

**TWITTER SENTIMENT ANALYSIS**

**PROJECT REPORT**

Submitted for CAL in B. Tech Object Oriented Software Development

CSE4028

Faculty: - Prof. Sharath Kumar Jagannathan

Slot-G2

(SCHOOL OF COMPUTER SCIENCE AND ENGINEERING)

GROUP MEMBERS: -

Piyush Bamel 16BCE1221

Kaustubh Nagar 16BCE1338

**CERTIFICATE**

This is to certify that the project **“Twitter Sentiment Analysis”** submitted under course Object **Oriented Software Development CSE4028** to **Prof. Sharath Kumar Jagannathan** is completed by our group comprising of **Piyush Bamel (16BCE1221) and Kaustubh Nagar (16BCE1338).**

It was under the able guidance of our teacher Prof. Sharath Kumar Jagannathan.

Place : Chennai

Signature of Students :

Kaustubh Nagar (16BCE1338)

Piyush Bamel (16BCE1221)

Signature of faculty-in-charge :

Prof. Sharath Kumar Jagannathan

**ACKNOWKEDGEMENT**

We would like to express our special thanks of gratitude to our Object Oriented Software Development faculty, Prof. Sharath Kumar Jagannathan, who gave us the golden opportunity to do this wonderful project on the topic “Twitter Sentiment Analysisi”, which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful to you. Secondly, we would also like to thank our group members in finalizing this project within the limited time frame.

**ABSTRACT**

Sentiment analysis is the classification of the polarity of a given text in a document, sentence or a phrase. The main goal is to determine whether the expressed opinion in the text is positive, negative or neutral. Sentiment analysis would help a lot in understanding the perspective of people about a personality or an event. It can be of use in business fields as they can know views of common man on their products and services which will help them in making business strategies. Knowing sentiment of the mass is also important for political parties so that they can know whether people will support their various programmes or not. We are doing sentiment analysis on live tweets because twitter is one of the most microblogging social website with millions of active users.

**INTRODUCTION**

Sentiment is an attitude, thought, or judgment prompted by feeling. Sentiment analysis, which is also known as opinion mining, studies people’s sentiments towards certain entities. Internet is a resourceful place with respect to sentiment information. From a user’s perspective, people are able to post their own content through various social media, such as forums, micro-blogs, or online social networking sites. From a researcher’s perspective, many social media sites release their application programming interfaces (APIs), prompting data collection and analysis by researchers and developers. For instance, Twitter currently has three different versions of APIs available, namely the REST API, the Search API, and the Streaming API. With the REST API, developers are able to gather status data and user information; the Search API allows developers to query specific Twitter content, whereas the Streaming API is able to collect Twitter content in real time.  A ground truth is more like a tag of a certain opinion, indicating whether the opinion is positive, negative, or neutral. The Stanford Sentiment 140 Tweet Corpus is one of the datasets that has ground truth and is also public available. The corpus contains 1.6 million machine-tagged Twitter messages. Microblogging websites have evolved to become a source of varied kind of information.

Microblogging websites have evolved to become a source of varied kind of information. This is due to nature of micro blogs on which people post real time messages about their opinions on a variety of topics, discuss current issues, complain, and express positive sentiment for products they use in daily life. In fact, companies manufacturing such products have started to poll these microblogs to get a sense of general sentiment for their product. Many time these companies study user reactions and reply to users on microblogs. One challenge is to build technology to detect and summarize an overall sentiment.

Our project resembles the analyze of tweets by the peoples on certain products of companies or brands or performed by political leaders. In order to do this we analyzed tweets from Twitter. Tweets are a reliable source of information mainly because people tweet about anything and everything they do including buying new products and reviewing them. Besides, all tweets contain hash tags which make identifying relevant tweets a simple task. A number of research works has already been done on twitter data. Most of which mainly demonstrates how useful this information is to predict various outcomes. Our current research deals with outcome prediction and explores localized outcomes.

We collected data using the Twitter public API which allows developers to extract tweets from twitter programmatically. The collected data, because of the random and casual nature of tweeting, need to be filtered to remove unnecessary information. Filtering out these and other problematic tweets such as redundant ones, and ones with no proper sentences was done next. As the preprocessing phase was done in certain extent it was possible to guarantee that analyzing these filtered tweets will give reliable results. Twitter does not provide the gender as a query parameter so it is not possible to obtain the gender of a user from his or her tweets. It turned out that twitter does not ask for user gender while opening an account so that information is seemingly unavailable.

**STATEMENT OF THE PROBLEM**

The problem at hand consists of two subtasks:

* Phrase Level Sentiment Analysis in Twitter :  Given a message containing a marked instance of a word or a phrase, determine whether that instance is positive, negative or neutral in that context.
* Sentence Level Sentiment Analysis in Twitter: Given a message, decide whether the message is of positive, negative, or neutral sentiment.

