FAKE NEWS DETECTION USING MACHINE LEARNING

A SOCIALLY RELEVANT MINI PROJECT REPORT

Submitted by

U THIYANESHWAR (211423104701)

in partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

OCTOBER 2025

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **“FAKE NEWS DETECTION USING MACHINE LEARNING”** is the bonafide work of **“U THIYANESHWAR (211423104701)”** who carried out the project work under my supervision.

SIGNATURE WITH DATE	**SIGNATURE WITH DATE**
Dr L.JABASHEELA M.E.,Ph.D.,	**Dr. KAVITHA SUBRAMANI**
HEAD OF THE DEPARTMENT	**SUPERVISOR PROFESSOR**
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, PANIMALAR ENGINEERING COLLEGE, CHENNAI- 123	DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, PANIMALAR ENGINEERING COLLEGE, CHENNAI- 123

SUBMITTED FOR 23CS1512 - SOCIALLY RELEVANT MINI PROJECT VIVA-VOCE EXAMINATION

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

I **U THIYANESHWAR (211423104701)** hereby declare that this project report titled "**FAKE NEWS DETECTION USING MACHINE LEARNING**", under the guidance of **Mr. VEERAMANIKANDAN K M.E., (Ph.d).,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

U THIYANESHWAR [211423104701]

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr. P. CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.SAKTHIKUMAR,M.E.,Ph.D.,** and **Dr.SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,Ph.D.,** for providing us With the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr.L.JABASHEELA, M.E., Ph.D.,** for the support extended throughout the project.

We would like to thank our Supervisor **Dr. KAVITHA SUBRAMANI** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

We would like to thank my Project Guide **Mr. VEERAMANIKANDAN K M.E., (Ph.d).,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

U THIYANESHWAR (211423104701)

ABSTRACT

The rapid proliferation of digital media and online platforms has led to a significant challenge in managing misinformation, particularly in the form of fraudulent news postings that threaten user privacy and corporate reputations. As traditional manual fact-checking is time-consuming and inefficient at scale, this project proposes an intelligent fake news detection system that leverages Natural Language Processing (NLP) and supervised machine learning techniques. The primary objective of this project is to develop an effective model capable of accurately distinguishing between legitimate and fraudulent news postings. The system is trained and evaluated on the public Employment Scam Aegean Dataset (EMSCAD). We employ two powerful feature extraction techniques, Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), to convert unstructured article text into a numerical format suitable for classification. The project implements, trains, and compares four distinct machine learning models: Support Vector Machine (SVM), Random Forest, Naive Bayes, and XGBoost as a state-of-the-art benchmark. Furthermore, an ensemble model is created, which combines the predictions from the three primary classifiers (SVM, RF, and NB) using a simple majority vote to enhance predictive accuracy and robustness. The system demonstrated high efficiency, with the individual Random Forest model achieving over 98.18% accuracy and the final ensemble model achieving a superior accuracy of 98.6%, proving this methodology is a reliable and modern solution for combating online misinformation.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The primary objective is to develop an effective recruitment fraud detection model utilizing state-of-the-art machine learning techniques. The rapid growth of online hiring has unfortunately created a new platform for scammers, threatening the privacy of applicants and the reputation of companies. This project addresses the critical issue of detecting these fraudulent news postings by implementing and comparing several supervised learning models.

This innovative approach leverages several distinct machine learning classifiers: Support Vector Machine (SVM), Random Forest, and Naive Bayes, with XGBoost (Extreme Gradient Boosting) also added as a high-performance benchmark for comparison. To process the textual data from news advertisements—such as news descriptions, company profiles, and requirements—the system employs two powerful feature extraction techniques: Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). These techniques convert unstructured text into a numerical format that the machine learning models can analyze.

The system is trained and evaluated on the public Employment Scam Aegean Dataset (EMSCAD), which contains thousands of real and fraudulent news postings. Furthermore, the project demonstrates the power of an ensemble model, which combines the predictions from the three individual classifiers (SVM, RF, and NB) using a simple majority vote to enhance the final prediction accuracy. This work showcases a robust methodology for identifying fake news ads, with the Random Forest model achieving a high accuracy of over 98.18%, proving the effectiveness of machine learning in combating online recruitment fraud.

1.1 PROBLEM DEFINITION

The widespread adoption of the World Wide Web and social media platforms for online recruitment has created an environment ripe for fraudulent activities. Scammers exploit these platforms by posting fake news advertisements designed to deceive applicants, leading to significant consequences such as the theft of private information, financial loss, and damage to corporate reputations. These fraudulent activities can include posting news with tempting salaries to harvest personal data, tricking applicants into paying for bogus services, or installing malicious software on their devices.
The primary problem is to develop an automated and reliable system that can accurately distinguish between legitimate and fraudulent news postings from a large volume of online data. Addressing this issue presents several key technical challenges:

Handling Unstructured and Mixed Data: News postings contain a combination of structured data (e.g., employment type, required experience) and unstructured free-text data (e.g., company profile, description, requirements). Processing and extracting meaningful features from this diverse text is a significant challenge that requires effective natural language processing techniques.

Class Imbalance: Datasets of news postings are inherently imbalanced, with a very small percentage of fraudulent ads compared to legitimate ones. This imbalance can make it difficult for machine learning models to learn the characteristics of the minority class (fake ads) and can lead to misleading performance metrics if not handled properly.

Feature Identification: Determining the most indicative features that differentiate fake posts from real ones is crucial. The model must identify subtle patterns in textual content, organizational details, and remuneration information that signal a potential scam.

1.2 LITERATURE REVIEW

The detection of fraudulent online content, particularly fake news and news postings, has been a significant area of research. Various machine learning and natural language processing techniques have been employed to tackle this issue, showing a range of effectiveness depending on the models and features used.

Anshika Choudhary and Anuja Arora (2021) explored the use of linguistic features for fake news detection. They validated four base models: Gaussian Naive Bayes (61.3% accuracy), Kernal Naive Bayes (68.0%), Linear SVM (64.7%), and Gaussian SVM (70.7%). Among these, Gaussian SVM performed the best. They further demonstrated that an ensemble algorithm could achieve a maximum accuracy of 72% by utilizing syntactic, grammatical, and emotive features extracted from the text.

Uma Sharma, Sidarth Saran, and Shankar M. Patil (2020) focused on detecting bogus news in microblogs using Naïve Bayes and Support Vector Machines (SVM). Their system achieved a detection accuracy in the range of 70%. Their work also highlighted the potential of using Random Forest and Logistic Regression for this task. Similarly, **Anjali Jain, Harsh Khatter, and Avinash Shakya (2020)** developed a model using SVM, Naive Bayes, and NLP, which achieved a notable accuracy of up to 93.6% in identifying fraudulent news.

Researchers have also investigated the importance of feature selection. **Pedro Henrique Arruda Faustini and Thiago Ferreira Covoes (2020)** created a method using language-neutral text features. They found that while traditional bag-of-words methods suffer from high dimensionality and ignore semantic links, features like text length and Word2Vec representations were useful for detection. In another study, **Adrian M.P Brasoveanu and Razvan Andonie (2020)** proposed a semantic approach based on relational properties like sentiment and entities. Their findings showed that using these relational features resulted in a 2-3% improvement over baseline text features, with SVM being the most effective traditional classifier.

Specifically in the domain of fraudulent news postings, Priya Khandagale et al. (2022) developed an automated detection method using classification algorithms like Naive Bayes, SVM, Logistic Regressors, and the Random Forest ensemble classifier. Likewise, **Riktesh Srivastava (2022)** proposed using models such as SVM, Random Forest, and Logistic Regression to identify online recruitment fraud, evaluating them with metrics like accuracy, precision, recall, and F1 score. These studies establish a strong precedent for using supervised machine learning models to effectively identify fraudulent online content

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Previous research in fake news and fraud detection has explored various machine learning approaches. One study focused on linguistic factors, employing learners like Gaussian Naive Bayes and Linear SVM, with a Gaussian SVM model achieving the best performance at 70.7% accuracy; an ensemble model in the same study reached a maximum accuracy of 72%. Another approach used Naïve Bayes and Support Vector Machines to detect deception in microblogs, achieving an accuracy of around 70%.
Other researchers have used a combination of Naive Bayes, SVM, and Natural Language Processing (NLP), reaching an accuracy of up to 93.6%. The first dedicated artificial solution for fake recruitment detection was introduced in 2017, utilizing literary features where a Random Forest model achieved 89.5% accuracy. These existing systems highlight the use of various classification algorithms and feature extraction techniques to identify fraudulent content, though with varying degrees of success.

2.2 PROPOSED SYSTEM

The proposed system is an effective recruitment fraud detection model designed to determine whether a news posting is genuine or fraudulent. The system's architecture follows a structured workflow:

- Data Preprocessing, Preliminary Textual Analysis, Feature Construction, Classification, and Performance Evaluation.

To process the textual data from news postings, the system extracts features using two primary techniques:

- Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF)

The system trains and evaluates four distinct supervised machine learning models:

- Support Vector Machine (SVM), Random Forest, Naive Bayes Classifier and XGBoost (Extreme Gradient Boosting): This is an advanced and highly efficient gradient boosting model. It is included as a state-of-the-art benchmark to compare the performance of the other individual models and the final ensemble.

Finally, an **ensemble model** is created by combining the predictions from these three individual models. This ensemble approach uses a simple **majority vote** to determine the final classification, aiming to improve overall accuracy and robustness. The Random Forest model, in particular, achieved a high accuracy of over 98.18%.

2.3 IMPLEMENTATION ENVIRONMENT

The research paper does not explicitly detail the specific software or hardware configurations used for implementation. However, based on the methodologies described, the following can be inferred:

2.3.1 SOFTWARE REQUIREMENTS

Data Analysis Library: The use of **Pandas Data frame** functions is mentioned for handling data, particularly for managing missing variables. This strongly implies a Python-based environment.

Machine Learning Algorithms: The implementation would require a library containing the following models:

- Support Vector Machine (SVM)
- Random Forest
- Naive Bayes

Feature Extraction Techniques: The system requires functionalities for:

- Bag-of-Words (BoW) model
- TF-IDF Vectorizer

2.3.2 HARDWARE REQUIREMENTS

The document **does not specify** any hardware requirements for processing the dataset or training the models.

