Practical 8

Tutorial

In this we will be working on the titanic dataset and plotting various different plots. The practical is already very simple and can be easily executed my reading the manual. We have given the final code at the last page. Check it if you have any errors.

First open anaconda and launch spyder.

We will first use the titanic dataset which is already defined in the library.

Let's see what the Titanic dataset looks like. Execute the following script:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


dataset = sns.load_dataset('titanic')dataset.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

The dataset contains 891 rows and 15 columns and contains information about the passengers who boarded the unfortunate Titanic ship. The original task is to predict whether or not the passenger survived depending upon different features such as their age, ticket, cabin they boarded, the class of the ticket, etc. We will use the Seaborn library to see if we can find any patterns in the data.

1. **Finding patterns of data.**

Patterns of data can be find out with the help of different types of plots

Types of plots are:

A. **Distribution Plots**

a. Dist-Plot

b. Joint Plot

c. d. Rug Plot

B. **Categorical Plots**

a. Bar Plot

b. Count Plot

c. Box Plot

d. Violin Plot

C. **Advanced Plots**

a. Strip Plot

b. Swarm Plot

D. **Matrix Plots**

a. Heat Map

b. Cluster Map

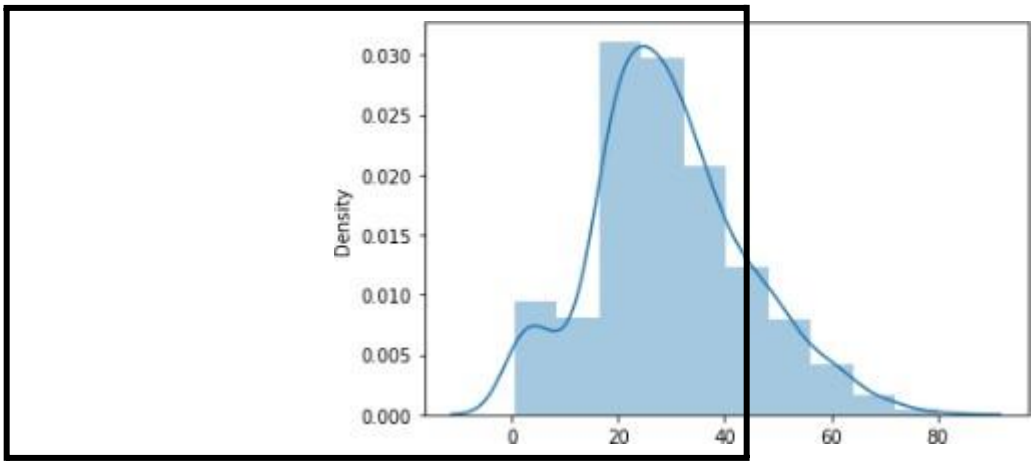
A. Distribution Plots:

These plots help us to visualise the distribution of data. We can use these plots to understand themean, median, range, variance, deviation, etc of the data.

a. Distplot

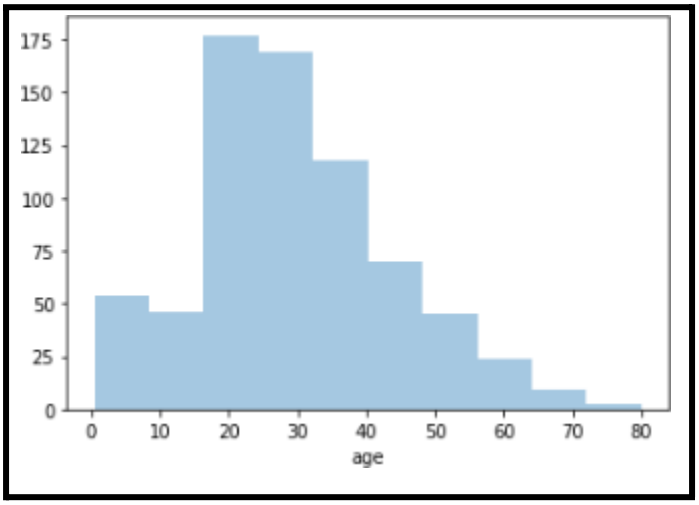
- Dist plot gives us the histogram of the selected continuous variable.
- It is an example of a univariate analysis.
- We can change the number of bins i.e. number of vertical bars in a histogram

```
import seaborn as sns

sns.distplot(x = dataset['age'], bins = 10)
```



The line that you see represents the kernel density estimation. You can remove this lineby passing False as the parameter for the kde attribute as shown below

```
sns.distplot(dataset['age'], bins = 10,kde=False)
```



