DETECTION OF SARCASM IN NEWS HEADLINES

[1]Abhinav Pillai

[1]SCIT, Manipal University Jaipur
Email: abhinav2319@gmail.com

***Abstract*— Sarcasm is a form of implicit communication that mocks, ridicules or expresses contempt by using words that contradict the underlying message. It is easiest to identify sarcasm through cues such as the tone of voice of the speaker, their facial expressions or hand movements, and yet quite difficult. However, when dealing with written texts, even these cues are lost. Due to this inherently ambiguous nature of sarcasm in text, Sarcasm Detection models have gained importance as a way of decoding texts to comprehend sarcasm with high accuracy. We have implemented Machine Learning algorithms such as Gaussian Naïve Bayes, Logistic Regression, Decision Tree, K Nearest-Neighbors and Support Vector Machines with a bevy of feature extractors on a 10,000 tuple dataset, and Neural Networks such as CNN and Bi-directional LSTM on the same and on larger data-sets, to analyze the prediction models for Sarcasm Detection. The results indicated that Bi-LSTM and CNN had the best accuracies at 90.7% and 85.9%.**

***Keywords*:** Feature Extraction, Machine Learning, Neural Networks, Sarcasm Detection.

I. INTRODUCTION

Newspapers are a rich informational source which provides us with a clear idea and understanding of what is happening in and around the world, on a daily basis. It is one of the greatest means of communication between people and the world, due to its high availability and low cost. Even though they have very detailed articles, more often than not, it is the Headline of an article which sparks an interest in the reader. So, news providing agencies tend to create catchy headlines to attract the reader's attention onto them, and this is how Sarcasm manages to find its way into News Headlines.

Sarcasm employs sentences that mean the opposite of what needs to be conveyed, i.e. it is a form of verbal irony. This may be done to criticize someone or something, to show annoyance or just to be funny. For instance, using "They're really on top of things" to describe a group of people who are disorganized, is sarcasm. Sarcasm when verbally expressed is easier to detect due to the added audiovisual cues which the speaker may showcase, such as a mocking tone, exaggerated expression using hands, or even a roll of the eyes. But the same task becomes significantly more difficult when presented in a textual format without these cues- when all the reader has as information to detect sarcasm, lies in the text itself, often mentioned without any context. That is where genuine news tends to become fake news if the reader wrongfully guesses the content while skimming through the headlines of the newspaper- as most studies show that only a sliver of the population actually reads the full contents of a newspaper, and not just the headlines.

This leads to the need of developing methods by which we can correctly predict whether a piece of text, or news for that matter, truthfully means what it says or is simply being sarcastic about it. And that is where Sarcasm Detection using Natural Language Processing comes in, as an attempt to develop a data processing model which can correctly guess whether a piece of text is sarcastic or not. There is no 100% accurate model as of now and thus, researchers have been trying to create models with the highest accuracies using a variety of methods in various parts of the process. Our paper is one among them.

We agree with [12] on the fact that sarcasm in text tends to have no fixed structure as such, and so, implement a new approach to data pre-processing, in an attempt to improve upon whatever structure is already present. With respect to this, we do not perform stop-word removal during pre-processing and add on further to the structure by converting numbers to words instead of outright removing them. This gives us an increase in accuracy up till 11.24%. Our goal is to analyze the effectiveness of sarcasm detection with various models in order to find the ones that work best for a given dataset which has been pre-processed as mentioned.

A dataset containing 55,329 tuples, consisting of news headlines from The Onion and the Huffington Post, taken from Kaggle [1], was used to employ the proposed approaches in this paper.

II. RELATED WORK

There were 3 papers which made use of the same dataset as ours in their respective approaches. [2] used Word level, n-gram level and Character level TF-IDF embedding combined

with Count Vectorizer for their research, concluding that supervised learning techniques fare better than deep learning methods for sarcasm detection. [3] used Part-of-Speech tagging to determine connections between words and utilized a rule-based approach to extract optimal features for their research. Their approach gave better results for SVM classifier compared to standard feature extraction. [12] used a Deep and Dense Neural Network to extract additional intrinsic information from the text for making better predictions for standalone utterances.

