PROJECT REPORT (Slot – E1+TE1)

ITE2011 – MACHINE LEARNING

TOPIC--MULTILINGUAL TOXIC COMMENT CLASSIFICATION

GROUP MEMBERS

SHIV NARAYAN SINGH - 17BIT0341

DEVESH GOYAL - 17BIT0103

Under the guidance of

Prof. DURAI RAJ VINCENT P.M [SITE,VIT VELLORE]

ABSTRACT

The main objective of the project is to determine all the toxic comments on social media including hate, abusive, obscene, threat and insulting comments in a dataset where the entries are stored in more than one language. In the dataset with more than one language there are many problems faced in writing the pseudo-code of the ML algorithms as the first part for each and every algorithm was to detect the language and then search/detect the word as to a toxic or a non-toxic comment. Multilingual Toxic comment classifications is becoming an active field of research with various recently proposed approaches. Although, while these approaches address some of the task's challenges the remaining ones still remain unsolved and directions for further research work is needed. Towards this end, we compare some of the deep learning and shallow approaches on a new, large comment data-set and introduce an ensemble that outperforms all individual models. We have contributed more than two data science algorithms to combine their respective results and to perform in a better satisfactory direction. Further, we validate our findings on a second/another dataset. The results of the combined algorithm of more than two algorithms (Data Science Algorithms) enable us to perform an extensive error analysis, which reveals open challenges for state-of-the-art methods and directions towards pending future research. These challenges contain missing paradigmatic context and non-consistent dataset labels.

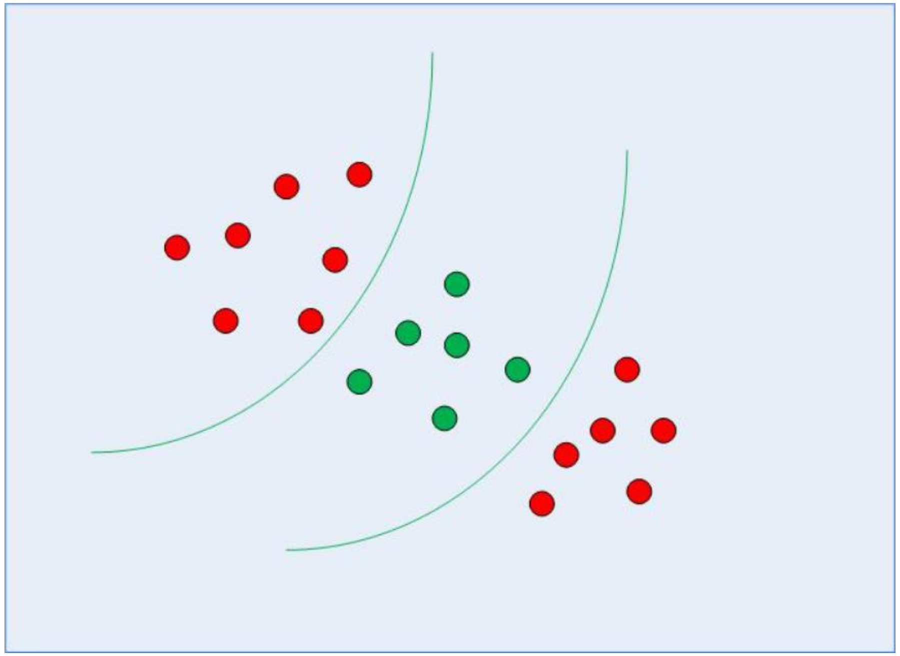
Objective:

The main objective of the project is to determine all the toxic comments on social media including hate, abusive, obscene, threat and insulting comments in a dataset where the entries are stored in more than one language.

INTRODUCTION:

1) KERNEL SVM

The simple SVM algorithm can be used to find the decision limit of equally divided data. However, in the case of separately separated data, such as those shown in Fig, the straight line cannot be used as a boundary of the decision.



In the case of inseparable separate data, the simple SVM algorithm cannot be used. Instead, a modified version of SVM, called Kernel SVM, is used.

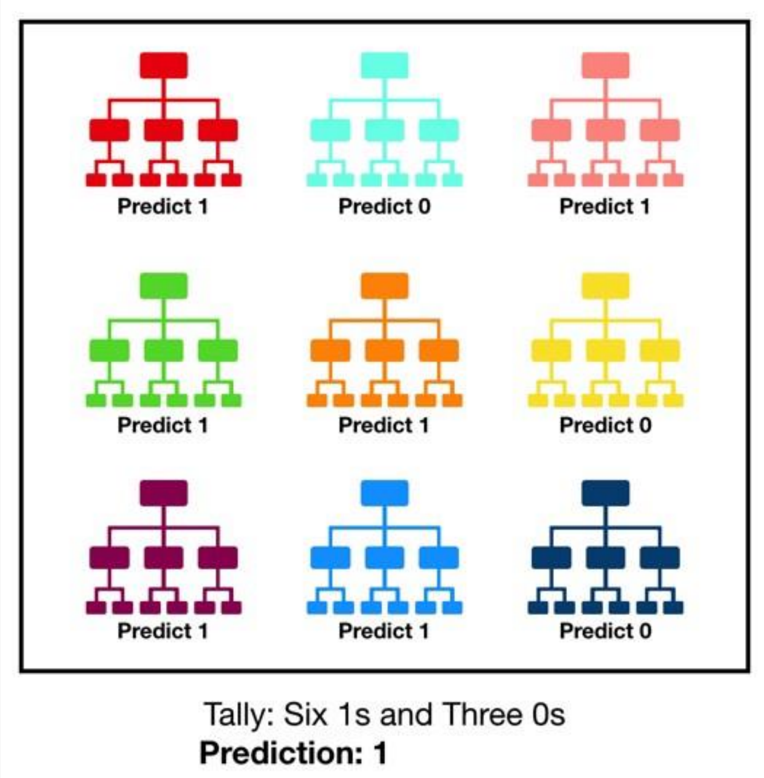
Basically, the kernel SVM proceeds to separate unequivocally sized data sets in such a way that data points for different classes are assigned to different sizes. Also, there are complex statistics involved in this, but you don't have to worry about it to use SVM. Instead we can just use Python's Scikit-Learn library to run and use the kernel SVM.

2) RANDOM FOREST

The Random forest, as its name implies, is home to an abundance of tropical trees that serve as a team. Each tree in a random forest spells out a category forecast and a category with the most votes becomes a prediction for our model (see image below).

The basic concept of a random forest is simple but powerful - the wisdom of the crowds. In the data talk point, the reason why the random forest model works so well is this:

A large number of unrelated species (trees) that function as a committee will surpass any of the different species.



View of Random Forest Model Performing Forecasting

Low integration between models is key. Just as low-correlated investments (such as stocks and bonds) come together to form a portfolio larger than the sum of its shares, uncomplicated models can produce more accurate forecasts than any other prediction. The reason for this positive effect is that trees protect each other from their individual mistakes (as long as they do not always deviate in the same way). While some trees may not be good, many other trees will be good, so as a group the trees are able to move in the right direction. The requirements for a random forest to do well are therefore:

1.There must be some actual signal in our options so models designed exploitation those options do higher than random guesswork.
2. The predictions (and thus the errors) created by the individual trees have to be compelled to have low correlations with one another.

