Machine Learning 2016

<期末報告>

題目: Cyber Security Attack Defender

Team name： NTU_r05942042_R20? Daikin R32 is the new trend! A_____A

Team members：

學號	姓名
r05942042	王奕翔
r05942013	楊凱勛
r05942073	周華佑

Work division：

大家齊心協力團隊合作～

Preprocessing/Feature Engineering：

這個比賽是要使用41維的feature做classification，原始的資料之中第2,3,4個feature是屬於categorical的feature, 這部分我們有試過dummy coding (每種類別給他一個獨特的正整數)，以及one-hot coding，也就是將categorical的feature展開，每個category都給他一個欄位，展開之後一共變成120維的feature。

在做dummy coding的preprocessing的時候，就僅有使有python做一些unique member的mapping而已。而one-hot coding 的部分則是使用了sklearn preprocessing 的 OneHotEncoder.

除了feature以外，這次比賽的資料之中一共有數十種的連線模式，但是實際上我們classification的結果只會有0, 1, 2, 3, 4共五種class，從數十種的連線模式到這五個class間的mapping方式由"training_attack_types.txt"提供。在實做的時候有兩種選擇，一種是將training data的連線模式事先map到這五個class，另一種則是做數十種class的classification，在最後轉換成submission format的時候再把它map回這五個class。我們最後主要採取的模式皆為前者，也就是在preprocessing時就轉換成5個class了，一方面我們有做過實驗，實驗結果確實比較差，predict 出來的結果幾乎就只有class 0 和 1了。此外，這次比賽的挑戰在於說我們要用現有的data 去 predict novel attacks，從training data的分布來看，推測應該是有不少的attack types被擷取放到testing data之中，因此事先將連線模式概括性地轉換成5個high-level hierarchy的概念是比較合理的。

如前段所述，事實上也因為這樣training data和testing data有所落差。例如column 19 (第20個column) 在training data全部都是0，而在testing data中會出現不少的1。此外，我們也有使用過PCA去降維，試圖透過比較少的feature去做出更好的預測，實驗結果如報告最後面的表1所示。此外，在做NN的時候我們有使用normalization，做法是把 training data 和testing data concatenate在一起，每個column一起做normalization。但是我們最後實驗結果發現random forest和decision tree還能比NN更上一層樓，而normalization對於這兩者而言其實沒有太大的影響，基本上只是因為random_state之類的因素導致在分每個branch的時候會有一點些微的差距。因此為了簡便就把normalization拿掉了。

最後，training data其實存在非常多的重複data，可以選擇是否要把重複的data剔除掉。如果保留的話，無論使用的是何種classifier，都可以當作是一種weighting來讓predict出來的結果不會偏離這個weighting太遠，講白了就是讓predicting的結果的variance減少。但其實後面為了Kaggle衝榜所採取的方法是希望variance能夠足夠大才行，因此最後有把這些重複的data刪掉來增加classifier的variance。

統整來講，最後Kaggle成績最好的prediction所使用的preprocessing流程大致如圖1所示：

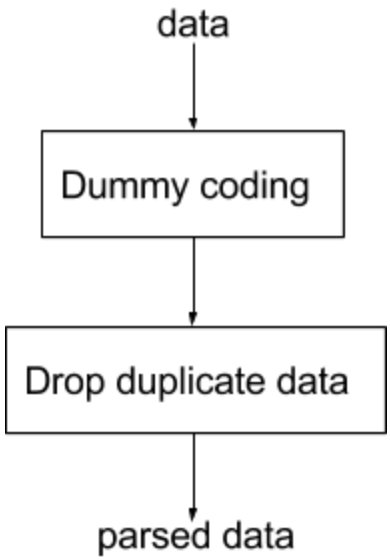

圖1、最後 Kaggle best 的 preprocessing 流程圖

Model Description：

Model1: Neural Network

我們有試過很多種不同架構的NN，基本上流程圖如下，詳細的layer架構和parameter設計可以參考"Experiments and Discussion"一節。

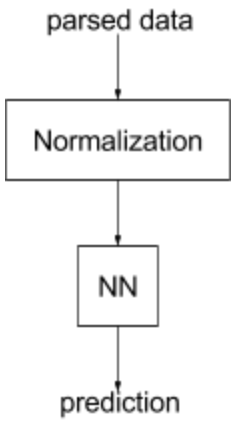

圖2、NN的流程圖

Model2: Decision Tree

與decision tree有關的架構基本上normalization影響不大，因此在model的設計上我們並沒有使用normalization。此外，我們所使用的套件是sklearn的DecisionTreeClassifier，大部分的實驗都是使用該model default的parameter。因此其實此model的架構相當的簡單，基本上就如下圖所示：