Sarcasm detection models implemented by different researchers primarily differ in the feature extraction and embedding process. As such, quite a few novel approaches have been invented which generate better results than traditional approaches. [4] developed novel deep learning models which extended the architecture used by Bidirectional Encoder Representations from Transformers (BERT) by incorporating affective and contextual features into it, which were extracted using a Bidirectional Long Short-term Memory (Bi-LSTM) model with multi-head attention. [5] used suitable pre-trained BERT and Global Vector (GloVe) word embedding techniques on their data. [6] extracted interjection words and intensifiers using Part-of-Speech tagging to build the feature set for 5-fold cross validation. [7] utilized positive-negative incongruities, generally observed in sarcastic sentences, as a part of their feature extraction.

[8] proposed an interesting approach based on an observation. Sentences having sarcasm in them have words which belong to one of two clusters- sarcastic targets and non-sarcastic targets, depending on the values according to their Local and Organization named-entity distribution and Part-of-Speech distribution. They utilized these features using LIWC and Empath [13] category fractional distribution dictionaries for their research. [9] utilized the incongruity of sarcastic sentences using exaggerated or comparative numerical values using rule-based, statistical machine learning and deep learning approaches for their classification. [10] categorized sarcastic sentences based on the function of the sarcasm- a complex form of expression, a means of conveying emotion, a possible function of familiarity or as a form of written expression, and evaluated them using the SCUBA framework. They procured higher performance than standard contrast-seeking frameworks. [11] applied an ensemble strategy based on models trained on different length conversational texts to make more accurate predictions. Last but not the least; [13] used a rule-based approach on the basis of the sentiment score of various features in data for their research.

III. METHODOLOGY

We acquired the labeled dataset on news headlines from Kaggle [1] and utilized Jupyter Notebook as the stage to code. Our approach includes utilization of various classification systems such as Support Vector Machine (SVM), K-Nearest Neighbors (K-NN), Logistic Regression, Decision Tree, Gaussian Naïve Bayes, Convolutional Neural Networks (CNN) and Bi-directional Long Short-term Memory model.

A. Dataset

A balanced subset of the 55,329 tuple data-set was used for the ML based models- consisting of 10,000 tuples which were pre-labeled as 1 for sarcastic and 0 for non-sarcastic. Similar balanced subsets having 15,000, 20,000 and 25,000 tuples were derived for the Neural Network based models. The full unbalanced data-set containing 25,358 sarcastic and 29,971 non-sarcastic entries was also used in the research. The datasets had the following attributes:

Attribute Information:
1. Article_link- A unique ID
2. Headline- Text to be classified
3. Is_sarcastic- 1 for Sarcastic, 0 for Not

The datasets were split into training and testing sets in a ratio of 75:25 for each classification model.

B. Data Pre-processing

Information pre-processing is an information mining system involving the conversion of raw information into a workable form. The text in the datasets was first converted to lowercase form in order to achieve uniformity of words. The text contained lots of contraction terms and thus, they were expanded using necessary modules. Numerical data is usually cleaned out during pre-processing. However, in order to retain more of a structure for more accurate sarcasm detection, the numerical data was converted to words using the num2words module, before cleaning out the non-alphabetical characters from the text. The removal of stop-words and the stemming or lemmatization of text significantly reduces the structure of the text and thus, they are not applied. These decisions improve the final accuracies of the implemented models by up till 11.24%, compared to that for conventionally pre-processed data. This increase in accuracy for each type of classification done through each type of feature extraction, compared to the accuracy for traditionally pre-processed data is shown in Table 2.