3) LOGISTIC REGRESSION

Logistic regression is a supervised classification algorithm used to predict the probability of a variety of objectives. The type of target or variable dependency is clear, meaning that there will be only two possible categorise.

In simple terms, a dependent variable is a binary in environment with coded data such as 1 (stands for success / yes) or 0 (stands for failure / no).

Statistically, the systematic model predicts P (Y = 1) as an X function. It is one of the simplest ML algorithms that can be used for a wide variety of diagnostic problems such as spam detection, diabetes prognosis, cancer detection etc.

Types of Logistic Regression

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types −

- **Binary or Binomial**

  In such a type of division, the dependent variable will have only two types present either 1 and 0. For example, this variable can represent success or failure, yes or no, win or lose etc.

- **Multinomial**

  In such a class of variables, the dependent variables may have 3 or more non-adjustable variables or non-adjustable variables. For example, this variation may indicate "Type A" or "Type B" or "Type C".

- **Ordinal**

  For such type of classification, dependent variables may have 3 or more types that may be ordered or types that have a measurement value. For example, these variables can represent "poor" or "good", "very good", "very good" and each category can have scores such as 0,1,2,3.

4) DECISION TREE

Decision Tree could be a Supervised learning technique that may be used for both classification and Regression problems, but mostly it's preferred for solving Classification problems. it's a tree-structured classifier, where internal nodes represent thefeatures of a dataset, branches represent the choice rules and every leaf node represents the result.

In a Decision tree, there are two nodes, which are the choice Node and Leaf Node. Decision nodes are wont to make any decision and have multiple branches, whereas Leaf nodes are the output of these decisions and don't contain any longer branches.

The decisions or the test are performed on the idea of features of the given dataset.

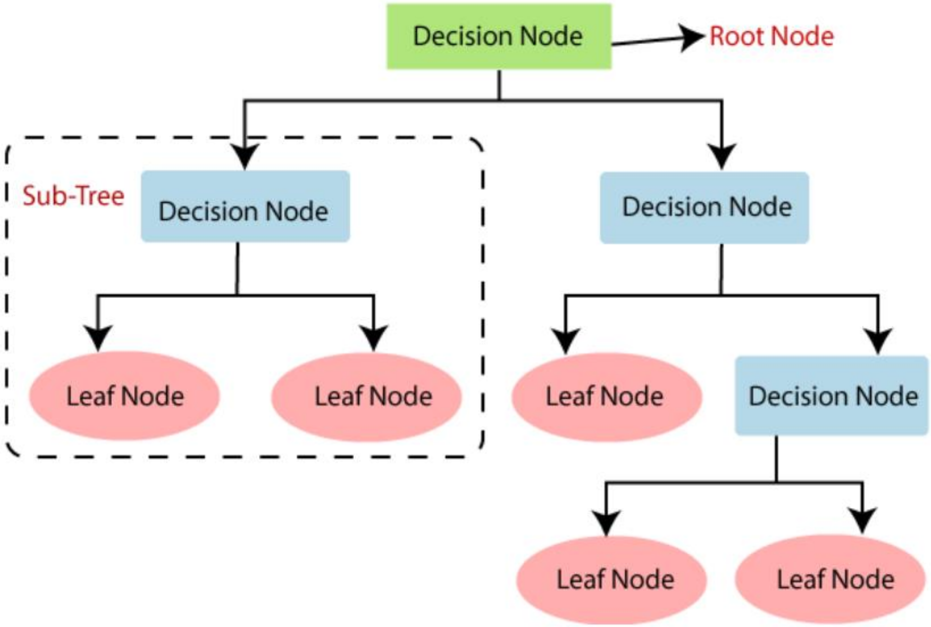
It is a graphical representation for getting all the possible solutions to a problem/decision supported given conditions.

It is called a call tree because, almost like a tree, it starts with the foundation node, which expands on further branches and constructs a tree-like structure.

In order to make a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a matter, and supported the solution (Yes/No), it further split the tree into subtrees.

Below diagram explains the general structure of a decision tree:



5) KNN

K-Nearest Neighbour is one among the only Machine Learning algorithms supported Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that's most kind of like the available categories.

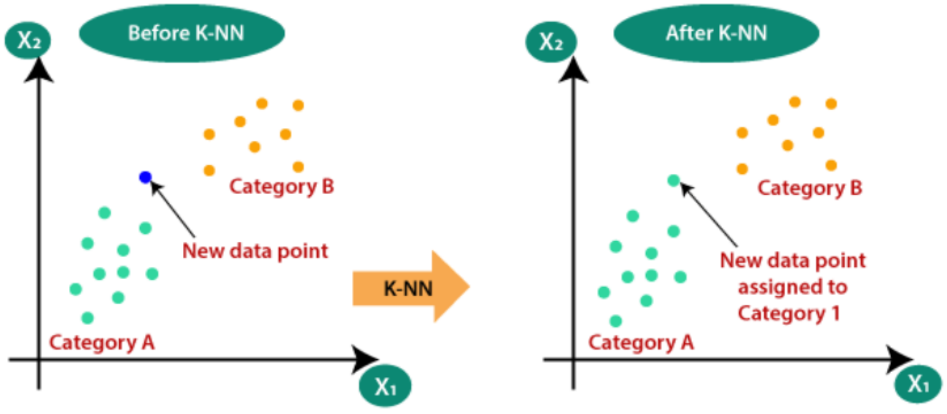
K-NN algorithm stores all the available data and classifies a brand new information supported the similarity. this implies when new data appears then it is easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm are often used for Regression further as for Classification but mostly it's used for the Classification problems. K-NN could be a non-parametric algorithm, which suggests it doesn't make any assumption on underlying data.

It is also called a lazy learner algorithm because it doesn't learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that's much the same as the new data.

Suppose there are two categories, i.e., Category A and Category B, and that we have a replacement datum x1, so this information will exist which of those categories. to unravel this sort of problem, we want a K-NN algorithm. With the assistance of K-NN, we will easily identify the category or class of a selected dataset.



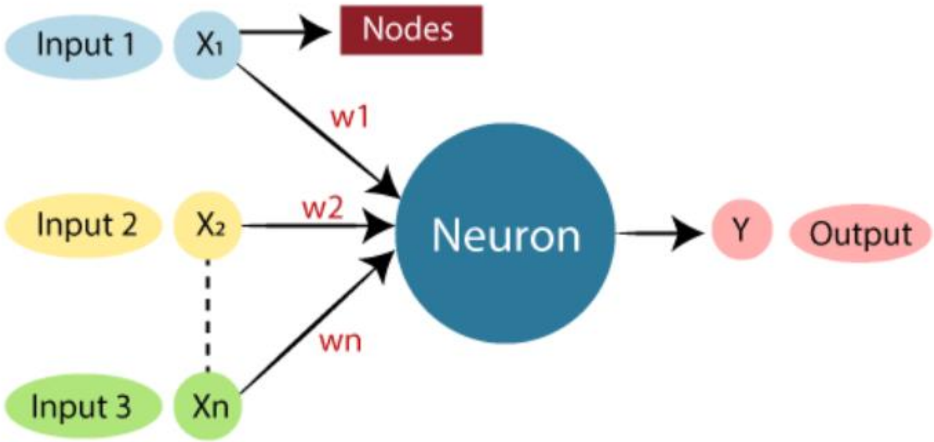
6) ANN

Artificial Neural Network Tutorial provides basic and advanced ideas of ANNs. Our Artificial Neural Network tutorial is developed for beginners moreover as professions.

