International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

Detecting and Characterizing Extremist Reviewer Groups in Online Product Reviews

Jothika. J[1], Nalagampalli MoniSree[1], Pavithra. M[2]
UG Scholar[1], Assistant Professor[2]
Department of CSE, Kingston College, Vellore, Tamil Nadu, India

Article Info
Volume 8, Issue 3
Page Number : 342-353

Publication Issue :
May-June-2022

Article History
Accepted: 03 June 2022
Published: 15 June 2022

ABSTRACT

Now a days, online marketplaces often witness opinion spam in the form of reviews. People are often hired to target specific brands for promoting or impeding them by writing highly positive or negative reviews. This is often collectively done in groups. Although some previous studies attempted to identify and analyze such opinion spam groups, little has been explored to spot those groups who target a brand as a whole, instead of just products. In this application, we collected the reviews from the Amazon product review site and manually labeled a set of 923 candidate reviewer groups. The groups are extracted using frequent item set mining over brand similarities such that users are clustered together if they have mutually reviewed (products of) a lot of brands. We hypothesize that the nature of the reviewer groups is dependent on eight features specific to a (group, brand) pair. We develop a feature-based supervised model to classify candidate groups as extremist entities. We run multiple classifiers for the task of classifying a group based on the reviews written by the users of that group to determine whether the group shows signs of extremity. A three-layer perceptron-based classifier turns out to be the best classifier. We further study behaviors of such groups in detail to understand the dynamics of brand-level opinion fraud better. These behaviors include consistency in ratings, review sentiment, verified purchase, review dates, and helpful votes received on reviews. Surprisingly, we observe that there are a lot of verified reviewers showing extreme sentiment, which, on further investigation, leads to ways to circumvent the existing mechanisms in place to prevent unofficial incentives on Amazon.

Keywords : Characterization, Detection, Online Products, Reviews.

I. INTRODUCTION

In today's world dominated by online marketplaces, review portals and websites play a crucial role in the buyer's decision for their next purchase. "It is a virtuous cycle–the more reviews, the more buys. The more buys, the more reviews. The more buys, the higher your rank in search and the more sales you get," says Alice, the owner of online cosmetic brand Elizabeth Mott. Undoubtedly, it is highly likely that some people write reviews that are less than truthful to manipulate widespread decision of buyers in their favor. These people act either individually or in groups. While individual reviewers write such reviews in a matter of frustration or joy, they do not influence the overall opinion on a product to a large extent but help other buyers by stating their experiences. However, a more compelling case is when multiple individuals form an intricate web, and due to sheer higher number of people reviewing (and certain other techniques, discussed in Section VIII), they end up being a major influence on the overall sentiment of the product. The extent of such influence is not just limited to the reviews by opinion spam. Previous work has shown that 10%–15% reviews are essentially echoing the earliest reviews, and thus, a misleading early review has an even higher influential potential. This is widespread opinion spam, and every review website must be aware of this activity and take appropriate measures for the identification and/or prevention of this phenomenon. This is a classic example of collective fraud behavior, where several users are part of a business network and work together to target and influence a particular product. This is a lesser known phenomenon, and most groups work following certain techniques to not make their collaboration obvious. However, since such groups are economically or otherwise incentivized, and several of these are generally run by a given organization, they have several targets for opinion spam, which often share certain common characteristics in their nature of reviews. These characteristics can be exploited to classify them better using a robust and thorough an analysis technique. Amazon India, to prevent opinion spam, has brought about a new policy that limits the number of reviews on a product in a day, as stated in. In order to still be effective, we claim that certain groups target brands in general and post extreme reviews across multiple products for a given target brand. This is a higher level of opinion spamming, deliberately writing highly positive or negative reviews for a brand in general in order to promote or demote them in the cut-throat competition of the online marketplace. Studies have been conducted to identify such groups that try to influence a product; however, groups exhibiting a brand-based opinion spamming is a phenomenon that remains widely unexplored. A detailed discussion is required for these brand-related activities because these practices are against the code of conduct of these review websites since they negatively skew the brand-based competition, giving innate (dis)advantages to certain brands. Since only the nonverified reviews are limited by the policies,1 reviewers from these groups can often purchase the Section IV of the Supplementary Material). In this article, we identify and study the behavioral characteristics of extremist reviewer groups. We also build a feature-based classifier based on the brand-specific activities of reviewer groups to identify the extremist groups on the Amazon India marketplace. We then further analyze our methodology to unfold behaviors that best signify such activities and compare and analyze the overall trend of these groups viz-a-viz their behaviors. The major contributions of this article are fourfold: 1) a manually labeled data set of 923 reviewer groups that are classified into "extremist" and "moderate" categories; 2) the first-ever characterization and study of the novel problem of identifying brand-level extremism; 3) detailed characterization of extremist reviewer groups; 4) design supervised approach to detect extremist reviewer groups. To encourage reproducible research, we have made the codes publicly available at https://github.com/virresh/ extremist-reviewers.2 This article is organized as follows. We briefly survey the various studies related to review extremism, applications developed using reviews, and fake review detection in Section II. Section III outlines the details of the collected data set and annotation methodology.

