Sentiment Analysis on Social Media Text (Hinglish)

Thesis Submitted
in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
IN
Machine Intelligence and Data Analytics

by
Kurakula James Paul
2020PMD4211

Under the Supervision of
DR. Vikas Maheshkar
Netaji Subhas University of Technology (Formerly NSIT), Delhi

To the

FACULTY OF INFORMATION TECHNOLOGY

NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY
(Formerly Netaji Subhas Institute of Technology)
Azad Hind Fauj Marg, Sector-3, Dwarka, New Delhi-110078

January, 2022

CERTIFICATE

Certified that **Kurakula James Paul (2020PMD4211**) has carried out their search work presented in this thesis entitled **"Sentiment Analysis on Social Media Text (Hinglish)"** for the award of **Master of Technology** from Netaji Subhas University of Technology, New Delhi, under my supervision. The thesis embodies results of original work, and studies are carried out by the student himself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/ Institution.

[illegible]

Signature

DR. VIKAS MAHESHKAR
Assistant Professor
Department of Information Technology,
Netaji Subhas University of Technology

Date: 16/01/2022

CONTENTS

ABSTRACT

Sentiment Analysis plays an Important role in monitoring Social media and as well customer reviews by obtaining the polarity of a tweet or comment or statement in social media.These days everyone is active on Social media and expressing their views in the digital world. Based on these views we can make decisions like which topic is trending currently or reacting to customer reviews based on customer feedback etc. Many ML techniques can be used to draw out the polarity of a statement like MNB , SVM, LR, RF etc and Deep learning techniques like some extension to RNNs like , LSTM ,GRU ,Bi-LSTM ,Bi-GRU and some advanced techniques like Transformers and Bert models .For the vector representation of the words in the sentences we used TF-IDF vectorizer for machine learning algorithms. This paper gives some rough idea of comparative study of ML algorithms applied on code mixed/Multi Language mixed data set like Hinglish i.e where each data point is a combination of english and hindi words and each english word is followed by ENG and each Hindi word followed by HIN. We Got this dataset from a codalab competition website named as Task:9 on the Semeval-2020 competition website.So far comparing all the algorithms on this dataset with F1 score as a metric .we got 0.771 ensemble soft voting classifier of (MNB , SVM, LR, RF) which is the best among all we have applied.

List of Figures

List of Tables

Abbreviations :

NLP- Natural language processing

MLP - Multilayer Perceptron

Bert -Bi directional encoder representations

ML- Machine learning

DL- Deep Learning

LR- Logistic Regression

SVM- Support vector machines

MNB -Multi Nominal Bayes

RF -Random Forest

LSTM -Longest short term memory

GRU - Gated recurrent unit

Bi-LSTM -Bi directional Longest short term memory

Bi-GRU - Bii directional Gated Recurrent unit

CNN- Convolutional Neural network

Neg -Negative

Pos- Positive

Neu- Neutral

Pre- Precision

Rec -Recall

F1- sco -F1 Score

Supp- Support

Acc -Accuracy

Mac Avg -Macro Average

Wei Avg -Weighted Average

TN- True Negative

FN -False Negative

FP -False Positive

TP -True Positive

Sentiment Analysis on Social Media Text (Hinglish)

Vikas Maheshkar[1] , James paul Kurakula[2]

vikas.maheshkar@nsut.ac.in[1], kurakulaj.md20@nsut.ac.in[2]

Key terms: Sentiment Analysis, Code mixing , Hinglish , NLP, ML algorithms ,DL algorithms..

1. Introduction:

Sentiment analysis is the technique for extracting information from text such as opinion, belief or point of view. Sentiment analysis focuses on categorising the views of a group of people. It tries to find whether someone has a neutral, negative or positive opinion towards something.Sentiment analysis is also named opinion mining, review mining, emotion analysis, views extraction etc. with respect to the different tasks.

As the use of social media has increased enormously, opinion mining or sentiment analysis are good tools that are used to analyse the views of people on politics, products, environment etc. It gives a good sense of what a large section of people has views on a particular topic, categorising it into either positive or negative. Sentiment analysis mainly focuses on three levels i.e. document level, aspect level and sentence level. Most of the statements or reviews that are found online are of few words and in code mixed with languages.Here we are considering the sentence level sentiment analysis and on the reviews like hindi statements being written in english or combination of hindi words and english words.

Usually Sentiment Analysis is done in three levels, document level which focuses on a document or set of paragraphs and determining the polarity of that document with respect to topic whether it is reflecting positive or negative. If neither of these two then it can come in to category of neutral , another level is sentence level which is the sentence might be a movie review or review on some product or some statement about a topic .sometimes neither document level or sentence level is not enough to conclude whether its positive or negative.like some ambiguity between both positive and negative since both likes and dislikes are there in that sentence. In that case Aspect level based sentiment analysis is used to directly check the opinion based on features or characteristics or dislikes in that sentence. So here feature level sentiment analysis might be required here in these cases to conclude the feedback.here we are using sentence level sentiment analysis.

2. Literature Survey:

In the paper with reference to [1] the research conducted by Gaurav Singh in 2021 .the maximum f1 score achieved among all experiments was 0.6907. He considered many iterations of preprocessing on the data set and applied machine learning algorithms on it like SVM,KNN,RF, MNB, LR(logistic) and ensemble soft voting classifiers. He considered many embeddings like one hot encoding, count vectorizer, word2vec, glove ,tf-idf vectorizer and conducted experiments on all combinations of these, the best model obtained was ensemble voting classifier of SVM,LR and RF with n estimators as 750 achieving the f1 score as 0.6907.

