NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL

Report

PROJECT TITLE

Fake News Detection

Subject
Data Science Fundamentals (CS1635)

Branch
M. TECH CSE 1st Year

Submitted By
Udit Jain (24CSM1R23) (MTECH CSE)

Submitted To
Prof. P. Radha Krishna

Index

Report on Fake News Detection

Chapter 1: Introduction

Introduction

Since the digital era, social media has become a means of carrying information, distributing it, and conveying it. Facebook, Twitter, and Instagram are just but a few of the platforms that bring news and information to a massive number of people in real-time. The fact that this instantaneous and wide spread transfer of information is useful is a very dire challenge-the rise of fake news. Fake news refers to any information published in a form where it is thought of as actual news, but this information is meant to be either false or misleading, usually with the intention of deceiving viewers or controlling opinions or advancing specific agendas.

The influence of the creation of fake news on society is huge. It has been known to cause the breakdown of trust from credible media resources and spreads falsehoods instantaneously, thus swaying public opinion on a great issue such as politics, health, or science. For example, in the elections, fake news can influence the voters behaviour, such news during health crises, spreads some pathogenic information about treatments or preventive measures. It becomes further worse with clickbait techniques, sensationalism, and echo chambers where fake information is spread uncontrolled and rapidly. So combating fake news and controlling its proliferation becomes the urgent agenda among the researchers and developers of technology as well as policymakers.

This growing concern can be possibly addressed using artificially intelligent and machine learning advancement. Machine learning models, especially those that are domain-specific, can automatically classify fake news from others by learning the patterns and markers for distinguishing legitimate information from false or exaggerated ones. The most extensive training of such models on big datasets of real and fake news articles helps it recognize subtle cues, linguistic patterns, and contextual elements that suggest something is fake. It enables the efficient, scalable, and automated detection of fake news operations at speeds and scales necessary to keep up with the high volume of content on the internet.

Problem Statement

This paper discusses the most difficult issue of mass fake news spread in online social media that thwarts societal integrity and democratic processes. Fake news, in which misleading or false information is issued on purpose, creates several negative effects, among which loss of public trust in media houses and shaping public opinion and choice stand out. The authors supplemented the detection and classification of fake news with supervised artificial intelligence-based algorithms. In this study, three different variants of supervised AI-algorithms, namely Passive Aggressive

Classifier, Perceptron and Decision Stump are applied to different datasets of social media. In this research 29 different models are evaluated for accuracy, precision, and recall to classify genuine and falsified news content effectively.

The study would thus propose a predictive model for selecting the best performing algorithm for varied datasets to improve the detection and classification process. Moreover, research has developed the potential ways sensors could be employed in data collection for IoT environments that can potentially improve the identification of fake news within smart cities. Authors shall thereby contribute a sound framework to the identification of fake news, which increases the information's integrity and responsible digital discourse.

Necessity to solve that problem

Fake news, the intentional process of spreading false or misleading information, has become a big problem in modern times through social media. The sheer quick development of social media companies can spread fake news to millions at breakneck speed thus causing far-reaching societal, political, and economic impacts.

1. Maintain Information Integrity

Information integrity is one of the basics, mainly in data management and information systems. Fake news imperils the credibility of information online by defiling cyberspace with false or misleading content. This can influence individual, government, business, and institutional ability to make the right decisions and hence face misguidance, confusion, and harm.

2. Impact on Democratic Processes and Societal Stability

Fake news is understood in terms of influencing democratic processes in an alarming manner related to the shaping of elections, public opinion, and social discourse. An example is that in the 2016 U.S. Presidential Elections, there were widespread perceptions that fake news influenced the way voters perceived things. Since the infrastructure of social media and search engines is based on computer science, algorithms and systems have to be designed to identify and control such content. It can, therefore, lead to undermining democratic processes, and even social unrest if handled not appropriately

3. Disinformation in Public Health and Safety

The current COVID-19 pandemic has clearly indicated how misinformation in public health can be extremely destructive. From fake news regarding the virus, vaccines, and treatments, they can multiply to mislead the masses, induce hesitance towards the adoption of vaccines, and encourage dangerous behavior. Such a scenario calls for the development of technological solutions that can identify and help counter such misinformation, especially related to the areas of public health and safety.

4. Cyber Security Threat

Fake news and disinformation campaigns often dovetail into other forms of cybersecurity threats. The strategies of disinformation mechanisms are used to formulate different types of attacks, like phishing scams and fraud, which are to a large extent based on misleading content to manipulate users. Improvements in deepfake technologies with AI-generated content to create audio, video, and images meant to be extremely realistic aggravate the issue. Such manipulations will need tools that can found their basis upon computer science research into detecting content manipulation to prevent attacks of this nature.

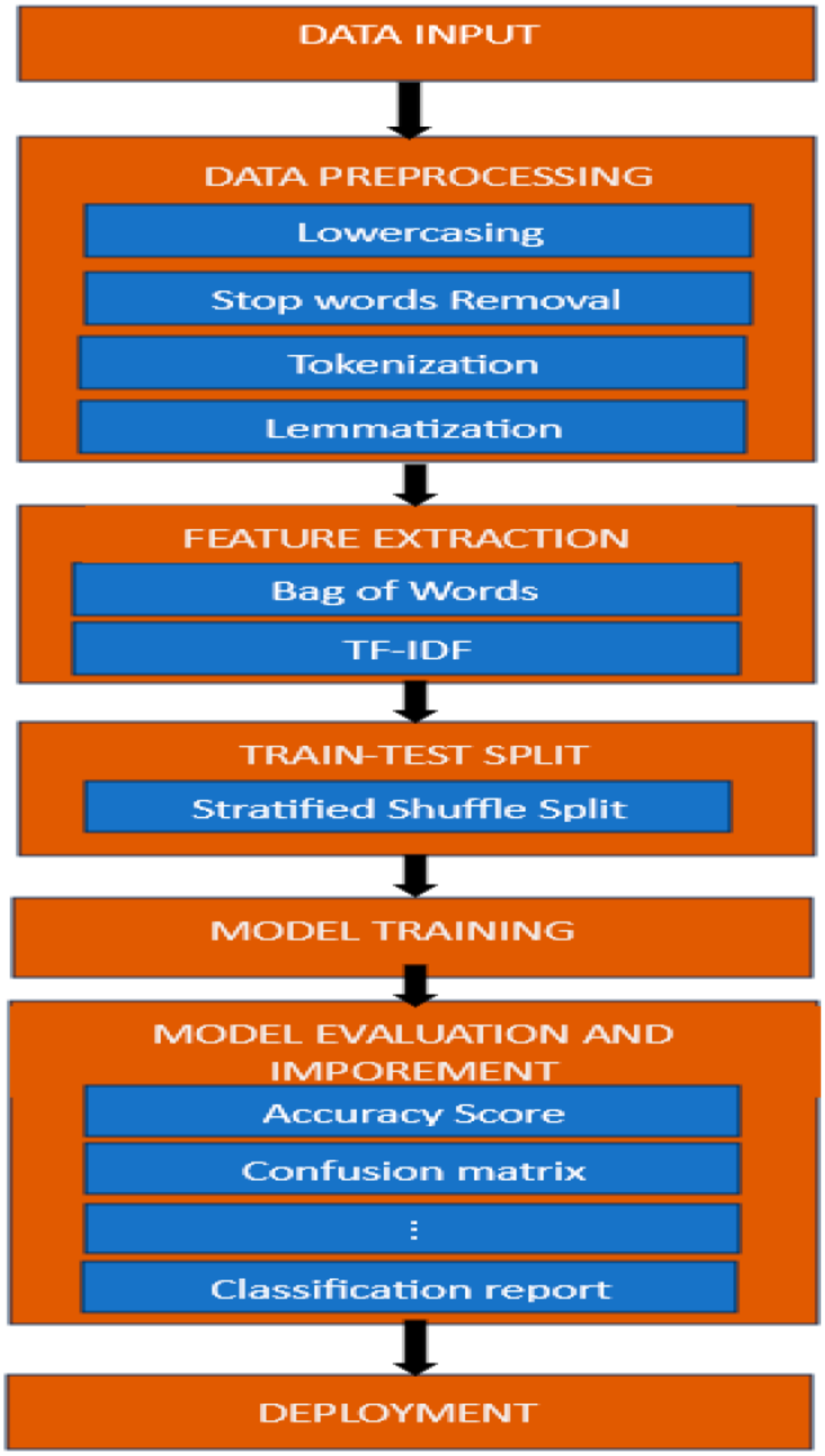
5. Impact on Financial Markets

Fake news would quickly be contrived to manipulate the stock prices or the value of the cryptocurrencies. For instance, fake news related to a particular company can cause drastic fluctuation in the stock markets that results in the loss of huge amounts of money from the investors. With the advancement in digital trading platforms and finance systems through technology, computer science is a fundamental element in designing algorithms and automated systems that could detect and neutralize the effect of fake financial news.

6. Ethical and Legal Issues

The spread of false news raises concern on a few ethical frontiers, especially in the development and deployment of machine learning algorithms. Biased or flawed algorithms could, by design, amplify fake news or suppress valid content. Here is where the issue from computer scientists to design ethics into systems that mediate free speech and prevent harm by fake news arises. Moreover, most governments have begun to start legal processes about the regulation of content on the internet. It is such an important aspect that the technology industries must come up with compliant solutions and avoid legal implications.

Chapter 2: Model Architecture



1. Data Preprocessing Layer

Focus: To clean up and transform raw text data into an analytical format.

This fake news detection model includes a strong text preprocessing pipeline in its paper that, via NLP methods, cleans and prepares textual data prior to classification. This is because raw news articles can only then be transformed into the format of being operable with machine learning algorithms and processed for effective analysis. Here comes the application of NLP-based text preprocessing in this scenario:

1.Lowercasing: All text is placed in lowercase for uniformity, which reduces the effect of case sensitivity on complexity, especially in a case where word frequency is being analyzed.

2.Tokenization: The text is broken down into tokens containing words or subwords. Tokenization is the division of sentences and paragraphs into smaller parts. Algorithms can break up a word into such smaller parts and analyse each contribution individually.

3. Remove Stop-words: The stop words include "the," "and," and "of." These are words that have very little meaning in the classification context. It reduces noise and focuses the model more on the informative words.

4. Removing Punctuation: Symbols and punctuation are generally removed as they do not have value in the classification task. This is to clean up the text so only meaningful characters remain.

5. Lemmatization and Stemming Lemmatization and stemming are reduction of words to their root form or base stem, respectively. For example, "running" and "ran" end up as "run". This helps in clustering similar words so that different forms of the same term would appear as one feature for the model to learn.

6. TF-IDF Transformation: Applicable after pre-processing, TF-IDF transforms text to numerical vectors, such as the importance of words in the document relative to the entire dataset. This step assigns greater weights to distinctive terms that are likely to differentiate fake news from real news.

This NLP preprocessing pipeline ensures that the dataset obtained at the end is quite optimized for all types of machine learning models so that there is a better probability of identifying linguistic patterns, which may predict whether the news is fake or real, in a standard, structured format.

