

```

In [96]: import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
dftrain=pd.read_csv("Data Science ZExercise_TRAINING_CONFIDENTIAL1.csv")
dftest = pd.read_csv("Data Science ZExercise_TEST_CONFIDENTIAL2.csv")

```

```

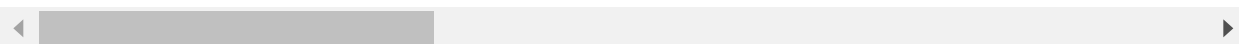
In [97]: dftrain.head()

```

Out[97]:

	PropertyID	SaleDollarCnt	TransDate	censusblockgroup	ZoneCodeCounty	Usecode	BedroomCr
0	48648941	285000.0	5/23/2015	5.300000e+11	R7	9	4.
1	48648982	309950.0	8/22/2015	5.300000e+11	R8P	9	3.
2	48649024	476000.0	8/27/2015	5.300000e+11	SF 7200	9	4.
3	48649040	324950.0	7/1/2015	5.300000e+11	R1	9	4.
4	48649057	325000.0	6/20/2015	5.300000e+11	LDR	9	4.

5 rows × 24 columns



```

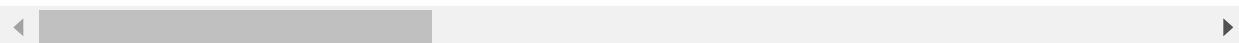
In [98]: dftest.head()

```

Out[98]:

	PropertyID	SaleDollarCnt	TransDate	censusblockgroup	ZoneCodeCounty	Usecode	BedroomCr
0	48735321	NaN	10/31/2015	5.300000e+11	SF 9600	9	5
1	48735471	NaN	11/6/2015	5.300000e+11	SF 9600	9	5
2	49128764	NaN	10/17/2015	5.300000e+11	SF 7200	9	4
3	48897535	NaN	11/19/2015	5.300000e+11	SF 7200	9	2
4	49083957	NaN	12/15/2015	5.300000e+11	SF 9600	9	4

5 rows × 24 columns



▶ In [5]: `dftest.isna().sum()`

```
Out[5]: PropertyID          0
SaleDollarCnt      4402
TransDate          0
censusblockgroup   0
ZoneCodeCounty     0
Usecode            0
BedroomCnt         0
BathroomCnt        0
FinishedSquareFeet 0
GarageSquareFeet   1138
LotSizeSquareFeet  0
StoryCnt           0
BuiltYear          0
ViewType           3404
Latitude           0
Longitude           0
BGMedHomeValue      7
BGMedRent           963
BGMedYearBuilt      62
BGPctOwn            0
BGPctVacant         0
BGMedIncome         0
BGPctKids           0
BGMedAge            0
dtype: int64
```

▶ In [6]: `dftrain.shape`

```
Out[6]: (11588, 24)
```

▶ In [7]: `dftest.shape`

```
Out[7]: (4402, 24)
```

In [8]:

```

# Function to calculate missing values by column# Funct
def missing_values_table(df):
    # Total missing values
    mis_val = df.isnull().sum()

    # Percentage of missing values
    mis_val_percent = 100 * df.isnull().sum() / len(df)

    # Make a table with the results
    mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
    # Rename the columns
    mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

    # Sort the table by percentage of missing descending
    mis_val_table_ren_columns = mis_val_table_ren_columns[
        mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)

    # Print some summary information
    print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
          "There are " + str(mis_val_table_ren_columns.shape[0]) +
          " columns that have missing values.")

    # Return the dataframe with missing information
    return mis_val_table_ren_columns

```

In [9]: missing_values_table(df_test)

Your selected dataframe has 24 columns.
 There are 6 columns that have missing values.

Out[9]:

	Missing Values	% of Total Values
SaleDollarCnt	4402	100.0
ViewType	3404	77.3
GarageSquareFeet	1138	25.9
BGMedRent	963	21.9
BGMedYearBuilt	62	1.4
BGMedHomeValue	7	0.2

► In [10]: `missing_values_table(dftrain)`

Your selected dataframe has 24 columns.
There are 5 columns that have missing values.

Out[10]:

	Missing Values	% of Total Values
ViewType	8956	77.3
GarageSquareFeet	2841	24.5
BGMedRent	2631	22.7
BGMedYearBuilt	247	2.1
BGMedHomeValue	6	0.1

► In [11]: `print("Shape of dftrain is : ",dftrain.shape)
dftrain_y=pd.DataFrame(dftrain['SaleDollarCnt'])
dftrain_X=dftrain.drop(['SaleDollarCnt'],axis=1)
print("Shape of X is : ",dftrain_X.shape)`

Shape of dftrain is : (11588, 24)
Shape of X is : (11588, 23)

► In [12]: `print("Shape of dftest is : ",dftest.shape)
dftest_y=pd.DataFrame(dftest['SaleDollarCnt'])
dftest_X=dftest.drop(['SaleDollarCnt'],axis=1)
print("Shape of X is : ",dftest_X.shape)
print(dftest_y.shape)`

Shape of dftest is : (4402, 24)
Shape of X is : (4402, 23)
(4402, 1)

► In [14]: `dfctest_X.dtypes=="object"`

Out[14]:

PropertyID	False
TransDate	True
censusblockgroup	False
ZoneCodeCounty	True
Usecode	False
BedroomCnt	False
BathroomCnt	False
FinishedSquareFeet	False
GarageSquareFeet	False
LotSizeSquareFeet	False
StoryCnt	False
BuiltYear	False
ViewType	False
Latitude	False
Longitude	False
BGMedHomeValue	False
BGMedRent	False
BGMedYearBuilt	False
BGPctOwn	False
BGPctVacant	False
BGMedIncome	False
BGPctKids	False
BGMedAge	False
dtype:	bool

► In [16]: `dftrain_X["TransDate"] = dftrain_X.TransDate.str.slice(0,1)`
`dftrain_X.head()`

Out[16]:

	PropertyID	TransDate	censusblockgroup	ZoneCodeCounty	Usecode	BedroomCnt	BathroomCn
0	48648941	5	5.300000e+11	R7	9	4.0	2.0
1	48648982	8	5.300000e+11	R8P	9	3.0	2.0
2	48649024	8	5.300000e+11	SF 7200	9	4.0	1.0
3	48649040	7	5.300000e+11	R1	9	4.0	2.2
4	48649057	6	5.300000e+11	LDR	9	4.0	1.7

5 rows × 23 columns



► In [37]: `dftrain_X.ZoneCodeCounty.value_counts()`

```
Out[37]: SF 5000      2243
          R6         1363
          R4         1120
          RA5        622
          R5         607
          SF 7200     465
          R8         416
          SR6        354
          RS7200      270
          R3.5        260
          RS7.2       252
          RSA 6       218
          R1         200
          RA2.5       182
          MU         178
          R6P        145
          R7         144
          RSX 7.2     121
          RS9.6      110
          URPS0       89
          UL7200      84
          R9.6        84
          RS 8.5      84
          LDR         83
          SR4.5       83
          R2          72
          R15         71
          UV          70
          LR1         65
          RS 7.2      55
          ...
          NC365       1
          BO          1
          MUR35       1
          CBS0        1
          LR2 RC      1
          RSX 8.5     1
          AI1         1
          MRG         1
          NCC         1
          RMF         1
          OS2         1
          PLA 3C      1
          R6C         1
          RM18        1
          TC4         1
          T           1
          RSE         1
          RS 11       1
          RM24        1
          IG2 U65     1
          MSC 4       1
          R 40000     1
          IB U85      1
```

