Data Dictionary

Merge 2011 and 2013 dataset based on VALUE. **Note due to memory error, use 2011 dataset alone to predict value**

Using data on 'Single Family Housing'. TYPE = 1 and STRUCTURETYPE = 1

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

import xgboost as xgb
from xgboost import XGBRegressor
from xgboost import plot_importance

%matplotlib inline
sns.set_style('dark')
sns.set(font_scale=1.2)

from sklearn.model_selection import cross_val_score, train_test_split, GridSearchC
V, RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, OneHo
tEncoder
from sklearn.metrics import confusion_matrix, classification_report, mean_absolute_
error, mean_squared_error,r2_score
from sklearn.metrics import plot_confusion_matrix, plot_precision_recall_curve, plo
t_roc_curve, accuracy_score
from sklearn.metrics import auc, f1_score, precision_score, recall_score, roc_auc_s
core

import warnings
warnings.filterwarnings('ignore')

import pickle
from pickle import dump, load

np.random.seed(0)

pd.set_option('display.max_columns',100)
#pd.set_option('display.max_rows',100)
pd.set_option('display.width', 1000)
```

Data Exploration and Analysis

In [2]:
```
df1 = pd.read_csv("2011.csv")
```

In [3]:
```
df1
```

Out[3]:

	CONTROL	AGE1	METRO3	REGION	LMED	FMR	IPOV	PER	ZINC2	ZADEQ	ZSMHC	STAT
0	'036000001146'	34	2	4	84200	2580	17849	3	159972	1	4240	
1	'036000001147'	43	2	4	84200	2241	22629	4	156772	1	3502	
2	'036000001149'	60	2	4	84200	2577	17399	3	1488496	1	5014	
3	'036000001150'	37	2	4	84200	2241	14985	2	124944	1	4609	
4	'036000001151'	33	2	4	84200	2580	22557	4	149972	1	4891	
...	...	...	...	...	...	...	...	...	...	...	...	
145526	'999900022229'	30	1	3	69100	891	17960	3	800	1	543	
145527	'999900022230'	80	1	4	74900	1406	17504	3	67000	1	301	
145528	'999900022231'	56	3	4	67985	1615	23524	4	66982	1	1622	
145529	'999900022232'	23	4	2	61959	663	11642	1	27000	1	650	
145530	'999900022233'	32	1	4	75202	1052	29721	6	33972	1	1484	

145531 rows × 27 columns

In [4]:
```
df1a = df1[(df1["TYPE"] == 1) & (df1["STRUCTURETYPE"] == 1)]
```

In [5]:
```
df1a
```

Out[5]:

	CONTROL	AGE1	METRO3	REGION	LMED	FMR	IPOV	PER	ZINC2	ZADEQ	ZSMHC	STAT
0	'036000001146'	34	2	4	84200	2580	17849	3	159972	1	4240	
1	'036000001147'	43	2	4	84200	2241	22629	4	156772	1	3502	
2	'036000001149'	60	2	4	84200	2577	17399	3	1488496	1	5014	
3	'036000001150'	37	2	4	84200	2241	14985	2	124944	1	4609	
4	'036000001151'	33	2	4	84200	2580	22557	4	149972	1	4891	
...	...	...	...	...	...	...	...	...	...	...	...	
145525	'999900022228'	48	1	1	64200	1403	27127	5	35164	1	1161	
145526	'999900022229'	30	1	3	69100	891	17960	3	800	1	543	
145527	'999900022230'	80	1	4	74900	1406	17504	3	67000	1	301	
145528	'999900022231'	56	3	4	67985	1615	23524	4	66982	1	1622	
145530	'999900022233'	32	1	4	75202	1052	29721	6	33972	1	1484	

97199 rows × 27 columns

In [6]:
```
df1a.describe()
```

Out[6]:

	AGE1	REGION	LMED	FMR	IPOV	PER	ZINC2	
count	97199.000000	97199.000000	97199.000000	97199.000000	97199.000000	97199.000000	9.719900e+04	9
mean	49.466209	2.801438	69564.176741	1263.172728	16623.397761	2.303378	7.657368e+04	
std	20.311459	1.000996	11363.282757	466.143109	6876.018447	2.351663	8.668939e+04	
min	-9.000000	1.000000	33700.000000	419.000000	-9.000000	-6.000000	-3.440000e+02	
25%	38.000000	2.000000	62800.000000	975.000000	13364.000000	2.000000	2.564850e+04	
50%	50.000000	3.000000	67985.000000	1134.000000	14942.000000	2.000000	5.598200e+04	
75%	63.000000	4.000000	72403.000000	1430.000000	22488.000000	4.000000	9.998750e+04	
max	93.000000	4.000000	126600.000000	3586.000000	49801.000000	17.000000	2.977104e+06	

In [7]:
```
df2 = pd.read_csv("2013.csv")
```

In [8]:
```
df2
```

Out[8]:

	CONTROL	AGE1	METRO3	REGION	LMED	FMR	IPOV	BEDRMS	BUILT	STATUS	TYPE	VALU
0	'100003130103'	82	3	1	73738	956	11067	2	2006	1	1	400(
1	'100006110249'	50	5	3	55846	1100	24218	4	1980	1	1	1300(
2	'100006370140'	53	5	3	55846	1100	15470	4	1985	1	1	1500(
3	'100006520140'	67	5	3	55846	949	13964	3	1985	1	1	2000(
4	'100007130148'	26	1	3	60991	737	15492	2	1980	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	
64530	'999900056779'	55	1	4	55929	556	12019	1	1930	1	1	
64531	'999900056781'	37	1	2	73600	966	28229	2	1950	1	1	
64532	'999900056784'	23	2	4	86300	2701	15517	3	1940	1	1	
64533	'999900056785'	57	1	4	79659	770	12055	1	1930	1	1	
64534	'999900056786'	66	4	3	50723	542	11114	1	2012	1	1	

64535 rows × 27 columns

In [9]:
```
df2a = df2[(df2["TYPE"] == 1) & (df2["STRUCTURETYPE"] == 1)]
```

In [10]:
```
df2a
```

Out[10]:

	CONTROL	AGE1	METRO3	REGION	LMED	FMR	IPOV	BEDRMS	BUILT	STATUS	TYPE	VALU
0	'100003130103'	82	3	1	73738	956	11067	2	2006	1	1	4000
1	'100006110249'	50	5	3	55846	1100	24218	4	1980	1	1	13000
2	'100006370140'	53	5	3	55846	1100	15470	4	1985	1	1	15000
3	'100006520140'	67	5	3	55846	949	13964	3	1985	1	1	20000
6	'100007540148'	50	1	3	60991	988	18050	3	1985	1	1	26000
...	...	...	...	...	...	...	...	...	...	...	...	
64530	'999900056779'	55	1	4	55929	556	12019	1	1930	1	1	
64531	'999900056781'	37	1	2	73600	966	28229	2	1950	1	1	
64532	'999900056784'	23	2	4	86300	2701	15517	3	1940	1	1	
64533	'999900056785'	57	1	4	79659	770	12055	1	1930	1	1	
64534	'999900056786'	66	4	3	50723	542	11114	1	2012	1	1	

41216 rows × 27 columns

In [11]:
```
df2a.describe()
```

Out[11]:

	AGE1	METRO3	REGION	LMED	FMR	IPOV	BEDRMS
count	41216.000000	41216.000000	41216.000000	41216.000000	41216.000000	41216.000000	41216.000000
mean	50.676800	2.410714	2.420322	67897.371579	1256.214334	17182.425636	3.156371
std	20.464115	1.281523	1.025246	12437.306212	396.242730	7106.034151	0.886295
min	-9.000000	1.000000	1.000000	38500.000000	421.000000	-6.000000	0.000000
25%	39.000000	2.000000	2.000000	60060.000000	984.000000	13936.000000	3.000000