Section IV presents the modeling of features at brand level and features.

II. RELATED WORKS

HawkesEye: Detecting fake retweeters using Hawkes process and topic modeling: Retweets are essential to boost the popularity of a tweet, and a large number of fake retweeters can contribute heavily to this aspect. We define a fake retweeter as a Twitter account that retweets spammy tweets, retweets an abnormally large amount of tweets in a short period, or misuses a trending hashtag to promote events irrelevant to the topic of discussion. We introduce an up-to-date, temporally diverse, trend-oriented labeled dataset to address the problem of fake retweeter detection. We develop a novel classifier, called HawkesEye which makes predictions based on a temporal window, in contrast to existing approaches which require a <italic>graph-like</italic> relationship between tweet entities, or the presence of the <italic>entire retweeting timeline</italic> of a retweeter. HawkesEye utilizes both temporal and textual information using a class-specific topic model and Hawkes processes.

Spotting collective behaviour of online frauds in customer reviews: Online reviews play a crucial role in deciding the quality before purchasing any product. Unfortunately, spammers often take advantage of online review forums by writing fraud reviews to promote/demote certain products. It may turn out to be more detrimental when such spammers collude and collectively inject spam reviews as they can take complete control of users' sentiment due to the volume of fraud reviews they inject. Group spam detection is thus more challenging than individuallevel fraud detection due to unclear definition of a group, variation of inter-group dynamics, scarcity of labeled group-level spam data, etc. Here, we propose DeFrauder, an unsupervised method to detect online fraud reviewer groups. It first detects candidate fraud groups by leveraging the underlying product review graph and incorporating several behavioral signals which model multi-faceted collaboration among reviewers. It then maps reviewers into an embedding space and assigns a spam score to each group such that groups comprising spammers with highly similar behavioral traits achieve high spam score. While comparing with five baselines on four real-world datasets (two of them were curated by us), DeFrauder shows superior performance by outperforming the best baseline with 17.11% higher NDCG@50 (on average) across datasets

What makes a helpful online review? A meta-analysis of review characteristics: In this study, we aim to clarify the determinants of online review helpfulness concerning review depth, extremity and timeliness. Based on a meta-analysis, we examine the effects of important characteristics of reviews employing 53 empirical studies yielding 191 effect sizes. Findings reveal that review depth has a greater impact on helpfulness than review extremity and timeliness with the exception of its sub-metric of review volume, which exerts the negative influence on review helpfulness. Specifically, readability is the most important factor in evaluating review helpfulness. Furthermore, we discuss important moderators of the relationships and find interesting insights regarding website and culture background. In accordance with the results, we propose several implications for researchers and E-business firms. Our study provides a much needed quantitative synthesis of this burgeoning stream of research.

Machine Learning Classifiers to Detect Malicious Websites: We address the problem of automating the process of network troubleshooting for a large-scale WiFi network. Specifically, we target identifying the causes of unnecessary active scans in WiFi networks, that are known to degrade the WiFi performance. We collect 340 hours worth of data with several thousands of episodes of active scans to train various machine learning models. Data is collected with 27 devices across vendors in varied network setups under a controlled setting. We study unsupervised and supervised machine learning techniques to conclude that a multilayer perceptron is the best model to detect the causes of active scanning. Further, we perform an in-vivo model validation in an uncontrolled real-world WiFi network.

Learning distributed representations from reviews for collaborative filtering: Motivated by the success of this approach, we introduce two different models of reviews and study their effect on collaborative filtering performance. While the previous state-of-the-art approach is based on a latent Dirichlet allocation (LDA) model of reviews, the models we explore are neural network based: a bag-of-words product-of-experts model and a recurrent neural network.

We demonstrate that the increased flexibility offered by the product-of-experts model allowed it to achieve state-of-the-art performance on the Amazon review dataset, outperforming the LDA-based approach. However, interestingly, the greater modeling power offered by the recurrent neural network appears to undermine the model's ability to act as a regularizer of the product representations.

III. Methodology

In proposed system, unlike other studies that majorly focus on fake review/reviewer detection, we here focus on extremist reviewer detection, which may not be fake. Moreover, we attempt to identify "groups" instead of detecting "individual user" by using machine learning algorithms.

