iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VIII **Month of publication:** Aug 2023

DOI: https://doi.org/10.22214/ijraset.2023.55210

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VIII Aug 2023- Available at www.ijraset.com

Sarcasm Detection in News Headlines: A Comparative Study of Multinomial Naive Bayes, Logistic Regression, and Support Vector Machine

Aryan Shetty[1], Aditya Rajiv Singh[2], Praful Shukla[3], Aryan Shetye[4]
[1, 2, 3, 4]*Dept. of Information Technology, Thakur College of Engineering and Technology, Mumbai*

Abstract: ***Sarcasm is frequently used in news, and it is difficult to detect sarcasm in news headlines for humans and let alone computers. Media outlets constantly make use of sarcasm in their news headlines to target certain set of people and to subtly spread misinformation by confusing viewers, who, in turn, tend to spread their misinterpreted news to their contacts. Hence, it is very important to develop a system that can accurately detect sarcasm. This study proposes a unique method for sarcasm detection in news headlines that makes use of the Multinomial Naive Bayes, Logistic Regression, and Support Vector Machine classifiers with Bag-of-Words representation. We collected and combined two datasets from Kaggle of labeled headlines for training and evaluation. Using the CountVectorizer, we transformed headlines into numerical vectors capturing word occurrences. Our model yielded great test set accuracies of roughly 87.15%, 89.93%, and 90.6% for Multinomial Naive Bayes, Logistic Regression, and Support Vector Machine respectively. The precision, recall, and F1-score demonstrated balanced detection capability, and were also high enough to accurately detect sarcasm. Finally, our study was able to compare the three classifiers and point out each classifier's strengths and weaknesses, while also suggesting the best classifier for sarcasm detection of news headlines.***

Keywords: Sarcasm detection, news headlines, Support Vector Machine classifier, Multinomial Naive Bayes classifier, Logistic Regression classifier, Bag-of-Words representation.

I. INTRODUCTION

A. Background and Motivation

Sarcasm is something we come across everyday. It is defined as the use of words that mean the opposite of what you really want to say. This is especially done in order to insult someone, or to show irritation, or just to be funny. Sarcasm Detection is the task of recognizing and predicting sarcasm in a text as understanding people's true sentiments, opinions, and views is very important. [1] The majority of sarcasm detection methods up to this point have mostly focused on text categorization. [2] Such text based approaches can sometimes be accurate but the are completely devoid from the notion of context. Human's speak in accordance with surrounding context, so ignoring context in our algorithm can hinder the accuracy of the algorithm, potentially giving an incorrect output. Hence, sarcasm detection is also considered as a very difficult task to implement as it involves understanding and depending on context, tone, and prior knowledge in which the sentence was written or spoken. In this study we utilized Multinomial Naive Bayes, Logistic Regression, and Support Vector Machine classifiers, as these can work significantly well in attaining our motive of building a sarcasm detection model based on news headlines. And once this model was built, we compared the results of the three classifiers to determine the best classifier to achieve the desired results for sarcasm detection. Readers' perceptions of an issue can be dramatically impacted by news headlines. And many times, this influences the public's reaction towards it. So, we examine the difficulty of detecting sarcasm by highlighting news headlines. Journalists, news aggregators, and moderators should identify humorous headlines appropriately in order to provide a thorough and objective assessment of public perceptions of news events, comparing various different models to determine the best model to achieve the desired results.

B. Problem Statement

While creating this model, our primary goal was to make such a sarcasm detection system that is not only precise but overall very effective. Our method takes into account the brevity and wordplay frequently found in this field and aims to distinguish between sarcastic and non-sarcastic headlines. We aim to create a model in order to effectively identify sarcasm in headlines across a variety of news topics and genres.

©IJRASET: All Rights are Reserved

C. Objectives

The specific objectives of this research are as follows:

1) *Data Collection and Preprocessing:* We gathered datasets of news headlines that have labeled sarcasm indicators and preprocess this data to ensure uniformity and cleanliness.
2) *Model Development:* We aimed to build a robust sarcasm detection model by leveraging the Support Vector Machine, Multinomial Naive Bayes, and Logistic Regression classifiers, and comparing them to assess which classifier gives the best results.
3) *Bag-of-Words Representation:* We implemented the Bag-of-Words (BoW) approach to convert the processed headlines into numerical vectors. By doing this, each headline was represented as a high-dimensional vector, capturing the word frequencies.
4) *Model Evaluation and Performance Analysis:* We evaluated the model's accuracy, precision, recall, and F1-score on a test set of news headlines and analyzed its strengths and weaknesses.

D. Scope of the Research

The sole subject of this study is the identification of sarcasm in news headlines. Although, there are many domains where sarcasm can be detected, news headlines require specialized attention due to their distinctive features. Support Vector Machine is one of the most popular supervised learning algorithms. And it is also often used to solve classification and regression issues, mostly in applications such as handwriting and face recognition. So, we thought of using it. Another effective algorithm for dealing with problems involving multiple classes and text data analysis is Multinomial Naive Bayes. [2] Many NLP applications, including marketing analysis, opinion mining, and information categorization, can benefit from the application of sarcasm detection. The research utilizes datasets for training and evaluation that have a labeled sarcasm indicator. Hence, at the end of this paper, we aim to establish different approaches for sarcasm detection, and based on experimental analysis, our objective is to discover the most accurate model to predict and detect sarcasm on a given dataset.

