SIX WEEKS SUMMER TRAINING REPORT

on

AI WITH IBM WATSON

Submitted by

Pranjal Bhalla

Registration No: 12016264

Programme Name: Btech. CSE With Specialization in AI/ML (3rd Year)

School of Computer Science & Engineering

Lovely Professional University, Phagwara

(June-July,2022)

DECLARATION

I hereby declare that I have completed my Six weeks summer training at ALL SOFT SOLUTIONS platform in AI with IBM Watson from May 12,2022 to July 20,2022. I declare that I have worked full dedication during their 6 weeks of training and my learning outcomes fulfil the requirements of training for the award of degree of B.tech. CSE , Lovely Professional University, Phagwara.

Date – 16 Sept. 2022

Name of Student –

Pranjal Bhalla

Registration no:

12016264

ACKNOWLEDGEMENTS

I would like to express my gratitude towards my university as well as IBM ALL SOFT SOLUTIONS for providing me the golden opportunity to do this wonderful summer training regarding Artificial Intelligence which also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things.
So, I am really thankful to them.

Moreover, I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am thankful to have such a good support of them as they always have my back whenever I need.

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Summer Training Certificate

Allsoft Solutions
An ISO 9001 : 2008 Certified Company

PROJECT COMPLETION CERTIFICATE

In recognition of the commitment to achieve professional excellence this is to certify that Ms./Mr.

Pranjal Bhalla

has successfully completed an Industry-oriented project.

Project Name	Sentiment Analysis
Technologies Used	Machine learning and NLP
Reference No.	AIP/CEP2022/IN/41585
Training Date	June 2022-July2022
Training Duration	6 Weeks
Training Location	Online Live Mode

Program Co-ordinator
Industry/Academic Alliance

Business Partner

Director
Training and Development
Allsoft Solutions and Services

BIG DATA - ANALYTICS | IoT | ORACLE | J2EE | PHP | CLOUD COMPUTING

Table of Contents

INTRODUCTION

Artificial intelligence (AI) is the ability of machines to replicate or enhance human intellect, such as reasoning and learning from experience. Artificial intelligence has been used in computer programs for years, but it is now applied to many other products and services. For example, some digital cameras can determine what objects are present in an image using artificial intelligence software. In addition, experts predict many more innovative uses for artificial intelligence in the future, including smart electric grids.

AI uses techniques from probability theory, economics, and algorithm design to solve practical problems. In addition, the AI field draws upon computer science, mathematics, psychology, and linguistics. Computer science provides tools for designing and building algorithms, while mathematics offers tools for modeling and solving the resulting optimization problems.

Although the concept of AI has been around since the 19th century, when Alan Turing first proposed an "imitation game" to assess machine intelligence, it only became feasible to achieve in recent decades due to the increased availability of computing power and data to train AI systems.

To understand the idea behind AI, you should think about what distinguishes human intelligence from that of other creatures – our ability to learn from experiences and apply these lessons to new situations. We can do this because of our advanced brainpower; we have more neurons than any animal species.

Today's computers don't match the human biological neural network – not even close. But they have one significant advantage over us: their ability to analyze vast amounts of data and experiences much faster than humans could ever hope.

AI lets you focus on the most critical tasks and make better decisions based on acquired data related to a use case. It can be used for complex tasks, such as predicting maintenance requirements, detecting credit card fraud, and finding the best route for a delivery truck. In other words, AI can automate many business processes leaving you to concentrate on your core business.

Research in the field is concerned with producing machines to automate tasks requiring intelligent behavior. Examples include control, planning and scheduling, the ability to answer diagnostic and consumer questions, handwriting, natural

language processing and perception, speech recognition, and the ability to move and manipulate objects.

History of AI and how it has progressed over the years

With so much attention on modern artificial intelligence, it is easy to forget that the field is not brand new. AI has had a number of different periods, distinguished by whether the focus was on proving logical theorems or trying to mimic human thought via neurology.

Artificial intelligence dates back to the late 1940s when computer pioneers like Alan Turing and John von Neumann first started examining how machines could "think." However, a significant milestone in AI occurred in 1956 when researchers proved that a machine could solve any problem if it were allowed to use an unlimited amount of memory. The result was a program called the General Problem Solver (GPS).

Over the next two decades, research efforts focused on applying artificial intelligence to real-world problems. This development led to expert systems, which allow machines to learn from experience and make predictions based on gathered data. Expert systems aren't as complex as human brains, but they can be trained to identify patterns and make decisions based on that data. They're commonly used in medicine and manufacturing today.

A second major milestone came in 1965 with the development of programs like Shakey the robot and ELIZA, which automated simple conversations between humans and machines. These early programs paved the way for more advanced speech recognition technology, eventually leading to Siri and Alexa.

The initial surge of excitement around artificial intelligence lasted about ten years. It led to significant advances in programming language design, theorem proving, and robotics. But it also provoked a backlash against over-hyped claims that had been made for the field, and funding was cut back sharply around 1974.

After a decade without much progress, interest revived in the late 1980s. This revival was primarily driven by reports that machines were becoming better than humans at "narrow" tasks like playing checkers or chess and advances in computer vision and speech recognition. This time, the emphasis was on building systems that could understand and learn from real-world data with less human intervention.

These developments continued slowly until 1992, when interest began to increase again. First, technological advances in computing power and information storage helped boost interest in research on artificial intelligence. Then, in the mid-1990s, another major boom was driven by considerable advances in computer hardware that had taken place since the early 1980s. The result has been dramatic improvements in performance on several significant benchmark problems, such as image recognition, where machines are now almost as good as humans at some tasks.

The early years of the 21st century were a period of significant progress in artificial intelligence. The first major advance was the development of the self-learning neural network. By 2001, its performance had already surpassed human beings in many specific areas, such as object classification and machine translation. Over the next few years, researchers improved its performance across a range of tasks, thanks to improvements in the underlying technologies.

The second significant advancement in this period was the development of generative model-based reinforcement learning algorithms. Generative models can generate novel examples from a given class, which helps learn complex behaviors from very little data. For example, they can be used to learn how to control a car from only 20 minutes of driving experience.

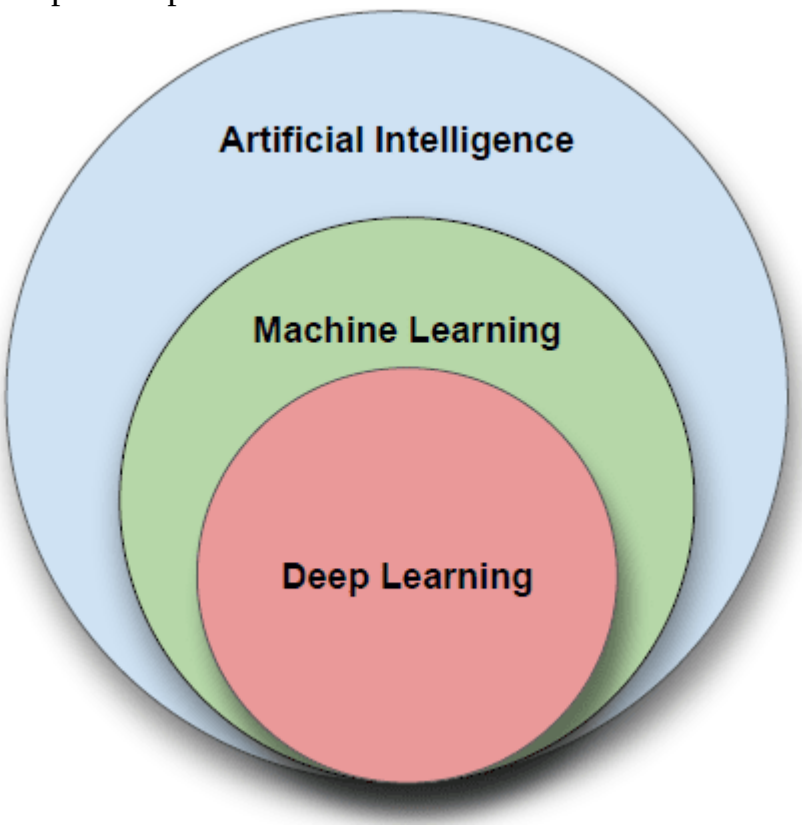
In addition to these two advances, there have been several other significant developments in AI over the past decade. There has been an increasing emphasis on using deep neural networks for computer vision tasks, such as object recognition and scene understanding. There has also been an increased focus on using machine learning tools for natural language processing tasks such as information extraction and question answering. Finally, there has been a growing interest in using these same tools for speech recognition tasks like automatic speech recognition (ASR) and speaker identification (SID).

Different fields under AI to clear common misconceptions

Artificial Intelligence is the most trending field of computer science. However, with all the new technology and research, it’s growing so fast that it can be confusing to understand what is what. Furthermore, there are many different fields within AI,

each one having its specific algorithms. Therefore, it's essential to know that AI is not a single field but a combination of various fields.

Artificial Intelligence (AI) is the general term for being able to make computers do things that require intelligence if done by humans. AI can be broken down into two major fields, Machine Learning (ML) and Neural Networks (NN). Both are subfields under Artificial Intelligence, and each one has its methods and algorithms to help solve problems.



