A Project Report On

Sentiment Analysis using Twitter

Submitted in Partial Fulfillment of the Requirement for the Award of Degree

BACHELOR OF TECHNOLOGY

Computer Science & Engineering

Session (2021-22)

Submitted To-
Dr. Sunil Verma Sir

Under Supervision of-
Mr. Amit Singh

Submitted By-
Durgesh Yadav (1818710038)
Anant Narayan Patel (1818710021)
Abhishek Kumar Jaiswal (1818710005)

FEROZE GANDHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

RAEBARELI, U.P., INDIA

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY
LUCKNOW, U.P., INDIA

DECLARATION

We, **Durgesh Yadav, Anant Narayan Patel** and **Abhishek Kumar Jaiswal**, hereby declare that the project, entitled "**Sentimental Analysis Of Twitter Data**" submitted to **Feroze Gandhi Institute Of Engineering And Technology** in partial fulfillment of the requirements for the award of the Degree of B.Tech is record of ongoing project undergone by us during session 2021-22 under the supervision and guidance of **Mr. Amit Singh, Assistant Professor, Department of Computer Science And Engineering,** Feroze Gandhi Institute of Engineering and Technology, Dr. A.P.J. Abdul Technical University, Lucknow. This project report has not been submitted partially or fully to any other University or Institution for the award of any degree.

Durgesh Yadav
Roll No:1818710038
B.Tech(8th sem)

Anant Narayan Patel
Roll No: 1818710021
B.Tech(8th sem)

Abhishek Kumar Jaiswal
Roll No: 1818710005
B.Tech(8th sem)

Dated:30/05/2022
Place: Raebareli, Uttar Pradesh

CERTIFICATE

This is to certify that Project Report entitled **“Sentiment Analysis Of Twitter Data”** which is submitted by **Durgesh Yadav, Anant Narayan Patel** and **Abhishek Kumar Jaiswal** in partial fulfillment of the requirement for the award of degree **Bachelor of Technology** in **Department of Computer Science & Engineering** of **Dr. A. P. J. Abdul Kalam Technical University, Lucknow** is a record of candidates’ own work carried out by them under my supervision. The matter embodied in this project is original and has not been submitted for award of any other degree.

Project Guide

Amit Singh
(Assistant Professor)
Department of Computer Science & Engineering

Date:

ACKNOWLEDGMENT

It gives us a great sense of pleasure to present the report of the B.Tech Project undertaken during Bachelor of Technology Final Year. We take this opportunity express our profound gratitude and deepest regards to our guide **Mr. Amit Singh,** Assistant Professor, Department of Computer Science and Engineering, **Feroze Gandhi Institute of Engineering & Technology, Raebareli** for this constant support and guidance throughout the course of our work. The blessing, help and guidance given by his time to time shall carry us a long way in journey of life on which we about to embark. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take this opportunity to express a deep sense of gratitude to **Dr. Sunil Verma,** Associate Professor & Head of Department of Computer Science & Engineering FGIET Raebareli for their cordial support, valuable information and guidance which helped us in completing this task through various stages.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and coop--ration during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of this project.

ABSTRACT

Social media platforms are witnessing a significant growth in both size and purpose. One specific aspect of social media platforms is sentiment analysis, by which insights into the emotions and feelings of a person can be inferred from their posted text. Research related to sentiment analysis is acquiring substantial interest as it is a promising filed that can improve user experience and provide countless personalized services.

Twitter is one of the most popular social media platforms, it has users from different regions with a variety of cultures and languages. It can thus provide valuable information for a diverse and large amount of data to be used to improve decision making. In this paper, the sentiment orientation of the textual features and emoji-based components is studied targeting “Tweets” and comments posted in Arabic on Twitter, during the 2018 world cup event.

This study also measures the significance of analysing texts including or excluding emojis. The data is obtained from thousands of extracted tweets, to find the results of sentiment analysis for texts and emojis separately. Results show that emojis support the sentiment orientation of the texts and that texts or emojis cannot separately provide reliable information as they complement each other to give the intended meaning.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: LITERATURE REVIEW

- Sentiment Analysis or Opinion Mining is a technique used to analyse the emotion in a text. We can extract the attitude or the opinion of a piece of text and get insights on it.

- In the context of machine learning, you can think of Sentiment Analysis as a Classification problem where the text can either have a positive sentiment, a negative sentiment or a neutral one.

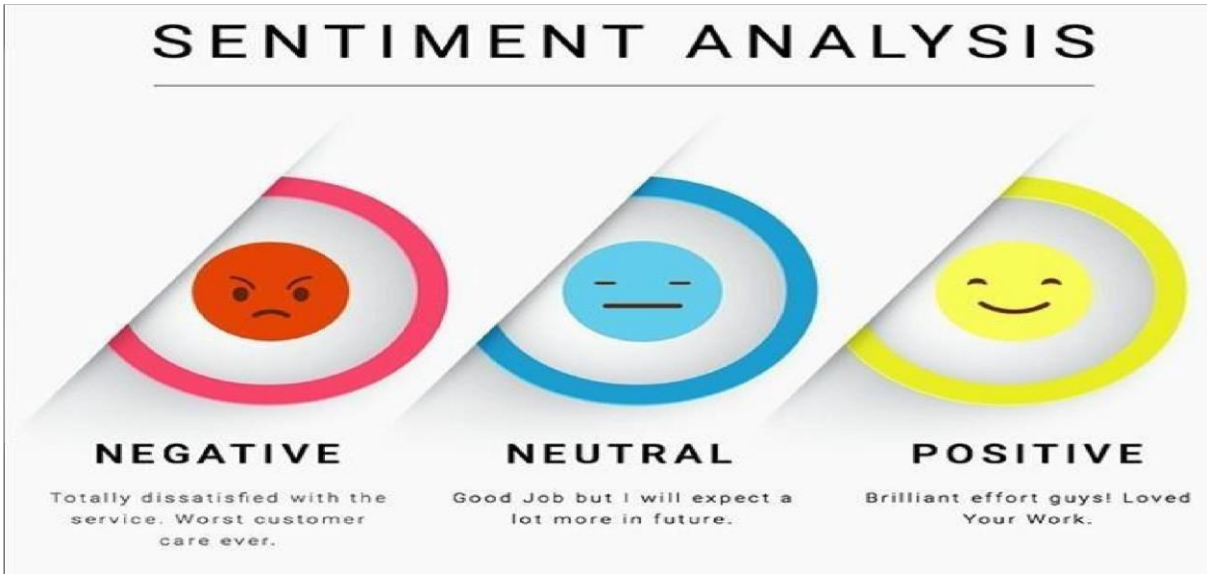


Fig 1.1 : Sentiment Analysis

Fig 1.2 : Emotion Analysis

1.1 What is Sentiment Analysis?

Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral. A sentiment analysis system for text analysis combines natural language processing (NLP) and machine learning techniques to assign weighted sentiment scores to the entities, topics, themes and categories within a sentence or phrase[1].

Sentiment analysis (or opinion mining) is also used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

Sentiment analysis helps data analysts within large enterprises gauge public opinion, conduct nuanced market research, monitor brand and product reputation, and understand customer experiences. In addition, data analytics companies often integrate third-party sentiment analysis APIs into their own customer experience management, social media monitoring, or workforce analytics platform, in order to deliver useful insights to their own customers.

1.2 Literature on Sentiment Analysis:

Sentiment analysis, also refers as opinion mining, is a sub machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral or negative. It is a really useful analysis since we could possibly determine the overall opinion about a selling objects, or predict stock markets for a given company like, if most people think positive about it, possibly its stock markets will increase, and so on. Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar,...) but it is also why it is very interesting to working on.

1.3 Emotion detection

This type of sentiment analysis aims to detect emotions, like happiness, frustration, anger, sadness, and so on. Many emotion detection systems use lexicons (i.e lists of words and the emotions they convey) or complex machine learning algorithms.

1.4 Multilingual sentiment analysis

Multilingual sentiment analysis can be difficult. It involves a lot of preprocessing and resources. Most of these resources are available online (e.g. sentiment lexicons), while others need to be created (e.g. translated corpora or noise detection algorithms), but you'll need to know how to code to use them.
Alternatively, you could detect language in texts automatically with MonkeyLearn's language classifier, then train a custom sentiment analysis model to classify texts in the language of your choice.

Basic sentiment analysis of text documents follows a straightforward process:

1. Break each text document down into its component parts (sentences, phrases, tokens and parts of speech)
2. Identify each sentiment-bearing phrase and component
3. Assign a sentiment score to each phrase and component (-1 to +1) 4. Optional: Combine scores for multi-layered sentiment analysis

1.5 How is machine learning used for sentiment analysis?

The primary role of machine learning in sentiment analysis is to improve and automate the low-level text analytics functions that sentiment analysis relies on, including Part of Speech tagging. For example, data scientists can train a machine learning model to identify nouns by feeding it a large volume of text documents containing pre-tagged examples. Using supervised and unsupervised machine learning techniques, such as neural networks and deep learning, the model will learn what nouns "look like".

Once the model is ready, the same data scientist can apply those training methods towards building new models to identify other parts of speech.

Machine learning also helps data analysts solve tricky problems caused by the evolution of language. For example, the phrase "sick burn" can carry many radically different meanings. Creating a sentiment analysis ruleset to account for every potential meaning is impossible. But if you feed a machine learning model with a few thousand pre-tagged examples, it can learn to understand what "sick burn" means in the context of video gaming, versus in the context of healthcare. And you can apply similar training methods to understand other double-meanings as well.

1.6 How Does Sentiment Analysis Work?

Sentiment analysis[6], otherwise known as opinion mining, works thanks to natural language processing (NLP) and machine learning algorithms, to automatically determine the emotional tone behind online conversations. There are different algorithms you can implement in sentiment analysis models, depending on how much data you need to analyze, and how accurate you need your model to be. We'll go over some of these in more detail, below.

1.7 Sentiment analysis algorithms fall into one of three buckets:

- Rule-based: these systems automatically perform sentiment analysis based on a set of manually crafted rules.
- Automatic: systems rely on machine learning techniques to learn from data.
- Hybrid systems combine both rule-based and automatic approaches.

1.7.1 Rule-based Approaches :

Usually, a rule-based system uses a set of human-crafted rules to help identify subjectivity, polarity, or the subject of an opinion.
These rules may include various NLP techniques developed in computational linguistics, such as:

- Stemming, tokenization, part-of-speech tagging and parsing.
- Lexicons (i.e. lists of words and expressions).

1.8 Here's a basic example of how a rule-based system works:

1. Defines two lists of polarized words (e.g. negative words such as bad, worst, ugly, etc and positive words such as good, best, beautiful, etc).
2. Counts the number of positive and negative words that appear in a given text.
3. If the number of positive word appearances is greater than the number of negative word appearances, the system returns a positive sentiment, and vice versa. If the numbers are even, the system will return a neutral sentiment.

1.8.1 Automatic Approaches :

Automatic methods, contrary to rule-based systems, don't rely on manually crafted rules, but on machine learning techniques. A sentiment analysis task is usually modeled as a classification problem, whereby a classifier is fed a text and returns a category, e.g. positive, negative, or neutral.

1.9 The Training and Prediction Processes

In the training process (a), our model learns to associate a particular input (i.e. a text) to the corresponding output (tag) based on the test samples used for training.
In the prediction process (b), the feature extractor is used to transform unseen text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted tags (again, positive, negative, or neutral).

1.9.1 Classification Algorithms

The classification step usually involves a statistical model like Naïve Bayes, Logistic Regression, Support Vector Machines, or Neural Networks:

- **Naïve Bayes:** a family of probabilistic algorithms that uses Bayes's Theorem to predict the category of a text.

- **Linear Regression:** a very well-known algorithm in statistics used to predict some value (Y) given a set of features (X).

- **Support Vector Machines:** a non-probabilistic model which uses a representation of text examples as points in a multidimensional space. Examples of different categories (sentiments) are mapped to distinct regions within that space. Then, new texts are assigned a category based on similarities with existing texts and the regions they're mapped to.

- **Deep Learning:** a diverse set of algorithms that attempt to mimic the human brain, by employing artificial neural networks to process data.

1.9.2 Hybrid Approaches :

Hybrid systems combine the desirable elements of rule-based and automatic techniques into one system. One huge benefit of these systems is that results are often more accurate.

CHAPTER 2 : METHODOLOGY

2.1 Dataset

In order to investigate the problem, a collection of product reviews from the Amazon.com is used as the dataset. Each review consists the textual comment about a given product and the rating given for that product (between 1 to 5). A total of 10,000 reviews are used for this analysis, which comprises of 2,000 reviews for each rating category.

2.2 Sentiment Analysis

Sentiment analysis algorithm is applied to each review to label the comment into positive, negative or neutral polarity. In this investigation, a simple sentiment analysis algorithm based on lexical dictionary is used. The dictionary contains 2,005 positive words and 4,783 negative words.

The lexical based approach for sentiment analysis relies on the distribution of opinion words within the text in order to predict the sentiment orientation of that text. The orientation is decided based on the occurrence of positive and negative words; more occurrences of positive words indicates that the text is positively oriented and vice versa. In the case of a balance, the text is assumed to be neutral. This approach has been shown to be effective in [5]. As such, it is assumed that the distribution of opinion words within comments will have a direct correlation with the ratings associated to the comments. The following are the phases of the sentiment analysis approach used in this paper:

1. First, in the pre-processing phase, each textual comment of the review is tokenized to generate bag of words. Duplicate words are removed to produce a list of unique words from the textual comment
2. Second, by comparing each word from the list to the entries in the lexical dictionary, the number of positive, and negative words is identified.
3. Finally, the textual comment is labeled positive if the number of positive words is more than the negative words, and vice versa. If the count is equal, the textual comment is labeled neutral.

CHAPTER 3 : TECHNOLOGIES USED

3.1 Python

Python[2] is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

3.2 Characteristics of Python

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.

3.3 Applications of Python

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

3.4 Pandas

Pandas[2] is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.[2] The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. [3] Its name is a play on the phrase "Python data analysis" itself.[4] Wes McKinney started building what would become pandas at AQR Capital while he was a researcher there from 2007 to 2010.

3.5 TextBlob Introduction

TextBlob is a Python (2 and 3) 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.

3.6 Lemmatization

Lemmatization refers to reducing the word to its root form as found in a dictionary.
To perform lemmatization via TextBlob, you have to use the `Word` object from the `textblob` library, pass it the word that you want to lemmatize and then call the `lemmatize` method.

3.7 Tweepy

If you are new to Tweepy[20], this is the place to begin. The goal of this tutorial is to get you set-up and rolling with Tweepy. We won't go into too much detail here, just some important basics.

3.7.1 Hello Tweepy

```
import tweepy

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

public_tweets = api.home_timeline()
for tweet in public_tweets:
    print tweet.text
```

This example will download your home timeline tweets and print each one of their texts to the console. Twitter requires all requests to use OAuth for authentication. The Authentication Tutorial goes into more details about authentication.

3.8 API

The API class provides access to the entire twitter RESTful API methods. Each method can accept various parameters and return responses. For more information about these methods please refer to API Reference.

3.9 Models

When we invoke an API method most of the time returned back to us will be a Tweepy model class instance. This will contain the data returned from Twitter which we can then use inside our application. For example the following code returns to us an User model:

```
# Get the User object for twitter...
user = api.get_user('twitter')
```

Models contain the data and some helper methods which we can then use:

```
print  user.screen_name
print  user.followers_count
for  friend  in  user.friends():
    print  friend.screen_name
```

3.10 Asgriref use in project

ASGI is a spiritual successor to WSGI, the long-standing Python standard for compatibility between web servers, frameworks, and applications.

WSGI succeeded in allowing much more freedom and innovation in the Python web space, and ASGI's goal is to continue this onward into the land of asynchronous Python.

3.10.1 What's wrong with WSGI?

You may ask "why not just upgrade WSGI"? This has been asked many times over the years, and the problem usually ends up being that WSGI's single-callable interface just isn't suitable for more involved Web protocols like WebSocket.

WSGI applications are a single, synchronous callable that takes a request and returns a response; this doesn't allow for long-lived connections, like you get with long-poll HTTP or WebSocket connections.

Even if we made this callable asynchronous, it still only has a single path to provide a request, so protocols that have multiple incoming events (like receiving WebSocket frames) can't trigger this.

3.10.2 How does ASGI work?

ASGI is structured as a single, asynchronous callable. It takes a scope, which is a dict containing details about the specific connection, send, an asynchronous callable, that lets the application send event messages to the client, and receive, an asynchronous callable which lets the application receive event messages from the client.

This not only allows multiple incoming events and outgoing events for each application, but also allows for a background coroutine so the application can do other things (such as listening for events on an external trigger, like a Redis queue).

In its simplest form, an application can be written as an asynchronous function, like this:

```
async def application(scope, receive, send):
    event = await receive()
    ...
    await send({"type": "websocket.send", ...})
```

Every event that you send or receive is a Python dict, with a predefined format. It's these event formats that form the basis of the standard, and allow applications to be swappable between servers.

These events each have a defined type key, which can be used to infer the event's structure. Here's an example event that you might receive from receive with the body from a HTTP request:

```
{
    "type": "http.request",
    "body": b"Hello World",
    "more_body": False,
}
```

And here's an example of an event you might pass to send to send an outgoing WebSocket message:

```
{
    "type": "websocket.send",
    "text": "Hello world!",
}
```

3.11 WSGI compatibility

ASGI is also designed to be a superset of WSGI, and there's a defined way of translating between the two, allowing WSGI applications to be run inside ASGI servers through a translation wrapper (provided in the asgiref library). A threadpool can be used to run the synchronous WSGI applications away from the async event loop.

3.12 Django

Django[7] is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

3.12.1 Django helps you write software that is:Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

3.12.2 Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with

any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django!

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.
Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (see Website security for more details of such attacks).

3.12.3 Scalable

Django uses a component-based “shared-nothing” architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

3.12.4 Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

3.12.5 Where did it come from?

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. After creating a number of sites, the team began to factor out and reuse lots of common code and design patterns. This common code evolved into a generic web development framework, which was open-sourced as the "Django" project in July 2005.

Django has continued to grow and improve, from its first milestone release (1.0) in September 2008 through to the recently-released version 3.1 (2020). Each release has added new functionality and bug fixes, ranging from support for new types of databases, template engines, and caching, through to the addition of "generic" view functions and classes (which reduce the amount of code that developers have to write for a number of programming tasks).

Django is now a thriving, collaborative open source project, with many thousands of users and contributors. While it does still have some features that reflect its origin, Django has evolved into a versatile framework that is capable of developing any type of website.

3.12.6 How popular is Django?

There isn't any readily-available and definitive measurement of popularity of server-side frameworks (although you can estimate popularity using mechanisms like counting the number of GitHub projects and StackOverflow questions for each platform). A better question is whether Django is "popular enough" to avoid the problems of unpopular platforms. Is it continuing to evolve? Can you get help if you need it? Is there an opportunity for you to get paid work if you learn Django?

Based on the number of high profile sites that use Django, the number of people contributing to the codebase, and the number of people providing both free and paid for support, then yes, Django is a popular framework!

High-profile sites that use Django include: Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest, and Open Stack (source: Django overview page).

3.12.7 Is Django opinionated?

Web frameworks often refer to themselves as "opinionated" or "unopinionated".

Opinionated frameworks are those with opinions about the "right way" to handle any particular task. They often support rapid development in a particular domain (solving problems of a particular type) because the right way to do anything is usually well-understood and well-documented. However they can be less flexible at solving problems outside their main domain, and tend to offer fewer choices for what components and approaches they can use.

Unopinionated frameworks, by contrast, have far fewer restrictions on the best way to glue components together to achieve a goal, or even what components should be used. They make

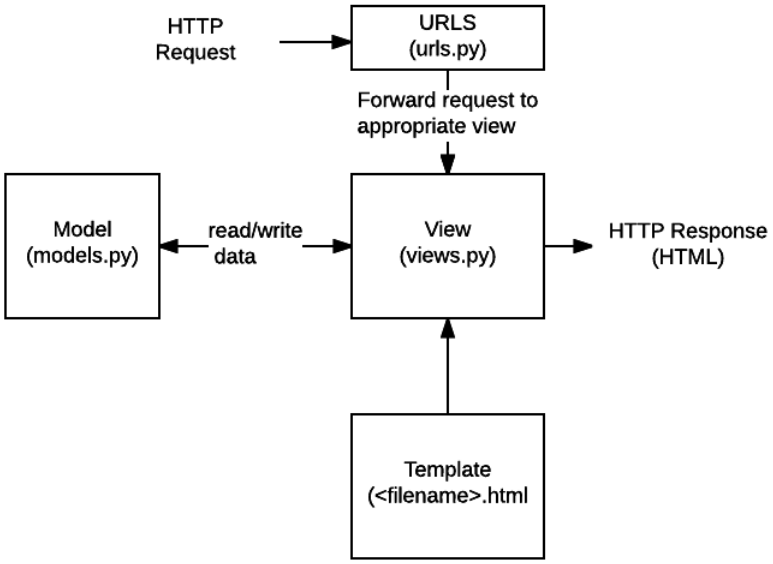
it easier for developers to use the most suitable tools to complete a particular task, albeit at the cost that you need to find those components yourself.

