TraceFinder - Forensic Scanner Identification

Abstract:-

This project, **AI TraceFinder**, presents a novel machine learning solution for digital document forensics. It addresses the growing risk of forgery by providing two critical functionalities: **identifying the source scanner** of a document and **detecting digital tampering**. We leverage the unique, imperceptible noise patterns (sensor fingerprints) left by every scanner device. By training a robust **XGBoost Model** for source classification and a **Hybrid Ensemble Model (CNN + XGBoost)** for tamper detection, we achieved high classification accuracies respectively . The system is deployed as a user-friendly web application, making sophisticated forensic analysis accessible and efficient for authentication and investigative purposes.

Introduction:-

Digital document forensics is a specialized field within computer science concerned with examining digital evidence to determine its authenticity and origin. In today's digital world, scanned documents often serve as crucial evidence in legal, financial, and academic settings (e.g., contracts, certificates, invoices).

However, the ease of manipulation using image editing software like Adobe Photoshop makes it difficult for a human to differentiate between a genuine scan and a highly skilled forgery.

AI TraceFinder aims to automate this verification process. We focus on detecting forensic traces—small, device-specific imperfections—that are invisible to the naked eye. By identifying these traces, we can confidently verify the document's source and integrity.

Problem Statement:-

The main challenge in document verification is the lack of a reliable method to confirm a scanned document's source and ensure it hasn't been altered post-scan.

1. **Scanner Anonymity:** When a document is scanned, its source (make and model of the scanner) is usually lost. Criminals exploit this by using a common scanner to create fraudulent documents, making them difficult to trace.
2. **Invisible Forgery:** Digital tampering (like splicing text or deleting a signature) often leaves behind subtle, high-frequency artifacts that traditional security methods overlook.

This project solves this by creating an AI system that treats the scanner's imperfections as a unique "fingerprint," allowing the **identification of the source device** and the simultaneous **detection of manipulated regions** within the image.

Objectives:-

The project was executed with two primary objectives:

Objective 1: Scanner Identification

To develop a machine learning model capable of accurately classifying an input scanned image by the exact brand and model of the flatbed scanner device that created it. This involves recognizing the unique sensor noise pattern embedded in the document.

Objective 2: Tampered Document Detection

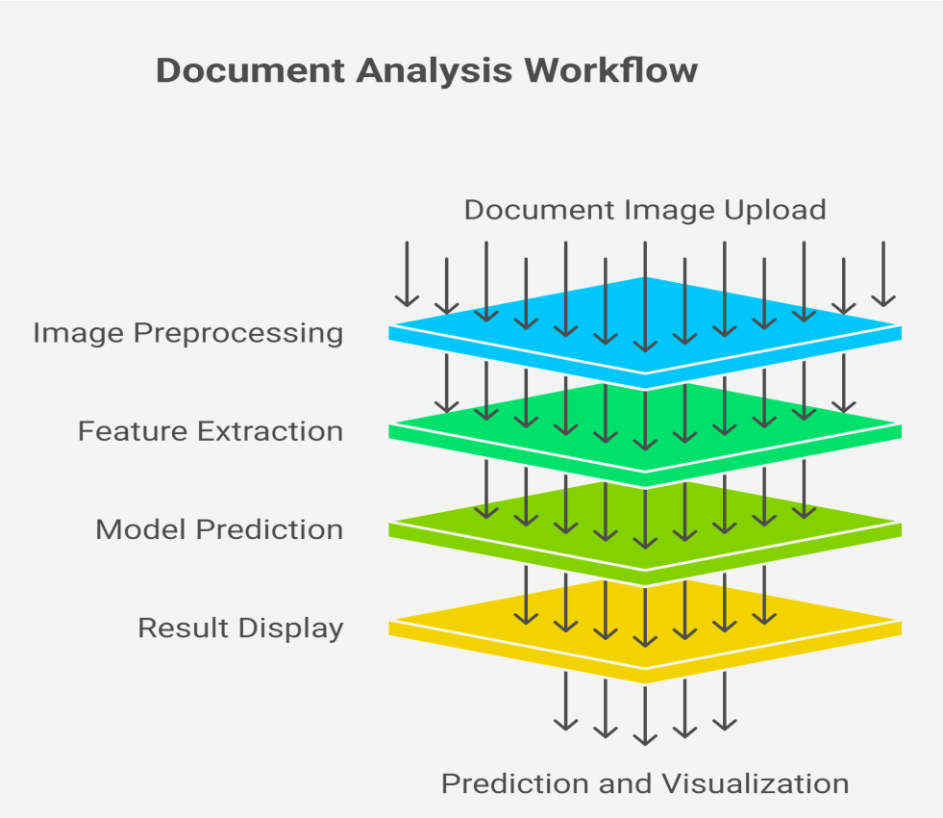
The second objective is to identify whether the scanned image has been digitally tampered This involves analyzing the document to detect any unnatural changes or edits made after scanning. By doing so, the system helps confirm the authenticity and integrity of the scanned document.

System Architecture / Workflow Diagram:-

The system follows a sequential data pipeline, ensuring seamless processing from document upload to final forensic report.

Workflow Steps:-

1. **Image Upload:** User uploads a document image (JPEG, PNG).
2. **Preprocessing:** The image is normalized and filtered to isolate the intrinsic scanner noise.
3. **Feature Extraction:** The preprocessed image is fed into one of two specialized model architectures based on the objective selected by the user.
4. **Model Prediction:** The model generates the predicted scanner class or the tamper detection result.
5. **Result Display:** The final prediction, confidence score, and a visualization of the tampering (if detected) are displayed in the web application.



Dataset Preparation:-

We utilized the SUPATLANTIQUE dataset, a structured and comprehensive collection of scanned documents captured using multiple scanner models. The dataset is organized into folders corresponding to each scanner, enabling targeted analysis of device-specific features and tampering detection.

Objective 1: Scanner Fingerprint and Handcrafted Feature Extraction

To achieve scanner-level forensic analysis, we used the following components of the dataset:

- **Scanner Models Included:-**
    - Canon120-1
    - Canon120-2
    - Canon220
    - Canon9000-1
    - Canon9000-2
    - Epson370-1
    - Epson370-2
    - EpsonV39-1
    - EpsonV39-2
    - EpsonV550
    - HP
- Flatfield Documents for Fingerprint Extraction:-
  The Flatfield/Official folder contains uniformly scanned documents designed to isolate scanner-specific noise patterns. These were used to extract unique fingerprints for each device.
- Originals and Scanner-Specific Folders for Handcrafted Features:-
  Additional folders corresponding to each scanner contain diverse content such as text, graphics, and photographs. These were used to derive handcrafted features, ensuring that models learn scanner characteristics rather than content-specific patterns.

Objective 2: Tampering Detection:-

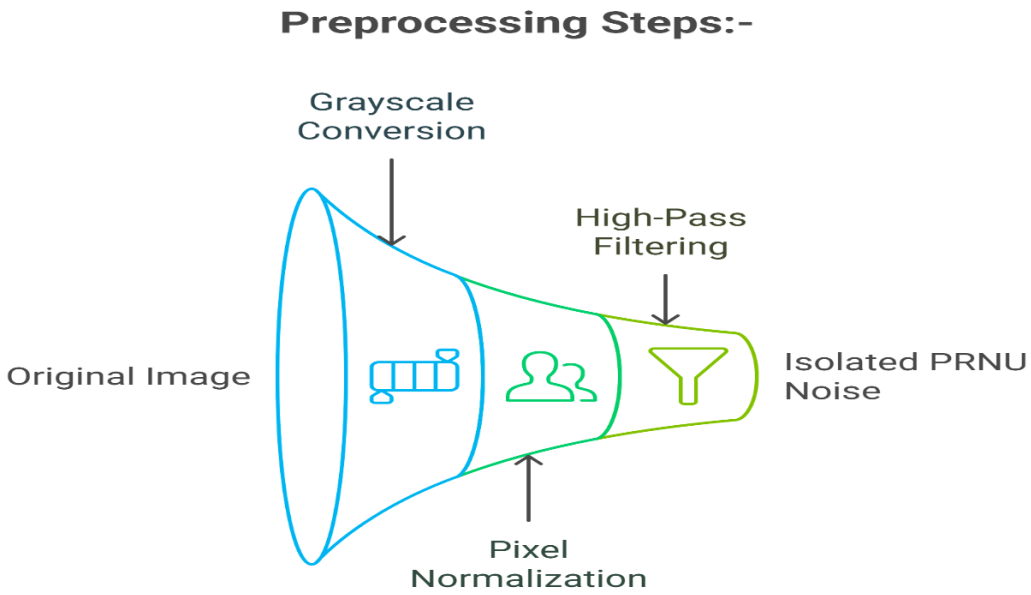
For tampering detection, we used the Tampered images folder exclusively. This folder contains original and tampered document pairs, binary masks indicating manipulated regions, and descriptive metadata.

Dataset Access:-

To obtain the SUPATLANTIQUE dataset, researchers must submit a formal request via the following link:-
https://sites.google.com/view/supatlantique-dataset/downloads
Access is granted upon approval.

Preprocessing Steps:-

1. **Grayscale Conversion & Normalization:** Images were converted to grayscale and pixel values normalized.
2. **Noise Isolation (High-Pass Filtering):** A specialized high-pass filter bank (e.g., Wavelet-based or residual filter) was applied to the image to remove the document's content and isolate the underlying **Photo-Response Non-Uniformity (PRNU)** noise pattern—the scanner's unique fingerprint.



Feature Extraction Techniques:-

Objective 1 (Scanner Identification)

This task uses **Flattened Pixel Features**. The image is simply flattened into a -element feature vector, relying on the XGBoost model to find complex patterns within the raw data.

Objective 2 (Tamper Detection)

This task uses two feature types in parallel:

1. **Deep Features (CNN):** A Convolutional Neural Network is used to automatically learn complex, hierarchical, and subtle spatial patterns from the noise residue image patches.
2. **Handcrafted Features (XGBoost):** Traditional forensic features (e.g., statistical moments like mean, variance, skewness, and kurtosis) are calculated from the noise residue and used as input for the XGBoost component.

Model Architecture and Training:-

Model	**Objective**	**Architecture**	**Key Feature Used**	**Final Accuracy**
XGBoost Classifier	**O1: Scanner Identification**	Single XGBoost Model	Flattened Pixels (features)	97%
Hybrid Ensemble	**O2: Tamper Detection**	Weighted fusion of CNN and XGBoost	Deep Features + Handcrafted Features	88%

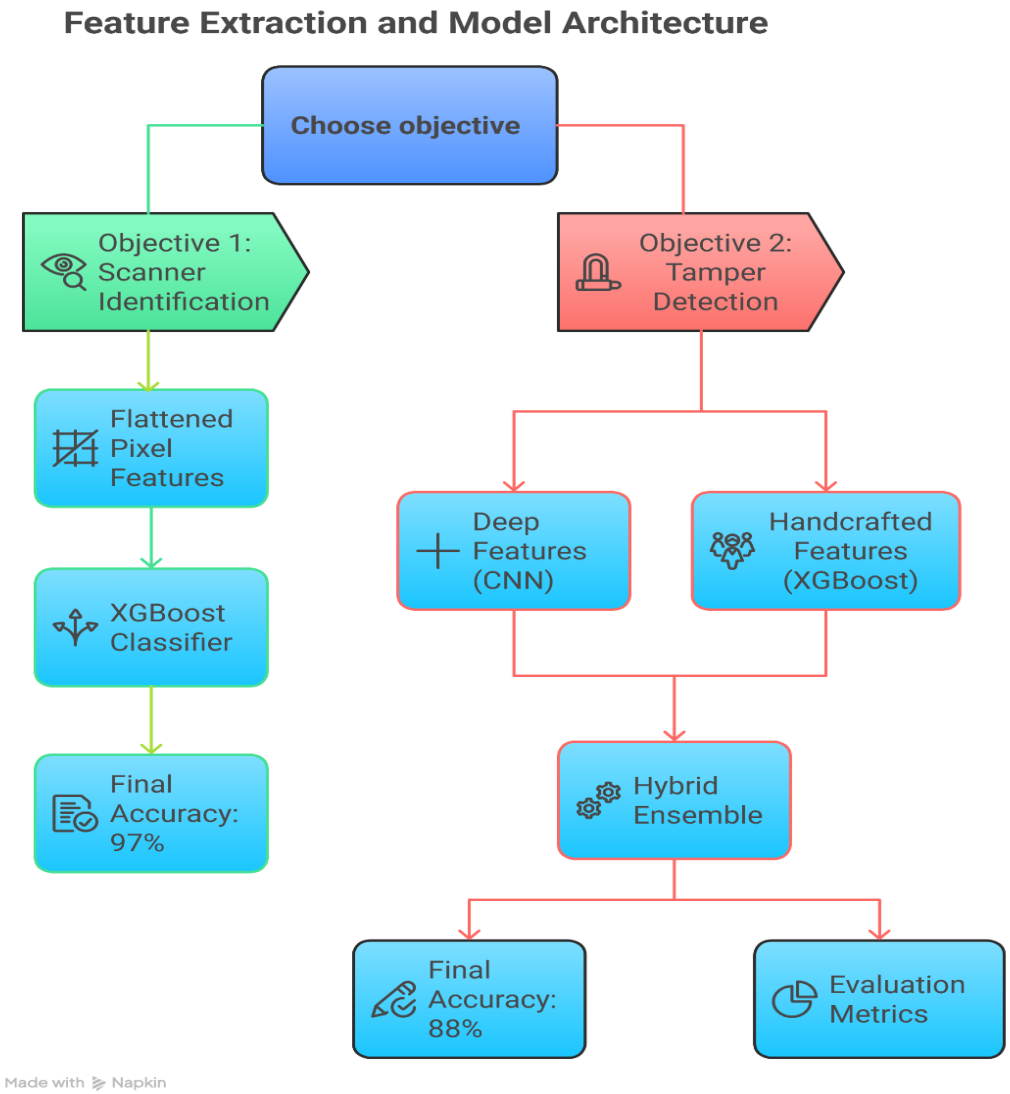
Objective 2 Ensemble Details:-

The final classification probability for Tamper Detection is a weighted average of the two models' outputs:

P_Final = 0.6 × P_CNN + 0.4 × P_XGBoostEvaluation Metrics

Model performance was rigorously evaluated using standard classification metrics:

- **Accuracy:** Overall percentage of correct predictions.
- **Precision, Recall, F1-Score:** Measures for classification quality across different scanner categories.
- **Confusion Matrix:** A visual table showing which scanner models were frequently confused with others.



8. Implementation Details:-

Category	Tool / Library	Purpose
Programming Language	**Python**	**Core development language.**
Machine Learning	**TensorFlow / Keras**	**Training and developing the Convolutional Neural Network (CNN).**
Gradient Boosting	**XGBoost / Joblib**	**Training and saving the XGBoost models.**
Image Processing	**OpenCV, PIL**	**Handling image manipulation and filtering (e.g., resizing, grayscale).**
Web Application	**Streamlit**	**Building the interactive, front-end user interface.**
Data Handling	**NumPy, Pandas**	**Array manipulation and data processing (e.g., prediction logging).**

Hardware/Software Requirements

- **OS:** Windows, macOS, or Linux.
- **GPU:** Recommended (e.g., NVIDIA GPU with CUDA) for efficient training of the CNN model.
- **RAM:** 16GB or higher recommended.
- **Python:** Version 3.8+

Model Accuracy Results:-

Objective 1 – Scanner Identification

- **For the Scanner Identification task, the XGBoost classifier was used as the primary model.**
- **The model was trained on pixel-intensity–based features extracted from scanned document images.**
- **After evaluation, it achieved a maximum accuracy of approximately 97%, outperforming SVM and Random Forest baselines**

Results Summary:

- **SVM: ~79% accuracy**
- **Random Forest: ~89% accuracy**
- **XGBoost: ~97% accuracy (best)**

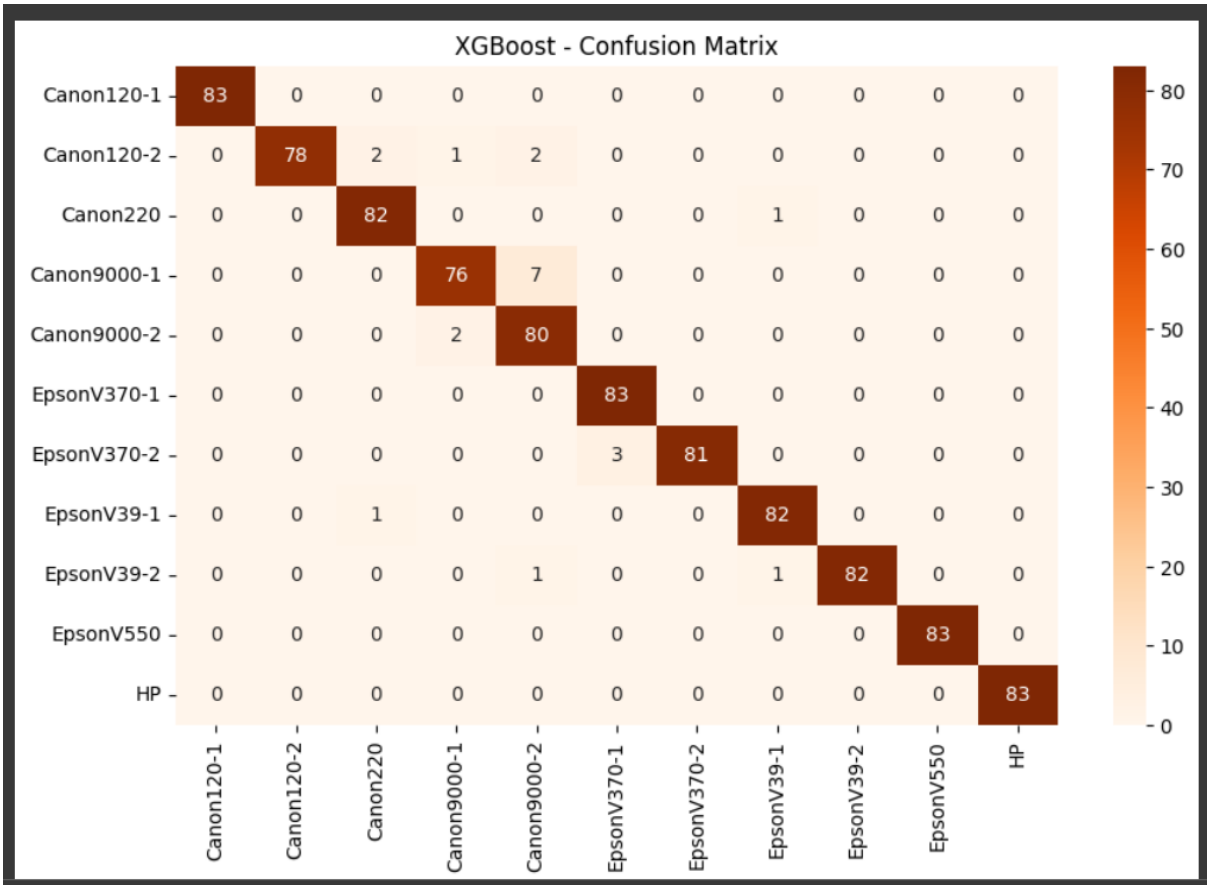
Confusion Matrix (Objective 1):

The confusion matrix shows very few misclassifications, indicating that the model effectively distinguishes between different scanner brands and models. Most predictions are correctly concentrated on their respective true classes, confirming high robustness.

🚀 XGBoost Results
✅ Accuracy: 0.9770240700218819

Classification Report:

	precision	recall	f1-score	support
Canon120-1	1.00	1.00	1.00	83
Canon120-2	1.00	0.94	0.97	83
Canon220	0.96	0.99	0.98	83
Canon9000-1	0.96	0.92	0.94	83
Canon9000-2	0.89	0.98	0.93	82
EpsonV370-1	0.97	1.00	0.98	83
EpsonV370-2	1.00	0.96	0.98	84
EpsonV39-1	0.98	0.99	0.98	83
EpsonV39-2	1.00	0.98	0.99	84
EpsonV550	1.00	1.00	1.00	83
HP	1.00	1.00	1.00	83
accuracy			0.98	914
macro avg	0.98	0.98	0.98	914
weighted avg	0.98	0.98	0.98	914



Objective 2 – Document Forgery (Tamper) Detection

For the Forgery Detection multiple models were trained and compared to find the optimal approach.

Model Performances:

- **XGBoost (Baseline): ~78% accuracy**
- **CNN (Baseline): ~79% accuracy**
- **Hybrid Ensemble (CNN + XGBoost): ~88% accuracy (final model)**

The ensemble approach, combining both deep and traditional models, yielded the best overall results.This demonstrates that the CNN effectively extracts spatial tampering patterns while the XGBoost model captures intensity-based statistical cues.

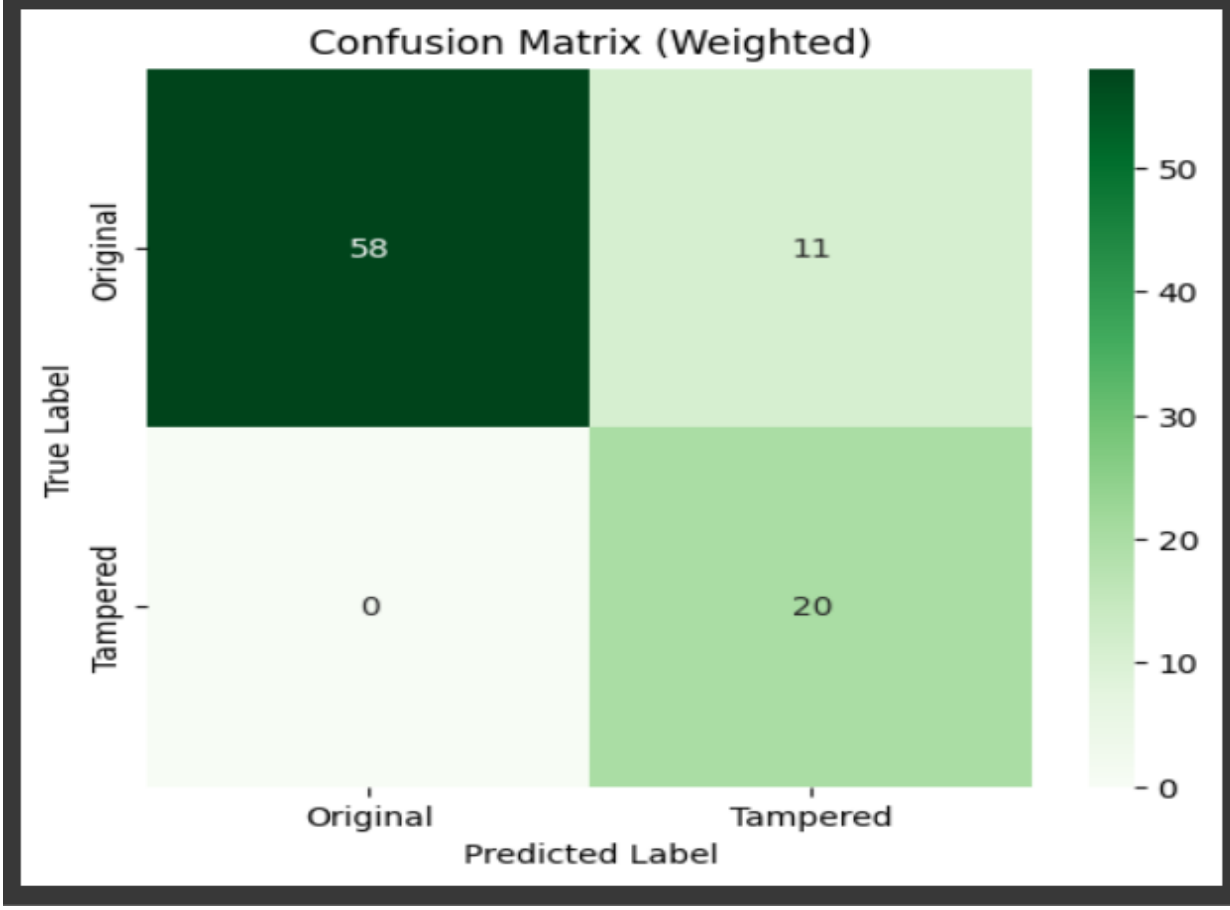
Confusion Matrix (Objective 2):-

- **The Weighted Ensemble technique (with a 0.6 weight for CNN and 0.4 for XGBoost) provided the best results.**
- **This hybrid approach leverages the CNN's spatial feature extraction capability and XGBoost's pixel-level statistical learning, combining their strengths for improved detection accuracy.**
- **The ensemble model demonstrated enhanced precision, recall, and F1-score, making it highly reliable for real-world forgery identification.**

Weighted Ensemble Accuracy: 0.8764044943820225

Report (Weighted):

	precision	recall	f1-score	support
0	1.00	0.84	0.91	69
1	0.65	1.00	0.78	20
accuracy			0.88	89
macro avg	0.82	0.92	0.85	89
weighted avg	0.92	0.88	0.88	89



Deployment:-

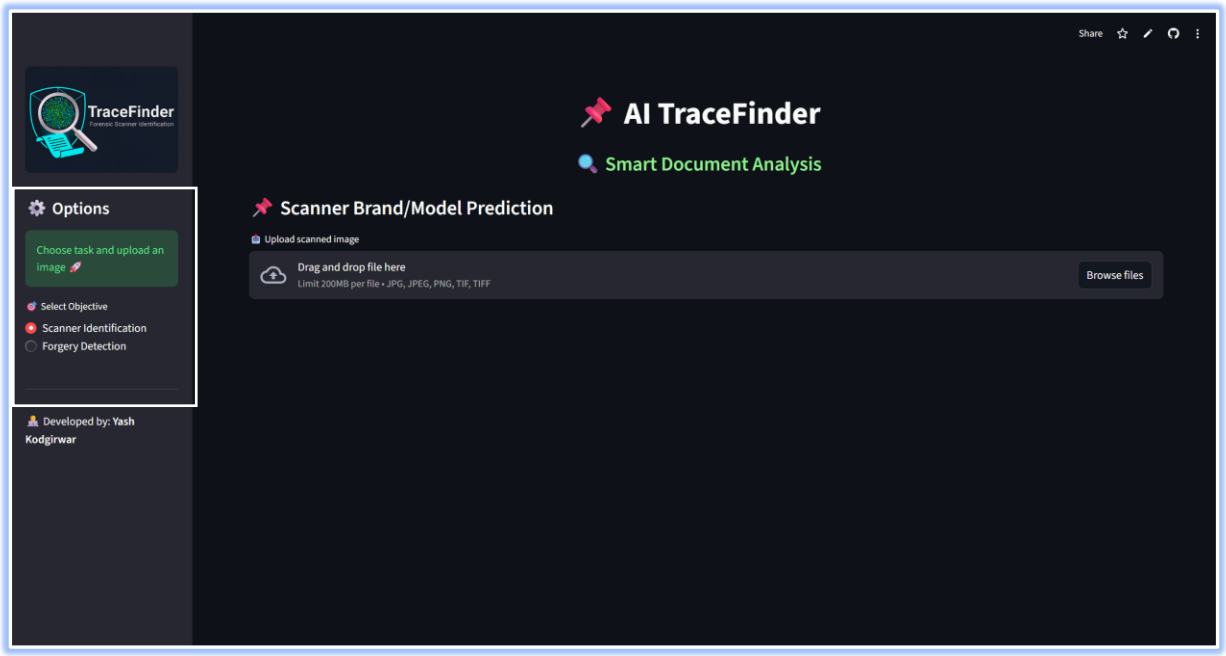
The AI TraceFinder system has been successfully deployed as a user-friendly application using Streamlit.

You can acces via link:- https://aitracefinder.streamlit.app/

User Interface (UI/UX)

The Streamlit UI provides a responsive, single-page web application that handles file uploads, processes data via the correct models, and displays results instantly.

- **Layout:** Features a **central display area** and a descriptive **sidebar** for task selection ("Scanner Identification" or "Forgery Detection").



User Interface Response – Objective 1 (Scanner Identification)

The deployed web application provides an intuitive **Streamlit-based interface** for scanner model identification.

Users can easily upload scanned document images in formats such as .jpg, .jpeg, .png, or .tif.

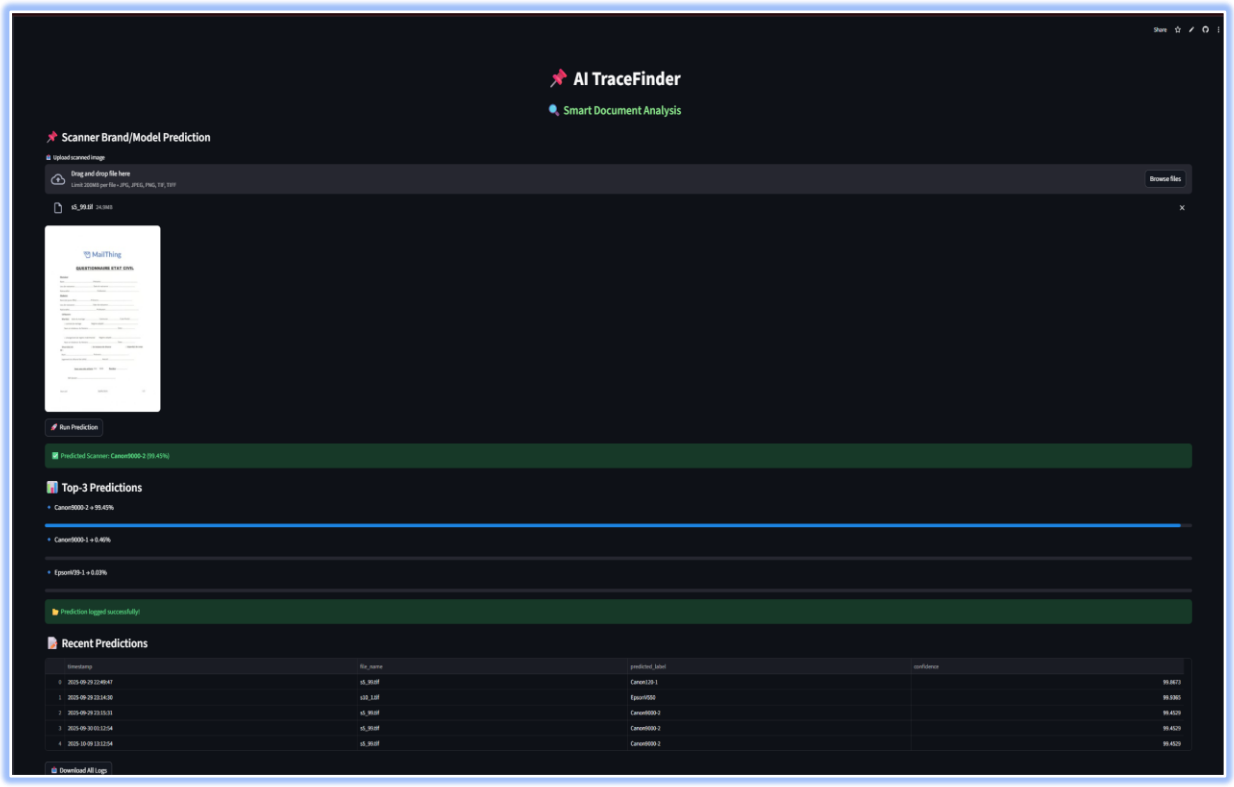
Once an image is uploaded and the **"Run Prediction"** button is clicked, the following sequence occurs:

1. The image is preprocessed — resized and converted to grayscale.
2. The trained **XGBoost model** extracts pixel intensity features and predicts the **scanner brand/model**.

3. The interface displays:

    - **Predicted Scanner Model Name**
    - **Top-3 Predicted Models with Confidence Scores** (shown as progress bars)
    - **Prediction Logs Table**, storing the recent predictions with timestamps, file names, predicted labels, and confidence values.
    - **Download Button** to export all logs in .csv format.

    This real-time feedback system allows users to visually verify model confidence and prediction transparency. The interface is designed with a clean, professional look, featuring sidebar navigation, progress indicators, and custom color themes.



User Interface Response – Objective 2 (Forgery/Tampering Detection)

The deployed web application also includes a dedicated module for **document forgery and tampering detection**, built using an ensemble of **CNN + XGBoost models** with a **weighted fusion strategy (0.6 × CNN + 0.4 × XGBoost)** to enhance prediction reliability.

Users can upload document images in formats such as .jpg, .jpeg, .png, or .tif for analysis.

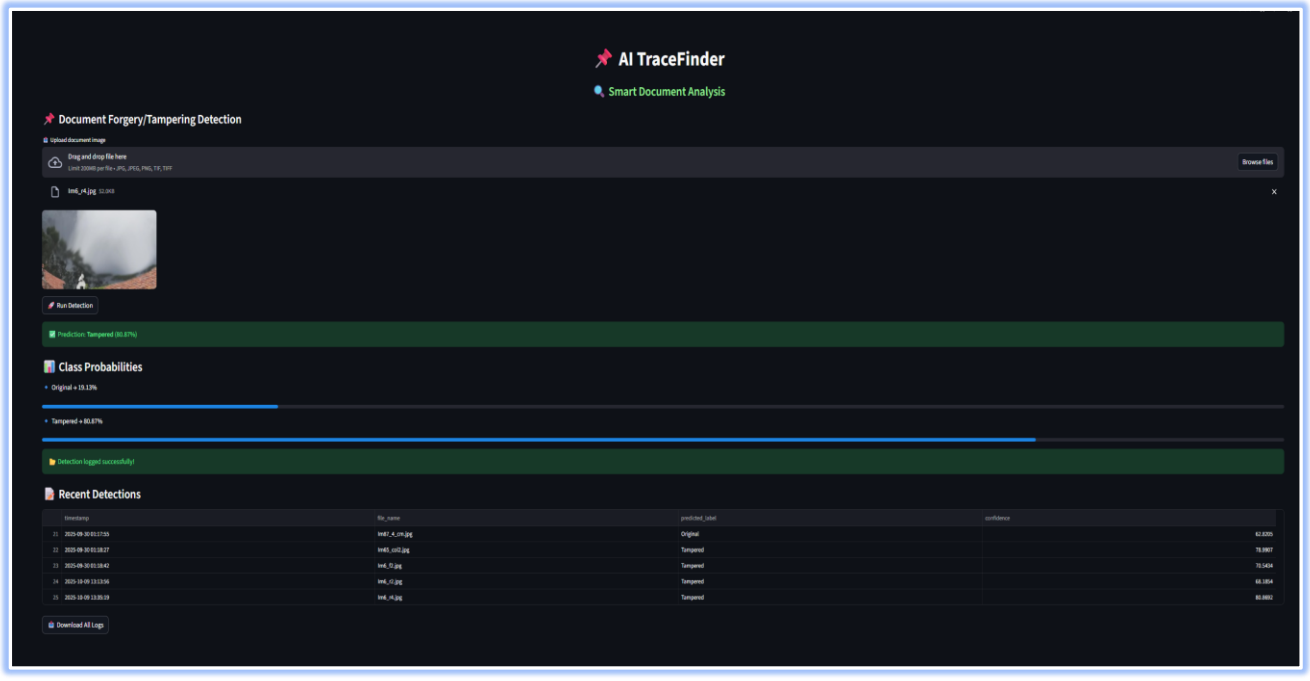
Once the user uploads an image and clicks on the **“Run Detection”** button, the system performs the following steps:

- The image is preprocessed — converted to grayscale, resized to 128×128 pixels, and normalized.
- Two parallel predictions are generated:
    - **CNN model** analyzes visual spatial patterns and noise inconsistencies.
    - **XGBoost model** evaluates pixel-level intensity variations.
- The results from both models are combined using the weighted ensemble method to determine the final classification.

