KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

(Deemed to be University)

SCHOOL OF COMPUTER APPLICATIONS

SPRING SEMESTER 2025-26

Date: 14.01.2026

1. **Course Code: BCA3004**
2. **Course title: Introduction to Data Science (IDS) Lab** **Marks = 4**

Assignment 2 (Data Visualization)

Title of the Assignment: Data Visualization

1. Use the inbuilt dataset 'titanic'. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.
2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.
3. Plot a box plot for distribution of age with respect to each gender along with they survived or not. (Column names : 'sex' and 'age') the information about whether
4. Write observations on the inference from the above statistics.

Objective of the Assignment: Students should be able to perform the data visualization using Python on any open source dataset

Contents for Theory:

Theory:

Seaborn which is another extremely useful library for data visualization in Python. The Seaborn library is built on top of Matplotlib and offers many advanced data visualization capabilities.

Though, the Seaborn library can be used to draw a variety of charts such as matrix plots, grid plots, regression plots etc., Seaborn library can be used to draw distributional and categorial plots. To draw regression plots, matrix plots, and grid plots, Seaborn library need to download.

Downloading the Seaborn Library

Seaborn library can be installed by using pip installer by executing following command:

pip install seaborn

Alternatively, Anaconda distribution of Python can be used, following command is executed to download the seaborn library:

conda install seaborn

The Dataset

Titanic dataset is used, which is downloaded by default with the Seaborn library. The load_dataset function is used to load the dataset and pass the name of the dataset.

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()
```

The script above loads the Titanic dataset and displays the first five rows of the dataset using the head function. The output looks like this:

survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
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. Seaborn library is used to find any patterns in the data.

Distributional Plots

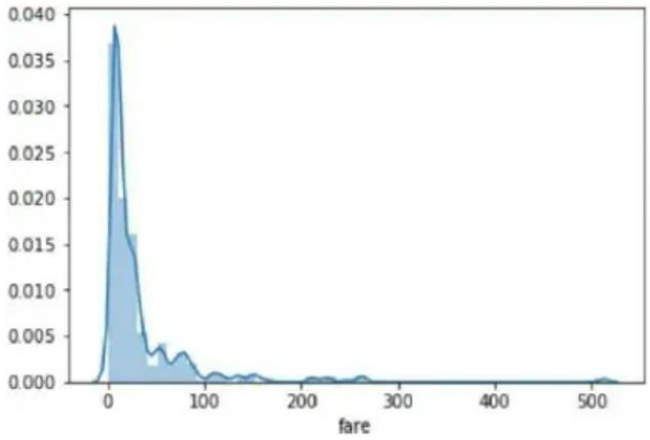
Distributional plots, as the name suggests are type of plots that show the statistical distribution of data.

The Dist Plot

The distplot() shows the histogram distribution of data for a single column. The column name is passed as a parameter to the distplot() function. To check how the price of the ticket for each passenger is distributed, execute the following script:

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

Output:

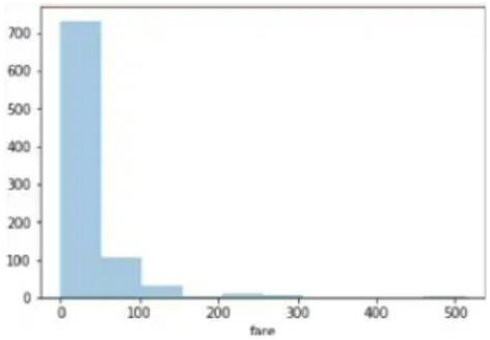


There is no line for the kernel density estimation on the plot. To pass the value for the bins parameter in order to find more or less details in the graph, execute the following script:

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

By setting the number of bins to 10, data distributed in 10 bins as shown in the following output:

Output:



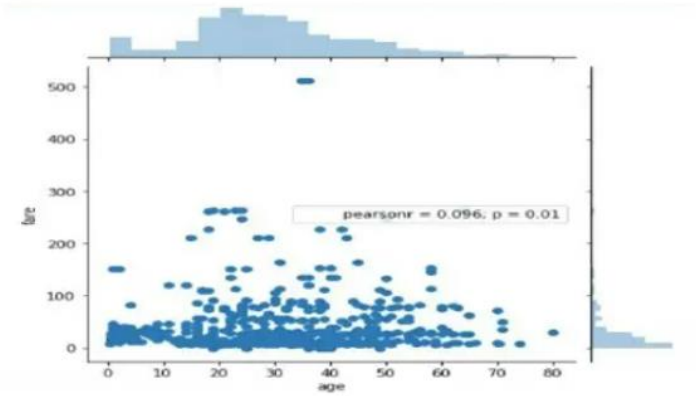
In the output, there are more than 700 passengers, the ticket price is between 0 and 50.

The Joint Plot

The jointplot()is used to display the mutual distribution of each column. There is need to pass three parameters to jointplot. The first parameter is the column name which display the distribution of data on x-axis. The second parameter is the column name which display the distribution of data on y-axis. Finally, the third parameter is the name of the data frame. Plot a joint plot of age and fare columns to see if there is any relationship between the two.

