In [136]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
```

In [98]:
```
data = pd.read_csv('C:/Users/Munazzam/Downloads/nyc-rolling-sales.csv')
```

In [99]:
```
data.head()
```

Out[99]:

	Unnamed: 0	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	EASE-MENT	BUI CL PRI
0	4	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	392	6		
1	5	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	26		
2	6	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	39		
3	7	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	402	21		
4	8	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	404	55		

5 rows × 22 columns

In [100]:
```
#Removing unnecessary columns
del data['EASE-MENT']
del data['Unnamed: 0']
del data['SALE DATE']
del data['ADDRESS']
del data['APARTMENT NUMBER']
```

In [101]:
```
#Checking for duplicates
sum(data.duplicated(data.columns))
```

Out[101]:
```
2871
```

In [102]:

```
#Removing duplicatesrked
data = data.drop_duplicates(data.columns, keep='last')
sum(data.duplicated(data.columns))
```

Out[102]:

```
0
```

In [103]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 81677 entries, 0 to 84547
Data columns (total 17 columns):
BOROUGH                           81677 non-null int64
NEIGHBORHOOD                      81677 non-null object
BUILDING CLASS CATEGORY           81677 non-null object
TAX CLASS AT PRESENT              81677 non-null object
BLOCK                             81677 non-null int64
LOT                               81677 non-null int64
BUILDING CLASS AT PRESENT         81677 non-null object
ZIP CODE                          81677 non-null int64
RESIDENTIAL UNITS                 81677 non-null int64
COMMERCIAL UNITS                  81677 non-null int64
TOTAL UNITS                       81677 non-null int64
LAND SQUARE FEET                  81677 non-null object
GROSS SQUARE FEET                 81677 non-null object
YEAR BUILT                        81677 non-null int64
TAX CLASS AT TIME OF SALE         81677 non-null int64
BUILDING CLASS AT TIME OF SALE    81677 non-null object
SALE PRICE                        81677 non-null object
dtypes: int64(9), object(8)
memory usage: 11.2+ MB
```

In [104]:

```
#Convert some of the columns to desired datatype
data['TAX CLASS AT TIME OF SALE'] = data['TAX CLASS AT TIME OF SALE'].as
type('category')
data['TAX CLASS AT PRESENT'] = data['TAX CLASS AT PRESENT'].astype('cate
gory')
data['LAND SQUARE FEET'] = pd.to_numeric(data['LAND SQUARE FEET'], error
s='coerce')
data['GROSS SQUARE FEET']= pd.to_numeric(data['GROSS SQUARE FEET'], erro
rs='coerce')
#data['SALE DATE'] = pd.to_datetime(data['SALE DATE'], errors='coerce')
data['SALE PRICE'] = pd.to_numeric(data['SALE PRICE'], errors='coerce')
data['BOROUGH'] = data['BOROUGH'].astype('category')
```

In [105]:

```
#checking missing values
data.columns[data.isnull().any()]
```

Out[105]:

```
Index(['LAND SQUARE FEET', 'GROSS SQUARE FEET', 'SALE PRICE'], dtype='o
bject')
```

In [106]:

```
miss=data.isnull().sum()/len(data)
miss=miss[miss>0]
miss.sort_values(inplace=True)
miss
```

Out[106]:

```
SALE PRICE           0.162592
LAND SQUARE FEET     0.305325
GROSS SQUARE FEET    0.321155
dtype: float64
```

In [107]:

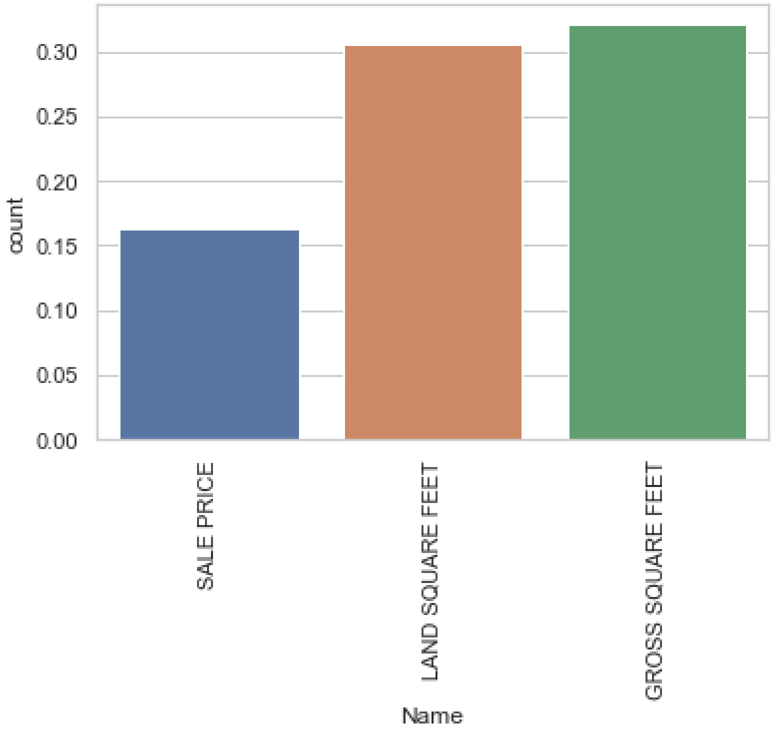
```
#Convert series to column DataFrame
miss=miss.to_frame()
#Set Column Name
miss.columns=['count']
#Set Index Name
miss.index.names=['Name']
#Create Column from Index
miss['Name']=miss.index
miss
```

Out[107]:

	count	**Name**
Name		
SALE PRICE	0.162592	SALE PRICE
LAND SQUARE FEET	0.305325	LAND SQUARE FEET
GROSS SQUARE FEET	0.321155	GROSS SQUARE FEET

In [108]:
```
#Plot the missing values
sns.set(style='whitegrid',color_codes=True)
sns.barplot(x='Name', y='count',data=miss)
plt.xticks(rotation=90)
sns
```

Out[108]:
```
<module 'seaborn' from 'C:\\Users\\Munazzam\\Anaconda3\\lib\\site-packages\\seaborn\\__init__.py'>
```



In [109]:
```
#Populating mean values for missing data
data['LAND SQUARE FEET']=data['LAND SQUARE FEET'].fillna(data['LAND SQUARE FEET'].mean())
data['GROSS SQUARE FEET']=data['GROSS SQUARE FEET'].fillna(data['GROSS SQUARE FEET'].mean())
```

In [110]:
```
# Splitting dataset
test=data[data['SALE PRICE'].isna()]
df=data[data['SALE PRICE'].isna()]
```

In [111]:
```
test = test.drop(columns='SALE PRICE')
```

In [112]:
```
print(test.shape)
test.head()
```

```
(13280, 16)
```

Out[112]:

	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	BUILDING CLASS AT PRESENT	ZIP CODE	RES
1	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	26	C7	10009	
2	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	39	C7	10009	
5	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	405	16	C4	10009	
7	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	407	18	C7	10009	
8	1	ALPHABET CITY	08 RENTALS - ELEVATOR APARTMENTS	2	379	34	D5	10009	

In [113]:

```
print(data.shape)
data.head(10)
```

```
(81677, 17)
```

Out[113]:

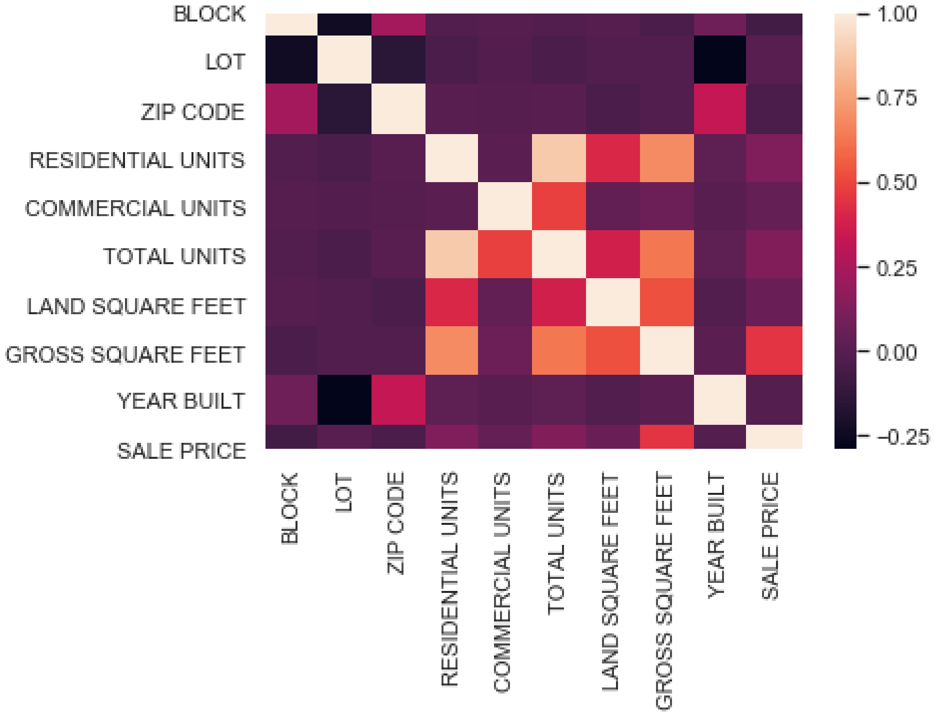
	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT	BLOCK	LOT	BUILDING CLASS AT PRESENT	ZIP CODE	RES
0	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	392	6	C2	10009	
1	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	26	C7	10009	
2	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	399	39	C7	10009	
3	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	402	21	C4	10009	
4	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2A	404	55	C2	10009	
5	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	405	16	C4	10009	
6	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2B	406	32	C4	10009	
7	1	ALPHABET CITY	07 RENTALS - WALKUP APARTMENTS	2	407	18	C7	10009	
8	1	ALPHABET CITY	08 RENTALS - ELEVATOR APARTMENTS	2	379	34	D5	10009	
9	1	ALPHABET CITY	08 RENTALS - ELEVATOR APARTMENTS	2	387	153	D9	10009	

In [114]:

```
#correlation between the features
corr = data.corr()
sns.heatmap(corr)
```

Out[114]:

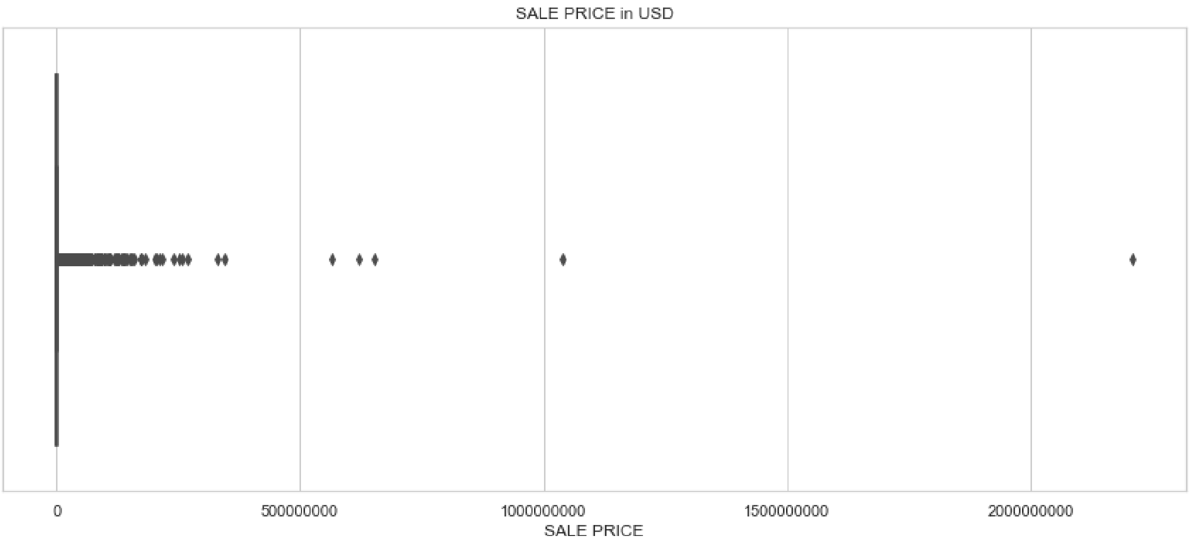
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ebcd094e08>
```



In [115]:

```
plt.figure(figsize=(15,6))

sns.boxplot(x='SALE PRICE', data=data)
plt.ticklabel_format(style='plain', axis='x')
plt.title('SALE PRICE in USD')
plt.show()
```



In [116]:

```
corr['SALE PRICE'].sort_values(ascending=False)
```

Out[116]:

```
SALE PRICE             1.000000
GROSS SQUARE FEET      0.453167
TOTAL UNITS            0.130027
RESIDENTIAL UNITS      0.127388
LAND SQUARE FEET       0.060080
COMMERCIAL UNITS       0.044521
LOT                    0.011888
YEAR BUILT            -0.003523
ZIP CODE              -0.033904
BLOCK                 -0.061507
Name: SALE PRICE, dtype: float64
```

In [117]:

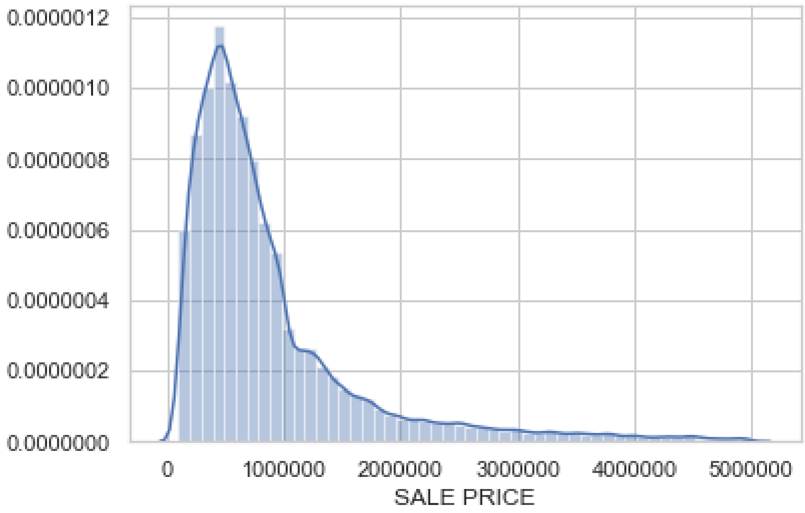
```
#Removing observations
data = data[(data['SALE PRICE'] > 100000) & (data['SALE PRICE'] < 500000
0)]
```

In [118]:

```
sns.distplot(data['SALE PRICE'])
```

Out[118]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ebcd092f48>
```



In [119]:

```
#skewness of SalePrice
data['SALE PRICE'].skew()
```

Out[119]:

```
2.334457139005557
```

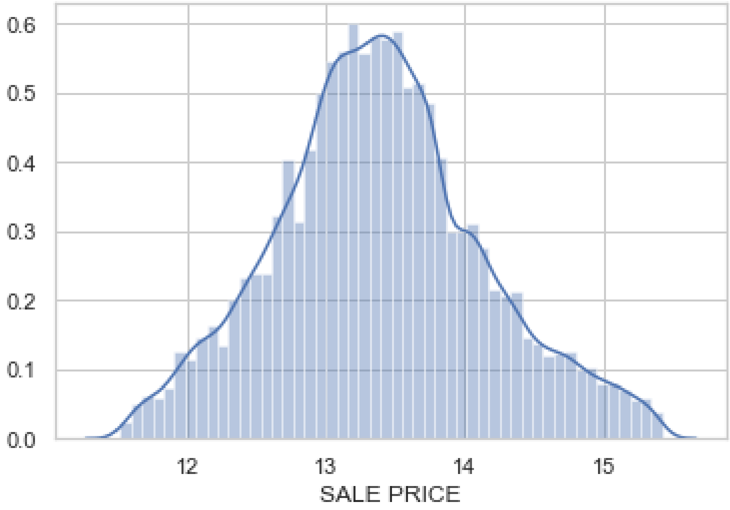
In [120]:

```
#Applying log transform to skew
sales=np.log(data['SALE PRICE'])
print(sales.skew())
sns.distplot(sales)
```

```
0.1976664857746098
```

