Type *Markdown* and LaTeX: α^2

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

In [2]:
```
#import dataset
df=pd.read_csv(r"E:\154\5_Instagram data.csv")
df
```

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]:
```
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]:
```
#to display top 5 rows
df.head()
```

Out[4]:

	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

Data cleaning and Pre-Processing

In [5]:
```
#To find null values
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]:
```
# To display summary of statistics
df.describe()
```

Out[6]:

	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	3.54
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(

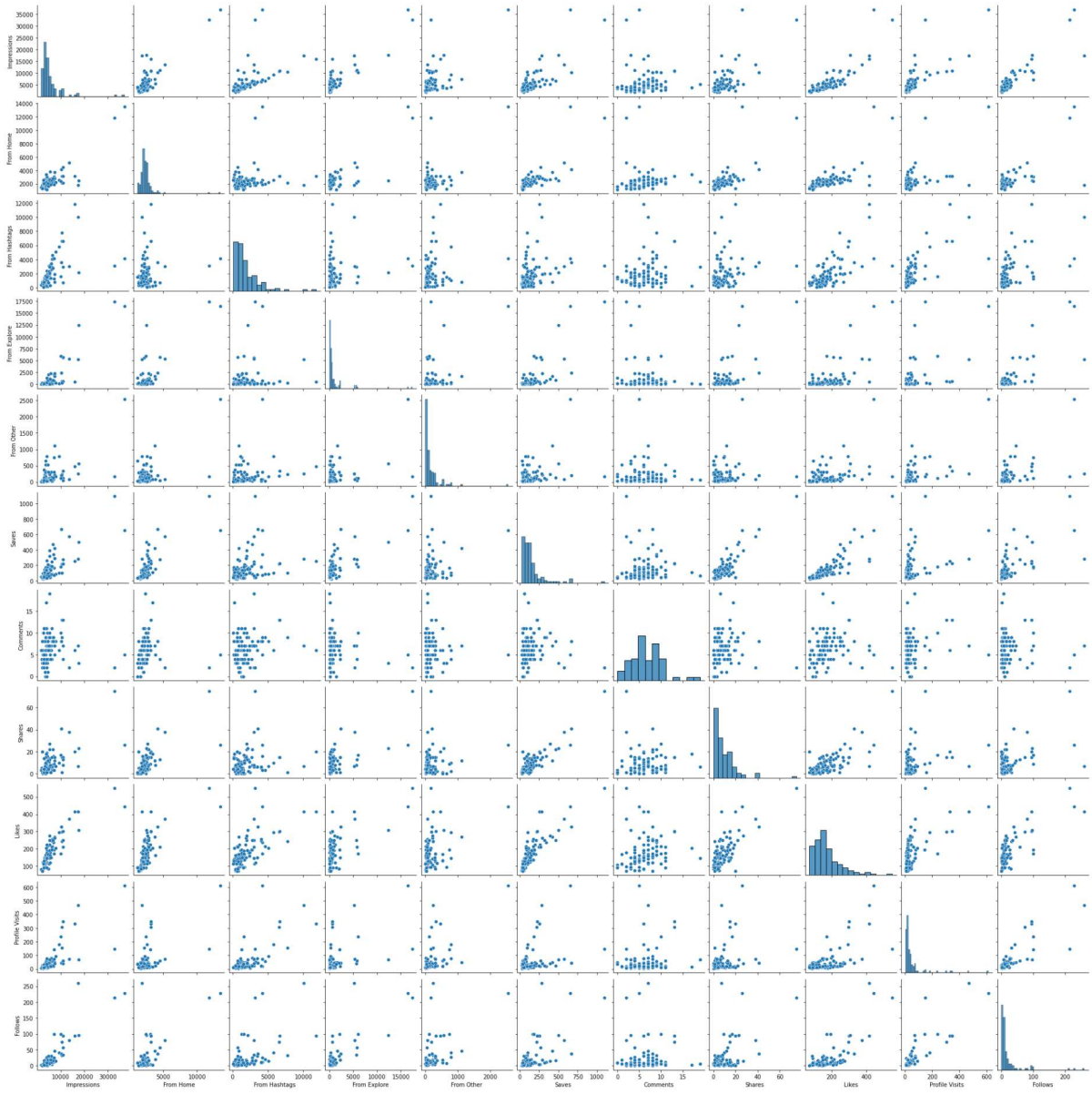
In [7]:
```
#To Display column heading
df.columns
```

Out[7]:
```
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 [8]:
```
sns.pairplot(df)
```

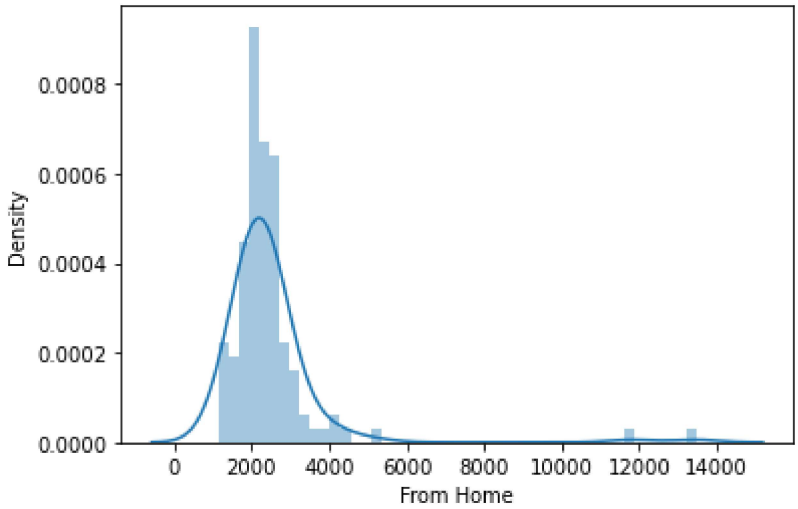
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x244318e87c0>
```



In [9]:
```
sns.distplot(df["From Home"])
```

```
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[9]:
```
<AxesSubplot:xlabel='From Home', ylabel='Density'>
```

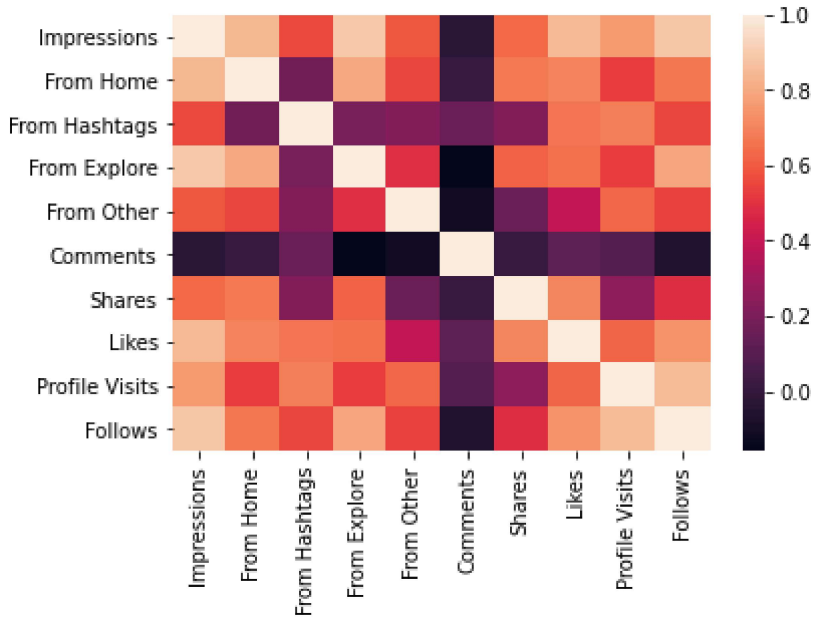


In [10]:
```
df1=df[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows']]
```

Plot Using Heat Map

In [11]:
```
sns.heatmap(df1.corr())
```

Out[11]: <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 [12]:
```
x=df1[['Impressions', 'From Hashtags', 'From Explore',
       'From Other', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows']]
y=df1['From Home']
```

To Split my dataset into training and test data

In [13]:
```
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]:
```
lr.intercept_
```

Out[15]:
```
-130.56537617317554
```

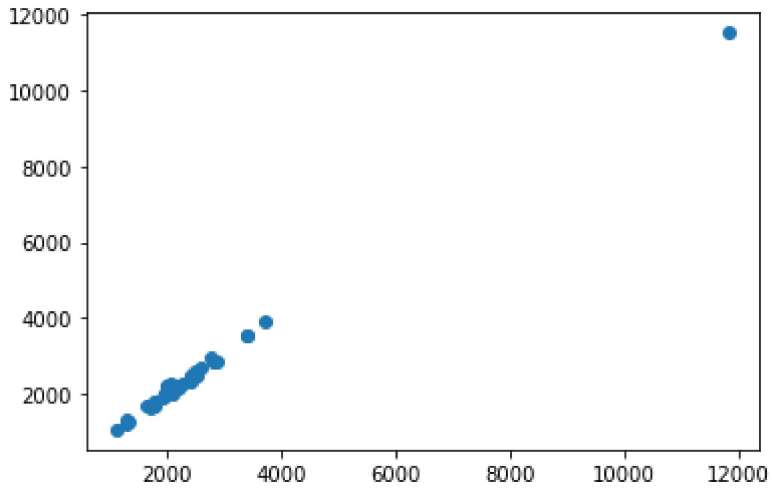
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Impressions	0.940087
From Hashtags	-0.970876
From Explore	-0.941479
From Other	-0.882566
Comments	7.402134
Shares	1.840128
Likes	1.049694
Profile Visits	0.462469
Follows	-1.251851

In [17]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x244381a6b50>
```



In [18]:
```
lr.score(x_test,y_test)
```

Out[18]:
```
0.995614271521736
```

Accuracy

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

Out[19]:
```
0.995614271521736
```

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

Out[20]:
```
0.9898330361605525
```

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

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

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

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

Out[23]:
```
0.995629369768717
```

In [24]:
```
la =Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]:
```
Lasso(alpha=10)
```

In [25]:
```
la.score(x_test,y_test)
```

Out[25]:
```
0.9957989368866552
```

ElasticNet

In [26]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[26]:
```
ElasticNet()
```

In [27]:
```
print(en.coef_)
```

```
[ 0.94021368 -0.97113055 -0.94165971 -0.88515349  6.88273839  1.80496255
  1.05765768  0.47562151 -1.27316315]
```

In [28]:
```
print(en.intercept_)
```

```
-127.9830186117033
```

In [29]:
```
print(en.predict(x_test))
```

```
[ 3890.38398002  2143.03000937  2846.64560506  2202.10596566
  2007.81914117  3551.00851306  1315.06315772  1688.99165644
  1667.66116324  1249.69485704  1688.99165644  2486.79964105
  2167.09204833  1188.35607268  1682.10881587  2716.36488752
  2164.80425787  1806.535467    2227.91773474  1767.72139457
  3551.00851306  2017.04244846  2276.76292166  1938.08441391
  1609.03586941  2599.43684662 11535.3440173   1882.17761047
  2257.40510971  2330.28978737  2828.07430472  1056.05270863
  2499.49841411  2236.15777813  2982.87566399  2498.21891937]
```

In [30]:
```
print(en.score(x_test,y_test))
```

```
0.9956820034102397
```

Evaluation Metrics

In [31]:
```
from sklearn import metrics
```

In [32]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 90.29060193236711
```

In [33]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 12477.17079948006
```

In [34]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root Mean Absolute Error: 111.70125692882806
```

In [35]:
```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root Mean Absolute Error: 111.70125692882806
```

In [ ]:
```

```