Internship Project Report

SCOIAL MEDIA SENTIMENT ANALYSIS

At

IC SOLUTIONS

Submitted by;

Pruthvi Kumar U [1GG18CS032]

Sagar Gowda B V [1GG18CS034]

INSTRUCTOR: NITHIN

Acknowledgement:

Being part of this internship was a great experience. It is with grateful heart that I thank the coordinators of IC Solutions for giving me an opportunity to intern with their organization. I express my gratitude to my instructor, NITHIN, for providing me with useful knowledge that can be applied on practical tasks and also for guiding and encouraging me to do my best in this internship.

During the period of my internship, I have received generous help and suggestions from many quarters, for which I would like to remember and thank them all with deep gratitude and great pleasure.

I must give special thanks to my family for supporting and encouraging me to complete the internship. Above all, I thank God for granting me the capability to complete this work.

About the Company:

IC Solution (ICS) is a digital service provider that aims to provide software, designing and marketing solutions to individuals and businesses. At ICS, we believe that service and quality is the key to success.

We provide all kinds of technological and designing solutions from Billing Software to Web Designs or any custom demand that you may have. Experience the service like none other!

Some of our services include:

Development - We develop responsive, functional and super-fast websites. We keep User Experience in mind while creating websites. A website should load quickly and should be accessible even on a small view-port and slow internet connection.

Mobile Application - We offer a wide range of professional Android, iOS & Hybrid app development services for our global clients, from a startup to a large enterprise.

Design - We offer professional Graphic design, Brochure design & Logo design. We are experts in crafting visual content to convey the right message to the customers.

Consultancy - We are here to provide you with expert advice on your design and development requirement.

Videos - We create a polished professional video that impresses your audience

INDEX

Summary

In this mini-project we have chosen to do sentiment analysis of social media websites such as twitter to gain insights into the people's opinion towards prime ministerial candidates for the Lok Sabha election 2019.

Social media provides a platform for people's opinion of a person or event or topic to be heard from anywhere at any time and is the easiest and fastest way for them to do it. So, analyzing these sentiments will be of immense use in knowing the trending topics and the mood of the people towards those topics among other things.

The mini-project also aims at implementing and comparing contemporary machine learning text classification algorithms to predict the sentiment of a piece of text.

The scope of the project is to analyze the sentiments of the texts posted on social media and useful knowledge regarding the mood of the people towards the Lok Sabha 2019 elections after which contemporary machine learning algorithms for text classification are implemented and compared as to which among them performs better.

Highlights of this project:

- Scrapping data from twitter based on keyword search
- Analyze the sentiments of the texts from the data collected.
- Gain useful knowledge after processing the collected data.
- Compare contemporary text classification machine learning algorithms and justification.

List of Tables

List of figures

Abbreviations

API	Application Program Interface
CSV	Comma Separated Values
HTML	Hyper Text Mark-up
Language NLP	Natural Language
API wrapper SA	Sentiment Analysis

Introduction

Preamble to the Chapter

Election is the backbone of democracy. India being the largest democracy in the world holds its national general election once every five years where every individual of legal age is allowed to cast their vote and decide the fate of the country. Unlike the US, where the people directly vote for the presidential candidates , India follows a parliamentary form of governance where people vote for representatives of their constituency who then select a prime minister for the nation. Moreover, the Indian constitution permits a multi-party system where any number of parties can contest the elections. The parties campaign exhaustively from rallies to social media.

Introduction to project

Election is the foundation on which democracy stands. It's the most vital instrument of democracy wherever the voters communicate with the representatives. One vital component in elections is the election polls/survey. Conducting polls can be time and resource consuming and may not accurately predict the election outcomes. thus, attempting to resolve the accuracy and resource problems, we explore the chance of exploitation of knowledge from social media because of the vast amount of opinions posted by people on these platforms to predict the result of election.

Social media has become the foremost widespread communication tool on the net. many scores of messages area unit being denote each day within the widespread social media sites like Twitter.[1] explicit in their paper that social media websites become valuable sources for opinion mining because of folks' post everything, from the main points of their existence, like the product and services they use, to opinions concerning current problems like their political and social views.

Humans don't observe clear criteria for evaluating the sentiment of a piece of text. Judging the sentiment for a particular piece of text is a subjective task which is heavily influenced by personal experiences, thoughts, and beliefs. By using a centralized sentiment analysis system, the same criteria can be applied to all the data. This helps to reduce errors and improve data consistency. There is just too much data to process manually. Sentiment analysis allows to process data at scale in an efficient and cost- effective way.

This project is centered around analyzing sentiments of text from multiple social media networking sites. One of the social media analyses tackled here is Twitter, which is the world's largest micro-blogging website and it is the place on the internet where people express their political views and ideologies in addition to other things through short messages do voters use this platform to openly support political parties but also to express their opinions on every called 'tweets'.

Not only current affair and issue happening in the country. Recently it has become a common ground for politicians and party leaders to convey their messages to the people of the nation and hold campaigns among other things. So, analyzing the sentiments of the voters on this media will allow us to paint a picture of the political sway of the country.

Finally, after getting useful knowledge from the analyzed tweets and comments, the project works on building and comparing contemporary machine learning algorithms for sentiment classification of text. As large amount of data will be collected and used for training these models, the entire process of building, training, testing, and analyzing the performance of these models will be undertaken using spark and python which leverages the in-memory processing ability of spark and parallelizing the necessary processes and making use of its pipelining feature. As traditional methods of model training are time consuming and resource intensive this proposed approach is an effort to overcome these drawbacks

Organization of the report

Chapter 2 is necessary at start as it sets the scene by discussion of related work and adds a research perspective to the project topic. Various principles and assumptions are described and explained in detail to avoid any ambiguities related to the project work. Some key definitions and their implications are discussed as its understanding is inevitable to truly appreciate the project work. Merits, Demerits and Applicability of the project is noted followed by relative comparison with peripheral topics. At the end related work is discussed to demonstrate our originality.

Chapter 3 mainly states the title of the project as it sets a base idea of what the entire project is about. It is then followed by the aim and objectives of the project which is essential to understand the intended outcomes of the project. These two entities are essential to know where the project is headed and how it plans to get there. The methodologies listed for each objective statement extensively describes the approach taken to reach the respective objectives along with the resources utilized.

Chapter 4 can be broadly classified into three phases

The first phase mainly solves the tasks of data collection and cleaning which is the processes of acquiring relevant data for processing and analysis. The collected data forms the bedrock of the entire project hence its necessary to gather quality data in large quantities in the right and most efficient and cost-effective way. The gathered data needs to be organized in a suitable way and transformed into a usable form from which useful knowledge can be obtained.
The second phase investigates the task of sentiment labelling and then visualizing the data. Categorizing the cleaned text into one the three sentiments is an important process as it helps to create training data for the nex phase. The process of visualization is fundamental to the project as one of the purposes is to gain as much knowledge as possible from the data.
The third and final phase is all about building a text classifier to predict the sentiment of text using machine learning techniques. This is given special importance in the overall project as the entire task of training the models is carried out using various features of spark tool.

Chapter 5 displays the results of the problem solved. The sentiments of the people towards the prime ministerial candidates are displayed and discussed in addition to the word clouds that show the most frequently used words in the tweets/comments, the most liked tweet/comment and the most shared tweet/ comment and various other visuals. The results also show a heat map of locations from where these tweets are originating from. The last part of the result analyses and discusses the performance of the trained machine learning models for sentiment classification of texts and their comparisons.

Chapter 6 summarizes the project costing; as this project is mainly research based and software oriented it is highly cost effective as most of the resources used are open-source software. The main project costs are the requirement of a stable internet connection for data acquisition and the heavy reliance on high-speed processors and sufficient memory for processing data. Another significant costing is man hours invested in th project as it requires intensive programming skills and time and effort spent in learning the open-source libraries used throughout the project.

Chapter 7 provides a conclusion for the project while highlighting the main purpose of the project and the effor made towards realizing the goals set for the project. It also suggests many useful works that can be done in th

future while also hinting at what can be done going further ahead.

Background Theory

Preamble to the Chapter

Theoretical basis forms the core of any project. In this chapter, we discuss several works related to sentiment analysis of social media for election result prediction along with approaches put forth by researchers in the field. This chapters also contains key definitions for understanding the implications of the project followed by merits, demerits, and applications of the project. Finally, further discussion of related work is done to defend the originality of the project.

