MACHINE LEARNING LAB PROJECT

Fake News Detection: A Machine Learning Approach

154446* – Fopa Yuffon Amadou Olabi[1], 160041082 – Mohamed Moctar[2], 160041083 – Tani Barakat Shalanyuy[3], 160041085 – Mikayilou Namba[4], 160041087 – Aly Abdel Kader Gelany[5], and 160041006 – Muhammed Barry[6]

[1,2,3,4,5,6] Department of Computer Science and Engineering, Islamic University of Technology

16th September 2020.

Abstract— Fake News and Scams started since the web time frame. The fake news design started basically to dupe per-clients, increase readership and is oftentimes used as a strategies for mental battling. Advances in development and the spread of news through different sorts of media, without truly checking the real factors, have extended the spread of fake news today. The essential purpose behind this endeavor is to devised a classifier which can isolate fake news from the certifiable news.
We propose in this research project, a fake news detection model that uses a Text Vectorizer and machine learning techniques to tackle the problem. Experimental evaluation yields the best performance using Term Frequency-Inverted Document Frequency (TF-IDF) as feature extraction technique, and Passive Aggressive Classifier as a classifier, with an accuracy of more than **97%**.

Keywords— Fake News, Real News, Machine Learning Techniques, Text classification, TF-IDF Vectorizer, Term Frequency (TF), Inverse Document Frequency (IDF), Passive Aggressive Classifier (PAC), Python.

I Introduction

The effects of fake news have increased exponentially in the recent past and something must be done to prevent this from continuing in the future. The dangerous effects of fake news, as previously defined, are made clear by events such as in which a man attacked a pizzeria due to a widespread fake news article. This story along with analysis provide evidence that humans are not very good at detecting fake news, possibly not better than chance. As such, the question remains whether machines can do a better job. A machine can solve the fake news problem using supervised learning that extracts features of the language and content only within the source in question, without utilizing any fact checker or knowledge base.
Do you trust all the news you hear from social media? All news is not real, right? So how will you detect the fake news? The answer is Python.
By practicing such an advanced python project of detecting fake news, we will easily make a difference between real and fake news. Before moving ahead in this advanced Python project, we have to be aware of related terms of fake news like the TF-IDF Vectorizer, and the Passive Aggressive Classifier.
This project will work you through the necessary steps and techniques used to implement such an analysis.

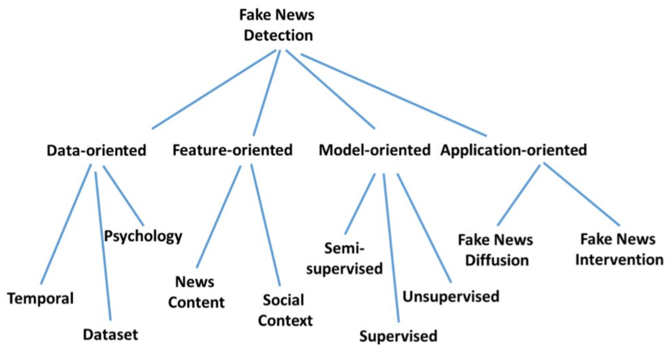


Fig.??: Future directions and open issues for fake news detection on social media

I.1 Problem Statement

This project proposes the question of whether it is possible to detect fake news through machine learning models. Specifically, the aim of this project is to determine the ideal model that is efficient in predicting fake news while also limiting the cost of memory and storage for computation. "Fake news" has been a very recent and prevalent problem within recent years.

I.2 Detail of your problem application area/domain

Fake news spreads like a wildfire and this is a big issue in this era. We can learn how to distinguish fake news from a real one. We will be using supervised learning approach to implement the model.

As a consequence of the increase in cases of fake news in recent years, efforts have been made to crackdown on the spread of misinformation throughout social media platforms. All popular social media platforms (Facebook, Twitter, Spotify, and YouTube) have permanently banned Alex Jones from using their networks (?) following the events of ”Pizza Gate” (?) in addition to multiple questionable accusations made by Jones, including an accusation made by Jones claiming that the Sandy Hook shooting was ”faked”.

Despite efforts of many social media websites and governments cracking down on fake news, many young people today generally are not able to tell the difference between fake news and real news. According to a Stanford study it found that many students have a very strong inability in discerning between fake news. In the study, high school students were given two posts announcing the candidacy of Donald Trump’s presidential Campaign. One post was given by an actual Fox News account another one posted by an account that ”looked” like it was from Fox News. 25% of the could not tell the difference between real and fake news sources. With over 30% of students favoring that the fake news account was more trustworthy.

Indeed, some politically charged or bogus articles that would be esteemed false frequently have more perspectives and offers via web-based media destinations than real news stories towards the most recent three months of the political race. As per an examination by Buzz-channel, posts and stories composed from the best twenty most noteworthy performing trick locales and hyper-hardliners had over 8.7 million offers, ”responses” and remarks contrasted with the main twenty most noteworthy performing significant news associations had about 7.4 million offers, ”responses” and remarks via online media destinations.

The research problem was initially defined through the following use cases: In light of a single event/story, the framework would decide whether certain sources or articles are regarded to be fake news dependent on a given likelihood. Through the sources analyzed the machine learning agent would assign an level of bias and factuality of these articles by comparing them to each other and assign scores of the sources bias and factuality.

A sort of sensationalist reporting, counterfeit news exemplifies bits of news that might be scams and is commonly spread through web-based media and other online media. This is regularly done to further or force certain thoughts and is frequently accomplished with political plans. Such news things may contain bogus and additionally overstated cases and may wind up being viral by calculations, and clients may wind up in a channel bubble.

I.3 Challenges and Motivation

Motivation— The motivation for research on this topic was that this is a relatively new area of research with many opinions but not many concrete solutions. Many implementations focus primarily on the host of the article, but even articles hosted on otherwise trustworthy websites can be classified as fake news. The primary motivation of this project was to bring awareness,propose a solution,and work towards minimizing the effects of fake news.

Challenges— Throughout the project and its analysis study, we faced some challenges such as:

- **Data Collection**: While collecting the data, we faced an issue with the relation of topics among news feeds. This is due to the variety of categorized topics covered by most informative platforms.
- **Data Analysis and Interpretation**: After collecting these data, we spend a lot of time in analyzing from a feature-based perspective.
- **Data Preprocessing Methodology**: After proper analysis and interpretation, we need to prepare and sanitize the data for it to be suitable for more insights on the learning process.
- **Learning Model Selection** (Which model suits well the given problem.)
- **Update the Model for better Accuracy**: Here we try to tune the parameters/hyperparameter to obtain a better accuracy from the selected model.

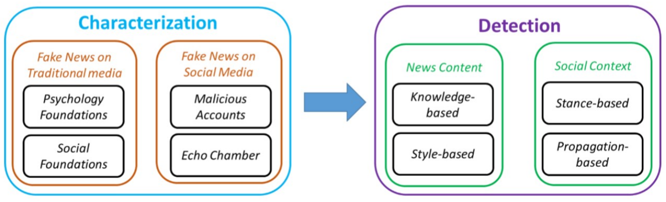


Fig.??: Fake news on social media: from characterization to detection

I.4 Objectives

The purpose of this paper is to design and implement a machine learning implementation that correctly predicts if a given article would be considered as fake news. The contributions of this paper are as follows

- Introduces the topic of fake news and the various machine learning algorithms to build a model to accurately classify a piece of news as REAL or FAKE.
- Provides an overview of the history and implications of fake news.
- This advanced python project of detecting fake news deals with fake and real news. Using SK-Learn tools, we will a **TfidfVectorizer** on our dataset.

- Then, we initialize a **PassiveAggressive** Classifier to fit the model, which will result a accuracy score and a confusion matrix that tell us how well our model fares.
- Presents a possible solution and lays some ground work in further study in this area.

I.5 Contribution

The project's contributions are split between five members. *Tani Barakat*[3] and *Muhammed Barry*[6] helped visualize the models' results through the Sci-Kit and Sea-born libraries. *Mohamed Moctar*[2] helped with research of the other related works done by external organizations. *Aly Abdel Kader G.*[5] and *Mikayilou Namba*[4] implemented the machine learning models of the project. *F. Y. Amadou Olabi*[1] served as the project lead in conceptualizing the idea and technical writing for the group. And also researched, cleansed and formalized the dataset used for the machine learning techniques and implemented the models.

II Background Study

Several groups and organization have also worked on similar ideas in their own implementations. These works highlight some of the challenges of fake news detection. One implementation by *Katharine Jarmul*, founder of data analysis company *Kjamistan*, uses a Passive Aggressive Classifier to detect fake news (**?**).The implementation is a tutorial on using different Bayesian models posted on DataCamp(**?**), which offers courses on a variety of data science topics including R and Python.

One paper titled *'Exploiting Network Structure to Detect Fake News'*, written by three Stanford University students, also implements a Neural Network for classifying fake news(**?**). Their implementation also takes into account the social context in addition to article-specific features such as the title and content in an article in an attempt to improve prediction accuracy. This is one of the few possible ways to improve prediction accuracy without improving upon the natural language processing.

Another paper titled *'Fake News Detection: A Deep Learning Approach'*, implemented three different neural network models to compare with the only difference between them being how they took in the article content and title(**?**). This indicates that the way one goes about processing text in an article makes a huge difference in the performance of a model. This makes sense considering that an article's content is generally the only thing that can be analyzed to truly determine its authenticity.

Finally, another paper entitled *'Online Passive-Aggressive Active learning'*, implemented a Passive-Aggressive Active (PAA) learning algorithms by adapting the Passive-Aggressive algorithms in online active learning settings**?**. Unlike conventional Perceptron-based approaches that employ only the misclassified instances for updating the model, this proposed PAA learning algorithms not only use the misclassified instances to update the classifier, but also exploit correctly classified examples with low prediction confidence.
Specifically, they propose several variants of PAA algorithms to tackle three types of online learning tasks: binary classification, multi-class classification, and cost-sensitive classification.