The term "Artificial neural network" refers to a biologically impressed sub-field of computing sculpturesque when the brain. a man-made neural network is typically a procedure network supported biological neural networks that construct the structure of the human brain. almost like somebody's brain has neurons interconnected to every alternative, artificial neural networks even have neurons that ar coupled to every alternative in numerous layers of the networks. These neurons are called nodes.

Artificial neural network tutorial covers all the aspects associated with the bogus neural network. during this tutorial, we are going to discuss ANNs, adaptational resonance theory, Kohonen self-organizing map, Building blocks, unattended learning, Genetic formula, etc.

The term "Artificial Neural Network" comes from Biological neural networks that develop the structure of somebody's brain. almost like the human brain that has neurons interconnected to at least one another, artificial neural networks even have neurons that ar interconnected to at least one another in numerous layers of the networks. These neurons are called nodes.



Related Work :

LITERATURE SURVEY(8 PAPERS)

BERT and fastText Embeddings for Automatic Detection of Toxic Speech

The ability to speak freely might be abused by certain individuals by contending forcefully, affronting others and spreading verbal savagery. As there is no reasonable qualification between the terms hostile, harsh, disdain and poisonous discourse, in this paper we consider the previously mentioned terms as harmful discourse. In numerous nations, online poisonous discourse is deserving of the law. Subsequently, it is imperative to naturally identify and eliminate poisonous discourse from online medias. Through this work, authors proposed programmed arrangement of poisonous discourse utilizing inserting portrayals of words and profound learning procedures. They performed parallel and multi-class arrangement utilizing a Twitter corpus and study two methodologies:

(an) a strategy which comprises in extricating of word embeddings and afterward utilizing a DNN classifier; (b) tweaking the pre-prepared BERT model. We saw that BERT adjusting performed much better. Proposed philosophy can be utilized for some other sort of online media remarks. They proposed new methodologies for programmed poisonous discourse identification in the online media. These methodologies depend on profound learning classifiers and word embeddings. Authors have investigated the arrangement from two points of view double characterization and multi-class order. For paired characterization they considered 'harmful discourse' (disdain discourse and hostile discourse together) and 'non-poisonous discourse'. For multi-class, they considered disdain discourse, hostile discourse and not one or the other. They proposed highlight based methodologies and tweaking of pre-prepared BERT model. In include based methodologies, fastText and BERT embeddings are utilized as information highlights to CNN and Bi-LSTM classifiers. Further, theye have contrasted these arrangements and adjusting of pre-prepared BERT model.

We saw that BERT tweaking performed far superior to highlight put together methodologies with respect to a Twitter corpus. The fundamental disarrays happen between hostile discourse and disdain discourse.

Overlapping Toxic Sentiment Classification using Deep Neural Architectures

By and large, it turns out to be critical to track such harmful posts/information to trigger required activities for example robotized labeling of posts as wrong. State of-the-craftsmanship arrangement procedures don't deal with the covering estimation classes of text information. In this paper, author proposed Deep Neural Network (DNN) designs to characterize the covering assumptions with high exactness. Additionally, they showed that their proposed order system doesn't need any difficult text pre-preparing and is equipped for taking care of text pre-handling (for example stop word expulsion, include building, and so forth.) naturally. Their experimental approval on a genuine world dataset upholds their claims by demonstrating the unrivaled exhibition of the proposed techniques.
This investigation draws a correlation among DNN models when there is a covering multi-name text order issue. In view of the exact outcomes, thinking about covering text characterization, author prescribed not to invest an excessive amount of energy in information pre-preparing. Truly, stop words, accentuation, and so forth end up being a fundamental constituent of information when it comes to the preparation of the models. Each DNN model shows improvement in model execution estimation measurements when prepared utilizing natural information. Author likewise suggest the utilization of central misfortune to manage imbalanced classes. Central misfortune mitigated the slanted class issue, however very little fundamentally, yet at the same time it didn't worsen the skewness issue. Utilization of numerous model presentation measurements can demonstrate

conclusive in the picking and rating DNN models. In covering multi-name text arrangement it is seen that the per-name estimation of execution measurements verifies the better understanding of model execution. Here five execution measurements, precision, ROC-AUC, accuracy, review and F1 guided us towards the decision of Bi-GRU as the best model for covering multilabel harmful content arrangement.

Automation in Social Networking Comments With the Help of Robust fastText and CNN

These days via online media stages, numerous individuals are underestimating these stages, they consider it to be an chance to pester and target others prompting cyberattack what's more, digital tormenting which lead to horrible encounters and self-destructive endeavors in outrageous cases. Physically distinguishing and ordering such remarks is an extremely long, tedious and untrustworthy cycle. To tackle this test, authors have built up a profound learning framework which will distinguish such negative substance on online conversation stages and effectively characterize them into appropriate marks. Their proposed model expects to apply the content based Convolution Neural System (CNN) with word inserting, utilizing fastText word inserting method. fastText has demonstrated effective and that's only the tip of the iceberg precise outcomes contrasted with Word2Vec and GLOVE model.

Their model plans to improve distinguishing various kinds of harmfulness to improve the web-based media experience. The model orders such remarks in six classes which are Toxic, Severe Poisonous, Obscene, Threat, Insult and Identity-disdain. Multi-Label Order encourages to give a robotized answer for managing the poisonous remarks issue we are confronting.

Author have introduced numerous approaches for harmful remark grouping utilizing fastText what's more, Word2Vec. Here utilizing the grouping acquired, online media stages can execute this framework and check negative effects via web-based media. As they have tried among fastText and Word2Vec and closed from the result, that fastText is more precise and is more exact when managing slangs, languages, composing botches and short structures utilized. The model particularly beats when there is high change in information and dataset size is huge. At long last, authors have introduced a functional framework executed with fastText over a Word2Vec model with exact outcome which is fascinating for research in harmful remark grouping space.

