## ■ Section 1 – Introduction:

It was decided to find a data set to make an algorithm to be able to determine if an article is fake news or not. We opted to use this data set. [1] This data set is divided into two separate CSVs, one of them contains all the fake news, and the other contains all the true and verified news. This dataset is a curated list of fact-checked articles by a reputable non-profit fact-checking newsroom, PolitiFact. Using these data sets, we were able to create 3 working machine learning models for data classification between fake and true news, with the help of natural language processing (NLP). We will be using 3 machine learning models: Logistic Regression, Support Vector Machine and Naive Bayes.

## Section 2 – Background:

### Logistic Regression

Since logistic regression is a highly useful classification algorithm, it is frequently employed for binary classification applications. Logistic regression, like linear regression, uses an equation as its representation. 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) (Ezukwoke & Zareian, 2019).

$$Logistic\ function = \frac{1}{1+e^{-x}}$$

Logistic regression does not require a linear relationship between input and output variables, unlike linear regression (Thorat, 2021). This is due to the odds ratio being transformed via a nonlinear log transformation. The dependent variable can only accept a certain number of values, indicating that it is categorical. When the number of possible outcomes is only two it is called Binary Logistic Regression.

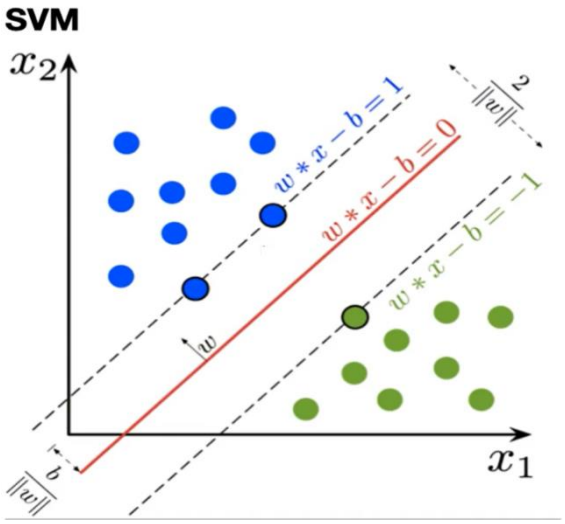
For logistic regression to achieve a high accuracy rate, the following assumptions must be made.

1. The outcome must be a binary value, e.g. Yes vs. No, 1 vs. 0, Fake vs. True.
2. There are no extreme values or outliers.
3. There are no high intercorrelations.

If the weighted sum of inputs is used as the output, like in Linear Regression, the result can be greater than 1. In this instance a value between 0 and 1 is required, as a result Linear Regression cannot be used, hence using Binary Logistic Regression instead.

## Support Vector Machine:

Support vector machine is a very popular algorithm that follows the idea to use a linear model and to find a linear decision boundary called a hyperplane that best separates the data. The best hyperplane is the one that represents the largest separation or the largest margin in between the two classes so that the distance from it to the nearest data point on each side is maximized.




The hyperplane must satisfy this equation w*x- b= 0 and we want to find the hyperplane so that the distance of both classes is maximized so we use the class +1 and -1 so the distance/margin is maximized.

**Linear Model**

$$w \cdot x - b = 0$$

$$\begin{aligned} w \cdot x_i - b &\geq 1 \quad \text{if } y_i = 1 \\ w \cdot x_i - b &\leq -1 \quad \text{if } y_i = -1 \end{aligned}$$

$$y_i(w \cdot x_i - b) \geq 1$$

We used the linear model:

W*X-B=0 and then the function should also satisfy the condition that:

W*X-B >= 1 for the class +1;

- all the samples must lie on the left side of this equation/this line here,

W*X-B <= -1 for the class -1;

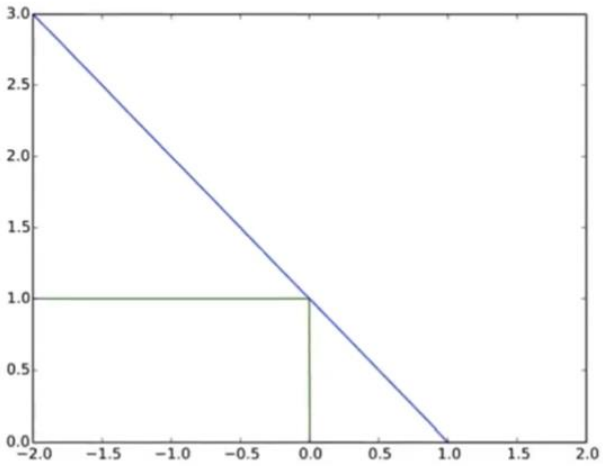
- all the samples must lie on the right side from this equation

Presented in one equation, it results in multiplying the linear function with the class label and this should be greater or equal than one.