II.1 Overview of the project and related terms

This project is about determining if a given news is *Fake* or *Real* through a set of mechanism illustrated in figure **??**

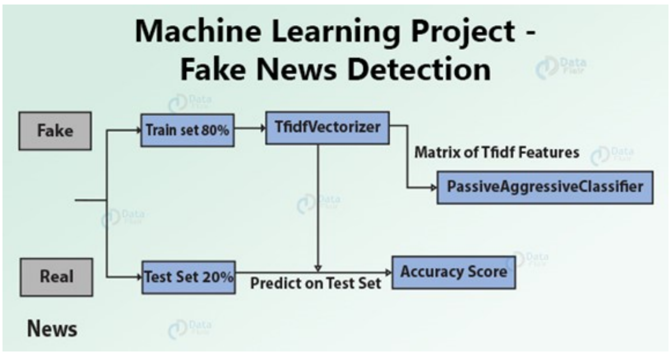


Fig.??: Project Overview

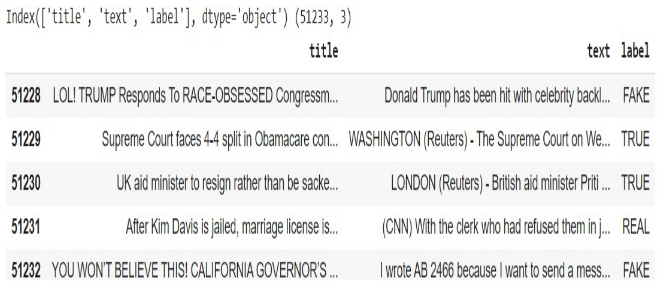
II.2 Data collection and feature analysis

Data Collection— Guardian newspaper and Kaggle provide an API(Application Program Interface) which enables to populate the model with up-to date news. These samples data are shared among 03 files under the names: and .

- *'news.csv'* : which have a sample data of shape 6335 by 4 with 04 features ('Unnamed: 0', 'title', 'text', 'label') and contains a mixture of fake and real news.
- *'Fake.csv'* : which have a sample data of shape 23481 by 4 with 04 features ('title', 'text', 'subject', 'date') and contains only fake news.
- *'True.csv'* : which have a sample data of shape 21417 by 4 with 04 features ('title', 'text', 'subject', 'date') and contains only real news.

Feature Analysis— From the compiled samples data obtained above, we formed our experimental dataset based on 03 features which are *'title', 'text', 'label'* with a shape of 51233 by 3 and contains a mixture of fake and real news.

- **'title'** : The first column contains the title of the news.
- **'text'** : The second column contains the plain-text news.
- **'label'** : The third column has labels denoting whether the news is REAL or FAKE.



```
Index(['title', 'text', 'label'], dtype='object') (51233, 3)
```

	title	text	label
51228	LOL! TRUMP Responds To RACE-OBSESSED Congressm...	Donald Trump has been hit with celebrity backl...	FAKE
51229	Supreme Court faces 4-4 split in Obamacare con...	WASHINGTON (Reuters) - The Supreme Court on We...	TRUE
51230	UK aid minister to resign rather than be sacke...	LONDON (Reuters) - British aid minister Priti ...	TRUE
51231	After Kim Davis is jailed, marriage license is...	(CNN) With the clerk who had refused them in j...	REAL
51232	YOU WON'T BELIEVE THIS! CALIFORNIA GOVERNOR'S ...	I wrote AB 2466 because I want to send a mess...	FAKE

Fig.**??**: Compiled Dataset

NB: These samples data were taken from Guardian newspaper and Kaggle website where it can be found under the title *Fake and Real news dataset***?**. Kaggle provide an API(Application Program Interface) which enables to populate the dataset with up-to date news. The training and testing data were collected using the API.
NB: This dataset is based on the topic of political news feeds in the USA and other countries with political instabilities.
In the next step, preprocessing of the dataset like removing stop words, punctuation marks, missing fields was done to sanitize the samples data.

II.3 Description of the Classification Model

Passive Aggressive algorithms are online learning algorithms. Such an algorithm remains passive for a correct classification outcome, and turns aggressive in the event of a miscalculation, updating and adjusting. Unlike most other algorithms, it does not converge. Its purpose is to make updates that correct the loss, causing very little change in the norm of the weight vector.

II.3.1 Classification Model used

For this research project, we decided to use a **Passive Aggressive Classification**(PAC) methodology as model.
Passive Aggressive Algorithms are a family of online learning algorithms (for both classification and regression) proposed by Crammer at al. (**?**). The idea is very simple and their performance has been proofed to be superior to many other alternative methods.
The Passive-Aggressive algorithms are a family of Machine learning algorithms that are not very well known by beginners and even intermediate Machine Learning enthusiasts. However, they can be very useful and efficient for certain applications.**?**

Note: This is a high-level overview of the algorithm explaining how it works and when to use it. It does not go deep into the mathematics of how it works.

Conceptual Understanding of PAC— ? Passive-Aggressive algorithms are generally used for large-scale learning. In online machine learning algorithms, the input data comes in sequential order and the machine learning model is updated step-by-step, as opposed to batch learning, where the entire training dataset is used at once. This is very useful in situations where there is a huge amount of data and it is computationally infeasible to train the entire dataset because of the sheer size of the data. We can simply say that an online-learning algorithm will get a training example, update the classifier, and then throw away the example.
A very good example of this would be to detect fake news on a social media website like Twitter, where new data is being added every second. To dynamically read data from Twitter continuously, the data would be huge, and using an online-learning algorithm would be ideal.
Passive-Aggressive algorithms are somewhat similar to a Perceptron model, in the sense that they do not require a learning rate. However, they do include a regularization parameter.
How Passive-Aggressive Algorithms Work:
Passive-Aggressive algorithms are called so because :

- **Passive**: If the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model.
- **Aggressive**: If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.

Mathematical Approach of PAC— (?) (?)
Let's suppose to have a dataset:

$$\begin{cases} X = \{\overline{x}_0, \overline{x}_1, \ldots, \overline{x}_t, \ldots\} where \quad \overline{x}_i \epsilon \mathbb{R}^n \\ Y = \{y_0, y_1, \ldots, y_t, \ldots\} where \quad y_i \epsilon \{-1, +1\} \end{cases} \tag{1}$$

The index t has been chosen to mark the temporal dimension. In this case, in fact, the samples can continue arriving for an indefinite time. Of course, if they are drawn from same data generating distribution, the algorithm will keep learning (probably without large parameter modifications), but if they are drawn from a completely different distribution, the weights will slowly forget the previous one and learn the new distribution. For simplicity, we also assume we're working with a binary classification based on bipolar labels.
Given a weight vector w, the prediction is simply obtained as:

$$\widetilde{y}_t = sign(\overline{w}^T * \overline{x}_t) \tag{2}$$

All these algorithms are based on the Hinge loss function (the same used by SVM):

$$L(\overline{\theta}) = \max(0, 1 - y * f(\overline{x}_i; \overline{\theta})) \tag{3}$$

The value of L is bounded between 0 (meaning perfect match) and K depending on $f(x(t), \theta)$ with $K > 0$ (completely wrong prediction). A Passive-Aggressive algorithm works generically with this update rule:

$$\begin{cases} \overline{w}_{t+1} = \arg\min_{\overline{w}} \frac{1}{2} \left\| \overline{w} - \overline{w}_t \right\|^2 + C\xi^2 \\ L(\overline{w}; \overline{x}_t, y_t) \leq \xi \end{cases} \tag{4}$$

To understand this rule, let's assume the slack variable $\xi = 0$ (and L constrained to be 0). If a sample $x(t)$ is presented, the classifier uses the current weight vector to determine the sign. If the sign is correct, the loss function is 0 and the $\arg\min$ is $w(t)$. This means that the algorithm is passive when a correct classification occurs. Let's now assume that a misclassification occurred as show in figure **??**:

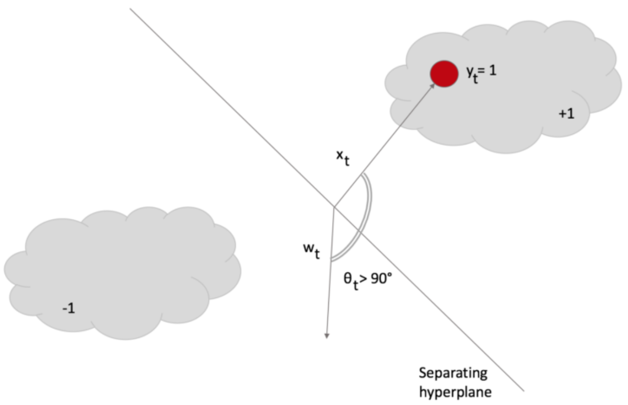


Fig.??

The angle $\theta > 90°$, therefore, the dot product is negative and the sample is classified as -1, however, its label is $+1$. In this case, the update rule becomes very aggressive, because it looks for a new w which must be as close as possible as the previous (otherwise the existing knowledge is immediately lost), but it must satisfy $L = 0$ (in other words, the classification must be correct).
The introduction of the slack variable allows to have soft-margins (like in SVM) and a degree of tolerance controlled by the parameter C. In particular, the loss function has to be $L \leq \xi$, allowing a larger error. Higher C values yield stronger aggressiveness (with a consequent higher risk of destabilization in presence of noise), while lower values allow a better adaptation. In fact, this kind of algorithms, when working online, must cope with the presence of noisy samples (with wrong labels). A good robustness is necessary, otherwise, too rapid changes produce consequent higher misclassification rates.
After solving both update conditions, we get the closed-form update rule:

$$\overline{w}_{t+1} = \overline{w}_t + \frac{\max(0, 1 - y_t(\overline{w}^T * \overline{x}_t))}{\|x_t\|^2 + \frac{1}{2C}} y_t \overline{x}_t \qquad (5)$$

This rule confirms our expectations: the weight vector is updated with a factor whose sign is determined by $y(t)$ and whose magnitude is proportional to the error. Note that if there's no misclassification the nominator becomes 0, so $w(t+1) = w(t)$, while, in case of misclassification, w will rotate towards $x(t)$ and stops with a loss $L \leq \xi$.
In the next figure **??**, the effect has been marked to show the rotation, however, it's normally as smallest as possible:

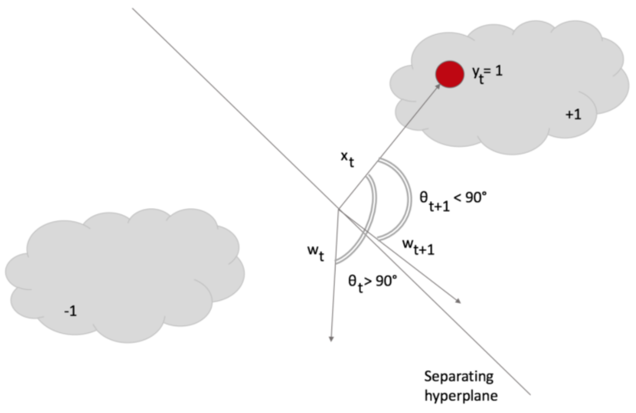
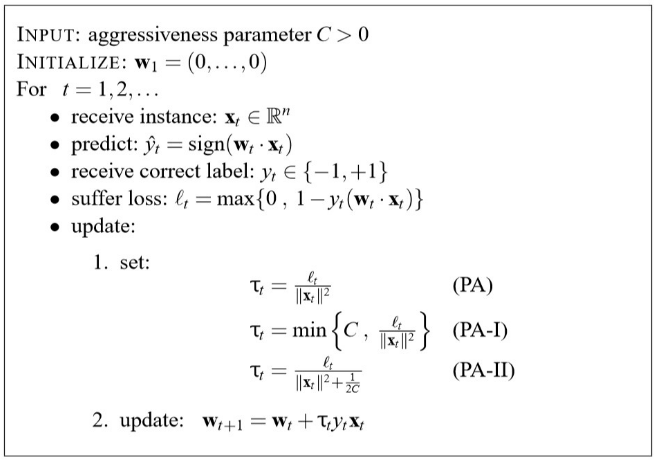


Fig.??

