Explorations, Patterns, and Observations

1. Basic Data Exploration

Initial exploration revealed several key patterns in the dataset:

- **Score Imbalance**: Scores were heavily skewed towards higher ratings (4 and 5), meaning the model could easily become biased toward these frequent classes.
- **Text Length and Sentiment Correlation**: Positive reviews often contained more descriptive text, suggesting that lengthier reviews correlated with higher scores. This pattern indicated that capturing review length could assist in distinguishing between positive and neutral feedback.
- **Time-Based Sentiment Variation**: Reviewing historical data revealed temporal patterns where older reviews tended to have more polarized sentiments, while recent reviews often had nuanced or neutral feedback. Retaining this time-based context could help in understanding shifting sentiment trends.

2. Preliminary Models and Feature Observations

Initially, I tested models like **K-Nearest Neighbors**[1] and **Random Forest**[2,3] using basic metadata features (e.g., Helpfulness and Review Age). Although these models achieved modest accuracy, they struggled to capture sentiment-based nuances due to a lack of text-focused features.

Pattern Observed: The absence of text-based features limited sentiment analysis, reinforcing the need for a deeper understanding of textual data. With insights from Biyani's book[4] and advice from a TA, I introduced **TF-IDF vectors**[5] and sentiment-related features, which significantly improved model depth and accuracy.

Feature Engineering Process Based on Patterns

1. Core Features and Their Implications

- **Helpfulness Ratio**: Calculated as HelpfulnessNumerator / HelpfulnessDenominator, this ratio was a key metric showing how users perceived each review's helpfulness. Reviews with higher helpfulness tended to be more positively rated, helping the model better differentiate between positive and neutral ratings.
- **Review Age**: Derived from the "Time" feature, this captures the review's age in days. Observing that newer reviews showed more polar sentiment, I retained this feature to give the model context on sentiment shifts over time.

2. Text-Derived Features

Textual features were critical for capturing nuanced sentiment:

- **TF-IDF Components**[5]: I vectorized the text using TF-IDF (3000 features), which allowed the model to weigh words based on their importance to sentiment. Using

Truncated SVD[6] to reduce TF-IDF to 100 components retained essential sentiment while keeping processing efficient.
- **Sentiment Keywords and Polarity Ratios**: Positive and negative word counts (e.g., "excellent" vs. "poor") were calculated to form polarity ratios, reflecting the relative weight of sentiment-laden words. Coupled with **Review Length**, this approach helped distinguish mid-range scores (e.g., 3) from more polarized ratings, providing the model with a balanced view of each review's sentiment.

These features enhanced the model's ability to capture patterns in sentiment effectively. For example, a high polarity ratio and review length tended to indicate a positive rating, while a low polarity ratio and short review length correlated with lower scores.

Model Development and Optimization Attempts Based on Observed Patterns

Attempt 1: Simple Models with Core Features

- **Models**: Random Forest[3] and K-Nearest Neighbors[1]
- **Features**: Core metadata features like Helpfulness, Review Age, and Review Length
- **Observation**: While these models performed moderately well, they struggled without text-based features, confirming the necessity for richer features to capture sentiment depth.

Attempt 2: Incorporating Text Features and Dimensionality Reduction

After observing how core metadata alone was insufficient, I introduced **TF-IDF vectors**[5] and used **SVD**[6] to reduce dimensionality to 100 components. Exploring boosting methods (inspired by both class discussions and literature[7]), I implemented **XGBoost**[8] and **Gradient Boosting**[9].

- **Observation**: The inclusion of text features markedly improved sentiment classification. However, it increased both data preprocessing and training time, especially for XGBoost. SVD was instrumental in managing complexity without compromising performance.

Attempt 3: Stacking Models for Enhanced Prediction

Recognizing the potential in ensemble methods to capture a range of patterns, I researched ensemble methods[11] and implemented **Stacking Ensemble**[12] using **XGBoost**[8], **Random Forest**[3], **SVC**[10], and **Gradient Boosting**[9] as base models, with **Logistic Regression**[13] as the meta-learner.

- **Observation**: The stacking ensemble produced a noticeable improvement in accuracy, particularly for minority classes. However, this model significantly increased training time, especially when handling text-based features.

Final Model Configuration and Pattern-Based Refinements

To achieve a balance between runtime efficiency and high accuracy, I applied targeted optimizations based on previously observed patterns:

1. **Balanced Downsampling**: I reduced the dataset to 5,000 rows, which retained class diversity while significantly decreasing runtime.
2. **Narrowed Parameter Ranges**: Using **RandomizedSearchCV**[14] with refined parameter ranges (based on earlier experiments) allowed for quicker hyperparameter tuning while maximizing model effectiveness.
3. **Parallel Processing**: With help from a discussion in our Discord group, I enabled parallel processing by setting `n_jobs=-1` in Random Forest and XGBoost, optimizing resource use and minimizing tuning time.

The final model configuration included four base models:

- **XGBoost**[8]: Effective at uncovering complex relationships, particularly when combined with polarity ratios and core metadata.
- **Random Forest**[3]: Good for handling high-variance features such as polarity ratios and word counts, capturing important patterns for extreme reviews.
- **SVC**[10]: Effective for distinguishing middle scores, which were less polarized and required a linear separator to classify accurately.
- **Gradient Boosting**[9]: A lighter alternative to XGBoost that captured subtler interactions, particularly in sentiment-driven features.

Logistic Regression as the meta-learner averaged out overconfident predictions from individual models, providing stable and balanced ensemble predictions. **Stratified Cross-Validation**[15] ensured balanced accuracy across classes, leading to reliable model performance.

Key Insights and Lessons Learned from Patterns

1. **Feature Scaling's Importance**: Unscaled features, such as polarity ratio and review length, destabilized SVC and tree-based models. Standard scaling resolved these issues, ensuring that all features contributed proportionally to predictions.
2. **Sentiment Polarity and Review Length**: Longer reviews often corresponded to higher scores, particularly when the polarity ratio was high. This pattern was instrumental in including both **Review Length** and **Polarity Ratio** as indicators of sentiment.
3. **Class Imbalance Addressed with Balanced Sampling**: Class imbalance caused a bias toward higher scores. Observing this, I applied downsampling to ensure balanced representation, resulting in improved recall for underrepresented classes (e.g., scores 1 and 2).
4. **Choosing the Meta-Learner**: Logistic Regression's averaging effect minimized overconfidence from individual models, ensuring balanced predictions across all scores.

Final Evaluation and Results

The final model demonstrated a more robust performance across all score classes. Cross-validation accuracy confirmed consistency, with significant improvement in minority class recall, as shown in the confusion matrix. This validated the ensemble stacking approach, balanced sampling, and feature engineering to capture sentiment nuances effectively.

Confusion Matrix Analysis The matrix showed reduced misclassifications for minority classes due to sentiment features and balanced sampling, underscoring the value of combining structured and text-derived features to capture sentiment accurately.

Citations:

1. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
2. Parsons 2015 https://rstudio-pubs-static.s3.amazonaws.com/127992_a060e7d374d549998df02fc11ac8c334.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
4. Akshat Biyani 2023 https://careerfoundry.com/en/blog/data-analytics/sentiment-analysis
5. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
7. https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/
8. https://xgboost.readthedocs.io/en/stable/python/python_intro.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
10. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
11. Khadka https://dataaspirant.com/stacking-technique/
12. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html
13. https://www.geeksforgeeks.org/understanding-logistic-regression/
14. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
15. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html