Sentiment Analysis to investigate human emotions by analysing News Headlines.

IT has also been flooded with immense amounts of data, which is being published every minute of every day, by millions of users, in the shape of comments, blogs, news sharing through blogs, social media micro-blogging websites and many more.

We have attempted to study sentiment analysis of news articles using various machine learning algorithms like Logistic Regression, Naive Bayes, KNN and Deep Learning.

This type of Sentiment analysis helps us to understand public opinion on any of the issues and, also helps us in gaining insights from customer feedback.

It also helps us understand how the public responds to the misinformation?

In this project, we are basically trying to distinguish between positive and negative news headlines.

Where is the data?

Imported dataset from Kaggle; "Millions News Headlines" Dataset

Dataset contains two important columns named publish_date and he adline_text that we have widely used.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1244184 entries, 0 to 1244183
Data columns (total 2 columns):
 #   Column         Non-Null Count    Dtype 
---  ------         --------------    ----- 
 0   publish_date   1244184 non-null  int64 
 1   headline_text  1244184 non-null  object
dtypes: int64(1), object(1)
memory usage: 19.0+ MB
```

Algorithm

- For automatic sentiment detection of news articles, we have used Dictionary based approach which uses Bag of Word technique for text mining.
- To build the polarity dictionary, we need two types of words collection, positive words and negative words. Then we can match the article's words against both these words list and count numbers of words appears in both the dictionaries and calculate the score of that document.
- We created the polarity words dictionary using general words with positive and negative polarity.
- For the news article, we are considering the string which contains headline.
- Naïve Bayes classification and Logistic Regression algorithms performs good in text classification. So, we are considering all two algorithms to classify the text and check each algorithm's accuracy. Also, results can be compared on basis of accuracy, precision, recall and other model evaluation methods.
- Here we have evaluated two models using machine learning and deep learning methods to build a binary classifier that can distinguish between positive and negative news headlines.
- Supervised learning through Scikit and NLP packages Logistic Regression, Naïve Bayes and KNN.