Avoiding Unintended Bias in Toxicity Classification with Neural Networks

A huge number of messages every day distributed by clients of a specific interpersonal organization must be examined progressively for balance to forestall the spread of different illicit or hostile data, dangers and different sorts of poisonous remarks. Obviously, such a lot of data can be handled rapidly just naturally. That is the reason it is important to figure out how to show a PC to "comprehend" a book composed by a man. It is a non-unimportant assignment, regardless of whether "comprehend" here methods just to distinguish or arrange. The quick improvement of AI advancements has prompted the boundless selection of new calculations. Numerous errands that for quite a long time were viewed as practically difficult to illuminate utilizing PC currently can be effectively comprehended with profound learning advances. In this paper, the author presents current ways to deal with taking care of the issue of harmful remarks identification utilizing profound learning innovations and neural systems. The creator presents two best in class neural system structures and furthermore exhibits how to utilize a relevant language portrayal model to identify harmfulness. Besides, in this paper will be introduced the aftereffects of the created calculations, just as the consequences of their group, tried on a huge preparing set, assembled and increased by Google and Jigsaw.

Profound learning innovations can limit human investment in the advancement of calculations, since formation of highlights explicit to a specific assignment is computerized. In this paper it was told the best way to utilize profound NNs to take care of the issue of recognizing poisonous remarks. The outcomes and precision of expectations got by the group and each created model independently, overpowered the aftereffects of the models from recently distributed deals with this point, which demonstrates the achievement of the work done.

Classification of Online Toxic Comments Using Machine Learning Algorithms:

This paper can consistently examine the extent of on-line harassment and classify the content into labels to examine the toxicity as properly as attainable. In this paper, they used six machine learning algorithms and apply them to our information to unravel the problem of text classification and to spot the simplest machine learning formula supported our analysis metrics for toxic comment classification. The main aim of this paper is that the examining the toxicity with high accuracy to limit down its adverse effects which can be an incentive for organizations to require the mandatory steps.

We have mentioned six machine learning techniques i.e. logistic regression, Naïve Bayes, Decision Tree, Random Forest, KNN classification and SVM Classifier, and compared their hamming loss, accuracy, and log loss during this paper. So, the final model choice are going to be supported the combination of playing loss and accuracy. Since we have a tendency to got the maximum accuracy i.e. 89.46% attempt to least possible hamming loss i.e. 2.43% in case of logistic regression.

So, In this paper they will choose logistic regression model as our final machine learning model since it works best for our knowledge.

Classification of Abusive Comments in Social Media using Deep Learning

In this paper, Kaggle's toxic comment dataset is used to train deep learning model and classifying the comments in following categories: toxic, severe toxic, obscene, threat, insult and identity hate. The dataset is trained with various deep learning techniques and analyze which deep learning model is better in the comment classification. The deep learning techniques such as LSTM with and without GLOVE embeddings , a CNN with or without GLOVE are used for classification.
In this paper, they used simple neural network, CNN and LSTM with or without GLOVE pre-trained models. They tend to use feature embeddings and LSTM to determine the foremost salient options of abusive comments. Lastly, they tend to fine-tune the models before testing the, on out datasets, and so compare these neural networks. The dataset which they tend to use is comments form Wikipedia's speak page edits. They tend to tested their models and used accuracy metrics, to know however well completely different model is functioning. The results show that deep RNNs outperforms the associated shallow counterpart the employ the same number of parameters. Deep RNNs can be used for Abuse classification.

Bangla Toxic Comment Classification (Machine Learning and Deep Learning Approach)

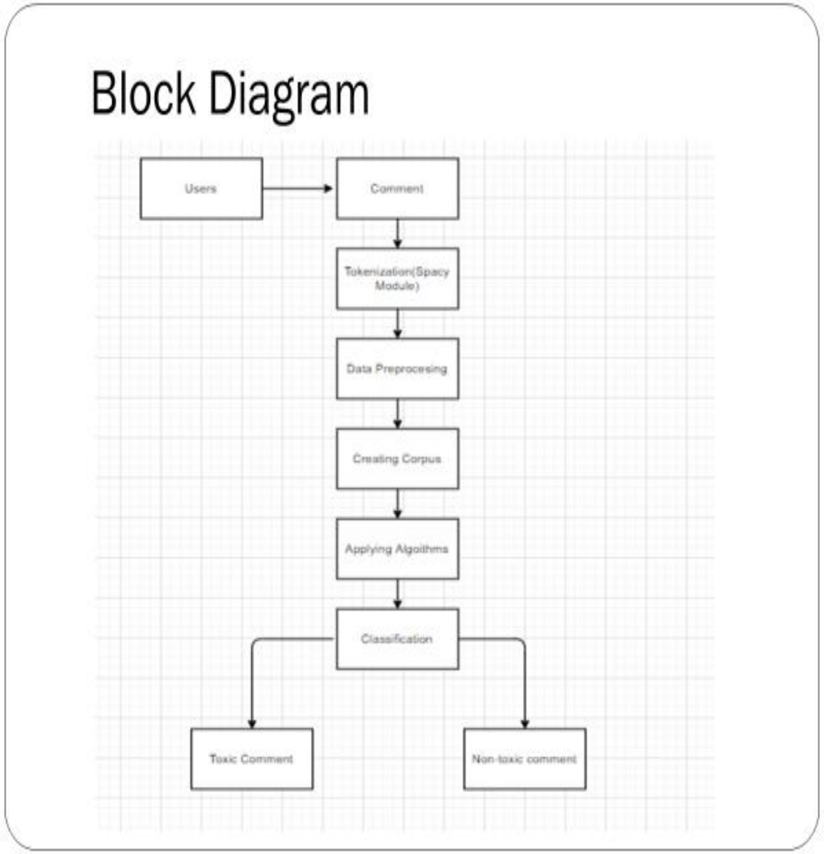
In this work, they have tendency to showed completely different approaches to classify Bangla toxic comments. Among them some worked fine. The Neural Networks worked best among all of classifier. In Bangla language, the NLP connected work is very few. To prevent the hate spreading in online Bangla community, this classifier will perform a significant role. Any system implementing the BP-MLL Neural Network, can perform smart in detecting toxic comment in Bangla online conversation. People ho are toxic in comments, can be warned even at extreme, can be banned from community. As a result, the community environment with each another will be clean.
So, based on the whole discussion, it is clear that BP-MLL Neural Network works well in Bangla Toxic Comment Classification problem. Any system, implemented by BP-MLL Neural Network will give better result. So, Bangla toxic comment in various community can be detected by implementing BP-MLL Neural Network and a system can be developed which will omit the detected toxic comment to make the in-community relationship better.

The accuracy can be improved in future. The stemming could make a dramatic change in the accuracy here. Important thing is, as this is the first attempt to such work with Bangla Language, there is a dataset already created by us for future research. So, people who are interested in work with Bangla NLP, can use the dataset.

TOXIC COMMENT CLASSIFICATION USING NEURAL NETWORKS AND MACHINE LEARNING

Recently the employment of Convolutional Neural Networks and Recurrent Neural Networks are approached for computational purposes for the text classification systems. during this work, authors have utilized this manner to pander to finding toxic comments, remarks in an intensive pool of records given by a current Kaggle's competition with relevance Wikipedia's talk page edits which has divided the extent of toxicity into 6 labels: toxicity, severe toxicity, obscenity, threat, insult or identity hate. There are ceaseless trials experimenting and computing the presence of toxicity of assorted kinds on the web platforms including the micro and macro blogging sites by the industries also because the research communities for an efficient model that detects and predicts the net toxic comments.
this is importance within the research field thanks to the tremendously growing online interactive communication among users. This work is devoted to finding the most effective possible optimum solutions for online toxic comment classification which further classifies the toxic comments into 6 labels provided by the datasets on kaggle platform.