50%	52.000000	2.000000	2.000000	64810.000000	1185.000000	15492.000000	3.000000
75%	64.000000	3.000000	3.000000	74008.000000	1436.000000	23401.000000	4.000000
max	93.000000	5.000000	4.000000	115300.000000	3511.000000	51635.000000	7.000000

In [12]:
```
#df3 = pd.merge(left=df1a, right=df2a, how='inner',on='VALUE',suffixes=('_2011', '_
2013'))
```

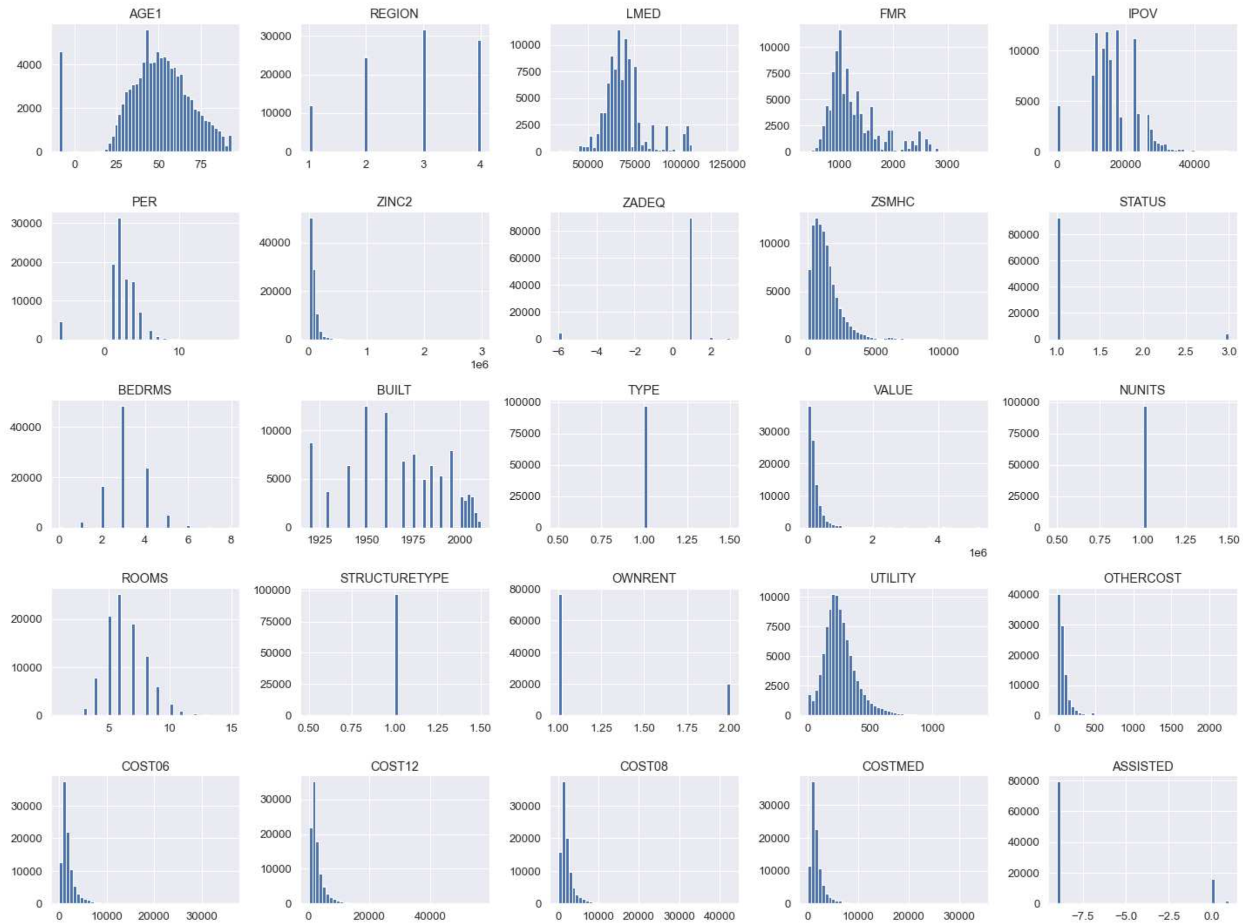
In [13]:
```
#df3
```

In [14]:
```
#df3.to_csv()
```

Data Visualization

Univariate Data Exploration

In [15]:
```
df1a.hist(bins=50, figsize=(20,15))
plt.tight_layout()
plt.show()
```



In [16]:
```
df1a.describe(include='all')
```

Out[16]:

	CONTROL	AGE1	METRO3	REGION	LMED	FMR	IPOV	
count	97199	97199.000000	97199.0	97199.000000	97199.000000	97199.000000	97199.000000	971
unique	97199	NaN	11.0	NaN	NaN	NaN	NaN	
top	'376000014086'	NaN	2.0	NaN	NaN	NaN	NaN	
freq	1	NaN	42592.0	NaN	NaN	NaN	NaN	
mean	NaN	49.466209	NaN	2.801438	69564.176741	1263.172728	16623.397761	
std	NaN	20.311459	NaN	1.000996	11363.282757	466.143109	6876.018447	
min	NaN	-9.000000	NaN	1.000000	33700.000000	419.000000	-9.000000	
25%	NaN	38.000000	NaN	2.000000	62800.000000	975.000000	13364.000000	
50%	NaN	50.000000	NaN	3.000000	67985.000000	1134.000000	14942.000000	
75%	NaN	63.000000	NaN	4.000000	72403.000000	1430.000000	22488.000000	
max	NaN	93.000000	NaN	4.000000	126600.000000	3586.000000	49801.000000	

In [17]:
```
df1a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 97199 entries, 0 to 145530
Data columns (total 27 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   CONTROL        97199 non-null  object 
 1   AGE1           97199 non-null  int64  
 2   METRO3         97199 non-null  object 
 3   REGION         97199 non-null  int64  
 4   LMED           97199 non-null  int64  
 5   FMR            97199 non-null  int64  
 6   IPOV           97199 non-null  int64  
 7   PER            97199 non-null  int64  
 8   ZINC2          97199 non-null  int64  
 9   ZADEQ          97199 non-null  int64  
 10  ZSMHC          97199 non-null  int64  
 11  STATUS         97199 non-null  int64  
 12  BEDRMS         97199 non-null  int64  
 13  BUILT          97199 non-null  int64  
 14  TYPE           97199 non-null  int64  
 15  VALUE          97199 non-null  int64  
 16  NUNITS         97199 non-null  int64  
 17  ROOMS          97199 non-null  int64  
 18  STRUCTURETYPE  97199 non-null  int64  
 19  OWNRENT        97199 non-null  int64  
 20  UTILITY        97199 non-null  float64
 21  OTHERCOST      97199 non-null  float64
 22  COST06         97199 non-null  float64
 23  COST12         97199 non-null  float64
 24  COST08         97199 non-null  float64
 25  COSTMED        97199 non-null  float64
 26  ASSISTED       97199 non-null  int64  
dtypes: float64(6), int64(19), object(2)
memory usage: 20.8+ MB
```

In [18]:
```
df1a
```

Out[18]:

	CONTROL	AGE1	METRO3	REGION	LMED	FMR	IPOV	PER	ZINC2	ZADEQ	ZSMHC	STATI
0	'036000001146'	34	2	4	84200	2580	17849	3	159972	1	4240	
1	'036000001147'	43	2	4	84200	2241	22629	4	156772	1	3502	
2	'036000001149'	60	2	4	84200	2577	17399	3	1488496	1	5014	
3	'036000001150'	37	2	4	84200	2241	14985	2	124944	1	4609	
4	'036000001151'	33	2	4	84200	2580	22557	4	149972	1	4891	
...	...	...	...	...	...	...	...	...	...	...	...	
145525	'999900022228'	48	1	1	64200	1403	27127	5	35164	1	1161	
145526	'999900022229'	30	1	3	69100	891	17960	3	800	1	543	
145527	'999900022230'	80	1	4	74900	1406	17504	3	67000	1	301	
145528	'999900022231'	56	3	4	67985	1615	23524	4	66982	1	1622	
145530	'999900022233'	32	1	4	75202	1052	29721	6	33972	1	1484	

97199 rows × 27 columns

In [19]:
```
df1a.reset_index(drop=True, inplace=True)
```

In [20]:
```
df1a
```

Out[20]:

	CONTROL	AGE1	METRO3	REGION	LMED	FMR	IPOV	PER	ZINC2	ZADEQ	ZSMHC	STATU
0	'036000001146'	34	2	4	84200	2580	17849	3	159972	1	4240	
1	'036000001147'	43	2	4	84200	2241	22629	4	156772	1	3502	
2	'036000001149'	60	2	4	84200	2577	17399	3	1488496	1	5014	
3	'036000001150'	37	2	4	84200	2241	14985	2	124944	1	4609	
4	'036000001151'	33	2	4	84200	2580	22557	4	149972	1	4891	
...	...	...	...	...	...	...	...	...	...	...	...	
97194	'999900022228'	48	1	1	64200	1403	27127	5	35164	1	1161	
97195	'999900022229'	30	1	3	69100	891	17960	3	800	1	543	
97196	'999900022230'	80	1	4	74900	1406	17504	3	67000	1	301	
97197	'999900022231'	56	3	4	67985	1615	23524	4	66982	1	1622	
97198	'999900022233'	32	1	4	75202	1052	29721	6	33972	1	1484	

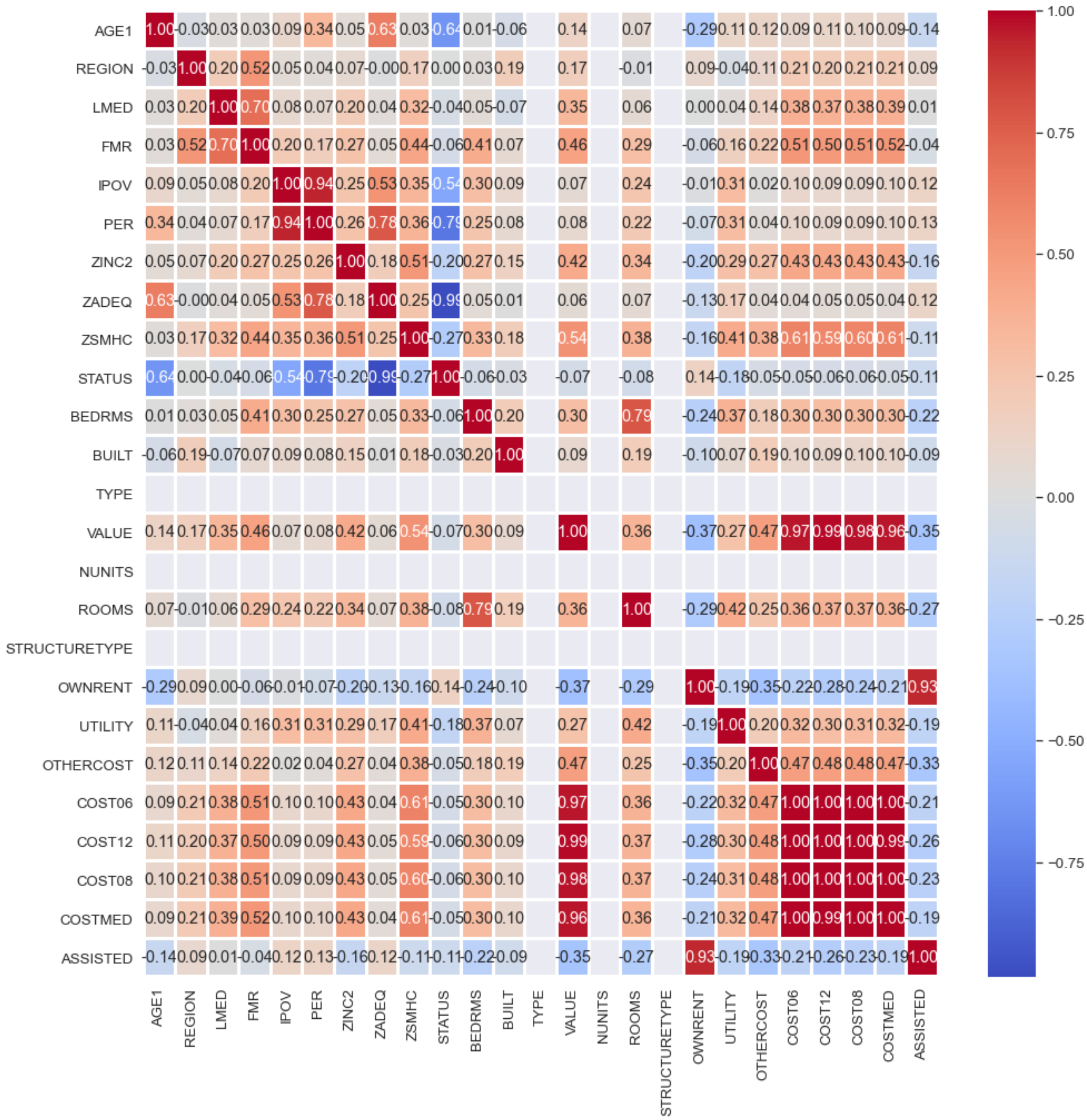
97199 rows × 27 columns

In [21]:
```
df1a.corr()
```

Out[21]:

	AGE1	REGION	LMED	FMR	IPOV	PER	ZINC2	ZADEQ	ZSM
AGE1	1.000000	-0.030955	0.027561	0.030896	0.094115	0.336538	0.046371	0.629915	0.027
REGION	-0.030955	1.000000	0.196705	0.520480	0.052567	0.037338	0.071225	-0.001720	0.167
LMED	0.027561	0.196705	1.000000	0.695302	0.078362	0.072938	0.197487	0.036113	0.317
FMR	0.030896	0.520480	0.695302	1.000000	0.201313	0.171381	0.267816	0.049625	0.442
IPOV	0.094115	0.052567	0.078362	0.201313	1.000000	0.939215	0.254810	0.531811	0.348
PER	0.336538	0.037338	0.072938	0.171381	0.939215	1.000000	0.264924	0.775399	0.357
ZINC2	0.046371	0.071225	0.197487	0.267816	0.254810	0.264924	1.000000	0.183488	0.506
ZADEQ	0.629915	-0.001720	0.036113	0.049625	0.531811	0.775399	0.183488	1.000000	0.253
ZSMHC	0.027638	0.167809	0.317332	0.442831	0.348631	0.357018	0.506021	0.253351	1.000
STATUS	-0.641274	0.000706	-0.041163	-0.056931	-0.538887	-0.786610	-0.196801	-0.985366	-0.266
BEDRMS	0.012849	0.034816	0.052675	0.407749	0.301361	0.250235	0.273090	0.052235	0.331
BUILT	-0.062295	0.185210	-0.069982	0.071314	0.089366	0.076472	0.147503	0.010289	0.184
TYPE	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
VALUE	0.137732	0.171838	0.345398	0.464327	0.065034	0.079957	0.417900	0.058190	0.541