II. RELATED WORK

In recent times, deep learning has been useful in detecting sarcasm. This is due to CNN's decent track record in being able to solve NLP issues. Using CNN, Poria *et al.* [3] made use of four datasets to extract four feature sets. Then these feature sets were merged and classified using an SVM classifier. Sarcastic tweets that contain hashtag keywords like #not, #sarcastic, #sarcasm are commonly available in sarcasm detection datasets [4]. Some studies [5] believe that these hashtags are the perfect indicators for sarcasm detection initially.

However, apart from the exclusive reliance on hashtags, a few studies also added rule-sets. One of these was Barbieri *et al.* [6] who used the rarity and frequency of words as their primary features. Bouazizi *et al.* [7] also used this technique but with some extra rules on deriving sarcastic word patterns. Their rules included counting the number of positive/negative words in a particular tweet, and also counting the number of greatly emotional positive/greatly emotional negative words. In a more recent study, Shmueli *et al.* [8] made use of a seed phrase "being sarcastic" as in "He was being sarcastic" or "They were being sarcastic". This seed was hence used in collecting sarcastic instances in twitter. This helped in creating a new dataset and also solving the problem of dataset scarcity for sarcasm detection. Riloff *et al.* [5] had used classifiers that looked for positive verbs found together with negative circumstances in a sentence. They created a whole lexicon for these verbs and situational words. This lexicon was then used to distinguish between sarcastic and ordinary sentences in their test. Their research motivated other research to use the same methodology of utilizing hashtags precipitated detection expanded by rule-sets [9].

As observed, a lot of research has been conducted using twitter datasets, but research that uses news headlines for sarcasm detection is scarce. Hence, with our research we'll be contributing to the lesser explored region of sarcasm detection.

III. DATA COLLECTION AND PREPROCESSING

A. Description of News Headlines Datasets

To ensure our research is top-notch, data collection plays a crucial role in shaping the model's performance and grasp of the subject matter. To tackle the detection of sarcasm in news headlines, we embarked on a quest to find datasets that contained well-annotated headlines showcasing indicators of sarcasm. After diligently scouring various publicly accessible sources, we handpicked two different datasets from Kaggle to construct an all-encompassing corpus for both training and evaluation purposes [10]. A variety of news headlines are included in the "Sarcasm_Headlines_Dataset.json" dataset, along with binary labels designating whether or not each headline is sarcastic (class label=1).

©IJRASET: All Rights are Reserved

Each entry in the dataset contains the three attributes “is_sarcastic,” “headline,” and “article_link.” The “is_sarcastic” attribute indicates whether the headline is sarcastic. The news headline text is contained in the “headline” attribute. The “article_link” attribute, but on the other the hand, acts as a hyperlink to the original news item. Our model can recognize sarcasm in a variety of contexts because of the dataset’s extensive coverage of news topics. We aimed to combine both datasets (“Sarcasm_Headlines_Dataset.json” and “Sarcasm_Headlines_Dataset_v2.json”) to increase the size and diversity of our corpus, improving the ability to generalize across various styles of headlines. The second dataset’s labeled headlines and corresponding attributes have a similar structure to the first dataset’s.

B. Data Cleaning and Preprocessing

To ensure a more accurate detection of sarcasm, we conducted various operations to refine and augment the text data. Our actions included the elimination of special characters, changing text to lowercase, extracting meaningful words, employing lemmatization, and dividing the headlines into individual units. These efforts guarantee that the model only learns from significant information. Using the `re.sub()` function along with the regular expression pattern `r'[^a-zA-Z\s]'`, we began by eliminating certain elements from the headlines. This included special characters, digits, punctuation marks, and symbols. This important initial step allowed us to concentrate solely on the textual content and discard any distracting noise that would not greatly aid in detecting sarcasm. Next, we used the `.lower()` method to convert all text to lowercase. By achieving uniformity throughout the dataset, this step stopped the model from considering words with different capitalizations as separate features. By doing this, regardless of whether a word is capitalized, the model can learn its semantic meaning. Common stop words like “the,” “and,” and “of” were then eliminated. Although they frequently appear in headlines, these words lack any significant semantic meaning. Eliminating stop words decreased data dimensionality, sped up processing, and improved sarcasm detection accuracy by focusing the model’s attention on more illuminating words. Lemmatization was then used to further enhance model performance. In order to ensure that different forms of a word are treated as having the same feature, lemmatization reduces words to their base or root form. The model can more effectively capture patterns related to sarcasm by combining word variations into a single representation. The `word_tokenize()` function from the NLTK library was used to tokenize the headlines. The Bag-of-Words representation, which is necessary for processing text data, was created by tokenizing the headlines into individual words or tokens. By giving the model this level of detail, it was able to comprehend the textual content better and identify patterns that were indicative of sarcasm. We loaded and combined the datasets using Pandas to carry out these data cleaning and preprocessing steps. Then, to apply these operations to the dataframe’s “headline” column, we defined the `clean_text` function. The ‘headline’ column was then cleaned and preprocessed using the `.apply()` method, making it ready for additional analysis and sarcasm detection modeling. We developed a high-quality and noise-free corpus by adhering to this stringent data cleaning and preprocessing pipeline, which allowed the model to generalize effectively and accurately identify sarcasm in news headlines across various domains.