SR30	1
PRR	1
MDR	1
GDC	1
RS35.0	1
MC	1
DC	1

Name: ZoneCodeCounty, Length: 178, dtype: int64

► In [38]: `dftest_X.ZoneCodeCounty.value_counts()`

```
Out[38]: SF 5000      818
          R6         499
          R4         424
          RA5        222
          R5         215
          SF 7200    213
          SR6        153
          R8         147
          RS7200     130
          RS7.2      113
          R3.5       99
          RSA 6      90
          RA2.5      66
          R7         59
          R1         58
          MU         56
          R6P        49
          RS9.6      46
          RSX 7.2    45
          UL7200     41
          LDR        40
          RS 8.5     36
          URPSO      30
          R2         29
          LR1        28
          SR4.5      26
          RS 7.2     25
          R2.5       24
          R9.6       23
          R 9600     20
          ...
          PRR        1
          RS         1
          SR30       1
          NC130      1
          TC         1
          NC2P40     1
          SFR 10.0   1
          R7.5       1
          MRRC       1
          MRG        1
          C2         1
          RSA 1      1
          RIN SINGLE F 1
          CR         1
          PR         1
          MU12       1
          UHUCR      1
          TL 10A     1
          RA10PSO    1
          R12P       1
          RMF        1
          RM12       1
          SFD        1
```



```
T          1
MR         1
R18P       1
RA2.5P     1
NC240      1
RM18       1
DNTNMU     1
```

Name: ZoneCodeCounty, Length: 143, dtype: int64

```
► In [41]: zonedf2=pd.DataFrame(data=dftest,columns=['ZoneCodeCounty','SaleDollarCnt'])
zonedf2.head()
```

Out[41]:

	ZoneCodeCounty	SaleDollarCnt
0	SF 9600	NaN
1	SF 9600	NaN
2	SF 7200	NaN
3	SF 7200	NaN
4	SF 9600	NaN

```
► In [42]: zonedf1=pd.DataFrame(data=dftrain,columns=['ZoneCodeCounty','SaleDollarCnt'])
zonedf1.head()
```

Out[42]:

	ZoneCodeCounty	SaleDollarCnt
0	R7	285000.0
1	R8P	309950.0
2	SF 7200	476000.0
3	R1	324950.0
4	LDR	325000.0

```

In [43]: zone1=zonedf1.groupby('ZoneCodeCounty').mean()
zone1["count"]=zonedf1.groupby('ZoneCodeCounty').count()
zone1

```

Out[43]:

	SaleDollarCnt	count
ZoneCodeCounty		
A10	4.618250e+05	16
A35	4.068126e+05	13
AI1	2.825000e+05	1
BO	3.150000e+05	1
C1	2.055750e+05	6
C140	3.967500e+05	2
CB	3.214333e+05	3
CBSO	2.499500e+05	1
CM2	1.549000e+05	1
DC	2.500000e+05	1
DCE	1.900000e+05	1
DUC	2.200000e+05	2
F	2.628250e+05	8
GDC	6.645000e+05	1
I	3.584750e+05	2
IB U85	2.750000e+05	1
IG2 U65	2.695000e+05	1
LDR	4.103939e+05	83
LR1	6.167631e+05	65
LR2	5.393516e+05	38
LR2 RC	3.390000e+05	1
LR3	5.985191e+05	38
LR3 RC	3.410000e+05	2
MC	2.910000e+05	1
MDR	2.500000e+05	1
MFM	7.966693e+05	3
MHO	2.645000e+05	2
MRD	2.472667e+05	9
MRG	3.238800e+05	1
MRM	2.550000e+05	3
...
RSX 35	1.204000e+06	4

ZoneCodeCounty	SaleDollarCnt	count
RSX 7.2	6.920782e+05	121
RSX 8.5	5.380000e+05	1
SF 5000	7.009578e+05	2243
SF 7200	6.808870e+05	465
SF 9600	1.293949e+06	36
SFD	4.138125e+05	8
SFE	7.235929e+05	7
SFR 10.0	1.410000e+06	1
SFS	7.225824e+05	54
SFSL	8.182029e+05	33
SR1	4.458000e+05	6
SR3	4.245069e+05	29
SR30	1.683000e+06	1
SR4.5	3.733027e+05	83
SR6	3.171909e+05	354
SR8	3.361533e+05	32
SVV	5.000000e+05	1
T	2.750000e+05	1
TC	3.836333e+05	3
TC4	4.100000e+05	1
UL15000	3.325000e+05	3
UL7200	2.882157e+05	84
UL9600	1.900000e+05	2
UM2400	2.144750e+05	2
UR	3.343276e+05	45
URPSO	6.732950e+05	89
UV	7.296507e+05	70
UVEV	7.858723e+05	40
WD II	2.525000e+06	2

178 rows × 2 columns

```

In [44]: zone2=zonedf2.groupby('ZoneCodeCounty').mean()
zone2["count"]=zonedf2.groupby('ZoneCodeCounty').count()
zone2

```

Out[44]:

	SaleDollarCnt	count
ZoneCodeCounty		
A10	NaN	0
A35	NaN	0
C1	NaN	0
C2	NaN	0
CR	NaN	0
DC	NaN	0
DCE	NaN	0
DNTNMU	NaN	0
EP	NaN	0
F	NaN	0
LDR	NaN	0
LR1	NaN	0
LR2	NaN	0
LR3	NaN	0
MFM	NaN	0
MR	NaN	0
MRG	NaN	0
MRRC	NaN	0
MRT16	NaN	0
MU	NaN	0
MU12	NaN	0
MUR45	NaN	0
MUR70	NaN	0
NC130	NaN	0
NC240	NaN	0
NC2P40	NaN	0
O	NaN	0
PR	NaN	0
PRR	NaN	0
PUD	NaN	0
...
RSA 6	NaN	0

ZoneCodeCounty	SaleDollarCnt	count
RSA 8	NaN	0
RSLTC	NaN	0
RSX 7.2	NaN	0
RSX 8.5	NaN	0
SF 5000	NaN	0
SF 7200	NaN	0
SF 9600	NaN	0
SFD	NaN	0
SFE	NaN	0
SFR 10.0	NaN	0
SFS	NaN	0
SFSL	NaN	0
SR1	NaN	0
SR3	NaN	0
SR30	NaN	0
SR4.5	NaN	0
SR6	NaN	0
SR8	NaN	0
SVV	NaN	0
T	NaN	0
TC	NaN	0
TL 10A	NaN	0
UHUCR	NaN	0
UL7200	NaN	0
UR	NaN	0
URPSO	NaN	0
US R1	NaN	0
UV	NaN	0
UVEV	NaN	0

143 rows × 2 columns

```
► In [17]: datedf1=pd.DataFrame(data=dftrain,columns=['TransDate','SaleDollarCnt'])
datedf1.head()
```

Out[17]:

	TransDate	SaleDollarCnt
0	5/23/2015	285000.0
1	8/22/2015	309950.0
2	8/27/2015	476000.0
3	7/1/2015	324950.0
4	6/20/2015	325000.0

```
► In [28]: datedf2=pd.DataFrame(data=dftest,columns=['TransDate','SaleDollarCnt'])
datedf2.head()
```

Out[28]:

	TransDate	SaleDollarCnt
0	10/31/2015	NaN
1	11/6/2015	NaN
2	10/17/2015	NaN
3	11/19/2015	NaN
4	12/15/2015	NaN

```
► In [18]: import matplotlib.pyplot as plt
%matplotlib inline

datedf1['TransDate'] = pd.to_datetime(datedf1['TransDate'])
```

```
► In [29]: datedf2['TransDate'] = pd.to_datetime(datedf2['TransDate'])
```

```
► In [20]: datedf1.head()
```

Out[20]:

	TransDate	SaleDollarCnt
0	2015-05-23	285000.0
1	2015-08-22	309950.0
2	2015-08-27	476000.0
3	2015-07-01	324950.0
4	2015-06-20	325000.0

▶ In [30]: datedf2.head()

Out[30]:

	TransDate	SaleDollarCnt
0	2015-10-31	NaN
1	2015-11-06	NaN
2	2015-10-17	NaN
3	2015-11-19	NaN
4	2015-12-15	NaN

▶ In [23]:

```
datedf1=datedf1.sort_values(by='TransDate')
datedf1.head()
```

Out[23]:

	TransDate	SaleDollarCnt
10314	2015-04-01	292500.0
5907	2015-04-01	586000.0
9526	2015-04-01	545000.0
11152	2015-04-01	680000.0
2718	2015-04-01	970000.0