Machine learning

Machine Learning (ML) makes computers learn from data and experience to improve their performance on some tasks or decision-making processes. ML uses statistics and probability theory for this purpose. Machine learning uses algorithms to parse data, learn from it, and make determinations without explicit programming. Machine learning algorithms are often categorized as supervised or unsupervised. Supervised algorithms can apply what has been learned in the past to new data sets; unsupervised algorithms can draw inferences from datasets. Machine learning algorithms are designed to strive to establish linear and non-linear relationships in a given set of data. This feat is achieved by statistical methods used to train the algorithm to classify or predict from a dataset.

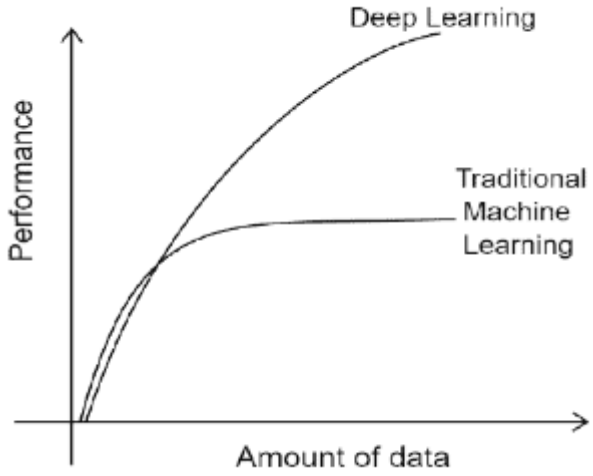
Deep learning

Deep learning is a subset of machine learning that uses multi-layered artificial neural networks to deliver state-of-the-art accuracy in object detection, speech

recognition and language translation. Deep learning is a crucial technology behind driverless cars and enables the machine analysis of large amounts of complex data — for example, recognizing the faces of people who appear in an image or video.

Neural networks

Neural networks are inspired by biological neurons in the human brain and are composed of layers of connected nodes called "neurons" that contain mathematical functions to process incoming data and predict an output value. Artificial neural network learns by example, similarly to how humans learn from our parents, teachers, and peers. They consist of at least three layers: an input layer, hidden layers, and an output layer. Each layer contains nodes (also known as neurons) which have weighted inputs that compute the output.



The performance of traditional machine learning models plateau and throwing any more data doesn't help improve the performance. Deep learning models continue to improve in performance with more data.

These fields have different algorithms, depending on the use case. For example, we have decision trees, random forests, boosting, support vector machines (SVM), k-nearest neighbors (kNN), and others for machine learning. For neural networks, we have convolutional neural networks (CNNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), and more.

However, classifying AI according to its strength and capabilities would mean further subdividing it into "narrow AI" and "general AI." Narrow AI is about getting machines to do one task really well, like image recognition or playing chess. General AI means devices that can do everything humans can do and more. Today's research focuses on narrow AI, but many researchers would like machine learning to eventually achieve general AI.

OVERVIEW OF THE COURSE CONTENT STUDIED

Machine Learning

What is machine learning?

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance**.**

Why is machine learning important?

Machine learning is important because it gives enterprises a view of trends in customer behaviour and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

What are the different types of machine learning?

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The type of algorithm data scientists choose to use depends on what type of data they want to predict.

- Supervised learning: In this type of machine learning, data scientists supply algorithms with labeled training data and define the

variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

- Unsupervised learning: This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.
- Semi-supervised learning: This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.
- Reinforcement learning: Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task

Natural Language Processing

What is natural language processing or NLP?

Natural Language Processing is a method for pre-processing text to turn it into numerical data. That data can then be modeled using Machine Learning algorithms.

NLP is also known as computational linguistics.

NLP is basically feature engineering. This isn't a machine learning algorithm. It’s a way of taking natural text and turning it into something that an algorithm can use. As a data scientist, 80% of your job is being a data janitor and trying to clean things up and turn them into data and features that can be worked with. So natural language processing is a way to process that textual data and turn it into numerical values or categorical values that you can use to actually model text.

How does natural language processing work?

As long as you know how to do supervised and unsupervised learning, natural language processing should be pretty straightforward.

In a supervised learning model, you already have something that's defined eg. important versus spam in your email inbox. If you create a dataset with important and spam messages, then you can determine the words that are associated with important messages and spam messages, so that when a new email comes in, you can find the similarity between that and the previous emails. A huge part of natural language processing is calculating the similarity between different words/datasets.

Another option is to use an unsupervised learning model and do something like topic modeling. Take a textbook, for example. It has a table of contents, which gives a good indication of what the topics are modeled as. When you see the results of topic modeling on the textbook, you'll see that it pretty much puts them in the same topics that you saw in the table of contents, because of so many similar words in this particular document. For all NLP you usually you take a corpus (every document that's contained), plus whatever you want to analyze, and then you try to find the similarities between different subjects in that.

Another method of NLP is tokenization of texts. Let's say that we have the words "natural language processing is." If we do an n-gram, the first column would be the word "natural", the second column would be the word "language", the third one the word "processing". If we did a bi-gram, the first column would be "natural-language", the second "language-processing", the third, "processing-is" and so forth. So it's the way you would turn the sentence into something that counts the frequency of how many times those different terms (or bigrams, or trigrams) or chunks of those terms are actually occurring.

A good example is a spam filter for email. A machine can assume that a message is spam or unimportant message based on the frequency count derived from bodies of text.

Computer Vision

What is computer vision?

Computer vision is the field of computer science that focuses on creating digital systems that can process, analyze, and make sense of visual data (images or videos) in the same way that humans do. The concept of computer vision is based on teaching computers to process an image at a pixel level and understand it. Technically, machines attempt to retrieve visual information, handle it, and interpret results through special software algorithms.

Here are a few common tasks that computer vision systems can be used for:

- Object classification. The system parses visual content and classifies the object on a photo/video to the defined category. For example, the system can find a dog among all objects in the image.
- Object identification. The system parses visual content and identifies a particular object on a photo/video. For example, the system can find a specific dog among the dogs in the image.
- Object tracking. The system processes video finds the object (or objects) that match search criteria and track its movement.

How does computer vision work?

Computer vision technology tends to mimic the way the human brain works. But how does our brain solve visual object recognition? One of the popular hypothesis states that our brains rely on patterns to decode individual objects. This concept is used to create computer vision systems.

Computer vision algorithms that we use today are based on pattern recognition. We train computers on a massive amount of visual data—computers process images, label objects on them, and find patterns in those objects. For example, if we send a

million images of flowers, the computer will analyze them, identify patterns that are similar to all flowers and, at the end of this process, will create a model "flower." As a result, the computer will be able to accurately detect whether a particular image is a flower every time we send them pictures.

Deep Learning

What is Deep Learning?

Deep learning can be considered as a subset of machine learning. It is a field that is based on learning and improving on its own by examining computer algorithms. While machine learning uses simpler concepts, deep learning works with artificial neural networks, which are designed to imitate how humans think and learn. Until recently, neural networks were limited by computing power and thus were limited in complexity. However, advancements in Big Data analytics have permitted larger, sophisticated neural networks, allowing computers to observe, learn, and react to complex situations faster than humans. Deep learning has aided image classification, language translation, speech recognition. It can be used to solve any pattern recognition problem and without human intervention.

Artificial neural networks, comprising many layers, drive deep learning. Deep Neural Networks (DNNs) are such types of networks where each layer can perform complex operations such as representation and abstraction that make sense of images, sound, and text. Considered the fastest-growing field in machine learning, deep learning represents a truly disruptive digital technology, and it is being used by increasingly more companies to create new business models.

How Does Deep Learning Work?

Neural networks are layers of nodes, much like the human brain is made up of neurons. Nodes within individual layers are connected to adjacent layers. The network is said to be deeper based on the number of layers it has. A single neuron in the human brain receives thousands of signals from other neurons. In an artificial neural network, signals travel between nodes and assign corresponding weights. A heavier weighted node will exert more effect on the next layer of nodes. The final layer compiles the weighted inputs to produce an output. Deep learning

systems require powerful hardware because they have a large amount of data being processed and involves several complex mathematical calculations. Even with such advanced hardware, however, training a neural network can take weeks.

uting. These technologies hinge on the ability to recognize patterns then, based on data observed in the past, predict future outcomes. This explains the suggestions, Amazon offers as you shop online or how Netflix knows your penchant for bad 80s movies. Although machines utilizing AI principles are often referred to as “smart,” most of these systems don’t learn on their own; the intervention of human programming is necessary. Data scientists prepare the inputs, selecting the variables to be used for predictive analytics. Deep learning, on the other hand, can do this job automatically.

About The Project:

Introduction:

Sentiment Analysis, as the name suggests, it means to identify the view or emotion behind a situation. It basically means to analyze and find the emotion or intent behind a piece of text or speech or any mode of communication.

We, humans, communicate with each other in a variety of languages, and any language is just a mediator or a way in which we try to express ourselves. And, whatever we say has a sentiment associated with it. It might be positive or negative or it might be neutral as well.
Suppose, there is a fast-food chain company and they sell a variety of different food items like burgers, pizza, sandwiches, milkshakes, etc. They have created a website to sell their food and now the customers can order any food item from their website and they can provide reviews as well, like whether they liked the food or hated it.