NUNITS	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
ROOMS	0.066275	-0.014606	0.062643	0.290822	0.244578	0.217438	0.344084	0.068893	0.378
STRUCTURETYPE	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
OWNRENT	-0.293108	0.085058	0.000734	-0.059917	-0.012364	-0.071441	-0.195138	-0.129494	-0.164
UTILITY	0.110868	-0.039971	0.035877	0.161919	0.314848	0.306349	0.286467	0.174207	0.411
OTHERCOST	0.118070	0.106464	0.136610	0.223425	0.015561	0.036239	0.271081	0.043782	0.381
COST06	0.090690	0.210810	0.383990	0.514116	0.096505	0.096966	0.432566	0.044549	0.606
COST12	0.108741	0.198075	0.372739	0.499745	0.085648	0.091446	0.430515	0.049925	0.587
COST08	0.098305	0.205733	0.379718	0.508685	0.092088	0.094772	0.432179	0.046836	0.598
COSTMED	0.087281	0.212949	0.385690	0.516264	0.098409	0.097889	0.432526	0.043517	0.608
ASSISTED	-0.141489	0.087393	0.011446	-0.042270	0.124868	0.126562	-0.155379	0.116745	-0.105

In [22]:
```python
plt.figure(figsize=(16,16))
sns.heatmap(df1a.corr(),cmap="coolwarm",annot=True,fmt='.2f',linewidths=2)
plt.show()
```



In [ ]:

Data Preprocessing

Treat Missing Values

In [23]:
```
df1a.isnull().sum()
```
Out[23]:
```
CONTROL          0
AGE1             0
METRO3           0
REGION           0
LMED             0
FMR              0
IPOV             0
PER              0
ZINC2            0
ZADEQ            0
ZSMHC            0
STATUS           0
BEDRMS           0
BUILT            0
TYPE             0
VALUE            0
NUNITS           0
ROOMS            0
STRUCTURETYPE    0
OWNRENT          0
UTILITY          0
OTHERCOST        0
COST06           0
COST12           0
COST08           0
COSTMED          0
ASSISTED         0
dtype: int64
```

Treat Duplicate Values

In [24]:
```
df1a.duplicated(keep='first').sum()
```
Out[24]:
```
0
```

Treat Outliers

In [25]:
```
df1a.describe()
```
Out[25]:

	AGE1	REGION	LMED	FMR	IPOV	PER	ZINC2
count	97199.000000	97199.000000	97199.000000	97199.000000	97199.000000	97199.000000	9.719900e+04
mean	49.466209	2.801438	69564.176741	1263.172728	16623.397761	2.303378	7.657368e+04
std	20.311459	1.000996	11363.282757	466.143109	6876.018447	2.351663	8.668939e+04
min	-9.000000	1.000000	33700.000000	419.000000	-9.000000	-6.000000	-3.440000e+02
25%	38.000000	2.000000	62800.000000	975.000000	13364.000000	2.000000	2.564850e+04
50%	50.000000	3.000000	67985.000000	1134.000000	14942.000000	2.000000	5.598200e+04
75%	63.000000	4.000000	72403.000000	1430.000000	22488.000000	4.000000	9.998750e+04
max	93.000000	4.000000	126600.000000	3586.000000	49801.000000	17.000000	2.977104e+06

Drop unwanted features

In [26]:
```
df1a.columns
```

Out[26]:
```
Index(['CONTROL', 'AGE1', 'METRO3', 'REGION', 'LMED', 'FMR', 'IPOV', 'PER', 'ZIN
C2', 'ZADEQ', 'ZSMHC', 'STATUS', 'BEDRMS', 'BUILT', 'TYPE', 'VALUE', 'NUNITS', '
ROOMS', 'STRUCTURETYPE', 'OWNRENT', 'UTILITY', 'OTHERCOST', 'COST06', 'COST12',
'COST08', 'COSTMED', 'ASSISTED'], dtype='object')
```

Drop all strings and some categoricals

In [27]:
```
df1a.drop(['CONTROL','METRO3', 'REGION','BUILT','STATUS','TYPE','NUNITS','STRUCTURE
TYPE',
          'ZADEQ','OWNRENT','COST06', 'COST12', 'COST08','ASSISTED'],axis=1,inplace
=True)
```

In [28]:
```
df1a
```

Out[28]:

	AGE1	LMED	FMR	IPOV	PER	ZINC2	ZSMHC	BEDRMS	VALUE	ROOMS	UTILITY	OTHERCO
0	34	84200	2580	17849	3	159972	4240	4	720000	8	300.000000	248.3333
1	43	84200	2241	22629	4	156772	3502	3	550000	5	256.000000	362.5000
2	60	84200	2577	17399	3	1488496	5014	5	720000	11	233.000000	180.0000
3	37	84200	2241	14985	2	124944	4609	3	450000	5	152.000000	290.0000
4	33	84200	2580	22557	4	149972	4891	4	700000	9	656.166667	181.6666
...	...	...	...	...	...	...	...	...	...	...	...	
97194	48	64200	1403	27127	5	35164	1161	2	-6	5	61.000000	0.0000
97195	30	69100	891	17960	3	800	543	2	-6	5	142.500000	0.0000
97196	80	74900	1406	17504	3	67000	301	2	-6	4	62.000000	5.5833
97197	56	67985	1615	23524	4	66982	1622	3	350000	6	227.583333	87.5000
97198	32	75202	1052	29721	6	33972	1484	2	-6	4	134.000000	0.0000

97199 rows × 13 columns

In [29]:
```
#Remove negative values
```

In [30]:
```
df3 = df1a[df1a["VALUE"] >= 0 ]
```

In [31]:
```
df3
```
Out[31]:

	AGE1	LMED	FMR	IPOV	PER	ZINC2	ZSMHC	BEDRMS	VALUE	ROOMS	UTILITY	OTHERCO:
0	34	84200	2580	17849	3	159972	4240	4	720000	8	300.000000	248.3333
1	43	84200	2241	22629	4	156772	3502	3	550000	5	256.000000	362.5000
2	60	84200	2577	17399	3	1488496	5014	5	720000	11	233.000000	180.0000
3	37	84200	2241	14985	2	124944	4609	3	450000	5	152.000000	290.0000
4	33	84200	2580	22557	4	149972	4891	4	700000	9	656.166667	181.6666
...	...	...	...	...	...	...	...	...	...	...	...	
97174	40	57215	619	18012	3	77982	392	1	250000	2	0.000000	100.0000
97180	49	62806	659	14895	2	125564	11	1	44000	3	0.000000	11.0000
97189	61	67715	648	11536	1	2200	245	1	105000	3	107.666667	26.2500
97192	70	53300	662	13487	2	77200	285	1	30000	4	93.000000	108.3333
97197	56	67985	1615	23524	4	66982	1622	3	350000	6	227.583333	87.5000

77007 rows × 13 columns

In [32]:
```
df3["VALUE"].value_counts()
```
Out[32]:
```
200000    3460
150000    3175
250000    2590
100000    2502
300000    2487
          ...
127400       1
203148       1
51471        1
495          1
426000       1
Name: VALUE, Length: 1699, dtype: int64
```

In [33]:
```
df3.columns
```
Out[33]:
```
Index(['AGE1', 'LMED', 'FMR', 'IPOV', 'PER', 'ZINC2', 'ZSMHC', 'BEDRMS', 'VALUE
', 'ROOMS', 'UTILITY', 'OTHERCOST', 'COSTMED'], dtype='object')
```

In [34]:
```
df3 = df3[['AGE1', 'LMED', 'FMR', 'IPOV', 'PER', 'ZINC2', 'ZSMHC', 'BEDRMS','ROOMS
', 'UTILITY', 'OTHERCOST', 'COSTMED', 'VALUE']]
```

In [35]:
```
df3
```

Out[35]:

	AGE1	LMED	FMR	IPOV	PER	ZINC2	ZSMHC	BEDRMS	ROOMS	UTILITY	OTHERCOST	CC
0	34	84200	2580	17849	3	159972	4240	4	8	300.000000	248.333333	5026