**OBJECTIVES OF THE PROJECT**

The objectives of this project are:

* To implement an algorithm for automatic classification of text into positive and negative
* Sentiment Analysis to determine the attitude of the mass is positive, negative or neutral towards the subject of interest
* Graphical representation of the sentiment

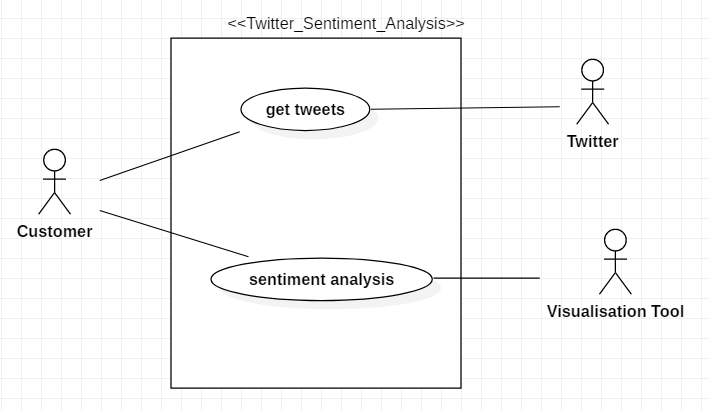
**SCOPE OF PROJECT**

This project will be helpful to the companies, political parties as well as to the common people. It will be helpful to political party for reviewing about the program that they are going to do or the program that they have performed. Similarly companies also can get review about their new product on newly released hardwares or softwares. Also the movie maker can take review on the currently running movie. By analyzing the tweets analyzer can get result on how positive or negative or neutral are peoples about it.

**SYSTEM OVERVIEW**

We are fetching live tweets from twitter on any topic we want and we will be performing sentiment analysis on the tweets and determine whether it is positive, negative or neutral.

USE CASE DIAGRAM :

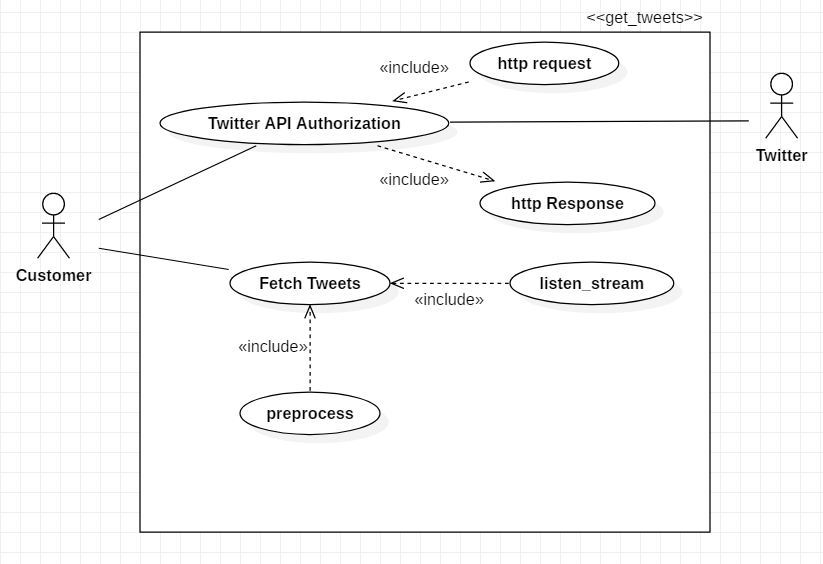


USE CASE DESCRIPTION :

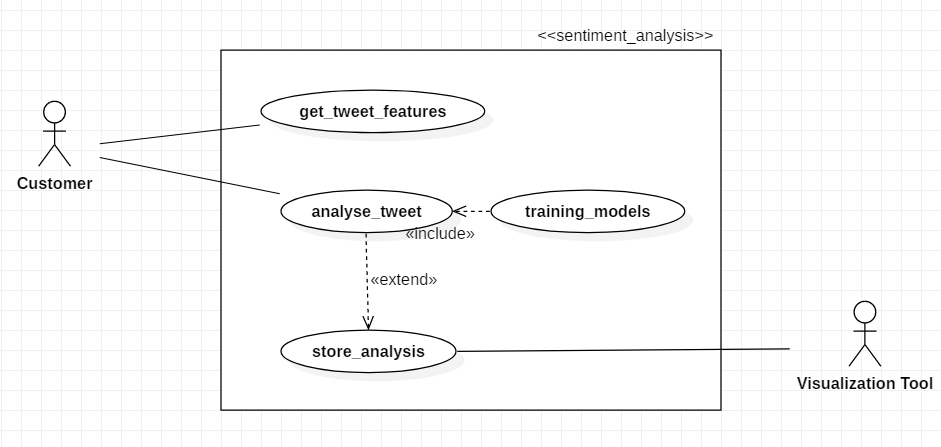
GET\_TWEETS : The customer will get tweets from the twitter using various keys such as consumer key, consumer secret, access token and access secret.

SENTIMENT\_ANALYSIS : Sentiment analysis is done on the fetched tweets through various training models and the analysis of the tweets is stored in the visualization tool.

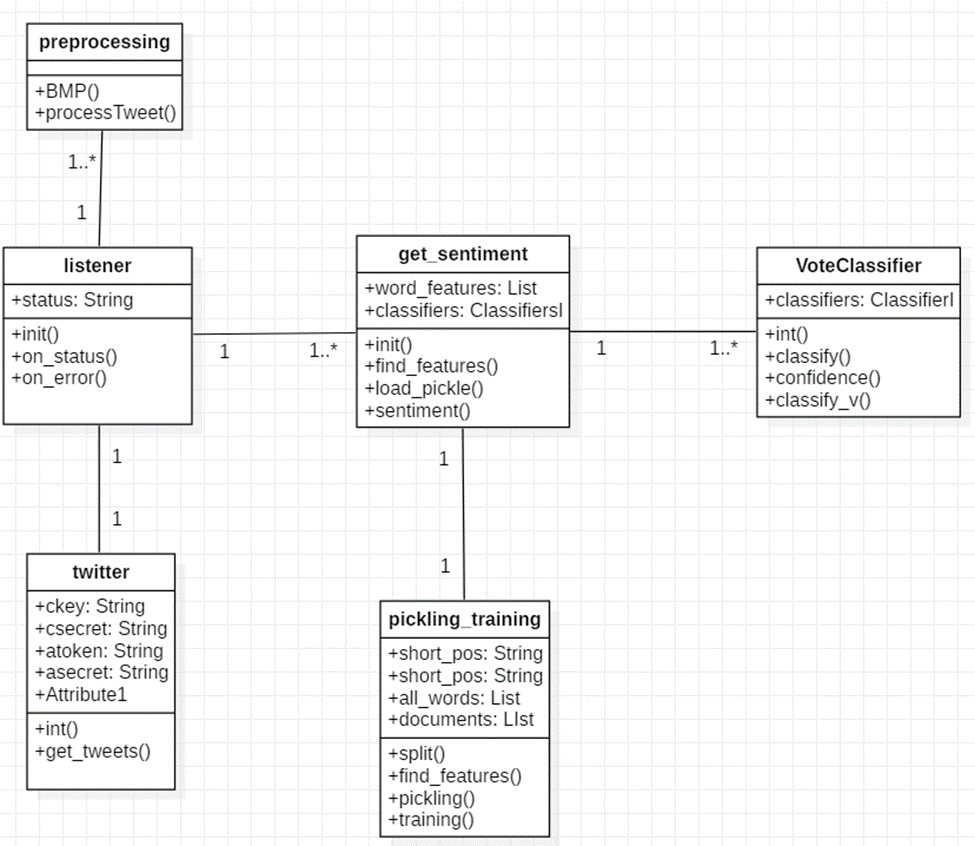
SUB USE CASE DIAGRAM FOR GET\_TWEETS :



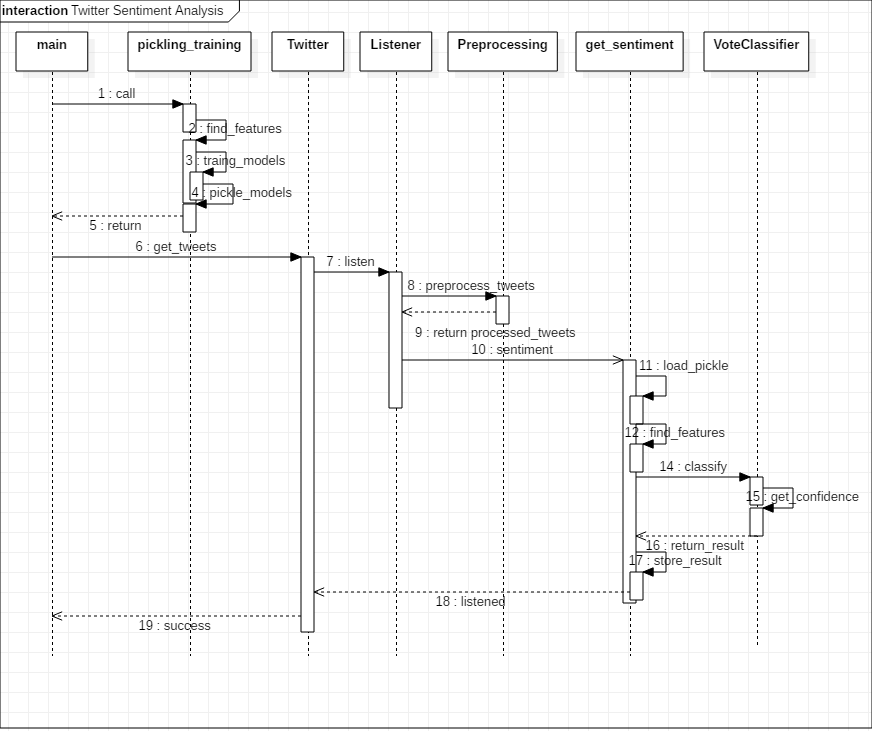
SUB USE CASE DIAGRAM FOR SENTIMENT ANALYSIS :



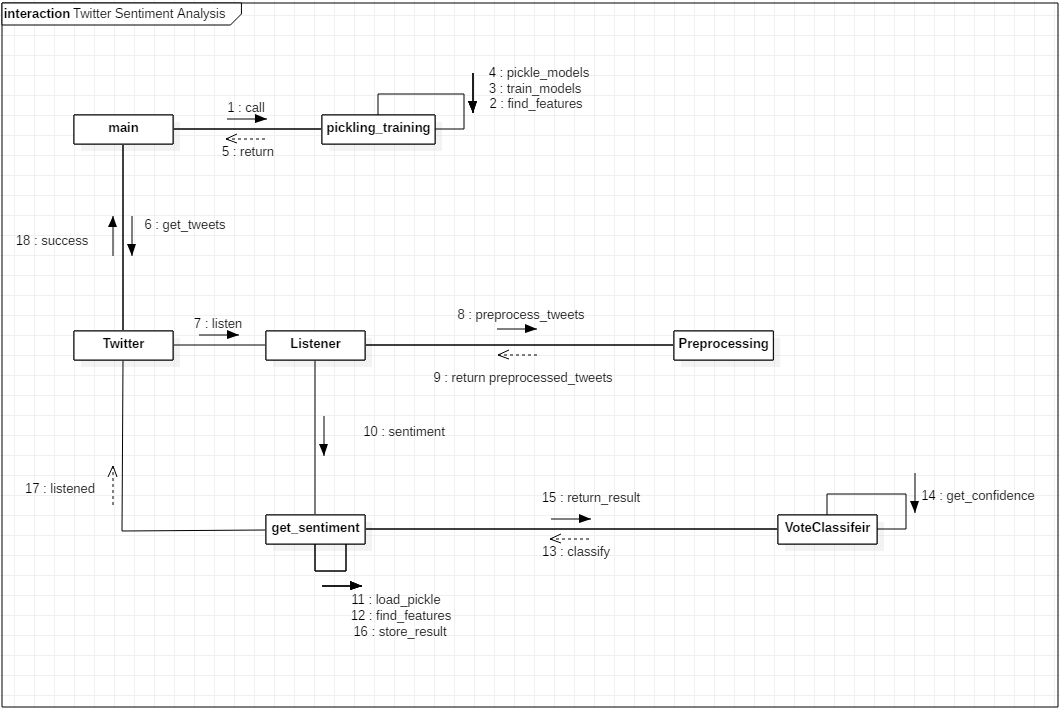
CLASS DIAGRAM :