2. Training Phase

Objective: Train several machine learning models to classify the news articles as "Fake" or "Real."

a.) Decision Tree Classifier:

The classifier uses a decision tree model, which is an algorithm in supervised learning used to classify news articles as "Fake" or "Real" based on certain textual features. This model makes use of the construction of tree-like structured decisions, where every internal node represents a feature in the data b.)while a branch represents a decision rule. Every leaf node represents a classification label which can be either "Fake" or "Real". Here is how it works step by step on fake news detection models, starting from feature selection and splitting of the training dataset in a real scenario.

1.Feature Selection and Splitting

- Every node in a Decision Tree is representing a decision based on a particular feature in the data, the kind used in this study derivatively coming from text data. In every step, the algorithm builds a tree using features that best split the data to separate articles with characteristics typical of fake news from articles with characteristics typical of real news.
- The best split can be based on Gini impurity or information gain:

$$Gini = 1 - \sum (p_i)^2$$

- **Gini Impurity:** Describes how "pure" each node is: how well separated classes are within it. The formula for Gini impurity represents the probability of each class within the node. The smaller the Gini value, the purer the split.
- **Information Gain**: Measures the decrease in entropy (disorder) after split so that "Fake" and "Real" news are as far apart as possible at each step. The more informative the split, the higher the information gain.

2. Recursive Splitting

- When the feature is selected, the data split into subgroups according to values of feature that get appended to make branches. Each of these subsets continues to split as more features are splitting them.

- When the recursive splitting happen till a stopping criterion is met. This includes reaching the limit into which a tree can go or when further splitting will not improve the method of classification.

3. Tree Structure and Leaf Nodes

- As the tree grows, it successively builds nodes and branches, but it finally forms a structure such that every leaf node is indeed a final decision. Every leaf node classifies the news article as "Fake" or "Real" according to which one of the classes is most common among the points that reach that leaf.
- This tree structure helps the Decision Tree Classifier comprehend the data hierarchically by segregating articles into two classes with a combination of features.

4. Classification

- A Decision Tree follows the rules of decisions for an unseen, new article to traverse from the root node down through a leaf node to a target based on the features of the article. This path, defined by specific features, brings the article to "Fake" or "Real."
- The final prediction of the model is decided by the class label assigned at the leaf node reached after considering the values of its features.

b.) Random Forest:

There is a variety of ensemble learning model known as the Random Forest Classifier that extends beyond the Decision Tree Classifier. It aggregates several decision trees that had been trained over different subsets of data and makes contributions to the overall prediction. This Random Forest algorithm can be found strongly suited for tasks such as fake news detection as it combines multiple strengths of the decision trees, therefore, building an overall version that reduces overfitting and boosts robustness. Here is a step-by-step explanation of how the Random Forest works in this context:

1. **Data sampling and tree construction**

Random Forest prepares an ensemble of decision trees by training each of them on a random subset of the data-this is done through bootstrap sampling. Such a sampling technique allows each tree to capture different patterns in the data.

- Tree uses a random subset of the features so that trees will not end up too similar and provides diversity in the ensemble. For example, only a portion of the features computed from text data such as term frequency or specific keywords are considered at each split.
- The Random Forest is less likely to overfit any one instance of training, since it selects features and data points randomly.

2.Tree Splitting and Feature Selection

- Each tree in the Random Forest behaves like a Decision Tree; however, it splits by some measure, say Gini impurity or information gain to maximize the purity or informativeness at the node.
- Since every tree is only using a random subset of features, every tree individually explores different decision paths. This feature selection will ensure that individual trees are not too closely correlated and capture a more extensive range of patterns in fake and real news content.

3.Generating Predictions from Each Tree

- Once trained, each decision tree in the forest will output a classification prediction either "Fake" or "Real" for a new, unseen article.
- Each tree is fitted to slightly different subsets of data and features; therefore, it is possible that for a given input, the same individual trees' outputs might be different. This diversity helps Random Forest avoid overfitting based on any single feature or pattern.

4. Aggregating Predictions (Voting)

Random Forest aggregates the predictions from all the individual trees using a voting mechanism:

- **Majority Voting**: Assign a Class to each tree, and then the class vote that gets most votes will be the prediction outcome. For instance, if most of the trees classify an article as "Fake," then the final outcome will be "Fake," and conversely for "Real."
- The noisy data points may wrongfully classify some trees, but the majority of the correct classifications outweigh them, making this a majority voting, and, therefore, the combined predictions give a more robust and accurate output.

c.) Perceptron Neural Network

The Perceptron is one of the most basic neural network models, that is to say good for simple binary classification tasks, for instance, "Fake" news vs "Real" news. Being a linear classifier, the Perceptron learns a decision boundary on input features and produces a binary output. There's a step-by-step explanation below of how a Perceptron works in classifying fake news:

1.Input Layer and Feature Vector

- A set of input features, derived from text preprocessing (such as term frequencies from TF-IDF), is used to form a feature vector. An input feature within the vector represents an attribute (e.g., word frequency or any indicator from the text) that can help classify news.
- The vector of features feeds into the Perceptron, where all features are mapped to an input node.

2. Initialization of Weights

- The weight is assigned to all the elements of the input vector. These weights reflect the importance or association with the features regarding whether it is "Fake" or "Real."
- With these initial weights, the Perceptron further trains them in such a way that eventually, with minimum classification error, the combination of weights gets tightened up.

3.Weighted Sum Calculation

- The Perceptron computes the weighted sum of its inputs as follows: multiply each input feature by its corresponding weight and then summation. The summation is the linear combination of the input features, and we can write it as

$$z = \sum (w_i \cdot x_i) + b$$

- where:
  - wi is the weight for feature iii,
  - xi is the input feature value for feature iii, and
  - b is the bias term, which helps shift the decision boundary to improve model flexibility.

4. Activation Function

- After calculating the weighted sum, the Perceptron applies an **activation function** (usually a step function) to determine the output.
- For the binary classification of fake news, the activation function is typically a step or threshold function:

$$f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

  - Here, an output of **1** might represent "Fake News," while **0** represents "Real News." This output classification depends on the linear combination of features being above or below a certain threshold.

5. Learning Process and Weight Update

- The Perceptron uses a learning algorithm that adjusts the weights on the basis of classification error.
- If the model's prediction is correct, the weights will not get modified.
- If the predicted output of the model is wrong, it updates the weights to decrease future errors. An update in this regard is achieved by moving the weights in the direction of the error using the formula

$$w_i = w_i + \Delta w_i$$

$$\Delta w_i = \eta \cdot (y - \hat{y}) \cdot x_i$$

    - **η** is the learning rate, which controls the step size for weight adjustments.
    - **y** is the actual label (1 for "Fake" and 0 for "Real").
    - **y^** is the predicted label.

- The model continues this iterative weight adjustment process for multiple epochs or until the model achieves a certain level of accuracy.

6. Binary Classification and Decision Boundary

- The Perceptron successfully finds a linear decision boundary that splits "Fake" and "Real" news based on the feature values. It learns the hyperplane in the feature space that divides the data points into two classes.
- For the purpose of detecting false news, this implies that the Perceptron attempts to discover the linear rule that would be able to split features that are more frequent in false news as opposed to the ones found in real news.

d.) AdaBoost Classifier

The AdaBoost Classifier is an ensemble learning algorithm that aggregates multiple weak classifiers to create a strong classifier. It trains a series of classifiers sequentially wherein each successive model emphasizes more the instances which are wrongly classified by previous models.

Here's an explanation of how the AdaBoost Classifier proceeds within the context of fake and real news stories:

1.Concept of Boosting and Poor Students

Boosting starts training a set of weak classifiers, which are mostly the simplest models, for example Decision Stumps or small Decision Trees.
Each of these weak classifiers is mostly much worse than random guessing, so that it can catch up with some patterns of the data but definitely not sufficient enough on its own.
• For the news detection fake application, a weak classifier learns a pattern in textual features such as particular keywords or linguistic structures that denote whether an article is "Fake" or "Real".

2.Weight Initialization

- Initial Weights: AdaBoost starts off by giving equal weights to all the training instances. Suppose there are n instances then each one gets a weight of 1/n1.
- These weights are utilized to represent how crucial each training example is. The initialization considers that in this context, all samples have equally important values.
- Objective: The model must update these weights in each one of its iterations in such a manner that harder-to-classify instances get more weight in subsequent rounds.

3.Training Weak Classifiers and Updating Weights

- **Error Calculation**: One uses the weighted training dataset for training each weak classifier. After training, it predicts class labels for the training examples.

- **Weighted Error Rate**: The error rate of each weak classifier is calculated based on the weighted misclassifications

$$\text{Error} = \sum w_i \cdot \mathrm{I}(y_i \neq \hat{y}_i)$$

- **Updating Weights**: AdaBoost increases the weights of incorrectly classified instances so that they receive more attention in the next round. Correctly classified instances have their weights reduced:

$$w_i = w_i \times e^{\alpha \cdot \mathrm{I}(y_i \neq \hat{y}_i)}$$

where α is a coefficient based on the weak classifier’s accuracy. This update makes the algorithm focus on instances that are harder to classify.

4. Alpha Calculation (Classifier Weight)

- Each weak classifier receives a weight α\alphaα that reflects its accuracy:

$$\alpha = \frac{1}{2} \ln \left(\frac{1 - \text{Error}}{\text{Error}} \right)$$

- A classifier with lower error is given a higher weight in the final decision, contributing more to the final classification.
- The calculated α\alphaα determines the importance of the weak classifier in the final ensemble.

5. Iterative Training of Weak Classifiers

- AdaBoost just goes on adding each new weak classifier at step after step. In every step it ensures that the classifiers used are focused on instances which previous classifiers had most difficulties with; thus, it slowly improves toward being a more accurate ensemble.
- The ensemble of weak classifiers learns progressively more complex patterns in the text data during the progression of training time, which results in higher accuracy to classify the fake news.

6. Final Prediction (Ensemble Voting)

- After multiple iterations, AdaBoost combines the predictions of all weak classifiers to make a final decision.
- **Weighted Voting**: The final prediction is a weighted vote, where each weak classifier’s prediction is weighted by its α\alphaα value. The overall prediction is determined as: Final Prediction

$$\text{Final Prediction} = \text{sign} \left(\sum \alpha_i \cdot h_i(x) \right)$$

- If the weighted sum is positive, the article is classified as “Fake”; if it is negative, it’s classified as “Real.” This weighted voting mechanism ensures that classifiers with higher accuracy have more influence on the final decision.

e.) Stochastic Gradient Descent (SGD) Classifier

The SGD Classifier is a linear model that uses an efficient optimization algorithm for minimizing the loss function by taking small steps in model weights. It is perfectly suited to big data, which makes it ideal for the application of text classification, such as fake news detection, since one has hundreds of thousands of instances and features. Now, let's see how SGD Classifier works within this framework:

1.Text Representation and Feature Extraction

- Firstly, textual data from articles needs to be pre-processed like tokenization, removal of stop-words, followed by TF-IDF transformation of it, which eventually converts it into numerical features.
- The SGD Classifier learns the numeral features that represent each article and helps it in distinguishing between "Fake" and "Real" news.