After the rotation, $\theta < 90°$ and the dot product becomes negative, so the sample is correctly classified as $+1$.

II.3.2 Architectural Diagram of the Classifier



```
INPUT: aggressiveness parameter $C > 0$
INITIALIZE: $\mathbf{w}_1 = (0, \dots, 0)$
For  $t = 1, 2, \dots$
    • receive instance: $\mathbf{x}_t \in \mathbb{R}^n$
    • predict: $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$
    • receive correct label: $y_t \in \{-1, +1\}$
    • suffer loss: $\ell_t = \max\{0\ ,\ 1 - y_t(\mathbf{w}_t \cdot \mathbf{x}_t)\}$
    • update:
        1. set:
                $\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2}$                              (PA)
                $\tau_t = \min\left\{C\ ,\ \frac{\ell_t}{\|\mathbf{x}_t\|^2}\right\}$        (PA-I)
                $\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}}$                     (PA-II)
        2. update:  $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$
```

Fig.??: Algorithm of Passive-Aggressive with its 03 variants(PA, PA-I, PA-II) used for binary classification.

II.3.3 Explanation of the Parameters —— Hyper-parameters

Understanding the science behind this calculation isn't exceptionally basic and is past the extent of this research project. This research project gives only a diagram of the calculation and a basic usage of it.

Important parameters:

- **C** : This is the regularization parameter, and denotes the penalization the model will make on an incorrect prediction

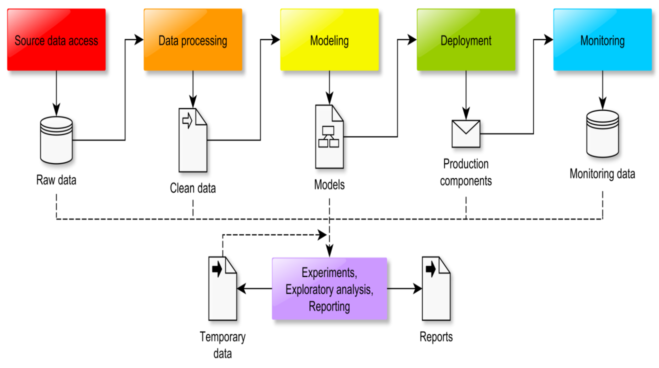
- **max_iter** : The maximum number of iterations the model makes over the training data.

- **tol** : The stopping criterion. If it is set to None, the model will stop when (loss > previous_loss – tol). By default, it is set to 1e-3.

III Implementation

Fake News Detection — is based on a Passive Aggressive Classification Algorithm. It was developed in Python using Google Colab Online IDE. The full implemented source code can be found under the GitHub Repository named *ML-Project* ?

III.1 Overview of the experiment



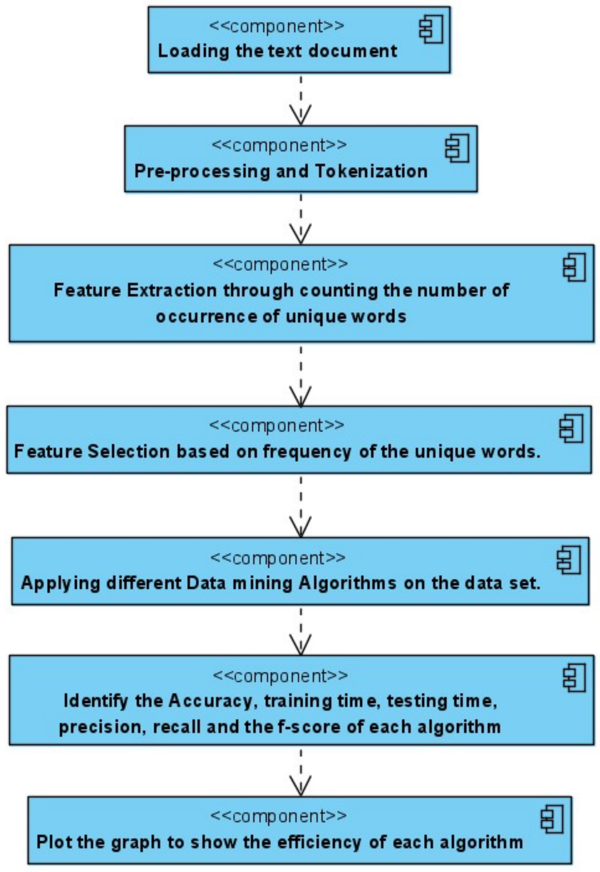

<<component>>
Loading the text document
<<component>>
Pre-processing and Tokenization
<<component>>
Feature Extraction through counting the number of occurrence of unique words
<<component>>
Feature Selection based on frequency of the unique words.
<<component>>
Applying different Data mining Algorithms on the data set.
<<component>>
Identify the Accuracy, training time, testing time, precision, recall and the f-score of each algorithm
<<component>>
Plot the graph to show the efficiency of each algorithm

Fig.??: Workflow diagram of the framework proposed.

Project Workflow Steps——

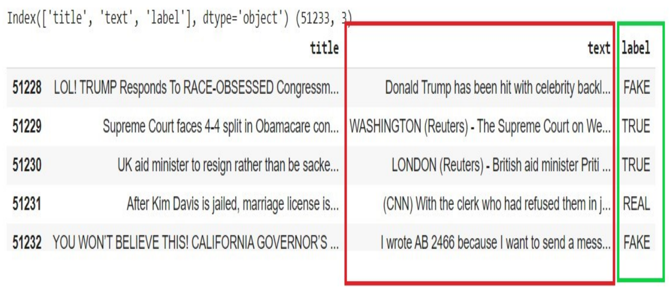
- Collecting Sample data related to Fake News Analysis.
- Building of a dataset from a set of known and well-done datasets.
- Loading and Analyzing dataset.
- Splitting the dataset into training and testing sets.
- Preprocessing of the dataset.
- Choose a Learning Model, Methodology or Schema for training the dataset.
- Fitting the Model with proper parameters and Predicting a feasible outcome(likelihood).
- Determining the Model Accuracy Score.
- Report and Visualization of the predicted outcomes.
- If the results are not that convincing, then Tuning and Optimizing Model with necessary algorithms, is needed.
- Testing the Optimized Model, and Reporting its whereabouts and results.
- After Previous.Step, if the obtained results are not still that convincing then ”Repeat Previous→Previous.Step” with a more efficient technique.
- Summary Report on the Model.

III.2 Feature engineering

Feature engineering—— is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithms. Feature engineering can be considered as applied machine learning itself.

Feature Extraction—— It uses data reduction which allows the elimination of less important features. The collection of words obtained through tokenization is sorted to find the unique words. The unique words and the count will be stored separately for further processing.
For this research project, we selected the features *’text’* and *’label’* from our compiled dataset shown in figure **??** below.

Feature Selection—— We used the feature *’text’* as **input data** or data to be analyzed and the feature *’label’* as **target data** or result/output from the input data.



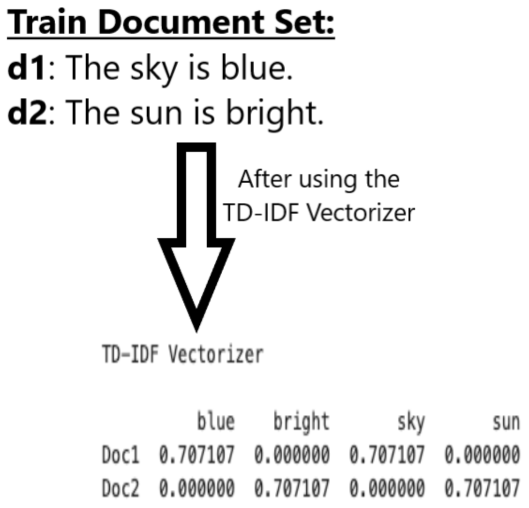
```
Index(['title', 'text', 'label'], dtype='object') (51233, 3)
```

	title	text	label
51228	LOL! TRUMP Responds To RACE-OBSESSED Congressm...	Donald Trump has been hit with celebrity backl...	FAKE
51229	Supreme Court faces 4-4 split in Obamacare con...	WASHINGTON (Reuters) - The Supreme Court on We...	TRUE
51230	UK aid minister to resign rather than be sacke...	LONDON (Reuters) - British aid minister Priti ...	TRUE
51231	After Kim Davis is jailed, marriage license is...	(CNN) With the clerk who had refused them in j...	REAL
51232	YOU WON'T BELIEVE THIS! CALIFORNIA GOVERNOR'S ...	I wrote AB 2466 because I want to send a mess...	FAKE

Fig.??: Compiled Dataset with Selected features.

Data Preprocessing—— To preprocess the data, we use a **TF-IDF Vectorizer** to transform the text to feature vectors that can be used as input to an estimator.

What is a TF-IDF Vectorizer?
TF-IDF is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. Let's take sample example and explore two different spicy sparse matrix before go into deep explanation.


Train Document Set:
d1: The sky is blue.
d2: The sun is bright.

```
TD-IDF Vectorizer

          blue    bright       sky       sun
Doc1  0.707107  0.000000  0.707107  0.000000
Doc2  0.000000  0.707107  0.000000  0.707107
```

Fig.**??**: Text Document converted to Spicy Sparse Matrix of TF-IDF Vectorizer

Here, we can see clearly that TF-IDF Vectorizer consider overall documents of weight of words. A vocabulary is a dictionary that converts each token (word) to a feature index in the matrix, each unique token gets a feature index. The TF-IDF Vectorizer converts a collection of raw documents into a matrix of TF-IDF features.