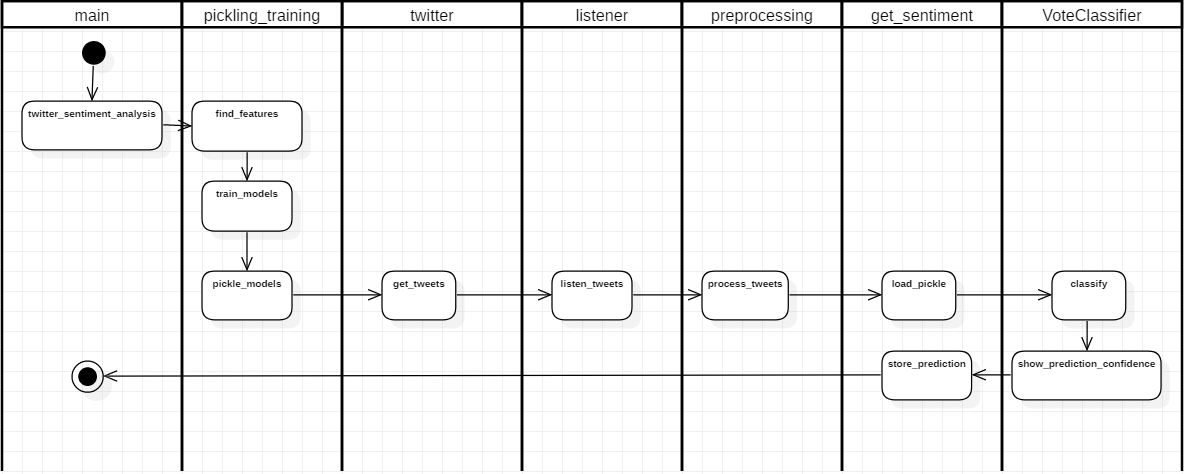
**SEQUENCE DIAGRAM:**



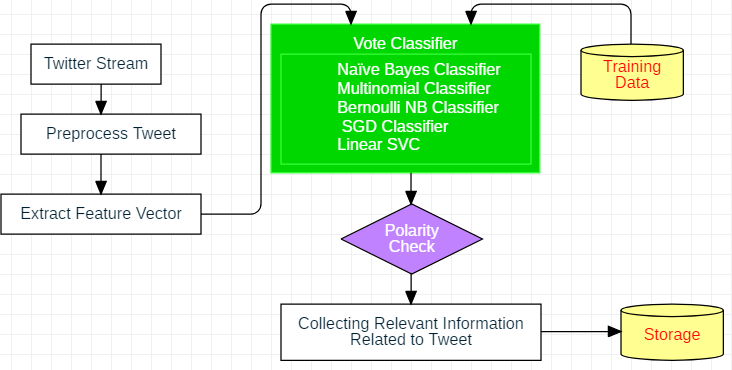
**COLLABORATION DIAGRAM :**



**ACTIVITY DIAGRAM :**



**FLOW CHART :**



CLASS IDENTIFIED:

• get\_sentiment

• pickling\_training

• VoteClassifier

• twitter

• listener

• preprocessing

CLASS DESCRIPTION :

get\_sentiment : It finds out the sentiment of the fetched tweet whether it is positive or negative.

It loads pickle and then finds the sentiment of the tweet.

pickling\_training : It splits the text, find it’s features through pickling and training.

VoteClassifier : It classifies the tweet and finds out the confidence and then append the result in

the csv file for further visualization.

twitter : It provides the consumer key, consumer secret, access token and access secret and thus

help in fetching live tweets from the twitter.

listener : It helps in listening the tweets which are fetched from the twitter.

preprocessing : It preprocesses the fetched tweet to get the sentiment of the tweet.

