디지털헬스 해커톤 2021 AI 트랙
결과 보고서

팀명 : 의학으학으악!!!

팀원 : 김난희

강민구

정해천

목차

Abstract

여러가지 AI 방법 중 저희는 문제해결 방향을 Deep Learning Modeling으로 초점을 맞췄습니다. 설계된 Deep Learning Model은 주어진 데이터에 대하여 High Performance를 보여줄 수 있어야 합니다. 뿐만 아니라 주어진 유전자 정보 300개 중에서 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**(이하 Top10 Genetic Variables)"를 뽑아낼 수 있어야 하는데, 이를 위해서는 Deep Learning Model이 결과에 대하여 Explainable해야 합니다. 따라서 저희는 주어진 데이터에 대하여 High Performance & Explainable Deep Learning Model을 설계하기 위하여 노력하였고, 저희가 설계한 여러가지 Deep Learning Model로 실험을 해보았습니다. 그 중 가장 좋은 Performance를 보여주는 Deep Learning Model을 채택하여 결과에 대한 인과관계가 높은 Top10 Genetic Variables를 추출하였고, 저희가 뽑은 Top10 Genetic Variables가 중요한 유전자 정보가 맞는지를 증명하기 위하여 Deep Learning & Machine Learning Model을 사용하여 실험하였습니다. 이 실험 결과는 저희가 뽑은 Top10 Genetic Variables가 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"에 가깝다는 것을 보여줍니다.

그리고 저희가 생각하는 저희의 문제해결 방법에 대한 독창적인 부분은 3가지입니다.

1. 기존 방식과 다른 새로운 Deep Learning Model을 설계.
2. Learnable Parameter를 통하여 Explainable Deep Learning Model을 설계.
3. 설계한 Deep Learning Model을 통하여 뽑은 Top10 Genetic Variables가 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"에 가깝다는 것을 실험을 통하여 증명.

Top10 Genetic Variables

G211, G179, G80, G27, G147, G139, G242, G264, G290, G130

Introduction

주어진 문제를 해결하기 위해서는 문제해결 방향을 설정해야 하는데, 그러기 위해서는 우선 주어진 데이터와 목표로 하는 결과에 대한 분석이 중요합니다. 현재 주어진 데이터는 Clinical Variables(10개의 변수), Genetic Variables(300개의 변수), Survival & Time Event(2개의 변수), Treatment(1개의 변수)이고, 총 1000개의 데이터로 이루어져 있습니다. 그리고 이를 통해 이루고자 하는 목표는 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"입니다. 현재 주어진 데이터의 문제점은 데이터 개수가 변수에 비하여 너무 적다는 것입니다. 주어진 데이터가 총 1000개인데, 주어진 변수가 총 313개 입니다. 데이터의 개수는 적은데 반해 변수가 지나치게 많은 상황입니다. 이런 상황에서 저희는 통계적 데이터 분석을 통하여 결과를 얻기에는 어렵다고 판단하였습니다. 다음 선택지는 Machine Learning Model로 학습하여 결과를 도출하는 것이었습니다. Machine Learning Model을 통한 학습은 어느정도 좋은 결과를 보여줄 수 있지만, "결과에 대하여 Explainable 할 수 있는가?"에 대하여 고민해보았습니다. Machine Learning Model의 결과에 대하여 Explainable 하기 위한 여러가지 Feature Selection Model들이 존재하긴 하지만, 이것 역시 사람이 눈으로 보기에 설득력이 좀 떨어진다 판단하였습니다. 따라서 저희는 문제에 적합한 새로운 Deep Learning Model을 설계하기로 하였습니다. 여기서 중요한 초점은 High Performance도 있지만 애초에 모델 설계 자체를 Explainable하게 설계하는 것이었습니다.

Deep Learning Modeling을 하기 위해서는 가장 먼저 Input과 Output을 설정해야 합니다. 간단하게 생각을 하면 원하고자 하는 결과가 "**유전자 후보 10개**"이기 때문에 이것을 Output으로 설정을 할 수 있습니다. 하지만 이것은 위에 언급한 데이터와 변수의 차이 때문에 어렵다고 판단하였습니다. 그렇다면 무엇을 Output으로 설정할 것인가. 여기서 저희가 주목한 부분은 "**치료 효과를 증가시키는**"입니다. 치료 여부와 그에 대한 효과를 Output으로 설정하는 것입니다. 치료 효과가 증가되었다는 것을 보기 위해서는 일단 치료 여부(Treatment Variable)가 중요합니다. 그리고 "치료 효과가 증가되었는가"를 알기 위해서는 치료에 대한 결과가 필요합니다. 이것을 저희는 생존 여부(Survival Variable)라고 생각 하였습니다. 따라서 Treatment & Survival Variable의 조합을 Output으로 설정하기로 하였고, 두 변수 모두 Binary Label이기에 두 개의 변수를 조합하여 총 4개의 Class로 구성된 새로운 Label을 생성하였습니다. 4개의 변수는 다음과 같고, Label.py 코드를 통하여 새로운 csv 파일(Label.csv)을 만들었습니다.

