**Sentiment Analysis**

**Introduction**

Sentiment Analysis is a method to analyze or identify the human feelings from the text the said or typed.Generally we have the three types of sentiments called Negative, Neutral and Positive.Here We are more look into positive and negative sentiment from the text. Because, Neutral sentiment means like no particular sentiment describes from the text. Also sentiment analysis we applied on text data.It is more depend on the human beings. If they are happy, then they will give positive other it will be Neutral or negative. And we can find sentiment or emotions from their face images or expressions. But we can not say what they think in their mind. It is more challenging to find the true sentiment from the face.

Along with this, we have different use cases with sentiment analysis in business problems like User experience about a particular product, customer relationship service, political polls of the people etc. These are the general and most commonly used use cases with sentiment analysis. This is a method of Natural language processing in machine learning. Today, we have transformers to find the sentiments I better way fro the text. But , here we are using legacy algorithms of machine learning for finding the sentiment. All the methods and algorithms we can use in our problem , but its depend on our data.Here, we have only 50 records for finding the sentiment. And it is an unstructured data, so , we don’t have labels. So we should apply unsupervised methods on top of the data.

Here we are finding the sentiment for news headlines. We have the dataset with news headlines along with description. After analyzing the headlines we need to predict the sentiment of the given headline like it is negative,neutral or positive.

**Technologies And Frameworks**

As part of programming , we used python, we used following packages from python to perform the sentiment analysis.

Pandas,re

Sci-kit Learn

NLTK

Text Blob

Word Cloud

Seaborn

Matplotlib

**Pandas** is a library for structured data analysis like csv, tabular data etc. It also supports 3dimensional data as panel. Here we are using two dimensional data called data frame. We read our ‘dataset.csv file as a data frame in python. And we can do the exploratory data analysis with that data frame. **Re** is the regular expression package in python. It allows us to clean the text as per the pattern or expression we given to the regular expression functions.

**Sci-kit Learn** is the general library for machine learning algorithms implementation. We used clustering and Multinomial Naive bayes classifier for grouping and predicting the sentiment of a text. We have different categories of naive bayes classifiers like Gaussian NB, Multinomial NB, BernoulliNB etc. While we are using text data, we can use Multinomial NB for our task. For clustering , we have used **Kmeans** and **MinibatchKmeans** algorithms for grouping the text according to their syntactic behaviour from the document. And , we need to vectorize our text before put in to model. For that we have different vectorization methods like bagofwords,n grams and tf-idf vectorizer etc. We implemented bagofwords and tf-idf vectorizer for this problem. Also, we need to apply Principal component analysis for dimensionality reduction and visualize our clusters in graph. For that, we can use scikit learn **PCA** for performing dimensionality reduction.

**NLTK** is a natural language processing tool kit for NLP task with python.With NLTK library, we can used to text preprocessing, tokenization, Stop words extraction, stemming and lemmatization etc. As part of analysis, we can create sentiment of the text using NLTK pretrained sentiment analyzer. From that, we understood most of the sentences are in neutral sentiment. After that we applied clustering and classification sequentially to explore our data more and we concluded the same.

**Text Blob** is a python library used for NLP tasks like part-of -speech tagging, sentiment analysis etc. It will give the sentiment polarity of the text given.It returns real values of positive and negative sentiment. If it is negative value, then we can say the sentiment is negative and if the polarity is positive, the sentiment will be positive and if it is zero polarity, that will be neutral.

**Word cloud**, Seabornand **Matrplotlib** are the libraries for visualizing the text in a better way. We can explore and will have more understanding about our data.

**Experimentation Setup**

**Steps:**

1. Upload the data To Google drive
2. Create new notebook in co lab and mount the google drive with notebook.[[Colab](https://colab.research.google.com/)]
3. Install the required libraries with pip

!pip install wordcloud

1. Run the code cell one by one and the saved result will save into the content folder of notebook itself. And the output will show after the each cell.

**Working Of The Proposed Model**

We implemented two machine learning model on top of our unstructured data. We tried a hybrid approach for sentiment analysis. For every data problem, First thing is we need to understand our data and do some exploratory data analysis. Here we have a tabular datasets which contains 5 columns. But we are only taking the headline and description columns for our analysis. Because , Our problem is to find the sentiment of the news headlines. Basically we are doing the following approaches.

* Clustering
* Classification

**Clustering**

Clustering is a type of unsupervised method to group the data according to the common behaviour among the dataset. Here we are grouping in to three clusters according to our sentiment, that is Negative, neutral and positive clusters. Before doing the clustering, we find the sentiment using NLTK sentiment analyzer to get the sentiment distribution in raw data. From that we got that all texts are tagged to neutral sentiment.

