Overview:

This code implements a predictive modeling pipeline aimed at predicting the star ratings associated with user reviews from Amazon Movie Reviews using various features derived from the dataset. The process begins by loading the training and testing datasets, followed by extensive data preprocessing and feature engineering to enhance the predictive power of the model. To create this predictive modeling pipeline, I began by researching effective techniques for feature engineering and classification methods that do not involve deep learning, as specified in the competition guidelines. I drew upon resources such as academic papers, online tutorials, and documentation for libraries like TextBlob and more to understand the principles of feature extraction and sentiment analysis. Key features added include helpfulness metrics, such as the ratio of helpfulness votes, as well as user and product review history statistics that capture user behavior patterns and product performance. Textual features, including word counts and sentiment analysis using TextBlob, are also computed to capture the sentiment expressed in reviews. By applying these insights, I constructed new features from the dataset, such as helpfulness metrics, user and product review histories, and textual features derived from sentiment analysis. The thoughtful selection of these features was critical, as they provide valuable insights into user behavior and review quality, thereby enhancing the model's predictive capabilities. After augmenting the dataset, I utilized a GradientBoostingClassifier to train the model and evaluate its performance through accuracy metrics and confusion matrix visualizations.The iterative process of refining the features and model parameters was guided by a focus on achieving high classification performance while adhering to the competition's constraints. This experience not only deepened my understanding of machine learning concepts but also emphasized the importance of creativity in data analysis and the intelligent construction of features.This hands-on experience not only sharpened my technical skills but also deepened my understanding of the importance of thoughtful feature construction in data science.

Features:

In building the predictive model for movie review scores, I prioritized selecting features that I believed would significantly enhance the model's accuracy and predictive power. First, I included helpfulness metrics, such as the ratio of helpfulness votes, which provided insight into user engagement with the reviews. This feature helped highlight reviews that were not only informative but also resonated with other users, suggesting a higher likelihood of correlation with favorable ratings. Additionally, I created binary indicators for whether a review received helpfulness votes and calculated the difference between helpfulness numerator and denominator. This nuanced approach allowed the model to discern between reviews that were genuinely helpful and those that might be perceived as less so, ultimately improving the model's classification performance.

I also recognized the importance of understanding user and product review history, so I computed average and median scores for users and products. By incorporating these metrics, I was able to capture patterns in user behavior and product performance, which are crucial in predicting how a new review might be rated based on historical data. For example, if a user consistently rated a product highly, it was likely that their future reviews would also reflect that trend, providing valuable context for the model. Furthermore, I tracked the total number of reviews each user submitted, as higher engagement often correlates with more informed ratings.

To assess the content of the reviews themselves, I derived textual features such as word count and unique word count, which indicated the richness and diversity of the language used in the reviews. These features suggested that longer reviews with a varied vocabulary might convey more nuanced opinions, potentially leading to higher star ratings. After researching various sentiment analysis libraries, I chose Python's TextBlob library for evaluating the subjectivity and polarity of sentences. TextBlob stood out to me due to its simplicity and effectiveness in handling natural language processing tasks. It provides a straightforward interface for conducting sentiment analysis, allowing me to easily compute the polarity scores that indicate the emotional tone of the reviews. Specifically, I leveraged TextBlob's ability to return a polarity score ranging from -1 (very negative) to 1 (very positive), which helped me quantify the sentiment expressed in user reviews. This quantification was crucial for my model, as it transformed subjective text into numerical values that could be integrated into the feature set. This sentiment polarity feature quantified the emotional tone of each review, allowing me to capture the underlying feelings that could strongly influence star ratings. For instance, a review with a high positive sentiment was likely to align with a higher rating, while a negative sentiment would correlate with lower ratings. The ease of use and robust performance of TextBlob made it an ideal choice for my project, ultimately contributing to the model's success in accurately predicting movie review scores. However, I encountered performance challenges, as the sentiment analysis process was computationally intensive and time-consuming. While the results were satisfactory, I recognized that there might be alternative libraries or optimization techniques that could improve processing speed.

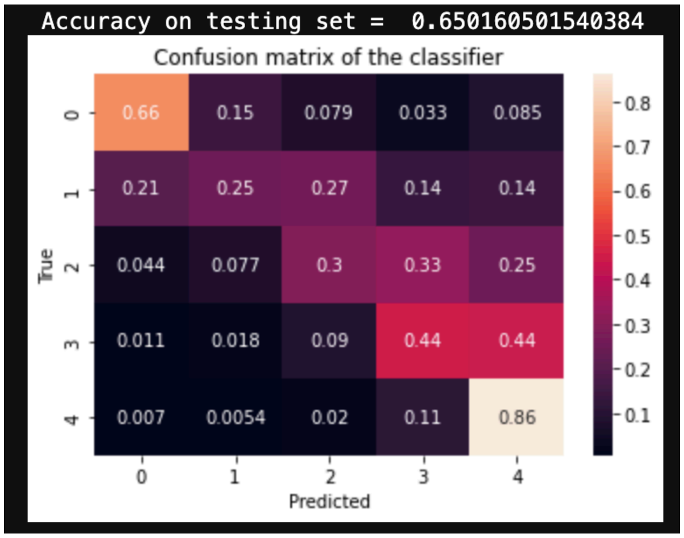
Ultimately, the combination of these thoughtfully selected features resulted in a model that performed well, yielding a classification accuracy I was satisfied with. By addressing different aspects of the data—from user engagement and review history to the content and sentiment of the reviews—I created a comprehensive feature set that provided a deeper understanding of the factors influencing movie review ratings. This structured approach not only improved the model's predictive capabilities but also reinforced my confidence in the insights drawn from the data.

