Merai Dandouch

Introduction

Big data is dominating the world in almost every field with a strong need and capacity for storing and handling large datasets (Hilbert and López, 2011). However, choosing correct analysis methods to bring about conclusive results from large datasets remains ambiguous. This paper attempts to predict Amazon review scores by analyzing and modeling 1,697,533 unique reviews from the Amazon Movie Review dataset, with their associated star ratings (Score) and metadata. Variables in the metadata included ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary, Text, and Id. The problem lies in that a large percentage of the reviews are 5 stars introducing bias in the data. Feature extraction, machine learning Decision Tree Classifier algorithm, and cross model validation and testing were implemented to increase the model accuracy on the testing dataset.

Preliminary Analysis / Exploration

Multiple significant figures were produced for preliminary analysis, such as the top 25 products with the most reviews, bottom 25 products with the most reviews, the top 25 reviewers, bottom 25 reviewers, and the top 100 words unique to each score and more. Specifically, the count of scores and helpfulness numerator, helpfulness ratio showed distinct patterns for each score.

Fig 1: On the left is a bar plot showing the binned count for each score (1-5), in the middle is a bar plot showing the helpfulness numerator value assigned to each score, on the right is a bin plot of the helpfulness ratio

The figure above shows a high count of review entries with score 5 elucidating potential data bias in class belonging to score of 5. The helpfulness numerator value for each entry described in figure 1 provides insightful information on the number of people who found each review helpful for each score. For example, reviews with a score of 1 tend to have a higher helpfulness rating than reviews with a 5 star rating. A clear distinction between classes belonging to score 1,2,3 and score 4,5 are made by dividing the HelpfulnessNumerator / HelpfulnessDenominator as shown in figure 1 (right figure).

Feature Extraction

Further future extraction was implemented to enhance model performance. Additional features included stemming words applying sentiment analysis (polarity scores) and TFIDF vectorizer for top featured words. In figure two, the top words in the dataset were *good, bad, great, better, fun, excel*.

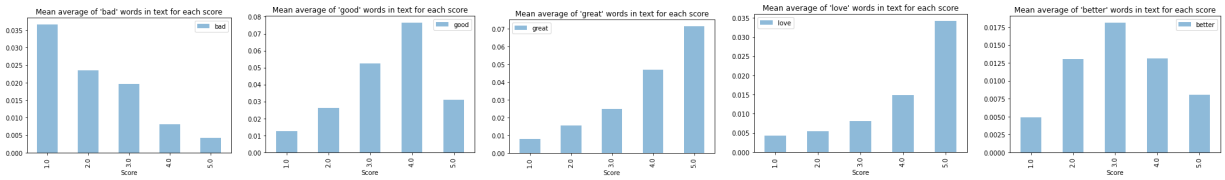

Fig 2: Bar plots featuring the top average words for each score.

Starting from the left, *bad* is frequently seen in reviews with score 1, and linearly decreases as score increases. In contrast, words *great* and *love* are regularly seen in reviews with a score of 5 and linearly decreases as score decreases. Between these two classes are words *good* and *better* which help distinguish words that are associated with reviews that have scores 3 and 4. Unfortunately, features differentiating class belonging to score 2 remain unknown. The SentimentIntensityAnalyzer() method from nltk.sentiment.vader package was used to assign polarity scores to summary reviews.

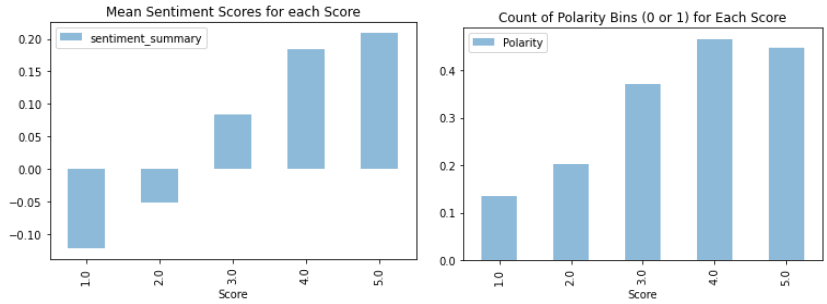

Fig 3: Bar plots showing the mean sentiment values for each score, the sentiment values were binned into 0 or 1 corresponding to positive and negative values.

Figure 3 shows the mean sentiment value for each score. Class belonging to score =1 tends to have sentiment values from -10 to 0 while scores = 2 have sentiment values from -0.05 to 0. Scores that are greater than 2 tend to have positive sentiment values. These two features were significant in differentiating score 1 from score 5. Binning the sentiment score smoothed the model decision classifier but decreased the model accuracy by 0.001.

Flow, Decisions, and Techniques Tried

Multiple techniques were implemented. A jupyter notebook was applied to visualize preliminary data and save the extracted features into a train_X.csv and test_X.csv file. According to Sci-Kit Learn's flow chart, the best estimator for the Amazon dataset would be Naive Bayes or SGD because the obtained data is labeled, categorical and greater than 100,000 entries. When I applied Multinomial Naive Bayes, my model lacked support and had an accuracy score of 50% and it did not allow any negative values. However when I applied Decision Tree Classification, my model experienced a 10% increase in accuracy. Following this improvement, I decided to use Adaboosting but the accuracy score did not change. Additionally, applying a linear kernel using SVD gave me a 40% accuracy. Furthermore, I noticed a similar trend between all the models in which they were predicting class 5 almost 90% of the time. This problem was resolved by

undersampling the remaining classes to the length of the lowest class (Score = 2). Afterwards, my score decreased dramatically (by 20%) which proves that passing in the raw data is not conclusive due to the overfitting trend from the models.

Model Validation/Testing

The process of analyzing the dataset required predicting the testing data by training different models against the training set. The dataset was split into training and test sets using train_test_split() function from sklearn.model_selection() where ¼ of the data was split into the testing set. I implemented a cross validation test for the CLF model to tune the hyperparameters. For example, I used cross_val_score to score the prediction of the KNN model for different values of k (1-21) and different values maxDepth for Decision Tree Classifier.

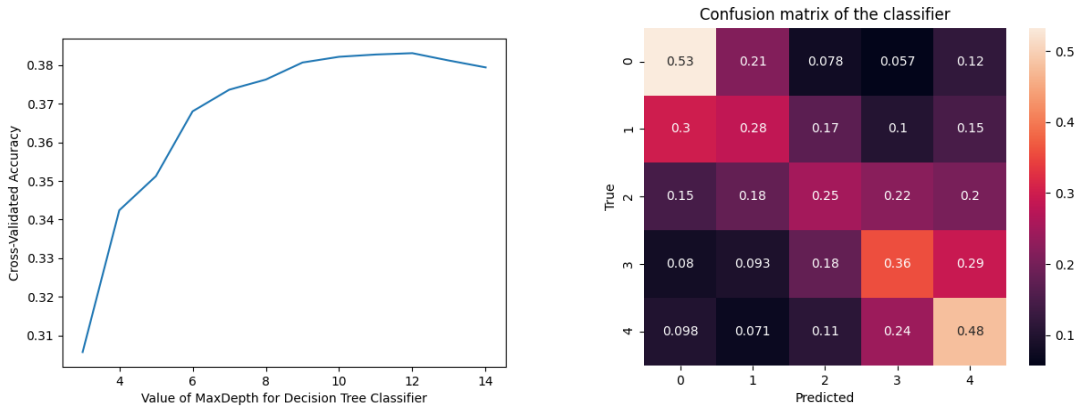

Fig 4: Cross-Validated Accuracy score for each MaxDepth value from 3-14, Confusion matrix for Decision Tree Classifier

In figure 4, a max depth of 12-13 returned the highest cross-validated accuracy. Moreover, the confusion matrix above illustrates a 50% true predication rate of scores 1 and 5 unlike its neighboring classes 2,3,4 where the true accuracy lies around 20%.

Creativity/Challenges/Effort

The goal of this paper was to predict Amazon review scores using significant features and appropriate estimators. Further work is needed to implement more effective and statistical strategies on this dataset. As explained above, features separating class belonging to score 2 were not found. This could explain a low accuracy score from the CLF model. Additionally, the confusion matrix shows a 70-80% confusion rate for scores between 2,3,4. The Helpfulness ratio clearly separates the classes into two buckets belonging to scores 1,2,3 and another belonging to scores 4,5. Moreover, sentiment polarity scores shown in figure 3, separate classes into two buckets belonging to score 1,2 and belonging to scores 3,4,5. These two features introduced contradicting probabilities when predicting score 3. To overcome these challenges I added TFIDF top words such as *good, better* to train the model against. This improved my overall accuracy score by 0.2. If I had more time and python skills, I would have liked to try linear regression models and movie text scraping tools such as beautiful soup.

References

API design for machine learning software: experiences from the scikit-learn project, Buitinck *et al.*, 2013.

Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, *332*(6025), 60 –65. doi:10.1126/science.1200970

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.