EasyChair Preprint

№ 6020

Audio and text toxics comments classification

Sangita Holkar, Sudhir Sawarkar and Shubhangi Vaikole

July 5, 2021

AUDIO AND TEXT TOXIC COMMENT CLASSIFICATION

Sangita Holkar, Dr. S.D .Sawarkar, Dr.Shubhangi Vaikole
[1]Student, [2]Principal, [3] Associate Professor
[1]Computer Department,
[1]Datta meghe college of Engineering, Airoli, India

Abstract : Social networking and online conversation platforms provide us with the power to share our views and ideas. However, nowadays on social media platforms, many people are taking these platforms for granted, they see it as an opportunity to harass and target others leading to cyber- attack and cyber-bullying which lead to traumatic experiences and suicidal attempts in extreme cases. Manually identifying and classifying such comments is a very long, tiresome and unreliable process. To solve this challenge, we have developed a deep learning system which will identify such negative content on online discussion platforms and successfully classify them into proper labels. Our proposed model aims to apply the text-based Convolution Neural Network (CNN) with NLP, using LOGISTIC REGRESSION,MULTINOMIALDB, LINEAR SVC word embedding technique. Our model aims to improve detecting different types of toxicity to improve the social media experience. Our model classifies such comments in six classes which are Toxic, Severe Toxic, Obscene, Threat, Insult and Identity-hate. Multi-Label Classification helps us to provide an automated solution for dealing with the toxic comments problem we are facing**.**

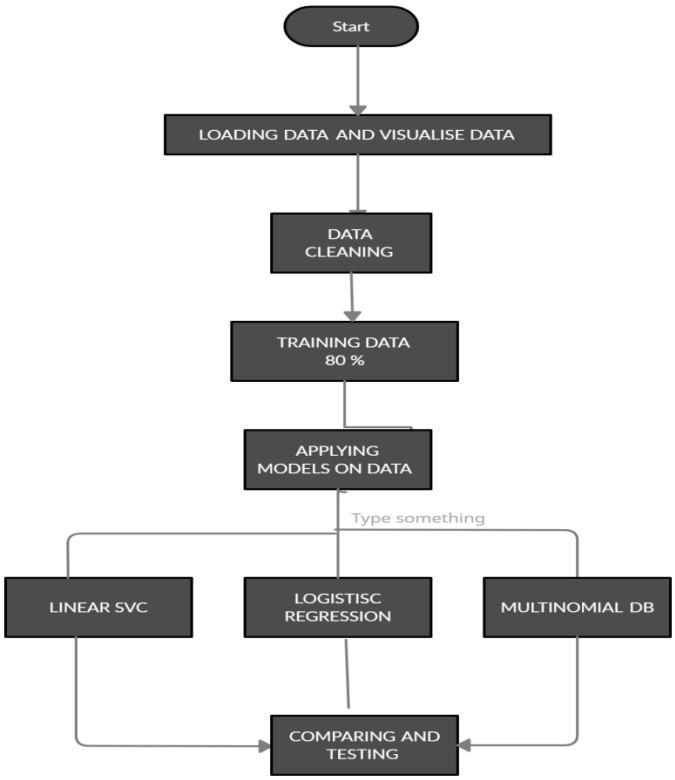
IndexTerms **-Convolution Neural Network (CNN), Python-tesseract, Logistic Regression,Multinomialdb, Linear Svc .**

I. INTRODUCTION

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to evidently facilitate conversations, leading many communities to limit or completely shut down user comments. Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to evidently facilitate conversations, leading many communities to limit or completely shut down user comments. The problem which I am trying to solve is a multiclass multilabel classification which means a comment can belong to a single class or more than one class or none.

The flow of the project is :

1. Loading the data and visualizing the data from dataset
1. Reducing the data instances which belong to none of the classes to make the dataset less skewed.
2. Standard Preprocessing for textual data, which are: lower casing the text, removing punctuations, removing stopwords, lemmatization of the text.
3. Converting textual data into numerical embedded data via tokenization API provided by Keras framework.
4. Defining the deep learning model for the classification task.



