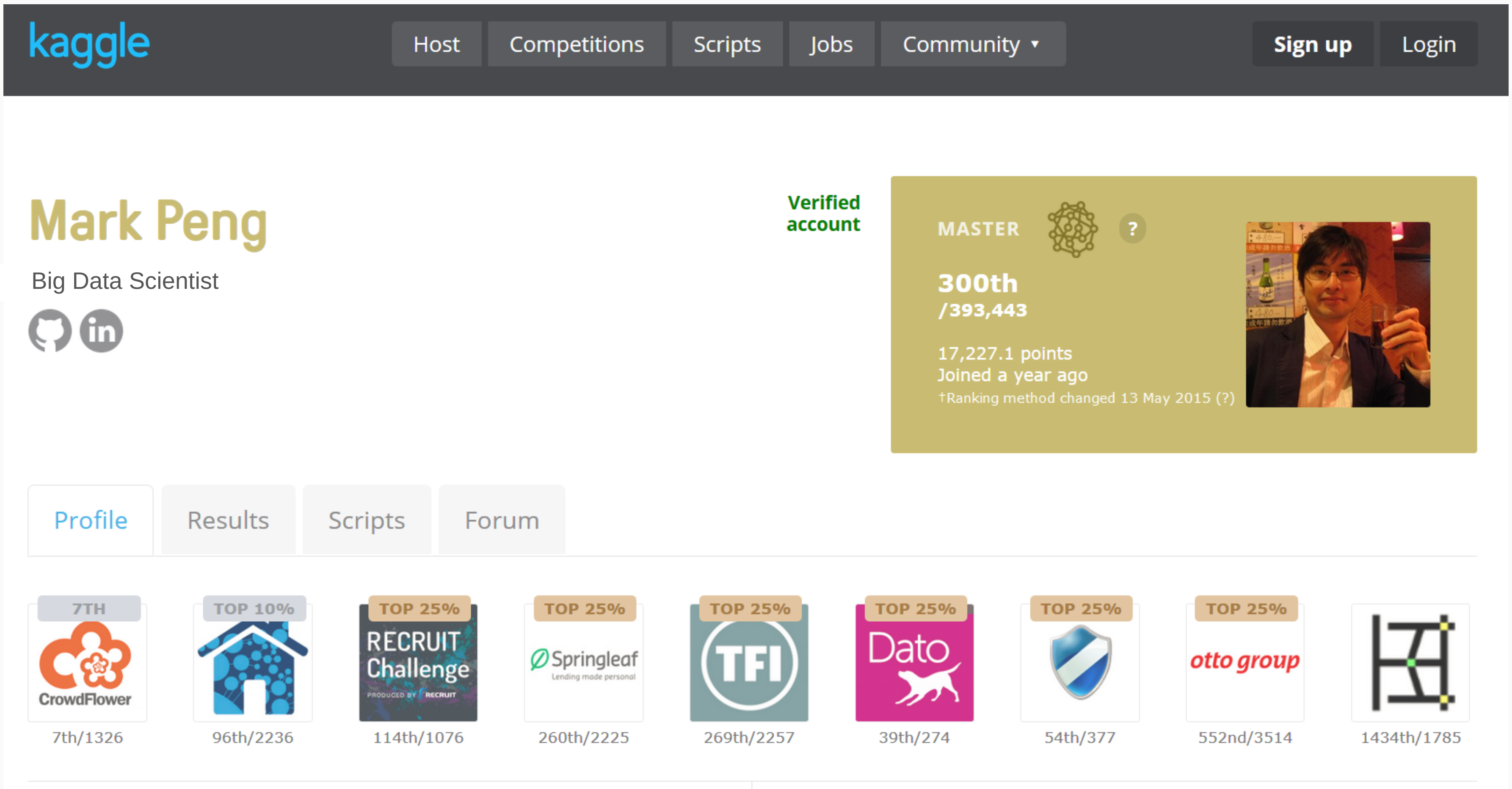
General Tips for participating kaggle Competitions

Mark Peng

2015.12.16 @ Spark Taiwan User Group

https://tw.linkedin.com/in/markpeng

Kaggle Profile (Master Tier)

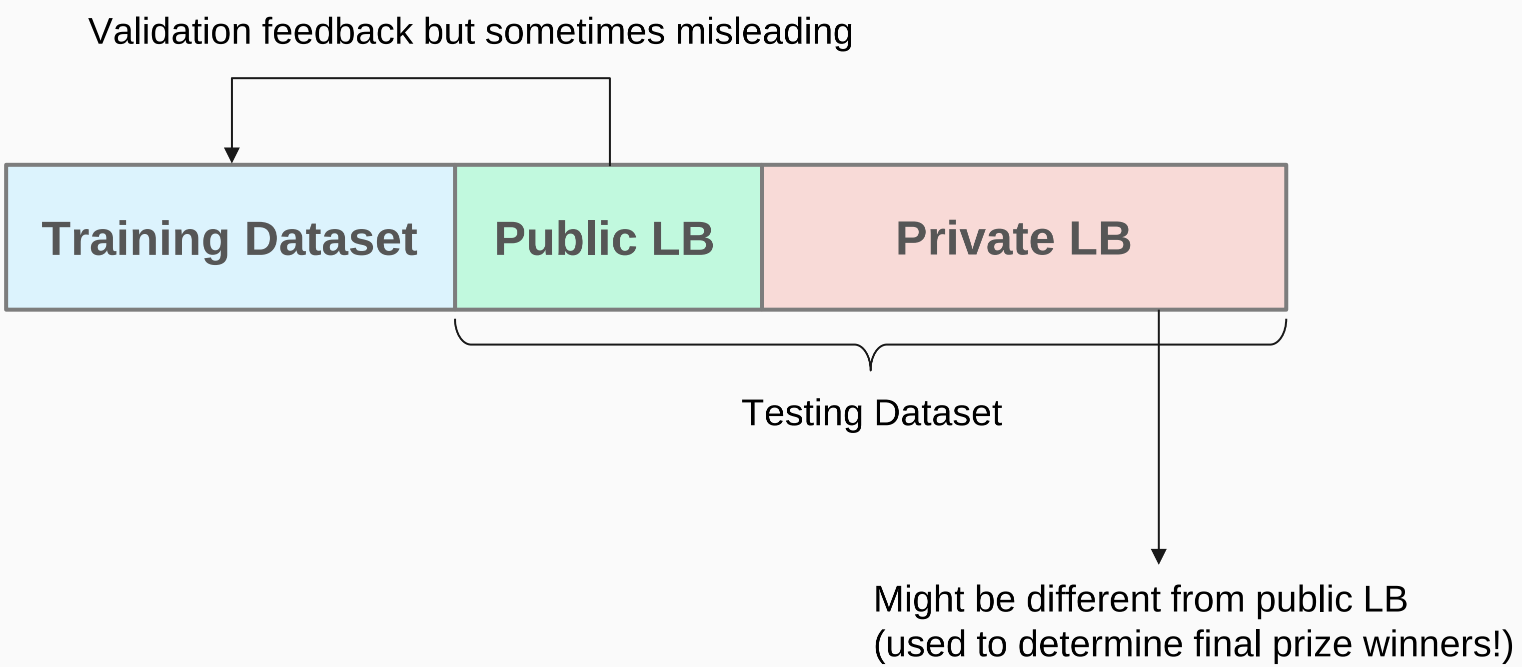


https://www.kaggle.com/markpeng

Things I Want to Share

- Quick overview to Kaggle rules
- Why cross-validation matters
- Mostly used ML models
- Feature engineering methods
- Ensemble learning
- Team up
- Recommended books, MOOCs and resources

Kaggle Competition Dataset and Rules



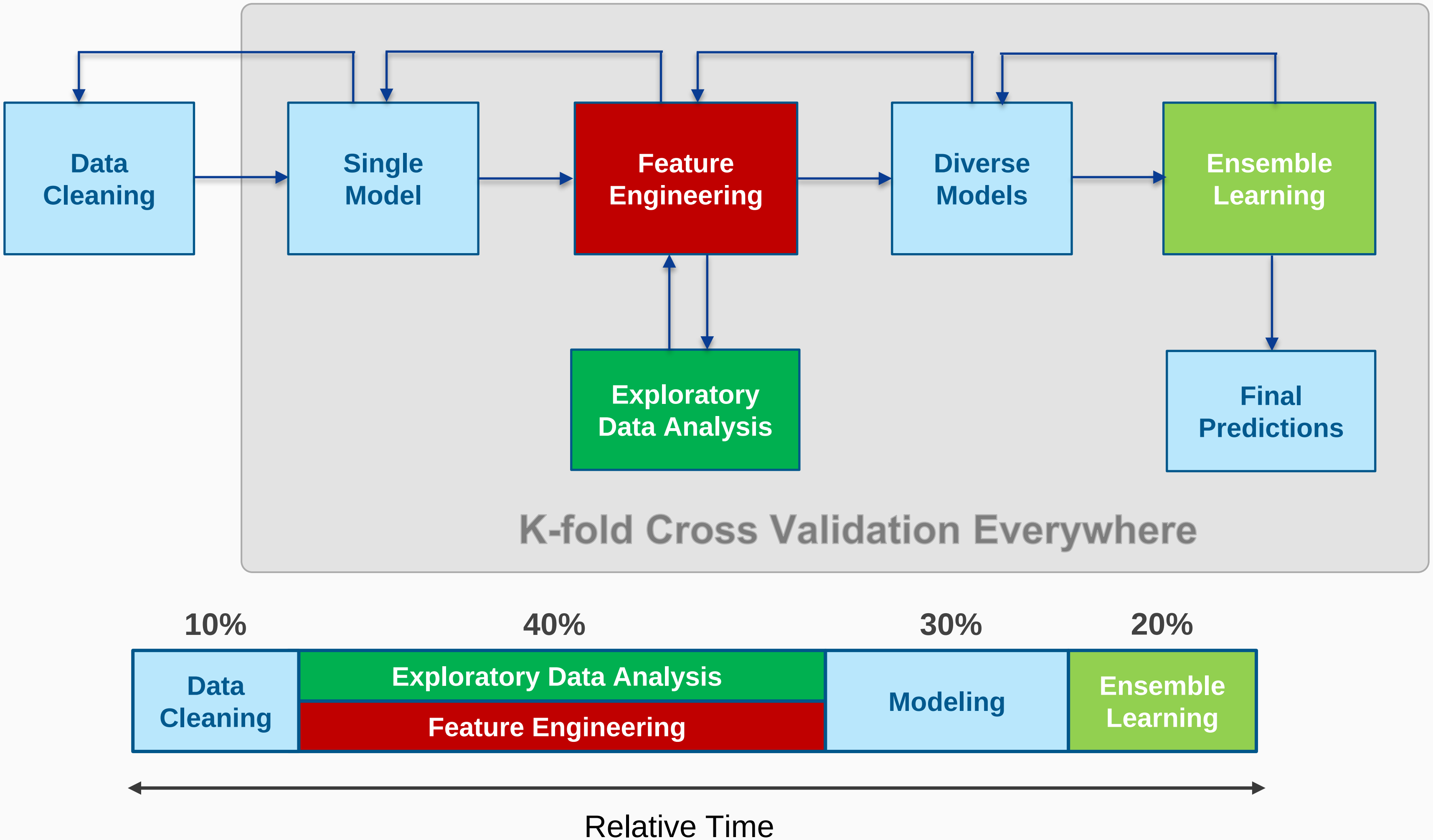
How to become a Kaggle Master

- To achieve Master tier, you must fulfill 2 criteria
  - *Consistency*: at least 2 Top 10% finishes in public competitions
  - *Excellence*: at least 1 of those finishes in the top 10 positions
- Note that not all competitions count toward earning Master tier!

Started: 7:29 pm, Monday 9 November 2015 UTC
Ends: 11:59 pm, Monday 8 February 2016 UTC (91 total days)
Points: this competition awards standard ranking points
Tiers: this competition counts towards tiers

Reference: https://www.kaggle.com/wiki/UserRankingAndTierSystem

Recommended Data Science Process (IMHO)

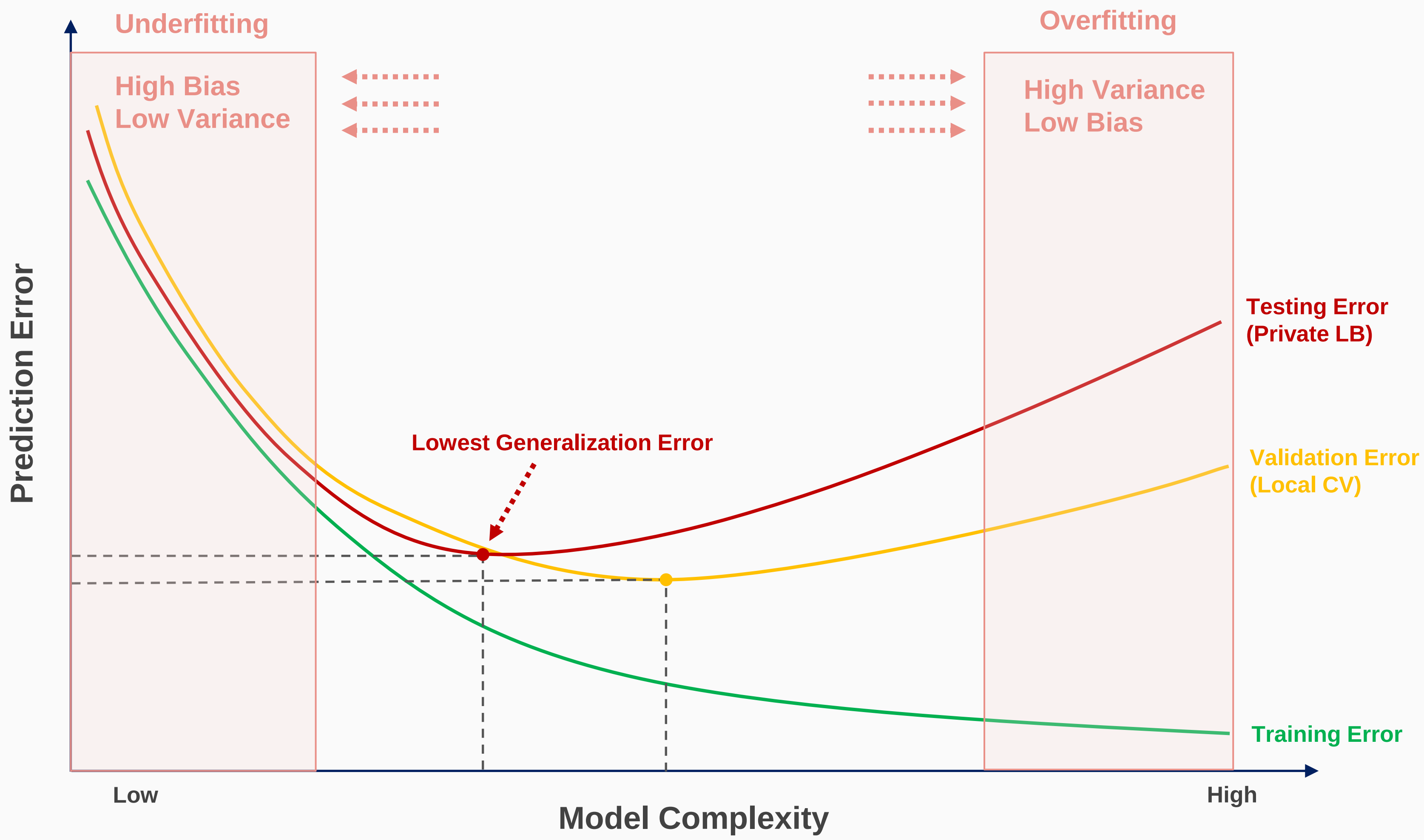


Cross Validation

The key to avoid overfitting

Underfitting and Overfitting

We want to find a model with lowest generalization error (hopefully)



Big Shake Up on Private LB!

TFI TAB FOOD INVESTMENTS

Completed • $30,000 • 2,257 teams

Restaurant Revenue Prediction

Mon 23 Mar 2015 – Mon 4 May 2015 (7 months ago)

Dashboard ▼

Private Leaderboard - Restaurant Revenue Prediction

This competition has completed. This leaderboard reflects the final standings.

See someone using multiple accounts? Let us know.

#	Δrank	Team Name ‡ model uploaded * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑205	Arsenal ‡ *	1727811.48554	21	Fri, 24 Apr 2015 03:14:27 (-23.9d)
2	↑42	Climent_Josep ‡ *	1729265.36129	27	Mon, 04 May 2015 06:32:55 (-28.2h)
3	↑338	OldSchool ‡ *	1732292.38960	72	Tue, 28 Apr 2015 08:53:55 (-3.9d)
4	↑170	McData	1734366.17757	53	Fri, 01 May 2015 06:01:48
5	↑294	Statsfreak			
6	↑61	Manuel Díaz (TFI)			
7	↑482	mj_coder			
8	↑345	Rynsin			
9	↑1058	Chapulist			
10	↑48	Pete_Ter			

687	↓125	Yinghao	1837568.63872	20	Wed, 22 Apr 2015 14:02:49 (-44.9h)
688	↓683	Analytic Bastard	1837732.75137	115	Mon, 04 May 2015 23:23:50 (-39.1h)
689	↓535	david li	1838065.25023	48	Tue, 28 Apr 2015 06:18:24 (-16.2d)
690	↑752	Caner	1838149.98065	4	Sun, 29 Mar 2015 07:02:49 (-5.1d)
691	↓246	Nithin	1838160.15572	9	Fri, 01 May 2015 18:16:20 (-7.1d)
692	↓399	kalyan_TFI	1838249.27788	29	Mon, 04 May 2015 11:42:58 (-0h)
693	↑16	Axel	1838258.11833	15	Fri, 27 Mar 2015 17:14:32 (-3.8d)
694	↓268	Nilesh Kadam	1838303.64440	4	Sun, 19 Apr 2015 15:44:32

Reference: https://www.kaggle.com/c/restaurant-revenue-prediction/leaderboard/private

Who is the King of Overfitting?

Public LB RMSE:

1	↑1794	BAYZ, M.D. *	0.00000	115	Mon, 04 May 2015 22:51:37 (-40.9h)

Private LB RMSE:

1962	↓1961	BAYZ, M.D.	2108790.68755	115	Mon, 04 May 2015 22:51:37

All Forums » Restaurant Revenue Prediction

Search

« Prev Topic

Our perfect submission

Next Topic »

Start Watching | View all posts

1 2 3 4 >

107

Hello fellow Kagglers!

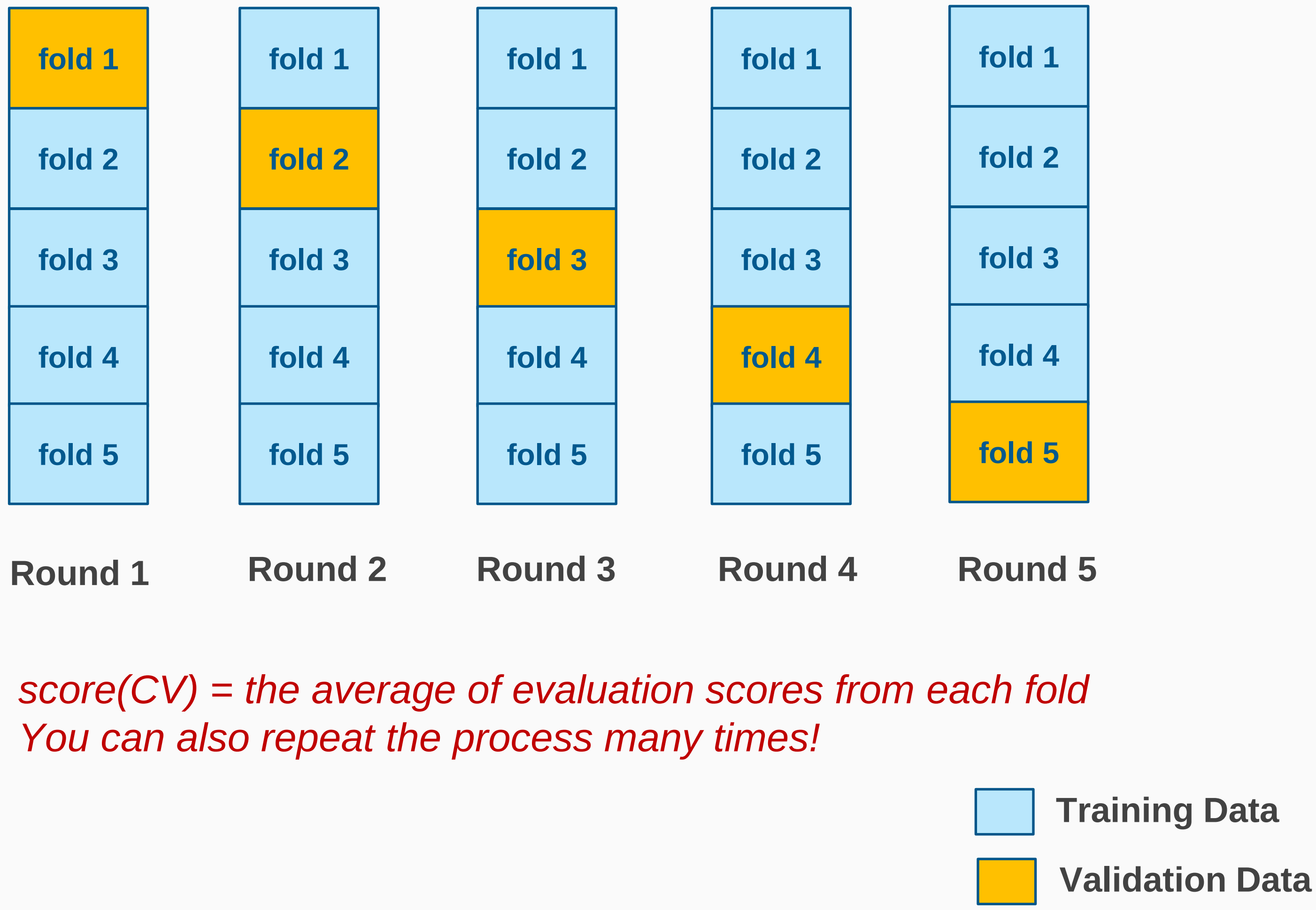
I am from team BAYZ, which as you can see currently occupies the top spot on the leaderboard with a PERFECT score (0.00000). You may be curious what's going on! Some of you were already wondering with our previous score (816496.58093) whether it was spurious and whether it would carry over to new data.

Num. of Features: 41
Training Data Size: 137
Testing Data Size: 10,000

They even wrote a post to show off their perfect overfitting!

Reference: https://www.kaggle.com/c/restaurant-revenue-prediction/forums/t/13950/our-perfect-submission

K-fold Cross Validation (K = 5)

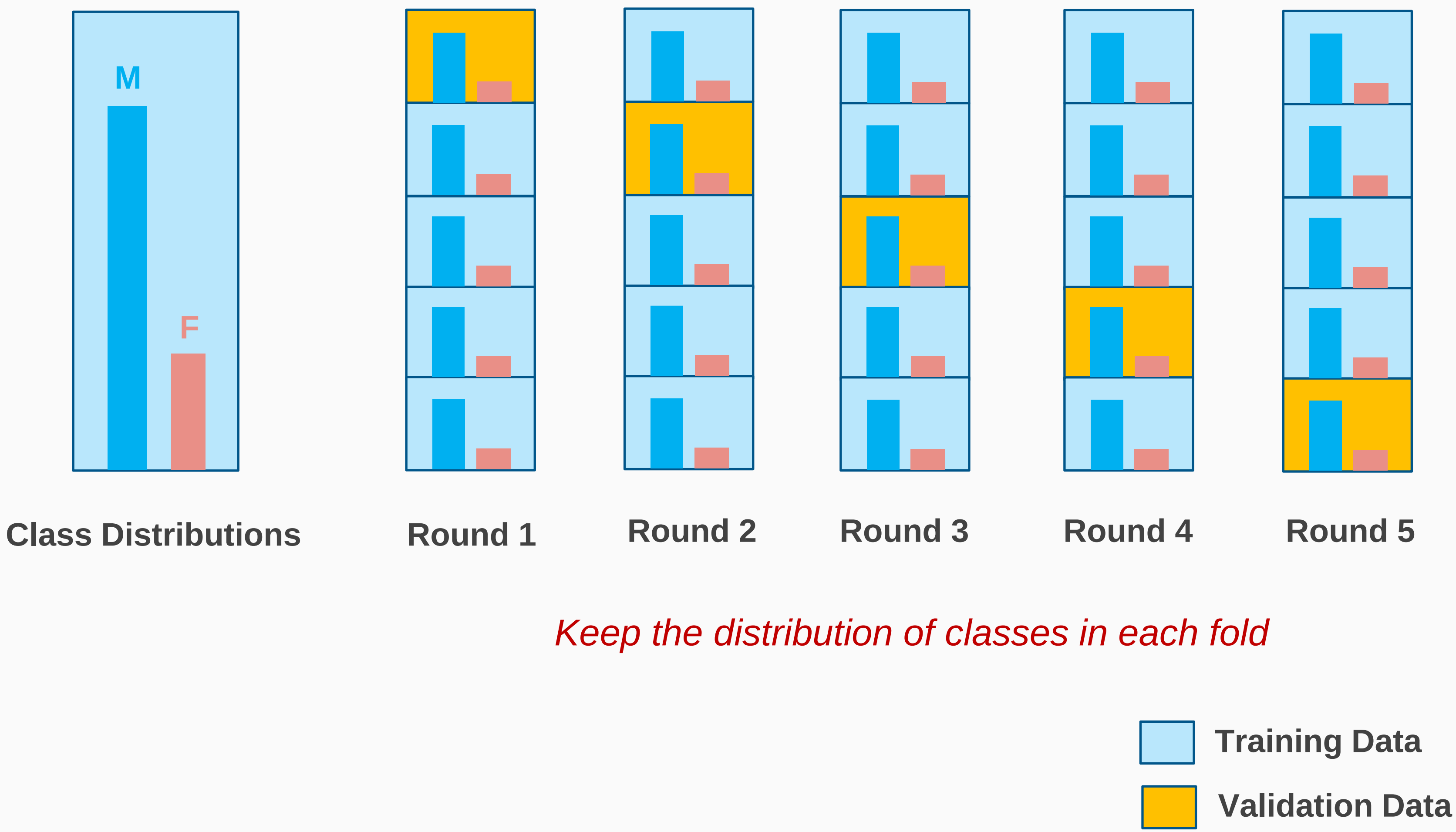


score(CV) = the average of evaluation scores from each fold
You can also repeat the process many times!

K-fold Cross Validation Tips

- It is normal to experience big shake up on private LB if not using local CV correctly
- 5 or 10 folds are not always the best choices (you need to consider the cost of computation time for training models)
- Which *K* to choose?
    - Depends on your data
    - Mimic the ratio of training and testing in validation process
    - Find a *K* with lowest gap between local CV and public LB scores
- Standard deviation of K-fold CV score matters more than mean!
- *Stratified K-fold CV* is important for imbalanced dataset, especially for classification problems

Stratified K-fold Cross Validation (K = 5)



Keep the distribution of classes in each fold

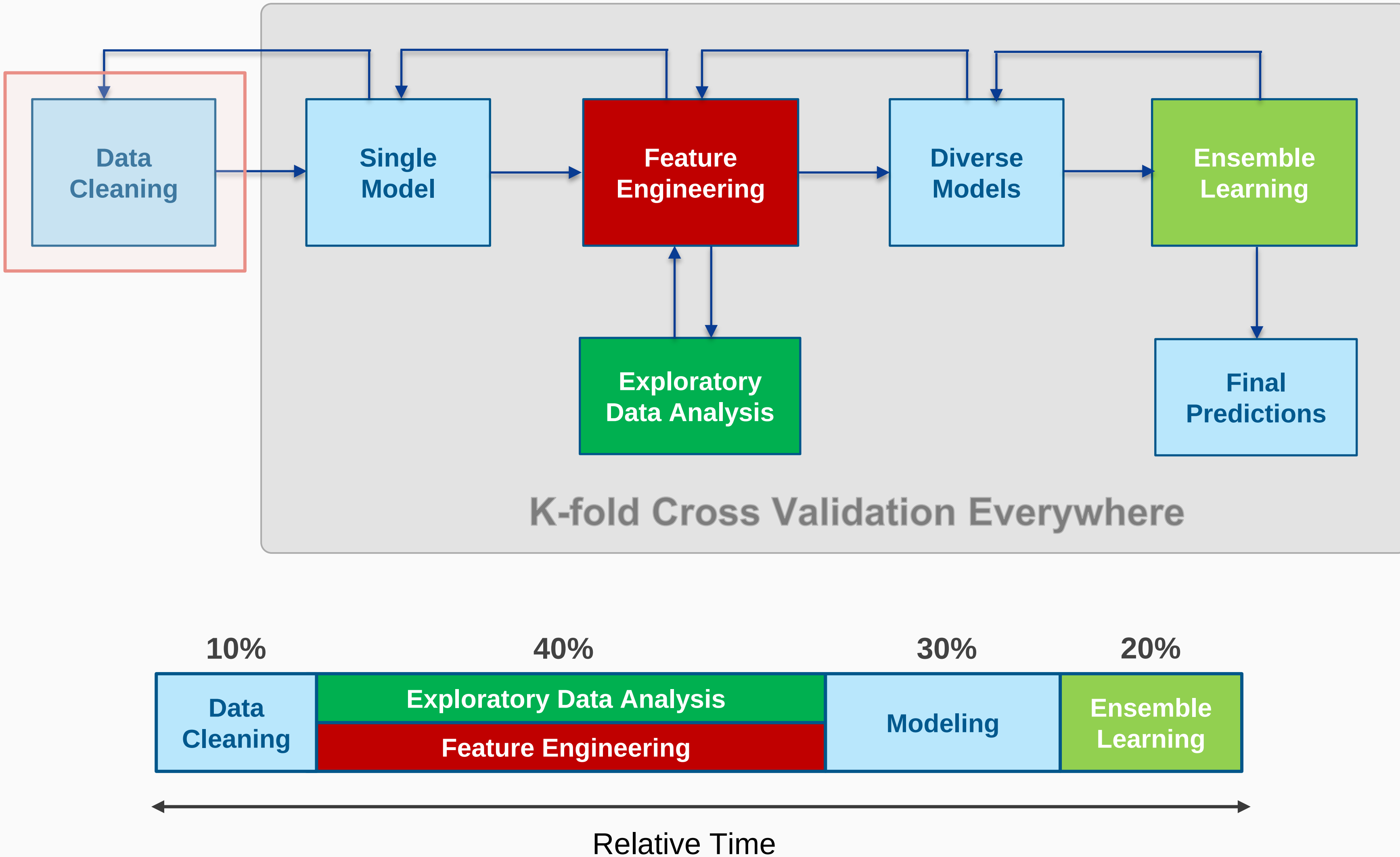
Should I Trust Public LB?

- Yes if you can find a *K*-fold CV that follows the same trend with public LB
  - High positive correlation between local CV and public LB
  - The score increases and decreases in both local CV and public LB
- Trust more in your local CV!

Data Cleaning

Making data more analyzable

Recommended Data Science Process (IMHO)



Data Cleaning Techniques

- Data cleaning is the removal of duplicates, useless data, or fixing of missing data
- Reduce dimensional complexity of dataset
  - Make training faster without hurting the performance
- Apply imputation methods to help (hopefully) utilize incomplete rows
  - Incomplete rows may contain relevant features (don’t just drop them!)
  - In the risk of distorting original data, so be cautious!

Data Cleaning Techniques (cont.)

- **Remove duplicate features**
  - Columns having the same value distribution or variance
  - Only need to keep one of them
- **Remove constant features**
  - Columns with only one unique value
  - R: `sapply(data, function(x) length(unique(x)))`
- **Remove features with near-zero variance**
  - R: `nearZeroVar(data, saveMetrics=T)` in *caret* package
- Be sure to know what you are doing before removing any features

Data Cleaning Techniques (cont.)

- **Some machine learning tools cannot accept NAs in the input**
- **Encode missing values to avoid NAs**
  - Binary features
    - -1 for negatives, 0 for missing values and 1 for positives
  - Categorical features
    - Encode as an unique category
    - “Unknown”, “Missing”, ….
  - Numeric features
    - Encode as a big positive or negative number
    - 999, -99999, ….

Data Cleaning Techniques (cont.)

- **Basic ways to impute missing values**
  - mean, median or most frequent value of feature
  - R: `impute(x, fun = mean)` in Hmisc package
  - Python: `Imputer(strategy='mean', axis=0)` in scikit-learn package

Data Cleaning Techniques (cont.)

- **Advanced: multiple imputation**
  - Impute incomplete columns based on other columns in the data
  - R: *mice* package (Multivariate Imputation by Chained Equations)
- Imputation would not always give you positive improvements, thus you have to validate it cautiously

Mostly Used Models

What models to use?

Mostly Used ML Models

Model Type	Name	R	Python
Regression	Linear Regression	• glm, glmnet	• sklearn.linear_model.LinearRegression
	Ridge Regression	• glmnet	• sklearn.linear_model.Ridge
	Lasso Regression	• glmnet	• sklearn.linear_model.Lasso
Instance-based	*K*-nearest Neighbor (KNN)	• knn	• sklearn.neighbors.KNeighborsClassifier
	Support Vector Machines (SVM)	• svm {e1071} • LiblinearR	• sklearn.svm.SVC, sklearn.svm.SVR • sklearn.svm.LinearSVC, sklearn.svm.LinearSVR
Hyperplane-based	Naive Bayes	• naiveBayes {e1071}	• sklearn.naive_bayes.GaussianNB • sklearn.naive_bayes.MultinomialNB • sklearn.naive_bayes.BernoulliNB
	Logistic Regression	• glm, glmnet • LiblinearR	• sklearn.linear_model.LogisticRegression
Ensemble Trees	Random Forests	• randomForest	• sklearn.ensemble.RandomForestClassifier • sklearn.ensemble.RandomForestRegressor
	Extremely Randomized Trees	• extraTrees	• sklearn.ensemble.ExtraTreesClassifier • sklearn.ensemble.ExtraTreesRegressor
	Gradient Boosting Machines (GBM)	• gbm • xgboost	• sklearn.ensemble.GradientBoostingClassifier • sklearn.ensemble.GradientBoostingRegressor • xgboost
Neural Network	Multi-layer Neural Network	• nnet • neuralnet	• PyBrain • Theano
Recommendation	Matrix Factorization	• NMF	• nimfa
	Factorization machines		• pyFM
Clustering	*K*-means	• kmeans	• sklearn.cluster.KMeans
	t-SNE	• Rtsne	• sklearn.manifold.TSNE

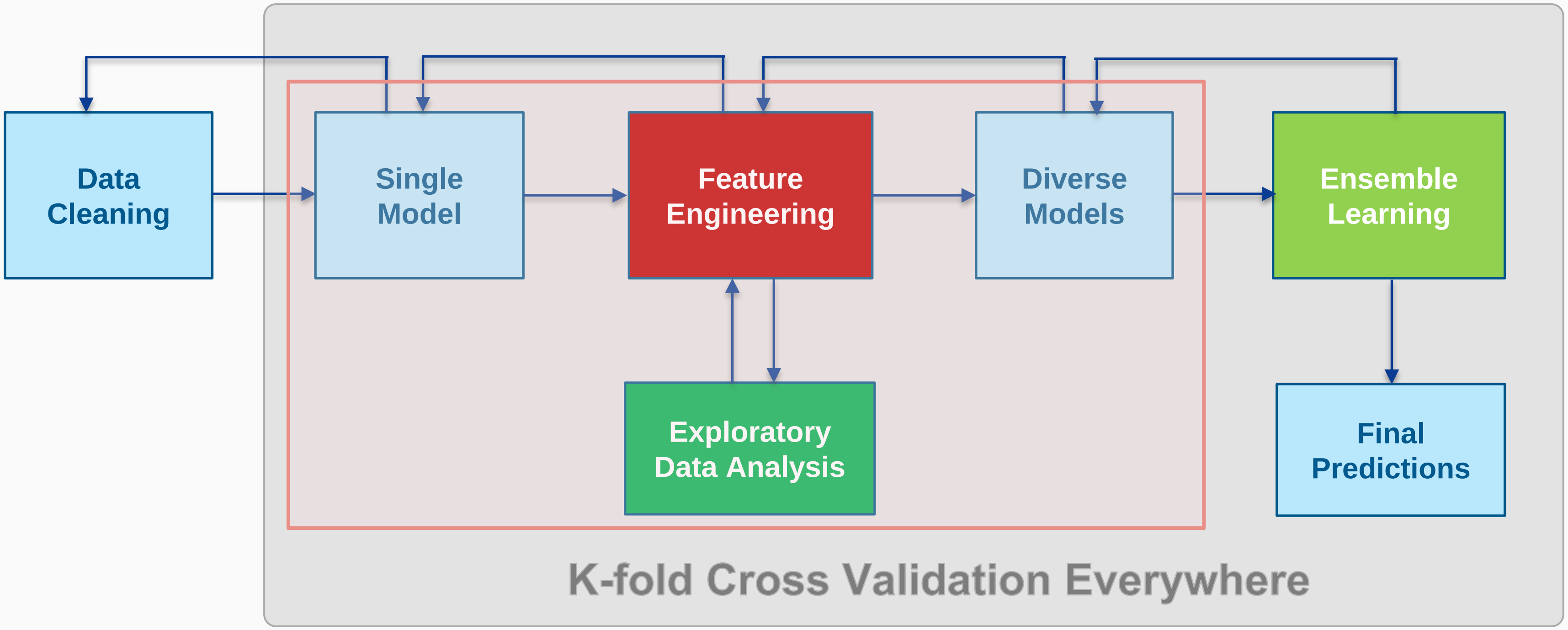
Mostly Used ML Models (cont.)

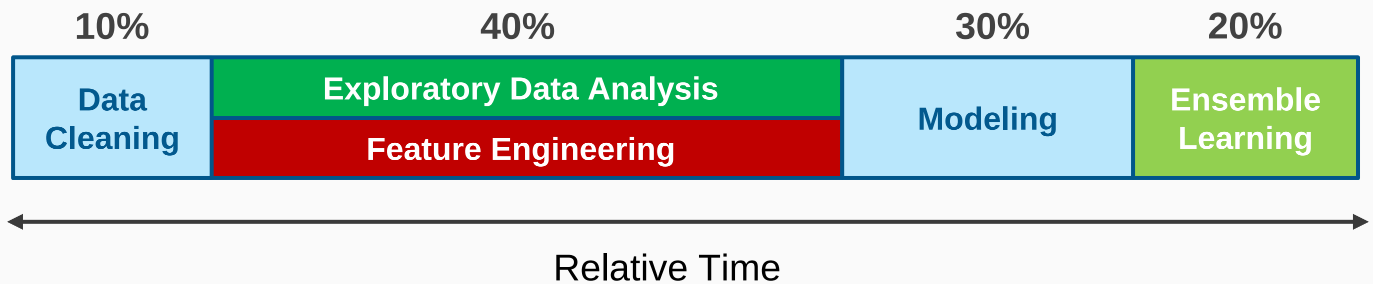
Model Type	Name	Mahout	Spark
Regression	Linear Regression		• LinearRegressionWithSGD
	Ridge Regression		• RidgeRegressionWithSGD
	Lasso Regression		• LassoWithSGD
Instance-based	*K*-nearest Neighbor (KNN)		
	Support Vector Machines (SVM)		• SVMWithSGD (only linearSVM)
Hyperplane-based	Naive Bayes	• trainnb • spark-trainnb	• NaiveBayes
	Logistic Regression	• runlogistic • trainlogistic	• LogisticRegressionWithLBFGS • LogisticRegressionWithSGD
Ensemble Trees	Random Forests	• org.apache.mahout.classifier.df.mapreduce.BuildForest	• RandomForest
	Extremely Randomized Trees		
	Gradient Boosting Machines (GBM)		• GradientBoostedTrees
Neural Network	Multi-layer Neural Network		
Recommendation	Matrix Factorization	• parallelALS	• ALS
	Factorization machines		
Clustering	*K*-means	• kmeans	• KMeans
	t-SNE		

Feature Engineering

The key to success

Recommended Data Science Process (IMHO)





Feature Engineering

- Extract more new *gold features,* remove irrelevant or noisy features
  - Simpler models with better results
- The most important factor for the success of machine learning
- Key Elements
  - Data Transformation
  - Feature Encoding
  - Feature Extraction
  - Feature Selection

Data Transformation

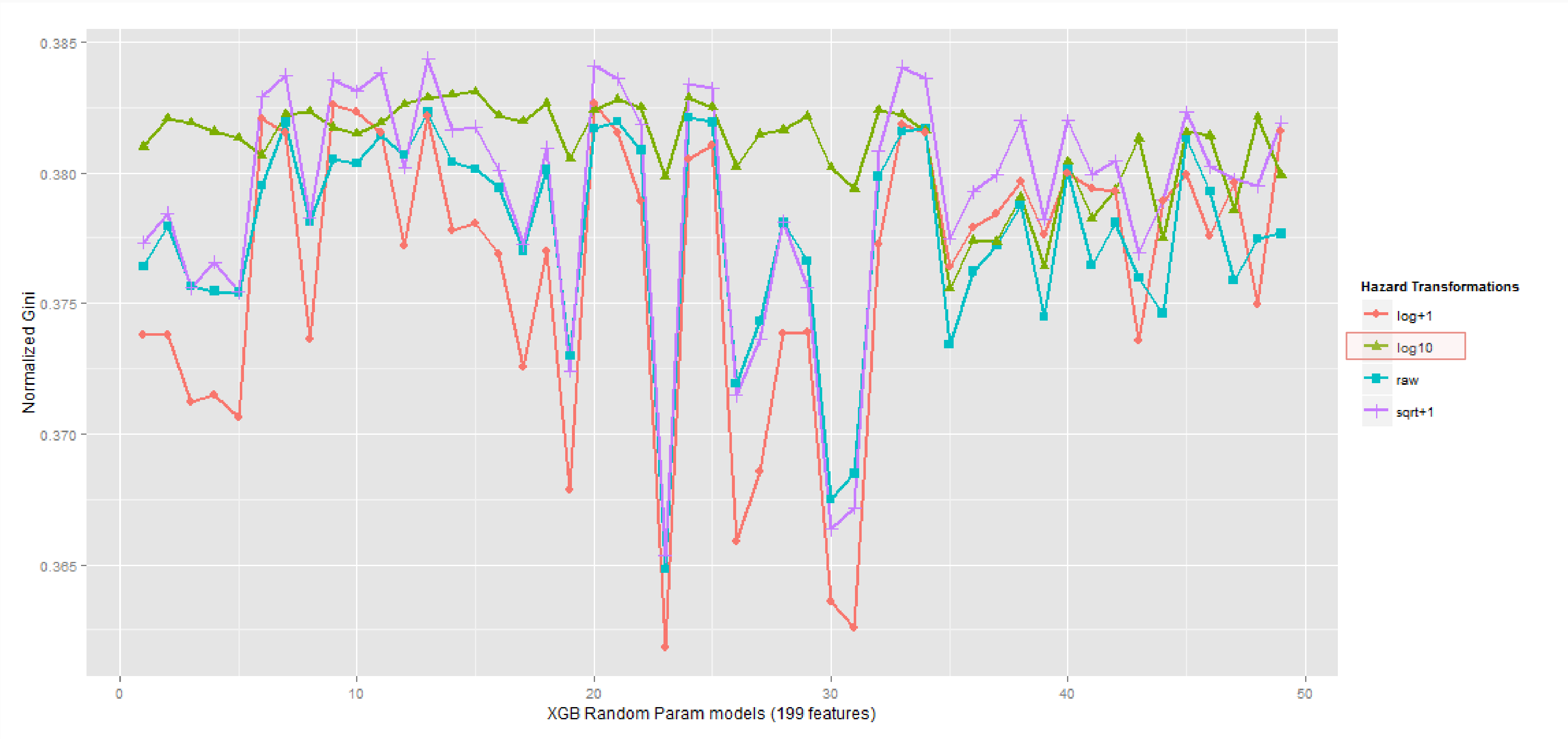
- Feature Scaling
  - Rescaling
    - Turn numeric features into the same scale (e.g., [-1,+1], [0,1], …)
    - Python scikit-learn: `MinMaxScaler`
  - Standardization
    - Removing the mean ($\mu = 0$) and scaling to unit variance ($\sigma = 1$)
    - Python scikit-learn: `StandardScaler`
    - R: `scale`
  - To avoid features in greater numeric ranges dominating those in smaller numeric ranges
  - Critical for *regularized linear models*, *KNN*, *SVM*, *K-Means*, etc.
  - Can make a big difference to the performance of some models
  - Also speed up the training of gradient decent
  - But not necessary to do it for tree-based models

Data Transformation (cont.)

- Predictor/Response Variable Transformation
  - Use it when variable shows a skewed distribution
  - make the *residuals* more close to “normal distribution” (bell curve)
  - Can improve model fit
  - log(x), log(x+1), sqrt(x), sqrt(x+1), etc.

Variable Transformation

In Liberty Mutual Group: Property Inspection Prediction



Different transformation of target response variable may lead to different results

Feature Encoding

- Turn categorical features into numeric features to provide more fine-grained information
  - Help explicitly capture non-linear relationships and interactions between the values of features
  - Some machine learning tools only accept numbers as their input
    - xgboost, gbm, glmnet, libsvm, liblinear, etc.

Feature Encoding (cont.)

- Labeled Encoding
  - Interpret the categories as ordered integers (mostly wrong)
  - Python scikit-learn: **`LabelEncoder`**
- One Hot Encoding
  - Transform categories into individual binary (0 or 1) features
  - Python scikit-learn: **`DictVectorizer`**, **`OneHotEncoder`**
  - R: **`dummyVars`** in *caret* package

Feature Extraction: HTML Data

- HTML files are often used as the data source for classification problems
  - For instance, to identify whether a given html is an AD or not
- Possible features inside
  - html attributes (id, class, href, src, ...)
  - tag names
  - inner content
  - javascript function calls, variables, ...
  - script comment text
- Parsing Tools
  - *Jsoup* (Java), *BeautyfulSoup* (Python), etc.

Feature Extraction: Textual Data

- Bag-of-Words: extract tokens from text and use their occurrences (or TF/IDF weights) as features
- Require some NLP techniques to aggregate token counts more precisely
    - Split token into sub-tokens by delimiters or case changes
    - N-grams at word (often 2-5 grams) or character level
    - Stemming for English words
    - Remove stop words (not always necessary)
    - Convert all words to lower case
    - ....
- Bag-of-Words Tools
    - R: *tm* package
    - Python: *CountVectorizer*, *TfidfTransformer* in scikit-learn package
    - Java: *Lucene*

Feature Extraction: Textual Data (cont.)

- Deep Learning for textual data
  - Turn each token into a vector of predefined size
  - Help compute "*semantic distance*" between tokens/words
    - For example, the semantic distance between user query and product titles in search results (how relevant?)
  - Greatly reduce the number of text features used for training
    - Use average vector of all words in given text
    - Vector size: 100~300 is often enough
  - Tools
    - Deeplearning4j (support Spark with GPUs)
      - Word2vec, Doc2vec
      - GloVe
    - Theano (support GPUs)

Feature Extraction

- There usually have some meaningful features inside existing features, you need to extract them manually
- Again you can use counts as features
- Some examples
  - Location
    - Address, city, state and zip code .... (categorical or numeric)
  - Time
    - Year, month, day, hour, minute, time ranges, ... (numeric)
    - Weekdays or weekend (binary)
    - Morning, noon, afternoon, evening, ... (categorical)
  - Numbers
    - Turn age numbers into ranges (ordinal or categorical)

Feature Extraction: Simple But Powerful

- Counts of symbols and spaces inside text can also be powerful features!

« Prev Topic

Beat the Benchmark 0.90388 with simple model

Next Topic »

Stop Watching | View all posts

1 2 3 >

42

In response **to this post** and to drum up a little more interest in this competition, I'm posting some starter code that performs decently (0.90388), needs quite a bit of work, leaves room for a lot of improvement, and it is a really basic model (just 7 features with RandomForest), showing that many competitors are ignoring some really basic features. It runs under 8 minutes from start to finish on a modern Mac Book Pro (uses multiprocessing, so runs 8 processes in parallel). Just in case you're wondering, I'm not a proponent of high performance btb code this close to competition end, but really, just 7 basic count features?

Enjoy!

My feature set:

```
values['lines'] = text.count('\n')
values['spaces'] = text.count(' ')
values['tabs'] = text.count('\t')
values['braces'] = text.count('{')
values['brackets'] = text.count('[')
values['words'] = len(re.split('\s+', text))
values['length'] = len(text)
```

Using only 7 simple count features gives you over 0.90 AUC!

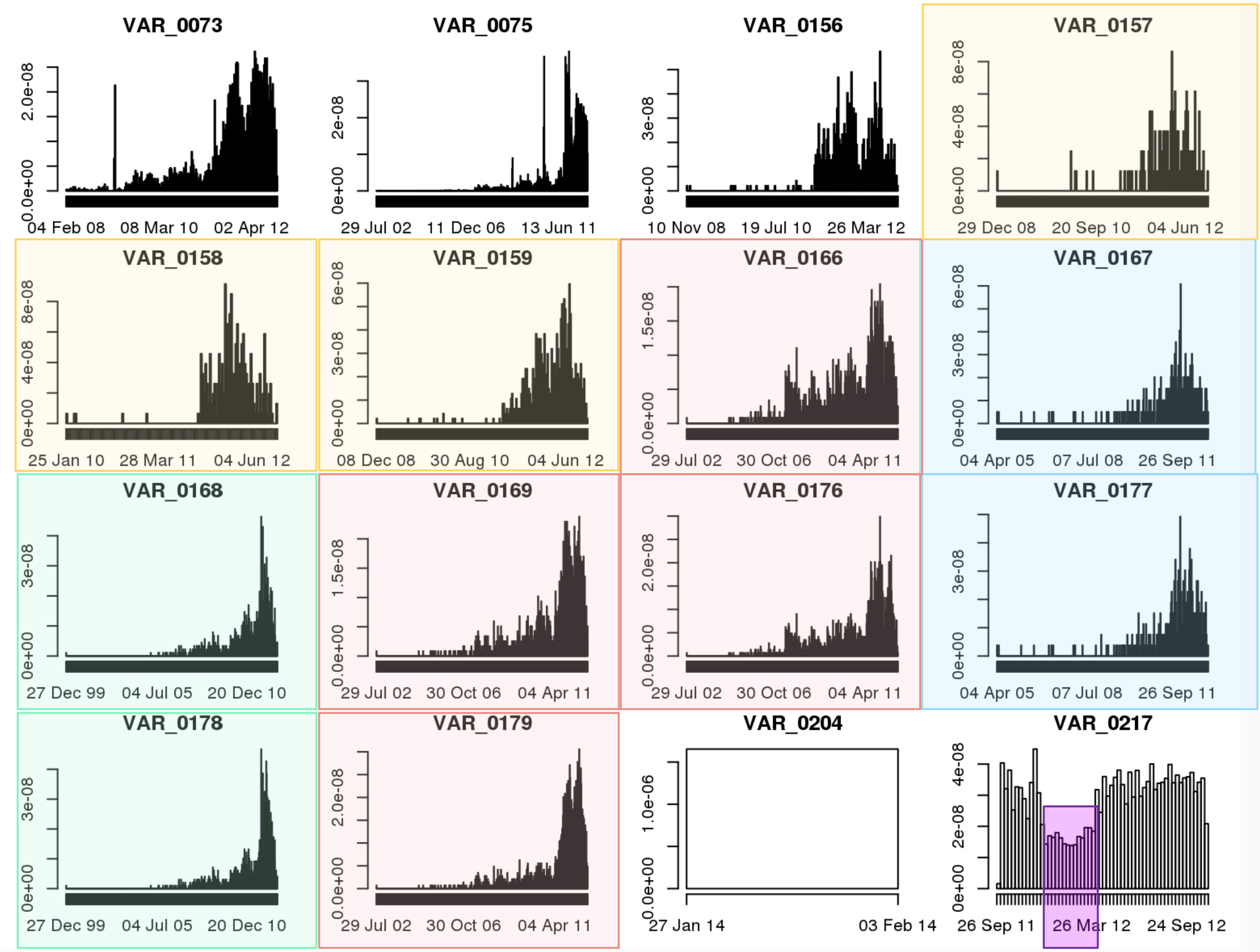
Reference: https://www.kaggle.com/c/dato-native/forums/t/16626/beat-the-benchmark-0-90388-with-simple-model

Feature Extraction: Hidden Features

- Sometimes there has some information leakage in the dataset provided by Kaggle competition
  - Timestamp information of data files
  - Some incautiously left meta-data inside HTML or text
- May lead to unfair results, so normally I skip this kind of competitions!

Exploratory Data Analysis is Your Friend

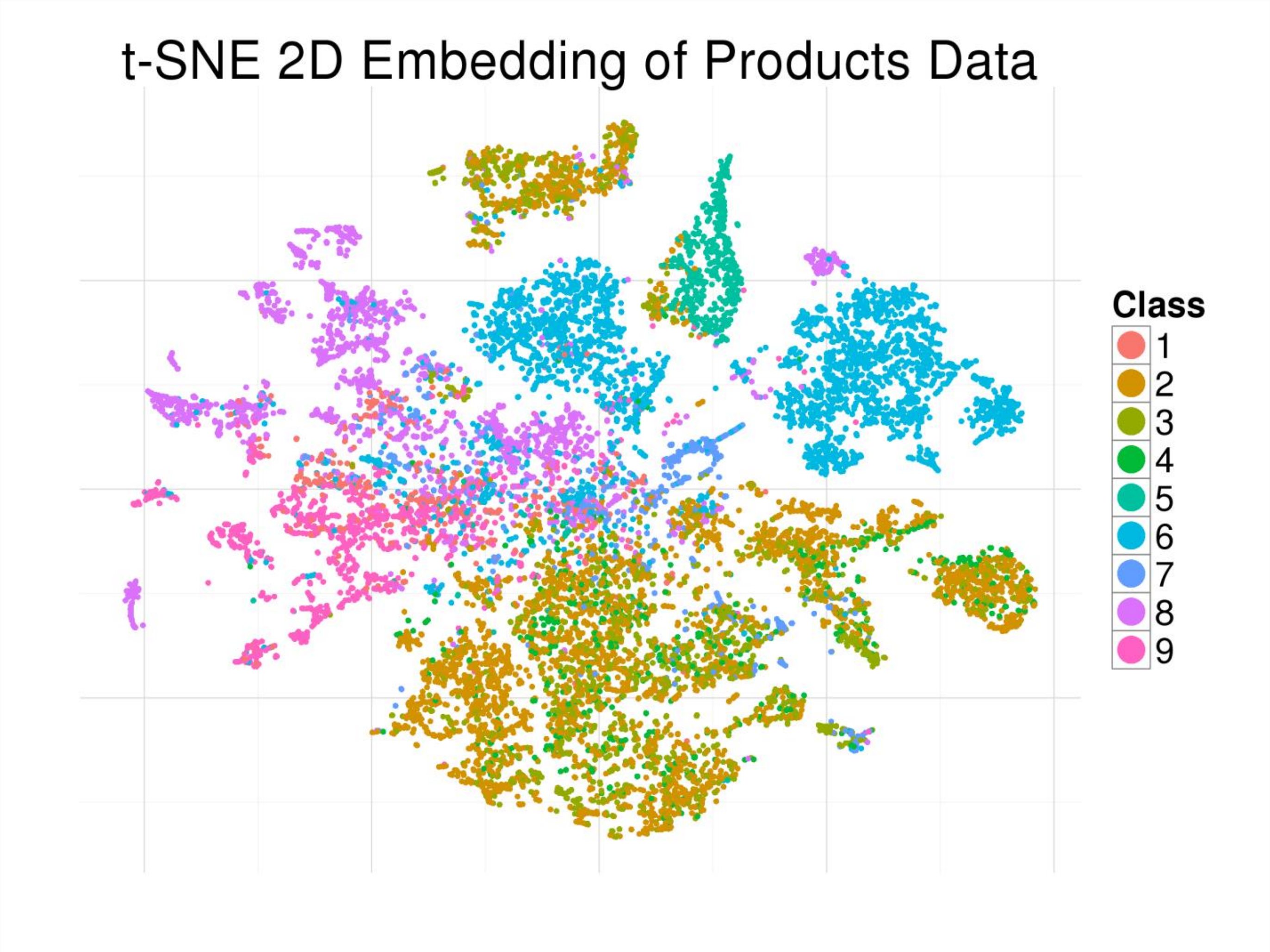
Finding patterns and correlations from time series data



Reference: https://www.kaggle.com/darraghdog/springleaf-marketing-response/explore-springleaf/notebook

Exploratory Data Analysis is Your Friend

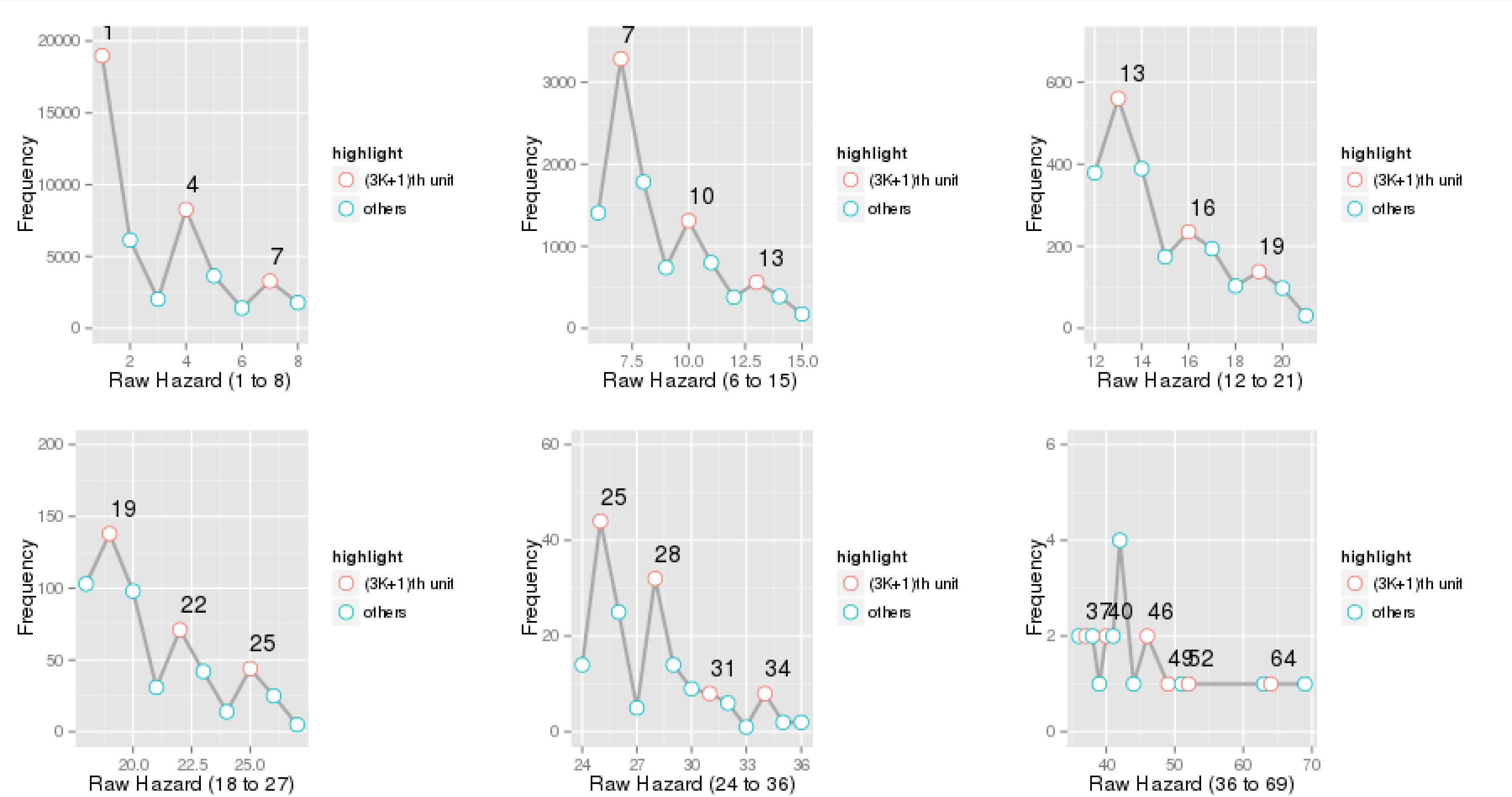
Using 2D visualization to observe the distribution of data in problem space



Reference: https://www.kaggle.com/benhamner/otto-group-product-classification-challenge/t-sne-visualization

Exploratory Data Analysis is Your Friend

Finding hidden groups inside the data



Reference: https://www.kaggle.com/markpeng/liberty-mutual-group-property-inspection-prediction/hazard-groups-in-training-set/files

Feature Selection

- Reduce the number of features by removing redundant, irrelevant or noisy features
- Feature Explosion after Feature Extraction!
  - More than 100,000 unique token features from textual data is common
  - Hard to put all of them in memory!
  - PCA or truncated SVD can help to select top-*N* informative features
    - With the risk of ignoring non-linear relationships between features and dropping important features
    - Use them only if there is no other choice

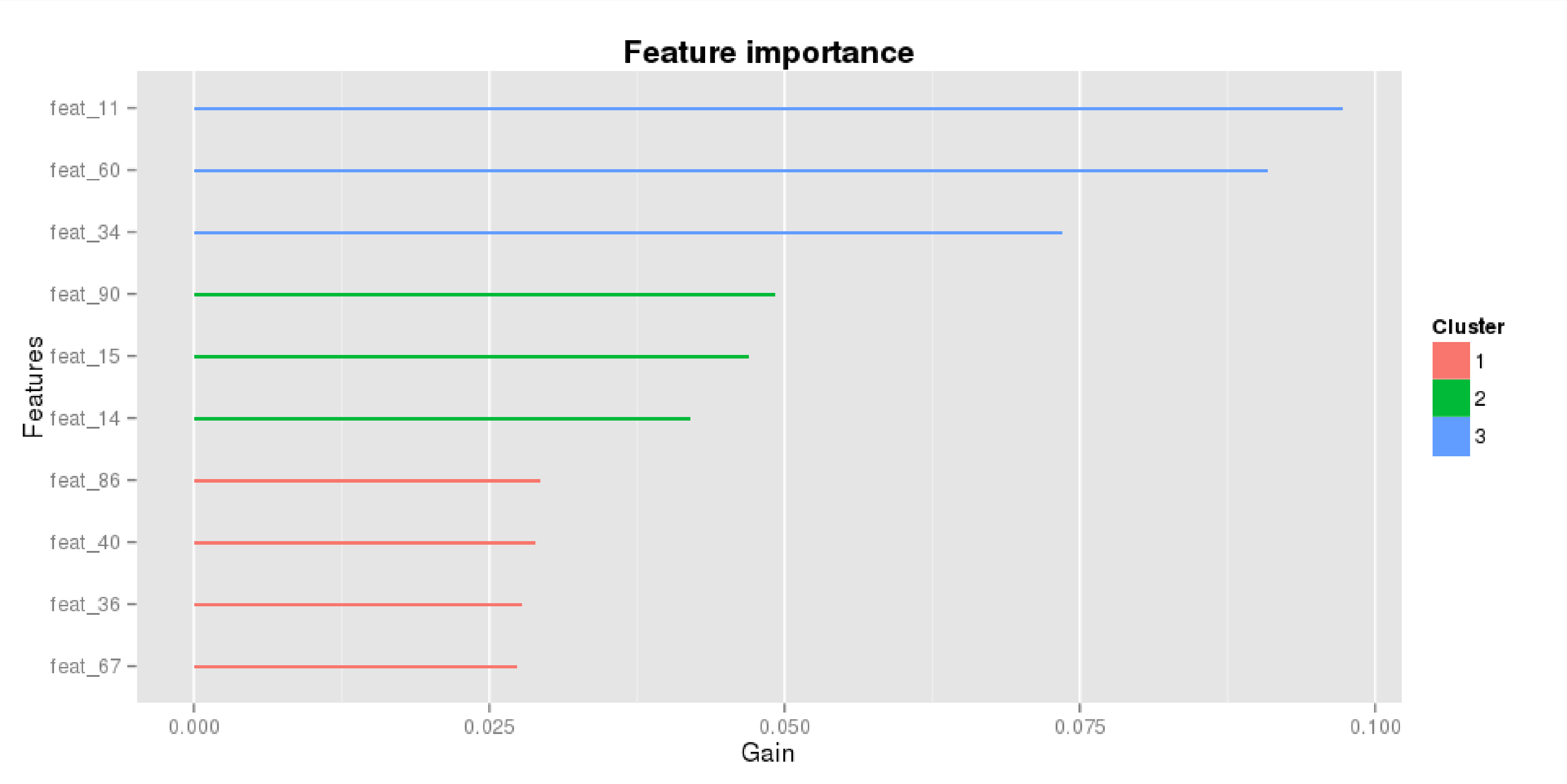
Feature Selection Methods

Type	Name	R	Python
Feature Importance Ranking	Gini Impurity	• randomForest • varSelRF	• sklearn.ensemble.RandomForestClassifier • sklearn.ensemble.RandomForestRegressor • sklearn.ensemble.GradientBoostingClassifier • sklearn.ensemble.GradientBoostingRegressor
	Chi-square	• Fselector	• sklearn.feature_selection.chi2
	Correlation	• Hmisc • Fselector	• scipy.stats.pearsonr • scipy.stats.spearmanr
	Information Gain	• randomForest • varSelRF • Fselector	• sklearn.ensemble.RandomForestClassifier • sklearn.ensemble.RandomForestRegressor • sklearn.ensemble.GradientBoostingClassifier • sklearn.ensemble.GradientBoostingRegressor • xgboost
	L1-based Non-zero Coefficients	• glmnet	• sklearn.linear_model.Lasso • sklearn.linear_model.LogisticRegression • sklearn.svm.LinearSVC
Feature Subset Selection	Recursive Feature Elimination (RFE)	• rfe {caret}	• sklearn.feature_selection.RFE
	Boruta Feature Selection	• Boruta	
	Greedy Search (forward/backward)	• Fselector	
	Hill Climbing Search	• Fselector	
	Genetic Algorithms	• gafs {caret}	

Feature Selection Methods

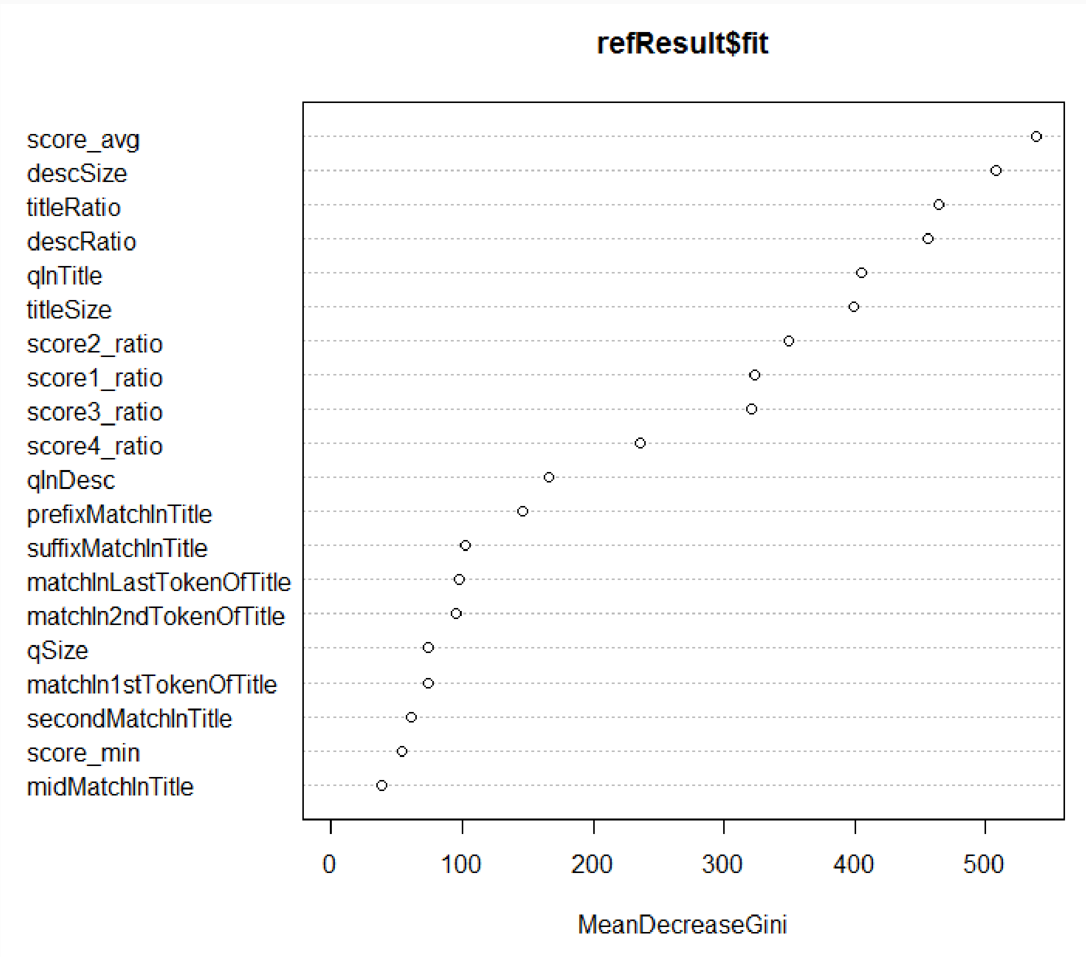
Type	Name	Spark
Feature Importance Ranking	Gini Impurity	• RandomForest (see SPARK-5133)
	Chi-square	• ChiSqSelector
	Correlation	
	Information Gain	• spark-mrmr-feature-selection
	L1-based Non-zero Coefficients	
Feature Subset Selection	Recursive Feature Elimination (RFE)	
	Boruta Feature Selection	
	Greedy Search (forward/backward)	
	Hill Climbing Search	
	Genetic Algorithms	

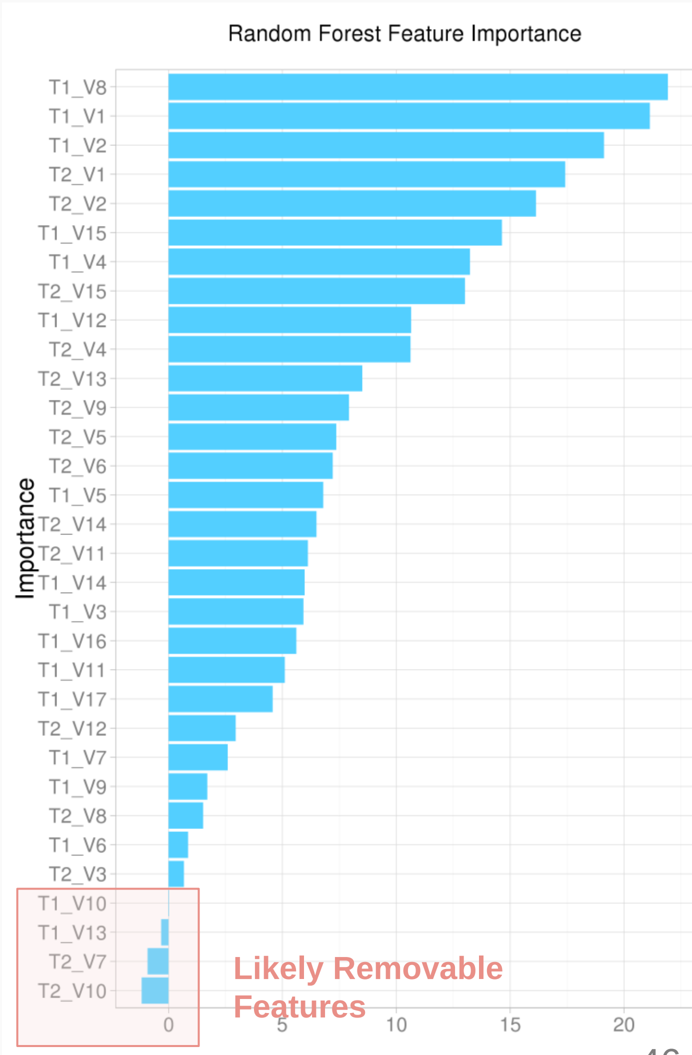
Feature Importance (xgboost)



Reference: https://www.kaggle.com/tqchen/otto-group-product-classification-challenge/understanding-xgboost-model-on-otto-data/notebook

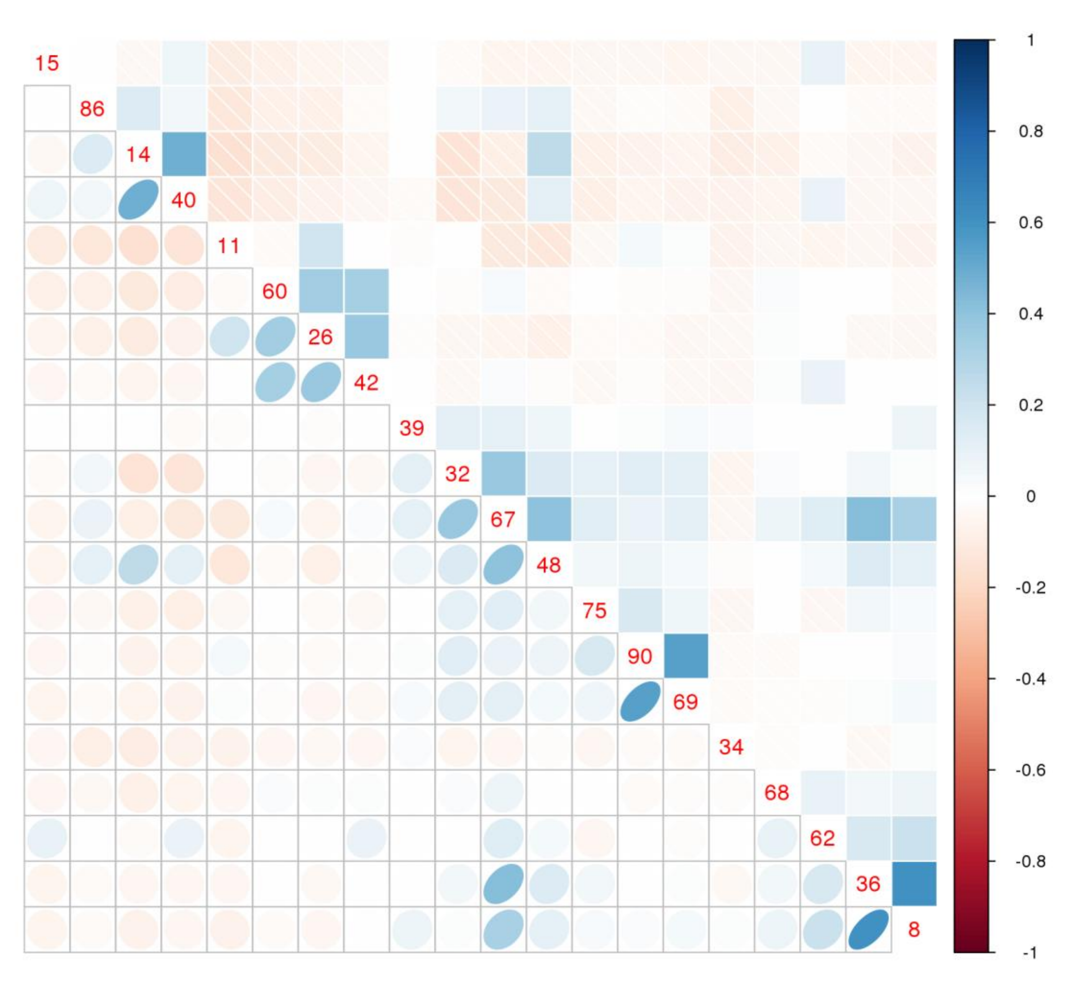
Feature Importance (randomForest in R)





Feature Correlations

Correlations between top-20 features



Hypothesis: "*Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other.*“ - Wikipedia

Reference: https://www.kaggle.com/benhamner/otto-group-product-classification-challenge/important-feature-correlations

Recursive Feature Elimination (RFE)

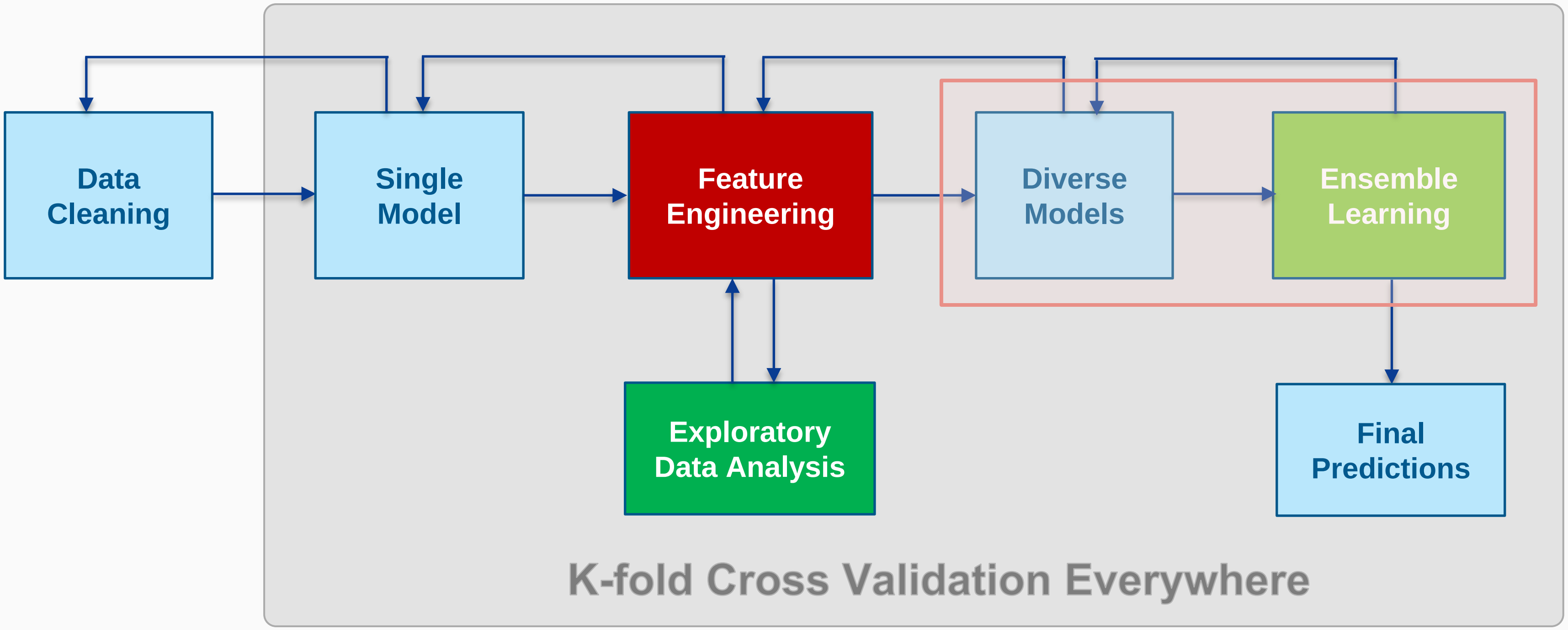
```

Recursive feature selection

Outer resampling method: Cross-Validated (3 fold, repeated 3 times)

Resampling performance over subset size:

 Variables wKappa wKappaSD Selected
        50 0.6715 0.008026
       100 0.6732 0.008263
       150 0.6773 0.007482        *
       200 0.6753 0.008301
       250 0.6762 0.007253
       300 0.6749 0.006091
       350 0.6741 0.006806
       400 0.6743 0.006093
       450 0.6756 0.004383
       500 0.6716 0.006937
       550 0.6743 0.008261
       600 0.6746 0.008223
       625 0.6731 0.007996

The top 5 variables (out of 150):
   C_377, C_417, C_381, C_379, C_457
```

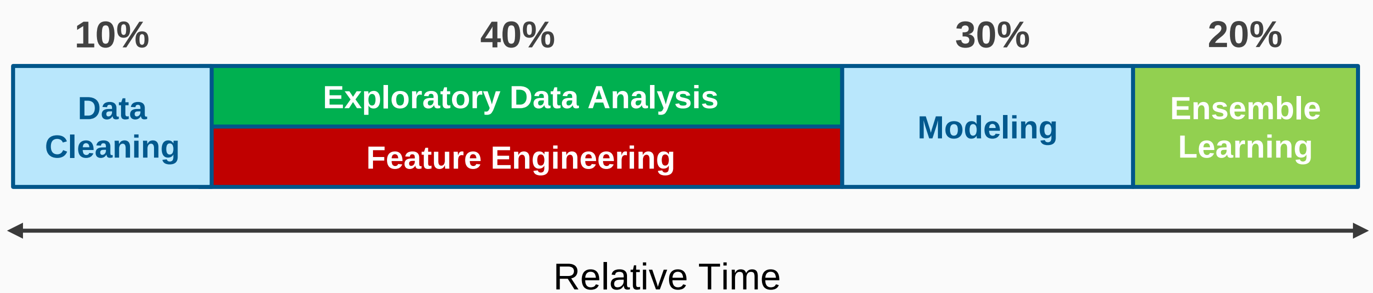
Find a subset of features with best performance

Ensemble Learning

Beat single strong models

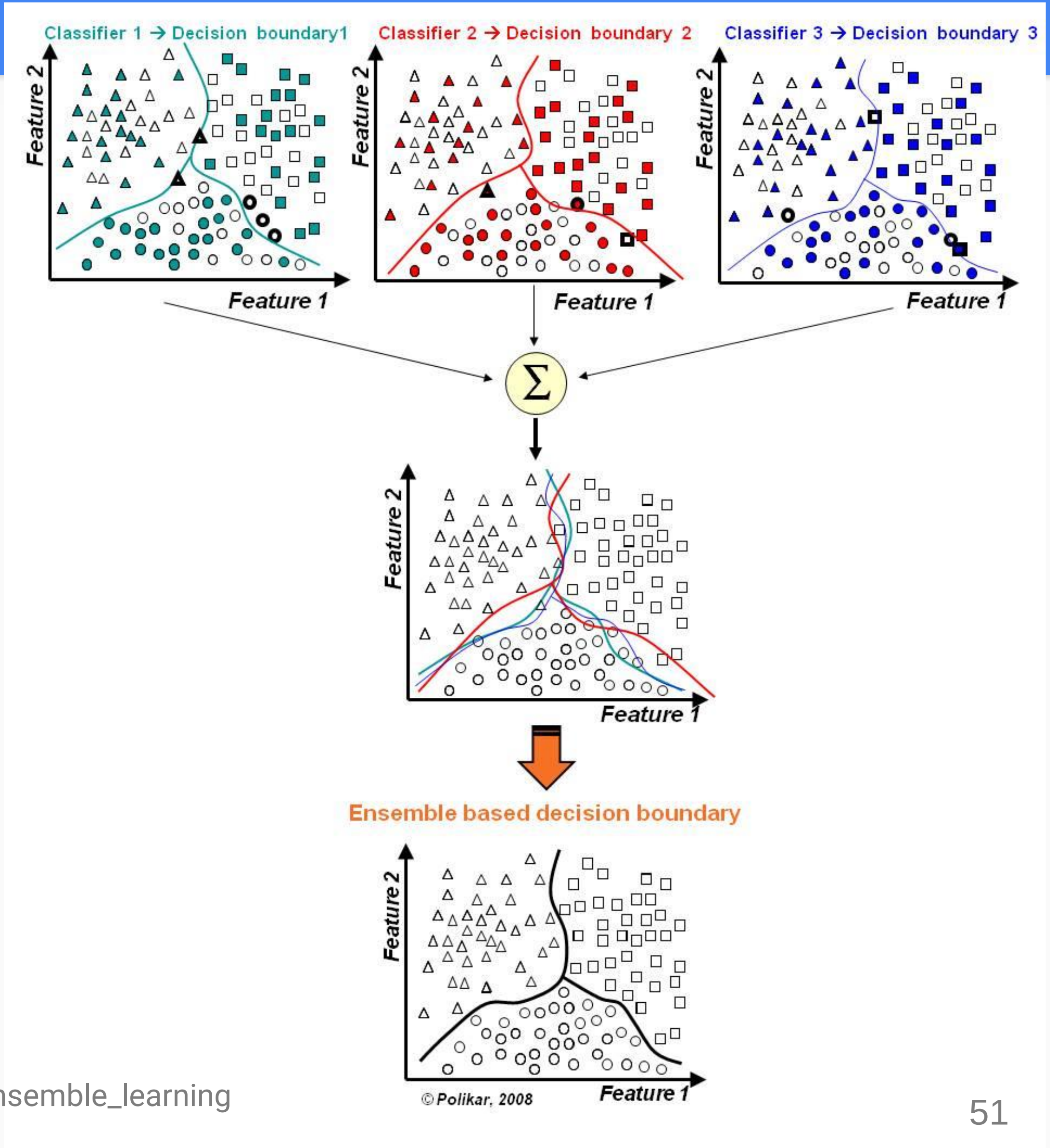
Recommended Data Science Process (IMHO)





Ensemble Learning - Illustration

- Assume that diverse models see different aspects of the problem space and are wrong in different ways
- Simplest way: the average (possibly weighted) of model predictions
- Less variance
- Less generalization error
- Less chance of overfitting
- Better chance to win!



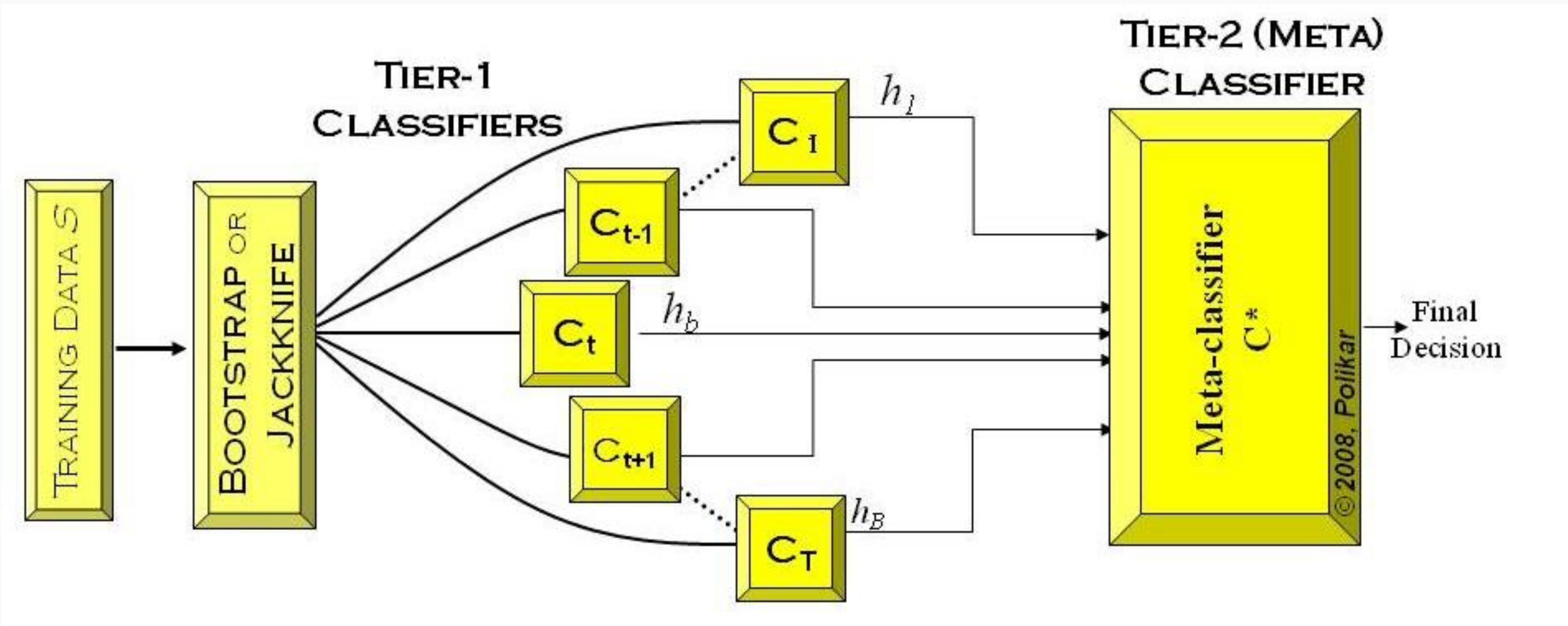
Reference: http://www.scholarpedia.org/article/Ensemble_learning

Ensemble Learning - Stacked Generalization

- Using the predictions of current level models as features for training next level models
  - What we called “*out-of-fold predictions*” or “*meta-features*”
- Usually we only build 2 levels of models, 4 levels at most
- Can combine meta-features with original feature sets for training
- Potential risk of overfitting if cross validation process is not correct

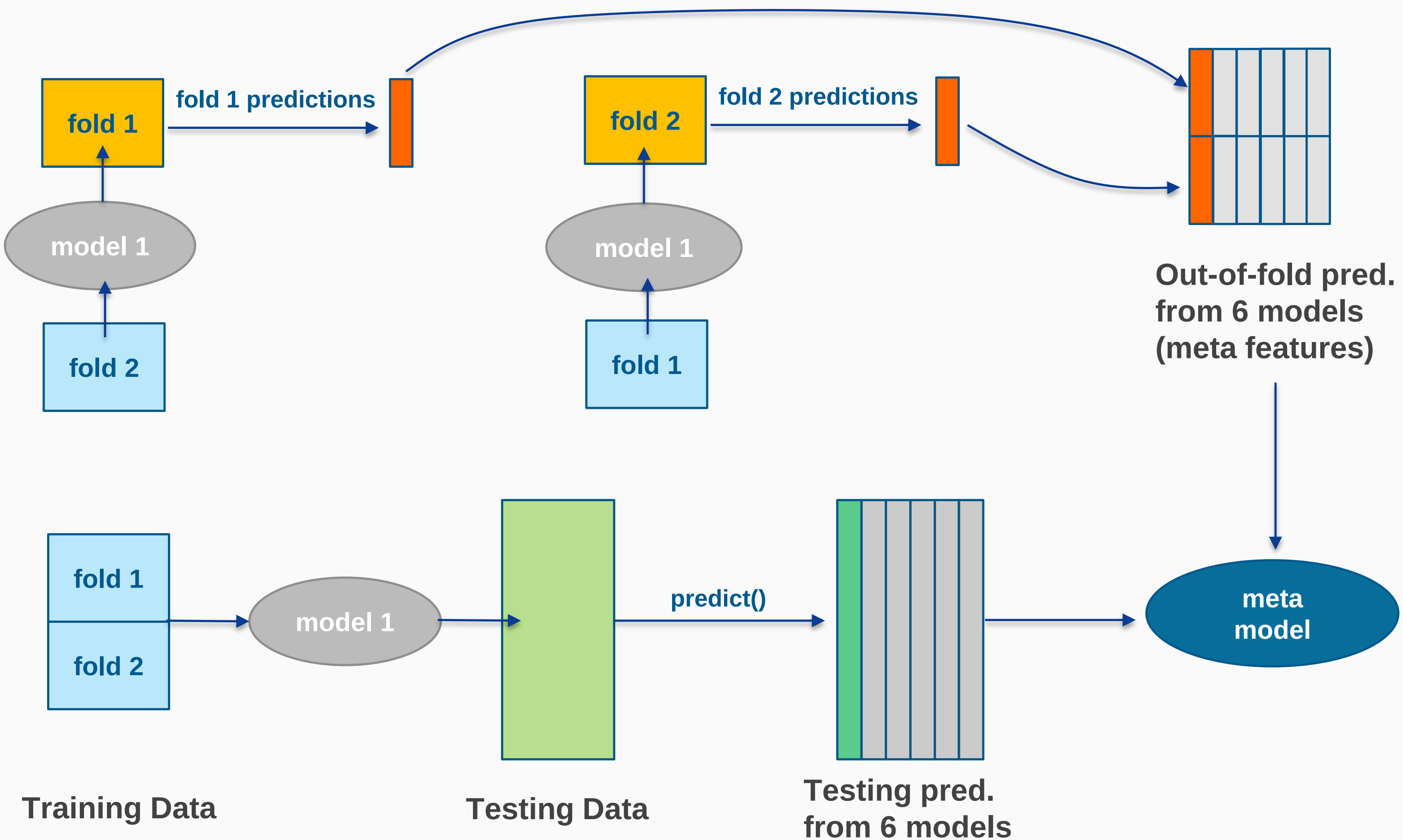
- Also called “Stacking”
- Aims to reduce *Generalization Error*

Ensemble Learning - Stacked Generalization



Reference: http://www.scholarpedia.org/article/Ensemble_learning

Out-of-fold Prediction (K = 2)



Out-of-fold Prediction - Generation Process

- For example, 2-fold CV with 2 levels stacking
  - Split the train set into 2 parts: *train_fold1* and *train_fold2*
  - Fit a first-level model on *train_fold1* and create predictions for *train_fold2*
  - Fit the same model on *train_fold2* and create predictions for *train_fold1*
  - Finally, fit the model again but on the entire training data, and create predictions for testing data
  - Using the predictions from the first-level model(s) as meta-features, train a second-level model

Reference: http://mlwave.com/kaggle-ensembling-guide/

Out-of-fold Prediction (cont.)

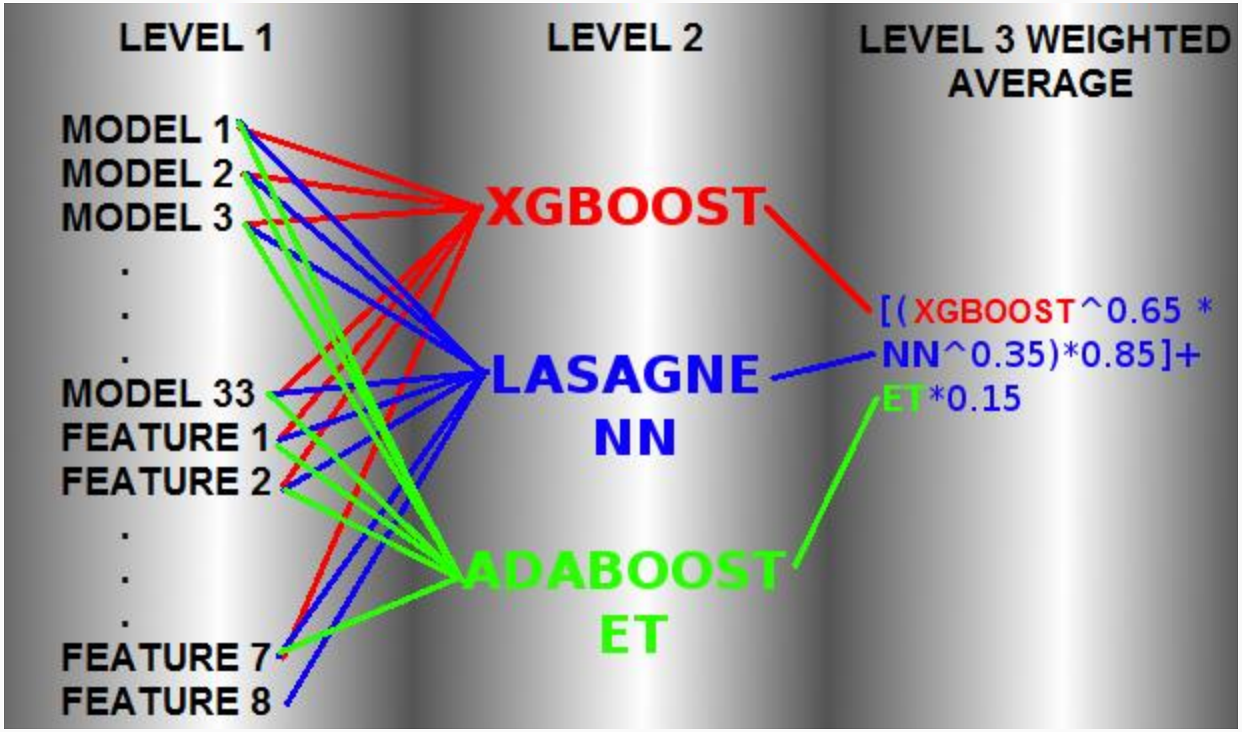
“I learned that you never, ever, EVER go anywhere without your out-of-fold predictions. If I go to Hawaii or to the bathroom I am bringing them with. Never know when I need to train a 2nd or 3rd level meta-classifier”

- T. Sharf

- Basic requirement for stacked generalization
- Can also combine with original features for modeling
- Keep it whenever building any models!

Reference: http://mlwave.com/kaggle-ensembling-guide/

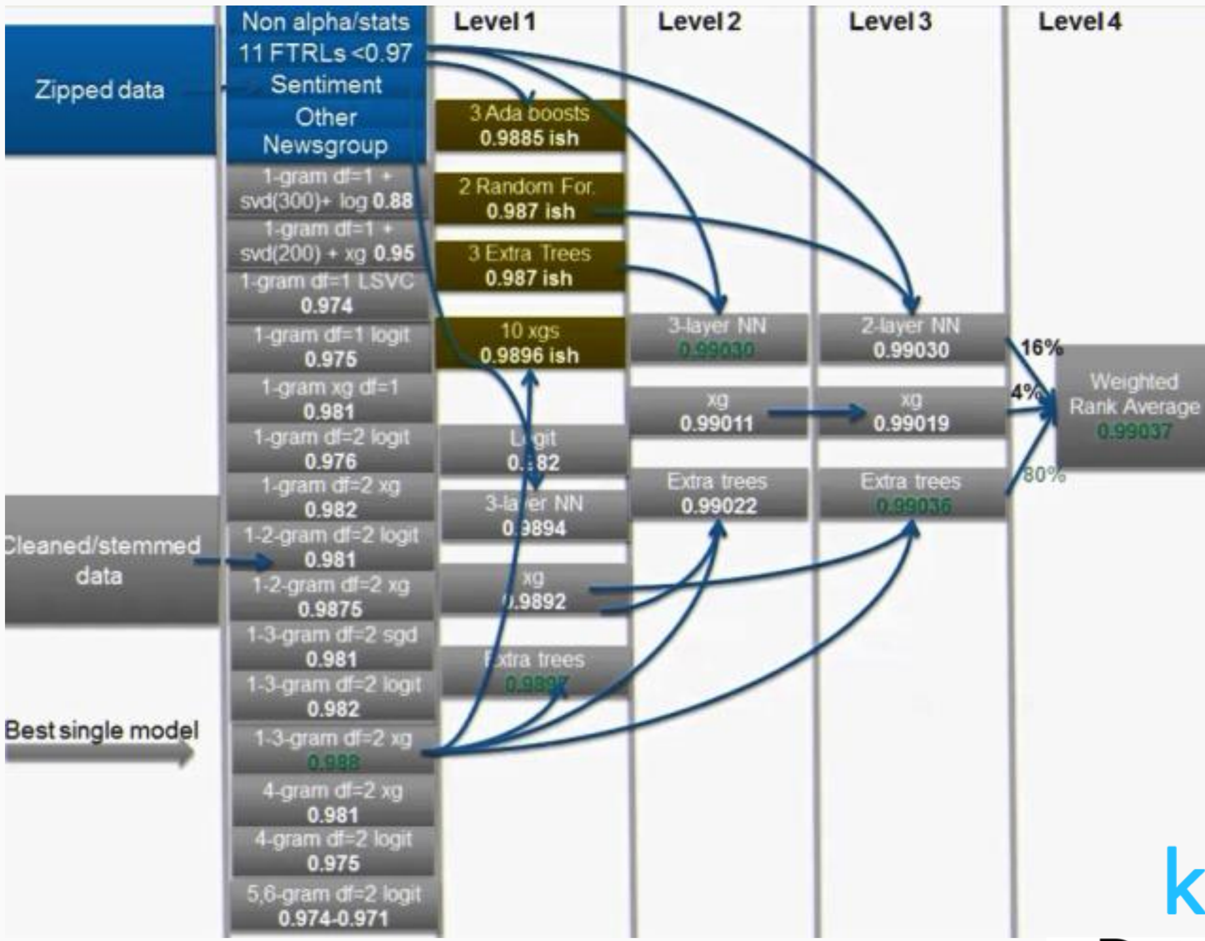
Stacked Generalization Used by *Gilberto Titericz Junior* (new #1) in *Otto Product Classification Challenge*



Reference: https://www.kaggle.com/c/otto-group-product-classification-challenge/forums/t/14335/1st-place-winner-solution-gilberto-titericz-stanislav-semenov

kaggle
otto group

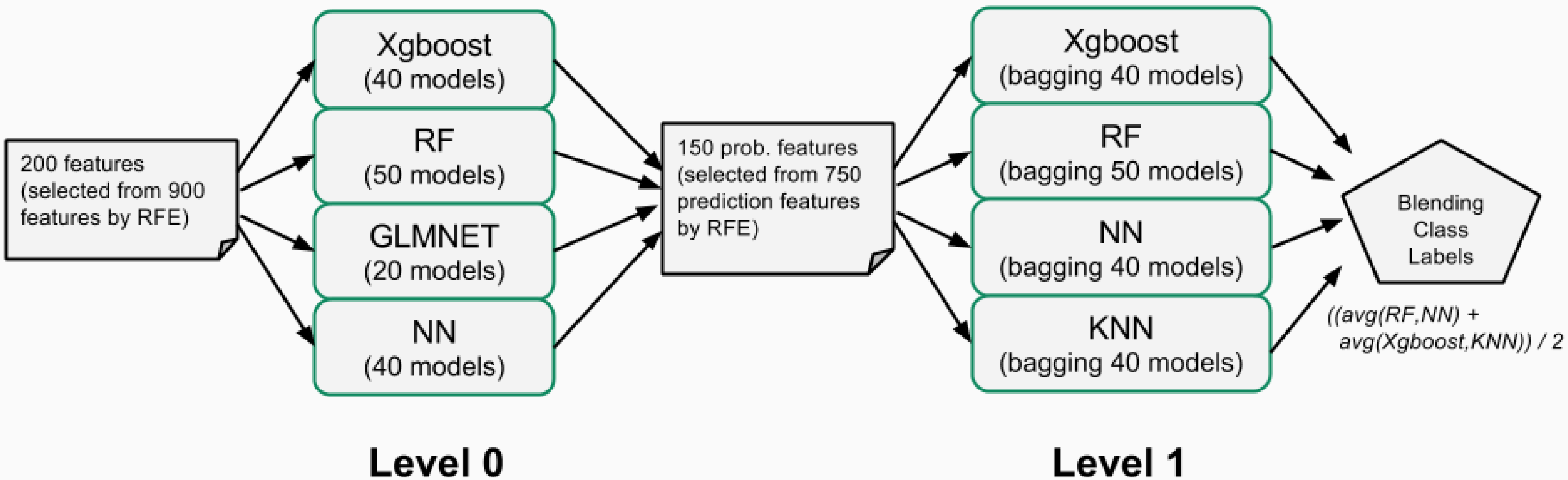
Stacked Generalization Used by the Winning Team in *Truly Native?* Competition (4 levels)



Reference: http://blog.kaggle.com/2015/12/03/dato-winners-interview-1st-place-mad-professors/

kaggle Dato

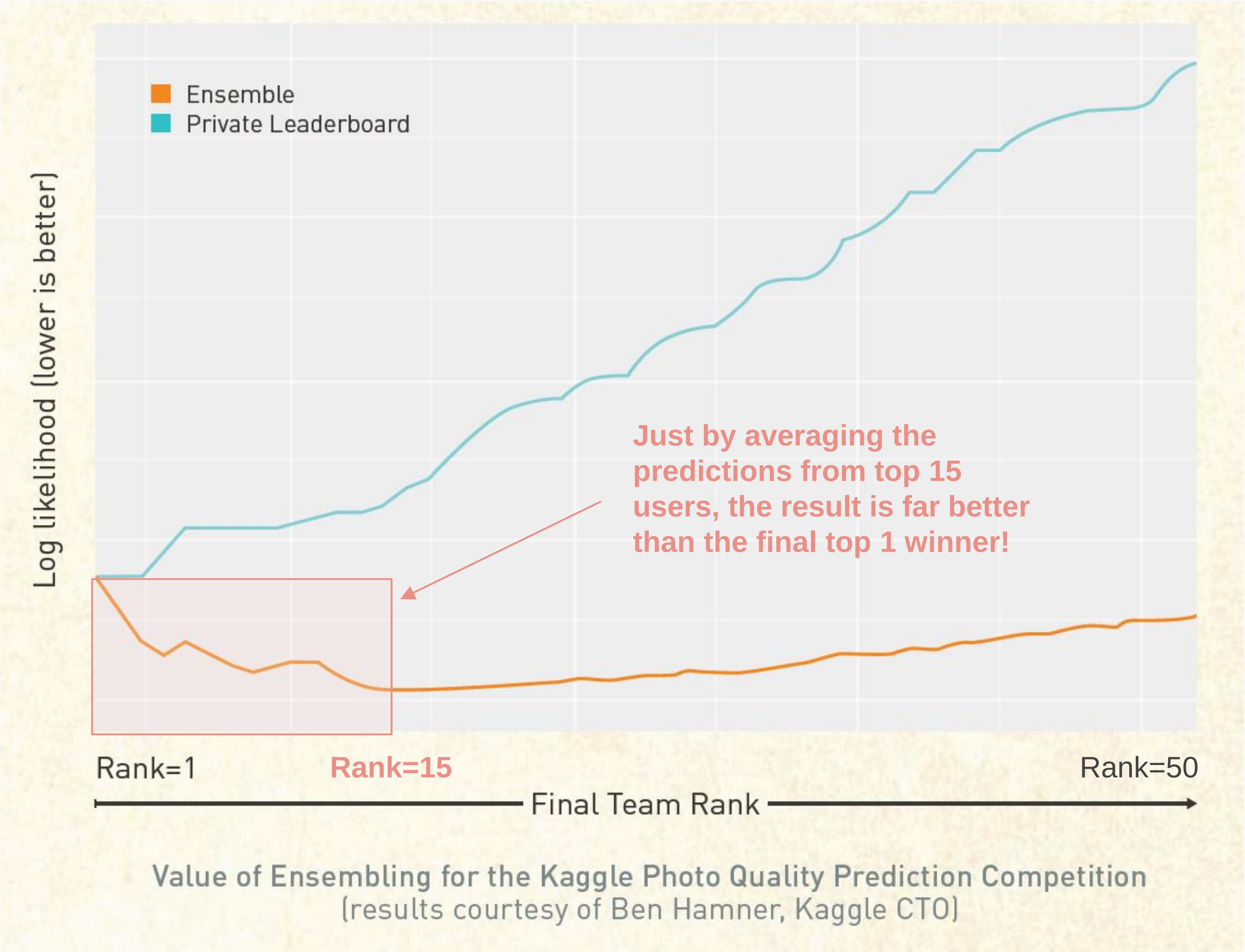
Stacked Generalization I Used in Search Results Relevance Competition



More technical details can be found in this post:
https://www.linkedin.com/pulse/ideas-sharing-kaggle-crowdflower-search-results-relevance-mark-peng

kaggle CrowdFlower

The Power of Ensemble Learning



Reference: Booz Allen Hamilton. (2015). "*Field Guide to Data Science,*" page 93.

Ensemble Learning - Tips

- Always start from simple blending first
- Even simple bagging of multiple models of the same type can help reduce variance
- Generally work well with less-correlated good models
    - Models with similar performance but their correlation lies in between 0.85~0.95
    - Gradient Boosting Machines usually blend well with Random Forests

- Must Read: [Kaggle Ensembling Guide]

Team Up

Working with clever data geeks worldwide

The Advantages of Team Up

- Fewer work loads for each one if divides up the work well
  - A focuses on feature engineering
  - B focuses on ensemble learning
  - C focuses on blending submissions
  - ……
- Each member can contribute some single models for blending and stacking

Ways to Team Up

- **Team up with some known friends**
  - Pros
    - Easy to discuss the ideas
    - Easy to divide up the work
    - Easy to share codes
  - Cons
    - Harder to think outside the box because each one’s ideas and thoughts might be tied together

Ways to Team Up (cont.)

- **Team up with other competitors worldwide (recommended)**
  - Pros
    - You will learn a lot from the others from different counties and diverse background
    - Feature sets and models are less-correlated, thus more likely to produce better predictions after blending or stacking
    - Easier to finish with the Top 10
  - Cons
    - Harder to discuss the ideas (through emails or instant messages)
    - Harder to divide up the work
    - More sleepless!
  - Cloud storage can help share feature sets and codes

When to Start Team Merger?

- Usually, each competition only allows 3 or 5 submissions per day per competitor
- After team merger, you have to share the quota with others in the team
- *Combined team must have a total submission count less than or equal to the maximum allowed as of the merge date*
- **A better strategy (IMHO)**
  - Start team merger 1-2 weeks before merger deadline
  - Give everyone enough daily submission to twist his/her mind and try out more ideas until exhausted

How to Produce Better Predictions

- Feature Set Sharing
  - Rebuild everyone's models using others' feature set
  - More good single models for blending
- Blending Submissions
  - The easiest way to boost your team's ranking in LB
  - Weighted average often gives better LB score
  - Tune model weightings by handcrafting
- Multi-level Stacking based on Out-of-fold Predictions
  - Have to ask everyone build models using the same *K* folds and provide good out-of-fold predictions
  - Require more time and resources to train models in parallel
  - Produce the most powerful predictions!

Recommended Resources

Materials to help you learn by doing

Books

- James G., Witten D., Hastie T., Tibshirani R. (2014). “*An Introduction to Statistical Learning: with Applications in R*,” http://www-bcf.usc.edu/~gareth/ISL/
  - Many R examples to learn, good for the beginners
- Hastie T., Tibshirani R., Friedman J. (2009). “*The Elements of Statistical Learning: Data Mining, Inference, and Prediction*,” http://statweb.stanford.edu/~tibs/ElemStatLearn/
  - Fundamental theories to statistics and machine learning
- Leskovec J., Rajaraman A., Ullman J. (2014). “*Mining of Massive Datasets*,” http://www.mmds.org
  - How to scale data mining and machine learning to large dataset
- Booz Allen Hamilton. (2015). “*Field Guide to Data Science*,” http://www.boozallen.com/insights/2015/12/data-science-field-guide-second-edition
  - A good overview about Data Science (Describe -> Discover -> Predict -> Advise)

Massive Online Open Courses (MOOCs)

- Machine Learning, Stanford Professor Andrew Ng, Coursera Inc.
- Mining Massive Datasets, Stanford University, Coursera Inc.
- Statistical Learning, Stanford University, Stanford Online
- The Analytics Edge, MIT, edX Inc.
    - Very practical, final exam is to compete with other students on Kaggle!
- Introduction to Big Data with Apache Spark, Berkeley, edX Inc.
- Scalable Machine Learning, Berkeley, edX Inc.

Other Resources

- Kaggle Competition Forums
  - https://www.kaggle.com/forums
  - Find similar competitions in the past
  - Pick out some code snippets and scripts shared by the others
  - Reading winners' codes is one of the best ways to learn
- "*Winning Data Science Competitions*" by Owen Zhang
- "*A Few Useful Things to Know about Machine Learning*" by Pedro Domingos
- "*Advice for applying Machine Learning*" by Andrew Ng

Believe in Yourself

You can make it to the 10 top along as well!

Then Team Up!

Prove yourself to team up with clever guys from all over the world

kaggle | Host | Competitions | Scripts | Jobs | Community ▾ | Mark Peng | Logout

CrowdFlower

$20,000 • 1,029 teams

Search Results Relevance

Merger and 1st Submission Deadline

Mon 11 May 2015 — Mon 6 Jul 2015 (14 days to go)

Dashboard ▾

Public Leaderboard - Search Results Relevance

This leaderboard is calculated on approximately 40% of the test data.
The final results will be based on the other 60%, so the final standings may be different.

See someone using multiple accounts?
Let us know.

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Last Submission)
1	↑1	Quartet *	0.70456	207	Mon, 22 Jun 2015 00:53:02
2	↓1	Chenglong Chen *	0.70448	106	Wed, 17 Jun 2015 16:21:53
3	↑5	Shize & Shail & Phil *	0.69944	177	Sun, 21 Jun 2015 23:56:37 (-9.7h)
4	↑3	Gzs_iceberg	0.69868	78	Sun, 21 Jun 2015 03:36:31 (-2.8d)
5	↓2	Alexander D'yakonov (PZAD, Russia)	0.69477	79	Sun, 21 Jun 2015 19:15:23 (-18d)
6	↓1	Jianmin Sun	0.69461	100	Sun, 21 Jun 2015 22:21:29
7	↑2	Off to Glastonbury to overfit beer	0.69359	100	Fri, 19 Jun 2015 15:33:47
8	↓2	JimingAndYuqi	0.69119	198	Sun, 21 Jun 2015 05:49:41
9	↓5	KazAnova meets H2O.ai	0.69048	87	Sun, 21 Jun 2015 09:00:55 (-33.3h)
10	↑44	**Mark Peng**	0.68552	62	Mon, 22 Jun 2015 01:57:18
11	↑1	CodingNeo	0.68379	53	Sun, 21 Jun 2015 09:00:44 (-3.6d)
12	↓2	A & A	0.68318	146	Sun, 21 Jun 2015 22:05:22 (-10.5d)
13	↓2	MarcusGroß	0.68263	87	Fri, 12 Jun 2015 20:56:14 (-6.4d)
14	—	sorpmaL	0.68145	51	Sun, 21 Jun 2015 20:55:10 (-5.1d)
15	↓2	carl & SRK	0.68104	67	Sun, 21 Jun 2015 06:48:46 (-3.6d)

Your Best Entry ↑

Top Ten!

You made the top ten by improving your score by 0.01512.

You just moved up 17 positions on the leaderboard. Tweet this!

Concluding Words

- “*Think more, try less*” - by Owen Zhang
- Find a competition with the dataset *you loved*
- Build your own reusable toolkit and framework for CV and modeling
- Use fixed seeds to make sure your results is reproducible
- Learn from Top Winners
- Be disciplined, focus on one competition at a time
- *Data science is an iterative process*
- Always trust more in your local CV!

Thanks!

Contact me:

Mark Peng
markpeng.ntu@gmail.com
LinkedIn Profile