- **TF (Term Frequency)**: The number of times a word appears in a document is its Term Frequency. A higher value means a term appears more often than others, and so, the document is a good match when the term is part of the search terms.
- **IDF (Inverse Document Frequency)**: Words that occur many times a document, but also occur many times in many others, maybe irrelevant. IDF is a measure of how significant a term is in the entire corpus.

Mathematical Understanding of TF-IDF———
TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a document with the number of documents the word appears in.

TF-IDF $= TF(t,d) * IDF(t)$—— where $TF(t,d)$ is the number of times term t appears in a document d as shown:

$$TF(t,d) = \sum_{x \epsilon d} fr(x,t) \tag{6}$$

where $fr(x,t)$ is a simple function defined as:

$$fr(x,t) = \begin{cases} 1, & if \rightarrow x = t \\ 0, & otherwise \end{cases} \tag{7}$$

and **IDF**(t) is the Inverse Document Frequency which is computed as follow:

$$IDF(t) = \log(\frac{1+n}{1+DF(d,t)}) + 1 \tag{8}$$

where n is the number of documents and **DF**(d,t) is the Document Frequency of the term t.

Remark——— In **TfidfVectorizer** we consider **overall document weightage** of a word. It helps us in dealing with most frequent words. Using it we can penalize them. TfidfVectorizer weights the word counts by a measure of how often they appear in the documents.

III.3 Training and Test Set Generation

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.
It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.**?**

Remark——— The train-test split procedure is appropriate when you have a very large dataset, a costly model to train, or require a good estimate of model performance quickly.

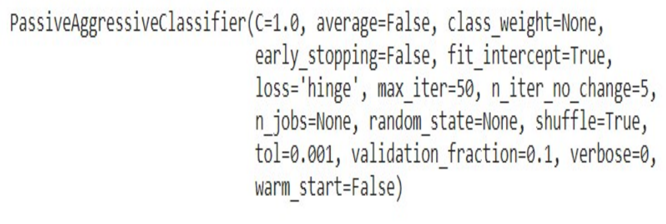
In this research project, we used a split percentage of Train: 80%, Test: 20% which is also called a **8:2 split ratio** on the compiled dataset (figure **??**) with shape of 51233 by 3. We split the dataset with the selected features shown in figure **??**.
After splitting the dataset, we got:

- **X-Train**: with length {40986} which represents 80% of the dataset.
- **X-Test**: with length {10247} which represents 20% of the dataset.
- **Y-Train** and **Y-Test**: which are the target data for **X-Train** and **X-Test** respectively with the same split ratio.

III.4 Running the Classifier

Pre-Execution——— Before executing the classifier, we use **TF-IDF Vectorizer** to sanitize, transform and preprocess both **X-Train** and **X-Test** as mentioned in the above sections. Thus, to obtain both **TF_IDF-Train** and **TF_IDF-Test** respectively (with more features) which are **vectorized versions** of X-Train and X-Test.

Execution—— We use the obtained vectorized versions **TF_IDF-Train** and **TF_IDF-Test** to train and test the Passive Aggressive Classifier with their respective target samples. While setting the PAC model, we configured some of its parameters with some values as shown below:



```
PassiveAggressiveClassifier(C=1.0, average=False, class_weight=None,
                            early_stopping=False, fit_intercept=True,
                            loss='hinge', max_iter=50, n_iter_no_change=5,
                            n_jobs=None, random_state=None, shuffle=True,
                            tol=0.001, validation_fraction=0.1, verbose=0,
                            warm_start=False)
```

IV Result Analysis

To evaluate the performance of algorithms for fake news detection problem, various evaluation metrics have been used. In this subsection, we review the most widely used metrics for fake news detection. Most existing approaches consider the fake news problem as a classification problem that predicts whether a news article is fake or not:

- True Positive (**TP**): when predicted fake news pieces are actually annotated as fake news;
- True Negative (**TN**): when predicted true news pieces are actually annotated as true news;
- False Negative (**FN**): when predicted true news pieces are actually annotated as fake news;
- False Positive (**FP**): when predicted fake news pieces are actually annotated as true news.

By formulating this as a classification problem, we can define following metrics,

$$Precision = \frac{|TP|}{|TP| + |FP|} \tag{9}$$

$$Recall = \frac{|TP|}{|TP| + |FN|} \tag{10}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{11}$$

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{12}$$

These metrics are commonly used in the machine learning community and enable us to evaluate the performance of a classifier from different perspectives. Specifically, accuracy measures the similarity between predicted fake news and real fake news. Precision measures the fraction of all detected fake news that are annotated as fake news, addressing the important problem of identifying which news is fake. However, because fake news datasets are often skewed, a high precision can be easily achieved by making fewer positive predictions. Thus, recall is used to measure the sensitivity, or the fraction of annotated fake news articles that are predicted to be fake news. F1 is used to combine precision and recall, which can provide an overall prediction performance for fake news detection. Note that for Precision, Recall, F1, and Accuracy, the higher the value, the better the performance.

The Receiver Operating Characteristics (**ROC**) curve provides a way of comparing the performance of classifiers by looking at the trade-off in the False Positive Rate (**FPR**) and the True Positive Rate (**TPR**). To draw the ROC curve, we plot the FPR on the x-axis and and TPR along the y-axis. The ROC curve compares the performance of different classifiers by changing class distributions via a threshold. TPR and FPR are defined as follows (note that TPR is the same as recall defined above):

$$TPR = \frac{|TP|}{|TP| + |FN|} \tag{13}$$

$$FPR = \frac{|FP|}{|FP| + |TN|} \tag{14}$$

Based on the ROC curve, we can compute the Area Under the Curve (AUC) value, which measures the overall performance of how likely the classifier is to rank the fake news higher than any true news. Based on **?**, AUC is defined as below.

$$AUC = \frac{\sum(n_0 + n_1 + 1 + r_i) - n_0(n_0 + 1)/2}{n_0 * n_1} \tag{15}$$

where r_i is the rank of i^{th} fake news piece and $n_0(n_1)$ is the number of fake (true) news pieces. It is worth mentioning that AUC is more statistically consistent and more discriminating than accuracy **?**, and it is usually applied in an imbalanced classification problem, such as fake news classification, where the number of ground truth fake news articles and and true news articles have a very imbalanced distribution.

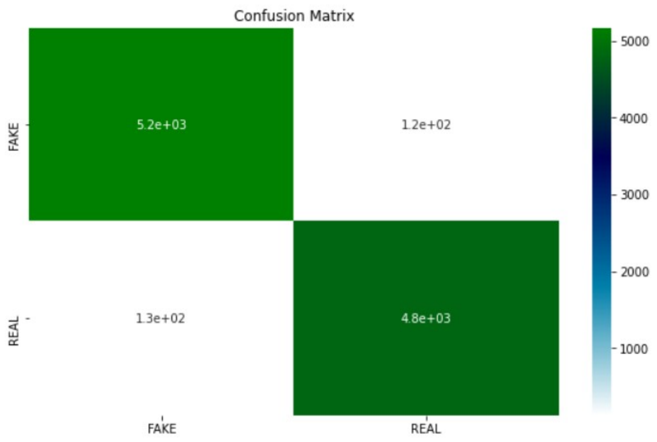
IV.1 Overview

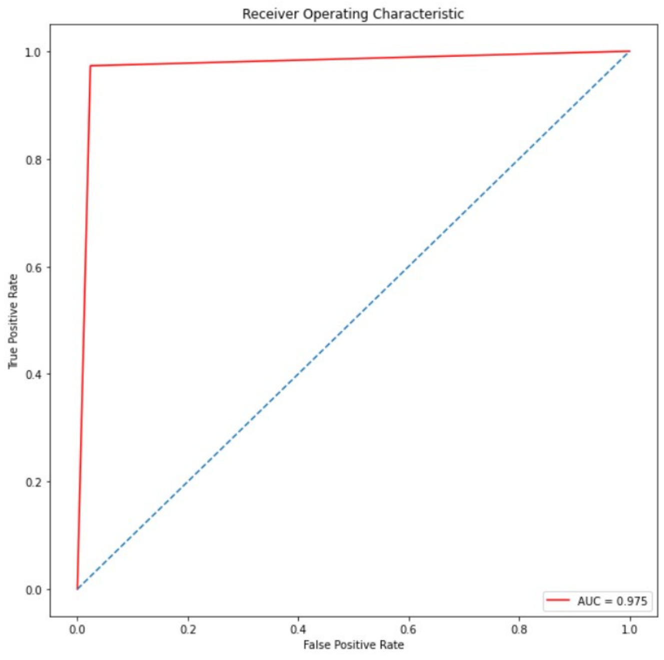
In training, the model had an accuracy score of more than 97%. The model was able to predict whether an article was fake news 70% of the time using the Fake News Net, Politifact and Kaggle dataset. This dataset was chosen because it was formatted differently than the Fake News Net dataset. A benefit of the Politifact portion of the dataset is that it had an almost equal amount of real vs vake news samples, This ensured that the model was not biased towards real or fake in testing.

