SIGNATURE VERIFICATION SYSTEM

A MACHINE LEARNING DRIVEN SIGNATURE AUTHENTICATION

TEAM MEMBERS

ABIKSHA R	2403717610622001
AKSHAYA S	2403717610622002
ALFRED RAJA SINGH A S	2403717610621003
ALLEN VICTOR B	2403717610621004
ANNALAKSHMI R	2403717610622005

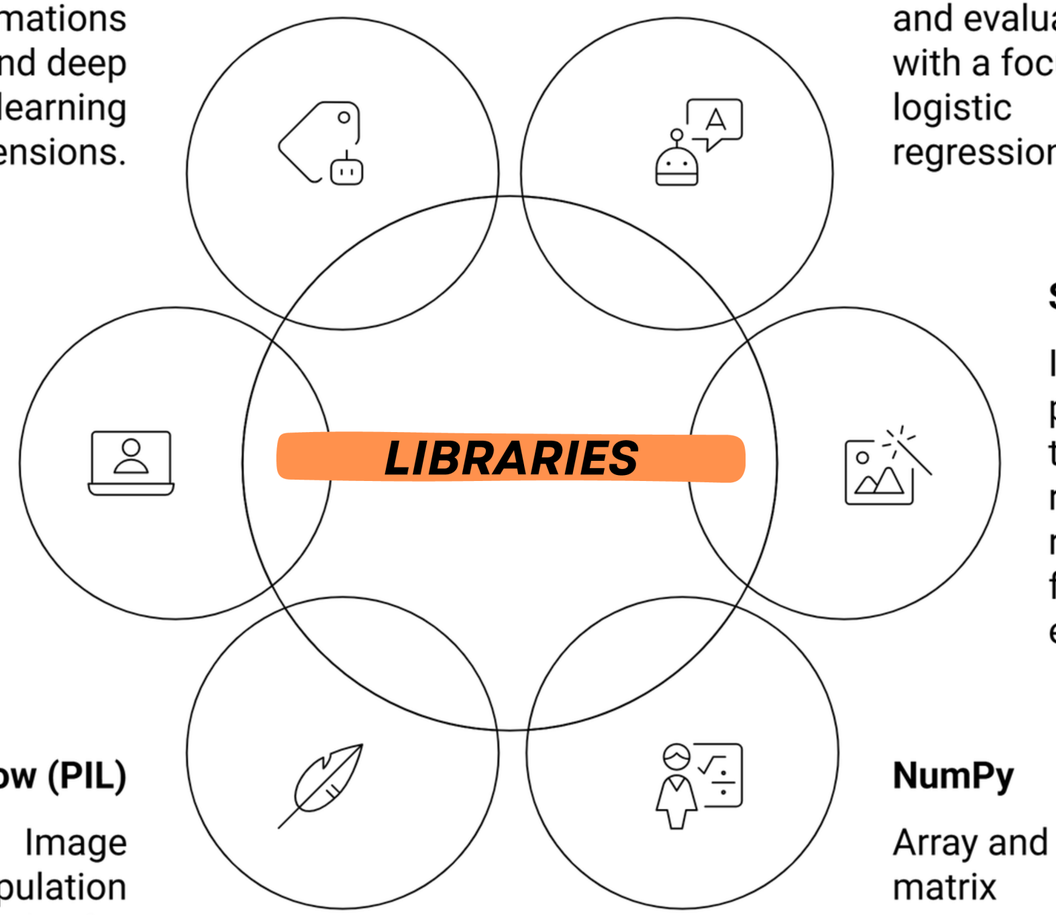
LIBRARIES USED IN THE PROJECT

Torch/Torchvision

Advanced transformations and deep learning extensions.

Scikit-learn

Model training and evaluation with a focus on logistic regression.



Gradio

Building web interfaces for model testing.

Scikit-image

Image processing tasks like reading, resizing, and feature extraction.

Pillow (PIL)

Image manipulation and display for web and app routines.

NumPy

Array and matrix operations throughout the data science workflow.