```
sns.jointplot(x='age', y='fare', data=dataset)
```

Output:

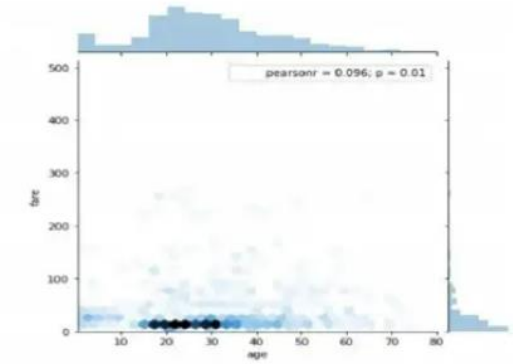


From the output, a joint plot has three parts. A distribution plot at the top for the column on the x-axis, a distribution plot on the right for the column on the y-axis and a scatter plot in between that shows the mutual distribution of data for both the columns. There is no correlation observed between prices and the fares.

To change the type of the joint plot by passing a value for the kind parameter. The distribution of data can be displayed in the form of a hexagonal plot, by passing the value hex for the kind parameter.

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

Output:



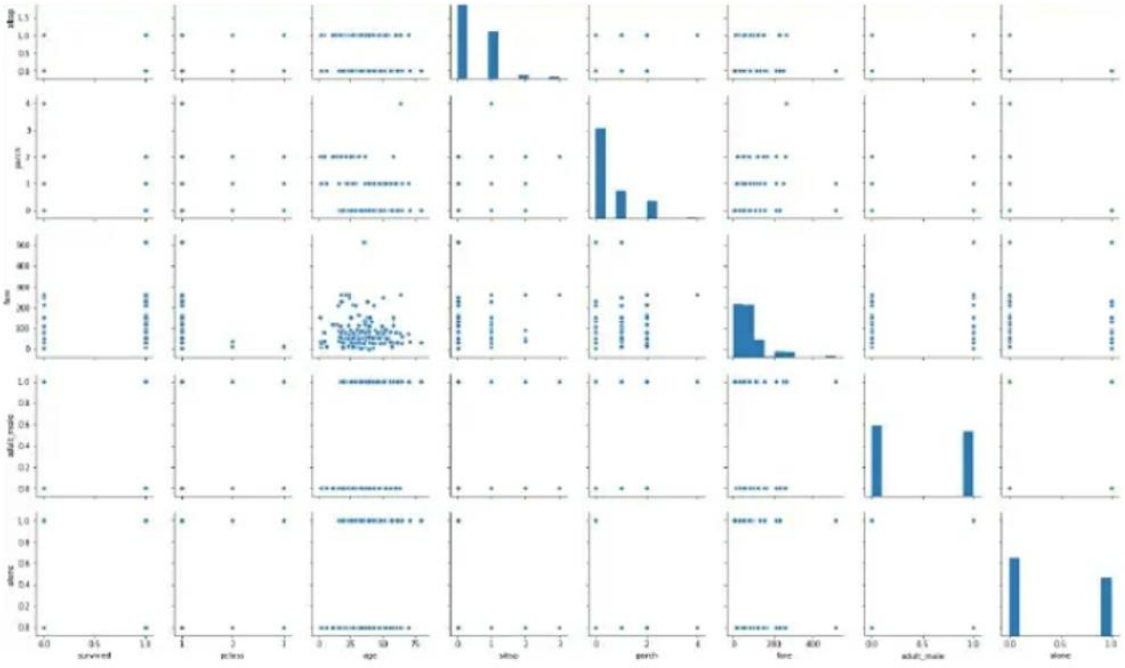
In the hexagonal plot, the hexagon with most number of points gets darker color. From the hexagonal plot, most of the passengers are between age 20 and 30 and most of them paid between 10-50 for the tickets.

The Pair Plot

The pairplot() is a type of distribution plot that basically plots a joint plot for all the possible combination of numeric and Boolean columns in dataset. The name of your dataset need to pass as the parameter to the pairplot() function as shown below:

```
sns.pairplot(dataset)
```

A snapshot of the portion of the output is shown below:



Note: Before executing the script above, remove all null values from the dataset using the following command:

```
dataset = dataset.dropna()
```

From the output of the pair plot , It is clear that joint plots for all the numeric and Boolean columns in the Titanic dataset.

To add information from the categorical column to the pair plot, The name of the categorical column have to pass to the hue parameter. For instance to plot the gender information on the pair plot, execute the following script:

```
sns.pairplot(dataset, hue='sex')
```

Output:

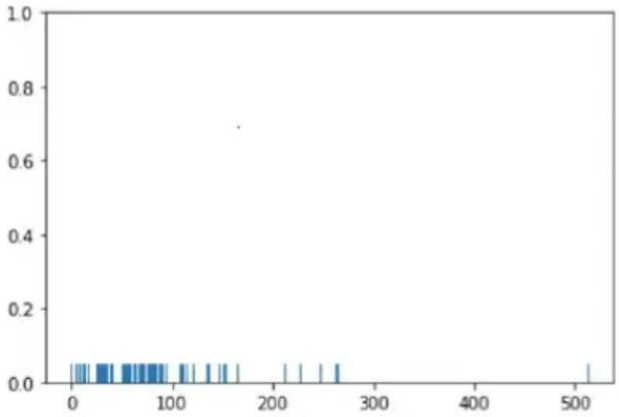
In the output, the information about the males in orange and the information about the female in blue (as shown in the legend). From the joint plot on the top left, it is clear that among the surviving passengers, the majority were female.

The Rug Plot

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

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

Output:



From the output, it is clear that as was the case with the distplot(), 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. Some of categorical plots in the Seaborn library as follows

Categorical Plots

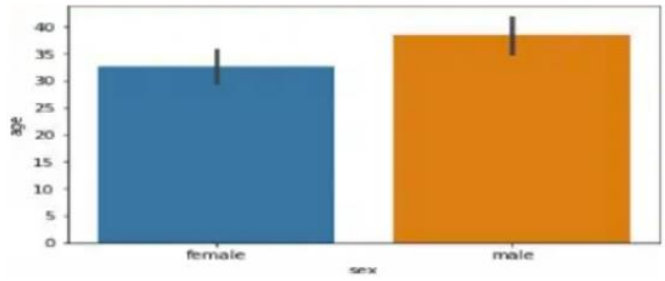
Categorical plots, as the name suggests are normally used to plot categorical data. The categorical plots plot the values in the categorical column against another categorical column or a numeric column. Most commonly used categorical data as follows:

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. To find the mean value of the age of the male and female passengers, use the bar plot as follows.

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

Output:



From the output, the average age of male passengers is just less than 40 while the average age of female passengers is around 33.

To find the average, the bar plot can also be used to calculate other aggregate values for each category. Pass the aggregate function to the estimator. To calculate the standard deviation for the age of each gender as follows:

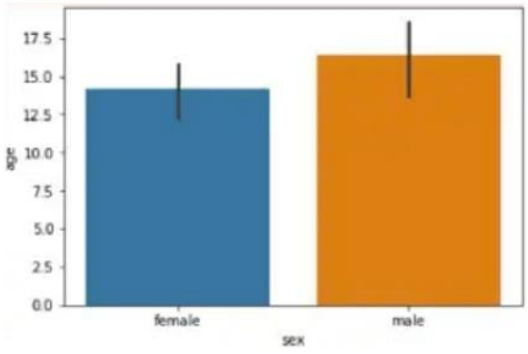
```
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