C. Dataset Statistics

We performed a statistical analysis of the dataset after cleaning and preprocessing the data to learn more about the distribution of sarcastic and non-sarcastic headlines. There are 55328 headlines in the combined dataset, of which 45.83% are classified as sarcastic and 54.17% as not. The distribution of sarcasm labels makes sure that both classes are fairly represented, preventing the model from favoring the class that is more numerous. The dataset’s headlines had an average length of 9.95 words, ranging from 2 words at the shortest length to 151 words at the longest. Taking into account the brevity of news headlines, this information informs the model's design and provides an overview of the dataset's characteristics [11].

Fig. 1 Dataset 1 Word Cloud

Fig. 2 Dataset 2 Word Cloud

Fig. 1: This word cloud visualizes the word frequency distribution from the "Sarcasm_Headlines_Dataset.json" file. The sizes of words reveal their frequency, revealing linguistic patterns in sarcastic headlines. Fig. 2: This word cloud displays common vocabulary and represents "Sarcasm_Headlines_Dataset_v2.json". Font sizes show word frequency, giving readers a glimpse into sarcastic headlines' linguistic traits.

IV. METHODOLOGY

A. Bag of Words Representation

Vectorization is the process in which the input data text is converted into vectors which are mainly used in training the deep learning or machine learning model. In natural processing, transforming the text data into numerical vectors is an important function and this technique is best represented by the Bag-of-Words (BoW) [12]. A vector represents a headline in which number of specific words represent the element in the headline.

The main advantage of BoW is that it exclusively focuses on the occurrence of word and eliminates the word order which allows the model to process text efficiently. Consider the following sentence, "John likes to watch movies". Now the given sentence is pre-processed as we mentioned in previous section, the sentence converted to five words – 'John,' 'likes,' 'to,' 'watch,' 'movies' [13]. Once the words are generated a histogram is created. Histogram is referred as frequency of words in each sentence or document. Now this histogram representation is converted into vector form, so basically this how the BoW works, and conversion of vector takes place.

CountVectorizer()` is an important tool provided by the scikit-learn library in python. With the help of `CountVectorizer()` it creates Bag-of-Words which represents the headlines present in the dataset. The main feature about the 'CountVectorizer' is it collects the text documents and converts it into a matrix of word count. Now the matrix which is created, its row corresponds to documents and each column corresponds to unique word in headline. Once the matrix is created, we move to the fitting and transforming the data. The corpus comprises of various unique words which are identified and collected as a vocabulary which can be used in representation of BoW. After the learning of vocabulary, the training headlines are transformed into numerical vectors.

B. Multinomial Naive Bayes

To perform the text classification task, Multinomial Naive Bayes is an efficient algorithm to deal with large text dataset. The primary task of Multinomial Naive Bayes is text classification, topic classification, including sentimental analysis, and spam detection.

The data is pre-processed by removing the punctuation, stop word, grammars, and converting the text to lowercase. In these the news headlines are annotated as sarcastic and non-sarcastic which they have collected from the labelled dataset of news headlines. Once the data is pre-processed the conversion of text headlines into numerical feature vector is performed. As most of the machine learning algorithms they require numerical input for their computations which makes the model to perform various text classification task efficiently.

Also, it reduces the dimensionality of the large text dataset which makes it more manageable and potentially improving the performance of machine learning.

The text dataset which we have is now split into two training and testing sets. The training set is now fed into the model which we have created to detect the sarcasm in the news headlines. To assess the effectiveness, we test the dataset and evaluate the performance of trained model.

C. Logistic Regression

It is one of the classifier algorithms that uses a simple sigmoid function and has a regression function. This model assumes independence, and probability is utilized as a judgement factor to decide the class. It is also considered as a powerful binary classification algorithm. Implementation of logistic regression can be done by using various programming algorithm and learning libraries. We used the popular 'scikit-learn' library in python for importing Logistic Regression classifier.

D. Support Vector Machine

An algorithm for supervised machine learning is Support Vector Machine. SVM is applicable to both classification as well as regression issues. SVM's main objective is to create the optimum decision boundary that can divide an n-dimensional space into distinct classes [14].

The selected hyperplane must offer the greatest possible separation between datapoints of different classes. The extreme points or vectors that aid in the formation of the hyperplane are known as support vectors. There are two SVM subtypes, the ones that are linear and the ones that are non-linear. By using a single straight line, linear SVM may divide data into two classes in a linear manner.

Data can be divided using a non-linear SVM. The SVM classifier distinguishes between classifier and non-classifier with the help of decision boundary [9]. The position relative to the decision boundary helps it to classify whether the headline is sarcastic or non-sarcastic.

Another advantage of SVM classifier is that it is very much effective in handling the high dimensional feature spaces, which makes appropriate for the Bag-of-Words with numerous unique vocabularies.

