PDF to Excel Converter Report

1. Introduction

In many industries, data is often stored in PDF format, making it difficult to extract structured information for further analysis. The **PDF to Excel Converter** is a tool designed to automate this process by extracting tables from PDF files and converting them into Excel spreadsheets. This project was implemented using **Python**, utilizing the **pdfplumber** library for extracting text and tables, and **pandas** for handling structured data. Additionally, a **Streamlit web application** was developed to provide a user-friendly interface.

2. Objectives

The main objectives of this project are:

- To efficiently extract tabular data from PDF documents.
- To convert extracted tables into an Excel file for easy accessibility.
- To provide a simple and interactive interface using Streamlit.
- To automate data extraction, reducing manual effort and errors.

3. Methodology

The project follows these key steps:

a. PDF Parsing:

- The application uses the **pdfplumber** library to read and extract data from PDF files.
- The text is processed to identify tabular structures.

b. Data Structuring:

- Extracted text is organized into rows and columns.
- A mapping function ensures proper alignment of data.

c. Excel Conversion:

- The structured data is stored in a **pandas DataFrame**.
- The data is written to an **Excel file** using **pandas' ExcelWriter**.

d. Streamlit Integration:

- A **Streamlit web application** was created for easy user interaction.

- Users can upload PDF files and receive an Excel file as output.

4. Implementation

The main components of the implementation include:

a. Python Script for PDF Processing:

- The script reads PDF files and processes tables using **pdfplumber**.
- Data is structured into a proper tabular format.

b. Web Application (app.py):

- A **Streamlit** interface allows users to upload PDF files.
- The application processes the file and generates a downloadable Excel file.

c. Running the Application:

- The application can be run using the command:

```
streamlit run app.py
```

- Users can upload a PDF, and the tool will generate an Excel file with extracted tables.

5. Results and Benefits

- The application successfully extracts tabular data from PDFs and converts it into Excel format.
- Users can save time and effort compared to manually copying data.
- The tool provides **accuracy** in data extraction and prevents errors.
- The **Streamlit interface** makes it easy to use, even for non-programmers.

6. Conclusion

The **PDF to Excel Converter** is a valuable tool for professionals working with tabular data in PDF documents. By automating the extraction and conversion process, it enhances efficiency and minimizes manual effort. The integration of **Streamlit** makes the application accessible and easy to use. Future improvements could include support for **image-based PDFs using OCR** and **enhanced table detection algorithms**.

---

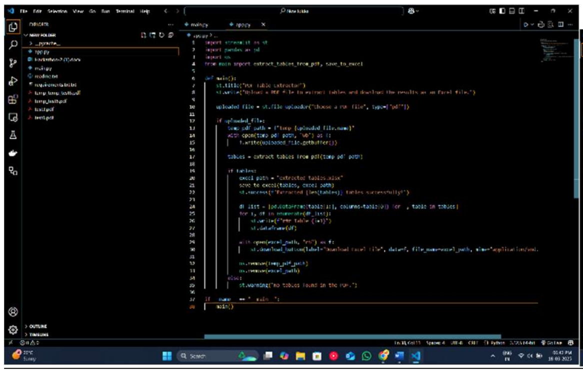
7. Future Enhancements

- Adding **OCR support** for scanned PDFs using Tesseract OCR.
- Improving **table detection** for more complex document layouts.

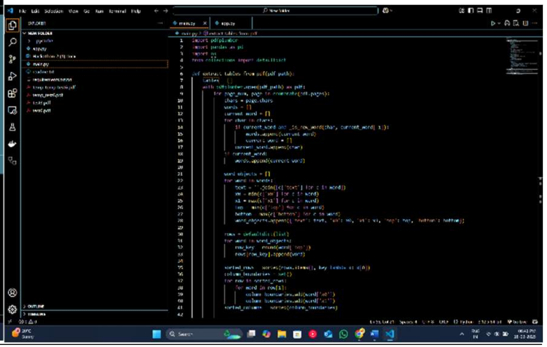
- Enhancing **user interface** with additional customization options.

This project successfully demonstrates the power of **Python** in automating data extraction and making information more accessible. It is a useful tool for professionals dealing with data extraction challenges in PDFs.

