FUNDAMENTAL OF DATA SCIENCE

Reg No: 230701024

Name: Akshaya sri S

Dept: CSE _ A (II nd yr)

Exp-1_Matplotseaborn

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

data=pd.read_csv('Iris.csv')

data
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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

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   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

data.value_counts('Species')

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

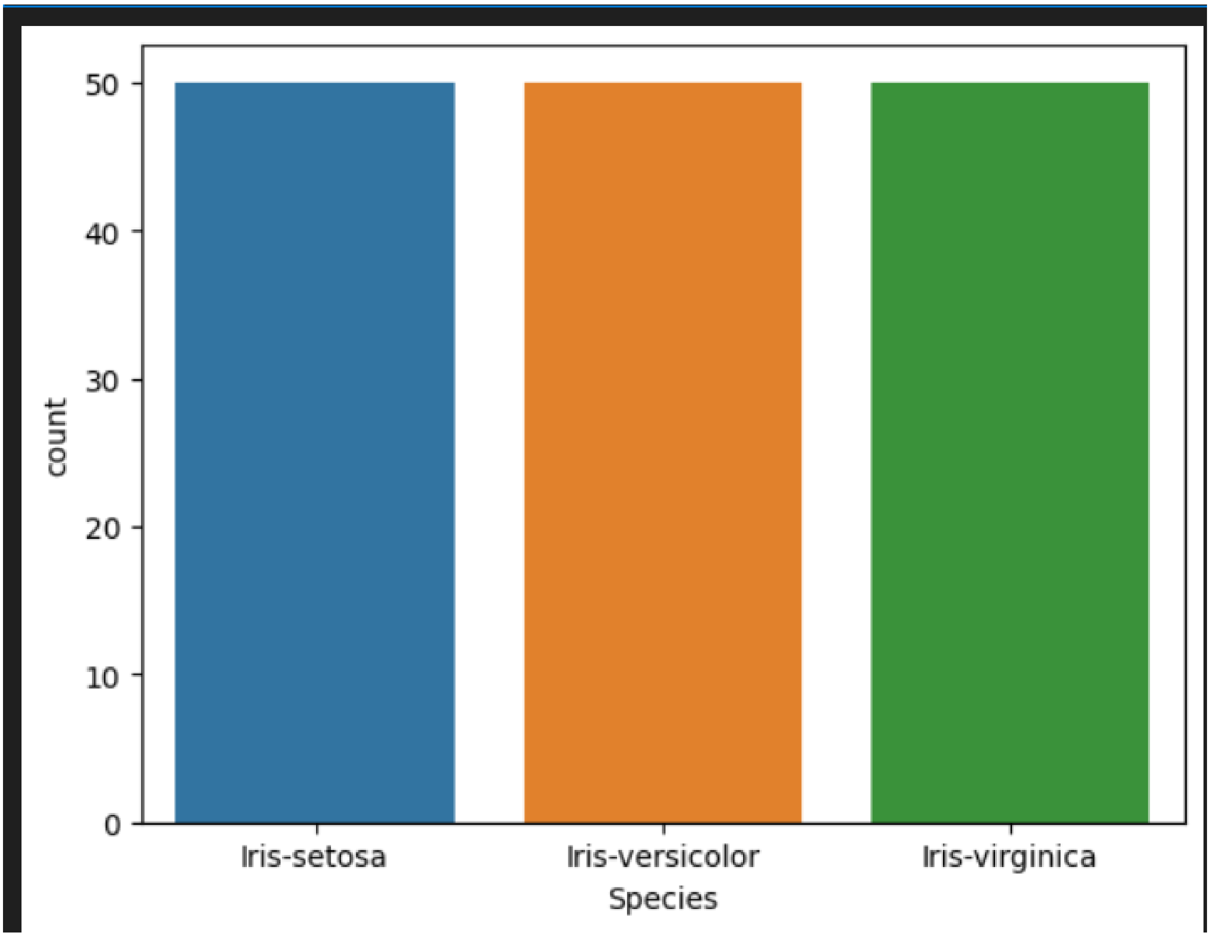
sns.countplot(x='Species',data=data,)

plt.show()

dummies=pd.get_dummies(data.Species)

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

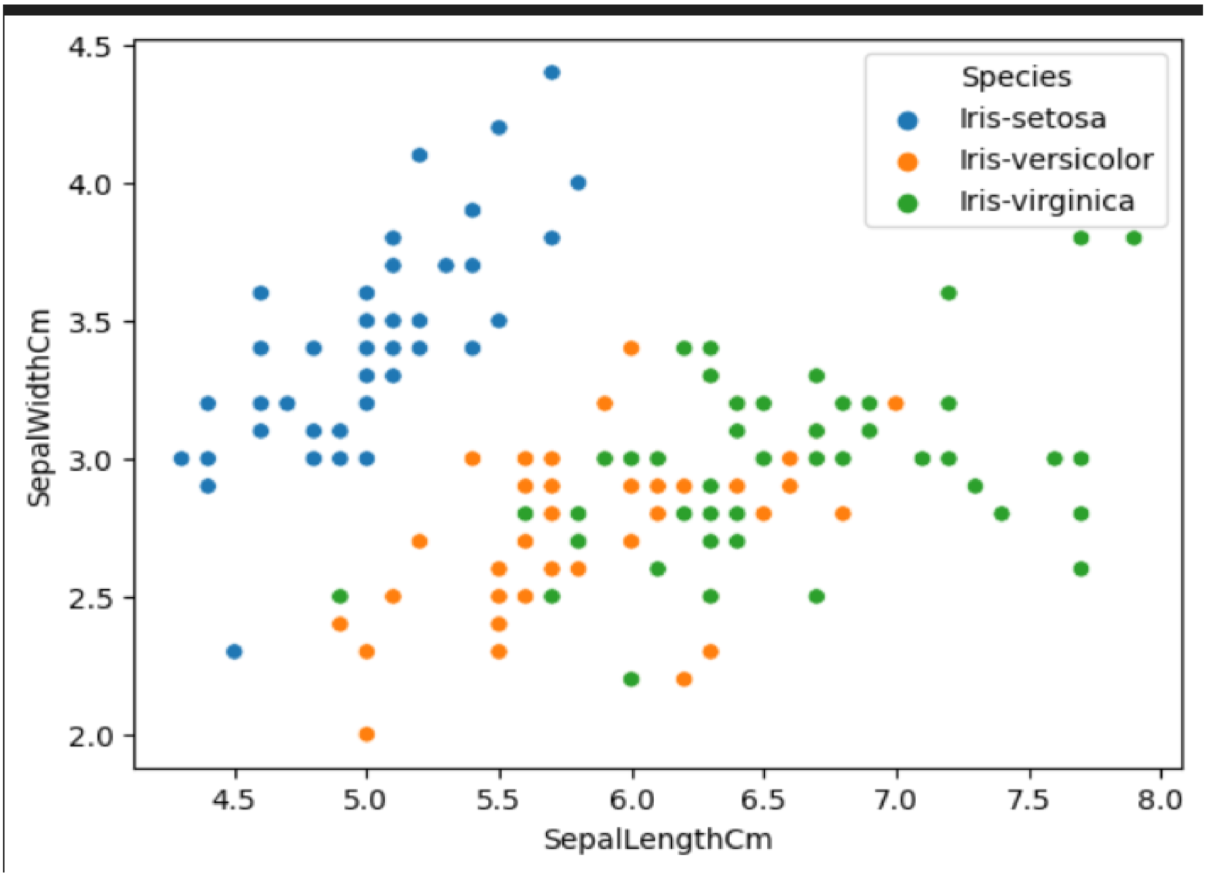
FinalDataset.head()



