Machine Learning Final Project

題目: Cyber Security Attack Defender

隊名: NTU\_b03901057\_XXXYYY什麼鬼都可以吧

隊員:

b03901057 詹晉誠

b03901138 張晏祐

b03902035 黃兆緯

工作分配:

詹晉誠：Task1、Task3

張晏祐：Task1、Task3

黃兆緯：Task1、report

1. Preprocessing and Feature Engineering

1. Dataset

在做任何處理之前，首先檢查dataset的基本資料：

|  |  |  |
| --- | --- | --- |
|  | Training set | Testing set |
| 出處 | DARPA’98 IDS evaluation program | |
| size | 4408587 | 606779 |
| # of features | 41 | |
| # of continuous features | 34 | |
| # of discrete features | 7 | |

這個task跟hw2的0/1分類問題很相似，因此我們想用類似的方法來進行這個task，這兩個問題的差別有：

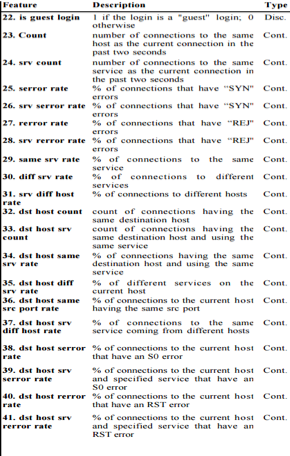
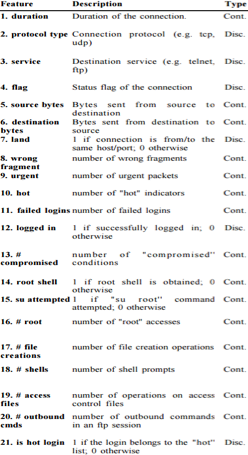
(a) hw2資料裡的feature都是連續的，這次有部份離散feature。

(b) hw2是0/1分類問題，這次是多分類問題（5個分類、40種攻擊）。

(c) hw2的資料集小很多，training set只有4000筆，而這次有四百萬筆。

為了用類似hw2的方法來進行，在前處理時需要將離散feature轉為連續的（有數字大小意義的）。使用的model需要可以處理多分類問題（one-versus-rest 或 multinomial）。另外，由於資料集較大，model的訓練時間是需要被考慮的因素。檢查完dataset的基本資料後，我們開始想辦法處理上述的三個問題。

1. Discrete features



根據投影片上的feature介紹，總共41個feature中，有7個discrete feature，分別是：protocol type、service、flag、land、logged in、is hot login、is guest login。應用一些網路攻擊的知識可以知道，這些discrete feature對於判斷攻擊類型有很大的作用。例如pod(Ping of Death)攻擊是使用ping，因此protocol type是ICMP。

為了讓這些discrete feature變為具有數值意義的feature，我們使用的是one-hot encoding，將每個種類變為一個維度，該維度僅有0/1兩種可能，0代表原始資料並不是這個種類，相反地1代表原始資料是這個種類。例如protocol type 有icmp、tcp、udp三種種類，則encoding舉例如下：

|  |  |
| --- | --- |
| 原始資料 | Encoded |
| icmp | (1, 0, 0) |
| udp | (0, 0, 1) |
| abc | (0, 0, 0) |

使用one-hot encoding後，discrete features都被轉為具有數值意義的feature（但這些新feature其實比較適合decision-based的模型，因為0/1的意義並不明確）。最後將原本的41維資料轉為122維資料，維度變為三倍。因此訓練時間變得更久，之後選擇模型要更加考慮訓練時間。

1. 多分類問題

從0/1分類問題變為5分類甚至40分類的多分類問題，其實並不是很大的問題，很多套件提供的模型都支援多分類問題（例如我們使用的scikit-learn），即使不支援多分類問題，也可以針對每個分類訓練一個0/1分類模型，再對這幾個模型的輸出進行組合即可。

1. Dataset Size

這個task的training set有440萬筆，是hw2的1100倍，這造成某些model的訓練時間太久，會不方便進行測試。

首先我們發現有一些data是完全一模一樣的，排除掉相同的data可以減少資料量，但這會導致training set的分布改變（完全相同的data也許是增加權重資訊），因此我們並沒有排除相同的data。