C. Feature Extraction

Count Vectorizer, TF-IDF Vectorizer and Hashing Vectorizer were used to extract features for the ML based while a pre-trained GloVe (100 dimensional, trained on 6 billion words) was used for the Neural Network based models. The working of each is as follows:

Count Vectorizer the Count Vectorizer tokenizes and generates a vocabulary of unique words from the

data. It produces an output for each tuple containing the number of times, the words of the vocabulary appeared in it, in a 1-dimensional matrix form. It is implemented using the Scikit-Learn module.

Term Frequency-Inverse Document Frequency Vectorizer the TF-IDF Vectorizer tokenizes the words of the data based on their frequency and relevancy in the dataset. Term Frequency is the number of times a word appears in a document. In order to equalize this value across documents of differing lengths, we calculate it as follows:

$$TF(t) = \frac{N}{T}$$

Where,
N = Total frequency of t in a document.
T = Total number of terms in the document.

The IDF is a measure of the relevancy or importance of a term. This is done to reduce the influence of words which repeat many times but have no real meaning or importance in the text. This value is normalized for effective usage:

$$IDF(t) = \log_e \frac{TD}{ND}$$

Where,
TD = Total number of documents
ND = Number of documents with t in them

It produces output values containing the product of the TF and IDF values for each word in the data in the form of a 1-dimensional matrix. It is implemented using the Scikit-Learn module.

Hashing Vectorizer the Hashing Vectorizer converts a collection of data into a sparse matrix based on token occurrence counts, by mapping the data to the fixed dimensions of the matrix. It works faster for larger datasets compared to Count Vectorizer at the cost of losing the actual values of the tokens. It is implemented using Scikit-Learn.

Global Vectors (GloVe) GloVe is a word vector embedding technique that leverages both global and local statistics of a corpus to come up with a Principal Loss function. Put simply, it generates a Co-occurrence matrix from which semantic relationships between words can be derived. The cost function for GloVe is as follows:

$$J = f(X_{ij})(w_i^T u_j + bw_i + bu_j - \log(X_{ij}))^2$$

where $f(X_{ij}) = (x/x_{max})^\alpha$ if $x < x_{max}$ else 0

bw, bu = biases of the network

Our paper uses a pre-trained GloVe model. This greatly reduces the computational load on the device, allowing for quick training and testing even for very large datasets.

D. Classification Algorithms

Gaussian Naïve Bayes the Gaussian Naïve Bayes is a probabilistic classification algorithm based on the Bayes Theorem in statistics. It assumes that the continuous values associated with each extracted feature are distributed according to a Gaussian distribution.

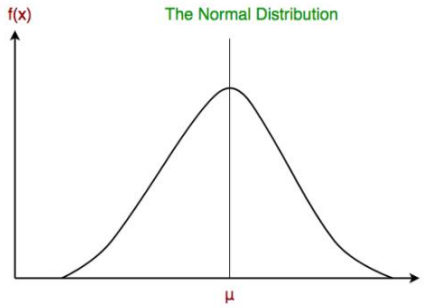

Fig.1. A plot showing a Gaussian distribution of values [15]

Since the likelihood of features is assumed to be Gaussian, the conditional probability for each x such that y occurs is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

This classifier is implemented using Scikit-Learn and works faster compared to its more complicated counterparts.

Decision Tree Algorithm the Decision Tree algorithm predicts the class or value of the target variable by learning simple decision rules, inferred from prior data.

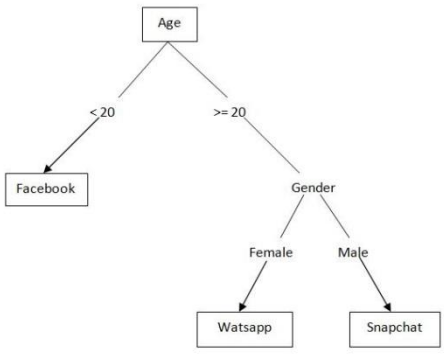

Fig.2. Structure of a Decision Tree- the right child has more homogeneous sub-nodes [16]

It generates a decision tree by splitting the nodes on all available variables and then selecting the node which results in the most homogeneous sub-nodes, using a greedy approach, i.e. a function that returns a locally optimized solution. It is implemented using Scikit-Learn.