Various text documents are converted into the BoW vectors which comprises of word frequency or term frequency in the document. Now the role of SVM classifier is it takes this as input vectors and distinguishes the class with the help of decision boundary which is also termed as hyperplane. The primary goal of SVM is to find the best hyperplane, which is the plane that is farthest from both classes with the BOW vectors [12]. In short, these numerical vectors can be used in conjunction with the machine learning method SVM to carry out text classification tasks. They are connected together as a result when BoW is used as a feature representation for SVM in NLP tasks.

V. EXPERIMENTAL SETUP AND EVALUATION METRICS

When starting with our code, the first step was to consider the essential libraries for the construction of our model. As we were working in a Python environment, the library we used for data manipulation was ‘pandas’; this helped us to represent our data in a tabular manner, which made it easier to perform operations on the dataset. We needed regular expressions for a single use, but an important one was to remove special characters and digits from the data’s text. For this purpose, we imported the ‘re’ library and used the ‘re.sub’ function. Further imports comprised of ‘train-test-split’ and ‘CountVectorizer’ for splitting data for training and testing and creating the Bag-of-Words representation, respectively. At this point, the only remaining imports were the classifiers, and we decided to test our data with three classifiers and choose the best fit for our model. The three classifiers we chose were ‘MultinomialNB’, ‘LogisticRegression’, and ‘SVC’. These models are capable of providing satisfactory results for sarcasm detection; hence, we limited our choices to these three. As data cleaning and preprocessing have already been discussed in the previous sections, we will now skip to the data-splitting phase. We used the typical route of training to testing ratio of 80:20 because the Pareto Principle is the best split possible for our sarcasm detection data split. We created two variables, ‘X’ and ‘y’ containing the preprocessed news headlines (‘headline’ column of ‘data’) and the corresponding labels (‘is_sarcastic’ column of ‘data’) respectively. Then to carry out the actual training and testing split we utilized the ‘train_test_split()’ function which basically shuffles the data and splits it into 80% training data (‘X_train’, ‘y_train’) and 20% testing data (‘X_test’, ‘y_test’). The proportion of the data to be tested was specified by the ‘test_size=0.2’ parameter, and ‘random_state=42’ was used to set the random seed for reproducibility. The use of the BoW representation allowed us to transform each headline into a numerical vector, with each element corresponding to the frequency of a specific word in the headline. Our Bag-of-Words representation has been touched upon in detail in the previous section; hence, we now move on to the major point of our code, that is, training our classifiers. The first classifier we trained was the Multinomial Naive Bayes classifier (clf_nb). This is initialized with the MultinomialNB class, and is a probabilistic model commonly used for test classification tasks. We set the ‘alpha=1.0’ parameter to apply Laplace smoothing during probability estimation, which helps prevent issues with zero probabilities for unseen words in the training data. Next, we trained the classifier using the ‘fit’ method by passing the Bag-of-Words representation of the training headlines (‘X_train_bow’) and their corresponding labels (‘y_train’). In this way, the model learns to classify headlines as either sarcastic (1) or non-sarcastic (0) based on word occurrences in the Bag-of-Words representation. The next classifier that we used was Logistic Regression (clf_lr), a linear classification model that works well for binary classification tasks. We initialized it with the LogisticRegression class.

We set the ‘max_iter=1000’ parameter to limit the maximum number of iterations for convergence, and ‘random_state=42’ to ensure reproducibility of the results. Similar to the Multinomial Naive Bayes classifier, we trained the Logistic Regression classifier using the fit method on the Bag-of-Words representation of the training headlines (‘X_train_bow’) and their corresponding labels (‘y_train’). The final classifier we used was the Support Vector Machine (clf_svm), which was initialized with the SVC class. A linear kernel is suitable for text classification tasks containing high-dimensional feature spaces; therefore, we specified the ‘kernel=linear’ parameter. Again, we set the ‘random_state=42’ parameter is set to ensure reproducibility of results and further trained the SVM classifier with the ‘fit’ method with the Bag-of-Words representation of the training headlines (‘X_train_bow’) and their corresponding labels (‘y_train’). Thus, we were ready to perform predictions using our classifiers.

Now was the time to start making our predictions on the test set using the models we had created. We started by setting the 'threshold' to 0.5 as this gave a fair probabilistic measure to determine the headline's class label based on the predicted probabilities. How this works is that if the sarcasm probability is higher than the threshold value, the headline is classified as sarcastic (1), and non-sarcastic (0) otherwise. The 'predict_proba' method is used for making predictions for each classifier. This functions by returning the probabilities of each headline belonging to each class (sarcastic and non-sarcastic); for example, the predicted probabilities of the Multinomial Naive Bayes classifier are contained in 'y_pred_probs_nb'. The final step is to adjust the classifiers' predictions using the threshold for the obtained predicted probabilities. In the case of the Multinomial Naive Bayes and Logistic Regression classifiers, we are interested in the sarcastic (class label 1) probabilities, which are assessed using '[:, 1]' from the 'y_pred_probs_nb' and 'y_pred_probs_lr' arrays. On the other hand, for the Support Vector Machine classifier, we used the decision function ('decision_function') to obtain the decision values, following which the threshold is directly applied to this value. Hence, by comparing the predicted probabilities or decision values to the threshold, we obtained the adjusted predictions ('y_pred_adjusted_nb', 'y_pred_adjusted_lr', and 'y_pred_adjusted_svm').

