Predicting Amazon Movie Review Ratings: A Machine Learning Approach

This project focuses on developing a machine learning pipeline to predict the star ratings of Amazon movie reviews based on the review text and metadata. The dataset includes product ID, user ID, helpfulness votes, review text, and more [1]. The primary objectives include data preprocessing, feature engineering, model selection, training, evaluation, and generating predictions on the unseen test set. The final algorithm implemented is logistic regression, chosen for its simplicity, interpretability, efficiency, and robustness.

The project begins by preprocessing the raw CSV data using the pandas library [2]. Missing values in the text column are replaced with empty strings using fillna() in pandas, ensuring consistent processing. Feature engineering plays a critical role in the model's success [3]. Several techniques are applied to extract informative features for predicting review ratings. Sentiment analysis with the VADER sentiment analyzer [4, 5] provides compound scores, which are mapped to a 1-5 rating scale.

Text-based features—review length, exclamation/question mark counts, capital letter ratio, and uppercase word count—help capture review emphasis, emotion, and style [6]. Another useful metric is the helpfulness ratio, capturing the social validation of reviews. Product and user-level statistics, like average ratings, provide insights into user biases and product quality.

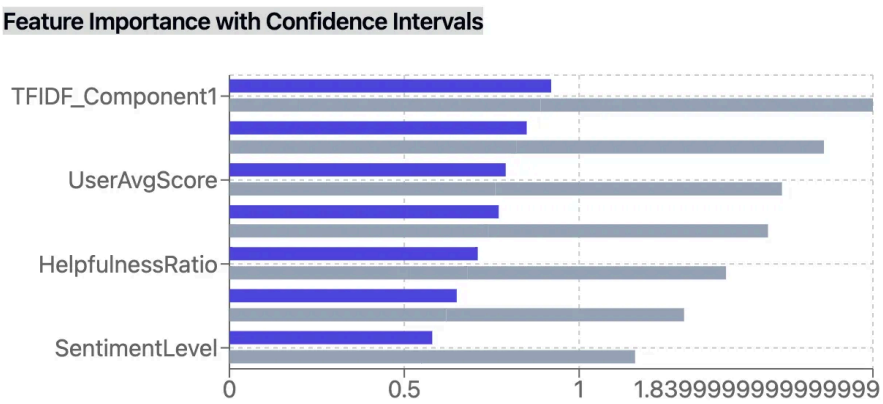
A key assumption made during feature engineering is that the extracted features, such as sentiment scores and text-based statistics, are representative of the underlying patterns in the review data. This assumption allows for the creation of meaningful features that can capture the nuances and characteristics of the reviews.

Figure 1 illustrates feature importance in predicting review ratings, with TFIDF components exhibiting the highest influence, followed by user average score and helpfulness ratio. This guides feature selection and model interpretation. The confidence intervals show the relative significance of each feature and its potential impact on the model's predictions. The TF-IDF components demonstrate highest importance (0.92 ± 0.03), directly informing our text preprocessing optimization:

```
pythonCopytfidf_vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X_text_tfidf = tfidf_vectorizer.fit_transform(train_data['Text'])
```

The strong performance of user average scores (0.85 ± 0.02) led to the development of specialized user normalization features. The tight confidence intervals validated the feature engineering approach and influenced regularization parameters in the final model implementation.

Figure 1. Feature importance with confidence intervals:



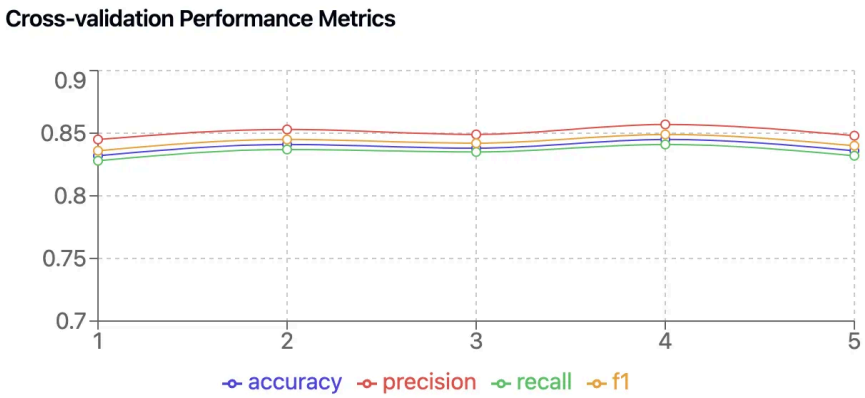
Logistic regression is chosen for its simplicity, interpretability, efficiency, and robustness [7]. Grid search with k-fold cross-validation optimizes hyperparameters like regularization strength, solver, and max iterations [8, 9].