After that, We checked missing values and applied text per-processing techniques such as removing links, special characters , numbers, and stop words like the, a etc from the text using nltk and python regular expressions.

Then tokenize the text and change the letters from upper case to lower case and joined back to get full text after processing the text.

Text Pre-Processing for Clustering

* Stop word Removal
* Removing special characters ,links, unwanted space ,numbers from text
* Tokenization Split long text to words
* Lowercase letters

Now, we have the processed data for clustering. Then we need to vectorize our text data using Tf-idf vectorizer. The process of transforming text in to a vector called vectorization. For machines , only understand numbers right? So we need to to convert text to vector representation to train the model. For that we have multiple methods like Bag of words, TF-IDF, Word embedding techniques, BERT etc.

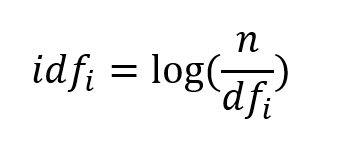
Now we are using Tf-idf and BOG for vectorization on our data.

**Bag of Words**

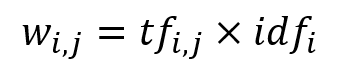
In this vectorization, most important thing is it will count the words from each document and create a document word matrix for the whole text in the corpus. The grammar and the order or position of the word not consider for vectorization. The number of times a word present in the document that is the main concern about bag of words.In scikit learn we have count vectorizer class for performing bag of words. We can also pass the n-gram range according to our data, by default it is 1. that means it consider single words only. N-gram is a n word sequence in a sentence. For example, social distance is a combination of social and distance . So the combination of social and distance gives different result compared to social and distance separately .

**TF-IDF Vectorizer**

In TF-IDF , we are combining two methods called term frequency and inverse document frequency. The term frequency is the number of occurrence of a term in a document. It also create a matrix in between document and term. And document frequency is the number of document containing specific term.It will give an idea common words in the document. Then Inverse document frequency is the just inverse of document frequency.



And finally, it gives the multiplication of term frequency matrix and inverse document frequency.

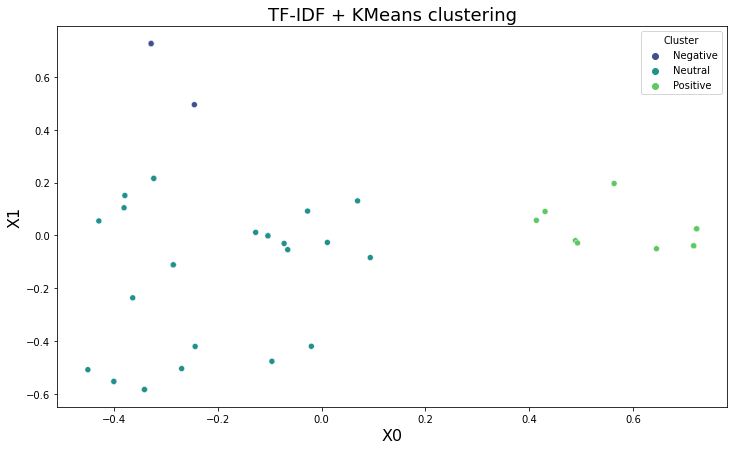


After all, now we have the data in vectorized array.in tf-idf , it does not capture the semantic features, only syntactic features. It depends on the weight of the term in the all document.So, some time it will gives the rare term has high weight and common word less weight. And It neglects the sequence of words in the text as well.

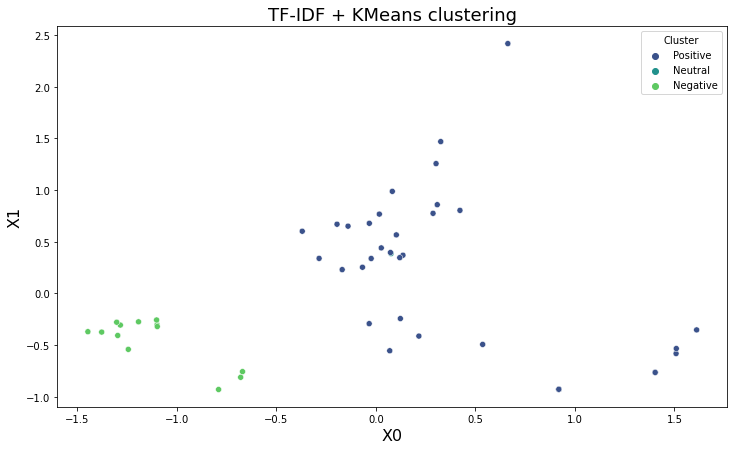
**Principal Component Analysis**

Now we have the vectorized data. Then we are applying dimensionality reduction technique called Principal component analysis on top of that to reduce the dimension in feature space. Here we are using for cluster data visualization. We are creating two component which will represent our data in the feature space.

