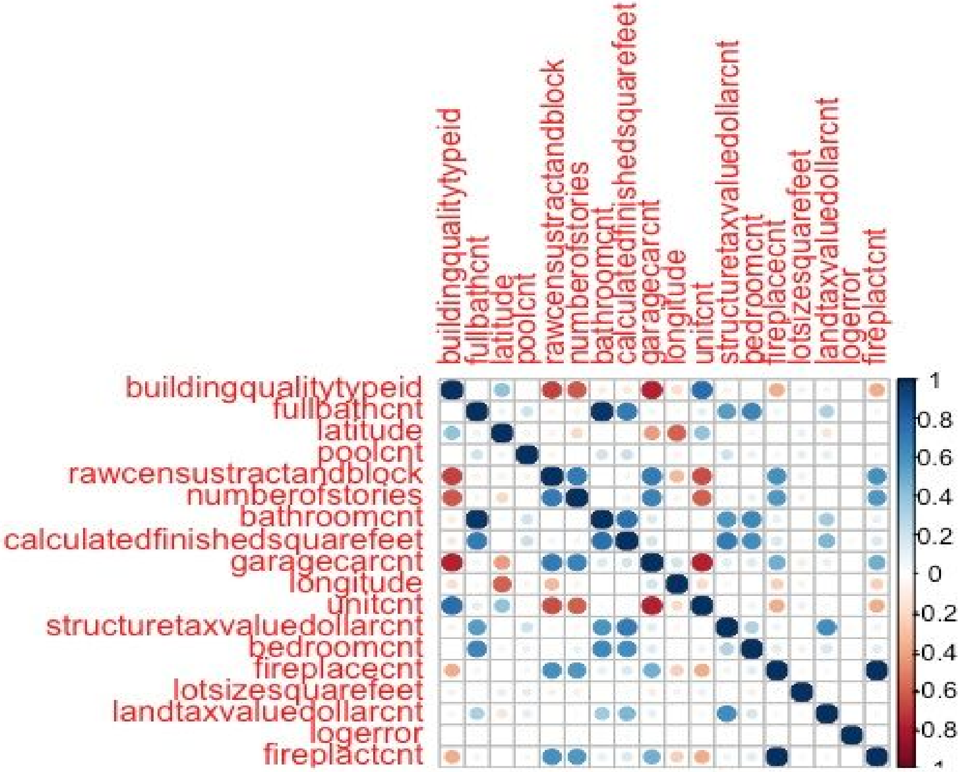
Best Team Ever

Zillow Kaggle Competition

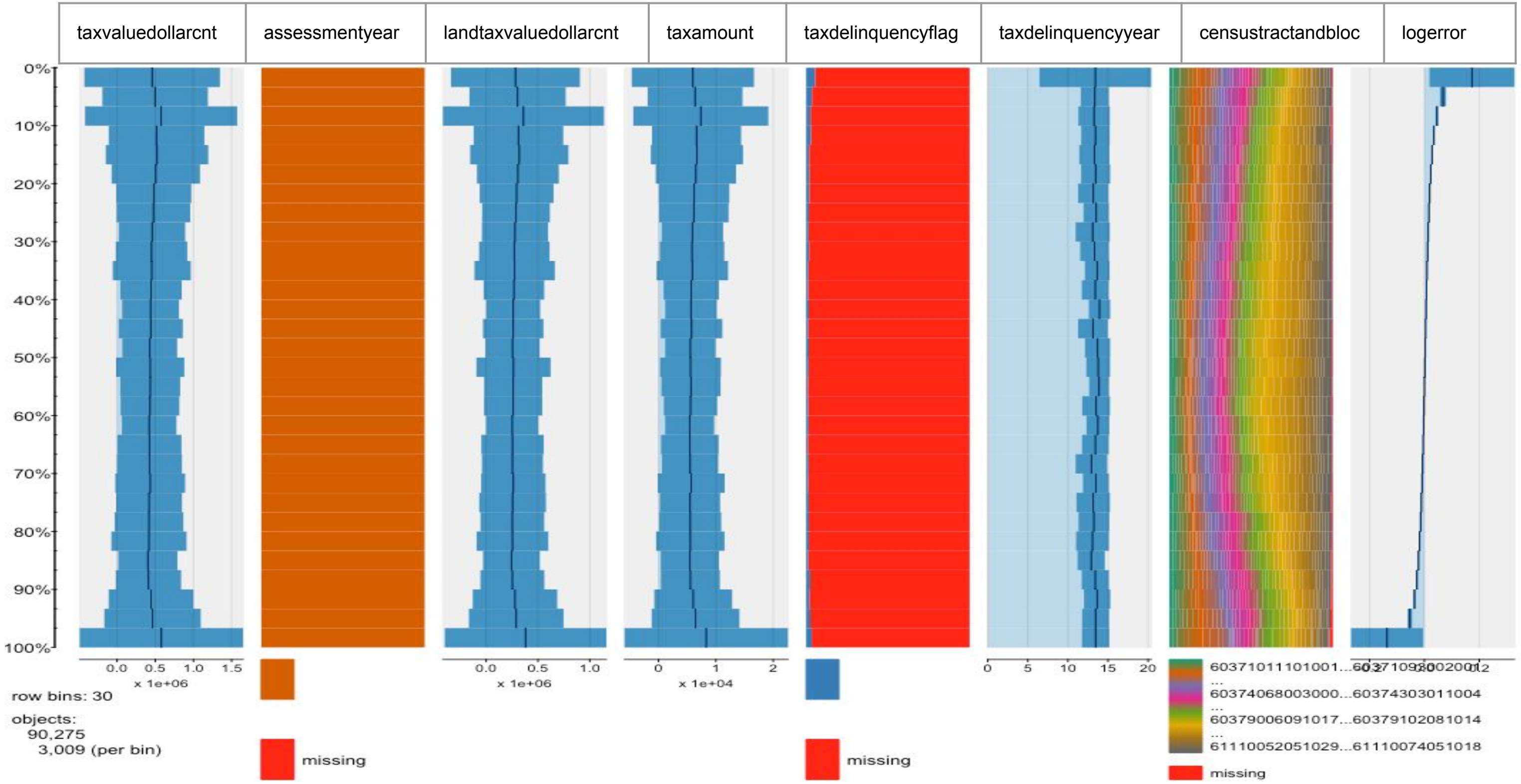
Patrick Masi-Phelps | Katie Critelli
Ningxi Xu | John Merrick