▶ In [31]:

```
datedf2=datedf2.sort_values(by='TransDate')
datedf2.head()
```

Out[31]:

	TransDate	SaleDollarCnt
3072	2015-10-01	NaN
3253	2015-10-01	NaN
3256	2015-10-01	NaN
767	2015-10-01	NaN
3275	2015-10-01	NaN

▶ In [32]:

```
datedf2.index=datedf2['TransDate']
date3=datedf2.resample('M').mean()
date3
date3['count']=datedf2.resample('M').count
date3['count']=datedf2.resample('M').count()
```

► In [33]: date3

Out[33]:

	SaleDollarCnt	count
TransDate		
2015-10-31	NaN	1852
2015-11-30	NaN	1034
2015-12-31	NaN	1450
2016-01-31	NaN	66

► In [26]:

```
datedf1.index=datedf1['TransDate']
date2=datedf1.resample('M').mean()
date2
date2['count']=datedf1.resample('M').count
date2['count']=datedf1.resample('M').count()
```

► In [27]: date2

Out[27]:

	SaleDollarCnt	count
TransDate		
2015-04-30	614949.954518	1671
2015-05-31	614010.690454	2116
2015-06-30	637301.737849	1934
2015-07-31	601015.601758	2275
2015-08-31	613819.837924	1888
2015-09-30	602209.944249	1704

► In [47]:

```
# Drop both date columns
print("X shape is : ",dftrain_X.shape)
dftrain_X.drop("TransDate",axis=1,inplace=True)
print("X shape is : ",dftrain_X.shape)
```

```
X shape is : (11588, 23)
X shape is : (11588, 22)
```

► In [48]:

```
# Drop both date columns
print("X shape is : ",dftest_X.shape)
dftest_X.drop("TransDate",axis=1,inplace=True)
print("X shape is : ",dftest_X.shape)
```

```
X shape is : (4402, 23)
X shape is : (4402, 22)
```



```

In [49]: print("Shape of X is : ",dftrain_X.shape)
dftrain_X=pd.get_dummies(dftrain_X)
print("Shape of X is : ",dftrain_X.shape)
dftrain_X.head()

```

```

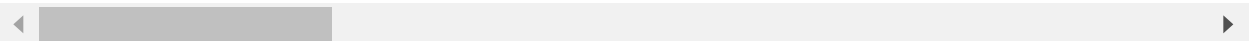
Shape of X is : (11588, 22)
Shape of X is : (11588, 199)

```

Out[49]:

	PropertyID	censusblockgroup	Usecode	BedroomCnt	BathroomCnt	FinishedSquareFeet	Garage
0	48648941	5.300000e+11	9	4.0	2.00	1900.0	
1	48648982	5.300000e+11	9	3.0	2.00	2170.0	
2	48649024	5.300000e+11	9	4.0	1.00	2150.0	
3	48649040	5.300000e+11	9	4.0	2.25	2560.0	
4	48649057	5.300000e+11	9	4.0	1.75	1720.0	

5 rows × 199 columns



```

In [50]: print("Shape of X is : ",dftest_X.shape)
dftest_X=pd.get_dummies(dftrain_X)
print("Shape of X is : ",dftrain_X.shape)
dftrain_X.head()

```

```

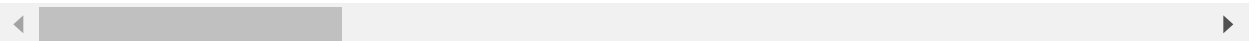
Shape of X is : (4402, 22)
Shape of X is : (4402, 164)

```

Out[50]:

	PropertyID	censusblockgroup	Usecode	BedroomCnt	BathroomCnt	FinishedSquareFeet	Garage
0	48735321	5.300000e+11	9	5.0	4.0	5540	
1	48735471	5.300000e+11	9	5.0	3.0	2470	
2	49128764	5.300000e+11	9	4.0	2.0	1680	
3	48897535	5.300000e+11	9	2.0	1.0	990	
4	49083957	5.300000e+11	9	4.0	3.0	2960	

5 rows × 164 columns



```

In [53]: dftrain_X.Usecode.value_counts()

```

```

Out[53]: 9    11588
Name: Usecode, dtype: int64

```

```

In [54]: dftrain_X.Usecode.value_counts()

```

```

Out[54]: 9    4402
Name: Usecode, dtype: int64

```

▶ In [55]: `dfctest_X.shape`

Out[55]: (4402, 164)

▶ In [56]: `# Drop both columns`
`print("X shape is : ",dftrain_X.shape)`
`dftrain_X.drop("Usecode",axis=1,inplace=True)`
`print("X shape is : ",dftrain_X.shape)`

`# Drop both columns`
`print("X shape is : ",dfctest_X.shape)`
`dfctest_X.drop("Usecode",axis=1,inplace=True)`
`print("X shape is : ",dfctest_X.shape)`

X shape is : (11588, 199)
X shape is : (11588, 198)
X shape is : (4402, 164)
X shape is : (4402, 163)

▶ In [57]: `dftrain_X.censusblockgroup.value_counts()`

Out[57]: 5.300000e+11 11588
Name: censusblockgroup, dtype: int64

▶ In [58]: `dfctest_X.censusblockgroup.value_counts()`

Out[58]: 5.300000e+11 4402
Name: censusblockgroup, dtype: int64

▶ In [59]: `# Drop both columns`
`print("X shape is : ",dftrain_X.shape)`
`dftrain_X.drop("censusblockgroup",axis=1,inplace=True)`
`print("X shape is : ",dftrain_X.shape)`

`# Drop both columns`
`print("X shape is : ",dfctest_X.shape)`
`dfctest_X.drop("censusblockgroup",axis=1,inplace=True)`
`print("X shape is : ",dfctest_X.shape)`

X shape is : (11588, 198)
X shape is : (11588, 197)
X shape is : (4402, 163)
X shape is : (4402, 162)

▶ In [60]: `print("X shape is : ",dfctest_X.shape)`

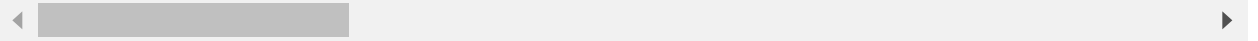
X shape is : (4402, 162)

▶ In [61]: `dftest_X.head()`

Out[61]:

	PropertyID	BedroomCnt	BathroomCnt	FinishedSquareFeet	GarageSquareFeet	LotSizeSquareFee
0	48735321	5.0	4.0	5540	NaN	2533
1	48735471	5.0	3.0	2470	510.0	2600
2	49128764	4.0	2.0	1680	NaN	874
3	48897535	2.0	1.0	990	260.0	1221
4	49083957	4.0	3.0	2960	550.0	2356

5 rows × 162 columns

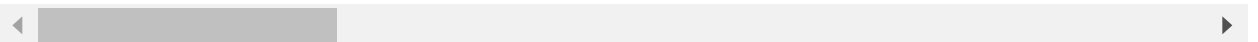


▶ In [62]: `dftrain_X.head()`

Out[62]:

	PropertyID	BedroomCnt	BathroomCnt	FinishedSquareFeet	GarageSquareFeet	LotSizeSquareFee
0	48648941	4.0	2.00	1900.0	480.0	748
1	48648982	3.0	2.00	2170.0	320.0	1420
2	48649024	4.0	1.00	2150.0	590.0	650
3	48649040	4.0	2.25	2560.0	NaN	1576
4	48649057	4.0	1.75	1720.0	NaN	862

5 rows × 197 columns



▶ In [66]: `dftrain_X.columns`

Out[66]: Index(['PropertyID', 'BedroomCnt', 'BathroomCnt', 'FinishedSquareFeet', 'GarageSquareFeet', 'LotSizeSquareFeet', 'StoryCnt', 'BuiltYear', 'ViewType', 'Latitude',
 ...,
 'ZoneCodeCounty_TC4', 'ZoneCodeCounty_UL15000', 'ZoneCodeCounty_UL7200', 'ZoneCodeCounty_UL9600', 'ZoneCodeCounty_UM2400', 'ZoneCodeCounty_UR', 'ZoneCodeCounty_URPSO', 'ZoneCodeCounty_UV', 'ZoneCodeCounty_UVEV', 'ZoneCodeCounty_WD II'],
 dtype='object', length=197)