Proposed Methodology :



This is the block diagram of the workflow of the model.

The pre-processing phase:

Before going for any other techniques, data pre-processing is necessary for any applications.

Here we will clean our dataset and make it suitable for further analysis. In NLP, data pre-processing is the very first step to build our model to make the model more efficient and accurate. There are some steps for cleaning the review data:

1) Remove numbers and punctuations if they are not relevant to the analysis.
2) Converting text to lowercase helps to reduce the size of the vocabulary for our text data.
3) Remove non-significant words that are not relevant into predicting whether the review is positive or negative.
4) Stemming: removes affixes at the beginning (prefixes) and the end (suffixes) of the words through string operation. It means taking the root of the words.
5) Tokenization: processing of segmenting running text into sentences and words. It splits all the different reviews into different words (only relevant data).

Firstly we have removed the extra spaces from the text and collected the text into a new variable, and then we have acquired/relocated the words that we are in a need of to their root words by using the BERT Tokeniser. Then we have got our corpse by doing the already known steps, after this we have generated the space matrix where we have separate column for each and every word and if that toxic word is present in the comment or text, we put a 1 in front of the word's row, otherwise a 0 is paced in that particular row.

After these steps are done we are only left with the algorithms that we need to apply to get a proper outcome of the model, **the algorithms** that we have applied are:

- SUPPORT VECTOR MACHINE (SVM)
- Random Forest Classification
- Voting Classification(Decision Tree, Logistic regression and KNN)

- Artificial Neural Network

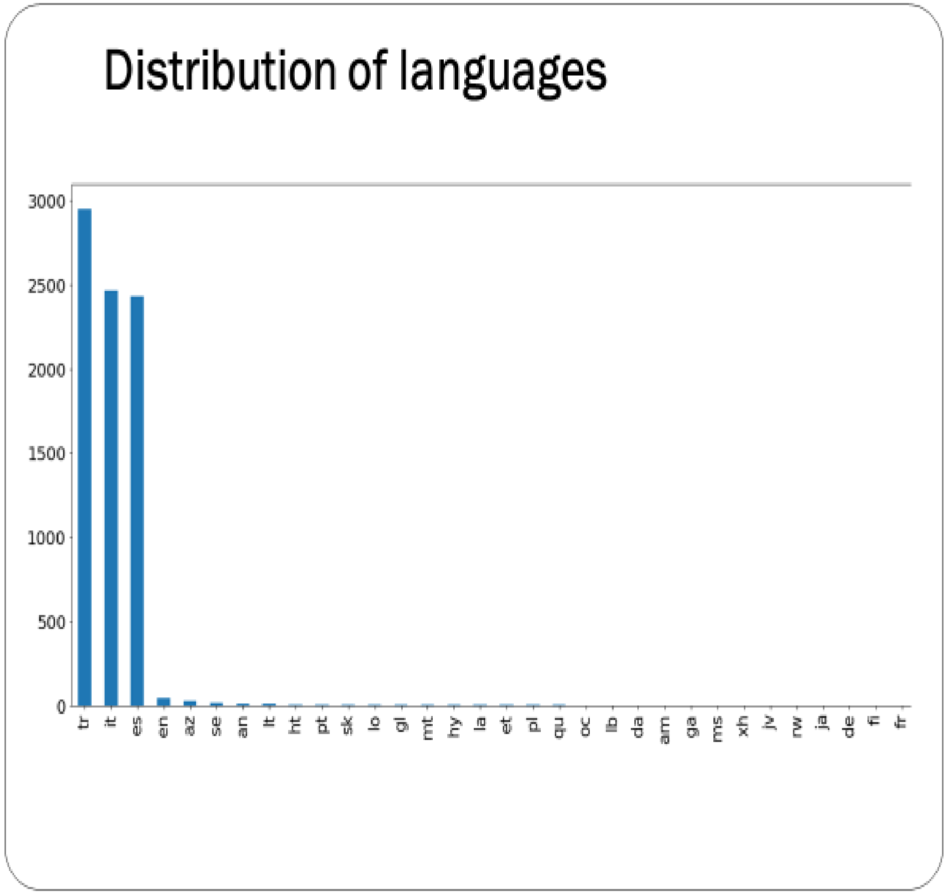
We got different accuracies for different algorithms that we have applied and then we have concluded which algorithm is the most accurate which our model needs.

Jigsaw multilingual toxic comments dataset(Kaggle)

This dataset contains 3 attributes :

1) Comment text

2) Language
3) Toxic label



This graph shows the distribution of different languages in which the comments are written. Our dataset contains 32 languages.

To simplify our model, we have translated all the different languages into one most commonly used language(English).

So,We have another dataset which contains only comments in english.

Remove all stop-words and create our own corpus

cor - List (8000 elements)

Index	Type	Size	Value
47	str	1	good morn examin later took care situat result success time read day 1 ...
48	str	1	sorri express littl harsh honestli descart strang chang spell mistak e ...
49	str	1	excus realli communist illiber cancel sergiu catu niger
50	str	1	poem least year old public date written next book mistak levent abi
51	str	1	kind neutral articl keep seem like pamphlet lament good guy full valu ...
52	str	1	greet would request translat follow english text use halloween iii ite ...
53	str	1	rapsar sometim realli troubl understand bad main substanc name substan ...
54	str	1	hello yetkin could see sea code charact chang exactli vito genoves
55	str	1	sorri gone ask ignor commun issu know help sorri greet lourd messag
56	str	1	make chang articl pleas take care write imparti encycloped languag sig ...
57	str	1	shit shit piss fuck dick cunt cunt cocksuck motherfuck tit nippl carli ...
58	str	1	hi mymmo open siev steven gerrard entri interest give suggest hi thank ...
59	str	1	chinguen reputis mother pendejet regaytonero pendejet pendejero long 1 ...
60	str	1	hello g khan page creat cesar paves jog nomin delet accord wikipedia f ...
61	str	1	page protect man andrea noth copi wiki page classwork realiz teacher c ...
62	str	1	batman begin put actor voic actor accordingli order place movi poster ...
63	str	1	char ezio png licens issu px left file copyright problem thank upload ...

Now, we have created our own corpus by removing all the stop words, punctuations, numbers and after that we used a tokenizer to tokenize the comments present in the dataset.