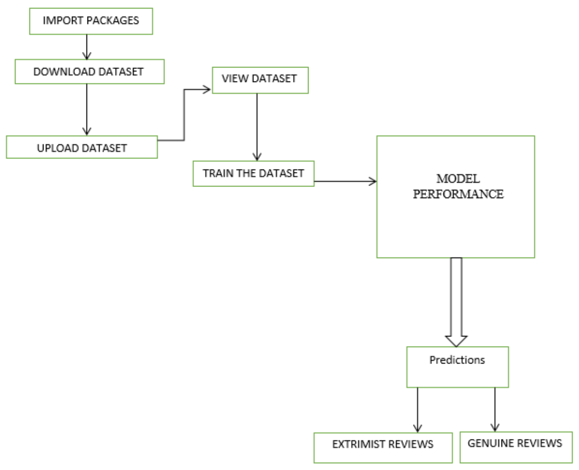


Figure 1 : Block diagram of proposed method

IV. IMPLEMENTATION

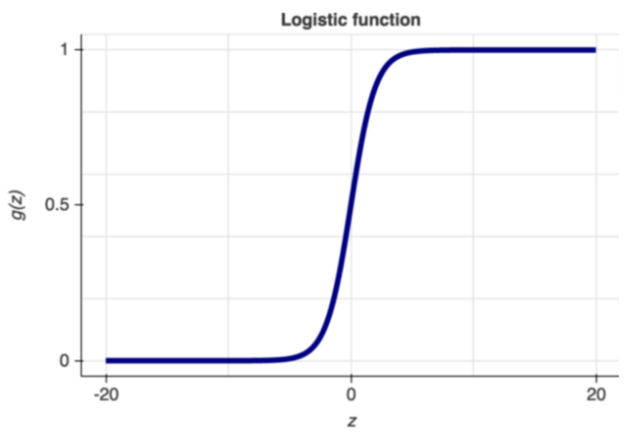
Logistic regression

Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

Step1: Logistic regression hypothesis

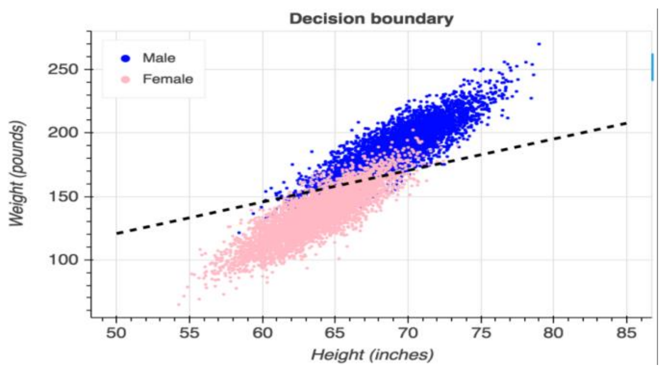
The logistic regression classifier can be derived by analogy to the logistic *regression* the function **g(z)** is the logistic function also known as the *sigmoid function*.

The logistic function has asymptotes at 0 and 1, and it crosses the y-axis at 0.5.



Step (1b): Logistic regression decision boundary

Since our data set has two features: height and weight, the logistic regression hypothesis is the following



Random forest classifier

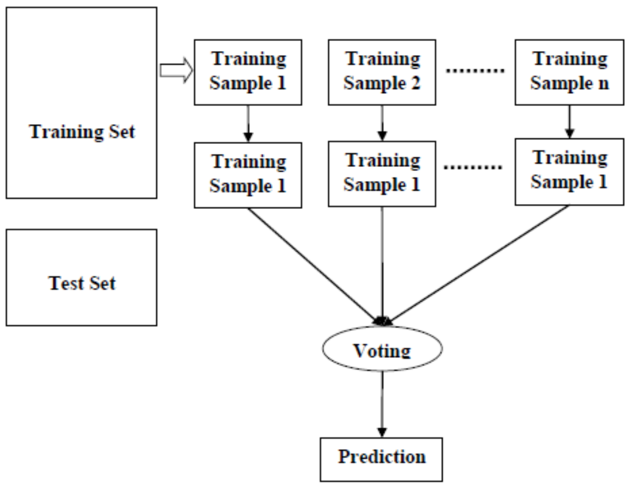
Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Working of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps –

- **Step 1** – First, start with the selection of random samples from a given dataset.
- **Step 2** – Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** – In this step, voting will be performed for every predicted result.
- **Step 4** – At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working –



Naive Bayes algorithm

Bayes' Theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes' Theorem is stated as:

P (class data) = (P (data class) * P(class)) / P(data)

Where P (class data) is the probability of class given the provided data.

1) **Step 1: Separate by Class**

This means that we will first need to separate our training data by class. A relatively straightforward operation.

We can create a dictionary object where each key is the class value and then add a list of all the records as the value in the dictionary.

2) Step 2: Summarize Dataset

We need two statistics from a given set of data.

We'll see how these statistics are used in the calculation of probabilities in a few steps. The two statistics we require from a given dataset are the mean and the standard deviation (average deviation from the mean).

The mean is the average value and can be calculated as:

- mean = sum(x)/n * count(x)

Where *x* is the list of values or a column we are looking.

3) Step 3: Summarize Data By Class

We require statistics from our training dataset organized by class.

Above, we have developed the *separate_by_class ()* function to separate a dataset into rows by class. And we have developed *summarize dataset ()* function to calculate summary statistics for each column.

We can put all of this together and summarize the columns in the dataset organized by class values.

4) Step 4: Gaussian Probability Density Function

Calculating the probability or likelihood of observing a given real-value like X1 is difficult.

One way we can do this is to assume that X1 values are drawn from a distribution, such as a bell curve or Gaussian distribution.

