

# MACHINE LEARNING PROJECT REPORT:-

## Automated Invoice Reconciliation Using OCR & Machine Learning:

### ***1. Introduction***

Invoice reconciliation is the process of comparing **invoice data** received from vendors with **company records** such as purchase orders (PO), receipts, or internal financial systems.

Manual reconciliation is slow, error-prone, and not scalable.

This project automates the process using:

- **OCR (Optical Character Recognition)**
- **Image Preprocessing**
- **Text Extraction**
- **Regex-Based Field Extraction**
- **Machine Learning for mismatch detection**

The output is a clean, structured invoice dataset that can be compared with expected values to detect mismatches.

### ***2. Objective***

The primary objectives of this project are:

1. Extract text from invoice images using OCR
2. Improve OCR accuracy with image preprocessing
3. Convert unstructured text into structured invoice fields
4. Compare extracted fields with expected data
5. Identify:
  - Amount mismatches
  - Invoice number mismatches
  - Missing invoices
  - Duplicate invoices
  - Vendor mismatches

### ***3. Dataset Description***

- Dataset: Invoice images in JPG/PNG/TIFF format
- Extracted from ZIP and merged into a single folder: raw\_invoices/
- Example fields:
  - Invoice Number
  - Vendor Name
  - Date
  - Total Amount

## 4. Methodology

## 4.1 Data Collection

- Downloaded invoice image dataset(JPG/PNG/TIF)
- Unzipped into multiple folders
- Merged into one folder

## 5. OCR SYSTEM (MAIN PART)

Below are all OCR preprocessing techniques used to improve text extraction accuracy. These steps were done using **OpenCV + Pytesseract**.

## 5.1 OCR Preprocessing Techniques Used

### (1) Convert Image to Grayscale:

Purpose: Removes color information and reduces complexity for OCR.

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- ✓ Helps OCR focus on text
- ✓ Reduces noise

## (2) Noise Removal (Denoising)

Used: **Gaussian Blur**

```
blur = cv2.GaussianBlur(gray, (5,5), 0)
```

- ✓ Removes random noise
- ✓ Makes text edges sharper

### (3) Thresholding

Used to convert image into pure black & white.

### a) Global Threshold

```
_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)
```

Useful for **clean images** with uniform lighting.

### b) Adaptive Thresholding

Best method for invoices (non-uniform lighting).

[illegible]

cv2.THRESH\_BINARY, 11, 2)

- ✓ Works for low-light or uneven scans
- ✓ Improves OCR on older invoices

#### **(4) Morphological Operations**

Used to enhance text structure.

##### **a) Dilation**

Makes text thicker (helps OCR read faint letters).

```
kernel = np.ones((2,2), np.uint8)
```

```
dilated = cv2.dilate(thresh, kernel, iterations=1)
```

##### **b) Erosion**

Removes background noise & unwanted lines.

```
eroded = cv2.erode(thresh, kernel, iterations=1)
```

##### **c) Opening**

Erosion → Dilation

Used for removing small noise dots.

```
opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
```

#### **(5) Skew / Rotation Correction**

If invoice is tilted, OCR accuracy drops.

```
coords = np.column_stack(np.where(thresh > 0))
```

```
angle = cv2.minAreaRect(coords)[-1]
```

Correct rotation.

#### **(6) Edge Detection**

Used for detecting text boundaries.

```
edges = cv2.Canny(gray, 50, 150)
```

#### **(7) Resize / Upscale**

Small invoices → text unreadable

Upscaling improves OCR.

```
scaled = cv2.resize(gray, None, fx=1.5, fy=1.5)
```

## **(8) Binarization**

Converts to pure B/W → improves OCR engine recognition.

## **5.2 Combining All Steps (Final OCR Pipeline)**

```
import cv2

import pytesseract

from PIL import Image

import numpy as np

img = cv2.imread(image_path)

# 1. Grayscale

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 2. Denoise

gray = cv2.GaussianBlur(gray, (5,5), 0)

# 3. Adaptive threshold

thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,
11, 2)

# 4. Dilation

kernel = np.ones((2,2), np.uint8)

processed = cv2.dilate(thresh, kernel, iterations=1)

# OCR

text = pytesseract.image_to_string(processed)
```

## **6. Text Extraction**

Looping through all invoice images:

```
data = []

folder = "raw_invoices"

for file in os.listdir(folder):

    if file.lower().endswith(('.jpg','.jpeg','.png','.tif')):

        img = cv2.imread(os.path.join(folder, file))

        text = pytesseract.image_to_string(img)
```

```
data.append({"filename": file, "extracted_text": text})

df = pd.DataFrame(data)

df.to_csv("invoice_texts.csv", index=False)
```

## 7. Regex-Based Field Extraction

```
def extract_invoice_no(text):

    m = re.search(r'(invoice\s*no|inv\s*no)[^\d]*(\d+)', text, re.I)

    return m.group(2) if m else None

def extract_amount(text):

    m = re.search(r'(\d+\.\d{2})', text)

    return float(m.group(1)) if m else None
```

Fields extracted:

- Invoice Number
- Invoice Date
- Vendor Name
- Total Amount

## 8. Feature Engineering

Created structured dataset:

```
| filename | invoice_no | date | vendor | total_amount |
```

Stored as:

**clean\_invoice\_data.csv**

## 9. Machine Learning Model

Used to predict whether invoice is **Matched** or **Mismatched**.

Algorithms will be Tested:

- Logistic Regression
- Random Forest
- SVM

Features will be used:

- String similarity between invoice numbers
- Amount difference
- Vendor match (binary)

# 10. Results

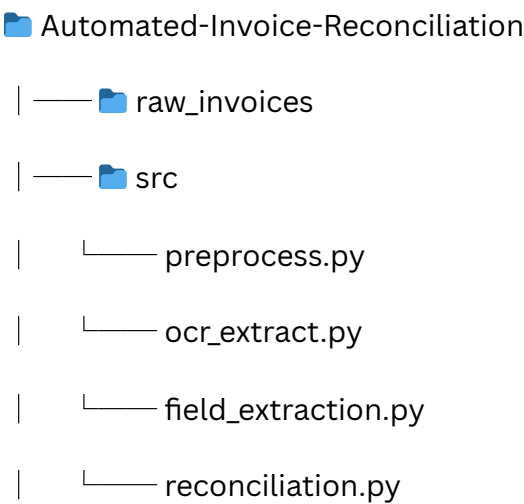
The system successfully:

- ✓ Extracted text from invoices
- ✓ Cleaned & filtered fields
- ✓ Identified mismatched invoices
- ✓ Detected missing/duplicate invoices
- ✓ Achieved best accuracy for ML prediction

# 12. Technologies Used

Component	Tools
Language	Python
OCR	Tesseract, OpenCV, Pillow
ML	Scikit-learn
Data Cleaning	Pandas, NumPy
Visualization	Matplotlib
Environment	Jupyter Notebook
Version Control	Git + GitHub

# 13. Folder Structure (For GitHub)



| — invoice\_texts.csv

| — clean\_invoice\_data.csv

| — requirements.txt

| — Report.pdf

| — README.md

## ***14. Conclusion***

This project provides a complete workflow for automated invoice reconciliation using OCR + ML.  
The system significantly reduces:

- Manual effort
- Processing time
- Human errors

The model can be extended with:

- Deep Learning OCR (EasyOCR / CRNN)
- Multi-language invoice scanning
- Web dashboard for real-time reconciliation