Kmeans Cluster Visualization

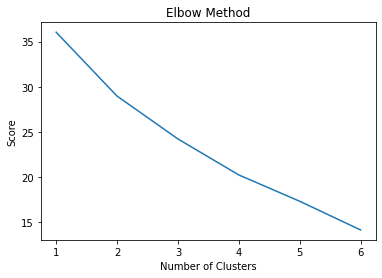


Mini Batch Kmeans Cluster Visualization



Before doing the Kmeans and Mini batch Kmeans algorithms, we need to find the optimal number of clusters using elbow method. Actually, we need three clusters like Negative, Neutral and Positive. And also which number is most suitable for our data.

Kmeans Elbow Curve



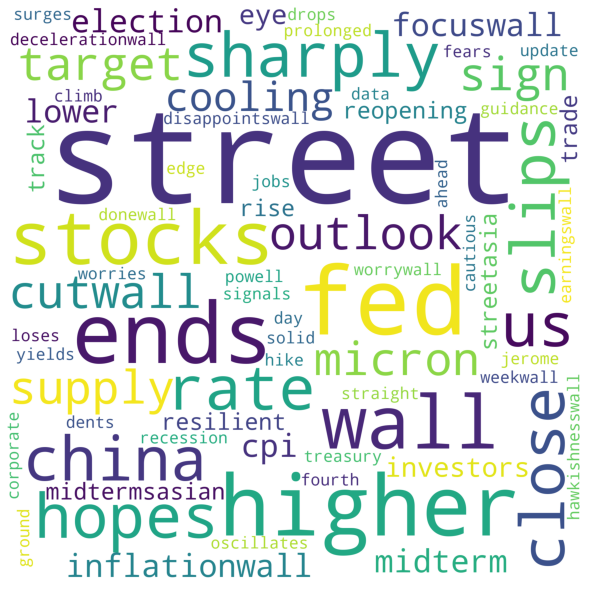
From the graph, the optimal value for the number of cluster is 2.But we will give 3 here and will train the Kmeans algorithm. And we defined random state=42. It will take data points randomly when we doing Kmeans.

After Kmeans clustering we are mapping cluster number to cluster name like zero mapped to neutral, 1 mapped to positive and 2 mapped to negative. So we have 31 data points in neutral sentiment,13 for positive and 6 for negative. According to cluster keywords, we determines the cluster name which is positive,negative and neutral.Actually we have the problem of less amount of data.Because, we have no keywords to match with positive or negative sentiment. All words are in neutral.

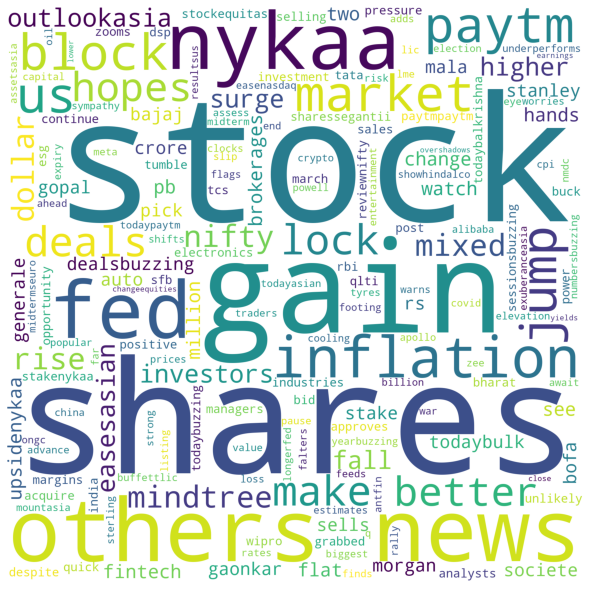
Negative Word cloud



Positive Word cloud



Neutral Word Cloud



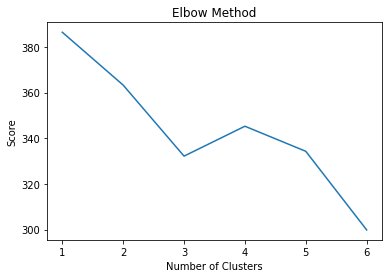
After the clustering we saved the result as csv file and this data passed to classification.