PROJECT ALGORITHM

SIGNATURE VERIFICATION PROCESS

Data Preparation

Image Organization

Signature images (genuine and forged) are organized into separate folders

Preprocessing

Image Processing

Images are read, converted to grayscale, resized to 100x100 pixels, and flattened into 1D feature vectors

Training

Model Training

Feature vectors are stacked, labeled (1 for genuine, 0 for forged), and split into training/test sets. A Logistic Regression model is trained

Prediction & Inference

Image Classification

New uploaded images undergo preprocessing and are classified by the trained model

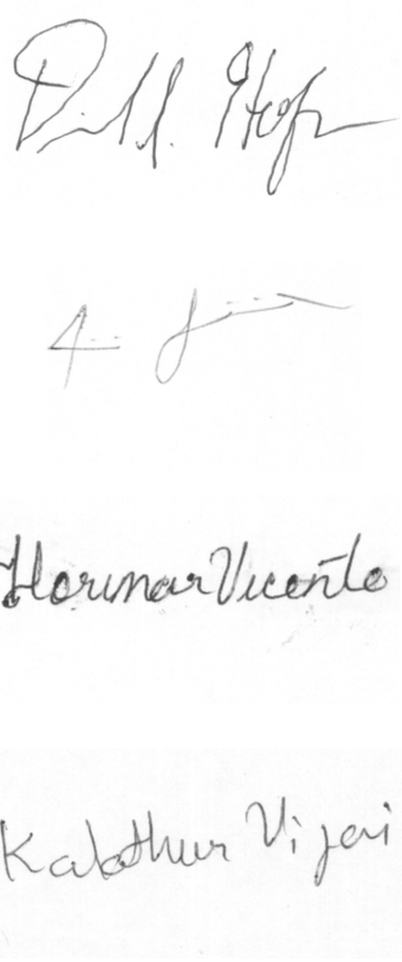
Result Display

Web Interface

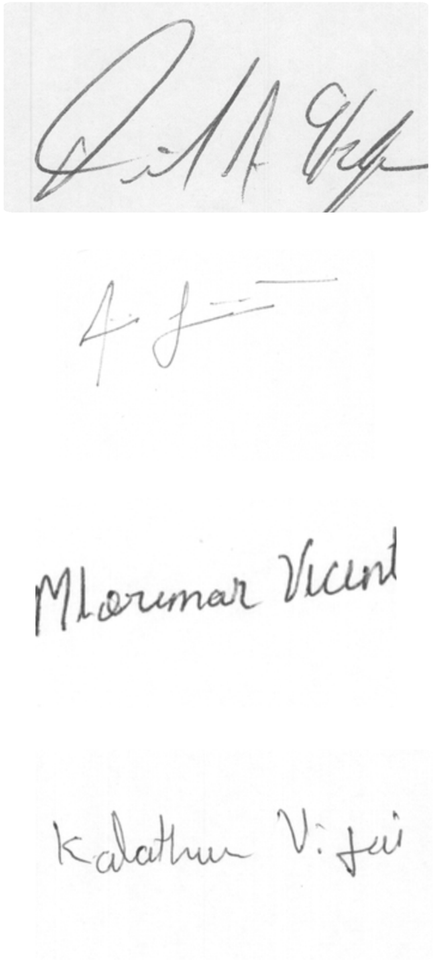
Result (“authorised” or “unauthorised”) displayed via Gradio web interface