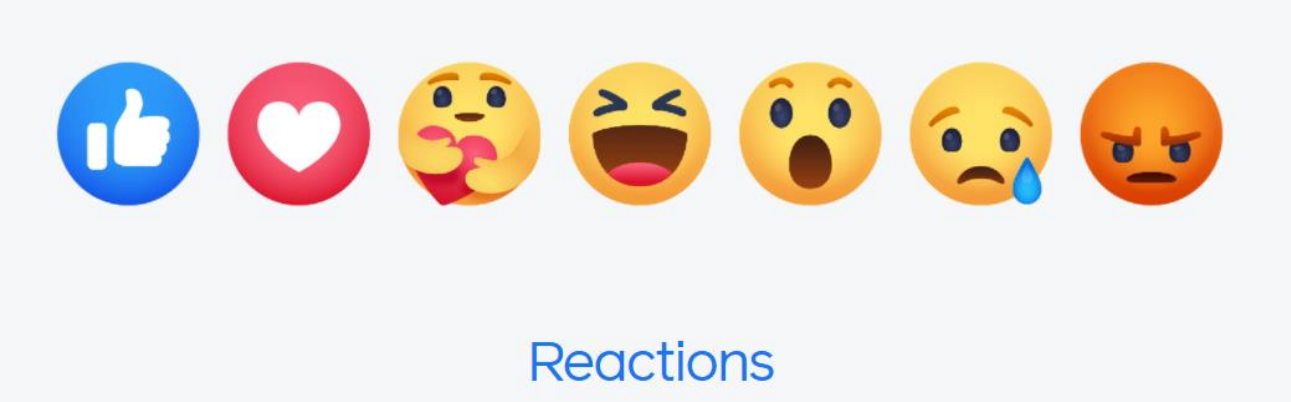
- User Review 1: I love this cheese sandwich, it's so delicious.
- User Review 2: This chicken burger has a very bad taste.
- User Review 3: I ordered this pizza today.

So, as we can see that out of these above 3 reviews,

The first review is definitely a positive one and it signifies that the customer was really happy with the sandwich.
The second review is negative, and hence the company needs to look into their burger department.
And, the third one doesn't signify whether that customer is happy or not, and hence we can consider this as a neutral statement.
By looking at the above reviews, the company can now conclude, that it needs to focus more on the production and promotion of their sandwiches as well as improve the quality of their burgers if they want to increase their overall sales.
But, now a problem arises, that there will be hundreds and thousands of user reviews for their products and after a point of time it will become nearly impossible to scan through each user review and come to a conclusion.

Neither can they just come up with a conclusion by taking just 100 reviews or so, because maybe the first 100-200 customers were having similar taste and liked the

sandwiches, but over time when the no. of reviews increases, there might be a situation where the positive reviews are overtaken by more no. of negative reviews. Therefore, this is where the Sentiment Analysis Model comes into play, which takes in a huge corpus of data having user reviews and finds a pattern and comes up with a conclusion based on real evidence rather than assumptions made on a small sample of



And, because of this upgrade, when any company promotes their products on Facebook, they receive more specific reviews which will help them to enhance the customer experience.
And because of that, they now have more granular control on how to handle their consumers, i.e. they can target the customers who are just “sad” in a different way as compared to customers who are “angry”, and come up with a business plan accordingly because nowadays, just doing the bare minimum is not enough.

Now, as we said we will be creating a Sentiment Analysis Model, but it’s easier said than done.
As we humans communicate with each other in a way that we call Natural Language which is easy for us to interpret but it’s much more complicated and messy if we really look into it.
Because, there are billions of people and they have their own style of communicating, i.e. a lot of tiny variations are added to the language and a lot of sentiments are attached to it which is easy for us to interpret but it becomes a challenge for the machines.
This is why we need a process that makes the computers understand the Natural Language as we humans do, and this is what we call Natural Language Processing(NLP). And, as we know Sentiment Analysis is a sub-field of NLP and

with the help of machine learning techniques, it tries to identify and extract the insights.

Social media has opened a whole new world for people around the globe. People are just a click away from getting huge chunk of information. With information comes people's opinion and with this comes the positive and negative outlook of people regarding a topic. Sometimes this also results into bullying and passing on hate comments about someone or something.

The objective of this project is to detect hate speech in tweets. For the sake of simplicity, we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets.

PRINCIPLES OF SYSTEM ANALYSIS

1. Understand the problem before you begin to create the analysis model.

2. Develop prototypes that enable a user to understand how human machine interaction will occur.

3. Record the origin of and the reason for every requirement.

4. Use multiple views of requirements like building data, function and behavioral models.

5. Work to eliminate ambiguity.

Hardware and Software requirement

PLATFORM	**Window 7 or above**
SOFTWARE	**Jupyter notebook, Power Point Presentation**
PROCESSOR	**Intel Core i3 and above**

RAM	**Minimum 4GB**

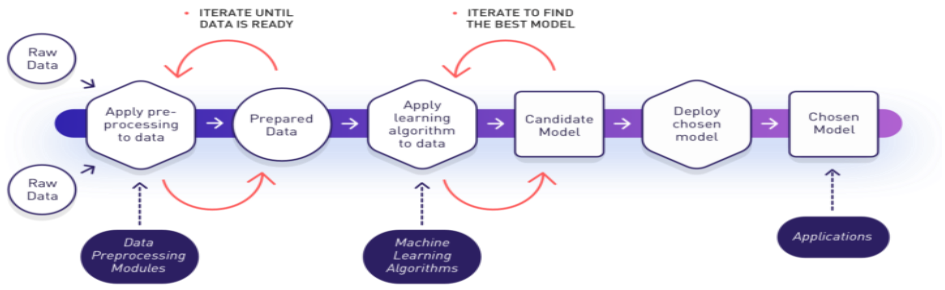
Project Pipeline

The various steps involved in the Machine Learning Pipeline are :

- Import Necessary Dependencies
- Read and Load the Dataset
- Exploratory Data Analysis
- Data Visualization of Target Variables
- Data Pre-processing
- Transforming Dataset
- Function for Model Evaluation
- Model Building
- Conclusion

Among all these, training the machine learning model is the most computationally intensive task.

Now if we talk about training the model, which generally requires a lot of computational power, the process could be frustrating if done without the right hardware. This intensive part of the neural network is made up of various matrix multiplications



Data Set Description

Formally, given a training sample of tweets and labels, where label ‘1’ denotes the tweet is racist/sexist and label ‘0’ denotes the tweet is not racist/sexist,our objective is to predict the labels on the given test dataset.

- id : The id associated with the tweets in the given dataset.
- tweets : The tweets collected from various sources and having either positive or negative sentiments associated with it.
- label : A tweet with label ‘0’ is of positive sentiment while a tweet with label ‘1’ is of negative sentiment.

Now, let us look at the code and description of the project that I made on Twitter Sentiment Analysis

Attaching the pdf of the jupyter notebook

Importing the necessary packages

In [1]:

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

Reading the train.csv Pandas file

In the first line we read the train.csv file using Pandas. In the second line as a safe backup we keep a copy of our original train.csv file. We make a copy of train data so that even if we have to make any changes in this dataset we would not lose the original dataset.

In [2]:

```
train = pd.read_csv('train_E6oV3lV.csv')

train_original=train.copy()
train.head()
```

Out[2]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

In [3]:

```
train.tail()
```

Out[3]:

	id	label	tweet
31957	31958	0	ate @user isz that youuu?ð ð ð ð ð...
31958	31959	0	to see nina turner on the airwaves trying to...
31959	31960	0	listening to sad songs on a monday morning otw...
31960	31961	1	@user #sikh #temple vandalised in in #calgary,...
31961	31962	0	thank you @user for you follow

In [4]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      31962 non-null  int64 
 1   label   31962 non-null  int64 
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

In [5]:

```
train.shape
```

Out[5]:

```
(31962, 3)
```

As you can see we have 3 attributes present in our dataset and a total of 31962 labeled tweets , ‘1’ standing for tweets with negative sentiment and ‘0’ for tweets with positive sentiments.

Reading the test.csv Pandas file

In the first line we read the test.csv file using Pandas. In the second line as a safe backup we keep a copy of our original test.csv file. We make a copy of test data so that even if we have to make any changes in this dataset we would not lose the original dataset.

In [6]:

```
test = pd.read_csv('test_tweets_anuFYb8.csv')

test_original=test.copy()
```

In [7]:

```
test.head()
```

Out[7]:

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

In [8]:

```
test.tail()
```

Out[8]:

	id	tweet
17192	49155	thought factory: left-right polarisation! #tru...
17193	49156	feeling like a mermaid ð #hairflip #neverre...
17194	49157	#hillary #campaigned today in #ohio((omg)) &am...
17195	49158	happy, at work conference: right mindset leads...
17196	49159	my song "so glad" free download! #shoegaze ...

