2주차 결과보고서

 전공: 컴퓨터공학과 학년: 2학년 학번: 20221519 이름: 구현모

1. 실험시간에 작성한 코드에 전체적인 진행 흐름을 **diagram**으로 그리고, 각 단계별로 어떤 기능을 구현했는지 서술하세요.

영수증, 텍스트, 도표, 라인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.다음은 실습한 코드의 전체적인 흐름을 다이어그램으로 그린 것이다.

데이터 로드 단계에서는 전체적으로 사용되는 라이브러리인 NumPy, Pandas, Matplotlib, Seaborn 라이브러리를 불러온다. 또한 데이터가 들어있는 train.csv, test.csv 파일을 각각 train과 test에 로드 해준다.

데이터 탐색 단계에서는 각 행의 별측치 비율을 구하고 결측치를 포함하고 있는 행 찾아낸다.

그 다음 단계는 데이터 전처리 단계이다. 데이터 참색 단계에서 찾은 결측치가 있는 행들의 데이터를 처리하고 범주형 데이터를 수치형 데이터로 변환하고 불필요한 데이터를 제거해주는 것이다. Age열은 결측치를 중앙값으로 채우고 Cabin열은 삭제하며 Embarked열은 결측치를 C로 채워주었다. 범주형 변수인 'Sex'와 'Embarked'를 원-핫 인코딩으로 수치형 데이터로 변환해 준다. PassengerId, Name, Ticket처럼 생존에 영향을 미치지 못할 만한 특성들을 제거해준다.

모델 구축 및 평가 단계는 RandomForestClassifier를 생성하고 분류해둔 데이터로 학습시키고 정확도를 평가한다. 특성과 타겟(Survived)을 분류하고 훈련 데이터와 검증 데이터로 3:1로 나누어 준다. RandomForestClassifier로 모델을 생설한다. 이때 하이퍼파라미터를 설정하여 보다 성능이 좋은 모델을 만들 수 있다. 나의 경우 max\_depth를 10으로 정하였고 나머지는 기본 값과 같게 설정하여 학습하였다. 훈련 데이터와 검증 데이터를 바탕으로 정확도를 각각 구한다.

최종 예측 단계에서는 test 데이터를 전처리 해준 뒤 test 데이터를 기반으로 예측을 수행한다. 올바르게 예측이 되었다면 예측 결과를 파일에 저장하는 과정을 수행한다.

2. Random Forest Classifier의 hyper-parameter를 바꿔가면서 실험을 해보세요. 그리고 다음 3가지의 실험을 수행했을 때 결과를 설명해보세요.

2-1. max\_depth를 3에서 15까지 바꿔가면서 실험

max\_depth는 트리의 최대 깊이를 제한하는 하이퍼파라미터로, 모델의 복잡도를 조절한다. 순서대로 max\_depth를 3, 6, 9, 12, 15로 바꾸어 가며 실험한 경과이다.

텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.3) 텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.6) 텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.9) 텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.12)텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.15)

max\_depth가 증가함에 따라서 Train ACC는 증가하고 있으나 Validation ACC는 6일 때 가장 높고 점차 감소한다. max\_depth가 3일 때는 트리가 너무 낮아 모델이 단순해져서 충분히 학습할 수 없기 때문에 Train ACC와 Validation ACC 모두 낮다. max\_depth가 6일 때는 적당한 깊이의 트리를 가지기 때문에 모델이 데이터를 잘 학습할 수 있다. 모델이 데이터를 잘 학습하여 Train ACC와 Validation ACC의 균형이 잘 맞는다. max\_depth가 9, 12, 15과 같이 너무 깊은 트리를 가지게 되면 너무 많은 특성을 고려하게 되어 과적합하게 된다. 그렇게 되면 train accuracy는 높아지지만, validation과 test accuracy가 감소한다.

2-2. max\_features를 0.3에서 1.0까지 바꿔가면서 실험

max\_features는 각 노드에서 고려할 최대 특성 수의 비율을 의미하며, 트리의 다양성과 일반화 능력에 영향을 준다. 순서대로 max\_features를 0.3, 0.5, 0.7, 0.9, 1로 바꾸어 가며 실험한 경과이다.

텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 0.3) 텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 0.5) 텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 0.7)

텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 0.9) 텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다. 1)

max\_features의 변화에 따라서 Train ACC는 변화가 없다. Validation ACC의 경우 0.5일 때 가장 높고 0.9로 갈수록 작아지고 1일 때 다시 커졌다. 0.3의 낮은 max\_features를 가질 때에는 특성의 일부만 사용하므로 트리 간 다양성이 증가한다. 하지만 이 경우에는 충분히 다양한 정보를 학습하지 못하여 성능이 부족한 경우인 것 같다. max\_features이 0.5이 경우에는 특성 사용 비율이 적당하여 모델이 데이터의 주요 패턴을 학습하는데 적절하였던 것으로 보인다. max\_features이 0.7, 09와 같이 너무 큰 값을 가질 때에는 많은 특성을 고려하여 train 데이터에 대하여 과적합 되어 Validation ACC가 낮게 나온 것으로 보인다.

Train ACC는 변화가 없는 이유로는 주요 패턴을 설명하는 데 필요한 특성이 max\_features 값이 낮은 상태에서도 충분하여 모델이 train 데이터에 대해 이미 충분히 적합 상태에 도달하였기 때문일 수 있다.

데이터의 일부에 중요한 특성이 있고 나머지는 덜 중요하다면, max\_features=0.5는 중요한 특성을 포함할 확률이 높고, max\_features=1.0은 이를 확실히 포함한다. 하지만 0.7과 0.9의 경우 중요한 특성과 덜 중요한 특성을 혼합해 선택하면서 오히려 노이즈의 영향을 받게 되어 Validation ACC가 0.5와 1에서 높고 0.7, 0.9일때는 낮을 수 있다.

2-3. n\_estimators를 50에서 1000까지 바꿔가면서 실험

n\_estimators는 Random Forest를 구성하는 트리의 개수로, 모델의 안정성과 계산 비용에 영향을 준다. 순서대로 n\_estimators가 50, 200, 400, 600, 800, 1000으로 바꾸어가며 실험한 결과이다.

텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.50) 텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.200) 텍스트, 폰트, 타이포그래피, 디자인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.400)

텍스트, 폰트, 타이포그래피, 디자인이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.600) 800) 텍스트, 폰트, 타이포그래피이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.1000)

n\_estimators가 50일 때 가장 Validation ACC값이 크고 점차 감소하는 추세를 가지고 있다. 일반적으로 트리가 많을수록 예측의 분산이 줄어들고 모델이 더 안정적이 되는 경향이 있지만 실험 결과 가장 작은 50에서 Validation ACC값이 가장 크게 나왔다. n\_estimators 증가하면 트리 수가 많아지면서 모델이 train 데이터에 지나치게 적합될 가능성이 있다. n\_estimators=50에서는 트리 수가 적당히 적어 train 데이터의 노이즈를 학습하지 않고 일반화 성능이 높게 나올 수 있다. 하지만 트리가 더 많아지면 개별 트리들이 노이즈를 더 많이 학습하게 되어 validation set에서 성능이 떨어질 수 있다. 또한 데이터의 양이 적다면 트리수가 많아질수록 노이즈를 더욱 증폭시킬 수 있다. 실험의 경우에 해당하는 사항인 것 같다.

Train ACC는 변화가 없는 것으로 보인다.

3. 제출했을 때의 test accuracy가 train, validation accuracy보다 낮다면 왜 차이가 발생했는지에 대한 생각을 서술해보세요.

제출했을 때의 test accuracy는 0.77033이 나왔다. 이는 train ACC가 0.9446, Validation ACC가 0.8072인 것에 비하여 낮다. 이러한 차이가 생긴 이유는 여러가지가 있을 수 있다.

먼저 과적합 된 경우이다. 모델이 Train 데이터에 과적합되어 새로운 데이터(test set)에 대한 일반화 능력이 부족하다면 test accuracy가 낮을 수 있다. 또한 max\_features가 크다면 모델이 train 데이터의 노이즈까지 학습했을 가능성이 있다. 실습에서 나온 train ACC 가 0.9446로 높고 Validation ACC가 0.8072로 그보다 낮기 때문에 모델이 과적합되어 있음을 알 수 있다. 따라서 test accuracy가 더 낮게 나올 수 있다.

하이퍼파라미터가 적절하지 않은 경우에도 test accuracy가 낮을 수 있다. 세 하이퍼파라미터(max\_depth, max\_features, n\_estimators)가 서로 상호작용하므로 모델이 train 데이터에 치우쳐 학습했을 수 있다. 그렇게 되면 과적합 될 가능성이 높다. 모델이 과적합 되었다면 test accuracy가 더 낮게 나올 수 있다.

교차검증을 통해 모델의 안정성을 높인다면 과적합이 줄어들어 test accuracy과 train, validation accuracy의 차이가 줄어들 것이다.