Code:

```
import pandas as pd # dataframes
import langid # language identification (i.e. what language is this?)
from nltk.classify.textcat import TextCat # language identification from NLTK
from matplotlib.pyplot import plot # not as good as ggplot in R :p
from sklearn.feature_extraction.text import CountVectorizer

import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

# read in our data
dataset = pd.read_csv("comment.csv")
dataset['comment_text'][0:5]

ids_langid = dataset['comment_text'].apply(langid.classify)
# get just the language label
langs = ids_langid.apply(lambda tuple: tuple[0])

# how many unique language labels were applied?
print("Number of tagged languages (estimated):")
print(len(langs.unique()))

langs_df = pd.DataFrame(langs)

# count the number of times we see each language
langs_count = langs_df.comment_text.value_counts()

# horrible-looking barplot
langs_count.plot.bar(figsize=(20,10), fontsize=20)

from nltk.corpus import stopwords

import googletrans

print(googletrans.LANGUAGES)

from googletrans import Translator
translator = Translator()
```

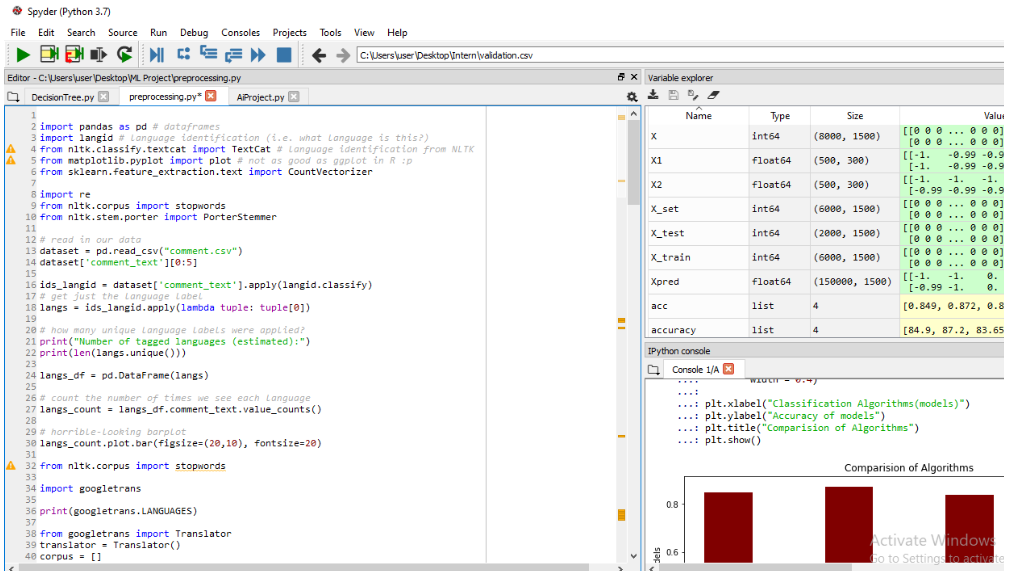
```
corpus = []
for i in range(0,500):
    comment = dataset['comment_text'][i]
    result = translator.translate(comment)
corpus.append(result.text)



from pandas import DataFrame
df = DataFrame(corpus,columns=['comment'])
df['toxic'] = dataset['toxic']
df.to_csv('main.csv')

data = pd.read_csv("main.csv")

cor = []
for i in range(0,8000):
 comment = re.sub('[^a-zA-Z]',' ',data['comment'][i])
 comment = comment.lower()
 comment= comment.split()
ps = PorterStemmer()
 comment = [ps.stem(word) for word in comment if not word in
set(stopwords.words('english'))]
 comment = ' '.join(comment)
cor.append(comment)

 # creating bag of words model
cv = CountVectorizer(max_features = 1500)

X = cv.fit_transform(cor).toarray()
y = data.iloc[:, 2].values

# splitting the data set into training set and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

#SVM
from sklearn.svm import SVC
classifier = SVC(kernel = "rbf" , random_state =0)
classifier.fit(X_train, y_train)
```

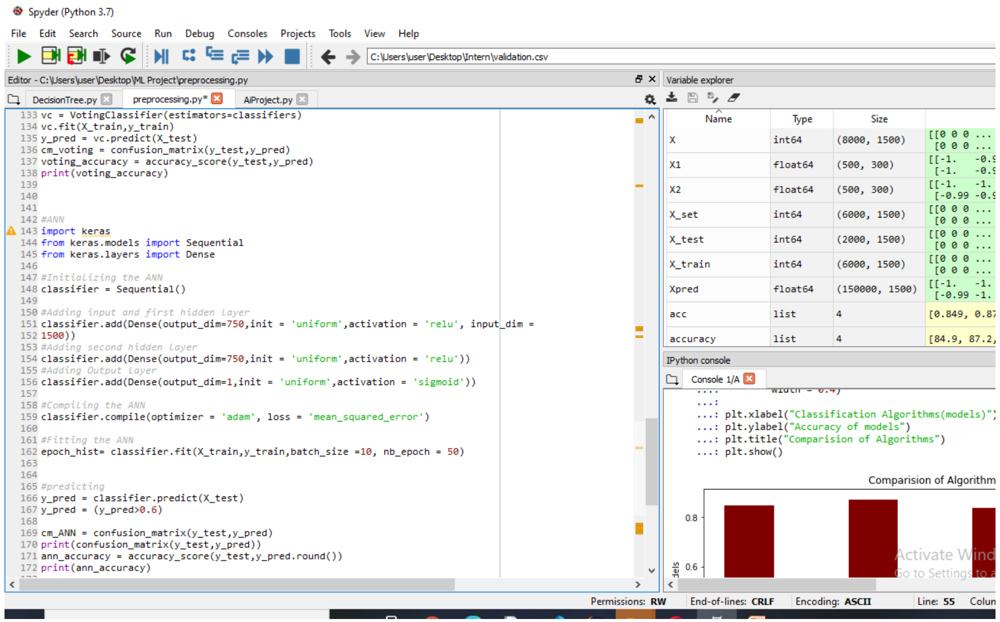
```
#prediction
y_pred = classifier.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score
cm_svm = confusion_matrix(y_test,y_pred)
print(confusion_matrix(y_test,y_pred))
svm_accuracy = accuracy_score(y_test,y_pred.round())
print(svm_accuracy)


#Random Forest
from sklearn.model_selection import RandomizedSearchCV
n_estimators = [int(x) for x in np.linspace(start = 50, stop = 150, num = 10)]
max_depth = [int(x) for x in np.linspace(start = 50, stop = 300, num = 5)]
random_grid = {'n_estimators': n_estimators,
               'max_depth' :max_depth}
rf = RandomForestClassifier()

rf_random  =  RandomizedSearchCV(estimator  =  rf,  param_distributions  =  random_grid,  n_iter  =  10,  cv  =  3,  verbose=2,
random_state=42, n_jobs = -1)
rf_random.fit(X_train, y_train)

rf_random.best_params_


from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators =127, criterion = 'entropy',random_state=0,max_depth=112)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
cm_random = confusion_matrix(y_test,y_pred)
print(confusion_matrix(y_test,y_pred))
random_accuracy = accuracy_score(y_test,y_pred.round())
print(random_accuracy)

#Voting Classifier

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier

lr = LogisticRegression(random_state=42)
```