A Gaussian distribution can be summarized using only two numbers: the mean and the standard deviation. Therefore, with a little math, we can estimate the probability of a given value. This piece of math is called a Gaussian Probability Distribution Function (or Gaussian PDF) and can be calculated as:

f(x) = (1 / sqrt(2 * PI) * sigma) * exp(-((x-mean)^2 / (2 * sigma^2)))

Where *sigma* is the standard deviation for *x*, *mean* is the mean for *x* and *PI* is the value of pi.

Support vector classifiers algorithm

Support Vector Machine or SVM algorithm is a simple yet powerful Supervised Machine Learning algorithm that can be used for building both regression and classification models. SVM algorithm can perform really well with both linearly separable and non-linearly separable datasets. Even with a limited amount of data, the support vector machine algorithm does not fail to show its magic.

Step 1: Load Pandas library and the dataset using Pandas

Step 2: Define the features and the target

Step 3: Split the dataset into train and test using sklearn before building the SVM algorithm model

Step 4: Import the support vector classifier function or SVC function from Sklearn SVM module. Build the

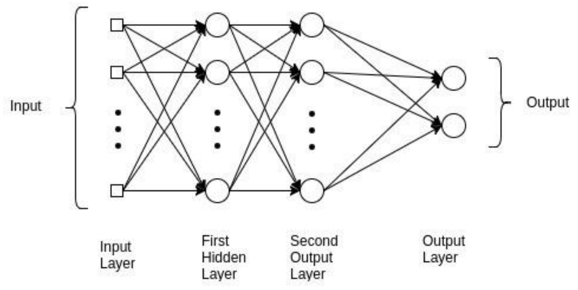
Support Vector Machine model with the help of the SVC function

Step 5: Predict values using the SVM algorithm model

Step 6: Evaluate the Support Vector Machine model

Multi linear perceptron:

In the Multilayer perceptron, there can more than one linear layer (combinations of **neurons**). If we take the simple example the three-layer network, first layer will be the *input layer* and last will be *output layer* and middle layer will be called *hidden layer.* We feed our input data into the input layer and take the output from the output layer. We can increase the number of the hidden layer as much as we want, to make the model more complex according to our task.



Feed Forward Network, is the most typical neural network model. Its goal is to approximate some function f (). Given, for example, a classifier **y = f * (x)** that maps an input x to an output class y, the MLP find the best approximation to that classifier by defining a mapping, y = f(x; θ) and learning the best parameters θ for it. The MLP networks are composed of many functions that are chained together. A network with three functions or layers would form f(x) = f (3)(f (2)(f (1)(x))). Each of these layers is composed of units that perform an affine transformation of a linear sum of inputs. Each layer is represented as y = f(WxT + b). Where f is the activation function (covered below), W is the set of parameter, or weights, in the layer, x is the input vector, which can also be the output of the previous layer, and b is the bias vector. The layers of an MLP consists of several fully connected layers because each unit in a layer is connected to all the units in the previous layer. In a fully connected layer, the parameters of each unit are independent of the rest of the units in the layer, that means each unit possess a unique set of weights.

In a supervised classification system, each input vector is associated with a label, or ground truth, defining its class or class label is given with the data. The output of the network gives a class score, or prediction, for each input. To measure the performance of the classifier, the loss function is defined. The loss will be high if the predicted class does not correspond to the true class, it will be low otherwise. Sometimes the problem of overfitting and underfitting occurs at the time of training the model. In this case, our model performs very well on training data but not on testing data. In order to train the network, an optimization procedure is required for this we need loss function and an optimizer. This procedure will find the values for the set of weights, W that minimizes the loss function.

A popular strategy is to initialize the weights to random values and refine them iteratively to get a lower loss. This refinement is achieved by moving on the direction defined by the gradient of the loss function. And it is important to set a learning rate defining the amount in which the algorithm is moving in every iteration.

K-NEAREST NEIGHBOUR:

This k-Nearest Neighbors tutorial is broken down into 3 parts:

Step 1: Calculate Euclidean Distance.
Step 2: Get Nearest Neighbors.
Step 3: Make Predictions

Step 1: Calculate Euclidean Distance.

His first step is to calculate the distance between two rows in a dataset.

Rows of data are mostly made up of numbers and an easy way to calculate the distance between two rows or vectors of numbers is to draw a straight line. This makes sense in 2D or 3D and scales nicely to higher dimensions

Euclidean Distance = sqrt(sum i to N (x1_i – x2_i)^2)
Where *x1* is the first row of data, *x2* is the second row of data and *i* is the index to a specific column as we sum across all columns.
With Euclidean distance, the smaller the value, the more similar two records will be. A value of 0 means that there is no difference between two records

5) **Step 2: Get Nearest Neighbours:**

Neighbors for a new piece of data in the dataset are the *k* closest instances, as defined by our distance measure.
To locate the neighbors for a new piece of data within a dataset we must first calculate the distance between each record in the dataset to the new piece of data. We can do this using our distance function prepared above.

