**FAKE NEWS DETECTION**

Name – Aditya Raj Reg no -11802784

Submitted to: Mr.Sagar Pande

Departement of Computer Science and Engineering

Lovely Professional university ,Punjab India

**Abstract**

In our modern era where internet is ubiquitous, everyone relies on various online resources for news. Along with the increase in use of social media platforms like Facebook, Twitter etc. news spread rapidly among millions of users within a very short span of time. The spread of fake news has far reaching consequences like creation of biased opinions to swaying election outcomes for the benefit of certain candidates. Moreover, spammers use appealing news headlines to generate revenue using advertisements via click-baits.

In this project, we aim to perform a binary classification of various news articles available online with the help of concepts pertaining to Artificial Intelligence, Natural Language Processing and Machine Learning.

**Introduction**

With the growing popularity of mobile technology and social media, information is accessible at one’s fingertips. Mobile applications and social media platforms have overthrown traditional print media in the dissemination of news and information. It is only natural that with the convenience and speed that digital media offers, people express preference towards using it for their daily information needs. Not only has it empowered consumers with faster access to diverse data, it has also provided profit seeking parties with a strong platform to capture a wider audience.

With the outburst of information, it is seemingly tedious for a layman to distinguish whether the news he consumes is real or fake. Fake news is typically published with an intent to mislead or create bias to acquire political or financial gains. Hence it may tend to have luring headlines or interesting content to increase viewership.

In the recent elections of India and other countries, there has been much debate regarding the authenticity of various news reports favoring certain candidates and the political motives behind them. Amidst such growing concerns, the detection of fake news gains utmost importance to prevent its negative impacts on individuals and society

The most common algorithms used by fake news detection systems include machine learning algorithms such as Support Vector Machines, Naïve Bayes, Stochastic Gradient Descent, Support vector classfier and so on. In this project we have attempted to implement someof these algorithms to train and test our results. We have used a combination of both off the shelf datasets as well as expanded it by crawling content on the web.

**Libraries used:**

**Numpy:-** NumPy is a general-purpose array-processing package. It provides a highperformance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with Python. As the whole project is based on whole complex stats ,we will use this fast calculations and provide results.

**Pandas:-** Pandas is the most popular python library that is used for data analysis. We will provide highly optimized performance with back-end source code with the use of Pandas.

**Matplotlib:-** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

**Scikit-learn:-** It is a Python library is associated with NumPy and SciPy. It is considered as one of the best libraries for working with complex data. There are a lot of changes being made in this library. We will use it for crossvalidation feature, providing the ability to use more than one metric. Lots of training methods like logistics regression will be used to provide some little improvements.

**Scikit.metrics-:** The sklearn. metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

**Classification \_report-:** A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report.

**confusion\_matrix-:**A confusion matrix is a summary of prediction results on a classification problem.The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix

The confusion matrix shows the ways in which your classification model is confused when it makes predictions.It gives you insight not only into the errors being made by your classifier but more importantly the types of errors that are being made.It is this breakdown that overcomes the limitation of using classification accuracy alone.

**Scikit.model\_selection-: Model\_selection** is a method for setting a **blueprint** to analyze data and then using it to measure new data. Selecting a proper model allows you to generate **accurate results** when making a prediction.

**model\_selection.train\_test split-:** train\_test\_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually.

By default, Sklearn train\_test\_split will make random partitions for the two subsets. However, you can also specify a random state for the operation.

**sklearn.feature\_extraction.text :-** The [sklearn.feature\_extraction](https://scikit-learn.org/stable/modules/classes.html" \l "module-sklearn.feature_extraction" \o "sklearn.feature_extraction) module can be used to extract features in a format supported by machine learning algorithms from datasets consisting of formats such as text and image.

**Feature extraction** is very different from [Feature selection](https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection): the former consists in transforming arbitrary data, such as text or images, into numerical features usable for machine learning. The latter is a machine learning technique applied on these features.

**TfidfVectorizer:-**TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in .

**HashingVectorizer-:** HashingVectorize sometimes used for text classification. HashingVectorizers require less memory and are faster (because they are sparse and use hashes rather than tokens) but are more difficult to introspect.

**sklearn.linear\_model :-** Linear \_model are a set of methods intended for regression in which the target value is expected to be a linear combination of the features.

