eda1

September 2, 2024

1 Working on EDA

[1]:
```
# to work with EDA we have to load the dataset to the model for that we have to
 ↪import pandas library
```

[2]:
```
import pandas as pd # import the library
```

[3]:
```
pd.__version__ # check the version
```

[3]:
```
'2.2.2'
```

[4]:
```
# now we load the data to the notebook
data = pd.read_excel(r'C:\Users\sunil\Desktop\NIT- Data Science and AI
 ↪Class\September\2nd august\2nd- EDA Practicle\EDA- Practicle\Rawdata.xlsx')
 # data set loading to model
```

[5]:
```
data.head() # check the data first five row that it successfullay loaded or not
```

[5]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year

[6]:
```
# To check the shape
data.shape # rows and columns
```

[6]:
```
(6, 6)
```

[7]:
```
data.columns # check which columns we have
```

[7]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

[8]:
```
# get data set information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       4 non-null      object
 3   Location  4 non-null      object
 4   Salary    6 non-null      object
 5   Exp       5 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

[9]:
```
data.isnull() # check wheather there is any null value or not
```

[9]:

	Name	Domain	Age	Location	Salary	Exp
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	True	True	False	False
3	False	False	True	False	False	True
4	False	False	False	True	False	False
5	False	False	False	False	False	False

[10]:
```
data.isnull().sum() # calculate at which column how many null value we have
```

[10]:
```
Name        0
Domain      0
Age         2
Location    2
Salary      0
Exp         1
dtype: int64
```

[11]:
```
#we have some of null value ,, 2 null value at age attribute, also 2 in
 ↪location, and 1 in Exp attribute
```

[12]:
```
data
```

[12]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy^	Testing	45' yr	Bangalore	10%%000	<3
2	Uma#r	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam*	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

[13]:
```
# in the data frame there are some special symbol and some unwanted character␣
↪are there which affect on data lets remove these
```

[14]:
```
# In Name attribute remove unwanted symbol, lets read first
data["Name"]
```

[14]:
```
0      Mike
1    Teddy^
2     Uma#r
3      Jane
4    Uttam*
5       Kim
Name: Name, dtype: object
```

[15]:
```
# to remove these unwanted symbol
data['Name'] = data['Name'].str.replace(r'\W','',regex=True) # this is the␣
↪prompt use take this string to proper format
```

[16]:
```
data['Name'] # check wheather the name Attribute change successfully or not
```

[16]:
```
0     Mike
1    Teddy
2     Umar
3     Jane
4    Uttam
5      Kim
Name: Name, dtype: object
```

[17]:
```
data.head() # check the head
```

[17]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience#$	34 years	Mumbai	5^00#0	2+
1	Teddy	Testing	45' yr	Bangalore	10%%000	<3
2	Umar	Dataanalyst^^#	NaN	NaN	1$5%000	4> yrs
3	Jane	Ana^^lytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67-yr	NaN	30000-	5+ year

[18]:
```
# now  do for Domain attribute
```

[19]:
```
data['Domain'] # print Domain attribute
```

[19]:
```
0     Datascience#$
1           Testing
2    Dataanalyst^^#
3       Ana^^lytics
4        Statistics
5               NLP
```

```
Name: Domain, dtype: object
```

[20]:
```
data["Domain"] = data['Domain'].str.replace(r'\W','',regex=True) # remove unwanted character form the Domain columns
data['Domain'] # printing the domain column
```

[20]:
```
0    Datascience
1        Testing
2    Dataanalyst
3      Analytics
4     Statistics
5            NLP
Name: Domain, dtype: object
```

[21]:
```
data
```

[21]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34 years	Mumbai	5^00#0	2+
1	Teddy	Testing	45' yr	Bangalore	10%%000	<3
2	Umar	Dataanalyst	NaN	NaN	1$5%000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67-yr	NaN	30000-	5+ year
5	Kim	NLP	55yr	Delhi	6000^$0	10+

[22]:
```
# we done for Name, Domain column now for Age
```

[23]:
```
data['Age'] # getting the age columns
```

[23]:
```
0    34 years
1      45' yr
2         NaN
3         NaN
4       67-yr
5        55yr
Name: Age, dtype: object
```

[24]:
```
data['Age'] = data['Age'].str.replace(r'\W','',regex=True) # same do for the Age columns
```

[25]:
```
data['Age'] # get Age columns
```

[25]:
```
0    34years
1      45yr
2       NaN
3       NaN
4      67yr
5      55yr
```