Once distances are calculated, we must sort all of the records in the training dataset by their distance to the new data. We can then select the top *k* to return as the most similar neighbors.

We can do this by keeping track of the distance for each record in the dataset as a tuple, sort the list of tuples by the distance (in descending order) and then retrieve the neighbor

6) **Step 3: Make Predictions**

The most similar neighbors collected from the training dataset can be used to make predictions.

In the case of classification, we can return the most represented class among the neighbors.

We can achieve this by performing the *max()* function on the list of output values from the neighbors. Given a list of class values observed in the neighbors, the *max()* function takes a set of unique class values and calls the count on the list of class values for each class value in the set.

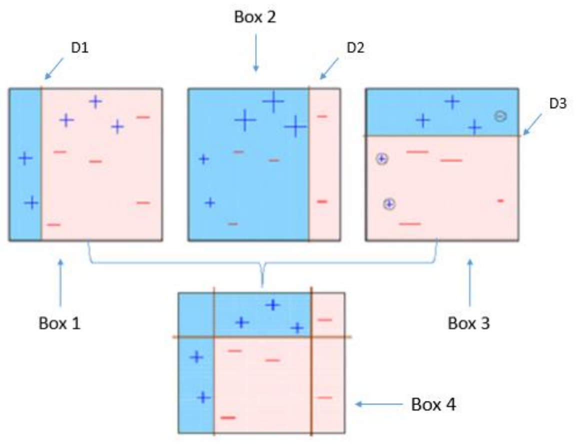
XG-BOOST

XGBoost is the most popular machine learning algorithm these days. Regardless of the data type (regression or classification), it is well known to provide better solutions than other ML algorithms. In fact, since its inception (early 2014), it has become the "true love" of kaggle users to deal with structured data. So, if you are planning to compete on Kaggle, xgboost is one algorithm you need to master.

XGBoost (**Ex**treme **G**radient **Boost**ing) is an optimized distributed gradient boosting library. Yes, it uses gradient boosting (GBM) framework at core. Yet, does better than GBM framework alone. XGBoost was created by Tianqi Chen, PhD Student, University of Washington. It is used for supervised ML problems.

XGBoost belongs to a family of boosting algorithms that convert weak learners into strong learners. A weak learner is one which is slightly better than random guessing. Let's understand **boosting first** (in general).

Boosting is a sequential process; i.e., trees are grown using the information from a previously grown tree one after the other. This process slowly learns from data and tries to improve its prediction in subsequent iterations. Let's look at a classic classification example:



XGBoost can used to solve both regression and classification problems. It is enabled with separate methods to solve respective problems. Let's see:

Classification Problems: To solve such problems, it uses booster = gbtree parameter; i.e., a tree is grown one after other and attempts to reduce misclassification rate in subsequent iterations. In this, the next tree is built by giving a higher weight to misclassified points by the previous tree (as explained above).

Regression Problems: To solve such problems, we have two methods: booster = gbtree and booster = gblinear. You already know gbtree. In gblinear, it builds generalized linear model and optimizes it using regularization (L1,L2) and gradient descent. In this, the subsequent models are built on residuals (actual - predicted) generated by previous iterations

SGD:

Before explaining Stochastic Gradient Descent (SGD), let’s first describe what Gradient Descent is. Gradient Descent is a popular optimization technique in Machine Learning and Deep Learning, and it can be used with most, if not all, of the learning algorithms. A gradient is the slope of a function. It measures the degree of change of a variable in response to the changes of another variable. Mathematically, Gradient Descent is a convex function whose output is the partial derivative of a set of parameters of its inputs. The greater the gradient, the steeper the slope.

Starting from an initial value, Gradient Descent is run iteratively to find the optimal values of the parameters to find the minimum possible value of the given cost function.

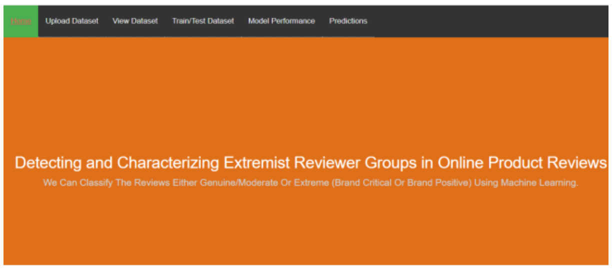
The word ‘*stochastic*‘ means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called “batch” which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy and less random manner, but the problem arises when our datasets gets big. Suppose, you have a million samples in your dataset, so if you use a typical Gradient Descent optimization technique, you will have to use all of the one million samples for completing one iteration while performing the Gradient Descent, and it has to be done for every iteration until the minima is reached. Hence, it becomes computationally very expensive to perform.

This problem is solved by Stochastic Gradient Descent. In SGD, it uses only a single sample, i.e., a batch size of one, to perform each iteration. The sample is randomly shuffled and selected for performing the iteration.

V. RESULTS AND DISCUSSION

The following images will visually depict the process of our project.