Logistic Regression the Logistic Regression classifier employs a generalized linear model and a loss function that minimizes the results to values between 0 and 1. The input values are combined linearly using weights to predict the output values. The cost function is given by:

$$\text{Cost}(\text{h}_\Theta(x), y) = \begin{cases} -log(h_\Theta(x)) & if \quad y = 1 \\ -log(1 - h_\Theta(x)) & if \quad y = 0 \end{cases}$$

It is implemented using Scikit-Learn and gives the second highest accuracy among the ML models when the features are extracted using Count Vectorizer.

Support Vector Machines the Support Vector Machine Classifier computes a hyperplane in an n-dimensional space that best separates the two classes of data.

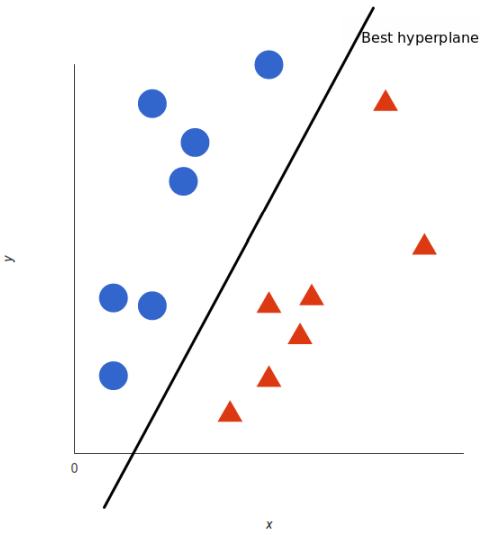

Fig.3. The SVM model computes the best hyperplane that can separate the binary groups of values [17]

It is implemented in a linear fashion using Scikit-Learn.

K-Nearest Neighbors the K-Nearest Neighbors is a classification algorithm which does not make any underlying assumptions about the data i.e. it is of a non-parametric form. It plots the classes on a graph and then tries to classify the test data based on the classes of its nearest ‘k’ neighbors. The value of ‘k’ is chosen such that it is the smallest odd number close to half the square root of the total number of tuples. For our 7,500 tuple training set, k is taken as 43.

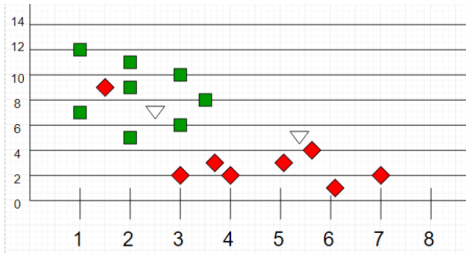

Fig.4. A plot showing 2 clusters for classification [18]

In the above plot, the 2 white triangles represent the unclassified data-points while the red and green ones represent the binary classes for the classification. The algorithm classifies the first white one as Green and the second one as Red based on the nearness of the ‘k' neighbors to it. The algorithm is implemented using the Scikit-Learn module.

Convolutional Neural Network CNN is a neural Network that has one or more convolutional layers i.e. filters over the input data that help make better predictions. These filters when used appropriately, can convert complex, multi-dimensional input data into a simple 1-dimensional form. It is implemented using the Keras module.

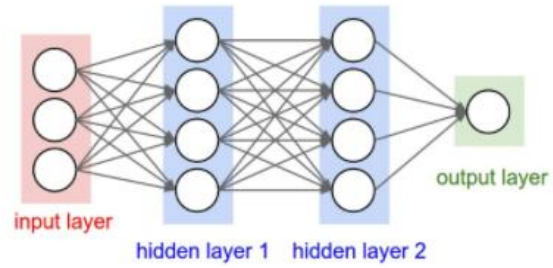

Fig.5. Structure of a Basic Convolutional Neural Network [19]

Bidirectional Long Short-Term Memory Bi-LSTM is a Sequence processing model that employs two LSTMs-working in forward and backward directions i.e. it is a type of Recurrent Neural Network (RNN). The backwards pass allows the neural network to effectively increase the amount of data available to it and also optimize its cost function at each iteration. Its advantage over CNNs is its ability to process data sequences such as sentences more effectively. It reuses activation functions in the data sequence to predict the outputs unlike CNNs which make use of filters.

