DADS6003-APPLIED MACHINE LEARNING
Project Firewall database

นายวศิน ถาวรวัฒนะ 6510422013
นายภัทรกร ผิวชอุ่ม 6510422024
นางสาววิชุนันท์ คำภิโร 6510422027

April 30, 2023

1 Get data

source from https:https://www.kaggle.com/datasets/tunguz/internet-firewall-data-set

Fatih Ertam (2018) ได้วิเคราะห์ข้อมูลการบันทึกบนอุปกรณ์ไฟร์วอลล์และควบคุมการรับส่งข้อมูลทางอินเทอร์เน็ตตามผลการวิเคราะห์ที่มาจากการบันทึกการใช้อุปกรณ์ไฟร์วอลล์ของ Firat University ซึ่งจำแนกข้อมูลออกเป็น 4 คลาส allow deny drop และ reset-both โดยใช้วิธี SVM ได้แก่ Linear, Polynomial, Sigmoid และ RBF วัดประสิทธิภาพของแบบจำลองโดยใช้ Precesion recall และ F1-score เราสามารถระบุได้ว่าข้อมูลที่สร้างขึ้นจากการจัดหมวดหมู่มีความเกี่ยวข้องกับข้อมูลที่ตั้งใจไว้มากน้อยเพียงใด ในการศึกษานี้ ใช้ 11 ลักษณะ ในการวิเคราะห์ข้อมูลเหตุการณ์จำนวน 65,532 ครั้ง ตามตารางด้านล่างนี้

TABLE I. FEATURES AND DESCRIPTION

Feature	Description
Source Port	Client Source Port
Destination Port	Client Destination Port
NAT Source Port	Network Address Translation Source Port
NAT Destination Port	Network Address Translation Destination Port
Elapsed Time (sec)	Elapsed Time for flow
Bytes	Total Bytes
Bytes Sent	Bytes Sent
Bytes Received	Bytes Received
Packets	Total Packets
pkts_sent	Packets Sent
pkts_received	Packets Received
Action	Class (allow, deny, drop, reset-both)

Figure 1: Features and Description.

SVM+Sigmoid ให้ค่า recall สูงสุด 98.5% ส่วน SVM+Linear ให้ค่า precision สูงสุดที่ 67.5% และ SVM+RBF ให้ค่าคะแนน F1 สูงสุดที่ 76.4% ในขณะที่ SVM+Polynomial ให้ค่า precision และ recall อยู่ในระดับต่ำ เมื่อใช้ค่าเฉลี่ย จะเห็นได้ชัดว่าตัวแยกประเภทตามฟังก์ชันการเปิดใช้งาน SVM+RBF นั้นดีที่สุดสำหรับแต่ละคลาส ตามตารางด้านล่าง

TABLE III. EVALUATION RESULTS

Method	F_1 Score	Precision	Recall
SVM Linear	75.4	67.5	85.3
SVM Polynomial	53.6	61.8	47.4
SVM RBF	76.4	63.0	97.1
SVM Sigmoid	74.8	60.3	98.5

Figure 2: Evaluation Result.

ข้อจำกัดในงานวิจัยนี้คือ SVM ใช้เพื่อจัดการปัญหาที่เกี่ยวข้องกับสองคลาสขึ้นไป ไม่เหมาะกับการใช้กับข้อมูลที่มีความสูงต่ำกว่า data point จึงต้องแปลงข้อมูลให้อยู่ใน input space ไปสู่ transformed Space ที่เรียกว่า Feature space จึง

สามารถใช้ตัวแบบ SVM ได้ในการแบ่ง ข้อมูลด้วย Hyperplane เช่น Polynomial Kernel, Radial Basis Function(RBF), Sigmoid Kernel นอกจากนี้ในงานวิจัยยังใช้แค่ SVM ในการทดลองทั้งที่ยังมี model classification อื่นๆอีกหลายแบบ

2 ขั้นตอน Collect data, Inspection data, Data Exploration (EDA)