	Iris-setosa	Iris-versicolor	Iris-virginica	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm
0	1	0	0	1	5.1	3.5	1.4
1	1	0	0	2	4.9	3.0	1.4
2	1	0	0	3	4.7	3.2	1.3
3	1	0	0	4	4.6	3.1	1.5
4	1	0	0	5	5.0	3.6	1.4

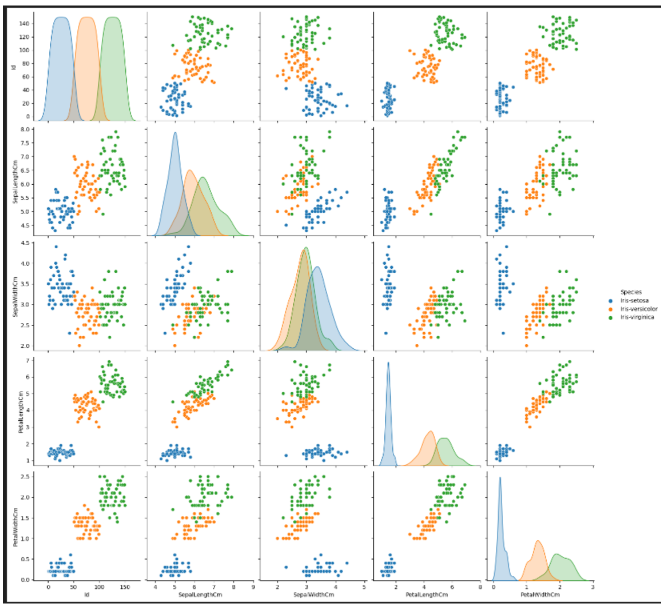
```
sns.scatterplot(x='PetalLengthCm',y='PetalWidthCm',hue='Species',data=data)
```

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

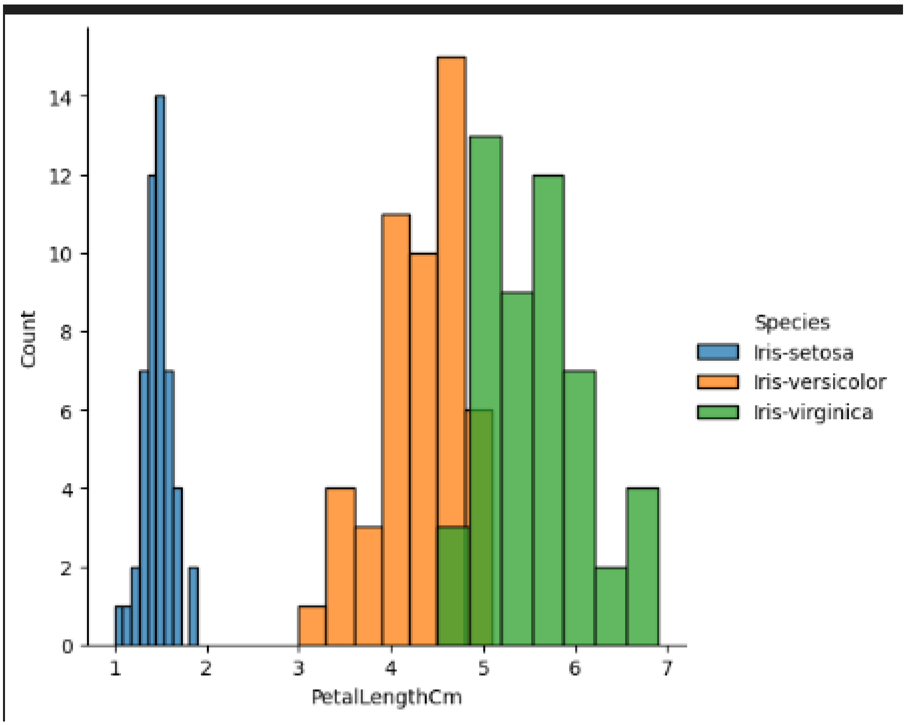



```
sns.pairplot(data,hue='Species',height=3);
```

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

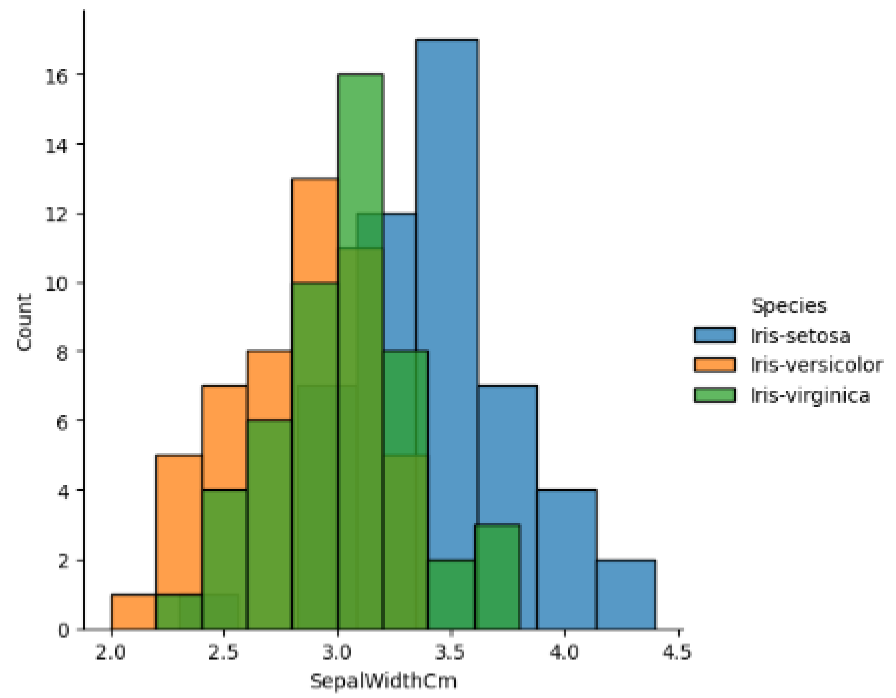


```
sns.FacetGrid(data,hue='Species',height=5).map(sns.histplot,'PetalLengthCm').add_legend(); plt.sho
```




