mk 31/07/23

In [ ]:

In [26]:
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:
```
x=pd.read_csv(r"C:\Users\user\Downloads\5_Instagram data.csv")
x
```

116	4139	1133	1538	1367	33	36	0	1	92	34	10	
117	32695	11815	3147	17414	170	1095	2	75	549	148	214	
118	36919	13473	4176	16444	2547	653	5	26	443	611	228	

In [30]:
```
x.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 [31]:
```
x.columns
```

Out[31]:
```
Index(['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows', 'Caption', 'Hashtags'],
      dtype='object')
```

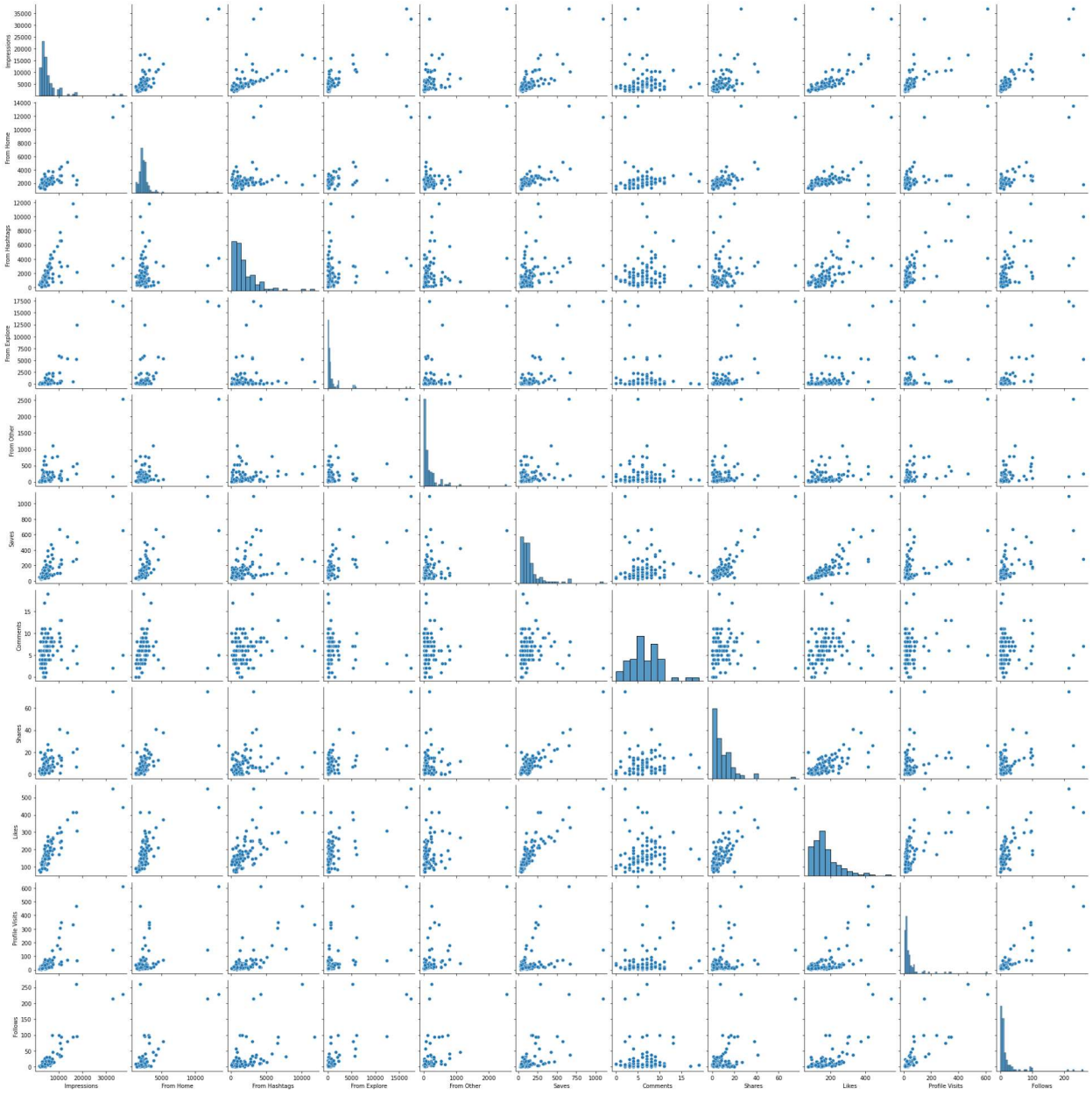
In [32]:
```
x.describe()
```

Out[32]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.
mean	5703.991597	2475.789916	1887.512605	1078.100840	171.092437	153.310924	6.663866	9.
std	4843.780105	1489.386348	1884.361443	2613.026132	289.431031	156.317731	3.544576	10.
min	1941.000000	1133.000000	116.000000	0.000000	9.000000	22.000000	0.000000	0.
25%	3467.000000	1945.000000	726.000000	157.500000	38.000000	65.000000	4.000000	3.
50%	4289.000000	2207.000000	1278.000000	326.000000	74.000000	109.000000	6.000000	6.
75%	6138.000000	2602.500000	2363.500000	689.500000	196.000000	169.000000	8.000000	13.
max	36919.000000	13473.000000	11817.000000	17414.000000	2547.000000	1095.000000	19.000000	75.

In [33]:
```
sns.pairplot(x)
```

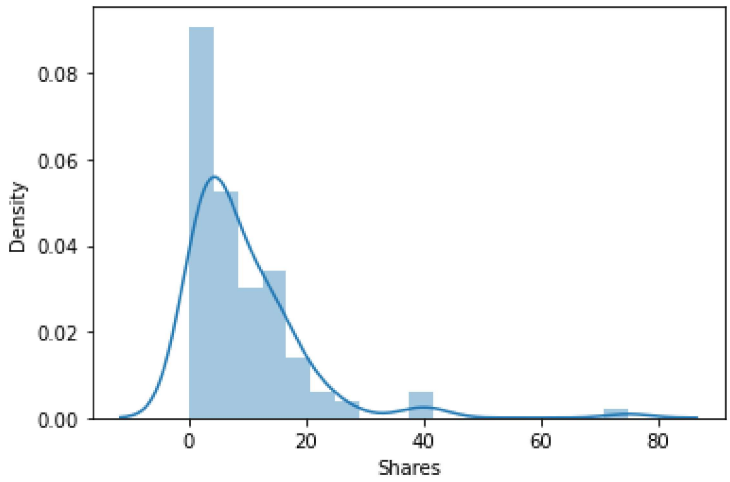
Out[33]:
```
<seaborn.axisgrid.PairGrid at 0x22333373c70>
```



In [34]:
```
sns.distplot(x['Shares'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

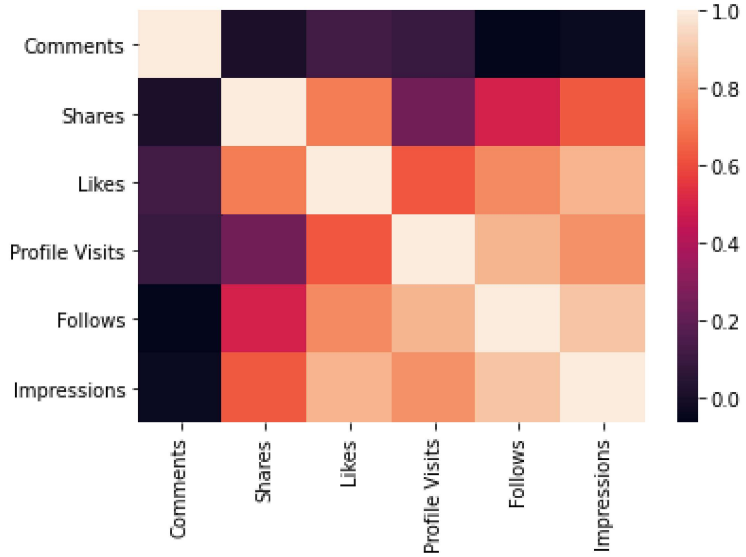
Out[34]: <AxesSubplot:xlabel='Shares', ylabel='Density'>



In [35]:
```
x1=x[['Comments','Shares','Likes','Profile Visits','Follows','Impressions']]
```

In [36]:
```
sns.heatmap(x1.corr())
```

Out[36]: <AxesSubplot:>



In [37]:
```
a=x1[['Comments','Shares','Likes','Profile Visits','Follows']]
b=x1['Impressions']
```

In [38]:
```
from sklearn.model_selection import train_test_split
a_train,a_test, b_train, b_test=train_test_split(a,b,test_size=0.3)
```

In [39]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(a_train,b_train)
```

Out[39]:
```
LinearRegression()
```

In [40]:
```
print(lr.intercept_)
```

```
823.3078249433393
```

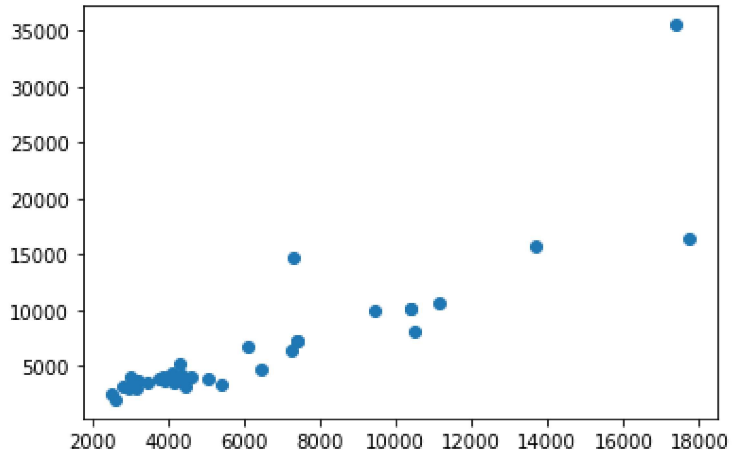
In [41]:
```
coeff=pd.DataFrame(lr.coef_,a.columns,columns=['Co-efficient'])
coeff
```

Out[41]:

	Co-efficient
Comments	-24.223180
Shares	-17.548712
Likes	18.463447
Profile Visits	-3.481187
Follows	111.257878

In [42]:
```
prediction=lr.predict(a_test)
plt.scatter(b_test,prediction)
```

Out[42]:
```
<matplotlib.collections.PathCollection at 0x2233a13bee0>
```



In [43]:
```
lr.score(a_test,b_test)
```

Out[43]:
```
0.24553761848396038
```

In [44]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [45]:
```
rr=Ridge(alpha=10)
rr.fit(a_train,b_train)
```

Out[45]:
```
Ridge(alpha=10)
```

In [46]:
```
rr.score(a_test,b_test)
```

Out[46]:
```
0.24628293331475137
```

In [47]:
```
la=Lasso(alpha=10)
la.fit(a_train,b_train)
```

Out[47]:
```
Lasso(alpha=10)
```

In [48]:
```
la.score(a_test,b_test)
```

Out[48]:
```
0.2464987229004253
```

In [51]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(a_train,b_train)
```

Out[51]:
```
ElasticNet()
```

In [53]:
```
en.coef_
```

Out[53]:
```
array([-23.52115573, -16.91781346,  18.44583646,  -3.39752581,
       110.97476804])
```

In [55]:
```
en.predict(a_test)
```

Out[55]:
```
array([ 4234.94584711, 15637.14139154,  4702.39716834,  5260.82935877,
        4032.52939008,  7161.72588189,  2470.37598897,  3871.11148693,
        9975.56049157,  3491.13249867,  1995.82586268,  4061.40686295,
        3411.35000234,  3609.86404238,  8113.55351306,  3794.80825315,
        6742.33928196,  2982.10800203, 16454.19693378,  3611.78372544,
        3525.41553903,  4034.39889288,  3649.05235307,  7161.72588189,
       10605.69848598,  4302.79677314,  3810.50518271,  3173.01981794,
       14705.06434119,  3140.28330171, 10059.58141832, 35473.99187608,
        3939.13651653, 10059.58141832,  2994.26220142,  6366.04396264])
```

In [56]:
```
en.intercept_
```

Out[56]:
```
816.8015547953592
```

In [57]:
```
en.score(a_test,b_test)
```

Out[57]:
```
0.24866267001336173
```

In [58]:
```
from sklearn import metrics
```

In [59]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(b_test,prediction))
```

```
Mean Absolute Error 1329.2569074940563
```

In [60]:
```
print("Mean Squared Error",metrics.mean_squared_error(b_test,prediction))
```

```
Root Mean Squared Error 11417227.25181067
```

In [61]:
```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(b_test,prediction))
```

```
Root Mean Squared Error 3378.9387759784386
```

In [ ]:
```

```