In the paper with reference to [3] conducted by S Baroi, N Singh ,R Das, T D Singh in 2020, they achieved the maximum f1 score of 0.617 using deep learning algorithms like lstm,bi lstm,cnn and ensemble voting classifier of these. The ensemble model of LSTM ,bi-LSTM ,LSTM+CNN ,CNN achieved the maximum f1 score of 0.617.

In the paper with reference to [2] conducted by Vivek Kumar Gupta in 2019 on the dataset consisting of three categories as Non offensive , offensive and hate inducing . he achieved the maximum accuracy of 0.79 considering the bi directional lstm, bidirectional gru, Besides this,the data set considered here is too small with 7934 data points in train dataset with hate inducing 2724, offensive 1638,non offensive 3572, and the test dataset consisting 700 data points representing non offensive as 228, offensive as 69 and hate inducing as 403.

With reference to [4] research conducted by Mishra et al. in 2018 performing on hinglish dataset obtained the maximum f1 score of 0.58 using the char(2,6) on tf-idf vectors using the machine learning algorithm support vector machines. The research conducted by Patra et al.with reference to [9] in((2018) gave the report of all the models from different institutions who participated in that competition. The top model performed by IIIT-NBP team got the maximum f1 score of 0.569.with the total data points as 12936 in train and 5525 in test.

.

3 .Dataset Overview:

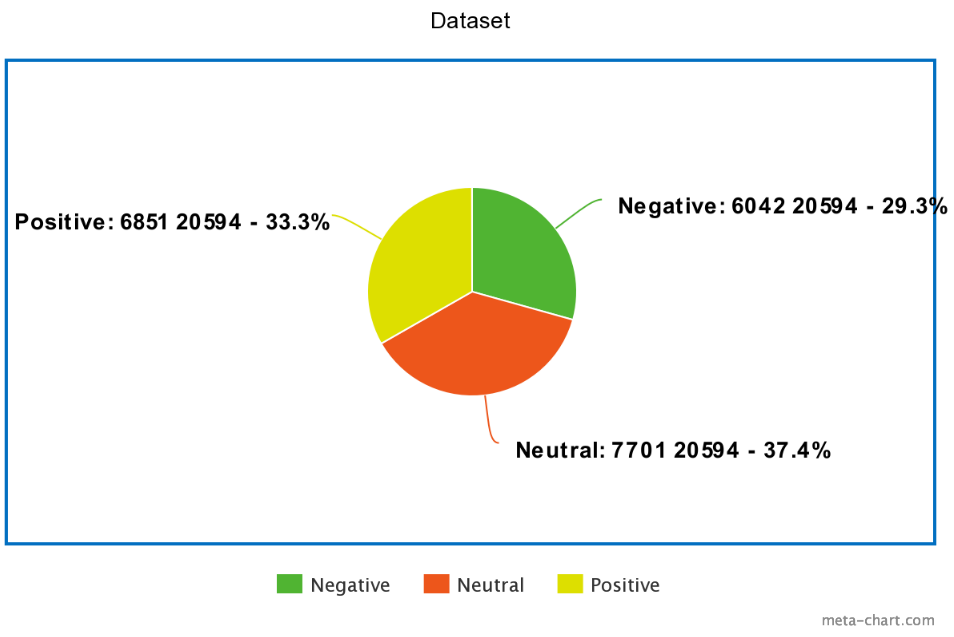
The dataset is obtained from a codalab competition website named as Task:9 on the Semeval-2020 competition website or refer to the github link of paper [3] . We combined all the train, validation ,test datasets into one dataset and did random split of 70:30 for train and test.Where the columns are ID,Text and polarity of categories positive, negative and neutral of 20594 points in which 6851 labelled as positive ,7701 points as neutral ,and 6042 as negative. We have done the random split of 70

:30 for the Train and Test dataset. Each sentence is a combination of english and hindi words where each word is followed by ENG if english word and HIN if hindi word.One instance of this dataset is given below.

Ex:

1	CongressENG madeENG terribleENG mistakeENG dividingENG AndhraENG pradeshENG intoENG twoENG statesENG keseeaarHIN neHIN kaangresHIN koHIN bevakoophHIN banaayaHIN	Negative

Figure 1.1 Dataset Overview



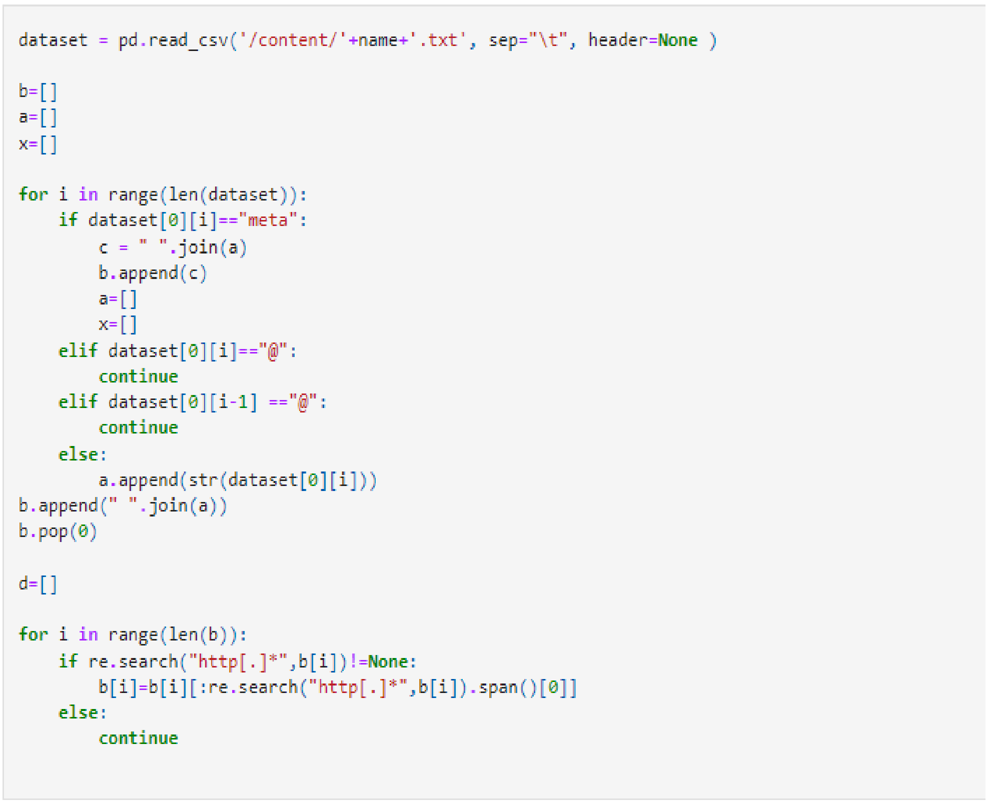
Refer: : https://github.com/jame27-1998/Sentiment_Analysis_Hinglish