- Deep learning using TensorFlow and Keras frame works.

Approach Deep Learning Model

By partitioning the dataset into 8:2 ratio of train and test split. The models predict on new instances with fresh headlines , either positive or negative.

In the second approach, we used **Keras** to load and train the Sequential model (neural network), which is suitable for a simple stack of layers with exactly one input tensor and one output tensor for each layer, and **TensorFlow** to pre-process and pad the data using the tokenizer class.

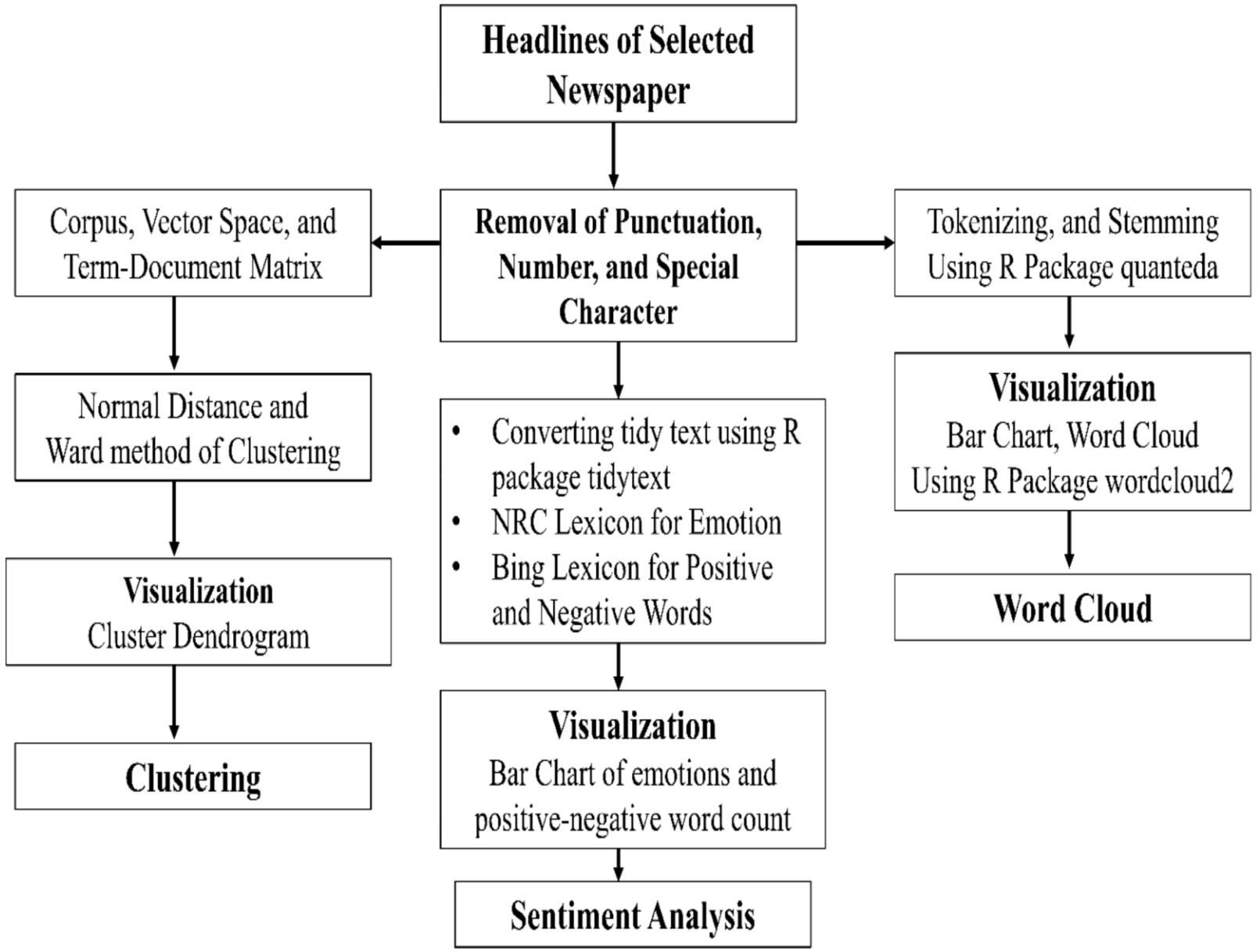
Created word encodings to word tokenizer from **tensorflow.keras** and sequences by using **texs_to_sequences** instance, and then padding these sequences to make it of equal length using the **pad_sequences** instance.

Built the model using a vocab size, embedding dimension, and input length embedding layer. Added layers include a dense layer ReLU that instructs the model to categorize events into two groups based on whether they are positive or negative, and a final sigmoid layer that provides probabilities between 0 and 1. **Hyperparameters** are also included within each layer to increase model performance.

The constructed neural network model with 100 running epochs has a very high accuracy of **96%**, lowering validation loss and increasing validation accuracy, which ensures a potent predictive performance and a low danger of generalization (overfitting) mistake.

Trained data using **Logistic Regression, Naïve Bayes and K-Nearest Neighbour** and achieved **90.92%, 88.86% and 77.10%** accuracy respectively.

Flow Chart


Headlines of Selected Newspaper
Removal of Punctuation, Number, and Special Character
Corpus, Vector Space, and Term-Document Matrix
Normal Distance and Ward method of Clustering
Visualization
Cluster Dendrogram
Clustering
Converting tidy text using R package tidytext
NRC Lexicon for Emotion
Bing Lexicon for Positive and Negative Words
Visualization
Bar Chart of emotions and positive-negative word count
Sentiment Analysis
Tokenizing, and Stemming Using R Package quanteda
Visualization
Bar Chart, Word Cloud Using R Package wordcloud2
Word Cloud

Comparing ML Models

Classification Matrix(Logistic Regression)

```
print(classification_report(y_test, y_pred_log))
```

	precision	recall	f1-score	support
0	0.86	0.89	0.87	3542
1	0.94	0.92	0.93	6384
accuracy			0.91	9926
macro avg	0.90	0.90	0.90	9926
weighted avg	0.91	0.91	0.91	9926

Classification Matrix(Naive Bayes):

