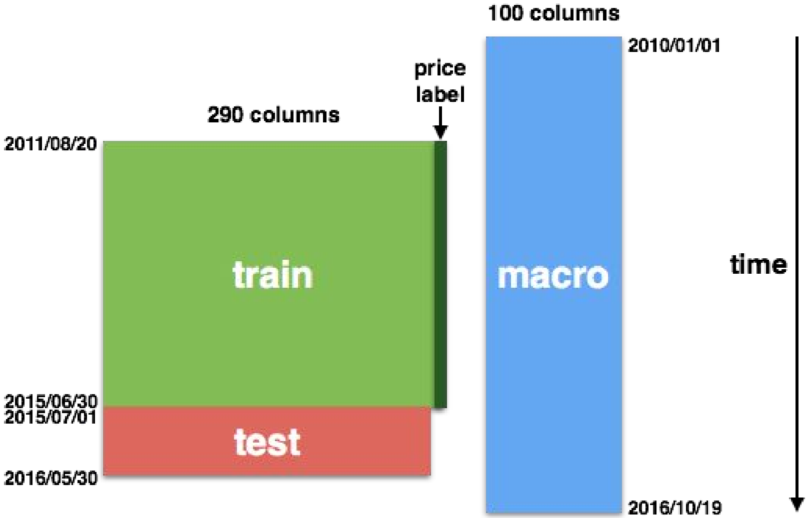
Kaggle Competition: Russian housing prediction

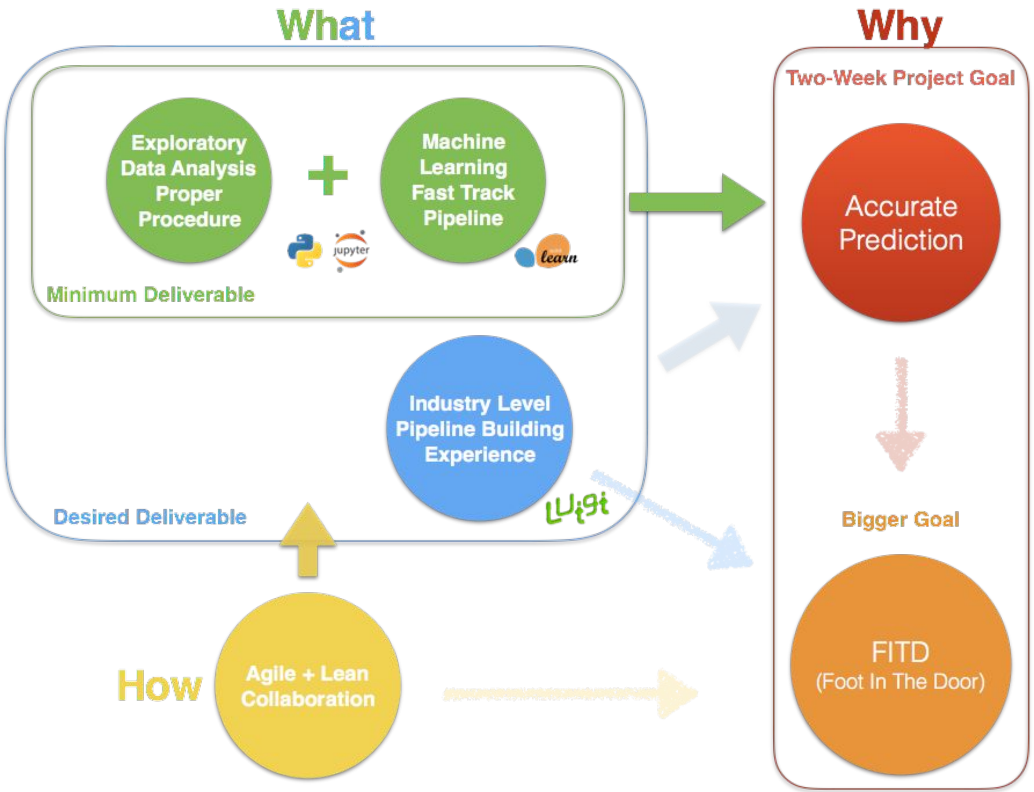
Chao Shi
William Zhou
Sam O'Mullane
Yabin Fan

NYC Data Science Academy
5/30/2017

Competition Intro

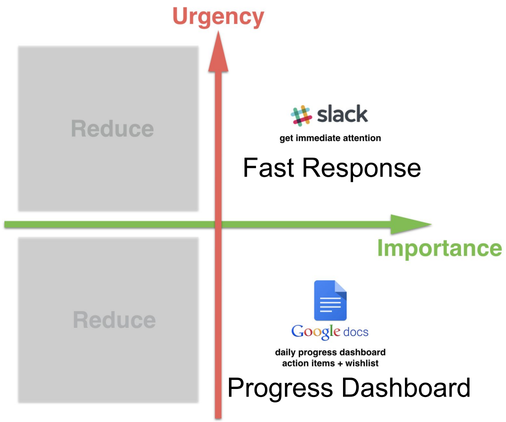


Project Summary



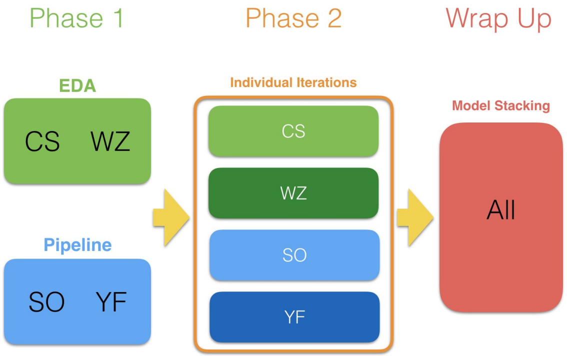
Agile + Lean

Communication Strategy



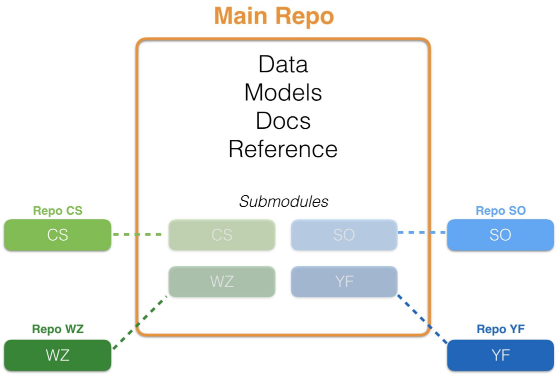
"Continuous Improvement"
"Eliminate Waste"

Project Schedule and Workload Balance
Initial Vision



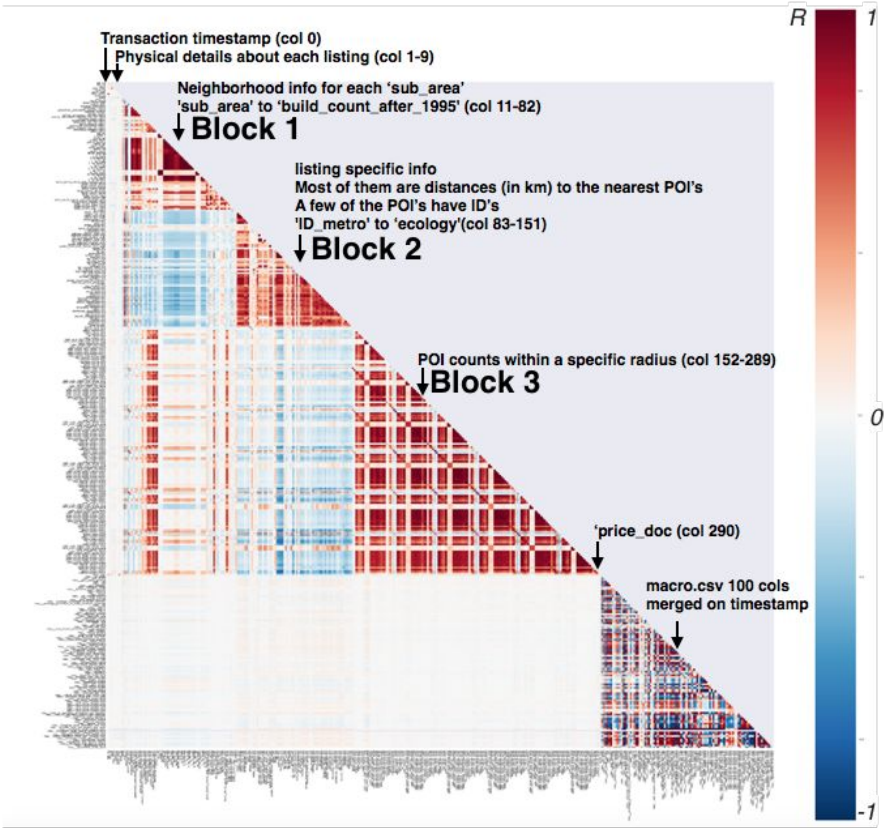
"Just In Time", "Reduce Wait Time"
"Regular Reflection & Adaptation"

Version Control and File Sharing
GitHub



"Simplicity"
"Flexibility"

EDA First Round: Multicollinearity



Mitigation

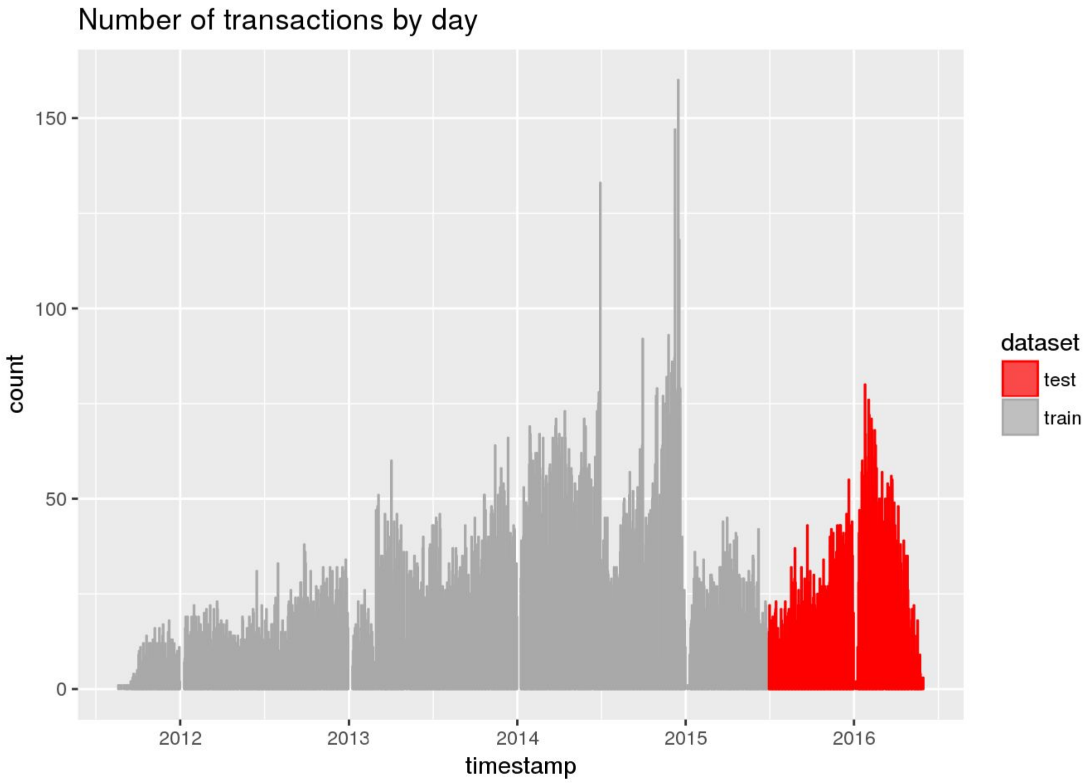
Block 1: most columns are dropped. For all listings in the same 'sub_area', the values are all the same in block 1

Block 2: highly correlated columns are treated. For example: Male / Female

Block 3: most columns for radius larger than 500m are dropped. Values are more and more correlated as radius grows

Macroeconomic data were skipped first, so we could initiate fast-track machine learning feedback loop asap

EDA First Round: Feature Generation (1)



Standard New Feature Examples

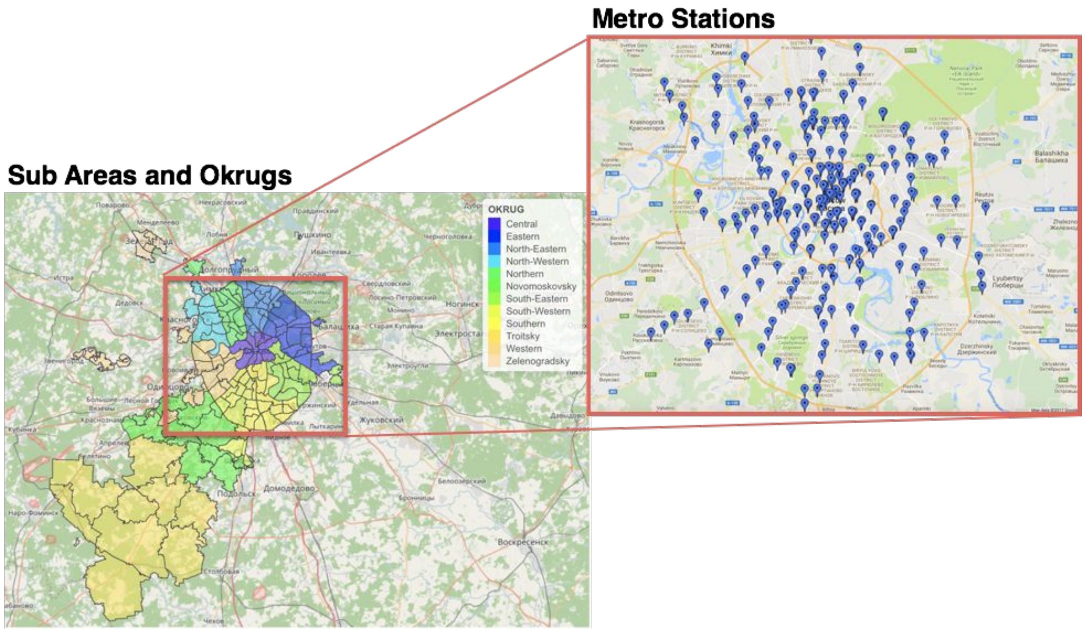
Age of apartment: timestamp - built year
Relative height: floor / max floor
Population Density: population / area

Number of Transaction Features*

Monthly transaction: 'month_year_cnt'
Weekly transaction: 'week_year_cnt'

**these obviously interact with macro*

EDA First Round: Feature Generation (2)



Categorical cols with high cardinality

Approach 1: helps tree based methods

merge classes with less observations
reduce dummified boolean columns

Approach 2: helps both regression and tree based methods

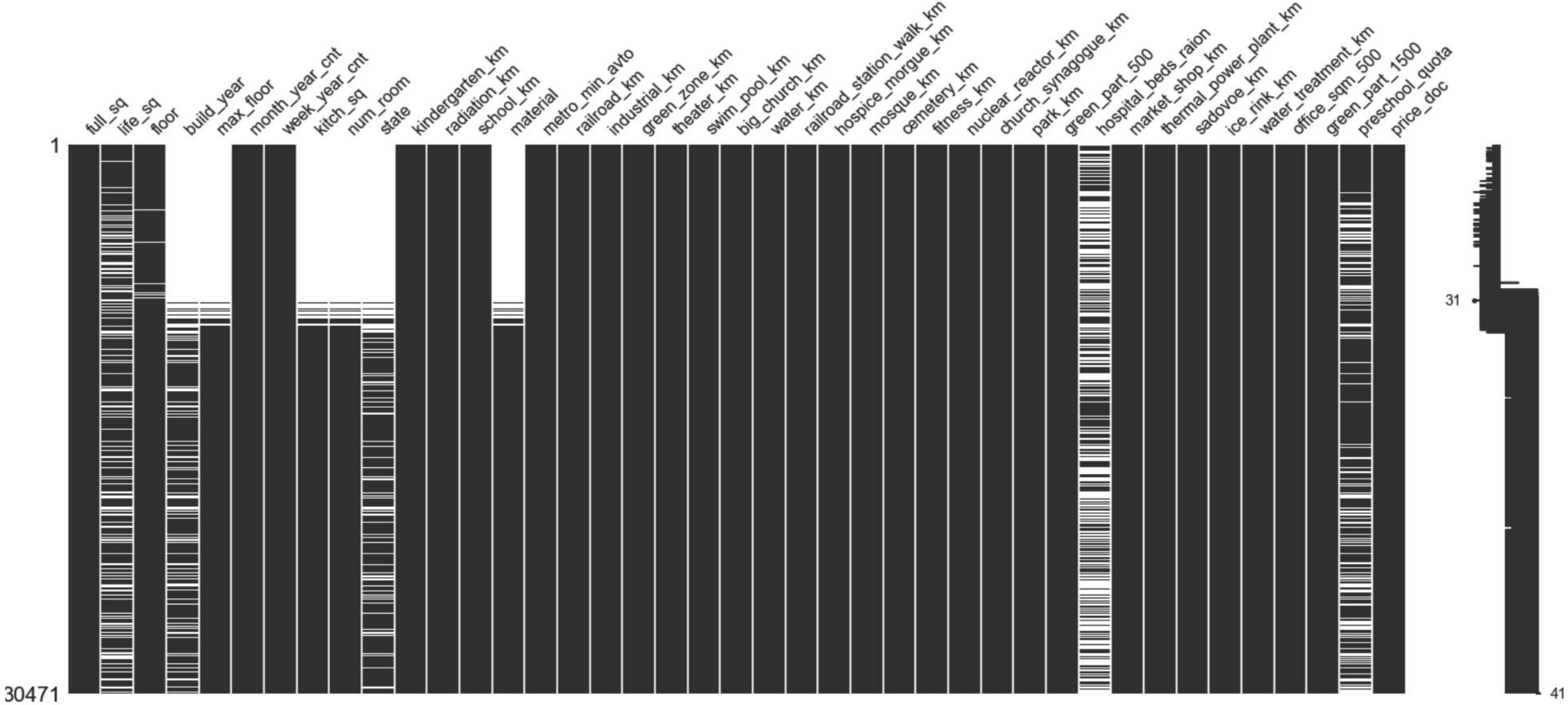
replace categorical features with numeric ones, for example:

'ID_metro' (nearest metro station ID)

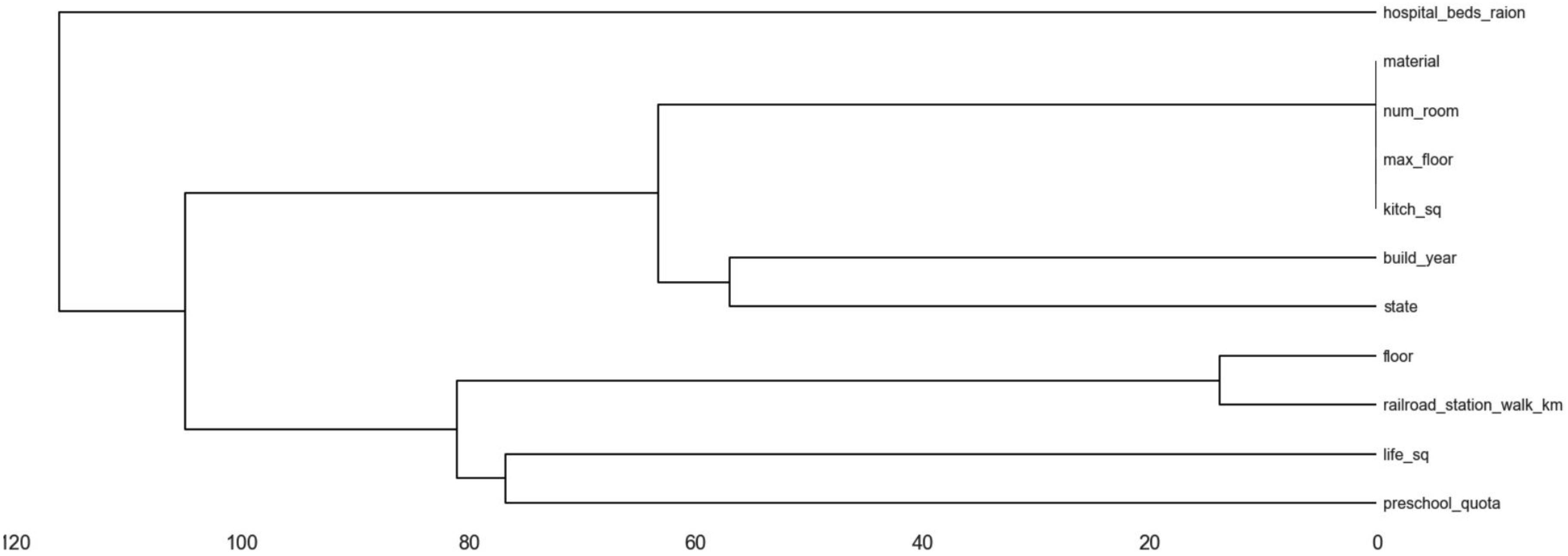
average unit price near this metro station*

**using these feature might lead to overfit issue*

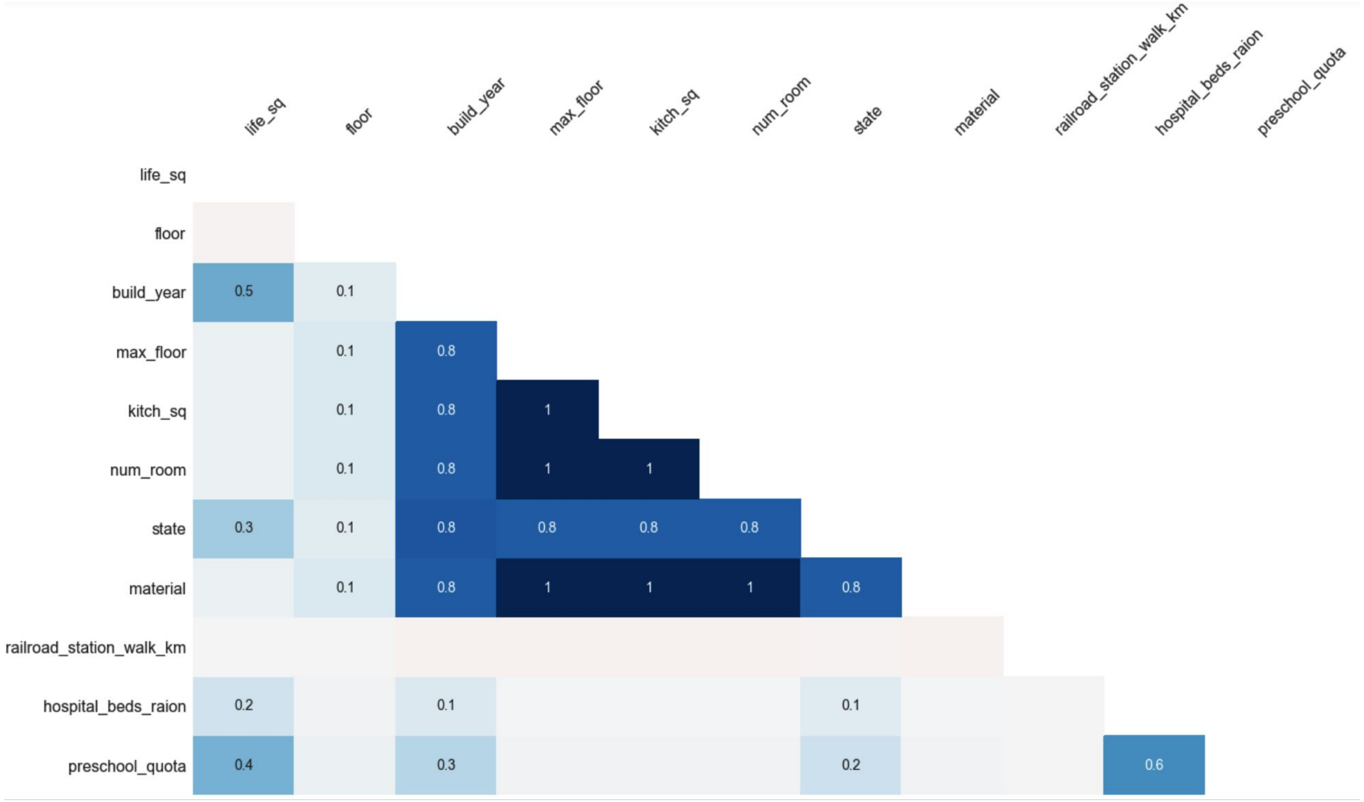
Dealing with the Following Missing Features



Missing Value: Dendrogram



Missing Value: Correlation



Compare Different Imputation Methods

```
In [64]: methods = ['MICE','KNN','Simple']
         #methods = ['MICE','Soft','KNN','Simple']
         # methods = ['MICE','KNN']
         cmpImpute(X,X_incomplete,missing_mask,test_col_name,methods)
```

```
[MICE] Starting imputation round 106/110, elapsed time 0.099
[MICE] Starting imputation round 107/110, elapsed time 0.100
[MICE] Starting imputation round 108/110, elapsed time 0.101
[MICE] Starting imputation round 109/110, elapsed time 0.101
[MICE] Starting imputation round 110/110, elapsed time 0.102
Imputing row 1/1000 with 0 missing, elapsed time: 0.215
Imputing row 101/1000 with 0 missing, elapsed time: 0.217
Imputing row 201/1000 with 0 missing, elapsed time: 0.218
Imputing row 301/1000 with 1 missing, elapsed time: 0.220
Imputing row 401/1000 with 1 missing, elapsed time: 0.221
Imputing row 501/1000 with 1 missing, elapsed time: 0.224
Imputing row 601/1000 with 0 missing, elapsed time: 0.226
Imputing row 701/1000 with 1 missing, elapsed time: 0.227
Imputing row 801/1000 with 0 missing, elapsed time: 0.228
Imputing row 901/1000 with 0 missing, elapsed time: 0.229
```

Out[64]:

	MICE	KNN	Simple
num_room	0.05629	0.037053	0.055877

- **SimpleFill:** Replaces missing entries with the mean or median of each column.
- **KNN:** Nearest neighbor imputations which apply weights on samples using the mean squared difference on features for which two rows both have observed data.
- **MICE:** Multivariate imputation by chained equations

Linear regression VIF in Python

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            price_doc_tr   R-squared:                       0.364
Model:                             OLS   Adj. R-squared:                  0.363
Method:                  Least Squares   F-statistic:                     446.4
Date:                 Sat, 27 May 2017   Prob (F-statistic):               0.00
Time:                         17:05:13   Log-Likelihood:                -20998.
No. Observations:                30471   AIC:                         4.208e+04
Df Residuals:                    30431   BIC:                         4.241e+04
Df Model:                           39
Covariance Type:             nonrobust
===================================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------------------
Intercept                     15.5307      0.011   1396.139      0.000      15.509      15.553
full_sq                        0.2745      0.004     71.108      0.000       0.267       0.282
sadovoe_km                    -0.1469      0.011    -13.427      0.000      -0.168      -0.125
```

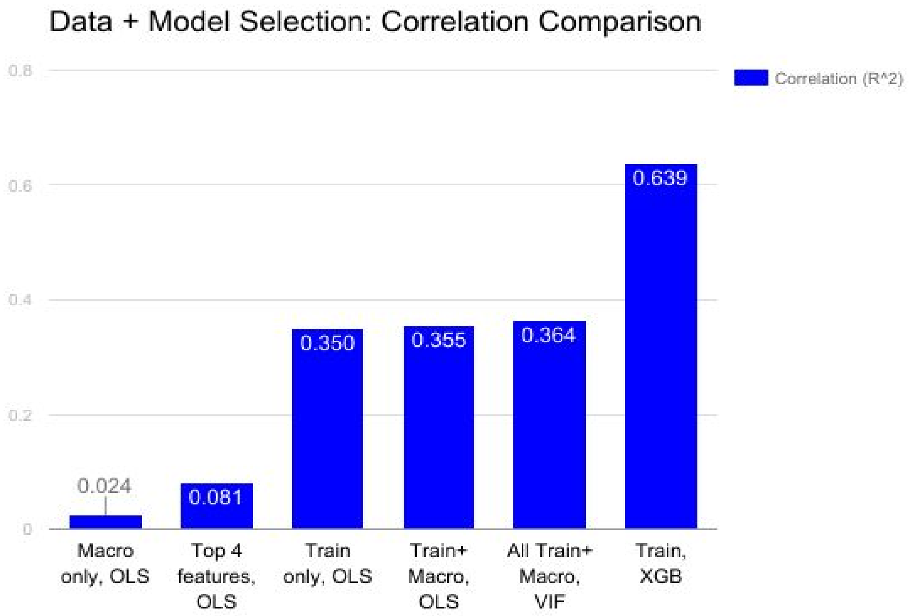
VIF Usage with Russian Housing Dataset

Start with reduced set of features (no correlation between features above threshold value)

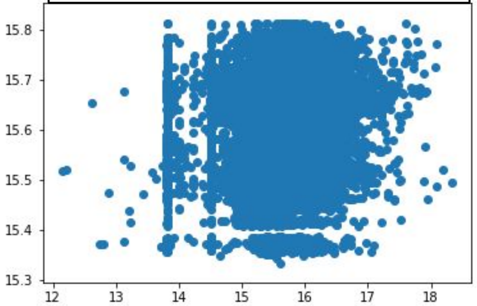
Feed through forward selection algorithm to select features

Run linear regression pipeline to determine optimal model performance

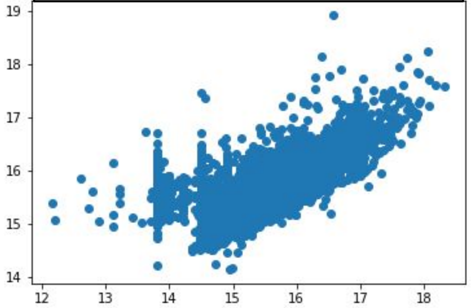
Linear regression model comparison



Predicted v. Actual Prices: Macro data, OLS model



Predicted v. Actual Prices: Train+Macro, Ridge model



XGBoost The algorithm that wins every competition

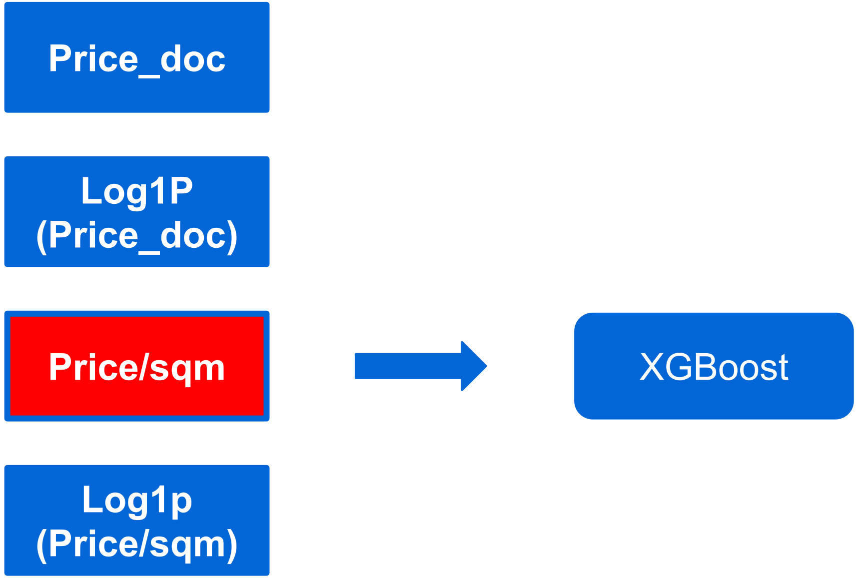
- xgboost learns the best direction for missing values, automatic handle missing value
- Handles both numeric and categorical columns (linear regression only takes numeric)
- Automatically provide estimates of feature importance from a trained predictive model
- Fast (time-management)

What XGBoost Can Not Do For You

- Feature engineering
- Hyper parameter tuning
- Interpretability (black box model)

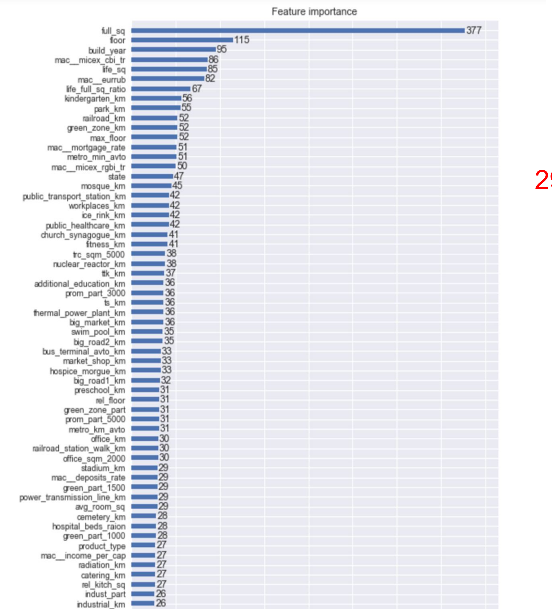
Fast track approach: Let test-rmse/LB score tell you the direction

1. General Data Cleaning
   a. Replace obviously wrong data with NaN, rather than guess the value by yourself.
2. Price transformation:



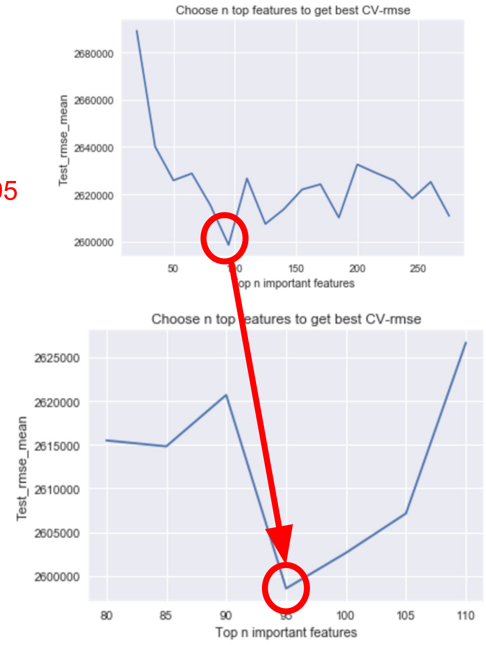
Feature selection methodology: (291 features → 95 features → 40 features)

1. Let XGBoost tells what is important



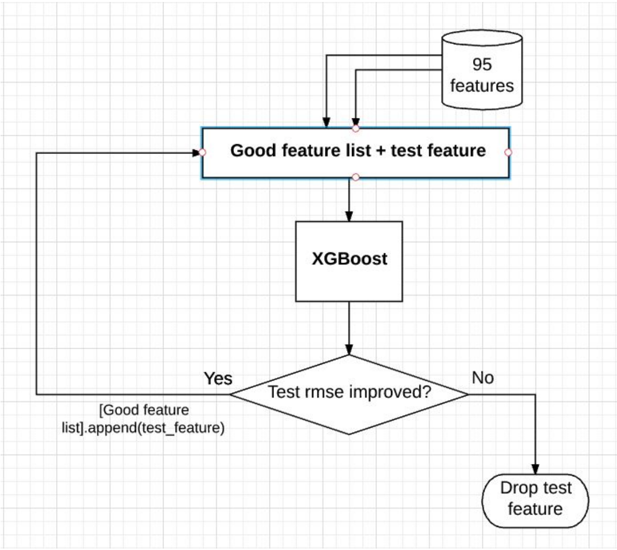
291 → 95

2. Train XGBoost with Top N important features



95 → 40

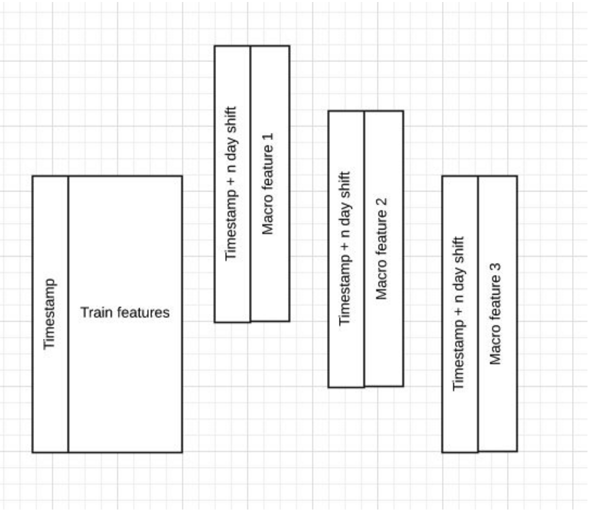
3. Greedy search within the 95 features



40 → ???

4 .Go back to test ‘bad’ features and new features

Try fast, fail fast



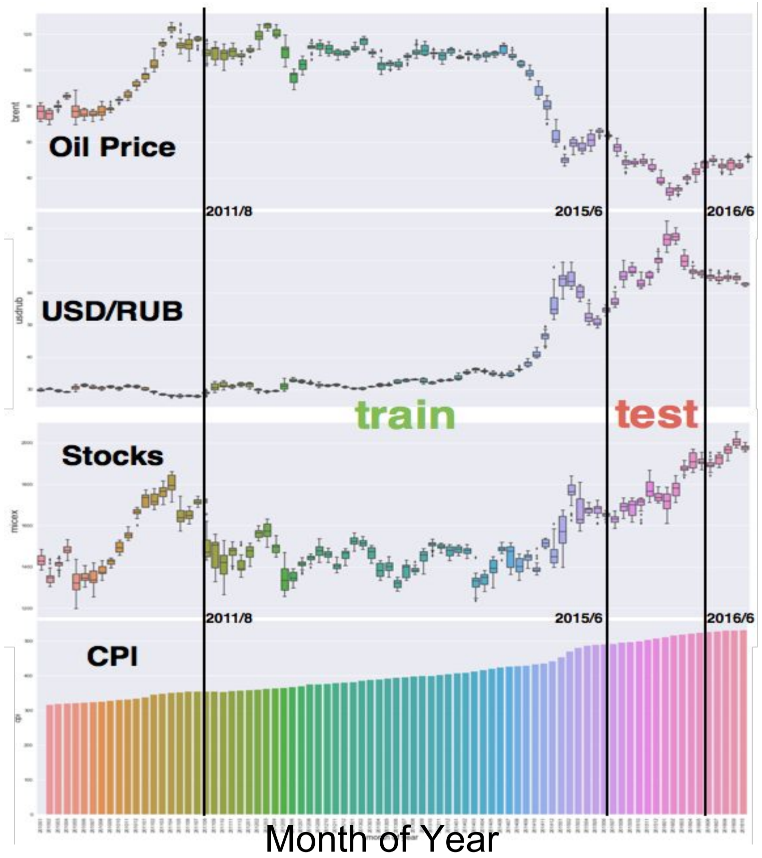
Macro.csv Time Series Data

Basic understanding of Russian Economy and Moscow Property Market

Cross Correlation to figure out the optimized time shift for hand picked columns

Moving Average Curve (high frequency curves were tried first, but not very helpful)

What Happened to Russian Economy?



A Few Comments

More than 50% of Russian export is oil and gas. Oil is globally traded in USD

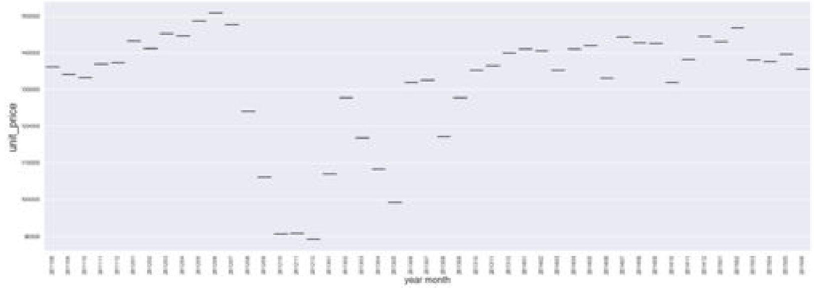
Ruble has moved in the **opposite** direction compared to **USD**, therefore the Russian domestic market felt less impact of the oil price drop

Stock market and **CPI (Consumer Price Index)** both moved **up** during the oil price drop started in 2014

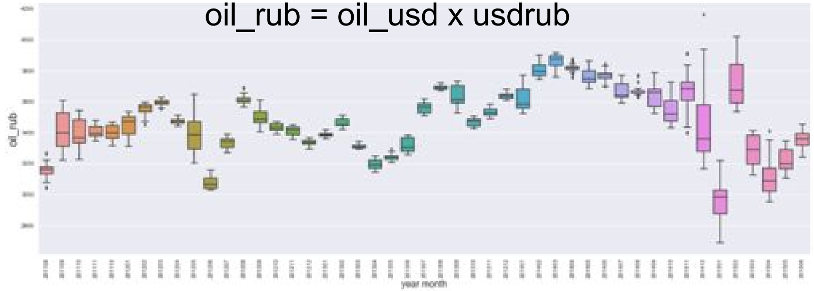
Macro Feature Engineering Example: oil_rub

Feature Engineering

monthly unit price

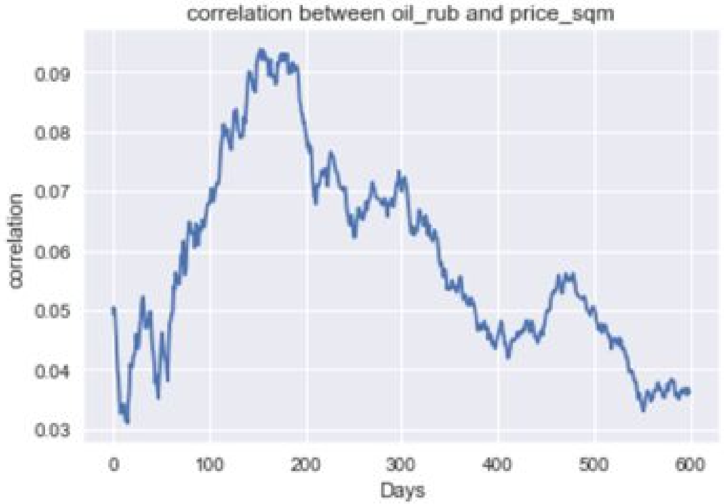


oil_rub = oil_usd x usdrub



Find Best Time Shift with xcorr

`oil_rub`

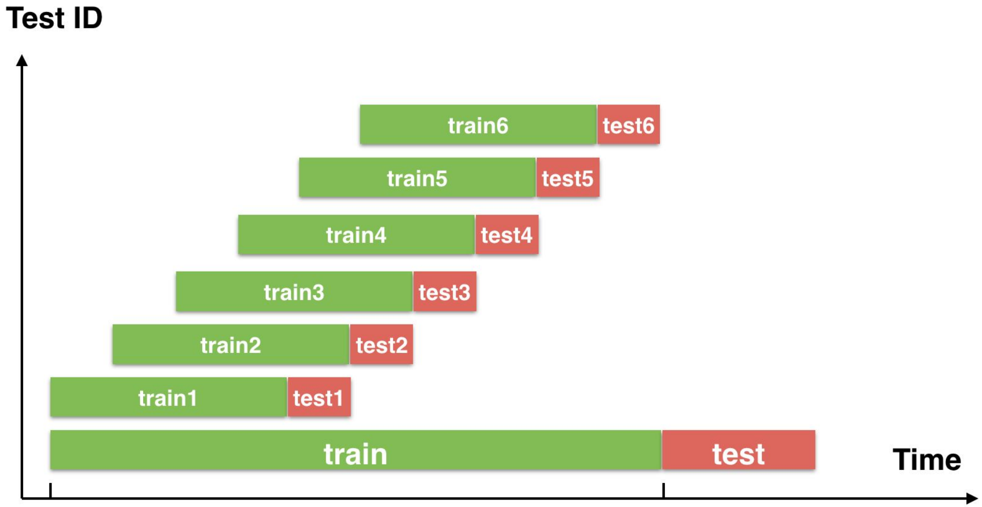



```
The best time lapse for oil_rub is 153 days
```

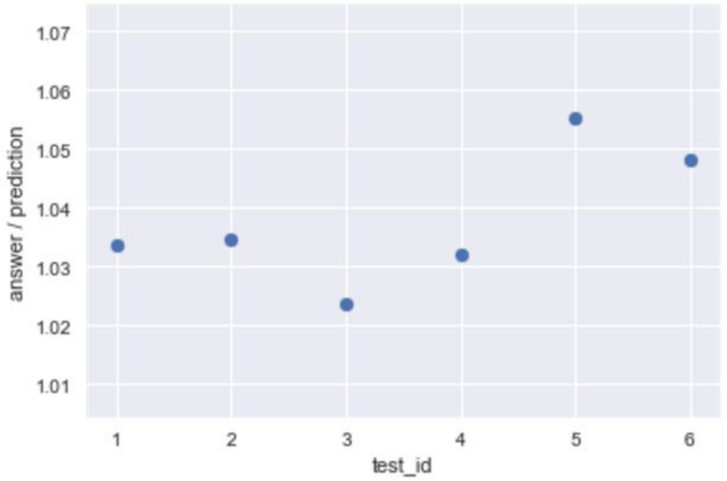
We only searched for **leading** indicators

Time Domain Prediction Accuracy Experiment

Experiment Design



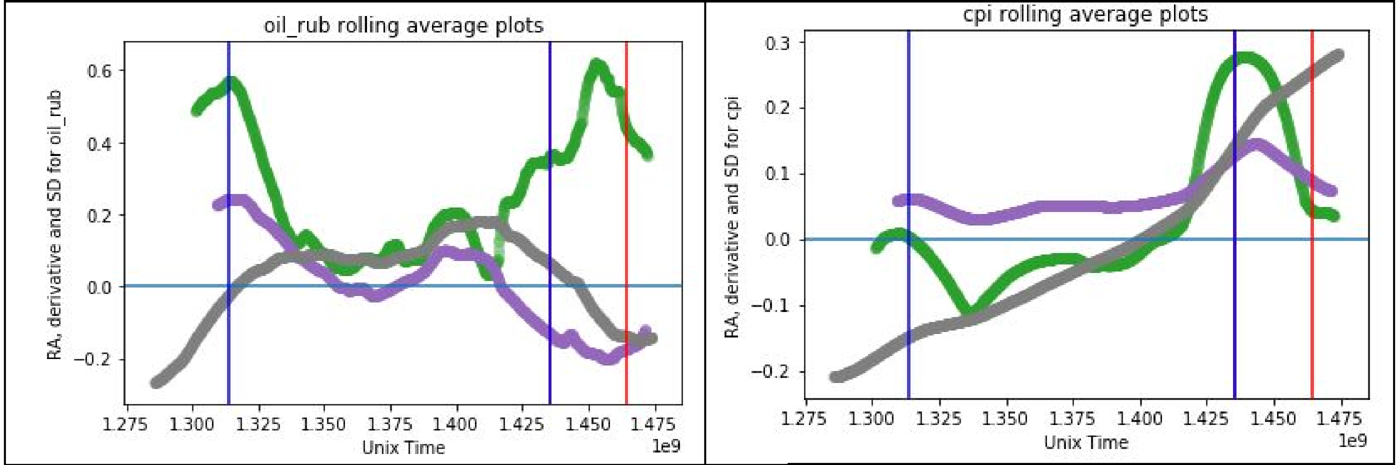
Truth / Prediction Result



With a 4:1 train:test ratio (along time axis),
Our xgboost model generally miss-predict by 2-5%

Macro dataset: explaining the multiplier

Derived quantities from Macro dataset



Feature Key

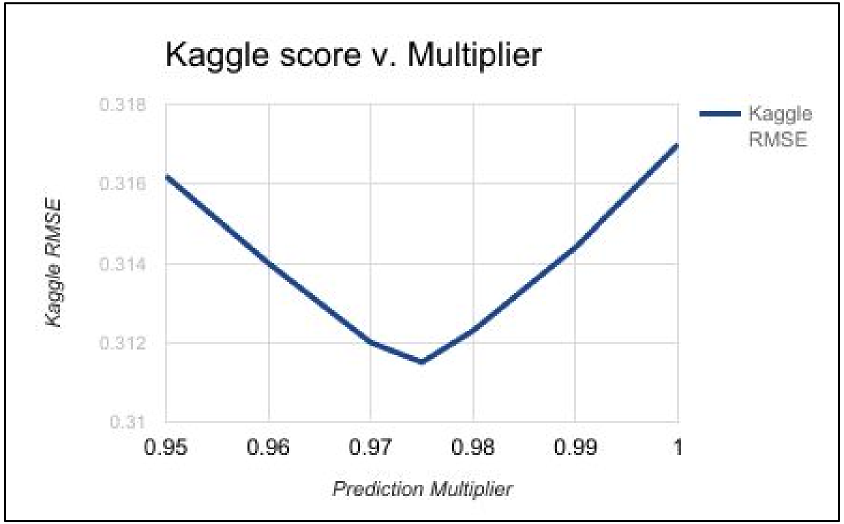
Grey = Rolling average (RA) of feature of interest
Purple = Derivative of RA curve
Green = Rolling standard deviation of feature of interest

Line Key

Blue = start/stop of “train” data
Red = start/stop of “test” data

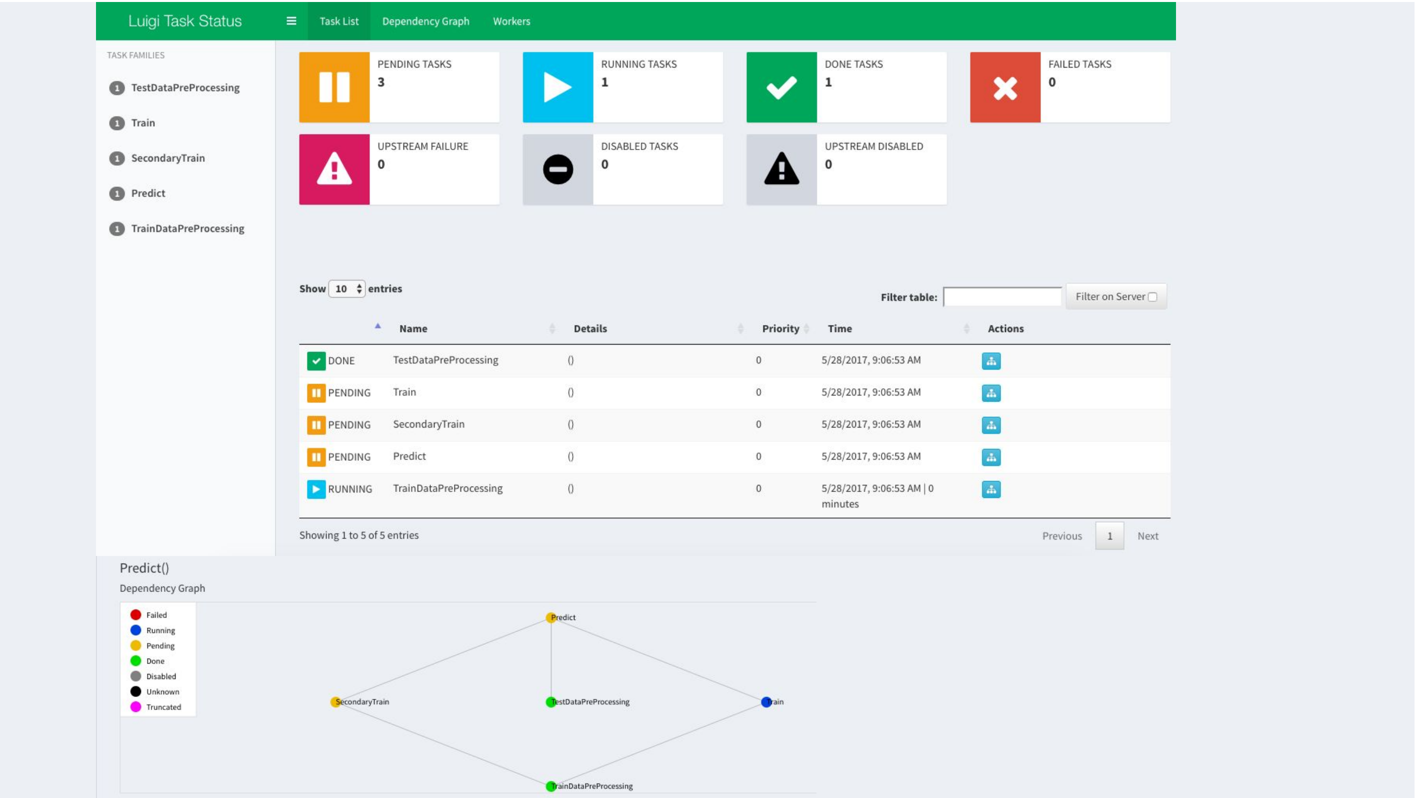
Macro dataset: explaining the multiplier, cont.

Empirically determine the optimal multiplier -- value of 0.975 close enough to minimum



Future direction: Use macro dataset to reproduce multiplier effect

Luigi Monitoring Interface



Using the Pipeline

Command Line Interface for Luigi

```
[redacted] $ sh run.sh
[redacted]

Available models:
XGB (XGBOOST),
RF (Random Forest),
FLR (forward-select LinReg),
RLR (ridge LinReg)
EN (elastic net lin reg)
xgbGrid (XGB with grid search on hyperparam)

Input the model you want to use, followed by [ENTER]:

xgb

Enter second model choice for stacking (or none if single)

flr
```

```
===== Luigi Execution Summary =====

Scheduled 5 tasks of which:
* 5 ran successfully:
    - 1 Predict()
    - 1 SecondaryTrain()
    - 1 TestDataPreProcessing()
    - 1 Train()
    - 1 TrainDataPreProcessing()

This progress looks :) because there were no failed tasks or missing external dependencies

===== Luigi Execution Summary =====
```

UserInput.py

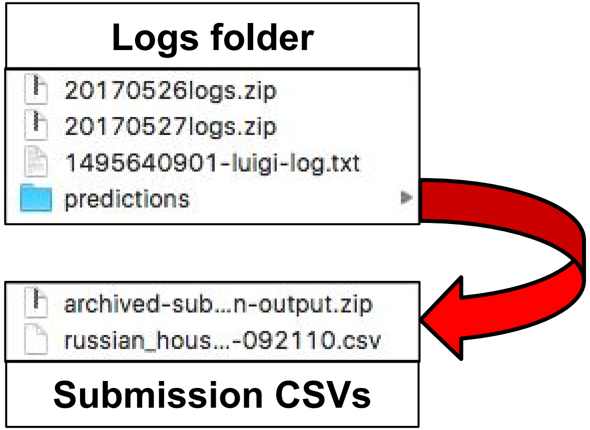
```python
#############  Tree  #################
## Single XGB parameter input ########
def user_xgb_param():
    return {'learning_rate': 0.05,
            'max_depth': 5,
            'subsample': 0.9,
            'colsample_bytree': 0.9,
            'objective': 'reg:linear',
            'eval_metric': 'rmse',
            'silent': 1,
            'seed': 0,
            'min_child_weight':1,
            'gamma':0
            }
```

Log File + Submission CSV

Custom Logging in Luigi

Example log file (partial)

```
20170526-114502: CV XGBoost best RMSE = 40841.6263017


20170526-114502: CV XGBoost output nround = 106


20170526-114517: XGBoost Final RMSE = 31762.1276932
XGBoost Final R2 = 0.673254306048

          names  values
14      full_sq     721
21      life_sq     490
18   build_year     407
9         floor     404
10  month_year_cnt  318

20170526-114517: Successfully trained model


20170526-114517: Successfully wrote model to pickle


20170526-114517: Predict Node initiated


20170526-114518: No (or invalid) 2nd model choice


20170526-114518: Write of submission to csv successful
```



Fully automated

- Activity logs
- Submission files

Easily customized

Optimize collaboration

Ensembling and Stacking

Trial:

Model Engineering: Validation and Ensemble (?)

Delivered Product

Price prediction with a top 2% accuracy ranking after 2 weeks effort

Executable Documentation: EDA + Machine Learning Algo Pipeline

Industry level pipeline building example with Luigi

Agile + Lean adaptation in a machine learning project

I plan to make a more detailed flowchart to show the sprints if we have time. This chart is a low priority item. - CS

Acknowledgments

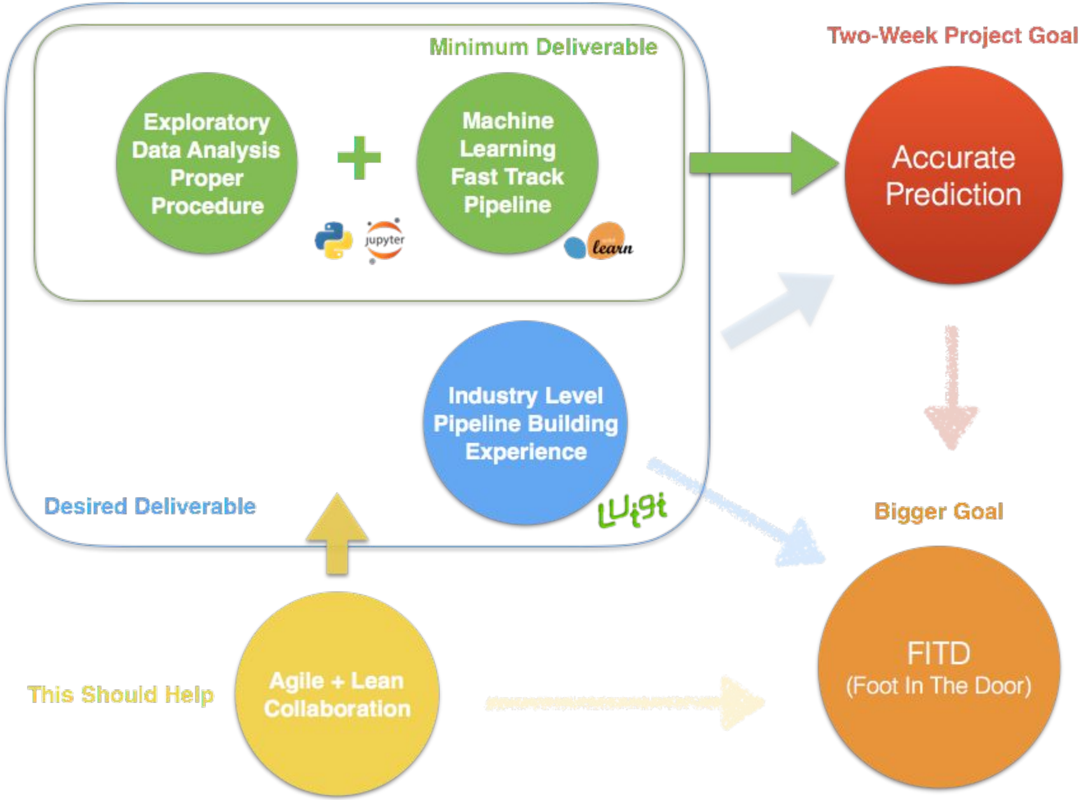
NYCDSA: Shu Yan for data cleaning and pipeline demonstration

NYCDSA: Aiko Liu, Luke Lin, Zeyu Zhang for guidance and discussion

Top voted kernels and discussions on Kaggle for inspiration

Agile workflow for Kaggle projects demonstrated by previous cohort teams

Thank you. Questions?



Backup

Agile + Lean Guideline Document. Why? How?

1) Effective communication strategy: "Continuous Improvement" + "Eliminate Waste"

General observation:
The four of us often have different schedules. We are at different places especially during the weekends. We only have 2 weeks on the project.

What do we want?
Yes -- early feedback, knowledge share, ability to make fast response
No -- frequent long meetings and interruptions

Yes -- informative dashboard so everyone knows other people's progress
No -- complicated/tiring documentation system

Yes -- version control and file sharing system
No -- sending files in unorganized fashion through multiple channels

Proposed solution:
a) Important and Urgent info goes to Slack. We exchange our cell numbers just in case.

b) Important but Non-urgent info goes to Google Docs. Google Docs = progress dashboard + key strategy reference + ticket pool (wishlist)

Each of us would write in our own sections. At the end of each day, type at least 1-2 sentences describing the progress. Feel free to write more if needed. As soon as we enter the iterative computational stage, start making and updating a picture to provide quantitative visual update.

The high level project plan is stored here. Feel free to choose a color arrow to mark where you are.

Record your non-urgent wish list here. After each sprint (1-2 days) we go over the wish-lists and decide what to work on for the next sprint.

c) File sharing and version control with GitHub. We would make one main project repository, while each of us create our own individual repo within our own accounts. We use the "submodule" method to link our individual repos to the main repo. Kaggle source data will be stored in the main repo; all files shared through all other channels will be copied to a "reference" folder in the main repo too.

d) Unimportant communication should generally be limited.

e) Avoid important discussion when certain members are missing -- we don't want to waste time saying the same things many times. We would assign relatively independent tasks to Wei, because of his different time schedule.

2) Workload balance and optimization: "Just In Time" + “Regular reflection & adaptation” + “Simplicity”

General observation:

Feature engineering and pipeline building are both adaptive and continuous effort throughout the 2-week project. Wei and Chao would physically see each other after 6pm; Sam, Yabin and Chao overlap during the day.

What do we want?

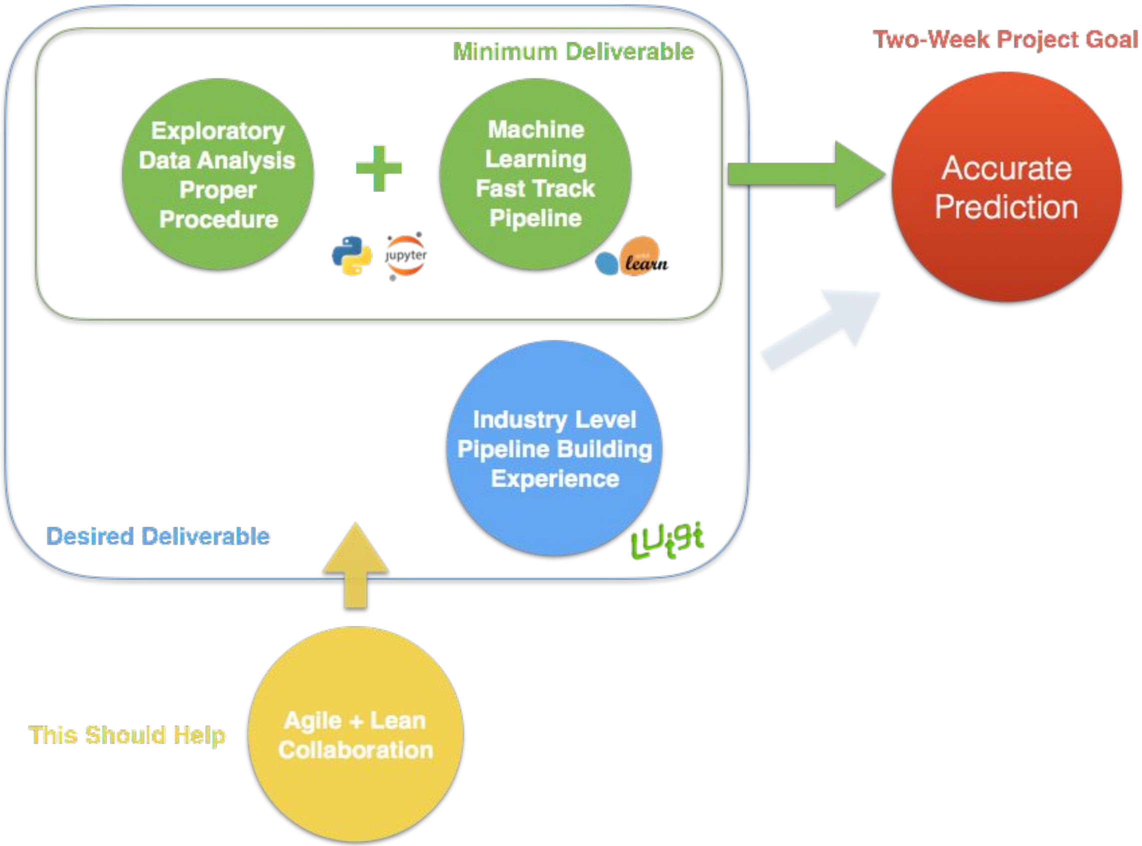
Start data digestion and pipeline building both on day 1. Enable fast-track solution delivery, effectively optimize computational schedules. Minimize overlapping effort while guarantee knowledge backup.

Proposed solution:

Two phased project cycle --

a) Phase 1, two sub-teams, daily sprints. During the initial stage, we divide our team into two sub-teams. Chao and Wei, Sam and Yabin. One team will focus more on Data cleaning, EDA and feature engineering, the other more on code structure design, pipeline building and testing. There will be frequent communication within a sub-team, but only one scheduled daily end-of-sprint communication for the full team.

b) Phase 2, four parallel working ants, 1-2 days sprints. After the initial phase, we would sync our understanding of data characteristics and algorithm pros and cons. We each pick one algorithm and move into four parallel data crunching tasks. Sprints are now longer. We would record our daily progress in the google doc, but only have a scheduled meeting every 1 or 2 days.


Minimum Deliverable
Exploratory Data Analysis Proper Procedure
Machine Learning Fast Track Pipeline
jupyter
learn
Two-Week Project Goal
Accurate Prediction
Industry Level Pipeline Building Experience
Desired Deliverable
Luigi
This Should Help
Agile + Lean Collaboration