1	43	84200	2241	22629	4	156772	3502	3	5	256.000000	362.500000	4039
2	60	84200	2577	17399	3	1488496	5014	5	11	233.000000	180.000000	4891
3	37	84200	2241	14985	2	124944	4609	3	5	152.000000	290.000000	3240
4	33	84200	2580	22557	4	149972	4891	4	9	656.166667	181.666667	5191
...	...	...	...	...	...	...	...	...	...	...	...	
97174	40	57215	619	18012	3	77982	392	1	2	0.000000	100.000000	1654
97180	49	62806	659	14895	2	125564	11	1	3	0.000000	11.000000	284
97189	61	67715	648	11536	1	2200	245	1	3	107.666667	26.250000	786
97192	70	53300	662	13487	2	77200	285	1	4	93.000000	108.333333	387
97197	56	67985	1615	23524	4	66982	1622	3	6	227.583333	87.500000	2492

77007 rows × 13 columns

In [36]:
```
df3.reset_index(inplace=True, drop=True)
```

In [37]:
```
df3
```

Out[37]:

	AGE1	LMED	FMR	IPOV	PER	ZINC2	ZSMHC	BEDRMS	ROOMS	UTILITY	OTHERCOST	CC
0	34	84200	2580	17849	3	159972	4240	4	8	300.000000	248.333333	5026
1	43	84200	2241	22629	4	156772	3502	3	5	256.000000	362.500000	4039
2	60	84200	2577	17399	3	1488496	5014	5	11	233.000000	180.000000	4891
3	37	84200	2241	14985	2	124944	4609	3	5	152.000000	290.000000	3240
4	33	84200	2580	22557	4	149972	4891	4	9	656.166667	181.666667	5191
...	...	...	...	...	...	...	...	...	...	...	...	
77002	40	57215	619	18012	3	77982	392	1	2	0.000000	100.000000	1654
77003	49	62806	659	14895	2	125564	11	1	3	0.000000	11.000000	284
77004	61	67715	648	11536	1	2200	245	1	3	107.666667	26.250000	786
77005	70	53300	662	13487	2	77200	285	1	4	93.000000	108.333333	387
77006	56	67985	1615	23524	4	66982	1622	3	6	227.583333	87.500000	2492

77007 rows × 13 columns

Create and save processed dataset

In [38]:
```
#df3.to_csv("house.csv",index=False)
```

Load Data

In [39]:
```
df = pd.read_csv("house.csv")
```

In [40]:
```
df
```

Out[40]:

	AGE1	LMED	FMR	IPOV	PER	ZINC2	ZSMHC	BEDRMS	ROOMS	UTILITY	OTHERCOST	CC
0	34	84200	2580	17849	3	159972	4240	4	8	300.000000	248.333333	5026
1	43	84200	2241	22629	4	156772	3502	3	5	256.000000	362.500000	4039
2	60	84200	2577	17399	3	1488496	5014	5	11	233.000000	180.000000	4891
3	37	84200	2241	14985	2	124944	4609	3	5	152.000000	290.000000	3240
4	33	84200	2580	22557	4	149972	4891	4	9	656.166667	181.666667	5191
...	...	...	...	...	...	...	...	...	...	...	...	
77002	40	57215	619	18012	3	77982	392	1	2	0.000000	100.000000	1654
77003	49	62806	659	14895	2	125564	11	1	3	0.000000	11.000000	284
77004	61	67715	648	11536	1	2200	245	1	3	107.666667	26.250000	786
77005	70	53300	662	13487	2	77200	285	1	4	93.000000	108.333333	387
77006	56	67985	1615	23524	4	66982	1622	3	6	227.583333	87.500000	2492

77007 rows × 13 columns

In [41]:
```
df.shape
```

Out[41]: (77007, 13)

In [42]:
```
df1 = df.sample(frac=0.13, random_state=0)
```

In [43]:
```
df1.shape
```

Out[43]: (10011, 13)

In [44]:
```
df1
```

Out[44]:

	AGE1	LMED	FMR	IPOV	PER	ZINC2	ZSMHC	BEDRMS	ROOMS	UTILITY	OTHERCOST	COS
58408	53	75159	1518	15026	2	78010	2197	3	6	180.666667	62.500000	802.9
71564	54	74900	1999	17454	3	134000	532	3	6	250.333333	123.333333	2550.6
36241	29	58300	1010	17849	3	110657	1432	3	6	211.000000	74.000000	1149.5
59739	61	75202	1520	14895	2	229972	2116	3	6	324.000000	100.000000	4155.8
32138	67	64028	963	13403	2	59286	1027	3	7	194.750000	41.000000	1255.8
...	...	...	...	...	...	...	...	...	...	...	...	
11817	67	67109	779	13445	2	20400	658	2	5	253.666667	62.500000	502.7
2510	-9	60802	1010	-9	-6	-6	-6	4	8	130.916667	100.000000	666.3
23898	66	61864	882	13364	2	189964	1342	3	10	637.666667	91.666667	4834.4
14750	30	69100	1160	22694	4	88000	816	3	6	438.916667	36.250000	1377.0
23715	67	51913	848	13403	2	74504	419	3	7	329.000000	43.750000	1461.2

10011 rows × 13 columns

Train Test Split

In [45]:
```python
X = df1.iloc[:,0:12]
y = df1.iloc[:,12]
```

In [46]:
```python
X
```

Out[46]:

	AGE1	LMED	FMR	IPOV	PER	ZINC2	ZSMHC	BEDRMS	ROOMS	UTILITY	OTHERCOST	COS
58408	53	75159	1518	15026	2	78010	2197	3	6	180.666667	62.500000	802.
71564	54	74900	1999	17454	3	134000	532	3	6	250.333333	123.333333	2550.
36241	29	58300	1010	17849	3	110657	1432	3	6	211.000000	74.000000	1149.
59739	61	75202	1520	14895	2	229972	2116	3	6	324.000000	100.000000	4155.
32138	67	64028	963	13403	2	59286	1027	3	7	194.750000	41.000000	1255.
...	...	...	...	...	...	...	...	...	...	...	...	
11817	67	67109	779	13445	2	20400	658	2	5	253.666667	62.500000	502.
2510	-9	60802	1010	-9	-6	-6	-6	4	8	130.916667	100.000000	666.
23898	66	61864	882	13364	2	189964	1342	3	10	637.666667	91.666667	4834.
14750	30	69100	1160	22694	4	88000	816	3	6	438.916667	36.250000	1377.
23715	67	51913	848	13403	2	74504	419	3	7	329.000000	43.750000	1461.

10011 rows × 12 columns

In [47]:
```python
X.values, y.values
```

Out[47]:
```
(array([[5.30000000e+01, 7.51590000e+04, 1.51800000e+03, ...,
         1.80666667e+02, 6.25000000e+01, 8.02951665e+02],
        [5.40000000e+01, 7.49000000e+04, 1.99900000e+03, ...,
         2.50333333e+02, 1.23333333e+02, 2.55060833e+03],
        [2.90000000e+01, 5.83000000e+04, 1.01000000e+03, ...,
         2.11000000e+02, 7.40000000e+01, 1.14955683e+03],
        ...,
        [6.60000000e+01, 6.18640000e+04, 8.82000000e+02, ...,
         6.37666667e+02, 9.16666667e+01, 4.83442332e+03],
        [3.00000000e+01, 6.91000000e+04, 1.16000000e+03, ...,
         4.38916667e+02, 3.62500000e+01, 1.37704250e+03],
        [6.70000000e+01, 5.19130000e+04, 8.48000000e+02, ...,
         3.29000000e+02, 4.37500000e+01, 1.46122083e+03]]),
 array([ 90000, 350000, 139000, ..., 660000, 145000, 175000], dtype=int64))
```

In [48]:
```python
X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=
0.2, random_state=0)
```

In [49]:
```python
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[49]:
```
((8008, 12), (2003, 12), (8008,), (2003,))
```