With this, our experimental setup was complete, and the only thing left was performance evaluation. We took several steps in forming our experimental setup, making careful considerations in establishing the most feasible configurations and procedures which can be used to evaluate our sarcasm detection model's performance. For this, we chose some evaluation metrics, which we explain below.

The counts of test records that the classification model correctly and erroneously predicted are used to assess the performance of the model. The confusion matrix gives a more insightful picture of a predictive model's performance, showing which classes are forecasted correctly and erroneously as well as the kind of errors that are being made. To give an example, the confusion matrix table below (Table 1) clearly shows how the four categorization metrics (TP, FP, FN, and TN) are produced as well as how our predicted value compares to the actual value.

TABLE I
CONFUSION MATRIX

		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

- True Positive (TP): Observation is positive and is predicted to be positive.
- False Negative (FN): Observation is positive but is predicted negative.
- True Negative (TN): Observation is negative and is predicted to be negative.
- False Positive (FP): Observation is negative but is predicted to be positive.

Now we'll discuss the metrics of confusion matrix that we utilized.

The first metric we used is accuracy. Accuracy is the measure of how often the classifier predicts correctly. It is the ratio of the number of correct predictions and the total number of predictions, see Formula 1.

$$Accuracy : A = \frac{tp + tn}{tp + fp + fn + tn} \#(1)$$

When a model gives an accuracy of 99%, it might not be true that the model is performing flawlessly, and this can be misleading in some situations. Accuracy proves useful when our target class is well balanced, but it is not the best choice when our classes might be unbalanced. To tackle the issues of using only one metric, we decided to use several of them to improve our evaluation statistics.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VIII Aug 2023- Available at www.ijraset.com

Precision and Recall are the next metrics we used. Precision explicates how many of the accurately predicted cases actually turned out to be positive and is useful in cases where False Positives are a greater concern than False Negatives. Precision for a label is defined as the number of true positives divided by the number of predicted positives, see Formula 2.

$$Precision: P = \frac{tp}{tp + fp} \#(2)$$

Recall describes how many of the actual positive cases our model was able to properly predict. When False Negative is more important than False Positive, it is a valuable metric. Recall for a label is defined as the number of true positives divided by the total number of actual positives, see Formula 3.

$$Recall: R = \frac{tp}{tp + fn} \#(3)$$

[15] Depending on the weight function β, the F-score is described as the weighted average of both precision and recall. see Formula 4. The F_1-score means the harmonic mean between precision and recall, see Formula 5, when it is written F-score it usually means F_1-score. The F-score is also called the F-measure. The F_1-score can have different indices giving different weights to precision and recall.

$$F\text{-}score: F_\beta = (1 + \beta^2) * \frac{P * R}{\beta^2 * P + R} \#(4)$$

With β = 1 the standard F-score is obtained, see Formula 5.

$$F\text{-}score: F_1 = F = 2 * \frac{P * R}{P + R} \#(5)$$

This concludes our experimental setup and evaluation metrics criteria, and the next section will highlight our results and the corresponding analysis and discussion of the three classifiers' performance.

VI. RESULTS AND ANALYSIS

Now that we had setup our model and set the evaluation metrics that we would be using to analyze our model's performance, we created the classification reports, confusion matrices, and visualizations to intuitively understand how each classifier was performing in detection sarcasm in the news headlines. We'll now delve into the findings and compare the capabilities of each classifier.

A. Model Performance Metrics

As discussed in the previous section, we compared the classifiers based on accuracy, precision, recall, and F-1 score to gain a comprehensive overview of how the classifiers were performing.

We discuss the results below.

1) *Accuracy:* Our goal was to achieve an accuracy close to 90%, and our model didn't disappoint on that end. All the classifiers were very close in their accuracies. The accuracy of the Multinomial Naive Bayes was 87.15%, which we expected to be the lowest. Logistic Regression on the other hand was very impressive with an 89.93% accuracy, almost touching our goal. However, Support Vector Machine performed the best for sarcasm detection with a extraordinary 90.6% accuracy. The exact accuracies can be viewed in Fig. 3.

```
Accuracy (Multinomial Naive Bayes): 0.8715886499186698
Accuracy (Logistic Regression): 0.8993312850171697
Accuracy (Support Vector Machine): 0.9060184348454726
```