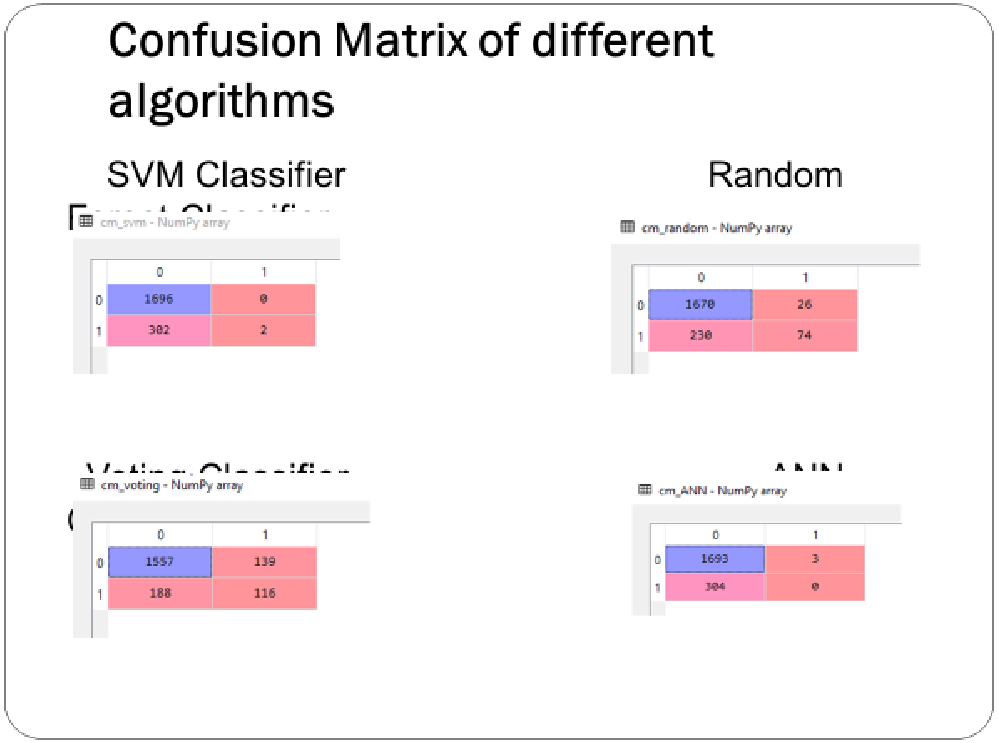
```
knn = KNeighborsClassifier()
dt = DecisionTreeClassifier(random_state=42)

classifiers = [('logisticRegression',lr),
               ('kNearest',knn),
               ('decisionTree',dt)]

for clf_name,clf in classifiers:
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
    print(accuracy_score(y_test,y_pred))

vc = VotingClassifier(estimators=classifiers)
vc.fit(X_train,y_train)
y_pred = vc.predict(X_test)
cm_voting = confusion_matrix(y_test,y_pred)
voting_accuracy = accuracy_score(y_test,y_pred)
print(voting_accuracy)



#ANN
import keras
from keras.models import Sequential
from keras.layers import Dense

#Initializing the ANN
classifier = Sequential()

#Adding input and first hidden layer
classifier.add(Dense(output_dim=750,init = 'uniform',activation = 'relu', input_dim =
1500))
#Adding second hidden layer
classifier.add(Dense(output_dim=750,init = 'uniform',activation = 'relu'))
#Adding Output layer
classifier.add(Dense(output_dim=1,init = 'uniform',activation = 'sigmoid'))

#Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'mean_squared_error')

#Fitting the ANN
```

```
epoch_hist= classifier.fit(X_train,y_train,batch_size =10, nb_epoch = 50)


#predicting
y_pred = classifier.predict(X_test)
y_pred = (y_pred>0.6)

cm_ANN = confusion_matrix(y_test,y_pred)
print(confusion_matrix(y_test,y_pred))
ann_accuracy = accuracy_score(y_test,y_pred.round())
print(ann_accuracy)

#Visualization
plt.plot(epoch_hist.history['loss'])
plt.title('Model Loss Progression During Training/Validation')
plt.ylabel('Training and Validation Losses')
plt.xlabel('Epoch Number')
plt.legend(['Training Loss'])

#Accuracy Comparisions
import matplotlib.pyplot as plt
data = {'SVM Kernel':svm_accuracy, 'Random Forest':random_accuracy, 'Voting Classifier':voting_accuracy,
        'ANN classifier':ann_accuracy}
algos = list(data.keys())
acc = list(data.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(algos,acc, color ='maroon',
        width = 0.5)

plt.xlabel("Classification Algorithms(models)")
plt.ylabel("Accuracy of models")
plt.title("Comparision of Algorithms")
plt.show()
```

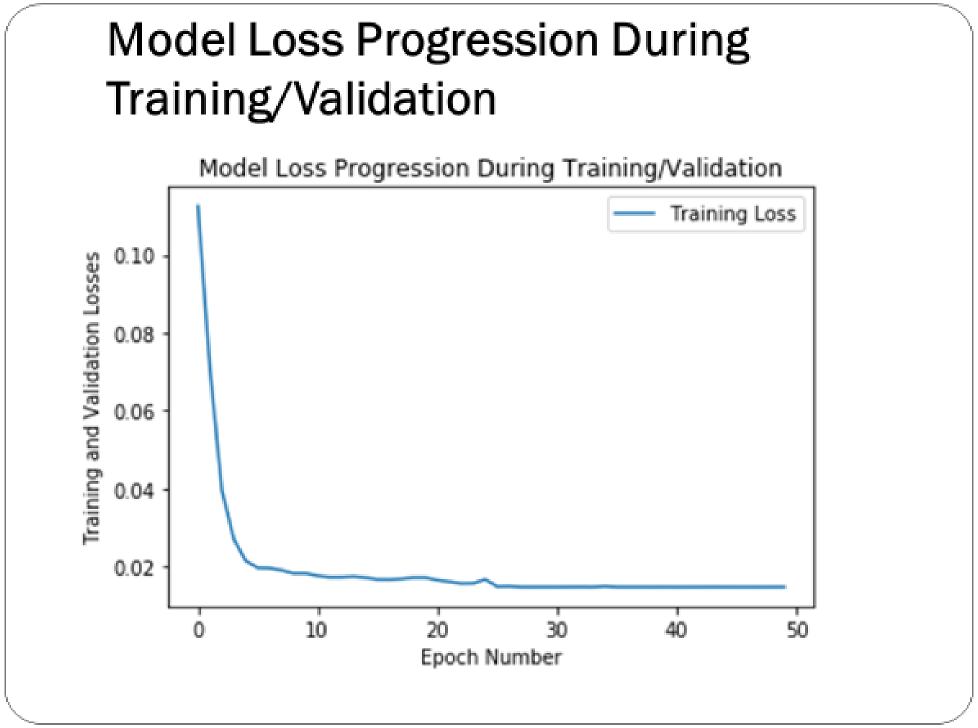
Implementation Screenshot







The above figure shows the confusion matrix of different algos.

