PREDICTING THE STAR RATINGS OF AMAZON MOVIE REVIEWS USING CLASSICAL MACHINE LEARNING ALGORITHMS

Name: Shwetha Krishnan *Email: shwethak@bu.edu* *Kaggle id: Shwetha_Krishnan18*

ABSTRACT

In this paper, my goal is to predict the star ratings associated with user reviews from Amazon Movie Reviews using classical machine learning algorithms such as k-nearest neighbors (KNN), Naïve Bayes, Random forest classifier, Linear regression and Support Vector Machines. I examine features such as the length of the summary and full text review(before and after removing stop words), when the review was written, average score for each product, average score given by each user, number of exclamation marks and uppercase characters, standard deviation from the mean score, etc. Finally, I evaluate what the best machine learning models by using the k-fold cross-validation technique. I conduct accuracy tests using root mean squared error and a confusion matrix.

INTRODUCTION – PRELIMINARY ANALYSIS

According to resent research, about 84% shoppers trust online reviews as much as a personal recommendation, and 91% of shoppers occasionally or regularly read online reviews.[i] Thus, online review data is very important for companies and making strong predictions using available data and machine learning algorithms can prove to be extremely profitable.

In this project, I use the Amazon Movie Ratings dataset.[ii] The training data has 1,697,533 unique reviews from Amazon Movie Reviews with the following data fields: ProductId, UserId, HelpfulnessNumerator, HelpfulnessDenominator,
Score, Time, Summary, Text and ID. With this, I conducted some preliminary analysis. Preliminary analysis is an important step as it helps us understand the data, it's key features, correlations between data fields, examining distributions and identifying outliers. This helps us create a mental summary of the data so we can prepare it for further analysis.

For my preliminary analysis I looked up and created simple boxplot, bar-graphs and pie chart vitalizations on the following:

1. The frequency of each score
2. The top 25 most rated products
3. The top 25 least rated products
4. The top 25 reviewers
5. The top 25 most helpful reviews
6. The top 25 least helpful reviews
7. Percentile values of how helpful each review score was.

WORKFLOW

The workflow that I used for this project was to first pre-process the data by pruning. This step involved removing filling NaN values with 0, removing redundant features, etc. Next, I did feature extraction to strengthen my predictions. For this I used the sentiment intensity analysis and other features. Next, I was model building through the different machine learning classifier algorithms and my finally step was model evaluation and testing. In this section, I will discuss this and the methods I used in detail.

DATA PRUNNING

Data pruning is the process of removing sub-optimal and redundant features to improve the accuracy of the machine learning model and it also helps to reduce the size of the data for better processing efficiency. In my project, I went about this task with the following steps:

- Reviews wherein the HelpfulnessNumerator was greater than the HelpfulnessDenominator were removed.
- It was found that a few Review and Summary texts had float values. Thus, all entries in the Text and Summary columns where converted into string values.

For text pre-processing I did the following:

- I first tokenized the words into a list of strings
- I removed stop words (such as "the", "a", "an", "in") that are common and hold no importance in our analysis.
- I then used SnowballStemmer to create stems of words.
- I also performed Lemmatization using WordNetLemmatizer but decided to remove it from the code as it increased runtime by a lot with not much increase in accuracy.

This text preprocessing step was essential as using the length of the processed text proved more accurate that the original. It is also helpful in creating vocabularies and measuring words frequencies using the tf-idf vectorization method.

DATA EXPLORATION AND FEATURE EXTRACTION

This is probably the most important step in our workflow and involves extracting features from the original dataset. This is involves converting text and data into numerical form that can be understood by our computers to achieve better performance.

(1) Helpfulness: This tells us how helpful a review entry was.
(2) Unhelpfulness: This tells us how unhelpful a review entry was.
(3) UserAvgScore: This tells us the average rating a user has given. It is observed that users usually rate close to their mean rating and usually don't go to extremes (A user who mostly rates a 5 rarely tends to rate a 2 or 1).
(4) ProductAvgScore: This tells us the average rating each movie got based on ProductId
(5) NumUppercase_T: This is helps us in providing a numeric value to a text by counting the number of Uppercase letters Review Text has. It is observed that the more the number of upper-case characters, the happier or angrier the review is. Thus, this comes in handy during sentiment analysis-based clustering.
(6) NumUppercase_S: Same as NumUppercase_T, but for Summary field.
(7) Month: It is observed that the months of December and November have products rated more highly than other months. This could be because of festive months. People might like Christmas movies a lot!
(8) numExclamation_T: Again, this helps us enumerate a sentiment. Higher the number of exclamation points in a review, the happier or angrier the sentiment. Reviews with higher number of exclamation points seem to have a higher rating.
(9) numExclamation_S: same as numExclamation_T but for the Summary data field
(10) CleanSummaryLength: This is the length of the Summary data field after removing stop words and is seen to be more effective than just the original Summary length.
(11) CleanReviewLength: same as CleanSummaryLength but for the Review data field
(12) SentimentIntensityAnalyzer: tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is. It takes in a string and returns a dictionary of scores in each of four categories: negative(neg), neutral(neu) positive(pos), compound(compound). I dropped the neu and compound scores as they did not have an impact on classification.
(13) Tf-idf vectorization of string: I decided not to use this method as the SentimentIntensityAnalyzer gave me a better accuracy when used, but this method is effective, nonetheless. This helps us vectorize words and create a dictionary based on how frequently it occurs in our dataset. This is powerful as it allows for parameters that improve accuracy such as ngram_range (best accuracy reached with (1, 3)), min_df which I set to 0.00001, max_df which I set to 0.5 and max_features that I set to 150000.