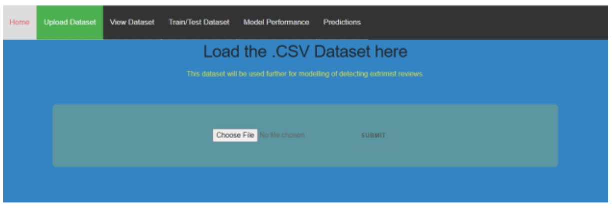
Home page:



Detecting and Characterizing Extremist Reviewer Groups in Online Product Reviews

We Can Classify The Reviews Either Genuine/Moderate Or Extreme (Brand Critical Or Brand Positive) Using Machine Learning.

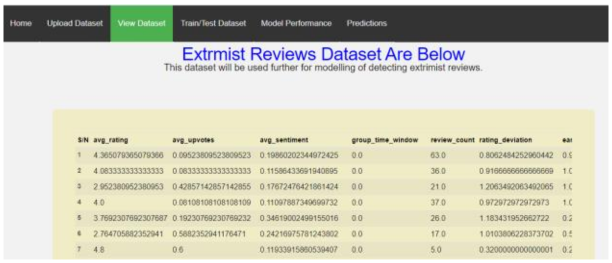
Upload dataset:



Load the .CSV Dataset here

This dataset will be used further for modelling of detecting extremist reviews.

View dataset:

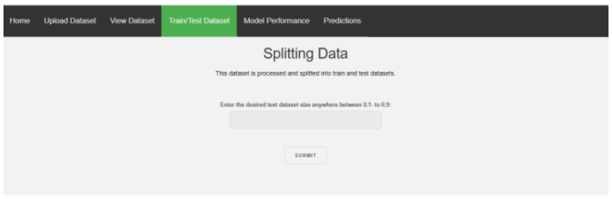


Extrmist Reviews Dataset Are Below

This dataset will be used further for modelling of detecting extrimist reviews.

S/N	avg_rating	avg_upvotes	avg_sentiment	group_time_window	review_count	rating_deviation	ea
1	4.365079365079366	0.09523809523809523	0.19860202344972425	0.0	63.0	0.8062484252960442	0.9
2	4.083333333333333	0.08333333333333333	0.11586433691940895	0.0	36.0	0.9166666666666669	1.0
3	2.952380952380953	0.42857142857142855	0.17672476421861424	0.0	21.0	1.2063492063492065	1.0
4	4.0	0.08108108108108109	0.11097887349699732	0.0	37.0	0.972972972972973	1.0
5	3.7692307692307687	0.19230769230769232	0.34619002499155016	0.0	26.0	1.183431952662722	0.2
6	2.764705882352941	0.5882352941176471	0.24216975781243802	0.0	17.0	1.0103806228373702	0.5
7	4.8	0.6	0.11933915860539407	0.0	5.0	0.3200000000000001	0.2

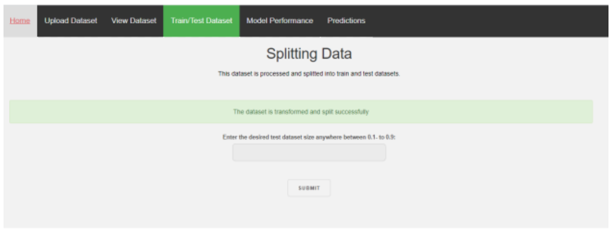
Train/Test dataset:



Splitting Data

This dataset is processed and splitted into train and test datasets.

Enter the desired test dataset size anywhere between 0.1- to 0.9:



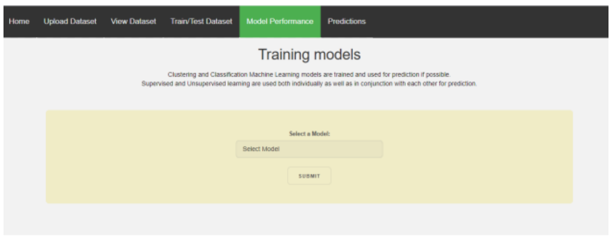
Splitting Data

This dataset is processed and splitted into train and test datasets.

The dataset is transformed and split successfully

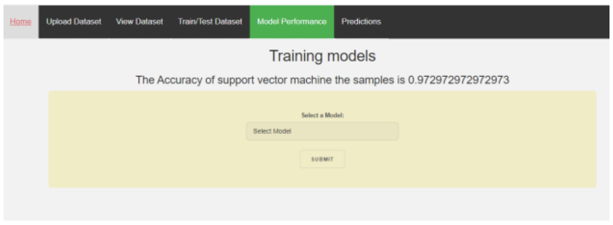
Enter the desired test dataset size anywhere between 0.1- to 0.9:

Model performance:



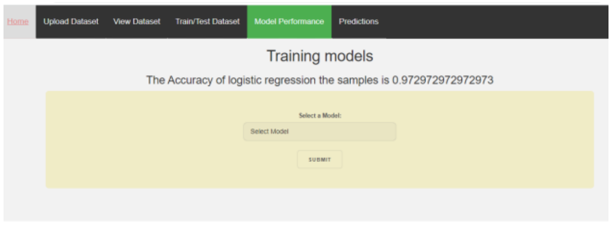
Training models

Clustering and Classification Machine Learning models are trained and used for prediction if possible.
Supervised and Unsupervised learning are used both individually as well as in conjunction with each other for prediction.