**PassiveAggressiveClassifier-:**The passive-aggressive algorithms are a family of algorithms for large-scale learning. They are similar to the Perceptron in that they do not require a learning rate. However, contrary to the Perceptron, they include a regularization parameter C. (Passive: if correct classification, keep the model; Aggressive: if incorrect classification, update to adjust to this misclassified example.)

**sklearn.naive\_bayes :-**The Naive Bayes Classifier technique is based on the Bayesian theorem and is particularly suited when then high dimensional data.It’s simple & out-performs many sophisticated methods.

**naive\_bayes\_MultinomialNB:-** Suited for classification of data with discrete features ( count data ).Very useful in text processing.Each text will be converted to vector of word count .Cannot deal with negative numbers.

**SGDClassifier:-** Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) [Support Vector Machines](https://en.wikipedia.org/wiki/Support_vector_machine) and [Logistic Regression](https://en.wikipedia.org/wiki/Logistic_regression). Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning.

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing.

**SVM:-** Support vector machine (SVM) is a supervised machine learning algorithm which can be used for classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs.

**Linear\_SVC:-** The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our uses, though you can use this for many situations.

**Itertools:-** Python’s Itertool is a module that provides various functions that work on iterators to produce complex iterators. This module works as a fast, memory-efficient tool that is used either by themselves or in combination to form iterator algebra.

**Libraries used Model Deployment**

**Flask:-** Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

**Flask** – **Request** (Object)**:-**The data from a client's web page is sent to the server as a global request object. In order to process the request data, it should be imported from the Flask module. Form − It is a dictionary object containing key and value pairs of form parameters and their values.

**Flask-render\_template:-** The render\_template() function invokes the Jinja2 template engine that comes bundled with the Flask framework. Jinja2 substitutes {{ ... }} blocks with the corresponding values, given by the arguments provided in the render\_template() call.

**Flask\_Cors:-** A Flask extension for handling Cross Origin Resource Sharing (CORS), making cross-origin AJAX possible.This package has a simple philosophy, when you want to enable CORS, you wish to enable it for all use cases on a domain

**Joblib:-** Joblib is a set of tools to provide lightweight pipelining in Python.

**Pickle:-** Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk. What pickle does is that it “serializes” the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

**OS\_module:-**The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

**Newspaper\_module:-** Newspaper is a Python module used for extracting and parsing newspaper articles. Newspaper use advance algorithms with web scrapping to extract all the useful text from a website. It works amazingly well on online newspapers websites. Since it use web scrapping too many request to a newspaper website may lead to blocking, so use it accordingly.

**Urllib:-** Urllib is a Python module that can be used for opening URLs. It defines functions and classes to help in URL actions. With Python you can also access and retrieve data from the internet like XML, HTML, JSON, etc. You can also use Python to work with this data directly

**NLTK:-** The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

**For web desigining :**

HTML , CSS, JAVA SCRIPT, BOOTSTRAP

**Machine Learning**

Machine learning (ML) is the [scientific study](https://en.wikipedia.org/wiki/Branches_of_science) of [algorithms](https://en.wikipedia.org/wiki/Algorithm) and [statistical models](https://en.wikipedia.org/wiki/Statistical_model) that [computer systems](https://en.wikipedia.org/wiki/Computer_systems) use to perform a specific task without using explicit instructions, relying on patterns and [inference](https://en.wikipedia.org/wiki/Inference) instead. It is seen as a subset of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). Machine learning algorithms build a [mathematical model](https://en.wikipedia.org/wiki/Mathematical_model) based on sample data, known as "[training data](https://en.wikipedia.org/wiki/Training_data)", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as [email filtering](https://en.wikipedia.org/wiki/Email_filtering) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task

**Supervised learning** .

Supervised learning is the [machine learning](https://en.wikipedia.org/wiki/Machine_learning) task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from *labeled*[*training data*](https://en.wikipedia.org/wiki/Training_set) consisting of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples

**Prediction Algorithms** ‘

We implemented two different algorithms from scratch for the prediction model which were: Logistic Regression model and the Naïve Bayes classifier model. The algorithms and the details of implementation have been explained in the sections below. In addition to these we also trained and tested our dataset on two other models: Random Forests model and Support Vector Machine model. Given the short time frame of the project, the last two algorithms were prudently implemented with the help of scikit-learn libraries.