▶ In []:

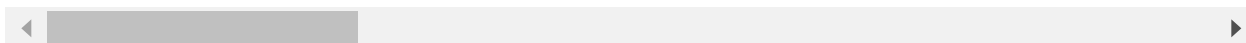
▶ In [67]: `cols = [col for col in dftest_X.columns if col in dftrain_X.columns]
 dftrain_X = dftrain_X[cols]`

► In [69]: `dftrain_X.head()`

Out[69]:

	PropertyID	BedroomCnt	BathroomCnt	FinishedSquareFeet	GarageSquareFeet	LotSizeSquareFeet
0	48648941	4.0	2.00	1900.0	480.0	748
1	48648982	3.0	2.00	2170.0	320.0	1420
2	48649024	4.0	1.00	2150.0	590.0	650
3	48649040	4.0	2.25	2560.0	NaN	1576
4	48649057	4.0	1.75	1720.0	NaN	862

5 rows × 145 columns



► In [70]: `colstoadd = [col for col in dftrain_X.columns if col not in dftrain_X.columns]`
`colstoadd`

Out[70]:

```
['ZoneCodeCounty_C2',
'ZoneCodeCounty_CR',
'ZoneCodeCounty_DNTNMU',
'ZoneCodeCounty_EP',
'ZoneCodeCounty_MR',
'ZoneCodeCounty_MRRC',
'ZoneCodeCounty_NC2P40',
'ZoneCodeCounty_PR',
'ZoneCodeCounty_R1P',
'ZoneCodeCounty_R30',
'ZoneCodeCounty_RA10DPA',
'ZoneCodeCounty_RM2400',
'ZoneCodeCounty_RSA 1',
'ZoneCodeCounty_RSLTC',
'ZoneCodeCounty_TL 10A',
'ZoneCodeCounty_UHUCR',
'ZoneCodeCounty_US R1']
```

▶ In [75]: `dftest_X[colstoadd].sum()`

Out[75]:

ZoneCodeCounty_C2	1
ZoneCodeCounty_CR	1
ZoneCodeCounty_DNTNMU	1
ZoneCodeCounty_EP	2
ZoneCodeCounty_MR	1
ZoneCodeCounty_MRRC	1
ZoneCodeCounty_NC2P40	1
ZoneCodeCounty_PR	1
ZoneCodeCounty_R1P	1
ZoneCodeCounty_R30	2
ZoneCodeCounty_RA10DPA	3
ZoneCodeCounty_RM2400	2
ZoneCodeCounty_RSA 1	1
ZoneCodeCounty_RSLTC	1
ZoneCodeCounty_TL 10A	1
ZoneCodeCounty_UHUCR	1
ZoneCodeCounty_US R1	2

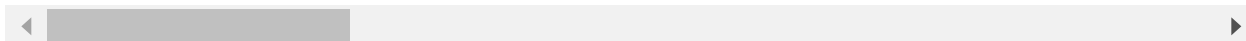
dtype: int64

▶ In [77]: `for col in colstoadd:`
`dftrain_X[col] = 0`
`dftrain_X.head()`

Out[77]:

	PropertyID	BedroomCnt	BathroomCnt	FinishedSquareFeet	GarageSquareFeet	LotSizeSquareFeet
0	48648941	4.0	2.00	1900.0	480.0	748
1	48648982	3.0	2.00	2170.0	320.0	1420
2	48649024	4.0	1.00	2150.0	590.0	650
3	48649040	4.0	2.25	2560.0	NaN	1576
4	48649057	4.0	1.75	1720.0	NaN	862

5 rows × 162 columns



▶ In [78]: `dftrain_X[colstoadd].sum()`

```
Out[78]: ZoneCodeCounty_C2      0
ZoneCodeCounty_CR      0
ZoneCodeCounty_DNTNMU  0
ZoneCodeCounty_EP      0
ZoneCodeCounty_MR      0
ZoneCodeCounty_MRRC    0
ZoneCodeCounty_NC2P40  0
ZoneCodeCounty_PR      0
ZoneCodeCounty_R1P     0
ZoneCodeCounty_R30     0
ZoneCodeCounty_RA10DPA  0
ZoneCodeCounty_RM2400  0
ZoneCodeCounty_RSA 1   0
ZoneCodeCounty_RSLTC   0
ZoneCodeCounty_TL 10A  0
ZoneCodeCounty_UHUCR   0
ZoneCodeCounty_US R1   0
dtype: int64
```

▶ In [83]: `traincolsthatwerenotadded=[col for col in dftrain.columns if col not in dftrain_X.columns]`

▶ In [84]: `traincolsthatwerenotadded`

```
Out[84]: ['SaleDollarCnt', 'TransDate', 'censusblockgroup', 'ZoneCodeCounty', 'Usecode']
```

▶ In [99]: `pqr=pd.DataFrame(data=dftrain)`
`pqr.drop(['SaleDollarCnt', 'TransDate', 'censusblockgroup', 'Usecode'],inplace=True)`

▶ In [100]: `pqr.shape`

```
Out[100]: (11588, 20)
```

▶ In [101]: `xyz=pd.get_dummies(pqr)`
`xyz.shape`

```
Out[101]: (11588, 197)
```

▶ In [102]: `dftrain.shape`

```
Out[102]: (11588, 24)
```

▶ In [108]: `xyz.shape`

```
Out[108]: (11588, 197)
```

▶ In [109]: `traincolsthatwerenotadded=[col for col in xyz.columns if col not in dftrain_X.columns]`

```

In [112]: xyz[traincolsthatwerenotadded].sum()

```

```

Out[112]: ZoneCodeCounty_AI1      1
          ZoneCodeCounty_BO      1
          ZoneCodeCounty_C140     2
          ZoneCodeCounty_CB       3
          ZoneCodeCounty_CBS0     1
          ZoneCodeCounty_CM2      1
          ZoneCodeCounty_DUC      2
          ZoneCodeCounty_GDC      1
          ZoneCodeCounty_I        2
          ZoneCodeCounty_IB U85    1
          ZoneCodeCounty_IG2 U65   1
          ZoneCodeCounty_LR2 RC    1
          ZoneCodeCounty_LR3 RC    2
          ZoneCodeCounty_MC       1
          ZoneCodeCounty_MDR      1
          ZoneCodeCounty_MHO      2
          ZoneCodeCounty_MRD      9
          ZoneCodeCounty_MRM      3
          ZoneCodeCounty_MSC 4     1
          ZoneCodeCounty_MUR      3
          ZoneCodeCounty_MUR35     1
          ZoneCodeCounty_NC365     1
          ZoneCodeCounty_NCC      1
          ZoneCodeCounty_OS2      1
          ZoneCodeCounty_PLA 17    1
          ZoneCodeCounty_PLA 3C    1
          ZoneCodeCounty_PLA 6D    2
          ZoneCodeCounty_PLA 6E    1
          ZoneCodeCounty_R        2
          ZoneCodeCounty_R 2800, OP 1
          ZoneCodeCounty_R 40000   1
          ZoneCodeCounty_R 5400A, OP 2
          ZoneCodeCounty_R150      2
          ZoneCodeCounty_R4C       1
          ZoneCodeCounty_R4P      13
          ZoneCodeCounty_R6C       1
          ZoneCodeCounty_RA3600     1
          ZoneCodeCounty_RB        1
          ZoneCodeCounty_RCC        1
          ZoneCodeCounty_RM1800     3
          ZoneCodeCounty_RM3600     2
          ZoneCodeCounty_R0        3
          ZoneCodeCounty_RS 11      1
          ZoneCodeCounty_RS 6.3     2
          ZoneCodeCounty_RS35.0     1
          ZoneCodeCounty_RSE       1
          ZoneCodeCounty_RSX 35     4
          ZoneCodeCounty_TC4       1
          ZoneCodeCounty_UL15000    3
          ZoneCodeCounty_UL9600     2
          ZoneCodeCounty_UM2400     2
          ZoneCodeCounty_WD II      2
          dtype: int64