Discussion of related work at start to set scene

In this section, we discuss related works about predicting the result of an election using social media opinions. Different researchers use different approaches, there are researchers who try to discover the political preference of a user, then relate it to the election and there are others who use selected text related to the upcoming election and figure out vote preference of the user using that data.

Various strategies used for inferring political leaning such as profile information, user behavior, user graph, Twitter specific feature (reply/re-tweet), sentiment from text content. For example, In[2], the authors used tex containing parties' name in several political events to assign a political/ideological leaning of the user who posted the text. Like the previous method, [3] used the text of a user regarding a political party to infer the political leaning. [4] assigned a score to every member which a user is following, then a political preference is assigned based on that score. In [5], the authors compared several features such as user's posting behavior, linguistic content, follower, reply and shares. They found out that the combination between user profile and linguistic outperform other features.

The second approach is by using selected data just days or weeks prior to the election. The prediction could b derived by comparing the number of texts mentioning each candidate or by comparing the number of tweets that has positive sentiments towards each candidate. The earliest research stated that the number of texts mentioning a party reflects the election result was shown in [6] where they found out that the prediction result from social media were only slightly worse than offline election polls. While [7] is the first research in which argued that sentiment detection approach from Twitter can replace the expensive and time intensive polling. So, in the project is to implement and infer the second approach to predict election results based on sentimen analysis using social media data.

Principles

There are many methods and algorithms to implement sentiment analysis systems, which can be classified as:
Rule-based systems that perform sentiment analysis based on a set of manually crafted rules.
Automatic systems that rely on machine learning techniques to learn from data.
Hybrid systems that combine both rules based and automatic approaches.
Rule-based Approach
Usually, rule-based approaches define a set of rules in scripting language that identify subjectivity, polarity, o the subject of an opinion.

The rules may use a variety of inputs, such as the following:
Classic NLP techniques like *stemming*, *tokenization*, *part of speech* tagging and parsing.

A basic example of a rule-based implementation would be the following:

Define two lists of polarized words (e.g. negative words such as *bad, worst, ugly,* etc. and positive words such as *good, best, beautiful*, etc.).

Given a text:
Count the number of positive words that appear in the text.
Count the number of negative words that appear in the text.

If the number of positive word appearances is greater than the number of negative word appearances return a positive sentiment, conversely, return a negative sentiment. Otherwise, return neutral.

This system is very naïve since it doesn't consider how words are combined in a sequence. A more advanced processing can be made, but these systems get very complex quickly. They can be very hard to maintain as new rules may be needed to add support for new expressions and vocabulary. Besides, adding new rules may have undesired outcomes because of the interaction with previous rules. As a result, these systems require important investments in manually tuning and maintaining the rules.

In this project we implement Automatic methods contrary to rule-based systems, which don't rely on manually crafted rules, but on machine learning techniques. The sentiment analysis task is usually modelled as a classification problem where a classifier is fed with a text and returns the corresponding category. The significance of this project is comparison of contemporary machine learning classification algorithms to predict the sentiment of text pertaining to elections by building machine learning pipelines in Pyspark.

Key definitions and its implications
Sentiment Analysis also known as *Opinion Mining* is a field within Natural Language Processing (NLP) that builds systems that try to identify and extract opinions within text. Currently, sentiment analysis is a topic of great interest and development since it has many practical applications of which election prediction is prominent. Since publicly and privately available information over Internet is mainly unstructured and constantly growing, a large number of texts expressing political opinions and opinions of political leaders are available in review sites, forums, blogs, and social media.
With the help of sentiment analysis systems, this unstructured information could be automatically transformed into structured data of public opinions about government services and schemes in addition to political events. This data can be very useful for parties involved and other benefactors and public relations.

Merits, Demerits and Applicability

Merits:
Multi source social media analysis.
Web scrapping relevant data directly from the site using API provided by the medias instead of using data collected by third party sources.
Election result prediction of high accuracy.
Fast model training and exhaustive performance comparison and analysis.
Use of only twitter for analysis.

Demerits:
Web scrapping obeys the request rate limits officially set by the API offered by the medias which could lead to more time consumption to collect more data.

Applicability:
Social media activity monitoring.
Targeted advertising and marketing.
Helps election contesting parties and other concerned persons to develop better campaigning strategies

Discussion of related work to demonstrate originality
Researches have tried to compare these two methods, for example, [8] that tried to predict congress and senate election in several states of the US. They showed that though the method is the same, the prediction error can vary greatly. The research also showed that lexicon-based sentiment analysis improves the prediction result, but the improvement also varies in different states. Same result was shown in [9] where they predict the result of Irish general election using both methods and [10] which predict the Italian primary election. All the research showed that sentiment detection does reduce the error of the prediction result. Because of that, several researchers focused on improving the sentiment analysis, such as [10] and [11] who used more sophisticated sentiment analysis than lexicon based in the US presidential election, France legislative election, and Italy primary election.

Aim and Objectives

Preamble to the Chapter
In this chapter, the aim of the project has been depicted. The objectives that are to be fulfilled are also listed. This chapter also contains the objective table that are used followed. The table contains statement of the objective, methodologies and the resources utilized.

Title of Project

Sentiment Analysis using PySpark on Social-Media Data.

Aim of Project

To design and develop sentiment analysis model for Indian general elections 2019 using Social-Media Datase (Twitter).

Objectives of Project

To conduct literature survey on sentiment analysis using Spark and python.
To arrive at requirement specification for sentiment analysis using multisource social-media data.
To design and develop a sentiment analysis model for Indian general election 2019 using PySpark
To implement and compare contemporary machine learning techniques for sentiment analysis.
To test and validate the developed sentiment analysis techniques.
To document the report by unifying all the results and outcomes.

Method and Methodology

Table 3.1 Method and Methodology to attain each objective

Objective No.	Statement of the Objective	Method/ Methodology	Resources Utilized
1	To conduct literature survey on sentiment analysis using Spark and python.	Big data tool such as Apache Spark Various methods of data acquisition and data cleaning Machine learning models for multi- class text classification Different Python and Spark libraries for different tasks.	Internet, Research papers, Online journals and conference papers
2	To arrive at requirement specification for sentiment analysis using twitter	Installation and configuration of Python with Spark Setting up jupyter notebook development environment. Acquiring data-sets using the API provided by the social media site. The preprocessing procedures that need to be done on the collected dataset.	Jupyter notebook, Spark, and Python
3	To design and develop a sentiment analysis model for Indian general election 2019 dataset using PySpark	Suitable pre-processing approaches and storage of data. Splitting the data-set into training-set and test-set. Building machine learning models and training them using the training data set. Evaluating the accuracies of the trained models using test data. Data Visualization of the classified texts.	Jupyter Notebook, Tweepy, and Python libraries
4	To implement and compare contemporary machine learning techniques for sentiment analysis	Training Text – Classification models using Logistic regression. Naïve-Bayes Method. Random forests. One vs Rest Method.	PySpark ML library

5	To test and validate the developed sentiment analysis techniques	Testing the developed models using test data. Validating them based on evaluation metrics such as precision and recall	PySpark ML library
6	To document the report by unifying all the results and outcomes	6.1 Valid formal representation of results and outcomes.	MS Word

Problem Solving

Preamble to the Chapter

Typically, when running machine learning algorithms, it involves a sequence of tasks including pre-processing, feature extraction, model fitting, and validation stages. For example, when classifying text documents might involve text segmentation and cleaning, extracting features, and training a classification model with cross-validation. Though there are many libraries we can use for each stage, connecting the dots is not as easy as it may look, especially with large-scale datasets. Most ML libraries are not designed for distributed computation or they do not provide native support for pipeline creation and tuning.

Data Collection

Data collection is concerned with the accurate acquisition of data; although methods may differ depending on the field, the emphasis on ensuring accuracy remains the same. Accurate data collection is essential to ensure the integrity of the research, regardless of the field of study or data preference.
In this project chosen social media platforms are Twitter. Data from these platform can be collected using respective API's such as tweepy.

Twitter Data Collection

Before one start coding, one need to register for the Twitter API https://apps.twitter.com/. Here we need to register an app to generate various keys associated with our API. The Twitter API can be used to perform many actions like create and search.

```
####input your credentials here
consumer_key = ''
consumer_secret = ''
access_token = ''
access_token_secret = ''

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

Figure 4. 1 Token Generation to access tweepy

Above code is used to generate secrete keys to use functions offered by tweepy.

