Data:

The data is of the housing price of City Delhi . In this Dataset consists of 12 columns and 1259 rows. 6 of the features are numerical valued and rest are categorical.

Dataset are:

Area: Area of the house in Sqft
BHK: no of bedrooms in the house
Bathroom : no of bathrooms in the house
Furnishing : Categorical data house is furnished or not
Locality : Categorical data of Location of the house
Parking : how many parting are available
Price : price of the house
Status : Categorical data of under construction or constructed
Transaction : Categorical data of new property or resale
Type : Categorical data type means apartment or individual house or etc
Per_Sqft : price / area

```
In [1]: import pandas as pd
```

```
In [2]: data=pd.read_csv("MagicBricks.csv")
```

```
In [3]: data.head(10)
```

Out[3]:

	Area	BHK	Bathroom	Furnishing	Locality	Parking	Price	Status	Transaction	Type	Per_Sqft
0	800.0	3	2.0	Semi-Furnished	Rohini Sector 25	1.0	6500000	Ready_to_move	New_Property	Builder_Floor	NaN
1	750.0	2	2.0	Semi-Furnished	J R Designers Floors, Rohini Sector 24	1.0	5000000	Ready_to_move	New_Property	Apartment	6667.0
2	950.0	2	2.0	Furnished	Citizen Apartment, Rohini Sector 13	1.0	15500000	Ready_to_move	Resale	Apartment	6667.0
3	600.0	2	2.0	Semi-Furnished	Rohini Sector 24	1.0	4200000	Ready_to_move	Resale	Builder_Floor	6667.0
4	650.0	2	2.0	Semi-Furnished	Rohini Sector 24 carpet area 650 sqft status R...	1.0	6200000	Ready_to_move	New_Property	Builder_Floor	6667.0
5	1300.0	4	3.0	Semi-Furnished	Rohini Sector 24	1.0	15500000	Ready_to_move	New_Property	Builder_Floor	6667.0
6	1350.0	4	3.0	Semi-Furnished	Rohini Sector 24	1.0	10000000	Ready_to_move	Resale	Builder_Floor	6667.0
7	650.0	2	2.0	Semi-Furnished	Delhi Homes, Rohini Sector 24	1.0	4000000	Ready_to_move	New_Property	Apartment	6154.0
8	985.0	3	3.0	Unfurnished	Rohini Sector 21	1.0	6800000	Almost_ready	New_Property	Builder_Floor	6154.0
9	1300.0	4	4.0	Semi-Furnished	Rohini Sector 22	1.0	15000000	Ready_to_move	New_Property	Builder_Floor	6154.0

```
In [4]: data.dtypes
```

Out[4]:

```
Area           float64
BHK              int64
Bathroom       float64
Furnishing      object
Locality        object
Parking        float64
Price            int64
Status          object
Transaction     object
Type            object
Per_Sqft       float64
dtype: object
```

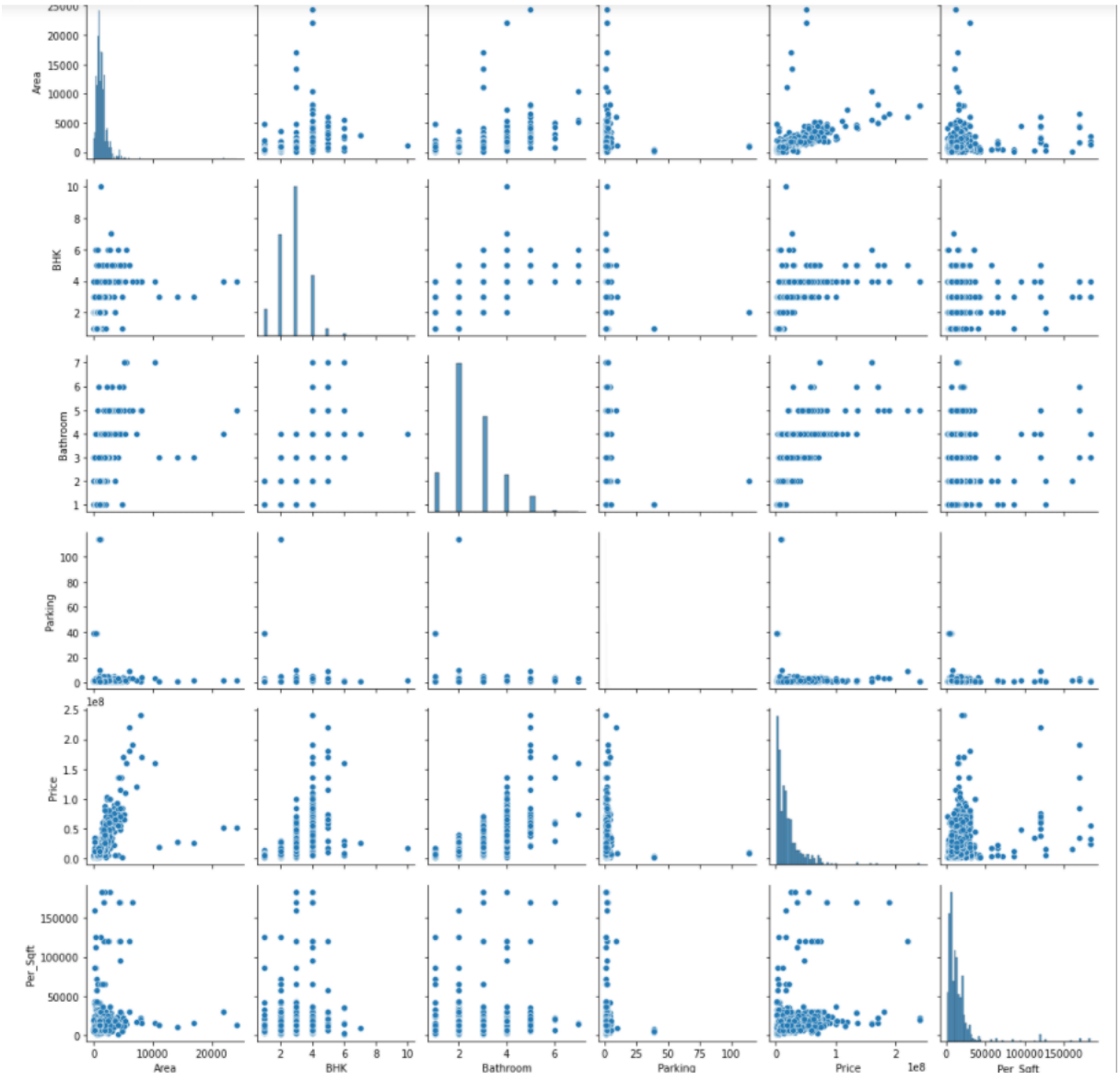
In [5]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:
```
data.describe()
```

Out[6]:

	Area	BHK	Bathroom	Parking	Price	Per_Sqft
count	1259.000000	1259.000000	1257.000000	1226.000000	1.259000e+03	1018.000000
mean	1466.452724	2.796664	2.556086	1.935563	2.130670e+07	15690.136542
std	1568.055040	0.954425	1.042220	6.279212	2.560115e+07	21134.738568
min	28.000000	1.000000	1.000000	1.000000	1.000000e+06	1259.000000
25%	800.000000	2.000000	2.000000	1.000000	5.700000e+06	6364.000000
50%	1200.000000	3.000000	2.000000	1.000000	1.420000e+07	11291.500000
75%	1700.000000	3.000000	3.000000	2.000000	2.550000e+07	18000.000000
max	24300.000000	10.000000	7.000000	114.000000	2.400000e+08	183333.000000

In [7]:
```
sns .pairplot(data)
```



In [10]:
```
%matplotlib inline
sns.set_style('white')
sns.set_context('talk')
sns.set_palette('dark')

one_hot_encode_cols = X.dtypes[X.dtypes == np.object]
one_hot_encode_cols = one_hot_encode_cols.index.tolist()


for col in one_hot_encode_cols:
    X[col] = pd.Categorical(X[col])
    X_poly = pd.Categorical(X_poly[col])

# Do the one hot encoding
X = pd.get_dummies(X, columns=one_hot_encode_cols)
X_poly = pd.get_dummies(X_poly, columns=one_hot_encode_cols)
```

In [11]:
```
X.head(5)
```

Out[11]:

In [12]:
```
from sklearn.model_selection import train_test_split

train_x,test_x, train_y,test_y = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [13]:
```
train_x.dtypes
```

Out[13]:
```
Area                          float64
BHK                             int64
Bathroom                      float64
Parking                       float64
Furnishing_Furnished            uint8
                               ...
Status_Ready_to_move            uint8
Transaction_New_Property        uint8
Transaction_Resale              uint8
Type_Apartment                  uint8
Type_Builder_Floor              uint8
Length: 366, dtype: object
```

In [14]:
```
scols = ['Area','BHK','Bathroom','Parking']
```

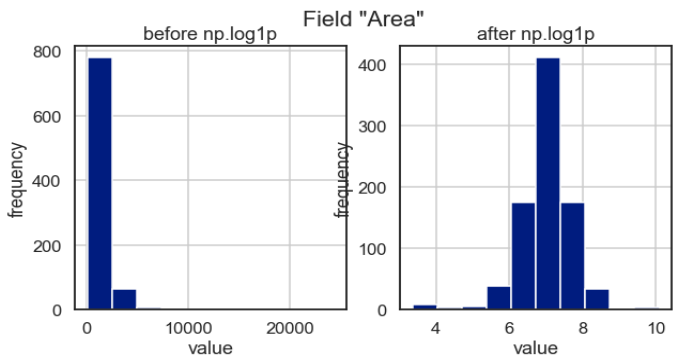
In [15]:
```
scols
```

Out[15]:
```
['Area', 'BHK', 'Bathroom', 'Parking']
```

In [20]:
```
skew_limit = 0.55
skew_vals = train_x[scols].skew()

skew_cols = (skew_vals
             .sort_values(ascending=False)
             .to_frame()
             .rename(columns={0:'Skew'})
             .query('abs(Skew) > {0}'.format(skew_limit)))

skew_cols
```

	Skew
Parking	18.452062
Area	8.577740
Bathroom	0.877278
BHK	0.701826

In [21]:
```
field = "Area"
fig, (ax_before, ax_after) = plt.subplots(1, 2, figsize=(10, 5))
train_x[field].hist(ax=ax_before)
train_x[field].apply(np.log1p).hist(ax=ax_after)
ax_before.set(title='before np.log1p', ylabel='frequency', xlabel='value')
ax_after.set(title='after np.log1p', ylabel='frequency', xlabel='value')
fig.suptitle('Field "{}"'.format(field));
```



In [23]:
```
pd.options.mode.chained_assignment = None

for col in skew_cols.index.tolist():
    train_x[col] = np.log1p(train_x[col])
    test_x[col]  = test_x[col].apply(np.log1p)
```

In [24]:
```
from sklearn.metrics import mean_squared_error


def rmse(ytrue, ypredicted):
    return np.sqrt(mean_squared_error(ytrue, ypredicted))
```

In [25]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()

linearRegression = lr.fit(train_x ,train_y)

linearRegression_rmse = rmse(test_y, linearRegression.predict(test_x))

print(linearRegression_rmse)
```

```
1.9591915469345322e+20
```

In [27]:
```
from sklearn.linear_model import RidgeCV

alphas = [0.005, 0.05, 0.1, 0.3, 1, 3, 5, 10, 15, 30, 80]

ridgeCV = RidgeCV(alphas=alphas,
                  cv=4).fit(train_x ,train_y)

ridgeCV_rmse = rmse(test_y, ridgeCV.predict(test_x))

print(ridgeCV.alpha_, ridgeCV_rmse)
```

```
0.1 19782195.087912865
```

In [28]:
```
from sklearn.linear_model import LassoCV

alphas2 = np.array([1e-5, 5e-5, 0.0001, 0.0005])

lassoCV = LassoCV(alphas=alphas2,
                  max_iter=5e4,
                  cv=3).fit(train_x ,train_y)

lassoCV_rmse = rmse(test_y, lassoCV.predict(test_x))

print(lassoCV.alpha_, lassoCV_rmse)
```

In [28]:
```
from sklearn.linear_model import LassoCV

alphas2 = np.array([1e-5, 5e-5, 0.0001, 0.0005])

lassoCV = LassoCV(alphas=alphas2,
                  max_iter=5e4,
                  cv=3).fit(train_x ,train_y)

lassoCV_rmse = rmse(test_y, lassoCV.predict(test_x))

print(lassoCV.alpha_, lassoCV_rmse)
```

```
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 7981129899647040.0, tolerance: 35482478836357.27
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 7981352998263640.0, tolerance: 35482478836357.27
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 7981003781756634.0, tolerance: 35482478836357.27
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 7981226350266338.0, tolerance: 35482478836357.27
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.1721023571883608e+16, tolerance: 3083401576798
5.992
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.1721071017000084e+16, tolerance: 3083401576798
5.992
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.1721082920320516e+16, tolerance: 3083401576798
5.992
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.1721042455755416e+16, tolerance: 3083401576798
5.992
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.1901246973375728e+16, tolerance: 3695445519613
6.37
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did n
ot converge. You might want to increase the number of iterations. Duality gap: 1.1901204425546868e+16, tolerance: 3695445519613
6.37
  model = cd_fast.enet_coordinate_descent_gram(
```

In [29]:
```
from sklearn.linear_model import ElasticNetCV

l1_ratios = np.linspace(0.1, 0.9, 9)

elasticNetCV = ElasticNetCV(alphas=alphas2,
                            l1_ratio=l1_ratios,
                            max_iter=1e4).fit(train_x,train_y)
elasticNetCV_rmse = rmse(test_y, elasticNetCV.predict(test_x))

print(elasticNetCV.alpha_, elasticNetCV.l1_ratio_, elasticNetCV_rmse)
```

```
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did
not converge. You might want to increase the number of iterations. Duality gap: 2.484204814696173e+16, tolerance: 42294820838
160.59
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did
not converge. You might want to increase the number of iterations. Duality gap: 2.366168980429462e+16, tolerance: 42294820838
160.59
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did
not converge. You might want to increase the number of iterations. Duality gap: 2.266326920364921e+16, tolerance: 42294820838
160.59
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did
not converge. You might want to increase the number of iterations. Duality gap: 2.678307490503239e+16, tolerance: 39426779969
051.086
  model = cd_fast.enet_coordinate_descent_gram(
C:\Users\yashg\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:525: ConvergenceWarning: Objective did
not converge. You might want to increase the number of iterations. Duality gap: 2.603580443336639e+16, tolerance: 39426779969
051.086
```

In [30]:
```
rmse_vals = [linearRegression_rmse, ridgeCV_rmse, lassoCV_rmse, elasticNetCV_rmse]

labels = ['Linear', 'Ridge', 'Lasso', 'ElasticNet']

rmse_df = pd.Series(rmse_vals, index=labels).to_frame()
rmse_df.rename(columns={0: 'RMSE'}, inplace=1)
rmse_df
```

Out[30]:

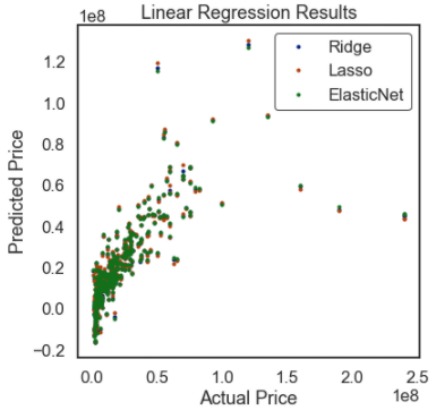
	RMSE
Linear	1.959192e+20
Ridge	1.978220e+07
Lasso	1.994333e+07
ElasticNet	1.969252e+07

In [31]:
```
data.head(4)
```

In [53]:
```
elasticNetCV.predict(test_x[test_x.index==103])
```

Out[53]:
```
array([46374850.88198759])
```

In [54]:
```
f = plt.figure(figsize=(6,6))
ax = plt.axes()

labels = ['Ridge', 'Lasso', 'ElasticNet']

models = [ridgeCV, lassoCV, elasticNetCV]

for mod, lab in zip(models, labels):
    ax.plot(test_y, mod.predict(test_x),
            marker='o', ls='', ms=3.0, label=lab)


leg = plt.legend(frameon=True)
leg.get_frame().set_edgecolor('black')
leg.get_frame().set_linewidth(1.0)

ax.set(xlabel='Actual Price',
       ylabel='Predicted Price',
       title='Linear Regression Results');
```



The main Objective of this Project is to create Linear regression model which take input of details about the house like no of rooms , parking , location and etc and train the model to fit it and predict the price of the house using these values .

The best model is of elasticnet cv as we can see by least rmse in the jupyter notebook.

Key Finding :

See the notebook

Flaws :

The main flaws as we can see is in the last diagram were predicted result can be skewed it can be fixed by introduction normalization or we can try applying polynomial features to fix the result .