NAME: Sanjay
Kishan D

ROLL NO :230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:30.07.2024

```
import pandas as pd import
numpy as np import
seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
data=pd.read_csv('/content/Iris_Dataset.csv') data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	variety
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149  Data columns
(total 6 columns):
 # Column Non-Null Count Dtype
-----  ----------  -----------------------  ---------
 0  Id 150 non-null int64
 1  SepalLengthCm 150 non-null float64
 2  SepalWidthCm 150 non-null float64
 3  PetalLengthCm 150 non-null float64
 4  PetalWidthCm 150 non-null float64
 5  variety 150 non-null object
dtypes: float64(4), int64(1), object(1) memory usage:
7.2+ KB
```

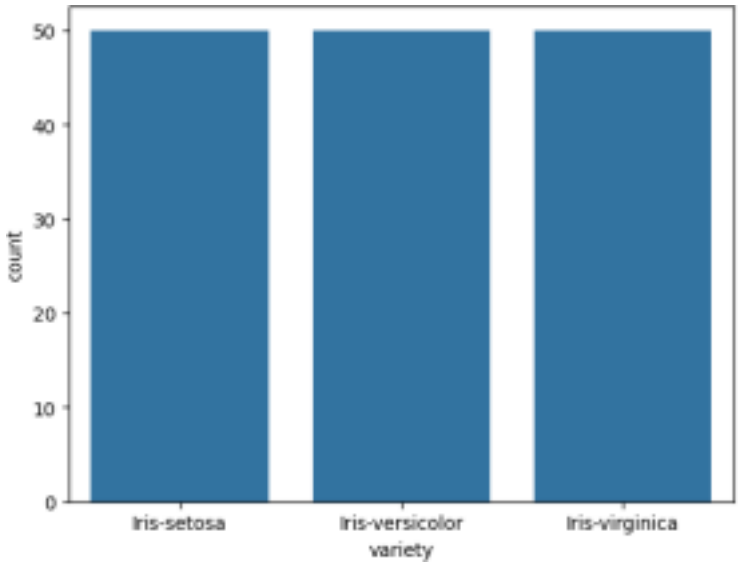
```
data.describe()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150 000000	7 900000	4 400000	6 900000	2 500000

```
data.value_counts('variety')
```

variety	count
Iris-setosa	50
Iris-versicolor	50

```
sns.countplot(x='variety',data=data,) plt.show()
```




```
dummies=pd.get_dummies(data.variety)
```

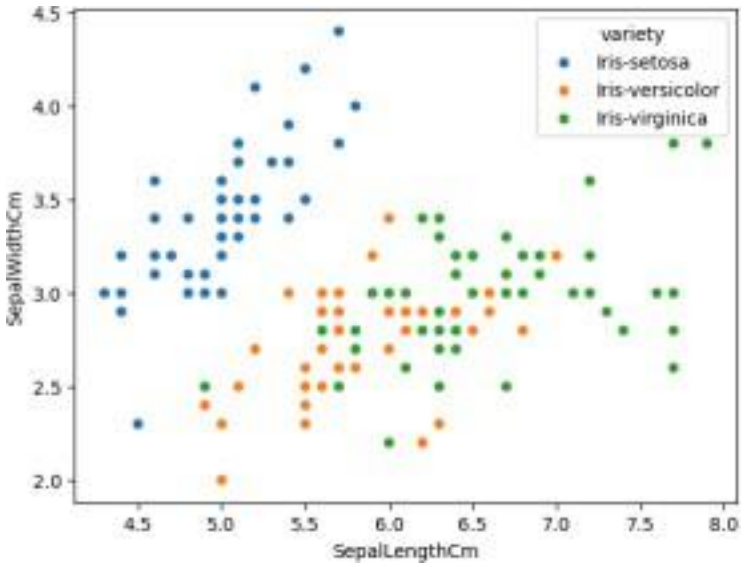
```
FinalDataset=pd.concat([pd.get_dummies(data.variety),data.iloc[:,[0,1,2,3]]],axis=1)
```

```
FinalDataset.head()
```

	Iris-setosa	Iris-versicolor	Iris-virginica	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	
0	True	False	False	1	5.1	3.5	1.4	
1	True	False	False	2	4.9	3.0	1.4	
2	True	False	False	3	4.7	3.2	1.3	
3	True	False	False	4	4.6	3.1	1.5	
4	True	False	False	5	5	0	3	6 1 4

```
sns.scatterplot(x='SepalLengthCm',y='SepalWidthCm',hue='variety',data=data,)
```

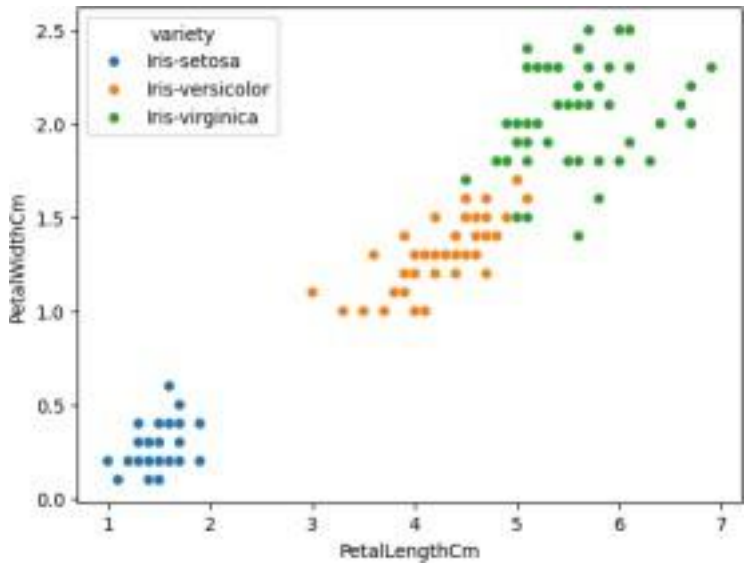
```
<Axes: xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



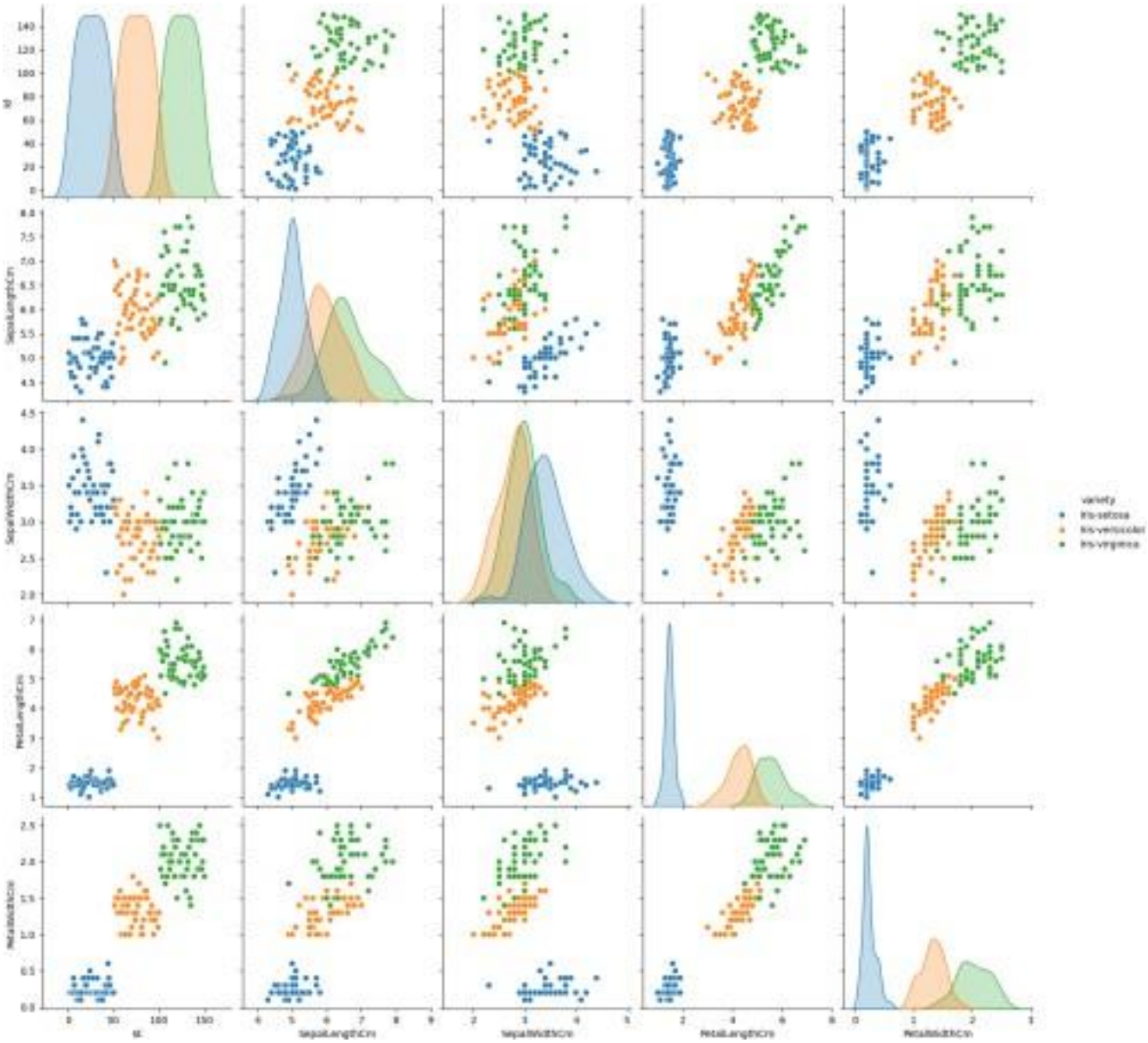

```
sns.scatterplot(x='PetalLengthCm',y='PetalWidthCm',hue='variety',data=data,)
```



```
<Axes: xlabel='PetalLengthCm', ylabel='PetalWidthCm'>
```

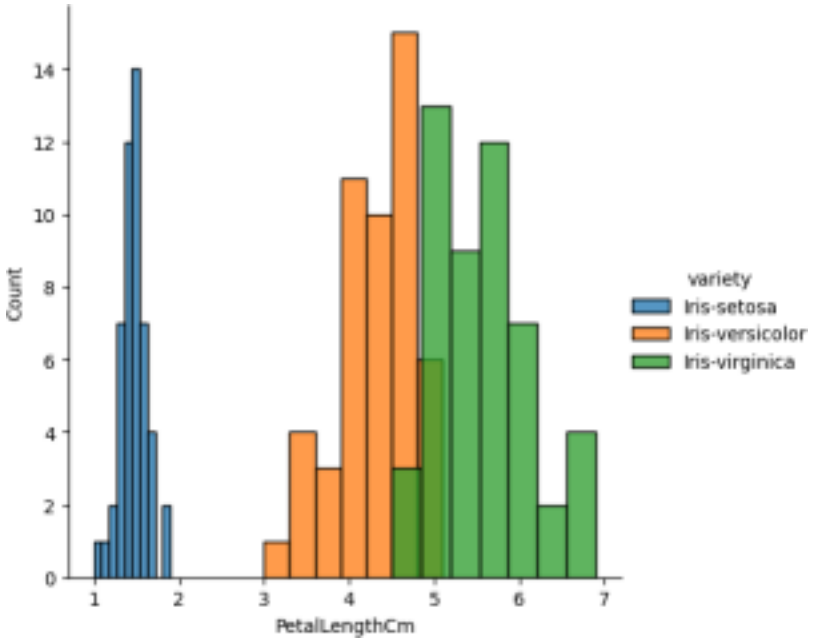



```
sns.pairplot(data,hue='variety',height=3);
```

