In [12]:

```
# imports
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.preprocessing import LabelEncoder
import lightgbm as lgb
import gc
from sklearn.linear_model import LinearRegression
import random
import datetime as dt

################
################
##  LightGBM changes ##
# V42 - sub_feature: 0.3 -> 0.35 : LB = 0.0643759
# V34 - sub_feature: 0.5 -> 0.42
# V33 - sub_feature: 0.5 -> 0.45 : LB = 0.0643866
# - sub_feature: 0.45 -> 0.3 : LB = 0.0643811 / 0.0643814
################
################

# Parameters
XGB_WEIGHT = 0.6415
BASELINE_WEIGHT = 0.0056
OLS_WEIGHT = 0.0828

XGB1_WEIGHT = 0.8083  # Weight of first in combination of two XGB models

BASELINE_PRED = 0.0115   # Baseline based on mean of training data, per Ole
g

##### READ IN RAW DATA
print( "\nReading data from disk ...")
prop = pd.read_csv('properties_2016.csv')
train = pd.read_csv("train_2016_v2.csv")
```

```
Reading data from disk ...
```

```
D:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:2717: DtypeW
arning: Columns (22,32,34,49,55) have mixed types. Specify dtype option on
import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [13]:

```
################
################
##  LightGBM  ##
################
################

##### PROCESS DATA FOR LIGHTGBM
print( "\nProcessing data for LightGBM ..." )
for c, dtype in zip(prop.columns, prop.dtypes):
    if dtype == np.float64:
        prop[c] = prop[c].astype(np.float32)
```

```
df_train = train.merge(prop, how='left', on='parcelid')
df_train.fillna(df_train.median(),inplace = True)

x_train = df_train.drop(['parcelid', 'logerror', 'transactiondate', 'proper
tyzoningdesc',
                         'propertycountylandusecode', 'fireplacecnt', 'fire
laceflag'], axis=1)
#x_train['Ratio_1'] = x_train['taxvaluedollarcnt']/x_train['taxamount']
y_train = df_train['logerror'].values
print(x_train.shape, y_train.shape)


train_columns = x_train.columns

for c in x_train.dtypes[x_train.dtypes == object].index.values:
    x_train[c] = (x_train[c] == True)

del df_train; gc.collect()

x_train = x_train.values.astype(np.float32, copy=False)
d_train = lgb.Dataset(x_train, label=y_train)



##### RUN LIGHTGBM
params = {}
params['max_bin'] = 10
params['learning_rate'] = 0.0021 # shrinkage_rate
params['boosting_type'] = 'gbdt'
params['objective'] = 'regression'
params['metric'] = 'l1'          # or 'mae'
params['sub_feature'] = 0.345
params['bagging_fraction'] = 0.85 # sub_row
params['bagging_freq'] = 40
params['num_leaves'] = 512        # num_leaf
params['min_data'] = 500         # min_data_in_leaf
params['min_hessian'] = 0.05     # min_sum_hessian_in_leaf
params['verbose'] = 0
params['feature_fraction_seed'] = 2
params['bagging_seed'] = 3

print("\nFitting LightGBM model ...")
clf = lgb.train(params, d_train, 430)

del d_train; gc.collect()
del x_train; gc.collect()

print("\nPrepare for LightGBM prediction ...")
print("   Read sample file ...")
sample = pd.read_csv('sample_submission.csv')
print("   ...")
sample['parcelid'] = sample['ParcelId']
print("   Merge with property data ...")
df_test = sample.merge(prop, on='parcelid', how='left')
print("   ...")
del sample, prop; gc.collect()
print("   ...")
#df_test['Ratio_1'] = df_test['taxvaluedollarcnt']/df_test['taxamount']
x_test = df_test[train_columns]
```

```
print("   ...")
del df_test; gc.collect()
print("   Preparing x_test...")
for c in x_test.dtypes[x_test.dtypes == object].index.values:
    x_test[c] = (x_test[c] == True)
print("   ...")
x_test = x_test.values.astype(np.float32, copy=False)
print("Test shape :", x_test.shape)

print("\nStart LightGBM prediction ...")
p_test = clf.predict(x_test)

del x_test; gc.collect()

print( "\nUnadjusted LightGBM predictions:" )
print( pd.DataFrame(p_test).head() )
```

```
Processing data for LightGBM ...
(90275, 53) (90275,)

Fitting LightGBM model ...

Prepare for LightGBM prediction ...
   Read sample file ...
   ...
   Merge with property data ...
   ...
   ...
   ...
   Preparing x_test...
   ...
Test shape : (2985217, 53)

Start LightGBM prediction ...