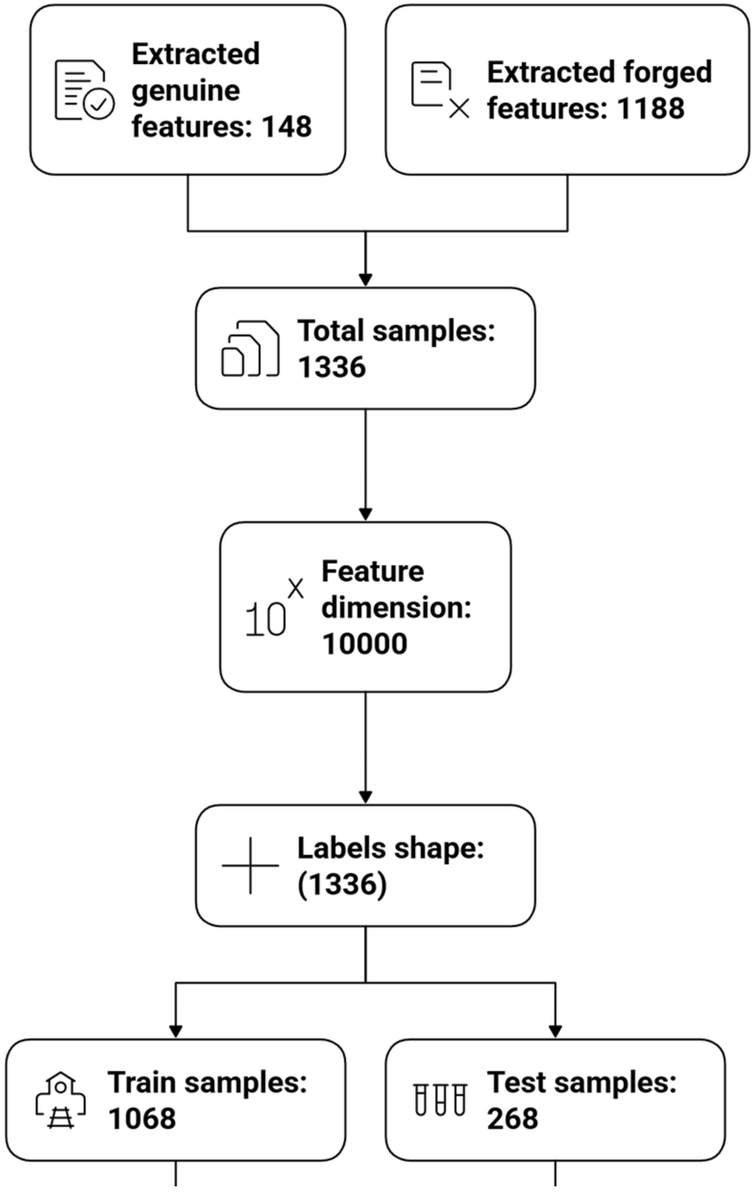
INPUT SAMPLES

genuine



forged





THE MAIN PART OF THE CODE

PREPROCESSING AND FEATURE EXTRACTION

```
def extract_features(img_path):
    img = imread(img_path)
    # If grayscale
    if img.ndim == 2:
        img_gray = img
    # If RGB
    elif img.ndim == 3:
        if img.shape[2] == 3:    # Standard RGB
            img_gray = rgb2gray(img)
        elif img.shape[2] == 4:   # RGBA
            img_gray = rgb2gray(img[..., :3])  # Drop alpha channel, use RGB only
        else:
            raise ValueError(f"Unsupported channel shape: {img.shape}")
    else:
        raise ValueError(f"Unsupported image shape {img.shape} for {img_path}")
    img_resized = resize(img_gray, (100, 100))
    return img_resized.flatten()


def load_images_from_folder(folder):
    features = []
    for ext in ["*.png", "*.jpg", "*.jpeg"]:
        for file in glob.glob(os.path.join(folder, "**", ext), recursive=True):
            try:
                features.append(extract_features(file))
            except Exception as e:
                print("Error loading:", file, e)
    return features
```

```
Train samples: 1068
Test samples: 268
```

```
Test Accuracy: 0.9365671641791045
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.97       238
           1       1.00      0.43      0.60        30

    accuracy                           0.94       268
   macro avg       0.97      0.72      0.79       268
weighted avg       0.94      0.94      0.93       268

Confusion Matrix:
 [[238   0]
 [ 17  13]]
```

TRAINING OF SAMPLES

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
y_pred = model.predict(X_test)
print("Test Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