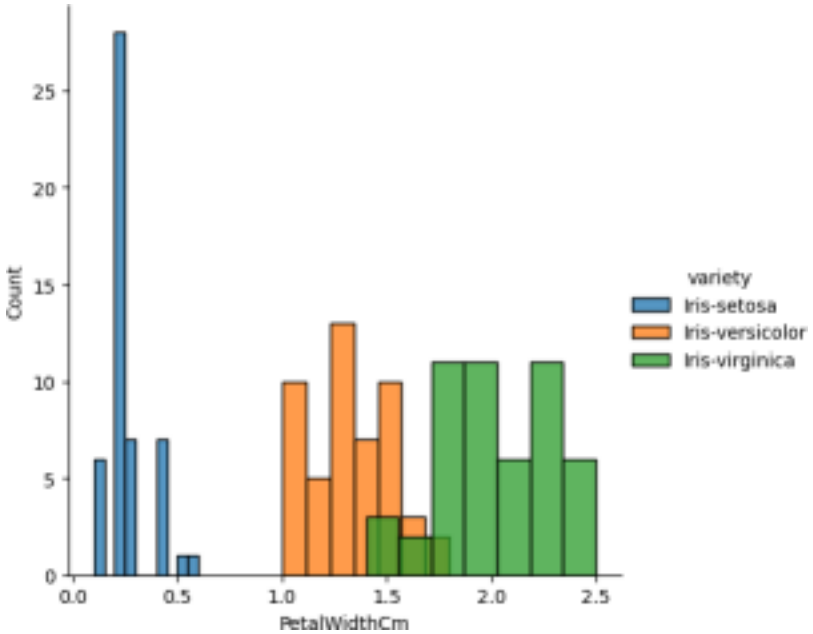


 plt.show()

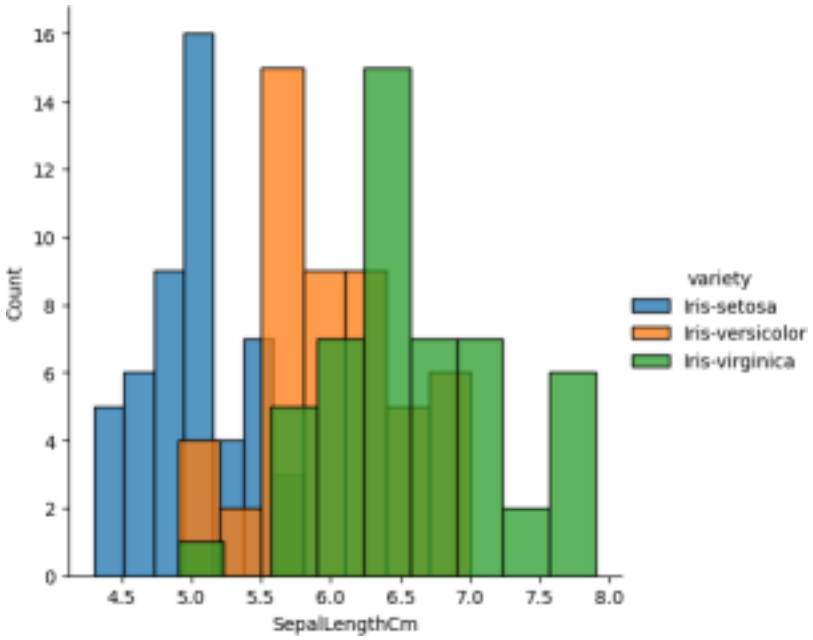
```
sns.FacetGrid(data,hue='variety',height=5).map(sns.histplot,'PetalLengthCm').add_legend();  plt.show();
```



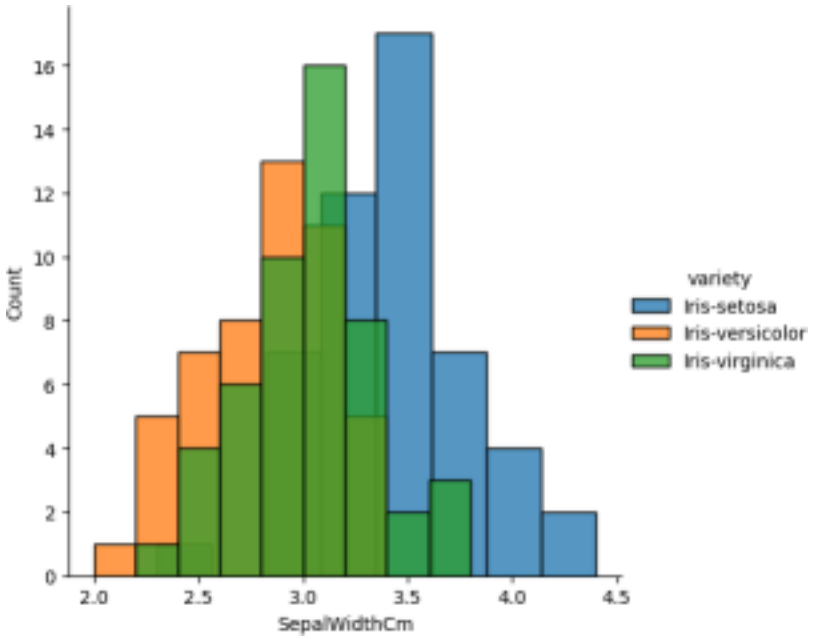

```
sns.FacetGrid(data,hue='variety',height=5).map(sns.histplot,'PetalWidthCm').add_legend();  plt.show();
```




```
sns.FacetGrid(data,hue='variety',height=5).map(sns.histplot,'SepalLengthCm').add_legend();  plt.show();
```




```
sns.FacetGrid(data,hue='variety',height=5).map(sns.histplot,'SepalWidthCm').add_legend();  plt.show();
```



NAME: SANJAY
KISHAN D

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:06.08.2024

```
import numpy as np
array=np.random.randint(1,100,9) array
```

```
array([83, 25, 19, 47, 62, 15, 96, 39, 51])
```

```
np.sqrt(array)
```

```
array([9.11043358, 5. , 4.35889894, 6.8556546 , 7.87400787,
       3.87298335, 9.79795897, 6.244998 , 7.14142843])
```

```
array.ndim 1
```

```
new_array=array.reshape(3,3)
```

```
new_array
```

```
array([[83, 25, 19],
       [47, 62, 15],
       [96, 39, 51]])
```

```
new_array.ndim
```

```
2
```

```
new_array.ravel()
```

```
array([83, 25, 19, 47, 62, 15, 96, 39, 51])
```

```
newm=new_array.reshape(3,3)
```

```
newm
```

```
array([[83, 25, 19],
       [47, 62, 15],
       [96, 39, 51]])
```

```
newm[2,1:3]
```

```
array([39, 51])
```

```
newm[1:2,1:3]
```

```
array([[62, 15]])
```

```
new_array[0:3,0:0]
```

```
array([], shape=(3, 0), dtype=int64)
```

```
new_array[0:2,0:1]
```

```
array([[83],
       [47]])
```

```
new_array[0:3,0:1]
```

```
array([[83], [47],
       [96]])
```

```
new_array[1:3]
```

```
array([[47, 62, 15],
       [96, 39, 51
```

BHISHEK S

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:13.08.2024

```
import numpy as np import
pandas as pd
list=[[1,'Smith',50000],[2,'Jones',60000]]


df=pd.DataFrame(list) df
```

	0	1	2
0	1	Smith	50000
1	2	Jones	60000

```
df.columns=['Empd','Name','Salary'] df
```

	Empd	Name	Salary
0	1	Smith	50000
1	2	Jones	60000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
2 entries, 0 to 1
Data columns (total 3 columns): # Column
 Non-Null Count Dtype
-----  -----------  -------------------------  ---------
 0  Empd 2 non-null int64
 1  Name 2 non-null object
 2  Salary 2 non-null int64 dtypes:
int64(2), object(1) memory usage:
176.0+ bytes
```

```
df=pd.read_csv("/content/50_Startups.csv")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
50 entries, 0 to 49
Data columns (total 5 columns): # Column
 Non-Null Count Dtype
-----  -----------  -------------------------  ---------
 0  R&D Spend 50 non-null float64
 1  Administration 50 non-null float64
 2  Marketing Spend 50 non-null float64
 3  State 50 non-null object
 4  Profit  50  non-null  float64  dtypes:
float64(4),  object(1)  memory  usage:
2.1+ KB
```

```
df.head()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107 34	91391 77	366168 42	Florida	166187 94

```
df.tail()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
45	1000.23	124153.04	1903.93	New York	64926.08
46	1315.46	115816.21	297114.46	Florida	49490.75
47	0.00	135426.92	0.00	California	42559.73
48	542.05	51743.15	0.00	New York	35673.41
49	0 00	116983 80	45173 06	California	14681 40

```
import numpy as np import
pandas as pd
df=pd.read_csv("/content/employee.csv")
```

```
df.head()
```

	emp id	name	salary
0	1	SREE VARSSINI K S	5000
1	2	SREEMATHI B	6000
2	3	SREYA G	7000
3	4	SREYASKARI MULLAPUDI	5000
4	5	SRI AKASH U G	8000

```
df.tail()
```

	emp id	name	salary
2	3	SREYA G	7000
3	4	SREYASKARI MULLAPUDI	5000
4	5	SRI AKASH U G	8000
5	6	SRI HARSHAVARDHANAN R	3000
6	7	SRI HARSHAVARDHANAN R	6000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
7 entries, 0 to 6
Data columns (total 3 columns): # Column
 Non-Null Count Dtype
------  ----------  -------------------------  ---------
 0  emp id 7 non-null int64
 1  name 7 non-null object
 2  salary 7 non-null int64 dtypes:
int64(2), object(1) memory usage:
296.0+ bytes
```

```
df.salary
```

	salary
0	5000
1	6000
2	7000
3	5000
4	8000
5	3000
6	6000

```
type(df.salary)
```

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None, fastpath: bool=False) ->
None

One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object supports both integer- and
label-based indexing and provides a host of methods for performing operations involving the
index. Statistical
    th d f d h b idd t t ti ll l d
```

```
df.salary.mean()
```

```
5714.285714285715
```



```
df.salary.median() 6000.0
```

```
df.salary.mode()
```

	salary
0	5000
1	6000

```
df.salary.var()
```

```
2571428.5714285714
```

```
df.salary.std()
```

```
1603.5674514745463
```

```
df.describe()
```

	emp id	salary
count	7.000000	7.000000
mean	4.000000	5714.285714
std	2.160247	1603.567451
min	1.000000	3000.000000
25%	2.500000	5000.000000
50%	4.000000	6000.000000
75%	5.500000	6500.000000
max	7 000000	8000 000000

```
df.describe(include='all')
```

	emp id	name	salary
count	7.000000	7	7.000000
unique	NaN	6	NaN
top	NaN	SRI HARSHAVARDHANAN R	NaN
freq	NaN	2	NaN
mean	4.000000	NaN	5714.285714
std	2.160247	NaN	1603.567451
min	1.000000	NaN	3000.000000
25%	2.500000	NaN	5000.000000
50%	4.000000	NaN	6000.000000
75%	5.500000	NaN	6500.000000
max	7 000000	NaN	8000 000000

```
empCol=df.columns
```

```
empCol
```

```
Index(['emp id', 'name ', 'salary'], dtype='object') emparray=df.values
```

```
emparray
```

```
array([[1, 'SREE VARSSINI K S', 5000], [2, 'SREEMATHI
 B', 6000],
 [3, 'SREYA G', 7000],
 [4, 'SREYASKARI MULLAPUDI', 5000],
 [5, 'SRI AKASH U G', 8000],
```



```
 [6, 'SRI HARSHAVARDHANAN R', 3000],
 [7, 'SRI HARSHAVARDHANAN R', 6000]], dtype=object)
```

```
employee_DF=pd.DataFrame(emparray,columns=empCol)
```

```
employee_DF
```

	emp id	name	salary

0 2 SREEMATHI B 6000

1 3 SREYA G 7000

2 4 SREYASKARI MULLAPUDI 5000

3 5 SRI AKASH U G 8000

4 6 SRI HARSHAVARDHANAN R 3000

5 7 SRI HARSHAVARDHANAN R 6000

NAME:SANJAY
KISHAN D

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:20.08.2024

```
#sample calculation for low range(lr) , upper range (ur),percentile import numpy as np
array=np.random.randint(1,100,16) # randomly generate 16 numbers between 1 to 100
array
```

