[Year]

Midterm Project

001672574
MENGQI ZHAO

In [7]:
```
from boto.s3.connection import S3Connection
import os
import json
import boto.s3
import sys
import datetime
import seaborn as sns
from boto.s3.key import Key
from pprint import pprint
import pandas as pd
import urllib
import csv
import io
import requests
import time
import json
import datetime
from pprint import pprint
import scipy
import numpy as np
import matplotlib.pyplot as plt
```

1.insert train_2016 data

rawdataspecificrows= pd.read_csv("train_2016_v2.csv") rawdataspecificrows.shape
rawdataspecificrows.head(2)

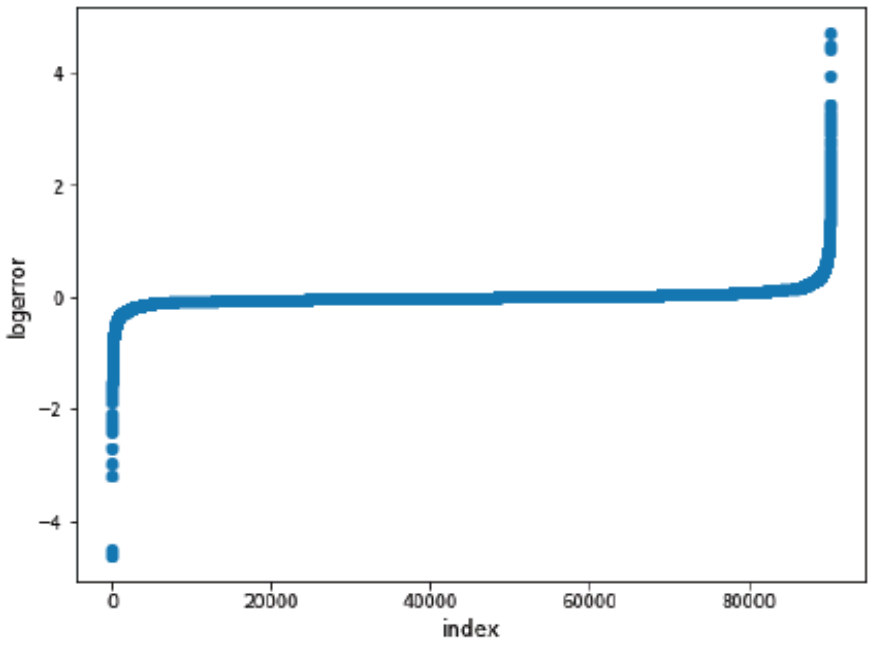
In [9]:
```
rawdataspecificrows.head(2)
```

Out[9]:

	parcelid	logerror	transactiondate
0	11016594	0.0276	2016-01-01
1	14366692	-0.1684	2016-01-01

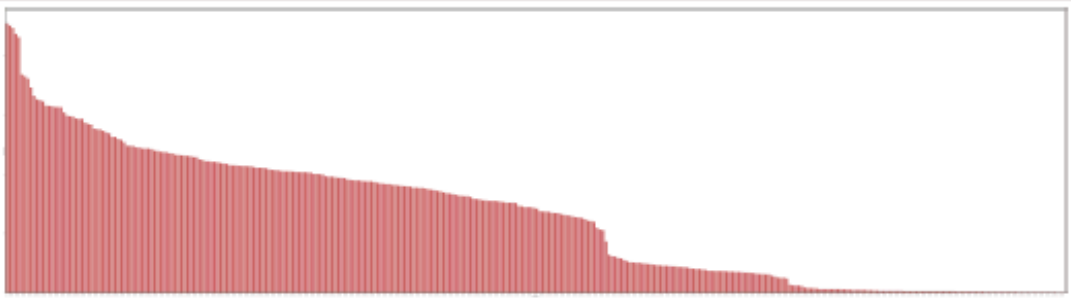
2. analysis the train_2016 by using graph

In [10]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(rawdataspecificrows.shape[0]), np.sort(rawdataspecific
rows.logerror.values))
plt.xlabel('index', fontsize=12)
plt.ylabel('logerror', fontsize=12)
plt.show()
```



3.show number of loggererror in every day

In [11]:
```
color = sns.color_palette()
rawdataspecificrows['transaction_month'] = rawdataspecificrows['transact
iondate']

cnt_srs = rawdataspecificrows['transaction_month'].value_counts()
plt.figure(figsize=(365,100))
sns.barplot(cnt_srs.index, cnt_srs.values, alpha=0.8, color=color[3])
plt.xticks(rotation='vertical')
plt.xlabel('Days of transaction', fontsize=12)
plt.ylabel('Number of Occurrences', fontsize=12)
plt.show()
```



In [12]:
```
(rawdataspecificrows['parcelid'].value_counts().reset_index())['parceli
d'].value_counts()
```

Out[12]:
```
1    90026
2      123
3        1
Name: parcelid, dtype: int64
```

insert the data of properties_2016

In [13]:
```
prop_df = pd.read_csv("properties_2016.csv")
prop_df.shape
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/IPython/core/interactiveshell.py:2728: DtypeWarning: Columns (2
2,32,34,49,55) have mixed types. Specify dtype option on import or set
low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[13]:
```
(2985217, 58)
```

In [14]:
```
prop_df.head()
```

Out[14]:

	parcelid	airconditioningtypeid	architecturalstyletypeid	basementsqft	bathroomcnt
0	10754147	NaN	NaN	NaN	0.0
1	10759547	NaN	NaN	NaN	0.0
2	10843547	NaN	NaN	NaN	0.0
3	10859147	NaN	NaN	NaN	0.0
4	10879947	NaN	NaN	NaN	0.0

5 rows x 58 columns

show how much null value in each column

```
In [15]: prop_df.isnull().sum()
```

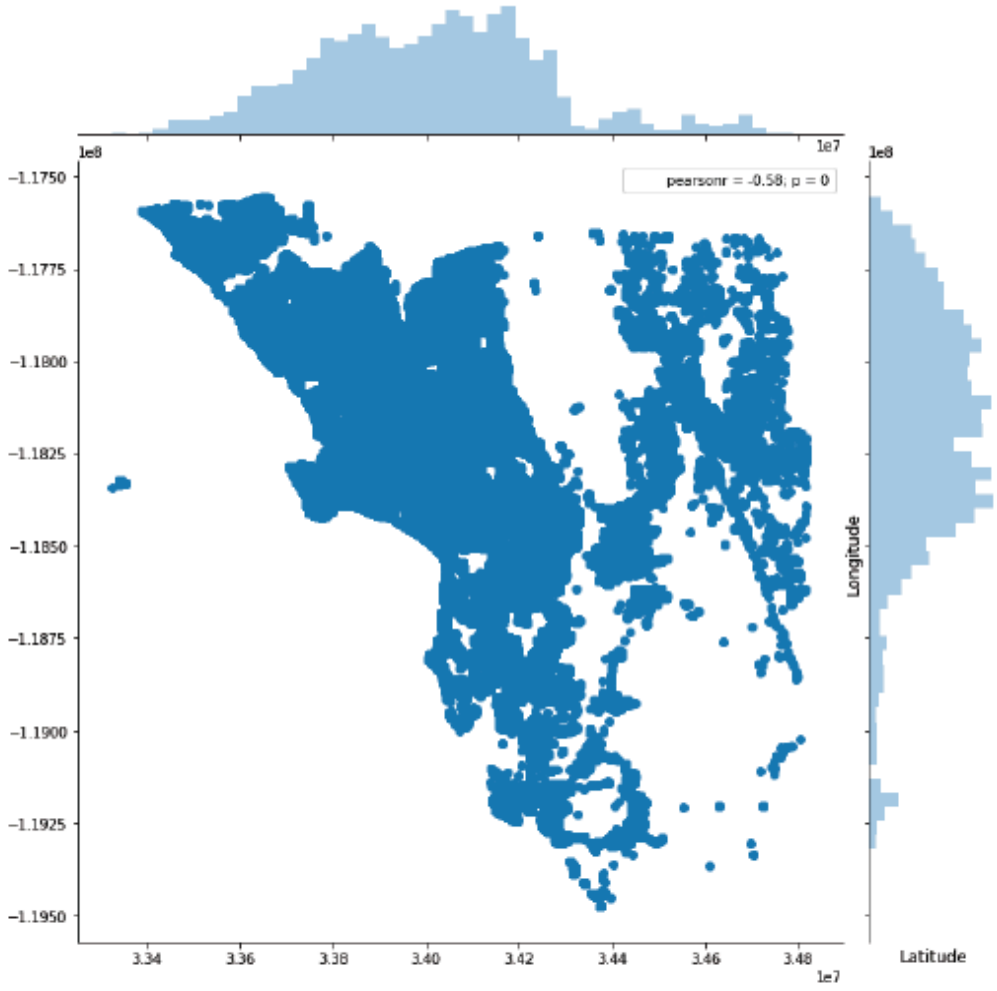
```
Out[15]: parcelid                              0
         airconditioningtypeid           2173698
         architecturalstyletypeid        2979156
         basementsqft                    2983589
         bathroomcnt                       11462
         bedroomcnt                        11450
         buildingclasstypeid             2972588
         buildingqualitytypeid           1046729
         calculatedbathnbr                128912
         decktypeid                      2968121
         finishedfloor1squarefeet        2782500
         calculatedfinishedsquarefeet      55565
         finishedsquarefeet12             276033
         finishedsquarefeet13            2977545
         finishedsquarefeet15            2794419
         finishedsquarefeet50            2782500
         finishedsquarefeet6             2963216
         fips                              11437
         fireplacecnt                    2672580
         fullbathcnt                      128912
         garagecarcnt                    2101950
         garagetotalsqft                 2101950
         hashottuborspa                  2916203
         heatingorsystemtypeid           1178816
         latitude                          11437
         longitude                         11437
         lotsizesquarefeet                276099
         poolcnt                         2467683
         poolsizesum                     2957257
         pooltypeid10                    2948278
         pooltypeid2                     2953142
         pooltypeid7                     2499758
         propertycountylandusecode         12277
         propertylandusetypeid             11437
         propertyzoningdesc              1006588
         rawcensustractandblock            11437
         regionidcity                      62845
         regionidcounty                    11437
         regionidneighborhood            1828815
         regionidzip                       13980
         roomcnt                           11475
         storytypeid                     2983593
         threequarterbathnbr             2673586
         typeconstructiontypeid          2978470
         unitcnt                         1007727
         yardbuildingsqft17              2904862
         yardbuildingsqft26              2982570
         yearbuilt                         59928
         numberofstories                 2303148
         fireplaceflag                   2980054
         structuretaxvaluedollarcnt        54982
         taxvaluedollarcnt                 42550
         assessmentyear                    11439
         landtaxvaluedollarcnt             67733
         taxamount                         31250
         taxdelinquencyflag              2928755
         taxdelinquencyyear              2928753
```

```
censustractandblock                75126
dtype: int64
```

analyis location by using graph

In [16]:
```
plt.figure(figsize=(12,12))
sns.jointplot(x=prop_df.latitude.values, y=prop_df.longitude.values, siz
e=10)
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
```

```
<matplotlib.figure.Figure at 0x17cfcb710>
```



merge data with two table on 'parcelid'

train_df = pd.merge(rawdataspecificrows, prop_df, on='parcelid', how='left') train_df.head()

In [20]:
```
pd.options.display.max_rows = 65

dtype_df = train_df.dtypes.reset_index()
dtype_df.columns = ["Count", "Column Type"]
dtype_df
```

Out[15]:

parcelid	0
airconditioningtypeid	2173698
architecturalstyletypeid	2979156
basementsqft	2983589
bathroomcnt	11462
bedroomcnt	11450
buildingclasstypeid	2972588
buildingqualitytypeid	1046729
calculatedbathnbr	128912
decktypeid	2968121
finishedfloor1squarefeet	2782500
calculatedfinishedsquarefeet	55565
finishedsquarefeet12	276033
finishedsquarefeet13	2977545
finishedsquarefeet15	2794419
finishedsquarefeet50	2782500
finishedsquarefeet6	2963216
fips	11437
fireplacecnt	2672580
fullbathcnt	128912
garagecarcnt	2101950
garagetotalsqft	2101950
hashottuborspa	2916203
heatingorsystemtypeid	1178816
latitude	11437
longitude	11437
lotsizesquarefeet	276099
poolcnt	2467683
poolsizesum	2957257
pooltypeid10	2948278
pooltypeid2	2953142
pooltypeid7	2499758
propertycountylandusecode	12277
propertylandusetypeid	11437
propertyzoningdesc	1006588
rawcensustractandblock	11437
regionidcity	62845
regionidcounty	11437
regionidneighborhood	1828815
regionidzip	13980
roomcnt	11475
storytypeid	2983593
threequarterbathnbr	2673586
typeconstructiontypeid	2978470
unitcnt	1007727
yardbuildingsqft17	2904862
yardbuildingsqft26	2982570
yearbuilt	59928
numberofstories	2303148
fireplaceflag	2980054
structuretaxvaluedollarcnt	54982
taxvaluedollarcnt	42550
assessmentyear	11439
landtaxvaluedollarcnt	67733
taxamount	31250
taxdelinquencyflag	2928755
taxdelinquencyyear	2928753

In [24]:
```
missing_df = train_df.isnull().sum(axis=0).reset_index()
missing_df.columns = ['column_name', 'missing_count']
missing_df['missing_ratio'] = missing_df['missing_count'] / train_df.sha
pe[0]
missing_df.ix[missing_df['missing_ratio']>0.99]
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/ipykernel_launcher.py:4: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is
-deprecated
  after removing the cwd from sys.path.
```

Out[24]:

	column_name	missing_count	missing_ratio
5	architecturalstyletypeid	90014	0.997109
6	basementsqft	90232	0.999524
9	buildingclasstypeid	90259	0.999823
12	decktypeid	89617	0.992711
16	finishedsquarefeet13	90242	0.999634
19	finishedsquarefeet6	89854	0.995336
44	storytypeid	90232	0.999524
46	typeconstructiontypeid	89976	0.996688
49	yardbuildingsqft26	90180	0.998948
52	fireplaceflag	90053	0.997541