```
Name: Age, dtype: object
```

[26]:
```
# now we have to extract only the numbers from the string data
```

[27]:
```
data['Age'] = data['Age'].str.extract('(\d+)') # to extract the numeric from␣
↪the text columns
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\sunil\AppData\Local\Temp\ipykernel_15212\3923553544.py:1:
SyntaxWarning: invalid escape sequence '\d'
  data['Age'] = data['Age'].str.extract('(\d+)') # to extract the numeric from
the text columns
```

[28]:
```
data['Age'] # print Age columns
```

[28]:
```
0     34
1     45
2    NaN
3    NaN
4     67
5     55
Name: Age, dtype: object
```

[29]:
```
data
```

[29]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5^00#0	2+
1	Teddy	Testing	45	Bangalore	10%%000	<3
2	Umar	Dataanalyst	NaN	NaN	1$5%000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	2000^0	NaN
4	Uttam	Statistics	67	NaN	30000-	5+ year
5	Kim	NLP	55	Delhi	6000^$0	10+

[30]:
```
# another 2 column we have Salary and Exp lets do
```

[31]:
```
data['Salary']
```

[31]:
```
0     5^00#0
1    10%%000
2    1$5%000
3     2000^0
4     30000-
5    6000^$0
Name: Salary, dtype: object
```

[32]:
```
# here we have to replace the raw string
data['Salary'] = data['Salary'].str.replace(r'\W','',regex=True) # replace the␣
 ↪string
data['Salary'] # print the salary
```

[32]:
```
0     5000
1    10000
2    15000
3    20000
4    30000
5    60000
Name: Salary, dtype: object
```

[33]:
```
data
```

[33]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2+
1	Teddy	Testing	45	Bangalore	10000	<3
2	Umar	Dataanalyst	NaN	NaN	15000	4> yrs
3	Jane	Analytics	NaN	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5+ year
5	Kim	NLP	55	Delhi	60000	10+

[34]:
```
# we done with all only Exp column left let do
```

[35]:
```
data['Exp']
```

[35]:
```
0        2+
1        <3
2     4> yrs
3       NaN
4    5+ year
5       10+
Name: Exp, dtype: object
```

[36]:
```
# here we do both of operation first replace then extract
```

[37]:
```
data['Exp'] = data['Exp'].str.replace(r'\W',"",regex=True) # replace the raw␣
 ↪string
data['Exp'] # print it
```

[37]:
```
0        2
1        3
2     4yrs
3      NaN
4    5year
5       10
```

```
Name: Exp, dtype: object
```

[38]:
```
# now we have to extract only the numeric
```

[39]:
```
data["Exp"] = data['Exp'].str.extract('(\d+)') # extract the intiger
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\sunil\AppData\Local\Temp\ipykernel_15212\4286978194.py:1:
SyntaxWarning: invalid escape sequence '\d'
  data["Exp"] = data['Exp'].str.extract('(\d+)') # extract the intiger
```

[40]:
```
data["Exp"] # printing the Exp columns
```

[40]:
```
0      2
1      3
2      4
3    NaN
4      5
5     10
Name: Exp, dtype: object
```

1.0.1 we successfully bring the data set to proper format but it not clean let clean the data

[41]:
```
# now ne have to chenge these null value by help of data cleaning
```

[42]:
```
# we perform these by individuals columns
```

[43]:
```
# first take Age columns
data['Age'] # identify the null value by individual attribute
```

[43]:
```
0     34
1     45
2    NaN
3    NaN
4     67
5     55
Name: Age, dtype: object
```

[44]:
```
# Lets remove the null velue from the all columns one by one
```

[ ]:
```

```

1.1 To fill the null value we have to import numpy library

2 EDA techenique applay(7 techeniques)

- variable identificaton
- univerate analysis
- baiveriate analysis
- variable creation
- variable transformation
- outlire tretment
- missing value tritment

3 Missing value Treatment

- In the missing value Treatment we have two types of value
- 1) numerical value - for thos we use mean,median,mode strategy
  2) categorical value - for this we use mode strategy or KNN strategy
-

[45]:
```
import numpy as np
```

[46]:
```
data['Age']
```

[46]:
```
0     34
1     45
2    NaN
3    NaN
4     67
5     55
Name: Age, dtype: object
```

[47]:
```
# As the column have numerical value we approch to fill the value is mean,␣
 ↪median, mode strategy
```

[48]:
```
# lets fill the null value with mean strategy
```

[49]:
```
data['Age'] = data['Age'].fillna(np.mean(pd.to_numeric(data['Age']))) # here we␣
 ↪fill at the age column with mean strategy to numeric
```

[50]:
```
data["Age"] # print age column
```

[50]:
```
0       34
1       45
2    50.25
3    50.25
4       67
5       55
```