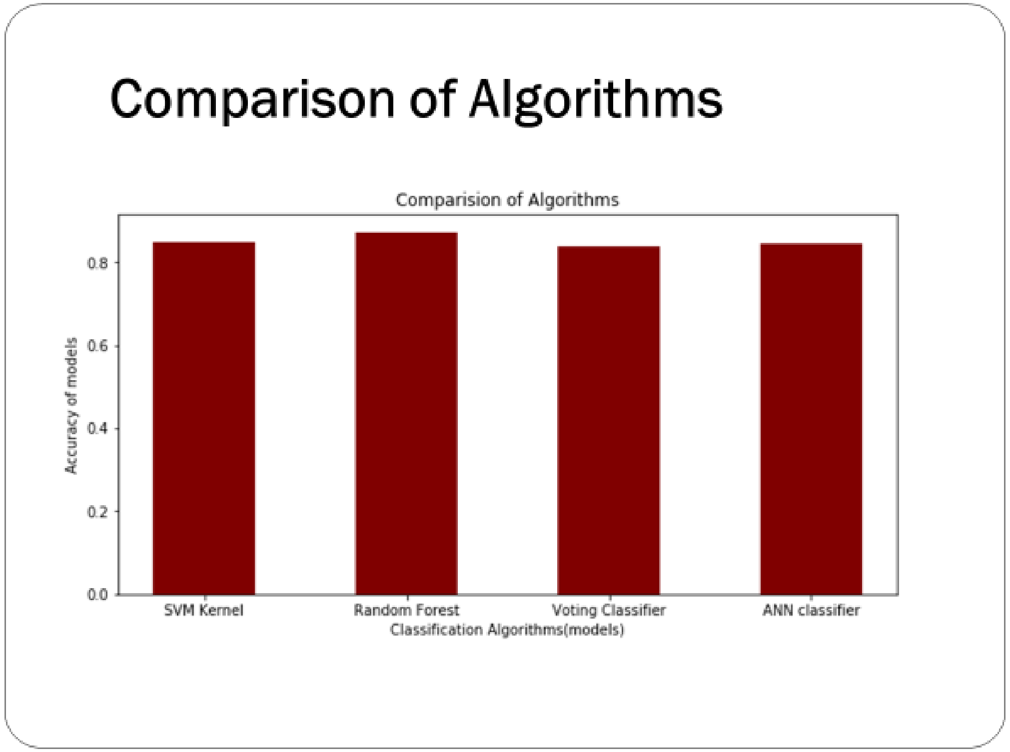


The above figure depicts the use of ANN algorithm. Here, we have used 50 epochs and the size of each epoch is 10. So this graph shows the loss occurred during the execution of each epoch.

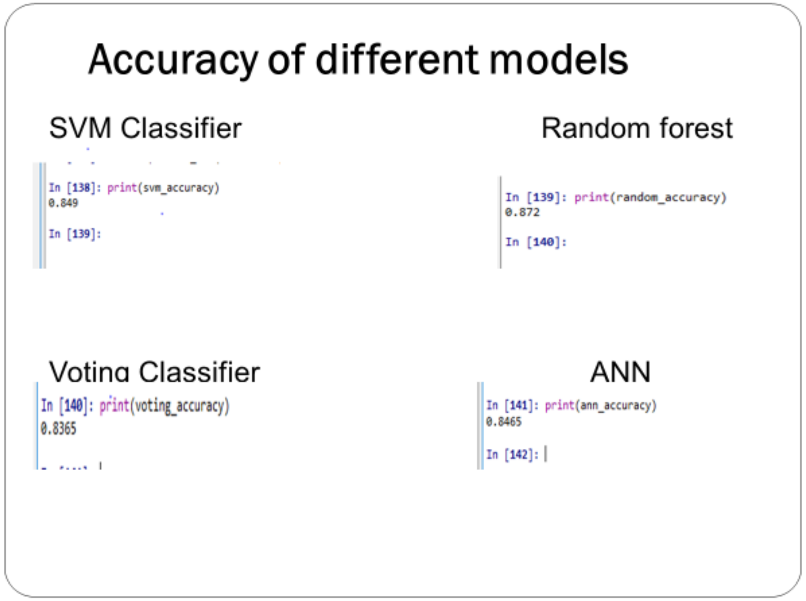
As the number of epochs increases , the loss during execution decreases which depicts that the accuracy of the model is high.

Result and Conclusion :

So in this project we have used four Machine Learning algorithms namely SVM, Random forest, Voting classifier and ANN algorithm. We have taken a dataset from Kaggle to predict whether a comments are toxic or non-toxic. So we have taken some dependent and one independent variable. For dependent variable (Comment) if value is 1 then we can conclude that the comment is toxic and if the value is 0 then we can say that thecomment is non-toxic.



So for our algorithms we have got the accuracy of 84.9(approx.) for SVM algorithm, 87.2 (approx.) for Random forest algorithm, 83.65% (approx.) for Voting classifier algorithm and 84.65% (approx.) for ANN algorithm. So from here we can conclude that Random forest algorithm is more efficient in calculating the result that whether the comment is toxic or not. And it will provide us more accurate results with less number of faulty results.



In this paper, we presented a multilingual hate speech dataset of comments in different languages .We analysed in details the difficulties related to the collection and annotation of this dataset. We have also performed multiatask as well as multilingual learning on our (comment data)corpora and potrayed that deep learning models perform much better than theconventional BOW-based models in major of the multilabel classification tasks. Multilingual multitask learning also facilitated tasks where every label had lesser annotated data linked with it.

References:

- d'Sa, A. G., Illina, I., & Fohr, D. (2020, February). BERT and fastText Embeddings for Automatic Detection of Toxic Speech. In *SIIE 2020-Information Systems and Economic Intelligence*.
- Saeed, H. H., Shahzad, K., & Kamiran, F. (2018, November). Overlapping toxic sentiment classification using deep neural architectures. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 1361-1366). IEEE.
- Mestry, S., Singh, H., Chauhan, R., Bisht, V., & Tiwari, K. (2019, April). Automation in Social Networking Comments With the Help of Robust fastText and CNN. In *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)* (pp. 1-4). IEEE.
- Morzhov, S. (2020, April). Avoiding Unintended Bias in Toxicity Classification with Neural Networks. In *2020 26th Conference of Open Innovations Association (FRUCT)* (pp. 314-320). IEEE.
- Kajla, H., Hooda, J., & Saini, G. (2020, May). Classification of Online Toxic Comments Using Machine Learning Algorithms. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 1119-1123). IEEE.
- Anand, M., & Eswari, R. (2019, March). Classification of Abusive Comments in Social Media using Deep Learning. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 974-977). IEEE.
- Ranasinghe, T., Zampieri, M., & Hettiarachchi, H. (2019). BRUMS at HASOC 2019: Deep Learning Models for Multilingual Hate Speech and Offensive Language Identification. In *FIRE (Working Notes)* (pp. 199-207).
- Jubaer, A. N. M., Sayem, A., & Rahman, M. A. (2019, November). Bangla Toxic Comment Classification (Machine Learning and Deep Learning Approach). In *2019 8th International Conference System Modeling and Advancement in Research Trends (SMART)* (pp. 62-66). IEEE.