AI 레포트_장홍서

분야 Ciber Security

상태 완료

학습일 2025/04/04

↗️ 자세한 코드 (Github)

https://github.com/Moomin03/WHS-AI-for-Malicious-URL-Analysis

↗️ 실습 4강까지 실행 후 성능 측정 결과 캡처 (매트릭스)

```
[[1304   66]
 [  72  407]]
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      1370
           1       0.86      0.85      0.86       479

    accuracy                           0.93      1849
   macro avg       0.90      0.90      0.90      1849
weighted avg       0.93      0.93      0.93      1849

기존 RF 모델 :  0.9572742022714981
개정된(?) RF 모델 :  0.9583558680367766
```

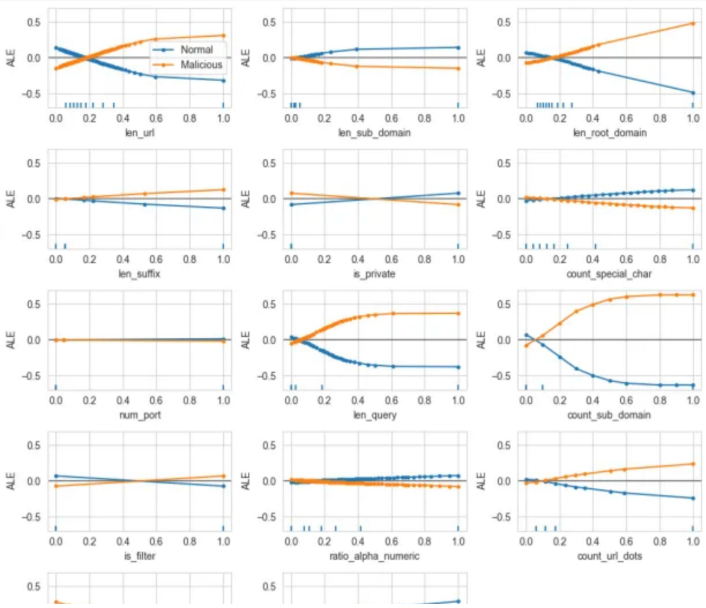
➡️ 정확도를 높일 수 있는 방법 1가지 구상해보기(파라미터 수정, 특징 및 데이터 추가, 다른 모델 사용 등등)

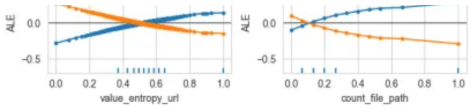
↗️ 어떻게 모델의 정확도를 높일 수 있을까?

✅ 정확도를 높일 수 있는 방법에 대해서 생각하다, 최근에 많이 사용도는 HardVoting이 떠올랐고, 3개 정도의 모델을 훈련시키기로 했습니다. 3개로 정한 이유는 짝수로 설정했을때는 타깃이 2:2로 나올 수 있기 때문입니다. (모델은 Historical Gradient Boosting Classifier, Gradient Boosting Classifier, RandomForest Classifier로 진행하겠습니다.)

⭐ `Hardvoting` 을 진행하려면, 데이터셋이 `경량화` 되어야합니다. 데이터 특성이 많으면 많을수록 과적합을 이끌어내기도 하고, 연산량이 증가하기 때문입니다. 그래서 저는 XAI를 활용하여 `모델의 Importance` 를 직접 판단하도록 하겠습니다.

↗️ alibi를 활용한 타깃데이터에 대한 변수 변화율 측정하기



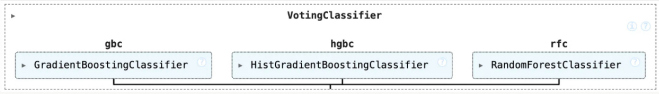


✅ 위의 그래프는 `모델을 학습시키는데 사용된 특성들이 각 클래스를 결정하는데 어떤 영향을 미치는지 알아본 그래프` 들입니다. 여기서 각 클래스는 총 두가지로 `악성인가 정상` 인가에 대한 부분이고, `ALE` (y축)의 값이 크면 클수록 해당 변수에 영향을 많이 받는다고 생각하시면 됩니다.

🌟 그래프를 보시면 아시겠지만, `촘촘하게 이어져있는 그래프` 는 좋은 그래프이고, `두 그래프의 차이가 많이 나는 그래프` 가 좋은 그래프입니다. 촘촘하게 이어져있는 그래프가 좋은 그래프인 이유는 `is_fillter` 같은 특성의 경우 0.0에 데이터 한개, 1.0에 데이터 한개로 `이분화` 되어있는데, 이런 그래프의 경우 해당 특성이 모델을 훈련하는데 `유의미하게 변화했다고 보기 어렵습니다` . 그리고 `num_port` 같은 변수의 경우 두 그래프의 사이 간격이 매우 좁아서 특성의 변화에 따라서 타깃값의 유의미한 변화를 보여준다고 보기에는 어렵습니다. 이런 기준으로 `num_port, is_filter, is_private` 는 제외하도록 하겠습니다.

✌️ 이후에 해당 데이터셋에 대한 훈련을 진행하였고 `optuna` 라이브러리를 통해 `best_params` 를 작성했고, `best_params` 로 각 모델을 훈련시킨 다음, 하드보팅한 결과는 아래와 같습니다.

```
last_gbc = GradientBoostingClassifier(**best_params_gbc)
last_hgbc = HistGradientBoostingClassifier(**best_params_hgbc)
last_rfc = RandomForestClassifier(**best_params_rfc)
hard_vote = VotingClassifier(voting='hard', estimators=[('gbc', last_gbc),
                                                        ('hgbc', last_hgbc),
                                                        ('rfc', last_rfc)])
hard_vote.fit(train_input, train_target)
```




```
print(f'하드 보팅 결과(훈련)는 : {hard_vote.score(train_input, train_target)}')
print(f'하드 보팅 결과(테스트)는 : {hard_vote.score(test_input, test_target)}')
```

```
하드 보팅 결과(훈련)는 : 0.9994591671173607
하드 보팅 결과(테스트)는 : 0.9626825310978907
```

➡️ Panda 전략을 이용한 데이터 증강 (심화학습)

↗️ 데이터가 불균형을 이루므로 SMOTE 기법을 적용하도록 하겠습니다!

```
# SMOTE 기법 적용
smote = SMOTE(sampling_strategy='auto', random_state=42)
train_aug, target_aug = smote.fit_resample(train_input, train_target)

# 모델 학습
panda_model = VotingClassifier(voting='hard', estimators=[('gbc', last_gbc),
                                                          ('hgbc', last_hgbc
                                                          ('rfc', last_rfc)]
panda_model.fit(train_aug, target_aug)

# 성능 평가
print(f"PANDA-SMOTE Train Accuracy: {panda_model.score(train_input, train_ta
print(f"PANDA-SMOTE Test Accuracy: {panda_model.score(test_input, test_targe
```

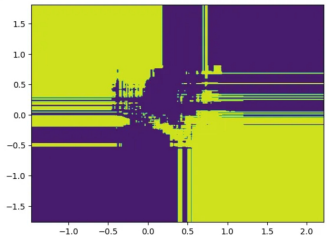
```
PANDA-SMOTE Train Accuracy: 0.9859
PANDA-SMOTE Test Accuracy: 0.9611
```

↗️ 점수 변동 도식화

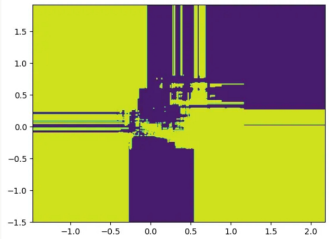
❤️ **기존 모델에 비해서 과적합에서 조금 벗어나 일반화된 모습을 보여주었다고 생각합니다.**

모델	HardVoting	HardVoting Panda
훈련세트 점수	99.94	96.26
테스트세트 점수	98.59	96.11

↗️ 결정 경계 그래프화



일반 HardVoting 모델로 양쪽 색 모두 비슷



Pandas HardVoting 모델로 노란색에 대해서 관대함