5,000 most informative words
- Applied to the cleaned version of each review stored in the cleaned_text column

Why TF-IDF?

TF-IDF is a widely adopted text representation method that balances two key metrics:

- **Term Frequency (TF)**: Measures how frequently a word appears in a document
- **Inverse Document Frequency (IDF)**: Measures how unique a word is across the entire corpus

Common terms like "the," "nice," and "good" are guaranteed to have less influence thanks to this weighting, while more distinctive and sentimental words like "fragile," "disappointed," and "beautiful" are given greater weight.

Justification for Choosing TF-IDF

- **Efficient**: Requires no additional training like embeddings or transformer models
- **Compatible**: Works well with classical ML algorithms (e.g., Logistic Regression, SVM)
- **Interpretability**: Each feature corresponds directly to a real word, aiding analysis
- **Scalable**: Handles large datasets and reduces dimensionality through max_features

4. Models:

A. Assumptions/Heuristics/Algorithms/Packages Used

VADER:

- **Assumptions:** VADER relies on a sentiment lexicon and assumes that sentiment can be derived from words and their context (such as punctuation and capitalization).
- **Algorithm:** It uses a set of rules to assign sentiment scores based on a lexicon and modifies those scores depending on the context, such as negations or intensifiers.
- **Package Used:** NLTK (Natural Language Toolkit) SentimentIntensityAnalyzer.

TextBlob:

- **Assumptions:** TextBlob assumes that polarity can be derived from the review text and uses sentiment-based machine learning models trained on large text corpora.
- **Algorithm:** It calculates sentiment by using the polarity and subjectivity of each review. A score above 0 indicates positive sentiment, below 0 indicates negative sentiment, and 0 means neutral.
- **Package Used:** TextBlob Python library.

Logistic Regression:

- **Assumptions:** Assumes a linear relationship between TF-IDF features and the log-odds of sentiment classes.
- **Algorithm:** A linear classifier that models the probability of each sentiment class using a sigmoid activation function.

- **Package Used:** sklearn.linear_model.LogisticRegression.

Support Vector Machine (SVM):

- **Assumptions:** Assumes the data is not linearly separable and uses the kernel trick to find a nonlinear decision boundary.
- **Algorithm:** Constructs hyperplanes in a high-dimensional space to separate classes with maximum margin.
- **Package Used:** sklearn.svm.SVC with kernel='rbf'.

B. Explain Each Model, How It Works

VADER:

VADER assigns sentiment scores based on a pre-established lexicon after parsing the words in a text. Additionally, it modifies ratings based on elements such as punctuation (e.g., "!!!"), capitalization, degree modifiers, and negations. It's well-suited for social media texts, reviews, and other informal content where tone and emphasis affect meaning.

TextBlob:

TextBlob analyzes the review and assigns a polarity score that ranges from -1 (negative) to 1 (positive), as well as a subjectivity score. It uses a naive Bayes classifier or predefined rules depending on implementation. It works well on formal or standard English but might miss nuances in slang or informal expressions.

Logistic Regression:

This model uses the TF-IDF feature set to calculate the weighted sum of features, then applies a sigmoid function to convert the result into a probability for each class. The class with the highest probability is chosen. It's interpretable and effective for linearly separable sentiment tasks.

Support Vector Machine (SVM):

SVM transforms the input data into a higher dimension using a radial basis function (RBF) kernel and finds the optimal hyperplane that separates different sentiment classes. It is especially useful in handling complex, non-linear relationships between features.

C. Fine-Tuning Steps

VADER:

- Domain-specific tuning can be done by extending the sentiment lexicon with context-relevant words or phrases (e.g., slang, brand-related expressions).
- Adjust thresholds or post-process predictions based on observed errors in validation.

TextBlob:

- Can be enhanced using a Naive Bayes classifier trained on a domain-specific corpus.
- Alternatively, extend its lexicon or retrain the model on customized labeled datasets.

Logistic Regression:

- Used max_iter=1000 to ensure the model converges during training.
- Fine-tuning options include adjusting C (regularization strength), using class_weight='balanced' to handle class imbalance, or performing cross-validation to identify optimal hyperparameters.

Support Vector Machine (SVM):

- Default hyperparameters were initially used.
- Tuning parameters like C (controls margin hardness), gamma (controls influence of individual data points), and class_weight (to boost performance for underrepresented classes) can improve results.
- GridSearchCV can be applied for systematic hyperparameter optimization.