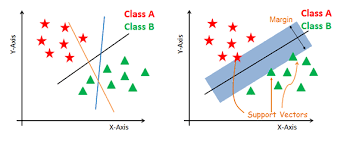
**Naïve Bayes Classifier**

This is a simple yet powerful classification model that works remarkably well. It uses probabilities of the elements belonging to each class to form a prediction. The underlying assumption in the Naïve Bayes model is that the probabilities of an attribute belonging to a class is independent of the other attributes of that class. Hence the name ‘Naive’.



**Support Vector Machine**

Support Vector Machines are machine learning models that perform supervised learning on data for classification and regression. When given a labeled training dataset, it computes the optimal hyperplane that categorizes the test data.



Data points are plotted in a multidimensional space, where the dimension is determined by the number of features at our disposal. The value of each feature is mapped to a point in the coordinate system. The algorithm then performs classification by finding the hyperplane that differentiates the two classes well.

The hyperplane having the maximum margin between the two classes in chosen. The advantages of the SVM model are that it performs very well for high dimensional spaces and also creates a clear margin of separation between data points. The disadvantages of using SVM were that it takes greater time to train the model compared to other models, especially when the dataset is large.

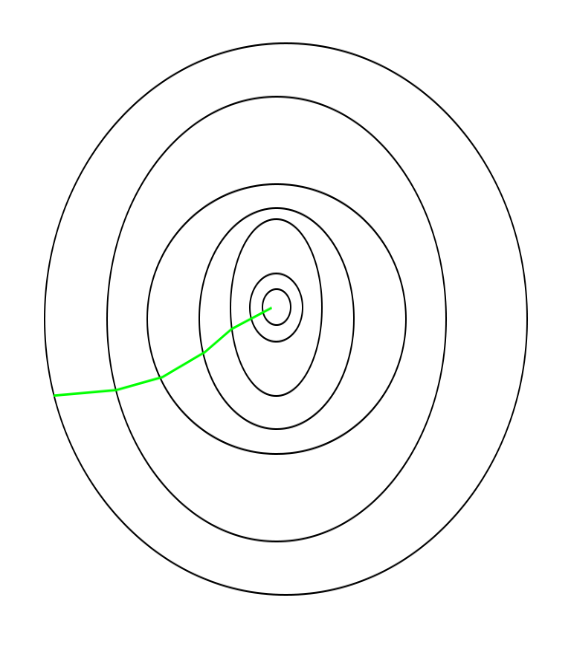
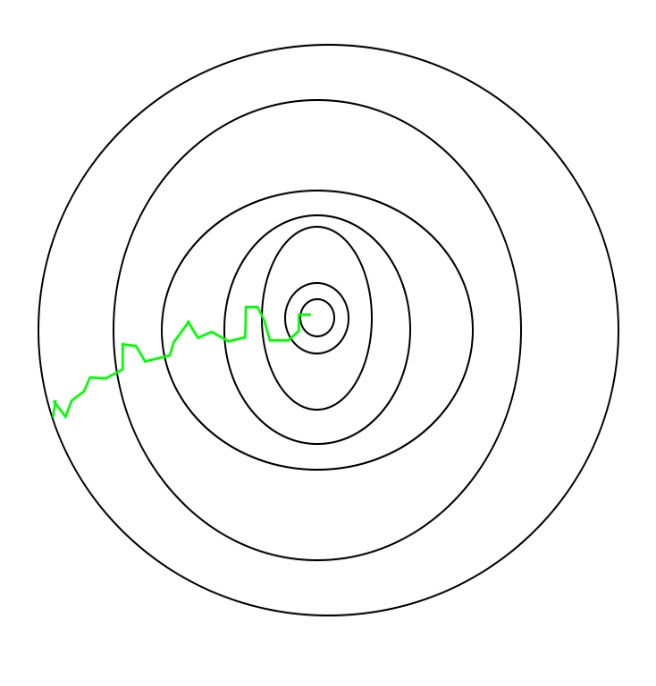
**Stochastic Gradient Descent (SGD):**

The word ‘*stochastic*‘ means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called “batch” which denotes the total number of samples from a dataset that is used for calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy or less random manner, but the problem arises when our datasets get really huge.

