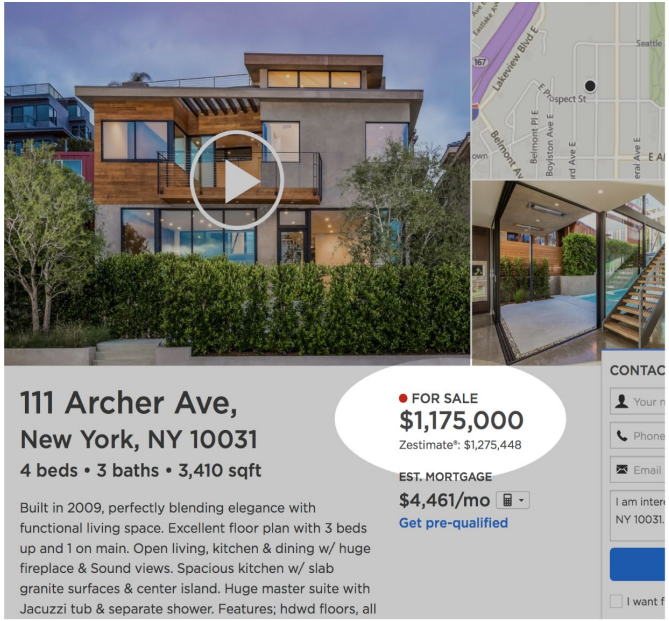
Zillow Real Estate Price Estimate

Claire Jiaying Wu / DEC 2018

Contents

Introduction

- **Background**: Zillow's "Zestimate" -- home valuation in the U.S. real estate industry
- **Project Objectives**: increasing the home valuation accuracy primarily by improving the residual error
- **Approaches**: leveraging statistical and machine learning models with data preprocessing, model selection, parameter tuning and cross validation
- **Metrics:** Mean Absolute Error (MAE) between the predicted log error & the actual log error, log error = log (Zestimate) – log (Sale Price)



Data Preprocessing

R

Drop Invalid Columns

Calculate the percentage of the missing values in each column.

Remove the columns with too many NAs (with over 80% missing values).

Correct Data types

Convert the data types for several columns to match the feature of the variables:

Character -> Logical
E.g. 'Hashottuborspa', 'fireplaceflag'

Numeric -> Character
E.g. 'regionidcity', 'yearbuilt'

Impute Missing Values

Impute the missing values upon below logic:

Numeric data -> impute by median
Categorical -> impute by mode
Logical -> impute by opposite

Drop Outliers

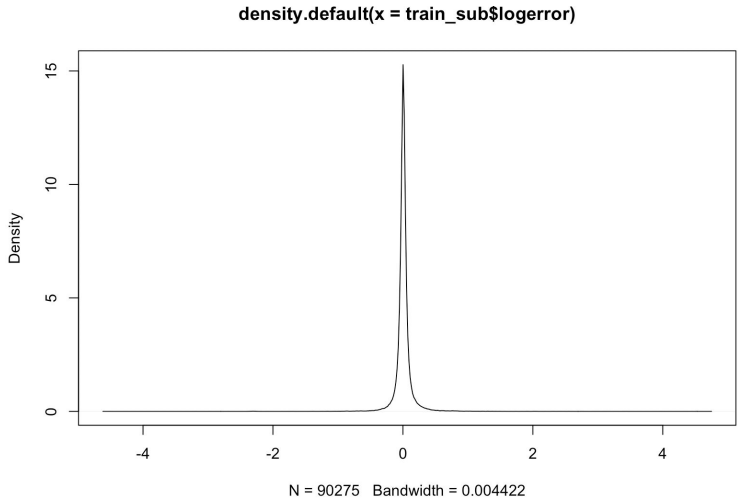
Drop the outliers:

log error values > 0.418;
log error values < -0.417

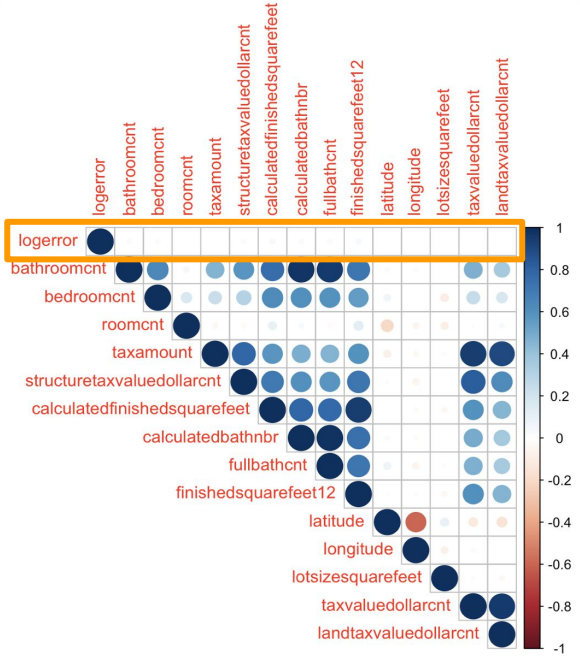
Exploratory Data Analysis

R

- **Numeric Variables**



Density plot of log error

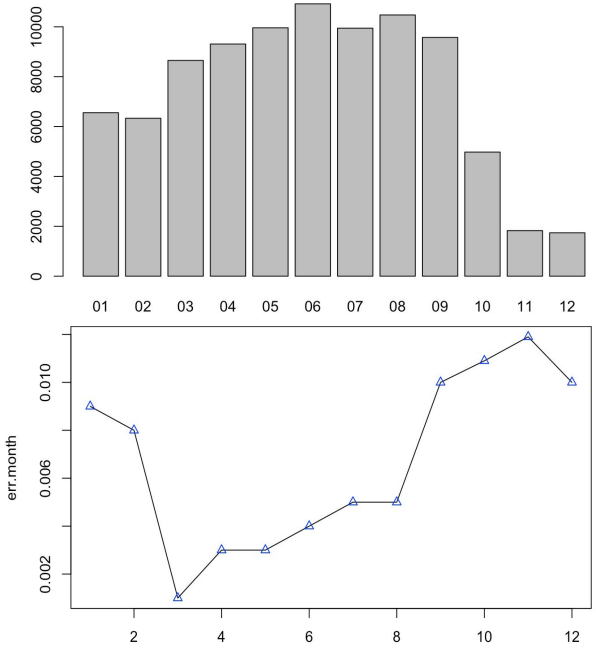


correlation plot of numeric variables

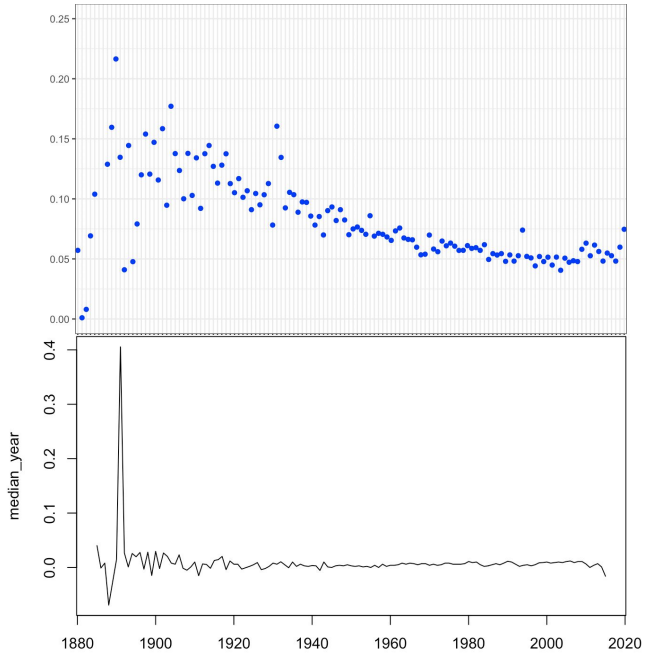
Exploratory Data Analysis

R

- **Categorical Variables**



*By transaction month**



By built year

Modeling and Interpretation Python

Step 1: Benchmark Setting

```
#Applying RandomForestRegressor
clf = RandomForestRegressor()
scores = cross_validation.cross_val_score(clf, X, df.logerror, cv=10, scoring='neg_mean_absolute_e
rror')
print("%s Accuracy: %0.5f" % (clf.__class__.__name__, scores.mean()))
print'-----------------------------------------------------'
```

```
RandomForestRegressor Accuracy: -0.08261
-----------------------------------------------------
```

Step 2: Feature Engineering

```
#proportion of living area
train['New_LivingAreaProp'] = train['calculatedfinishedsquarefeet']/train['lotsizesquarefeet']

#Number of properties in the zip
zip_count = train['regionidzip'].value_counts().to_dict()
train['New_zip_count'] = train['regionidzip'].map(zip_count)

#Number of properties in the city
city_count = train['regionidcity'].value_counts().to_dict()
train['New_city_count'] = train['regionidcity'].map(city_count)
```

Modeling and Interpretation

Python

Step 3: Model Selection - 5 out of 1 - XGBoost

```
Returning the best model
Model: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
Loss: -0.0530361127108
```

```
print("Running XGB Regression")
model = XGBRegressor()
predicted = cross_val_score(model, train, test, scoring='neg_mean_absolute_error', cv=10)
allModels[model] = predicted.mean()
print("Running Ridge Regression")
model = linear_model.Ridge()
predicted = cross_val_score(model, train, test, scoring='neg_mean_absolute_error', cv=10)
allModels[model] = predicted.mean()
print("Running Lasso Regression")
model = linear_model.Lasso()
predicted = cross_val_score(model, train, test, scoring='neg_mean_absolute_error', cv=10)
allModels[model] = predicted.mean()
print("Running Random Forest")
model = RandomForestRegressor()
predicted = cross_val_score(model, train, test, scoring='neg_mean_absolute_error', cv=10)
allModels[model] = predicted.mean()
print("Running Gradient Boosted Regressor")
model = GradientBoostingRegressor()
predicted = cross_val_score(model, train, test, scoring='neg_mean_absolute_error', cv=10)
allModels[model] = predicted.mean()
```

Model & Score Obtained

1. XGB: -0.05303611
2. Gradient Boosted:-0.05309042
3. Lasso: -0.05335322
4. Ridge: -0.05338599
5. Random Forest: -0.05949888

Modeling and Interpretation Python

Step 4: Model Tuning - GridSearchCV

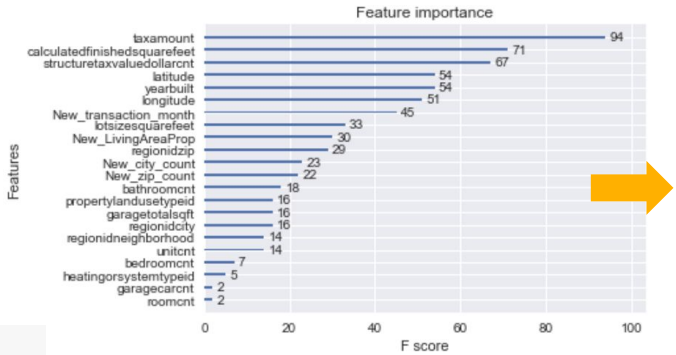
```
params = {
            "n_estimators": [10,100,200,1000],
            "max_depth": [3,5,7],
            "min_child_weight": [1,3,5]
         }

xgb_model = XGBRegressor()
xgb_cv    = GridSearchCV(xgb_model,params,verbose = 10,scoring='neg_mean_absolute_error')
xgb_cv.fit(train, test)
```

```
Best Score- -0.053170955805
Best Parameters- {'n_estimators': 100, 'max_depth': 3, 'min_child_weight': 5}
Best Estimator- XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=5, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

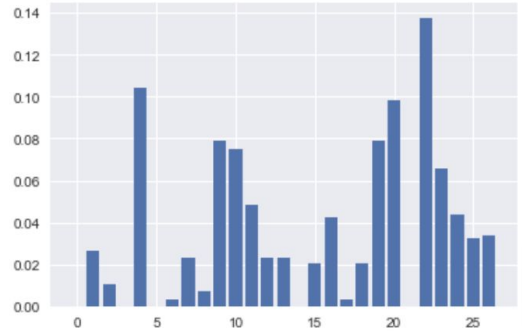
Modeling and Interpretation Python

Step 5: Best Score & Feature Importance

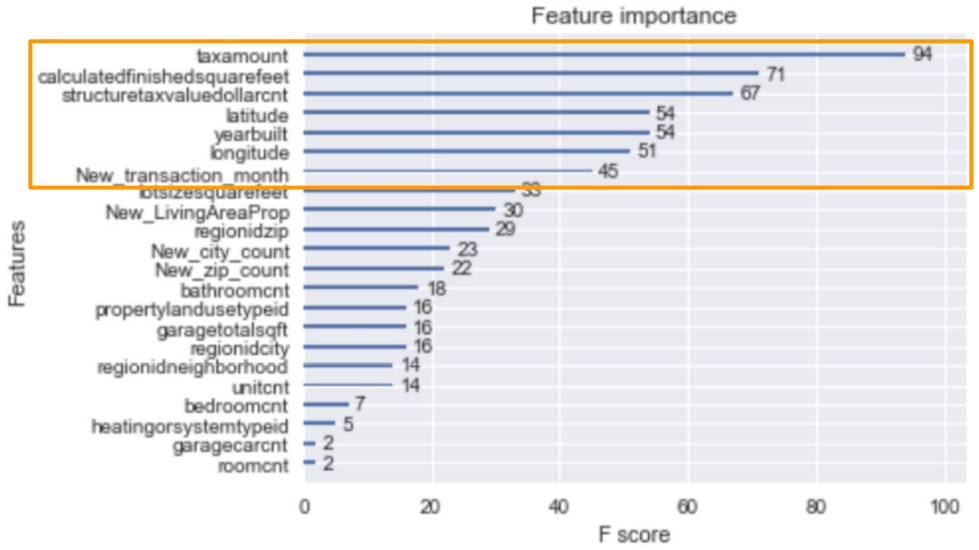
```
Cross validation again using best estimated parameters
Printing Final Score
MAE: 0.05 (+/- 0.00)
Printing Feature Importances
[ 0.          0.02635432  0.0102489   0.          0.10395315  0.
  0.00292826  0.02342606  0.00732064  0.07906296  0.07467057  0.04831625
  0.02342606  0.02342606  0.          0.0204978   0.04245974  0.00292826
  0.0204978   0.07906296  0.09809663  0.          0.13762811  0.0658858
  0.04392387  0.03221083  0.03367496]
```




```
#Importing xgboost
from xgboost import XGBRegressor
from xgboost import plot_importance

print("Cross validation again using best estimated parameters")
clf = XGBRegressor(learning_rate =0.1,max_depth= 3,min_child_weight=1,n_estimators=100)
scores = cross_validation.cross_val_score(clf, train, test, cv=10,  scoring='neg_mean_absolute_err
or')
print("Printing Final Score")
print("MAE: %0.2f (+/- %0.2f)" % (abs(scores.mean()), scores.std()*2))

clf.fit(train, test)
print("Printing Feature Importances")
print clf.feature_importances_
plt.bar(range(len(clf.feature_importances_)),clf.feature_importances_)
plt.show()

plot_importance(clf)
plt.show()
```



Feature Importance



Conclusion

Result

- Benchmark MAE: 0.082
- XGBoost MAE: 0.053
- XGBoost tuned MAE: 0.050

Future Improvement

- Impute missing value: KNN
- Models: LightGBM & CatBoost
- Feature engineering: more new features generated from domain knowledge

THANKS!

Any questions?