1. 치료o → 생존
2. 치료o → 사망
3. 치료x → 생존
4. 치료x → 사망

Model의 Output이 Treatment & Survival Variable로 설정되었으므로 나머지 모든 변수를 Input으로 설정을 하였습니다. 데이터의 Input & Output을 설정 하였으므로, 다음으로 데이터 전처리 과정이 필요합니다. 데이터 전처리는 모델이 Raw 데이터로 학습을 했을 때 생기는 여러가지 문제점들을 방지하기 위해서 Outlier 제거, Data Normalization 등 모델에 데이터를 입력하기 전 데이터를 정제하는 과정입니다. 우선 주어진 데이터의 Outlier를 조사해 보았고, 다음과 같이 2개의 부분에 대해서 Outlier를 처리해 주었습니다.

1. Survival Time Variable에 음수 값이 존재 → 해당 값의 절대값으로 대체
2. Clinic Variable에 0 ~ 9 범위를 넘어가는 값이 존재 → 9보다 큰 수를 9로 대체

   (0보다 작은 수는 존재하지 않았음)

그리고 모델에 데이터를 넣어줄 때, 변수들 간의 값의 범위가 비슷한 것이 좋습니다. 따라서 Data Normalization을 해주게 되는데, 저희는 다음과 같이 Clinic & Genetic Variables에 대하여 Data Normalization을 진행하여 주었습니다.

1. Clinic Variables 범위 0 ~ 9 → 0.1 ~ 1.0
2. Genetic Variables 범위 0 or 1 → -0.5 or 0.5

위의 과정들을 통하여 문제해결 방향 설정, Input & Output 설정, 데이터 전처리를 완료하였고, 이를 기반으로 Deep Learning Modeling을 시작하였습니다.

Method

현재 주어진 문제를 해결하기 위하여 새로운 Deep Learning Model을 설계하기로 하였고, 설계한 Deep Learning Model은 High Performance & Explainable해야 합니다.

우선 Explainable한 모델을 만들기 위해서 저희는 Treatment Variables와 Genetic Variables에 학습 가능한 파라미터(Learnable Parameter)를 부여하기로 하였습니다. 각각의 Variables에 Learnable Parameter를 부여하게 되면 모델은 해당 파라미터를 학습할 것입니다. 학습이 끝나고 나서 각각의 파라미터들의 값을 비교해보면 모델이 결정을 하기 위하여 어떤 Variables에 얼마만큼의 가중치를 주었는지를 사람이 직접 수치로 파악할 수 있습니다. 이 Learnable Parameter들은 아래 그림1과 같이 각각의 Variables의 값들과 곱해지게 됩니다. 이것은 Label Embedding의 개념과 비슷하지만 약간 다릅니다. 곱해져서 나온 이 값들은 각각의 Variables를 Learnable Parameter를 통해서 특정 Latent Space로 Embedding한 것과 비슷합니다. 따라서 이 벡터들을 각각 Clinic Embedded, Genetic Embedded라고 하였습니다. 저희는 모델의 학습이 끝난 후 Genetic Embedded를 통하여 모델이 어떤 Genetic Variables에 많은 가중치를 주었는지를 판단할 것입니다.

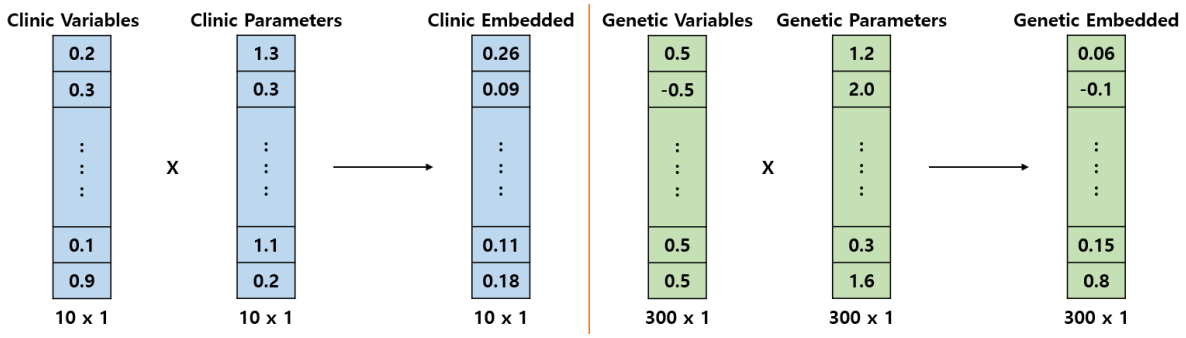


그림1. Learnable Parameter를 통한 Clinic & Genetic Variables의 Embedding

다음으로 High Performance를 얻기 위해서는 변수들 간의 상호 관계를 파악하여 결과를 도출해야 한다고 생각했습니다. Treatment 변수와 Genetic 변수 간의 상호 관계를 파악하기 위해서는 그림2와 같이 두 개의 벡터를 곱하여 Association Matrix인 Clinic-Genetic(10x300)을 만들 수 있습니다. 이 뿐만 아니라 Clinic 변수들끼리의 Association Matrix인 Clinic-Clinic(10x10)과 Genetic 변수들끼리의 Association Matrix인 Genetic-Genetic(300x300)까지 총 3개의 Association Matrix를 생성합니다. 이렇게 생성된 총 3개의 Association Matrix를 아래와 같이 Reshape을 통하여 Vector화 시킵니다.

1. Clinic-Clinic Matrix (10 x 10) → Clinic-Clinic Vector (1 x 100)
2. Clinic-Genetic Matrix (10 x 300) → Clinic-Genetic Vector (1 x 3000)
3. Genetic-Genetic Matrix (300 x 300) → Genetic-Genetic Vector (1 x 90000)

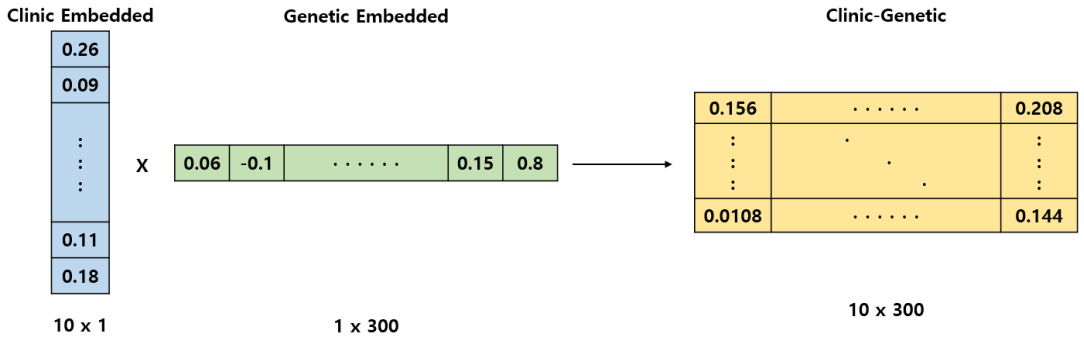


그림2. Clinic Embedded와 Genetic Embedded를 곱하여 Clinic과 Genetic의 Association Matrix를 구합니다. 이를 Clinic과 Clinic, Genetic과 Genetic에 대해서도 진행합니다.

위에서 구한 3개의 Association Vector들과 Survival Time Variable을 모두 Concatenate하여 하나의 Feature Vector(93101x1)를 만듭니다. 이 Feature Vector를 Out Layer에 통과시켜서 최종 Logit(4x1)을 결과로 얻어 CrossEntropy를 통하여 Label과 매칭합니다.

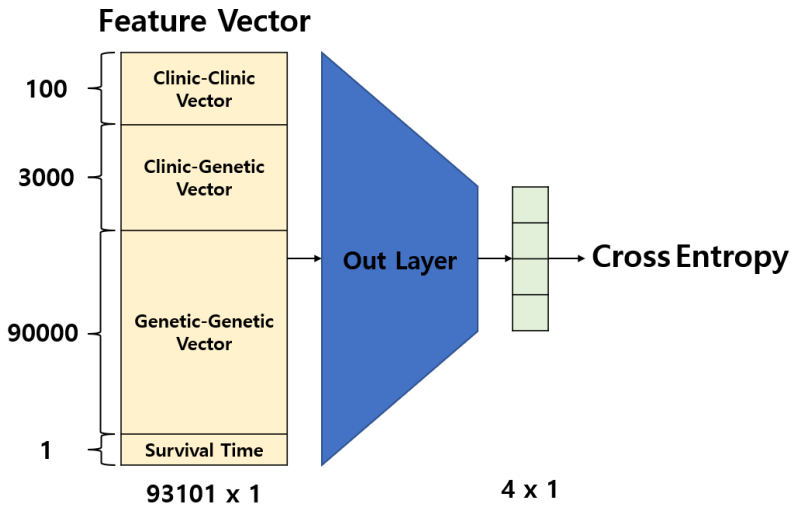


그림3. 3개의 Association Vector와 Survival Time Variable을 Concatenate한 후, Out Layer를 통과시켜 최종 Logit을 뽑아냅니다.

Train & Test Set은 8:2로 분할하였고, Train Set은 Random Shuffle되어 모델에 제공됩니다. 각각의 Learnable Parameter들은 1로 초기화를 하였고, Out Layer의 Parameter는 랜덤 초기화 하였습니다. 모델의 Initialization에 랜덤성이 있기 때문에 동일한 모델을 여러 번 학습해 보았습니다. 그래서 총 100개의 동일 모델을 1000 Epoch 학습시켰습니다. 학습시킨 그래프의 개형은 그림4와 같이 나왔으며, 표1과 같이100개의 모델의 평균 Train Accuracy는 99.89%, 평균 Test Accuracy는 54.14%가 나왔습니다. Test Accuracy가 54.14%면 상당히 낮은 수치로 보일 수 있지만, 현재 데이터의 개수 자체가 변수의 개수에 비하여 상당히 적은 상황이며 새로운 Label을 사용하여 Binary Classification이 아닌 4 Class Classification을 진행한 것이기에 상당히 높은 수치라고 생각합니다. 그리고 이후에 증명 실험을 통하여 Deep Learning Model을 통하여 뽑은 Top10 Genetic Variables가 다른 조합의 10개의 Genetic Variables보다 Output과의 연관성이 높다는 것을 보여줍니다.

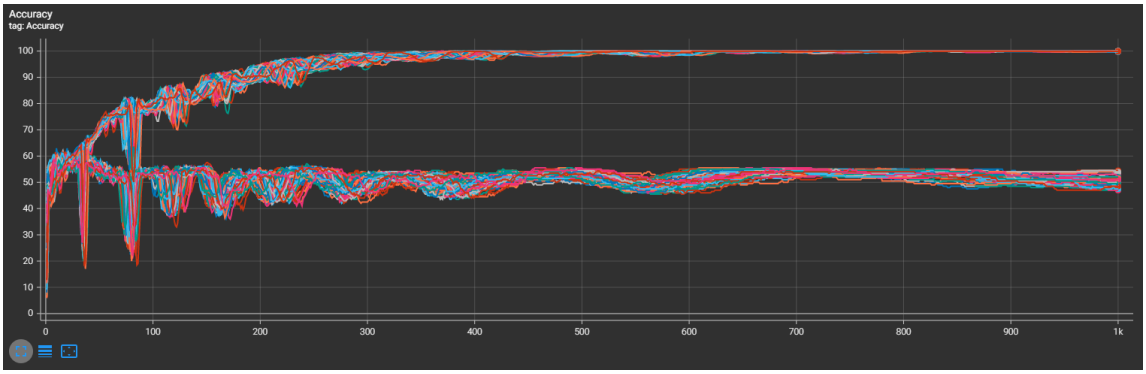
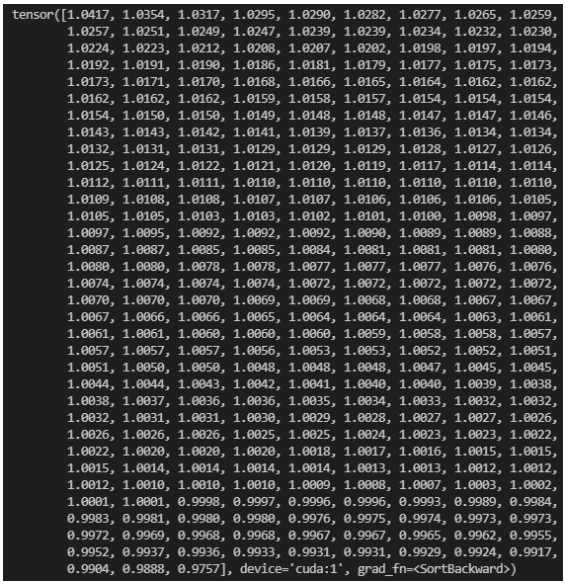


그림4. Random Initialization된 동일한 100개의 모델을 학습한 그래프. 계속 증가하는 그래프는 Train Accuracy이고, 증가하다가 평행선을 유지하는 그래프는 Test Accuracy입니다.

	Accuracy
Train	**99.89%**
Test	**54.14%**

표1. Proposed Model을 100개의 모델에 대하여 1000 Epoch씩 실행하여 얻은 평균 Accuracy.

모델 학습 종료 후, 저장된 모델의 Genetic Parameter를 Loading하여 학습하며 Genetic Variables에 대해 부여한 가중치들을 확인해 본 것이 그림5 입니다. 여기서 값이 큰 상위 10개의 Genetic Index를 추출하였습니다. 100개의 모델들에 대하여 추출된 상위 10개의 Genetic Index의 빈도수를 측정한 것이 첨부파일의 Top10_Genetic_Candidate.txt이고, 그 중 Top10을 뽑은 것이 표2 입니다. 저희는 표2의 10개 Genetic Index를 "**결과와 인과관계가 높은 변수들**"로 판단하였습니다. 동일한 방식으로 Clinic Parameter에 대하여 Top3를 뽑아봤는데, <Var0, Var1, Var3>이 선정되었습니다.



```
tensor([1.0417, 1.0354, 1.0317, 1.0295, 1.0290, 1.0282, 1.0277, 1.0265, 1.0259,
        1.0257, 1.0251, 1.0249, 1.0247, 1.0239, 1.0239, 1.0234, 1.0232, 1.0230,
        1.0224, 1.0223, 1.0212, 1.0208, 1.0207, 1.0202, 1.0198, 1.0197, 1.0194,
        1.0192, 1.0191, 1.0190, 1.0186, 1.0181, 1.0179, 1.0177, 1.0175, 1.0173,
        1.0173, 1.0171, 1.0170, 1.0168, 1.0166, 1.0165, 1.0164, 1.0162, 1.0162,
        1.0162, 1.0162, 1.0162, 1.0159, 1.0158, 1.0157, 1.0154, 1.0154, 1.0154,
        1.0154, 1.0150, 1.0150, 1.0149, 1.0148, 1.0148, 1.0147, 1.0147, 1.0146,
        1.0143, 1.0143, 1.0142, 1.0141, 1.0139, 1.0137, 1.0136, 1.0134, 1.0134,
        1.0132, 1.0131, 1.0131, 1.0129, 1.0129, 1.0129, 1.0128, 1.0127, 1.0126,
        1.0125, 1.0124, 1.0122, 1.0121, 1.0120, 1.0119, 1.0117, 1.0114, 1.0114,
        1.0112, 1.0111, 1.0111, 1.0110, 1.0110, 1.0110, 1.0110, 1.0110, 1.0110,
        1.0109, 1.0108, 1.0108, 1.0107, 1.0107, 1.0106, 1.0106, 1.0106, 1.0105,
        1.0105, 1.0105, 1.0103, 1.0103, 1.0102, 1.0101, 1.0100, 1.0098, 1.0097,
        1.0097, 1.0095, 1.0092, 1.0092, 1.0092, 1.0090, 1.0089, 1.0089, 1.0088,
        1.0087, 1.0087, 1.0085, 1.0085, 1.0084, 1.0081, 1.0081, 1.0081, 1.0080,
        1.0080, 1.0080, 1.0078, 1.0078, 1.0077, 1.0077, 1.0077, 1.0076, 1.0076,
        1.0074, 1.0074, 1.0074, 1.0074, 1.0072, 1.0072, 1.0072, 1.0072, 1.0072,
        1.0070, 1.0070, 1.0070, 1.0069, 1.0069, 1.0068, 1.0068, 1.0067, 1.0067,
        1.0067, 1.0066, 1.0066, 1.0065, 1.0064, 1.0064, 1.0064, 1.0063, 1.0061,
        1.0061, 1.0061, 1.0060, 1.0060, 1.0060, 1.0059, 1.0058, 1.0058, 1.0057,
        1.0057, 1.0057, 1.0057, 1.0056, 1.0053, 1.0053, 1.0052, 1.0052, 1.0051,
        1.0051, 1.0050, 1.0050, 1.0048, 1.0048, 1.0048, 1.0047, 1.0045, 1.0045,
        1.0044, 1.0044, 1.0043, 1.0042, 1.0041, 1.0040, 1.0040, 1.0039, 1.0038,
        1.0038, 1.0037, 1.0036, 1.0036, 1.0035, 1.0034, 1.0033, 1.0032, 1.0032,
        1.0032, 1.0031, 1.0031, 1.0030, 1.0029, 1.0028, 1.0027, 1.0027, 1.0026,
        1.0026, 1.0026, 1.0026, 1.0025, 1.0025, 1.0024, 1.0023, 1.0023, 1.0022,
        1.0022, 1.0020, 1.0020, 1.0020, 1.0018, 1.0017, 1.0016, 1.0015, 1.0015,
        1.0015, 1.0014, 1.0014, 1.0014, 1.0014, 1.0013, 1.0013, 1.0012, 1.0012,
        1.0012, 1.0010, 1.0010, 1.0010, 1.0009, 1.0008, 1.0007, 1.0003, 1.0002,
        1.0001, 1.0001, 0.9998, 0.9997, 0.9996, 0.9996, 0.9993, 0.9989, 0.9984,
        0.9983, 0.9981, 0.9980, 0.9980, 0.9976, 0.9975, 0.9974, 0.9973, 0.9973,
        0.9972, 0.9969, 0.9968, 0.9968, 0.9967, 0.9967, 0.9965, 0.9962, 0.9955,
        0.9952, 0.9937, 0.9936, 0.9933, 0.9931, 0.9931, 0.9929, 0.9924, 0.9917,
        0.9904, 0.9888, 0.9757], device='cuda:1', grad_fn=<SortBackward>)
```

```
tensor([210,  26, 241,  79, 178, 146, 263, 289, 138,   2,  91, 256, 212, 214,
        262, 129, 181, 254,  17, 281, 270,  74, 218, 126, 284, 164, 235, 102,
        234,  51,  10,   3, 250,  71, 169, 141, 131, 171,  75, 278, 118,  44,
        257,  34, 163, 116, 142, 271, 200, 217,  41, 119, 266, 291, 152, 229,
         20,  45, 283, 173, 150, 239,  30,  48,  90, 166, 275, 267,  76,  11,
        237,  99, 147,  46, 109, 188, 286, 274, 216,  58, 240, 273, 151, 206,
        191, 228, 245, 190, 134, 226, 280, 282, 179, 222,  49,  22,   7, 105,
        162, 135, 252,  56,  69,  70, 153,  98,  43,  61, 121, 253,  33, 148,
        296,  47, 299, 184, 220, 127,   4, 156,  78,  15, 221, 208, 276,  87,
        176, 168, 298, 172,  50, 165, 224,  66,  67, 259, 294,  80,  94, 213,
        192, 195,  96, 223, 285,  88,   9, 145,  14,   8, 140, 101, 108, 297,
        265, 288, 293, 167,  72,  21, 207, 128, 137, 113,  93, 144, 215, 103,
         60, 211,  13, 251, 203,  36, 104, 124, 170,  32, 158, 182, 106,  19,
        269,  52,  40, 261,  53, 243, 247, 268, 130, 161, 272,   5,  37,  68,
        264, 287,  62, 159,  59, 177, 132, 292, 111,  81, 209, 112,  57,  23,
        193, 260, 174,  89,  54, 154, 183, 295, 139,  28,  42, 290,  86, 205,
         35, 123, 233,  16, 149, 189, 204, 230, 136, 120, 100, 196,  25, 231,
         97, 277, 238,  55, 249,  77, 227, 133,  92,  82, 279, 187, 242, 155,
         84, 122, 219,  85, 246,  83, 202,  73, 157,   6, 244,  29,  12, 201,
        143, 175,  38,  39,   0,  63, 180, 199, 225, 110, 125,  64,  65, 114,
        255, 194,  95, 186,  31, 115, 160, 236, 107, 198, 248, 258,  27,   1,
        197, 117, 232, 185,  24,  18], device='cuda:1')
```

그림5. 왼쪽 그림은 학습된 Genetic Parameter의 값들을 내림차순으로 정렬한 것이고, 오른쪽 그림은 가중치 값들에 해당하는 Genetic Variable의 Index입니다.

Genetic Index	빈도수
G211	100
G179	97
G80	89
G27	87
G147	85
G139	82
G242	80
G264	74
G290	59
G130	34

표2. 100개의 모델에 대하여 각각 가중치가 높은 Top10 Genetic Index를 추출하여, 각각의 Top10 Genetic Index의 빈도수를 측정하였습니다. G211은 100개의 모델 모두에서 Top10 안에 포함된 것이고, G130은 34개의 모델에서 Top10에 포함된 것입니다.

Experiments

위의 Method에서 언급된 Deep Learning Model을 통해 학습하여 표4와 같이 Top10 Genetic Variables를 산출하였습니다. 산출된 Top10 Genetic Variables가 정말로 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"에 가까운 세트가 맞는 것인지를 확인하기 위한 실험을 설계하였습니다. 실험 방식은 저희가 뽑은 Top10 Genetic Variables와 나머지 변수들(Clinic Variables, Survival Time Variable)을 모델에 Input으로 모델의 결과를 산출합니다. 그리고 비교군으로서 Random하게 뽑힌 Genetic Variables 10개와 나머지 변수들(Random Set)을 모델에 Input으로 모델의 결과를 산출했을 때, 전자의 결과에 비하여 후자의 결과 수치가 낮다면 전자의 Variables Set가 더 좋다는 것을 실험적으로 보여줄 수 있다고 생각했습니다. 이 실험 방식을 증명하기 위해서 Deep Learning과 Machine Learning 모델을 사용하였습니다.

Deep Learning 학습으로 산출한 Top10 Genetic Variables(이후부터 Selected Set).

G211, G179, G80, G27, G147, G139, G242, G264, G290, G130

표4. Deep Learning 학습으로 산출한 Top10 Genetic Variables

Deep Learning 모델로 테스트.

Method에 나온 Deep Learning Model과 Architecture 형태는 동일한데, Genetic Variables가 300개가 아니라 10개를 넣어주었습니다.

Selected Set의 Deep Learning Model 테스트.

1. Selected Set을 Input으로 하여 모델을 10,000 Epoch 학습하여 Train & Test Accuracy 산출
2. 1번을 20번 반복하여 20개의 Train & Test Accuracy 산출 → 평균

Random Set의 Machine Learning Model 테스트.

1. 300개의 Genetic Variables 중 랜덤하게 10개의 Genetic Variables를 선택(Random Set)
2. Random Set을 Input으로 하여 모델을 10,000 Epoch 학습하여 Train & Test Accuracy 산출
3. 1, 2번을 반복하여 20개의 Train & Test Accuracy 산출 → 평균

	Train Accuracy	Test Accuracy
Selected Set	**68.20**	**68.03**
Random Set	**63.80**	**64.68**

표5. Selected Set과 Random Set에 대해 Deep Learning Model로 학습하여 산출한 Test Accuracy.

테스트할 Machine Learning 모델 선정.

Machine Learning 모델은 2가지로 선택합니다.

모델은 표6과 같이 Train accuracy가 100%인 경우와 100%가 아닌 경우 2가지로 나눠서 합니다.

Train accuracy가 100%인 모델의 경우	Random Forest Classifier(RandomForestClassifier)
Train accuracy가 100%가 아닌 모델의 경우	LightGBM Classifier(LGBMClassifier)

표6. Train accuracy가 100%에 도달할 경우(Overfitting)와 100%에 도달하지 않은 경우를 나누어 실험하기 위한 Machine Learning 모델.

2가지 경우로 나누어 테스트하는 이유는 Overfitting이 안 된 경우와 Overfitting이 된 경우를 보기 위함입니다. Train 정확도가 100%가 나온다는 것은 Overfitting이 되었다는 것을 의미합니다. Overfitting은 Train Set에 대해 모델이 과도하게 Fitting되는 현상입니다. 일반적으로 Overfitting이 발생하게 되면 Test Set에서의 성능이 떨어지게 됩니다. 그러나, 현재 데이터셋은 데이터 개수가 너무 적기 때문에 학습 초기에 Overfitting이 발생하게 됩니다. 따라서 Train 정확도가 100%가 나온 경우(Overfitting)에 대해서도 Test 정확도를 측정하여 보았습니다. 이상적인 경우인 Overfitting이 안된 경우에 대해서도 Test 정확도를 비교해보기 위해서, 학습의 횟수를 줄여 Train 정확도가 100%가 나오지 않도록 설정하여 그 때의 Test Accuracy를 비교해보았습니다.

Selected Set의 Machine Learning Model 테스트.

1. Deep Learning 모델링시 했던 데이터 전처리 조건을 동일하게 설정.
2. 동일한 데이터 셋과 동일한 Machine Learning 모델로 학습을 하더라도 매번 Test 성능에 차이가 있기 때문에 동일 세팅으로 20번 반복 수행하여 Train & Test의 최대, 최소값 산출.

Random Set의 Machine Learning Model 테스트.

Train Accuracy가 100%가 나오는 RandomForest Model과 Train Accuracy가 100%가 나오지 않는 LightGBM Classifier에 모두 동일하게 아래와 같은 실험 절차를 거쳤습니다.

1. Deep Leaning 모델링시 했던 데이터 전처리 조건을 동일하게 설정.
2. 300개의 유전자 변수 중에서 랜덤으로 10개의 유전자 변수를 뽑아 Random Set을 생성.
3. 동일한 데이터 셋과 동일한 Machine Learning 모델로 학습을 하더라도 매번 Test 성능에 차이가 있기 때문에 동일 세팅으로 20번 반복 수행하여 Train & Test의 최대, 최소값을 산출.
4. 위 2, 3번 과정을 100번 반복. 즉, 100개의 Random Set에 대하여 동일 모델로 20번 반복 수행하여 Train & Test의 최대, 최소값을 총 100개씩 산출.

실험 결과

Selected Set과 Random Set을 저희가 설계한 Deep Learning Model에 대하여 여러번 실험을 진행하고 그 평균을 산출하여 그 결과가 표5와 같이 나왔습니다. Train & Test Accuracy 모두에서 Selected Set이 Random Set보다 높은 수치를 보여주었습니다. 특히 주목할만한 점은 Test Accuracy가 Genetic Variables 300개를 통해 학습한 것보다 높다는 것입니다. 이를 통해 저희가 산출한 Selected Set이 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"에 가깝다는 것을 증명하였습니다.

Selected Set과 Random Set을 RandomForest와 LightGBM Classifier 모델에 대하여 위의 절차대로 각각 실험을 진행하였습니다. 우선 Selected Set에 대한 두 모델의 실험 결과는 그림6과 같습니다. 그리고 Random Set에 대한 두 모델의 실험 결과는 그림7과 그림8에서 Selected Set과 비교하여 그래프로 표현하였습니다. 그림7은 RandomForest 모델로 학습한 것으로 Train Accuracy가 전부 100%인 경우에 Test Accuracy의 Max와 Min을 표현하였습니다. 그림8는 LightGBM 모델로 학습한 것으로 Train Accuracy가 100%가 아닌 경우에 Test Accuracy의 Max를 표현한 것입니다. 다음의 결과들을 보시면 저희가 설계한 Deep Learning 모델로 학습하여 선정한 Selected Set이 다른 Random Set들보다 항상 높은 Test Accuracy를 보여주는 것을 알 수 있습니다. 이것을 통해 저희의 Selected Set이 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"에 가깝다는 것을 증명하였습니다.

	Max	Min
RandomForest Train	1.000	1.000
RandomForest Test	0.650	0.565
LightGBM Train	0.892	0.892
LightGBM Test	0.635	0.635

그림6. Selected Set으로 학습하여 테스트한 결과

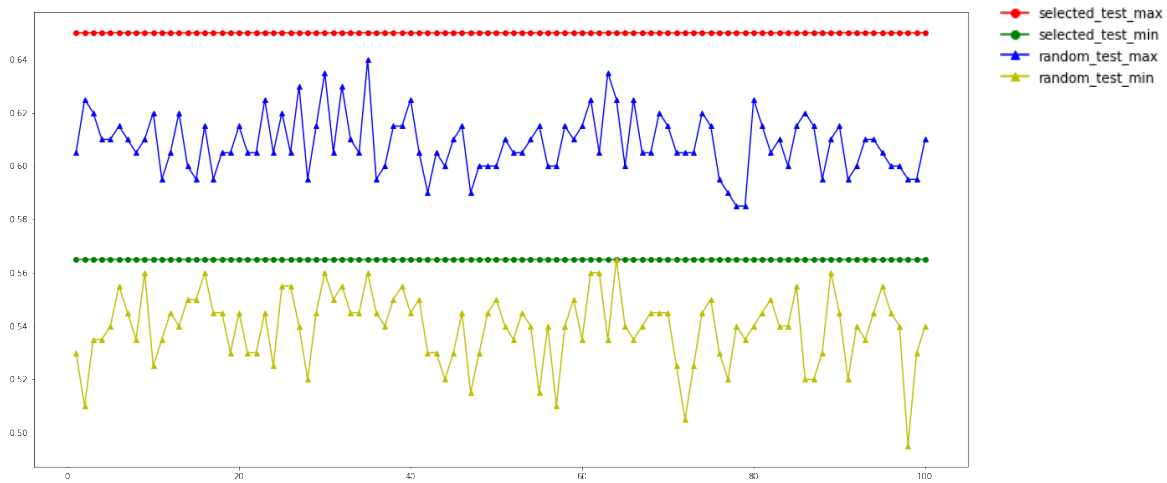


그림7. RandomForest 모델로 학습한 Train Accuracy가 100%인 경우로 빨간색과 초록색은 각각 Selected Set의 Max & Min Test Accuracy이고, 파란색과 노란색은 Random Set의 Max & Min Test Accuracy 입니다.

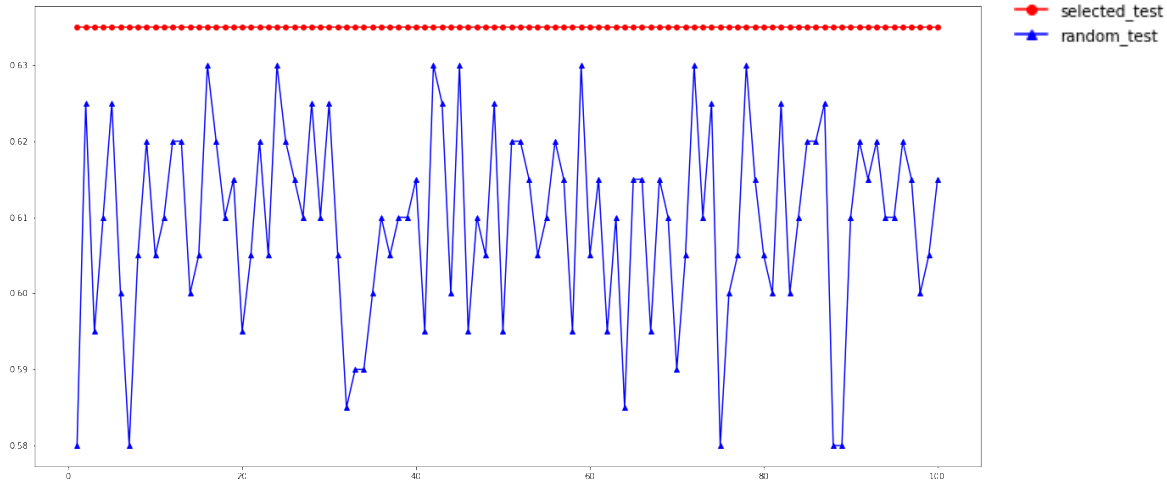


그림8. LightGBM 모델로 학습한 Train Accuracy가 100%가 아닌 경우로 빨간색이 Selected Set의 Max Test Accuracy이고, 파란색이 Random Set의 Max Test Accuracy 입니다.

Results

지금까지 저희가 이번 디지털헬스 해커톤 문제를 해결하기 위한 과정과 그 결과를 보여드렸습니다. 여러가지 다양한 Deep Learning Model 아이디어와 많은 실험을 통해서 High Performance & Explainable Deep Learning Model을 설계하였고, 이 모델을 통하여 해당 과제에서 요구하는 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"를 산출하였습니다. 여기서 끝이 아니라 저희가 뽑은 치료 효과를 증가시키는 유전자 후보가 정말 인과 관계가 확실한지 증명하기 위한 실험을 추가적으로 진행하였고, 이 실험들의 결과는 저희가 뽑은 유전자 후보가 치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보라는 것을 보여주었습니다.

저희가 생각하는 저희 방법에서 독창적인 부분은 첫번째로 기존 방식과 다른 새로운 Deep Learning Model을 설계하였다는 점입니다. Deep Learning은 사람이 잡아내기 힘든 미세한 부분까지 잡아낼 수 있는 특징이 있습니다. 데이터 개수가 적고 변수가 많은 데이터셋 상황에서 사람이 직접 통계적 데이터 분석을 통하여 데이터들의 인과 관계를 알아내기는 쉽지 않을 것입니다. 저희는 이것을 인과 관계를 찾을 수 있는 새로운 Deep Learning Model을 만들어 문제를 해결한 것이 하나의 독창적인 부분이라 생각합니다.

두번째로 Learnable Parameter를 통하여 Explainable Deep Learning Model을 설계하였다는 점입니다. Deep Learning Model은 전통적으로 Explainable하기 어려운 부분이 있습니다. 그렇기 때문에 Explainable AI라는 분야가 따로 있기도 합니다. 저희는 애초에 결과를 Explainable 할 수 있도록 모델을 설계하기 위해 고심하여 Learnable Parameter의 개념을 고안했고, 이를 통하여 인과 관계를 설명할 수 있는 유전자 후보를 골라낼 수 있었습니다.

세번째로 결과를 도출하는 것만으로 끝나는 것이 아닌 결과 세트가 정말로 인과 관계를 설명할 수 있는 세트인가를 증명하기 위한 실험 과정을 추가한 것입니다. 그리고 해당 실험 결과를 통하여 저희가 선정한 세트가 "**치료 효과를 증가시키는 인과 관계가 확실한 유전자 후보 10개**"에 가깝다는 것을 보였습니다.