```
array([27, 50, 44, 6, 58, 61, 23, 86, 67, 20, 75, 7, 79, 61, 90, 54])
```

```
array.mean() 50.5
```

```
np.percentile(array,25) 26.0
```

```
np.percentile(array,50) 56.0
```

```
np.percentile(array,75) 69.0
```

```
np.percentile(array,100) 90.0
```

```
#outliers detection
def outDetection(array):
    sorted(array) Q1,Q3=np.percentile(array,[25,75])
    IQR=Q3-Q1
    lr=Q1-(1.5*IQR)
    ur=Q3+(1.5*IQR)
    return lr,ur
```

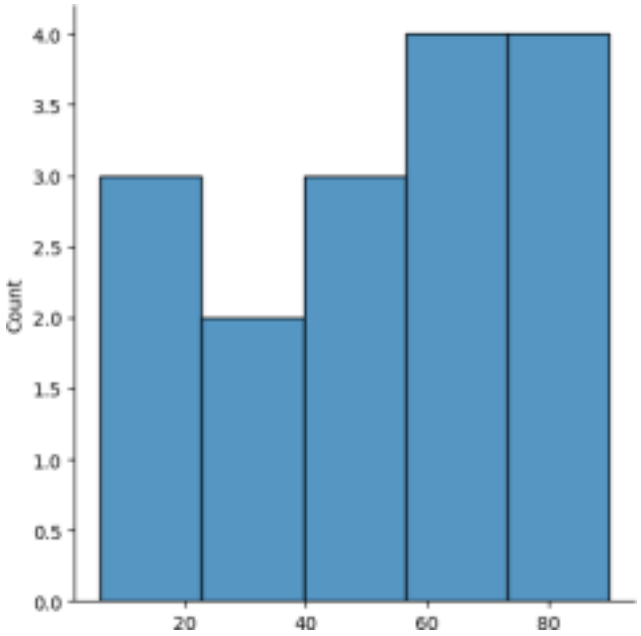
```
lr,ur=outDetection(array)
```

```
lr,ur
```

```
(-38.5, 133.5)
```

```
import seaborn as sns
%matplotlib inline
sns.displot(array)
```

```
<seaborn.axisgrid.FacetGrid at 0x78f3291c2710>
```




```
sns.distplot(array)
```

<ipython-input-19-d72101983c40>:1: UserWarning:

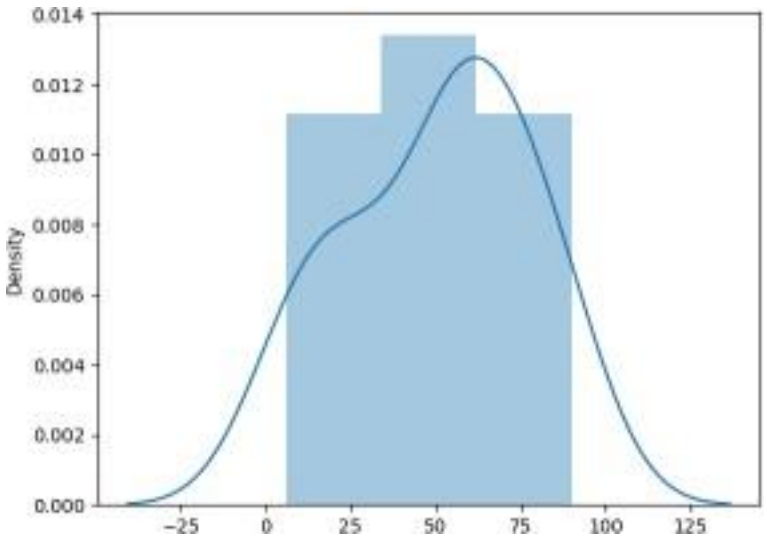
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).

For a guide to updating your code to use the new functions, please see https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

```
sns.distplot(array)
```

<Axes: ylabel='Density'>

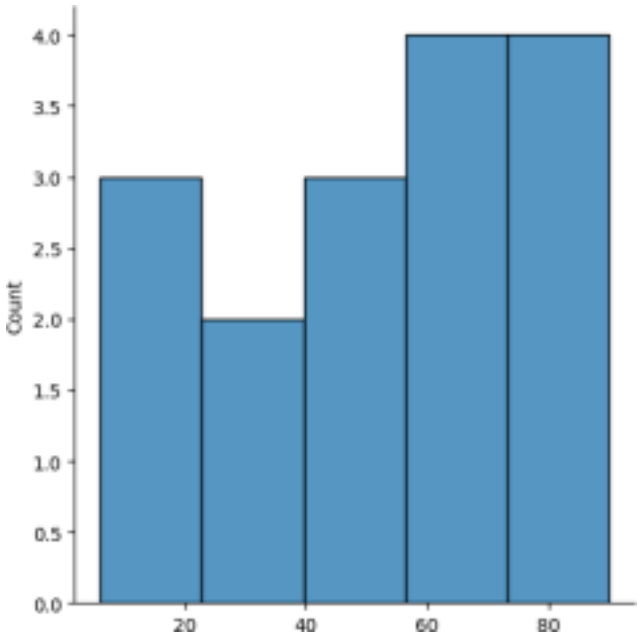



```
new_array=array[(array>lr) & (array<ur)] new_array
```

array([27, 50, 44, 6, 58, 61, 23, 86, 67, 20, 75, 7, 79, 61, 90, 54])

```
sns.displot(new_array)
```

<seaborn.axisgrid.FacetGrid at 0x78f2e09bb580>




```
lr1,ur1=outDetection(new_array) lr1,ur1
```

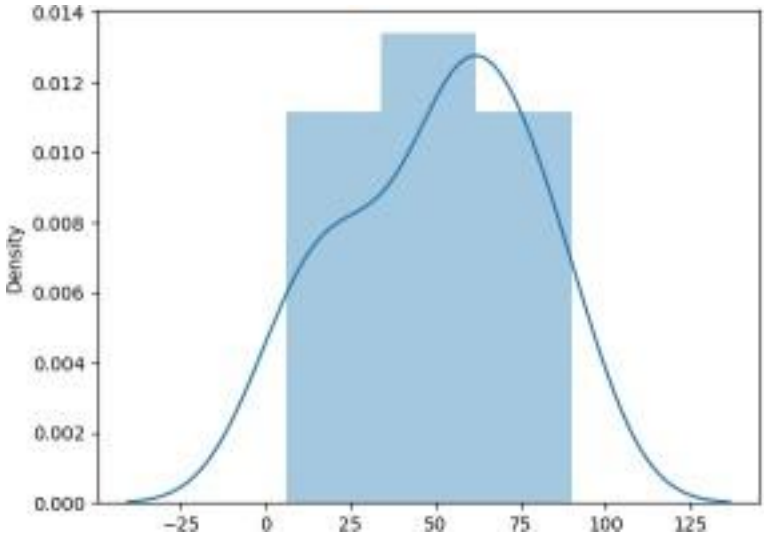
(-38.5, 133.5)

```
final_array=new_array[(new_array>lr1) & (new_array<ur1)] final_array
```

array([27, 50, 44, 6, 58, 61, 23, 86, 67, 20, 75, 7, 79, 61, 90, 54])

```
sns.distplot(final_array)
```

```
<ipython-input-18-7ba96ada5b76>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an
axes-level function for histograms).

For a guide to updating your code to use the new functions, please see https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

 sns.distplot(final_array)
<Axes: ylabel='Density'>
```



NAME:SANJAY
KISHAN D
ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:27.08.2024

Handling Missing and Inappropriate Data in a Dataset

Aim: Demonstrate an experiment to handle missing data and inappropriate data in a Data set using Python Pandas Library for Data Preprocessing.

Dataset Given:

Hotel.csv

CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group
1	20-25	4	Ibis	veg	1300	2	40000	20-25
2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
3	25-30	6	RedFox	Veg	1322	2	30000	25-30
4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
5	35+	3	Ibis	Vegetarian	989	2	45000	35+
6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

About Dataset:

No.of Columns =9 (called as series – CustomerID, Age_Group, Rating(1-5),Hotel, FoodPreference, Bill, NoOfPax, EstimatedSalary)

CutomerID: Numerical Continuous data

Age: Categorical Data

Rating (1-5): Numerical Discrete Data

Hotel: Categorical Data

Food: Categorical Data

Bill: Numerical Continuous data

NoOfPax: Numerical Discrete

EstimatedSalary: Numerical Continuous data

Python Code:

Upload Hotel.csv and convert it into dataFrame

import numpy as np

import pandas as pd

df=pd.read_csv("Hotel_Dataset.csv")

df

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

#From the dataframe identify the duplicate row(i.e row 9)

The duplicated() method returns a Series with True and False values that describe which rows in the DataFrame are duplicated and not.

df.duplicated()

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9     True
10   False
dtype: bool
```

The info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values).

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11 entries, 0 to 10
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   CustomerID        11 non-null     int64
 1   Age_Group         11 non-null     object
 2   Rating(1-5)       11 non-null     int64
 3   Hotel             11 non-null     object
 4   FoodPreference    11 non-null     object
 5   Bill              11 non-null     int64
 6   NoOfPax           11 non-null     int64
 7   EstimatedSalary   11 non-null     int64
 8   Age_Group.1       11 non-null     object
dtypes: int64(5), object(4)
memory usage: 924.0+ bytes
```

The drop_duplicates() method removes duplicate rows.

df.drop_duplicates(inplace=True)

df

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
10	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

#While removing duplicate record row index also removed

The len() function to return the length of an object. With a dataframe, the function returns the number of rows.

len(df)

10

#Reset the index

index=np.array(list(range(0,len(df))))

df.set_index(index,inplace=True)

index

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

df

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary	Age_Group.1
0	1	20-25	4	Ibis	veg	1300	2	40000	20-25
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000	30-35
2	3	25-30	6	RedFox	Veg	1322	2	30000	25-30
3	4	20-25	-1	LemonTree	Veg	1234	2	120000	20-25
4	5	35+	3	Ibis	Vegetarian	989	2	45000	35+
5	6	35+	3	Ibys	Non-Veg	1909	2	122220	35+
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122	35+
7	8	20-25	7	LemonTree	Veg	2999	-10	345673	20-25
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999	25-30
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777	30-35

Axis refers to the dimensions of a DataFrame (index and columns) or Series (index only) Use axis=0 to apply functions row-wise along the index. Use axis=1 to apply functions column-wise across columns.

df.drop(['Age_Group.1'],axis=1,inplace=True)

df

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1	20-25	4	Ibis	veg	1300	2	40000
1	2	30-35	5	LemonTree	Non-Veg	2000	3	59000
2	3	25-30	6	RedFox	Veg	1322	2	30000
3	4	20-25	-1	LemonTree	Veg	1234	2	120000
4	5	35+	3	Ibis	Vegetarian	989	2	45000
5	6	35+	3	Ibys	Non-Veg	1909	2	122220
6	7	35+	4	RedFox	Vegetarian	1000	-1	21122
7	8	20-25	7	LemonTree	Veg	2999	-10	345673
8	9	25-30	2	Ibis	Non-Veg	3456	3	-99999
9	10	30-35	5	RedFox	non-Veg	-6755	4	87777

The function . loc is typically used for label indexing and can access multiple columns.

df.CustomerID.loc[df.CustomerID<0]=np.nan

df.Bill.loc[df.Bill<0]=np.nan

df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan

df

```
C:\Users\SANJAY KISHAN D\AppData\Local\Temp\ipykernel_5300\2580639570.py:1: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame See
the caveats in the documentation: https://pandas.pydata.org/pandas
```

docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
df.CustomerID.loc[df.CustomerID<0]=np.nan
df.Bill.loc[df.Bill<0]=np.nan
C:\Users\SANJAY KISHAN D\AppData\Local\Temp\ipykernel_5300\2580639570.py:2: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4.0	Ibis	veg	1300.0	2	40000.0
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3	59000.0
2	3.0	25-30	NaN	RedFox	Veg	1322.0	2	30000.0
3	4.0	20-25	NaN	LemonTree	Veg	1234.0	2	120000.0
4	5.0	35+	3.0	Ibis	Vegetarian	989.0	2	45000.0
5	6.0	35+	3.0	Ibys	Non-Veg	1909.0	2	122220.0
6	7.0	35+	4.0	RedFox	Vegetarian	1000.0	-1	21122.0
7	8.0	20-25	NaN	LemonTree	Veg	2999.0	-10	345673.0
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3	NaN
9	10.0	30-35	5.0	RedFox	non-Veg	NaN	4	87777.0

df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan

df

```
C:\Users\SANJAY KISHAN D\AppData\Local\Temp\ipykernel_5300\2129877948.py:1: S
ettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

 See the caveats in the documentation: https://pandas.pydata.org/pandas
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan
```

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4.0	Ibis	veg	1300.0	2.0	40000.0
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	NaN	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	NaN	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3.0	Ibis	Vegetarian	989.0	2.0	45000.0
5	6.0	35+	3.0	Ibys	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4.0	RedFox	Vegetarian	1000.0	NaN	21122.0
7	8.0	20-25	NaN	LemonTree	Veg	2999.0	NaN	345673.0
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3.0	NaN
9	10.0	30-35	5.0	RedFox	non-Veg	NaN	4.0	87777.0

df.Age_Group.unique()

