Predict assessed property value for the purpose of property tax assessment

```
import numpy as np # numpy used for mathematical operation on array
import pandas as pd  # pandas used for data manipulation on dataframe
import seaborn as sns # seaborn used for data visualization
import matplotlib.pyplot as plt # matplotlib used for data visualization
```

```
df = pd.read_csv('/content/Train.csv')
```

```
df.head()
```

	PropertyID	Borough	State	Surroundings	Address	ZipCode	Block	Lot	BldgClassCategory	NoOfResidentialUnits	NoOfCommercialUnits	TotalNoOfUn
0	PR11101	1	NY	BATHGATE	1473 CROTONA PLACE	10456	2927	127	02 TWO FAMILY DWELLINGS	2	0	
1	PR11102	1	NY	BATHGATE	453 EAST 181	10457	3037	110	03 THREE FAMILY DWELLINGS	3	0	
2	PR11103	1	NY	BATHGATE	511 EAST 171 STREET	10457	2912	155	02 TWO FAMILY DWELLINGS	2	0	
3	PR11104	1	NY	BATHGATE	547 CLAREMONT PARKWAY	10457	2929	134	02 TWO FAMILY DWELLINGS	2	0	
4	PR11105	1	NY	BATHGATE	2063 WASHINGTON AVENUE	10457	3036	140	02 TWO FAMILY DWELLINGS	2	0	

```
df.tail()
```

	PropertyID	Borough	State	Surroundings	Address	ZipCode	Block	Lot	BldgClassCategory	NoOfResidentialUnits	NoOfCommercialUnits	TotalNoOfU
16805	PR27906	4	NY	CIVIC CENTER	30 PARK PLACE	10007	123	1213	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16806	PR27907	4	NY	CIVIC CENTER	56 LEONARD STREET	10013	176	1054	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16807	PR27908	4	NY	CIVIC CENTER	56 LEONARD STREET	10013	176	1102	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16808	PR27909	4	NY	TRIBECA	250 SOUTH END AVENUE	10280	16	2249	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16809	PR27910	4	NY	HARLEM-UPPER	415 WEST 150 STREET	10031	2065	22	08 RENTALS - ELEVATOR APARTMENTS	16	1	

```
df.shape
```

```
(16810, 19)
```

```
df.columns
```

```
Index(['PropertyID', 'Borough', 'State', 'Surroundings', 'Address', 'ZipCode',
       'Block', 'Lot', 'BldgClassCategory', 'NoOfResidentialUnits',
       'NoOfCommercialUnits', 'TotalNoOfUnits', 'LandAreaInSqFt',
       'GrossAreaInSqFt', 'YearOfConstruction', 'TaxClass_AtEvaluationTime',
       'BldgClass_AtEvaluationTime', 'DateOfEvaluation',
       'PropertyEvaluationvalue'],
      dtype='object')
```

```
df.dtypes
```

```
PropertyID                object
Borough                    int64
State                     object
Surroundings              object
Address                   object
ZipCode                    int64
Block                      int64
Lot                        int64
BldgClassCategory         object
NoOfResidentialUnits       int64
```

```
NoOfCommercialUnits            int64
TotalNoOfUnits                 int64
LandAreaInSqFt                 int64
GrossAreaInSqFt                int64
YearOfConstruction             int64
TaxClass_AtEvaluationTime      int64
BldgClass_AtEvaluationTime    object
DateOfEvaluation              object
PropertyEvaluationvalue        int64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16810 entries, 0 to 16809
Data columns (total 19 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   PropertyID                   16810 non-null  object
 1   Borough                      16810 non-null  int64
 2   State                        16810 non-null  object
 3   Surroundings                 16810 non-null  object
 4   Address                      16810 non-null  object
 5   ZipCode                      16810 non-null  int64
 6   Block                        16810 non-null  int64
 7   Lot                          16810 non-null  int64
 8   BldgClassCategory            16810 non-null  object
 9   NoOfResidentialUnits         16810 non-null  int64
 10  NoOfCommercialUnits          16810 non-null  int64
 11  TotalNoOfUnits               16810 non-null  int64
 12  LandAreaInSqFt               16810 non-null  int64
 13  GrossAreaInSqFt              16810 non-null  int64
 14  YearOfConstruction           16810 non-null  int64
 15  TaxClass_AtEvaluationTime    16810 non-null  int64
 16  BldgClass_AtEvaluationTime   16810 non-null  object
 17  DateOfEvaluation             16810 non-null  object
 18  PropertyEvaluationvalue      16810 non-null  int64
dtypes: int64(12), object(7)
memory usage: 2.4+ MB
```

```
df.isnull().sum()
```

```
PropertyID                    0
Borough                       0
State                         0
Surroundings                  0
Address                       0
ZipCode                       0
Block                         0
Lot                           0
BldgClassCategory             0
```

```
NoOfResidentialUnits          0
NoOfCommercialUnits           0
TotalNoOfUnits                0
LandAreaInSqFt                0
GrossAreaInSqFt               0
YearOfConstruction            0
TaxClass_AtEvaluationTime     0
BldgClass_AtEvaluationTime    0
DateOfEvaluation              0
PropertyEvaluationvalue       0
dtype: int64
```

```
#remove duplicate value if any

df = df.drop_duplicates()
```

```
df.shape
```

```
(16810, 19)
```

```
# Creating the Data Dictionary with first column being datatype.

Data_dict = pd.DataFrame(df.dtypes)
Data_dict
```

	0
PropertyID	object
Borough	int64
State	object
Surroundings	object
Address	object
ZipCode	int64
Block	int64
Lot	int64
BldgClassCategory	object
NoOfResidentialUnits	int64
NoOfCommercialUnits	int64
TotalNoOfUnits	int64
LandAreaInSqFt	int64
GrossAreaInSqFt	int64
YearOfConstruction	int64
TaxClass_AtEvaluationTime	int64
BldgClass_AtEvaluationTime	object
DateOfEvaluation	object
PropertyEvaluationvalue	int64

```
# Identifying unique values . For this we used nunique() which returns unique elements in the object.

Data_dict['UniqueVal'] = df.nunique()
Data_dict
```

