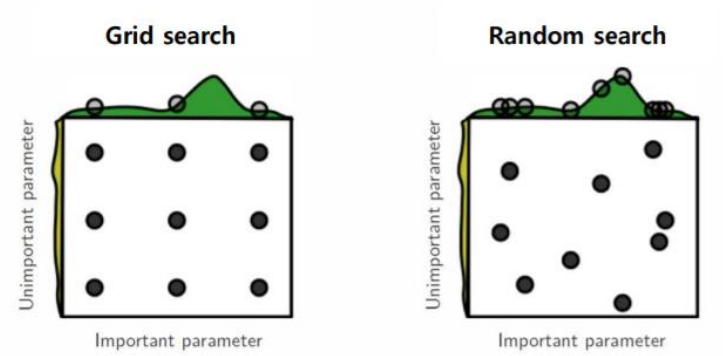
## 회고 및 하이퍼파라미터 튜닝 방법

전처리 및 모델을 구축한 뒤 성능 평가와 튜닝을 모두 해보기 위해서 데이터들을 뒤지던 와중에 진료내역정보를 공공데이터포털에서 제공하는 것을 확인하여 성능 향상을 통해 보험사의 보험료 책정 및 시민들이 스스로 얼마나 진료비가 나올지를 책정해보는데 도움이 될 수 있도록 수진자의 본인 부담금을 예측하는 프로젝트를 해보았다.

- 전처리 및 스케일링
- 학습 및 튜닝(Random Forest Regression)
  - 하이퍼 파라미터를 조정하지 않은 기본 모델 : 0.64
  - 그리드 탐색 적용 : 0.7
  - 랜덤 그리드 탐색 적용(그리드 탐색으로 찾은 좋은 성능의 하이퍼파라미터 및 추가 파라미터 이용) : **0.71**

진료내역정보에서 본인 부담금에 영향을 미치는 요소들이 중요하기 때문에 이런 요소들을 찾기 위해서 **연관관계**를 계산하고, Random Forest의 분기를 결정하는 **feature Importance**를 함께 이용하면 의미 있는 결과를 낼 수 있겠다고 생각하였다. 특히 입원, 원내 투약 등 진료비에 직접적인 연관성이 있는 것뿐 아니라 **직접적인 연관성을 파악하기 힘든 feature(병명, 진료 과목 등)들에서 의미를 찾아내는 것이 중요하다고 생각**되어 **성능**이 잘 나오는 것이 중요했다. (성능이 높으면 직접적인 연관성을 파악하기 힘든 feature에서 나온 **0.05**(연관관계 및 Feature Importance) 정도의 낮은 값들도 본인 부담금에 영향이 있다고 생각할 수 있기 때문) **0.71**정도의 결과로는 이런 feature들이 의미가 있다고 확신을 갖기가 어려웠다.

Random Forest의 성능을 높이기 위해 hyperparameter 튜닝 시에 앞서 보인 대로 Grid Search를 적용한 후에 성능이 잘 나온 hyperparameter들만 뽑고 몇 개의 파라미터를 추가하여 Randomized Grid Search를 진행하였다. 여기서 의구심이 들었던 부분은 (1) **원래 이렇게 하는게 맞는가** 라는 생각과 (2) **많은 데이터가 아니고(400MB, 전처리 후 더 줄어듦), hyperparameter의 후보를 얼마 두지 않았는데도 불구하고 좋은 성능의 hyperparameter를 찾는 데에 많은 시간이 걸렸는데(물론 노트북으로 했지만) 더 큰 데이터, 더 많은 hyperparameter의 후보를 두면 더 오랜 시간이 걸릴 텐데 이렇게 일일이 Grid Search(Randomized Grid Search)를 적용할 수 있나?** 라는 생각이 들어 찾아보았다.




[그림 1] Grid search, Random search

(많은 시간을 들여도 하이퍼파라미터 후보군 내에서 **최적의 성능을 찾지 못할 수도 있다**.)

(출처 : Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In

Advances in neural information processing systems (pp. 2951-2959).)

생각한대로 Grid Search나 Randomized Grid Search는 파라미터의 조합 수가 커지면 매우 많은 튜닝 시간이 걸리기 때문에 (2) **실제로는 적용하기 어려운 부분이 존재**한다고 하며, 이전 단계에서 (1) **성능이 개선된 하이퍼파라미터를 기반으로 성능 개선의 방향성을 잡는다는 것이 효율적**이라고 한다.

위의 튜닝 과정에서는 Grid Search로 개선된 하이퍼파라미터를 이용해서 몇 개의 parameter를 추가하여 다시 Randomized Grid Search를 진행하였다. 하지만 Bayesian Optimization이라는 다른 방법이 있었다. 이는 딥러닝 모델의 하이퍼파라미터 튜닝시에도 현실적으로 사용이 가능한 모델이었다.

Bayesian Optimization에는 두 가지 필수 요소가 있는데 이 중 **Surrogate Model**은 하이퍼파라미터 후보군과 모델의 성능 사이의 관계를 예측해주는 역할(하이퍼파라미터 집합의 subset만으로 일반화 성능을 예측할 수 있다.), 또 하나인 **Acquisition Function**은 가장 최적의 하이퍼파라미터를 찾는 함수이다.

새로운 하이퍼파라미터 집합이 생겼을 때, 앞에서 관찰된 결과로 일반화 성능을 진단하기 때문에 새롭게 튜닝해야 할 하이퍼파라미터가 생길 때마다 재학습시킬 필요가 없다. 이 방식은 구글의 AutoML에서도 사용하는 방식이다.

(참고 : http://dmqm.korea.ac.kr/activity/seminar/285)