Only unique words are present after tokenization that is it provides a list of words which constitutes the sentence. Tokenization is important because the text's meaning may be easily deduced by examining the words in the text.

*3) **Data Pre-Processing**:* After the tokenization of the data, it needs to be pre processed or cleaned only then it can be fed to the machine. Many tokens obtaining from the previous step may provide minimal information or are simply no value of all. Sentences can include numbers or special characters like "@" and "". These tokens do not provide any meaningful information about the context of the sentence. For example, a list of tokens is given as ["Sam's", "house", "number", "is", "684", "", "Sam", "lives", "alone"], if just tokens "684" and "" are removed from the list it becomes ["Sam's", "house", "number", "is", "Sam", "lives", "alone"]. It can be observed that the collection of words can still deduce the meaning of the sentence. Removing numbers and special characters can reduce the space and time complexity of the algorithm.[12] Data can also include words in different cases(upper case and lower case). A machine may not be able to distinguish between them, so all words have to be changed to the same case usually lower case. This will also increase the efficiency of the algorithm.

Removing Stop Words Stop words are the words which are frequently used in a language, some of the English language stop words are ["a", "the", "and", "are"]. The words constitute the major part of the sentence and are low information words. Stop words are commonly regarded as a "single group of words." It can signify a variety of things depending on the use. For example, eliminating all stop words including adjectives which provide quality to the sentence may be an acceptable stop word list in some problems. But in some cases this could lead to loosing critical information for example, in sentimental analysis removing the adjectives ('good', 'bad') can result in losing critical information about the sentence.[13] In such situation, a list of simply determiners, determiners with prepositions, or just coordinating conjunctions may be used.

Finding the Root Word Many words can be generated by combining basic words with various prefixes and suffixes.A root word is a base word to which affixes (prefixes and suffixes) are added in order to give different meaning to the words. The meaning of the word revolves around the root word. Below table show some examples of root words.[14]

TABLE I SHOWING ROOT WORDS.

Root word	Prefix	Suffix
Use	Misuse	Useless
Clear	Unclear	Clearly
Happy	Unhappy	Happiness
Employ	Unemployment	Employment
Act	Enact	Acted

Finding the root word for various tokens and representing them by it also increases the time and space complexity of the model. For example, a token set is given as ["Program", "Programmed", "Programming"], these three words can be represented by a single root word "program". There are mainly two techniques in natural language processing for finding a root word Stemming and Lemmatization.

1) **Stemming** : Stemming is a process of retrieving the root word from a given a word. In stemming the context of the word in the sentence is discarded and it is treated as independent. It does so by removing the affixes from the word [15]. Affixes are group of letters which are added at the beginning or end of the word in order to give a new meaning to the word. Affixes which are added at the beginning of the word are called Prefixes and affixes which when placed at the end of the word are called suffixes.
   Stemming trims down the word in order to remove affixes and eventually reach to the root word. Most of the stemming algorithms remove only suffixes. One of most used algorithm to remove suffixes is Porters Stemmer. Porter Stemmer takes approximately all the suffixes into account and observes the way words are combined together, rules are created based on these observation. This algorithm uses different rules for removing different suffixes in a word. For example, the rule that states if the word has at least one vowel and consonant, as well as the ending "EED," the ending should be changed to "EE.". So "disagreed" becomes "disagree"
   It is also used in search engine optimisation, various search engines have adopted to the stemming technique because on searching for a word on the internet the algorithm returned irrelevant searched, for example, on searching for "employment" it also gave result including "Unemployment" ,"Employee". This problem increases the time for searching.[16] Errors could happen while performing above stemming approach. There are two types of errors that could happen, which are Over Stemming and Under Stemming.
   Over Stemming: Over stemming is the practise of removing a considerably bigger portion of a word than is necessary. By doing so, it can trim two or more words in such a way that the same root word is generated while there should have two or more root words representing different words or tokens. For example these set of words are given "City" and "Citation". Some stemming algorithms may reduce both the words to the root word "City", this would suggest that both terms signify the same thing, which is patently incorrect.
   Under Stemming: Under stemming can result in the improper reduction of two or more words to more than one root word while they should have been reduced to the same root word. For example, two words are given as "trouble" and "troublesome", some algorithms will trim these words to "troubl" and "troublesom". Both of the words have to be reduced to the same root word. So, choosing the algorithm to stem the word should be done

carefully. Over stemming and Under stemming works proportionally that is if one is decreased the chance of the other to increase is more

2) **Lemmatization** : Lemma is the reference word that is registered in the dictionary, a lemma word can be a reference of many words. For example, words like ”come”, ”coming”, ”came” are having same lemma ”come”. Basically lemma is the root word for the given words. Stemming and Lemmatization gets confused with each other, the basic difference is that in stemming a set of rules are given by which trimming or stemming of the words are done while as in Lemmatization a dictionary is given where lemmas are linked with different words. Stemming can sometimes provide root words which do not have any meaning while as in lemmatization root words are predefined in the dictionary. Programming languages have packages which include the type of corpus or database where lemmas of different words could be found. Parts of speech can also be defined in lemmatization, it allows to find lemmas in different tenses. Lemmatization is more accurate than stemming. Below is the table that shows the difference between stemming and lemmatization.[17]

TABLE II SHOWING STEMMING AND LEMMATIZATION.