IV.2 Generation of Confusion-Matrix & Heat-Map

After training and testing the model, a confusion-matrix was generated and visualized as shown below:

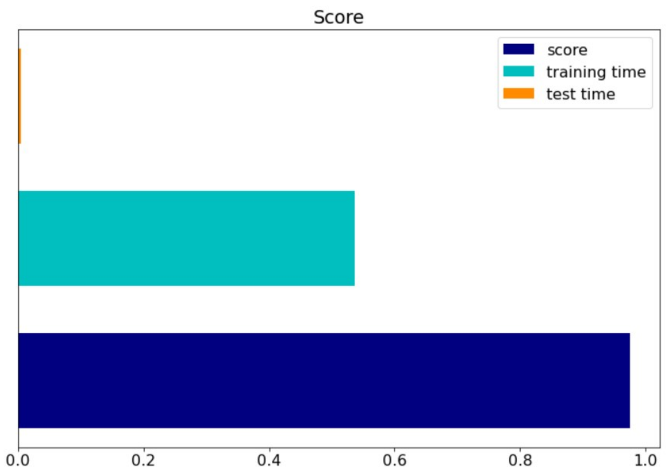
$$Confusion\_Matrix = \begin{bmatrix} 5168 & 125 \\ 134 & 4820 \end{bmatrix} \tag{16}$$

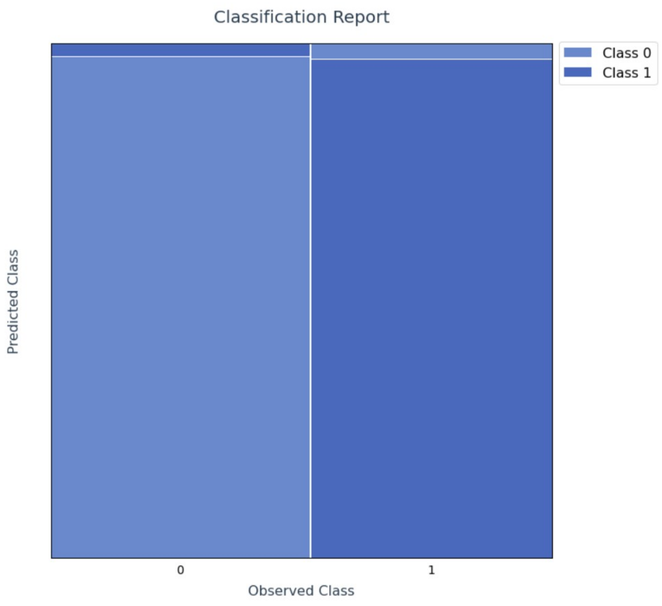




IV.3 Result analysis

```
-----------------=====Model Report=====----------------

              precision    recall  f1-score   support

        FAKE       0.97      0.98      0.98      5293
        REAL       0.97      0.97      0.97      4954

    accuracy                           0.97     10247
   macro avg       0.97      0.97      0.97     10247
weighted avg       0.97      0.97      0.97     10247

-------------------------------------------------------

::Confusion-Matrix::
 [[5168  125]
 [ 134 4820]]

-------------------------------------------------------

Precision:      0.97

Recall:         0.97

Density:        0.605463

Dimensionality: 129377

-------------------------------------------------------

Train Time:     0.536s

Test Time:      0.005s

Data-Preprocessing Time: 12.507s

-------------------------------------------------------

Accuracy Score: 97.47%

-------------------------------------------------------
```





V Conclusion

The problem of fake news has gained attention in 2016, especially in the aftermath of the last US presidential elections. Recent statistics and research show that 62% of US adults get news on social media **??**.
With the increasing popularity of social media, more and more people consume news from social media instead of traditional news media. However, social media has also been used to spread fake news, which has strong negative impacts on individual users and broader society.

It is certainly possible to classify news content into two types: fake news and real news, however there will always be an inherent bias to this classification based on the researcher's own personal beliefs. Even though this is true, with tools like this research project it could be possible to at least cut down on the amount of objectively fake news that exists in the world today. With a preliminary result of 70% this project could potentially contribute to accurately finding fake news and publicizing it, without the need for humans to have to do that work themselves.

The Efficiency can be improved using about five classifier models like Support Vector Machines, logistic Regression, Logistic Regression CV, which can perform better classification and can give a better accuracy. Using these classifiers, if the outputs are (REAL, REAL, FAKE, FAKE, REAL), then the output would be REAL as it is the majority. Apart from the classifier, we can also build a fact detector and a stance detector. Combination of all these tools would be the best way to classify the news accurately.
Fake news detection is an emerging research area with few public datasets. We run our model on an existing dataset, showing that our model outperforms the original approach published by the authors of the dataset.

In our future work, we will run our model on the few other publicly available datasets, such as the LIAR dataset which was released only recently, after we completed the current phase of our research.

Appendix A: Code snippets

In the new page, a snapshot of the entire source code of this research project can be found which is entitled ***ML_Project_Fake_News_Detector***. It contains a detail implementation of the proposed solution that tackles the stated problem illustrated in the above sections. It is written in iPython using Google Colab Online IDE and Jupyter Notebook IDE.

ML_Project_Fake_News_Detector

September 18, 2020

1 Initialization tweaks

- Initiating GoogleDrive mounting
- Necessary Library's imports
- Load Dataset [**Optional**]

[ ]:
```
#-Google Drive Access-Granting
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
# files paths
path = '/content/drive/My Drive/ThesisFile/datasets/ml-project/
 ↪fake_real-news-dataset/'
if path[-1] != '/':
  path += '/'
print('Good to go  ')
```

```
Good to go
```

[ ]:
```
#-Library imports
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import statistics
import math
import nltk
import string
from time import time
import seaborn as sns
from sklearn import preprocessing
from sklearn import linear_model
from sklearn import svm
from sklearn import metrics
from sklearn.utils import validation
```

```python
from sklearn.utils.extmath import density
from sklearn.cluster import KMeans
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression, PassiveAggressiveClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import normalize
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, pairwise_distances,␣
 ↪mean_absolute_error, classification_report, confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_blobs
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
from seaborn import heatmap
from math import exp
from random import randint, seed
import itertools
from statsmodels.graphics.mosaicplot import mosaic
from matplotlib.patches import Patch
from collections import deque

%matplotlib inline
```

```python
[ ]: def nclass_classification_mosaic_plot(n_classes, results):
         """
         build a mosaic plot from the results of a classification

         parameters:
         n_classes: number of classes
         results: results of the prediction in form of an array of arrays

         In case of 3 classes the prdiction could look like
         [[10, 2, 4],
          [1, 12, 3],
          [2, 2, 9]
         ]
         where there is one array for each class and each array holds the
         predictions for each class [class 1, class 2, class 3].
```

```
    This is just a prototype including colors for 6 classes.
    """
    class_lists = [range(n_classes)]*2
    mosaic_tuples = tuple(itertools.product(*class_lists))

    res_list = results[0]
    for i, l in enumerate(results):
        if i == 0:
            pass
        else:
            tmp = deque(l)
            tmp.rotate(-i)
            res_list.extend(tmp)
    data = {t:res_list[i] for i,t in enumerate(mosaic_tuples)}

    fig, ax = plt.subplots(figsize=(11, 10))
    plt.rcParams.update({'font.size': 16})

    font_color = '#2c3e50'
    pallet = [
        '#6a89cc',
        '#4a69bd',
        '#1e3799',
        '#0c2461',
        '#82ccdd',
        '#60a3bc',
    ]
    colors = deque(pallet[:n_classes])
    all_colors = []
    for i in range(n_classes):
        if i > 0:
            colors.rotate(-1)
        all_colors.extend(colors)

    props = {(str(a), str(b)):{'color':all_colors[i]} for i,(a, b) in␣
↪enumerate(mosaic_tuples)}

    labelizer = lambda k: ''

    p = mosaic(data, labelizer=labelizer, properties=props, ax=ax)

    title_font_dict = {
        'fontsize': 20,
        'color' : font_color,
    }
    axis_label_font_dict = {
        'fontsize': 16,
        'color' : font_color,
    }

    ax.tick_params(axis = "x", which = "both", bottom = False, top = False)
    ax.axes.yaxis.set_ticks([])
    ax.tick_params(axis='x', which='major', labelsize=14)

    ax.set_title('Classification Report', fontdict=title_font_dict, pad=25)
    ax.set_xlabel('Observed Class', fontdict=axis_label_font_dict, labelpad=10)
    ax.set_ylabel('Predicted Class', fontdict=axis_label_font_dict, labelpad=35)

    legend_elements = [Patch(facecolor=all_colors[i], label='Class {}'.
 ↪format(i)) for i in range(n_classes)]
    ax.legend(handles=legend_elements, bbox_to_anchor=(1,1.018), fontsize=16)

    plt.tight_layout()
    plt.show()
```

```
[ ]: # Loading Dataset
file0 = 'news.csv'
file1 = 'Fake.csv'
file2 = 'True.csv'

df0 = pd.read_csv(path + file0)
df1 = pd.read_csv(path + file1)
df2 = pd.read_csv(path + file2)

print(df0.columns, df0, sep='\n', end='\n\n')
print(df1.columns, df1, sep='\n', end='\n\n')
print(df2.columns, df2, sep='\n', end='\n\n')

df1['label'] = 'FAKE'
df2['label'] = 'REAL'

dataset = pd.concat([df0[['title','text','label']] ,␣
 ↪df1[['title','text','label']] , df2[['title','text','label']]],␣
 ↪ignore_index=True)

dataset = dataset.sample(frac=1).reset_index(drop=True)
print(dataset.columns, dataset.shape)
dataset.tail()

# print('Dataset Attributes::', dataset.columns)
# print('Dataset Dim::', dataset.shape)
# print(dataset.head())
# print('\nData Output::\n', dataset.label)
```

```
Index(['Unnamed: 0', 'title', 'text', 'label'], dtype='object')
      Unnamed: 0  … label
0           8476  …  FAKE
1          10294  …  FAKE
2           3608  …  REAL
3          10142  …  FAKE
4            875  …  REAL
…            …  …   …
6330        4490  …  REAL
6331        8062  …  FAKE
6332        8622  …  FAKE
6333        4021  …  REAL
6334        4330  …  REAL

[6335 rows x 4 columns]

Index(['title', 'text', 'subject', 'date'], dtype='object')
                                                   title  …                date
0        Donald Trump Sends Out Embarrassing New Year'…  …  December 31, 2017
1        Drunk Bragging Trump Staffer Started Russian …  …  December 31, 2017
2        Sheriff David Clarke Becomes An Internet Joke…  …  December 30, 2017
3        Trump Is So Obsessed He Even Has Obama's Name…  …  December 29, 2017
4        Pope Francis Just Called Out Donald Trump Dur…  …  December 25, 2017
…                                                    …  …                 …
23476  McPain: John McCain Furious That Iran Treated …  …   January 16, 2016
23477  JUSTICE? Yahoo Settles E-mail Privacy Class-ac…  …   January 16, 2016
23478  Sunnistan: US and Allied 'Safe Zone' Plan to T…  …   January 15, 2016
23479  How to Blow $700 Million: Al Jazeera America F…  …   January 14, 2016
23480  10 U.S. Navy Sailors Held by Iranian Military …  …   January 12, 2016

[23481 rows x 4 columns]

Index(['title', 'text', 'subject', 'date'], dtype='object')
                                                   title  …
date
0      As U.S. budget fight looms, Republicans flip t…  …  December 31, 2017
1      U.S. military to accept transgender recruits o…  …  December 29, 2017
2      Senior U.S. Republican senator: 'Let Mr. Muell…  …  December 31, 2017
3      FBI Russia probe helped by Australian diplomat…  …  December 30, 2017
4      Trump wants Postal Service to charge 'much mor…  …  December 29, 2017
…                                                    …  …
…
21412  'Fully committed' NATO backs new U.S. approach…  …    August 22, 2017
21413  LexisNexis withdrew two products from Chinese …  …    August 22, 2017
21414  Minsk cultural hub becomes haven from authorities  …    August 22, 2017
21415  Vatican upbeat on possibility of Pope Francis …  …    August 22, 2017
21416  Indonesia to buy $1.14 billion worth of Russia…  …    August 22, 2017
```

```
[21417 rows x 4 columns]

Index(['title', 'text', 'label'], dtype='object') (51233, 3)
```

