Practical No. 7:		
	Guru Gobind Singh Foundation's **Guru Gobind Singh College of Engineering and Research Centre, Nashik**	
Experiment No: 07		

Title of the Assignment:

1. Extract Sample document and apply following document pre-processing methods: Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization. 2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.

Student Name:			
Class:	**TE (Computer)**		
Div:		**Batch:**	
Roll No.:			
Date of Attendance (Performance):			
Date of Evaluation:			
Marks (Grade) Attainment of CO Marks out of 10			
CO Mapped			
Signature of Subject Teacher			

Objective of the Assignment: Students should be able to perform **Text Analysis** using TFIDF Algorithm

---

Prerequisite:

1. **Basic of Python Programming**
2. **Basic of English language.**

Contents for Theory:

1. **Basic concepts of Text Analytics**
2. **Text Analysis Operations using natural language toolkit**
3. **Text Analysis Model using TF-IDF.**
4. **Bag of Words (BoW)**
5. **Basic concepts of Text Analytics**

One of the most frequent types of day-to-day conversion is text communication. In our everyday routine, we chat, message, tweet, share status, email, create blogs, and offer opinions and criticism. All of these actions lead to a substantial amount of unstructured text being produced. It is critical to examine huge amounts of data in this sector of the online world and social media to determine people's opinions.

Text mining is also referred to as text analytics. Text mining is a process of exploring sizable textual data and finding patterns. Text Mining processes the text itself, while NLPprocesses with the underlying metadata. Finding frequency counts of words, length of the sentence, presence/absence of specific words is known as text mining. Natural language processing is one of the components of text mining. NLP helps identify sentiment, finding entities in the sentence, and category of blog/article. Text mining is preprocessed data for text analytics. In Text Analytics, statistical and machine learning algorithms are used to classify information.

6. **Text Analysis Operations using natural language toolkit**
7.

NLTK(natural language toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and many more.

Analysing movie reviews is one of the classic examples to demonstrate a simple NLP Bag-of-words model, on movie reviews.

8. Tokenization:

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentences is called Tokenization. Token is a single entity that is the building blocks for a sentence or paragraph.

- Sentence tokenization : split a paragraph into **list of sentences** using **sent_tokenize()** method
- Word tokenization : split a sentence into **list of words** using **word_tokenize()** method

9. Stop words removal

Stopwords considered as noise in the text. Text may contain stop words such as is,am, are, this, a, an, the, etc. In NLTK for removing stopwords, you need to createa list of stopwords and filter out your list of tokens from these words.

10. Stemming and Lemmatization

Stemming is a normalization technique where lists of tokenized words are converted into shortened root words to remove redundancy. Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form.
A computer program that stems word may be called a stemmer.E.g.
A stemmer reduces the words like fishing, fished, and fisher to the stem fish. The stem need not be a word, for example the Porter algorithm reduces, argue,argued, argues, arguing, and argus to the stem argu .

Lemmatization in NLTK is the algorithmic process of finding the lemma of a word depending on its meaning and context. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word known as the lemma.
Eg. Lemma for studies is study

11. Lemmatization Vs Stemming

Stemming algorithm works by cutting the suffix from the word. In a broader sensecuts either the beginning or end of the word.

On the contrary, Lemmatization is a more powerful operation, and it takes into consideration morphological analysis of the words. It returns the lemma which is the base form of all its inflectional forms. In-depth linguistic knowledge is

required to create dictionaries and look for the proper form of the word. Stemming is a

general operation while lemmatization is an intelligent operation where the proper form will be looked in the dictionary. Hence, lemmatization helps in forming better machine learning features.

12. POS Tagging

POS (Parts of Speech) tell us about grammatical information of words of the sentence by assigning specific token (Determiner, noun, adjective , adverb ,verb,Personal Pronoun etc.) as tag (DT,NN ,JJ,RB,VB,PRP etc) to each words.
Word can have more than one POS depending upon the context where it is used. We can use POS tags as statistical NLP tasks. It distinguishes a sense of word which is very helpful in text realization and infer semantic information from text for sentiment analysis.

13. Text Analysis Model using TF-IDF.

Term frequency–inverse document frequency(TFIDF) , is a numerical statistic that is intended to reflect how important a word is to a document in a collection orcorpus.

14. Term Frequency (TF)

It is a measure of the frequency of a word (w) in a document (d). TF is defined as the ratio of a word9s occurrence in a document to the total number of words in a document. The denominator term in the formula is to normalize since all the corpus documents are of different lengths.

$$TF(w,d) = \frac{occurences\ of\ w\ in\ document\ d}{total\ number\ of\ words\ in\ document\ d}$$

Example:

Documents	Text	Total number of words in a document
A	Jupiter is the largest planet	5
B	Mars is the fourth planet from the sun	8

The initial step is to make a vocabulary of unique words and calculate TF for each document. TF will be more for words that frequently appear in a document and less for rare words in a document.

Inverse Document Frequency (IDF)

It is the measure of the importance of a word. Term frequency (TF) does not consider the importance of words. Some words such as9 of9, 8and9, etc. can be most frequently present

Words	TF (for A)	TF (for B)	IDF	TFIDF (A)	TFIDF (B)
Jupiter	1/5	0	ln(2/1) = 0.69	0.138	0
Is	1/5	1/8	ln(2/2) = 0	0	0
The	1/5	2/8	ln(2/2) = 0	0	0
largest	1/5	0	ln(2/1) = 0.69	0.138	0
Planet	1/5	1/8	ln(2/2) = 0	0.138	0
Mars	0	1/8	ln(2/1) = 0.69	0	0.086
Fourth	0	1/8	ln(2/1) = 0.69	0	0.086
From	0	1/8	ln(2/1) = 0.69	0	0.086
Sun	0	1/8	ln(2/1) = 0.69	0	0.086

but are of little significance. IDF provides weightage to each word based on its frequency in the corpus D.

$$IDF(w, D) = \ln\left(\frac{Total\ number\ of\ documents\ (N)\ in\ corpus\ D}{number\ of\ documents\ containing\ w}\right)$$

In our example, since we have two documents in the corpus, N=2.