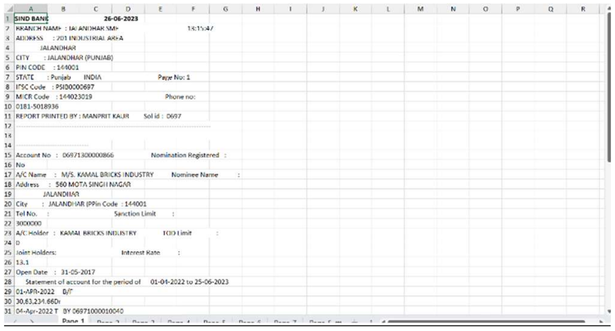
<u>Screenshots</u>



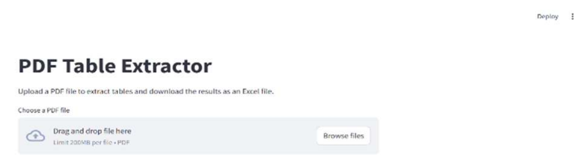
```
import streamlit as st
import pandas as pd
import os
from main import extract_tables_from_pdf, save_to_excel

def main():
    st.title("PDF Table Extractor")
    st.write("Upload a PDF file to extract tables and download the results as an Excel file.")

    uploaded_file = st.file_uploader("Choose a PDF file", type=["pdf"])

    if uploaded_file:
        temp_pdf_path = f"temp_{uploaded_file.name}"
        with open(temp_pdf_path, "wb") as f:
            f.write(uploaded_file.getbuffer())

        tables = extract_tables_from_pdf(temp_pdf_path)

        if tables:
            excel_path = "extracted_tables.xlsx"
            save_to_excel(tables, excel_path)
            st.success(f"Extracted {len(tables)} tables successfully!")

            df_list = [pd.DataFrame(table[1:], columns=table[0]) for _, table in tables]
            for i, df in enumerate(df_list):
                st.write(f"### Table {i+1}")
                st.dataframe(df)

            with open(excel_path, "rb") as f:
                st.download_button(label="Download Excel File", data=f, file_name=excel_path, mime="application/vnd.

            os.remove(temp_pdf_path)
            os.remove(excel_path)
        else:
            st.warning("No tables found in the PDF.")

if __name__ == "__main__":
    main()
```



```
import pdfplumber
import pandas as pd
import os
from collections import defaultdict

def extract_tables_from_pdf(pdf_path):
    tables = []
    with pdfplumber.open(pdf_path) as pdf:
        for page_num, page in enumerate(pdf.pages):
            chars = page.chars
            words = []
            current_word = []
            for char in chars:
                if current_word and _is_new_word(char, current_word[-1]):
                    words.append(current_word)
                    current_word = []
                current_word.append(char)
            if current_word:
                words.append(current_word)

            word_objects = []
            for word in words:
                text = ''.join([c['text'] for c in word])
                x0 = min(c['x0'] for c in word)
                x1 = max(c['x1'] for c in word)
                top = min(c['top'] for c in word)
                bottom = max(c['bottom'] for c in word)
                word_objects.append({'text': text, 'x0': x0, 'x1': x1, 'top': top, 'bottom': bottom})

            rows = defaultdict(list)
            for word in word_objects:
                row_key = round(word['top'])
                rows[row_key].append(word)

            sorted_rows = sorted(rows.items(), key=lambda x: x[0])
            column_boundaries = set()
            for row in sorted_rows:
                for word in row[1]:
                    column_boundaries.add(word['x0'])
                    column_boundaries.add(word['x1'])
            sorted_columns = sorted(column_boundaries)
```



	A	B	C	D	E	F
1	SIND BANK			26-06-2023		
2	BRANCH NAME : JALANDHAR SME					13:15:47
3	ADDRESS : 201 INDUSTRIAL AREA					
4	JALANDHAR					
5	CITY : JALANDHAR (PUNJAB)					
6	PIN CODE : 144001					
7	STATE : Punjab INDIA				Page No: 1	
8	IFSC Code : PSIB0000697					
9	MICR Code : 144023019				Phone no:	
10	0181-5018936					
11	REPORT PRINTED BY : MANPRIT KAUR			Sol id : 0697		
12	--					
13						
14	----------------------					
15	Account No : 06971300000866				Nomination Registered :	
16	No					
17	A/C Name : M/S. KAMAL BRICKS INDUSTRY				Nominee Name :	
18	Address : 560 MOTA SINGH NAGAR					
19	JALANDHAR					
20	City : JALANDHAR (PPin Code : 144001					
21	Tel No. :			Sanction Limit :		
22	3000000					
23	A/C Holder : KAMAL BRICKS INDUSTRY				TOD Limit :	
24	0					
25	Joint Holders:			Interest Rate :		
26	13.1					
27	Open Date : 31-05-2017					
28	Statement of account for the period of 01-04-2022 to 25-06-2023					
29	01-APR-2022 B/F					
30	30,63,234.66Dr					
31	04-Apr-2022 T BY 06971000010040					

