機器學習期末報告

R04921008	R05921012	B02901136	R04921010
邱沛淮	吳宗澤	蓁家志	徐世曦

***Abstract*—隨著行動網路的普及、實體服務的網路化，網路與資訊的安全也越發重要。近年來駭客的手段層出不窮，如何偵測與分辨網路攻擊已經成為重要的課題。本次專題以機器學習的方式，比較 deep neural network(DNN) 、random forest classfier(RFC) 、 self-training、boosting 與 binary classifiers 在偵測網路攻擊上的表現，我們的方法在 DAPRA 1998 dataset 作為測試資料，實驗證明我們的方法針對 imbalance 資料仍能達到很好的辨識率。**

I. Problem Formulation

此次專題選擇 Cyber Security Attack Defender 作為題目，目標是在 DAPRA1998 data set 中做網路行為偵測，在此 data 中共可分成 41 種攻擊與正常連線(normal)，此次專題目標為將輸入的網路行為 features 分成五類:normal, DOS(denial of service), R2L(unauthorized acc-ess from a remote machine),U2R(un-authorized access to local superuser root pri-vileges), probing(surveillance and other probing)。而 training 與 testing 的 data 都有 41 維 feature，每個維度對應的內容請參考 *Fig. 1*。traing data 共約 450 萬筆並被 label 為 42 類(即 41 種攻擊與正常連線)之中，另外有約 60 萬筆的 testing set。

Feature	Description	Type
1. duration	Duration of the connection.	Cont.
2. protocol type	Connection protocol (e.g. tcp, udp)	Disc.
3. service	Destination service (e.g. telnet, ftp)	Disc.
4. flag	Status flag of the connection	Disc.
5. source bytes	Bytes sent from source to destination	Cont.
6. destination bytes	Bytes sent from destination to source	Cont.
7. land	1 if connection is from/to the same host/port; 0 otherwise	Disc.
8. wrong fragment	number of wrong fragments	Cont.
9. urgent	number of urgent packets	Cont.
10. hot	number of "hot" indicators	Cont.
11. failed logins	number of failed logins	Cont.
12. logged in	1 if successfully logged in; 0 otherwise	Disc.
13. # compromised	number of "compromised" conditions	Cont.
14. root shell	1 if root shell is obtained; 0 otherwise	Cont.
15. su attempted	1 if "su root" command attempted; 0 otherwise	Cont.
16. # root	number of "root" accesses	Cont.
17. # file creations	number of file creation operations	Cont.
18. # shells	number of shell prompts	Cont.
19. # access files	number of operations on access control files	Cont.
20. # outbound cmds	number of outbound commands in an ftp session	Cont.
21. is hot login	1 if the login belongs to the "hot" list, 0 otherwise	Disc.
22. is guest login	1 if the login is a "guest" login; 0 otherwise	Disc.
23. Count	number of connections to the same host as the current connection in the past two seconds	Cont.
24. srv count	number of connections to the same service as the current connection in the past two seconds	Cont.
25. serror rate	% of connections that have "SYN" errors	Cont.
26. srv serror rate	% of connections that have "SYN" errors	Cont.
27. rerror rate	% of connections that have "REJ" errors	Cont.
28. srv rerror rate	% of connections that have "REJ" errors	Cont.
29. same srv rate	% of connections to the same service	Cont.
30. diff srv rate	% of connections to different services	Cont.
31. srv diff host rate	% of connections to different hosts	Cont.
32. dst host count	count of connections having the same destination host	Cont.
33. dst host srv count	count of connections having the same destination host and using the same service	Cont.
34. dst host same srv rate	% of connections having the same destination host and using the same service	Cont.
35. dst host diff srv rate	% of different services on the current host	Cont.
36. dst host same src port rate	% of connections to the current host having the same src port	Cont.
37. dst host srv diff host rate	% of connections to the same service coming from different hosts	Cont.
38. dst host serror rate	% of connections to the current host that have an S0 error	Cont.
39. dst host srv serror rate	% of connections to the current host and specified service that have an S0 error	Cont.
40. dst host rerror rate	% of connections to the current host that have an RST error	Cont.
41. dst host srv rerror rate	% of connections to the current host and specified service that have an RST error	Cont.

Fig. 1 training set 各個維度的定義

TABLE I
隊員分工

隊名	NTU_r05921012_四個大帥哥要成為 ML 訓練師		
分工	**研究**	**程式**	**報告**
宗澤	✓	✓	✓
家志	✓	✓	✓
沛淮	✓	✓	✓
世曦	✓	✓	✓

II. Team Name And Work Division

隊伍與各個隊員的分工列表在 TABLE I。

III. PreProcessing and feature engineering

根據統計，training data 在 5 類行為的分布列在 TABLE II。由此得知，5 類在 training set 的分布極不均勻。在 training 時將因為看太多或太少某一類的 examples 使得 train 的結果只是把所有 input data 分類到最常出現的那類。

A. *Discretization of Symbolic Features*

因為有些維度 feature 的 value 是 dis-crete 的 symbol，為了能夠作數學運算，需要對其做額外的處理，本次專題共分別使用了兩種方法：

1) *One-Hot Encoding*

One-Hot Encoding 將一個有限種類的值映射到一個向量。對 feature 中的任一個有 m 種值的維度f_i，$f_i \in \{d_1, d_2, \ldots, d_m\}$，定義函數 OHE(.)將 feature f_I 映射到向量 v_i。

$$v_i = \mathrm{OHE}(f_i) = (\delta_{d_1 f_i}, \delta_{d_2 f_i}, \ldots, \delta_{d_m f_i}) \quad (1)$$

其中$\delta_{d_j f_i}$在$d_j = f_i$時為 1,否則為 0。此方法將原本 feature 從 41 維擴展成 122 維。

2) *Dummy Coding*

Dummy Coding 將一個有限種類的值映射到整數。定義函數 DC(.)將 feature f_i 映射到向量 u_i。

$$u_i = DC(f_i) = \Sigma_j \delta_{d_j f_i} \cdot j \quad (2)$$

因為u_i是 1 維，所以總 feature 數在此情況還是 41 維。

B. *Feature Reduction*

1) Select-K-Best Features by Variance:

藉由計算每個 feature 維度在 training set 上的 variance，保留 variance 最大的 K 個 features ，其餘捨棄。

2) 依據 confidence 大小選

將 training data 用較簡單的分類器分類(i.e. random forest)後，計算每個 feature 對此分類結果的重要性(confidence)(i.e. Gini impurity)。最後，一筆 training data $x = (x_1, x_2, .., x_N)$可藉由函數$FR(x)$降維。

$$FR(x) = (v_a, v_b, \dots, v_c) \quad (3)$$

其中，$v_a = x_a, v_b = x_b, \dots v_c = x_c$ 且$x_a, x_b, .., x_c$符合使此簡單分類器能分類效果較好的特性。例如:選 confidence 最高的前 c 個或選 confidence 大於某個閥值的。

C. *Depulicate Training Data*

如 III 開頭所述，因為 data 分布極不平均,example 少的 data 將可能影響不了 model(從 neural network 的角度來看，neuron 被少量 data 的類更新的次數少，因此 model 對這樣的 data 的反應也小)。因此增加少量 data 的類的 data 數將能改善此問題。直接複製 training data 使得同一筆 data 可以被 model 看到更多次。

D. *Generate Training Data*

另一種增加少量 data 的類能在 training 中被看到更多此的方法是直接產生新的 data。即對少量的 data 做 oversampling[1]。

TABLE II
training data 的分佈

	攻擊方式				
	normal	DOS	U2R	R2L	probing
百分比	19.8%	79.2%	0.00097%	0.022%	0.8%

E. *Label Processing*

本次的 task 是將網路行為區分為 5 類，然而 training data 的 label 是 42 種攻擊方式。另外，實際出現在 training data 裡的攻擊僅有 23 種。所以引申出 3 種可能的 model:(i)輸出 5 類的 model，(ii)輸出 23 類的 model，(iii)輸出 42 類的 model。針對(i)，須將原本 label 的 42 種攻擊映射到 5 類，例如：label 為 smurf 的網路攻擊對應的攻擊方式是 DOS，因此這筆 data 的 label 改為 DOS。針對(ii)跟(iii)則是先 train 一個 model，之後根據 model 預測出的 label 再映射回 5 類。例如：model 預測 label 為 smurf，則結果應為 DOS 攻擊。

IV. PROPSED MODELS

介紹完 training data 中的 feature 的選取和處理的方法之後，接下來會針對此問題所使用的 classification model 討論。在 III 中談過此一問題為在分布不均的 data 上進行分類，所以我們提出了三種不同方向的 model，分別為 DNN(Deep Neuron Network),Decision tree based classifiers, 以及 weak classifier based model，接下來會簡單介紹這三種模型的原理。

A. *DNN based models*

基本的 neuron network 是由一層 input layer 和中間的 hidden layer 以及做末端的 output layer 組成，其中 input layer 和 output layer 的維度會取決於 input data 和 output class 的維度，而中間的 hidden layer 則含有特定數量的 node，node 的數量須以研究者的經驗來調整多寡，以取得針對特定問題最好的 model。

DNN 全稱 Deep neuron network，是除了 input layer 和 output layer 之外，有著複數 hidden layer 層的類神經網路， 其特色在於可以將分類問題中需要學習的 feature 透過多層的架構來達成自動化的學習。有別於傳統的分類問題，需要科學家以長年的實驗結果和驚人的洞見來摸索出利於分類的關鍵 feature，DNN 可以使

input data 經過多層的轉化，自動的變成分類所需的 feature。此類方法與其衍生方法因為操作容易，應用廣泛，目前在很多領域中的分類問題上仍然十分活躍。

在資安防治的問題上我們使用了 DNN based model，除了調整其 hidden layer 的結構之外，還調整了其他 model 的細部參數，關於詳情在 V 章節會有更詳細的敘述。

B. *Decision tree based models*

在 Decision Tree 中，我們假設所有的 class 與其 feature 可以藉由樹狀圖表示。最下面的 leaf 是最後結果的 class，而中間分岔的決策節點就是由 feature 所組成，決定要往哪邊展開。在 training 的過程中，我們藉由遞迴呼叫的方式不斷的延展我們的 tree，直到分支不能再進行分割為止，便是一棵最簡單的決策樹。

決策樹的優點在於，不含有太多複雜的數學模型，且不同於其他學習方法，它所需要的前處理是較簡單的。然而它也有些許缺點，包括當處理的分支非常複雜且大量的時候，它所要消耗的時間也是同等的龐大。

最後，因為一棵 Decision Tree 的隨機性太大，之後又發展出了一次訓練多棵，並使用它們的眾數或是平均來預測結果。在之後所會用到的 random forest classifier(RFC)跟 extra tree classifier，即是類似這樣的變形。

C. *Weak classfier based models*

1) *Adaboost on tree based models*

因為個別的 tree base model 都無法得到足夠高的辨識率，並且每個 model 至少都有超過 50%的辨識率。這樣的條件符合 adaboost 需要的 weak classifier 的限制。藉由 adaboost 每次增加辨識錯誤的 example 的 weighting，最後 train 完的 model 預期將比完全不做 adaboost 的有更好的辨識率。

2) *One vs One Classfier*

因為各個類的 data 分佈不平均，因此能提升資料使用效率的方法預期能提升辨識率。因此使用 One vs One Classifier，它的基本結構如 Fig. 3。

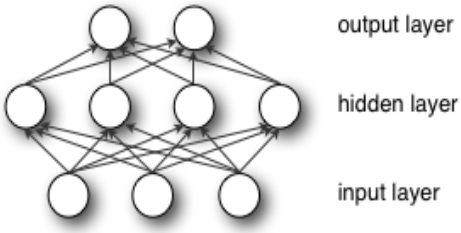

Fig. 2 Basic idea of neuron network

針對C_0個 class(這次實驗$C_0 = 5$), 每次選擇兩個 classes 的 data，train 一個 binary classifier，當 testing 時輸入的 data 將經過$C_2^{C_0}$個 binary classifier, 並選投票票數最高者當作 predict 的 class。以 5 個 classes 為例，此 model 將產生 10 個 binary classifier，並且因為投票決定的關係，既使少數幾個情況因為某兩類數量差距過大而 training 效果不夠好，仍可預期總體投票結果將改善辨識率。

V. Experiment and Discussion

A., *Testing on Validation Sets*

因為只有 training data 有 label，切出部分 training data 作為 validation set 能驗證 train 完的 model 是否夠好。

然而，這樣的方法隱含著一個假設：training data 的資料分佈跟 validation 的資料分佈相似，也跟 testing data 的分佈相似。Training set 跟 validation set 的相似性可以用 random sample 的方式產生 validation set 來滿足。而 testing set 的資料分佈在這個 task 中則與 training set 相當不同(約差 5~10%的 data)，因此大部分 validation score 能達到 0.9999 的 model，在 testing set 的表現則變化很大(約 0.95)。

因此針對需要達到極高辨識率的情況(即本

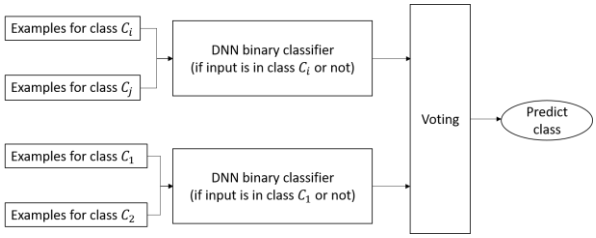

Fig. 3 One vs One Classifier

次題目)，validation score 無法提供足夠判斷 model 好壞的資訊。相反的，在要求的辨識率並非這麼高時，validation score 的高低跟 model 的好壞仍有正相關。

B. *Implementation details*

因為 training 資料有四百多萬筆 data，使用 python 內建的 list type 雖然速度快，但是會需要過多的記憶體(8GB RAM 最多只能放置約兩百萬筆 data)。所以在實驗中將著重在兩種解決方法,並對應得調整 training 的方法：(i)將 training set 切成N_t份,此方法等同於讓 model train 在N_t個 independent set 上.(ii)使用 numpy 提供的 array type 來存資料，然而在 array 後新加上一列 training data 進去的速度很慢，因此用 list 存一部份後再將此 list 加進 array 的方法能同時有高速又有高記憶體效率的優點。

C. *Testing on Real Data*

(1).DNN based models

首先，我們嘗試了單層 512 與 3 層 256-512-1024 兩種不同的 DNN 架構(TABLE III)，可以發現不論是否有作 validation 來讓 training 有 early stopping，單層 512 的效果表現都比多層來好。

1) 接下來，因為越深的 network 通常效果比單層多 neurons 效果還好，我們選了參數量差不多的 128-128-128 來比較。除此之外，我們測試了不同的 activation function，一個是 ReLu (Rectified Linear Unitsnits)，另一個則是 Leaky ReLu，並在每層用上了 Batch Normalization。

$$\text{ReLu: } f(x) = \max(0, x) \tag{4}$$

TABLE III
不同 hidden layers 在 testing set 上分數的比較

	single layer 512	256-512-1024
Without validation split	0.9528	0.71101
validation split 10% of training data	0.89416	0.76271

$$\text{Leaky ReLu: } \begin{cases} f(x) = \alpha x & \text{if } x < 0 \\ f(x) = x & \text{if } x \geq 0 \end{cases} \tag{5}$$

這兩種方法的差別在於負值地方的處理，一種在小於零的時候讓值等於 0，另一種則會在小於零的時候讓值與 x 成正比。經過測試，LeakyReLu 效果明顯比 ReLu 還好。將 training epoch 加到 10 以後效果也更加卓越。

另外將 hidden layers 改為 64-64-64、128-128-128 和 256-256-256(TABLE IV)， 64-64-64 效果則和 128-128-128 差不多，但 256-256-256 很明顯效果變差的，原因為 overfitting。

接著，藉由分析了 training set 裡面的資料，發現雖然 attack list 上面共 42 種攻擊，但實際上出現在 traing set 的只有 23 種而已，也發現最後 testing 所需分成的 5 大類的分佈與在 training set 裡的散佈極為不同，可能導致我們 model 不能成功辨識 testing data，大多數的參數變化只受到那些大量資料的影響而改變。

因為原本是將 training output 定成 42 類，後來將 output 定為 5 種模式或 3 大種模式(在 training set 可以發現主要出現的 3 大種是 DOS、Normal、probing)及 23 種的模式(在 training set 上只出現過 23 種模式)。

為了處理資料不平均的問題，我們嘗試了兩種不同的方法，首先，把每筆不同 label 的資料乘上不同的 weighting 去作參數的更新，而那些參數是和每種的數量成反比，另一種方法則是直接將比較少的資料複製個幾倍丟進去 train，比較多的資料則 resampling 幾倍丟進去 train。

最後，分成 5 個 class 且對各個 class 加上 weighting 的效果最好。直接複製 data 效果不好的原因則是因為使用 adam 做 gradient decent 的時候，會受到 data 的數量影響，因此可以在 TABLE IV 最後一列很明顯看到差異。

我們在 DNN 試的最後一種方法則是 semi-supervised 方法的一種 self-training，將每次判斷完 testing data 裡信心度(confidence)超過一定程度的加回去 training set 中去做下一個 epoch 的 training，

由先前實驗可以發現，我們的 model 在分 normal 與 DOS (第 0 class、第 1 class)時表現很好，其他 class 則較差。因此我們針對後 3 類用較低的 confidence 而前 2 類則用較高的 confidence 作 self-training。結果在 TABLE V 可以發現 self-training 的確有助於效果更好，且信心度選擇大的效果比較好。

在做了各式各樣不同的 DNN 測試以後，我們的 model 分數一直超不過去 0.96 的一個瓶頸，因此，去嘗試了 Decision tree based model。

(2). Decision tree based model

一開始我們嘗試 random forest class-ifier (RFC)跟 extra tree classifier (ETC)以及不同維度的 features 來當 input data，主要分為 41 維和 122 維。由 TABLE VII 可以發現對 symbolic data 做 one-hot-encoding(122 維)的效果比較好。

TABLE IV
Class weighting 與複製 data 的比較

Hidden layers	Output layer	Class weighting	score
128-128-128	3	No	0.95157
128-128-128	5	No	0.93755
128-128-128	3	Yes	0.95424
128-128-128	5	Yes	0.95566
128-128-128	23	Yes	0.94982
128-128-128	42	Yes	0.94335
512	5	Yes	0.92941
128-128-128*	5	No	0.88585

*u2r data 複製 200 次，r21 20 次，DOS resample 到 normal 的 10%的量

TABLE V
Self-training 在不同 confidence 的比較

Hidden layers	Confidence (class 0 and 1)	Confidence (class 2 to 4)	score
128-256-512	0.999	0.7	0.95936
128-256-512	0.999	0.9	0.95982

TABLE VI
Decision tree 與 RFC 不同參數的比較

method	estimators	Feature dimension	score
RFC	10	41	0.96021
RFC	10	122	0.96087
RFC	20	41	0.96105
RFC	20	122	0.96076
ETC	20	41	0.95908
Decision Tree	N/A	122	0.96087

之後，我們開始多做了其它的參數調動: estimators(tree 的數量)改為 20 個，並測試了 41 維及 122 維的不同、只用 Decision Tree 配 122 維(TABLE VI)，但效果沒有明顯差異。

在這階段實驗中，最高的是 RFC estimators = 20，41 feature dimensions 這樣的配置，我們發現 random seed 的影響似乎很大。

接著，我們還試了 sklearn 裡面的 OneHotEncoder、LabelEcoder 及 SelectKBest (K=1000)做資料進去 RFC 的前處理。一次是對分完 41 維再做，另一次則是直接把連 symbolic feature 都還沒處理過就直接做。

可是發現做了這樣的 processing 以後效果反

TABLE VII
不同 tree classifier 與不同 feature vector 的比較

method	Number of tree	Feature dimension	score
RFC	10	41	0.96021
RFC	10	122	0.96087
ETC	20	41	0.95908

TABLE VIII
RFC 做前處理

After our own preprocessing(41 維)	0.95774
Before our own preprocessing	0.95897

而沒有變好(TABLE VIII)。並且，根據 RFC 的算法，發現的確不需要這樣的前處理。

除此之外，我們還將 RFC 配合了 adaboost，可是效果也沒有明顯進步(TABLE IX)。可能的原因是 adaboost 次數太多在 imbalanced data 的情況會導致 overfitting

之後，我們開始調各種 estimator 的數量以及每顆樹能使用的 feature 數量來觀察 oob(out-of-bag)的 score 來比較，期望能從中選出在 testing 上最好的 model。

從 TABLE XIII 中，可以發現在差不多 20 個 features 以上的時候，每個不同 estimators 在 oob score 的表現都差不多接近滿分了，但因為這些只是在 training set 上每個樹的平均表現，並不一定會代表在 testing set 上會有很好的表現。

從後面的測試(TABLE XIII)後也會發現 oob 的分數似乎沒有很大的參考價值，因為相近的 oob score 在不同的 estimators 數目或是 features 數量還是會很明顯的影響到結果。

TABLE X 為 ETC 中兩種不同 estimators 數量及 output class 變為 23 大類在 kaggle 上的分數，可以發現 estimators 的分數一直突破不了 0.96。

TABLE XI 則是我們測試各種 RFC 參數的表，當 estimators 數量適中的時候有最好的表現，並控制 feature 數在 27 時能有最高的 score。

TABLE IX
Adaboost 結合 RFC

Adaboost iterations	estimators	score
100	10	0.95994
100	80	0.96063

TABLE X
使用 ETC 的比較

estimators	Number of classes	score
20	5	0.95908
30	5	0.95967
30	23	0.95965

D. *Model comparision*

根據前面的實驗結果，我們可以得到一組符合此分類問題的最佳化參數，稍後會將以此組參數帶入我們的各種 model 中並進行最後的分類成果比較。在那之前我們先將最終選定的數個 model 做簡短的介紹。1)首先是根據 IV.A 部分曾經介紹過的 DNN based model，我們在這裡最終選擇的是三層皆是 128 維的 DNN，配上每層都有做 batch normalization 和 LeakyReLu 來優化 model；2)而關於 decision tree based model 我們選擇了 Random Forest Classifier 配上 25 個 estim-ators 和最大深度 27 來作為 model 的參數；另外 3)我們也選用性質相似的 Extra Tree Classifier[2]來進行比較。接著 4)，以 DNN 的 model 為基礎我們使用了 semi-supervise learning 的概念來將 test data set 中的 data 也進行 label 並加入 training，這樣的做法證實的確對於分類器的訓練有幫助；5)另一個 model 則是源自老師上課講到的 Ada-boost，我們也將其實作並加入比較，6)最後一個 model 是根據我們在參考網路上的資訊和從以前到現在的

TABLE XI
RFC 各種參數的比較

estimators	Number of classes	score
10	5	0.96021
20	5	0.96105
100	5	0.96028
50	5	0.95967
50	23	0.9605
25	20	0.9612
25	**27**	**0.962**
25	30	0.9617
25	35	0.96152

TABLE XII
Model comparison

method	score
DNN	0.95784
RFC	0.962
ETC	0.96019
Self Training+DNN	0.95982
Adaboost+RFC	0.96063
1vs1	0.8961

做法的途中找到的一種概念作為發想而來的，其核心概念是希望透過訓練 1 對 1 分類器來實現準確分類的目標，以我們有五個 class 為例，我們必須先訓練出C_2^5個 1 對 1 分類器，再將之分類的結果結合來得到正確的分類結果。

根據 TABLE XII 分數顯示，Random forest classify\er 獲得最高的分數，其原因我們可以推測是由於能夠將弱勢的 class 也成功的分類的關係，而這一切都要歸功於其 model 隨機的特性，使得分布不均的資料的影響能夠減小。而 DNN 和 Self-training 的 DNN 分別緊追在後，但是卻仍有一段不小的差距，這是因為純 DNN based 的方法終究還是對於這種不均勻的資料分布感到棘手，即便是加上了 self-training 的機制，使得分類器有更多機會學到弱勢 class 的分辨方法，仍然是一傅眾咻，無力回天。

而 Extra tree classifier 表現雖然也不錯，但是卻在訓練中顯示出中 over fitting 的跡象，例如其 accuracy 很高，卻沒有辦法在 kaggle 和 validation set 上展現同樣的高分，足見其已經 over-fitting 了。而老師介紹的 Adaboost 也是表現優異，但是之所以沒有得到更高的成績原因可能出在其所用的 RFC 分類器已經被訓練得很好了，故沒辦法再有所突破。最後 1vs1 分類器並沒有如文獻上所看到的發揮作用，背後的原因很多，我們認為最主要的原因可能是 class 之間的分布實在過於不均勻(將近好幾個數量級)，導致個別分類器的訓練上有所困難，以至於整體效果不佳。

VI. CONLCUSION

綜上所述，我們可以發現，其實大部分實驗中的 model 都可以有不錯的成績(辨識率 95%)，然而，在 DAPRA1998 這份 data set 裡面，真正困難的是對於小眾 class 的辨識，如何在不足 training set 的情況下依然能夠將最少的兩個 class 分辨出來，即是能不能百尺竿頭更進一步的依據(96~97%)。而這也符合我們實作資訊安全的一些特性：如果駭客使用的是之前甚少出現過的攻擊方式，我們能不能將其辨識出並且阻擋在外？最後由實驗結果發現，在加上這方

面的考量後，Decision Tree based 的方法加上 feature 的選取似乎是一個不錯的選擇。

REFERENCES

[1] G. Batista, R. C. Prati, M. C. Monard. “A study of the behavior of several methods for balancing machine learning training data,” ACM Sigkdd Explorations Newsletter 6 (1), 20-29, 2004.

[2] Geurts, P., et al. (2006). "Extremely rando-mized trees." Machine Learning 63(1): 3-42.s

TABLE XIII
OOB score

Estimator / Features	2	7	12	17	22
1	0.652	0.922	0.956	0.950	0.959
6	0.676	0.965	0.995	0.998	0.999
11	0.679	0.967	0.996	0.999	0.999
16	0.679	0.967	0.996	0.999	0.999
21	0.679	0.967	0.996	0.999	0.999
26	0.679	0.967	0.996	0.999	0.999
31	0.679	0.967	0.996	0.999	0.999