```
query = 'modi'
# Open/Create a file to append data
csvFile = open('modi_data_200k.csv', 'w', encoding='utf-8')
placeCsvFile = open('modi_geo_data.csv', 'w', encoding='utf-8')
#Use csv Writer
csvWriter = csv.writer(csvFile)
placeCsvWriter = csv.writer(placeCsvFile)

cursor = tweepy.Cursor(api.search, q=query+' -RT', count=1000, tweet_mode='extended',
                       lang="en", since="2019-01-01").items(200000)

tweet_count = 0
print(tweet_count,datetime.now().time())
```

```
for tweet_info in cursor:
    time.sleep(0.1)
    try:
        tweet_text = tweet_info.extended_tweet.full_text
    except AttributeError:
        tweet_text = tweet_info.full_text

    csvWriter.writerow([tweet_info.created_at,
                        tweet_info.user.screen_name,
                        tweet_text,
                        tweet_info.favorite_count,
                        tweet_info.retweet_count])

    if tweet_info.place is not None:
        placeCsvWriter.writerow([tweet_info.place.bounding_box.coordinates,
                             tweet_info.place.full_name,
                             tweet_info.place.country])

    tweet_count+=1
    if tweet_count%10000 == 0:
        print(tweet_count,datetime.now().time())
```

Figure 4. 2 Scrapping Tweets from twitter

In this above code Search query that is Modi is searched in twitter as hashtag. All tweets that contains Modi a hash tag is returned to CSV file. Cursor function will pass the parameter into the method for us whenever it makes a request.

The search comment's function will search the most recent comments with the term "Modi" in the body of the comment. This search is not case-sensitive, so it will find any occurrence of the term "Modi" regardless of capitalization. The API defaults to sorting by recently made comments first. All comments are returned to CSV file.

Data Pre-processing

Pre-processing the data is the process of cleaning and preparing the text for classification. Online texts contain usually lots of noise and uninformative parts such as HTML tags, scripts, and advertisements. In addition, on words level, many words in the text do not have an impact on the general orientation of it. Keeping those words makes the dimensionality of the problem high and hence the classification more difficult since each word in the text is treated as one dimension.

Twitter Data Pre-processing

```
col_names=['date','user','text','likes','retweets']

df = pd.read_csv('modi_data_200k.csv', names=col_names)

df.head()
#df.count()
```

```
df.shape
```

```
(199990, 5)
```

```
df = df.drop_duplicates('text')
df.shape
```

```
(191484, 5)
```

Figure 4. 5 Representing CSV file in Panda's frame

Representing CSV file twitter data set in pandas' data frame. Pandas' library allows many functions to apply on pandas Dataframe. Hence representing in pandas Dataframe is efficient for pre-processing. Duplicate tweets are dropped by using drop duplicates function of panda's library.

```
df['date'] = pd.to_datetime(df['date'])
df = df.sort_values(by='date',ascending=True)
df = df.reset_index().drop('index',axis=1)
df.head()
```

Figure 4. 6 Code for sorting twitter frames by Date

Sentiment Labelling

Before developing any model for predicting sentiment data must be labelled this process is called sentiment labelling. After this dataset can be split into Train set and Test set. To label the datasets python library Textblob is used. Textblob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```
def analyze_sentiment(text):
    analysis = TextBlob(text)

    if analysis.sentiment.polarity > 0:
        return 1
    elif analysis.sentiment.polarity == 0:
        return 0
    else:
        return -1


df['category'] = df['clean_comment'].apply(analyze_sentiment)
df.head()
```

Figure 4. 9 Sentiment Labelling of cleaned text using textblob

The sentiment property returns a named tuple of the form Sentiment (polarity, subjectivity). Polarity is a float value within the range [-1.0 to 1.0] where 0 indicates neutral, +1 indicates a very positive sentiment and -1 represents a very negative sentiment. Subjectivity is a float value within the range [0.0 to 1.0] where 0.0 is very objective and 1.0 is very subjective. Subjective sentence expresses some personal feelings, views, beliefs, opinions, and speculations whereas Objective sentences are factual.

Textblob goes along finding words and phrases it can assign polarity and subjectivity to, and it averages them all together for longer text. Data are labelled based on polarity values. All polarity values which is greater than 1 are classified into positive sentiment that is +1 and values less than 0 classified into negative sentiment that is -1 and polarity value with zero are returned as neutral sentiment that is 0.

Data visualization

Data visualization methods refer to the creation of graphical representations of information. Visualization plays an important part of data analytics and helps interpret big data in a real-time structure by utilizing complex sets of numerical or factual figures.

With the seemingly infinite streams of data readily available to today's business across industries, the challenges lie in data interpretation, which is the most valuable insight to the individual organization as well as its aims, goals, and long-term objectives. Due to the way the human brain processes information, presenting insights in charts or graphs to visualize significant amounts of complex data is more accessible than relying on spreadsheets or reports. Visualizations offer a swift, intuitive, and simpler way of conveying critical concepts universally.

Data Visualization Techniques: Diagrams, charts, graphs
Python libraries used for Data visualization – Matplotlib, seaborn and folium.
The extracted tweets are stored in a csv file and is read as df, 'numpy' Is used as 'np', a package of python for general purpose array processing. 'Max' is a method of numpy that returns the maximum of array.

```
#extract the tweet with more FAVs and more RTs:

fav_max = np.max(df['likes'])
rt_max = np.max(df['retweets'])

fav = df[df.likes == fav_max].index[0]
rt = df[df.retweets == rt_max].index[0]

# Max FAVs:
print("\nThe tweet with more likes is: \n{}".format(df['text'][fav]))
print("Number of likes: {}".format(fav_max))
#print("{} characters.\n".format(data['len'][fav]))

# Max RTs:
print("\nThe tweet with more retweets is: \n{}".format(df['text'][rt]))
print("Number of retweets: {}".format(rt max))
```

Figure 4. 10 Statistical analyses of data. The above snippet is a code that prints the 'tweet' that has maximum number of likes and retweets.

```
# Create time series for the data:

#tlen = pd.Series(data=data['len'].values, index=data['Date'])
tfav = pd.Series(data=df['likes'].values, index=df['date'])
tret = pd.Series(data=df['retweets'].values, index=df['date'])

# Likes vs retweets visualization:
tfav.plot(figsize=(32,8), label="Likes", legend=True)
tret.plot(figsize=(32,8), label="Retweets", legend=True)
```

Figure 4. 11 Code for plotting a time-series plot

A time series plot is obtained by having Time(period) on X-axis and number of likes and Retweets on Y-axis.

```
#function to extract hashtags from every tweet
def hashtag_extract(tweets):
    hashtags = []
    # Loop over the words in the tweet
    for tweet in tweets:
        ht = re.findall(r"#(\w+)", tweet)
        hashtags.append(ht)

    return hashtags
```

Figure 4. 12 Code for extracting Hashtags from tweets.

Creating a 'hastag_extract' method that extracts all the words that are hashtagged in each tweet , this is being done using python 'Regex'- used for describing a search pattern in a string (Here finding for words that are processed by the symbol '#')and will append the obtained hashtages to the list 'hashtages'.

```
HT_regular = hashtag_extract(df['text'])

# unnesting list
HT_unnested = sum(HT_regular,[])
```

```
a = nltk.FreqDist(HT_unnested)
d = pd.DataFrame({'Hashtag': list(a.keys()),
                  'Count': list(a.values())})
# selecting top 10 most frequent hashtags
d = d.nlargest(columns="Count", n = 10)
plt.figure(figsize=(36,15))
ax = sns.barplot(data=d, x= "Hashtag", y = "Count")
ax.set(ylabel = 'Count')
plt.show()
```

Figure 4. 13 Code for plotting a Bar graph

Calling the 'hashtag_extract ' on 'df' will find the hashtagged words , using FreqDist method from nltk(natural language toolkit) package for frequency distribution , which actually records the number of times the 'hashtagged' word has encountered. 'barplot' is a method in 'seaborn' package, used to obtain bar graph. Hashtagged words are taken on X-axis and Count on Y-axis.

```
df['likes'].corr(df['retweets'])
```

```
0.9639484930965402
```

```
df.plot.scatter(x='likes',y='retweets',c='r')
```