[ ]:
```
                                                     title  … label
 51228  Erdogan, Putin to discuss Syria, Jerusalem dur…  …  REAL
 51229  LET THE BOYCOTTS BEGIN: KEURIG COFFEE and 4 Ma…  …  FAKE
 51230  Egypt blocks Human Rights Watch website amid w…  …  REAL
 51231  Minnesota governor to undergo surgery for canc…  …  REAL
 51232   Bernie Sanders Will Visit The Vatican To Spea…  …  FAKE

 [5 rows x 3 columns]
```

2 Analyzing Dataset

- Determining the Labels(outputs) and Inputs

[ ]:
```python
# Outputs or Labels
labels = dataset.label;
print(labels)

# Sample Data
df = dataset['text']
df
```

```
0        FAKE
1        REAL
2        REAL
3        REAL
4        FAKE
         …
51228    REAL
51229    FAKE
51230    REAL
51231    REAL
51232    FAKE
Name: label, Length: 51233, dtype: object
```

[ ]:
```
 0        While many in the United States have come to t…
 1        MANCHESTER, England (Reuters) - British Prime …
 2        ANKARA (Reuters) - Turkey received bids last F…
 3        WASHINGTON (Reuters) - Two former top U.S. int…
 4        What s it going to take for someone to step in…
                                ...
 51228    ANKARA (Reuters) - Russian President Vladimir …
 51229    Before Keurig, and other major companies commi…
 51230    CAIRO (Reuters) - Egypt has blocked the websit…
```

```
51231    (Reuters) - Minnesota Governor Mark Dayton wil…
51232    Presidential Candidate Bernie Sanders, who fac…
Name: text, Length: 51233, dtype: object
```

3 Splitting the dataset

- Split the dataset into training and testing sets.

```
[ ]: # Split Dataset with a 8:2 ratio for training and testing
     #         sampple data:: the 'text' column only.
     x_train,x_test,y_train,y_test = train_test_split(df, labels, test_size=0.2,
                                                      random_state=7)
     print(x_train)
     print(y_train)

     print(x_test)
```

```
6890     Police in this country had a great run. Let s …
45936    SYDNEY (Reuters) - Australians turned in 51,00…
22230    A Wisconsin judge has refused to order local o…
40082    LIMA (Reuters) - Brazilian builder Odebrecht […
42905    Many Americans have suspected that the rise of…
                                  …
13927    The NAACP staged a sit-in at the office of Ala…
919      WASHINGTON (Reuters) - U.S. Congressman Steve …
38467    BEIRUT (Reuters) - The Syrian government rejec…
10742
49689    Sean Spicer gave Hillary Clinton a little dig …
Name: text, Length: 40986, dtype: object
6890     FAKE
45936    REAL
22230    FAKE
40082    REAL
42905    FAKE
         …
13927    FAKE
919      REAL
38467    REAL
10742    FAKE
49689    FAKE
Name: label, Length: 40986, dtype: object
23499    Private jets, lots of cash, presidential suite…
5965     Trump administration officials are mulling an …
47211    **Want FOX News First in your inbox every day?…
25930     Members Of The Intelligence Committee Simply …
20885    TWITTER IS ABUZZ OVER THE FLY THAT LANDED ON H…
                                  …
```

```
28054    WASHINGTON (Reuters) - A group of Democratic l…
12110    KABUL (Reuters) - Vice President Mike Pence ma…
36322    West Virginia has been devastated by a loss of…
44194    B..b..but That can t be right Maybe we should …
44946    WASHINGTON (Reuters) - U.S. President Donald T…
Name: text, Length: 10247, dtype: object
```

4 Dataset Preprocessing using a TfidfVectorizer

Let's initialize a ***TfidfVectorizer*** with stop words from the English language and a maximum document frequency of 0.7 (terms with a higher document frequency will be discarded). Stop words are the most common words in a language that are to be filtered out before processing the natural language data. And a *TfidfVectorizer* turns a collection of raw documents into a matrix of TF-IDF features.

- **TF (Term Frequency)**: The number of times a word appears in a document is its Term Frequency. A higher value means a term appears more often than others, and so, the document is a good match when the term is part of the search terms.
- **IDF (Inverse Document Frequency):** Words that occur many times a document, but also occur many times in many others, may be irrelevant. IDF is a measure of how significant a term is in the entire corpus.

[ ]:
```
# Initialize a TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

# Fit and transform train set, transform test set
t0 = time()
tfidf_train = tfidf_vectorizer.fit_transform(x_train)
tfidf_test = tfidf_vectorizer.transform(x_test)
preprocess_time = time() - t0
print(tfidf_test)
```

```
  (0, 128393)    0.09160219142317329
  (0, 123421)    0.050692675829555334
  (0, 120018)    0.13358732532461629
  (0, 115093)    0.0572543300124491
  (0, 114933)    0.09845238614140546
  (0, 110795)    0.16255093777486046
  (0, 109086)    0.039520328193992676
  (0, 108049)    0.12782150832497632
  (0, 107958)    0.1345110883906628
  (0, 105019)    0.06957679124535648
  (0, 100453)    0.11024026684077948
  (0, 100003)    0.06632491121109792
  (0, 97064)     0.06050491380951155
  (0, 96725)     0.11157827801659405
  (0, 96543)     0.0747981500020641
  (0, 95274)     0.1397670183327725
  (0, 93563)	0.09798703217176453
  (0, 92821)	0.10223071197467873
  (0, 91620)	0.06732768161628239
  (0, 91280)	0.048005949675386045
  (0, 91183)	0.28908697090133667
  (0, 85962)	0.15111257704319278
  (0, 85626)	0.08882542864779046
  (0, 84505)	0.06068187171259706
  (0, 80052)	0.05359030768161428
  :	:
  (10246, 61957)	0.09524361237419485
  (10246, 57904)	0.10579831006675786
  (10246, 57355)	0.2059306774710701
  (10246, 57349)	0.11435602692754665
  (10246, 57118)	0.10385007438687317
  (10246, 55620)	0.12558886057179253
  (10246, 50944)	0.1456763258561254
  (10246, 50680)	0.1477767435225432
  (10246, 49653)	0.13879756022869624
  (10246, 44876)	0.2116990616578496
  (10246, 44873)	0.1721598615396174
  (10246, 37026)	0.05354472949786926
  (10246, 34395)	0.15432588922068582
  (10246, 33043)	0.08903448147152995
  (10246, 29124)	0.10468837600568447
  (10246, 29118)	0.167842154618641
  (10246, 28995)	0.14385537407476698
  (10246, 28993)	0.27219953112183265
  (10246, 21445)	0.1303388370844514
  (10246, 18643)	0.09024454380991116
  (10246, 9465)	0.07757296809157904
  (10246, 9298)	0.10567525369416872
  (10246, 9126)	0.09530504316714665
  (10246, 7233)	0.1478275291099032
  (10246, 1)	0.0805323369953822
```

5 Learning Model Selection

Choose a Learning Model, Methodology or Schema for training the dataset.

Here, as it's a classification problem, we are using a ***PassiveAggressiveClassifier*** due to the fact that we have vectorized the sample data during the preprocessing step using a *TfidfVectorizer*.

For explaination purpose, **Passive Aggressive algorithms** are online learning algorithms. Such an algorithm remains passive for a correct classification outcome, and turns aggressive in the event of a miscalculation, updating and adjusting. Unlike most other algorithms, it does not converge. Its purpose is to make updates that correct the loss, causing very little change in the norm of the weight vector.

```
[ ]: # Initialize a PassiveAggressiveClassifier
     model = PassiveAggressiveClassifier(max_iter=50)
     model
```

```
[ ]: PassiveAggressiveClassifier(C=1.0, average=False, class_weight=None,
                                 early_stopping=False, fit_intercept=True,
                                 loss='hinge', max_iter=50, n_iter_no_change=5,
                                 n_jobs=None, random_state=None, shuffle=True,
                                 tol=0.001, validation_fraction=0.1, verbose=0,
                                 warm_start=False)
```

6 Fitting the Model and Predicting its outcomes.

Here we will fit the model with the trained vectorized sample data from `tfidf_train` of the **TfidfVectorizer** and the `y_train` of the initial sample data.

```
[ ]: # Fitting the Model with the tfidf_train & y_train.
     t0 = time()
     history = model.fit(tfidf_train,y_train)
     train_time = time() - t0


     # Predict on the test set tfidf_test from the TfidfVectorizer
     t0 = time()
     y_pred = model.predict(tfidf_test)
     test_time = time() - t0
     print('Predicted Outcomes -->', y_pred, y_pred.shape)

     #           and

     # Calculate the accuracy with accuracy_score()
     score = accuracy_score(y_test,y_pred)
     # score = model.score(tfidf_test,y_pred)
     print(f'\nModel-Accuracy: {round(score*100,2)}%')
```

```
Predicted Outcomes --> ['FAKE' 'FAKE' 'REAL' … 'FAKE' 'REAL' 'REAL'] (10247,)

Model-Accuracy: 97.47%
```

7 Report and Visualization on the selected Model

From the Previous Step, after training and testing the model using *PassiveAggressiveClassifier* we got an *accuracy* of more than **97%** on the testing sample.

For more insights, we will print out the **confusion matrix** to view the number of false and true negatives and positives.