Fig. 3 Accuracy of our classifiers

©IJRASET: All Rights are Reserved

2) *Classification Report:* We used classification reports to aggregate all the metrics like precision, recall, and F-1 score. This also contains other metrics like support, macro average, and weighted average. In terms of precision, the models performed as follows: The Multinomial Naive Bayes classifier had same precisions for both class 0 (non-sarcastic headlines) and class 1 (sarcastic headlines), them being 0.87. Logistic Regression's precisions were quite close to each other, them being 0.89 for class 0 and 0.91 for class 1. Support Vector machine's precisions were not as tight though, at 0.87 precision for class 0 and 0.96 precision for class 1. The next metric in the classification report is recall, and we observed a bit more variance when it came to this. The recall for the Multinomial Naive Bayes classifier was 0.89 for class 0 and 0.85 for class 1. A similar result was shown in Logistic Regression as well with a recall of 0.93 for class 0 and 0.87 for class 1. Support Vector Machine, on the other hand, had a higher variability with an impressive recall of 0.97 for class 0, but only 0.83 for class 1. Lastly the results for the F-1 score were as such: Multinomial Naive Bayes had an F-1 score of 0.88 and 0.86 for class 0 and class 1 respectively. With Logistic Regression, we got F-1 scores of 0.91 for class 0 and 0.89 for class 1. Finally, Support Vector Machine gave the best results with F-1 scores of 0.92 for class 0 and 0.89 for class 1. The complete classification report for the Multinomial Naive Bayes (Fig. 4), Logistic Regression (Fig. 5), and Support Vector Machine (Fig. 6) classifiers can be viewed below.

Classification Report (Multinomial Naive Bayes):

	precision	recall	f1-score	support
0	0.87	0.89	0.88	5878
1	0.87	0.85	0.86	5188
accuracy			0.87	11066
macro avg	0.87	0.87	0.87	11066
weighted avg	0.87	0.87	0.87	11066

Fig. 4 Classification Report for Multinomial Naive Bayes classifier

Classification Report (Logistic Regression):

	precision	recall	f1-score	support
0	0.89	0.93	0.91	5878
1	0.91	0.87	0.89	5188
accuracy			0.90	11066
macro avg	0.90	0.90	0.90	11066
weighted avg	0.90	0.90	0.90	11066

Fig. 5 Classification Report for Logistic Regression classifier

Classification Report (Support Vector Machine):

	precision	recall	f1-score	support
0	0.87	0.97	0.92	5878
1	0.96	0.83	0.89	5188
accuracy			0.91	11066
macro avg	0.92	0.90	0.90	11066
weighted avg	0.91	0.91	0.91	11066

Fig. 6 Classification Report for Support Vector Machine classifier

These classification reports helped us in inferring that the Support Vector Machine (SVM) classifier clearly outperform both Multinomial Naïve Bayes and Logistic Regression classifiers. In terms of precision, recall, and F-1 score for both classes SVM provided the best results, and in terms of accuracy (90.6%) as well it was the highest. SVM demonstrated superior performance in differentiating sarcastic headlines from non-sarcastic headlines. Hence, the classification report provided valuable insights into both the strengths and weaknesses of each classifier, in turn, aiding us in nominating the most apt model to perform sarcasm detection tasks on news headlines.

Apart from the classification report we also created confusion matrices to summarize each classifier's true positive (TP), false positive (FP), true negative (TN), and false negative (FN) predictions. We'll now explain the interpretations we acquired from each confusion matrix.

The Multinomial Naive Bayes classifier was able to correctly classify 4418 sarcastic headlines (TP) and 5227 non-sarcastic headlines (TN). But it also misclassified 651 non-sarcastic headlines as sarcastic (FP) and 770 sarcastic headlines as non-sarcastic (FN). Through this confusion matrix we were able to infer that Multinomial Naive Bayes displays satisfactory performance but has some difficulty in accurately identifying sarcastic headlines, in particular, the false negative cases.

Next, the Logistic Regression classifier correctly identified 4495 sarcastic headlines (TP) and 5457 non-sarcastic headlines (TN). On the misclassification end, it inaccurately predicted 421 non-sarcastic headlines as sarcastic (FP) and 693 sarcastic headlines as non-sarcastic (FN). Overall, we noticed that Logistic Regression performs slightly better than Multinomial Naive Bayes.

Finally, we studied the confusion matrix of Support Vector Machine and identified the best results. It correctly identified 4316 sarcastic headlines (TP) and 5710 non-sarcastic headlines (TN), while misclassifying only 168 non-sarcastic headlines as sarcastic (FP) and 872 sarcastic headlines as non-sarcastic (FN). The SVM classifier demonstrated the best performance among all the classifiers by effectively distinguishing between sarcastic and non-sarcastic headlines. These confusion matrices helped us collecting detailed information about each classifier's performance and assessing their strengths and weaknesses in detecting sarcasm in news headlines. Confusion matrices of the three classifiers can be viewed below in Fig. 7.

```
Confusion Matrix (Multinomial Naive Bayes):
 [[5227  651]
 [ 770 4418]]
Confusion Matrix (Logistic Regression):
 [[5457  421]
 [ 693 4495]]
Confusion Matrix (Support Vector Machine):
 [[5710  168]
 [ 872 4316]]
```

Fig. 7 Confusion Matrices of our classifiers

Finally, we created some visualizations to better understand the three classifiers' performance in an intuitive manner. The bar plot of accuracy comparison (Fig. 8) and heat maps of confusion matrices (Fig. 9) are added below.