Figure 4. 14 Code for plotting a Scatter plot

Finding Correlation between likes and retweets on the tweet. A scatter plotted, Likes along X-axis and Retweets along Y-axis. Notice that it is positively corelated, means number of retweets increases as likes increases.

```
# check the number of positive vs. negative tagged sentences
positives = df['category'][df.category == 1]
negatives = df['category'][df.category == -1]
neutrals = df['category'][df.category == 0]


print('number of positve categorized text is:  {}'.format(len(positives)))
print('number of negative categorized text is: {}'.format(len(negatives)))
print('number of neutral categorized text is: {}'.format(len(neutrals)))
print('total length of the data is:            {}'.format(df.shape[0]))
```

```
number of positve categorized sentences is:  72250
number of negative categorized sentences is: 35510
number of neutral categorized sentences is: 55213
total length of the data is:            162973
```

Figure 4. 15 Evaluation of categorized data

Sentiment Data Visualization

The Positive, negative, and neutral tweets are classified using 'Category' method. 1 for positive, -1 for negative and 0 for neutral tweet. All the tweets are classified, and the number of positive, negative, and neutral tweets count are printed.

```
#import matplotlib.pyplot as plt

slices_len = [len(positives), len(negatives), len(neutrals)]
category = ['positives', 'negatives', 'neutrals']
colors = ['r', 'g', 'b']

plt.pie(slices_len, labels=category, colors=colors, startangle=90, autopct='%.1f%%')
plt.show()
```

Figure 4. 16 Code for plotting a Pie chart

Pie chart is obtained for the categorized data. 'Red' represents Positive tweets, 'Green' for Negatives and 'Blue' for Neutral. Pie chart is support by 'matplotlib' using method 'pie'. Each tweet is tokenized into words and graph is plotted that shows the Word Rank.

```
# plot word frequency distribution of first few words without narendra, modi
# no_modi
no_modi = []
for ls in df['no_modi']:
    words = [w for w in ls]
    for word in words:
        no_modi.append(word)
plt.figure(figsize=(12,5))
plt.xticks(fontsize=13, rotation=90)
fd = nltk.FreqDist(no_modi)
fd.plot(25,cumulative=False)

# log-log  of all words -- no_modi
word_counts = sorted(Counter(no_modi).values(), reverse=True)
plt.figure(figsize=(12,5))
plt.loglog(word_counts, linestyle='-', linewidth=1.5)
plt.ylabel("Freq")
plt.xlabel("Word Rank")
```

Figure 4. 17 Code for plotting a line graph

Like earlier 'nltk' is used for frequency distribution, words along X-axis and their count along Y-axis. Word Rank plot gives the maximum number of times the respective word is used in all the tweets.

Word Cloud Generation:

```
# split sentences to get individual words
all_words = []

for line in df['no_modi']: # try 'tokens'
    all_words.extend(line)

# create a word frequency dictionary
wordfreq = Counter(all_words)
# draw a Word Cloud with word frequencies
wordcloud = WordCloud(width=900,
                      height=500,
                      max_words=50,
                      max_font_size=100,
                      relative_scaling=0.5,
                      colormap='Blues',
                      normalize_plurals=True).generate_from_frequencies(wordfreq)
plt.figure(figsize=(17,14))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Figure 4. 18 Creation of Word Cloud

Word cloud is an image composed of words, in which the size of each word indicates its frequency. 'Wordcloud' is a python package that generates a cloud of words, using 'Wordcloud' method. This method takes height, width, words, color(here Blue) as

parameters. ‘imshow’ is used to display the Wordcloud. The Wordcloud is generated for the Negative and positive categorized tweets and comments as well.

Visualization of Geo Spatial data

```
col_names=['bbox_cord','city','country']

df_geo = pd.read_csv('modi_geo_data.csv', names=col_names)

df_geo.head()
```

	bbox_cord	city	country
0	[[[80.397161, 16.283456], [80.843816, 16.28345...	Vijayawada, India	India
1	[[[82.671054, 25.166781], [83.194857, 25.16678...	Varanasi, India	India
2	[[[76.84252, 28.397657], [77.347652, 28.397657...	New Delhi, India	India
3	[[[73.6246953, 18.2801029], [74.1954594, 18.28...	Haveli, India	India
4	[[[79.380325, 28.959342], [79.422686, 28.95934...	Rudrapur, India	India

```
df_geo.shape
#df_geo = df_geo.drop_duplicates()
```

```
(8317, 3)
```

Figure 4. 19 Representation of stored Geo-data in Pandas Dataframe

A separate csv file is made that stores the Bbox_coordinates , city and county from which tweets were made. A total of ‘8317’(rows) locations were determined.

```
import numpy as np
import json
def geo_mean(x):
    y = json.loads(x)
    y = np.asarray(y)
    mean_geoloc = np.add(np.add((y[:][0][0]),(y[:][0][1]))/2,\
                         (np.add((y[:][0][2]),(y[:][0][3]))/2))/2
    return np.around(mean_geoloc,decimals=6)

# append new column  and clean up df
df_geo['geo_code'] = df_geo['bbox_cord'].apply(geo_mean)
df_geo = df_geo.drop(['bbox_cord'],axis=1)
df_geo.head()
```

	city	country	geo_code
0	Vijayawada, India	India	[80.620488, 16.610865]
1	Varanasi, India	India	[82.932955, 25.373376]
2	New Delhi, India	India	[77.095086, 28.63849]
3	Haveli, India	India	[73.910077, 18.508161]
4	Rudrapur, India	India	[79.401506, 28.974728]

Figure 4. 20 Determination of Longitude and Latitude coordinates

For each Bbox_coordinates , mean is obtained that gives the ‘Geocode’ This Geocode composed of longitude and latitude.

```
# get latitudes and longitudes

# some helper funtions to get longs and lats
def lats(x):
    return x[1]

def longs(x):
    return x[0]

# ----------------------------------------------------------#
# append longs and lats to dframe
df_geo ['latitude'] = df_geo ['geo_code'].apply(lats)
df_geo ['longitude'] = df_geo ['geo_code'].apply(longs)
df_geo.head()
```

	city	country	geo_code	latitude	longitude
0	Vijayawada, India	India	[80.620488, 16.610865]	16.610865	80.620488
1	Varanasi, India	India	[82.932955, 25.373376]	25.373376	82.932955
2	New Delhi, India	India	[77.095086, 28.63849]	28.638490	77.095086
3	Haveli, India	India	[73.910077, 18.508161]	18.508161	73.910077
4	Rudrapur, India	India	[79.401506, 28.974728]	28.974728	79.401506

Figure 4. 21 Separation of coordinates

Longitude and latitude are separated from the list using defined methods ‘lats’ and ‘longs.
Now that the data is all prepped, we can make some maps with Folium.

```
from IPython.display import IFrame
from folium import plugins

# use the folium library to create all tweet origins in the dataset on map of India
#df_india = df_geo[df_geo.country=='India']
geoplots = []

for index, row in df_geo[['latitude','longitude']].iterrows():
    geoplots.append([row['latitude'],row['longitude']])
mus = folium.Map(location=[39, -99], zoom_start=4)

plugins.Fullscreen(
    position='topright',
    title='Expand me',
    title_cancel='Exit me',
    force_separate_button=True).add_to(mus)

mus.choropleth(
    geo_data='india_states.geojson',
    fill_color='blue',
    fill_opacity=0.1,
    line_opacity=0.2,
    name='india_states')

mus.add_child(plugins.HeatMap(geoplots,
                              name='Twitter HeatMap',
                              radius=10,
                              max_zoom=1,
                              blur=10,
                              max_val=3.0))