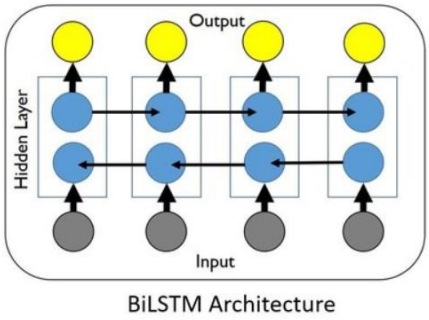

Fig.6. Architecture of a Bi-LSTM Neural Network [20]

IV. EXPERIMENTS AND RESULTS

A. *Evaluation Metrics*

The evaluation metrics were derived from the confusion matrices generated by the models used.

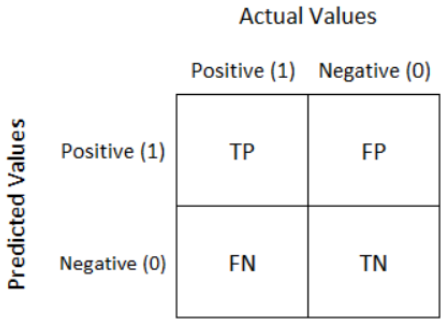

Fig.7. Structure of a Confusion Matrix [21]

They are as follows:

Accuracy: It represents the number of correctly classified data instances over the total number of data instances. It is computed by:

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

Precision: It represents the number of positive class predictions that actually belong to the positive class. It helps evaluate models when the costs associated with false positives is high. It is computed by:

$$Precision = \frac{TP}{TP + FP}$$

Recall: It represents the number of negative class predictions that actually belong to the negative class. It helps evaluate models when the costs associated with false negatives is high. It is computed by:

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: It is the weighted average of the Precision and Recall of a model. It is computed by:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

B. *Experiments*

Each of the Machine Learning based models were fed features extracted through Count Vectorizer, TF-IDF Vectorizer and Hashing Vectorizer, and their corresponding confusion Matrices were generated for evaluation.

The Neural network based models used pre-trained GloVe Embeddings for feature extraction, and were trained on 5 different pre-processed balanced datasets of sizes 10k (the same used for ML models), 15k, 20k, 25k and 55k. This process could not be followed for the other models due to hardware limitations.

C. *Results*

The comparison between the Neural Network based models for the varying dataset sizes has been shown in Table 1. The highest accuracies of both Bi-LSTM and CNN were obtained for the dataset with 55k tuples. They were 90.7% and 85.9% respectively. This is an improvement on the work of [3] as the methods used produce higher accuracy overall. Though the SVM accuracy in [3] seems to be higher than that for this paper by 0.8%, the dataset sizes differ by 17,000 entries.

The Bi-LSTM and CNN accuracies were 83.28% and 76.4% for the dataset common to both ML and Neural Network models. The accuracy for these 2 models increased by an average of 2% for the applied pre-processing.

	10k	15k	20k	25k	55k
CNN	76.4	81.07	81.04	81.6	85.9
Bi-LSTM	83.28	83.65	84.46	85.71	90.7

Table 1. Table showing results of CNN and Bi-LSTM when working with datasets of increasing sizes