In addition to this dataset there are some other code mixed language datasets like Spanish+English (spanglish),Tamil+English (Tamlish) , Malayalam+English (Malayalish) and try with different text embeddings like bag of words, tf-idf, glove vectors, word2vec with respect to different machine learning algorithms to know how results are varying [3].

4. Data Preprocessing:

- Removing the noise in every sentence like mentions which starts with @ and url links like http and https.

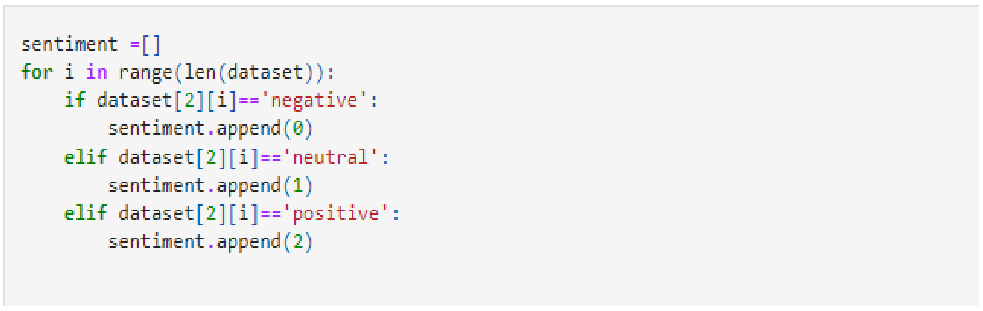
Figure 2.1 Removing Noise



```python
dataset = pd.read_csv('/content/'+name+'.txt', sep="\t", header=None )

b=[]
a=[]
x=[]

for i in range(len(dataset)):
    if dataset[0][i]=="meta":
        c = " ".join(a)
        b.append(c)
        a=[]
        x=[]
    elif dataset[0][i]=="@":
        continue
    elif dataset[0][i-1] =="@":
        continue
    else:
        a.append(str(dataset[0][i]))
b.append(" ".join(a))
b.pop(0)

d=[]

for i in range(len(b)):
    if re.search("http[.]*",b[i])!=None:
        b[i]=b[i][:re.search("http[.]*",b[i]).span()[0]]
    else:
        continue
```

- Labelling the categories with negative as 0 neutral as 1 and positive as 2.

Figure 2.2 Labelling

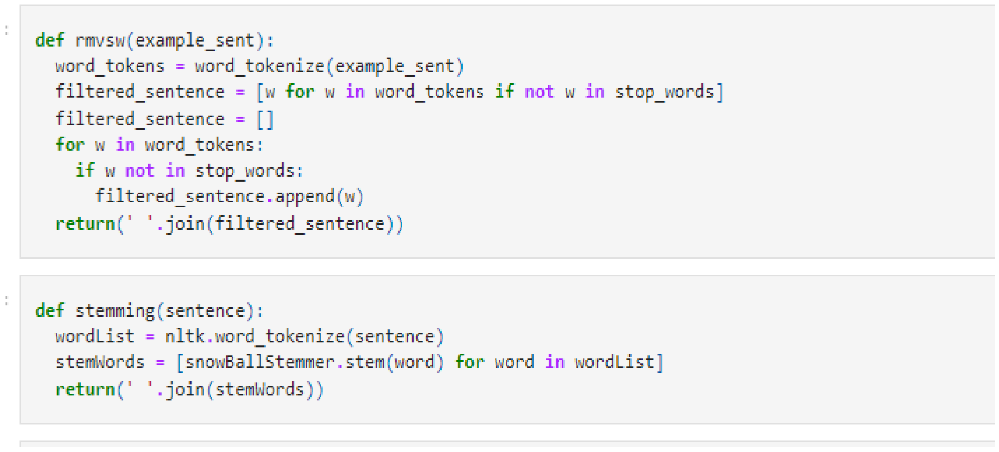


```python
sentiment =[]
for i in range(len(dataset)):
    if dataset[2][i]=='negative':
        sentiment.append(0)
    elif dataset[2][i]=='neutral':
        sentiment.append(1)
    elif dataset[2][i]=='positive':
        sentiment.append(2)
```

- Removing the english stop words since we have some collection of english stop words in nltk.

- Making all the words which give the common meaning like synonyms too some base word using stemming. We used a snowball stemmer algorithm.