Django is "somewhat opinionated", and hence delivers the "best of both worlds". It provides a set of components to handle most web development tasks and one (or two) preferred ways to use them. However, Django's decoupled architecture means that you can usually pick and choose from a number of different options, or add support for completely new ones if desired.

3.12.8 What does Django code look like?

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template[19].

Django web applications typically group the code that handles each of these steps into separate files



- **URLs:** While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.

- **View:** A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via models, and delegate the formatting of the response to templates.

- **Models:** Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database.

- **Templates:** A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A view can dynamically create an HTML page using an HTML template, populating it with data from a model. A template can be used to define the structure of any type of file; it doesn't have to be HTML!

The sections below will give you an idea of what these main parts of a Django app look like (we'll go into more detail later on in the course, once we've set up a development environment).

3.12.9 Sending the request to the right view (urls.py)

A URL mapper is typically stored in a file named **urls.py**. In the example below, the mapper (urlpatterns) defines a list of mappings between routes (specific URL patterns) and corresponding view functions. If an HTTP Request is received that has a URL matching a specified pattern, then the associated view function will be called and passed the request.

```
urlpatterns = [

    path('admin/', admin.site.urls),

    path('book/<int:id>/', views.book_detail, name='book_detail'),

    path('catalog/', include('catalog.urls')),

    re_path(r'^([0-9]+)/$', views.best),

]
```

The urlpatterns object is a list of path() and/or re_path() functions (Python lists are defined using square brackets, where items are separated by commas and may have an optional trailing comma. For example: [item1, item2, item3,]).

The first argument to both methods is a route (pattern) that will be matched. The path() method uses angle brackets to define parts of a URL that will be captured and passed through to the view function as named arguments. The re_path() function uses a flexible pattern matching approach known as a regular expression. We'll talk about these in a later article!

The second argument is another function that will be called when the pattern is matched. The notation views.book_detail indicates that the function is called book_detail() and can be found in a module called views (i.e. inside a file named views.py)

3.12.10 Handling The Requests(Views.py)

Views are the heart of the web application, receiving HTTP requests from web clients and returning HTTP responses. In between, they marshal the other resources of the framework to access databases, render templates, etc.
The example below shows a minimal view function index(), which could have been called by our URL mapper in the previous section. Like all view functions it receives an HttpRequest object as a parameter (request) and returns an HttpResponse object. In this case we don't do anything with the request, and our response returns a hard-coded string. We'll show you a request that does something more interesting in a later section.

```
# filename: views.py (Django view functions)

from django.http import HttpResponse



def index(request):

    # Get an HttpRequest - the request parameter

    # perform operations using information from the request.

    # Return HttpResponse

    return HttpResponse('Hello from Django!')
```

3.12.11 Defining data models (models.py)

Django web applications manage and query data through Python objects referred to as models. Models define the structure of stored data, including the field types and possibly also their maximum size, default values, selection list options, help text for documentation, label text for forms, etc. The definition of the model is independent of the underlying database — you can choose one of several as part of your project settings. Once you've chosen what database you want to use, you don't need to talk to it directly at all — you just write your model structure and other code, and Django handles all the "dirty work" of communicating with the database for you.

The code snippet below shows a very simple Django model for a Team object. The Team class is derived from the django class models.Model. It defines the team name and team level as character fields and specifies a maximum number of characters to be stored for each record. The team_level can be one of several values, so we define it as a choice field and provide a mapping between choices to be displayed and data to be stored, along with a default value.

3.12.12 # filename: models.py

```
from django.db import models
```

```
class Team(models.Model):

 team_name = models.CharField(max_length=40)



    TEAM_LEVELS = (

        ('U09', 'Under 09s'),

        ('U10', 'Under 10s'),

        ('U11', 'Under 11s'),

        ...  #list other team levels

    )

    team_level = models.CharField(max_length=3, choices=TEAM_LEVELS, default='U11')
```

3.12.13 Querying data (views.py)

The Django model provides a simple query API for searching the associated database. This can match against a number of fields at a time using different criteria (e.g. exact, case-insensitive, greater than, etc.), and can support complex statements (for example, you can specify a search on U11 teams that have a team name that starts with "Fr" or ends with "al").

The code snippet shows a view function (resource handler) for displaying all of our U09 teams. The list_teams = Team.objects.filter(team_level__exact="U09") line shows how we can use the model query API to filter for all records where the team_level field has exactly the text 'U09' (note how this criteria is passed to the filter() function as an argument, with the field name and match type separated by a double underscore: **team_level__exact**).

```
## filename: views.py



from django.shortcuts import render

from .models import Team

def index(request):

    list_teams = Team.objects.filter(team_level__exact="U09")

    context = {'youngest_teams': list_teams}

    return render(request, '/best/index.html', context)
```

This function uses the render() function to create the HttpResponse that is sent back to the browser. This function is a shortcut; it creates an HTML file by combining a specified HTML template and some data to insert in the template (provided in the variable named "context"). In the next section we show how the template has the data inserted in it to create the HTML.

3.12.14 Rendering data (HTML templates)

Template systems allow you to specify the structure of an output document, using placeholders for data that will be filled in when a page is generated. Templates are often used to create HTML, but can also create other types of document. Django supports both its native templating system and another popular Python library called Jinja2 out of the box (it can also be made to support other systems if needed).

The code snippet shows what the HTML template called by the render() function in the previous section might look like. This template has been written under the assumption that it will have access to a list variable called youngest_teams when it is rendered (this is contained in the context variable inside the render() function above). Inside the HTML skeleton we have an expression that first checks if the youngest_teams variable exists, and then iterates it in a for loop. On each iteration the template displays each team's team_name value in an <li> element.

3.12.15 # filename: best/templates/best/index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <title>Home page</title>

</head>

<body>

  {% if youngest_teams %}

    <ul>

      {% for team in youngest_teams %}

        <li>{{ team.team_name }}</li>
```

```
    {% endfor %}

  </ul>

 {% else %}

  <p>No teams are available.</p>

 {% endif %}

</body>

</html>
```

3.13 NumPY

NumPy[24] is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with

the corresponding element in another sequence of the same length. If the data are stored in two Python lists, a and b, we could iterate over each element:

```
c = []
for i in range(len(a)):
    c.append(a[i]*b[i])
```

This produces the correct answer, but if a and b each contain millions of numbers, we will pay the price for the inefficiencies of looping in Python. We could accomplish the same task much more quickly in C by writing (for clarity we neglect variable declarations and initializations, memory allocation, etc.)

```
for (i = 0; i < rows; i++): {
  c[i] = a[i]*b[i];
}
```

This saves all the overhead involved in interpreting the Python code and manipulating Python objects, but at the expense of the benefits gained from coding in Python. Furthermore, the coding work required increases with the dimensionality of our data. In the case of a 2-D array, for example, the C code (abridged as before) expands to

```
for (i = 0; i < rows; i++): {
  for (j = 0; j < columns; j++): {
    c[i][j] = a[i][j]*b[i][j];
  }
}
```

NumPy gives us the best of both worlds: element-by-element operations are the "default mode" when an ndarray is involved, but the element-by-element operation is speedily executed by pre-compiled C code. In NumPy

```
c = a * b
```

does what the earlier examples do, at near-C speeds, but with the code simplicity we expect from something based on Python. Indeed, the NumPy idiom is even simpler! This last example illustrates two of NumPy's features which are the basis of much of its power: vectorization and broadcasting.

3.13.1 Why is NumPy Fast?

Vectorization describes the absence of any explicit looping, indexing, etc., in the code - these things are taking place, of course, just "behind the scenes" in optimized, pre-compiled C code. Vectorized code has many advantages, among which are:

- vectorized code is more concise and easier to read
- fewer lines of code generally means fewer bugs
- the code more closely resembles standard mathematical notation (making it easier, typically, to correctly code mathematical constructs)
- vectorization results in more "Pythonic" code. Without vectorization, our code would be littered with inefficient and difficult to read for loops.

Broadcasting is the term used to describe the implicit element-by-element behavior of operations; generally speaking, in NumPy all operations, not just arithmetic operations, but logical, bit-wise, functional, etc., behave in this implicit element-by-element fashion, i.e., they broadcast. Moreover, in the example above, a and b could be multidimensional arrays of the same shape, or a scalar and an array, or even two arrays of with different shapes, provided that the smaller array is "expandable" to the shape of the larger in such a way that the resulting broadcast is unambiguous. For detailed "rules" of broadcasting see basics.broadcasting.

3.13.2 Who Else Uses NumPy?

NumPy fully supports an object-oriented approach, starting, once again, with ndarray. For example, ndarray is a class, possessing numerous methods and attributes. Many of its methods are mirrored by functions in the outer-most NumPy namespace, allowing the programmer to code in whichever paradigm they prefer. This flexibility has allowed the NumPy array dialect and NumPy ndarray class to become the de-facto language of multi-dimensional data interchange used in Python.

3.14 Python pytz

pytz brings the Olson tz database into Python. This library allows accurate and cross platform timezone calculations using Python 2.4 or higher. It also solves the issue of ambiguous times at the end of daylight saving time, which you can read more about in the Python Library Reference (datetime.tzinfo).

Almost all of the Olson timezones are supported.

This library differs from the documented Python API for tzinfo implementations; if you want to create local wallclock times you need to use the localize() method documented in this document. In addition, if you perform date arithmetic on local times that cross DST boundaries, the result may be in an incorrect timezone (ie. subtract 1 minute from 2002-10-27 1:00 EST and you get 2002-10-27 0:59 EST instead of the correct 2002-10-27 1:59 EDT). A normalize() method is provided to correct this. Unfortunately these issues cannot be resolved without modifying the Python datetime implementation (see PEP-431)

3.14 OAuthLib

OAuthLib is a framework which implements the logic of OAuth1 or OAuth2 without assuming a specific HTTP request object or web framework. Use it to graft OAuth client support onto your favorite HTTP library, or provide support onto your favourite web framework. OAuthLib maintainer of such a library, write a thin veneer on top of OAuthLib and get OAuth support for very little effort.

3.14.1 Which web frameworks are supported?

The following packages provide OAuth support using OAuthLib.

- For Django there is django-oauth-toolkit, which includes Django REST framework support.
- For Flask there is flask-oauthlib and Flask-Dance.

- For Pyramid there is pyramid-oauthlib.
- For Bottle there is bottle-oauthlib.

3.15 HyperText Markup Language – HTML

HyperText Markup Language (HTML)[10] is the set of markup symbols or codes inserted into a file intended for display on the Internet. The markup tells web browsers how to display a web page's words and images.

Each individual piece markup code (which would fall between "<" and ">" characters) is referred to as an element, though many people also refer to it as a tag. Some elements come in pairs that indicate when some display effect is to begin and when it is to end.

- HyperText Markup Language (HTML) is the basic scripting language used by web browsers to render pages on the world wide web.
- HyperText allows a user to click a link and be redirected to a new page referenced by that link.
- Early versions of HTML were static (Web 1.0), while newer iterations feature a great deal of dynamic flexibility (Web 2.0, 3.0).
- Markup is the text that appears between two pointed brackets (e.g., <footnote>), and content is everything else.

3.15.1 HTML Explained

Fig 3.1 : HTML 5

HyperText Markup Language[11] is the computer language that facilitates website creation. The language, which has code words and syntax just like any other language, is relatively easy to comprehend and, as time goes on, increasingly powerful in what it allows someone to create. HTML continues to evolve to meet the demands and requirements of the Internet under the guise of the World Wide Web Consortium, the organization that designs and maintains the language; for instance, with the transition to Web 2.0.

HyperText is the method by which Internet users navigate the web. By clicking on special text called hyperlinks, users are brought to new pages. The use of hyper means it is not linear, so users can go anywhere on the Internet simply by clicking on the available links. Markup is what HTML tags do to the text inside of them; they mark it as a specific type of text. For

example, markup text could come in the form of boldface or italicized type to draw specific attention to a word or phrase.

3.15.2 HyperText Markup Language Basics

At its core, HTML[12] is a series of short codes typed into a text-file. These are the tags that power HTML's capabilities. The text is saved as an HTML file and viewed through a web browser. The browser reads the file and translates the text into a visible form, as directed by the codes the author used to write what becomes the visible rendering. Writing HTML requires tags to be used correctly to create the author's vision.

The tags are what separate normal text from HTML code. Tags are the words between what are known as angle-brackets, which allow graphics, images, and tables to appear on the webpage. Different tags perform different functions. The most basic tags apply formatting to text. As web interfaces need to become more dynamic, Cascading Style Sheets (CSS) and JavaScript applications may be used. CSS makes web pages more accessible and JavaScript adds power to basic HTML.

3.15.3 INTRODUCTION TO HTML TAGS

Tags can be defined as the instructions which are being directly embedded in the text of an HTML document. The types of tags used in the HTML document are responsible to tell a web browser to do something (follow the instruction) instead of just displaying text. In an HTML document, all tag names are differentiated from other simple text. The tag names are enclosed in between angle brackets or a 'less than' and a 'greater than' symbol, (<) and (>)[13].

Top 3 Types of Tags in HTML

An HTML document is created using different types of tags. HTML tags can be defined and divided based on a different basis. Let's see them in the coming parts of this article. We have divided HTML tags based on the following classifications:

1. Paired and Unpaired Tags

Following are the paired and unpaired tags in HTML explained in detail with the help of examples.

Paired Tags

An HTML tag is known as a paired tag when the tag consists of an opening tag and a closing tag as its companion tag. An HTML Paired tag starts with an opening tag: the tag name enclosed inside the angle brackets; for example, a paragraph opening tag is written as ' '. The content follows the opening tag, which ends with an ending tag: the tag name starting with a forward slash; for example, an ending paragraph tag is written as ' '. The first tag can be referred to as the 'Opening Tag', and the second tag can be called Closing Tag.

Unpaired Tags

An HTML tag is called an unpaired tag when the tag only has an opening tag and does not have a closing tag or a companion tag. The Unpaired HTML tag does not require a closing tag; an opening tag is sufficient in this type. Unpaired tags are sometimes also named as Standalone Tags or Singular Tags since they do not require a companion tag.

2. Self-Closing Tags

Self-Closing Tags are those HTML tags that do not have a partner tag, where the first tag is the only necessary tag that is valid for the formatting. The main and important information is contained WITHIN the element as its attribute. An image tag is the classic example of a self-closing tag.

3. Utility-Based Tags

The HTML tags can be widely differentiated on the basis of their utility, that is, on the basis of the purpose they serve. We can divide them basically into three categories as discussed below:

Formatting Tags

The HTML tags that help us in the formatting of the texts like the size of the text, font styles, making a text bold, etc. This is done using tags like <font>, <b>, <u>, etc. Tables, divisions, and span tags are also those tags that help format a web page or document and set the layout of the page

Structure Tags

The HTML tags that help in structuring the HTML document are called Structure Tags. Description, head, html, title, body, etc., form the group of the page structure tags. The structure tags only assist in creating or forming the basic html page from the root; that is, they do not affect or has any hand in the formatting of texts.

3.15.4 Project Homepage HTML CODE : Home.html

```
<!DOCTYPE html>
 <html lang="en">
 {% load static %}
   <head>
     <title>Social Media Sentiment Analysis</title>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
     <link href="https://fonts.googleapis.com/css?family=Rubik:300,400,500" rel="stylesheet">
     <link                href="https://fonts.googleapis.com/css?family=Poppins:100,300,500"
rel="stylesheet">

     <!-- insertion templates -->
```

```
    <link                                                                    rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Open+Sans:300,400">
          <link rel="stylesheet" href="{% static 'insertion/css/tooplate-style.css' %}">
          <link rel="stylesheet" href="{% static 'insertion/css/fontawesome-all.min.css' %}">
  </head>
  <body>

    <nav class="navbar navbar-expand-lg navbar-dark ftco_navbar bg-dark ftco-navbar-light"
id="ftco-navbar" data-aos="fade-down" data-aos-delay="500">
      <div class="container">
        <a class="navbar-brand" href="/">Home</a>
        <div class="collapse navbar-collapse" id="ftco-nav">
          <ul class="navbar-nav ml-auto">
           <li       class="nav-item"><a       href="/sentiment"       class="nav-link">Sentiment
Analysis</a></li>
          </ul>
        </div>
      </div>
    </nav>
    <!-- END nav -->


    <section class="ftco-cover"  id="section-home" style="background-image: url({% static
'home/images/twitter_back.jpg' %});"  data-aos="fade"  data-stellar-background-ratio="0.5">
      <div class="container">
        <div class="row align-items-center ftco-vh-100">
          <div class="col-md-7">
            <h1 class="ftco-heading mb-3" data-aos="fade-up" data-aos-delay="500">Welcome
To The World Of Tweets</h1>
           <h2 class="h5 ftco-subheading mb-5" data-aos="fade-up"          data-aos-
delay="600">Tweet Sentiment/Emotion Analysis</h2>
             <div class="row">
             <div class="box counter-left md-6 col-lg-4" data-aos="fade-up" data-aos-
delay="400">
                <h1 class="counter" style="color: #6cbb23">350632</h1>
                <p>Tweets sent per minute</p>
                  </div>
                <div class="box counter-left md-6 col-lg-4" data-aos="fade-up" data-aos-
delay="400">
             <h1 class="counter" style="color: #6cbb23">321056231</h1>
             <p>Total Number of monthly active users on Twitter</p>
                </div>
                <div class="box counter-left md-6 col-lg-4" data-aos="fade-up" data-aos-
delay="400">
             <h1 class="counter" style="color:#6cbb23">7447680</h1>
             <p>Minutes Since First Tweet Ever</p>
                </div>
             </div>
            {% if user.is_authenticated == False %}
```

```
            <div>
              <br>
              <br>
              <br>
            <h3 style="color:White" data-aos="fade-up" data-aos-delay="500">Perform <a
href="/sentiment" style="color:#D7C004">Sentiment</a> Analysis on Tweets</h3>
          </div>
          {% endif %}
          </div>
          </div>
          </div>
    </section>
    <!-- END section -->

    <section class="ftco-cover" id="section-about">
      <div class="container mx-auto" data-aos="fade-up">
        <!-- <div class="row"> -->
          <!-- <div class="col-12  mb-5" data-aos="fade-up"> -->
            <h1>"Twitter is a great place to tell the world what you're thinking before you've had
a chance to think about it." - Chris Pirillo</h1>
          <!-- </div> -->
        <!-- </div> -->

      </div>
    </section>
    <!-- END section -->



    <section class="ftco-section tm-bg-pink-light">
      <div class="row container grid mx-auto" data-aos="fade-up">
        <div class="col-6 tm-album-col">
          <a href="/sentiment">
          <figure class="effect-sadie">
            <img src="{% static 'insertion/img/insertion-260x390-01.jpg' %}" alt="Image"
class="img-fluid">
            <figcaption>
              <h2 style="color:white"><b>Sentiment Analysis</b></h2>
              <p>Categorises tweet under Positive, Neutral or Negative</p>
            </figcaption>
          </figure>
      </a>
        </div>
        <div class="col-6 tm-album-col mx-auto">
          <a href="/emotion">
          <figure class="effect-sadie">
            <img src="{% static 'insertion/img/insertion-260x390-02.jpg' %}" alt="Image"
class="img-fluid">
          </figure>
```

```
      </a>
        </div>
      </div>
    </section>
  <div class="ftco-section row justify-content-center text-center" data-aos="fade-up">
        <h2>Developer</h2>
  </div>
      <section class="ftco-section tm-bg-pink-light">
        <div class="col-3 mx-auto"  data-aos="fade-up">

          <div class="flip-container">
            <div class="flipper">
              <div class="front" style="background-image: url({% static 'home/images/puneet.jpg'
%});">
          </div> <!-- .flip-container -->
        </div>
    </section>
      </div>

  <!-- loader -->
  <div id="ftco-loader" class="show fullscreen"><svg class="circular" width="48px"
height="48px"><circle class="path-bg" cx="24" cy="24" r="22" fill="none" stroke-width="4"
stroke="#eeeeee"/><circle class="path" cx="24" cy="24" r="22" fill="none" stroke-width="4"
stroke-miterlimit="10" stroke="#F96D00"/></svg></div>

  <script src="{% static 'register/js/jquery.min.js' %}"></script>

    <!-- insertion static -->


    <!-- <script src="{% static 'insertion/js/jquery-3.2.1.slim.min.js' %}"></script> -->