2. Gradient Descent Optimization

- The SGD Classifier is a gradient descent-based classifier, which is an iterative optimization algorithm that sets the model weights in such a way that minimizes a given loss function.
- For binary classification, log loss, or logistic regression, is the most frequently used loss function for fake news detection. It computes the difference between predicted and actual labels.
- Gradient descent is a method used in finding the minimum of the loss; it repeatedly steps in the direction of the negative gradient (slope) of the loss function, which minimizes the error.

3. How SGD Works

- In Stochastic Gradient Descent, after each individual training instance, the model updates weights instead of computing gradient over the whole dataset (as in traditional gradient descent). Thus, SGD can be much faster and may be used for high-sized datasets.

4. Learning Rate and Convergence

- Learning Rate (η): This is the control of step size for every weight update; the larger the value is for the learning rate, the faster the model learns, however may overshoot the optimal weights and vice versa, because a lower rate will make training slower but gives stability.

- Convergence: Since SGD updates weights after every example it encounters, the path is noisy to convergence. This makes the algorithm faster but noisy in terms of convergence. Convergence to a value is obtained if changes in the loss function become sufficiently small over the iterations.

5. Regularization to Avoid Overfitting

SGD classifiers often involve regularization terms to prevent overfitting. Some of the common regularization techniques used include:

- L2 Regularization (Ridge): Adds a penalty proportional to the square of the weights, encouraging the model to keep weights small.
- L1 Regularization (Lasso): Adds a penalty proportional to the absolute value of the weights, which can even lead to sparse models (some weights get zeroed).
- Regularization improves the generalizing ability of the model, as it discourages a complex configuration of weights so that it functions well on previously unseen data.

6. Making Predictions

- Once trained, the SGD Classifier predicts the class label (either “Fake” or “Real”) based on the learned weights and input features:

$$\hat{y} = \text{sign}(w \cdot x + b)$$

  - Here, $w \cdot x$ is the dot product of the weight vector and the feature vector, and b is the bias term.
  - The **sign** function outputs 1 if $w \cdot x + b \geq 0$ (e.g., classifying as “Fake”) and 0 otherwise (e.g., classifying as “Real”).

f.) Logistic Regression

Logistic Regression in Fake News Detection

Logistic Regression is a form of binary classification model. It applies the sigmoid function to predict the probability of a given instance belonging to one of two classes-say, "Fake" or "Real" news. Using logistic regression in fake news detection, it is these patterns within the text features that help classify articles as being either deceptive or legitimate. How logistic regression works within this context:

1.Text Representation and Feature Extraction

Such pre-processing steps on the text data include tokenization, removing stop words, and transforming into TF-IDF format. These are the numeric inputs to logistic regression, which tries to learn coefficients for all the features such that accuracy for "Fake" or "Real" classification is maximized

.

2.Sigmoid Function for Probability Prediction

Logistic regression uses the **sigmoid function** to convert the linear combination of feature values and weights into a probability score:

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\cdots+\beta_n X_n)}}$$

- β0 is the bias term,
- β1,…,βn are the coefficients (weights) associated with each feature, and
- X1…,Xn are the input feature values. This score, between 0 and 1, represents the likelihood that an article is classified as “Fake.”

3.Loss Function and Optimization

To adjust model weights effectively, logistic regression minimizes a **log-loss function**, which quantifies the difference between predicted probabilities and actual labels:

$$\text{Log-loss} = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i\log(p_i)+(1-y_i)\log(1-p_i)\right]$$

where:

- m is the number of training examples,
- yi is the actual label for instance iii,

- pi is the predicted probability that yi=1. The aim is to find the weight values that minimize this log-loss function, making the model as accurate as possible.

4.Gradient Descent Process

The model's weights are optimized through **gradient descent** or **stochastic gradient descent (SGD)**:

- For each iteration, the model calculates the gradient of the log-loss function with respect to each weight,

The weights are updated by moving them in the direction that minimizes the loss:

$$\beta_j = \beta_j - \eta \cdot \frac{\partial \text{Log-loss}}{\partial \beta_j}$$

where η is the learning rate, controlling the step size.

5.Regularization for Overfitting Prevention

Logistic regression models often incorporate regularization to avoid overfitting:

- **L2 Regularization** (Ridge): Adds a penalty proportional to the sum of squared weights, encouraging smaller weights overall.
- **L1 Regularization** (Lasso): Adds a penalty proportional to the absolute values of weights, promoting sparsity by setting some weights to zero. Regularization helps the model generalize better, making it more reliable on unseen data.

6.Making Predictions

After training, logistic regression uses the learned weights and bias to predict class labels:

$$\hat{y} = \text{sign}(\beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n)$$

where:

- y^is the predicted label (1 for "Fake" or 0 for "Real"),
- The sign function outputs 1 if the prediction is greater than or equal to 0 (classifying as "Fake") and 0 otherwise (classifying as "Real")

3. Prediction Phase

Process:

- Objective: Use each model to make independent predictions on new data. Each trained model makes predictions individually, outputting whether an article is classified as "Fake" or "Real.

- Predictions are stored separately for each model, allowing for analysis of individual model performance and comparison across models.

4. Evaluation and Analysis Phase

Goal: In part, test each model in isolation since the models do not aggregate to an output

Measures:

- **Accuracy:** percentage of articles that were correct

- **Precision:** correctness of positive predictions

- **Recall:** measure of ability of a model to capture all true positives

- **F1-score:** the harmonic mean of precision and recall, typically more useful than the others for imbalanced datasets

As it will evaluate each model independently, these models can be identified as those which excel for specific conditions or the characteristics of the dataset.

Chapter 3: Specification and Technology Used

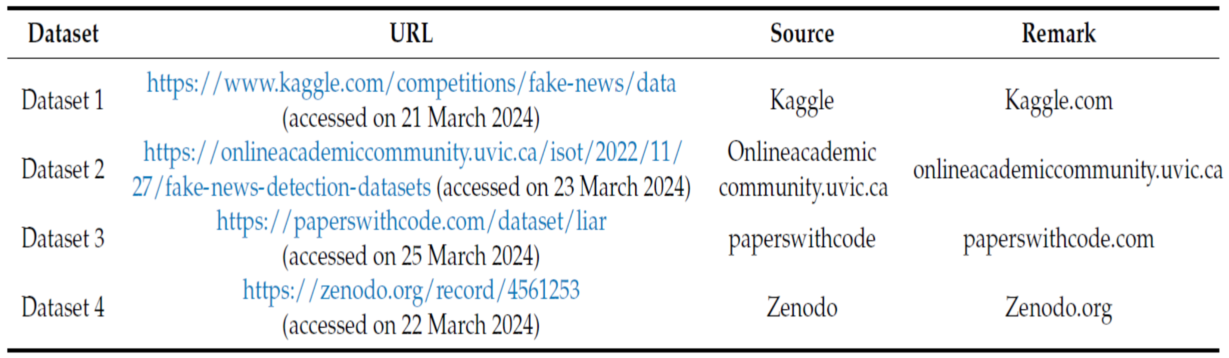
3.1 HARDWARE REQUIREMENTS

- PROCESSOR : Intel Core i3
- RAM : Minimum 4GB
- HARD DISK : 60GB and above

3.2 SOFTWARE REQUIREMENTS

- SOFTWARE : Vscode,Jupyter Notebook
- Windows 7 or Newer Version of Windows

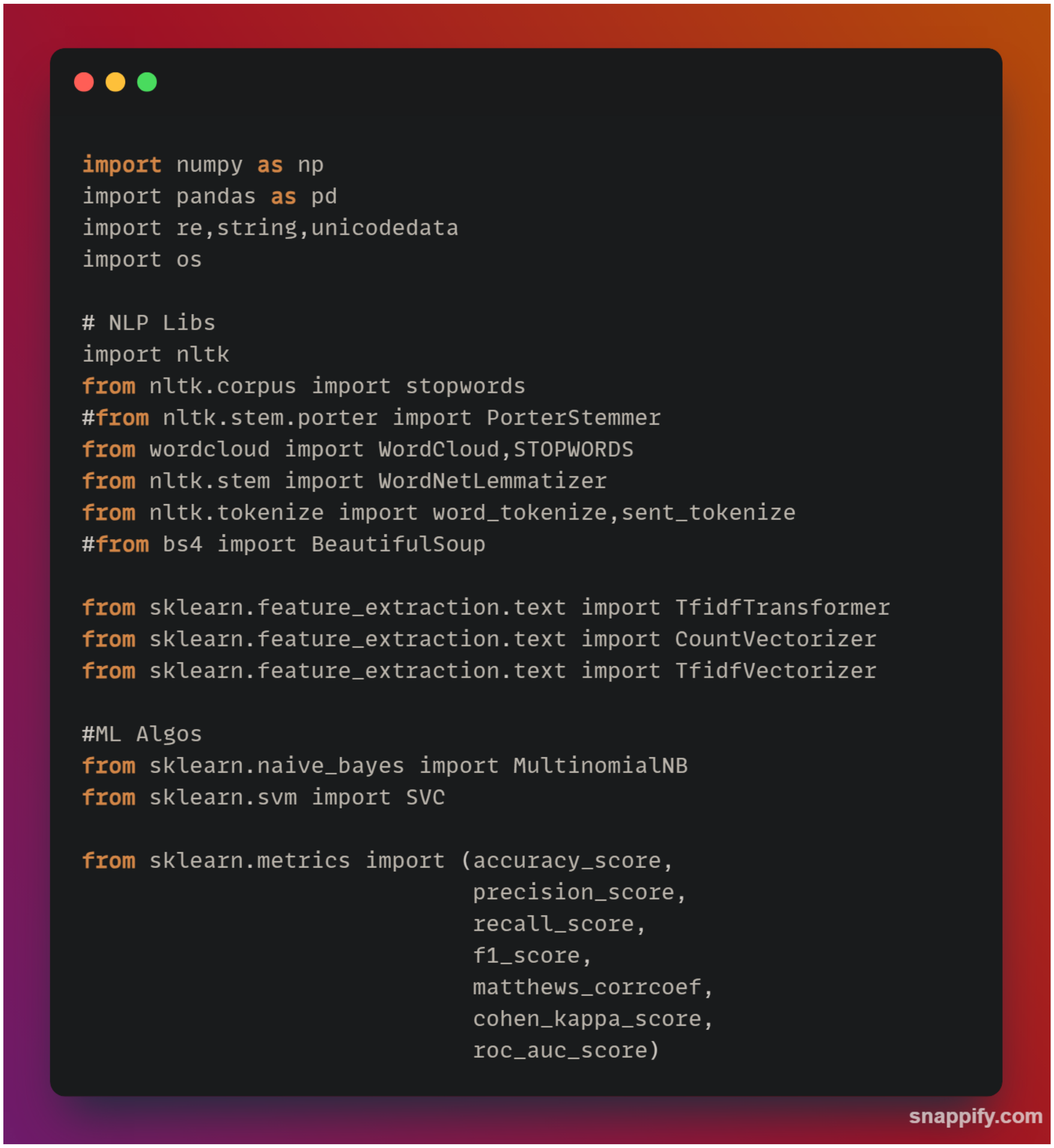
Chapter 4: Dataset



Dataset	URL	Source	Remark
Dataset 1	https://www.kaggle.com/competitions/fake-news/data (accessed on 21 March 2024)	Kaggle	Kaggle.com
Dataset 2	https://onlineacademiccommunity.uvic.ca/isot/2022/11/27/fake-news-detection-datasets (accessed on 23 March 2024)	Onlineacademic community.uvic.ca	onlineacademiccommunity.uvic.ca
Dataset 3	https://paperswithcode.com/dataset/liar (accessed on 25 March 2024)	paperswithcode	paperswithcode.com
Dataset 4	https://zenodo.org/record/4561253 (accessed on 22 March 2024)	Zenodo	Zenodo.org