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

plt.show();
```



Exp-2_Numpy buildin functions

import numpy as np

array=np.random.randint(1,100,9)

array

```
array([69, 82, 26, 46, 64, 44, 96, 48, 81])
```

np.sqrt(array)

```
array([8.30662386, 9.05538514, 5.09901951, 6.78232998, 8.
       6.63324958, 9.79795897, 6.92820323, 9.        ])
```

array.ndim

```
1
```

new_array=array.reshape(3,3)

new_array

```
array([[69, 82, 26],
       [46, 64, 44],
       [96, 48, 81]])
```

new_array.ravel()

```
array([69, 82, 26, 46, 64, 44, 96, 48, 81])
```

newm=new_array.reshape(3,3)

newm

```
array([[69, 82, 26],
       [46, 64, 44],
       [96, 48, 81]])
```

newm[2,1:3]

newm[1:2,1:3]
new_array[0:2,0:1]
new_array[1:3]

```
array([48, 81])
```

```
array([[64, 44]])
```

```
array([[69],
       [46]])
```

```
array([[46, 64, 44],
       [96, 48, 81]])
```

Exp-3_Pandas

	0	1	2
0	1	Smith	50000
1	2	Jones	60000

	Empd	Name	Salary
0	1	Smith	50000
1	2	Jones	60000

```
<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  <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
```

	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

	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

	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

	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

```
<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
```

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

```
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

```
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],
```

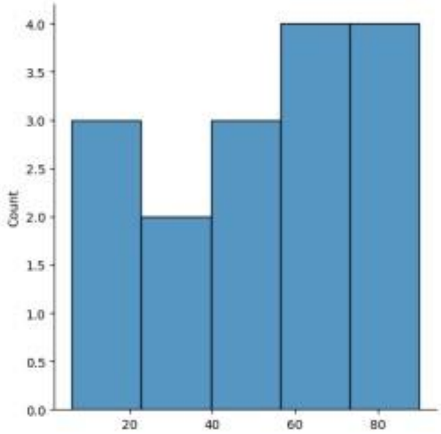


```
[6, 'SRI HARSHAVARDHANAN R', 3000],
[7, 'SRI HARSHAVARDHANAN R', 6000]], dtype=object)
```

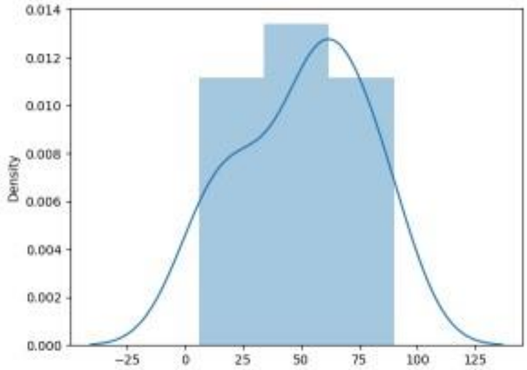
	emp id	name	salary
0	1	SREE VARSSINI K S	5000
1	2	SREEMATHI B	6000
2	3	SREYA G	7000
3	4	SREYASKARI MULLAPUDI	5000
4	5	SRI AKASH U G	8000
5	6	SRI HARSHAVARDHANAN R	3000
6	7	SRI HARSHAVARDHANAN R	6000

EXP-4_Outlier Detection

<seaborn.axisgrid.FacetGrid a0x78f3291c2710>



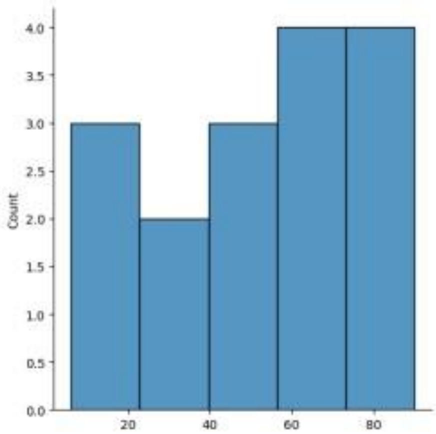
<Axes: ylabel='Density'>



new_array

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

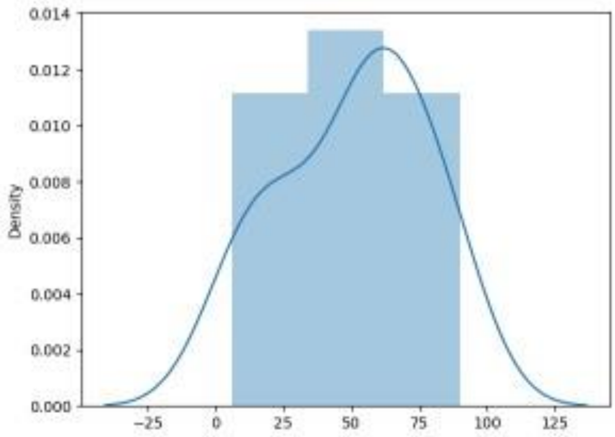
<seaborn.axisgrid.FacetGrid at 0x78f2e09bb580>



lr1,ur1

(-38.5, 133.5) final_array array([27, 50, 44, 6, 58, 61, 23, 86, 67, 20, 75, 7, 79, 61, 90, 54])

<Axes: ylabel='Density'>



Exp-5_Missing and inappropriate data

Dataset

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

	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

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

```
<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
```

	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

	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

	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

	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

	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

	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

Exp-6_Feature Scaling

```
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

Next steps: `df.head()`

Generate code with `df` View recommended plots New interactive sheet

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

Next steps:

Generate code with `df` View recommended plots New interactive sheet

```
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
SimpleImputer()
```

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

```
▾ SimpleImputer
SimpleImputer()
```

```
SimpleImputer()
```

```
▾ SimpleImputer
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)
feat_minmax_scaler=mms.transform(final_set)
feat_minmax_scaler
```

```
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]])
```

Exp-7_Data Preprocessing

```
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	True	False	False	44.0	72000.0	0
1	False	False	True	27.0	48000.0	1
2	False	True	False	30.0	54000.0	0
3	False	False	True	38.0	61000.0	0
4	False	True	False	40.0	63778.0	1
5	True	False	False	35.0	58000.0	1
6	False	False	True	38.0	52000.0	0
7	True	False	False	48.0	79000.0	1
8	True	False	False	50.0	83000.0	0
9	True	False	False	37 0	67000 0	1

