Final Project Milestone2

Data Cleaning/formatting flat file source

In this phase of the project, we will cleanup the properties dataset. The transactions dataset and properties datasets have an ID that is common between them, so we will merge them by that index.

Please note the webscraping and API sections have been pushed further down to allow room for documenting this phase of the project.

Dataset from file

Data Source

https://www.kaggle.com/c/zillow-prize-1 (https://www.kaggle.com/c/zillow-prize-1)

Description

There are two data sets with over 1 million records each and 58 columns. properties_2016 and properties_2017 datasets contain data for each year. The data we will use for this project will be a small sample of the master data.

The two datasets are linked by parcleid.

I transactions dataset, the trabsaction date shows the date the property was sold and logerror is the log10(estimated price - price sold).

Properties dataset has the physical information about the properities. The columns on the properties dataset will have to be renamed. Subsets of data can be used to group by region, and other features such as number of bedrooms, square footage, etc.

In [75]:
```
# Load Libraries
import pandas as pd
import matplotlib.pyplot as plt
import xlrd

# Load Data
transactions_2016 = "Data/transactions_2016.json"
transactions_2017 = "Data/transactions_2017.json"

properties_2016  =  "Data/properties_2016.csv"
properties_2017  =  "Data/properties_2017.csv"
data_dictionary = "Data/data_dictionary.xlsx"

transactions_2016 = pd.read_json(transactions_2016)
transactions_2017 = pd.read_json(transactions_2017)
properties_2016 = pd.read_csv(properties_2016)
properties_2017 = pd.read_csv(properties_2017)
data_dictionary = pd.read_excel(data_dictionary)
```

```
c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc540\finalproject\venv\lib\site-packages\IPython\core\interactiveshell.p
y:3063: DtypeWarning: Columns (50) have mixed types.Specify dtype option
on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
c:\users\safar\documents\github\safarie1103\bellevue university\courses\d
sc540\finalproject\venv\lib\site-packages\IPython\core\interactiveshell.p
y:3063: DtypeWarning: Columns (23,50) have mixed types.Specify dtype opti
on on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [76]:
```
transactions_2016.head()
```

Out[76]:

	parcelid	logerror	transactiondate
0	11016594	0.0276	2016-01-01
1	14366692	-0.1684	2016-01-01
2	12098116	-0.0040	2016-01-01
3	12643413	0.0218	2016-01-02
4	14432541	-0.0050	2016-01-02

In [77]:
```
properties_2016.head()
```

Out[77]:

	Unnamed: 0	parcelid	airconditioningtypeid	architecturalstyletypeid	basements
0	0	10754147	NaN	NaN	N
1	1	10759547	NaN	NaN	N
2	2	10843547	NaN	NaN	N
3	3	10859147	NaN	NaN	N
4	4	10879947	NaN	NaN	N

5 rows × 59 columns

In [78]:
```
print(len(properties_2016.columns))
print(properties_2016.columns)
```

```
59
Index(['Unnamed: 0', 'parcelid', 'airconditioningtypeid',
       'architecturalstyletypeid', 'basementsqft', 'bathroomcnt', 'bedroo
mcnt',
       'buildingclasstypeid', 'buildingqualitytypeid', 'calculatedbathnb
r',
       'decktypeid', 'finishedfloor1squarefeet',
       'calculatedfinishedsquarefeet', 'finishedsquarefeet12',
       'finishedsquarefeet13', 'finishedsquarefeet15', 'finishedsquarefee
t50',
       'finishedsquarefeet6', 'fips', 'fireplacecnt', 'fullbathcnt',
       'garagecarcnt', 'garagetotalsqft', 'hashottuborspa',
       'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
et',
       'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
id7',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
       'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
t',
       'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
       'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
t',
       'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
       'censustractandblock'],
      dtype='object')
```

In [79]:
```
print(len(properties_2017.columns))
print(properties_2017.columns)
```

