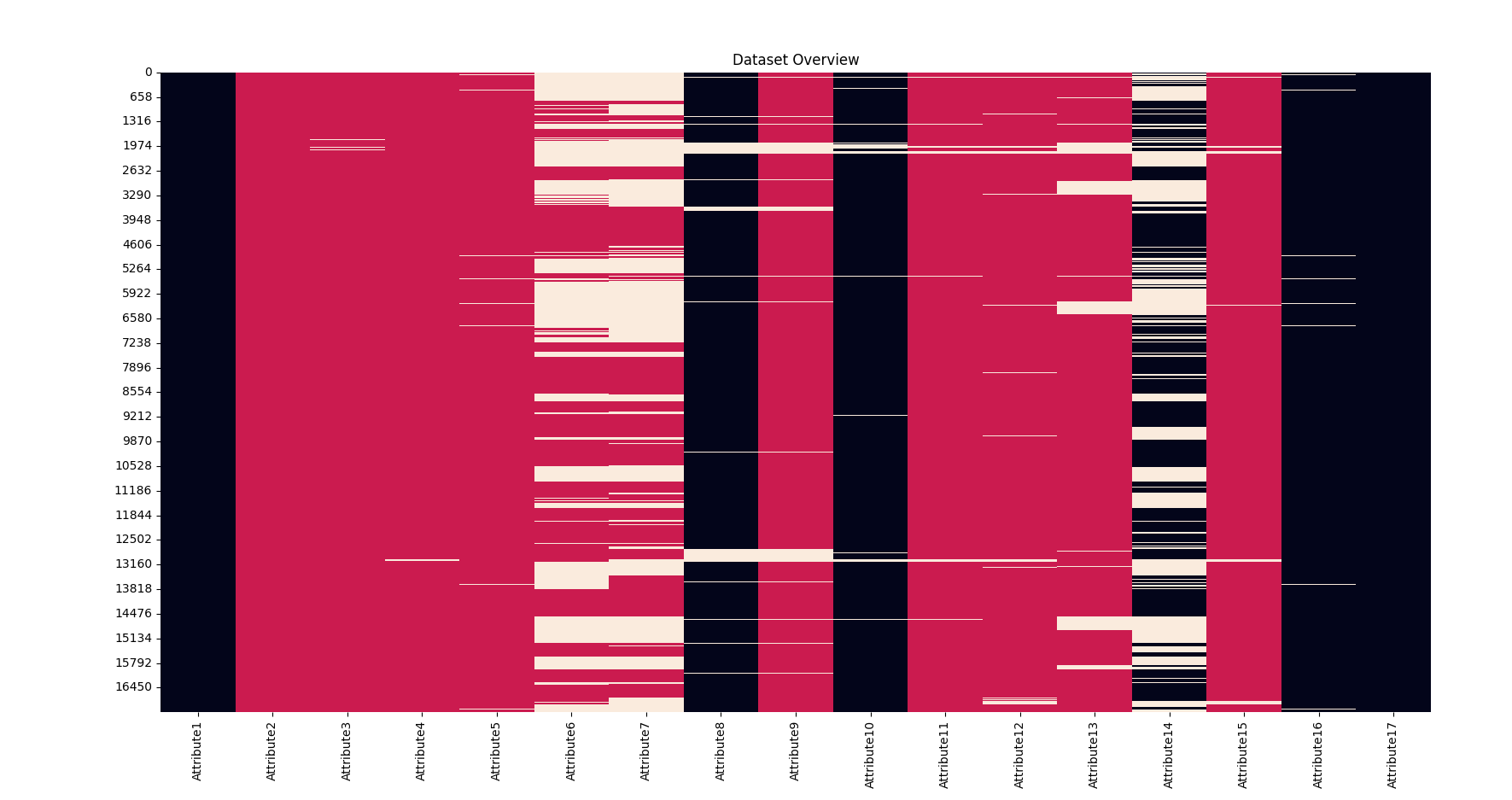