And we tried Mini batch kmeans with same data and we got the result with some level of variation from kmeans. For mini batch kmeans , we used bag of words as vectrorization method.

In mini batch, the input data passed as batches to the model. That is, it takes rand small batches of data for the model. So it is faster processing than normal Kmeans algorithm.

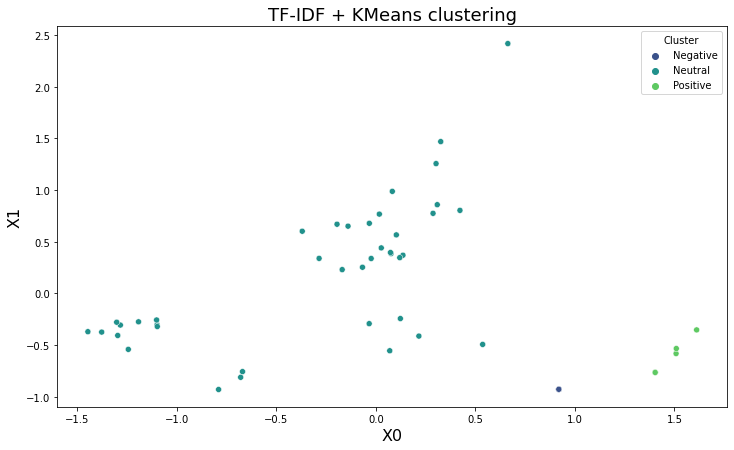
Visualization From Mini Batch Kmeans algorithm.

Elbow Curve:



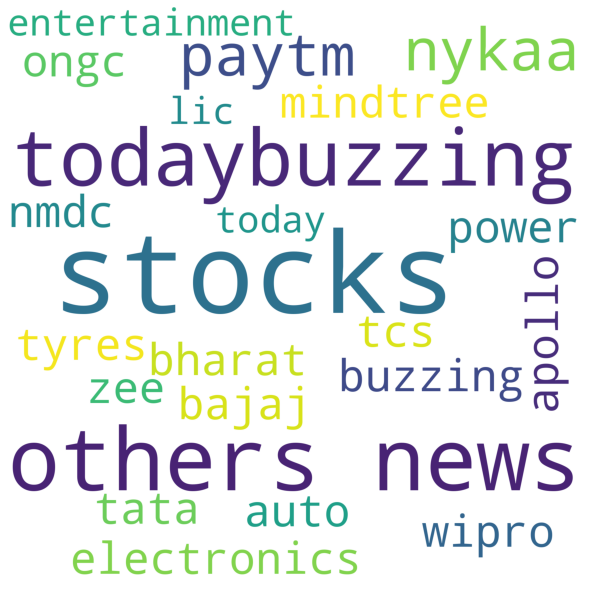
Here, we can select the value three for the number of clusters.

Pca Visualization:



After clustering , we have 40 samples for neutral cluster and five samples for each positive and negative samples.

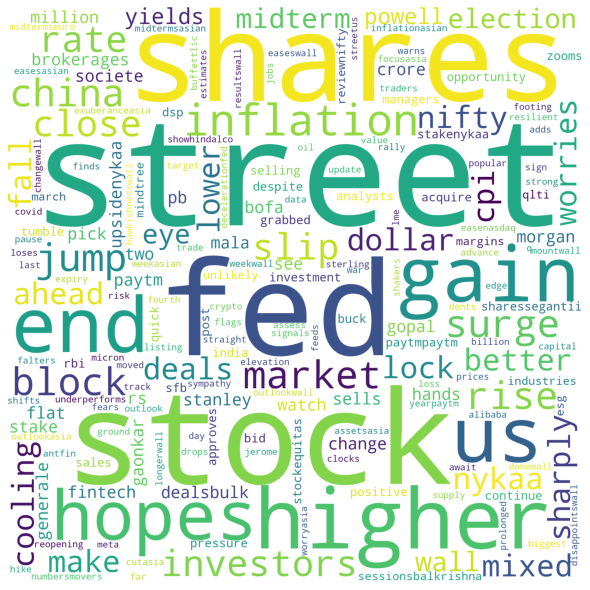
Positive Word Cloud:



Negative Word Cloud:



Neutral Words cloud:



After the clustering the result saved in to disk for further study.