```
59
Index(['Unnamed: 0', 'parcelid', 'airconditioningtypeid',
       'architecturalstyletypeid', 'basementsqft', 'bathroomcnt', 'bedroo
mcnt',
       'buildingclasstypeid', 'buildingqualitytypeid', 'calculatedbathnb
r',
       'decktypeid', 'finishedfloor1squarefeet',
       'calculatedfinishedsquarefeet', 'finishedsquarefeet12',
       'finishedsquarefeet13', 'finishedsquarefeet15', 'finishedsquarefee
t50',
       'finishedsquarefeet6', 'fips', 'fireplacecnt', 'fullbathcnt',
       'garagecarcnt', 'garagetotalsqft', 'hashottuborspa',
       'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
et',
       'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
id7',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
       'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
t',
       'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
       'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
t',
       'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
       'censustractandblock'],
      dtype='object')
```

In [80]:
```
print(len(transactions_2016.columns))
print(transactions_2016.columns)
```

```
3
Index(['parcelid', 'logerror', 'transactiondate'], dtype='object')
```

In [81]:
```
print(len(transactions_2017.columns))
print(transactions_2017.columns)
```

```
3
Index(['parcelid', 'logerror', 'transactiondate'], dtype='object')
```

In [82]:
```
data_dictionary.head()
```

Out[82]:

	Feature	Description
0	'airconditioningtypeid'	Type of cooling system present in the home (i...
1	'architecturalstyletypeid'	Architectural style of the home (i.e. ranch, ...
2	'basementsqft'	Finished living area below or partially below...
3	'bathroomcnt'	Number of bathrooms in home including fractio...
4	'bedroomcnt'	Number of bedrooms in home

Cleaning/formatting flat file sources

We will first combine the properties_2016 and properties_2017 and calle the result properties. We will also combine the two transactions datasets.

In [83]:
```
properties = pd.concat([properties_2016,properties_2017],axis=0)
print(properties_2016.shape)
print(properties_2017.shape)
print(properties.shape)
```

```
(20000, 59)
(20000, 59)
(40000, 59)
```

In [84]:
```
transactions = pd.concat([transactions_2016,transactions_2017],axis=0)
print(properties_2016.shape)
print(properties_2017.shape)
print(properties.shape)
```

