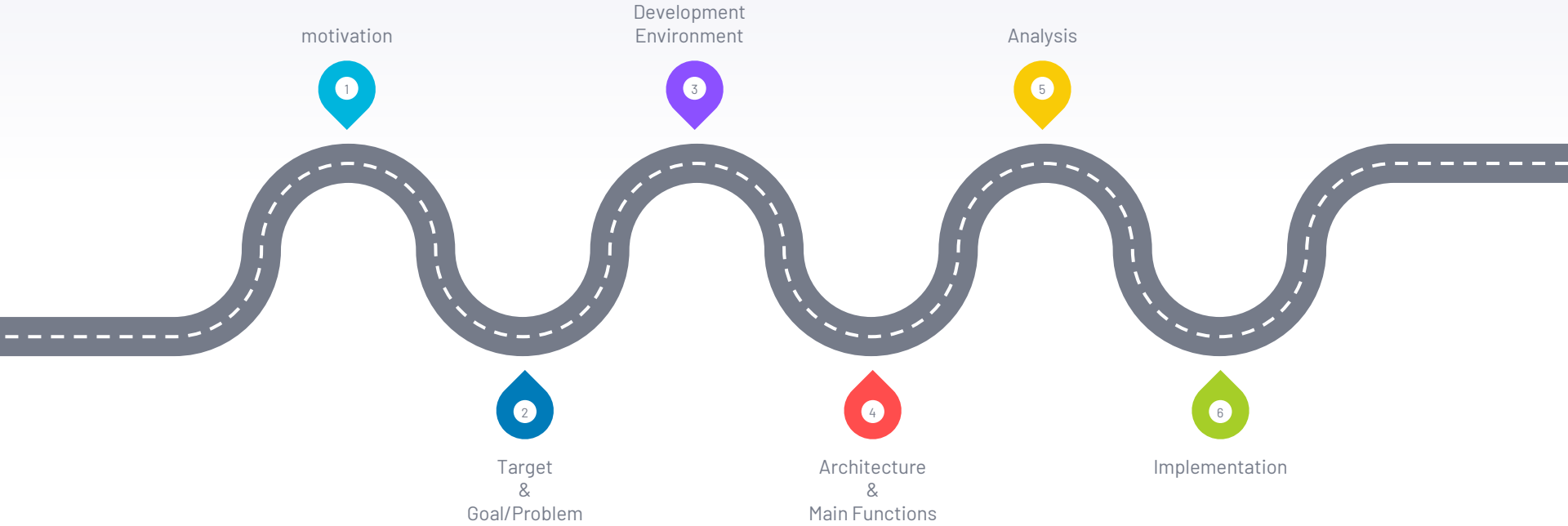
# Prepare for coding test efficiently!

2019110627 CSE Kim MyeongJu
2016104131 CSE An YoungMin
2019110634 CSE Lee ChangRyeol

# Index


motivation
1
Target
&
Goal/Problem
2
Development
Environment
3
Architecture
&
Main Functions
4
Analysis
5
Implementation
6

BEGINNERS FACE VARIOUS DIFFICULTIES IN PREPARING FOR A CODING TEST.

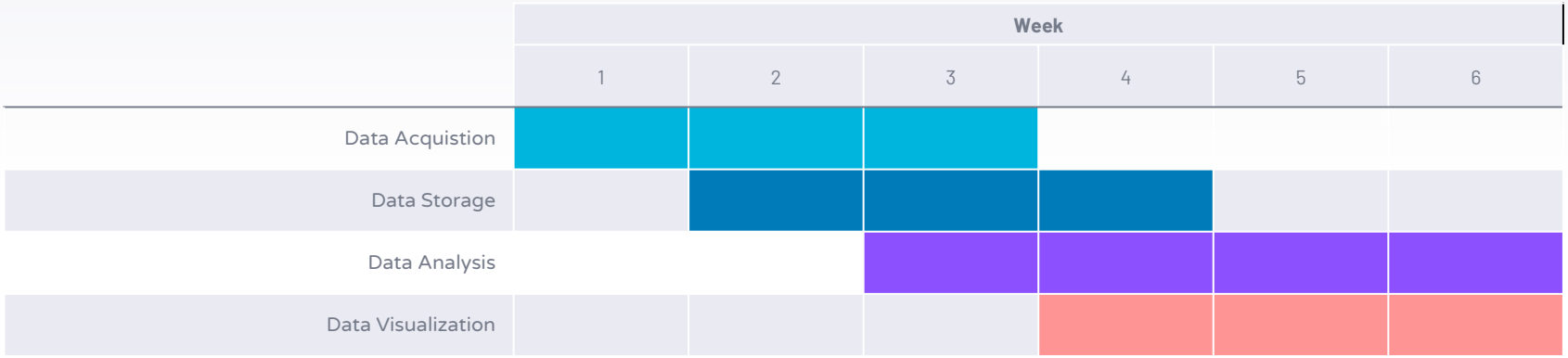
MOST PROBLEM-SOLVING SITES SHOW OVERALL INFORMATION, SO IT IS DIFFICULT TO FIND INFORMATION THAT SUITS THE USER AT A GLANCE.

# Motivation

# Team Presentation

| Kim MyeongJu | An YoungMin | Lee ChangRyeol |
| --- | --- | --- |
| Team Leader | Data Analysis | Data Acquisition |
| Scrum Master | Data Storage | Data Analysis |
| Data Acquisition | Optimizing  Performance | Project Document |
| Data Visualization | | |

# Gantt chart


Week
1
2
3
4
5
6
Data Acquistion
Data Storage
Data Analysis
Data Visualization

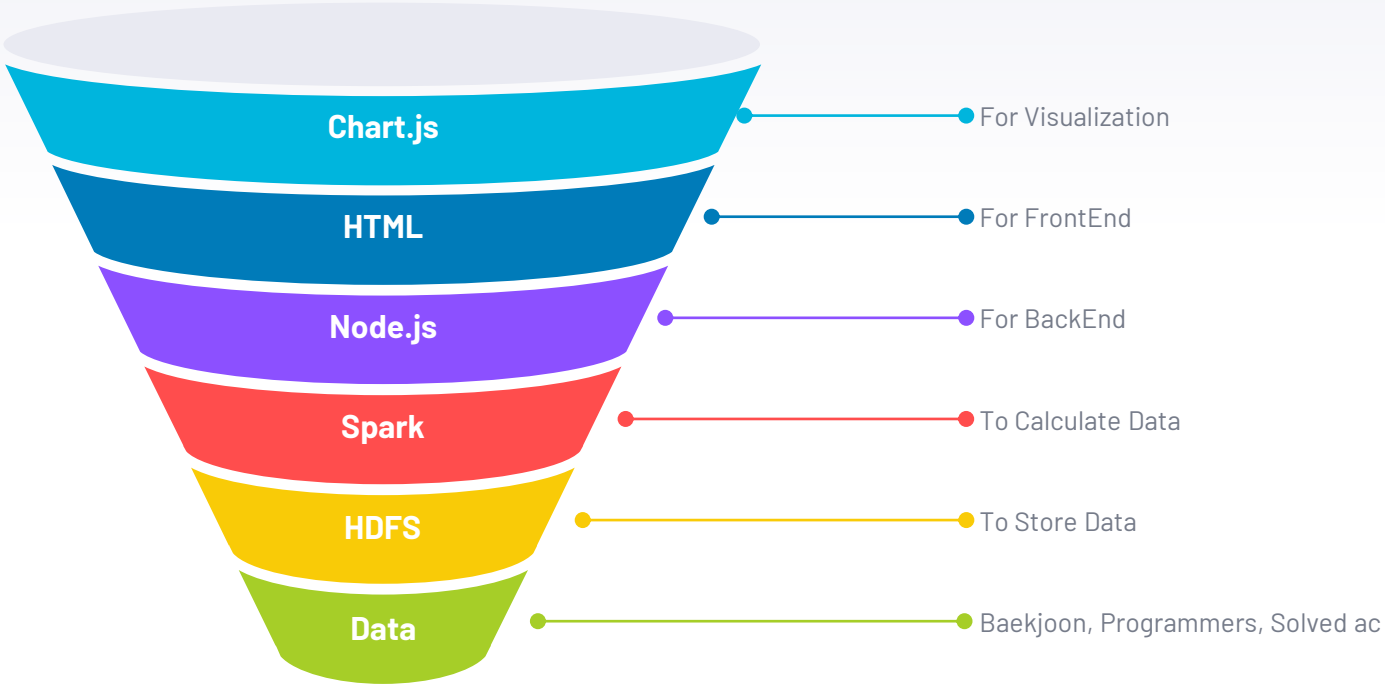
# Target & Goal/Problem

For Anyone preparing for a coding test

Problem : There is little information on companies for coding test

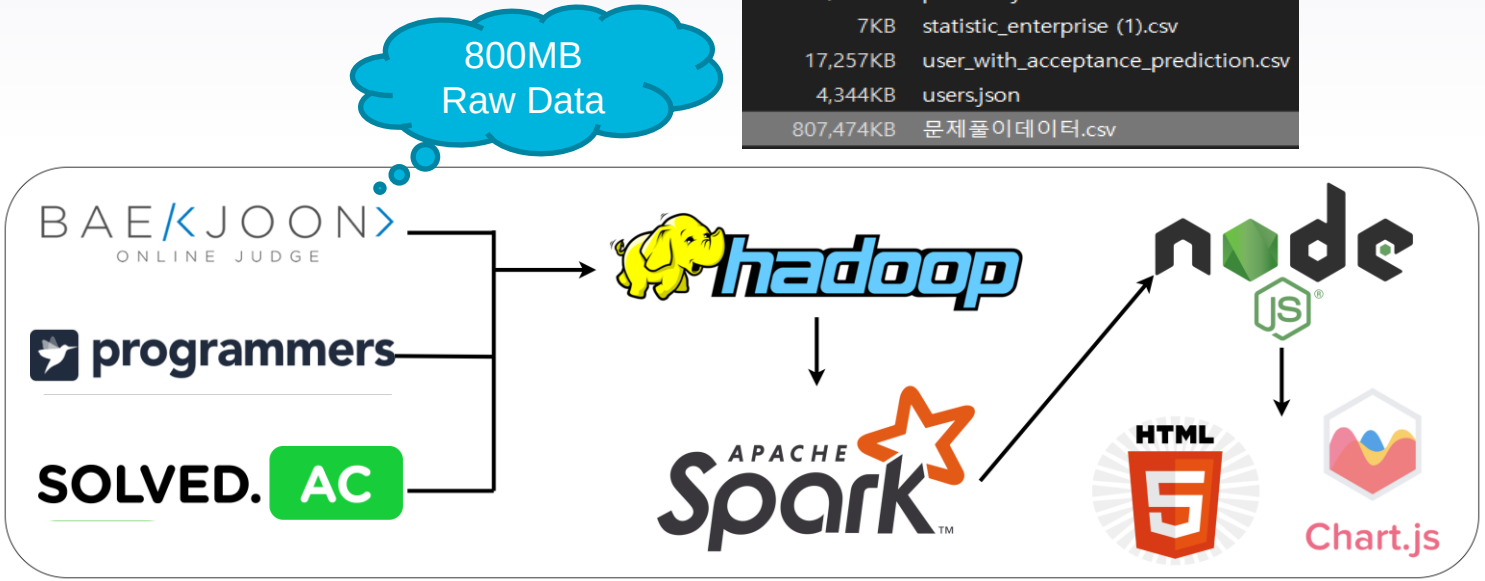
Goal : Show what types of problems do users need to solve more in order to go to a certain company

# Development Environment



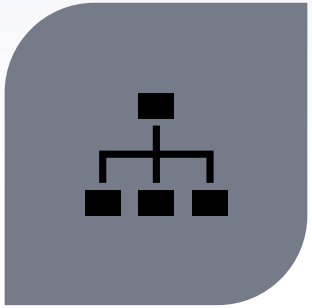
# Architecture

| | |
|---|---|
| 25KB | company.json |
| 1KB | company_top5_score_criteria.csv |
| 3,402KB | newformat_problem.json |
| 9,492KB | newformat_users.json |
| 2,043KB | problem.json |
| 7KB | statistic_enterprise (1).csv |
| 17,257KB | user_with_acceptance_prediction.csv |
| 4,344KB | users.json |
| 807,474KB | 문제풀이데이터.csv |



# Main Functions

WHAT TYPES OF PROBLEMS USERS HAVE SOLVED THE MOST



STUDY STRUCTURE WITH PEOPLE OF SIMILAR LEVEL

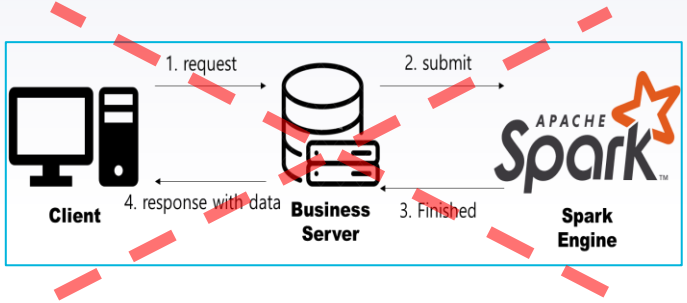
PASS PREDICTION

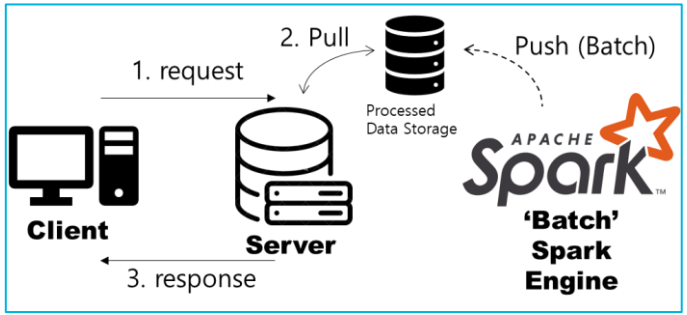
1

# Analysis

The Process, we Analyzed

# Batch Processing

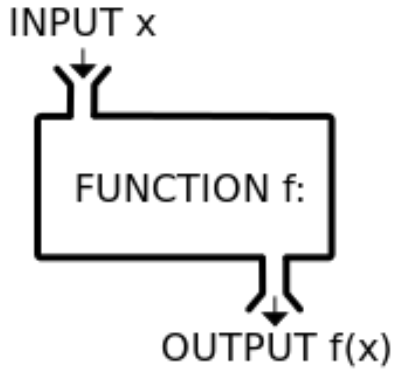



Why we took the batch processing?

because all needed output is

**deterministic!!!**



# RAW Data

user.json

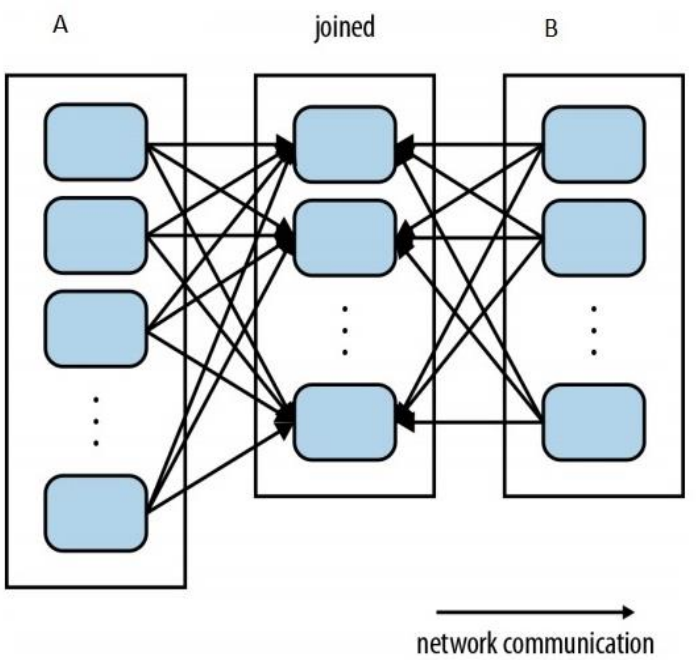
problem.json

| | userId | rank |
|---|---|---|
| type | String | Enum(RANK) |
| ex | abc123 | 마스터 |

| | questionId | title | difficulty | type |
|---|---|---|---|---|
| type | Integer | String | Enum(RANK) | Slash-Seperated-Value |
| ex | 1000 | A+B | 브론즈 5 | 수학/구현/사칙연산 |

company.json

| | company | title | difficulty | type |
|---|---|---|---|---|
| type | String | String | Enum(RANK) | Slash-Seperated-Value |
| ex | kakao | 추석 트래픽 | 실버 2 | 브루트포스/구현/자료구조 |




trials.csv

| | user | user_rank* | questionId | title | result | memory | time | language | volume | date |
|---|---|---|---|---|---|---|---|---|---|---|
| ex | abc123 | 마스터 | 1000 | A+B | 틀렸습니다!! | N/A | N/A | C++ 17 | 730 | 2022-05-14 00:00:00 PM |

# Monthly Top Algorithm Analysis

trial_df

| | user | type | … | result | date |
|---|---|---|---|---|---|
| ex | abc123 | 수학/구현/사칙연산 | | 틀렸습니다!! | 2022-05-14 00:00:00 PM |

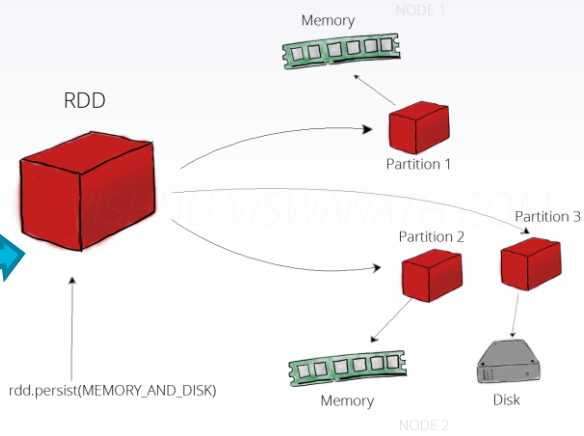
withColumn(“month, split(“date”, “-”)[1])
col(“type”).split(“/”).exploed(“type”)

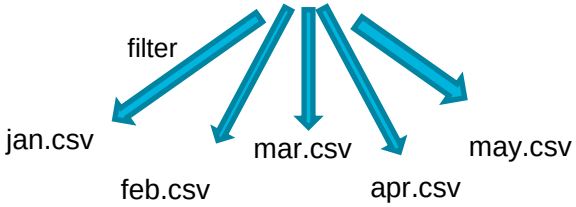
groupBy(“month”, “type”)

trial_per_type_df

| | month | type | … | count |
|---|---|---|---|---|
| ex | 02 | 수학 | | 510 |

trial_per_type_df
.**persist**(MEMORY_AND_DISK)

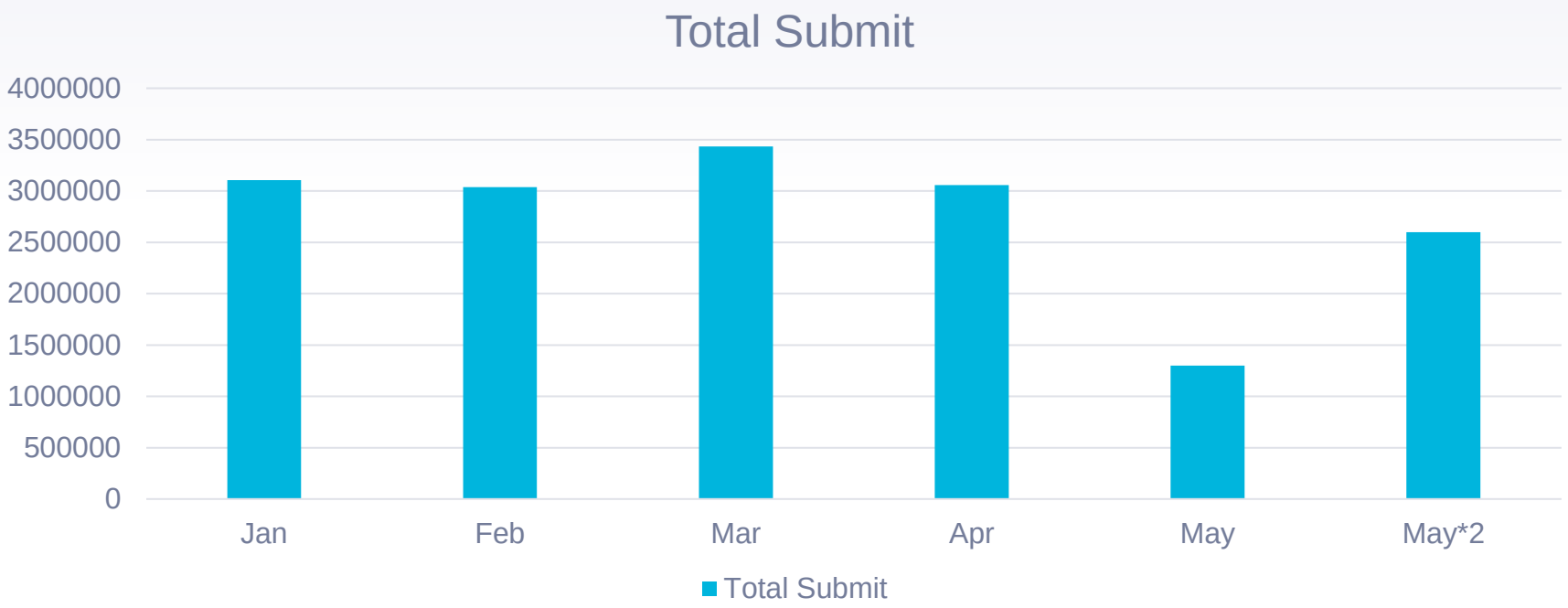


# Monthly Top Algorithm Analysis

| 순위 | 1월 | 제출합 | 2월 | 제출합 | 3월 | 제출합 | 4월 | 제출합 | 5월 | 제출합 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 구현 | 573055 | 구현 | 502249 | 구현 | 601114 | 구현 | 557489 | 구현 | 223313 |
| 2 | 수학 | 537963 | 수학 | 454039 | 수학 | 554561 | 수학 | 487943 | 수학 | 207147 |
| 3 | 사칙연산 | 275737 | 사칙연산 | 232013 | 사칙연산 | 305384 | 사칙연산 | 266794 | 사칙연산 | 108169 |
| 4 | 그래프 이론 | 132869 | 그래프 이론 | 157594 | 그래프 이론 | 182649 | 그래프 이론 | 163517 | 그래프 이론 | 62522 |
| 5 | 자료 구조 | 129680 | 자료 구조 | 137495 | 그래프 탐색 | 143376 | 그래프 탐색 | 129326 | 자료 구조 | 59980 |
| 6 | 문자열 | 126303 | 그래프 탐색 | 121482 | 다이나믹 프로그래밍 | 138367 | 자료 구조 | 112730 | 문자열 | 52726 |
| 7 | 다이나믹 프로그래밍 | 117649 | 다이나믹 프로그래밍 | 120728 | 자료 구조 | 137610 | 다이나믹 프로그래밍 | 108485 | 다이나믹 프로그래밍 | 48256 |
| 8 | 정렬 | 103045 | 문자열 | 116597 | 문자열 | 123693 | 문자열 | 108205 | 그래프 탐색 | 48046 |
| 9 | 그래프 탐색 | 101461 | 정렬 | 103484 | 너비 우선 탐색 | 114599 | 너비 우선 탐색 | 105267 | 정렬 | 44291 |
| 10 | 브루트포스 | 96642 | 브루트포스 | 99322 | 정렬 | 109605 | 브루트포스 | 101127 | 브루트포스 | 38536 |

# Monthly Top Algorithm Analysis

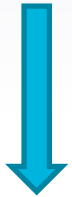


# Top Algorithm per user and tier group

trial_df

| | user | type | … | result | date |
|---|---|---|---|---|---|
| Ex) | abc123 | 수학/구현/사칙연산 | | 틀렸습니다!! | 2022-05-14 00:00:00 PM |

['맞았습니다!!', '맞았습니다!! (2/3점)',..] -> **correct**
['틀렸습니다', '메모리 초과', '시간 초과'] -> **wrong**
['런타임 에러(..)', '컴파일 에러(..)', '출력 오류 ..'] -> **error**
etc -> **etc**

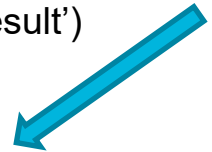


categorize_udf

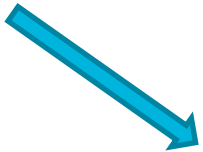
categorized_df

| | user | type | … | result | date |
|---|---|---|---|---|---|
| Ex) | abc123 | 수학 | | wrong | 2022-05-14 00:00:00 PM |

groupBy('user', 'type', 'result')
agg(count)
sort(count.desc)



correct_stat_user.csv



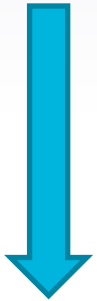
groupBy('user_rank', 'type', 'result')
agg(count)
sort(count.desc)

correct_stat_rankgroup.csv

# Top Algorithm per company

company_df

| | company | difficulty | title | type |
|---|---|---|---|---|
| Ex) | kakao | 실버 5 | 문자열 압축 | 문자열/구현 |



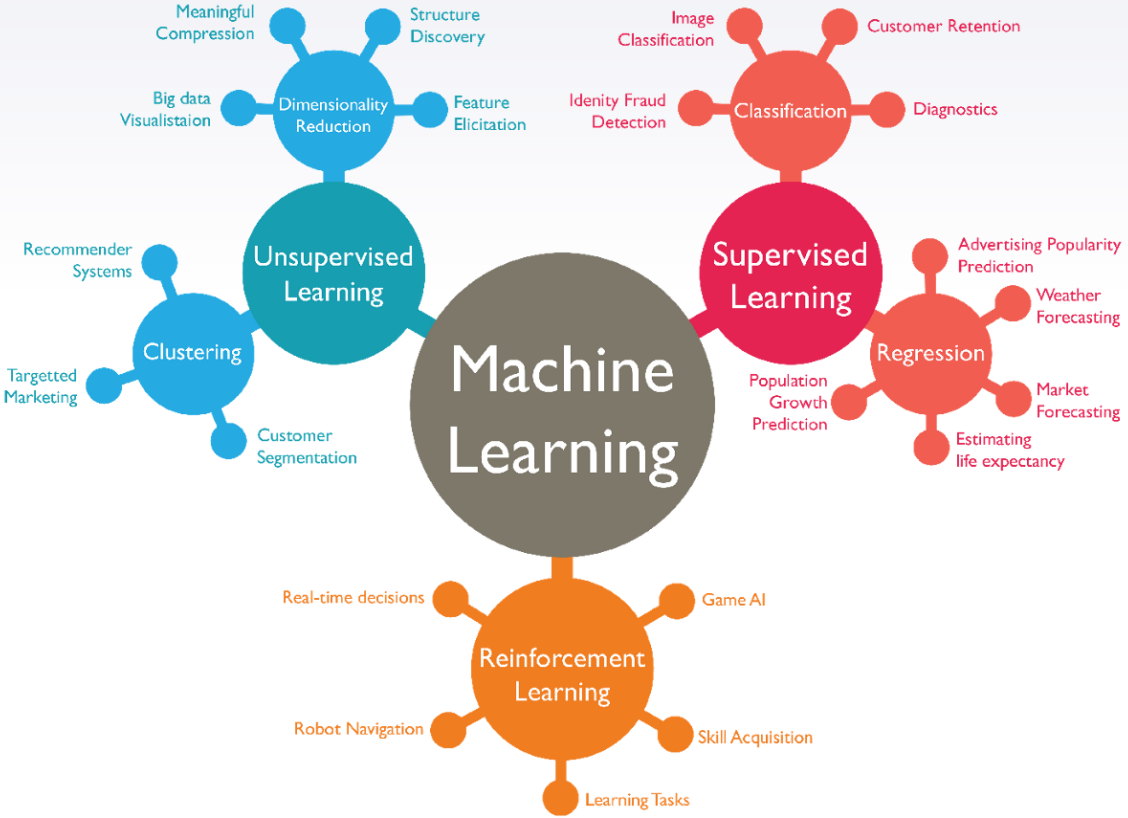
브론즈 5 = 1.0
브론즈 4 = 2.0
…
다이아몬드 2 = 24.0
다이아몬드 2 = 25.0

company_stat_df

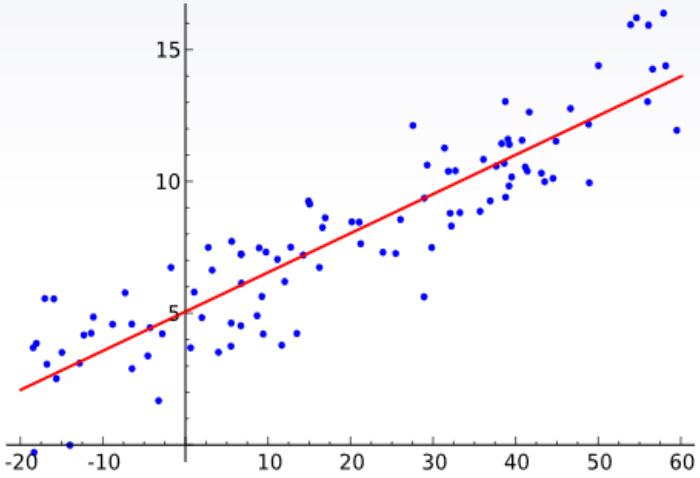
| | company | type | count | avg_of_difficulty |
|---|---|---|---|---|
| Ex) | coupang | 문자열 | 22 | 10.36 |

| 순위 | Naver | KaKao | Line | Coupang | Samsung |
|---|---|---|---|---|---|
| 1 | 문자열 | 구현 | 구현 | 구현 | 구현 |
| 2 | 구현 | 문자열 | 브루트 포스 | 자료 구조 | 시뮬레이션 |
| 3 | 자료 구조 | 브루트 포스 | 문자열 | 문자열 | 브루트 포스 |
| 4 | 수학 | 그래프 이론 | 수학 | 수학 | 그래프 이론 |
| 5 | 그래프 탐색 | 다이나믹 프로그래밍 | 자료 구조 | 브루트 포스 | 그래프 탐색 |
| 6 | 그래프 이론 | 깊이 우선 탐색 | 그래프 탐색 | 백트래킹 | 백트래킹 |
| 7 | 해시 | 너비 우선 탐색 | 그래프 이론 | 두 포인터 | 자료 구조 |

# Predicting the probability of acceptance

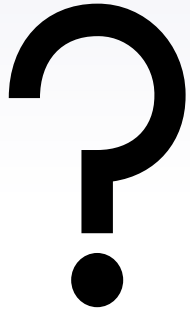


# Predicting the probability of acceptance




Typical regression

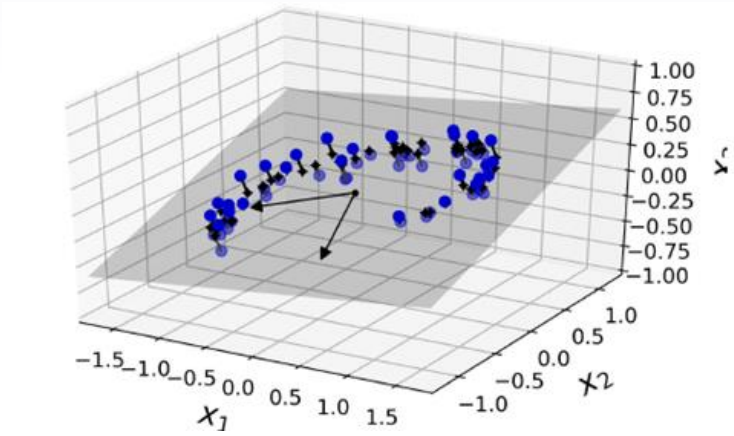
But we do not have **y** value

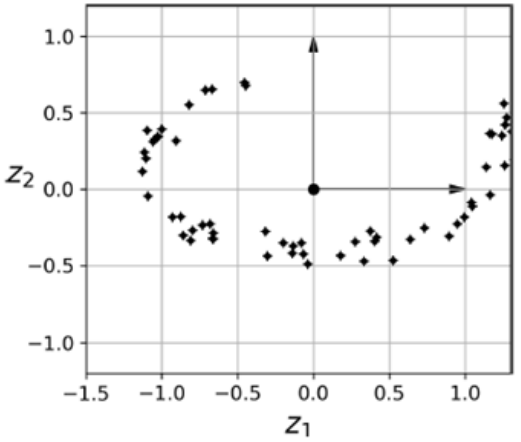


Let’s Create statistics-based Probabilistic Model

# Predicting the probability of acceptance

## Dimensionality Reduction



# Predicting the probability of acceptance

## Dimensionality Reduction – Feature Selection

| type | 다익스트라 | 다이나믹 프로그래밍 | 그래프 탐색 | 구현 | 파싱 | 수학 | 깊이 우선 탐색 | 너비 우선 탐색 | 그래프 이론 | 그리디 알고리즘 | 두 포인터 | 문자열 | 백트래킹 | 자료 구조 | 시뮬레이션 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| type | | | | | | | | | | | | | | | |
| 다익스트라 | 1.000000 | 0.988020 | 0.993974 | 0.920017 | 0.989428 | 0.945196 | 0.987721 | 0.992501 | 0.994437 | 0.990049 | 0.999600 | 0.957273 | 0.963842 | 0.992050 | 0.977442 |
| 다이나믹 프로그래밍 | 0.988020 | 1.000000 | 0.998983 | 0.969471 | 0.999956 | 0.984260 | 0.999998 | 0.999475 | 0.998779 | 0.999905 | 0.991992 | 0.990433 | 0.993419 | 0.999586 | 0.998326 |
| 그래프 탐색 | 0.993974 | 0.998983 | 1.000000 | 0.957429 | 0.999363 | 0.975290 | 0.998894 | 0.999919 | 0.999991 | 0.999508 | 0.996678 | 0.983204 | 0.987244 | 0.999867 | 0.994703 |
| 구현 | 0.920017 | 0.969471 | 0.957429 | 1.000000 | 0.967122 | 0.997546 | 0.969942 | 0.961020 | 0.956176 | 0.966009 | 0.930737 | 0.994033 | 0.991177 | 0.962017 | 0.982029 |
| 파싱 | 0.989428 | 0.999956 | 0.999363 | 0.967122 | 1.000000 | 0.982554 | 0.999936 | 0.999736 | 0.999200 | 0.999991 | 0.993136 | 0.989091 | 0.992297 | 0.999813 | 0.997738 |
| 수학 | 0.945196 | 0.984260 | 0.975290 | 0.997546 | 0.982554 | 1.000000 | 0.984598 | 0.978019 | 0.974329 | 0.981737 | 0.954056 | 0.999231 | 0.998024 | 0.978769 | 0.992833 |
| 깊이 우선 탐색 | 0.987721 | 0.999998 | 0.998894 | 0.969942 | 0.999936 | 0.984598 | 1.000000 | 0.999411 | 0.998682 | 0.999877 | 0.991747 | 0.990697 | 0.993637 | 0.999529 | 0.998436 |

# Predicting the probability of acceptance

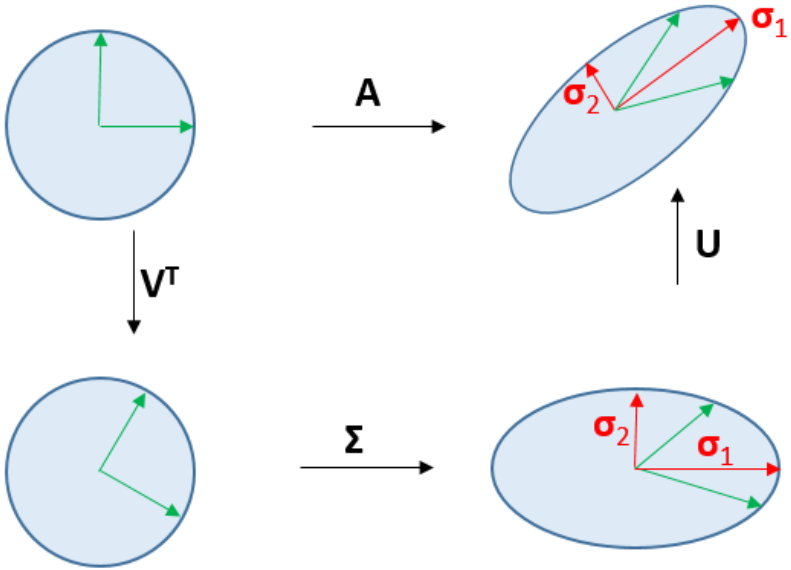
## Dimensionality Reduction – Feature Selection

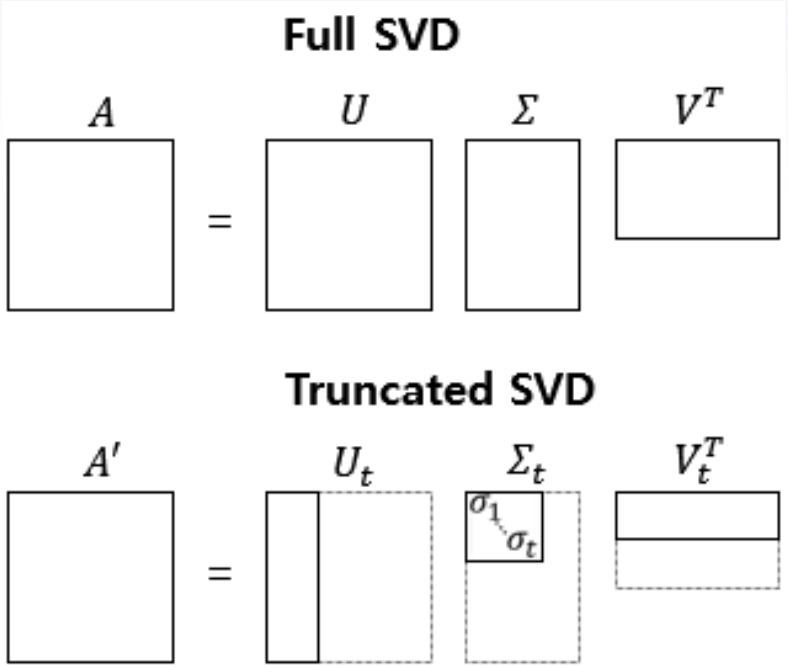
Select the correct ratio of each company's Top 5 algorithm

| user | 구현 | 브루트포스 알고리즘 | 문자열 | 자료 구조 | 수학 |
|---|---|---|---|---|---|
| 1223hyo | 0.51428574 | 0.0 | 0.0 | 0.0 | 0.5 |
| 1998phy | 0.75 | 0.5 | 0.5 | 0.25 | 0.45454547 |
| 1dilumn0 | 0.5652174 | 0.37209302 | 0.39130434 | 0.39215687 | 0.6 |
| 1x2x257 | 0.0 | 0.0 | 0.0 | 0.0 | 0.8333333 |
| 20201785 | 0.71428573 | 1.0 | 0.0 | 0.0 | 0.5 |
| 2david2 | 0.80487806 | 0.0 | 0.54545456 | 0.0 | 0.61764705 |
| 4vm89092n7890 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 |
| 5gkfka5 | 0.8333333 | 0.33333334 | 0.0 | 0.26666668 | 0.6 |
| aa4060 | 0.33333334 | 0.0 | 0.0 | 0.0 | 0.3 |
| aaabbb4202 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| abyoon07 | 0.66035496 | 0.6666667 | 0.71428573 | 0.7647059 | 0.7129713 |

# Predicting the probability of acceptance

Dimensionality Reduction – Truncated Singular Value Decomposition (5D -> 2D)



# Predicting the probability of acceptance

## Dimensionality Reduction – Truncated Singular Value Decomposition (5D -> 2D)

```
from pyspark.mllib.linalg import Vectors
from pyspark.mllib.linalg.distributed import RowMatrix, DenseMatrix
from pyspark.ml.feature import VectorAssembler
```

```
mat = RowMatrix(temp_rdd)
svd = mat.computeSVD(2, computeU=True)
```

(m=NumOfUser, n=5(top5 correct))

| user | 구현 | 브루트포스 알고리즘 | 문자열 | 자료 구조 | 수학 |
|---|---|---|---|---|---|
| 1223hyo | 0.51428574 | 0.0 | 0.0 | 0.0 | 0.5 |
| 1998phy | 0.75 | 0.5 | 0.5 | 0.25 | 0.45454547 |
| 1dilumn0 | 0.5652174 | 0.37209302 | 0.39130434 | 0.39215687 | 0.6 |
| 1x2x257 | 0.0 | 0.0 | 0.0 | 0.0 | 0.8333333 |
| 20201785 | 0.71428573 | 1.0 | 0.0 | 0.0 | 0.5 |
| 2david2 | 0.80487806 | 0.0 | 0.54545456 | 0.0 | 0.61764705 |
| 4vm89092n7890 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 |
| 5gkfka5 | 0.8333333 | 0.33333334 | 0.0 | 0.26666668 | 0.6 |
| aa4060 | 0.33333334 | 0.0 | 0.0 | 0.0 | 0.3 |
| aaabbb4202 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| abyoon07 | 0.66035486 | 0.666667 | 0.71428573 | 0.76470 | 0.71287 |

Truncated SVD
using Spark MLlib

(m=NumOfUser, t=2(score1, score2))

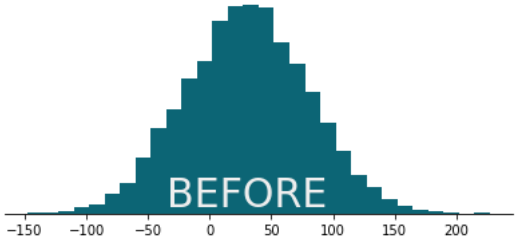
| vectored_score1 | vectored_score2 |
|---|---|
| [-0.0041346723130510875] | [-1.8365840019368479E-4] |
| [-0.004060063896791902] | [-4.608305899533307E-4] |
| [-0.004966066530773475] | [-9.107137065034617E-4] |
| [0.0] | [0.0] |
| [-0.0026479857131488493] | [0.004824286092871272] |
| [-0.003262813488919546] | [0.005989079767576516] |
| [-0.0035749415369504337] | [0.006560411456178430 5] |
| [-0.0036245078136471 84] | [0.004064016054085769] |
| [-0.005084874603735277] | [-0.003433330878991991] |
| [-0.005174017866012666] | [-0.002254957141646251] |

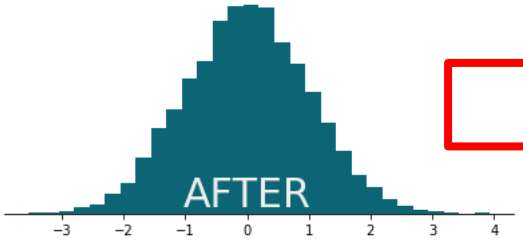
# Predicting the probability of acceptance

## Standardization

```
from pyspark.ml.feature import StandardScaler
```

$$z = \frac{x - \mu}{\sigma}$$

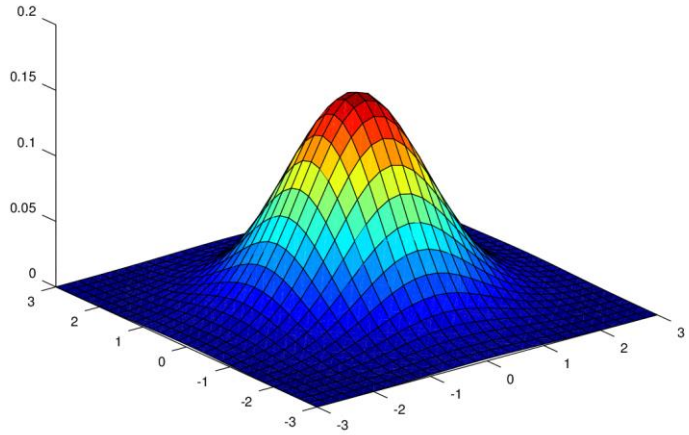


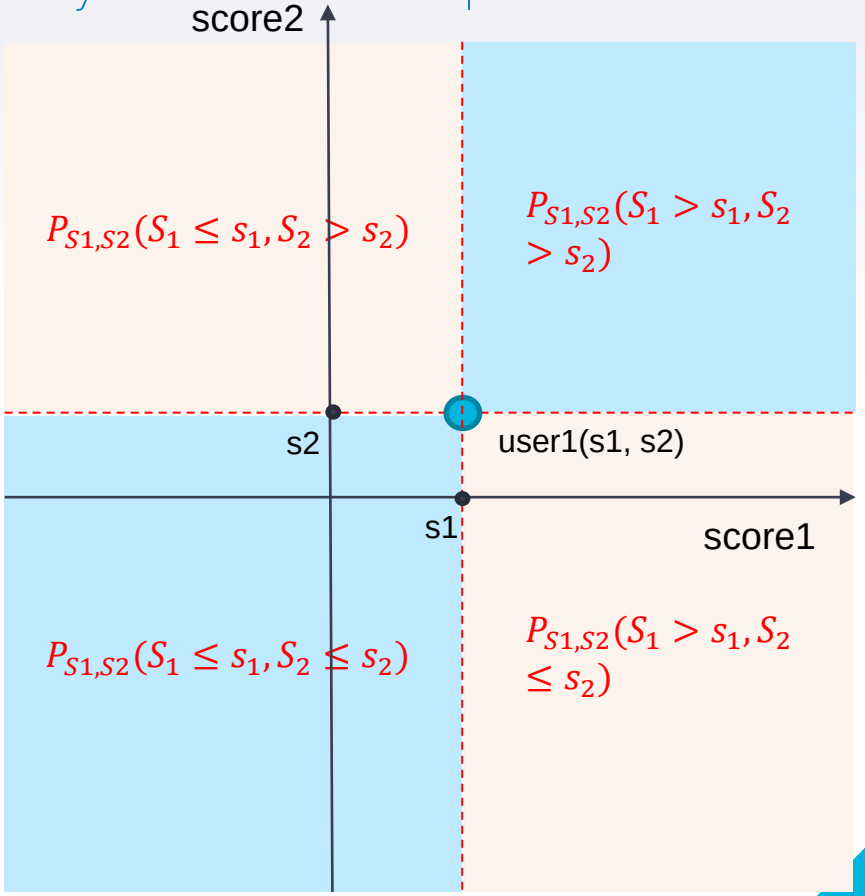

| summary | scaled_score1 | scaled_score2 |
|---|---|---|
| nt | 64337 | 64337 |
| n | 2.6876311306137O7E-9 | -8.237840748820714E-10 |
| dev | 1.00000000811591 | 0.9999999997615333 |
| | -2.6177993 | -3.6584785 |
| | 2.1589222 | 1.8754575 |

# Predicting the probability of acceptance

## Bivariate Standard Normal Distribution




$$P(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{z}{2(1-\rho^2)}\right],$$



# Predicting the probability of acceptance

## Binomial Distribution Formula

If Company C have n applicants, and choose (n-k) applicants

The probability that k applicants are inferior to User A and (n-k) are superior to User B

$$B(n,p) = \binom{n}{k} p^k (1-p)^{n-k}$$

Then we can simply estimate the User A's probability of acceptance for Company C

$$P(Acceptance) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

# Predicting the probability of acceptance

## Bivariate Standard Normal Distribution

$$Score2 - s_2 = -(Score1 - s_1)$$

$$Score2 = -Score1 + (s_1 + s_2)$$

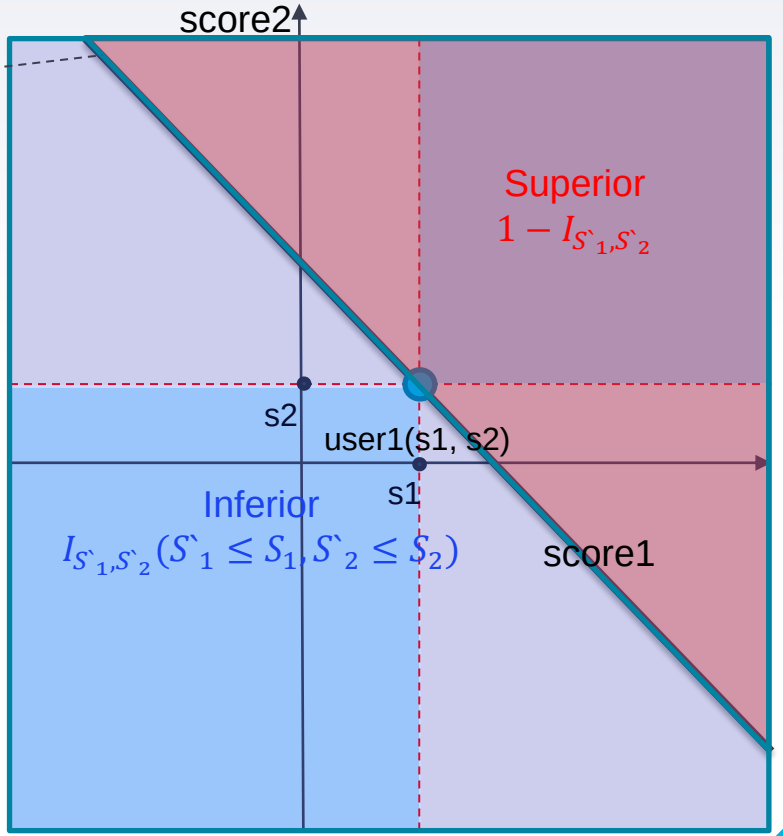
$$Score1 + Score1 = (s_1 + s_2)$$

```
test_df = stat_df.withColumn("Z", F.col("scaled_score1") + F.col("scaled_score2")).select("Z").describe()
```

```
+-------+-------------------+
|summary|                  Z|
+-------+-------------------+
|  count|              64337|
|   mean|4.713122354412475E-9|
| stddev|  1.6222895088638696|
+-------+-------------------+
```

$$Score1 + Score2 \leq (s_1 + s_2)$$

$$Assume\ Z = Score1 + Score2\ and\ Score1, Score2\ are\ independent$$
$$than\ Z\ is\ also\ Normal\ distribution\ with\ \sqrt{1^2 + 1^2} stddev$$



# Predicting the probability of acceptance

## Calc each user's Probability of acceptance

$$P(Acceptance) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i} \ , p = N(score1 + score2; 0, \sqrt{2})$$

```
from scipy.stats import norm
square_root_two = 2 ** (1/2)
inferior_cdf_udf = F.udf(lambda x, y: norm.cdf((x+y) / square_root_two).tolist(), FloatType())
```

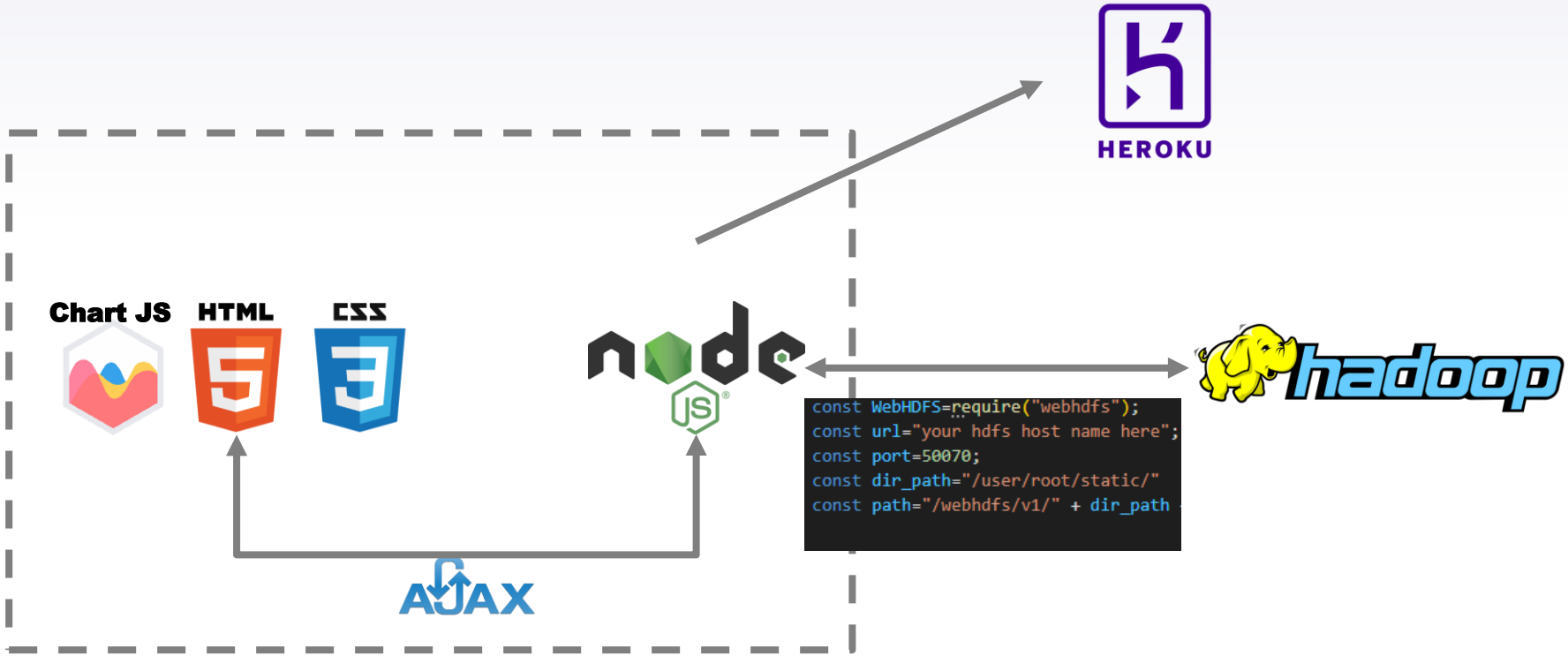
| user | scaled_score1 | scaled_score2 | inferior | superior | acceptance_probability_line |
|---|---|---|---|---|---|
| beth_shan | -0.33633533 | -0.19340631 | 0.35398498 | 0.64601505 | 7.9419286E-16 |
| black_203 | -0.29130945 | -0.26445845 | 0.34716445 | 0.65283555 | 2.3850674E-16 |
| dongkum0417 | -0.83807826 | -0.3797844 | 0.19457525 | 0.80542475 | 5.6589176E-33 |
| bokunoeiyuu20 | 2.1589222 | -0.14632612 | 0.9226494 | 0.07735062 | 1.0 |
| ty0603ty | 0.5608738 | 1.0903629 | 0.8785161 | 0.12148392 | 0.99987984 |
| dozinguy | 0.18982783 | 1.3889537 | 0.86786747 | 0.13213253 | 0.999541 |
| dntlr03 | 0.0014598591 | 1.5354125 | 0.86142254 | 0.13857746 | 0.9990533 |
| ech913 | -0.028453182 | 0.89547026 | 0.7300862 | 0.2699138 | 0.37440872 |
| byj9402 | -0.9097784 | -1.0264486 | 0.085481 | 0.914519 | 0.0 |
| tiaeld | -0.963576 | -0.72437656 | 0.11632454 | 0.88367546 | 0.0 |

2

# Implementation

The Process, we Visualized

# Visualization Architecture

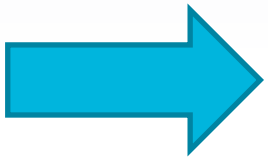


# BackEnd - Trend for Month

Mon, Type, Try

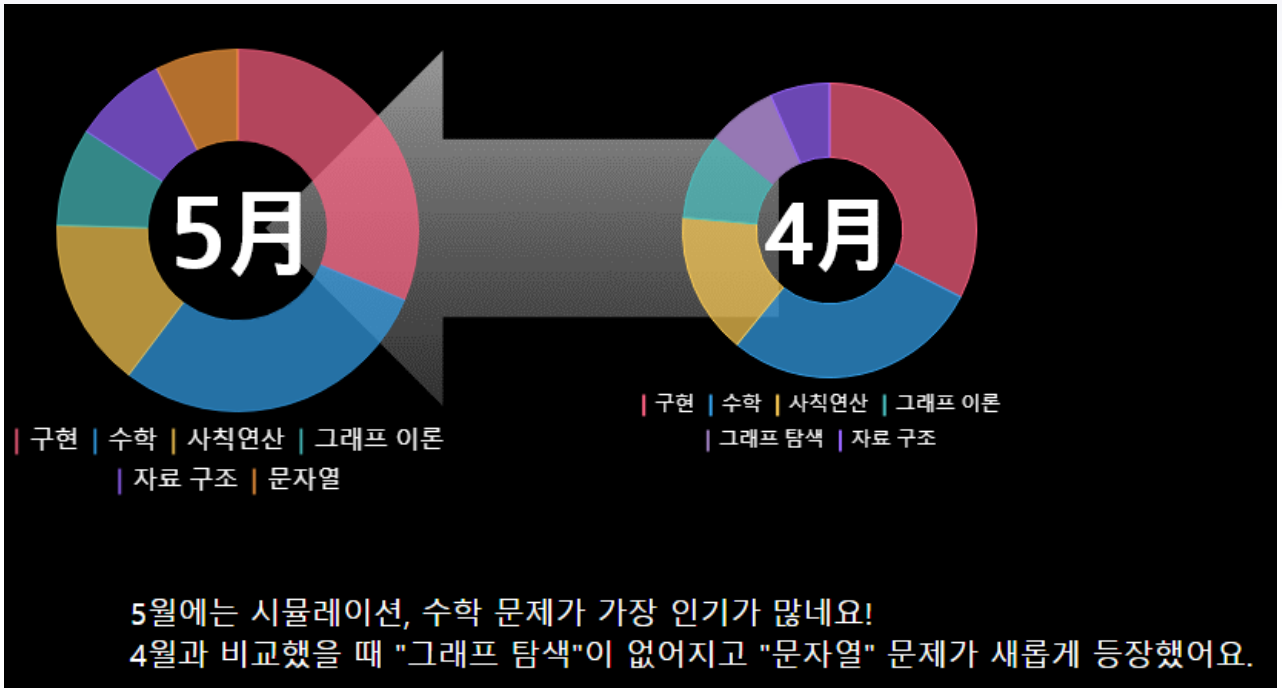
```
05,구현,223313
05,수학,207147
05,사칙연산,108169
05,그래프 이론,62522
05,자료 구조,59980
05,문자열,52726
05,다이나믹 프로그래밍,48256
05,그래프 탐색,48046
05,정렬,44291
05,브루트포스 알고리즘,38536
05,너비 우선 탐색,36830
05,그리디 알고리즘,30367
```

```
Client
GET /trend
data : {
    "cur": "CURRENT MONTH",
    "bef": "BEFORE MONTH"
}
```



```
Server
response
[
    {
        "type":[
            "구현",
            "수학",
            "사칙연산",
            "그래프 이론",
            "자료 구조",
            "문자열"
            ]
    },
    {
        "type":[
            "구현",
            "수학",
            "사칙연산",
            "그래프 이론",
            "그래프 탐색",
            "자료 구조"
            ]
    }

]
```

# FrontEnd - Trend for Month



# BackEnd – Analysis for User

```
501778  mjoo1106,골드 2,구성적,1,1,0,1.0
501779  mjoo1106,골드 2,두 포인터,6,5,1,0.8333333333333334
501780  mjoo1106,골드 2,비트마스킹,3,2,1,0.6666666666666666
501781  mjoo1106,골드 2,게임 이론,5,3,2,0.6
501782  mjoo1106,골드 2,배낭 문제,5,3,2,0.6
501783  mjoo1106,골드 2,누적 합,5,3,2,0.6
501784  mjoo1106,골드 2,시뮬레이션,17,10,7,0.5882352941176471
501785  mjoo1106,골드 2,문자열,29,17,12,0.586206896551724 1
501786  mjoo1106,골드 2,플로이드-와샬,7,4,3,0.5714285714285714
501787  mjoo1106,골드 2,너비 우선 탐색,52,28,24,0.5384615384615384
501788  mjoo1106,골드 2,백트래킹,19,10,9,0.5263157894736842
501789  mjoo1106,골드 2,구현,55,28,27,0.509090909090909
501790  mjoo1106,골드 2,분할 정복,10,5,5,0.5
501791  mjoo1106,골드 2,트리를 사용한 집합과 맵,4,2,2,0.5
```

User_Stats

```
골드 2,선인장,1,1,0,1.0
골드 2,함수 개형을 이용한 최적화,1,1,0,1.0
골드 2,로프,3,3,0,1.0
골드 2,2-sat,9,9,0,1.0
골드 2,오목 다각형 내부의 점 판정,1,1,0,1.0
골드 2,번사이드 보조정리,1,1,0,1.0
골드 2,최소 비용 최대 유량,6,6,0,1.0
골드 2,접미사 트리,1,1,0,1.0
골드 2,쌍대성,1,1,0,1.0
골드 2,z,4,3,1,0.75
골드 2,사칙연산,17032,12665,4367,0.7436002818224519
골드 2,폴라드 로,23,16,7,0.695652173913043 5
골드 2,순열 사이클 분할,118,79,39,0.6694915254237288
골드 2,볼록 껍질을 이용한 최적화,9,6,3,0.6666666666666666
```

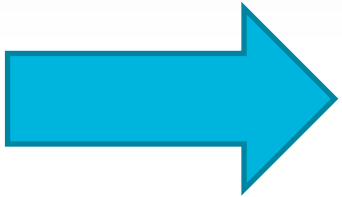
Group_Stats

```
for(var i=0; i<rows.length; i++){
    var temp=rows[i].split(",")
    if(temp[0]==req.query.name){
        user_rows.push(temp)
        total+=Number(temp[3])
        usum+=Number(temp[4])
    }
}

user_rows.sort(function(a,b){
    return Number(b[3])-Number(a[3])
});
```

```
for(var i=0; i<rows2.length; i++){
    var temp2=rows2[i].split(",")
    if(temp2[0]==tier){
        user_rows2.push(temp2)
        total2+=Number(temp2[2])
        usum2+=Number(temp2[3])
    }
}
apt.push(Math.round((usum2/total2)*100))
```

# BackEnd – Analysis for User

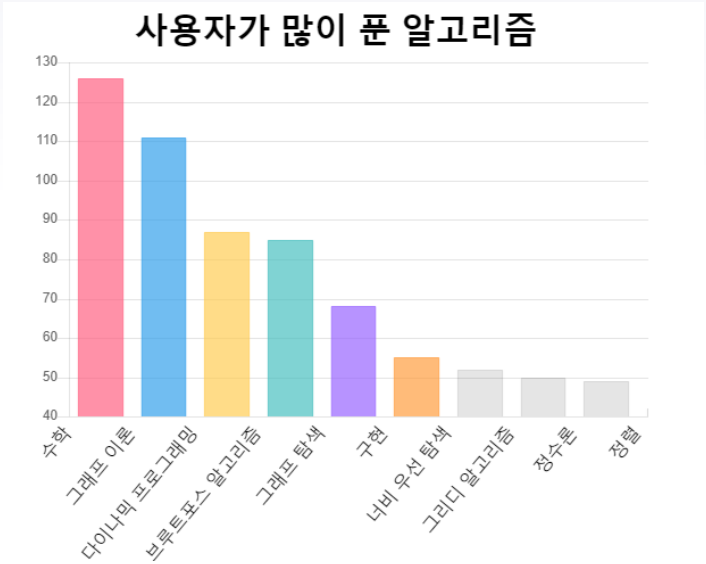
```
Client
GET /analysis
data : {
    "name":"user id"
}
```

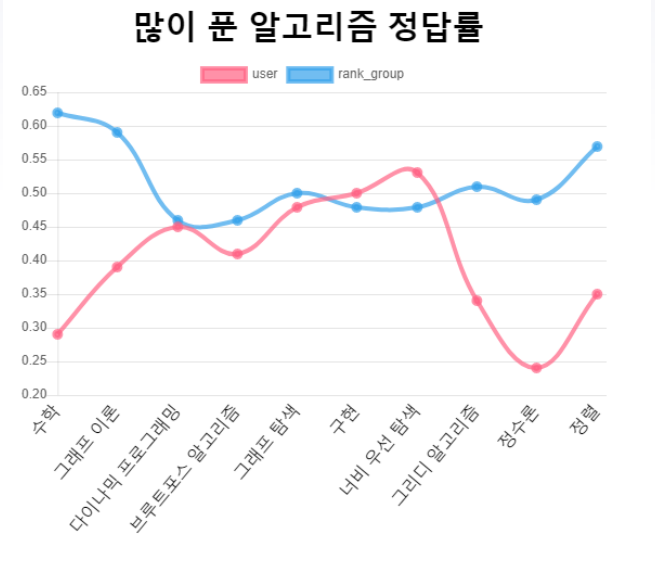


```
Server
Response
[
{
    "type": [
        "수학",
        …(생략) 10 data"
        ],
    "try":[
        "120",
        …(생략) 10 data
            ]
    "ans": [
        "0.35",
        …(생략) 10 data

},
{
    "type": [
        "구현",
        …(생략) 10 data"
        ],
    "try":[
        "85672",
        …(생략) 10 data
            ]
    "ans": [
        "0.65",
        …(생략) 10 data

},
```

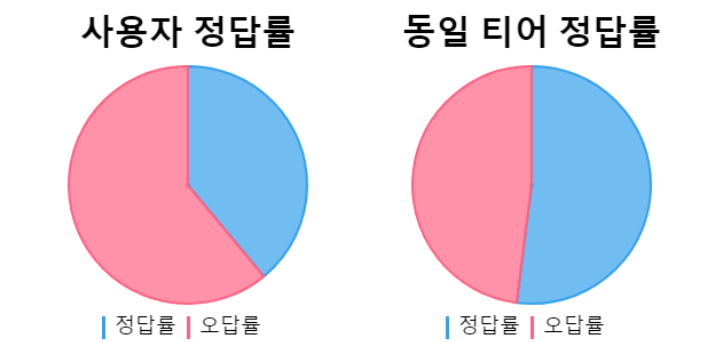
```
{
    "user_ans": "39",
    "tier_ans": "52",
    "diff": "-13"
    "weak": [
            "정확도"
        ],
    "strong":[
            "유형",
            "정답률"
        ]
},
```

# FrontEnd – Analysis for User



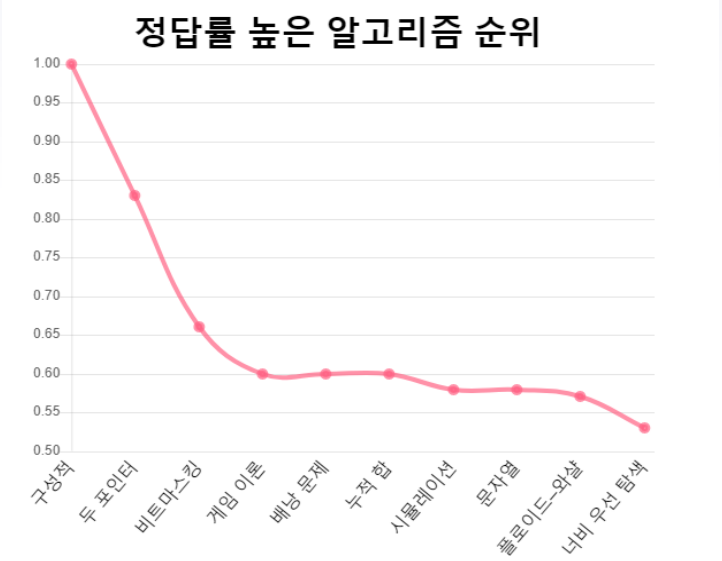



# FrontEnd – Analysis for User

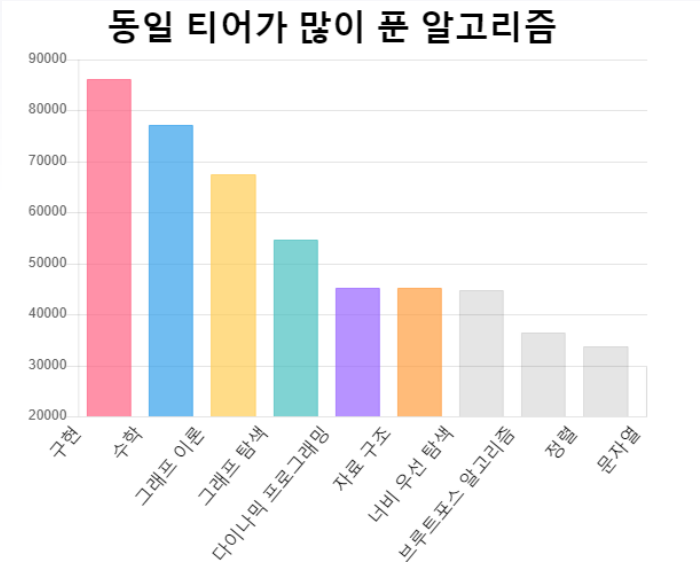



– 13%

**정확도가 필요할 시점입니다!**
**같은 수준의 사람들 보다 현재 정답률이 낮습니다.**



# FrontEnd – Analysis for User




주의

정확도가 동일 티어 보다 떨어집니다.
알고리즘 유형이 동일 티어와 유사합니다.
상위 알고리즘의 정답률이 높습니다.

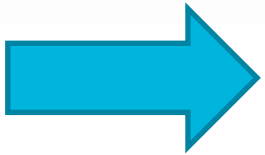
# BackEnd – Study Group

## Criteria

1. Same
2. Weak Algorithm

```
for(var i=0; i<rows.length; i++){
    var temp=rows[i].split(",")
    if(temp[1]==user_tier && temp[3]>=50){
        for(var j=0; j<5; j++)
            if(temp[2]==weak[j])
                arg.push(temp[0]);
    }
    if(arg.length>=5)
        break;
}
```

```
Client
GET /study
data : {
    "name":"user id"
}
```



```
Server
response
{
    "weak": ["자료구조", "완전탐색"],
    "level": "골드 2",
    "user_name": [
            "0321minji",
            "054679860",
            "0913vision",
            "0h328",
            "oxe82de"

}
```

# FrontEnd – Study Group

**mjoo1106**

**“알고리즘 능력”**


**“취약 알고리즘”**

**자료구조**

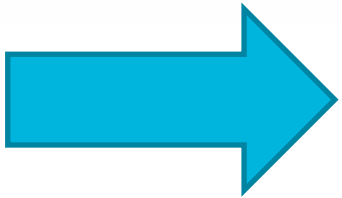
**완전탐색**

**클릭하면 자동으로 쪽지가 전송됩니다.**

- **0321minji**
- **054679860**
- **0913vision**
- **0h328**
- **0xe82de**

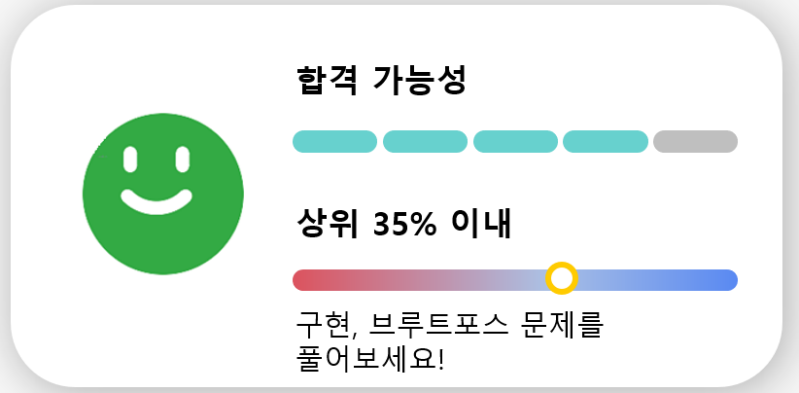
# BackEnd - Enterprise Acceptance Prediction Rate

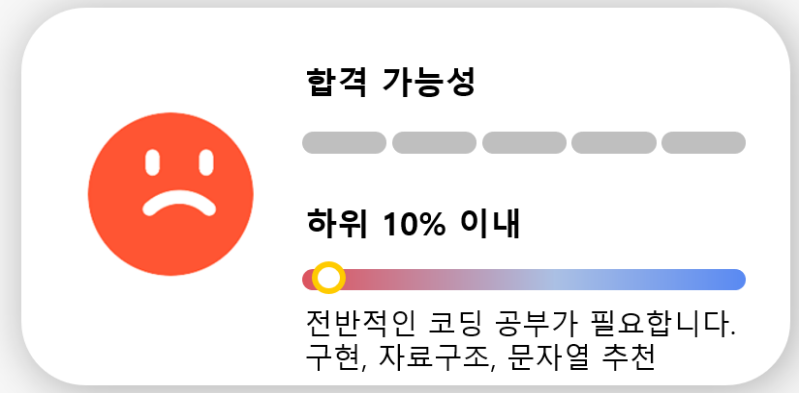
```
Client
GET /company
data : {
    “name”:“user id”,
    “company”:”회사”
}
```



```
Server
response

{
    “percent”:0.66,
    “weak”:[
        “구현“,
        “자료구조”
    ]
}
```

# FrontEnd - Enterprise Acceptance Prediction Rate



master origin/master origin/HEAD **Merge pull request #19 from philip-lee-khu/develop**

origin/develop Merge pull request #18 from philip-lee-khu/feature/captures

origin/feature/captures Update demo video & image

Merge pull request #17 from philip-lee-khu/feature/storage

origin/feature/storage Upload storage file from hdfs

Merge pull request #16 from philip-lee-khu/feature/webhdfs

origin/feature/webhdfs Change how files are read (csv to webhdfs)

Merge pull request #15 from philip-lee-khu/feature/acquistion

origin/feature/acquistion Acquistion big-data file

Merge pull request #14 from philip-lee-khu/feature/visualization

origin/feature/visualization Arrangement variables and images file

Create study group page

Create login page

Modify company analysis page

Modify same tier comparison graph

Merge pull request #13 from philip-lee-khu/feature/analysis/create_model_acceptance_prediction

origin/feature/analysis/create_model_acceptance_prediction Add notebook for predictinig company acceptance

Create user analysis page by char.js

Connect server & trend page

Merge pull request #11 from philip-lee-khu/feature/analysis/static_algo_enterprise

origin/feature/analysis/static_algo_enterprise Add notebook for generating stat each enterprise

Merge pull request #9 from philip-lee-khu/feature/analysis/statistic_algo_with_user_and_tiergroup

Create trend analysis page

origin/feature/analysis/statistic_algo_with_user_and_tiergroup Add notebook for stat correct rate to users and rankgroup

Create layout page & init server

Merge pull request #5 from philip-lee-khu/feature/analysis/monthly_top_algo

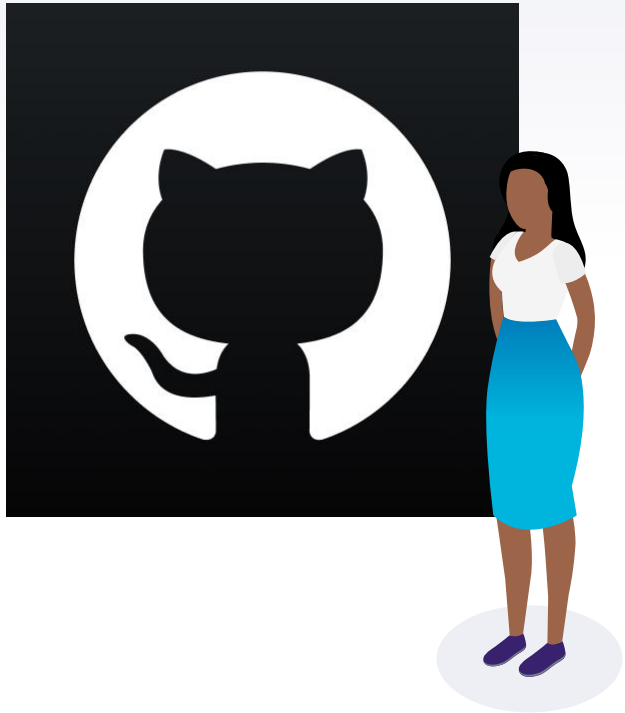
origin/feature/analysis/monthly_top_algo Add script for generating montly top algorithm

Append stop method to SparkContext

Add notebook for generating montly algorithm count

Update init folders

Initial commit



# https://bigdata-server.herokuapp.com/

You can Access with this Address!

# THANKS!

## Any questions?