Word	Stemming	Lemmatization
Won	Won	Win
Relies	Reli	Rely
Remembered	Rememb	Remember
Ate	Ate	Eat

*4) **Word Vectorization**:* Till now a list of unique tokens is created by removing numbers, special characters, stop words and converting every word to the lower case. Each token has been reduced to its root word by either stemming or lemmatization. As it is discussed earlier that machine does not understand the categorical data it only understands numbers and can only do calculations on numbers. So, tokens are con-verted into numbers or vectors. The process is known as Word Vectorization or Word Embeddings, where words or phrases in the dictionary are represented by a vector of real numbers. These numbers are used to perform mathematical calculations to determine word predictions and word similarities/semantics. Word similarities are further calculated using either Euclidean Distance or Cosine Similarities.[18]

Vectorization may be accomplished in a variety of meth-ods, as it will be discussed shortly, ranging from simple binary word occurrence features to complex context-aware feature representations. Depending on the data set and problem statement one of the vectorization methods is applied. Two methods of the vectorization methods will be discussed, One hot Encoding and TF-IDF Vectorizer.

One hot Encoding: The tokens and document indexes are put into a tabular fashion where every unique token is represents a column of a table, and each document index is a row in the matrix. The value of each cell is nothing but the 1 in case if the word is present in the particular text sample or 0 otherwise. One hot encoding is useful for data that has no relationship to each other.

The idea is simple, create a vector that has as many dimensions as your corpus has words or tokens. Each to-ken has a unique dimension and will be represented by 1 or 0. Suppose three sets of tokens generated after execut-
ing previously described techniques, 1. [”name”, ”tom”], 2. [”tom”,”live”, ”USA”], 3.[”mom”, ”live”,”him”]. Below is the figure that shows how one hot encoding will transform into vector matrix.[19]

TABLE III SHOWING ONE HOT ENCODING.

Token set	”name	”tom”	”live”	”USA”	”mom”	”him”
1	1	1	0	0	0	0
2	0	1	1	1	0	0
3	0	0	1	0	1	1

The benefit of One-Hot-Encoding is that the output is binary rather than ordinal and that everything is in an orthogonal vector space. The disadvantage is When the number of output labels is huge, it does not scale properly. In language mod-elling, for example, the number of output labels equals the size of the vocabulary. As a result, each input feature (word) will be represented as a massive vector.[20]

TF-IDF Vectorizer: Full form of TF is Term frequency, which indicates how frequently a word is present in the doc-ument and full form of IDF is Inverse Document Frequency, which indicated how frequently a word is present in different documents. The idea behind TF-IDF vectorizer is that, if a word is present more frequently across the corpus then the word is having more importance and more weightage will be given to that word. When performing mathematical calculation on these assigned vectors, then these more frequently occur-ring words will have more impact on the calculations.[21]

There are mathematical formulas for determining the values of TF and IDF for every word in the document

$$TF = \frac{\text{Number of times a word is present in doc}}{\text{Total number of words in the doc}} \tag{1}$$

$$IDF = \log \frac{\text{Total number of documents}}{\text{Number of documents that contain the word}} \tag{2}$$