THE MAIN PART OF THE CODE

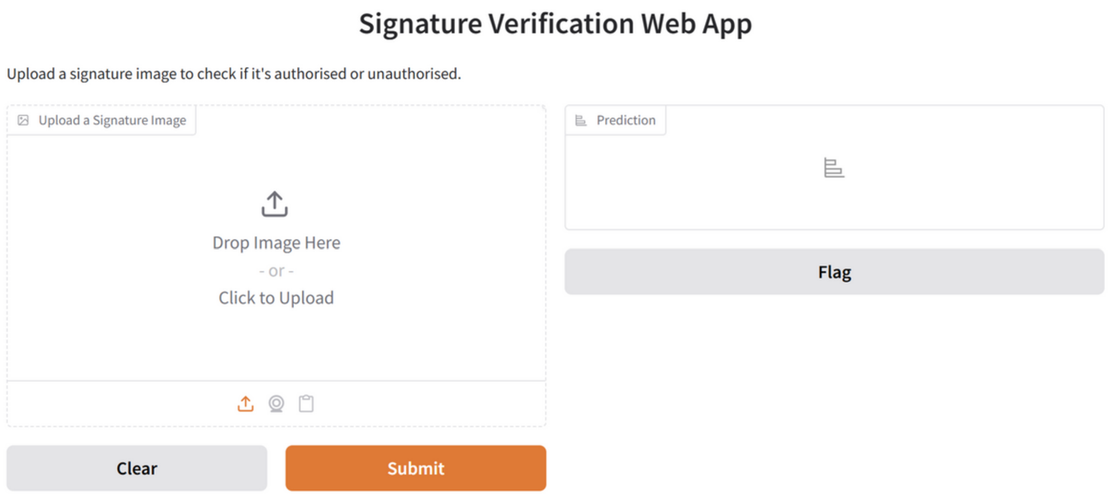
DEFINE PREDICTION FUNCTION

```
def predict_signature(image):
    try:
        img = image.convert("RGB")
        img_t = transform(img).unsqueeze(0)  # Shape: [1, C, H, W]

        # Flatten for sklearn model (since it expects 1D feature vectors)
        img_flat = img_t.view(1, -1).numpy()

        # Run the sklearn model
        # Check if the model has predict_proba
        if hasattr(model, 'predict_proba'):
            prob = model.predict_proba(img_flat)[0]
            pred = model.predict(img_flat)[0]
            confidence = max(prob)
        elif hasattr(model, 'predict'):
            # If no predict_proba, just use predict and set confidence to 1.0
            pred = model.predict(img_flat)[0]
            confidence = 1.0 # Or some other default if needed
        else:
            return "⚠️ Error: Loaded model does not have 'predict' or 'predict_proba'
method."
        # Labels (adjust if swapped)
        labels = ["unauthorised", "authorised"]

        # Ensure pred is an integer index
        predicted_label = labels[int(pred)]


        return {predicted_label: float(confidence)}

    except Exception as e:
        return f"⚠️ Error: {str(e)}"
```

GRADIO INTERFACE

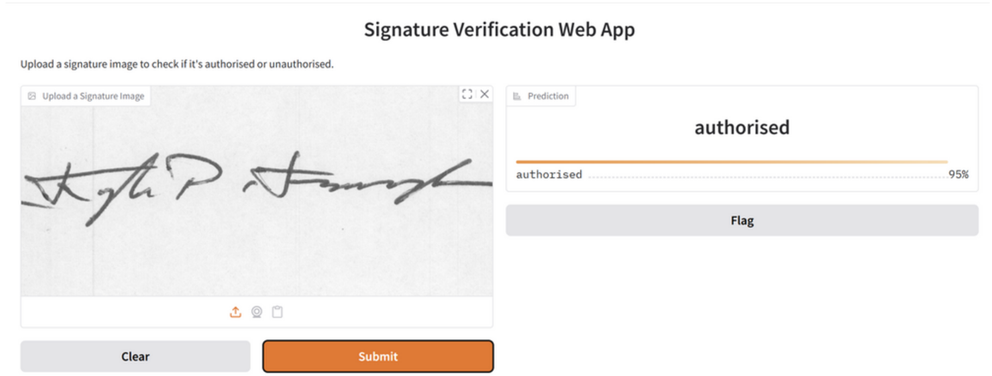
```
interface = gr.Interface(
    fn=predict_signature,
    inputs=gr.Image(type="pil", label="Upload a Signature Image"),
    outputs=gr.Label(num_top_classes=2, label="Prediction"),
    title="Signature Verification Web App",
    description="Upload a signature image to check if it's authorised or unauthorised."
)
```

