Problem Statement

Linear Regression

Import Libraries

In [1]:

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

In [2]:

```
a=pd.read_csv("insta.csv")
a
```

Out[2]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follows
0	3920	2586	1028	619	56	98	9	5	162	35	2
1	5394	2727	1838	1174	78	194	7	14	224	48	10
2	4021	2085	1188	0	533	41	11	1	131	62	12

To display top 10 rows

In [3]:

```
c=a.head(15)
c
```

Out[3]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits
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
10	2218	1597	411	162	15	28	6	3	81	29
11	3234	2414	476	185	75	122	8	14	151	15
12	4344	2168	1274	673	40	119	7	11	162	8

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits
13	3216	2524	212	201	223	121	5	5	142	20
14	9453	2525	5799	208	794	100	6	10	294	181

To find Missing values

In [4]:

```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Impressions     15 non-null     int64 
 1   From Home       15 non-null     int64 
 2   From Hashtags   15 non-null     int64 
 3   From Explore    15 non-null     int64 
 4   From Other      15 non-null     int64 
 5   Saves           15 non-null     int64 
 6   Comments        15 non-null     int64 
 7   Shares          15 non-null     int64 
 8   Likes           15 non-null     int64 
 9   Profile Visits  15 non-null     int64 
 10  Follows         15 non-null     int64 
 11  Caption         15 non-null     object
 12  Hashtags        15 non-null     object
dtypes: int64(11), object(2)
memory usage: 1.6+ KB
```

To display summary of statistics

In [5]:

```
a.describe()
```

Out[5]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	C
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	11
mean	5703.991597	2475.789916	1887.512605	1078.100840	171.092437	153.310924	
std	4843.780105	1489.386348	1884.361443	2613.026132	289.431031	156.317731	
min	1941.000000	1133.000000	116.000000	0.000000	9.000000	22.000000	
25%	3467.000000	1945.000000	726.000000	157.500000	38.000000	65.000000	
50%	4289.000000	2207.000000	1278.000000	326.000000	74.000000	109.000000	
75%	6138.000000	2602.500000	2363.500000	689.500000	196.000000	169.000000	
max	36919.000000	13473.000000	11817.000000	17414.000000	2547.000000	1095.000000	1

To display column heading

In [6]:

```
a.columns
```

Out[6]:

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

Pairplot

In [7]:

```
s=a.dropna(axis=1)
s
```

Out[7]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits
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
118	36919	13473	4176	16444	2547	653	5	26	443	61

119 rows × 13 columns

In [8]:

```
s.columns
```

Out[8]:

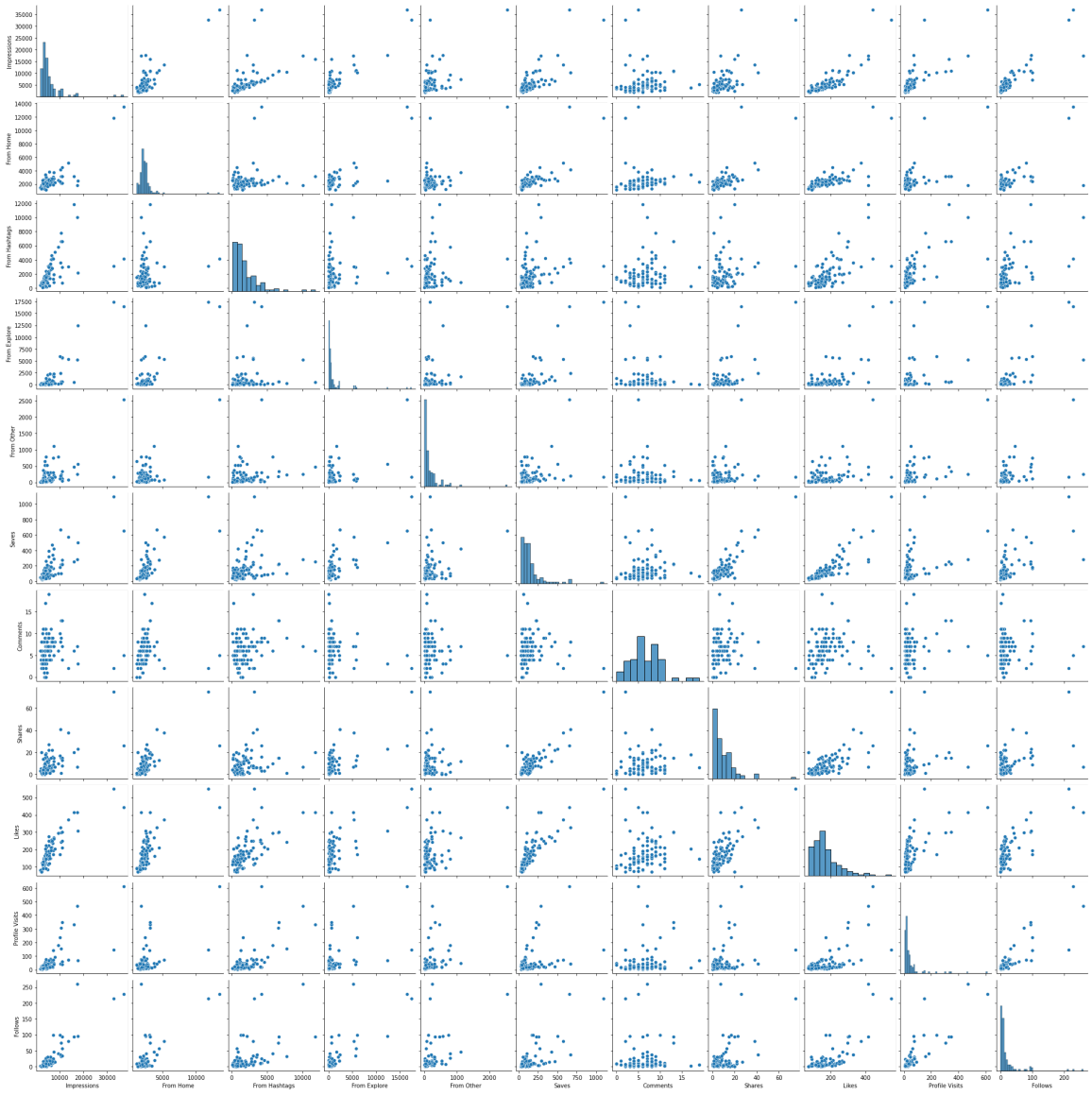
```
Index(['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visi
ts',
       'Follows', 'Caption', 'Hashtags'],
      dtype='object')
```

In [9]:

```
sns.pairplot(a)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x1e640a0c5b0>
```



Distribution Plot

Loading [MathJax]/extensions/Safe.js

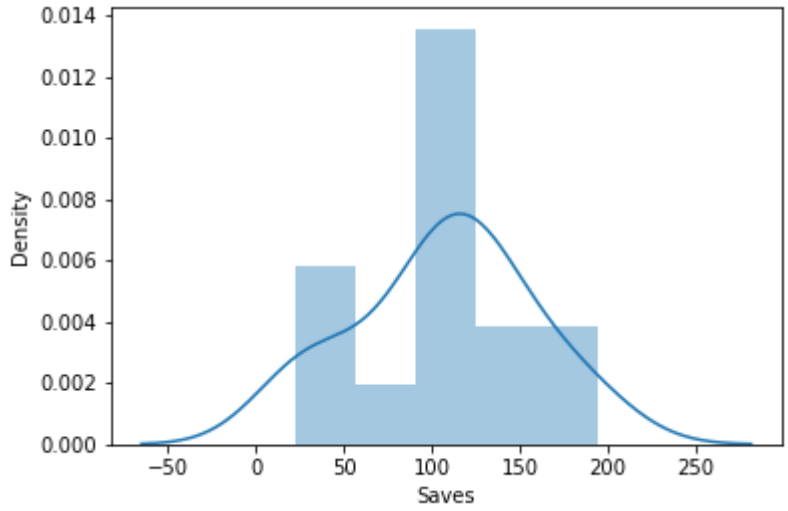
In [10]:

```
sns.distplot(c['Saves'])
```

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

Out[10]:

```
<AxesSubplot:xlabel='Saves', ylabel='Density'>
```



Correlation

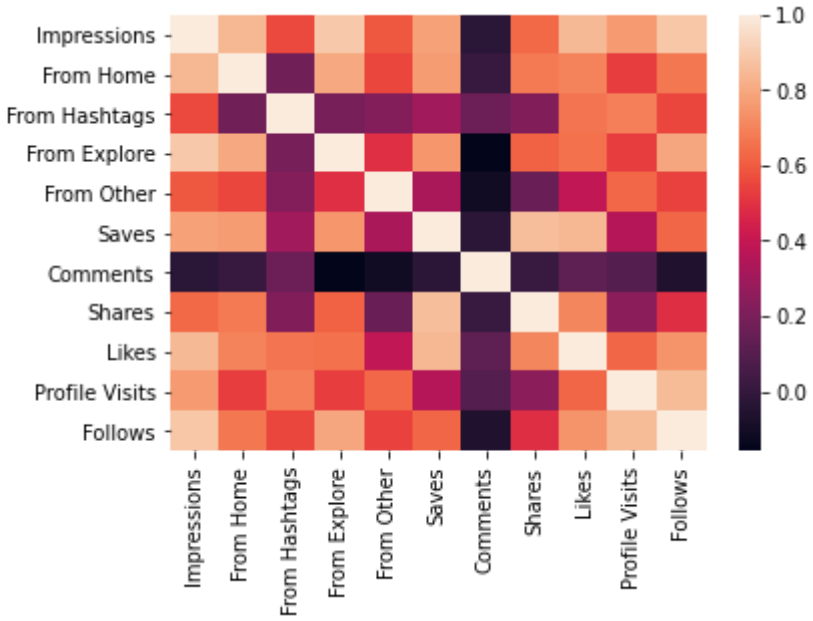
Train the model - Model Building

In [11]:

```
b=a[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows', 'Caption', 'Hashtags']]
sns.heatmap(b.corr())
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
g=c[['Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows']]
h=c['Saves']
```

To split dataset into training end test

In [13]:

```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [14]:

```
from sklearn.linear_model import LinearRegression
```

In [15]:

```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
-108.33670499199668
```

Coeffecient

In [17]:

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

Out[17]:

	Co-effecient
Comments	-21.428831
Shares	-21.979625
Likes	4.071678
Profile Visits	6.310581
Follows	-45.491049

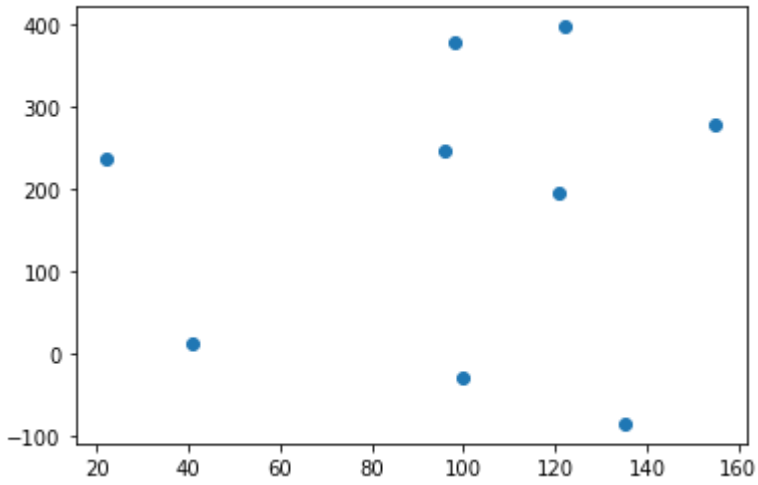
Best Fit line

In [18]:

```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[18]:

```
<matplotlib.collections.PathCollection at 0x1e647249b20>
```



To find score

In [19]:

```
print(lr.score(g_test,h_test))
```

```
-20.02227741333439
```

Import Lasso and ridge

In [20]:

```
from sklearn.linear_model import Ridge,Lasso
```

Ridge

In [21]:

```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[21]:

```
Ridge(alpha=5)
```

In [22]:

```
ri.score(g_test,h_test)
```

Out[22]:

```
-2.7837551858507816
```

In [23]:

```
ri.score(g_train,h_train)
```

Out[23]:

```
0.9782919976056234
```

Lasso

In [24]:

```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

Out[24]:

```
Lasso(alpha=6)
```

In [25]:

```
l.score(g_test,h_test)
```

Out[25]:

```
-2.7437752963491495
```

In [26]:

```
ri.score(g_train,h_train)
```

Out[26]:

```
0.9782919976056234
```

ElasticNet

In [27]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(g_train,h_train)
```

Out[27]:

```
ElasticNet()
```

Coeffecient,intercept

In [28]:

```
print(e.coef_)
```

```
[ 2.25817442  0.07514746  1.19538143  1.10013117 -4.08410165]
```

In [29]:

```
print(e.intercept_)
```

```
-100.26318522870334
```

Prediction

In [30]:

```
d=e.predict(g_test)
d
```

Out[30]:

```
array([293.0709474 ,  30.55523311,  46.37010063, 127.22100326,
        67.16124155, 151.4286125 ,  86.81380353, 144.42430055,
       100.44576035])
```

In [31]:

```
print(e.score(g_test,h_test))
```

```
-2.697359321973825
```

Evaluation

In [32]:

```
from sklearn import metrics
print("Mean Absolute error:",metrics.mean_absolute_error(h_test,d))
```

```
Mean Absolute error: 57.37318943794954
```

In [33]:

```
print("Mean Squared error:",metrics.mean_squared_error(h_test,d))
```

```
Mean Squared error: 6059.104646898833
```

In [34]:

```
print("Mean Squared error:",np.sqrt(metrics.mean_squared_error(h_test,d)))
```

```
Mean Squared error: 77.84025081472203
```

In [ ]:

```
```