In [89]:
```
#train_df.drop(['fireplaceflag','basementsqft','buildingclasstypeid','de
cktypeid','finishedsquarefeet13','yardbuildingsqft26','typeconstructiont
ypeid','storytypeid','finishedsquarefeet6'],axis=1,inplace=True)
#train_df.drop(['taxdelinquencyflag','propertyzoningdesc','propertycount
ylandusecode','hashottuborspa','transactiondate','transaction_month'],ax
is=1,inplace=True)
#train_df.drop(['architecturalstyletypeid'],axis=1,inplace=True)
train_df.head()
train_df.to_csv('values.csv',mode='a',encoding='utf-8',index=False)
train_df.head()
```

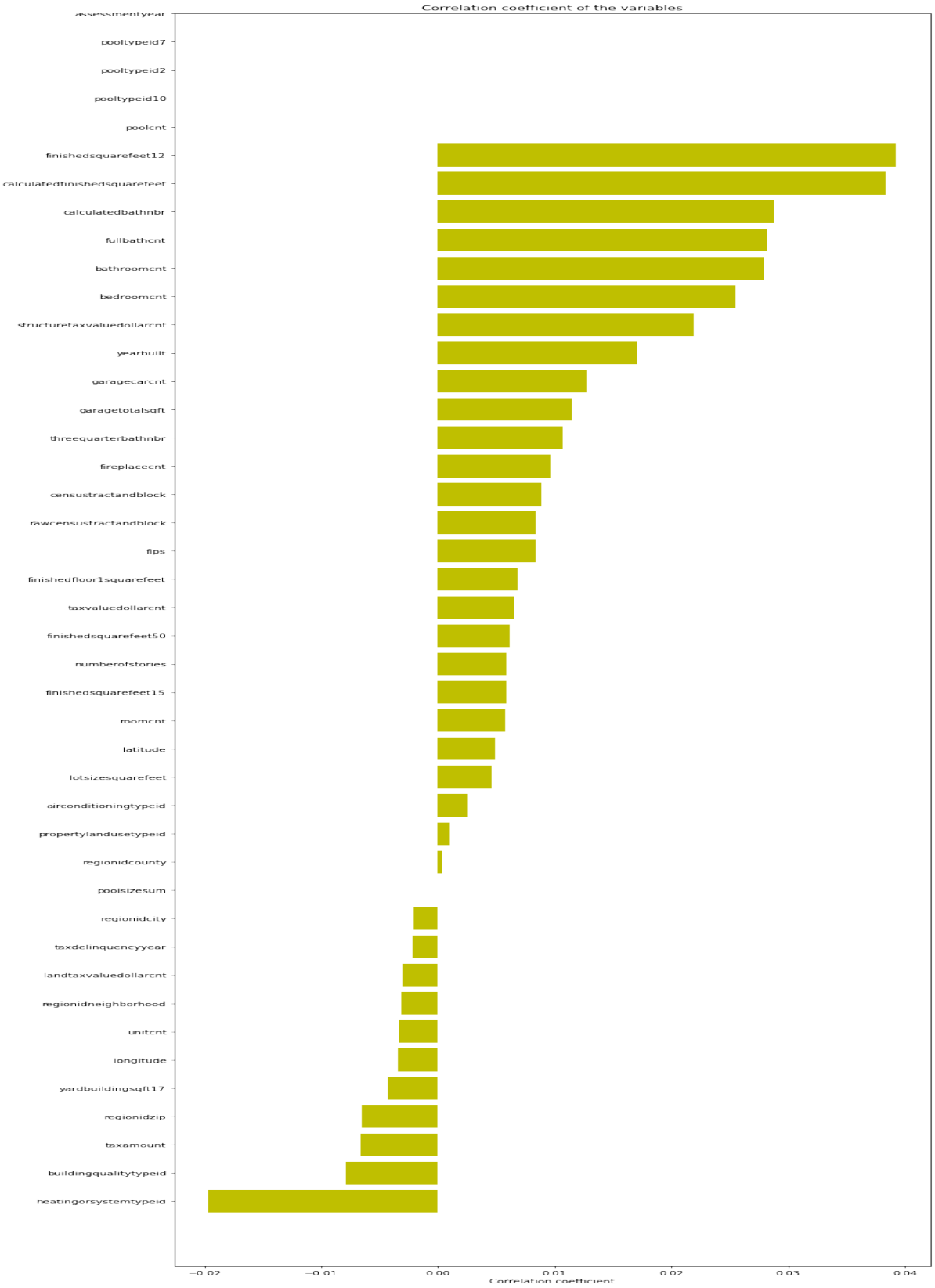
Out[89]:

	parcelid	logerror	airconditioningtypeid	bathroomcnt	bedroomcnt	buildingqualityty
0	11016594	0.0276	1.000000	2.0	3.0	4.000000
1	14366692	-0.1684	1.816372	3.5	4.0	5.565407
2	12098116	-0.0040	1.000000	3.0	2.0	4.000000
3	12643413	0.0218	1.000000	2.0	2.0	4.000000
4	14432541	-0.0050	1.816372	2.5	4.0	5.565407

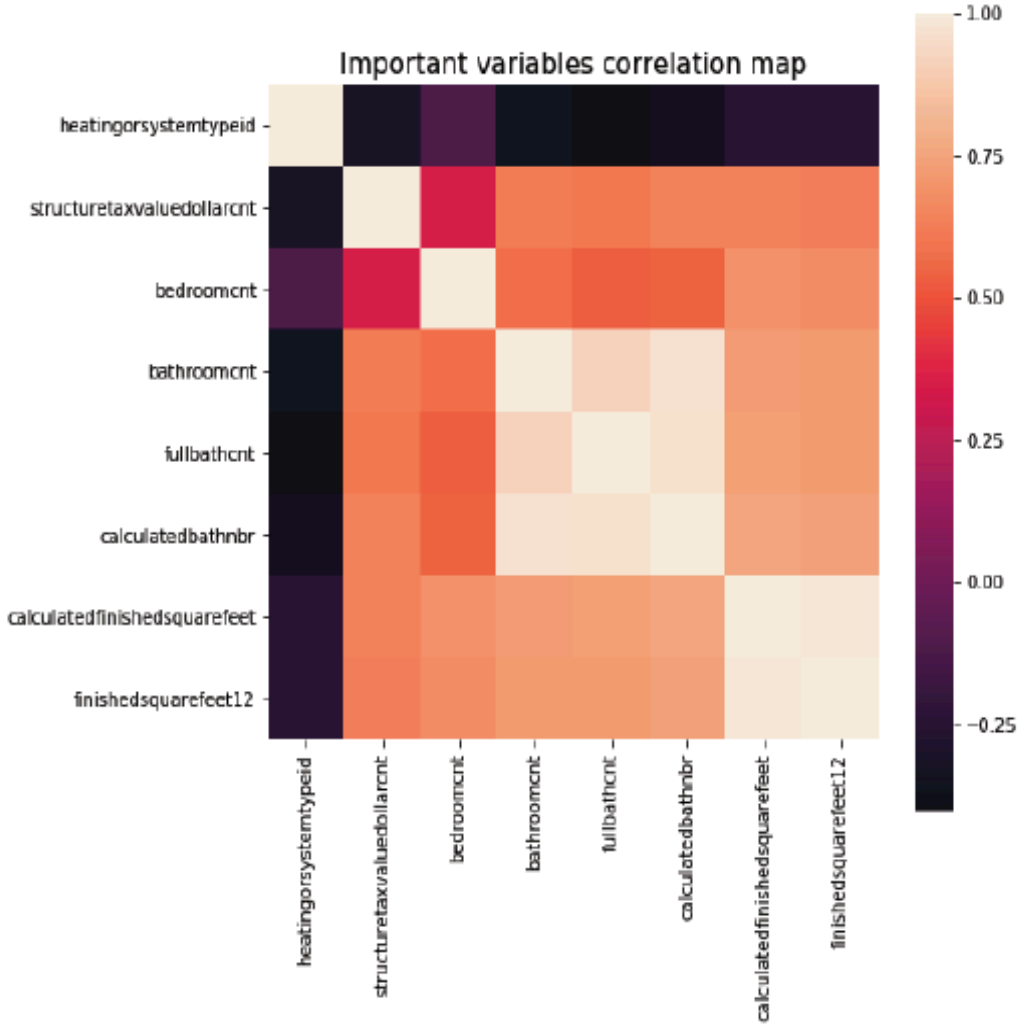
5 rows x 45 columns

analysis data

In [32]:
```
# Let us just impute the missing values with mean values to compute corr
elation coefficients #
mean_values = train_df.mean(axis=0)
train_df_new = train_df.fillna(mean_values, inplace=True)

# Now let us look at the correlation coefficient of each of these variab
les #
x_cols = [col for col in train_df_new.columns if col not in ['logerror']
 if train_df_new[col].dtype=='float64']

labels = []
values = []
for col in x_cols:
    labels.append(col)
    values.append(np.corrcoef(train_df_new[col].values, train_df_new.log
error.values)[0,1])
corr_df = pd.DataFrame({'col_labels':labels, 'corr_values':values})
corr_df = corr_df.sort_values(by='corr_values')

ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(12,40))
rects = ax.barh(ind, np.array(corr_df.corr_values.values), color='y')
ax.set_yticks(ind)
ax.set_yticklabels(corr_df.col_labels.values, rotation='horizontal')
ax.set_xlabel("Correlation coefficient")
ax.set_title("Correlation coefficient of the variables")
#autolabel(rects)
plt.show()
```


Correlation coefficient of the variables
assessmentyear
pooltypeid7
pooltypeid2
pooltypeid10
poolcnt
finishedsquarefeet12
calculatedfinishedsquarefeet
calculatedbathnbr
fullbathcnt
bathroomcnt
bedroomcnt
structuretaxvaluedollarcnt
yearbuilt
garagecarcnt
garagetotalsqft
threequarterbathnbr
fireplacecnt
censustractandblock
rawcensustractandblock
fips
finishedfloor1squarefeet
taxvaluedollarcnt
finishedsquarefeet50
numberofstories
finishedsquarefeet15
roomcnt
latitude
lotsizesquarefeet
airconditioningtypeid
propertylandusetypeid
regionidcounty
poolsizesum
regionidcity
taxdelinquencyyear
landtaxvaluedollarcnt
regionidneighborhood
unitcnt
longitude
yardbuildingsqft17
regionidzip
taxamount
buildingqualitytypeid
heatingorsystemtypeid
−0.02
−0.01
0.00
0.01
0.02
0.03
0.04
Correlation coefficient

In [34]:
```
cols_to_use = corr_df_sel.col_labels.tolist()

temp_df = train_df[cols_to_use]
corrmat = temp_df.corr(method='spearman')
f, ax = plt.subplots(figsize=(8, 8))

# Draw the heatmap using seaborn
sns.heatmap(corrmat, vmax=1., square=True)
plt.title("Important variables correlation map", fontsize=15)
plt.show()
```



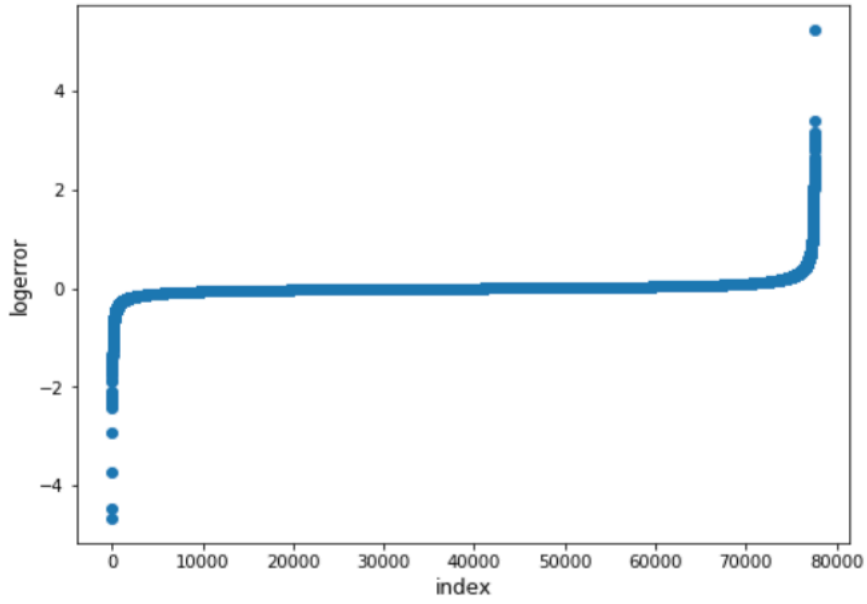
In [35]:
```
corr_df_sel.to_csv('influence_values.csv',mode='a',encoding='utf-8',inde
x=False)
```

In [96]:
```
train_df.to_csv('midterm_2016.csv',mode='a',encoding='utf-8',index=False
)
```

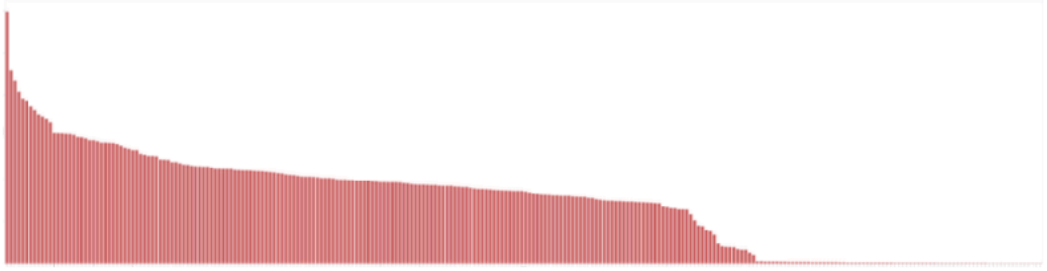
In [2]:
```
rawdataspecificrows= pd.read_csv("train_2017.csv")
rawdataspecificrows.shape
rawdataspecificrows.head(2)
```

Out[2]:

In [4]:
```
plt.figure(figsize=(8,6))
plt.scatter(range(rawdataspecificrows.shape[0]), np.sort(rawdataspecific
rows.logerror.values))
plt.xlabel('index', fontsize=12)
plt.ylabel('logerror', fontsize=12)
plt.show()
```