1.1 Loading And Visualizing The Dataset

The process of converting data to something a computer can understand is referred to as pre-processing. One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop word

What are Stop words?

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory. home/pratima/nltk_data/corpora/stopwords is the directory address.(Do not forget to change your home directory name)

Example

Imports

```
import pandas as pd
import numpy as np
from nltk.corpus import stopwords
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import seaborn as sns
```

1.1.1 Removing Stop Words With NLTK

The following program removes stop words from a piece of text:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent = """This is a sample sentence,
                showing off the stop words filtration."""
 stop_words = set(stopwords.words('english'))
 word_tokens = word_tokenize(example_sent)
 filtered_sentence = [w for w in word_tokens if not w in stop_words]
 filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
  print(word_tokens)
print(filtered_sentence)
```

Output:

```
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing',
'off', 'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop',
'words', 'filtration', '.']
```

Performing the Stopwords operations in a file

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

On loading the data we can find out the counts of the classified data individually With data to be visualize the data is shown in a tabular format with the command (df:describe) Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated into the data structures from pandas.

1.1.2 Box Plot

A box plot helps to maintain the distribution of quantitative data in such a way that it facilitates the comparisons between variables or across levels of a categorical variable. The main body of the box plot showing the quartiles and the median’s confidence intervals if enabled. The medians have horizontal lines at the median of each box and while whiskers have the vertical lines extending to the most extreme, non-outlier data points and caps are the horizontal lines at the ends of the whiskers.

```
df.describe()
```

	toxic	severe_toxic	obscene	threat	insult	identity_hate
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

After classification of the data the graph is plotted showing which of the classification is more aacurate

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
id               159571 non-null object
comment_text     159571 non-null object
toxic            159571 non-null int64
severe_toxic     159571 non-null int64
obscene          159571 non-null int64
threat           159571 non-null int64
insult           159571 non-null int64
identity_hate    159571 non-null int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

1.1.3 To Find The Data Visulalise The Data Individually

```
#only identity hate
df_identity_hate = df[(df['identity_hate']==1) & ( df['insult']==0)  & ( df['threat']==0)  & ( df['obscene']==0)  & ( df['severe_
df_identity_hate.head()
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
3620	09b51ed1ee5e29a1	Mate, sound like you are jewish\n\nGayness is ...	0	0	0	0	0	1
5839	0f9a7ba1acde6b6e	it is obvius to, me, a black man, that orange ...	0	0	0	0	0	1
13201	22f28f593011019e	The Israelis are committing massacres in Gaza,...	0	0	0	0	0	1
17279	2da09631a5e75737	"\n\n How does it feel \n\nTo be a negress? Do...	0	0	0	0	0	1
29220	4d7a10ae7a7c7621	"\n\n A hit list for Jews \nThere is a site ca...	0	0	0	0	0	1

1.1.4 Visualizing The Data From The Dataset Finding The Various Clasiification Into The Dataset

```
#only threat
df_threat = df[(df['identity_hate']==0) & ( df['insult']==0)  & ( df['threat']==1)  & ( df['obscene']==0)  & ( df['severe_toxic'
df_threat.head()
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
3712	09eb7d87b8c24ca5	Please stop. If you continue to ignore our pol...	0	0	0	1	0	0
16939	2cb0ead532923065	That's funny. You was personally offended? So ...	0	0	0	1	0	0
17210	2d6e671ffae726a3	Wow dude. As your physician I suggest you sli...	0	0	0	1	0	0
29295	4db479fe05e88395	Regarding your passing \n\nBecause you willful...	0	0	0	1	0	0
48209	80e0b91bdea43fd3	personal attacks \n\ndont tell me what i can a...	0	0	0	1	0	0

DATA PREPROCESSING