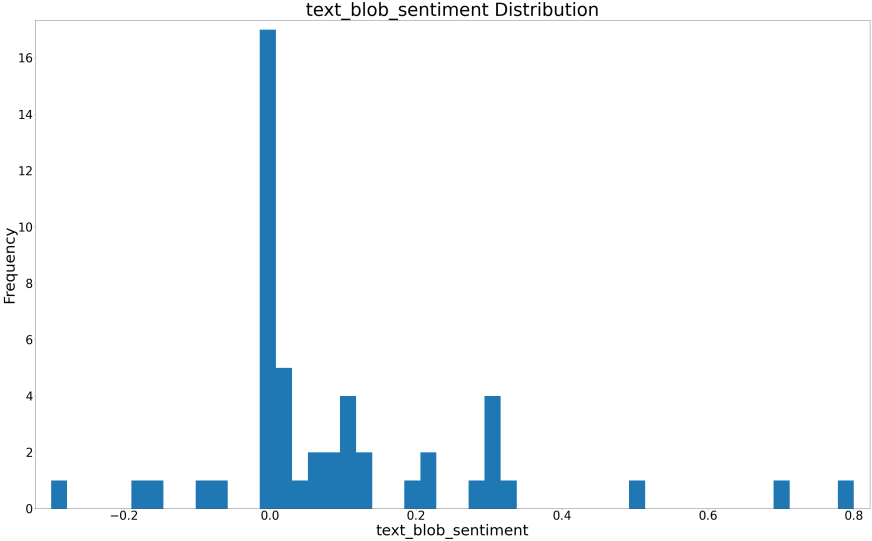
**Classification**

After Clustering, we are doing classification on top of the clustered result. We are doing supervised learning with clustered sentiments as labels. We are using naive bayes classifier for classifying the sentiment with given labeled data.Basically, we are doing Multinomial Naive bayes classification, because our data is text format. And one difference is here we are using news description column rather than headlines. Because we need more granularity about the text for predicting the sentiment.

**Data Pre-Processing**

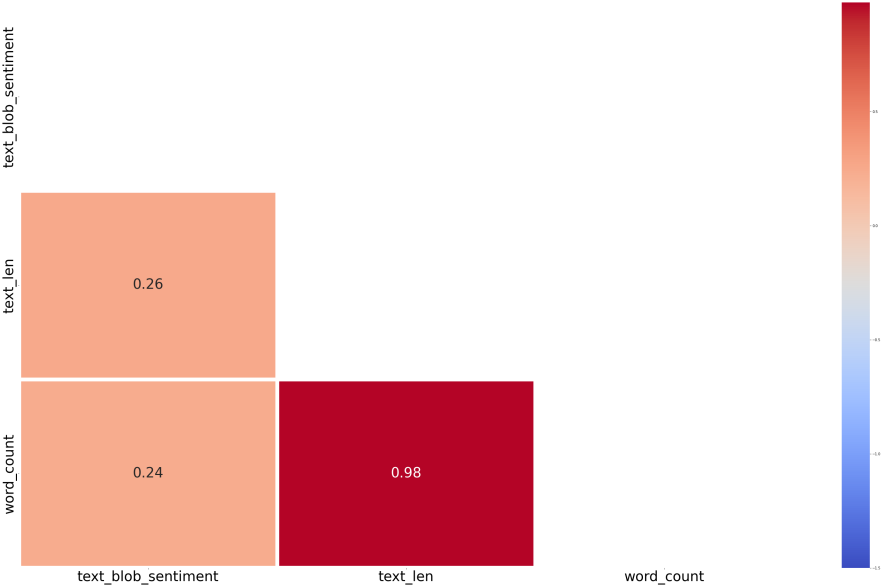
Also, we are doing the preprocessing techniques such as removing special characters, stop words from nltk tool, numbers, whit space from the text. Other than this we are applying lemmatization on our data. We are using word net lemmatizer from nltk library. Lemmatization is the process of converting different forms of a word in to one form with keeping the context. And last we converted text into lower case.

Then, we checked sentiment polarity of our data with Text blob. We got the values from negative to positive.