Numerical Values were standardized using the Standard Scalar. This is important as it rescales the data in such a way that all the variables and their values are on the same scale. The Standard Scalar standardizes a feature by subtracting the mean and then scaling to unit variance. This step also reduces the influence of outliers and helps the data converge.

For my final input I used Helpfulness, UserAvgScore, NumUppercase_T, NumUppercase_S, Month, numExclamation_T, CleanReviewLength and pos, neg values from SentimentIntensityAnalyzer.

6. MODELS TESTED

The machine learning classifier models that I tested on were:

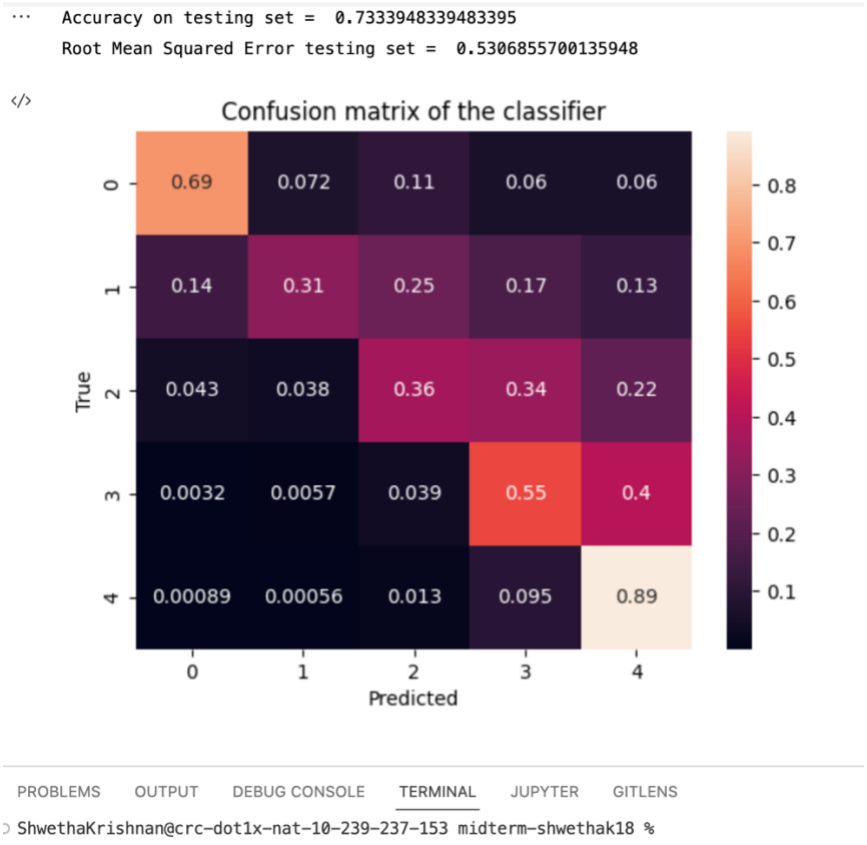
(1) KNN - KNeighborsClassifier
(2) Decision Trees – RandomForestClassifier
(3) Logistic Regression – LogisticRegression
(4) Naïve Bayes – GaussianNB
(5) Support Vector Machine – SVC

My predictions where the most accurate using the RandomForestClassifier with an accuracy on testing set of 73% and a root mean squared error on testing set of 0.530685. I used parameters min/max_deapth = 12, min_sample_split = 8, n_estimators=50. The Random forest classifier uses ensemble technique- bagging and boosting. Random forests are created from data subsets and the final output is an average ranking to ensure that overfitting is taken care of. The random forest classifier randomly selects observations and builds a decision tree. The average result is then taken. Individual decision trees are constructed for each sample and each decision tree generates an output, the average of which is used in classification.

8. EVALUATION

I evaluated my models using:

(1) Confusion matrix: I used the confusion_matrix() method from sklearn.metrics. The confusion matrix gives us a summary of the predictions made by the classification algorithm and gives us insights on the types of errors being made. The result of my confusion matrix is shown below.



(2) Accuracy score: I used the accuracy_score() method of sklearn.metrics. This accepts true labels of the samples and the labels predicted by the model as its parameter and computes the accuracy score as a float value. The higher the accuracy score, the higher is the efficiency of the model.

(3) Root mean squared error: I used the mean_squared_error() method from sklearn.metrics. and applied the sqrt function to it that I imported from NumPy. This is used to calculate the transformation between values predicted by a model and actual values – Y_test and Y_test_predicted. The lower the RMSE, better is the fit and accuracy.

(4) K-fold cross validation, cross-validation score: I used the KFold() and cross_val_score methods imported from sklearn.model_selection with parameters: n_splits = 10 and shuffle=True. In this method, the dataset is split into k=10 splits or fold and is used to evaluate the model's ability in predicting Y_train from Y_train_processed.

9. CONCLUSIONS

I really enjoyed working on this project and it was a great learning experience! Some new skills I learnt are: SentimentIntensityAnalyzer, tf-idf and k-fold cross validation. Some challenges I faced where that my Kernel often crashed while running the whole dataset and the large runtime made testing cumbersome as just running it on a sample doesn't ensure that problems won't arise on the entire dataset. For example, when I ran my model a sample size of 100000, I'd not face any issues but running it on the whole dataset resulted in problems such as float/null values seen when a string value is expected. I also had trouble fitting my tf-idf vectorize method on the whole dataset as it led to the kernel crashing within minutes.

[i] https://www.inc.com/craig-bloem/84-percent-of-people-trust-online-reviews-as-much-.html

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