```
array(['20-25', '30-35', '25-30', '35+'], dtype=object)
```

df.Hotel.unique()

```
array(['Ibis', 'LemonTree', 'RedFox', 'Ibys'], dtype=object)
```

Using the inplace=True keyword in a pandas method changes the default behaviour such that the operation on the dataframe doesn't return anything, it instead 'modifies the underlying data

df.Hotel.replace(['Ibys'],'Ibis',inplace=True)

df.FoodPreference.unique

```
<bound method Series.unique of 0 veg
1 Non-Veg
2 Veg
3 Veg
4 Vegetarian
5 Non-Veg
6 Vegetarian
7 Veg
8 Non-Veg
9 non-Veg
Name: FoodPreference, dtype: object>
```

df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)

df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)

Fillna is a Pandas function to fill the NA/NaN values with the specified method.

If column or feature is numerical continuous data then replace the missing(NaN) value by taking mean value.

If column or feature is numerical discrete data then replace the missing(NaN) value by taking median value.
If column or feature is non-numerical i.e Categorical data then replace the missing(NaN) value by taking mode value.

df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)

df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)

df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)

df.Bill.fillna(round(df.Bill.mean()),inplace=True)

df

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	EstimatedSalary
0	1.0	20-25	4.0	Ibis	Veg	1300.0	2.0	40000.0
1	2.0	30-35	5.0	LemonTree	Non-Veg	2000.0	3.0	59000.0
2	3.0	25-30	4.0	RedFox	Veg	1322.0	2.0	30000.0
3	4.0	20-25	4.0	LemonTree	Veg	1234.0	2.0	120000.0
4	5.0	35+	3.0	Ibis	Veg	989.0	2.0	45000.0
5	6.0	35+	3.0	Ibis	Non-Veg	1909.0	2.0	122220.0
6	7.0	35+	4.0	RedFox	Veg	1000.0	2.0	21122.0
7	8.0	20-25	4.0	LemonTree	Veg	2999.0	2.0	345673.0
8	9.0	25-30	2.0	Ibis	Non-Veg	3456.0	3.0	96755.0
9	10.0	30-35	5.0	RedFox	Non-Veg	1801.0	4.0	87777.0

NAME: SANJAY
KISHAN D

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:03.09.2024

```
import numpy as np
import pandas as pd
df=pd.read_csv('/content/pre-process_datasample.csv') df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	NaN	50.0	83000.0	No
9	France	37.0	67000.0	Yes

Generate code with df View recommended plots New interactive sheet

Next steps: df.head()

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40	0	NaN Yes

New interactive sheet

Next steps:
Generate code with df View recommended plots

```
df.Country.fillna(df.Country.mode()[0],inplace=True)
features=df.iloc[:,:-1].values
```

```
<ipython-input-5-20665a0bbaa1>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame o The behavior will change in pandas 3.0. This inplace method will never work because the intermediate ob

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inpla
```

```
df.Country.fillna(df.Country.mode()[0],inplace=True)
```

```
label=df.iloc[:,-1].values
```

Start coding or generate with AI.


```
from sklearn.impute import SimpleImputer

age=SimpleImputer(strategy="mean",missing_values=np.nan)

Salary=SimpleImputer(strategy="mean",missing_values=np.nan)

age.fit(features[:,[1]])
```

▾ SimpleImputer i ?

SimpleImputer()

```
Salary.fit(features[:,[2]])
```

▾ SimpleImputer i ?

SimpleImputer()

```
SimpleImputer()
```

▾ SimpleImputer i ?

SimpleImputer()

```
features[:,[1]]=age.transform(features[:,[1]])

features[:,[2]]=Salary.transform(features[:,[2]])

features
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['France', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

```
from sklearn.preprocessing import OneHotEncoder

oh = OneHotEncoder(sparse_output=False)
```

```
Country=oh.fit_transform(features[:,[0]])
```

```
Country
```

```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
```



```
       [1., 0., 0.],
       [1., 0., 0.]])
```

```
final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
```

```
final_set
```

```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [1.0, 0.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

```
from sklearn.preprocessing import StandardScaler sc=StandardScaler()
sc.fit(final_set) feat_standard_scaler=sc.transform(final_set)
```

```
feat_standard_scaler
```

```
array([[ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01, 7.58874362e-
        01, 7.49473254e-01],
       [-1.00000000e+00, -5.00000000e-01, 1.52752523e+00,
       -1.71150388e+00, -1.43817841e+00],
       [-1.00000000e+00, 2.00000000e+00, -6.54653671e-01,
       -1.27555478e+00,  -8.91265492e-01],
       [-1.00000000e+00, -5.00000000e-01, 1.52752523e+00,
       -1.13023841e-01,  -2.53200424e-01],
       [-1.00000000e+00, 2.00000000e+00, -6.54653671e-01,
       1.77608893e-01, 6.63219199e-16],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
       -5.48972942e-01,  -5.26656882e-01],
       [-1.00000000e+00, -5.00000000e-01, 1.52752523e+00, 0.00000000e+00,
       -1.07356980e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01, 1.34013983e+00,
       1.38753832e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01, 1.63077256e+00,
       1.75214693e+00],
       [ 1.00000000e+00, -5.00000000e-01, -6.54653671e-01,
       -2.58340208e-01,  2.93712492e-01]])
```

```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler(feature_range=(0,1)) mms.fit(final_set)
```

```
array([[1. , 0. , 0. , 0.73913043, 0.68571429],
       [0. , 0. , 1. , 0. , 0. ],
       [0. , 1. , 0. , 0.13043478, 0.17142857],
       [0. , 0. , 1. , 0.47826087, 0.37142857],
       [0. , 1. , 0. , 0.56521739, 0.45079365],
       [1. , 0. , 0. , 0.34782609, 0.28571429],
       [0. , 0. , 1. , 0.51207729, 0.11428571],
       [1. , 0. , 0. , 0.91304348, 0.88571429],
       [1. , 0. , 0. , 1. , 1. ],
       [1. , 0. , 0. , 0.43478261, 0.54285714]])
```

AME:SANJAY
KISHAN D

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:10.09.2024

```
import numpy as np import
pandas as pd
df=pd.read_csv("/content/pre-process_datasample.csv")
df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	NaN	50.0	83000.0	No
9	France	37.0	67000.0	Yes

Double-click (or enter) to edit

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
10 entries, 0 to 9
Data columns (total 4 columns): # Column
 Non-Null Count Dtype
-----  -----------  -------------------------  ---------
 0  Country 9 non-null object
 1  Age 9 non-null float64
 2  Salary 9 non-null float64
 3  Purchased 10 non-null object dtypes:
float64(2), object(2) memory usage:
448.0+ bytes
```

```
df.Country.mode()
```

	Country
0	France

```
df.Country.mode()[0]
```

```
type(df.Country.mode())
```

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None, fastpath: bool=False) ->
None
 ~~index is not None, the resulting Series is reindexed with the index values.~~ dtype : str, numpy.dtype, or
ExtensionDtype, optional
  Data type for the output Series. If not specified, this will be inferred from `data`.
  See the :ref:`user guide <basics.dtypes>` for more usages. name : Hashable,
default None
      The name to give to the Series
```

```
df.Country.fillna(df.Country.mode()[0],inplace=True)
```

```
df.Age.fillna(df.Age.median(),inplace=True)
```

```
df.Salary.fillna(round(df.Salary.mean()),inplace=True)
```

```
df
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	63778.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	38.0	52000.0	No
7	France	48.0	79000.0	Yes
8	France	50.0	83000.0	No
9	France	37 0	67000 0	Yes

```
pd.get_dummies(df.Country)
```

	France	Germany	Spain
0	True	False	False
1	False	False	True
2	False	True	False
3	False	False	True
4	False	True	False
5	True	False	False
6	False	False	True
7	True	False	False
8	True	False	False
9	True	False	False

```
updated_dataset=pd.concat([pd.get_dummies(df.Country),df.iloc[:,[1,2,3]]],axis=1)
```

```
updated_dataset
```

	France	Germany	Spain	Age	Salary	Purchased
0	True	False	False	44.0	72000.0	No
1	False	False	True	27.0	48000.0	Yes
2	False	True	False	30.0	54000.0	No
3	False	False	True	38.0	61000.0	No
4	False	True	False	40.0	63778.0	Yes
5	True	False	False	35.0	58000.0	Yes
6	False	False	True	38.0	52000.0	No
7	True	False	False	48.0	79000.0	Yes
8	True	False	False	50.0	83000.0	No
9	True	False	False	37 0	67000 0	Yes

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
10 entries, 0 to 9
Data columns (total 4 columns): # Column
 Non-Null Count Dtype
-----  -----------  -------------------------  ---------
 0  Country 10 non-null object
 1  Age 10 non-null float64
 2  Salary 10 non-null float64
 3  Purchased 10 non-null object dtypes:
float64(2), object(2) memory usage:
448.0+ bytes
```

```
updated_dataset.Purchased.replace(['No','Yes'],[0,1],inplace=True)
```



```
updated_dataset
```

France Germany Spain Age Salary Purchased

0 False True False 30.0 54000.0 0

1 False False True 38.0 61000.0 0

2 False True False 40.0 63778.0 1

3 True False False 35.0 58000.0 1

4 False False True 38.0 52000.0 0

5 True False False 48.0 79000.0 1

6 True False False 50.0 83000.0 0

7 True False False 37 0 67000 0 1

Start coding or generate with AI.

NAME: SANJAY
KISHAN D

ROLL NO:230701043
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:08.10.2024

```
import seaborn as sns import
pandas as pd import numpy as
np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
tips=sns.load_dataset('tips')
```

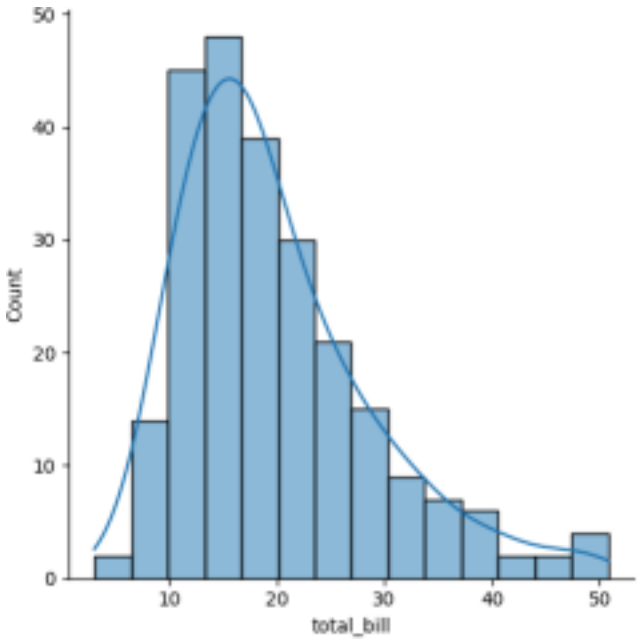
```
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Code Text

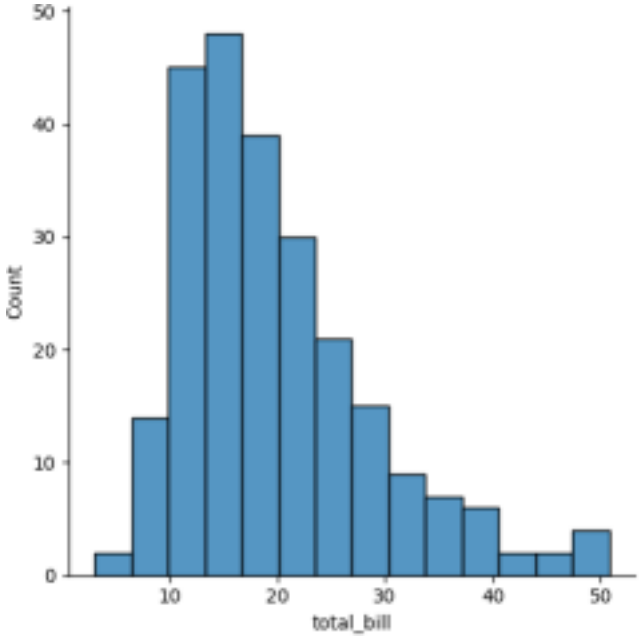
```
sns.displot(tips.total_bill,kde=True)
```

```
<seaborn.axisgrid.FacetGrid at 0x79bb4c7ea680>
```



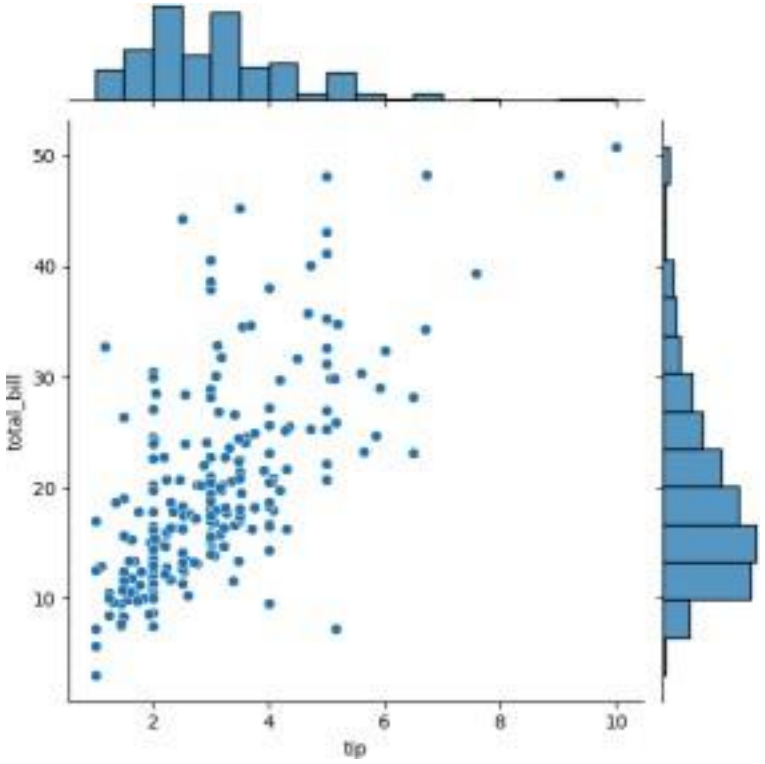

```
sns.displot(tips.total_bill,kde=False)
```

```
<seaborn.axisgrid.FacetGrid at 0x79bb0b0af580>
```




```
sns.jointplot(x=tips.tip,y=tips.total_bill)
```

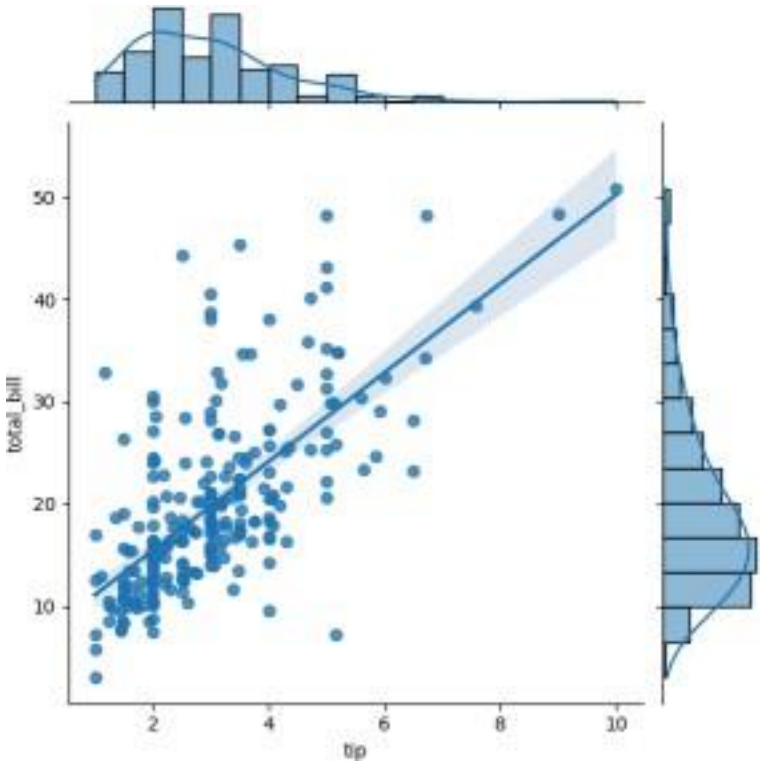
```
<seaborn.axisgrid.JointGrid at 0x79bb08fc96c0>
```




```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg")
```

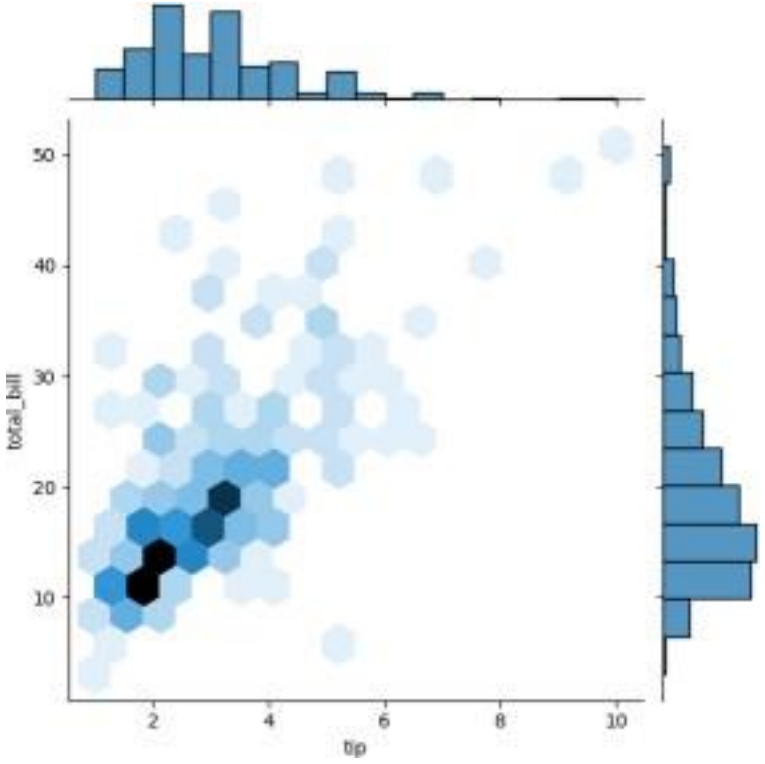


```
<seaborn.axisgrid.JointGrid at 0x79bb08fc9cf0>
```

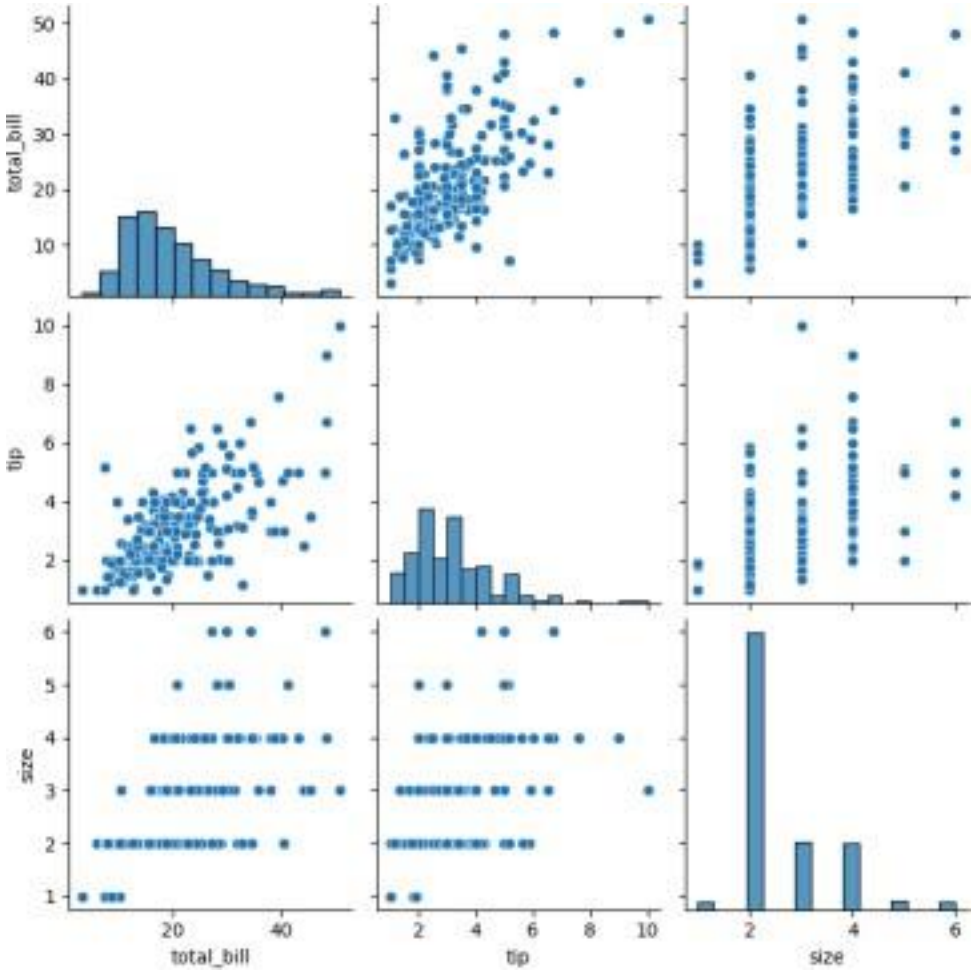



```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
```

```
<seaborn.axisgrid.JointGrid at 0x79bb088f4730>
```




```
sns.pairplot(tips)
```