5. Training results summary (only phase #2).

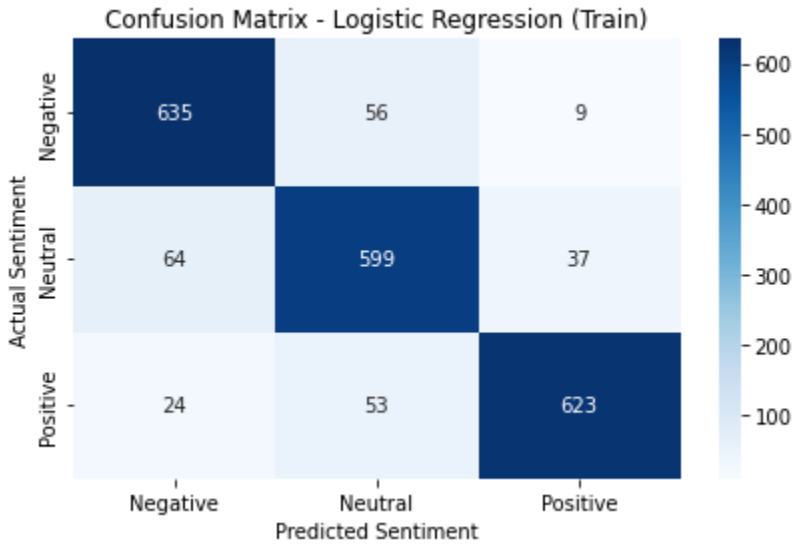
Following TF-IDF transformation of the text data, we used 70% of the dataset (2,100 samples) to train two supervised machine learning models: Support Vector Machine (SVM) using an RBF kernel and Logistic Regression. The outcomes of their performance on the training set are shown below.

Logistic Regression (Train Set):

- **Accuracy**: 88.4%
- **Macro Precision**: 0.885
- **Macro Recall**: 0.884
- **Macro F1 Score**: 0.884

Sentiment	Precision	Recall	F1-Score
Negative	0.88	0.91	0.89
Neutral	0.85	0.86	0.85
Positive	0.93	0.89	0.91

Confusion Matrix:

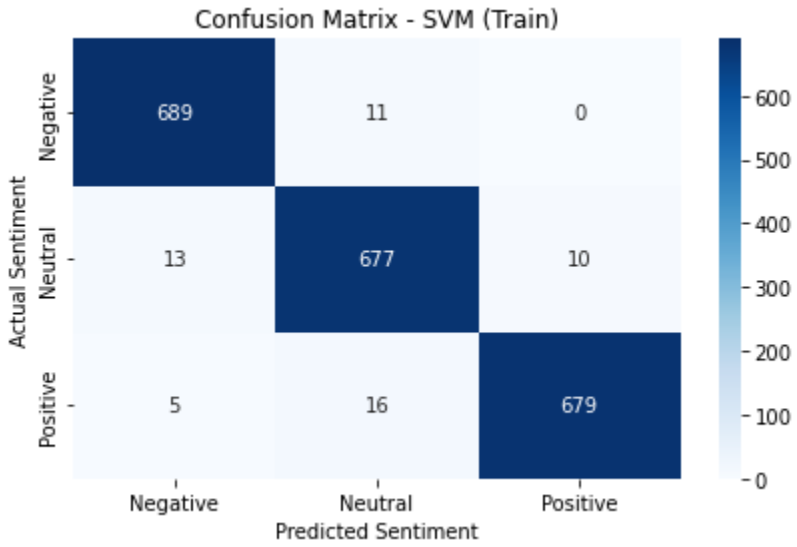


Support Vector Machine (Train Set)

- **Accuracy**: 97.4%
- **Macro Precision**: 0.974
- **Macro Recall**: 0.974
- **Macro F1 Score**: 0.974

Sentiment	Precision	Recall	F1-Score
Negative	0.97	0.98	0.98
Neutral	0.96	0.97	0.96
Positive	0.99	0.97	0.98

Confusion Matrix:

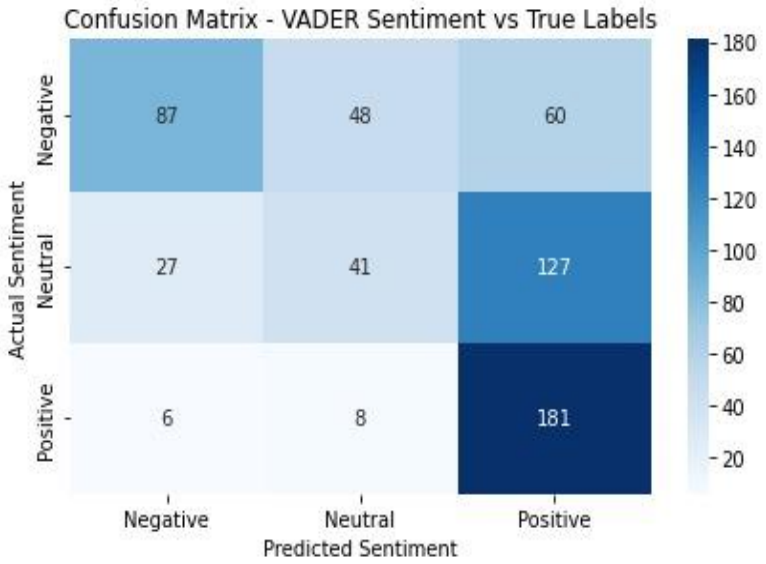


6. Testing results summary.

Metric	VADER	TextBlob	Logistic Regression	SVM (Linear Kernel)
Accuracy	**0.53**	**0.47**	**0.64**	**0.65**

Precision (Negative)	0.72	0.66	0.64	0.62
Recall (Negative)	0.45	0.38	0.67	0.68
F1-Score (Negative)	0.55	0.48	0.65	0.65
Precision (Neutral)	0.42	0.35	0.54	0.55
Recall (Neutral)	0.21	0.06	0.51	0.57
F1-Score (Neutral)	0.28	0.10	0.52	0.56
Precision (Positive)	0.49	0.43	0.74	0.79
Recall (Positive)	0.93	0.96	0.72	0.69
F1-Score (Positive)	0.64	0.59	0.73	0.74

VADER:

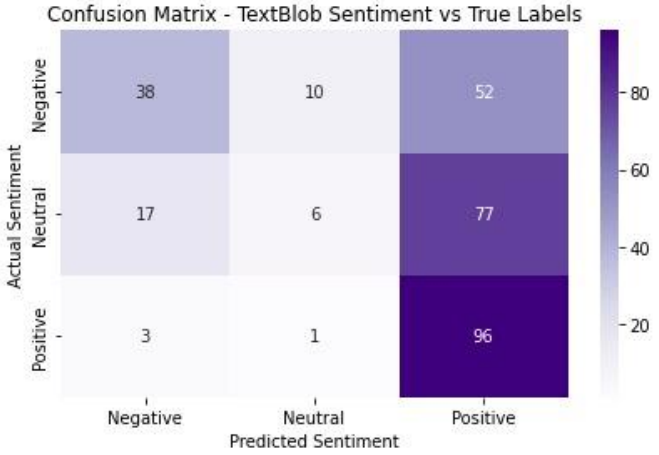


Analysis: VADER performed reasonably well in detecting **positive sentiment** (high recall of 0.93), but its performance on **neutral** and **negative sentiment** was subpar. The **recall for neutral** sentiment was very low (0.21), indicating it often misclassified neutral reviews as positive. It also struggled with **negative sentiment** detection, with a recall of only 0.45.

Confusion Matrix Insights:

- The matrix revealed that VADER had a tendency to **misclassify neutral reviews** as positive, which is a key limitation when analyzing product reviews, where neutral opinions are common.

Textblob:

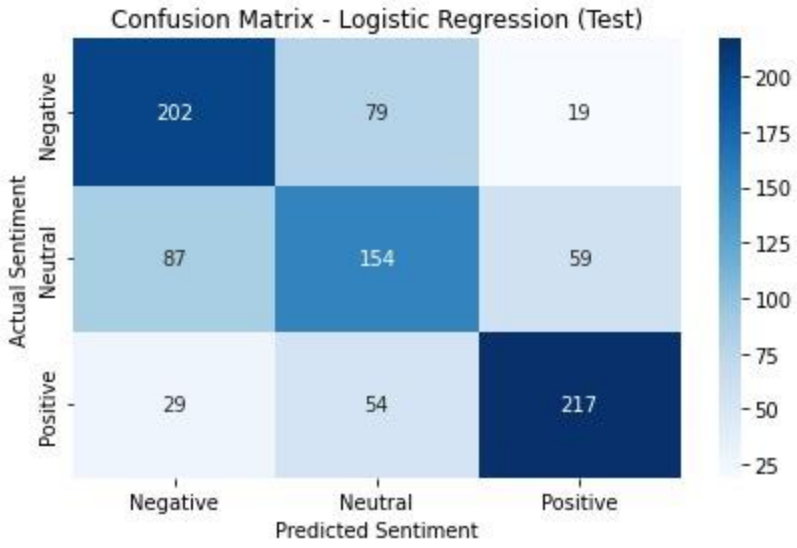


Analysis: TextBlob showed strong performance in detecting **positive sentiment** with a high **recall** of 0.96. However, its performance for **neutral sentiment** was very weak, with a **recall of only 0.06**, which indicates that the model often misclassified neutral reviews as positive. The **recall for negative sentiment** (0.38) was also low, meaning TextBlob missed many negative reviews.

Confusion Matrix Insights:

- Similar to VADER, TextBlob struggled to distinguish **neutral** sentiment and tended to label them as **positive**. It performed reasonably well on positive sentiment but needs improvement for negative and neutral reviews.

Logistic Regression

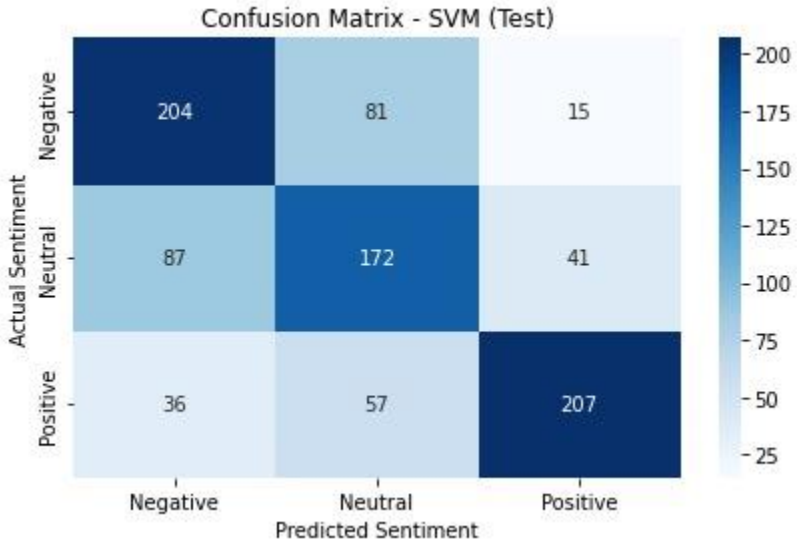


Analysis: Logistic Regression outperformed both VADER and TextBlob, with better overall performance across all sentiment categories. Its **recall for negative sentiment** (0.67) was the highest, indicating it was better at detecting negative reviews. It also demonstrated **balanced performance** for neutral and positive sentiments. With an **accuracy of 0.64**, it performed better than both lexicon-based models.

Confusion Matrix Insights:

- The confusion matrix for Logistic Regression shows that it classified **negative**, **neutral**, and **positive** sentiments more accurately than VADER and TextBlob, with a more even distribution of predictions across the three categories.

SVM



Analysis: The **SVM** model showed slightly better overall performance than **TextBlob**, with **accuracy** of **0.65**. It excelled at detecting **negative sentiment** with a **recall of 0.68**, the highest of all models tested. However, like Logistic Regression, SVM also demonstrated balanced performance across **neutral** and **positive** sentiment categories.

Confusion Matrix Insights:

- The SVM confusion matrix showed strong performance in identifying **negative reviews**, with a good **balance** between detecting **positive** and **neutral** sentiments. The model's overall classification accuracy was higher than VADER and TextBlob, especially for negative reviews.

7. Recommendation System

a. Enhancing Ratings Using Review Data

By identifying emotional cues in review text, we improved conventional rating values, drawing inspiration from the methodology used by Moshfeghi et al. (2011). The main thesis is that a review's expressed emotions—such as happiness, rage, trust, or sadness—often reveal more about the user's preferences than the rating's numerical value alone.