```
(20000, 59)
(20000, 59)
(40000, 59)
```

In [85]:
```
properties.columns
```

Out[85]:
```
Index(['Unnamed: 0', 'parcelid', 'airconditioningtypeid',
       'architecturalstyletypeid', 'basementsqft', 'bathroomcnt', 'bedroo
mcnt',
       'buildingclasstypeid', 'buildingqualitytypeid', 'calculatedbathnb
r',
       'decktypeid', 'finishedfloor1squarefeet',
       'calculatedfinishedsquarefeet', 'finishedsquarefeet12',
       'finishedsquarefeet13', 'finishedsquarefeet15', 'finishedsquarefee
t50',
       'finishedsquarefeet6', 'fips', 'fireplacecnt', 'fullbathcnt',
       'garagecarcnt', 'garagetotalsqft', 'hashottuborspa',
       'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
et',
       'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
id7',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
       'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
t',
       'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
       'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
t',
       'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
       'censustractandblock'],
      dtype='object')
```

Get rid of the Unamed column.

In [86]:
```
properties = properties.loc[:, ~properties.columns.str.contains('^Unname
d')]
properties.columns
```

Out[86]:
```
Index(['parcelid', 'airconditioningtypeid', 'architecturalstyletypeid',
       'basementsqft', 'bathroomcnt', 'bedroomcnt', 'buildingclasstypei
d',
       'buildingqualitytypeid', 'calculatedbathnbr', 'decktypeid',
       'finishedfloor1squarefeet', 'calculatedfinishedsquarefeet',
       'finishedsquarefeet12', 'finishedsquarefeet13', 'finishedsquarefee
t15',
       'finishedsquarefeet50', 'finishedsquarefeet6', 'fips', 'fireplacec
nt',
       'fullbathcnt', 'garagecarcnt', 'garagetotalsqft', 'hashottuborsp
a',
       'heatingorsystemtypeid', 'latitude', 'longitude', 'lotsizesquarefe
et',
       'poolcnt', 'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltype
id7',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'rawcensustractandblock', 'regionidcity',
       'regionidcounty', 'regionidneighborhood', 'regionidzip', 'roomcn
t',
       'storytypeid', 'threequarterbathnbr', 'typeconstructiontypeid',
       'unitcnt', 'yardbuildingsqft17', 'yardbuildingsqft26', 'yearbuil
t',
       'numberofstories', 'fireplaceflag', 'structuretaxvaluedollarcnt',
       'taxvaluedollarcnt', 'assessmentyear', 'landtaxvaluedollarcnt',
       'taxamount', 'taxdelinquencyflag', 'taxdelinquencyyear',
       'censustractandblock'],
      dtype='object')
```

Rename column names in properties dataset.

In [87]:
```
properties = properties.rename(columns=
                        {
  'parcelid':'parcelid',
  'yearbuilt':'build_year',
  'basementsqft':'area_basement',
  'yardbuildingsqft17':'area_patio',
  'yardbuildingsqft26':'area_shed',
  'poolsizesum':'area_pool',
  'lotsizesquarefeet':'area_lot',
  'garagetotalsqft':'area_garage',
  'finishedfloor1squarefeet':'area_firstfloor_finished',
  'calculatedfinishedsquarefeet':'area_total_calc',
  'finishedsquarefeet6':'area_base',
  'finishedsquarefeet12':'area_live_finished',
  'finishedsquarefeet13':'area_liveperi_finished',
  'finishedsquarefeet15':'area_total_finished',
  'finishedsquarefeet50':'area_unknown',
  'unitcnt': 'num_unit',
  'numberofstories': 'num_story',
  'roomcnt':'num_room',
  'bathroomcnt':'num_bathroom',
  'bedroomcnt':'num_bedroom',
  'calculatedbathnbr':'num_bathroom_calc',
  'fullbathcnt':'num_bath',
  'threequarterbathnbr':'num_75_bath',
  'fireplacecnt':'num_fireplace',
  'poolcnt': 'num_pool',
  'garagecarcnt':'num_garage',
  'regionidcounty':'region_county',
  'regionidcity':'region_city',
  'regionidzip':'region_zip',
  'regionidneighborhood':'region_neighbor',
  'taxvaluedollarcnt':'tax_total',
  'structuretaxvaluedollarcnt':'tax_building',
  'landtaxvaluedollarcnt':'tax_land',
  'taxamount':'tax_property',
  'assessmentyear':'tax_year',
  'taxdelinquencyflag':'tax_delinquency',
  'taxdelinquencyyear':'tax_delinquency_year',
  'propertyzoningdesc':'zoning_property',
  'propertylandusetypeid':'zoning_landuse',
  'propertycountylandusecode':'zoning_landuse_county',
  'fireplaceflag':'flag_fireplace',
  'hashottuborspa':'flag_tub',
  'buildingqualitytypeid':'quality',
  'buildingclasstypeid':'framing',
  'typeconstructiontypeid':'material',
  'decktypeid':'deck',
  'storytypeid':'story',
  'heatingorsystemtypeid':'heating',
  'airconditioningtypeid':'aircon',
  'architecturalstyletypeid':'architectural_style'
})
```

In [88]:
```
properties.columns
```

Out[88]:
```
Index(['parcelid', 'aircon', 'architectural_style', 'area_basement',
       'num_bathroom', 'num_bedroom', 'framing', 'quality',
       'num_bathroom_calc', 'deck', 'area_firstfloor_finished',
       'area_total_calc', 'area_live_finished', 'area_liveperi_finished',
       'area_total_finished', 'area_unknown', 'area_base', 'fips',
       'num_fireplace', 'num_bath', 'num_garage', 'area_garage', 'flag_tub',
       'heating', 'latitude', 'longitude', 'area_lot', 'num_pool', 'area_pool',
       'pooltypeid10', 'pooltypeid2', 'pooltypeid7', 'zoning_landuse_county',
       'zoning_landuse', 'zoning_property', 'rawcensustractandblock',
       'region_city', 'region_county', 'region_neighbor', 'region_zip',
       'num_room', 'story', 'num_75_bath', 'material', 'num_unit',
       'area_patio', 'area_shed', 'build_year', 'num_story', 'flag_fireplace',
       'tax_building', 'tax_total', 'tax_year', 'tax_land', 'tax_property',
       'tax_delinquency', 'tax_delinquency_year', 'censustractandblock'],
      dtype='object')
```

In [89]:
```
# Check new column names
properties[['num_bedroom','num_bathroom']]
```

Out[89]:

	num_bedroom	num_bathroom
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0
...	...	...
19995	2.0	1.0
19996	5.0	3.0
19997	8.0	5.0
19998	4.0	2.0
19999	2.0	1.0

40000 rows × 2 columns

Rename column names in transactions dataset.

In [90]:
```
transactions = transactions.rename(columns={'parcelid':'parcelid','date':
'transactiondate'})
```

In [91]:
```
transactions.columns
```

Out[91]:
```
Index(['parcelid', 'logerror', 'transactiondate'], dtype='object')
```

Check out the new columns

In [92]:
```
transactions[['parcelid','transactiondate']]
```

Out[92]:

	parcelid	transactiondate
0	11016594	2016-01-01
1	14366692	2016-01-01
2	12098116	2016-01-01
3	12643413	2016-01-02
4	14432541	2016-01-02
...	...	...
77608	10833991	2017-09-20
77609	11000655	2017-09-20
77610	17239384	2017-09-21
77611	12773139	2017-09-21
77612	12826780	2017-09-25

167888 rows × 2 columns

In [93]:
```
propertiesAndTransactions = pd.merge(properties,transactions,on='parcelid')
```

check out the merge

In [146]:
```
propertiesAndTransactions[['parcelid','num_bedroom','transactiondate','logerror']].head()
```

Out[146]:

	parcelid	num_bedroom	transactiondate	logerror
0	17054981	4.0	2017-06-15	-0.013099
1	17054981	4.0	2017-06-15	-0.013099
2	17055743	3.0	2017-07-26	0.073985
3	17055743	3.0	2017-07-26	0.073985
4	17068109	3.0	2017-07-28	0.071886

let's take of missings

In [147]:
```
column_names = propertiesAndTransactions.columns
print('sum\n', propertiesAndTransactions.isnull()[column_names].sum())
```

	sum
parcelid	0
aircon	1485
architectural_style	2234
area_basement	2234
num_bathroom	0
num_bedroom	0
framing	2234
quality	705
num_bathroom_calc	26
deck	2214
area_firstfloor_finished	2000
area_total_calc	9
area_live_finished	102
area_liveperi_finished	2234
area_total_finished	2145
area_unknown	2000
area_base	2230
fips	0
num_fireplace	1982
num_bath	26
num_garage	1593
area_garage	1593
flag_tub	2192
heating	752
latitude	0
longitude	0
area_lot	216
num_pool	1708
area_pool	2206
pooltypeid10	2216
pooltypeid2	2210
pooltypeid7	1732
zoning_landuse_county	0
zoning_landuse	0
zoning_property	678
rawcensustractandblock	0
region_city	42
region_county	0
region_neighbor	1186
region_zip	2
num_room	0
story	2234
num_75_bath	1984
material	2234
num_unit	679
area_patio	2137
area_shed	2234
build_year	11
num_story	1792
flag_fireplace	2234
tax_building	6
tax_total	0
tax_year	0
tax_land	0
tax_property	0
tax_delinquency	2166