Suppose, you have a million samples in your dataset, so if you use a typical Gradient Descent optimization technique, you will have to use all of the one million samples for completing one iteration while performing the Gradient Descent, and it has to be done for every iteration until the minima is reached. Hence, it becomes computationally very expensive to perform.

In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn’t matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with significantly shorter training time.

**Path taken by Batch Gradient Descent Path taken by Stochastic Gradient Descent**



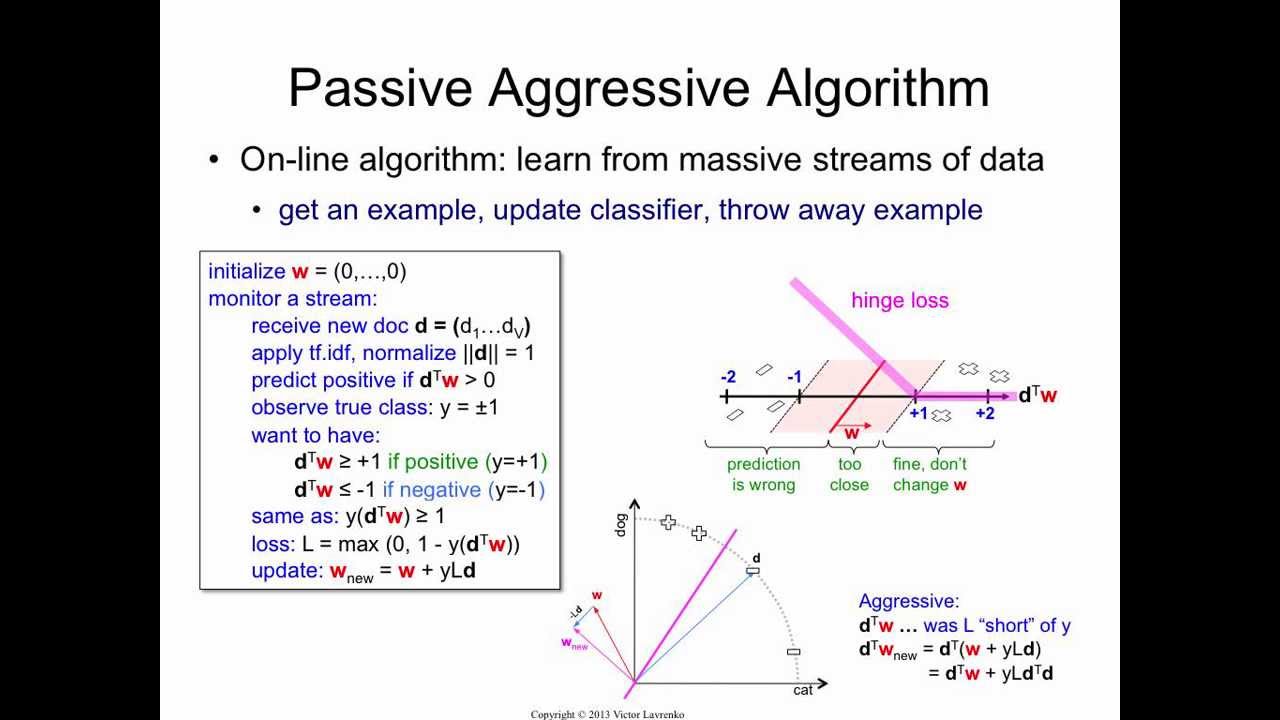
One thing to be noted is that, as SGD is generally noisier than typical Gradient Descent, it usually took a higher number of iterations to reach the minima, because of its randomness in its descent. Even though it requires a higher number of iterations to reach the minima than typical Gradient Descent, it is still computationally much less expensive than typical Gradient Descent. Hence, in most scenarios, SGD is preferred over Batch Gradient Descent for optimizing a learning algorithm.