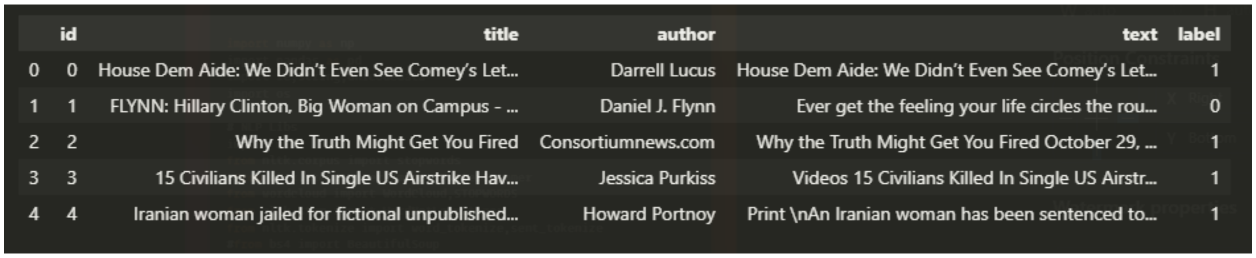
Chapter 5: Project Code Screenshots

Importing of Libraries



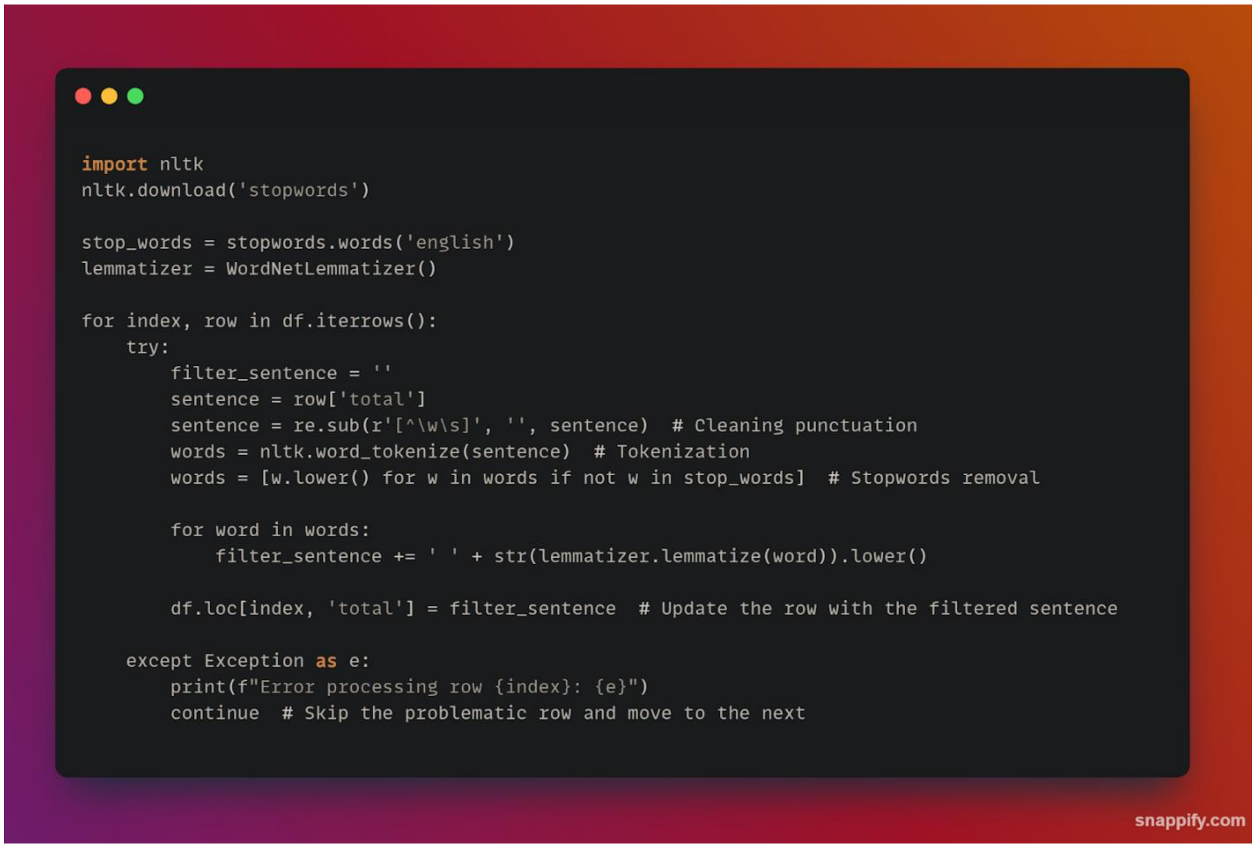
```python
import numpy as np
import pandas as pd
import re,string,unicodedata
import os

# NLP Libs
import nltk
from nltk.corpus import stopwords
#from nltk.stem.porter import PorterStemmer
from wordcloud import WordCloud,STOPWORDS
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
#from bs4 import BeautifulSoup

from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

#ML Algos
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC

from sklearn.metrics import (accuracy_score,
                             precision_score,
                             recall_score,
                             f1_score,
                             matthews_corrcoef,
                             cohen_kappa_score,
                             roc_auc_score)
```

Data Frame



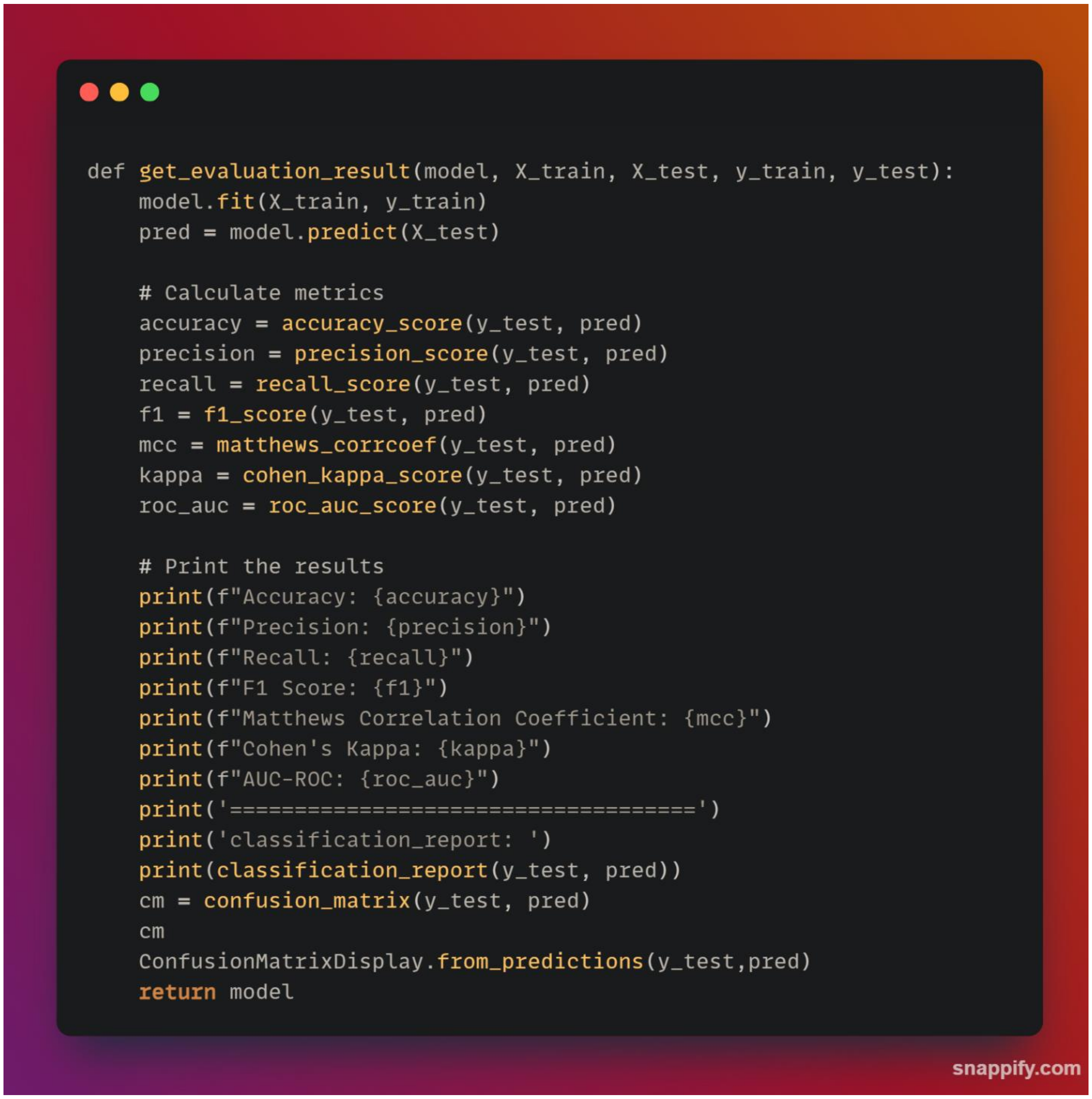
	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