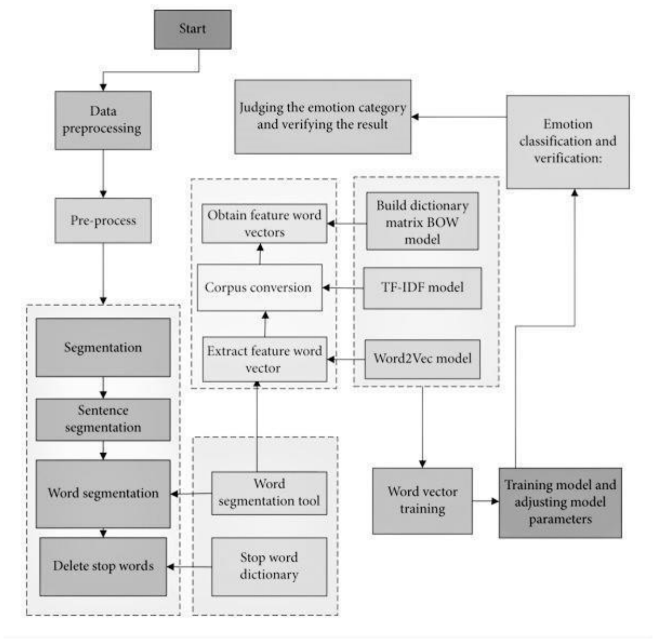
The value of TF-IDF is simply obtained by multiplying the above values that are TF multiplied by IDF. Every word or token will have its TF-IDF value and these scores represent the word importance. The higher the TF-IDF score for a word the more important it is to the document. Below is the table showing the TF-IDF values of previously used data. TF-IDF is most frequently used technique for recognizing the most important word in the corpus. The numerical weight assigned to each word gives programmer a flexibility to choose the words with higher weightage. Hence it reduce complexities by removing low weightage words unlike One hot encoding which cannot recognize the importance of words, it just shows if a word is present or not [22].

TABLE IV SHOWING TF-IDF VALUES.

Token set	”name	”tom”	”live”	”USA”	”mom”	”him”
1	0.23	0.08	0	0	0	0
2	0	0.23	0.058	0.15	0	0
3	0	0	0.058	0	0.15	0.15

A detailed Flow chart of Natural language Processing is given below in figure 2:

Fig. 2. NLP FLow chart


Start
Data preprocessing
Pre-process
Segmentation
Sentence segmentation
Word segmentation
Delete stop words
Judging the emotion category and verifying the result
Obtain feature word vectors
Corpus conversion
Extract feature word vector
Word segmentation tool
Stop word dictionary
Build dictionary matrix BOW model
TF-IDF model
Word2Vec model
Word vector training
Training model and adjusting model parameters
Emotion classification and verification:

IV. Results and Analysis

: After changing the words into numbers, the next thing is to feed these vectors to a machine learning model and predict our outcome. The data needs to be split into train and test sections, so that we can validate our model for unseen data. We have used five algorithms namely Logistic regression, XG boost classifier, Multi layer perceptron classifier, Naive Bayes and Support vector classifier. The evaluation metrics used are Accuracy score, classification report and confusion matrix. All of the mentioned algorithms and metrics can be implemented using python’s sklearn module. Below are some graphs showing comparison of these algorithms. We can observe that MLP and XG boost classifier have considerably high accuracy rates.

Fig. 3. Accuracy Scores

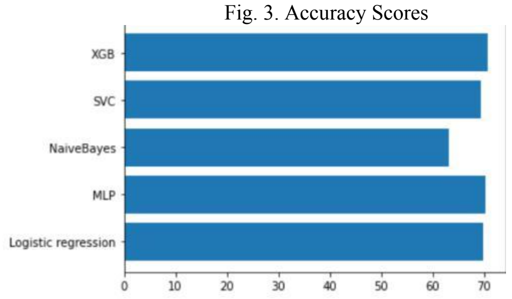


Fig. 4. Recall Scores

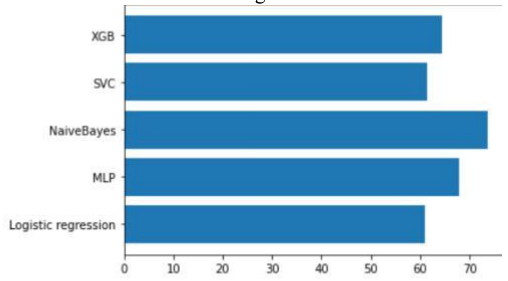
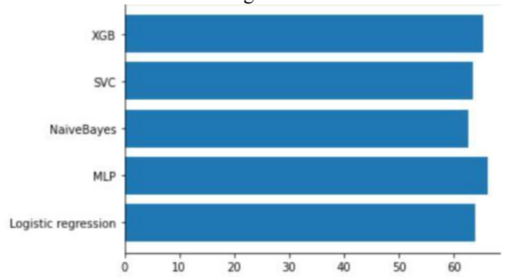


Fig. 5. Precision Scores



A. Applications

Natural language processing has seen immense growth in the technological world for the past previous few years that no one could have predicted. In order to understand how Natural language processing is impacting on our lives, understanding the real life application is necessary. Some of the few applications of our model are as below

- **Automated Paper evaluation in Universities**: In universities scholars tend to write research papers and faculty of that university evaluates the paper. Scores are provided by the faculty and they add up to the final marks of the scholars. Our model can be beneficial in a way where a faculty member can write a review for a paper and then that review is fed to our model. Our model will be able to predict the probabilities for ”accept” and ”reject” classes and based on those probabilities machine can automatically assign marks.