CLASSIFIER	FEATURE EXTRACTOR	ACCURACY	PRECISION	RECALL	F1 SCORE	INCREASE IN ACCURACY
GAUSSIAN NAÏVE BAYES	COUNT VEC	66.52	50.23	74.92	60.16	3.40%
	TF-IDF VEC	63.83	54.46	68.27	60.59	1.11%
	HASHING VEC	57.8	60.73	58.31	59.5	4.68%
DECISION TREE	COUNT VEC	73.48	77.74	71.85	74.68	3.56%
	TF-IDF VEC	73.48	77.74	71.85	74.68	3.56%
	HASHING VEC	53.32	55.24	53.5	54.36	0.40%
LOGISTIC REGRESSION	COUNT VEC	81.39	82.03	81.19	81.61	6.07%
	TF-IDF VEC	79.2	79.09	79.47	79.28	7.40%
	HASHING VEC	55	54.84	55.33	55.08	1.17%
K NEAREST NEIGHBORS	COUNT VEC	62.4	48.72	67.51	56.6	11.24%
	TF-IDF VEC	49.72	0.08	100	0.16	0.04%
	HASHING VEC	56.87	57.23	57.14	57.18	2.87%
SUPPORT VECTOR MACHINES	COUNT VEC	78	77.26	78.64	77.94	8.33%
	TF-IDF VEC	77.24	76.39	77.93	77.15	7.96%
	HASHING VEC	55.96	58.93	56.58	57.73	2.16%

Table 2: Table showing results of ML based models. The best results have been highlighted in green. The last column shows the rise in accuracy for each model on utilizing the proposed pre-processing process, compared to traditional pre-processing

The proportional increase in accuracy on increasing the size of the dataset and subsequently the training set, has been shown in Fig 8. The accuracy for CNN starts below the highest value for the ML models and stays similar throughout, showing a non-linear increase till the last dataset. On the other hand, the accuracy for Bi-LSTM starts higher than the ML models and increases almost linearly over the datasets.

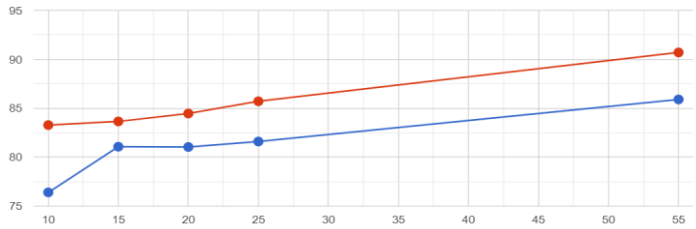

Fig.8. Scatter plot showing increase in accuracy with increase in size of dataset for CNN (blue) and Bi-LSTM (red)

The highest accuracies among the ML models which were pre-processed as proposed were 81.39% for the LR model with Count Vectorizer, 79.2% for the LR model with TF-IDF Vectorizer and 78% for the SVM model with Count Vectorizer. Their corresponding F1-Scores were 81.61%, 79.28% and 77.94% respectively. The complete results for the ML based models have been shown in Table 2.

From the Machine learning model results, we observe that the accuracy of models using Count Vectorizer was consistently higher than that of its peers. Also, the accuracies of all models using Hashing Vectorizer stayed in the 50% to 60% range for all classification models, causing the advantage produced by its faster computation speed to be overruled by the significant loss in accuracy. The models utilizing the TF-IDF Vectorizer always came close to those using Count Vectorizer, but they did not produce a higher accuracy for this dataset.

The average increase in accuracy for each model has been shown in Table 3. As observed, Support Vector Machines benefit the most among the ML models from the proposed pre-processing technique.

CLASSIFIER	AVG RISE IN ACCURACY
NAÏVE BAYES	3.06%
DECISION TREE	2.51%
LOGISTIC REGRESSION	4.88%
K-NEAREST NEIGHBORS	4.71%
SUPPORT VECTOR MACHINE	6.15%

Table 3: Table showing the average increase in model accuracies due to proposed pre-processing

V. CONCLUSION

In this paper, we have presented a better way to pre-process text data to be used for Sarcasm detection. Our method incorporates the usage of stop-words and numbers as words, in order to improve upon the structure of the otherwise almost unstructured sarcastic elements. It also overrules stemming and lemmatization for the same reason. This implementation causes both Machine Learning and Neural Network based model accuracies to increase by a good margin. Our results demonstrate that Sarcasm can be better classified by building upon the structure of the sentences through already available potential contractions instead of taking away from them.