To achieve this:

- We used the **NRCLex** library to extract emotion scores from the cleaned review text.
- Positive emotions (joy, trust) and negative emotions (anger, fear, sadness, disgust) were aggregated to compute an overall **emotion score** between -1 and +1.
- We adjusted the original rating by:

$$\text{adjusted\_rating} = \text{rating} \times (1 + \text{emotion\_score})$$

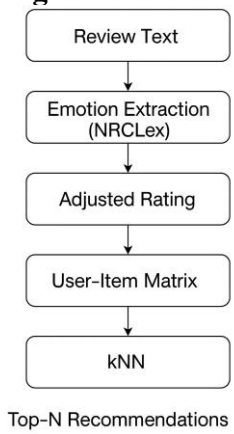
- Values were clipped between 1 and 5 to match the original rating scale.

This adjustment reflects the **emotional intensity and tone** of each review, offering a richer signal for downstream recommendation tasks.

b. Selected Method, Diagram, and Pseudo-code

We used the emotion-space-based modeling approach, which is explained in Section 4.3.5 of the study. This approach adjusts ratings based on the inference of user preferences from emotional context.

Diagram: Emotion-Enhanced Recommender Workflow



Pseudo-code

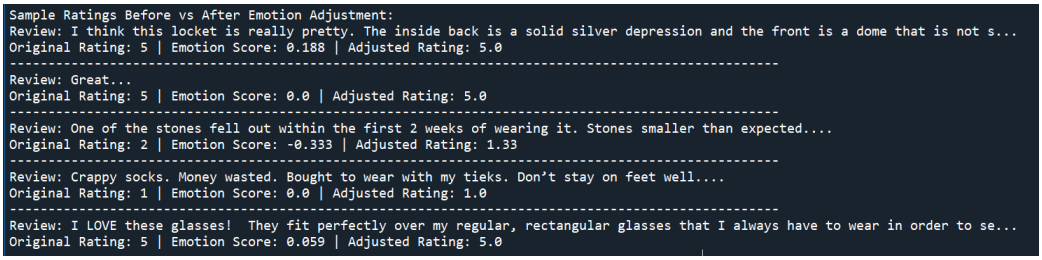
```
#summary of  Pseudo-code
for review in reviews:
    emotion_score = extract_emotion_score(review.text)
    adjusted_rating = rating * (1 + emotion_score)
    store(user_id, item_id, adjusted_rating)

user_item_matrix = build_matrix(user_id, item_id, adjusted_rating)
knn_model = train_KNN(user_item_matrix)
recommend(user_id) = knn_model.get_top_neighbors(user_id)

```

c. Implementation & Results

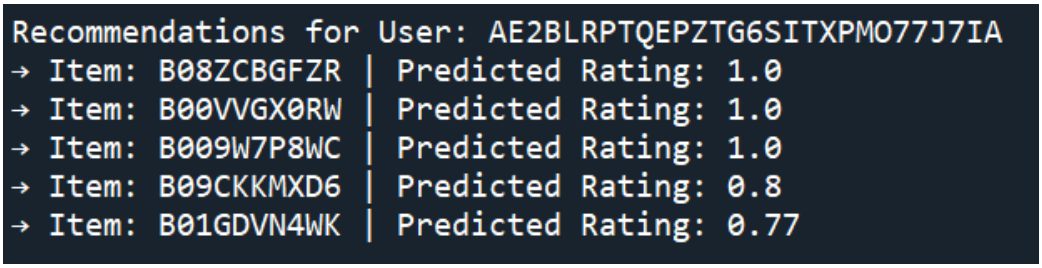
We implemented the approach on the full Amazon Fashion JSONL dataset. Here's how it worked:

- extracted and cleaned the dataset's user_id, asin, text, and rating.
- calculated emotion scores for almost 2.4 million reviews using NRCLex.
- used pandas to compute adjusted_rating and produce a user-item matrix.
- used cosine similarity to train a K-Nearest Neighbors (KNN) recommender.
- tested suggestions for a user chosen at random.

Sample Output (Before vs After Adjustment)



Generated Recommendations for User AE2BLRPTQEPZTG6SITXPMO77J7IA:



As demonstrated by Moshfeghi et al. (2011), these findings demonstrate that emotion scores can alter raw ratings to enhance preference signals, which is very useful in sparse data conditions.

8. LLM summarization

The LLM used for this task was the **T5-small** model from Hugging Face. This model was selected due to its lightweight architecture, which allows for fast loading and efficient summarization without compromising too much on quality. When processing a review, the model identifies and prioritizes key phrases that carry positive or negative sentiments.