NLP preprocessing



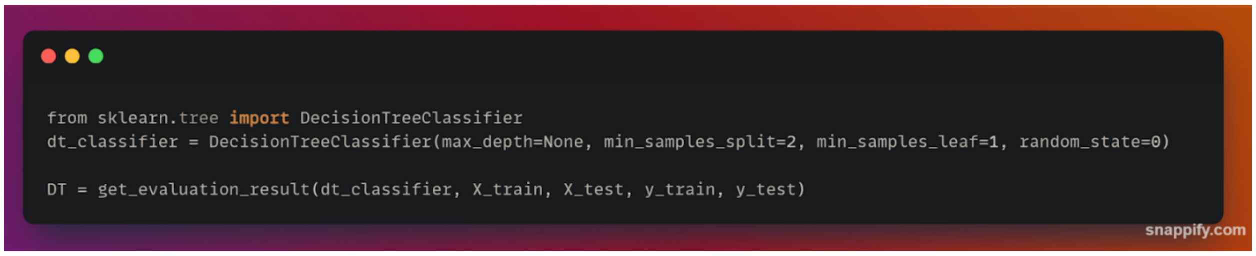
```python
import nltk
nltk.download('stopwords')

stop_words = stopwords.words('english')
lemmatizer = WordNetLemmatizer()

for index, row in df.iterrows():
    try:
        filter_sentence = ''
        sentence = row['total']
        sentence = re.sub(r'[^\w\s]', '', sentence)  # Cleaning punctuation
        words = nltk.word_tokenize(sentence)  # Tokenization
        words = [w.lower() for w in words if not w in stop_words]  # Stopwords removal

        for word in words:
            filter_sentence += ' ' + str(lemmatizer.lemmatize(word)).lower()

        df.loc[index, 'total'] = filter_sentence  # Update the row with the filtered sentence

    except Exception as e:
        print(f"Error processing row {index}: {e}")
        continue  # Skip the problematic row and move to the next
```

Metrics Used for Evaluating Results



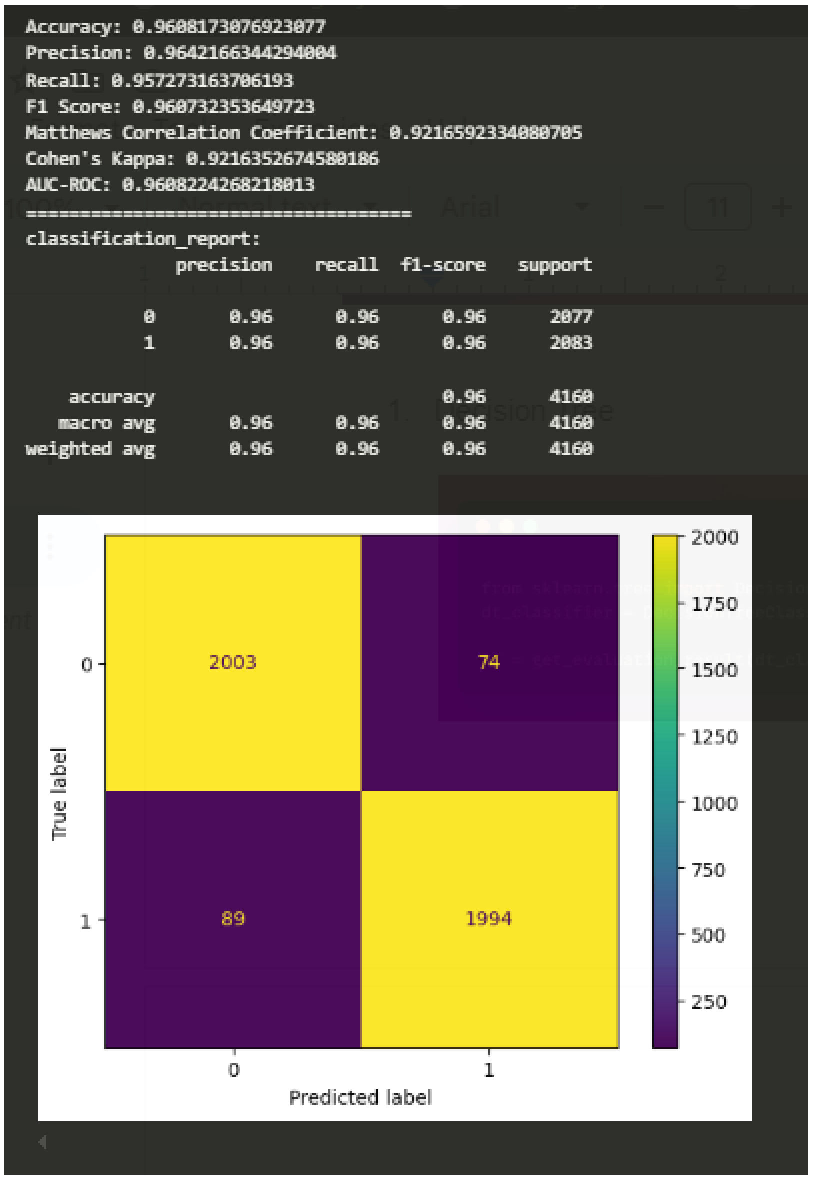
```
def get_evaluation_result(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    pred = model.predict(X_test)

    # Calculate metrics
    accuracy = accuracy_score(y_test, pred)
    precision = precision_score(y_test, pred)
    recall = recall_score(y_test, pred)
    f1 = f1_score(y_test, pred)
    mcc = matthews_corrcoef(y_test, pred)
    kappa = cohen_kappa_score(y_test, pred)
    roc_auc = roc_auc_score(y_test, pred)

    # Print the results
    print(f"Accuracy: {accuracy}")
    print(f"Precision: {precision}")
    print(f"Recall: {recall}")
    print(f"F1 Score: {f1}")
    print(f"Matthews Correlation Coefficient: {mcc}")
    print(f"Cohen's Kappa: {kappa}")
    print(f"AUC-ROC: {roc_auc}")
    print('====================================')
    print('classification_report: ')
    print(classification_report(y_test, pred))
    cm = confusion_matrix(y_test, pred)
    cm
    ConfusionMatrixDisplay.from_predictions(y_test,pred)
    return model
```

1. Decision Tree



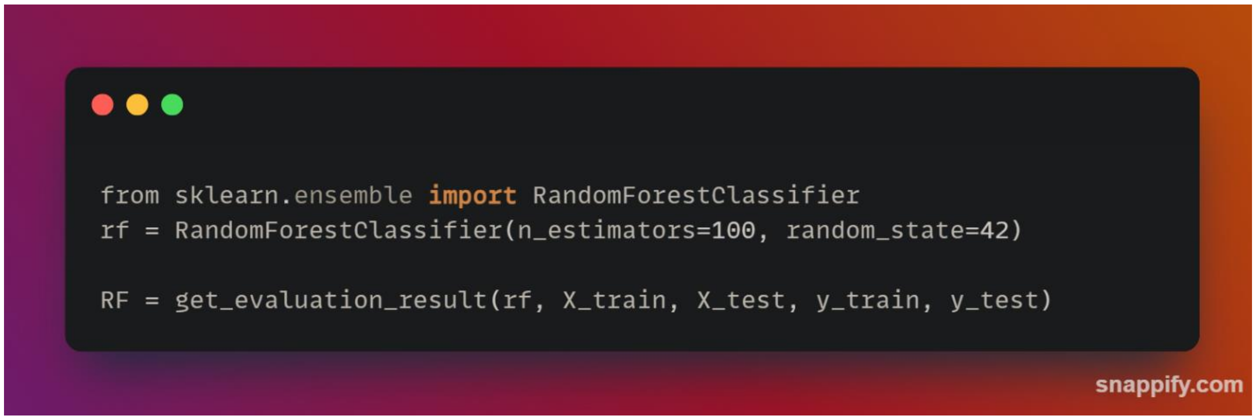
```
from sklearn.tree import DecisionTreeClassifier
dt_classifier = DecisionTreeClassifier(max_depth=None, min_samples_split=2, min_samples_leaf=1, random_state=0)

DT = get_evaluation_result(dt_classifier, X_train, X_test, y_train, y_test)
```

Result



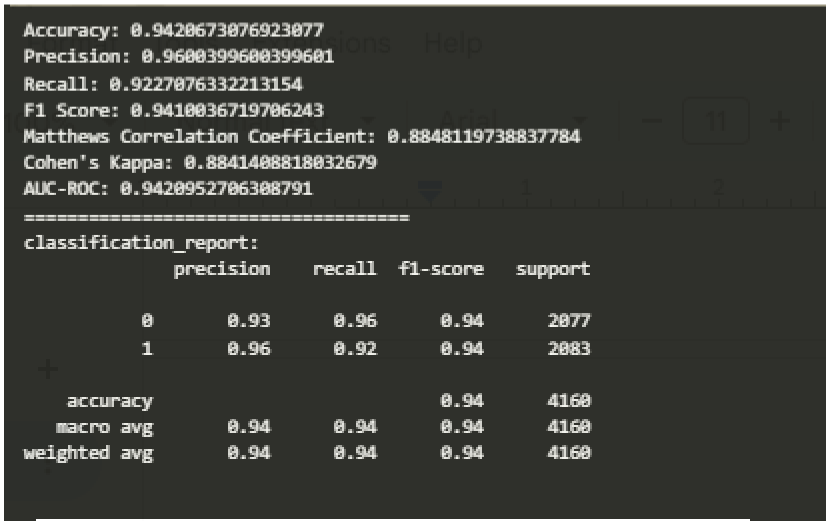
```
Accuracy: 0.9608173076923077
Precision: 0.9642166344294004
Recall: 0.957273163706193
F1 Score: 0.960732353649723
Matthews Correlation Coefficient: 0.9216592334080705
Cohen's Kappa: 0.9216352674580186
AUC-ROC: 0.9608224268218013
====================================
classification_report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      2077
           1       0.96      0.96      0.96      2083

    accuracy                           0.96      4160
   macro avg       0.96      0.96      0.96      4160
weighted avg       0.96      0.96      0.96      4160
```

2. Random Forest

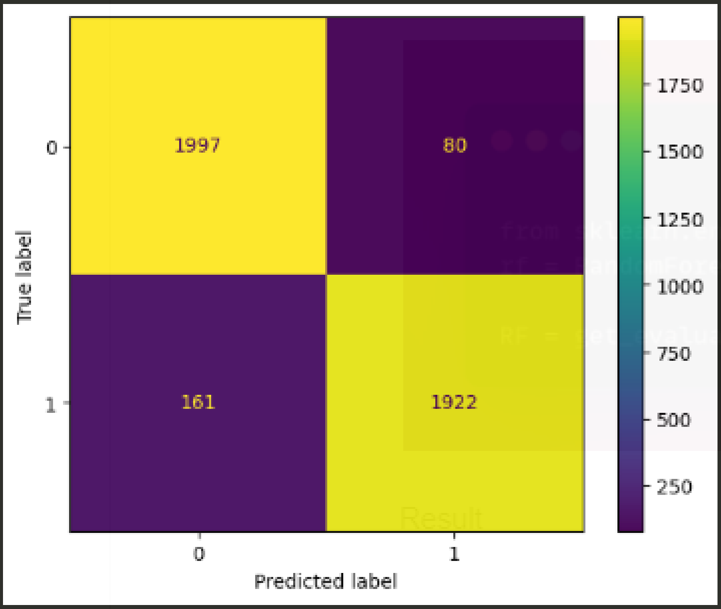


```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)

RF = get_evaluation_result(rf, X_train, X_test, y_train, y_test)
```