**Naive Bayes classifier for multinomial models**

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

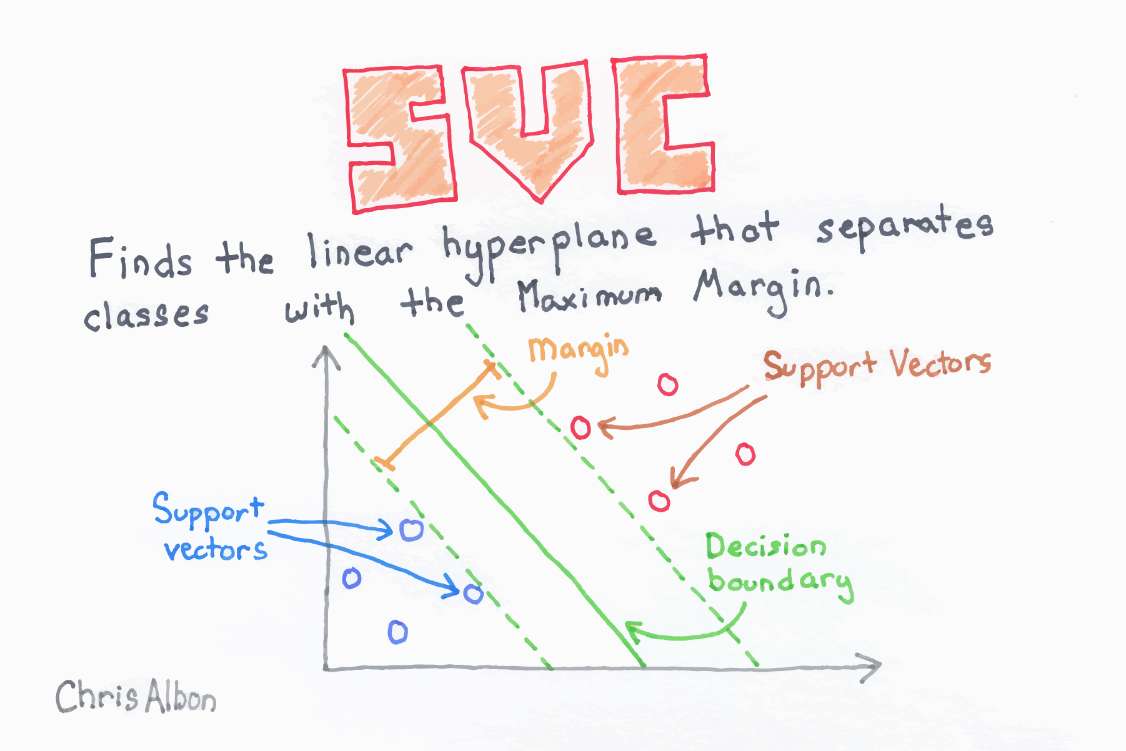
**PassiveAggressiveClassifier**

The passive-aggressive algorithms are a family of algorithms for large-scale learning. They are similar to the Perceptron in that they do not require a learning rate. However, contrary to the Perceptron, they include a regularization parameter C. (Passive: if correct classification, keep the model; Aggressive: if incorrect classification, update to adjust to this misclassified example.)



**Support Vector Classifier**

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our uses, though you can use this for many situations.