So, this is the condition that we want to satisfy and now we want to come up with the W and the B for the weights and the bias and for this we use the cost function and then apply gradient descent.

**Hinge Loss**

$$l = max(0, 1 - y_i(w \cdot x_i - b))$$




$$l = \begin{cases} 0 & \text{if } y \cdot f(x) \geq 1 \\ 1 - y \cdot f(x) & \text{otherwise.} \end{cases}$$

The cost function in this case the hinge loss is split into 2 parts:

*Part 1:*

It is defined as; max of 0 & 1, minus Y multiplied by the linear model.

This means that if we plot the hinge loss (diagonal line), if Y multiplied by the function >= 1 it is 0.

- If they are correctly classified and are larger than 1 then the loss is zero, otherwise, the further away from the decision boundary line the higher is the loss

## Add Regularization

$$J = \lambda \|w\|^2 + \frac{1}{n} \sum_{i=1}^{n} max(0, 1 - y_i(w \cdot x_i - b))$$

if $y_i \cdot f(x) \geq 1$:

$$J_i = \lambda \|w\|^2$$

else:

$$J_i = \lambda \|w\|^2 + 1 - y_i(w \cdot x_i - b)$$

*Part 2:*

- To maximize the margin between these two classes; the margin is defined as; 2 over the magnitude of W;
- The margin is therefore dependent on the weight vector, so we want to maximize the margin and subsequently want to minimize the magnitude so add this to the cost function.
- Therefore, the magnitude of W to the power of 2 is multiplied by a lambda parameter and the hinge loss is added.
- The lambda parameter tries to find a trade-off between these two terms and selects the more important
- It is important to have the right classification to lie on the correct side of decision boundary line, however it is also important to have a line such that the margin is also maximized.

If we are on the correct side of the lines, if Yi multiplied F(x) is >= 1, then the only term (the magnitude of W to the power of 2 multiplied by a lambda parameter) is because this is the hinge loss is 0.

Else;

The cost function (the magnitude of W to the power of 2 is multiplied by a lambda parameter, plus 1 minus Yi(W*X-B) is to be minimized to get the derivatives or the gradients of the cost function.

**Gradients**

if $y_i \cdot f(x) \geq 1$:

$$\frac{dJ_i}{dw_k} = 2\lambda w_k$$

$$\frac{dJ_i}{db} = 0$$

else:

$$\frac{dJ_i}{dw_k} = 2\lambda w_k - y_i \cdot x_i$$

$$\frac{dJ_i}{db} = y_i$$

If Yi*f(X) is greater or equal than 1;

- The derivative is (2 * lambda * W), the derivative with respect to the B is 0.

If Yi*f(X) is not greater or equal than 1;

- The derivative with respect to the W is (2 * lambda * W) minus Yi multiplied by Xi and the derivative with respect to the bias is only Yi.

**Update rule**

For each training sample $x_i$ :

$$w = w - \alpha \cdot dw$$
$$b = b - \alpha \cdot db$$

The update rule where the new weight = the old weight minus the learning rate multiplied by the derivative.

## Naive Bayes.

Naïve Bayes is particularly useful when the probability of a class is determined by the probabilities of some other factor. For example, the relative frequency of all terms provided enough information to infer a belonging to a class. Or in other words, the supplied text was enough to determine if the news was fake or not.

**There are 3 options to use:**

**Multinomial**: Input is discreet value. Since the input is based on the whole language and its tokenization, it cannot be considered discreet.

**Gaussian**: Input follows a standard gaussian distribution. Since the input is based on tokenization of English text, this is the best option to use.