**CODE :**

**SENTIMENT\_MOD\_OO :**

import nltk

import random

import csv

from nltk.classify.scikitlearn import SklearnClassifier

import pickle

from sklearn.naive\_bayes import MultinomialNB, GaussianNB, BernoulliNB

from sklearn.linear\_model import LogisticRegression, SGDClassifier

from sklearn.svm import SVC, LinearSVC, NuSVC

from nltk.classify import ClassifierI

from statistics import mode

from nltk.tokenize import word\_tokenize

class VoteClassifier(ClassifierI):

def \_\_init\_\_(self, \*classifiers):

self.\_classifiers = classifiers

def classify(self, features):

votes = []

csv\_add=[]

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

return max(set(votes), key = votes.count)

def confidence(self, features):

votes = []

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

choice\_votes = votes.count(max(set(votes), key = votes.count))

conf = choice\_votes / len(votes)

return conf

def classify\_v(self,features):

csv\_add=[]

for c in self.\_classifiers:

v=c.classify(features)

csv\_add.append(v)

return csv\_add

class get\_sentiment:

def \_\_init\_\_(self):

self.word\_features=self.load\_pickle("word\_features.pickle")

self.classifier=self.load\_pickle("originalnaivebayes.pickle")

self.MNB\_classifier=self.load\_pickle("MNB\_classifier.pickle")

self.BernoulliNB\_classifier=self.load\_pickle("BernoulliNB\_classifier.pickle")

self.LogisticRegression\_classifier=self.load\_pickle("LogisticRegression\_classifier.pickle")

self.LinearSVC\_classifier=self.load\_pickle("LinearSVC\_classifier.pickle")

self.SGDC\_classifier=self.load\_pickle("SGDC\_classifier.pickle")

def find\_features(self,document):

word = word\_tokenize(document)

features = {}

for w in self.word\_features:

features[w] = (w in word)

return features

def load\_pickle(self,filename):

open\_file = open(filename,"rb")

classifier = pickle.load(open\_file)

open\_file.close()

return classifier

def sentiment(self,text):

feats = self.find\_features(text)

voted\_classifier = VoteClassifier(self.classifier, self.MNB\_classifier, self.BernoulliNB\_classifier, self.LogisticRegression\_classifier, self.SGDC\_classifier, self.LinearSVC\_classifier)

csv\_add=[]

csv\_add=voted\_classifier.classify\_v(feats)

print(csv\_add)

csv\_add.append(text.encode("utf-8"))

csv\_add.append("Facebook")

print(csv\_add)

with open("twitter\_1.csv", "a",newline='') as fp:

wr = csv.writer(fp, dialect='excel')

wr.writerow(csv\_add)

return voted\_classifier.classify(feats),voted\_classifier.confidence(feats)

**TRAINING\_PICKLING\_OO :**

import nltk

import random

from nltk.classify.scikitlearn import SklearnClassifier

import pickle

from sklearn.naive\_bayes import MultinomialNB, BernoulliNB

from sklearn.linear\_model import LogisticRegression, SGDClassifier

from sklearn.svm import SVC, LinearSVC, NuSVC

from nltk.classify import ClassifierI

##from statistics import mode

from nltk.tokenize import word\_tokenize

class VoteClassifier\_train(ClassifierI):

def \_\_init\_\_(self, \*classifiers):

self.\_classifiers = classifiers

def classify(self, features):

votes = []

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

return (max(set(votes), key=votes.count))

def confidence(self, features):

votes = []

for c in self.\_classifiers:

v = c.classify(features)

votes.append(v)

choice\_votes = votes.count(max(set(votes), key=votes.count))

conf = choice\_votes / len(votes)

return conf

class pickling\_training:

short\_pos = open("short\_reviews/positive.txt", "r").read()

short\_neg = open("short\_reviews/negative.txt", "r").read()

# move this up here

all\_words = []

documents = []

# j is adject, r is adverb, and v is verb

# allowed\_word\_types = ["J","R","V"]

allowed\_word\_types = ["J"]

word\_features=[]

def split(self):

for p in self.short\_pos.split('\n'):

self.documents.append((p, "pos"))

words = word\_tokenize(p)

pos = nltk.pos\_tag(words)

for w in pos:

if w[1][0] in self.allowed\_word\_types:

self.all\_words.append(w[0].lower())

for p in self.short\_neg.split('\n'):

self.documents.append((p, "neg"))

words = word\_tokenize(p)

pos = nltk.pos\_tag(words)

for w in pos:

if w[1][0] in self.allowed\_word\_types:

self.all\_words.append(w[0].lower())

save\_documents = open("document.pickle", "wb")

pickle.dump(self.documents, save\_documents)

save\_documents.close()

self.all\_words = nltk.FreqDist(self.all\_words)

self.word\_features = list(self.all\_words.keys())[:5000]

save\_word\_features = open("word\_features.pickle", "wb")

pickle.dump(self.word\_features, save\_word\_features)

save\_word\_features.close()

def find\_features(self,document):

words = word\_tokenize(document)

features = {}

for w in self.word\_features:

features[w] = (w in words)

return features

def pickling(self,classifier,filename):

save\_classifier = open(filename, "wb")

pickle.dump(classifier, save\_classifier)

save\_classifier.close()

def training(self):

featuresets = [(self.find\_features(rev), category) for (rev, category) in self.documents]

random.shuffle(featuresets)

print(len(featuresets))

testing\_set = featuresets[10000:]

training\_set = featuresets[:10000]

save\_featuresets = open("featuresets.pickle", "wb")

pickle.dump(featuresets, save\_featuresets)

save\_featuresets.close()

classifier = nltk.NaiveBayesClassifier.train(training\_set)

print("Naive\_Bayes\_classifier accuracy percent:", (nltk.classify.accuracy(classifier, testing\_set)) \* 100)

self.pickling(classifier,"originalnaivebayes.pickle")

MNB\_classifier = SklearnClassifier(MultinomialNB())