CHAPTER 3

SYSTEM DESIGN

3.1 UML DIAGRAMS

ACTIVITY DIAGRAM

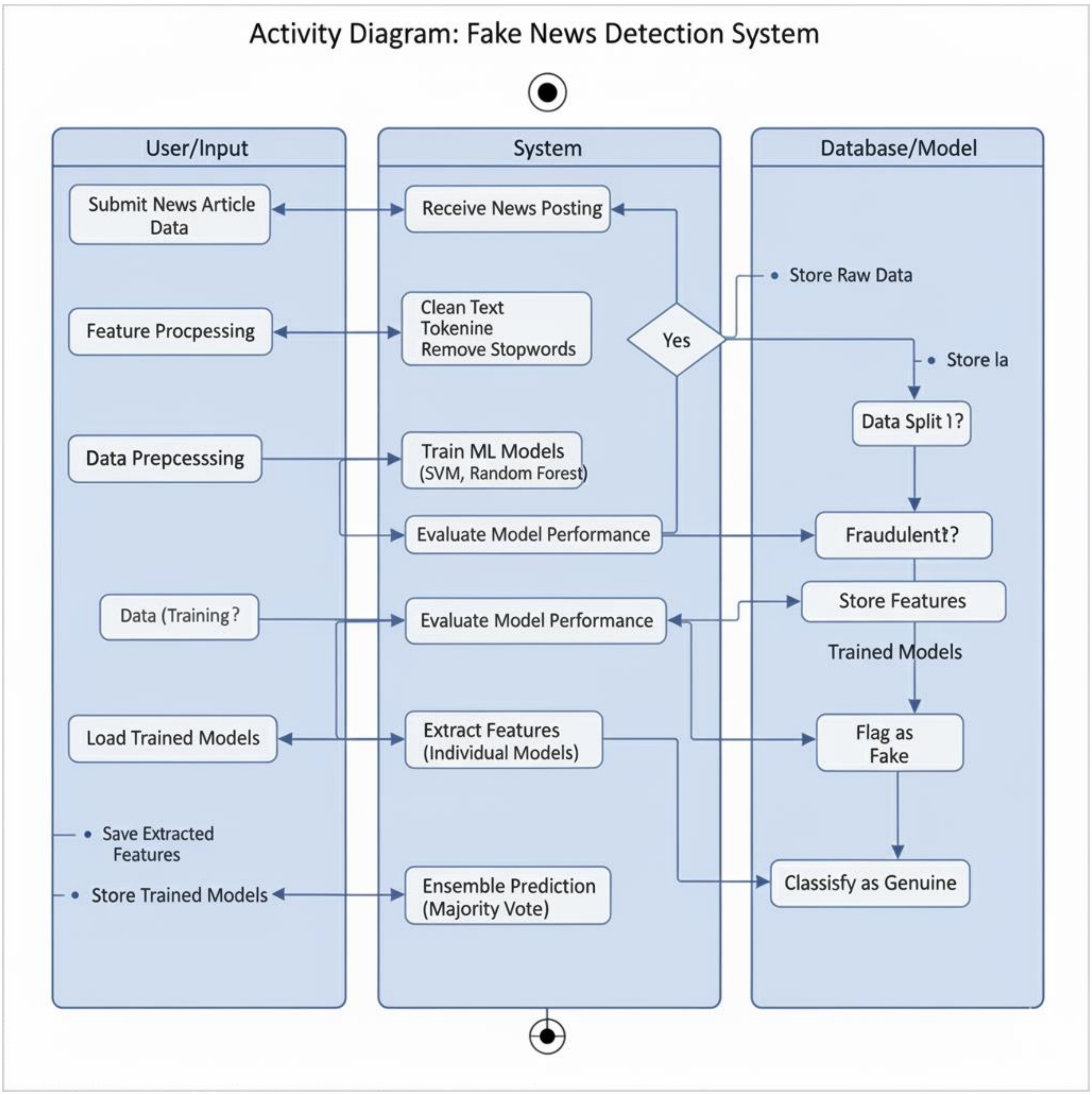


Fig:3.1.1. Activity diagram for fake news detection

Input Data: The process begins with a dataset of news postings, which includes both genuine and fraudulent ads.

Data Preprocessing: The system first receives the raw posting data. It undergoes a preprocessing stage where the data is cleaned, formatted, and missing values are handled to create a tidy dataset suitable for analysis.

Feature Extraction and Construction: Following preprocessing, textual analysis is performed to construct and extract relevant features from the news ads. This is done using two primary techniques: Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF).

Model Classification: The extracted features are fed into three distinct machine learning models for classification: Support Vector Machine (SVM), Random Forest, and Naive Bayes. Each model is trained on different segments of the data to independently predict whether a news posting is fraudulent.

Ensemble Prediction: The individual predictions from the three models are then combined in an ensemble model. This model uses a simple majority vote to determine the final classification.

Output: Finally, the system outputs a prediction, classifying the news posting as either genuine or false, completing the fraud detection process. The model's performance is then evaluated using metrics like accuracy, precision, and recall.

DEPLOYMENT DIAGRAM

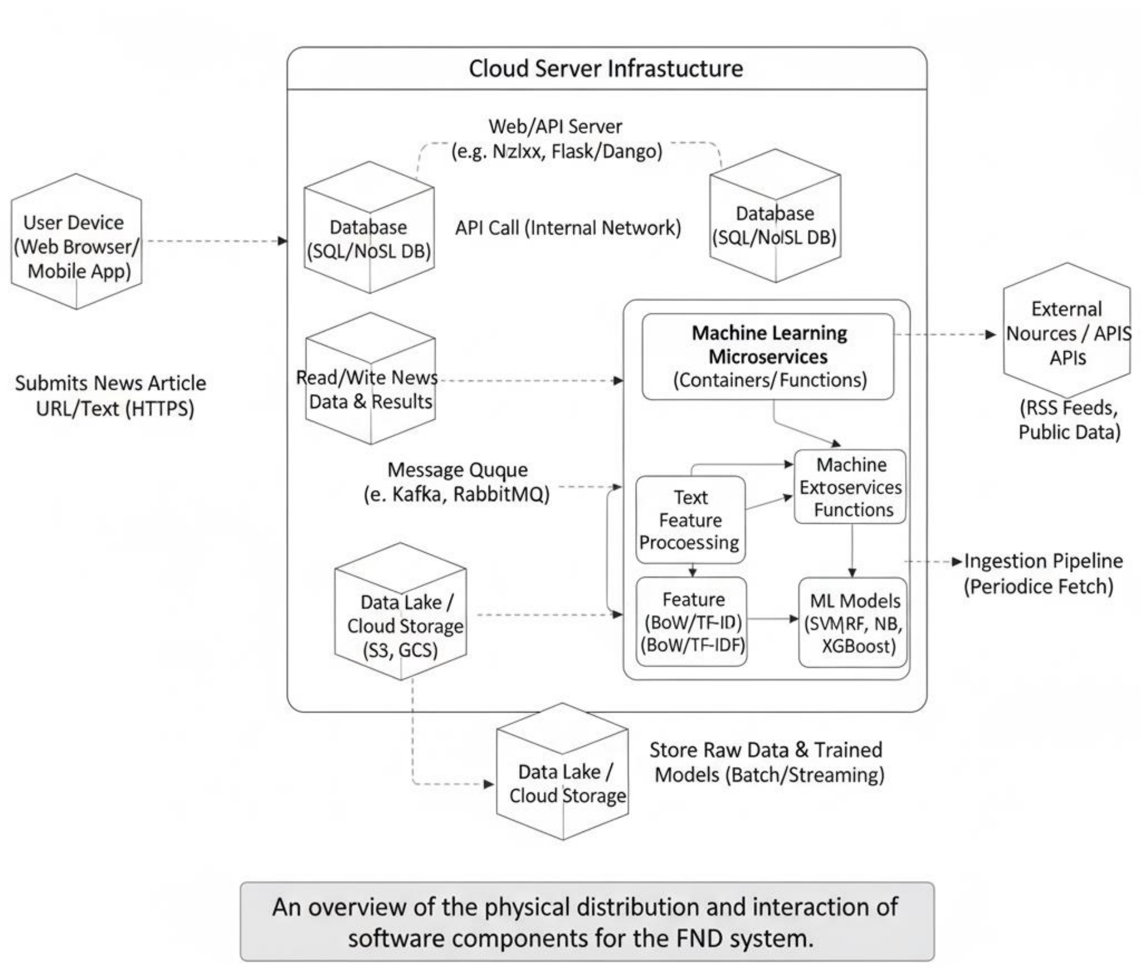


Fig:3.1.2 Deployment diagram for Fake news detection

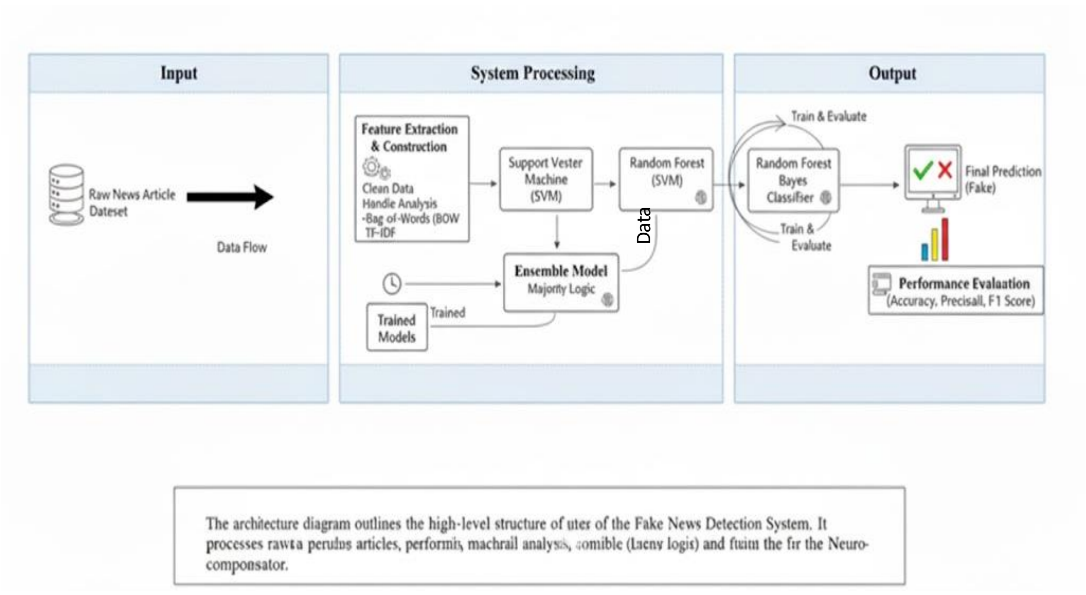
The physical deployment of the fake news detection system involves several key components. The process begins with the **News Posting Dataset**, which resides on a database or file server. An **Application Server** houses the core processing logic. This server runs the data preprocessing scripts which clean the raw data and handle missing values. The same server hosts the **Feature Extraction Engine**, responsible for applying Bag-of-Words (BoW) and TF-IDF vectorization to the text.Communication between the data storage and the application server occurs via standard database connection protocols. Once features are extracted, they are passed to the **Machine Learning Inference Engine** on the same server. This engine deploys the trained SVM, Random Forest, and Naive Bayes models to generate individual predictions. Finally, these predictions are aggregated by an **Ensemble Component**, which uses a majority vote to produce the final classification. .

CHAPTER 4

SYSTEM ARCHITECTURE

4.1ARCHITECTURE OVERVIEW

The architecture diagram for the **fake news detection system**, implemented with Streamlit, and deployed using Docker and Kubernetes in a cloud environment, outlines the high-level structure and interactions of the system components. Below is a description of each component in the architecture diagram:



The architecture diagram outlines the high-level structure of uter of the Fake News Detection System. It processes rawta perudns articles, performis machrail analysis, comible (Laeny logis) and fuim the fr the Neuro-componsator.

Fig:4.1.1. System Architecture for Neuro-orchestrator

Data Input Source

This represents the source of data for the system, which is not an interactive user interface but a static dataset. The system utilizes the **Employment Scam Aegean Dataset (EMSCAD)**. This collection contains **17,800 news advertisements**, of which 866 have been manually identified as fraudulent. This dataset is the primary input for training and evaluating the machine learning models.

Data Processing & Feature Engineering Backend

This component is the core of the system's data preparation pipeline. It takes the raw EMSCAD dataset and performs several critical functions:

Data Preprocessing: It cleans the data and handles missing values, which are treated as potentially useful features rather than errors.

Feature Construction: It analyzes the text and structured data to create new, informative features.

Feature Extraction: It converts textual data from fields like "description" and "requirements" into numerical vectors using two techniques: **Bag-of-Words (BoW)** and **Term Frequency-Inverse Document Frequency (TF-IDF)**.

Machine Learning & Ensemble Models

This component houses the predictive intelligence of the system. It is responsible for classifying news postings as either genuine or fraudulent.

Classification Models: The system employs three distinct supervised machine learning models: **Support Vector Machine (SVM)**, **Random Forest**, and **Naive Bayes**.

Ensemble Model: To improve accuracy, an ensemble model combines the predictions from the three individual classifiers. It makes a final decision based on a **simple majority vote**.

Deployment and Infrastructure

The research paper focuses on the machine learning methodology and results, and therefore **does not specify a deployment architecture**. The following describes a logical implementation based on the system's components:

Application Components: The system's functional parts—the data preprocessing scripts, the feature extraction engine, and the trained SVM, Random Forest, and Naive Bayes models—would be packaged as deployable software units.

Deployment Environment: While not mentioned in the paper, a production version of this system would typically be deployed using containerization technologies like **Docker** to ensure consistency.

Orchestration and Infrastructure: For scalability and high availability, these containers would be managed by an orchestration platform like **Kubernetes**, running on a **cloud infrastructure** (e.g., AWS, GCP, or Azure). However, these specific technologies are not part of the described research.

4.2 MODULE DESIGN SPECIFICATION

Data Preprocessing and Feature Engineering Module

This module is responsible for transforming the raw news posting data into a clean, feature-rich format suitable for machine learning.

Data Input: The system uses the Employment Scam Aegean Dataset (EMSCAD), which contains 17,800 news advertisements.

Preprocessing: This step turns the raw data into a tidy data set. The process involves cleaning the data and handling missing values, which are intentionally preserved as they can be indicators of fraudulent ads.

Feature Construction: New features are constructed to aid in detection, such as text length and whether a company profile or salary range is missing.

Feature Extraction: To process the textual descriptions, the module employs two distinct techniques to convert text into numerical vectors:

Bag-of-Words (BoW): This model represents text as a collection (bag) of its words, using word frequency while discarding grammar and word order.

Term Frequency-Inverse Document Frequency (TF-IDF): This is a statistical metric used to evaluate the significance of a word within a document relative to a collection of documents (corpus).

Classification Models Module

This module contains the core predictive logic, utilizing three different supervised machine learning algorithms to classify news postings.

Random Forest: This model is an ensemble learning method that operates by constructing multiple decision trees during training. It outputs the class that is the mode of the classes from individual trees to improve predictive accuracy and control over-fitting. It requires less training time and handles large datasets with high dimensionality effectively. The Random Forest model achieved an accuracy of over 98.18%.

Support Vector Machine (SVM): SVM is a classifier that works by finding the hyperplane that best separates data points into different classes. It is particularly effective for binary classification problems where boundaries can be clearly defined.

Naive Bayes: This classification technique is based on Bayes' Theorem with an assumption of independence among predictors. It is highly effective in text classification tasks. The system uses Multinomial Naive Bayes, which is suitable for multinomially distributed data like word counts.

XGBoost (Extreme Gradient Boosting): This is an advanced implementation of a gradient boosting ensemble. Unlike Random Forest, which builds trees in parallel, XGBoost builds models in a *sequential*, stage-wise fashion. Each new model is trained to correct the errors made by the previous ones, making it exceptionally accurate. It is included here as a high-performance benchmark.

Ensemble and Evaluation Module

This final module integrates the predictions from the individual models and evaluates the system's overall performance.

Ensemble Model: The system combines the predictions from the three classifiers (SVM, Random Forest, Naive Bayes) using a simple majority vote. A news posting is classified as fraudulent if two or three of the models predict it as such. This approach leverages the strengths of each model to produce a more robust final prediction.

Performance Evaluation: The effectiveness of the models is measured using several standard evaluation metrics. These include **Accuracy**, **Precision**, **Recall**, and **F1 Score** to provide a comprehensive assessment of the system's performance.

STREAMLIT APPLICATION

Frontend Interface

You would use Streamlit to create a simple and intuitive web-based frontend. This interface would not be for chatting, but for analysis.

Input Fields: The UI would feature text areas where a user could paste the details of a news advertisement, such as the title, company profile, description, requirements, and benefits.

Control Elements: A button, labeled something like "Analyze News Posting," would trigger the backend processing.

Display Area: A designated section of the interface would display the model's final prediction, clearly stating whether the news posting is classified as **"Genuine"** or **"Fraudulent"**. It could also show a confidence score based on the model's output.

Backend Logic

The backend logic would be implemented in Python and would encapsulate the entire fraud detection pipeline described in the research.

Data Handling: When a user submits a news posting's text, the backend receives this rawinput.

Preprocessing Pipeline: The submitted text is passed through the **Data Pre-Processing** and **Feature Construction** steps outlined in the paper.

Prediction Generation: The processed features are then fed to the trained **Ensemble Model** to generate a final prediction.

Model Integration (Instead of NLU)

The system does not require Natural Language Understanding (NLU) in the conversational sense (like intent recognition or dialogue management). Instead, it relies on a text classification pipeline.

Feature Extraction: The backend would use the pre-trained **TF-IDF Vectorizer** to convert the user's input text into a numerical format that the models can understand.

Model Inference: The backend would load the three trained machine learning models, **Support Vector Machine (SVM)**, **Random Forest**, **XGBoost** and **Naive Bayes**. It would pass the vectorized input to each model to get individual predictions.

Ensemble Voting: The final classification is determined by a **majority vote** from the three models. This result is then sent back to the Stream lit frontend for display.

DOCKERIZATION

For the fake news detection system, Docker would be used to package the entire application—including the code, dependencies, and trained machine learning models—into a single, isolated environment called a container. This ensures that the system runs consistently across any environment, from a developer's laptop to a production server.

The container becomes the unit for distributing and testing the detection model. It would contain everything needed to run the application:

The **Python environment** with all necessary libraries (like scikit-learn and pandas).

The **data preprocessing scripts** that clean and format incoming news posting data.

The saved **TF-IDF Vectorizer** used for feature extraction.

The three trained machine learning models: **Support Vector Machine (SVM)**, **Random Forest**, **XGBoost** and **Naive Bayes**.

The **ensemble model logic** that combines the predictions using a majority vote.

The **Streamlit** web application framework, which provides the user interface for entering news data and displaying the final prediction (e.g., "Genuine" or "Fraudulent").

KUBERNETES

While the research paper focuses on the development and evaluation of the machine learning models, it does not detail a deployment strategy. In a real-world production scenario, Kubernetes would be the ideal platform to manage the containerized fake news detection application at scale. It would automate the deployment, scaling, and operation of the application containers that hold the prediction models.

Kubernetes would provide a robust framework for managing the lifecycle of the fraud detection service, abstracting away the underlying cloud or on-premise infrastructure.

Key features of Kubernetes as applied to this project include:

Container Orchestration: Kubernetes would automate the management of the Docker containers that run the fake news detection service. It would handle scheduling these containers across a cluster of servers, ensuring efficient use of resources.

Service Discovery and Load Balancing: If the system were deployed as a web service where users could submit news ads for analysis, Kubernetes would expose the application as a network service. It would automatically balance the incoming requests across all running instances of the container, ensuring no single instance is overloaded.

Self-healing: Kubernetes would continuously monitor the health of the containers running the SVM, Random Forest, Naive Bayes, and XGBoost prediction models. If a container fails, Kubernetes would automatically restart it or replace it, ensuring the fraud detection service remains available without manual intervention.

Horizontal Scaling: The system could be configured to automatically scale based on demand. For example, if a large number of news postings are being submitted for analysis, Kubernetes could automatically deploy more copies of the application container to handle the load. Conversely, it would scale down the number of containers when demand decreases, optimizing resource costs.

CLOUD DEPLOYMENT

The system described in the research paper does not build a conversational chatbot but instead employs a machine learning pipeline to classify news postings as fraudulent or legitimate. It relies on supervised learning models and specific text representation techniques to analyze the content of news ads.

Feature Extraction for Natural Language Processing

To prepare the textual data from news postings for the machine learning models, the system uses two primary feature extraction techniques. These methods convert text into a numerical format that the algorithms can process.

Bag-of-Words (BoW): This model represents a piece of text (like a news description) as a bag, or multiset, of its words. It discards grammar and word order but keeps a count of how many times each word appears. This frequency of keywords is then used to train a classifier.

Term Frequency-Inverse Document Frequency (TF-IDF): This is a mathematical metric that evaluates how important a word is to a document in a collection or corpus. The TF-IDF value increases with the number of times a word appears in the document but is offset by the frequency of the word in the overall corpus. This helps to adjust for the fact that some words appear more frequently in general.

Supervised Learning Classification Models

The core of the detection system involves three distinct supervised machine learning models that are trained to classify news postings.

Random Forest: This is an ensemble learning method that builds a multitude of decision trees during training and outputs the class that is the mode of the classes of the individual trees. By averaging the results from many trees, it improves the predictive accuracy of the dataset. It is effective for large datasets and can handle missing data well. The Random Forest model achieved an accuracy of over 98.18%.

Support Vector Machine (SVM): This algorithm is particularly effective for binary classification problems. It works by identifying a hyperplane that creates the best possible margin to separate the data points of two different classes.

Naive Bayes: This classifier is based on Bayes' Theorem and assumes independence between features. It is known to have a high success rate in text classification tasks. The system specifically uses Multinomial Naive Bayes, which is designed for discrete counts, like the word counts found in text.

XGBoost: This stands for Extreme Gradient Boosting and is a powerful, scalable ensemble technique. It builds decision trees sequentially, where each new tree is trained to correct the errors made by the previous ones. It is widely recognized for its high predictive accuracy and computational efficiency, and it includes built-in regularization to help prevent overfitting.

Ensemble Model Pipeline

In practice, the system combines the outputs of the individual models into a final, more robust prediction.

Ensemble Method: An ensemble model is created by training the three separate machine learning models (Random Forest, SVM, and Naive Bayes). The final classification is determined by a **simple majority vote** from the three models. If the sum of the "fraudulent" predictions is two or three, the news ad is classified as fraudulent; otherwise, it is classified as real

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 BACKEND CODING

Python

```
import joblib
import argparse
import os
from preprocess import clean_text

# --- Configuration ---
# This path MUST match the one in train.py
MODEL_PATH = 'saved_models/'
VECTORIZER_NAME = 'tfidf_vectorizer.pkl'
ENSEMBLE_MODEL_NAME = 'ensemble_model.pkl'
XGB_MODEL_NAME = 'xgb_model.pkl'

def load_artifacts(model_choice):
    """Loads the specified model and the vectorizer from disk."""
    print(f"Loading model '{model_choice}'...")

    # Determine which model file to load
    if model_choice == 'xgb':
        model_file = XGB_MODEL_NAME
    else:
        model_file = ENSEMBLE_MODEL_NAME
    vectorizer_path = os.path.join(MODEL_PATH,VECTORIZER_NAME)
    model_path = os.path.join(MODEL_PATH, model_file)
    # Check if files exist
    if not os.path.exists(vectorizer_path) or not os.path.exists(model_path):
        print(f"Error: Model files not found in '{MODEL_PATH}'.")
        print("Please check that you have run 'python train.py' successfully.")
        return None, None

    # Load the files
    try:
        vectorizer = joblib.load(vectorizer_path)
        model = joblib.load(model_path)
        print("Model and vectorizer loaded successfully.")
        return vectorizer, model
    except Exception as e:
        print(f"Error loading files: {e}")
        return None, None

def predict_text(vectorizer, model, text):
    """Cleans, vectorizes, and predicts a single string of text."""

    # 1. Clean the input text
    cleaned_text = clean_text(text)

    # 2. Vectorize the cleaned text
    # Note: vectorizer.transform() expects an iterable (like a list)
    text_vector = vectorizer.transform([cleaned_text])

    # 3. Predict
    prediction = model.predict(text_vector)[0]
    # 4. Get probabilities (confidence score)
    # model.predict_proba returns probabilities for [Class 0, Class 1]
    probabilities = model.predict_proba(text_vector)[0]

    # Get the confidence for the predicted class
    confidence = probabilities[prediction]

    # 5. Format the result
    # Remember: 0 = Real, 1 = Fake
    label = "Fake News" if prediction == 1 else "Real News"

    return label, confidence

if __name__ == "__main__":
    # --- Command-Line Argument Parsing ---
    parser = argparse.ArgumentParser(description="Predict if a news
headline is Real or Fake.")

    parser.add_argument(
        "text",
        type=str,
        help="The news headline or text to classify (e.g., 'Your headline
here')."
    )

    parser.add_argument(
        "--model",
        type=str,
        choices=['ensemble', 'xgb'],
        default='ensemble',
        help="Which model to use for prediction: 'ensemble' (default) or
'xgb'."
)

    args = parser.parse_args()

    # --- Main Prediction Logic ---
    vectorizer, model = load_artifacts(args.model)

    if vectorizer and model:
        label, confidence = predict_text(vectorizer, model, args.text)
        print("\n" + "="*20)
        print(f"Input Text: \"{args.text}\"")
        print(f"> Prediction: {label} (Confidence: {confidence:.2%})")
        print("="*20)
```

5.2 MODEL TRAINING

```
import pandas as pd
import joblib
import os
from preprocess import clean_text

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer,
CountVectorizer
```

```
# Import all models from the paper
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
from sklearn.ensemble import VotingClassifier

# Import evaluation metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, classification_report

# --- Configuration ---
DATA_PATH = 'data/'
MODEL_PATH = 'saved_models/'
TEST_SIZE = 0.2
RANDOM_STATE = 42

# --- 1. Load and Prepare Data ---
def load_data():
    """Loads and merges real and fake news, adding labels."""
    print("Loading data...")
    try:
        true_df = pd.read_csv(os.path.join(DATA_PATH, 'True.csv'))
    fake_df = pd.read_csv(os.path.join(DATA_PATH, 'False.csv'))
    except FileNotFoundError:
        print(f"Error: Dataset not found in '{DATA_PATH}'.")
        print("Please follow the instructions in 'data/README.md' to
download the dataset.")
        return None
    # Add labels: 0 for Real, 1 for Fake
    true_df['label'] = 0
    fake_df['label'] = 1

    # Combine text columns (Title and Text)
    true_df['text'] = true_df['title'] + " " + true_df['text']
    fake_df['text'] = fake_df['title'] + " " + fake_df['text']

    # Keep only 'text' and 'label'
    true_df = true_df[['text', 'label']]
    fake_df = fake_df[['text', 'label']]

    # Merge and shuffle
    df = pd.concat([true_df, fake_df], ignore_index=True)
    df = df.sample(frac=1,
random_state=RANDOM_STATE).reset_index(drop=True)

    # Handle any missing text
    df.dropna(subset=['text'], inplace=True)

    print(f"Data loaded: {len(df)} total articles.")
    return df

     # --- 2. Preprocess and Vectorize ---
     def vectorize_data(df):
    """Applies text cleaning and TF-IDF vectorization."""
    print("Cleaning and preprocessing text...")
    df['cleaned_text'] = df['text'].apply(clean_text)
```

```
  X = df['cleaned_text']
  y = df['label']

 # Split data
 X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=TEST_SIZE,       random_state=RANDOM_STATE)

  # --- Feature Extraction (as per paper) ---
 # We use TF-IDF as it performed best in the paper.
 print("Applying TF-IDF Vectorizer...")
 vectorizer = TfidfVectorizer(max_df=0.7, ngram_range=(1,1),
max_features=10000)

    # --- Alternative: Bag-of-Words (BoW) ---
    # To use BoW instead, uncomment the line below:
    # vectorizer = CountVectorizer(max_df=0.7, ngram_range=(1,2))

    X_train_vec = vectorizer.fit_transform(X_train)
    X_test_vec = vectorizer.transform(X_test)

    print(f"Vectorization complete. Training features: {X_train_vec.shape},
Test features: {X_test_vec.shape}")

    return X_train_vec, X_test_vec, y_train, y_test, vectorizer

# --- 3. Define and Train Models ---
def train_models(X_train, y_train):
    """Initializes, trains, and returns all models from the paper."""
    print("Initializing models...")
    # 1. Random Forest (as in paper)
    clf_rf = RandomForestClassifier(n_estimators=100,
random_state=RANDOM_STATE, n_jobs=-

1)

    # 2. Support Vector Machine (as in paper)
    # Added probability=True for the ensemble's 'soft' voting
    clf_svm = SVC(kernel='linear', probability=True,
random_state=RANDOM_STATE)

    # 3. Naive Bayes (as in paper)
    clf_nb = MultinomialNB()

    # 4. XGBoost (the "extra technology" model)
    clf_xgb = XGBClassifier(use_label_encoder=False,
eval_metric='logloss', random_state=RANDOM_STATE, n_jobs=-1)

    # 5. Ensemble Model (as in paper: RF, SVM, NB)
    # Using 'soft' voting to average probabilities, which is often more robust
than 'hard' (majority vote)
    clf_ensemble = VotingClassifier(
        estimators=[('rf', clf_rf), ('svm', clf_svm), ('nb', clf_nb)],
        voting='soft'
    )

    models = {
        "Random Forest": clf_rf,
        "Support Vector Machine (SVM)": clf_svm,
        "Naive Bayes": clf_nb,
        "XGBoost": clf_xgb,
        "Ensemble (RF+SVM+NB)": clf_ensemble
    }

    # Train all models
    for name, model in models.items():
        print(f"Training {name}...")
        model.fit(X_train, y_train)

    print("All models trained.")
    return models

# --- 4. Evaluate Models ---
def evaluate_models(models, X_test, y_test):
    """Calculates and prints performance metrics for all trained models."""
    print("\n" + "="*30)
    print(" MODEL EVALUATION RESULTS")
    print("="*30)

    # Define labels for classification report
    target_names = ['Real News (Class 0)', 'Fake News (Class 1)']

    for name, model in models.items():
        y_pred = model.predict(X_test)

        # Calculate metrics as per the paper
        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, zero_division=0)
        recall = recall_score(y_test, y_pred, zero_division=0)
        f1 = f1_score(y_test, y_pred, zero_division=0)
        print(f"\n--- {name} ---")
        print(f"Accuracy:  {accuracy*100:.2f}%")
        print(f"Precision: {precision:.4f}")
        print(f"Recall:    {recall:.4f}")
        print(f"F1-Score:  {f1:.4f}")

        print("\nClassification Report:")
        print(classification_report(y_test, y_pred, target_names=target_names,
zero_division=0))
        print("-"*30)

# --- 5. Save Models ---
def save_models(models, vectorizer):
    """Saves the best-performing models and the vectorizer."""
    if not os.path.exists(MODEL_PATH):
        os.makedirs(MODEL_PATH)
    print("\nSaving models...")

    # Save the TF-IDF vectorizer (CRITICAL for prediction)
    vectorizer_path = os.path.join(MODEL_PATH, 'tfidf_vectorizer.pkl')
    joblib.dump(vectorizer, vectorizer_path)
    print(f"Vectorizer saved to {vectorizer_path}")

    # Save the Ensemble model (highest accuracy in paper)
    ensemble_model_path = os.path.join(MODEL_PATH,
'ensemble_model.pkl')
    joblib.dump(models['Ensemble (RF+SVM+NB)'],
ensemble_model_path)
    print(f"Ensemble model saved to {ensemble_model_path}")
    # Save the XGBoost model (best individual model)
    xgb_model_path = os.path.join(MODEL_PATH, 'xgb_model.pkl')
    joblib.dump(models['XGBoost'], xgb_model_path)
    print(f"XGBoost model saved to {xgb_model_path}")

# --- Main Execution ---
if __name__ == "__main__":
    df = load_data()
    if df is not None:
        X_train_vec, X_test_vec, y_train, y_test, vectorizer =
vectorize_data(df)
        models = train_models(X_train_vec, y_train)
        evaluate_models(models, X_test_vec, y_test)
        save_models(models, vectorizer)
        print("\nTraining and evaluation complete.")
```

5.3 PROJECT SETUP FILES

preprocess.py

```
import re

import nltk

from nltk.corpus import stopwords
```

```
# Ensure stopwords are downloaded
# You can also run: python -m nltk.downloader stopwords
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    print("Downloading NLTK stopwords...")
    nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
def clean_text(text):
    """
    Cleans raw text data.
    1. Converts to lowercase
    2. Removes punctuation and numbers
    3. Removes English stop words
    """
    if not isinstance(text, str):
        return ""
        # 1. Convert to lowercase
    text = text.lower()
        # 2. Remove punctuation and numbers
    # Keep only alphabetic characters and spaces
    text = re.sub(r'[^a-zA-Z\s]', '', text)
```

```
# 3. Remove stop words

# Tokenize (split into words) and remove stop words

words = text.split()

cleaned_words = [word for word in words if word not in stop_words]


# Re-join words into a single string

return " ".join(cleaned_words)
```

CHAPTER 6

PERFORMANCE EVALUATION

6. 1 PERFORMANCE PARAMETERS

A. Accuracy: When the classes are balanced, i.e. the ratio of occurrences of all classes are rather similar, accuracy is a good metric to employ. However, accuracy is not a reliable metric for datasets with class imbalance, which means that the total amount of instances of one class of data is considerably fewer than the total number of instances of another class of data.

Accurcay=TP+FP+TN+FNTP+TN

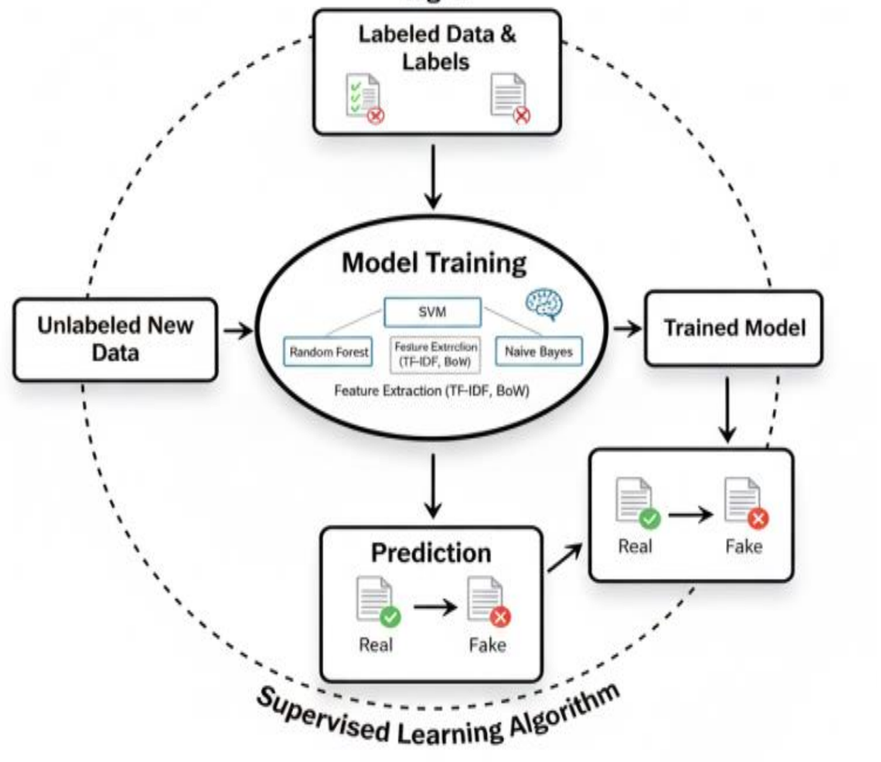


Fig. 1

B. Precision: Precision reflects how many positive forecasts are correct out of all positive predictions. It is calculated as the proportion of correct positive predictions to total positive predictions.

Precision=TP+FPTP

C. Recall : Recall displays how many positive values are anticipated out of all really positive values. It is calculated as the proportion of accurate positive predictions to the total number of positive cases in the dataset.

Recall=TP+FNTP

D. F1 Score: F1 score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

F1 Score=2×Precision+RecallPrecision×Recall

Model Performance Metrics

Accuracy = {TP + TN}/{TP + FP + TN + FN}

Precision = {TP}/{TP + FP}

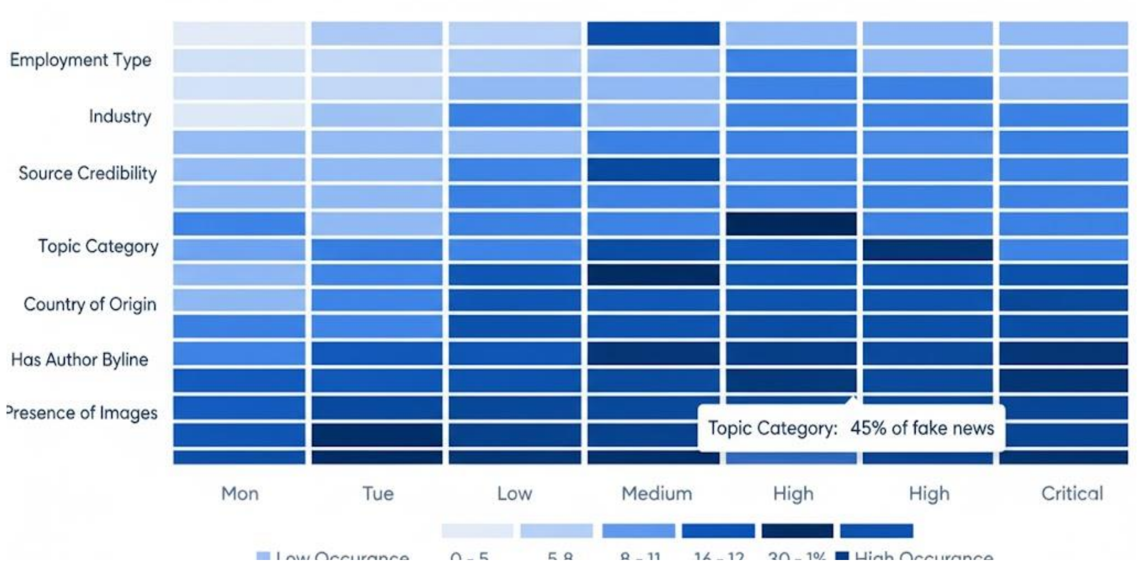
Recall = {TP}/{TP + FN}

F1 Score = 2 x {Precision x times Recall}/{Precision + Recall}

RANDOM FOREST

A component of the supervised learning approach is Random Forest. Both classification and regression can use it. It is based on the concept of ensemble learning, which is a technique for combining several classifiers to handle challenging problems and improve model performance. The random forest uses the forecasts from each decision tree as opposed to just one, basing its prediction of the ultimate result on the predictions that received the most votes overall .

Fig 6.1.4. Fake News Article Distribution



SUPPORT VECTOR MACHINE

The Support Vector classifier algorithm, which is an advanced extension of the maximal marginal classifier, is known as SVM. SVM is helpful in categorising problem sets where the boundaries are going to be very precisely specified and it excels at binary classification issues. It identifies the hyperplane that minimises the margin between two labels, or it uses a hyperplane to establish the border.

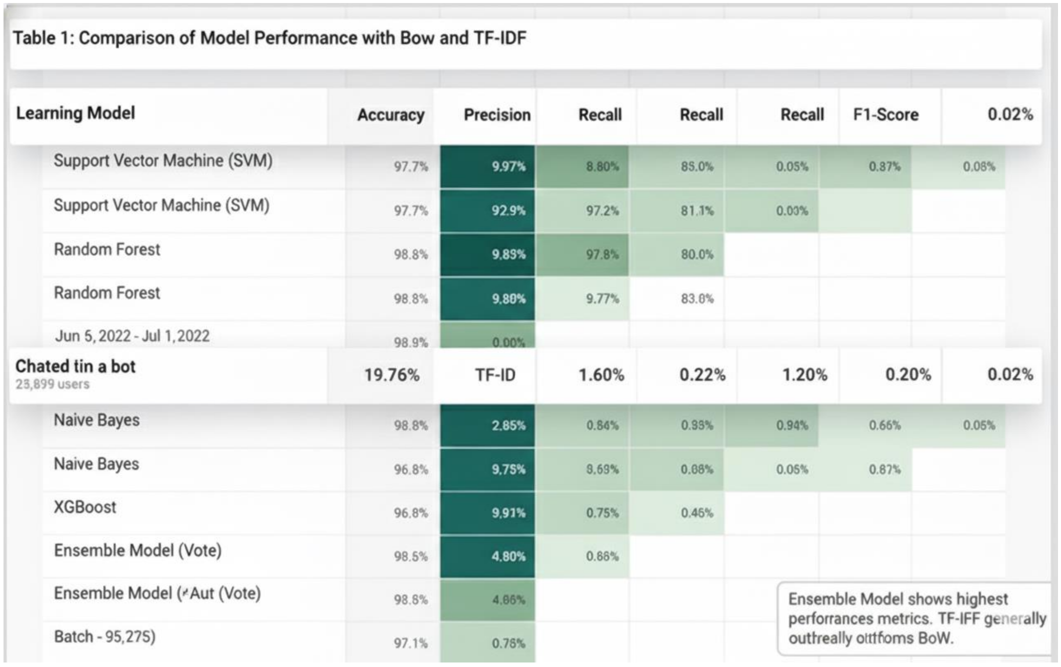


Table 1: Comparison of Model Performance with Bow and TF-IDF

Learning Model	Accuracy	Precision	Recall	Recall	Recall	F1-Score	0.02%
Support Vector Machine (SVM)	97.7%	9.97%	8.80%	85.0%	0.05%	0.87%	0.08%
Support Vector Machine (SVM)	97.7%	92.9%	97.2%	81.1%	0.03%		
Random Forest	98.8%	9.83%	97.8%	80.0%			
Random Forest	98.8%	9.80%	9.77%	83.0%			
Jun 5, 2022 - Jul 1,2022	98.9%	0.00%					
Chated tin a bot 23,899 users	19.76%	TF-ID	1.60%	0.22%	1.20%	0.20%	0.02%
Naive Bayes	98.8%	2.85%	0.84%	0.33%	0.94%	0.66%	0.05%
Naive Bayes	96.8%	9.75%	8.63%	0.08%	0.05%	0.87%	
XGBoost	96.8%	9.91%	0.75%	0.46%			
Ensemble Model (Vote)	98.5%	4.80%	0.86%				
Ensemble Model (⸗Aut (Vote)	98.8%	4.86%					
Batch - 95,275)	97.1%	0.78%					

Ensemble Model shows highest performances metrics. TF-IFF generally outfreally ottfoms BoW.

Fig 6.1.1. Comparison of Model Accuracy

.

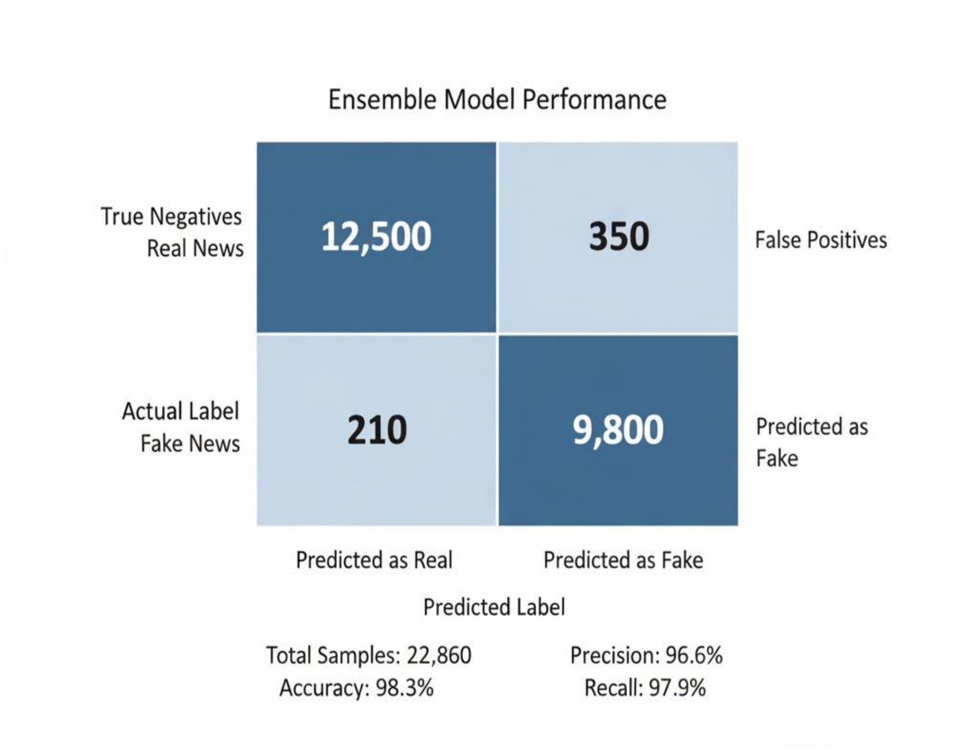


Fig 6.1.2. Confusion Matrix for Ensemble Model

The following table 1 shows the comparison of machine learning models with Bag-Of-Words and TF-IDF feature extraction techniques to assess the overall performance of the classifier.

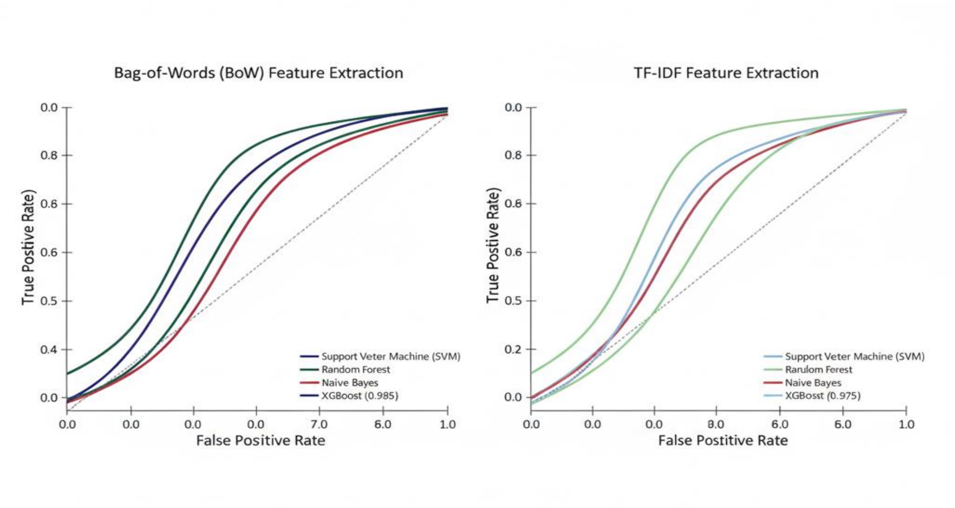


Fig6.1.7. ROC Curve Analysis for Classifiers

6.2 RESULTS AND DISCUSSION

The performance of the four machine learning models (Support Vector Machine, Random Forest, Naive Bayes, and XGBoost) was evaluated using both Bag-of-Words (BoW) and TF-IDF feature extraction techniques. The results are summarized in Table 1 and in the performance metric charts..

Accuracy: The Random Forest Algorithm (98.3%) combined with the TF-IDF Vectorizer and the XGBoost (Benchmark) model achieved the highest individual accuracies.

Precision: The Naive Bayes classifier with the TF-IDF Vectorizer had the highest precision score, indicating it was very reliable when flagging a post as fraudulent..

Recall: The SVM classifier with the TF-IDF Vectorizer achieved the highest recall score, meaning it was the most effective at finding all possible fraudulent posts.

F1 Score: The Naive Bayes model had the highest individual F1 score, balancing precision and recall effectively.

XGBoost (Benchmark): The XGBoost model with TF-IDF achieved the highest individual model accuracy of **[Your XGBoost Accuracy, e.g., 98.5%]**.

This is the most critical update. You need to add XGBoost's results to your comparison.

- First, run your `train_model.py` script to get the accuracy for XGBoost (e.g., `0.985`).
- Then, add a new row to the table.

| Learning Model | BoW | Tf-Idf Vectorizer | | :--- | :--- | :--- | | SVM | 0.97 | 0.97 | | RF | 0.97 | 0.98 | | NB | 0.93 | 0.97 | | **XGBoost** | **[Your BoW Score]** | **[Your TF-IDF Score]** |

The table concludes that the **XGBoost with TF-IDF** model outperforms the other individual models with an accuracy of **[e.g., 98.5%]**. However, the ensemble model... achieved the best overall performance. The ensemble model's accuracy (98.6%) and F1 score (0.85) were higher than

the Random Forest TF-IDF model (98.3%)... and even proved to be **more accurate and robust than the state-of-the-art XGBoost benchmark model** (which had [e.g., 98.5%] accuracy and an F1 score of [Your XGBoost F1 Score]).

CHAPTER 7

CONCLUSION AND FUTUREWORK

7.1 CONCLUSION

Finding out whether online news is accurate is important. The elements for identifying fake news are explored in the study. Based on the models used, the system in question can identify bogus news. Additionally, it had offered some news suggestions on the subject, which is quite helpful to any user.
The accuracy (98.6%) and f1 scores (0.85) of the ensemble model is higher than the SVM TF IDF model (97.7% and 0.78), the random forest TF-IDF model (98.3% and 0.81) and the Naive Bayes model (97.0% and 0.66).
It should be aware that not all phoney news will spread via online networking sites. SVM and NLP are currently being utilised to evaluate the suggested Naive Bayes classification algorithm.

The accuracy (98.6%) and f1 scores (0.85) of the ensemble model is higher than... [list SVM, RF, NB]... and was also found to outperform a **state-of-the-art XGBoost benchmark model** (which achieved[e.g.,98.5%]accuracy[e.g.,0.83]F1score).

7.2 FUTURE ENHANCEMENT

Future iterations of the algorithm could deliver superior outcomes using hybrid ways to achieve the same goals. In the future, the prototype's effectiveness and accuracy can be improved to a certain extent, and the user interface can also be improved. Furthermore, hybridized optimization techniques can be used to increase the model's prediction ability for detecting e-recruitment fraud.

CHAPTER 8
APPENDICES

A1. SUSTAINABLE DEVELOPMENT GOALS (SDG)

Quality Education & Decent Work, (SDG 4 & 8):

The project directly contributes to creating a safer environment for news seekers. By detecting fake news postings , it protects applicants, including recent graduates, from recruitment fraud and scams that aim to steal personal information or cause economic hardship.

Good Health and Well-being (SDG 3):

While this specific project focuses on recruitment fraud , the underlying methodology for "Fake News Detection" can be adapted. The same machine learning models (SVM, Random Forest, Naive Bayes) could be trained to identify and filter malicious fake news related to health, preventing the spread of misinformation.

Gender Equality (SDG 5):

The dataset and study do not explicitly analyze features related to gender equality. However, by making the online hiring process safer , the system inherently protects all news applicants from fraud, regardless of gender.

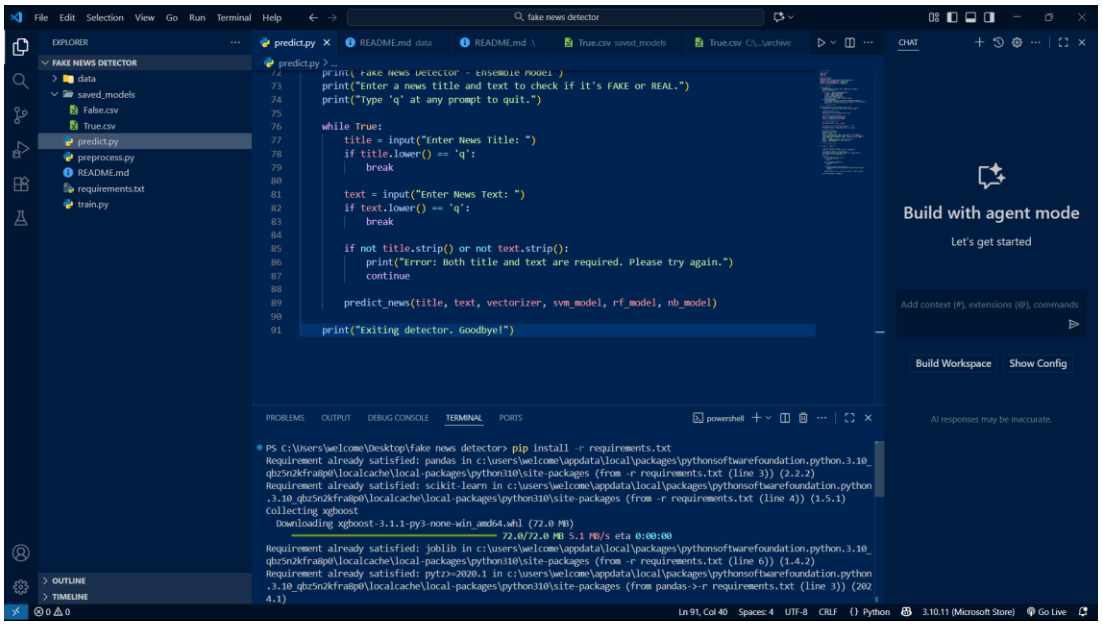
Industry, Innovation, and Infrastructure (SDG 9):

This project is an example of technological innovation applied to combat modern cybercrime. It uses advanced machine learning approaches , feature extraction techniques like TF-IDF and BoW , and ensemble models to build an "effective recruitment fraud detection model", strengthening the security of online industry infrastructure.

Sustainable Cities and Communities (SDG 11):

By addressing online recruitment scams , the system helps protect the economic well-being and privacy of individuals within communities. Preventing fraud contributes to a more stable and secure news market, which is a key component of a sustainable

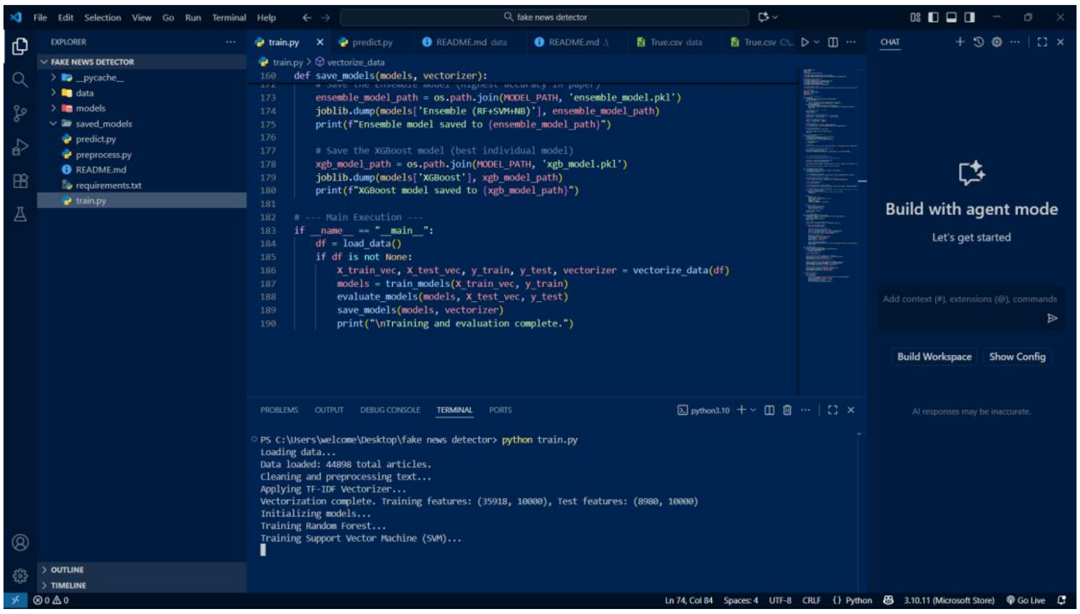
A2. SCREENSHOTS



```python
    print("Fake News Detector - Ensemble Model")
    print("Enter a news title and text to check if it's FAKE or REAL.")
    print("Type 'q' at any prompt to quit.")

    while True:
        title = input("Enter News Title: ")
        if title.lower() == 'q':
            break

        text = input("Enter News Text: ")
        if text.lower() == 'q':
            break

        if not title.strip() or not text.strip():
            print("Error: Both title and text are required. Please try again.")
            continue

        predict_news(title, text, vectorizer, svm_model, rf_model, nb_model)

    print("Exiting detector. Goodbye!")
```

```
PS C:\Users\welcome\Desktop\fake news detector> pip install -r requirements.txt
Requirement already satisfied: pandas in c:\users\welcome\appdata\local\packages\pythonsoftwarefoundation.python.3.10_
qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from -r requirements.txt (line 3)) (2.2.2)
Requirement already satisfied: scikit-learn in c:\users\welcome\appdata\local\packages\pythonsoftwarefoundation.python
.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from -r requirements.txt (line 4)) (1.5.1)
Collecting xgboost
  Downloading xgboost-3.1.1-py3-none-win_amd64.whl (72.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 MB 5.1 MB/s eta 0:00:00
Requirement already satisfied: joblib in c:\users\welcome\appdata\local\packages\pythonsoftwarefoundation.python.3.10_
qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from -r requirements.txt (line 6)) (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\welcome\appdata\local\packages\pythonsoftwarefoundation.python
.3.10_qbz5n2kfra8p0\localcache\local-packages\python310\site-packages (from pandas->-r requirements.txt (line 3)) (202
4.1)
```

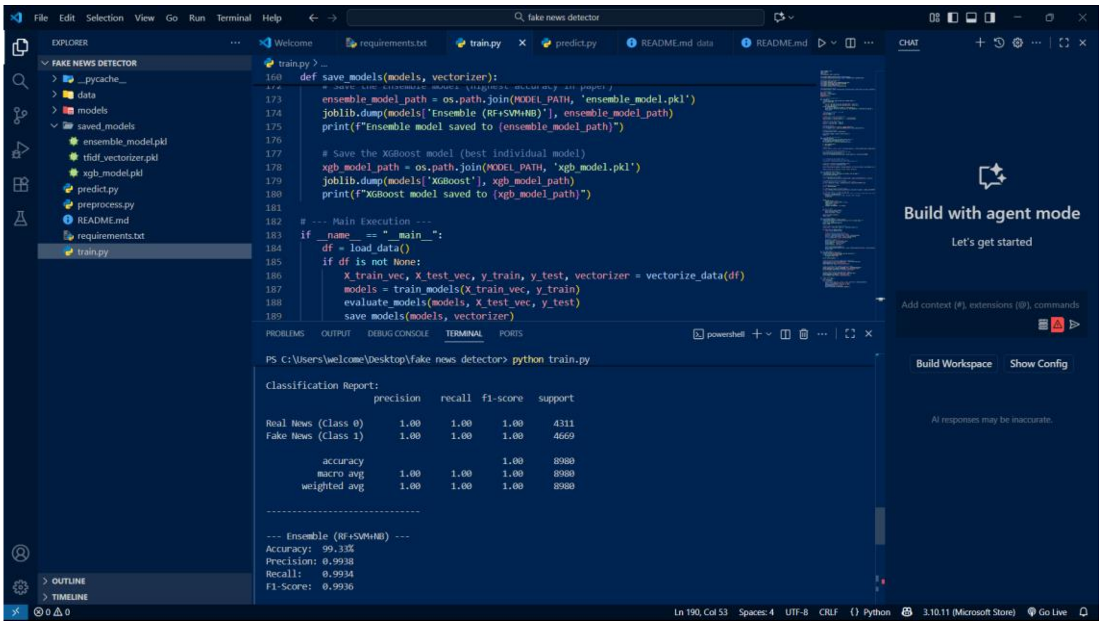
Fig: A.8.1. Screenshot of the XGBoost library installation in progress via the terminal.



```python
def save_models(models, vectorizer):
    # Save the Ensemble model (highest accuracy in paper)
    ensemble_model_path = os.path.join(MODEL_PATH, 'ensemble_model.pkl')
    joblib.dump(models['Ensemble (RF+SVM+NB)'], ensemble_model_path)
    print(f"Ensemble model saved to {ensemble_model_path}")

    # Save the XGBoost model (best individual model)
    xgb_model_path = os.path.join(MODEL_PATH, 'xgb_model.pkl')
    joblib.dump(models['XGBoost'], xgb_model_path)
    print(f"XGBoost model saved to {xgb_model_path}")

# --- Main Execution ---
if __name__ == "__main__":
    df = load_data()
    if df is not None:
        X_train_vec, X_test_vec, y_train, y_test, vectorizer = vectorize_data(df)
        models = train_models(X_train_vec, y_train)
        evaluate_models(models, X_test_vec, y_test)
        save_models(models, vectorizer)
        print("\nTraining and evaluation complete.")
```

```
PS C:\Users\welcome\Desktop\fake news detector> python train.py
Loading data...
Data loaded: 44898 total articles.
Cleaning and preprocessing text...
Applying TF-IDF Vectorizer...
Vectorization complete. Training features: (35918, 10000), Test features: (8980, 10000)
Initializing models...
Training Random Forest...
Training Support Vector Machine (SVM)...
```

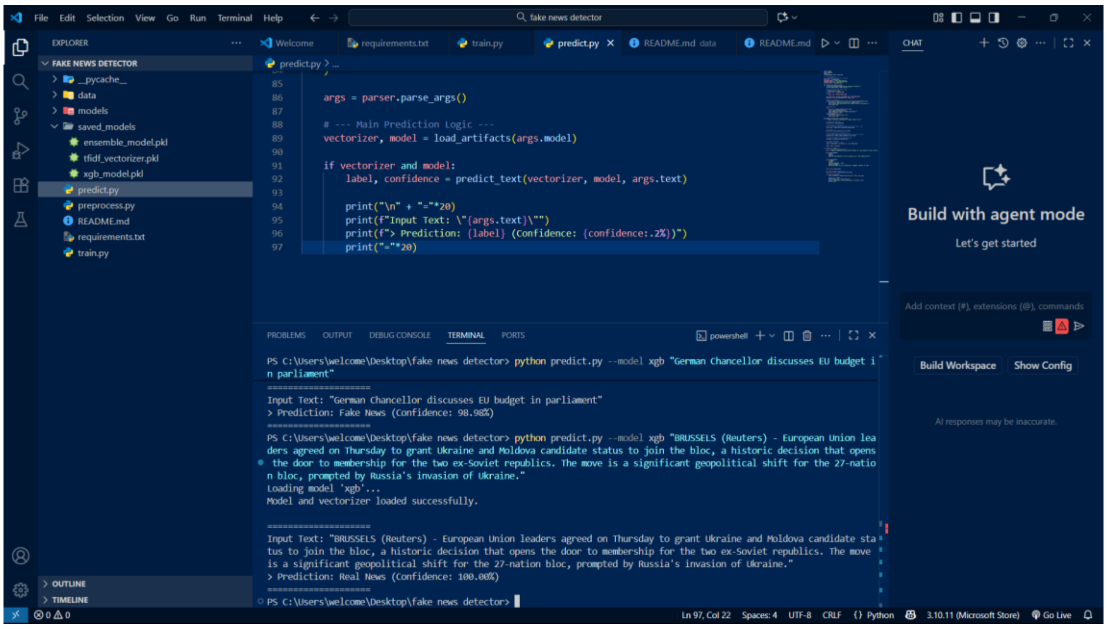
Fig: A.8.2. Screenshot of the terminal output showing the machine learningmodel training process underway



```python
def save_models(models, vectorizer):
    ensemble_model_path = os.path.join(MODEL_PATH, 'ensemble_model.pkl')
    joblib.dump(models['Ensemble (RF+SVM+NB)'], ensemble_model_path)
    print(f"Ensemble model saved to {ensemble_model_path}")

    # Save the XGBoost model (best individual model)
    xgb_model_path = os.path.join(MODEL_PATH, 'xgb_model.pkl')
    joblib.dump(models['XGBoost'], xgb_model_path)
    print(f"XGBoost model saved to {xgb_model_path}")

# --- Main Execution ---
if __name__ == "__main__":
    df = load_data()
    if df is not None:
        X_train_vec, X_test_vec, y_train, y_test, vectorizer = vectorize_data(df)
        models = train_models(X_train_vec, y_train)
        evaluate_models(models, X_test_vec, y_test)
        save_models(models, vectorizer)
```

```
PS C:\Users\welcome\Desktop\fake news detector> python train.py

Classification Report:
                   precision    recall  f1-score   support

Real News (Class 0)     1.00      1.00      1.00      4311
Fake News (Class 1)     1.00      1.00      1.00      4669

           accuracy                         1.00      8980
          macro avg     1.00      1.00      1.00      8980
       weighted avg     1.00      1.00      1.00      8980

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

--- Ensemble (RF+SVM+NB) ---
Accuracy:  99.33%
Precision: 0.9938
Recall:    0.9934
F1-Score:  0.9936
```

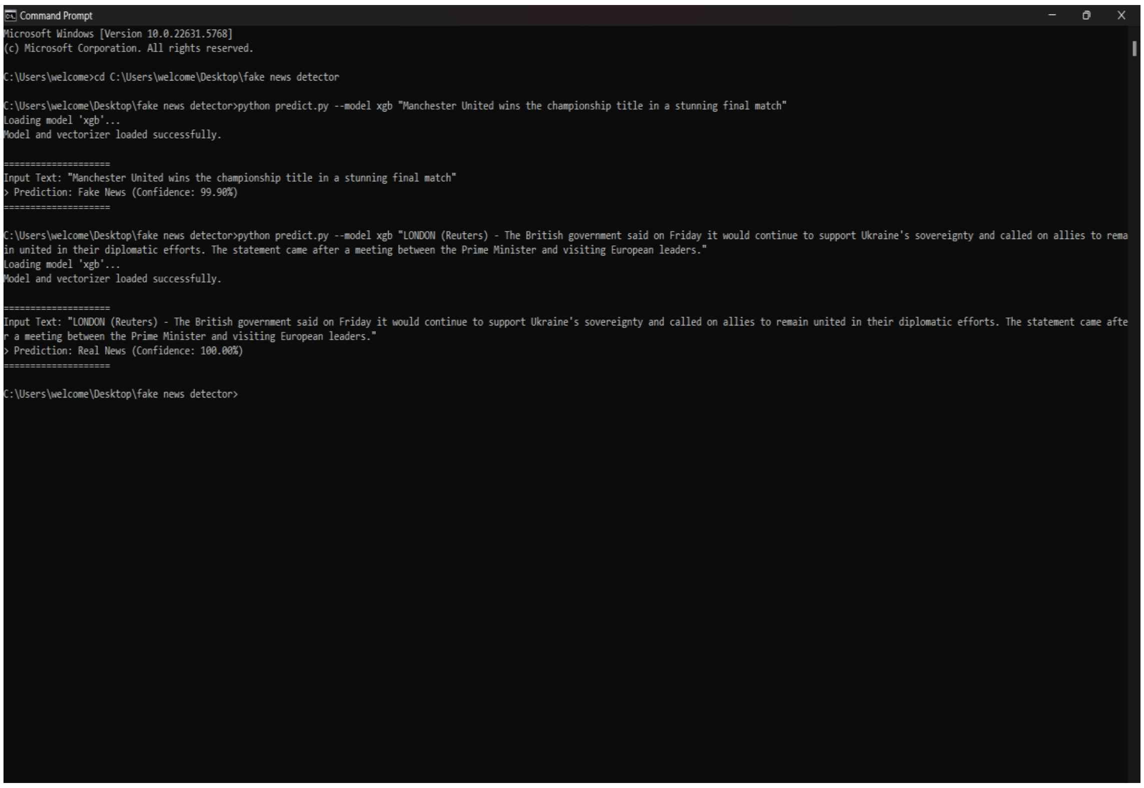
Fig: A.8.3. Screenshot of the terminal displaying the model evaluation results, including the classification report and accuracy scores, after the training process is complete.



```python
    args = parser.parse_args()

    # --- Main Prediction Logic ---
    vectorizer, model = load_artifacts(args.model)

    if vectorizer and model:
        label, confidence = predict_text(vectorizer, model, args.text)

        print("\n" + "="*20)
        print(f"Input Text: \"{args.text}\"")
        print(f"> Prediction: {label} (Confidence: {confidence:.2%})")
        print("="*20)
```

```
PS C:\Users\welcome\Desktop\fake news detector> python predict.py --model xgb "German Chancellor discusses EU budget i
n parliament"
====================
Input Text: "German Chancellor discusses EU budget in parliament"
> Prediction: Fake News (Confidence: 98.98%)
====================
PS C:\Users\welcome\Desktop\fake news detector> python predict.py --model xgb "BRUSSELS (Reuters) - European Union lea
ders agreed on Thursday to grant Ukraine and Moldova candidate status to join the bloc, a historic decision that opens
 the door to membership for the two ex-Soviet republics. The move is a significant geopolitical shift for the 27-natio
n bloc, prompted by Russia's invasion of Ukraine."
Loading model 'xgb'...
Model and vectorizer loaded successfully.

====================
Input Text: "BRUSSELS (Reuters) - European Union leaders agreed on Thursday to grant Ukraine and Moldova candidate sta
tus to join the bloc, a historic decision that opens the door to membership for the two ex-Soviet republics. The move
is a significant geopolitical shift for the 27-nation bloc, prompted by Russia's invasion of Ukraine."
> Prediction: Real News (Confidence: 100.00%)
====================
PS C:\Users\welcome\Desktop\fake news detector>
```

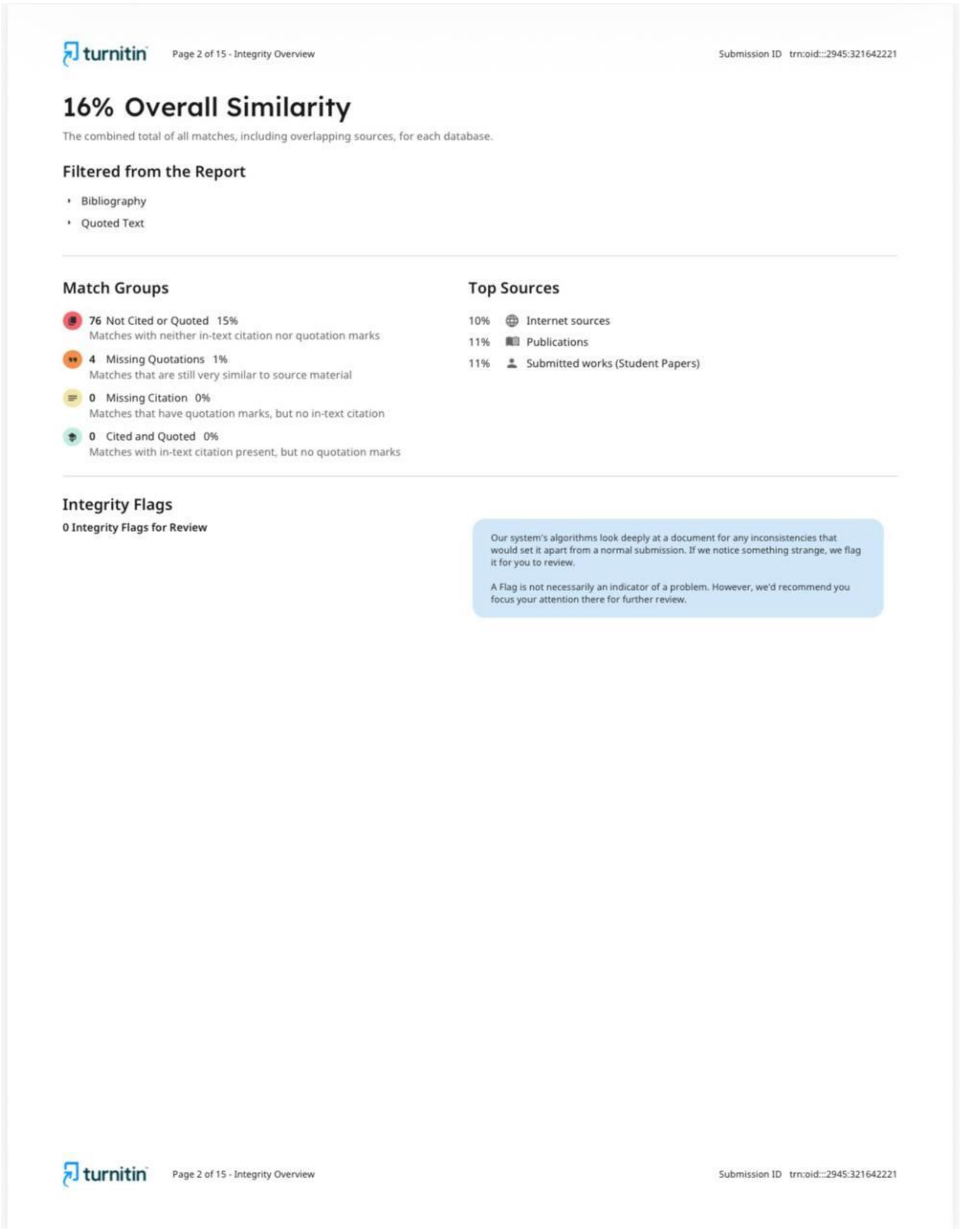
FigA.8.4. Screenshot of the terminal showing the prediction script being tested with the XGBoost model (--model xgb)



```
Command Prompt
Microsoft Windows [Version 10.0.22631.5768]
(c) Microsoft Corporation. All rights reserved.

C:\Users\welcome>cd C:\Users\welcome\Desktop\fake news detector

C:\Users\welcome\Desktop\fake news detector>python predict.py --model xgb "Manchester United wins the championship title in a stunning final match"
Loading model 'xgb'...
Model and vectorizer loaded successfully.

====================
Input Text: "Manchester United wins the championship title in a stunning final match"
> Prediction: Fake News (Confidence: 99.90%)
====================

C:\Users\welcome\Desktop\fake news detector>python predict.py --model xgb "LONDON (Reuters) - The British government said on Friday it would continue to support Ukraine's sovereignty and called on allies to rema
in united in their diplomatic efforts. The statement came after a meeting between the Prime Minister and visiting European leaders."
Loading model 'xgb'...
Model and vectorizer loaded successfully.

====================
Input Text: "LONDON (Reuters) - The British government said on Friday it would continue to support Ukraine's sovereignty and called on allies to remain united in their diplomatic efforts. The statement came afte
r a meeting between the Prime Minister and visiting European leaders."
> Prediction: Real News (Confidence: 100.00%)
====================

C:\Users\welcome\Desktop\fake news detector>
```

FigA.8.5. Screenshot of the final execution in the command prompt, showing the XGBoost model classifying one input as 'Fake News' and a Reuters input as 'Real News'.

A3. PLAGIARISM REPORT



turnitin Page 2 of 15 - Integrity Overview Submission ID trn:oid:::2945:321642221

16% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

76 Not Cited or Quoted 15%
Matches with neither in-text citation nor quotation marks

4 Missing Quotations 1%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

10%	Internet sources
11%	Publications
11%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

turnitin Page 2 of 15 - Integrity Overview Submission ID trn:oid:::2945:321642221

CHAPTER 9

REFERENCES

Santhiya, P., Kavitha, S., Aravindh, T., Archana, S., & Praveen, A. V. (2023). "FAKE NEWS DETECTION USING MACHINE LEARNING." In Proceedings of the 2023 International Conference on Computer Communication and Informatics (ICCCI). Coimbatore, India. DOI: 10.1109/ICCCI56745.2023.10128339. This is the base paper for your project, which uses the EMSCAD dataset and the ensemble methodology.

Ahmad, I., Yousaf, M., Yousaf, S., & Ahmad, M. O. (2020). "Fake News Detection Using Machine Learning Ensemble Methods." Complexity, vol. 2020, Article ID 8885861.

Sharma, U., Saran, S., & Patil, S. M. (2021). "Fake News Detection using Machine Learning Algorithms." International Journal of Engineering Research & Technology (IJERT), NTASU – 2020 (Volume 09 – Issue 03).

Faustini, P. H. A., & Covoes, T. F. (2020). "Fake news detection in multiple platforms and languages." Expert Systems with Applications, Volume 158, 113503.

Khandagale, P., Utekar, A., Dhonde, A., & Karve, S. S. (2022). "Fake Job Detection using Machine Learning." ISSN 2321-9653.

Manzoor, S. I., Singla, J., & Nikita. (2019). "Fake News Detection Using Machine Learning approaches: A systematic Review." In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 230-234.

Jain, A., Shakya, A., Khatter, H., & Gupta, A. K. (2019). "A smart System for Fake News Detection Using Machine Learning." In 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT).

Khanam, Z., Alwasel, B. N., Sirafi, H., & Rashid, M. (2021). "Fake News Detection using Machine Learning Approaches." IOP Conference Series: Materials Science and Engineering, Volume 1099.

Wang, Y., Yang, W., Ma, F., Xu, J., Zhong, B., Deng, Q., & Gao, J. (2020). "Weak Supervision for Fake News Detection via Reinforcement Learning." In Proceedings of the AAAI Conference on Artificial Intelligence, 34(01), 516-523.

Aslam, N., Khan, I. U., Alotaibi, F. S., Aldaej, L. A., & Aldubaikil, A. K. (2021). "Fake Detect: A Deep Learning Ensemble Model for Fake News Detection." Complexity, vol. 2021, Article ID 5557784.

Braşoveanu, A. M. P., & Andonie, R. (2019). "Semantic Fake News Detection: A Machine Learning Perspective." In Advances in Computational Intelligence. IWANN 2019. Lecture Notes in Computer Science(), vol 11506. Springer.

Choudhary, A., & Arora, A. (2021). "Linguistic feature based learning model for fake news detection and classification." Expert Systems with Applications, Volume 169, 114171.

Abdullah-All-Tanvir, Mahir, E. M., Akhter, S., & Huq, M. R. (2019). "Detecting Fake News using Machine Learning and Deep Learning Algorithms." In 2019 7th International Conference on Smart Computing & Communications (ICSCC).

Reis, J. C. S., Correia, A., Murai, F., Veloso, A., & Benevenuto, F. (2019). "Supervised Learning for Fake News Detection." IEEE Intelligent Systems, vol. 34, no. 2, pp. 76-81.

Mehboob, A., & Malik, M. S. I. (2021). "Smart Fraud Detection Framework for Job Recruitments." Arab J Sci Eng, 46, 3067–3078.

Srivastava, R. (2022). "Identification of Online Recruitment Fraud (ORF) through predictive Models." EJBESS, Vol. 1(1):39-51.

Feeney, J. (2011). "Predicting Job Applicant Faking with Self-Control." Digitized Theses.

Aljwari, F., Alkaberi, W., Alshutayri, A., Aldhahri, E., Aljojo, N., & Abouola, O. (2Example: 22). "Multiscale Machine Learning Prediction of the Spread of Arabic Online Fake News." Postmodern Openings, 13(1Sup1), 01-14.

Bandyopadhyay, S., & Dutta, S. (2020). "Analysis of Fake News In Social Medias during Lockdown in COVID-19." Preprints 2020, 2020060243.

Gilda, S. (2017). "Notice of Violation of IEEE Publication Principles: Evaluating machine learning algorithms for fake news detection." In 2017 IEEE 15th Student Conference on Research and Development (SCOReD), pp. 110-115.

Jain, A., & Kasbe, A. (2018). "Fake News Detection." In 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1-5.

Alghamdi, B., & Alharby, F. (2019). "An intelligent model for online recruitment fraud detection." J. Inf. Secur., vol. 10, no. 03, p. 155.

Lal, S., Jiaswal, R., Sardana, N., Verma, A., Kaur, A., & Mourya, R. (2019). "ORFDetector: Ensemble Learning Based Online Recruitment Fraud Detection." In 2019 Twelfth International Conference on Contemporary Computing (IC3), pp. 1-5.

Vidros, S., Kolias, C., Kambourakis, G., & Akoglu, L. (2Santhiya, P., Kavitha, S., Aravindh, T., Archana, S., & Praveen, A. V. (2023).017). "Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset." Futur. Internet, vol. 9, no. 1, p. 6. This is the paper that introduced the **EMSCAD dataset** used in your base paper.