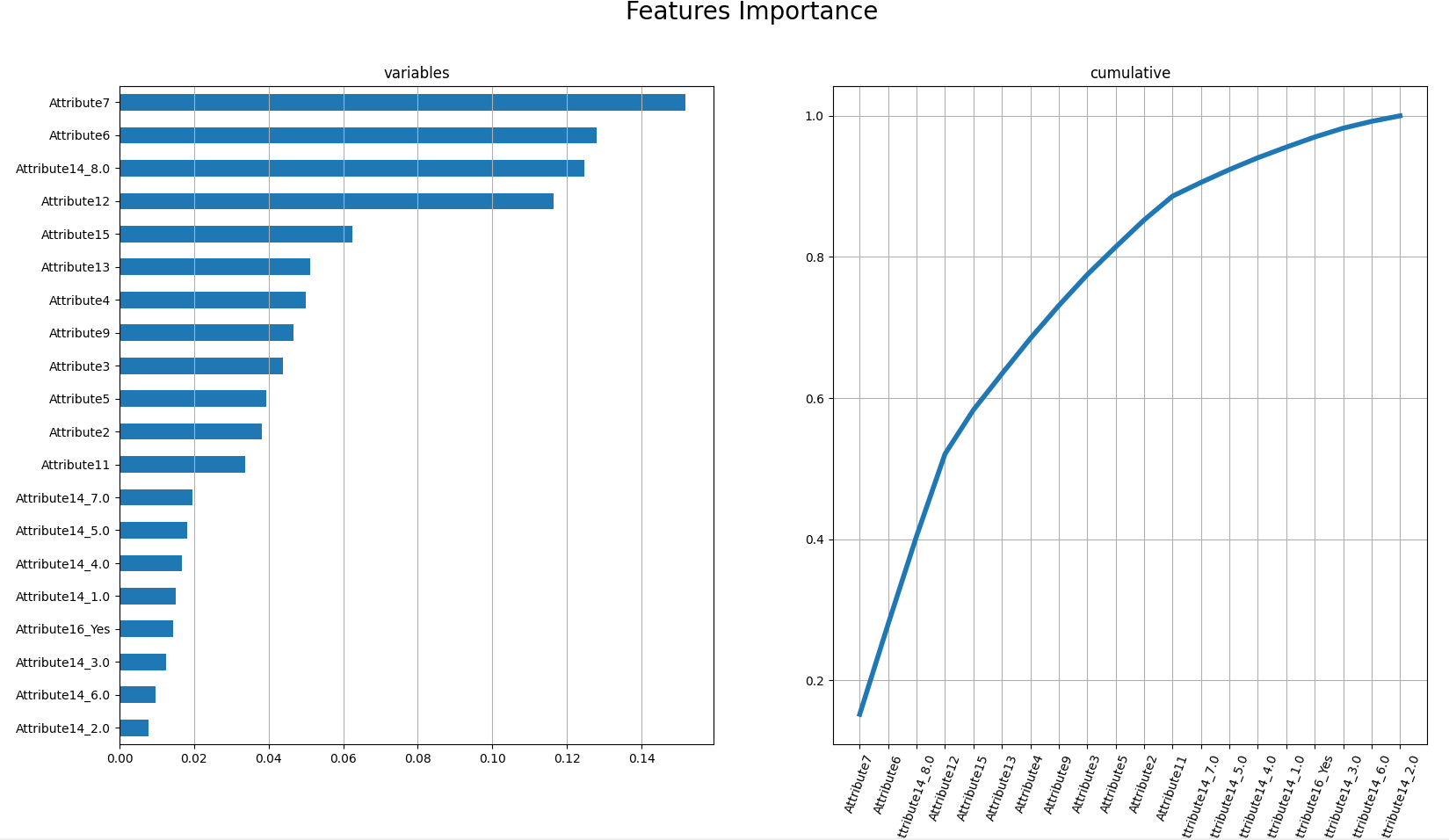
Data Science HW1 B10815044 謝鈞曜

# 觀察

觀察Attribute1的資料可以發現它就是代表資料的時間，基本上跟預測的結果不太有關係，因此我直接就剃除這個feature了，再來分析資料的情況(如圖表1)，發現有些資料是遺失的所以需要補值，後續透過sklearn分析feature importance(如圖2)發現Attribute8跟Attribute10基本上也是沒什麼用的資訊，因此也相繼剃除了，開始分析之後發現預測結果很極端TP跟FN都很高，推測是資料分布太不平均所導致。



圖表1 黑色代表category data 紅色numerical data 白色代表missing data



圖表 2 分析Feature importance

# 資料預處理

## 資料分布

首先我先把Yes跟No的資料分成兩堆，然後我希望Yes跟No的資料數量可以大概各一半，因為No的資料比較多，所以如果資料(observation)不是完整的就直接drop掉，但是Yes的資料相對少很多，因此如果有缺漏的話用K-NN來補值(用一堆Yes的資料來做K-NN也可以得到相對精準的值)，處理過後No堆的資料有5696筆，而Yes則是3139，此時資料還是不平均的狀況，所以我在訓練的模型的時候會從No堆的5696隨機抓3139筆資料，這樣就可以達到各半了。

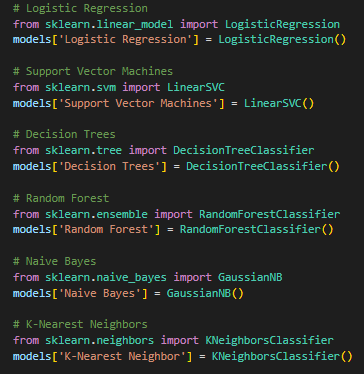
## 訓練集與測試集比例

7:3, 8:2, 9:1都嘗試過，結論是沒有很顯著的影響，可能是資料集還算夠。

## Data imputation

因為分類為Yes的資料明顯比No的少很多，用過K-nn跟mean補值，基本上訓練出來的模型效果都不佳，後來發現要先將訓練集跟驗證集分開再做補值，但結果依舊不如有空缺值的observation移除掉好，因此最後我就決定跳過此步驟了。

# 模型訓練



## 訓練模型

這幾種的模型我都試過，結果比較好的是Logistic Regression, Support Vector Machines, Random Forest

## 投票機制

如上述，同時利用三種模型去做訓練，並對投票結果進行投票，取最高票當結果，這樣做的好處是，平均的準確率會高，但單次的最佳準確率不會太高。

# 結果

基本上Accurracy基本上上在80~82%浮動，根據train data不同的分配結果也會訓練出不同的Accurracy

# 結論

以上內容是最後弄出最佳解的結論，當然沒有包含剛開始走過的坑，所以沒有像上述內容那麼順利，也沒有辦法一開始就能發現這麼多細節，全部都是自己慢慢爬文研究得出的結果，嘗試各種能讓預測變的更準的可能性，剛開始有點太糾結於模型參數的調整，但後來我發現比較重要的是Data Mining跟Data Preprocessing，主要把資料分布平均之後整個準度就上升了，因此我的測試結果直接從大概74%直接跳到80%，透過此作業也發現其實自己蠻喜歡做資料分析的，特別是資料視覺化的部分，我覺得很療育，但有時候也是蠻煩的，要一直等模型跑完，然後準確度又沒有提升，結論還是不要太糾結，有空想一下想一下，想到東西再回來做做看就可以了

此外還有發現如果Accuracy呈現爆炸高的情況，基本上就是不太對勁了，因為有高機率是overfitting = =

# 參考資料

https://towardsdatascience.com/machine-learning-with-python-classification-complete-tutorial-d2c99dc524ec