이진 분류 모형에서 잘못 라벨링된 데이터의 영향

엄석훈
서강대학교
tommya98@naver.com

Abstract

이 프로젝트는 두개의 카테고리로 분류하는 이진 분류 문제에서 잘못 라벨링된 데이터가 여러 분류 모형의 결과에 미치는 영향을 확인하기 위해 진행된 프로젝트입니다. 기본적인 분류 모형인 KNN, SVM, Tree 모델에 대해서 그리드 서치로 하이퍼파리미터에 변둥을 주고 오라벨된의 데이터의 비율을 바꿔가며 모델의 예측 성과를 비교하였습니다. 그 결과 오차율이 증가할 때 모델의 성과가 나빠지는 당연한 결과를 얻을 수 있었습니다. 다만 오차율이 증가하였을 때 같은 모델에서도 조금 더 데이터에 둔감하게 반응하도록 파라미터를 조절한 모델에서 조금 더 좋은 성과를 확인할 수 있었습니다. 또한 오차율이 10%정도로 크게 증가하더라도 이진 분류에서는 모델의 성과가 확 나빠지는 모습을 보이지는 않았습니다. 즉 어느 정도의 오차가 예상되는 데이터셋에서는 파라미터의 조절을 통해 제약을 가해 모델의 성과를 높일 수 있다는 결론을 얻을 수 있었습니다.

1. 서론

1.1 프로젝트 배경

최근 머신러닝이 중요해지면서 데이터의 중요성도 커지고 있다. 잘 동작하는 머신러닝 모델을 만들기 위해서는 그에 맞는 올바른 데이터가 중요하다. 그러나 MIT, Cleanlab의 최근 연구[1]에 따르면 모델의 성과 벤치마크 측정으로 자주 사용되는 데이터 셋에서 약 3.3% 이상의 라벨링 오류가 발견되었다. 또한 우리가 일반적으로 접할 수 있는 다양한 데이터 셋에서도 충분히 잘못 라벨링된 데이터가 존재할 수 있다.

이러한 상황에서 잘못 라벨링된 데이터를 올바르게 수정하기란 매우 어려운 작업이다. 또한 일반적으로 데이터 셋에 대한 완벽한 이해가 있지 않는 한 잘못 라벨링된 데이터를 식별하는 것 또한 어렵다. 이러한 상황에서 최선의 대처는 모델과 데이터의 특성을 보다 잘 이해하고 잘못 라벨링된 데이터가 있는 상황에서도 잘 작동하는 모델을 사용하는 것 일것이라 생각하여 프로젝트를 시작하게 되었다.

1.2 프로젝트 목적

프로젝트의 목적은 잘못 라벨링된 데이터가 있는 데이터 셋을 가지고 여러 이진 분류 모형에 적용하여 그 반응을 살펴보는 것이다. 이를 위해 일반적으로 이진 분류에서 쉽게 사용할 수 있는 KNN, SVM, Tree 모델에 적용하여 각 모델의 성능 변화를 살펴보고 각 모델의 특성을 보다 잘 이해하는 것이 프로젝트의 가장 큰 목적이다.

또한 이 과정에서 데이터의 오류가 예상되는 상황이나 조금 더 안전하게 분류 모형을 사용하고 싶을 때 어떤 분류 모형이 더 잘못 라벨링된 데이터에 강한 내성을 보이는지 알아보고 이러한 상황에서 모델을 선택하는데 도움을 주는 것 또한 프로젝트의 추가적인 목적 중 하나이다.

1.3 프로젝트 진행 방법

하나의 데이터셋을 가지고 잘못 라벨링된 데이터가 없을 때부터 비율을 늘려가며 최대 10%정도의 잘못 라벨링된 데이터가 있는 경우에 각 모델을 학습시키고 결과를 살펴본다. 다만 실제 프로젝트에서 사용하는 데이터셋 또한 라벨링이 잘못된 경우가 있을 수 있지만 이는 무시하고 진행한다. 또한 데이터의 특성에 따라 잘못 라벨링된 데이터의 편향이 있을 수 있지만 이 또한 무시하고 랜덤하게 데이터셋에서 비율을 정해 class를 반대로 뒤집어 준다.

프로젝트에서 사용하는 데이터셋은 Kaggle의 Airline Passenger Satisfaction 데이터셋[2]으로 비행기 탑승 손님의 만족, 불만족 여부와 관련 feature로 이루어진 데이터셋이다. 데이터셋의 자세한 분석은 2. 데이터 분석에서 살펴본다. 해당 데이터 셋을 분석하여 각 모델에서 알맞게 사용할 수 있는 데이터 형태로 바꿔준다. 그리고 그 상태에서 class에서만 특정 비율로 오류를 발생시킨다. 그리고 모델에서 그리드 서치로 하이퍼파라미터를 바꿔가며 학습시키고 그 결과를 살펴보고 결과를 분석한다. 이후 추가적으로 프로젝트 목적에 따라 잘못 라벨링된 데이터의 비율이 늘어갈 때 각 모델에서 어떤 방법으로 성과를 늘릴 수 있는지 또한 확인해본다.

1.4 프로젝트 예상 결과

잘못 라벨링된 데이터라는 것을 조금만 바꿔 생각해보면 이상치라고도 생각할 수 있다. 데이터 하나하나에 민감해서 이상치에 영향을 많이 받는 모델은 마찬가지로 잘못 라벨링된 데이터에도 영향을 많이 받을 가능성이 크다. 다시 말하면 오버피팅에 취약한 모델들이 잘못 라벨링된 데이터에 취약할 것이라고 생각된다. 이러한 점으로 보았을 때는 Tree모형이 일반적으로 오버피팅에 취약함으로 가장 잘못 라벨링된 데이터에 영향을 많이 받을 것이고 SVM과 KNN은 Tree모형에 비해서는 일반적으로 오버피팅에 강하기 때문에 잘못 라벨링된 데이터에 영향을 덜 받을 것이라고 생각된다.

또한 각 모델들은 파라미터에 따라서 제약이 걸려 데이터에 민감하거나 둔감한 정도를 어느정도 결정할 수 있다. 따라서 같은 모델이더라도 옵션에 따라서 오히려 잘못 라벨링된 데이터가 증가할 때 모델의 예측 결과도 좋아질 수 있다고 생각된다. 즉 일부 잘못 라벨링된 데이터는 모델의 오버피팅을 줄이고 데이터에 둔감하게 해 오히려 긍정적인 효과를 줄 수도 있을 것이라 생각된다.

2. 데이터 분석

2.1 데이터 개요

프로젝트에서 사용할 데이터는 비행기 탑승 만족도 설문조사에 관한 데이터이다. 예측하고 싶은 내용은 만족도이고 이와 관련하여 비행 관련정보와 다른 설문 내용이 데이터로 주어진다. 이번 프로젝트를 위해 이 데이터셋을 선택한 이유는 쉽게 구할 수 있는 binary classfication을

위한 데이터셋이기 때문이다. 또한 프로젝트의 목표가 라벨링 에러의 모델에 대한 영향을 살펴보는 것이 프로젝트의 가장 큰 목표이기 때문에 적당히 큰 데이터셋에 특이한 부분이나 결측치, 이상치가 없이 온전히 라벨링 에러에 의한 영향을 살펴보기에 좋은 데이터셋이라 생각했기 때문이다. 마지막으로 미리 train데이터와 test데이터가 분리되어 있어 test데이터는 전혀 보지 않고 작업을 할 수 있다는 장점도 있는 전반적으로 잘 정리된 데이터셋이라 선택하게 되었다.