1. Feature Extraction

利用one-hot encoding後，變成122維的data。首先我們考慮每個feature間的交互關係，可以加上兩兩相乘的feature，但這樣一來每筆data就會超過10000維，遠遠超過我們能處理的大小，因此並不考慮這個方法。

利用chi-square test和相關係數來檢查feature和label的相關性。檢查的結果為除了特定幾個feature和label相關性特別高，其他feature和label的相關性都差不多，因此進行了下面兩個實驗：

(a) 只取相關性最高的幾個feature

(b) 取所有feature

最後實驗的結果為 (b) 較優，因此我們都取全部的feature來訓練。

2. Model Description

進行完資料前處理和feature選擇後，我們選定了幾種模型來做接下來的實驗。首先仿照hw2的作法，使用Logistic Regression和DNN，另外，由於我們對discrete features做了one-hot encoding，這樣的feature會比較適合decision-based的模型，因此選用

Random Forest 和 Gradient Boosting Decision Tree。

1. Simple models

包含regression-based的Logistic Regression和DNN，以及decision-based的Random Forest 和 Gradient Boosting Decision Tree。這些模型在scikit-learn中都有實作，我們都是使用scikit-learn的模型。

(a) Logistic Regression

課堂上有介紹過的模型，輸出，可以用gradient descent 來接近最佳解。

(b) DNN

多層類神經網路，最基本的DNN就是Multi-layer perceptron，也可以使用gradient descent來接近最佳解，但隨著初始化不同會有不同的走向。最後一層使用softmax當作激活函數就可以解多分類問題。

(c) Random Forest

使用decision-based的Decision Tree當作基本的小模型，利用bootstrap（取後放回的抽樣）的技巧來給予每棵樹不同的training data，再將這些分類能力比較弱的樹作bagging ensemble，得到最後的結果。