Feature Scaling

In [50]:
```
X_train
```

Out[50]:
```
array([[5.00000000e+01, 6.91320000e+04, 1.01700000e+03, ...,
         2.57000000e+02, 1.16666667e+02, 1.43103833e+03],
        [5.90000000e+01, 7.52020000e+04, 1.52000000e+03, ...,
         1.97666667e+02, 0.00000000e+00, 7.57451665e+02],
        [3.70000000e+01, 6.99000000e+04, 1.31900000e+03, ...,
         2.57000000e+02, 7.37500000e+01, 2.16560083e+03],
        ...,
        [4.80000000e+01, 6.91000000e+04, 1.16000000e+03, ...,
         4.50000000e+02, 8.33333333e+01, 1.44764883e+03],
        [3.90000000e+01, 6.55000000e+04, 1.59600000e+03, ...,
         4.02000000e+02, 5.00000000e+01, 2.31795000e+03],
        [3.70000000e+01, 6.80420000e+04, 1.06800000e+03, ...,
         3.67333333e+02, 6.50000000e+01, 1.42750666e+03]])
```

In [51]:
```
scaler = StandardScaler()
```

In [52]:
```
X_train_scaled = scaler.fit_transform(X_train)
```

In [53]:
```
X_test_scaled = scaler.transform(X_test)
```

In [54]:
```
X_train_scaled
```

Out[54]:
```
array([[-0.12068029, -0.03780161, -0.55955832, ..., -0.13124982,
          0.22176716, -0.30580565],
        [ 0.34964198,  0.50132567,  0.51414939, ..., -0.57334221,
         -0.95056768, -0.63667274],
        [-0.80003467,  0.03041087,  0.08509323, ..., -0.13124982,
         -0.20948458,  0.05501289],
        ...,
        [-0.22519635, -0.04064379, -0.25430941, ...,  1.30679228,
         -0.11318565, -0.29764654],
        [-0.69551861, -0.36038979,  0.67637958, ...,  0.9491445 ,
         -0.44813846,  0.1298471 ],
        [-0.80003467, -0.13461359, -0.45069332, ...,  0.69084333,
         -0.2974097 , -0.30754041]])
```

In [55]:
```
X_test_scaled
```

Out[55]:
```
array([[ 0.61093213,  0.66102103, -0.43148185, ...,  0.82620308,
         -0.53187666, -0.34063023],
        [ 0.0360938 , -0.04064379, -0.25430941, ...,  0.09228004,
          0.47298177, -0.34864045],
        [-0.06842226, -1.08790075, -0.51046234, ...,  0.19659397,
         -0.12993329, -0.27083288],
        ...,
        [ 0.19286789,  1.30051303,  2.05320158, ...,  1.38316498,
          1.38572818,  1.01190256],
        [-0.43422847, -1.56716447, -1.33015172, ..., -1.49478197,
         -0.89530046, -0.90246965],
        [-0.27745438,  2.01994152,  1.30181965, ..., -0.58576054,
          0.93354188,  1.47103721]])
```

In [ ]:

Model Training

Using XGBoost (Scikit-Learn)

Using RandomSearchCV

In [56]:
```
model = XGBRegressor(random_state=0, n_estimators=100, objective='reg:squarederror
')
```

In [57]:
```
parameters = {'max_depth': np.arange(3,10,1),
              'learning rate': np.arange(0.05,0.3,0.03),
              'n_estimators':np.arange(100,1000,100),
              'min_child_weight': np.arange(1,4,1),
              'gamma':np.arange(0,50,2),
              'subsample':np.arange(0.5,0.9,0.1),
              'colsample_bytree':np.arange(0.5,0.9,0.1)
             }
```

In [58]:
```
randm = RandomizedSearchCV(estimator=model, param_distributions = parameters, cv =
5, n_iter = 20,
                           n_jobs=-1, scoring='neg_mean_squared_error')
```

In [59]:
```
randm.fit(X_train_scaled, y_train)
```

Out[59]:
```
RandomizedSearchCV(cv=5, estimator=XGBRegressor(objective='reg:squarederror'),
                   n_iter=20, n_jobs=-1,
                   param_distributions={'colsample_bytree': array([0.5, 0.6, 0.
7, 0.8]),
                                        'gamma': array([ 0,  2,  4,  6,  8, 10,
12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32,
       34, 36, 38, 40, 42, 44, 46, 48]),
                                        'learning rate': array([0.05, 0.08, 0.1
1, 0.14, 0.17, 0.2 , 0.23, 0.26, 0.29]),
                                        'max_depth': array([3, 4, 5, 6, 7, 8,
9]),
                                        'min_child_weight': array([1, 2, 3]),
                                        'n_estimators': array([100, 200, 300, 40
0, 500, 600, 700, 800, 900]),
                                        'subsample': array([0.5, 0.6, 0.7, 0.
8])},
                   scoring='neg_mean_squared_error')
```