Words	TF (for A)	TF (for B)	IDF
Jupiter	1/5	0	ln(2/1) = 0.69
Is	1/5	1/8	ln(2/2) = 0
The	1/5	2/8	ln(2/2) = 0
largest	1/5	0	ln(2/1) = 0.69
Planet	1/5	1/8	ln(2/2) = 0
Mars	0	1/8	ln(2/1) = 0.69
Fourth	0	1/8	ln(2/1) = 0.69
From	0	1/8	ln(2/1) = 0.69
Sun	0	1/8	ln(2/1) = 0.69

Term Frequency — Inverse Document Frequency (TFIDF)

It is the product of TF and IDF.TFIDF gives more weightage to the word that is rare in the corpus (all the documents).TFIDF provides more importance to the word that is more frequent in the document.

$$TFIDF\ (w, d, D) = TF(w, d) * IDF(w, D)$$

After applying TFIDF, text in A and B documents can be represented as a TFIDF vector ofdimension equal to the vocabulary words. The value corresponding to each word representsthe importance of that word in a particular document.

TFIDF is the product of TF with IDF. Since TF values lie between 0 and 1, not using ***ln*** can result in

high IDF for some words, thereby dominating the TFIDF. We don9t want that, and therefore, we use ***ln*** so that the IDF should not completely dominate the TFIDF.

Disadvantage of TFIDF

It is unable to capture the semantics. For example, funny and humorous are synonyms, but TFIDF does not capture that. Moreover, TFIDF can be computationally expensive if the vocabulary is vast.

Bag of Words (BoW)

Machine learning algorithms cannot work with raw text directly. Rather, the text must beconverted into vectors of numbers. In natural language processing, a common technique for extracting features from text is to place all of the words that occur in the text in a bucket. This approach is called a bag of words model or BoW for short. It9s referred to as a <bag= of words because any information about the structure of the sentence is lost.

Algorithm for Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization:

Step 1: Download the required packages

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

Step 2: Initialize the text

```
text= "Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunkssuch as words or sentences is called Tokenization."
```

Step 3: Perform Tokenization

```
#Sentence Tokenization
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)

#Word  Tokenization
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

Step 4: Removing Punctuations and Stop Word

```
# print stop words of English
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
text= "How to remove stop words with NLTK library in Python?"text=
re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
      if w not in stop_words:
            filtered_text.append(w)
print("Tokenized Sentence:",tokens) print("Filterd
Sentence:",filtered_text)
```

Step 5 : Perform Stemming

```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"]ps
=PorterStemmer()
for w in e_words:
       rootWord=ps.stem(w)print(rootWord)
```

Step 6: Perform Lemmatization

```
 from nltk.stem import    WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer() text =
"studies studying cries cry" tokenization =
nltk.word_tokenize(text) for w in tokenization:
      print("Lemma       for       {}       is       {}".format(w,
      wordnet_lemmatizer.lemmatize(w)))
```

Step 7: Apply POS Tagging to text

```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
 for word in words: print(nltk.pos_tag([word]))
```

Algorithm for Create representation of document by calculating TFIDF Step 1: Import the necessary libraries.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

Step 2: Initialize the Documents.

```
documentA = 'Jupiter is the largest Planet' documentB =
'Mars is the fourth planet from the              Sun'
```

Step 3: Create BagofWords (BoW) for Document A and B.

```
bagOfWordsA = documentA.split(' ')
 bagOfWordsB = documentB.split(' ')
```

Step 4: Create Collection of Unique words from Document A and B.

```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

Step 5: Create a dictionary of words and their occurrence for each document in the corpus

```
numOfWordsA = dict.fromkeys(uniqueWords, 0)

for word in bagOfWordsA:
      numOfWordsA[word] += 1
      numOfWordsB = dict.fromkeys(uniqueWords, 0)for
word in bagOfWordsB:
      numOfWordsB[word] += 1
```

Step 6: Compute the term frequency for each of our documents.

```
def computeTF(wordDict, bagOfWords):
tfDict = {}
bagOfWordsCount =  len(bagOfWords) for
word, count in  wordDict.items():
    tfDict[word] = count / float(bagOfWordsCount)
return tfDict
tfA = computeTF(numOfWordsA, bagOfWordsA)tfB =
computeTF(numOfWordsB,  bagOfWordsB)
```

Step 7: Compute the term Inverse Document Frequency.

```
 def computeIDF(documents):
import math
N = len(documents)


idfDict = dict.fromkeys(documents[0].keys(), 0)for
document in documents:
    for word, val in document.items():if
        val > 0:
            idfDict[word] += 1


for word, val in idfDict.items(): idfDict[word] =
    math.log(N / float(val))
return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])idfs
```

Step 8: Compute the term TF/IDF for all words.

```
def computeTFIDF(tfBagOfWords, idfs):tfidf
= {}
for word, val in tfBagOfWords.items():
    tfidf[word] = val * idfs[word]
return tfidf
```

```
tfidfA = computeTFIDF(tfA, idfs) tfidfB
=  computeTFIDF(tfB,  idfs)df =
pd.DataFrame([tfidfA, tfidfB])df
```

Conclusion:

In this way we have done text data analysis using TF IDF algorithm.

Assignment Question:

1) **Perform Stemming for** `text = "studies studying cries cry".` **Compare the results generated with Lemmatization. Comment on your answer how Stemming and Lemmatization differ from each other.**

Write Python code for removing stop words from the below documents, conver the documents into lowercase and calculate the TF, IDF and TFIDF score for each document.

```
documentA = 'Jupiter is the largest Planet' documentB =
'Mars is the fourth planet from the Sun'
```

Group A

Assignment No: 8

Contents for Theory:

1. **Seaborn Library Basics 2. Know your Data**
3. **Finding patterns of data.**
4. **Checking how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.**

Theory:

Data Visualisation plays a very important role in Data mining. Various data scientists spent their time exploring data through visualisation. To accelerate this process we need to have a welldocumentation of all the plots.

Even plenty of resources can't be transformed into valuable goods without planning and architecture

1. Seaborn Library Basics

Seaborn is a Python data visualisation library based on matplotlib. It provides a highlevel interface for drawing attractive and informative statistical graphics.

For the installation of Seaborn, you may run any of the following in your command line.

```
pip install seaborn conda
install seaborn
```

To import seaborn you can run the following command.

```
import seaborn as sns
```

2. Know your data

The dataset that we are going to use to draw our plots will be the Titanic dataset, which is downloaded by default with the Seaborn library. All you have to do is use the load_dataset function and pass it the name of the dataset.

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.