```
tax_delinquency_year        2166
censustractandblock            8
logerror                       0
transactiondate                0
dtype: int64
```

In [148]:
```
print('mean\n', propertiesAndTransactions.isnull()[column_names].mean())
```

	mean
parcelid	0.000000
aircon	0.664727
architectural_style	1.000000
area_basement	1.000000
num_bathroom	0.000000
num_bedroom	0.000000
framing	1.000000
quality	0.315577
num_bathroom_calc	0.011638
deck	0.991047
area_firstfloor_finished	0.895255
area_total_calc	0.004029
area_live_finished	0.045658
area_liveperi_finished	1.000000
area_total_finished	0.960161
area_unknown	0.895255
area_base	0.998209
fips	0.000000
num_fireplace	0.887198
num_bath	0.011638
num_garage	0.713071
area_garage	0.713071
flag_tub	0.981200
heating	0.336616
latitude	0.000000
longitude	0.000000
area_lot	0.096688
num_pool	0.764548
area_pool	0.987466
pooltypeid10	0.991943
pooltypeid2	0.989257
pooltypeid7	0.775291
zoning_landuse_county	0.000000
zoning_landuse	0.000000
zoning_property	0.303491
rawcensustractandblock	0.000000
region_city	0.018800
region_county	0.000000
region_neighbor	0.530886
region_zip	0.000895
num_room	0.000000
story	1.000000
num_75_bath	0.888093
material	1.000000
num_unit	0.303939
area_patio	0.956580
area_shed	1.000000
build_year	0.004924
num_story	0.802149
flag_fireplace	1.000000
tax_building	0.002686
tax_total	0.000000
tax_year	0.000000
tax_land	0.000000
tax_property	0.000000
tax_delinquency	0.969561