  </body>
 </html>
```

3.16 CSS (Cascading Style Sheets)

Fig 3.2 : CSS

CSS[14] stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

3.16.1 What does CSS do

- You can add new looks to your old HTML documents.
- You can completely change the look of your website with only a few changes in CSS code.

3.16.2 Why use CSS

These are the three major benefits of CSS:

1) Solves a big problem

Before CSS[15], tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will be become a long and expensive process. CSS was created to solve this problem. It was a W3C recommendation.

2) Saves a lot of time

CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

3.16.3 CSS Syntax

A CSS rule set contains a selector and a declaration block.

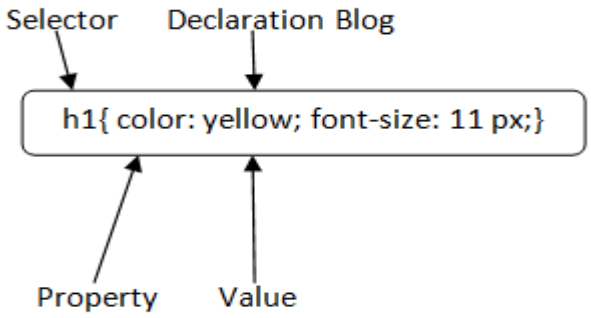


Fig 3.3 : CSS Syntax

Selector: Selector indicates the HTML element you want to style. It could be any tag like <h1>, <title> etc.
Declaration Block: The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

1. color: yellow;
2. font-size: 11 px;

Each declaration contains a property name and value, separated by a colon.

Property: A Property is a type of attribute of HTML element. It could be color, border etc.

Value: Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

3.16.4 Project CSS Code : Style.css

```
:root {
  --blue: #007bff;
  --indigo: #6610f2;
  --purple: #6f42c1;
  --pink: #e83e8c;
  --red: #dc3545;
  --orange: #fd7e14;
  --yellow: #ffc107;
  --green: #28a745;
  --teal: #20c997;
  --cyan: #17a2b8;
  --white: #fff;
  --gray: #6c757d;
  --gray-dark: #343a40;
```

```
  --primary: #4ECCA3;
  --secondary: #6c757d;
  --success: #28a745;
  --info: #17a2b8;
  --warning: #ffc107;
  --danger: #dc3545;
  --light: #f8f9fa;
  --dark: #343a40;
  --breakpoint-xs: 0;
  --breakpoint-sm: 576px;
  --breakpoint-md: 768px;
  --breakpoint-lg: 992px;
  --breakpoint-xl: 1200px;
  --font-family-sans-serif: "Rubik", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
  --font-family-monospace: SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace; }

*,
*::before,
*::after {
  -webkit-box-sizing: border-box;
  box-sizing: border-box; }

html {
  font-family: sans-serif;
  line-height: 1.15;
  -webkit-text-size-adjust: 100%;
  -ms-text-size-adjust: 100%;
  -ms-overflow-style: scrollbar;
  -webkit-tap-highlight-color: transparent; }

@-ms-viewport {
  width: device-width; }

article, aside, dialog, figcaption, figure, footer, header, hgroup, main, nav, section {
  display: block; }

body {
  margin: 0;
  font-family: "Rubik", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
  font-size: 1rem;
  font-weight: 400;
  line-height: 1.5;
  color: #212529;
  text-align: left;
  background-color: #fff;
```

```
}

[tabindex="-1"]:focus {
  outline: 0 !important; }

hr {
  -webkit-box-sizing: content-box;
  box-sizing: content-box;
  height: 0;
  overflow: visible; }

h1, h2, h3, h4, h5, h6 {
  margin-top: 0;
  margin-bottom: 0.5rem; }

p {
  margin-top: 0;
  margin-bottom: 1rem; }

abbr[title],
abbr[data-original-title] {
  text-decoration: underline;
  -webkit-text-decoration: underline dotted;
  text-decoration: underline dotted;
  cursor: help;
  border-bottom: 0; }

address {
  margin-bottom: 1rem;
  font-style: normal;
  line-height: inherit; }
```

3.17 JavaScript

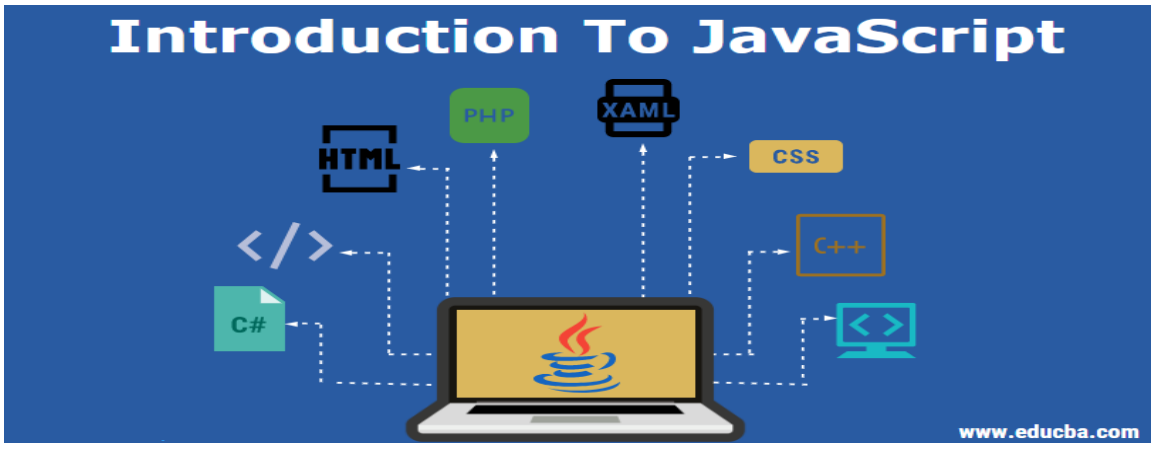


Fig 3.4 : Javascript

JavaScript is a lightweight, cross-platform, and interpreted scripting language. It is well-known for the development of web pages, many non-browser environments also use it. JavaScript can be used for **Client-side** developments as well as **Server-side** developments. JavaScript contains a standard library of objects, like **Array**, **Date**, and **Math**, and a core set of language elements like **operators**, **control structures**, and **statements**.

- **Client-side:** It supplies objects to control a browser and its Document Object Model (DOM). Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as **mouse clicks**, **form input**, and **page navigation**. Useful libraries for the client-side are **AngularJS**, **ReactJS**, **VueJS** and so many others.

- **Server-side:** It supplies objects relevant to running JavaScript on a server. Like if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server. The useful framework which is the most famous these days is **node.js**.

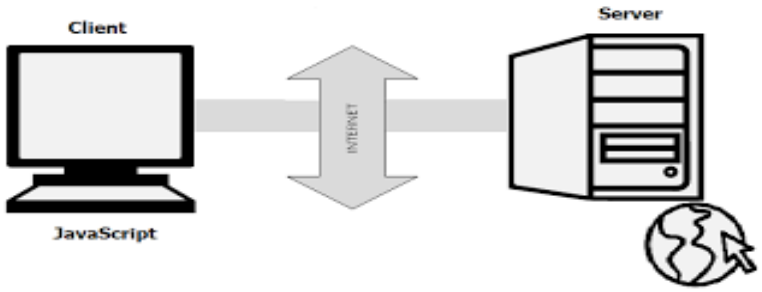


Fig 3.5 : node.js Server Side

3.17.1 JavaScript can be added to your HTML file in two ways:

- **Internal JS:** We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.
- **External JS:** We can write JavaScript code in other file having an extension .js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

Syntax:

```
<script>
   // JavaScript Code
</script>
```

History of JavaScript: It was created in 1995 by Brendan Eich while he was an engineer at Netscape. It was originally going to be named LiveScript but was renamed. Unlike most

programming languages, the JavaScript language has no concept of input or output. It is designed to run as a scripting language in a host environment, and it is up to the host environment to provide mechanisms for communicating with the outside world. The most common host environment is the browser.

Features of JavaScript: According to a recent survey conducted by **Stack Overflow**, JavaScript is the most popular language on earth.
With advances in browser technology and JavaScript having moved into the server with Node.js and other frameworks, JavaScript is capable of so much more. Here are a few things that we can do with JavaScript:

- JavaScript was created in the first place for DOM manipulation. Earlier websites were mostly static, after JS was created dynamic Web sites were made.
- Functions in JS are objects. They may have properties and methods just like another object. They can be passed as arguments in other functions.
- Can handle date and time.
- Performs Form Validation although the forms are created using HTML.
- No compiler needed.

3.17.2 Applications of Javascript

- **Web Development:** Adding interactivity and behavior to static sites JavaScript was invented to do this in 1995. By using AngularJS that can be achieved so easily.

- **Web Applications:** With technology, browsers have improved to the extent that a language was required to create robust web applications. When we explore a map in Google Maps then we only need to click and drag the mouse. All detailed view is just a click away, and this is possible only because of JavaScript. It uses Application Programming Interfaces(APIs) that provide extra power to the code. The Electron and React is helpful in this department.

3.18 BOOTSTRAP

- Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.
- It is absolutely free to download and use.
- It is a front-end framework used for easier and faster web development.
- It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.
- It can also use JavaScript plug-ins.
- It facilitates you to create responsive designs.

3.18.1 Why use Bootstrap

Following are the main advantage of Bootstrap:

- It is very easy to use. Anybody having basic knowledge of HTML and CSS can use Bootstrap.
- It facilitates users to develop a responsive website.
- It is compatible on most of browsers like Chrome, Firefox, Internet Explorer, Safari and Opera etc.

3.18.2 Project Javascript Code: main.js

```
:root {
  --blue: #007bff;
  --indigo: #6610f2;
  --purple: #6f42c1;
  --pink: #e83e8c;
  --red: #dc3545;
  --orange: #fd7e14;
  --yellow: #ffc107;
  --green: #28a745;
  --teal: #20c997;
  --cyan: #17a2b8;
  --white: #fff;
  --gray: #6c757d;
  --gray-dark: #343a40;
  --primary: #4ECCA3;
  --secondary: #6c757d;
  --success: #28a745;
  --info: #17a2b8;
  --warning: #ffc107;
  --danger: #dc3545;
  --light: #f8f9fa;
  --dark: #343a40;
  --breakpoint-xs: 0;
  --breakpoint-sm: 576px;
  --breakpoint-md: 768px;
  --breakpoint-lg: 992px;
  --breakpoint-xl: 1200px;
  --font-family-sans-serif: "Rubik", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
  --font-family-monospace: SFMono-Regular, Menlo, Monaco, Consolas, "Liberation Mono", "Courier New", monospace; }

*,
*::before,
```

```
*::after {
  -webkit-box-sizing: border-box;
  box-sizing: border-box; }

html {
  font-family: sans-serif;
  line-height: 1.15;
  -webkit-text-size-adjust: 100%;
  -ms-text-size-adjust: 100%;
  -ms-overflow-style: scrollbar;
  -webkit-tap-highlight-color: transparent; }

@-ms-viewport {
  width: device-width; }

article, aside, dialog, figcaption, figure, footer, header, hgroup, main, nav, section {
  display: block; }

body {
  margin: 0;
  font-family: "Rubik", -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica
Neue", Arial, sans-serif, "Apple Color Emoji", "Segoe UI Emoji", "Segoe UI Symbol";
  font-size: 1rem;
  font-weight: 400;
  line-height: 1.5;
  color: #212529;
  text-align: left;
  background-color: #fff;
}

[tabindex="-1"]:focus {
  outline: 0 !important; }

hr {
  -webkit-box-sizing: content-box;
  box-sizing: content-box;
  height: 0;
  overflow: visible; }