3. 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

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

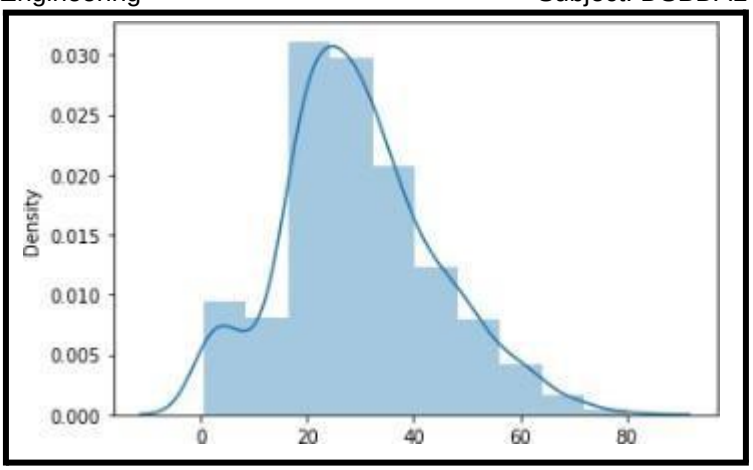
A. Distribution Plots:

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

a. Distplot

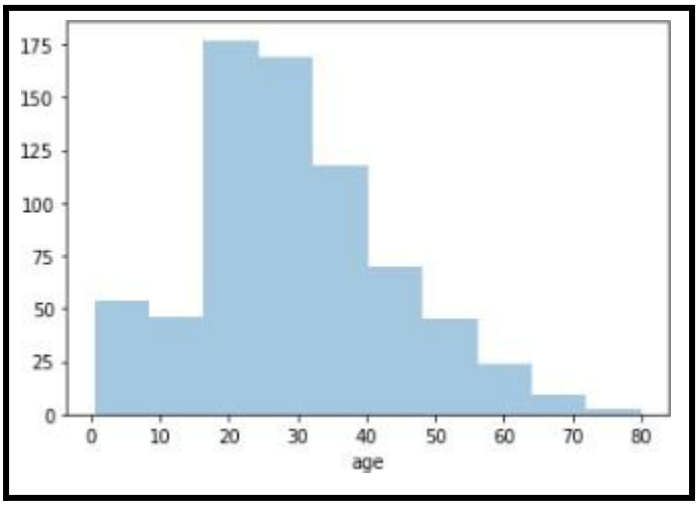
- Dist plot gives us the histogram of the selected continuous variable.
- It is an example of a univariate analysis.
- We can change the number of bins i.e. number of vertical bars in a histogram

```
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 line by 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.

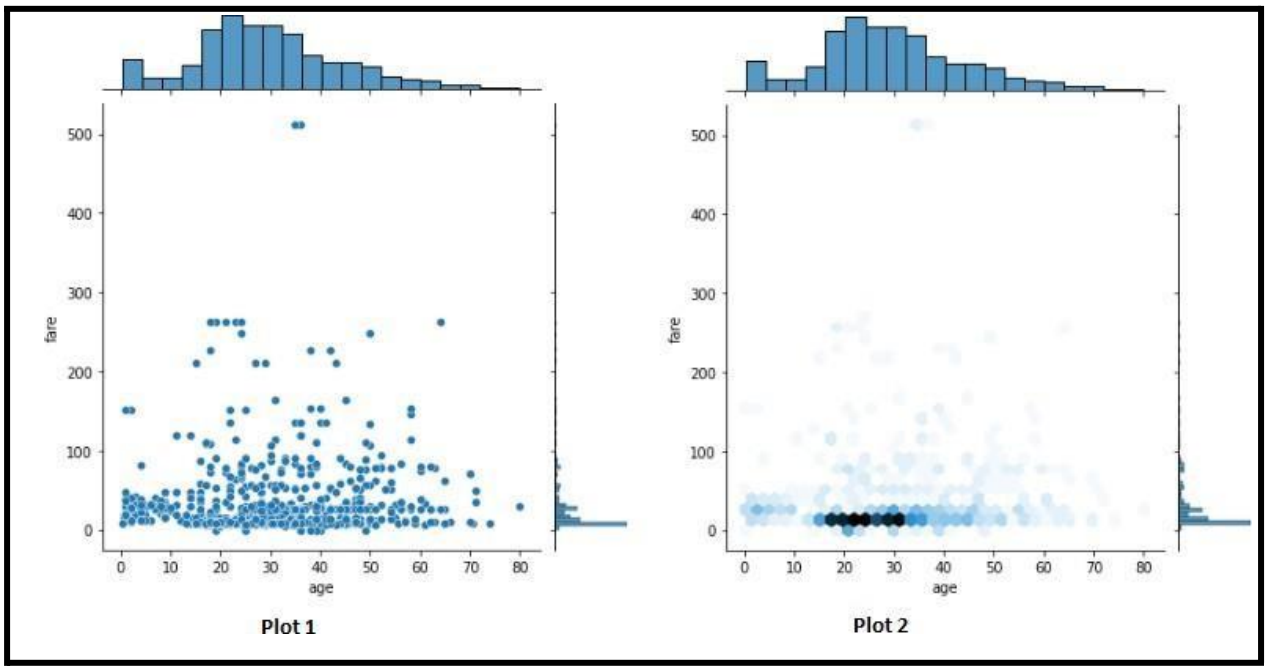
- We additionally obtain a scatter plot between the variables to reflect their linear relationship. We can customise the scatter plot into a hexagonal plot, where, the more the colour intensity, the more will be the number of observations.

```
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 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. You can see that there is no correlation observed between prices and the fares.

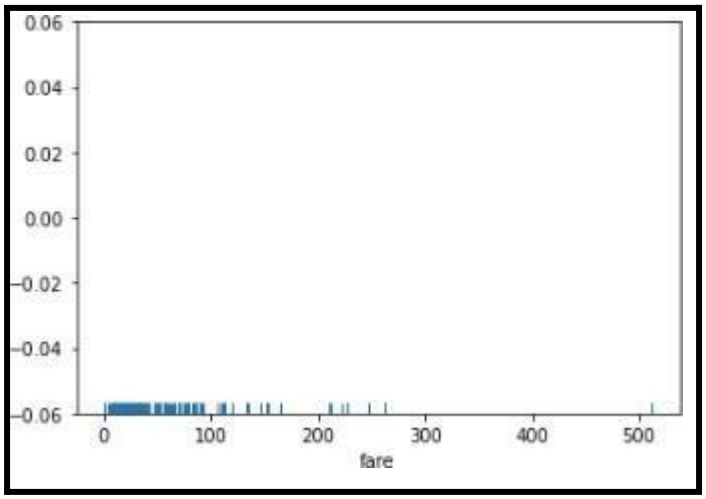
- You can change the type of the joint plot by passing a value for the kind parameter. For instance, if instead of a scatter plot, you want to display the distribution of data in the form of a hexagonal plot, you can pass the value hex for the kind parameter.