```

▶ In [114]: dftrain_X.shape

Out[114]: (11588, 162)

▶ In [115]: dftest_X.shape

Out[115]: (4402, 162)

▶ In [116]: check1=[col for col in dftrain_X.columns if col not in dftest_X.columns]

▶ In [117]: check1

Out[117]: []

▶ In [119]: check2=[col for col in dftrain_X.columns if col in dftest_X.columns]
len(check2)

Out[119]: 162

▶ In [122]: dftrain_X['Missing ViewType']=(np.isfinite(dftrain_X['ViewType'])==False)
dftrain_X['Missing ViewType']= dftrain_X['Missing ViewType'].astype(int)
dftrain_X['Missing GarageSquareFeet']=(np.isfinite(dftrain_X['GarageSquareFeet'])==False)
dftrain_X['Missing GarageSquareFeet']= dftrain_X['Missing GarageSquareFeet'].astype(int)
dftrain_X['Missing BGMedYearBuilt']=(np.isfinite(dftrain_X['BGMedYearBuilt'])==False)
dftrain_X['Missing BGMedYearBuilt']= dftrain_X['Missing BGMedYearBuilt'].astype(int)
dftrain_X['Missing BGMedRent']=(np.isfinite(dftrain_X['BGMedRent'])==False)
dftrain_X['Missing BGMedRent']= dftrain_X['Missing BGMedRent'].astype(int)
dftrain_X['Missing BGMedHomeValue']=(np.isfinite(dftrain_X['BGMedHomeValue'])==False)
dftrain_X['Missing BGMedHomeValue']= dftrain_X['Missing BGMedHomeValue'].astype(int)
print("Shape of train is : ",dftrain_X.shape)
dftrain_X.head()

Shape of train is : (11588, 167)

Out[122]:

inty_RSLTC	ZoneCodeCounty_TL 10A	ZoneCodeCounty_UHUCR	ZoneCodeCounty_US R1	Missing ViewType	Garages
0	0	0	0	1	
0	0	0	0	0	
0	0	0	0	1	
0	0	0	0	0	
0	0	0	0	0	


```

In [123]: dftest_X['Missing ViewType']=(np.isfinite(dftest_X['ViewType'])==False)
dftest_X['Missing ViewType']= dftest_X['Missing ViewType'].astype(int)
dftest_X['Missing GarageSquareFeet']=(np.isfinite(dftest_X['GarageSquareFeet'])==False)
dftest_X['Missing GarageSquareFeet']= dftest_X['Missing GarageSquareFeet'].astype(int)
dftest_X['Missing BGMedYearBuilt']=(np.isfinite(dftest_X['BGMedYearBuilt'])==False)
dftest_X['Missing BGMedYearBuilt']= dftest_X['Missing BGMedYearBuilt'].astype(int)
dftest_X['Missing BGMedRent']=(np.isfinite(dftest_X['BGMedRent'])==False)
dftest_X['Missing BGMedRent']= dftest_X['Missing BGMedRent'].astype(int)
dftest_X['Missing BGMedHomeValue']=(np.isfinite(dftest_X['BGMedHomeValue'])==False)
dftest_X['Missing BGMedHomeValue']= dftest_X['Missing BGMedHomeValue'].astype(int)
print("Shape of test is : ",dftest_X.shape)
dftest_X.head()

```

Shape of test is : (4402, 167)

Out[123]:

County_URPSO	ZoneCodeCounty_US R1	ZoneCodeCounty_UV	ZoneCodeCounty_UVEV	Missing ViewType	GarageSquareFeet
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	0

```

In [124]: dftrain_X['BedroomCnt'].value_counts()

```

```

Out[124]: 3.000000    5107
4.000000    4177
2.000000    1102
5.000000     959
6.000000     114
1.000000     103
7.000000      15
9.000000       5
8.000000       3
3.615385       1
3.384615       1
3.461538       1
Name: BedroomCnt, dtype: int64

```

```
▶ In [125]: dfctest_X['BedroomCnt'].value_counts()
```

```
Out[125]: 3.000000    1942
          4.000000    1570
          2.000000     424
          5.000000     363
          6.000000      49
          1.000000     39
          8.000000       5
          7.000000       4
          3.076923       2
          12.000000       2
          3.538462       1
          3.307692       1
          Name: BedroomCnt, dtype: int64
```

```
▶ In [127]: dftrain_X = dftrain_X.reindex(sorted(dftrain_X.columns), axis=1)
```

```
▶ In [128]: dfctest_X = dfctest_X.reindex(sorted(dfctest_X.columns), axis=1)
```

```
▶ In [129]: dftrain_X.head()
```

```
Out[129]:
```

	BGMedAge	BGMedHomeValue	BGMedIncome	BGMedRent	BGMedYearBuilt	BGPctKids	BGPctt
0	48.6	107800.0	42854	844.0	1975.0	0.1924	0.6
1	42.6	181500.0	54013	925.0	1969.0	0.3718	0.5
2	40.7	344300.0	56782	733.0	1946.0	0.3207	0.6
3	40.0	284200.0	44200	900.0	1977.0	0.3359	0.5
4	44.4	290100.0	65282	802.0	1972.0	0.1633	0.4

5 rows × 167 columns

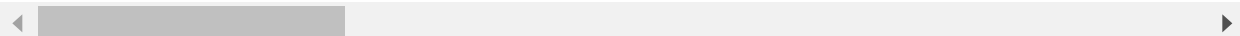


```
▶ In [130]: dfctest_X.head()
```

```
Out[130]:
```

	BGMedAge	BGMedHomeValue	BGMedIncome	BGMedRent	BGMedYearBuilt	BGPctKids	BGPctt
0	49.6	527700.0	113450	1750.0	1956.0	0.2524	0.9
1	49.6	527700.0	113450	1750.0	1956.0	0.2524	0.9
2	49.6	527700.0	113450	1750.0	1956.0	0.2524	0.9
3	49.6	527700.0	113450	1750.0	1956.0	0.2524	0.9
4	49.6	527700.0	113450	1750.0	1956.0	0.2524	0.9

5 rows × 167 columns



```

In [134]: submissiondf=pd.DataFrame(data=dftest_X['PropertyID'])
          submissiondf.head()

```

Out[134]:

	PropertyID
0	48735321
1	48735471
2	49128764
3	48897535
4	49083957

```

In [135]: submissiondf.shape

```

Out[135]: (4402, 1)

KNN imputation from fancy_impute was done on Kaggle Kernel as local system was facing some errors with KNN imputation technique. After that, the files was stored from Server to local PC and then we have KNN imputed files.

```

In [137]: dftrain_X=pd.read_csv("dftrain_X_imputed.csv")
          dftest_X = pd.read_csv("dftest_X_imputed.csv")

```

```

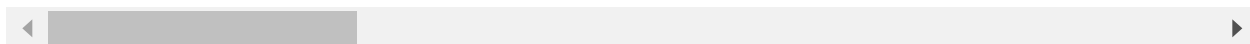
In [138]: dftrain_X.head()

```

Out[138]:

	Unnamed: 0	BGMedAge	BGMedHomeValue	BGMedIncome	BGMedRent	BGMedYearBuilt	BGPctK
0	0	1.312359	-1.829768	-1.433279	-0.992881	0.092411	-1.193
1	1	0.420317	-1.416003	-1.125734	-0.787479	-0.244938	0.083
2	2	0.137837	-0.502016	-1.049419	-1.274357	-1.538110	-0.280
3	3	0.033766	-0.839428	-1.396183	-0.850874	0.204861	-0.171
4	4	0.687930	-0.806304	-0.815157	-1.099385	-0.076264	-1.400

5 rows × 168 columns

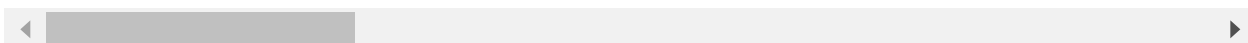


▶ In [139]: `dftest_X.head()`

Out[139]:

	Unnamed: 0	BGMedAge	BGMedHomeValue	BGMedIncome	BGMedRent	BGMedYearBuilt	BGPctK
0	0	1.442128	0.568365	0.557939	1.356873	-0.940332	-0.739
1	1	1.442128	0.568365	0.557939	1.356873	-0.940332	-0.739
2	2	1.442128	0.568365	0.557939	1.356873	-0.940332	-0.739
3	3	1.442128	0.568365	0.557939	1.356873	-0.940332	-0.739
4	4	1.442128	0.568365	0.557939	1.356873	-0.940332	-0.739

5 rows × 168 columns



▶ In [143]: `dftrain_X.drop(["Unnamed: 0"],axis=1,inplace=True)`

▶ In []:

▶ In [144]: `dftest_X.drop(["Unnamed: 0"],axis=1,inplace=True)`

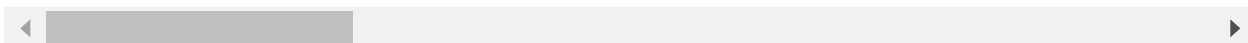
▶ In [145]: `dftrain_X = dftrain_X.reindex(sorted(dftrain_X.columns), axis=1)`
`dftest_X = dftest_X.reindex(sorted(dftest_X.columns), axis=1)`

▶ In [146]: `dftrain_X.head()`

Out[146]:

	BGMedAge	BGMedHomeValue	BGMedIncome	BGMedRent	BGMedYearBuilt	BGPctKids	BGPctK
0	1.312359	-1.829768	-1.433279	-0.992881	0.092411	-1.193404	-0.403
1	0.420317	-1.416003	-1.125734	-0.787479	-0.244938	0.083579	-0.878
2	0.137837	-0.502016	-1.049419	-1.274357	-1.538110	-0.280155	-0.584
3	0.033766	-0.839428	-1.396183	-0.850874	0.204861	-0.171960	-1.030
4	0.687930	-0.806304	-0.815157	-1.099385	-0.076264	-1.400540	-1.635

5 rows × 167 columns



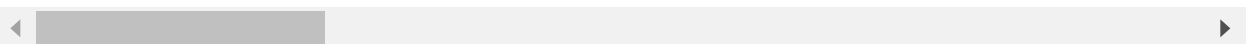
```
In [147]: result = pd.concat([dftrain_y, dftrain_X], axis=1, sort=False)
result.corr()
```

Out[147]:

	SaleDollarCnt	BGMedAge	BGMedHomeValue	BGMedIncome	BGMedRent
SaleDollarCnt	1.000000	0.173956	0.681915	0.427578	0.285612
BGMedAge	0.173956	1.000000	0.262665	0.159478	0.104708
BGMedHomeValue	0.681915	0.262665	1.000000	0.684423	0.458181
BGMedIncome	0.427578	0.159478	0.684423	1.000000	0.621630
BGMedRent	0.285612	0.104708	0.458181	0.621630	1.000000
BGMedYearBuilt	-0.116480	-0.153470	-0.121641	0.193459	0.233537
BGPctKids	-0.028768	-0.473199	0.052913	0.367387	0.273633
BGPctOwn	0.094028	0.357654	0.254099	0.602389	0.459024
BGPctVacant	0.010543	0.039717	-0.046364	-0.096249	-0.094842
BathroomCnt	0.506672	0.060423	0.316697	0.352881	0.246985
BedroomCnt	0.310897	0.022350	0.185010	0.203819	0.160628
BuiltYear	0.139941	-0.053678	-0.002238	0.213867	0.209572
FinishedSquareFeet	0.678446	0.126701	0.451460	0.421043	0.286116
GarageSquareFeet	0.282781	0.102894	0.177867	0.287018	0.221304
Latitude	0.317772	0.093957	0.438911	0.239937	0.178521
Longitude	-0.020657	-0.080426	0.042980	0.337984	0.270668
LotSizeSquareFeet	0.067874	0.128735	0.042843	0.054269	-0.036392
Missing BGMedHomeValue	-0.009463	-0.049708	-0.019036	-0.023734	-0.015104
Missing BGMedRent	0.050122	0.221339	0.162539	0.370692	0.212484
Missing BGMedYearBuilt	0.015801	-0.104180	0.064138	0.160738	0.165488
Missing GarageSquareFeet	-0.081234	0.020496	-0.025990	-0.189072	-0.202730
Missing ViewType	-0.265936	-0.178623	-0.218149	-0.089974	-0.017195
PropertyID	0.024807	-0.121188	-0.070863	0.023011	0.019197
StoryCnt	0.267300	-0.082155	0.181544	0.212164	0.118088
ViewType	0.030121	0.005822	0.065888	0.173572	0.164393
ZoneCodeCounty_A10	-0.012339	0.036878	-0.010913	-0.006664	0.001983
ZoneCodeCounty_A35	-0.015148	0.051685	-0.013500	-0.016842	-0.013628
ZoneCodeCounty_C1	-0.020294	-0.018238	-0.028867	-0.027676	-0.019935
ZoneCodeCounty_C2	NaN	NaN	NaN	NaN	NaN
ZoneCodeCounty_CR	NaN	NaN	NaN	NaN	NaN
...
ZoneCodeCounty_RSA 6	-0.017218	-0.031341	-0.026786	-0.019406	0.043847

	SaleDollarCnt	BGMedAge	BGMedHomeValue	BGMedIncome	BGMedRent
ZoneCodeCounty_RSA 8	0.004212	-0.012531	-0.003531	-0.001087	0.018307
ZoneCodeCounty_RSLTC	NaN	NaN	NaN	NaN	NaN
ZoneCodeCounty_RSX 7.2	0.017586	-0.048495	-0.002511	-0.012116	0.034779
ZoneCodeCounty_RSX 8.5	-0.001537	-0.005211	-0.011561	-0.006611	-0.000894
ZoneCodeCounty_SF 5000	0.093375	-0.039478	0.146622	-0.113508	-0.195087
ZoneCodeCounty_SF 7200	0.030004	0.129687	0.031195	-0.070525	-0.055253
ZoneCodeCounty_SF 9600	0.082959	0.057907	0.063362	0.048893	0.042489
ZoneCodeCounty_SFD	-0.011479	0.004991	-0.013741	-0.027947	-0.025582
ZoneCodeCounty_SFE	0.005902	-0.017707	0.032653	-0.007318	-0.001894
ZoneCodeCounty_SFR 10.0	0.016161	0.005976	0.013637	0.015782	0.011973
ZoneCodeCounty_SFS	0.016274	0.050876	0.036276	0.022181	-0.016802
ZoneCodeCounty_SFSL	0.023874	-0.007128	0.029884	0.057119	-0.030391
ZoneCodeCounty_SR1	-0.008349	-0.001657	-0.012559	-0.005732	0.000145
ZoneCodeCounty_SR3	-0.020704	-0.022395	-0.028158	-0.031804	0.011278
ZoneCodeCounty_SR30	0.021701	0.006253	0.029541	0.010661	0.016969
ZoneCodeCounty_SR4.5	-0.044610	0.019056	-0.055431	-0.000128	0.002170
ZoneCodeCounty_SR6	-0.114994	-0.105985	-0.156554	-0.097459	-0.017317
ZoneCodeCounty_SR8	-0.031909	-0.002428	-0.049825	-0.041919	-0.010297
ZoneCodeCounty_SVV	-0.002308	-0.006592	-0.005740	-0.010876	-0.008341
ZoneCodeCounty_T	-0.006874	-0.000239	-0.007518	-0.011348	-0.011522
ZoneCodeCounty_TC	-0.008089	0.012506	-0.014412	-0.019103	-0.017006
ZoneCodeCounty_TL 10A	NaN	NaN	NaN	NaN	NaN
ZoneCodeCounty_UHUCR	NaN	NaN	NaN	NaN	NaN
ZoneCodeCounty_UL7200	-0.060764	-0.016201	-0.095693	-0.097016	-0.054025
ZoneCodeCounty_UR	-0.038110	0.003449	-0.035577	-0.000763	0.023170
ZoneCodeCounty_URPSO	0.011451	0.124730	0.032169	0.019685	0.119470
ZoneCodeCounty_US R1	NaN	NaN	NaN	NaN	NaN
ZoneCodeCounty_UV	0.019745	-0.076844	0.035798	0.062884	0.085470
ZoneCodeCounty_UVEV	0.022135	-0.046138	0.057218	0.052581	0.025389

168 rows × 168 columns



```
▶ In [149]: dftrain_X.isna().sum().sum()
```

```
Out[149]: 0
```

All Data preprocessing complete

Machine Learning Modeling Started

```

In [167]: from lightgbm import LGBMRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
import numpy as np
from xgboost import XGBRegressor
from sklearn import ensemble
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
def cross_validation_function4(X,y,models,fold,w):
    size=int(len(X)/fold)
    l=0
    u=size
    scores=[]
    for f in range(fold):
#         print("L =",l,"u =",u,"")
        dfxtest=X.iloc[l:u,:]
        dfytest=y.iloc[l:u,:]
        dfxtrain=X.drop(X.iloc[l:u].index,axis=0)
        dfytrain=y.drop(X.iloc[l:u].index,axis=0)

        y_true=pd.Series(dfytest.iloc[:,0])

        model1.fit(dfxtrain,dfytrain.values.ravel())
        model2.fit(dfxtrain,dfytrain.values.ravel())
        model3.fit(dfxtrain,dfytrain.values.ravel())
        model4.fit(dfxtrain,dfytrain.values.ravel())
        model5.fit(dfxtrain,dfytrain.values.ravel())
        model6.fit(dfxtrain,dfytrain.values.ravel())
        model7.fit(dfxtrain,dfytrain.values.ravel())
        model8.fit(dfxtrain,dfytrain.values.ravel())
        model9.fit(dfxtrain,dfytrain.values.ravel())
        model10.fit(dfxtrain,dfytrain.values.ravel())
        model11.fit(dfxtrain,dfytrain.values.ravel())

        y_pred1=model1.predict(dfxtest)
        y_pred2=model2.predict(dfxtest)
        y_pred3=model3.predict(dfxtest)
        y_pred4=model4.predict(dfxtest)
        y_pred5=model5.predict(dfxtest)
        y_pred6=model6.predict(dfxtest)
        y_pred7=model7.predict(dfxtest)
        y_pred8=model8.predict(dfxtest)
        y_pred9=model9.predict(dfxtest)
        y_pred10=model10.predict(dfxtest)
        y_pred11=model11.predict(dfxtest)

        p=[0]*12
        for i in range(0,len(w)):
            p[i+1]=w[i]

        y_pred=(p[1]*y_pred1 + p[2]*y_pred2 + p[3]*y_pred3 + p[4]*y_pred4 + p[5]*y
        sc=np.mean(np.abs((y_true - y_pred) / y_true))
        scores.append(sc)
        print("          FOLD :",f)

```



```
print("      Score :", sc)
l=l+size
u=u+size
print("Final score is: ",np.mean(scores))
return np.mean(scores)
```

► In []:

► In [168]: `w=[1/11]*11`

```
model1 = LGBMRegressor(boosting_type='dart', max_depth=8, learning_rate=0.13, n_estimators=100)
model2 = LGBMRegressor(boosting_type='dart', max_depth=9, learning_rate=0.19, n_estimators=100)
model3 = LGBMRegressor(boosting_type='dart', max_depth=8, learning_rate=0.24, n_estimators=100)
model4 = LGBMRegressor(boosting_type='dart', num_iterations=900, max_depth=7, learning_rate=0.13, n_estimators=100)
model5 = LGBMRegressor(boosting_type='dart', num_leaves=29, min_data_in_leaf=15, max_depth=7, learning_rate=0.13, n_estimators=100)
model6 = LGBMRegressor(boosting_type='dart', num_leaves=25, max_depth=7, learning_rate=0.13, n_estimators=100)
model7 = ensemble.GradientBoostingRegressor(n_estimators=300, max_depth=7, learning_rate=0.13, n_estimators=100)
model8 = XGBRegressor(max_depth=8, learning_rate=0.04, n_estimators=500, booster='gbtree')
model9 = XGBRegressor(max_depth=8, learning_rate=0.081, n_estimators=100, booster='gbtree')
model10 = RandomForestRegressor(max_depth=18, random_state=30, n_estimators=91)
model11 = KNeighborsRegressor(n_neighbors=6, weights="distance")
```

```
model1.fit(dftrain_X, dftrain_y.values.ravel())
model2.fit(dftrain_X, dftrain_y.values.ravel())
model3.fit(dftrain_X, dftrain_y.values.ravel())
model4.fit(dftrain_X, dftrain_y.values.ravel())
model5.fit(dftrain_X, dftrain_y.values.ravel())
model6.fit(dftrain_X, dftrain_y.values.ravel())
model7.fit(dftrain_X, dftrain_y.values.ravel())
model8.fit(dftrain_X, dftrain_y.values.ravel())
model9.fit(dftrain_X, dftrain_y.values.ravel())
model10.fit(dftrain_X, dftrain_y.values.ravel())
model11.fit(dftrain_X, dftrain_y.values.ravel())
```

```
y_pred1=model1.predict(dftrain_X)
y_pred2=model2.predict(dftrain_X)
y_pred3=model3.predict(dftrain_X)
y_pred4=model4.predict(dftrain_X)
y_pred5=model5.predict(dftrain_X)
y_pred6=model6.predict(dftrain_X)
y_pred7=model7.predict(dftrain_X)
y_pred8=model8.predict(dftrain_X)
y_pred9=model9.predict(dftrain_X)
y_pred10=model10.predict(dftrain_X)
y_pred11=model11.predict(dftrain_X)
```

```
y_true=pd.Series(dftrain_y.iloc[:,0])
```

```
p=[0]*12
```

```
for i in range(0, len(w)):
```

```
    p[i+1]=w[i]
```

```
y_pred=(p[1]*y_pred1 + p[2]*y_pred2 + p[3]*y_pred3 + p[4]*y_pred4 + p[5]*y_pred5 + p[6]*y_pred6 + p[7]*y_pred7 + p[8]*y_pred8 + p[9]*y_pred9 + p[10]*y_pred10 + p[11]*y_pred11)
```

```
print("Training Error : ", np.mean(np.abs((y_true - y_pred) / y_true)))
```

```
model1 = LGBMRegressor(boosting_type='dart', max_depth=8, learning_rate=0.13, n_estimators=100)
model2 = LGBMRegressor(boosting_type='dart', max_depth=9, learning_rate=0.19, n_estimators=100)
model3 = LGBMRegressor(boosting_type='dart', max_depth=8, learning_rate=0.24, n_estimators=100)
model4 = LGBMRegressor(boosting_type='dart', num_iterations=900, max_depth=7, learning_rate=0.13, n_estimators=100)
model5 = LGBMRegressor(boosting_type='dart', num_leaves=29, min_data_in_leaf=15, max_depth=7, learning_rate=0.13, n_estimators=100)
model6 = LGBMRegressor(boosting_type='dart', num_leaves=25, max_depth=7, learning_rate=0.13, n_estimators=100)
model7 = ensemble.GradientBoostingRegressor(n_estimators=300, max_depth=7, learning_rate=0.13, n_estimators=100)
```

```

model8 = XGBRegressor(max_depth=8,learning_rate=0.04, n_estimators=500,booster='gbt')
model9 = XGBRegressor(max_depth=8,learning_rate=0.081, n_estimators=100,booster='gbt')
model10 = RandomForestRegressor(max_depth=18, random_state=30,n_estimators=91)
model11= KNeighborsRegressor(n_neighbors=6,weights="distance")

listofmodels=[model1,model2,model3,model4,model5,model6,model7,model8,model9,model10,model11]

newscore=cross_validation_function4(dftrain_X,dftrain_y,listofmodels,5,w)
print("Testing Cross Validation Error: ",newscore)