folium.TileLayer('cartodbpositron').add_to(mus)
folium.TileLayer('cartodbdark_matter').add_to(mus)
folium.TileLayer('Mapbox Control Room').add_to(mus)
folium.LayerControl().add_to(mus)
mus.save("twitter_india_map.html")
IFrame('twitter_india_map.html', width=960, height=520)
```

Figure 4. 22 Code for drawing a Heat Map

'Folium' package is used for drawing Heatmaps , longitude and latitude are extracted by iterating through the rows using iterons and is appended into 'geoplots' list. Create a map object 'mus' centered around the area of interest, also set them zoom level to 4 to see the individual points . Choropleth map made using the Folium library, need elements to build a chloropleth map. A **shape file** in the **geojson** format: it gives the **boundaries** of every zone that you want to represent. A **data frame** that gives the **values** of each zone, a color(blue). Heatmap is generated using 'Heatmap' method that takes, Geocode as a parameter. Twitter API has a feature of sharing location with the tweet, so we were able to obtain the location.

Pipelining

MLlib standardizes APIs for machine learning algorithms to make it easier to combine multiple algorithms into a single pipeline, or workflow. The ML Pipelines is a High-Level API for MLlib that lives under the spark.ml package. A pipeline consists of a sequence of stages.

Pipeline components

Transformers

A Transformer is an abstraction that includes feature transformers and learned models. Technically, a Transformer implements a method transform (), which converts one Data Frame into another, generally by appending one or more columns

4.5.2.2 Estimators

An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data. Technically, an Estimator implements a method fit (), which accepts a Data Frame and produces a Model, which is a Transformer. For example, a learning algorithm such as Logistic Regression

is an Estimator, and calling fit () trains a Logistic Regression Model, which is a Model and hence a Transformer.

Pipeline working Flow

In machine learning, it is common to run a sequence of algorithms to process and learn from data. E.g., a simple text document processing workflow might include several stages:

Split each document's text into words.

Convert each document's words into a numerical feature vector.

Learn a prediction model using the feature vectors and labels

MLlib represents such a workflow as a Pipeline, which consists of a sequence of Pipeline Stages (Transformers and Estimators) to be run in a specific order.

Flow Diagram

A Pipeline is specified as a sequence of stages, and each stage is either Transformer or an Estimator. These stages are run in order, and the input Data Frame is transformed as it passes through each stage. For Transformer stages, the transform () method is called on the Data Frame. For Estimator stages, the fit() method is called to produce a Transformer (which becomes part of the Pipeline Model, or fitted Pipeline), and that Transformer's transform() method is called on the Data Frame.

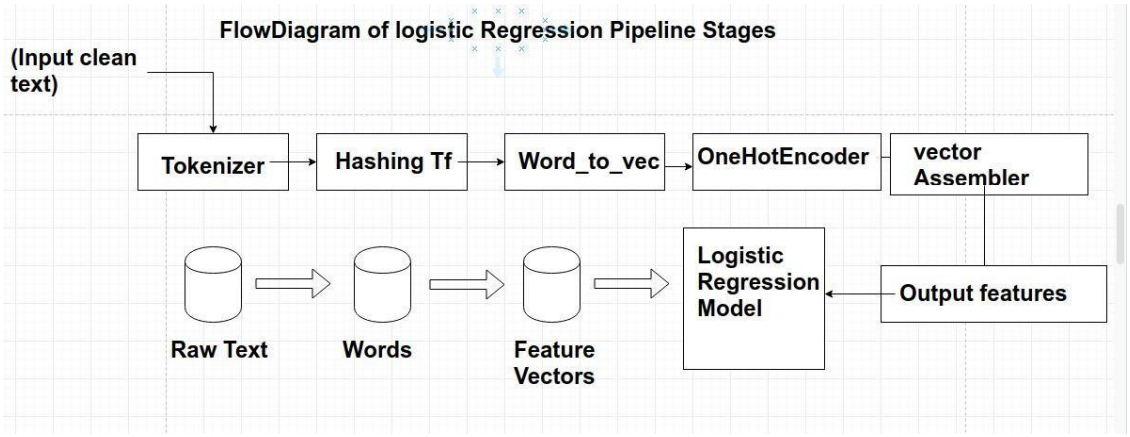


Figure 4.23: Typical Pipeline Flow Diagram for a logistic Regression Model

Above, the top row represents a Pipeline with three stages. The first two (Tokenizer and HashingTF) are Transformers, and the Logistic Regression Model is an Estimator. The bottom row represents data flowing through the pipeline, where cylinders indicate Data Frames. The Pipeline. Fit () method is called on the original Data Frame, which has raw text documents and labels. The Tokenizer. Transform () method splits the raw text documents into words, adding a new column with words to the Data Frame. The HashingTF. transform() method converts the words column into feature vectors, adding a new column with those vectors to the Data Frame. Now, since Logistic Regression is an Estimator, the Pipeline first calls Logistic Regression. fit() to produce a Logistic Regression Model. If the Pipeline had more Estimators, it would call the Logistic Regression Model's transform () method on the Data Frame before passing the data Frame to the next stage.

Machine Learning Models

Logistic Regression

Logistic regression is a popular method to predict a categorical response. It is a special case of Generalized Linear models that predicts the probability of the outcomes. In spark.ml logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial

logistic regression which can be changed by Using the family parameter to select between these two algorithms or leave it unset and Spark will infer the correct variant.
Logistic Regression for Multiclass Classification:
Multiclass classification is supported via multinomial logistic (SoftMax) regression. In multinomial logistic regression, the algorithm produces K sets of coefficients, or a matrix of dimension K×J where K is the number of outcome classes and J is the number of features. If the algorithm is fit with an intercept term, then a length K vector of intercepts is available.
The conditional probabilities of the outcome classes k∈1,2,…,K are modelled using the SoftMax function.

We minimize the weighted negative log-likelihood, using a multinomial response model, with elastic-net penalty

$$P(Y = k|\mathbf{X}, \boldsymbol{\beta}_k, \beta_{0k}) = \frac{e^{\boldsymbol{\beta}_k \cdot \mathbf{X} + \beta_{0k}}}{\sum_{k'=0}^{K-1} e^{\boldsymbol{\beta}_{k'} \cdot \mathbf{X} + \beta_{0k'}}}$$

to control for overfitting.

$$\min_{\boldsymbol{\beta}, \beta_0} - \left[\sum_{i=1}^{L} w_i \cdot \log P(Y = y_i|\mathbf{x}_i) \right] + \lambda \left[\frac{1}{2}(1-\alpha) \, ||\boldsymbol{\beta}||_2^2 + \alpha ||\boldsymbol{\beta}||_1 \right]$$

When using multinomial logistic regression, one category of the dependent variable is chosen as the reference category. Separate odds ratios are determined for all independent variables for each category of the dependent variable with the exception of the reference category, which is omitted from the analysis. The exponential beta coefficient represents the change in the odds of the dependent variable being in a particular category vis-a-vis the reference category, associated with a one-unit change of the corresponding independent variable.

Random Forest
Random forests are ensembles of decision trees. Random forests are one of the most successful machine learning models for classification and regression. They combine

many decision trees in order to reduce the risk of overfitting. Like decision trees, random forests handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.
spark.mllib supports random forests for binary and multiclass classification and for regression, using both continuous and categorical features. spark.mllib implements random forests using the existing decision tree implementation. Please see the decision tree guide for more information on trees.
The random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + \dots$$

where the final model g is the sum of simple base models fi. Here, each base classifier is a simple decision tree.
This broad technique of using multiple models to obtain better predictive performance is called model ensemble.
In random forests, all the base models are constructed independently using a different subsample of the data.
Basic Working of a Random Forest Algorithm:
Random forests train a set of decision trees separately, so the training can be done in parallel. The algorithm injects randomness into the training process so that each decision tree is a bit different. Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data.

Training

The randomness injected into the training process includes:
Subsampling the original dataset on each iteration to get a different training set (a.k.a. bootstrapping).
Considering different random subsets of features to split on at each tree node.

Apart from these randomizations, decision tree training is done in the same way as for individual decision trees.

Prediction

To make a prediction on a new instance, a random forest must aggregate the predictions from its set of decision trees. This aggregation is done differently for classification and regression.
Classification: Majority vote. Each tree's prediction is counted as a vote for one class. The label is predicted to be the class which receives the most votes.
Regression: Averaging. Each tree predicts a real value. The label is predicted to be the average of the tree predictions.

Naïve Bayes

Naive Bayes is a simple multiclass classification algorithm with the assumption of independence between every pair of features. Naive Bayes can be trained very efficiently. Within a single pass to the training data, it computes the conditional probability distribution of each feature given label, and then it applies Bayes' theorem to compute the conditional probability distribution of label given an observation and use it for prediction.
spark.mllib supports multinomial naive Bayes and Bernoulli naive Bayes. These models are typically used for document classification. Within that context, each observation is a document and each feature represent a term whose value is the frequency of the term (in multinomial naive Bayes) or a zero or one indicating whether the term was found in the document (in Bernoulli naive Bayes). Feature values must be nonnegative. The model type is selected with an optional parameter "multinomial" or "Bernoulli" with "multinomial" as the default. Additive smoothing can be used by setting the parameter λ (default to 1.0). For document classification, the input feature vectors are usually sparse, and sparse vectors should be supplied as input to take advantage of sparsity. Since the

training data is only used once, it is not necessary to cache it. An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification.
Principle of Naive Bayes Classifier:
A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.
Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

One-Vs-Rest Classifier

One-vs-the-rest (OvR) multiclass/multilabel strategy

Also known as one-vs-all, this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n_classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice.
One-Vs-Rest is an example of a machine learning reduction for performing multiclass classification given a base classifier that can perform binary classification efficiently. It is also known as "One-vs-All."
One-Vs-Rest is implemented as an Estimator. For the base classifier it takes instances of Classifier and creates a binary classification problem for each of the k classes. The classifier

for class I is trained to predict whether the label is I or not, distinguishing class I from all other classes. Predictions are done by evaluating each binary classifier and the index of the most confident classifier is output as label.
In pseudocode, the training algorithm for an OvA learner constructed from a binary classification learner L is as follows:
Inputs:
L, a learner (training algorithm for binary classifiers)
samples X
labels y where yi ∈ {1, … K} is the label for the sample Xi

Outputs:
a list of classifiers fk for k ∈ {1, …, K}

Procedure:
For each *k* in {1, …,*K*}
Construct a new label vector z where zi = yi if yi = k and zi = 0 otherwise
Apply L to X, z to obtain fk

Making decisions means applying all classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score:

$$\hat{y} = \underset{k \in \{1 \dots K\}}{\operatorname{argmax}} \; f_k(x)$$

Although this strategy is popular, it is a heuristic that suffers from several problems. Firstly, the scale of the confidence values may differ between the binary classifiers. Second, even if the class distribution is balanced in the training set, the binary classification learners see unbalanced distributions because typically the set of negatives they see is much larger than the set of positives.

Results

Preamble to the Chapter

Data Visualization Results

