Study Title Day 8

- Statistic/Math
1 검정력(statistical power)은 무엇일까요?
- Machine Learning
2 앙상블 방법엔 어떤 것들이 있나요?
- Deep Learning
3 딥러닝 발달 이전에 사물을 Detect할 때 자주 사용하던 방법은 무엇인가요?
4 Fatser R-CNN의 장점과 단점은 무엇인가요?
5 dlib은 무엇인가요?
- Data Base
6 Key란 무엇인가요?
7 Key의 다섯 가지 종류에 대해 설명해주세요.
- Algorithm
8 Problem - Programmers - 타겟 넘버
9 Problem - Programmers - N으로 표현

1. 검정력(statistical Power): 대립가설(H_1)이 사실일 때, 이를 사실로서 결정할 확률
(H₀가 기각할 확률)

2. 앙상블 기법

· 배깅(Bagging): Bootstrap aggregating ⇒ 병렬적으로 부트스트랩을 집계하여 복원 추출

부트스트랩(Bootstrap): 통계학에서 사용하는 언어로, random sampling을 적용하여 데이터 증식 방법을 일컫는 말

→ 부트스트랩으로 집계하여 학습 데이터 충분하지 않더라도 충분학습 효과를 주어 높은 bias의 과소적합이나 과대적합 방지

· 부스팅(Boosting): 순차적인 복원 추출로 가중치 부여

배깅의 경우 각각의 분류기들이 학습시 상호영향을 주지 않는 상황에서 학습이 끝난 다음 결과를 종합하는 방면, Boosting은 이전 분류기의 학습 결과를 토대로 순차적으로 가중치를 조정

일반적으로 오답에 대해 높은 가중치를 부여하므로 정확도가 높은 편

· 스태킹(Stacking): 교차 검증을 통해 서로 상이한 모델 조합

⇒ 개별 모델이 예측한 데이터를 다시 training set로 사용해 학습

3. · grayscale - adaptive threshold를 이용한 segmentation
· 특징 검출을 이용한 object detection

4. Faster R-CNN

5. dlib: 이미지 처리, 선형대수 뿐만 아니라 다양한 머신러닝 알고리즘을 활용할 수 있는 라이브러리로서, 특히 HOG 특성을 사용하여 얼굴 검출하는 기능에 사용됨

6. Key: 검색, 정렬시 Tuple을 구분할 수 있는 기준이 되는 attribute

7. · Candidate key(후보키): Tuple을 유일하게 식별하기 위해 사용하는 속성들의 부분 집합(기본키의 후보들)
  - 유일성: key로 하나의 Tuple을 유일하게 식별할 수 있음
  - 최소성: 꼭 필요한 속성으로만 구성
· Primary key(기본키): 후보키 중 선택한 Main key
  - Null 값은 가질 수 없음
  - 중복 불가능
· Alternate key(대체키): 후보키 중 기본키를 제외한 나머지 키(보조키)
· Super key(슈퍼키): 유일성은 만족하지만, 최소성을 만족하지 못하는 키
· Foreign key(외래키): 다른 릴레이션의 기본키를 그대로 참조하는 속성의 집합

Study Title Day 8

8.

```cpp
void dfs(vector<int> numbers, int target, int sum, int idx) {
    if (idx == numbers.size() && sum == target) {
        answer++;
        return;
    }
    if (idx >= numbers.size()) return;
    dfs(numbers, target, sum + numbers[idx], idx + 1);
    dfs(numbers, target, sum - numbers[idx], idx + 1);
}
```

9.

```python
def solution(N, number):
    if N == number:
        return 1

    cache = [set() for _ in range(9)]
    # N, NN, NNN, NNNN ...
    for i in range(1, 9):
        cache[i].add(int(str(N) * i))

    for i in range(1, 9):
        for j in range(1, i):
            for k in cache[j]:
                for a in cache[i - j]:
                    cache[i].add(k + a)
                    cache[i].add(k - a)
                    cache[i].add(k * a)
                    if a != 0:
                        cache[i].add(k // a)
        if number in cache[i]:
            return i

    return -1
```