```
import re
import string
import emoji
#text = "game is on □ □"
#print(emoji.demojize(text, delimiters=("", "")))
hastag_pat = r'#[0-9a-z_]+'
mentions_pat = r'\@[0-9a-z]+'
number_pat = r'\d+'
links_pat = 'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
emoji_pat = r'[^\w\s,]'
emoticons_pat = r'(?::|;|=)(?:-)?(?:\)|\(|D|P)'

def find_hashtags(tweet):
    return re.findall(hastag_pat, tweet, flags=re.I)

def find_mentions(tweet):
    return re.findall(mentions_pat, tweet, flags=re.I)

def find_links(text):
    return re.findall(links_pat, text,flags=re.I)

def find_emoji(text):
    return re.findall(emoji_pat, text,flags=re.I)

def find_emoticons(text):
    return re.findall(emoticons_pat,text,flags=re.I)

def remove_hashtags(text):
    return re.sub(hastag_pat, '', text)

def remove_mentions(text):
    return re.sub(mentions_pat, '', text)

def remove_links(text):
    return re.sub(links_pat, '', text)

def remove_emoji(text):
    return re.sub(emoji_pat, '', text)

def remove_emoticons(text):
    return re.sub(emoticons_pat, '', text)
def remove_number(text):
    return re.sub(number_pat, '', text)

def remove_punctuations(text):
    return ' '.join(word.strip(string.punctuation) for word in text.split())