Model Creation:

In developing the predictive model for movie review scores, I chose the GradientBoostingClassifier for its strong performance with complex datasets and its ability to manage various feature types effectively. Gradient boosting is an ensemble learning technique that builds multiple decision trees sequentially, where each tree corrects the errors of the previous ones. This iterative approach enhances accuracy compared to traditional single-tree classifiers by minimizing the loss function, which in turn improves the model's predictive capability. The classifier employs gradient descent to optimize predictions based on the gradient of the loss function, enabling it to learn from prior mistakes and make more accurate predictions over time. During my experimentation, I systematically adjusted several hyperparameters to optimize performance. I started with n_estimators, initially setting it to 200 under the assumption that more trees would enhance accuracy. However, this led to increased training times and overfitting, where the model performed well on training data but poorly on the test set. I ultimately settled on 130 estimators as a better compromise between performance and computational efficiency. Next, I experimented with the learning_rate, trying values from 0.05 to 0.3. A lower rate of 0.05 allowed for smoother convergence but required an impractical number of estimators, while a higher rate of 0.3 caused instability and erratic performance. After careful evaluation, I found that a learning rate of 0.2 balanced convergence speed and stability effectively. I also explored different max_depth values for the trees, ranging from 3 to 8. Deeper trees (max_depth of 8) captured complex patterns but increased the risk of overfitting, whereas a max_depth of 3 resulted in underfitting. Ultimately, I determined that a max_depth of 5 provided optimal complexity without overfitting. These systematic adjustments and evaluations were crucial in guiding my final hyperparameter selections, resulting in a robust model with balanced performance and efficiency.

Reflection:

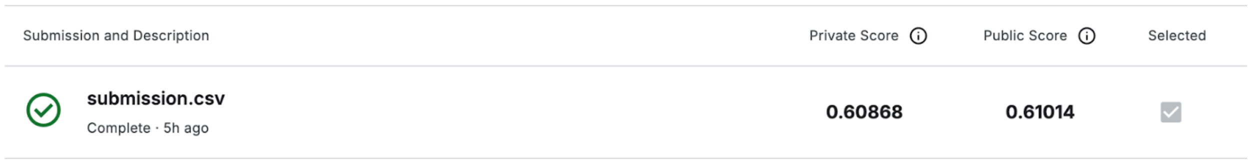
My journey in developing the predictive modeling pipeline for Amazon Movie Reviews has been both enlightening and challenging. I began by extensively researching feature engineering and classification techniques, which led me to recognize the importance of grouping data into smaller, more manageable parts. By segmenting reviews based on features such as helpfulness metrics and user behavior patterns, I was able to capture more nuanced insights that improved the model's accuracy. This approach allowed me to identify specific trends and correlations within subsets of data, ultimately enhancing my feature selection process. However, I encountered challenges in hyperparameter optimization for the GradientBoostingClassifier, where I learned to balance model complexity with computational efficiency. While my approach yielded satisfactory results, I realized that I could improve further by exploring data aggregation techniques and experimenting with more efficient libraries for sentiment analysis. Adopting advanced methods such as cross-validation for hyperparameter tuning and utilizing ensemble techniques would also help refine my model. Moving forward, I aim to deepen my understanding of these strategies and experiment with alternative machine learning frameworks to enhance performance.



```
Accuracy on testing set =  0.650160501540384
```

Strong performance for high ratings (4-5 stars): 86% accuracy for 4-star and 44% for 3-star reviews - Moderate/low performance for neutral ratings: 25% accuracy for 2-star reviews. Lower accuracy for extreme negative ratings

Kaggle Submission:



Submission and Description	Private Score	Public Score	Selected
submission.csv Complete · 5h ago	0.60868	0.61014	✓

Works Cited:

GradientBoostingClassifier. (n.d.). Scikit-Learn. Retrieved October 28, 2024, from https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

TextBlob: Simplified Text Processing — TextBlob 0.18.0.post0 documentation. (n.d.). Retrieved October 28, 2024, from https://textblob.readthedocs.io/en/dev/

Python Software Foundation. (n.d.). *Pickle — Python 3.10.12 documentation*. https://docs.python.org/3/library/pickle.html

McKinney, W. (2021). *Pandas: A foundational library for data analysis in Python*. In *Python for Data Analysis* (2nd ed.). O'Reilly Media. https://pandas.pydata.org/

Waskom, M. (2021). *Seaborn: Statistical data visualization*. https://seaborn.pydata.org/

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering, 9*(3), 90-95. https://doi.org/10.1109/MCSE.2007.55

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Blondel, M., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830. http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature, 585*, 357-362. https://doi.org/10.1038/s41586-020-2649-2

Abernethy, J. (2013). TextBlob: Simplified Text Processing. https://textblob.readthedocs.io/en/dev/