PDF Report Generation Using Weasyprint And Pandas In Python

[1]Chittoor Ravimanya Susrith, [2]Prof. Pavithra H

[1]Student, [2]Assistant Professor,
[1,2]Department of Computer Science & Engineering,
[1,2] RV College of Engineering, Bangalore, India.

Abstract : Reports are essential in every field. Reports will provide useful information that can be used to help develop marketing plans, manage workforce and improve decision-making thereby increasing the performance of an organization. Automating the process of generating reports is a tedious process. Automating the report generation process can reduce efforts, time taken as compared to manual report generation. This paper discusses an implementation of the automated report generation process using pandas, jinja, weasyprint libraries in python. These libraries provide many functions to analyze the data and finally generate pdf reports of the useful underlying information extracted from the data. Pandas library is used to apply some data analytics operations and extract some useful information from the data set. The pdf generating engine used is the weasyprint. This paper gives an understanding about the functions that are required to manipulate pandas DataFrames, plot graphical visualizations, generate pdf reports with DataFrames and plots using weasyprint.

IndexTerms – Report generation, Pandas, Weasyprint, Jinja.

I. INTRODUCTION

In the present world automation has very high significance. Everyone is working towards the goal to reduce manual intervention. This paper discusses generating pdf reports of excel files using pandas data frames and weasyprint library. Pandas is a flexible and fast tool for data analysis and manipulation that is used to work on data sets using data frames [6]. It is built on the top of the Python language. Weasyprint is a rendering engine for CSS and HTML that can be used to convert the output data frames and graphical visualizations into pdf. Jinja is a web template engine for Python. It is used to render HTML templates in the implementation. We write CSS and HTML codes and convert that output into PDF using functions in the weasyprint library. There are many libraries for pdf generation in python like fpdf, reportlab, weasyprint e.t.c. But the weasyprint library provides a greater flexibility in manipulating the PDF output by using HTML and CSS as compared to other libraries for PDF generation. The data used for study are the weekly rosters that are taken from a multinational company which contains employee data. Roster files are excel files that contain data of employees of a company.

The aspect used for example in report generation is the headcount of employees. We notice the changes in the headcount every week and check for the presence of any change or anomaly. The main aim of this study is to define an implementation to generate reports that show the data of changes in the headcount of employees on a weekly basis based on various attributes like location, pyramid, function e.t.c. The final output of the implementation is a PDF report that contains various tables and graphs that is user friendly and understandable by non-technical staff also.

Rest of the paper is organized as follows, Section II contains the related work of pandas library since there is no previous work regarding any pdf generation library, Section III describes the methodology of the implementation, Section IV explains the implementation with appropriate code snippets, section V describes results and discussion, Section VIII concludes research work with future directions and improvements.

II. RELATED WORK

In [1] proposed by Wes McKinney the author described the Pandas library. It explained how to work on data frames, their attributes and also functions from Pandas library.

Satish Premshankar Yadav and Adarsh S.K Singh have explained about the functions from Pandas and Scipy libraries which are necessary for data analytics. It included functions required for data manipulation and interpretation like groupby, plot and many other numpy functions also [2].

I.Stančin and A. Jović have considered more than twenty libraries that are related to data mining and data analysis and compared them. They concluded by recommending that pandas library work best with data preparation because other libraries in the data preparation field have bigger issues than that of python [3].

In the paper [4] by Wes McKinney the author discussed some of the specific design issues encountered in the course of developing pandas. They also used few examples and comparisons with the R language. They gave information on using python language for statistical computing and data analysis.

Pandas is one of the essential libraries in python. It provides functions and rich data structures that makes us work with structured data in a simple, quick and eloquent manner. The main and most commonly used object in pandas is the DataFrame. It is a two dimensional column-oriented data structure. It has labels for both rows and columns [5].

III. METHODOLOGY

The data used in this study is the roster data of a multinational company that contains information about employees of the company. Rosters files are of excel format. A roster is a list of employees of an organization or a list of people who are available to do a particular job. They contain details of employees like name, id, address, work location, pyramid, function, ethnicity, gender, business e.t.c. The rosters will be in the google drive and the system must be able to extract rosters from google drive.

First the required most latest roster names are found and the corresponding roster files are extracted from the google drive. Once the roster files are extracted pandas, data frames are created out of the extracted rosters. The data frames contain nothing but the whole roster.

The first step with data frames is data pre-processing. Data preprocessing is one of the techniques of data mining that involves converting raw data into a readable format. Real world data is often inconsistent, incomplete and is likely to contain many types of errors. Data preprocessing is a method to resolve such types of issues. In this step all the undefined, unwanted and null values are removed from the data frames.

After preprocessing the data frames of rosters, the data frames will be modified, manipulated to arrive at a final output. Operations like groupby, merge, filter, remove columns, add columns e.t.c would be applied on the data frames and final data frames are created and saved. Once the final data frames are created graphical plots are generated based on the final data frames and saved.