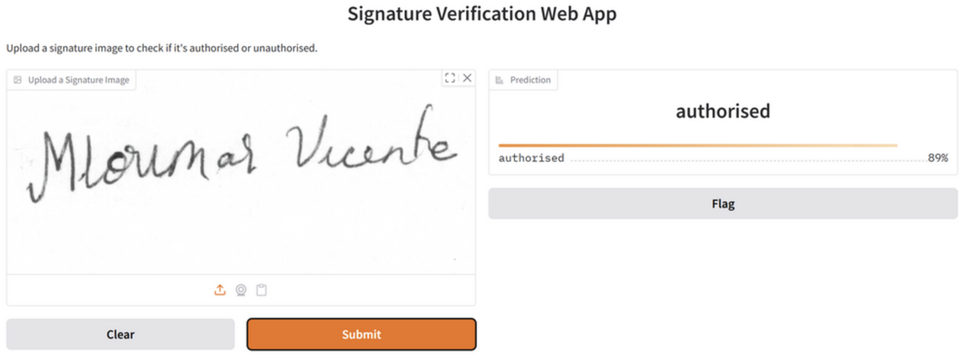
INTERFACE OUTPUT



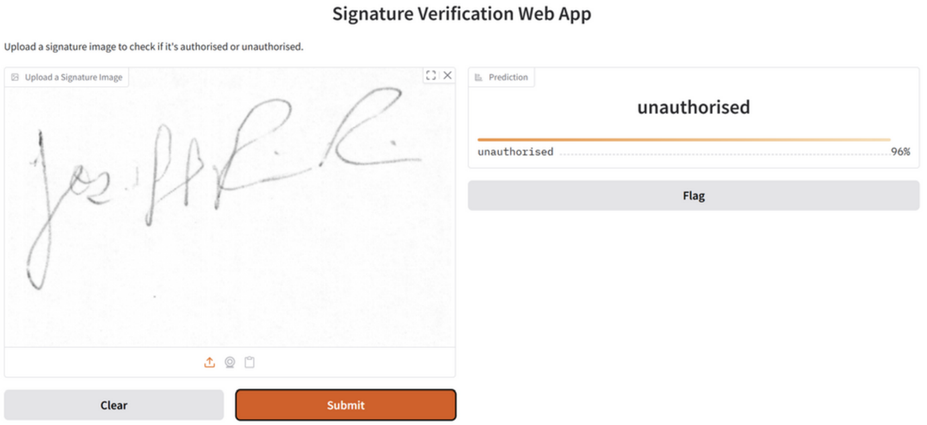
Output & Results

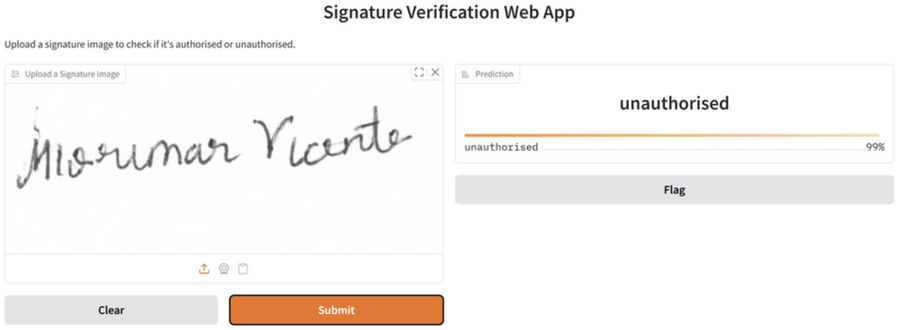
Authorised output





Unauthorised output





- *Functionality: Users upload a signature image and immediately see if it is “authorised” or “unauthorised”.*
- *Performance: Demo results show correct classification of test images, with confidence percentage.*

CONTRIBUTION OF EACH STUDENT

ALFRED A S - Data Preparation

- Collected and organized genuine and forged signature samples from multiple sources.
- Developed scripts for automated image reading, grayscale conversion, and data integrity checking.
- Implemented image preprocessing (resize, flatten) and extracted feature vectors for model input.

AKSHAYA S - Model Development & Training

- Selected and implemented the core classification algorithm (Logistic Regression) for signature authenticity detection.
- Designed data splitting strategy to ensure robust model training and avoid data leakage.
- Optimized classifier parameters and validated model performance with confusion matrix and accuracy scoring.

ABIKSHA R - Web Interface & Visualization

- Built the interactive Gradio web application for end-user signature upload and instant prediction.
- Developed result panels and integrated accuracy feedback into the prediction display.
- Managed live demo deployment using Colab and explained user flow from image selection to result .

ALLEN VICTOR B - Testing & Validation

- Created extensive test sets with diverse authorized and unauthorized signatures.
- Automated regression testing for feature extraction and classification steps.
- Analyzed model results, identified edge cases, and contributed to documentation of result interpretation.

ANNALAKSHMI R - Documentation & Integration

- Drafted comprehensive documentation covering workflow, code modules, and user instructions.
- Coordinated final slide and presentation design highlighting contributions from each team member.