In [60]:
```
randm.best_estimator_
```

Out[60]:
```
XGBRegressor(colsample_bytree=0.7999999999999999, gamma=8, learning rate=0.2,
             min_child_weight=3, n_estimators=300, objective='reg:squarederror',
             subsample=0.6)
```

In [61]:
```
randm.best_score_
```

Out[61]:
```
-410930042.38920146
```

In [62]:
```
randm.best_params_
```

Out[62]:
```
{'subsample': 0.6,
 'n_estimators': 300,
 'min_child_weight': 3,
 'max_depth': 3,
 'learning rate': 0.2,
 'gamma': 8,
 'colsample_bytree': 0.7999999999999999}
```

In [63]:
```
xgbmodel = XGBRegressor(random_state=0, n_estimators=600, objective='reg:squarederror',subsample=0.6,
                        max_depth=3,learning_rate=0.2,gamma=8,colsample_bytree=0.8,min_child_weight=3)
```

In [64]:
```
xgbmodel.fit(X_train_scaled,y_train,eval_set=[(X_test_scaled,y_test)],eval_metric='
rmse',early_stopping_rounds=10)
```

```
[0]	validation_0-rmse:306347
Will train until validation_0-rmse hasn't improved in 10 rounds.
[1]	validation_0-rmse:246520
[2]	validation_0-rmse:198875
[3]	validation_0-rmse:161143
[4]	validation_0-rmse:131475
[5]	validation_0-rmse:106631
[6]	validation_0-rmse:93525.4
[7]	validation_0-rmse:84150.6
[8]	validation_0-rmse:70259.1
[9]	validation_0-rmse:59517.9
[10]	validation_0-rmse:51879
[11]	validation_0-rmse:45746.2
[12]	validation_0-rmse:43873.9
[13]	validation_0-rmse:40069.6
[14]	validation_0-rmse:39038.6
[15]	validation_0-rmse:36270
[16]	validation_0-rmse:34093.7
[17]	validation_0-rmse:33658.4
[18]	validation_0-rmse:31805.3
[19]	validation_0-rmse:30040.8
[20]	validation_0-rmse:29616
[21]	validation_0-rmse:29143.6
[22]	validation_0-rmse:28868.6
[23]	validation_0-rmse:27960.5
[24]	validation_0-rmse:27030.5
[25]	validation_0-rmse:26077.5
[26]	validation_0-rmse:25720
[27]	validation_0-rmse:25066.7
[28]	validation_0-rmse:24045.1
[29]	validation_0-rmse:23355.5
[30]	validation_0-rmse:22645
[31]	validation_0-rmse:22077.1
[32]	validation_0-rmse:21684
[33]	validation_0-rmse:21277.7
[34]	validation_0-rmse:20704.9
[35]	validation_0-rmse:20026.5
[36]	validation_0-rmse:19790.4
[37]	validation_0-rmse:19244.9
[38]	validation_0-rmse:18930.8
[39]	validation_0-rmse:18616.4
[40]	validation_0-rmse:18437.8
[41]	validation_0-rmse:17993.9
[42]	validation_0-rmse:17710.7
[43]	validation_0-rmse:17501.2
[44]	validation_0-rmse:16861.3
[45]	validation_0-rmse:16659.2
[46]	validation_0-rmse:16357.3
[47]	validation_0-rmse:16086.2
[48]	validation_0-rmse:15824.3
[49]	validation_0-rmse:15692.2
[50]	validation_0-rmse:15285.3
[51]	validation_0-rmse:15175.1
[52]	validation_0-rmse:14853
[53]	validation_0-rmse:14626.9
[54]	validation_0-rmse:14341.5
[55]	validation_0-rmse:14236.8
[56]	validation_0-rmse:13972
[57]	validation_0-rmse:13889.9
[58]	validation_0-rmse:13862.5
[59]	validation_0-rmse:13797.7
[60]	validation_0-rmse:13743.3
[61]	validation_0-rmse:13717.1
[62]	validation_0-rmse:13601
[63]	validation_0-rmse:13463.9
```

```
[64]	validation_0-rmse:13183.9
[65]	validation_0-rmse:13024.9
[66]	validation_0-rmse:13033.6
[67]	validation_0-rmse:12984.9
[68]	validation_0-rmse:12831.7
[69]	validation_0-rmse:12720.5
[70]	validation_0-rmse:12627.1
[71]	validation_0-rmse:12580.7
[72]	validation_0-rmse:12508.1
[73]	validation_0-rmse:12474.8
[74]	validation_0-rmse:12384.6
[75]	validation_0-rmse:12317.5
[76]	validation_0-rmse:12207.4
[77]	validation_0-rmse:12190.8
[78]	validation_0-rmse:12159.3
[79]	validation_0-rmse:12169.5
[80]	validation_0-rmse:12146.6
[81]	validation_0-rmse:12118.1
[82]	validation_0-rmse:12060.9
[83]	validation_0-rmse:11993.2
[84]	validation_0-rmse:11851.2
[85]	validation_0-rmse:11702.3
[86]	validation_0-rmse:11631.7
[87]	validation_0-rmse:11647.4
[88]	validation_0-rmse:11594.1
[89]	validation_0-rmse:11526.7
[90]	validation_0-rmse:11504.9
[91]	validation_0-rmse:11481.7
[92]	validation_0-rmse:11348.5
[93]	validation_0-rmse:11277.6
[94]	validation_0-rmse:11131.6
[95]	validation_0-rmse:11067
[96]	validation_0-rmse:11057.2
[97]	validation_0-rmse:11011.3
[98]	validation_0-rmse:10992.4
[99]	validation_0-rmse:10914.1
[100]	validation_0-rmse:10871.5
[101]	validation_0-rmse:10792.5
[102]	validation_0-rmse:10693.6
[103]	validation_0-rmse:10658.4
[104]	validation_0-rmse:10657.8
[105]	validation_0-rmse:10552.1
[106]	validation_0-rmse:10544.3
[107]	validation_0-rmse:10511.4
[108]	validation_0-rmse:10535
[109]	validation_0-rmse:10567.7
[110]	validation_0-rmse:10573.3
[111]	validation_0-rmse:10520.6
[112]	validation_0-rmse:10462
[113]	validation_0-rmse:10388.2
[114]	validation_0-rmse:10366.2
[115]	validation_0-rmse:10351.4
[116]	validation_0-rmse:10353.8
[117]	validation_0-rmse:10311.3
[118]	validation_0-rmse:10321.5
[119]	validation_0-rmse:10313.4
[120]	validation_0-rmse:10217.4
[121]	validation_0-rmse:10195.5
[122]	validation_0-rmse:10113.6
[123]	validation_0-rmse:10049.3
[124]	validation_0-rmse:9921.02
[125]	validation_0-rmse:9861.99
[126]	validation_0-rmse:9859.48
[127]	validation_0-rmse:9845.17
[128]	validation_0-rmse:9837.61
```

```
[129]	validation_0-rmse:9764.51
[130]	validation_0-rmse:9740.27
[131]	validation_0-rmse:9726.12
[132]	validation_0-rmse:9697.77
[133]	validation_0-rmse:9629.73
[134]	validation_0-rmse:9615.9
[135]	validation_0-rmse:9612.57
[136]	validation_0-rmse:9607.93
[137]	validation_0-rmse:9590.58
[138]	validation_0-rmse:9594.4
[139]	validation_0-rmse:9594.21
[140]	validation_0-rmse:9563.47
[141]	validation_0-rmse:9546.27
[142]	validation_0-rmse:9441.99
[143]	validation_0-rmse:9475.52
[144]	validation_0-rmse:9451.41
[145]	validation_0-rmse:9410.16
[146]	validation_0-rmse:9400.48
[147]	validation_0-rmse:9388.4
[148]	validation_0-rmse:9371.49
[149]	validation_0-rmse:9321.24
[150]	validation_0-rmse:9287.12
[151]	validation_0-rmse:9283.12
[152]	validation_0-rmse:9316.25
[153]	validation_0-rmse:9326.12
[154]	validation_0-rmse:9314.78
[155]	validation_0-rmse:9314.74
[156]	validation_0-rmse:9263.54
[157]	validation_0-rmse:9214.01
[158]	validation_0-rmse:9204.26
[159]	validation_0-rmse:9189.57
[160]	validation_0-rmse:9175.91
[161]	validation_0-rmse:9209.61
[162]	validation_0-rmse:9181.77
[163]	validation_0-rmse:9178.97
[164]	validation_0-rmse:9168.02
[165]	validation_0-rmse:9135
[166]	validation_0-rmse:9100.88
[167]	validation_0-rmse:9124.11
[168]	validation_0-rmse:9109.22
[169]	validation_0-rmse:9087.11
[170]	validation_0-rmse:9071.96
[171]	validation_0-rmse:9052.74
