2022학년도 2학기

인공지능과 정보보안

Malware Detection Exercise

32184045 산업보안학과 전호영

목차

1. AdaBoost

```
[27] from sklearn.ensemble import AdaBoostClassifier

     AdaBoost = AdaBoostClassifier(n_estimators = 32, learning_rate = 1)
     AdaBoost.fit(x_train, y_train)

     y_pred = AdaBoost.predict(x_test)
```

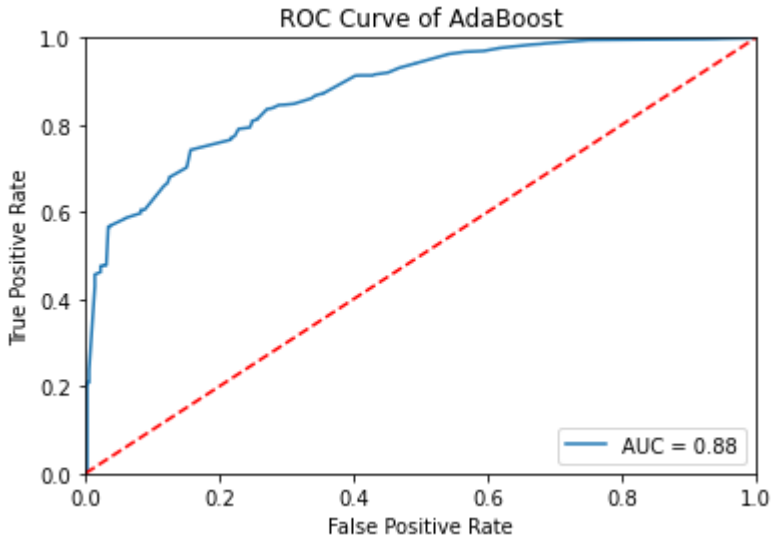
- n_estimators

생성할 weak learner의 갯수를 의미한다.

```
[ ]  accuracy = accuracy_score(y_pred, y_test) # 전체 중 모델이 바르게 분류한 비율!
     print('Accuracy: {}'.format(accuracy))
     print('F1 score: {}'.format(f1_score(y_test, y_pred, zero_division=1))) # f1_score == Prediction과 Recall의 조화평균! Data가 imbalanced 할 때!
     print(classification_report(y_test, y_pred, target_names=['normal', 'malware']))
```

```
Accuracy: 0.9601570620163992
F1 score: 0.9796183611980859
              precision    recall  f1-score   support

      normal       0.56      0.07      0.12       350
     malware       0.96      1.00      0.98      8309

    accuracy                           0.96      8659
   macro avg       0.76      0.53      0.55      8659
weighted avg       0.95      0.96      0.94      8659
```

```
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
import matplotlib.pyplot as plt

y_scores = AdaBoost.predict_proba(x_test) # class별 확률 값을 반환!
fpr, tpr, threshold = roc_curve(y_test, y_scores[:, 1])
roc_auc = auc(fpr, tpr) # auc = area under curve

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of AdaBoost')
plt.show()
```



본 보고서에서 모델의 성능비교는 AUC의 너비를 통해 하도록 하겠습니다.

ROC Curve란 TPR, FPR을 축으로 해서 분류모델의 성능을 비교하는 커브입니다.

ROC Curve에서 모델의 성능이 좋다는 것은 AUC(Area Under Curve)의 넓이가 넓다는 것을 의미합니다.

위 AdaBoost의 경우 n_estimator를 20~50 사이로 변경하며 AUC를 비교해본 결과 n_estimators = 32 인 경우 AUC가 0.88로 가장 높은 성능을 보였습니다.