MNB\_classifier.train(training\_set)

print("MNB\_classifier accuracy percent:", (nltk.classify.accuracy(MNB\_classifier, testing\_set)) \* 100)

self.pickling(classifier,"MNB\_classifier.pickle")

BernoulliNB\_classifier = SklearnClassifier(BernoulliNB())

BernoulliNB\_classifier.train(training\_set)

print("BernoulliNB\_classifier accuracy percent:", (nltk.classify.accuracy(BernoulliNB\_classifier, testing\_set)) \* 100)

self.pickling(classifier,"BernoulliNB\_classifier.pickle")

LogisticRegression\_classifier = SklearnClassifier(LogisticRegression())

LogisticRegression\_classifier.train(training\_set)

print("LogisticRegression\_classifier accuracy percent:",(nltk.classify.accuracy(LogisticRegression\_classifier, testing\_set)) \* 100)

self.pickling(classifier,"LogisticRegression\_classifier.pickle")

LinearSVC\_classifier = SklearnClassifier(LinearSVC())

LinearSVC\_classifier.train(training\_set)

print("LinearSVC\_classifier accuracy percent:", (nltk.classify.accuracy(LinearSVC\_classifier, testing\_set)) \* 100)

self.pickling(classifier,"LinearSVC\_classifier.pickle")

SGDC\_classifier = SklearnClassifier(SGDClassifier())

SGDC\_classifier.train(training\_set)

print("SGDClassifier accuracy percent:", nltk.classify.accuracy(SGDC\_classifier, testing\_set) \* 100)

self.pickling(classifier,"SGDC\_classifier.pickle")

voted\_classifier = VoteClassifier\_train(classifier, MNB\_classifier, BernoulliNB\_classifier, LogisticRegression\_classifier, SGDC\_classifier, LinearSVC\_classifier)

print("voted\_classifier accuracy percent:",(nltk.classify.accuracy(voted\_classifier, testing\_set))\*100)

**TWEET\_DATA\_OO :**

from tweepy import Stream

from tweepy import OAuthHandler

from tweepy.streaming import StreamListener

from tweepy import API

import json

import re

from sentiment\_mod\_oo import get\_sentiment

from sentiment\_mod\_oo import VoteClassifier

class twitter:

#consumer key, consumer secret, access token, access secret.

def \_\_init\_\_(self):

self.ckey=" "

self.csecret=" "

self.atoken=" "

self.asecret=" "

def get\_tweets(self):

auth = OAuthHandler(self.ckey, self.csecret)

auth.set\_access\_token(self.atoken, self.asecret)

twitterStream = Stream(auth, listener())

twitterStream.filter(track=["Facebook","facebook"], languages = ["en"], stall\_warnings = True)

class listener(StreamListener):

def \_\_init\_\_(self, api=None):

self.ob=preprocessing()

self.ob1=get\_sentiment()

super(listener, self).\_\_init\_\_()

self.num\_tweets = 0

def on\_status(self, status):

if hasattr(status, 'retweeted\_status'):

try:

self.tweet = status.retweeted\_status.extended\_tweet["full\_text"]

except:

self.tweet = status.retweeted\_status.text

else:

try:

self.tweet = status.extended\_tweet["full\_text"]

except AttributeError:

self.tweet = status.text

if status.coordinates:

print ('coords:', status.coordinates)

if status.place:

print ('place:', status.place.full\_name)

self.tweet=self.ob.BMP(self.tweet)

self.tweet=self.ob.processTweet2(self.tweet)

print("Name",status.user.screen\_name)

sentiment\_value, confidence = self.ob1.sentiment(self.tweet)

print(self.tweet, sentiment\_value, confidence)

if (confidence\*100>=80):

output = open("twitter-out.txt","a")

output.write(sentiment\_value)

output.write("\n")

output.close()

print("\n\n")

self.num\_tweets += 1

if self.num\_tweets < 10:

return True

else:

return False

def on\_error(self, status):

print(status)

class preprocessing:

def BMP(self,s):

return "".join((i if ord(i) < 10000 else '' for i in s))

def processTweet2(self,tweet):

tweet = tweet.lower()

tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)

tweet = re.sub('@[^\s]+','AT\_USER',tweet)

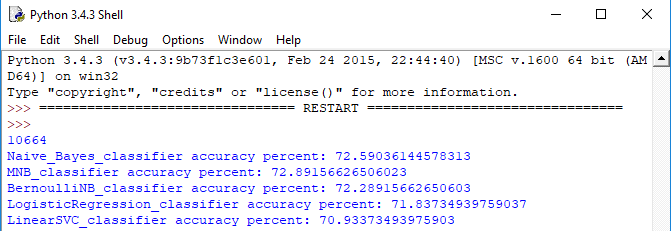
tweet = re.sub('[\s]+', ' ', tweet)

tweet = re.sub(r'#([^\s]+)', r'\1', tweet)