In [9]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17197 entries, 0 to 17196
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      17197 non-null  int64 
 1   tweet   17197 non-null  object
dtypes: int64(1), object(1)
memory usage: 268.8+ KB
```

In [10]:

```
test.shape
```

Out[10]:

```
(17197, 2)
```

As we can see we have 2 attributes present here that is ‘id’ and ‘tweets’. This is the dataset on which we are going to test our Machine Learning models so it is unlabeled.

Data Pre-Processing

Let's begin with the pre-processing of our dataset.

STEP — 1 :

Combine the train.csv and test.csv files.

Pandas dataframe.append() function is used to append rows of other dataframe to the end of the given dataframe, returning a new dataframe object.

In [11]:

```
combine = train.append(test,ignore_index=True,sort=True)
```

Overview of the combined train and test dataset.

In [12]:

```
combine.head()
```

Out[12]:

	id	label	tweet
0	1	0.0	@user when a father is dysfunctional and is s...
1	2	0.0	@user @user thanks for #lyft credit i can't us...
2	3	0.0	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation

In [13]:

```
combine.tail()
```

Out[13]:

	id	label	tweet
49154	49155	NaN	thought factory: left-right polarisation! #tru...
49155	49156	NaN	feeling like a mermaid ð #hairflip #neverre...
49156	49157	NaN	#hillary #campaigned today in #ohio((omg)) &am...
49157	49158	NaN	happy, at work conference: right mindset leads...
49158	49159	NaN	my song "so glad" free download! #shoegaze ...

Columns not in the original dataframes are added as new columns and the new cells are populated with NaN value.

STEP — 2

Removing Twitter Handles(@User)

In our analysis we can clearly see that the Twitter handles do not contribute anything significant to solve our problem. So it's better if we remove them in our dataset.

Given below is a user-defined function to remove unwanted text patterns from the tweets. It takes two arguments, one is the original string of text and the other is the pattern of text that we want to remove from the string. The function returns the same input string but without the given pattern. We will use this function to remove the pattern '@user' from all the tweets in our data.

In [14]:

```
def remove_pattern(text,pattern):

    # re.findall() finds the pattern i.e @user and puts it in a list for further task
    r = re.findall(pattern,text)

    # re.sub() removes @user from the sentences in the dataset
    for i in r:
        text = re.sub(i,"",text)

    return text
```

Here NumPy Vectorization 'np.vectorize()' is used because it is much more faster than the conventional for loops when working on datasets of medium to large sizes.

In [15]:

```
combine['Tidy_Tweets'] = np.vectorize(remove_pattern)(combine['tweet'], "@[\w]*")

combine.head()
```

Out[15]:

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation	factsguide: society now #motivation

STEP — 3

Removing Punctuation, Numbers, and Special Characters

Punctuation, numbers and special characters do not help much. It is better to remove them from the text just as we removed the twitter handles. Here we will replace everything except characters and hashtags with spaces.

In [16]:

```
combine['Tidy_Tweets'] = combine['Tidy_Tweets'].str.replace("[^a-zA-Z#]", " ")

combine.head(10)
```

Out[16]:

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can t use cause th...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation	factsguide society now #motivation
5	6	0.0	[2/2] huge fan fare and big talking before the...	huge fan fare and big talking before the...
6	7	0.0	@user camping tomorrow @user @user @user @use...	camping tomorrow danny
7	8	0.0	the next school year is the year for exams.ð ...	the next school year is the year for exams ...
8	9	0.0	we won!!! love the land!!! #allin #cavs #champ...	we won love the land #allin #cavs #champ...
9	10	0.0	@user @user welcome here ! i'm it's so #gr...	welcome here i m it s so #gr

STEP — 4

Removing Short Words

We have to be a little careful here in selecting the length of the words which we want to remove. So, I have decided to remove all the words having length 3 or less. These words are also known as Stop Words.

For example, terms like “hmm”, “and”, “oh” are of very little use. It is better to get rid of them.

In [17]:

```
combine['Tidy_Tweets'] = combine['Tidy_Tweets'].apply(lambda x: ' '.join([w for w in x.spli

combine.head(10)
```

Out[17]:

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguide society #motivation
5	6	0.0	[2/2] huge fan fare and big talking before the...	huge fare talking before they leave chaos disp...
6	7	0.0	@user camping tomorrow @user @user @user @use...	camping tomorrow danny
7	8	0.0	the next school year is the year for exams.ð ...	next school year year exams think about that #...
8	9	0.0	we won!!! love the land!!! #allin #cavs #champ...	love land #allin #cavs #champions #cleveland #...
9	10	0.0	@user @user welcome here ! i'm it's so #gr...	welcome here

STEP — 5

Tokenization

Now we will tokenize all the cleaned tweets in our dataset. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.

Here we tokenize our sentences because we will apply Stemming from the “NLTK” package in the next step.

In [18]:

```
tokenized_tweet = combine['Tidy_Tweets'].apply(lambda x: x.split())

tokenized_tweet.head()
```

Out[18]:

```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                               [bihday, your, majesty]
3                     [#model, love, take, with, time]
4                   [factsguide, society, #motivation]
Name: Tidy_Tweets, dtype: object
```

STEP — 6

Stemming

Stemming is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.

For example — “play”, “player”, “played”, “plays” and “playing” are the different variations of the word — “play”

In [19]:

```
from nltk import PorterStemmer

ps = PorterStemmer()

tokenized_tweet = tokenized_tweet.apply(lambda x: [ps.stem(i) for i in x])

tokenized_tweet.head()
```

Out[19]:

```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, caus, they, offer, whee...
2                              [bihday, your, majesti]
3                    [#model, love, take, with, time]
4                        [factsguid, societi, #motiv]
Name: Tidy_Tweets, dtype: object
```

Now let’s stitch these tokens back together

In [20]:

```
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = ' '.join(tokenized_tweet[i])

combine['Tidy_Tweets'] = tokenized_tweet
combine.head()
```

Out[20]:

	id	label	tweet	Tidy_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunct selfish drag kid into dys...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thank #lyft credit caus they offer wheelchair ...
2	3	0.0	bihday your majesty	bihday your majesti
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguid societi #motiv

So finally these are the basic steps to follow when we have to Pre-Process a dataset containing textual data.

Data Visualisation

So Data Visualisation is one of the most important steps in Machine Learning projects because it gives us an

approximate idea about the dataset and what it is all about before proceeding to apply different machine learning models.

So, let's dive in.

WordCloud

One of the popular visualisation techniques is WordCloud.

WordCloud is a visualisation wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.

So, in Python we have a package for generating WordCloud.

Let's dive into the code to see how can we generate a WordCloud.

Importing packages necessary for generating a WordCloud

In [21]:

```
from wordcloud import WordCloud,ImageColorGenerator
from PIL import Image
import urllib
import requests
```

Generating WordCloud for tweets with label '0'.

Store all the words from the dataset which are non-racist/sexist.

In [22]:

```
all_words_positive = ' '.join(text for text in combine['Tidy_Tweets'][combine['label']==0])
```

In [23]:

```
# combining the image with the dataset
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitter-

# We use the ImageColorGenerator library from Wordcloud
# Here we take the color of the image and impose it over our wordcloud
image_colors = ImageColorGenerator(Mask)

# Now we use the WordCloud function from the wordcloud library
wc = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(all_wo
```

In [24]:

```
# Size of the image generated
plt.figure(figsize=(10,20))

# Here we recolor the words from the dataset to the image's color
# recolor just recolors the default colors to the image's blue color
# interpolation is used to smooth the image generated
plt.imshow(wc.recolor(color_func=image_colors),interpolation="hamming")

plt.axis('off')
plt.show()
```

We can see most of the words are positive or neutral. With happy, smile, and love being the most frequent ones. Hence, most of the frequent words are compatible with tweets in positive sentiment.

Generating WordCloud for tweets with label '1'.

Store all the words from the dataset which are non-racist/sexist.

In [25]:

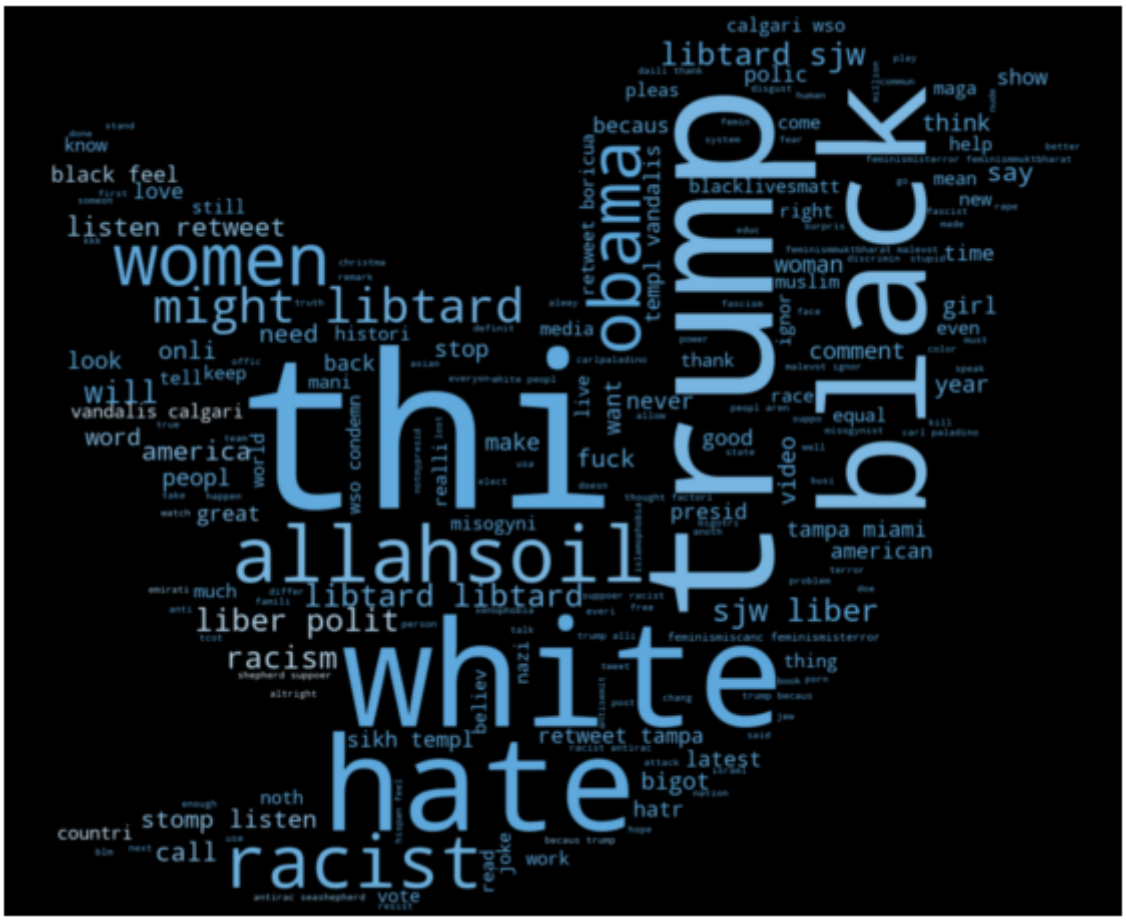
```
all_words_negative = ' '.join(text for text in combine['Tidy_Tweets'][combine['label']==1])
```

In [26]:

```
# combining the image with the dataset
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitter-

# We use the ImageColorGenerator library from Wordcloud
# Here we take the color of the image and impose it over our wordcloud
image_colors = ImageColorGenerator(Mask)

# Now we use the WordCloud function from the wordcloud library
wc = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(all_wo
```

In [27]:

```
# Size of the image generated
plt.figure(figsize=(10,20))

# Here we recolor the words from the dataset to the image's color
# recolor just recolors the default colors to the image's blue color
# interpolation is used to smooth the image generated
plt.imshow(wc.recolor(color_func=image_colors),interpolation="gaussian")

plt.axis('off')
plt.show()
```



We can clearly see, most of the words have negative connotations. So, it seems we have a pretty good text

data to work on.

Understanding the impact of Hashtags on tweets sentiment

Hash-tagging on Twitter can have a major impact when it comes to your follower count by using general and non-specific hashtags. If you hashtag general words, like #creative, or events, like #TIFF, that are going on, it is more likely that your tweet will reach beyond your follower list.

So we will look how we can extract the hashtags and see which hashtags fall into which category.

Function to extract hashtags from tweets

In [28]:

```
def Hashtags_Extract(x):
    hashtags=[]

    # Loop over the words in the tweet
    for i in x:
        ht = re.findall(r'#(\w+)',i)
        hashtags.append(ht)

    return hashtags
```

A nested list of all the hashtags from the positive reviews from the dataset.

In [29]:

```
ht_positive = Hashtags_Extract(combine['Tidy_Tweets'][combine['label']==0])

ht_positive
```

Out[29]:

```
[['run'],
 ['lyft', 'disapoint', 'getthank'],
 [],
 ['model'],
 ['motiv'],
 ['allshowandnogo'],
 [],
 ['school', 'exam', 'hate', 'imagin', 'actorslif', 'revolutionschool', 'gi
rl'],
 ['allin', 'cav', 'champion', 'cleveland', 'clevelandcavali'],
 [],
 ['ireland', 'blog', 'silver', 'gold', 'forex'],
 ['orlando',
  'standwithorlando',
  'pulseshoot',
  'orlandoshoot',
  'biggerproblem',
  'selfish',
```

Here we unnest the list

In [30]:

```
ht_positive_unnest = sum(ht_positive,[])
```

A nested list of all the hashtags from the negative reviews from the dataset

In [31]:

```
ht_negative = Hashtags_Extract(combine['Tidy_Tweets'][combine['label']==1])

ht_negative
```

Out[31]:

```
[['cnn', 'michigan', 'tcot'],
 ['australia',
  'opkillingbay',
  'seashepherd',
  'helpcovedolphin',
  'thecov',
  'helpcovedolphin'],
 [],
 [],
 ['neverump', 'xenophobia'],
 ['love', 'peac'],
 [],
 ['race', 'ident', 'med'],
 ['altright', 'whitesupremaci'],
 ['linguist', 'race', 'power', 'raciolinguist'],
 ['brexit'],
 ['peopl', 'trump', 'republican'],
 ['michelleobama'],
```

Here we unnest the list

In [32]:

```
ht_negative_unnest = sum(ht_negative,[])
```

Plotting Bar-plots

For Positive Tweets in the dataset

Counting the frequency of the words having Positive Sentiment

In [33]:

```
word_freq_positive = nltk.FreqDist(ht_positive_unnest)

word_freq_positive
```

Out[33]:

```
FreqDist({'love': 1654, 'posit': 917, 'smile': 676, 'healthi': 573, 'thank':
534, 'fun': 463, 'life': 425, 'affirm': 423, 'summer': 390, 'model': 375,
...})
```

Creating a dataframe for the most frequently used words in hashtags

In [34]:

```
df_positive = pd.DataFrame({'Hashtags':list(word_freq_positive.keys()),'Count':list(word_fr

df_positive.head(10)
```

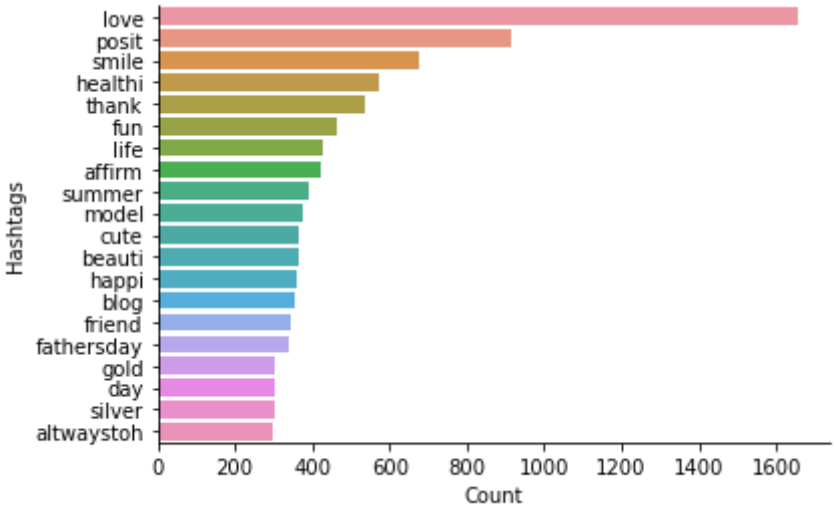
Out[34]:

	Hashtags	Count
0	run	72
1	lyft	2
2	disapoint	1
3	getthank	2
4	model	375
5	motiv	202
6	allshowandnogo	1
7	school	30
8	exam	9
9	hate	27

Plotting the barplot for the 20 most frequent words used for hashtags

In [35]:

```
df_positive_plot = df_positive.nlargest(20,columns='Count')

sns.barplot(data=df_positive_plot,y='Hashtags',x='Count')
sns.despine()
```



For Negative Tweets in the dataset

Counting the frequency of the words having Negative Sentiment

In [36]:

```
word_freq_negative = nltk.FreqDist(ht_negative_unnest)

word_freq_negative
```

Out[36]:

```
FreqDist({'trump': 136, 'polit': 95, 'allahsoil': 92, 'liber': 81, 'libtar
d': 77, 'sjw': 75, 'retweet': 63, 'black': 46, 'miami': 46, 'hate': 37,
...})
```

Creating a dataframe for the most frequently used words in hashtags

In [37]:

```
df_negative = pd.DataFrame({'Hashtags':list(word_freq_negative.keys()),'Count':list(word_fr

df_negative.head(10)
```

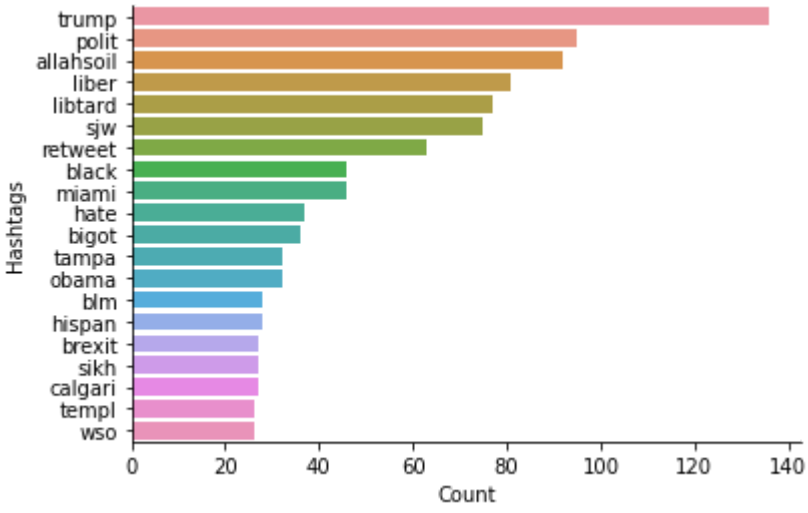
Out[37]:

	Hashtags	Count
0	cnn	10
1	michigan	2
2	tcot	14
3	australia	6
4	opkillingbay	5
5	seashepherd	22
6	helpcovedolphin	3
7	thecov	4
8	neverump	8
9	xenophobia	12

Plotting the barplot for the 20 most frequent words used for hashtags

In [38]:

```
df_negative_plot = df_negative.nlargest(20,columns='Count')

sns.barplot(data=df_negative_plot,y='Hashtags',x='Count')
sns.despine()
```



Extracting Features from cleaned Tweets

Bag-of-Words Features

Bag of Words is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set.

Consider a corpus (a collection of texts) called C of D documents {d1,d2…..dD} and N unique tokens extracted out of the corpus C. The N tokens (words) will form a list, and the size of the bag-of-words matrix M will be given by D X N. Each row in the matrix M contains the frequency of tokens in document D(i).

For example, if you have 2 documents-

D1: He is a lazy boy. She is also lazy. D2: Smith is a lazy person. First, it creates a vocabulary using unique words from all the documents.

[‘He’ , ’She’ , ’lazy’ , ‘boy’ , ‘Smith’ , ’person’]

As we can see in the above list we don’t consider “is” , “a” , “also” in this set because they don’t convey the necessary information required for the model.

Here, D=2, N=6 The matrix M of size 2 X 6 will be represented as:

	He	She	lazy	boy	Smith	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1

The above table depicts the training features containing term frequencies of each word in each document. This is called bag-of-words approach since the number of occurrence and not sequence or order of words matters in this approach.

So,let’s apply this word embedding technique to our available dataset.

We have a package called CountVectorizer to perform this task.

In [39]:

```
from sklearn.feature_extraction.text import CountVectorizer

bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='engl

# bag-of-words feature matrix
bow = bow_vectorizer.fit_transform(combine['Tidy_Tweets'])

df_bow = pd.DataFrame(bow.todense())

df_bow
```

Out[39]:

	0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996	997	998
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49154	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
49155	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
49156	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
49157	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
49158	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0

49159 rows × 1000 columns

TF-IDF Features

TF-IDF stands for Term Frequency-Inverse Document Frequency, and the TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Typically, the TF-IDF weight is composed by two terms:

Term Frequency (TF) :

The first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document.

$$TF(\boldsymbol{term}) = \frac{Number\ of\ times\ \boldsymbol{term}\ appears\ in\ a\ document}{Total\ number\ of\ items\ in\ the\ document}$$

Example :-

Consider a document containing 100 words wherein the word cat appears 3 times.

The Term Frequency (TF) for cat is then (3 / 100) = 0.03

Inverse Document Frequency (IDF) :

The second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

$$IDF(\boldsymbol{term}) = \log\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ \boldsymbol{term}\ in\ it}\right)$$

Example :-

Assume we have 10 million documents and the word cat appears in one thousand of these.

Then, the Inverse Document Frequency (IDF) is calculated as

log(10,000,000 / 1,000) = 4.

TF-IDF Example :

Formula for finding the TF-IDF weight :-

$$TFIDF(\boldsymbol{term}) = TF(\boldsymbol{term}) * \mathrm{IDF}(\mathbf{term})$$

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

From the above examples the Term Frequency is 0.03 and Inverse Document Frequency is 4.

Thus, the TF-IDF weight is the product of these quantities : 0.03 * 4 = 0.12.

Let us apply this technique to our dataset using Python.

We have a package available for this in Scikit-Learn known as TfidfVectorizer.

In [40]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf=TfidfVectorizer(max_df=0.90, min_df=2,max_features=1000,stop_words='english')

tfidf_matrix=tfidf.fit_transform(combine['Tidy_Tweets'])

df_tfidf = pd.DataFrame(tfidf_matrix.todense())

df_tfidf
```

Out[40]:

	0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49154	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
49155	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
49156	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
49157	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
49158	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

49159 rows × 1000 columns

These are the Word Embedding techniques which we have used on our dataset for feature extraction.m

Splitting our dataset into Training and Validation Set

From the above two techniques that is Bag-of-Words and TF-IDF we have extracted features from the tweets present in our dataset.

Now, we have one dataset with features from the Bag-of-Words model and another dataset with features from TF-IDF model.

First task is to split the dataset into training and validation set so that we can train and test our model before applying it to predict for unseen and unlabeled test data.

Using the features from Bag-of-Words for training set

In [41]:

```
train_bow = bow[:31962]

train_bow.todense()
```

Out[41]:

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Using features from TF-IDF for training set

In [42]:

```
train_tfidf_matrix = tfidf_matrix[:31962]

train_tfidf_matrix.todense()
```

Out[42]:

```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

Splitting the data into training and validation set

In [43]:

```
from sklearn.model_selection import train_test_split
```

Bag-of-Words Features

In [44]:

```
x_train_bow, x_valid_bow, y_train_bow, y_valid_bow = train_test_split(train_bow,train['labe
```

TF-IDF features

In [45]:

```
x_train_tfidf, x_valid_tfidf, y_train_tfidf, y_valid_tfidf = train_test_split(train_tfidf_m
```

Applying Machine Learning Models

The underlying problem we are going to solve comes under the Supervised Machine Learning category. So, let us have a brief discussion about this topic before moving on to apply different Machine Learning models on our dataset.

Supervised Machine Learning :-

The majority of practical machine learning uses supervised learning.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

Y = f(X) The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into regression and classification problems.

Classification:

A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease” or in our case “Positive” or “Negative”

Regression:

A regression problem is when the output variable is a real value, such as “dollars” or “weight”. Our problem comes under the classification category because we have to classify our results into either Positive or Negative class.

There is another category of Machine Learning algorithm called Unsupervised Machine Learning where you have an input data but no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. But that is of no concern for us for this problem statement.

Moving on :-

So from the above splitting of dataset we see that we will use features from the Bag-of-Words and TF-IDF for our Machine Learning Models.

We generally use different models to see which best fits our dataset and then we use that model for predicting results on the test data.

Here we will use 3 different models

Logistic Regression

XGBoost

Decision Trees

and then we will compare their performance and choose the best possible model with the best possible feature extraction technique for predicting results on our test data.

Importing f1_score from sklearn

We will use F1 Score throughout to asses our model's performance instead of accuracy. You will get to know why at the end of this article.

In [46]:

```
from sklearn.metrics import f1_score
```

Logistic Regression

In [47]:

```
from sklearn.linear_model import LogisticRegression
Log_Reg = LogisticRegression(random_state=0,solver='lbfgs')
```

Bag-of-Words Features

Fitting the Logistic Regression Model.

In [48]:

```
Log_Reg.fit(x_train_bow,y_train_bow)
```

Out[48]:

```
LogisticRegression(random_state=0)
```

Predicting the probabilities.

In [49]:

```
prediction_bow = Log_Reg.predict_proba(x_valid_bow)

prediction_bow
```

Out[49]:

```
array([[9.86501156e-01, 1.34988440e-02],
       [9.99599096e-01, 4.00904144e-04],
       [9.13577383e-01, 8.64226167e-02],
       ...,
       [8.95457155e-01, 1.04542845e-01],
       [9.59736065e-01, 4.02639345e-02],
       [9.67541420e-01, 3.24585797e-02]])
```

The output basically provides us with the probabilities of the tweet falling into either of the classes that is Negative or Positive.

Calculating the F1 score

In [50]:

```
# if prediction is greater than or equal to 0.3 than 1 else 0
# Where 0 is for positive sentiment tweets and 1 for negative sentiment tweets
prediction_int = prediction_bow[:,1]>=0.3

# converting the results to integer type
prediction_int = prediction_int.astype(np.int)
prediction_int

# calculating f1 score
log_bow = f1_score(y_valid_bow, prediction_int)

log_bow
```

Out[50]:

```
0.5721352019785655
```

TF-IDF Features

Fitting the Logistic Regression Model.

In [51]:

```
Log_Reg.fit(x_train_tfidf,y_train_tfidf)
```

Out[51]:

```
LogisticRegression(random_state=0)
```

Predicting the probabilities.

In [52]:

```
prediction_tfidf = Log_Reg.predict_proba(x_valid_tfidf)

prediction_tfidf
```

Out[52]:

```
array([[0.98487907, 0.01512093],
       [0.97949889, 0.02050111],
       [0.9419737 , 0.0580263 ],
       ...,
       [0.98630906, 0.01369094],
       [0.96746188, 0.03253812],
       [0.99055287, 0.00944713]])
```

Calculating the F1 Score

In [53]:

```
# if prediction is greater than or equal to 0.3 than 1 else 0
# Where 0 is for positive sentiment tweets and 1 for negative sentiment tweets
prediction_int = prediction_tfidf[:,1]>=0.3

prediction_int = prediction_int.astype(np.int)
prediction_int

# calculating f1 score
log_tfidf = f1_score(y_valid_tfidf, prediction_int)

log_tfidf
```

Out[53]:

```
0.5862068965517241
```

XGBoost

The next model we use is XGBoost.

In [54]:

```
from xgboost import XGBClassifier
```

Bag-of-Words Features

In [55]:

```
model_bow = XGBClassifier(random_state=22,learning_rate=0.9)
```

Fitting the XGBoost Model

In [56]:

```
model_bow.fit(x_train_bow, y_train_bow)
```

Out[56]:

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.9, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=
1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=
22,
              reg_alpha=0, reg_lambda=1, ...)
```

Predicting the probabilities.