Figure 2.3 Stop Words Removal and Snowball Stemmer



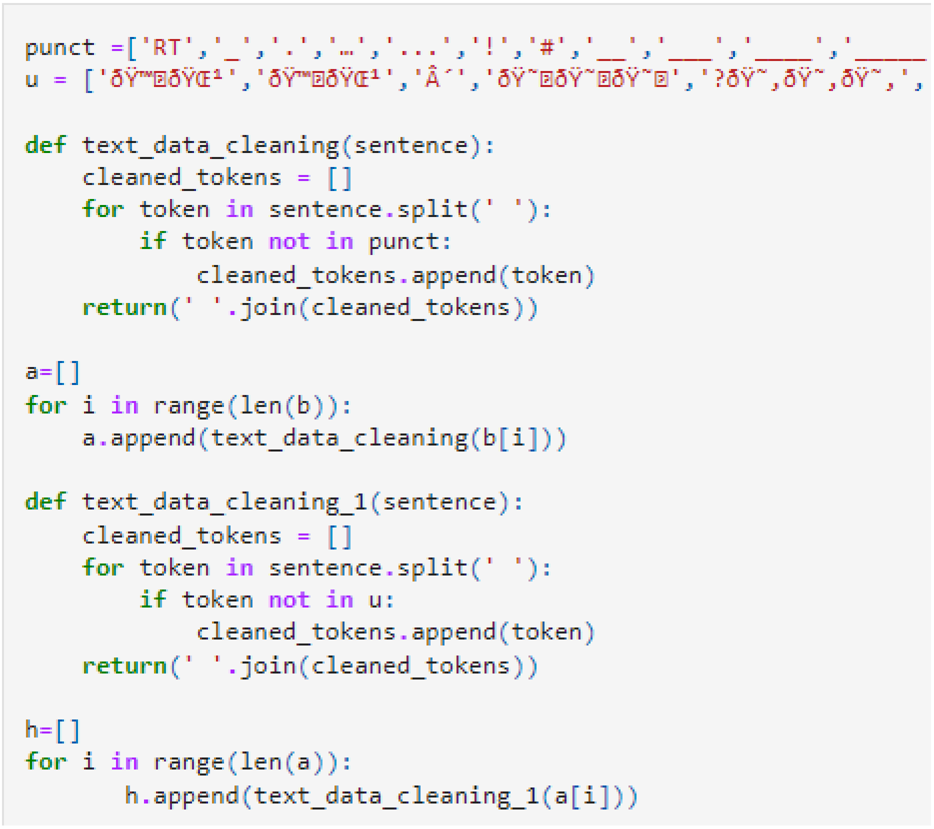
```
def rmvsw(example_sent):
  word_tokens = word_tokenize(example_sent)
  filtered_sentence = [w for w in word_tokens if not w in stop_words]
  filtered_sentence = []
  for w in word_tokens:
    if w not in stop_words:
      filtered_sentence.append(w)
  return(' '.join(filtered_sentence))
```

```
def stemming(sentence):
  wordList = nltk.word_tokenize(sentence)
  stemWords = [snowBallStemmer.stem(word) for word in wordList]
  return(' '.join(stemWords))
```

.

- Removing all the special symbols and punctuation marks.

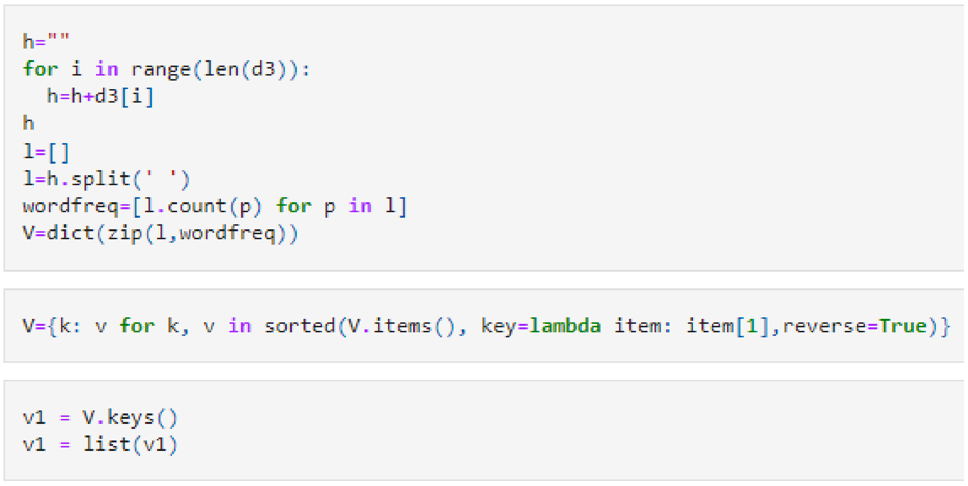
Figure 2.4 Removal of Special Symbols, Mentions @, Hyperlinks http



```
punct =['RT','_','.','…','...','!','#','__','___','____','_____
u = ['ðŸ™ðŸŒ¹','ðŸ™ðŸŒ¹','Â´','ðŸ˜ðŸ˜ðŸ˜','?ðŸ˜,ðŸ˜,ðŸ˜,',

def text_data_cleaning(sentence):
    cleaned_tokens = []
    for token in sentence.split(' '):
        if token not in punct:
            cleaned_tokens.append(token)
    return(' '.join(cleaned_tokens))

a=[]
for i in range(len(b)):
    a.append(text_data_cleaning(b[i]))

def text_data_cleaning_1(sentence):
    cleaned_tokens = []
    for token in sentence.split(' '):
        if token not in u:
            cleaned_tokens.append(token)
    return(' '.join(cleaned_tokens))

h=[]
for i in range(len(a)):
      h.append(text_data_cleaning_1(a[i]))
```

- Since there is no collection of stop words in hindi we removed all the most frequent words in the entire dataset which crossed some threshold as 1000 times.

