Problem Statement:

A real estate agent want to help to predict the house price for regions in USA.He gave us the dataset to work on to use Linear Regression modelCreate a Model that helps him to estimate of what the house would sell for

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("Instagram.csv")
df
```

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
3	4528	2700	621	932	73	172	10	7	213	23	8
4	2518	1704	255	279	37	96	5	4	123	8	0
...	...	...	...	...	...	...	...	...	...	...	...
114	13700	5185	3041	5352	77	573	2	38	373	73	80

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follows
115	5731	1923	1368	2266	65	135	4	1	148	20	18
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

119 rows × 13 columns

In [3]:
```
df.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 [4]:
```
df.head()
```

Out[4]:

	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	i da
1	5394	2727	1838	1174	78	194	7	14	224	48	10	
2	4021	2085	1188	0	533	41	11	1	131	62	12	L m
3	4528	2700	621	932	73	172	10	7	213	23	8	c pr
4	2518	1704	255	279	37	96	5	4	123	8	0	an v

Data cleaning and Pre-Processing

In [5]:
```
df.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 [6]:
```
df.describe()
```

Out[6]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	1
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	
mean	5703.991597	2475.789916	1887.512605	1078.100840	171.092437	153.310924	6.663866	
std	4843.780105	1489.386348	1884.361443	2613.026132	289.431031	156.317731	3.544576	
min	1941.000000	1133.000000	116.000000	0.000000	9.000000	22.000000	0.000000	
25%	3467.000000	1945.000000	726.000000	157.500000	38.000000	65.000000	4.000000	
50%	4289.000000	2207.000000	1278.000000	326.000000	74.000000	109.000000	6.000000	
75%	6138.000000	2602.500000	2363.500000	689.500000	196.000000	169.000000	8.000000	
max	36919.000000	13473.000000	11817.000000	17414.000000	2547.000000	1095.000000	19.000000	

In [7]:
```
df.dropna(axis='columns')
```

Out[7]:

	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
3	4528	2700	621	932	73	172	10	7	213	23	8
4	2518	1704	255	279	37	96	5	4	123	8	0

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follows
...	...	...	...	...	...	...	...	...	...	...	...
114	13700	5185	3041	5352	77	573	2	38	373	73	80
115	5731	1923	1368	2266	65	135	4	1	148	20	18
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

119 rows × 13 columns

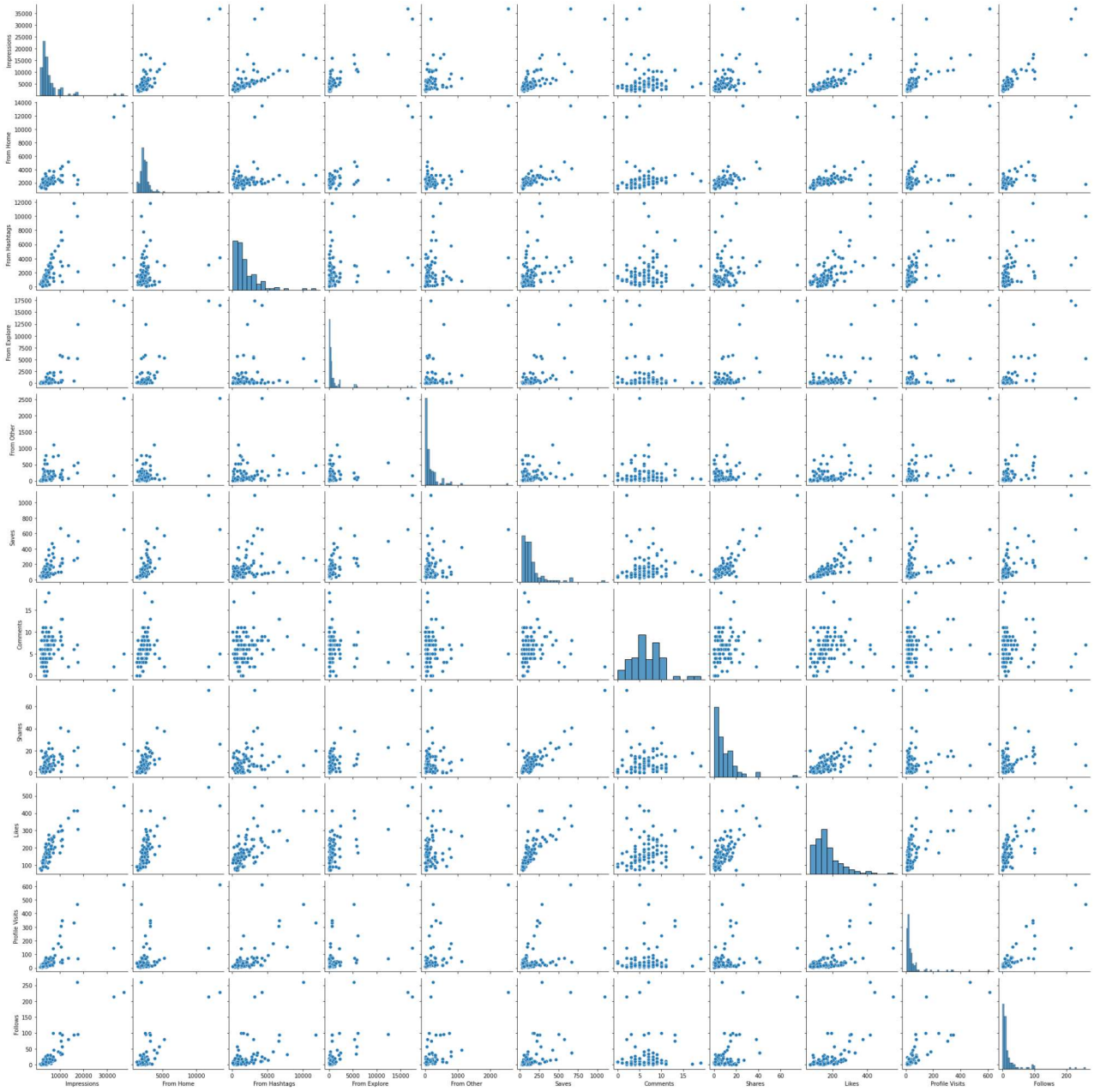
In [8]:
```
df.columns
```

Out[8]:
```
Index(['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows', 'Caption', 'Hashtags'],
      dtype='object')
```

EDA and VISUALIZATION

In [9]:
```
sns.pairplot(df)
```

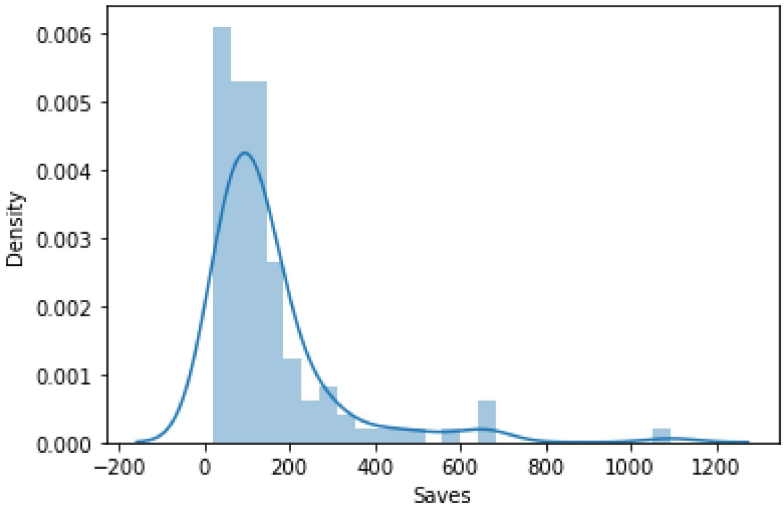
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x185a380afd0>
```



In [10]:
```
sns.distplot(df['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 adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]: <AxesSubplot:xlabel='Saves', ylabel='Density'>

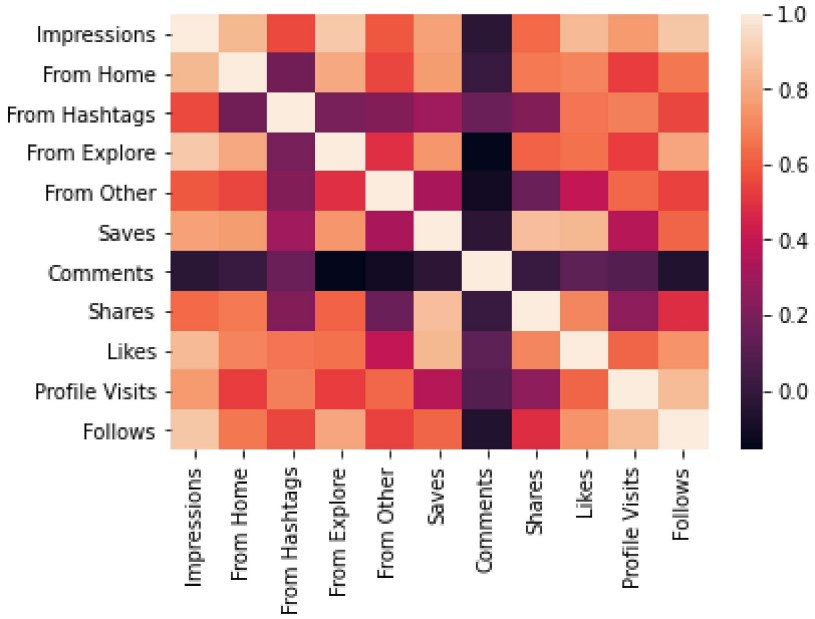


In [11]:
```
df1=df[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows', 'Caption', 'Hashtags']]
```

Plot Using Heat Map

In [12]:
```
sns.heatmap(df1.corr())
```

Out[12]: <AxesSubplot:>



To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [13]:
```
x=df1[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other',  'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows' ]]
y=df1['Saves']
```

To Split my dataset into training and test data

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
lr.intercept_
```

Out[16]:
```
-79.15425540475647
```

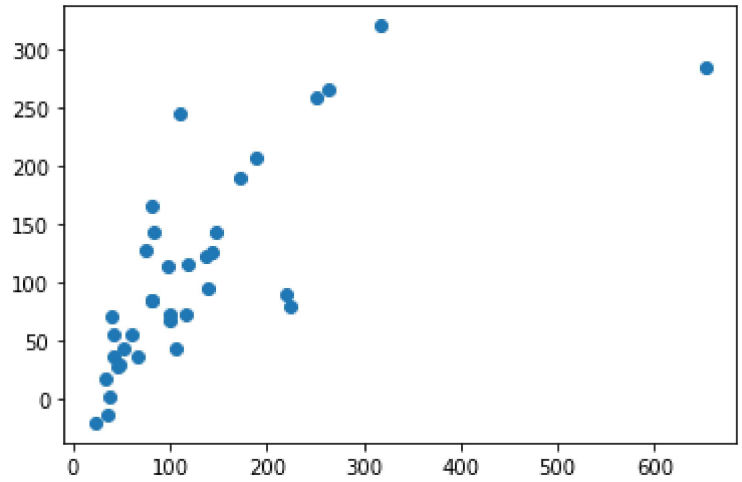
In [17]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
Impressions	-0.035930
From Home	0.031827
From Hashtags	0.034006
From Explore	0.038846
From Other	0.075352
Comments	0.769941
Shares	5.571887
Likes	1.183423
Profile Visits	-1.198582
Follows	1.525062

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

Out[18]:
```
<matplotlib.collections.PathCollection at 0x185aa0aba30>
```



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

Out[19]: 0.521450484906552

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[21]: 0.9219868959172489

In [22]:
```
rr.score(x_test,y_test)
```

Out[22]: 0.5241369935655398

In [23]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:5
30: ConvergenceWarning: Objective did not converge. You might want to increase the numbe
r of iterations. Duality gap: 92321.34562170038, tolerance: 237.3672915662651
  model = cd_fast.enet_coordinate_descent(
```

Out[23]: Lasso(alpha=10)

In [24]:
```
la.score(x_test,y_test)
```

Out[24]: 0.6209265287205321