CS 349 Final Project

CDs & Vinyls Dataset: Task 2

Ian Shi

Task 1: Feature Engineering

1 Unique Reviews
Total number of unique reviews each product received

2 Mean Upvote Rate
Average number of upvotes across the reviews per product

3 Verification
Proportion of reviews submitted by verified users per product

4 Earliest Review
Provides timestamp for the earliest review each product received

5 Image Percentage*
Proportion of reviews per product that contained an image

6 Sentiment
Average sentiment across product reviews and summaries

7 Compound
Mean compound sentiment of each review/summary for a specific product

8 Positive
Mean positive sentiment of each review/summary for a specific product

9 Negative
Mean negative sentiment of each review/summary for a specific product

10 Sentiment Threshold
Overall sentiment of a product based on its reviews/summaries (0, 1).

11 Sentiment Ratio
Ratio of each products positive to negative sentiment

12 Review vs Summary
Each products review vs summary sentiment ratio

*This feature was bugged in task 1, fixed in task 2

Task 1: Hyperparameter Optimization

Recursive Feature Elimination

- Sklearn’s RFECV function was used to to recursively test which features chosen during our feature engineering step should be prioritized in our model
- Each feature is tested on a 10-fold cross-validation
- Feature rankings are recorded, giving us a list of features for each model that should be used

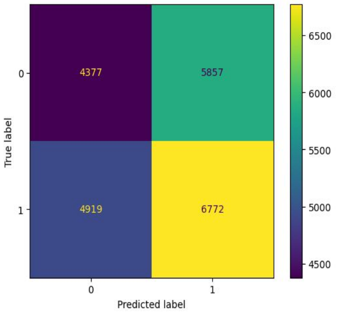
Grid Search

- Sklearn’s GridSearchCV function was used to iteratively test different hyperparameters for each relevant model
- Each hyperparameter pair was tested on a 10-fold cross-validation, with their mean F1 score recorded
- Collected data was exported to a .csv file where we could analyze which set of hyperparameters gave us the best results

Task 1: Model Performance

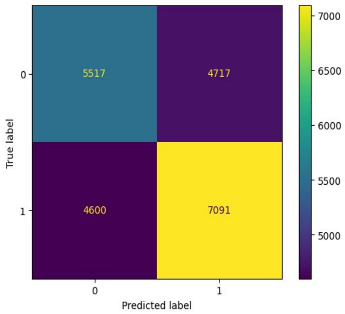
K Nearest Neighbors

N_neighbors: 9 | weights: uniform



Precision: 0.54 | Recall: 0.58
F1 Score: 0.56

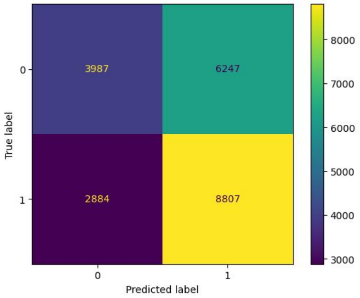
Gaussian NB



Precision: 0.60 | Recall: 0.61
F1 Score: 0.60

Decision Tree

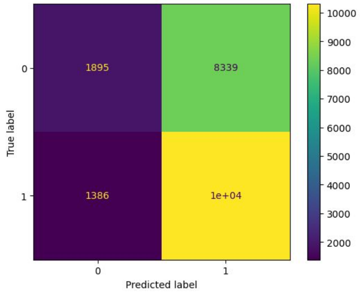
Criterion: entropy | max_depth = 3



Precision: 0.59 | Recall: 0.75
F1 Score: 0.66

SVM

Kernel: rfb

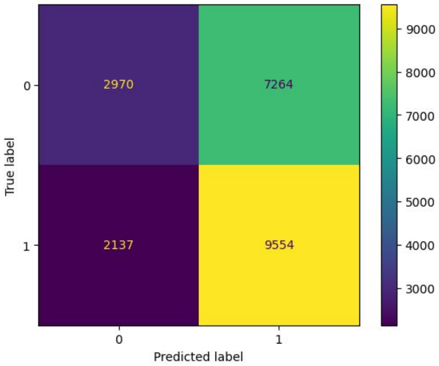


Precision: 0.55 | Recall: 0.88
F1 Score: 0.68

Random Forest

max_depth: 1 | n_estimators: 28

Features: ['verified', 'rev_posSentiment', 'rev_negSentiment', 'summ_negSentiment', 'rev_posNegRatio', 'summ_posNegRatio', 'summToRev']



Precision: 0.57 | Recall: 0.82
F1 Score: 0.68

Task 2: Feature Engineering

1

Length of Review

Average number of words contained in each review per product

2

Normalized Time

Issue with task 1 “earliest review” feature: scale was too high. Normalizing prevents that*

3

Review Period

Time between first and last review of product (normalized).