For example, given the original review:

"I really liked this shirt when I received it. It is a really nice top. I ordered the brown one. The color of the shirt really looks nice with the floral sleeves. The surprise is that the sleeves are not T-shirt material. They are of a good quality chiffon. The knit fabric is good quality, also.
I am 5'4" and weigh about 172 lbs. I am a 36 DDD. I ordered an XXL. It is slightly on the big size. It looks good, I would just prefer it a bit more form fitting. The shirt covers my hips (length wise)."

The model generated the following summary:
"The color of the shirt really looks nice with the floral sleeves. The shirt is of a good quality chiffon. It covers my hips (length wise)."

This output reflects the model's ability to extract key positive features from the review, such as design details and fabric quality, while omitting less relevant information like sizing preferences and measurements.

```
Selected Review (with question):
I love these, I had the previous iteration (or maybe two earlier?) and finally wore them out after
several years. They are definitely a light-weight training shoe so I wouldn't get them if you are
going to be doing a lot of running. But this also makes them extremely comfortable, I hadn't realized
how hot running shoes made my feet until I switched to these. Good support and my feet stay cool all
day, also noticeably light but well-made and long lasting.

I have worn them hiking in a pinch but don't really recommend it because the material is mesh-like and
will let a lot of dirt in. But if you mainly take cardio classes and/or do weight training or just
need something to wear around town look no further.

Auto-Generated Service Representative Response:
 True
```

9. LLM generation as customer service

To simulate a customer service representative's response, we selected the t5-small model due to its lightweight architecture and fast inference time. However, during testing with an example input “Ugh... way too large – is this for a man?” the model responded with “True,” which highlights its limitations in handling nuanced, conversational queries. This suggests that t5-small may not be well-suited for generating contextually appropriate dialogue in customer support scenarios.

```
Review 1 Original:
I really liked this shirt when I received it.  It is a really nice top.  I ordered the brown one.  The color of the shirt really
looks nice with the floral sleeves. The surprise is that the sleeves are not T-shirt material.  They are of a good quality chiffon.
The knit fabric is good quality, also.<br /><br />I am 5’4” and weigh about 172 lbs.  I am a 36 DDD.  I ordered an XXL.  It is
slightly on the big size.  It looks good, I would just prefer it a bit more form fitting.  The shirt covers my hips (length wise).

Review 1 Summary:
the color of the shirt really looks nice with the floral sleeves . the shirt is of a good quality chiffon . it covers my hips
(length wise).
--------------------------------------------------------------------------------
Review 2 Original:
If you are large chested, this isn’t for you.  The pleated bust area, did not cover my chest.  The bottom of the pleats should be
under the bust and were not.  It would be difficult to wear a bra with this dress.  The sides (arm pit area) of my bra showed.  I
hate my bra showing.<br /><br />The color was nice.  I ordered the light blue color.  The material is soft, but kind of thin.<br /
><br />I am 5’4” and weigh about 172 lbs.  I am a 36 DDD.  The dress basically fit as expected.  It was the correct size.  I
ordered an XL.
--------------------------------------------------------------------------------
Review 2 Original:
If you are large chested, this isn’t for you.  The pleated bust area, did not cover my chest.  The bottom of the pleats should be
under the bust and were not.  It would be difficult to wear a bra with this dress.  The sides (arm pit area) of my bra showed.  I
hate my bra showing.<br /><br />The color was nice.  I ordered the light blue color.  The material is soft, but kind of thin.<br /
><br />I am 5’4” and weigh about 172 lbs.  I am a 36 DDD.  The dress basically fit as expected.  It was the correct size.  I
ordered an XL.

Review 2 Summary:
the pleated bust area did not cover my chest . the bottom of the pleats should be under the bust and were not . I ordered the light
blue color .
--------------------------------------------------------------------------------
```

10. Final conclusion.

In this study, four sentiment analysis models—VADER, TextBlob, Logistic Regression, and SVM—were compared to evaluate their performance in classifying product reviews into positive, neutral, and negative sentiments. According to the findings, although VADER and TextBlob can detect positive sentiment effectively, they have severe shortcomings in detecting neutral and negative sentiments. Specifically, VADER did well in detecting positive sentiment with recall 0.93 but did not do well with neutral (recall: 0.21) and negative sentiments (recall: 0.45). TextBlob also had high recall for positive sentiment (0.96) but low recall for neutral reviews (0.06) and negative reviews (0.38).