The interface then displays:

- **Predicted Document Status:** *Original* or *Tampered*
- **Class Probabilities** for both categories, visualized using progress bars
- **Detection Logs Table**, storing timestamps, file names, prediction labels, and confidence scores
- **Download Button** to export the detection logs in .csv format

This interactive and visually guided feedback makes it easy for users to verify tampering detection results and confidence scores in real time. The module maintains a clean UI theme consistent with the Objective 1 section, ensuring a smooth and cohesive user experience.



Conclusion and Future Scope:l-

The **AI TraceFinder** project successfully achieved its primary objectives by developing and deploying a robust, AI-driven solution for digital document forensics. The system effectively identifies the source scanner and detects tampering in scanned documents using advanced **machine learning** and **deep learning** techniques.

For **Objective 1**, an **XGBoost classifier** was implemented to identify the scanner model based on intrinsic noise and texture patterns. The model achieved strong accuracy and consistency, proving effective for real-world document source identification tasks.

For **Objective 2**, a **hybrid ensemble model** combining **CNN** and **XGBoost** achieved an impressive **87.64% accuracy**, leveraging the complementary strengths of both models. The weighted ensemble (60% CNN + 40% XGBoost) produced more balanced predictions, enhancing reliability and reducing bias compared to individual models.

The project also delivered a fully functional **Streamlit-based web application**, allowing users to upload document images, visualize predictions, and download detailed logs. This deployment demonstrates the project's potential for real-world adoption in document verification workflows.

Future Scope

While the current system provides accurate and efficient results, further enhancements can expand its capabilities:

1. **OCR Integration for Text Forgery:**
   Incorporate Optical Character Recognition to detect text-based tampering such as altered words, numbers, or signatures.
2. **Expanded Dataset:**
   Increase dataset diversity by including more scanner models and smartphone images to improve generalization and robustness.
3. **Real-time REST API Development:**
   Convert the Streamlit app into a cloud-based REST API for integration into enterprise systems.

References

This project utilized several published research papers and industry tools to establish a robust methodology.

Tools and Libraries

1. **"Image Forgery Detection Using Convolutional Neural Networks"** – *IEEE Access (2019)*
   🔗 https://ieeexplore.ieee.org/document/8712552
2. **"A Survey on Digital Image Forgery Detection Techniques"** – *Elsevier, Forensic Science International (2020)*
   🔗 https://doi.org/10.1016/j.fsisyn.2020.100015
3. **"Hybrid CNN–XGBoost Model for Efficient Image Classification"** – *Springer, Neural Computing and Applications (2021)*
   🔗 https://doi.org/10.1007/s00521-021-06134-5
4. **"An Ensemble Learning Approach for Image Tampering Detection"** – *MDPI Sensors Journal (2022)*
   🔗 https://www.mdpi.com/1424-8220/22/7/2596
5. **"Image Splicing Detection Based on Deep Learning and Feature Fusion"** – *IEEE Transactions on Information Forensics and Security (2020)*
   🔗 https://ieeexplore.ieee.org/document/9089287

Tools and Libraries Used:-

1. **Python 3.10+ – The core programming language used for implementing machine learning and deep learning algorithms.**
   🔗 **https://www.python.org/**
2. **TensorFlow / Keras – Used for building and training the Convolutional Neural Network (CNN) models.**
   🔗 **https://www.tensorflow.org/**
3. **XGBoost – A gradient boosting framework used for high-performance classification and ensemble learning.**
   🔗 **https://xgboost.readthedocs.io/en/stable/**

4. **scikit-learn – Used for data preprocessing, model evaluation, and metrics calculation.**
   🔗 **https://scikit-learn.org/stable/**
5. **OpenCV – Utilized for image reading, resizing, grayscale conversion, and other preprocessing operations.**
   🔗 **https://opencv.org/**
6. **Streamlit – Framework used for deploying the web-based application and building the user interface.**
   🔗 **https://streamlit.io/**
7. **NumPy – Used for numerical operations, image flattening, and matrix manipulation.**
   🔗 **https://numpy.org/**
8. **Pandas – Employed for data logging, storage of prediction results, and CSV management.**
   🔗 **https://pandas.pydata.org/**
9. **Matplotlib – Used for generating feature importance plots and confusion matrix visualizations.**
   🔗 **https://matplotlib.org/**
10. **Joblib – Used for saving and loading machine learning models efficiently.**
    🔗 **https://joblib.readthedocs.io/en/latest/**