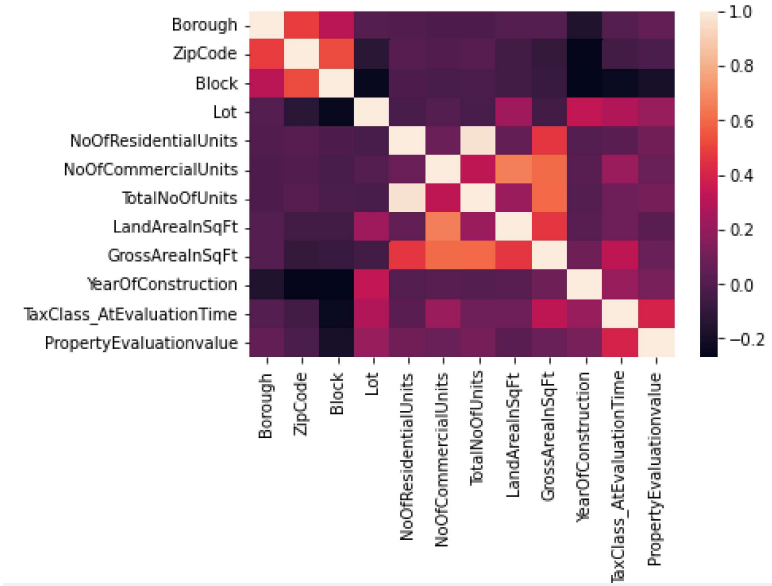
	0	UniqueVal
PropertyID	object	16810
Borough	int64	5
State	object	1
Surroundings	object	243
Address	object	15219
ZipCode	int64	174
Block	int64	6405
Lot	int64	1299
BldgClassCategory	object	52
NoOfResidentialUnits	int64	70
NoOfCommercialUnits	int64	18
TotalNoOfUnits	int64	75
LandAreaInSqFt	int64	3898
GrossAreaInSqFt	int64	3350
YearOfConstruction	int64	72
TaxClass_AtEvaluationTime	int64	3
BldgClass_AtEvaluationTime	object	101
DateOfEvaluation	object	572
PropertyEvaluationvalue	int64	3694

```
corr = df.corr()
corr
```

	Borough	ZipCode	Block	Lot	NoOfResidentialUnits	NoOfCommercialUnits	TotalNoOfUnits	LandAreaInSqFt	GrossAreaIn
Borough	1.000000	0.488406	0.313951	0.005866	-0.005122	-0.006849	-0.006620	0.002994	0.00
ZipCode	0.488406	1.000000	0.528438	-0.125573	0.014876	-0.002288	0.013504	-0.046456	-0.09
Block	0.313951	0.528438	1.000000	-0.240481	-0.008005	-0.024037	-0.013785	-0.046750	-0.08
Lot	0.005866	-0.125573	-0.240481	1.000000	-0.025278	-0.000308	-0.024030	0.234273	-0.05
NoOfResidentialUnits	-0.005122	0.014876	-0.008005	-0.025278	1.000000	0.073065	0.966332	0.049807	0.46
NoOfCommercialUnits	-0.006849	-0.002288	-0.024037	-0.000308	0.073065	1.000000	0.327214	0.660527	0.60
TotalNoOfUnits	-0.006620	0.013504	-0.013785	-0.024030	0.966332	0.327214	1.000000	0.217598	0.59
LandAreaInSqFt	0.002994	-0.046456	-0.046750	0.234273	0.049807	0.660527	0.217598	1.000000	0.46
GrossAreaInSqFt	0.005231	-0.090817	-0.083749	-0.050164	0.467357	0.604961	0.598885	0.465828	1.00
YearOfConstruction	-0.168352	-0.261036	-0.269244	0.339635	0.005098	0.020689	0.010168	0.018417	0.08
TaxClass_AtEvaluationTime	0.003318	-0.044437	-0.234643	0.280392	0.038005	0.219687	0.092686	0.090533	0.32
PropertyEvaluationvalue	0.049743	-0.012603	-0.186906	0.207938	0.098409	0.068230	0.110843	0.034867	0.06

```
# Visulaizing the heatmap of complete dataset

sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8300dac110>
```




```
#since TotalNoOfUnits and NoOfResidentialUnits have strong co relation drop one column

df = df.drop(['TotalNoOfUnits'],  axis=1)


df.columns
```

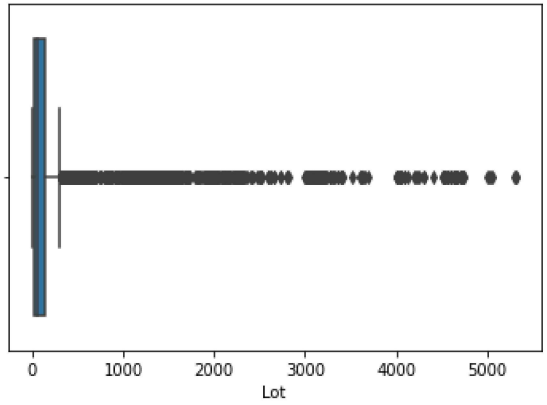
```
Index(['PropertyID', 'Borough', 'State', 'Surroundings', 'Address', 'ZipCode',
       'Block', 'Lot', 'BldgClassCategory', 'NoOfResidentialUnits',
       'NoOfCommercialUnits', 'LandAreaInSqFt', 'GrossAreaInSqFt',
       'YearOfConstruction', 'TaxClass_AtEvaluationTime',
       'BldgClass_AtEvaluationTime', 'DateOfEvaluation',
       'PropertyEvaluationvalue'],
      dtype='object')
```

```
df.describe()
```

	Borough	ZipCode	Block	Lot	NoOfResidentialUnits	NoOfCommercialUnits	LandAreaInSqFt	GrossAreaInSqFt	YearOfConstructi
count	16810.000000	16810.000000	16810.000000	16810.000000	16810.000000	16810.000000	1.681000e+04	1.681000e+04	16810.000(
mean	3.439143	10871.027543	5428.122308	266.327722	1.862225	0.137775	1.092492e+04	3.612157e+03	1973.982
std	1.398950	522.996869	3710.031921	542.420132	8.027805	2.185857	5.837620e+04	1.627378e+04	23.632
min	1.000000	10001.000000	6.000000	1.000000	0.000000	0.000000	1.000000e+00	1.000000e+00	1941.000(
25%	2.000000	10312.000000	2401.250000	25.000000	1.000000	0.000000	2.100000e+03	1.224000e+03	1951.000(
50%	3.000000	11207.000000	5050.000000	52.000000	1.000000	0.000000	3.200000e+03	1.750000e+03	1968.000(
75%	5.000000	11364.000000	7376.750000	138.000000	2.000000	0.000000	5.000000e+03	2.449750e+03	1998.000(
max	5.000000	11694.000000	16314.000000	5323.000000	437.000000	252.000000	5.612000e+06	1.172005e+06	2012.000(

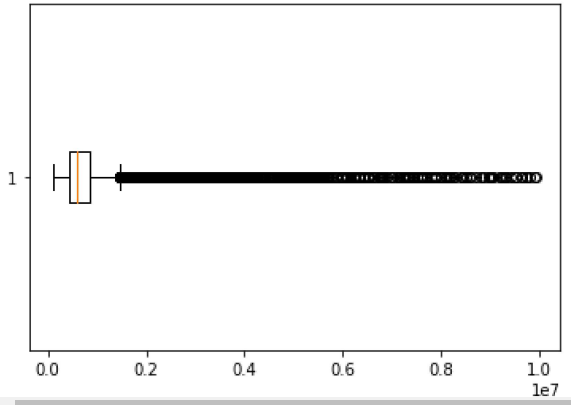
```
sns.boxplot(df['Lot'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82fb227c50>
```



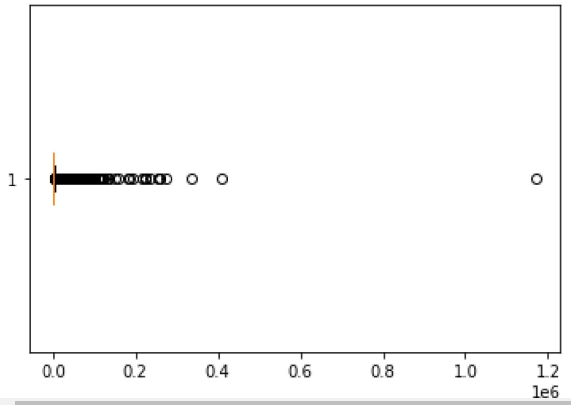

```
plt.boxplot(df['PropertyEvaluationvalue'], vert = False)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f82facbe090>,
  <matplotlib.lines.Line2D at 0x7f82facbe690>],
 'caps': [<matplotlib.lines.Line2D at 0x7f82facbebd0>,
  <matplotlib.lines.Line2D at 0x7f82facc6150>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f82fad37b10>],
 'medians': [<matplotlib.lines.Line2D at 0x7f82facc6650>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f82facc6b90>],
 'means': []}
```



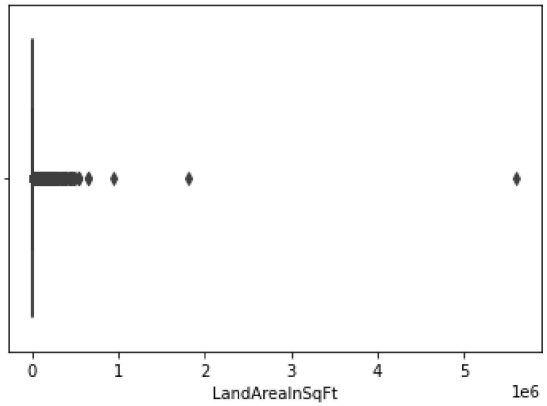

```
plt.boxplot(df['GrossAreaInSqFt'], vert = False)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f82faca8a10>,
  <matplotlib.lines.Line2D at 0x7f82faca8f50>],
 'caps': [<matplotlib.lines.Line2D at 0x7f82facad4d0>,
  <matplotlib.lines.Line2D at 0x7f82facada10>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f82faca8490>],
 'medians': [<matplotlib.lines.Line2D at 0x7f82facadf90>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f82facb6510>],
 'means': []}
```



```
sns.boxplot(df['LandAreaInSqFt'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82face4810>
```




```
#outlier treatment by upper bound and lower bound

per25 = df['Lot'].quantile(0.25)
per75 = df['Lot'].quantile(0.75)


per25
```

```
25.0
```

```
per75
```

```
138.0
```

```
iqr = per75 - per25


iqr
```

```
113.0
```

```
upper_limit = per75 + 1.5 * iqr
lower_limit = per25 - 1.5 * iqr


upper_limit
```

```
307.5
```

```
lower_limit
```

```
-144.5
```

```
df[df['Lot'] > upper_limit]
```

	PropertyID	Borough	State	Surroundings	Address	ZipCode	Block	Lot	BldgClassCategory	NoOfResidentialUnits	NoOfCommercialUnits	LandAreaI
51	PR11152	3	NY	NEW DORP	26 EVA AVENUE	10306	4058	348	01 ONE FAMILY DWELLINGS	1	0	
58	PR11159	3	NY	NEW DORP	40 EVA AVENUE	10306	4058	354	01 ONE FAMILY DWELLINGS	1	0	
78	PR11179	3	NY	NEW DORP	48A FRANCINE COURT	10306	4229	1051	04 TAX CLASS 1 CONDOS	1	0	
88	PR11189	3	NY	NEW DORP	36B FRANCINE COURT	10306	4229	1058	04 TAX CLASS 1 CONDOS	1	0	
139	PR11240	3	NY	GRANT CITY	175 ZOE STREET, 1E	10305	3543	1005	13 CONDOS - ELEVATOR APARTMENTS	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	
16804	PR27905	4	NY	CIVIC CENTER	56 LEONARD ST	10013	176	1067	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16805	PR27906	4	NY	CIVIC CENTER	30 PARK PLACE	10007	123	1213	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16806	PR27907	4	NY	CIVIC CENTER	56 LEONARD STREET	10013	176	1054	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16807	PR27908	4	NY	CIVIC CENTER	56 LEONARD STREET	10013	176	1102	13 CONDOS - ELEVATOR APARTMENTS	1	0	
16808	PR27909	4	NY	TRIBECA	250 SOUTH END AVENUE	10280	16	2249	13 CONDOS - ELEVATOR APARTMENTS	1	0	

2931 rows × 18 columns

```
df['Lot'] = np.where(df['Lot'] > upper_limit, upper_limit,
                     np.where(df['Lot'] < lower_limit, lower_limit,
                     df['Lot'] ))
```

```
df['Lot'].describe()
```

Show hidden output

```
sns.boxplot(df['Lot'])
```

Show hidden output

```
#outlier treatment for PropertyEvaluationvalue

per25 = df['PropertyEvaluationvalue'].quantile(0.25)
per75 = df['PropertyEvaluationvalue'].quantile(0.75)


per25
```

```
425000.0
```

```
per75
```

```
845000.0
```

```
iqr = per75 - per25


iqr
```

```
420000.0
```

```
upper_limit = per75 + 1.5 * iqr
lower_limit = per25 - 1.5 * iqr


upper_limit
```

```
1475000.0
```

```
lower_limit
```

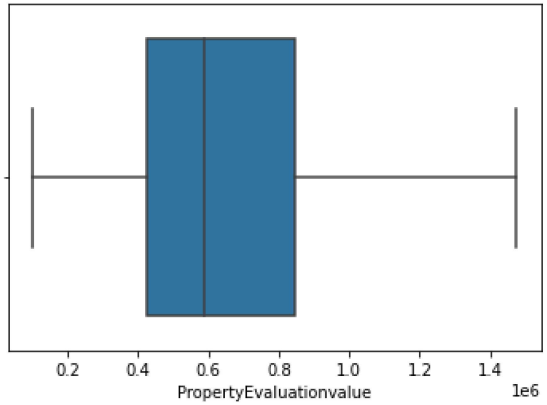
```
-205000.0
```

```
df['PropertyEvaluationvalue'] = np.where(df['PropertyEvaluationvalue'] > upper_limit, upper_limit,
                    np.where(df['PropertyEvaluationvalue'] < lower_limit, lower_limit,
                    df['PropertyEvaluationvalue'] ))


df['PropertyEvaluationvalue'].describe()
```

```
count    1.681000e+04
mean     6.774504e+05
std      3.534746e+05
min      1.000440e+05
25%      4.250000e+05
50%      5.900000e+05
75%      8.450000e+05
max      1.475000e+06
Name: PropertyEvaluationvalue, dtype: float64
```

```
sns.boxplot(df['PropertyEvaluationvalue'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82fab59710>
```




```
#outlier treatment for GrossAreaInSqFt

per25 = df['GrossAreaInSqFt'].quantile(0.25)
per75 = df['GrossAreaInSqFt'].quantile(0.75)
```

```
per25
```

```
1224.0
```

```
per75
```

```
2449.75
```

```
iqr = per75 - per25
```

```
iqr
```

```
1225.75
```

```
upper_limit = per75 + 1.5 * iqr
lower_limit = per25 - 1.5 * iqr
```

```
upper_limit
```

```
4288.375
```

```
lower_limit
```

```
-614.625
```

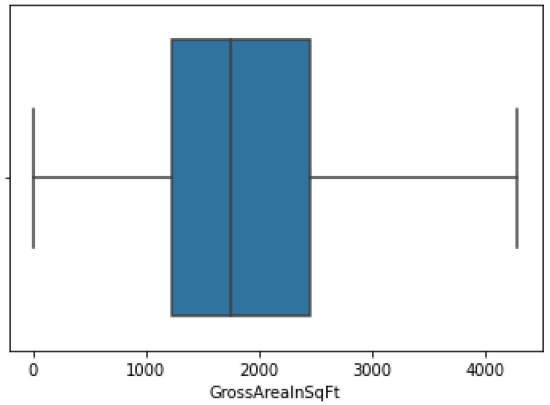
```
df['GrossAreaInSqFt'] = np.where(df['GrossAreaInSqFt'] > upper_limit, upper_limit,
                    np.where(df['GrossAreaInSqFt'] < lower_limit, lower_limit,
                    df['GrossAreaInSqFt'] ))
```

```
df['GrossAreaInSqFt'].describe()
```

```
count    16810.000000
mean      1959.677521
std        966.934568
min          1.000000
25%       1224.000000
50%       1750.000000
75%       2449.750000
max       4288.375000
Name: GrossAreaInSqFt, dtype: float64
```

```
sns.boxplot(df['GrossAreaInSqFt'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82fe6eae10>
```




```
#outlier treatment for LandAreaInSqFt

per25 = df['LandAreaInSqFt'].quantile(0.25)
per75 = df['LandAreaInSqFt'].quantile(0.75)


per25
```

```
2100.0
```

```
per75
```

```
5000.0
```

```
iqr = per75 - per25


iqr
```

```
2900.0
```

```
upper_limit = per75 + 1.5 * iqr
lower_limit = per25 - 1.5 * iqr


upper_limit
```

```
9350.0
```

```
lower_limit
```

```
-2250.0
```

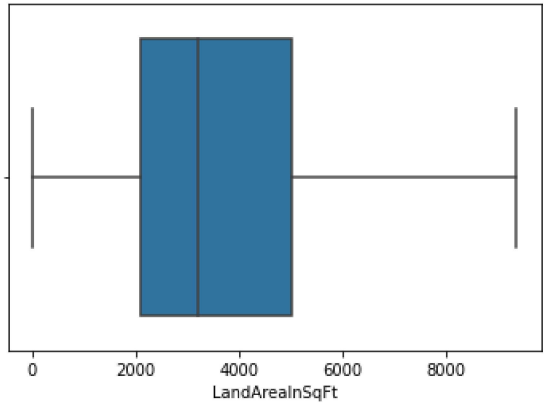
```
df['LandAreaInSqFt'] = np.where(df['LandAreaInSqFt'] > upper_limit, upper_limit,
                    np.where(df['LandAreaInSqFt'] < lower_limit, lower_limit,
                    df['LandAreaInSqFt'] ))
```

```
df['LandAreaInSqFt'].describe()
```

```
count    16810.000000
mean      4058.441701
std       2575.798611
min          1.000000
25%       2100.000000
50%       3200.000000
75%       5000.000000
max       9350.000000
Name: LandAreaInSqFt, dtype: float64
```

```
sns.boxplot(df['LandAreaInSqFt'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82faa42210>
```




```
#since state has only one unique value remove column

df = df.drop(['State'],  axis=1)
```

```
df.columns
```

```
Index(['PropertyID', 'Borough', 'Surroundings', 'Address', 'ZipCode', 'Block',
       'Lot', 'BldgClassCategory', 'NoOfResidentialUnits',
       'NoOfCommercialUnits', 'LandAreaInSqFt', 'GrossAreaInSqFt',
       'YearOfConstruction', 'TaxClass_AtEvaluationTime',
       'BldgClass_AtEvaluationTime', 'DateOfEvaluation',
       'PropertyEvaluationvalue'],
      dtype='object')
```

```
#remove property id since it is identity value

df = df.drop(['PropertyID'],  axis=1)
```

```
df = df.drop(['Surroundings', 'Address', 'BldgClass_AtEvaluationTime', 'BldgClassCategory'],  axis=1)
```

```
df.columns
```

```
Index(['Borough', 'ZipCode', 'Block', 'Lot', 'NoOfResidentialUnits',
       'NoOfCommercialUnits', 'LandAreaInSqFt', 'GrossAreaInSqFt',
       'YearOfConstruction', 'TaxClass_AtEvaluationTime', 'DateOfEvaluation',
       'PropertyEvaluationvalue'],
      dtype='object')
```

```
#converting date into a year as there wouldn't be huge change in tax evaluation within a year

date_col = pd.DatetimeIndex(df['DateOfEvaluation'])
```

```
date_col
```

```
DatetimeIndex(['2014-05-30', '2014-09-13', '2014-09-28', '2014-05-06',
               '2015-07-02', '2015-03-27', '2014-07-08', '2015-02-25',
               '2014-11-27', '2014-07-23',
               ...
               '2012-12-26', '2014-08-16', '2014-08-13', '2014-04-04',
               '2014-07-19', '2014-11-30', '2014-12-19', '2014-11-16',
               '2014-09-17', '2013-03-22'],
              dtype='datetime64[ns]', name='DateOfEvaluation', length=16810, freq=None)
```

```
df['DateOfEvaluation'] = date_col.year
```

```
df['DateOfEvaluation'].head()
```

```
0    2014
1    2014
2    2014
3    2014
```

```
4    2015
Name: DateOfEvaluation, dtype: int64
```

```
df.dtypes
```

```
Borough                       int64
ZipCode                       int64
Block                         int64
Lot                         float64
NoOfResidentialUnits          int64
NoOfCommercialUnits           int64
LandAreaInSqFt              float64
GrossAreaInSqFt             float64
YearOfConstruction            int64
TaxClass_AtEvaluationTime     int64
DateOfEvaluation              int64
PropertyEvaluationvalue     float64
dtype: object
```

```
c_df = df
```

```
c_df = pd.get_dummies(c_df,columns = ['Borough', 'TaxClass_AtEvaluationTime'], drop_first = True)
```

```
c_df.head()
```

	ZipCode	Block	Lot	NoOfResidentialUnits	NoOfCommercialUnits	LandAreaInSqFt	GrossAreaInSqFt	YearOfConstruction	DateOfEvaluation	PropertyEvalua
0	10456	2927	127.0	2	0	1900.0	2394.0	1990	2014	
1	10457	3037	110.0	3	0	1687.0	3037.0	2002	2014	
2	10457	2912	155.0	2	0	2000.0	2400.0	1989	2014	
3	10457	2929	134.0	2	0	2356.0	2394.0	1991	2014	
4	10457	3036	140.0	2	0	3346.0	2308.0	1993	2015	

```
c_df.dtypes
```

```
ZipCode                       int64
Block                         int64
Lot                         float64
NoOfResidentialUnits          int64
NoOfCommercialUnits           int64
LandAreaInSqFt              float64
GrossAreaInSqFt             float64
YearOfConstruction            int64
DateOfEvaluation              int64
```

```
PropertyEvaluationvalue        float64
Borough_2                        uint8
Borough_3                        uint8
Borough_4                        uint8
Borough_5                        uint8
TaxClass_AtEvaluationTime_2      uint8
TaxClass_AtEvaluationTime_4      uint8
dtype: object
```

```
X_train = c_df.loc[:,c_df.columns!="PropertyEvaluationvalue"]
```

```
y_train = df["PropertyEvaluationvalue"]
```

```
y_train.head()
```

```
0    540000.0
1    750000.0
2    470000.0
3    570000.0
4    640000.0
Name: PropertyEvaluationvalue, dtype: float64
```

```
X_train.shape
```

```
(16810, 15)
```

Reading test data

```
df_test = pd.read_csv('/content/Test.csv')
```

```
df_test.describe()
```

	Borough	ZipCode	Block	Lot	NoOfResidentialUnits	NoOfCommercialUnits	TotalNoOfUnits	LandAreaInSqFt	GrossAreaInSqFt	Ye
count	2936.000000	2936.000000	2936.000000	2936.000000	2936.000000	2936.000000	2936.000000	2936.000000	2936.000000	
mean	3.385559	10837.683924	5170.971730	279.483651	1.656335	0.136921	1.793256	11015.183924	4060.355586	
std	1.364623	526.594771	3581.310989	562.104975	4.456347	1.211945	4.733782	37702.788446	14807.827309	
min	1.000000	10001.000000	31.000000	1.000000	0.000000	0.000000	1.000000	1.000000	80.000000	
25%	2.000000	10310.000000	2373.000000	25.000000	1.000000	0.000000	1.000000	2100.000000	1222.000000	
50%	3.000000	11105.000000	4772.500000	54.000000	1.000000	0.000000	1.000000	3126.500000	1720.000000	
75%	5.000000	11361.250000	7106.000000	146.000000	2.000000	0.000000	2.000000	5000.000000	2440.250000	
max	5.000000	11694.000000	16262.000000	5026.000000	128.000000	50.000000	130.000000	484555.000000	231891.000000	

```
df_test.shape
```

```
(2936, 18)
```

```
df_test = df_test.drop_duplicates()
```

```
Data_dict = pd.DataFrame(df_test.dtypes)
Data_dict
```

	0
PropertyID	object
Borough	int64
State	object
Surroundings	object
Address	object
ZipCode	int64
Block	int64
Lot	int64
BldgClassCategory	object
NoOfResidentialUnits	int64
NoOfCommercialUnits	int64
TotalNoOfUnits	int64
LandAreaInSqFt	int64
GrossAreaInSqFt	int64
YearOfConstruction	int64
TaxClass_AtEvaluationTime	int64
BldgClass_AtEvaluationTime	object
DateOfEvaluation	object

```
Data_dict['UniqueVal'] = df_test.nunique()
Data_dict
```

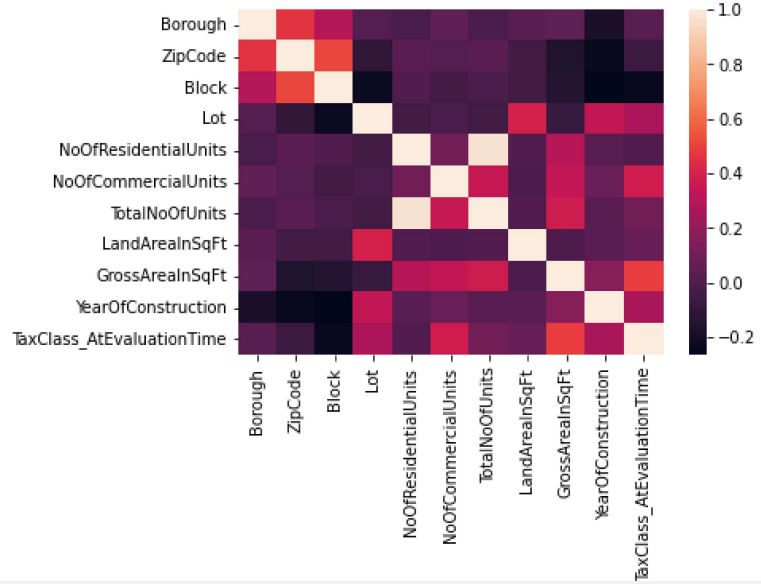
	0	UniqueVal
PropertyID	object	2936
Borough	int64	5
State	object	1
Surroundings	object	223
Address	object	2766
ZipCode	int64	164
Block	int64	2131
Lot	int64	583
BldgClassCategory	object	37
NoOfResidentialUnits	int64	24
NoOfCommercialUnits	int64	12
TotalNoOfUnits	int64	24
LandAreaInSqFt	int64	1259
GrossAreaInSqFt	int64	1394
YearOfConstruction	int64	72
TaxClass_AtEvaluationTime	int64	3
BldgClass_AtEvaluationTime	object	65
DateOfEvaluation	object	508

```
corr = df_test.corr()
corr
```

	Borough	ZipCode	Block	Lot	NoOfResidentialUnits	NoOfCommercialUnits	TotalNoOfUnits	LandAreaInSqFt	GrossAreaIn
Borough	1.000000	0.457944	0.299355	0.009729	-0.028404	0.046972	-0.014714	0.039233	0.04
ZipCode	0.457944	1.000000	0.511020	-0.100147	0.031245	0.004580	0.030587	-0.049038	-0.16
Block	0.299355	0.511020	1.000000	-0.227373	-0.001826	-0.041608	-0.012371	-0.046156	-0.13
Lot	0.009729	-0.100147	-0.227373	1.000000	-0.047423	-0.020048	-0.049777	0.399012	-0.07
NoOfResidentialUnits	-0.028404	0.031245	-0.001826	-0.047423	1.000000	0.100063	0.967011	-0.001254	0.30
NoOfCommercialUnits	0.046972	0.004580	-0.041608	-0.020048	0.100063	1.000000	0.350219	-0.006003	0.33
TotalNoOfUnits	-0.014714	0.030587	-0.012371	-0.049777	0.967011	0.350219	1.000000	-0.002717	0.37
LandAreaInSqFt	0.039233	-0.049038	-0.046156	0.399012	-0.001254	-0.006003	-0.002717	1.000000	-0.00
GrossAreaInSqFt	0.043270	-0.165990	-0.139460	-0.076744	0.308146	0.333557	0.375484	-0.007201	1.00
YearOfConstruction	-0.179691	-0.239600	-0.266536	0.341295	0.022131	0.071432	0.039122	0.038060	0.16
TaxClass_AtEvaluationTime	0.028197	-0.066878	-0.239526	0.276031	-0.002647	0.386007	0.096334	0.068779	0.48

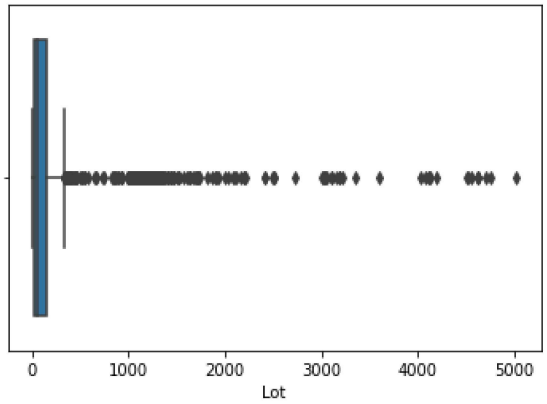
```
sns.heatmap(corr)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82f9897090>
```



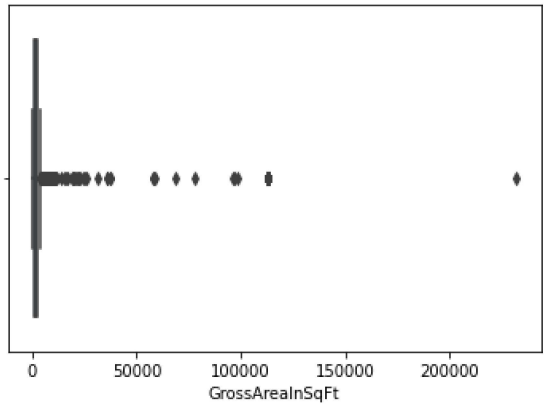
```
#since TotalNoOfUnits and NoOfResidentialUnits have strong co relation drop one column

df_test = df_test.drop(['TotalNoOfUnits'],  axis=1)


sns.boxplot(df_test['Lot'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82f97d69d0>
```



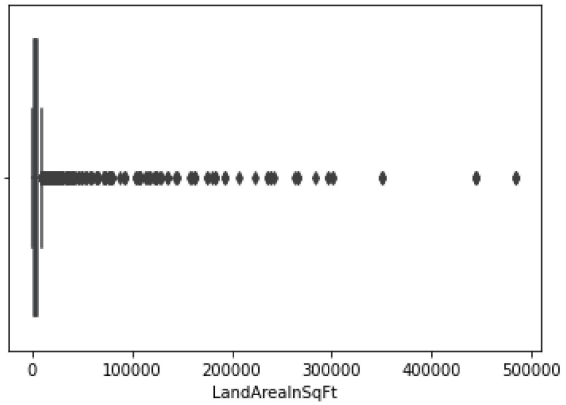

```
sns.boxplot(df_test['GrossAreaInSqFt'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82f974bf10>
```




```
sns.boxplot(df_test['LandAreaInSqFt'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82f973bd50>
```




```
#outlier treatment by upper bound and lower bound

per25 = df_test['Lot'].quantile(0.25)
per75 = df_test['Lot'].quantile(0.75)


per25
```

```
25.0
```

```
per75
```

```
146.0
```

```
iqr = per75 - per25


iqr
```

```
121.0
```

```
upper_limit = per75 + 1.5 * iqr
lower_limit = per25 - 1.5 * iqr


upper_limit
```

```
327.5
```

```
lower_limit
```

```
-156.5
```

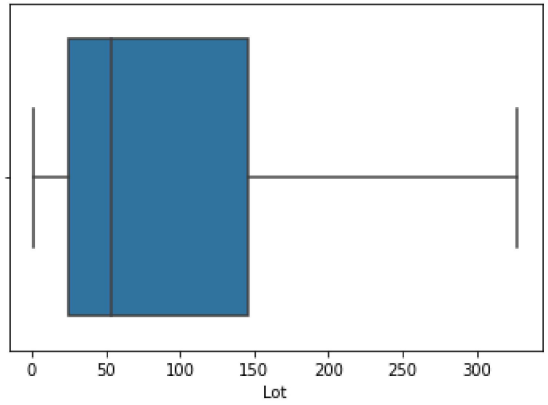
```
df_test['Lot'] = np.where(df_test['Lot'] > upper_limit, upper_limit,
                    np.where(df_test['Lot'] < lower_limit, lower_limit,
                    df_test['Lot'] ))
```

```
df_test['Lot'].describe()
```

```
count    2936.000000
mean      107.877895
std       115.291943
min         1.000000
25%        25.000000
50%        54.000000
75%       146.000000
max       327.500000
Name: Lot, dtype: float64
```

```
sns.boxplot(df_test['Lot'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82f97d6c10>
```



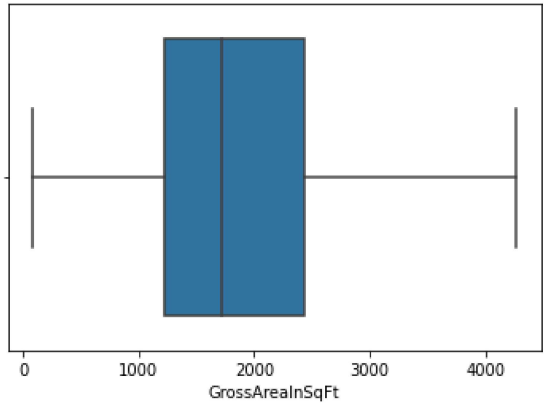

```
#outlier treatment for GrossAreaInSqFt

per25 = df_test['GrossAreaInSqFt'].quantile(0.25)
per75 = df_test['GrossAreaInSqFt'].quantile(0.75)
```

```
iqr = per75 - per25


upper_limit = per75 + 1.5 * iqr
lower_limit = per25 - 1.5 * iqr


df_test['GrossAreaInSqFt'] = np.where(df_test['GrossAreaInSqFt'] > upper_limit, upper_limit,
                    np.where(df_test['GrossAreaInSqFt'] < lower_limit, lower_limit,
                    df_test['GrossAreaInSqFt'] ))


df_test['GrossAreaInSqFt'].describe()
```

```
count    2936.000000
mean     1942.514561
std       966.164450
min        80.000000
25%      1222.000000
50%      1720.000000
75%      2440.250000
max      4267.625000
Name: GrossAreaInSqFt, dtype: float64
```

```
sns.boxplot(df_test['GrossAreaInSqFt'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82f962d490>
```



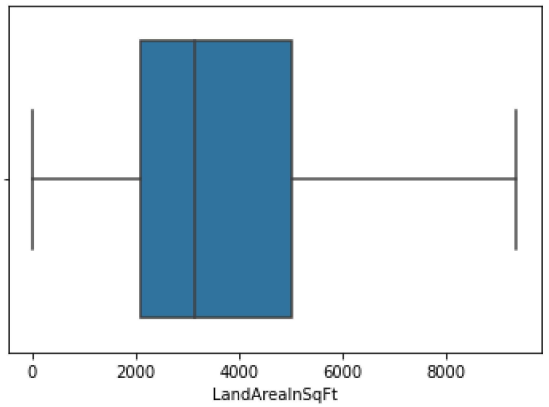

```
#outlier treatment for LandAreaInSqFt

per25 = df_test['LandAreaInSqFt'].quantile(0.25)
per75 = df_test['LandAreaInSqFt'].quantile(0.75)
```

```
iqr = per75 - per25


upper_limit = per75 + 1.5 * iqr
lower_limit = per25 - 1.5 * iqr


df_test['LandAreaInSqFt'] = np.where(df_test['LandAreaInSqFt'] > upper_limit, upper_limit,
                    np.where(df_test['LandAreaInSqFt'] < lower_limit, lower_limit,
                    df_test['LandAreaInSqFt'] ))


df_test['LandAreaInSqFt'].describe()
```

```
count    2936.000000
mean     4050.566417
std      2586.540249
min         1.000000
25%      2100.000000
50%      3126.500000
75%      5000.000000
max      9350.000000
Name: LandAreaInSqFt, dtype: float64
```

```
sns.boxplot(df_test['LandAreaInSqFt'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f82f9597b10>
```




```
#since state has only one unique value remove column

df_test = df_test.drop(['State'],  axis=1)
```

```
df_test = df_test.drop(['Surroundings', 'Address', 'BldgClass_AtEvaluationTime', 'BldgClassCategory'],  axis=1)


#remove property id since it is identity value

df_test = df_test.drop(['PropertyID'],  axis=1)


#converting date into a year as there wouldn't be huge change in tax evaluation within a year

date_col = pd.DatetimeIndex(df_test['DateOfEvaluation'])


date_col
```

```
DatetimeIndex(['2013-06-24', '2014-08-28', '2015-01-31', '2015-12-03',
               '2014-07-06', '2014-04-19', '2014-08-24', '2012-01-08',
               '2012-08-22', '2015-04-03',
               ...
               '2015-02-28', '2014-08-17', '2015-03-20', '2014-10-09',
               '2012-10-23', '2015-02-21', '2014-04-06', '2014-06-08',
               '2014-04-30', '2014-10-25'],
              dtype='datetime64[ns]', name='DateOfEvaluation', length=2936, freq=None)
```

```
df_test['DateOfEvaluation'] = date_col.year


df_test['DateOfEvaluation'].head()
```

```
0    2013
1    2014
2    2015
3    2015
4    2014
Name: DateOfEvaluation, dtype: int64
```

```
df_test.dtypes
```

```
Borough                        int64
ZipCode                        int64
Block                          int64
Lot                          float64
NoOfResidentialUnits           int64
NoOfCommercialUnits            int64
LandAreaInSqFt               float64
GrossAreaInSqFt              float64
YearOfConstruction             int64
TaxClass_AtEvaluationTime      int64
DateOfEvaluation               int64
dtype: object
```

```
c_df_test = df_test
```

```
c_df_test = pd.get_dummies(c_df_test,columns = ['Borough',  'TaxClass_AtEvaluationTime'], drop_first = True)
```

```
c_df_test.head()
```

	ZipCode	Block	Lot	NoOfResidentialUnits	NoOfCommercialUnits	LandAreaInSqFt	GrossAreaInSqFt	YearOfConstruction	DateOfEvaluation	Borough_2	Bor
0	10457	2929	117.0	2	0	2498.0	2394.000	1995	2013	0	
1	10457	3036	65.0	2	0	2033.0	2340.000	1994	2014	0	
2	10457	3048	51.0	4	1	2500.0	4267.625	1963	2015	0	
3	10457	2899	34.0	0	1	3439.0	3320.000	1951	2015	0	
4	10457	3083	40.0	3	0	1348.0	2931.000	1997	2014	0	

```
c_df_test.shape
```

```
(2936, 15)
```

```
X_test = c_df_test
```

```
X_test.shape
```

```
(2936, 15)
```

```
y_test = pd.read_csv('/content/samplesubmission.csv')
```

```
y_test = y_test.drop(['PropertyID'], axis=1)
```

```
y_test
```

	PropertyEvaluationvalue
0	1064701
1	1114550
2	1108206
3	1154822