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ebcd2a1ac8>
```



In [121]:

```
#Removing few columns
del data['BUILDING CLASS AT PRESENT']
del data['BUILDING CLASS AT TIME OF SALE']
del data['NEIGHBORHOOD']
```

In [122]:

```
#Select the variables to be one-hot encoded
one_hot_features = ['BOROUGH', 'BUILDING CLASS CATEGORY','TAX CLASS AT P
RESENT','TAX CLASS AT TIME OF SALE']
```

In [123]:

```
# Convert categorical variables into dummy/indicator variables (i.e. one-hot encoding).
one_hot_encoded = pd.get_dummies(data[one_hot_features])
one_hot_encoded.info(verbose=True, memory_usage=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 53930 entries, 3 to 84545
Data columns (total 62 columns):
BOROUGH_1
53930 non-null uint8
BOROUGH_2
53930 non-null uint8
BOROUGH_3
53930 non-null uint8
BOROUGH_4
53930 non-null uint8
BOROUGH_5
53930 non-null uint8
BUILDING CLASS CATEGORY_01 ONE FAMILY DWELLINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_02 TWO FAMILY DWELLINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_03 THREE FAMILY DWELLINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_04 TAX CLASS 1 CONDOS
53930 non-null uint8
BUILDING CLASS CATEGORY_05 TAX CLASS 1 VACANT LAND
53930 non-null uint8
BUILDING CLASS CATEGORY_06 TAX CLASS 1 - OTHER
53930 non-null uint8
BUILDING CLASS CATEGORY_07 RENTALS - WALKUP APARTMENTS
53930 non-null uint8
BUILDING CLASS CATEGORY_08 RENTALS - ELEVATOR APARTMENTS
53930 non-null uint8
BUILDING CLASS CATEGORY_09 COOPS - WALKUP APARTMENTS
53930 non-null uint8
BUILDING CLASS CATEGORY_10 COOPS - ELEVATOR APARTMENTS
53930 non-null uint8
BUILDING CLASS CATEGORY_11 SPECIAL CONDO BILLING LOTS
53930 non-null uint8
BUILDING CLASS CATEGORY_11A CONDO-RENTALS
53930 non-null uint8
BUILDING CLASS CATEGORY_12 CONDOS - WALKUP APARTMENTS
53930 non-null uint8
BUILDING CLASS CATEGORY_13 CONDOS - ELEVATOR APARTMENTS
53930 non-null uint8
BUILDING CLASS CATEGORY_14 RENTALS - 4-10 UNIT
53930 non-null uint8
BUILDING CLASS CATEGORY_15 CONDOS - 2-10 UNIT RESIDENTIAL
53930 non-null uint8
BUILDING CLASS CATEGORY_16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT
53930 non-null uint8
BUILDING CLASS CATEGORY_17 CONDO COOPS
53930 non-null uint8
BUILDING CLASS CATEGORY_21 OFFICE BUILDINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_22 STORE BUILDINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_23 LOFT BUILDINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_26 OTHER HOTELS
53930 non-null uint8
```

```
BUILDING CLASS CATEGORY_27 FACTORIES
53930 non-null uint8
BUILDING CLASS CATEGORY_28 COMMERCIAL CONDOS
53930 non-null uint8
BUILDING CLASS CATEGORY_29 COMMERCIAL GARAGES
53930 non-null uint8
BUILDING CLASS CATEGORY_30 WAREHOUSES
53930 non-null uint8
BUILDING CLASS CATEGORY_31 COMMERCIAL VACANT LAND
53930 non-null uint8
BUILDING CLASS CATEGORY_32 HOSPITAL AND HEALTH FACILITIES
53930 non-null uint8
BUILDING CLASS CATEGORY_33 EDUCATIONAL FACILITIES
53930 non-null uint8
BUILDING CLASS CATEGORY_35 INDOOR PUBLIC AND CULTURAL FACILITIES
53930 non-null uint8
BUILDING CLASS CATEGORY_36 OUTDOOR RECREATIONAL FACILITIES
53930 non-null uint8
BUILDING CLASS CATEGORY_37 RELIGIOUS FACILITIES
53930 non-null uint8
BUILDING CLASS CATEGORY_38 ASYLUMS AND HOMES
53930 non-null uint8
BUILDING CLASS CATEGORY_39 TRANSPORTATION FACILITIES
53930 non-null uint8
BUILDING CLASS CATEGORY_41 TAX CLASS 4 - OTHER
53930 non-null uint8
BUILDING CLASS CATEGORY_42 CONDO CULTURAL/MEDICAL/EDUCATIONAL/ETC
53930 non-null uint8
BUILDING CLASS CATEGORY_43 CONDO OFFICE BUILDINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_44 CONDO PARKING
53930 non-null uint8
BUILDING CLASS CATEGORY_45 CONDO HOTELS
53930 non-null uint8
BUILDING CLASS CATEGORY_46 CONDO STORE BUILDINGS
53930 non-null uint8
BUILDING CLASS CATEGORY_47 CONDO NON-BUSINESS STORAGE
53930 non-null uint8
BUILDING CLASS CATEGORY_48 CONDO TERRACES/GARDENS/CABANAS
53930 non-null uint8
TAX CLASS AT PRESENT_
53930 non-null uint8
TAX CLASS AT PRESENT_1
53930 non-null uint8
TAX CLASS AT PRESENT_1A
53930 non-null uint8
TAX CLASS AT PRESENT_1B
53930 non-null uint8
TAX CLASS AT PRESENT_1C
53930 non-null uint8
TAX CLASS AT PRESENT_2
53930 non-null uint8
TAX CLASS AT PRESENT_2A
53930 non-null uint8
TAX CLASS AT PRESENT_2B
53930 non-null uint8
TAX CLASS AT PRESENT_2C
```

```
53930 non-null uint8
TAX CLASS AT PRESENT_3
53930 non-null uint8
TAX CLASS AT PRESENT_4
53930 non-null uint8
TAX CLASS AT TIME OF SALE_1
53930 non-null uint8
TAX CLASS AT TIME OF SALE_2
53930 non-null uint8
TAX CLASS AT TIME OF SALE_3
53930 non-null uint8
TAX CLASS AT TIME OF SALE_4
53930 non-null uint8
dtypes: uint8(62)
memory usage: 3.6 MB
```

In [124]:

```
# Replacing categorical columns with dummies
fdf = data.drop(one_hot_features,axis=1)
fdf = pd.concat([fdf, one_hot_encoded] ,axis=1)
```

In [125]:

```
#Train/Test Split
Y_fdf = fdf['SALE PRICE']
X_fdf = fdf.drop('SALE PRICE', axis=1)
X_fdf.shape , Y_fdf.shape
```

Out[125]:

```
((53930, 71), (53930,))
```

In [128]:

```
X_train ,X_test, Y_train , Y_test = train_test_split(X_fdf , Y_fdf , tes
t_size = 0.3 , random_state =34)
```

In [129]:

```
# Training set
X_train.shape , Y_train.shape
```

Out[129]:

```
((37751, 71), (37751,))
```

In [130]:

```
# RMSE
def rmse(y_test,y_pred):
      return np.sqrt(mean_squared_error(y_test,y_pred))
```

In [133]:

```
#Linear Regression
linreg = LinearRegression()
linreg.fit(X_train, Y_train)
Y_pred_lin = linreg.predict(X_test)
rmse(Y_test,Y_pred_lin)
```

Out[133]:

```
619211.2788762044
```

In [137]:

```
#Random Forest
rf_regr = RandomForestRegressor()
rf_regr.fit(X_train, Y_train)
Y_pred_rf = rf_regr.predict(X_test)
rmse(Y_test,Y_pred_rf)
```

```
C:\Users\Munazzam\Anaconda3\lib\site-packages\sklearn\ensemble\forest.p
y:245: FutureWarning: The default value of n_estimators will change fro
m 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[137]:

```
478820.45249953575
```

In [ ]:

```

```