```
tax_delinquency_year        0.969561
censustractandblock         0.003581
logerror                    0.000000
transactiondate             0.000000
dtype: float64
```

Let's look at columns woth more than 80% missing values

In [149]:
```
propertiesAndTransactions.isnull()[column_names].sum()
# this shows columns and the number of NaN's.Note parcelID has no missing
values.
```

Out[149]:

parcelid	0
aircon	1485
architectural_style	2234
area_basement	2234
num_bathroom	0
num_bedroom	0
framing	2234
quality	705
num_bathroom_calc	26
deck	2214
area_firstfloor_finished	2000
area_total_calc	9
area_live_finished	102
area_liveperi_finished	2234
area_total_finished	2145
area_unknown	2000
area_base	2230
fips	0
num_fireplace	1982
num_bath	26
num_garage	1593
area_garage	1593
flag_tub	2192
heating	752
latitude	0
longitude	0
area_lot	216
num_pool	1708
area_pool	2206
pooltypeid10	2216
pooltypeid2	2210
pooltypeid7	1732
zoning_landuse_county	0
zoning_landuse	0
zoning_property	678
rawcensustractandblock	0
region_city	42
region_county	0
region_neighbor	1186
region_zip	2
num_room	0
story	2234
num_75_bath	1984
material	2234
num_unit	679
area_patio	2137
area_shed	2234
build_year	11
num_story	1792
flag_fireplace	2234
tax_building	6
tax_total	0
tax_year	0
tax_land	0
tax_property	0
tax_delinquency	2166
tax_delinquency_year	2166

```
censustractandblock           8
logerror                      0
transactiondate               0
dtype: int64
```

Make a list of columns with moe than 80% missing data

In [152]:
```
remove_columns = propertiesAndTransactions.columns[propertiesAndTransacti
ons.isnull().mean() > .8]
print(remove_columns)
```

```
Index(['architectural_style', 'area_basement', 'framing', 'deck',
       'area_firstfloor_finished', 'area_liveperi_finished',
       'area_total_finished', 'area_unknown', 'area_base', 'num_fireplac
e',
       'flag_tub', 'area_pool', 'pooltypeid10', 'pooltypeid2', 'story',
       'num_75_bath', 'material', 'area_patio', 'area_shed', 'num_story',
       'flag_fireplace', 'tax_delinquency', 'tax_delinquency_year'],
      dtype='object')
```

Drop the columns

In [153]:
```
propertiesAndTransactions = propertiesAndTransactions.drop(columns = remo
ve_columns)
```

Check results

In [154]:
```
print(len(propertiesAndTransactions.columns))
print(propertiesAndTransactions.columns)
```