def clean_data(text):
    text = text.lower()
    clean_sent = remove_hashtags(text)
    clean_sent = remove_mentions(clean_sent)
    clean_sent = remove_links(clean_sent)
    #clean_sent = remove_emoji(clean_sent)
    clean_sent = remove_emoticons(clean_sent)
    #clean_sent = emoji.demojize(clean_sent, delimiters=("", ""))
    clean_sent = remove_number(clean_sent)
    clean_sent = remove_punctuations(clean_sent)
    #clean_sent = [word for word in clean_sent if not word in set(stopwords.words('english'))]
    clean_sent = ''.join(clean_sent)
    return clean_sent
```

22]:

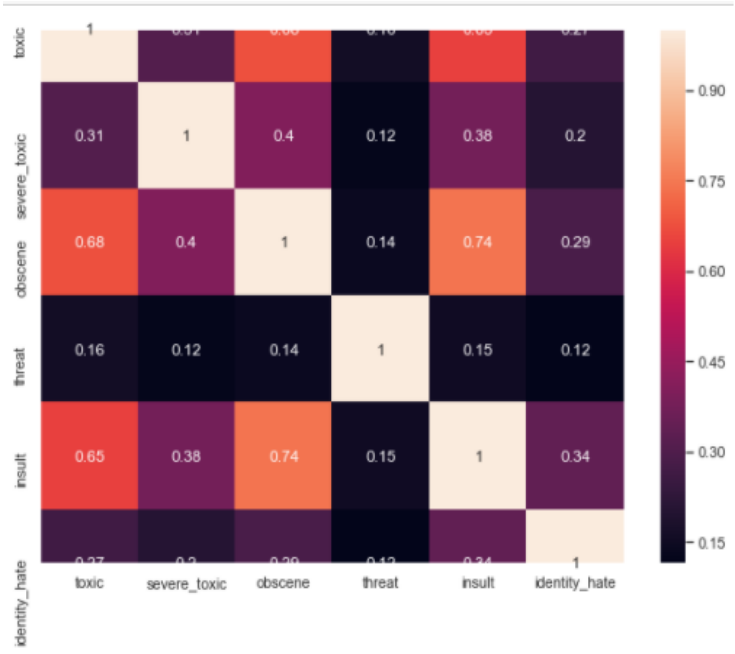
	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
6	0002bcb3da6cb337	cocksucker before you piss around on my work	1	1	1	0	1	0
12	0005c987bdfc9d4b	hey what is it talk what is it an exclusive...	1	0	0	0	0	0
16	0007e25b2121310b	bye don't look come or think of comming back t...	1	0	0	0	0	0
42	001810bf8c45bf5f	you are gay or antisemmitian archangel white t...	1	0	1	0	1	1
43	00190820581d90ce	fuck your filthy mother in the ass dry	1	0	1	0	1	0

cleaning up data with syop words and other techniques gives you the data cleaning byapplying filters such as passing function of removing hash tags , punctuation marks ,colon ,semicolon ,fullstop ,removing the reapeated characters after the words .The result will show you the filter and clean data
which comes after cleaning

	category	count
0	id	0000997932d777bf000103f0d9cfb60f000113f07ec002...
1	toxic	15294
2	severe_toxic	1595
3	obscene	8449
4	threat	478
5	insult	7877
6	identity_hate	1405

1.2 Correlation Matrix

Here in the below ouput the values showing the indentity shows that the data is of no use theere where as the other data is very useful Correlation Matrix also shows that the data whose value is 0.99 or 0.98 are again to the indentity hence these data are also not used for testing because that contains the non toxic data . The indentity matrix is a colourful matrix where white colour is used to define for the indentity matrix .The colour of the matrix is seen in mostly 3 to 4 colours as they are not fixed they can be taken as random the x-axis and y -axis which are differentiated on the basis od the multilbel clasiification .The Classification is shown is expressed as toxic,severe toxic,obscene ,threat ,insult ,identity hate .

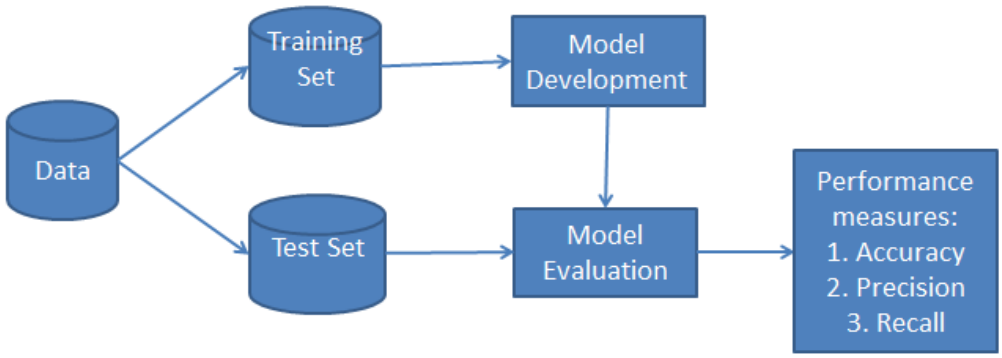


1.3 **Applying Model For The Database**

TYPES OF MODELS TO BE USED

1. **LINEAR REGRESSION**
2. **MULTINOMIAL DB MODEL**
3. **LINEAR SVC**

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
cvec = CountVectorizer()
tvec = TfidfTransformer()
model1 = MultinomialNB()
```



Naive Bayes Classifier Naive Bayes is a kind of classifier which uses the Bayes Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class. This is also known as Maximum A Posteriori (MAP).
A.The MAP for a hypothesis is:
MAP(H)
= max(P(H|E))
= max((P(E|H)*P(H))/P(E))
= max(P(E|H)*P(H))
P(E) is evidence probability, and it is used to normalize the result. It remains same so, removing it won't affect.
Naive Bayes classifier assumes that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature. We can use example for explaining the logic i.e.,
A fruit may be considered to be an apple if it is red, round, and about 4″ in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.
In real datasets, we test a hypothesis given multiple evidence(feature). So, calculations become complicated. To simplify the work, the feature independence approach is used to 'uncouple' multiple evidence and treat each as an independent one.
P(H|Multiple Evidences) = P(E1| H)* P(E2|H) ……*P(En|H) * P(H) / P(Multiple Evidences)

1.3.1 Contervectoriztion

In order to use textual data for predictive modeling, the text must be parsed to remove certain words – this process is called tokenization. These words need to then be encoded as integers, or floating-point values, for use as inputs in machine learning algorithms. This process is called feature extraction (or vectorization).Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

```
Data = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', ' the', 'lazy', 'dog']
```

	The	quick	brown	fox	jumps	over	lazy	dog
Data	2	1	1	1	1	1	1	1

TF-IDF is an information retrieval and information extraction subtask which aims to express the importance of a word to a document which is part of a colection of documents which we usually name a corpus. It is usually used by some search engines to help them obtain better results which are more relevant to a specific query. In this article we are going to discuss what exactly is TF-IDF, explain the math behind it and then we will see how we can implement it in Python by using the Scikit-Learn library.

1.3.2 Tf-Idf

TF-IDF stands for Term Frequency — Inverse Document Frequency and is a statistic that aims to better define how important a word is for a document, while also taking into account the relation to other documents from the same corpus.
This is performed by looking at how many times a word appears into a document while also paying attention to how many times the same word appears in other documents in the corpus.
The rationale behind this is the following:

- a word that frequently appears in a document has more relevancy for that document, meaning that there is higher probability that the document is about or in relation to that specific word
- a word that frequently appears in more documents may prevent us from finding the right document in a collection; the word is relevant either for all documents or for none. Either way, it will not help us filter out a single document or a small subset of documents from the whole set.

So then TF-IDF is a score which is applied to every word in every document in our dataset. And for every word, the TF-IDF value increases with every appearance of the word in a document, but is gradually decreased with every appearance in other documents. And the maths for that is in the next section.

Tf-Idf Formula Explained

Now let's take a look at the simple formula behind the TF-IDF statistical measure. First let's define some notations:

- N is the number of documents we have in our dataset
- d is a given document from our dataset
- D is the collection of all documents
- w is a given word in a document

First step is to calculate the term frequency, our first measure if the score.
Term Frequency Formula

$$tfidf(w, d, D) = tf(w, d) * idf(w, D)$$

Here f(w,d) is the frequency of word w in document d.
Inverse Document Frequency Formula
With N documents in the dataset and f(w, D) the frequency of word w in the whole dataset, this number will be lower with more appearances of the word in the whole dataset.
while using TfidfTransformer will require you to use the CountVectorizer class from Scikit-Learn to perform Term Frequency.

1.3.3 Linear Svc

```
from sklearn.svm import LinearSVC
model2 = LinearSVC()
for category in labels:
    model2.fit(X_train, train[category])
    accuracy = model2.score(X_test, test[category])
    accuracies[1].append(accuracy)
```

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our uses, though you can use this for many situations. Let's get started.

Output

```
Accuracy For toxic Class Is 96.19%
Accuracy For severe_toxic Class Is 99.06%
Accuracy For obscene Class Is 97.9%
Accuracy For threat Class Is 99.74%
Accuracy For insult Class Is 97.18%
Accuracy For identity_hate Class Is 99.22%
```

```
from sklearn.linear_model import LogisticRegression
model3 = LogisticRegression(n_jobs=1, solver='liblinear')
for category in labels:
    model3.fit(X_train, train[category])
    accuracy = model3.score(X_test, test[category])
    accuracies[2].append(accuracy)
    print("Accuracy For {0} Class Is {1}%".format(category,round(accuracy*100,2)))
```

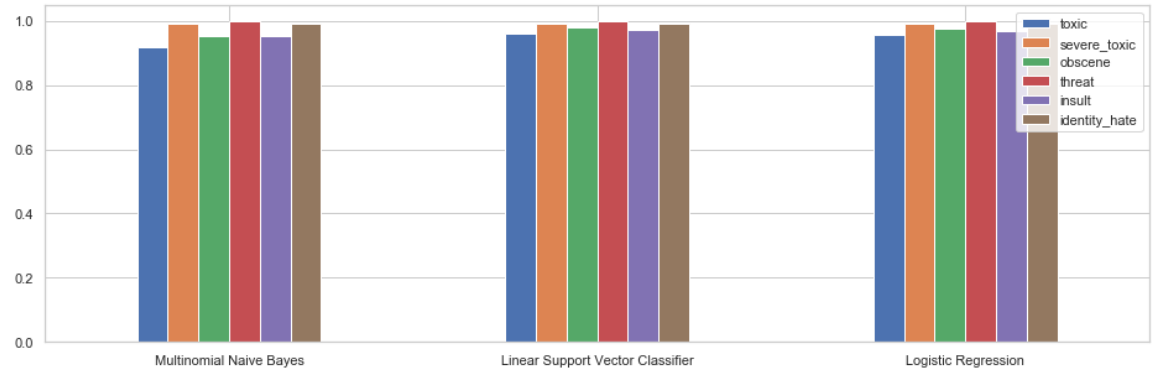
This dataset contains approximately 10,000 tweets that have been labeled using the CrowdFlower platform as conveying classification. That makes this dataset a unique perspective on the popular topic of sentiment analysis. While sentiment analysis typically focuses on expressions of positive or negative opinion, this data is alternatively more grounded in emotional states.

1.3.3.1 Features Of Linear Svc

: This column contains the text to which the label applies. It will get transformed into features used by the model during training and prediction.This data is subjected to three standard transformation/cleaning steps:
Converting the "label" column to be a categorical variable
Removing any rows for which the label column is missing a value
Stripping out non-alphanumeric characters and converting text to lower case
Feature extraction is done using AML's native Feature Hashing module, here set to fairly conservative parameters of unigram features and 12-bit hashing.
A logistic regression classifier is used. While the example data included with this experiment only contains two labels to predict, the model is created as one-vs-all multiclass.
In addition to training up a model, cross-validation is included (defaults to 5-fold). Summary statistics for cross-validation can be viewed directly via the output port of the Evaluate Model node, and predictions from the cross-validation run (averaged across folds) are also exported to CSV for inspection of model predictions.

1.4 Conclusion

```
accuracies = pd.DataFrame(accuracies)
fig = accuracies.plot.bar(figsize=(16, 5), grid=True)
plt.xticks(np.arange(3),('Multinomial Naive Bayes','Linear Support Vector Classifier','Logistic Regression'),rotation=0)
plt.legend(labels)
plt.show()
```



1.5 Future Analysis

The future prospects of this work lie in achieving a striking difference between these three algorithms by increasing the size of the dataset to achieve high degrees of accuracies with three of these models. The increase in size of the dataset will provide an increased number of features and hence, the feature extraction and modelling process will achieve correctness and accuracy in terms of predicting the sentiment of news article on the reader.

1.6 Acknowledgments

.This System Is Implemented Under Guidance Of Prof S.D .Sawarkar And Prof Shubhangi Vaikole ,Department Of Computer Engineering ,Airoli, India

1.7 References

[1] Thedora Chu, Max Wang, Kylie Jue. “Comment Abuse Classification with Deep Leraning.” Stanford University.
[2] Karthik Diankar, Roi Riechart, Henry Lieberman. “Modeling the Detection of Textual Cyberbullying.” Massachusetts Institute of Technology, Cambridge MA 02139 USA.
[3] Xin Wang, Yuanchao Li, Chengjie Sun, Baoxum Wang and Xialong Wang. “Polarities of Tweets by Composing Word Embeddings with Long Short Term Memory.” 7th International Joint Conference of Natural Language Processing. July-2005.
[4] S. V. Georgakopoulus, A. G. Vrahatis, S. K. Tasoulis, V. P. Plagianakos. “Convolutional Neural Networks for Toxic Comment Classification.” arXiv:1802.099574v1 [cs.CL], 27 Feb 2018.
[5] Manav Kohli, Emily Kuehler and John Palowitch. “Paying attention to toxic comments.” Stanford University.
[6] E. Wulcyzn, N.Thain and L.Dixon. “Ex-Machina-Personal attacks seen at scale.” 2017 International World Wide Web Conference Committee (IW3C2), ACM 978-1-44501-4913- - 10/17/04