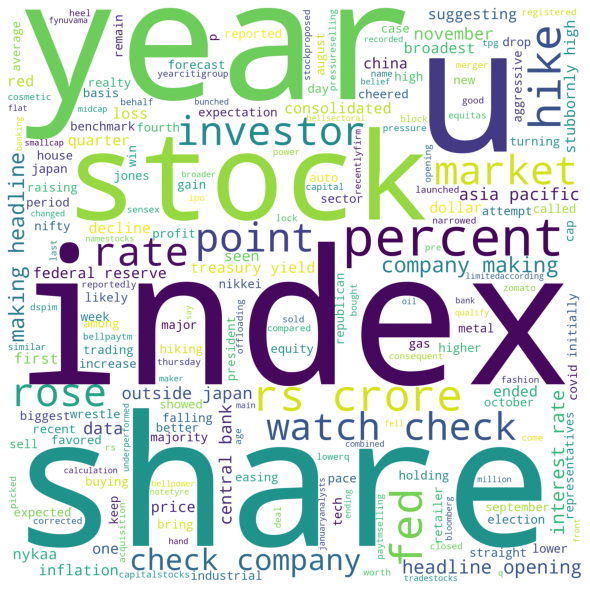
From the graph, we can say , we have more number of words indicate neutral sentiment.

After that, we calculated word count for each row and text length in each description with pandas. Maximum word count from the data is 30 and minimum word count is 8. And maximum length of the string is 195 and minimum length is 51. And we have no missing values from the data.



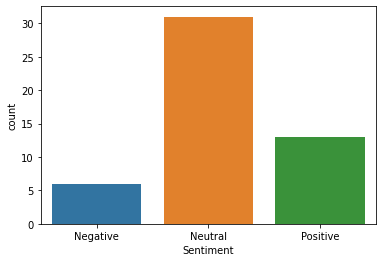
From the correlation plot, we have not much correlation between the numeric features.So we are not using this features for train the model.

Words in the news description :



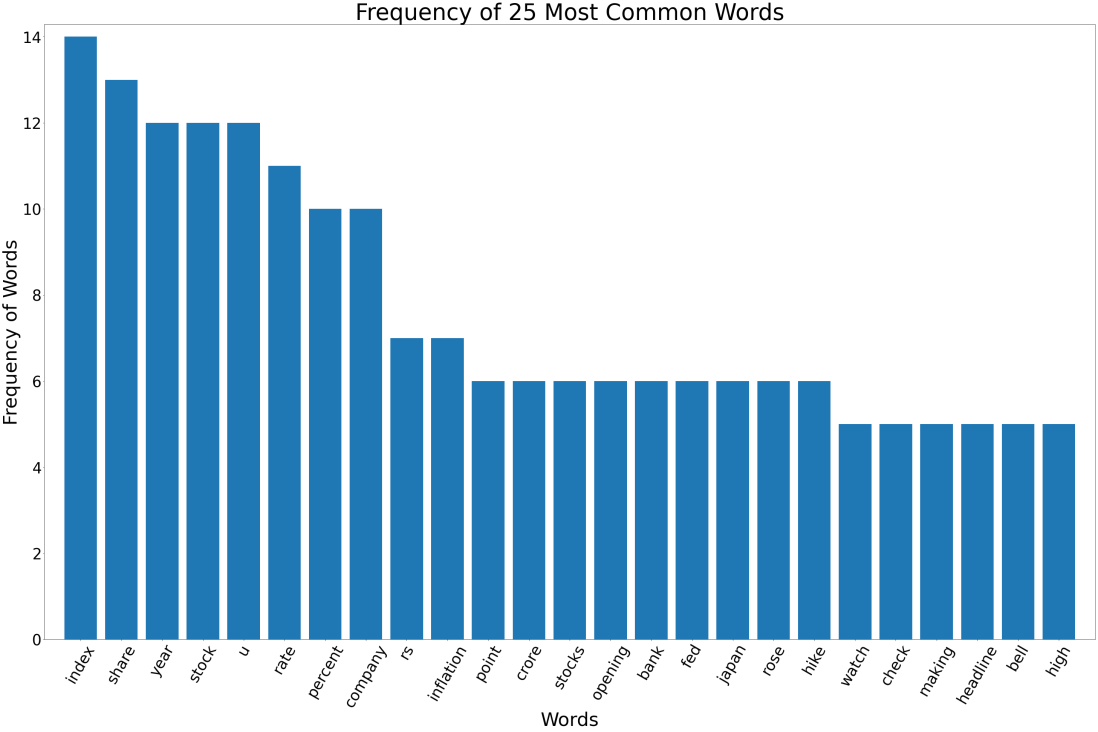
Also, as part of exploratory data analysis we have plotted the count plot and histogram with our data.