```
Name: Age, dtype: object
```

[51]:
```
# age column fill the null value done successfully
```

[52]:
```
data
```

[52]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50.25	NaN	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	NaN
4	Uttam	Statistics	67	NaN	30000	5
5	Kim	NLP	55	Delhi	60000	10

[53]:
```
# then we have Location where 2 null value to fill
# as the Location column is categorical data have we apply mode strategy to␣
 ↪fill it
```

[54]:
```
data['Location']
```

[54]:
```
0       Mumbai
1    Bangalore
2          NaN
3     Hyderbad
4          NaN
5        Delhi
Name: Location, dtype: object
```

[55]:
```
data["Location"] = data["Location"].fillna(data['Location'].mode()[0]) # this␣
 ↪is the way to fill  the categotical null value data
```

[56]:
```
data["Location"] # print the Location column
```

[56]:
```
0       Mumbai
1    Bangalore
2    Bangalore
3     Hyderbad
4    Bangalore
5        Delhi
Name: Location, dtype: object
```

[57]:
```
# Location column done success fully
```

[58]:
```
data
```

[58]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2

```
1  Teddy      Testing     45  Bangalore  10000    3
2   Umar  Dataanalyst  50.25  Bangalore  15000    4
3   Jane    Analytics  50.25   Hyderbad  20000  NaN
4  Uttam   Statistics     67  Bangalore  30000    5
5    Kim          NLP     55      Delhi  60000   10
```

[59]:
```
# then we have only Exp. Column to fill the null value
```

[60]:
```
data['Exp']
```

[60]:
```
0      2
1      3
2      4
3    NaN
4      5
5     10
Name: Exp, dtype: object
```

[61]:
```
# as it numeric here our approch is mean
```

[62]:
```
data['Exp'] = data['Exp'].fillna(np.mean(pd.to_numeric(data['Exp']))) # fill␣
 ↪the null value as calculation to mean and fill it
```

[63]:
```
data['Exp'] # print
```

[63]:
```
0      2
1      3
2      4
3    4.8
4      5
5     10
Name: Exp, dtype: object
```

[64]:
```
# Done successfully
```

[65]:
```
data # lets check the data frame
```

[65]:
```
     Name       Domain    Age   Location Salary  Exp
0   Mike  Datascience     34     Mumbai   5000    2
1  Teddy      Testing     45  Bangalore  10000    3
2   Umar  Dataanalyst  50.25  Bangalore  15000    4
3   Jane    Analytics  50.25   Hyderbad  20000  4.8
4  Uttam   Statistics     67  Bangalore  30000    5
5    Kim          NLP     55      Delhi  60000   10
```

[66]:
```
# here we have the clean data lets copy these clean data
```

[67]:
```
Emp_data = data.copy()
```

[68]:
```
Emp_data
```

[68]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50.25	Bangalore	15000	4
3	Jane	Analytics	50.25	Hyderbad	20000	4.8
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

[69]:
```
Emp_data.info() # to get the information about the data frame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       6 non-null      object
 3   Location  6 non-null      object
 4   Salary    6 non-null      object
 5   Exp       6 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

[70]:
```
# here we have all values are non- null because there is no any null values
```

3.0.1 Now our work to change the datatype

[71]:
```
# we have to types of data categorical(text), numerical(number)
```

[72]:
```
# we have Name, Domain and Location as categorical data and Age, Salary, and␣
↪Exp as numerical data
```

[73]:
```
Emp_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      object
 1   Domain    6 non-null      object
 2   Age       6 non-null      object
```

```
 3   Location  6 non-null      object
 4   Salary    6 non-null      object
 5   Exp       6 non-null      object
dtypes: object(6)
memory usage: 420.0+ bytes
```

[74]:
```
# lets change the type for category
```

[75]:
```
Emp_data['Name'] =Emp_data['Name'].astype("category") # this is for Name␣
 ↪attribute data type
Emp_data['Domain'] = Emp_data['Domain'].astype('category') # this is for Domain␣
 ↪attribute datatype
Emp_data['Location'] = Emp_data['Location'].astype('category') # this is for␣
 ↪Location attribute Location
```

[76]:
```
# lets do change data type for numeric
```

[77]:
```
Emp_data['Age'] = Emp_data['Age'].astype(int)#change type of age to int
Emp_data['Salary'] = Emp_data['Salary'].astype(int) # change typeof the salary␣
 ↪to int
Emp_data['Exp'] = Emp_data['Exp'].astype(int) # change type of the Exp
```