h1, h2, h3, h4, h5, h6 {
  margin-top: 0;
  margin-bottom: 0.5rem; }

p {
  margin-top: 0;
  margin-bottom: 1rem; }

abbr[title],
```

```
abbr[data-original-title] {
  text-decoration: underline;
  -webkit-text-decoration: underline dotted;
  text-decoration: underline dotted;
  cursor: help;
  border-bottom: 0; }
```

3.19 Final Output of Homepage

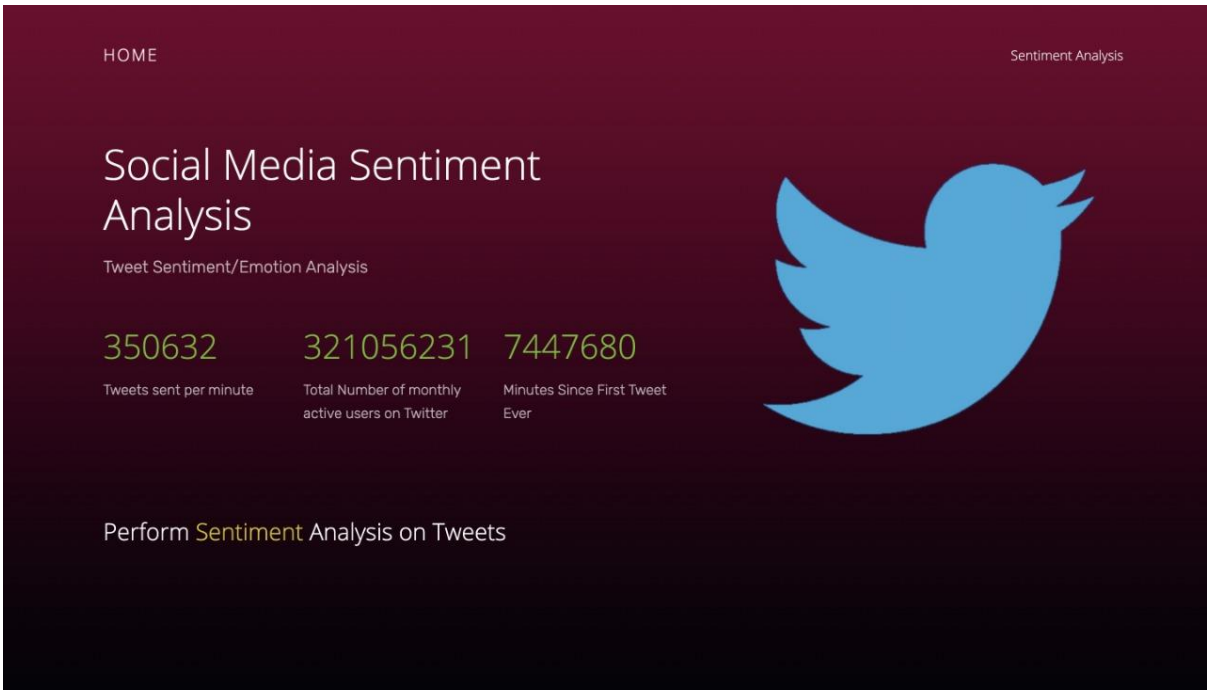


Fig 3.6 : Home Page

CAPTER 4 : PROPOSED SYSTEM

4.1 System Working

Sentiments[9] are the words or sentences that represent view or opinion that is held or expressed that can be positive, negative or neutral. We are going to propose a novel hybrid approach involving both corpus-based and dictionary-based techniques, which will find the semantic orientation of the sentiments words in tweets. We will also consider features like emoticons, neutralization, negation handling and capitalization as they have recently become a huge part of the internet language.

The proposed Sentiment Analysis on twitter data is based on two important parts viz Data Extraction, pre-processing of extracted data and classification.

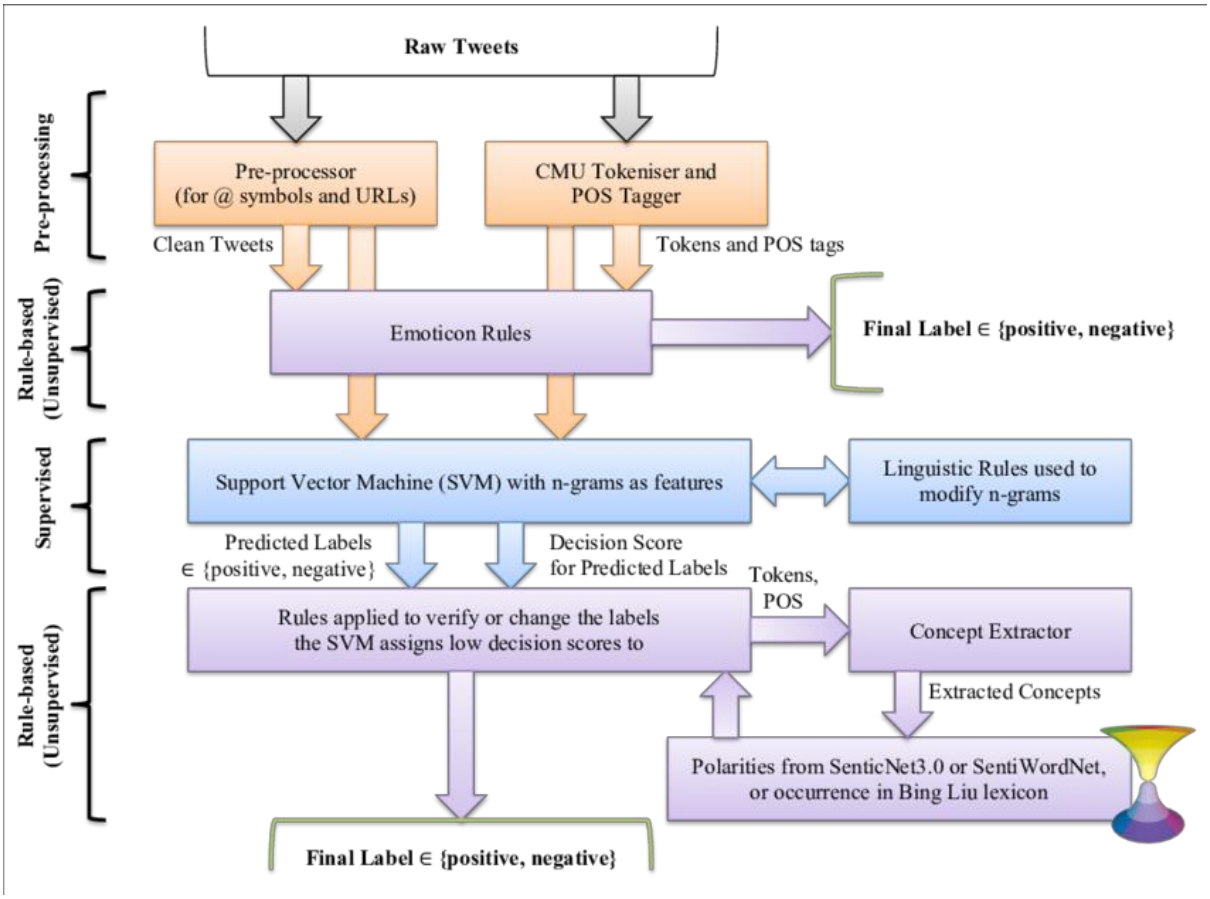


Fig 4.1 : Working of Sentiment Analysis

To uncover the sentiments, we will first extract the opinion words from tweets and then we find out their orientation, i.e., to decide whether each sentiment word reflects exaggerated and self-indulgent feelings of tenderness, sadness, or nostalgia.

The following steps will expound the process of the proposed system which is shown in above fig:

1. Retrieval of tweets

2. Pre-processing of extracted data

3. Parallel processing

4.Sentiment scoring module

5. Output sentiment

- **These steps are explained below:**

4.1.1 Retrieval of tweets :

As twitter is the most exaggerated part of social networking site, it consists of various blogs which are related to various topics worldwide. Instead of taking whole blogs, we will rather search on particular topic and download all its web pages then extracted them in the form of text files by using mining tool i.e. Weka which provides sentiment classifier..

4.1.2 Pre-processing of extracted data:

After retrieval of tweets Sentiment analysis tool is applied on raw tweets but in most of cases results to very poor performance. Therefore, preprocessing techniques are necessary for obtaining better results as given in. We extract tweets i.e. short messages from twitter which are used as raw data. This raw data needs to be preprocessed. So, preprocessing involves following steps which constructs n-grams:

4.1.3 Filtering:

Filtering is nothing but cleaning of raw data. In this step, URL links (E.g. http://twitter.com), special words in twitter (e.g. “RT” which means ReTweet), user names in twitter (e.g. @Ron - @ symbol indicating a user name), emoticons are removed.

4.2 Tokenization:

Tokenization is nothing but Segmentation of sentences. In this step, we will tokenize or segment text with the help of splitting text by spaces and punctuation marks to form container of words.

i. Removal of Stopwords:

Articles such as “a”, “an”, “the” and other stopwords such as “to”, “of”, “is”, “are”, “this”, “for” removed in this step.

ii. Construction of n-grams:

Set of n-grams can make out of consecutive words. Negation words such as “no”, “not” is attached to a word which follows or precedes it. For Instance: “I do not like remix music” has two bigrams: “I do+not”, “do+not like”, “not+like remix music”. So the accuracy of the classification improves by such procedure, because negation plays an important role in sentiment analysis. Paper [3] represents that negation needs to be taken into account, because it is a very common linguistic construction that affects polarity.

4.3 Parallel processing:

Sentiment classifier which classifies the sentiments builds using multinomial Naïve Bayes Classifier or Support Vector Machines (SVMs). Training of classifier data is the main motive of this step. Every database has hidden information which can be used for decision-making. Classification and prediction are two forms of data analysis which can be used to extract models describing important data and future trends. Classification is process of finding a set of models or functions that describe and distinguish data classes or concepts, for the purpose of being able to use the model for predicting the class of objects whose class label is unknown.

The derived model is based on the analysis of a set of training data. Training data consists of data objects whose class labels are known. The derived model can be represented in various forms, such as classification (IF-THEN) rules, decision trees, mathematical formulae, or neural networks.

Classification process is done in a two step process. First step is Model Construction in which we will build a model from the training set. And step2 is Model Usage in which we will check the accuracy of the model and use it for classifying new data.

4.4 Sentiment scoring module:

Prior polarity of words is the basic of our number of features. The dictionary is used in [1] in which English language words assigns a score to every word, between 1 (Negative) to 3 (Positive). So, this scoring module is going to determine score of sentiments in the sentiment analysis of data.

Output sentiment:

Based on the dictionary assignment of score, the proposed system interprets whether the tweet is positive, negative or neutral.

CHAPTER 5 : MACHINE LEARNING

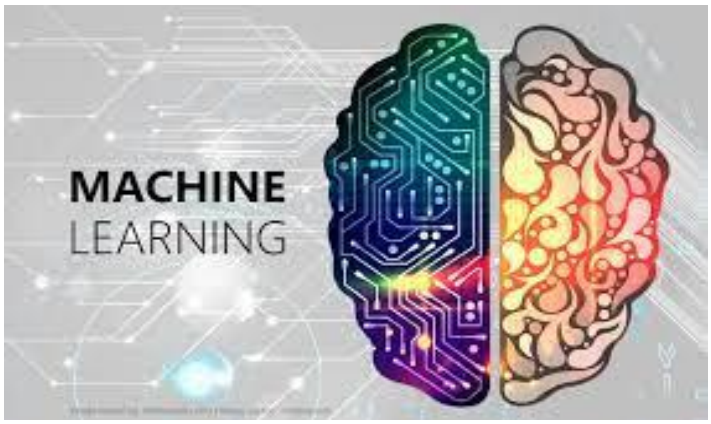


Fig 5.1 : Machine Learning

Machine learning (ML)[18] is a branch of artificial intelligence (AI) that enables computers to "self-learn" from training data and improve over time, without being explicitly programmed. Machine learning algorithms are able to detect patterns in data and learn from them, in order to make their own predictions. In short, machine learning algorithms and models learn through experience.

In traditional programming, a computer engineer writes a series of directions that instruct a computer how to transform input data into a desired output. Instructions are mostly based on an IF-THEN structure: when certain conditions are met, the program executes a specific action.

Machine learning, on the other hand, is an automated process that enables machines to solve problems with little or no human input, and take actions based on past observations.

While artificial intelligence and machine learning are often used interchangeably, they are two different concepts. AI is the broader concept – machines making decisions, learning new skills, and solving problems in a similar way to humans – whereas machine learning is a subset of AI that enables intelligent systems to autonomously learn new things from data.

Instead of programming machine learning algorithms to perform tasks, you can feed them examples of labeled data (known as training data), which helps them make calculations, process data, and identify patterns automatically.

Put simply, Google's Chief Decision Scientist describes machine learning as a fancy labeling machine. After teaching machines to label things like apples and pears, by showing them examples of fruit, eventually they will start labeling apples and pears without any help – provided they have learned from appropriate and accurate training examples.

Machine learning can be put to work on massive amounts of data and can perform much more accurately than humans. It can help you save time and money on tasks and analyses, like solving customer pain points to improve customer satisfaction, support ticket automation, and data mining from internal sources and all over the internet.

5.1 Classification of Machine Learning-

Machine learning[22] implementations are classified into three major categories, depending on the nature of the learning "signal" or "response" available to a learning system which are as follows:-

5.1.1 Supervised learning:

Supervised learning models make predictions based on labeled training data. Each training sample includes an input and a desired output. A supervised learning algorithm analyzes this sample data and makes an inference – basically, an educated guess when determining the labels for unseen data.

This is the most common and popular approach to machine learning. It's "supervised" because these models need to be fed manually tagged sample data to learn from. Data is labeled to tell the machine what patterns (similar words and images, data categories, etc.) it should be looking for and recognize connections with.

For example, if you want to automatically detect spam, you would need to feed a machine learning algorithm examples of emails that you want classified as spam and others that are important, and should not be considered spam.

Which brings us to our next point – the two types of supervised learning tasks: classification and regression.

Classification in supervised machine learning

There are a number of classification algorithms used in supervised learning, with Support Vector Machines (SVM) and Naive Bayes among the most common.

In classification tasks, the output value is a category with a finite number of options. For example, with this free pre-trained sentiment analysis model, you can automatically classify data as positive, negative, or neutral.

Let's say you want to analyze customer support conversations to understand your clients' emotions: are they happy or frustrated after contacting your customer service team? A sentiment analysis classifier can automatically tag responses for you, like in the below:

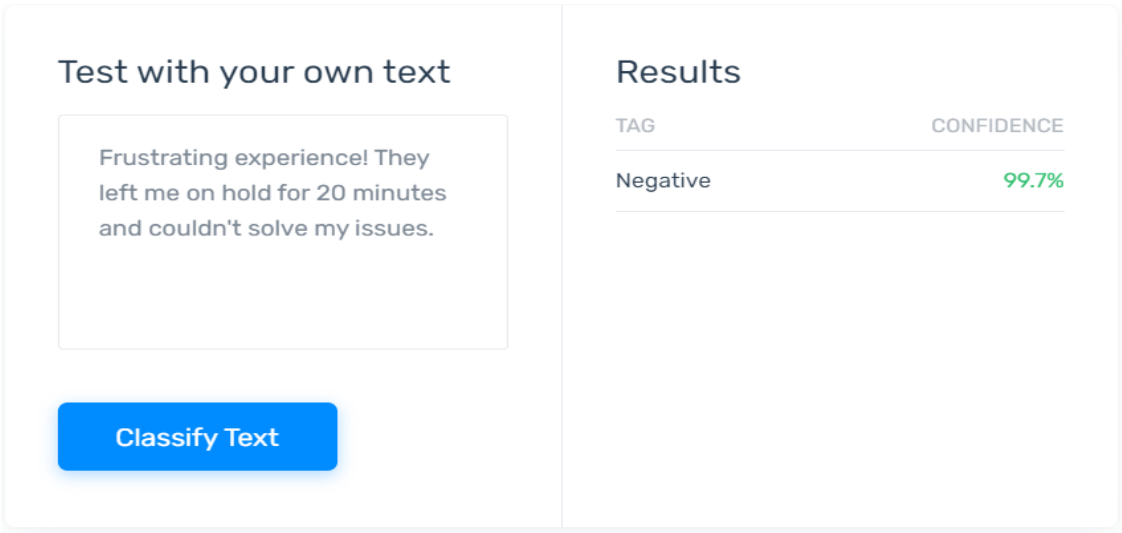


Fig 5.2 : Classifying Text

In this example, a sentiment analysis model tags a frustrating customer support experience as “Negative”.

Regression in supervised machine learning

In regression tasks, the expected result is a continuous number. This model is used to predict quantities, such as the probability an event will happen, meaning the output may have any number value within a certain range. Predicting the value of a property in a specific neighborhood or the spread of COVID19 in a particular region are examples of regression problems.

5.1.2 Unsupervised learning

Uncover insights and relationships in unlabeled data. In this case, models are fed input data but the desired outcomes are unknown, so they have to make inferences based on circumstantial evidence, without any guidance or training. The models are not trained with the “right answer,” so they must find patterns on their own.

One of the most common types of unsupervised learning is clustering, which consists of grouping similar data. This method is mostly used for exploratory analysis and can help you detect hidden patterns or trends.

For example, the marketing team of an e-commerce company could use clustering to improve customer segmentation. Given a set of income and spending data, a machine learning model can identify groups of customers with similar behaviors.

Segmentation allows marketers to tailor strategies for each key market. They might offer promotions and discounts for low-income customers that are high spenders on the site, as a way to reward loyalty and improve retention.

5.1.3 Semi-Supervised Learning

In semi-supervised learning, training data is split into two. A small amount of labeled data and a larger set of unlabeled data.

In this case, the model uses labeled data as an input to make inferences about the unlabeled data, providing more accurate results than regular supervised-learning models.

This approach is gaining popularity, especially for tasks involving large datasets such as image classification. Semi-supervised learning doesn't require a large number of labeled data, so it's faster to set up, more cost-effective than supervised learning methods, and ideal for businesses that receive huge amounts of data.

5.1.4 Reinforcement Learning

When you present the algorithm with examples that lack labels, as in unsupervised learning. However, you can accompany an example with positive or negative feedback according to the solution the algorithm proposes comes under the category of Reinforcement learning, which is connected to applications for which the algorithm must make decisions (so the product is prescriptive, not just descriptive, as in unsupervised learning), and the decisions bear consequences. In the human world, it is just like learning by trial and error. Errors help you learn because they have a penalty added (cost, loss of time, regret, pain, and so on), teaching you that a certain course of action is less likely to succeed than others. An interesting example of reinforcement learning occurs when computers learn to play video games by themselves. In this case, an application presents the algorithm with examples of specific situations, such as having the gamer stuck in a maze while avoiding an enemy. The application lets the algorithm know the outcome of actions it takes, and learning occurs while trying to avoid what it discovers to be dangerous and to pursue survival. You can have a look at how the company Google Deep Mind has created a reinforcement learning program that plays old Atari's videogames. When watching the video, notice how the program is initially clumsy and unskilled but steadily improves with training until it becomes a champion.

5.1.5 Deep Learning (DL)

models can be supervised, semi-supervised, or unsupervised (or a combination of any or all of the three). They're advanced machine learning algorithms used by tech giants, like Google, Microsoft, and Amazon to run entire systems and power things, like self driving cars and smart assistants.

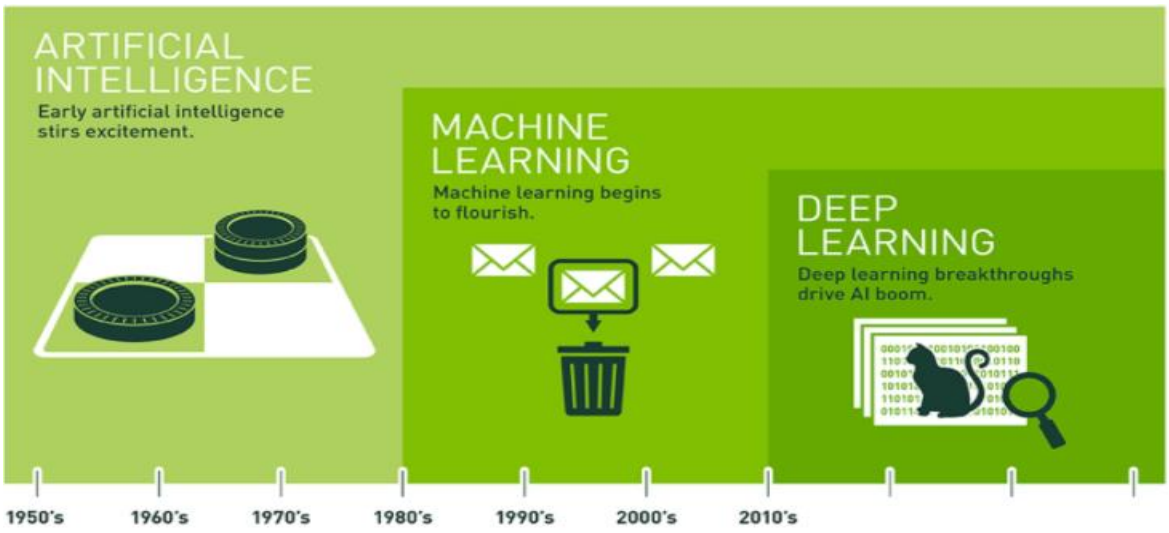


Fig 5.3 : Deep Learning

Deep learning is based on Artificial Neural Networks (ANN), a type of computer system that emulates the way the human brain works. Deep learning algorithms or neural networks are built with multiple layers of interconnected neurons, allowing multiple systems to work together simultaneously, and step-by-step.

When a model receives input data – which could be image, text, video, or audio – and is asked to perform a task (for example, text classification with machine learning), the data passes through every layer, enabling the model to learn progressively. It's kind of like a human brain that evolves with age and experience!

Deep learning is common in image recognition, speech recognition, and Natural Language Processing (NLP). Deep learning models usually perform better than other machine learning algorithms for complex problems and massive sets of data. However, they generally require millions upon millions of pieces of training data, so it takes quite a lot of time to train them

5.2 How Machine Learning Works

In order to understand how machine learning works, first you need to know what a "tag" is. To train image recognition, for example, you would "tag" photos of dogs, cats, horses, etc., with the appropriate animal name. This is also called data labeling.

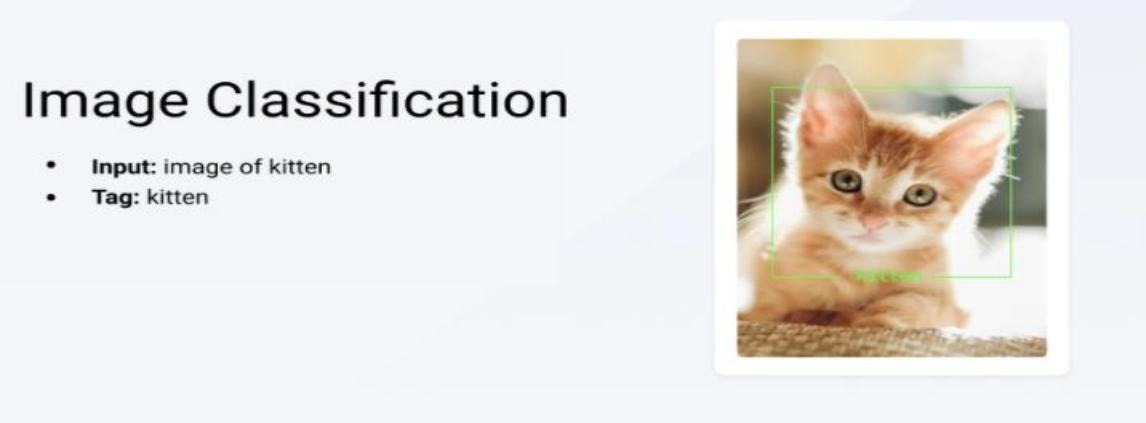


Fig 5.4 : Image Classification

When working with machine learning text analysis, you would feed a text analysis model with text training data, then tag it, depending on what kind of analysis you're doing. If you're working with sentiment analysis, you would feed the model with customer feedback, for example, and train the model by tagging each comment as Positive, Neutral, and Negative.

Take a look at the diagram below:

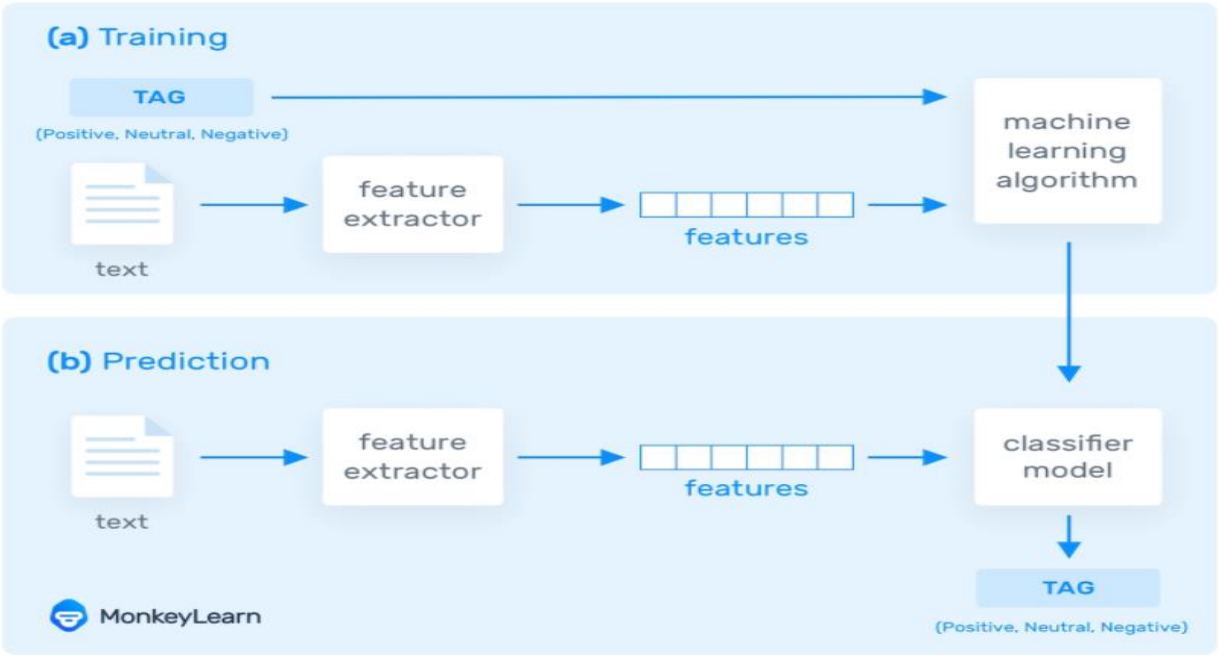


Fig 5.5 : Extracting Tweets

5.3 At its most simplistic, the machine learning process involves three steps:

1. Feed a machine learning model training input data. In our case, this could be customer comments from social media or customer service data.
2. Tag training data with a desired output. In this case, tell your sentiment analysis model whether each comment or piece of data is Positive, Neutral, or Negative. The model transforms the training data into text vectors – numbers that represent data features.

3. Test your model by feeding it testing (or unseen) data. Algorithms are trained to associate feature vectors with tags based on manually tagged samples, then learn to make predictions when processing unseen data.

If your new model performs to your standards and criteria after testing it, it's ready to be put to work on all kinds of new data. If it's not performing accurately, you'll need to keep training. Furthermore, as human language and industry-specific language morphs and changes, you may need to continually train your model with new information.

5.4 Machine Learning Use Cases

Machine Learning Application and use cases are nearly endless, especially as we begin to work from home more (or have hybrid offices), become more tied to our smartphones, and use machine learning-guided technology to get around.

Machine learning in finance, healthcare, hospitality, government, and beyond, is already in regular use. Businesses are beginning to see the benefits of using machine learning tools to improve their processes, gain valuable insights from unstructured data, and automate tasks that would otherwise require hours of tedious, manual work (which usually produces much less accurate results).

For example, UberEats uses machine learning to estimate optimum times for drivers to pick up food orders, while Spotify leverages machine learning to offer personalized content and personalized marketing. And Dell uses machine learning text analysis to save hundreds of hours analyzing thousands of employee surveys to listen to the voice of employee (VoE) and improve employee satisfaction.

How do you think Google Maps predicts peaks in traffic and Netflix creates personalized movie recommendations, even informs the creation of new content ? By using machine learning, of course.

There are many different applications of machine learning, which can benefit your business in countless ways. You'll just need to define a strategy to help you decide the best way to implement machine learning into your existing processes. In the meantime, here are some common machine learning use cases and applications that might spark some ideas:

- Social Media Monitoring
- Customer Service & Customer Satisfaction
- Image Recognition
- Virtual Assistants
- Product Recommendations
- Stock Market Trading
- Medical Diagnosis

5.4.1 Social Media Monitoring

Using machine learning you can monitor mentions of your brand on social media and immediately identify if customers require urgent attention. By detecting mentions from angry customers, in real-time, you can automatically tag customer feedback and respond right away.

You might also want to analyze customer support interactions on social media and gauge customer satisfaction (CSAT), to see how well your team is performing.

Natural Language Processing gives machines the ability to break down spoken or written language much like a human would, to process “natural” language, so machine learning can handle text from practically any source.

5.4.2 Customer Service & Customer Satisfaction

Machine learning allows you to integrate powerful text analysis tools with customer support tools, so you can analyze your emails, live chats, and all manner of internal data on the go. You can use machine learning to tag support tickets and route them to the correct teams or auto-respond to common queries so you never leave a customer in the cold.

Furthermore, using machine learning to set up a voice of customer (VoC) program and a customer feedback loop will ensure that you follow the customer journey from start to finish to improve the customer experience (CX), decrease customer churn, and, ultimately, increase your profits.

5.4.3 Image Recognition

Image recognition is helping companies identify and classify images. For example, facial recognition technology is being used as a form of identification, from unlocking phones to making payments.

Self-driving cars also use image recognition to perceive space and obstacles. For example, they can learn to recognize stop signs, identify intersections, and make decisions based on what they see.

5.4.4 Virtual Assistants

Virtual assistants, like Siri, Alexa, Google Now, all make use of machine learning to automatically process and answer voice requests. They quickly scan information, remember related queries, learn from previous interactions, and send commands to other apps, so they can collect information and deliver the most effective answer.

Customer support teams are already using virtual assistants to handle phone calls, automatically route support tickets, to the correct teams, and speed up interactions with customers via computer-generated responses.

5.4.5 Product Recommendations

Association rule-learning is a machine learning technique that can be used to analyze purchasing habits at the supermarket or on e-commerce sites. It works by searching for relationships between variables and finding common associations in transactions (products that consumers usually buy together). This data is then used for product placement strategies and similar product recommendations.

Associated rules can also be useful to plan a marketing campaign or analyze web usage.

5.4.6 Stock Market Trading

Machine learning algorithms can be trained to identify trading opportunities, by recognizing patterns and behaviors in historical data. Humans are often driven by emotions when it comes to making investments, so sentiment analysis with machine learning can play a huge role in identifying good and bad investing opportunities, with no human bias, whatsoever. They can even save time and allow traders more time away from their screens by automating tasks.

5.4.7 Medical Diagnosis

The ability of machines to find patterns in complex data is shaping the present and future. Take machine learning initiatives during the COVID-19 outbreak, for instance. AI tools have helped predict how the virus will spread over time, and shaped how we control it. It's also helped diagnose patients by analyzing lung CTs and detecting fevers using facial recognition, and identified patients at a higher risk of developing serious respiratory disease.

Machine learning is driving innovation in many fields, and every day we're seeing new interesting use cases emerge. In business, the overall benefits of machine learning include:

- **It's cost-effective and scalable.** You only need to train a machine learning model once, and you can scale up or down depending on how much data you receive.
- **Performs more accurately than humans.** Machine learning models are trained with a certain amount of labeled data and will use it to make predictions on unseen data. Based on this data, machines define a set of rules that they apply to all datasets, helping them provide consistent and accurate results. No need to worry about human error or innate bias. And you can train the tools to the needs and criteria of your business.
- **Works in real-time, 24/7.** Machine learning models can automatically analyze data in real-time, allowing you to immediately detect negative opinions or urgent tickets and take action.

5.5 Get Started With Machine Learning Tools

When you're ready to get started with machine learning tools it comes down to the Build vs. Buy Debate. If you have a data science and computer engineering background or are prepared to hire whole teams of coders and computer scientists, building your own with open-source libraries can produce great results. Building your own tools, however, can take months or years and cost in the tens of thousands.

Using SaaS or MLaaS (Machine Learning as a Service) tools, on the other hand, is much cheaper because you only pay what you use. They can also be implemented right away and new platforms and techniques make SaaS tools just as powerful, scalable, customizable, and accurate as building your own.

Whether you choose to build or buy you machine learning tools, here are some of the best of each:

5.5.1 Top SaaS Machine Learning Tools

Some of the best SaaS machine learning tools on the market:

- MonkeyLearn
- BigML
- IBM Watson
- Google Cloud ML

MonkeyLearn

MonkeyLearn is a powerful SaaS machine learning platform with a suite of text analysis tools to get real-time insights and powerful results, so you can make data-driven decisions from all manner of text data: customer service interactions, social media comments, online reviews, emails, live chats, and more.

Just connect your data and use one of the pre-trained machine learning models to start analyzing it. You can even build your own no-code machine learning models in a few simple steps, and integrate them with the apps you use every day, like Zendesk, Google Sheets and more.

Fig 5.6 : Working of monkey learn

MonkeyLearn is scalable to handle any amount of data – from just a few hundred surveys, to hundreds of thousands of comments from all over the web – to get real-world results from your data with techniques, like topic analysis, sentiment analysis, text extraction, and more.

And you can take your analysis even further with MonkeyLearn Studio to combine your analyses to work together. It's a seamless process to take you from data collection to analysis to striking visualization in a single, easy-to-use dashboard.

Take a look at this MonkeyLearn Studio aspect-based sentiment analysis of online reviews of Zoom:

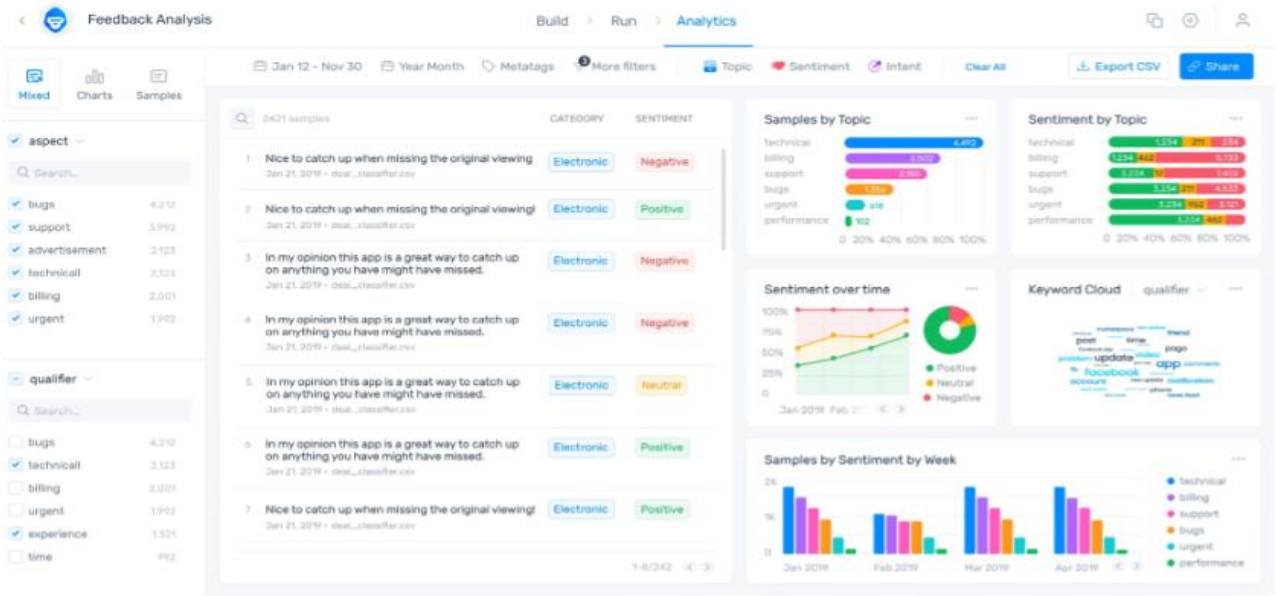


Fig 5.7 : Feedback Analysis

Aspect-based sentiment analysis first categorizes the customer opinions by “aspect” (topic or subject): Usability, Reliability, Pricing, etc. Each comment is then sentiment analyzed to show whether it’s Positive, Negative, or Neutral. This allows you to see which aspects of your business are particularly positive and which are negative.

Other techniques, like intent classification are especially useful for incoming emails or social media inquiries to automatically show why the customer is writing. Also, at the bottom right you can see word clouds that show the most used and most important words and phrases by sentiment.

BigML

The goal of BigML is to connect all of your company’s data streams and internal processes to simplify collaboration and analysis results across the organization. They specialize in industries, like aerospace, automotive, energy, entertainment, financial services, food, healthcare, IoT, pharmaceutical, transportation, telecommunications, and more, so many of their tools are ready to go, right out of the box.

You can use pre-trained models or train your own with classification and regression and time series forecasting.

IBM Watson

IBM Watson is a machine learning juggernaut, offering adaptability to most industries and the ability to build to huge scale across any cloud.

Watson Speech-to-Text is one of the industry standards for converting real-time spoken language to text, and Watson Language Translator is one of the best text translation tools on the market.

Watson Studio is great for data preparation and analysis and can be customized to almost any field, and their Natural Language Classifier makes building advanced SaaS analysis models easy.

See products page for pricing.

Google Cloud ML

Google Cloud ML is a SaaS analysis solution for image and text that connects easily to all of Google's tools: Gmail, Google Sheets, Google Slides, Google Docs, and more.

Google AutoML Natural Language is one of the most advanced text analysis tools on the market, and AutoML Vision allows you to automate the training of custom image analysis models for some of the best accuracy, regardless of your needs.

Google Cloud AI and ML pricing

5.5.2 Top Open Source Libraries for Machine Learning

Open source machine learning libraries offer collections of pre-made models and components that developers can use to build their own applications, instead of having to code from scratch. They are free, flexible, and can be customized to meet specific needs.

Some of the most popular open-source libraries for machine learning include:

- Scikit-learn
- PyTorch
- Kaggle
- NLTK
- TensorFlow

Scikit-learn

Scikit-learn is a popular Python library and a great option for those who are just starting out with machine learning. Why? It's easy to use, robust, and very well documented. You can use this library for tasks such as classification, clustering, and regression, among others.

PyTorch

Developed by Facebook, PyTorch is an open source machine learning library based on the Torch library with a focus on deep learning. It's used for computer vision and natural language processing, and is much better at debugging than some of its competitors. If you want to start out with PyTorch, there are easy-to-follow tutorials for both beginners and advanced coders. Known for its flexibility and speed, durgesh is ideal if you need a quick solution.

Kaggle

Launched over a decade ago (and acquired by Google in 2017), Kaggle has a learning-by-doing philosophy, and it's renowned for its competitions in which participants create models to solve real problems. Check out this online machine learning course in Python, which will have you building your first model in next to no time.

NLTK

The Natural Language Toolkit (NLTK) is possibly the best known Python library for working with natural language processing. It can be used for keyword search, tokenization and classification, voice recognition and more. With a heavy focus on research and education, you'll find plenty of resources, including data sets, pre-trained models, and a textbook to help you get started.

TensorFlow

An open-source Python library developed by Google for internal use and then released under an open license, with tons of resources, tutorials, and tools to help you hone your machine learning skills. Suitable for both beginners and experts, this user-friendly platform has all you need to build and train machine learning models (including a library of pre-trained models). Tensorflow is more powerful than other libraries and focuses on deep learning, making it perfect for complex projects with large-scale data. However, it may take time and skills to master. Like with most open-source tools, it has a strong community and some tutorials to help you get started.

5.6 Final Note

Monkeylearn is an easy-to-use SaaS platform that allows you to create machine learning models to perform text analysis tasks like topic classification, sentiment analysis, keyword extraction, and more.

MonkeyLearn offers simple integrations with tools you already use, like Zendesk, Freshdesk, SurveyMonkey, Google Apps, Zapier, Rapidminer, and more, to streamline processes, save time, and increase internal (and external) communication.

Take a look at the MonkeyLearn Studio public dashboard to see how easy it is to use all of your text analysis tools from a single, striking dashboard. Play around with abhishek, category, and more.

Ready to take your first steps with MonkeyLearn's low-code, no code solution?

Request a demo and start creating value from your data.

5.7 Categorizing on the basis of required Output-

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

1. **Classification:** When inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multilabel classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

2. **Regression:** Which is also a supervised problem, A case when the outputs are continuous rather than discrete.

3. **Clustering :** When a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

CHAPTER 6 : MACHINE LEARNING ALGORITHMS

6.1 Introduction to Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.
In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).
Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

6.1.1 Logistic Function

Logistic regression is named for the function used at the core of the method, the logistic function.
The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

6.1.2 Logistic Regression Assumptions-

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same −

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-co linearity in the model, which means the independent variables must be independent of each other. We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

6.1.3 Representation Used for Logistic Regression

Logistic regression uses an equation as the representation, very much like linear regression. Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value. Below is an example logistic regression equation:

$$y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.
The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

6.1.4 Logistic Regression Predicts Probabilities (Technical Interlude)

Logistic regression models the probability of the default class (e.g. the first class).
For example, if we are modeling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

$$P(sex=male|height)$$

Written another way, we are modeling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

$$P(X) = P(Y=1|X)$$

We're predicting probabilities? I thought logistic regression was a classification algorithm?
Note that the probability prediction must be transformed into a binary values (0 or 1) in order to actually make a probability prediction. More on this later when we talk about making predictions.
Logistic regression is a linear method, but the predictions are transformed using the logistic function. The impact of this is that we can no longer understand the predictions as a linear combination of the inputs as we can with linear regression, for example, continuing on from above, the model can be stated as:

$$p(X) = e^{(b0 + b1*X)} / (1 + e^{(b0 + b1*X)})$$

I don't want to dive into the math too much, but we can turn around the above equation as follows (remember we can remove the e from one side by adding a natural logarithm (ln) to the other):

$$\ln(p(X) / 1 - p(X)) = b0 + b1 * X$$

This is useful because we can see that the calculation of the output on the right is linear again (just like linear regression), and the input on the left is a log of the probability of the default class.