Sentiment Distribution

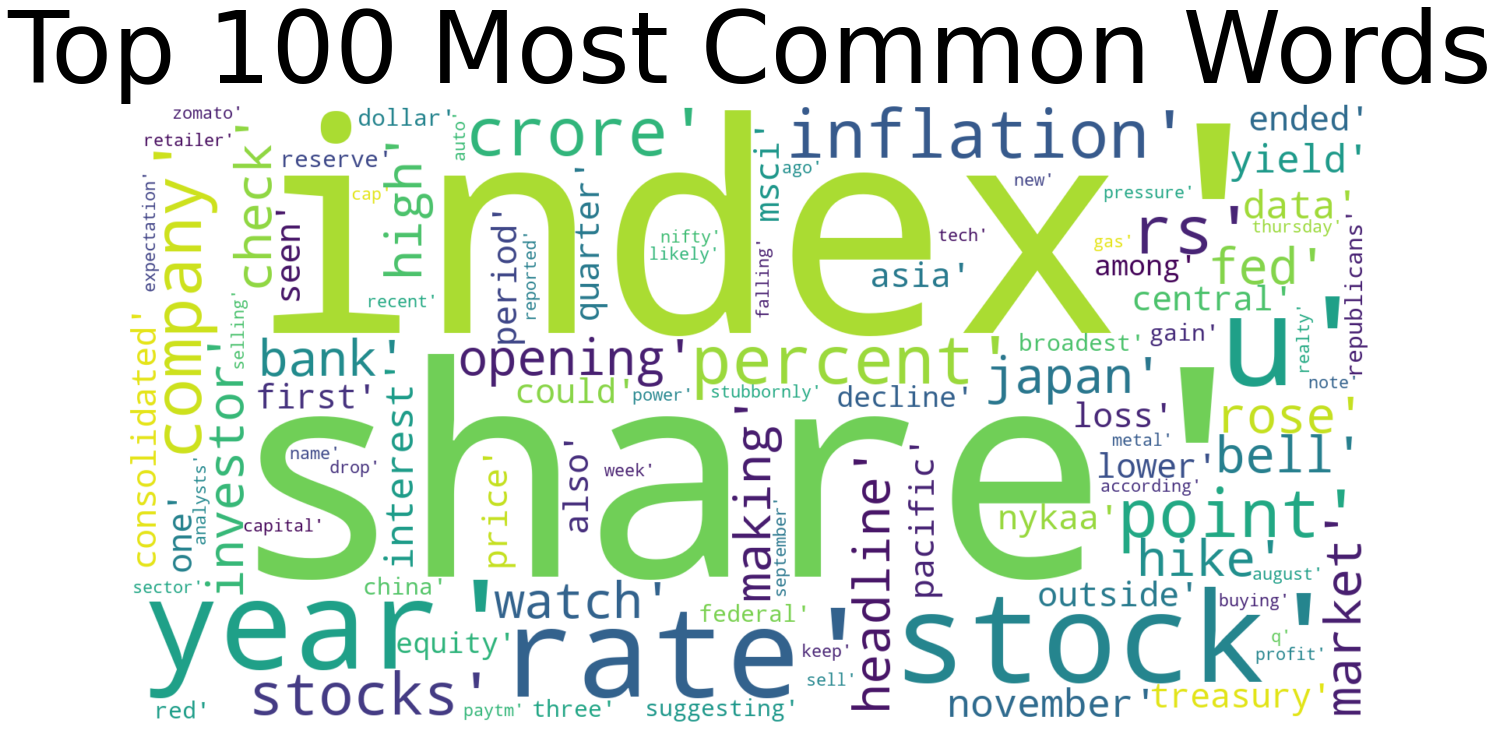


From the plot, we have more sentiment on Neutral and then positive.And also our data is not balanced here. We will do some approach to solve the data imbalance in our data in future.

Word Frequency Plot:



Here, we plotted twenty five most common words in our dataset. We have index, share, year, stock are the most frequent words in the data. So these words do not give positive or negative sentiment. So our analysis is on the right path. Then also we created a word cloud for the 100 common words in the dataset.



**Data Preparation For Training The Model**

For training the mode, we are using the two columns. One is cleaned news description and sentiment or label columns. Then we need to encode the target sentiment column to numbers. Here we have 3 categories in target column, so we can do the label encoding and one-hot encoding with our variable. Because we have only few records and few categories in target variable sentiment. And also converted target column type into category type.

Then, we splitted data into features and labels and after that we splitted into train and test for the training. Before doing the train and test split we applied TF-IDF vectorization with our feature column cleaned news description. Then we splitted data into train and test. Ten percent data for test and ninty percent data for train. Also we selected a random split for train and test. And we applied stratified on target column to get the equal samples for each random split.