```
y_pred_cnb = cnb.predict(X_test)
print(classification_report(y_test, y_pred_cnb))
```

	precision	recall	f1-score	support
0	0.85	0.84	0.84	3542
1	0.91	0.92	0.91	6384
accuracy			0.89	9926
macro avg	0.88	0.88	0.88	9926
weighted avg	0.89	0.89	0.89	9926

Confusion Matrix(KNN)

```
y_pred_knn = modelknn.predict(X_test)
print(classification_report(y_test, y_pred_knn))
```

	precision	recall	f1-score	support
0	0.67	0.71	0.69	3542
1	0.83	0.81	0.82	6384
accuracy			0.77	9926
macro avg	0.75	0.76	0.75	9926
weighted avg	0.77	0.77	0.77	9926

Accuracy of Deep Learning Model

Classification Matrix and f1 score Deep Learning

```
[15]: y_pred_deep = model.predict(testing_padded)
for i in range(y_pred_deep.shape[0]):
 if y_pred_deep[i]>0.5:
    y_pred_deep[i] = 1
    # print("1")
 else:
    # print("0")
    y_pred_deep[i] = 0
# print(y_pred_deep)
# print(testing_labels)
print(classification_report(testing_labels, y_pred_deep))

from sklearn.metrics import *
deepf1=f1_score(testing_labels, y_pred_deep)
print(deepf1)
```

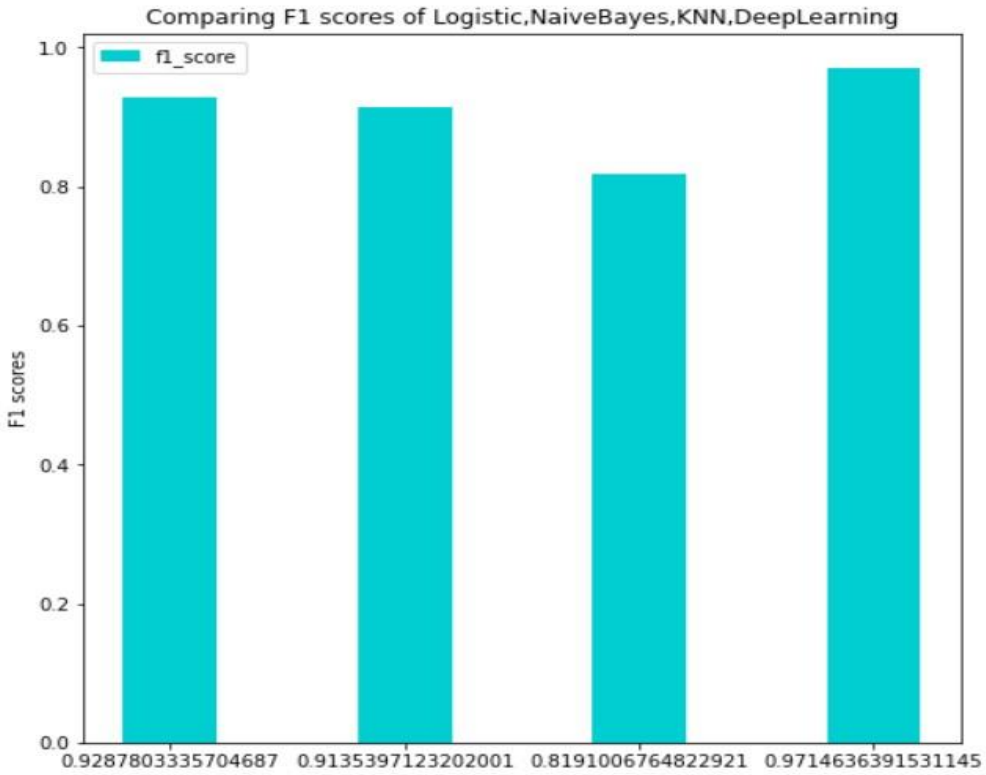
```
              precision    recall  f1-score   support