[78]:
```
# lets check the whole data
```

[79]:
```
Emp_data.info() # information about the data frame
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Name      6 non-null      category
 1   Domain    6 non-null      category
 2   Age       6 non-null      int32
 3   Location  6 non-null      category
 4   Salary    6 non-null      int32
 5   Exp       6 non-null      int32
dtypes: category(3), int32(3)
memory usage: 866.0 bytes
```

[80]:
```
# it done successfully
```

[81]:
```
Emp_data
```

[81]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3

```
2   Umar  Dataanalyst   50  Bangalore   15000    4
3   Jane    Analytics   50   Hyderbad   20000    4
4  Uttam   Statistics   67  Bangalore   30000    5
5    Kim          NLP   55      Delhi   60000   10
```

[82]:
```
#we have to convert the to csv to download the dataframe
```

[83]:
```
Emp_data.to_csv("Emp_data.csv") # change the file to csv
```

[84]:
```
# change successfully
```

[85]:
```
import os # import os
os.getcwd() # get the file location where the file is
```

[85]:
```
'C:\\Users\\sunil'
```

[ ]:
```

```

[86]:
```
Emp_data
```

[86]:
```
     Name       Domain  Age   Location  Salary  Exp
0   Mike  Datascience   34     Mumbai    5000    2
1  Teddy      Testing   45  Bangalore   10000    3
2   Umar  Dataanalyst   50  Bangalore   15000    4
3   Jane    Analytics   50   Hyderbad   20000    4
4  Uttam   Statistics   67  Bangalore   30000    5
5    Kim          NLP   55      Delhi   60000   10
```

[ ]:
```

```

[87]:
```
# we done with variable identification
```

3.1 Univariate analysis

- Univariate analysis define to plot the graph using a single variable

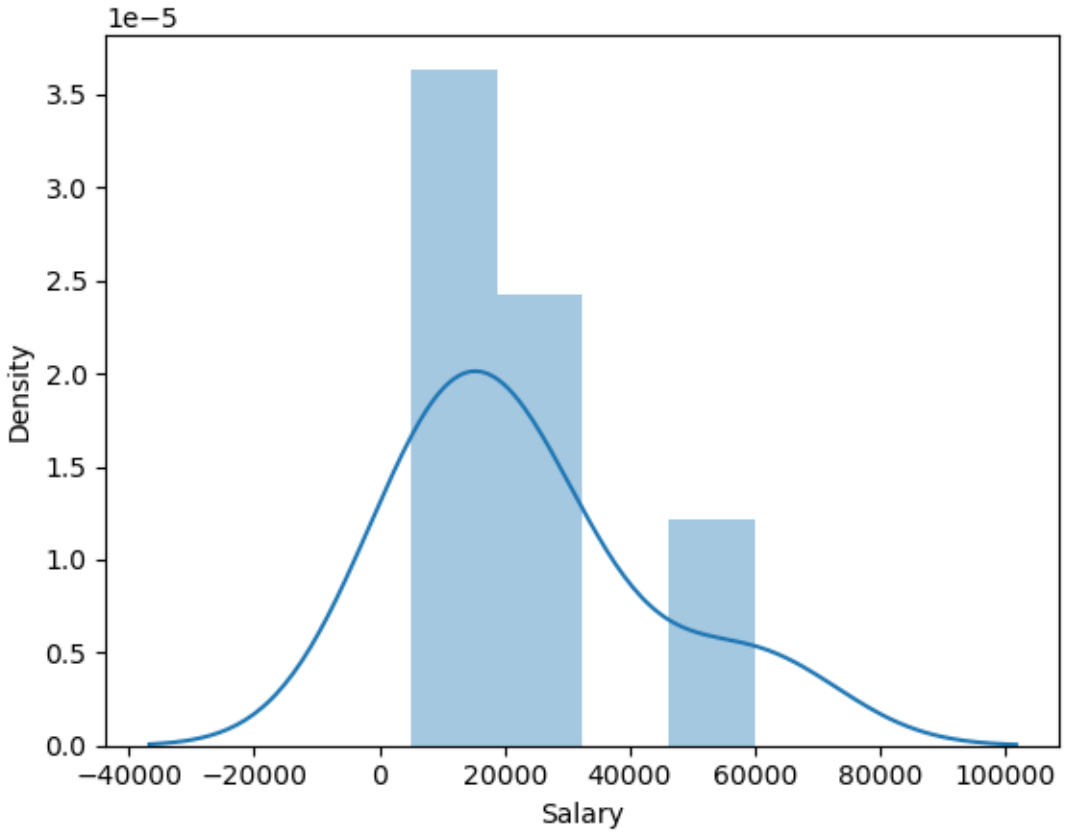
[88]:
```
# for doing rest all EDA techenique we have to import matplotlib and seaboarn␣
↪library to model
```