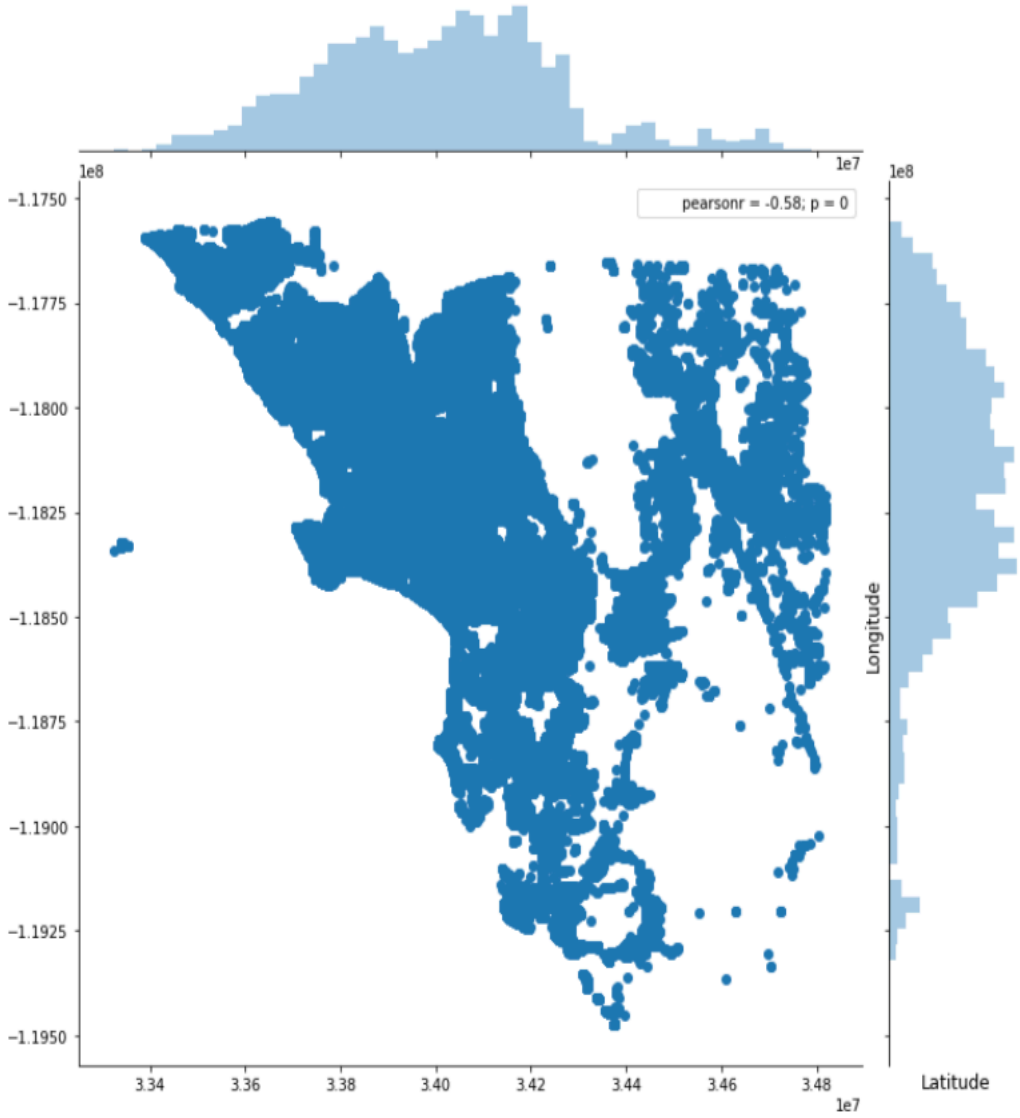


In [5]:
```
color = sns.color_palette()
rawdataspecificrows['transaction_month'] = rawdataspecificrows['transact
iondate']

cnt_srs = rawdataspecificrows['transaction_month'].value_counts()
plt.figure(figsize=(365,100))
sns.barplot(cnt_srs.index, cnt_srs.values, alpha=0.8, color=color[3])
plt.xticks(rotation='vertical')
plt.xlabel('Days of transaction', fontsize=12)
plt.ylabel('Number of Occurrences', fontsize=12)
plt.show()
```



In [26]:
```
plt.figure(figsize=(12,12))
sns.jointplot(x=prop_df.latitude.values, y=prop_df.longitude.values, siz
e=10)
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
```

```
<matplotlib.figure.Figure at 0x10b2f3198>
```



In [28]:
```
dtype_df.groupby("Column Type").aggregate('count').reset_index()
```

Out[28]:

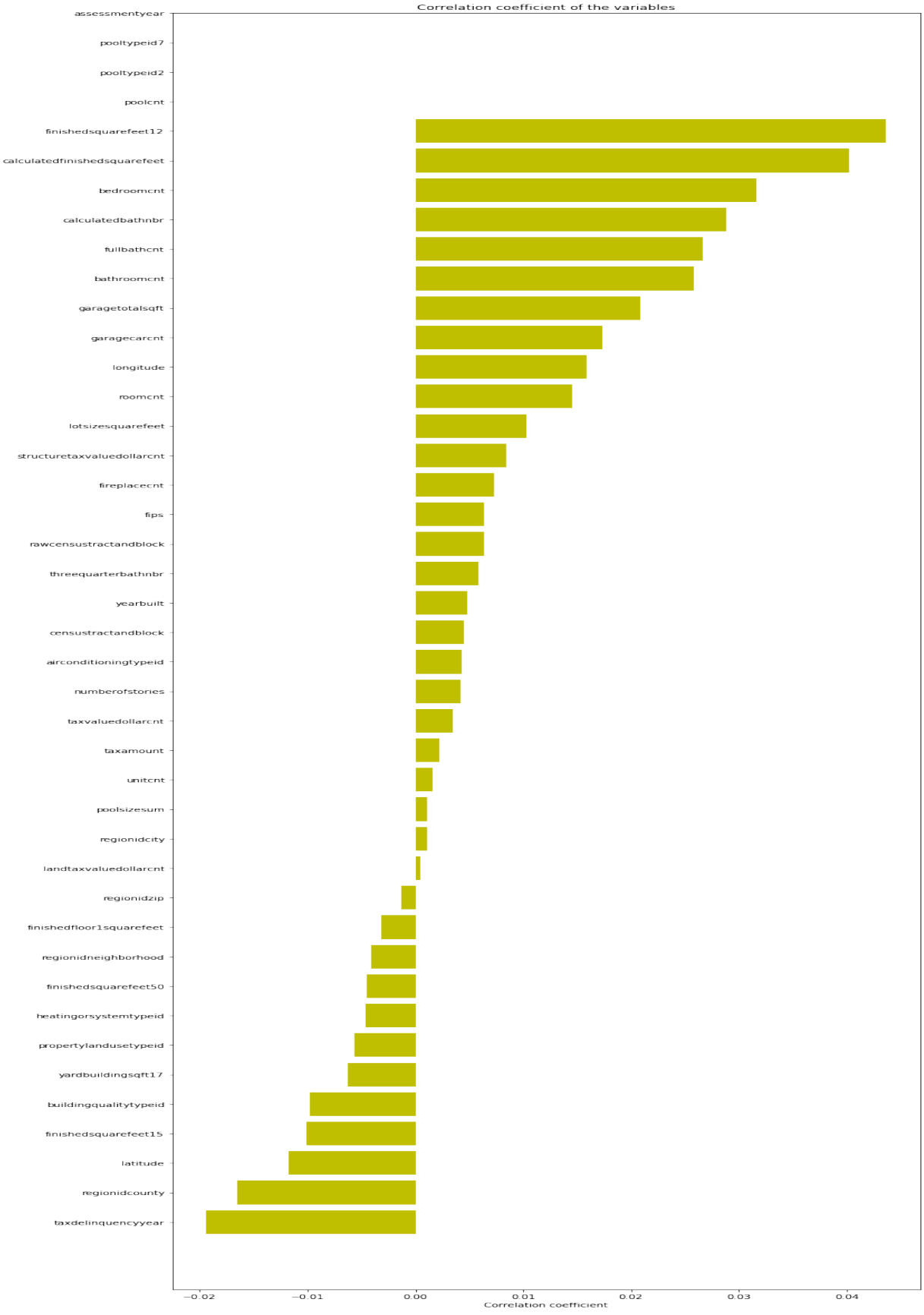
	Column Type	Count
0	int64	1
1	float64	53
2	object	7

In [29]:
```
missing_df = train_df.isnull().sum(axis=0).reset_index()
missing_df.columns = ['column_name', 'missing_count']
missing_df['missing_ratio'] = missing_df['missing_count'] / train_df.sha
pe[0]
missing_df.ix[missing_df['missing_ratio']>0.99]
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/ipykernel_launcher.py:4: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is
-deprecated
  after removing the cwd from sys.path.
```

Out[29]:

	column_name	missing_count	missing_ratio
5	architecturalstyletypeid	77406	0.997333
6	basementsqft	77563	0.999356
9	buildingclasstypeid	77598	0.999807
12	decktypeid	76999	0.992089
16	finishedsquarefeet13	77571	0.999459
19	finishedsquarefeet6	77227	0.995027
32	pooltypeid10	77148	0.994009
44	storytypeid	77563	0.999356
46	typeconstructiontypeid	77390	0.997127
49	yardbuildingsqft26	77543	0.999098
52	fireplaceflag	77441	0.997784


Correlation coefficient of the variables
assessmentyear
pooltypeid7
pooltypeid2
poolcnt
finishedsquarefeet12
calculatedfinishedsquarefeet
bedroomcnt
calculatedbathnbr
fullbathcnt
bathroomcnt
garagetotalsqft
garagecarcnt
longitude
roomcnt
lotsizesquarefeet
structuretaxvaluedollarcnt
fireplacecnt
fips
rawcensustractandblock
threequarterbathnbr
yearbuilt
censustractandblock
airconditioningtypeid
numberofstories
taxvaluedollarcnt
taxamount
unitcnt
poolsizesum
regionidcity
landtaxvaluedollarcnt
regionidzip
finishedfloor1squarefeet
regionidneighborhood
finishedsquarefeet50
heatingorsystemtypeid
propertylandusetypeid
yardbuildingsqft17
buildingqualitytypeid
finishedsquarefeet15
latitude
regionidcounty
taxdelinquencyyear
−0.02
−0.01
0.00
0.01
0.02
0.03
0.04
Correlation coefficient

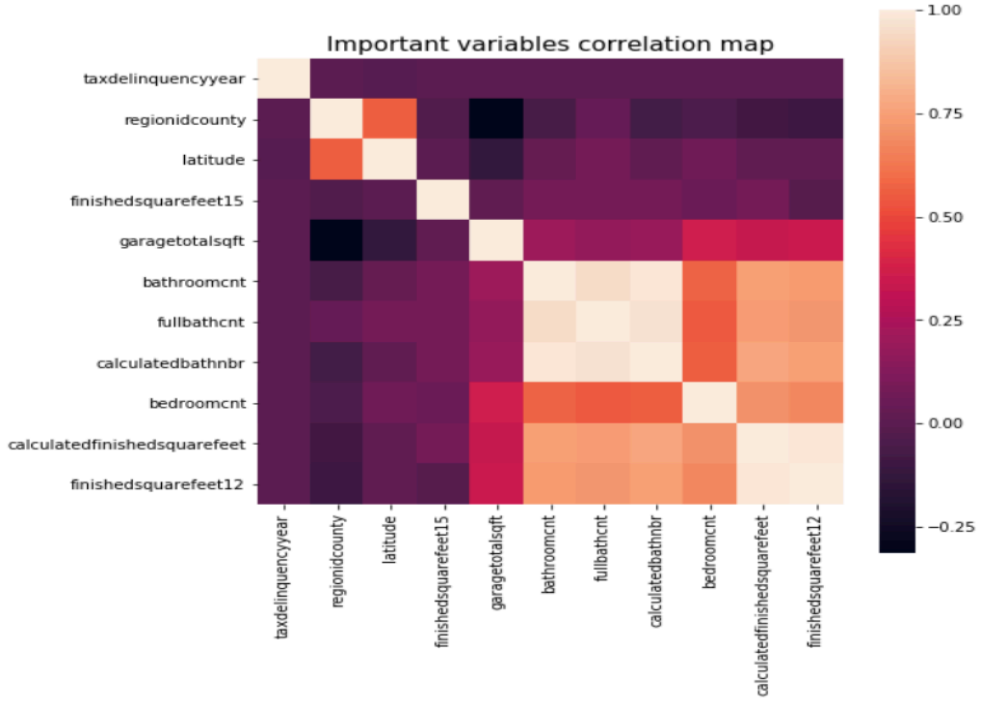
In [48]:
```
corr_df_sel = corr_df.ix[(corr_df['corr_values']>0.02) | (corr_df['corr_values'] < -0.01)]
corr_df_sel
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-deprecated
  """Entry point for launching an IPython kernel.
```

Out[48]:

	col_labels	corr_values
40	taxdelinquencyyear	-0.019361
26	regionidcounty	-0.016475
16	latitude	-0.011713
8	finishedsquarefeet15	-0.010099
14	garagetotalsqft	0.020888
1	bathroomcnt	0.025788
12	fullbathcnt	0.026632
4	calculatedbathnbr	0.028789
2	bedroomcnt	0.031603
6	calculatedfinishedsquarefeet	0.040233
7	finishedsquarefeet12	0.043625

In [49]:
```
cols_to_use = corr_df_sel.col_labels.tolist()

temp_df = train_df[cols_to_use]
corrmat = temp_df.corr(method='spearman')
f, ax = plt.subplots(figsize=(8, 8))

# Draw the heatmap using seaborn
sns.heatmap(corrmat, vmax=1., square=True)
plt.title("Important variables correlation map", fontsize=15)
plt.show()
```



In [50]:
```
train_df.to_csv('midterm_2017.csv',mode='a',encoding='utf-8',index=False
)
```

In [51]:
```
form2017= pd.read_csv("midterm_2017.csv")
form2016= pd.read_csv("midterm_2016.csv")
combine_data = pd.merge(form2016, form2017, on='parcelid', suffixes=('_2
016','_2017'))
combine_data.head()
```

Out[51]:

	parcelid	logerror_2016	airconditioningtypeid_2016	bathroomcnt_2016	bedroomcnt_
0	14366692	-0.1684	1.816372	3.5	4.0
1	10854446	0.3825	1.816372	2.0	2.0
2	11188425	-0.0747	1.000000	2.0	2.0
3	11624877	-0.0294	1.000000	2.0	2.0
4	13980837	-0.0253	1.816372	3.0	4.0

5 rows × 88 columns

In [52]:
```
combine_data.to_csv('combine2016^2017.csv',mode='a',encoding='utf-8',ind
ex=False)
```