Figure 2.5 Generating Top 1000 Hindi Words



```
h=""
for i in range(len(d3)):
  h=h+d3[i]
h
l=[]
l=h.split(' ')
wordfreq=[l.count(p) for p in l]
V=dict(zip(l,wordfreq))
```

```
V={k: v for k, v in sorted(V.items(), key=lambda item: item[1],reverse=True)}
```

```
v1 = V.keys()
v1 = list(v1)
```

5 .Metrics:

5.1 Three class Confusion matrix :

Table 1.1 : Three class Confusion matrix

	Predicted class (Neg)-0	**Predicted class (Neu)-1**	**Predicted class (Pos)-2**
Actual Class (Neg)-0	+ve 1	-ve 2	-ve 3
Actual Class (Neu)-1	-ve 4	+ve 5	-ve 6
Actual Class (Pos)-2	-ve 7	- ve 8	+ve 9

Figure 3.1 Possible Outcomes for each class

Negative-0	Neutral-1	Positive-2
TP = $Cell_1$	TP = $Cell_5$	TP = $Cell_9$
FP = $Cell_2 + Cell_3$	FP = $Cell_4 + Cell_6$	FP = $Cell_7 + Cell_8$
TN = $Cell_5 + Cell_6 + Cell_8 + Cell_9$	TN = $Cell_1 + Cell_3 + Cell_7 + Cell_9$	TN = $Cell_1 + Cell_2 + Cell_4 + Cell_5$
FN = $Cell_4 + Cell_7$	FN = $Cell_2 + Cell_8$	FN = $Cell_3 + Cell_6$

5.2 Accuracy:

Accuracy is the ratio of all correct predictions predicted by model with respect to the actual original values.

$$Accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ total\ predictions}$$

5.3 Precision:

Precision is the ratio of True Positives and Sum of True Positives and False Positives.

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

5.4 Recall: Recall is the ratio of True Positives and Sum of True Positives and False Negatives.

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

5.5 F1-Score: F1 score combines both precision and recall into a single metric. it means that
the F1 score gives equal weight to Precision and Recall:

- If both Precision and Recall are high a model will obtain a **high F1 score**.
- if both Precision and Recall are low a model will obtain a **low F1 score**.
- if one of Precision and Recall is low and the other is high.
- a model will obtain a **medium F1 score**

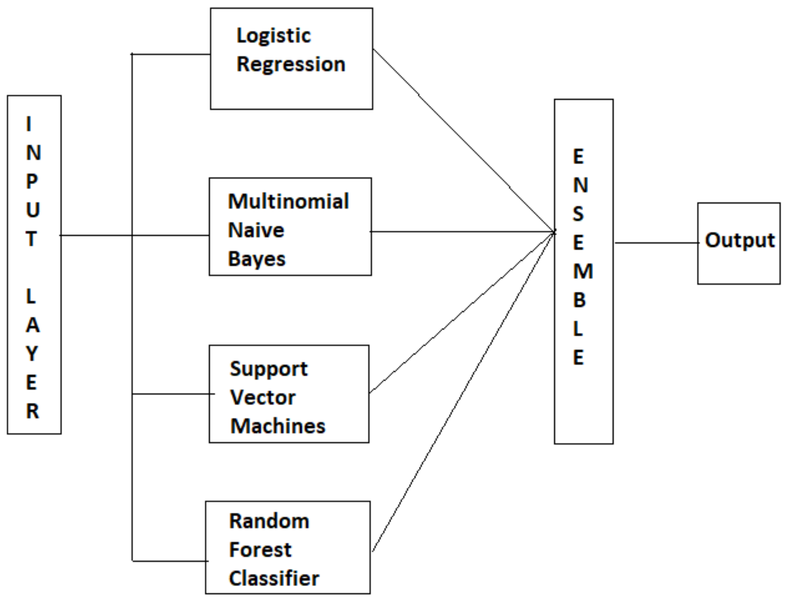
$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

After all this preprocessing we used tf-idf as embedding and we also applied machine learning algorithms like LR (logistic) ,SVM ,MNB ,RF, with n=10 estimators on this preprocessed data set. The classification reports of each of these algorithms are shown below however the ensemble voting classifier of all these achieved 0.7715 f1-score.

6. Model Architecture:

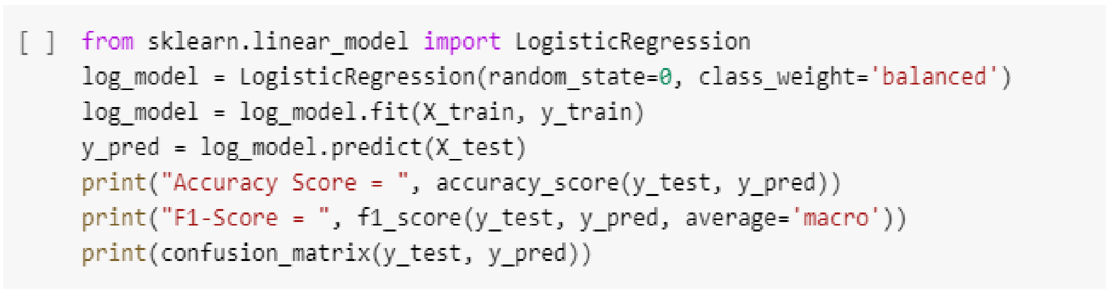
The working strategy here is the ensemble of LR (Logistic), SVM , MNB, and RF techniques.The parameters for the logistic regression are random state as 0 and class weight as balanced. The parameters for the support vector machines are set as random state as 0, class weight as balanced as kernel as linear. For the Random forest algorithm we tried estimators as n 10,50,250 however n=250 giving better results compared to others. And finally the parameter probability is set as true for SVM in the ensemble model. Each of this model is appended and the voting classifiers gives the argmax of sum of probabilities for these class labels