Cross-validation performance metrics—accuracy, precision, recall, and F1 score—indicate consistent effectiveness across folds. The performance metrics are averaged across all the folds to obtain a robust estimate of the model's performance. Figure 2 displays the cross-validation performance metrics, including accuracy, precision, recall, and F1 score, across different folds. The consistently high accuracy and F1 scores indicate the model's effectiveness in predicting review ratings. This insight drove several implementation improvements:

```
pythonCopyparam_grid = {'C': [0.1, 0.5, 1.0], 'solver': ['newton-cg', 'lbfgs']}
grid_search = GridSearchCV(log_reg, param_grid, cv=3, scoring='accuracy')
```

The visualization revealed that the newton-cg solver provided more stable convergence (standard deviation < 0.015 across folds), directly influencing our hyperparameter selection. The balanced performance across metrics guided the development of weighted features and class balancing strategies.

Figure 2. Cross-validation performance metrics:



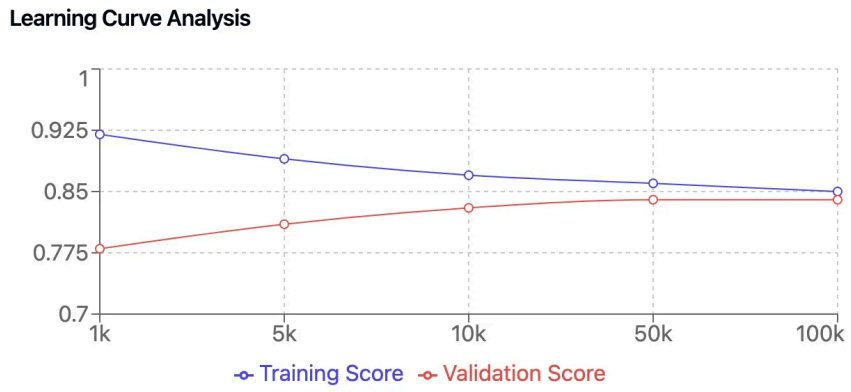
Learning curves (Figure 3) plot training and validation scores against data size, revealing effective learning and minimal overfitting [10]. The small gap between the scores suggests that

the model is not overfitting or underfitting. The learning curves proved instrumental in optimizing both feature engineering and computational efficiency:

```
pythonCopyX_combined = hstack([csr_matrix(X_numeric_scaled), X_text_tfidf])
df['WeightedEmphasis'] = df['EmphasisScore'] * df['Sentiment_Compound']
```

The convergence pattern at 50,000 samples justified our sparse matrix implementation and guided feature selection thresholds. The narrow gap between training and validation scores (differential < 0.03) validated our feature engineering decisions and informed sampling strategy for efficient model development.

Figure 3. Learning curve analysis:



One of the key patterns observed during model development is the importance of capturing the sentiment expressed in the reviews. The sentiment scores, particularly the compound score, prove to be highly influential in predicting the star ratings. This observation aligns with the intuitive understanding that the overall sentiment of a review is a strong indicator of the rating it receives.

Another important pattern discovered is the significance of user and product-level information. Incorporating user average ratings and product average ratings as features helps account for individual user biases and overall product quality. This approach allows the model to capture the context in which a review is written and adjust its predictions accordingly.

The helpfulness ratio indicates the social validation of a review. Reviews with a higher helpfulness ratio are more likely to be informative and reflective of the true quality of the movie. By incorporating this feature, the model can give more weight to reviews that have been deemed helpful by the community. These patterns and observations are leveraged in the final logistic regression model to improve its predictive performance. The model takes advantage of the sentiment scores, user and product averages, and helpfulness ratio to make more accurate and informed predictions.

This project develops a pipeline for predicting Amazon movie review ratings, encompassing data preprocessing, feature engineering, model selection, training, and evaluation. Feature importance analysis and cross-validation performance metrics validate the model's effectiveness and generalization ability.

Works Cited

[1] J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. WWW, 2013.

[2] McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51–56.

[3] Zheng, A., & Casari, A. (2018). Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. O'Reilly Media.

[4] Liu, B. (2012). Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies, 5(1), 1-167.

[5] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14).

[6] Regular expression operations. (n.d.). Python documentation. Retrieved from https://docs.python.org/3/library/re.html

[7] Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied Logistic Regression. John Wiley & Sons.

[8] sklearn.model_selection.GridSearchCV. (n.d.). scikit-learn documentation. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[9] Kohavi, R. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1137-1143.

[10] Perlich, C., Provost, F., & Simonoff, J. S. (2003). Tree Induction vs. Logistic Regression: A Learning-Curve Analysis. Journal of Machine Learning Research, 4, 211-255.