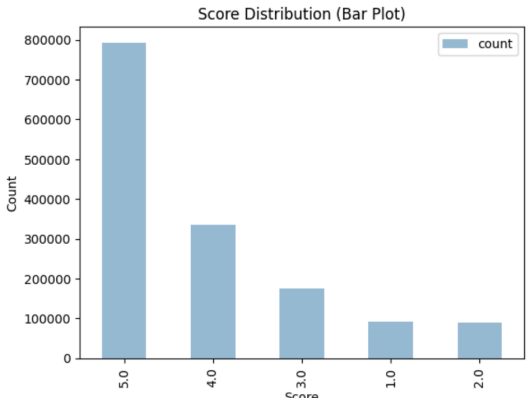
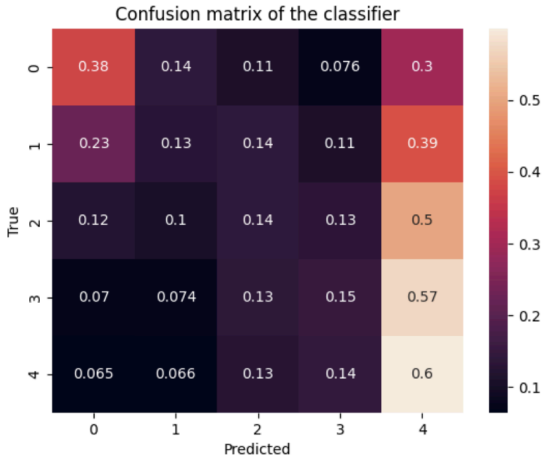
Lauren Kong

This project is supposed to predict the star ratings that is affiliated with reviews from users through Amazon Movie without the use of deep learning or neural networks. I decided to use K Nearest Neighbors. In order to run this dataset I decided to utilize GoogleColab and mounted my dataset to GoogleDrive due to the sizing of the file. To speed up the data preprocessing process, I also utilized pandarallel. This allowed the computations to be split across multiple CPU cores. To learn more about this process I used the Python Index Package website to help me, and this allowed the time it took to clean and preprocess the data to decrease significantly. This was specifically very helpful in such a large dataset like the one we are working in.

I started with looking through the data such as Product ID, user ID, helpfulness scores, and the star rating. To sort the data I proceeded with data preprocessing since it will significantly affect the performance of the prediction. To find the "score" which is the variable we are trying to predict with the highest accuracy. The qualitiy of our data will directly impact the accuracy of the mode. In the data, for the columns such as helpfulness numerator and helpfulness denominator I had to cover the missing values therefore I implemented "helpfulness" which is the ratio of the two variables of numerator and denominator. Since we cannot divide by zero, if the denominator is zero than we know the total value is zero. The first thing I looked at was this chart to see that there was an unbalanced distribution and a majority of scores were 5's.





Next, I conducted a detailed analysis of each data category to understand its relationship with the score and prediction accuracy. The key categories I focused on were the numerator, denominator, time, and helpfulness. These particular features were chosen because they appeared to have the greatest impact on review helpfulness, the timing of when the review was posted, and other contextual information that could significantly enhance the model's understanding and

predictions. By including features like the numerator and denominator, I aimed to quantify aspects of helpfulness and reliability, while features like time provided temporal context that might influence how relevant or valuable a review is perceived to be.

During this analysis I realized that not all the features were available in the test set. To help mitigate this I merged several columns from the training set to ensure consistency.This preprocessing step was important to make sure that both datasets have comparable information. This would then help the model make a more accurate and generalizable prediction. A consistent set between training and testing data decreases the risk of introducing bias or overfitting, thus improving the model's dependability if it is applied to new unseen data.

After preparing the data, I moved on to the process of selecting and training a model for generalization. For this midterm project, I decided to use the K-Nearest Neighbors (KNN) algorithm to predict the star ratings of reviews. I chose KNN because it is a relatively simple algorithm that makes fewer assumptions about the underlying data distribution. The simplicity of KNN allows for a straight forward understanding of how the many features contribute to the model's predictions allowing for valuable insights during this trial phase to make predicitions.To establish this baseline I began by setting the number of neighbors (n-neighbors) to 3, which allowed me to see the initial performance and accuracy of the model on the given dataset. KNN's performance mainly depends on the organization or relationships of features as it uses distance metrics between data points to determine similarities.It was also vital to ensure that features were well-prepared and that any inconsistencies were decreased as much as possible prior to training.

To evaluate the model's potential performance on unseen data I split the training dataset into two subsets which was 80% for training and 20% for testing. This 80 20 split allowed me to visualize how the model might behave when predicting for new data by providing a controlled environment for predicting its generalizability. By testing on a portion of the data I was allowed to notice any potential overfitting issues early and make sure that the model's predictions would be applicable to data beyond the original training set. This step gave me a better understanding of the model's strengths and weaknesses, allowing me to make any necessary adjustments for the data parameters or feature engineering.

The KNN model was slightly difficult was the initial missing values in X-train and y-train. To make sure everything was accurate, I concatenated the X-train and y-train, dropped any of the rows with missing values, and then split them into train and test variables. This helped keep everything organized and consistent. Furthermore, another challenge I faced was to make sure X-train and y-train were both consistent after merging and any actions that may have changed the data. To simplify this process, I combined both trains into a single dataframe and dropped any values that were not numerical. Lastly, I struggled with some columns not being in the test data. As mentioned briefly earlier, to fix this I was able to merge some missing features from the training set and filled the remaining gaps with default values of zero. I also used

StandardScaler to normalize numerical features since KNN can be more sensitive to different scales. This helped keep the distance calculations consistent and improved the model's overall performance.

The model ended up with an accuracy of around 40% on the validation set, which isn't as high as I would have liked. The challenge in this problem set was that KNN is not the best at understanding complex, context-heavy data, like the large training data set that I had. A deep learning model may have been better. To better understand where the model was going wrong, I plotted the confusion matrix. This model shows that higher ratings were often confused with neighboring ratings, which makes sense since the differences between a 4-star and a 5-star rating can sometimes be pretty minor, especially based on the information I was using. One potential issue was the imbalance in the training data which may have caused prediction inaccuracy. After doing research, in the future there are different functions such as SMOTE, which may help ensure that all classes are adequately represented during the training phase of the data allowing better representation and equality across all the classes. Increasing the accuracy proved to be a significant challenge for me as the relationships between the data points were sometimes subtle to me.

One of the big takeaways for me was how useful the helpfulness feature was. It gave the model a sense of how other users perceived the value of a review, which seemed to have a significant impact on accuracy. I chose to mostly focus on improving the stability of KNN's predictions. I also spent time experimenting with different values for n-neighbors to find a balance between underfitting and overfitting. In the end, using 3 neighbors provided a reasonable trade-off between accuracy and computational cost, and it worked pretty well for this dataset. Furthermore, through this assignment I have learned more about data normalization. Since KNN would rely on distance metrics, the features with larger scales may dominate the calculations which lead to biased predictions.

This project was a good opportunity to understand the strengths and limitations of simpler models like KNN, especially when dealing with real world data. I learned a lot about the importance of thoughtful preprocessing, feature engineering, and experimentation with parameters. These steps are vital for making traditional machine learning models perform as accurate as possible. I would also like to experiment with more advanced models like Random Forest or XGBoost which have come up during my research for my final project. I would be intrigued to see if they could potentially provide better accuracy and handle complex relationships in the data more effectively.

Citations

Pandarallel. PyPI. (n.d.). https://pypi.org/project/pandarallel/

Chatgpt. (n.d.). https://chatgpt.com/