[89]:
```
import matplotlib.pyplot as plt  # import the matplotlib library
import seaborn as sns # import seaborn library
```

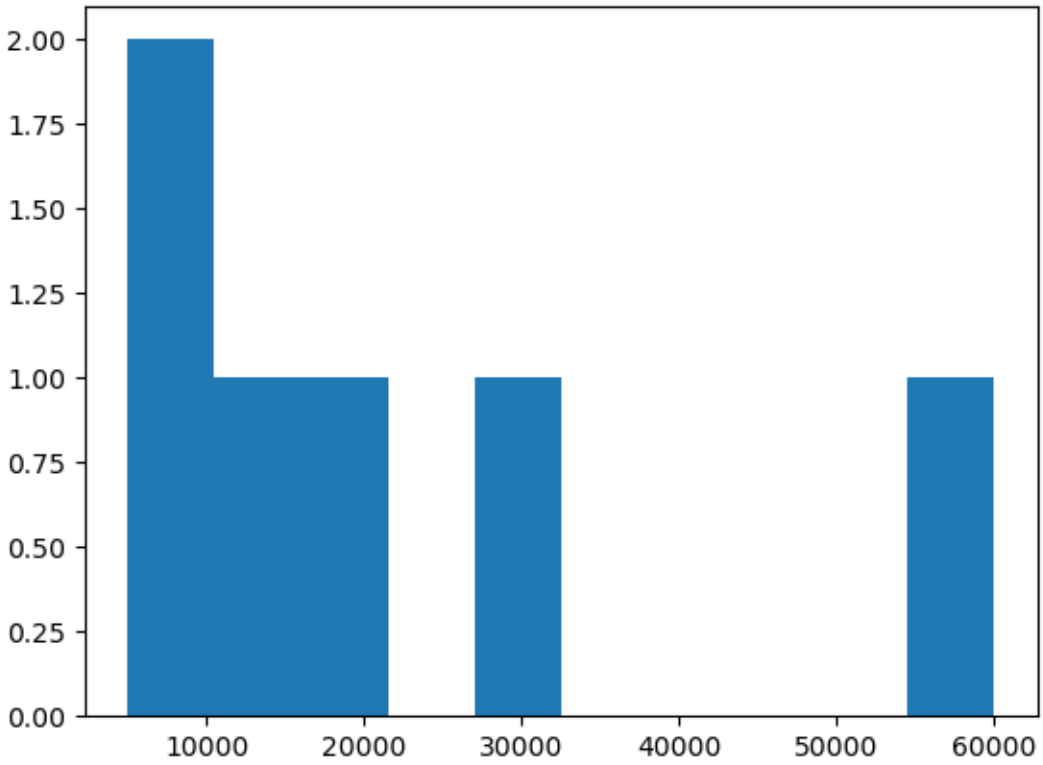
[90]:
```
# before doing the plot we have to remove all unwanted warning messeges
```

[91]:
```
import warnings # import warning library
warnings.filterwarnings('ignore') #to ignore all un wanted messeges
```