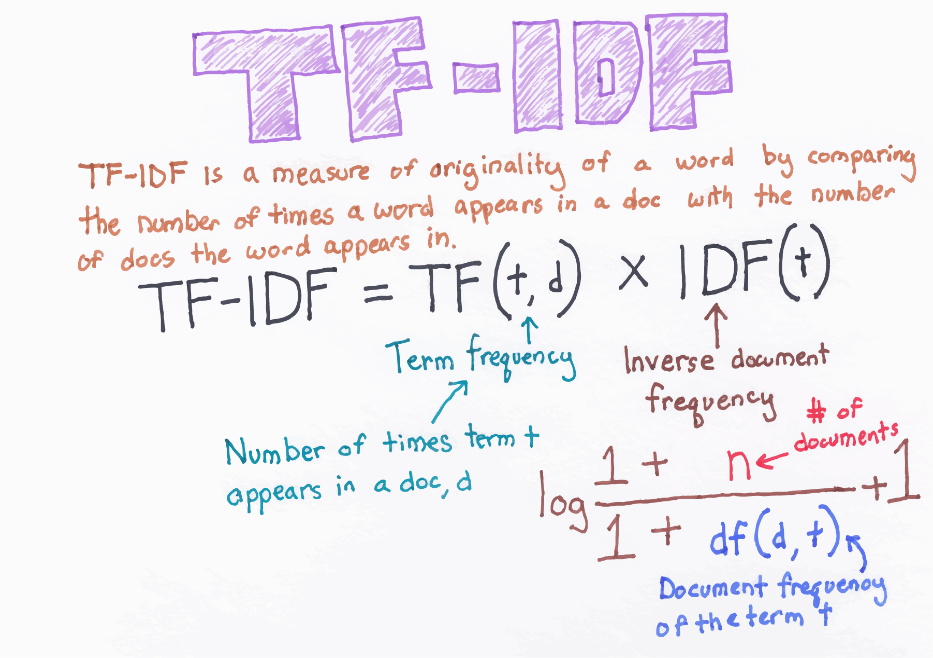


**Feature Generation**

For generation of features from the given data, we first performed tokenization on the raw text of articles. We then generated tf-idf feature vectors as described below

**Term Frequency - Inverse Document Frequency**

The tf-idf is a statistical measure that reflects the importance of a particular word with respect to a document in a corpus. It is often used in information retrieval and text mining as one of the components for scoring documents and performing searches. It is a weighted measure of how often a word occurs in a document relative to how often it occurs across all documents in the corpus. Term frequency is the number of times a term occurs in a document. Inverse document frequency is the inverse function of the number of documents in which it occurs.



Hence a term like “the” that is common across a collection will have lesser tf-idf values, as its weight is diminished by the idf component. Hence the weight computed by tf-idf represents the importance of a term inside a document. The tokenized data was used to generate a sparse matrix of tf-idf features for representation. This represented our feature vector and was used in subsequent prediction algorithms.

**Evaluation Metrics**

We used the following three metrics for the evaluation of our results. The use of more than one matrix helped us evaluate the performance of the models from different perspectives

**Classification Accuracy**

This depicts the number of accurate predictions made out of the total number of predictions made. Classification accuracy is calculated by dividing the total number of correct result by the total number of test data records and multiplying by 100 to get the percentag

**Confusion Matrix**

This is a great visual way to depict the predictions as four categories:

1. False Positive: Predicted as fake news but are actually true news.

2. False Negative: Predicted as true news but are actually fake news.

3. True Positive: Predicted as fake news and are actually fake news.

4. True Negative: Predicted as true news and are actually true news.

**Precision and Recall**

Precision which is also known as the positive predictive value is the ratio of relevant instances to the retrieved instances.

Precision = No. of True Positives / (No. of True Positives + No. of False Positives)

Recall which is also known as sensitivity is the proportion of relevant instances retrieved among the total number of relevant instances.

Recall = No. of True Positives / (No. of True Positives + No. of False Negatives)

**Conclusion**

Naïve Bayes performed very good in Bag of words model. Passive aggressive classfier with tf-idf vector as a feature improved this result substantially. Naïve bayes surprisingly performed very well, as observed from the above results support vector classfier performed slightly better than Support vector machine . The result can further be improved if the n-grams are used to generate tf-idf vectors and then used as a feature and due data avilable very less in hindi format.So we can further improve our model by adding it for hindi fake news detection and adding the deep learning in it can we can further exapand it that it train it self by own after evey search done on it .

**REFERENCES :**

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html>

<https://scikit-learn.org/stable/modules/naive_bayes.html>

<https://scikit-learn.org/stable/modules/svm.html>

<https://medium.com/@awantikdas/a-comprehensive-naive-bayes-tutorial-using-scikit-learn-f6b71ae84431>

<https://pythonprogramming.net/linear-svc-example-scikit-learn-svm-python/>

<https://medium.com/nlpgurukool/tfidf-vectorizer-5421f1528402>

<https://web.stanford.edu/class/archive/cs/cs109/cs109.1176/lectures/22-NaiveBayes.pdf>

En.wikipedia.org. Support vector machine. <https://en.wikipedia.org/wiki/Support_vector_machine>

Lecture Notes (Naïve Bayes) of CS 5100: Fundamentals of Artificial Intelligence, CCIS, Northeastern University by Prof. David Smith.

[Daniel Jurafsky, Jones H. Martin] Naïve Bayes and Sentiment Classification, Speech and Language Processing, Chapter 6. <https://web.stanford.edu/~jurafsky/slp3/6.pd>

<https://getbootstrap.com/docs/3.3/components/#glyphicons>

<https://www.w3schools.com/>