

Identifying Major Social Media Events from Twitter

**Shayma**

**University ID**

**Month/Year**

# Abstract

Social Media is slowly evolving. Evolving into a tool with multiple uses rather than just a platform for connecting with friends and family. People on social media have started realizing the prowess of social media over the years. Whether it be online marketing, fame gathering, news networks, forums or anything else from any domain, social media is now becoming the go-to for all.

This revelation has brought about the radical increase in data created on social media. The amount of data created makes it difficult for people to search or see the highlights or popular events on social media. This research project is an experimental approach towards handling this problem using Machine Learning. This project provides experimental data and analysis on the performance of Machine Learning for this problem as an output. The problem this research will be solving is the identification and extraction of major events from Twitter.

**Keywords:** Machine Learning, Experimental Data and Analysis, social media.

# Acknowledgement

My sincere gratitude to my dissertation supervisor <Insert Supervisor Name Here>. Their advice, constructive criticism and unwavering support have been invaluable throughout the process of this work. I will forever be grateful for their encouragement to pursue this avenue of research. I would also like to thank my family without whom this work would not have been possible.

Table of Contents

[Abstract 2](#_Toc74151376)

[Acknowledgement 3](#_Toc74151377)

[Table of Figures 5](#_Toc74151378)

[1. Introduction 6](#_Toc74151379)

[a. Technical Background 7](#_Toc74151380)

[i. Neural Networks 7](#_Toc74151381)

[ii. Support Vector Machines 9](#_Toc74151382)

[iii. Naïve Bayes Classifier 12](#_Toc74151383)

[iv. Count Vectorizer 13](#_Toc74151384)

[v. TD-IDF Transformer 13](#_Toc74151385)

[b. Research Question 15](#_Toc74151386)

[c. Aims and Objectives 16](#_Toc74151387)

[2. Literature Review 16](#_Toc74151388)

[a. Content Feature Extraction in the Context of Social Media Behaviour 16](#_Toc74151389)

[b. Feature Extraction and Analysis for Identifying Disruptive Events from social media 17](#_Toc74151390)

[c. Argument Extraction from News, Blogs, and social media 19](#_Toc74151391)

[3. Experimental Design 21](#_Toc74151392)

[a. Data Selection 21](#_Toc74151393)

[b. Data Preprocessing 22](#_Toc74151394)

[c. Machine Learning 24](#_Toc74151395)

[d. Algorithm Training and Application 24](#_Toc74151396)

[e. Performance Metrics 25](#_Toc74151397)

[i. Accuracy Score 25](#_Toc74151398)

[ii. Classification Report 25](#_Toc74151399)

[4. Results 26](#_Toc74151400)

[5. Discussion 29](#_Toc74151401)

[6. Conclusion 30](#_Toc74151402)

[7. Future Work 32](#_Toc74151403)

[8. References 33](#_Toc74151404)

# Table of Figures

[Figure 1 - Basic Architecture of a Neural Network 7](#_Toc74071802)

[Figure 2 - Popular Activation Functions 8](#_Toc74071803)

[Figure 3 - Binary SVM Hyperplane Graph 11](#_Toc74071804)

[Figure 4 - The first five instances of the dataset to see the appearance of the dataset. 17](#_Toc74071805)

[Figure 5 - Splitting the dataset's features and outputs. 18](#_Toc74071806)

[Figure 6 - Splitting the dataset into training and testing datasets. 18](#_Toc74071807)

[Figure 7 - Pipeline Initialization, Model Training using fit () and prediction using predict () functions. 19](#_Toc74071808)

[Figure 8 - Classification Report for Multinomial NB. 20](#_Toc74071809)

[Figure 9 - Accuracy Scores for all Algorithms Implemented 21](#_Toc74071810)

[Figure 10 - Classification report for Multinomial NB 21](#_Toc74071811)

[Figure 11 - Classification report for Support Vector Machine Classifier 22](#_Toc74071812)

[Figure 12 - Classification report for Multi-layer Perceptron 22](#_Toc74071813)

[Table 1 - Summary of the Results from the Experiment Section. 26](#_Toc74071853)

# Introduction

The analysis of data from social platforms has become a mandatory necessity to understand what the masses think about a certain event in the world (Rafea and Mostafa, 2013).

Social media innovation has also guided a period empowering remarkable freedom for people to interface with each other (Martinez, Tsou and Spitzberg, 2019). From amusement to legislative issues, news and discussions on arrangements of different sorts, web-based media is taking over quickly. This has likewise helped the creation of information dramatically. Analysts have now begun exploring for approaches to extricate data without any problem. Recent work in this domain has shown that even though these social platforms are meant to help people socialize, they are gaining higher popularity as a source of information (Alsaedi and Burnap, 2021).

Micro-blogging has also caught the attention of analysts in recent years as one of the most popular and convenient methods of sharing news and opinions on anything going around the world (Alsaedi and Burnap, 2021). An example for consideration is Twitter, a popular social media platform that is used to discuss real-world events (Alsaedi and Burnap, 2021). Since information sharing has become very easy, people tend towards sharing their opinions on whatever real-world event piques their interest.

Data mining from web-based media content has as of late become a functioning examination point due to the vast popularity of social media as a news sharing platform. Following early trials that demonstrated this type to be very trying for best-in-class calculations (Bontcheva and Derczynski, 2016). For finding arising information, there is today another and very incredible source: socially created content. Interpersonal organizations are gigantic (1.8 billion clients) and new (their substance is created while occasions happen) (Brambilla et al., 2017).

With such massive data available from social media platforms like Twitter, the existence of a tool that can automate the process of digesting huge data and make sense out of it has been prioritized (Rafea and Mostafa, 2013). This topic is not new; therefore, several researchers have put out their ideas on this topic with different approaches (Rafea and Mostafa, 2013). Any attempts in using data generated by social media platforms like Twitter and exploring it statistically depends on the identification of the features of the content data from Twitter (Neumann et al., 2017).

The extraction of features from content data brings in the Natural Language processing approach to handling data. There are various tools that the Natural Language processing domain of machine learning provides that can help in extracting the hidden features of textual data. Textual data refers to the content data from Twitter data. This research will employ the combination of a Count Vectorizer and a TF-IDF transformer to extract and mould the dataset into a feature set that can be easily used to train three multiclass classifiers namely; Multinomial Naïve Bayes Classifier, Support Vector Machine Classifier and Multi-layer Perceptron Classifier. The performances of these algorithms will be compared to determine which algorithm best suits this problem. The performance metrics and the algorithms will be used from the **scikit-learn** library. The performance metrics used for measuring the performance of the algorithms are **precision, recall, accuracy score and f1-score**.

The work in this document is organized as follows; section 2 provides detailed background on the problem statement and the related work done on the same topic. Section 3 will then discuss the experimental designs and the methodologies. Section 4 presents the results of the experiments. Section 5 discusses the results in the context of a wider field. Finally, section 6 concludes the work.

## Technical Background

This section provides a brief technical background and reviews relevant work. The notation used throughout this report is introduced here. Due to the context of this research, sections regarding various algorithms are focused on multiclass classification.

### Neural Networks

The inspiration for this algorithm arises from the biology of the human nervous system i.e. the network of neurons and their working inside a human brain. Initial implementations of this idea resulted in the most basic form of the “neural network” known as the feedforward neural network. The components of this neural network are multiple layers with different responsibilities that consists of nodes or neurons that are interconnected between layers. The responsibilities of the layers classify them into three classes’ i.e., **Input Layer**, which is responsible for taking in the data and distributing it to the next layers for processing by assigning them weights for differentiation. **Hidden Layer**, which is responsible for the identification of patterns in the dataset that lead to the classification of data into requisite classes. **Output Layer**, which provides the output and the number of neurons in the output layer = number of output classes in the dataset.

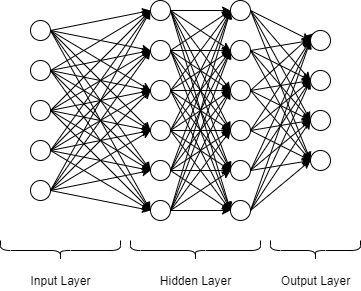
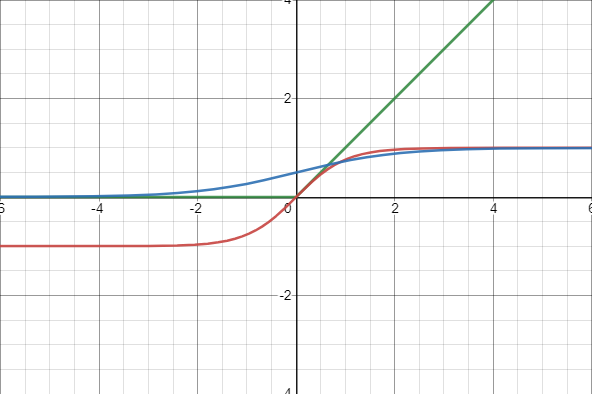


Figure 1 - Basic Architecture of a Neural Network

The working of a feedforward neural network is as follows; the input layer as mentioned before is responsible for taking in the data. This data is taken in the form of a vector. This data is then passed on forward to the next layer using weighted connections between the two layers. The responsibility of each neuron in the hidden layers is to calculate the weighted sum of the inputs received from all the neurons in the previous layer. The mathematical expression for this process is as follows:

An activation function is implemented in each of the neurons. This helps the neurons to express non-linear functions.



Tanh

Sigmoid

ReLU

Figure 2 - Popular Activation Functions

Figure 2 graphically demonstrates three of the most popular activation functions for neural networks. The red line shows the hyperbolic tangent, the blue line shows the infamous sigmoid function and the red line shows the Rectified Linear Unit (Glorot et al., 2010).

The ReLU was introduced by (Glorot et al., 2010) and it shed light on the issues of training Deep Neural Networks with previously popular activation functions like the sigmoid and tan (h). The issue with the hyperbolic tangent was that it was saturated by both high and low neuron outputs that resulted in smaller gradients. This resulted in a numerical underflow that discontinued backpropagation and hindered the training process. This is shown in the graph in Figure 1 where the red line representing the hyperbolic tan function has values both above and below zero while ReLU solves this problem by only activating neurons with values above zero. However, this working method of ReLU causes an issue again. Once a neuron is shut down, reactivating it is impossible for most optimization algorithms. To solve this issue, an asymptotic approach is taken with neurons below zero i.e. very small gradient values are assigned to them (Xu et al., 2015).

The output layer of a Deep Neural Network is referred to as logits which are un-normalized log probabilities represented by Z. To normalize these probabilities, a SoftMax function is used to create a normal probability distribution.

Training a Deep Neural Network refers to an optimization problem that focuses on the optimization of a function that is used as a performance measure for the Neural Network. But this performance measure is intractable in a Neural Network architecture. Therefore, a proxy function known as a cost function is used to determine the performance of the Neural Network. This cost function, denoted by J denotes how poorly the Neural Network performs. The problem assumes that in minimizing the cost function J, the intractable performance measure will increase. The cost functions for each Deep Neural Network are optimized according to the dataset they are meant to train on. The most common cost function used when it comes to multiclass classification is the cross-entropy loss defined as:

Where θ are the hyper-parameters of a Deep Neural Network. F(x) is the output of a Deep Neural Network when an input vector x is provided, y is the true classification of the input vector and F (y | x) is the score of confidence given to the correct output class by the algorithm.

The layers in the neural network each are represented by separate functions. Chaining the functions representing each layer can result in an expression that represents the complete Deep Neural Network. This characteristic of the Neural Network is used to calculate the gradient of the cost function concerning the hyper-parameters of the Neural Network. The algorithm preferred for computing these gradients is the backpropagation algorithm.

Training DNNs usually involves tuning the values of DNN parameters with a combination of gradient-based methods and backpropagation. Using calculated by backpropagation, DNN parameters can be updated to reduce the cost using gradient-descent based methods. Various such methods have been devised, but the most notable are stochastic gradient descent (SGD) (Bottou et al., 2010); root means squared prop (RMSProp) (Tieleman et al., 2012) and adaptive momentum estimation (Adam) (Kingma et al., 2014).

### Support Vector Machines

Vladmir Vapnik along with Alexey Chervonenkis created the infamous Vladmir-Chervonenkis theory for statistical learning from distribution-free data. This computational learning theorem is the seed of birth for the Support Vector Machine (Ukil, 2007). Using this work, Vapnir also in the same year introduced the idea of a linear classifier that separates data classes using hyperplanes optimized for that dataset (Ukil, 2007). However, the first wave did not catch on. The 1990s proved to be fruitful for the development of SVM when Bernhard Boser, Isabelle Guyon and Vladimir Vapnir introduced a method that provided a route to create non-linear classifiers by applying the concept of a kernel function to maximum-margin hyperplanes (Boser et al., 1992) (Ukil, 2007). The kernel function is an addition for the linear classifier that helps map the original observations on a higher dimensional plane to accommodate for non-linear behaviour.

It is one of the classical machine learning technique that can still assist in solving big data problems (Suthaharan, 2016). Support vector machines are algorithms that learn to classify data based on example (Noble, 2006). Its functionality is basic terms can be explained as the maximization of a mathematical function for a particular dataset (Noble, 2006). The SVM mathematical model can be explained with the following simple equation:

This equation represents the linear version of the Support vector machine algorithm (Suthaharan, 2016). The support vector machine algorithm can be used for both linear and non-linear applications (Suthaharan, 2016). The steps required for both the applications are the same in concept however differing in implementation. The steps required for linear SVM are:

* Mapping of the dataset into a response set.
* Division of the dataset.

The steps required for non-linear applications are as follows:

* Mapping of the dataset into a feature space by employing a kernel function.
* Mapping of the feature set into a response set.
* Division of the data domain.

(Suthaharan, 2016) explains this phenomenon mathematically in his study of Support vector machines as:

The application of the kernel function in the linear equation turns the SVM into a non-linear SVM (Suthaharan, 2016). The working of the SVM revolves around two mathematical concepts i.e., parametrization and optimization of the objectives obtained from the topology of the dataset domain.

Since this research employs multiclass classification, the linear and non-linear classifier version of the SVM will be discussed accordingly. The multiclass classification using a linear SVM is achieved by employing an ensemble of binary classifier SVMs. Considering a dataset, a binary classifier SVM divides the dataset into two different classes represented by the following equations (Suthaharan, 2016):

The parametrization objective can be defined based on the above two equations as follows:

These two parametrization equations define the straight lines also referred to as hyperplanes that define the boundaries between the binary classes in the dataset. The optimization objective can now be defined using the parametrization equations as the distance between the two boundaries of the binary classes of the dataset (Suthaharan, 2016). The assumption made here is that the hyperplanes are parallel to each other and the optimization objective is to maximize the distance between the two hyperplanes (Suthaharan, 2016). The maximization occurs on the hyperparameters of the SVM defined in the equations of the parametrization objectives (Suthaharan, 2016). The assumption that the hyperplanes are parallel to each other allows the definition of the optimization objective equation using the standard equation for the distance between two parallel lines given by (Suthaharan, 2016):

The resulting optimization equation is given by (Suthaharan, 2016) as; where the slopes of both the hyperplanes are equal to w and their intercepts b1 = γ+1 and b2 = γ-1. Equating these variables into the distance equation results in; (Suthaharan, 2016). However, in rea-word applications of the SVM, the equation is written using the mathematical norm form which results in (Suthaharan, 2016). This equation states that instead of the maximization of the distance term, the minimization of the error term w can be done i.e., the minimization of the error term w results in the maximization of the distance term. Therefore, the error function for SVM is as follows (Suthaharan, 2016):

This error function is critical in designing the optimization objectives for the SVM. The following figure provides a deeper insight into the working of a binary SVM.



Figure 3 - Binary SVM Hyperplane Graph

To explain clearly the working of a non-linear SVM, the perception of a linear graph must be boosted with additional parameters to increase the dimensions of the graph. Increasing the dimensions of a linear equation turns it into a quadratic equation (Vapnik, 1995). The hyperplane will then be represented by the following equation (Schölkopf and Smola, 2001):

Where n represents the number of training samples, yi represents the label value for each data instance, Ψ represents the kernel function responsible for mapping the linear objective to higher dimensions and α is the hyper-parameter that needs to be calibrated in a way that maximizes a Lagrangian representation (Ukil, 2007). A Lagrangian is a function of the dynamic variables of the system (Ukil, 2007). Subject to the constraints αi ≥ 0 and there is a value for αi for each training point and training points that are closer to the hyperplane have non-zero values for αi. These non-zero values for αi are called support-vectors (Ukil, 2007).

However, for real-world implementations, things like noisy data and other real-world biases need to be accounted for. To account for these, the equation for the hyperplanes is deflated by constraining the training points that are misclassified (Ukil, 2007). The equation now becomes; where provides the relaxation required to measure the misclassification from the constraints. To adapt to this, the optimization objective changes to become (Ukil, 2007):

Here, µ represents a regularization parameter responsible for governing the balance between maximizing the margin and maintaining the training error (Burges 1998). This machine that now utilizes the Support Vectors is referred to as the Support Vector Machine with two hyper-parameters (Burges 1998):

* The kernel function.
* The regularization parameter µ.

### Naïve Bayes Classifier

Naïve Bayes classifier is a simple learning algorithm that utilizes the infamous Bayes’ theorem from probability with an assumption that the features of an event are conditionally independent given the output class (Webb, 2016). This algorithm provides a technique for estimating the probability of an output class appearing given the feature vector of the dataset i.e., P (y | x). These estimates can then be further used for classification purposes (Webb, 2016).

To understand the functioning of the Naïve Bayes classifier, the understanding of the basic principles of the classifier itself is mandatory. A classifier is defined in (Murphy, 2006) as a function that maps the input vector onto the output vector. The input vector belongs to the feature space. It is normally assumed that the feature space is itself a subset of Real Numbers but the real world can provide datasets with features of any type whether discrete or continuous (Murphy, 2006). Another assumption made while using classifiers is that the data is unordered (categorical) and mutually exclusive (Murphy, 2006). If an input variable belongs to more than one output class, then the problem is a multiclass classification problem (Murphy, 2006). The goal therefore with classification is to learn the function that maps the input vector onto the output vector.

The input and the output vector comes from the dataset this research is working on. The dataset consists of columns that are features that result in output when combined in the real world. These features make up the input vector. The output vector will be the output these features result in. Let *x*, be the input vector and *y*, be the output vector. Therefore function *f* is the learning objective that requires x and y from **a labelled dataset** such that (xn, yn), n = 1: N; this is also referred to as supervised learning.

The probabilistic classifiers like the Naïve Bayes classifiers all return p (y | x) as the output (Murphy, 2006). Based on the methods to achieve this output, there are two types of probabilistic classification models (Murphy, 2006). The first method to achieve the desired output is to directly learn the function that provides the output p (y | x). This is the **discriminative model** (Murphy, 2006). It is referred to as the discriminative model since it differentiates between different output classes given the input (Murphy, 2006). The next method is to **class-conditional density** p (x | y) for each value of y and to learn **class priors** p (y). The application of the Bayes theorem to these parameters produces:

This is referred to as the generative model (Murphy, 2006). It is named so because of its ability to generate the features vector x for each output class ‘y’. A relatively easy approach to generative and discriminative learning is to not consider probability altogether and initialize a function that directly maps x to ‘y’ i.e., also known as the discriminant equation (Murphy, 2006).

#### Generative Classifiers

With the assumption that the number of classes C is small, the ***class prior*** can be estimated with ease using the function; where π represents the vector of class probabilities. Then, the Maximum Likelihood Estimation will be, where NC is the number of training examples that have class label c.

The issue that stands now is the determination of the class conditional density. The typical assumption that goes into this is that the parameters of each conditional distributions of the same type are independent (Murphy, 2006). This assumption allows the estimation of p (x | y) separately for each value of y. There are three methods of estimating class conditional densities:

1. Gaussian:
2. If the input vector x belongs to {0, 1} then, Bernoulli:
3. If x belongs to {1, …, K} then Multinomial:

The reason behind adding the “Naïve” in the Naïve Bayes theorem is the assumption that all the features of the data are conditionally independent given the class label (Murphy, 2006):

This equation is false since the features are usually dependent and there is a high degree of correlation in the dataset, but this model is easy to train and works well (Murphy, 2006).

### Count Vectorizer

Count vectorizer is one of the traditional methods for extracting hidden features from the text (Kulkarni and Shivananda, 2019). It is a process of converting categorical variables into features or columns by using 0s and 1s for different output classes i.e., binary (Kulkarni and Shivananda, 2019). The way a count vectorizer works is that for a given word in the dataset, it will provide the frequency of the word in the data as the encoding for the word (Kulkarni and Shivananda, 2019).

### TD-IDF Transformer

Term Frequency-Inverse Document Frequency is one among the foremost popular approaches for term weighting in today’s applications of tongue Processing (Aizawa, 2003). TF-IDF has been since an extended time considered an empirical metric, especially when seen from a probabilistic viewpoint (Aizawa, 2003). By its definition, tf–idf may be a metric that multiplies the 2 quantities tf and idf. Here, tf provides an immediate estimation of the occurrence probability of a term when it's normalized by the entire frequency within the document, or the document collection, counting on the scope of the calculation (Aizawa, 2003). the essential function consistent with this definition is that given a component of textual data like a document or a term, the importance of the component is expressed as a product of the probability that it occurs and therefore the amount of data that it represents (Aizawa, 2003).

TF-IDF (Term Frequency-Inverse Document Frequency) works by determining the frequency of words during a specific data instance/document compared to the inverse proportion of that word over the whole dataset/corpus (Ramos, 2003). This calculation provides the relevance of the word within the document as a ratio (Ramos, 2003). Words that are common during a single or small group of documents tend to possess higher TF-IDF numbers than common words like articles and prepositions (Ramos, 2003).

The mathematical representation of TF-IDF accommodates minor changes with different applications but the general approach works according to the following equation:

Where we, d represents the frequency of the word w in document d. |D| is the size of the complete dataset/corpus and fw, D represents the number of documents that the word w occurs in (Ramos, 2003).

The code is simple for TF-IDF. Given a query composed of a set of words, the WI, d is calculated by keeping a sum record of the fw, d and fw, D. Once the sum is complete, the WI, d can be easily calculated (Ramos, 2003). This can then be used to return a set of documents D\* that maximize the following term (Ramos, 2003):

The equation returns the documents in descending order. This is the traditional process for the implementation of TF-IDF (Ramos, 2003). The starting of TF-IDF can be traced back to information theory (Aizawa, 2003). Assuming a joint probability distribution P (x, y) where x belongs to X and y belongs to Y, then using P (x, y) and marginal distribution, the probability of x =>.

These equations simply convey the fact that in the event x is observed, the presence of y is confirmed in the frame of observation and vice versa (Aizawa, 2003). From information theory, the definition for the amount of information can be taken as Let a, b be two variables that represent distinct events from X, Y. The amount of information expected from these two distinct events, also known as the self-entropy denoted by H(X) and H(Y) can be calculated as (Aizawa, 2003):

This metric known as self-entropy represents the ambiguity in the certainty of the occurrence of the event under consideration (Aizawa, 2003). Self-entropy is directly proportional to the number of events under consideration. Now taking both events into the same frame of observation, the pairwise mutual information criterion denoted by M (a, b) is defined as (Aizawa, 2003):

Considering both the events ‘a’ and b into the same frame of observation, the pairwise mutual information criterion helps accommodate the interaction between both the events and the exchange of information due to the interaction (Aizawa, 2003). Applying the general property of conditional probability to the pairwise mutual information criterion gives the expected mutual information (Aizawa, 2003):

This representation of mutual information between two events can result in the derivation of an information-theoretic view of TF-IDF (Aizawa, 2003). Let us consider a document provided as a set of unordered terms. Let D be a set of documents, and W be the set of distinct terms contained in D. Let N represent the total number of documents and M represent the total number of terms in N documents. Let d denote the event of selecting a random document from D and w denote the event of selecting a random term from W. Let e, f be the random variables initialized by the events d and w. Introducing these random variables initialized from the events d and w, the situation now displays query submission and result in retrieval as probability distributions over D. The objective now, is to find the expected mutual information between e and f to observe how well the documents in D are specified by the query given. The self-entropy of the system is given by (Aizawa, 2003):

The two assumptions required at this stage are as follows (Aizawa, 2003):

1. All documents in D are potential candidates at the beginning.
2. The subset of D which contain the selected term f is known.

The second assumption means that the documents that do not contain the selected term do not contribute to the system in any way (Aizawa, 2003). Selecting a query term f from inside of d and representing the frequency of the occurrence of e in f using j and the frequency of w in the set D as J, the expected mutual information is calculated as (Aizawa, 2003):

The above equation represents the products of the tf in the form of j. and the idf factor is divided by the constant J. This shows that the information-theoretic view of tf-idf can be interpreted as the quantity required for the calculation of the expected mutual information (Aizawa, 2003).

## Research Question

The motivation behind this research is provided by the following question:

1. Machine Learning has always been at the forefront of real-world data problems. How well will Machine Learning be able to assist researchers and Twitter users in surfing material related to any topic/niche easily from the enormous density of existing data?

## Aims and Objectives

The aim of this research is to automate the process of the identification of the topic/niche of a tweet from Twitter. To achieve this aim, the following objectives are to be completed:

* **Data selection:** The research topic is highly specific i.e., to identify the topic/niche of a tweet under consideration. To proceed with this research topic, a highly specific dataset that contains tweets and is also properly annotated to denote the topic/niche of every tweet is required for this research.
* **Data preprocessing:** Due to the nature of this research topic, the dataset selected for this project will have tweets in the dataset. These tweets will be in textual form, i.e., categorical data. Therefore, processing the data so that the Machine Learning algorithms can be trained without any issues require the use of Natural Language Processing techniques namely Count Vectorizer and TF-IDF vectorizer to extract the hidden features from the textual feature in the dataset and convert it into a form that the algorithms can easily fit on.
* **Machine Learning:** The review of the problem statement and the dataset selected turns the Machine Learning paradigm towards multiclass classification. Three algorithms from this paradigm in Machine Learning will be applied in this research namely Support Vector Machine Classifier, Multilayer Perceptron and Multinomial Naïve Bayes algorithm.
* **Performance Metrics:** Once the training of the Machine Learning algorithms is complete, the performance of all the algorithms will be compared to conclude with the algorithm that is best suited for the purpose of topic identification of tweets. The performance metrics used will be well suited for multiclass classifications namely, F1-scores, Accuracy Scores, Precision and Recall.

# Literature Review

The importance of literature review is highly rated in research. The process of reviewing work done in the same domain on the same topic by felloe researchers to learn from them and their work is called Literature Review. This process provides integrity to the content in your paper through means of referencing and validity to the work done in the research. The following papers have been thoroughly studied for the Literature Review for this paper.

## Content Feature Extraction in the Context of Social Media Behaviour

The paper published by (Neumann et al., 2017) provides a potentially useful set for content features hidden within the depths of data generated by Twitter. The activity on social media, Twitter in the context of this research and the data generated by it has always been considered to be made by two parts i.e., metadata and content data (Neumann et al., 2017). Metadata for Twitter data refers to data like the time of tweet posting, username of the tweet author, number of retweets, etc. The content data refers to as the tweet text and content i.e., the body of the tweet.

(Neumann et al., 2017) claims that a large amount of content data collected can be used in such a way that content feature extraction can be applied to it to enable the classification of tweets. The features that (Neumann et al., 2017) extracts from this content data in his research are as follows:

* The reuse of complex terminology and the variation in the usage of words.
* The mean and variance for word length.
* Usage of Emojis.
* The frequency and presence of Hashtags
* Misspellings
* Vulgar Content
* Hot-button words and clickbait
* The rarity of words with respect to overall usage across twitter i.e., TF-IDF scores
* URLs in the content
* Tweet redundancy

A text vector for each user was created by summing and normalizing the extracted data from each of the user’s tweets (Neumann et al., 2017). This extraction of data from each of the user’s tweets is a multistep process with the following steps (Neumann et al., 2017):

* Collection of the each of the user’s most recent 200 tweets into a single collection.
* Converting the whole collection into Uppercase for term matching.
* Scanning the collection for emojis, vulgarity, hashtags, URLs and other features mentioned above to account for their presence or absence.
* Cleaning the collection of special characters.
* Creating a Redundancy score for the collection. This requires the use of heuristic tools namely, Euclidean Distance, RMS Distance, L1 Distance L-Infinity Distance, Cosine Distance, and the norm—weighted average of all these distances. These provide the sum and normalization of all the extracted features from the collection.

The above process resulted in the creation of a dataset with 23 features for each user that were extracted from the 200 tweets for each user. (Neumann et al., 2017) uses this process to create this dataset for the classification of tweets created by bots and humans. However, the process of extraction of features from content data of each tweet can work for this research. The TF-IDF method for feature extraction is applied inspired by (Neumann et al., 2017) in this research.

## Feature Extraction and Analysis for Identifying Disruptive Events from social media

The research done by (Alsaedi and Burnap, 2021) aims to classify disruptive content available on social media. Disruptive content means content that can threaten the social safety and security of people using and interacting with that content (Alsaedi and Burnap, 2021). To this extent (Alsaedi and Burnap, 2021) discusses in his paper, two types of features that can be useful in identifying disruptive content from social media. These features are referred to by (Alsaedi and Burnap, 2021) as temporal and textual features.

Microblogging has taken off with high intensity in the past few years according to (Alsaedi and Burnap, 2021) and this has made social media very popular regarding its ability to allow people to microblog freely and without any restrictions. Microblogging is done by users of social media of real-world topics that are highly sensitive in nature. These events, due to their sensitivity can cause massive waves that can pose a threat to the social security and safety of the users that interact with a microblog of a certain sensitive social event (Alsaedi and Burnap, 2021).

For this research, (Alsaedi and Burnap, 2021) has chosen Twitter to be the target social media service from which the data will be collected for the analysis of disruptive tweets and methods to identify them. (Alsaedi and Burnap, 2021) proposes a methodology that can help identify these disruptive tweets that involves an in-depth understanding of the features of these disruptive tweets.

To efficiently analyze the tweets for data patterns that can help classify the data into disruptive tweets, the following features have to be considered (Alsaedi and Burnap, 2021):

* Temporal Features: Related to the speed of the diffusion with time related to the “speed” of diffusion over time by highlighting the "quality" of tweets created by users in different time frames;
* Textual features: which are representative of the text content published to Twitter.

This research done by (Alsaedi and Burnap, 2021) primarily focuses on the improvement of the performance of classification algorithms by improving the feature extraction from the Twitter data. To this extent, the two types of features defined by (Alsaedi and Burnap, 2021) are hypothesized to define the features of disruptive tweets clearly. Therefore, to extract these features, and event detection framework is created.

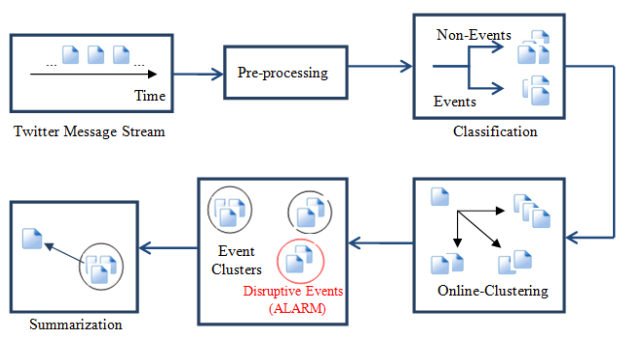


Figure - Event Detection Framework (Alsaedi and Burnap, 2021)

For the feature selection, (Alsaedi and Burnap, 2021) chose to implement the unsupervised method of feature selection implemented in (Mitra, Murthy and Pal, 2002). The advantage of using this method for this research is that is reduces the complexity involved in this process that results in acceptable computation times for even large datasets and real-time data digestion.

Temporal features have been ignored in many of the event detection and feature extraction researches related to social media. With eh sheer number of tweets being produced, the time for expiry for the tweets can be reduced an hour or even less. This reason has compelled (Alsaedi and Burnap, 2021) to use a range of time frames from which the most frequent terms can be extracted and utilized so as to keep the data collected relevant and fresh.

There exist two main tasks during this paper relating to text-based features: initial, (Alsaedi and Burnap, 2021) analyze totally different matter options so as to pick out the simplest contributors to the task of event detection. Second, (Alsaedi and Burnap, 2021) specialize in ranking options with performance measures in mind and take away immaterial options that may introduce bigger procedure value. These matter options area unit as follows:

* Near-duplicate measure: the typical content similarity over all pairs of tweets denotes within the previous hour.
* Retweet Ratio: this attribute is calculated by normalizing the quantity of times a tweet seems during a timeframe to the full number of tweets in this timeframe.
* Mention Ratio: The quantity of mentions for one user divided by the full number of tweets within the cluster.
* Hashtag Ratio: The magnitude relation of tweets containing hashtag over the full variety of tweets in this timeframe.
* uniform resource locator Ratio: The fraction of tweets with URL to the full variety of tweets during a timeframe.
* Tweet Sentiment: whether or not the tweet is negative or positive in nature.
* Dictionary-based Features: Grammatical options like tense, verbs, nouns, etc.

For testing their hypothesis that temporal and matter options outline unquiet events on Twitter, a dataset of one,698,517 tweets, and was collected from fifteen Gregorian calendar month 2013 to 05 Nov 2013 victimization Twitter’s Streaming API. The Formula one Motor athletics auto race was chosen thanks to its international interest. The formula used because the event classifier was the Naïve Thomas Bayes formula. The ensuing accuracy was measured by the F-measure performance metric to be 80.89% capable.

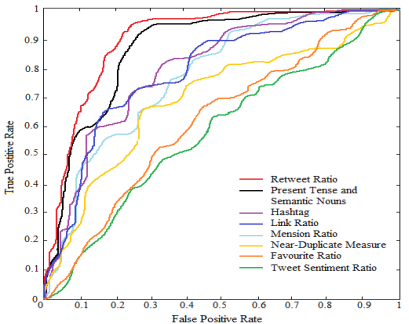


Figure - ROC curves for Textual Features (Alsaedi and Burnap, 2021).

## Argument Extraction from News, Blogs, and social media

(Goudas, Louizos, Petasis and Karkaletsis, 2014) gift a ballroom dancing approach for argument extraction from social media texts. throughout the primary step, the planned approach tries to classify the sentences into “sentences that contain arguments” and “sentences that don’t contain arguments”. within the second step, it tries to spot the precise fragments that contain the premises from the sentences that contain arguments, by utilizing conditional random fields (Goudas, Louizos, Petasis and Karkaletsis, 2014). Arguments is sometimes rotten into a claim and one or a lot of premises justifying it. Argument extraction from social media is more difficult as a result of texts might not forever contain arguments, as is that the case of legal documents or scientific publications sometimes studied. additionally, being less formal in nature, texts in social media might not even have correct syntax or orthography (Goudas, Louizos, Petasis and Karkaletsis, 2014).

(Goudas, Louizos, Petasis and Karkaletsis, 2014) study the pertinence of existing approaches on the domain of social media. Following a ballroom dancing approach, (Goudas, Louizos, Petasis and Karkaletsis, 2014) classify sentences as litigious (containing arguments) or not, through the employment of machine learning techniques, like logistical Regression, Random Forest, Support Vector Machines, etc. As a second step, Conditional Random Fields area unit used so as to extract segments that correspond to premises in litigious sentences. The main analysis axis as outlined by (Goudas, Louizos, Petasis and Karkaletsis, 2014) isn't to spot the simplest playing machine learning formula for the task, however rather to check the pertinence of options from the state of the art to the domain of social media. The options that we've got examined is classified in 2 categories: options elect from the progressive approaches, and new options that look promising for the domain of our application, that involves texts from social media.

The options taken from the progressive approaches are:

* Position: The position of the text within the text.
* Comma token number: the quantity of commas within the text.
* Connective number: the quantity of connectives within the sentence.
* Verb variety
* Cue words: This feature indicates the existence and therefore the variety of cue words (also called discourse indicators). Cue words area unit known through a predefined, manually made, lexicon (Goudas, Louizos, Petasis and Karkaletsis, 2014).
* Domain entities variety: this feature indicates the existence and therefore the number of entities mentions of named-entities relevant to our domain, within the context of a sentence.
* Adverb variety
* Word variety
* Word mean length: this can be a metric of the typical length (in characters) of the words within the context of a sentence

The performance metrics that will be used by (Goudas, Louizos, Petasis and Karkaletsis, 2014) in order to evaluate this approach is accuracy, precision, recall and F1-measure. The accuracy denotes the correctness of the prediction for the instances of both classes that are to be classified. In the case of (Goudas, Louizos, Petasis and Karkaletsis, 2014) where arguments are sparse compared to the sentences that do not contain arguments, accuracy is not enough as (Goudas, Louizos, Petasis and Karkaletsis, 2014) is mainly interested in the detection of sentences that contain arguments. Precision, recall and F1-measure can complement this task. Precision denotes how well the classifier can classify instances correctly within the performed classifications, whereas recall measures the fraction of relevant instances that are correctly retrieved from all the possible instances. F1-measure combines precision and recall as the harmonic mean (Goudas, Louizos, Petasis and Karkaletsis, 2014).

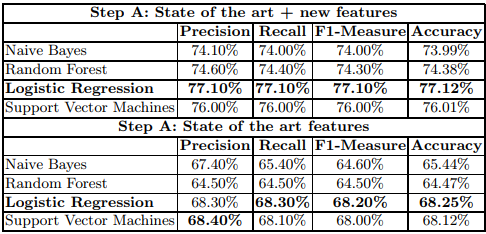


Figure - Results of the research done by (Goudas, Louizos, Petasis and Karkaletsis, 2014)

# Experimental Design

This research aims to develop a Machine Learning algorithm that can help classify between different niches and genres of tweets. To achieve this, the following steps are tracked:

* Dataset selection
* Data preprocessing
* Machine Learning
* Algorithm Training and Application
* Performance Metrics to determine the best Algorithm for the Job.

This section will explain in detail the procedures and processes executed to achieve the above-mentioned objectives. The experiment required intensive knowledge of the scikit-learn library for Machine Learning in Python. Further information on the library available here: <https://scikit-learn.org/stable/>.

## Data Selection

The dataset chosen for this research is the Twitter User Data available from <https://data.world/data-society/twitter-user-data>. The dataset originally contains 20,000 rows, each with a username, a random tweet made by the user, account profile and image and location information.

## Data Preprocessing

The dataset originally consisted of 20,000 instances along with 26 features. The purpose with which this dataset was created was to automate the process of identifying the gender of a Twitter user based on its profile information i.e., the features in the dataset. Therefore, the purpose for the creation of this dataset does not align with the aim of this research. The reason behind the selection of this dataset is the availability of random tweets from 20,000 different users.

The topic for this research is highly specific. It requires a dataset that consists of tweets that are labelled according to their niche i.e., fashion, science, news, etc. However, according to the research of the author in the limited timeframe of creating this thesis, the author did not come across a dataset that can fulfil this requirement. Therefore, the first step in data preprocessing was data cleaning. The only thing required from this complete dataset were the tweets. Therefore, the rest of the dataset was discarded and the tweets were preserved. The tweets in the dataset are represented by the **“text”** feature. Only the “text” feature was preserved, the rest of the features were discarded. The next step was to create a column in the dataset for the output labels since the project required it.

The reason behind the continuous insistence of the requirement of labels is that from the problem statement of this research, the following points are clear:

* The data to be dealt with in this research is textual.
* The algorithm needs to be able to classify between different types of tweets.

These points lead to two possible Machine Learning paradigms i.e., Multiclass classification since there can be an uncountable number of niches associated with tweets and Natural Language Processing since the research is dealing with textual data. Both of these approaches require that labels be present in the dataset. However, labels that satisfy the needs of this research are not available online. Therefore, manual annotation is preferred by the author. Once the useful features i.e., “text” was retained from the dataset, the author then proceeded to the next step, manual annotation. The author proceeded to read the text in the tweet and to the best of her knowledge proceeded with identifying the tweet niche and writing it down.

Since this process is highly time-consuming, the author labelled manually only 250 tweets for the project. The final dataset to be used now contains 250 tweets and two columns that are as follows:

* **Text**: the text content from the tweet.
* **Labe**l: the niche of the tweet, the output class of the dataset.

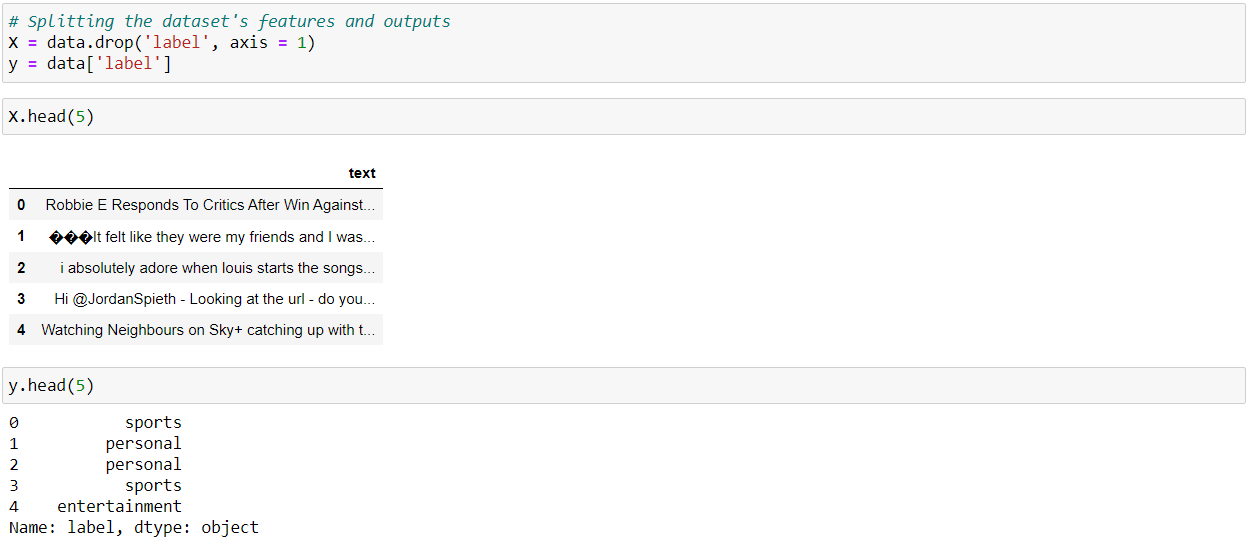
The output classes of the dataset after manual labelling and the number of instances associated with each class are as follows:

1. Anime: 4 tweets
2. Business: 6 tweets
3. Entertainment: 55 tweets
4. Erotic: 1 tweet
5. Fashion: 2 tweets
6. Fitness: 1 tweet
7. Food: 4 tweets
8. Health: 7 tweets
9. Music: 13 tweets
10. Nature: 2 tweets
11. News: 19 tweets
12. Personal: 58 tweets
13. Politics: 38 tweets
14. Promotion: 1 tweet
15. Religion: 2 tweets
16. Science: 3 tweets
17. Sports: 17 tweets
18. Tech: 16 tweets



Figure - The first five instances of the dataset to see the appearance of the dataset.

The dataset was also checked for any duplicate instances or any instances where the value for the “text” feature was null but anomalies and issues like this did not exist in the manually annotated dataset. The next step of data pre-processing is to split the features and the label in the dataset into two different data structures. This is done to make easy the process of passing the features to the algorithm for training. Instead of passing each feature separately, if the label is separated from the features itself then the complete feature set can be passed to the algorithm for training.

Figure - Splitting the dataset's features and outputs.

The next step after splitting the dataset as shown in Figure 5 is splitting the complete dataset into separate training and test dataset. This is possible using the ***test\_train\_split ()***from the scikit-learn library. Figure 6 shows the code snippet that makes this split happen.



Figure - Splitting the dataset into training and testing datasets.

As seen in Figure 6, the dataset is split with a ratio of 70:30 where the 30% is given to the test set while the remaining dataset is reserved for training purposes. The reason for doing this split is cross-validation. It is the process of validating how well the algorithm has been trained. This concept will further be explained in the Performance Metrics subsection.

## Machine Learning

The Machine Learning approach used for designing the experiment for this research is a hybrid between Multiclass Classification and Natural Language Processing. The popular approach when it comes to textual data is Natural Language Processing. The proceedings of a procedure like this would be the extraction of hidden features in the textual data and employ a probabilistic classifier like a Naïve Bayes classifier to train on and provide the predictions. The multiclass classification approach tries to learn the patterns in the data associated with each output class and predicts based on those patterns. Generic multiclass classification algorithms like the Support Vector Machines, Decision trees, Artificial Neural Networks also known as the Multi-Layer Perceptron are applied in multiclass classification. The reason behind considering multiclass classification for this research as an option is the existence of 18 output classes in the dataset.

The approach in this research is hybrid since the feature extraction methods from Natural Language Processing will be applied following Multiclass classification algorithms. The results will be compared to a pure Natural Language Processing approach using the Naïve Bayes algorithm. Thus, the final algorithms selected for application in this research are the Support Vector Machine, the Multi-layer Perceptron representing the Multiclass classification approach hybrid and the Multinomial Naïve Bayes algorithm representing pure Natural Language Processing approach. The reason behind employing the Multinomial Naïve Bayes algorithm is the existence of 18 output classes in the dataset.

## Algorithm Training and Application

For the implementation of the Machine Learning algorithms in the experiment, Pipelines from scikit-learn will be used to simultaneously apply the Natural Language Processing tools for feature extraction and the selected algorithms to the training dataset. Figure 7 shows the code snippet for the Pipeline initialization and the training process for each algorithm.

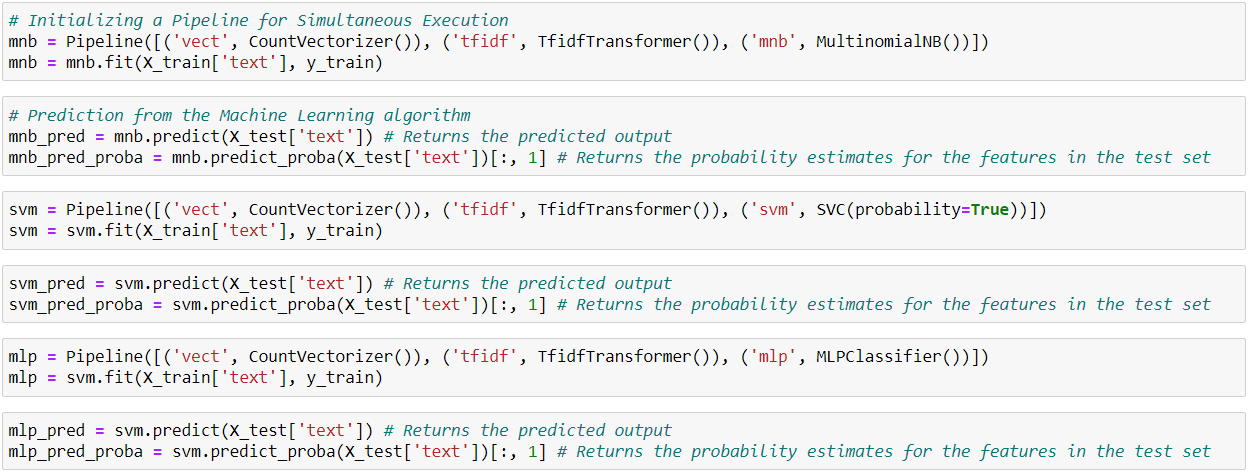


Figure - Pipeline Initialization, Model Training using fit () and prediction using predict () functions.

## Performance Metrics

Performance Metrics are mathematical values that provide a benchmark for the measurement of how well the algorithm employed has been trained and how well it will perform. This is done by using mathematical formulae and statistical approaches that return empirical values or graphs denoting the performance of an algorithm. There are multitudes of approaches for performance metrics (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021). Like the generic algorithms of Machine Learning, the approaches for performance metrics are also divided based on the Machine Learning Paradigm, i.e., Performance metrics for Regression and Performance Metrics for Classification. Since this project employs Classification algorithms, this section will concentrate on performance metrics for classification only.

Classification is the process of identifying patterns in the dataset that associate with each output class and predicting information based on the patterns identified. Various methods can be applied to classification algorithms to provide numerical and graphical output displaying the performance of an algorithm. All of these methods, however, do not apply to all classification algorithms equally (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021). Each of the performance metrics methods has a set of parameters that need to be followed and satisfied to successfully apply it to the algorithm. The performance metrics for Binary classification cannot be applied to Multiclass classification algorithms and vice versa (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021).

Some metrics are essentially defined for binary classification tasks like the ROC and AUC scores or the F1-Scores (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021). When extending the problem from binary to multiclass classification, the data is treated as a collection of binary problems i.e., one for each class. The problem now becomes easier since the only requirement is to map the performance metrics meant for binary classification across each class in multiclass classification (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021).

In cases like these, a complete grasp of the performance metrics methods must be present. There are three classification algorithms applied in this research in comparison. For each of the algorithms, different performance metrics have been applied. These performance metrics have been taken from the scikit-learn library for multiclass classification.

### Accuracy Score

In multiclass classification, this computes the subset accuracy i.e., the set of labels predicted for a sample must exactly match the corresponding set of labels in the test dataset. This function, provided by scikit-learn calculates accuracy either as or COUNT (correctly predicted samples == all samples). In pure mathematical form,

. , is the predicted value of the ith sample and yi is its corresponding true value (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021).

### Classification Report

This is an efficient function provided by scikit-learn. This function builds a text report showing the main classification metrics. The performance metrics shown in the classification report for multiclass classification are as shown in the figure below:

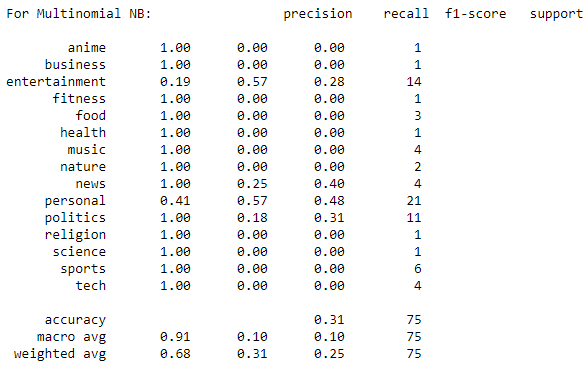


Figure - Classification Report for Multinomial NB.

Other measures of accuracy include precision and recall, f1-score and support. It is clear from Figure 8 that f1-score and support columns are empty because as mentioned before, f1-score is a performance metric for binary classification. The **precision** metric is the measure of the proportion of predictions that were accurate. Precision is calculated as (Classification: Precision and Recall | Machine Learning Crash Course, 2021). The **recall** metric is the measure of the proportion of the actual positives that were classified correctly. The recall is calculated as (Classification: Precision and Recall | Machine Learning Crash Course, 2021). To properly and effectively evaluate the performance of the model, it is mandatory to perform both the precision and recall tests. However, the issue is the clash between the precision and recall metrics i.e., improving the precision for algorithm results in the reduction of the recall metric and vice-versa (Classification: Precision and Recall | Machine Learning Crash Course, 2021).

# Results

The results from the **accuracy\_score** metric from scikit-learn is shown in Figure 9.

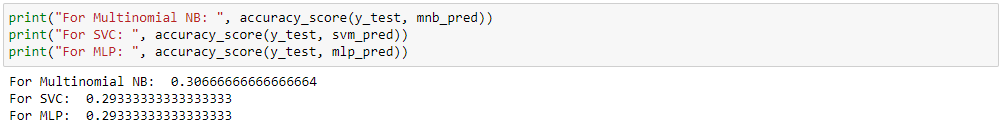


Figure - Accuracy Scores for all Algorithms Implemented

The classification reports for all the algorithms are shown in Figures 10 to 12 as follows:––

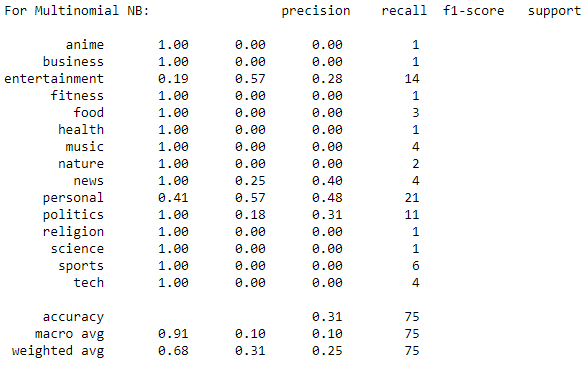


Figure - Classification report for Multinomial NB

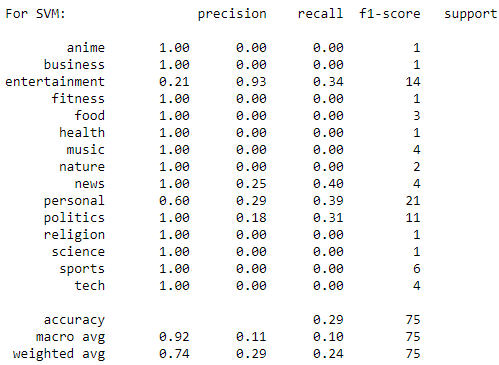


Figure - Classification report for Support Vector Machine Classifier

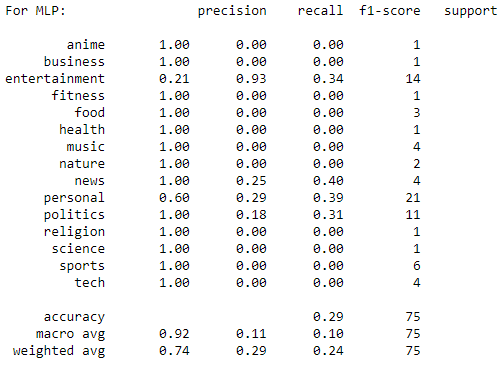


Figure - Classification report for Multi-layer Perceptron

The objective of the experiment in this research was to train multiple algorithms to identify the niche of the tweet under consideration and compare their performance to determine the right approach for achieving the aim. As it can be seen from Figures 9 to 12, the accuracy scores for all the algorithms lie in the range of 30%. The reason for such low accuracy from all the algorithms is the sparseness in the dataset. The sparseness in the dataset refers to the ratio of the number of classes to the number of instances per class.

This can be inferred from the classification reports of all the algorithms. If the precision and recall values for each output class in the classification report are observed, we can see that more than 50% of the output classes have 0 precision and 0 recall. The reason behind this is the sparsity of data per output class. This means that the algorithms do not have enough data per output class to learn the patterns and identify the patterns that lead to a specific output class. This causes a heavy inherent bias in the dataset since output classes like “personal”, “politics” and “entertainment” have more than around 50 data instances under them while the other classes have instances in the single digits.

Since the algorithms do not have enough data to train on per class and overall as well to counter the inherent bias, the results get limited to 30% for each algorithm. This limitation is caused by the availability of data. Since the data is manually labelled, only 250 instances were annotated due to time and resource limitations. The algorithms can perform much better if supplied with more annotated data. The reason behind choosing the manual annotation route is as discussed previously in the document i.e., the topic for this research is very specific and requires highly specialized data for proceeding with any kind of algorithm training.

Another observation in the experimentation process is the fact that no algorithm has been tuned for the dataset. The default settings that are provided by scikit-learn have been used. This can be observed in the code snippet provided in Figure 7. There are no parameters passed with custom tuning for any of the algorithms. On further analysis of the results from the classification reports of the algorithms applied, it can be determined that any changes to the hyper-parameter tuning of all the algorithms will not be effective and the results will not budge from the average of 30% overall. The reason behind this is not in the implementation of the algorithms but the dataset.

# Discussion

This research aimed to produce a Machine Learning algorithm that can identify the niche of any given tweet. This was done using the hybrid of a multiclass classification approach and also using Natural Language Processing. The results of the experiments conducted are interpreted concerning the aim and objectives of this research.

The experimental design addressing the aim of the project was set up. This strategy was to implement a hybrid of a multiclass classification and Natural Language Processing approach, i.e., to combine the use of a Count Vectorizer and a TD-IDF transformer with a Support Vector Machine (Wang and Xu, 2004) and a Multilayer Classifier instead of a typical Naive Bayes classifier and compare it to the traditional Natural Language Processing approach of employing a Multinomial Naïve Bayes algorithm. Throughout the experiment, three Pipelines were created, separately for each algorithm and they were trained to spew out the results.

The results were very mediocre with the SVM combination performing at a measly 29% (Wang and Xu, 2004), the Multilayer Perceptron performing at 29% and the traditional Natural Language Processing approach gaining ahead by just 2% at a poor 32%.

The first observation to make here is that the accuracy of three different algorithms has stopped at an almost similar accuracy ±2% difference between all. This drives to the conclusion that there may be something that is bottlenecking the classification ability of all the algorithm equally. Upon further inspection, the only factor commonly affecting all the three algorithms simultaneously is the dataset. The dataset passed to all the algorithms is the same, this similarity can be the only explanation behind why the accuracies of all the algorithms are capped at almost 30%.

Another observation to be made is in the classification reports of all the Machine Learning algorithms. The precision and recall for each algorithm are provided per output class and it can be noticed that the precision and recall values for many of the output classes are = 0. This means that the algorithm is not learning the patterns from the data associated with that particular class which is why it is incapable of identifying any of the inputs related to the output classes with 0 precision and 0 recall.

The performance metrics for all three algorithms were measured using the inbuilt functions from scikit-learn which are the **accuracy\_score** and the **classification\_report** (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021)**.** These provided great empirical insights into the performance of each algorithm and brought out the reasons for poor performance in the experimentation section. The observations made above will after further explanation clear the reason behind this mediocre performance and accuracy measures from all the algorithms.

The first observation was that all the algorithms had almost the same poor accuracy around an average of 30%. It was observed that since the only thing common between the algorithms is the dataset used, the dataset might be bottlenecking the classification ability of all the algorithms equally (Zhu, Vondrick, Fowlkes and Ramanan, 2015). The reason behind this is the dataset. The dataset has only 250 instances that were manually labelled by the author herself. The reason behind manual annotation is explained in subsection 3.a (Zhu, Vondrick, Fowlkes and Ramanan, 2015). Due to the lack of time and proper resources, the author could only manage 250 instances. Although 250 is not a bad number of instances in a dataset, it becomes an issue when the number of output classes increases in such a small dataset (Zhu, Vondrick, Fowlkes and Ramanan, 2015).

There are 18 output classes in a small dataset of just 250 instances. Among these 18 output classes, only 4 classes harbour the majority of the data instances and the rest of the classes have data instances in the single digits (Zhu, Vondrick, Fowlkes and Ramanan, 2015). The issue with this is that this induces a heavy inherent bias in the dataset where the algorithms pattern detection automatically moves towards where the concentration of data is more. This bias is due to the imbalance in the number of instances for each output class. The lesser the data for an output class, the lesser the training the algorithm receives in identifying that output class (Zhu, Vondrick, Fowlkes and Ramanan, 2015).

The second observation also supports the point made in the explanation of the first observation. The second observation points out that the precision and recall values for many of the output classes are equal to 0 (Zhu, Vondrick, Fowlkes and Ramanan, 2015). The reason behind that is the unavailability of enough data for the algorithm to train and identify patterns following those output classes and thus the precision and recall values are 0 (Zhu, Vondrick, Fowlkes and Ramanan, 2015). Another explanation is the fact that the dataset was split in a 70:30 ratio even after the dataset is so small. Therefore, the test dataset did not have samples that pertained to all the classes and neither did the training set. Due to the lack of data instances in both the test and training datasets, the precision and recall values for most of the output classes equal to zero (Zhu, Vondrick, Fowlkes and Ramanan, 2015).

Another observation that can be considered is the availability of the f1-score as a performance metric in the classification report for the Multinomial Naïve Bayes only (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021). It has been already mentioned previously in this document that f1-scores only work for binary classification problems however there are ways to map it onto a multiclass classification problem. The only issue was with the f1-score performance metric is that it does might not work with probabilistic classifiers but it did which is why the f1-score column is not empty in the classification report for Multinomial Naïve Bayes algorithm (3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation, 2021). The main issue, therefore, identified from the experiment is that there is no availability of a dataset that is specifically catered to fulfil the needs of this research and manual annotation is required. The main inference from this experiment that can be taken is that even though it is by a measly 2% but the traditional Natural Language Processing approach with the Multinomial Naïve Bayes algorithm performed better than the multiclass classification algorithms.

# Conclusion

The world continues to depend highly on social media to express opinions on many topics and niches across the world. Especially in times of the COVID-19 pandemic, the usage of social media has skyrocketed. The amount of data generated from just social media is increasing exponentially and rapidly. Social media is starting to turn into a source of information and news for many people across the world. Social media is also gaining popularity as a source of information due to its popularity. The main reason for this is the efficiency of social media in spreading a piece of text or any type of content across the world instantly.

This popularity of social media has resulted in the generation of tons of data from just social media. Social media platforms like Twitter which is preferred by even popular people, big names from various industries and famous politicians attract even more users to the platform which in turn initiates the production of more data on various topics, niches and issues of the real world. All of this data when considered has a lot of use and due to its sheer amount and the density of the data, it is not difficult to get lost in it while finding what you are looking for.

To make sense of all this data and to classify it based on the topic discussed in the content posted on social media, researchers have gathered to research and create tools that can automate the process of classifying and thus identifying the topic of the content posted on social media. The leading approach in this regard across the world is Machine Learning. In this research, the author also relies on the prowess of Machine Learning to create an algorithm for the automation of the process of identifying the topics of the tweets on Twitter. To that extent, the author selects a dataset, performs detailed data preprocessing on the dataset, designs an experiment to test the application of Machine Learning to this domain and records the results.