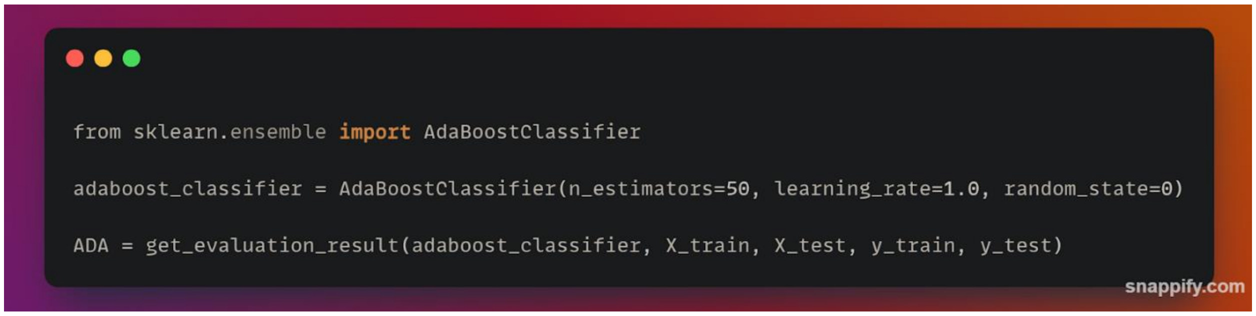
Result



```
Accuracy: 0.9420673076923077
Precision: 0.9600399600399601
Recall: 0.9227076332213154
F1 Score: 0.9410036719706243
Matthews Correlation Coefficient: 0.8848119738837784
Cohen's Kappa: 0.8841408818032679
AUC-ROC: 0.9420952706308791
====================================
classification_report:
              precision    recall  f1-score   support

           0       0.93      0.96      0.94      2077
           1       0.96      0.92      0.94      2083

    accuracy                           0.94      4160
   macro avg       0.94      0.94      0.94      4160
weighted avg       0.94      0.94      0.94      4160
```

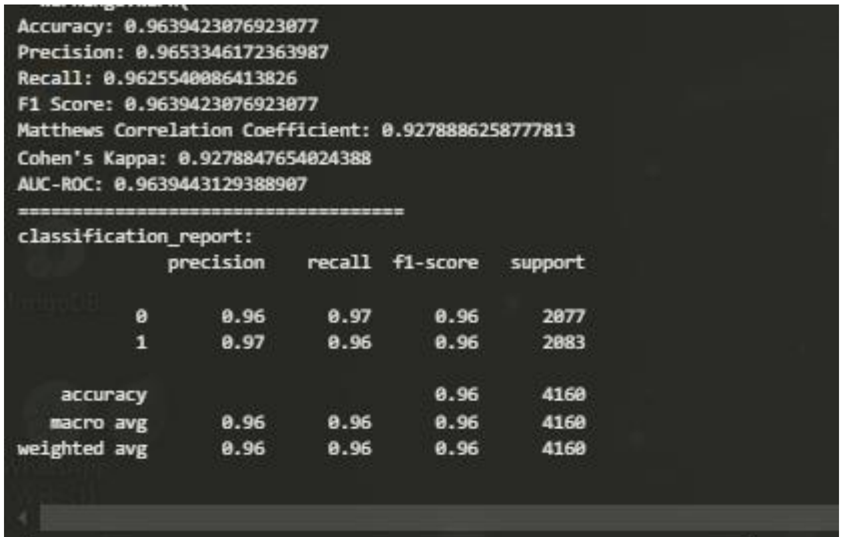


3.Adaboost

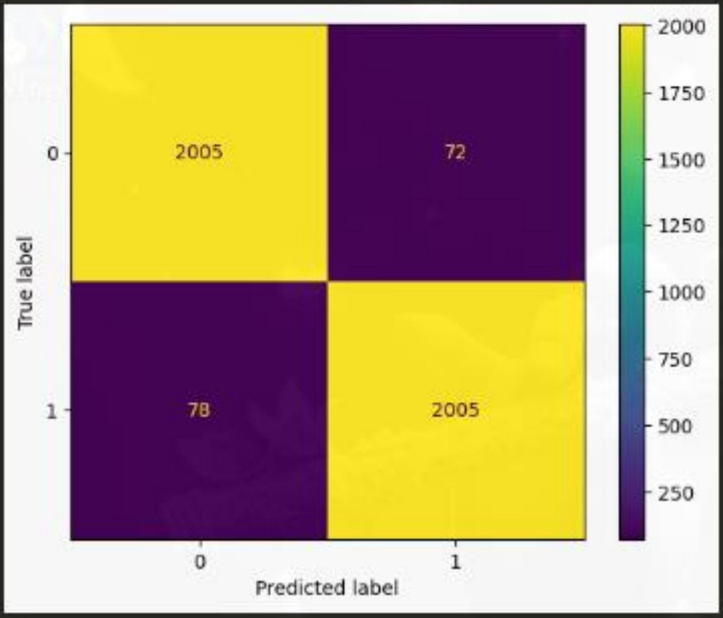


```
from sklearn.ensemble import AdaBoostClassifier

adaboost_classifier = AdaBoostClassifier(n_estimators=50, learning_rate=1.0, random_state=0)

ADA = get_evaluation_result(adaboost_classifier, X_train, X_test, y_train, y_test)
```

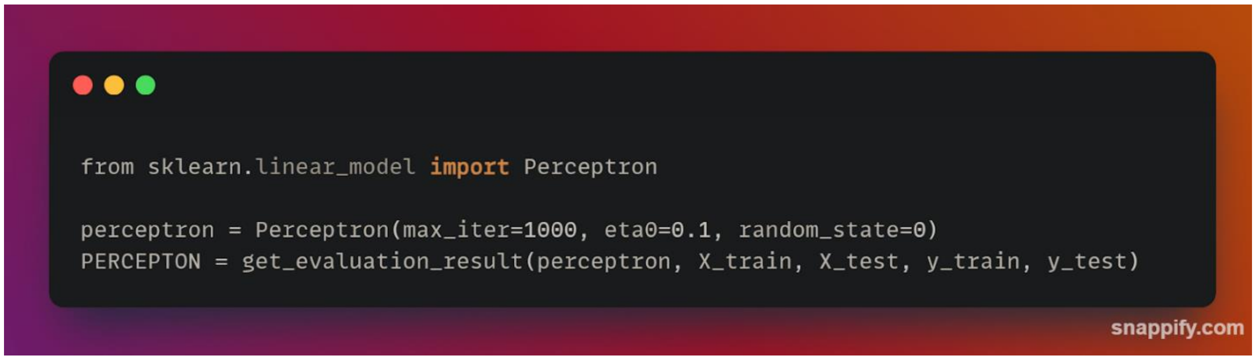
Result



```
Accuracy: 0.9639423076923077
Precision: 0.9653346172363987
Recall: 0.9625540086413826
F1 Score: 0.9639423076923077
Matthews Correlation Coefficient: 0.9278886258777813
Cohen's Kappa: 0.9278847654024388
AUC-ROC: 0.9639443129388907
====================================
classification_report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      2077
           1       0.97      0.96      0.96      2083

    accuracy                           0.96      4160
   macro avg       0.96      0.96      0.96      4160
weighted avg       0.96      0.96      0.96      4160
```

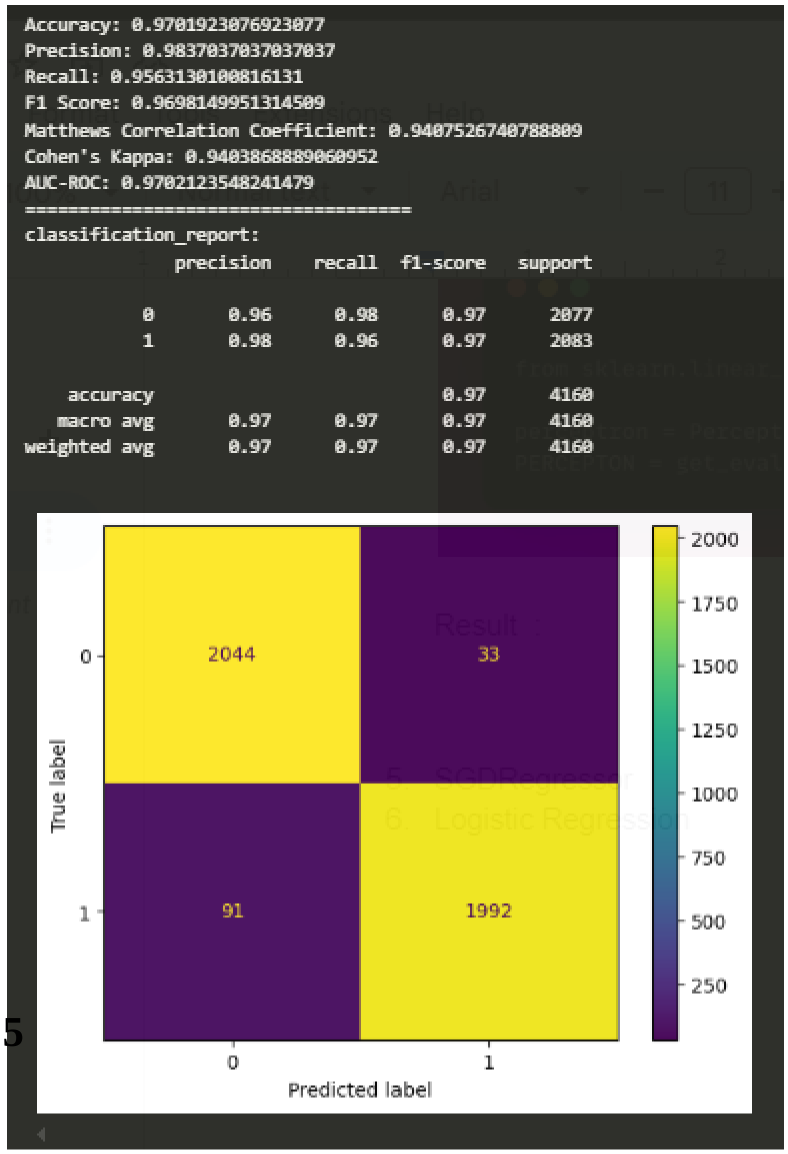


4.Perceptron



```python
from sklearn.linear_model import Perceptron

perceptron = Perceptron(max_iter=1000, eta0=0.1, random_state=0)
PERCEPTON = get_evaluation_result(perceptron, X_train, X_test, y_train, y_test)
```