2.1 จำนวน Action ที่เกิดขึ้นในtransaction

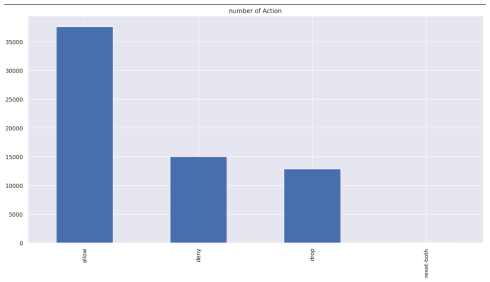

Figure 3: จำนวน Action ที่เกิดขึ้นในtransaction

2.2 การหาค่า mean ของ Elapsed time (sec)

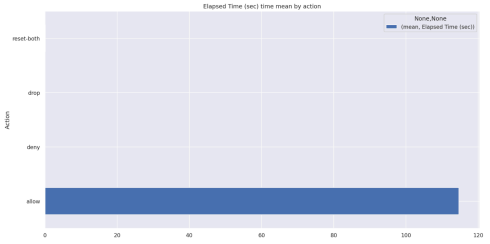

Figure 4: ค่า mean ของ Elapsed time (sec)

2.3 การหาค่า mean ของ Bytes

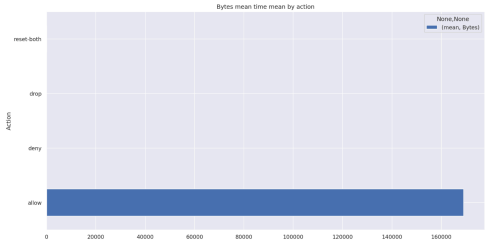

Figure 5: การหาค่า mean ของ Bytes

2.4 การหา correlation ของ Bytes และ time

แบบไม่ได้ remove bytes ที่เป็น outliner ออก, แบบปรับค่า น้อยกว่าเท่ากับ 1,000, แบบปรับค่า น้อยกว่าเท่ากับ 10,000 และ แบบปรับค่า น้อยกว่าเท่ากับ 100,000

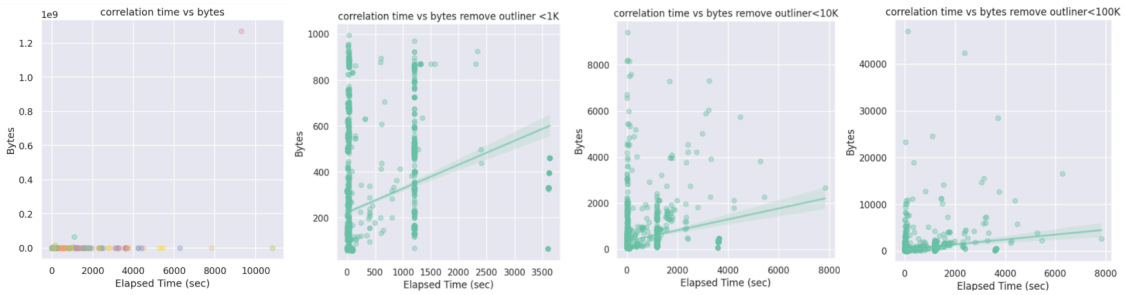

Figure 6: การหา correlation ของ Bytes และ time

2.5 การหา Top Destination port และ Top Source port ที่มีการเกิด transaction มากที่สุด

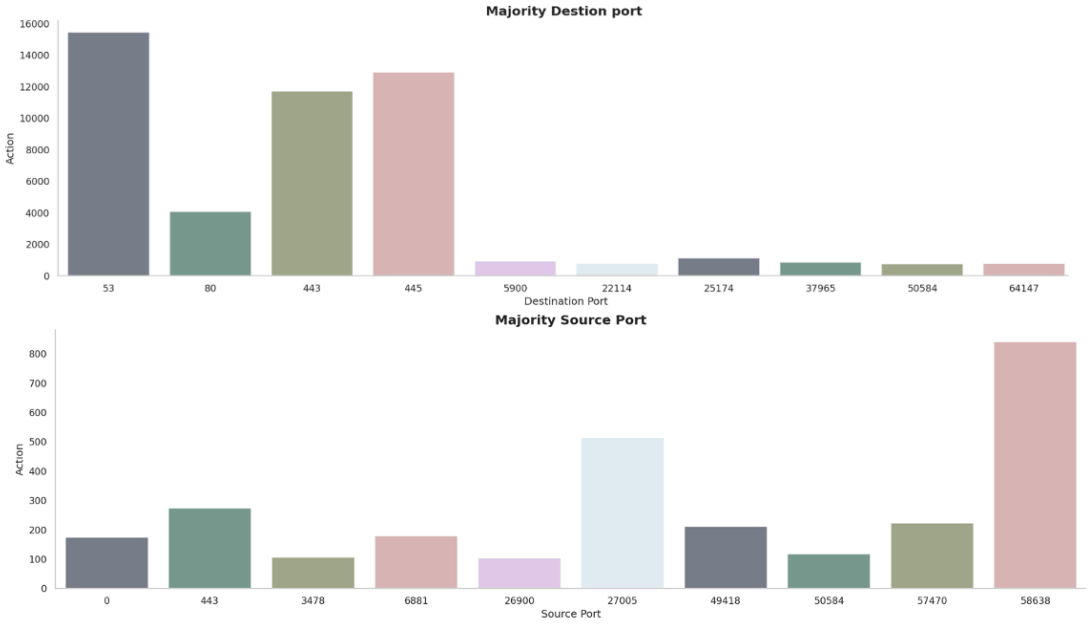

Figure 7: Top Destination port และ Top Source port ที่มีการเกิด transaction มากที่สุด

2.6 การหา Correlation heatmap เพื่อดูความสัมพันธ์ของFeature ทุกตัว

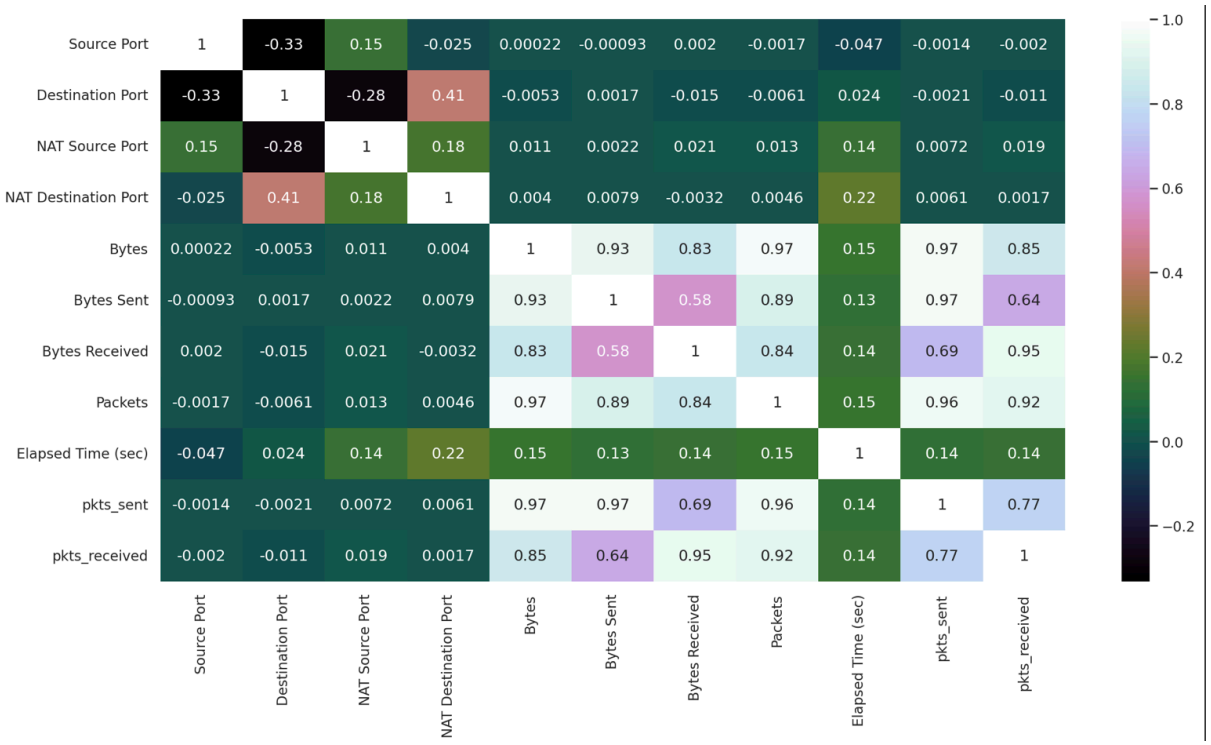

Figure 8: Correlation heatmap เพื่อดูความสัมพันธ์ของFeature ทุกตัว

3 Training and Testing Model

3.1 Cart Method

จากการrun model Cart จากข้อมูลtest 30% ได้F1 score = 0.9977 20% ได้F1 score = 0.9975 10% ได้ F1 score = 0.9971

```
#GridSearCV
        tree_clas = DecisionTreeClassifier(random_state=1024)
        param_tree = {'max_features': ['auto', 'sqrt', 'log2'],
                      'ccp_alpha': [0.1, .01, .001],
                      'max_depth' : [3, 4, 5, 6, 7, 8, 9, 10],
                      'criterion' :['gini', 'entropy']}
        grid_tree = GridSearchCV(estimator=tree_clas,
                param_grid=param_tree, cv=5, verbose=True)
        grid_tree.fit(x_train, y_train)
        grid_tree.best_estimator_
        ans_tree = grid_tree.predict(x_test)
        print(classification_report(y_test, ans_tree))จากการทำ
```

GridSearchCV ได้ค่าพารามิเตอร์คือ `DecisionTreeClassifier(ccp_alpha=0.001, criterion='entropy', max_depth=9, max_features='auto', random_state=1024)`

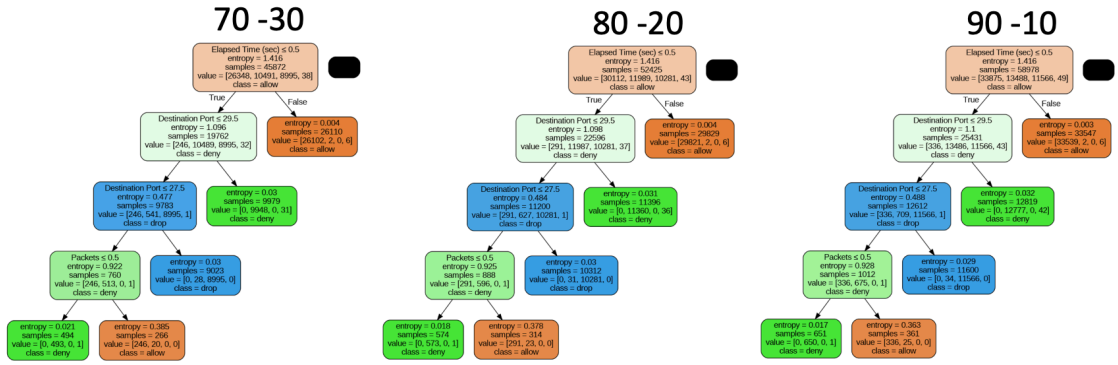

Figure 9: Cart Method

3.2 Random Forest

```
#GridSearCV
        RT_clas = RandomForestClassifier(n_estimators=100,random_state=1024)
        param_RT = {'max_features': ['auto', 'sqrt', 'log2'],
                            'ccp_alpha': [0.1, .01, .001],
                            'max_depth' : [2,3, 4, 5, 6, 7],
                            'criterion' :['gini', 'entropy']}
        grid_RT = GridSearchCV(estimator=RT_clas, param_grid=param_RT, cv=5, verbose=True)
        grid_RT.fit(x_train, y_train)
        grid_RT.best_estimator_
        ans_RT = grid_RT.predict(x_test)
        print(classification_report(y_test, ans_RT))จากการทำ
```

GridSearchCV ได้ค่าพารามิเตอร์คือ RandomForestClassifier(ccp_alpha=0.001, criterion='entropy', max_depth=7, max_features='auto', random_state=1024)

70 -30

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11292
1	1.00	1.00	1.00	4496
2	1.00	1.00	1.00	3856
3	0.00	0.00	0.00	16
accuracy			1.00	19660
macro avg	0.75	0.75	0.75	19660
weighted avg	1.00	1.00	1.00	19660

80 -20

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7528
1	1.00	0.99	1.00	2998
2	1.00	1.00	1.00	2570
3	0.00	0.00	0.00	11
accuracy			1.00	13107
macro avg	0.75	0.75	0.75	13107
weighted avg	1.00	1.00	1.00	13107

90 -10

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3765
1	1.00	0.99	1.00	1499
2	1.00	1.00	1.00	1285
3	0.00	0.00	0.00	5
accuracy			1.00	6554
macro avg	0.75	0.75	0.75	6554
weighted avg	1.00	1.00	1.00	6554

Figure 10: Random Forest

3.3 XGBoost

```
        xgb_cls = xgb.XGBClassifier(use_label_encoder=False, n_estimators=10)
        xgb_cls.fit(x_train, y_train.astype(int))

        ans_xgb = xgb_cls.predict(x_test)
        print(classification_report(y_test, ans_xgb))
```

```
plot_tree(xgb_cls, rankdir='UT', num_trees=1)
fig = plt.gcf()
fig.set_size_inches(15, 10)

plt.show()
plt.savefig('log2-xgboot.png')
```

70 -30

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11292
1	1.00	1.00	1.00	4496
2	1.00	1.00	1.00	3856
3	1.00	0.12	0.22	16
accuracy			1.00	19660
macro avg	1.00	0.78	0.80	19660
weighted avg	1.00	1.00	1.00	19660

80 -20

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7528
1	1.00	1.00	1.00	2998
2	1.00	1.00	1.00	2570
3	1.00	0.27	0.43	11
accuracy			1.00	13107
macro avg	1.00	0.82	0.86	13107
weighted avg	1.00	1.00	1.00	13107

90 -10

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3765
1	1.00	0.99	0.99	1499
2	0.99	1.00	1.00	1285
3	1.00	0.40	0.57	5
accuracy			1.00	6554
macro avg	1.00	0.85	0.89	6554
weighted avg	1.00	1.00	1.00	6554

Figure 11: XGBoost

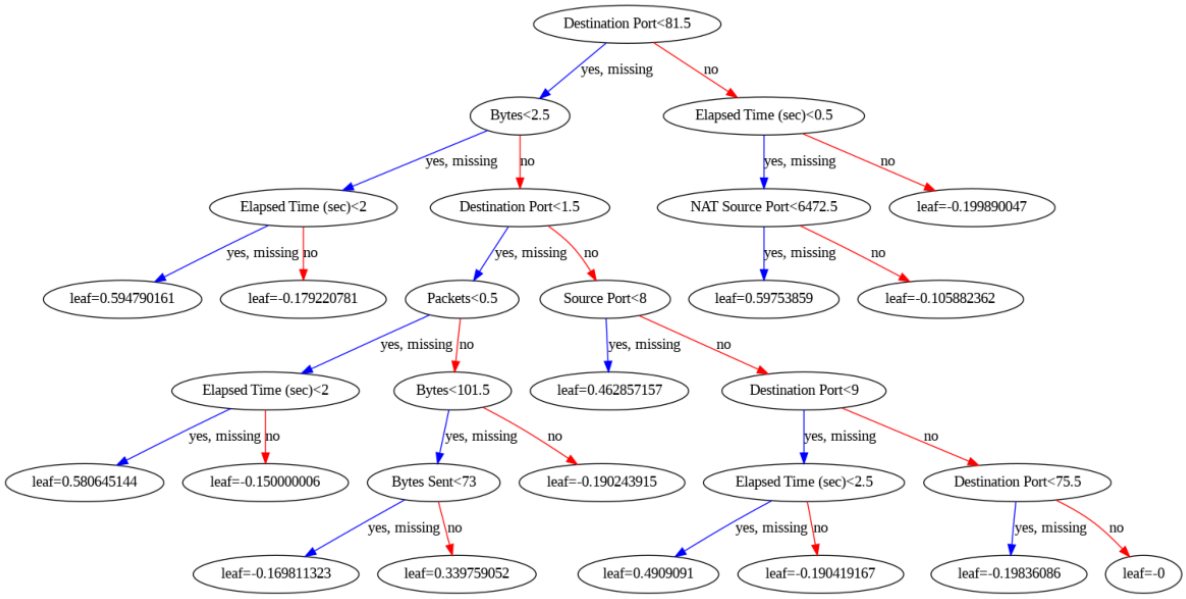

Figure 12: XGBoost_tree

3.4 K-NN

จากการrun model KNN ค่าF1จากการ testข้อมูล 30% ให้F1 score = 0.9936 ในขณะที่ test 20% ได้F1 score 0.9933 และ F1 score 0.9924 Test 10% นั้นสะท้อนในให้เห็นว่ายิ่งมีdataในการtest มากขึ้นยิ่งเพิ่มประสิทธิภาพของmodel

```
#GridSearCV
        knn = KNeighborsClassifier()
        parameters = {"n_neighbors": range(1,11),
                      "metric": ('minkowski','euclidean','manhattan'),
                      "weights": ('uniform','distance')}
        grid_knn = GridSearchCV(knn, parameters, cv=5)
        grid_knn.fit(x_train,y_train)
        grid_knn.best_estimator_จากการทำ
```

GridSearchCV ได้ค่าพารามิเตอร์คือ KNeighborsClassifier(metric='manhattan', n_neighbors=1)

70 -30

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11292
1	0.99	0.99	0.99	4496
2	0.98	1.00	0.99	3856
3	0.00	0.00	0.00	16
accuracy			0.99	19660
macro avg	0.74	0.74	0.74	19660
weighted avg	0.99	0.99	0.99	19660

80 -20

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7528
1	0.99	0.99	0.99	2998
2	0.99	1.00	0.99	2570
3	0.33	0.09	0.14	11
accuracy			0.99	13107
macro avg	0.83	0.77	0.78	13107
weighted avg	0.99	0.99	0.99	13107

90 -10

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3765
1	0.99	0.99	0.99	1499
2	0.99	1.00	0.99	1285
3	0.00	0.00	0.00	5
accuracy			0.99	6554
macro avg	0.74	0.75	0.74	6554
weighted avg	0.99	0.99	0.99	6554

Figure 13: K-NN

4 Evaluation)

4.1 ผล precision recall และ F1 Score

สรุปผล: จากการทดลอง จะเห็นได้อย่างชัดเจนว่าของ F1 score ของ xgboost มากที่สุด = 99.79% ในขณะที่ Model อื่น average อยู่ที่ 87.99% นอกจากนี้ยังสามารถเพิ่ม F1 score ของSVM ได้มากกว่าในreport ที่กล่าวมาข้างต้นจากเดิม 76.4% โดยปรับimblance data และ Feature scaling (normalization) ใน pycarte F1 score = 93.34%

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.9977	0.9997	0.9977	0.9981	0.9979	0.9960	0.9960	3.595
lightgbm	Light Gradient Boosting Machine	0.9973	0.9997	0.9973	0.9982	0.9977	0.9953	0.9953	0.984
et	Extra Trees Classifier	0.9971	0.9995	0.9971	0.9976	0.9973	0.9950	0.9950	1.139
rf	Random Forest Classifier	0.9968	0.9997	0.9968	0.9982	0.9974	0.9944	0.9944	1.304
dt	Decision Tree Classifier	0.9966	0.9982	0.9966	0.9975	0.9970	0.9941	0.9941	0.550
nb	Naive Bayes	0.9912	0.9990	0.9912	0.9970	0.9940	0.9849	0.9850	0.529
gbc	Gradient Boosting Classifier	0.9899	0.9997	0.9899	0.9981	0.9936	0.9826	0.9827	8.451
knn	K Neighbors Classifier	0.9888	0.9973	0.9888	0.9972	0.9928	0.9808	0.9810	0.699
lr	Logistic Regression	0.9164	0.9977	0.9164	0.9888	0.9449	0.8605	0.8688	1.591
svm	SVM - Linear Kernel	0.9026	0.0000	0.9026	0.9876	0.9334	0.8381	0.8494	0.584
ada	Ada Boost Classifier	0.8660	0.9991	0.8660	0.9706	0.8713	0.7866	0.8178	1.088
ridge	Ridge Classifier	0.7676	0.0000	0.7676	0.9130	0.8052	0.6406	0.6708	0.538
lda	Linear Discriminant Analysis	0.7592	0.9879	0.7592	0.9104	0.7983	0.6301	0.6623	0.567
qda	Quadratic Discriminant Analysis	0.5766	0.8569	0.5766	0.8024	0.5769	0.3672	0.4611	0.553
dummy	Dummy Classifier	0.5744	0.5000	0.5744	0.3299	0.4191	0.0000	0.0000	0.545

Figure 14: ผล precision recall และ F1 Score

4.2 ผล Confusion Matrix

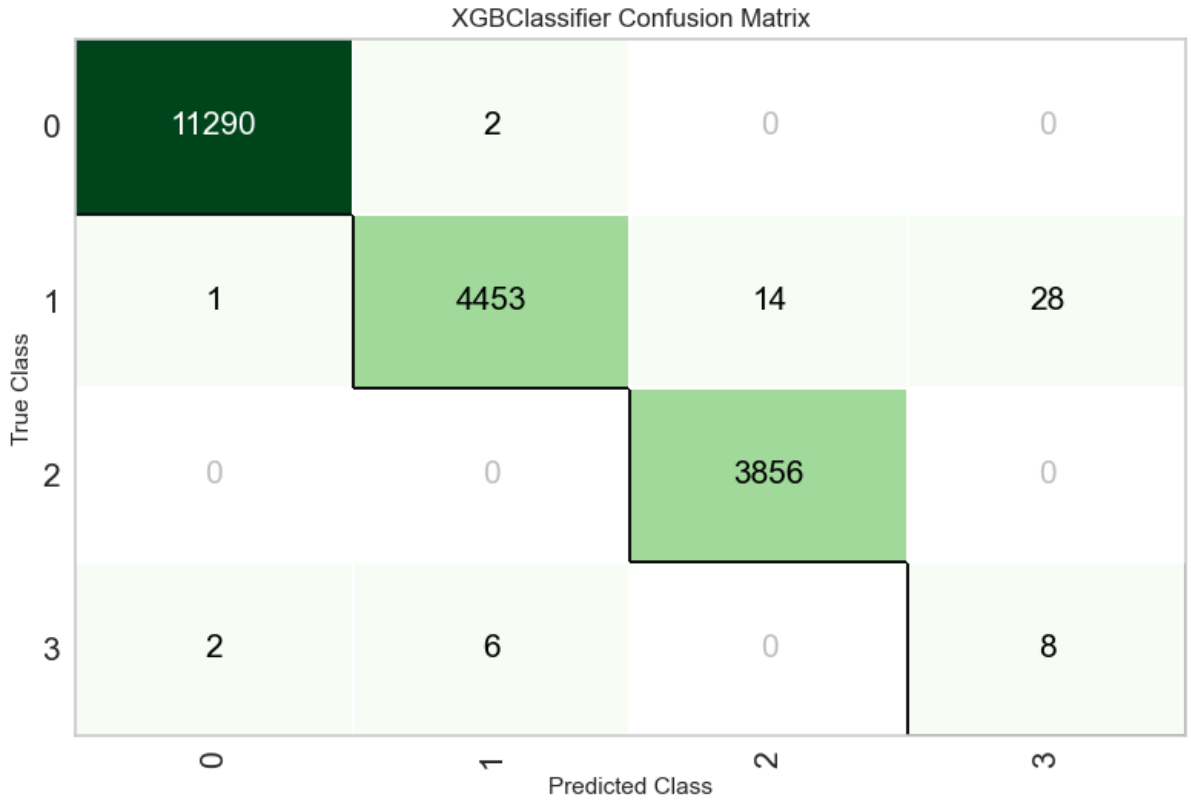

Figure 15: ผล Confusion Matrix

4.3 ผล ROC Curve

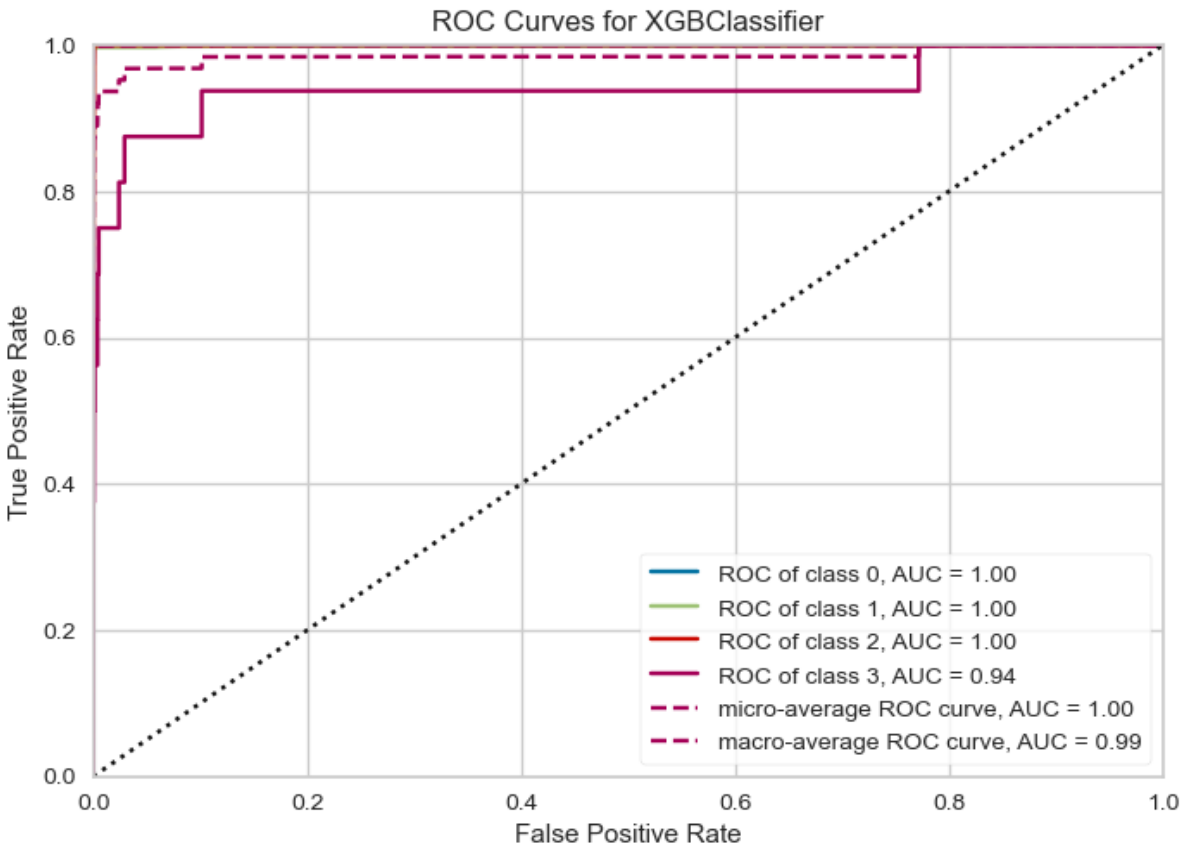

Figure 16: ผล ROC Curve