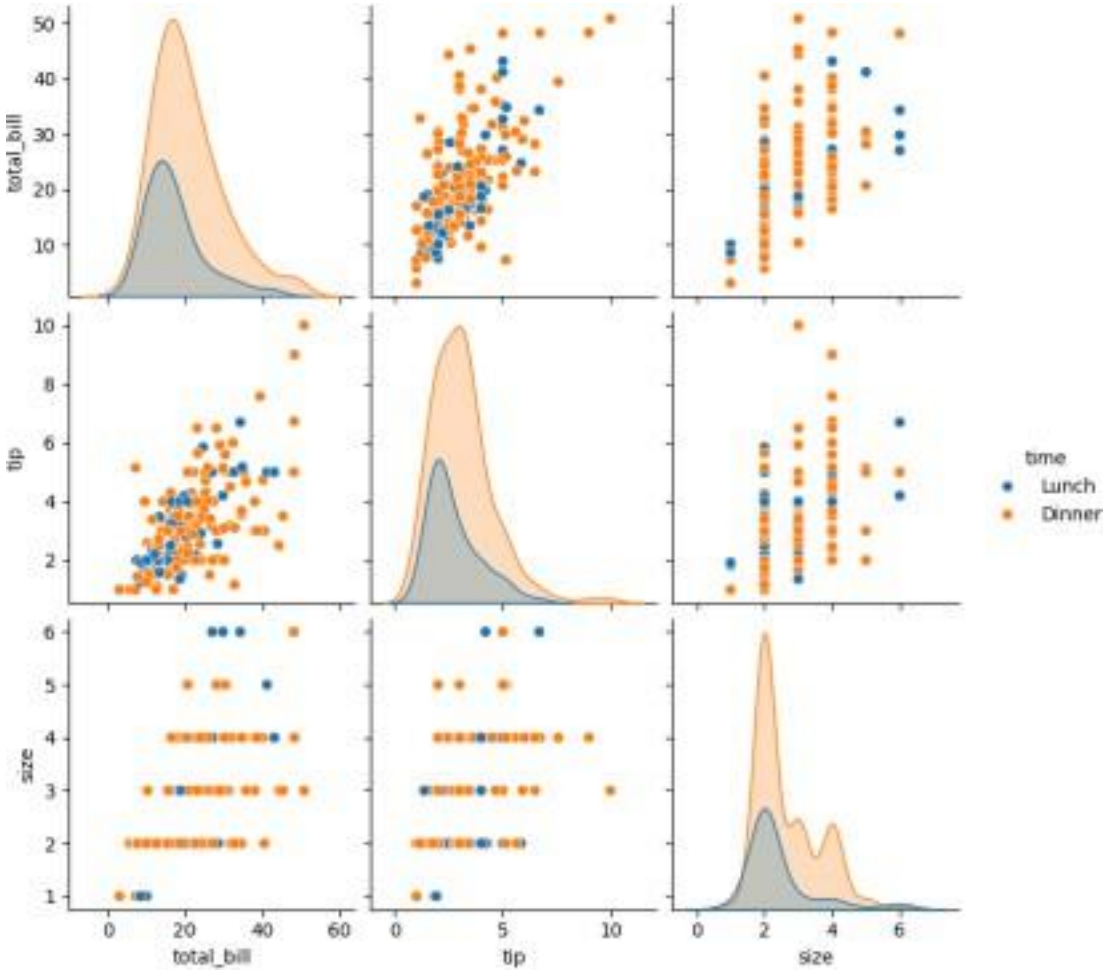



```
tips.time.value_counts()
```

time	count
Dinner	176
Lunch	68

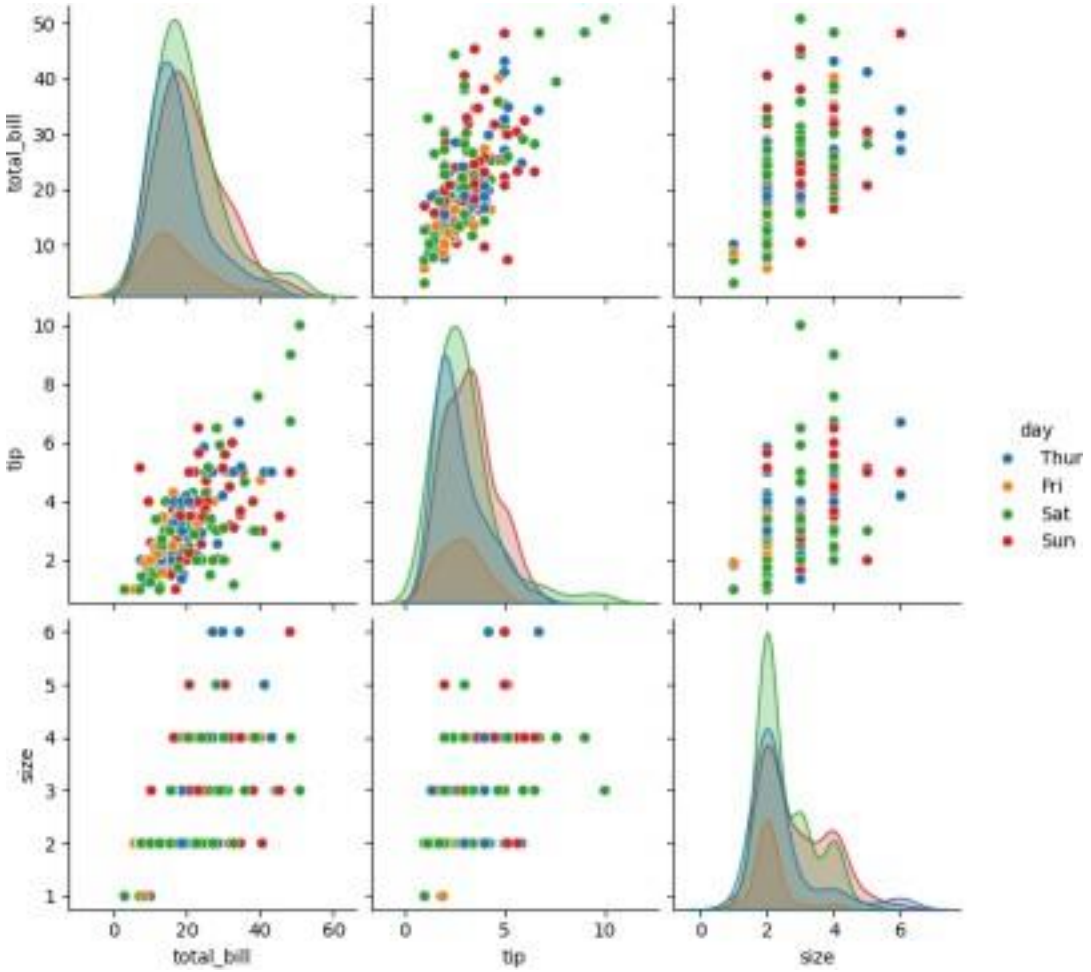
dtype: int64

```
sns.pairplot(tips,hue='time')
```



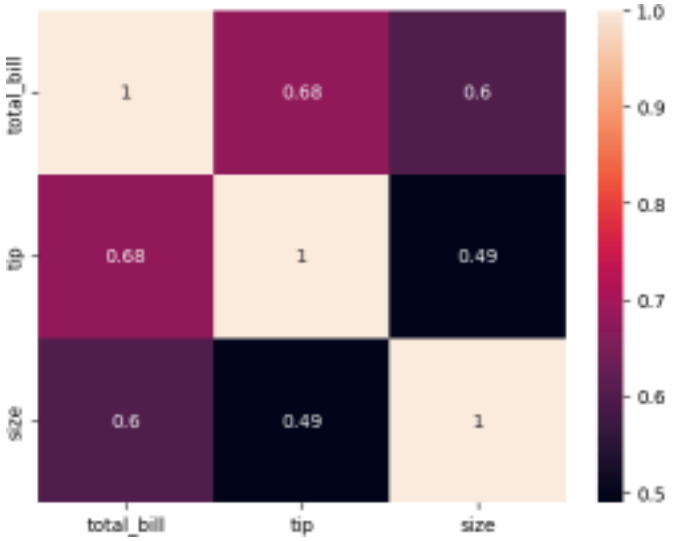
sns.pairplot(tips,hue='day')

<seaborn.axisgrid.PairGrid at 0x79bb08f1f6a0>



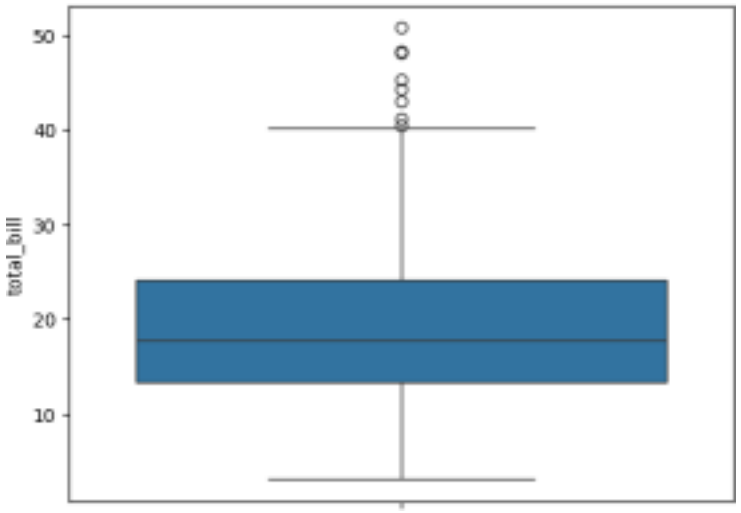
sns.heatmap(tips.corr(numeric_only=True),annot=True)

<Axes: >



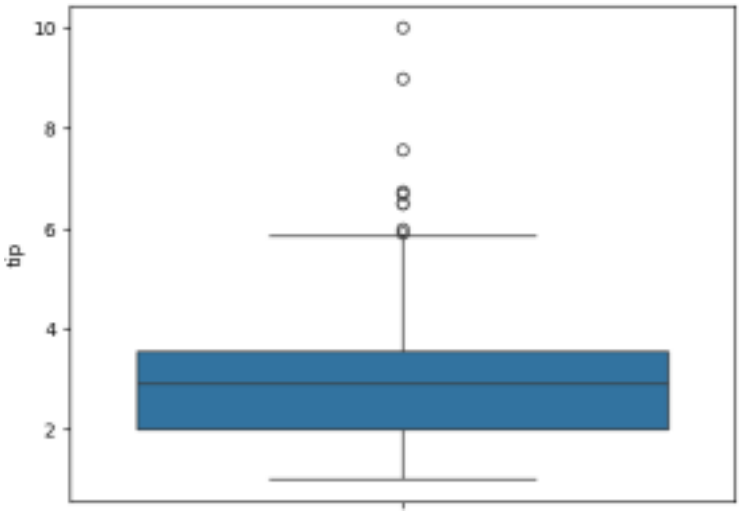
sns.boxplot(tips.total_bill)

<Axes: ylabel='total_bill'>



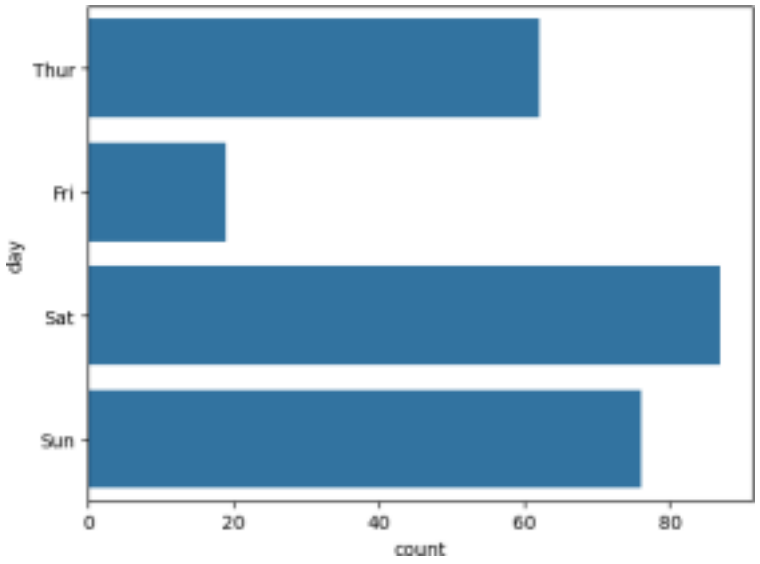
sns.boxplot(tips.tip)

<Axes: ylabel='tip'>



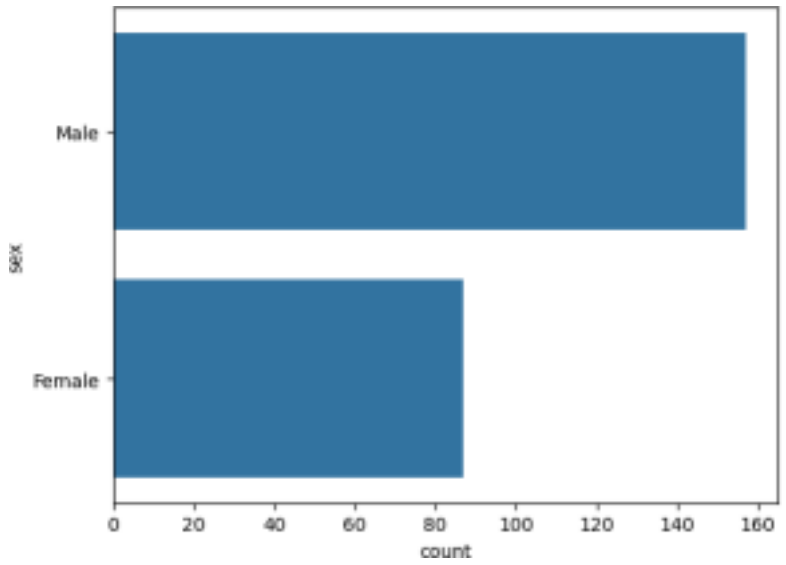
sns.countplot(tips.day)

<Axes: xlabel='count', ylabel='day'>



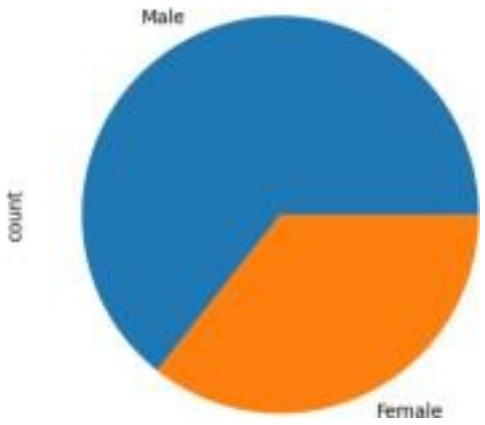
sns.countplot(tips.sex)