We will also print out the **heat map** generated from the confusion matrix.

```
[ ]: # Model Report
report = classification_report(y_test,y_pred, labels=['FAKE','REAL'])

print('-----------------=====Model Report=====-----------------\n')
print(report)
print('------------------------------------------------------\n')


# Confusion-Matrix
conf_mat = confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])

print('::Confusion-Matrix::')
print('',conf_mat)
print('\n------------------------------------------------------\n')


# Model Precision: what percentage of positive tuples are labeled as such?
print("Precision: \t%.2f\n" % metrics.precision_score(y_test, y_pred,␣
 ↪average='weighted'))

# Model Recall: what percentage of positive tuples are labelled as such?
print("Recall: \t%.2f\n" % metrics.recall_score(y_test, y_pred,␣
 ↪average='weighted'))

print("Density: \t%f\n" % density(model.coef_))

print("Dimensionality: %d\n" % model.coef_.shape[1])

print('------------------------------------------------------\n')

print("Train Time: \t%.3fs\n" % train_time)

print("Test Time:  \t%.3fs\n" % test_time)

print(f'Data-Preprocessing Time: {round(preprocess_time, 3)}s\n')

print('------------------------------------------------------\n')

print(f'Accuracy Score: {round(score*100,2)}%\n')

print('------------------------------------------------------\n')
```

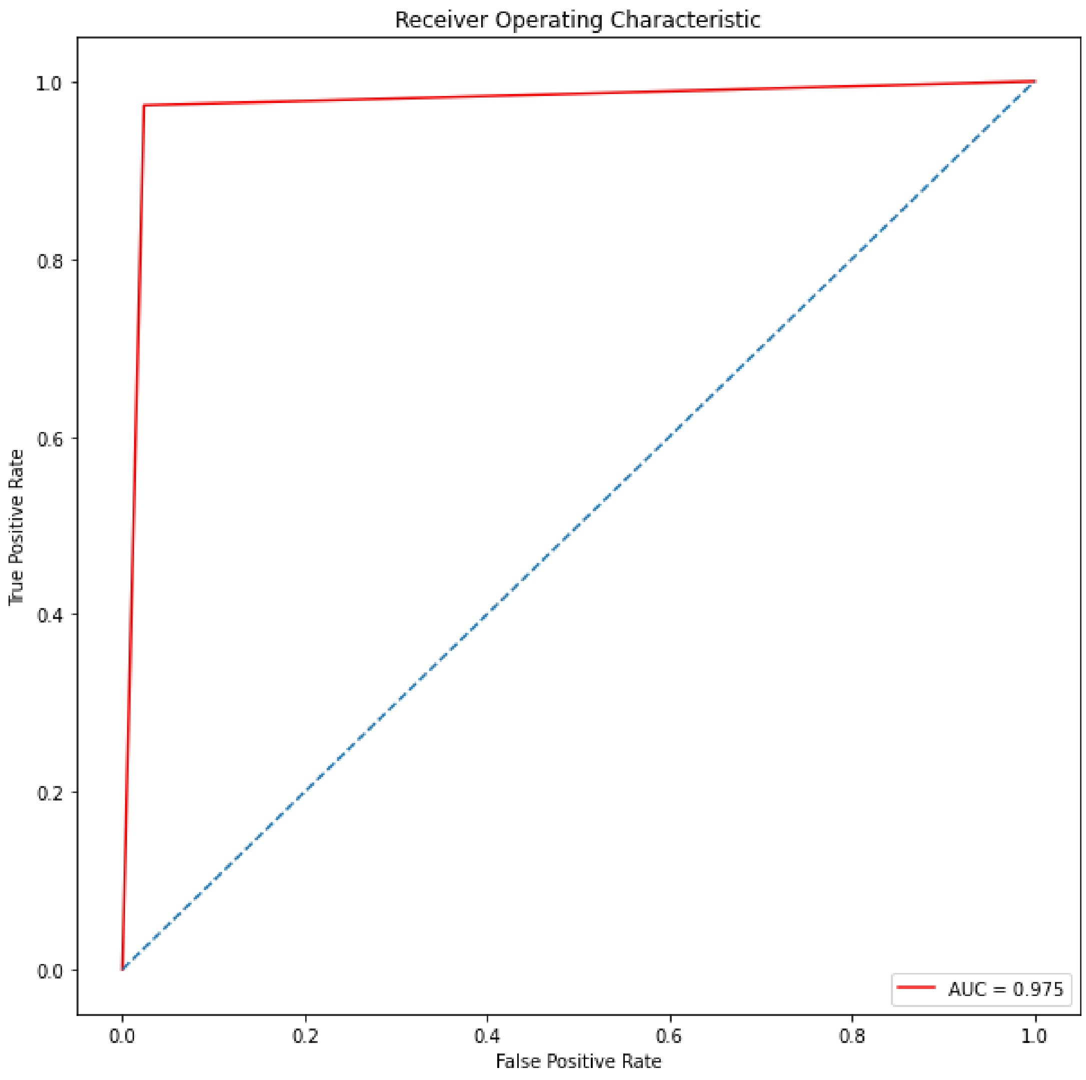
```
-----------------=====Model Report=====-----------------

              precision    recall  f1-score   support

        FAKE       0.97      0.98      0.98      5293
        REAL       0.97      0.97      0.97      4954

    accuracy                           0.97     10247
   macro avg       0.97      0.97      0.97     10247
weighted avg       0.97      0.97      0.97     10247

------------------------------------------------------

::Confusion-Matrix::
 [[5168  125]
 [ 134 4820]]

------------------------------------------------------

Precision:      0.97

Recall:         0.97

Density:        0.605463

Dimensionality: 129377

------------------------------------------------------

Train Time:     0.536s

Test Time:      0.005s

Data-Preprocessing Time: 12.507s

------------------------------------------------------

Accuracy Score: 97.47%

------------------------------------------------------
```

[ ]:
```
# transform
y_tt = []
y_pr = []
for yt in y_test:
  if yt == 'REAL':
    y_tt.append(1)
  elif yt == 'FAKE':
    y_tt.append(0)

for yp in y_pred:
  if yp == np.str_('REAL'):
    y_pr.append(1)
  elif yp == np.str_('FAKE'):
    y_pr.append(0)

# print(len(y_tt), len(y_pr))
false_positive_rate, true_positive_rate, thresholds = roc_curve(np.
 ↪array(y_tt),np.array(y_pr))
roc_auc = auc(false_positive_rate, true_positive_rate)
print('ROC-AUC =', roc_auc)
```

```
ROC-AUC = 0.9746675269269267
```

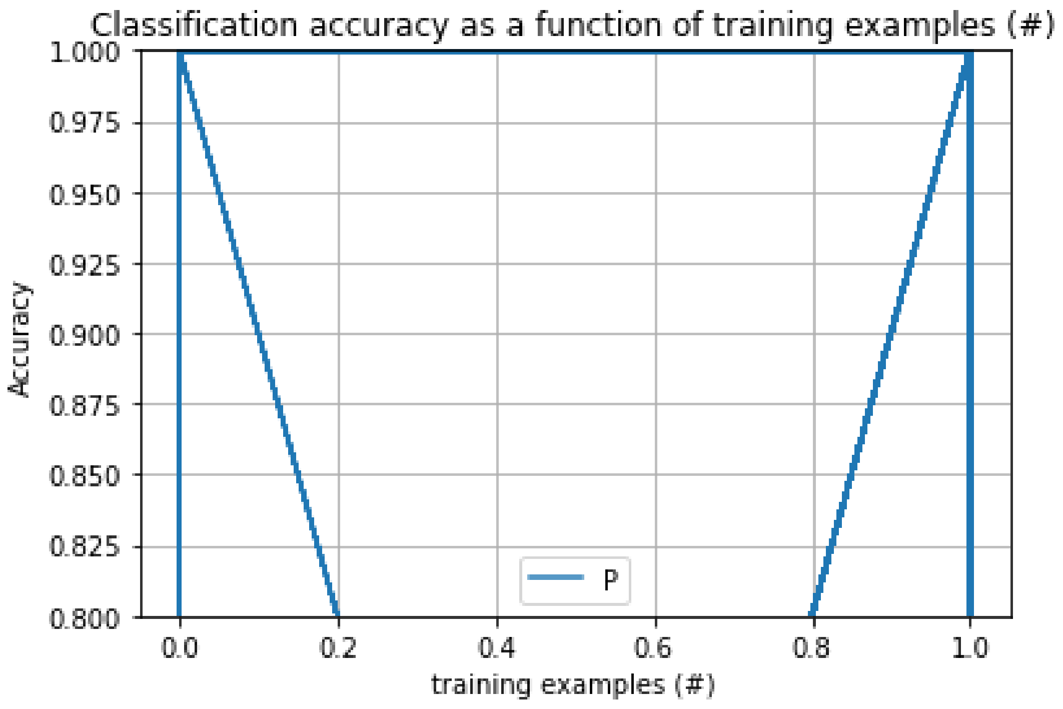
[ ]:
```
plt.figure(figsize=(10,10))
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate,true_positive_rate, color='red',label = 'AUC = %0.
 ↪3f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],linestyle='--')
plt.axis('tight')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
```

[ ]:
```
Text(0.5, 0, 'False Positive Rate')
```

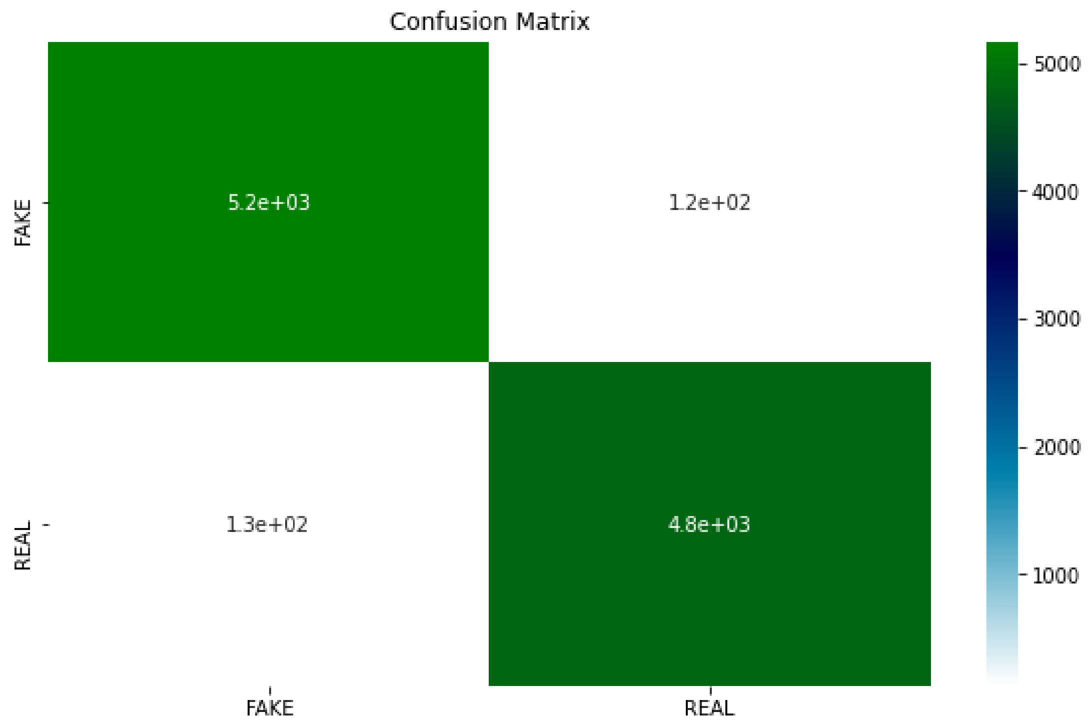


```
[ ]: def plot_accuracy(x, y, x_legend):
         """Plot accuracy as a function of x."""
         x = np.array(x)
         y = np.array(y)
         plt.title('Classification accuracy as a function of %s' % x_legend)
         plt.xlabel('%s' % x_legend)
         plt.ylabel('Accuracy')
         plt.grid(True)
         plt.plot(x, y)

     plt.figure()
     plot_accuracy(y_tt,y_pr,"training examples (#)")
     ax = plt.gca()
```