Figure 4.1 Model Architecture



6.1 Sub strategy 1 :Logistic Regression:

Figure 5.1 Sub Strategy 1 : Logistic Regression Model



```
[ ]  from sklearn.linear_model import LogisticRegression
     log_model = LogisticRegression(random_state=0, class_weight='balanced')
     log_model = log_model.fit(X_train, y_train)
     y_pred = log_model.predict(X_test)
     print("Accuracy Score = ", accuracy_score(y_test, y_pred))
     print("F1-Score = ", f1_score(y_test, y_pred, average='macro'))
     print(confusion_matrix(y_test, y_pred))
```

Report:

Table 2.1 Classification report of Logistic Regression

	Pre	**Rec**	**F1-Sco**	**Supp**
0 (Neg)	0.73	0.77	0.75	1964
1 (Neu)	0.68	0.63	0.65	2225
2 (Pos)	0.76	0.78	0.77	1981

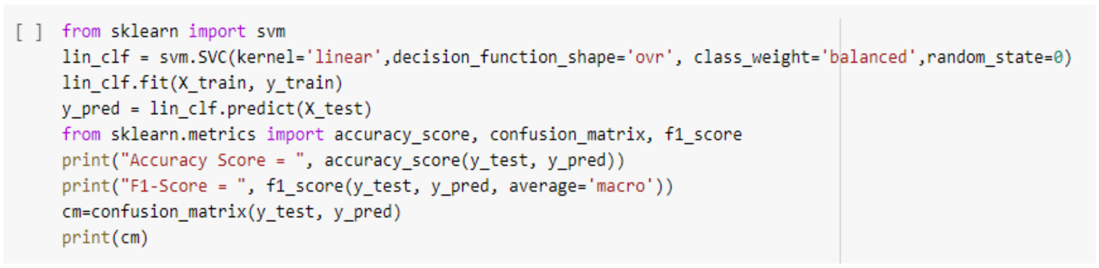
Acc			0.72	6170
Mac avg	0.72	0.73	0.73	6170
Wei avg	0.72	0.72	0.72	6170

F1- Score:- **0.725**

Refer for code: https://github.com/jame27-1998/Sentiment_Analysis_Hinglish

6.2 Sub strategy 2 :Support vector machines:

Figure 5.2 Sub Strategy 2 : Support Vector Machines Model



```
[ ] from sklearn import svm
    lin_clf = svm.SVC(kernel='linear',decision_function_shape='ovr', class_weight='balanced',random_state=0)
    lin_clf.fit(X_train, y_train)
    y_pred = lin_clf.predict(X_test)
    from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
    print("Accuracy Score = ", accuracy_score(y_test, y_pred))
    print("F1-Score = ", f1_score(y_test, y_pred, average='macro'))
    cm=confusion_matrix(y_test, y_pred)
    print(cm)
```

Report:

Table 2.2 Classification report of Support vector Machines

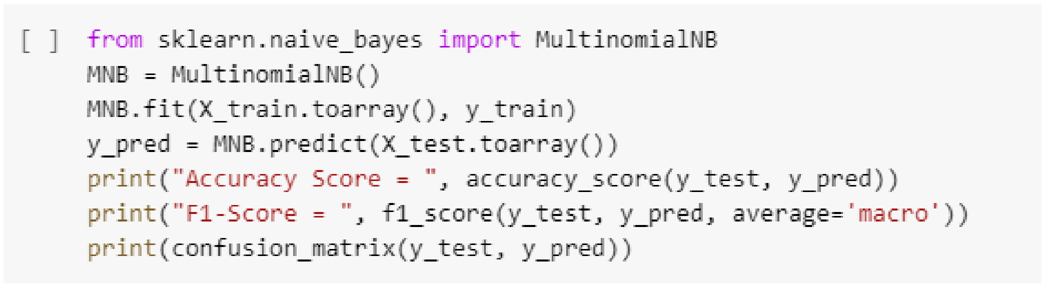
	Pre	Rec	F1-Sco	Supp
0 (Neg)	0.73	0.78	0.75	1964
1 (Neu)	0.68	0.63	0.66	2225
2 (Pos)	0.77	0.78	0.78	1981

Acc			0.73	6170
Mac avg	0.73	0.73	0.73	6170
Wei avg	0.73	0.73	0.73	6170

F1- Score:- **0.728**

6.3 Sub strategy 3 :Multinomial naive bayes :

Figure 5.3 Sub Strategy 3 : MultiNomial Naive Bayes Model



```
[ ] from sklearn.naive_bayes import MultinomialNB
    MNB = MultinomialNB()
    MNB.fit(X_train.toarray(), y_train)
    y_pred = MNB.predict(X_test.toarray())
    print("Accuracy Score = ", accuracy_score(y_test, y_pred))
    print("F1-Score = ", f1_score(y_test, y_pred, average='macro'))
    print(confusion_matrix(y_test, y_pred))
```

Report:

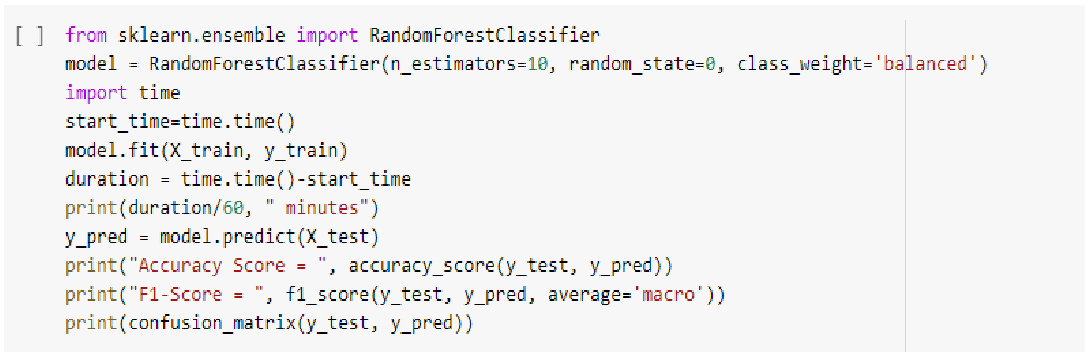
Table 1 3. Classification report of Multinomial Naive bayes

	Prec	Rec	F1-Sco	Supp
0 (Neg)	0.79	0.70	0.74	1964
1 (Neu)	0.63	0.69	0.66	2225
2 (Pos)	0.75	0.76	0.75	1981
Acc			0.71	6170
Mac avg	0.72	0.72	0.72	6170
Wei avg	0.72	0.71	0.72	6170

F1-Score: **0.717**

6.4 Sub strategy4: Random forest classifier:-

Figure 5.4 Sub Strategy 3 : Random Forest Classifier Model



```
[ ] from sklearn.ensemble import RandomForestClassifier
    model = RandomForestClassifier(n_estimators=10, random_state=0, class_weight='balanced')
    import time
    start_time=time.time()
    model.fit(X_train, y_train)
    duration = time.time()-start_time
    print(duration/60, " minutes")
    y_pred = model.predict(X_test)
    print("Accuracy Score = ", accuracy_score(y_test, y_pred))
    print("F1-Score = ", f1_score(y_test, y_pred, average='macro'))
    print(confusion_matrix(y_test, y_pred))
```

Report:

Table 2.4 Classification report of Random Forest Classifier

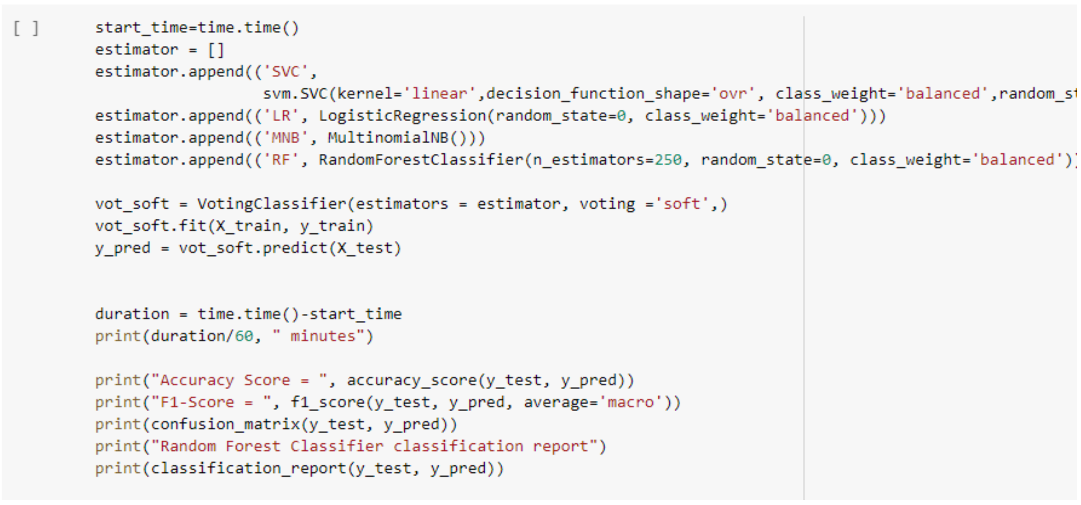
	Prec	Rec	F1-Sco	Supp
0 (Neg)	0.75	0.76	0.76	1964
1 (Neu)	0.63	0.71	0.69	2225
2 (Pos)	0.83	0.75	0.79	1981

Acc			0.74	6170
Mac avg	0.75	0.74	0.75	6170
Wei avg	0.75	0.74	0.74	6170

F1-score: **0.7450**

6.5 Ensembling all strategies : Ensemble Voting Classifier(soft):

Figure 5.5 Ensemble Soft Voting Classifier model



```
[ ]     start_time=time.time()
        estimator = []
        estimator.append(('SVC',
                          svm.SVC(kernel='linear',decision_function_shape='ovr', class_weight='balanced',random_st
        estimator.append(('LR', LogisticRegression(random_state=0, class_weight='balanced')))
        estimator.append(('MNB', MultinomialNB()))
        estimator.append(('RF', RandomForestClassifier(n_estimators=250, random_state=0, class_weight='balanced')

        vot_soft = VotingClassifier(estimators = estimator, voting ='soft',)
        vot_soft.fit(X_train, y_train)
        y_pred = vot_soft.predict(X_test)


        duration = time.time()-start_time
        print(duration/60, " minutes")

        print("Accuracy Score = ", accuracy_score(y_test, y_pred))
        print("F1-Score = ", f1_score(y_test, y_pred, average='macro'))
        print(confusion_matrix(y_test, y_pred))
        print("Random Forest Classifier classification report")
        print(classification_report(y_test, y_pred))
```

Report:

Table 2.4 Classification report of Random Forest Classifier

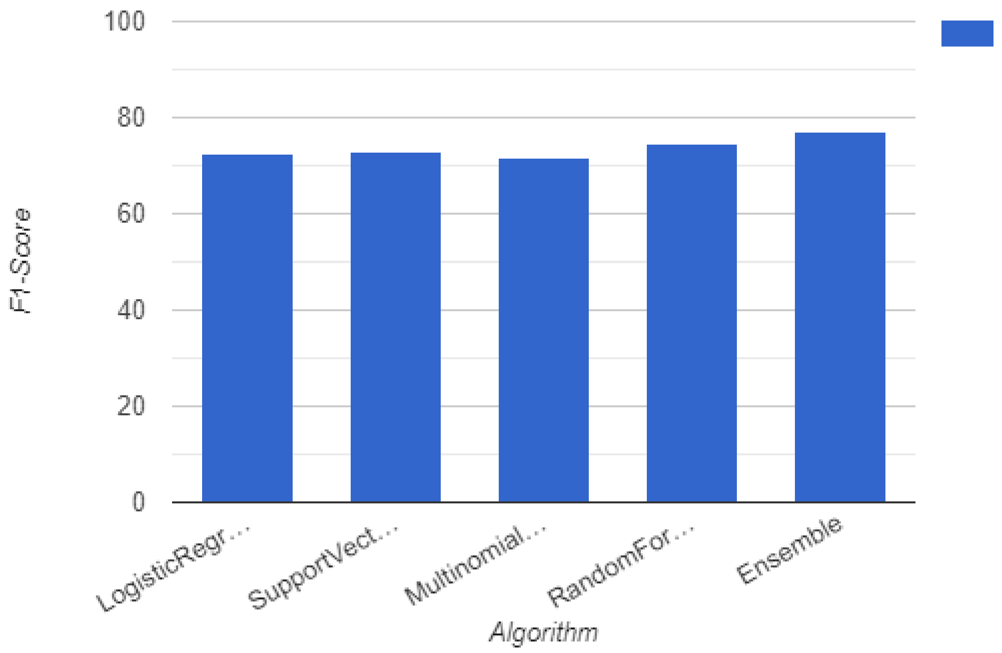
	Prec	**Rec**	**F1-Sco**	**Supp**
0 (Neg)	0.81	0.77	0.79	1964
1 (Neu)	0.71	0.73	0.72	2225
2 (Pos)	0.81	0.81	0.81	1981

Acc			0.77	6170
Mac avg	0.77	0.77	0.77	6170
Wei avg	0.77	0.77	0.77	6170

F1-score: **0.7715**

7. Comparison:

6.1 Comparison between all the models



The results of this Ensemble strategy giving better results compared with the previous work mentioned in the literature survey which is 0.7715 f1 score.The f1-scores of models LR ,SVM, MNB RF and Ensemble are 0.725, 0.728, 0.717, 0.745 and 0.771.The Ensemble strategy gave better results among all algorithms which is **0.7715** and better when we compare with the related past work mentioned in literature survey paper [1] ,paper [3], paper[2] ,paper [4] paper [9] are 0.6907, 0.617, 0.79 ,0.58 ,0.56 . The paper [2] got 0.79 because of the small dataset as I mentioned in the literature survey.

8. Conclusion & Future Work :

Out of all experiments performed among Machine learning algorithms the Ensemble voting (soft) classifier of ML algorithms achieved an F1-score of 0.**7715** but one of the Sub strategy random forest classifiers obtained 0.**747** However this work can be extended to an even better solution if someone got predefined stopwords in hindi and some stemming algorithm for hindi to the conversions of all synonyms to some base word. Here instead of removing the stop words we removed most frequent occurring words greater than 1000.and also can find even better solutions in combinations with different embeddings , one hot encoding, frequency encoding, tf-idf, word2vec,glove along with one gram, bi gram ,tri gram and n gram.can Also improve rectifying the mistakes made by users in english vocabulary or hindi words written in English.Various deep learning algorithms too like MLP,CNN,LSTM,GRU,BERT etc in combinations with different embeddings can also be worked..

9. References:

[1] Singh, G.,. Sentiment Analysis of Code-Mixed Social Media Text (Hinglish). arXiv preprint arXiv:2102.12149.(2021).

[2] Gupta, V.K.,. " Hinglish" Language--Modeling a Messy Code-Mixed Language. arXiv preprint arXiv:1912.13109.(2019).

[3] Baroi, S.J., Singh, N., Das, R. and Singh, T.D.,. NITS-Hinglish-SentiMix at SemEval-2020 Task 9: Sentiment Analysis For Code-Mixed Social Media Text Using an Ensemble Model. arXiv preprint arXiv:2007.12081.3. (2020)

[4] Sproģis, U. and Rikters, M.,. What Can We Learn From Almost a Decade of Food Tweets. arXiv preprint arXiv:2007.05194. (2020)

[5] Bhange, M. and Kasliwal, N., . HinglishNLP: Fine-tuned Language Models for Hinglish Sentiment Detection. arXiv preprint arXiv:2008.09820. (2020)

[6] Singh, G., . Decision Tree J48 at SemEval-2020 Task 9: Sentiment Analysis for Code-Mixed Social Media Text (Hinglish). arXiv preprint arXiv:2008.11398. (2020)

[7] Singh, P. and Lefever, E., . LT3 at SemEval-2020 Task 9: Cross-lingual embeddings for sentiment analysis of hinglish social media text. arXiv preprint arXiv:2010.11019. (2020)

[8] Mishra, P., Danda, P. and Dhakras, P., . Code-mixed sentiment analysis using machine learning and neural network approaches. arXiv preprint arXiv:1808.03299. (2018)

[9] Patra, B.G., Das, D. and Das, A.,. Sentiment Analysis of Code-Mixed Indian Languages. ICON..(2017)

[10] Singh, P. and Lefever, E.,, May. Sentiment analysis for hinglish code-mixed tweets by means of cross-lingual word embeddings. In Proceedings of the The 4th Workshop on Computational Approaches to Code Switching (pp. 45-51) (2020)