```
The tweet with more likes is:
Congress led UPA
Surgical Strike : Don't do it
Air Strike: Don't do it
A-SAT Missile: Don't do it

Modi Sarkar
Surgical Strike: Go For It
Air Strike: Go For It
A-SAT Missile: Go For It

Modi Hai To Mumkin Hai. #MissionShakti
Number of likes: 43433

The tweet with more retweets is:
Congress led UPA
Surgical Strike : Don't do it
Air Strike: Don't do it
A-SAT Missile: Don't do it

Modi Sarkar
Surgical Strike: Go For It
Air Strike: Go For It
A-SAT Missile: Go For It

Modi Hai To Mumkin Hai. #MissionShakti
Number of retweets: 13829
```

Figure 5.1 Statistical analysis of twitter Dataset

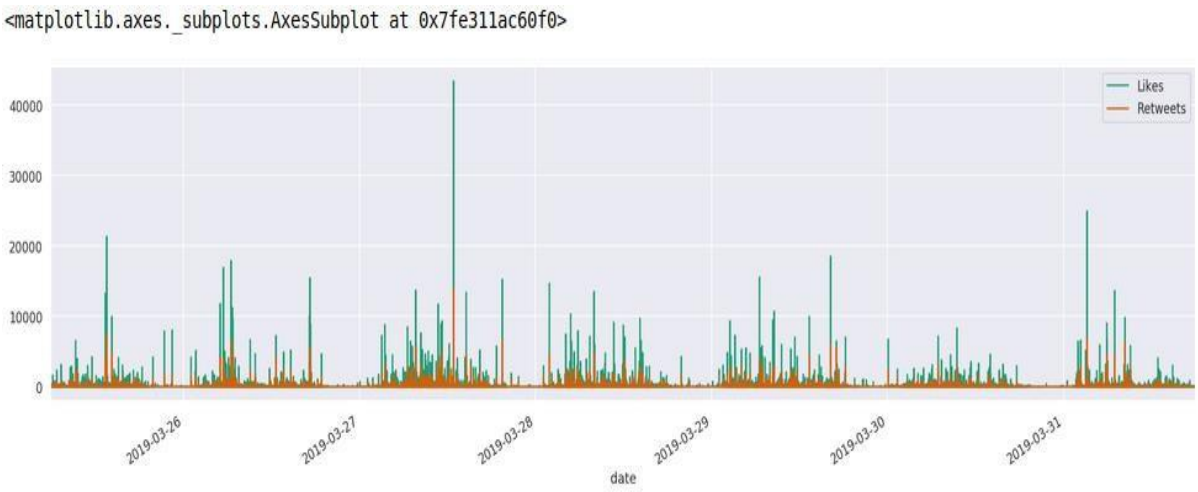


Figure 5.2 Time-series plot for Twitter

A time series plot obtained shows the correlation between likes and retweets , it is plotted from period ‘26-03-2019’ to ‘31-03-2019’.

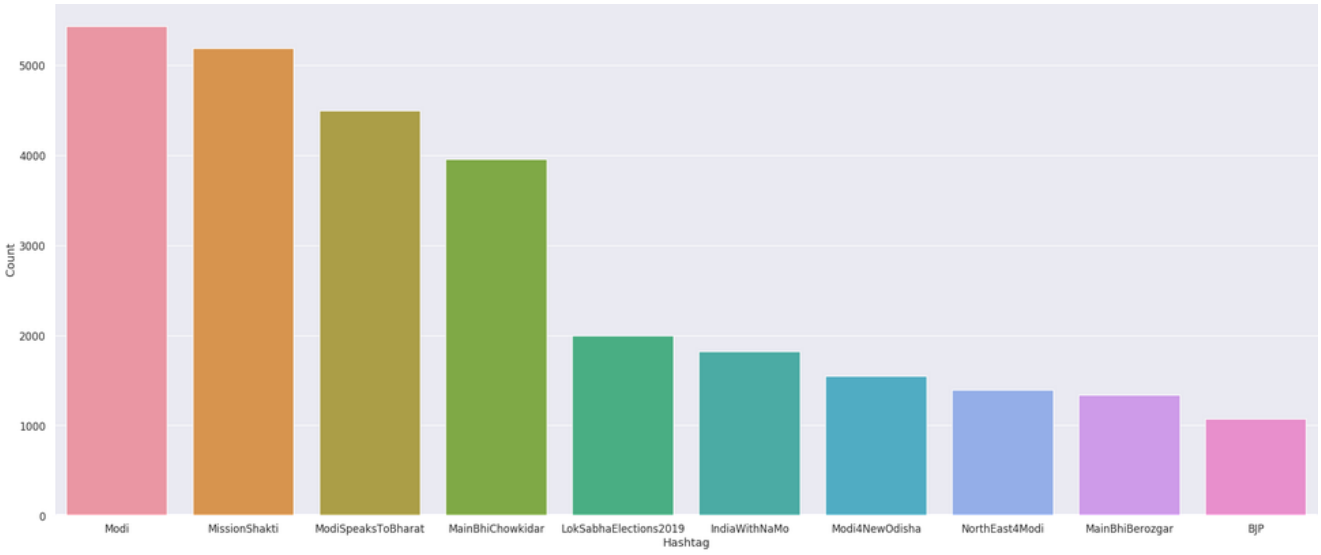


Figure 5.3 Bar chart that depicts the maximum usage of Hashtags.

A bar chart shows the Hashtagged word along with-it frequency that are being hashtagged in tweets.

<matplotlib.axes._subplots.AxesSubplot at 0x7ff221fd4cc0>

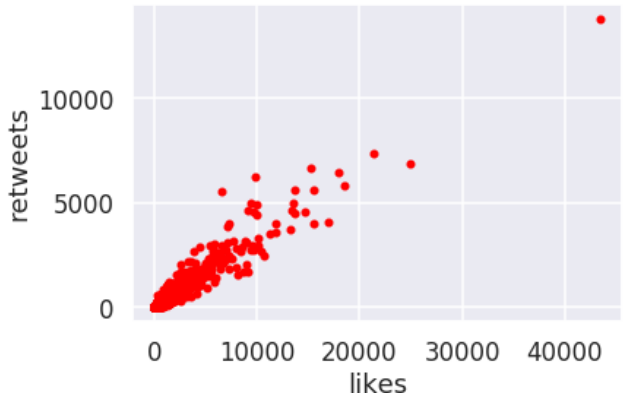


Figure 5.4 Scatter plot

The above scatter plot shows correlation plot between likes a and retweets of the tweet. Notice that as like increases, retweets also increase.

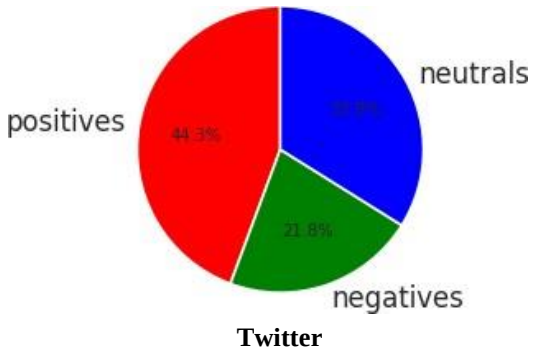


Figure 5.5 Pie chart for Categorized data of Twitter

The above pie chart that depicts the percentage of positive, negative, and neutral tweets and comments on Twitter

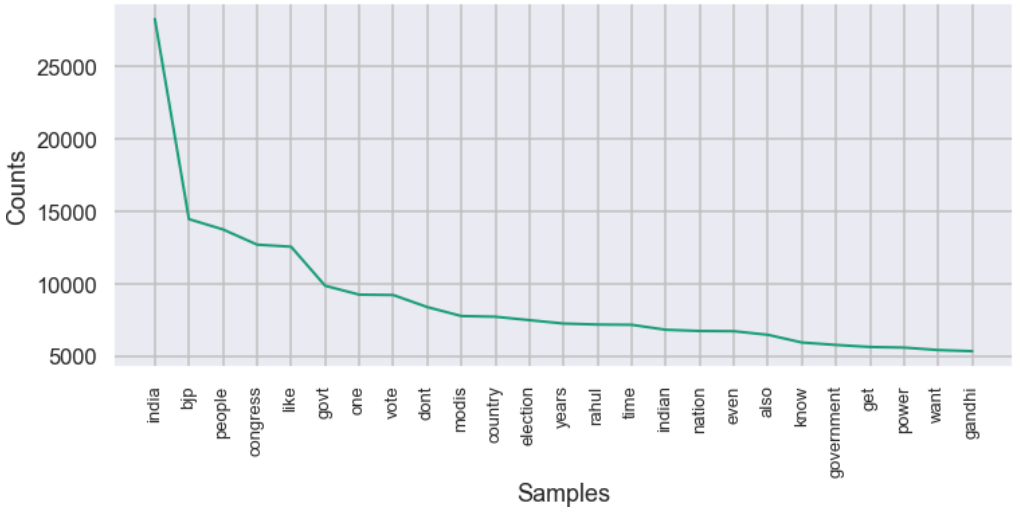
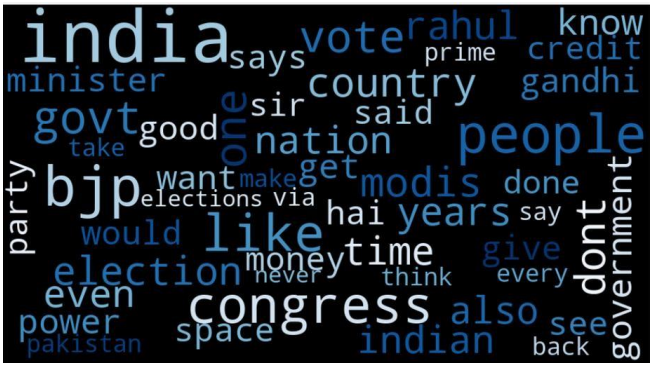


Figure 5.6-line plot that shows Word Rank

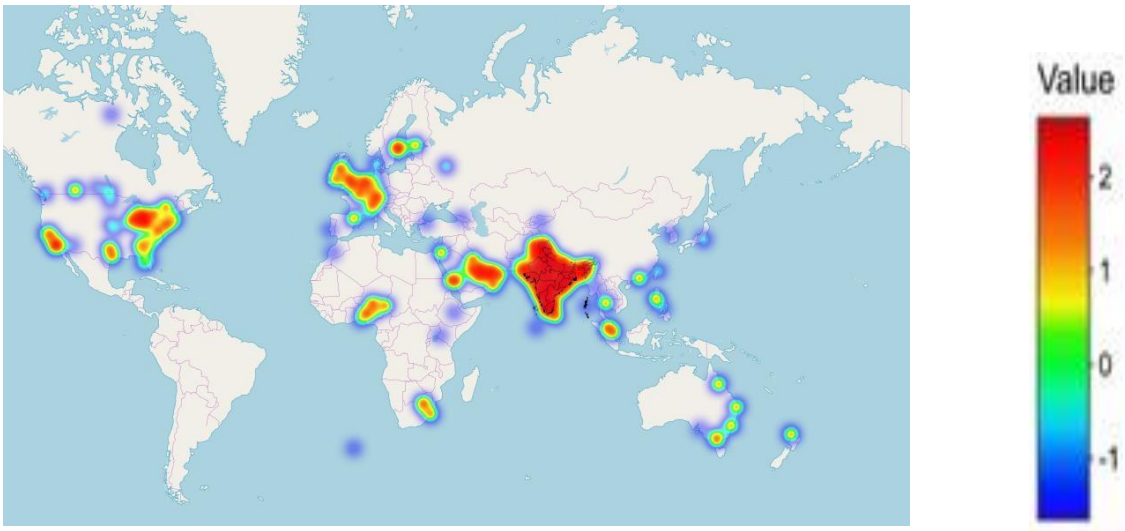
Line plot that tells the number of times(frequency) a particular word is used in tweets.
'India' is used more than 25k times in tweets that are collected.



Twitter

Figure 5.7 Word Cloud for Twitter

The above Word cloud depicts , 'India' is used highest number of times in the tweets , and color intensity indicates how much positively that word is used.



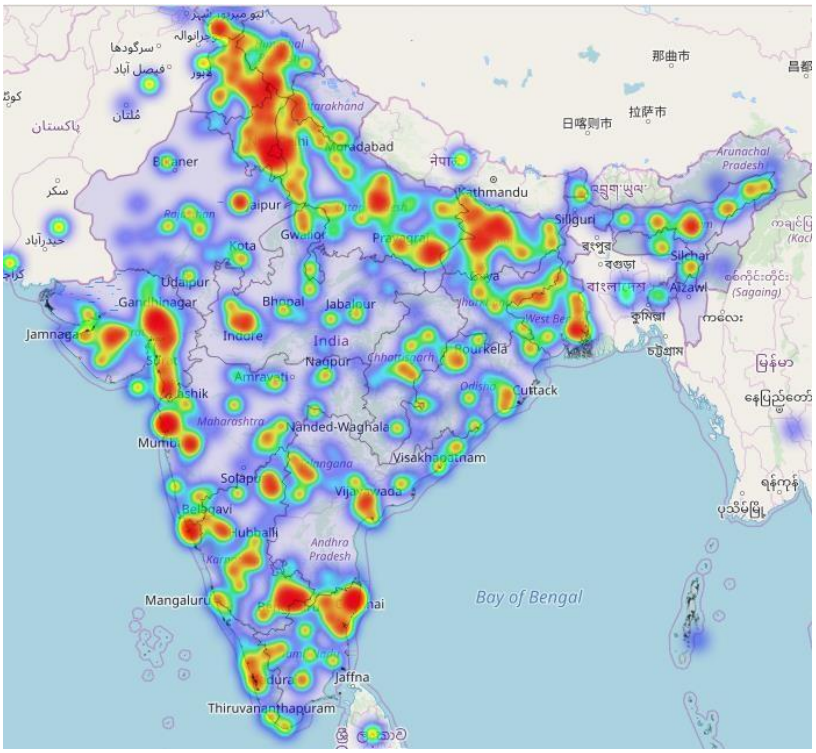


Figure 5.8 A Global and India Heat Map

Global and India heat signatures red being the highest as shown in scale, indicates the number of tweets coming from that location is higher, and blue the least.

Analysis of Machine Learning Classification Models by results Overview and goal

The question of which model type to apply to a Machine Learning task can be a daunting one given the immense number of algorithms available in the literature. It can be difficult to compare the relative merits of machine learning model methods, as one can outperform the other in a certain class of problems while consistently coming in behind for another class. So, in this project four of such Machine Learning Models were Applied to yield the better results in terms of **Evaluation Metrics.**

The four Machine Learning Classification Models were applied and their Evaluation Metrics Such as the Prediction of the accuracy of the Models will be compared.

In this project, we'll explore the differences between and the advantages of all four Machine Learning Models for the classification problems and try to highlight scenarios where one might be recommended over the other.
Advantages and the Disadvantages of Machine Learning Models

In the context of low-dimensional data (i.e., when the number of covariates is small compared to the sample size), logistic regression is considered a standard approach for Multiclass classification. This is especially true in scientific fields such as medicine or psycho-social sciences where the focus is not only on prediction but also on explanation. Since its invention 17 years ago, the random forest (RF) prediction algorithm, which focuses on prediction rather than explanation, has strongly gained popularity and is increasingly becoming a common "standard tool" also used by scientists without any strong background in statistics or machine learning.