**Bernoulli/Binomial:** Input is binary (true/false or 1/0)

Assuming two events A and B, the probabilities P(A) and P(B) can be correlated with the conditional probabilities P(A|B) and P(B|A) using the product rule:

$$P(B \cap A) = P(B|A) \times P(A)$$

$$P(A \cap B) = P(A|B) \times P(B)$$

P(A|B) translates to probability of A occurring given B has occurred. Considering the intersection is commutative (AxB=BxA) such that P(AnB) = P(BnA), therefore Bayes Theorem can be derived using simple logic

$$P(A|B) \times P(B) = P(B|A) \times P(A)$$

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

In other words, the aim is to determine P(B|A) or in other words P(Outcome|Evidence) using P(A|B) which is the P(Evidence|Outcome) which is known from the training data.

P(A|B) is known as the posterior probability of the class. It's called posterior probability as is the revised or updated probability of an even occurring after taking into consideration new information.

P(B|A) is probability of likelihood of evidence. If there are more than one likelihood, they can be multiplied only because naïve Baynes assumes each is independent. This can be found by the training set and filtering all records where a particular class is determined.

P(A) is known as the Prior which is the overall probability. In other words, prior is count where the class was determined divided by the overall count.

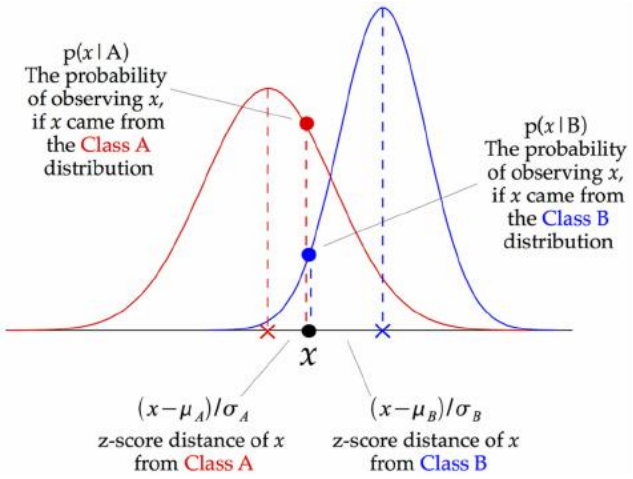
P(B) is the probability of evidence.

## Gaussian distribution in Naïve Bayes

Since our data is continuous data, an assumption is made that the values with each class are distributed accordingly to a Gaussian distribution. The equation for Gaussian distribution is

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Since the implementation uses Naïve Bayes there is an assumption of independence between dimensions. The model can be fit by using the mean and standard deviation of the points within each label.




## Normalisation and Rescaling:

Data normalization is a data-organization method used mainly in databases. Here normalisation is the process of modifying the form of a distribution and adjusting the values of numerical data to a common scale without affecting the range. It is critical to normalize a database in order to reduce redundancy (duplicate data) and guarantee that only related data is saved in each table. It also prevents any problems that may arise as a result of database adjustments such as insertions, deletions, and update.

When scaling, the range of the data being used changes. The practice of decreasing or extending data to fit inside a particular range is known as scaling. Scaling can be used to compare two distinct variables on an equal footing. This is very beneficial when dealing with variables that employ distance measurements. Because Euclidean Distance models are sensitive to distance magnitude, scaling aids in balancing the weight of all parameters. This is important because if one variable is more heavily weighted than the other, it introduces bias into the analysis. [2]

The methods used to transform the data:

## Min-Max Scaling:

The purpose of Min-Max scaling is to get the values closer to the mean of the column. This method adapts the data to a certain range, usually [0, 1] or [-1, 1]. One consequence of limiting this data to a narrow-specified range is that we will end up with lesser standard deviations, which will lessen the weight of outliers in the data. [2]

$$z = \frac{\mathrm{x} - min(x)}{[\max(x) - min(x)]}$$

### Cross Validation:

Cross-Validation is a statistical method for examining and comparing learning algorithms that separates data into two sections: one for learning or training a model and the other for verifying the model. It is a resampling approach in which the dataset is divided into two parts: training data and test data. The model is trained using train data, and predictions are created using test data that has not yet been observed. If the model performs well on the test data and has a high accuracy, it means that the model did not overfit the training data and can be used for prediction.

### Dimensionality reduction and feature selection:

Dimensionality reduction is the process of reducing features into a lower dimension while maintaining some of the original data's significant traits. Methods are classed as linear or nonlinear, as well as feature selection or feature extraction. Dimensionality reduction can be used to decrease noise, show data, perform cluster analysis, or as a preliminary step to aid in further studies. Feature extraction and dimension reduction can be combined in a single operation using principal component analysis (PCA).