In [107]:
```
#rawdata.to_csv('wrangleddata.csv', index=False)
import boto3
from botocore.client import Config
from boto.s3.connection import S3Connection

s3 = boto3.resource('s3')
for bucket in s3.buckets.all():
    print(bucket.name)

with open('s3.json') as data_file:
    data = json.load(data_file)
# secret keys

AWSAccess1=data["AWSAccess"]
AWSSecret1=data["AWSSecret"]

#Connection variables

c = boto.connect_s3(AWSAccess1, AWSSecret1)
conn = S3Connection(AWSAccess1, AWSSecret1)
bucket = c.get_bucket('zillowdata')
b = c.get_bucket(bucket, validate=False)


#
BUCKET_NAME ='zillowdata'
FILE_NAME='combine2016^2017.csv'
data = open(FILE_NAME, 'rb')

s3.Bucket(BUCKET_NAME).put_object(Key=FILE_NAME, Body=data,ACL='public-r
ead')


print('successfully uploaded to s3')
```

```
pleasepass
pleasepassokkk
zhaomengqi8000
zillowdata
successfully uploaded to s3
```

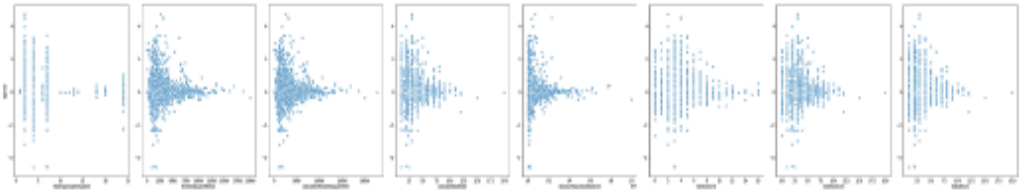
In [91]:
```
rawdataspecificrows= pd.read_csv("midterm_2016.csv")
rawdataspecificrows.head(5)
```

Out[91]:

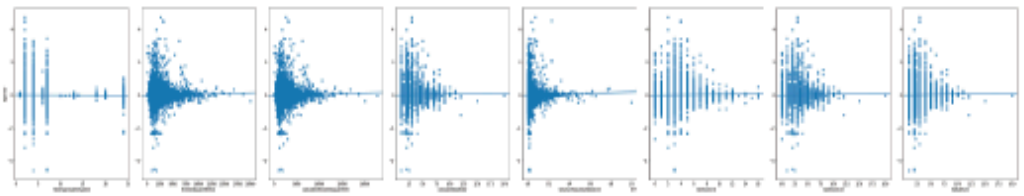
	parcelid	logerror	airconditioningtypeid	bathroomcnt	bedroomcnt	buildingqualityty
0	11016594	0.0276	1.000000	2.0	3.0	4.000000
1	14366692	-0.1684	1.816372	3.5	4.0	5.565407
2	12098116	-0.0040	1.000000	3.0	2.0	4.000000
3	12643413	0.0218	1.000000	2.0	2.0	4.000000
4	14432541	-0.0050	1.816372	2.5	4.0	5.565407

5 rows x 45 columns

In [41]:
```
# visualize the relationship between the features and the response using
 scatterplots
sns.pairplot(rawdataspecificrows, x_vars=['heatingorsystemtypeid','finis
hedsquarefeet12','calculatedfinishedsquarefeet','calculatedbathnbr','str
ucturetaxvaluedollarcnt','bedroomcnt','bathroomcnt','fullbathcnt'], y_va
rs='logerror', size=7, aspect=0.8)
plt.show()
```



In [38]:
```
# visualize the relationship between the features and the response using
 scatterplots
sns.pairplot(rawdataspecificrows, x_vars=['heatingorsystemtypeid','finis
hedsquarefeet12','calculatedfinishedsquarefeet','calculatedbathnbr','str
ucturetaxvaluedollarcnt','bedroomcnt','bathroomcnt','fullbathcnt'], y_va
rs='logerror', size=7, aspect=0.8, kind='reg')
plt.show()
```



In [42]:
```
feature_cols=['heatingorsystemtypeid','finishedsquarefeet12','calculated
finishedsquarefeet','calculatedbathnbr','structuretaxvaluedollarcnt','be
droomcnt','bathroomcnt','fullbathcnt']
X = rawdataspecificrows[feature_cols]
X.head(5)
```

Out[42]:

	heatingorsystemtypeid	finishedsquarefeet12	calculatedfinishedsquarefeet	calculatedl
0	2.000000	1684.0	1684.0	2.0
1	3.926979	2263.0	2263.0	3.5
2	2.000000	2217.0	2217.0	3.0
3	2.000000	839.0	839.0	2.0
4	3.926979	2283.0	2283.0	2.5

split the data into train and test dataset

In [43]:
```
Y = rawdataspecificrows.logerror
print(Y.head(5))

from sklearn.cross_validation import train_test_split
X_train,X_test, Y_train, Y_test = train_test_split(X, Y, random_state=1)

print (X_train.shape)
print (Y_train.shape)
print (X_test.shape)
print (Y_test.shape)
```

```
0    0.0276
1   -0.1684
2   -0.0040
3    0.0218
4   -0.0050
Name: logerror, dtype: float64
(67706, 8)
(67706,)
(22569, 8)
(22569,)

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/sklearn/cross_validation.py:41: DeprecationWarning: This module
was deprecated in version 0.18 in favor of the model_selection module i
nto which all the refactored classes and functions are moved. Also note
that the interface of the new CV iterators are different from that of t
his module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [44]:
```
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
model=linreg.fit(X_train, Y_train)
print(model)
print(linreg.coef_)
print(linreg.intercept_)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=F
alse)
[ -6.39747093e-04   2.67350438e-06   5.14741221e-06  -3.37058489e-03
   1.58712522e-10   5.47664113e-04   2.33778519e-04   1.64990211e-03]
0.00273201629783

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pa
ckages/scipy/linalg/basic.py:1018: RuntimeWarning: internal gelsd drive
r lwork query error, required iwork dimension not returned. This is lik
ely the result of LAPACK bug 0038, fixed in LAPACK 3.2.2 (released July
21, 2010). Falling back to 'gelss' driver.
  warnings.warn(mesg, RuntimeWarning)
```

In [45]:
```
zip(feature_cols,linreg.coef_)
```

Out[45]:
```
<zip at 0x10dfaa3c8>
```

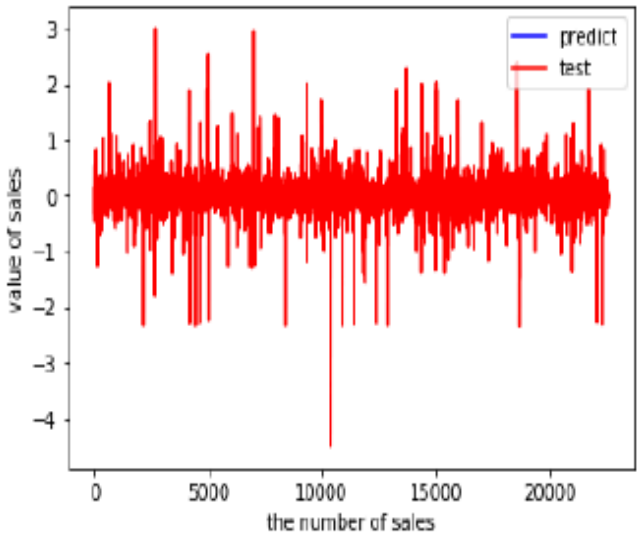
In [46]:
```
Y_pred = linreg.predict(X_test)
print(Y_pred)
print (type(Y_pred))
```

```
[ 0.02001145  0.01799294  0.02732893 ...,  0.0094094   0.00834821
  0.01109462]
<class 'numpy.ndarray'>
```

Get RMSE and MAE value

In [47]:
```
# use RMES
print (type(Y_pred),type(Y_test))
print (len(Y_pred),len(Y_test))
print (Y_pred.shape,Y_test.shape)
from sklearn import metrics
import numpy as np
sum_mean=0
for i in range(len(Y_pred)):
    sum_mean+=(Y_pred[i]-Y_test.values[i])**2
sum_erro=np.sqrt(sum_mean/50)
# calculate RMSE by hand
print("RMSE by hand:",sum_erro)

# make ROC graph
import matplotlib.pyplot as plt
plt.figure()
plt.plot(range(len(Y_pred)),Y_pred,'b',label="predict")
plt.plot(range(len(Y_pred)),Y_test,'r',label="test")
plt.legend(loc="upper right") #显示图中的标签
plt.xlabel("the number of sales")
plt.ylabel('value of sales')
plt.show()
```

```
<class 'numpy.ndarray'> <class 'pandas.core.series.Series'>
22569 22569
(22569,) (22569,)
RMSE by hand: 3.2353815736
```



In [48]:
```
#R-squared score of this model
train_pred=linreg.predict(X_train)
from sklearn.metrics import*
r2_score(Y_train,train_pred)
test_pred=linreg.predict(X_test)
#Mean absolute percentage error (MAPE)
print(mean_absolute_error(Y_test,test_pred)*100)
#Mean squared error
print(mean_squared_error(Y_test,test_pred))
#Median absolute error
print(median_absolute_error(Y_test,test_pred))
```

```
6.67315435027
0.0231904247569
0.0320428477077
```

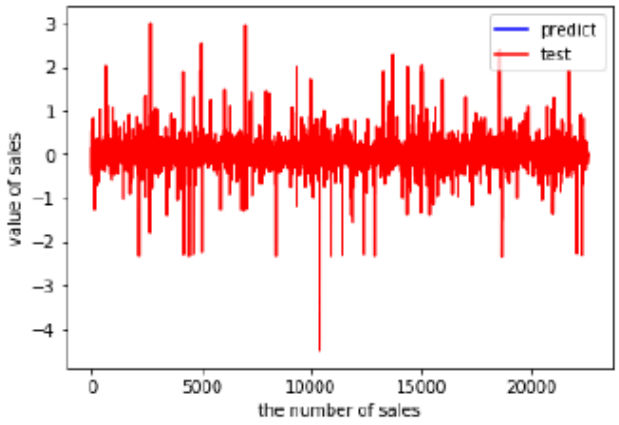
1.cancel heatingorsystemtypeid

In [50]:
```
feature_cols=['finishedsquarefeet12','calculatedfinishedsquarefeet','cal
culatedbathnbr','structuretaxvaluedollarcnt','bedroomcnt','bathroomcnt',
'fullbathcnt']
X = rawdataspecificrows[feature_cols]
X.head(5)
```

Out[50]:

	finishedsquarefeet12	calculatedfinishedsquarefeet	calculatedbathnbr	structuretaxval
0	1684.0	1684.0	2.0	122754.0
1	2263.0	2263.0	3.5	346458.0
2	2217.0	2217.0	3.0	61994.0
3	839.0	839.0	2.0	171518.0
4	2283.0	2283.0	2.5	169574.0

```
<class 'numpy.ndarray'> <class 'pandas.core.series.Series'>
22569 22569
(22569,) (22569,)
RMSE by hand: 3.23534651548
```




```
6.67339787519
0.0231899221836
0.0320751677121
```

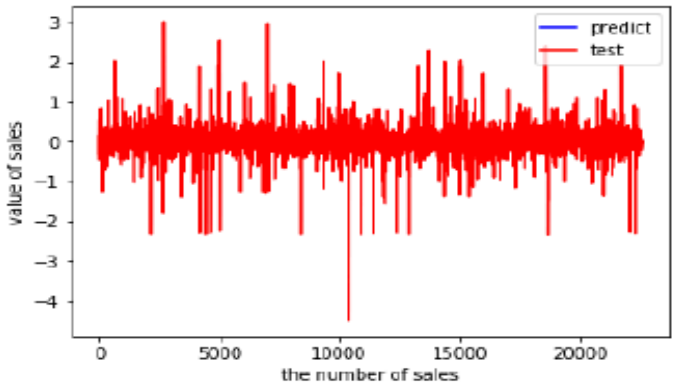
In [ ]:
```
#2.继续取消finishedsquarefeet12
feature_cols=['heatingorsystemtypeid','calculatedfinishedsquarefeet','ca
lculatedbathnbr','structuretaxvaluedollarcnt','bedroomcnt','bathroomcnt'
,'fullbathcnt']
X = rawdataspecificrows[feature_cols]
X.head(5)
```

In [56]:
```
Y = rawdataspecificrows.logerror
print(Y.head(5))

from sklearn.cross_validation import train_test_split
X_train,X_test, Y_train, Y_test = train_test_split(X, Y, random_state=1)

print (X_train.shape)
print (Y_train.shape)
print (X_test.shape)
print (Y_test.shape)
```

```
0    0.0276
1   -0.1684
2   -0.0040
3    0.0218
4   -0.0050
Name: logerror, dtype: float64
(67706, 7)
(67706,)
(22569, 7)
(22569,)
```

In [78]:
```
#结论:需要取消calculatedfinishedsquarefeet, calculatedbathnbr, structuretax
valuedollarcnt, bedroomcnt, bathroomcnt, fullbathcnt
feature_cols=['heatingorsystemtypeid','finishedsquarefeet12']
X = rawdataspecificrows[feature_cols]
X.head(5)
Y = rawdataspecificrows.logerror
print(Y.head(5))

from sklearn.cross_validation import train_test_split
X_train,X_test, Y_train, Y_test = train_test_split(X, Y, random_state=1)

print (X_train.shape)
print (Y_train.shape)
```

```
0    0.0276
1   -0.1684
2   -0.0040
3    0.0218
4   -0.0050
Name: logerror, dtype: float64
(67706, 2)
(67706,)
(22569, 2)
(22569,)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=F
alse)
[ -5.99574675e-04   6.97838732e-06]
0.0023225067142
[ 0.02097687  0.01836695  0.02620368 ...,  0.00924205  0.00978658
  0.0108931 ]
<class 'numpy.ndarray'>
<class 'numpy.ndarray'> <class 'pandas.core.series.Series'>
22569 22569
(22569,) (22569,)
RMSE by hand: 3.23503083134
```