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

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



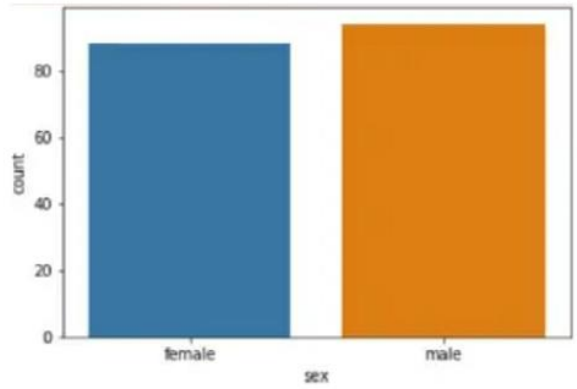
The Count Plot

The count plot is similar to the bar plot, It displays the count of the categories in a specific column. To count the number of males and women passenger,use count plot as follows:

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

The output shows the count as follows:

Output:



The Box Plot

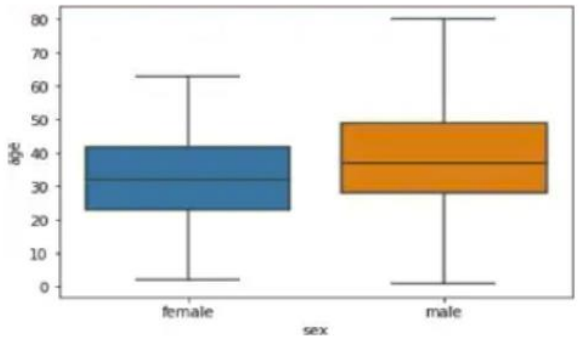
The box plot is used to display the distribution of the categorical data in the form of quartiles. The center of the box shows the median value. The value from the lower whisker to the bottom of the box shows the first quartile. From the bottom of the box to the middle of the box lies the second quartile.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

To plot a box plot that displays the distribution for the age with respect to each gender,to pass the categorical column as the first parameter (which is sex) and the numeric column (age) as the second parameter. The dataset is passed as the third parameter.

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

Output:



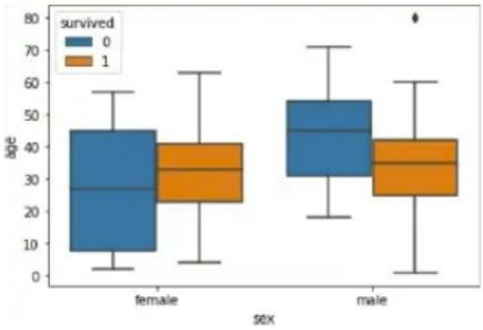
From the above plot,the first quartile starts at around 5 and ends at 22 which means that 25% of the passengers are aged between 5 and 25. The second quartile starts at around 23 and ends at around 32 which means that 25% of the passengers are aged between 23 and 32. Similarly, the third quartile starts and ends between 34 and 42, hence 25% passengers are aged within this range and finally the fourth or last quartile starts at 43 and ends around 65.

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.

To see the box plots of forage of passengers of both genders, along with the information about whether or not they survived, pass the survived as value to the hue parameter as shown below:

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

Output:



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

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

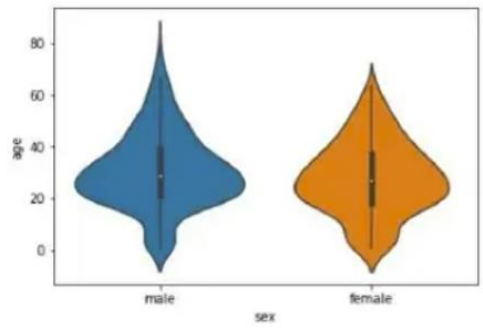
The Violin Plot

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)
```

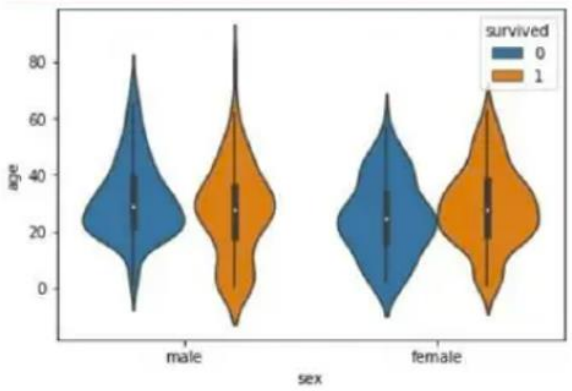
Output:



To 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')
```