4

Total Reviews

Total number of raw reviews per product (task 1 contained total unique reviews)

5

Max Upvotes

Maximum number of upvotes on a product’s review page

*Normalizing/scaling done using sklearn MinMaxScale()

Task 2: New Algorithms & Methods

1 **New Feature Generation**
Task 1 algorithms using new features

2 **Logistic Regression**
Logistic regression algorithm using scikit learn

3 **Adaboost**
Adaboost algorithm using scikit learn

4 **Late Fusion**
Ensemble method using scikit learn

Task 2: Hyperparameter Optimization

Recursive Feature Elimination

- Sklearn's RFECV function was used to to recursively test which features chosen during our feature engineering step should be prioritized in our model
- Each feature is tested on a 10-fold cross-validation
- Feature rankings are recorded, giving us a list of features for each model that should be used
- For task 2, we added 5 new features and fixed a broken feature (image) from task 1
- Also created RFE functions for classifiers w/o RFECV support

Grid Search

- Sklearn's GridSearchCV function was used to iteratively test different hyperparameters for each relevant model
- Each hyperparameter pair was tested on a 10-fold cross-validation, with their mean F1 score recorded
- Collected data was exported to a .csv file where we could analyze which set of hyperparameters gave us the best results

Randomized Search

- Sklearn's RandomizedSearchCV function was used as another optimization method
- Compared to Grid Search (which tests every combination), Randomized Search randomly selects combinations from a given subset. This is beneficial when testing on larger datasets like ours
- Each hyperparameter pair was tested on a 10-fold cross-validation, with their mean F1 score recorded

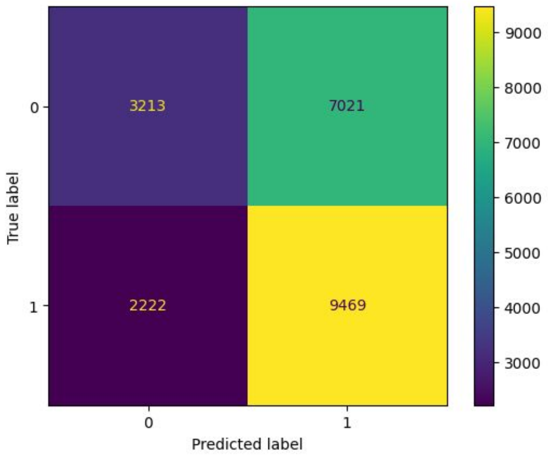
Task 2: Model Performance

Part 1: Rerun Task 1 models & hyperparameter optimization with new and fixed features