EDA: Correlation of Numeric Variables



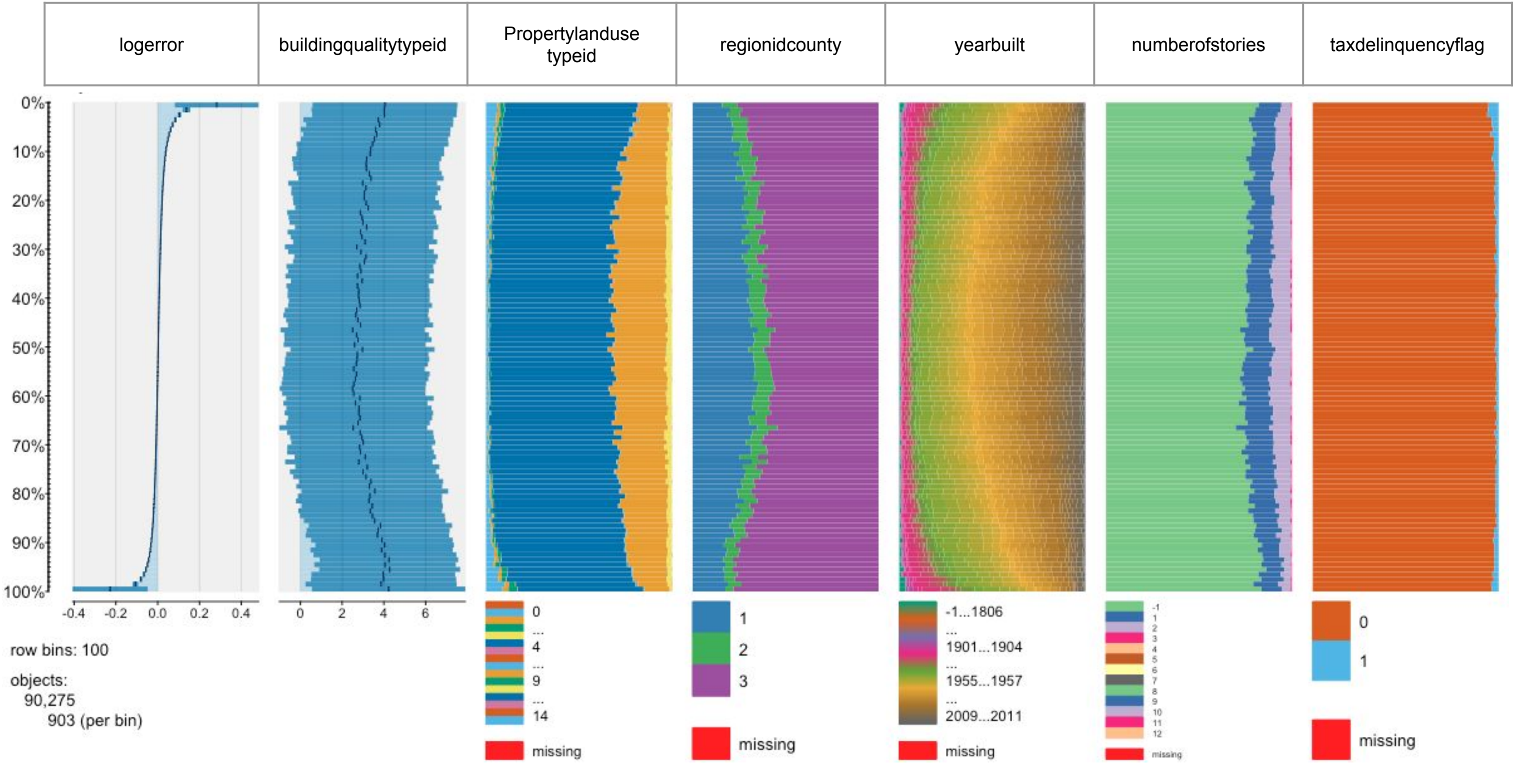
EDA: Tableplot Visualizations

- Predictor values sorted by logerror



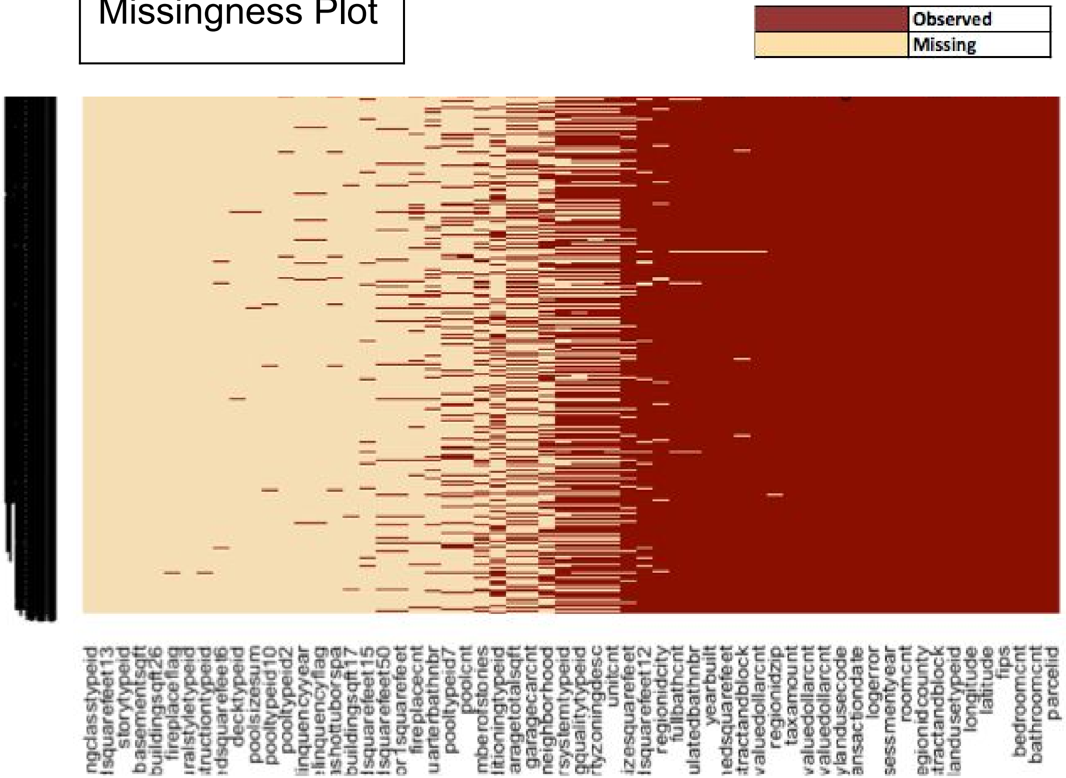
EDA: Tableplot Visualizations

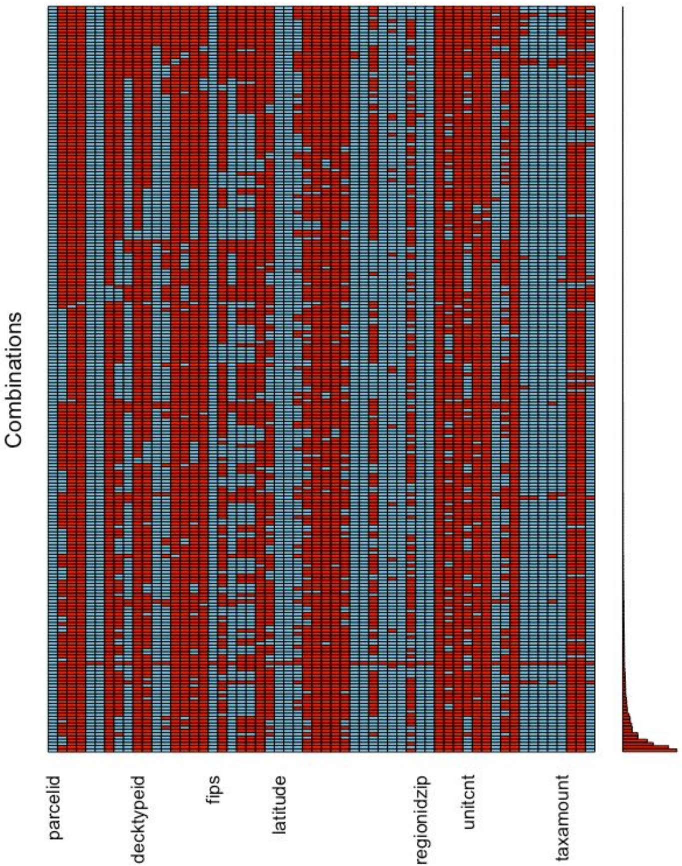
- Predictor values sorted by logerror



EDA: Visualization of Missingness

Missingness Plot





Data Cleaning and Imputation

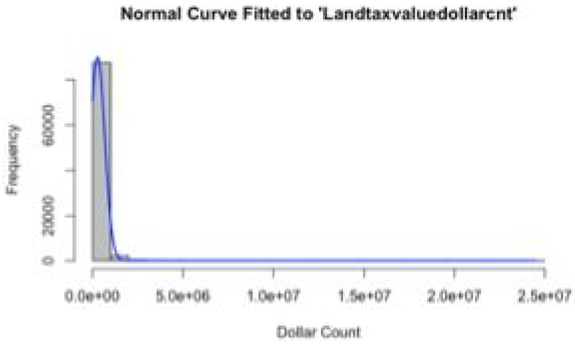
Data Cleaning and Imputation

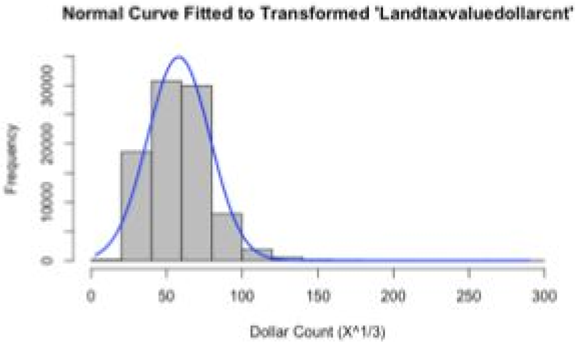
- Does the property contain the item classified by the variable?
  - Assumed NA means this property does not contain the item: imputed NAs with 0
  - Taxdelinquencyyear, taxdelinquencyflag, lotsizesquarefeet, fireplacecnt, garagecarcnt, garagetotalsqft, poolcnt, pooltypeid7, pooltypeid2, poolsizesum, basementsqft, threequarterbathnbr, hashottuborspa
  - Imputed heatingorsystemtypeid with 13 and airconditioningtypeid with 5 (equivalent to none)
- Reason for NA is unknown: imputed with -1 per Shu's lecture code
  - Preserving any reasons for missingness that could be inherent in these particular observations
  - Unitcnt, bedroomcnt, bathroomcnt, fullbathcnt, buildingqualitytypeid, numberofstories, yearbuilt
- Other methods
  - Calculatedfinishedsquarefeet: mean imputation
    - Dropped other floor space area variables as redundant/overly missing
  - Region variables: random imputation
    - Small number of missing data points

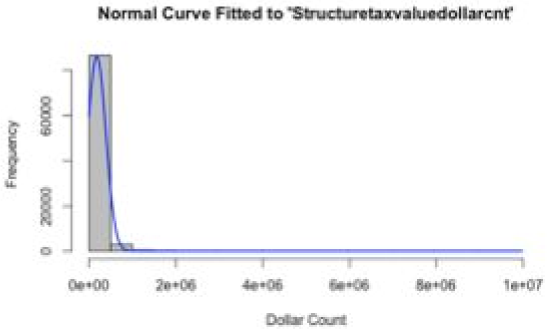
Data Cleaning and Imputation

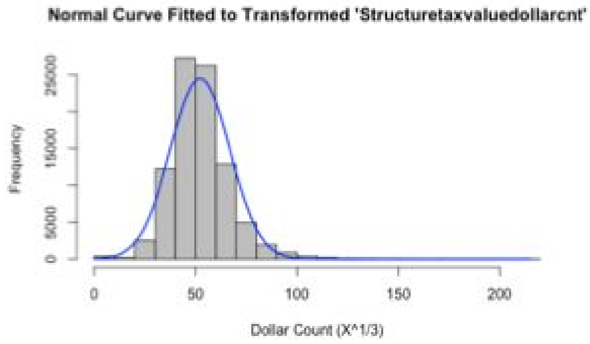
- Missing tax building and land assessment values
  - Examine property taxes paid
    - **23,887/23,902** observations had zero bathrooms and zero bedrooms (not missing)
      - **Likely no building**
    - Divide property taxes paid by median property tax rate (across all properties)
      - Impute this value for land assessment value
      - Impute zero for building assessment value
  - For properties with no taxation values
    - Impute average building and land assessment values
      - Group by zip code, num bed, num bath

Data Transformations



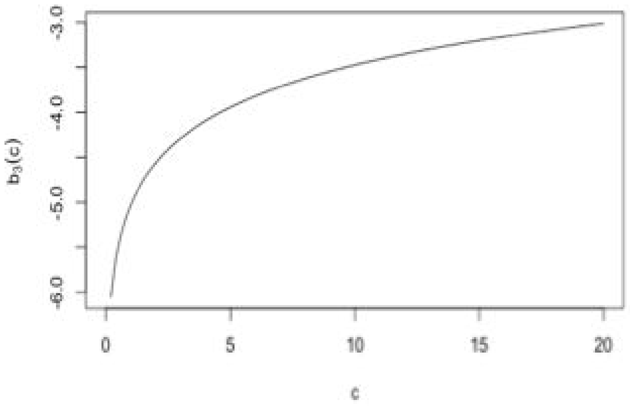


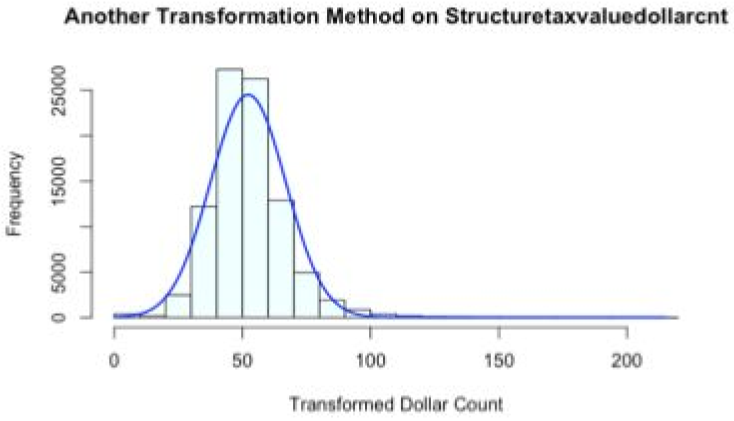




Data Transformations

- Flexible technique
  - Measured skewness of each variable
  - Found a constant C for which the skewness of the transformed function, log(X + C), was minimized





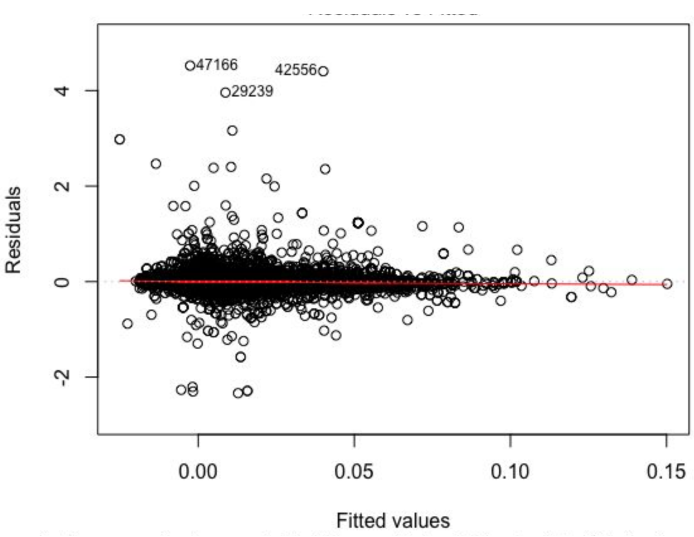
Models

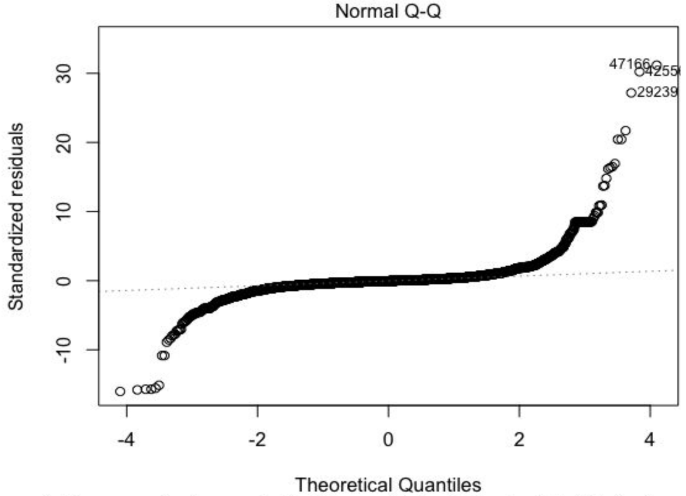
GLM process trial and error

- **Fitting lr1**: logerror ~ bathroomcnt + bedroomcnt + buildingqualitytypeid + calculatedfinishedsquarefeet + fireplacecnt + fullbathcnt + garagecarcnt + lotsizesquarefeet + poolcnt + taxdelinquencyflag + structuretaxvaluedollarcnt + unitcnt + yearbuilt, numberofstories + landtaxvaluedollarcnt
    - **Result:** poor predictive power and several features with high VIFs (including garagecarcnt, poolcnt, etc)

- **Remedy #1:** Fit lr2, leaving out features with both high VIFs and high p-values
    - **Result:** R-squared still low, violated linear model assumptions (see selected graphs)

- **Remedy #2:** Fit lr3, discarding additional features likely to be multicollinear with others





OLS to GLM with coefficient shrinkage

- R-squared still extremely low in lr3 after dropping majority of features (0.002606):
  - OLS likely not optimal model, because linear model assumptions all appeared to be in violation
  - P-value of F-test and all VIFs are significant
  - We do not believe that there is a legitimate linear relationship between logerror and variables
- As a result, we **fit a GLM model with regularization**
  - Fit 100 ridge and lasso models to improve model accuracy
  - Used lambda that yielded the lowest MSE
  - Reduced features used to only 6
  - Significant shrinkage in coefficients

Multiple Linear Regression with Regularization

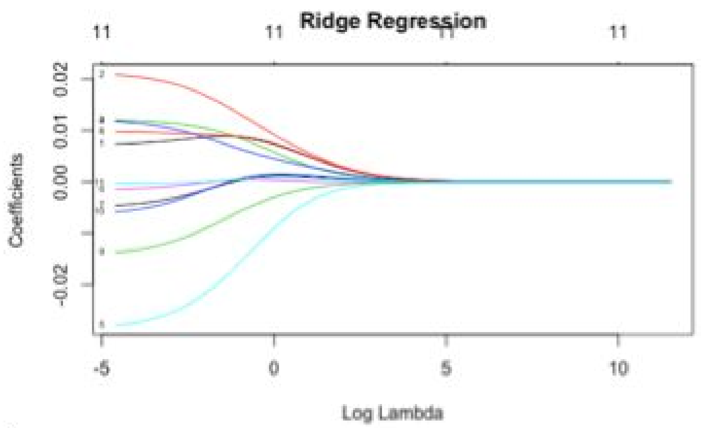
Backward AIC Results Summary

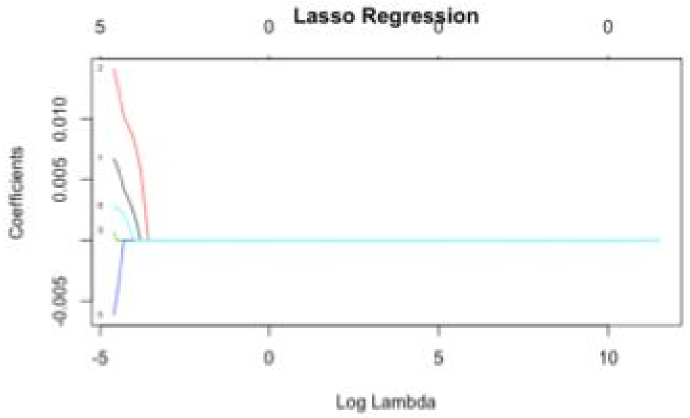
```
Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                        3.433e-15  3.326e-03   0.000 1.000000
calculatedfinishedsquarefeet.scaled 2.219e-02  5.519e-03   4.020 5.82e-05 ***
lotsizesquarefeet.scaled           1.209e-02  3.535e-03   3.420 0.000627 ***
structuretaxvaluedollarcnt.scaled  1.408e-02  5.095e-03   2.764 0.005717 **
landtaxvaluedollarcnt.scaled      -2.759e-02  4.197e-03  -6.574 4.94e-11 ***
bedroomcnt.scaled                  1.139e-02  4.537e-03   2.510 0.012084 *
unitcnt.scaled                    -1.060e-02  3.521e-03  -3.011 0.002607 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9992 on 90268 degrees of freedom
Multiple R-squared:  0.001577,	Adjusted R-squared:  0.001511
F-statistic: 23.76 on 6 and 90268 DF,  p-value: < 2.2e-16
```

Ridge and Lasso Regression Plots

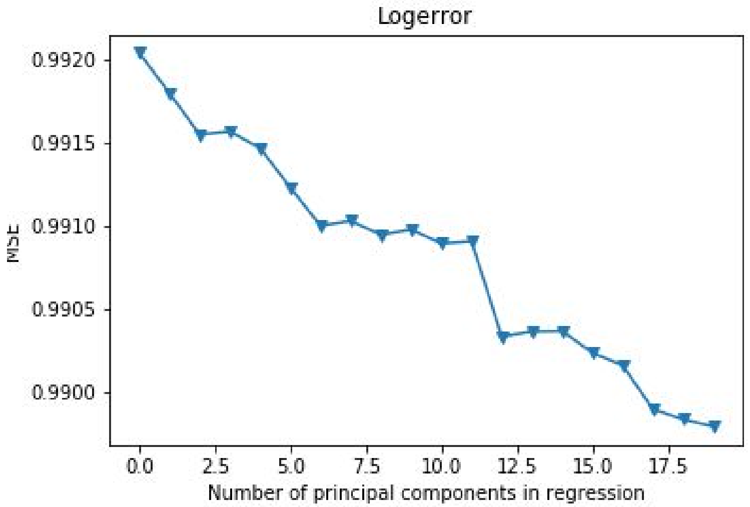
- Lasso Regression Variable Importance (lambda = .2595)
  - calculatedfinishedsquarefeet
  - bathroomcnt
  - bedroomcnt
  - landtaxvaluedollarcnt
  - lotsizesquarefeet





Principal Components Regression in Python

- Preparation for PCA
  - Scaled and centered all numeric predictors
  - LinearRegression in Scikitlearn: plotted change in MSE with each added component
  - Smallest MSE at ~17 components
  - Calculated cumulative variance explained by each added component
  - Trained regression model on training subset
  - Predicted logerror on test subset
  - Overall MSE of 1.02



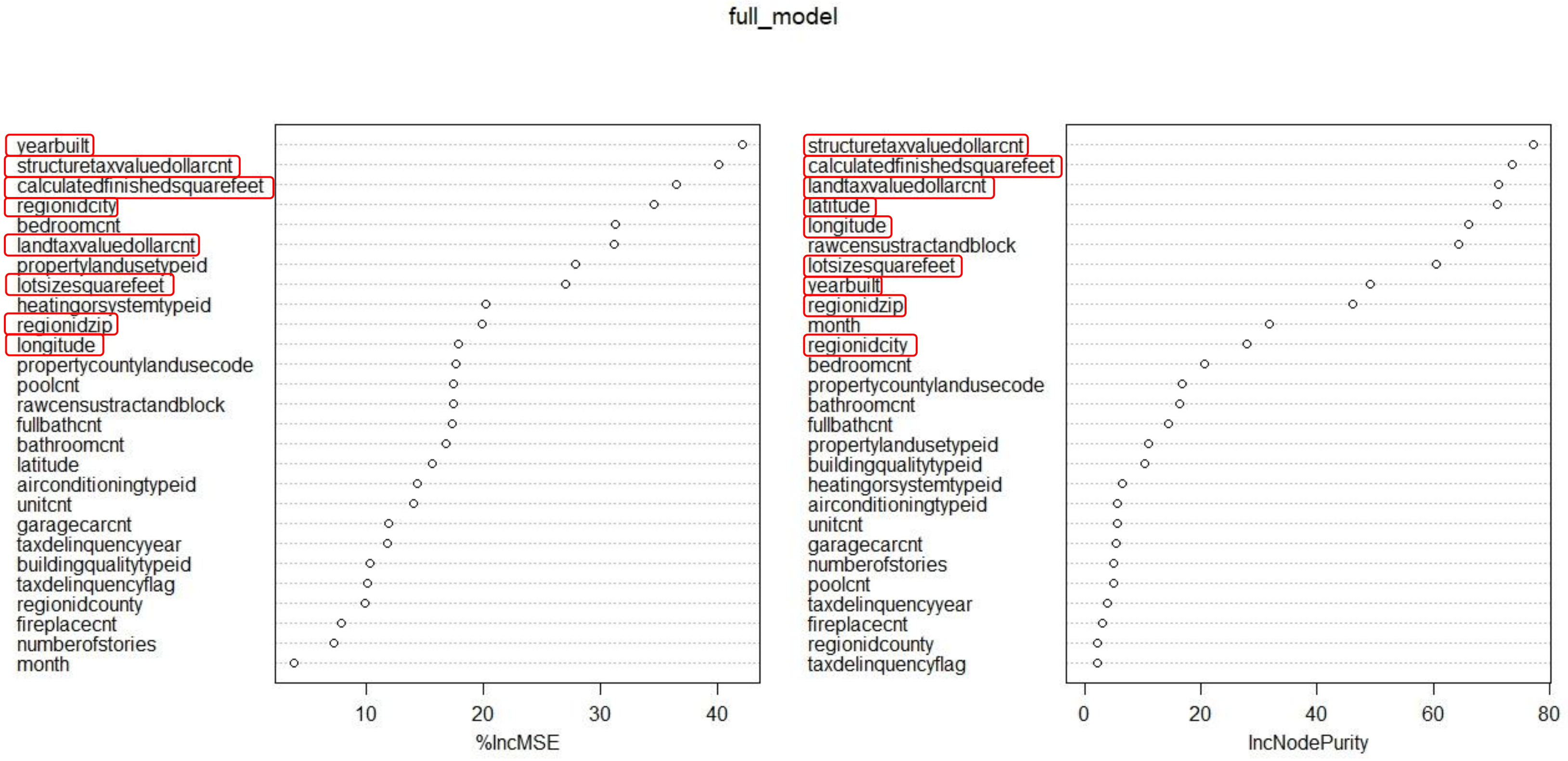
Random Forest: Regression Trees

- Key tuning parameters
  - Number of randomly sampled variables considered at each split: mtry = variables/3
  - Number of trees to grow: ntree = 500
  - Minimal size of terminal nodes: nodesize = 5
  - Maximum terminal nodes (maxnodes): if not limited by nodesize, will grow to max value
- Variable selection
  - Tested with all variables and gradually narrowed, eliminating less important variables
  - Broad model performed best
- Tuning and cross validation
  - Sampled 10%-25% of data set for initial broad grid searches and cross validation
    - High computational complexity of random forest calculations
    - Efficiently narrow down the search for optimal parameters
  - 75%/25% training/test split for precise model tuning

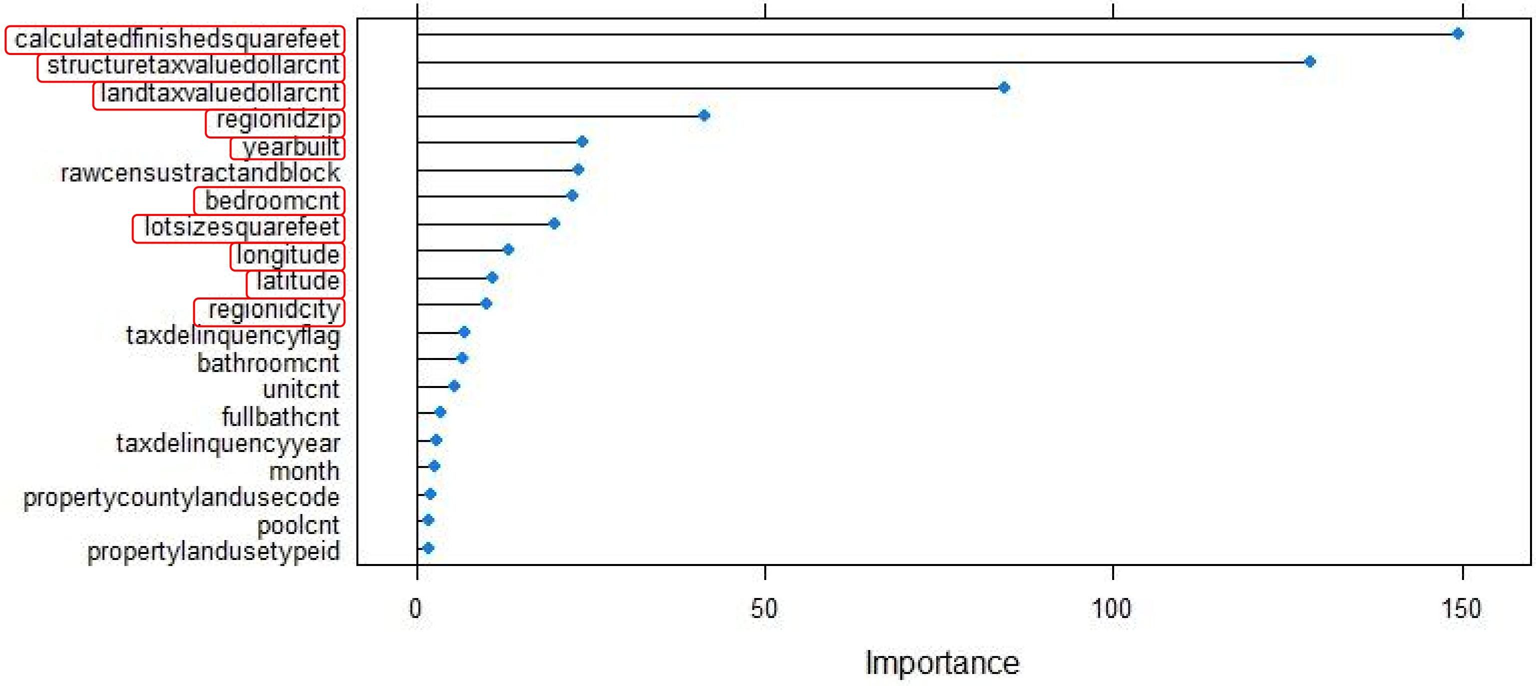
Random Forest: Regression Trees

- Optimal parameters
  - Number of variables considered at each split: 2
  - Number of trees: 1000
  - Minimal terminal node size: 12 observations
- Takeaways
  - Kaggle scoring: final RF model is worse than basic two-variable MLR!
    - MLR (logerror~yearbuilt+calculatedfinishedsquarefeet): 0.0651 MAE
    - RF: 0.0658 MAE
  - Problem in the model, or reflecting the challenges inherent in trying to predict this particular dependent variable/scoring metric?
  - Random forest model useful for assessing variable importance

Variable Importance - Random Forest



Variable Importance - Gradient Boosting



Gradient Boosting Regressor

- Numeric variables
  - Bathroomcnt, Bedroomcnt, Calculatedfinishedsquarefeet, Fireplacecnt, Fullbathcnt, Garagecarcnt, Latitude, Longitude, Lotsizesquarefeet, Poolcnt, Yearbuilt, Numberofstories, Structuretaxvaluedollarcnt, Landtaxvaluedollarcnt
- Categorical variables
  - Airconditioningtypeid, Propertylandusetypeid, Heatingorsystemtypeid, Regionidcounty, Buildingqualitytypeid, Unitcnt
- Tuning and cross validation
  - Used 5-fold grid search CV to minimize mean absolute error
- Optimal hyperparameters
  - Learning rate: .06
  - Maximum features per split: 11
  - Minimum samples split: 600
  - Subsample: .85
  - Max_depth: 11
  - Min samples per leaf: 30
  - Number of estimators: 40
- 80/20 train/test split
  - Mean Absolute Error: 0.0516168

XGBoost

- Variables - same as sklearn gradient boosting regressor (previous slide)
- Regularization, tuning and cross validation
  - Used 5-fold grid search CV to minimize mean absolute error
  - Ridge (MAE = .052879) chosen over Lasso (MAE = .052996) or Elasticnet (α = .5) (MAE = .053169)
- Optimal hyperparameters
  - Learning rate: .06
  - Column sample by tree: .7
  - Max_depth: 7
  - Min child weight: 1
  - Number estimators: 1000
  - Subsample: .85
  - Reg alpha: 0
  - Reg lambda: 1 (Ridge)
- 80/20 train/test split
  - Mean Absolute Error: 0.0528799

Underneath the hood

Regularized Objective Function (one example)

Loss/error function

$$\mathcal{L}(\phi) = \boxed{\sum_i l(\hat{y}_i, y_i)} + \boxed{\sum_k \Omega(f_k)}$$

Complexity penalty function

$$\text{where } \Omega(f) = \gamma T + \boxed{\frac{1}{2}\lambda \|w\|^2}$$

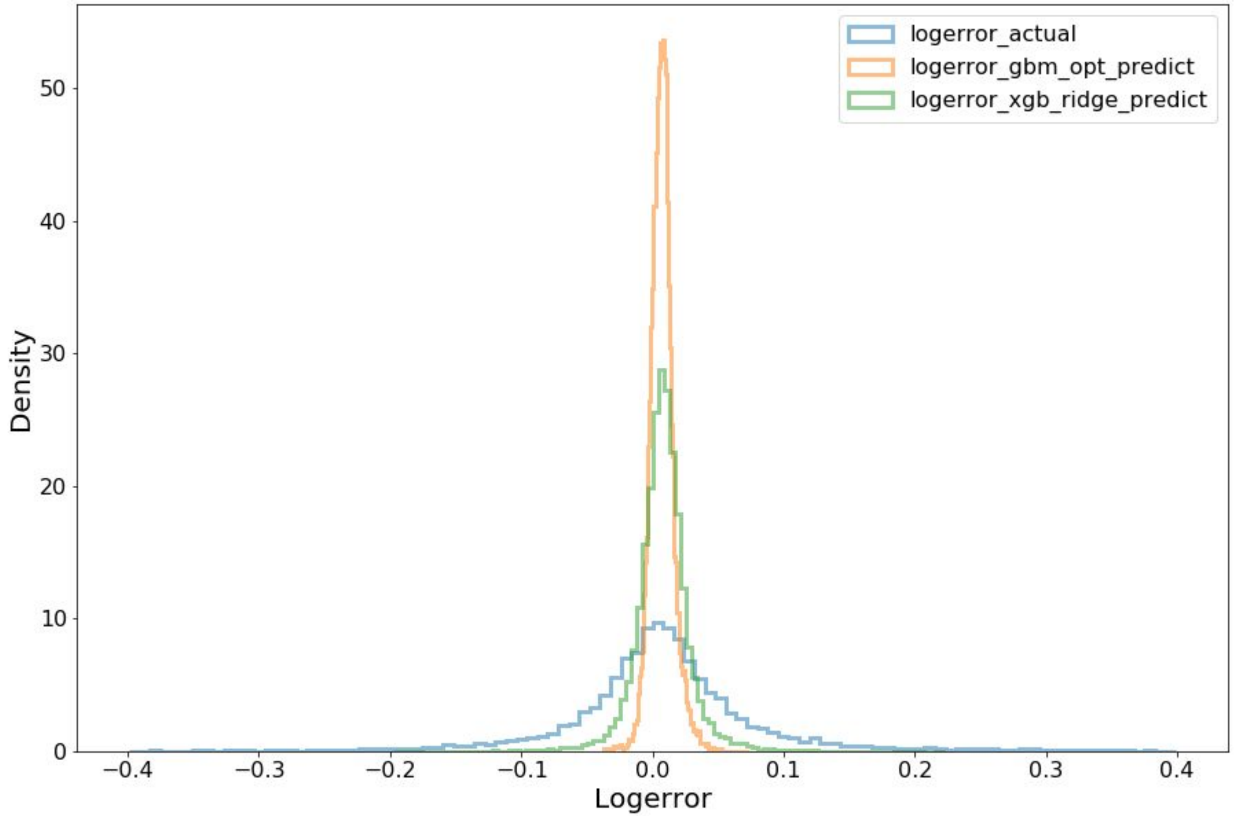
λ = 1 asserts L2 Ridge regression. L1 Lasso and ElasticNet are alternative options

At the t-th iteration...

"Weak learning" trees incorporate estimates from the prior tree

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \boxed{\hat{y}_i^{(t-1)}} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

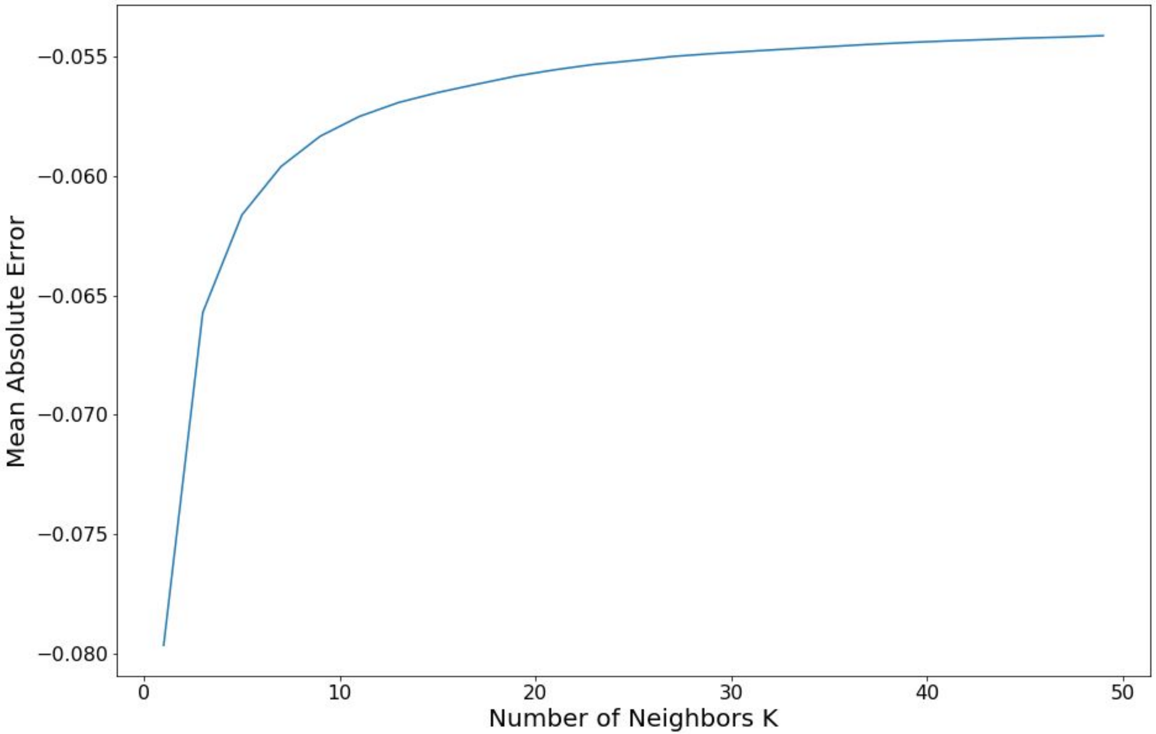
Boosted Models: Logerror Distributions



K-Nearest Neighbors

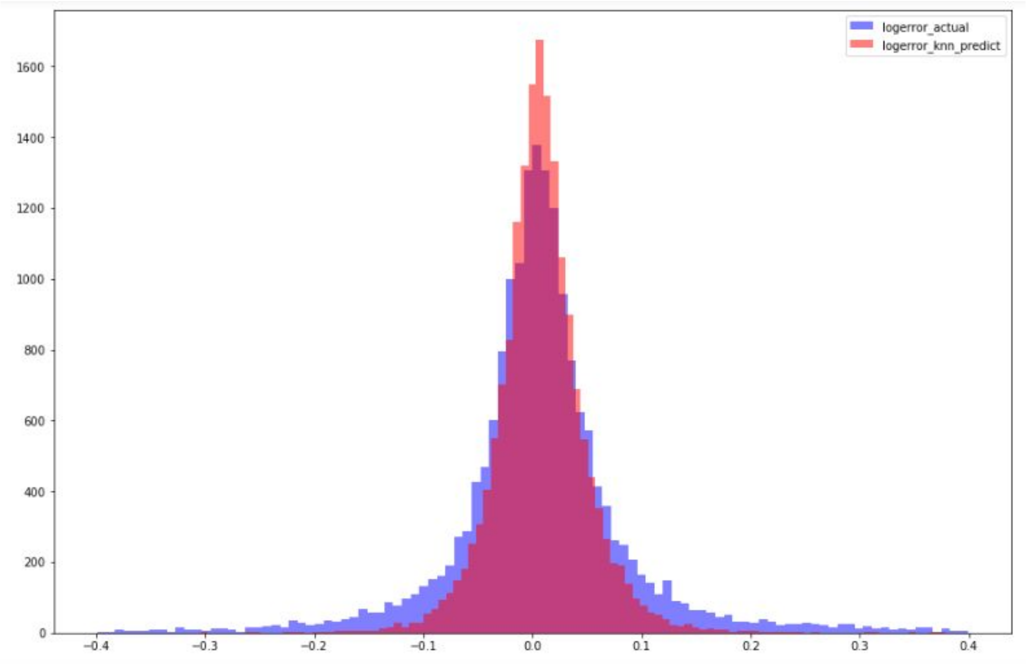
- Tuning and cross validation

- Used 10-fold CV to minimize mean absolute error across K from 1-50



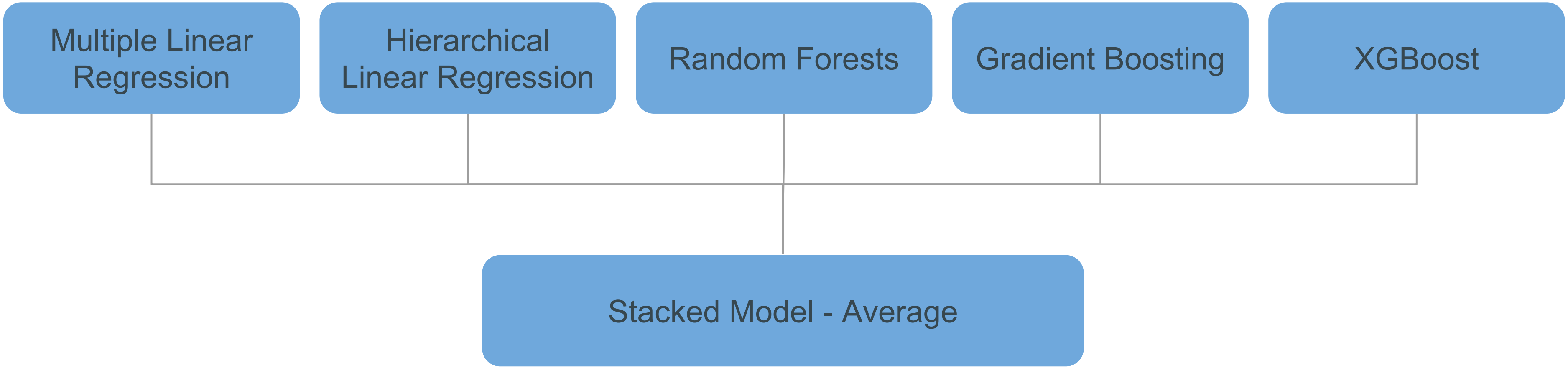
K-Nearest Neighbors

- Optimal hyperparameters:
    - Weights: uniform (rather than distance)
    - K-Neighbors: 10 (near elbow)
    - R^2: .1144
    - Mean Absolute Error: 0.0566785



- Run time on training/test data set (~90,000 rows): ~ 10 minutes
- Run time on full properties data set (~3,000,000 rows): **unknown (7+ hours)**
- **Has potential to be a predictive model - difficult to scale - more processing needed**

Model Stacking - Kaggle Submission



Our stacked model took the average logerror prediction across all models (equal weighting) for each property in the specified months.

Public Kaggle MAE: 0.0646177 - Rank #774

Any Questions?

Appendix: Hierarchical Linear Model

- Feature importance plots suggest that location variables are important
- HLM can highlight in-subgroup variation that is otherwise obscured
  - Treated regionidcity as the level-one subgroup
  - Coefficient of each regionidcity value was used as random variable to estimate a new linear model
  - Selected most important variables using random forest feature importance plot
    - logerror ~ 1 + structuretaxvaluedollarcnt + landtaxvaluedollarcnt + calculatedfinishedsquarefeet + lotsizesquarefeet + bedroomcnt + longitude + latitude
    - groups = train['propertycountylandusecode']
- Didn't perform well independently (MAE of 0.0660), but improved the performance of the stacked Kaggle submission