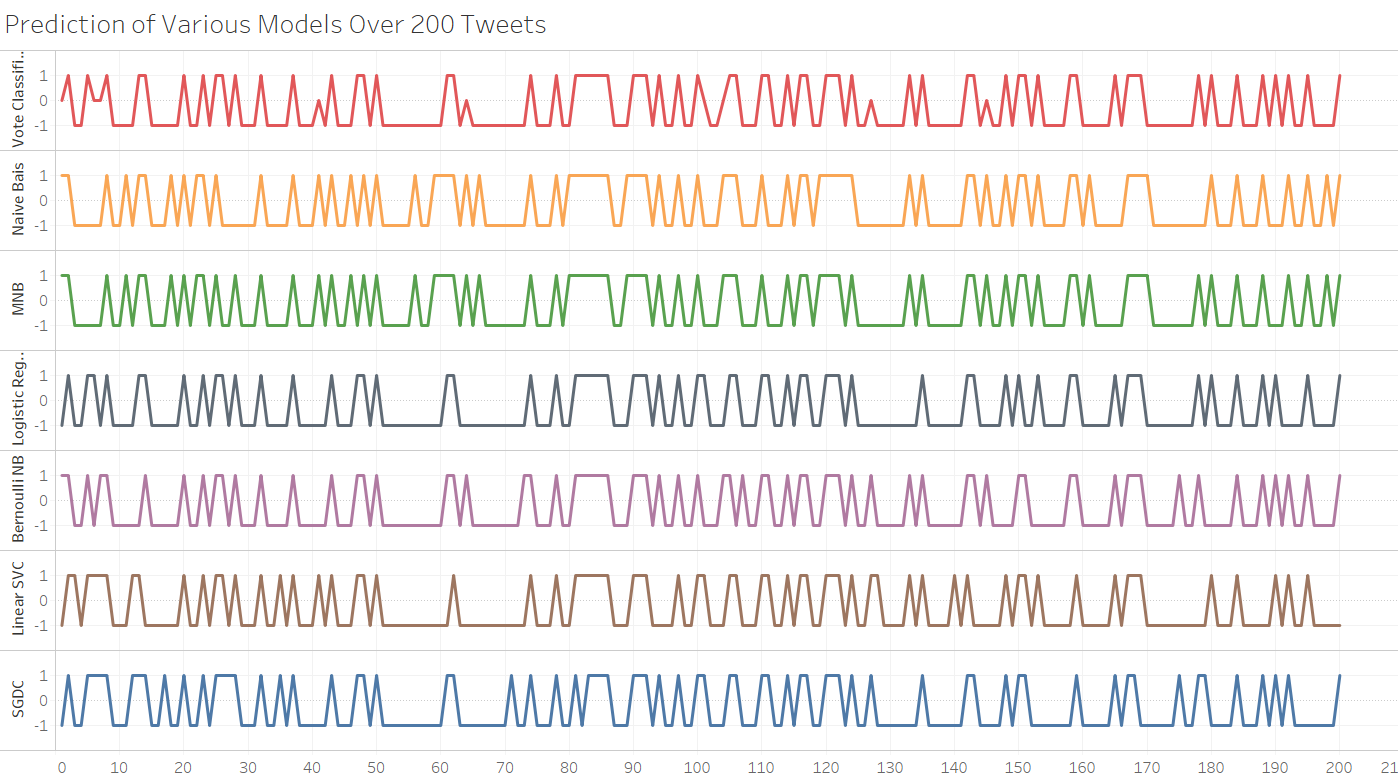
tweet = tweet.strip('\'"')