```
37
Index(['parcelid', 'aircon', 'num_bathroom', 'num_bedroom', 'quality',
       'num_bathroom_calc', 'area_total_calc', 'area_live_finished', 'fip
s',
       'num_bath', 'num_garage', 'area_garage', 'heating', 'latitude',
       'longitude', 'area_lot', 'num_pool', 'pooltypeid7',
       'zoning_landuse_county', 'zoning_landuse', 'zoning_property',
       'rawcensustractandblock', 'region_city', 'region_county',
       'region_neighbor', 'region_zip', 'num_room', 'num_unit', 'build_ye
ar',
       'tax_building', 'tax_total', 'tax_year', 'tax_land', 'tax_propert
y',
       'censustractandblock', 'logerror', 'transactiondate'],
      dtype='object')
```

Let's check the missing values mean

In [155]:
```
print('mean\n', propertiesAndTransactions.isnull()[propertiesAndTransactions.columns].mean())
# we see the means to all be below 80%.
```

```
mean
 parcelid                  0.000000
aircon                     0.664727
num_bathroom               0.000000
num_bedroom                0.000000
quality                    0.315577
num_bathroom_calc          0.011638
area_total_calc            0.004029
area_live_finished         0.045658
fips                       0.000000
num_bath                   0.011638
num_garage                 0.713071
area_garage                0.713071
heating                    0.336616
latitude                   0.000000
longitude                  0.000000
area_lot                   0.096688
num_pool                   0.764548
pooltypeid7                0.775291
zoning_landuse_county      0.000000
zoning_landuse             0.000000
zoning_property            0.303491
rawcensustractandblock     0.000000
region_city                0.018800
region_county              0.000000
region_neighbor            0.530886
region_zip                 0.000895
num_room                   0.000000
num_unit                   0.303939
build_year                 0.004924
tax_building               0.002686
tax_total                  0.000000
tax_year                   0.000000
tax_land                   0.000000
tax_property               0.000000
censustractandblock        0.003581
logerror                   0.000000
transactiondate            0.000000
dtype: float64
```

Are there any duplicate?

In [156]:
```
propertiesAndTransactions[propertiesAndTransactions.duplicated(keep=False)]
# There are no duplocates
```

Out[156]:

parcelid	aircon	num_bathroom	num_bedroom	quality	num_bathroom_calc	are

0 rows × 37 columns

The two datasets have been merges, columns with more than 80% missing values were removed. The final dataset 'propertiesAndTransactions' will be used in the next milestone.

Webscaraping Data Source

Description

Using webscraping techniques, we will use 'latitude', 'longitude' from properties dataset to access properties and get current data for those locations. THe property description of homes in given region will be stored into a dataset with as many features as in properties dataset we can grab. This dataset can then be used to do some price comparision between properties in 2016 and 2017. Getting data from years prior(say 10 years), we will be able to create trend charts and see market fluctuations.

In [29]:
```
# Load Libraries
from selenium import webdriver
from bs4 import BeautifulSoup


from selenium.webdriver import Chrome

driver = Chrome("C:/Users/safar/Downloads/chromedriver_win32/chromedrive
r")

#with Chrome() as driver:
products=[] #List to store name of the product
prices=[] #List to store price of the product
ratings=[] #List to store rating of the product
# This open the chromium web browser. This web browswer will be under the
control of this application
driver.get("https://www.zillow.com")

# The field "enter an address will be inspected and filled in for the que
ries"
```

data from API

Description

Googlemap API and matplotlib or equivalant will be used to locate properties by zipcode and display them on the map of the Unites States. We will convert 'longitude' and 'latitude' columns in properties dataset to zip code and use the zipcode in the API call.We will show the density of homes sold in various regions in the dataset. We will also show the properties we extracted using webscraping techniques.

In [26]:
```
# This is a sample code and does not pertain to this project. We will try
to implement a function s
import googlemaps
from datetime import datetime

gmaps = googlemaps.Client(key='I HAVE OBTAINED A KEY FROM GOOGLE WEB SIT
E')

# Geocoding an address
geocode_result = gmaps.geocode('1600 Amphitheatre Parkway, Mountain View,
CA')

# Look up an address with reverse geocoding
reverse_geocode_result = gmaps.reverse_geocode((40.714224, -73.961452))

# Request directions via public transit
now = datetime.now()
directions_result = gmaps.directions("Sydney Town Hall",
                                     "Parramatta, NSW",
                                     mode="transit",
                                     departure_time=now)
```