REFERENCES

1. Rishabh Misra. News Headlines Dataset for Sarcasm Detection. Retrieved from: *https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection?select=Sarcasm_Headlines_Dataset.json*

2. Onyinye Chudi-Iwueze and Haithem Afli. Detecting Sarcasm in News Headlines. In: *CERC, 2020*

3. Vaishvi Prayag Jariwala. Optimal Feature Extraction based Machine Learning Approach for Sarcasm type Detection in News Headlines. In*: International Journal of Computer Applications, 2020*

4. Nastaran Babanejad, Heidar Davoudi, Aijun An and Manos Papagelis. Affective and Contextual Embedding for Sarcasm Detection. In: *International Conference on Computational Linguistics, 2020*

5. Akshay Khatri, Pranav P and Dr. Anand Kumar M. Sarcasm Detection in Tweets with BERT and GloVe Embeddings. In: *ACL, 2020*

6. Santosh Kumar Bharti, Reddy Naidu and Korra Sathya Babu. Hyperbolic Feature-Based Sarcasm Detection in Tweets: A Machine Learning Approach. In: *IEEE Xplore, 2020*

7. Hongliang Pan, Zheng Lin, Peng Fu and Weiping Wang. Modeling the Incongruity between Sentence Snippets for Sarcasm Detection. In: *ECAI, 2020*

8. Jasabanta Patro, Srijan Bansal and Animesh Mukherjee. A deep-learning framework to detect sarcasm targets. In: *ACL, 2019*

9. Abhijeet Dubey, Lakshya Kumar, Arpan Soumani, Aditya Joshi and Pushpak Bhattacharyya. "Where Numbers Matter!": Detecting Sarcasm in Numerical Portions of Text. In: *ACL, 2019*

10. Ashwin Rajadesingan, Reza Zafarani and Huan Liu. Sarcasm Detection on Twitter: A Behavioral Modeling Approach. In: *ACM, 2015*

11. Nikhil Jaiswal. Neural Sarcasm Detection using Conversation Text. In: *ACL, 2020*

12. Devin Pelser and Hugh Murrell. Deep and Dense Sarcasm Detection. In: *arXiv.org*

13. Rahul Gupta, Jitendra Kumar, Harsh Agarwal and Kunal. A Statistical Approach for Sarcasm Detection using Twitter Data. In: *ICICCS, 2020*

14. Ethan Fast, Binbin Chen and Michael S. Bernstein. Empath: Understanding Topic Signals in Large Scale Text. In: *CHI, 2016*

15. Nikhil Kumar (Creator). A plot of a Gaussian distribution. Retrieved from: *https://www.geeksforgeeks.org/naive-bayes-classifiers/*

16. Sagarnil Das (Creator). Structure of a Decision Tree. Retrieved from: *https://www.linkedin.com/pulse/how-does-id3-algorithm-works-decision-trees-sagarnil-das/*

17. Bruno Stecanella (Creator). (2017, July 22). The best Hyperplane for a 2D plot. Retrieved from: *https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/#:~:text=A%20support%20vector%20machine%20(SVM,able%20to%20categorize%20new%20text*.

18. Anannya Uberoi (Creator). K-NN plot clusters. Retrieved from: *https://www.geeksforgeeks.org/k-nearest-neighbours/*

19. Xin Li (Creator). (2018, May 18). A 3-layer Neural Network. Retrieved from: *http://www.xincoder.com/blogs/convolutional_network/*

20. Arvind Mohan (Creator). LSTM and Bi-LSTM Architectures. Retrieved from: *https://www.researchgate.net/figure/LSTM-and-BiLSTM-Architectures_fig2_324769532*

21. Sarang Narkhede (Creator). Structure of a Confusion Matrix. Retrieved from: *https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62*