Random Forest

max_depth: 1 | n_estimators: 28

Features : all

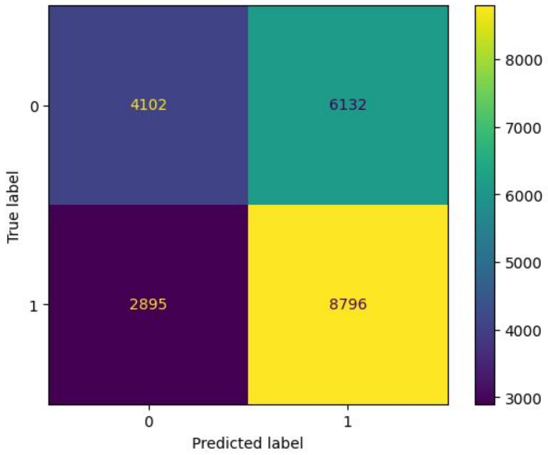


Precision: 0.57 | Recall: 0.81 | F1 Score: 0.67

Decision Tree

max_depth: 5

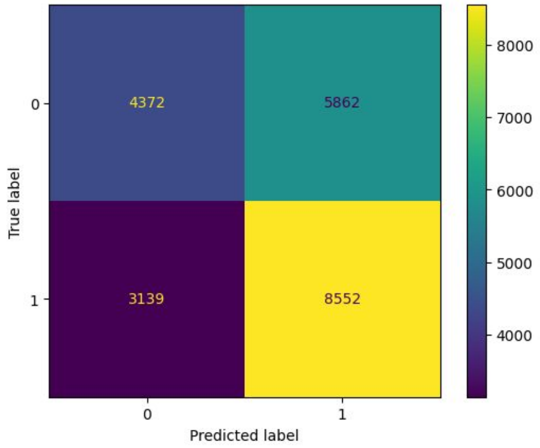
Features: ['verified', 'rev_negSentiment', 'summ_negSentiment', 'total_reviews', 'norm_time_diff']



Precision: 0.59 | Recall: 0.75 | F1 Score: 0.66

Gaussian NB

Features: ['vote', 'verified', 'rev_Sentiment', 'rev_posSentiment', 'rev_negSentiment', 'summ_negSentiment', image', 'rev_posNegRatio', 'normalized_time', 'max_upvote', 'norm_time_diff']



Precision: 0.59 | Recall: 0.73 | F1 Score: 0.66

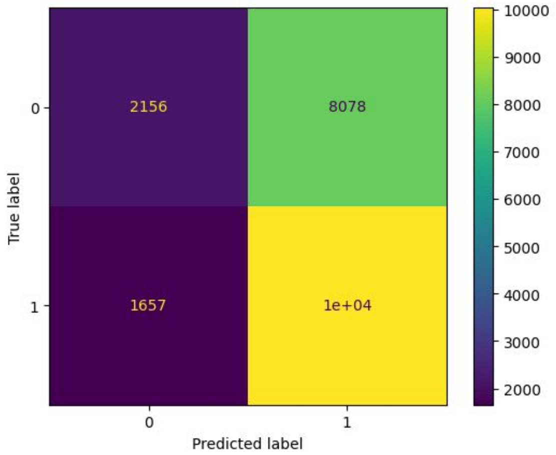
Task 2: Model Performance (cont.)

Part 2: Implement new classifiers & algorithms

Logistic Regression

fit_intercept: False | solver: 'sag' | multi_class: 'multinomial'

Features : ['reviewerID', 'vote', 'verified', 'rev_Sentiment', 'summ_compSentiment', 'rev_posSentiment', 'summ_posSentiment', 'rev_negSentiment', 'summ_negSentiment', 'image', 'rev_posNegRatio', 'summ_posNegRatio', 'summToRev', 'normalized_time', 'max_upvote', 'av_word_count', 'total_reviews', 'norm_time_diff']

