ZILLOW DATA SCIENCE PIPELINE

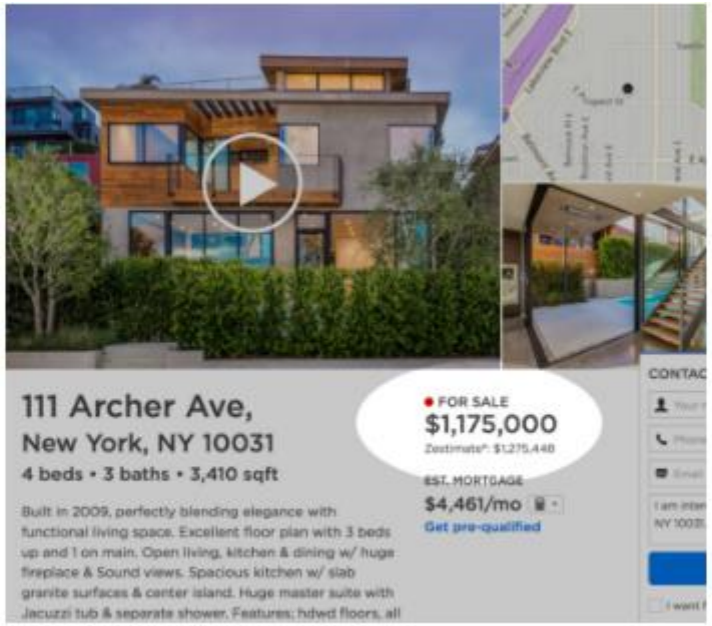
12/8/2017

GLOBAL DIRECT FRANCE

redefining digital insurance

Zillow's Home Value Prediction

Competition description



111 Archer Ave,
New York, NY 10031
4 beds • 3 baths • 3,410 sqft

Built in 2009, perfectly blending elegance with functional living space. Excellent floor plan with 3 beds up and 1 on main. Open living, kitchen & dining w/ huge fireplace & Sound views. Spacious kitchen w/ slab granite surfaces & center island. Huge master suite with Jacuzzi tub & separate shower. Features: hdwd floors, all

FOR SALE
$1,175,000
Zestimate®: $1,275,448

EST. MORTGAGE
$4,461/mo
Get pre-qualified

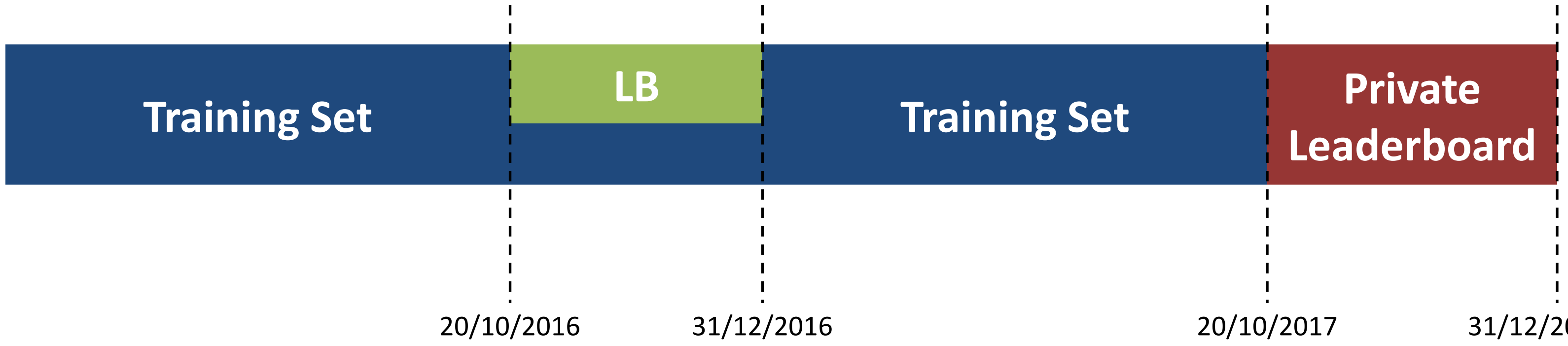
CONTAC

Competition description

- Zillow is an online real estate company that was founded in 2006 by two of the Microsoft executives, it has since become established as one of the largest, most trusted marketplaces for real estate information in the U.S. and a leading example of impactful machine learning.
- The goal of this competition is to complete Zillow's prediction model by predicting its residuals.

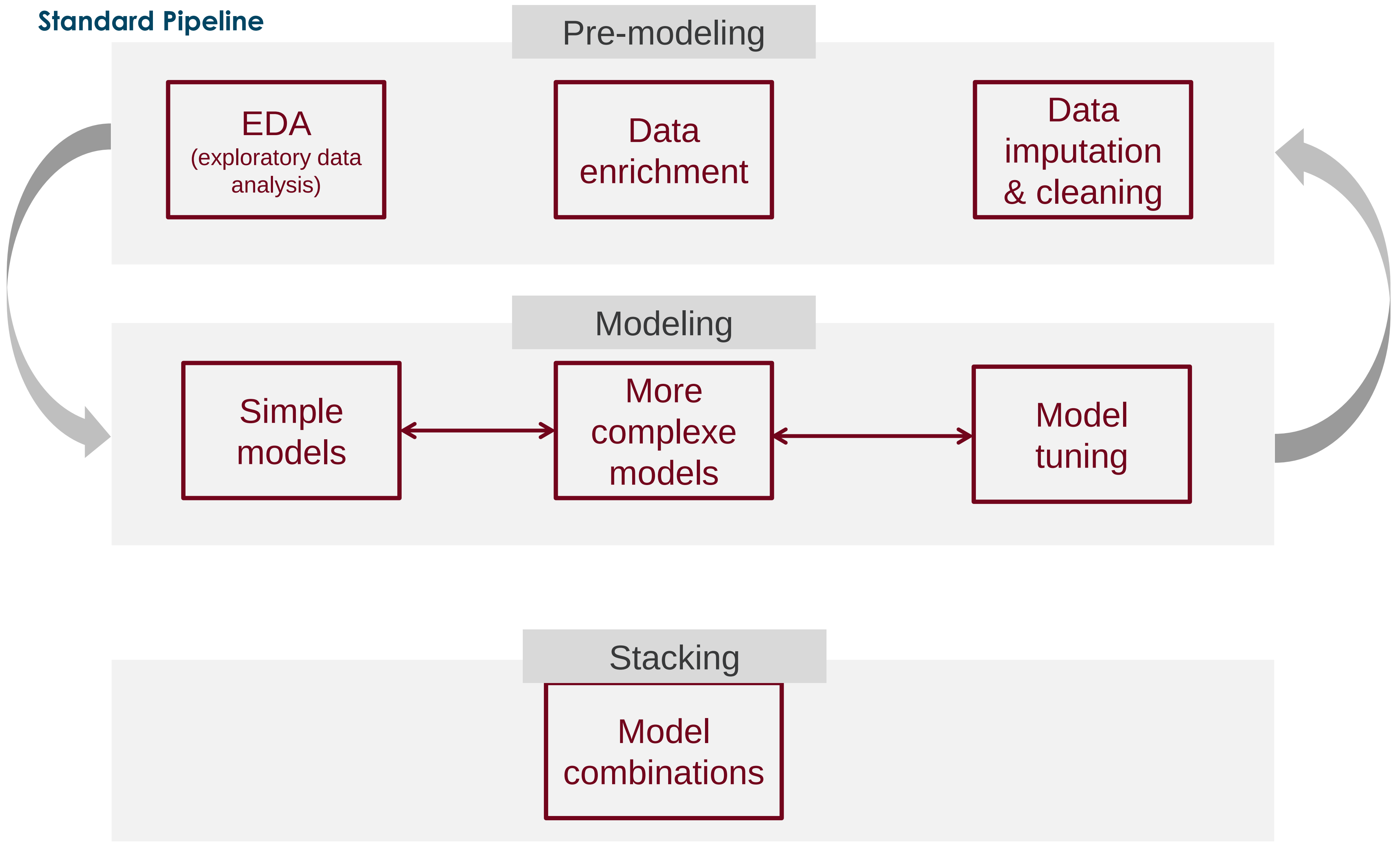
$$\log error = \log(Zestimate) - \log(SalePrice)$$

- Evaluation metric: Mean Absolute Error
- Expected submissions: predictions for the last 3 months of 2017.
- The problem of predicting the error is different from the standard problem because of high signal / noise ratio.



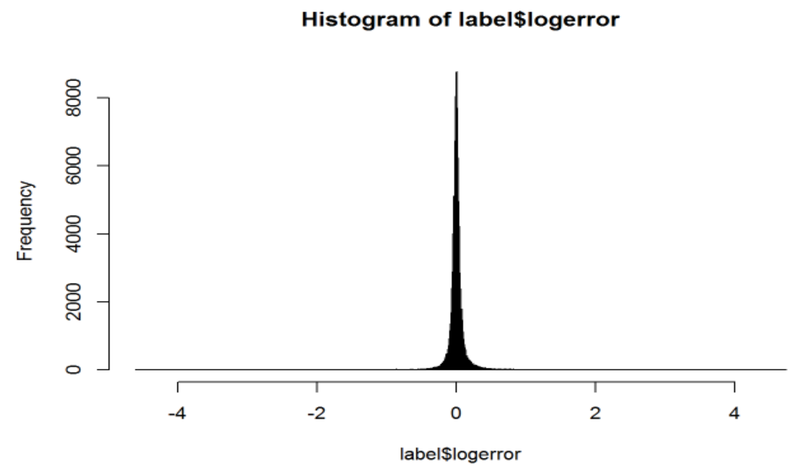
Zillow's Home Value Prediction

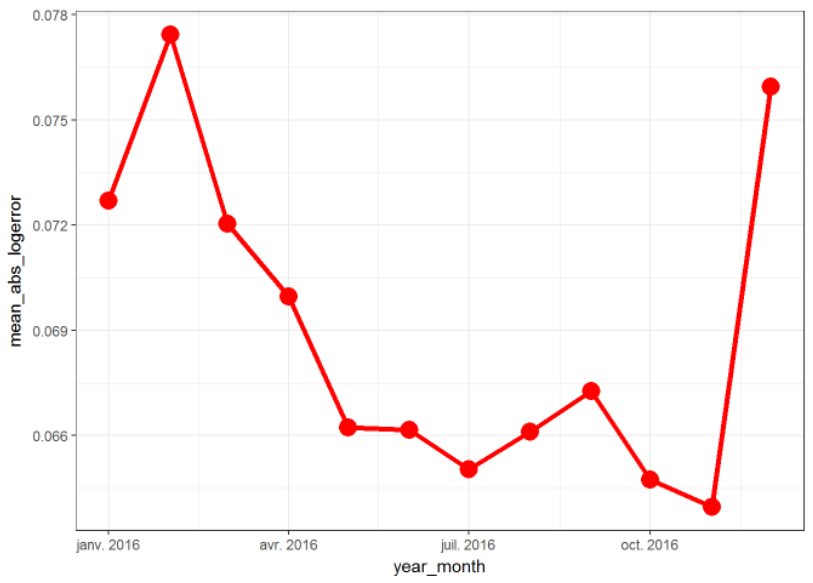
Standard Pipeline



Explolatory Data Analysis

Overview

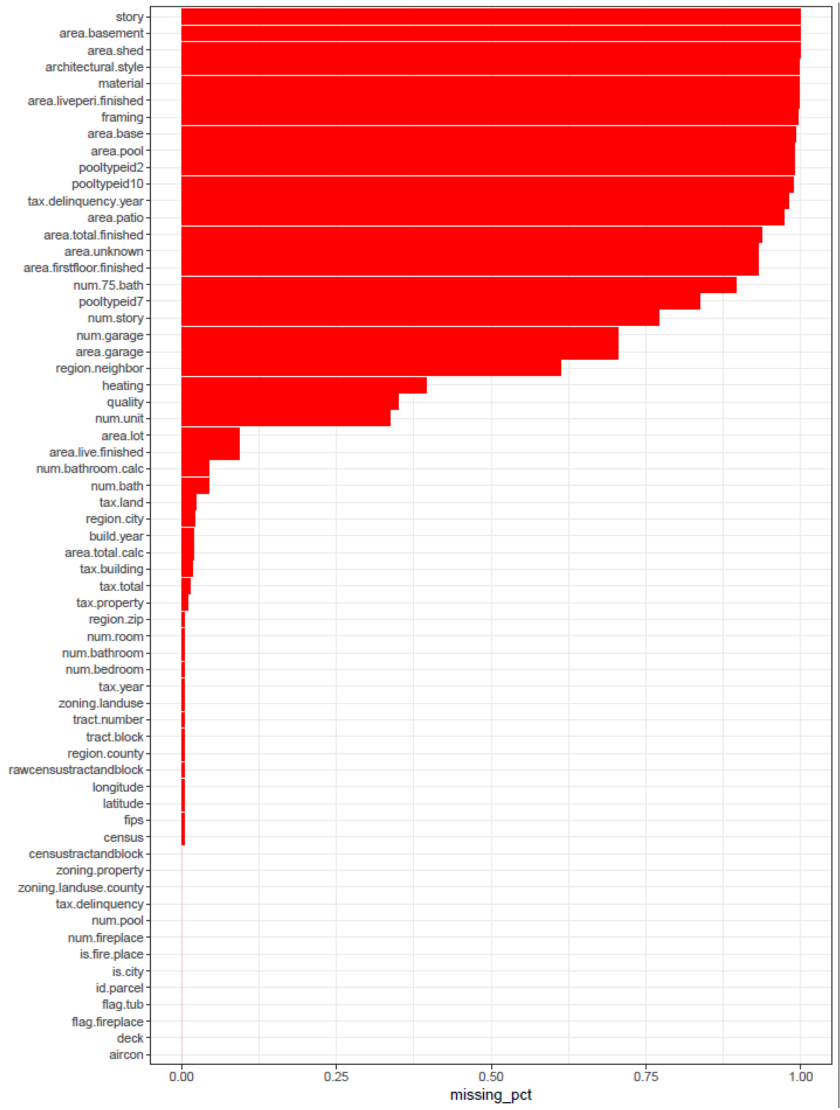




- One can simply put the entire data to a blackbox model and get the results.
- But one can also spend time to understand the data hence get the very first ideas for better modelling.
  - → **Outliers in the response: capping / collaring could be beneficial.**
  - → **Necessarily of making several models.**
  - → **Response variable changes over time -> How to deal with this issue?**
  - → **Try different weights for observations of different months.**

Explolatory Data Analysis

Missing value

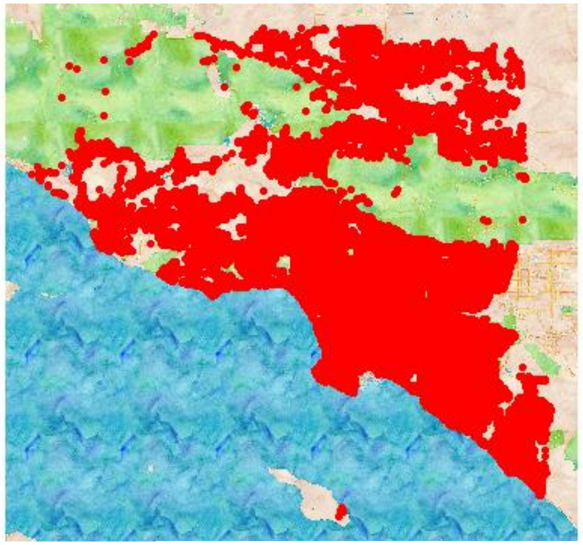


- Remove predictors with more than 98% missing value.
- Data imputation:
  - → **Replace NA by extreme value (-1, -999)**
  - → **Replace NA by the median**
  - → **Build a simple model to predict the missing value**
  - → **Replace by a value that make sense.**

The choice between two methods is done by heuristic.

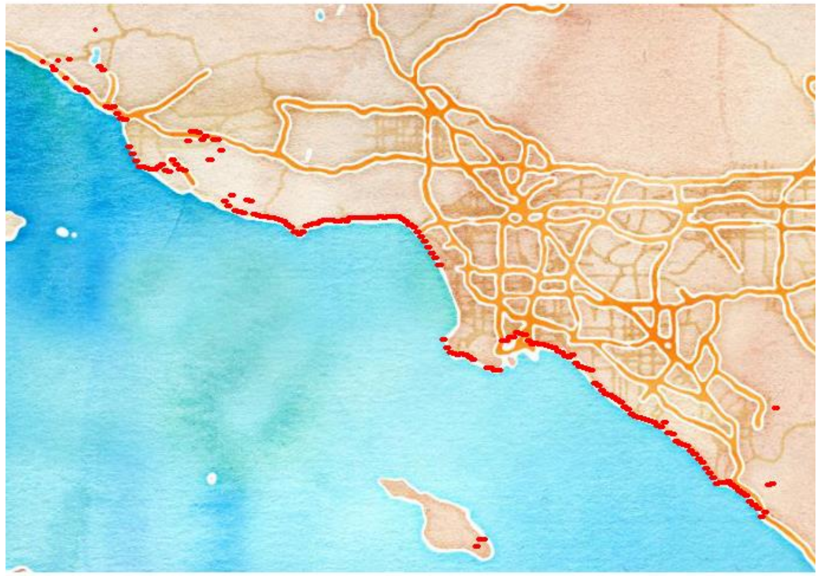
Feature Engineering

Ocean Distance



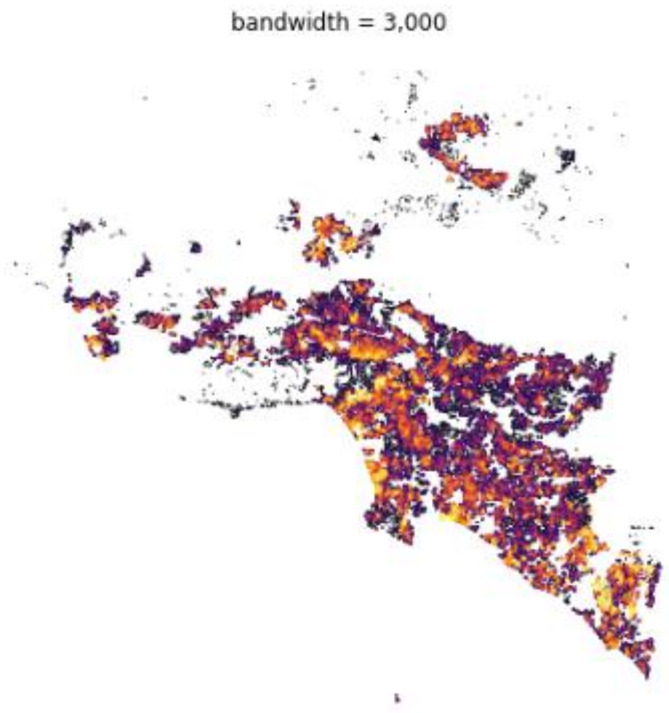
Ocean distance

- The intuition is that the nearer to the coast a house is, the more expensive it becomes.
- External data is not allowed, so we need to calculate the longitude / latitude of the coast from the available data.
- We can take the
  - min of longitude by different groups of latitude
  - Min of the latitude by different group of longitude
  - And combine them to have the desired information without using external data



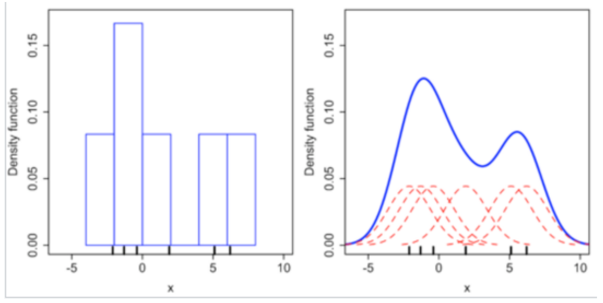
Feature Engineering

Parcel Density



Kernel Density Estimation

- Kernel density is the non-parametric estimation of the probability density function (PDF) of a given data point.
- Example of 1-dimension KDE:



- The data points here are the geographical coordinate of the houses.
- The non-parametic nature of the technique allows us to easily fit the data on a 2-dimension space and on each local segmentation.
- It gives us an additional information of how concentrated the parcel is, independently from the pre-defined geographical zone.

Reference: https://en.wikipedia.org/wiki/Kernel_density_estimation

Modelling

Penalized Regression Models

4.Modelling phase 1

- .Building simple models on the whole data:
  - Lasso regression
  - Ridge regression
  - Elastic net
- Tree bases models:
  - Random Forest
  - Gradient Boosting

With splines

5.Modelling phase 2

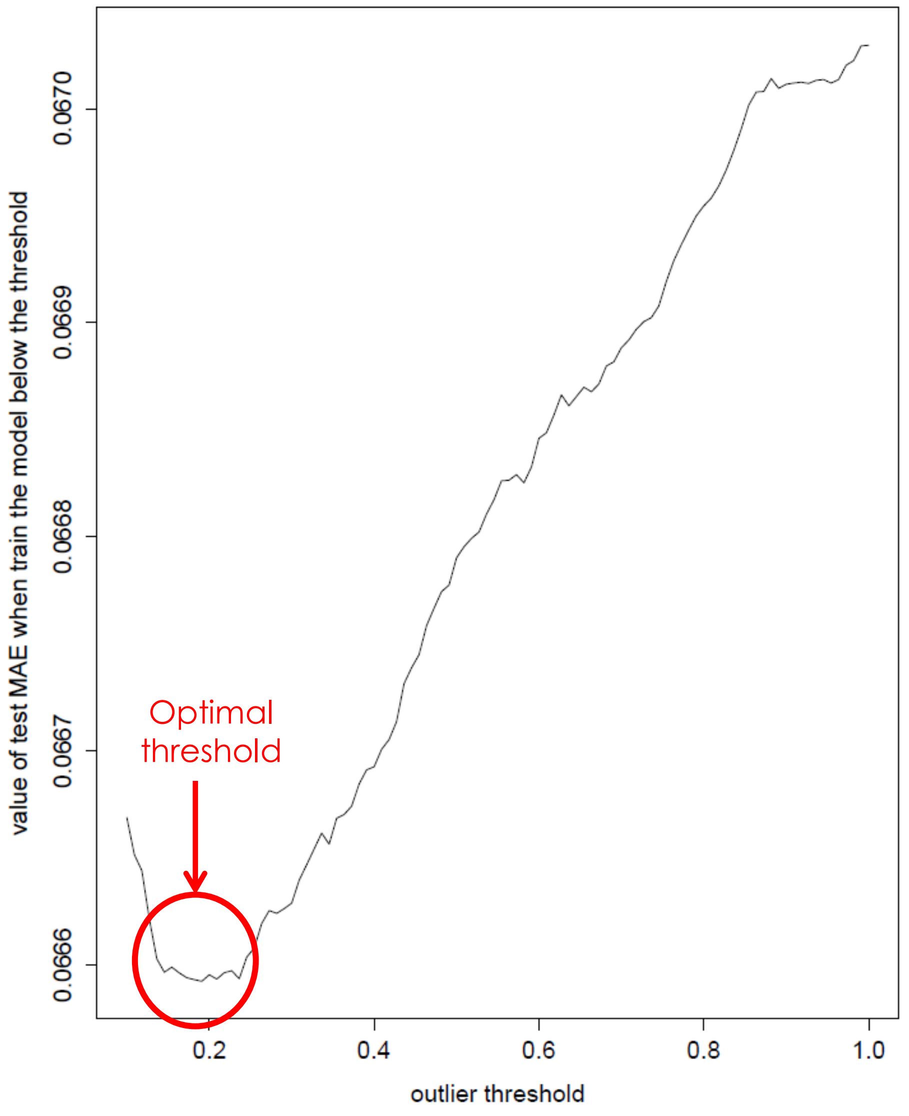
- Intuition: Zillow model already doing well on the traditional variables (not the longitude and latitude).
- So we can:
  - First, fit all the traditional variables to the zestimate residuals to complete Zillow model
  - Fit the second model only on longitude & latitude with some special treatments.

6.Modelling phase 3

- .Building 3 different models for 3 different cities, as the impact of each predictor on the price is not the same between each city.
- Building n models for n clusters.

Some little tricks

Outlier threshold detection

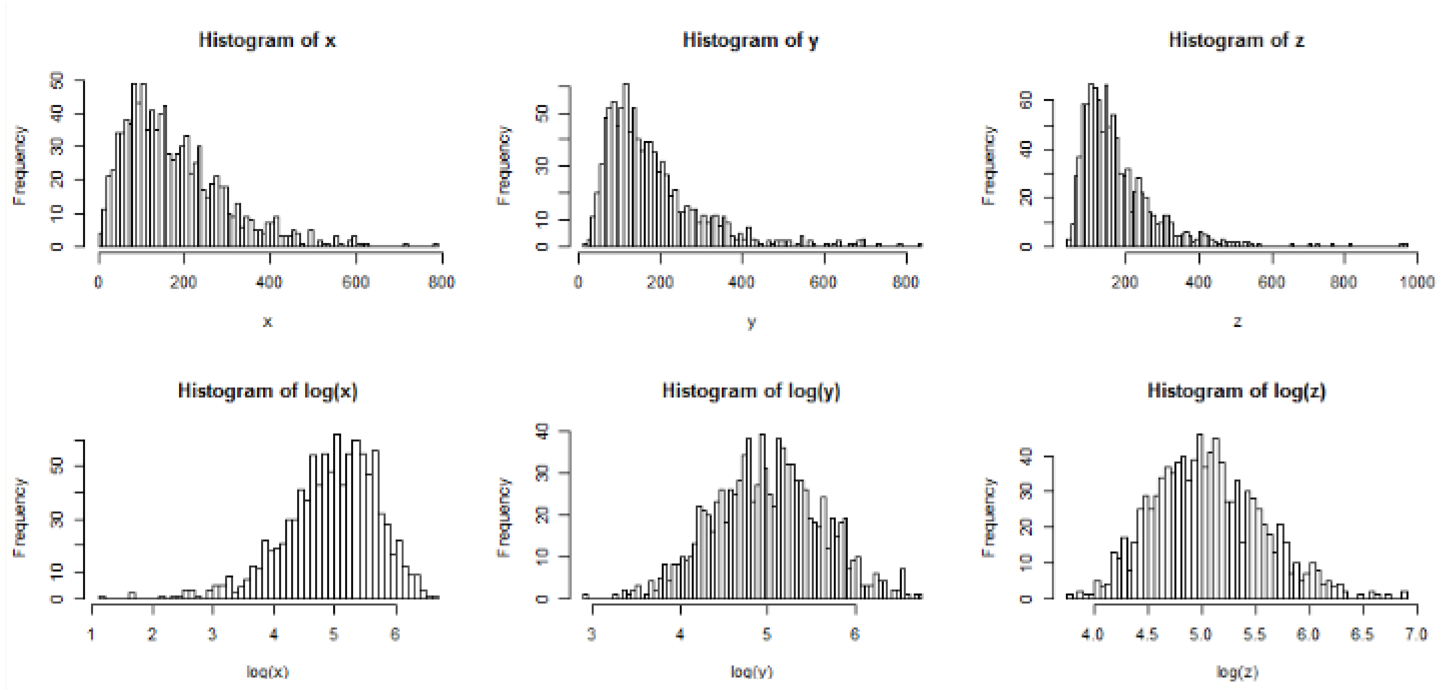


Outlier threshold detection

- Build n regression models on different training set with n levels of outlier threshold.
- Measure the their performance on an independent testing set without threshold.
- The model's performance is strongly impacted by the choice of this threshold as shown in the graphics.

Some little tricks

log-transformations of right-skewed variables

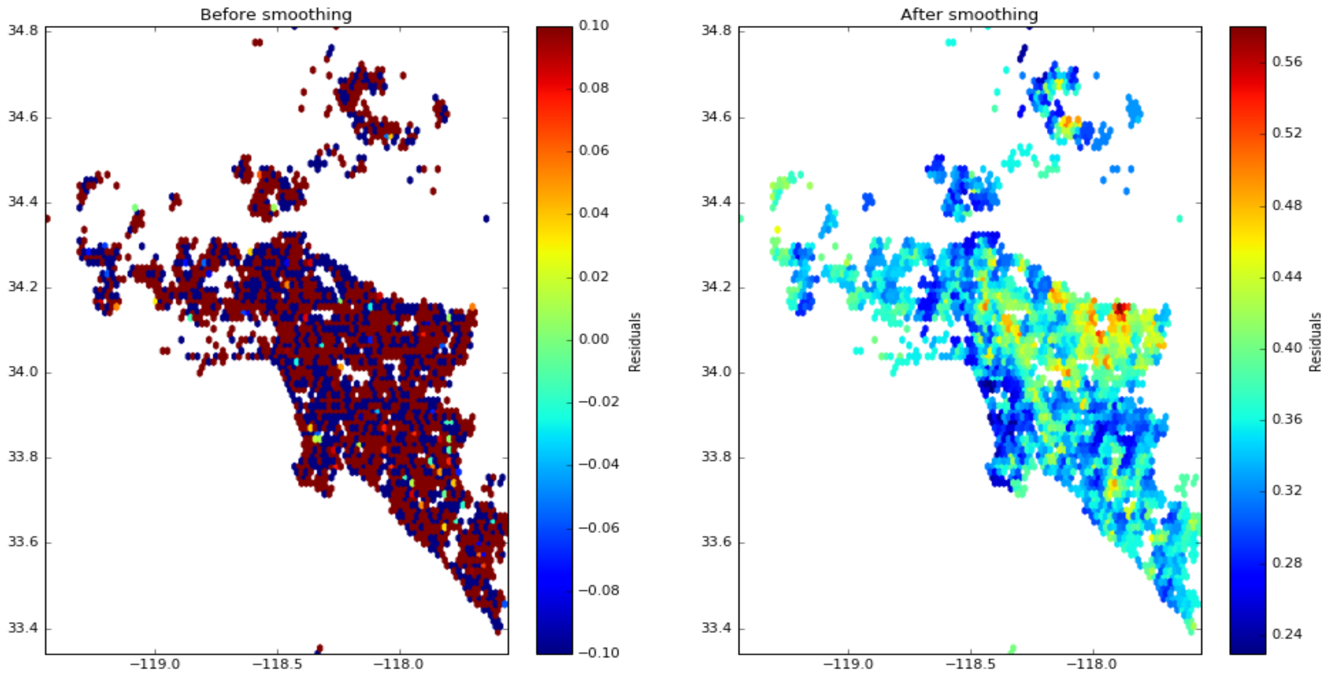


Log transformation

- Log – transformation of a right-skewed variable:
  - Makes its distribution looks more normal,
  - Pulls the extreme values on the right closer to the median.
  - Pushes the extreme values on the left further away from its median.

Spatial Smoothing

K-nearest neighbors



- Spatial smoothing was done using the automated tool of AGD.
- Input:
  - longitude
  - latitude
  - residuals (from a simple model)
  - exposure
- Output: smoothed residuals by 3 classes.

Penalized Linear Models

LASSO, RIDGE and ELASTIC NET

- **Classic Regression concept:**
  - → $Y = \sum_{j=1}^{p} \beta_j X_j$
    - *Y: response variable with sample $y_{i,i=1..n}$*
    - *Xj, ij = 1..p: p explicative variable with sample: $X_{i,j}$, i = 1...n, j = 1..p*
    - *Matrix expression: Y = U β*
  - → The idea is for find the relation between Y and each X, in the way: 1% increase in X will create β increase in Y
  - → Least-squares solution: $\min_{\beta \epsilon R^p} ||y - U\beta||_2^2$ (*)
  - → If rank(U) = p then the solution is unique
  - → If rank(U) < p then there are infinite set of solution which makes the analysis meaningless
- **Problem: dependence among X → rank(U) < p**
- **Solution: Penalizing β_j**
  - → Lasso regression

$$\min_{\beta \epsilon R^p} ||y - U\beta||_2^2 \textit{ subject to } ||\beta||_1 \le t$$

$$\textit{Lagrange form: } \min_{\beta \epsilon R^p} ||y - U\beta||_2^2 + \Delta\, ||\beta||_1$$

  - → Ridge regression

$$\min_{\beta \epsilon R^p} ||y - U\beta||_2^2 \textit{ subject to } ||\beta||_2 \le t$$

$$\textit{Lagrange form: } \min_{\beta \epsilon R^p} ||y - U\beta||_2^2 + \Delta\, ||\beta||_2$$

  - → Elastic net regression

$$\min_{\beta \epsilon R^p} ||y - U\beta||_2^2 + \Delta_1\, ||\beta||_1 + \Delta_2\, ||\beta||_2$$

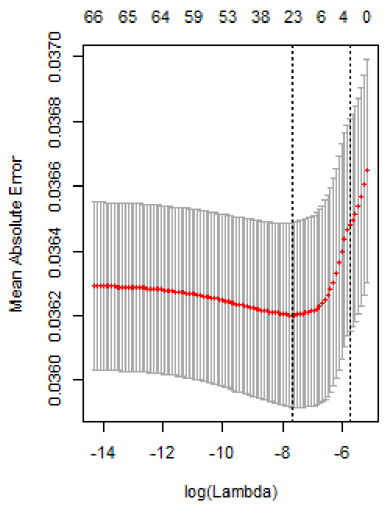
(*) $||x||_p = (\sum_1^n x_i^p)^{1/p}$

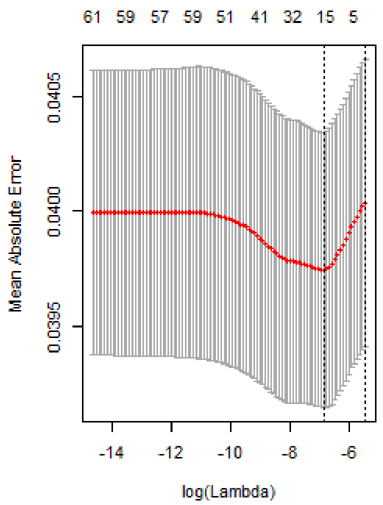
Penalized Linear Models

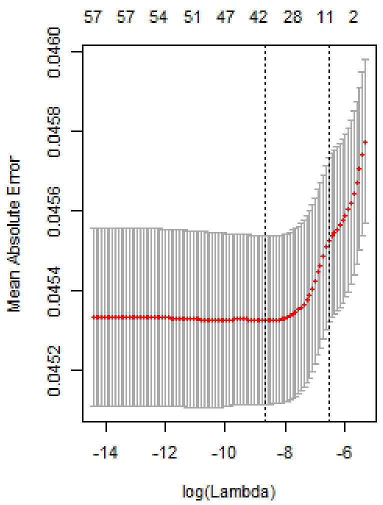
LASSO

```
cv.out = cv.glmnet (x.train, y.train, alpha = 1, nfolds = 5, type.measure = "mae", parallel = T)
plot(cv.out)
bestlam = cv.out$lambda.min #best lambda
```

- Begin with a simple model, then enrich it with the more complicated models later on.
- 3 populations from 3 counties have different best lambda.
- Dividing the population by 3 leads to a trade-off between variance & bias.
- The less populated county has the largest lambda’s variance, it’s worth regrouping this county together with the third one.
- Ridge Regression & Elastic Net have also been tested but didn’t provide better CV (and LB) results than Lasso.







GBDT family

GBM

- The traditional boosting algorithm by Breiman & Friedman.
- Ensemble of trees, to fit the objective function:

$$obj(\ ) = \sum_{i}^{n} l(y_i, \widehat{y_i})$$

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

  - → **Easy to tune & good accuracy**
  - → **Slow training**

GBDT family

XGBoost (Penalized Gradient Boosting)

- Compare to the classical GBM by Friedman, ▯L1 and L2 regularization parameters are available:

$$L(\quad) = \sum_i l(\hat{y}_i, y_i) + \sum_i \Omega(f_k)$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|w\|^2$$

Where l measure the difference between the prediction $\hat{y}_i$ and the target y_i, and the second term Ω penalizes the complexity of the individual tree. When the regularization parameters are equal 0, the objective falls back to classical GBM.

- Pros:
  - **Faster training**
  - **Better accuracy**
  - **Customizable loss function**
- Cons:
  - **Difficult installation**
  - **More parameters to be tuned**

XGBoost parameters	General parameters	nthread	number of parallel threads
		booster	gbtree or gblinear
	Booster parameters	eta	shrinkage parameters
		gamma	minimum loss reduction required to make a split
		max_depth	maximum depth of a tree
		min_child_weight	minimum sum exposure needed in a child
		subsample	subsample ratio of the training sample
		colsample_bytree	subsample ratio of columns when constructing each tree
		lambda	L2 regularization on weights
		alpha	L1 regularization on weights
		lambda_bias	L2 regularization on bias
	Task parameters	objectives	regression / classification
		evaluation_metric	mae, rmse, logloss, customized loss, ….

GBDT family

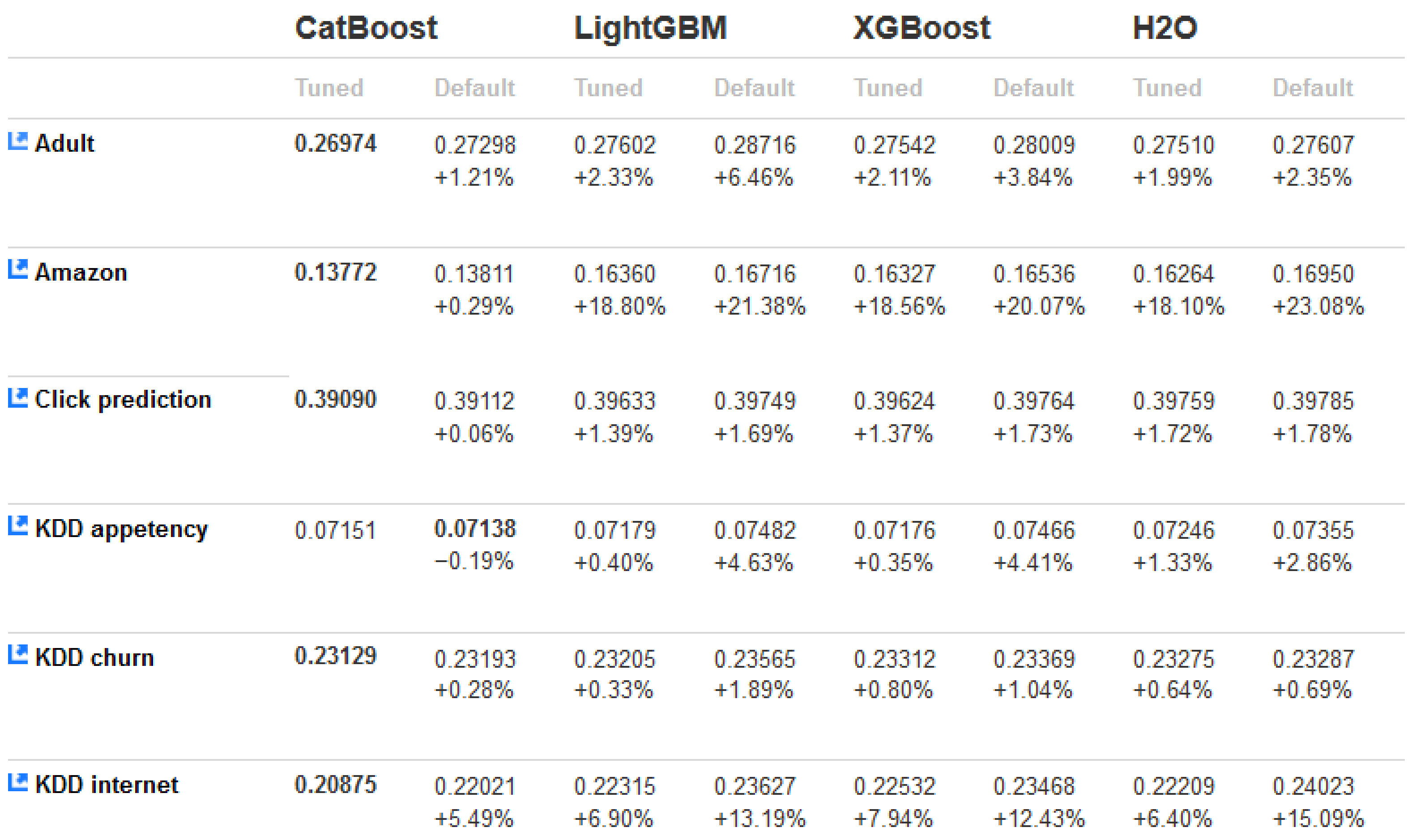
Dynamic Boosting - Catboost

- GBM, XGBoost, LightGBM are susceptible to a bias in a pointwise gradient estimates that lead to reduced accuracy.
- Catboost proposes a dynamic boosting approach that avoids the estimation bias at a minimal cost of variance.
- Catboost can outperform tuned GBM, XGBoost, LightGBM even with its default parameters.
- Catboost works well with categorical features.
- Catboost doesn’t have good training time.
- Mathematical background of dynamic boosting can be found at: https://arxiv.org/abs/1706.09516

GBDT family

Benchmark

- Logloss values for classification tasks obtained by different models on several data sets.
- CatBoost outperforms other model of the GBDT family even with its default parameters.



	CatBoost		LightGBM		XGBoost		H2O	
	Tuned	Default	Tuned	Default	Tuned	Default	Tuned	Default
Adult	**0.26974**	0.27298 +1.21%	0.27602 +2.33%	0.28716 +6.46%	0.27542 +2.11%	0.28009 +3.84%	0.27510 +1.99%	0.27607 +2.35%
Amazon	**0.13772**	0.13811 +0.29%	0.16360 +18.80%	0.16716 +21.38%	0.16327 +18.56%	0.16536 +20.07%	0.16264 +18.10%	0.16950 +23.08%
Click prediction	**0.39090**	0.39112 +0.06%	0.39633 +1.39%	0.39749 +1.69%	0.39624 +1.37%	0.39764 +1.73%	0.39759 +1.72%	0.39785 +1.78%
KDD appetency	0.07151	**0.07138** −0.19%	0.07179 +0.40%	0.07482 +4.63%	0.07176 +0.35%	0.07466 +4.41%	0.07246 +1.33%	0.07355 +2.86%
KDD churn	**0.23129**	0.23193 +0.28%	0.23205 +0.33%	0.23565 +1.89%	0.23312 +0.80%	0.23369 +1.04%	0.23275 +0.64%	0.23287 +0.69%
KDD internet	**0.20875**	0.22021 +5.49%	0.22315 +6.90%	0.23627 +13.19%	0.22532 +7.94%	0.23468 +12.43%	0.22209 +6.40%	0.24023 +15.09%

Stacking

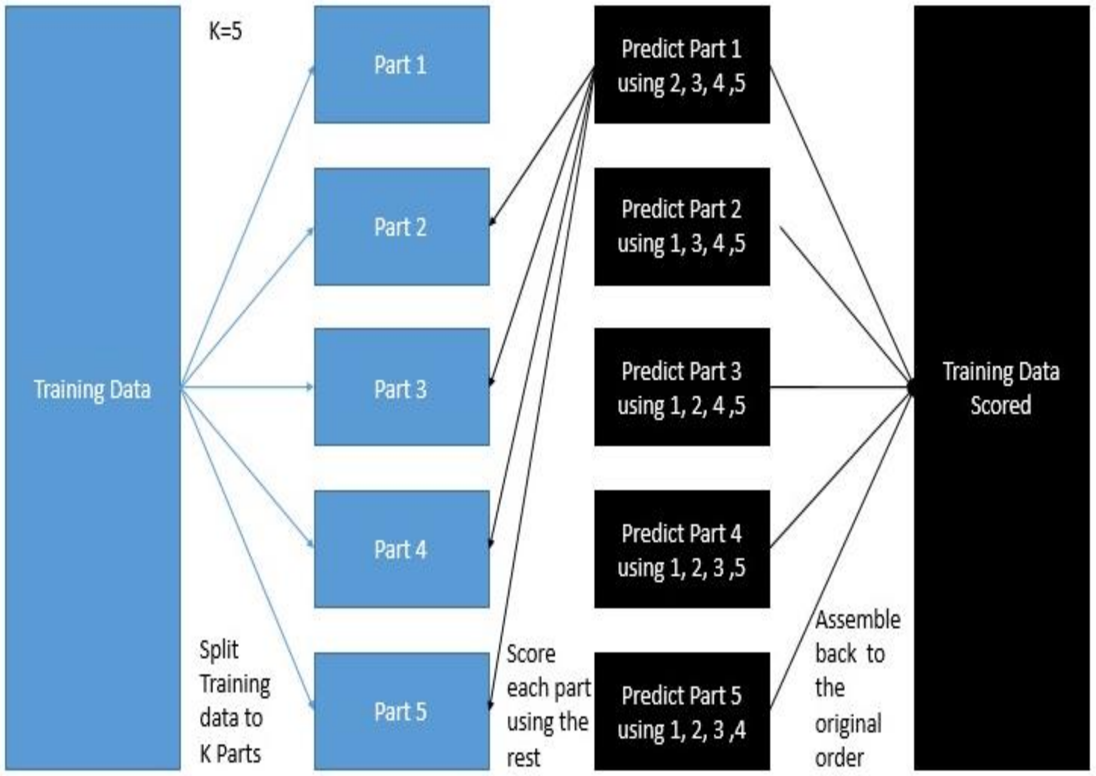
Combination of models

Overview

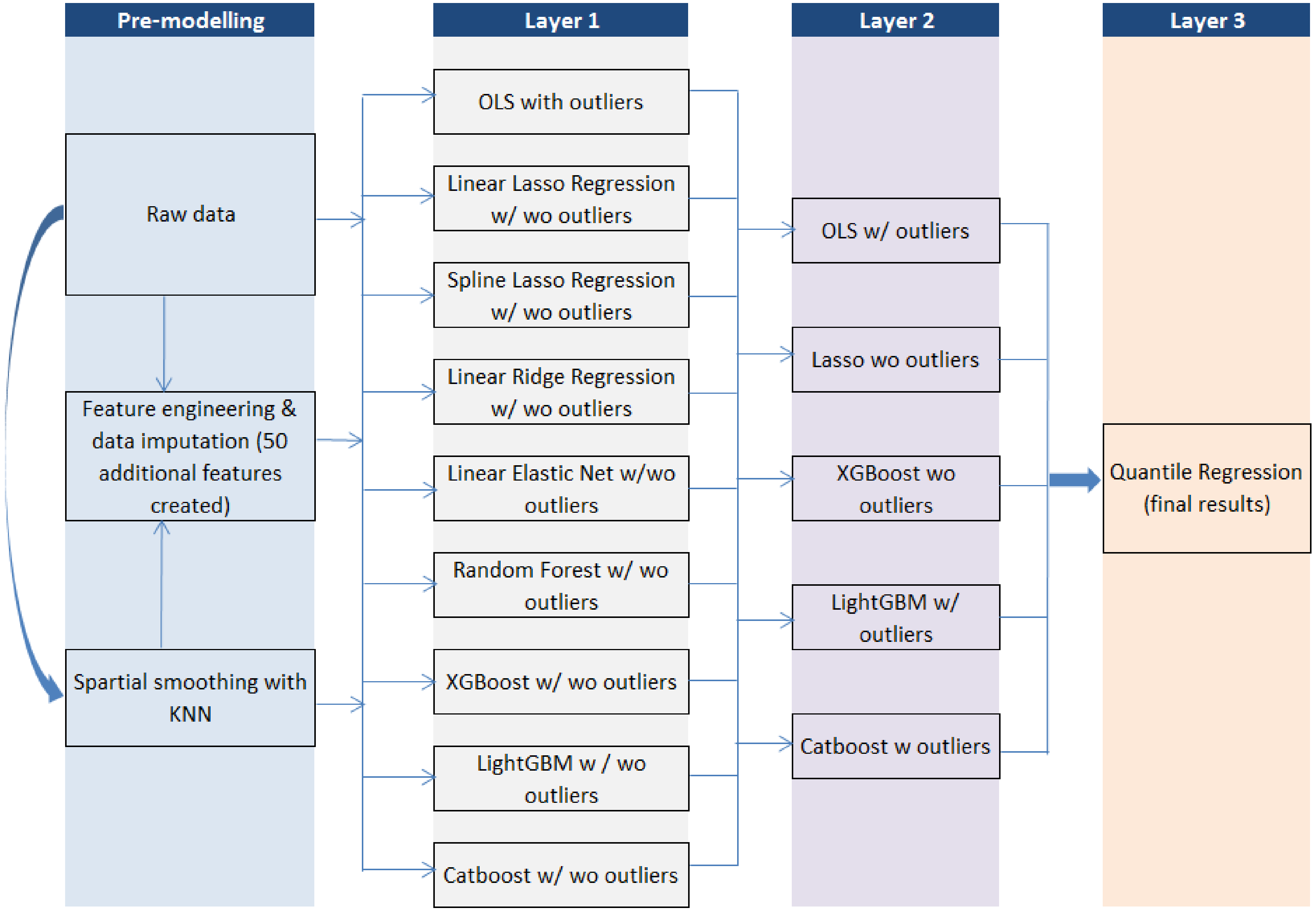
- Stacking consists of combining predictions of different models to make a new model. It leverages the best of each model, and discredit the base model where it performs poorly.
- The metal model for stacking is often simple, and resistant to outliers (ex: quantile regression)

Details

- Training data set is divided into n parts: $D_1, D_2, D_3, \dots, D_n$
- We train various machine learning algorithms on data set $\{D_i\}_{i \neq k}$
- We make predictions for each of the algorithms for data set D_k , and we create new data set D'_k that contains only these predictions.
- We aggregate the out-of-bag predictions $D'_1, D'_2, D'_3, \dots, D'_n$, in order to construct a **meta training set**.
- We train each base learner on the whole training set, and store the predictions on the prediction set, in order to construct a **meta prediction set**.
- We train a new (simple) model (called **meta learner**) on the meta training set.
- We apply our meta learner on the prediction set to make the final predictions.


Training Data
K=5
Part 1
Part 2
Part 3
Part 4
Part 5
Split Training data to K Parts
Score each part using the rest
Predict Part 1 using 2, 3, 4 ,5
Predict Part 2 using 1, 3, 4 ,5
Predict Part 3 using 1, 2, 4 ,5
Predict Part 4 using 1, 2, 3 ,5
Predict Part 5 using 1, 2, 3 ,4
Assemble back to the original order
Training Data Scored

Final Solution



Pre-modelling
Layer 1
Layer 2
Layer 3
Raw data
Feature engineering & data imputation (50 additional features created)
Spartial smoothing with KNN
OLS with outliers
Linear Lasso Regression w/ wo outliers
Spline Lasso Regression w/ wo outliers
Linear Ridge Regression w/ wo outliers
Linear Elastic Net w/wo outliers
Random Forest w/ wo outliers
XGBoost w/ wo outliers
LightGBM w / wo outliers
Catboost w/ wo outliers
OLS w/ outliers
Lasso wo outliers
XGBoost wo outliers
LightGBM w/ outliers
Catboost w outliers
Quantile Regression (final results)

Resumé

Lessons learned

- Creative feature engineering makes the difference.
- Careful treatment of outliers is important.
- Spend time on parameter tuning is worth doing

What's more?

- Time series analysis on the change of the absolute value log-error over time.
- Clustering the data into groups.
- Better and proper stacking.
- Better models with spline (Emblem?)
- Special treatments of spatial features.