2. DecisionTree

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=10, random_state = 50)
tree.fit(x_train, y_train)

y_pred = tree.predict(x_test)
```

```
print("Traning set accuracy: {:.3f}".format(tree.score(x_train, y_train)))
print("Test set accuracy: {:.3f}".format(tree.score(x_test, y_test)))
```

```
Traning set accuracy: 0.967
Test set accuracy: 0.966
```

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=10, random_state = 50)
tree.fit(x_train, y_train)

y_pred = tree.predict(x_test)
```

```
print("Traning set accuracy: {:.3f}".format(tree.score(x_train, y_train)))
print("Test set accuracy: {:.3f}".format(tree.score(x_test, y_test)))
```

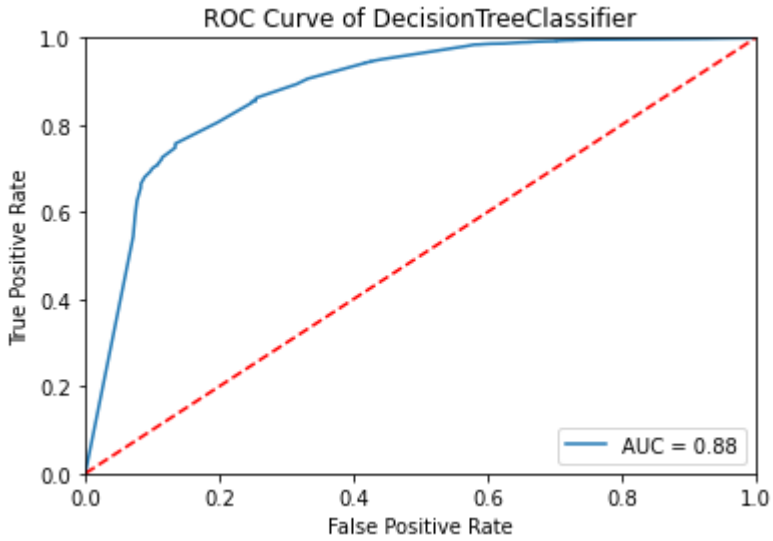
```
Traning set accuracy: 0.967
Test set accuracy: 0.966
```

```
from sklearn.metrics import confusion_matrix
confmat = confusion_matrix(y_true = y_test, y_pred = y_pred)
print(confmat)
```

```
[[  93  257]
 [  41 8268]]
```

```
accuracy = accuracy_score(y_pred, y_test)
print('Accuracy: {}'.format(accuracy))
print('F1 score: {}'.format(f1_score(y_test, y_pred, zero_division=1)))
print(classification_report(y_test, y_pred, target_names=['normal', 'malware']))
```

```
Accuracy: 0.96558494052431
F1 score: 0.9822977307829392
              precision    recall  f1-score   support

      normal       0.69      0.27      0.38       350
     malware       0.97      1.00      0.98      8309

    accuracy                           0.97      8659
   macro avg       0.83      0.63      0.68      8659
weighted avg       0.96      0.97      0.96      8659
```



DecisionTree의 AUC는 0.88을 나타냈습니다.

3. RandomForest

```
[34] from sklearn.ensemble import RandomForestClassifier
```

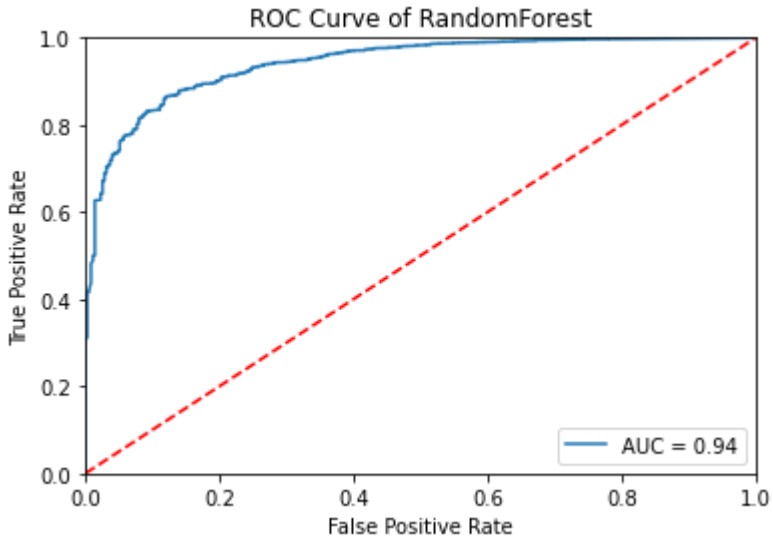
```
[59] from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth=10, random_state=50) # max_depth가 5~11까지 올라갔을 때 계속 정확도가 상승함! AUC도 커짐!
rf.fit(x_train, y_train) # rf 알고리즘으로 x_train dataset과 y_train dataset을 fitting 하겠다!

y_pred = rf.predict(x_test) # x_test값을 통해 모델 값을 예측! 즉 성능평가
```

```
# confusion matrix -> 오차행렬
from sklearn.metrics import confusion_matrix
confmat = confusion_matrix(y_true = y_test, y_pred = y_pred)
print(confmat)
```

```
[[ 147  203]
 [  66 8243]]
```

```
[57] accuracy = accuracy_score(y_pred, y_test) # 전체 중 모델이 바르게 분류한 비율!
     print('Accuracy: {}'.format(accuracy))
     print('F1 score: {}'.format(f1_score(y_test, y_pred, zero_division=1))) # f1_score == Prediction과 Recall의 조화평균! Data가 imbalanced 할 때!
     print(classification_report(y_test, y_pred, target_names=['normal', 'malware']))

     Accuracy: 0.9689340570504678
     F1 score: 0.9839450910176066
                   precision    recall  f1-score   support

           normal       0.69      0.42      0.52       350
          malware       0.98      0.99      0.98      8309

         accuracy                           0.97      8659
        macro avg       0.83      0.71      0.75      8659
     weighted avg       0.96      0.97      0.97      8659
```

```
from sklearn.metrics import roc_curve
from sklearn.metrics import auc
import matplotlib.pyplot as plt

y_scores = rf.predict_proba(x_test) # class별 확률 값을 반환!
fpr, tpr, threshold = roc_curve(y_test, y_scores[:, 1])
roc_auc = auc(fpr, tpr) # auc = area under curve

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0,1])
plt.ylim([0,1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of RandomForest')
plt.show()
```

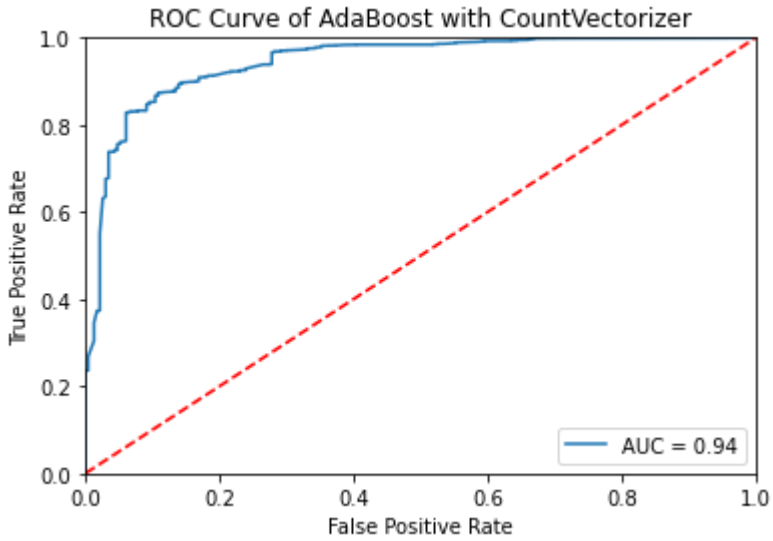


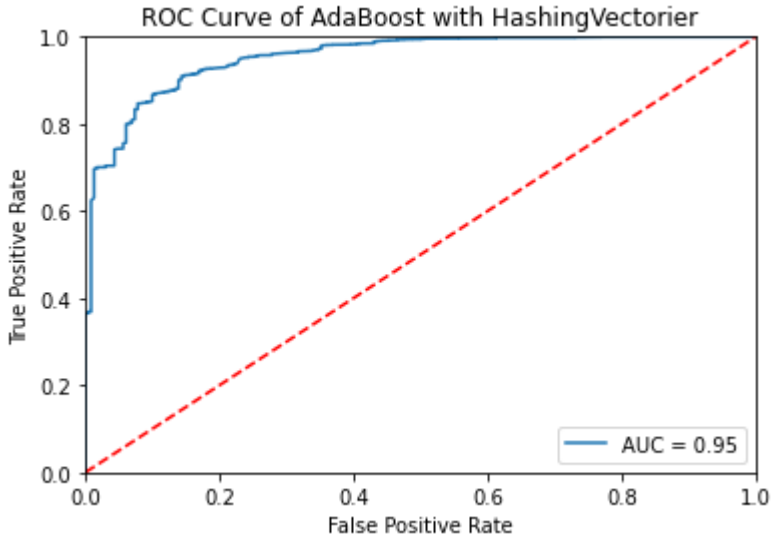
RandomForestClassifier 를 사용했을때 AUC는 0.94로 가장 높은 AUC를 나타냈습니다.

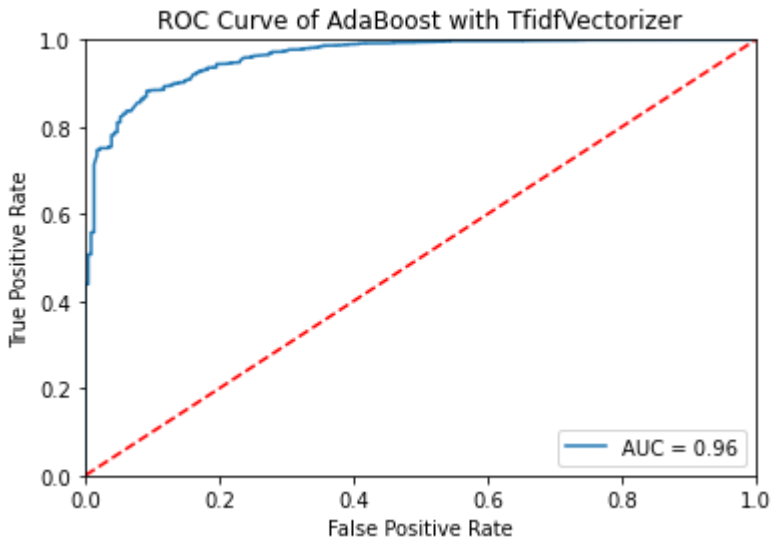
4. Algorithms with Vectorizer

이번 목차에선 코드를 제외하고 ROC Curve만 첨부하겠습니다.

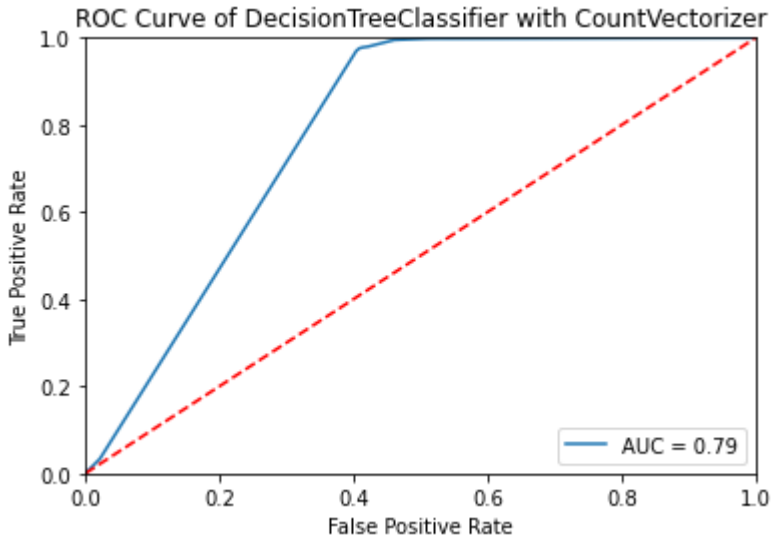
- AdaBoost with Vectorizers.

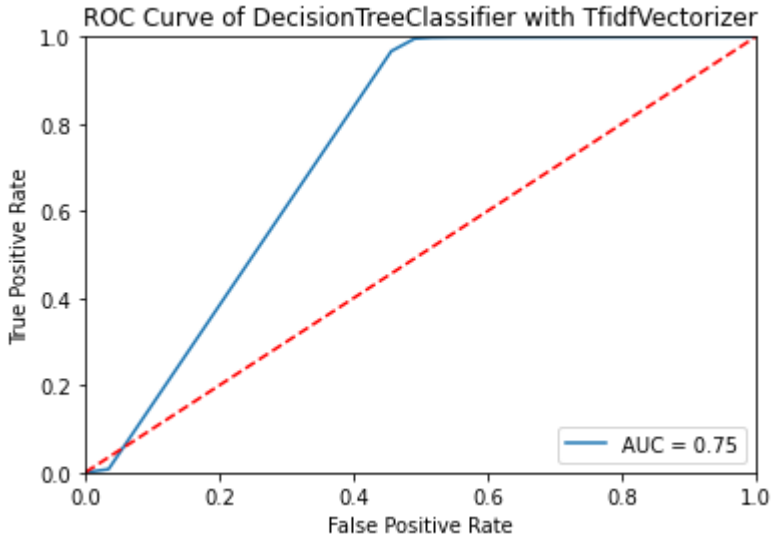


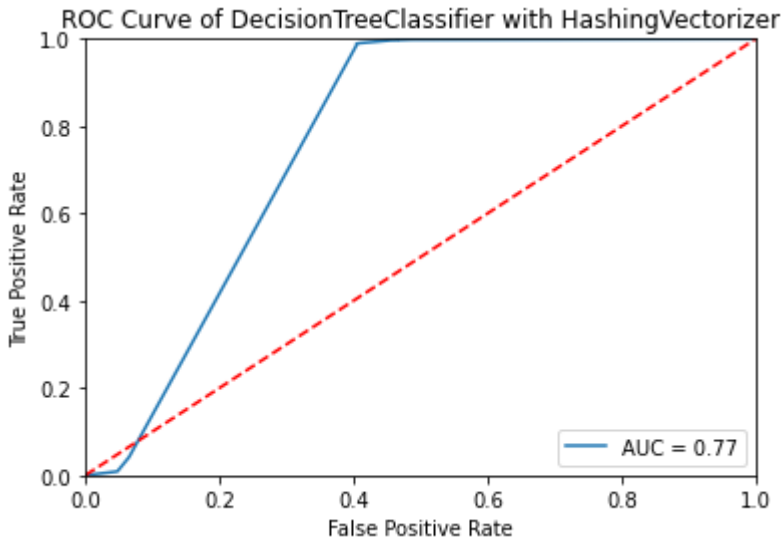




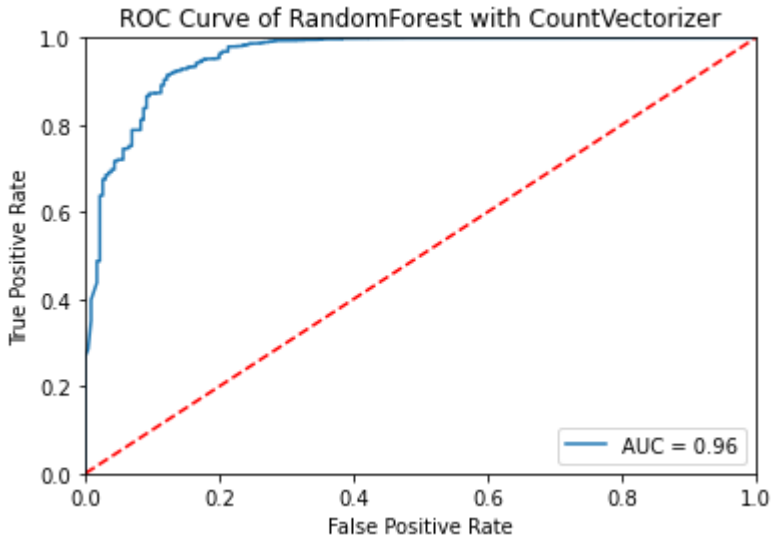
- DecisionTree with Vectorizers.

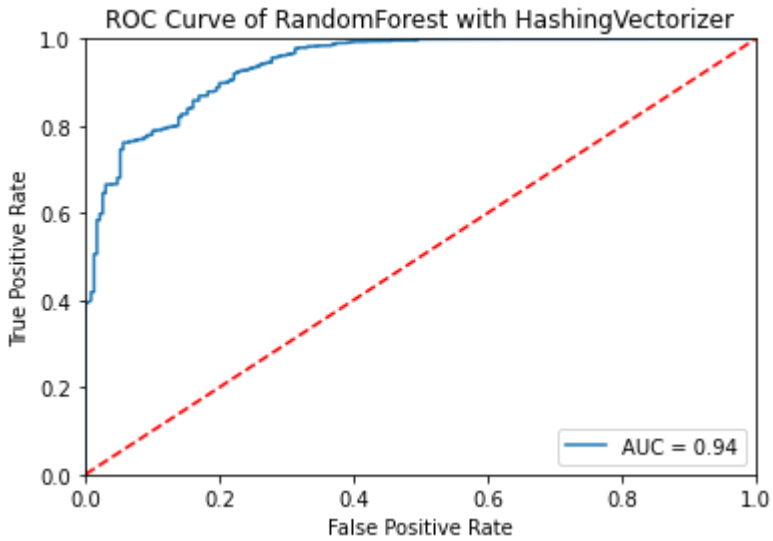


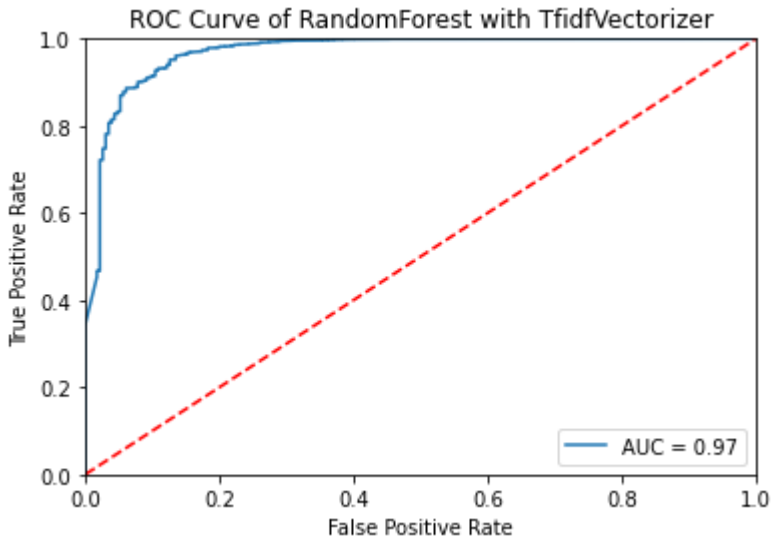




- RandomForest with Vectorizers.


ROC Curve of RandomForest with CountVectorizer
True Positive Rate
False Positive Rate
AUC = 0.96
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve of RandomForest with HashingVectorizer
True Positive Rate
False Positive Rate
AUC = 0.94
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve of RandomForest with TfidfVectorizer
True Positive Rate
False Positive Rate
AUC = 0.97
0.0
0.2
0.4
0.6
0.8
1.0

5. 성능 비교

상기 모델의 성능비교를 위해 AUC값을 표로 정리하겠습니다.

	x	CountVectorizer	HashingVectorizer	TfidfVectorizer
AdaBoost	0.86	0.94	0.95	0.96
DecisionTree	0.88	0.79	0.75	0.77
RandomForest	0.94	0.96	0.94	0.97

평가기준을 AUC로 잡았을 경우,

Vectorization을 적용하지 않은 경우, RandomForest가 가장 높은 성능을 보였습니다.

Vectorization을 적용했을 경우, TfidfVectorizer를 사용한 RandomForest가 AUC 값 0.97로 가장 높은 AUC값을 나타내며, 가장 좋은 성능을 보여줍니다.