In contrast, Logistic Regression and SVM showed more balanced performance on all three classes of sentiment. These models were generally more precise (0.64 for Logistic Regression and 0.65 for SVM) and performed better than the lexicon-based models at identifying negative sentiment. SVM, specifically, had a high recall of 0.68 for negative sentiment, reflecting its excellence in identifying negative reviews.

Since they differentiate between negative and neutral sentiments more accurately, Logistic Regression and SVM are better for applications where strong sentiment classification is necessary across all sentiment classes.

In conclusion, while VADER and TextBlob remain effective at general sentiment classification tasks, and in particular for positive sentiment, the employment of Logistic Regression or SVM is recommended when performing finer-grained tasks, particularly where accurate classification of negative and neutral sentiments are vital. Marginally greater improvements can be achieved by combining lexicon-based and machine learning-based models, which are able to reinforce the overall classification, particularly in picking up subtle differences in sentiments.

11. Assumptions.

- **Accuracy of Sentiment Labels**: The sentiment labels (positive, neutral, negative) in the dataset are assumed to be reliable and accurately reflect the sentiment of the reviews.
- **Pre-processing Impact**: It is assumed that pre-processing steps (lowercasing, removing stopwords and punctuation) effectively cleaned the data and improved sentiment classification.
- **Random Sampling**: The random sample of 1000 reviews is assumed to be representative of the full dataset, providing accurate insights into model performance.
- **Lexicon Model Limitations**: It is assumed that **VADER** and **TextBlob** struggle with detecting contextual nuances, such as sarcasm or mixed sentiments, which may lead to misclassification.
- **Rating-Based Sentiment Labels**: The sentiment labels based on ratings (4-5 = positive, 3 = neutral, 1-2 = negative) are assumed to reflect the overall sentiment accurately.
- **Dataset Balance**: The dataset is assumed to have a reasonable balance of positive, neutral, and negative reviews, allowing the models to generalize effectively.
- **Review Length**: It is assumed that review length does not significantly impact the models' ability to classify sentiment, as both short and long reviews were handled in the analysis.
- **Feature Engineering Effectiveness**: The feature extraction techniques (e.g., bag-of-words, TF-IDF) are assumed to be effective in transforming review text into meaningful features for sentiment classification.
- **Data Integrity**: The data is assumed to be clean, with minimal missing or inconsistent values after pre-processing.

Future Improvements :

- **Contextual Models**: Advanced models (e.g., **BERT**) could improve sentiment detection by understanding context and nuances.
- **Hybrid Models**: Combining lexicon-based methods with machine learning models (e.g., **SVM**) could enhance performance for all sentiment categories.
- **Data Imbalance Handling**: Techniques like **oversampling** or **class weighting** could improve detection of **negative** and **neutral sentiments**.
- **Improved Feature Engineering**: Using **word embeddings** or **deep learning** for feature extraction could boost model performance.

References:

- Amazon Review Data (2018). https://nijianmo.github.io/amazon/index.html
- Valence Aware Dictionary and Sentiment Reasoner (VADR) model. https://github.com/cjhutto/vaderSentiment
- TextBlob model. https://textblob.readthedocs.io/en/dev/quickstart.html
- **Logistic Regression for Sentiment Analysis**:
  Peng, R. D., & Matsui, E. (2015). *Logistic Regression*. In *Data Science for Business* (pp. 175-200). O'Reilly Media.
- **Support Vector Machines (SVM) for Sentiment Analysis**:
  Cortes, C., & Vapnik, V. (1995). *Support-vector networks*. Machine Learning, 20(3), 273-297.
  URL: https://link.springer.com/article/10.1007/BF00994018

