

# kaggle<sup>TM</sup> Tutorial

김연민

Kaggle Break

# Who?



**yeonmin**

at

Seongnam-si, Gyeonggi-do, South Korea











Joined 2 years ago · last seen in the past day

Followers 7

Following 13



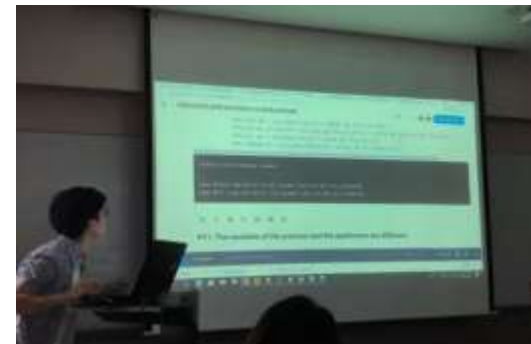
Competitions  
Expert

	<b>Zillow Prize: Zillow's Home Value Prediction (Zestimate)</b> Can you improve the algorithm that changed the world of real estate? <i>Featured</i> · 7 months ago · housing, real estate	 43/3779 Top 2%
	<b>Mercedes-Benz Greener Manufacturing</b> Can you cut the time a Mercedes-Benz spends on the test bench? <i>Featured</i> · 6 years ago · automobiles, tabular data, regression	 72/3835 Top 2%
	<b>Two Sigma Connect: Rental Listing Inquiries</b> How much interest will a new rental listing on RentHop receive? <i>Recruitment</i> · 1 year ago · housing, tabular data, text data, multiclass classification	 117/2488 Top 5%
	<b>Quora Question Pairs</b> Can you identify question pairs that have the same intent? <i>Featured</i> · 6 years ago · linguistics, internet, tabular data, text data, duplicate detection	 197/3307 Top 6%
	<b>Toxic Comment Classification Challenge</b> Identify and classify toxic online comments <i>Featured</i> · 4 months ago · arguments, text data	 336/4551 Top 8%

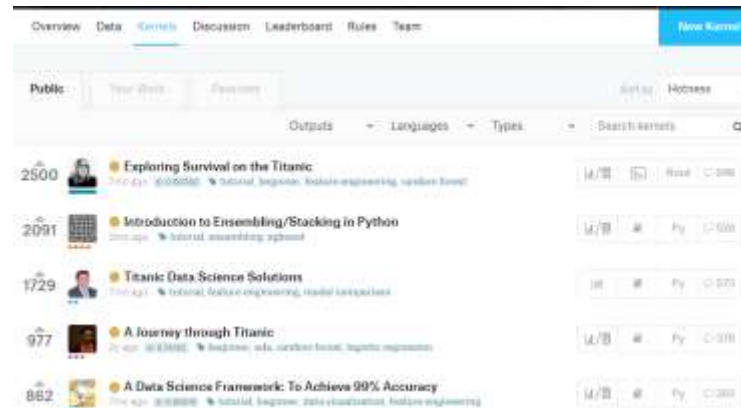
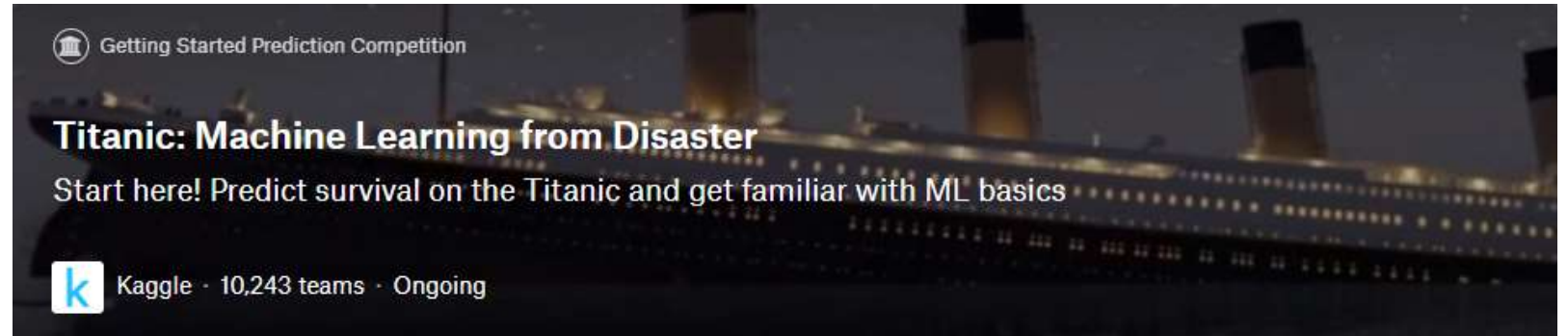
오늘 Tutorial을 도와주실 분들  
의 소개가 있겠습니다~



<https://facebook.com/groups/kagglebreak/>



# Why?



여러가지 강의보고

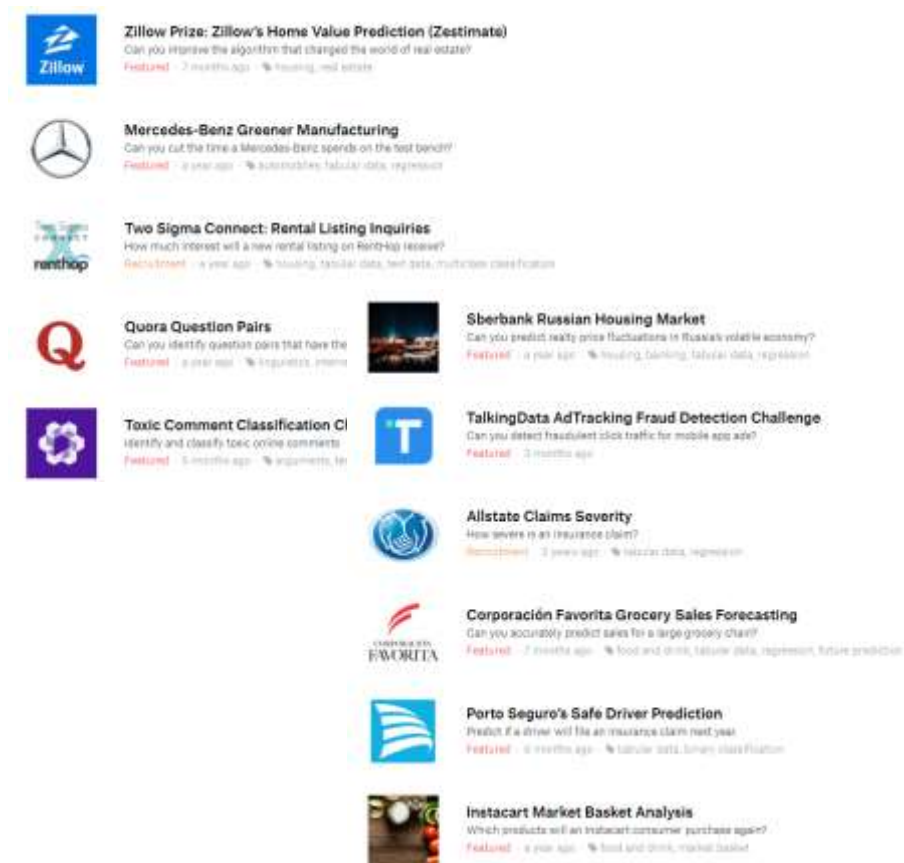
문제를 풀어보고 싶어서 무작정  
kaggle에서 Titanic부터 시작

아무것도 몰랐고  
Kernel 따라해가면서 하나 하나씩  
무식하게 공부했습니다.

[manuallymodified.csv](#)  
2 years ago by yeonmin  
[add submission details](#)

0.82775

# Why?



저처럼 아무것도 모르는 상태에서 시작하면  
힘들고 지치고 흥미를 잃어버리기 쉽기때문에

처음 시작하시는 분들이 알면 좋은 내용

Kaggle을 재밌게 할 수있는 방법, Know-how을 공유

궁극적으로는 Kaggle Leaderboard에 한국사람이 많아졌  
으면 좋겠습니다.

# Tutorial Contents

- **What is Kaggle ?**
- **Feature Engineering Part 1**
  - 실습
- **Feature Engineering Part 2**
  - 실습
- **Modeling**
- **Winner Solution**
  - 실습

# Tutorial Contents

How to Win a Data Science Competition: Learn  
from Top Kagglers

by National Research University Higher School of Economics

**Coursera 강의와**

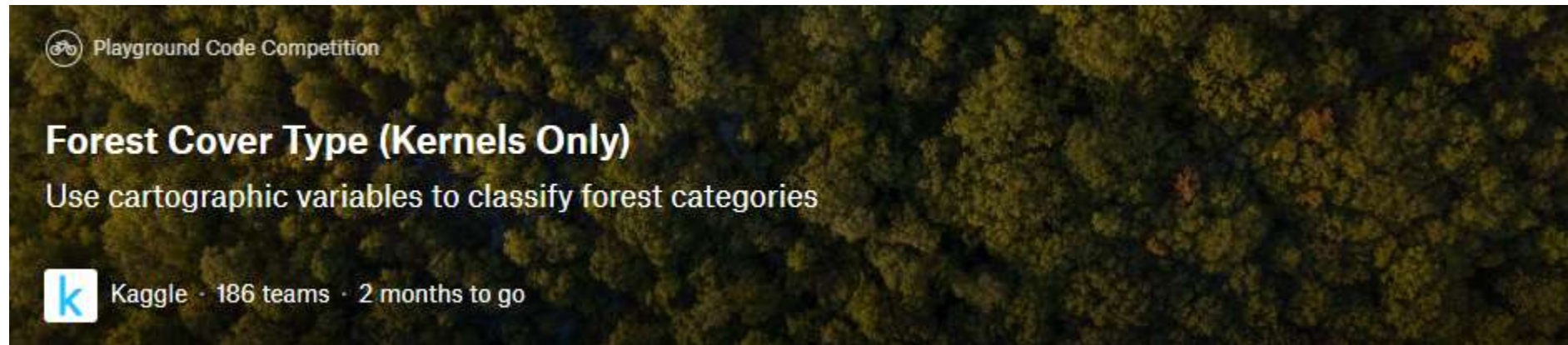
**2년간 Kaggle에서 배운 내용을 바탕으로 만들어졌습니다.**

**Data Science Competition** 잘 하시고 싶은 분이면 이 강의 꼭 들으세요! 강추입니다. 진작 나왔으면..

<https://www.coursera.org/learn/competitive-data-science/>



# Tutorial Data

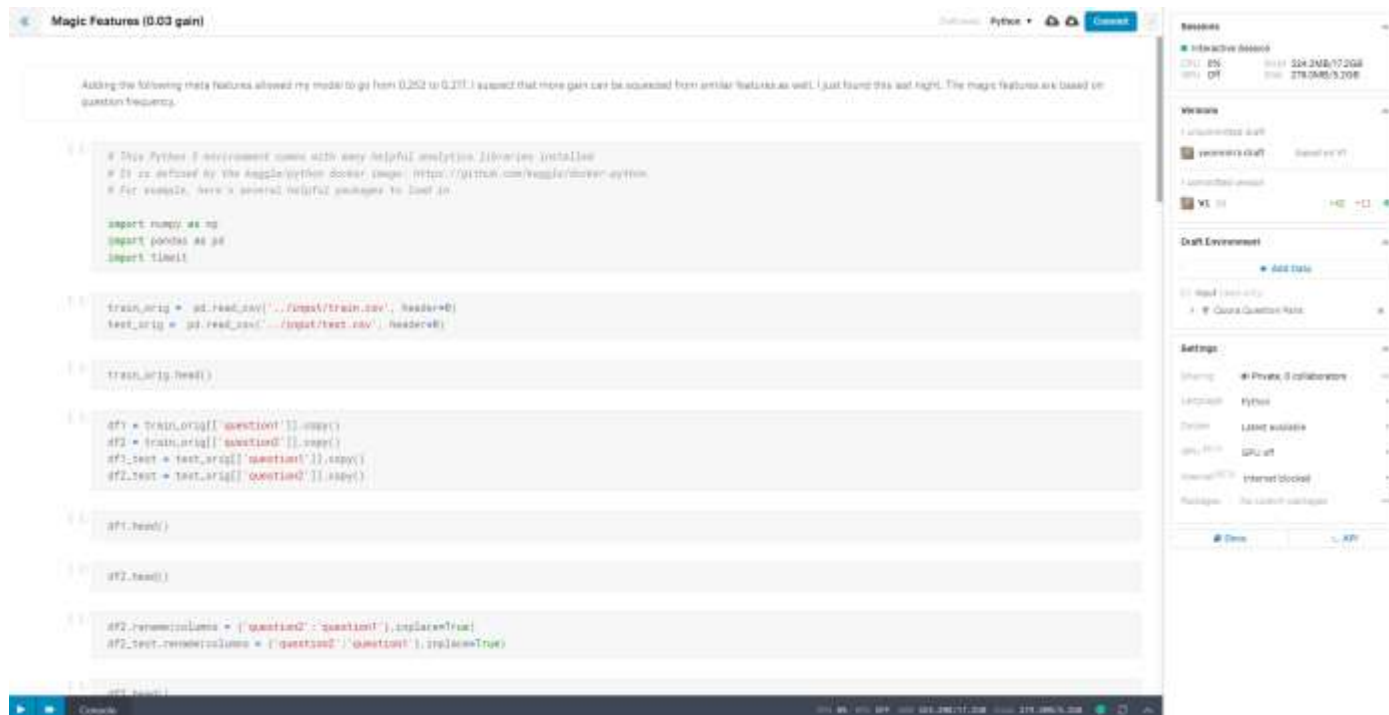


<https://www.kaggle.com/c/pycon-korea-2018-tutorial>

# 실습 환경

Kaggle Kernel을 활용하여 진행하도록 하겠습니다.

※ 개인 노트북 있으신 분은 개인 컴퓨터에서 진행하셔도 됩니다.  
**Anaconda python 3 version + Keras + Lightgbm**



The screenshot displays a Kaggle Kernel interface. The main area contains a code editor with the following Python code:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np
import pandas as pd
import time

train_orig = pd.read_csv("../input/train.csv", header=0)
test_orig = pd.read_csv("../input/test.csv", header=0)

train_orig.head()

df1 = train_orig[['question1']].copy()
df2 = train_orig[['question2']].copy()
df1_test = test_orig[['question1']].copy()
df2_test = test_orig[['question2']].copy()

df1.head()

df2.head()

df2.rename(columns = {'question2': 'question1'}, inplace=True)
df2_test.rename(columns = {'question2': 'question1'}, inplace=True)
```

On the right side, there is a sidebar with system information:

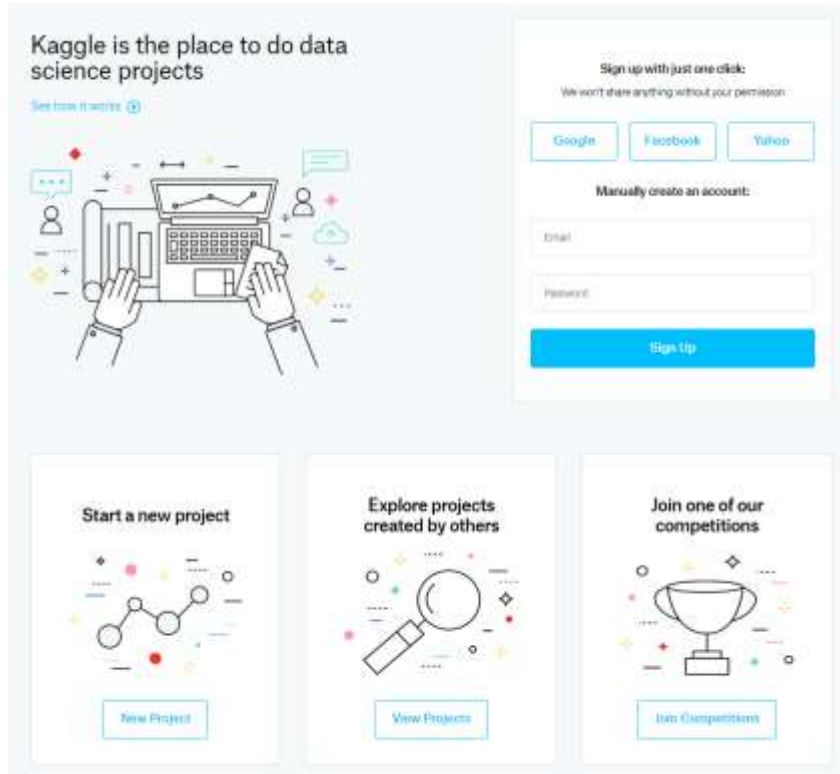
- Basics:** Interactive Session, CPU: 25%, RAM: 524.2MB/7.2GB, GPU: 0%, 278.0MB/3.2GB.
- Versions:** 1 kaggle/docker, 1 tensorflow/tensorflow, 1 tensorflow/tensorflow, 1 tensorflow/tensorflow.
- Draft Environment:** Add Data, Add Data, Add Data.
- Settings:** Sharing: Private & Collaborative, Language: Python, Dataset: Latest available, GPU: GPU, Internet: Internet blocked, Packages: The default packages.

At the bottom, there is a console output area showing the execution of the code.

# Library

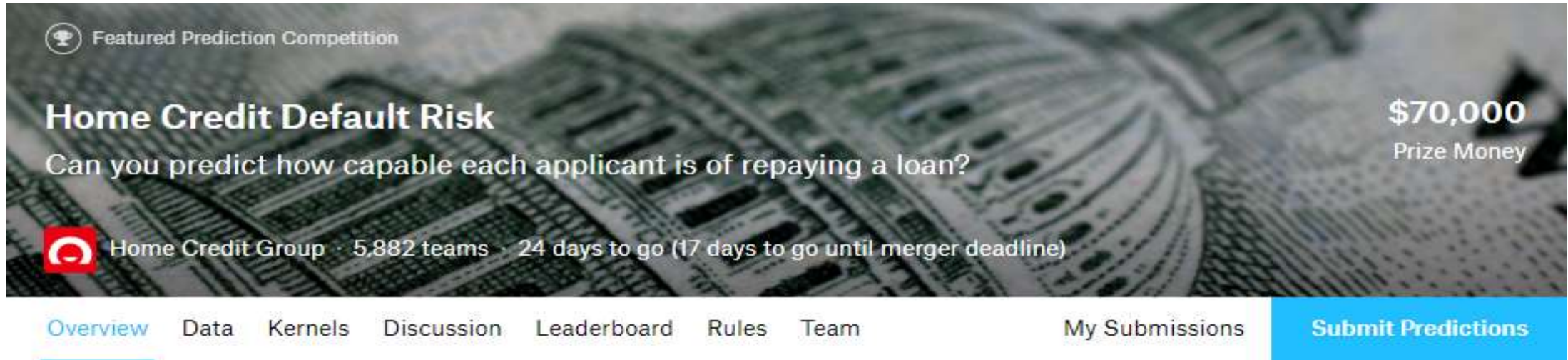
- **Python 3**
- **Numpy**
- **Pandas**
- **Matplotlib**
- **Seaborn**
- **sklearn**
- **lightgbm**

# What is Kaggle?



직접 가서 살펴보도록  
하겠습니다.

# What is Kaggle Competition?



Featured Prediction Competition

## Home Credit Default Risk

Can you predict how capable each applicant is of repaying a loan?

**\$70,000**  
Prize Money

Home Credit Group · 5,882 teams · 24 days to go (17 days to go until merger deadline)

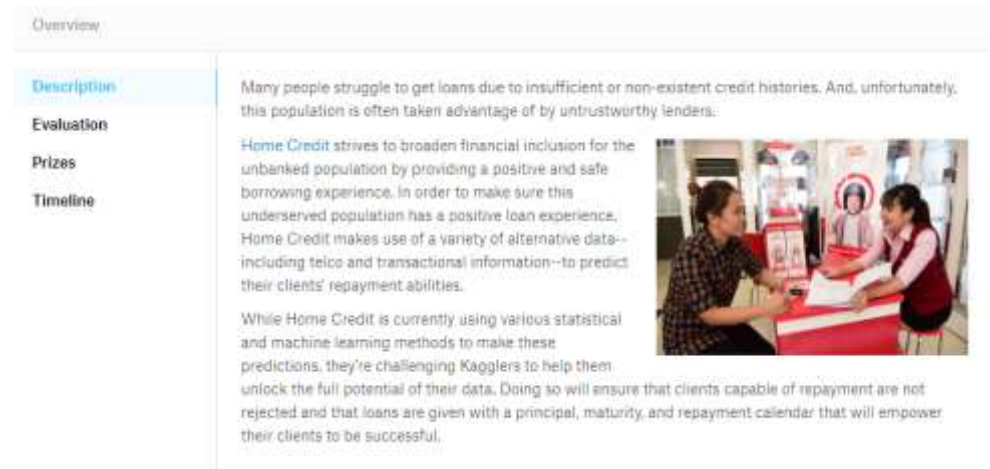
[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [My Submissions](#) [Submit Predictions](#)

보통은 위와 같은 8가지 메뉴가 존재합니다.

## Overview

대회에 대한 전반적인 설명이 존재  
대회 목적, Evaluation, Prizes, Timeline이 존재

대회 처음 Join하게 되면 제일 먼저 살펴봐야 합니다.



Overview

**Description**

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.


**Evaluation**

**Prizes**

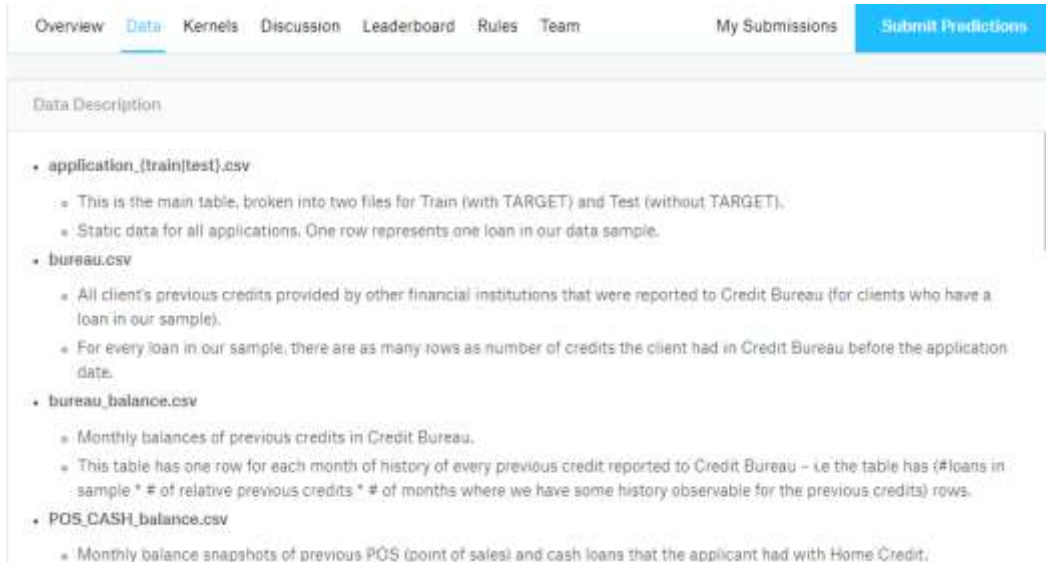
**Timeline**

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.



# What is Kaggle Competition?



The screenshot shows the Kaggle competition interface. At the top, there are navigation tabs: Overview, Data, Kernels, Discussion, Leaderboard, Rules, Team, My Submissions, and Submit Predictions. The 'Data' tab is selected. Below the tabs, the 'Data Description' section is visible. It lists four data files: application\_(train|test).csv, bureau.csv, bureau\_balance.csv, and POS\_CASH\_balance.csv. Each file has a brief description of its contents.

Overview Data Kernels Discussion Leaderboard Rules Team My Submissions Submit Predictions

Data Description

- application\_(train|test).csv
  - This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET).
  - Static data for all applications. One row represents one loan in our data sample.
- bureau.csv
  - All client's previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample).
  - For every loan in our sample, there are as many rows as number of credits the client had in Credit Bureau before the application date.
- bureau\_balance.csv
  - Monthly balances of previous credits in Credit Bureau.
  - This table has one row for each month of history of every previous credit reported to Credit Bureau - i.e the table has (#loans in sample \* # of relative previous credits \* # of months where we have some history observable for the previous credits) rows.
- POS\_CASH\_balance.csv
  - Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.

## Data에 대한 설명과 데이터 파일을 Download

Data는 csv, json, Image 등으로 직접 Download 받을 수 있고 Torrent나 기타 다른 사이트를 통해서 받는 경우도 있습니다.

- Kaggle-renthop.7z - (optional) listing images organized by listing\_id. Total size: 78.5GB compressed. Distributed by BitTorrent (Kaggle-renthop.torrent).

## 일반적인 Data 구성

Train / Test / Sample\_Submission

## File descriptions

- train.json - the training set
- test.json - the test set

# What is Kaggle Competition?

데이터가 항목별로 별도로 존재하여 직접 구성해야 되는 대회도 있습니다.

Data Description	
• application_train/test.csv	» This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET). » Static data for all applications. One row represents one loan in our data sample.
• bureau.csv	» All client's previous credits provided by other financial institutions that were reported to Credit Bureau for clients who have a loan in our sample. » For every loan in our sample, there are as many rows as number of credits the client had in Credit Bureau before the application date.
• bureau_balance.csv	» Monthly balances of previous credits in Credit Bureau. » This table has one row for each month of history of every previous credit reported to Credit Bureau - ie the table has (Months in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.
• POS_CASH_balance.csv	» Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.

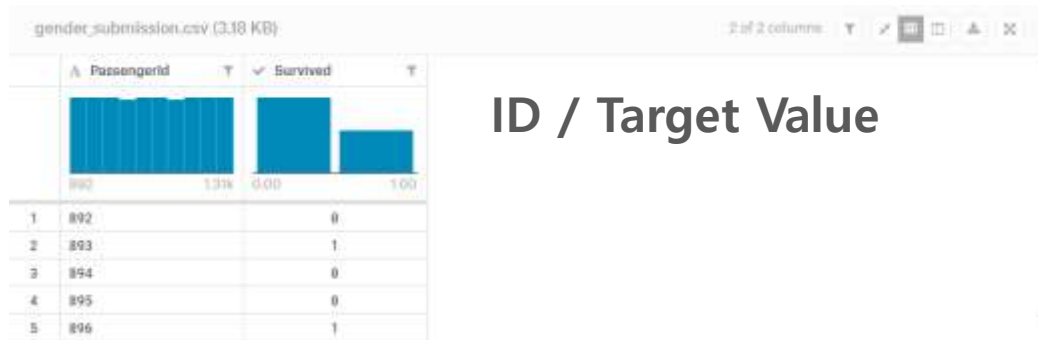
처음에 각 항목의 데이터가 어떤 데이터인지 파악하는 것이 중요합니다.  
Column 상세 명세가 존재할 수도 있습니다.

Table	Row	Description	Special		
1 application_train/test.csv	SK_ID_CURR	ID of loan in our sample			
2 application_train/test.csv	TARGET	Target variable (1 - client with payment difficulties)			
5 application_train/test.csv	NAME_CONTRACT_TYPE	Identification if loan is cash or revolving			
6 application_train/test.csv	CODE_GENDER	Gender of the client			
7 application_train/test.csv	FLAG_OWN_CAR	Flag if the client owns a car			
8 application_train/test.csv	FLAG_OWN_REALTY	Flag if client owns a house or flat			
9 application_train/test.csv	CNT_CHILDREN	Number of children the client has			
10 application_train/test.csv	AMT_INCOME_TOTAL	Income of the client			
11 application_train/test.csv	AMT_CREDIT	Credit amount of the loan			
12 application_train/test.csv	AMT_ANNUITY	Loan annuity			



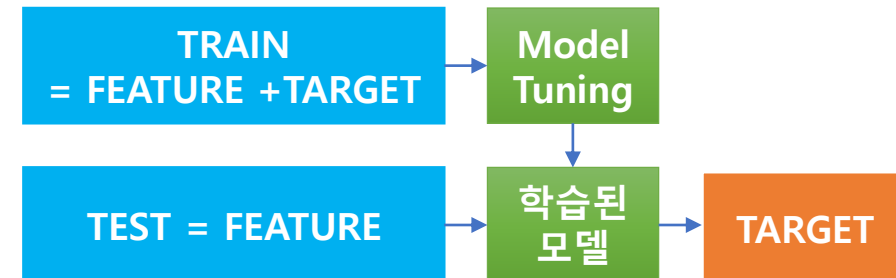
# What is Kaggle Competition?

## Sample Submission



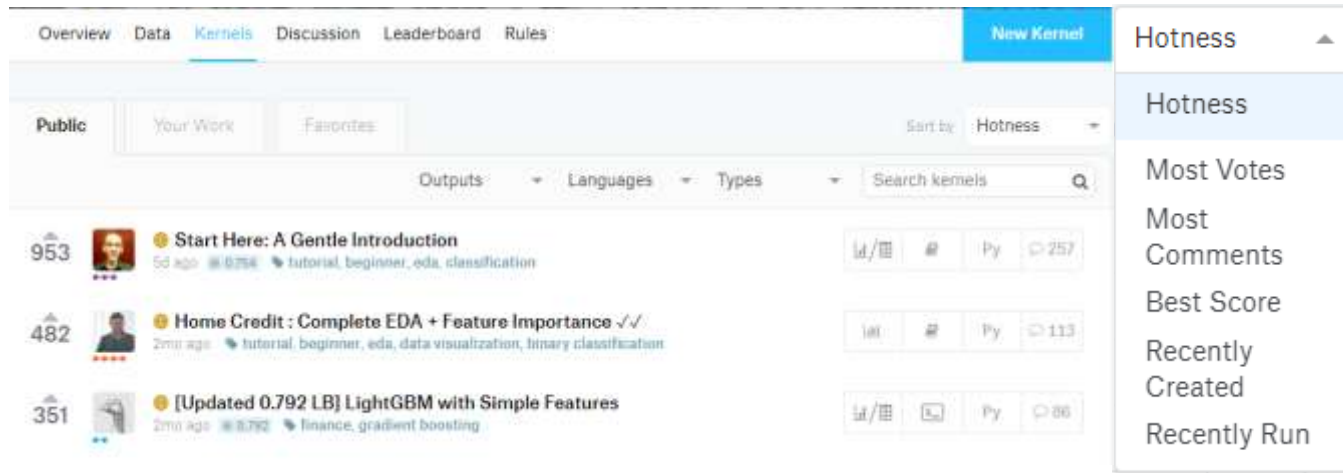
ID는 Test Item 하나에 대한 고유한 값이고,  
Target 값은 우리가 예측해야 되는 값입니다.

Target값은 Train에만 존재하고, Test에는 존재하지 않기 때문에  
Train Data로 학습하여 Model을 만들고  
이 모델을 바탕으로 Test Data의 Target 값을 예측하는 것입니다.



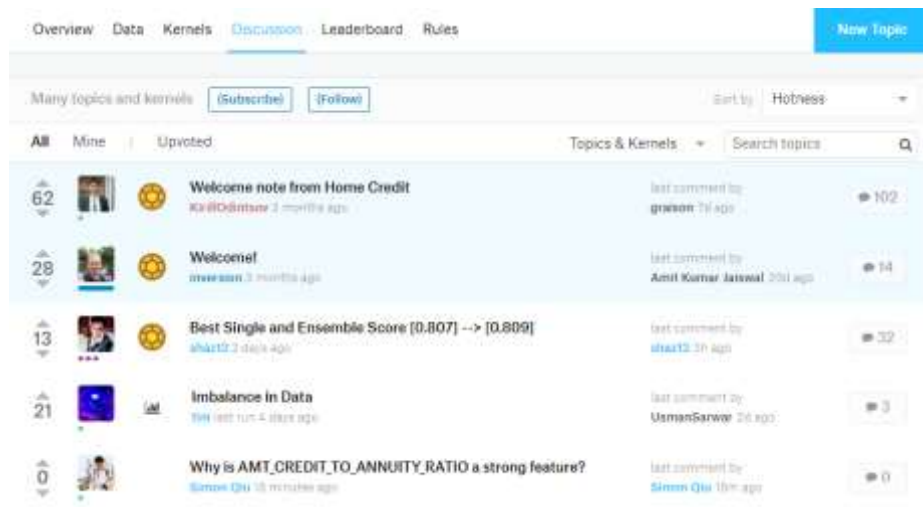


# What is Kaggle Competition?



**Discussion**은 다 읽어보는 것이 좋습니다.  
다양한 생각, 좋은 방법들을 배울 수 있습니다.

**Kernel**도 처음에는 한 줄 한 줄 따라해보는 것이 좋습니다.



# What is Kaggle Competition?

Public Leaderboard

Private Leaderboard

This leaderboard is calculated with approximately 20% of the test data.  
The final results will be based on the other 80%, so the final standings may be different.

[Raw Data](#) [Refresh](#)

In the money

Gold

Silver

Bronze

#	△1w	Team Name	Kernel	Team Members	Score ?	Entries	Last
1	▲4	Large hypothesis space			0.809	181	1d
2	▲124	SandHill			0.809	189	4h
3	▼2	vegetable chicken			0.809	303	5h
4	▲2	Simplicity			0.808	127	3d

대회 기간

대회가 끝나고 실제 Final Score

Public Leaderboard

Private Leaderboard

**Public과 Private 2가지  
LeaderBoard가 존재**

**Public Leaderboard**  
대회기간 동안 **Test데이터의 일부  
만 사용하여 Score 측정**

**Private Leaderboard**  
대회 기간이 끝나고 **나머지 Test  
데이터를 사용하여 Score 측정**

왜 처음부터 모두 사용하지 않을까?  
**Test data** 값 하나하나 바뀌가면서  
제출해보기 때문!!

# What is Kaggle Competition?

## Rules

### One account per participant

You cannot sign up to Kaggle from multiple accounts and therefore you cannot submit from multiple accounts.

### No private sharing outside teams

Privately sharing code or data outside of teams is not permitted. It's okay to share code if made available to all participants on the forums.

### Team Mergers

Team mergers are allowed and can be performed by the team leader. In order to merge, the combined team must have a total submission count less than or equal to the maximum allowed as of the merge date. The maximum allowed is the number of submissions per day multiplied by the number of days the competition has been running.

### Team Limits

There is no maximum team size.

### Submission Limits

You may submit a maximum of 5 entries per day.

You may select up to 2 final submissions for judging.

### Competition Timeline

Start Date: May 17, 2018

Merger Deadline: August 22, 2018

Entry Deadline: August 22, 2018

End Date: August 29, 2018 - 11:59 PM UTC

**Rule**을 잘 지켜야합니다.

하루 제출횟수가 정해져 있기 때문에 **ID 여러 개 만들어서 제출하시는 분들** 있는데, 다 찾아냅니다!

그래서 대회 끝나고 일주일정도 **Invalid User** 잡아내는데 수백팀이 삭제되기도 합니다.


그리고 **Kernel** 혹은 **Discussion**을 제외한 외부팀에게 자신이 한 것을 알리면 안됩니다.

# What is Kaggle Competition?


초보자가 하기 좋은 대회

대회 입상보다는 경험을 중심으로, 배울 것이 많은 대회에 참가하자!

많은 사람들이 참여하고 있고 상위권 랭크에 1~2번 제출로 순위가 바뀌지 않는 대회



**Home Credit Default Risk**  
Can you predict how capable each applicant is of repaying a loan?  
Featured · 24 days to go · home, banking, tabular data



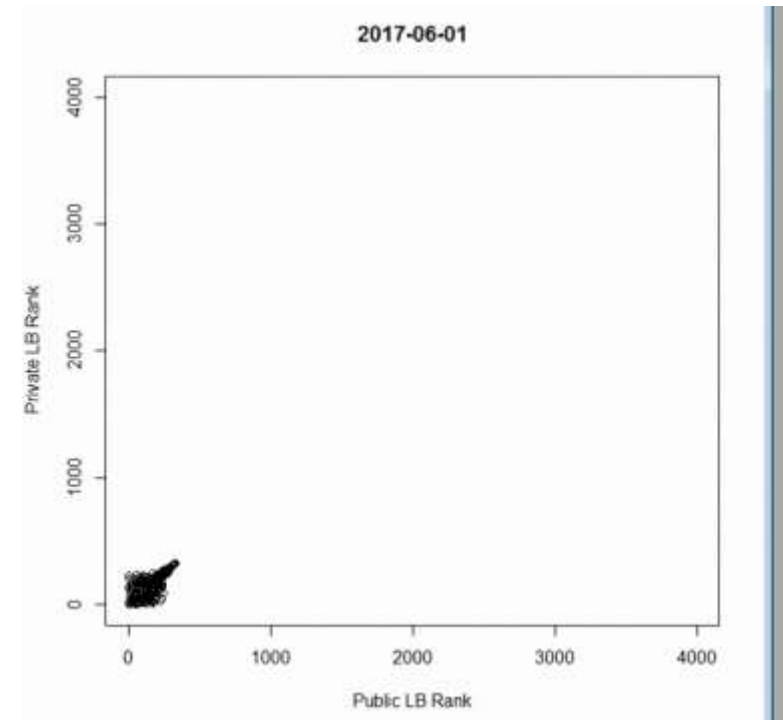
**Zillow Prize: Zillow's Home Value Prediction (Zestimate)**  
Can you improve the algorithm that changed the world of real estate?  
Featured · 7 months ago · housing, real estate

이런 대회 좋지 않아요!



**Mercedes-Benz Greener Manufacturing**  
Can you cut the time a Mercedes-Benz spends on the test bench?  
Daimler · 3,835 teams · a year ago

**\$25,000**  
Prize Money



# Feature Engineering Part 1

## Feature Engineering이란?

Feature Engineering 은 데이터의 [도메인 지식](#) 을 사용하여 [기계 학습](#) 알고리즘을 작동 시키는 [기능](#) 을 만드는 프로세스입니다 . [<wiki - Feature Engineering>](#)

즉, 모델의 성능을 높이기 위해서 주어진 데이터를 가공하고 기능을 추가, 삭제, 변경하는 것

Feature Engineering을 힘들고 고된 작업  
하지만 매우 중요한 작업

대부분의 우승자들도 이 부분에 많은 시간을 사용

기능을 생각해내는 것은 어렵고 시간이 오래 걸리기 때문에 전문 지식이 필요합니다. "응용 기계 학습"은 기본적으로 피쳐 엔지니어링입니다.

- [Andrew Ng](#) , [Brain 시뮬레이션을 통한 기계 학습 및 AI](#) <sup>[1]</sup>

# Feature Engineering Part 1

## Feature Type

- **Numerical Feature**

- 양적 자료(**quantitative data**, 정량적 자료)는 수치로 측정이 가능한 자료이다. 또는 수치적 자료(**Numerical data**)라고 하기도 합니다.
- 예: 온도, 지능지수, 절대온도, 가격, 주가지수, 실업률, 매출액, 기업내 과장의 수 등.

- **Categorical and Ordinal Feature**

- 질적 자료(**qualitative data**, 정성적 자료)는 수치로 측정이 불가능한 자료이다. 분류 자료 또는 범주형 자료(**categorical data**)라고도 한다.
- 예: 전화번호, 등번호, 성별, 혈액형, 계급, 순위, 등급, 종교 분류 등.

- **Datetime and Coordinate**

- 기타 등등 (**String..**)

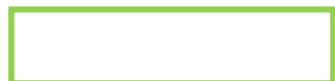
# Feature Engineering Part 1

## Feature Type

Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
Rattata	Normal		253	30	56	35	25	35	72	1	FALSE
Raticate	Normal		413	55	81	60	50	70	97	1	FALSE
Spearow	Normal	Flying	262	40	60	30	31	31	70	1	FALSE
Fearow	Normal	Flying	442	65	90	65	61	61	100	1	FALSE
Ekans	Poison		288	35	60	44	40	54	55	1	FALSE
Arbok	Poison		438	60	85	69	65	79	80	1	FALSE
Pikachu	Electric		320	35	55	40	50	50	90	1	FALSE
Raichu	Electric		485	60	90	55	90	80	110	1	FALSE
Sandshrew	Ground		300	50	75	85	20	30	40	1	FALSE
Sandslash	Ground		450	75	100	110	45	55	65	1	FALSE
Nidoran?	Poison		275	55	47	52	40	40	41	1	FALSE



Numerical Feature



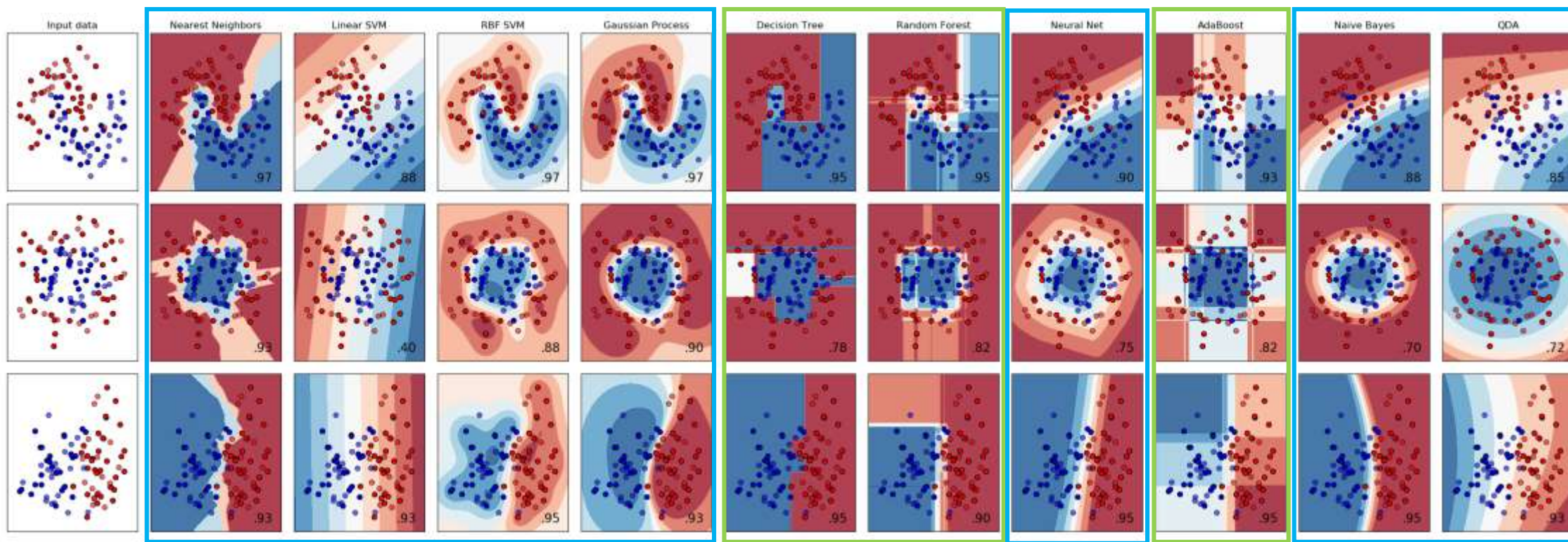
Categorical Feature



# Feature Engineering Part 1

## Numerical Feature

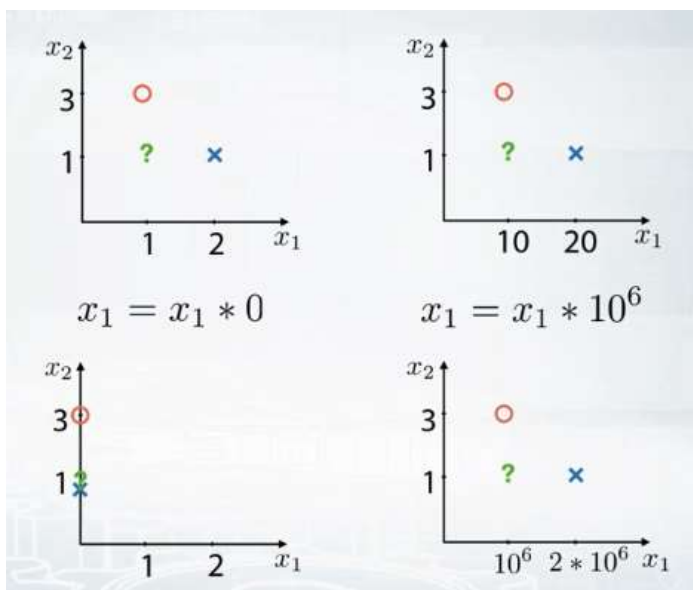
이제부터 Feature를 다룰 때 **Tree Model**과 **Non-Tree Model**로 나뉘서 생각할 것입니다.





# Feature Engineering Part 1

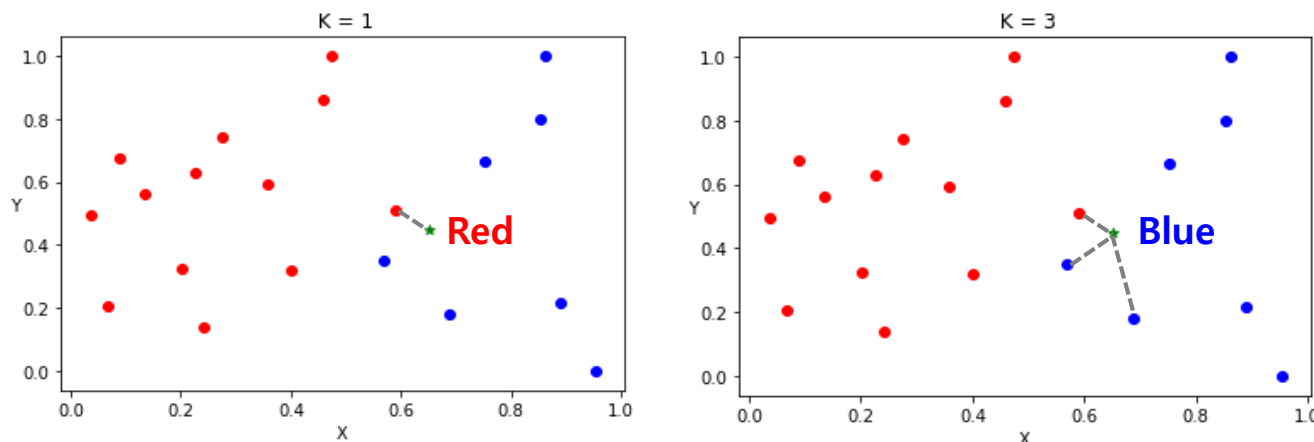
## Numerical Feature



KNN 같은 경우,  
왼쪽 그림에서는 **x**로 분류되나,  
**Scaling**이 다른 오른쪽 그림에서는 **o**로 분류된다.


**Scaling**에 따라서 결과값이 바뀝니다.

## KNN



# Feature Engineering Part 1

## Numerical Feature

$$\min_{\beta} \|Y - X\beta\|_2^2 + \lambda R(\beta)$$


$$Z = XD$$

$$\|Y - Z\theta\|_2^2 + \lambda R(\theta) = \|Y - XD\theta\|_2^2 + \lambda R(\theta)$$

Now set  $\beta = D\theta$

$$\begin{aligned} \|Y - XD\theta\|_2^2 + \lambda R(\theta) &= \|Y - X\beta\|_2^2 + \lambda R(D^{-1}\beta) \\ &\neq \|Y - X\beta\|_2^2 + \lambda R(\beta) \end{aligned}$$

또한 선형모델에서도 **Regularization**를 사용할 때 **Feature**간 **Scaling**이 다르면 선형모델의 계수들에 영향을 주어서 성능이 달라집니다.

**D**가 10000이라고 하고,  
기존 **X**값에 10000을 곱한 **z**값이 있다고 해볼게요.

정규화식을 안하면 **X**에 **D**를 곱하더라도 **Coefficeint**가 작아지면 되니깐 괜찮은데, 정규화를 하면 같은 **Coefficeint**를 두가지 식에서 공유하기 때문에 문제가 생깁니다.

# Feature Engineering Part 1

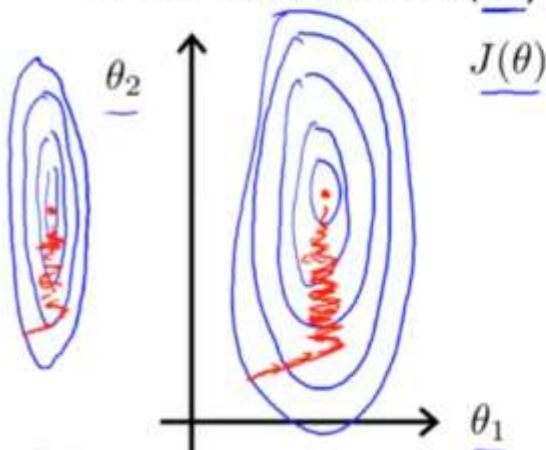
## Numerical Feature

### Feature Scaling

Idea: Make sure features are on a similar scale.

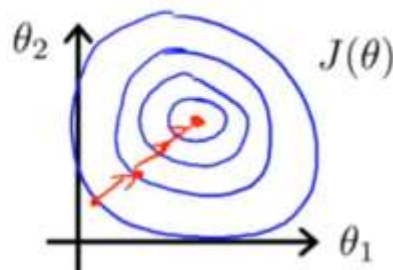
E.g.  $x_1 = \text{size (0-2000 feet}^2\text{)}$  ←

$x_2 = \text{number of bedrooms (1-5)}$  ←



$$\rightarrow x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$\rightarrow x_2 = \frac{\text{number of bedrooms}}{5}$$



**Gradient Descent Algorithm**을 사용하는 모델들이 빠르게 최적화 지점을 찾을 수 있게 합니다.

# Feature Engineering Part 1

## Numerical Feature

Algorithm	Problem Type	Results interpretable by	Easy to explain algorithm	Average predictive accuracy	Training speed	Prediction speed	Amount of parameter tuning	Performs well with small	Handles lots of irrelevant	Automatically learns feature	Gives calibrated probabilities?	Parameter c?	Features might need
KNN	Either	Yes	Yes	Lower	Fast	Depends on n	Minimal	No	No	No	Yes	No	Yes
Linear regression	Regression	Yes	Yes	Lower	Fast	Fast	None (excluding regularization)	Yes	No	No	N/A	Yes	No (unless regularized)
Logistic regression	Classification	Somewhat	Somewhat	Lower	Fast	Fast	None (excluding regularization)	Yes	No	No	Yes	Yes	No (unless regularized)
Naive Bayes	Classification	Somewhat	Somewhat	Lower	Fast (excluding feature extraction)	Fast	Some for feature extraction	Yes	Yes	No	No	Yes	No
Decision trees	Either	Somewhat	Somewhat	Lower	Fast	Fast	Some	No	No	Yes	Possibly	No	No
Random Forests	Either	A little	No	Higher	Slow	Moderate	Some	No	Yes (unless noise ratio is very high)	Yes	Possibly	No	No
AdaBoost	Either	A little	No	Higher	Slow	Fast	Some	No	Yes	Yes	Possibly	No	No
Neural networks	Either	No	No	Higher	Slow	Fast	Lots	No	Yes	Yes	Possibly	No	Yes

# Feature Engineering Part 1

## **Numerical Feature**

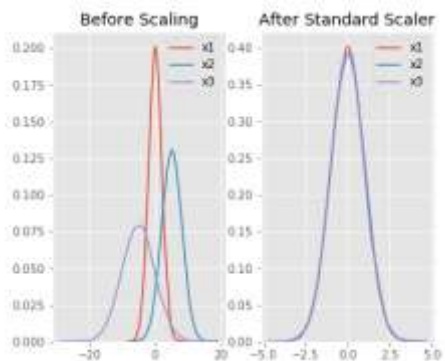
- **StandardScaler**
- **MinMaxScaler**
- **Winsorization**
- **Rank Transform**
- **Log Transform**
- **Raising to the Power**

# Feature Engineering Part 1

## Numerical Feature

### StandardScaler

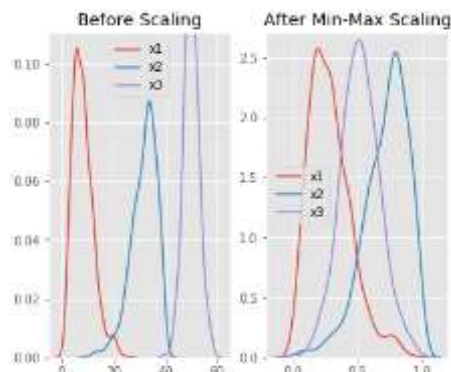
$$\frac{x_i - \text{mean}(x)}{\text{stdev}(x)}$$



```
scale=StandardScaler()  
scale.fit(x_train)  
x_train=scale.transform(x_train)  
x_valid=scale.transform(x_valid)
```

### MinMaxScaler

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$



뭘 써야할지 정확한 답은 없는 것 같습니다.

MinMaxScaler는 Outlier에 취약하기 때문에 Outlier가 있는 데이터에서 좋지 않습니다.

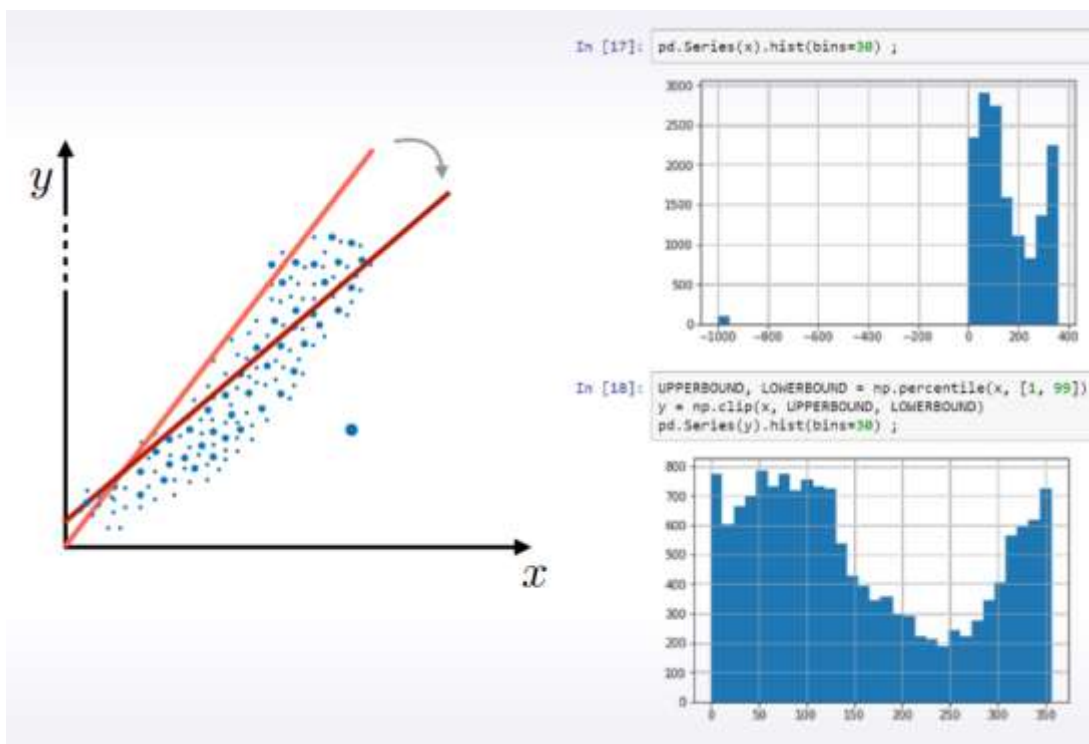
다만 Range가 정확히 있어야하는 변환(RGB[0~255])에서는 MinMaxScaler를 사용해야 합니다.

PCA 할 때는 StandardScaler

Tree Model에서 성능은 거의 같습니다.

# Feature Engineering Part 1

## Numerical Feature



**Outlier** 이야기를 잠깐하면,

심한 **Outlier**는 성능에 큰 영향을 줄 수 있습니다.

그래서 **Outlier**를 제거 하거나 **Winsorizing** 할수도 있고 **Outlier**에 영향을 덜 받는 **Scaling**을 수행할 수 있습니다.

## Winsorizing

```
[ -40  -5  10  13  15  19  26  28  41  58  78  85  86  89  
 89  91  92 101 101 103]  
np.percentile(a, 15): 12.55  
np.percentile(a, 85): 93.35  
scipy.stats.mstats.winsorize(a, limits=0.15): [13 13 13 13 15 19 26 28 41 58 78 85 86 89 89 91 92 92 92 92]
```

Data를 삭제하는 것이 아니라 변경

# Feature Engineering Part 1

## Numerical Feature

- `rank([-100, 0, 1e5]) == [0, 1, 2]`
- `rank([1000, 1, 10]) = [2, 0, 1]`

`scipy.stats.rankdata`

## Rank Transform

Outlier가 있다면 MinMaxScaler보다 좋은 방법  
일 수 있습니다.

Outlier를 하나하나 다룰 수 없을 때 Linear  
Model과 NN에서 좋은 성능을 낼 수 있습니다.

변환할때는 Train/Test 모두 합쳐서 변환해야 합  
니다.



# Feature Engineering Part 1

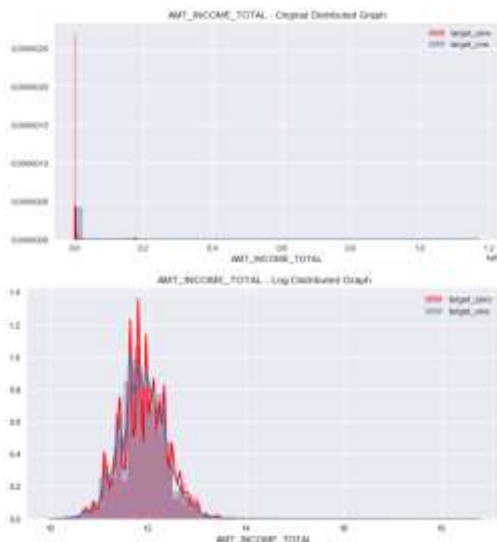
## Numerical Feature

1. Log transform:

$\text{np.log}(1 + x)$

2. Raising to the power  $< 1$ :

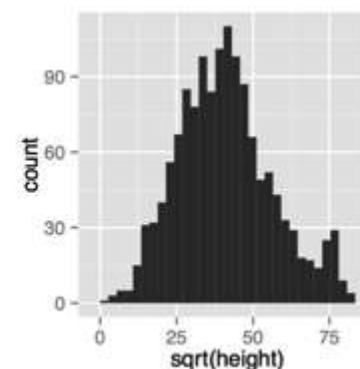
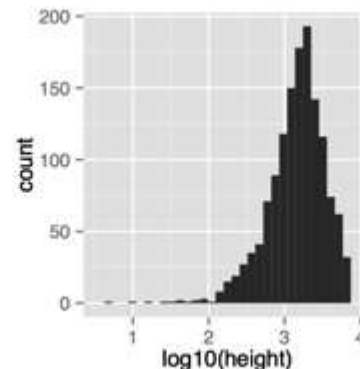
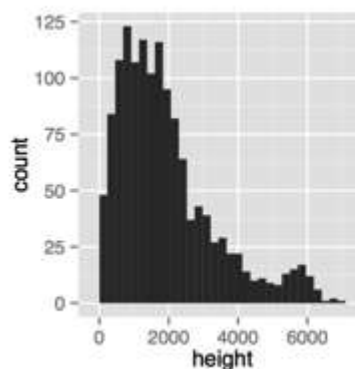
$\text{np.sqrt}(x + 2/3)$



**Log** 나 **power transform** 같은 경우 꽤 자주 사용됩니다.

값이 너무 큰 경우 **Log** 변환을 하게 되면 작은 값으로 변경됩니다.

**Log Graph** 특성으로 0 근처에 있는 값을 더 크게, 큰 값을 작게하여 데이터가 정규성을 띄게 해줍니다.



# Feature Engineering Part 1

## Numerical Feature

학습할 때 하나의 **Scaling**만 사용하는 것이 아닌 데이터를 각각 다르게 **Scaling**하여 **Feature**에 중복으로 추가하여 사용하거나 모델을 만들 때 각각 다른 스케일로 변환된 데이터로 학습하면 좋을 수 있습니다.



plantsgo **Topic Author** • (1st in this Competition) • a year ago • Options • Reply

8

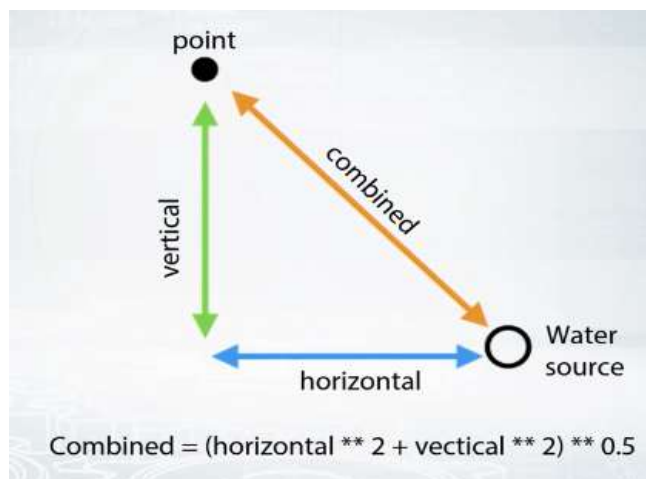
```
clf = Sequential()
clf.add(Dense(64, input_dim=tr_x.shape[1], activation="relu",
W_regularizer=l2()))
clf.add(Dense(64, activation="relu", W_regularizer=l2()))
clf.add(Dense(class_num, activation="softmax"))
```

The features are same as xgboost, actually all of my models used the same features. I think the improve comes from the Transform. When I only use Standardscaler, nn score: 0.54+ in cv. Then I use log10(X+1) before Standardscaler, nn score: 0.532 in cv.

# Feature Engineering Part 1

## Numerical Feature

Feature를 만드는 것은 쉬운 것부터 만듭니다.



price	fractional_part
0.99	0.99
2.49	0.49
1.0	0.0
9.99	0.99

# Feature Engineering Part 1

## Numerical Feature

2<sup>nd</sup> order Feature 추가

Feature 끼리 곱하고, 더하고, 빼고 나누고

f1	f2		f_join
1.2	0.0		0.0
3.4	0.1	Mul →	0.34
5.6	1.0		5.6
7.8	-1.0		-7.8

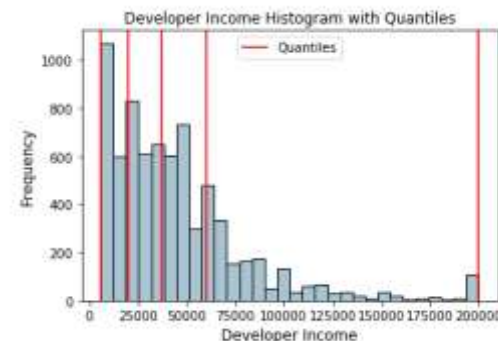
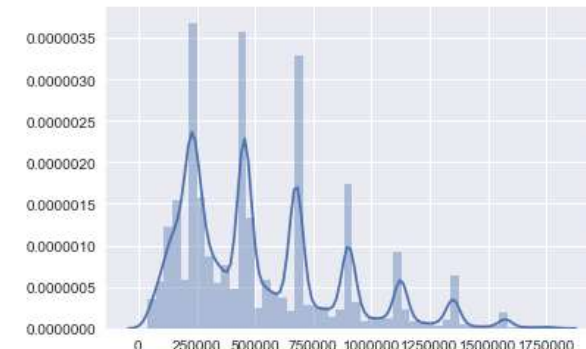
# Feature Engineering Part 1

## Numerical Feature

### Binning

Numerical Value의 분포가 일부 데이터는 비정상적으로 크고 일부는 매우 작은 왜곡된 형태  
Numerical Value가 세밀하고 넓게 분포된 상황에서 Overfitting이 될 수 있는 것을 방지  
Numerical Feature를 Category Feature로 변경하여 다른 Feature와 상호작용 가능

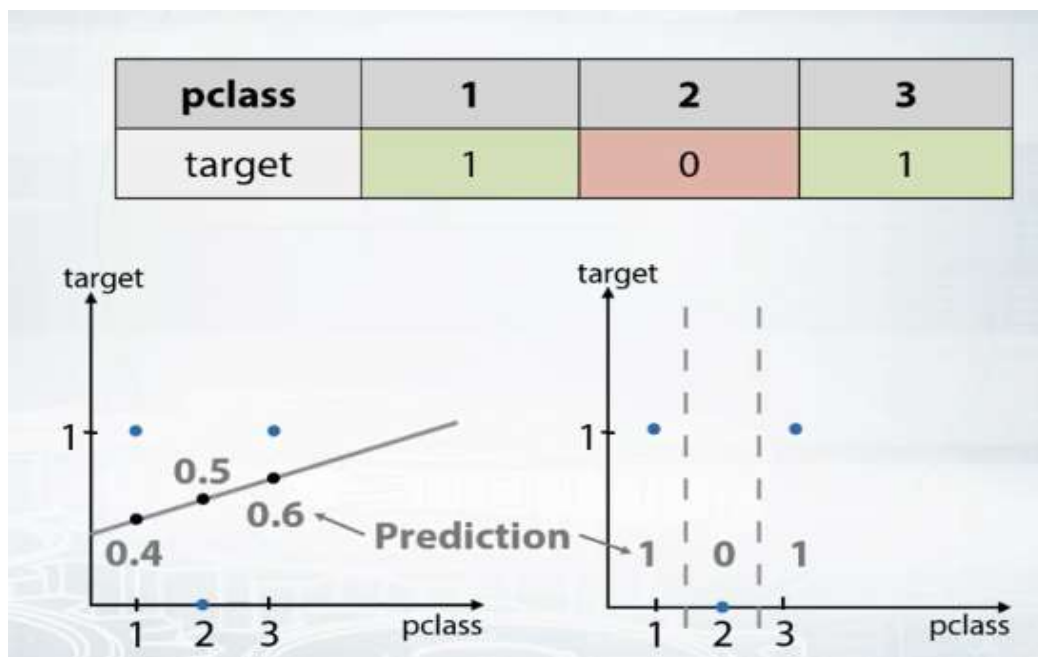
- Fixed Binning
  - 고정된 값으로 Bin을 구성하여 Numerical Feature를 나눔, Bin안에 Data가 없거나 매우 소수의 데이터만 포함될 수 있음
- Adaptive Binning
  - Quantile 같은 Data의 크기를 보고 나누는 방법



# Feature Engineering Part 1

## Categorical Feature

Non Tree Model과 선형모델은 pclass를 바탕으로 Target 값을 판단할 때도 다르게 판단할 수 있습니다. Categorical Feature 역시 모델별로 다르게 다뤄야 합니다.



# Feature Engineering Part 1

## Categorical Feature

보통 알고리즘들은 **String**으로 되어있는 **Category Feature**를 그대로 사용할 수 없기 때문에 변환을 해주고 이 전 **Slide**와 같은 이유로 **Encoding**을 수행해야 합니다.

- **Label Encoding**
- **Frequency Encoding**
- **One-Hot Encoding**

# Feature Engineering Part 1

## Categorical Feature

### Label Encoding

**Category Data**를 숫자를 기반으로 한 **Category**로 변환  
실용적(**string** < **int**), 특정 알고리즘 지원, 성능  
예) 콜라, 펩시, 사이다, 환타 -> 0,1,2,3

많은 방법들이 있겠지만 보통은 **sklearn**의 **LabelEncoder**를 사용하는 방법과, **pandas factorize**를 이용하는 방법이 있습니다.

### Factorize와 Label Encoder의 차이점

**Factorize**는 **NaN** 값을 자동으로 **-1**로 채우는데 반해 **Label Encoder**는 **Error** 발생

**Sample** 마다 차이가 있겠지만 **Factorize**가 같은 **Data**를 변환하는데 약 2.4배 정도 빠름



# Feature Engineering Part 1

## Categorical Feature

### Label Encoding

#### Sort = True

```
cat84 ['C' 'A' 'D' 'B']
cat84 [2 0 3 1]
=====
cat85 ['B' 'A' 'C' 'D']
cat85 [1 0 2 3]
=====
cat86 ['D' 'B' 'C' 'A']
cat86 [3 1 2 0]
=====
cat87 ['B' 'C' 'D' 'A']
cat87 [1 2 3 0]
```

순서가 정렬되어 있지 않더라도 자동으로 알파벳 순서에 맞게 변환됨

#### Sort = False

```
cat84 ['C' 'A' 'D' 'B']
cat84 [0 1 2 3]
=====
cat85 ['B' 'A' 'C' 'D']
cat85 [0 1 2 3]
=====
cat86 ['D' 'B' 'C' 'A']
cat86 [0 1 2 3]
=====
cat87 ['B' 'C' 'D' 'A']
cat87 [0 1 2 3]
```

Unique 순서대로 변환 됨

즉 Data가 의미있는 순서로 정렬되어 있는 상태라면 factorize를 사용해야 합니다.

LabelEncoder는 정렬후 Encoding을 수행

# Feature Engineering Part 1

## Categorical Feature

### Frequency Encoding

K	[S,C,Q] -> [0.5, 0.3, 0.2]
embarked	
S	encoding = titanic.groupby('Embarked').size()
C	encoding = encoding/len(titanic)
S	titanic['enc'] = titanic.Embarked.map(encoding)
S	
S	
Q	

값 분포에 대한 정보가 잘 보존

리니어, Tree 모델 모두 도움이 될 수 있습니다.

밸류의 빈도가 타겟과 연관이 있으면 아주 유용

하지만 Encoding했을 때 Encoding한 Category끼리 같은 Frequency를 가진다면 Feature로 추가하지 않아도 됩니다.

# Feature Engineering Part 1

## Categorical Feature

### One Hot Encoding

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

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

**Tree Model**에서는 느려질 수 있고 성능이 항상 올라가는 것은 아닙니다. 하지않아도 된다고 합니다.

거리기반 **Algorithm**, 선형, **NN** 모델에서는 **One Hot Encoding** 이 필요합니다.

컬럼이 많아져 차원이 많아지므로 학습시간이 오래 걸릴 수 있는 단점이 있습니다.

# Feature Engineering Part 1

## Datetime and Coordinate

### 1. Periodicity

Day number in week, month, season, year  
second, minute, hour.

### 2. Time since

- a. Row-independent moment  
For example: since 00:00:00 UTC, 1 January 1970;
- b. Row-dependent important moment  
Number of days left until next holidays/ time passed after last holiday.

### 3. Difference between dates

$\text{datetime\_feature\_1} - \text{datetime\_feature\_2}$

간단하게 Datetime Feature이 주어졌을 때  
각 항목을 나눠서 Feature에 추가

기준점으로부터 기간이 얼마나 지났는지  
TimeSeries 대회에서 많이 사용

Datetime끼리 빼서 Feature를 추가하기도 합니다.

Date	week day	daynumber_ since_year_2 014	is_holiday	days_till_ holidays	sales
01.01.14	5	0	True	0	1213
02.01.14	6	1	False	3	938
03.01.14	0	2	False	2	2448
04.01.14	1	3	False	1	1744
05.01.14	2	4	True	0	1732
06.01.14	3	5	False	9	1022

# Feature Engineering Part 1

## Datetime and Coordinate

### 1. Periodicity

Day number in week, month, season, year  
second, minute, hour.

### 2. Time since

- a. Row-independent moment  
For example: since 00:00:00 UTC, 1 January 1970;
- b. Row-dependent important moment  
Number of days left until next holidays/ time passed  
after last holiday.

### 3. Difference between dates

`datetime_feature_1 - datetime_feature_2`

간단하게 Datetime Feature이 주어졌을 때  
각 항목을 나눠서 Feature에 추가

기준점으로부터 기간이 얼마나 지났는지  
TimeSeries 대회에서 많이 사용

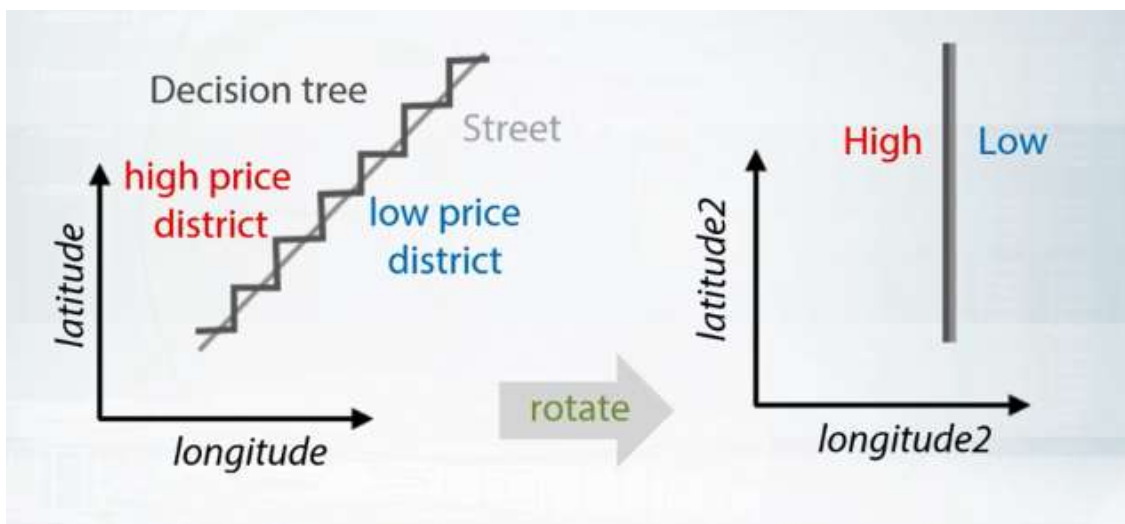
Datetime끼리 빼서 Feature를 추가하기도 합니다.

Date	week day	daynumber_ since_year_2 014	is_holiday	days_till_ holidays	sales
01.01.14	5	0	True	0	1213
02.01.14	6	1	False	3	938
03.01.14	0	2	False	2	2448
04.01.14	1	3	False	1	1744
05.01.14	2	4	True	0	1732
06.01.14	3	5	False	9	1022

user _id	registration _date	last_purchase_ date	last_call_d ate	date_diff	churn
14	10.02.2016	21.04.2016	26.04.2016	5	0
15	10.02.2016	03.06.2016	01.06.2016	-2	1
16	11.02.2016	11.01.2017	11.01.2017	1	1
20	12.02.2016	06.11.2016	08.02.2017	94	0

# Feature Engineering Part 1

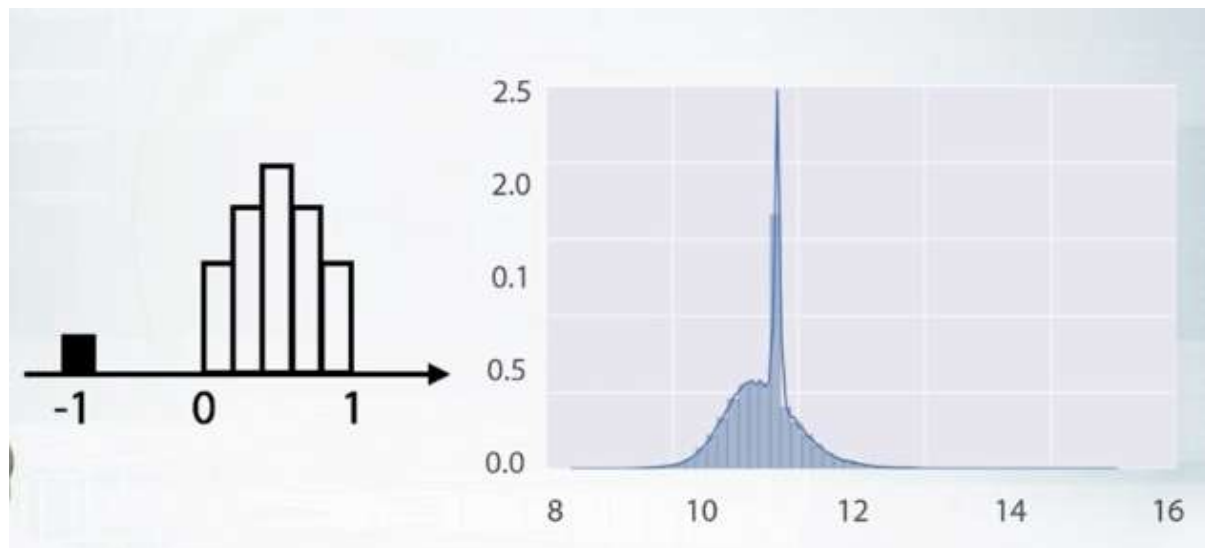
## Datetime and Coordinate



Coordinate는 위도 경도 Feature들을 말하는데 Tree Model 같은 경우 Numerical Feature들이 선형성을 띄면 구분하기 힘들기 때문에 회전을 주기도 합니다.

# Feature Engineering Part 1

## Missing Value



어떤 Competition에는 모든 Column에 Null이 없을 때도 있습니다.

하지만 변수 하나하나 Plot을 해보면 주최 측에서 Missing Value를 채워 넣었다는 것을 추측할 수 있습니다.

# Feature Engineering Part 1

## Missing Value

어떤 방법으로 처리하는 것이 제일 좋은지는 해봐야 알 것 같습니다...ㅠ

저 같은 경우 데이터 하나하나 보면서 처리합니다.

특히 컬럼명세나 Column 명을 통해서 빈 값을 유추할 수 있으면 그렇게 먼저해봅니다.  
최대한 조심스럽게 채우려고 노력합니다. (KNN을 사용하기도 합니다.)

Row삭제나 Column 삭제는 웬만하면 하지 않습니다.

아무리 Column에 Missing Value가 많더라도 Missing Value가 아닌 값들을 Target 별로 보면서  
각 분포가 다르다면 삭제하지 않고 분포가 같아 구별할 수 없다면 삭제합니다.

여러가지 변수를 보고 채우는 것은

예를 들어 Building ID라는 값이 NULL이면, 같은 위도, 경도에 있는 BuildingID의 Mode로 채우거나 그렇게 합니다.



# Feature Engineering Part 1

## Missing Value

Tree Model

-999, -1

Linear Model, Neural Network

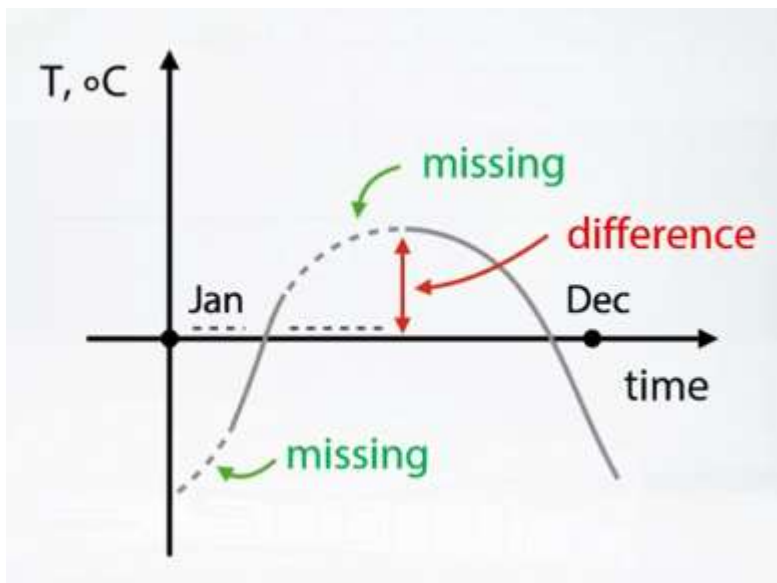
Mean, Median

feature	isnull
0.1	False
0.95	False
NaN	True
-3	False
NaN	True

IsNull feature 추가

# Feature Engineering Part 1

## Missing Value



그냥 단순히 Median으로 채우면 문제가 생길수도 있다는 것을 보여드리고 싶었습니다.

조심스럽게 다뤄주세요.

# Feature Engineering Part 1

## Data Cleaning

is_train	f0	f1	f2	f3	f4	f5
True	13	H	1.2	1.2	A	C
True	13	H	36.6	36.6	B	A
False	13	H	0	0	A	C
False	13	G	-14	-14	C	B

```
traintest.nunique(axis=1) == 1
```

f0 Column은 삭제

F1 Column은 삭제하거나 Test에 이 값이 많다면 f1을 target으로 하는 새로운 model을 만들어 feature를 추가 할 수도 있습니다. Binary Classification으로 G의 확률 값을 추가

F2, f3는 둘중 하나만 사용

F4, F5는 달라보이지만 LabelEncoding하면 같은 Feature 둘중 하나만 사용

# Feature Engineering Part 1

## Data Cleaning

	ID	ad_id	ad_count	ad_mean	ad_median	ad_std	ad_min	ad_max
369	4020	[4136, 3933, 4044, 4080, 4176, 4020]	6	110.595000	110.595	21.559686	95.35000	125.84000
16	134	[159, 233, 97, 157, 231, 134]	6	101.196667	95.570	15.315839	89.49000	118.53000
190	1561	[1617, 1737, 1556, 1666, 1700, 1733, 1561]	7	111.836667	114.260	13.222611	97.57000	123.68000
402	4404	[4426, 4318, 4508, 4536, 4587, 4626, 4633, 464...]	9	106.210000	106.210	11.440988	98.12000	114.30000
426	4689	[4711, 4992, 5066, 4830, 4860, 4689]	6	97.830000	93.385	11.060491	90.28000	114.27000
220	1951	[2000, 2147, 1780, 2056, 2093, 1951]	6	115.720000	111.890	11.016202	107.13000	128.14000

Benz 대회하면서 가장 시간을 많이 들인 부분이 모든 Feature가 같은데 Target Value가 다른 이유를 찾으려고 노력했고 결국 찾아내지 못했습니다.

데이터가 Train/Test 합쳐서 8000개 밖에 안되는데, 중복 Row가 1400개!!

보통은 이런 경우 index leak이 숨어있을 수도 있습니다!

# Feature Engineering Part 1 실습

# Feature Engineering Part 2

## EDA

### Price Comparison

Comparing [this listing](#) against median prices for [3BR / 1BA apartments in Gramercy Park with Doorman, Elevator](#).



The price of this apartment is **\$275** cheaper than the median price.

```
In [16]: def add_display_roomtype_price(df):
          data = df.copy()

          data['bedroom_cat'] = data['bedrooms'].apply(lambda x: 4 if x>=4 else x)
          data['bathroom_cat'] = data['bathrooms'].apply(lambda x: 5 if x>=5 else x)

          disp_roomtype_price_median = data.groupby(by=('display_address', 'fea_studio', 'fea_lo
          disp_roomtype_price_median.rename(columns={'price': 'disp_roomtype_price_median'}, i

          data = data.merge(disp_roomtype_price_median, on=('display_address', 'fea_studio', 'fea
          data['how_many_expansive'] = data['price'] - data['disp_roomtype_price_median']

          del data['bedroom_cat'], data['bathroom_cat']
          return data
```

대회 시작 초기에 도메인을 알고 가는 것도 좋은 방법

실제로 Two Sigma : Renthop 대회 시 회사 홈페이지를 보고 추가한 Feature

# Feature Engineering Part 2


## EDA

id	x1	x2	x3	x4	x5	x6
1	m268i97y	0	NO	105.4	14	
2	j0gheu6	1	YES	25.631	12	
3	26fmsp6u	1	NO	12.0	12	m268i97y
4	13e5dpzp	0	NO	140.12	14	m268i97y

데이터가 익명화 되어 있다면 Column의 의미를 파악해보거나 Feature들 간의 관계를 찾아보는 과정을 되풀이합니다.

# Feature Engineering Part 2

## EDA



PC Jimmy  
3207th place

### 50 Years in Quality - X10 to X385 - what they mean

posted in Mercedes-Benz Greener Manufacturing a year ago

47

X10 to X385 are tests. Each test addresses one or more features in the car configuration defined in X0 to X8 (the car).

A group of the test columns are supplementary to each other. For example, X205 is used on a very large number of the cars. The 4 cars not tested to the X205 procedure are likely tested to one of the columns with sum = 4 for the columns. But it can also be two different columns that have sum = 2 (for 4 ones, etc.) Additional groupings of tests fall in this same category. Two columns that are never 1 for the same car are good candidates for alternate versions of a similar test method.

A number of tests run concurrently and have differing test length times. Those with test times much faster than the y value should fall out as not significant with almost any method of examination.

Some tests must be executed after another test is completed.

The reported test time y is the longest critical path for the tests required to be performed. Most tests have some time variation. The large variation in y when duplicate cars are examined (the required tests for those duplicates are likewise duplicates) may be the result of a long critical path where the individual tests on the path have variation.

The variation in duplicates may also be the result of other features of the car not captured in X0-X8. You don't have to spend much time in the automotive business to know that 8 columns are not enough to describe all cars from a single assembly plant. The 8 columns are sufficient to tell the testing system what tests to run. A whole bunch more columns would be needed to have a complete car description. Those columns we don't know may have features that impact the test time. For example, the testing system may not need to know the tire size on the vehicle. But the tires might make a difference in the time it takes to run tests related to running the car at varying and high speeds.

We don't know if all the cars came out of the same assembly plant or the same test machine. My belief is that multiple assembly plants and/or test machines are in the dataset. Machine to machine variation might be part of the duplicate car.

또한 Kernel이나 Discussion을 보면 익명화된 Item을 자신의 경험을 바탕으로 설명해주는 사람도 있습니다.

50년 동안 자동차 품질쪽에 계셨던 분이 Feature에 대하여 추정해주셨습니다.



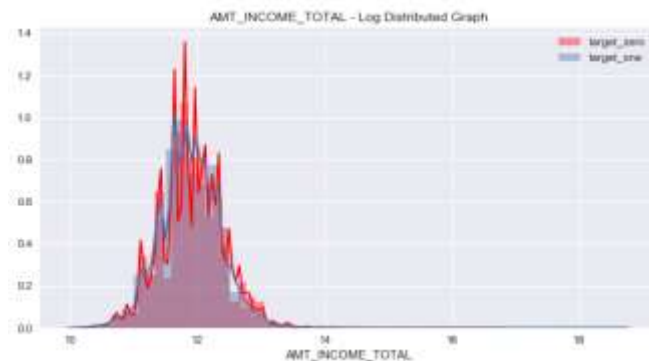
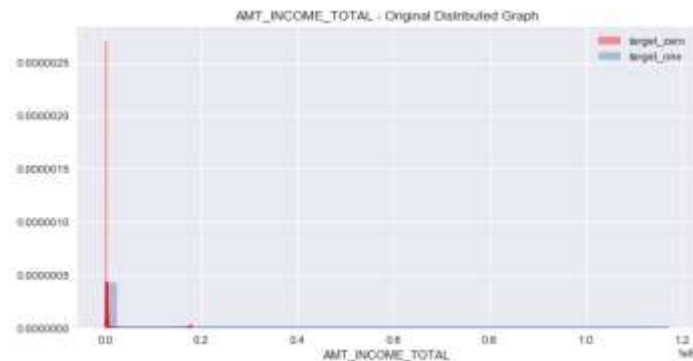
# Feature Engineering Part 2

## EDA

Column의 의미를 해석하지 못할 수도 있지만, 결국 중요한 것은 Column이 Category인지 Numeric인지 판단하는 것

계속 데이터를 들여다보면서 계속 깊게 들어가는 것

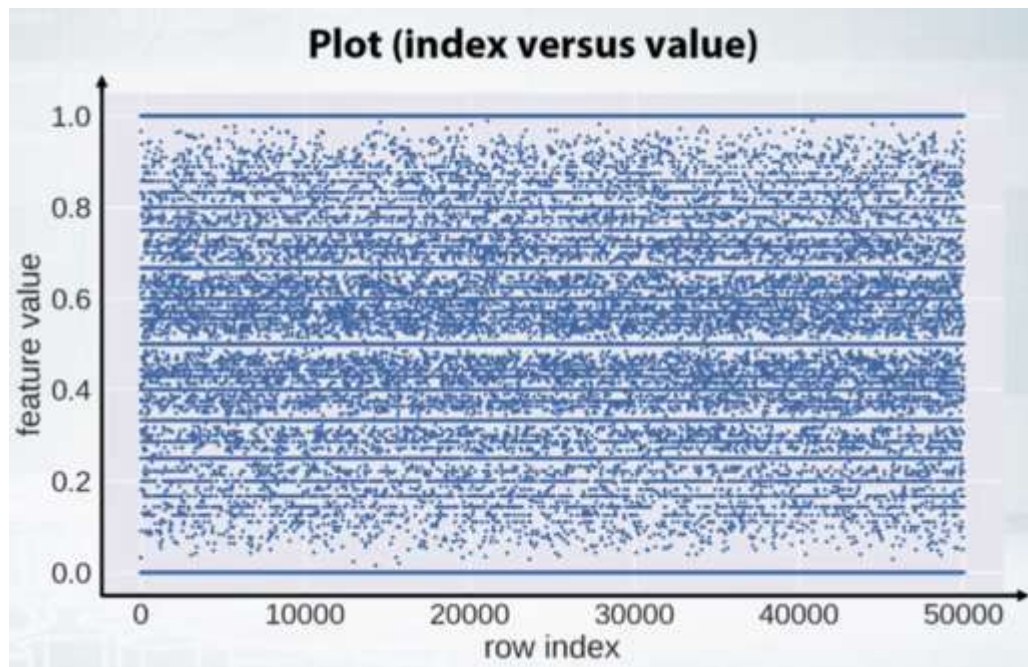
한가지 방법으로만 Visualization 하지 말고 여러가지 방법으로 해보는 것도 좋음



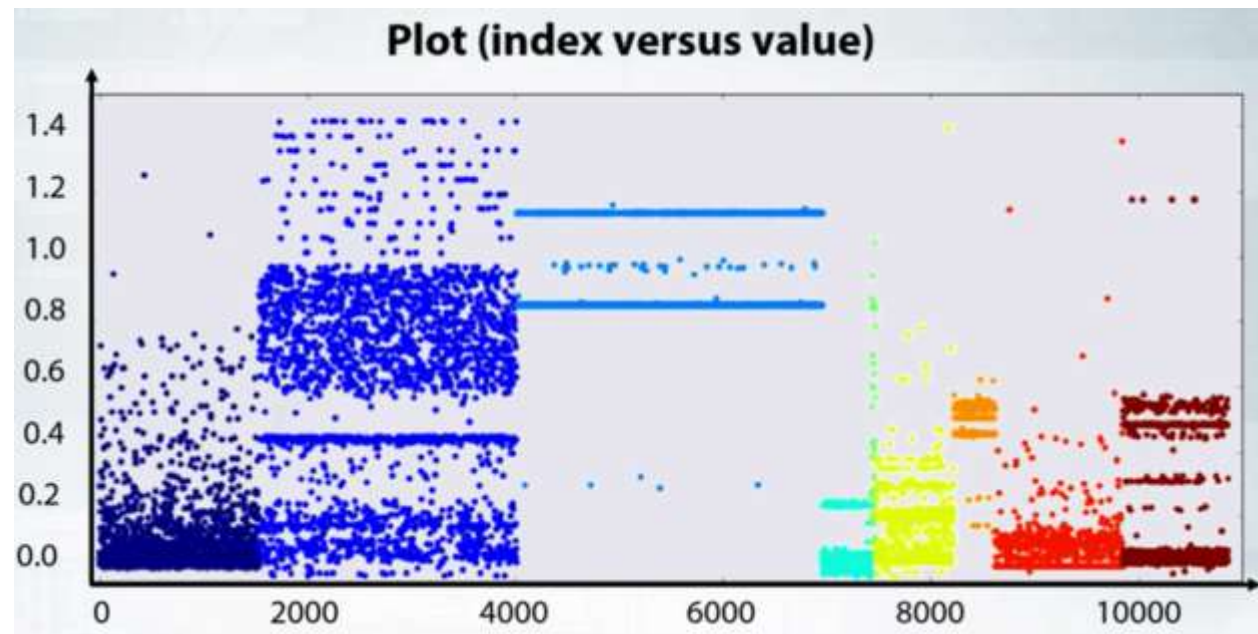
# Feature Engineering Part 2

## EDA

데이터가 적절하게 잘 섞임



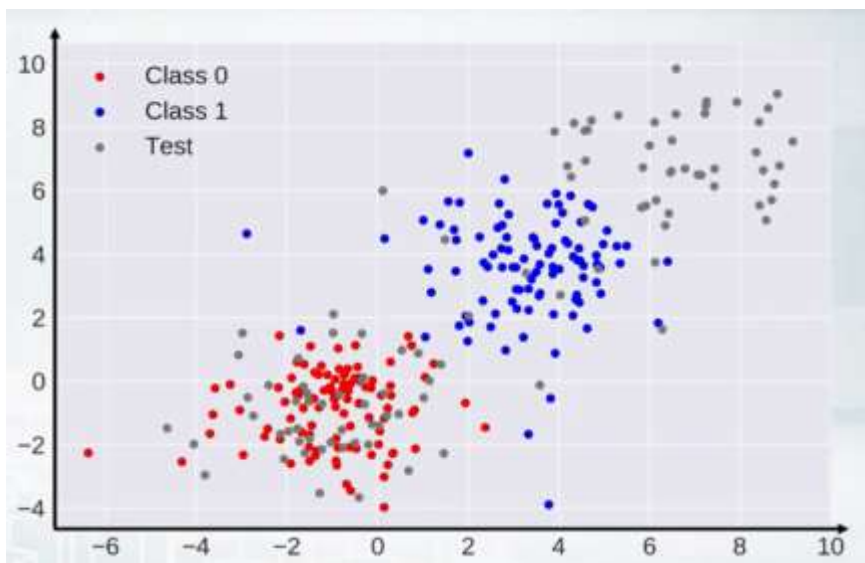
데이터가 적절하게 잘 섞이지 않음



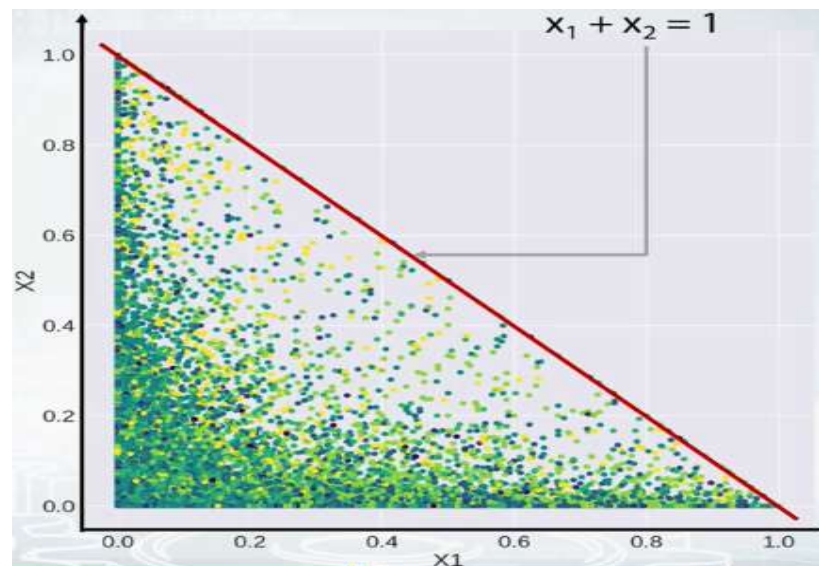
# Feature Engineering Part 2

## EDA

Test 데이터가 Class1과 분포가 다르다.

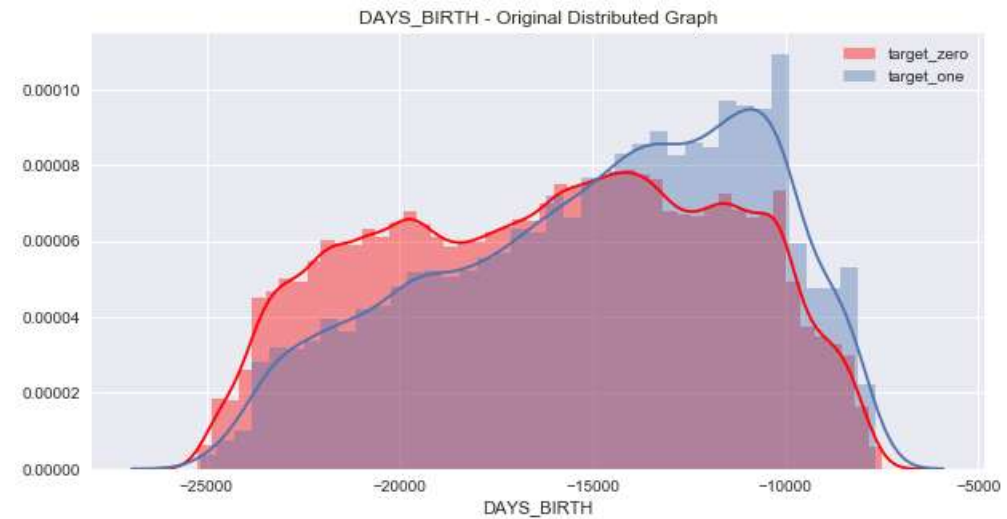
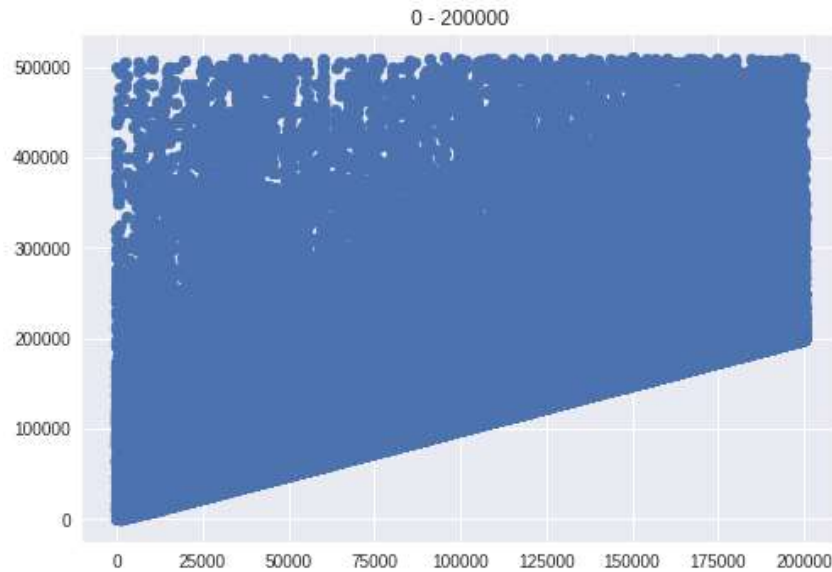


Tree 모델같은 경우 두 Feature간 차이나  $X_2/X_1$  같은 Feature를 추가하면 좋음



# Feature Engineering Part 2

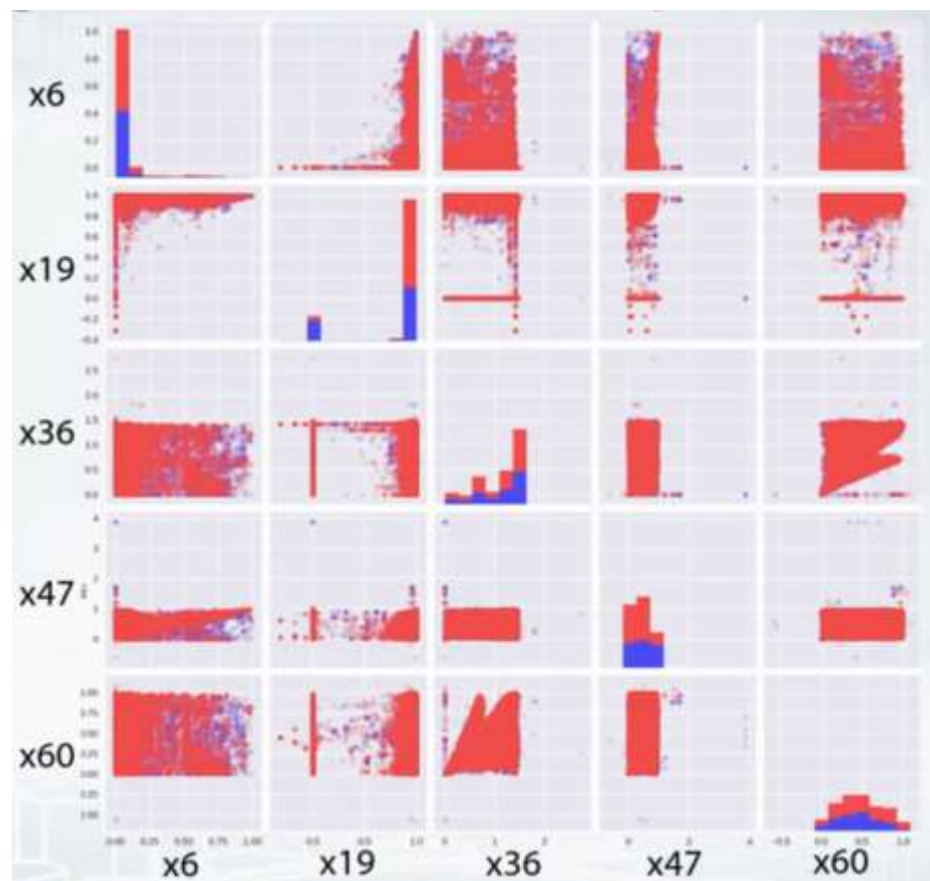
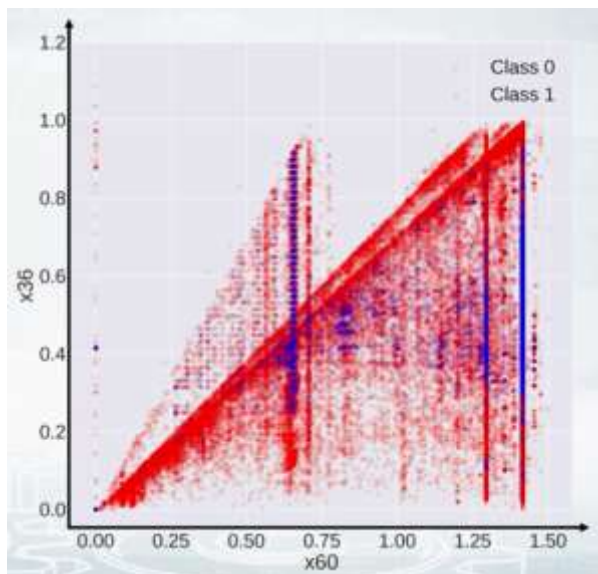
## EDA



# Feature Engineering Part 2

## EDA

삼각형이 보이는데 Class가 어떤 삼각형이 포함되는지 New Feature로 추가



계속 EDA, EDA, EDA!

그래프를 통하여 직관을 얻고 Feature간 상관관계를 확인

Custom Feature를 계속적으로 추가해나간다.



# Feature Engineering Part 2

## EDA

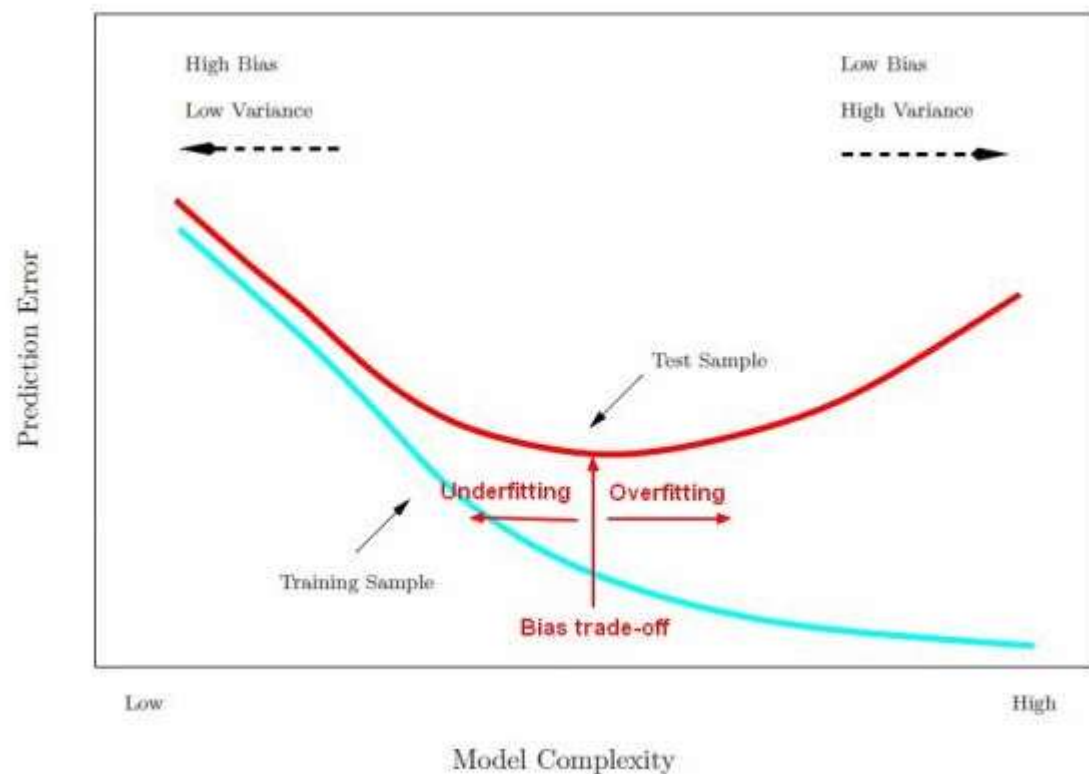
Data:

AdGroupId	AdNetworkType2	MaxCpc	Slot	Clicks	Impressions	is_incorrect
78db044136	s	0.28	s_2	3	0	True
68a0110c33	s	1	s_2	1	13	False
2r39fw11w3	p	1.2	p_1	3	419	False

이상한 데이터는 이렇게 Feature를 추가해주는  
것도 방법

# Feature Engineering Part 2

## Cross-Validation







Cross Validation은 자신의 모델을 평가하는 방법

잘 짜여진 Cross Validation 전략으로 우승한 Competition도 있을만큼 평가하는 방법을 적절하게 이해하고 구축하는 것은 매우 중요합니다.

# Feature Engineering Part 2

## Cross-Validation

HoldOut	
KFOLD	
	
	
	
LOOCV	
	
	









Data가 많거나 KFOLD를 했을 때 Validation Set Score 분포가 동일하게 나올때

시간이 다른 방법에 비해 오래 걸리지 않음



# Feature Engineering Part 2

## Cross-Validation

HoldOut	
KFOLD	
	
	
	
LOOCV	
	
	

KFOLD는 K값 즉 몇 번으로 나눌건지 정하고 K번만큼 Validation을 수행









각 Validation Set이 겹치지 않게 검증하게 되고 결과는 Average하여 성능을 평가

시간이 오래걸림

보통 5회 사용, 많으면 10정도

# Feature Engineering Part 2

## Cross-Validation

HoldOut	
KFOLD	
	
	
	
LOOCV	
	
	

Validation을 train set의 개수만큼 수행

Train데이터 중 하나만남기고 모두 Train하고 남은 하나로 Validation을 수행

시간이 매우 오래걸리기 때문에 데이터가 매우 작지않은 이상 잘 사용되지 않음

# Feature Engineering Part 2

## Cross-Validation

### Stratification

Samples and their target values

0	1	0	0	1	1	1	0
0	1	0	0	1	1	1	0
0.5		0		1		0.5	
0	1	0	0	1	1	1	0
0.5	0.5	0.5	0.5	0.5	0.5		

Stratification is useful for:

- Small datasets
- Unbalanced datasets
- Multiclass classification

# Feature Engineering Part 2

## Cross-Validation

Competition에서 가장 중요한 것 중에 하나는 Cross Validation Score가 Public Leaderboard 점수와 동일하게 따라가야 되는 것입니다.

즉, Cross Validation 점수가 오르면 LB 점수도 올라가야 합니다. 이 상황을 Cross Validation Score가 Stable해졌다고 말합니다.

Cross Validation Score가 LB와 동일하게 나오지 않는 이유

- Data Set이 너무 작다.
- Cross Validation Set을 적절하게 구성하지 못하였다.
- Overfitting 되었다.
- Cross Validation과 Test Set의 Target값 분포가 다르다.

# Feature Engineering Part 2

## Cross-Validation

Cross Validation Set을 적절하게 구성하지 못하였다.

Moving window validation

week1	week2	week3	week4	week5	week6
train			validation		
train				validation	
train					validation

# Feature Engineering Part 2

## Cross-Validation

Cross Validation과 Test Set의 Target값 분포가 다르다.

아래 같은 경우 LB의 Target 비율을 예측할 수 있습니다. 그리하여 실제 Train Data의 Target 분포와 Test 의 Target 분포가 다를 경우 Upsampling, Downsampling으로 Train하고 Validation의 비율이라도 Test와 동일하게 사용합니다.

Binary Classification, Logloss Metric  $-(y \log(p) + (1 - y) \log(1 - p))$

10000 Rows가 Test Set에 있습니다.  
모든 예측 값을 0.3으로 하였을 때 Public Score가 1.01이 나왔습니다.

$$10000 = a + b$$

$$1.01 = -1/10000((a)*(-1.2) + (10000-a)*-0.37)$$

$$10100 = -(-1.2a + (-3700+0.37a))$$

$$6400 = 0.83a$$

$$7711 = a$$

$$2289 = b$$

# Feature Engineering Part 2

## Cross-Validation 정리

- 데이터셋이 작으면 CV를 믿어야합니다.
- 제출 시에는 보통 2개를 선택할 수 있는데 LB 높은 것, CV 높은 것 이런 전략도 좋습니다.
- KFOLD는 보통 5, 5도 충분
- 만약 Kfold CrossValidation Score가 각 Iteration마다 편차가 크다면 Iteration별로 Target값 분포가 비슷한지 확인합니다. 그리고 각 Iteration 별 차이점이 무엇인지 확인합니다. 각 Iteration 별 CV Score의 편차를 확인하여 LB가 예측가능한 숫자인지 확인합니다.
- CV 전략과 관련하여 Discussion에 많이 올라오니 참고가능합니다.
- CV 시 Randomseed는 동일하게 해야 Feature 추가 후 성능이 개선되었는지 파악가능합니다.

# Feature Engineering Part 2

## Cross-Validation 정리

CV전략은 주최측의 데이터 분할 전략에 따라 달라지고

즉, 단순히 KFold나 Holdout을 수행하여 점수를 확인하는 것이 아니라,

Validation Set과 Test Set을 동일하게 하여, CV Score가 LB 점수와 비슷한 양상을 가지도록 하는 것이 핵심입니다.



# Feature Engineering Part 2

## Data Leak

주최측이 의도하지 않은 데이터에서 발견되는 점수를 크게 향상시킬 수 있는 방법

보통 발견하면 Discussion에 올리는 것이 암묵적인 룰인 것으로 보이나, 안올리는 사람도 있습니다. 왜냐하면 마지막에 Leak을 공개하면 대회 자체를 흔들어 버릴 수 있습니다.

### Tip

만약 갑자기 등수가 급락한다면 Discussion이나 Kernel을 확인합니다.

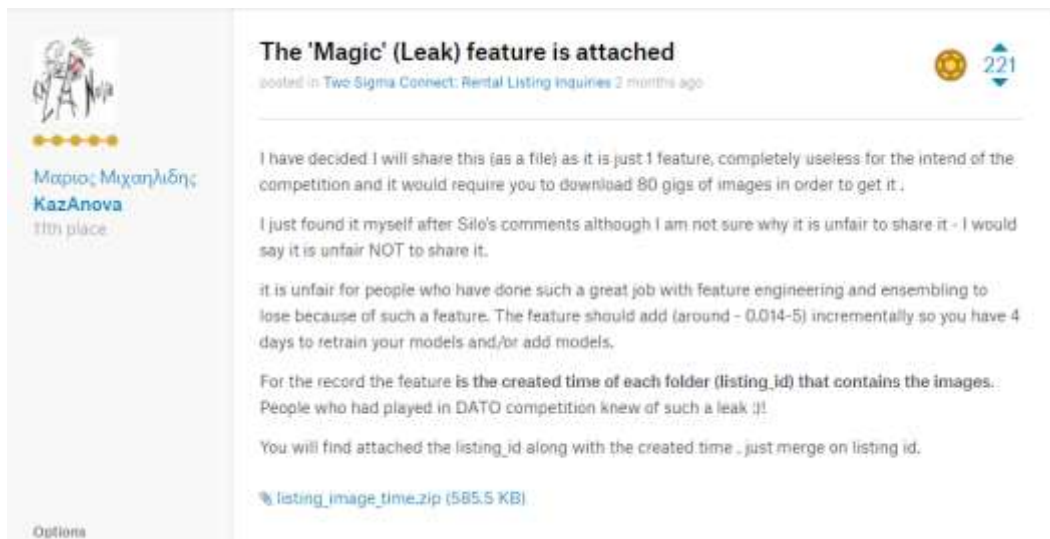
새로운 Data Leak을 누군가 공개했거나,

남들이 시도하지 않는 새로운 방법의 모델이 Kernel에 올라와 Ensemble하여 모델이 개선되었을 수도 있습니다.

# Feature Engineering Part 2

## Data Leak

### Two Sigma Data Leak



### Image 폴더의 생성시간

보통사람들이 데이터를 수집할 때,  
그룹화지어서 **Data**를 수집하기 때문에 이러한 문제가 발생할 수 있습니다.

**EX)**

양품 데이터 먼저 수집하고, 불량 데이터를 나중에  
수집하게 되었을때

```
In [14]: def image_timestamp(df):
          data = df.copy()

          leakage_image = pd.read_csv('listing_image_time.csv')
          data = data.merge(leakage_image, on='listing_id', how='left')

          del leakage_image
          return data
```

# Feature Engineering Part 2

## Data Leak


### Quora Question Pair

<https://www.kaggle.com/sudalairajkumar/simple-leaky-exploration-notebook-quora>

**Quora Question Pair**에서 질문이 중복되는 경우가 많아서 이를 이용하여 **Network Feature**를 추가한 예

# Feature Engineering Part 2

## Mean Encoding



Faron  
2nd place

### 2nd Place Solution

posted in Two Sigma Connect: Rental Listing Inquiries 1 year ago

94

#### Feature Engineering

Feature engineering was by far the most time consuming part for me and I created a lot of statistics either based on the entire dataset or based on subgroups like for example (manager\_id, bathrooms, bedrooms, dayofyear). The most common statistics are mean prices, differences from the mean prices, price ratios, counts of unique manager\_id's, total amount of listings and in general differences from mean or median feature values for the given group. I also created some timeseries features like target leads, lags and cumsums within groups, creation time deltas, rolling target means etc.

I initially thought, that they would help to better predict the groups of duplicated or very similar listings, but it did not work out. In general, the duplicated listings were much harder to predict and I spent quite some time to find something to close the loss-gap in comparison to the unique listings. Especially the groups of listings which differ only in created time and where all the 3 labels occurred were troublesome. My best single model scores 0.47400 on unique listings and 0.58954 on duplicates (which accounted for roughly 20% of the dataset).



Little Boat  
2nd place

### 三个臭皮匠's approach(s)

posted in Porto Seguro's Safe Driver Prediction 8 months ago

165

It definitely came as a surprise to me that there was this huge shake up in the end. I personally thought that the shake up should be relatively small because my stacking model performance on public LB and local cv was always aligned although for single model they could vary quite a bit. We did trust mostly on local CV and we tried very hard on nonlinear stacking (without success) so we relied on weighted average cv score and only submitted when there was an improvement on local cv.

4) feature aggregation: pick two features (e.g. ps\_car\_13, ps\_ind\_03), and then use one as group variable, the other as value variable, do mean, std, max, min, median. Still top important features are picked only

이 밖에도 많은 대회 우승자 솔루션에서 흔히 볼 수 있는 Mean Encoding

Kernel을 보면 min, max, std, mean 같은 Feature들을 추가하는 것을 볼 수 있습니다.

# Feature Engineering Part 2

## Mean Encoding

	feature	feature_label	feature_mean	target
0	Moscow	1	0.4	0
1	Moscow	1	0.4	1
2	Moscow	1	0.4	1
3	Moscow	1	0.4	0
4	Moscow	1	0.4	0
5	Tver	2	0.8	1
6	Tver	2	0.8	1
7	Tver	2	0.8	1
8	Tver	2	0.8	0
9	Klin	0	0.0	0
10	Klin	0	0.0	0
11	Tver	2	0.8	1

Mean Encoding은 Categorical Feature를 기준으로 하여 Target값과 통계적인 연산을 하는 것

Mean, Median, Std, Min, Max 등의 Feature를 추가해줍니다.

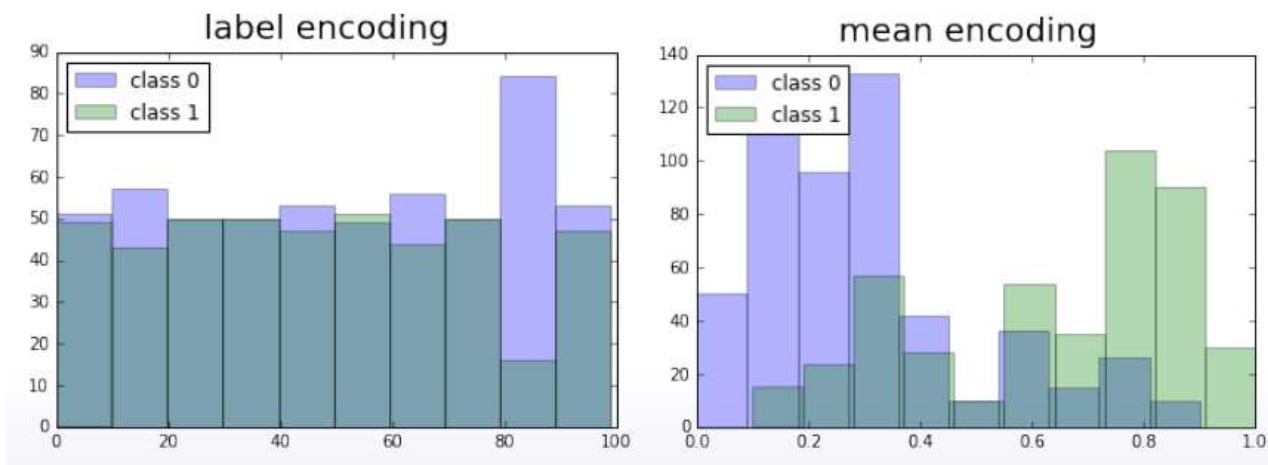
Zillow 대회에서도 이 방법을 사용하여 LB점수를 향상시켰습니다.

```
region_city_log = df_train.groupby(['regionidcity'])['logerror'].agg({'city_log_median':'median','city_log_mean':'mean',
                                                                    'city_log_std':'std','city_log_size':'size'}).reset_index()
df_train = df_train.merge(region_city_log,on='regionidcity',how='left')
df_test = df_test.merge(region_city_log,on='regionidcity',how='left')

region_zip_log = df_train.groupby(['regionidzip'])['logerror'].agg({'zip_log_median':'median','zip_log_mean':'mean',
                                                                    'zip_log_std':'std','zip_log_size':'size'}).reset_index()
df_train = df_train.merge(region_zip_log,on='regionidzip',how='left')
df_test = df_test.merge(region_zip_log,on='regionidzip',how='left')
```

# Feature Engineering Part 2

## Mean Encoding

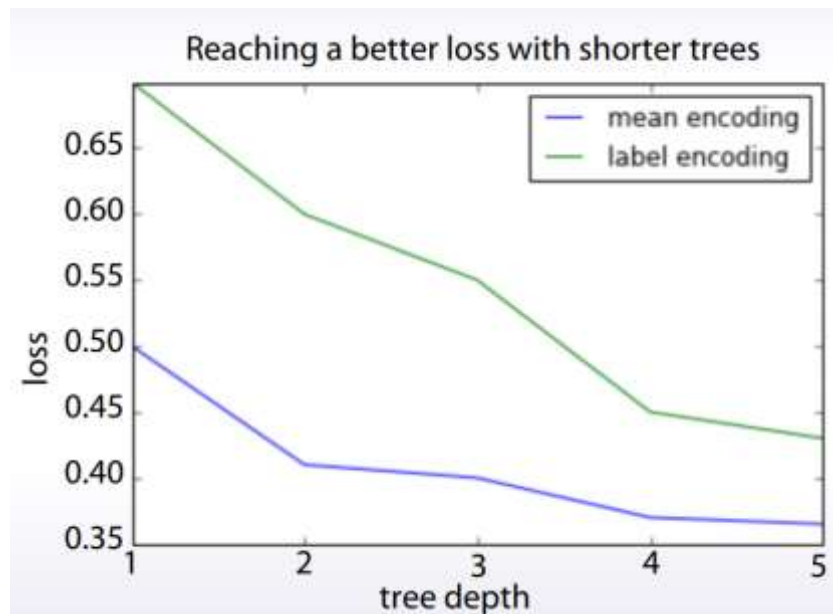


**Label Encoding** 시에는 Target 값과 직접적으로 연관이 없는 단순 숫자에 불과하지만

**Mean Encoding**을 하면 Target 값과 직접적으로 연관성이 생깁니다.

# Feature Engineering Part 2

## Mean Encoding



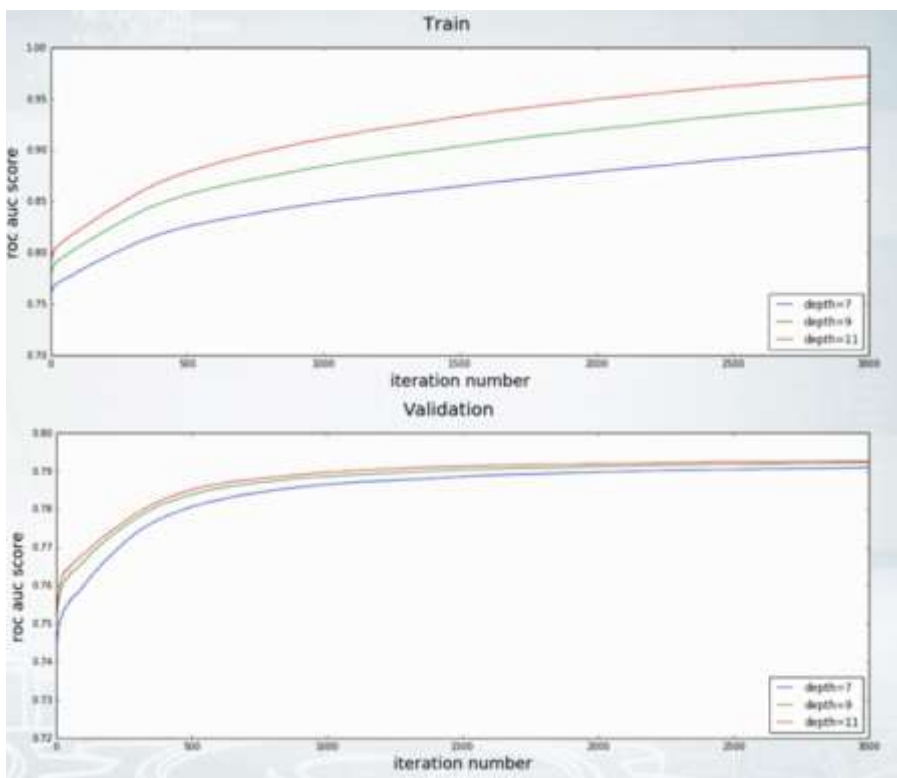
Tree Model이 매우 강력하지만 Category Feature가 High Cardinality일 경우 분할이 많이 필요하여 Tree Model에서 다루기 힘듭니다. (Overfitting을 방지하기 위해 Tree Depth를 제한)

Mean Encoding은 Label Encoding 보다 작은 Depth에서도 좋은 성능을 냅니다.

복잡하고 Target 값과 Non-Linear 관계에서 좋은 성능을 낸다고 합니다.

# Feature Engineering Part 2

## Mean Encoding



즉 Tree Depth를 증가시켰을 때, 계속해서 점수가 증가하고 Validation Score도 Tree Depth를 증가시켰을 때 같이 증가한다면(Overfitting 아닌 상황) Tree Model이 분할에 대한 정보가 더 필요한 상황

이럴 때 Mean Encoding 사용



# Feature Engineering Part 2

## Mean Encoding

Goods - number of ones in a group,

Bads - number of zeros

- $Likelihood = \frac{Goods}{Goods+Bads} = mean(target)$
- $Weight\ of\ Evidence = \ln\left(\frac{Goods}{Bads}\right) * 100$
- $Count = Goods = sum(target)$
- $Diff = Goods - Bads$

만능인 것 같아도 단점이 있는데,

Target값과 직접적으로 연관시키다 보니 Overfitting이 발생할 수 있습니다.

대회를 진행하다 보면 Discussion에 이 방법을 사용할 때는 Leak이 생기지 않도록 사용해야 된다고 합니다.

# Feature Engineering Part 2

## Mean Encoding

```
y_tr = df_tr['target'].values #target variable
skf = StratifiedKFold(y_tr,5, shuffle=True,random_state=123)

for tr_ind, val_ind in skf:
    X_tr, X_val = df_tr.iloc[tr_ind], df_tr.iloc[val_ind]
    for col in cols: #iterate though the columns we want to encode
        means = X_val[col].map(X_tr.groupby(col).target.mean())
        X_val[col+ '_mean_target'] = means
    train_new.iloc[val_ind] = X_val

prior = df_tr['target'].mean() #global mean
train_new.fillna(prior,inplace=True) #fill NaNs with global mean
```

Overfitting을 피하는 Mean Encoding

CV Loop 사용

Category중의 희소한 것은 Validation에만 있을 경우가 있는데 그러할 경우 Null 값으로 되어 Global Mean으로 대체하는 방법

Category 수가 너무 작은 경우 Target Encoding할 경우 직접적인 Hint -> Overfitting

# Feature Engineering Part 2

## Mean Encoding

- Alpha controls the amount of regularization
- Only works together with some other regularization method

$$\frac{\text{mean}(\text{target}) * \text{nrows} + \text{globalmean} * \text{alpha}}{\text{nrows} + \text{alpha}}$$

Overfitting을 피하는 Mean Encoding

Smoothing

Alpha를 사용하여 regularization

Alpha 0이면 mean(target), alpha가 크면 globalmean 사용

보통 alpha는 categorysize

# Feature Engineering Part 2

## Mean Encoding

```
if noise:
    np.random.seed(self.seed)
    _noise = noise if isinstance(noise, float) else self.noise
    if _noise > 1E-12:
        xx = np.abs(xx + normal(0, scale=_noise, size=xx.shape))
        xx = xx.div(xx.sum(axis=1), axis=0)
```

Overfitting을 피하는 Mean Encoding

Noise 추가

Mean Encoding한 값에 Noise를 추가하는 방법도 있는데, 얼마나 Noise를 추가해야 되는지 Tuning이 필요하여 잘 사용하지는 않음

# Feature Engineering Part 2

## Mean Encoding

```
cumsum = df_tr.groupby(col)['target'].cumsum() - df_tr['target']
cumcnt = df_tr.groupby(col).cumcount()
train_new[col+' mean target'] = cumsum/cumcnt
```

	cat	target	expanding_mean_encoding	mean_encoding
0	1	0	NaN	0.285714
1	1	0	0.000000	0.285714
2	2	1	NaN	0.333333
3	3	1	NaN	0.833333
4	1	0	0.000000	0.285714
5	3	1	1.000000	0.833333
6	1	1	0.000000	0.285714
7	2	0	1.000000	0.333333
8	3	1	1.000000	0.833333
9	2	0	0.500000	0.333333
10	1	1	0.250000	0.285714
11	1	0	0.400000	0.285714
12	3	1	1.000000	0.833333
13	3	0	1.000000	0.833333
14	3	1	0.800000	0.833333
15	1	0	0.333333	0.285714

Overfitting을 피하는 Mean Encoding

Expanding mean

일반적인 Mean Encoding은 각 Category변수에 하나의 값으로 나오지만, expanding mean은 왼쪽 결과처럼 결과가 uniform하지 않습니다.

하지만 이 방법은 leak이 적고 catboost에 구현되어 있는데 많은 성능향상을 가져왔다고 합니다.

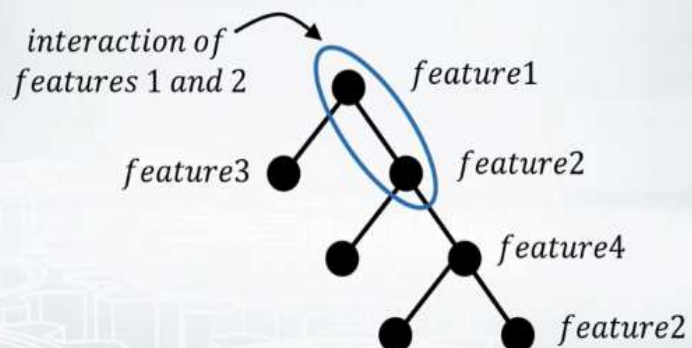
# Feature Engineering Part 2

## Mean Encoding

그 밖에 Tip

### Interactions and numerical features

- Analyzing fitted model
- Binning numeric and selecting interactions



Tree의 상호작용을 분석하여 Mean Encoding 수행

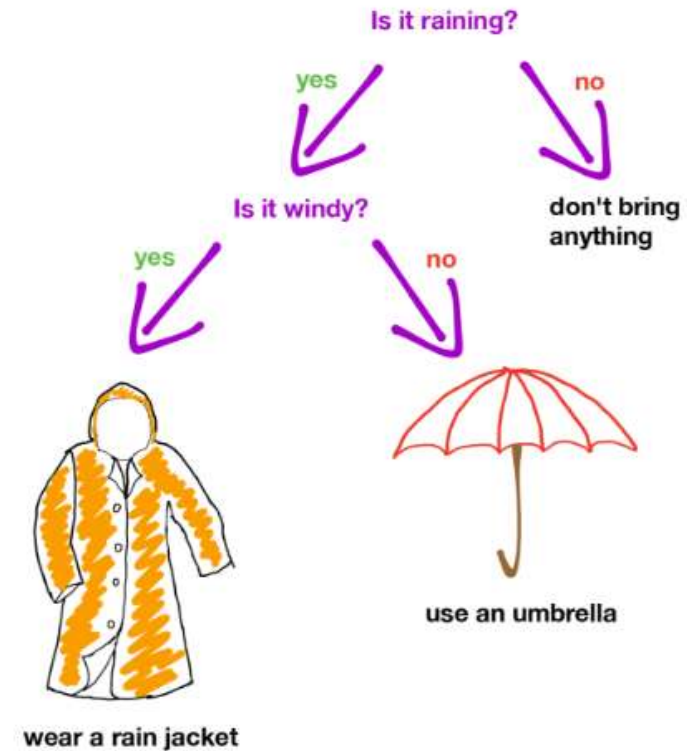
Tree Model이 Split할 때 feature1과 feature2 처럼되면 서로 상호작용하는건데 이러한 분할이 많을수록 Mean Encoding 시 성능이 잘 나온다고 합니다.

Feature끼리 Combine하여 Target Mean Encoding 수행

# Feature Engineering Part 2 실습

# Modeling

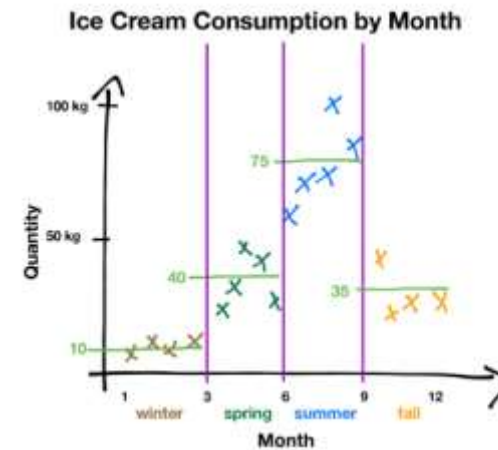
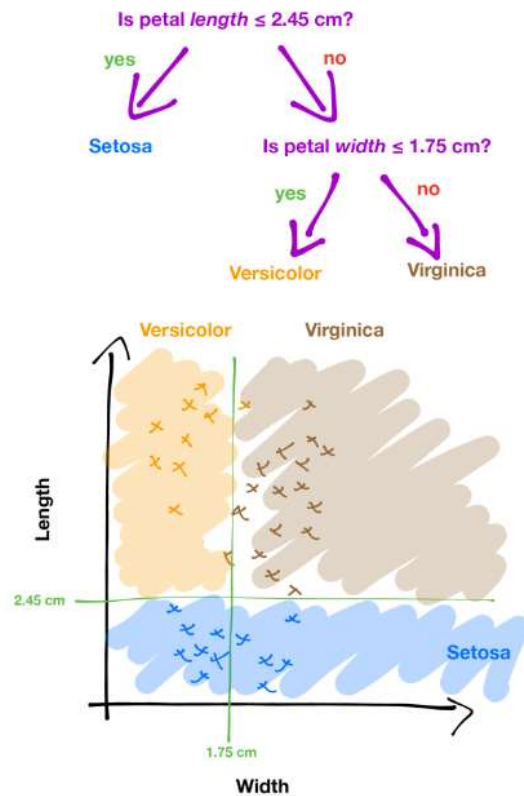
## Tree 알고리즘





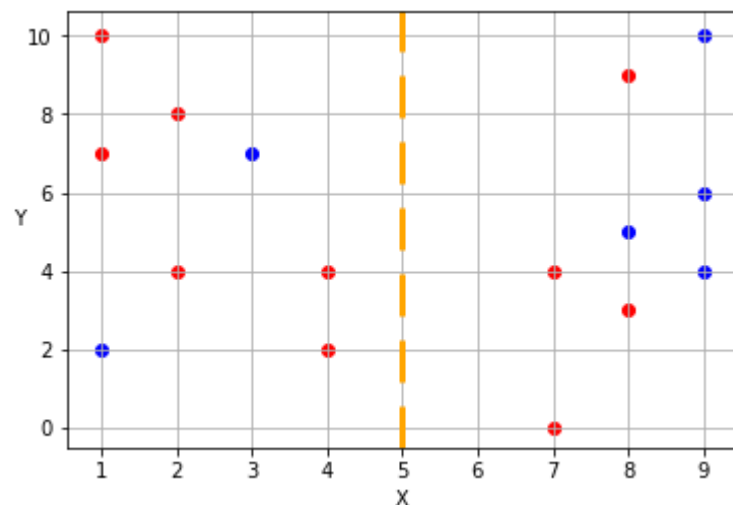
# Modeling

## Tree 알고리즘



# Modeling

## Tree 알고리즘



Decision Tree는 Tree가 나뉘지고 나서 각 영역의 순도가 증가하고, 불순도 혹은 불확실성이 최대한 감소하도록 하는 방향으로 학습을 진행합니다.

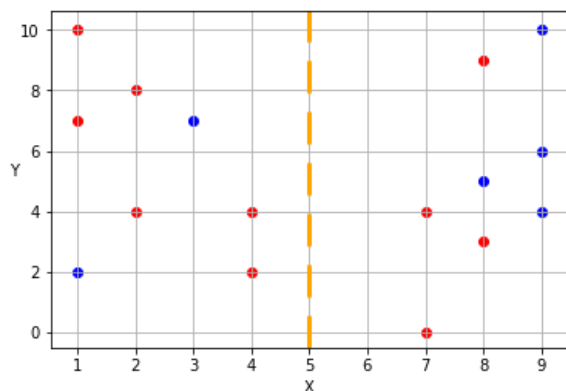
순도가 증가/불확실성이 감소하는 것을 두고 정보이론에서는 정보획득이라고 합니다.

$$Entropy(A) = - \sum_{k=1}^m p_k \log_2 (p_k)$$

모든 레코드가 동일한 Category면 엔트로피는 0  
동일하게 반반씩 섞여 있을 경우는 불확실성이 최대라서 엔트로피는 1

# Modeling

## Tree 알고리즘



엔트로피가 감소하는 방향으로  
학습을 진행, 전체 영역을 반으  
로 나눴을 때 엔트로피 계산

$$Entropy(A) = - \sum_{k=1}^m p_k \log_2(p_k)$$

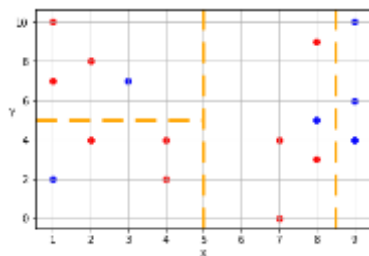
```
def calculate_entropy(frame):  
    entropy = 0  
    for p in frame['target'].value_counts() / frame.shape[0]:  
        entropy -= p * np.log2(p)  
    return entropy
```

```
calculate_entropy(frame)
```

0.954434002924965

```
0.5 * calculate_entropy(frame.query('x < 5')) + 0.5 * calculate_entropy(frame.query('x > 5'))
```

0.9056390622295665



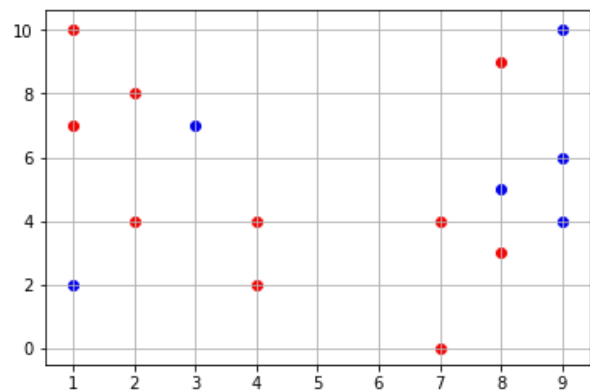
```
0.25 * calculate_entropy(frame.query('x < 5 and y > 5')) + 0.25 * calculate_entropy(frame.query('x < 5 and y < 5')) + 0.25 * calculate_entropy(frame.query('x > 5 and y > 5')) + 0.25 * calculate_entropy(frame.query('x > 5 and y < 5'))
```

0.6312415918818671

# Modeling

## Tree 알고리즘

$$G.I(A) = \sum_{i=1}^d \left( R_i \left( 1 - \sum_{k=1}^m p_{ik}^2 \right) \right)$$



엔트로피 외에 불순도 지표로 많이 쓰이는  
Gini Index도 있음

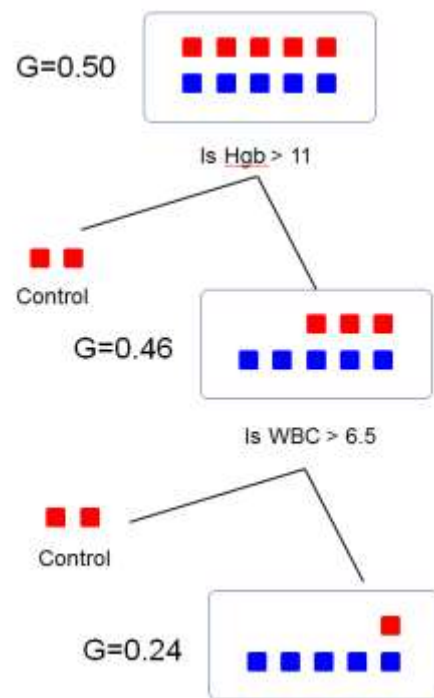
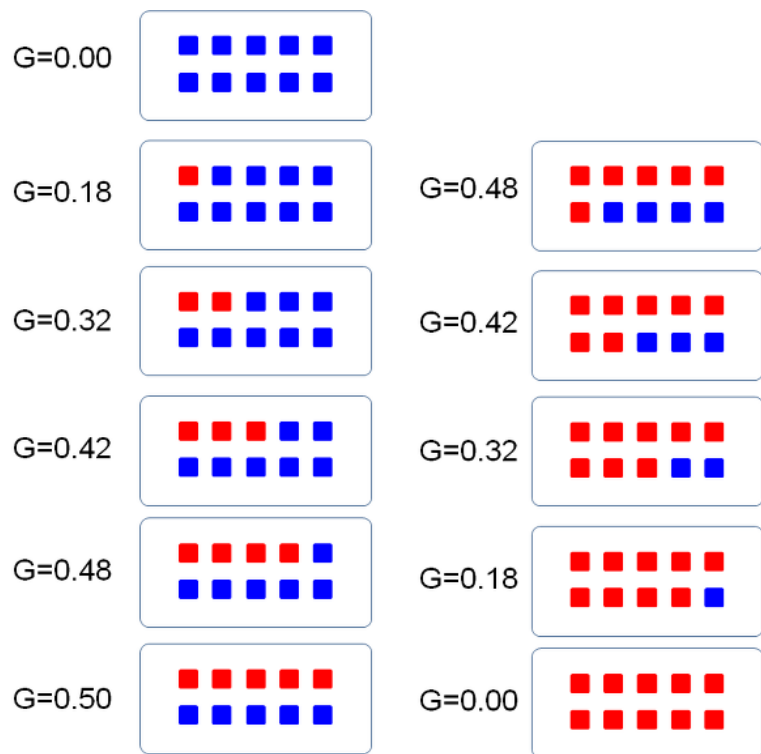
```
def gini(frame):  
    impurity = 1  
    for p in frame['target'].value_counts() / frame.shape[0]:  
        impurity -= p**2  
    return impurity
```

```
gini(frame)
```

```
0.46875
```

# Modeling

## Tree 알고리즘



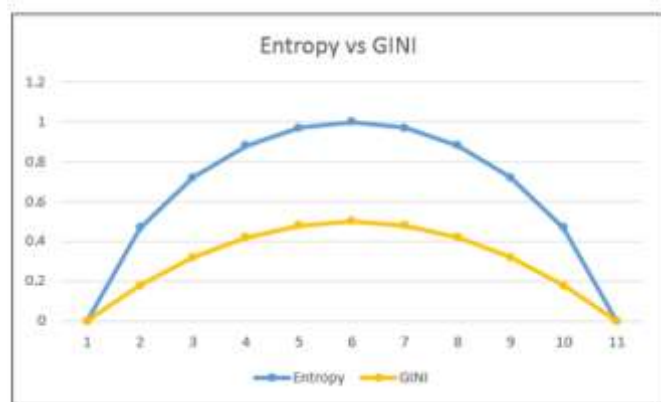
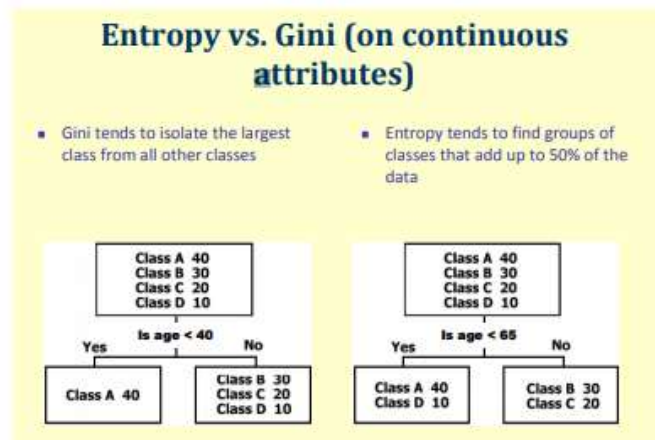
# Modeling

## Tree 알고리즘

분류가 조금 달라지긴 하지만 비슷한 Tree를 만들어냅니다. 2% 미만

사이킷런 Default는 Gini Index

Gini Index가 계산속도가 더 빠름  
다만, Gini Index는 가장 빈도 높은 클래스를 한쪽 가지로 고립시키는 경향이 있는 반면 엔트로피는 조금 더 균형 잡힌 트리를 만듭니다.



# Modeling

## Tree 알고리즘

만약 제한을 두지 않고 무한하게 Tree를 만든다면?  
훈련 데이터에 아주 가깝게 맞추려고 해서 **Overfitting이 발생!**

그래서 어느정도 진행되면 그만하라는 parameter 들이 필요합니다. (Regularization)

각 parameter가 어떤 의미를 가지는지 알아야, Overfitting, Underfitting을 제어할 수 있어야 합니다.

사이킷런 파라미터 공통점

min으로 시작하는 매개변수를 증가시키거나, max로 시작하는 매개변수를 감소시키면 모델에 규제가 커집니다. 즉 Model을 더 Underfitting시킵니다.

Min_parameter	Max_parameter	Model이 받는 영향
증가	감소	Underfitting
감소	증가	Overfitting

# Modeling

## Tree 알고리즘

```
class sklearn.tree.DecisionTreeClassifier(criterion='gini',  
splitter='best', max_depth=None, min_samples_split=2,  
min_samples_leaf=1, min_weight_fraction_leaf=0.0,  
max_features=None, random_state=None, max_leaf_nodes=None,  
min_impurity_decrease=0.0, min_impurity_split=None,  
class_weight=None, presort=False) ¶ \[source\]
```

Criterion: gini or entropy (default='gini')  
split의 품질을 측정하는 변수

Splitter: best or random (default='best')  
각 노드에서 split을 선택하는 전략을 정의

Max\_feature: int, float, string or None, optional (default=None)  
가장 좋은 split 기준을 찾을 때 고려하는 feature 수



# Modeling

## Tree 알고리즘

```
class sklearn.tree.DecisionTreeClassifier(criterion='gini',  
splitter='best', max_depth=None, min_samples_split=2,  
min_samples_leaf=1, min_weight_fraction_leaf=0.0,  
max_features=None, random_state=None, max_leaf_nodes=None,  
min_impurity_decrease=0.0, min_impurity_split=None,  
class_weight=None, presort=False)¶ \[source\]
```

**max\_depth** : int or None, optional (default=None)  
트리의 최대 깊이, int or None, None일 경우 모든 잎이 pure하거나  
min\_samples\_split 보다 작아질때까지 분할한다.

**min\_samples\_split** : int, float, optional (default=2)  
Node를 분할할때 필요한 sample들의 최소 개수  
Float은  $\text{ceil}(\text{min\_samples\_split} * n\_samples)$

**min\_samples\_leaf** : int, float, optional (default=1)  
리프 노드에 있어야하는 최소 샘플 수

# Modeling

## Tree 알고리즘

```
class sklearn.tree.DecisionTreeClassifier(criterion='gini',
splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0,
max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
class_weight=None, presort=False)¶ \[source\]
```

**random\_state** : int, RandomState instance or None, optional  
(default=None)

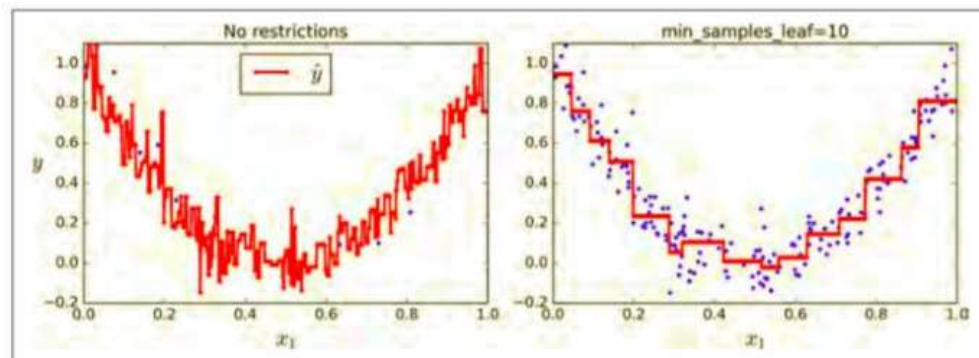
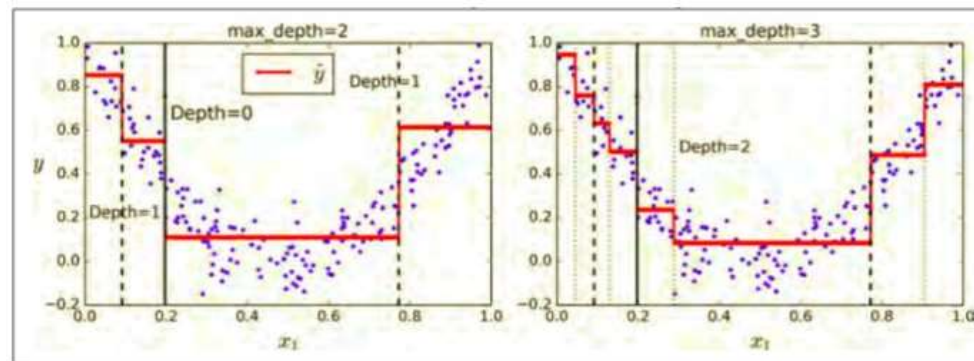
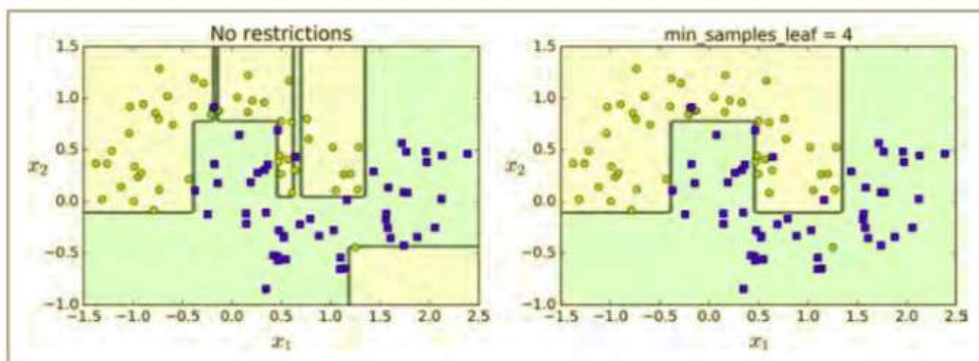
랜덤한 숫자를 만들기 위한 seed값 입니다.

각 노드에서 평가할 후보 특성을 무작위로 선택합니다.

아까 max\_features 매개변수에서 분할에 사용할 특성의 최대 개수를 지정할 수 있었는데, 데이터셋의 특성 개수보다 작게 설정하면 무작위로 일부 특성이 선택됩니다. 이 때 무작위의 seed가 되는 변수이고 이 값이 같으면 tree가 같지만 다르다면 tree결과가 다를 수 있습니다.

# Modeling

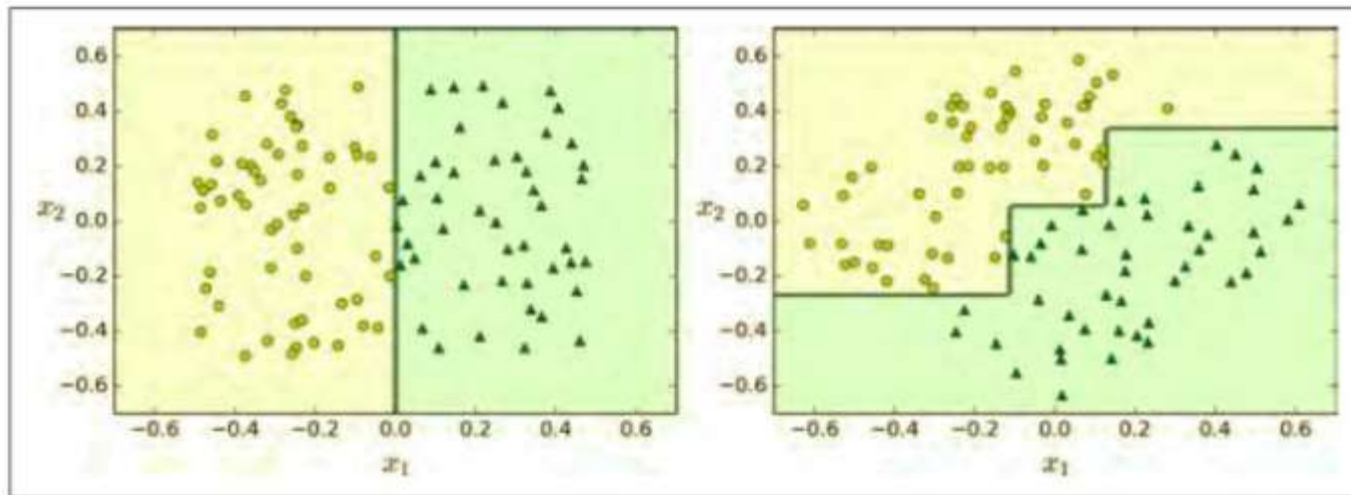
## Tree 알고리즘



규제를 하면 Tree가 공간을 나누는 횟수가 줄어들고, 규제를 하지 않으면 더 많이 분할합니다.

# Modeling

## Tree 알고리즘

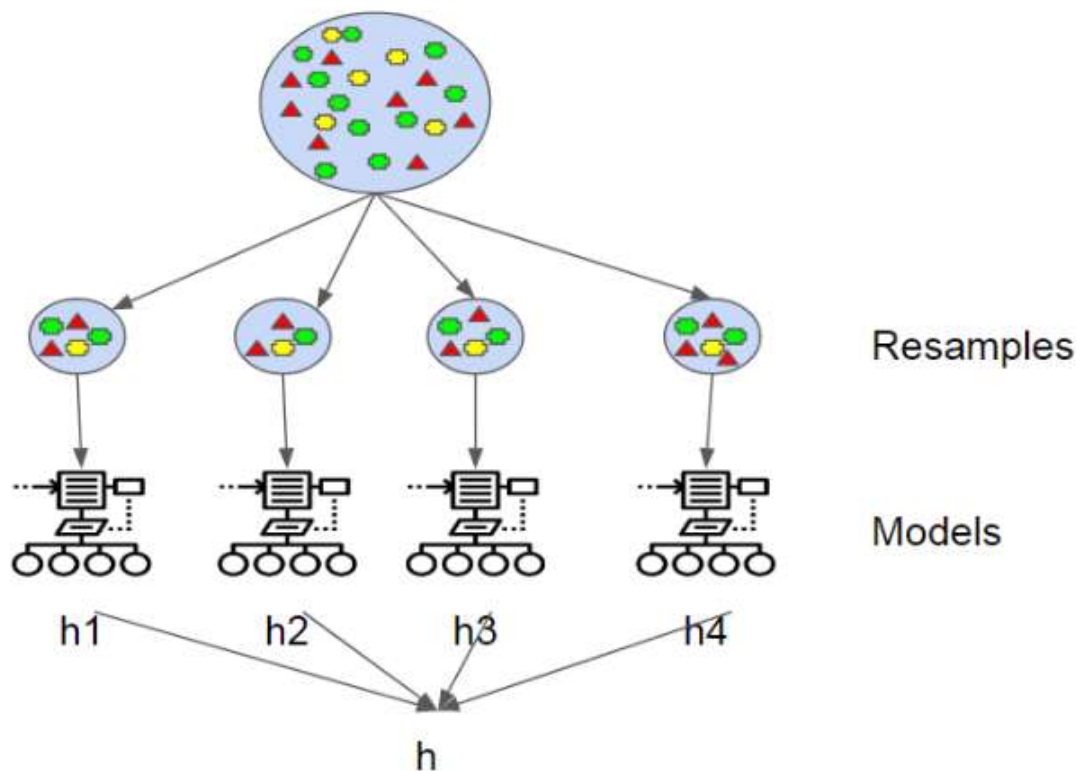


훈련 세트의 회전에 민감합니다.  
그래서 이러한 문제를 해결하는 한 가지  
방법은 훈련 데이터를 더 좋은 방향으로  
회전시키는 방법입니다.

훈련 데이터의 작은 변화에도 매우 민감  
(outlier 한 점만 빠져도 tree의 분할이  
바뀔 수 있습니다.)

# Modeling

## Tree 알고리즘



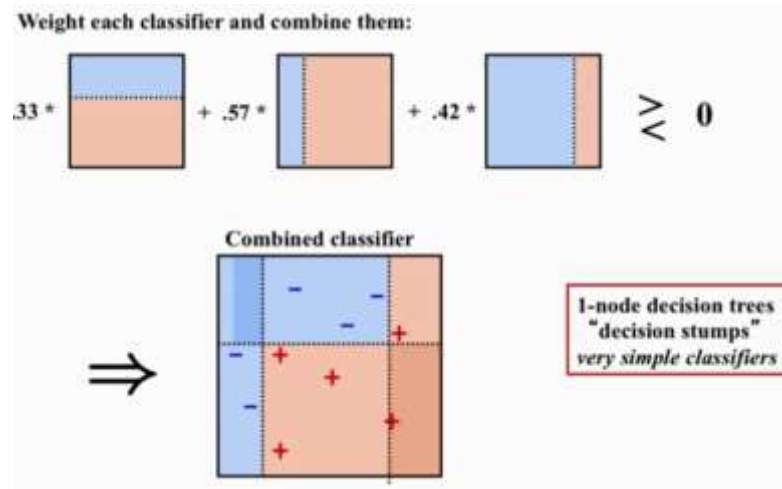
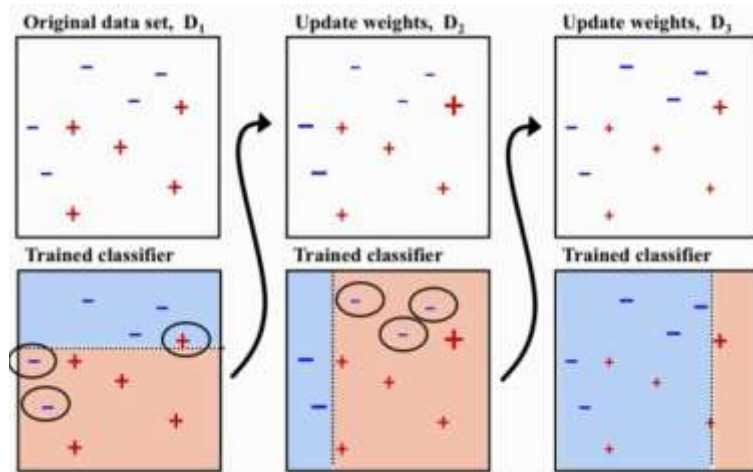
Bagging은 Train Set에서 일부를 추출하여 Model을 만든 뒤 합치는 방법을 말합니다.

Bagging은 소수의 강한 Feature가 전체 Tree를 구성할 때 영향을 주어 전체 Feature를 고려하지 못하고 다양성이 떨어질 수 있습니다.

대표적인 알고리즘은 Random Forest  
일부 Feature를 선택하여 선택된 Feature들 중에 최적의 특성을 찾는 방법

# Modeling

## Tree 알고리즘



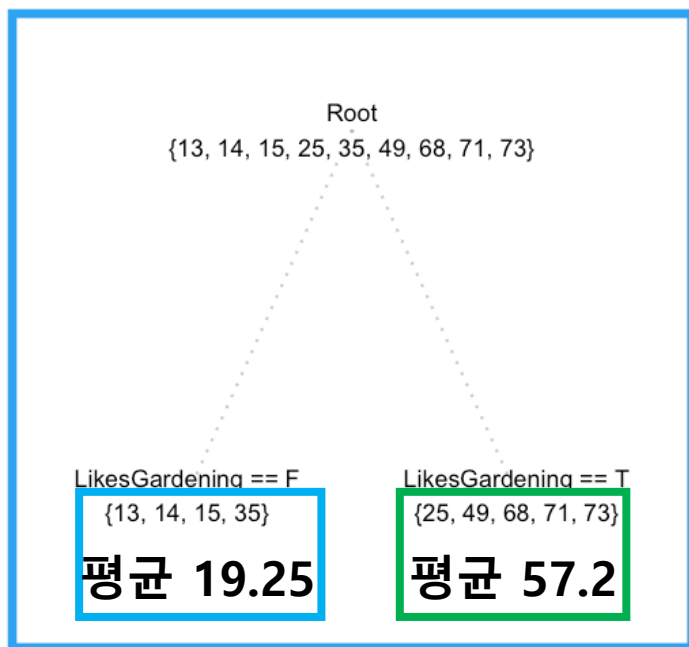
각 Iteration에서 발생한 오류가 다음 Iteration에 영향을 주면서 Simple한 Tree를 만듭니다. 나중에는 Iteration별 가중치를 바탕으로 각 Tree를 결합하여 강한 Model을 만드는 방식입니다.



# Modeling

## Tree 알고리즘

Tree 1



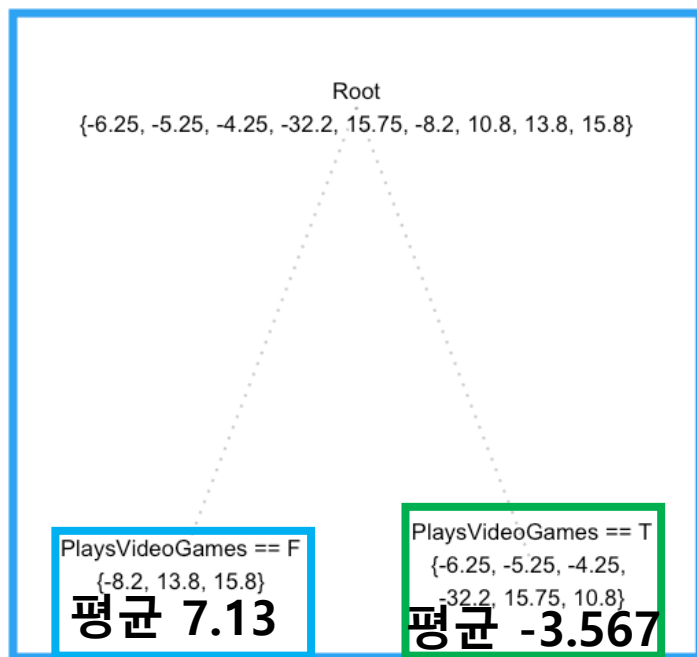
PersonID	Age	LikesGardening	PlaysVideoGames	LikesHats
1	13	FALSE	TRUE	TRUE
2	14	FALSE	TRUE	FALSE
3	15	FALSE	TRUE	FALSE
4	25	TRUE	TRUE	TRUE
5	35	FALSE	TRUE	TRUE
6	49	TRUE	FALSE	FALSE
7	68	TRUE	TRUE	TRUE
8	71	TRUE	FALSE	FALSE
9	73	TRUE	FALSE	TRUE

PersonID	Age	Tree1 Prediction	Tree1 Residual = Tree1 Prediction - Age
1	13	19.25	-6.25
2	14	19.25	-5.25
3	15	19.25	-4.25
4	25	57.2	-32.2
5	35	19.25	15.75
6	49	57.2	-8.2
7	68	57.2	10.8
8	71	57.2	13.8
9	73	57.2	15.8

# Modeling

## Tree 알고리즘

Tree2



$$\text{Combined Prediction} = \text{Tree1 Prediction} + \text{Tree2 Prediction}$$

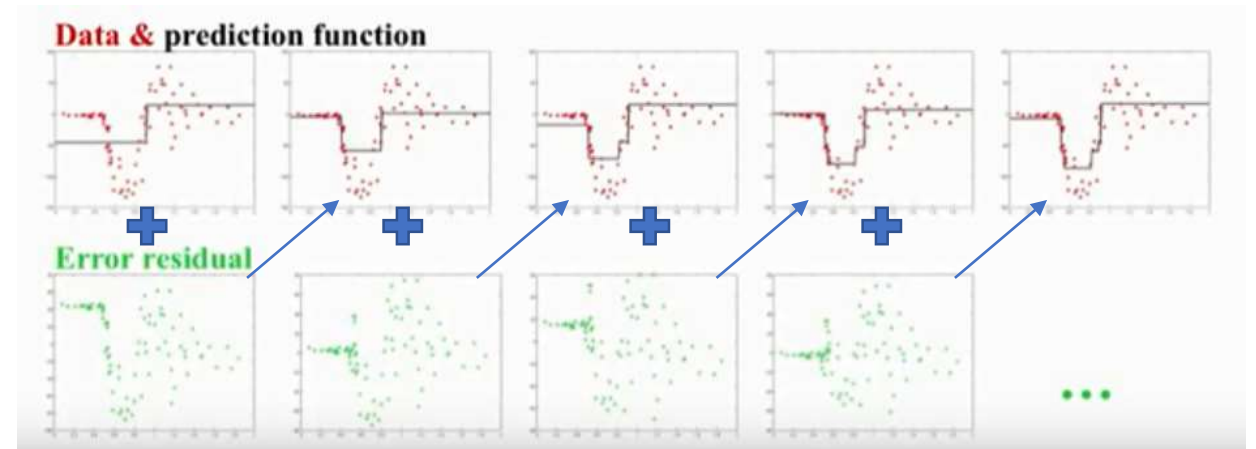
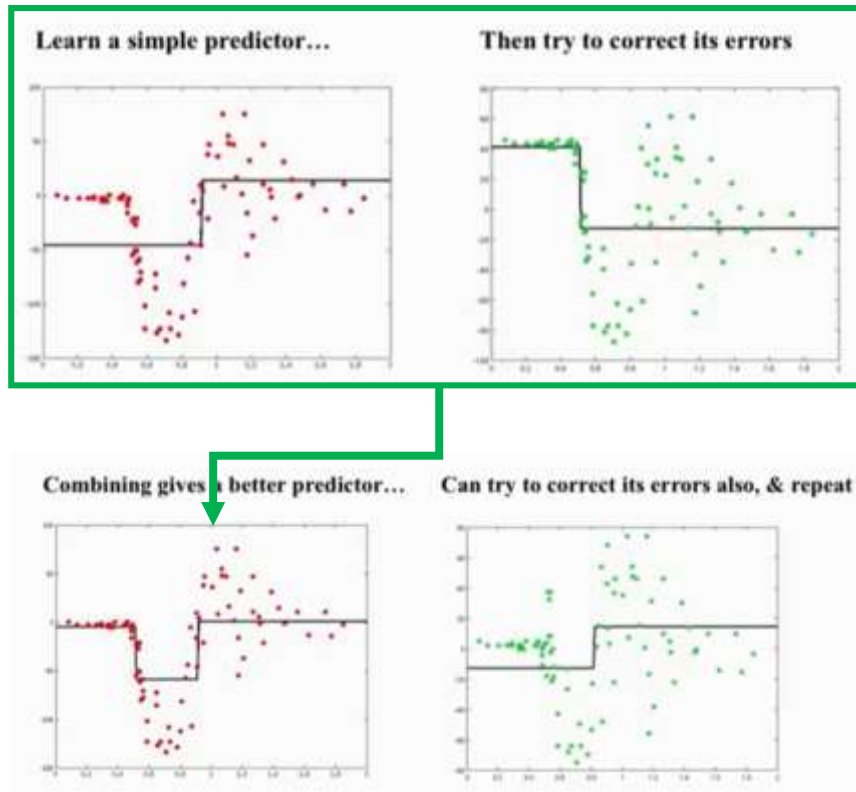
$$\text{Final Residual} = \text{Combined Prediction} - \text{Age}$$

PersonID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.567	15.68	2.683
2	14	19.25	-5.25	-3.567	15.68	1.683
3	15	19.25	-4.25	-3.567	15.68	0.6833
4	25	57.2	-32.2	-3.567	53.63	28.63
5	35	19.25	15.75	-3.567	15.68	-19.32
6	49	57.2	-8.2	7.133	64.33	15.33
7	68	57.2	10.8	-3.567	53.63	-14.37
8	71	57.2	13.8	7.133	64.33	-6.667
9	73	57.2	15.8	7.133	64.33	-8.667
Tree1 SSE		Combined SSE				
1994		1765				



# Modeling

## Tree 알고리즘



# Modeling

## Tree 알고리즘

$$\begin{aligned} F_1(x) &= y \\ h_1(x) &= y - F_1(x) \\ F_2(x) &= F_1(x) + h_1(x) \end{aligned}$$

$$\begin{aligned} h_2(x) &= y - F_2(x) \\ F_3(x) &= F_2(x) + h_2(x) \end{aligned}$$

$$\begin{aligned} h_3(x) &= y - F_3(x) \\ F_4(x) &= F_3(x) + h_3(x) \end{aligned}$$

$$\begin{aligned} h_m(x) &= y - F_m(x) \\ F_{m+1}(x) &= F_m(x) + h_m(x) \end{aligned}$$

$$\begin{aligned} F_1(x) &= y \\ h_1(x) &= y - F_1(x) \\ F_2(x) &= F_1(x) + h_1(x) \\ h_2(x) &= y - F_2(x) \\ F_3(x) &= F_2(x) + h_2(x) \\ h_3(x) &= y - F_3(x) \\ F_4(x) &= F_3(x) + h_3(x) \\ h_m(x) &= y - F_m(x) \\ F_{m+1}(x) &= F_m(x) + h_m(x) \end{aligned}$$

# Modeling

## Tree 알고리즘

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$

$$F_1(x) = y$$

$$h_1(x) = y - F_1(x) = -\frac{\partial J}{\partial F(x)}$$

$$F_2(x) = F_1(x) - \frac{\partial J}{\partial F(x)}$$

$$h_2(x) = y - F_2(x) = -\frac{\partial J}{\partial F(x)}$$

$$F_3(x) = F_2(x) - \frac{\partial J}{\partial F(x)}$$

$$h_3(x) = y - F_3(x) = -\frac{\partial J}{\partial F(x)}$$

$$F_4(x) = F_3(x) - \frac{\partial J}{\partial F(x)}$$

Loss function  $L(y, F(x)) = (y - F(x))^2/2$

We want to minimize  $J = \sum_i L(y_i, F(x_i))$  by adjusting  $F(x_1), F(x_2), \dots, F(x_n)$ .

Notice that  $F(x_1), F(x_2), \dots, F(x_n)$  are just some numbers. We can treat  $F(x_i)$  as parameters and take derivatives

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \sum_i L(y_i, F(x_i))}{\partial F(x_i)} = \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} = F(x_i) - y_i$$

So we can interpret residuals as negative gradients.

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

# Modeling

## Tree 알고리즘

XGBoost	LightGBM	의미	영향
max_depth/max_leaves	max_depth/num_leaves	Tree의 Depth를 조절, 0은 제한없음, 보통 4~12	증가하면 Tree가 더욱 깊어져 overfitting 감소하면 Tree를 나눌 수 없어 underfitting
subsample	subsample	Tree를 만들 때 Row sampling의 비율, 매 boosting iteration마다 subsample 수행, 0.5~0.9	overfitting을 조절, 값을 낮추면 overfitting이 완화된다.
colsample_bytree	colsample_bytree	Tree를 만들 때 Column sampling의 비율, 매 boosting iteration마다 subsample 수행, 0.5~0.9	overfitting을 조절, 값을 낮추면 overfitting이 완화된다.
scale_pos_weight	scale_pos_weight	binary 문제에 사용가능하고, unbalanced 문제에 도움을 줌 $\text{sum(negative instances)} / \text{sum(positive instances)}$	binary 문제에서 불균형 해소
min_child_weight	min_child_weight	tree를 분할할 때 필요한 최소 weight의 합 tuning 필요	overfitting을 조절, 값을 크게하면 overfitting이 완화된다.
reg_alpha	reg_alpha	L1 regularization, default 0	overfitting을 조절, 값을 크게하면 overfitting이 완화된다.
reg_lambda	reg_lambda	L2 regularization, default 1	overfitting을 조절, 값을 크게하면 overfitting이 완화된다.
eta	eta	얼마나 update를 수행할 것인지 결정하는 step size, 보통 0.1 ~ 0.001, 튜닝 필요	수렴하는 속도를 조절, 이 값을 작게하면 round도 늘어나야 한다. 이 값을 튜닝하는데 시간을 보내지 말고 이 값은 트레이닝 시간을 조절하는데 사용
boosting(gbtree, dart)	boosting(gdbt, rf, dart, goss)	algorithm, 기본은 gdbt, dart는 dropout이 있는 gbm	
num_round	num_iterations	tree를 얼마나 만들 것인지 결정, CV를 통한 early stopping으로 결정	
seed	seed	random으로 결정되는 것이 많은데 그 때 필요한 seed값	

Kaggle에서 많이 사용되는 대표적인 Gradient Boosting Algorithm **xgboost**, **lightgbm**

두 모델에서 중요한 Parameter만 설명해놓았습니다.

아래 사이트들 가면 정리 잘해놓았습니다.

<https://github.com/kaz-Anova/StackNet/blob/master/parameters/PARAMETERS.MD>

<https://sites.google.com/view/lauraep/parameters>

# Modeling

## Tree 알고리즘

- 최적의 Parameter Tuning은 시간이 오래 걸려서, 잘 하지 않습니다. 저같은 경우에는 어느정도 성능이 나오는 선에서 마무리 합니다. 어차피 아주 많은 모델을 사용해야 하는데, 하나하나 최적으로 튜닝하는 것보다 Feature Engineering해서 향상시키는 것이 좋다고 합니다.
- 매번 결과 그래프 보면서 tuning하는데, 시간이 좀 걸리지만 hyperopt나 beysian optimization 같은 것 돌려 놓는 것도 한가지 방법입니다.
- 완전 오버피팅 시키고 규제 걸어서 튜닝하는 방법도 있습니다.
- Learning rate와 estimator는 정확하게 선형은 아니더라도 estimator가 100일 때 learning rate가 0.1이라면 estimator를 200으로 할 경우 learning rate는 0.05로 해야 된다고 합니다.
- Learning rate, estimator, row/col sampling, regularization term(lambda, alpha, min\_child\_weight), input model, boosting type(gdbt, dart)

# Winner Solution

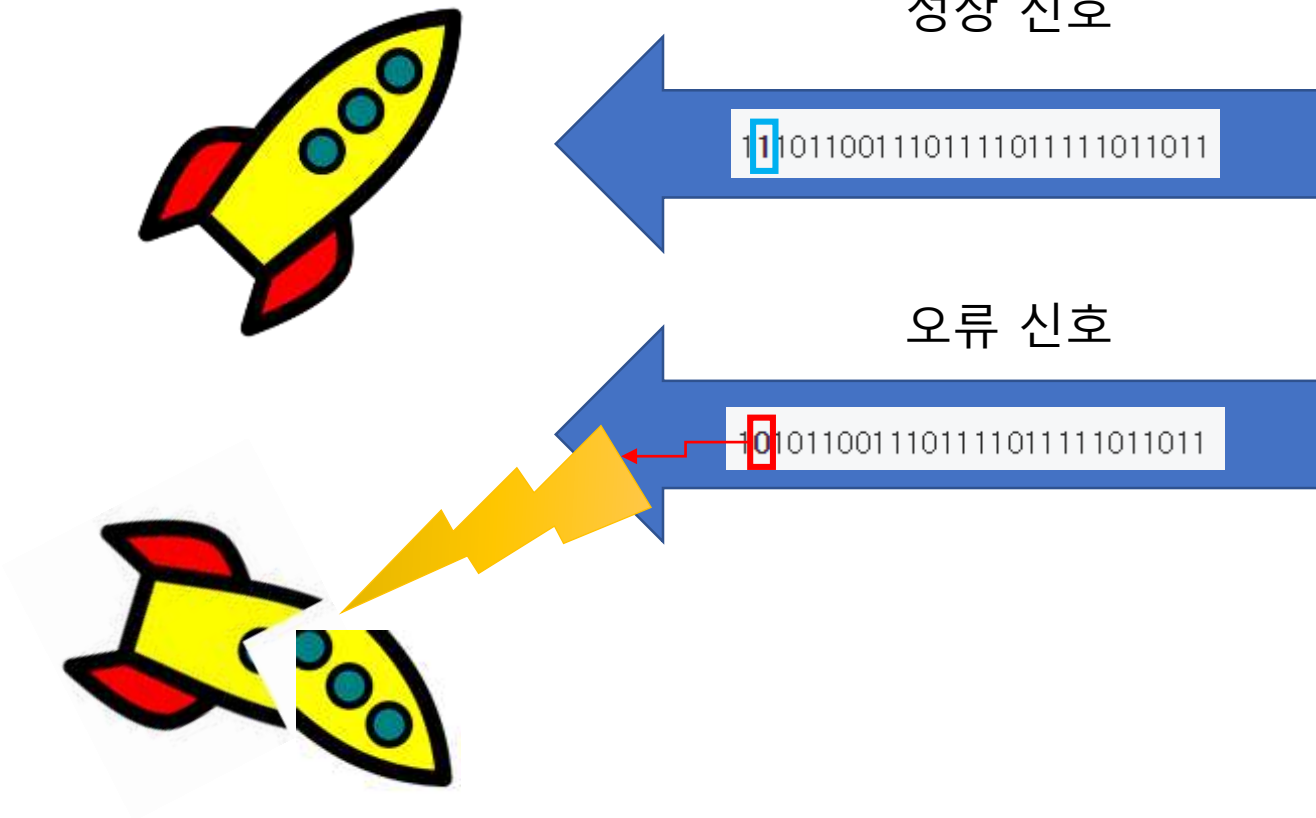
왜 여러가지 모델을 사용하면 성능이 좋아질까요?

정상 신호

1110110011101111011111011011

오류 신호

1010110011101111011111011011



# Winner Solution

왜 여러가지 모델을 사용하면 성능이 좋아질까요?

가장 단순한 오차를 수정하는 방법은 동일한 크기의 데이터를 여러 개 받아 투표하는 방법

Original signal:

1110110011

Encoded:

10,3 101011001111101100111110110011

Decoding:

1010110011

1110110011

1110110011

Majority vote:

1110110011

신호 오류는 매우 드물고 가끔 발생하기 때문

# Winner Solution

왜 여러가지 모델을 사용하면 성능이 좋아질까요?

10개의 참 정보가 있는 데이터셋

1111111111

70%의 정확도의 3개의 A, B, C의 이진분류기

70%의 확률로 1이 나오고 30%의 확률로 0이 나오는 난수 생성 모델

A, B, C의 모델을 투표방식 앙상블 기법을 통해 78%의 정확도를 얻을 수 있음



# Winner Solution

왜 여러가지 모델을 사용하면 성능이 좋아질까요?

All three are correct

$$0.7 * 0.7 * 0.7$$

$$= 0.3429$$

Two are correct

$$0.7 * 0.7 * 0.3$$

$$+ 0.7 * 0.3 * 0.7$$

$$+ 0.3 * 0.7 * 0.7$$

$$= 0.4409$$

2가지 이상 맞았기 때문에  
잘못 예측된 Model은  
Correction 됨

Two are wrong

$$0.3 * 0.3 * 0.7$$

$$+ 0.3 * 0.7 * 0.3$$

$$+ 0.7 * 0.3 * 0.3$$

$$= 0.189$$

All three are wrong

$$0.3 * 0.3 * 0.3$$

$$= 0.027$$

예측 확률은 0.7838

# Winner Solution

왜 여러가지 모델을 사용하면 성능이 좋아질까요?

Weighted Majority Vote

모든 결과를 동일하게 투표하는 것이 아닌 성능이 좋은 결과에 좀 더 가중치를 두어 투표하는 방식

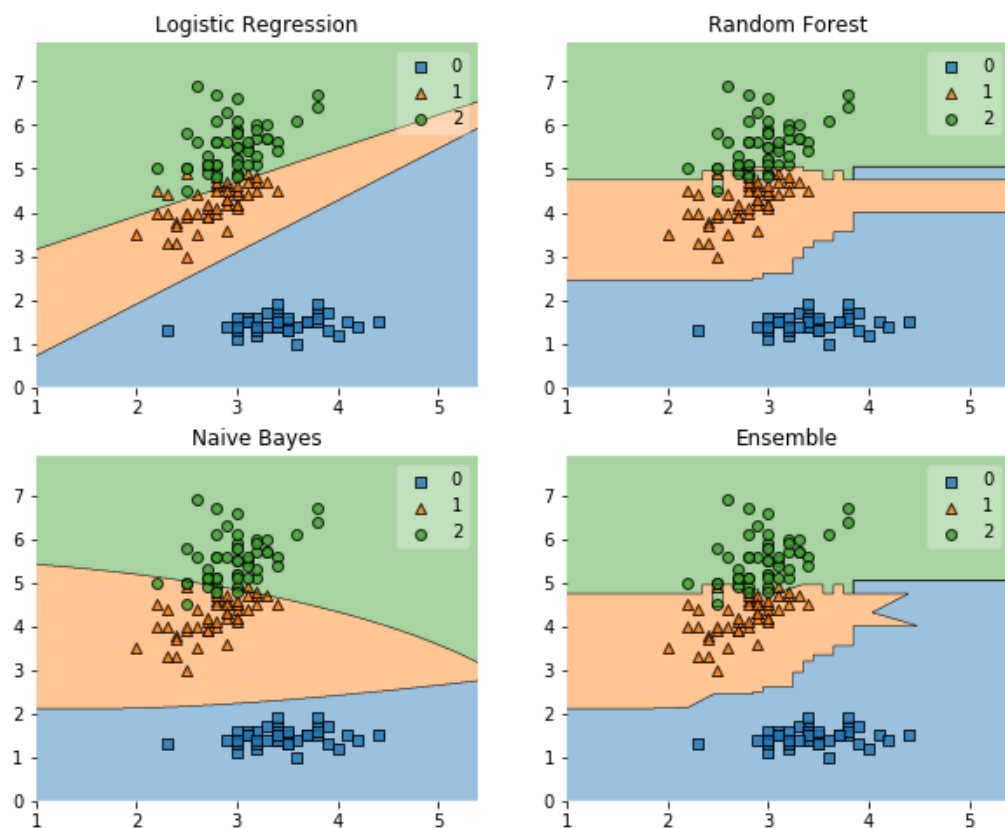
MODEL	PUBLIC ACCURACY SCORE
GradientBoostingMachine	0.65057
RandomForest Gini	0.75107
RandomForest Entropy	0.75222
ExtraTrees Entropy	0.75524
ExtraTrees Gini (Best)	0.75571
Voting Ensemble (Democracy)	0.75337
Voting Ensemble (3*Best vs. Rest)	0.75667

LB 점수에 기반하여 Weight 값 조절

<a href="#">7vs3ensemble_folder_ensemble_20170602.csv</a> a year ago by yeonmin	0.15061	0.14747
xgboost 0.150(page rank 추가)과 lstm(cleaned question) 0.180 7:3 ensemble		
<a href="#">lstm_submission_20170601.csv</a> ✖ 0.3 a year ago by JunyoungPark	0.18329	0.18063
LSTM with GloVe and magic features (val-score 0.1887), Clean questions		
+		
<a href="#">xgboost_submission_ex30.csv</a> ✖ 0.7 a year ago by yeonmin	0.15530	0.15158
page rank feature 추가하여 xgboost [2328] train-logloss:0.13286 valid-logloss:0.176092 0.176123298938		

# Winner Solution

왜 여러가지 모델을 사용하면 성능이 좋아질까요?



```
from mlxtend.classifier import EnsembleVoteClassifier

eclf = EnsembleVoteClassifier(clfs=[clf1, clf2, clf3], weights=[1,1,1])

labels = ['Logistic Regression', 'Random Forest', 'Naive Bayes', 'Ensemble']
for clf, label in zip([clf1, clf2, clf3, eclf], labels):

    scores = model_selection.cross_val_score(clf, X, y,
                                              cv=5,
                                              scoring='accuracy')

    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
          % (scores.mean(), scores.std(), label))
```

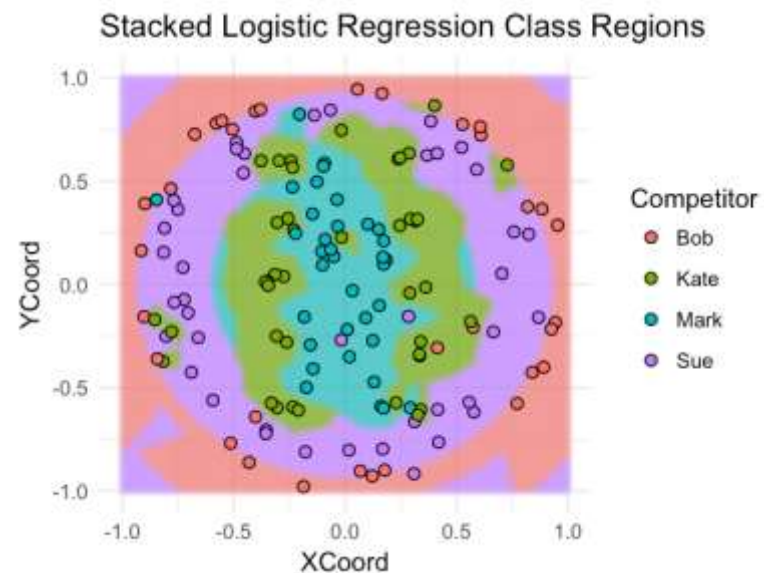
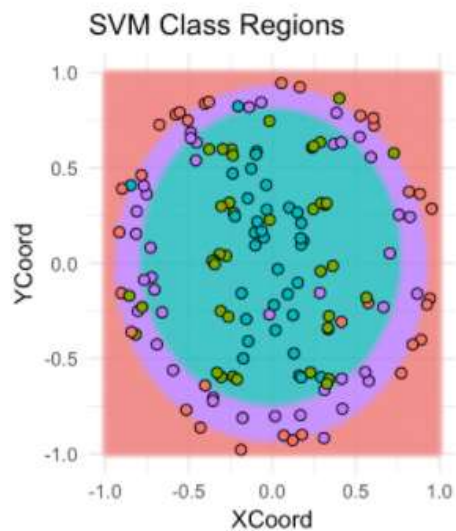
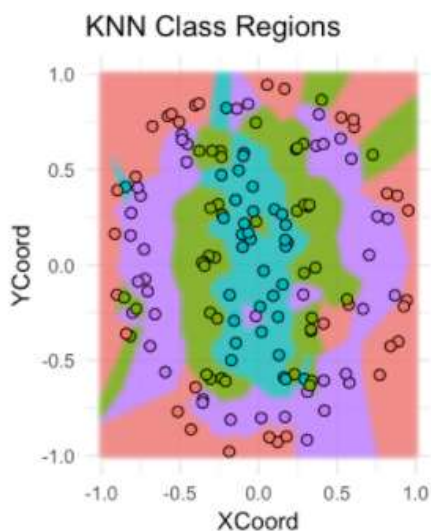
```
Accuracy: 0.90 (+/- 0.05) [Logistic Regression]
Accuracy: 0.93 (+/- 0.05) [Random Forest]
Accuracy: 0.91 (+/- 0.04) [Naive Bayes]
Accuracy: 0.95 (+/- 0.05) [Ensemble]
```

# Winner Solution

왜 여러가지 모델을 사용하면 성능이 좋아질까요?

KNN: 외곽영역이 Overfitting

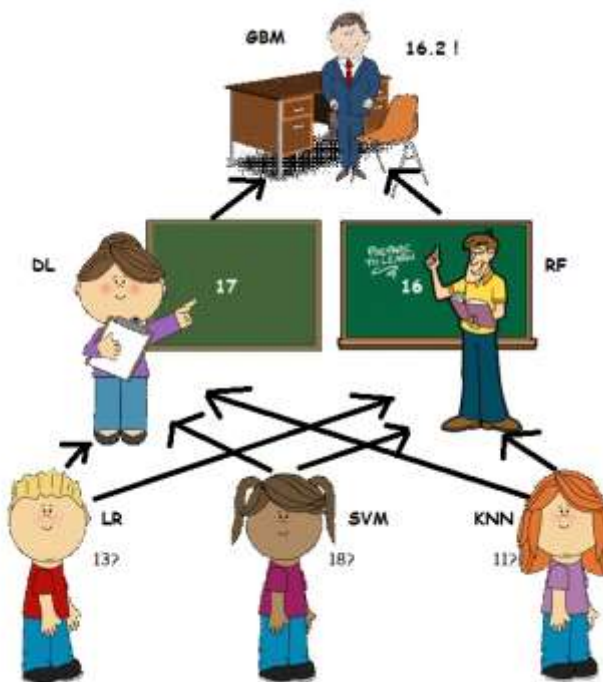
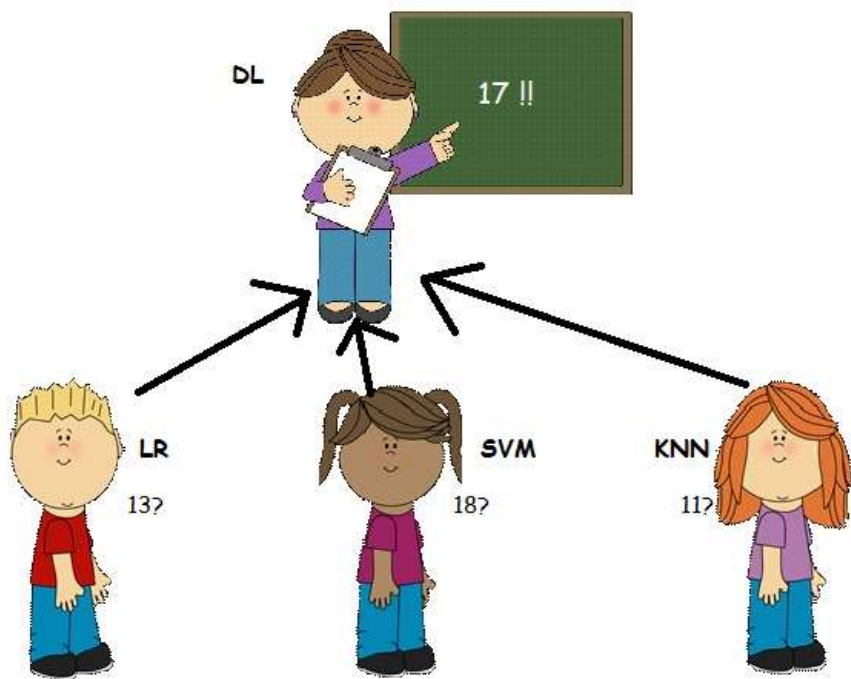
SVM: 중심에 있는 요소를 잘 구분하지 못함



# Winner Solution

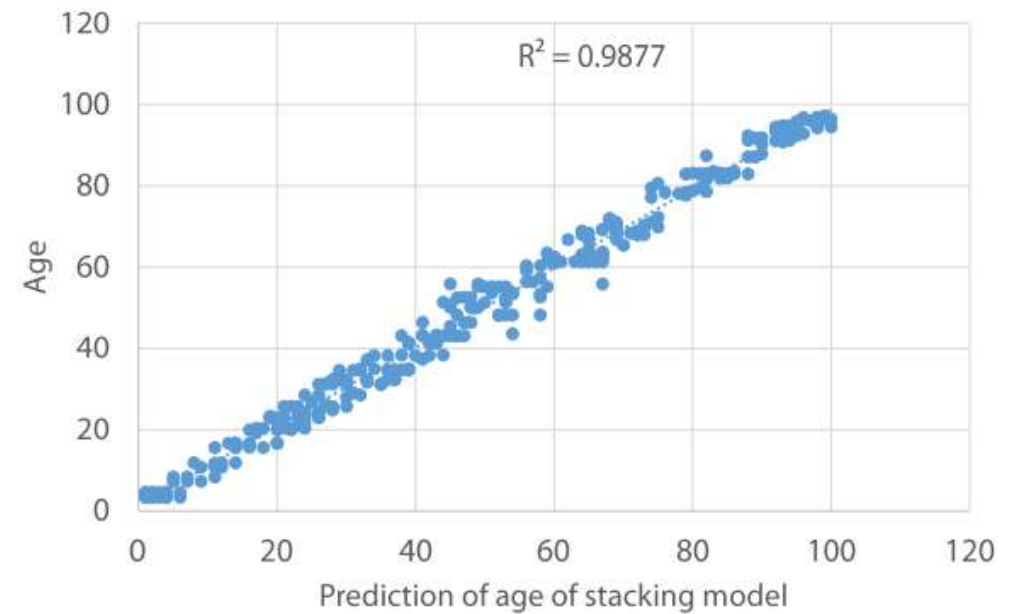
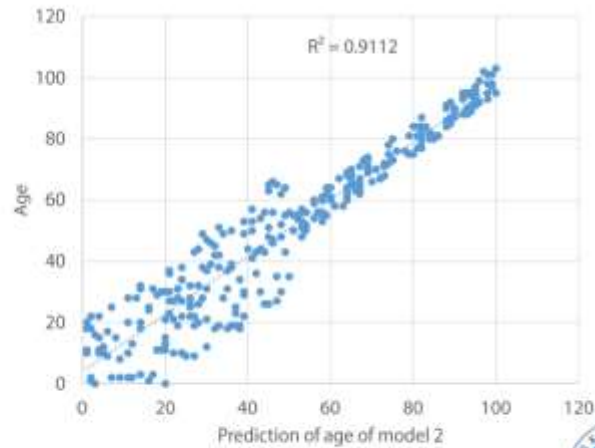
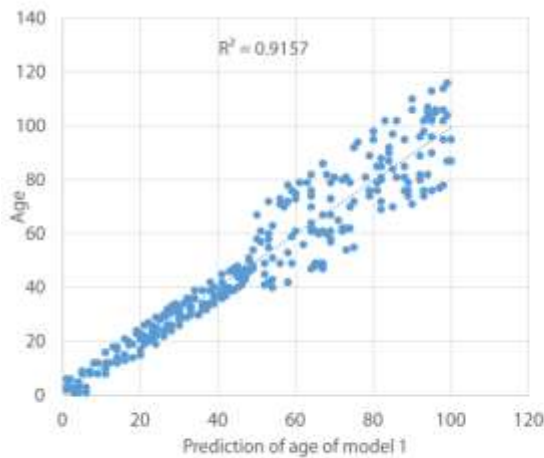
## Stacking

각 알고리즘들이 예측한 결과를 train set으로 두고 다시 모델링하여 결과를 내는 방법



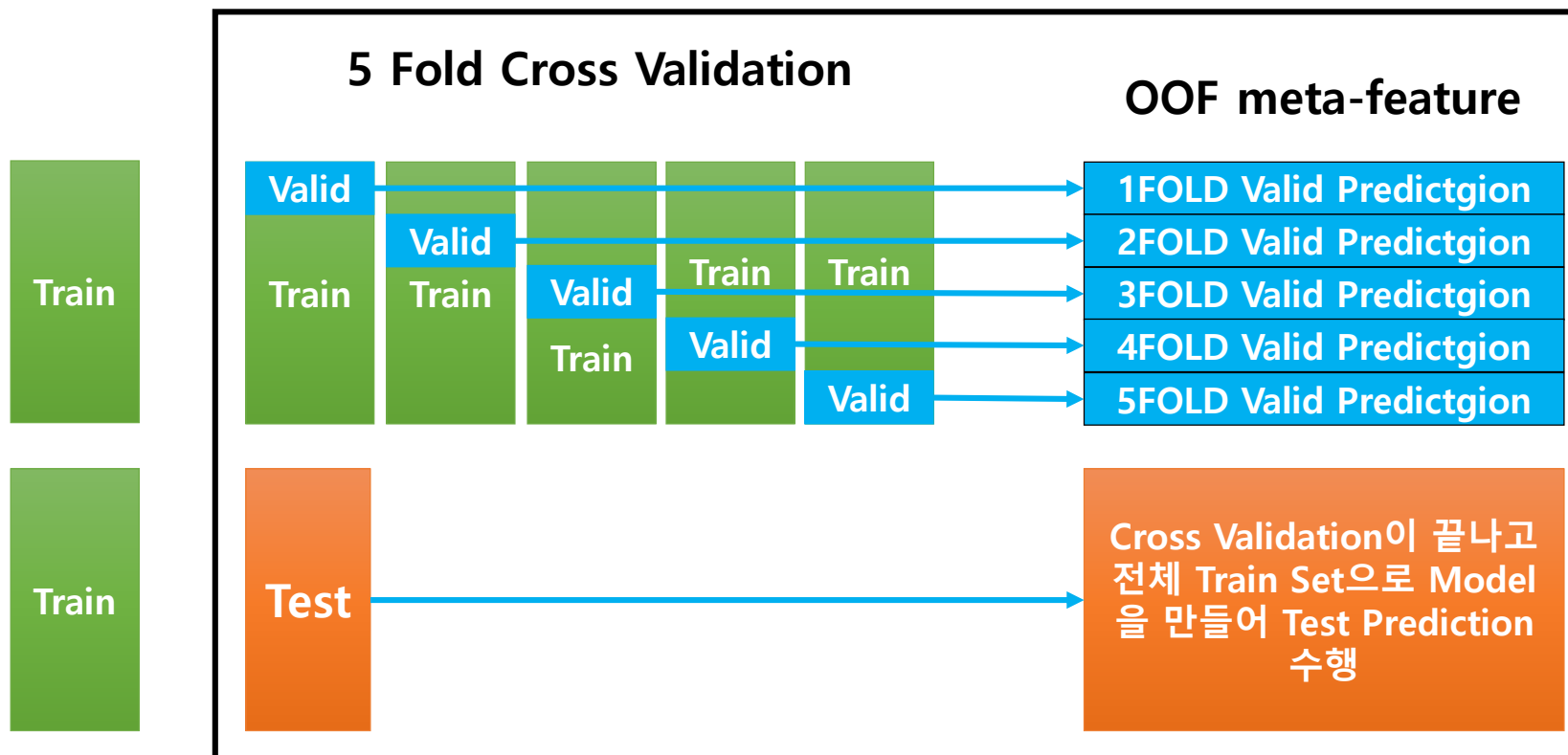
# Winner Solution

## Stacking



# Winner Solution

## Stacking



한 모델마다 이러한 과정을 반복

4가지 모델과 2가지 유형의 데이터셋이 있다면 총 8가지 OOF meta-feature 생성

A, B DataSet  
Random Forest(RF), ExtraTree(ET),  
Neural Network(NN),  
XGBoost(XG)

A-RF	A-ET	A-NN	A-XG	B-RF	B-ET	B-NN	B-XG
1	43	2	5	4	34	4	2
34	24	34	22	34	53	24	43
3	4	4	3	2	45	3	24
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
34	24	34	22	34	53	24	43
3	4	4	3	2	45	3	24

# Winner Solution

## Stacking

- 다양성을 확보하는 것이 매우 중요합니다.
- 다양성을 확보하는 방법
  - 다른 알고리즘 사용
  - 다른 Input Data를 사용
- 보통은 많이 추가하면 할 수록 성능이 좋아지지만 더 이상 개선되지 않는 시점이 발생합니다. 얼마나 많은 모델을 추가해야 할지는 알 수 없지만, 성능향상이 거의 없다면 추가하지 않고 다른 방법을 찾아야 합니다.



# Winner Solution

## Stacking

- GBM 2~3개
  - GBM 모델 추가할 때 Algorithm의 Depth를 다르게 하는데 깊게, 중간, 얇게 이렇게 추가하면 다양성이 향상됩니다.
- NeuralNetwork도 2~3개
  - GBM과 마찬가지로, layer 깊게, 중간, 얇게
- 랜덤포레스트 혹은 ExtraTree 1~2개
- 1~2개 리니어 모델
- 1~2개 KNN
- Factorization Machine (libFM)
- 데이터가 크기 않다면 SVM
- 정해진 숫자는 없으나 GrandMaster의 경험상 7.5개 모델마다 하나의 새로운 모델을 사용하면 좋음

```
gbm_peras1 = {
    'boosting': 'gbdt', 'num_leaves':10, 'learning_rate':0.01, 'min_sum_hessian_in_leaf':0.1,
    'max_depth':14, 'feature_fraction':0.5, 'min_data_in_leaf':14, 'poisson_max_delta_step':0.7,
    'bagging_fraction':0.8, 'min_gain_to_split':0, 'scale_pos_weight':1.0, 'lambda_2':0.1, 'lambda_1':0.1, 'huber_delta':0.05,
    'lambda_12':0.1, 'lambda_11':0.1, 'huber_delta':0.0, 'bagging_freq':1,
    'objective': 'regression_l1', 'seed':1, 'categorical_feature':0, 'xgbboost_dart_mode':False,
    'drop_rate':0.1, 'skip_drop':0.5, 'max_drop':50, 'top_rate':0.1, 'other_rate':0.1,
    'max_bin':255, 'min_data_in_bin':50, 'bin_construct_sample_cnt':1000000,
    'tree_round':False, 'uniform_drop':False, 'metric': 'auc'
}

gbm_peras2 = {
    'boosting': 'gbdt', 'num_leaves':24, 'learning_rate':0.03, 'min_sum_hessian_in_leaf':0.1,
    'max_depth':16, 'feature_fraction':0.5, 'min_data_in_leaf':50, 'poisson_max_delta_step':0.7, 'bagging_fraction':0.8,
    'min_gain_to_split':0, 'scale_pos_weight':1.0, 'lambda_12':0.1, 'lambda_11':0.1, 'huber_delta':0.05,
    'lambda_12':0.1, 'lambda_11':0.1, 'huber_delta':0.05,
    'bagging_freq':1, 'objective': 'huber', 'seed':1, 'categorical_feature':0, 'xgbboost_dart_mode':False, 'drop_rate':0.1,
    'skip_drop':0.5, 'max_drop':50, 'top_rate':0.1, 'other_rate':0.1, 'max_bin':255, 'min_data_in_bin':50,
    'bin_construct_sample_cnt':1000000, 'tree_round':False, 'uniform_drop':False, 'metric': 'auc'
}

gbm_peras3 = {
    'boosting': 'gbdt', 'num_leaves':28, 'learning_rate':0.03, 'min_sum_hessian_in_leaf':0.1, 'max_depth':17,
    'feature_fraction':0.6, 'min_data_in_leaf':70, 'poisson_max_delta_step':0.7, 'bagging_fraction':0.8,
    'min_gain_to_split':0, 'scale_pos_weight':1.0, 'lambda_12':0.1, 'lambda_11':0.1, 'huber_delta':0.0, 'bagging_freq':1,
    'objective': 'fair', 'seed':1, 'categorical_feature':0, 'xgbboost_dart_mode':False, 'drop_rate':0.1, 'skip_drop':0.5,
    'max_drop':50, 'top_rate':0.1, 'other_rate':0.1, 'max_bin':255, 'min_data_in_bin':50, 'bin_construct_sample_cnt':1000000,
    'tree_round':False, 'uniform_drop':False, 'metric': 'auc'
}

gbm_peras4 = {
    'boosting': 'gbdt', 'num_leaves':16, 'learning_rate':0.003, 'min_sum_hessian_in_leaf':0.1, 'max_depth':17,
    'feature_fraction':0.5, 'min_data_in_leaf':70, 'poisson_max_delta_step':0.7, 'bagging_fraction':0.8,
    'min_gain_to_split':0, 'scale_pos_weight':1.0, 'lambda_12':0.1, 'lambda_11':0.1, 'huber_delta':0.0,
    'seed':1, 'categorical_feature':0, 'xgbboost_dart_mode':False, 'drop_rate':0.1, 'skip_drop':0.5, 'max_drop':50,
    'top_rate':0.1, 'other_rate':0.1, 'max_bin':255, 'min_data_in_bin':50, 'bin_construct_sample_cnt':1000000,
    'tree_round':False, 'uniform_drop':False, 'metric': 'auc'
}
```

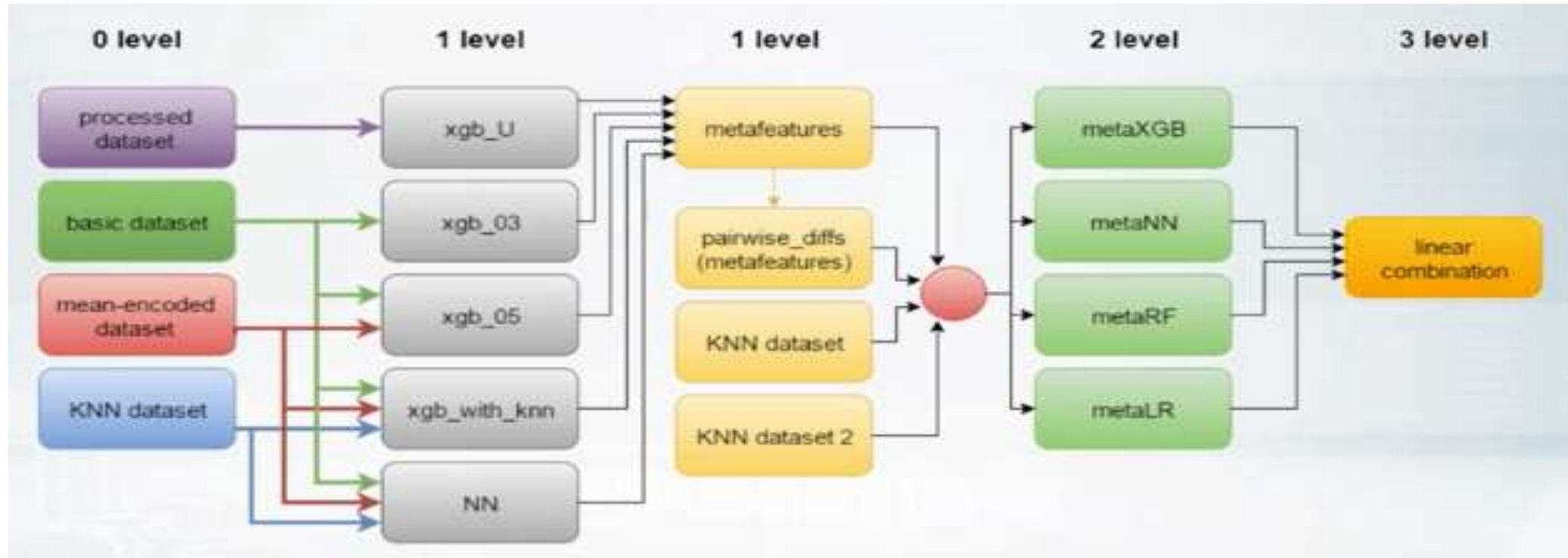
# Winner Solution

## Stacking

- 인풋데이터도 다양하게 하는데 각 모델에 맞게 튜닝 필요
- Category Feature
  - One Hot Encoding, Label Encoding, Mean Target Encoding, Count 등
- Numerical Feature
  - Outlier, Binning, Derivatives, Percentiles, Scaling
- Interaction
  - Category Combine, Col1\*/+-Col2, groupby, unsupervised(PCA, NMF) feature add

# Winner Solution

## Winner Solution



**Stacking 실습**

# 부록 GrandMaster Pipeline - KazAnova

## The Pipeline



어떤 문제인지 파악한다.

Data이 크기를 보고 Hardware 사양을 결정한다.

Metric을 살펴보고 이전에 비슷한 대회가 있었는지 살펴본다.

EDA를 시작한다. 모든 Feature에 대하여 Histogram을 그려본다. Train/Test 분포가 다른 Feature가 있는지 확인한다. 분포가 다르다면 삭제하거나, scale 변경하거나 추후 처리를 위해 기록해놓는다.

# 부록 GrandMaster Pipeline - KazAnova

## The Pipeline



시간 vs Target Value를 Plot해보고 시간에 따른 target 변화량을 확인한다.  
그리고 다른 변수들도 시간에 따라 변하는지 확인한다.

계속 데이터를 탐색해가면서 Feature들과 Target 값의 연관성을 탐색한다.

Feature들과 Feature들간의 상호관계도 파악한다.

데이터를 어느정도 파악했다면 Cross Validation 전략을 구상합니다. Cross Validation 전략은 TestSet과 동일하게 작성합니다.

# 부록 GrandMaster Pipeline - KazAnova

## Feature engineering

- The type of problem defines the feature engineering.
- **Image classification:** Scaling, shifting, rotations, CNNs. Suggestion [previous data science bowls](#).
- **Sound classifications:** Fourier , Mfcc, specgrams, scaling . [Tenso flow speech recognition](#)
- **Text classification:** Tf-idf, svd, stemming, spell checking, stop words' removal, x-grams. [StumbleUpon Evergreen Classification](#).
- **Time series:** Lags, weighted averaging, exponential smoothing . [Walmart recruitment](#).
- **Categorical :** Target enc, freq, one-hot, ordinal, label encoding. [Amazon employee](#)
- **Numerical :** Scaling , binning, derivatives ,outlier removals, dimensionality reduction. [Africa soil](#).
- **Interactions:** multiplications, divisions, group-by features . Concatenations. [Homesite](#).
- **Recommenders:** Features on transactional history. Item popularity, frequency of purchase. [Acquire Valued Shoppers](#).
- This process **can be automated** using selection with cross validation.





# 부록 GrandMaster Pipeline - KazAnova

## Modeling

- The type of problem defines the feature engineering.
- **Image classification:** CNNs (Resnet, VGG, densenet...)
- **Sound classifications:** CNNs(CRNN), LSTM
- **Text classification:** GBMs, Linear, DL, Naïve bayes, KNNs, LibFM, LIBFFM
- **Time series:** Autoregressive models, ARIMA, linear, GBMs, DL, LSTMs
- **Categorical features:** GBMs, Linear models, DL , LibFM, libFFm
- **Numerical Features:** GBMs, Linear models, DL, SVMs
- **Interactions:** GBMs, Linear models, DL
- **Recommenders:** CF, DL, LibFM, LIBFFM, GBMs

