

The Kaggle Competition



Data Application Lab

Henry

Kaggle Joins Google Cloud

Anthony Goldbloom | 03.08.2017



Intro of Kaggle



Welcome to Kaggle Competitions

Challenge yourself with real-world machine learning problems



New to Data Science?

Get started with a tutorial on our most popular competition for beginners, [Titanic: Machine Learning from Disaster](#).



Build a Model

Get the data & use whatever tools or methods you prefer to make predictions.

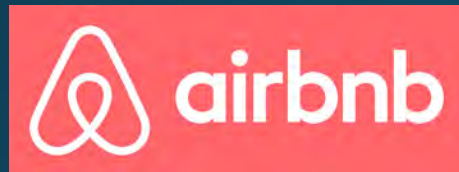


Make a Submission

Upload your prediction file for real-time scoring & a spot on the leaderboard.

- The most influencing and active data science platform
- 500,000 data scientists from 200 countries
- Partnered with big names such as Google, Facebook, Microsoft, Amazon, Airbnb, NASA, and
- Cradle of pioneered algorithms and approaches that are now being widely used by industries

The Coolest Dataset



Featured Prediction Competition

Intel & MobileODT Cervical Cancer Screening

Which cancer treatment will be most effective?

\$100,000
Prize Money

Featured Prediction Competition

Google Cloud & YouTube-8M Video Understanding Challenge

Can you produce the best video tag predictions?

\$100,000
Prize Money

Featured Prediction Competition


Planet: Understanding the Amazon from Space

Use satellite data to track the human footprint in the Amazon rainforest

\$60,000
Prize Money







IMDB 5000 Movie Dataset

5000+ movie data scraped from IMDB website



Prize and Jobs I.



	Data Science Bowl 2017 Can you improve lung cancer detection? <i>Featured</i> · 10 days ago	\$1,000,000 1,972 teams
	Heritage Health Prize Identify patients who will be admitted to a hospital within the next year using historical claims data. (Enter by 06:59:59 UTC Oct 4 2012) <i>Featured</i> · 4 years ago	\$500,000 1,353 teams
	Flight Quest 2: Flight Optimization, Milestone Phase Optimize flight routes based on current weather and traffic. <i>Ge</i> · 4 years ago	\$250,000 129 teams
	GE Flight Quest Think you can change the future of flight? <i>Ge</i> · 4 years ago	\$250,000 173 teams
	Flight Quest 2: Flight Optimization, Final Phase Final Phase of Flight Quest 2 <i>Ge</i> · 3 years ago	\$220,000 33 teams
	Flight Quest 2: Flight Optimization, Main Phase Optimize flight routes based on current weather and traffic. <i>Ge</i> · 3 years ago	\$220,000 121 teams

- 1st place - \$500,000
- 2nd place - \$200,000
- 3rd place - \$100,000
- 4th place - \$25,000
- 5th place - \$25,000
- 6th place - \$25,000
- 7th place - \$25,000
- 8th place - \$25,000
- 9th place - \$25,000
- 10th place - \$25,000

Competition Sponsors

Laura and John Arnold Foundation
Cancer Imaging Program of the National Cancer Institute
American College of Radiology
Amazon Web Services
NVIDIA






Data Support Providers

National Lung Screening Trial
The Cancer Imaging Archive
Diagnostic Image Analysis Group, Radboud University
Lahey Hospital & Medical Center
Copenhagen University Hospital

Supporting Organizations

Bayes Impact
Black Data Processing Associates
Code the Change
Data Community DC
DataKind
Galvanize
Great Minds in STEM
Hortonworks
INFORMS
Lesbians Who Tech
NSBE
Society of Asian Scientists & Engineers
Society of Women Engineers
University of Texas Austin, Business Analytics Program,
McCombs School of Business
US Dept. of Health and Human Services
US Food and Drug Administration
Women in Technology
Women of Cyberjutsu

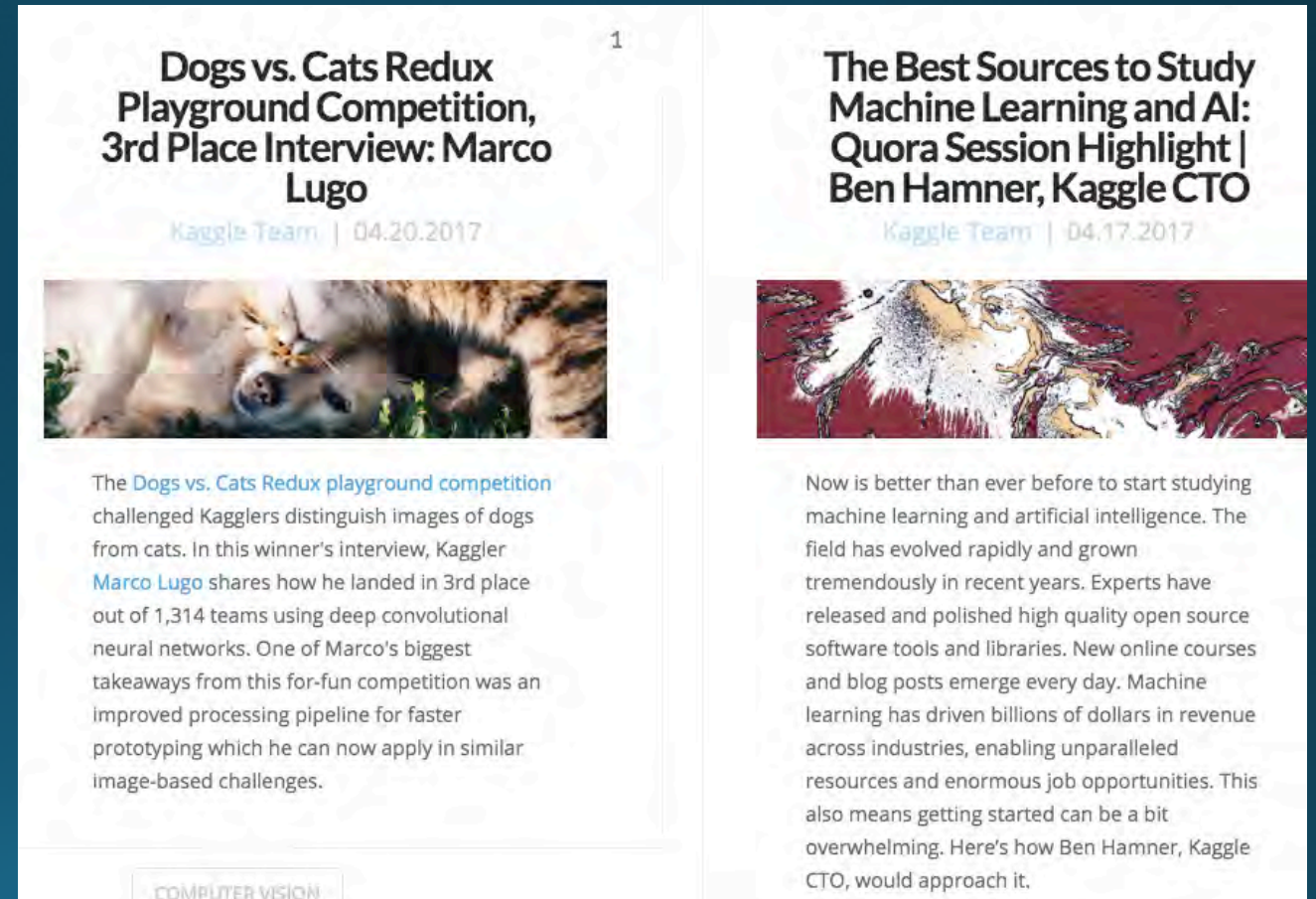
Prize and Jobs II.

	Two Sigma Connect: Rental Listing Inquiries How much interest will a new rental listing on RentHop receive? <i>Recruitment</i> · 3 days to go	Jobs 2,432 teams
	Allstate Claims Severity How severe is an insurance claim? <i>Recruitment</i> · 4 months ago · Entered	Jobs 3,055 teams
	Facebook V: Predicting Check Ins Identify the correct place for check ins <i>Recruitment</i> · 10 months ago	Jobs 1,212 teams
	Yelp Restaurant Photo Classification Predict attribute labels for restaurants using user-submitted photos <i>Recruitment</i> · A year ago	Jobs 355 teams
	Telstra Network Disruptions Predict service faults on Australia's largest telecommunications network <i>Recruitment</i> · A year ago	Jobs 974 teams

	★ Data Stream Processing Engineer Uniberg GmbH · Frankfurt a. M.; Germany <i>posted 2 days ago</i>	132 views
	★ Data Systems Engineer HomeAdvisor · Golden, CO <i>posted 3 days ago</i>	207 views
	★ Sr. Data Engineer Clear Capital · Reno, NV <i>posted 8 days ago</i>	449 views
	★ Predictive Analyst Regions Financial Corporation · Birmingham, AL <i>posted 9 days ago</i>	468 views
	★ Decision Scientist Regions Financial Corporation · Birmingham, AL <i>posted 9 days ago</i>	318 views

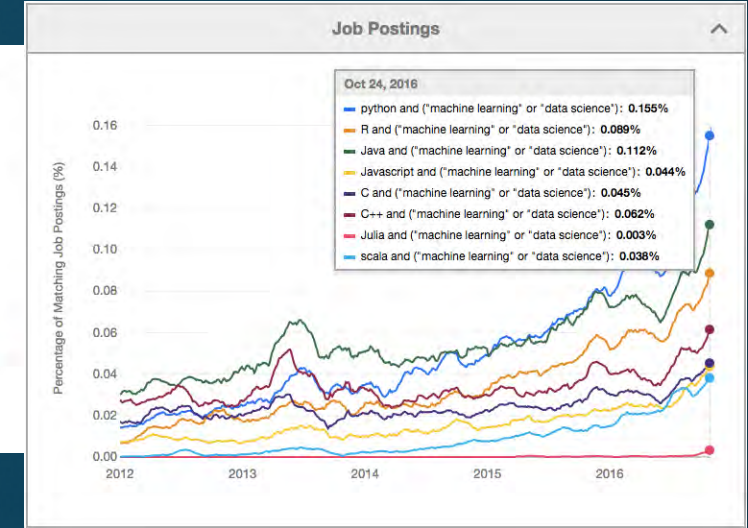
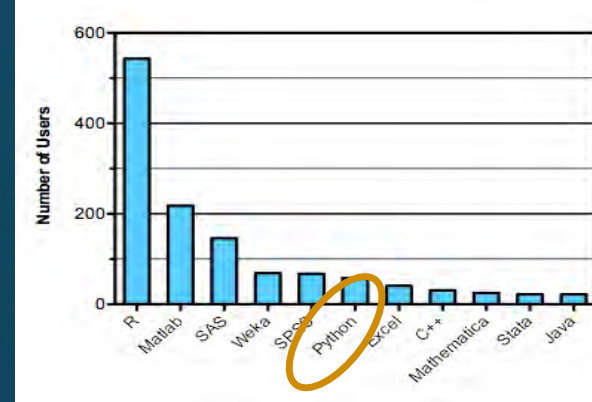
No Free Lunch

- Official Kaggle blog
- Categories:
 - Data science news
 - Kaggle news
 - Kernels
 - Tutorials
 - **Winners' interviews**



Tools

- R: <https://CRAN.R-project.org/view=MachineLearning>
- Python:
 - **scikit-learn: general purpose of machine learning**
 - **scipy, pandas, numpy, matplotlib**
 - **xgboost(dmlc) & lightgbm(microsoft)**
 - **Keras (tensorflow & theano), mxnet, torch, caffe**
 - **nlTK: NLP**
 - **rpy2: R interface**
 - **pyspark: big data**

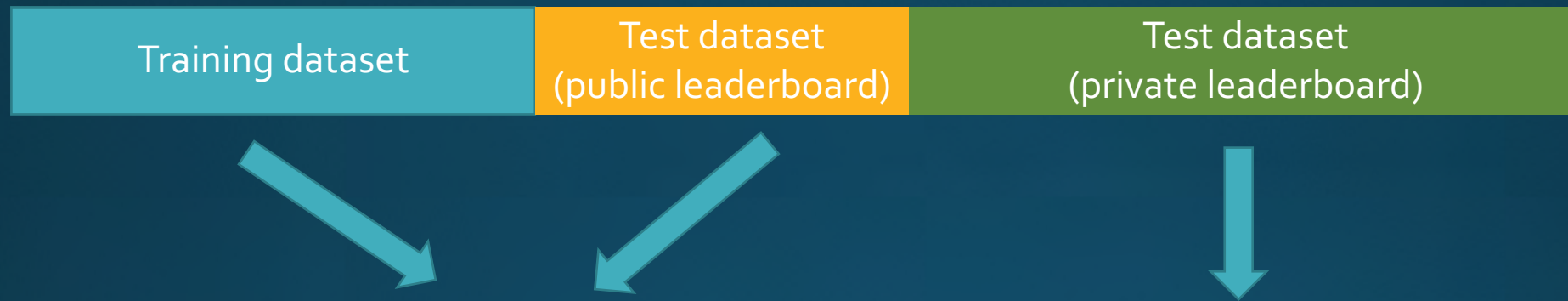


Let's Do It!

- Data preparation
- Data visualization
- Feature engineering
- Basic model
- Hyper-parameter tuning
- Model ensemble(stacked generalization)
- *Then, good luck...*



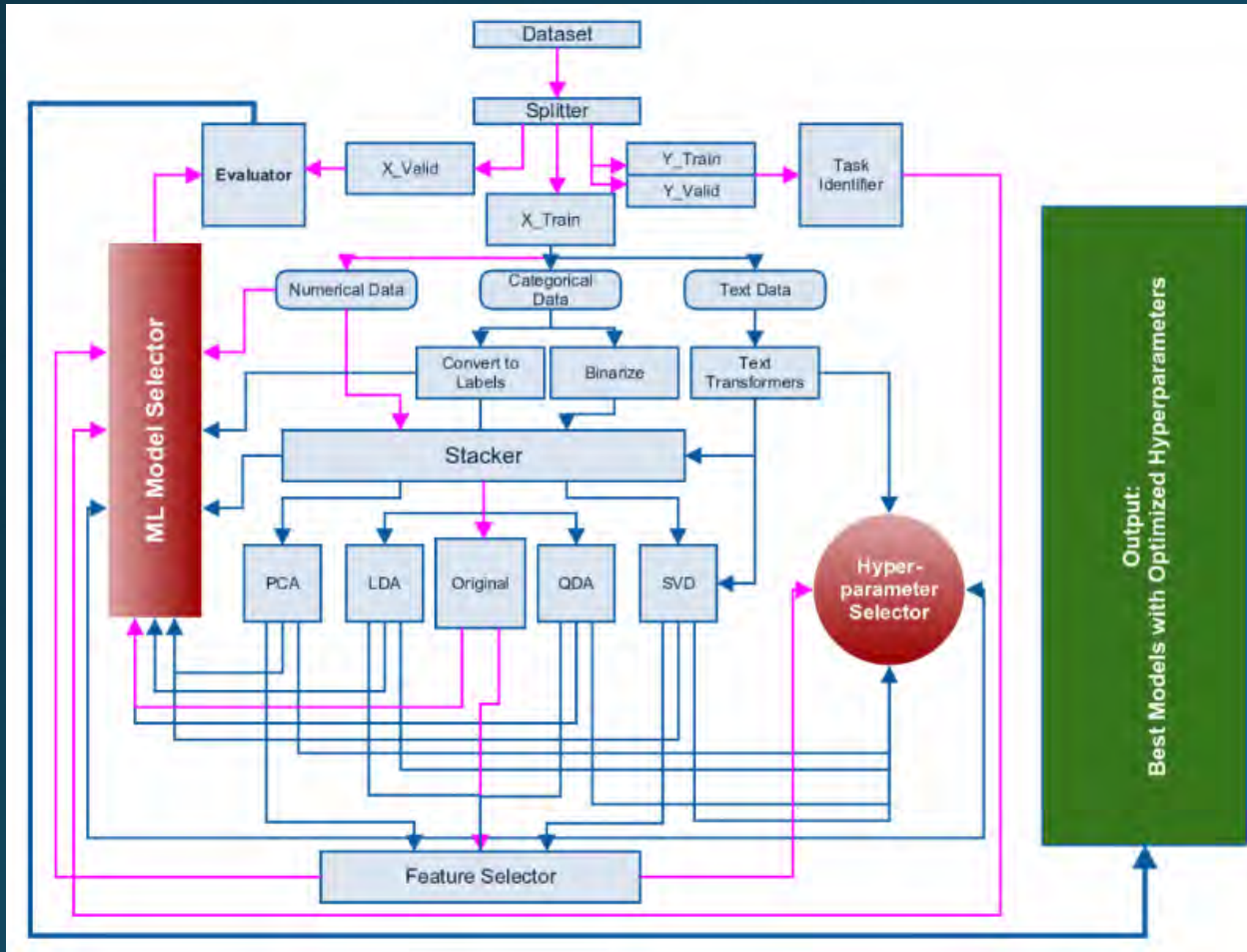
Kaggle Dataset and Leaderboard



- Judging the performance of your learning model
- Should trust CV more than public LB.

- Final rank!
- Differ from public leaderboard (overfitting!)

Kaggle Machine Learning Framework



Don't get afraid!

<http://blog.kaggle.com/2016/07/21/approaching-almost-any-machine-learning-problem-abhishek-thakur/>

Data Preparation: Basics

- Basics: select, filter, missing data, duplicates, ...
- Remove useless features and dimensionality
 - Curse of dimensionality
- Be careful ! Don't drop data easily otherwise in risk of distorting dataset



Data Preparation: Missing Values

- Encoding missing values NAs:
 - Categorical features:
 - Unique encode: “unknown”, “missing”
 - Binary features:
 - 1 for positives and -1 for negatives. 0 for missing values
 - Numerical features
 - Encode as a big positive or negative number: 99999 or -9999
- Impute missing values: mean, median, etc ..
- KNN or other algorithm imputation

Data Preparation: Feature Removal

- Duplicate features: columns sharing same distribution or high correlation
 - Only need to keep one of them (cautious!)
- Constant features: one unique value or near-zero variance
 - `df.unique()`
 - `df.describe()`
 -
- Be careful with removal of any feature!

```
In [51]: df_titanic['Parch'].unique()  
Out[51]: array([0, 1, 2, 5, 3, 4, 6])
```

df_titanic.describe()							
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Data Visualization

- Boxplot to check distribution
- Scatter plot to check outliers, distribution or clusters
- Histogram or counterplot
- Correlations between features
- More exploratory data analysis

Data visualization

The screenshot shows the 'Allstate Claims Severity' competition page on Kaggle. The page header includes the Allstate logo and the competition title. Below the header, there are tabs for 'Overview', 'Data', 'Kernels', 'Discussion', 'Leaderboard', and 'More'. A 'New Kernel' button is located on the right. The 'Kernels' tab is selected, showing a list of 113 kernels. The kernels are sorted by 'Most Votes'. A blue circle highlights the first kernel, 'Exploratory study on ML algorithms', which has 317 votes and was created 7 months ago by Santhosh Sharma. The kernel is categorized under 'Visualization' and 'Py'.



Allstate
You're in good hands.

Allstate Claims Severity
How severe is an insurance claim?
3,055 teams · 5 months ago





Overview Data **Kernels** Discussion Leaderboard More [New Kernel](#)

113 kernels Sort by Most Votes

All Mine

317   Exploratory study on ML algorithms
run 7 months ago by [Santhosh Sharma](#)

All Languages Visualization

   Py 70 

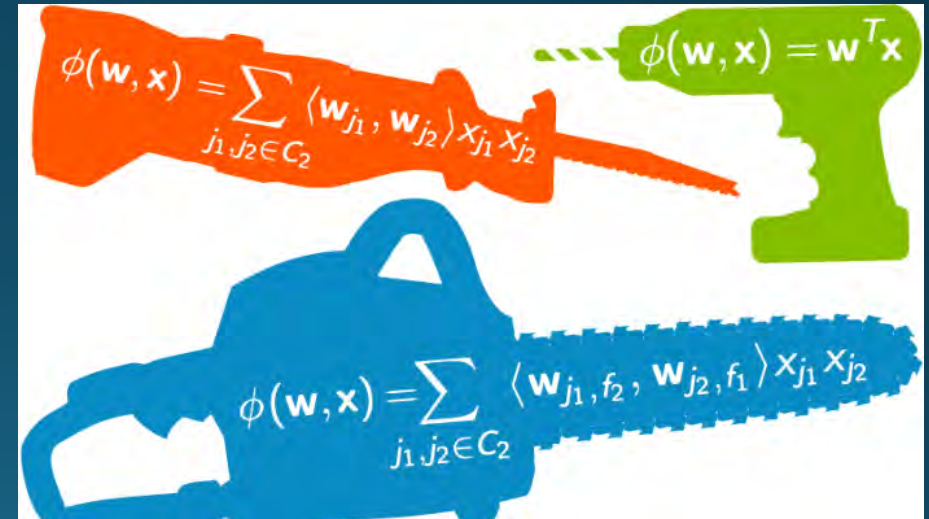
Feature Engineering: **Key** Procedure!



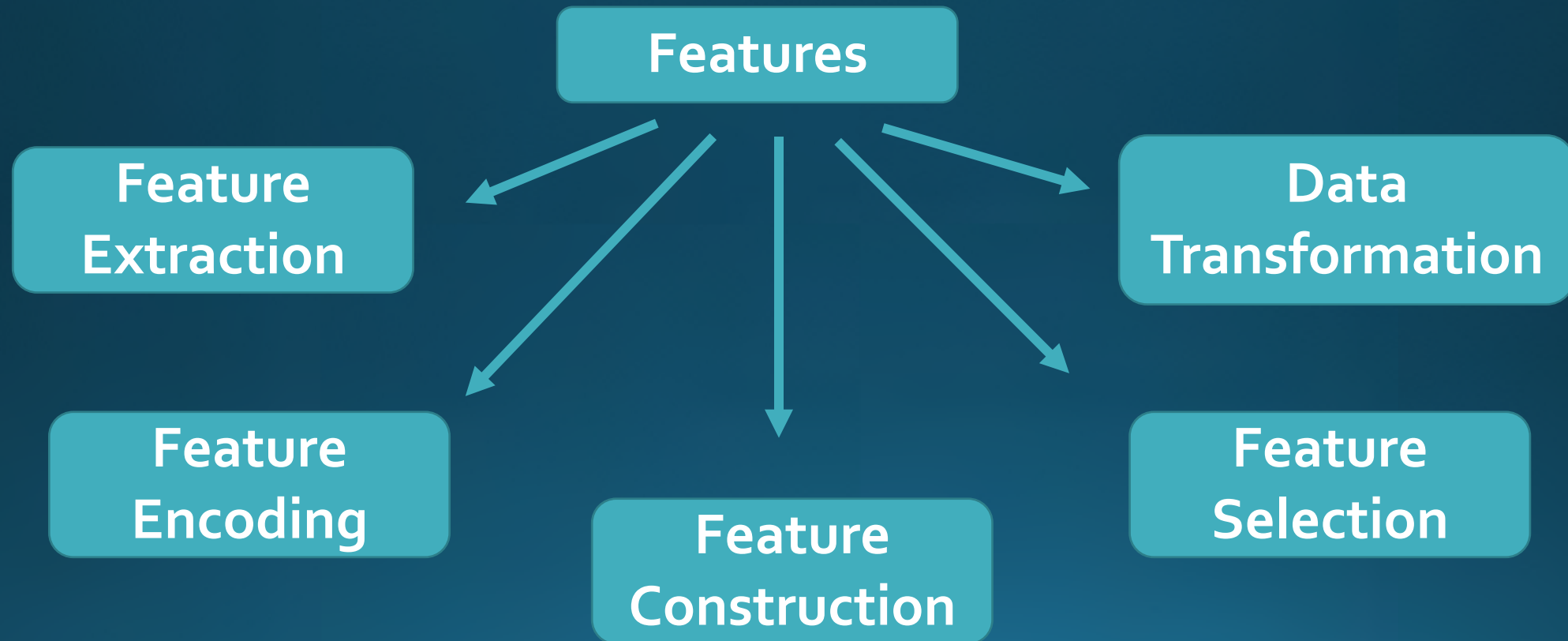
Simple model, best model!

Feature engineering: 60 ~ 80%

Algorithm improvement: 20 ~ 40%



Feature Engineering: **Key Procedure!**



I. Feature Extraction: Text Data

- Bag of words
 - A multiset or bag of words that appear in a text
 - Do not encode any of the text's syntax, order; disregard the grammar

```
(1) John likes to watch movies. Mary likes movies too.
```

```
(2) John also likes to watch football games.
```

```
Bag of words: [ "John", "likes", "to", "watch", "movies", "also",  
"football", "games", "Mary", "too" ]
```

```
Sentence (1): [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]  
Sentence (2): [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]
```

I. Feature Extraction: Text Data

- Bag of words:
 - BeautifulSoup: parse HTML data (tag, attributes, etc)
 - re & formatting: punctuation, numbers, lowercase, etc.
 - NLTK: stop words, stemming & lemmatization.
 - sklearn.feature_extraction: CountVectorizer, TfidfVectorizer

```
def review_to_words( raw_review ):  
    # Function to convert a raw review to a string of words  
    # The input is a single string (a raw movie review), and  
    # the output is a single string (a preprocessed movie review)  
    #  
    # 1. Remove HTML  
    review_text = BeautifulSoup(raw_review).get_text()  
    #  
    # 2. Remove non-letters  
    letters_only = re.sub("[^a-zA-Z]", " ", review_text)  
    #  
    # 3. Convert to lower case, split into individual words  
    words = letters_only.lower().split()  
    #  
    # 4. In Python, searching a set is much faster than searching  
    # a list, so convert the stop words to a set  
    stops = set(stopwords.words("english"))  
    #  
    # 5. Remove stop words  
    meaningful_words = [w for w in words if not w in stops]  
    #  
    # 6. Join the words back into one string separated by space,  
    # and return the result.  
    return( " ".join( meaningful_words ) )
```

```
>>> from sklearn.feature_extraction.text import CountVectorizer  
>>> count_vect = CountVectorizer()  
>>> X_train_counts = count_vect.fit_transform(twenty_train.data)  
>>> X_train_counts.shape  
(2257, 35788)
```

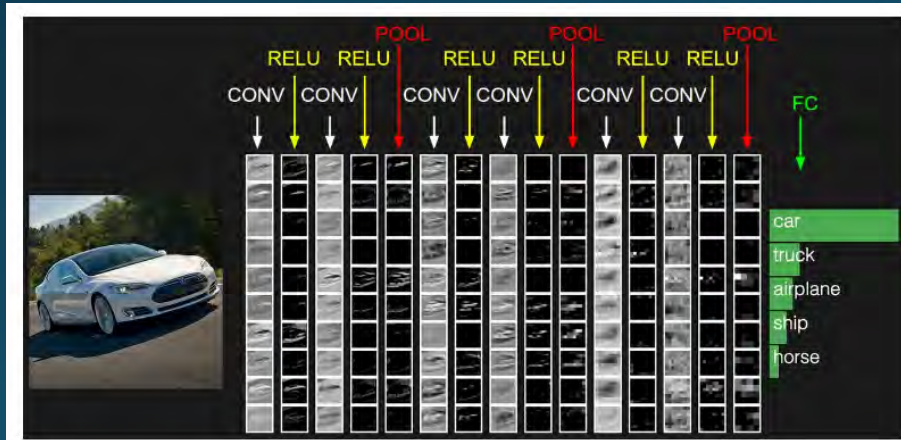
```
>>> from sklearn.feature_extraction.text import TfidfTransformer  
>>> tf_transformer = TfidfTransformer(use_idf=False).fit(X_train_counts)  
>>> X_train_tf = tf_transformer.transform(X_train_counts)  
>>> X_train_tf.shape  
(2257, 35788)
```

- Word2vec

Ref: <https://www.kaggle.com/c/word2vec-nlp-tutorial>

I. Feature Extraction: Image

- Patch extraction:
 - `sklearn.feature_extraction.image.extract_patches_2d`
- Deep learning: Convolutional neural network

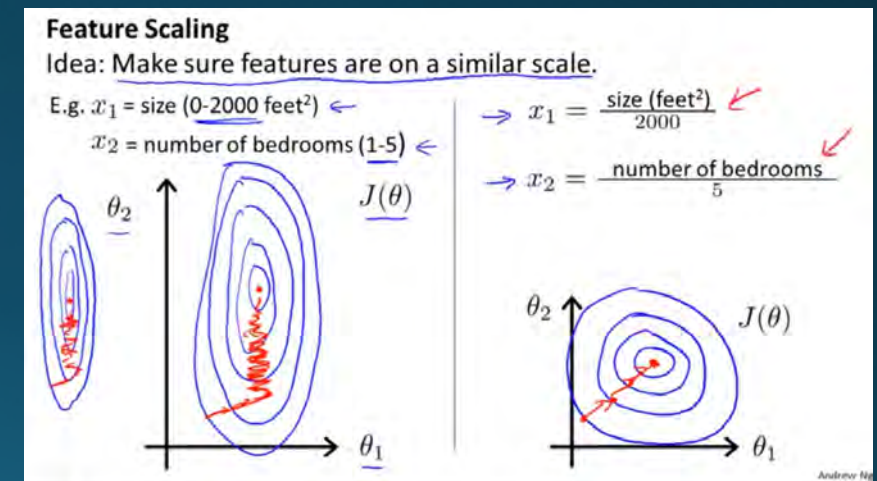


<http://cs231n.github.io/convolutional-networks/>

II. Data Transformation

1. Feature scaling:

- Avoid dominance of features with large numerical values
- Speed up the gradient descent training
- Improve the performance greatly for some models
- Python scikit-learn: MinMaxScaler, StandardScaler



II. Data Transformation

- Models that are sensitive to rescaling: distance sensitive models!
 - Regularized linear models
 - KNN
 - SVM
 - K-Means
- Models that are NOT sensitive to rescaling:
 - Tree based methods
 - Naive Bayes

II. Data Transformation

2. Normalizing transformations

- When variable shows a skewed distribution: “symmetry broken”
- Normal distribution
- Methods: sqrt(x+n), log(x+n), box-cox;
- Exploratory data analysis

	Default	Name of	
Transformation	Parameter	New Variable	Equation
log(Y+a)	$a = 0$	Log_Y	$\log(Y + a), \quad Y + a > 0$
log10(Y+a)	$a = 0$	Log10_Y	$\log_{10}(Y + a), \quad Y + a > 0$
sqrt(Y+a)	$a = 0$	Sqrt_Y	$\sqrt{Y + a}, \quad Y + a > 0$
exp(Y)		Exp_Y	$\exp(Y)$
power(Y;a)	$a = 1$	Pow_Y	$Y^a, \quad Y > 0$ if a is not integral
arcsinh(Y)		Arcsinh_Y	$\log(Y + \sqrt{Y^2 + 1})$
Box-Cox(Y;a)	MLE	BC_Y	See text.

<http://support.sas.com/documentation/cdl/en/stsug/62259/HTML/default/viewer.htm#ugvartransform.html>

III. Feature Encoding

1. Label encoding:

- Encode the categorical features as ordered integers
- “Encoded integers” are usually meaningless.

```
>>> le = preprocessing.LabelEncoder()
>>> le.fit(["paris", "paris", "tokyo", "amsterdam"])
LabelEncoder()
>>> list(le.classes_)
['amsterdam', 'paris', 'tokyo']
>>> le.transform(["tokyo", "tokyo", "paris"])
array([2, 2, 1]...)
>>> list(le.inverse_transform([2, 2, 1]))
['tokyo', 'tokyo', 'paris']
```

III. Feature Encoding

2. One hot encoding:

- Encode categorical features into sparse matrix with binary (0,1) values
- Better option!
- pandas dummy variable or scikit-learn: OneHotEncoder

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4

One hot encoding

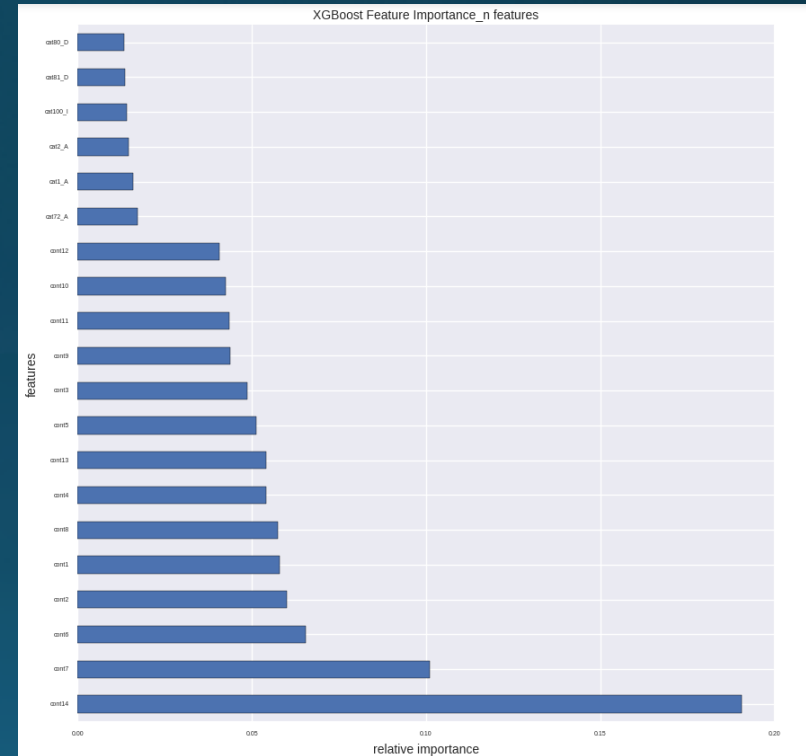
Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

IV. Feature Selection

- Select a subset of features to increase the performance of machine learning algorithm
- Reduce the cost of computing
- Filter out redundant/irrelevant/noisy features or outliers
- “Curse of dimensionality”

IV. Feature Selection


- Selection of subset features:
 - `sklearn.feature_selection.RFE`
- Feature importance:
 - Information gain/Gini impurity: Tree models
 - L1 regularization: Lasso
- Dimensionality reduction:
 - PCA, LDA, etc
 - Be careful! Risk of losing important features by ignoring the non-linearity between features
- Always with attention to drop any feature!



V. Feature Construction

- Generate new features
 - Dates: year, month, hour, weekday/weekend, season, time span, ...
 - ...
- Computationally efficient
- Generalizable to different algorithms
- Domain knowledge
- Hard work, creativity & luck!

Allstate Claims Severity Competition



Allstate
You're in good hands.

Allstate Claims Severity






How severe is an insurance claim?
3,055 teams · 5 months ago

[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [More](#) [My Submissions](#) [Submit Predictions](#)

Overview

[Description](#)
[Evaluation](#)
[Timeline](#)

When you've been devastated by a serious car accident, your focus is on the things that matter the most: family, friends, and other loved ones. Pushing paper with your insurance agent is the last place you want your time or mental energy spent. This is why **Allstate**, a personal insurer in the United States, is continually seeking fresh ideas to improve their claims service for the over 16 million households they protect.



Allstate is currently developing automated methods of predicting the cost, and hence severity, of claims. In this recruitment challenge, Kagglers are invited to show off their creativity and flex their technical chops by creating an algorithm which accurately predicts claims severity. Aspiring competitors will demonstrate insight into better ways to predict claims severity for the chance to be part of Allstate's efforts to ensure a worry-free customer experience.

[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [More](#) [My Submissions](#) [Submit Predictions](#)

Overview

[Description](#)
[Evaluation](#)
[Timeline](#)

Submissions are evaluated on the **mean absolute error (MAE)** between the predicted loss and the actual loss.

Submission File

For every id in the test set, you should predict the loss value. The file should contain a header and have the following format:

```
id,loss
4,0
6,1
9,99,3
etc.
```

Allstate Claims Severity Competition

- Predict expected loss value for insurance claims
- Evaluation metrics: mean absolute error (MAE)
 - MAE: non-differentiable
- 3055 teams participated ultimately

I. Data Preparation

Basic information of the dataset:

- Training data: (188318, 132)

Test data: (125546, 131)

- 130 raw features
 - 116 categorical features
 - 14 numerical features

- No missing values in all the raw data. 😊

- However ... 😞

train.head()

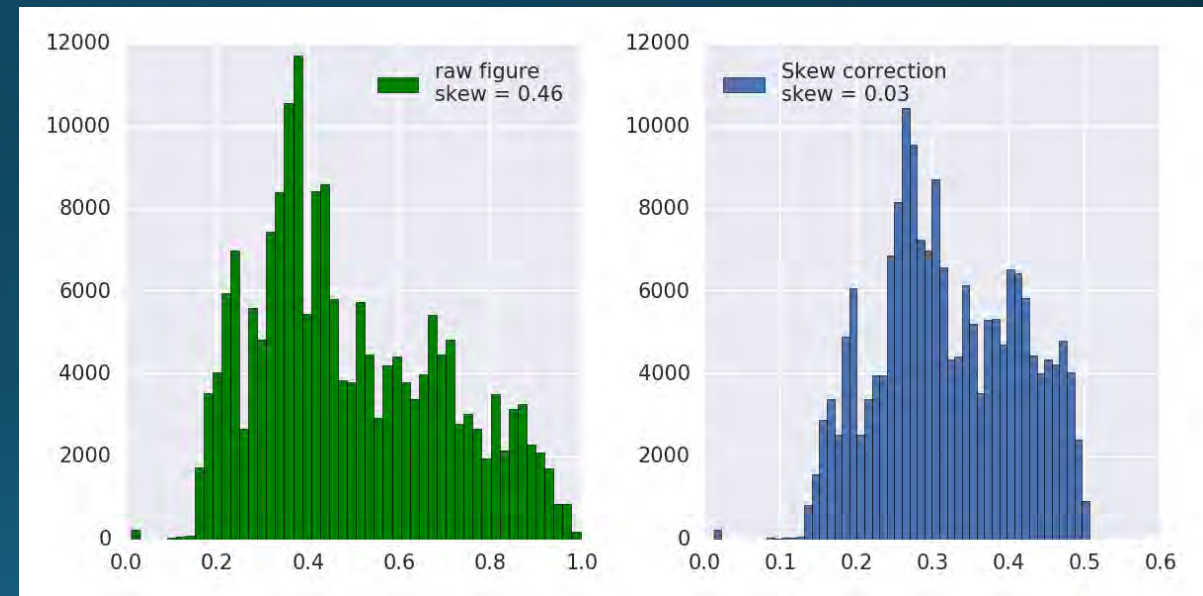
	id	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	...	cont6	cont7	cont8	cont9	cont10	cont11	cont12	cont13	cont14
0	1	A	B	A	B	A	A	A	A	B	...	0.718367	0.335060	0.30260	0.67135	0.83510	0.569745	0.594646	0.822493	0.71484
1	2	A	B	A	A	A	A	A	A	B	...	0.438917	0.436585	0.60087	0.35127	0.43919	0.338312	0.366307	0.611431	0.30449
2	5	A	B	A	A	B	A	A	A	B	...	0.289648	0.315545	0.27320	0.26076	0.32446	0.381398	0.373424	0.195709	0.77442
3	10	B	B	A	B	A	A	A	A	B	...	0.440945	0.391128	0.31796	0.32128	0.44467	0.327915	0.321570	0.605077	0.60264
4	11	A	B	A	B	A	A	A	A	B	...	0.178193	0.247408	0.24564	0.22089	0.21230	0.204687	0.202213	0.246011	0.43260

5 rows x 132 columns

II. Feature Engineering: Numeric Features

For skewed numerical features:

1. `scipy.boxcox(); scipy.skew()`
2. `sqrt(sklearn.preprocessing.minmax_scale())`



III. Feature Engineering: Categorical Features

1. One hot encoding the categorical features

```
from sklearn.preprocessing import OneHotEncoder
onehotenc = OneHotEncoder(categorical_features= 'all')
df_onehot_dummy = pd.get_dummies(df)
df_onehot_onehotenc = DataFrame(onehotenc.fit_transform(df))
```

2. Feature selection:

	feat
0	A
1	B
2	C

OHE

	feat_A	feat_B	feat_C
0	1.0	0.0	0.0
1	0.0	1.0	0.0
2	0.0	0.0	1.0

Training dataset

	feat
0	A
1	B
2	D

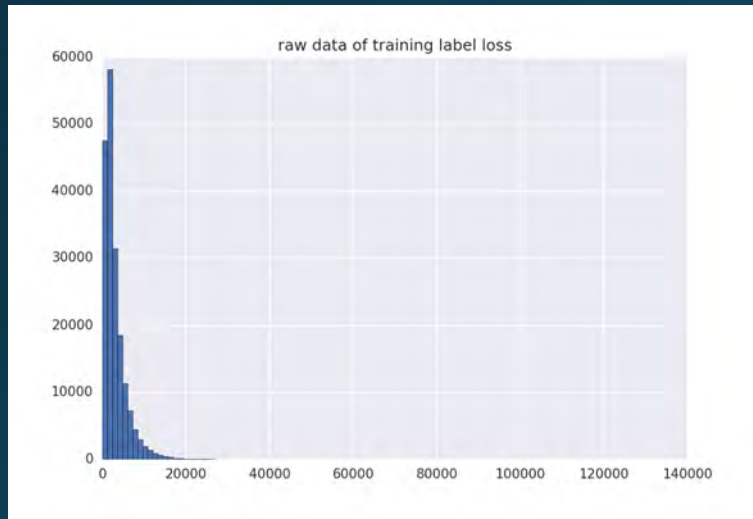
OHE

	feat_A	feat_B	feat_D
0	1.0	0.0	0.0
1	0.0	1.0	0.0
2	0.0	0.0	1.0

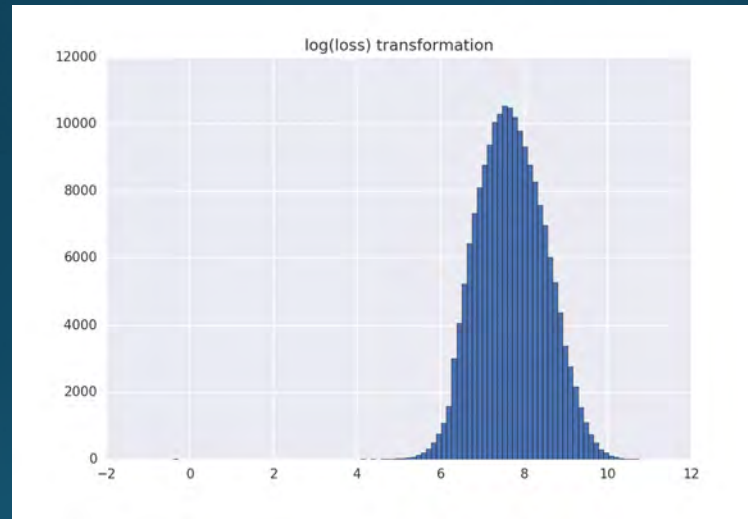
Test dataset

IV. Data Transformation: Label

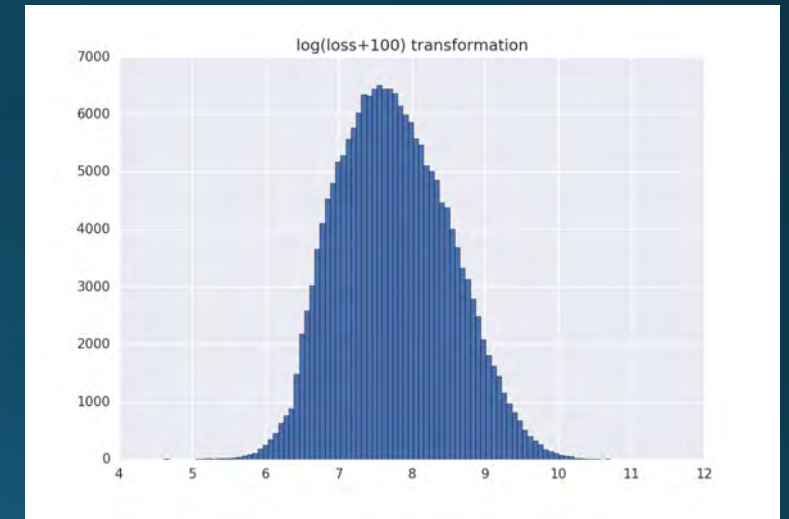
Label before transformation



Label after $\log()$



Label after $\log(x + n)$



Change the evaluation metrics correspondingly.



Kaggle II. Modelling

Henry

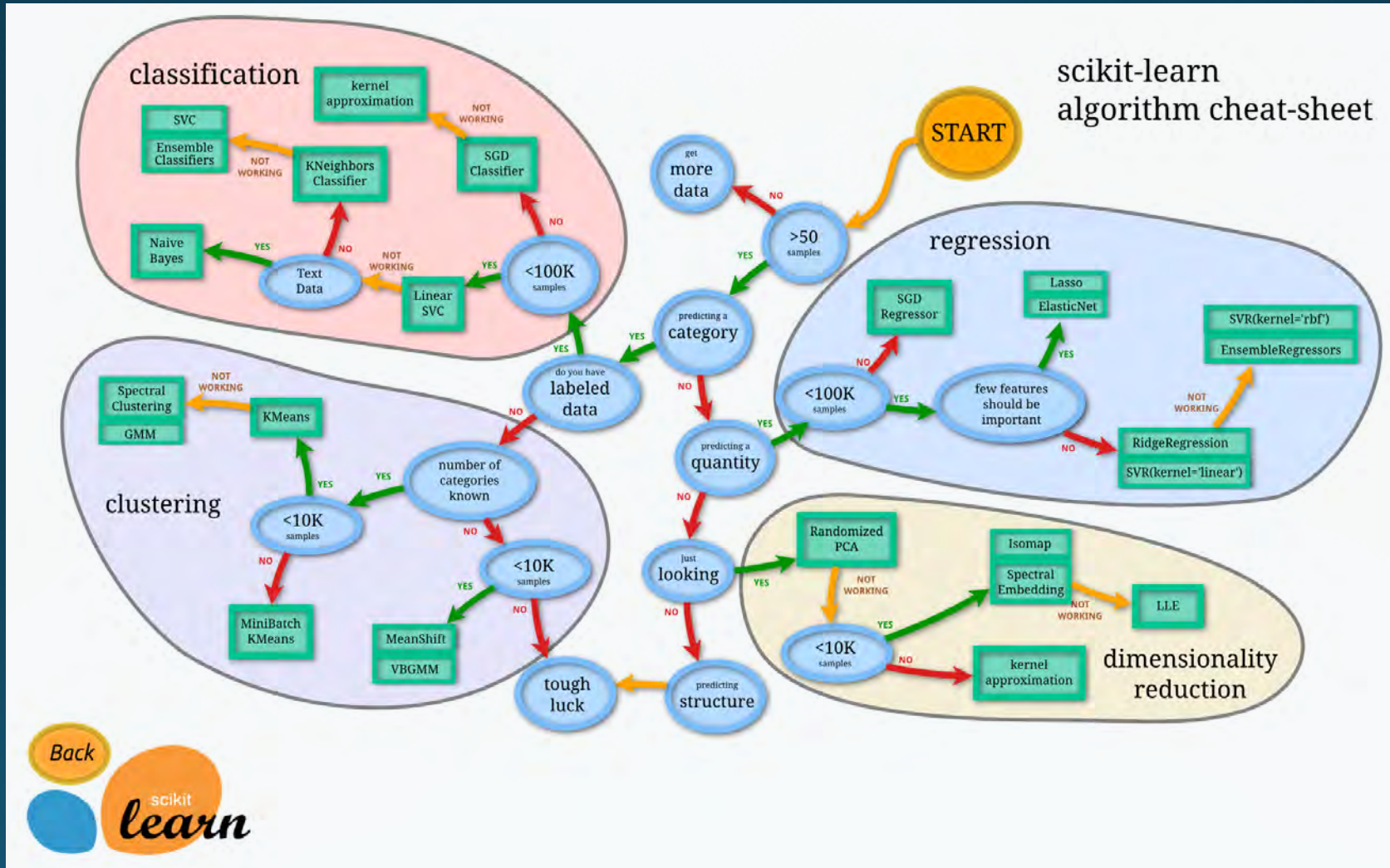
Build Your Models

- Start from simple: single best model
- Model ensemble: architecture
- Fine tune

Basic Model: Start from Simple

- Get your first model to work.
 - Cross-validation: 5 ~ 10 fold usually
 - Understand model evaluation metrics
 - Initial hyper-parameters tuning of estimators
 - Submission: watch CV score and public LB
- Try different algorithms
- Baseline as a single model

Algorithms:



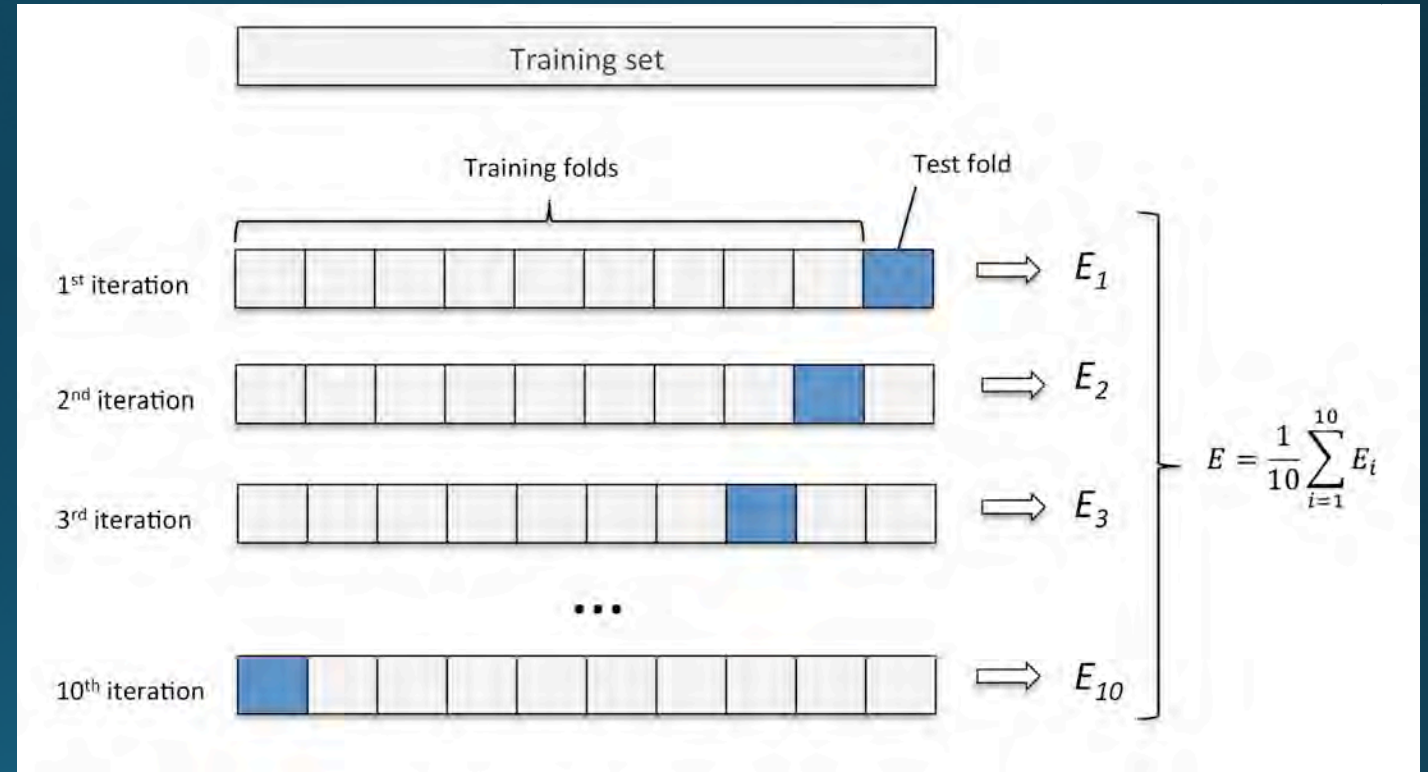
Algorithms:

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

Credit: Mark Peng

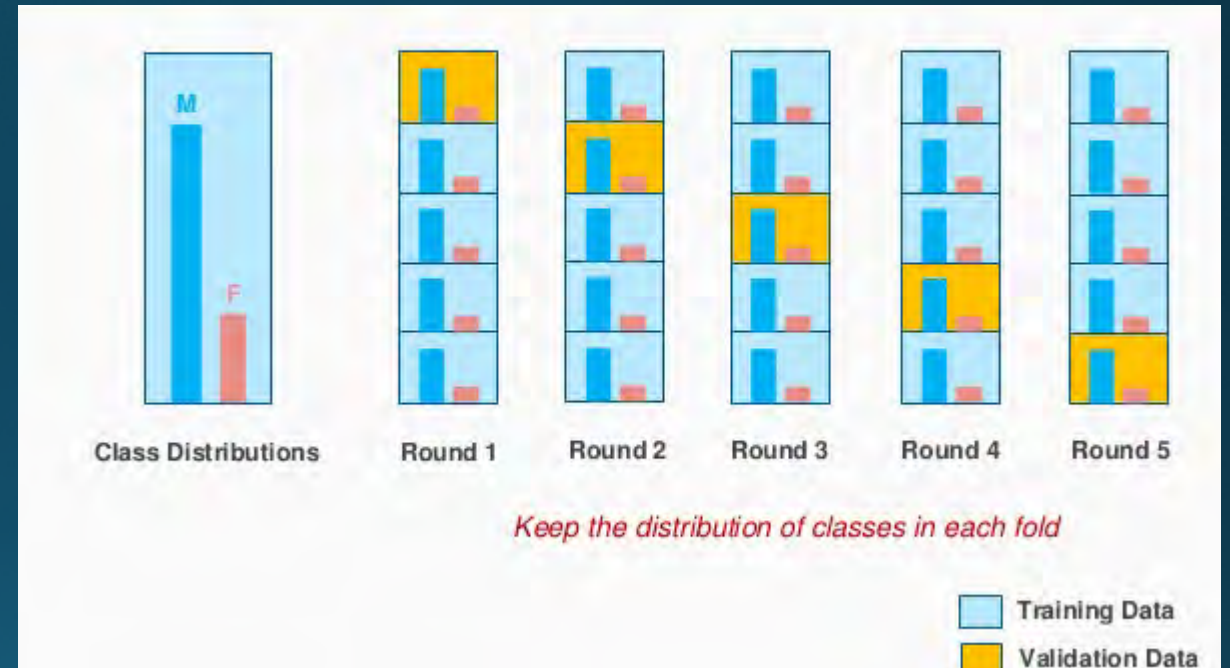
Cross-validation

- Prevent overfitting
- Generalization
- 5 ~ 10 fold
- Out of fold predictions!



Cross-validation

- Imbalanced dataset:
 - StratifiedKFold
 - Down-sampling majority
 - Over-sampling minority



<http://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation>

Model Evaluation:

- Depend on the competition requirements
- Regression:
 - Mean absolute error (MAE)
 - Root mean squared error (RMSE)
 -

Model Evaluation:

- Classification:

- Logarithmic loss (log-loss) or multiclass log-loss

- Mean consequential error (MCE) $MCE = \frac{1}{n} \sum_{y \neq y'} 1$

- Area under curve (AUC)

- Hamming loss $HammingLoss(y, y') = \frac{1}{n_sample} \sum_{i=1}^{n_sample} \frac{xor(y, y')}{n_feature}$

- ...

Model Evaluation:

1. Default evaluation metrics by estimator.
2. Customized evaluation metrics and objective functions:
 - Not differentiable, such as MAE
 - Slow performance
 - Biased in competition
 - Implementation in some algorithm: such as xgboost

Hyper-parameter Optimization

- Grid search
 - Brute force
 - Expensive computation
 - Less efficient
- Random search

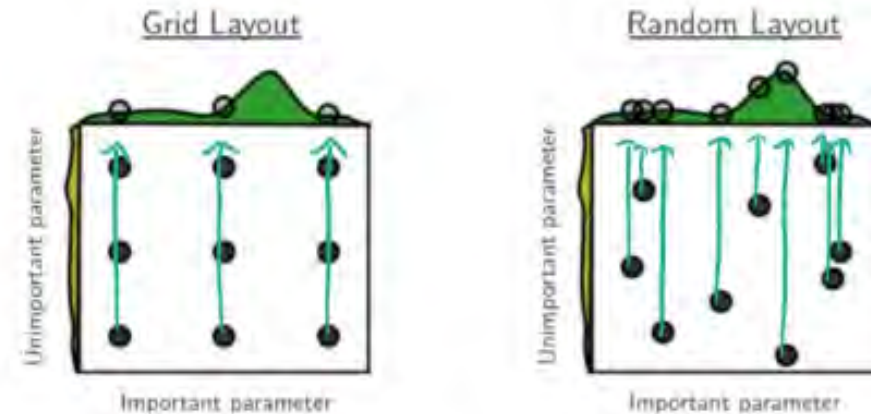


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g . This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

Hyper-parameter Optimization

Hyperopt:

- <https://github.com/hyperopt/hyperopt>
- Random search
- Tree of Parzen Estimators (TPE)

```
# define an objective function
def objective(args):
    case, val = args
    if case == 'case 1':
        return val
    else:
        return val ** 2

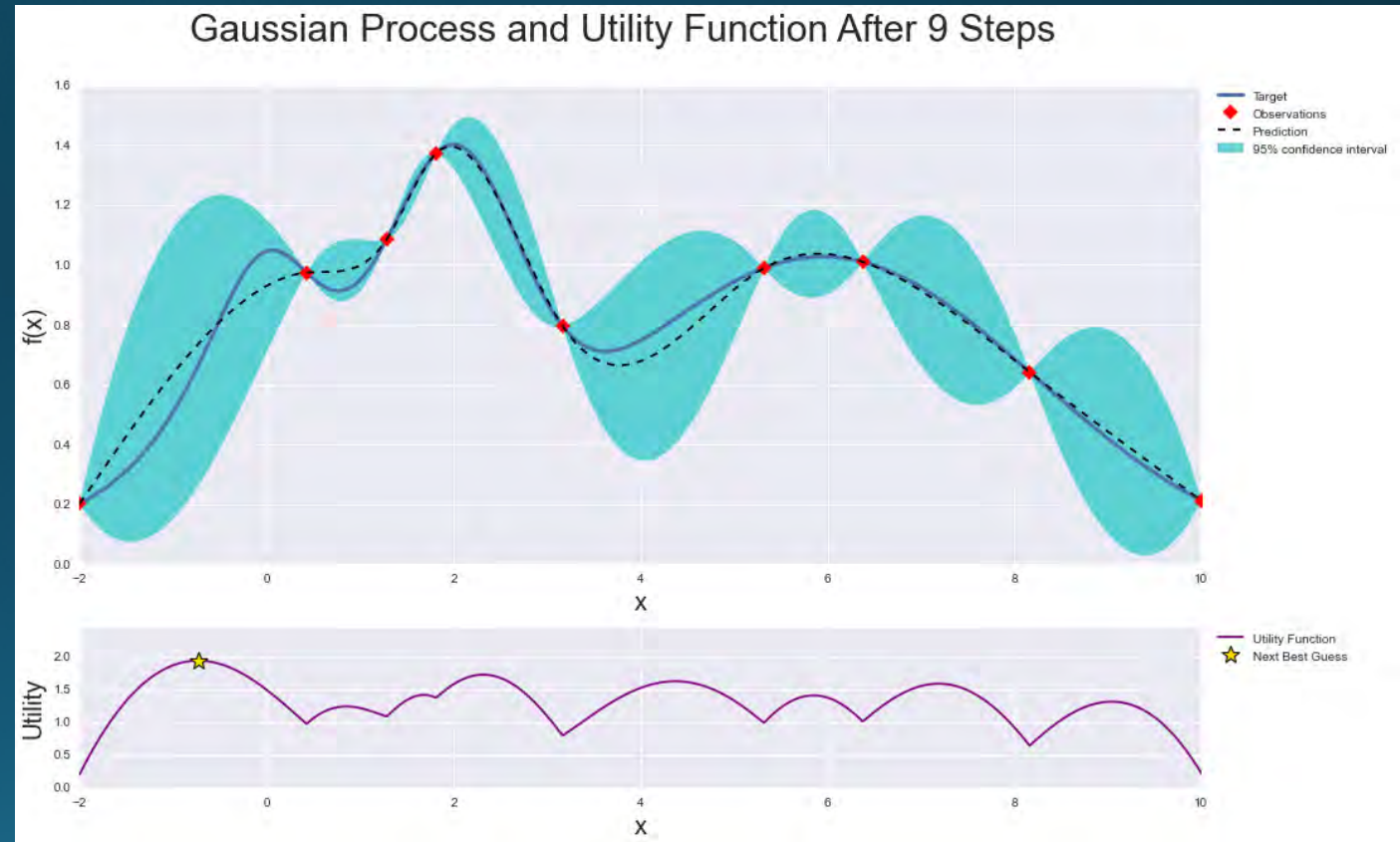
# define a search space
from hyperopt import hp
space = hp.choice('a',
    [
        ('case 1', 1 + hp.lognormal('c1', 0, 1)),
        ('case 2', hp.uniform('c2', -10, 10))
    ])

# minimize the objective over the space
from hyperopt import fmin, tpe
best = fmin(objective, space, algo=tpe.suggest, max_evals=100)

print best
# -> {'a': 1, 'c2': 0.01420615366247227}
print hyperopt.space_eval(space, best)
# -> ('case 2', 0.01420615366247227)
```

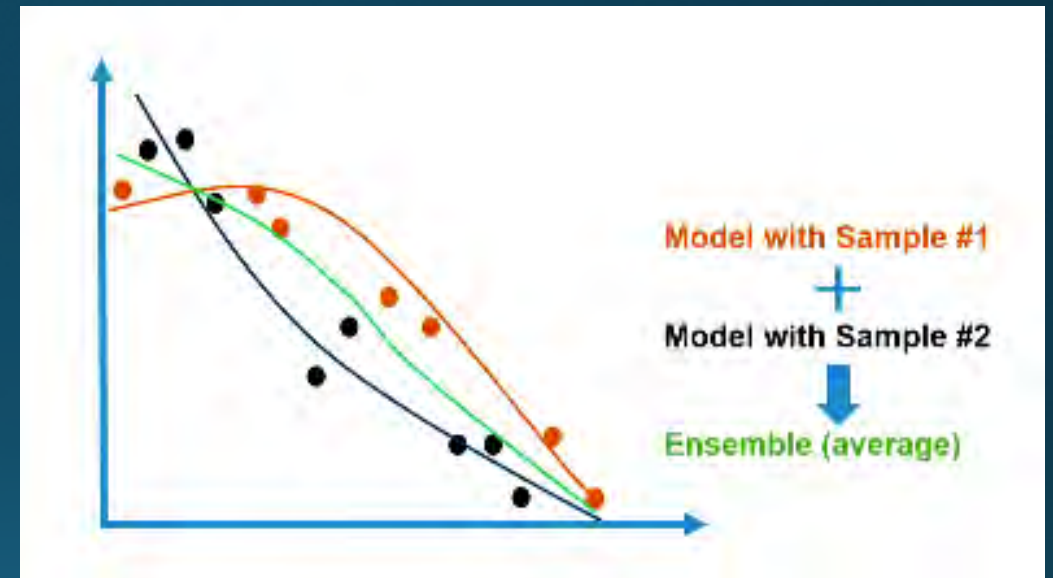
Hyper-parameter Optimization

- BayesOptimization:
 - Fast and efficient
 - <https://github.com/fmfn/BayesianOptimization/>
 - xgboost demo in github



Model Ensemble

- Make weak estimators **STRONG!**
- Less variance
- Less generalization error
- Less possibility of overfitting



Model Ensemble: Bagging

- Bag of models:
 - Same algorithm with different parameters, class weights and subset of features
 - Different algorithms
 - (Weighted) voting or average
 - Diversity!

<https://mlwave.com/kaggle-ensembling-guide/>

Model Ensemble: Bagging

High correlation between models

```
1111111100 = 80% accuracy  
1111111100 = 80% accuracy  
1011111100 = 70% accuracy.
```

These models are highly correlated in their predictions. When we take a majority vote we see no improvement:

```
1111111100 = 80% accuracy
```

Low correlation between models

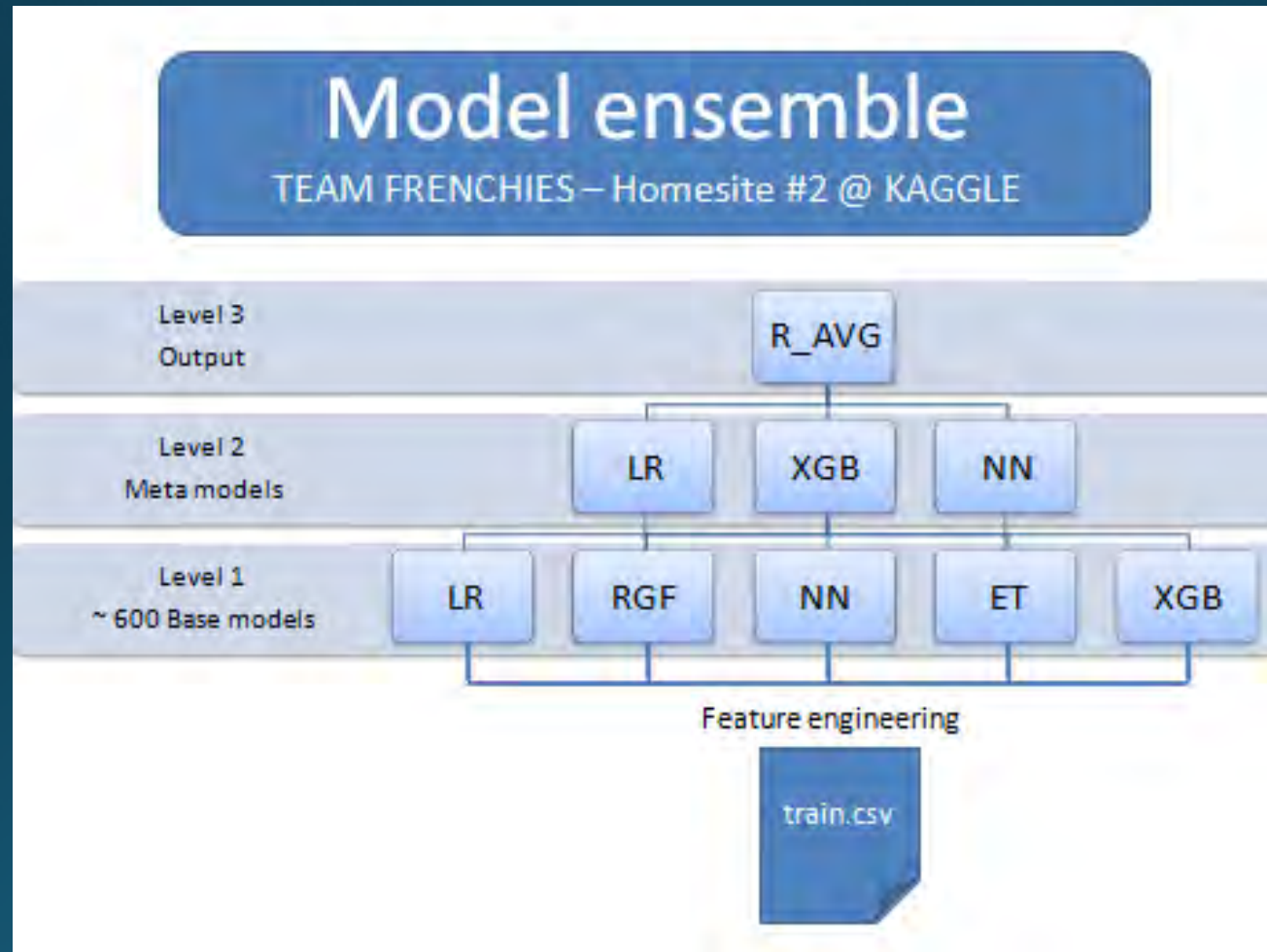
```
1111111100 = 80% accuracy  
0111011101 = 70% accuracy  
1000101111 = 60% accuracy
```

When we ensemble this with a majority vote we get:

```
1111111101 = 90% accuracy
```

<https://mlwave.com/kaggle-ensembling-guide/>

Model Ensemble: Architecture



Architecture I. Blending

1. Create a holdout of 10% of the train set
2. Fit n first stage models on the 90% train set without the holdout
3. Optimize the weights of second stage model using the holdout dataset
 - Brute force
 - A stacker model

Architecture I. Blending

- Pros:
 - Working better than bagging
 - Simpler than stacking
 - More prone to overfitting.
- Cons:
 - Less data, especially on the 2nd stage

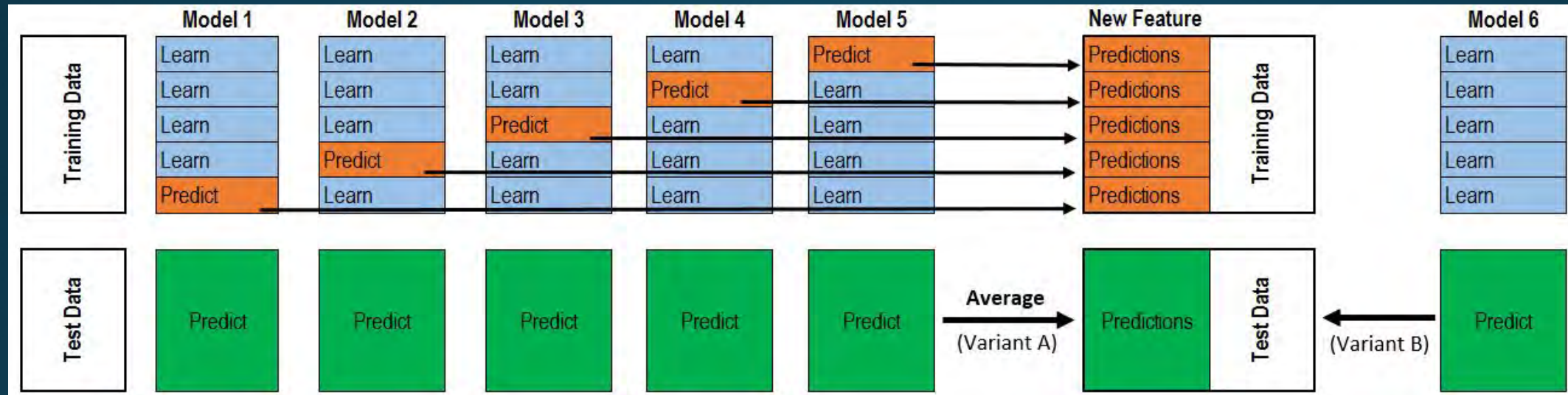
Model Ensemble: Stacked Generalization

- Winning solution!

The basic idea behind stacked generalization is to use a pool of base classifiers, then using another classifier to combine their predictions, with the aim of reducing the generalization error.

- Instead of simple averaging, we train the predictions from the first layer

Architecture II. Stacked Generalization



Out-of-fold predictions as new features!

<https://www.kaggle.com/mmueller/discussion>

Architecture II. 2-Fold Stacking

- Split the train set in 2 parts: train_a and train_b
- Fit a first-stage model on train_a and create predictions for train_b
- Fit the same model on train_b and create predictions for train_a
- Finally fit the model on the entire train set and create predictions for the test set.
- Now train a second-stage stacker model on the probabilities from the first-stage model(s).

<https://mlwave.com/kaggle-ensembling-guide/>

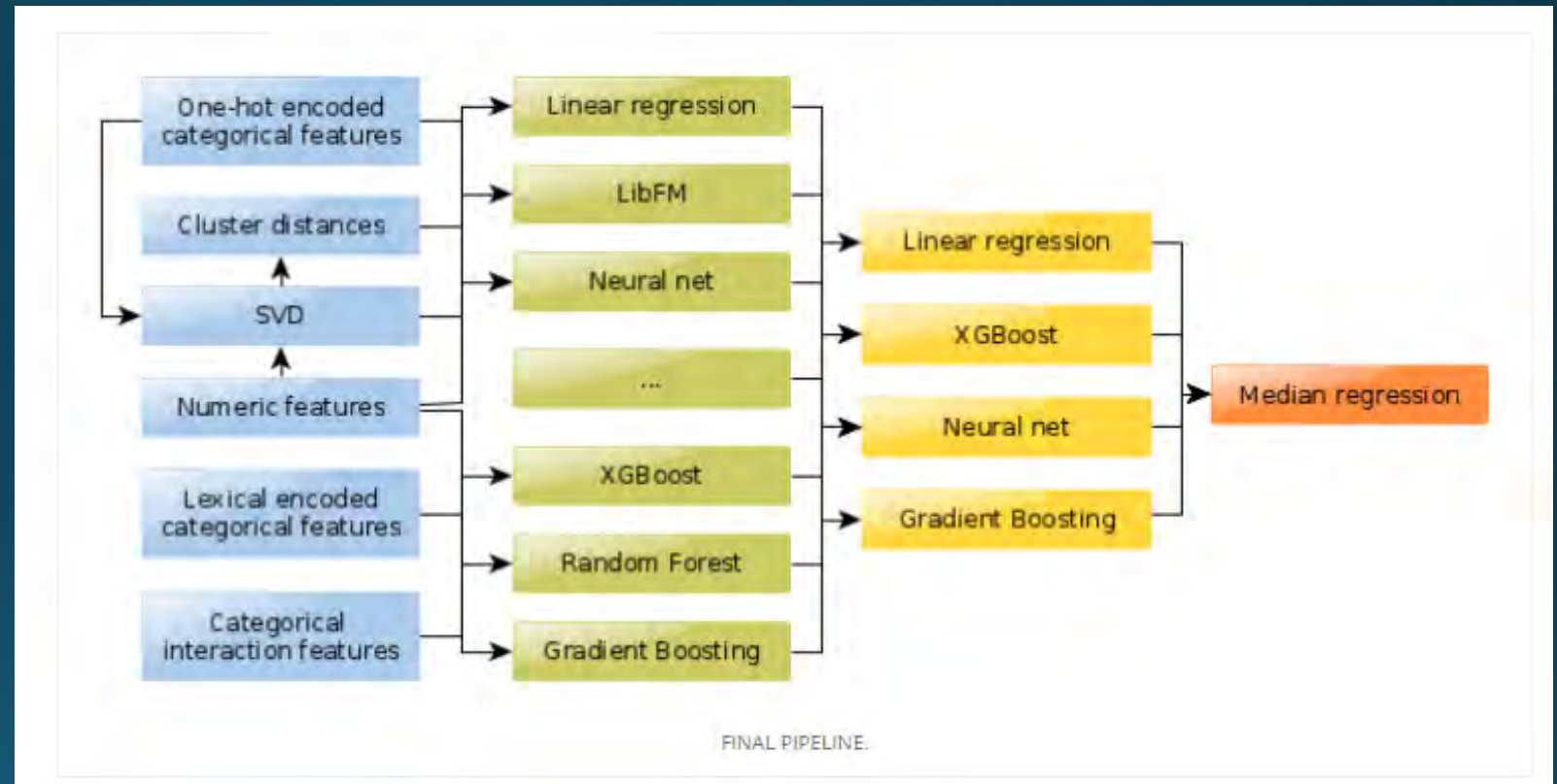
Architecture II. Stacking

- Pros:
 - Efficiently utilizing the dataset
 - Winner solution! Work the best in practice
- Cons:
 - Information leakage

Real Game: Allstate Claims Severity

2nd Place Winner:

Alexey Noskov



<http://blog.kaggle.com/2017/02/27/allstate-claims-severity-competition-2nd-place-winners-interview-alexey-noskov/>

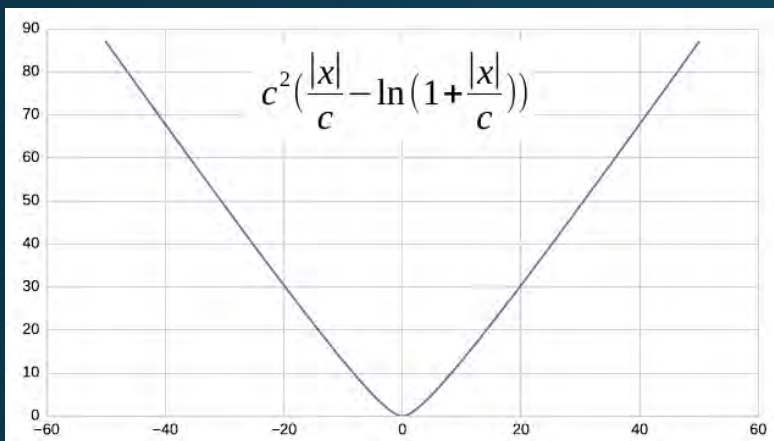
Feature Engineering

1. Feature encoding: one-hot-encoding
2. Feature selection: SVD & PCA
3. Feature construction:
 - Unsupervised methods: distance to cluster centers
 - Lexical encoding on categorical interaction features

First Level Models

Linear regression 1189.65998	Random forest 1176.44433	Extra trees 1166.85430	Gradient boosting 1126.30971	LibFM 1150.37290	xgboost 1105.43686	Nerual network 1116.44915
------------------------------------	--------------------------------	---------------------------	------------------------------------	---------------------	-----------------------	---------------------------------

xgboost



Replace $|x|$ with this objective function

Averaging

Averaging multiple runs of XGBoost with different seeds - it helps to reduce model variance;

Features

Adding categorical combination features;

Obj. func.

Modifying objective function to be closer to MAE;

Tuning

Tuning model parameters

Neural Network

Bagging	Averaging multiple runs, again;
Moving Average	Applying exponential moving average to weights of single network;
Features	Adding SVD and cluster features;
Batch norm. & Dropout	Adding batch normalization and dropout;

Second Level Models



Only Neural Network: enough to get top-16 in public, and top-8 in private

Final Layer: Optimizing Weights

Trick 1.

- Training model on many subsamples and averaging predictions;

Trick 2.

- Reducing input dimensionality - grouping similar models of previous layers and using group averages as features;

Trick 3.

- Averaging best 10 submissions for a final one

One More Thing To Look

Nerual network
(two layers)

1098.91721

Final solution
(three layers)

1098.07061

This is Kaggle!

Tips:

- Insight of the datasets
- Feature engineering
- Model ensemble, especially stacking
- Partner!

Thank
you

Remember
to have
FUN.