Feature selection is just the process of selecting and excluding characteristics without changing them. The basic notion when adopting a feature selection strategy is that the data has certain features that are either redundant or unnecessary and may thus be eliminated without causing major information loss. Feature selection returns a subset of the features. A feature selection technique combines a search approach for recommending new feature subsets with an assessment measure for grading the different feature subsets. The most basic strategy is to analyse each candidate subset of features to see which has the lowest mistake rate.

### Quantitative measurements used:

Accuracy score will be used as the primary measurement. Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions the model got classified correctly. Formally, accuracy has the following definition:

Accuracy = [Number of correct predictions / Total number of predictions]

## ■ Section 3 – Experiments:

To process the data set:

1. We imported both the data sets; consists of true and fake news with up to 5000 rows.
    - Images: CD1, CD2
2. We created a text-based description column and added labels for each type of news has been added, where 1 represents true news and 0 represents fake news.
    - Images:CD3, CD4
3. After checking for any missing values, the final data frame has been created using only the description and label columns.
    - Images:CD5, CD6, CD7, CD8
4. With the final data set, we used NLP techniques to clean to data by removing all the stop words and punctuations. Then we used lemmatization and Bert tokenization to get the array form required. [3]
    - Images: PD1, PD2, PD3.

In natural language processing most of the data that we handle consists of raw text however machine learning models cannot read or understand text in its raw form, they can only work with numbers so the tokenizer's objective will be to translate the text into numbers. There are several possible approaches to this conversion and the objective is to find the most meaningful representation. Three distinct tokenization algorithms that we considered using where word-based, character-based and sub word-based. [4] [5] [6] We ultimately used a subworld based algorithm, bidirectional encoder representations from transformers (BERT) which produces more meaningful subworlds and ultimately regarded as more accurate.

- Images: PD4, PD5, PD6, PD7, PD8

The WordPiece tokenization was used to train BERT. It signifies that a word can be divided into several sub-words. This type of tokenization is useful when dealing with terms that are not in the dictionary, and it may assist to better represent difficult words. The sub-words are generated during the training process and are determined by the corpus on which the model was trained. Of course, we could use any other tokenization strategy, but we'll obtain the best results if we use the same tokenizer that the BERT model was trained on.[7] [8]

5. For the last experiment we checked the cross-validation score, and it didn't provide much help, as accuracies for every model remains like that of sklearn method.
    - Images: 3PM3

***Experiments carried out:***

1. The predictor variables are scaled using Minmaxscaler
2. Used PCA to get the most important features that explains up to 90% of the variance
3. Then we compared between PCA and kBest methods.
4. Feature selection keeps the features intact and chooses k best features among them, removing the redundant and co-linear features and gives us the best set of features which can be used for classification with low error rate. This set is a subset of original features.
5. Images: 3PM1, 3PM2, 3PM3, 3PM4.

***Implementation of the 3 ML techniques chosen:***

For all three models, that is Logistic, Support Vector and Naive Bayes' we used our own models. The models are built from scratch based on the first principle of python language.

## Logistic Regression:

For a logistic regression model to be implemented correctly, first a predictive function must be created. The model consists of two components, the sigmoid function and the features with weights. Implementing the sigmoid function allows for the feature inputs S(x) to return results between 1 and 0, hence resulting in its output to be used as the actual prediction.

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} = 1 - S(-x).$$

With the value outputted from the Sigmoid function, a loss function is used to evaluate the result. For the model to be trained, a gradient decent is used.

Using the gradient decent will allow the features to have their weights adjusted accordingly, whilst also updating the weights until the minimum is reached. For this to be done correctly, the function must loop over each epoch, loop over each row in the training data set and finally loop each coefficient and update it for the respective row.

Image:LR1, LR2, LR3, LR4.

## Support Vector Machine:

With reference to the technical description of support vector machines in section 2 , we proceeded to implement the mathematical algorithm in python code which is explained in the image below with the help of inline comments.

**Support Vector Machine Manual Implementation:**

In [48]:
```
class SVM:
    # learning rate @ default, lambda parameter @ default,number of iterations @ default )
    def __init__(self, learning_rate=0.001, lambda_param=0.01, n_iters=1000):
        # storing values
        self.lr = learning_rate
        self.lambda_param = lambda_param
        self.n_iters = n_iters
        self.w = None
        self.b = None

    # define 2 functions -  fit method and predict method
    # fit method: training of model using data
    # predict method: predicts the labels of the test samnples
    def fit(self, X, y):
        # our input vector X is in numpy ndarray where the number of rows is the number of samples
        # and the number of columns is the number of features
        n_samples, n_features = X.shape
        # making sure that y has only values of +1 and -1
        y_ = np.where(y == 0, -1, 1)
        # initialise w and b ( wieght and bias components)
        self.w = np.zeros(n_features)
        self.b = 0

        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                # Calculating the derivtive of the cost function with respect of W and B
                # If condition is Yi*f(X)>=1
                condition = y_[idx] * (np.dot(x_i, self.w) - self.b) >= 1
                # If true
                if condition:
                    #If Yi*f(X) is greater or equal than 1;The derivative is (2 * lambda * W)
                    self.w -= self.lr * (2 * self.lambda_param * self.w)
                # If false
                else:
                    # If Yi*f(X) is not greater or equal than 1;
                    # The derivative with respect to the W is (2 * lambda * W) minus Yi multiplied by Xi
                    self.w -= self.lr * (2 * self.lambda_param * self.w - np.dot(x_i, y_[idx]))
                    # The derivative with respect to B is only Yi.
                    self.b -= self.lr * y_[idx]

    def predict(self, X):
        # predit method,applying linear model w*x -b = 0 and deduce sign +1 or -1
        approx = np.dot(X, self.w) - self.b
        # returns the sign of the linear model
        return np.sign(approx)
```

Image: SVM1, SVM2

## Naive Bayes:

| **Text** | **Tag** |
|---|---|
| "A great game" | Sports |
| "The election was over" | Not sports |
| "Very clean match" | Sports |
| "A clean but forgettable game" | Sports |
| "It was a close election" | Not sports |

We want to predict what “A very close game” belongs to. The way this is done, is to calculate the probability that it is a sport and the probability it is not a sport. In other words, we want to calculate the following P(Sports | a very close game).

*Note: The above is an example. As with the assignment we don't run it on actual words but on the tokenization of such words*

The first thing we need to decide is what we are going to use as features. In the assignment we used tools like PCA but in this example we will use word frequency as the features. In the assignment we used lemmatisation and stemming to make sure similar words are grouped and provide a more accurate frequency count. The difference is that stemming removes ends of word (flower, flowers). Lemmatisation is based on context and therefore works on meaning (semantics) and not just the syntax. For example, better, good have the same lemma, good. But the word meeting for example can mean to meet someone or have a business meeting. Lemmatisation can differentiate whereas stemming cannot.

Using Baynes theorem, we need to define the following:

$$P(sports|a\,very\,close\,game) = \frac{P(a\,very\,close\,game|sports) \times P(sports)}{P(a\,very\,close\,game)}$$

Since we are trying to find which, one has the highest probability (if it's a sport or not) we can discard the denominator and just have:

$$P(a\,very\,close\,game|Sports) \times P(Sports)$$

And

$$P(a\,very\,close\,game|Not\,Sports) \times P(Not\,Sports)$$

As mentioned earlier, Baynes assumes that all the words are independent when calculating probability. That means, that the example becomes:

$$P(a\,very\,close\,game|Sports) = P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports)$$

**<u>Calculating probability</u>**

Step 1- calculate Priori which is based on the test data:

P(Sport) = 3/5

P(Not sport) = 2/5

Next, the probability is calculated based on word frequency. For example, the word 'game' is calculate using P(game|Sport) = 2/11

The sentence was picked because when we calculate the probability, for the word 'close' the frequency is 0. If in the end we will multiply all the probabilities for each word, the result is 0 if the probability of any word is 0. For this reason, we could have implemented Laplace smoothing which adds 1 to every count so it's never 0. In the assignment this was not done as logs where used, as to prevent small numbers from overflowing.

The possible words are (a, great, very, over, it, but, game, election, clean, close, the, was, forgettable, match)

To calculate the probability of a word we must use the number of times it appeared + 1 divided by the 14 words + the number of words used if it's a sport or not. In the example, there are 11 words that are in Sports (A great game, very clean match, a clean but forgettable game) and 9 which are not (The election was over, it was a close election).

