DATA COLLECTION ¶

In [1]:

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\Instagram.csv")
sd
```

Out[2]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	F
0	3920	2586	1028	619	56	98	9	5	162	35	
1	5394	2727	1838	1174	78	194	7	14	224	48	
2	4021	2085	1188	0	533	41	11	1	131	62	
3	4528	2700	621	932	73	172	10	7	213	23	
4	2518	1704	255	279	37	96	5	4	123	8	
...	...	...	...	...	...	...	...	...	...	...	
114	13700	5185	3041	5352	77	573	2	38	373	73	
115	5731	1923	1368	2266	65	135	4	1	148	20	
116	4139	1133	1538	1367	33	36	0	1	92	34	
117	32695	11815	3147	17414	170	1095	2	75	549	148	

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	F
118	36919	13473	4176	16444	2547	653	5	26	443	611	

119 rows × 13 columns

In [3]:

```
# to display top 10 rows
sd.head(10)
```

Out[3]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Foll
0	3920	2586	1028	619	56	98	9	5	162	35	
1	5394	2727	1838	1174	78	194	7	14	224	48	
2	4021	2085	1188	0	533	41	11	1	131	62	
3	4528	2700	621	932	73	172	10	7	213	23	
4	2518	1704	255	279	37	96	5	4	123	8	
5	3884	2046	1214	329	43	74	7	10	144	9	
6	2621	1543	599	333	25	22	5	1	76	26	
7	3541	2071	628	500	60	135	4	9	124	12	
8	3749	2384	857	248	49	155	6	8	159	36	
9	4115	2609	1104	178	46	122	6	3	191	31	

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119 entries, 0 to 118
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Impressions     119 non-null    int64 
 1   From Home       119 non-null    int64 
 2   From Hashtags   119 non-null    int64 
 3   From Explore    119 non-null    int64 
 4   From Other      119 non-null    int64 
 5   Saves           119 non-null    int64 
 6   Comments        119 non-null    int64 
 7   Shares          119 non-null    int64 
 8   Likes           119 non-null    int64 
 9   Profile Visits  119 non-null    int64 
 10  Follows         119 non-null    int64 
 11  Caption         119 non-null    object
 12  Hashtags        119 non-null    object
dtypes: int64(11), object(2)
memory usage: 12.2+ KB
```

In [5]:
```
# to display summary of statistics
sd.describe()
```

Out[5]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comm
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.00
mean	5703.991597	2475.789916	1887.512605	1078.100840	171.092437	153.310924	6.66
std	4843.780105	1489.386348	1884.361443	2613.026132	289.431031	156.317731	3.54
min	1941.000000	1133.000000	116.000000	0.000000	9.000000	22.000000	0.00
25%	3467.000000	1945.000000	726.000000	157.500000	38.000000	65.000000	4.00
50%	4289.000000	2207.000000	1278.000000	326.000000	74.000000	109.000000	6.00
75%	6138.000000	2602.500000	2363.500000	689.500000	196.000000	169.000000	8.00
max	36919.000000	13473.000000	11817.000000	17414.000000	2547.000000	1095.000000	19.00

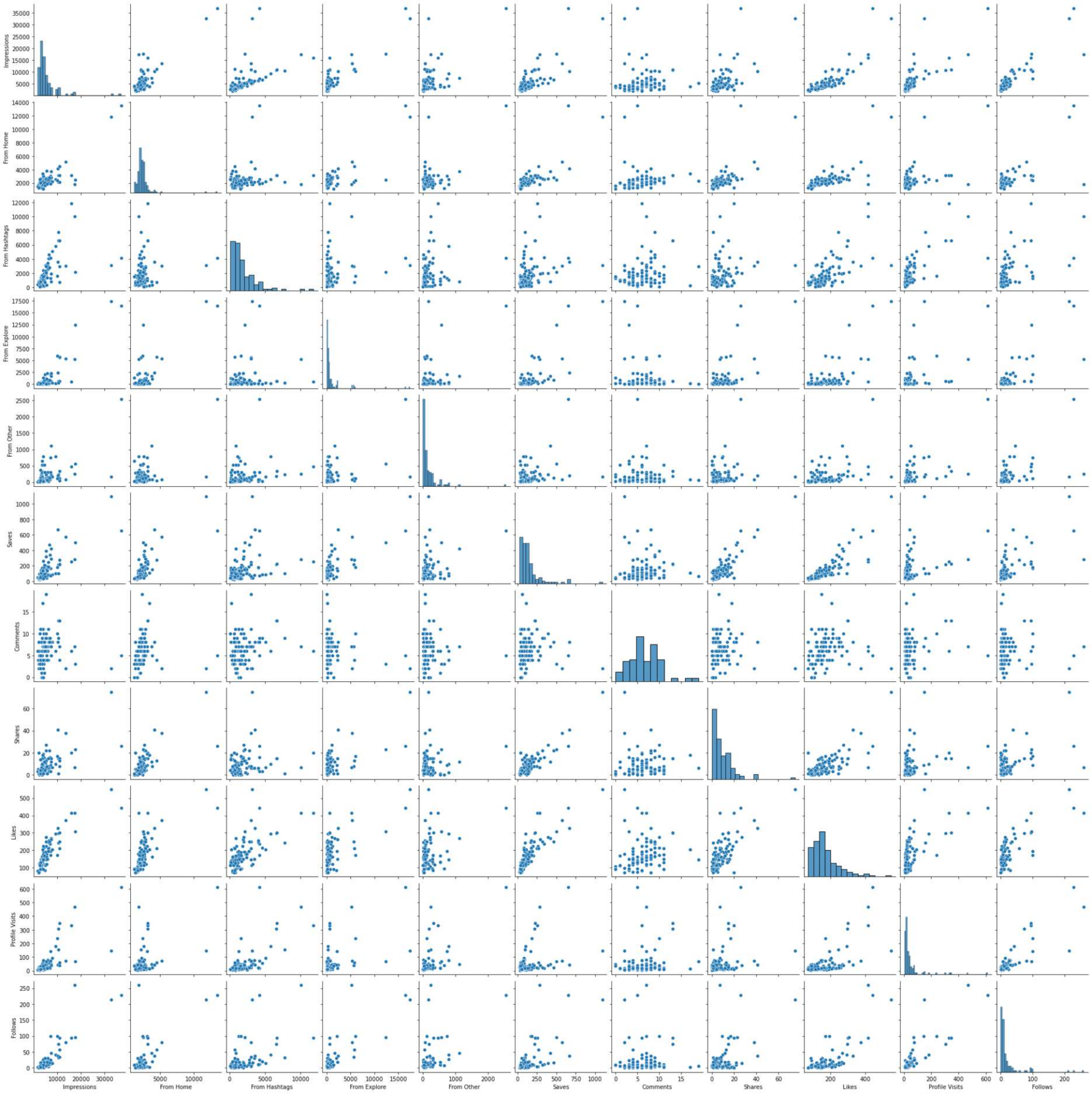
In [6]:
```
#to display colums heading
sd.columns
```

Out[6]:
```
Index(['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visit
s',
       'Follows', 'Caption', 'Hashtags'],
      dtype='object')
```

EDA and visualization

In [7]:
```
sns.pairplot(sd)
```

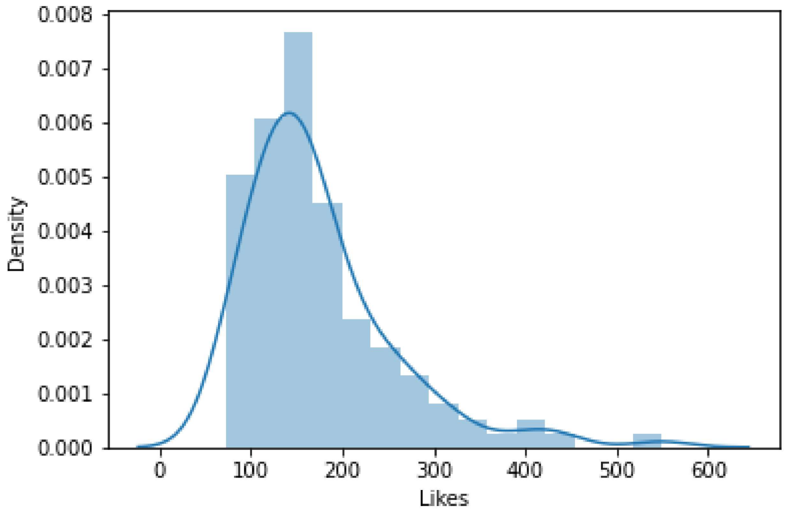
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x207d1b95a60>
```



```
In [8]: sns.distplot(sd['Likes'])
```

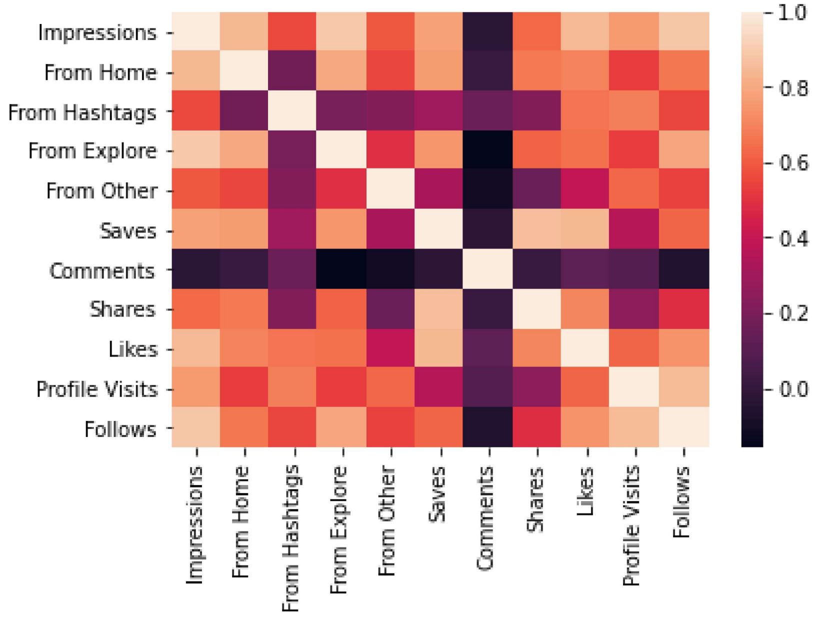
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

```
Out[8]: <AxesSubplot:xlabel='Likes', ylabel='Density'>
```



```
In [9]: sns.heatmap(sd.corr())
```

```
Out[9]: <AxesSubplot:>
```



```
In [10]: sd1=sd[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
                'From Other', 'Saves', 'Comments', 'Shares', 'Likes']]
```

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [12]:

```
x= sd1[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares']]
y=sd1['Likes']
```

In [13]:

```
# To split my dataset  into training data and test data
from sklearn .model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [14]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:

```
LinearRegression()
```

In [15]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
print(lr.intercept_)
```

```
48.16285509979126
```

In [17]:

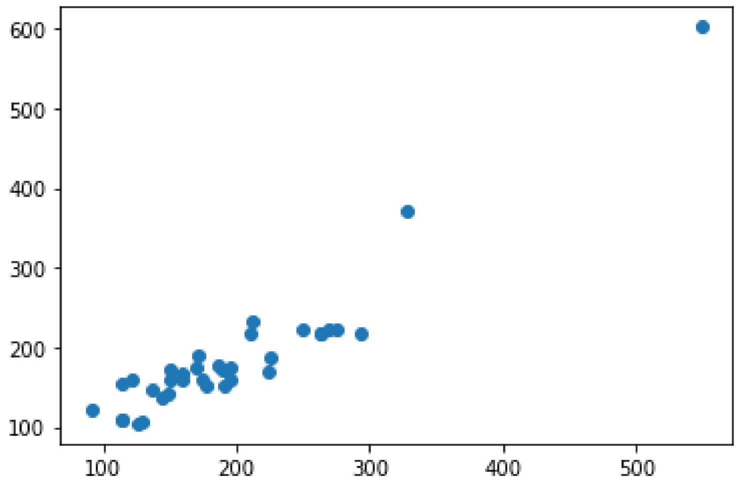
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
Impressions	0.034668
From Home	-0.022656
From Hashtags	-0.014381
From Explore	-0.032049
From Other	-0.059554
Saves	0.284167
Comments	1.650921
Shares	-0.166533

In [18]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x207d84c4ca0>
```



In [19]:
```
print(lr.score(x_test,y_test))
```

```
0.846317631408496
```

In [20]:
```
lr.score(x_train,y_train)
```

Out[20]:
```
0.9242316202137021
```

In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [22]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[22]:
```
Ridge(alpha=10)
```

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

Out[23]:
```
0.8461734715924772
```

In [24]:
```
dr.score(x_train,y_train)
```

Out[24]:
```
0.9242310915143706
```

In [25]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_d
escent.py:530: ConvergenceWarning: Objective did not converge. You might want
to increase the number of iterations. Duality gap: 18720.540962154744, tolera
nce: 53.4106
  model = cd_fast.enet_coordinate_descent(
```

Out[25]:
```
Lasso(alpha=10)
```

In [26]:
```
la.score(x_test,y_test)
```

Out[26]:
```
0.8420764286891529
```

In [27]:
```
la.score(x_train,y_train)
```

Out[27]:
```
0.9225124512045798
```

ElasticNet

In [28]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_d
escent.py:530: ConvergenceWarning: Objective did not converge. You might want
to increase the number of iterations. Duality gap: 20334.63129157652, toleran
ce: 53.4106
  model = cd_fast.enet_coordinate_descent(
```

Out[28]:
```
ElasticNet()
```

In [29]:
```
print(en.coef_)
```

```
[ 0.02368373 -0.0116045  -0.00335457 -0.02100008 -0.04789254  0.2793984
  1.52069937 -0.09961933]
```

In [30]:
```
print(en.intercept_)
```

```
49.770335353776375
```

In [31]:
```
prediction=en.predict(x_test)
```

In [32]:
```
print(en.score(x_test,y_test))
```

```
0.8504032469290994
```

Evaluation metrics

In [33]:
```
from sklearn import metrics
```

In [34]:
```
print("mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolytre Error: 26.034025251956084
```

In [35]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 999.0819153840098
```

In [36]:

```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 31.60825707602382
```

In [ ]:

```
```