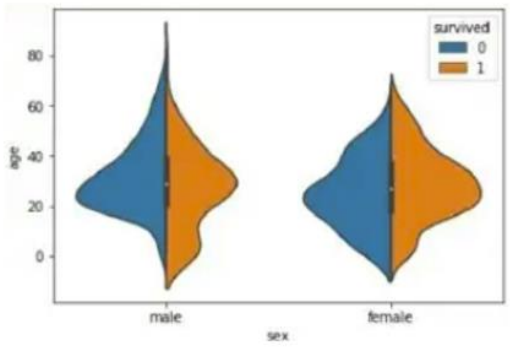


Now to find a lot of information on the violin plot. For instance, if you look at the bottom of the violin plot for the males who survived (left-orange), you can see that it is thicker than the bottom of the violin plot for the males who didn't survive (left-blue). This means that the number of young male passengers who survived is greater than the number of young male passengers who did not survive. The violin plots convey a lot of information, however, on the downside; it takes a bit of time and effort to understand the violin plots.

Instead of plotting two different graphs for the passengers who survived and those who did not, you can have one violin plot divided into two halves, where one half represents surviving while the other half represents the non-surviving passengers. To do so, you need to pass True as value for the split parameter of the violinplot() function. Let's see how we can do this:

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

The output looks like this:



Now it can clearly observed the comparison between the age of the passengers who survived and who did not for both males and females.

Both violin and box plots can be extremely useful. However, as a rule of thumb if you are presenting your data to a non-technical audience, box plots should be preferred since they are easy to comprehend. On the other hand, if you are presenting your results to the research community it is more convenient to use violin plot to save space and to convey more information in less time.

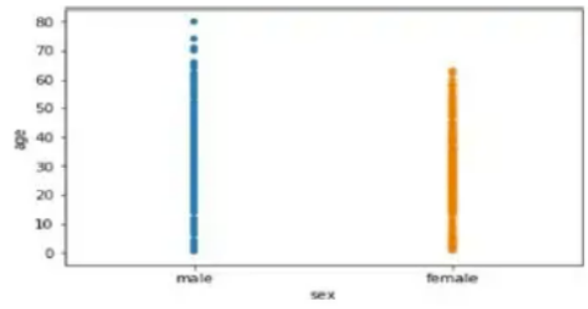
The Strip Plot

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 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)
```

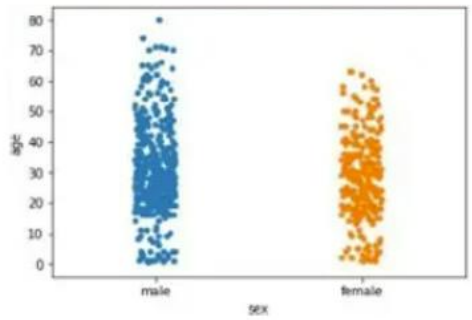
Output:



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 the data, 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)
```

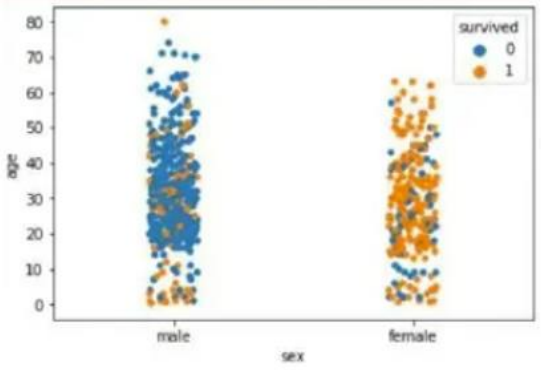
Output:



Now better view for the distribution of age across the genders can be observed.

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')
```

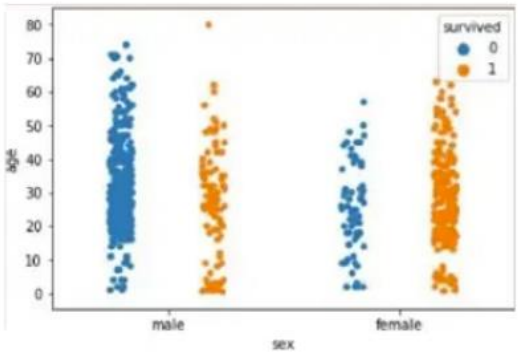


Again you can see there are more points for the males who survived near the bottom of the plot compared to those who did not survive.

Like violin plots, we can also split the strip plots. Execute the following script:

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

Output:

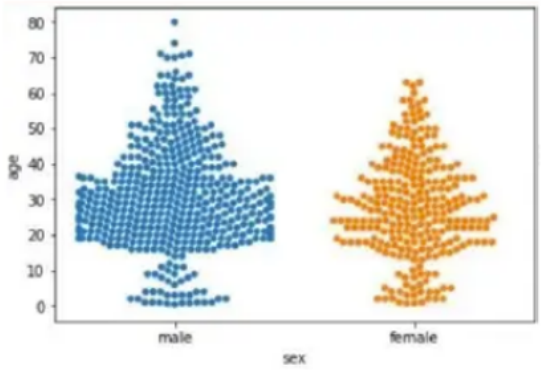


Now you can clearly see the difference in the distribution for the age of both male and female passengers who survived and those who did not survive.

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)
```

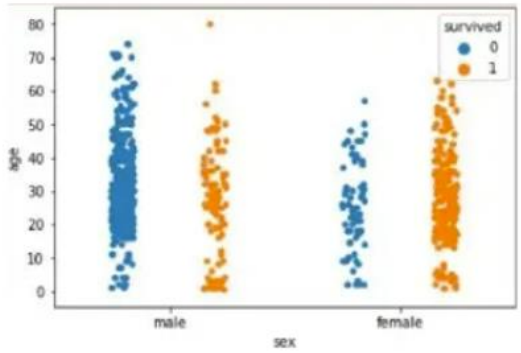


You can clearly see that the above plot contains scattered data points like the strip plot and the data points are not overlapping. Rather they are arranged to give a view similar to that of a violin plot.

