Classification on Firewall: ทำนาย Action ของ Firewall จากพฤติกรรมที่เกิดขึ้น โดย Action จะประกอบไปด้วย 1. Allow 2. Deny 3. Drop 4. Reset-both

โดยในงานนี้จะพิจารณาจากพฤติกรรมที่เกิดขึ้นเท่านั้น จะไม่นับรวม Source Port หรือ เลข Destination Port มาเกี่ยวข้อง

Dataset มีดังนี้:

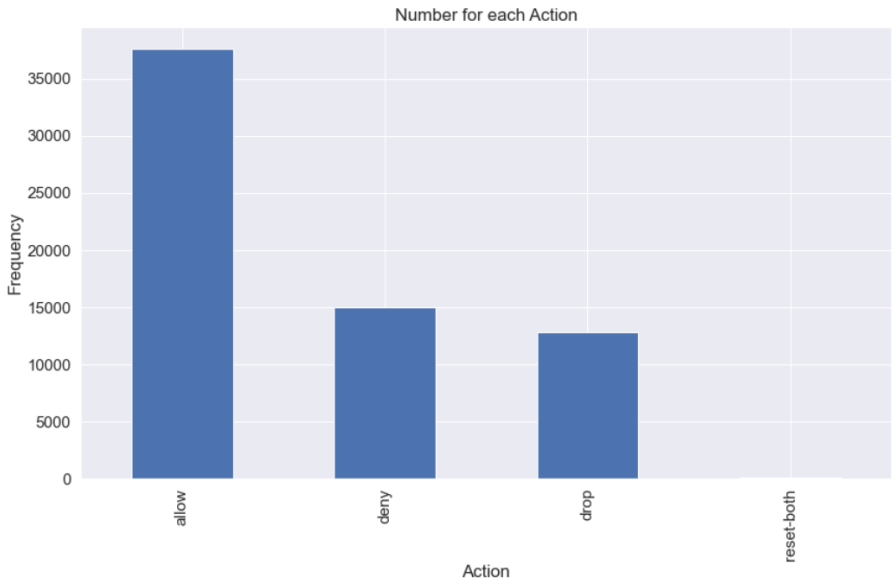
1. Source Port: Client Source Port
2. Destination Port: Client Destination Port
3. NAT Source Port: Network Address Translation Source Port
4. NAT Destination Port: Network Address Translation Destination Port
5. Elapsed Time (sec): Elapsed Time for flow
6. Bytes: Total Bytes
7. Bytes Sent: Bytes Sent
8. Bytes Received: Bytes Received
9. Packets: Total Packets
10. pkts_sent: Packets Sent
11. pkts_received: Packets Received
12. Action Class: (allow, deny, drop, reset-both)

นาย คณิศร หงษ์ทอง 6420422024

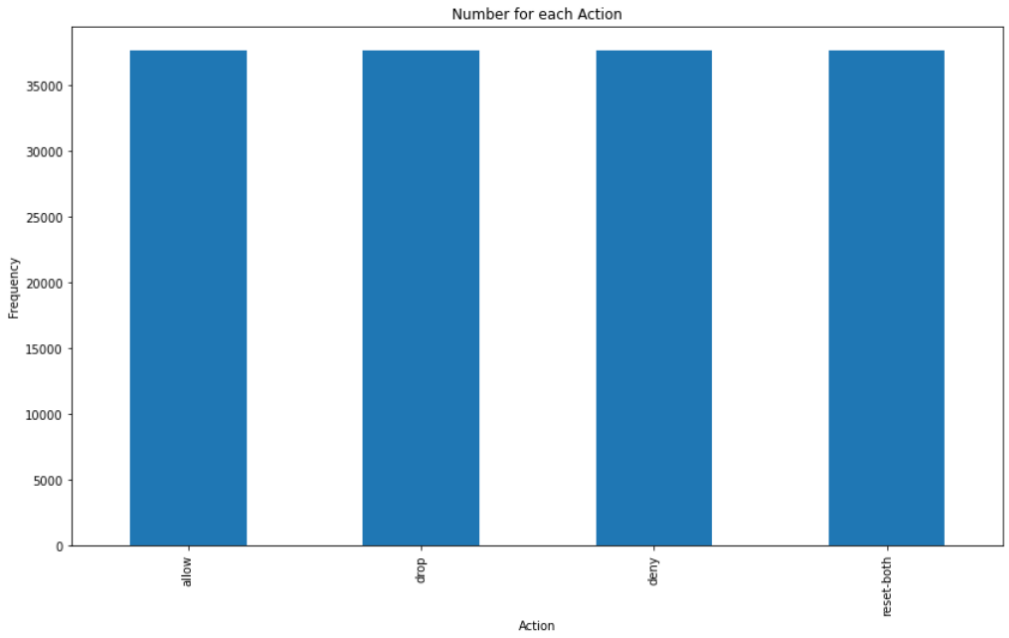
EDA & Data Cleansing

1. Imbalance data

ข้อมูลแต่ละ Action ที่ได้มาจะมีปริมาณที่ไม่เท่ากัน ทำให้เกิดข้อผิดพลาดจากการ Predict ได้ จึงต้องทำการแก้ไขด้วยการทำ Oversampling



(ลักษณะข้อมูลก่อนทำ Oversampling)



(ลักษณะข้อมูลหลังทำ Oversampling)

2. Source Port VS NAT Source Port

พิจารณาเบื้องต้นว่าทั้ง Source Port มีความเหมือนกับ NAT Source port อย่างไร และความเหมือนกับความแตกต่างจะส่งผลต่อ Action หรือไม่

พิจารณา Source Port

Check-SP-NSP-Equal	False	True
label		
allow	34477.0	3163.0
deny	37374.0	266.0
drop	37640.0	NaN
reset-both	29710.0	7930.0

- จากการตรวจสอบ Source Port จะเห็นได้ว่า ไม่ว่า Source Port จะเหมือนกับ NAT Source Port หรือไม่ ความแตกต่างระหว่าง Label ทั้ง 4 ก็ยังไม่ชัดเจน เพราะฉะนั้นเราจะไม่เลือกตัวแปรนี้เข้าสู่ Model

3. NAT Source/Destination Port

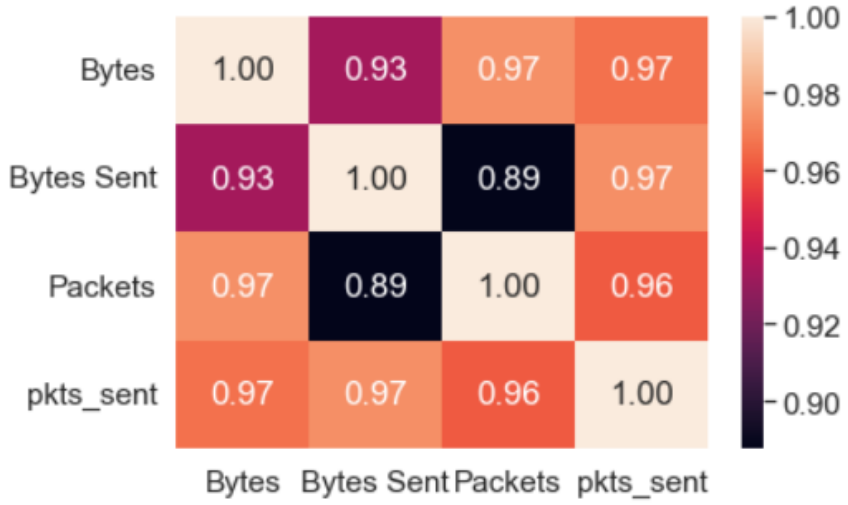
พิจารณา NAT Destination Port

is-NDP-0	False	True
label		
allow	37076.0	564.0
deny	34.0	37606.0
drop	NaN	37640.0
reset-both	9052.0	28588.0

- จากการตรวจสอบพบว่า หาก NAT Destination Port มีค่าเป็น 0 จะให้ผลต่างระหว่าง Label ทั้ง 4 ที่ชัดเจน โดยเฉพาะอย่างยิ่งคือ Label “Allow” เพราะฉะนั้นเราจะเลือกตัวแปรนี้เข้าสู่ Model

4. Bytes, Bytes Sent, Packets, pkts_sent

ตัวแปรทั้ง 4 ตัวนี้ จากการตรวจสอบพบว่ามีความใกล้เคียงกัน โดยเราจะใช้ค่า Correlation มาใช้วัดผลความสัมพันธ์กันของข้อมูล ได้ตารางดังนี้

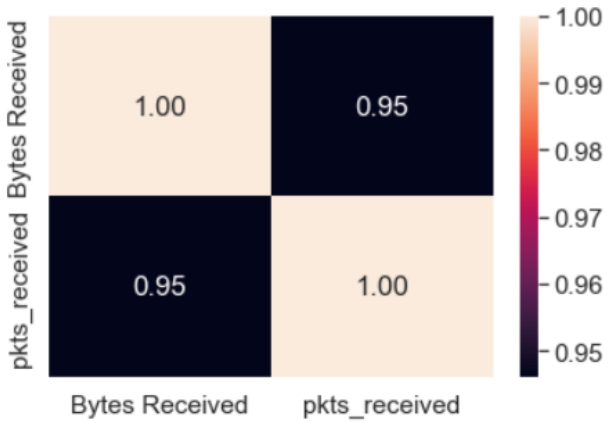


- เราจะพบว่าข้อมูลทั้ง 4 ชุด มี Correlation เป็น Strong Positive ทั้งหมด ทั้งนี้ถึงแม้ว่าเทคนิคที่ใช้อย่าง XGBoost จะสามารถรับมือกับชุดข้อมูลที่มี Correlation สูงได้ เราสามารถนำข้อมูลทั้ง 4 ชุดเข้าไปได้ แต่เราจะเลือกเพียง Column เดียวไปใช้งานเท่านั้น โดยเราจะเลือก Bytes Sent ไปใช้งาน

label	Bytes	Bytes Sent	Packets	pkts_sent
allow	169037.953241	38917.411743	178.348565	71.334883
deny	82.070510	81.846015	1.003587	1.002311
drop	68.711211	68.711211	1.000000	1.000000
reset-both	157.944421	142.150691	1.662380	1.436610

5. Bytes Receive & pkts_Receive

เช่นเดียวกับกลุ่ม Bytes Sent ตัวแปร Bytes Receive และ pkts_Receive ก็มี correlation ที่สูงเช่นกัน ทำให้เราจะเลือกข้อมูลเพียงชุดเดียวเท่านั้นไปใช้ใน Model ในทีนี้จะเลือก Bytes Receive ไปใช้



label	Bytes Received	pkts_received
allow	130120.541498	107.013682
deny	0.224416	0.001249
drop	0.000000	0.000000
reset-both	15.778773	0.210680

6. สรุปตัวแปรที่ถูกเอาเข้าไปใน Model มีดังต่อไปนี้

- Bytes Sent
- Bytes Received
- Elapsed Time
- CheckNAT (NAT Destination port = 0 หรือไม่)

Model & Evaluation

1. แบ่ง Train/Test Split ด้วยสัดส่วน 70%/30% ตามลำดับ

```
# split data into train and test sets
seed = 7
test_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X_over, y_over, test_size=test_size, random_state=seed)
```

2. สร้าง Model ด้วย XGBClassifier

```
model2 = XGBClassifier()
model2.fit(X_train_x_2, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
```

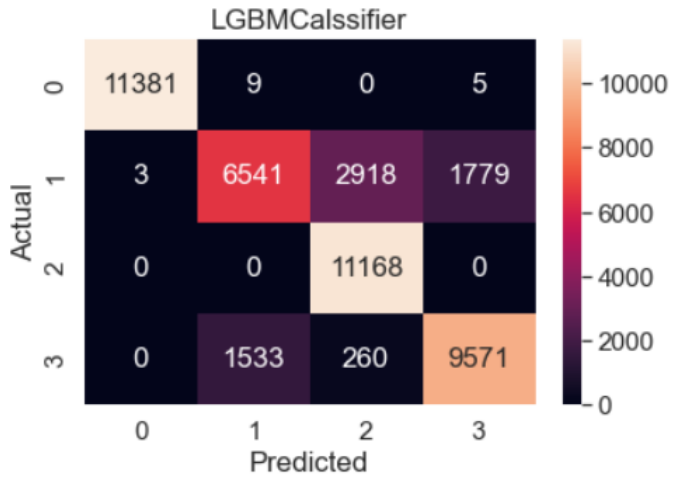
3. ทดสอบผลลัพธ์ด้วย Test set

- **Accuracy**

```
# evaluate predictions
accuracy_2 = accuracy_score(y_test, predictions_2)
print("Accuracy: %.2f%%" % (accuracy_2 * 100.0))
```

```
Accuracy: 85.59%
```

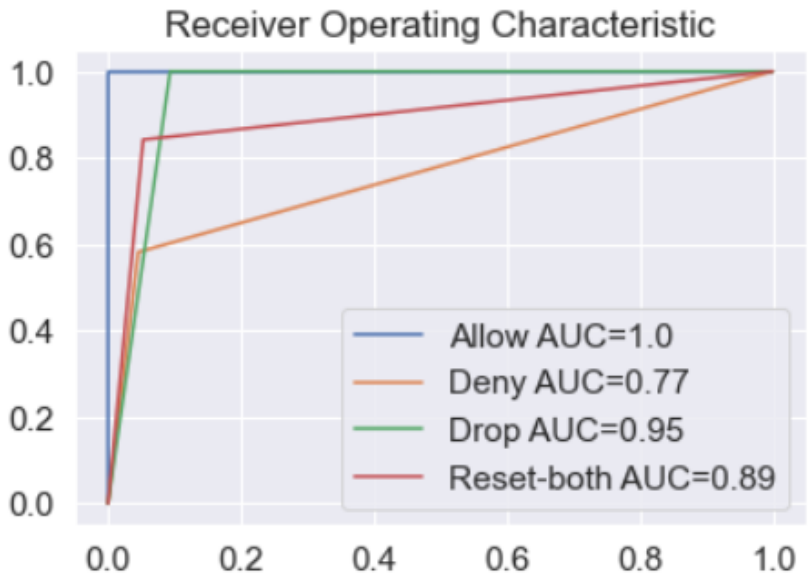
- **Confusion Matrix**
-



	precision	recall	f1-score	support
0	1.00	1.00	1.00	11395
1	0.81	0.58	0.68	11241
2	0.78	1.00	0.88	11168
3	0.84	0.84	0.84	11364
accuracy			0.86	45168
macro avg	0.86	0.86	0.85	45168
weighted avg	0.86	0.86	0.85	45168

(0 - “Allow” , 1 - “Deny”, 2 – “Drop”, 3 – “Reset-both”)

- **ROC (One VS Rest)**



สรุป

Accuracy โดยรวมจะอยู่ที่ 0.855 แต่เนื่องด้วย เรื่องนี้เกี่ยวข้องกับงาน Firewall ซึ่งต้องอาศัยความถูกต้องสูง จึงจะพิจารณา Recall เป็นหลัก โดยตัวที่มี Score Recall ต่ำอย่าง “Deny” จำเป็นต้องมีตัวแปรที่บ่งบอกถึง Action “Deny” ที่แตกต่างจากตัวอื่นมาพิจารณาเพิ่มเติม ประกอบกับ ROC curve ที่ตัวของ “Deny” มีคะแนนต่ำสุดเช่นกัน แสดงว่า Action “Deny” หาความแตกต่างแยกกว่า Action อื่นๆ