return tweet

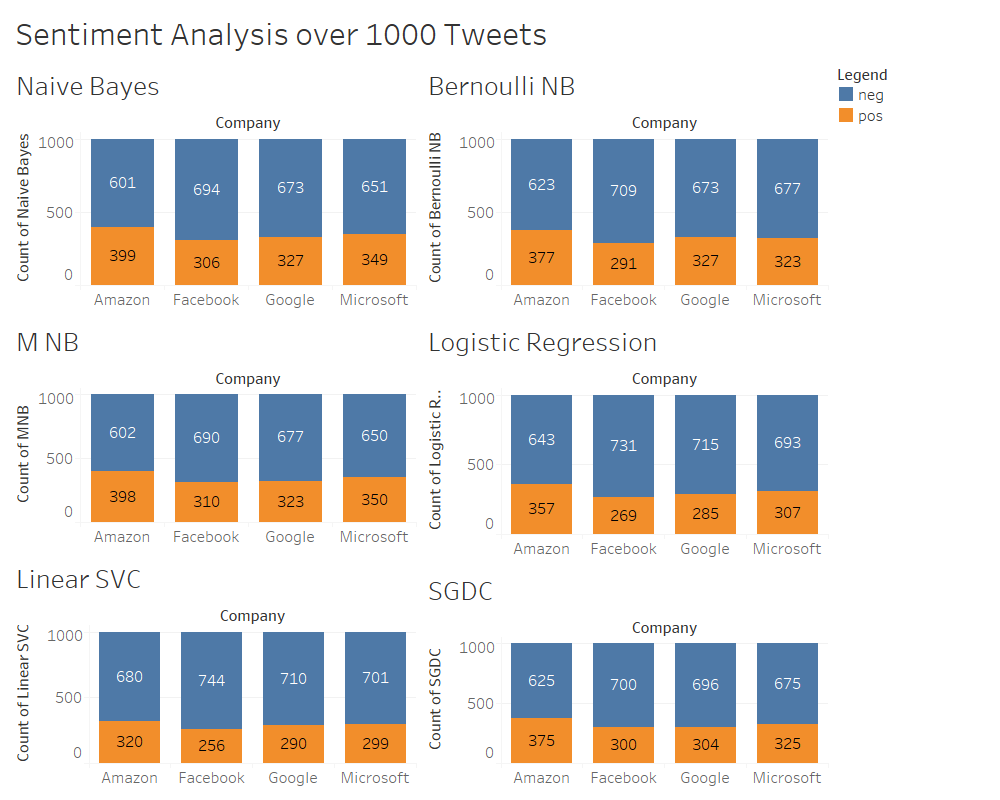
**SCREENSHOT :**

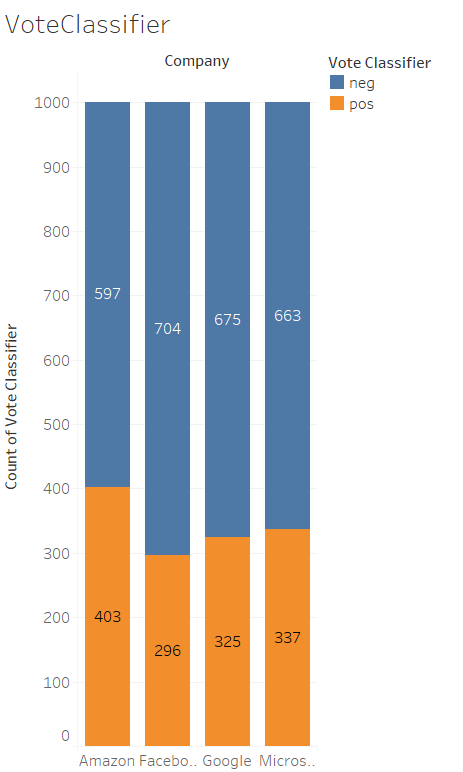


**Visualizations**

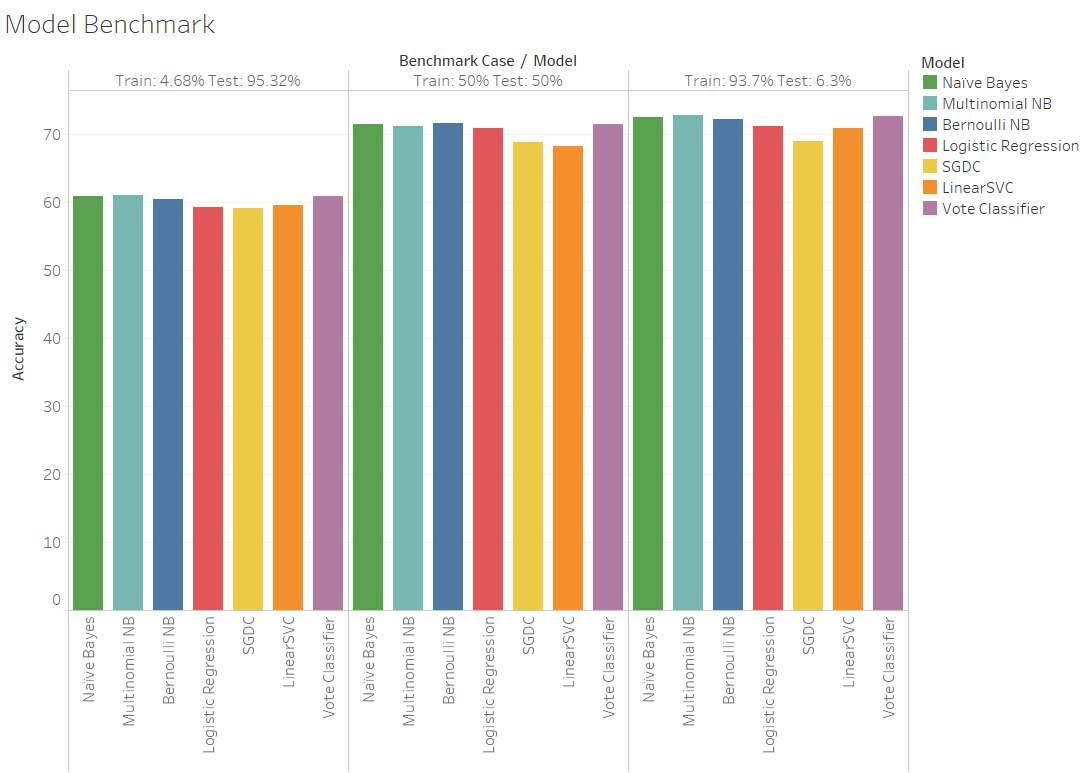


By this, we can easily notice how single tweet can be differently classified by various models resulting in need of better consistency which is achieved by collective voting of different classification models to vote classifier.





With this observe, how classification are different across models. With this visualization, it comes as a shock to engineering aspirants that most of the sentiments about the companies are negative. By going through quite of them, its requested by people that they improve their services.



In benchmarking it is observed that in the training and testing, Naive Bayes and Vote Classifier has comparable accuracies all over board. Vote Classifiers come ahead when we training dataset is significantly large as it utilizes vote of each classifier to determine its result. It provides with a significant new effectiveness that it provide confidence with prediction with can help us to sort through results of low confidence and focus on one with much higher confidence.

**ANALYSIS AND RESULTS**

**Analysis**

Analysis We collected dataset containing positive and negative data. Those dataset were trained data and was classified using Naïve Bayes Classifier. Before training the classifier unnecessary words, punctuations, meaning less words were cleaned to get pure data. To determine positivity and negativity of tweets we collected data using twitter API. Those data were stored in database and then retrieved back to remove those unnecessary word and punctuations for pure data. To check polarity of test tweet we train the classifier with the help of trained data.

**Result**

After facing a number of errors, successful elimination of those errors we have completed our project with continuous efforts.

**CONCLUSION**

We have completed our project using python as language. We were able to determine the positivity and negativity of each tweet. Based on those tweets we represented them in various diagrams like bar graph, scatter-plot.