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

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

Output:

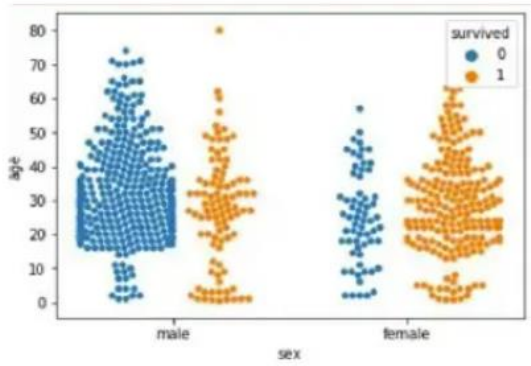


From the output, it is evident that the ratio of surviving males is less than the ratio of surviving females. Since for the male plot, there are more blue points and less orange points. On the other hand, for females, there are more orange points (surviving) than the blue points (not surviving). Another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

We can also split swarm plots as we did in the case of strip and box plots. Execute the following script to do so:

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

Output:



Now you can clearly see that more women survived, as compared to men.

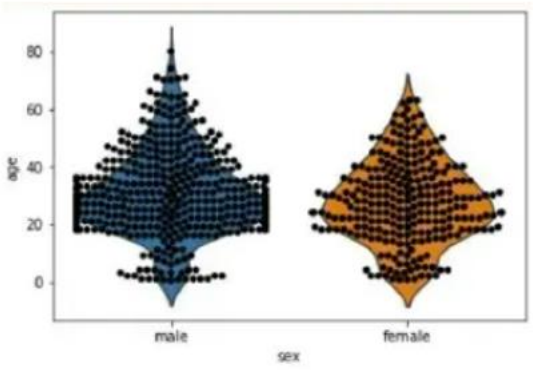
Combining Swarm and Violin Plots

Swarm plots are not recommended if you have a huge dataset since they do not scale well because they have to plot each data point. If you really like swarm plots, a better way is to combine two plots. For instance, to combine a violin plot with swarm plot, you need to execute the following script:

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

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

Output:



Exploratory Data Analysis

There are various techniques to understand the data, And the basic need is the knowledge of Numpy for mathematical operations and Pandas for data manipulation. Titanic dataset is used. For demonstrating some of the techniques, use an inbuilt dataset of seaborn as tips data which explains the tips each waiter gets from different customers.

Import libraries and loading Data

```
import numpy as np
import pandas pd
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import load_dataset
#titanic dataset
data = pd.read_csv("titanic_train.csv")
#tips dataset
tips = load_dataset("tips")
```

Univariate Analysis

Univariate analysis is the simplest form of analysis where we explore a single variable. Univariate analysis is performed to describe the data in a better way. we perform Univariate analysis of Numerical and categorical variables differently because plotting uses different plots.

Categorical Data:

A variable that has text-based information is referred to as categorical variables. Now following are various plots which we can use for visualizing Categorical data.

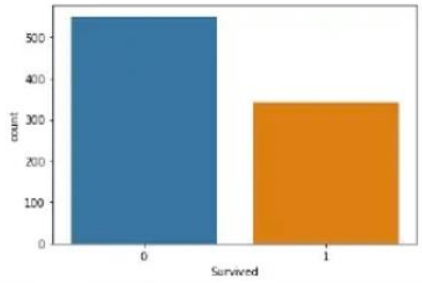
1) CountPlot:

Countplot is basically a count of frequency plot in form of a bar graph. It plots the count of each category in a separate bar. When we use the pandas' value counts function on any column. It is the same visual form of the value counts function. In our data-target variable is survived and it is categorical so plot a countplot of this.

```
sns.countplot(data['Survived'])

plt.show()
```

OUTPUT:



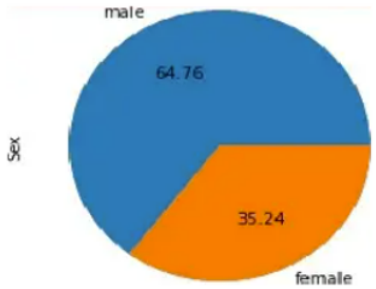
2) Pie Chart:

The pie chart is also the same as the countplot, only gives us additional information about the percentage presence of each category in data means which category is getting how much weightage in data. Now we check about the Sex column, what is a percentage of Male and Female members traveling.

```
data['Sex'].value_counts().plot(kind="pie", autopct="%.2f")

plt.show()
```

OUTPUT:



Numerical Data:

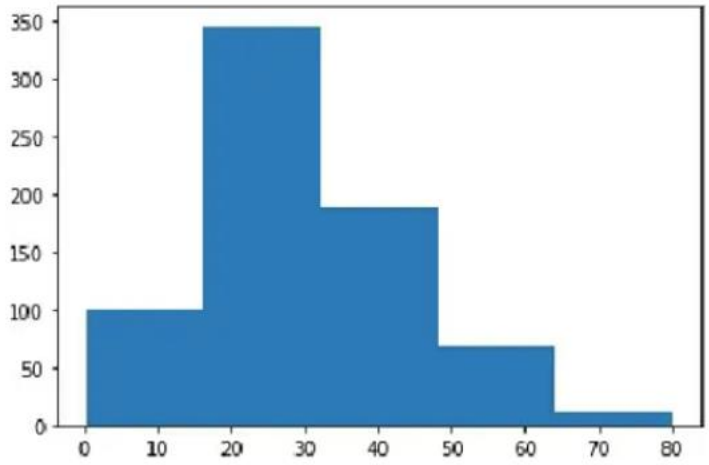
Analyzing Numerical data is important because understanding the distribution of variables helps to further process the data. Most of the time, we will find much inconsistency with numerical data so we have to explore numerical variables.

1) Histogram:

A histogram is a value distribution plot of numerical columns. It basically creates bins in various ranges in values and plots it where we can visualize how values are distributed. We can have a look where more values lie like in positive, negative, or at the center(mean). Let's have a look at the Age column.

```
plt.hist(data['Age'], bins=5)
plt.show()
```

OUTPUT:



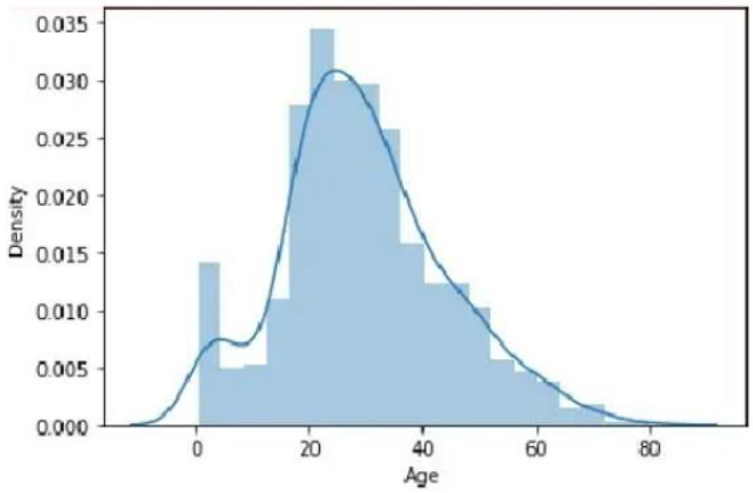
2) Distplot:

Distplot is also known as the second Histogram because it is a slight improvement version of the Histogram. Distplot gives us a KDE(Kernel Density Estimation) over histogram which explains PDF(Probability Density Function) which means what is the probability of each value occurring in this column.

```
sns.distplot(data['Age'])

plt.show()
```

OUTPUT:



3) Boxplot:

Boxplot is a very interesting plot that basically plots a 5 number summary. to get 5 number summary some terms we need to describe.

- Median – Middle value in series after sorting
- Percentile – Gives any number which is number of values present before this percentile like for example 50 under 25th percentile so it explains total of 50 values are there below 25th percentile
- Minimum and Maximum – These are not minimum and maximum values, rather they describe the lower and upper boundary of standard deviation which is calculated using Interquartile range(IQR).

IQR = Q3 - Q1

Lower_boundary = Q1 - 1.5 * IQR

Upper_bounday = Q3 + 1.5 * IQR

Here Q1 and Q3 is 1st quantile (25th percentile) and 3rd Quantile(75th percentile).

Bivariate/ Multivariate Analysis:

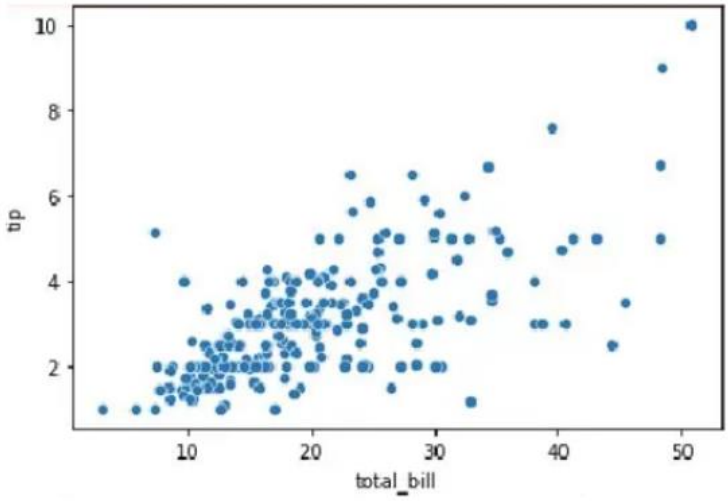
We have study about various plots to explore single categorical and numerical data. Bivariate Analysis is used when we have to explore the relationship between 2 different variables and we have to do this because, in the end, our main task is to explore the relationship between variables to build a powerful model. And when we analyze more than 2 variables together then it is known as Multivariate Analysis. we will work on different plots for Bivariate as well on Multivariate Analysis.

Explore the plots when both the variable is numerical.

1) Scatter Plot:

To plot the relationship between two numerical variables scatter plot is a simple plot to do. Let us see the relationship between the total bill and tip provided using a scatter plot.

sns.scatterplot(tips["total_bill"], tips["tip"])



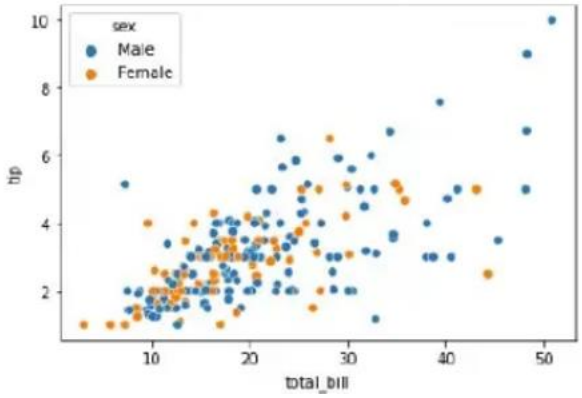
Multivariate analysis with scatter plot:

We can also plot 3 variable or 4 variable relationships with scatter plot. suppose we want to find the separate ratio of male and female with total bill and tip provided.

sns.scatterplot(tips["total_bill"], tips["tip"], hue=tips["sex"])

plt.show()

OUTPUT:

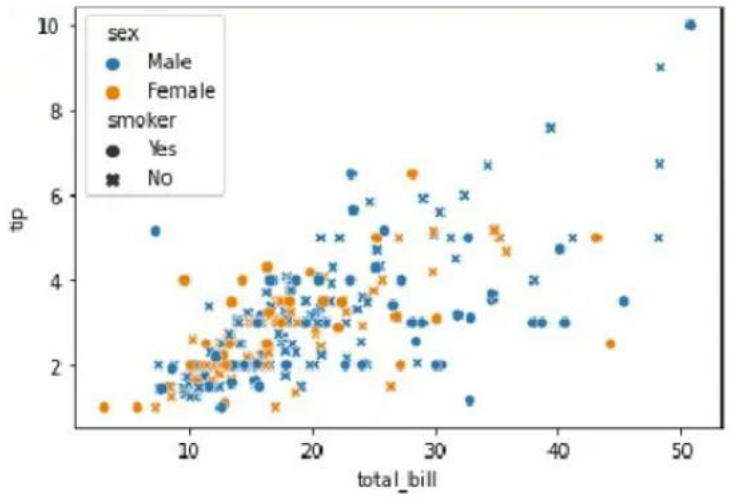


We can also see 4 variable multivariate analyses with scatter plots using style argument. Suppose along with gender we also want to know whether the customer was a smoker or not so we can do this.

```
sns.scatterplot(tips["total_bill"], tips["tip"], hue=tips["sex"], style=tips['smoker'])

plt.show()
```

OUTPUT:



Numerical and Categorical:

If one variable is numerical and one is categorical then there are various plots that we can use for Bivariate and Multivariate analysis.

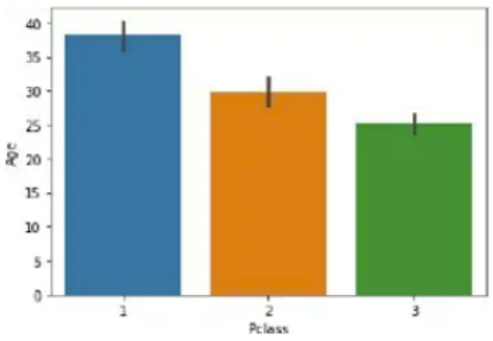
1) Bar Plot:

Bar plot is a simple plot which we can use to plot categorical variable on the x-axis and numerical variable on y-axis and explore the relationship between both variables. The blacktip on top of each bar shows the confidence Interval. let us explore P-Class with age.

```
sns.barplot(data['Pclass'], data['Age'])

plt.show()
```

OUTPUT:



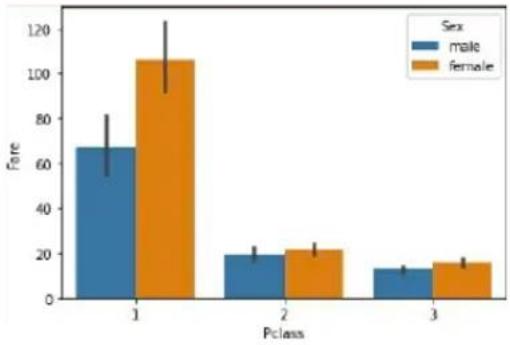
Multivariate analysis using Bar plot:

Hue's argument is very useful which helps to analyze more than 2 variables. Now along with the above relationship we want to see with gender.

```
sns.barplot(data['Pclass'], data['Fare'], hue = data["Sex"])

plt.show()
```

OUTPUT:

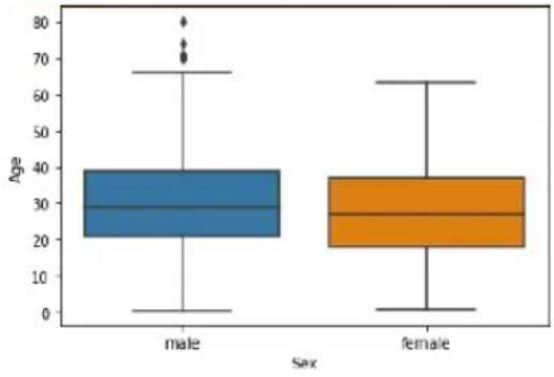


2) Boxplot:

We have already study about boxplots in the Univariate analysis above. we can draw a separate boxplot for both the variable. let us explore gender with age using a boxplot.

```
sns.boxplot(data['Sex'], data["Age"])
```

OUTPUT:



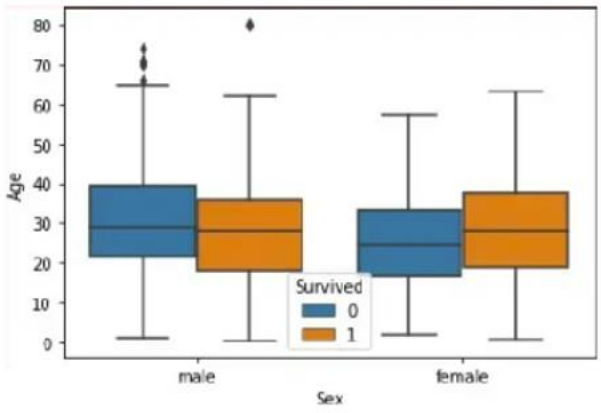
Multivariate analysis with boxplot:

Along with age and gender let's see who has survived and who has not.

```
sns.boxplot(data['Sex'], data["Age"], data["Survived"])
```

plt.show()

OUTPUT:

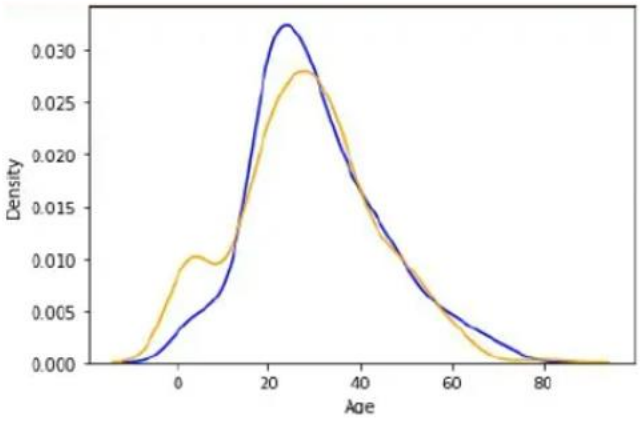


3) Distplot:

Distplot explains the PDF function using kernel density estimation. Distplot does not have a hue parameter but we can create it. Suppose we want to see the probability of people with an age range that of survival probability and find out whose survival probability is high to the age range of death ratio.

```
sns.distplot(data[data['Survived'] == 0]['Age'], hist=False, color="blue")
sns.distplot(data[data['Survived'] == 1]['Age'], hist=False, color="orange")
plt.show()
```

OUTPUT:



In above graph, the blue one shows the probability of dying and the orange plot shows the survival probability. If we observe it we can see that children's survival probability is higher

than death and which is the opposite in the case of aged peoples. This small analysis tells sometimes some big things about data and it helps while preparing data stories.

Categorical and Categorical:

Now, we will work on categorical and categorical columns.

1) Heatmap:

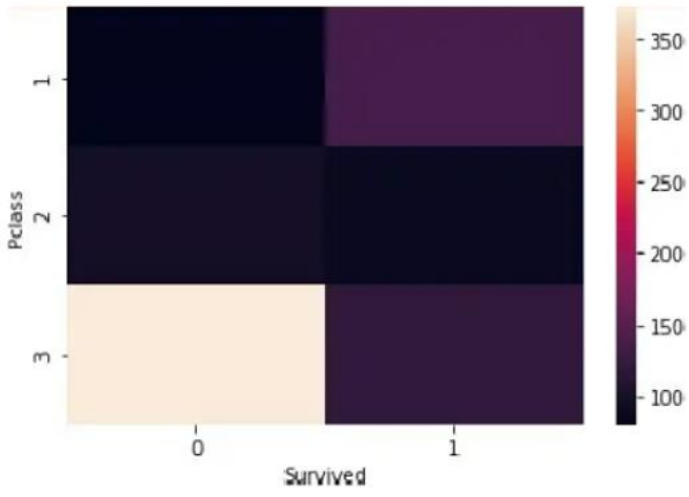
If you have ever used a crosstab function of pandas then Heatmap is a similar visual representation of that only. It basically shows that how much presence of one category concerning another category is present in the dataset. let me show first with crosstab and then with heatmap.

pd.crosstab(data['Pclass'], data['Survived'])

Survived	0	1
Pclass		
1	80	136
2	97	87
3	372	119

Now with heatmap, we have to find how many people survived and died.

sns.heatmap(pd.crosstab(data['Pclass'], data['Survived']))



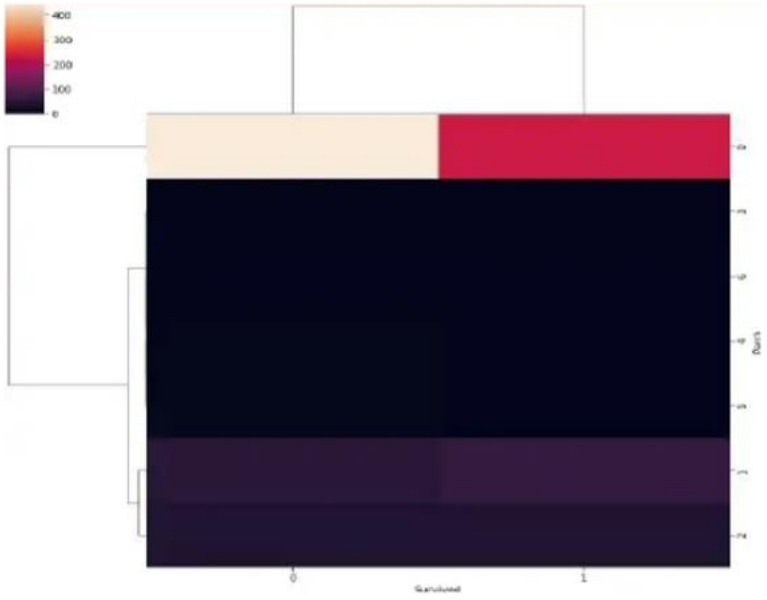
2) Cluster map:

We can also use a cluster map to understand the relationship between two categorical variables. A cluster map basically plots a dendrogram that shows the categories of similar behavior together.

```
sns.clustermap(pd.crosstab(data['Parch'], data['Survived']))

plt.show()
```

OUTPUT:



Viva Questions:

1. What is Data visualization?
2. How to calculate min,max,range and standard deviation?
3. How to create boxplot for each feature in the daset?
4. How to create histogram?