In [1]:

```
import pandas as pd
import numpy as np

from lightgbm import LGBMRegressor
from sklearn.linear_model import RidgeCV
from xgboost import XGBRegressor
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

import pickle
from sklearn.externals import joblib
import warnings
warnings.filterwarnings("ignore")

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:

```
def rmse(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))

def train_save_model(model, X, y, filename):
    # fit model
    model.fit(X, y)

    # save model
    if isinstance(model, GridSearchCV):
        joblib.dump(model.best_estimator_, filename)
    else:
        joblib.dump(model, filename)


def load_eval_model(filename, X_train, X_val, y_train, y_val):
    saved_model = joblib.load(filename)
    if isinstance(saved_model, RidgeCV):
        print(saved_model)
        print("best alpha: ", saved_model.alpha_)
    else:
        print(saved_model)

    # convert back to original scale
    train_preds = saved_model.predict(X_train)
    exp_train_preds = np.exp(train_preds)
    exp_y_train = np.exp(y_train)

    val_preds = saved_model.predict(X_val)
    exp_val_preds = np.exp(val_preds)
    exp_y_val = np.exp(y_val)

    print("----- Training scores -----")
    print("R2 on log scale: ", saved_model.score(X_train, y_train))
    print("RMSE on log scale: ", rmse(y_train, train_preds))
    print("RMSE on original $ scale: ", rmse(exp_y_train, exp_train_preds))

    print("----- Validation scores -----")
    print("R2 on log scale: ", saved_model.score(X_val, y_val))
    print("RMSE on log scale: ", rmse(y_val, val_preds))
    print("RMSE on original $ scale: ", rmse(exp_y_val, exp_val_preds))

    # plot
    fig, ax = plt.subplots(1, 2, figsize=(12, 5))
    sns.regplot(y_val, val_preds, scatter_kws={'alpha':0.2}, ax=ax[0])
    ax[0].set_xlabel("True Value (log)")
    ax[0].set_ylabel("Predictions (log)")
    ax[0].set_title("Log $ Scale")

    sns.regplot(exp_y_val, exp_val_preds, scatter_kws={'alpha':0.2}, ax=ax[1])
    ax[1].set_xlabel("True Value")
    ax[1].set_ylabel("Predictions")
    ax[1].set_title("Original $ Scale")
    plt.tight_layout()

    fig, ax = plt.subplots(1, 2, figsize=(12, 5))
    sns.residplot(y_val, val_preds, scatter_kws={'alpha':0.2}, lowess=True, ax=ax[0])
    ax[0].set_ylabel("Residuals")
    ax[0].set_title("Residual Plot Log $ Scale")

    sns.residplot(exp_y_val, exp_val_preds, scatter_kws={'alpha':0.2}, lowess=True, ax=ax[1])
    ax[1].set_ylabel("Residuals")
    ax[1].set_title("Residual Plot Original $ Scale")
    plt.tight_layout()
```

In [3]:
```
with open('../data/features/CON_feats_all_cleaned.pkl', 'rb') as file:
    con_data_all = pickle.load(file)
con_data_all.fillna(-1, inplace=True)
print("data shape: ", con_data_all.shape)
con_data_all.head()
```

```
data shape:  (40936, 2086)
```

Out[3]:

	MLSNUM	LISTPRICE	SOLDPRICE	DOM	DTO	AGE	LOTSIZE	GARAGE	LISTMONTH	SOLDMONTH	...	resnet_2038	resnet_2039	resnet_2040	resnet_2041	resnet_2042	res
0	71498924	169900.0	177500.0	709	618.0	11.0	-1.0	2.0	3	1	...	0.189716	0.112382	0.940322	0.104880	0.147426	
1	71905628	242000.0	235000.0	91	81.0	14.0	-1.0	1.0	9	1	...	0.189537	0.097927	2.036207	0.169374	0.386018	
2	71918879	209000.0	209000.0	78	37.0	32.0	-1.0	0.0	10	1	...	0.054234	0.028724	0.801957	0.303854	0.065759	
3	71952614	339900.0	350695.0	1	1.0	3.0	-1.0	1.0	1	1	...	0.149968	0.252659	2.011606	0.069289	0.159074	
4	71912071	299900.0	280000.0	83	71.0	10.0	-1.0	1.0	9	2	...	0.091527	0.222217	1.062608	0.104116	0.112099	

5 rows × 2086 columns

In [4]:
```
zillow_features = ['AGE', 'LOTSIZE', 'GARAGE', 'LISTMONTH', 'LAT', 'LNG', 'beds', 'baths', 'sqft_unfinished', 'sqft_finished']
redfin_features = ['num_photo'] + [col for col in con_data_all.columns if '_space' in col] + [col for col in con_data_all.columns i
img_features = [col for col in con_data_all.columns if 'resnet_' in col]
response = ['SOLDPRICE']
print("Number of features from zillow: ", len(zillow_features))
print("Number of features from redfin: ", len(redfin_features))
print("Number of features from images: ", len(img_features))
```

```
Number of features from zillow:  10
Number of features from redfin:  20
Number of features from images:  2048
```

Feature Dimension Reduction Using PCA

Since image feature dimension is large, we plan to use PCA to reduce dimension. Here, we explore how many PCA components captures at least 90% of data variance.

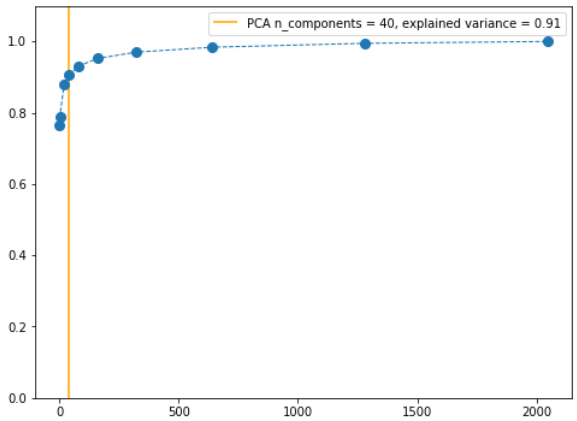
In [5]:
```
df = con_data_all[zillow_features + redfin_features + img_features + response]
data = con_data_all[zillow_features + redfin_features]