[172]	validation_0-rmse:9023.11
[173]	validation_0-rmse:8974.05
[174]	validation_0-rmse:9001.82
[175]	validation_0-rmse:8955.05
[176]	validation_0-rmse:8943.46
[177]	validation_0-rmse:8960.63
[178]	validation_0-rmse:8945.86
[179]	validation_0-rmse:8936.03
[180]	validation_0-rmse:8931.21
[181]	validation_0-rmse:8961.66
[182]	validation_0-rmse:8927.66
[183]	validation_0-rmse:8883.47
[184]	validation_0-rmse:8876.89
[185]	validation_0-rmse:8877.62
[186]	validation_0-rmse:8841.79
[187]	validation_0-rmse:8812.77
[188]	validation_0-rmse:8807.95
[189]	validation_0-rmse:8795.41
[190]	validation_0-rmse:8779.35
[191]	validation_0-rmse:8764.14
[192]	validation_0-rmse:8755.41
[193]	validation_0-rmse:8742.67
```

```
[194]	validation_0-rmse:8737.15
[195]	validation_0-rmse:8723.52
[196]	validation_0-rmse:8743.2
[197]	validation_0-rmse:8750.08
[198]	validation_0-rmse:8744.57
[199]	validation_0-rmse:8744.94
[200]	validation_0-rmse:8734.89
[201]	validation_0-rmse:8710.95
[202]	validation_0-rmse:8743.59
[203]	validation_0-rmse:8718.02
[204]	validation_0-rmse:8696.01
[205]	validation_0-rmse:8685.91
[206]	validation_0-rmse:8692.97
[207]	validation_0-rmse:8695.6
[208]	validation_0-rmse:8684.63
[209]	validation_0-rmse:8678.82
[210]	validation_0-rmse:8663.99
[211]	validation_0-rmse:8670.15
[212]	validation_0-rmse:8645.76
[213]	validation_0-rmse:8647.29
[214]	validation_0-rmse:8638.28
[215]	validation_0-rmse:8605.57
[216]	validation_0-rmse:8606.19
[217]	validation_0-rmse:8603.52
[218]	validation_0-rmse:8616.96
[219]	validation_0-rmse:8621.61
[220]	validation_0-rmse:8589.86
[221]	validation_0-rmse:8584.12
[222]	validation_0-rmse:8573.62
[223]	validation_0-rmse:8576.38
[224]	validation_0-rmse:8574.49
[225]	validation_0-rmse:8570.71
[226]	validation_0-rmse:8558.77
[227]	validation_0-rmse:8563.56
[228]	validation_0-rmse:8561.9
[229]	validation_0-rmse:8563.8
[230]	validation_0-rmse:8531.46
[231]	validation_0-rmse:8534.44
[232]	validation_0-rmse:8542
[233]	validation_0-rmse:8539.86
[234]	validation_0-rmse:8534.27
[235]	validation_0-rmse:8519.79
[236]	validation_0-rmse:8523.64
[237]	validation_0-rmse:8520.25
[238]	validation_0-rmse:8511.65
[239]	validation_0-rmse:8522.4
[240]	validation_0-rmse:8511.42
[241]	validation_0-rmse:8513.41
[242]	validation_0-rmse:8476.84
[243]	validation_0-rmse:8470.21
[244]	validation_0-rmse:8456.24
[245]	validation_0-rmse:8437.35
[246]	validation_0-rmse:8430.54
[247]	validation_0-rmse:8422.75
[248]	validation_0-rmse:8412.72
[249]	validation_0-rmse:8410.29
[250]	validation_0-rmse:8409.39
[251]	validation_0-rmse:8422.19
[252]	validation_0-rmse:8403.75
[253]	validation_0-rmse:8374.26
[254]	validation_0-rmse:8341.24
[255]	validation_0-rmse:8320.52
[256]	validation_0-rmse:8295.95
[257]	validation_0-rmse:8317.43
[258]	validation_0-rmse:8305.65
```

```
[259]	validation_0-rmse:8298.65
[260]	validation_0-rmse:8288.64
[261]	validation_0-rmse:8285.11
[262]	validation_0-rmse:8284.73
[263]	validation_0-rmse:8272.5
[264]	validation_0-rmse:8260
[265]	validation_0-rmse:8239.67
[266]	validation_0-rmse:8230.58
[267]	validation_0-rmse:8231.08
[268]	validation_0-rmse:8233.48
[269]	validation_0-rmse:8231.14
[270]	validation_0-rmse:8223.31
[271]	validation_0-rmse:8218.93
[272]	validation_0-rmse:8211.24
[273]	validation_0-rmse:8202.64
[274]	validation_0-rmse:8202.92
[275]	validation_0-rmse:8204.48
[276]	validation_0-rmse:8209.05
[277]	validation_0-rmse:8208.7
[278]	validation_0-rmse:8226.79
[279]	validation_0-rmse:8203.83
[280]	validation_0-rmse:8206
[281]	validation_0-rmse:8181.95
[282]	validation_0-rmse:8160.87
[283]	validation_0-rmse:8175.25
[284]	validation_0-rmse:8166.4
[285]	validation_0-rmse:8165.82
[286]	validation_0-rmse:8187.87
[287]	validation_0-rmse:8177.68
[288]	validation_0-rmse:8180.08
[289]	validation_0-rmse:8183.94
[290]	validation_0-rmse:8153.91
[291]	validation_0-rmse:8143.01
[292]	validation_0-rmse:8129.65
[293]	validation_0-rmse:8146.84
[294]	validation_0-rmse:8137.54
[295]	validation_0-rmse:8159.54
[296]	validation_0-rmse:8154.02
[297]	validation_0-rmse:8139.15
[298]	validation_0-rmse:8162.39
[299]	validation_0-rmse:8160.86
[300]	validation_0-rmse:8161.86
[301]	validation_0-rmse:8127.09
[302]	validation_0-rmse:8118.24
[303]	validation_0-rmse:8109.6
[304]	validation_0-rmse:8104.89
[305]	validation_0-rmse:8113.9
[306]	validation_0-rmse:8107.1
[307]	validation_0-rmse:8095.15
[308]	validation_0-rmse:8088.12
[309]	validation_0-rmse:8101.78
[310]	validation_0-rmse:8104.05
[311]	validation_0-rmse:8095.88
[312]	validation_0-rmse:8092.81
[313]	validation_0-rmse:8095.98
[314]	validation_0-rmse:8075.13
[315]	validation_0-rmse:8064.88
[316]	validation_0-rmse:8044.5
[317]	validation_0-rmse:8036.14
[318]	validation_0-rmse:8042.1
[319]	validation_0-rmse:8035.65
[320]	validation_0-rmse:8035.76
[321]	validation_0-rmse:8038.84
[322]	validation_0-rmse:8036.75
[323]	validation_0-rmse:8030.32
```

```
[324]	validation_0-rmse:8033.03
[325]	validation_0-rmse:8031.37
[326]	validation_0-rmse:8025.75
[327]	validation_0-rmse:8020.95
[328]	validation_0-rmse:8005.22
[329]	validation_0-rmse:7998.32
[330]	validation_0-rmse:7997.68
[331]	validation_0-rmse:7994.55
[332]	validation_0-rmse:7987.29
[333]	validation_0-rmse:7988.17
[334]	validation_0-rmse:7976.72
[335]	validation_0-rmse:7952.68
[336]	validation_0-rmse:7942.03
[337]	validation_0-rmse:7962.13
[338]	validation_0-rmse:7958.61
[339]	validation_0-rmse:7952.73
[340]	validation_0-rmse:7951.35
[341]	validation_0-rmse:7949.31
[342]	validation_0-rmse:7948.9
[343]	validation_0-rmse:7923.9
[344]	validation_0-rmse:7909.56
[345]	validation_0-rmse:7905.37
[346]	validation_0-rmse:7903.81
[347]	validation_0-rmse:7893.32
[348]	validation_0-rmse:7900.47
[349]	validation_0-rmse:7901.41
[350]	validation_0-rmse:7873.41
[351]	validation_0-rmse:7866.24
[352]	validation_0-rmse:7861.57
[353]	validation_0-rmse:7857.11
[354]	validation_0-rmse:7856.24
[355]	validation_0-rmse:7855.39
[356]	validation_0-rmse:7838.3
[357]	validation_0-rmse:7840.04
[358]	validation_0-rmse:7827.7
[359]	validation_0-rmse:7820.83
[360]	validation_0-rmse:7833.01
[361]	validation_0-rmse:7839.33
[362]	validation_0-rmse:7846.39
[363]	validation_0-rmse:7857.2
[364]	validation_0-rmse:7841.41
[365]	validation_0-rmse:7834.43
[366]	validation_0-rmse:7835.18
[367]	validation_0-rmse:7838.01
[368]	validation_0-rmse:7848.69
[369]	validation_0-rmse:7840
Stopping. Best iteration:
```