```
6.67192752436
0.0231853969598
0.032009358968
```

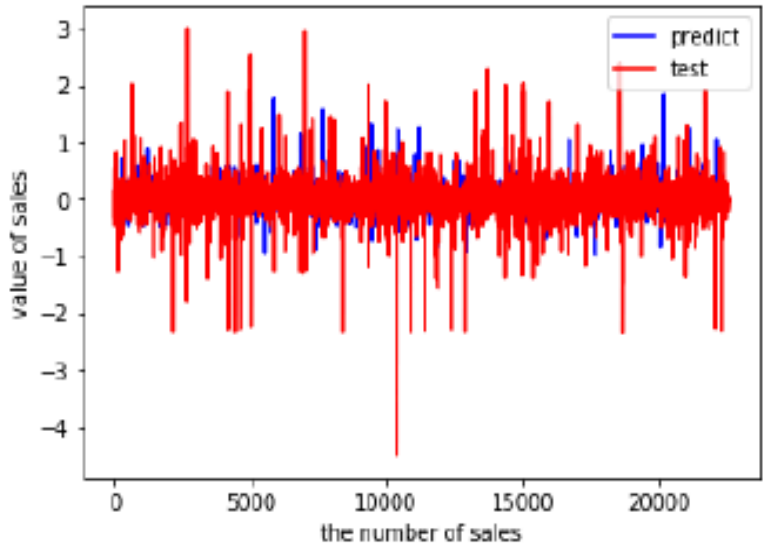
In [67]:
```
#Use Random Forest

#test 8 features

import pandas
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import cross_val_score

df = pd.read_csv('midterm_2016.csv')
df.index = df['parcelid'].tolist()

column=['heatingorsystemtypeid','finishedsquarefeet12','calculatedfinish
```



```
edsquarefeet','calculatedbathnbr','structuretaxvaluedollarcnt','bedroomc
nt','bathroomcnt','fullbathcnt']
X = df[column]
#X =array[:,1]
Y = df['logerror']
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,  random_state=
1)


#print (Y_test.head(2))
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor()
model=regressor.fit(X_train, Y_train)
print(model)
from sklearn.metrics import accuracy_score
Y_pred = regressor.predict(X_test)
#print(accuracy_score(Y_test, Y_pred))


# make ROC graph
import matplotlib.pyplot as plt
plt.figure()
plt.plot(range(len(Y_pred)),Y_pred,'b',label="predict")
plt.plot(range(len(Y_pred)),Y_test,'r',label="test")
plt.legend(loc="upper right") #显示图中的标签
plt.xlabel("the number of sales")
plt.ylabel('value of sales')
plt.show()
```

```
#R-squared score of this model
train_pred=regressor.predict(X_train)
from sklearn.metrics import*
r2_score(Y_train,train_pred)
#Mean absolute percentage error (MAPE)
print(mean_absolute_error(Y_test,Y_pred)*100)
#Mean squared error
print(mean_squared_error(Y_test,Y_pred))
#Median absolute error
print(median_absolute_error(Y_test,Y_pred))
```

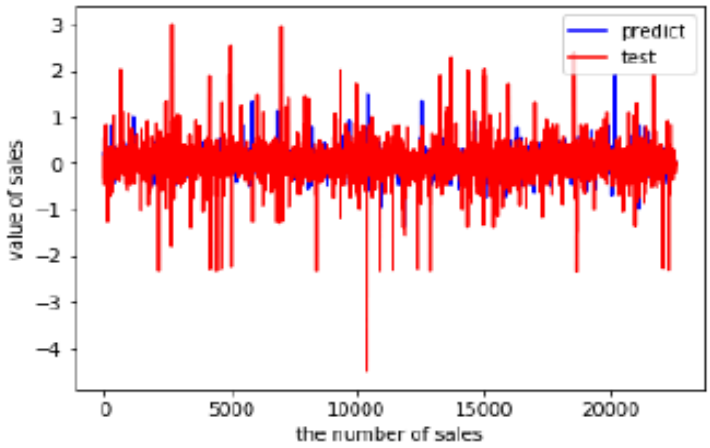
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=Fa
lse)
```




```
RMSE by hand: 3.60992771638
8.55672317984
0.0288705262029
0.04637
```

In [68]:
```
#Use Random Forest

#1.取消heatingorsystemtypeid

import pandas
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import cross_val_score

df = pd.read_csv('midterm_2016.csv')
df.index = df['parcelid'].tolist()

column=['finishedsquarefeet12','calculatedfinishedsquarefeet','calculate
dbathnbr','structuretaxvaluedollarcnt','bedroomcnt','bathroomcnt','fullb
athcnt']
X = df[column]
#X =array[:,1]
Y = df['logerror']
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,  random_state=
1)
```

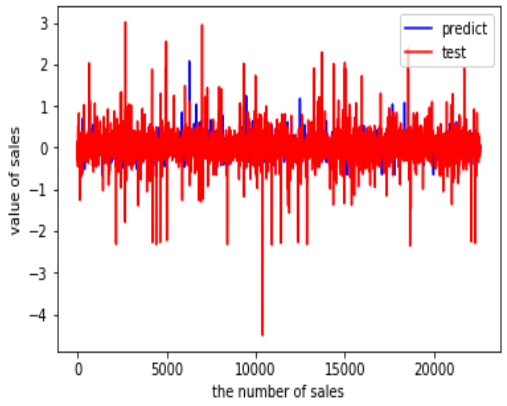
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=Fa
lse)
```




```
RMSE by hand: 3.59167719431
8.62035200474
0.0285793457134
0.04704
```

In [76]:
```
#结论:需要取消calculatedfinishedsquarefeet, calculatedbathnbr, structuretaxvaluedollarcnt, be
#Use Random Forest


import pandas
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import cross_val_score

df = pd.read_csv('midterm_2016.csv')
df.index = df['parcelid'].tolist()

column=['heatingorsystemtypeid','finishedsquarefeet12']
X = df[column]
#X =array[:,1]
Y = df['logerror']
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,  random_state=1)


#print (Y_test.head(2))
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor()
model=regressor.fit(X_train, Y_train)
print(model)
from sklearn.metrics import accuracy_score
Y_pred = regressor.predict(X_test)
#print(accuracy_score(Y_test, Y_pred))
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```




```
RMSE by hand: 3.44668314442
7.95736097138
0.0263184560636
0.0423624671717
```

In [77]:
```
# Use Neural networks
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from sklearn import preprocessing

# load pima indians dataset
df =  pd.read_csv("midterm_2016.csv")
# split into input (X) and output (Y) variables
column=['heatingorsystemtypeid','finishedsquarefeet12','calculatedfinishedsquarefeet','cal
X = np.array(df[column])
Y = np.array(df[['logerror']])

neural_net = MLPRegressor(
    activation='logistic',
    learning_rate_init=0.001,
    solver='sgd',
    learning_rate='invscaling',
    hidden_layer_sizes=(200,),
    verbose=True,
    max_iter=2000,
    tol=1e-6
)

# Scaling the data
max_min_scaler = preprocessing.MinMaxScaler()
X_scaled = max_min_scaler.fit_transform(X)
Y_scaled = max_min_scaler.fit_transform(Y)

neural_net.fit(X_scaled[0:4001], Y_scaled[0:4001].ravel())

predicted = neural_net.predict(X_scaled[5001:5051])

# Scale back to actual scale
max_min_scaler = preprocessing.MinMaxScaler(feature_range=(Y[5001:5051].min(), Y[5001:5051
predicted_scaled = max_min_scaler.fit_transform(predicted.reshape(-1, 1))

print("Root Mean Square Error ", mean_squared_error(Y[5001:5051], predicted_scaled))
```

```
Iteration 1, loss = 0.04041079
Iteration 2, loss = 0.01019061
Iteration 3, loss = 0.02102910
Iteration 4, loss = 0.01856962
Iteration 5, loss = 0.01536955
Training loss did not improve more than tol=0.000001 for two consecutive epochs. Stoppin
g.
Root Mean Square Error  0.212767050844
```

The same with 2017.

#import h2o and run

In [2]:
```
import h2o
h2o.init()
```

```
Checking whether there is an H2O instance running at http://localhost:54321..... not fou
nd.
Attempting to start a local H2O server...
  Java Version: java version "1.8.0_121"; Java(TM) SE Runtime Environment (build 1.8.0_1
21-b13); Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
  Starting server from /Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/s
ite-packages/h2o/backend/bin/h2o.jar
  Ice root: /var/folders/s0/d7szw5jj2890t7dmcj8gl9w40000gn/T/tmpc2lxms1m
  JVM stdout: /var/folders/s0/d7szw5jj2890t7dmcj8gl9w40000gn/T/tmpc2lxms1m/h2o_zhaomengq
i_started_from_python.out
  JVM stderr: /var/folders/s0/d7szw5jj2890t7dmcj8gl9w40000gn/T/tmpc2lxms1m/h2o_zhaomengq
i_started_from_python.err
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321... successful.
```

H2O cluster uptime:	03 secs
H2O cluster version:	3.14.0.7
H2O cluster version age:	14 days, 18 hours and 38 minutes
H2O cluster name:	H2O_from_python_zhaomengqi_6xausm
H2O cluster total nodes:	1
H2O cluster free memory:	1.778 Gb
H2O cluster total cores:	4
H2O cluster allowed cores:	4
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:54321
H2O connection proxy:	None
H2O internal security:	False
H2O API Extensions:	XGBoost, Algos, AutoML, Core V3, Core V4
Python version:	3.6.2 final

#Convert Pandas data frame into H2o frame

In [3]:
```
import pandas as pd

df = pd.read_csv('midterm_2016.csv')

df.drop(['parcelid','airconditioningtypeid'],axis=1,inplace=True)
df.drop(['buildingqualitytypeid','finishedfloor1squarefeet','finishedsquarefeet15','finishe

wine=h2o.H2OFrame(df)

wine.head(5)
```

Parse progress: |██| 100%

logerror	bathroomcnt	bedroomcnt	calculatedbathnbr	calculatedfinishedsquarefeet	finishedsquarefeet12
0.0276	2	3	2	1684	1684
-0.1684	3.5	4	3.5	2263	2263
-0.004	3	2	3	2217	2217
0.0218	2	2	2	839	839
-0.005	2.5	4	2.5	2283	2283

Out[3]:

In [4]:
```
feature = list(wine.columns)
feature.remove('logerror')
feature
```

Out[4]:
```
['bathroomcnt',
 'bedroomcnt',
 'calculatedbathnbr',
 'calculatedfinishedsquarefeet',
 'finishedsquarefeet12',
 'fullbathcnt',
 'heatingorsystemtypeid',
 'structuretaxvaluedollarcnt']
```

In [5]:
```
wine_split=wine.split_frame(ratios=[0.75])
wine_train=wine_split[0]
wine_test=wine_split[1]
```

In [6]:
```
wine_train.shape
```

Out[6]: (67825, 9)

In [7]:
```
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
glm_default=H2OGeneralizedLinearEstimator(family='gaussian',model_id='glm_default')
```

In [8]:
```
glm_default.train(x=feature,y='logerror',training_frame=wine_train)
```

```
glm Model Build progress: |████████████████████████████████████████████████| 100%
```

In [9]:
```
glm_default
```

```
Model Details
=============
H2OGeneralizedLinearEstimator :  Generalized Linear Modeling
Model Key:  glm_default


ModelMetricsRegressionGLM: glm
** Reported on train data. **

MSE: 0.026261115373876945
RMSE: 0.16205281661815368
MAE: 0.06837790575463598
RMSLE: NaN
R^2: 0.0016267425290013016
Mean Residual Deviance: 0.026261115373876945
Null degrees of freedom: 67824
Residual degrees of freedom: 67816
Null deviance: 1784.0623603490517
Residual deviance: 1781.1601502332037
AIC: -54361.33013131289
Scoring History:
```

timestamp	duration	iterations	negative_log_likelihood
2017-11-04 13:16:20	0.000 sec	0	1784.0623603490517

Out[9]:

In [105]:
```
yhat_test_glm =glm_default.predict(wine_test)
yhat_test_glm
```

```
glm prediction progress: |████████████████████████████████████████████████| 100%
```

predict
0.0134297
0.0131736
0.00502152
0.0167455
0.015629
0.00772776
0.00880944
0.00505806

Out[105]:

In [10]:
```
from h2o.estimators.random_forest import H2ORandomForestEstimator
drf_default=H2ORandomForestEstimator(seed=1234,model_id='drf_default')
drf_default.train(x=feature,y='logerror',training_frame=wine_train)
drf_default
```

```
drf Model Build progress: |████████████████████████████████████████████████| 100%
Model Details
=============
H2ORandomForestEstimator :  Distributed Random Forest
Model Key:  drf_default


ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.027277447300287215
RMSE: 0.16515885474381087
MAE: 0.07174529551321478
RMSLE: NaN
Mean Residual Deviance: 0.027277447300287215
Scoring History:
```

timestamp	duration	number_of_trees	training_rmse	training_mae	training_deviance
2017-11-04 13:16:29	0.168 sec	0.0	nan	nan	nan
2017-11-04 13:16:29	0.769 sec	1.0	0.17444630780710	0.070304315	
2017-11-04 13:16:29	1.023 sec	2.0	0.17984100785559	0.070323428	
2017-11-04 13:16:30	1.268 sec	3.0	0.17872080777237	0.070319436	
2017-11-04 13:16:30	1.501 sec	4.0	0.17762030771563	0.070315493	
---	---	---	---	---	---
2017-11-04 13:16:32	3.739 sec	20.0	0.16737550730828	0.070280146	
2017-11-04 13:16:32	3.826 sec	21.0	0.16699610729770	0.070278877	
2017-11-04 13:16:32	4.038 sec	22.0	0.16699960729852	0.070278889	

Out[10]:

In [11]:
```
drf_test_glm =drf_default.predict(wine_test)
drf_test_glm
```

```
drf prediction progress: |████████████████████████████████████████████████| 100%
```

predict
0.0142056
0.00969776
0.0279526
0.00725515
0.0121618
0.027
0.0064791
0.000176644
0.0105517
0.00452557

Out[11]:

In [12]:
```
feature = list(wine.columns)
#feature.remove('logerror')
feature.remove('calculatedfinishedsquarefeet')
feature.remove('calculatedbathnbr')
feature.remove('structuretaxvaluedollarcnt')
feature.remove('bedroomcnt')
feature.remove('bathroomcnt')
feature.remove('fullbathcnt')

feature
```

Out[12]:
```
['logerror', 'finishedsquarefeet12', 'heatingorsystemtypeid']
```

In [13]:
```
wine_split=wine.split_frame(ratios=[0.75])
wine_train=wine_split[0]
wine_test=wine_split[1]
```