<Axes: xlabel='count', ylabel='sex'>



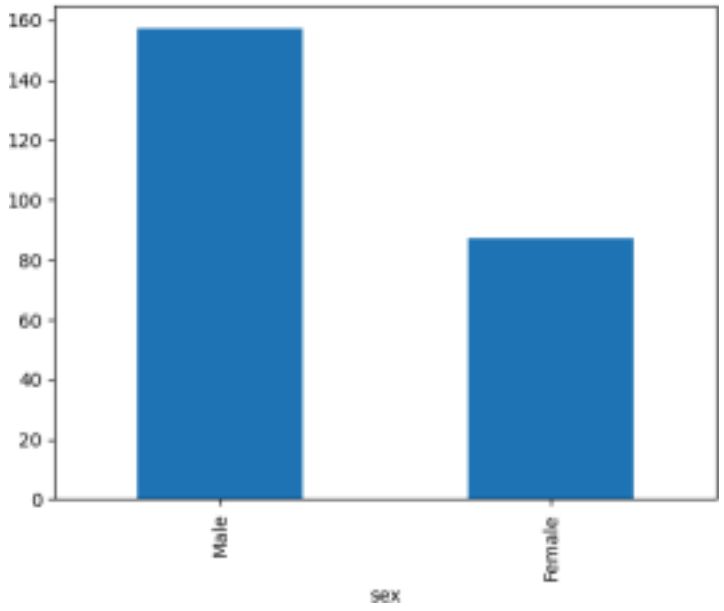

```
tips.sex.value_counts().plot(kind='pie')
```

```
<Axes: ylabel='count'>
```




```
tips.sex.value_counts().plot(kind='bar')
```

```
<Axes: xlabel='sex'>
```




```
sns.countplot(tips[tips.time=='Dinner']['day'])
```

BHISHEK S

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:08.10.2024

In [ ]: In [19]:

```
'pandas.core.frame.DataFram e'>
RangeIndex: 30 entries,
0 to 29
Data columns (total 2 columns):
# Column Non-Null Count Dtype
----------=====-======--===----- 0
YearsExperience 30 non-null
float64 1 Salary 30
non-null int64  dtypes: float64(1),
int64(1) memory usage: 612.0
bytes
```

In [3]: In [4]:

```
df.dropna(inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFram
e'> RangeIndex: 30 entries,
0 to 29
Data columns (total 2 columns):
# Column Non-Null Count Dtype
----------=====-======--===----- 0
YearsExperience 30 non-null
float64 1 Salary 30
non-null int64  dtypes:
float64(1),  int64(1)
memory usage: 612.0 bytes
```

In [5]:

```
import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data
.csv')
df
```

```
df.describe()
```

```
df.info()
```

```
<class
```

Out[5]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [6]: In [7]: In [20]:

```
features=df.iloc[:,[0]].values label=df.iloc[:,[1]].values
```

```
from sklearn.model_selection import
train_test_split x_train,x_test,y_train,y_test=train_test_split(
features,label,test_size=0.2,random_st
```

```
from sklearn.linear_model import
LinearRegression model=LinearRegression()
model.fit(x_train,y_train)
```

Out[20]: ▾ LinearRegression

LinearRegression()

In [21]: model.score(x_train,y_train)

Out[21]: 0.9603182547438908

In [23]: model.score(x_test,y_test)

Out[23]: 0.9184170849214232

In [24]: model.coef_

Out[24]: array([[9281.30847068]])

In [25]: model.intercept_

Out[25]: array([27166.73682891])

In [26]:

In [27]:

In [28]:

```
yr_of_exp=float(input("Enter Years of Experience: "))
yr_of_exp_NP=np.array([[yr_of_exp]])
Salary=model.predict(yr_of_exp_NP)
```

Enter Years of Experience: 44

In [ ]:

In [29]:

```
print("Estimated Salary for {} years of experience is {}: " .format(yr_of_exp,Salary)
```

Estimated Salary for 44.0 years of experience is [[435544.30953887]]:

In [ ]:

```
import pickle
pickle.dump(model,open('SalaryPred.model','wb'))

model=pickle.load(open('SalaryPred.model','rb'))
```

NAME: SANJAY
KISHAN D

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:22.10.2024

In [1]:
```
import numpy as np
import pandas as pd

df=pd.read_csv('Iris.csv'
)
```

In [2]:
```
df.info()
```

```
<class
'pandas.core.frame.DataFr ame'>
RangeIndex: 150
entries, 0 to 149 Data
columns (total 5 columns):
# Column Non-Null Count Dtype
-------  -----  -----  -----  -----  0
sepal.length 150 non-null
float64 1 sepal.width 150
non-null float64 2
petal.length 150 non-null
float64 3 petal.width 150
non-null float64 4
variety 150 non-null object
dtypes:
float64(4), object(1) memory
usage: 6.0+ KB
```

In [3]:
```
df.variety.value_counts()
```

Out[3]:
```
Setosa 50
Versicolor 50
Virginica 50
Name: variety, dtype: int64 df.head(
 )
```

In [4]:

Out[4]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

In [5]:
```
features=df.iloc[:,:-1].values
label=df.iloc[:,4].values
```

In [6]:
```
from sklearn.model_selection import
train_test_split
```

In [8]:
```
from sklearn.neighbors import
KNeighborsClassifier

xtrain,xtest,ytrain,ytest=train_test_split
(features,label,test_size=.2,rando
model_KNN=KNeighborsClassifier(n_neighbors
=5)
model_KNN.fit(xtrain,ytrain)
```

Out[8]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.

On GitHub, the HTML representation is unable to render, please try loading this page

In [9]:
```
print(model_KNN.score(xtrain,ytrain))
print(model_KNN.score(xtest,yt
est))
```

```
0.9583333333333334
1.0
```

In [10]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(label,model_KNN.predict(features))
```

Out[10]:
```
array([[50, 0, 0],
       [ 0, 47, 3],
       [ 0, 2, 48]], dtype=int64)
```

In [11]:
```
from sklearn.metrics import classification_report
print(classification_report(label,mo del_KNN.predict(features)))
```

```
  precision recall f1-score support

  Setosa 1.00 1.00 1.00 50 Versicolor 0.96 0.94 0.95 50   Virginica 0.94 0.96 0.95 50

  accuracy 0.97 150   macro avg 0.97 0.97 0.97 150 weighted avg 0.97 0.97 0.97 150
```

BHISHEK S

ROLL NO:230701287

SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE

DATE:29.10.2024

```
import pandas as pd
df=pd.read_csv('Social_N etwork_Ads.csv') df
```

In [1]:

```
import numpy as np
```

Out[1]: **User ID Gender Age EstimatedSalary Purchased** **0** 15624510

Male 19 19000 0 **1** 15810944 Male 35 20000 0 **2** 15668575

Female 26 43000 0 **3** 15603246 Female 27 57000 0 **4** 15804002

Male 19 76000 0 **...** ... ...

395 15691863 Female 46 41000 1 **396** 15706071 Male 51 23000

1 **397** 15654296 Female 50 20000 1 **398** 15755018 Male 36

33000 0 **399** 15594041 Female 49 36000 1

400 rows × 5 columns

In [2]:

```
df.head(
)
```

Out[2]: **User ID Gender Age EstimatedSalary Purchased**

0 15624510 Male 19 19000 0

1 15810944 Male 35 20000 0

2 15668575 Female 26 43000 0

3 15603246 Female 27 57000 0

4 15804002 Male 19 76000 0

```
                                2,3]].values
In [4]:                         label=df.iloc[:,4].v
features=df.iloc[:,[alues features


Out[4]: array([[ 19, 19000],   [
                35, 20000],
                [   26,    43000],
                [   27,    57000],
                [   19,    76000],
                [   27,    58000],
                [   27,    84000],
                [   32,    150000],
                [   25,    33000],
                [   35,    65000],
                [   26,    80000],
                [   26,    52000],
                [   20,    86000],
                [   32,    18000],
                [   18,    82000],
                [   29,    80000],
                [   47,    25000],
                [   45,    26000],
                [   46,    28000],
                      [ 48 29000]
          label
In
[5]:

                                                Out[5]: array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1,  1,  1,  1,  1,  1,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,                0, 0, 0, 0, 0,  0,
       0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,                0, 1, 0, 0,  0, 0,
       0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1, 0, 0,        0,  0,  0,
       0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0, 0,  0,       0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 0,  0,       0,  0,  0,
       1,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0, 0, 0,    0,  0,  1,

       1,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0, 0, 0, 0,        0,  0,  0,  0,  0,  0,  0,  1,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 0,  0,  0,      0,  0,  0,  1,  0,  1,  0,
       1,  0,  1,  0,  1,  1,  0,  0,  0,  1,  0,  0,  0, 1,  0,  1,      1,  1,  0,  0,  1,  1,  0,
       1,  1,  0,  1,  1,  0,  1,  0,  0,  0,  1,  1,  0,     1,  1,  0,  1,  0,  1,  0,  1,  0,  0,
       1,  1,  0,  1,  0,  0,  1,  1,  0,  1,     1, 0,  1,   1,  0,  0,  1,  0,  0,  1,  1,  1,  1,
       1,  0,  1,  1,  1,  1,  0,  1,  1,  0,     1,  0, 1,   0,  1,  1,  1,  1,  0,  0,  0,  1,  1,
       0,  1,  1,  1,  1,  1,  0,  0,  0,  1,              1,  0,  0,  1,  0,  1,  0,  1,  1,  0,  1,  0,  1,
       1,  0,  1,  1,  0,  0,  0,        1, 1,  0,  1,  0,  0,  1,  0,  1,  0,  0,  1,  1,  0,  0,  1,
       1,  0,  1,  1,  0,  0,  1,        0,  1, 0,  1,  1,  1,  0,  1,  0,  1,  1,  1,  0,  1,  1,  1,
       1,  0,  1,  1,  1,  0,  1,        0,  1, 0,  0,  1,  1,  0,  1,  1,  1,  1,  1,  1,  0,  1,  1,
       1, 1, 1, 1, 0, 1,  1, 1, 0, 1], dtype=int64)

                                        import train_test_split from
                                        sklearn.linear_model import
In [6]:                                 LogisticRegression
from sklearn.model_selection
```

In [7]:
```
for i in range(1,401):
x_train,x_test,y_train,y_test=train_test_split(features,label,test_size=0.
finalModel=LogisticRegression()
finalModel.fit(x_train,y_train)
train_score=finalModel.score(x_train,y_train)
test_score=finalModel.score(x_test,y_test)
if test_score>train_score:
print("Test {} Train{} Random State {}".format(test_score,train_score,i)
```

```
Test 0.6875 Train0.63125 Random State 3
Test 0.7375 Train0.61875 Random State 4
Test 0.6625 Train0.6375 Random State 5
Test 0.65 Train0.640625 Random State 6
Test 0.675 Train0.634375 Random State 7
Test 0.675 Train0.634375 Random State 8
Test 0.65 Train0.640625 Random State 10
Test 0.6625 Train0.6375 Random State 11
Test 0.7125 Train0.625 Random State 13
Test 0.675 Train0.634375 Random State 16
Test 0.7 Train0.628125 Random State 17
Test 0.7 Train0.628125 Random State 21
Test 0.65 Train0.640625 Random State 24
Test 0.6625 Train0.6375 Random State 25
Test 0.75 Train0.615625 Random State 26
Test 0.675 Train0.634375 Random State 27
Test 0.7 Train0.628125 Random State 28
Test 0.6875 Train0.63125 Random State 29
Test 0.6875 Train0.63125 Random State 31
T t 0 6625 T i 0 6375 R d St t 37
```

In [8]:
```
x_train,x_test,y_train,y_test=train_test_splitfinalModel=LogisticRegression()
finalModel.fit(x_train,y_train)
```

Out[8]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:
```
print(finalModel.score(x_train,y_train))
print(finalModel.score(x_test,y_test))
```

```
0.834375
0.9125
```

In [10]:
```
from sklearn.metrics import classification_report
print(classification_report(label,finalModel.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.85      0.93      0.89       257
           1       0.84      0.71      0.77       143

    accuracy                           0.85       400
   macro avg       0.85      0.82      0.83       400
weighted avg       0.85      0.85      0.85       400
```

NAME: SANJAY KISHAN D

ROLL NO:230701287
SUBJECT NAME:CS23332-FUNDAMENTALS OF DATA SCIENCE
DATE:05.11.2024

In [1]:
```
import seaborn as sns
%matplotlib inline
```

In [2]:
```
df=pd.read_csv('Mall_Customer s.csv')
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
 0 CustomerID 200 non-null int64
 1 Gender 200 non-null object
 2 Age 200 non-null int64
 3 Annual Income (k$) 200 non-null int64
 4 Spending Score (1-100) 200 non-null int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