2.2 Class, Feature 분석

우리가 예측하고 싶은 class는 satisfaction으로 비행기 탑승 후 만족도 조사의 결과이다. Satisfaction과 neutral or dissatisfaction로 2 개의 class를 가지고 있다. 아래 [그림 1]과 같이 정확히 반반은 아니지만 비슷한 분포를 가지고 있다.

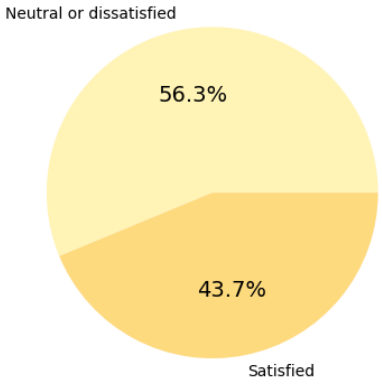

그림 1 class 분포

그 외에 총 23 개의 feature를 가지고 있는데 그 중에서 숫자형 feature는 19 개로 [그림 2]에서 각 feature의 분포를 확인할 수 있다. 이 중에서 id, Age, Flight Distance, Departure Delay in Minutes, Arrival Delay in Minutes까지 5 개의 연속형 feature를 가지고 있다. 그 중에서 id는 예측과는 무관함으로 무시한다. 이 중에서 Flight Distance는 right skewed되어 있어 추후 보다 정확한 예측을 위해 로그 변환을 적용할 수 있다. 그리고 Departure Delay in Minutes와 Arrival Delay in Minutes는 대부분이 0 이고 둘은 [그림 3]에서 확인할 수 있는 것처럼 두 변수가 매우 높은 선형 상관관계를 가지는 것을 확인할 수 있다. 따라서 하나의 feature를 제거하고 로그 변환을 적용할 수 있다.

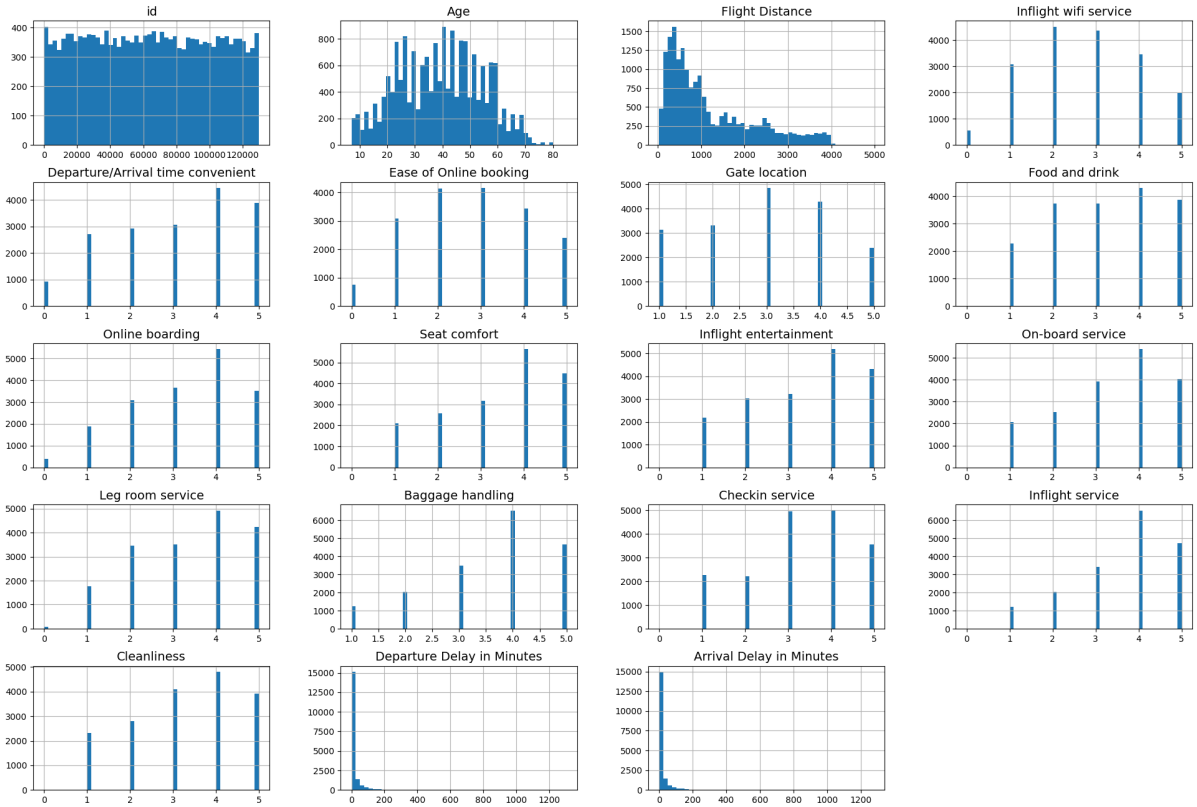

그림 2 숫자형 feature 분포

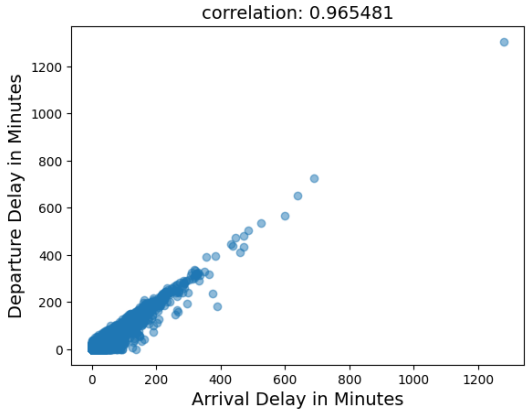

그림 3 두 Delay의 다중공선성

나머지 숫자형 feature는 모두 5 개의 카테고리를 가지는 만족도와 관련된 설문 데이터이다. 이들은 모두 이산분포를 띄고 있기 때문에 one-hot-encoding을 사용해서 여러 feature로 분리해도 되지만 만족도 설문이기 때문에 해당 점수의 차이가 유의미하다고 생각되어 특별한 전처리 없이 그대로 사용해도 괜찮은 데이터라고 생각된다.

다음으로 나머지 범주형 feature는 Gender, Customer Type, Type of Travel, Class가 존재한다. 각 feature별로 분포는 아래 [그림 4]와 같이 이루어져 있는데 성별은 반반으로 특별한 특징이 없다. 다른 눈여겨볼 점으로는 많은 승객이 loyal customer로 어떻게 보면 당연한 이야기지만 계속해서 항공사를 이용하는 승객이 많은 것을 알 수 있다. 또한 실제 비행기에서 class별 좌석 수에 대한

사전 정보는 없지만 아마도 데이터의 비율과 유사할 것이라 생각된다. 그 외에도 특별한 부분은 없어 범주형 데이터에서 class만 one-hot-encoding을 통해 처리하고 나머지 feature는 0 과 1 로 label-encoding을 사용해서 처리할 수 있을 것이다.

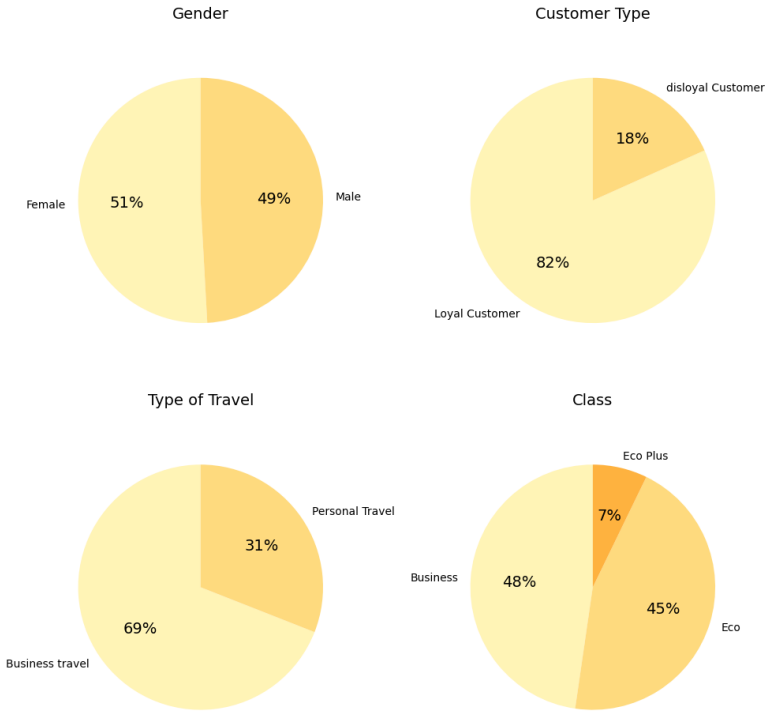

그림 4 범주형 feature 분포

2.3 데이터 전처리 과정

먼저 불필요한 데이터를 정리하기 위해서 id는 삭제해주었다. 또한 데이터에서 Arrival Delay in Minutes라는 feature에서 310 개의 결측치가 있어 해당 데이터는 제외하였다. 그 결과 train데이터로 103594 개의 데이터가 남아있게 되었다.

다음으로 위에서 살펴보았던 범주형 feature에 대해서 범주가 2 개인 경우는 label-encoding으로 그 이상인 경우는 one-hot-encoding을 통해 범주형 feature를 정리하였다. 이를 통해 [그림 5]와 같이 satisfication을 제외하고 총 24 개의 feature를 가지는 데이터셋으로 정리하였다.

```
Index(['Gender', 'Customer Type', 'Age', 'Type of Travel', 'Flight Distance',
       'Inflight wifi service', 'Departure/Arrival time convenient',
       'Ease of Online booking', 'Gate location', 'Food and drink',
       'Online boarding', 'Seat comfort', 'Inflight entertainment',
       'On-board service', 'Leg room service', 'Baggage handling',
       'Checkin service', 'Inflight service', 'Cleanliness',
       'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction', 'Class_Business', 'Class_Eco', 'Class_Eco Plus'],
      dtype='object')
(17917, 25)
```

그림 5 범주형 데이터 처리후 feature

다음으로 숫자형 데이터에서 Flight Distance는 right-skewed되어 있음으로 로그 변환을 취해주었다. 또한 두가지 Delay는 상관관계가 매우 높기 때문에 하나를 제거해 임의로 Arrival

Delay in Minutes는 삭제하고 Departure Delay in Minutes만 남겨주었다. 로그 변환 후에도 log_Departure Delay in Minutes는 0 의 분포가 매우 많은 것을 확인할 수 있다. 따라서 이를 범주형으로 바꿔 Delay가 있는 경우와 없는 경우로 나눠주었다. 그리고 마지막으로 Age와 log_Flight Distance의 scale을 맞추기 위해 표준화를 진행하였다.

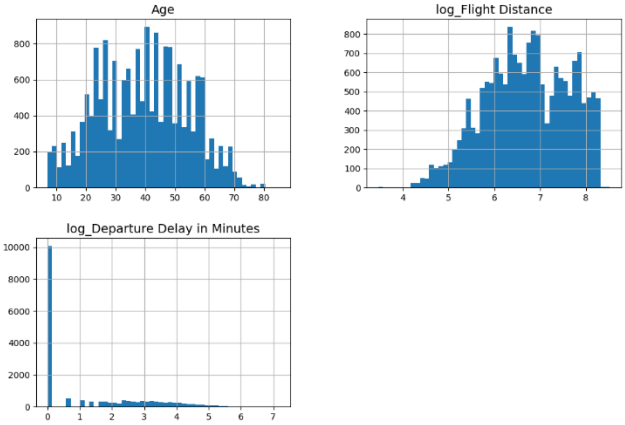

그림 6 숫자형 데이터 처리후 분포

2.4 최종 전처리 데이터 결과

전처리 과정에서 불필요한 feature를 삭제하고 결측치를 제거하였다. 그리고 범주형 데이터의 라벨링을 바꿔주고 숫자형 데이터를 skew되지 않도록 하고 표준화하그 그 과정에서 숫자형 데이터를 범주형 데이터로 바꾸기도 하였다. 그 결과 [그림 7]과 같이 총 23 개의 feature를 가지게 되었다.

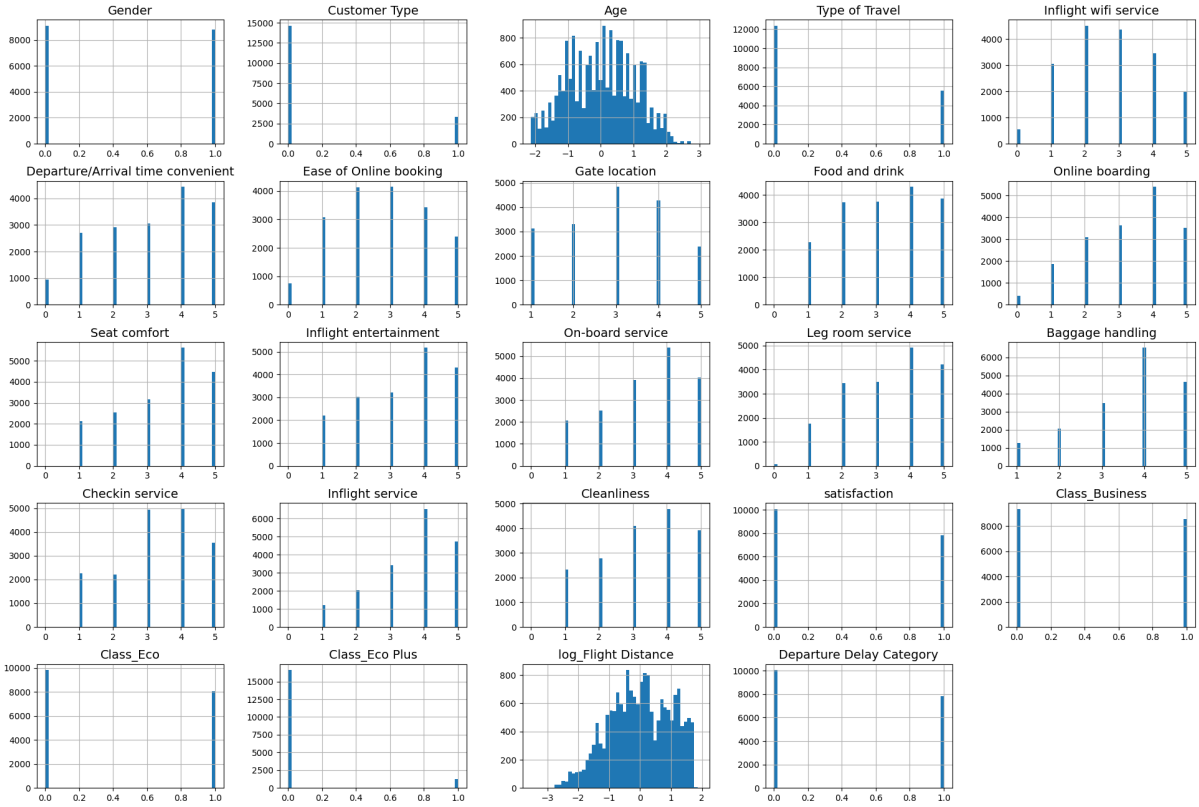

그림 7 최종 feature 분포

3. 모델 적용 내용

3.1 모델 하이퍼파라미터 선정

먼저 라벨링 오류가 없는 경우에 그리드 서치를 사용해서 가장 훌륭한 성과를 보여주는 파라미터의 조합을 찾아보았다. 사용한 모델은 KNN, SVM, Tree 모형이다.

KNN의 경우 인접한 데이터의 개수 K인 n_neigbors를 5, 10, 20, 50, 100 으로 점점 늘려가며 사용하도록 하였다. 또한 p는 1 과 2 를 사용해 거리를 계산할 때 Manhattan distance와 Euclidean distance를 비교하여 사용하도록 파라미터를 결정하였다. 데이터의 개수가 꽤 크기 때문에 KNN에서 K에 해당하는 값을 100 까지 크게 사용하도록 하였다. 하지만 그 이상도 해보고 싶었지만 너무 연산이 오래걸려 시간상 진행하지 못하였다.

SVM의 경우 규제 정도를 나타내는 C값을 0.1 과 1 을 사용하여 일반적인 경우의 1 과 오버피팅을 많이 제한한 0.1 두가지를 확인해 보았다. 또한 결정경계의 부드러운 정도를 조절하는 gamma는 기본인 scale과 1/n_features값인 auto 두가지를 사용하였다. 그리고 kernel은 가우시안 커널 rbf를 사용해주었다. C값을 조금 더 세분하거나 다른 커널을 사용하거나 gamma값도 실수로 주어 확인해보고 싶었지만 SVM도 연산량이 너무 많아 우선은 4 가지 경우만 확인해 보았다.

마지막으로 Tree모형은 연산이 다른 두 모델에 비해서 빨라 max_depth를 3, 5, 10, 15 까지 다양하게 해보고 min_samples_split은 10, 30, 50, 100, 300, 500, 1000 까지 매우 다양하게 확인해 보았다.

3.2 라벨링 오류가 없는 경우

라벨링 오류가 없는 경우 위에서 이야기한 하이퍼 파리미터를 이용해 그리드 서치를 수행한 결과 KNN, SVM, Tree 모델에서 각각 [그림 8]과 같은 최적의 파라미터와 그에 따른 모델의 예측 결과를 보여주었다.

KNN은 K가 5 이고 Manhattan distance를 사용하는 경우에 가장 accuracy가 0.929 정도로 좋게 나왔다. SVM은 C가 1 이고 gamma가 auto일 때 0.946 정도로 상당히 높게 나왔다. 마지막으로 Tree 모형은 max_depth는 10, min_samples_split는 100 일 때 0.939 정도로 유사하게 높게 나왔다. KNN에서는 데이터가 10 만개가 넘는데 K가 5 밖에 안되는 것으로 보아 오버피팅이 조금 의심된다. 마찬가지로 Tree 모형에서도 depth가 10 으로 약간의 오버피팅이 조금 의심된다. SVM은 정확히는 알기 힘들지만 모든 모델이 이후 동일한 파라미터에서 결과를 살펴보면 조금 더 분석이 가능할 것으로 보인다. 그래도 전반적으로 전부 모델의 성과과 90%넘게 나와 좋은 성과를 보여주었다.

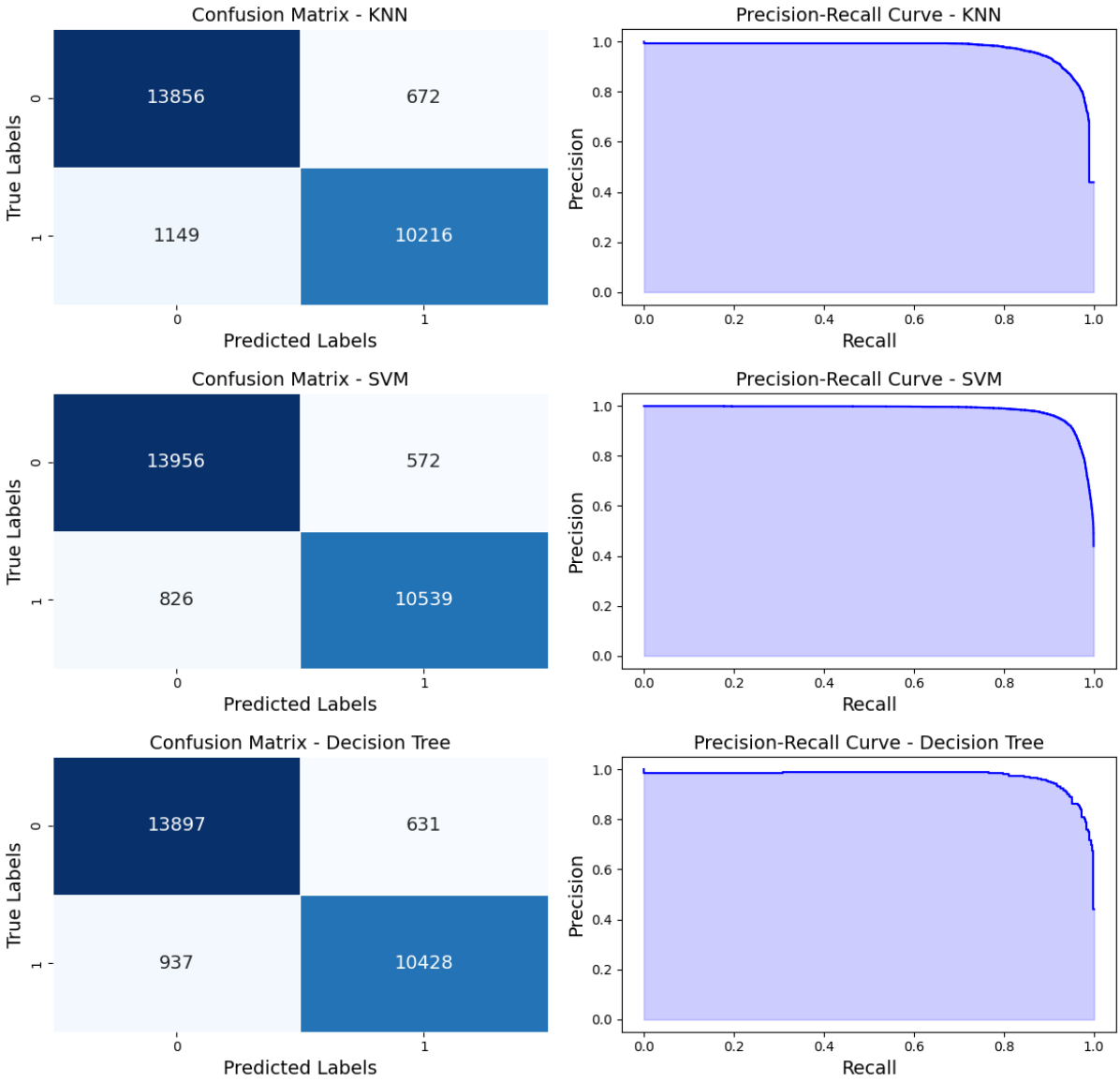

그림 80% 오류 그리드 서치 결과

3.3 라벨링 오류 1% 적용

랜덤한 라벨링 오류를 1% 적용한 경우에 오류가 0%일 때와 파라미터를 동일하게 선택한 경우 [그림 9]와 같은 결과가 나왔다. Accuracy는 KNN이 0.928, SVM이 0.945, Tree가 0.939 정도로 KNN과 SVM은 아주 조금 성과가 떨어지고 Tree는 0.939 로 비슷한 성과가 나왔다. 살펴볼만한 특이한 점으로는 KNN에서 FP는 그대로이고 FN만 증가하였다. 이 경우 물론 예측 성과는 떨어졌지만 오퍼피팅이 감소하였다고도 해석할 수 있을 것 같다. 또한 Tree모형은 FP는 감소하고 FN이 증가하는 모습을 보였는데 이 또한 모델이 조금 더 균형있게 예측하는 것으로 보아 오버피팅이 감소하였다고 생각할 수도 있다.

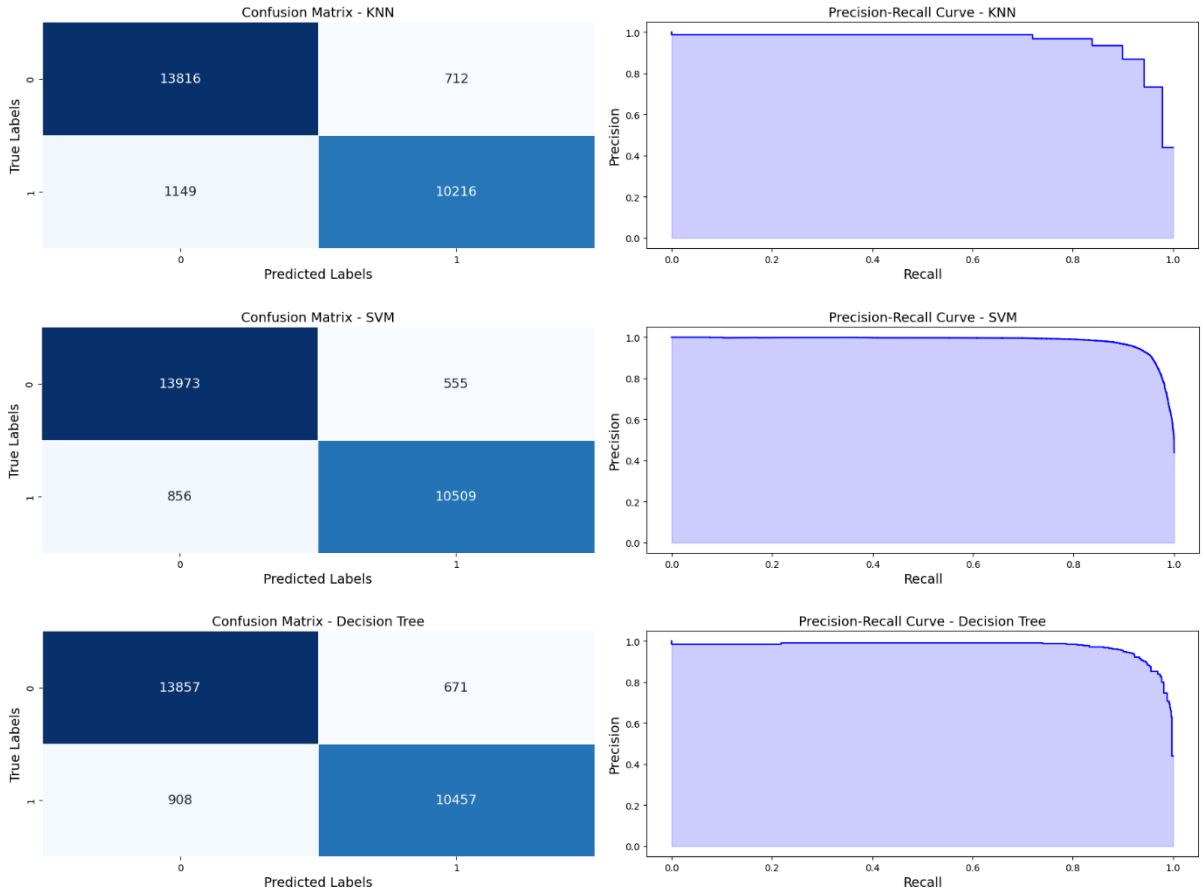

그림 9 1% 오류 동일 파라미터 결과

다음으로 오류가 1%일 때 그리드 서치를 통해 살펴본 결과 KNN과 SVM은 동일한 파라미터를 보여주었다. 반면 Tree모델의 경우 max_depth: 15, min_samples_split: 50 으로 기존 파라미터보다 더 오버피팅된 파라미터과 결과로 [그림 10]과 같이 나왔다. 하지만 accuracy는 0.939 정도로 동일하였고 기존보다 FP가 증가하고 FN이 감소하여 모델이 보다 positive로 예측을 많이 하게 동작하는 모습을 확인할 수 있다.

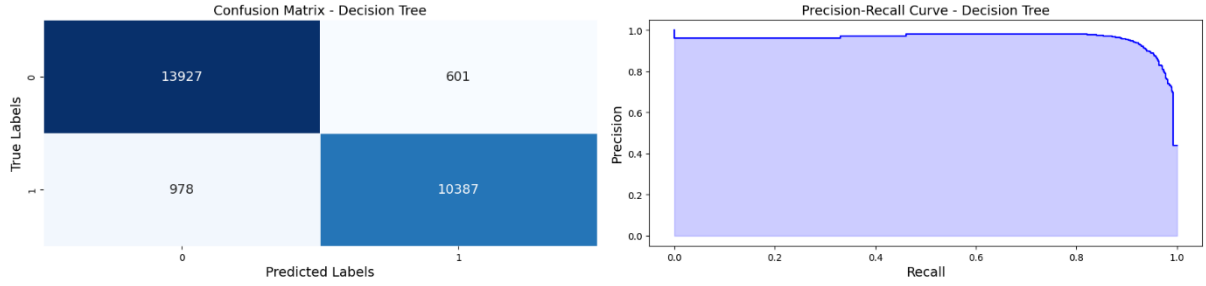

그림 10 1% 오류 Tree 모델 그리드 서치 결과

3.4 라벨링 오류 3% 적용

랜덤한 라벨링 오류를 3% 적용한 경우 동일한 파라미터를 적용한 결과는 [그림 11]처럼 accuracy가 각각 KNN이 0.926, SVM이 0.944, Tree가 0.938 로 KNN과 SVM, Tree 모델 모두 accuarcy가 아주 소폭씩 감소한 모습을 보여주었다.

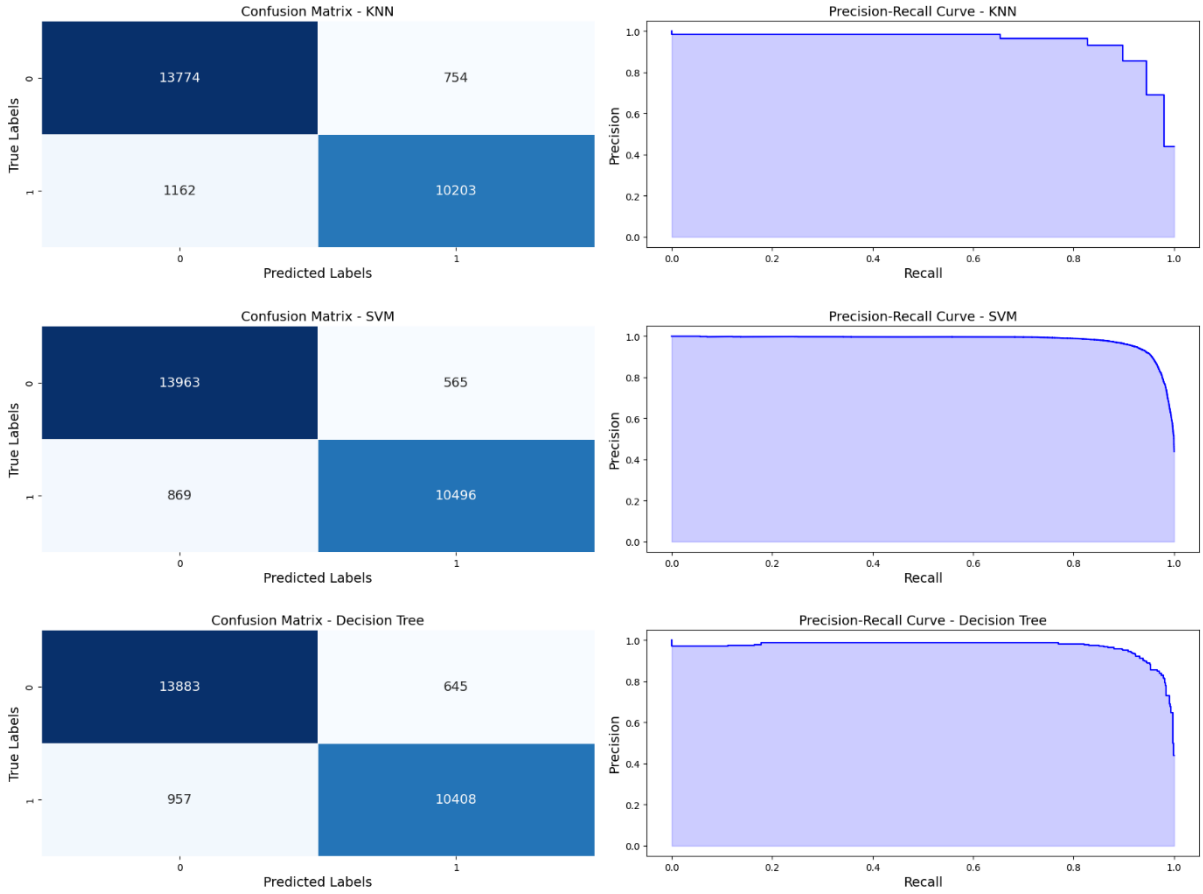

그림 11 3% 오류 동일 파라미터 결과

마찬가지로 오류가 3%일 때 그리드 서치를 진행한 결과 KNN과 SVM은 동일하였다. 반면 Tree 모델은 max_depth: 15, min_samples_split: 50 일 때 [그림 12]와 같은 결과로 accuracy가 0.939 정도로 소폭 증가하는 모습을 보였다. 이 경우에는 오버피팅이 조금 줄어들어 예측 결과가 높아졌다고 생각할 수 있다.

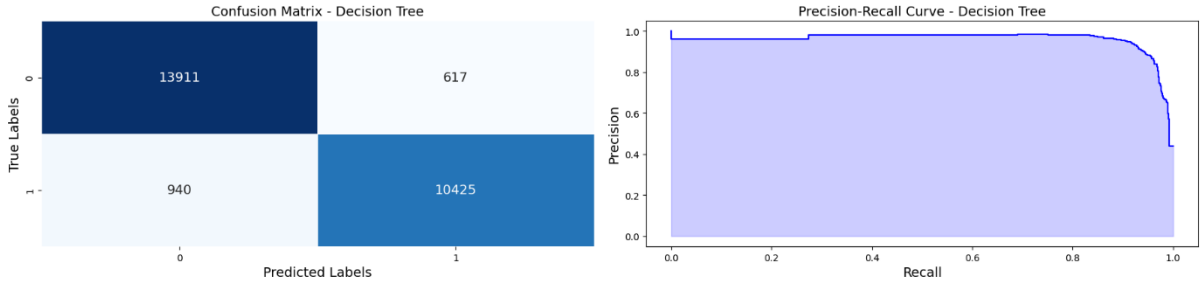

그림 12 3% 오류 Tree 모델 그리드 서치 결과

3.5 라벨링 오류 5% 적용

랜덤한 라벨링 오류를 5% 적용한 경우 동일한 파라미터를 적용한 결과는 [그림 13]과 같이 accuracy가 KNN은 0.924, SVM은 0.943, Tree는 0.936 으로 3 가지 모델 모두 소폭 감소하였다. 이 부분에서는 별 다른 특이한 점을 확인할 수 없었다.

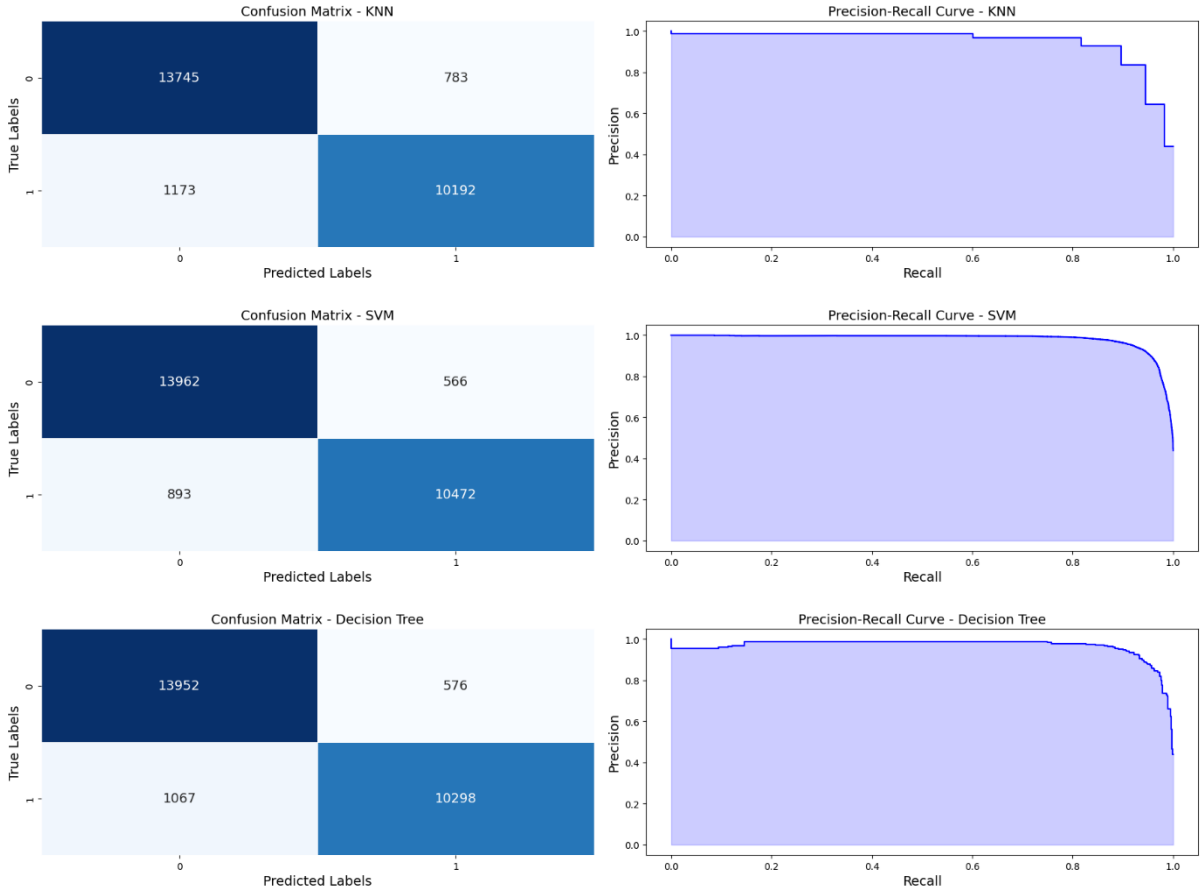

그림 13 5% 오류 동일 파라미터 결과

오류가 5%정도일 때 그리드 서치의 결과는 조금은 유의미한 결과가 나왔다. SVM은 동일한 파라미터가 나왔지만 KNN은 n_neighbors가 10 으로 증가하였고, Tree에서는 min_samples_split이 다시 100 으로 증가하였다. KNN의 결과는 [그림 14]처럼 나왔다. 오류가 늘어나다보니 주변에 관찰 값을 많이 살펴 오버피팅을 줄이는 과정으로 보인다. 다만 그 과정에서 모델이 negative로 예측하는 경향이 커졌다. Tree 모델의 결과는 위의 동일한 파라미터의 결과와 동일한데 오류가 1%, 3%일 때는 min_samples_split이 50 이었다가 100 으로 다시 증가해 오버피팅을 조금은 더 억제하는 모델이 선택되었다.

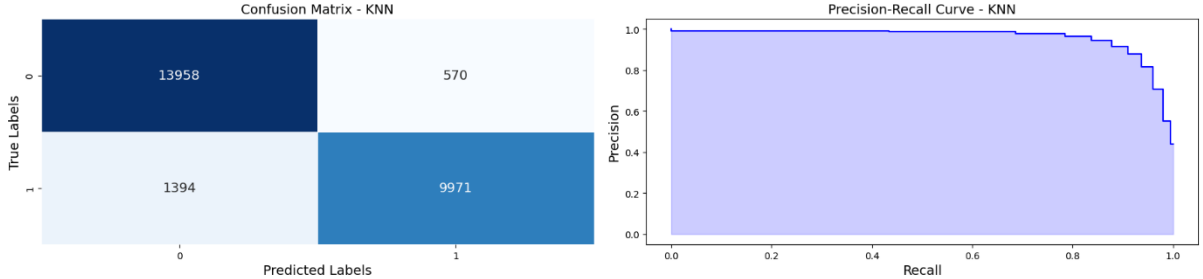

그림 14 5% 오류 KNN모델 그리드 서치 결과

3.6 라벨링 오류 10% 적용

랜덤한 라벨링 오류를 10% 적용한 경우 동일한 파라미터를 적용하였을 때 [그림 15]처럼 accuracy가 KNN은 0.913, SVM은 0.940, Tree는 0.934 로 전부 소폭 감소하였다. 하지만 예상과는 다르게 오류가 10%까지 크게 증가하여도 모델의 정확도가 급격하게 감소하는 모습을 보이지는 않았다.

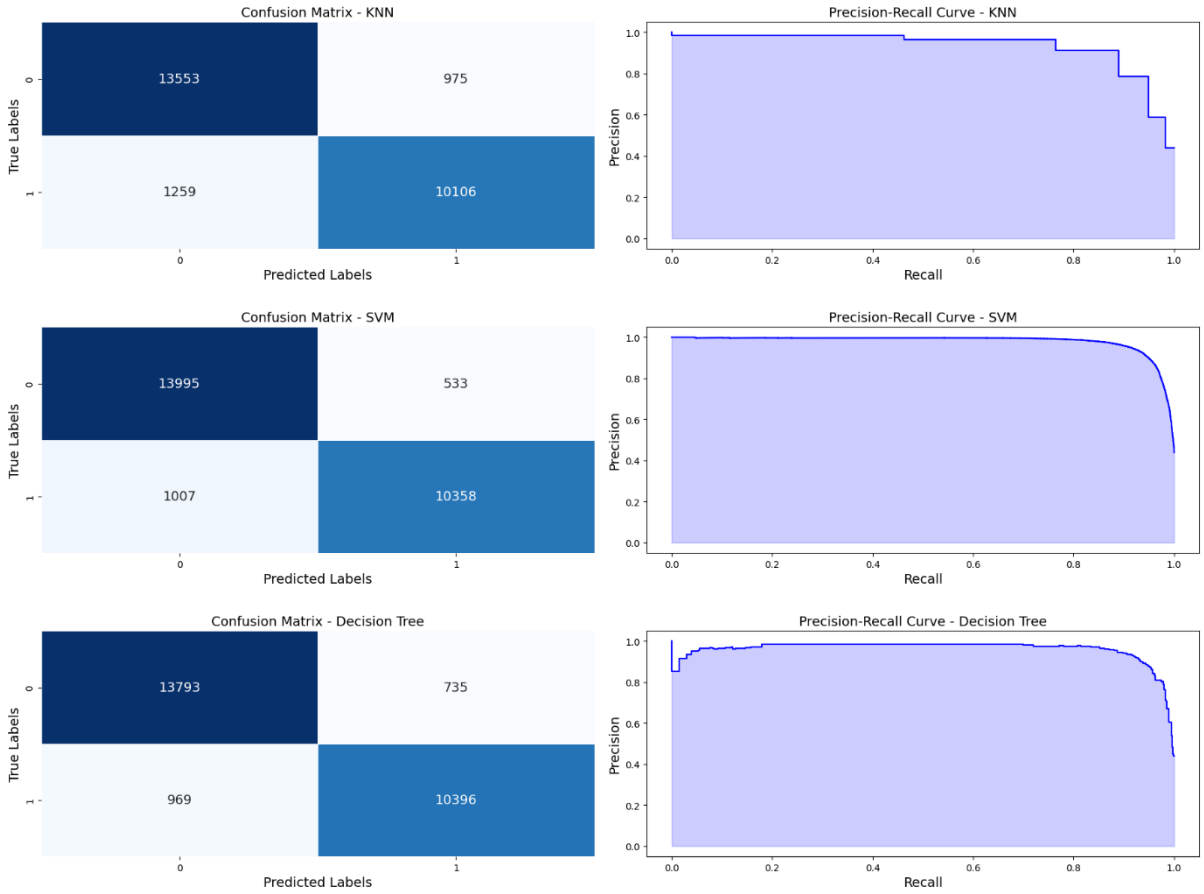

그림 15 10% 오류 동일 파라미터 결과

그리드 서치를 통한 결과도 5%와 동일하게 SVM과 Tree는 오류가 0%일 때와 동일한 파라미터이고 KNN만 K값이 10 으로 증가해 [그림 16]과 같이 accuarcy가 0.921 정도로 K가 5 인 경우보다 소폭 증가한 성과가 결과로 나왔다.

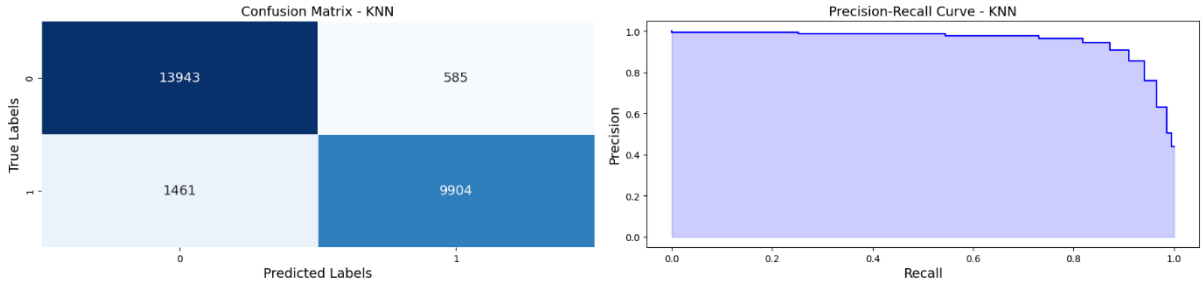

그림 16 10% 오류 KNN모델 그리드 서치 결과

4. 분석 결과

4.1 오차율에 따른 모델의 변화

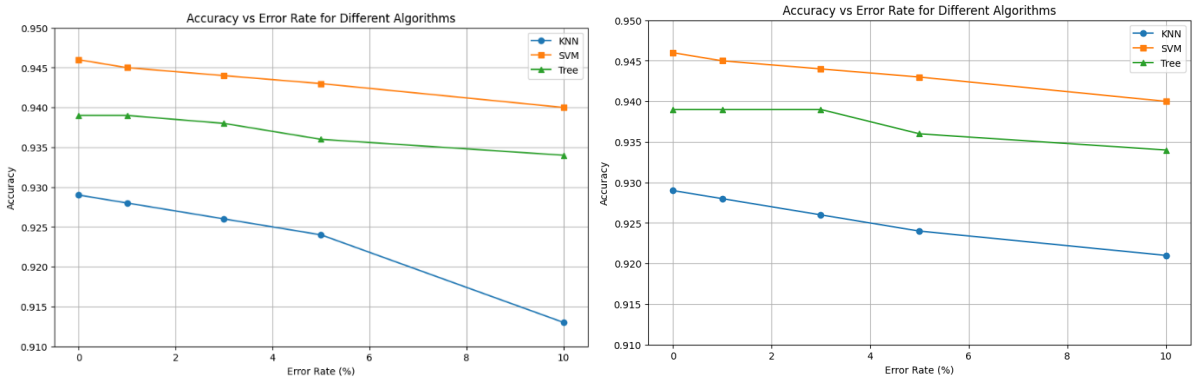

그림 17 오차율 변화에 따른 accuarcy

모든 모델이 오차율이 증가함에 따라 예측력이 감소하는 모습을 보였다. 이때 모델을 평가하는 지표로 accuracy를 선택한 것은 positive와 negative의 비율이 비슷하고 둘 중 어느 하나가 더 중요하지 않다고 생각하였기 때문에 가장 예측을 잘 맞추는 것을 보여주는 accuarcy를 평가 지표로 사용하였다.

프로젝트 시작 전 기대한 바로는 오차율이 증가할 때 오버피팅이 감소하여 모델의 성과가 좋아지지 않을까라고 생각하였던 부분은 적어도 이 데이터셋에서는 확인할 수 없었다. 그리고 쉽게 생각할 수 있는 방식으로 오차율이 증가할 때 성과가 나빠지는 어찌보면 당연한 결과를 얻을 수 있었다. 다만 오차율이 크게 증가하여도 모델의 성과가 확 나빠지는 등의 모습은 보이지 않았다.

그러나 시사할 수 있는 점은 오차율이 증가할 때 모델에 제약을 가해 오버피팅을 감소시키는 방향으로 파라미터를 조절해야 높은 오차율에서도 조금 더 높은 성과를 보여줄 수 있다. 또한 데이터에 아주 약간의 오차는 모델 훈련과 예측에 큰 영향을 주지 못하는 것을 확인할 수 있었다.

4.2 프로젝트 한계

더 많은 오차율의 변화를 주거나 그리드 서치가 아니라 그냥 모든 파라미터에 대해서 결과를 살펴보지 못한 것이 아쉽다. 또한 그리드 서치에서도 보다 많은 파라미터를 확인해보지 못한 것이 아쉽다. 그리고 데이터셋이 너무 단순해서 오차율 변화에 따른 예측 결과의 차이가 크지 않았던 것 같다. 조금 더 확실히 구분되는 binary classification을 위한 데이터 셋이나 이상치가 있는 등 조금은 보다 잘 정리되지 않은 데이터 셋을 사용해보지 못한 점도 아쉬운 점 중 하나이다.

따라서 후 더 프로젝트가 진행된다면 다른 형태의 데이터셋에서 오차율의 범위를 0%에서 최대 30%정도까지 큰 범위에 대해서 모델의 반응을 확인해보고 싶다. 또한 더 진행된다면 동일한 오차율에서도 데이터 전처리 과정에 따른 성과의 변화 또한 확인할 수 있다면 좋은 프로젝트가 될 수 있을 것 같다.

5. 결론

단순히 생각하면 데이터에서 오차가 증가하였을 때 모델의 성과가 나빠지는 것은 당연해 보일 수 있습니다. 하지만 이를 확실히 살펴보고 이때 모델들의 반응을 살펴보기 위해서 프로젝트를 진행하였습니다. 동일한 데이터 전처리 과정을 거친 후 동일한 데이터셋에서 오차율과 모델의 파라미터만 바꿔가며 모델의 성과를 살펴보았습니다.

그 결과 당연하게도 오차가 증가하였을 때 모델의 성과가 나빠지는 당연한 결과를 확인할 수 있었습니다. 그럼에도 오차율이 증가할 때 같은 모델에서도 모델에 조금 더 제약을 가해서 데이터에 둔감하게 만드는 것이 중요하다는 것을 확인할 수 있었습니다. 따라서 데이터에서 어느정도 오차가 예상될 때는 오버피팅을 줄이고 모델에 제약을 가하는 방식으로 파라미터를 조절하는 것이 더 좋은 성과를 낼 수 있다는 결론을 얻을 수 있었습니다.

하지만 이러한 결과가 이번에 사용한 데이터에서만 적용되는 결과일 수 있고 다른 원인에 의해서 이러한 결과가 나왔을 수 있다는 점은 충분히 검증하지 못했습니다. 약간은 아쉬운 결론을 낸 프로젝트이지만 추후 다른 방식으로 재검증을 통해 추가적인 내용이나 더 정확한 해석을 할 수 있을 것이라 기대합니다.

6. 참고 자료

[1] Curtis G. Northcutt, Anish Athalye, Jonas Mueller. (2021). Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. International Conference on Learning Representations Workshop Track.
[2] TJ Klein. (2019). Airline Passenger Satisfaction. Kaggle