Predict the house price

Perform following tasks:

1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation.
4. Implement linear regression
5. Evaluate the models and compare their respective scores like R2, RMSE, etc.

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

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

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

In [ ]:
```
df.info()
```

In [ ]:
```
df.shape
```

In [ ]:
```
df.isnull().sum()
```

In [ ]:
```
df.drop(labels='Posted On',axis=1,inplace=True)
df.drop(labels='Point of Contact',axis=1,inplace=True)
df.drop(["Area Locality"], axis="columns", inplace=True)
```

In [ ]:
```
df["Floor Number"]=df["Floor"].apply(lambda x:str(x).split()[0])
```

In [ ]:
```
df["Total Floor"]=df["Floor"].apply(lambda x:str(x).split()[-1])
```

In [ ]:
```
del df['Floor']
```

In [ ]:
```
df['Floor Number'].value_counts()
```

In [ ]:
```
df["Floor Number"] = df["Floor Number"].replace(['Ground'],0)
df["Floor Number"] = df["Floor Number"].replace(['Upper'],-1)
```

In [ ]:
```
df["Floor Number"] = df["Floor Number"].replace(['Lower'],-2)
```

In [ ]:
```
df["Floor Number"].value_counts()
```

In [ ]:
```
df["Total Floor"].value_counts()
```

In [ ]:
```
df["Total Floor"] = df["Total Floor"].replace(['Ground'],1)
```

In [ ]:
```
df["Total Floor"].value_counts()
```

In [ ]:
```
df["Area Type"].unique()
```

In [ ]:
```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['Area Type'] = label_encoder.fit_transform(df['Area Type'])
df['Area Type'].unique()
```

In [ ]:
```
label_encoder = preprocessing.LabelEncoder()
df["Furnishing Status"]= label_encoder.fit_transform(df["Furnishing Statu
df["Furnishing Status"].unique()
```

In [ ]:
```
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column "Area Type".
df["Tenant Preferred"]= label_encoder.fit_transform(df["Tenant Preferred"

df["Tenant Preferred"].unique()
```

In [ ]:
```
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column "Area Type".
df['City']= label_encoder.fit_transform(df['City'])

df['City'].unique()
```

In [ ]:
```
from sklearn.preprocessing import minmax_scale

df["Rent"] = minmax_scale(df["Rent"])
df["Size"] = minmax_scale(df["Size"])
```

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

In [ ]:
```
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(df.corr(), annot = True)
```

In [ ]:
```
def find_outliers_IQR(df):
   q1 = df.quantile(0.25)
   q3 = df.quantile(0.75)
   IQR = q3-q1
   outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
   return outliers
```

In [ ]:
```
outliers = find_outliers_IQR(df["Size"])
print("number of outliers: "+ str(len(outliers)))
print("max outlier value: "+ str(outliers.max()))
print("min outlier value: "+ str(outliers.min()))
outliers
```

In [ ]:
```
x = df.drop("Rent", axis = 1)
x
```

In [ ]:
```
y = df['Rent']
y
```

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

x_train, x_test, y_train, y_test = train_test_split(x, y, random_state =
```

In [ ]:
```
x.head()
```

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

In [ ]:
```
pred = lr.predict(x_test)
```

In [ ]:
```
actual = y_test
```

In [ ]:
```
base = pd.DataFrame()
base['actual'] = actual
base['predictions'] = pred
```

In [ ]:
```
base
```

In [ ]:
```
from sklearn.metrics import r2_score
r2_score(y_test, pred)
```

In [ ]:
```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test, pred))
```