Unadjusted LightGBM predictions:
          0
0  0.032601
1  0.033056
2  0.033039
3  0.028782
4  0.023099
```

In [14]:

```
################
################
##  XGBoost   ##
################
################

##### RE-READ PROPERTIES FILE
##### (I tried keeping a copy, but the program crashed.)

print( "\nRe-reading properties file ...")
properties = pd.read_csv('properties_2016.csv')

##### PROCESS DATA FOR XGBOOST
print( "\nProcessing data for XGBoost ...")
for c in properties.columns:
```

```
    properties[c]=properties[c].fillna(-1)
    if properties[c].dtype == 'object':
        lbl = LabelEncoder()
        lbl.fit(list(properties[c].values))
        properties[c] = lbl.transform(list(properties[c].values))

train_df = train.merge(properties, how='left', on='parcelid')
x_train = train_df.drop(['parcelid', 'logerror','transactiondate'], axis=1)
x_test = properties.drop(['parcelid'], axis=1)
# shape
print('Shape train: {}\nShape test: {}'.format(x_train.shape, x_test.shape)
)

# drop out ouliers
train_df=train_df[ train_df.logerror > -0.4 ]
train_df=train_df[ train_df.logerror < 0.419 ]
x_train=train_df.drop(['parcelid', 'logerror','transactiondate'], axis=1)
y_train = train_df["logerror"].values.astype(np.float32)
y_mean = np.mean(y_train)

print('After removing outliers:')
print('Shape train: {}\nShape test: {}'.format(x_train.shape, x_test.shape)
)




##### RUN XGBOOST
print("\nSetting up data for XGBoost ...")
# xgboost params
xgb_params = {
    'eta': 0.037,
    'max_depth': 5,
    'subsample': 0.80,
    'objective': 'reg:linear',
    'eval_metric': 'mae',
    'lambda': 0.8,
    'alpha': 0.4,
    'base_score': y_mean,
    'silent': 1
}

dtrain = xgb.DMatrix(x_train, y_train)
dtest = xgb.DMatrix(x_test)

num_boost_rounds = 250
print("num_boost_rounds="+str(num_boost_rounds))

# train model
print( "\nTraining XGBoost ...")
model = xgb.train(dict(xgb_params, silent=1), dtrain,
num_boost_round=num_boost_rounds)

print( "\nPredicting with XGBoost ...")
xgb_pred1 = model.predict(dtest)

print( "\nFirst XGBoost predictions:" )
print( pd.DataFrame(xgb_pred1).head() )
```

```
Re-reading properties file ...
```

```
D:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:2717: DtypeW
arning: Columns (22,32,34,49,55) have mixed types. Specify dtype option on
import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

```
Processing data for XGBoost ...
Shape train: (90275, 57)
Shape test: (2985217, 57)
After removing outliers:
Shape train: (88528, 57)
Shape test: (2985217, 57)

Setting up data for XGBoost ...
num_boost_rounds=250

Training XGBoost ...

Predicting with XGBoost ...

First XGBoost predictions:
          0
0 -0.042947
1 -0.029738
2  0.027966
3  0.069254
4  0.014018
```

In [15]:

```
##### RUN XGBOOST AGAIN
print("\nSetting up data for XGBoost ...")
# xgboost params
xgb_params = {
    'eta': 0.033,
    'max_depth': 6,
    'subsample': 0.80,
    'objective': 'reg:linear',
    'eval_metric': 'mae',
    'base_score': y_mean,
    'silent': 1
}

num_boost_rounds = 150
print("num_boost_rounds="+str(num_boost_rounds))

print( "\nTraining XGBoost again ...")
model = xgb.train(dict(xgb_params, silent=1), dtrain,
num_boost_round=num_boost_rounds)

print( "\nPredicting with XGBoost again ...")
xgb_pred2 = model.predict(dtest)

print( "\nSecond XGBoost predictions:" )
print( pd.DataFrame(xgb_pred2).head() )



##### COMBINE XGBOOST RESULTS

xgb_pred = XGB1_WEIGHT*xgb_pred1 + (1-XGB1_WEIGHT)*xgb_pred2
```

```
#xgb_pred = xgb_pred1

print( "\nCombined XGBoost predictions:" )
print( pd.DataFrame(xgb_pred).head() )

del train_df
del x_train
del x_test
del properties
del dtest
del dtrain
del xgb_pred1
del xgb_pred2
gc.collect()
```

```
Setting up data for XGBoost ...
num_boost_rounds=150

Training XGBoost again ...

Predicting with XGBoost again ...

Second XGBoost predictions:
          0
0 -0.048613
1 -0.022864
2  0.016268
3  0.056134
4  0.005422

Combined XGBoost predictions:
          0
0 -0.044033
1 -0.028420
2  0.025723
3  0.066739
4  0.012370
```

Out[15]:

```
164
```

In [16]:

```
################
################
##    OLS     ##
################
################

# This section is derived from the1owl's notebook:
#    https://www.kaggle.com/the1owl/primer-for-the-zillow-pred-approach
# which I (Andy Harless) updated and made into a script:
#    https://www.kaggle.com/aharless/updated-script-version-of-the1owl-s-ba
sic-ols
np.random.seed(17)
random.seed(17)

train = pd.read_csv("train_2016_v2.csv", parse_dates=["transactiondate"])
properties = pd.read_csv("properties_2016.csv")
submission = pd.read_csv("sample_submission.csv")
```

```
print(len(train),len(properties),len(submission))

def get_features(df):
    df["transactiondate"] = pd.to_datetime(df["transactiondate"])
    df["transactiondate_year"] = df["transactiondate"].dt.year
    df["transactiondate_month"] = df["transactiondate"].dt.month
    df['transactiondate'] = df['transactiondate'].dt.quarter
    df = df.fillna(-1.0)
    return df

def MAE(y, ypred):
    #logerror=log(Zestimate)-log(SalePrice)
    return np.sum([abs(y[i]-ypred[i]) for i in range(len(y))]) / len(y)

train = pd.merge(train, properties, how='left', on='parcelid')
y = train['logerror'].values
test = pd.merge(submission, properties, how='left', left_on='ParcelId', rig
ht_on='parcelid')
properties = [] #memory

exc = [train.columns[c] for c in range(len(train.columns)) if train.dtypes[c
] == 'O'] + ['logerror','parcelid']
col = [c for c in train.columns if c not in exc]

train = get_features(train[col])
test['transactiondate'] = '2016-01-01' #should use the most common training
date
test = get_features(test[col])

reg = LinearRegression(n_jobs=-1)
reg.fit(train, y); print('fit...')
print(MAE(y, reg.predict(train)))
train = [];  y = [] #memory

test_dates = ['2016-10-01','2016-11-01','2016-12-01','2017-10-01','2017-11-
01','2017-12-01']
test_columns = ['201610','201611','201612','201710','201711','201712']




########################
########################
##  Combine and Save  ##
########################
########################

##### COMBINE PREDICTIONS
print( "\nCombining XGBoost, LightGBM, and baseline predicitons ..." )
lgb_weight = (1 - XGB_WEIGHT - BASELINE_WEIGHT) / (1 - OLS_WEIGHT)
xgb_weight0 = XGB_WEIGHT / (1 - OLS_WEIGHT)
baseline_weight0 =  BASELINE_WEIGHT / (1 - OLS_WEIGHT)
pred0 = xgb_weight0*xgb_pred + baseline_weight0*BASELINE_PRED + lgb_weight*
p_test

print( "\nCombined XGB/LGB/baseline predictions:" )
print( pd.DataFrame(pred0).head() )

print( "\nPredicting with OLS and combining with XGB/LGB/baseline predicito
ns: ..." )
for i in range(len(test_dates)):
```

```
for i in range(len(test_dates)):
    test['transactiondate'] = test_dates[i]
    pred = OLS_WEIGHT*reg.predict(get_features(test)) +
(1-OLS_WEIGHT)*pred0
    submission[test_columns[i]] = [float(format(x, '.4f')) for x in pred]
    print('predict...', i)

print( "\nCombined XGB/LGB/baseline/OLS predictions:" )
print( submission.head() )
```

```
D:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:2717: DtypeW
arning: Columns (22,32,34,49,55) have mixed types. Specify dtype option on
import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

```
90275 2985217 2985217
```

```
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:21:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:22:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:23:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy
D:\Anaconda\lib\site-packages\ipykernel_launcher.py:24:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/
stable/indexing.html#indexing-view-versus-copy
```

```
fit...
0.0683700875103

Combining XGBoost, LightGBM, and baseline predicitons ...

Combined XGB/LGB/baseline predictions:
          0
0 -0.018183
1 -0.007088
2  0.030773
3  0.057822
4  0.017609

Predicting with OLS and combining with XGB/LGB/baseline predicitons: ...
predict... 0
predict... 1
```

```
predict... 2
predict... 3
predict... 4
predict... 5

Combined XGB/LGB/baseline/OLS predictions:
   ParcelId  201610  201611  201612  201710  201711  201712
0  10754147 -0.0194 -0.0194 -0.0195 -0.0194 -0.0194 -0.0195
1  10759547 -0.0097 -0.0097 -0.0098 -0.0097 -0.0097 -0.0098
2  10843547  0.0819  0.0819  0.0819  0.0819  0.0819  0.0819
3  10859147  0.0562  0.0562  0.0562  0.0562  0.0562  0.0562
4  10879947  0.0187  0.0187  0.0186  0.0187  0.0187  0.0186
```

In [17]:

```
##### WRITE THE RESULTS
from datetime import datetime
print( "\nWriting results to disk ..." )
submission.to_csv('sub{}.csv'.format(datetime.now().strftime('%Y%m%d_%H%M%S
')), index=False)
print( "\nFinished ...")
```

```
Writing results to disk ...

Finished ...
```