This ratio on the left is called the odds of the default class (it's historical that we use odds, for example, odds are used in horse racing rather than probabilities). Odds are calculated as a ratio of the probability of the event divided by the probability of not the event, e.g. 0.8/(1-0.8) which has the odds of 4. So we could instead write:

$$\ln(odds) = b0 + b1 * X$$

Because the odds are log transformed, we call this left hand side the log-odds or the probit. It is possible to use other types of functions for the transform (which is out of scope_, but as such it is common to refer to the transform that relates the linear regression equation to the probabilities as the link function, e.g. the probit link function.

We can move the exponent back to the right and write it as:

$$odds = e^{(b0 + b1 * X)}$$

All of this helps us understand that indeed the model is still a linear combination of the inputs, but that this linear combination relates to the log-odds of the default class.

6.2 Introduction to SVM

Support vector machines (SVMs)[21] are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990.

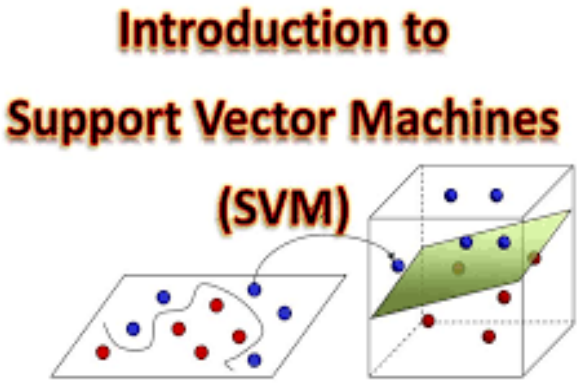


Fig 6.1 : SVM

SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

6.2.1 Working of SVM-

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).

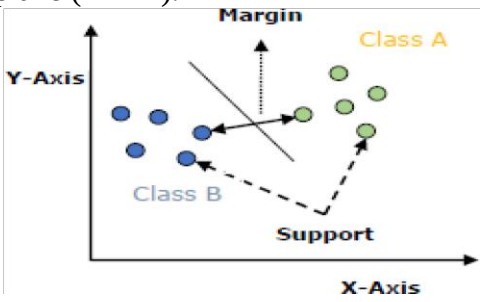


Fig 6.2 : Working of SVM

The followings are important concepts in SVM −

- **Support Vectors** – Data points that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

- **Hyperplane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

- **Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps –

1. First, SVM will generate hyperplanes iteratively that segregates the classes in best way.

2. Then, it will choose the hyperplane that separates the classes correctly.

6.2.2 Nonlinear data

Now this example was easy, since clearly the data was linearly separable — we could draw a straight line to separate red and blue. Sadly, usually things aren't that simple. Take a look at this case:

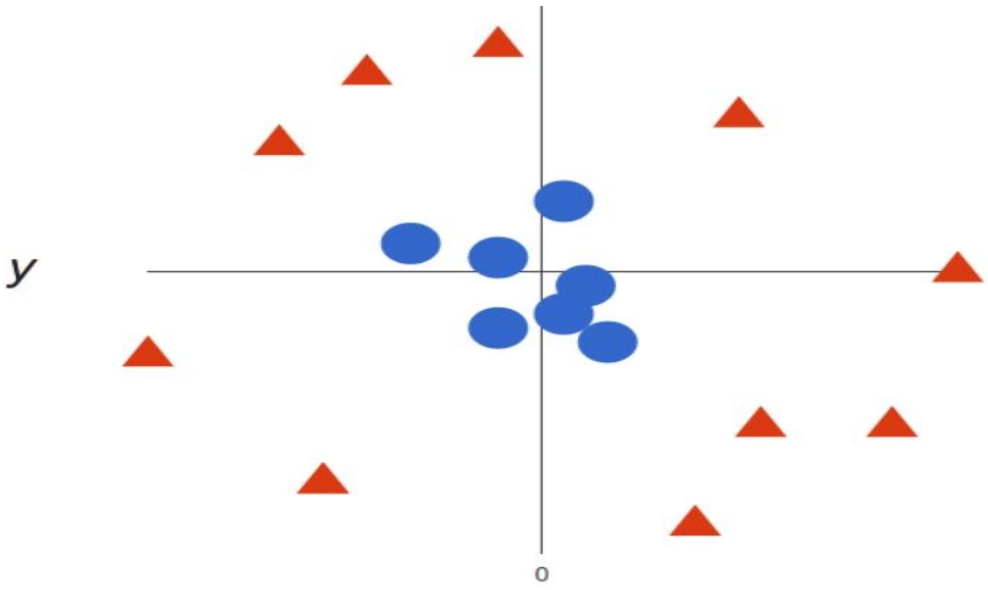


Fig 6.3 : A more complex dataset

It's pretty clear that there's not a linear decision boundary (a single straight line that separates both tags). However, the vectors are very clearly segregated and it looks as though it should be easy to separate them.

So here's what we'll do: we will add a third dimension. Up until now we had two dimensions: x and y. We create a new z dimension, and we rule that it be calculated a certain way that is convenient for us: $z = x^2 + y^2$ (you'll notice that's the equation for a circle).

This will give us a three-dimensional space. Taking a slice of that space, it looks like this:

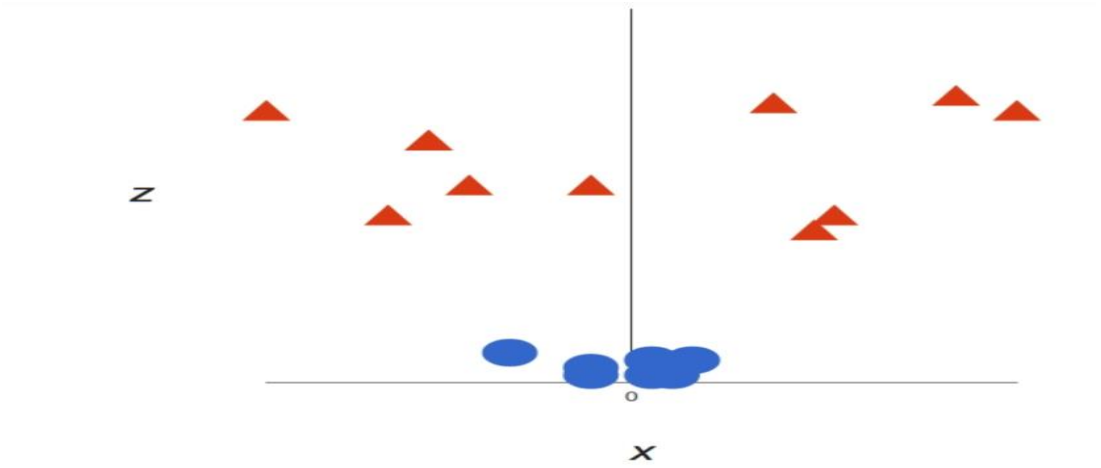


Fig 6.4 : Linearnly Separated Groups

From a different perspective, the data is now in two linearly separated groups

What can SVM do with this? Let's see:

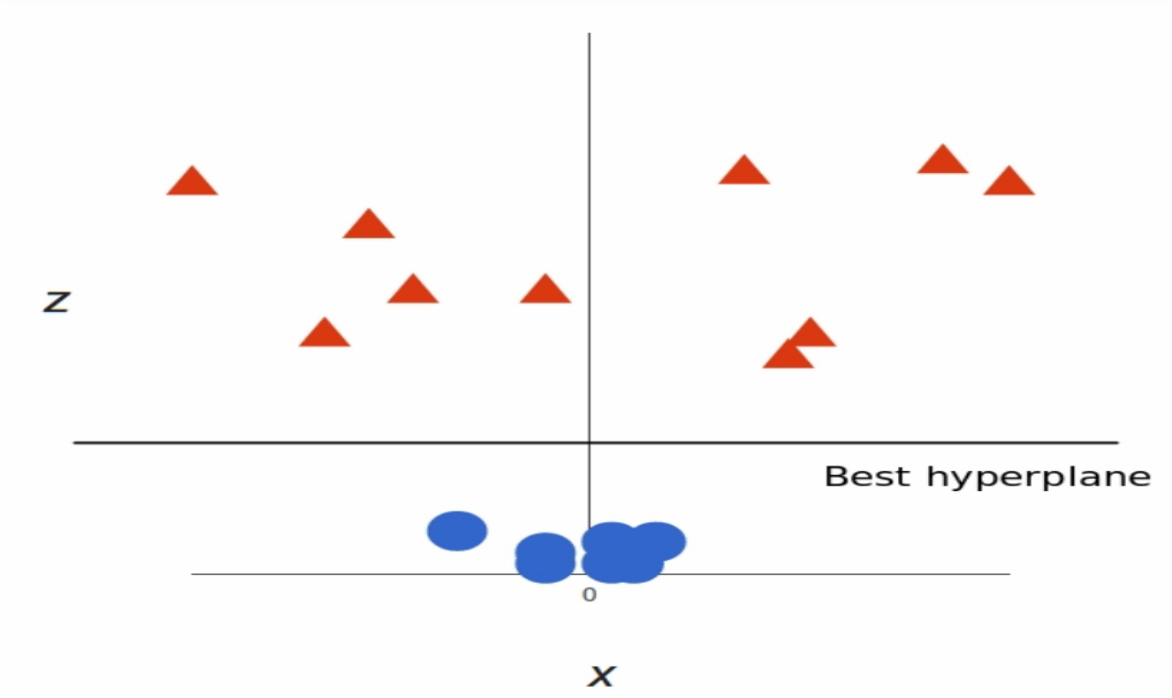


Fig 6.5 : SVM Hyperplane

That's great! Note that since we are in three dimensions now, the hyperplane is a plane parallel to the x axis at a certain z (let's say z = 1).

What's left is mapping it back to two dimensions:

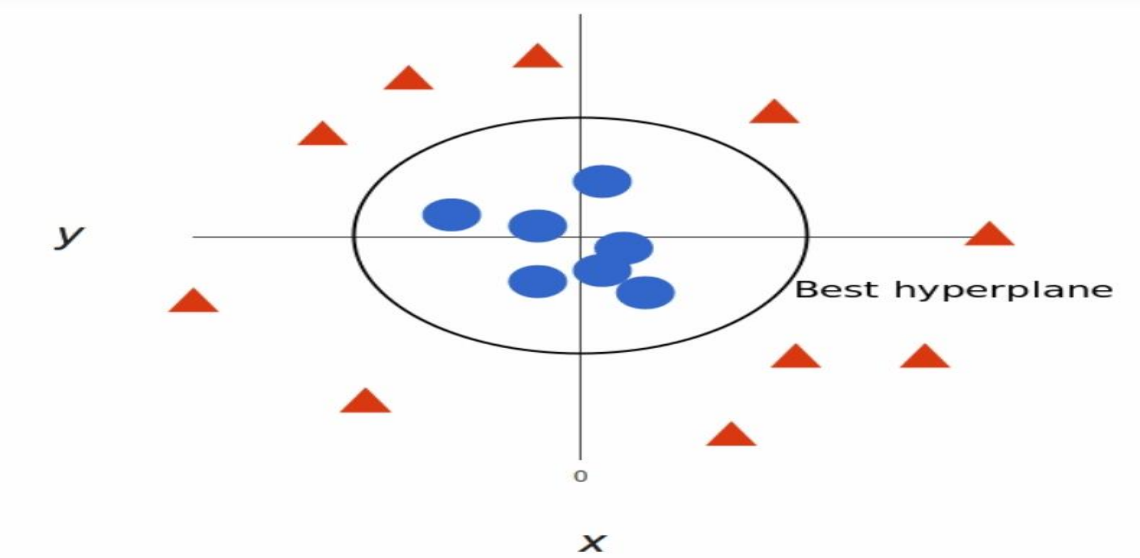


Fig 6.6 : SVM Left Mapping

Back to our original view, everything is now neatly separated

6.2.3 The kernel trick

In our example we found a way to classify nonlinear data by cleverly mapping our space to a higher dimension. However, it turns out that calculating this transformation can get pretty computationally expensive: there can be a lot of new dimensions, each one of them possibly involving a complicated calculation. Doing this for every vector in the dataset can be a lot of work, so it'd be great if we could find a cheaper solution.

And we're in luck! Here's a trick: SVM doesn't need the actual vectors to work its magic, it actually can get by only with the dot products between them. This means that we can sidestep the expensive calculations of the new dimensions.

This is what we do instead:

- Imagine the new space we want:

$z = x^2 + y^2$

- Figure out what the dot product in that space looks like:

$a \cdot b = xa \cdot xb + ya \cdot yb + za \cdot zb$

$a \cdot b = xa \cdot xb + ya \cdot yb + (xa^2 + ya^2) \cdot (xb^2 + yb^2)$

- Tell SVM to do its thing, but using the new dot product — we call this a **kernel function**.

That's it! That's the **kernel trick**, which allows us to sidestep a lot of expensive calculations. Normally, the kernel is linear, and we get a linear classifier. However, by using a nonlinear

kernel (like above) we can get a nonlinear classifier without transforming the data at all: we only change the dot product to that of the space that we want and SVM will happily chug along.

Note that the kernel trick isn't actually part of SVM. It can be used with other linear classifiers such as logistic regression. A support vector machine only takes care of finding the decision boundary.

6.3 Using SVM with Natural Language Classification

So, we can classify vectors in multidimensional space. Great! Now, we want to apply this algorithm for text classification, and the first thing we need is a way to transform a piece of text into a vector of numbers so we can run SVM with them. In other words, which **features** do we have to use in order to classify texts using SVM?

The most common answer is word frequencies, just like we did in Naive Bayes. This means that we treat a text as a bag of words, and for every word that appears in that bag we have a feature. The value of that feature will be how frequent that word is in the text.

This method boils down to just counting how many times every word appears in a text and dividing it by the total number of words. So in the sentence "All monkeys are primates but not all primates are monkeys" the word monkeys has a frequency of 2/10 = 0.2, and the word but has a frequency of 1/10 = 0.1 .

For a more advanced alternative for calculating frequencies, we can also use TF-IDF.

Now that we've done that, every text in our dataset is represented as a vector with thousands (or tens of thousands) of dimensions, every one representing the frequency of one of the words of the text. Perfect! This is what we feed to SVM for training. We can improve this by using preprocessing techniques, like stemming, removing stopwords, and using n-grams.

6.3.1 Choosing a kernel function

Now that we have the feature vectors, the only thing left to do is choosing a kernel function for our model. Every problem is different, and the kernel function depends on what the data looks like. In our example, our data was arranged in concentric circles, so we chose a kernel that matched those data points.

Taking that into account, what's best for natural language processing? Do we need a nonlinear classifier? Or is the data linearly separable? It turns out that it's best to stick to a linear kernel. Why?

Back in our example, we had two features. Some real uses of SVM in other fields may use tens or even hundreds of features. Meanwhile, NLP classifiers use thousands of features, since they can have up to one for every word that appears in the training data. This changes the problem a little bit: while using nonlinear kernels may be a good idea in other cases, having this many features will end up making nonlinear kernels overfit the data. Therefore, it's best to just stick to a good old linear kernel, which actually results in the best performance in these cases.

6.3.2 Putting it all together

Now the only thing left to do is training! We have to take our set of labeled texts, convert them to vectors using word frequencies, and feed them to the algorithm — which will use our chosen kernel function — so it produces a model. Then, when we have a new unlabeled text that we want to classify, we convert it into a vector and give it to the model, which will output the tag of the text.

6.4 Simple SVM Classifier Tutorial

To create your own SVM classifier, without dabbling in vectors, kernels, and TF-IDF, you can use one of MonkeyLearn's pre-built classification models to get started right away. It's also easy to create your own, thanks to the platform's super intuitive user interface and no-code approach.

It's also great for those who don't want to invest large amounts of capital in hiring machine learning experts.

Let's show you how easy it is to create your SVM classifier in 8 simple steps. Before you get started, you'll need to sign up to MonkeyLearn for free.

1. Create a new classifier

Go to the dashboard, click on "Create a Model" and choose "Classifier".

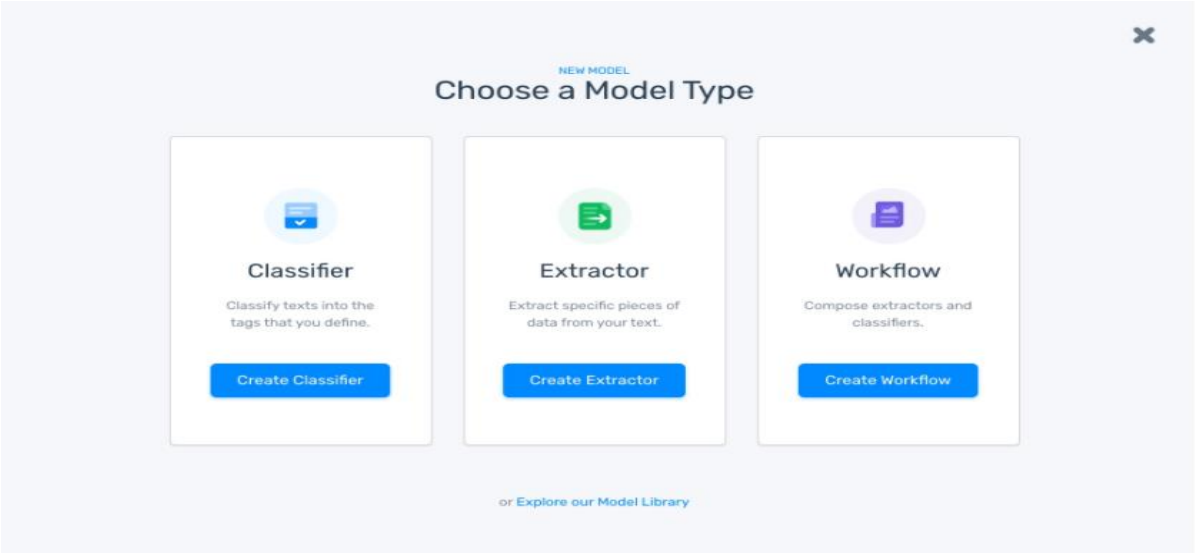


Fig 6.7 : Creating Classifier

2. Select how you want to classify your data

We're going to opt for a "Topic Classification" model to classify text based on topic, aspect or relevance.

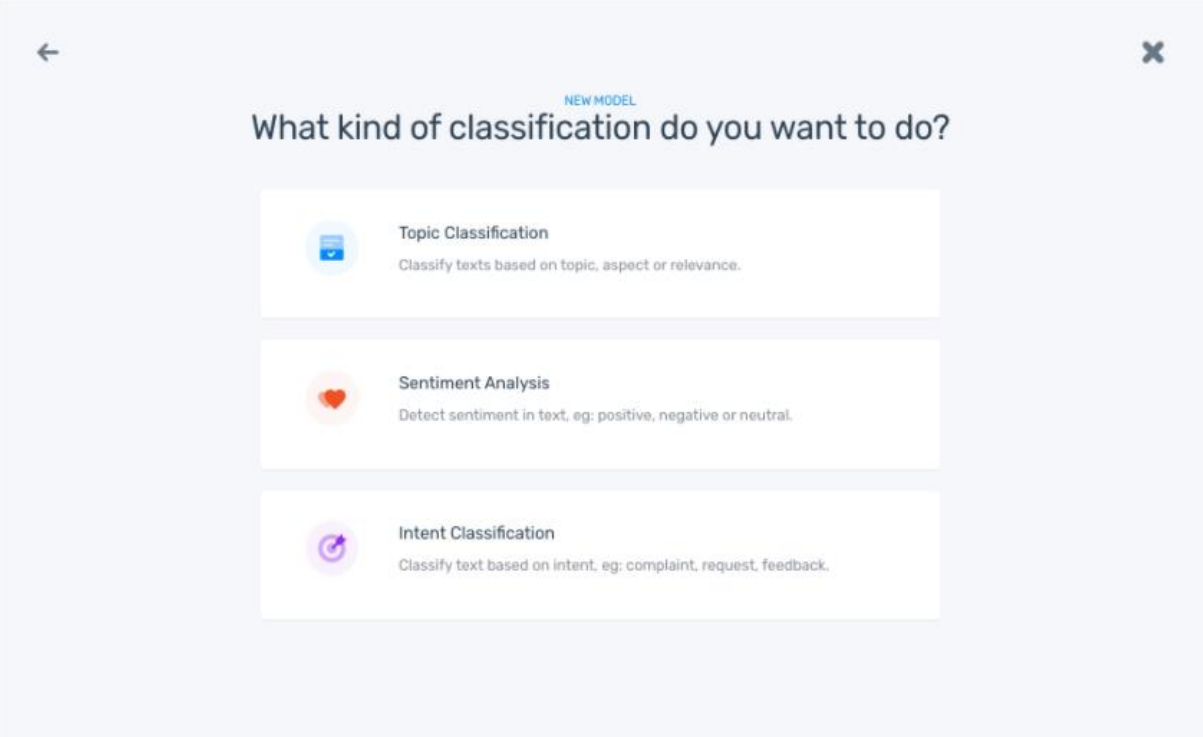


Fig 6.8 : Classifying Data

3. Import your training data

Select and upload the data that you will use to train your model. Keep in mind that classifiers learn and get smarter as you feed it more training data. You can import data from CSV or Excel files.

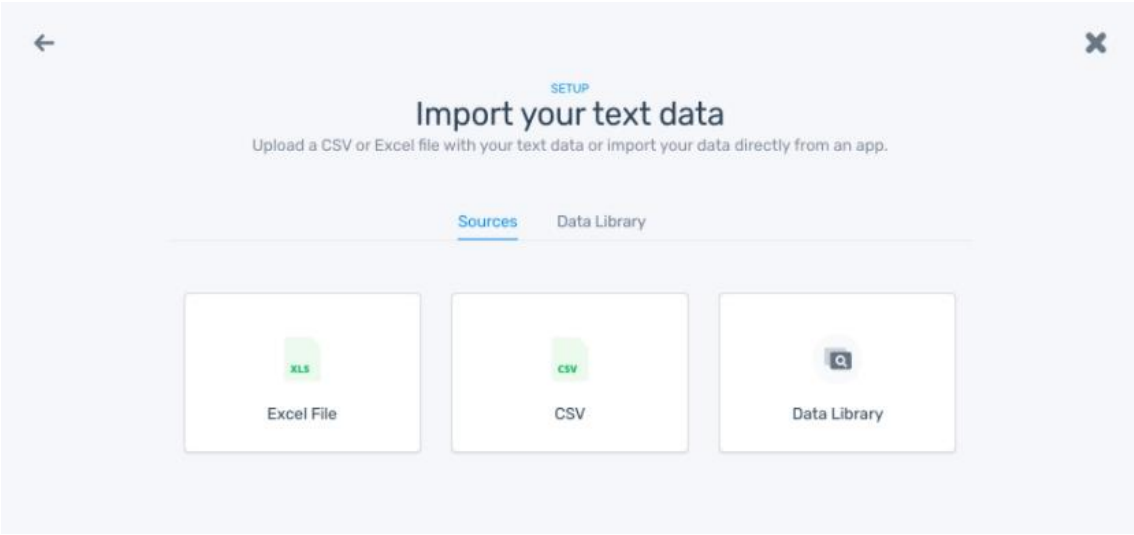


Fig 6.9 : Importing Text Data

4. Define the tags for your SVM classifier

It’s time to define your tags, which you’ll use to train your topic classifier. Add at least two tags to get started – you can always add more tags later.

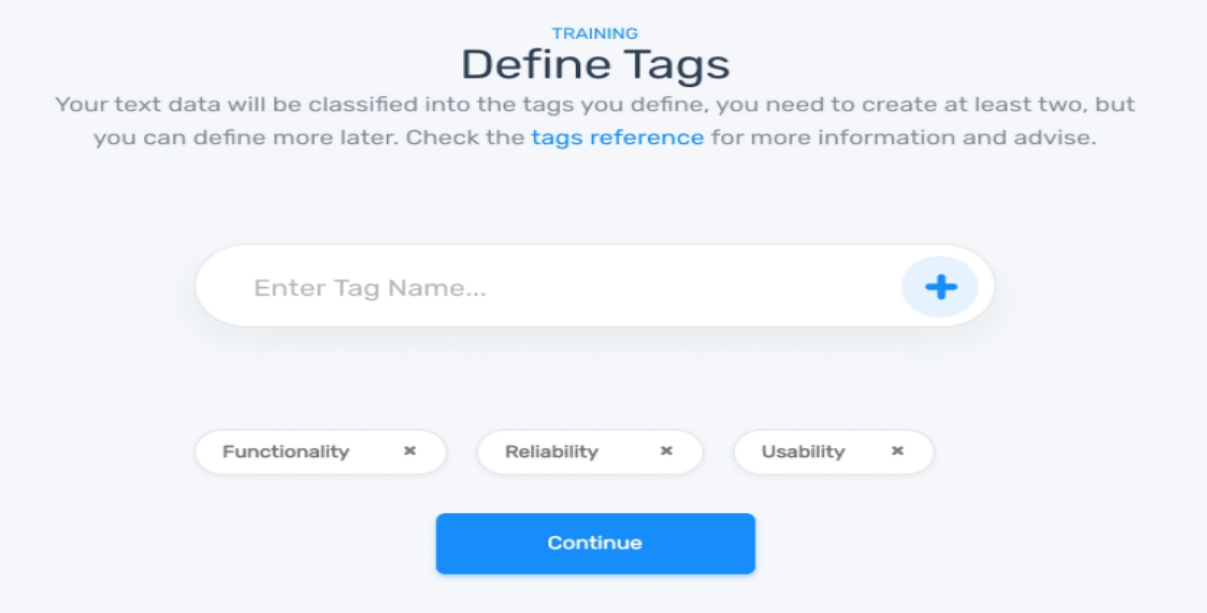


Fig 6.10 : Defining Tags

5. Tag data to train your classifier

Start training your topic classifier by choosing tags for each example:

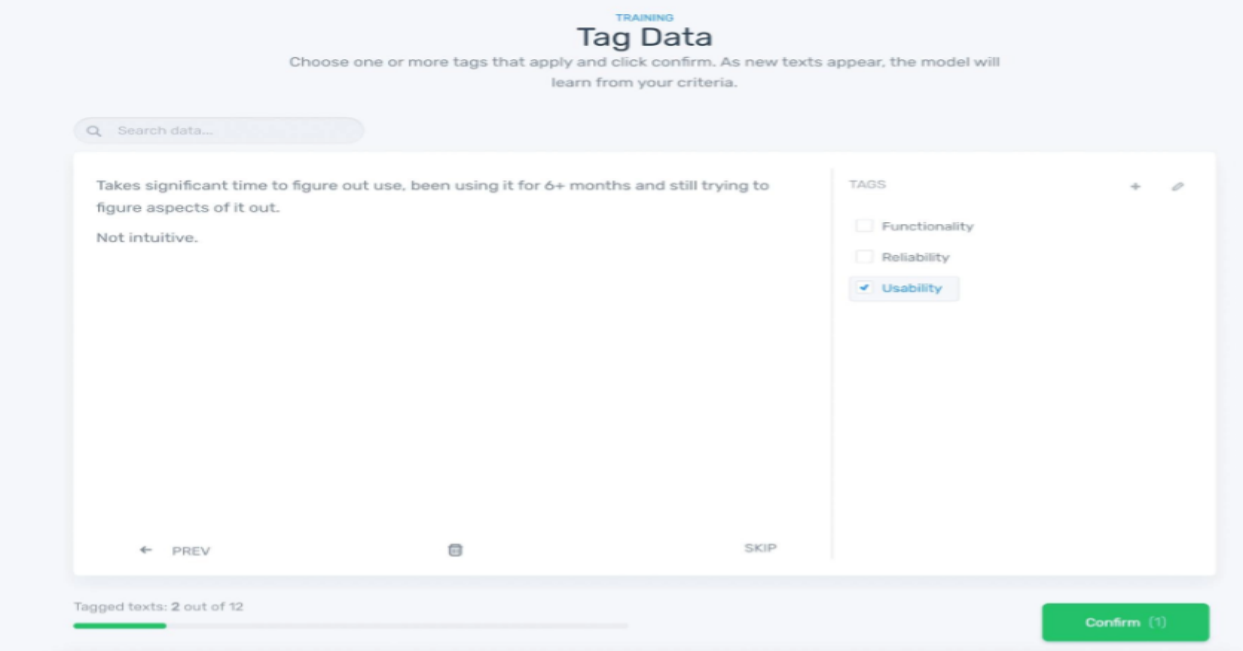


Fig 6.11 : Tag Data

After manually tagging some examples, the classifier will start making predictions on its own. If you want your model to be more accurate, you'll have to tag more examples to continue training your model.

The more data you tag, the smarter your model will be.

6. Set your algorithm to SVM

Go to settings and make sure you select the SVM algorithm in the advanced section.

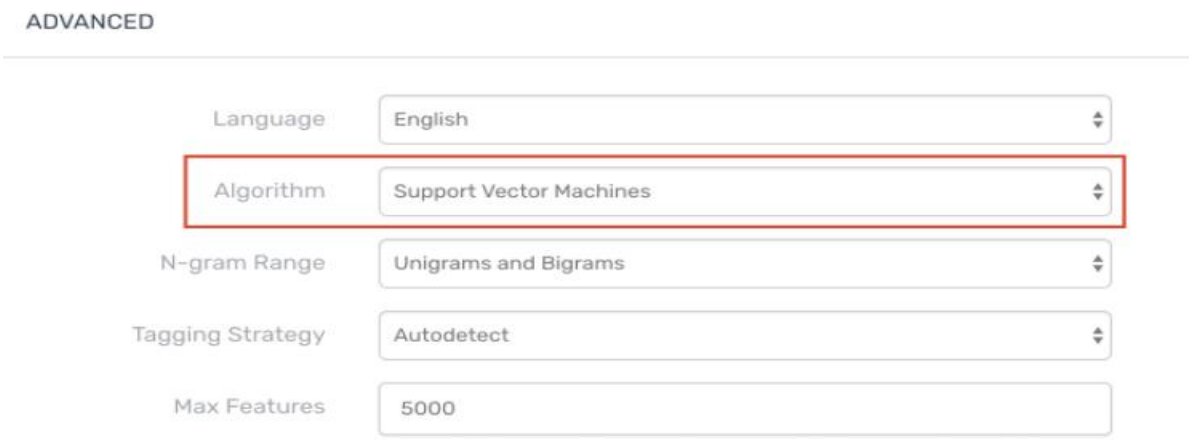


Fig 6.12 : Algorithm of SVM

7. Test Your Classifier

Now you can test your SVM classifier by clicking on “Run” > “Demo”. Write your own text and see how your model classifies the new data:

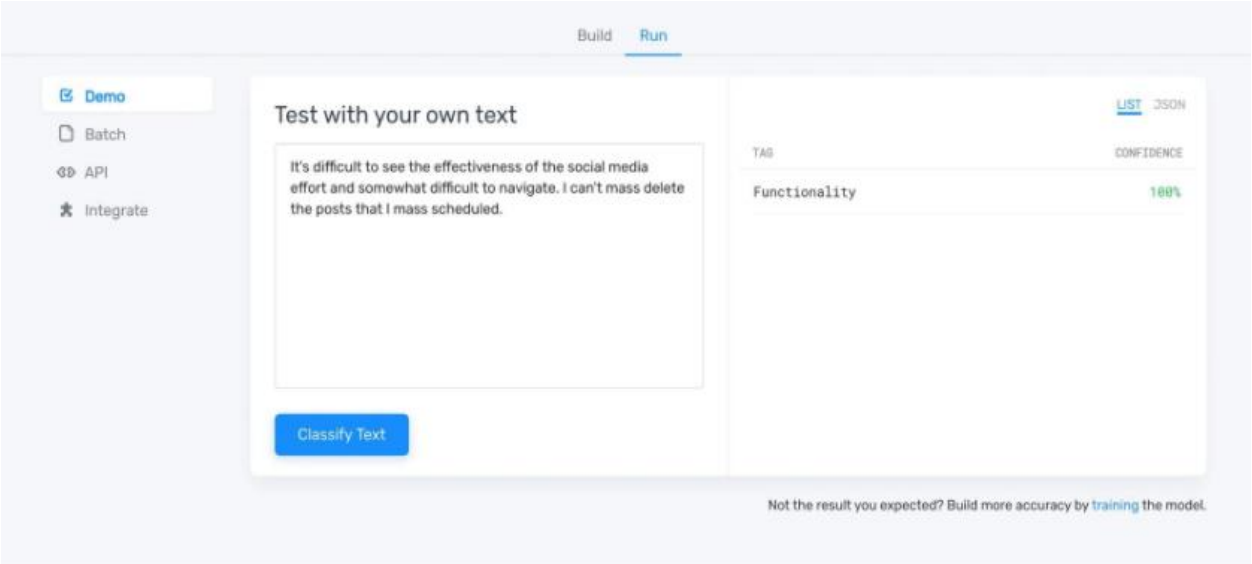


Fig 6.13 : Testing Classifier

8. Integrate the topic classifier

You’ve trained your model to make accurate predictions when classifying text. Now it’s time to upload new data! There are three different ways to do this with MonkeyLearn:

1. **Batch processing:** go to “Run” > “Batch” and upload a CSV or Excel file. The classifier will analyze your data and send you a new file with the predictions.
2. **API:** use MonkeyLearn API to classify new data from anywhere.
3. **Integrations:** connect everyday apps to automatically import new text data into your classifier. Integrations such as Google Sheets, Zapier, and Zendesk can be used without having to type a single line of code:

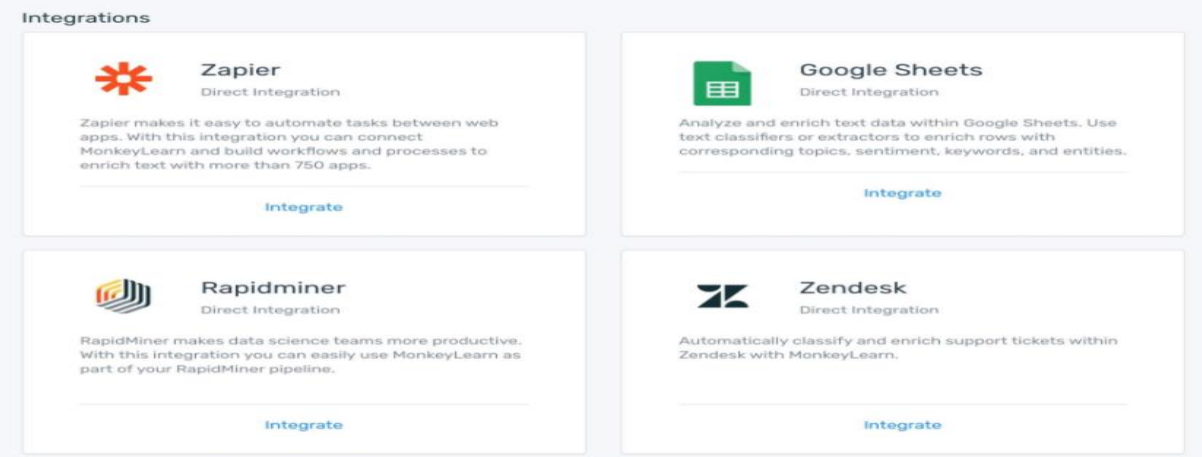


Fig 6.14 : Integrated Classifier

6.5 Final words

And that’s the basics of Support Vector Machines!

To sum up:

- A support vector machine allows you to classify data that’s linearly separable.
- If it isn’t linearly separable, you can use the kernel trick to make it work.
- However, for text classification it’s better to just stick to a linear kernel.

With MLaaS tools like MonkeyLearn, it's extremely simple to implement SVM for text classification and get insights right away.

6.6 Introduction to Decision Tree-

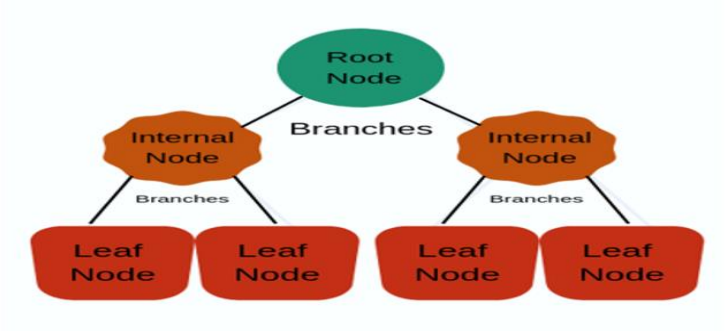


Fig 6.15: Decision Tree

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions tress are the most powerful algorithms that falls under the category of supervised algorithms.
They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below −

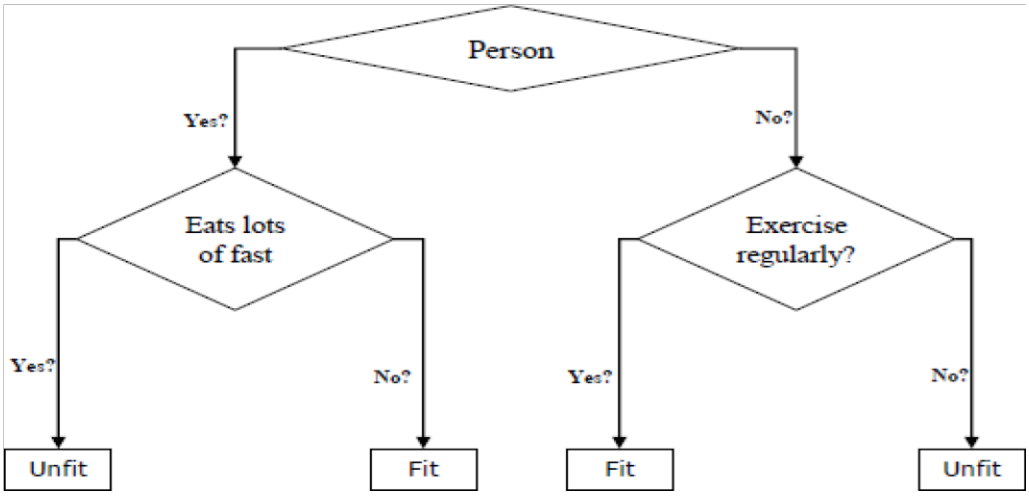


Fig 6.16 : Decision Tree Model

In the above decision tree, the question are decision nodes and final outcomes are leaves. We have the following two types of decision trees.

- **Classification decision trees** − In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.
- **Regression decision trees** − In this kind of decision trees, the decision variable is continuous.

6.7 Introduction to Naïve Bayes Algorithm-

Naïve Bayes algorithms is a classification technique based on applying Bayes' theorem with a strong assumption that all the predictors are independent to each other. In simple words, the assumption is that the presence of a feature in a class is independent to the presence of any other feature in the same class. For example, a phone may be considered as smart if it is having touch screen, internet facility, good camera etc. Though all these features are dependent on each other, they contribute independently to the probability of that the phone is a smart phone. In Bayesian classification, the main interest is to find the posterior probabilities i.e. the probability of a label given some observed features, ($L \mid features$). With the help of Bayes theorem, we can express this in quantitative form as follows −
P(L|features)=P(L)P(features|L)P(features)P(L|features)=P(L)P(features|L)P
(features) Here, ($L \mid features$) is the posterior probability of class.
(L) is the prior probability of class.

($features$ $|L$) is the likelihood which is the probability of predictor given class.
($features$) is the prior probability of predictor.

6.7.1 Building model using Naïve Bayes in Python-

Python library, Scikit learn is the most useful library that helps us to build a Naïve Bayes model in Python. We have the following three types of Naïve Bayes model under Scikit learn Python library −

Gaussian Naïve Bayes-

It is the simplest Naïve Bayes classifier having the assumption that the data from each label is drawn from a simple Gaussian distribution.

Multinomial Naïve Bayes-

Another useful Naïve Bayes classifier is Multinomial Naïve Bayes in which the features are assumed to be drawn from a simple Multinomial distribution. Such kind of Naïve Bayes are most appropriate for the features that represents discrete counts.

Bernoulli Naïve Bayes-

Another important model is Bernoulli Naïve Bayes in which features are assumed to be binary (0s and 1s). Text classification with 'bag of words' model can be an application of Bernoulli Naïve Bayes.

6.7.2 Applications of Naïve Bayes classification-

The following are some common applications of Naïve Bayes classification −

- **Real-time prediction** − Due to its ease of implementation and fast computation, it can be used to do prediction in real-time.
- **Multi-class prediction** − Naïve Bayes classification algorithm can be used to predict posterior probability of multiple classes of target variable.
- **Text classification** − Due to the feature of multi-class prediction, Naïve Bayes classification algorithms are well suited for text classification. That is why it is also used to solve problems like spam-filtering and sentiment analysis.
- **Recommendation system** − Along with the algorithms like collaborative filtering, Naïve Bayes makes a Recommendation system which can be used to filter unseen information and to predict weather a user would like the given resource or not.

6.8 Introduction to Regression-

Regression is another important and broadly used statistical and machine learning tool. The key objective of regression-based tasks is to predict output labels or responses which are continues numeric values, for the given input data. The output will be based on what the model has learned in training phase. Basically, regression models use the input data features (independent variables) and their corresponding continuous numeric output values (dependent or outcome variables) to learn specific association between inputs and corresponding outputs.

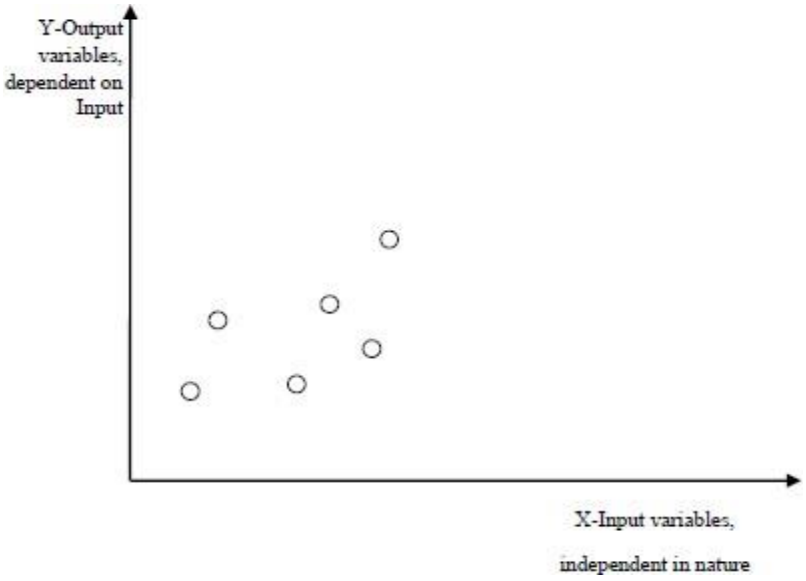


Fig 6.17 : Regression Model

6.8.1 Types of Regression Models-

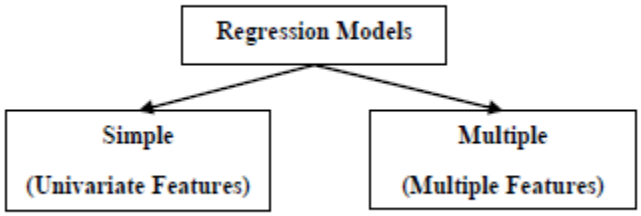


Fig 6.18 : Regression Types

Regression models are of following two types −

Simple regression model –

This is the most basic regression model in which predictions are formed from a single, univariate feature of the data.

Multiple regression model − As name implies, in this regression model the predictions are formed from multiple features of the data.

6.9 Building a Regressor in Python

Regressor model in Python can be constructed just like we constructed the classifier. Scikit-learn, a Python library for machine learning can also be used to build a regressor in Python.

In the following example, we will be building basic regression model that will fit a line to the data i.e. linear regressor. The necessary steps for building a regressor in Python are as follows

Step 1: Importing necessary python package

For building a regressor using scikit-learn, we need to import it along with other necessary packages. We can import the by using following script −

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
```