Exp-8_EDA -

```
import seaborn as sns  import
pandas as pd  import numpy as
```

```
np  import matplotlib.pyplot as
plt  %matplotlib inline
```

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

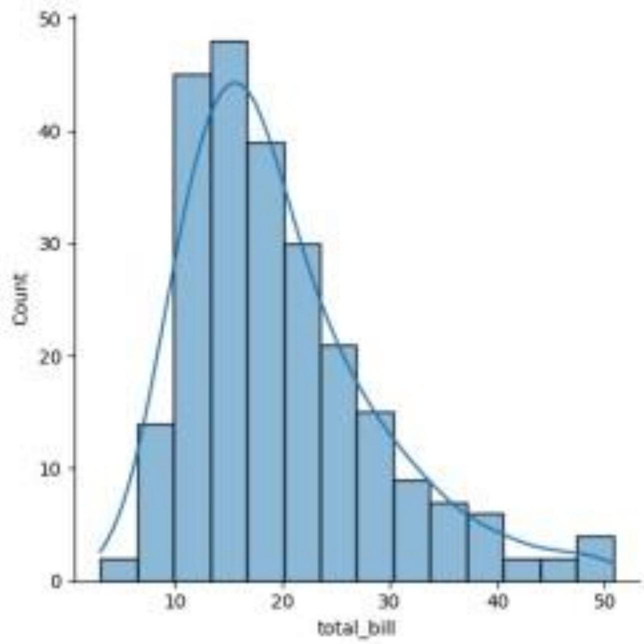
```
tips.head()
```

	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

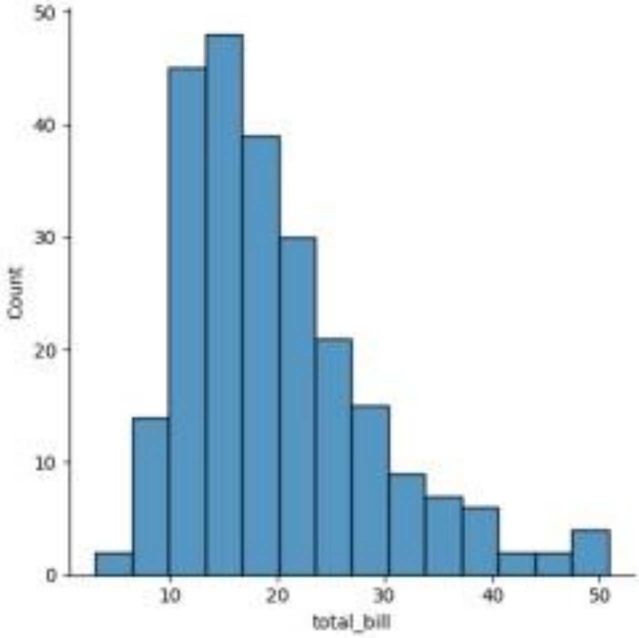
```
sns.displot(tips.total_bill,kde=True)
```

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



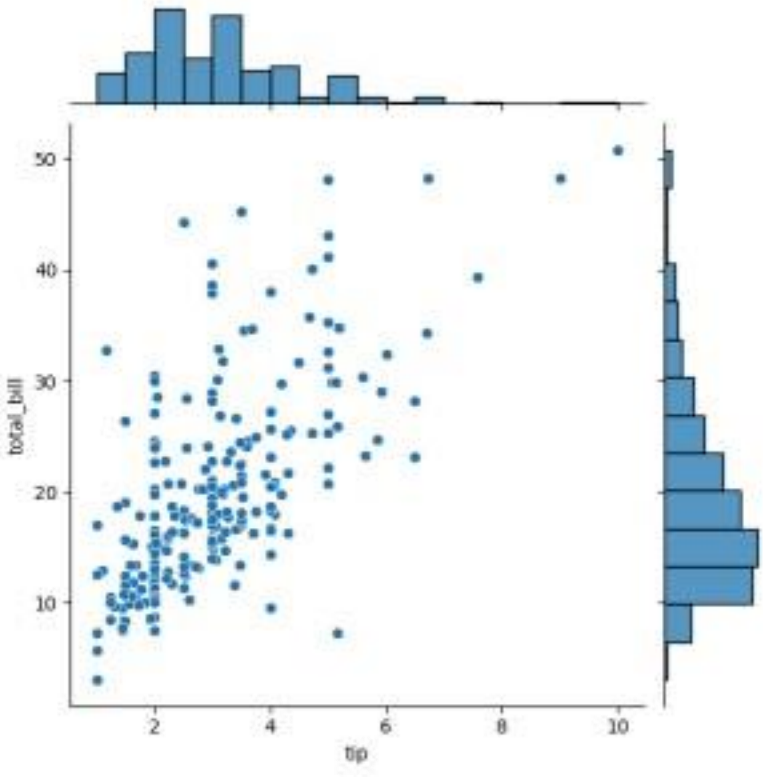

```
sns.displot(tips.total_bill,kde=False)
```

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

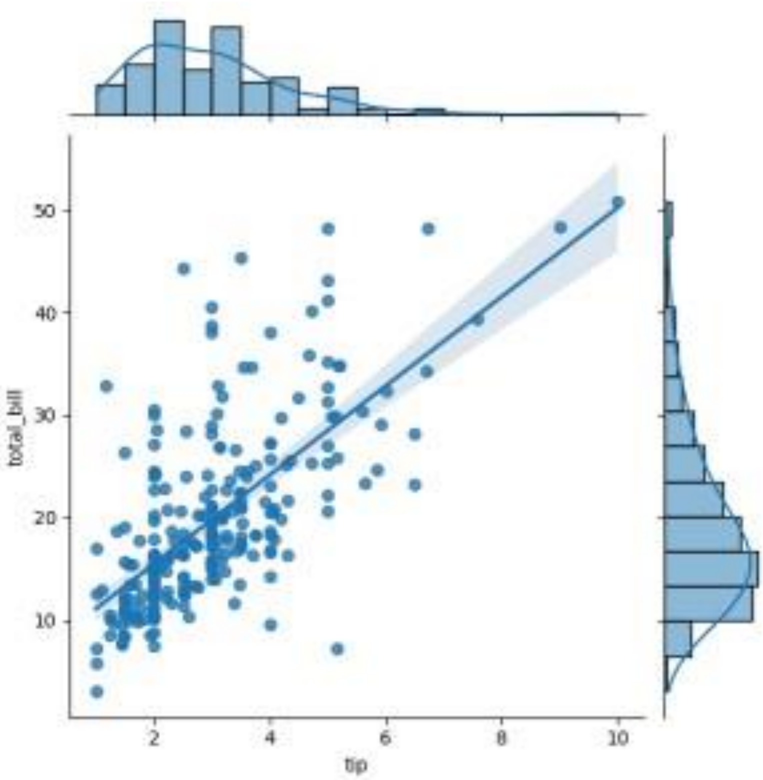



```
sns.jointplot(x=tips.tip,y=tips.total_bill)
```

