

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

```
In [3]: data=pd.read_csv("instagram_reach.csv")
```

```
In [4]: data.head()
```

Out[4]:

	Unnamed: 0	S.No	USERNAME	Caption	Followers	
0	0	1	mikequindazzi	Who are #DataScientist and what do they do? >>...	1600	#MachineLearning
1	1	2	drgorillapaints	We all know where it's going. We just have to ...	880	#deck .#mac #macintosh#sayhe
2	2	3	aitrading_official	Alexander Barinov: 4 years as CFO in multinati...	255	#whoiswho #aitrading #ai #aitr
3	3	4	opensourcedworkplace	sfad	340	#iot #cre#workplace #CDO #bigd
4	4	5	crea.vision	Ever missed a call while your phone was chargi...	304	#instamachinelearning #instabi

```
In [5]: data.isnull().sum()
```

Out[5]:

Unnamed: 0	0
S.No	0
USERNAME	0
Caption	6
Followers	0
Hashtags	0
Time since posted	0
Likes	0
dtype: int64	

```
In [6]: data.duplicated().sum()
```

Out[6]: 0

```
In [7]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   Unnamed: 0            100 non-null   int64  
1   S.No                  100 non-null   int64  
2   USERNAME              100 non-null   object  
3   Caption               94 non-null    object  
4   Followers              100 non-null   int64  
5   Hashtags              100 non-null   object  
6   Time since posted     100 non-null   object  
7   Likes                 100 non-null   int64  
dtypes: int64(4), object(4)
memory usage: 6.4+ KB
```

```
In [8]: data.shape
```

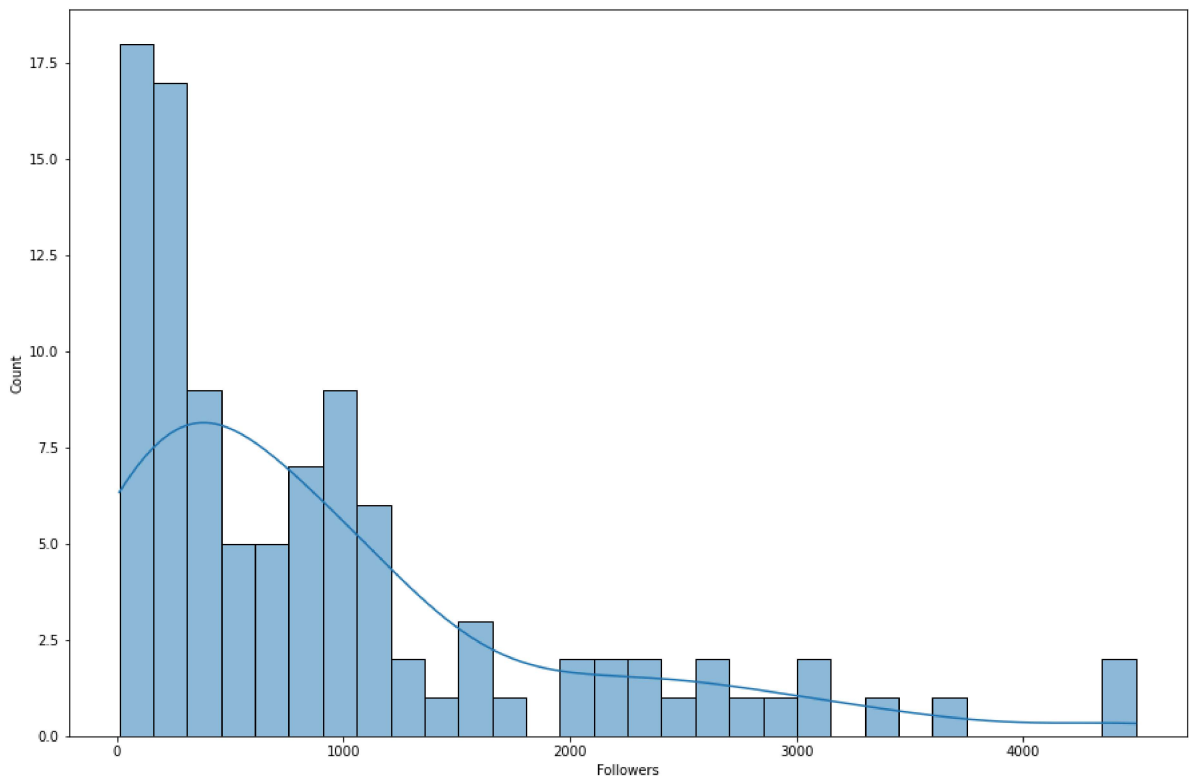
```
Out[8]: (100, 8)
```

```
In [9]: data.describe().T
```

```
Out[9]:
```

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	100.0	8.94	6.639064	0.0	4.00	8.0	12.25	26.0
S.No	100.0	16.24	7.384286	1.0	10.75	16.5	22.25	30.0
Followers	100.0	961.96	1014.625670	11.0	252.75	612.0	1197.00	4496.0
Likes	100.0	46.48	55.086980	8.0	19.00	29.0	46.00	349.0

```
In [10]: plt.figure(figsize=(15,10))
sns.histplot(data=data, x='Followers', bins=30, kde=True)
plt.show()
```



```
In [11]: from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

data['USERNAME']=le.fit_transform(data['USERNAME'])
data['Caption']=le.fit_transform(data['Caption'])
data['Hashtags']=le.fit_transform(data['Hashtags'])
```

```
In [12]: data['Time since posted'] = data['Time since posted'].str.extract('(\d+)').ast
```

```
In [13]: data
```

```
Out[13]:
```

