Deena 20104016

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
```

Problem Statement

LINEAR REGRESSION

In [2]:
```
a = pd.read_csv("wine.csv ")
```

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	

HEAD

In [3]:

Out[3]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Fol
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	

Data Cleaning and Preprocessing

In [4]:

Out[4]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Fol
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	

In [5]:

Out[5]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comn
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.0(
mean	5703.991597	2475.789916	1887.512605	1078.100840	171.092437	153.310924	6.6(
std	4843.780105	1489.386348	1884.361443	2613.026132	289.431031	156.317731	3.5
min	1941.000000	1133.000000	116.000000	0.000000	9.000000	22.000000	0.0(
25%	3467.000000	1945.000000	726.000000	157.500000	38.000000	65.000000	4.0(
50%	4289.000000	2207.000000	1278.000000	326.000000	74.000000	109.000000	6.0(
75%	6138.000000	2602.500000	2363.500000	689.500000	196.000000	169.000000	8.0(
max	36919.000000	13473.000000	11817.000000	17414.000000	2547.000000	1095.000000	19.0(

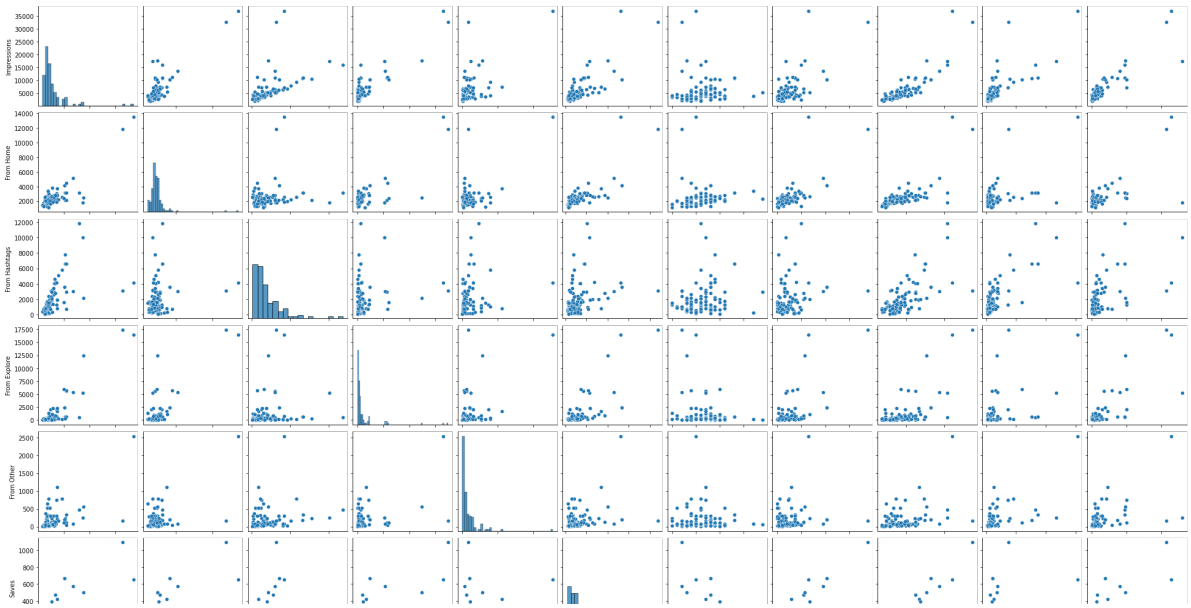
To display heading

In [6]:

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

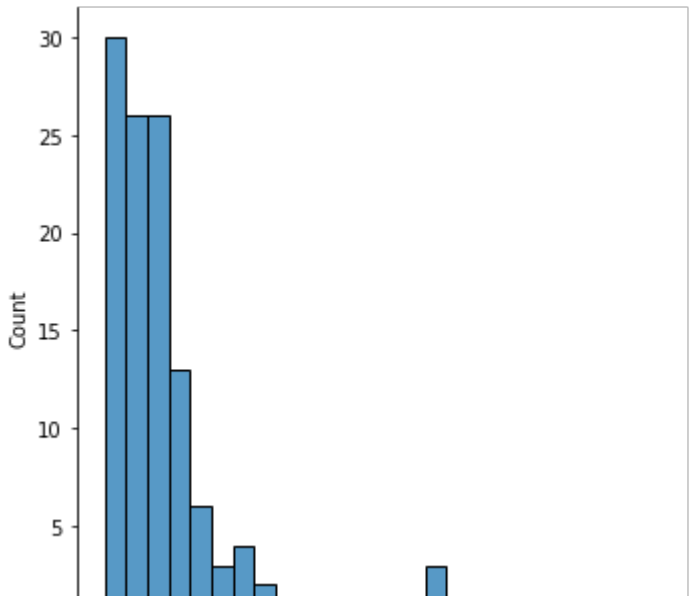
In [7]:

Out[7]: <seaborn.axisgrid.PairGrid at 0x1d99ba8aaf0>



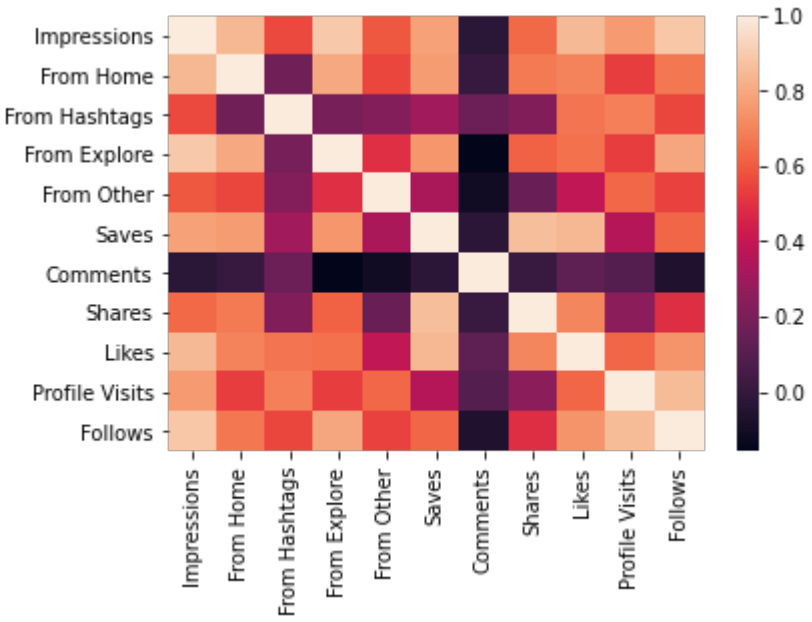
In [8]:

Out[8]: <seaborn.axisgrid.FacetGrid at 0x1d9a0844910>



In [9]:

Out[9]: <AxesSubplot:>



TO TRAIN THE MODEL - MODEL BUILDING

In [10]:
```
x = a[['Comments']]
```

In [11]:
```
# to split my dataset into training and test data
from sklearn.model_selection import train_test_split
```

In [12]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

Out[12]: LinearRegression()

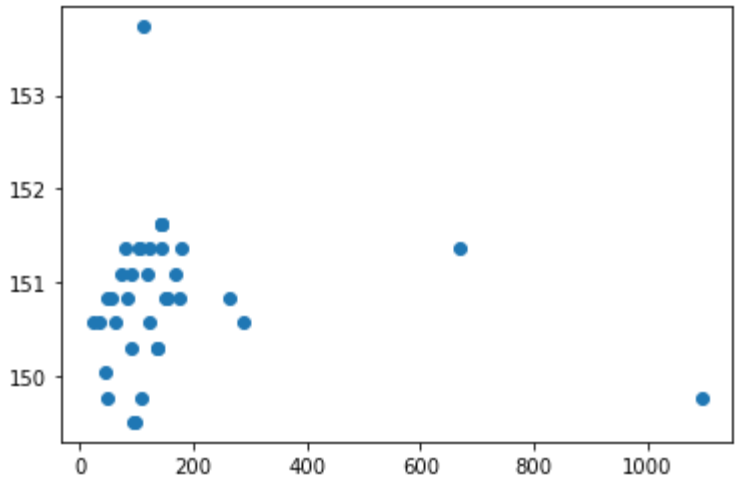
In [13]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
```

Out[13]:

	Co-efficient
Comments	0.264266

In [14]:
```
prediction= lr.predict(x_test)
```

Out[14]: <matplotlib.collections.PathCollection at 0x1d9a22e6f40>



In [15]:

Out[15]: -0.002385850633604303

LASSO & RIDGE

In [16]:

In [17]:
```
rr=Ridge(alpha=10)
```

Out[17]: Ridge(alpha=10)

In [18]:

Out[18]: -0.0023781758046372836

In [19]:
```
la=Lasso(alpha=10)
```

Out[19]: Lasso(alpha=10)

In [20]:

Out[20]: -0.0015057484609570793

In [21]:
```
from sklearn.linear_model import ElasticNet
a=ElasticNet()
```

Out[21]: ElasticNet()

In [22]:
```
print(a.coef_)
print(a.intercept_)
print(a.score(x_test,y_test))
```

```
[0.22058211]
149.5441600585825
-0.0022379296441388252
[150.64707062 151.30881696 151.08823485 150.64707062 150.64707062
 151.08823485 151.30881696 150.42648851 151.30881696 150.64707062
 150.42648851 150.86765274 151.52939908 151.30881696 151.30881696
 151.30881696 150.86765274 150.64707062 149.76474217 151.30881696
 149.98532429 153.29405598 151.52939908 150.86765274 151.52939908
 150.86765274 149.76474217 150.86765274 150.2059064  149.98532429
 149.98532429 151.08823485 150.42648851 151.08823485 150.86765274
 150.86765274]
```

In [23]:
```
from sklearn import metrics
print(" Mean Absolute Error :",metrics.mean_absolute_error(y_test,prediction))
print(" Mean Squared Error :",metrics.mean_squared_error(y_test,prediction))
```

```
 Mean Absolute Error : 91.45602265538054
 Mean Squared Error : 36453.40348387815
 Root Mean Absolute Error : 9.563264225952379
```