1. https://www.kaggle.com/datasets/dipayanbiswas/parkinsons-disease-speech-signal-features
파킨슨 병 예측을 위한 데이터셋으로, physician's examination을 통해 얻은 환자의 speech data에 Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs) 등 다양한 알고리즘을 적용하여 임상적으로 유용한 정보를 추출한 데이터셋이다. 데이터의 개수는 756개, feature column의 개수는 754개이다.

2. 먼저 저희는 real world에서 적용가능한 주제로 수행하고자 하였습니다. 파킨슨 병 분류를 인공지능을 통해 빠르고 정확도 높게 구분할 수 있게 된다면, 부족한 수의 의료계 종사자분들의 업무강도를 확실히 낮출 수 있으며, 의료 서비스 단가를 낮추어 모든 사람들이 서비스를 누릴 수 있을 것입니다. 또한 환자의 Speech data를 이용해서 분석을 진행하기 때문에, 많은 환자들이 병원에 가지 않고도 edge device에서 간편하게 파킨슨 병의 발병을 예측할 수 있을 것으로 기대합니다. 이번 기회를 통해 분류 예측을 진행하고 분류 결과에 영향을 미치는 요소들을 직접 경험해보면서 인공지능을 익숙하게 다루고자 합니다.

3. 각 알고리즘 별 필요 단계 및 실험 결과

알고리즘	필요 단계
공통	Scaling, StratifiedShuffledSplit (imbalanced)
SVM	Kernel, Transform data
NN	drop, Best K-Choice, Distances
Naive Bayes	Probability_model selection(GaussianNB, BernoulliNB) , cross_validation_scoring(cross_val_score) cv-selection
DT	GridSearchCV(max_depth, max_features)

알고리즘	Test Accuracy
SVM(SGD)	0.7460317
NN(k=5)	0.8134920634920635
DT(GridSearchCV)	0.845815

Naive Bayes(bernoulli-model)	0.7764550264550264

각 4가지의 baseline 알고리즘으로 실험한 결과, Decision Tree 알고리즘이 가장 높은 정확도를 보였다.

보고서

1. 데이터 선택 및 분석

(https://www.kaggle.com/datasets/dipayanbiswas/parkinsons-disease-speech-signal-features)
사용한 데이터는 파킨슨 병 예측을 위한 데이터셋으로, physician's examination을 통해 얻은 환자의 speech data에 Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs) 등 다양한 알고리즘을 적용하여 임상적으로 유용한 정보를 추출한 데이터셋이다. 데이터의 개수는 756개, feature column의 개수는 754개이다. 파킨슨 병이 걸린 사람, 안 걸린 사람의 음성에서 추출된 feature에는 Baseline, Intensity, Formant, Bandwidth, Vocal Fold, MFCC, Wavelet 등 대부분 오디오 신호에서 추출되어, 미리 유효성 있는 feature들로만 전처리되어 특별한 전처리가 필요없다.
y_label로 사용될 'class' column은 파킨슨 병에 걸렸는지, 안걸렸는지에 대해 binary로 전처리되었다. 해당 데이터는 파킨슨 병에 걸린 사람의 데이터 수가 걸리지 않은 사람의 데이터 수보다 훨씬 많기에 데이터의 분포가 불균형하다.

2. NN

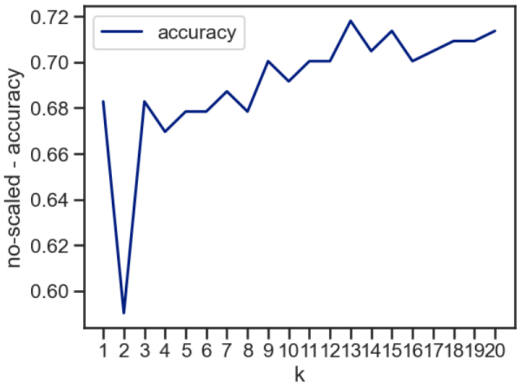
'Parkinson's Disease (PD) classification' 데이터셋을 KNN 모델로 학습시키고 테스트해보았다. KNN 모델은 이해하기에 쉽고, 모든 train data를 기억하는 학습 방식으로 fit이 빠르지만, 거리를 계산하는 방식으로 predict가 느리다는 특징이 있다. 또, boundary가 linear하기 보다 flexible해서 data가 irregular한 상황에서 더 효과적일 것이다. K를 얼마로 할 것인지, 어떤 종류의 거리 계산 방식을 사용할 것인지, 어떤 scaling을 사용할 것인지 그리고 가중치로 distance를 줄 것인지에 대한 결정이 KNN에서 결과를 다르게 나오게 만든다.

비교군에 사용한 요소는 다음과 같다.

k: 1 ~ 20
distance: Manhattan, Euclidean
scaler: no-scaled, standard, minmax, maxabs, robust

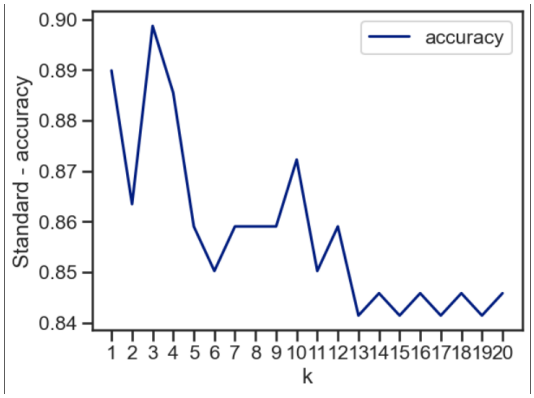
비교에 앞서, 해당 데이터셋은 domain 지식이 없는 상태에서 전처리하기가 까다로웠으므로 전처리 과정은 진행하지 않았음을 알린다. 또, train과 test data split 비율은 7:3으로 통일하였다.

(4) No-scaled



그래프와 같이 K가 13일 때 accuracy가 가장 높았으나, 최대 accuracy가 0.72로 추후에 기술한 scaling을 적용한 다른 모델보다 현저히 accuracy가 떨어짐을 보였다.

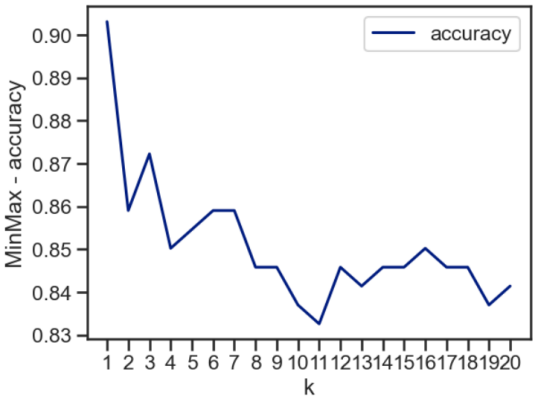
(2) Standard Scaler



그래프와 같이 K가 3일 때 accuracy가 가장 높았으며, Euclidean distance의 경우에서 더 좋은 결과를 보였다. 다음은 Standard Scaler, K=3, Euclidean distance의 경우의 결과이다.

	train	test
accuracy	0.965974	0.898678
precision	0.967980	0.890710
recall	0.987437	0.981928
f1	0.977612	0.934097
balanced accuracy	0.944100	0.827029

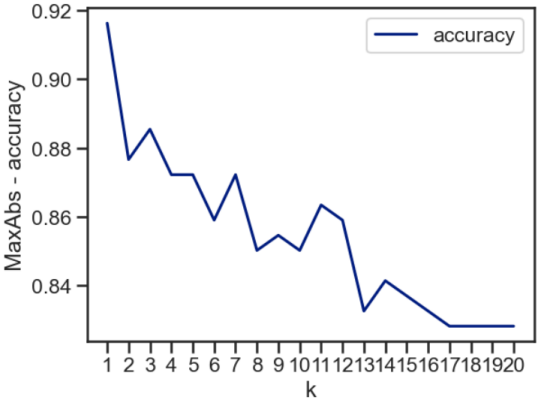
(3) MinMax Scaler



그래프와 같이 K가 1일 때 accuracy가 가장 높았으며, Manhattan distance의 경우에서 더 좋은 결과를 보였다. 다음은 MinMax Scaler, K=1, Manhattan distance의 경우의 결과이다.

	train	test
accuracy	1.0	0.911894
precision	1.0	0.945122
recall	1.0	0.933735
f1	1.0	0.939394
balanced accuracy	1.0	0.893097

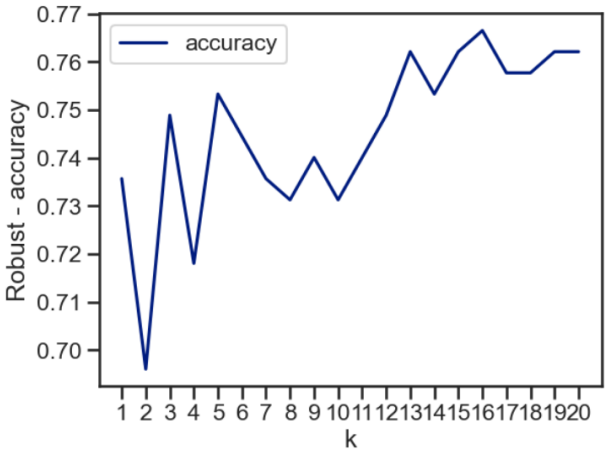
(4) MaxAbs Scaler



그래프와 같이 K가 1일 때 accuracy가 가장 높았으며, Manhattan distance의 경우에서 더 좋은 결과를 보였다. 다음은 MaxAbs Scaler, K=1, Manhattan distance의 경우의 결과이다.

	train	test
accuracy	1.0	0.938326
precision	1.0	0.952381
recall	1.0	0.963855
f1	1.0	0.958084
balanced accuracy	1.0	0.916354

(5) Robust Scaler



그래프와 같이 K가 16일 때 accuracy가 가장 높았으며, Euclidean distance의 경우에서 더 좋은 결과를 보였다. 다음은 Robust Scaler, K=1, Euclidean distance의 경우의 결과이다.

	train	test
accuracy	0.805293	0.766520
precision	0.802875	0.775610
recall	0.982412	0.957831
f1	0.883616	0.857143
balanced accuracy	0.624794	0.601866

결론적으로, best case로 k=1, Manhattan distance, maxAbsScaler인 case의 KNN이 가장 좋은 성능을 보였다. best case의 결과는 다음과 같다.

	train	test
accuracy	1.0	0.938326
precision	1.0	0.952381
recall	1.0	0.963855
f1	1.0	0.958084
balanced accuracy	1.0	0.916354

3. Naive Bayes

비교 option :

- bernoulliNB vs GaussianNB
- no-scaled data vs MinMaxScaler vs StandardScaler
- smoothing parameter : {0.01,1,3,5,7,10,100}

<best case에서의 error measure- gaussian + standard + smoothing = 8>

metrics	train	test
accuracy	0.90154	0.811423
precision	0.895321	0.820416
recall	0.913323	0.803075
f1	0.881041	0.820416
balanced accuracy	0.716011	0.694833

(1) 실험 진행 : 주어진 data에 대해서 (already encoded) 전처리로서 data scaling을 1) no scaled, 2) minmax scaling 3) standard scaling 3가지 케이스를 적용, Naive bayes model로서 BernoulliNB와 GaussianNB를 적용하여 테스트를 진행하였다.

```
#pipe lining set
pipe_1 = Pipeline([('model',BernoulliNB())])
pipe_2 = Pipeline([('scaler',MinMaxScaler()),('model',BernoulliNB())])
pipe_3 = Pipeline([('scaler',StandardScaler()),('model',BernoulliNB())])
pipe_4 = Pipeline([('model',GaussianNB())])
pipe_5 = Pipeline([('scaler',MinMaxScaler()),('model',GaussianNB())])
pipe_6 = Pipeline([('scaler',StandardScaler()),('model',GaussianNB())])
```

(2) accuracy comparison

- 현재 데이터셋이 파킨슨 병 발병의 환자에게 편향되어있기 때문에 test-train data를 나누는데 있어서 stratified k-fold를 수행하였으며, 모두 같은 tran-test set을 머신에 제공하여, cross validation을 수행한 이후, accuracy들을 비교해 보았다.
- Bernoulli + no scaling : 0.746692
- Bernoulli + MinMax : 0.746692
- Bernoulli + Standard : 0.724008
- Gaussian + no scaling : 0.746692
- Gaussian + MinMax : 0.803403
- Gaussian + Standard : 0.820416
- * Bernoulli의 smoothing은 3, Gaussian은 7일때 best를 보여줌

(3) Reasoning

- Gaussian vs Bernoulli : 모든 케이스에 대해서 GaussianNB가 BeroulliNB에 비해 월등한 결과를 보여주었다. 특히, scaling이 들어간 이후에서의 성능은 precision에서 10%, Recall에서는 10~15프로 정도의 우월성을 보여주었다. 그 이유를 짐작해 보자면, Bernoulli는 이진 데이터를 이용해 주로 텍스트와 같은 데이터 셋을 구분하는데 장점을 보이는 모델이고, GaussianNB 자체가 연속적인 numerical data를 구분하는데 더 장점을 보이는 모델이고, 현재 우리에게 주어진 데이터는 categorical은 이미 encoded되어 제거되었으며, 대부분이 float형태의 numerical data이므로 Gaussian이 더 성능이 좋은것은 당연하다.
- scaling comparison : 각각의 model에 대해서 no scaling < MinMax < Standard순으로 performance가 우월함을 확인할 수 있었다. naive bayes 알고리즘 자체가 distance, similarity가 아닌 실제 dataset에서의 통계적 확률을 사용하기 때문에 scaling에 큰 영향을 받지 않는다고 생각하였으나, 실험 결과 유의미한 차이가 발생했다.
- smoothing : naive bayes에서 발생하는 zero probability에 대한 handling을 결정하는 parameter이다. 현재 우리의 데이터셋이 모두 float형태의 numerical attribute를 가지고 있으며, 따져야 하는 attribute의 수가 755개이기 때문에 zero probability의 위험성이 상당히 높다. 그렇기 때문에 기본 handling parameter인 1에 비해 3~7배 높은 value로 zero-probability에 대해 relaxing을 해 줘야 더 좋은 성능을 얻을 수 있음을 확인할 수 있었다.

4. SVM

비교 option: no-scaling vs MinMaxScaler vs StandardScaler, 세 데이터에 대해 Linear SVM, Gaussian kernel SVM을 비교하여 테스트해 보았다. Gaussian kernel의 경우 gamma 값을 0.0001, 0.001, 0.01, 0.1, 1, 10, C 값을 0.0001, 0.001, 0.01, 0.1, 1, 10, 100으로 값의 리스트를 만들어 가장 성능이 좋은 gamma값과 C값의 조합을 찾기 위해 GridSearchCV를 사용하여 진행했다. GridSearchCV에는 balanced accuracy를 scoring parameter로 사용했으며, scaling에 따른 세가지 데이터에 대해 모두 진행했다.

<Best case error measures: SVC(gamma=0.1, C=10), MinMaxScaler>

metrics	train	test
accuracy	1.0	0.920705
precision	1.0	0.921788
recall	1.0	0.976331
f1	1.0	0.948276
specificity	1.0	0.758621
balanced accuracy	1.0	0.867476

(1) Linear SVM

우선, scaling을 하지 않은 데이터, MinMaxScaler, StandardScaler로 scaling된 데이터 3가지에 대해 LinearSVC를 이용해 train 시키고 test를 진행했다. Scaling을 하지 않은 데이터에 대해서는, LinearSVC 모델이 모든 test input에 대해 전부 하나의 값으로 prediction을 하는 모습을 확인할 수 있었다. 이 경우 recall 값과 specificity 값 중 하나는 매우 높게 나오나, 다른 하나는 0으로 나오는 것을 알 수 있었다.

MinMaxScaler와 StandardScaler로 scaling된 두 데이터의 경우 accuracy 0.85, balanced accuracy 0.8 정도의 비슷한 성능을 보였다.

(2) Gaussian kernel SVM

Gaussian kernel을 적용한 SVM의 경우에도, 우선 scaling을 적용하지 않은 데이터에 대해서, GridSearchCV를 이용해 얻은 best model의 balanced accuracy가 0.5가 나오는 것을 확인했다. 이 경우에도 결과를 분석해보니, 모든 prediction 값이 하나로 나오는 것을 알 수 있었으며, recall 값이 1.0, specificity 값이 0이 나오는 것을 확인했다.

Scaling을 한 데이터에 대해서는, best model의 parameter가 두 데이터 모두 C 값은 10으로 나왔고, gamma 값의 경우 MinMaxScaler를 거친 데이터는 0.1, StandardScaler를 거친 데이터는 0.001이 나오는 것을 확인했다. Best model의 balanced accuracy의 경우 MinMaxScaler 데이터가 0.83, StandardScaler 데이터가 0.78로 나왔다.

(3) Comparison

Linear SVM 과 Gaussian kernel SVM에서 모두 scaling 되지 않은 데이터는 모델의 prediction 값이 하나의 값으로 나오는 결과를 얻을 수 있었다. 또한, MinMaxScaler를 사용하는 것이 두 모델에서 모두 StandardScaler를 사용하는 것보다 더 좋은 결과를 도출하는 것을 알 수 있었다. StandardScaler를 사용하는 경우 Linear SVM과 Gaussian kernel SVM의 balanced accuracy가 모두 0.77정도로 비슷한 결과를 보여주었으나, MinMaxScaler를 사용하는 경우 Linear SVM의 balanced accuracy는 0.80, Gaussian kernel SVM의 balanced accuracy는 0.87 정도로 Gaussian kernel를 사용하는 것이 월등히 더 좋은 성능을 보여주었다. 데이터의 각 feature column과 label column의 correlation을 분석했을 때, 최대 값이 0.4를 넘지 않는 것을 알 수 있다. 이는 feature column을 그대로 이용하여 label을 linear하게 나누기 힘들다는 것을 알 수 있다.

5. Decision Tree

'Parkinson's Disease (PD) classification' 데이터셋을 Decision Tree 모델로 학습시키고 테스트해보았다. Decision Tree는 데이터에 있는 규칙을 학습을 통해서 자동으로 찾아내 트리 기반의 분류 규칙을 만드는 것이다. if, then, else를 기반으로 예측을 위한 규칙을 만들어 그 트리를 따라가면 결과를 도출할 수 있다. 결정 트리에서는 데이터의 어떤 기준을 바탕으로 규칙을 만들 것인지가 알고리즘의 성능을 크게 좌우하게 된다.

(1) Base Algorithm

먼저, 가장 기본 알고리즘으로 모델을 훈련시켜보았다. Decision Tree algorithm은 데이터 분산에 민감하지 않기 때문에 feature Scaling이나 preprocessing을 진행할 필요가 없어 다른 처리를 하지 않았다. 또한, 사용한 데이터가 ylabel feature의 value가 불균형하기 때문에 해당 비율을 유지하면서 데이터를 샘플링하기 위해 StratifiedShuffleSplit를 사용했다. 그 결과, 3:7로 나눈 test data와 train data의 ylabel 비가 전체 데이터와 똑같이 0.25:0.75정도로 잘 나눠진 것을 확인할 수 있었다.
데이터를 Base algorithm에 fit해 트리를 빌드한 결과, tree node는 67개, max depth는 12의 값이 나왔고 정확도는 아래와 같이 나왔다.

	train	test
accuracy	1.0	0.792952
precision	1.0	0.867470
recall	1.0	0.852071
specificity	1.0	0.620690
balancedAccuracyScore	1.0	0.736380
f1	1.0	0.859701

결과를 분석해보면, train dataset의 정확도는 모두 1.0으로, test dataset의 정확도보다 훨씬 높은 것으로 볼 수 있다. 이는 Decision Tree의 문제점인 과적합으로 인한 것으로 보인다.

(2) Improved Algorithm

처음에 실행한 base algorithm을 개선시키기 위해 문제점을 분석하고, 성능을 향상시켜보겠다. Decision Tree의 마지막 노드의 결과를 도출할 때 100% pure한 정답을 도출하려면 트리의 깊이가 깊어지고 overfit이 생길 수밖에 없다. 따라서 한 번 트리를 빌드시킨 후 prune 하는 과정을 거쳐야한다. 위에서 base 알고리즘으로 트리를 빌드시켜보니 과적합의 문제가 있었다. 따라서 먼저 GridSearchCV를 이용해 max_depth와 max_features 값을 조정해보며 tree를 pruning해보았다.
데이터를 최대깊이와 최대 피처를 조정하며 pruning한 결과, tree node는 31개, max depth는 5의 값이 나왔고 정확도는 아래와 같이 나왔다.

	train	test
accuracy	0.950851	0.845815
precision	0.946731	0.872222
recall	0.989873	0.928994
specificity	0.835821	0.603448
balancedAccuracyScore	0.912847	0.766221
f1	0.967822	0.899713

결과를 분석해보면, 기존 base algorithm과 비교했을때보다 유의미한 성능개선을 보여주었지만, 더 개선할 수 있는 방법을 고민했다. 이에 다른 파라미터들도 함께 조정하고, max_depth도 2만큼 늘리는 것이 아닌, 1만큼 늘려서 많은 경우의 GridSearchCV를 시행하였다.
GridSearchCV를 이용해 교차검증과, 최적의 하이퍼파라미터 튜닝을 하고자 했다.
데이터를 최대깊이, 최대 피처, 노드를 분할하기 위한 최소한의 샘플 데이터 수, 말단 노드가 되기 위한 최소한의 샘플 데이터 수를 조정하며 pruning한 결과, tree node는 37개, max depth는 6의 값이 나왔고 정확도는 아래와 같이 나왔다.

	train	test
accuracy	0.956522	0.828194
precision	0.947115	0.857143
recall	0.997468	0.923077
specificity	0.835821	0.551724
balancedAccuracyScore	0.916645	0.737401
f1	0.971640	0.888889

노드를 분할하기 위한 최소한의 샘플 데이터 수, 말단 노드가 되기 위한 최소한의 샘플 데이터 수를 파라미터로 조정하고, 안하고는 큰 유의미한 차이를 보이지 않았다. 가장 중요한 max depth를 조정해본 결과, 두번째 알고리즘보다 향상된 결과를 도출할 수 없었는데, 이는 train을 시킬 때 Bruteforce 방법이 아닌, 각 노드에서의 최적값을 찾아내는 휴리스틱 기법인 탐욕법을 기반으로 하고 있어 잘못 search 된 것으로 보인다. 또한 파킨슨병에 실제로 이환된 사람이 검사를 받았을 때 양성 판정을 받는 비율인 민감도와 이환되지 않은 정상인이 검사를 받았을 때 음성 판정을 받는 비율인 특이도가 매우 낮은 수치를 보인다.

(3) Result
모든 알고리즘 결과에서 볼 수 있듯이, 해당 데이터에서는, 파킨슨 병에 걸린 사람의 데이터 수보다 걸리지 않은 사람의 데이터 수가 훨씬 많기에(사용한 데이터는 약 1:3의 비율) 실제 True인 것 중에서 모델이 True라고 예측한 recall score가 다른 score보다 가장 높은 것을 볼 수 있었다. 또한, 데이터의 label이 불균형 구조이므로, Precision과 recall score의 조화평균인 F1 score이 모델의 성능을 가장 잘 표현한다고 할 수 있다.
따라서 가장 좋은 성능은 tree node가 31개일때, max depth가 5일때로, Base algorithm보다 단순한, pruning된 트리 구조가 도출되었다.

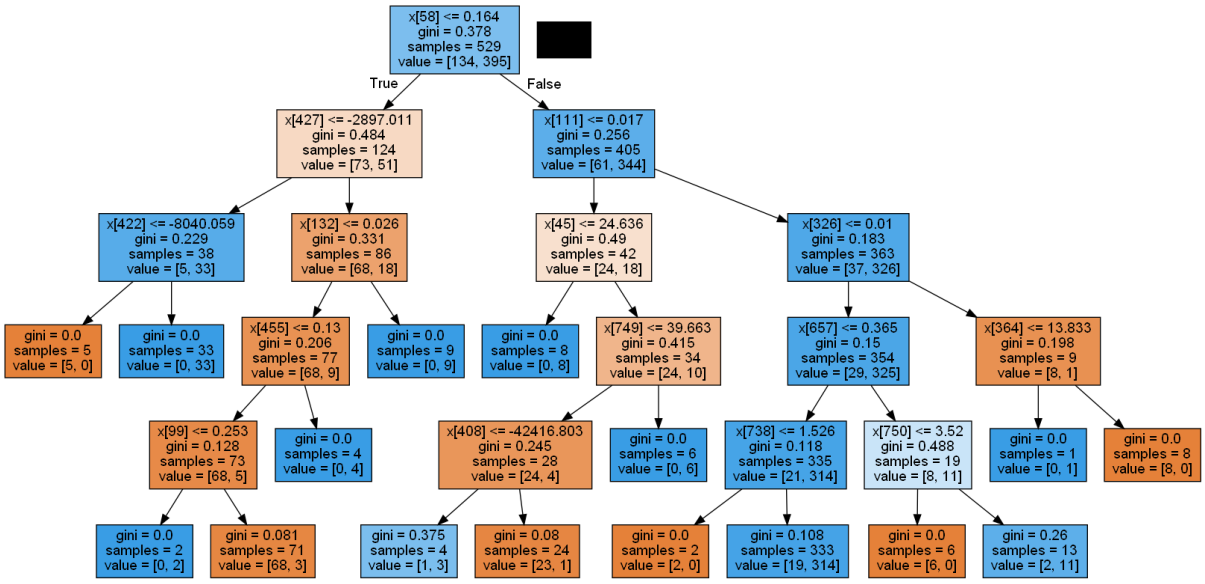
Decision Tree는 해석하기 쉽고, 트리를 그릴 수 있어서 시각적인 inspection이 가능하다는 장점이 있다. 또한, 이후 추가 개발에서 insight를 얻기 쉽기 때문에, 이와 같은 의료 데이터의 경우, 정확도를 우선하는게 아닌 분석을 위해서라면 사용하는 데 큰 이점을 가진다.

6. Conclusion

4가지 NN, naive bayes, svm, decision tree 알고리즘으로 데이터를 훈련시키고 테스트한 결과를 비교해보았다. 데이터가 의료 데이터이기 때문에, accuracy, precision, recall, f1 score, specificity, balanced accuracy 중 balanced accuracy score + recall이 가장 모델의 성능을 표현하기 좋다고 판단했다.
이에 따라 가장 성능이 좋은 알고리즘은 recall score에서 0.952663, balanced accuracy에서 0.916354의 지표를 가지는 k-NN이었다.
k-NN이 가장 높은 성능을 보이고, 나머지 알고리즘들이 그에 비해 낮은 성능을 보인 이유는 다음과 같다.

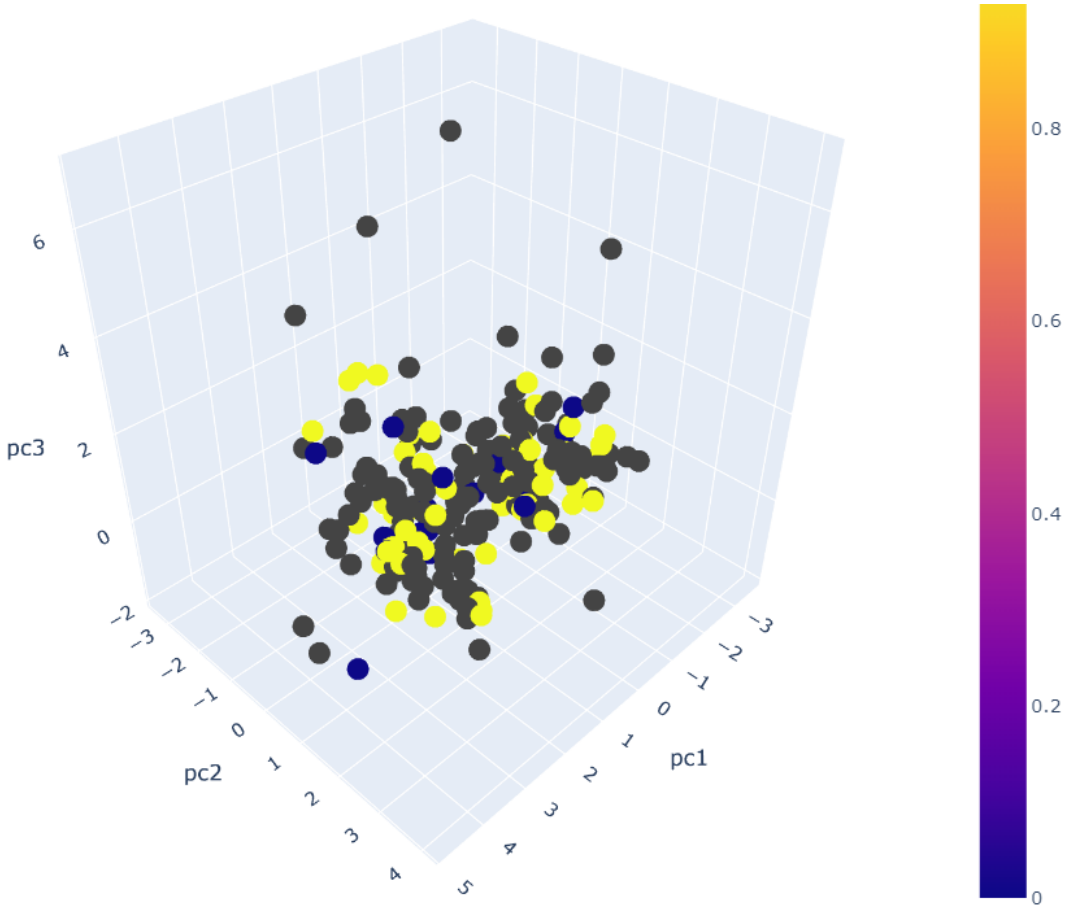
1) Naive Bayes : naive bayes는 2가지 이유에서 parkinson disease problem에 적합하지 않음을 확인할 수 있었다. 첫번째 이유로, 현재 주어진 dataset의 차원이 굉장히 높고, 모든 column이 numerical value를 가짐을 들 수 있다. naive bayes는 각각의 column들에 대하여 각각의 조건부 확률을 확인, 이를 종합하여 최종적인 input pattern에 대한 판단을 실시하게 된다. 이 때 pattern의 차원이 높아질 경우 각각에 곱연산되는 attribute들의 p(x|class)의 개수가 증가하게 된다. 이 때 각각의 p(x|class)는 relaxing을 안한다고 가정시 0이상 1이하의 값을 가지게 되고, 곱해지면 곱해질 수록 전체 pattern의 확률값은 낮아지게 된다. 모든 확률의 값이 낮아지게 되면서 확률 비교의 power가 낮아지게 되어 오류가 발생할 가능성이 높아지게 된다. 현재 우리에게 주어진 data의 pattern의 수는 754개로 pattern의 개수에 대비(756) 상당히 많은 것을 확인할 수 있다. 또한 모든 attribute value들이 numerical attribute임을 확인할 수 있는데, numerical attribute에 대해 p(x|class)를 적용하게 될 경우, numerical의 continuous로 인해 일반적인 categorical에 비해 훨씬 낮은 값이 나올 수 밖에 없다.
   두번째 이유로는 naive bayes의 '모든 attribute는 독립적이다'라는 가정으로 인해 오류율이 높아졌다는 것을 들 수 있다. naive bayes는 pattern의 조건부 확률을 계산하기 위해 모든 attribute의 조건부 확률을 독립적으로 산출, 곱연산한다. 이 경우 각 attribute간의 relationship은 전부 무시되게 되고 단순 1개 attribute의 조건부 확률만을 pattern의 판단에 고려한다. 그러나 우리에게 주어진 데이터를 분석해 보면 1) 각각의 attribute 하나 하나와 label간의 correlation은 매우 낮고 2) attribute사이의 correlation이 상당히 높음을 확인할 수 있다. 이런 데이터 셋에 naive bayes의 가정을 적용하게 될 경우, 판단에서 높은 비중을 가지는 attribute간의 correlation이 무시되게 되어 accuracy를 낮추는 원인이 된다.

2) Decision Tree: Decision Tree가 Best algorithm에 비해 성능이 나오지 않았던 이유는, 각 노드에서의 최적값을 찾아낼 때 휴리스틱 기법인 탐욕 알고리즘을 기반으로 하고 있어 최적 결정 트리를 알아낸다고 보장할 수는 없기 때문이다.
또한, 전체적인 Decision Tree 알고리즘 결과를 봤을 때, train data의 성능이 test data보다 훨씬 높은 것을 볼 수 있고, 완성된 트리를 이미지로 저장하여 출력해보면, 복잡한 트리 구조를 볼 수 있다. 이는 Decision Tree의 단점인 과적합으로 인해 높은 성능을 낼 수 없다는 것을 보여준다.



3) SVM: SVM이 Naive Bayes나 Decision Tree보다는 조금 더 나은 성능을, k-NN보다는 덜 좋은 성능을 보였다. 우선 Naive Bayes와 Decision Tree보다 좋은 성능을 보인 이유는, 두 알고리즘과는 달리 SVM에서는 데이터를 higher dimension으로 transform 한다는 점이다. 데이터의 feature를 그대로 사용하는 것으로는 prediction의 성능이 나오지 않지만, feature transformation을 적용했을 경우 higher dimension에서 데이터의 label이 잘 분류되는 것을 각 feature와 label의 correlation이 높지 않다는 점에서 유추할 수 있다. 반대로, k-NN보다 성능이 낮은 이유는, SVM이 black-box의 형태를 하고 있어 정확한 분석은 어려우나, 몇가지 이유를 유추할 수 있다. 우선, 데이터의 개수가 적은 것에 비해 높은 차원을 가진다. k-NN은 input data point 가장 가까운 data point를 찾으면 되지만, SVM의 경우 이상적인 hyper-plane을 training dataset에서 찾아야 하는데, 알맞는 support vector와 이상적인 hyper-plane을 찾기에는 data의 개수가 적다는 사실을 알 수 있다. 또한, 이론적으로 data가 noisy할 경우, 즉 label이 많이 겹쳐있는 경우 SVM의 성능이 좋지 않다는 점이 있다. 데이터의 차원이 매우 높아 noisy하다는 사실을 밝히기는 어려우나, k-NN의 성능이 더 잘 나온다는 점에서 데이터가 noisy하기 때문에 hyper-plane을 찾기 어렵다는 점을 유추할 수 있다.

4) KNN: 그렇다면 KNN이 좋은 성능을 보인 이유로는,



그림과 같이 PCA를 이용해 차원을 3차원으로 drop시킨 뒤 data plotting을 진행해보았을 때, 데이터의 분포가 불균형하고 irregular하다는 사실을 알 수 있었다. KNN이 기본적으로 연산 비용이 매우 높지만, 실험을 진행한 데이터 개수가 적은데다, 데이터의 노이즈에 영향을 크게 받지 않고 irregular한 데이터셋에 강하다는 KNN의 특성 상, 다른 모델보다 좋은 성능을 낼 수 있었다.

결론적으로, 파킨슨 병 분류에 k-NN 알고리즘을 사용하면, 병의 유무의 진단을 빠르고 정확도 높게 할 수 있을 것이다. 이는 부족한 수의 의료계 종사자분들의 업무강도를 확실히 낮출 수 있으며, 의료 서비스 단가를 낮추어 모든 사람들이 서비스를 누릴 수 있을 것이다. 또한 환자의 Speech data를 이용해서 분석을 진행하기 때문에, 많은 환자들이 병원에 가지 않고도 edge device에서 간편하게 파킨슨 병의 발병을 예측할 수 있을 것으로 기대된다.