	Unnamed: 0	S.No	USERNAME	Caption	Followers	Hashtags	Time since posted	Likes
0	0	1	64	86	1600	17	11	139
1	1	2	29	82	880	97	2	23
2	2	3	3	13	255	87	2	25
3	3	4	71	89	340	65	3	49
4	4	5	23	30	304	64	3	30
...
95	8	19	63	6	614	29	3	31
96	9	21	30	24	450	88	3	42
97	10	22	31	83	182	36	3	10
98	11	24	55	85	2039	62	3	222
99	12	25	73	62	741	68	3	109

100 rows × 8 columns

```
In [14]: X=data.drop(labels=['S.No','Time since posted','Likes'],axis=1)
```

```
In [15]: X
```

```
Out[15]:
```

	Unnamed: 0	USERNAME	Caption	Followers	Hashtags
0	0	64	86	1600	17
1	1	29	82	880	97
2	2	3	13	255	87
3	3	71	89	340	65
4	4	23	30	304	64
...
95	8	63	6	614	29
96	9	30	24	450	88
97	10	31	83	182	36
98	11	55	85	2039	62
99	12	73	62	741	68

100 rows × 5 columns

```
In [16]: print(data.columns)
```

```
Index(['Unnamed: 0', 'S.No', 'USERNAME', 'Caption', 'Followers', 'Hashtags',  
      'Time since posted', 'Likes'],  
      dtype='object')
```

```
In [17]: print(data.columns)
```

```
Index(['Unnamed: 0', 'S.No', 'USERNAME', 'Caption', 'Followers', 'Hashtags',  
      'Time since posted', 'Likes'],  
      dtype='object')
```

```
In [18]: y = data[['Time since posted', 'Likes']]
```

```
In [19]: y
```

```
Out[19]:
```

	Time since posted	Likes
0	11	139
1	2	23
2	2	25
3	3	49
4	3	30
...
95	3	31
96	3	42
97	3	10
98	3	222
99	3	109

100 rows × 2 columns

```
In [20]: from sklearn.model_selection import train_test_split  
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.25)
```

```
In [21]: X_train
```

```
Out[21]:
```

	Unnamed: 0	USERNAME	Caption	Followers	Hashtags
89	2	47	94	1196	11
40	1	62	79	258	82
54	15	43	16	2785	49
4	4	23	30	304	64
6	6	56	22	259	92
...
72	10	53	94	955	75
3	3	71	89	340	65
88	1	20	58	1158	56
77	4	58	29	631	54
91	4	85	53	971	46

75 rows × 5 columns

```
In [22]: y_train
```

```
Out[22]:
```

	Time since posted	Likes
89	2	16
40	2	24
54	7	65
4	3	30
6	3	35
...
72	2	20
3	3	49
88	2	29
77	2	55
91	2	46

75 rows × 2 columns

```
In [23]: from sklearn.preprocessing import StandardScaler  
scaler=StandardScaler()
```

```
In [24]: X_train_scaled=scaler.fit_transform(X_train)
```

```
In [25]: X_test_scaled=scaler.transform(X_test)
```

```
In [26]: from sklearn.linear_model import LinearRegression,Ridge,Lasso,ElasticNet
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score,mean_absolute_error, mean_squared_error
```

```
In [27]: def evaluate_model(true, predict):
r2=r2_score(true, predict)
mae=mean_absolute_error(true,predict)
mse=mean_squared_error(true,predict)

return r2, mae,mse,
```

```
In [28]: models={

    'LinearRegression':LinearRegression(),
    'Lasso':Lasso(),
    'Ridge':Ridge(),
    'Elasticnet':ElasticNet(),
    'Randomforest':RandomForestRegressor()

}
```

```
In [29]: for i in range(len(models)):
    model=list((models.values()))[i]
    print(model)

r2_list=[]
```

```
LinearRegression()
Lasso()
Ridge()
ElasticNet()
RandomForestRegressor()
```

```
In [30]: from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

```
In [31]: def evaluate_model(true, predict):  
         r2=r2_score(true, predict)  
         mae=mean_absolute_error(true,predict)  
         mse=mean_squared_error(true,predict)  
  
         return r2, mae,mse,
```



```
In [32]: for i in range(len(models)):
        model=list(models.values())[i]

        model.fit(X_train,y_train)

        #make_prediction
        y_pred=model.predict(X_test)

        #this is for the validaiton
        R2,MAE,MSE=evaluate_model(y_test,y_pred)

        print("model training performance",model)
        print("MSE:", MSE)
        print("MAE:",MAE)
        print("R2 SCORE:",R2)

        r2_list.append(R2)

        print("="*40)
        print("\n")
```

```
model training performance LinearRegression()  
MSE: 2378.288395843672  
MAE: 19.51379397956818  
R2 SCORE: -0.06604153058052276  
=====
```

```
model training performance Lasso()  
MSE: 2377.923974921195  
MAE: 19.523872989715127  
R2 SCORE: -0.05613652992760665  
=====
```

```
model training performance Ridge()  
MSE: 2378.281817541027  
MAE: 19.513765128850377  
R2 SCORE: -0.06603836376172012  
=====
```

```
model training performance ElasticNet()  
MSE: 2377.8522624838133  
MAE: 19.517082243149638  
R2 SCORE: -0.060924639498182276  
=====
```

```
model training performance RandomForestRegressor()  
MSE: 2443.547738  
MAE: 18.979  
R2 SCORE: 0.017484435664138875  
=====
```

In [33]:

```
r2_list
```

Out[33]: [-0.06604153058052276,
-0.05613652992760665,
-0.06603836376172012,
-0.060924639498182276,
0.017484435664138875]

RandomForestRegressor has high r2 value and low mae and mse than other models, so it is best for our project