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

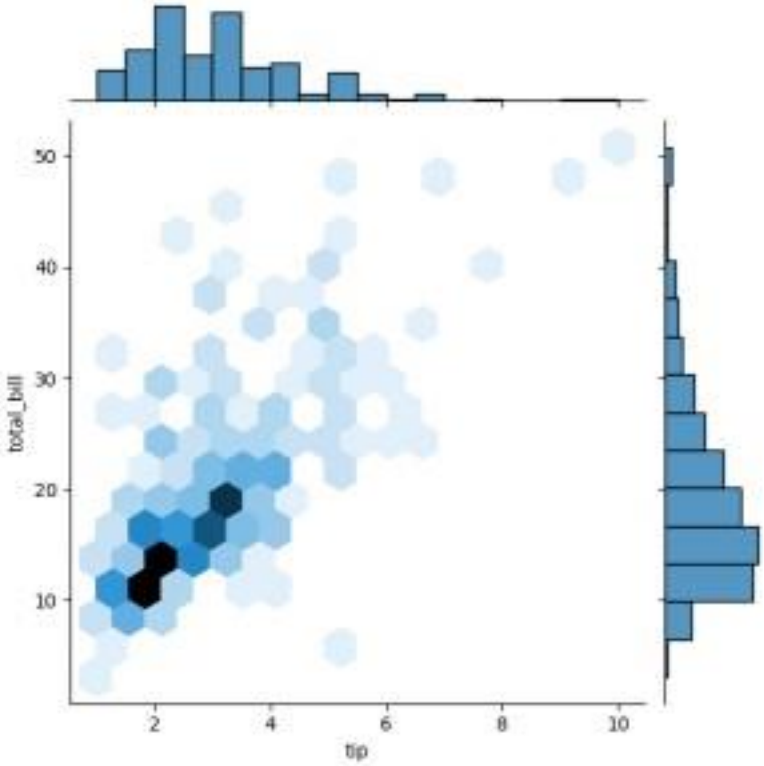



```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="reg"
)  <seaborn.axisgrid.JointGrid at 0x79bb08fc9cf0>
```

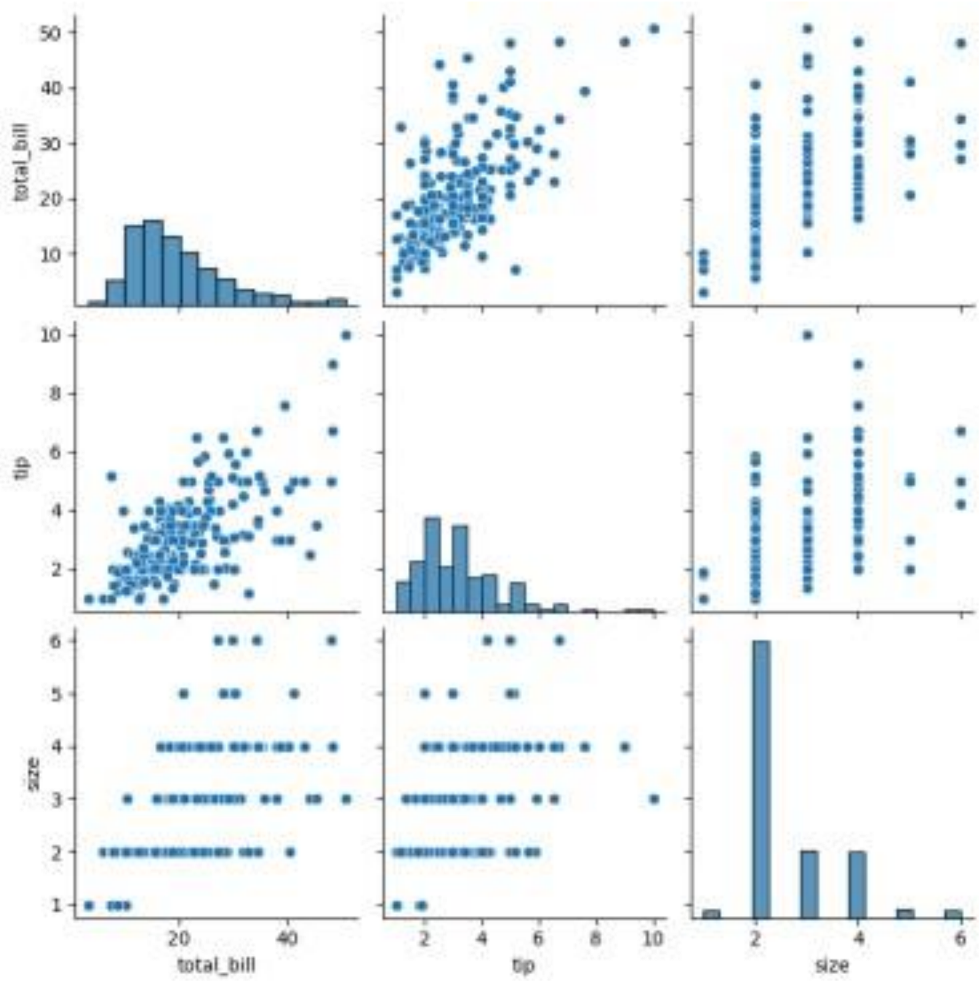



```
sns.jointplot(x=tips.tip,y=tips.total_bill,kind="hex")
```

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




```
sns.pairplot(tips)
     <seaborn.axisgrid.PairGrid at 0x79bb06fc3d30>
```