After the completion of feature engineering we can do the training.

**Naive Bayes Classifier Training**

The algorithm is called naive bayes, because it takes naive assumption , that is each feature is independent from other features. But that is not possible in real life scenario. And it also using conditional probability by Baye’s theorem. This theorem states that the probability of an event , based on the prior knowledge of conditions that might be related to the particular event. Also, naive bayes well suited for small data set like our data, Here we are using Multinomial Naivebayes algorithm for classification. We are not changing any parameter for Multinomial NB classifier. After training we saved our predicted values and actual values in a data frame for evaluating the result. We can tune the parameter alpha as smoothening in training if needed. We have two options like Laplace or lid stone smoothening. The default value is 1.0. That means smoothening will happen in training.And we have class prior and fit prior as well as hyper parameters. We have no class prior probabilities and we set fit\_prior is true.

After training , we need to evaluate our model predictions with actual test data and with some performance metrices like confusion matrix and f1 score. Because of we are implement classification algorithm on top of text data. So we need to evaluate our predicted sentiment with performance metrices.

**Evaluation**

For classification, we have different evaluation metrices like accuracy, precision,recall and f1-score etc. And also we can plot confusion matrix for the same. Here we have train and test data for evaluating the model. In our scenario we can consider Accuracy and f1-score for the performance metrics of the model.Evaluation metrics will give the basic idea about how well performing our model with our train and test data. Accuracy is not a good evaluation metric for imbalanced data. Because it will give the high accuracy even though the model performed poorly on one class. So we need to validate our performance metrics with multiple measurements like f1-score or precision etc.

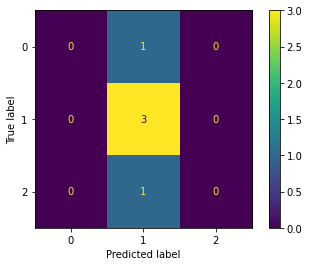
Accuracy will give the overall performance status of the model. It consider all true positives, true negatives, false positives and false negatives. And F1-score will give the balanced precision and recall. It calculates the harmonic mean of the precision and recall.

IMG_256

Here, our data is not well balanced. So we can consider both F1-score and accuracy as evaluation metrics.

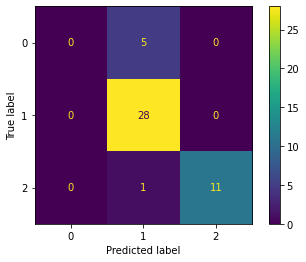
Confusion metrics : Test Data

1. Negative
2. Neutral
3. Positive



From the confusion metrics, we can say, we have no positives and negative sentiments correctly predicted our model. Only few neutral sentiments predicted correctly. This is showing our model is not working well with our data. But we have the 60% accuracy. Which is not give any sense with our confusion metrics right? And when we consider F1-score we have only 25% for the three classes. This make sense with our confusion matrix.

Confusion Matrix: Train data



From train data, we have

Accuracy: 87%

F1-score: 62%

**Conclusion**

From these analysis, we have some points and recommendations to develop a good sentiment analysis model. Like, Our data is very less for training a machine learning model and data have no insight full and variant information to get the exact sentiment. We have more neutral headlines in our dataset. That is not much needed for sentiment analysis

And we can improve our result and analysis when we have the following recommendations in our mind.

* We need at least 1000 records per class for training data
* Try to collect meaning full and insightful data for sentiment analysis. Because model should have differentiate the text for each sentiment like positive, negative and neutral.
* If we have more data, we can also try to solve imbalance problem in our data
* We can improve and do some more data per-processing task for getting better understanding about our data.
* We can use tree based an ensemble based methods, if we have more variance in our features.
* We can apply deep learning techniques such as RNN or LSTM model if we have more data points or records in our dataset.

For a real world scenario, we will have different challenges and problems. But we can solve that if we have the meaningful data which is relevant to r business.

I am concluding here as part of my analysis. I understood machine learning we can not apply every problems in real world until or unless we have good amount of qualitative data.

**References**

* <https://towardsdatascience.com/nlp-part-3-exploratory-data-analysis-of-text-data-1caa8ab3f79d>
* <https://medium.com/analytics-vidhya/evaluation-metrics-for-classification-models-e2f0d8009d69>
* <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html>
* <https://medium.com/analytics-vidhya/a-complete-guide-on-dimensionality-reduction-62d9698013d2>