The final data frames and graphs created are used to generate pdfs using weasyprint and jinja. Weasyprint is an engine for rendering HTML and CSS that can be used to convert the output data frames and graphical visualizations into pdf. Jinja is a web template engine for Python. Thus we write CSS and HTML codes and convert the output of the HTML template rendered by jinja into PDF using functions in the weasyprint library. So the final output of the implementation is a PDF report that contains various tables and graphs.

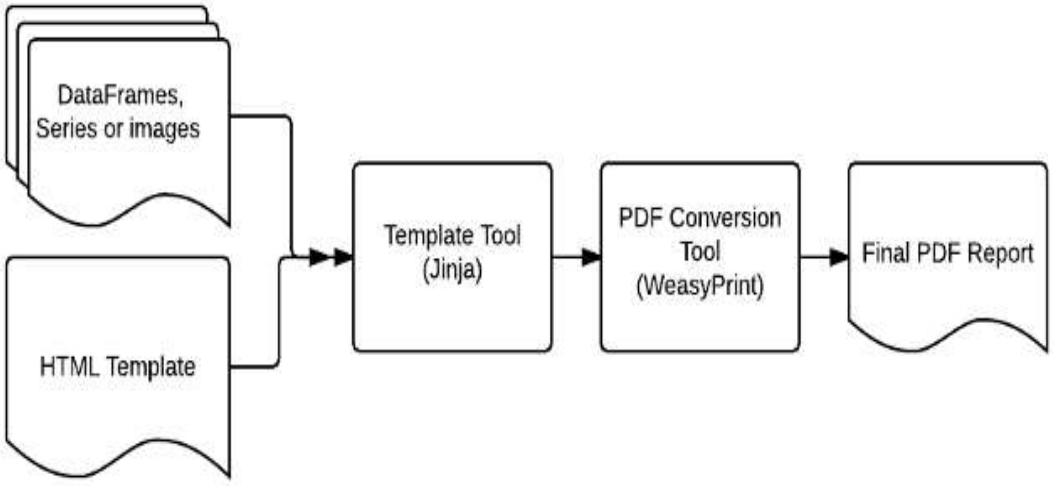

Figure 1: Block diagram of working of weasyprint library

IV. IMPLEMENTATION

In this paper we will concentrate on how to implement the automated pdf generation system using python. The main libraries used are Pandas and weasyprint. The data store used to extract the data and save the output pdf reports is google drive.

The first step in the implementation is to import necessary libraries which are pandas, weasyprint, jinja2, os, re and google.colab. In order to access data from google drive we need to mount the drive. This can be done by the following function.

```
drive.mount('/content/drive')
```

Once we mount the drive we access the list of names of weekly rosters from the corresponding folder in the drives using the following function.

```
os.listdir('/content/drive/folder_name')
```

From the list of file names extracted from the drive the two latest weekly files are found by applying regular expressions on file names. Once the file names are found the corresponding files are extracted. After extracting the rosters pandas data frames are to be created out of the extracted rosters. It can be done by the following function.

```
df = pandas.read_excel(file_name)
```

After the data extraction module comes the important part, the data comparison module. A function is used for implementing this feature. The function is used for data preparation. It involves data preprocessing, adding extra columns, removing unnecessary

columns and many operations like groupby, merge, filter e.t.c. The final output of this module is a compact data frame that contains summary statistics. The syntax for the functions is as follows

To group data frame by a column on count:

```
df_groupby_column = df.groupby(by = [column]).count()
```

To merge two data frames (perform an inner join):

```
df_merged = df1.merge(df2, on = column,how="inner")
```

To drop columns from data frame:

```
df.drop(columns = [column_list], inplace = True)
```

A sample data frame looks as in Figure 2:

	Classification	Employee count current	Change in Employee count	Hires and Exits
0	FULL-TIME	5538	DECREASED by 12	1
1	PART-TIME	416	DECREASED by 1	0
2	TEMPORARY	61	DECREASED by 1	0

Figure 2. Snapshot of a final dataframe

Once the final data frames are generated graphical representations are generated for a better understanding to the users. Plots of data frames can be created by using the plot method. The syntax is as follows

```
df.plot(x = column,y = [column_list], figsize, kind)
```

X is label for x-axis, y attribute allows plotting of one column versus another, figsize is used to define the size of the plot and kind attribute is used to specify the type of plot like bar, horizontal bar, histogram, box e.t.c. After the generation of plots they will be saved by using the following code.

```
plt.savefig('df_plot.png')
```

At this point data is ready to generate the report where data consists of data frames and plots that are saved. Now we create a jinja environment which is the core component of jinja that contains important shared variables like filters, tests, configuration and others. Once it is created we get the HTML template from local directory, The code snippet is as given below

```
env = Environment(loader=FileSystemLoader('.'))
template = env.get_template("test.html")
```

Now the final step is to pass template variables, render the template and export the output into pdf format. Template variables would be passed by creating a dictionary and populating it with variables we want to pass to the template. Templates can be rendered using the render method. This would create a string that we would pass it to the pdf generator i.e. weasyprint to generate pdf. Adding a style sheet makes the pdf aesthetic otherwise which looks ugly. We can modify the HTML and CSS aspects like background colours, fonts, spacing, table styles e.t.c according to our report appearance requirements. The code snippet is as follows:

```
template_vars = {"Title":"Report","dataframe":df.to_html(),
"dataframe_plot":"df_plot.png"}

html_out = template.render(template_vars)

HTML(string=html_out).write_pdf("report.pdf", stylesheets=["style.css"])
```

This would store the report in the current directory which we can access directly.

V. RESULTS AND DISCUSSION

The ultimate result of the implementation is a pdf report that consists of pandas data frames and corresponding plots formatted in a user-friendly and aesthetic manner. As we can see in the Figure 2 the output contains columns: comparing attribute, present employee count, change in employee count compared to the previous week and hires and exits that depicts the effect of the people joined and left the company in the current week on the change in headcount of the employees. The comparison of headcount of employees on a weekly basis was considered as an example to generate dataframes and graphical plots and use them for generating reports. Figure 2 shows a sample snapshot of the generated PDF report.

Comparison based on Classification

Classification	Employee count current	Change in Employee count	Hires and Exits
FLOATER	65	INCREASED by 1	1
PART-TIME	417	INCREASED by 1	4
SHORT HOUR	29	INCREASED by 1	1
TEMPORARY	63	INCREASED by 2	1
FULL-TIME	5573	INCREASED by 35	49

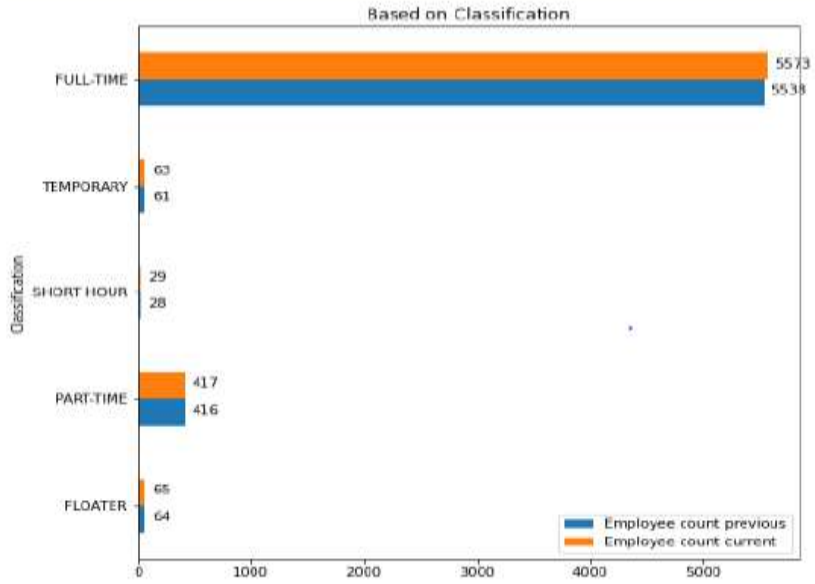

Figure 3. Snapshot of the output PDF report

VI. CONCLUSION

This paper presents a way to implement an automated report generation process using pandas and weasyprint in python. This implementation could automate the process of report generation, otherwise one should work manually on making it which is more time taking. We believe that this study will help many teams across the globe that work on generating reports by making the process of generating reports automated, time saving, efficient and reliable. The PDF generating engine weasyprint allows a greater flexibility to users by allowing them to manipulate the output even in the case of pandas DataFrames in a numerous desirable ways through HTML and CSS. It enables users to modify many parameters of the output PDF like layout, background, colours, fonts, table styles e.t.c in such a way the final pdf report would be aesthetic and understandable by everyone.

Our future work is to include report delivery process also. It involves fetching data of required people and delivering them the generated reports in an automated manner whenever there is a requirement. This will reduce manual overhead for the process of delivery of reports.

REFERENCES

[1] Wes McKinney, "pandas: a Foundational Python Library for Data Analysis and Statistics", *Python High Performance Science Computer*, 2011.

[2] Satish Premshankar Yadav, Adarsh S.K Singh, "Big Data Analytics with Pandas and Scipy Python Tools", International Research Journal of Engineering and Technology, Volume 7, Issue 6, 2020.

[3] Stančin and A. Jović, "An overview and comparison of free Python libraries for data mining and big data analysis", 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija Croatia, 2019.

[4] Wes McKinney, "Data Structures for Statistical Computing in Python", In the Proceedings of the 9th Python in Science Conference, 2010.

[5] Wes McKinney, "Python for Data Analysis", O'Reilly, pp. 3-6, 2012.

[6] Akshansh Sharma, Firoj Khan, Deepak Sharma, Dr. Sunil Gupta, "Python: The Programming Language of Future", International Journal of Innovative Research in Technology, Volume 6, Issue 12, 2020.