Our experience as authors, reviewers and readers is that random forest can now be used routinely in many scientific fields without justification and without the audience strongly questioning this choice. While its use was in the early years limited to innovation-friendly scientists interested (or experts) in machine learning, random forests are now increasingly well-known in various non-computational communities.

The Random Forest (RF) algorithm for regression and classification has considerably gained popularity since its introduction in 2001. Meanwhile, it has grown to a standard classification approach competing with logistic regression in many innovation-friendly scientific fields.

Logistic Regression and trees differ in the way that they generate decision boundaries i.e., the lines that are drawn to separate different classes.

Decision Trees bisect the space into smaller and smaller regions, whereas Logistic Regression fits a single line to divide the space exactly into two. Of course, for higher- dimensional data, these lines would generalize to planes and hyperplanes. A single linear boundary can sometimes be limiting for Logistic Regression.

The problem of logistic regression being hard to interpret is much more serious than it first appears. As most people are not able to interpret it correctly, they end up not even noticing when they have stuffed it up, leadin to a double boo-boo, whereby they inadvertently create a model that is rubbish, which they then go on to misinterpret. The great thing about decision trees is that they are as simple as they appear. No advanced statis cal knowledge is required to use them or interpret them correctly.

Advantages of the Decision Tree Over the Other three Machine Learning algorithms
Decision trees implicitly perform variable screening or feature selection

Decision trees require relatively little effort from users for data preparation

Nonlinear relationships between parameters do not affect tree performance

The best feature of using trees for analytics - easy to interpret and explain to executives!

Logistic regression will work better if there is a single decision boundary, not necessarily parallel to the axis.

Decision trees can be applied to situations where there is not just one underlying decision boundary, but man , and will work best if the class labels roughly lie in hyper-rectangular regions.

Logistic regression is intrinsically simple, it has low variance and so is less prone to over-fitting.

Decision trees can be scaled up to be very complex, are more liable to over-fit. Pruning is applied to avoid th s.

Pros and cons of the Naïve Bayes Model Pros:
It is easy and fast to predict class of test data set. It also performs well in multi class prediction
When assumption of independence holds, a Naive Bayes classifier performs better compare to other models li e logistic regression and you need less training data.

It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.

Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Project Costing

Preamble to the Chapter

This chapter deals with project costing for designed project. Project Cost is the total funds needed to complete the project or work that consists of a Direct Cost and Indirect Cost. Now in this chapter we estimate the total cost to design and implement the project.

Total Manhours

Manhour required per person in week: 12 hrs per week / person
Manhours required per person for 16 weeks: 12*16=160 hrs/person
Total Man Hours required for project=160*4=640 hrs.

Cost Estimation

Component 1
Average pay per for entry level graduate = 6000 per month
For 16 weeks or 4 months = 24,000 Rs per person
Laptop of 20,000 Rs
Total cost required per person =24000+20000=44000 Rs /person
Total cost foe project=44000*4=1,16,000 Rs Component 2
Miscellaneous cost=4000 per month
For 16 weeks or 4 months = 16000 Rs

Final Total cost required for project=1,16,000 +16,000=**1,32,000 Rs**

Conclusions and Suggestions for Future Work

Preamble to the Chapter

This chapter mainly consists of conclusions to the entire project work summarizing all the goals and the efforts made towards achieving them. The second part of this chapter lists few exciting future works that this project encourages interested readers to carry out.