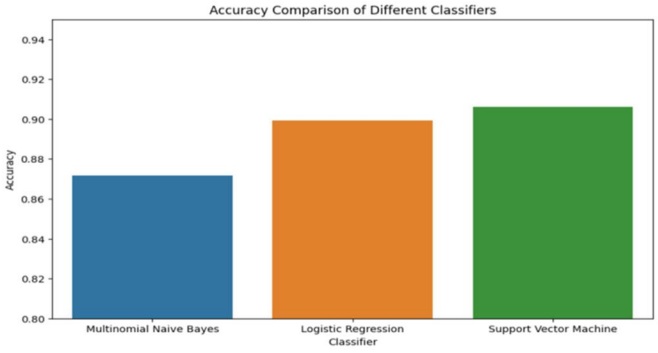


Fig. 8 Accuracy Comparison of Classifiers

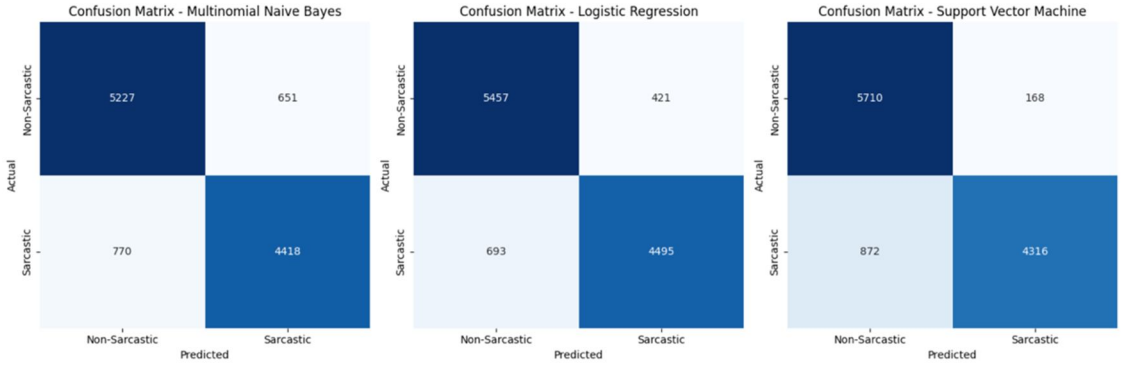


Fig. 9 Confusion Matrices Heat Maps

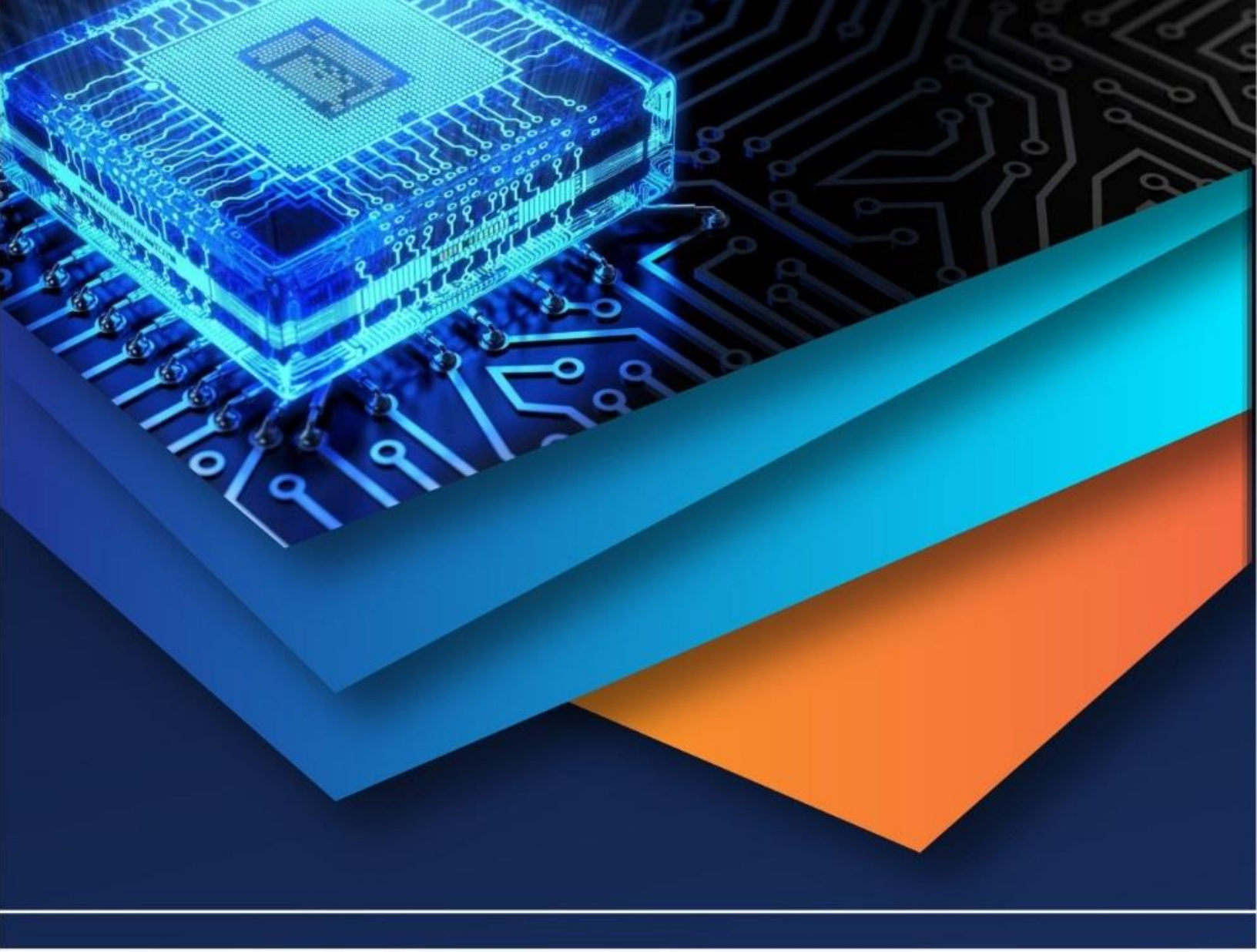
With this, we're done with our results and analysis section, and have completed our model through comparison. We achieved fantastic results with each classifier, successfully creating a model that effectively identifies sarcasm in news headlines. In the next section we'll conclude our research and suggest some future advancement that can be made in this field of research.

VII. CONCLUSIONS

Sarcasm detection is a difficult task due to several nuances in text and context. It is a challenging task to build an efficient sarcasm detection model, but we successfully built one with our extensive research. We were able to do this by utilizing the Bag-of-Words representation and comparing three classifiers, namely, Multinomial Naive Bayes, Logistic Regression, and Support Vector Machine. The primary evaluation metrics we used in our research were accuracy, precision, recall, and F1-score. All of the three classifiers that we used yielded excellent results against these metrics, the most impressive being Support Vector Machine. By analyzing the false positives and negatives we unveiled significant insights into our model's strengths and limitations, and this emphasized the importance of contextual understanding and advanced NLP techniques. In-depth investigations of misclassifications enriched our understanding of future research possibilities. Prospective research directions for the future scope include employing pre-trained contextual models like BERT or RoBERTa for enhanced content understanding and exploring transfer learning for diverse datasets and languages. Our research emphasizes the efficiency of our sarcasm detection model, contributing to the field of natural language processing and sentiment analysis. By continuing to refine our model, we foresee significant impact on enhancing sarcasm detection and understanding human communication.

REFERENCES

[1] D. Bamman and N. A. Smith, "Contextualized sarcasm detection on twitter," in Proc. Int. AAAI Conf. Web Social Media, 2015, pp. 1-4.
[2] L. Xu and V. Xu, "Project Report: Sarcasm Detection," published in, 2019.
[3] S. Poria, E. Cambria, D. Hazarika, and P. Vij, "A deeper look into sarcastic tweets using deep convolutional neural networks," 2016, arXiv:1610.08815. [Online]. Available: http://arxiv.org/abs/1610.08815
[4] D. Davidov, O. Tsur, and A. Rappoport, "Semi-supervised recognition of sarcasm in twitter and amazon," in Proc. 14th Conf. Comput. Natural Lang. Learn., 2010, pp. 107-116.
[5] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, "Sarcasm as contrast between a positive sentiment and negative situation," in Proc. Conf. Empirical Methods Natural Lang. Process., 2013, pp. 704-714.
[6] F. Barbieri, H. Saggion, and F. Ronzano, "Modelling sarcasm in twitter, a novel approach," in Proc. 5th Workshop Comput. Approaches Subjectivity, Sentiment Social Media Anal., 2014, pp. 50-58.
[7] M. Bouazizi and T. Otsuki Ohtsuki, "A pattern-based approach for sarcasm detection on Twitter," IEEE Access, vol. 4, pp. 5477–5488, 2016.
[8] B. Shmueli, L.-W. Ku, and S. Ray, "Reactive supervision: A new method for collecting sarcasm data," 2020, arXiv:2009.13080. [Online]. Available: http://arxiv.org/abs/2009.13080
[9] A. Joshi, P. Bhattacharyya, and M. J. Carman, "Automatic sarcasm detection: A survey," ACM Comput. Surv., vol. 50, no. 5, pp. 1–22, 2017.
[10] P. Shrikhande, V. Setty and D. A. Sahani, "Sarcasm Detection in Newspaper Headlines," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), RUPNAGAR, India, 2020, pp. 483-487.
[11] R. Misra and P. Arora, "Sarcasm detection using news headlines dataset," AI Open, vol. 4, 2023, pp. 13-18.
[12] A. Almestekawy and M. Abdulsalam, "Sentiment Analysis of Product Reviews Using Bag of Words and Bag of Concepts," I.J. of Electronics and Information Engineering, vol. 11, no. 2, 2019, pp. 49-60.
[13] J. Aboobaker and E. Ilavarasan, "A Survey on Sarcasm Detection Approaches," Indian Journal of Computer Science and Engineering, vol. 11, no. 6, pp. 751-771
[14] S. Gaonkar, S. Itagi, R. Chalippatt, A. Gaonkar, S. Aswale and P. Shetgaonkar, "Detection Of Online Fake News : A Survey," 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 2019, pp. 1-6, doi: 10.1109/ViTECoN.2019.8899556.
[15] H. Dalianis, Evaluation Metrics and Evaluation. In: Clinical Text Mining, Springer-Cham, 2018, https://doi.org/10.1007/978-3-319-78503-5_6



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)