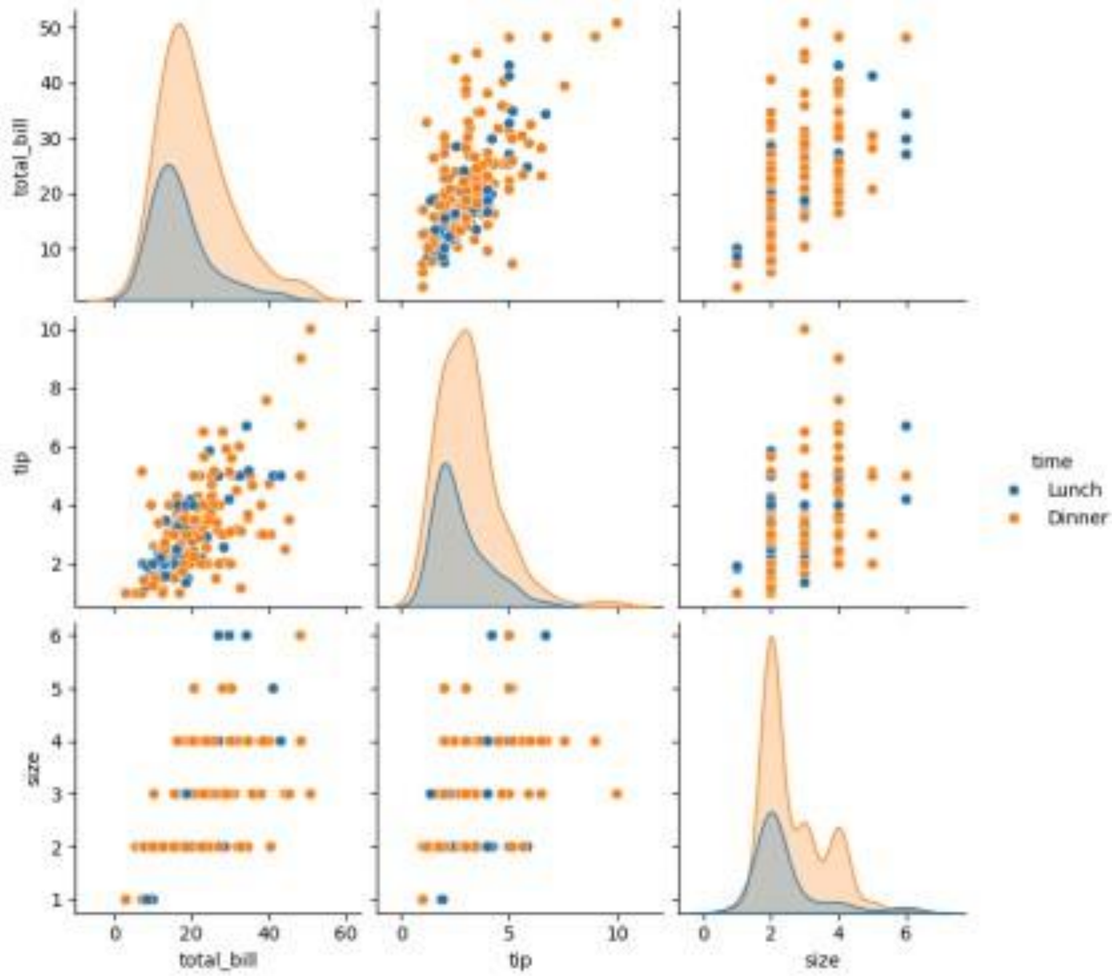



```
tips.time.value_counts()
```

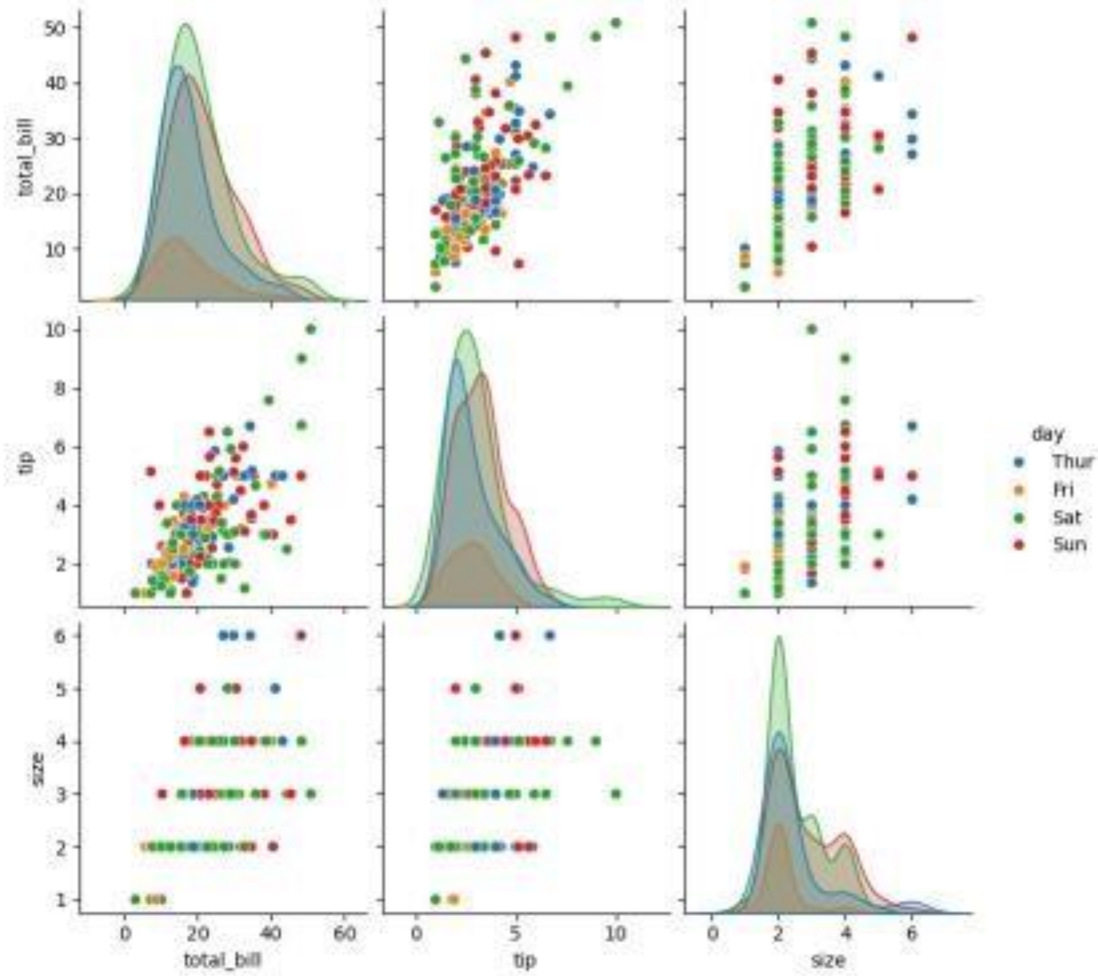
time	count
Dinner	176
Lunch	68

dtype: int64

```
sns.pairplot(tips,hue='time')
<seaborn.axisgrid.PairGrid at 0x79bb088f4670>
```

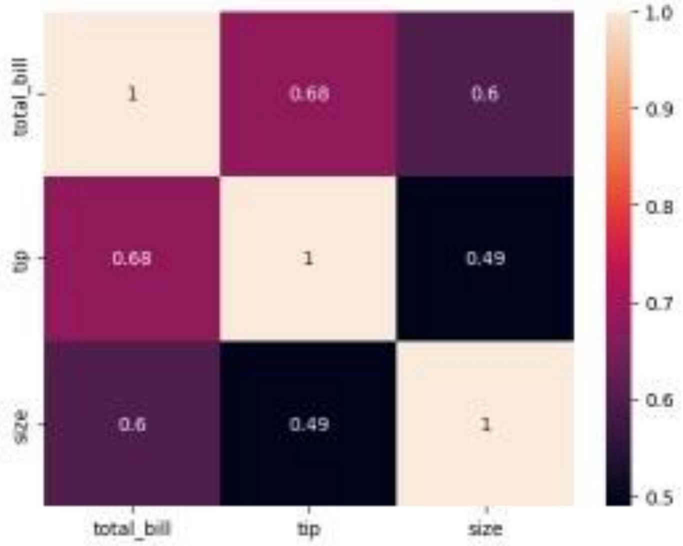



```
sns.pairplot(tips,hue='day')
<seaborn.axisgrid.PairGrid at 0x79bb08f1f6a0>
```



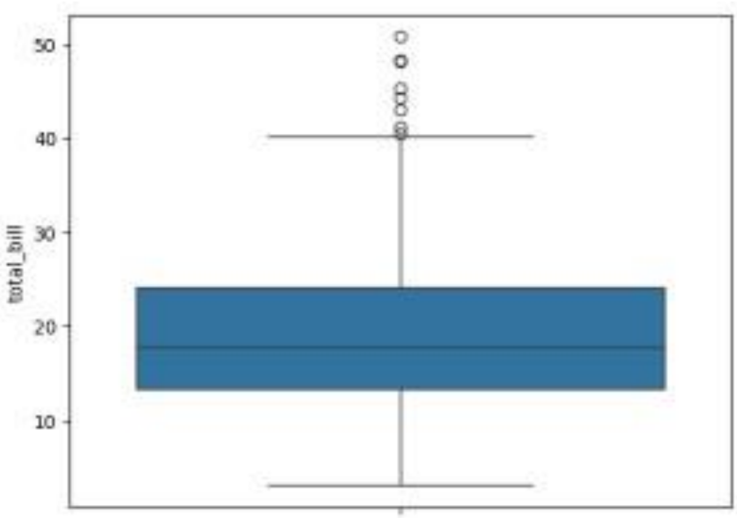

```
sns.heatmap(tips.corr(numeric_only=True),annot=True)
```

<Axes: >



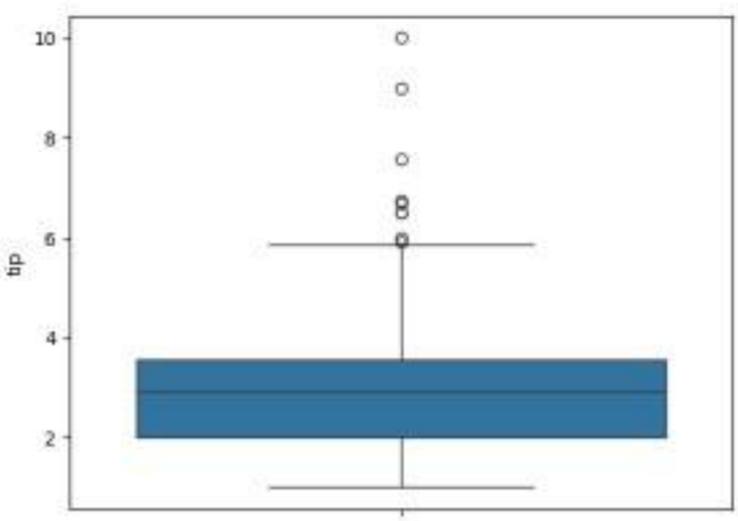

```
sns.boxplot(tips.total_bill)
```

<Axes: ylabel='total_bill'>



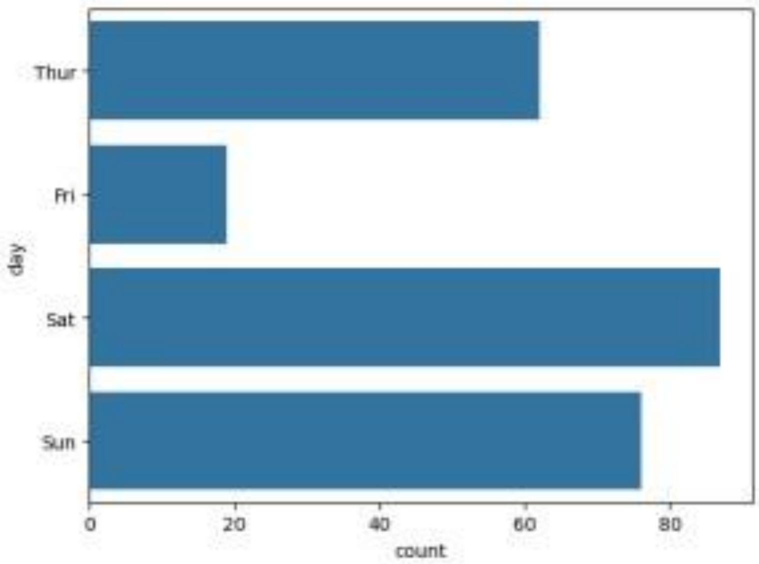

```
sns.boxplot(tips.tip)
```

<Axes: ylabel='tip'>




```
sns.countplot(tips.day)
```

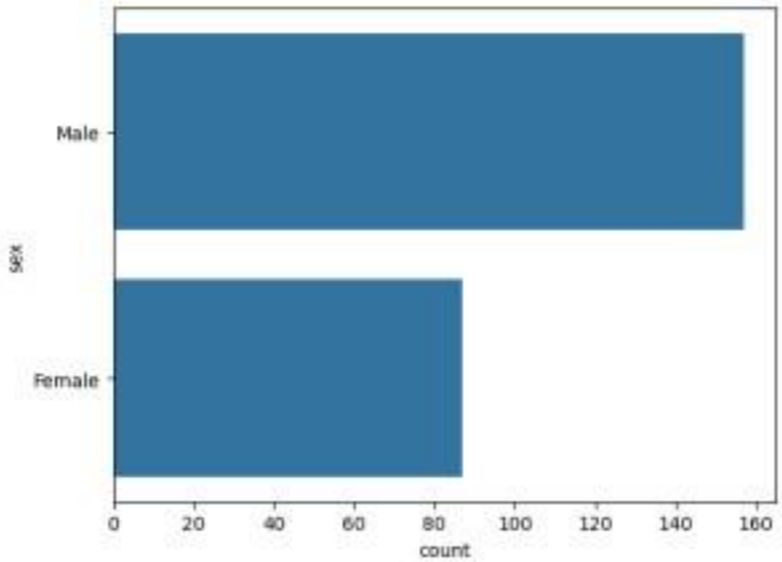
<Axes: xlabel='count', ylabel='day'>




```
sns.countplot(tips.sex)
```

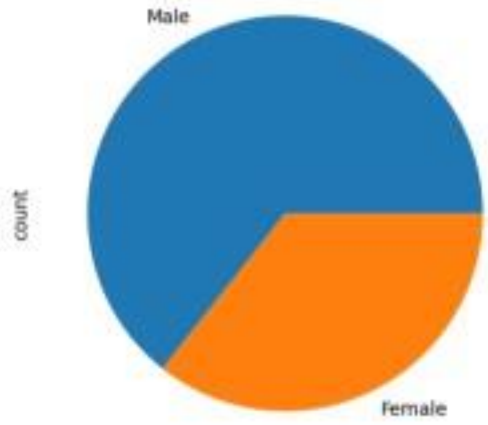


```
<Axes: xlabel='count', ylabel='sex'>
```



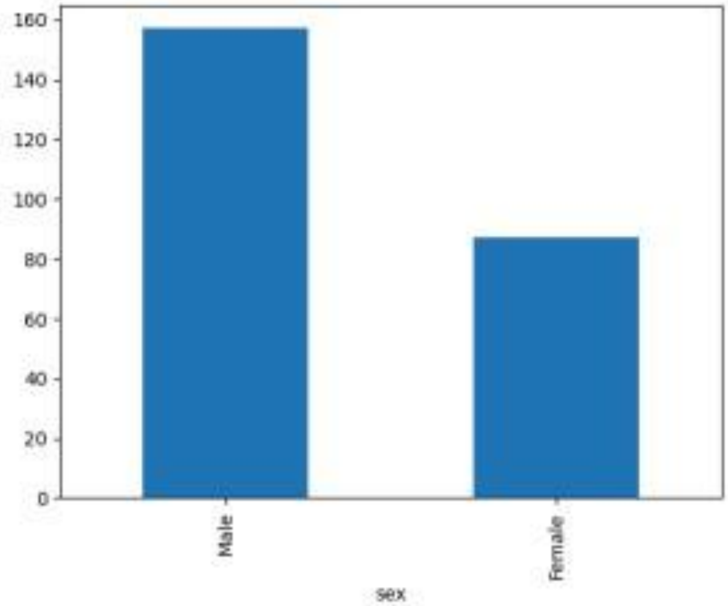

```
tips.sex.value_counts().plot(kind='pie')
```

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




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

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




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


Exp-9_Regression

```
In [ ]: In [19]:
In [3]: In [4]:
```

```
In [5]:  import numpy as np  import
pandas as pd
df=pd.read_csv('Salary_data.csv')
df

df.info()

<class 'pandas.core.frame.DataFrame'> 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

df.dropna(inplace=True)

df.info()

<class 'pandas.core.frame.DataFrame'> 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  df.describe()
```

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]:

In [ ]: In [29]:
In [ ]:
```
import pickle
pickle.dump(model,open('SalaryPred.model','wb'))

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

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
```
print("Estimated Salary for {} years of experience is {}: " .format(yr_of_exp,Salary)
```
Estimated Salary for 44.0 years of experience is

Exp-10_KNN

```
import math

import numpy as np

from statsmodels.stats.weightstats import ztest

from scipy.stats import norm


sample_marks = [650,730,510,670,480,800,690,530,590,620,710,670,640,780,650,490,800,600,510,700]


# Method 1 : Using Z-score


sample_mean = np.mean(sample_marks)

sample_size = np.count_nonzero(sample_marks)

population_mean = 600

population_std = 100

alpha = 0.05

z_score = (sample_mean-population_mean)/(population_std/math.sqrt(sample_size))

critical_value = 1.645 # from z table

if(z_score<critical_value):

   print('Null hypothesis is accepted!')

else:

   print('Null hypothesis is rejected. \nAlternate hypothesis is accepted!')



# Method 2: Using built in function of ztest


ztest_score, pval = ztest(sample_marks,value=population_mean,alternative='larger')

print('Z-test Score:',ztest_score,'\nP-value:',pval)

if(pval>alpha):

    print('Null hypothesis is accepted!')
```

```
else:
    print('Null hypothesis is rejected. \nAlternate hypothesis is accepted!')


# Method 3: Creating a function

def ztest(x,mu,sigma,n):
    deno = sigma/math.sqrt(n)
    z = (x-mu)/deno
    p = 2*(1-norm.cdf(abs(z)))
    return z,p

s_mean = np.mean(sample_marks)
p_mean = 600
p_std = 100
s_size = np.count_nonzero(sample_marks)

ztest(s_mean,p_mean,p_std,s_size)

ztest(641,600,100,20)
```

```
Null hypothesis is rejected.
Alternate hypothesis is accepted!
Z-test Score: 1.831744911595958
P-value: 0.03349471703839336
Null hypothesis is rejected.
Alternate hypothesis is accepted!

(1.8335757415498277, 0.06671699590108493)
```

Exp-11_Logistics Regression

```
# Import necessary libraries
import numpy as np
from scipy import stats


# Given student scores
```

```
student_scores = np.array([72, 89, 65, 73, 79, 84, 63, 76, 85, 75])

# Hypothesized population mean
mu = 70

# Perform one-sample t-test
t_stat, p_value = stats.ttest_1samp(student_scores, mu)
print("T statistic:", t_stat)
print("P-value:", p_value)

# Setting significance level
alpha = 0.05

# Interpret the results
if p_value < alpha:
    print("Reject the null hypothesis; there is a significant difference between the sample mean and the hypothesized population mean.")
else:
    print("Fail to reject the null hypothesis; there is no significant difference between the sample mean and the hypothesized population mean.")
```

```
T statistic: 2.2894683580127317
P-value: 0.047816221110566944
Reject the null hypothesis; there is a significant difference between the sample mean and the hypothesized population mean.
```

EXP-12_K-Mean clustering

```
import numpy as np
from scipy.stats import f_oneway

# Sample data: Exam scores for three teaching methods
np.random.seed(42)
```

```
method_A_scores = np.random.normal(loc=80, scale=10, size=30)

method_B_scores = np.random.normal(loc=85, scale=10, size=30)

method_C_scores = np.random.normal(loc=90, scale=10, size=30)


# Perform one-way ANOVA

f_statistic, p_value = f_oneway(method_A_scores, method_B_scores, method_C_scores)


print("F-Statistic:", f_statistic)

print("P-Value:", p_value)
```

```
F-Statistic: 12.20952551797281
P-Value: 2.1200748140507065e-05
```