Conclusion

In this project, we present a large-scale benchmarking experiment based on the datasets containing the tweets and the comments of the people on the 2019 Indian General Elections and finally we compare the prediction performance of the original version of Random Forest with default parameters and Logistic Regression, Decision Trees and One- Vs-Rest classifier as Multiclass classification tools.

Comparison of accuracy of all the Machine Learning Models

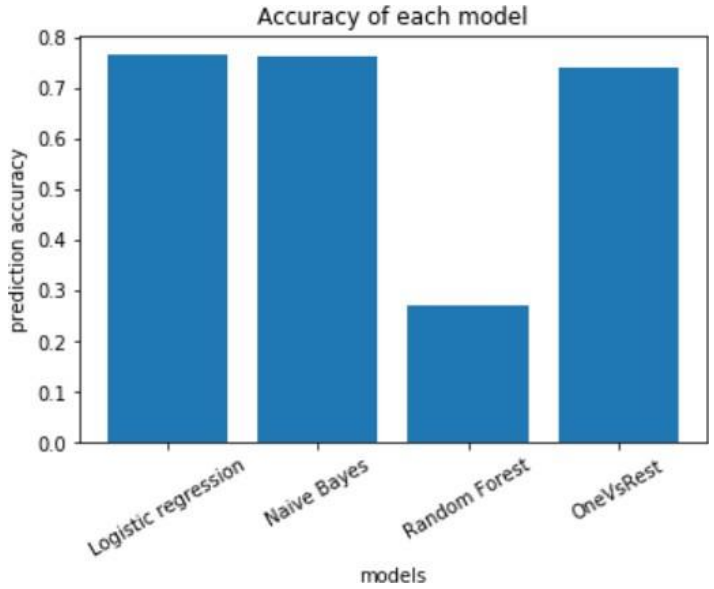


Figure 7.1: Bar graph for accuracy of sentiment prediction of each trained model

Around 10 lakh tweets were combined using the pandas Library and all those combined tweets and the comments was further stored in the Spark data frames after.
Machine Learning Models were Applied on those Spark data frames using the spark Mllib Library and the accuracy of each Model was Predicted as Shown in the above figure.
Logistic Regression and Naïve Bayes Machine Learning algorithms performed better than One-Vs-Rest and the Random Forest according to the considered accuracy measured in approximately 80% of the datasets.

Future Work

This project successfully collected data from twitter social media to analyze sentiments and learn about the political stand of the people, a better picture can be drawn if more data is collected from other social websites.
The models trained to predict sentiments are trained by combining text twitter tweets, going further a more careful inspection can be done on these texts to combine them in a better way to gain more precise results.
The results shown for analysis of tweets are only for data gathered during a short period of time i.e., roughly for a duration of one week before commencement of elections, further improvement can be done by gathering more data during different intervals of time during the past of year or since the last general elections.
The data used in this project is historical which is collected well before beginning any analysis of training, a future work can real time analysis of data from these social media sites by using spark streaming feature of the spark tool.
This project implements contemporary machine learning techniques for sentiment classification of text, we encourage implementing deep learning techniques to do the same and compare the results.

References

Pak, A. &. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. LREC

Wong, F. M. (2013). Quantifying Political Leaning from Tweets and Retweets. ICWSM. [3]Boutet, A. K. (201). What is in your Tweets? I know who you supported in the UK 2010 general election. Proceedings of the Internati nal AAAI Conference on Weblogs and social media.

Golbeck, J. &. (2011). Computing political preference among twitter followers. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Pennacchiotti, M. &. (2011). Democrats, republicans, and Starbucks afficionados: user classification in twitte . Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining , 430-438.

Tumasjan, A. S. (2010). Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment. ICWSM, 10, 178-185.

O'Connor, B. B. (2010). From tweets to polls: Linking text sentiment to public opinion time series. ICWSM, 11, 122-129

Gayo-Avello, D. M. (2011). Limits of electoral predictions using twitter. ICWSM.

Bermingham, A. &. (2011). On using Twitter to monitor political sentiment and predict election results.

Ceron, A. C. (2014). Using Sentiment Analysis to Monitor Electoral Campaigns:
Method

Ceron, A. C. (2013). Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. New Media & Society, 16(2), 340-358

Non periodical web document, Dr. Neeti Kasliwal,S. Taruna, Sentiment Analysis- Strategy for Text Pre-Processing, May 11,2019 from https://www.academia.edu/36436742/Sentiment_Analysis-Strategy_for_Text_Pre-Processing

Non periodical web document, Avinash Navlani, Text Analytics for Beginners using NLTK, May 11,2019 from https://www.datacamp.com/community/tutorials/text-analytics-beginners-nltk