1. Treat the data to replace all missing data with median/ mode (whichever applicable), and remove all rows with outliers in column: '*Price*'

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

  df = pd.read_csv('Dataset_Day6.csv')
  print(df.info())
  missing_value_percent = df.isna().sum() /
  len(df) * 100
  print(missing_value_percent)
  df["Bathroom"].fillna(df["Bathroom"].median(),
  inplace=True)
  df["Furnishing"].fillna(df["Furnishing"].mode()
  [0], inplace=True)  # mode() because this is
  categorical data
  df["Parking"].fillna(df["Parking"].median(),
  inplace=True)
  df["Type"].fillna(df["Type"].mode()[0],
  inplace=True)
  print(df.info())
  print(df[["Price"]])
  OutlierData = pd.DataFrame()
  temp = df[["Price"]]
  for col in ["Price"]:
      Q1 = temp[col].quantile(0.25)  # Gives 25th
  Percentile or Q1
      Q3 = temp[col].quantile(0.75)  # Gives 75th
  Percentile or Q3

      IQR = Q3 - Q1

      UpperBound = Q3 + 1.5 * IQR
      LowerBound = Q1 - 1.5 * IQR

      OutlierData[col] = temp[col][(temp[col] <
  LowerBound) | (temp[col] > UpperBound)]
      df_OutlierFree = df.drop(OutlierData.index,
  axis=0)
  print(OutlierData.index)
  print(len(OutlierData))
```

```python
print(df_OutlierFree[["Price"]])
print(df_OutlierFree[["Bathroom"]])
df_OutlierFree.info()
```

```
3 RangeIndex: 1259 entries, 0 to 1258
4 Data columns (total 9 columns):
5  #   Column       Non-Null Count  Dtype
6 ---  ------       --------------  -----
7  0   Area         1259 non-null   float64
8  1   BHK          1259 non-null   int64
9  2   Bathroom     1257 non-null   float64
10 3   Furnishing   1254 non-null   object
11 4   Parking      1226 non-null   float64
12 5   Price        1259 non-null   int64
13 6   Status       1259 non-null   object
14 7   Transaction  1259 non-null   object
15 8   Type         1254 non-null   object
16 dtypes: float64(3), int64(2), object(4)
17 memory usage: 88.7+ KB
18 None
19 Area           0.000000
20 BHK            0.000000
21 Bathroom       0.158856
22 Furnishing     0.397141
23 Parking        2.621128
24 Price          0.000000
25 Status         0.000000
26 Transaction    0.000000
27 Type           0.397141
28 dtype: float64
29 <class 'pandas.core.frame.DataFrame'>
30 RangeIndex: 1259 entries, 0 to 1258
31 Data columns (total 9 columns):
32  #   Column       Non-Null Count  Dtype
33 ---  ------       --------------  -----
34  0   Area         1259 non-null   float64
35  1   BHK          1259 non-null   int64
36  2   Bathroom     1259 non-null   float64
37  3   Furnishing   1259 non-null   object
38  4   Parking      1259 non-null   float64
39  5   Price        1259 non-null   int64
40  6   Status       1259 non-null   object
41  7   Transaction  1259 non-null   object
42  8   Type         1259 non-null   object
43 dtypes: float64(3), int64(2), object(4)
44 memory usage: 88.7+ KB
45 None
46          Price
47 0      6500000
48 1      5000000
49 2     15500000
50 3      4200000
51 4      6200000
52 ...        ...
```

```
[1259 rows x 1 columns]
Index([  50,   57,   94,  109,  209,  211,  215,  222,  225,  228,
       ...
       1213, 1216, 1219, 1221, 1223, 1224, 1226, 1227, 1232, 1245],
      dtype='int64', length=104)
104
         Price
0      6500000
1      5000000
2     15500000
3      4200000
4      6200000
...        ...
1254  55000000
1255  12500000
1256  17500000
1257  11500000
1258  18500000

[1155 rows x 1 columns]
      Bathroom
0          2.0
1          2.0
2          2.0
3          2.0
4          2.0
...        ...
1254       5.0
1255       2.0
1256       3.0
1257       2.0
1258       3.0

[1155 rows x 1 columns]
<class 'pandas.core.frame.DataFrame'>
Index: 1155 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1155 non-null   float64
 1   BHK          1155 non-null   int64
 2   Bathroom     1155 non-null   float64
 3   Furnishing   1155 non-null   object
 4   Parking      1155 non-null   float64
 5   Price        1155 non-null   int64
 6   Status       1155 non-null   object
 7   Transaction  1155 non-null   object
 8   Type         1155 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 90.2+ KB

Process finished with exit code 0
```

2.Use One Hot Encoding to encode all character variables.

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

```python
import seaborn as sns

df = pd.read_csv('Dataset_Day6.csv')
print(df.info())
temp = df[["Furnishing", "Status", "Transaction",
"Type"]]
# this function converts all categorical variables
in the dataframe to one hot encoded variables
new_encoded_data = pd.get_dummies(temp)
print(new_encoded_data)
```

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day6Q2.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1259 non-null   float64
 1   BHK          1259 non-null   int64
 2   Bathroom     1257 non-null   float64
 3   Furnishing   1254 non-null   object
 4   Parking      1226 non-null   float64
 5   Price        1259 non-null   int64
 6   Status       1259 non-null   object
 7   Transaction  1259 non-null   object
 8   Type         1254 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 88.7+ KB
None
      Furnishing_Furnished  ...  Type_Builder_Floor
0                    False  ...                True
1                    False  ...               False
2                     True  ...               False
3                    False  ...                True
4                    False  ...                True
...                    ...  ...                 ...
1254                 False  ...                True
1255                 False  ...                True
1256                 False  ...                True
1257                 False  ...                True
1258                 False  ...                True

[1259 rows x 9 columns]

Process finished with exit code 0

```

3.Split the data into 80% training and 20% testing data. Then, create a multiple linear regression model with target variable as ‘Price’.

a. Print the model performance metrics. R2, adjusted R2, MAE

```
2.import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  import seaborn as sns
  from sklearn.linear_model import
  LinearRegression
  from sklearn.model_selection import
  train_test_split
  from sklearn.metrics import mean_squared_error,
  r2_score, mean_absolute_error

  df = pd.read_csv('Dataset_Day6.csv')
  print(df.info())
  missing_value_percent = df.isna().sum() /
  len(df) * 100
  print(missing_value_percent)
  df["Bathroom"].fillna(df["Bathroom"].median(),
  inplace=True)
  df["Furnishing"].fillna(df["Furnishing"].mode()
  [0], inplace=True)  # mode() because this is
  categorical data
  df["Parking"].fillna(df["Parking"].median(),
  inplace=True)
  df["Type"].fillna(df["Type"].mode()[0],
  inplace=True)
  print(df.info())
  X = df.drop('Price', axis=1)  # all columns
  except 'Price'
  y = df['Price']  # target Variable
  temp = df[["Furnishing", "Status",
  "Transaction", "Type"]]
  # this function converts all categorical
  variables in the dataframe to one hot encoded
  variables
  new_encoded_data = pd.get_dummies(temp)
  # Concatenate encoded categorical variables
  with remaining features
  X =
```

```
pd.concat([X.select_dtypes(exclude=['object']),
new_encoded_data], axis=1)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=50)
lm = LinearRegression()
lm = lm.fit(X_train, y_train)
y_pred = lm.predict(X_test)
print(lm.coef_)  # scale parameter
print(lm.intercept_)  # intercept parameter
r2 = r2_score(y_test, y_pred)
adjusted_r2 = 1 - (1 - r2) * (len(y_test) - 1)
/ (len(y_test) - X_test.shape[1] - 1)
mae = mean_absolute_error(y_test, y_pred)
print("R2 Score:", r2)
print("Adjusted R2 Score:", adjusted_r2)
print("Mean Absolute Error (MAE):", mae)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1259 non-null   float64
 1   BHK          1259 non-null   int64
 2   Bathroom     1257 non-null   float64
 3   Furnishing   1254 non-null   object
 4   Parking      1226 non-null   float64
 5   Price        1259 non-null   int64
 6   Status       1259 non-null   object
 7   Transaction  1259 non-null   object
 8   Type         1254 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 88.7+ KB
None
Area           0.000000
BHK            0.000000
Bathroom       0.158856
Furnishing     0.397141
Parking        2.621128
Price          0.000000
Status         0.000000
Transaction    0.000000
Type           0.397141
dtype: float64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1259 non-null   float64
 1   BHK          1259 non-null   int64
 2   Bathroom     1259 non-null   float64
 3   Furnishing   1259 non-null   object
 4   Parking      1259 non-null   float64
 5   Price        1259 non-null   int64
 6   Status       1259 non-null   object
 7   Transaction  1259 non-null   object
 8   Type         1259 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 88.7+ KB
None
[ 5.30984492e+03  4.85967750e+05  1.30622279e+07  1.07695584e+05
 -1.22844024e+06 -4.84332719e+05  1.71277296e+06  2.34134349e+06
 -2.34134349e+06  2.41435659e+06 -2.41435659e+06  3.90473960e+05
 -3.90473960e+05]
-18606322.05682992
R2 Score: 0.5046575836208773
Adjusted R2 Score: 0.47760106507916056
Mean Absolute Error (MAE): 10400538.892955128
```

4.Repeat the above process for a Ridge Regression and show from the new evaluation metrics if there is any improvement in the model performance?

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import
train_test_split
from sklearn.metrics import mean_squared_error,
r2_score, mean_absolute_error
from sklearn.linear_model import Ridge


df = pd.read_csv('Dataset_Day6.csv')
print(df.info())
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
df["Bathroom"].fillna(df["Bathroom"].median(),
inplace=True)
df["Furnishing"].fillna(df["Furnishing"].mode()[0],
inplace=True)  # mode() because this is categorical
data
df["Parking"].fillna(df["Parking"].median(),
inplace=True)
df["Type"].fillna(df["Type"].mode()[0],
inplace=True)
print(df.info())
X = df.drop('Price', axis=1)  # all columns except
'Price'
y = df['Price']  # target Variable
temp = df[["Furnishing", "Status", "Transaction",
"Type"]]
# this function converts all categorical variables
in the dataframe to one hot encoded variables
new_encoded_data = pd.get_dummies(temp)
# Concatenate encoded categorical variables with
remaining features
X = pd.concat([X.select_dtypes(exclude=['object']),
new_encoded_data], axis=1)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=50)
```

```
alpha = [1e-50, 1e-20, 1e-15, 1e-10, 1e-8, 1e-3,
1e-2, 1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100]
print(alpha)

for param in alpha:
    ridgeModel = Ridge(alpha=param)
    ridgeModel.fit(X_train, y_train)

    y_pred = ridgeModel.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)

    r2 = r2_score(y_test, y_pred)
    print(r2)
    print("alpha = {}".format(param))
    print("mse for above alpha = {}".format(mse))
    best_r2 = 0
    best_alpha = 0
    best_mse = mse

    if r2 > best_r2:
        best_r2 = r2

    if mse < best_mse:
        best_mse = mse
        best_alpha = param

print("Best value of alpha/lambda =
{}".format(best_alpha))
print("MSE for this alpha/lambda =
{}".format(best_mse))
print("Best R square score = {}".format(best_r2))
```

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day6Q4.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1259 non-null   float64
 1   BHK          1259 non-null   int64
 2   Bathroom     1257 non-null   float64
 3   Furnishing   1254 non-null   object
 4   Parking      1226 non-null   float64
 5   Price        1259 non-null   int64
 6   Status       1259 non-null   object
 7   Transaction  1259 non-null   object
 8   Type         1254 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 88.7+ KB
None
Area           0.000000
BHK            0.000000
Bathroom       0.158856
Furnishing     0.397141
Parking        2.621128
Price          0.000000
Status         0.000000
Transaction    0.000000
Type           0.397141
dtype: float64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1259 non-null   float64
 1   BHK          1259 non-null   int64
 2   Bathroom     1259 non-null   float64
 3   Furnishing   1259 non-null   object
 4   Parking      1259 non-null   float64
 5   Price        1259 non-null   int64
 6   Status       1259 non-null   object
 7   Transaction  1259 non-null   object
 8   Type         1259 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 88.7+ KB
None
[1e-50, 1e-20, 1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100]
0.4541913846851654
alpha = 1e-50
mse for above alpha = 291845213565044.8
0.5046575836276497
alpha = 1e-20
mse for above alpha = 264860812449115.7
0.5046575836208771
alpha = 1e-15
mse for above alpha = 264860812452737.03
0.5046575836209053
alpha = 1e-10
```

```
mse for above alpha = 264860812452721.9
0.5046575836194864
alpha = 1e-08
mse for above alpha = 264860812453480.56
0.5046574403016636
alpha = 0.001
mse for above alpha = 264860889085874.16
0.5046561503858298
alpha = 0.01
mse for above alpha = 264861578807053.78
0.5045137976457408
alpha = 1
mse for above alpha = 264937695168480.0
0.5039298271796369
alpha = 5
mse for above alpha = 265249945617838.47
C:\Users\tejas\PycharmProjects\pythonProject\venv\Lib\site-packages\sklearn\
   linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=4.
   24583e-20): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
C:\Users\tejas\PycharmProjects\pythonProject\venv\Lib\site-packages\sklearn\
   linear_model\_ridge.py:211: LinAlgWarning: Ill-conditioned matrix (rcond=4.
   24575e-18): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
0.5031821241642653
alpha = 10
mse for above alpha = 265649744265352.44
0.5016388159162803
alpha = 20
mse for above alpha = 266474954994153.5
0.5000470304850191
alpha = 30
mse for above alpha = 267326086592485.3
0.499237469650079
alpha = 35
mse for above alpha = 267758960768776.34
0.498420692126058
alpha = 40
mse for above alpha = 268195693726528.5
0.49759787420242485
alpha = 45
mse for above alpha = 268635656500859.84
0.4967700333781435
alpha = 50
mse for above alpha = 269078305032563.34
0.49593805276367364
alpha = 55
mse for above alpha = 269523167120296.9
0.48836399721653057
alpha = 100
mse for above alpha = 273573033312743.28
Best value of alpha/lambda = 0
MSE for this alpha/lambda = 273573033312743.28
Best R square score = 0.48836399721653057

Process finished with exit code 0

```

5.Also do above for Lasso Regression (research code online)

(There's a convergence warning in all prevailing outputs, meaning: the alpha range must be increased from 100 to let's say 1000 or 10000 etc for the regression model to converge)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import
train_test_split
from sklearn.metrics import mean_squared_error,
r2_score, mean_absolute_error
from sklearn.linear_model import Lasso

df = pd.read_csv('Dataset_Day6.csv')
print(df.info())
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
df["Bathroom"].fillna(df["Bathroom"].median(),
inplace=True)
df["Furnishing"].fillna(df["Furnishing"].mode()[0],
inplace=True)  # mode() because this is categorical
data
df["Parking"].fillna(df["Parking"].median(),
inplace=True)
df["Type"].fillna(df["Type"].mode()[0],
inplace=True)
print(df.info())
X = df.drop('Price', axis=1)  # all columns except
'Price'
y = df['Price']  # target Variable
temp = df[["Furnishing", "Status", "Transaction",
```

```
"Type"]]
# this function converts all categorical variables
in the dataframe to one hot encoded variables
new_encoded_data = pd.get_dummies(temp)
# Concatenate encoded categorical variables with
remaining features
X = pd.concat([X.select_dtypes(exclude=['object']),
new_encoded_data], axis=1)
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=50)
alpha = [1e-50, 1e-20, 1e-15, 1e-10, 1e-8, 1e-3,
1e-2, 1, 5, 10, 20, 30, 35, 40, 45, 50, 55, 100]
print(alpha)

for param in alpha:
    lassoModel = Lasso(alpha=param)
    lassoModel.fit(X_train, y_train)

    y_pred = lassoModel.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)

    r2 = r2_score(y_test, y_pred)
    print(r2)
    print("alpha = {}".format(param))
    print("mse for above alpha = {}".format(mse))
    best_r2 = 0
    best_alpha = 0
    best_mse = mse

    if r2 > best_r2:
        best_r2 = r2

    if mse < best_mse:
        best_mse = mse
        best_alpha = param

print("Best value of alpha/lambda =
{}".format(best_alpha))
print("MSE for this alpha/lambda =
{}".format(best_mse))
print("Best R square score = {}".format(best_r2))
```

```
print("In Lasso Model there's a minimal improvement
in the performance as there's a minimal change in
the R2 score")
```

```
  tejas\PycharmProjects\pythonProject\START\Day6Q5.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1259 non-null   float64
 1   BHK          1259 non-null   int64
 2   Bathroom     1257 non-null   float64
 3   Furnishing   1254 non-null   object
 4   Parking      1226 non-null   float64
 5   Price        1259 non-null   int64
 6   Status       1259 non-null   object
 7   Transaction  1259 non-null   object
 8   Type         1254 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 88.7+ KB
None
Area           0.000000
BHK            0.000000
Bathroom       0.158856
Furnishing     0.397141
Parking        2.621128
Price          0.000000
Status         0.000000
Transaction    0.000000
Type           0.397141
dtype: float64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Area         1259 non-null   float64
 1   BHK          1259 non-null   int64
 2   Bathroom     1259 non-null   float64
 3   Furnishing   1259 non-null   object
 4   Parking      1259 non-null   float64
 5   Price        1259 non-null   int64
 6   Status       1259 non-null   object
 7   Transaction  1259 non-null   object
 8   Type         1259 non-null   object
dtypes: float64(3), int64(2), object(4)
memory usage: 88.7+ KB
None
[1e-50, 1e-20, 1e-15, 1e-10, 1e-08, 0.001, 0.01, 1, 5, 10, 20, 30, 35, 40, 45,
50, 55, 100]
C:\Users\tejas\PycharmProjects\pythonProject\venv\Lib\site-packages\sklearn\
linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not
converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 1.372e+17,
tolerance: 6.896e+13 Linear regression models with null weight for the l1
regularization term are more efficiently fitted using one of the solvers
implemented in sklearn.linear_model.Ridge/RidgeCV instead.
  model = cd_fast.enet_coordinate_descent(
0.5046575836209197
alpha = 1e-50
mse for above alpha = 264860812452714.22
```

```
0.5046575836209195
alpha = 1e-20
mse for above alpha = 264860812452714.34
C:\Users\tejas\PycharmProjects\pythonProject\venv\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 1.372e+17, tolerance: 6.896e+13 Linear regression models with null weight for the l1 regularization term are more efficiently fitted using one of the solvers implemented in sklearn.linear_model.Ridge/RidgeCV instead.
  model = cd_fast.enet_coordinate_descent(
C:\Users\tejas\PycharmProjects\pythonProject\venv\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 1.372e+17, tolerance: 6.896e+13
  model = cd_fast.enet_coordinate_descent(
0.5046575836209195
alpha = 1e-15
mse for above alpha = 264860812452714.34
0.5046575836209186
alpha = 1e-10
mse for above alpha = 264860812452714.84
C:\Users\tejas\PycharmProjects\pythonProject\venv\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 1.371e+17, tolerance: 6.896e+13
  model = cd_fast.enet_coordinate_descent(
C:\Users\tejas\PycharmProjects\pythonProject\venv\Lib\site-packages\sklearn\linear_model\_coordinate_descent.py:628: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check the scale of the features or consider increasing regularisation. Duality gap: 1.368e+17, tolerance: 6.896e+13
  model = cd_fast.enet_coordinate_descent(
0.504657583620919
alpha = 1e-08
mse for above alpha = 264860812452714.62
0.5046575836602472
alpha = 0.001
mse for above alpha = 264860812431685.72
0.5046575840142009
alpha = 0.01
mse for above alpha = 264860812242425.84
0.504657622949104
alpha = 1
mse for above alpha = 264860791423837.03
0.5046577802605332
alpha = 5
mse for above alpha = 264860707309027.62
0.5046579768981364
alpha = 10
mse for above alpha = 264860602166415.94
0.5046583701665819
alpha = 20
mse for above alpha = 264860391884807.62
0.5046587634249349
alpha = 30
mse for above alpha = 264860181608595.8
```