Step 2: Importing dataset

After importing necessary package, we need a dataset to build regression prediction model. We can import it from sklearn dataset or can use other one as per our requirement. We are going to use our saved input data. We can import it with the help of following script −

```
input = r'C:\linear.txt'
```

Next, we need to load this data. We are using np.loadtxt function to load it.

```
input_data = np.loadtxt(input, delimiter=',')
X, y = input_data[:, :-1], input_data[:, -1]
```

Step 3: Organizing data into training & testing sets

As we need to test our model on unseen data hence, we will divide our dataset into two parts: a training set and a test set. The following command will perform it −

```
training_samples = int(0.6 * len(X))
testing_samples = len(X) - num_training
X_train, y_train = X[:training_samples], y[:training_samples]
X_test, y_test = X[training_samples:], y[training_samples:]
```

Step 4: Model evaluation & prediction

After dividing the data into training and testing we need to build the model. We will be using LineaRegression() function of Scikit-learn for this purpose. Following command will create a linear regressor object.

```
reg_linear = linear_model.LinearRegression()
```

Next, train this model with the training samples as follows −

```
reg_linear.fit(X_train, y_train)
```

Now, at last we need to do the prediction with the testing data.

```
y_test_pred = reg_linear.predict(X_test)
```

Step 5: Plot & visualization

After prediction, we can plot and visualize it with the help of following script −

```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_test, y_test_pred, color = 'black', linewidth = 2)
plt.xticks(())
plt.yticks(())
plt.show()
```

Output

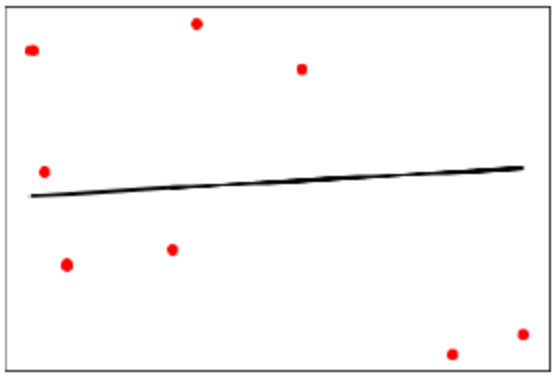


Fig 6.19 : Regression Output

In the above output, we can see the regression line between the data points.

Step 6: Performance computation

We can also compute the performance of our regression model with the help of various performance metrics as follows.

```
print("Regressor model performance:")
print("Mean absolute error(MAE) =", round(sm.mean_absolute_error(y_test, y_test_pred), 2))
print("Mean squared error(MSE) =", round(sm.mean_squared_error(y_test, y_test_pred), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test, y_test_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y_test, y_test_pred), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))
```

Output

```
Regressor model performance:
Mean absolute error(MAE) = 1.78
Mean squared error(MSE) = 3.89
Median absolute error = 2.01
Explain variance score = -0.09
R2 score = -0.09
```

6.10 Types of ML Regression Algorithms

The most useful and popular ML regression algorithm is Linear regression algorithm which further divided into two types namely −

- Simple Linear Regression algorithm
- Multiple Linear Regression algorithm.

6.11 Applications

6.11.1 The applications of ML regression algorithms are as follows −

Forecasting or Predictive analysis − One of the important uses of regression is forecasting or predictive analysis. For example, we can forecast GDP, oil prices or in simple words the quantitative data that changes with the passage of time.

Optimization − We can optimize business processes with the help of regression. For example, a store manager can create a statistical model to understand the peek time of coming of customers.

Error correction − In business, taking correct decision is equally important as optimizing the business process. Regression can help us to take correct decision as well in correcting the already implemented decision.

Economics − It is the most used tool in economics. We can use regression to predict supply, demand, consumption, inventory investment etc.

Finance − A financial company is always interested in minimizing the risk portfolio and want to know the factors that affects the customers. All these can be predicted with the help of regression model.

CHAPTER 7 : TWITTER SENTIMENT ANALYSIS

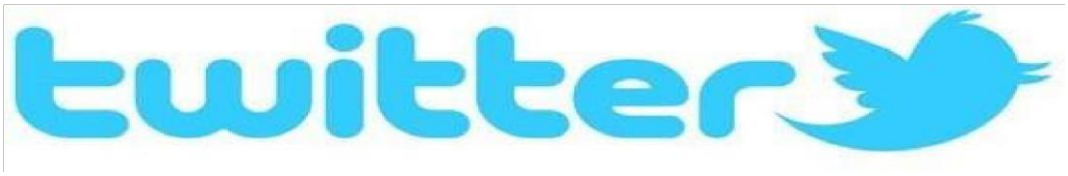


Fig 7.1 : Twitter

7.1 How Sentiment Analysis Works

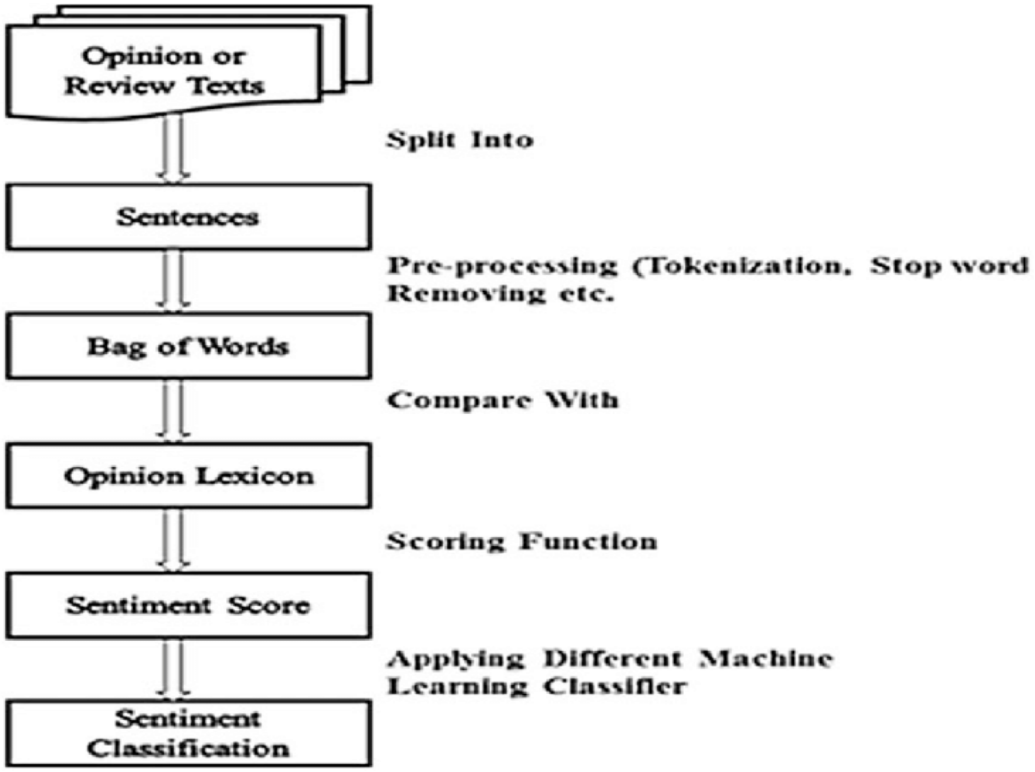


Fig 7.2 : Sentiment Analysis

7.2 Data Set Description[8]

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.