In [14]:
```
wine_train.shape
```

Out[14]:
```
(67618, 9)
```

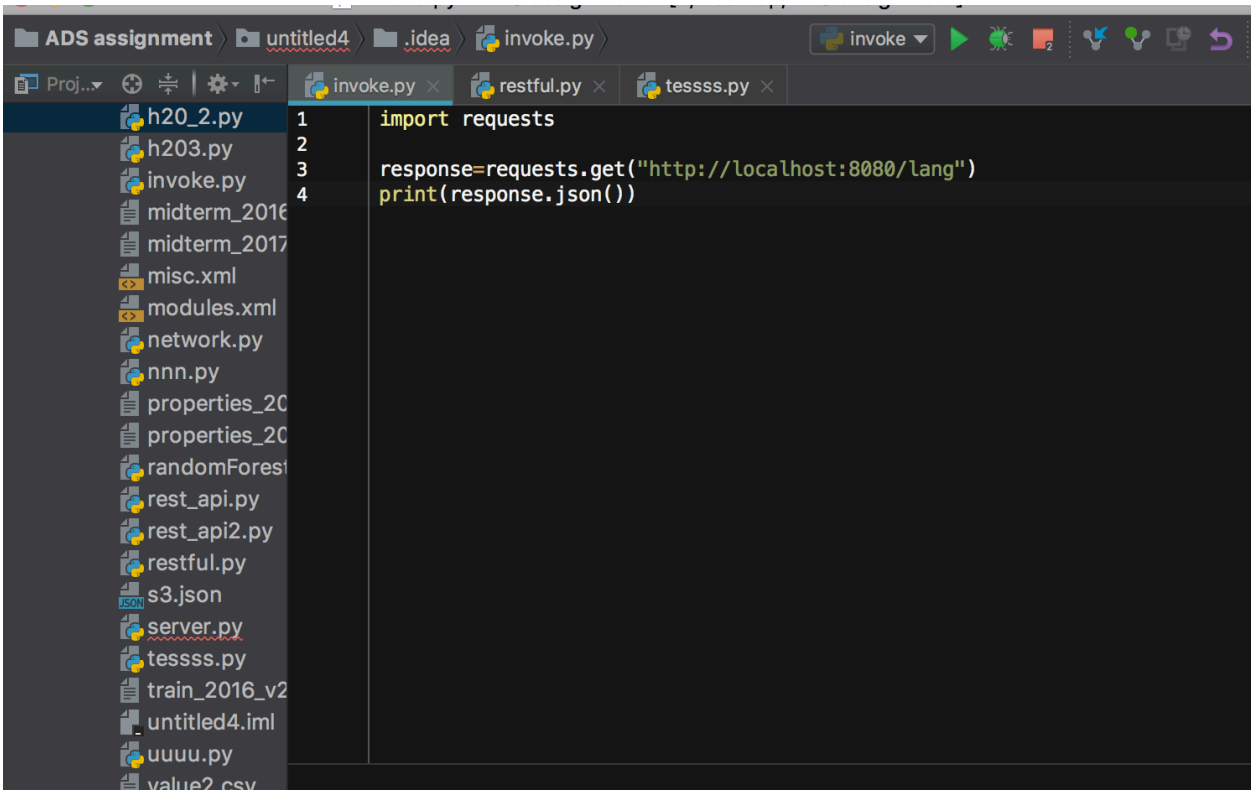
In [15]:
```
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
glm_default=H2OGeneralizedLinearEstimator(family='gaussian',model_id='glm_default')
```

In [16]:
```
glm_default.train(x=feature,y='logerror',training_frame=wine_train)
```

```
glm Model Build progress: |████████████████████████████████████████████████| 100%
```

```
In [17]: from h2o.estimators.random_forest import H2ORandomForestEstimator
         drf_default=H2ORandomForestEstimator(seed=1234,model_id='drf_default')
         drf_default.train(x=feature,y='logerror',training_frame=wine_train)
         drf_default

drf Model Build progress: |████████████████████████████████████████████████| 100%
Model Details
=============
H2ORandomForestEstimator :  Distributed Random Forest
Model Key:  drf_default


ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.02624302485907395
RMSE: 0.1619969902778257
MAE: 0.06841834701650601
RMSLE: NaN
Mean Residual Deviance: 0.02624302485907395
Scoring History:
```

Json API



Invoke Rest API

```
from boto.s3.connection import S3Connection
import os
import json
import boto.s3
import sys
import datetime
import seaborn as sns
from boto.s3.key import Key
from pprint import pprint
import pandas as pd
import urllib
import csv
import io
import requests
import time
import json
import datetime
from pprint import pprint
import scipy
import numpy as np
import matplotlib.pyplot as plt

import pandas
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
from sklearn.model_selection import cross_val_score
import boto3
from botocore.client import Config
from boto.s3.connection import S3Connection
from sklearn.metrics import *
from h2o.estimators.random_forest import H2ORandomForestEstimator
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error
from sklearn import preprocessing
import h2o
h2o.init()
import pandas as pd
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
import h2o
h2o.init()
import pandas as pd
from h2o.estimators.random_forest import H2ORandomForestEstimator
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
```

```
#create url
from flask import Flask
from flask import jsonify
from flask import request


#Rest API
app =Flask(__name__)
languages=[{'name':'JavaScript'},{'name':'python'},{'name':'Ruby'}]
@app.route('/lang',methods=['GET'])

def returnAll():
    # 2016数据清理和合并

    # 清理和填充数据

    # 1.导入数据
    rawdataspecificrows = pd.read_csv("train_2016_v2.csv")
    prop_df = pd.read_csv("properties_2016.csv")

    # 2.合并表格
    train_df = pd.merge(rawdataspecificrows, prop_df, on='parcelid', how='left')
    print(train_df.shape)

    # 3.分析数据train和properties_2016

print((rawdataspecificrows['parcelid'].value_counts().reset_index())['parcelid'].value
_counts())
    print(prop_df.isnull().sum())

    pd.options.display.max_rows = 65
    dtype_df = train_df.dtypes.reset_index()
    dtype_df.columns = ["Count", "Column Type"]
    print(dtype_df)

    missing_df = train_df.isnull().sum(axis=0).reset_index()
    missing_df.columns = ['column_name', 'missing_count']
    missing_df['missing_ratio'] = missing_df['missing_count'] / train_df.shape[0]
    print(missing_df.ix[missing_df['missing_ratio'] > 0.99])

    # 4.数据清理drop column of missing>0.99 and column which is not folat
    #
train_df.drop(['transactiondate','hashottuborspa','hashottuborspa','propertycountyland
usecode','propertyzoningdesc','fireplaceflag','taxdelinquencyflag'],axis=1,inplace=Tru
e)
    #
train_df.drop(['architecturalstyletypeid','basementsqft','buildingclasstypeid','deckty
peid','finishedsquarefeet13','finishedsquarefeet6','storytypeid','typeconstructiontype
id','yardbuildingsqft26','fireplaceflag'],axis=1,inplace=True)
    train_df = pd.read_csv("values.csv")
    print(train_df.shape)

    # 5数据转化
    # train_df.to_csv('midterm_2016.csv',mode='a',encoding='utf-8',index=False)

    # 2017数据清理和合并
    train_df = pd.read_csv("value2.csv")
```

```
    print(train_df.shape)

    # 2016和2017表格合并
    form2017 = pd.read_csv("midterm_2017.csv")
    form2016 = pd.read_csv("midterm_2016.csv")
    combine_data = pd.merge(form2016, form2017, on='parcelid', suffixes=('_2016',
'_2017'))
    train_df.to_csv('conbinedata.csv', mode='a', encoding='utf-8', index=False)

    # upload into S3
    # rawdata.to_csv('wrangleddata.csv', index=False)
    # rawdata.to_csv('wrangleddata.csv', index=False)
    # rawdata.to_csv('wrangleddata.csv', index=False)


    s3 = boto3.resource('s3')
    for bucket in s3.buckets.all():
        print(bucket.name)

    with open('s3.json') as data_file:
        data = json.load(data_file)
    # secret keys

    AWSAccess1 = data["AWSAccess"]
    AWSSecret1 = data["AWSSecret"]

    # Connection variables

    c = boto.connect_s3(AWSAccess1, AWSSecret1)
    conn = S3Connection(AWSAccess1, AWSSecret1)
    bucket = c.get_bucket('zillowdata')
    b = c.get_bucket(bucket, validate=False)

    #
    BUCKET_NAME = 'zillowdata'
    FILE_NAME = 'midterm_2017.csv'
    data = open(FILE_NAME, 'rb')

    s3.Bucket(BUCKET_NAME).put_object(Key=FILE_NAME, Body=data, ACL='public-read')

    print('successfully uploaded to s3')

    rawdataspecificrows = pd.read_csv("midterm_2016.csv")
    #
feature_cols=['heatingorsystemtypeid','finishedsquarefeet12','calculatedfinishedsquare
feet','calculatedbathnbr','structuretaxvaluedollarcnt','bedroomcnt','bathroomcnt','ful
lbathcnt']
    feature_cols = ['heatingorsystemtypeid', 'finishedsquarefeet12']
    X = rawdataspecificrows[feature_cols]
    Y = rawdataspecificrows.logerror

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=1)

    linreg = LinearRegression()
    model = linreg.fit(X_train, Y_train)
    Y_pred = linreg.predict(X_test)

    sum_mean = 0
    for i in range(len(Y_pred)):
        sum_mean += (Y_pred[i] - Y_test.values[i]) ** 2
    sum_erro = np.sqrt(sum_mean / 50)
```

```
    # calculate RMSE by hand
    print("RMSE by hand:", sum_erro)
    # R-squared score of this model
    train_pred = linreg.predict(X_train)

    r2_score(Y_train, train_pred)
    test_pred = linreg.predict(X_test)
    # Mean absolute percentage error (MAPE)
    print(mean_absolute_error(Y_test, test_pred) * 100)
    # Mean squared error
    print(mean_squared_error(Y_test, test_pred))
    # Median absolute error
    print(median_absolute_error(Y_test, test_pred))

    # TREE
    # Use Random Forest

    # test 8 features



    df = pd.read_csv('midterm_2016.csv')
    df.index = df['parcelid'].tolist()
    #
feature_cols=['heatingorsystemtypeid','finishedsquarefeet12','calculatedfinishedsquare
feet','calculatedbathnbr','structuretaxvaluedollarcnt','bedroomcnt','bathroomcnt','ful
lbathcnt']
    column = ['heatingorsystemtypeid', 'finishedsquarefeet12']
    X = df[column]
    # X =array[:,1]
    Y = df['logerror']

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=1)

    # print (Y_test.head(2))

    regressor = RandomForestRegressor()
    model = regressor.fit(X_train, Y_train)
    print(model)

    Y_pred = regressor.predict(X_test)
    # print(accuracy_score(Y_test, Y_pred))


    # make ROC graph


    # use RMES

    sum_mean = 0
    for i in range(len(Y_pred)):
        sum_mean += (Y_pred[i] - Y_test.values[i]) ** 2
    sum_erro = np.sqrt(sum_mean / 50)
    # calculate RMSE by hand
    print("RMSE by hand:", sum_erro)

    # R-squared score of this model
    train_pred = regressor.predict(X_train)

    r2_score(Y_train, train_pred)
    # Mean absolute percentage error (MAPE)
```

```
    print(mean_absolute_error(Y_test, Y_pred) * 100)
    # Mean squared error
    print(mean_squared_error(Y_test, Y_pred))
    # Median absolute error
    print(median_absolute_error(Y_test, Y_pred))

    # Neural networks
    # Use Neural networks


    # load pima indians dataset
    df = pd.read_csv("midterm_2016.csv")
    # split into input (X) and output (Y) variables
    column = ['heatingorsystemtypeid', 'finishedsquarefeet12',
'calculatedfinishedsquarefeet', 'calculatedbathnbr',
              'structuretaxvaluedollarcnt', 'bedroomcnt', 'bathroomcnt',
'fullbathcnt']
    X = np.array(df[column])
    Y = np.array(df[['logerror']])

    neural_net = MLPRegressor(
        activation='logistic',
        learning_rate_init=0.001,
        solver='sgd',
        learning_rate='invscaling',
        hidden_layer_sizes=(200,),
        verbose=True,
        max_iter=2000,
        tol=1e-6
    )

    # Scaling the data
    max_min_scaler = preprocessing.MinMaxScaler()
    X_scaled = max_min_scaler.fit_transform(X)
    Y_scaled = max_min_scaler.fit_transform(Y)

    neural_net.fit(X_scaled[0:4001], Y_scaled[0:4001].ravel())

    predicted = neural_net.predict(X_scaled[5001:5051])

    # Scale back to actual scale
    max_min_scaler = preprocessing.MinMaxScaler(feature_range=(Y[5001:5051].min(),
Y[5001:5051].max()))
    predicted_scaled = max_min_scaler.fit_transform(predicted.reshape(-1, 1))

    print("Root Mean Square Error ", mean_squared_error(Y[5001:5051],
predicted_scaled))

    # part3
    # 使用h2o重做的2016年linear分析和结论


    df = pd.read_csv('midterm_2016.csv')

    df.drop(['parcelid', 'airconditioningtypeid'], axis=1, inplace=True)
    df.drop(
        ['buildingqualitytypeid', 'finishedfloor1squarefeet', 'finishedsquarefeet15',
'finishedsquarefeet50', 'fips',
         'fireplacecnt', 'garagecarcnt', 'garagetotalsqft', 'latitude', 'longitude',
'lotsizesquarefeet', 'poolcnt',
         'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltypeid7',
```

```
'propertylandusetypeid', 'rawcensustractandblock',
        'regionidcity', 'regionidcounty', 'regionidneighborhood', 'regionidzip',
'roomcnt', 'threequarterbathnbr',
        'unitcnt', 'yardbuildingsqft17', 'yearbuilt', 'numberofstories',
'taxvaluedollarcnt', 'assessmentyear',
        'landtaxvaluedollarcnt', 'taxamount', 'taxdelinquencyyear',
'censustractandblock'], axis=1, inplace=True)

    wine = h2o.H2OFrame(df)

    wine.head(5)

    feature = list(wine.columns)
    feature.remove('logerror')

    wine_split = wine.split_frame(ratios=[0.75])
    wine_train = wine_split[0]
    wine_test = wine_split[1]


    glm_default = H2OGeneralizedLinearEstimator(family='gaussian',
model_id='glm_default')

    glm_default.train(x=feature, y='logerror', training_frame=wine_train)
    print(glm_default)
    yhat_test_glm = glm_default.predict(wine_test)
    print(yhat_test_glm)

    # 使用h2o重做的2016年Tree分析和结论


    df = pd.read_csv('midterm_2016.csv')

    df.drop(['parcelid', 'airconditioningtypeid'], axis=1, inplace=True)
    df.drop(
        ['buildingqualitytypeid', 'finishedfloor1squarefeet', 'finishedsquarefeet15',
'finishedsquarefeet50', 'fips',
        'fireplacecnt', 'garagecarcnt', 'garagetotalsqft', 'latitude', 'longitude',
'lotsizesquarefeet', 'poolcnt',
        'poolsizesum', 'pooltypeid10', 'pooltypeid2', 'pooltypeid7',
'propertylandusetypeid', 'rawcensustractandblock',
        'regionidcity', 'regionidcounty', 'regionidneighborhood', 'regionidzip',
'roomcnt', 'threequarterbathnbr',
        'unitcnt', 'yardbuildingsqft17', 'yearbuilt', 'numberofstories',
'taxvaluedollarcnt', 'assessmentyear',
        'landtaxvaluedollarcnt', 'taxamount', 'taxdelinquencyyear',
'censustractandblock'], axis=1, inplace=True)

    wine = h2o.H2OFrame(df)

    wine.head(5)

    feature = list(wine.columns)
    feature.remove('logerror')

    wine_split = wine.split_frame(ratios=[0.75])
    wine_train = wine_split[0]
    wine_test = wine_split[1]

    drf_default = H2ORandomForestEstimator(seed=1234, model_id='drf_default')
    drf_default.train(x=feature, y='logerror', training_frame=wine_train)
```

```
    print(drf_default)
    drf_test_glm = drf_default.predict(wine_test)
    print(drf_test_glm)

    feature = list(wine.columns)
    # feature.remove('logerror')
    feature.remove('calculatedfinishedsquarefeet')
    feature.remove('calculatedbathnbr')
    feature.remove('structuretaxvaluedollarcnt')
    feature.remove('bedroomcnt')
    feature.remove('bathroomcnt')
    feature.remove('fullbathcnt')
    wine_split = wine.split_frame(ratios=[0.75])
    wine_train = wine_split[0]
    wine_test = wine_split[1]
    glm_default = H2OGeneralizedLinearEstimator(family='gaussian',
model_id='glm_default')
    glm_default.train(x=feature, y='logerror', training_frame=wine_train)

    drf_default = H2ORandomForestEstimator(seed=1234, model_id='drf_default')
    drf_default.train(x=feature, y='logerror', training_frame=wine_train)
    print(drf_default)

    return jsonify({'languaages':languages})


if __name__=='__main__':
    app.run(debug=True, port=8080)
```