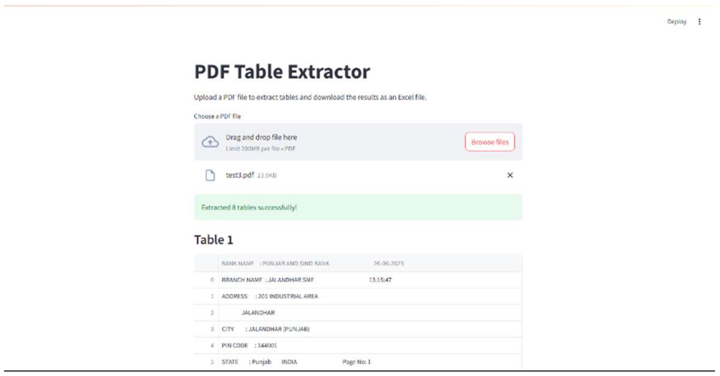


PDF Table Extractor

Upload a PDF file to extract tables and download the results as an Excel file.

Choose a PDF file

Drag and drop file here
Limit 200MB per file • PDF



PDF Table Extractor

Upload a PDF file to extract tables and download the results as an Excel file.

Choose a PDF file

Drag and drop file here
Limit 200MB per file • PDF

test3.pdf 23.9KB

Extracted 8 tables successfully!

Table 1

	BANK NAME : PUNJAB AND SIND BANK	26-06-2023
0	BRANCH NAME : JALANDHAR SME	13:15:47
1	ADDRESS : 201 INDUSTRIAL AREA	
2	JALANDHAR	
3	CITY : JALANDHAR (PUNJAB)	
4	PIN CODE : 144001	
5	STATE : Punjab INDIA Page No: 1	

Main.py Code

```python
def extract_tables_from_pdf(pdf_path):
    tables = []
    with pdfplumber.open(pdf_path) as pdf:
        for page_num, page in enumerate(pdf.pages):
            chars = page.chars
            words = []
            current_word = []
            for char in chars:
                if current_word and _is_new_word(char, current_word[-1]):
                    words.append(current_word)
                    current_word = []
                current_word.append(char)
            if current_word:
                words.append(current_word)

            word_objects = []
            for word in words:
                text = ''.join([c['text'] for c in word])
                x0 = min(c['x0'] for c in word)
                x1 = max(c['x1'] for c in word)
                top = min(c['top'] for c in word)
                bottom = max(c['bottom'] for c in word)
                word_objects.append({'text': text, 'x0': x0, 'x1': x1, 'top':
top, 'bottom': bottom})

            rows = defaultdict(list)
            for word in word_objects:
                row_key = round(word['top'])
                rows[row_key].append(word)

            sorted_rows = sorted(rows.items(), key=lambda x: x[0])
            column_boundaries = set()
            for row in sorted_rows:
                for word in row[1]:
                    column_boundaries.add(word['x0'])
                    column_boundaries.add(word['x1'])
            sorted_columns = sorted(column_boundaries)

            table = []
            for row_key, words in sorted_rows:
                row_dict = {}
                for word in words:
                    col_start = max([c for c in sorted_columns if c <=
word['x0']], default=sorted_columns[0])
                    col_end = min([c for c in sorted_columns if c >=
word['x1']], default=sorted_columns[-1])
                    col_index = sorted_columns.index(col_start)
                    row_dict[col_index] = word['text']
                max_col = max(row_dict.keys()) if row_dict else 0
                row_data = [row_dict.get(i, '') for i in range(max_col+1)]
                table.append(row_data)

            if table:
                tables.append((f"Page_{page_num+1}", table))

    return tables

def _is_new_word(new_char, prev_char):
    return (new_char['x0'] - prev_char['x1'] > 3 or new_char['top'] -
prev_char['top'] > 5)

def save_to_excel(tables, output_path):
    with pd.ExcelWriter(output_path) as writer:
        for name, table in tables:
            df = pd.DataFrame(table[1:], columns=table[0])
            df.to_excel(writer, sheet_name=name, index=False)
```