7.3 Importing the necessary packages :

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",category=DeprecationWarning)

%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.

train = pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/train.csv') train_original=train.copy()

7.4 Overview of the training dataset :

	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
5	6	0	[2/2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...
7	8	0	the next school year is the year for exams.ð□□...
8	9	0	we won!!! love the land!!! #allin #cavs #champ...
9	10	0	@user @user welcome here ! i'm it's so #gr...
10	11	0	â□□ #ireland consumer price index (mom) climb...
11	12	0	we are so selfish. #orlando #standwithorlando ...
12	13	0	i get to see my daddy today!! #80days #getti...
13	14	1	@user #cnn calls #michigan middle school 'buil...
14	15	1	no comment! in #australia #opkillingbay #se...
15	16	0	ouch...junior is angryð□□□#got7 #junior #yugyo...
16	17	0	i am thankful for having a paner. #thankful #p...
17	18	1	retweet if you agree!
18	19	0	its #friday! ð□□□ smiles all around via ig use...
19	20	0	as we all know, essential oils are not made of...
20	21	0	#euro2016 people blaming ha for conceded goal ...

Fig 7.3 : Training Dataset 1

17188	49151	black professor demonizes, proposes nazi style...
17189	49152	learn how to think positive. #positive #ins...
17190	49153	we love the pretty, happy and fresh you! #teen...
17191	49154	2_damn_tuff-ruff_muff__techno_city-(ng005)-web...
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 ...

17197 rows × 2 columns

Fig 7.4 : Training Dataset 2

7.5 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.

```
Test=pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/test.csv')

test_original=test.copy()
```

7.6 Overview of the test dataset

	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...
5	31968	choose to be :) #momtips
6	31969	something inside me dies ð□□¦ð□□¿â□¨ eyes nes...
7	31970	#finished#tattoo#inked#ink#loveitâ□¤ï¸□ #â□¤ï¸...
8	31971	@user @user @user i will never understand why...
9	31972	#delicious #food #lovelife #capetown mannaep...

Fig 7.5 : Test Dataset 1

17189	49152	learn how to think positive. #positive #ins...
17190	49153	we love the pretty, happy and fresh you! #teen...
17191	49154	2_damn_tuff-ruff_muff__techno_city-(ng005)-web...
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 ...

17197 rows × 2 columns

Fig 7.6 : Test Dataset 2

7.7 Data Pre-Processing :

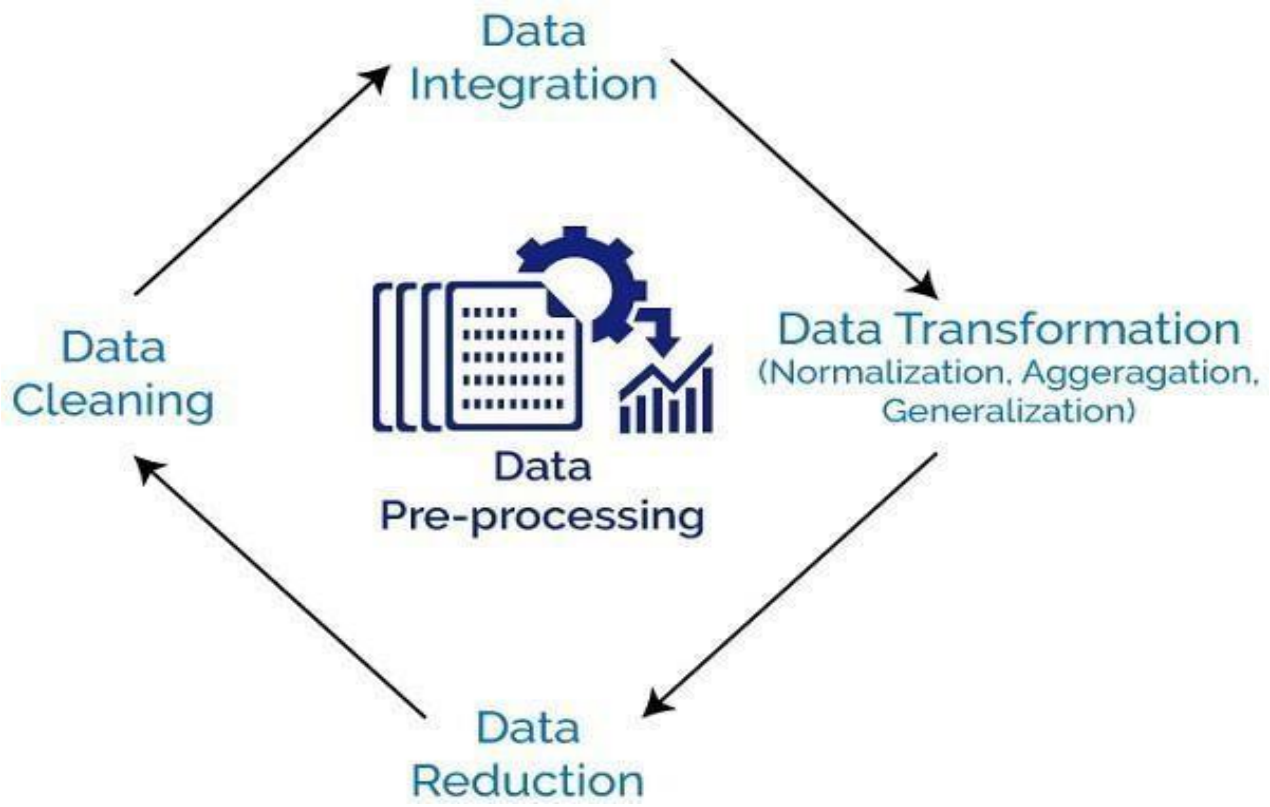


Fig 7.7 : Data Pre-Processing

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.

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

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.

```
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.

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

combine.head()

	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

Fig 7.8 : Removing Twitter Data

After removing the twitter handles.

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.

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

combine.head(10)

	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

Fig 7.9 : Removing Punctuation, Numbers, and Special Characters

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.

```
combine['Tidy_Tweets'] = combine['Tidy_Tweets'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
combine.head(10)
```

	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

Fig 7.10 : Removing Stop Words

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.

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

```
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
```

Fig 7.11 : Results after Tokenization

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”

```
from nltk import PorterStemmer ps = PorterStemmer()
tokenized_tweet = tokenized_tweet.apply(lambda x: [ps.stem(i) for i in x])

tokenized_tweet.head()
```

```
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
```

Fig 7.12 : Stemming

7.8 Now let’s stitch these tokens back together.

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


combi['tidy_tweet'] = tokenized_tweet
```

7.9 Story Generation and Visualization from Tweets

In this section,[3] we will explore the cleaned tweets text. Exploring and visualizing data, no matter whether its text or any other data, is an essential step in gaining insights. Do not limit yourself to only these methods told in this tutorial, feel free to explore the data as much as possible.

Before we begin exploration, we must think and ask questions related to the data in hand. A few probable questions are as follows:

- What are the most common words in the entire dataset?
-
- What are the most common words in the dataset for negative and positive tweets, respectively?
-
- How many hashtags are there in a tweet?

Understanding the common words used in the tweets: WordCloud

Now I want to see how well the given sentiments are distributed across the train dataset. One way to accomplish this task is by understanding the common words by plotting wordclouds.

A wordcloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes.

Let's visualize all the words our data using the wordcloud plot.

```
all_words = ' '.join([text for text in combi['tidy_tweet']])

from wordcloud import WordCloud

wordcloud = WordCloud(width=800, height=500, random_state=21,
max_font_size=110).generate(all_words)


plt.figure(figsize=(10, 7))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis('off')

plt.show()
```

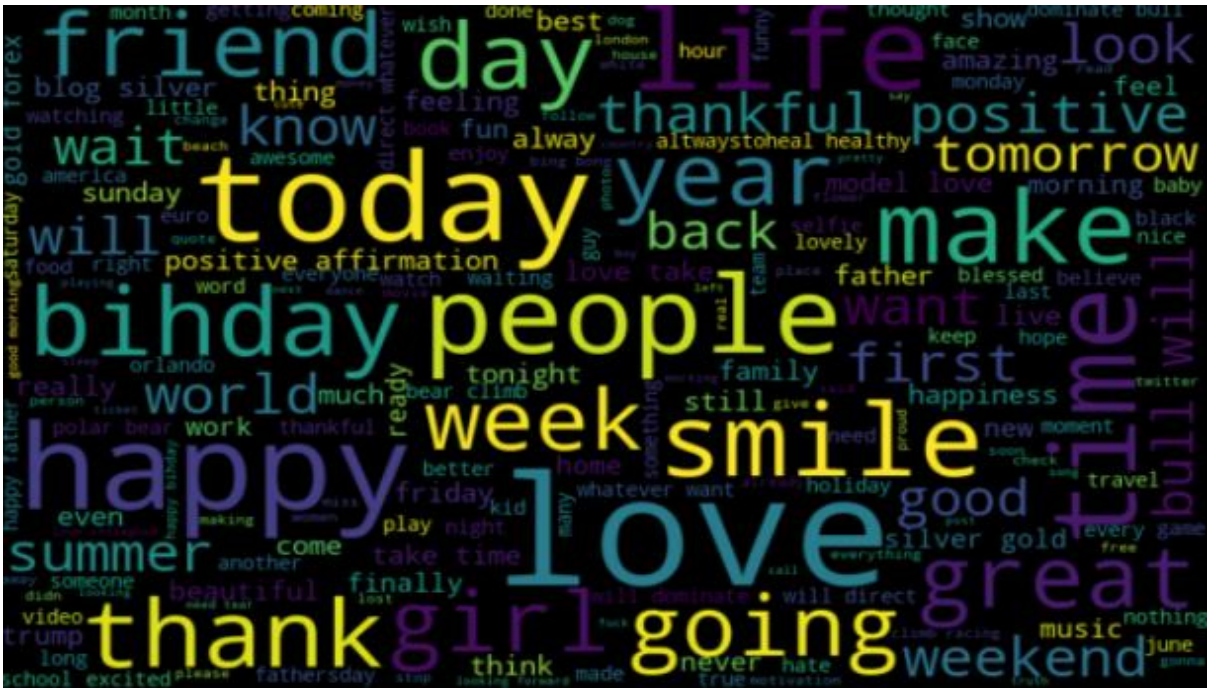


Fig 7.13 : Worldcloud

We can see most of the words are positive or neutral. With *happy* and *love* being the most frequent ones. It doesn't give us any idea about the words associated with the racist/sexist tweets. Hence, we will plot separate wordclouds for both the classes(racist/sexist or not) in our train data.

7.10 Words in non racist/sexist tweets

```
normal_words =' '.join([text for text in combi['tidy_tweet'][combi['label'] ==
0]])


wordcloud = WordCloud(width=800, height=500, random_state=21,
max_font_size=110).generate(normal_words)

plt.figure(figsize=(10, 7))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis('off')

plt.show()
```

Fig 7.14 : Non Racist Worldcloud

7.11 Racist/Sexist Tweets

```
negative_words = ' '.join([text for text in combi['tidy_tweet'][combi['label'] ==
1]])

wordcloud = WordCloud(width=800, height=500,

random_state=21, max_font_size=110).generate(negative_words)

plt.figure(figsize=(10, 7))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis('off')

plt.show()
```

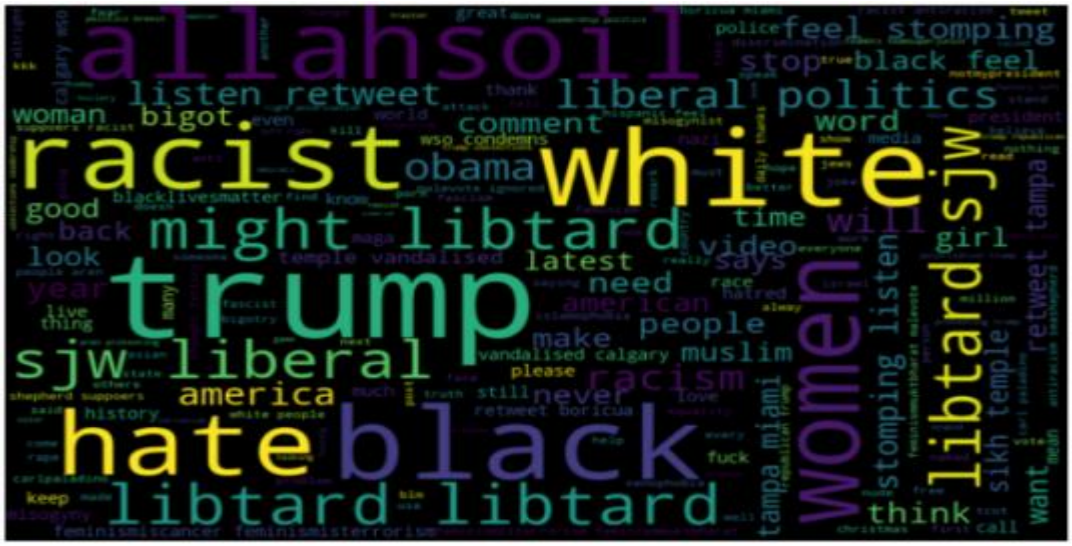


Fig 7.15 : Racist Worldcloud

CHAPTER 8 : PROJECT REUIREMENTS

- Python 3.7.3 IDLE
- HTML (Hyper Text Markup Language**)**
- CSS (Cascading Style Sheets)
- JAVASCRIPT
- Bootstrap
- VS Code
- Google.collab

8.1 Language Used:

- **Python**

8.2 Packages Used :

- asgiref==3.3.4
- click==8.0.1
- Django==2.2
- joblib==1.0.1
- nltk==3.6.2
- numpy==1.21.2
- oauthlib==3.1.1
- pandas==1.3.2
- PySocks==1.7.1
- pytz==2021.1
- regex==2021.8.28
- requests-oauthlib==1.3.0
- sqlparse==0.4.1
- textblob==0.15.3
- tweepy==3.10.0

CHAPTER 9 : CONCLUSION & FUTURE SCOPE

Conclusion

We have seen that sentiment analysis can be used for Analyzing opinions in blogs, articles, Product reviews, Social Media websites, Movie-review websites where a third person narrates his views. We also studied NLP and Machine Learning approaches for Sentiment Analysis.

Sentiment Analysis has many applications and it is an important field to study. It has Strong Commercial interest because Companies wants to know how their products are provided and also Prospective consumers want to know what existing customer think.

Sentiment analysis or opinion mining is a field of study that analyzes people's sentiments, attitudes, or emotions towards certain entities. This paper tackles a fundamental problem of sentiment analysis, sentiment polarity categorization. Online product reviews from Amazon.com are selected as data used for this study. A sentiment polarity categorization process (Figure 2) has been proposed along with detailed descriptions of each step. Experiments for both sentence-level categorization and review-level categorization have been performed.

Future Scope

Sentiment analysis[5] is a uniquely powerful tool for businesses that are looking to measure attitudes, feelings and emotions regarding their brand. To date, the majority of sentiment analysis projects have been conducted almost exclusively by companies and brands through the use of social media data, survey responses and other hubs of user-generated content. By investigating and analyzing customer sentiments, these brands are able to get an inside look at consumer behaviors and, ultimately, better serve their audiences with the products, services and experiences they offer.

In our exploratory study, investigation on the concurrent result of exploring several kinds of news with past numeric values for understanding stock market trend. The proposed model has worked on the views/ opinions of the reviewers on the shares.

The views of the experts affect to the traders who want to invest into the market. Our proposed model has enhanced the forecast accuracy on the upcoming trend of stock market, by analyzing several types of everyday news with several values of numeric characteristics in a time domain.

Further, research work can be applied in sequence to get more improved results in this domain. In future, the result of many other feature selection methods can be explored. Additional researches can be performed for evaluating the effect of numerous areas and domain-based factors. Enhancing the application of sentiment analysis to extended areas may result in interesting facts.

In future, more mixture of n-grams and feature allowance giving a improved accuracy than the current one can be used. This research is only linked to sentiment categorization into two basic

classes called binary classification. Sentiment can belong to either a positive emotion or a negative emotion. For improvement in future, more than one class for sentiment categorization can be considered like neutral, negative, positive etc.

In this paper, the emphasis is on discovering features which appear clearly in the form of nouns or nominal phrases in the evaluations. The implicit feature study is kept for future work. Since the ensemble learning approaches require much computation time, the parallel computing methods can be explored to handle this issue.

A main constraint of ensemble learning approaches is the absence of result depiction and the information gathered by ensembles is tough for human understanding. So, improvement of ensemble interpretability can be another vital research track.

Future opinion analysis structures need wider and in depth general knowledge base which will result in a improved consideration of natural language opinions and will be more effective to link the gap among multimodal and machine sensible data.

Combination of scientific theories regarding emotion with the applied technology targets to analyze sentiments of natural language script will result in additional bio-inspired methods to design intelligent opinion analysisstructures able to handle semantic information, analogy creation, learning actual information, and perceiving, identifying, and sensing emotions.

During design and development of our proposed model, we will be discussed and analyze the performance of four prominent machine learning algorithms. We have explored, how these algorithms are effective in predicting stock market trends based on user reviews and comments.

We have also seen that there is a vast scope of sentiment analysis in predicting the stock market value which will reduce the risk of the share investors.

Confusion Matrix can also be used to identify classifier performance based on sample data. And also show how our proposed model can improve prediction accuracy. We will also investigate the effects of pandemic using live examples of the effects caused by corona virus infections around the world.

We will analyze several types of news with numerical values to enable understand trends prevalent in stock markets based on historical patterns. In our proposed model, we will consider views and opinions posted by reviewers on shares.

The views shared by reviewers may influence traders investing in stock market and also, we will prove that our proposed model can predict market trends as there is a firm co-relation between news on pandemic and spread of infectious disease with that of movement of share prices in stock markets. In future, research work can be expanded to include individual share prices and combine daily news and posts gathered from multiple financial and stock market web sites.

REFERENCES

1) https://towardsdatascience.com/social-media-sentiment-analysis- (Last Visited : 08/05/22)

2) https://en.wikipedia.org/wiki/Pandas_(software) (Last Visited : 08/05/22)

3) https://en.wikipedia.org/wiki/Python_(programming_language) (Last Visited : 08/05/22)

4) http://ijece.iaescore.com/index.php/IJECE/article/view/22285 (Last Visited : 08/05/22)

5) https://www.linkedin.com/pulse/future-sentiment-analysis-shahbazanwar#:~:text=The%20future%20of%20sentiment%20analysis%20is%20going%20to%20continue%20to,the%20consumers%20behind%20the%20screens (Last Visited : 02 /05/22)

6) https://towardsdatascience.com/social-media-sentiment-analysis49b395771197?gi=e49c56f6d865 (Last Visited : 02/05/22)

7) https://www.digitalocean.com/community/tutorials/how-to-performsentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk (Last Visited : 25/05/22)

8) https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysisdataset-python/ (Last Visited : 05/05/22)

9) https://www.lexalytics.com/technology/sentiment-analysis (Last Visited : 05/05/22)

10) https://www.investopedia.com/terms/h/html.asp (Last Visited : 08/05/22)

11) https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics (Last Visited : 08/05/22)

12) http://www.columbia.edu/~sss31/html/html-tags.html (Last Visited : 08/05/22)

13) https://kullabs.com/class-11/computer-science-1/web-page-designing/introduction-to-html-and-types-of-tags (Last Visited : 28/05/22)

14) https://www.mygreatlearning.com/blog/css-tutorial/ (Last Visited : 28/05/22)

15) https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS (Last Visited : 10/05/22)

16) https://blog.devmountain.com/what-is-css-and-why-use-it/ (Last Visited : 10/05/22)

17) https://www.javatpoint.com/project-created-by-Anant-Patel (Last Visited : 10/05/22)

18) https://monkeylearn.com/machine-learning/ (Last Visited : 26/05/22)

19) https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction (Last Visited : 26/05/22)

20) https://docs.tweepy.org/en/v3.5.0/getting_started.html (Last Visited : 26/05/22)

21) https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/ (Last Visited : 26/08 /21)

22) https://monkeylearn.com/machine-learning/ (Last Visited : 27/05/22)

23) https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_regression_algorithms_overview.htm (Last Visited : 28/05/22)

24) https://numpy.org/doc/stable/user/whatisnumpy.html (Last Visited : 29/05/22)