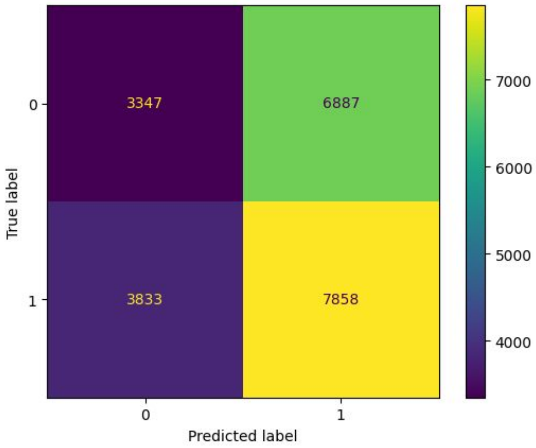


Precision: 0.55 | Recall: 0.86 | F1 Score: 0.67

Adaboost

classifiers: Gaussian NB, Decision Tree, Random Forest, Logistic Regression

Features: all



Precision: 0.53 | Recall: 0.67 | F1 Score: 0.59

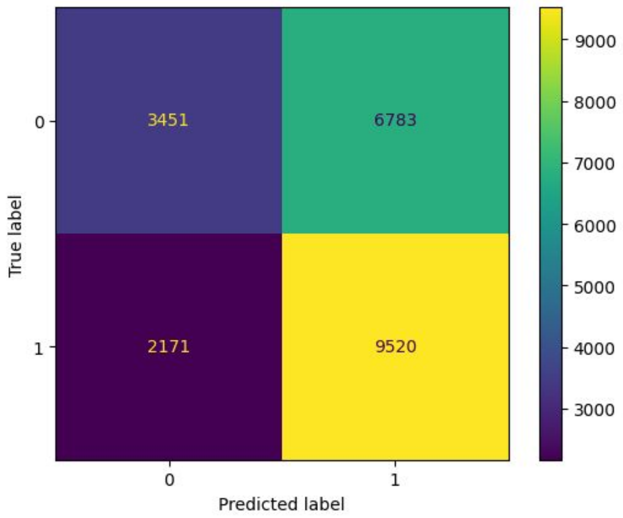
Task 2: Model Performance (cont.)

Part 3: Late Fusion with VotingClassifier() to create final classification model

Late Fusion

estimators = random forest, decision tree, logistic regression

Features: all



Precision: 0.58 | Recall: 0.61 | F1 Score: 0.69

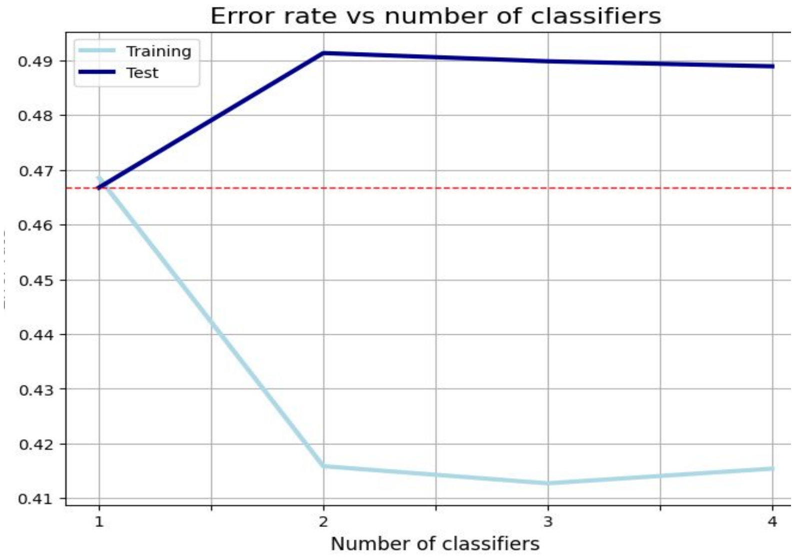
Areas of Concern

- Overall, our training model performance was again not as optimal as we would have liked (unable to achieve an F1 score above 0.7, relatively low precision)
- These may have to do with suboptimal features
  - If given more time on this project, we would have liked to explore more feature options
- Our implementation of Adaboost ran into issues with changing the number of features used with each classifier, preventing us from using our results from feature optimization
  - This may have negatively impacted the ability of Adaboost to improve its predictions
  - Some classifiers have particularly inaccurate results when using all features
    - E.g., logistic regression sorts all vectors into 1
  - More information on the next slide

Areas of Concern (cont.)

Aside: Adaboost

Adaboost Error



- After implementing Adaboost for testing multiple classifiers, we calculated error rate based on the number of classifiers included in the boosting
- Data was split into training (70%) and testing (30%)
- While performance improved when training and boosting the classifiers, it appears the boosting actually increased error while boosting on the test data

Next Steps

Neural Networks & Deep Learning

- Parts 1 & 2 of the final project were restricted to binary classifiers, which performed mediocre on the task of predicting “awesome” vs “not awesome” products based on the information given
- In the individual portion, exploring different neural network and deep learning algorithms--while adding complexity--could help the performance of our model

Sentiment Analysis

- Sentiment analysis was one of the biggest “black-box” unknowns during our preprocessing stage
- Trying different models such as TF-IDF, or spaCY instead of nltk could product different or more accurate results

Feature Engineering

- The features we chose gave mean results that made it hard to distinguish between “awesome” and “not awesome” products. While task 2 refined and added new features, it will be important to continue to develop and monitor the best features to extract from our dataset