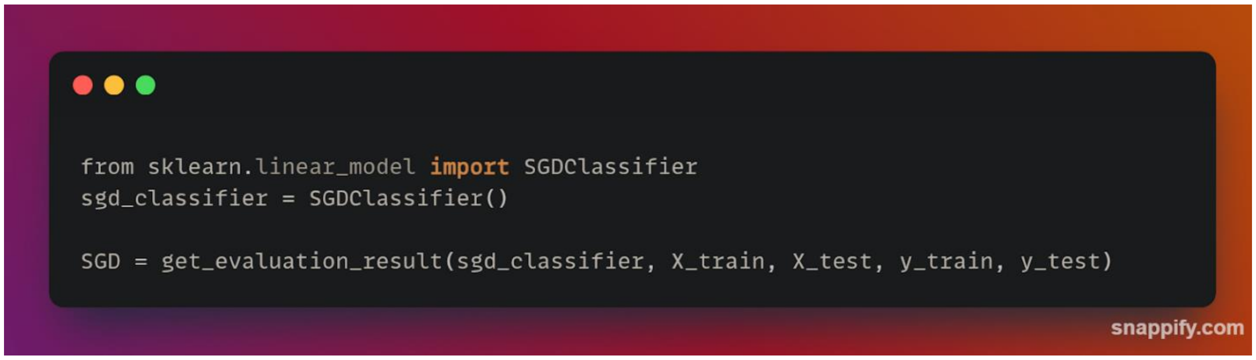
Result



```
Accuracy: 0.9701923076923077
Precision: 0.9837037037037037
Recall: 0.9563130100816131
F1 Score: 0.9698149951314509
Matthews Correlation Coefficient: 0.9407526740788809
Cohen's Kappa: 0.9403868889060952
AUC-ROC: 0.9702123548241479
====================================
classification_report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      2077
           1       0.98      0.96      0.97      2083

    accuracy                           0.97      4160
   macro avg       0.97      0.97      0.97      4160
weighted avg       0.97      0.97      0.97      4160
```

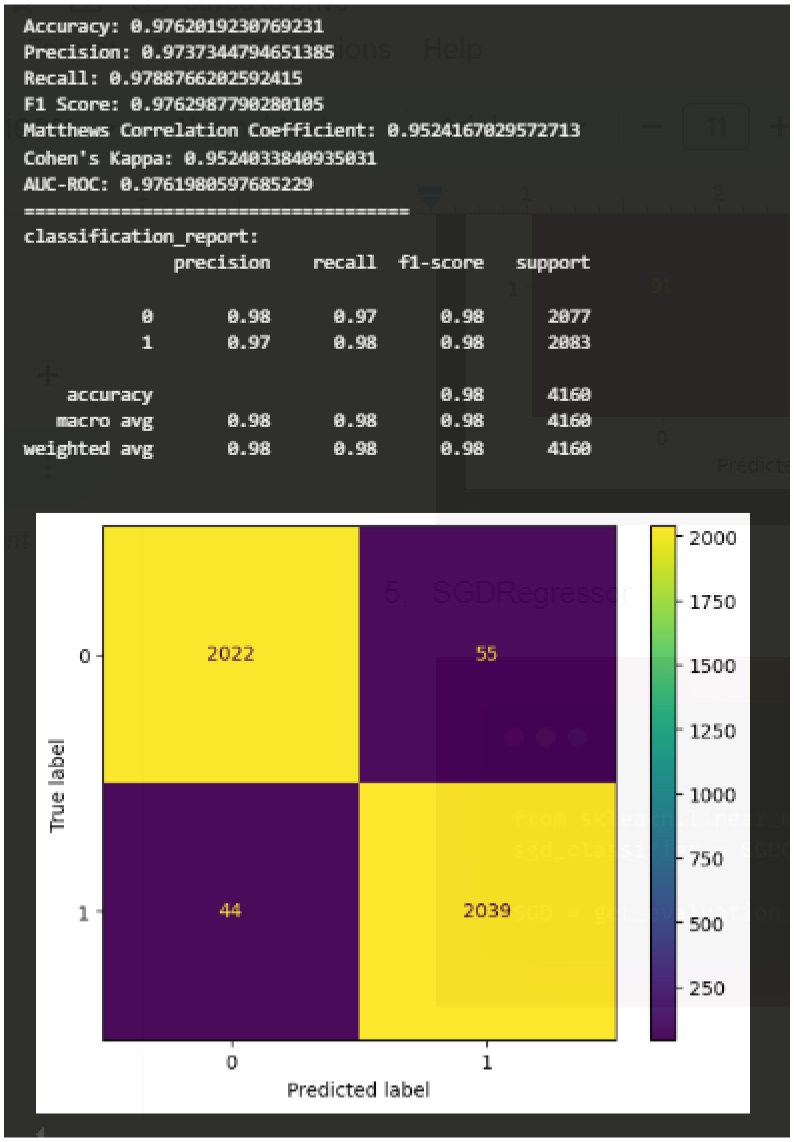
5

5. SGD Classifier



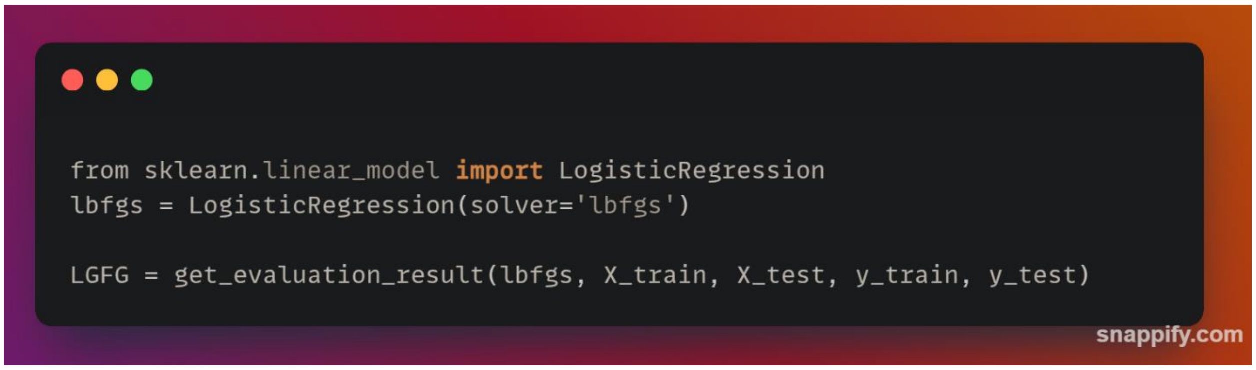
```
from sklearn.linear_model import SGDClassifier
sgd_classifier = SGDClassifier()

SGD = get_evaluation_result(sgd_classifier, X_train, X_test, y_train, y_test)
```

Result



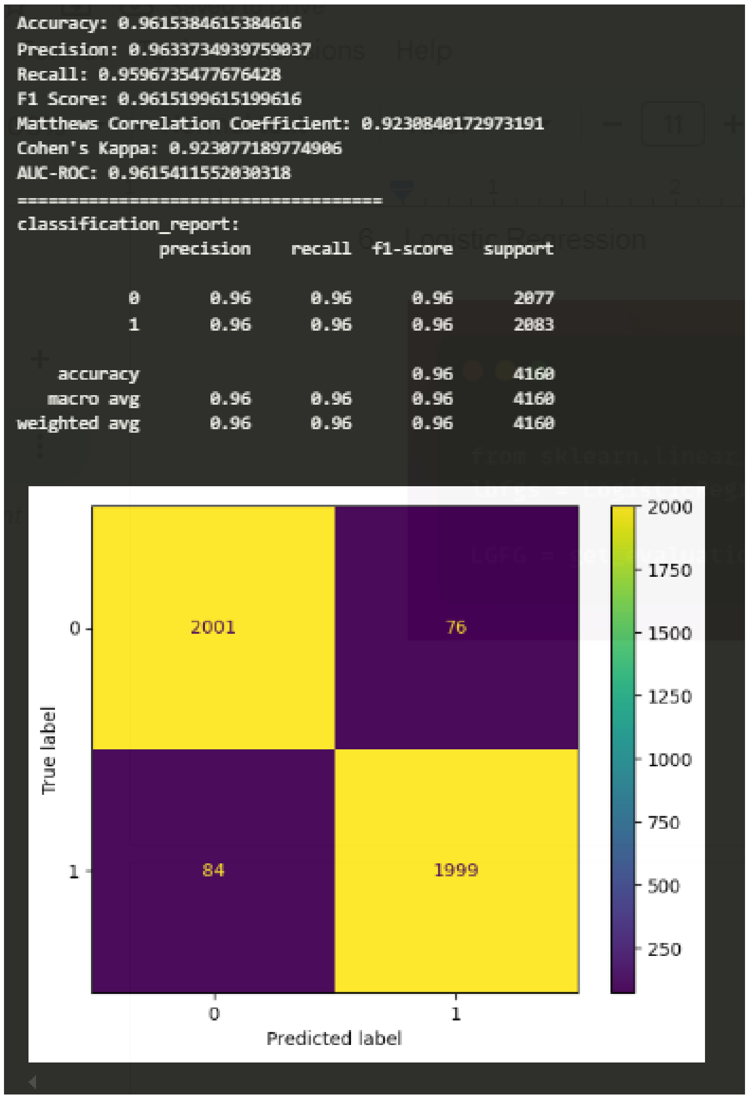
```
Accuracy: 0.9762019230769231
Precision: 0.9737344794651385
Recall: 0.9788766202592415
F1 Score: 0.9762987790280105
Matthews Correlation Coefficient: 0.9524167029572713
Cohen's Kappa: 0.9524033840935031
AUC-ROC: 0.9761980597685229
====================================
classification_report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      2077
           1       0.97      0.98      0.98      2083

    accuracy                           0.98      4160
   macro avg       0.98      0.98      0.98      4160
weighted avg       0.98      0.98      0.98      4160
```

6. Logistic Regression



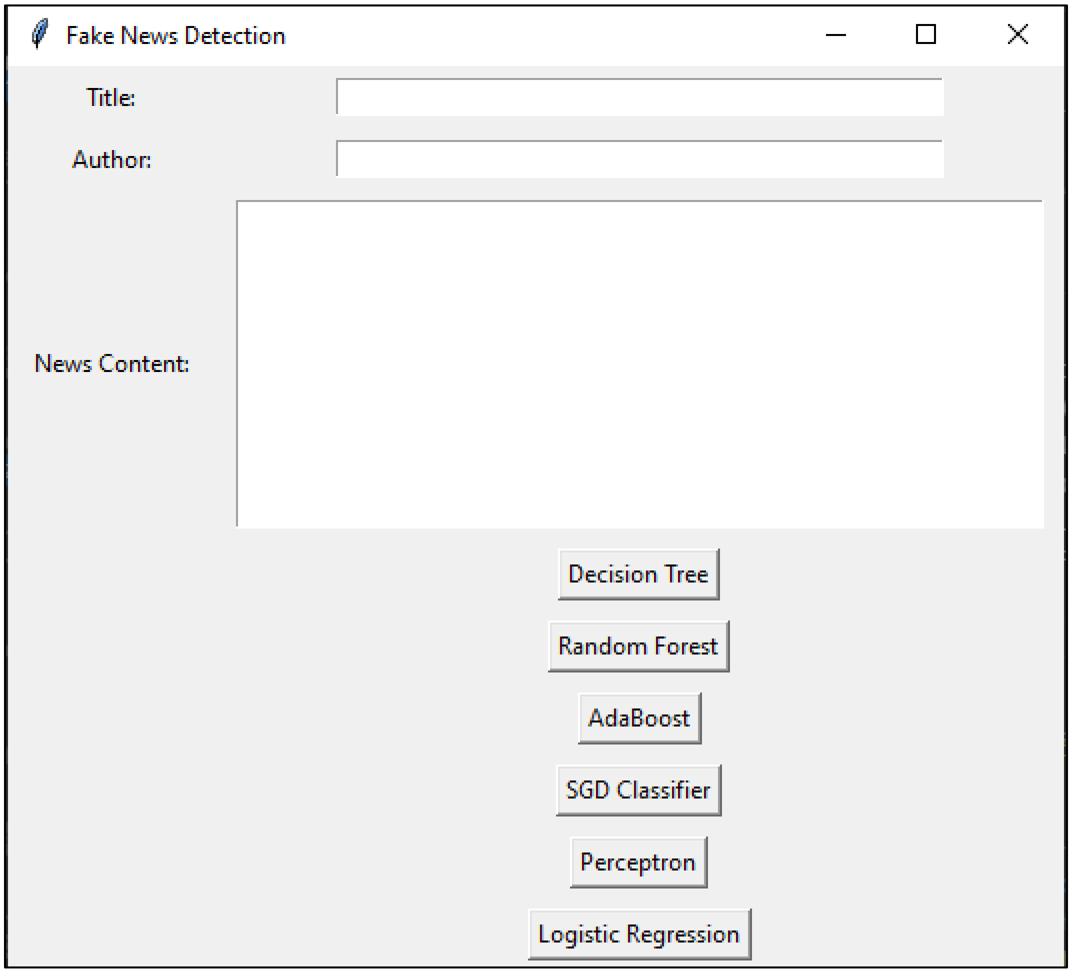
```
from sklearn.linear_model import LogisticRegression
lbfgs = LogisticRegression(solver='lbfgs')

LGFG = get_evaluation_result(lbfgs, X_train, X_test, y_train, y_test)
```