Out[64]:
```
XGBRegressor(colsample_bytree=0.8, gamma=8, learning_rate=0.2,
             min_child_weight=3, n_estimators=600, objective='reg:squarederror',
             subsample=0.6)
```

In [65]:
```
y_pred = xgbmodel.predict(X_test_scaled)
```

In [66]:
```
y_pred
```

Out[66]:
```
array([149618.08 , 147258.4  , 175656.98 , ..., 552318.9  ,  20316.541,
       777536.5  ], dtype=float32)
```

In [67]:
```
y_test
```

Out[67]:
```
array([150000, 147137, 180000, ..., 550000,  22000, 750000], dtype=int64)
```

Model Evaluation

In [68]:
```
mse = mean_squared_error(y_test,y_pred)
mse
```

Out[68]: 61165427.647468194

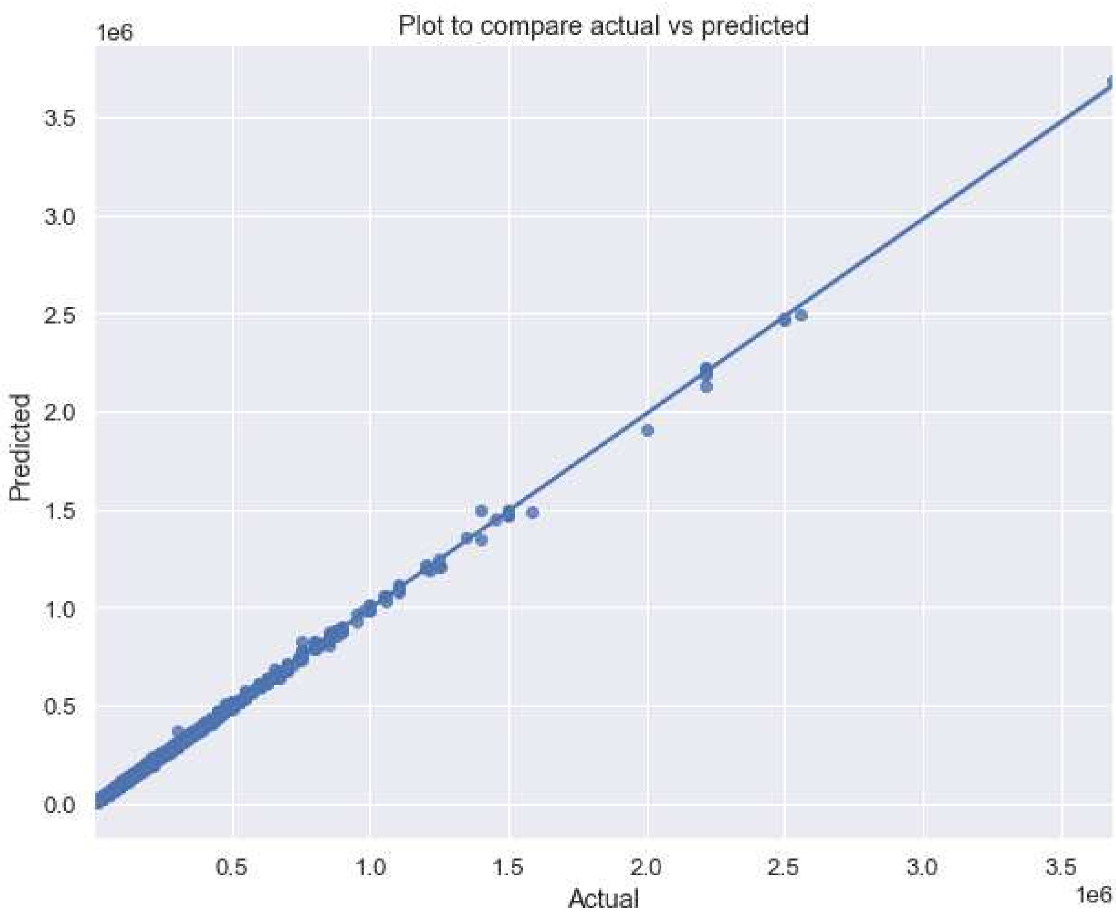
In [69]:
```
rmse = np.sqrt(mse)
rmse
```

Out[69]: 7820.832925428607

In [70]:
```
r2score = r2_score(y_test,y_pred)
r2score
```

Out[70]: 0.9991812748989954

In [71]:
```
fig, ax = plt.subplots(figsize=(10,8))
sns.regplot(x=y_test, y=y_pred, ax=ax)
plt.title("Plot to compare actual vs predicted")
plt.ylabel("Predicted")
plt.xlabel("Actual")
plt.show()
```



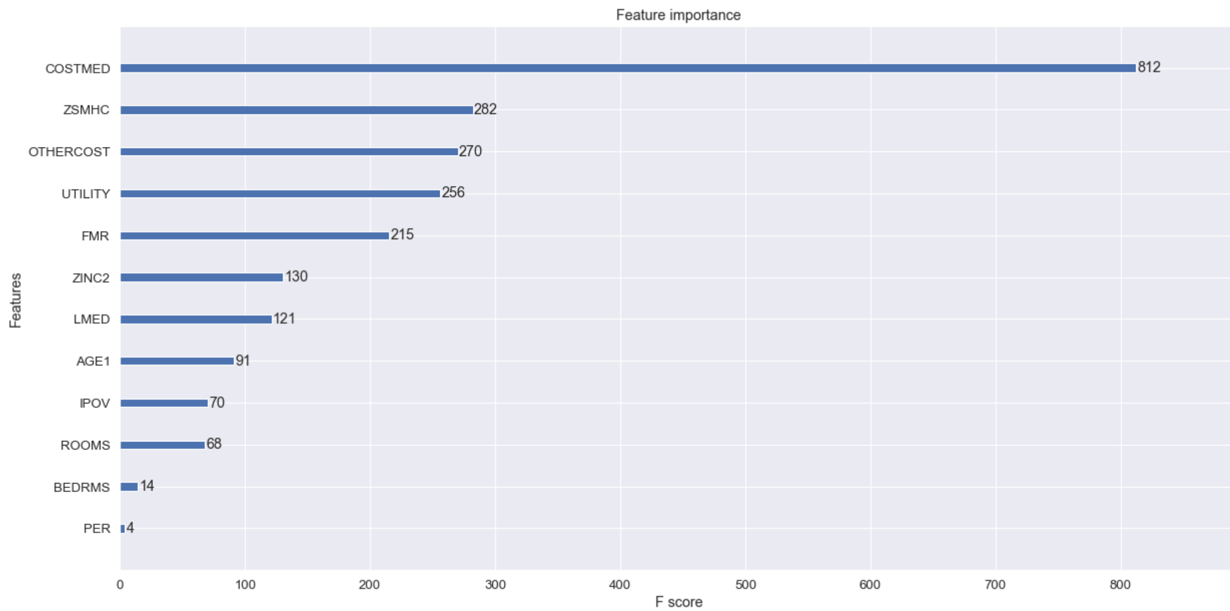
In [ ]:

Available importance_types = [‘weight’, ‘gain’, ‘cover’, ‘total_gain’, ‘total_cover’]

In [72]:
```
X.columns
```
Out[72]:
```
Index(['AGE1', 'LMED', 'FMR', 'IPOV', 'PER', 'ZINC2', 'ZSMHC', 'BEDRMS', 'ROOMS
', 'UTILITY', 'OTHERCOST', 'COSTMED'], dtype='object')
```

In [73]:
```
xgbmodel.get_booster().feature_names = ['AGE1', 'LMED', 'FMR', 'IPOV', 'PER', 'ZINC
2', 'ZSMHC', 'BEDRMS', 'ROOMS', 'UTILITY', 'OTHERCOST', 'COSTMED']
```

In [74]:
```
fig, ax = plt.subplots(figsize=(20,10))
xgb.plot_importance(xgbmodel.get_booster(),ax=ax)
plt.show()
```



Cross-Validation

In [75]:
```
cv = cross_val_score(xgbmodel,X.values,y.values,cv=5,verbose=1,scoring='r2')
```
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:   24.2s finished
```

In [76]:
```
cv.mean()
```
Out[76]:
```
0.9959882064388215
```

In [ ]: