1 Introduction:

For this Kaggle competition, the goal was to predict the star rating associated with user reviews, which required using both text and numeric data. Throughout this project, I implemented various techniques and models that were directly inspired by key concepts from lectures such as feature extraction, model evaluation, and ensemble methods. Additionally, I incorporated my own research and widely accepted practices of machine learning to enhance my model performance. In this report, I will outline the methods I used and the patterns I identified during the analysis.

2 Data Processing and Feature Engineering:

2.1: TF-IDF Vectorization:

In order to handle the text data, such as user reviews, I used TF-IDF (Term Frequency-Inverse Document Frequency) to convert the text into numerical vectors. This method ensures the model captures the importance of its word and this concept was discussed in the lecture about Latent Semantic Analysis. However, this specific implementation of TF-IDF was inspired by ChatGPT. I asked it to help me integrate vectorization into my code.

2.2: Handling Numeric Data:

The numeric features initially caused significant error in my code due to the different scales between the text and numeric data. After looking at the performance, I determined that I should implement scaling features and so I scaled using StandardScaler. This is a method commonly used in machine learning for normalizing data and mentioned in the Scikit-learn Documentation in there preprocessing data section.

3 Model Selecting and Tuning

3.1: Naive Bayes:

I tried to use the Naive Bayes model discussed in lecture because of its simplicity and performance when dealing with text data. However, I struggled to get it to work when handling both textual and numerical features. Therefore I had to research other models that handled both textual and numeric data.

3.2: Logistic Regression:

Logistic Regression one of the models I utilized to classify the target variable based on the combination of text and numeric features. I used the TF-IDF vectorizer to convert the text data into numerical form, and then fed it into the Logistic Regression model alongside other numeric features like helpfulness ratings and review timestamps. Logistic Regression provided a good starting point to understand the relationship between features and the target. I compared this model along with the Multinomial Naive Bayes model, and Random Forest Classifier to find the model that provides me with the highest accuracy given the data. Ultimately it was the Logistic Regression model that provided me with the highest accuracy.

3.3: Random Forest Classifier:

Initially I utilized the default parameters but ChatGPT suggested that I implement grid search and fine tune hyperparameters. This process shows how ensemble methods can handle complex datasets with textual features and with numeric ones. The Random Forest Classifier also removed some overfitting issues encountered.

4 Model Evaluation:

4.1: Cross-Validation and Confusion Matrix:

Cross-Validation ensures that overfitting is prevented. I initially used a train/test split to evaluate my model, and I noticed that my results tended to change everytime I ran my model. ChatGPT suggested using cross-validation to provide more reliable performance estimates and helped in

ensuring that the model generalized well on previously unseen data. Confusion matrices for analyzing misclassification helped with determining where the model struggled and which classes had the most misclassifications.

4.2: Handling Class Imbalance:

One of the main challenges I encountered was class imbalances in my dataset. I experimented with oversampling to rebalance these classes. This method helped to improve my overall accuracy.

5 Conclusion

In this project I utilized TF-IDF, Naive Bayes, Decision Trees, Random Forests and model evaluation with cross-validation and confusion matrices. Although I struggled initially with scaling numeric features and using hyperparameter tuning I had to do my research with the help of ChatGPT, the lectures, and external resources to improve the accuracy of my model. This ultimately resulted in me deciding to utilize 3 different training models and choosing the one that provided me with the highest accuracy.

Works Cited

Bonnet, Alexandre. “Fine-tuning Models: Hyperparameter Optimization.” *Encord*, 22 August 2023, https://encord.com/blog/fine-tuning-models-hyperparameter-optimization/. Accessed 28 October 2024.

Jain, Sandeep. “Understanding the Confusion Matrix in Machine Learning.” *GeeksforGeeks*, 8 July 2024, https://www.geeksforgeeks.org/confusion-matrix-machine-learning/. Accessed 28 October 2024.

“LogisticRegression — scikit-learn 1.5.2 documentation.” *Scikit-learn*, https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed 28 October 2024.

“RandomForestClassifier — scikit-learn 1.5.2 documentation.” *Scikit-learn*, https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed 28 October 2024.

Singh, Sandeep. “Cross Validation and Hyperparameter Tuning: A Beginner's Guide.” *Medium*, 14 February 2023, https://medium.com/@sandeepmaths04/cross-validation-and-hyperparameter-tuning-a-beginners-guide-96d258eedee7. Accessed 28 October 2024.

“6.3. Preprocessing data — scikit-learn 1.6.dev0 documentation.” *Scikit-learn*, https://scikit-learn.org/dev/modules/preprocessing.html. Accessed 28 October 2024.