- 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 the ages 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.

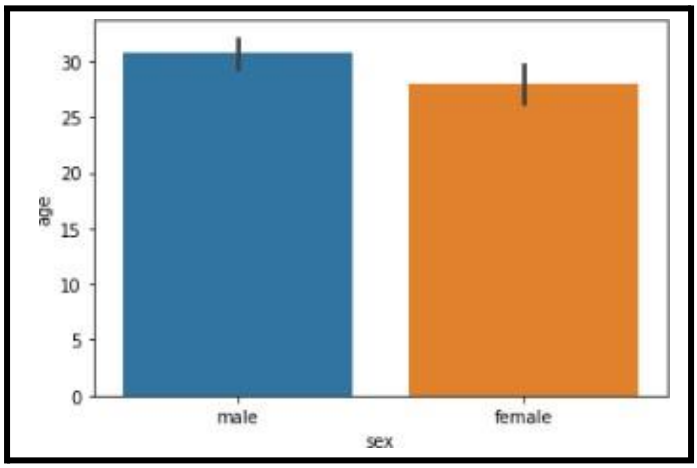
2. Categorical Plots

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 or a 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.

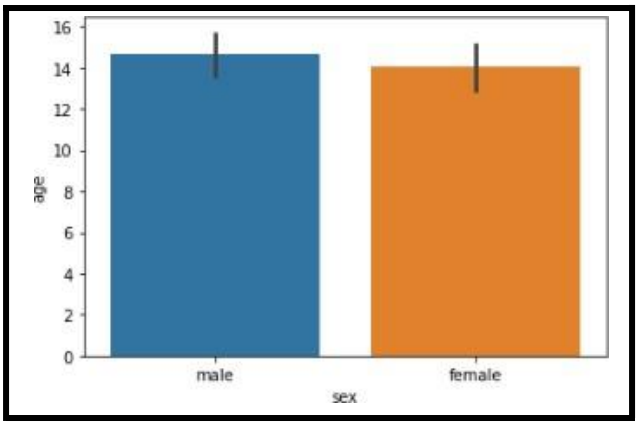
In addition to finding the average, the bar plot can also be used to calculate other aggregate values for each category. To do so, you need to pass the aggregate function to the estimator. For instance, you can calculate the standard deviation for the age of each gender as follows:

```
import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

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

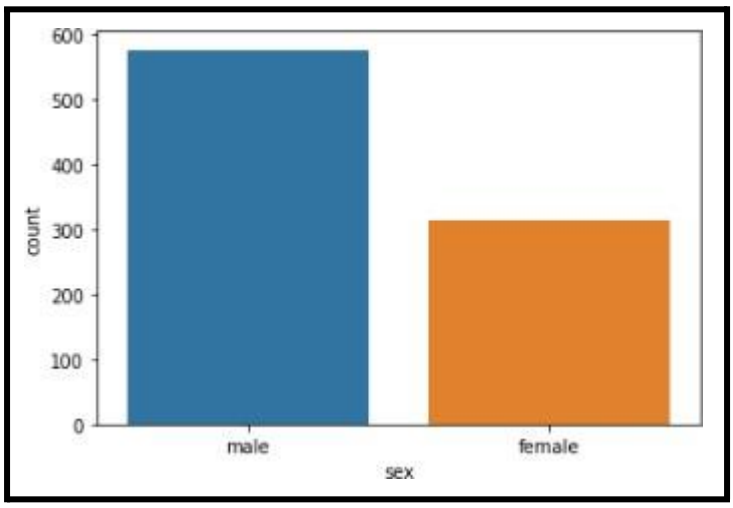
Notice, in the above script we use the std aggregate function from the numpy library to calculate the standard deviation for the ages of male and female passengers. The output looks like this:



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 we can do so using count plot as follows:

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

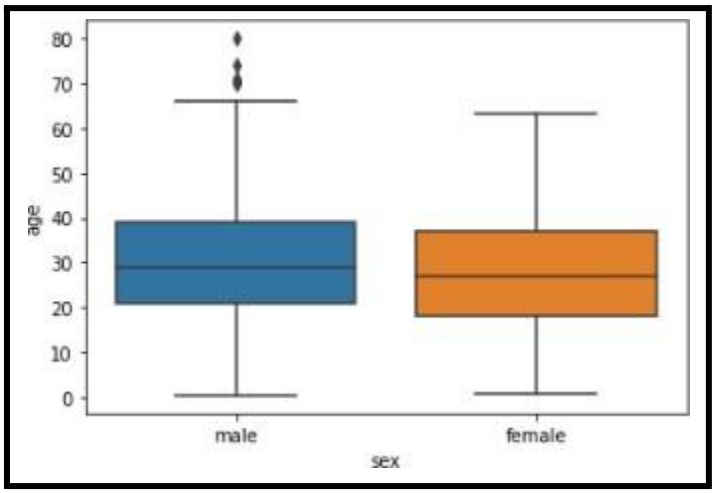


d. The Box Plot

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)
```