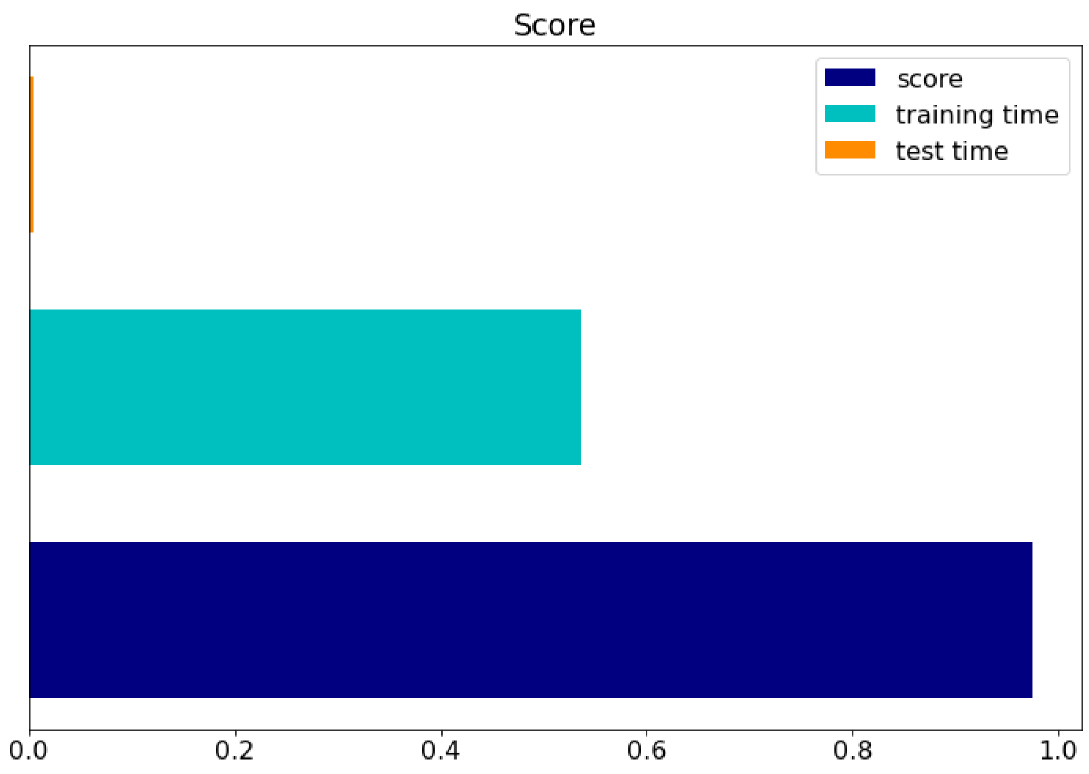
```
ax.set_ylim((0.8, 1))
plt.legend("PAC", loc='best')
plt.show()
```



[ ]:
```
# Visualization of confusion-matrix with a heat-map
fig = plt.figure(figsize=(10,6))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

sns.heatmap(conf_mat, annot=True, cmap=plt.cm.ocean_r,
            xticklabels=['FAKE','REAL'],
            yticklabels=['FAKE','REAL'])
plt.show()
```

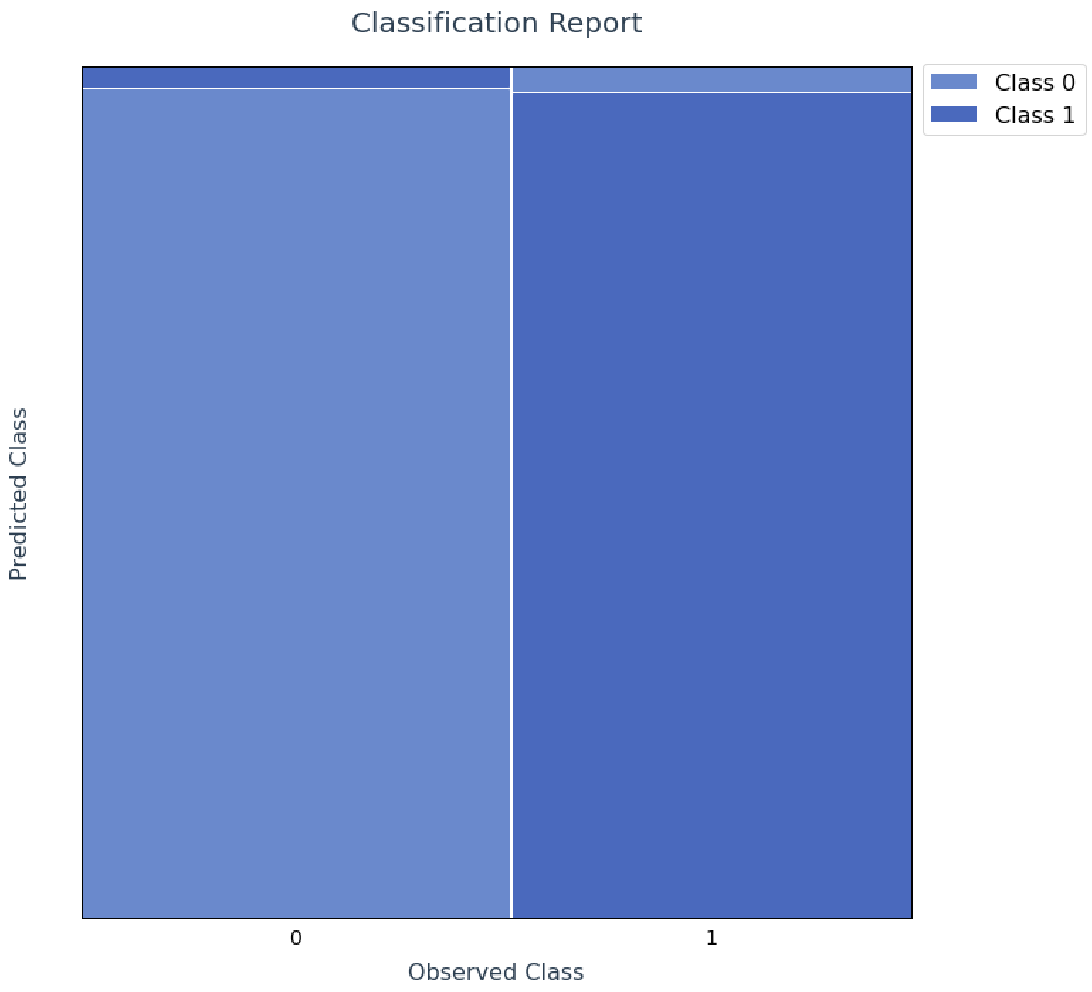


```
[ ]: indices = np.arange(1)
     plt.figure(figsize=(12, 8))
     plt.title("Score")
     plt.barh(indices, score, .2, label="score", color='navy')
     plt.barh(indices + .3, train_time, .2, label="training time",
              color='c')
     plt.barh(indices + .6, test_time, .2, label="test time", color='darkorange')
     plt.yticks(())
     plt.legend(loc='best')
     # plt.text(-.4, indices[0], 'Passive-Aggressive Classifier')
     plt.show()
```



[ ]:
```
# Converting numpyarray to list
results = list()
for line in conf_mat:
  results.append([data for data in line])
print(type(results), results)

# visualize the results
nclass_classification_mosaic_plot(len(results), results)
```

```
<class 'list'> [[5168, 125], [134, 4820]]
```



8 Testing The Model with user input

```
[ ]: text_data = "In recent weeks, Microsoft has detected cyberattacks targeting␣
      ↪people and organizations involved in the upcoming presidential election,␣
      ↪including unsuccessful attacks on people associated with both the Trump and␣
      ↪Biden campaigns, as detailed below. We have and will continue to defend our␣
      ↪democracy against these attacks through notifications of such activity to␣
      ↪impacted customers, security features in our products and services, and␣
      ↪legal and technical disruptions. The activity we are announcing today makes␣
      ↪clear that foreign activity groups have stepped up their efforts targeting␣
      ↪the 2020 election as had been anticipated, and is consistent with what the U.
      ↪S. government and others have reported. We also report here on attacks␣
      ↪against other institutions and enterprises worldwide that reflect similar␣
      ↪adversary activity." #@param {type:"raw"}
     print(text_data)
```

```python
x = [text_data,]

tfidf_test = tfidf_vectorizer.transform(x)

pred = model.predict(tfidf_test)

pred

print('This news is', pred[0])
```

```
In recent weeks, Microsoft has detected cyberattacks targeting people and
organizations involved in the upcoming presidential election, including
unsuccessful attacks on people associated with both the Trump and Biden
campaigns, as detailed below. We have and will continue to defend our democracy
against these attacks through notifications of such activity to impacted
customers, security features in our products and services, and legal and
technical disruptions. The activity we are announcing today makes clear that
foreign activity groups have stepped up their efforts targeting the 2020
election as had been anticipated, and is consistent with what the U.S.
government and others have reported. We also report here on attacks against
other institutions and enterprises worldwide that reflect similar adversary
activity.
This news is REAL
```