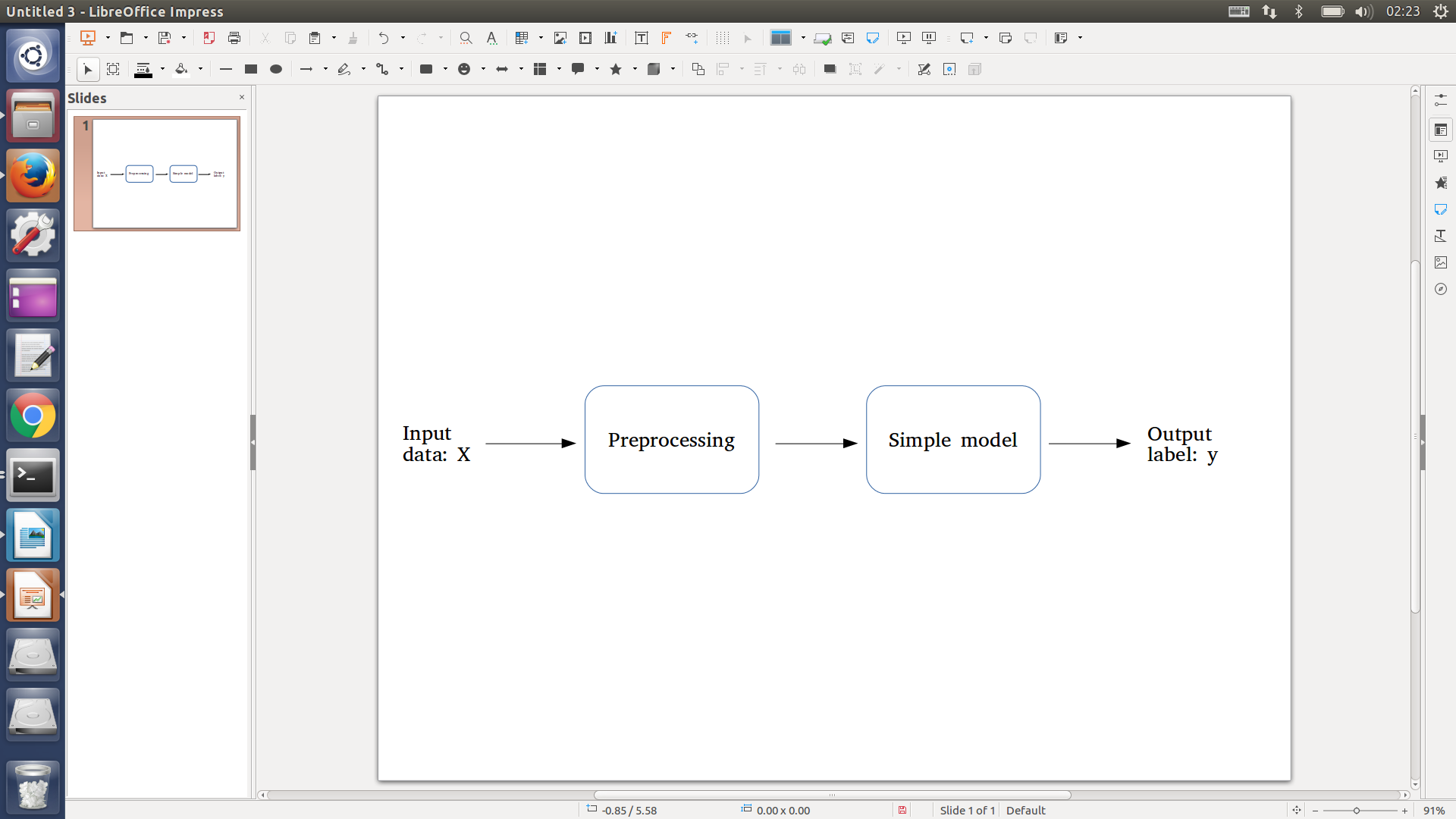
(d) Gradient Boosting Decision Tree

Boosting是一種「改變錯誤」的技巧，每次訓練完一個小模型後，將這個模型分類錯誤的data的權重調高，再進行下一次訓練，如此一來可以將錯誤調整回來。而Gradient Boosting則是使用Boosting概念的一種優化方法，和

Linear Regression的gradient descent很像，但Gradient Boosting是計算先前模型的loss function的gradient，往gradient的反方向走。