# standardize image features
for col in df.columns:
    if col in img_features:
        if df[col].std() == 0:
            data[col] = 0
        else :
            data[col] = (df[col] - df[col].mean()) / df[col].std()

print("data shape: ", data.shape)
```

```
data shape:  (40936, 2078)
```

In [6]:
```
# standardize image features
X_img = data[img_features]
n_components_list = [1, 2, 20, 40, 80, 160, 320, 640, 1280, 2048]
explained_var = []

# try different n_components and print variance explained
for n in n_components_list:
    pca = PCA(n_components=n)
    X_pca = pca.fit_transform(X_img)
    explained_var.append(pca.explained_variance_ratio_.sum())
```

In [7]:
```
fig, ax = plt.subplots(1, 1, figsize=(8, 6))
ax.plot(n_components_list, explained_var, 'o--', linewidth=1, markersize=8)
ax.vlines(x=40, ymin=0, ymax=1.1, color='orange', label='PCA n_components = 40, explained variance = {0:.2f}'.format(explained_var
ax.set_ylim(0, 1.1)
plt.legend()
```

Out[7]:
```
<matplotlib.legend.Legend at 0x7ff87a97c588>
```



It appears that 40 components explains most of the variance (> 90%). Therefore, we will use n_components = 40.

In [8]:
```
# get pca transform of image features only
N_COMPONENTS = 40
pca = PCA(n_components=N_COMPONENTS)
img_pca = pd.DataFrame(pca.fit_transform(X_img), index=data.index)
img_pca_features = ['pca_' + str(i) for i in range(1, N_COMPONENTS+1)]
img_pca.columns = img_pca_features
data = pd.concat([data, img_pca], axis=1)
data['SOLDPRICE'] = df['SOLDPRICE']
del df
print("data shape: ", data.shape)
```

```
data shape:  (40936, 2119)
```

Train test split

In [9]:
```
# train test split
X_train_all, X_val_all, y_train_all, y_val_all = train_test_split(data.iloc[:, :-1],
                                                                  data.iloc[:, -1],
                                                                  test_size=0.1, random_state=9001)
col_names = X_train_all.columns.values
# normalize X_train and X_val
scaler = MinMaxScaler()
scaler.fit(X_train_all)
X_train_all = pd.DataFrame(scaler.transform(X_train_all), columns=col_names)
X_val_all = pd.DataFrame(scaler.transform(X_val_all), columns=col_names)

# take log of responses
y_train_all = np.log(y_train_all)
y_val_all = np.log(y_val_all)

# zillow only, without images
X_train_zil = X_train_all[zillow_features]
X_val_zil = X_val_all[zillow_features]

# zillow + redfin, without images
X_train = X_train_all[zillow_features + redfin_features]
X_val = X_val_all[zillow_features + redfin_features]

# zillow + redfin with images (pca)
X_train_img = X_train_all[zillow_features + redfin_features + img_pca_features]
X_val_img = X_val_all[zillow_features + redfin_features + img_pca_features]

print("shape of zillow only, no images data: ", X_train_zil.shape, X_val_zil.shape)
print("shape of zillow + redfin, no images data: ", X_train.shape, X_val.shape)
print("shape of zillow + redfin, with images data: ", X_train_img.shape, X_val_img.shape)
```

```
shape of zillow only, no images data:  (36842, 10) (4094, 10)
shape of zillow + redfin, no images data:  (36842, 30) (4094, 30)
shape of zillow + redfin, with images data:  (36842, 70) (4094, 70)
```

Models

We compared 3 different models (Ridge, XGBoost and LightGBM) with 3 different feature sets. The feature sets are:

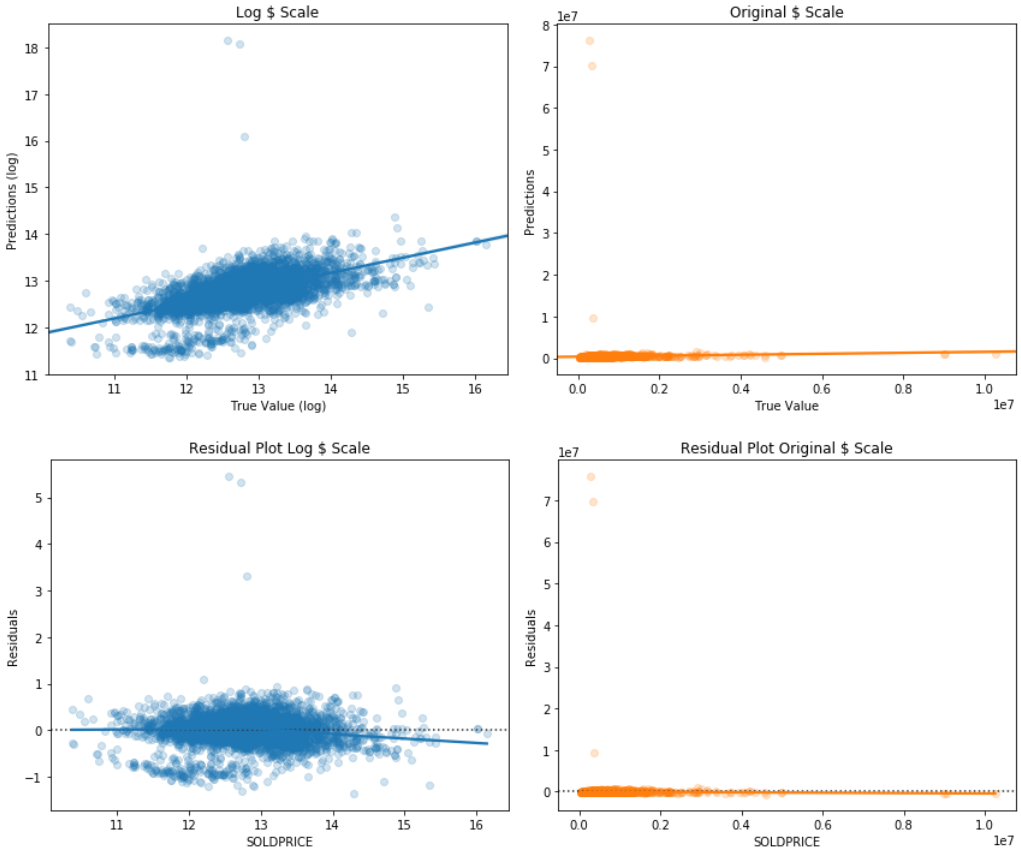
- Zillow data only, no images
- Zillow + Redfin data, no images
- Zillow + Redfin data, with images

Ridge - Zillow data only, no images

In [10]:

```
# Ridge
train_features = X_train_zil
val_features = X_val_zil
# model = RidgeCV(alphas=(1, 1.5, 2, 2.5, 3))
filename = '../data/models/zillow_no_img_ridge.pkl'

print("Ridge model: ")

# train and save model
# train_save_model(model, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
Ridge model:
RidgeCV(alphas=(1, 1.5, 2, 2.5, 3), cv=None, fit_intercept=True,
    gcv_mode=None, normalize=False, scoring=None, store_cv_values=False)
best alpha:  1.5
----- Training scores -----
R2 on log scale:  0.3286758961295816
RMSE on log scale:  0.5855916876599608
RMSE on original $ scale:  1146823.3414184856
----- Validation scores -----
R2 on log scale:  0.3106189421325055
RMSE on log scale:  0.5876581975861629
RMSE on original $ scale:  1681887.7903889953
```

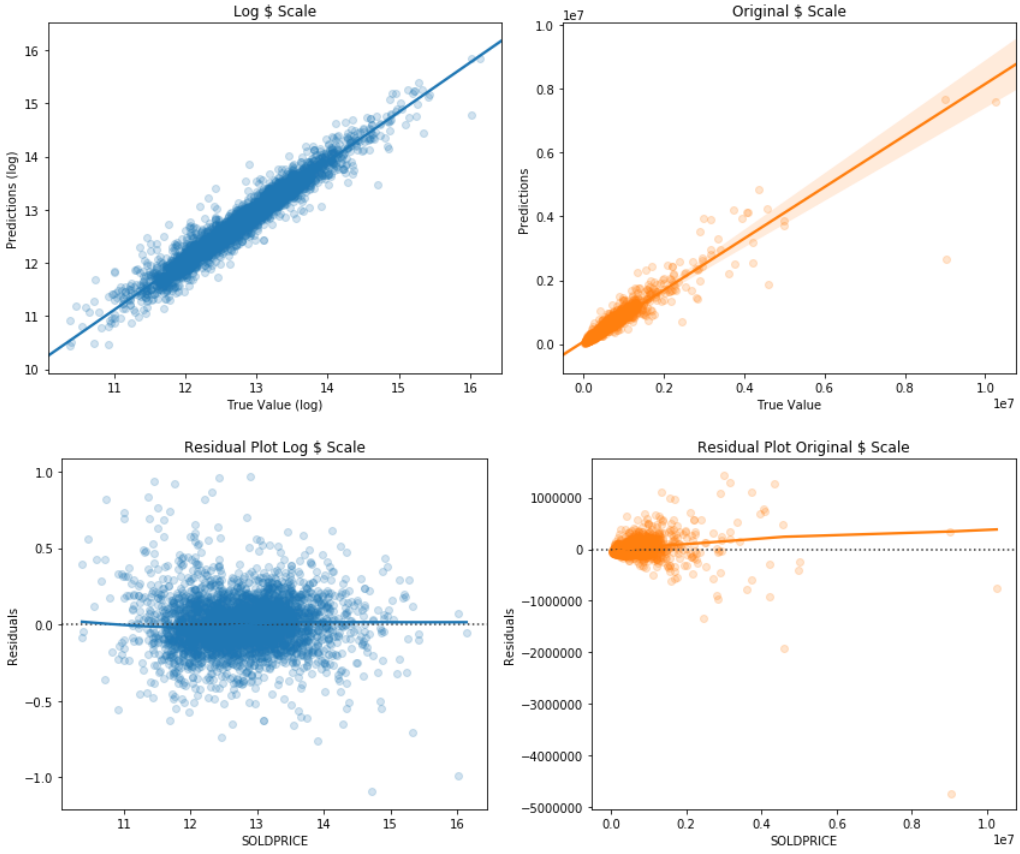


XGBoost - Zillow data only, no images

In [11]:

```
train_features = X_train_zil
val_features = X_val_zil
filename = '../data/models/zillow_no_img_XGBoost.pkl'
params = {
    'max_depth':range(21,28,2),
    'gamma':[i/10.0 for i in range(0,3)],
    'reg_alpha':[1e-2, 0.1, 1, 10]
}

# model
# model = XGBRegressor(random_seed=9001)
# grid = GridSearchCV(model, params, verbose=1, n_jobs=-1)

print("XGBoost model: ")

# train and save model
# train_save_model(grid, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
XGBoost model:
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0.1, learning_rate=0.1, max_delta_step=0,
       max_depth=23, min_child_weight=1, missing=nan, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_seed=9001,
       random_state=0, reg_alpha=0.1, reg_lambda=1, scale_pos_weight=1,
       seed=None, silent=True, subsample=1)
----- Training scores -----
R2 on log scale:  0.9758028165064417
RMSE on log scale:  0.11117603773609616
RMSE on original $ scale:  90313.59113564565
----- Validation scores -----
R2 on log scale:  0.9389638558729383
RMSE on log scale:  0.1748592738897229
RMSE on original $ scale:  169796.0087085459
```

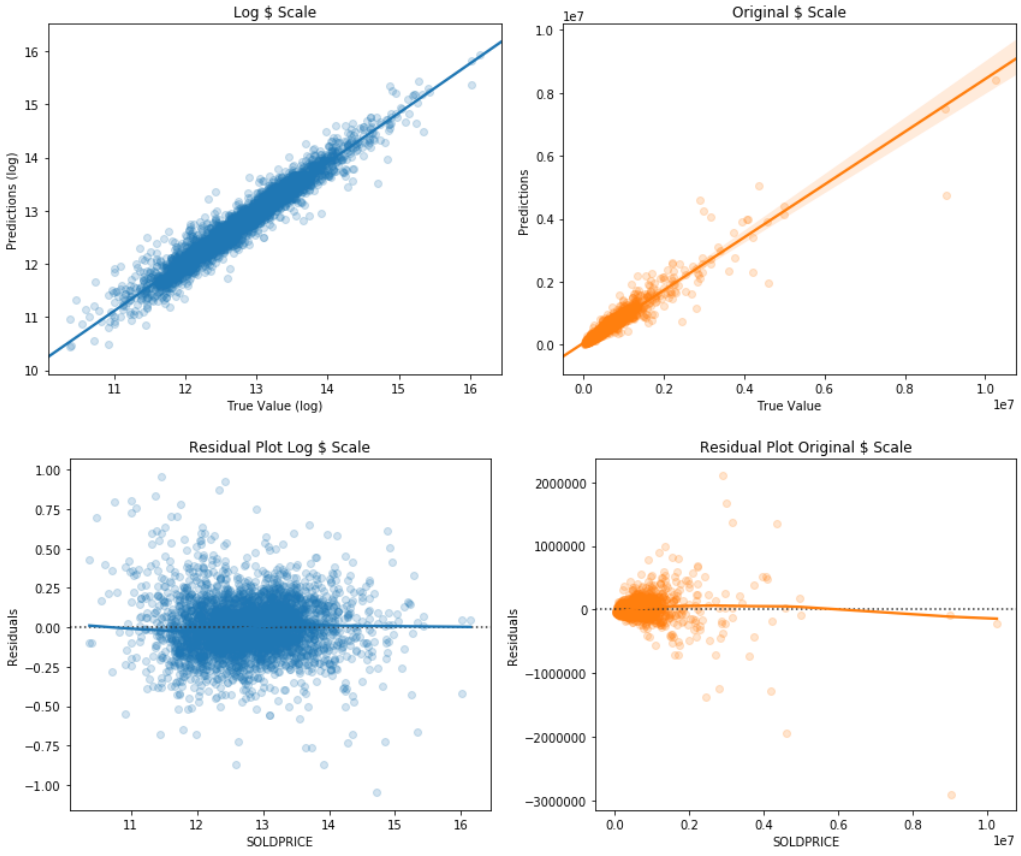


LightGBM - Zillow data only, no images

In [12]:

```
train_features = X_train_zil
val_features = X_val_zil
filename = '../data/models/zillow_no_img_LGBM.pkl'
params = {'num_leaves': [30, 100, 200],
          'max_depth':[-1, 16, 32, 64],
          'learning_rate':[0.01, 0.1, 1],
          'n_estimators':[128, 256, 512]
         }

# model = LGBMRegressor(random_state=9001)
# grid = GridSearchCV(model, params, verbose=1)

print("LightGBM model: ")

# train and save model
# train_save_model(grid, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
LightGBM model:
LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
       learning_rate=0.1, max_depth=-1, min_child_samples=20,
       min_child_weight=0.001, min_split_gain=0.0, n_estimators=256,
       n_jobs=-1, num_leaves=100, objective=None, random_state=9001,
       reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0,
       subsample_for_bin=200000, subsample_freq=1)
----- Training scores -----
R2 on log scale:  0.9571370588786566
RMSE on log scale:  0.14796866884183266
RMSE on original $ scale:  122528.75136794106
----- Validation scores -----
R2 on log scale:  0.93809634640469
RMSE on log scale:  0.17609753093421682
RMSE on original $ scale:  151362.55185733686
```

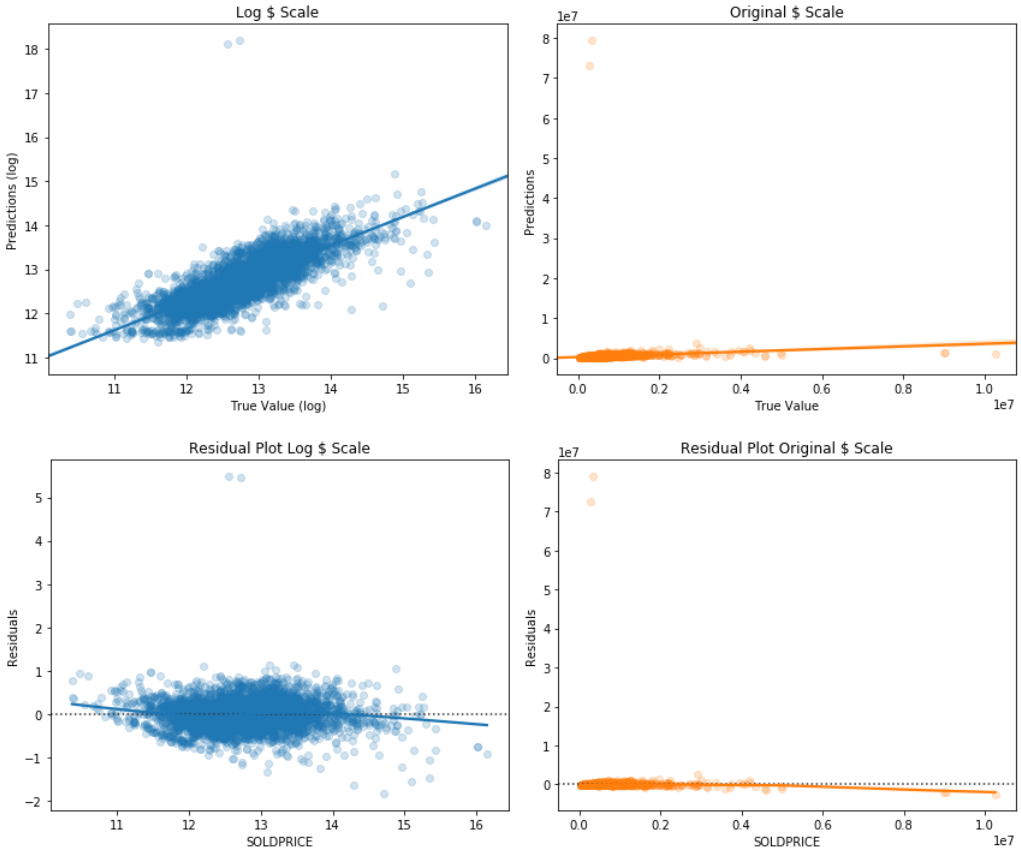


Ridge - Zillow + Redfin data, no images

In [13]:

```
# Ridge
train_features = X_train
val_features = X_val
# model = RidgeCV(alphas=(1, 1.5, 2, 2.5, 3))
filename = '../data/models/zillow-redfin_no_img_ridge.pkl'

print("Ridge model: ")

# train and save model
# train_save_model(model, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
Ridge model:
RidgeCV(alphas=(1, 1.5, 2, 2.5, 3), cv=None, fit_intercept=True,
    gcv_mode=None, normalize=False, scoring=None, store_cv_values=False)
best alpha:  1.0
----- Training scores -----
R2 on log scale:  0.6382971002880207
RMSE on log scale:  0.4298379716712192
RMSE on original $ scale:  678681.9120223082
----- Validation scores -----
R2 on log scale:  0.6292163815737917
RMSE on log scale:  0.43097829120084225
RMSE on original $ scale:  1722852.169374959
```

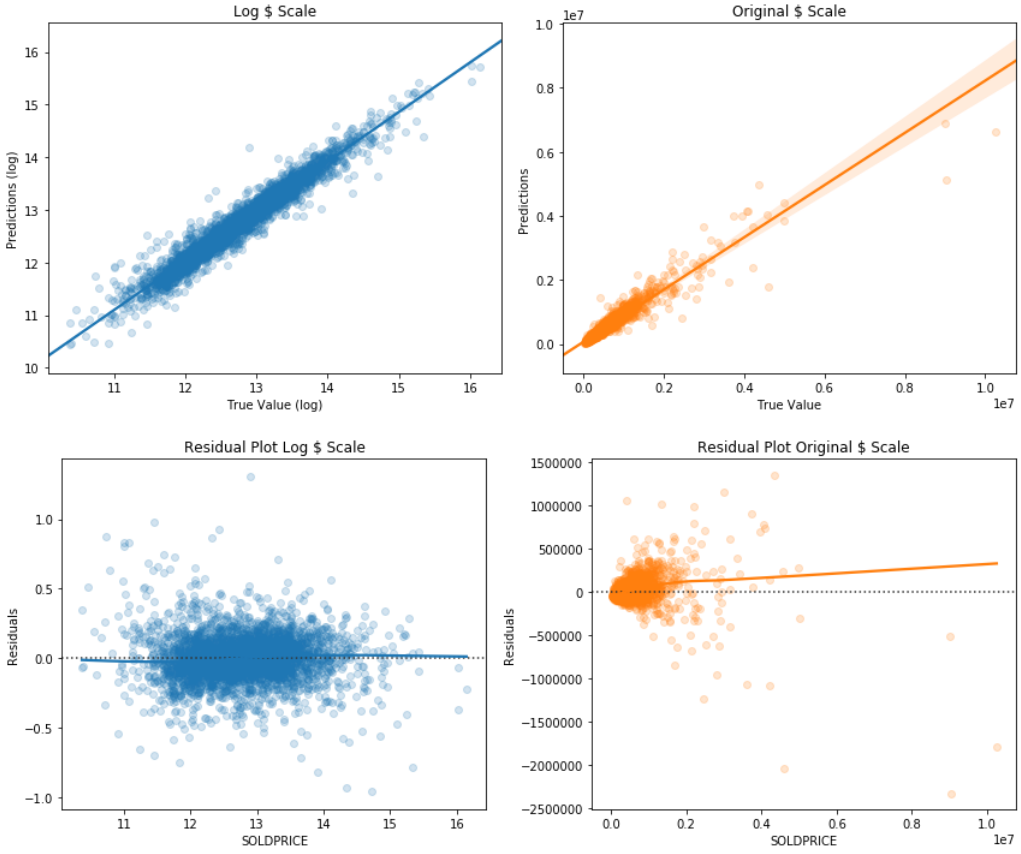


XGBoost - Zillow + Redfin data, no images

In [14]:

```
train_features = X_train
val_features = X_val
filename = '../data/models/zillow-redfin_no_img_XGBoost.pkl'
params = {
    'max_depth':range(21,28,2),
    'gamma':[i/10.0 for i in range(0,3)],
    'reg_alpha':[1e-2, 0.1, 1, 10]
}

# model
# model = XGBRegressor(random_seed=9001)
# grid = GridSearchCV(model, params, verbose=1, n_jobs=-1)

print("XGBoost model: ")

# train and save model
# train_save_model(grid, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
XGBoost model:
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0.0, learning_rate=0.1, max_delta_step=0,
       max_depth=23, min_child_weight=1, missing=nan, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_seed=9001,
       random_state=0, reg_alpha=1, reg_lambda=1, scale_pos_weight=1,
       seed=None, silent=True, subsample=1)
----- Training scores -----
R2 on log scale:  0.9865527050239751
RMSE on log scale:  0.08287933328159086
RMSE on original $ scale:  84592.1591625857
----- Validation scores -----
R2 on log scale:  0.9452556746101984
RMSE on log scale:  0.16560165771758065
RMSE on original $ scale:  153095.5352369099
```

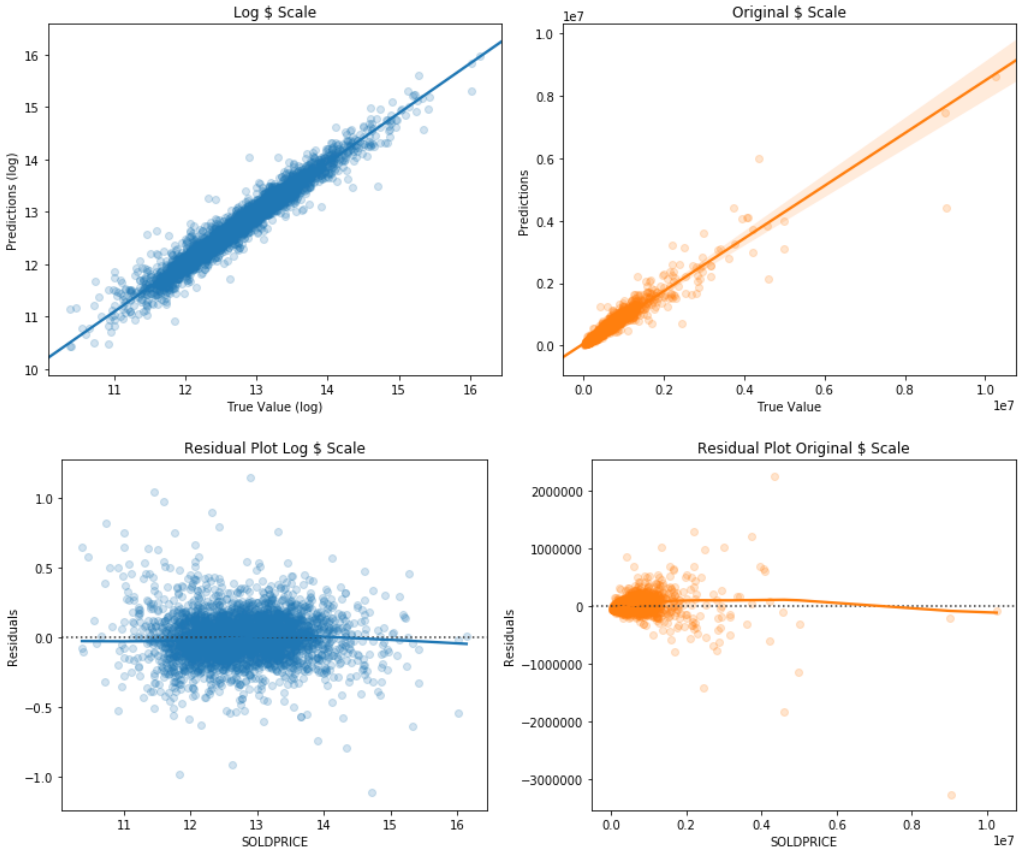


LightGBM - Zillow + Redfin data, no images

In [15]:

```
train_features = X_train
val_features = X_val
filename = '../data/models/zillow-redfin_no_img_LGBM.pkl'
params = {'num_leaves': [15, 30, 100],
          'max_depth':[-1, 8, 16, 32],
          'learning_rate':[0.01, 0.1, 1],
          'n_estimators':[512, 1024, 2048]
         }

# model = LGBMRegressor(random_state=9001)
# grid = GridSearchCV(model, params, verbose=1)
print("LightGBM model: ")

# # train and save model
# train_save_model(grid, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
LightGBM model:
LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
       learning_rate=0.1, max_depth=16, min_child_samples=20,
       min_child_weight=0.001, min_split_gain=0.0, n_estimators=2048,
       n_jobs=-1, num_leaves=30, objective=None, random_state=9001,
       reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0,
       subsample_for_bin=200000, subsample_freq=1)
----- Training scores -----
R2 on log scale:  0.9832844145177274
RMSE on log scale:  0.0924037463746428
RMSE on original $ scale:  72523.90971867189
----- Validation scores -----
R2 on log scale:  0.9475539824570391
RMSE on log scale:  0.1620881941121294
RMSE on original $ scale:  147193.13596905538
```

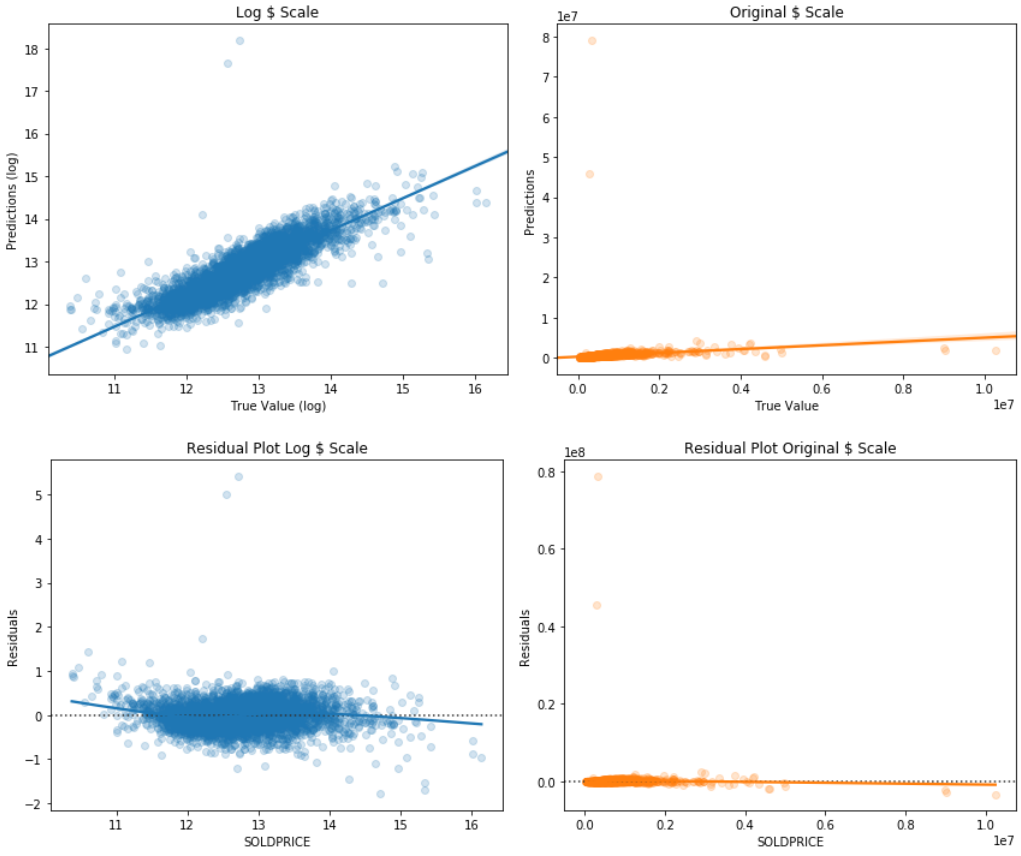


Ridge - Zillow + Redfin data, with images (as 40 PCA components)

In [16]:

```
# Ridge
train_features = X_train_img
val_features = X_val_img
# model = RidgeCV(alphas=(1, 1.5, 2, 2.5, 3))
filename = '../data/models/zillow-redfin_img_ridge.pkl'

print("Ridge model: ")

# train and save model
# train_save_model(model, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
Ridge model:
RidgeCV(alphas=(1, 1.5, 2, 2.5, 3), cv=None, fit_intercept=True,
    gcv_mode=None, normalize=False, scoring=None, store_cv_values=False)
best alpha:  1.0
----- Training scores -----
R2 on log scale:  0.7347720512297995
RMSE on log scale:  0.36807678446348874
RMSE on original $ scale:  527325.3530370378
----- Validation scores -----
R2 on log scale:  0.7304057963806038
RMSE on log scale:  0.36749412530999487
RMSE on original $ scale:  1460744.732897762
```

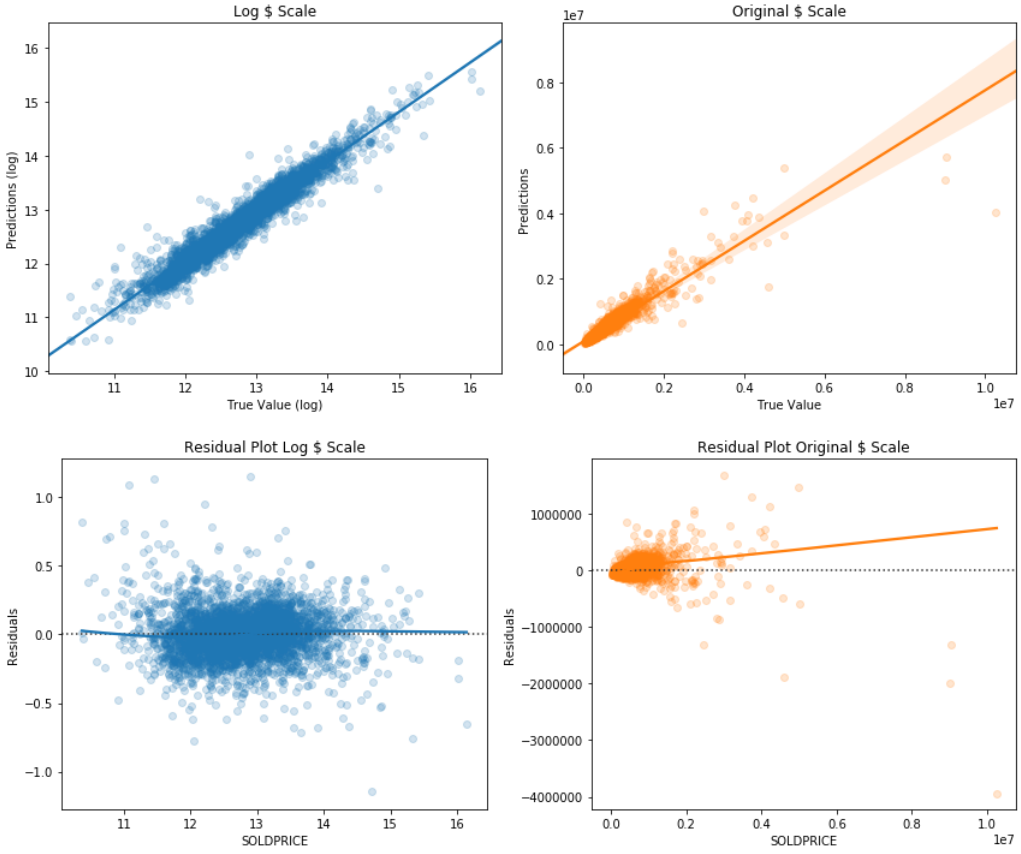


XGBoost - Zillow + Redfin data, with images (as 40 PCA components)

In [18]:

```
train_features = X_train_img
val_features = X_val_img
filename = '../data/models/zillow-redfin_img_XGBoost.pkl'
params = {
    'max_depth':range(21,28,2),
    'gamma':[i/10.0 for i in range(0,3)],
    'reg_alpha':[1e-2, 0.1, 1, 10]
}

# model
# model = XGBRegressor(random_seed=9001)
# grid = GridSearchCV(model, params, verbose=1, n_jobs=-1)

print("XGBoost model: ")

# train and save model
# train_save_model(grid, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
XGBoost model:
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0.1, learning_rate=0.1, max_delta_step=0,
       max_depth=15, min_child_weight=1, missing=nan, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_seed=9001,
       random_state=0, reg_alpha=0.01, reg_lambda=1, scale_pos_weight=1,
       seed=None, silent=True, subsample=1)
----- Training scores -----
R2 on log scale:  0.9827072176769607
RMSE on log scale:  0.09398557880591102
RMSE on original $ scale:  79199.83873206531
----- Validation scores -----
R2 on log scale:  0.9362575438618631
RMSE on log scale:  0.17869381571877585
RMSE on original $ scale:  182360.26115872202
```

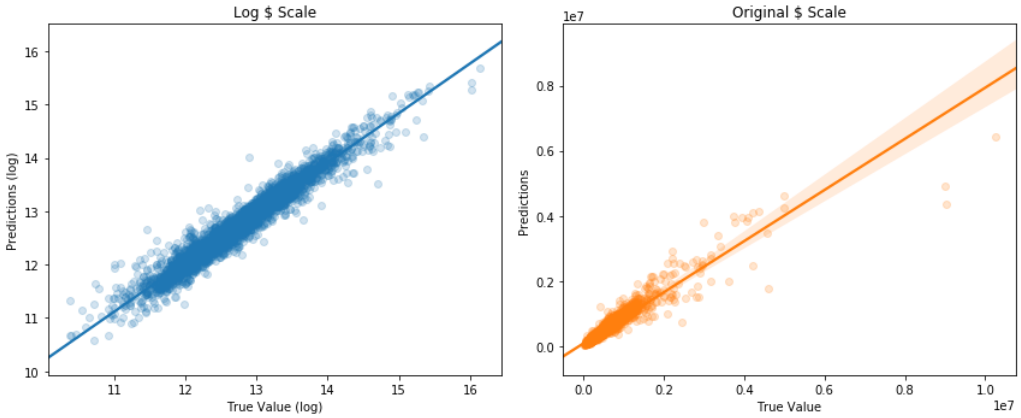


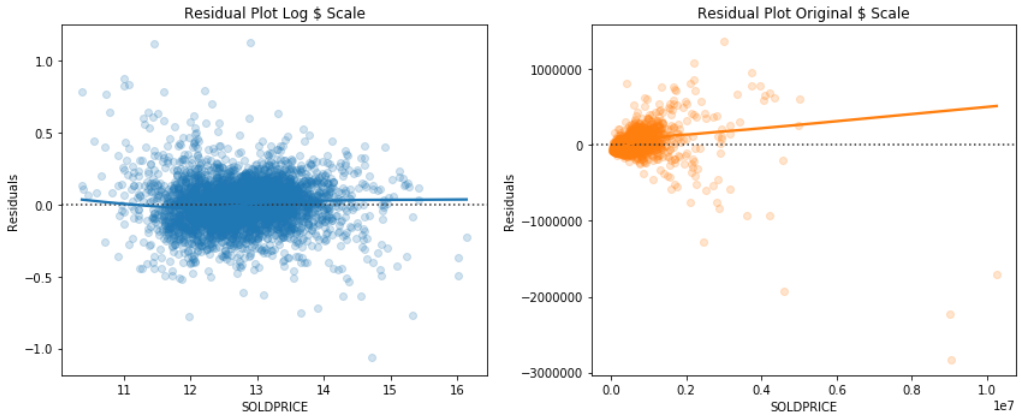
LightGBM - Zillow + Redfin data, with images (as 40 PCA components)

In [19]:

```
train_features = X_train_img
val_features = X_val_img
filename = '../data/models/zillow-redfin_img_LGBM.pkl'
params = {'num_leaves': [15, 30, 100],
          'max_depth':[-1, 8, 16, 32],
          'learning_rate':[0.01, 0.1, 1],
          'n_estimators':[512, 1024, 2048]
         }

# model = LGBMRegressor(random_state=9001)
# grid = GridSearchCV(model, params, verbose=1)
print("LightGBM model: ")

# # train and save model
# train_save_model(grid, X=train_features, y=y_train_all, filename=filename)

# load savced model and evaluate model performance
load_eval_model(filename=filename, X_train=train_features, X_val=val_features, y_train=y_train_all, y_val=y_val_all)
```

```
LightGBM model:
LGBMRegressor(boosting_type='gbdt', class_weight=None, colsample_bytree=1.0,
       learning_rate=0.01, max_depth=-1, min_child_samples=20,
       min_child_weight=0.001, min_split_gain=0.0, n_estimators=2048,
       n_jobs=-1, num_leaves=100, objective=None, random_state=9001,
       reg_alpha=0.0, reg_lambda=0.0, silent=True, subsample=1.0,
       subsample_for_bin=200000, subsample_freq=1)
----- Training scores -----
R2 on log scale:  0.9721911203031899
RMSE on log scale:  0.11918470603484396
RMSE on original $ scale:  107664.2847901731
----- Validation scores -----
R2 on log scale:  0.9421342114852634
RMSE on log scale:  0.1702574260427202
RMSE on original $ scale:  169534.0383353532
```





In [ ]:

```
1
```