Restful API out put
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6
"/Users/zhaomengqi/Desktop/ADS assignment/untitled4/.idea/restful.py"
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
Checking whether there is an H2O instance running at http://localhost:54321. connected.
------------------------- --
H2O cluster uptime: 3 hours 39 mins
H2O cluster version: 3.14.0.7
H2O cluster version age: 14 days, 22 hours and 17 minutes
H2O cluster name: H2O_from_python_zhaomengqi_6xausm
H2O cluster total nodes: 1
H2O cluster free memory: 1.551 Gb
H2O cluster total cores: 4
H2O cluster allowed cores: 4
H2O cluster status: locked, healthy
H2O connection url: http://localhost:54321
H2O connection proxy:

```
H2O internal security:      False
H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
Python version:             3.6.2 final
--------------------------  ----------------------------------------
Checking whether there is an H2O instance running at http://localhost:54321. connected.
--------------------------  ----------------------------------------
H2O cluster uptime:         3 hours 39 mins
H2O cluster version:        3.14.0.7
H2O cluster version age:    14 days, 22 hours and 17 minutes
H2O cluster name:           H2O_from_python_zhaomengqi_6xausm
H2O cluster total nodes:    1
H2O cluster free memory:    1.551 Gb
H2O cluster total cores:    4
H2O cluster allowed cores:  4
H2O cluster status:         locked, healthy
H2O connection url:         http://localhost:54321
H2O connection proxy:
H2O internal security:      False
H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
Python version:             3.6.2 final
--------------------------  ----------------------------------------
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated in
version 0.18 in favor of the model_selection module into which all the refactored classes and
functions are moved. Also note that the interface of the new CV iterators are different from
that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
Checking whether there is an H2O instance running at http://localhost:54321. connected.
--------------------------  ----------------------------------------
H2O cluster uptime:         3 hours 39 mins
H2O cluster version:        3.14.0.7
H2O cluster version age:    14 days, 22 hours and 17 minutes
H2O cluster name:           H2O_from_python_zhaomengqi_6xausm
H2O cluster total nodes:    1
H2O cluster free memory:    1.551 Gb
H2O cluster total cores:    4
H2O cluster allowed cores:  4
H2O cluster status:         locked, healthy
H2O connection url:         http://localhost:54321
H2O connection proxy:
H2O internal security:      False
H2O API Extensions:         XGBoost, Algos, AutoML, Core V3, Core V4
Python version:             3.6.2 final
--------------------------  ----------------------------------------
```

```
Checking whether there is an H2O instance running at http://localhost:54321. connected.
-------------------------  ----------------------------------------
H2O cluster uptime:        3 hours 39 mins
H2O cluster version:       3.14.0.7
H2O cluster version age:   14 days, 22 hours and 17 minutes
H2O cluster name:          H2O_from_python_zhaomengqi_6xausm
H2O cluster total nodes:   1
H2O cluster free memory:   1.551 Gb
H2O cluster total cores:   4
H2O cluster allowed cores: 4
H2O cluster status:        locked, healthy
H2O connection url:        http://localhost:54321
H2O connection proxy:
H2O internal security:     False
H2O API Extensions:        XGBoost, Algos, AutoML, Core V3, Core V4
Python version:            3.6.2 final
-------------------------  ----------------------------------------
WARNING:werkzeug: * Debugger is active!
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/flask/app.py:1982: DtypeWarning: Columns (22,32,34,49,55) have mixed types.
Specify dtype option on import or set low_memory=False.
  response = self.full_dispatch_request()
(90275, 60)
1    90026
2      123
3        1
Name: parcelid, dtype: int64
parcelid                      0
airconditioningtypeid         2173698
architecturalstyletypeid      2979156
basementsqft                  2983589
bathroomcnt                   11462
bedroomcnt                    11450
buildingclasstypeid           2972588
buildingqualitytypeid         1046729
calculatedbathnbr             128912
decktypeid                    2968121
finishedfloor1squarefeet      2782500
calculatedfinishedsquarefeet  55565
finishedsquarefeet12          276033
finishedsquarefeet13          2977545
finishedsquarefeet15          2794419
finishedsquarefeet50          2782500
finishedsquarefeet6           2963216
```

fips 11437
fireplacecnt 2672580
fullbathcnt 128912
garagecarcnt 2101950
garagetotalsqft 2101950
hashottuborspa 2916203
heatingorsystemtypeid 1178816
latitude 11437
longitude 11437
lotsizesquarefeet 276099
poolcnt 2467683
poolsizesum 2957257
pooltypeid10 2948278
pooltypeid2 2953142
pooltypeid7 2499758
propertycountylandusecode 12277
propertylandusetypeid 11437
propertyzoningdesc 1006588
rawcensustractandblock 11437
regionidcity 62845
regionidcounty 11437
regionidneighborhood 1828815
regionidzip 13980
roomcnt 11475
storytypeid 2983593
threequarterbathnbr 2673586
typeconstructiontypeid 2978470
unitcnt 1007727
yardbuildingsqft17 2904862
yardbuildingsqft26 2982570
yearbuilt 59928
numberofstories 2303148
fireplaceflag 2980054
structuretaxvaluedollarcnt 54982
taxvaluedollarcnt 42550
assessmentyear 11439
landtaxvaluedollarcnt 67733
taxamount 31250
taxdelinquencyflag 2928755
taxdelinquencyyear 2928753
censustractandblock 75126
dtype: int64

	Count	Column	Type
0		parcelid	int64

1	logerror	float64
2	transactiondate	object
3	airconditioningtypeid	float64
4	architecturalstyletypeid	float64
5	basementsqft	float64
6	bathroomcnt	float64
7	bedroomcnt	float64
8	buildingclasstypeid	float64
9	buildingqualitytypeid	float64
10	calculatedbathnbr	float64
11	decktypeid	float64
12	finishedfloor1squarefeet	float64
13	calculatedfinishedsquarefeet	float64
14	finishedsquarefeet12	float64
15	finishedsquarefeet13	float64
16	finishedsquarefeet15	float64
17	finishedsquarefeet50	float64
18	finishedsquarefeet6	float64
19	fips	float64
20	fireplacecnt	float64
21	fullbathcnt	float64
22	garagecarcnt	float64
23	garagetotalsqft	float64
24	hashottuborspa	object
25	heatingorsystemtypeid	float64
26	latitude	float64
27	longitude	float64
28	lotsizesquarefeet	float64
29	poolcnt	float64
30	poolsizesum	float64
31	pooltypeid10	float64
32	pooltypeid2	float64
33	pooltypeid7	float64
34	propertycountylandusecode	object
35	propertylandusetypeid	float64
36	propertyzoningdesc	object
37	rawcensustractandblock	float64
38	regionidcity	float64
39	regionidcounty	float64
40	regionidneighborhood	float64
41	regionidzip	float64
42	roomcnt	float64
43	storytypeid	float64
44	threequarterbathnbr	float64

```
45      typeconstructiontypeid     float64
46                    unitcnt     float64
47          yardbuildingsqft17     float64
48          yardbuildingsqft26     float64
49                   yearbuilt     float64
50             numberofstories     float64
51              fireplaceflag      object
52    structuretaxvaluedollarcnt     float64
53           taxvaluedollarcnt     float64
54              assessmentyear     float64
55       landtaxvaluedollarcnt     float64
56                   taxamount     float64
57          taxdelinquencyflag      object
58          taxdelinquencyyear     float64
59         censustractandblock     float64
             column_name  missing_count  missing_ratio
4   architecturalstyletypeid         90014       0.997109
5             basementsqft         90232       0.999524
8       buildingclasstypeid         90259       0.999823
11             decktypeid         89617       0.992711
15     finishedsquarefeet13         90242       0.999634
18      finishedsquarefeet6         89854       0.995336
43             storytypeid         90232       0.999524
45    typeconstructiontypeid         89976       0.996688
48       yardbuildingsqft26         90180       0.998948
51            fireplaceflag         90053       0.997541
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/flask/app.py:1982: DtypeWarning: Columns
(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,3
4,35,36,37,38,39,40,41,42,43,44) have mixed types. Specify dtype option on import or set
low_memory=False.
  response = self.full_dispatch_request()
(180551, 45)
(77613, 44)
pleasepass
pleasepassokkk
zhaomengqi8000
zillowdata
successfully uploaded to s3
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/scipy/linalg/basic.py:1018: RuntimeWarning: internal gelsd driver lwork query error,
required iwork dimension not returned. This is likely the result of LAPACK bug 0038, fixed in
LAPACK 3.2.2 (released July 21, 2010). Falling back to 'gelss' driver.
  warnings.warn(mesg, RuntimeWarning)
```

RMSE by hand: 3.23503083134
6.67192752436
0.0231853969598
0.032009358968
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
oob_score=False, random_state=None, verbose=0, warm_start=False)
RMSE by hand: 3.46778994315
7.98806420749
0.0266417809602
0.0427743333333
Iteration 1, loss = 0.04286001
Iteration 2, loss = 0.01081448
Iteration 3, loss = 0.02226176
Iteration 4, loss = 0.01962681
Iteration 5, loss = 0.01623865
Training loss did not improve more than tol=0.000001 for two consecutive epochs. Stopping.
Root Mean Square Error 0.254378497323
Parse progress: |██| 100%
glm Model Build progress: |██| 100%
Model Details
==========
H2OGeneralizedLinearEstimator : Generalized Linear Modeling
Model Key: glm_default

ModelMetricsRegressionGLM: glm
** Reported on train data. **

MSE: 0.02540524033007391
RMSE: 0.15939021403484566
MAE: 0.06809191486452887
RMSLE: NaN
R^2: 0.0020341405236042887
Mean Residual Deviance: 0.02540524033007391
Null degrees of freedom: 67562
Residual degrees of freedom: 67556
Null deviance: 1719.9528782692646
Residual deviance: 1716.4542524207836
AIC: -56393.885597457025
Scoring History:

```
   timestamp           duration    iterations    negative_log_likelihood    objective
--  ------------------  ----------  ------------  -------------------------  -----------
   2017-11-04 16:56:27  0.000 sec   0            1719.95                    0.025457

glm prediction progress: |████████████████████████████████████████████████| 100%
  predict
----------
0.0129944
0.0170058
0.00712228
0.0202184
0.00934625
0.00922398
0.00547681
0.0118972
0.00778651
0.00889289

[22712 rows x 1 column]

Parse progress: |████████████████████████████████████████████████| 100%
drf Model Build progress: |████████████████████████████████████████████████| 100%
Model Details
=============
H2ORandomForestEstimator :  Distributed Random Forest
Model Key:  drf_default


ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.026001739759042998
RMSE: 0.16125054963950666
MAE: 0.07150445789746532
RMSLE: NaN
Mean Residual Deviance: 0.026001739759042998
Scoring History:
    timestamp           duration    number_of_trees    training_rmse       training_mae        training_deviance
---  ------------------  ----------  -----------------  ------------------  ------------------  --------------------
    2017-11-04 16:56:31  0.020 sec   0.0                nan                 nan                 nan
    2017-11-04 16:56:31  0.146 sec   1.0                0.18034649381622783  0.0793302085091173  0.032524857831806704
```

2017-11-04 16:56:31 0.225 sec 2.0 0.17930975771931595 0.07902296101959518 0.03215198921335979
2017-11-04 16:56:31 0.304 sec 3.0 0.17755547239971525 0.07807291819021552 0.031525945779086045
2017-11-04 16:56:31 0.364 sec 4.0 0.17358654302209944 0.07682897375336568 0.03013228791836318
--- --- --- --- --- --- ---
2017-11-04 16:56:34 3.734 sec 44.0 0.16127545761063564 0.07156597860310664 0.02600977322751993
2017-11-04 16:56:34 3.804 sec 45.0 0.16130305579929677 0.07156640977856138 0.026018675810191048
2017-11-04 16:56:34 3.892 sec 46.0 0.16127838673080605 0.07151837903118605 0.026010718026491436
2017-11-04 16:56:34 3.965 sec 47.0 0.1612596860709552 0.07153270257101148 0.02600468635170302
2017-11-04 16:56:35 4.185 sec 50.0 0.16125054963950666 0.07150445789746532 0.026001739759042998

