The title of the talk can even be much longer than this

Dennis Koehn
Linxi Wang
Mingyang Li

Ladislaus von Bortkiewicz Chair of Statistics
C.A.S.E. – Center for Applied Statistics
and Economics
Humboldt–Universität zu Berlin
http://lvb.wiwi.hu-berlin.de
www.case.hu-berlin.de

Outline

1. Introduction ✓
2. Pre-processing Steps
3. Model Selection
4. Variable Importance and Dimensionality Reduction
5. Results
6. Conclusion and Outlook

Formal Problem Setting

- *training set*: inputs $X = (x_1, \ldots, x_n) \in \mathbb{R}^{n \times d}$ and labels $Y = (y_1, \ldots, y_n) \in \mathbb{R}^n$
- *test set*: inputs $X' = (x'_1, \ldots, x'_t) \in \mathbb{R}^{t \times d}$ without labels

Find a function

$$f : X \to Y \tag{1}$$

s.t. the labels of *test set* are predicted as accurately as possible, i.e.

$$f(X') \approx Y' \tag{2}$$

Ames House Price Data

Characteristics and Sale Price of Houses in Ames, Iowa provided by kaggle.com:

- 79 variables as inputs plus sale price in the training set
- 1460 Observations in training set
- 1459 Observations in test set (thus 1459 labels to predict)

Only Kaggle knows the 1459 Labels of the test set. We submitted our predictions at the website and obtain the RMSE of the log labels.

$$RMSE = \sqrt{\frac{1}{t}\sum_{i=1}^{t}\left(log(\hat{y}_i) - log(y_i')\right)^2}$$

Pre-processing

Several transformations and cleaning steps needed before putting the data into an algorithm, e.g.

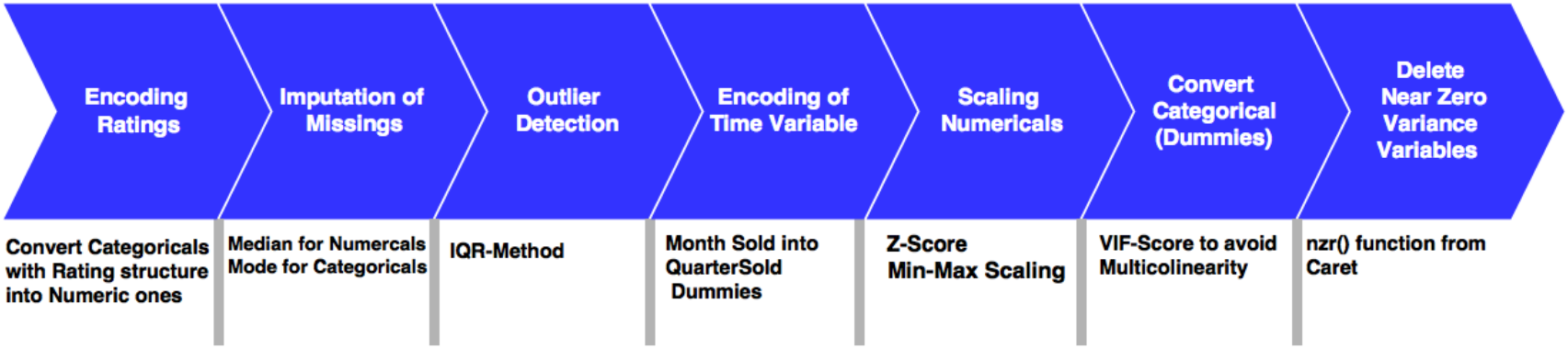

Figure 1: Workflow of Pre-Processing Steps

All transformation need to be preformed on the test set as well!

```
1  basic_preprocessing = function(X_com, y, scaler="
     gaussian")
2  {
3      source("replace_ratings.R")
4      source("convert_categoricals.R")
5      source("impute_data.R")
6      source("encode_time_variables.R")
7      source("impute_outliers.R")
8      source("scale_data.R")
9      source("delete_nearzero_variables.R")
10     X_ratings = replace_ratings(X_com)
11     X_imputed = naive_imputation(X_ratings)
12     X_no_outlier = data.frame(lapply(X_imputed,
         iqr_outlier))
13     X_time_encoded = include_quarter_dummies(
         X_no_outlier)
```

dataProcessing

```
    X_scaled = scale_data(X_time_encoded,
      scale_method = scaler)
    X_encoded = data.frame(lapply(X_scaled,
      cat_to_dummy))
    X_com = delect_nz_variable(X_encoded)
    idx_train = c(1:length(y))
    train = cbind(X_com[idx_train, ], y)
    test = X_com[-idx_train, ]
    return(list(train = train, X_com = X_com, test =
       test))
}
```

dataProcessing

Considered Algorithms

- Random Forest from `library('h2o')`
  - tuning `maxdepth`, `gamma` and `sample_size`
  - determining `ntree` through `early_stopping()` option
- Gradient Boosting Machines from `library('xgboost')`
  - tuning `max_depth`, `gamma` `subsample` and `col_by_tree`
  - determining `nrounds` through `early_stopping()` option
- Support Vector Regression with Gaussian Kernel from `library('kernlab')`
  - tuning `lambda` and `sigma` via `caret`

Optimizing Hyper-parameters

Algorithm 1: t-time k-fold crossvalidation and gridSearch

```
foreach i in 1:t do
    Randomly split the data into k folds of the same size
    foreach j in 1:k do
        Use jth fold as test set and the union of remaining folds as training set
        foreach p in 1:grid do
            Fit model on training set using parameter set p
            Predict on test set and calculate RMSE
        end
    end
    foreach p in 1:grid do
        Calculate average RMSE over the t × k-runs
    end
    choose p with the lowest RMSE
end
```

xgbTuning rfTuning svmTuning

GBM tunning results

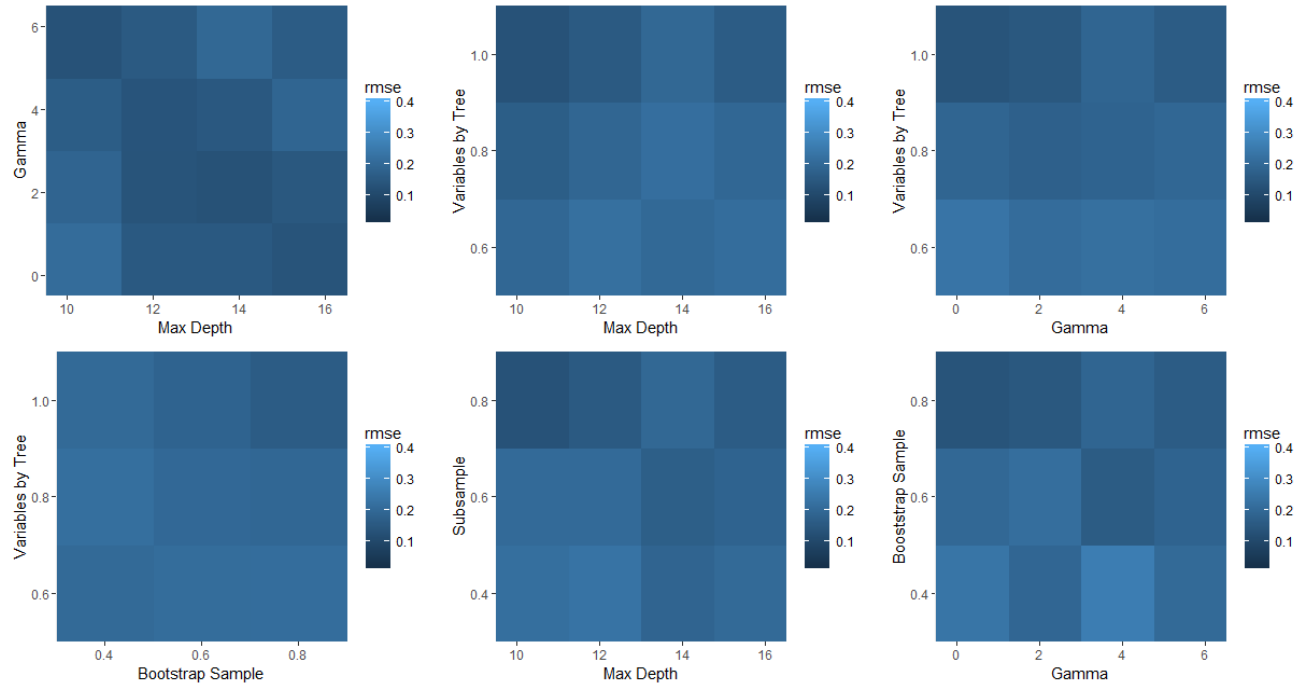

Figure 2: Heatmap of GBM tuning results

xgbTuning

Taking on the curse of dimensionality

Problem:

- many variable (99 after pre-processing)
- small training set ($n = 1460$)
- variables are correlated with each other

Our approaches:

- Variable selection through variable importance ranking
- Extract a smaller set of variable using PCA

Variable Importance of Tree-Based Methods

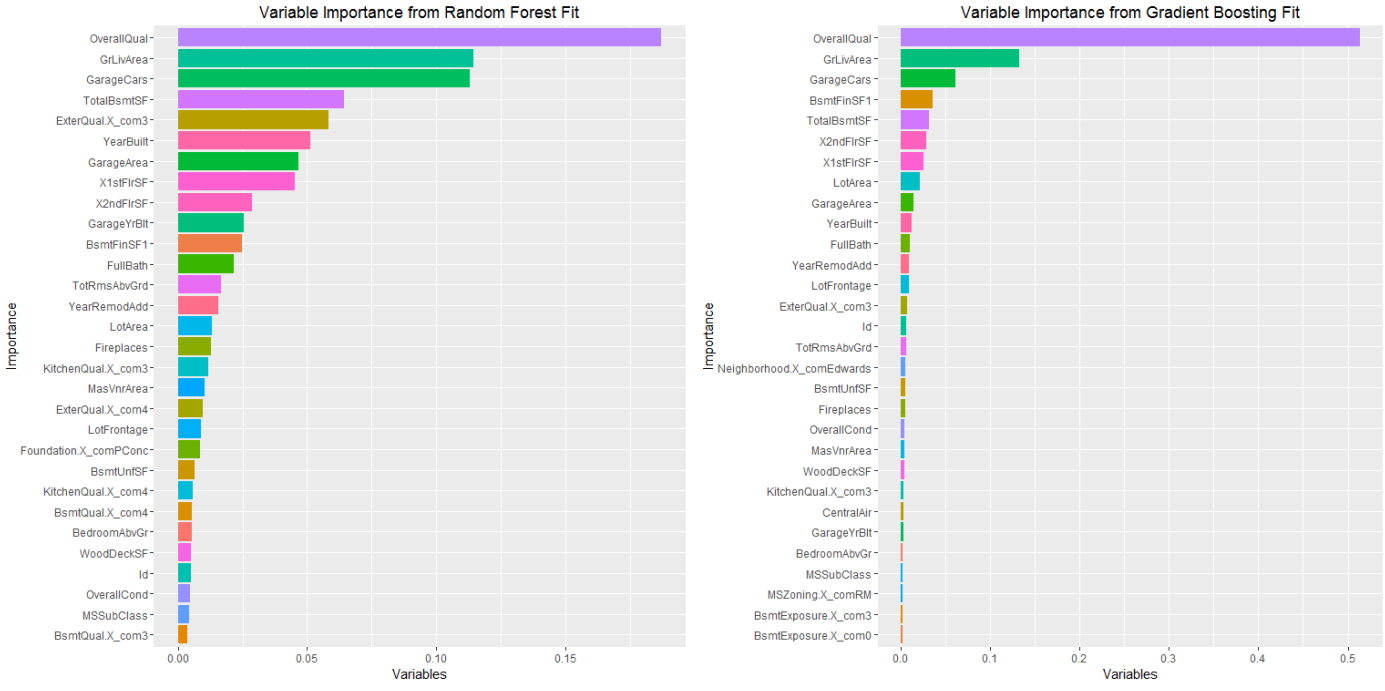

Figure 3: RF vs. GBM Variable Importance

rfFeatureSelection

xgbFeatureSelection

Recursive Feature Elimination using SVR

- Recursive Feature Elimination suggests that the full set of variables performs best

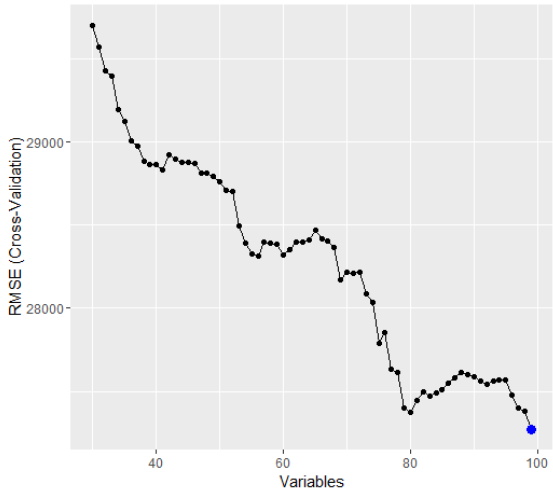

Figure 4: RFE using SVR with Gaussian Kernel

svmRFE

Principal Component Analysis

- We use the first 55 principal components as input data, which make up 0.8 of the total variance

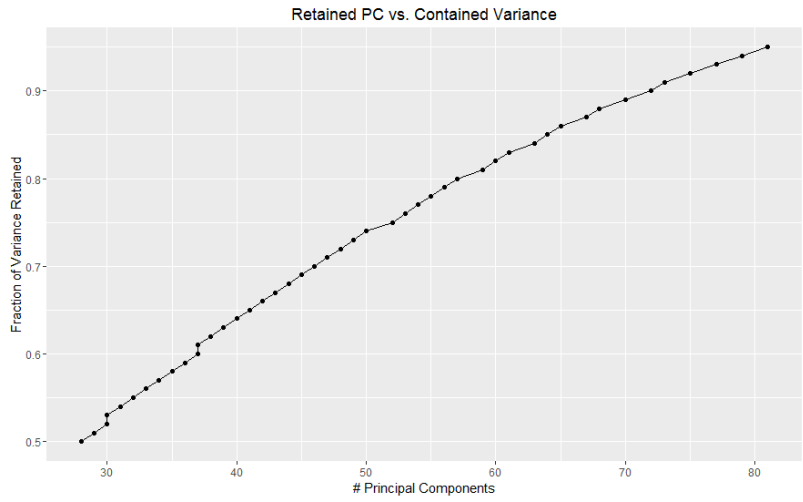

Figure 5: Reverse Elbow Plot

PCA

Results

- Gaussian SVR with all variable is the single best model
- PCA did not work well
- Models perform best with the full set of variables as Figure 4 suggested

Inputs	Gaussian SVR	Random Forest	GBM
All Variables	**0.1308**	0.1484	0.1333
Top 30	0.1323	0.1515	0.1436
PCA	0.1607	0.1657	0.1657

Table 1: RMSE of submitted predictions

finalModels

Conclusion and Outlook

- Unexpected result: SVR outperforms the tree-based ensembles
  - SVR benefits more from pre-processing steps e.g. outlier detection and scaling
- further improvement possible by
  - building ensembles different models
  - detailed feature engineering

References

Blei, David M., Andrew Y. Ng, and Michael I. Jordan
"Latent dirichlet allocation." Journal of machine Learning research:993-1022
available on http://www.jmlr.org, 2003

Blei, David M., and John D. Lafferty
"Topic models." Text mining: classification, clustering, and applications
available on http://www.seas.harvard.edu, 2009

Blei, David M
"Probabilistic topic models." Communications of the ACM 55.4: 77-84.
available on http://dl.acm.org, 2012