In [57]:

```
xgb = model_bow.predict_proba(x_valid_bow)

xgb
```

Out[57]:

```
array([[9.9109417e-01, 8.9058345e-03],
       [9.9910688e-01, 8.9310389e-04],
       [9.2758572e-01, 7.2414286e-02],
       ...,
       [8.9067906e-01, 1.0932093e-01],
       [9.2758572e-01, 7.2414286e-02],
       [9.2758572e-01, 7.2414286e-02]], dtype=float32)
```

Calculating the F1 Score

In [58]:

```
# if prediction is greater than or equal to 0.3 than 1 else 0
# Where 0 is for positive sentiment tweets and 1 for negative sentiment tweets
xgb=xgb[:,1]>=0.3

# converting the results to integer type
xgb_int=xgb.astype(np.int)

# calculating f1 score
xgb_bow=f1_score(y_valid_bow,xgb_int)

xgb_bow
```

Out[58]:

```
0.5811023622047244
```

TF-IDF Features

In [59]:

```
model_tfidf = XGBClassifier(random_state=29,learning_rate=0.7)
```

Fitting the XGBoost model

In [60]:

```
model_tfidf.fit(x_train_tfidf, y_train_tfidf)
```

Out[60]:

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.7, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=
1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=
29,
              reg_alpha=0, reg_lambda=1, ...)
```

Predicting the probabilities.

In [61]:

```
xgb_tfidf=model_tfidf.predict_proba(x_valid_tfidf)

xgb_tfidf
```

Out[61]:

```
array([[0.9958908 , 0.00410918],
       [0.9872918 , 0.01270822],
       [0.9527768 , 0.04722321],
       ...,
       [0.99192214, 0.00807783],
       [0.9852176 , 0.01478244],
       [0.9974108 , 0.0025892 ]], dtype=float32)
```

Calculating the F1 Score

In [62]:

```
# if prediction is greater than or equal to 0.3 than 1 else 0
# Where 0 is for positive sentiment tweets and 1 for negative sentiment tweets
xgb_tfidf=xgb_tfidf[:, 1]>=0.3

# converting the results to integer type
xgb_int_tfidf=xgb_tfidf.astype(np.int)

# calculating f1 score
score=f1_score(y_valid_tfidf,xgb_int_tfidf)

score
```

Out[62]:

```
0.5792
```

Decision Trees

In [63]:

```
from sklearn.tree import DecisionTreeClassifier
dct = DecisionTreeClassifier(criterion='entropy', random_state=1)
```

Bag-of-Words Features

Fitting the Decision Tree model.

In [64]:

```
dct.fit(x_train_bow,y_train_bow)
```

Out[64]:

```
DecisionTreeClassifier(criterion='entropy', random_state=1)
```

Predicting the probabilities.

In [65]:

```
dct_bow = dct.predict_proba(x_valid_bow)

dct_bow
```

Out[65]:

```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

Calculating the F1 Score

In [66]:

```
# if prediction is greater than or equal to 0.3 than 1 else 0
# Where 0 is for positive sentiment tweets and 1 for negative sentiment tweets
dct_bow=dct_bow[:,1]>=0.3

# converting the results to integer type
dct_int_bow=dct_bow.astype(np.int)

# calculating f1 score
dct_score_bow=f1_score(y_valid_bow,dct_int_bow)

dct_score_bow
```

Out[66]:

```
0.5141776937618148
```

Fitting the Decision Tree model

In [67]:

```
dct.fit(x_train_tfidf,y_train_tfidf)
```

Out[67]:

```
DecisionTreeClassifier(criterion='entropy', random_state=1)
```

Predicting the probabilities.

In [68]:

```
dct_tfidf = dct.predict_proba(x_valid_tfidf)

dct_tfidf
```

Out[68]:

```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [1., 0.],
       [1., 0.],
       [1., 0.]])
```

Calculating the F1 Score

In [69]:

```
# if prediction is greater than or equal to 0.3 than 1 else 0
# Where 0 is for positive sentiment tweets and 1 for negative sentiment tweets
dct_tfidf=dct_tfidf[:,1]>=0.3

# converting the results to integer type
dct_int_tfidf=dct_tfidf.astype(np.int)

# calculating f1 score
dct_score_tfidf=f1_score(y_valid_tfidf,dct_int_tfidf)

dct_score_tfidf
```

Out[69]:

```
0.5498821681068342
```

Model Comparison

Now, let us compare the different models we have applied on our dataset with different word embedding techniques.

Bag of Words

In [70]:

```
Algo_1 = ['LogisticRegression(Bag-of-Words)','XGBoost(Bag-of-Words)','DecisionTree(Bag-of-W

score_1 = [log_bow,xgb_bow,dct_score_bow]

compare_1 = pd.DataFrame({'Model':Algo_1,'F1_Score':score_1},index=[i for i in range(1,4)])

compare_1.T
```

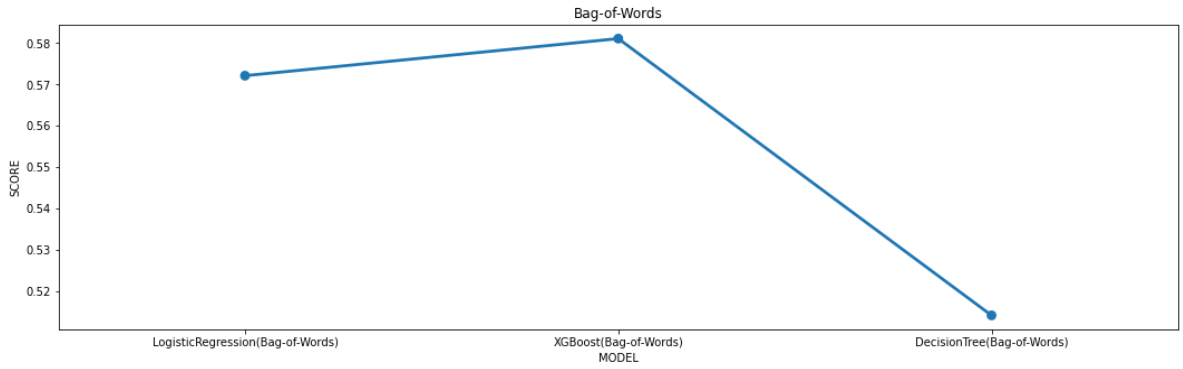
Out[70]:

	1	2	3
Model	LogisticRegression(Bag-of-Words)	XGBoost(Bag-of-Words)	DecisionTree(Bag-of-Words)
F1_Score	0.572135	0.581102	0.514178

Comparison Graph

In [71]:

```
plt.figure(figsize=(18,5))

sns.pointplot(x='Model',y='F1_Score',data=compare_1)

plt.title('Bag-of-Words')
plt.xlabel('MODEL')
plt.ylabel('SCORE')

plt.show()
```



TF-IDF

In [72]:

```
Algo_2 = ['LogisticRegression(TF-IDF)','XGBoost(TF-IDF)','DecisionTree(TF-IDF)']

score_2 = [log_tfidf,score,dct_score_tfidf]

compare_2 = pd.DataFrame({'Model':Algo_2,'F1_Score':score_2},index=[i for i in range(1,4)])

compare_2.T
```

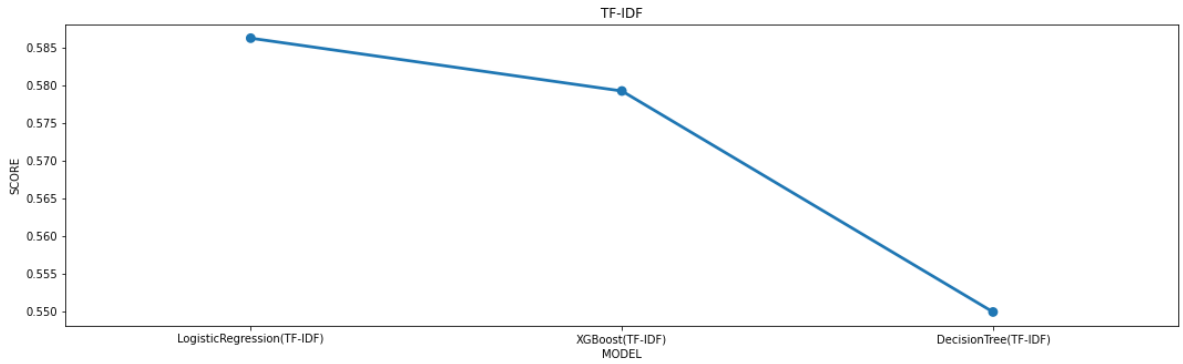
Out[72]:

	1	2	3
Model	LogisticRegression(TF-IDF)	XGBoost(TF-IDF)	DecisionTree(TF-IDF)
F1_Score	0.586207	0.5792	0.549882

Comparison Graph

In [73]:

```
plt.figure(figsize=(18,5))

sns.pointplot(x='Model',y='F1_Score',data=compare_2)

plt.title('TF-IDF')
plt.xlabel('MODEL')
plt.ylabel('SCORE')

plt.show()
```



As we can see the best possible model from both Bag-of-Words and TF-IDF is Logistic Regression.