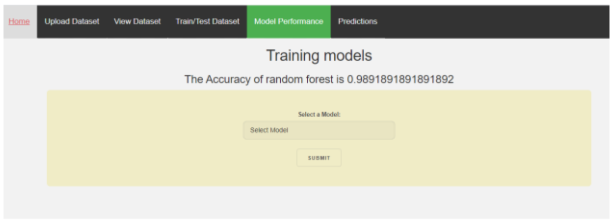
Training models

The Accuracy of support vector machine the samples is 0.972972972972973



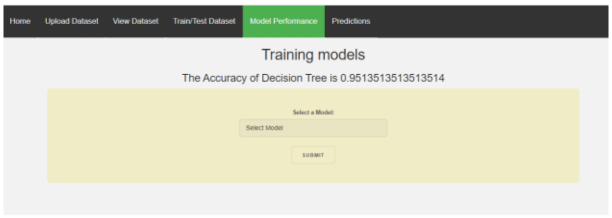
Training models

The Accuracy of logistic regression the samples is 0.972972972972973



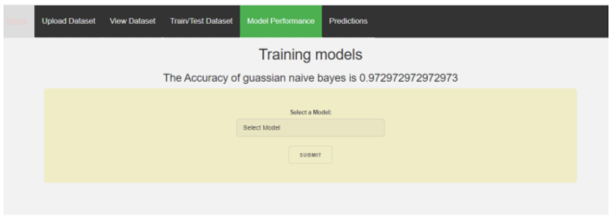
Training models

The Accuracy of random forest is 0.9891891891891892



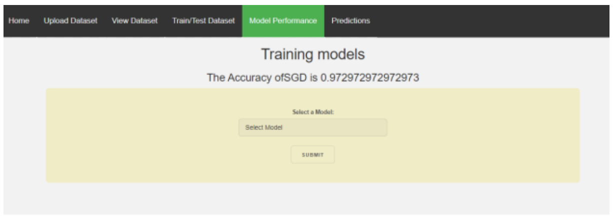
Training models

The Accuracy of Decision Tree is 0.9513513513513514



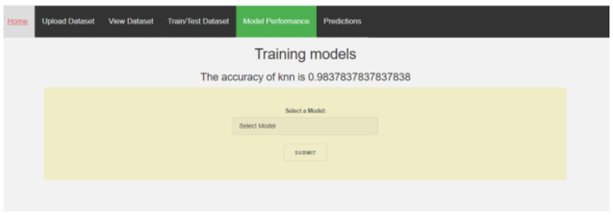
Training models

The Accuracy of guassian naive bayes is 0.972972972972973



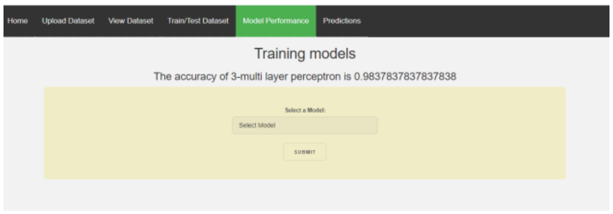
Training models

The Accuracy ofSGD is 0.972972972972973



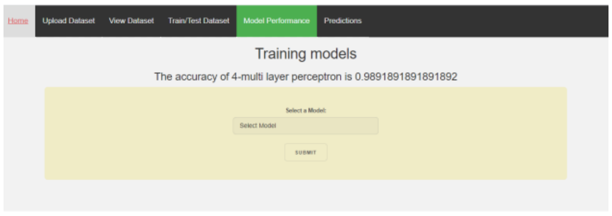
Training models

The accuracy of knn is 0.9837837837837838



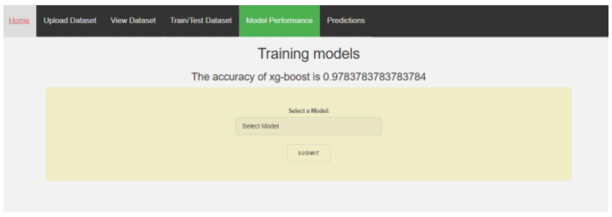
Training models

The accuracy of 3-multi layer perceptron is 0.9837837837837838



Training models

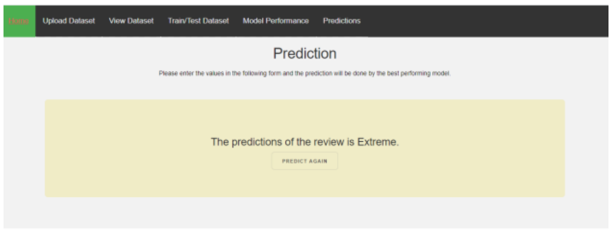
The accuracy of 4-multi layer perceptron is 0.9891891891891892



Training models

The accuracy of xg-boost is 0.9783783783783784

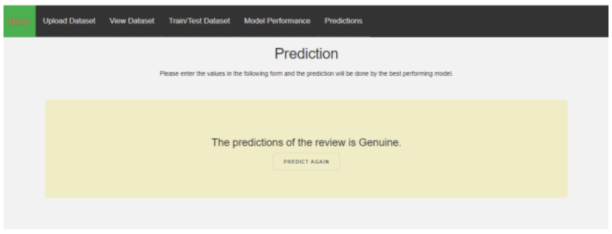
Predictions:



Prediction

Please enter the values in the following form and the prediction will be done by the best performing model.

The predictions of the review is Extreme.



Prediction

Please enter the values in the following form and the prediction will be done by the best performing model.

The predictions of the review is Genuine.

VI. CONCLUSION

In this article, we discussed an unexplored form of opinion spam, where spammers target brands as a whole, posting extreme reviews, to change the overall sentiment about the brand. These groups are often part of a complex business Web that is capable of influencing the overall popularity and reputation of several brands on review websites. This article is the first step toward linking brand-level group activities and extremism in reviews, which uncovers important insights about marketplace activities. These insights would help in developing a better recommendation that makes use of online reviews. A set of candidate spam groups was retrieved using FIM, and extremist groups were identified by observing their actions as a group based on various features, using a supervised learning technique based on a ground truth of manually annotated labels. We then classified extremist and moderate groups and compared the accuracy across multiple classification methods. After classifying these groups, we observed the behaviors for extremist groups in detail to gain further insights about the phenomenon and the overall trends of how these groups target these brands. We have also released the codes and annotated data set for further studies.

VII. REFERENCES

[1]. A. Kim. (2017). that review you wrote on Amazon? Priceless. [Online]. Available: https://www.usatoday.com/story/tech/news/2017/03/20/review-you-wrote-amazon-pricess/99332602/

[2]. E. Gilbert and K. Karahalios, “Understanding deja reviewers,” in Proc. ACM Conf. Comput. supported Cooperat. Work (CSCW), 2010, pp. 225–228, doi: 10.1145/1718918.1718961.

[3]. Amazon.in. (2018). Review Community Guidelines. [Online]. Available: https://www.amazon.in/gp/help/customer/display.html?nodeId= 201929730

[4]. A. Mukherjee, B. Liu, and N. Glance, “Spotting fake reviewer groups in consumer reviews,” in Proc. 21st Int. Conf. World Wide Web (WWW), 2012, pp. 191–200.

[5]. Y. Lu, L. Zhang, Y. Xiao, and Y. Li, “Simultaneously detecting fake reviews and review spammers using factor graph model,” in Proc. 5th Annu. ACM Web Sci. Conf. (WebSci), 2013, pp. 225–233.

[6]. S. Rayana and L. Akoglu, “Collective opinion spam detection: Bridging review networks and metadata,” in Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2015, pp. 985–994.

[7]. S. Dhawan, S. C. R. Gangireddy, S. Kumar, and T. Chakraborty, “Spotting collective behaviour of online frauds in customer reviews,” 2019,

arXiv:1905.13649. [Online]. Available: http://arxiv.org/abs/1905. 13649

[8]. K. Dave, S. Lawrence, and D. M. Pennock, “Mining the peanut gallery: Opinion extraction and semantic classification of product reviews,” in Proc. 12th Int. Conf. World Wide Web (WWW), 2003, pp. 519–528.

[9]. B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment classification using machine learning techniques,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2002, pp. 79–86.

[10]. K. Mouthami, K. N. Devi, and V. M. Bhaskaran, “Sentiment analysis and classification based on textual reviews,” in Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES), Feb. 2013, pp. 271–276.

[11]. Q. Ye, Z. Zhang, and R. Law, “Sentiment classification of online reviews to travel destinations by supervised machine learning approaches,” Expert Syst. Appl., vol. 36, no. 3, pp. 6527–6535, Apr. 2009.

[12]. M. Chelliah and S. Sarkar, “Product recommendations enhanced with reviews,” in Proc. 11th ACM Conf. Recommender Syst., Aug. 2017, pp. 398–399.

[13]. L. Chen and F. Wang, “Preference-based clustering reviews for augmenting e-commerce recommendation,” Knowl.-Based Syst., vol. 50, pp. 44–59, Sep. 2013.

[14]. J. Feuerbach, B. Loepp, C.-M. Barbu, and J. Ziegler, “Enhancing an interactive recommendation system with review-based information filtering,” in Proc. IntRS@RecSys, 2017, pp. 10–55.

Cite this article as :

Jothika. J, Nalagampalli MoniSree, Pavithra. M, "Detecting and Characterizing Extremist Reviewer Groups in Online Product Reviews", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8 Issue 3, pp. 342-353, May-June 2022.

Journal URL : https://ijsrcseit.com/CSEIT2283107