           0       0.97      0.92      0.95      3651
           1       0.96      0.99      0.97      6425

    accuracy                           0.96     10076
   macro avg       0.97      0.95      0.96     10076
weighted avg       0.96      0.96      0.96     10076

0.9714636391531145
```

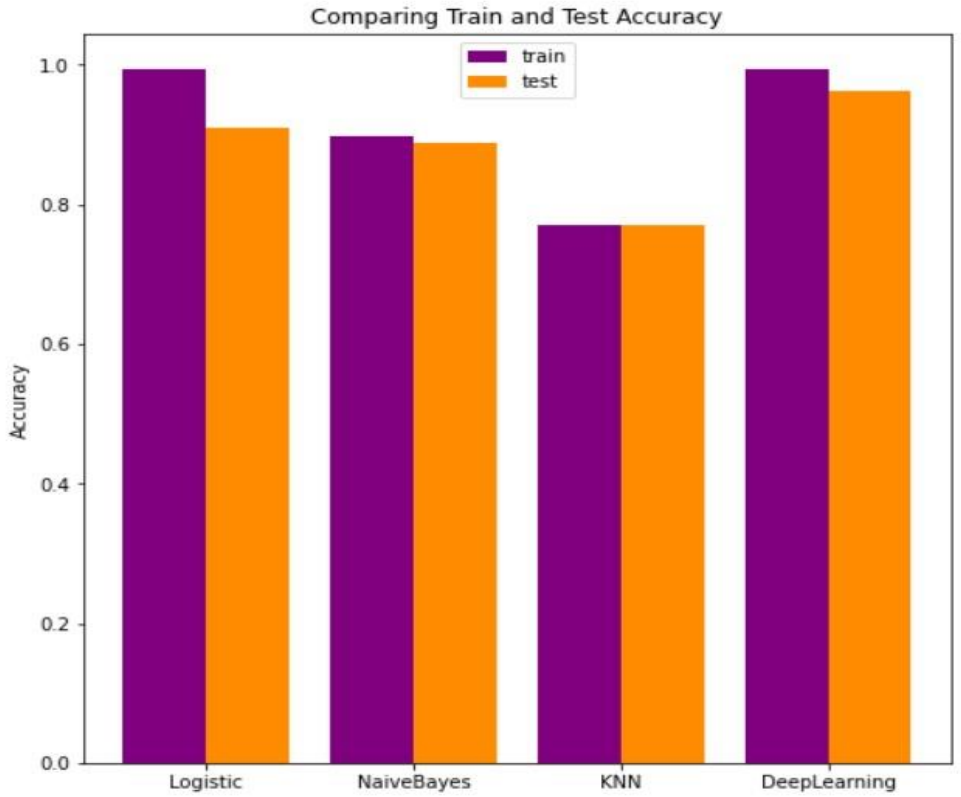
Comparing F1 Score




```
]: f1score=[logf1,cnbf1,knnf1,deepf1]
   print(f1score)
```

```
[0.9287803335704687, 0.9135397123202001, 0.8191006764822921, 0.9714636391531145]
```

Comparing Test and Train Accuracy



comparing test and train accuracy of all methods

```
Train_acc=[train_acc_log,train_acc_cnb,train_acc_knn,eval_result_train[1]]
Test_acc=[test_acc_log,test_acc_cnb,test_acc_knn,eval_result_test[1]]
print(Train_acc)
print(Test_acc)
```

```
[0.9940083519941899, 0.8970525933547177, 0.7692912909277976, 0.9947500228881836]
[0.9092282893411243, 0.8885754583921015, 0.7710054402579085, 0.9630805850028992]
```

Conclusion

F1 Score which is greater than 0.7 is considered as a fairly to good model.

In that case out of all the models in Supervised learning Logistic regression is considered as a good model and Deep learning as fairly good model.

Of all the algorithms, KNN has the least impact on sentiment analysis

	Accuracy	F1 Score
Logistic Regression	90.92%	0.92
Naïve Bayes	88.86%	0.91
KNN	77.10%	0.81
Deep Learning	99.4%	0.97

Thank you