See the whole table with table.as_data_frame()
Variable Importances:

variable	relative_importance	scaled_importance	percentage
structuretaxvaluedollarcnt	5088.23	1	0.334588
calculatedfinishedsquarefeet	3881.46	0.762831	0.255234
finishedsquarefeet12	3358.9	0.660131	0.220872
bedroomcnt	1107.95	0.217747	0.0728555
heatingorsystemtypeid	628.469	0.123514	0.0413264
calculatedbathnbr	425.791	0.0836816	0.0279989
bathroomcnt	424.947	0.0835157	0.0279434
fullbathcnt	291.706	0.0573295	0.0191818

drf prediction progress: |██| 100%

predict
0.0100671
0.0223408
0.00342415
0.00905111
0.00427072
0.0102113
0.0153378
0.00712938
0.0113307
-0.0280612

[22439 rows x 1 column]

glm Model Build progress: |██| 100%
drf Model Build progress: |██| 100%
Model Details
=============
H2ORandomForestEstimator : Distributed Random Forest
Model Key: drf_default

ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.026884545516929623
RMSE: 0.16396507407655334
MAE: 0.06884542436550746
RMSLE: NaN
Mean Residual Deviance: 0.026884545516929623
Scoring History:

	timestamp	duration	number_of_trees	training_rmse	training_mae	training_deviance
	2017-11-04 16:56:37	0.022 sec	0.0	nan	nan	nan
	2017-11-04 16:56:37	0.065 sec	1.0	0.15748534589050053	0.06809722927477124	0.02480163417025059
	2017-11-04 16:56:37	0.106 sec	2.0	0.16619897398837222	0.06904248034252604	0.02762209895478762
	2017-11-04 16:56:37	0.138 sec	3.0	0.1615684433695799	0.06858945436736477	0.02610436189286915
	2017-11-04 16:56:37	0.171 sec	4.0	0.16195595335182522	0.06856257023813372	0.02622973082609859
...	...	...	...	...	...	...
	2017-11-04 16:56:38	1.771 sec	46.0	0.1640001608298215	0.06887366837294336	0.02689605275220732
	2017-11-04 16:56:38	1.793 sec	47.0	0.1639922526668773	0.06886704584374913	0.026893458934756927
	2017-11-04 16:56:38	1.819 sec	48.0	0.16398118434172687	0.06885910449744495	0.026889828818115406
	2017-11-04 16:56:38	1.845 sec	49.0	0.16398547946105496	0.06885614837787014	0.02689123747407208
	2017-11-04 16:56:38	1.869 sec	50.0	0.16396507407655334	0.06884542436550746	0.026884545516929623

See the whole table with table.as_data_frame()
Variable Importances:

variable	relative_importance	scaled_importance	percentage
finishedsquarefeet12	1131.61	1	0.893962
heatingorsystemtypeid	134.227	0.118616	0.106038

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/flask/app.py:1982: DtypeWarning: Columns (22,32,34,49,55) have mixed types. Specify dtype option on import or set low_memory=False.
  response = self.full_dispatch_request()
(90275, 60)
1   90026
2   123
3   1
Name: parcelid, dtype: int64
parcelid   0
airconditioningtypeid   2173698
architecturalstyletypeid   2979156
basementsqft   2983589
bathroomcnt   11462
bedroomcnt   11450
buildingclasstypeid   2972588
buildingqualitytypeid   1046729
calculatedbathnbr   128912
decktypeid   2968121
finishedfloor1squarefeet   2782500
calculatedfinishedsquarefeet   55565
finishedsquarefeet12   276033
finishedsquarefeet13   2977545
finishedsquarefeet15   2794419
finishedsquarefeet50   2782500
finishedsquarefeet6   2963216
fips   11437
fireplacecnt   2672580
fullbathcnt   128912
garagecarcnt   2101950
garagetotalsqft   2101950
hashottuborspa   2916203
heatingorsystemtypeid   1178816
latitude   11437
longitude   11437
lotsizesquarefeet   276099
poolcnt   2467683

poolsizesum 2957257
pooltypeid10 2948278
pooltypeid2 2953142
pooltypeid7 2499758
propertycountylandusecode 12277
propertylandusetypeid 11437
propertyzoningdesc 1006588
rawcensustractandblock 11437
regionidcity 62845
regionidcounty 11437
regionidneighborhood 1828815
regionidzip 13980
roomcnt 11475
storytypeid 2983593
threequarterbathnbr 2673586
typeconstructiontypeid 2978470
unitcnt 1007727
yardbuildingsqft17 2904862
yardbuildingsqft26 2982570
yearbuilt 59928
numberofstories 2303148
fireplaceflag 2980054
structuretaxvaluedollarcnt 54982
taxvaluedollarcnt 42550
assessmentyear 11439
landtaxvaluedollarcnt 67733
taxamount 31250
taxdelinquencyflag 2928755
taxdelinquencyyear 2928753
censustractandblock 75126
dtype: int64

	Count	Column	Type
0		parcelid	int64
1		logerror	float64
2		transactiondate	object
3		airconditioningtypeid	float64
4		architecturalstyletypeid	float64
5		basementsqft	float64
6		bathroomcnt	float64
7		bedroomcnt	float64
8		buildingclasstypeid	float64
9		buildingqualitytypeid	float64
10		calculatedbathnbr	float64
11		decktypeid	float64

12 finishedfloor1squarefeet float64
13 calculatedfinishedsquarefeet float64
14 finishedsquarefeet12 float64
15 finishedsquarefeet13 float64
16 finishedsquarefeet15 float64
17 finishedsquarefeet50 float64
18 finishedsquarefeet6 float64
19 fips float64
20 fireplacecnt float64
21 fullbathcnt float64
22 garagecarcnt float64
23 garagetotalsqft float64
24 hashottuborspa object
25 heatingorsystemtypeid float64
26 latitude float64
27 longitude float64
28 lotsizesquarefeet float64
29 poolcnt float64
30 poolsizesum float64
31 pooltypeid10 float64
32 pooltypeid2 float64
33 pooltypeid7 float64
34 propertycountylandusecode object
35 propertylandusetypeid float64
36 propertyzoningdesc object
37 rawcensustractandblock float64
38 regionidcity float64
39 regionidcounty float64
40 regionidneighborhood float64
41 regionidzip float64
42 roomcnt float64
43 storytypeid float64
44 threequarterbathnbr float64
45 typeconstructiontypeid float64
46 unitcnt float64
47 yardbuildingsqft17 float64
48 yardbuildingsqft26 float64
49 yearbuilt float64
50 numberofstories float64
51 fireplaceflag object
52 structuretaxvaluedollarcnt float64
53 taxvaluedollarcnt float64
54 assessmentyear float64
55 landtaxvaluedollarcnt float64

```
56                taxamount     float64
57          taxdelinquencyflag      object
58          taxdelinquencyyear     float64
59         censustractandblock     float64
            column_name  missing_count  missing_ratio
4   architecturalstyletypeid         90014       0.997109
5             basementsqft         90232       0.999524
8       buildingclasstypeid         90259       0.999823
11             decktypeid         89617       0.992711
15     finishedsquarefeet13         90242       0.999634
18      finishedsquarefeet6         89854       0.995336
43             storytypeid         90232       0.999524
45    typeconstructiontypeid         89976       0.996688
48       yardbuildingsqft26         90180       0.998948
51           fireplaceflag         90053       0.997541
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
packages/flask/app.py:1982: DtypeWarning: Columns
(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,3
4,35,36,37,38,39,40,41,42,43,44) have mixed types. Specify dtype option on import or set
low_memory=False.
  response = self.full_dispatch_request()
(180551, 45)
(77613, 44)
pleasepass
pleasepassokkk
zhaomengqi8000
zillowdata
successfully uploaded to s3
RMSE by hand: 3.23503083134
6.67192752436
0.0231853969598
0.032009358968
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
          max_features='auto', max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
          oob_score=False, random_state=None, verbose=0, warm_start=False)
RMSE by hand: 3.47316860189
7.9871539194
0.0267244896477
0.04271
Iteration 1, loss = 0.02254974
Iteration 2, loss = 0.00588820
```

Iteration 3, loss = 0.01205640
Iteration 4, loss = 0.01063225
Iteration 5, loss = 0.00880838
Training loss did not improve more than tol=0.000001 for two consecutive epochs. Stopping.
Root Mean Square Error 0.10389649083
Parse progress: |██| 100%
glm Model Build progress: |██| 100%
Model Details
=============
H2OGeneralizedLinearEstimator : Generalized Linear Modeling
Model Key: glm_default

ModelMetricsRegressionGLM: glm
** Reported on train data. **

MSE: 0.025235205028121918
RMSE: 0.15885592537932577
MAE: 0.06819324932391854
RMSLE: NaN
R^2: 0.001962284375828127
Mean Residual Deviance: 0.025235205028121918
Null degrees of freedom: 67595
Residual degrees of freedom: 67588
Null deviance: 1709.1527628433803
Residual deviance: 1705.7989190809292
AIC: -56873.373608276066
Scoring History:

	timestamp	duration	iterations	negative_log_likelihood	objective
	2017-11-04 22:54:54	0.000 sec	0	1709.15	0.0252848

glm prediction progress: |██| 100%

predict
0.0138061
0.0168696
0.0149531
0.00527981
0.00426544
0.00964355
0.00694458
0.0100362
0.0287963

0.0135877

[22679 rows x 1 column]

Parse progress: |██| 100%
drf Model Build progress: |██| 100%
Model Details
=============
H2ORandomForestEstimator : Distributed Random Forest
Model Key: drf_default

ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.026632846518593895
RMSE: 0.1631957306996537
MAE: 0.071795417770584
RMSLE: NaN
Mean Residual Deviance: 0.026632846518593895
Scoring History:

timestamp	duration	number_of_trees	training_rmse	training_mae	training_deviance
2017-11-04 22:54:57	0.018 sec	0.0	nan	nan	nan
2017-11-04 22:54:57	0.123 sec	1.0	0.17711160459809203	0.07796146280238758	0.03136852048331089
2017-11-04 22:54:57	0.213 sec	2.0	0.18065860623088956	0.07820451008008268	0.03263753200528761
2017-11-04 22:54:57	0.297 sec	3.0	0.17699516995489298	0.0769372694404006	0.03132729018736145
2017-11-04 22:54:58	0.407 sec	4.0	0.17631562030517853	0.07736774871330407	0.031087197963599884
...	...	...	...	...	...
2017-11-04 22:55:01	3.680 sec	38.0	0.1635524191033711	0.07218594971827089	0.026749393794564744
2017-11-04 22:55:01	3.766 sec	39.0	0.16346428270602148	0.07211552883768944	0.026720571720594114
2017-11-04 22:55:01	3.872 sec	40.0	0.1634409682682223	0.07211527393629295	0.026712950108454047
2017-11-04 22:55:01	3.972 sec	41.0	0.16341725764993448	0.07209041567785153	0.026705200097825073
2017-11-04 22:55:03	6.189 sec	50.0	0.1631957306996537	0.071795417770584	0.026632846518593895

See the whole table with table.as_data_frame()
Variable Importances:

variable	relative_importance	scaled_importance	percentage
structuretaxvaluedollarcnt	5498.26	1	0.349451
calculatedfinishedsquarefeet	3989	0.725502	0.253527
finishedsquarefeet12	3436.2	0.624961	0.218393
bedroomcnt	996.559	0.18125	0.0633379
heatingorsystemtypeid	642.163	0.116794	0.0408137
bathroomcnt	442.647	0.0805067	0.0281331
calculatedbathnbr	370.415	0.0673696	0.0235424
fullbathcnt	358.759	0.0652495	0.0228015

drf prediction progress: |██| 100%

predict
0.00834995
0.0130596
0.0128961
-0.0107799
0.0237716
0.018338
0.00413142
0.017087
-0.00349518
0.0126421

[22466 rows x 1 column]

glm Model Build progress: |██| 100%
drf Model Build progress: |██| 100%
Model Details
==========
H2ORandomForestEstimator : Distributed Random Forest
Model Key: drf_default

ModelMetricsRegression: drf
** Reported on train data. **

MSE: 0.026019161237669154
RMSE: 0.1613045604986702
MAE: 0.06872969270932267

RMSLE: NaN
Mean Residual Deviance: 0.026019161237669154
Scoring History:

timestamp	duration	number_of_trees	training_rmse	training_mae	training_deviance
2017-11-04 22:55:05	0.007 sec	0.0	nan	nan	nan
2017-11-04 22:55:05	0.051 sec	1.0	0.15838229843056795	0.06897981857699179	0.02508495245614949
2017-11-04 22:55:05	0.078 sec	2.0	0.1600144070181115	0.06873334304303697	0.025604610453357855
2017-11-04 22:55:05	0.100 sec	3.0	0.16156572393574462	0.06903996121723598	0.026103483150881242
2017-11-04 22:55:05	0.130 sec	4.0	0.16160583450127536	0.06900852681624903	0.026116445744853597
...	...	...	...	...	...
2017-11-04 22:55:07	1.814 sec	46.0	0.1613565454886926	0.06875272672506137	0.026035934772044522
2017-11-04 22:55:07	1.842 sec	47.0	0.16135002293129944	0.06875010295329419	0.02603382989993085
2017-11-04 22:55:07	1.861 sec	48.0	0.16132164048883715	0.06874009839852625	0.02602467169000962
2017-11-04 22:55:07	1.891 sec	49.0	0.16131022250977164	0.0687350118765633	0.02602098788615204
2017-11-04 22:55:07	1.912 sec	50.0	0.1613045604986702	0.06872969270932267	0.026019161237669154

See the whole table with table.as_data_frame()
Variable Importances:

variable	relative_importance	scaled_importance	percentage
finishedsquarefeet12	1197.26	1	0.89499
heatingorsystemtypeid	140.475	0.117331	0.10501

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/flask/app.py:1982: DtypeWarning: Columns (22,32,34,49,55) have mixed types. Specify dtype option on import or set low_memory=False.
  response = self.full_dispatch_request()

Json API will out put
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Users/zhaomengqi/Desktop/ADS assignment/untitled4/.idea/invoke.py"
{'ok': [{'work': well}, {Rest_APi: well}, {Json: well}]}
Process finished with exit code 0