SCHOOL OF COMPUTER SCIENCES

UNIVERSITI SAINS MALAYSIA

CPC152:Foundations and Programming for Data Analytics

Assignment (20%)

(Individual Works)

15 May 2023

Prepared by:

Name:	Matric Number:	Student Email:
DANISH RAIMI BIN FAZRUL EDLIN	164108	danishraimi77@student.usm.my

Lecturer:

Ts Dr Chew XinYing

I. Abstract

This report we will compare two most popular Python Data Visualization Libraries to start with .The two well-known Python visualisation libraries Matplotlib and Seaborn will be compared in this report , along with an examination of their advantages and disadvantages.We will highlights for this two Python Data Visualisation Libraries interm of what can both of them can do , and what limitation of each of them.We will also have Discussion and recommendation for implementation for overall this research . This report will be useful for Data Analysts and Scientists students who are looking to compare these libraries and choose the one that best suits their needs.

II. Introduction

Data analysis is currently the most in-demand skill in the world. Therefore, before beginning to learn about data analytics, a person must select their Python Data Visualisation Library as their initial startup tool to analyse the raw data and to learn how to shape the data into information . Python was created by Guido van Rossum, and first released on February 20, 1991(*Python®)* .Python has been around since 1991 and is the programming language that data analysts most frequently employ.Due to how simple and fast it is to use, it supports a wide range of data analytics operations, including data gathering, analysis, modelling, and visualisation. In order for Data analyst to shape and model the data . it need to have a library to done it . There is a lot of Library that can be used , and the popular one is Matplotlib and Seaborn. Powerful Python data visualisation toolkit Matplotlib was first made available in 2003. It is a popular option for data analysis and visualisation because Seaborn, which was introduced in 2012, builds on Matplotlib and offers a high-level interface for producing appealing and instructive statistical visuals.

III. Part (1)

1. Highligths of the Python Data Visualisation Libraries

(A) Matplotlib

Matplotlib is a data visualization library for Python.It was first realeased in 2003 and now one of the most widely used data visualisation libraries in the world . This library was built by a John Hunter who is along with several contributors, and it had put in a greater amount of time into prompting this software used by every scientist and philosopher across the globe.(Sial, A. H., Rashdi, S. Y. S., & Khan, A. H.,2021).It allows for a high degree of customization , enabling users to adjust everything from the color and size of marker to the font and style of the text . It can be used in combination with other libraries , such as NumPy and Pandas, to create more complex visualizations . It has also been used to create figures of publishing quality for tasks like data visualisation, plotting in scripts, and interactive web visualisation in Python.

(B) Seaborn

Seaborn Module in Python is a wrapper around matplotlib so it uses matplotlib and allows you to design and to create attractive and good- looking statistical graph and provides high level interface , it makes it easier for you to create a graph.Seaborn has been around for easily Visualize a data from a dataset .It is a alternative to a matplotlib to design a graph for a data.This library offer pre-installed colour schemes to make the graph Visually beautiful .

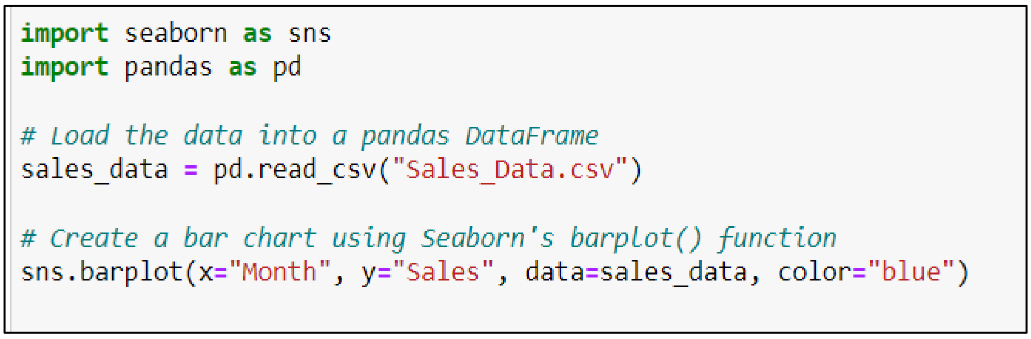
2. Comparison among the 2 Python Data Visualisation Libraries

This comparison will compare by Ease of use of this two Python libraries , Customization option , Plot types that are avaible , lastly is Community support for this Python libraries.Seaborn Library was craeted by Michael Wascom

(A) Ease of Use (Which one is more easier to use ?)

Both Matplotlib and Seaborn can be use to visualize the data but there is a difference between them in term of syntax.Generally , Seaborn is considered easier to use than Matplotlib for creating visually appealing and informative plots quicky , especialy for users who are new to data visualization , we can take programming language for example , C++ has more complex syntax than Python syntax in general , same like Mathplotlib and Seaborn . Mathplotlib has more complex syntax than syntax of Seaborn .In contrast , Mathplotlib has a lower level of abstraction , which means that it provides more flexibility and control over the detailds of a plot while Seaborn has a higher level of abstraction which means that it offer simpler and more concise functions to creates advances visualisations with fewer lines of codes.

Figure 1 and Figure 2 is the example of code by Mathplotlib and Seaborm to create barplot graph :

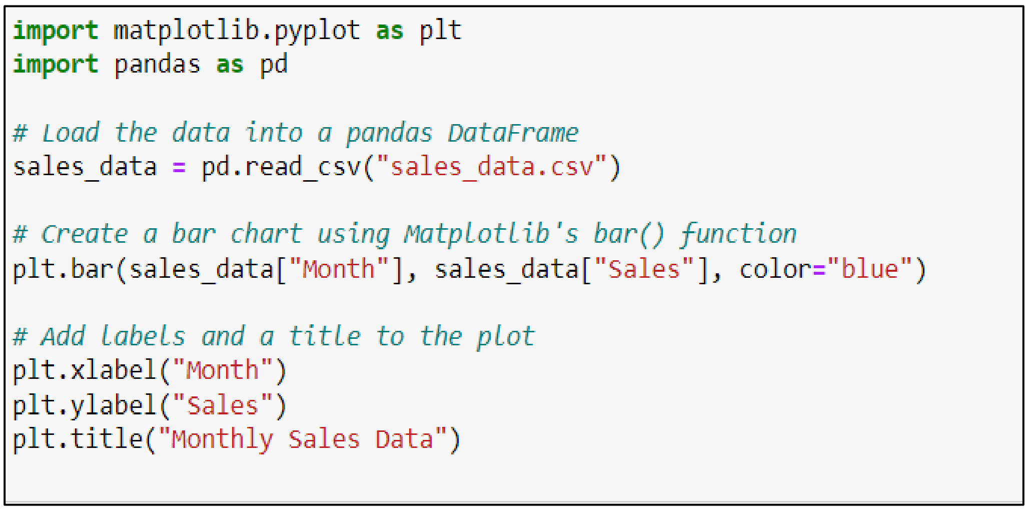


```
import seaborn as sns
import pandas as pd

# Load the data into a pandas DataFrame
sales_data = pd.read_csv("Sales_Data.csv")

# Create a bar chart using Seaborn's barplot() function
sns.barplot(x="Month", y="Sales", data=sales_data, color="blue")
```

Figure 1 Code by Seaborn



```python
import matplotlib.pyplot as plt
import pandas as pd

# Load the data into a pandas DataFrame
sales_data = pd.read_csv("sales_data.csv")

# Create a bar chart using Matplotlib's bar() function
plt.bar(sales_data["Month"], sales_data["Sales"], color="blue")

# Add labels and a title to the plot
plt.xlabel("Month")
plt.ylabel("Sales")
plt.title("Monthly Sales Data")
```

Figure 2 Code by Matplotlib

As we can see code from figure 1 has a simpler line of codes than Figure 2 but has the same kind of visual outputs .

(B) Customization Color and Style

For data visualisation, both Matplotlib and Seaborn provide a lot variety of options for modification for example for color and style .However, Seaborn has more high-level functions that allow more practical customization options and better default settings that means it easier to use it , whereas Matplotlib is more low-level and necessitates more code to produce complicated visualisations and needed more of modification toward the plot.

COLOR

In term of customization of color,There is a clear distinction between Matplotlib and Seaborn in terms of colour customization. With the flexibility to select colours via RGB values, hexadecimal codes, or predefined colour names, Matplotlib provides a wide range of colour options. While having this much control might be advantageous, it can also make it difficult to select acceptable colours and produce figures that are pleasing to the eye. Individual plot features like lines, markers, fill colours, and backgrounds can be changed with Matplotlib, but this needs more manual work and trial & error.

As Analytics Vidhya (2023) explains that Seaborn provides beautiful default styles and color palettes to make statistical plots more attractive on the other hand, adopts a different strategy by making colour customisation simple. It offers a selection of well created and eye-catching colour palettes that you can quickly use for your plots. You can build beautiful colour schemes using these pre-defined colour palettes without having to make a lengthy colour selection. Additionally, Seaborn provides features that automatically allocate colours depending on category data, minimising the need to manually do colour assignment. The default colour schemes in Seaborn have been chosen with care to be pleasing to the eyes and appropriate for many kinds of data visualisations.

STYLE

In term of styling , Plot styles in Matplotlib can be completely customised, providing you complete control over how your charts look. Aspects like line styles, marker styles, colours, font sizes, and more can all be manually changed. This amount of customisation allows you versatility but might take a lot of work, especially if you want to maintain a uniform appearance across different plots. Even though Matplotlib offers a few predefined styles like 'ggplot' or 'fivethirtyeight', these styles might not be as comprehensive or aesthetically pleasing as those provided by Seaborn.

By providing a variety of pre-defined styles, Seaborn makes the process of plot styling more simpler. These aesthetic appearance are immediately applied to your plots by these styles, giving them a mre beautiful and appealing appearance. You can easily change the appearance of your plots by picking a certain style, such as "darkgrid," "whitegrid," or "ticks." The default styles in Seaborn have been carefully crafted to look good and work with different kinds of data visualisations. They let you avoid the time and effort required for manual customization.

In conclusion, Matplotlib provides for considerable customisation of plot layout but may take more time and effort. The pre-defined styles offered by Seaborn, on the other hand, provide a quick and simple approach to give your plots a consistent and eye-catching appearance.

(C) Plot Types

Popular Python libraries used for data visualisation include Matplotlib and Seaborn. They provide a range of plot styles and methods for making visualisations.

In terms of Plot types, Matplotlib can support a variety of Plot for example line plots , scatter plots , barplots , boxplots , piechart and more. Matplotlib provides a tools for building more complex Plots , but it often requires more code and effort to create visually plots compared to Seaborn .While Seaborn offer specialised plot types , such as Categorical Plots and Distribution Plots. It offers a streamlined method with an focus on producing aesthetically or beuatiful pleasing and educational statistical graphs or visual of data .

Figure 3 and Figure 4 are the example of Matplotlib graph and Seaborn graph:

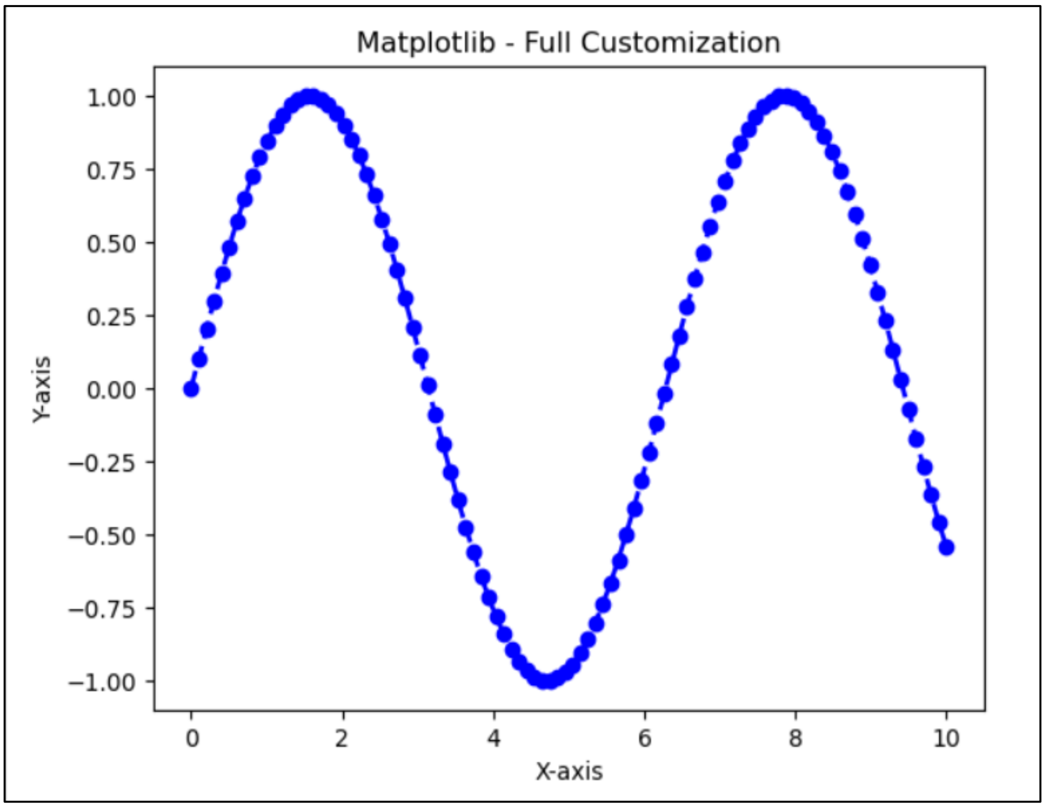


Figure 3 Plots by Matplotlib

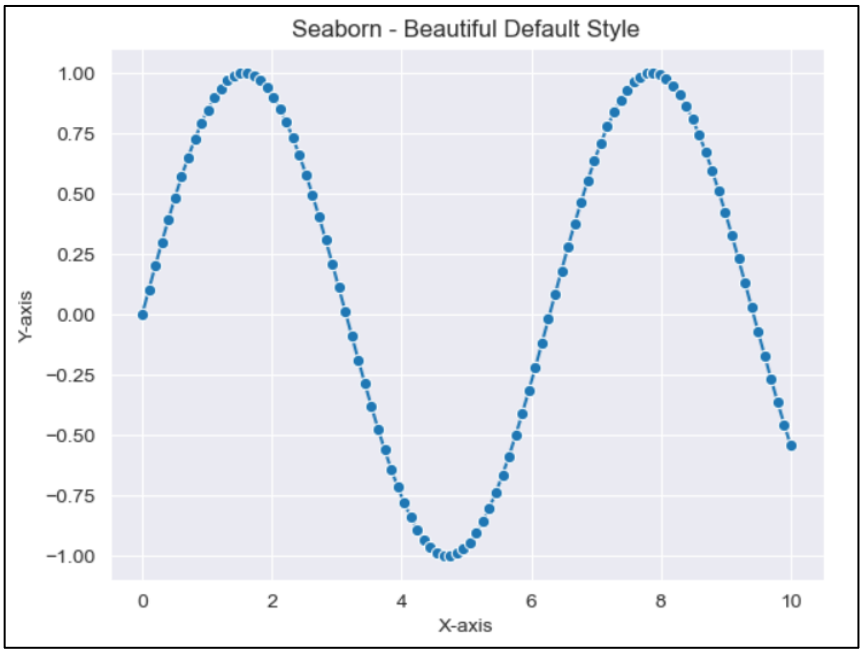


Figure 4 Plot by Seaborn

We can conclude, Seaborn give users who love stunning and visual appealing data visualisations while Matplotlib is for people who want total control and customization. The user's specific objective and preferences, including whether they want default styles that are beautifully pleasing or require a lot customisation, will determine whether Matplotlib or Seaborn is the better option. Both of them libraries are good resources that offer several methods of data visualisation to the plot , giving users choices based on their needs and objectives.

(D) Community Support

In terms Community Support , Both of them has an active and supportive community . Community Support is very important for Programmer in terms of Learning and Growth , Problem solving , Collaboration, and Keeping up with the Industry Trends.It is very crucial for Programmer to be in community support to look up which Python Data Visualization Library is the best one that can help them to use the Python Library .Matplotlib also have their documentation on their website that can help Programmer like us to read If if we have a problem with the library .

3. Discussion and recommendations

The decision between Seaborn and Matplotlib will depend on your own requirements and preferences.

Matplotlib is the suggested option if you need complete access to and control over the visual components of your charts. With its extensive customization features, you may precisely adjust the axes, labels, colours, and markers on your plot. You have the freedom to design almost any style of plot with Matplotlib, from straightforward line plots to intricate 3D visualisations. However, bear in mind that this flexibility necessitates writing more code and possibly requiring you to devote more time to designing and styling your plots.

However, Seaborn is the way to go if you want a simpler user interface and attractive plots without falling down in detailed customising . In order to make Matplotlib more approachable and user-friendly for data visualisation tasks, Seaborn offers a higher-level API built on top of it (Waskom, M. L. ,2021). It offers a wide range of statistical visualisation functions that are particularly tailored to deal with typical data analysis scenarios. Additionally, Seaborn has built-in themes and colour schemes that let you make aesthetically appealing plots with just a few lines of code. It makes handling categorical data, showing distributions, and illustrating relationships between variables more easier when charting.

4. Concluding Remark

In conclusion, the reliable Python data visualisation libraries Matplotlib and Seaborn both have their own advantages and disadvantages. While Seaborn is a high-level library with more practical modification options and better default settings, Matplotlib is a low-level library that gives complete control over the plot's specifics. While Seaborn is better suited for rapidly and efficiently constructing aesthetically appealing and educational plots with less code, Matplotlib is better for creating unique visualisations and offers a greater variety of plot types. The decision between Matplotlib and Seaborn ultimately comes down to your unique requirements and preferences for data visualisation

IV. Part (2)

1. Goals

For Part 2 , I will visualize a data set that I got from Kaggle which is Data Science Job Salaries 2023.As we know Data science Job was called “The Sexiest Job in the word” since 2012.The terms emphized the growing demand and importance of data science skills in nowdays Industries.So then , I will Visualize 3 features to have more corellation or importance towards the target output , Which is :

a. *Bar Chart* to Visualize the distribution of experience levels
b. *Line Plot* to show the trend of average salary over the year
c. *Pie Chart* to represent the distribution of company sizes using a pie chart to visualize the proportion of small , medium, and large companies in dataset

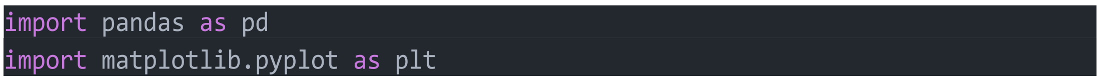
2. Justification Of Choice

For this Part, I decided to use Matplotlib as Python Data Visualization Because It offers a wide range of plot types and customization options, allowing you to tailor your visualizations to your specific needs.In addition, Matplotlib provides extensive documentation and has an active user community, ensuring that you can quickly find solutions to any issues you encounter .This community can help me to Visualize the data and can find a quick solution if I have a problem to build the Visualization .

3. Steps To build the Visuals

Import Needed Library

So, first of all , we need to imported the necessary libraries to work out on our Data Visualization, In this case we will imported Pandas and matplotlib as out library to Visualize the Data.

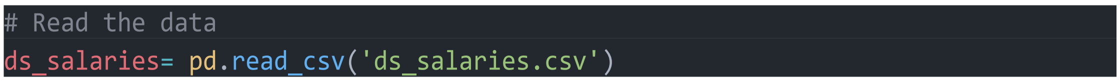


```python
import pandas as pd
import matplotlib.pyplot as plt
```

Figure 5 Code for Import pandas and matplotlib libraries

Import Dataset

Next before we start to visualize our data , we need to import the Dataset that we want to Visualize it .



```python
# Read the data
ds_salaries= pd.read_csv('ds_salaries.csv')
```

Figure 6 Code to read csv file from the same file directory with teh .iypnb file

In this case , I will command the program to read csv file that calles “ds_salaries.csv”.

Data Cleansing

Next , we will Clean the dataset that has been imported . we done this to make sure that the accuracy of the output will be increasing by removing some missing values in the data set .Below is the code :

```
# Removed the null values
ds_salaries = ds_salaries.dropna()
ds_salaries.apply(lambda x: sum(x.isnull()),axis=0)
```

Figure 7 Code to removed the null data in the dataset

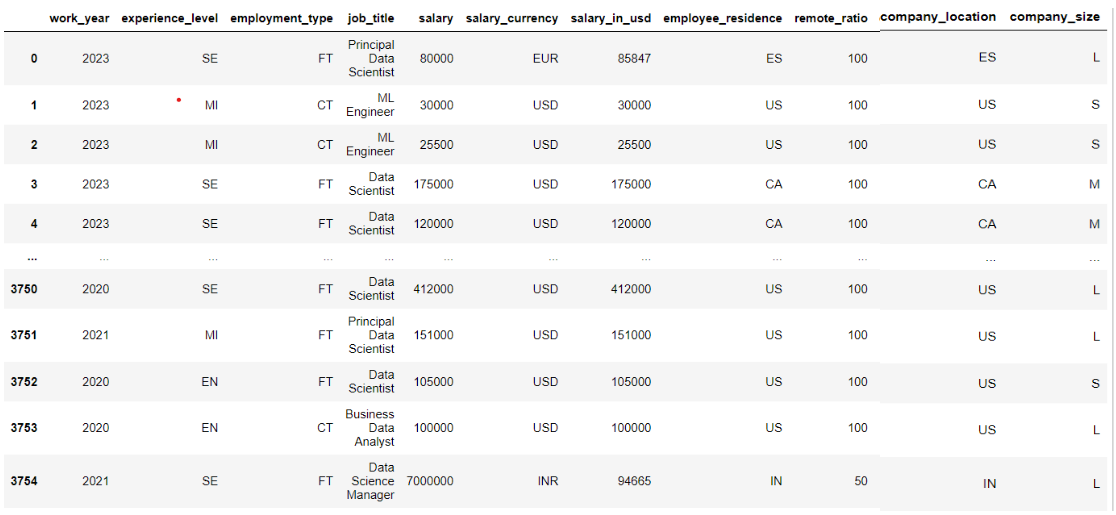
Exploring Data

Next , We will look at the data that already we imported, Figure 8 is the code how to Display the data

```
#Print the data
ds_salaries
```

Figure 8 Print the dataset

Output:



	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
0	2023	SE	FT	Principal Data Scientist	80000	EUR	85847	ES	100	ES	L
1	2023	MI	CT	ML Engineer	30000	USD	30000	US	100	US	S
2	2023	MI	CT	ML Engineer	25500	USD	25500	US	100	US	S
3	2023	SE	FT	Data Scientist	175000	USD	175000	CA	100	CA	M
4	2023	SE	FT	Data Scientist	120000	USD	120000	CA	100	CA	M
...	...	...	...	...	...	...	...	...	...	...	...
3750	2020	SE	FT	Data Scientist	412000	USD	412000	US	100	US	L
3751	2021	MI	FT	Principal Data Scientist	151000	USD	151000	US	100	US	L
3752	2020	EN	FT	Data Scientist	105000	USD	105000	US	100	US	S
3753	2020	EN	CT	Business Data Analyst	100000	USD	100000	US	100	US	L
3754	2021	SE	FT	Data Science Manager	7000000	INR	94665	IN	50	IN	L

We will also look up at the count of Rows and Coloumn by command this code :

```
#Print the data
ds_salaries.shape()
```

Output :

```
(3755, 11)
```

As we see , This Dataset has 11 Coloumn and 3755 of rows.

Features 1: Bar Chart To visualize distribution of Experience Level

Full Code:

```
# Calculate the count of each experience level
experience_counts = ds_salaries['experience_level'].value_counts()

# Generate a bar plot of the experience distribution
distribution_experience = experience_counts.plot(kind='bar', color='blue',
alpha=0.7)

# Set labels and title
plt.xlabel('Experience')
plt.ylabel('Count')
plt.title('Experience Distribution')

# Change the labels on the X-axis
new_labels = ['Senior-Level','Mid-Level', 'Entry-Level', 'Executive']  #
Modify the labels for the x-axis
plt.xticks(range(len(experience_counts)), new_labels)

# Add value labels to the bars
for p in distribution_experience.patches:
    distribution_experience.annotate(f"{p.get_height()}", (p.get_x() +
p.get_width() / 2., p.get_height()), ha='center',
                                      va='center', xytext=(0, 5),
textcoords='offset points')

# Customize the plot appearance
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--')

# Display the plot
plt.tight_layout()
plt.show()
```

Step by Step :

(1)Calculate the count

```
# Calculate the count of each experience level
experience_counts = ds_salaries['experience_level'].value_counts()
```

This line of code will count the unique value of experience_level and store it in experience_count varible .

(2)Generate a Barplot

```
# Generate a bar plot of the experience distribution
distribution_experience = experience_counts.plot(kind='bar', color='blue',
alpha=0.7)
```

This line of code will generate the barplot with color of blue with alpha transperancy with 0.7.

(3)Set label and title for barplot

```
# Set labels and title
plt.xlabel('Experience')
plt.ylabel('Count')
plt.title('Experience Distribution')
```

Next, This line of code will set and create labels for barplot which is for y-axis it willbenamed "Experience", while the x-axis will be named "Count". And lastly we will create a title for this barplot which is "Experience Distribution".

(3)Change the Label for Element in X-axis

```
# Change the labels on the X-axis
new_labels = ['Senior-Level','Mid-Level', 'Entry-Level', 'Executive']  #
Modify the labels for the x-axis
plt.xticks(range(len(experience_counts)), new_labels)
```

Next we will change the Element within X-Label with "Senior-Level","Mid-Level","Entry-level"and "Executive".

(4)Customize the plot

```
# Customize the plot appearance
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--')
```

Next we will customize the plot appearance by Rotate the element in X-axis by 45 degree and put some grid in the plot In Y-Axis with line Style "--".

(5)Display the Plot

```
# Display the plot
plt.tight_layout()
plt.show()
```

And Finally we will Display the plot .The function tight_layout() is to automactically adjust the spacing between subplots and make the plot more neatly and clean .

Output:

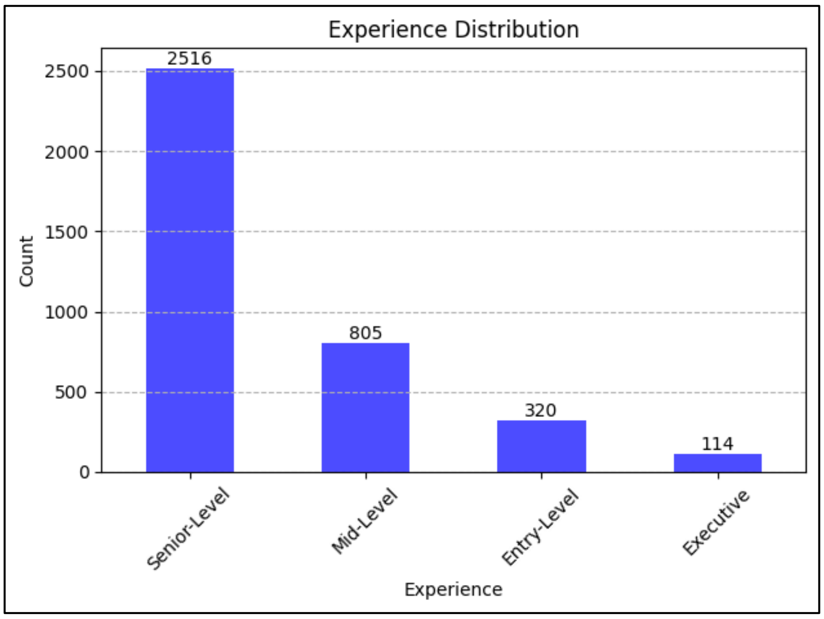


Figure 9 Output

From Figure 9,The output of the code reveals the distribution of experience levels in the dataset. Among the different positions, the Senior level has the highest count with 2516 individuals, followed by Mid-level with 805 individuals. The Entry-Level position has 320 individuals, while the Executive position has the lowest count of 114 individuals.

Features 2: Line plot about Trend of Salary with Experience Position

Full Code:

```python
# Calculate the average salary for each experience level
avg_salary = ds_salaries.groupby('experience_level')['salary_in_usd'].mean()

# Create a figure and axes
fig, avg_salary_plot = plt.subplots(figsize=(8, 8))

# Create a line plot of the average salary
avg_salary_plot.plot(avg_salary, marker='o', color='red')

# Set labels and title
avg_salary_plot.set_xlabel('Experience Level')
avg_salary_plot.set_ylabel('Average Salary')
avg_salary_plot.set_title('Average Salary by Experience Level')

# Rename the x-axis data
avg_salary_plot.set_xticklabels(['Entry Level','Executive', 'Mid-Level',
'Senior Level'])

#Create legend for x-axis
avg_salary_plot.legend(['Average Salary'], loc='upper left')

# Customize the plot appearance
avg_salary_plot.grid(axis='y', linestyle='--')

# Display the plot
plt.tight_layout()
plt.show()
```

Step By Step :

(1) Calculate the average of Salary or Mean of salary

```python
# Calculate the average salary for each experience level
avg_salary = ds_salaries.groupby('experience_level')['salary_in_usd'].mean()
```

This code will calculate the average of salary_in_usd for each experience_level in the dataset of ds_salaries.The average salary will be put in the avg_salary variable.

(2) Calculate the average of Salary or Mean of salary

```
# Create a figure and axes
fig, avg_salary_plot = plt.subplots(figsize=(8, 8))
```

Next , we will create a figure and axes using subplots().

(3)Create a plot line

```
# Create a line plot of the average salary
avg_salary_plot.plot(avg_salary, marker='o', color='red')
```

After that , we will create a line plot of the average salary with color of red and with marker represent by circle "o".

(4) Set label and title

```
# Set labels and title
avg_salary_plot.set_xlabel('Experience Level')
avg_salary_plot.set_ylabel('Average Salary')
avg_salary_plot.set_title('Average Salary by Experience Level')
```

Next , we will create a label and title for this line plot . In this case , we will create X-axis label that named "Experience level", "Average Salary " for Y-axis and "Average Salary by Experience Level " for the title .

(5) Renamed X-axis element

```
# Rename the x-axis data
avg_salary_plot.set_xticklabels(['Entry Level','Executive', 'Mid-Level',
'Senior Level'])
```

After that we will renamed our X-axis name for better visulisation and understanding about the plot line.

(6) Create a legend

```
#Create legend for x-axis
avg_salary_plot.legend(['Average Salary'], loc='upper left')
```

This code will created a legend for us with detail named " Average Salary ", in Upper left location of the plot .

(7) Customize the Y-axis

```
# Customize the plot appearance
avg_salary_plot.grid(axis='y', linestyle='--')
```

This code will add grid for Y axis with dotted line.

(8) Show the plotted data

```
# Display the plot
plt.tight_layout()
plt.show()
```

Using the show() function to Display the plotted data .

Output:

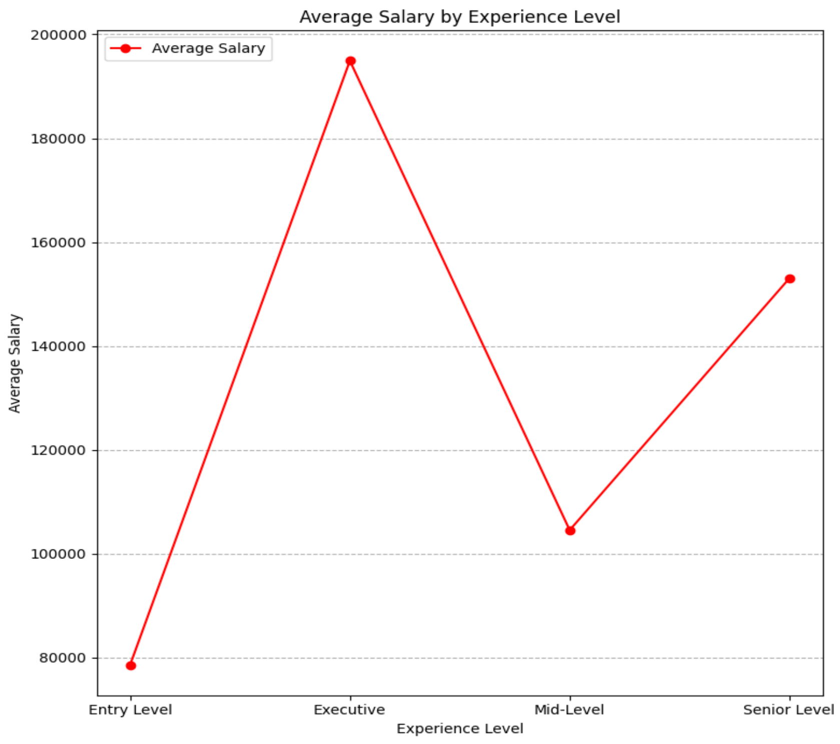


Figure 10

As we can see the line plot shown in Figure 10 is The average salary of each position which are Entry Level , Executive , Mid-Level and Senior Level .As we can see Executive cumulate the highest Salary in usd which is more than 18000 Us Dollar ,and The Lowest one is Entry Level which is below that 80000 US dollar .We can conclude that the higher position you get , the higher salary that you can get .

Features 3: Pie Chart For Count of Company Size

Full Code:

```
# Calculate the count of each company size
company_size_counts = ds_salaries['company_size'].value_counts()

# Create a figure and axes
fig, company_sizes = plt.subplots(figsize=(8, 8))

# Create a pie chart of the company size counts
colors = ['#FF7F50', '#87CEEB', '#9ACD32']
labels = ["Medium", "Large", "Small"]  # Put the legend names
sizes = company_size_counts.values
company_sizes.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%',
startangle=90)

# Set title and legend
company_sizes.set_title('Distribution of Company Sizes', fontsize=14)
company_sizes.legend(labels, loc='upper right')

# Equal aspect ratio ensures that pie is drawn as a circle
company_sizes.axis('equal')

# Display the plot
plt.tight_layout()
plt.show()
```

Step By Step :

(1)Calculate the count of each company size

```
# Calculate the count of each company size
company_size_counts = ds_salaries['company_size'].value_counts()
```

Firstly , created the count of each uniques company size in the ‘ds_salaries’ dataset . The result will be ut inside company_size_counts variable

(2)Create a figure and axes

```
# Create a figure and axes
fig, company_sizes = plt.subplots(figsize=(8, 8))
```

Creates a figure and axes for the plot using the subplots() function from the 'matplotlib.pyplot' module. It assigns the figure object to the variable 'fig' and the axes object to the variable 'company_sizes'. The 'figsize' parameter specifies the width and height of the figure in inches.

(3)Create a pie chart of the company size counts

```
# Create a pie chart of the company size counts
colors = ['#FF7F50', '#87CEEB', '#9ACD32']
labels = ["Medium", "Large", "Small"]  # Put the legend names
sizes = company_size_counts.values
company_sizes.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%',
startangle=90)
```

creates a pie chart using the 'pie()' method of the 'company_sizes' axes object. It takes the sizes array as input, which represents the count of each company size. The 'labels' parameter provides the labels for each slice of the pie. The 'colors' parameter indicate the colors of the pie slices. The 'autopct' parameter formats the percentage values displayed on each slice with one decimal place.

(4)Set a title and legend

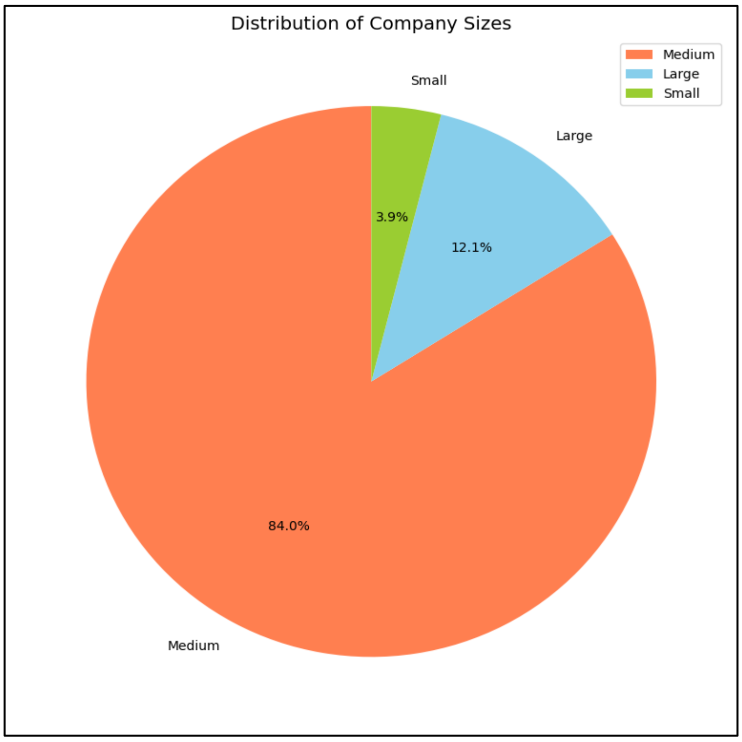
```
# Set title and legend
company_sizes.set_title('Distribution of Company Sizes', fontsize=14)
company_sizes.legend(labels, loc='upper right')
```

Set a title as "Distributin of Company Sizes " and Create a legend with labels variable in upper right location f the plots .

(5)Display the plots

```
# Display the plot
plt.tight_layout()
plt.show()
```

Output:



As we can see , the proportional of Medium , Large and Small Company .The majority of the Worker in this dataset is working with Medium Size company with 84.0 percent . Followed by 12.1 percent of Large Size company and The least one is Small company size with 3.9 percent.

4. Result and Discussion (Analysis and recommendations)

From the data Visualazation that we generate , we already can easily evealuate information for our data analysis .We startes with the first Features 1 in Figure 9 which is Bar chart that shown a distribution of experience Level in the dataset , we can conclude that The senior level has the highest count (2516 individuals), followed by Mid-Level (805 individuals), entry-Level (320 individuals), and Executive (114 individuals).

For the line plots , we can evealuate the aveargae salary forf each position . The executive position has the highest average salary (above $18,000 USD) while the Entry level position has the minimum average salary which I s below $80,000 USD.

For the Pie chart we can evaluate the proportion of workers in different company sized .The majority of worker in the dataset are employed by medium-sized company (84.0%) , followed by (12.1%) large-sized company and the smallest proportion(3.9) in small -sized company

Conclusion, The dataset shows that the higher positions tends to have higher salaries , and the majority of workers are employed by medium-sized companies .

V. Conclusion

This report offers a comprehensive comparison of two popular Python libraries for data visualization, Matplotlib and Seaborn. For part 1 , It highlights the strengths and weaknesses of each library, providing a clear contrast between them. The objective is to help readers gain insights into the characteristics of these libraries and make an informed decision regarding their suitability for specific visualization needs. By examining this report, you will have a better understanding of Matplotlib and Seaborn, enabling you to choose the library that best aligns with your requirements.

For part 2 , we can summarize on how to use the library and how we can visualize it . in this cases Im using Matplotlib to Visualize the data. And base on the Visualization we can make a conclusion to the Set of data. We also can learn the syntax of the library and the way they Visualize all the features.

VI. References

Growth Tribe. (2023, April 26). *8 reasons why data analytics is the career for you in 2023*. Growth Tribe. https://growthtribe.io/blog/data-analytics-career#:~:text=Data%20analysts%20are%20in%20demand,5%25%20for%20all%20other%20industries.

Sial, A. H., Rashdi, S. Y. S., & Khan, A. H. (2021). Comparative analysis of data visualization libraries Matplotlib and Seaborn in Python. *International Journal*, *10*(1).

Python® – the language of today and Tomorrow. About Python. (n.d.). https://pythoninstitute.org/about-python#:~:text=Python%20was%20created%20by%20Guido,called%20Monty%20Python's%20Flying%20Circus.

Analytics Vidhya. (2023, March 8*). Introduction to Matplotlib and Seaborn*. Medium. Retrieved from https://medium.com/analytics-vidhya/introduction-to-matplotlib-and-seaborn-e2dd04bfc821

Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021.

Seaborn Development Team. (n.d.*). Introduction to seaborn. In Seaborn: Statistical Data Visualization*. Retrieved September 5, 2021, from https://seaborn.pydata.org/tutorial/introduction.html