Since the number are 14, using game as example we get

P(game|sports) = 2+1/11+14

The full results using LaPlace smoothing are:

| Word | P (word \| Sports) | P (word \| Not Sports) |
|---|---|---|
| **a** | **(2+1) / (11+14)** | **(1+1) / (9+14)** |
| **very** | **(1+1) / (11+14)** | **(0+1) / (9+14)** |
| **close** | **(0+1) / (11+14)** | **(1+1) / (9+14)** |
| **game** | **(2+1) / (11+14)** | **(0+1) / (9+14)** |

| Word | P (word \| Sports) | P (word \| Not Sports) |
|---|---|---|
| **a** | **0.12** | **0.08695217** |
| **very** | **0.08** | **0.04347869** |
| **close** | **0.04** | **0.08695217** |
| **game** | **0.12** | **0.04347869** |

Finally, the probabilities are multiplied, results below:

= P(a|Sports) x P(very|Sports) x P(close|Sports) x P(game|Sports) x P(Sports)

= 0.12 x 0.08 x 0.04 x 0.12

= 0.0000276

And

= P(a|Not Sports) x P(very| Not Sports) x P(close| Not Sports) x P(game|Sports) x P(Not Sports)

= 0.08695217 x 0.04347869 x 0.08695217 x 0.04347869

= 0.00000572

Since 0.0000276 > 0.00000572 the result is Sports

Images:NB1, NB2.

***Comparison against a third-party implementation of the same techniques.***

The comparison of the results:

1. For Logistic the training and testing accuracies using third party libraries are: 86% & 84% and using the first principle is: 74.8% & 74.4%.
2. For SVM the training and testing accuracies using third party libraries are: 95% & 86% and using the first principle is: 49.9% & 50.0%
3. For Naive Bayes the training and testing accuracies using third party libraries are: 69.8% & 69% and using the first principle is: 69.8% & 69%.

## ■ Section 4 – Conclusions:

***Draw conclusions from the experiments.***

- Feature Selection:
  - Feature selection with kBest and PCA doesn't give any clear recommendation about which model is better as both models give almost same result when comparing logistic and SVM accuracies at: 86% and 95% respectively. But while comparing Naive Bayes kBest method gives higher accuracy at 69% compared to 60% when using PCA.
- Model Selection:
  - Logistic regression provided maximum accuracy for training and testing when using both first principle python languages and third-party libraries at 74.8% & 74.4% with 86% & 84% respectively.
- First Principle vs Third party libraries:
  - Naive bayes model from scratch provided most similar result then other models, so it can be concluded that naive bayes from scratch was the, although in most of the cases third party libraries provided much higher accuracies.
- Cross Validation:
  - Using cross validation did not provided much improvement in the model, as all the accuracies remains almost similar.
- SVM:
  - The support vector machine model provides the least accuracy at 50% compared to its alternative version using sklearn, which gives accuracy at 95%.
- Further Improvements:
  - To further improve the model, we could have used hyper parameter tuning or different machine learning techniques, that might have improved the results.

# Works Cited


[1] C. Bisaillon, "Fake and real news dataset," [Online]. Available: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset.

[2] I. Lindgren, "Transformations, Scaling and Normalization," [Online]. Available: https://medium.com/@isalindgren313/transformations-scaling-and-normalization-420b2be12300.

[3] A. Ranjan, "https://medium.com/," Analytics Vidhya, 7 Mar 2020 . [Online]. Available: https://medium.com/analytics-vidhya/data-cleaning-in-natural-language-processing-1f77ec1f6406.

[4] HuggingFace, "Word-based tokenizers," [Online]. Available: https://www.youtube.com/watch?v=nhJxYji1aho.

[5] HuggingFace, "Character-based tokenizers," [Online]. Available: https://www.youtube.com/watch?v=ssLq_EK2jLE.

[6] HuggingFace, "Subword-based tokenizers," [Online]. Available: https://www.youtube.com/watch?v=zHvTiHr506c&feature=youtu.be.

[7] CodeEmporium, "BERT Neural Network - EXPLAINED!," [Online]. Available: https://www.youtube.com/watch?v=xI0HHN5XKDo.

[8] P. Prakash, "An Explanatory Guide to BERT Tokenizer," September 9, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/09/an-explanatory-guide-to-bert-tokenizer/.

[9] Bruno Stecanella, "A practical explanation of a Naive Bayes classifier", May 25th, 2017. [Online] Available: https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/

[10] Prateek Majumder, "Gaussian Naive Bayes", [Online] Available: https://iq.opengenus.org/gaussian-naive-bayes/