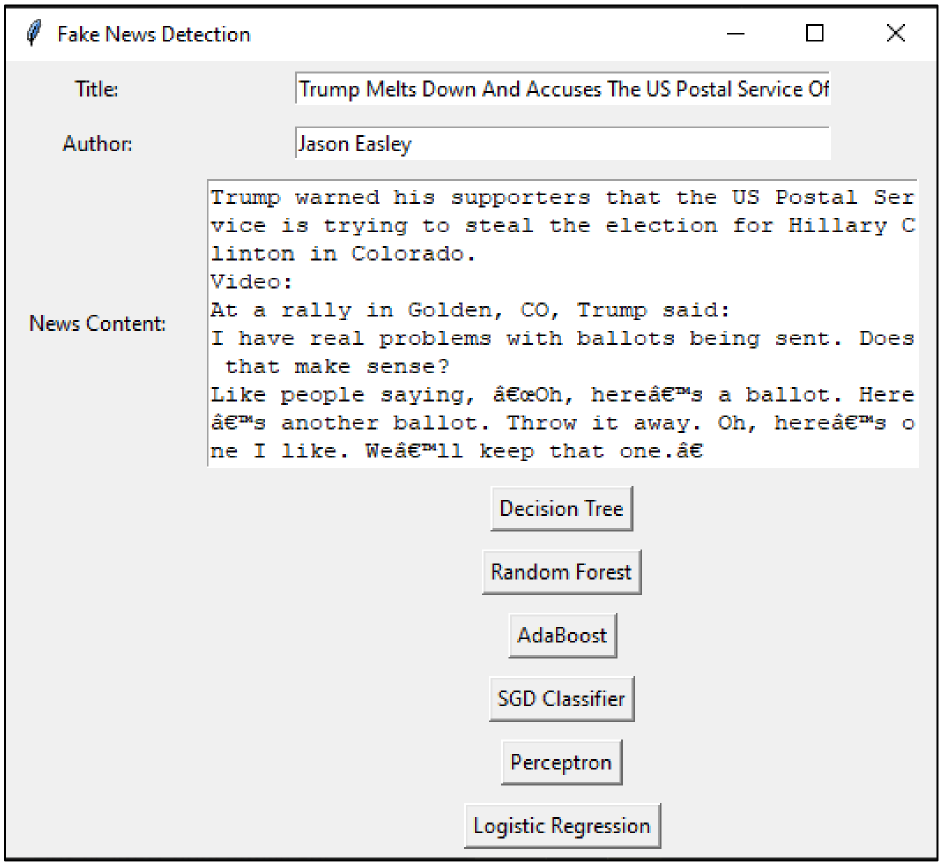
Result



```
Accuracy: 0.9615384615384616
Precision: 0.9633734939759037
Recall: 0.9596735477676428
F1 Score: 0.9615199615199616
Matthews Correlation Coefficient: 0.9230840172973191
Cohen's Kappa: 0.923077189774906
AUC-ROC: 0.9615411552030318
=====================================
classification_report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      2077
           1       0.96      0.96      0.96      2083

    accuracy                           0.96      4160
   macro avg       0.96      0.96      0.96      4160
weighted avg       0.96      0.96      0.96      4160
```

User Interface





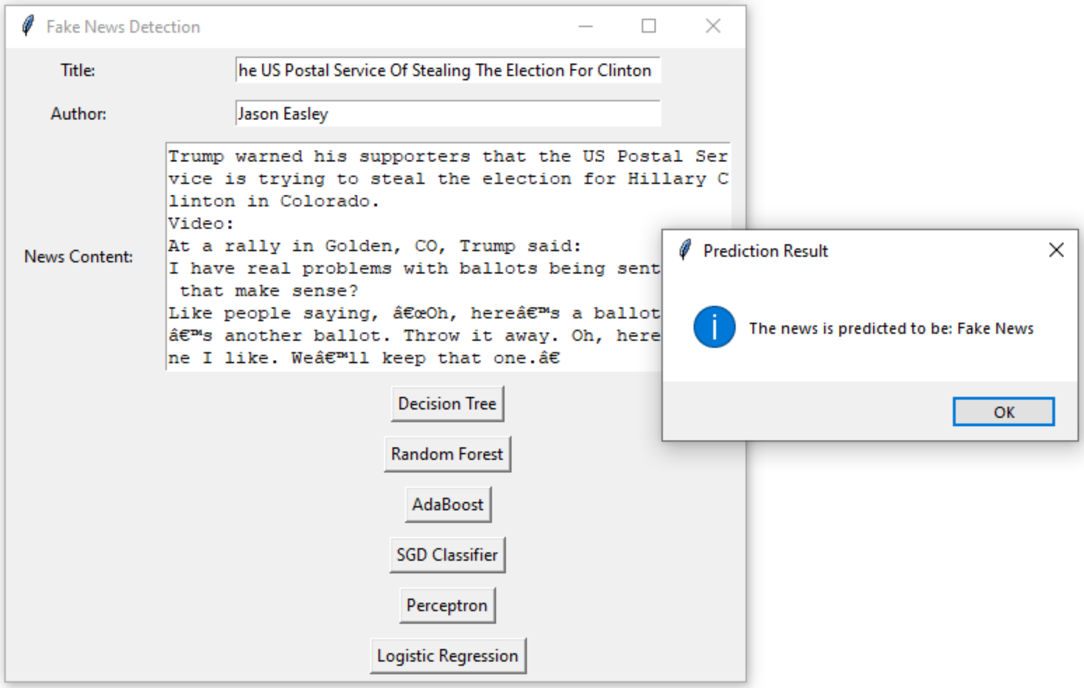
Fake News Detection

Title: Trump Melts Down And Accuses The US Postal Service Of

Author: Jason Easley

News Content:
```
Trump warned his supporters that the US Postal Ser
vice is trying to steal the election for Hillary C
linton in Colorado.
Video:
At a rally in Golden, CO, Trump said:
I have real problems with ballots being sent. Does
 that make sense?
Like people saying, â€œOh, hereâ€™s a ballot. Here
â€™s another ballot. Throw it away. Oh, hereâ€™s o
ne I like. Weâ€™ll keep that one.â€
```

Result



Fake News Detection

Title: he US Postal Service Of Stealing The Election For Clinton

Author: Jason Easley

News Content:
```
Trump warned his supporters that the US Postal Ser
vice is trying to steal the election for Hillary C
linton in Colorado.
Video:
At a rally in Golden, CO, Trump said:
I have real problems with ballots being sent
 that make sense?
Like people saying, â€œOh, hereâ€™s a ballot
â€™s another ballot. Throw it away. Oh, here
ne I like. Weâ€™ll keep that one.â€
```

Prediction Result

The news is predicted to be: Fake News

Chapter 6:

Conclusion

It is research, which clearly shows the usage success of myriad machine learning models as well as identifying and tagging fake news in social media networks using Decision Tree, Random Forest, Perceptron, AdaBoost, Stochastic Gradient Descent, and Voting Classifier. The ensemble method, especially the Voting Classifier, is quite useful to combine multiple model predictions to achieve higher accuracy and reliability. Research work also emphasizes the need for preprocessing techniques such as tokenization and TF-IDF, improving data quality and interpretability for the models. This framework would offer an effective tool that media organizations, independent fact-checkers and social media can deploy in their battles against misinformation, ultimately contributing toward the ends of conserving public trust and advancing accurate information dissemination.

Future Scope

Future work may then concentrate on the following improvement areas for the fake news detection framework:

1. Improved deep models: More complex deep learning models might be used in the fake news-detection framework, such as transformer-based architectures like BERT or GPT, to pick up more complicated linguistic patterns in the content of fake news: thus making the classification accuracy higher.

2. Multimodal Data Integration: The inclusion of other forms of data, such as images and videos, or even metadata from social posts, would make for a much more comprehensive fake news system. Fake news, after all, usually involves text merged with deceptive images or other media.

3. Real-Time Detection: Valuable in events that trigger high volumes of misinformation, like elections or health crises, it would be a real-time detection system, which analyzes and flags fake news as it spreads in social platforms.

4. Analysis of User Behavior Patterns: The study of the behavior patterns of users in sharing and engagement with the content may determine characteristics of spreaders of misinformation, so targeted interventions can be made.

5. Cross-Language and Cross-Platform Adaptability: It would provide the utility and applicability of the framework globally across cultures, linguistic contexts, and social media by aiding to effectively deal with misinformation while engaging multiple languages and adapting to various social media platforms.

This future scope introduces potential improvements that make fake news detection as accurate, scalable and adaptable to the dynamic nature of digital misinformation as it can be.

Reference

A Predictive Model for Benchmarking the Performance of Algorithms for Fake and Counterfeit News Classification in Global Networks
DOI: 10.3390/s24175817

Characterization, Classification and Detection of Fake News in Online Social Media Networks
DOI: 10.1109/MysuruCon52639.2021.9641517

A Comprehensive Review on Fake News Detection With Deep Learning
DOI: 10.1109/ACCESS.2021.3129329

Fake News detection Using Machine Learning
DOI: 10.1109/IHSH51661.2021.9378748

Elevating Fake News Detection Through Deep Neural Networks, Encoding Fused Multi-Modal Features
DOI: 10.1109/ACCESS.2024.3411926