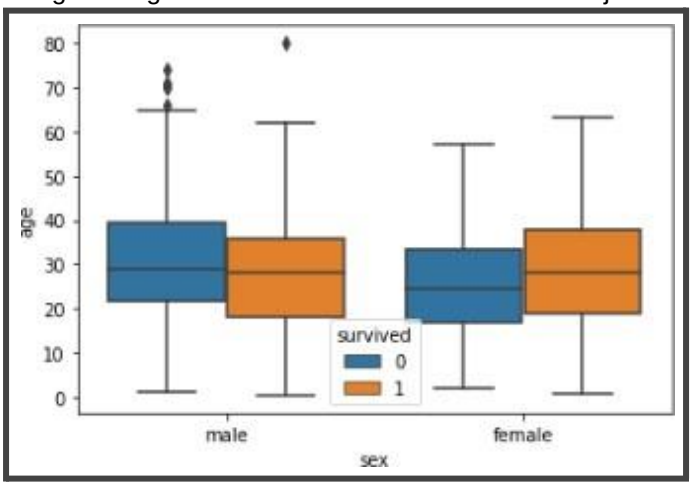


Let's try to understand the box plot for females. The first quartile starts at around 1 and ends at 20 which means that 25% of the passengers are aged between 1 and 20. The second quartile starts at around 20 and ends at around 28 which means that 25% of the passengers are aged between20 and 28. Similarly, 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")
```



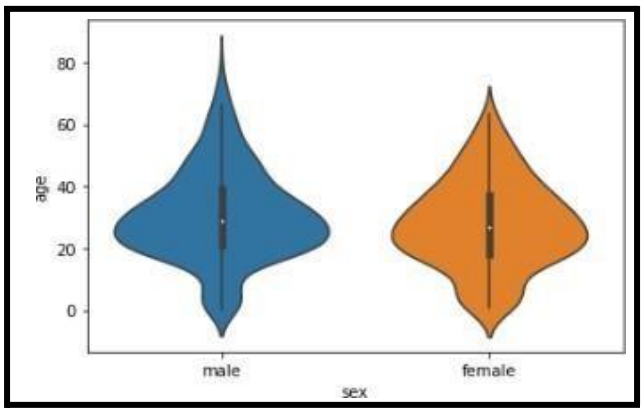
Now in addition to the information about the age of each gender, you can also see the distribution of the passengers who survived. For instance, you can see that among the male passengers, on average more younger people survived as compared to the older ones. Similarly, you can see that the variation 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.

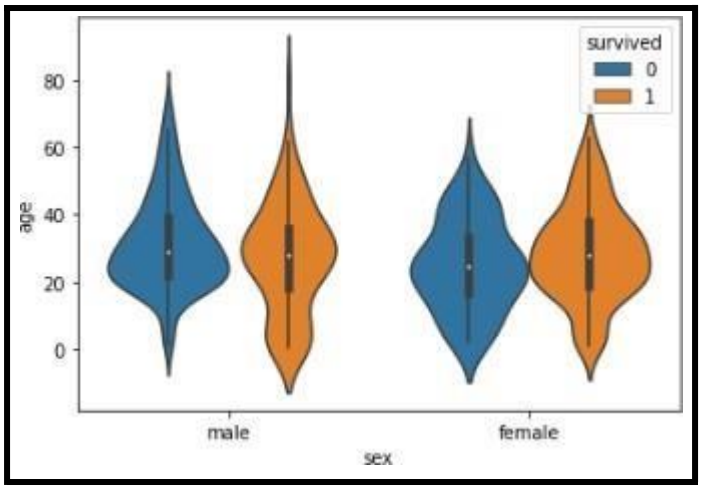
```
sns.violinplot(x='sex', y='age', data=dataset)
```



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:

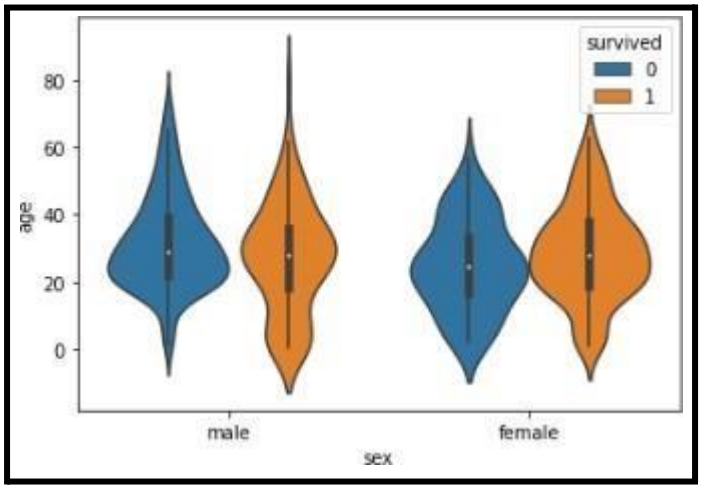
a. The Strip Plot

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.

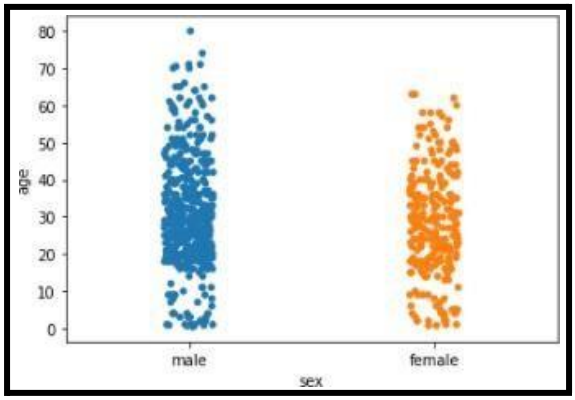
The stripplot() function is used to plot the violin plot. Like the box plot, the first parameter is the categorical column, the second parameter is the numeric column while the third parameter is the dataset. Look at the following script:

```
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 the data, pass True for the jitter parameter which adds some random noise to the data. Look at the following script:

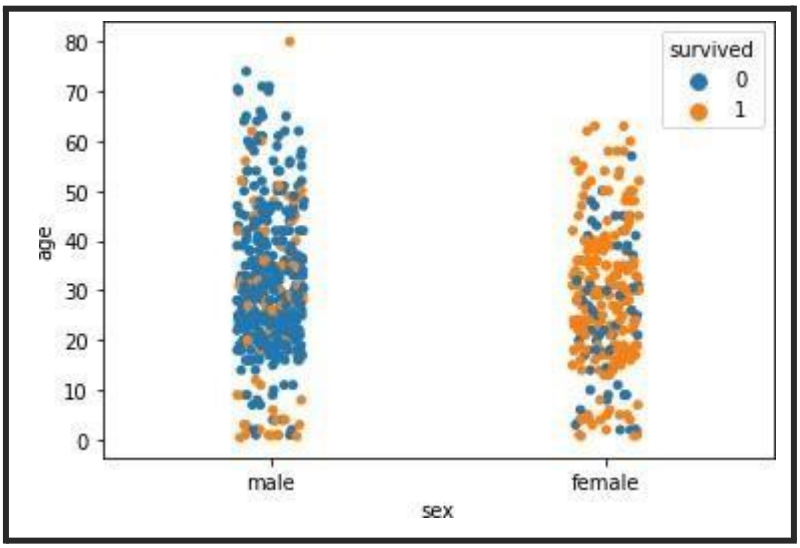
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True)
```



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:

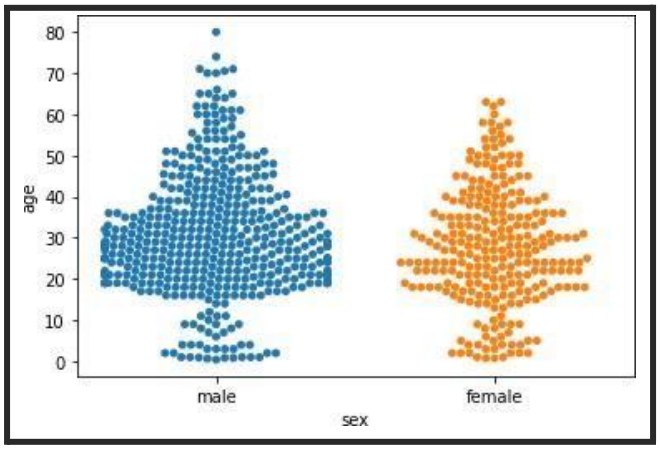
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True, hue='survived')
```