Appendix 1

Week	Date Range	Milestone / Focus	Deliverables / Activities	Team Involvement
Week 1	Jan 8 – Jan 14	Course Intro & Group Formation	Team formed; review of project guidelines	All members attended first meetings
Week 2	Jan 15 – Jan 21	Dataset Familiarization	Dataset format (JSONL/CSV) explored; early inspection of structure and fields	Individual exploration
Week 3	Jan 22 – Jan 28	Data Cleaning – Phase 1	Removed duplicates, handled missing values, initial visualizations	Group sync on preprocessing rules
Week 4	Jan 29 – Feb 4	Lexicon-based Sentiment Models – VADER & TextBlob	Preprocessing `reviewText`; Sentiment labeling based on rating	Alireza led coding, others contributed validation
Week 5	Feb 5 – Feb 11	Lexicon Evaluation + Export for Phase 1	Accuracy, F1, confusion matrices; saved processed reviews	Results reviewed together
Week 6	Feb 12 – Feb 18	TF-IDF Text Representation Setup – Phase 2	Cleaned text, vectorized using TfidfVectorizer	Alireza coded; group reviewed feature size/quality
Week 7	Feb 19 – Feb 25	ML Pipeline (Logistic Regression, SVM) Setup	Train-test split (70/30 stratified); Model training on TF-IDF features	Alireza trained models; others handled test evaluation
Week 8	Feb 26 – Mar 3	ML Model Evaluation & Confusion Matrix Plotting	Model accuracy, precision, recall, F1 scores; plotted train/test confusion matrices	Joint result analysis
Week 9	Mar 4 – Mar 10	Begin Emotion-Aware Recommendation Research	Studied Moshfeghi et al. (2011); explored review emotion modeling with NRCLex	Alireza proposed solution, shared research insights
Week 10	Mar 11 – Mar 17	Emotion Score Computation & Rating Adjustment	Scaled raw ratings using positive/negative emotion ratio	Full pipeline built by Alireza

Week 11	Mar 18 – Mar 24	KNN Recommender System Using Adjusted Ratings	Built user-item matrix and tested KNN recommendations	Recommender results shared with group
Week 12	Mar 25 – Mar 31	LLM-Based Review Summarization & Q&A	Used T5-small model to summarize and simulate customer service responses	Results integrated by Alireza; evaluated by group
Week 13	Apr 1 – Apr 7	Final Report Drafting & Presentation Planning	Sections 1–10 written, citations added, visualizations integrated	Report work split across group
Week 14	Apr 8 – Apr 15	Final Submission & Presentation Delivery	Final edits, printed submission, rehearsed presentation; appended meeting logs and project plan	All members contributed to delivery and presentation

Appendix 2

Meeting register

Date	Time	Participants	Subjects Discussed	Assignments
01.30.25	6:00 pm - 6:30 pm	Alireza Farkhondeh Vishkasogh, Leor Gerber, Artem Kamov, Arash Karimi, Harsh Kumar	Data exploration and preprocessing steps	Everyone should complete Data exploration and preprocessing steps individually.
02.06.25	6:00 pm - 6:30 pm	Alireza Farkhondeh Vishkasogh, Leor Gerber, Artem Kamov, Arash Karimi, Harsh Kumar	Data exploration and preprocessing steps update	Everyone collaborated on Data exploration and preprocessing steps update during the meeting.
02.13.25	6:00 pm - 6:30 pm	Alireza Farkhondeh Vishkasogh, Leor Gerber, Artem Kamov, Arash Karimi, Harsh Kumar	The rest of the coding steps for phase 1 and Report requirements	Everyone should complete the rest of the coding steps individually. Everyone should collaborate on the report. Details are discussed in the group chat.
02.26.25	6:00 pm - 6:30 pm	Alireza Farkhondeh Vishkasogh, Leor Gerber, Artem Kamov, Arash Karimi, Harsh Kumar	The best code selection, ppt slides parts, and split of the roles for the presentation	Everyone should complete the rest of the coding steps individually. Everyone should collaborate on the ppt slides. Details are discussed in the group chat.
03.03.25	6:00 pm - 6:30 pm	Alireza Farkhondeh Vishkasogh, Leor Gerber, Artem Kamov, Arash Karimi, Harsh Kumar	Finalizing model evaluation results and report outline	Everyone should complete their individual model evaluations. Collaborate on the report and outline.
03.10.25	6:00 pm - 6:30 pm	Alireza Farkhondeh Vishkasogh, Leor Gerber, Artem Kamov, Arash Karimi, Harsh Kumar	Review results and discussion of presentation slides	Everyone should finalize their sections of the report. Collaborate on the final presentation slides.

03.17.25	6:00 pm - 6:30 pm	Alireza Farkhondeh Vishkasogh, Leor Gerber, Artem Kamov, Arash Karimi, Harsh Kumar	Review of final report and preparation for submission	Everyone should finalize the report and presentation. Prepare for submission.