Now, let us compare the score of the Logistic Regression model with both the feature extraction techniques that is Bag-of-Words and TF-IDF.

In [74]:

```
Algo_best = ['LogisticRegression(Bag-of-Words)','LogisticRegression(TF-IDF)']

score_best = [log_bow,log_tfidf]

compare_best = pd.DataFrame({'Model':Algo_best,'F1_Score':score_best},index=[i for i in ran

compare_best.T
```

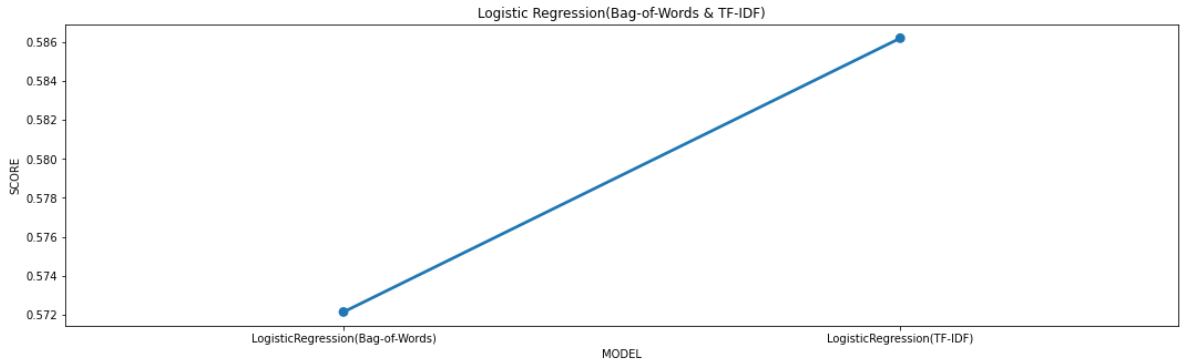
Out[74]:

	1	2
Model	LogisticRegression(Bag-of-Words)	LogisticRegression(TF-IDF)
F1_Score	0.572135	0.586207

Comparison Graph

In [75]:

```
plt.figure(figsize=(18,5))

sns.pointplot(x='Model',y='F1_Score',data=compare_best)

plt.title('Logistic Regression(Bag-of-Words & TF-IDF)')
plt.xlabel('MODEL')
plt.ylabel('SCORE')

plt.show()
```



Predicting the results for our test data

From the above comparison graph we can clearly see that the best possible F1 Score is obtained by the Logistic Regression Model using TF-IDF features.

In [76]:

```
test_tfidf = tfidf_matrix[31962:]
test_pred = Log_Reg.predict_proba(test_tfidf)

test_pred_int = test_pred[:,1] >= 0.3
test_pred_int = test_pred_int.astype(np.int)

test['label'] = test_pred_int

submission = test[['id','label']]
submission.to_csv('result.csv', index=False)
```

Results after Prediction

In [77]:

```
res = pd.read_csv('result.csv')
res
```

Out[77]:

	id	label
0	31963	0
1	31964	0
2	31965	0
3	31966	0
4	31967	0
...	...	...
17192	49155	1
17193	49156	0
17194	49157	0
17195	49158	0
17196	49159	0

17197 rows × 2 columns

From the above output we can see that our Logistic Regression model with TF-IDF features predicts whether a tweets falls into the category of Positive — label : 0 or Negative — label : 1 sentiment.

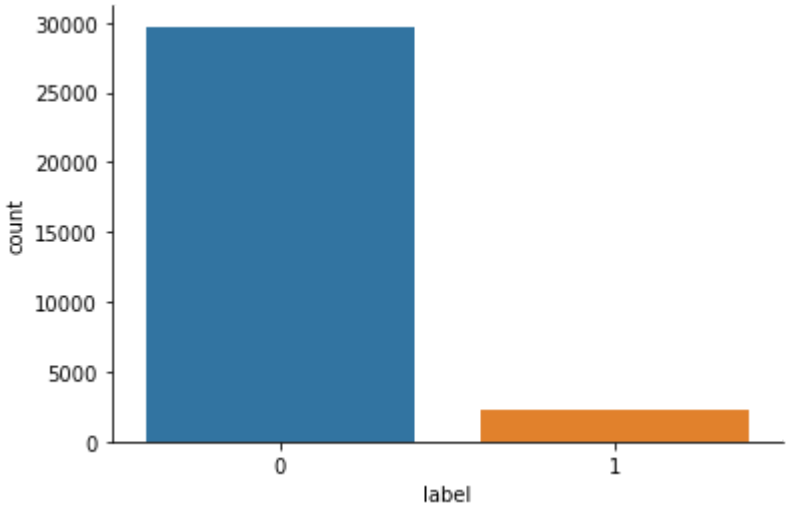
The questions that arises are “What is F1 Score ?” and “Why F1 Score instead of accuracy ?”.

Why F1 Score instead of Accuracy ?

Let us generate a countplot for our training dataset labels i.e. ‘0’ or ‘1’ .

In [78]:

```
sns.countplot(x=train_original['label'])
sns.despine()
```



From the above countplot generated above we see how imbalanced our dataset is.

We can see that the values with Positive — label : 0 sentiments are quite high in number as compared to the values with Negative — label : 1 sentiments.

So when we keep Accuracy as our evaluation metric there may be cases where we may encounter high number of false positives. So that is why we use F1 Score as our evaluation metric instead of Accuracy.

What is F1 Score ?

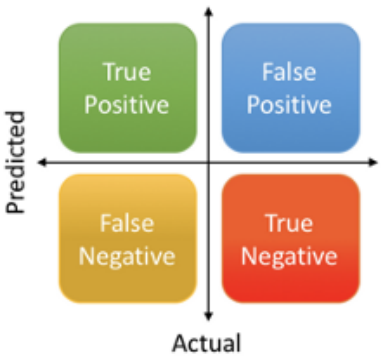
To know about F1 Score we first have to know about Precision and Recall.

Precision means the percentage of your results which are relevant. Recall refers to the percentage of total relevant results correctly classified by your algorithm.

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



We always face a trade-off situation between Precision and Recall i.e. High Precision gives low Recall and vice versa.

In most problems, you could either give a higher priority to maximising precision, or recall, depending upon the problem you are trying to solve. But in general, there is a simpler metric which takes into account both precision and recall, and therefore, you can aim to maximise this number to make your model better. This metric is known as F1-score, which is simply the harmonic mean of Precision and Recall.

$$\textbf{F1 Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

So this metric seems much more easier and convenient to work with, as you only have to maximise one score, rather than balancing two separate scores.

REASON FOR CHOOSING AI

Continuous Improvement

Machine Learning and Deep Learning algorithms are capable of learning from the data we provide. As new data is provided, the model's accuracy and efficiency to make decisions improve with subsequent training. Giants like Amazon, Walmart, etc collect a huge volume of new data every day. The accuracy of finding associated products or recommendation engine improves with this huge amount of training data available.

Automation for everything

A very powerful utility of AI is its ability to automate various decision-making tasks. This frees up a lot of time for developers to use their time to more productive use. For example, some common use we see in our daily life is social media sentiment analysis and chatbots. The moment a negative tweet is made related to a product or service of a Company, a chatbot instantly replies as first-level customer support. AIis changing the world with its automation for almost everything we can think of.

Trends and patterns identification

This advantage is a no brainer. All of us interested in Machine Learning technology are well aware of how the various Supervised, Unsupervised and Reinforced learning algorithms can be used for various classification and regression problems. We identify various trends and patterns with a huge amount of data using this technology. For example, Amazon analyzes the buying patterns and search trends of its customers and predicts products for them using Machine Learning algorithms.

Wide range of applications

AI is used in every industry these days, for example from Defence to Education. Companies generate profits, cut costs, automate, predict the future, analyze trends and patterns from the past data, and many more. Applications like GPS Tracking for traffic, Email spam filtering, text prediction, spell check and correction, etc are a few used widely these days.

CONCLUSION

Practical knowledge means the visualization of the knowledge, which we read in our books. For this, we perform experiments and get observations. Practical knowledge is very important in every field. One must be familiar with the problems related to that field so that he may solve them and become a successful person.

After achieving the proper goal in life, an engineer has to enter in professional life. According to this life, he has to serve an industry, may be public or private sector or self-own. For the efficient work in the field, he must be well aware of the practical knowledge as well as theoretical knowledge.

Due to all above reasons and to bridge the gap between theory and practical, our Engineering curriculum provides a practical training of 45 days. During this period a student work in the industry and get well all type of experience and knowledge about the working of companies and hardware and software tools.

I have undergone my 45 days summer training in 5th semester from AllSoft Solutions. This report is based on the knowledge, which I acquired during my 45 days of summer training**.**

BIBLIOGRAPHY

- ALLSOFT SOLUTION Course
- Kaggle.com
    - Google Research papers