The dataset selected is accessed from ‘**data. world**’, a public repository for people and organizations to post datasets on various domains for different purposes. The dataset contained 20,000 instances originally which were reduced to 250. The reason behind such a drastic reduction is the fact that the author chose to manually annotate the dataset selected for this purpose of this research since the topic of this research is highly specific and datasets that satisfied were not available according to the extent of the research done and literature reviewed by the author.

The study of the problem statement and the dataset was necessary for the selection of the Machine Learning algorithm since the Machine Learning algorithms were classified according to the type of the problem and the dataset available i.e., regression and classification. Upon the analysis of the problem statement and the dataset, the Machine Learning paradigm that was decided upon was the classification paradigm. The objective of the experiment was to compare three different algorithms to see which algorithm performed the best for this scenario. The selection of the algorithms was done based on the type of the dataset and the datatype of the features in the data. The only feature required for this research were the tweets, therefore only the tweets were reserved from the dataset and the rest of the features from the dataset were discarded.

The only two columns in the dataset were the ones that contained the tweet content and the labels added by the author. The next step was to select and finalize the algorithms. After considering the dataset, since the dataset had only one categorical column, (categorical meaning textual in the context of this research), the idea of including Natural Language Processing in the experimental design seemed natural. Therefore, the algorithms selected were the Support Vector Machine Classifier, the Multilayer Perceptron and the Multinomial Naïve Bayes. The inclusion of the Natural Language Processing comes in the data preprocessing section of the experiment where the data was subjected to feature extraction using two tools from the Natural Language Processing archive namely the Count Vectorizer and the TF-IDF transformer. The purpose of using these tools was to extract keywords from the dataset according to each output class. These keywords are also referred to as hidden features since they are not normally visible separately in the dataset until extracted.

The combination of these Natural Language Processing techniques is common with probabilistic classifiers like the Naïve Bayes algorithm for handling textual data. Therefore, the combination of these techniques with multiclass classifiers can be seen as a hybrid approach for tackling the aim of this research. After the finalization of the algorithms, the time to fit them on the training set and test them by getting them to predict the test set has come. The training and test sets have been split from the same dataset of 250 instances in a 70:30 ratio. The reason for splitting the dataset into training and test datasets is for an algorithm performance measuring technique called cross-validation. Cross-validation is a technique to validate the training of the algorithm using a part of the dataset for testing since the labels for the test features are already present and the predictions can be compared to the labels of the test set to measure how accurately the algorithm performs prediction.

The results for the algorithms are summarized in the table figure below:

Table - Summary of the Results from the Experiment Section.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Accuracy Score (%)** | **Classification Report** | | | |
| **Accuracy (%)** | **Average Precision** | **Average Recall** | **Average F1-Score** |
| SVM | 29.33 | 29 | 0.92 | 0.11 | 0.1 |
| MLP | 29.33 | 29 | 0.92 | 0.11 | 0.1 |
| Multinomial NB | 30.66 | 32 | 0.91 | 0.1 | 0.1 |

# Future Work

The limited timeframe provided for this research since it was an academic endeavour has held back the author a lot in this research. However, with more time to invest in this research, it is possible to improve the project in many different aspects and modules. The first improvement comes to the coding section. The code for this research is done in the Jupyter notebook. The project code provides only a proof of concept that helps the reader to understand the basic aim of the research. However. The code can be improved to serve much better purposes. The code can have a front-end that can be made interactive for the users and researchers around the world to play with. The front end would take in a tweet URL or just a simple tweet text from the user and classify the tweet and display the results to the user along with the keywords highlighted.

The code requires more work in the design section as well. The data preprocessing section of the experiment requires extreme work and human resources for manual annotation. 20,000 instances must be completely annotated for this research. The handling of the text encoding of the dataset also was not performed and there were a lot of ASCII characters mixed in the dataset. These characters reduce the quality of the dataset and hinder the feature extraction process. These can be removed to further improve the feature extraction process.

There was significant confusion as to whether apply multiclass classification or Natural Language Processing. The features in the dataset were not in any way contributing to the label of the dataset since the dataset was created without a label. However, this claim by the author is not backed by evidence like correlation matrices. Therefore, including the use of correlation matrices and more features in the dataset in the future might help in better decision regarding the features in the dataset. The inclusion of Natural Language Processing in the experiment design is also one of the factors that resulted in the author neglecting the features of the dataset that were discarded. The clarity in the procedures to be applied in the experiment design is extremely important.

Another aspect that can be further improved in the code for the future is the application of hyperparameter tuning to the algorithms used in the research. There are many hyperparameters for each of the algorithm provided in the documentation of scikit-learn. The usage of those hyperparameters to tune the algorithm to suit the purposes of this research can be a vital addition to the code. The scikit-learn library provides a Grid Search function that automatically determines the perfect hyperparameters for a classification algorithm given the dataset features, dataset labels and the initialized model instance. Using the Grid Search after performing the proper data preprocessing as mentioned in the previous paragraph can significantly improve the results gained from the experiment performed.

# References

1. Alsaedi, N. and Burnap, P., 2015. Arabic Event Detection in social media. Computational Linguistics and Intelligent Text Processing, pp.384-401.
2. Bontcheva, K. and Derczynski, L., 2016. Extracting Information from social media with GATE. Working with Text, pp.133-158.
3. Neumann, S., Li, C., Lo, C., Lee, C., Rajwani, S., Sood, S., Foster, B., Hadgis, T., Savir, Y., Michaels, F., Ahmed, A., Bernobic, N. and Hollander, M., 2017. Content Feature Extraction in the Context of Social Media Behavior. Lecture Notes in Computer Science, pp.558-570.
4. Glorot, X. and Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, [online] 9, pp.249-256. Available at: <http://proceedings.mlr.press/v9/glorot10a.html> [Accessed 8 June 2021].
5. Lopardo, A. and Brambilla, M., 2018. Analyzing and Predicting the US Midterm Elections on Twitter with Recurrent Neural Networks. 2018 IEEE International Conference on Big Data (Big Data).
6. Martinez, L., Tsou, M. and Spitzberg, B., 2019. A Case Study in Belief Surveillance, Sentiment Analysis, and Identification of Informational Targets for E-Cigarettes Interventions. Proceedings of the 10th International Conference on social media and Society.
7. Rafea, A. and Mostafa, N., 2013. Topic extraction in social media. 2013 International Conference on Collaboration Technologies and Systems (CTS).
8. Wang, P., Xu, J., Xu, B., Liu, C., Zhang, H., Wang, F. and Hao, H., 2015. Semantic Clustering and Convolutional Neural Network for Short Text Categorization. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers).
9. Bottou, L., 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. Proceedings of COMPSTAT'2010, pp.177-186.
10. Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5 - rmsprop.
11. Kingma, D. and Ba, J., 2015. Adam: A Method for Stochastic Optimization. international conference on learning representations.
12. Ukil, A., 2007. Support Vector Machine. Power Systems, pp.161-226.
13. Suthaharan, S., 2016. Support Vector Machine. Machine Learning Models and Algorithms for Big Data Classification, pp.207-235.
14. Noble, W., 2006. What is a support vector machine? Nature Biotechnology, 24(12), pp.1565-1567.
15. Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine Learning, 20(3), pp.273-297.
16. Williamson, R., Smola, A. and Scholkopf, B., 2001. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. IEEE Transactions on Information Theory, 47(6), pp.2516-2532.
17. Burges, C., 1998. Data Mining and Knowledge Discovery, 2(2), pp.121-167.
18. Zaidi, N., Petitjean, F. and Webb, G., 2016. Preconditioning an Artificial Neural Network Using Naive Bayes. Advances in Knowledge Discovery and Data Mining, pp.341-353.
19. Murphy, K., 2006. Naive bayes classifiers. University of British Columbia.
20. Kulkarni, A. and Shivananda, A., 2019. Converting Text to Features. Natural Language Processing Recipes, pp.67-96.
21. Aizawa, A., 2003. An information-theoretic perspective of tf–idf measures. Information Processing & Management, 39(1), pp.45-65.
22. Ramos, J., 2003. Using tf-idf to determine word relevance in document queries.
23. Zhu, X., Vondrick, C., Fowlkes, C. and Ramanan, D., 2015. Do We Need More Training Data? International Journal of Computer Vision, 119(1), pp.76-92.
24. Scikit-learn.org. 2021. 3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 0.24.2 documentation. [online] Available at: <https://scikit-learn.org/stable/modules/model\_evaluation.html#:~:text=Metrics%20and%20scoring%3A%20quantifying%20the%20quality%20of%20predictions,-There%20are%203&text=metrics%20module%20implements%20functions%20assessing,Regression%20metrics%20and%20Clustering%20metrics.> [Accessed 8 June 2021].
25. Google Developers. 2021. Classification: Precision and Recall | Machine Learning Crash Course. [online] Available at: <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall> [Accessed 8 June 2021].
26. Mitra, P., Murthy, C. and Pal, S., 2002. Unsupervised feature selection using feature similarity. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(3), pp.301-312.
27. Goudas, T., Louizos, C., Petasis, G. and Karkaletsis, V., 2014. Argument Extraction from News, Blogs, and social media. Artificial Intelligence: Methods and Applications, pp.287-299.