```

C:\ProgramData\Anaconda3\lib\site-packages\lightgbm\engine.py:116: UserWarning: Found `num_iterations` in params. Will use it instead of argument
 warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))

Training Error : 0.07195523378492123

C:\ProgramData\Anaconda3\lib\site-packages\lightgbm\engine.py:116: UserWarning: Found `num_iterations` in params. Will use it instead of argument
 warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))

FOLD : 0
 Score : 0.1275520988360718

C:\ProgramData\Anaconda3\lib\site-packages\lightgbm\engine.py:116: UserWarning: Found `num_iterations` in params. Will use it instead of argument
 warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))

FOLD : 1
 Score : 0.1313034818099944

C:\ProgramData\Anaconda3\lib\site-packages\lightgbm\engine.py:116: UserWarning: Found `num_iterations` in params. Will use it instead of argument
 warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))

FOLD : 2
 Score : 0.12364673538951014

C:\ProgramData\Anaconda3\lib\site-packages\lightgbm\engine.py:116: UserWarning: Found `num_iterations` in params. Will use it instead of argument
 warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))

FOLD : 3
 Score : 0.1308716112652768

C:\ProgramData\Anaconda3\lib\site-packages\lightgbm\engine.py:116: UserWarning: Found `num_iterations` in params. Will use it instead of argument
 warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))

FOLD : 4
 Score : 0.12847116418944649

Final score is: 0.1283690182980599

Testing Cross Validation Error: 0.1283690182980599

▶ In []: `w=[1/11]*11`

Final Model

► In [187]: w=[1/10]*10

```
model1 = LGBMRegressor(boosting_type='dart', max_depth=8, learning_rate=0.13, n_estimators=100)
model2 = LGBMRegressor(boosting_type='dart', max_depth=9, learning_rate=0.19, n_estimators=100)
model3 = LGBMRegressor(boosting_type='dart', max_depth=8, learning_rate=0.24, n_estimators=100)
model4 = LGBMRegressor(boosting_type='dart', num_iterations=900, max_depth=7, learning_rate=0.13, n_estimators=100)
model5 = LGBMRegressor(boosting_type='dart', num_leaves=29, min_data_in_leaf=15, max_depth=7, learning_rate=0.13, n_estimators=100)
model6 = LGBMRegressor(boosting_type='dart', num_leaves=25, max_depth=7, learning_rate=0.13, n_estimators=100)
model7 = ensemble.GradientBoostingRegressor(n_estimators=300, max_depth=7, learning_rate=0.13, n_estimators=100)
model8 = XGBRegressor(max_depth=8, learning_rate=0.04, n_estimators=500, booster='gbtree')
model9 = XGBRegressor(max_depth=8, learning_rate=0.081, n_estimators=100, booster='gbtree')
model10 = RandomForestRegressor(max_depth=18, random_state=30, n_estimators=91)
```

```
model1.fit(dftrain_X, dftrain_y.values.ravel())
model2.fit(dftrain_X, dftrain_y.values.ravel())
model3.fit(dftrain_X, dftrain_y.values.ravel())
model4.fit(dftrain_X, dftrain_y.values.ravel())
model5.fit(dftrain_X, dftrain_y.values.ravel())
model6.fit(dftrain_X, dftrain_y.values.ravel())
model7.fit(dftrain_X, dftrain_y.values.ravel())
model8.fit(dftrain_X, dftrain_y.values.ravel())
model9.fit(dftrain_X, dftrain_y.values.ravel())
model10.fit(dftrain_X, dftrain_y.values.ravel())
```

```
y_pred1=model1.predict(dftest_X)
y_pred2=model2.predict(dftest_X)
y_pred3=model3.predict(dftest_X)
y_pred4=model4.predict(dftest_X)
y_pred5=model5.predict(dftest_X)
y_pred6=model6.predict(dftest_X)
y_pred7=model7.predict(dftest_X)
y_pred8=model8.predict(dftest_X)
y_pred9=model9.predict(dftest_X)
y_pred10=model10.predict(dftest_X)
```

```
# y_true=pd.Series(dftest_y.iloc[:,0])
```

```
p=[0]*11
```

```
print("p is :", p)
```

```
for i in range(0, len(w)):
```

```
    p[i+1]=w[i]
```

```
print("p is :", p)
```

```
y_pred=(p[1]*y_pred1 + p[2]*y_pred2 + p[3]*y_pred3 + p[4]*y_pred4 + p[5]*y_pred5 + p[6]*y_pred6 + p[7]*y_pred7 + p[8]*y_pred8 + p[9]*y_pred9 + p[10]*y_pred10)
```

```
# print("Training Error : ", np.mean(np.abs((y_true - y_pred) / y_true)))
```

C:\ProgramData\Anaconda3\lib\site-packages\lightgbm\engine.py:116: UserWarning: Found `num_iterations` in params. Will use it instead of argument
warnings.warn("Found `{}` in params. Will use it instead of argument".format(alias))

```
p is : [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
p is : [0, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
```

```
▶ In [188]: submissiondf['SaleDollarCnt']=y_pred
```

```
▶ In [189]: submissiondf.head()
```

Out[189]:

	PropertyID	SaleDollarCnt
0	48735321	2.098852e+06
1	48735471	1.021119e+06
2	49128764	5.483433e+05
3	48897535	4.510677e+05
4	49083957	1.173107e+06

```
▶ In [190]: submissiondf.isna().sum()
```

Out[190]:

PropertyID	0
SaleDollarCnt	0
dtype: int64	

```
▶ In [191]: submissiondf.to_csv("Zillow_submission.csv",index=False)
```

```
▶ In [192]: temp=pd.read_csv("Zillow_submission.csv")
```

```
▶ In [194]: temp.head()
```

Out[194]:

	PropertyID	SaleDollarCnt
0	48735321	2.098852e+06
1	48735471	1.021119e+06
2	49128764	5.483433e+05
3	48897535	4.510677e+05
4	49083957	1.173107e+06

▶ In [193]: temp.describe()

Out[193]:

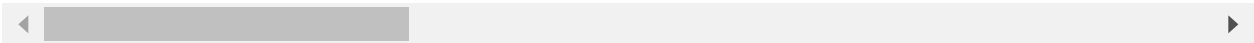
	PropertyID	SaleDollarCnt
count	4.402000e+03	4.402000e+03
mean	5.348500e+07	6.104573e+05
std	1.363566e+07	4.168253e+05
min	4.864910e+07	1.731052e+05
25%	4.880002e+07	3.611430e+05
50%	4.894242e+07	5.103534e+05
75%	4.909140e+07	7.216855e+05
max	1.244396e+08	5.768310e+06

▶ In [186]: dftrain.describe()

Out[186]:

	PropertyID	SaleDollarCnt	censusblockgroup	Usecode	BedroomCnt	BathroomCnt	Finis
count	1.158800e+04	1.158800e+04	1.158800e+04	11588.0	11588.000000	11588.000000	
mean	5.502866e+07	6.137157e+05	5.300000e+11	9.0	3.451800	2.327628	
std	1.605832e+07	4.577593e+05	0.000000e+00	0.0	0.865682	0.872601	
min	4.864894e+07	2.000000e+04	5.300000e+11	9.0	1.000000	0.750000	
25%	4.880374e+07	3.550000e+05	5.300000e+11	9.0	3.000000	1.750000	
50%	4.895489e+07	5.050000e+05	5.300000e+11	9.0	3.000000	2.500000	
75%	4.910697e+07	7.150000e+05	5.300000e+11	9.0	4.000000	3.000000	
max	1.244354e+08	7.880000e+06	5.300000e+11	9.0	9.000000	9.500000	

8 rows × 22 columns



▶ In []: