|  |
| --- |
| **금융공학 인공지능 및 기계학습 중간과제** |
| **20203884 임채빈** |
| 1. 사용 데이터 전처리   - 데이터의 분류는 다음과 같이 하였음.  (ㄱ) train1 : row(0~999행)  (ㄴ) valid1 : row(1000~1700행)  (ㄷ) test1 : row(1700~2000행)  (ㄹ) train2 : row(2000~2700행)  (ㅁ) valid2 : row(2700~3000행)  (ㅂ) test2 : row(3000~3300행)  (ㅅ) tvdata : row(1000~1700행) : 처음 사용해보고 아님을 깨닫고 그냥 진행함. |
| 1. 사용한 Supervised Learning Algorithm   - Logistic Regression  - Decision Tree  - AdaBoost |
| 1. 구체적 Process 2. 처음 시행착오   - 처음 Decision Tree를 이용하는 과정에서 train1, test1으로 hyper parameter tuning을 하고, 거기서 정한 hyper parameter로 train1, train2가 학습 + 각각 test1, test2를 분류하려고 했으나, 계속해서 드는 한 가지 의문이 있었음.    - 위 사진은 train1, valid1을 통해서 결정한 hyperparameter를 이용하여 데이터row(2000~3000)를 학습하고, test2를 분류한 Logistic Regression의 Confusion matrix임. 정확도가 57%로 나쁘지 않음. 하지만 여기서 한 가지 의문이 생겼음. “주가 데이터는 시계열 데이터이기 때문에, row(1000~1700)에서 정한 hyperparameter가 test2에서 더 좋은 영향을 끼칠 수 있을까? 하는 것이었음. **만약, 다른 성격의 시장으로 시장 자체가 변한다면, Hyper parameter도 비교적 최신의 데이터를 활용해 튜닝되어야 하는 것이 아닌가?** 라는 생각이 들었음.  - 따라서 이후의 과정에서는 row(2000~3000) 데이터도 train2(2000~2700rows), valid2(2700~3000rows)로 나누어서 테스트를 시도해 보았음. 그 결과가 아래와 같이 나왔음.    - 비록 정확도는 0.01만큼 아주 조금 상승했지만, 0 부분의 precision과 recall이 심하게 저조하게 나왔음. 따라서, 비교적 많은 데이터(700rows)를 활용한 valid1을 이용해서 튜닝한 파라미터를 사용하기로 결정하였음.   1. Logistic Regression 2. Hyper parameter 튜닝   - Logistic Regression에서의 hyperparameter 튜닝 과정은 “어떤 threshold를 사용해서 분류할 것인가?”이다.  - Threshold를 결정하는 과정에서, (0.01 ~ 0.99)중에서 모델의 F1-Score를 가장 높게 하는 Threshold를 결정하도록 분석을 진행하였음.  - 결과 : threshold는 0.16의 값을 가졌음.    - 위 그림은 각 Threshold별 F1-Score의 분포임.   1. 분석 결과   - 위 Threshold(0.16)을 이용하여 test1 (by train1), test2 (by train2)를 예측한 결과가 아래와 같음. 각각 왼쪽, 오른쪽 Confusion matrix를 참고하면 됨.    - test1의 경우 분류 성능이 49%, test2의 경우 57%의 분류 성공률이 나왔음.   1. Decision tree 2. Hyper Parameter 튜닝   - Decision Tree에서의 hyperparameter 튜닝 과정은 “어떤 max\_depth를 가진 트리를 사용할 것인가?”이다.  - max\_depth를 결정하는 과정에서, 1~10의 depth를 고려하였고, F1-Score를 가장 높게 하는 depth를 결정하도록 분석을 진행하였음.  - 결과 : optimal max depth는 1이었음.     1. 분석 결과   - 위 Threshold(max\_depth = 1)을 이용하여 test1 (by train1), test2 (by train2)를 예측한 결과가 아래와 같음. 각각 왼쪽, 오른쪽 Confusion matrix를 참고하면 됨.    - test1의 경우, 분류 성능이 50%, test2의 경우 60%의 분류 성공률이 나왔음.   1. Adaboost 2. Hyper parameter 튜닝   - Adaboost에서의 hyperparameter 튜닝은 “몇 개의 estimator로 앙상블할 것인가?”이다. (decision tree의 max depth는 1로 잡고 진행 하였음)  - # of estimators를 결정하는 과정에서, 1~150의 estimator 갯수를 고려하였고, F1-Score를 가장 높게 하는 #를 결정하도록 분석을 진행하였음.  - 결과 : Optimal # of estimators는 1이었음.    앙상블을 진행할수록, 오버피팅이 가속화되는 느낌일 뿐, 크게 F1-Score가 개선되지 않았음.   1. 분석 결과   - # of estimator가 1이 나왔기 때문에, Decision Tree와 같은 결과가 나왔음. |
| 1. 가장 좋은 알고리즘 선택   - 주식을 살까 말까 하는 의사결정을 하는 과정에서 가장 중요한 것은 “정확도”라는 생각이 들었음. 그 이유는 내려갈 것이라 예상되면 팔아야 하고, 올라갈 것이라 예상되면 사야 되기 때문에, 다른 특성들 중에서 더 중요한 것을 찾지 못했기 때문임.  - 따라서 정확도가 높은 알고리즘을 사용해야겠다는 생각이 들었음. 따라서 Decision Tree를 사용하여 주식 가격의 상승과 하락을 예측(분류)하는 모델을 사용하는 것이 바람직하다는 결론을 내렸음.  - 성능이 0.6으로 그리 높진 않은데, 이를 극복하기 위해서는 시계열적 고려가 필요할 것으로 보임. 가령, 차분이나 Detrending과 같은 작업도 필요할 것으로 보임. 하지만, 이는 아직까지 풀리지 않은 금융인들의 숙제라고 볼 수 있음. 주가가 Geometric Brownian Motion을 따른다는 이론은 실무와 맞지 않는 경우가 많기 때문임. 시장은 계속해서 바뀌고, 한정된 데이터에서 OutPerform한 모델이 다른 데이터를 바탕으로 했을 때에도 그러한지는 논란의 여지가 있음. 따라서 시계열적 고려가 필요할 것으로 보임. |