DATA WAREHOUSE SYSTEM

1조 Final Presentation

2018321251 김용현
2018321260 권소현
2019311707 공태용

CONTENTS

1 Tool: Spark

2 Data Construct Process

3 Schema : Star Constellation

4 ETL

5 Information Package & Dimensional Modeling

6 Data Analysis

7 Future Work

1. Tool: SPARK

- **What is Spark?**
  - 빅데이터 애플리케이션 개발에 필요한 통합 플랫폼
  - 하드웨어 성능 향상 한계로 인한 병렬 처리 최적화 목적으로 등장
  - 데이터 ETL부터 SQL 처리, 머신 러닝 그리고 스트림 등 다양한 데이터 연산 엔진과 API 제공

- **Why is Spark?**
  - 오픈 소스 프레임워크: 다양한 API 제공
  - 소프트웨어 영역의 통합 플랫폼 제공: 파이썬, 자바 등
  - 컴퓨팅 능력 최적화: 하둡에 비해 처리 속도 빠름

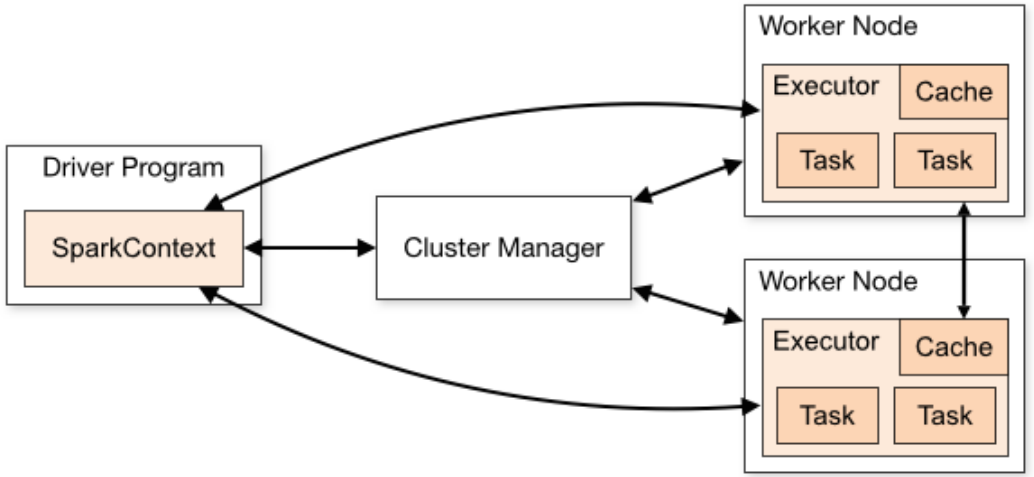
1. Tool: SPARK

- **사용 스파크 모드: Standalone Local Mode**
  - JVM 프로세스 기반
  - 별도의 데이터 저장소(하둡, 클라우드 등) 환경을 구축할 필요 없음
  - 모든 데이터 분석 프로세스가 하나의 환경에서 작동

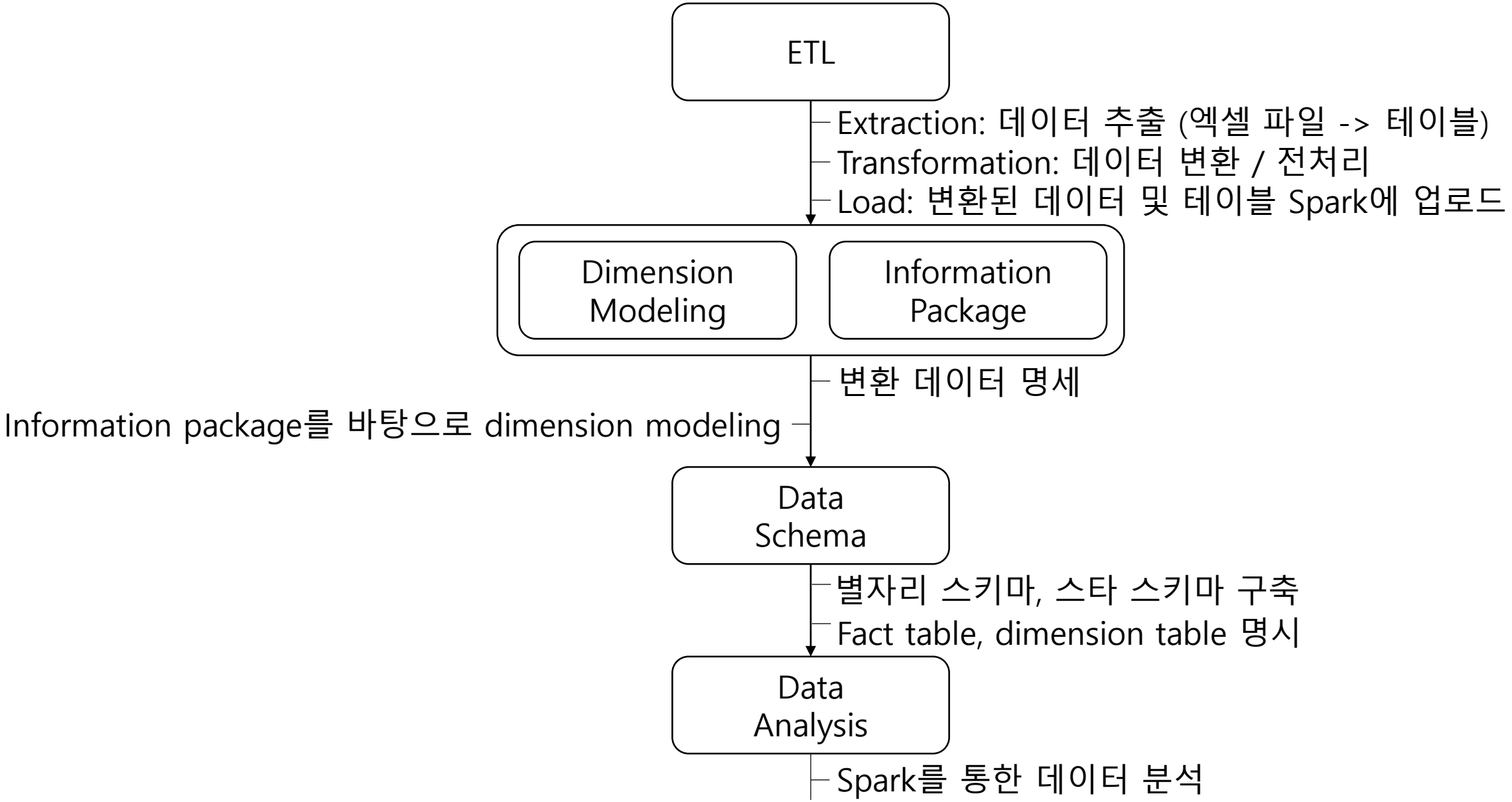
- **구축 환경**
  - JVM
  - PL language: Python
  - Library: PySpark

1. Tool: SPARK

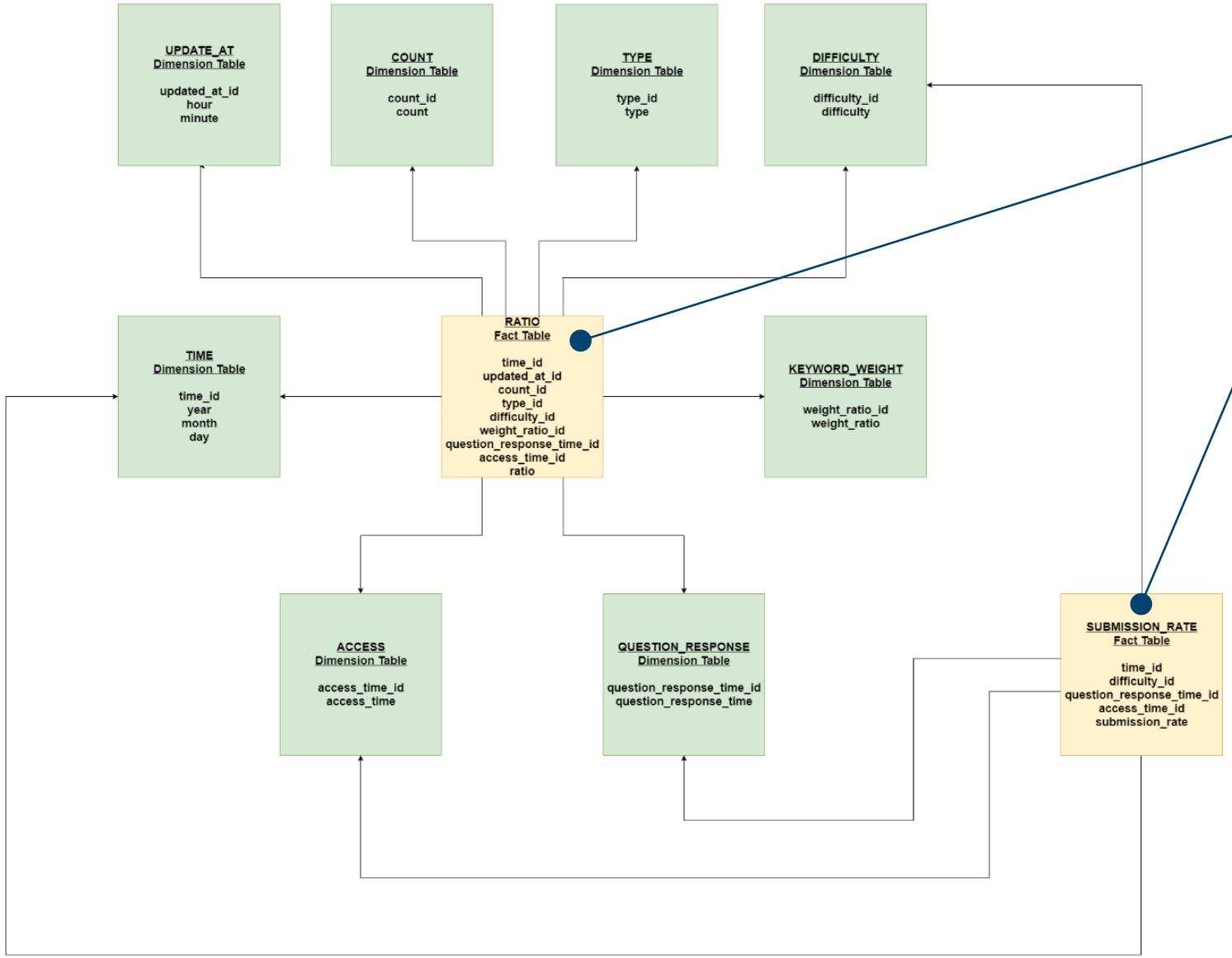
- **PySpark 장점**
  - Cluster Manager 역할
  - 로컬 저장소 기능 제공
  - 파티션 등 익스큐터 병렬 작업 지원
  - RDD 인터페이스를 이용하는 저수준 API 지원



2. Data Construct Process



3. Schema : Star Constellation



FACT 1. 문항 획득 점수로 보는
학생들의 학업 성취도 분석

FACT 2. 문항 제출 비율로 보는
학생들의 수업 참여도 분석

4. ETL

- Extraction
  - Preswot Data 사용
  - STUDENT_ANSWER_LOGS, QUESTIONS, LECTURE_KEYWORDS, LECTURE_ITEM_RESPONSE_LOGS, QUESTION_KEYWORDS

STUDENT_ANSWER_LOGS	
student_answer_log_id	학생 답안 로그 ID
ratio	점수 (1점 만점 환산)
count	제출 횟수
created_at	최초 제출 시간
updated_at	최종 제출 시간
lecture_id	소속 강의 ID
question_id	문항 ID
student_id	제출 학생 유저 ID

QUESTIONS	
question_id	문항 ID
type	문항 유형
difficulty	난이도

LECTURE_KEYWORDS	
lecture_id	소속 강의 ID
keyword	키워드
weight_ratio	강의 내 키워드 비중

LECTURE_ITEM_RESPONSE_LOGS	
log_id	로그 ID
student_id	학생 유저 ID
lecture_id	강의 ID
start_time	문항 시작/자료 열람 시작 시간
end_time	문항 제출/자료 열람 종료 시간

QUESTION_KEYWORDS	
question_id	문항 ID
keyword	키워드
lecture_id	소속 강의 ID

4. ETL

- Transformation
  - Time Dimension Table 생성을 위한 Transformation: STUDENT_ANSWER_LOGS Table의 student_answer_log_id와 updated_at에서 추출

STUDENT_ANSWER_LOGS	
student_answer_log_id	학생 답안 로그 ID
ratio	점수 (1점 만점 환산)
count	제출 횟수
created_at	최초 제출 시간
updated_at	최종 제출 시간
lecture_id	소속 강의 ID
question_id	문항 ID
student_id	제출 학생 유저 ID

TIME Dimension Table	
time_id(FK)	시간 로그 ID
year	연도
month	월
day	일

2019 - 04 - 03 14:12

```
root
 |-- time_id: integer (nullable = true)
 |-- year: integer (nullable = true)
 |-- month: integer (nullable = true)
 |-- day: integer (nullable = true)

+-------+----+-----+---+
|time_id|year|month|day|
+-------+----+-----+---+
|   3848|2019|    4|  3|
|   3849|2019|    4|  3|
|   3855|2019|    4|  3|
|   3856|2019|    4|  3|
|   3857|2019|    4|  3|
+-------+----+-----+---+
only showing top 5 rows
```

4. ETL

- Transformation
  - UPDATED_AT Dimension Table 생성을 위한 Transformation: STUDENT_ANSWER_LOGS Table의 student_answer_log_id와 updated_at에서 추출

STUDENT_ANSWER_LOGS	
student_answer_log_id	학생 답안 로그 ID
ratio	점수 (1점 만점 환산)
count	제출 횟수
created_at	최초 제출 시간
updated_at	최종 제출 시간
lecture_id	소속 강의 ID
question_id	문항 ID
student_id	제출 학생 유저 ID

UPDATED_AT Dimension Table	
updated_at_id(FK)	최종 제출 시간 ID
hour	시간
minute	분

2019-04-03 14 : 12

```
root
 |-- updated_at_id: integer (nullable = true)
 |-- hour: integer (nullable = true)
 |-- minute: integer (nullable = true)

+-------------+----+------+
|updated_at_id|hour|minute|
+-------------+----+------+
|         3848|  14|    11|
|         3849|  14|    12|
|         3855|  14|    12|
|         3856|  14|    12|
|         3857|  14|    12|
+-------------+----+------+
only showing top 5 rows
```

4. ETL

- Transformation
  - QUESTION_RESPONSE Dimension Table 생성을 위한 Transformation: LECTURE_ITEM_RESPONSE_LOGS Table의 log_id, start_time, end_time에서 추출

STUDENT_ANSWER_LOGS	
student_answer_log_id	학생 답안 로그 ID
ratio	점수 (1점 만점 환산)
count	제출 횟수
created_at	최초 제출 시간
updated_at	최종 제출 시간
lecture_id	소속 강의 ID
question_id	문항 ID
student_id	제출 학생 유저 ID

LECTURE_ITEM_RESPONSE_LOGS	
log_id	로그 ID
student_id	학생 유저 ID
lecture_id	강의 ID
start_time	문항 시작/자료 열람 시작 시간
end_time	문항 제출/자료 열람 종료 시간

QUESTION_RESPONSE Dimension Table	
question_response_time_id	문제 풀이 시간 ID
question_response_time	문제 풀이 시간

문항에 대한 시간만 분리
자료 열람 시간을 제외함

end_time – start_time

```
root
 |-- question_response_time_id: integer (nullable = true)
 |-- question_response_time: integer (nullable = true)

+------------------------+----------------------+
|question_response_time_id|question_response_time|
+------------------------+----------------------+
|                     206|                     1|
|                     207|                     0|
|                     208|                     0|
|                     209|                     0|
|                     210|                     0|
+------------------------+----------------------+
only showing top 5 rows
```

- Load: Spark 저장소에 target database 저장

5. Information Package & Dimensional Modeling

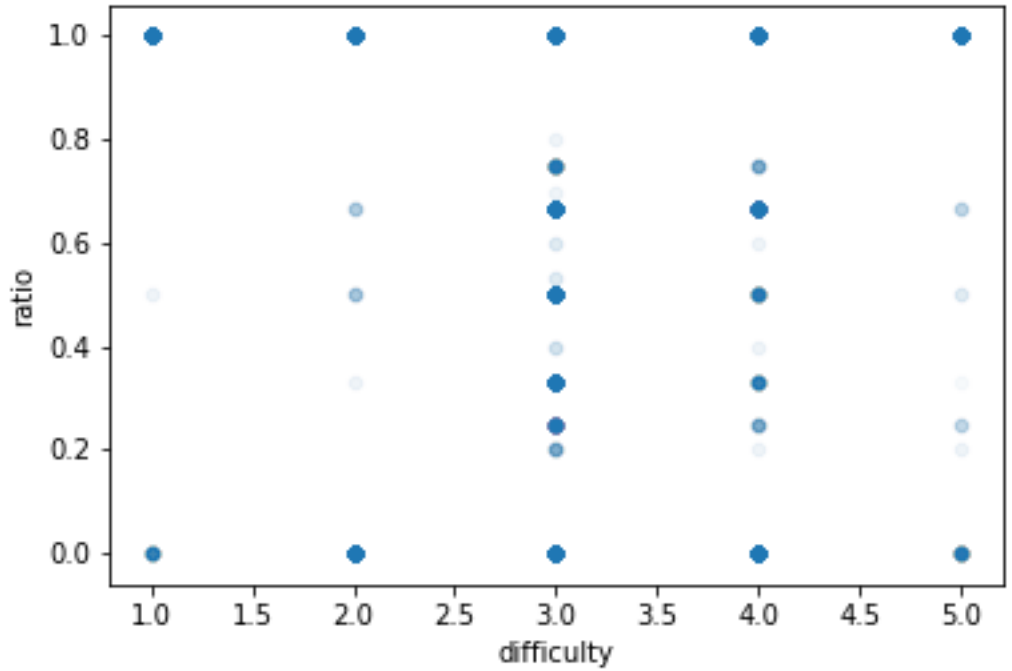
- 학생들의 학업 성취도 분석

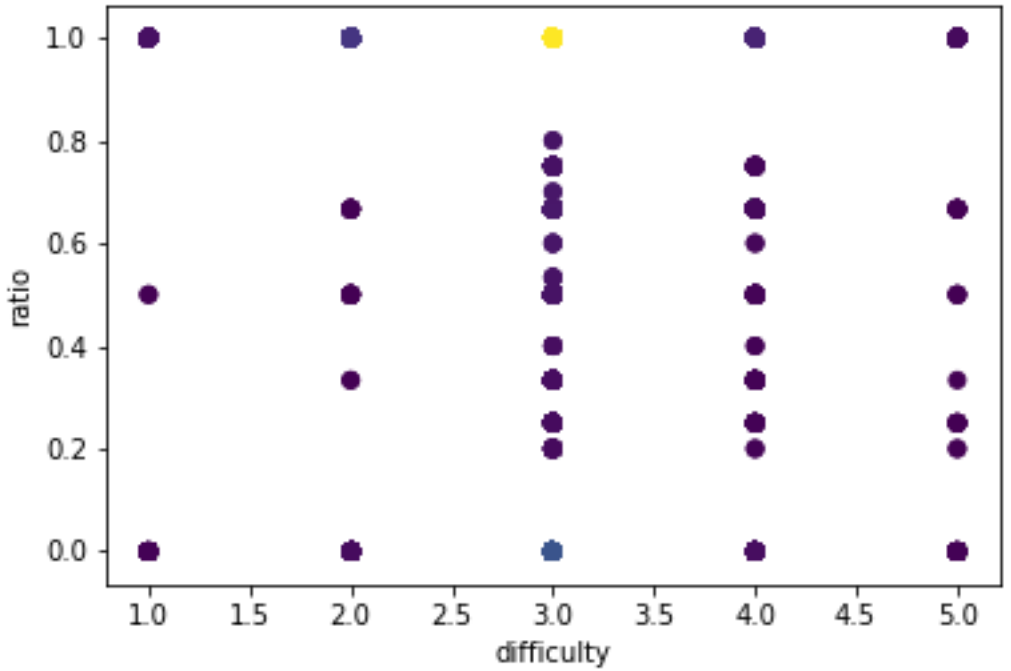
TIME	UPDATED_AT	COUNT	TYPE	DIFFICULTY	KEYWORD_WEIGHT	QUESTION_RESPONSE	ACCESS
time_id	updated_at_id	count_id	type_id	difficulty_id	weight_ratio_id	question_response_time_id	access_time_id
year	hour	count	type	difficulty	weight	question_response_time	access_time
month	minute						
day							
Fact : student question score ratio, student question submission rate							

- Dimensional Modeling
  - Step 1. Identify the business process
    - 학생들의 학업 성취도 분석
  - Step 2. Identify the grain
    - 날짜별, 제출 시간별, 제출 횟수별, 문항 유형별, 문항 난이도별, 키워드 비중별, 문항 응답 소요 시간별, 강좌 접속 여부별 문항 획득 점수 및 문항 제출 여부
  - Step 3. Identify the dimensions
    - TIME, UPDATED_AT, COUNT, TYPE, DIFFICULTY, KEYWORD_WEIGHT, QUESTION_RESPONSE, ACCESS
  - Step 4. Identify the fact
    - 날짜 별, 제출 시간별, 제출 횟수별, 문항 유형별, 문항 난이도별, 키워드 비중별, 문항 응답 소요 시간별, 강좌 접속 여부별 학생들의 문항 획득 점수 및 제출 비율

6. Data Analysis

- difficulty - ratio

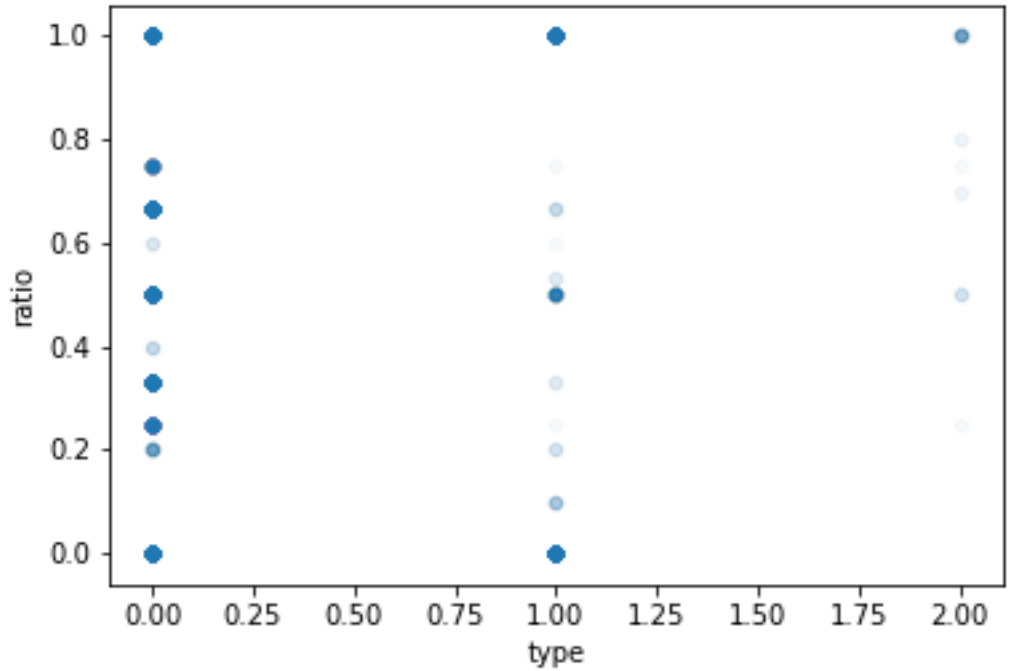


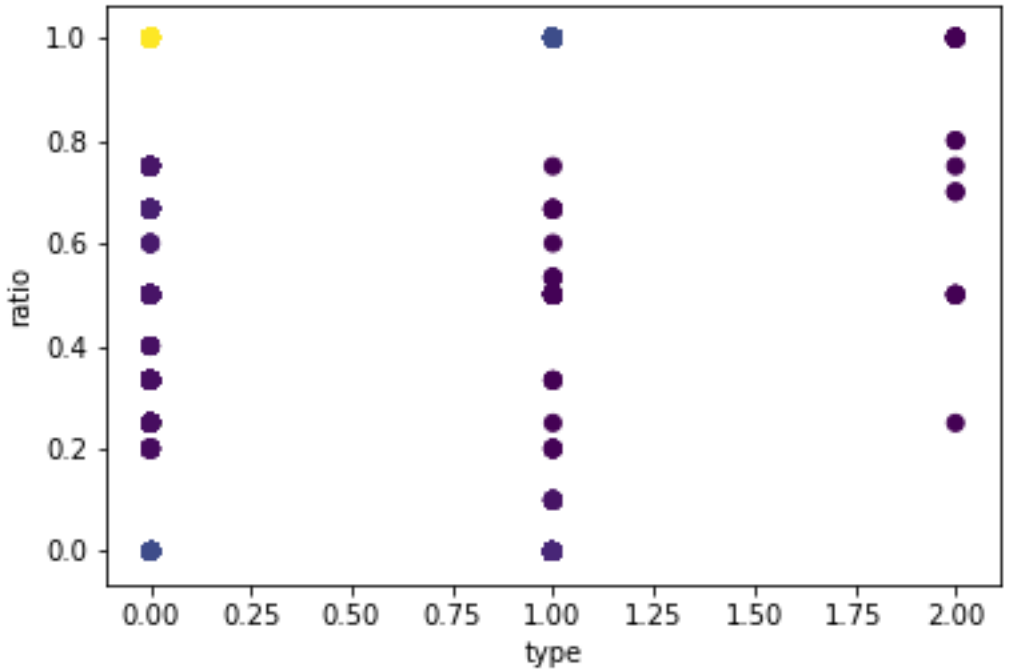


- 3점 문제의 정답률이 가장 높다.
- 5점 문제로 갈수록 정답률이 낮아지지만 데이터의 개수가 많지 않아 섣부른 판단은 이르다.

6. Data Analysis

- type - ratio

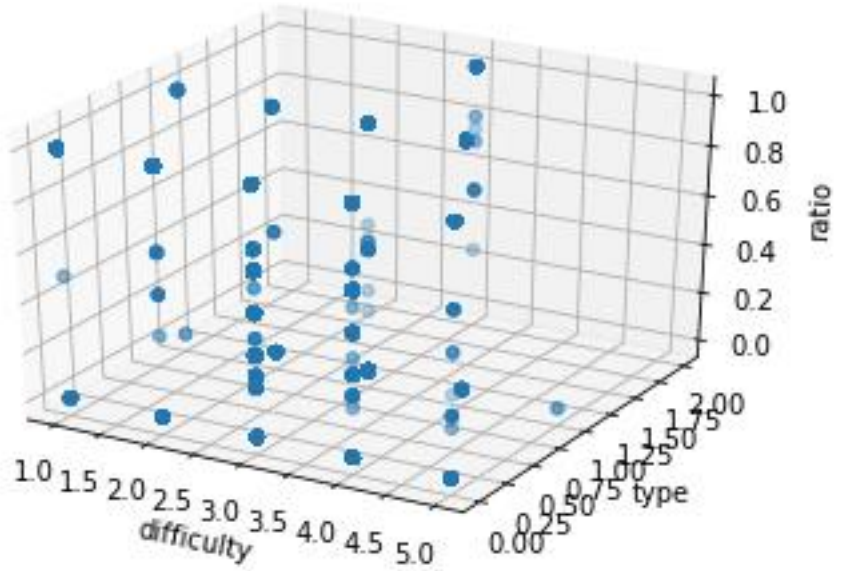


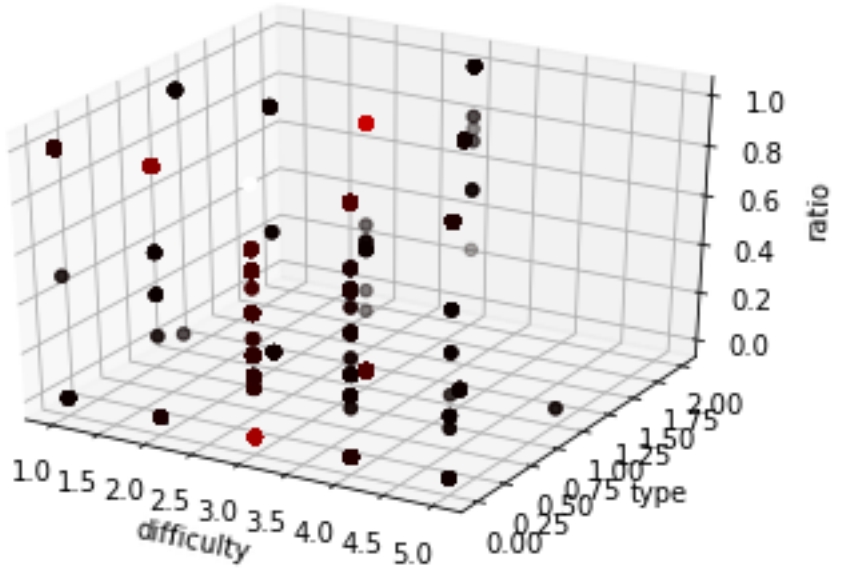


- 객관식 정답률이 가장 높다.
- 단답형 정답률은 고루 분포한다.
- 주관식 정답률은 주로 높다.

6. Data Analysis

- difficulty - type - ratio

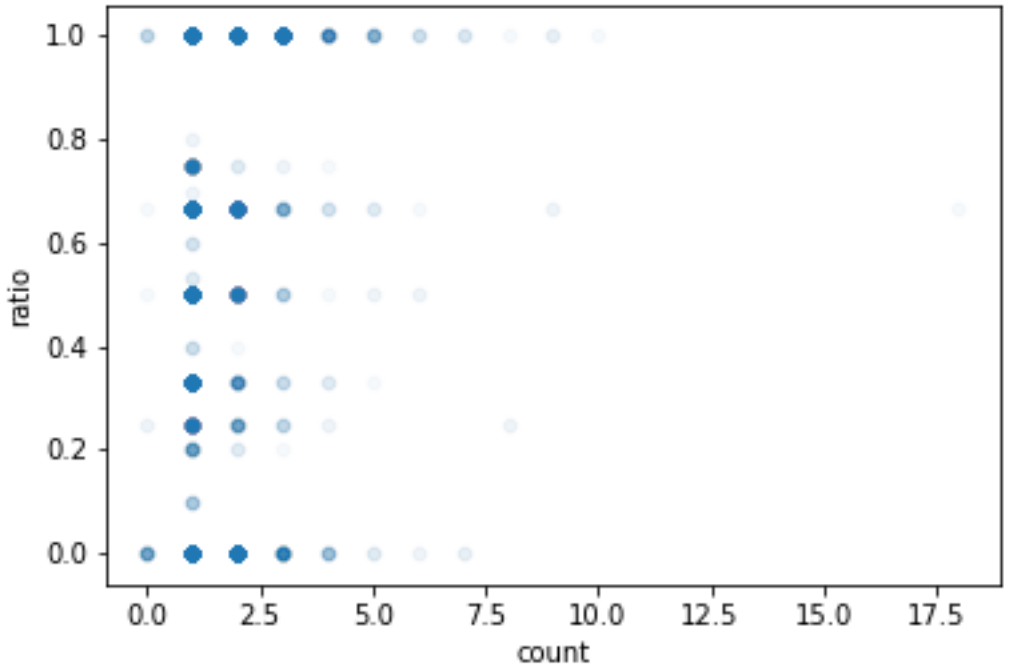


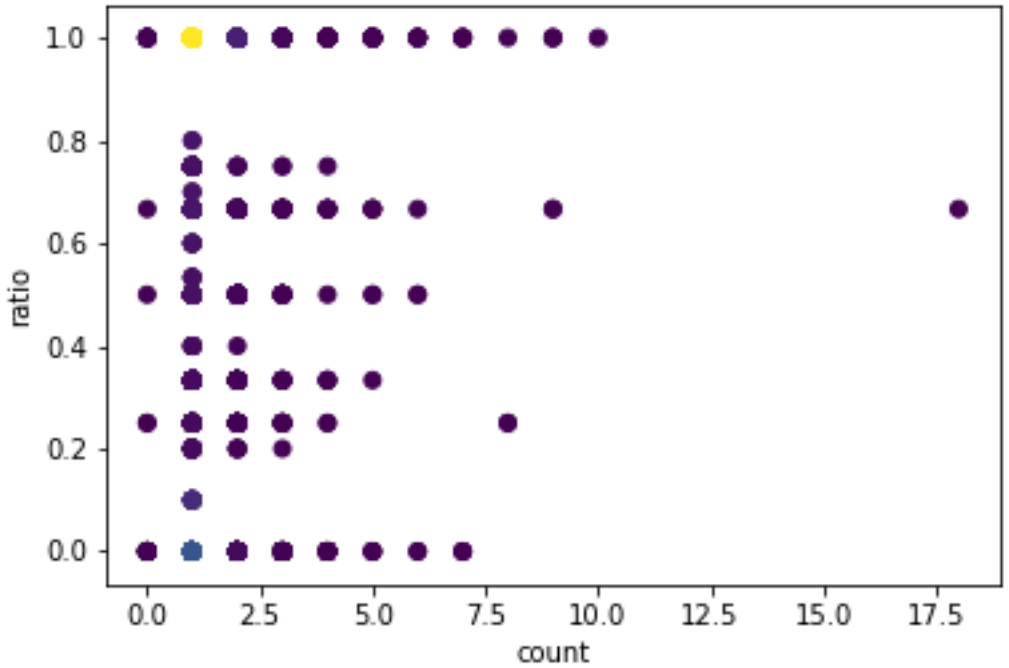


- 주관식에 쉬운 문제가 많다.
- 객관식에 쉬운 문제가 많다.

6. Data Analysis

- count - ratio

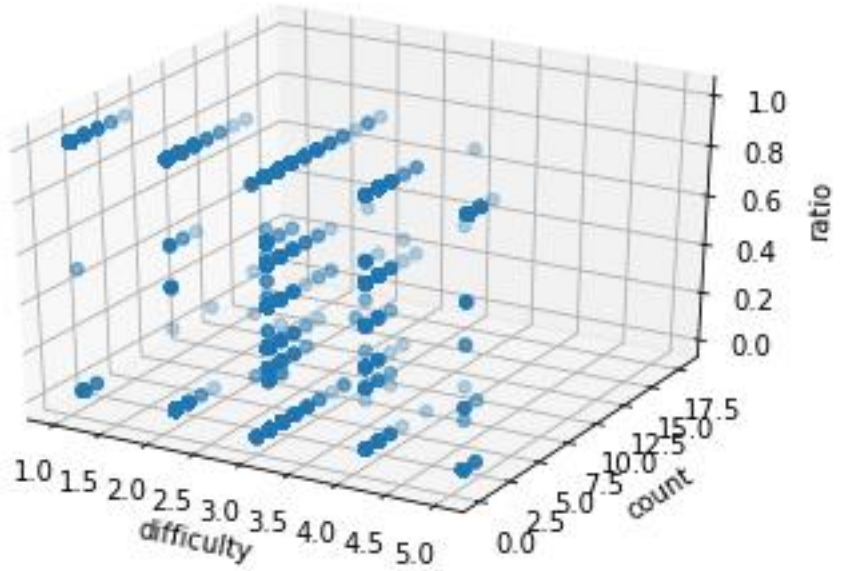


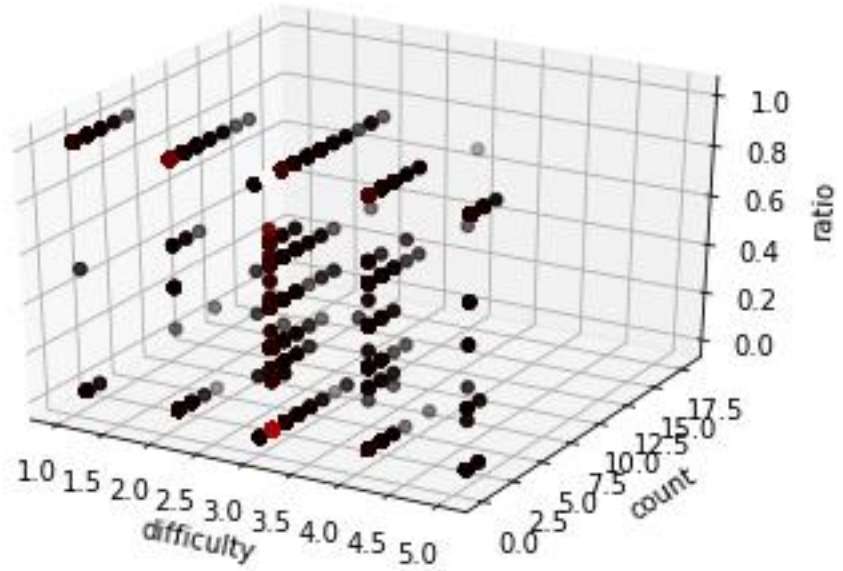


- 제출 횟수 1회 일 때 가장 높은 정답률을 보인다.
- 답을 고칠수록 낮은 점수 비율이 줄어든다.

6. Data Analysis

- difficulty - count - ratio

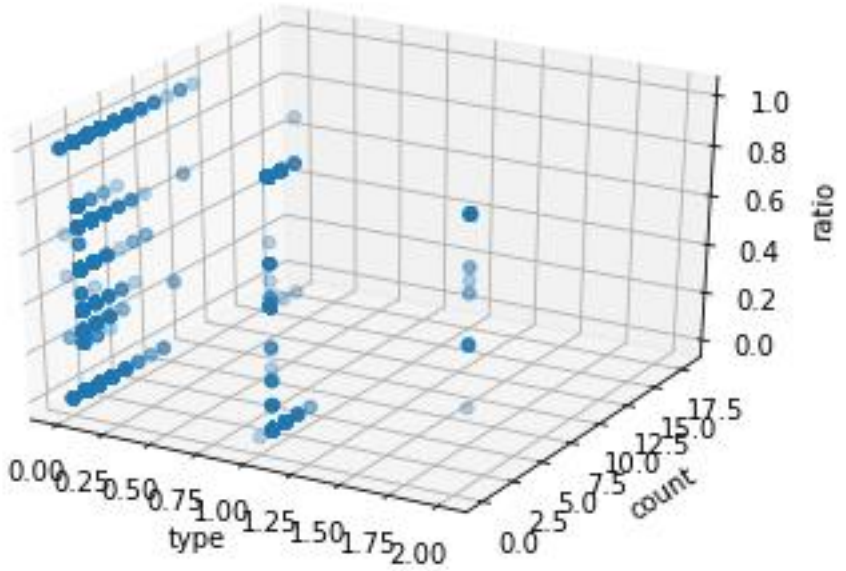


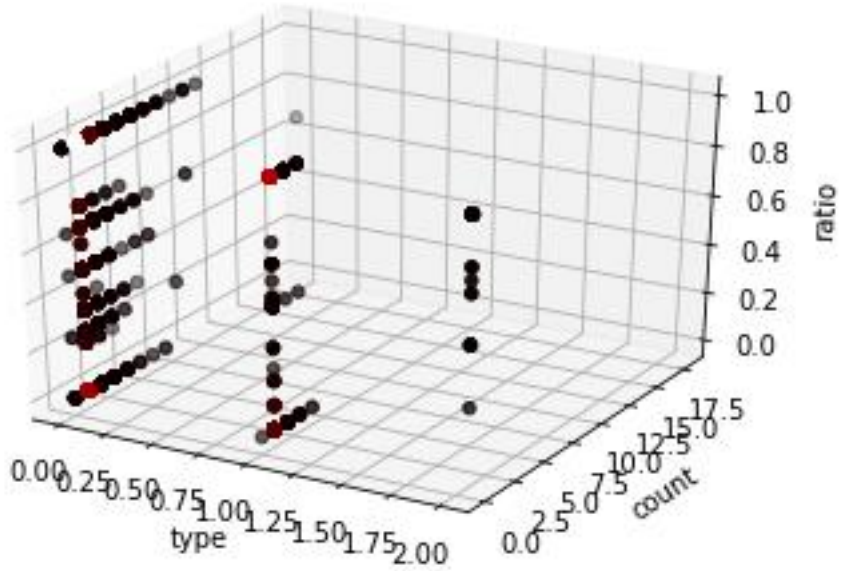


- 난이도 분포가 고르기 때문에 앞의 분석 결과가 유의미하다.

6. Data Analysis

- type - count - ratio

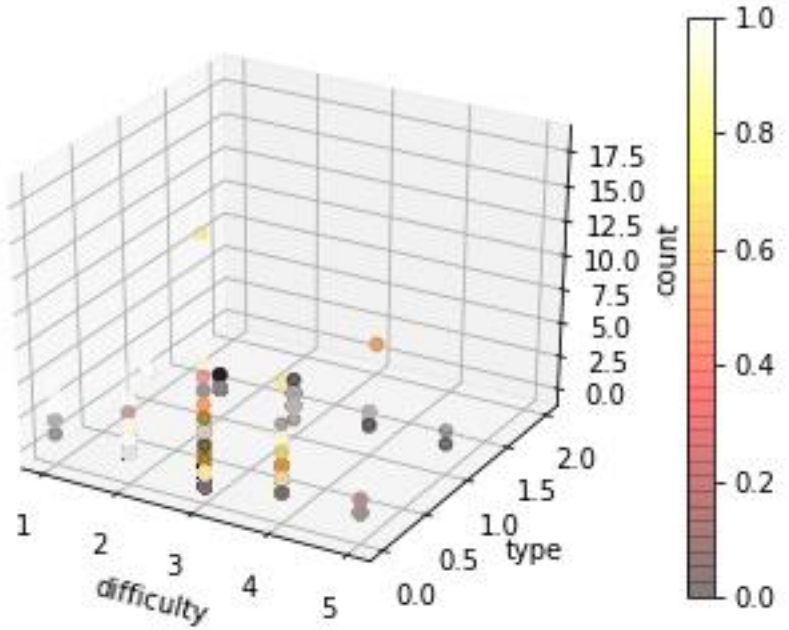




- 단답형이 고친 횟수가 가장 많다.

6. Data Analysis

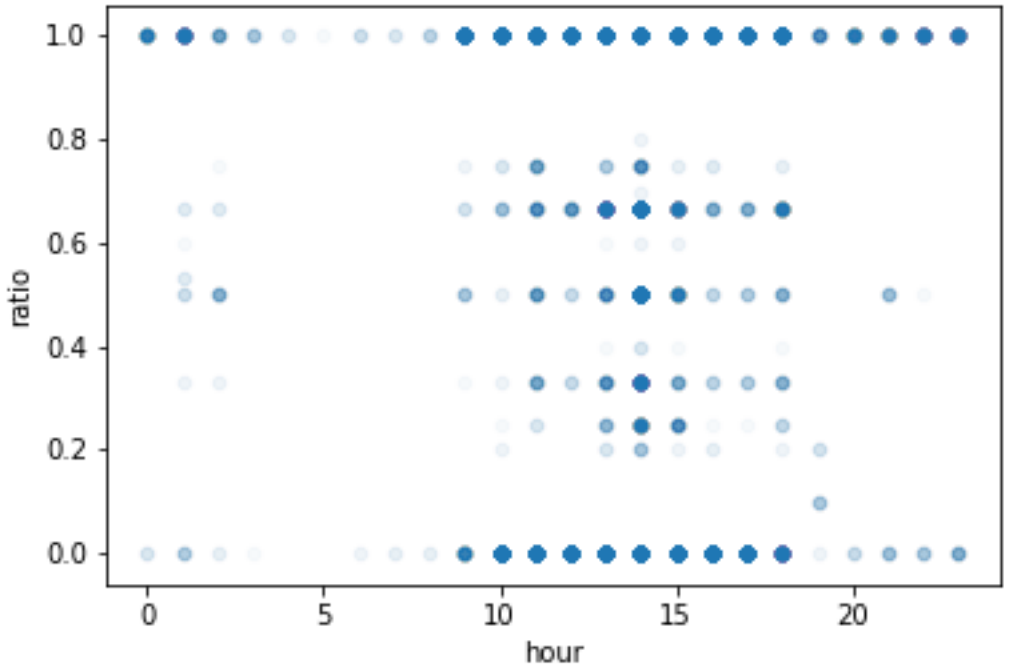
- difficulty - type - count - ratio

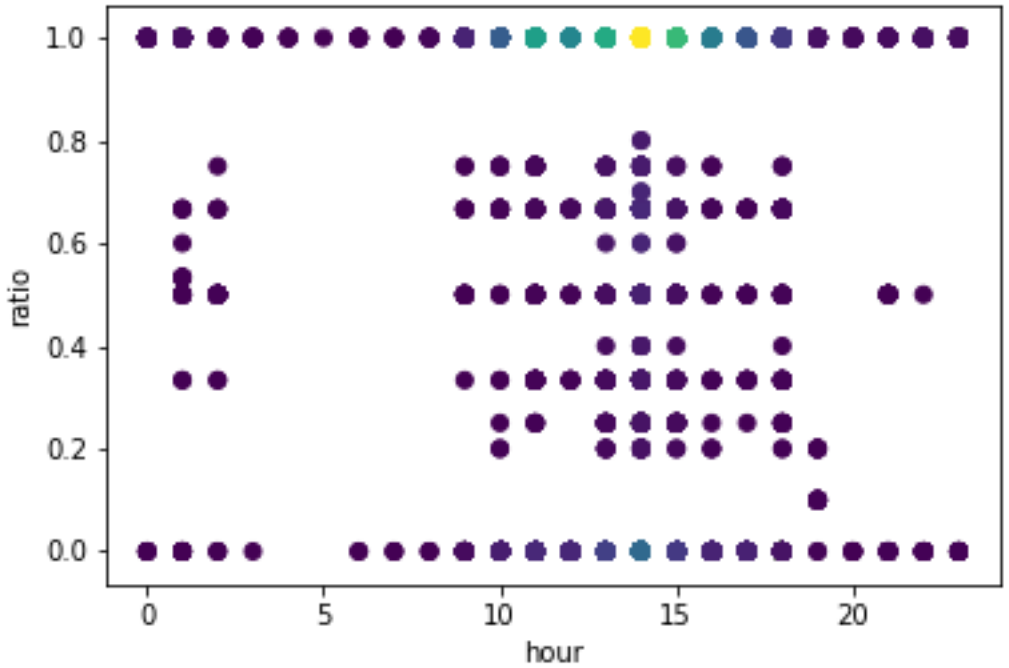


- 객관식은 1, 2회 고치는것이 정답률이 가장 높다.

6. Data Analysis

- hour - ratio

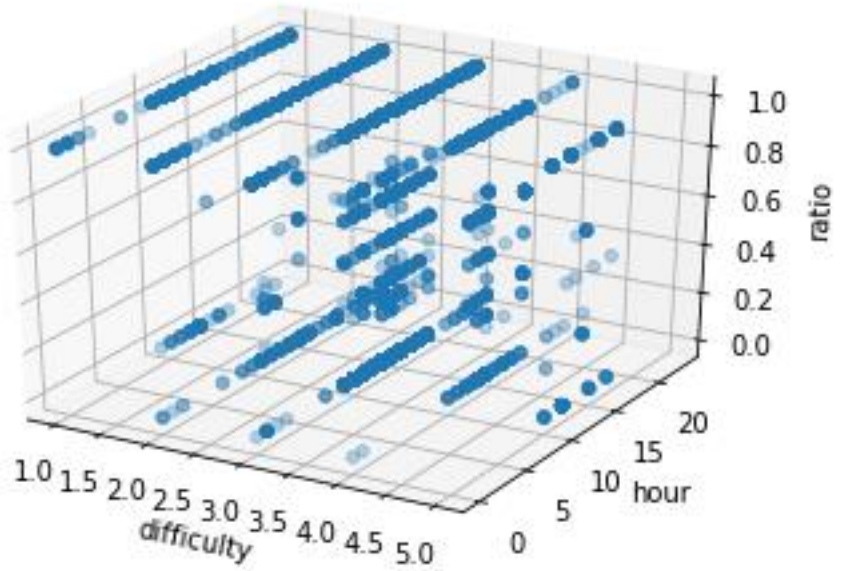


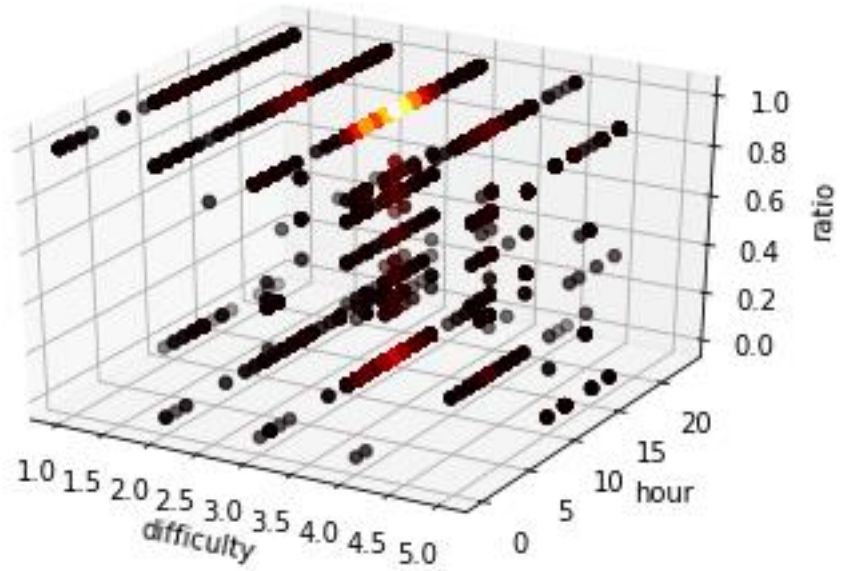


- 오후 2시에 가장 많은 문제 제출이 이루어졌으며 점수가 가장 높다.
- 오전일수록 점수가 높고 오후일수록 점수가 낮다.

6. Data Analysis

- difficulty - hour - ratio

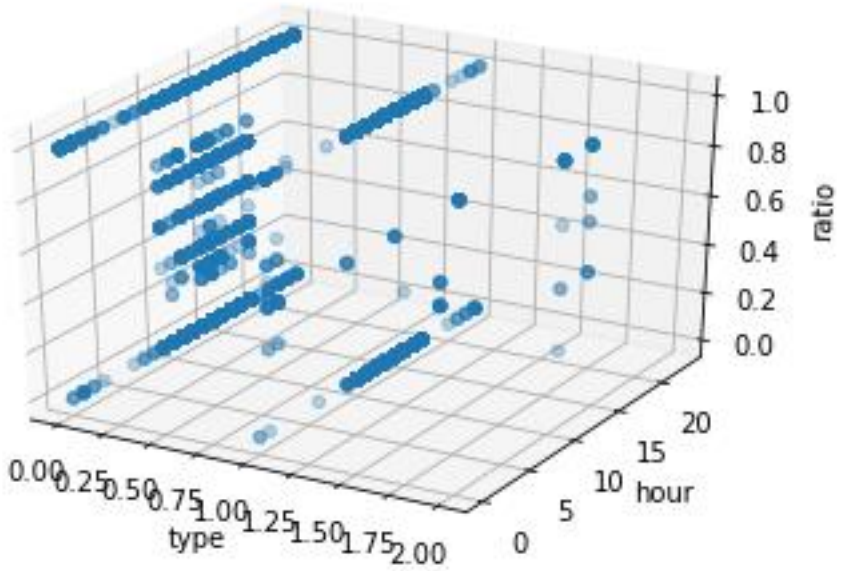


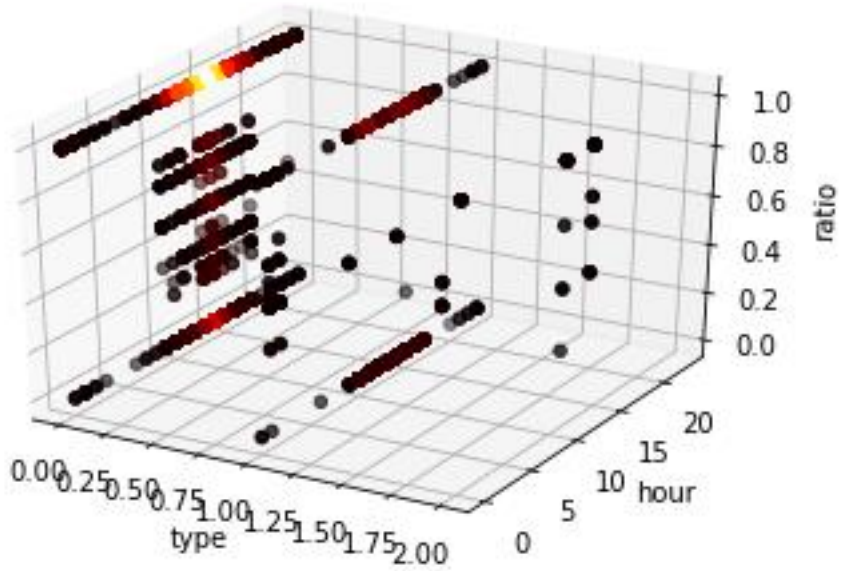


- 난이도 분포가 고르기 때문에 앞의 분석 결과가 유의미하다.

6. Data Analysis

- type - hour - ratio

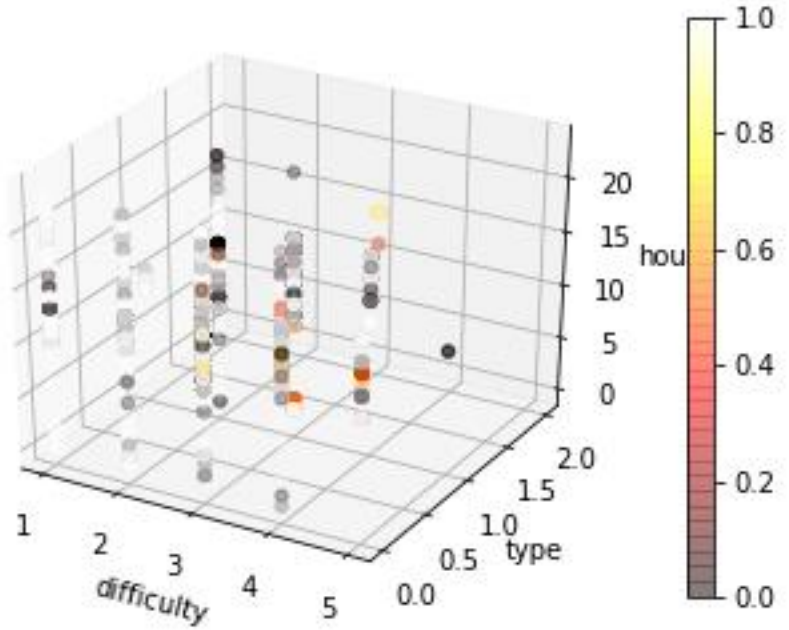




- 대부분 유형의 문제가 고르게 분포하므로
앞의 분석 결과가 유의미하다.

6. Data Analysis

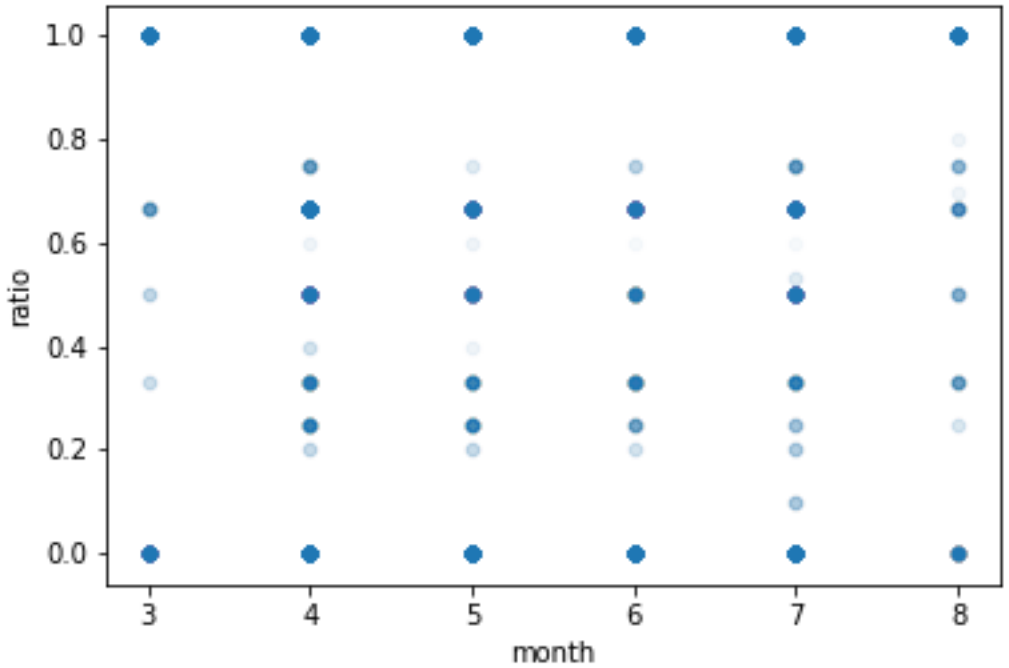
- difficulty - type - hour - ratio

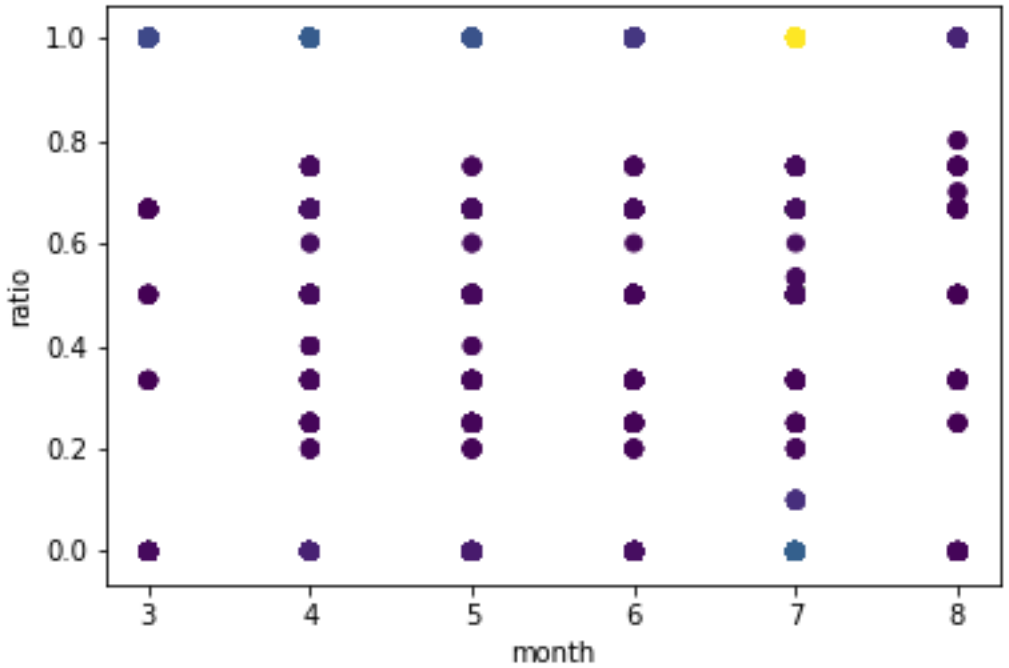


- 난이도 3, 주관식 문제는 오후 3시쯤 풀 때 가장 좋은 점수를 받는다.
- 난이도 4, 주관식 문제는 새벽에 풀 때 가장 낮은 점수를 받는다.

6. Data Analysis

- month - ratio

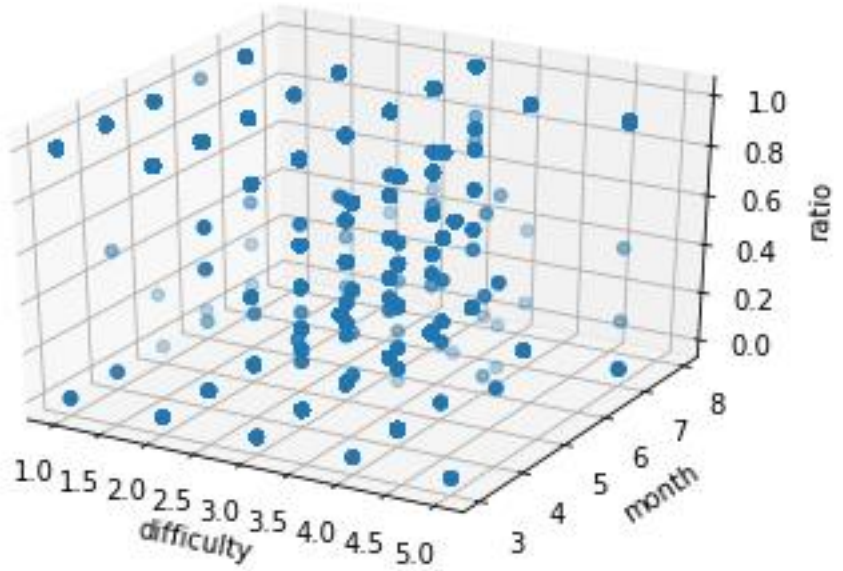


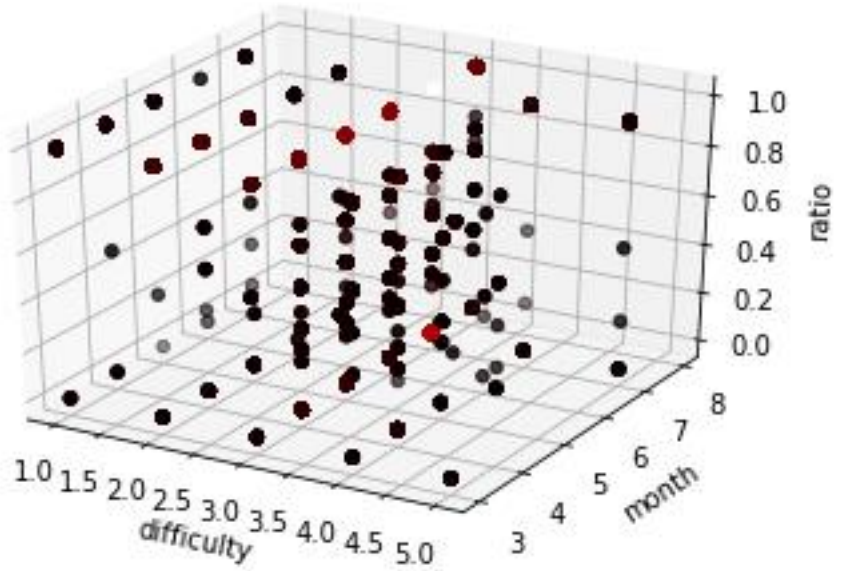


- 월을 거듭할수록 성적이 올라간다.

6. Data Analysis

- difficulty - month - ratio

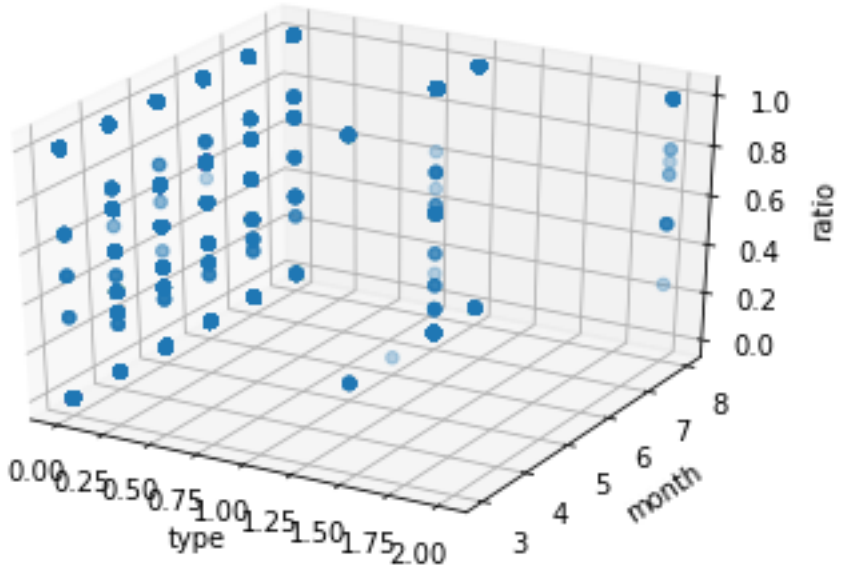


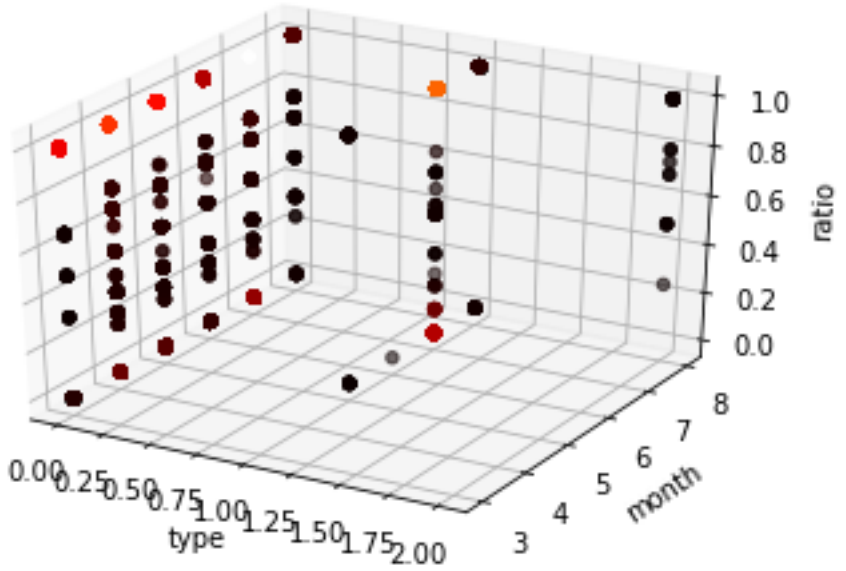


- 월별 난이도 분포 고르기 때문에 앞의 분석 결과가 유의미하다.

6. Data Analysis

- type - month - ratio

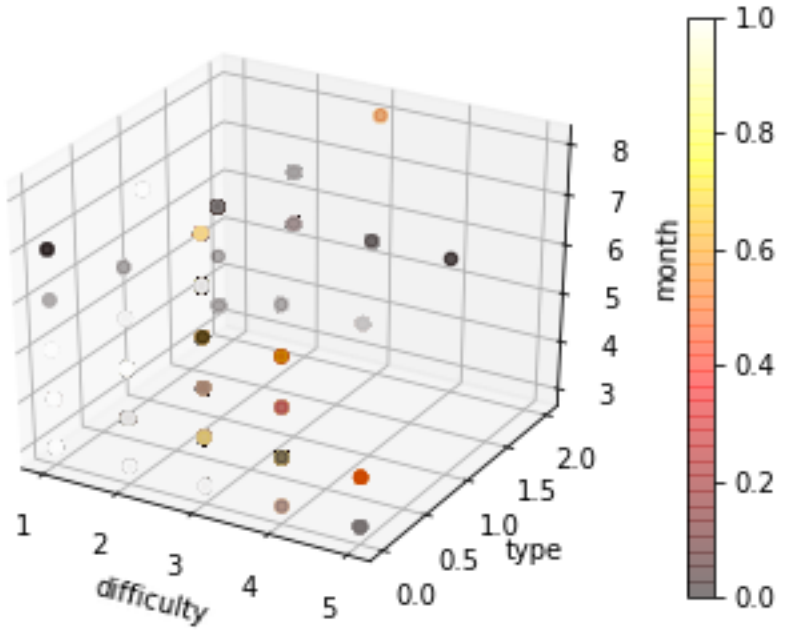




- 월말에 단답형 문제가 많아짐에도 불구하고 좋은 성적을 받고 있다.

6. Data Analysis

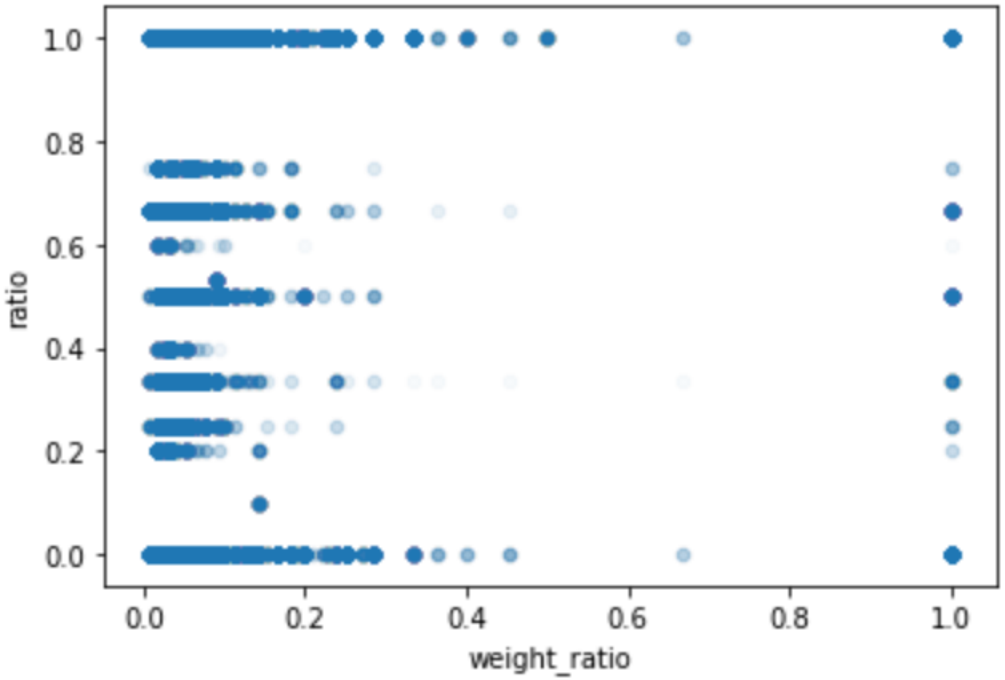
- difficulty - type - month - ratio



- 5월에 난이도 2, 주관식 문제가 가장 좋은 성적을 받고있다.
- 5월에 난이도 4, 주관식 문제가 가장 낮은 성적을 받고있다.

6. Data Analysis

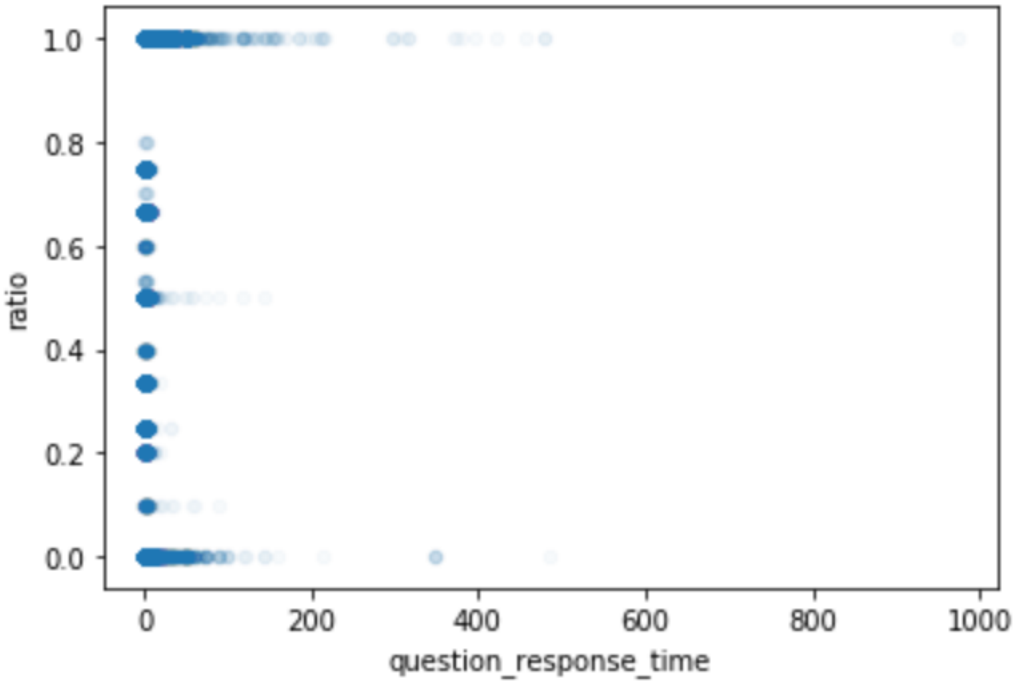
- weight_ratio - ratio



- 가중치가 올라 갈수록 중간 점수는 줄어들고 점수가 높아지는 경향을 보인다.

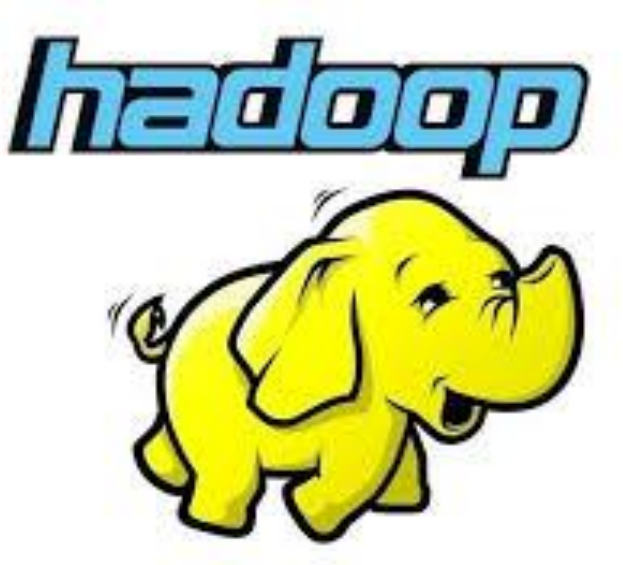
6. Data Analysis

- question_response_time - ratio

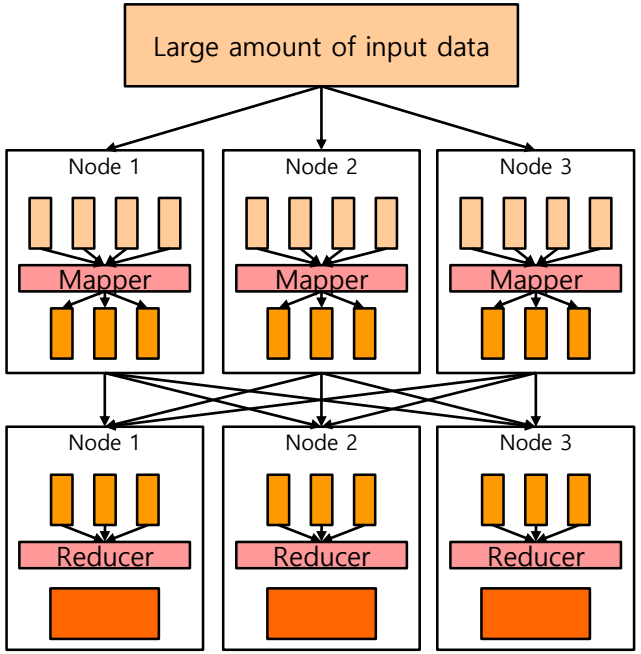


- 응답시간이 오래 걸린 학생은 중간 점수의 분포가 없어지고 정답을 확실하게 맞추거나 아예 못 맞추는 경우가 주를 이룬다.

7. Future Work

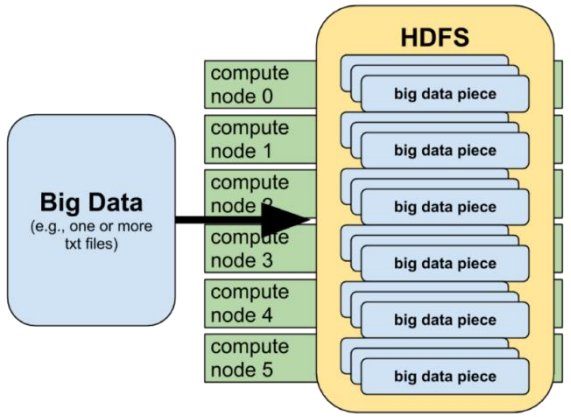


- **What is Hadoop?**
  - TB 단위의 Big data 대상
  - 빅데이터 문제(poor data quality, difficult to access, high computational complexity 등) 해결 목적
  - 분산 데이터 저장 (맵리듀스) 및 처리

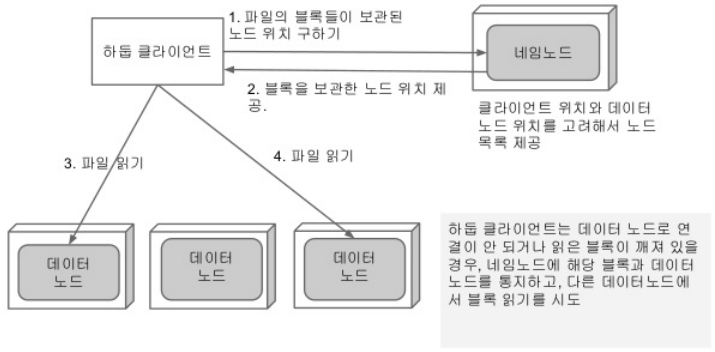


7. Future Work

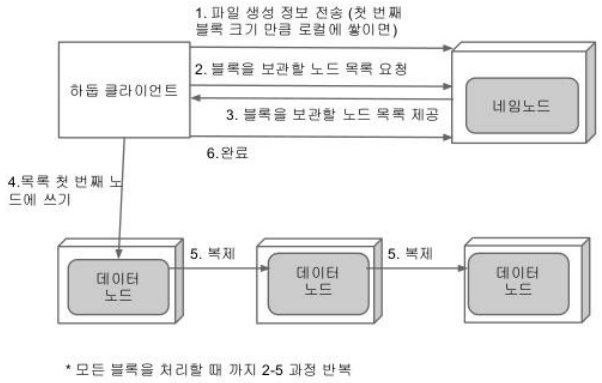
- **HDFC (Hadoop Distributed FS)**



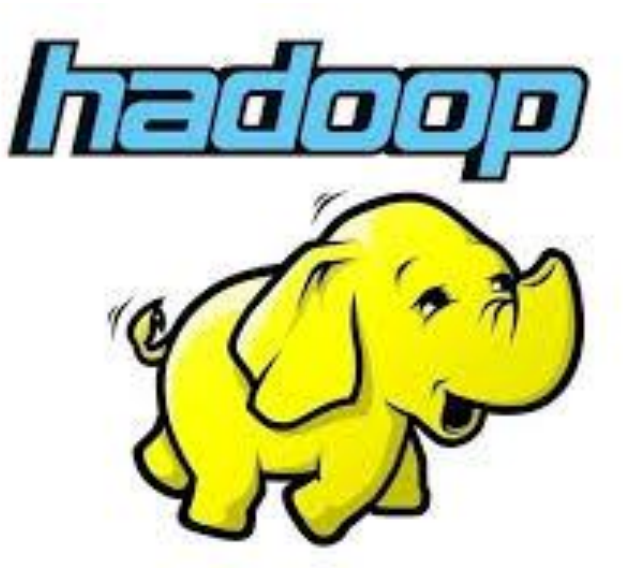
파일 읽기



파일 쓰기



7. Future Work



- 구현

```java
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

```java
  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

```
//Usage

//Environment Setting
export JAVA_HOME=/usr/java/default
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar

//Compile and Create JAR
$ bin/hadoop com.sun.tools.javac.Main WordCount.java
$ jar cf wc.jar WordCount*.class

//Sample text-files as input
$ bin/hadoop fs -ls /user/joe/wordcount/input/ /user/joe/wordcount/input/file01 /user/joe/wordcount/input/file02
$ bin/hadoop fs -cat /user/joe/wordcount/input/file01
>> Hello World Bye World
$ bin/hadoop fs -cat /user/joe/wordcount/input/file02
>> Hello Hadoop Goodbye Hadoop

//Run
$ bin/hadoop jar wc.jar WordCount /user/joe/wordcount/input /user/joe/wordcount/output

//Output
$ bin/hadoop fs -cat /user/joe/wordcount/output/part-r-00000`
>> Bye 1
>> Goodbye 1
>> Hadoop 2
>> Hello 2
>> World 2`
```