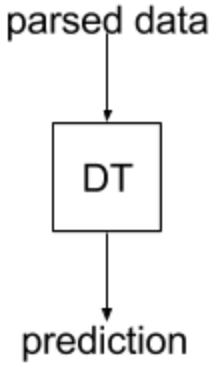

圖3、DT的流程圖

Model3: Random Forest

我們所使用的套件也是sklearn的RandomForestClassifier，主要與default不同的參數為n_estimator=1。之所以會使用n_estimator的原因是在於經過多次的submission發現大部分的classifier即便training的validation都可以很輕易地達到99.9%以上，但是public score頂多做到accuracy約96%就無法再上去了，這主要是因為testing data要predict的連線模式似乎大部分是training data沒有見過的，因此我們要從固有的data去做prediction會有它的極限。因此使用n_estimator=1主要是希望搭配default為true的Bootstrap功能，在train的時候使用取後放回的方式，透過此方式使得random forest能夠建出"有特色的" tree。

基本上model的架構和DT一模一樣，只把DT的block改成RF而已

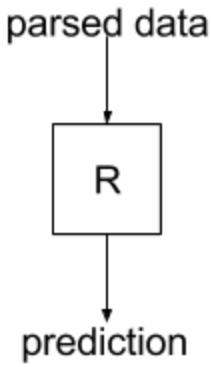

圖4、RF的流程圖

Model4: Random Ents (Kaggle best)

(Mixing random forests each with n_estimator=1)

在presentation過後，我們得知了testing data各class的真實分佈。因此我們最後主要的目的在於說去盡量fit真實的分佈，但同時又要維持predict的準確性。從過往的submission經驗得知說class 3 往往是最後準確率的關鍵，因此我們選了一個class 3

predict結果相當多的RF1，以及一個class 1 predict相當多的RF2，再加上原先kaggle上最好的RF3，依序由RF1~RF3做最後prediction的寫入。之所以會叫做random ents是因為他們本身其實已經失去了forest的概念(n_estimator=1)，而他們各自其實也都算是萬中選一的相當不錯的RF。model架構如下圖所示：

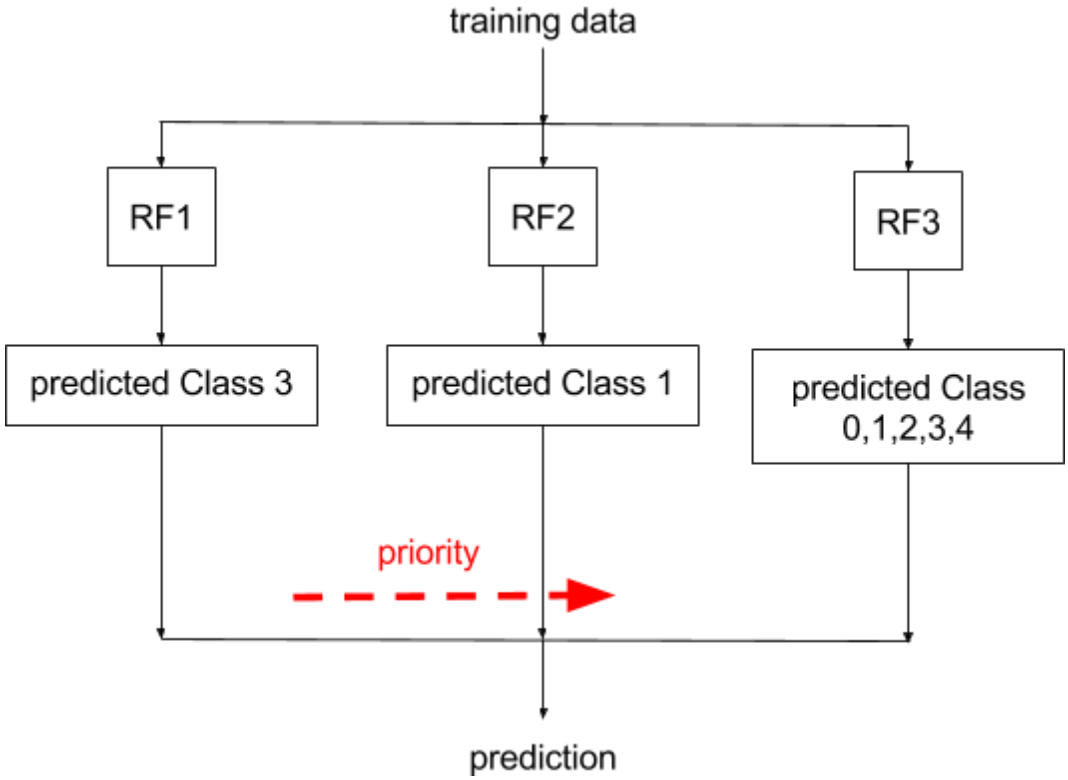

圖5、RE的流程圖

我們也曾經試過使用各個class各自分工的方式去對各個class做prediction，但是結果都沒有很好(如最後一節的圖6所示)，最後兩天發現主要的prediction還是仰賴一個比較穩健的RF model比較好，例如上圖所示的RF3。

Experiments and Discussion：

ANOVA f-test of features:

ANOVA (Analysis of Variance) F-test scores為檢定samples 與number of class的一個test，即F-test的scores越高，number of class (in our case, number of class is 5)。更精確來說，我們可以把ANOVA視為一個Hypothesis testing 所得的optimal test，而 $H_o : \mu_1 = \mu_2 = ... = \mu_k$，其中 k 為number of class（亦即 H_o 表示number of class 不為5的假說），而F-test越高則表示 H_o 不為真的機率越大。
ANOVA則是這個假說檢定的一個test。其F-test scores公式如下：

$$\frac{BMSS}{WMSS} = \frac{\sum_i (Y_{i-}\overline{Y})^2}{k-1} \Big/ \frac{\sum_i \sum_j ((Y_{ij-}\overline{Y})^2}{N-k}$$

也就是組內變異與組間變異的比值。

然而ANOVA的假設有二：樣本必須接近常態分佈，並且組內變異數相等，但這在我們的情況卻未必符合。

圖表6中的左圖是 dummy coded features 的 f-test score，而右圖則是 1-hot coded features 的 f-test score，也就是說右圖的第2~50個bar是左圖的第2~4個bar展開後所得到的feature。

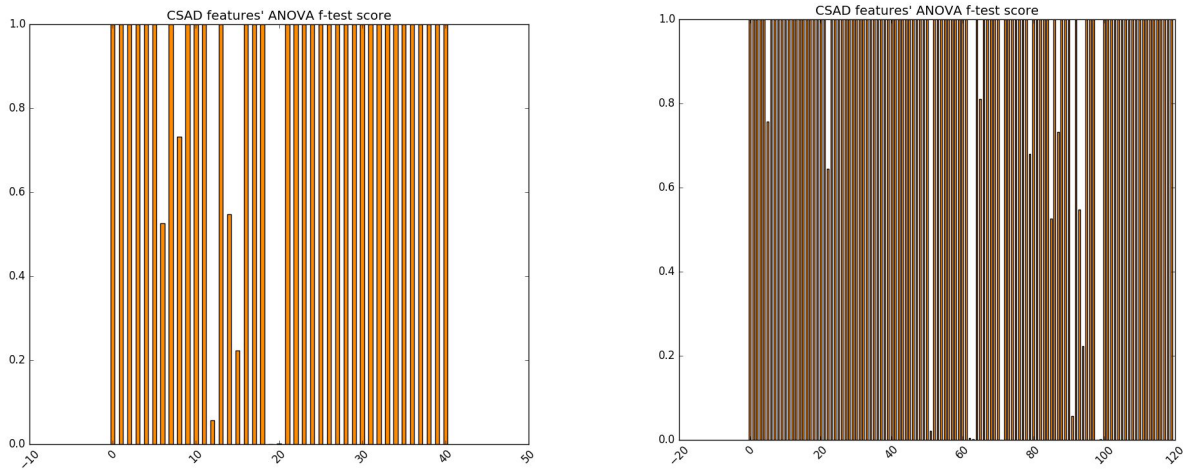

圖6、dummy coded feature 的 f-test score

Discussion：

可以看得出來絕大多數的 feature 對於這個 classification 的 task 都相當具有代表性。左圖中，第20個 column 之所以會是0是因為 f-test 是一種supervised的 feature analysis，因此只能被應用在有label的training data 上，而如第一段所述，在 training data中第20個column 都是0，但其實在testing data中卻不然，因此也不能說其沒有代表性。如果今天的task不是multi-class classification而是單純的 anomaly detection，應該可以把這應用到實務上。

Classifiers:

在做這個題目的時候，基本上在training phase的時候任何classifier都可以很容易達到validation accuracy>99.99%，因此後面這個段落所著重的比較metric為Kaggle的public accuracy score.

Neural Network (NN):

#	NN setup	score
1	MLPClassifier, hidden_layer_sizes=(20, 10, 5, 2)	0.94504
2	MLPClassifier, hidden_layer_sizes=(100, 50, 20, 10, 5, 2)	0.96087
3	和 #2 相同，但有對feature先使用PCA，把最爛的兩個feature捨棄掉	0.95633
4	Keras, 5x512FC, output = 2 (即只predict 0, 1 兩個class)	0.94834
5	Keras, 5x512FC, output = 5	0.96079

6	Keras, 4x512FC, output = 5	0.95987
7	Keras, 4x512,1x256 output = 5	0.96040

表1、NN相關的experiments

Discussion：

其實我們NN相關的submission共有18筆，但其實能夠達到的極限就差不多在接近0.961左右而已。由於testing data有很多training data沒有的pattern，因此NN layer數不夠多的話，很可能class 2, 3, 4這些相對比較小眾的class會predict不出來。但即便用keras堆了很多層fully connected layer，還是無法達到很高的準確率。

此外，從#2和#3這兩筆測資可以看出這個task做PCA其實並不會比較好。另外，如#4所示，我們也嘗試過只判斷0和1，但是結果也爛爛的。

事實上隨著submission 越丟越多我們越意識到其實關鍵不是在於class 0、1的分類，而是在class 2, 3, 4這幾個比較小眾的，特別是3和4。起初我們以為重點是在class 4，但是觀察丟出去的prediction結果我們是猜測說可能class 3在這兩者之中還比較關鍵。而在presentation那天其他組別有去硬試出實際的data分布，更是佐證了我們的判斷。

Decision Tree (DT):

事實上在NN無法做出更進一步的突破之後，我們首先跳槽的對象是random forest (RF)，但是最後的發展其實是：NN→RF→DT→RF，最後也是RF得到的結果比較好，因此這邊就先討論我們針對DT所做的實驗。

我們所使用的DT為sklearn的DecisionTreeClassifier，他所能調整的參數其實也不多。

#	DT setup	score
1	random_state=0	0.96137
2	random_state=0, 無視training data前三個column的feature	0.96029
3	random_state = 0, 刪除重複的data	0.95914
4	random_state = 1	0.96197
5	random_state = 1, class_weight = ‘balanced’	0.96013
6	random_state = 2	0.96180
7	random_state = 3	0.96296
8	random_state = 3, criterion = ‘entropy’ (default 是 gini)	0.95992

表2、DT相關的experiments

Discussion：

我們DT相關的submission也其實有接近30筆，表二所列的基本上是比較有代表性的，從#1,#4,#6,#7 可以看出其實random_state不同對DT的影響其實不小，特別是這題大家那時都擠在0.96出頭，因此能夠衝上#7的0.96296是還蠻振奮人心的，因此我們後面試了蠻多不同的random_state...但其實大抵都是在這分數附近。

當初之所以試#2是因為從分布猜測說testing data有很多training沒有的pattern，而從人眼判斷就可以發現其實前三個column對一些attack模式幾乎是deterministic的關係，因此想說故意去impair我們的classifier，讓他透過其他比較看似沒代表性的featrue去建立tree，事實上其實對分數也影響不大，由此亦可判斷出來training data和testing data差異不小。

Random Forest (RF):

在DT卡在約0.963之後，我們又做了一些其他嘗試，例如adaboost和random forest， adaboost的accuracy大概只有0.94左右而已，而之所以會轉為使用random forest作為此次作業的主軸是因為透過過往的submission經驗，發現n_estimator=1的RF variance相當大，有些random_state下predict出來的class分佈好像比先前NN和DT的都還好，因此最後主要著重的classifier為RF。

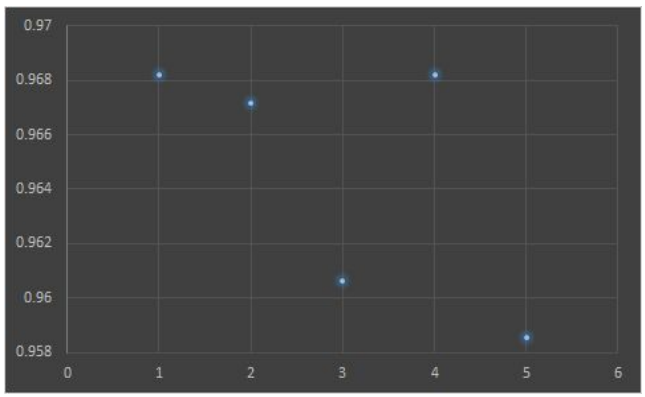

圖7、5筆RF的分數

此外，我們後來也發現了如果把training data中重複的data刪掉，RF的variance會更大，結果如下圖圖8所示。

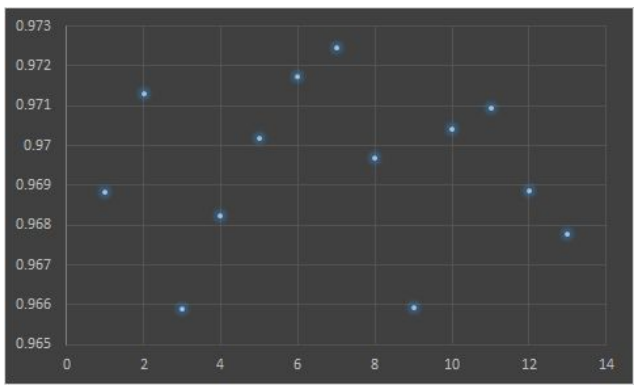

圖8、13筆RF的分數

Random Ents (Mixed Random Forest):

在presentation後，我們得知了其他組別試出來的真實distribution，如表3 “Real”那列所示，而"Kaggle best"那列是我們分數最高的結果。

	Class 0	Class 1	Class 2	Class 3	Class 4
Real	157556	432209	230	8568	8216
Kaggle best	168712	423960	43	6374	7690

表3、testing data各class的分佈

因此最後衝Kaggle時所採用的主要策略為mix一些random forest (n_estimator分別都為1)去硬湊出更接近該distribution的prediction結果。

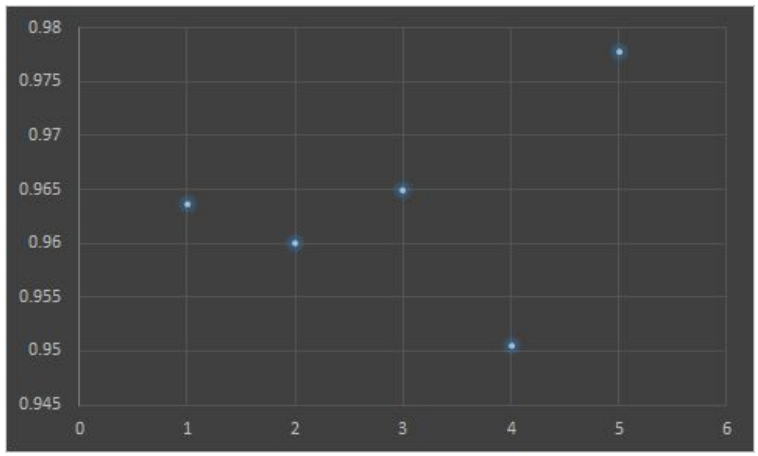

圖9、5筆mixed RF的分數

Discussion：

如果用一些mixed strategy去硬湊出符合real distribution的結果，會使得最後分數的variance更大。從直覺上來說是相當合理的，畢竟原先的predicted class大部分集中在class 0 和 1，這在統計上本身就佔有相對的優勢了。如果硬是把分布的absolute difference減少，很可能會落得全盤皆輸的下場。因此，由於data的限制，採用的個別model架構盡量越簡單越好，而且最好要有一個整體分數相對不錯的model當作為基底(如p.4所述)，圖9中的第五個點就是用這個方法得出的。

如果用p.4所述的調整方式，基本上分數會在0.977左右。但是這部份由於時間有限，我們也來不及多做嘗試。含圖9中的第五個點在內，一共僅投了三筆submission，大致上都落在0.977附近。