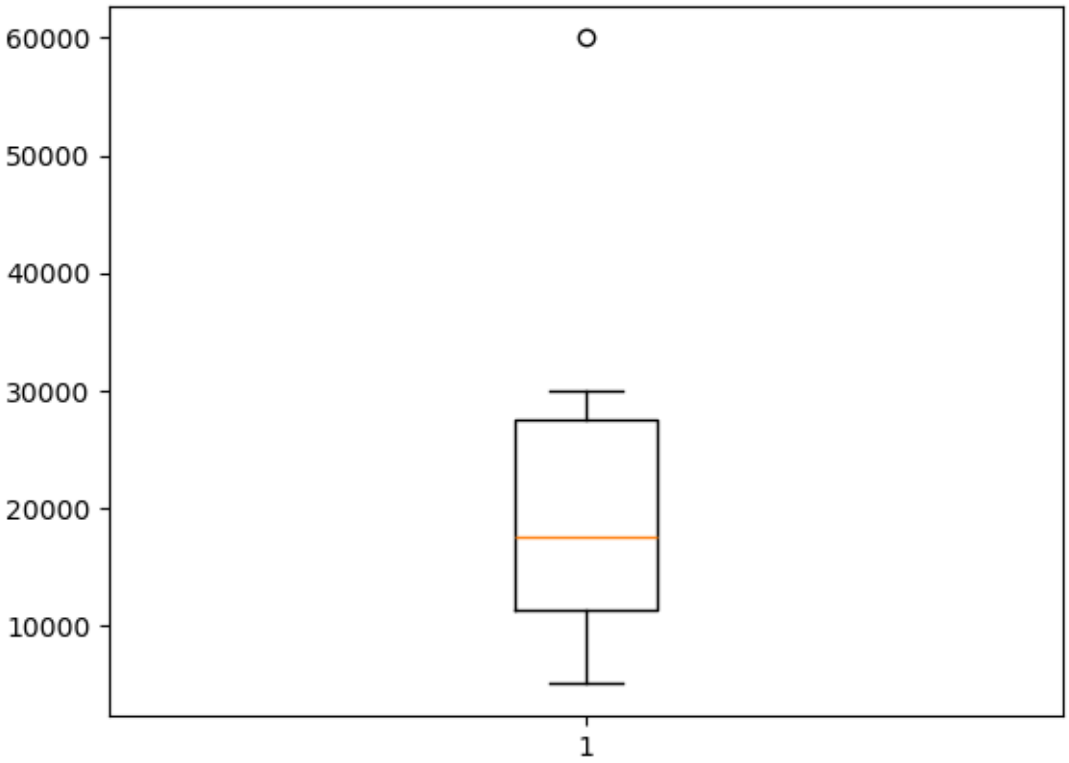
[92]:
```
# plot first visuals distplot
vis1 = sns.distplot(Emp_data['Salary']) # here we plot the data using single␣
↪variaable Salary
```



[93]:
```
# lets do with hisplot
vis2 = plt.hist(Emp_data['Salary'])
```



[94]:
```
vis3 = plt.boxplot(Emp_data['Salary'])
```



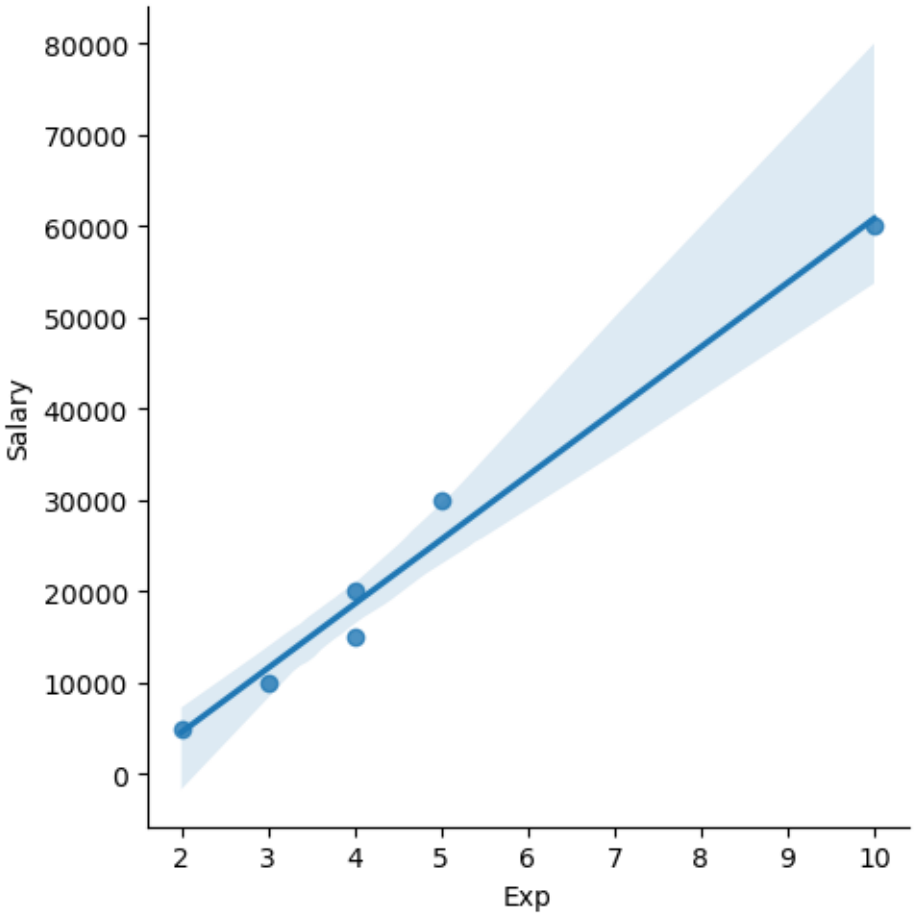
4 outlier Treatment

- Before to treat the outlire we have to detect the outlier
- The outlier we dect by the help of visualization
- from above figure we got 60000 as outlier bcauge its different form other figure

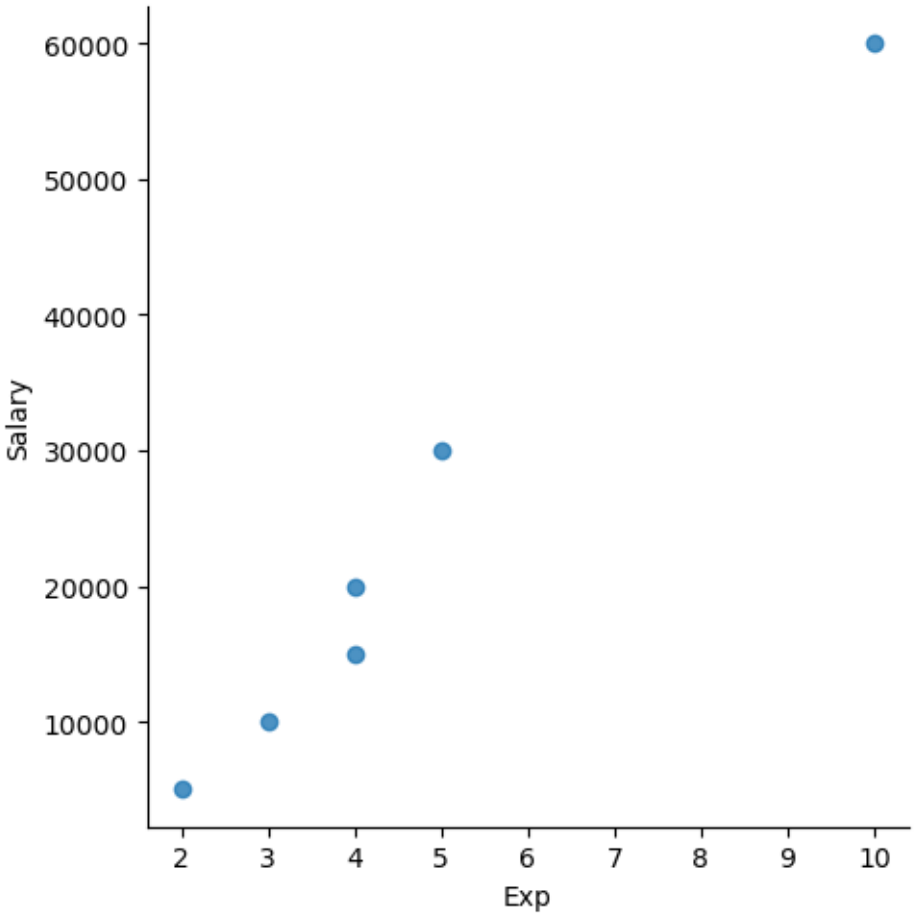
5 Bivariate analysis

- The analysis Done help to Two variable

[95]:
```
vis4 = sns.lmplot(Emp_data,x= "Exp", y="Salary") # here we have two variable␣
↪Exp ans Salary  attribute
```



[96]:
```
vis4 = sns.lmplot(Emp_data,x= "Exp", y="Salary", fit_reg= False) # remove the␣
↪regressinon line
```



5.1 Variable identification

- we have two types of variable dependent and independet variable
- as the salary increasing continously and rest of the columns depends over the salary its dependent variable(Y)
- and rest all columns are independent variable (X1,X2,X3,X4,X5) Name, DOmain,Age,Location, and Exp respectively

[97]:
```
Emp_data.columns # get the columns
```

[97]:
```
Index(['Name', 'Domain', 'Age', 'Location', 'Salary', 'Exp'], dtype='object')
```

[98]:
```
# lets split the data set to two part one is dependatn and indeoendent variable
```

5.2 Dependent variable

[99]:
```
y_dip_v = Emp_data[['Salary']]# extract the salary as one dataset
```

[100]:
```
y_dip_v
```

[100]:

	Salary
0	5000
1	10000
2	15000
3	20000
4	30000
5	60000

5.3 Independent Variable

[101]:
```
# similarly do for independent variablr
```

[102]:
```
x_idip_v = Emp_data[['Name','Domain','Age','Location','Exp']] # extract the␣
↪independent variable from data set
x_idip_v
```

[102]:

	Name	Domain	Age	Location	Exp
0	Mike	Datascience	34	Mumbai	2
1	Teddy	Testing	45	Bangalore	3
2	Umar	Dataanalyst	50	Bangalore	4
3	Jane	Analytics	50	Hyderbad	4
4	Uttam	Statistics	67	Bangalore	5
5	Kim	NLP	55	Delhi	10

6 Now we have 3 data set

- Emp_data
- y_dip_v(dependent variable)
- x_indip_v(independent variable)

[103]:
```
# lets print all the dataset
```

[104]:
```
Emp_data.head()
```

[104]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5

[105]:
```
y_dip_v.head()
```

[105]:

	Salary
0	5000
1	10000
2	15000
3	20000
4	30000

[106]:
```
x_idip_v.head()
```

[106]:

	Name	Domain	Age	Location	Exp
0	Mike	Datascience	34	Mumbai	2
1	Teddy	Testing	45	Bangalore	3
2	Umar	Dataanalyst	50	Bangalore	4
3	Jane	Analytics	50	Hyderbad	4
4	Uttam	Statistics	67	Bangalore	5

7 variable creation and Variable Transformation

[107]:
```
# here we work with transformer to take this data into machine readable format
```

[108]:
```
imputation = pd.get_dummies(Emp_data) # here we creat dummy variable to␣
 ↪transfer the variable and creationthe variable
```

[109]:
```
imputation # lets print
```

[109]:

	Age	Salary	Exp	Name_Jane	Name_Kim	Name_Mike	Name_Teddy	Name_Umar	Name_Uttam	Domain_Analytics	Domain_Dataanalyst	Domain_Datascience	Domain_NLP	Domain_Statistics	Domain_Testing	Location_Bangalore
0	34	5000	2	False	False	True	False	False	False	False	False	True	False	False	False	False
1	45	10000	3	False	False	False	True	False	False	False	False	False	False	False	True	True
2	50	15000	4	False	False	False	False	True	False	False	True	False	False	False	False	False
3	50	20000	4	True	False	False	False	False	False	True	False	False	False	False	False	False
4	67	30000	5	False	False	False	False	False	True	False	False	False	False	False	False	False
5	55	60000	10	False	True	False	False	False	False	False	False	False	False	False	False	False

```
2       False              False           False                True
3       False              False           False               False
4       False               True           False                True
5        True              False           False               False

   Location_Delhi  Location_Hyderbad  Location_Mumbai
0           False              False             True
1           False              False            False
2           False              False            False
3           False               True            False
4           False              False            False
5            True              False            False
```

[110]:
```
#as we get the data as true and false its bool type lets convert its type to int
```

[114]:
```
imputation=imputation.astype(int)
```

[115]:
```
# we got the output as we required
```

[116]:
```
imputation
```

[116]:
```
    Age  Salary  Exp  Name_Jane  Name_Kim  Name_Mike  Name_Teddy  Name_Umar  \
0   34    5000    2          0         0          1           0          0
1   45   10000    3          0         0          0           1          0
2   50   15000    4          0         0          0           0          1
3   50   20000    4          1         0          0           0          0
4   67   30000    5          0         0          0           0          0
5   55   60000   10          0         1          0           0          0

   Name_Uttam  Domain_Analytics  Domain_Dataanalyst  Domain_Datascience  \
0           0                 0                   0                   1
1           0                 0                   0                   0
2           0                 0                   1                   0
3           0                 1                   0                   0
4           1                 0                   0                   0
5           0                 0                   0                   0

   Domain_NLP  Domain_Statistics  Domain_Testing  Location_Bangalore  \
0           0                  0               0                   0
1           0                  0               1                   1
2           0                  0               0                   1
3           0                  0               0                   0
4           0                  1               0                   1
5           1                  0               0                   0

   Location_Delhi  Location_Hyderbad  Location_Mumbai
0               0                  0                1
```

1	0	0	0
2	0	0	0
3	0	1	0
4	0	0	0
5	1	0	0

[117]:
```
Emp_data
```

[117]:

	Name	Domain	Age	Location	Salary	Exp
0	Mike	Datascience	34	Mumbai	5000	2
1	Teddy	Testing	45	Bangalore	10000	3
2	Umar	Dataanalyst	50	Bangalore	15000	4
3	Jane	Analytics	50	Hyderbad	20000	4
4	Uttam	Statistics	67	Bangalore	30000	5
5	Kim	NLP	55	Delhi	60000	10

8 we have completed EDA part successfully

9

- data set load
- remove numwanted this from data from every column
- treat missing value mean ana mode strategy
- outlier dectection
- univariate and bivariate analysis
- variable identification
- dummy vari