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:

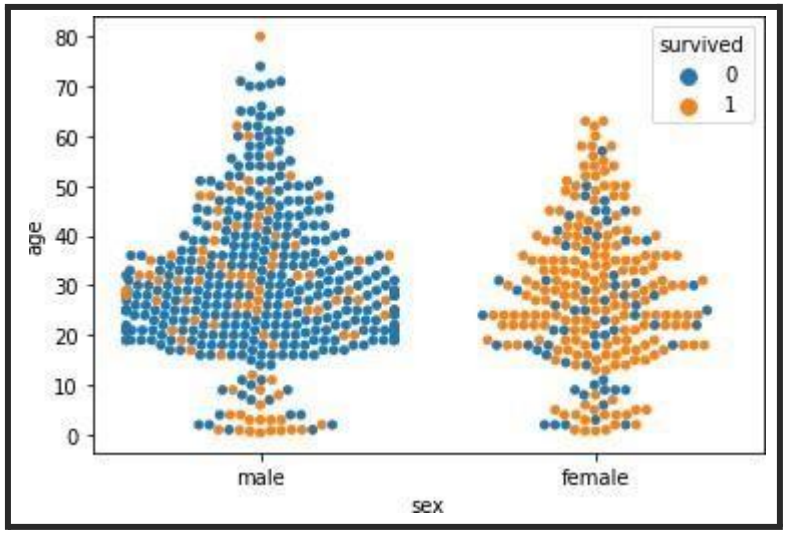
```
sns.swarmplot(x='sex', y='age', data=dataset)
```



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 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

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:

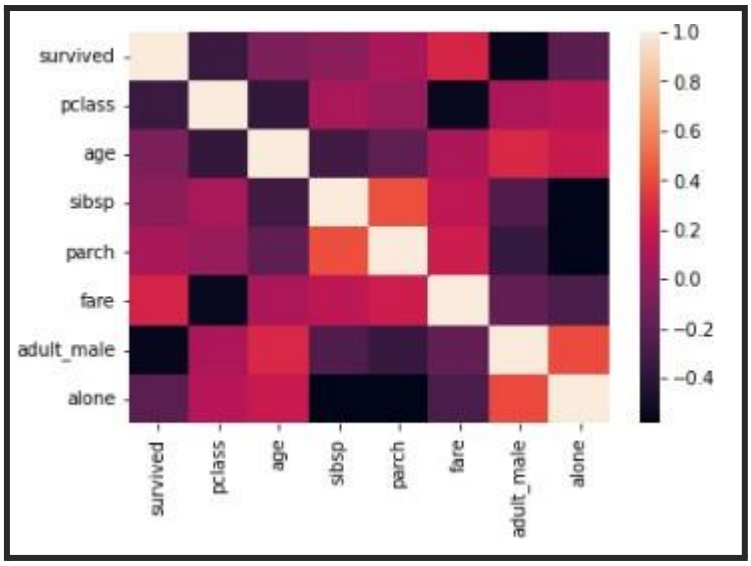
```
dataset.corr()
```

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)
```



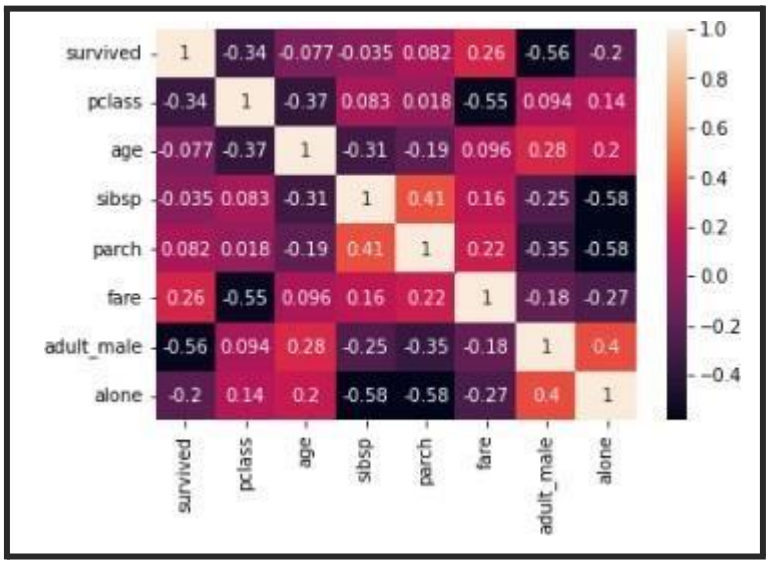
From the output, it can be seen that what heatmap essentially does is that it plots a box for every combination of rows and column value. The colour of the box depends upon the gradient. For instance, in the above image if there is a high correlation between two features, the corresponding cell or the 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:

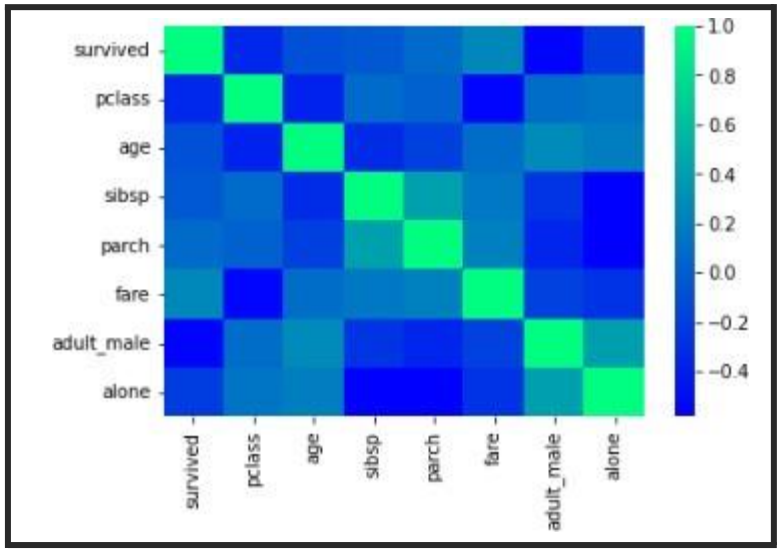
```
corr = dataset.corr() sns.heatmap(corr,

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:

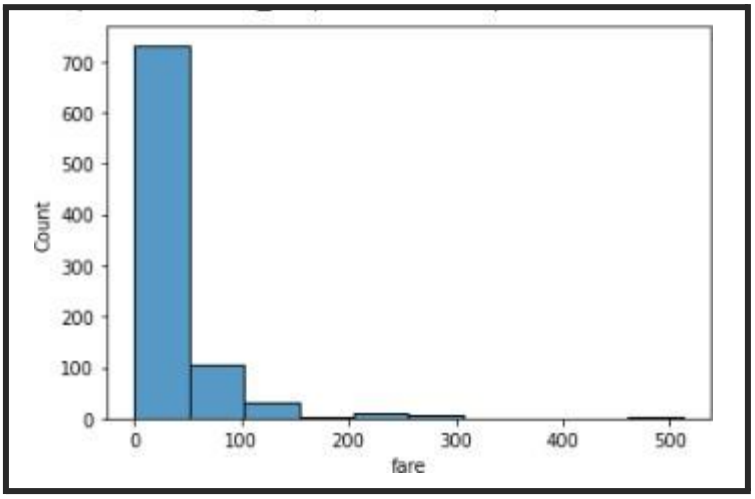
In addition to the heat map, another commonly used matrix plot is the cluster map. The cluster map basically uses Hierarchical Clustering to cluster the rows and columns of the matrix.

Let's plot a cluster map for the number of passengers who travelled in a specific month of a 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.

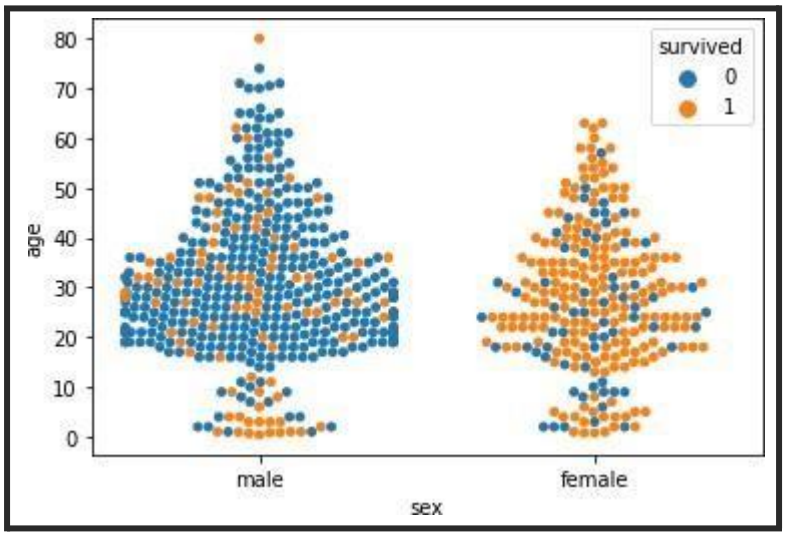
Conclusion-

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 Seaborn library. 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.

Assignment Questions

1. **List out different types of plot to find patterns of data**
2. **Explain when you will use distribution plots and when you will use categorical plots.**

3. Write the conclusion from the following swarm plot (consider titanic dataset)



4. Which parameter is used to add another categorical variable to the violin plot, Explain with syntax and example.

Group A

Assignment No: 9

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

---

Prerequisite:

1. Basic of Python Programming
2. Seaborn Library, Concept of Data Visualization.

An introduction to seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures.

Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them.

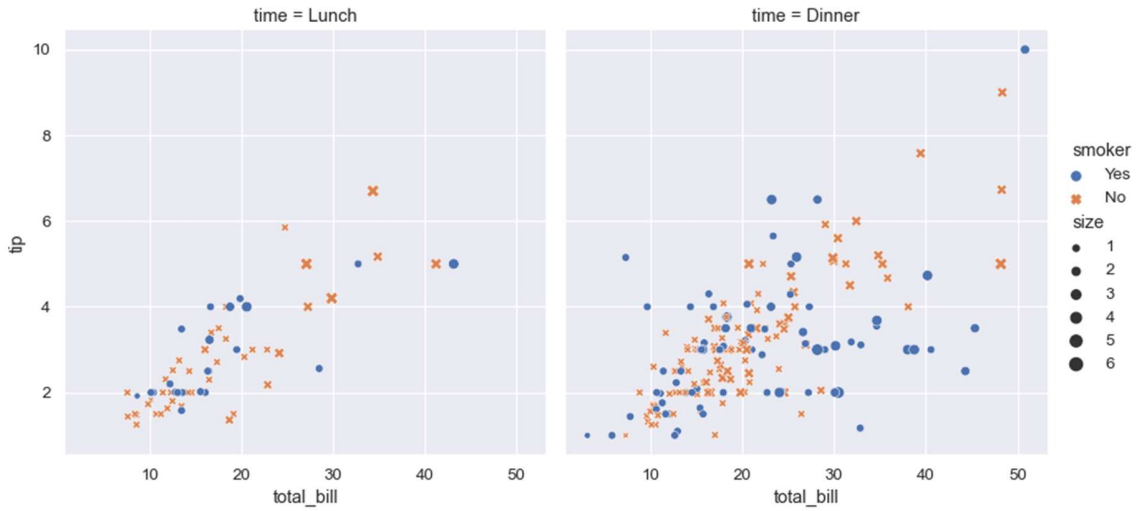
Here's an example of what seaborn can do:

```
# Import seaborn
import seaborn as sns

# Apply the default theme
sns.set_theme()

# Load an example dataset
tips = sns.load_dataset("tips")

# Create a visualization
sns.relplot(
    data=tips,
    x="total_bill", y="tip", col="time",
    hue="smoker", style="smoker", size="size",
)
```



A few things have happened here. Let's go through them one by one:

```
# Import seaborn
import seaborn as sns
```

Seaborn is the only library we need to import for this simple example. By convention, it is imported with the shorthand sns.

Behind the scenes, seaborn uses matplotlib to draw its plots. For interactive work, it's recommended to use a Jupyter/IPython interface in matplotlib mode, or else you'll have to call **matplotlib.pyplot.show()** when you want to see the plot.

```
# Apply the default theme
sns.set_theme()
```

This uses the matplotlib rcParam system and will affect how all matplotlib plots look, even if you don't make them with seaborn. Beyond the default theme, there are several other options, and you can independently control the style and scaling of the plot to quickly translate your work between presentation contexts (e.g., making a version of your figure that will have readable fonts when projected during a talk). If you like the matplotlib defaults or prefer a different theme, you can skip this step and still use the seaborn plotting functions.

```
# Load an example dataset
tips = sns.load_dataset("tips")
```

Most code in the docs will use the **load_dataset()** function to get quick access to an example dataset. There's nothing special about these datasets: they are just pandas dataframes, and we could have loaded them with **pandas.read_csv()** or built them by hand. Most of the examples in the documentation will specify data using pandas dataframes, but seaborn is very flexible about the data structures that it accepts.

```
# Create a visualization
sns.relplot(
    data=tips,
    x="total_bill", y="tip", col="time",
    hue="smoker", style="smoker", size="size",
)
```

Assignment Questions

1. Write down the code to use inbuilt dataset 'titanic' using seaborn library. 2. Write code to plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not.

3. Write the observations from the box plot.

Group A

Assignment No: 10

Contents for Theory:

Implement a dataset into a dataframe. Implement the following operations:

1. Display data set details.
2. Calculate min, max ,mean, range, standard deviation, variance.
3. Create histograph using hist function.
4. Create boxplot using boxplot function.

Theory:

How to Find the Mean, Median, Mode, Range, and Standard Deviation

Simplify comparisons of sets of number, especially large sets of number, by calculating the center values using mean, mode and median. Use the ranges and standard deviations of the sets to examine the variability of data.

Calculating Mean

The mean identifies the average value of the set of numbers. For example, consider the data set containing the values 20, 24, 25, 36, 25, 22, 23.

Formula

To find the mean, use the formula: Mean equals the sum of the numbers in the data set divided by the number of values in the data set. In mathematical terms: Mean=(sum of all terms)÷(how many terms or values in the set).

Adding Data Set

Add the numbers in the example data set: 20+24+25+36+25+22+23=175.

Finding Divisor

Divide by the number of data points in the set. This set has seven values so divide by 7.

Finding Mean

Insert the values into the formula to calculate the mean. The mean equals the sum of the values (175) divided by the number of data points (7). Since 175÷7=25, the mean of this data set equals 25. Not all mean values will equal a whole number.

Calculating Range

Range shows the mathematical distance between the lowest and highest values in the data set. Range measures the variability of the data set. A wide range indicates greater variability in the data, or perhaps a single outlier far from the rest of the data. Outliers may skew, or shift, the mean value enough to impact data analysis.

Identifying Low and High Values

In the sample group, the lowest value is 20 and the highest value is 36.

Calculating Range

To calculate range, subtract the lowest value from the highest value. Since 36-20=16, the range equals 16.

Calculating Standard Deviation

Standard deviation measures the variability of the data set. Like range, a smaller standard deviation indicates less variability.

Formula

Finding standard deviation requires summing the squared difference between each data point and the mean [$\sum(x-\mu)^2$], adding all the squares, dividing that sum by one less than the number of values (N-1), and finally calculating the square root of the dividend.

Mathematically, start with calculating the mean.

Calculating the Mean

Calculate the mean by adding all the data point values, then dividing by the number of data points. In the sample data set, 20+24+25+36+25+22+23=175. Divide the sum, 175, by the number of data points, 7, or 175÷7=25. The mean equals 25.

Squaring the Difference

Next, subtract the mean from each data point, then square each difference. The formula looks like this: $\sum(x-\mu)^2$, where $\sum$ means sum, x represents each data set value and μ represents the mean value. Continuing with the example set, the values become: 20-25=-5 and $-5^2=25$; 24-25=-1 and $-1^2=1$; 25-25=0 and $0^2=0$; 36-25=11 and $11^2=121$; 25-25=0 and $0^2=0$; 22-25=-3 and $-3^2=9$; and 23-25=-2 and $-2^2=4$.

Adding the Squared Differences

Adding the squared differences yields: 25+1+0+121+0+9+4=160. The example data set has 7 values, so N-1 equals 7-1=6. The sum of the squared differences, 160, divided by 6 equals approximately 26.6667.

Standard Deviation

Calculate the standard deviation by finding the square root of the division by N-1. In the example, the square root of 26.6667 equals approximately 5.164. Therefore, the standard deviation equals approximately 5.164.

Evaluating Standard Deviation

Standard deviation helps evaluate data. Numbers in the data set that fall within one standard deviation of the mean are part of the data set. Numbers that fall outside of two standard deviations are extreme values or outliers. In the example set, the value 36 lies more than two standard deviations from the mean, so 36 is an outlier. Outliers may represent erroneous data or may suggest unforeseen circumstances and should be carefully considered when interpreting data.

Facilities: Windows/Linux Operating Systems, RStudio, jdk.

Application:

1. The histogram is suitable for visualizing distribution of numerical data over a continuous interval, or a certain time period. The histogram organizes large amounts of data, and produces visualization quickly, using a single dimension.

2. The box plot allows quick graphical examination of one or more data sets. Box plots may seem more primitive than a histogram but they do have some advantages. They take up less space and are therefore particularly useful for comparing distributions between several groups or sets of data. Choice of number and width of bins techniques can heavily influence the appearance of a histogram, and choice of bandwidth can heavily influence the appearance of a kernel density estimate.

3. Data Visualization Application lets you quickly create insightful data visualizations, in minutes.

Data visualization tools allow anyone to organize and present information intuitively. They enables users to share data visualizations with others.

Input:

Structured Dataset: Iris

Dataset File: iris.csv

Output:

1. Display Dataset Details.
2. Calculate Min, Max, Mean, Variance value and Percentiles of probabilities also Display Specific use quantile.
3. Display the Histogram using Hist Function.
4. Display the Boxplot using Boxplot Function.

Conclusion:

Hence, we have studied using dataset into a dataframe and compare distribution and identify outliers.

Assignment Questions

1. **For the iris dataset, list down the features and their types.**
2. **Write a code to create a histogram for each feature. (iris dataset)**
3. **Write a code to create a boxplot for each feature. (iris dataset)**
4. **Identify the outliers from the boxplot drawn for iris dataset.**