**Data 602 Final Project**

EDA of Amazon Cellphone reviews

Atal Sai Sooraj Mutyala

**About the Dataset:**

The datasets deal with cell-phone ratings and reviews on the e-commerce website Amazon. It contains 82,815 rows and 8 columns like unique id for each review (asin), name of the reviewer, the rating they gave for each product, if the review is verified, title of the review, the content of the review and helpful votes.

**Data Cleaning:**

The data to be analyzed is primarily text data. After importing the csv file, it is imperative to check the data types of each column. Boolean values in the ‘verified’ column were converted to integer values in the format 1 and 0 for ease of execution. Later, I converted ‘helpfulVotes’ column into an integer data type.

A screenshot of a cell phone

Description automatically generated

From the data frame, I checked if there were any missing values in each column and if there are NaN values.

A screenshot of a cell phone

Description automatically generated

We can ignore columns that do not contribute to our analysis and can possibly cause over-fitting of a machine learning algorithm. For example, the ‘asin’,’name’ and ‘date’ column can be ignored as they do not directly affect the rating of the smartphone.



NaN values in the ‘helpfulVotes’ column have been filled with 0. After this, we can confidently state that the dataset is clean and apt for our analysis.

**Exploratory Data Analysis:**

Before we begin our core analysis, it is wise to know what major topics the data revolves around. **Topic modeling** is an unsupervised technique that intends to analyze large volumes of text data by clustering the documents into groups. In the case of topic modeling, the text data do not have any labels attached to it. Rather, topic modeling tries to group the documents into clusters based on similar characteristics. Types of Topic modeling include:

* **Latent Semantic Analysis (LSA):** The main goal of Latent Semantic Analysis (LSA) is to create vector-based representation for texts to make semantic content. By vector representation, LSA computes the similarity between texts to pick the heist efficient related words.
* **Probabilistic Latent Semantic Analysis (PLSA):** The main goal of PLSA is identifying and distinguishing between different contexts of word usage without recourse to a dictionary or thesaurus. It includes two important implications: First one, it allows to disambiguate polysemy, i.e., words with multiple meanings. Second thing, it discloses typical similarities by grouping together words that shared a common context.
* **Latent Dirichlet Allocation (LDA):** LDA is a generative model that tries to mimic what the writing process is. The basic idea of the process is, each document is modeled as a mixture of topics, and each topic is a discrete probability distribution that defines how likely each word is to appear in each topic.

It is important to mention here that it is extremely difficult to evaluate the performance of topic modeling since there are no right answers. It depends upon the user to find similar characteristics between the documents of one cluster and assign it an appropriate label or topic. From the results obtained, I believe PLSA was able to garner most words that were important towards that topic better than the other techniques due its underlying ability of detecting context.

To use a machine learning algorithm or a statistical technique on any form of text, it is prescribed to transform the text into some numeric or vector representation. To convert our text data into numeric data, we can employ **TfidfVectorizer**. It gives us the term frequency of each word in the string over an entire column in the data-frame. We can now split the numeric data into test (20%) and training (80%) sets to employ classifier algorithms.

**Data used for Classification:**

For the classification of data, I used subsets of data from the data frame for each classifier to test and evaluate the importance of it in the analysis. It would also give a broader spectrum of accuracy scores for each model enlightening us on how an algorithm performs when the training data is varied.

* **Title column:** This column contains the title of every review given by a customer. As discussed previously, I converted this text data into numerical vectors using the TfidfVectorizer.
* **Body column:** This articulates the customer experience describing their opinions on the product they ordered. As above, I converted this text into a series of NumPy arrays.
* **Title, Body, Verified, Helpful Votes columns with NMF(n\_components=10): In** this case I used the arrays from title and body columns employing a Dimensionality Reduction (NMF) to reduce the number of columns to 10. Later, I appended these columns to Verified and Helpful Votes columns.

Classification algorithms like Decision Tree, Random Forest, KNearestNeighbor and MultinomialNB were employed to find the best fit for our data.

**Decision Tree:** A Decision Tree is a simple representation for classifying examples. It is a supervised machine learning algorithm where the data is continuously split according to a certain parameter. We can think of a decision tree as a series of yes/no questions asked about our data eventually leading to a predicted class.

**Random Forests:** Random forest, like its name implies, consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction**.**

**KNearestNeighbor:** An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

**MultinomialNB:** The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). It estimates the conditional probability of a word given a class as the relative frequency of term t in documents belonging to class(c). The variation considers the number of occurrences of term t in training documents from class (c), including multiple occurrences.

**Results and Conclusions:**

After implementing the above classifiers to each of the data subsets, I evaluated the accuracy scores of each classifier.

The high accuracy scores of Neural network can be attributed to the existence of **back-propagation** which gives the classifier a flexibility of propagating the total loss back into the neural network to know how much of the loss every node is responsible for, and subsequently updating the weights in each epoc ,in such a way that it minimizes the loss by giving the nodes with higher error rates lower weights and vice-versa.

|  |  |  |  |
| --- | --- | --- | --- |
| Classifier Algorithm | Accuracy when data subset column=’body’ | Accuracy when data subset  column=’title’ | Accuracy when NMF (n\_components=10)  columns=’body’,’title’,’verified’,’helpfulVotes’ |
| Random Forest | 68.24 | 72.13 | 69.78 |
| KNN | 55.32 | 67.42 | 61.09 |
| MultinomialNB | 65.88 | 71.25 | 53.16 |
| Decision Tree | 58.62 | 69.58 | 61.71 |
| Neural Network | 87.33 | 80.87 | - |

As the number of epocs given to the neural network is 100, I believe this provides the classifier ample scope and time to constantly learn from its errors and improve itself. Neural Networks had the best accuracy scores when employed on both ‘body’ and title’, outweighing the rest by a comfortable margin. Hence, **the most efficient classifier is Neural Networks**.

The most important feature of the dataset is the ‘title’ column since every algorithm performed well in predicting it.