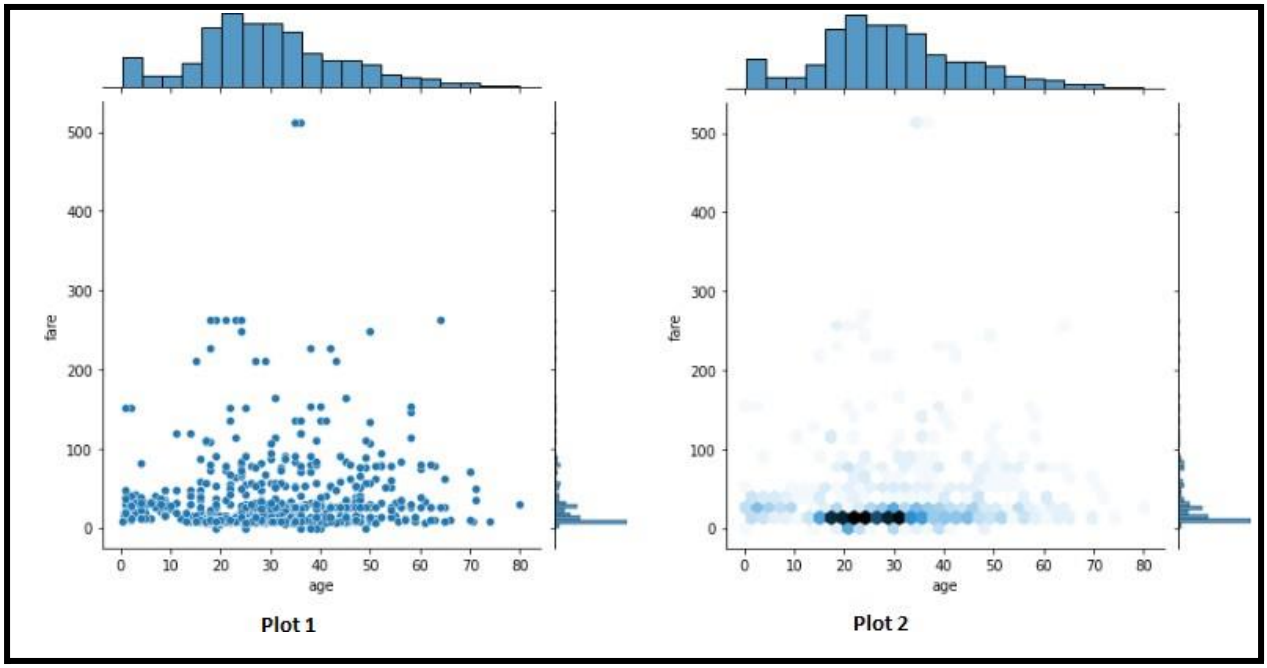
Here the x-axis is the age and the y-axis displays frequency. For example, for bins = 10,there are around 50 people having age 0 to 10

i.b. Joint Plot

- It is the combination of the distplot of two variables.
- It is an example of bivariate analysis.

- We additionally obtain a scatter plot between the variables to reflect their linear relationship. We can customise the scatter plot into a hexagonal plot, where, themore the colour intensity, the more will be the number of observations.

```
import seaborn as sns# For Plot 1

sns.jointplot(x = dataset['age'], y = dataset['fare'], kind ='scatter')

# For Plot 2

sns.jointplot(x = dataset['age'], y = dataset['fare'], kind = 'hex')
```



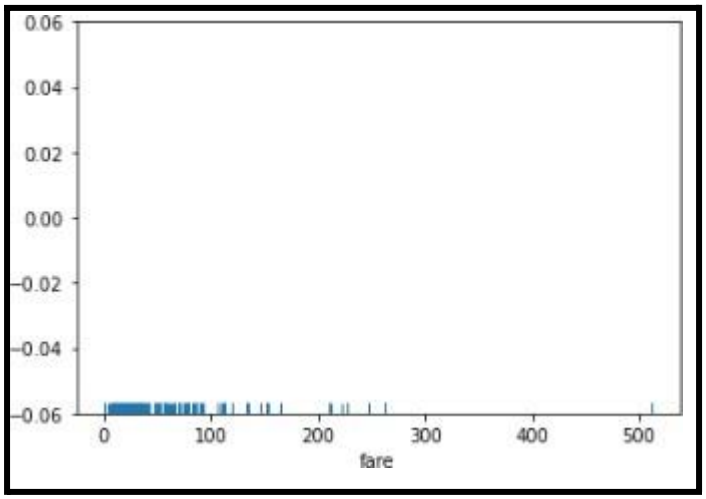
- From the output, you can see that a joint plot has three parts. A distribution plot at the topfor the column on the x-axis, a distribution plot on the right for the column on the y-axis and a scatter plot in between that shows the mutual distribution of data for both the columns. You can see that there is no correlation observed between prices and the fares.
- You can change the type of the joint plot by passing a value for the kind parameter. For instance, if instead of a scatter plot, you want to display the distribution of data in the form of a hexagonal plot, you can pass the value hex for the kind parameter.

- In the hexagonal plot, the hexagon with the most number of points gets darker colour. So if you look at the above plot, you can see that most of the passengers are between theages of 20 and 30 and most of them paid between 10-50 for the tickets.

a. **c. The Rug Plot**

b. The rugplot() is used to draw small bars along the x-axis for each point in the dataset. To plot a rug plot, you need to pass the name of the column. Let's plot a rug plot for fare.

```
sns.rugplot(dataset['fare'])
```



From the output, you can see that most of the instances for the fares have values between 0 and 100.

These are some of the most commonly used distribution plots offered by the Python's Seaborn Library. Let's see some of the categorical plots in the Seaborn library.

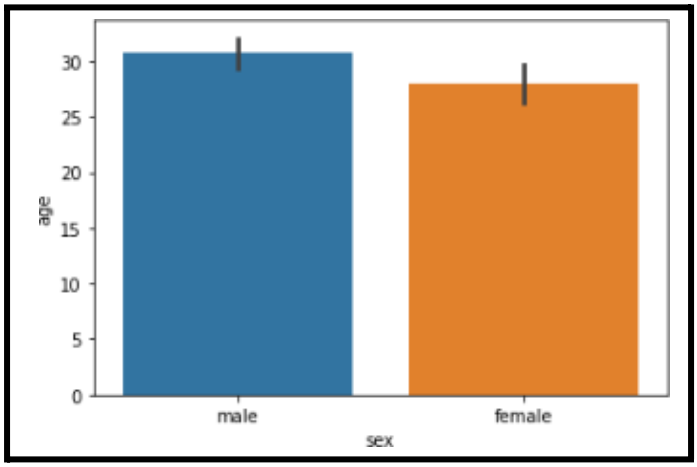
2. Categorical Plots

Categorical plots, as the name suggests, are normally used to plot categorical data. The categorical plots plot the values in the categorical column against another categorical column ora numeric column. Let's see some of the most commonly used categorical data.

b. The Bar Plot

The barplot() is used to display the mean value for each value in a categorical column, against a numeric column. The first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. For instance, if you want to know the mean value of the age of the male and female passengers, you can use the bar plot as follows.

```
sns.barplot(x='sex', y='age', data=dataset)
```

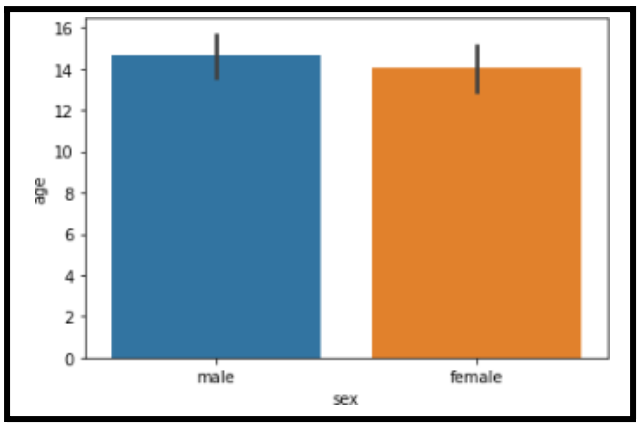


From the output, you can clearly see that the average age of male passengers is just less than 40 while the average age of female passengers is around 33.

In addition to finding the average, the bar plot can also be used to calculate other aggregate values for each category. To do so, you need to pass the aggregate function to the estimator. For instance, you can calculate the standard deviation for the age of each gender as follows:

```
import numpy as np

import matplotlib.pyplot as pltimport seaborn

as sns

sns.barplot(x='sex', y='age', data=dataset, estimator=np.std)
```

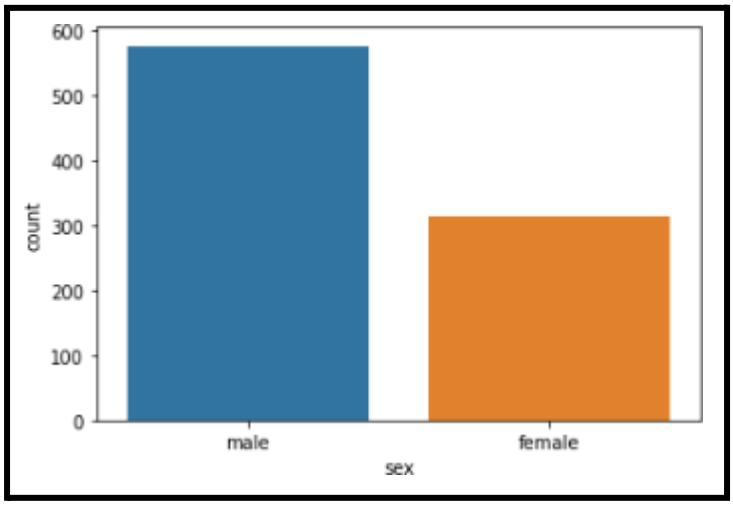
Notice, in the above script we use the std aggregate function from the numpy library to calculate the standard deviation for the ages of male and female passengers. The output looks like this:



c. The Count Plot

The count plot is similar to the bar plot, however it displays the count of the categories in a specific column. For instance, if we want to count the number of males and women passenger wecan do so using count plot as follows:

```
sns.countplot(x='sex', data=dataset)
```

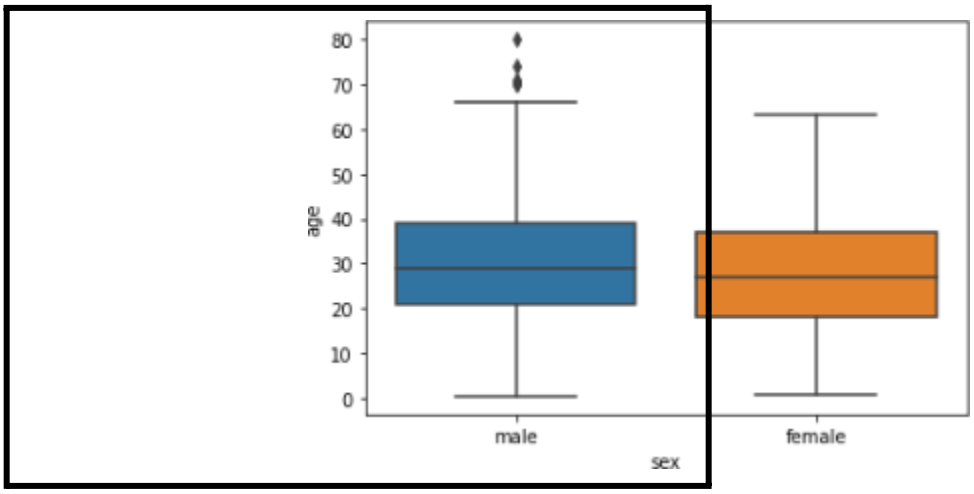


d. The Box Plot

The box plot is used to display the distribution of the categorical data in the form of quartiles. The centre of the box shows the median value. The value from the lower whisker to the bottom of the box shows the first quartile. From the bottom of the box to the middle of the box lies the second quartile. From the middle of the box to the top of the box lies the third quartile and finally from the top of the box to the top whisker lies the last quartile.

Now let's plot a box plot that displays the distribution for the age with respect to each gender. You need to pass the categorical column as the first parameter (which is sex in our case) and the numeric column (age in our case) as the second parameter. Finally, the dataset is passed as the third parameter, take a look at the following script:

```
sns.boxplot(x='sex', y='age', data=dataset)
```

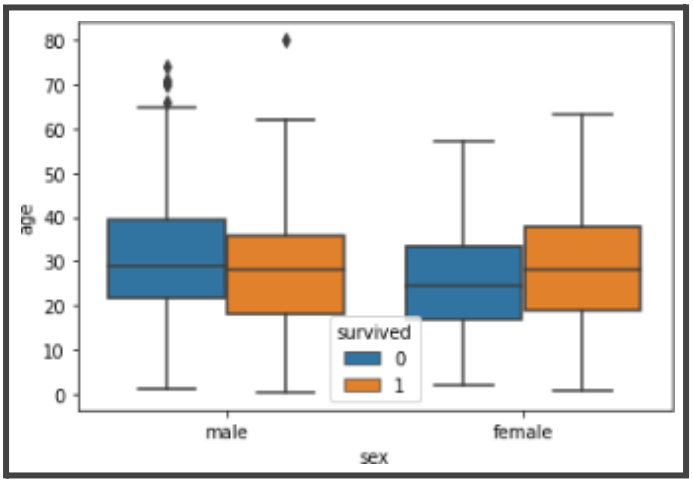


Let's try to understand the box plot for females. The first quartile starts at around 1 and ends at 20 which means that 25% of the passengers are aged between 1 and 20. The second quartile starts at around 20 and ends at around 28 which means that 25% of the passengers are aged between20 and 28. Similarly, the third quartile starts and ends between 28 and 38, hence 25% passengers are aged within this range and finally the fourth or last quartile starts at 38 and ends around 64.

If there are any outliers or the passengers that do not belong to any of the quartiles, they are called outliers and are represented by dots on the box plot.

You can make your box plots more fancy by adding another layer of distribution. For instance, if you want to see the box plots of forage of passengers of both genders, along with the information about whether or not they survived, you can pass the survived as value to the hue parameter as shown below:

```
sns.boxplot(x='sex', y='age', data=dataset, hue="survived")
```



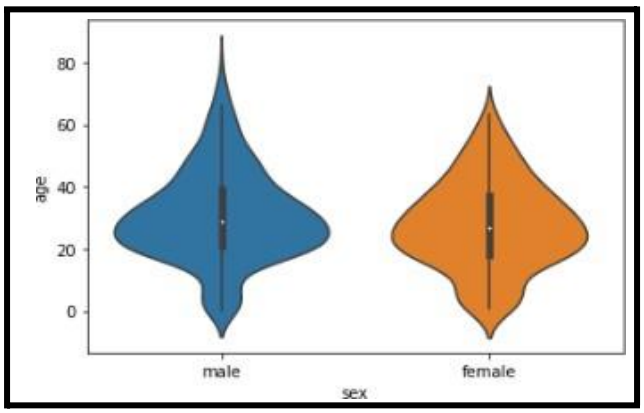
Now in addition to the information about the age of each gender, you can also see the distribution of the passengers who survived. For instance, you can see that among the male passengers, on average more younger people survived as compared to the older ones. Similarly, you can see that the variation among the age of female passengers who did not survive is much greater than the age of the surviving female passengers.

e. The Violin Plot

The violin plot is similar to the box plot, however, the violin plot allows us to display all the components that actually correspond to the data point. The violinplot() function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset.

Let's plot a violin plot that displays the distribution for the age with respect to each gender.

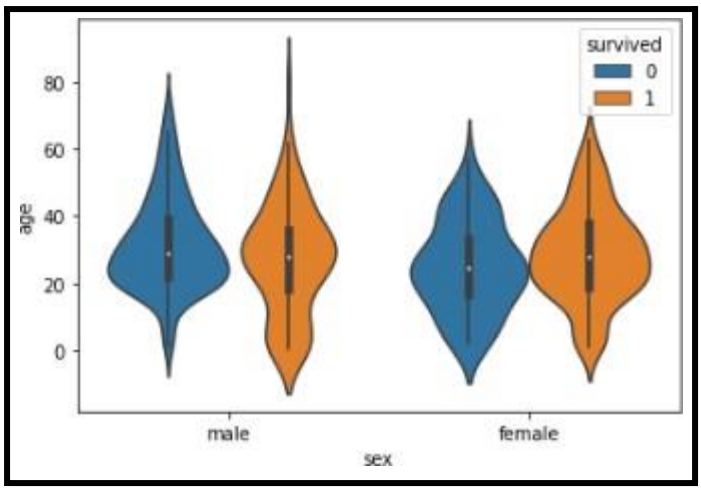
```
sns.violinplot(x='sex', y='age', data=dataset)
```



You can see from the figure above that violin plots provide much more information about the data as compared to the box plot. Instead of plotting the quartile, the violin plot allows us to see all the components that actually correspond to the data. The area where the violin plot is thicker has a higher number of instances for the age. For instance, from the violin plot for males, it is clearly evident that the number of passengers with age between 20 and 40 is higher than all the rest of the age brackets.

Like box plots, you can also add another categorical variable to the violin plot using the hue parameter as shown below:

```
sns.violinplot(x='sex', y='age', data=dataset, hue='survived')
```



Now you can see a lot of information on the violin plot. For instance, if you look at the bottom of the violin plot for the males who survived (left-orange), you can see that it is thicker than the bottom of the violin plot for the males who didn't survive (left-blue). This means that the number of young male passengers who survived is greater than the number of young male passengers who did not survive

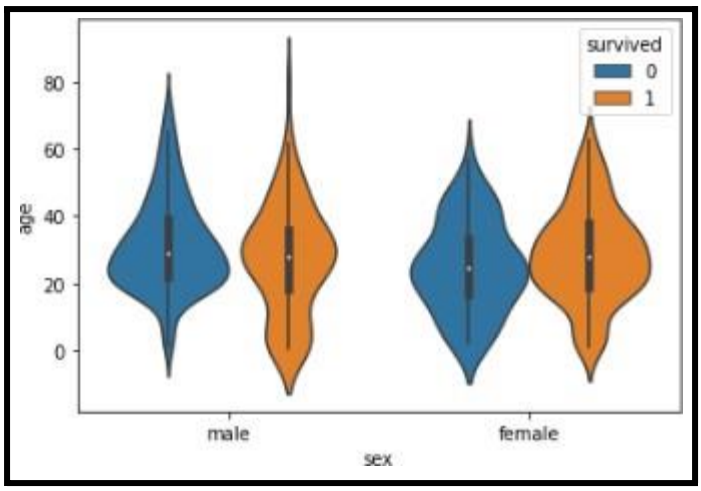
Advanced Plots:

a. The Strip Plot

The strip plot draws a scatter plot where one of the variables is categorical. We have seen scatter plots in the joint plot and the pair plot sections where we had two numeric variables. The strip plot is different in a way that one of the variables is categorical in this case, and for each category in the categorical variable, you will see a scatter plot with respect to the numeric column.

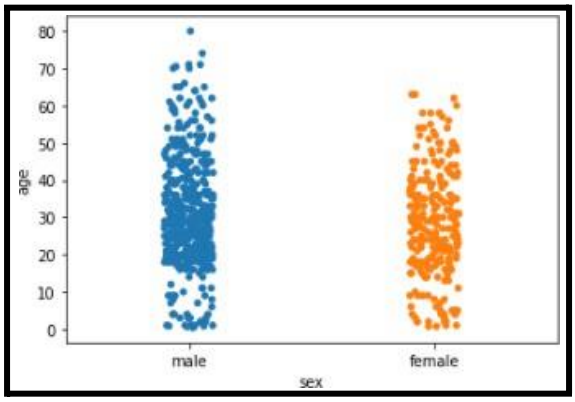
The stripplot() function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. Look at the following script:

```
sns.stripplot(x='sex', y='age', data=dataset, jitter=False)
```



You can see the scattered plots of age for both males and females. The data points look like strips. It is difficult to comprehend the distribution of data in this form. To better comprehend thedata, pass True for the jitter parameter which adds some random noise to the data. Look at the following script:

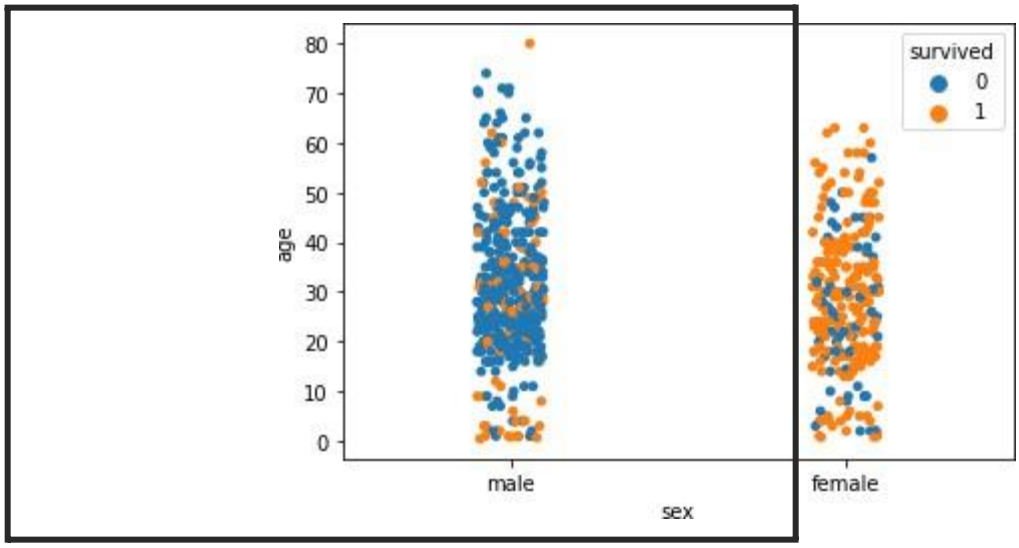
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True)
```



Now you have a better view for the distribution of age across the genders.

Like violin and box plots, you can add an additional categorical column to strip plot using hue parameter as shown below:

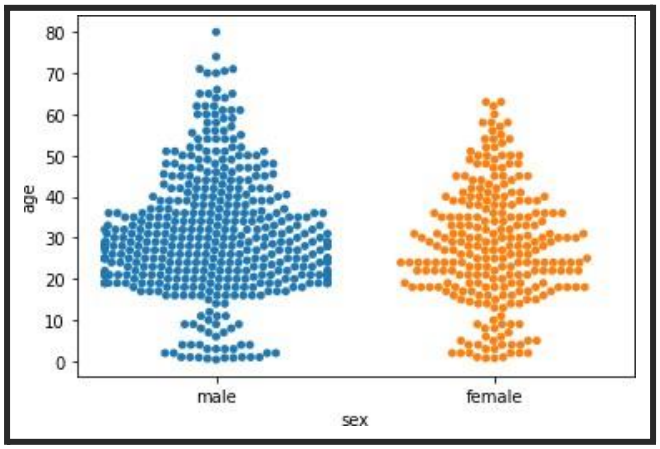
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True, hue='survived')
```



b. The Swarm Plot

The swarm plot is a combination of the strip and the violin plots. In the swarm plots, the points are adjusted in such a way that they don't overlap. Let's plot a swarm plot for the distribution of age against gender. The swarmplot() function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. Look at the following script:

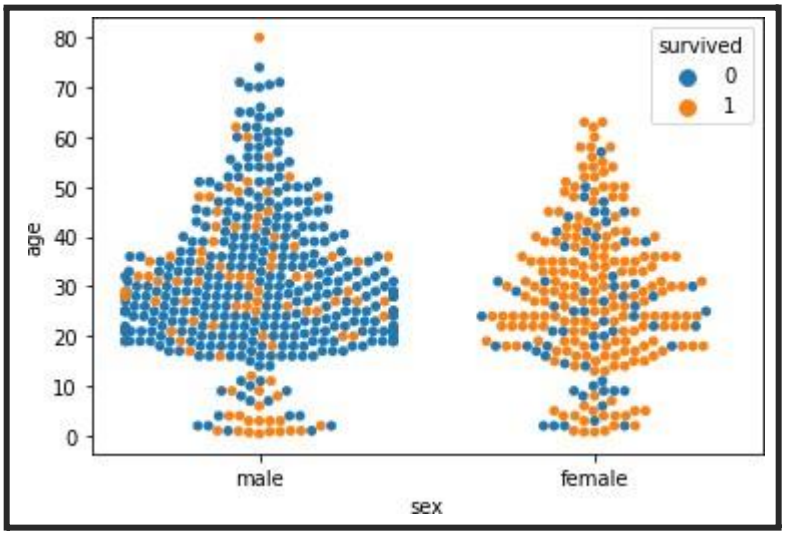
```
sns.swarmplot(x='sex', y='age', data=dataset)
```



You can clearly see that the above plot contains scattered data points like the strip plot and the data points are not overlapping. Rather they are arranged to give a view similar to that of a violin plot.

Let's add another categorical column to the swarm plot using the hue parameter.

```
sns.swarmplot(x='sex', y='age', data=dataset, hue='survived')
```



From the output, it is evident that the ratio of surviving males is less than the ratio of surviving females. Since for the male plot, there are more blue points and less orange points. On the other hand, for females, there are more orange points (surviving) than the blue points (not surviving). Another observation is that amongst males of age less than 10, more passengers survived as compared to those who didn't.

1. Matrix Plots

Matrix plots are the type of plots that show data in the form of rows and columns. Heat maps are the prime examples of matrix plots.

a. Heat Maps

Heat maps are normally used to plot correlation between numeric columns in the form of a matrix. It is important to mention here that to draw matrix plots, you need to have meaningful information on rows as well as columns. Let's plot the first five rows of the Titanic dataset to see if both the rows and column headers have meaningful information. Execute the following script:

```
import pandas as pdimport

numpy as np

import matplotlib.pyplot as pltimport seaborn

as sns

dataset = sns.load_dataset('titanic')dataset.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

From the output, you can see that the column headers contain useful information such as passengers surviving, their age, fare etc. However the row headers only contain indexes 0, 1, 2, etc. To plot matrix plots, we need useful information on both columns and row headers. One way to do this is to call the corr() method on the dataset. The corr() function returns the correlation between all the numeric columns of the dataset. Execute the following script:

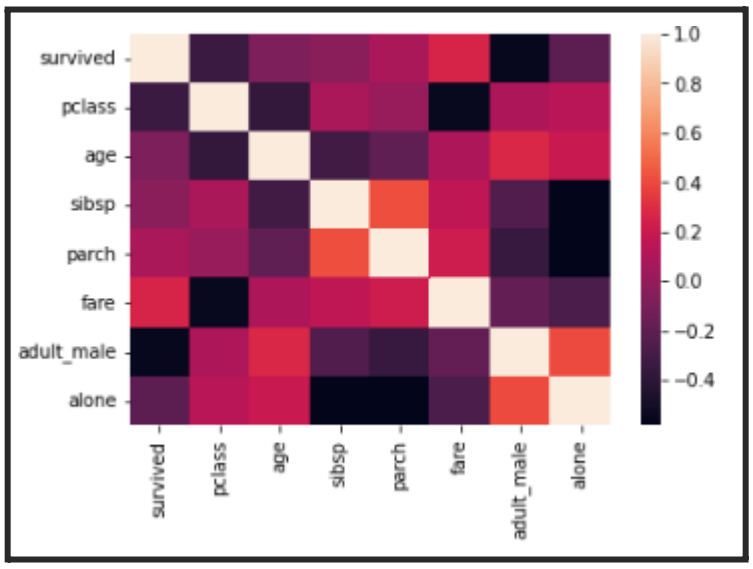
```
dataset.corr()
```

In the output, you will see that both the columns and the rows have meaningful header information, as shown below:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	-0.557080	-0.203367
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	0.094035	0.135207
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.280328	0.198270
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.253586	-0.584471
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.349943	-0.583398
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	-0.182024	-0.271832
adult_male	-0.557080	0.094035	0.280328	-0.253586	-0.349943	-0.182024	1.000000	0.404744
alone	-0.203367	0.135207	0.198270	-0.584471	-0.583398	-0.271832	0.404744	1.000000

Now to create a heat map with these correlation values, you need to call the heatmap() function and pass it your correlation dataframe. Look at the following script:

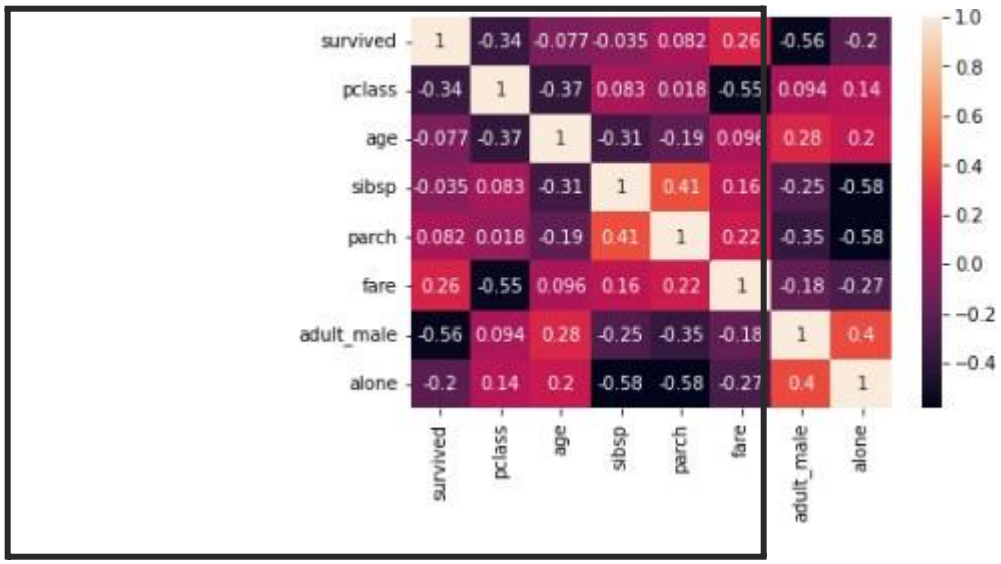
```
corr = dataset.corr()

sns.heatmap(corr)
```



From the output, it can be seen that what heatmap essentially does is that it plots a box for every combination of rows and column value. The colour of the box depends upon the gradient. For instance, in the above image if there is a high correlation between two features, the corresponding cell or the box is white, on the other hand if there is no correlation, the corresponding cell remains black.

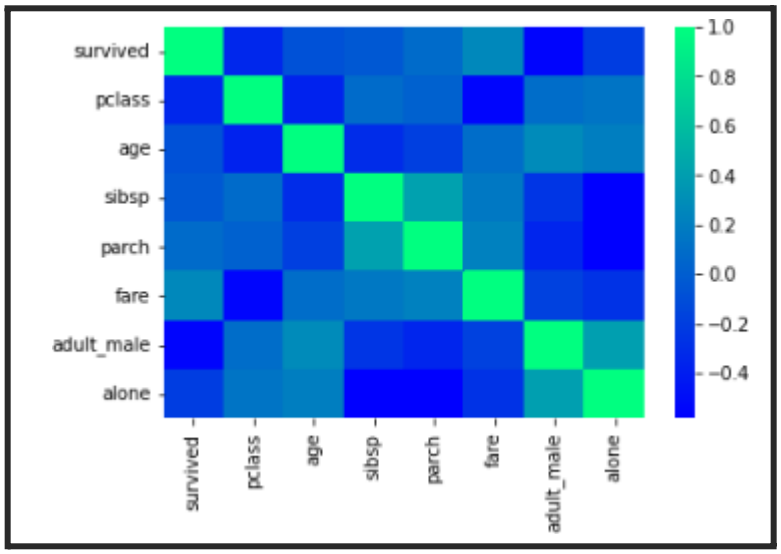
The correlation values can also be plotted on the heatmap by passing True for the annot parameter. Execute the following script to see this in action:

```
corr = dataset.corr() sns.heatmap(corr,

annot=True)
```



You can also change the colour of the heatmap by passing an argument for the cmap parameter.For now, just look at the following script:

```
corr = dataset.corr()

sns.heatmap(corr)
```



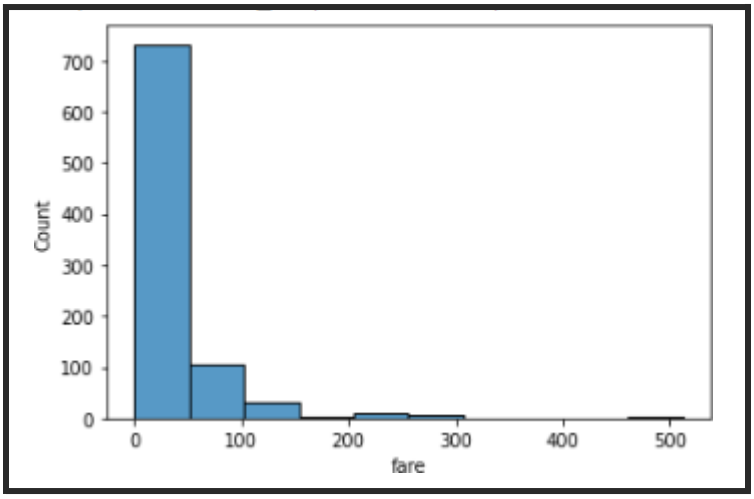
b. Cluster Map:

In addition to the heat map, another commonly used matrix plot is the cluster map. The cluster map basically uses Hierarchical Clustering to cluster the rows and columns of the matrix.

Let's plot a cluster map for the number of passengers who travelled in a specific month ofa specific year. Execute the following script:

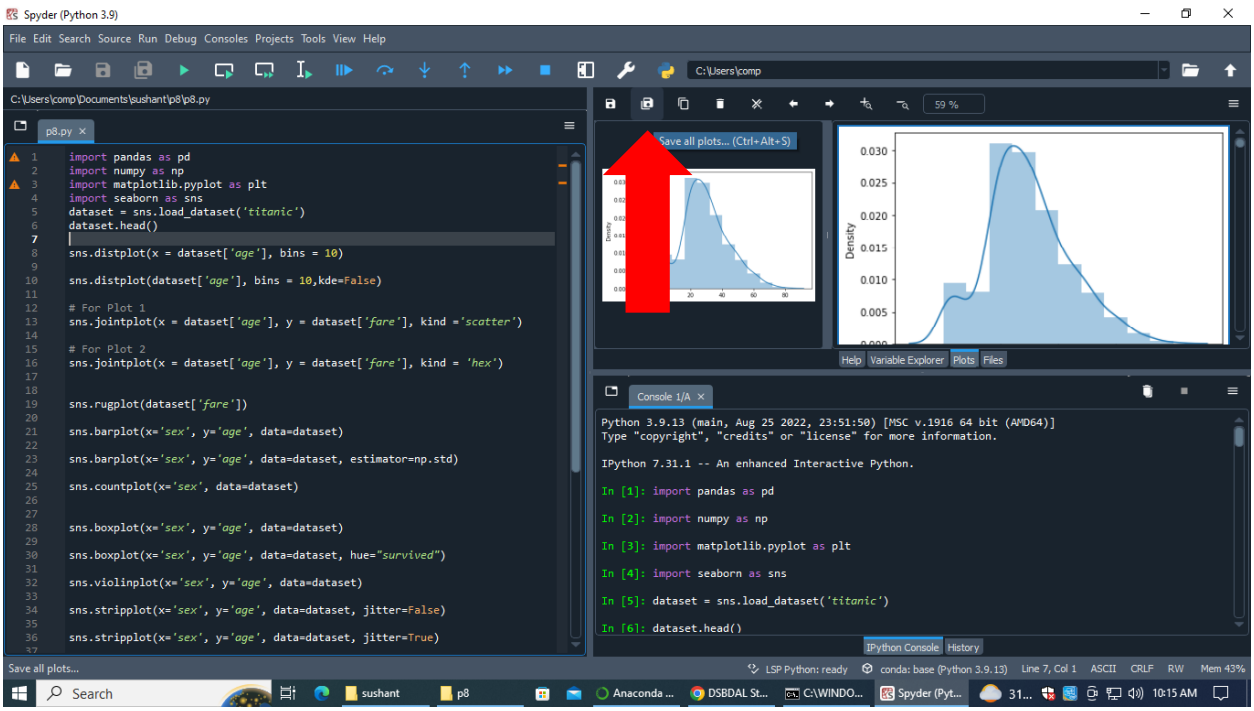
4. Checking how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

```
import seaborn as sns

dataset = sns.load_dataset('titanic') sns.histplot(dataset['fare'], kde=False,

bins=10)
```



From the histogram, it is seen that for around 730 passengers the price of the ticket is 50.For 100 passengers the price of the ticket is 100 and so on.

Lastly save the file and the histogram by clicking on the save all plots button above the console window.



Conclusion-

Seaborn is an advanced data visualisation library built on top of Matplotlib library. In this assignment, we looked at how we can draw distributional and categorical plots using the Seabornlibrary. We have seen how to plot matrix plots in Seaborn. We also saw how to change plot styles and use grid functions to manipulate subplots.