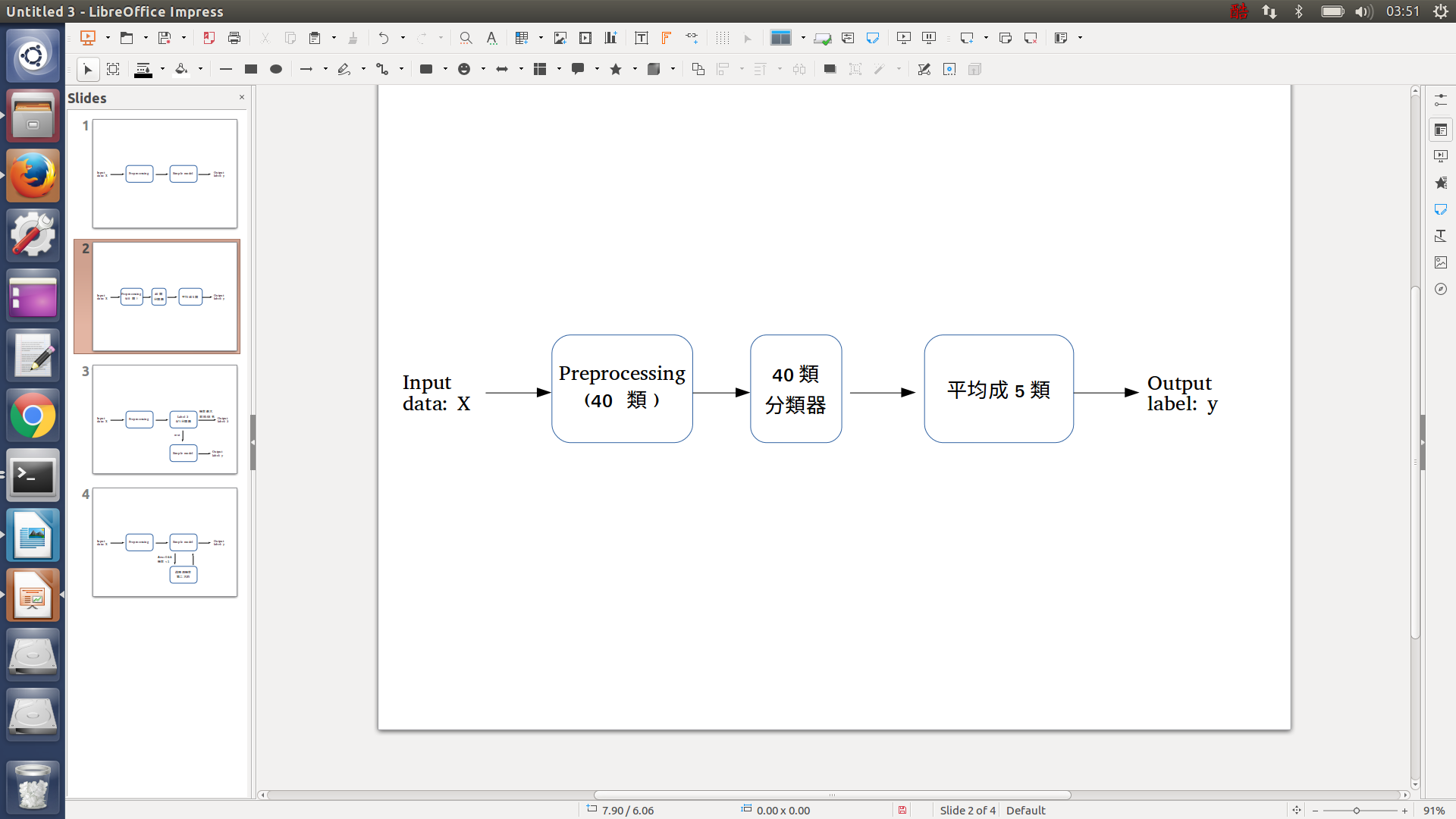
Gradient Boosting Decision Tree就是使用Decision Tree 作為base model的模型。

Simple model的整體架構非常簡單，模型示意圖如下：



1. 40 Labels classifier

由於原本的training data的label是40種，然後再對應到normal和四種攻擊，因此我們嘗試不直接把training data對應成5種分類，而是直接訓練一個40分類的分類器，再將四種攻擊對應到的分類的機率平均起來，最後輸出成5種分類，模型示意圖更改如下：

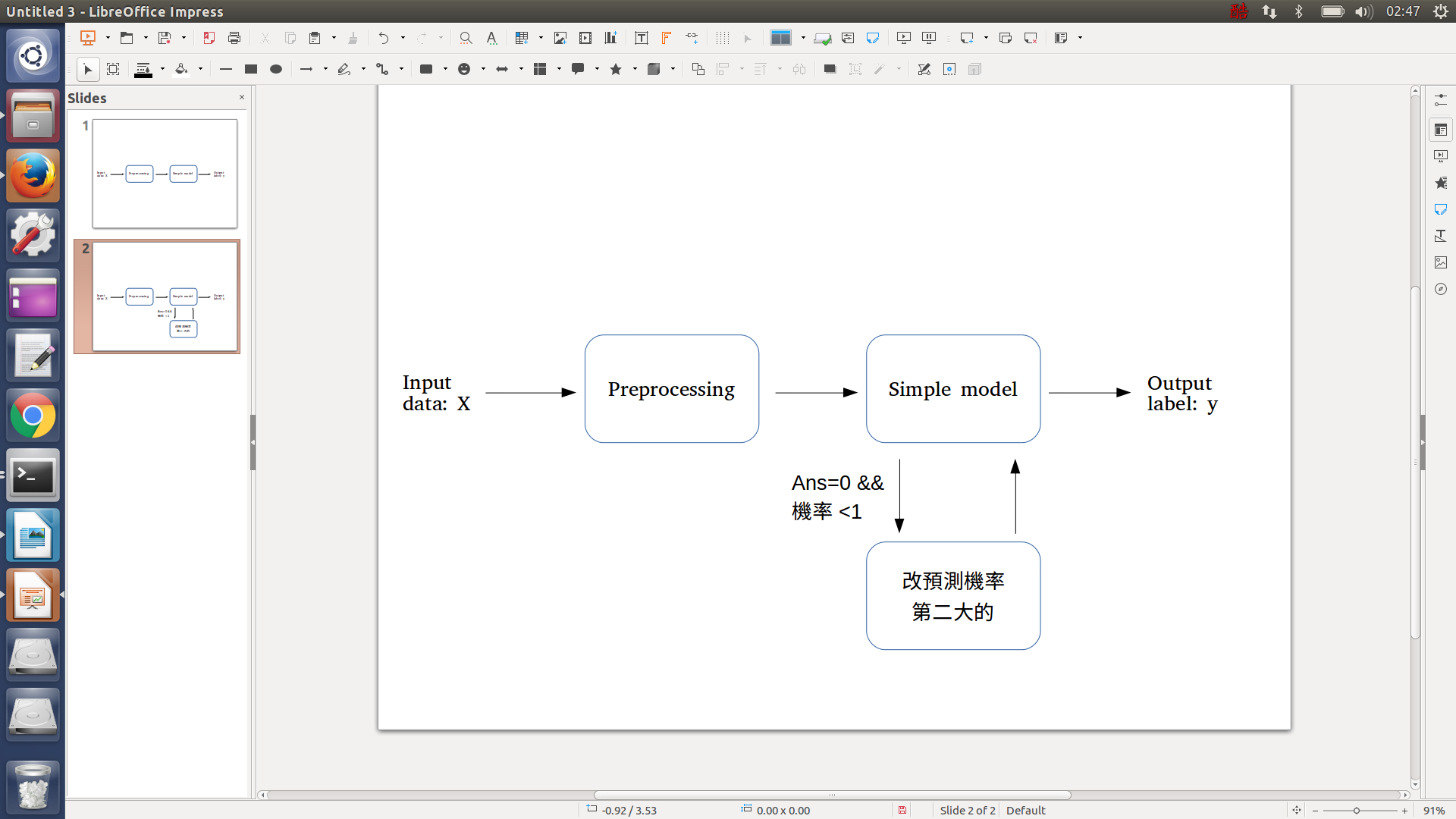


1. Altered models

由於這個task的training set 和testing set分布相當不同，因此適當的矯正模型的輸出分布可以使預測testing set的效果變好（我們可以透過kaggle知道testing set的分布）。Altered model就是在simple model的基礎上，對模型輸出的分布做調整，而我們做的調整有兩種：

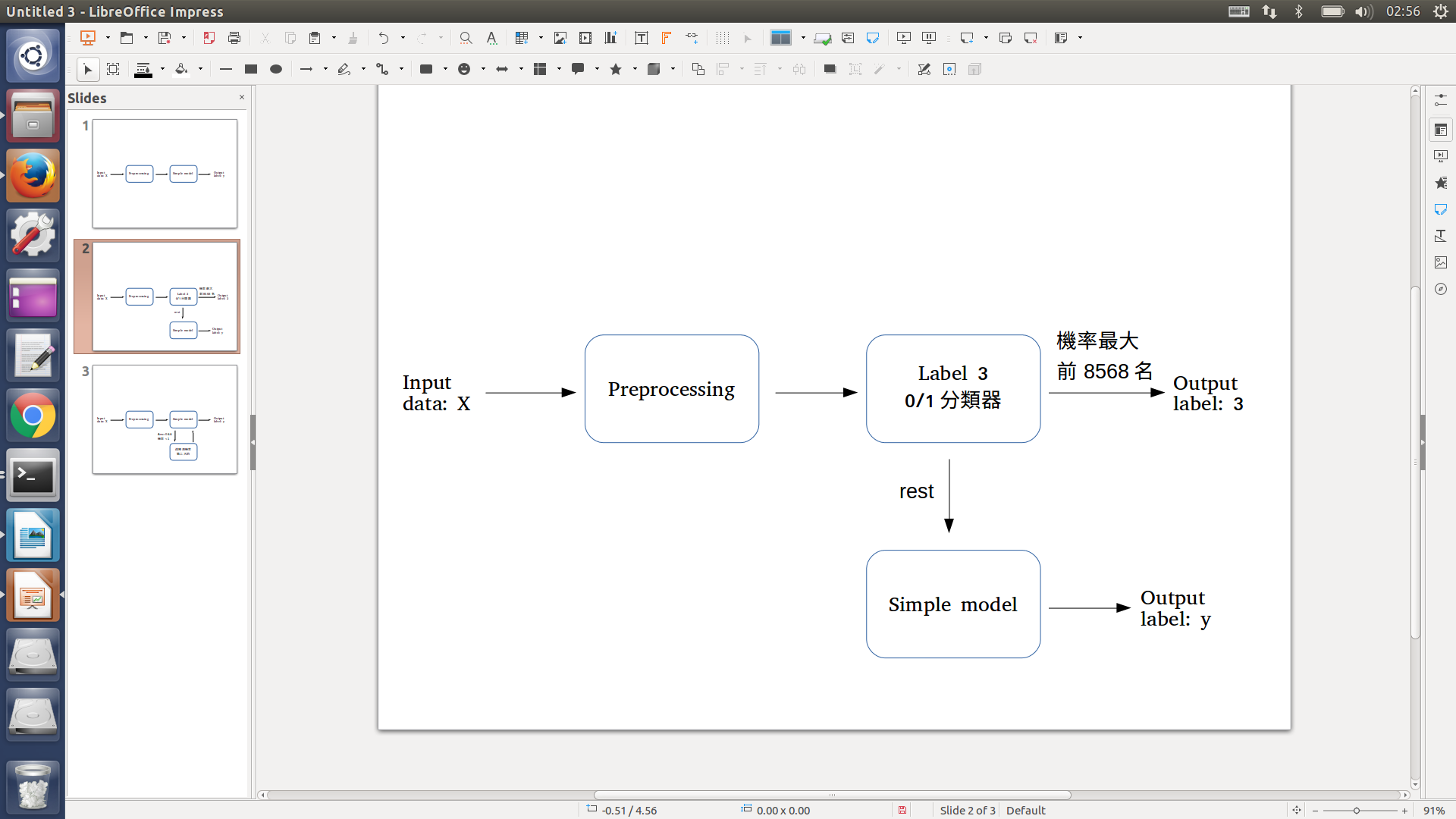
(a) 將預測為label 0 重新檢查

由於預測label 0的比例太高了，因此將label 0預測機率小於1的所有data，改為預測機率第二大的label，模型示意圖更改如下：



(b) 優先填滿分布差異較大的分類

在比較training set和testing set的分布差異時，我們發現label 3的分布差異最大，因此優先考慮label 3，訓練一個label 3的0/1分類模型，先將這個模型預測機率最大的data標記為label 3，剩下的data再給simple model預測得到剩下的label，模型示意圖如下：



3. Experiments and Discussions

1. Model比較

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | CV/OOB score | Public score | Training time |
| Simple model | Logistic Regression |  |  | >12 hours |
| DNN | 0.9997 | 0.95702 | 10 mins |
| Random Forest  n=100 | 0.99999 | 0.95752 | 3 mins / 8 thread |
| Random Forest  n=1000 | 0.99999 | 0.96128 | 30 mins / 8 thread |
| GBDT, n=100 | 0.99999 | 0.96265 | 6 hours / 1 thread |
| 40 labels | Random Forest  n=1000 | 0.99999 | 0.96038 | 1 hour / 8 thread |
| Altered model | 2.(a) |  | 0.96415 |  |
| 2.(b) |  | 0.96895 |  |

可以看到幾乎所有model的CV/OOB score都是0.999以上，這代表對於training set的分布來說，這些model都可以分類的很好。然而到了預測testing set時，分數就降到了0.96左右，被分布差異影響很大。

訓練時間的部份，Logistic Regression訓練花費太久（超過12小時），因此之後都不使用；DNN的訓練時間雖短，但性能太不穩定，因此也不使用；Random Forest和GBDT的性能差不多，但訓練時間短很多，所以之後都經常使用Random Forest來做實驗。

Altered models的部份，2(a)更改label 0 答案的模型，對於public score有微小的進步，但是不能確定這個進步是運氣好或是實際有效；2(b)強調label 3的模型，public score進步的幅度就比較明顯。

1. 分布差異

從先前的實驗中可以發現，training set和testing set的分布有著很大的差異，而testing set (public set) 的分布可以透過kaggle的分數來知道，因此我們用了4次submission來看testing set每個分類各佔了多少比例，得出來的數字如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| Training set | 19.856 | 79.281 | 0.00098 | 0.023 | 0.839 |
| Testing set | 25.966 | 71.230 | 0.038 | 1.412 | 1.354 |
| 預測結果 | 29.858 | 68.840 | 0.00066 | 0.066 | 1.235 |

在training set和testing set的分布中，差最多筆的雖然是label 0 和label 1，但以比例來說差異最大的是label 3。從預測結果來看也可以發現，label 3幾乎沒辦法正確地預測出來，而label 0則是預測錯誤太多，錯誤的4%中幾乎全部都是誤判為

label 0。

在這個分布差異下，我們想到了兩種模型來解決這個問題，也就是上面所提到的Altered models。首先2(a) 是想要將預測結果的label 0數量減少，而2(b) 是想要將預測結果的label 3增加，Altered model都是基於調整分布的想法建構的。

1. Altered models

使用Altered model來調整分布後，public score都有上升，因此我們認為Altered model是有用的，以下簡單分析兩個模型

(a) 更改label 0 答案 [2.(a)]

這個模型的public score是0.96415，而原本的simple model是使用Random Forest (n=1000)，所以原本的public score 為0.96128，大約有0.3%的進步。進步幅度雖然小，但仔細檢查後發現，更改答案後label 0大約佔26%，其中正確的有24.9%，也就是說誤判為label 0的錯誤從原本有4%，更改後剩下1%，這是很大的進步，也證明這個模型是可行的，可惜即使判斷出哪些是誤判，還是很難重新判斷是哪種攻擊，因此這個模型的進步幅度並不大。

(b) 強調label 3 [2.(b)]

這個模型的public score是0.96895，而原本的simple model是使用Random Forest (n=1000)，所以原本的public score 為0.96128，大約有0.75%的進步，這個進步幅度比起上一個模型來的明顯多了，原因我們推測是由於 label 3的分布差異太大了，原本的simple model幾乎沒有能力分類出label 3，要是能完美分類出label 3，那麼就能有1.4%的進步，因此現在0.75%的進步是可以想像的。

1. Normalization

由於training set 和testing set的連續性feature的大小不一定意義相同（時間不同可能會有差異），而且我們使用的是decision-based的模型，對於這種scale可能不同的因素很敏感，因此我們想要使用normalization來減少這種差別。

我們對training set和testing set分別做feature-wise的normalization，接著一樣使用simple model來進行訓練，最後的結果如下：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | CV/OOB score | Public score |
| Simple model | Random Forest  n=1000 | 0.99999 | 0.95028 |
| GBDT, n=100 | 0.99999 | 0.95296 |

這個結果並沒有達到我們的預期，因此實驗都沒有使用normalization

1. 40 label classifier

由於原本的training data的label是40種，然後再對應到normal和四種攻擊，因此我們直接訓練一個40分類的分類器，再將預測出來的機率作平均得到5分類的結果。這樣的model的public score為0.96038，略遜於原本的simple model。我們推測原因為這40種label其實也是非常不平均，大部分集中在其中幾個label，因此其他小label一樣難以預測，所以效果和原本的5分類問題非常接近，甚至稍微遜於原本的模型。

5. Reference

1. MIT Lincoln Lab. DARPA Intrusion Detection Evaluation <https://www.ll.mit.edu/ideval/docs/attackDB.html#pod>
2. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, Anna L. Buczak and Erhan Guven, IEEE