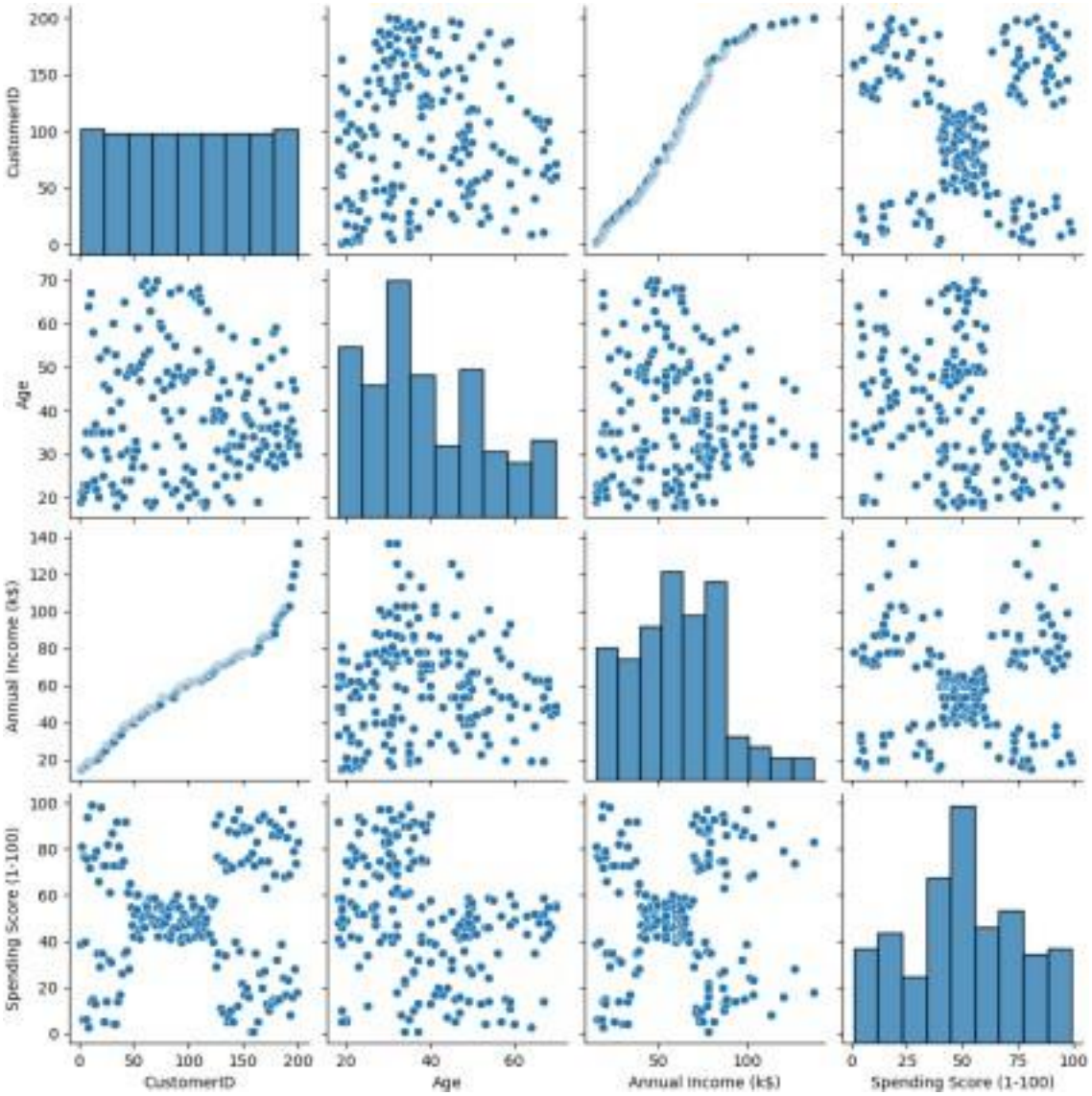
```
df.head()
```

Out[4]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [5]:
```
sns.pairplot(df)
```

Out[5]: <seaborn.axisgrid.PairGrid at 0x170e8e47850>




```
features=df.iloc[:,[3,4]].values
```

In [6]:

In [7]:

```
from sklearn.cluster import KMeans
model=KMeans(n_clusters=5)
```

```
model.fit(features)
KMeans(n_clusters=5)

C:\Users\SANJAY KISHAN D\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust
er\_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
warnings.warn(
C:\Users\SANJAY KISHAN D\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust
er\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by
setting the environment variable OMP_NUM_THREADS=1. warnings.warn(
```

Out[7]:
```
KMeans(n_clusters=5)
```

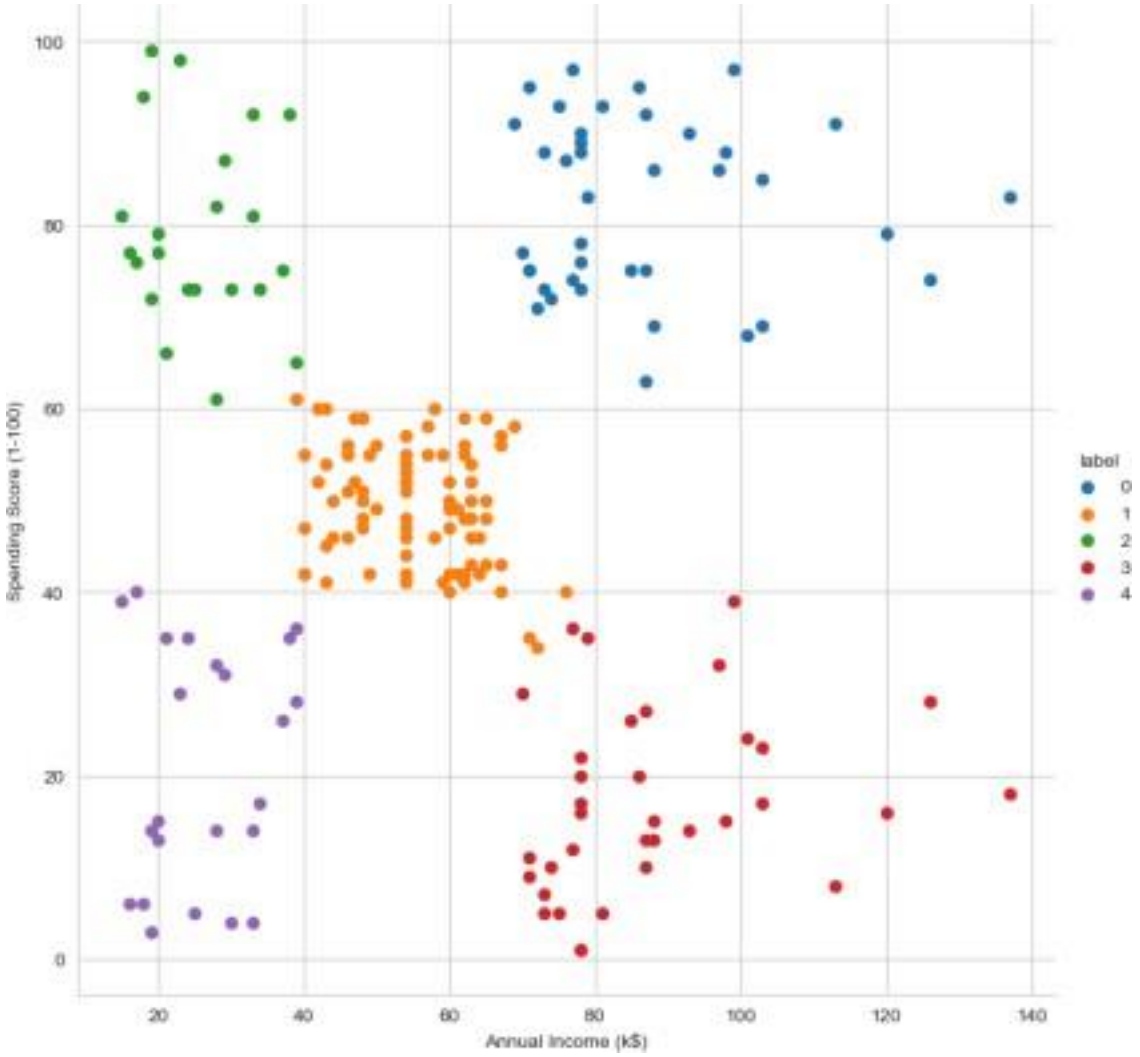
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [8]:
```
Final=df.iloc[:,[3,4]]
Final['label']=model.predict(features) Final.head()

C:\Users\SANJAY KISHAN D\AppData\Local\Temp\ipykernel_8116\470183701.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a view-versus-a-copy)
Final['label']=model.predict(features)
```

Out[8]:

	Annual Income (k$)	Spending Score (1-100)	label
0	15	39	4
1	15	81	2
2	16	6	4
3	16	77	2
4	17	40	4

In [9]:
```
sns.set_style("whitegrid")
sns.FacetGrid(Final,hue="label",height=8) \
.map(plt.scatter,"Annual Income (k$)", "Spending Score (1-100)") \
.add_legend();
plt.show()
```



In [10]:
```
features_el=df.iloc[:,[2,3,4]].values
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,10):
  model=KMeans(n_clusters=i)
  model.fit(features_el)
  wcss.append(model.inertia_)
plt.plot(range(1,10),wcss)
```

C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust

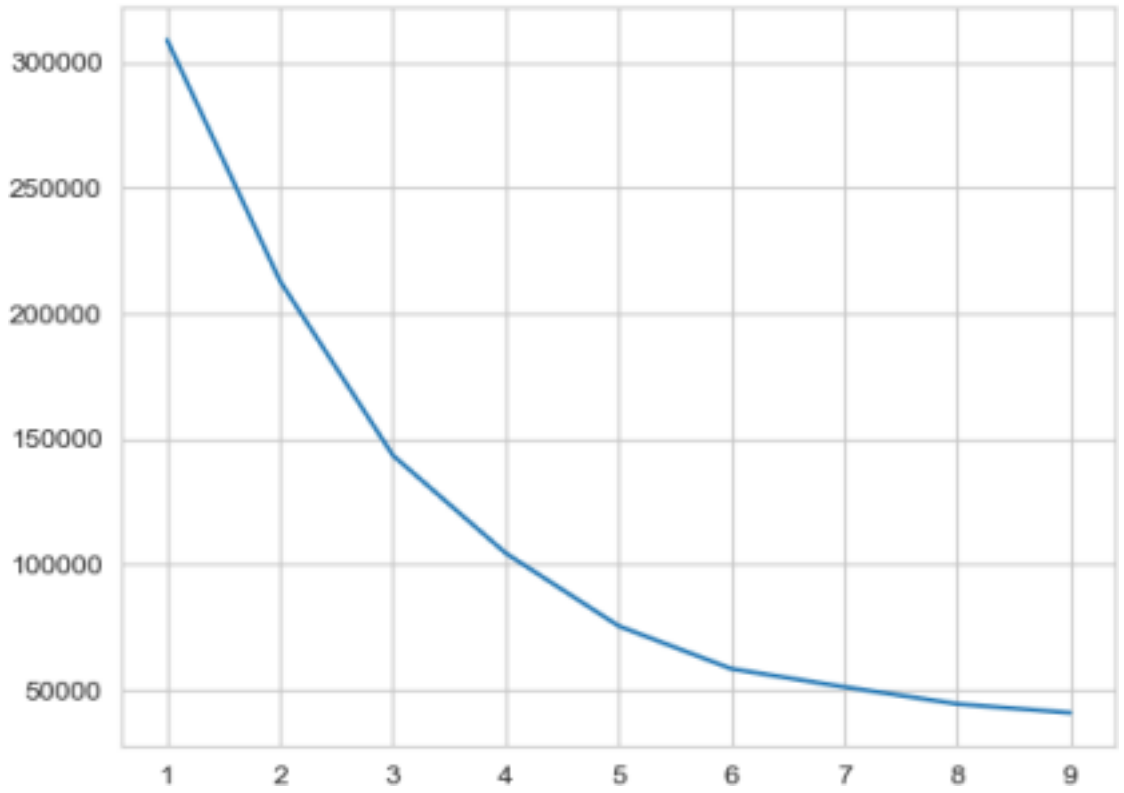
er_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
  warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on

Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust er_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
warnings.warn(
C:\Users\SANJAYKISHA D\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust

er_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust

```
er\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust
er\_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
  warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust
er\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust
er\_kmeans.py:870: FutureWarning: The default value of `n_init` will chang e from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre ss the warning
  warnings.warn(
C:\Users\SANJAYKISHAND\AppData\Local\anaconda3\Lib\site-packages\sklearn\clust
er\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You c an avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[10]: [<matplotlib.lines.Line2D at 0x170e99f3550>]



In [ ]: