National Tsing Hua University

Institute of Statistics

IBM HR Analytics Employee Attrition and Performance

Author:
109024522 宋承恩
109024502 王浚驊
109024515 林鼎智
109024516 李泓緯
109024519 呂文翔

Supervisor:
鄭又仁 教授

A Final Project Submitted for the Statistical Learning

January 13, 2021

目錄

1 緒論及資料說明

1.1 分析目的

人員流動性在公司內部管理一直是相當大的議題，員工離職除了會造成公司隱性成本增加，也會有後續招募人員的問題，若人力資源（Human Resource, HR）部門能提前找出潛在離職（Attrition）的員工，可以針對這些族群進行談話，找出導致有離職念頭的原因，加以改善；或提前招募相關崗位人才，以利後續工作接洽順利進行。因此，我們這次專案想透過每個員工在公司內部的資料，建立模型以嘗試預測該名員工的離職傾向，提供給人力資源部門作為參考依據。

1.2 資料說明

本專案採用 Kaggle 的 IBM HR Analytics Employee Attrition Performance 中的資料[1]。此資料共有1470 筆資料、28 個變數，其中一個變數為EmployeeNumber 為每筆觀測值的員工編碼，因此我們將此作為辨別員工的變數；反應變數（response variables）為離職與否（Attrition），解釋變數（predictors）則為如表 1 所示

表 1: 變數說明

Code	Definition and source
Attrition（response）	0 = No, 1 = Yes
BusinessTravel	Non-Travel, Travel-Frequently, Travel-Rarely
DistanceFromHome	Distance from home
Education	1 = Bellow College, 2 = College, 3 = Bachelor, 4 = Master, 5 = Doctor
EducationField	Human Resource, Life Sciences, Marketing, Medical, Other Technical Degree
EnvironmentSatisfaction	1 = Low, 2 = Medium, 3 = High, 4 = Very High
Gender	0 = Female, 1 = Male
JobInvolvement	1 = Low, 2 = Medium, 3 = High, 4 = Very High

[1] https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset

Code	Definition and source
Joblevel	1 = Very Low, 2 = Low, 3 = Medium, 4 = High, 5 = Very High
JobRole	Healthcare Representative, Human Resources, Laboratory Technician, Manger, Manufacturing Director, Research Director, Research Scientist, Sale Executive, Sales Representative
JobSatisfaction	1 = Low, 2 = edium, 3 = High, 4 = Very High
MaritalStatus	Divorced, Married, Single
MonthlyIncome	Monthly income
NumCompaniesWorked	The number of companies the employee have worked
OverTime	0 = No, 1 = Yes
PercentSalaryHike	Income markup
PerformanceRating	1 = Low, 2 = Good, 3 = Excellent, 4 = Outstanding
RelationshipSatisfaction	1 = Low, 2 = Medium, 3 = High, 4 = Very High
TotalWorkingYears	Total job tenure（year）
TrainingTimesLastYear	Training times
WorkLifeBalance	1 = Bad, 2 = Good, 3 = Better, 4 = Best
YearsAtCompany	Total job tenure in this company（year）
YearsInCurrentRole	Years in the current role（position）
YearsSinceLastPromotion	Years since the last promotion
YearsWithCurrManager	Years with current manager

1.3 分析流程

本次專案的分析流程如下，先對進行資料探索式分析（Explorator Data Analysis, EDA），再來採用五個模型解決二元預測問題（Binary Classification Problem），分別為 Logistic Regression、Naive Bayes、Classification Tree、Random Forest、gradient boosting decision tree，最後提供分類問題的相關指標如準確率（Accuracy）、特異度（Specificity）、敏感度（Sensitivity）、接收者操作特征曲線（Receiver Operating Characteristic Curve, ROC）以及 AUC（Area Under the Curve）作為最終模型選擇的依據。

本專案使用 R 語言作為分析工具，模型則主要使用 caret 套件來實現。

2 探索性資料分析

2.1 連續型變數

對 Age、MonthlyIncome、DistanceFromHome、NumCompaniesWorked 繪製直方圖以及分別和反應變數 Attrition（離職）做箱型圖，如圖 1 所示

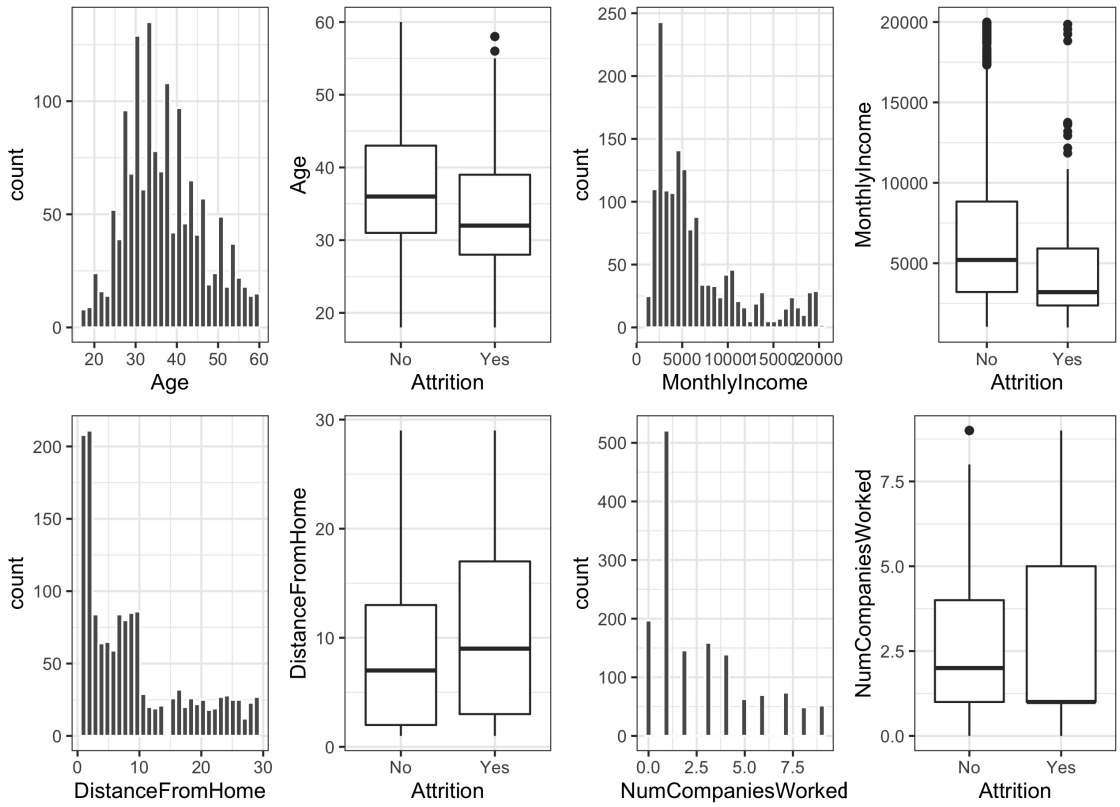


Figure 1: 連續型變數之視覺化圖表

- 由年齡的直方圖可以發現，公司員工的年齡多數為 30∼40 歲，年齡對離職的箱型圖我們可以發現，不離職的員工主要是年紀較高的員工，我們猜想因為一般公司會主要錄用年輕人，所以年紀較大的員工會較不敢隨意離職。
- 由月收入的直方圖，很明顯看到一間公司的高薪會集中在少數員工，並從箱型圖可以看出相對高薪的員工較不易離職，這符合一般公司會有的情形。

- 從公司與家的距離的直方圖可以看出普遍大家的住家不會離公司太遠，而從箱型圖可以看出公司與家距離較遠的員工比較會有離職的情況。我們猜想可能是公司離家太遠，導致每天需要花一定的通勤時間，間接造成員工會有想離職的原因之一。
- 從待過幾間公司的箱型圖則看出待過較多公司的員工比較會有離職的情況，會合理懷疑符合此類型員工普遍會有輕意離職的想法。

對 PercentSalaryHike、TrainingTimesLastYear、TotalWorkingYears、YearsAtCompany 繪製直方圖以及分別和反應變數 Attrition（離職）做箱型圖，如圖 2 所示

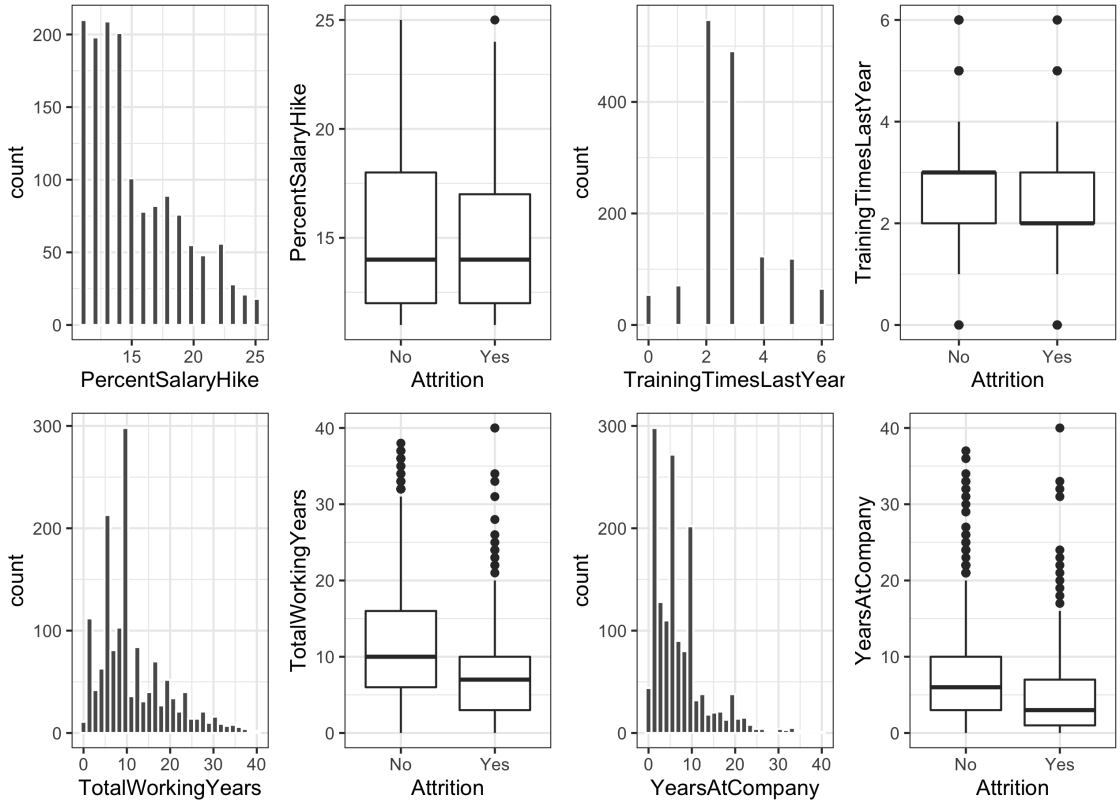


Figure 2: 連續型變數之視覺化圖表

- PercentSalaryHike 的箱型圖可以看出薪水漲幅較高的員工比較不會有離職的情況，而從直方圖可以看出公司員工的薪水漲幅是低的。
- 透過 TrainingTimesLastYear 可以看出去年員工受訓時間大多數為2-3週。透過箱型圖我們會覺得員工去年受訓的時間似乎與離職沒有明顯的關係。

- 由 TotalWorkingYears 直方圖可以看出此變數散佈呈現右尾分佈。透過 TotalWorkingYears 的箱型圖可以發現工作總年資較長的員工普遍較不會有離職的情況。
- 由 YearsAtCompany 直方圖可以看出此變數散佈呈現右尾分佈。YearAtCompany 的箱型圖可以看到在此公司的年資較長員工普遍較不會離職。

對 YearsInCurrentRole、YearsWithCurrManager、YearSinceLastPromotion 繪製直方圖以及分別和反應變數 Attrition（離職）做箱型圖，如圖 3 所示

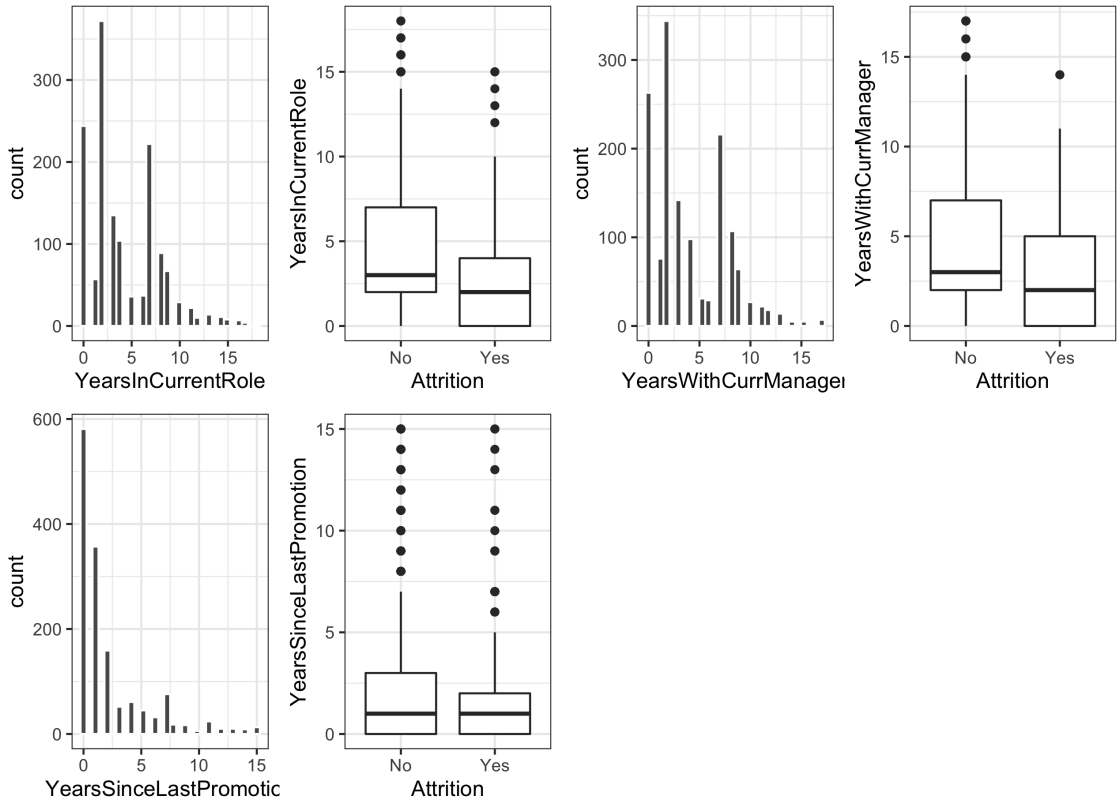


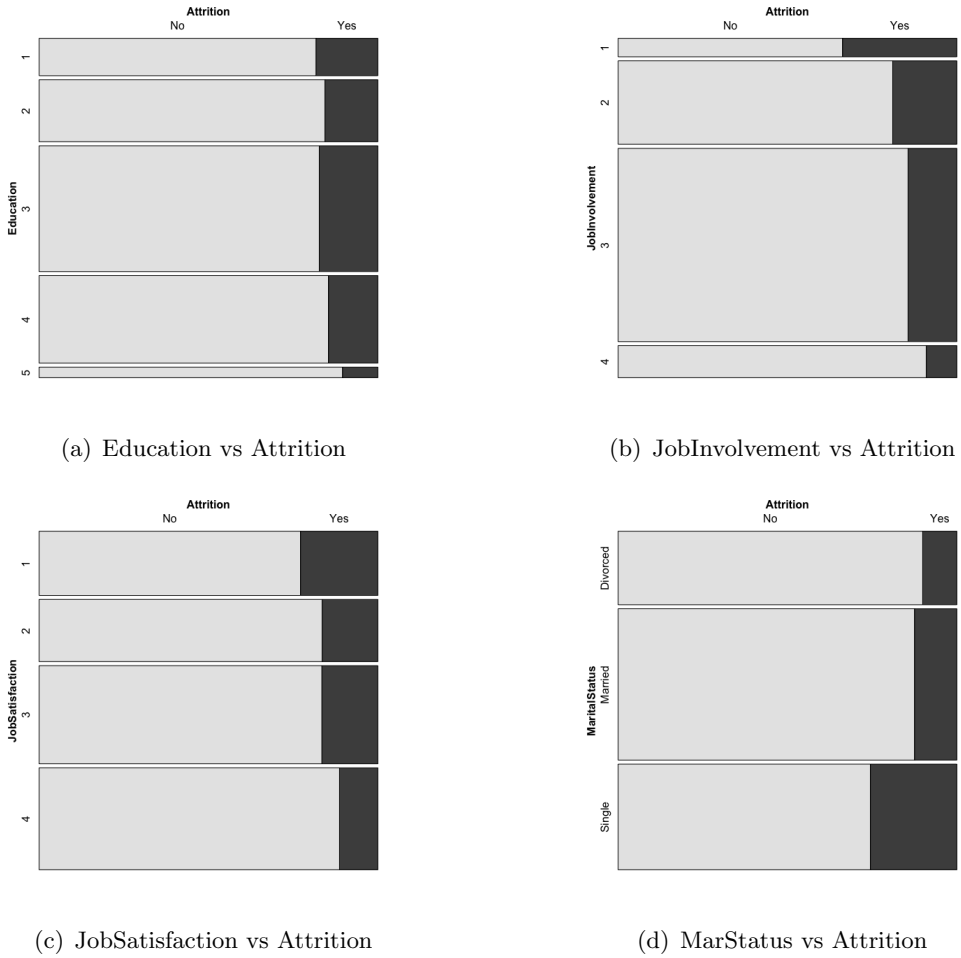
Figure 3: 連續型變數之視覺化圖表

- 由 YearsInCurrentRole 直方圖可以看出此變數散佈呈現右尾分佈。從箱型圖可以發現，現任職務年資越長，越不會離職。可能是因為在公司工作一段時間，普遍大家不會選擇離職。
- 由 YearsWithCurrManager 直方圖可以看出此變數散佈呈現右尾分佈。現任職務工作年份越長，越不容易離職。

- 由 YearSinceLastPromotion 直方圖可以看出此變數散佈呈現右尾分佈。距離上次升遷的時間越長，越不會離職。

透過上方資料探索的發現，我們會推斷 Age、MonthlyIncome、DistanceFrome、NumCompaniesWorked、PercentSalaryHike、TotalWorkingYears、YearsAtCompany、YearsInCurrentRole、YearsWithCurrManager、YearSinceLastPromotion 這些變數對反應變數 Attrition 是有明顯關聯的。

2.2 類別型變數



(a) Education vs Attrition

(b) JobInvolvement vs Attrition

(c) JobSatisfaction vs Attrition

(d) MarStatus vs Attrition

Figure 4: 離散型變數之視覺化圖表

- 由圖 4(a) 發現，不同的教育程度與有無離職的比例並無明顯差異。

- 由圖 4(b) 發現，不同的工作投入度與有無離職的比例有明顯的差異。越投入工作，越不容易離職；對工作的投入度越低，越可能離職。比較合理的解釋是，對工作的投入度越高的員工，表示對這份工作的依賴性越高，越不容易離職；相反的，工作投入度越低，可能是因為對工作的歸屬感較低，所以較容易離職。

- 由圖 4(c) 發現，工作滿意度越高，離職的比例越少。這是很直覺的現象，舉例來說，薪資比較高的人相對薪資較低的人，工作滿意度會比較高，而薪資較高的人也比薪資較低的人不易離職。

- 由圖 4(d) 發現，未婚的人比有（過）婚姻的人容易離職。可能的原因是，未婚的人相對有（過）婚姻的人自由，需要考量的面向也較少，離職造成的負擔也較低，所以離職的比例也比較高。

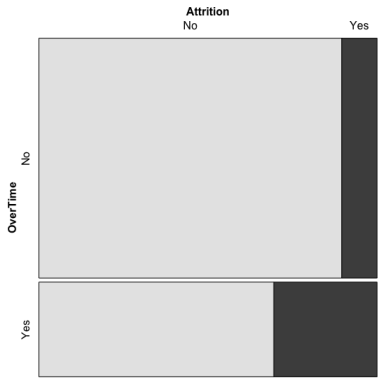


Figure 5: 離散型變數之視覺化圖表

- 由圖 5 發現，超時工作的員工，離職的比例較高。可能的原因是，超時工作的員工壓力會比正常工時的員工大，所以離職的比例會較高。

綜合上述結果，我們預測不同背景與經歷的人與一些個人的工作指標，會影響該名員工的離職傾向。

3 模型方法

3.1 Logistic Regression Model

3.1.1 模型設定

定義羅吉斯函數（logistic function），其為一種 sigmoid function

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}; \quad t \in R^1 \tag{3.1}$$

其可將任意一維的實數 t 映射到 $[0, 1]$，其亦是有界、可微的實數函數，包含一個反曲點，在數學上有優良性質。接著建構模型，給定 $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$，假設

$$y_i \overset{\text{i.i.d.}}{\sim} \text{Bin}(1, P_{i,0}) \tag{3.2}$$

$$P_{i,0} = \sigma(\boldsymbol{x}_i'\beta_0) = \frac{e^{\boldsymbol{x}_i'\beta_0}}{1 + e^{\boldsymbol{x}_i'\beta_0}}, \quad i = 1, ..., n \tag{3.3}$$

（3.2）式的精神如下所述，在二元分類問題，我們想了解解釋變數 $\boldsymbol{x}_i = \{x_{i1}, ..., x_{ip}\}$ 如何線性影響反應變數 y_i，故將線性組合 $\boldsymbol{x}_i\beta_0$ 透過 logistic function 對 y_i 建立參數化模型。則

$$\eta(P_{i,0}) = \sigma^{-1}(P_{i,0}) = \text{logit}(P_{i,0}) = \ln(\frac{P_{i,0}}{1 - P_{i,0}}) = x_i'\beta_0, \tag{3.4}$$

故

$$y_i = \mathbb{E}(y_i) + y_i - \mathbb{E}(y_i) = P_{i,0} + \epsilon_i, \quad \text{where } \epsilon_i \sim (0, \quad P_{i,0}(1 - P_{i,0})) \tag{3.5}$$

其中 y_i 為反應變數、$\boldsymbol{x}_i$ 為 p 維的解釋變數、β_0 為 p 維的迴歸係數、η 為 logit 連結函數 （link function），$P_{i,0}$ 為真實參數值。

3.1.2 估計方法

在上述模型中，假設 $y_i \sim \text{Bin}(1, P_{i,0})$，故可以使用最大概似估計法（Maximum Likelihood Estimation, MLE）估計迴歸係數 β_0。Logistic regression model 的 log-likelihood

function 為

$$\begin{aligned}\ell(\beta) = \ln \mathcal{L}(\beta) &= \sum_{i=1}^{n} \ln \{P_i^{y_i}(1-P_i)^{1-y_i}\} = \sum_{i=1}^{n} y_i \ln (\frac{P_i}{1-P_i}) + \sum_{i=1}^{n} \ln (1-P_i) \\ &= \sum_{i=1}^{n} y_i x_i' \beta - \sum_{i=1}^{n} \ln \{1 + e^{x_i'\beta}\}\end{aligned} \tag{3.6}$$

但因為不像誤差為常態假設的傳統迴歸模型，MLE 有封閉解，Logistic regression model 必需依賴迭代法對（3.5）求最佳解。可以使用牛頓法（Newton's method）或者梯度上升法（Gradient descent method）解以下的最佳化問題

$$\hat{\beta} := \arg\max_{\beta \in \mathcal{B}} \ell(\beta) \tag{3.7}$$

最後若使用 0.5 作為決策界線（threshold），則

$$\hat{y}_i = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \hat{\beta} > 0.5 \\ 0, & \text{otherwise} \end{cases} \qquad i = 1, 2, ..., n \tag{3.8}$$

3.2 Naive Bayes Classifier

3.2.1 模型設定

樸素貝葉斯分類器（Naive Bayes Classifier）為一個條件機率模型（conditional probability model），亦即給定有 K 種類別的分類標的 y_k 及 $\boldsymbol{x} = \{x_1, ..., x_p\}$，它會計算

$$P(y_k | x_1, ..., x_p), \quad k = 1, ..., K \tag{3.9}$$

但當 p 很大時，計算式 （3.9）很難做到，故可以先利用貝式定理（Bayes theorem）

$$P(y_k | x_1, ..., x_p) = \frac{P(y_k, x_1, ..., x_p)}{P(x_1, ..., x_p)} \tag{3.10}$$

再利用條件機率的定義將結合機率密度函數拆解

$$\begin{aligned} P(y_k, x_1, ..., x_p) &= P(x_1, ..., x_p, y_k) \\ &= P(x_1|x_2, ..., x_p, y_k)P(x_2, ..., x_p, y_k) \\ &= P(x_1|x_2, ..., x_p, y_k)P(x_2|x_3..., x_p, y_k)P(x_3, ..., x_p, y_k) \\ &... \\ &= P(x_1|x_2, ..., x_p, y_k)P(x_2|x_3..., x_p, y_k)...P(x_{p-1}|x_p, y_k)P(x_p|y_k)P(y_k) \end{aligned} \tag{3.11}$$

接著做條件獨立（conditional independence）的假設，假設所有的解釋變數在給定反應變數下為相互獨立（mutual independent），意即

$$P(x_i|x_{i+1}, ..., x_p, y_k) = P(x_i|y_k) \tag{3.12}$$

故式（3.10）變成

$$\begin{aligned} P(y_k|x_1, ..., x_p) &\propto P(x_1, ..., x_p, y_k) \\ &\propto P(y_k)P(x_1|y_k)...P(x_p|y_k) \\ &\propto P(y_k)\prod_{j=1}^{p} P(x_j|y_k) \end{aligned} \tag{3.13}$$

最後，樸素貝葉斯分類器為

$$\hat{y} = \underset{k\in\{k=1,...,K\}}{\arg\max}\ P(y_k|x_1, ..., x_p) = \underset{k\in\{k=1,...,K\}}{\arg\max}\ P(y_k)\prod_{j=1}^{p} P(x_j|y_k) \tag{3.14}$$

3.2.2 模型估計

Naive Bayes Classifier 的估計可以分為兩個部分，第一部分為先驗機率（Prior probability）$P(y_k)$ 的估計

$$\hat{P}(y_k) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \in k) \tag{3.15}$$

其中 n 為總資料個數。第二部分為 $P(x_j|y_k)$ 的估計，根據解釋變數 $\boldsymbol{x}$ 的類型分為兩種解

法。若解釋變數 $\boldsymbol{x}$ 為類別型變數，則假設給定 y_k 下，其服從多項式分配（Multinomial distribution），稱為多項式樸素貝葉斯分類器（Multinomial Naive Bayes Classifier），其 likelihood 函數為

$$\mathcal{L}(\boldsymbol{p}_k) = P(\boldsymbol{x}|y_k) = \frac{(\sum_j x_j)!}{\prod_j x_j} \prod_j p_{kj}^{x_j}, \quad \text{where } p_{kj} = P(x_j|y_k) \tag{3.16}$$

又 log-likelihood 函數為

$$\ell(\boldsymbol{p}_k) = \ln \mathcal{L}(\boldsymbol{p}_k) \propto \ln (P(y_k)) + \sum_j x_i \ln p_{kj} = b + \boldsymbol{w}_k' \boldsymbol{x} \tag{3.17}$$

故估計值

$$\hat{\boldsymbol{p}_k} = \arg\max_{\boldsymbol{p}_k} \ \ell(\boldsymbol{p}_k) \tag{3.18}$$

但在解式（3.18）時，若給定 y_k，但解釋變數並沒有同時出現在現有資料或訓練集（training data），會使估計值為 0，這會造成其他資訊消失的問題，故通常在估計 Multinomial Naive Bayes Classifier 時會加一個修正項，其名為 Pseudocount，它能避免機率估計為 0，而導致相關問題，這種正則化方法（Regularization）也叫做拉普拉斯平滑（Laplace smoothing），這在訓練模型時需要透過交叉驗證（Cross-Validation）選取適合的平滑參數。

若解釋變數 $\boldsymbol{x}$ 為連續型變數，假設 $x|y_k \sim N(\mu_k, \sigma_k^2)$，則一樣透過最大概似估計法，估計常態參數進而估計機率 $P(x_j|y_k)$。

實務上若遇到混合型資料（Mixed-type predictors），亦即存在連續型資料及類別型資料，典型方法是轉換連續型變數為類別型變數，意即利用直方圖的方式切割連續型變數，那平滑程度則用上方的 Laplace smoother 所決定。

3.3 Classification Tree

決策樹（Classification Tree）是運用 Classification And Regression Trees（CART）演算法建構而成，其採用二分遞迴分割的技術將分析資料的樣本切割成兩個子樣本，使生成的決策樹每一個非葉節點只會有兩個分支，所以此演算法所生成的 Classification Tree 會

是結構為二叉樹（Binary Tree），以下對二元分類樹問題 $y \in \{1, 2\}$ 建立模型。若給定切割區域 R_t

$$\hat{p}(k|t) = \frac{1}{n_t} \sum_{x_i \in R_t} I(y_i = k) \tag{3.19}$$

其中 $t = 1, ..., T$ 為節點（node）、n_t 代表第 t 個節點中觀察到的資料個數。

令 t 節點上的分類器為

$$\kappa(t) = \arg\max_k \ \hat{p}(k|t); \quad t = 1, ..., T \tag{3.20}$$

以及三種不純度函數（Impurity function）$Q_t(\tilde{T})$，其旨在利用每個節點中類別的分佈測量該節點的純度 (purity)，其中 $|\tilde{T}|$ 為樹的個數

- Missclassfication rate

$$1 - \hat{p}(\kappa(t)|t) = \frac{1}{n_t} \sum_{x_i \in R_t} I(y_i \neq \kappa(t)) \tag{3.21}$$

- Entropy:

$$-\sum_{j=1}^{K} p(j|t) \, log \ p(j|t) \tag{3.22}$$

- Gini index:

$$\sum_{j=1}^{K} p(j|t)(1 - p(j|t)) = 1 - \sum_{j=1}^{K} p(j|t)^2 \tag{3.23}$$

模型透過演算法生成分類樹時，會參照上方 Impurity measure 作為是否節點分裂的指標。其中，分錯率（Misclassification rate）越低越好，Gini index 判別此節點中樣本的純度，其越高表示此群樣本越不純，因此會希望 Gini index 越低越好。熵（Entropy）是在當分類樹在生成某節點時，衡量此生成的節點是否會增加我們分群的不純度，當 Entropy 值越高，表示分完群後不純度相較原先未分群的不純度高，則不會生成此節點，因此在分類上會希望 Entropy 指標越低越好。

用 Impurity measure 切割資料前，先定義分類樹 T 的 Information rate $R(T)$:

$$R(T) = \sum_{t=1}^{\tilde{T}} R(t) \tag{3.24}$$

- $\tilde{T}$ 為 T 樹的終端節點（terminal nodes）的個數。
- $R(t) = r(t)p(t)$ and $r(t) = i - \underset{j}{max}\ p(j|t) = 1 - p(\kappa(t)|t)$
- $r(t)$ 可以為 $misclassification\ \ rate$、$Entropy$、$Gini\ \ index$在第 t 個 node 的值。
- $p(t)$ 是資料落在 t 節點的機率，$\hat{p}(t) = \frac{n_t}{n}$

則確定決策樹此節點要切割須滿足此下條件，若 $\phi(p_1, ..., p_K)$ 是凹函數且

$$R(t) \geq R(t_L) + R(t_R) \tag{3.25}$$

也就是說，若我們繼續切割第 t 個 node，會降低不純度。

最後，當決策樹時長到一定程度時，複雜度太高則分類的準確度會下降。因此在建立決策樹太複雜時會進行剪枝的行為，而判斷是否剪枝的行為在 CART 演算法會參照 cp 指標。

- 定義 error-complexity measure $R_\alpha(T)$ ：

$$R_\alpha T = R(T) + \alpha|\tilde{T}|\ ;\ \ where\ \ \alpha = cp \tag{3.26}$$

其中，$|\tilde{T}|$ 表示 T 的終端節點集合，透過 error-complexity 來抑制決策樹不會複雜度太高。

模型參數調控如表 2:

表 2: 參數說明

模型參數	
cp	complexity parameter CART 演算法當增加一個節點後其分類的純度變化量小於樹複雜度變化 cp 倍時，則此節點必須減去。

3.4 Random Forest

Random Forest 是一種 Ensemble Learning 的演算法，基本原理是每次用類似 Bootstrap Aggregation 的方法產生新的 Decision Tree，再將多棵不同的 Decision Tree 結合。

Bootstrap 是隨機抽取原有的樣本，抽取過程是採抽後放回。Bagging（Bootstrap Aggregation）用 Bootstrap 產生的多組樣本建多棵 Decision Tree，每棵 Decision Tree 最後用 Majority Vote 得到最終結果。

透過 Bagging 結合不同 Decision Tree，不僅能夠保持 Decision Tree 的差異性，還能減少 fully-grown 造成的 overfitting。Random Forest 除了利用 Bootstrap 抽後放回的隨機抽取外，連變數也是採取隨機抽取的方式，因此也讓 Random Forest 具有高度多樣性。

$$Model : \hat{f}^{rf}(x) = \underset{k=1,\ldots,K}{argmax} \sum_{b=1}^{B} I\{\hat{f}^{tree,b}(x) = k\} \tag{3.27}$$

樣本利用 Bootstrap 抽取後，會有一部分資料沒被取用，而 Random Forest 將這些沒被抽取到的資料當作類似測試集，稱為 Out-of-Bag Data，並利用這筆資料計算 Out-of-Bag Error，再利用 Out-of-Bag Error 選取模型最適合的參數。

參數調控：用 grid search 的方法去調整參數，針對每個參數組合都配適出一個模型，然後從中挑選表現最佳的模型。主要調控的參數有：mtry、nodesize、sampsize。

表 3: 參數說明

模型參數	
ntree	Decision Tree 的個數。足夠的 Decision Tree 可以穩定模型誤差，[2]。
mtry	每次切割變數時，選擇要隨機抽樣的變數個數。
nodesize	指定 Decision Tree 節點的最小個數。
samplesize	每棵 Decision Tree 的樣本隨機抽樣比例。

3.5 XGBOOST

所謂提升算法（Boosting）就是一種將許多弱分類器集合起來變成一個較强大的强分類器，極限梯度提升（eXtreme Gradient Boosting, XGboost）則是一種梯度提升樹（Gradient Boosted Tree, GBDT），它會在每一次保留原來的模型，並且加入新的函數至模型中，來修正上一棵樹的錯誤，去提升模型

首先我們先假設 XGboost 的模型有 M 棵樹，以下為其模型：

$$\hat{f}(x_i) = \sum_{m=1}^{M} h_m(x_i) = \hat{f}^{(M-i)}(x) + h_M(x) \tag{3.28}$$

其中 M 為其迭代的次數，h_m 則為每次迭代後新增的弱分類器，$m = 1, 2, ..., M$

我們需要藉由極小化我們的目標函數（objective function）

$$\sum_{i=1}^{n} L_n(y_i, \hat{f}(x_i)) + \sum_{m=1}^{M} \Omega(h_m) \tag{3.29}$$

來找到估計函數，其中 $\Omega(h_m)$ 為限制函數（constraint function）

我們使用羅吉斯損失函數（Logistic loss function）

$$L(y, f(x)) = y \ln(1 + e^{-f(x)}) + (1 - y) \ln\{1 + e^{f(x)}\} \tag{3.30}$$

做了 t 次迭代之後，其 objective function 會變成

$$Obj^{(t)} = \sum_{i=1}^{n} L_n(y_i, \hat{f}^{(t-1)}(x_i) + h_t(x_i)) + \Omega(h_t) + constant \tag{3.31}$$

因為（3.31）看來還是相當複雜，於是以下去對 objective function 做泰勒展開（Taylor expansion）

$$\begin{aligned} & L_n(y, \hat{f}^{(t-1)}(x) + h_t(x)) \\ \approx & L_n(y, \hat{f}^{(t-1)}(x)) + L'_n(y, \hat{f}^{(t-1)}(x))h_t(x) + \frac{1}{2}L''_n(y, \hat{f}^{(t-1)}(x))h_t^2(x) \end{aligned} \tag{3.32}$$

接者我們定義

$$g_i = \partial_{\hat{f}^{(t-1)}} L(y_i, \hat{f}^{(t-1)}(x_i)) \tag{3.33}$$

$$s_i = \partial^2_{\hat{f}^{(t-1)}} L(y_i, \hat{f}^{(t-1)}(x_i)) \tag{3.34}$$

因為 $L_n(y_i, \hat{f}^{(t-1)}(x_i))$ 為已知常數項，所以 objective function 會變成

$$\begin{aligned} Obj^{(t)} &\approx \sum_{i=1}^{n} L_n \left[y_i, \hat{f}^{(t-1)}(x_i) + g_i h_t(x_i) + \frac{1}{2} s_i h_t^2(x_i) \right] + \Omega(h_t) + constant \\ &= \sum_{i=1}^{n} \left[g_i h_t(x_i) + \frac{1}{2} s_i h_t^2(x_i) \right] + \Omega(h_t) + constant \end{aligned} \tag{3.35}$$

我們將樹的定義與前面結合，去近似 $h_t(x)$

$$h_t(x) \approx \sum_{j=1}^{T} w_j I(x \in R_{jt}) \tag{3.36}$$

其中 T 為葉子的數量，也就是樹的大小並且定義其複雜度為

$$\Omega(h_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{3.37}$$

Objection function 又可以重新改寫成

$$\begin{aligned} Obj^{(t)} &\approx \sum_{i=1}^{n} \left[g_i h_t(x_i) + \frac{1}{2} s_i h_t^2(x_i) \right] + \Omega(h_t) \\ &= \sum_{i=1}^{n} \left[g_i \sum_{j=i}^{T} w_j I(x \in R_{jt}) + \frac{1}{2} s_i \sum_{j=i}^{T} w_j^2 I(x_{jt}) \right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[\left(\sum_{x_i \in R_{jt}} g_i \right) w_j + \frac{1}{2} \left(\sum_{x_i \in R_{jt}} s_i + \lambda \right) w_j^2 \right] + \gamma T \end{aligned} \tag{3.38}$$

定義 $G_j = \sum_{i \in R_{jt}} g_i$，$S_j = \sum_{i \in R_{jt}} s_i$

$$\begin{aligned} Obj^{(t)} &= \sum_{j=1}^{T} \left[\left(\sum_{x_i \in R_{jt}} g_i \right) w_j + \frac{1}{2} \left(\sum_{x_i \in R_{jt}} s_i + \lambda \right) w_j^2 \right] + \gamma T \\ &= \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2}(S_j + \lambda) w_j^2 \right] + \gamma T \end{aligned} \tag{3.39}$$

把 $Obj^{(t)}$ 對 w_j 做微分求最大值，可以解出 $w_j^* = -\frac{G_j}{S_j + \lambda}$，並帶回 $Obj^{(t)}$

$$Obj^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{S_j + \lambda} + \gamma T \tag{3.40}$$

一開始從深度為 0 做切割，並找出最大 Gain 值的切割點（值越大則損失下降越多）

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{S_L+\lambda} + \frac{G_R^2}{S_R+\lambda} - \frac{G^2}{S+\lambda}\right] - \gamma \tag{3.41}$$

最後，我們去做 $\hat{f}(x) = \hat{f}^{(t-1)}(x) + \upsilon h_t(x)$ 其中 υ 為學習率（learning rate）這樣的用意為我們不會在每個步驟中做完全優化（full optimization），這樣可以防止 overfitting

表 4: 參數說明

模型參數	
eta	shrinkage 參數，用於做迭代時，乘以該係數來避免步長過長。
gamma	控制模型防止 overfitting，此最佳值須取決於數據及其他參數值。
max_depth	每棵 Decision Tree 的最大深度，樹越深，則越容易 overfitting。
min_child_weight	在分類中，控制葉子節點中二階導數計算的和的最小值，若小於 min_child_weight，樹則停止分裂。
subsample	控制每棵 Decision Tree 所需要的樣本的數量比例，過多則可能導致 overfitting。
colsample_bytree	控制每棵 Decision Tree 所需要的變數 (variable) 的數量比例。

4 模型配適

4.1 Logistic Regression Model

利用牛頓法執行迭代求解，Equation (3.6) 的參數估計結果如表 5 所示；準確率為 0.8692 ；圖 6 為 logistic regression model 的 ROC 以及 AUC值為 0.8154。

term	estimate	std.error	statistic	p.value
(Intercept)	1.54347	0.93437	1.65188	0.09856
BusinessTravelTravel_Frequently	2.2757	0.50478	4.50828	0.00001
BusinessTravelTravel_Rarely	1.41524	0.46192	3.06384	0.00219
DistanceFromHome	0.05573	0.01249	4.46143	0.00001
EducationFieldLifeSciences	-0.99915	0.33707	-2.96426	0.00303
EducationFieldMarketing	-0.67034	0.44078	-1.52081	0.12831
EducationFieldMedical	-1.0412	0.34586	-3.01049	0.00261
EducationFieldOther	-1.54064	0.5695	-2.70523	0.00683
EnvironmentSatisfaction	-0.42222	0.09699	-4.35333	0.00001
GenderMale	0.41387	0.21296	1.94339	0.05197
JobInvolvement	-0.58838	0.14106	-4.17117	0.00001
JobRoleHumanResources	0.96647	0.50612	1.90957	0.05619
JobRoleLaboratoryTechnician	1.10384	0.28648	3.85316	0.00001
JobRoleSales Executive	0.86193	0.30705	2.80713	0.005
JobRoleSalesRepresentative	1.78911	0.3979	4.49633	0.00001
JobSatisfaction	-0.45472	0.09456	-4.80855	0.00001
MaritalStatusMarried	0.43752	0.30509	1.43407	0.15155
MaritalStatusSingle	1.66099	0.31449	5.28157	0.00001
NumCompaniesWorked	0.14944	0.04469	3.34429	0.00001
OverTimeYes	2.11536	0.22916	9.23074	0.00001
RelationshipSatisfaction	-0.27543	0.09751	-2.82457	0.00473
TotalWorkingYears	-0.08422	0.02089	-4.03097	0.00001
TrainingTimesLastYear	-0.20293	0.08364	-2.42614	0.01526
WorkLifeBalance	-0.45452	0.14385	-3.15961	0.00158
YearsInCurrentRole	-0.17064	0.04471	-3.8168	0.00001
YearsSinceLastPromotion	0.19887	0.04527	4.39264	0.00001

表 5: Parameter Estimations

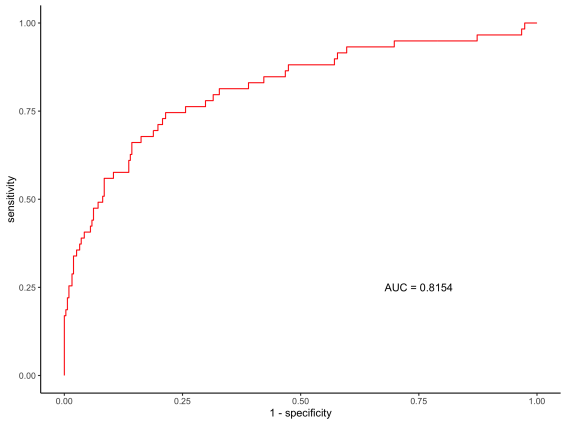


Figure 6: Logistic regression: ROC and AUC

4.2 Naive Bayes

因為此筆資料的解釋變數為混和型變數，我們需要將連續型變數分成許多區間來轉變為類別型變數。此外，我們需要加修正項，避免估計值為 0 的問題。然而，模型方法中所提到的正則化方法，需要決定是否常態（kernal）以及選取適合的平滑參數（fl）與帶寬（bandwith）此三個參數來訓練模型。根據圖 7 可以發現 ROC 表現最好的是 nonparametric（kernal=0） 中 bandwith = 4、fl = 1 的模型。透過此模型根據 testing data 去計算其 ROC，根據圖 8 可以得知 AUC 值為 0.8079 以及透過 caret 套件計算出的準確率為 0.8583。

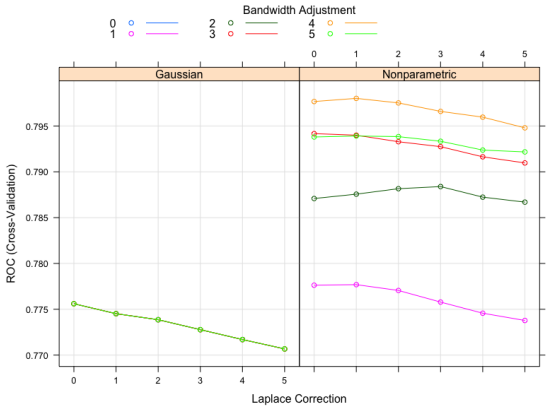


Figure 7: NB

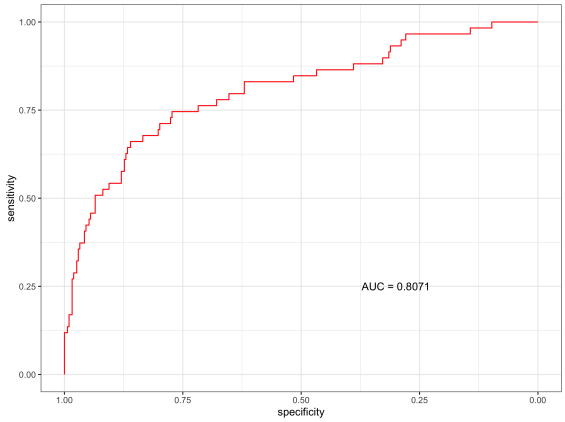


Figure 8: NB: ROC and AUC

4.3 Tree

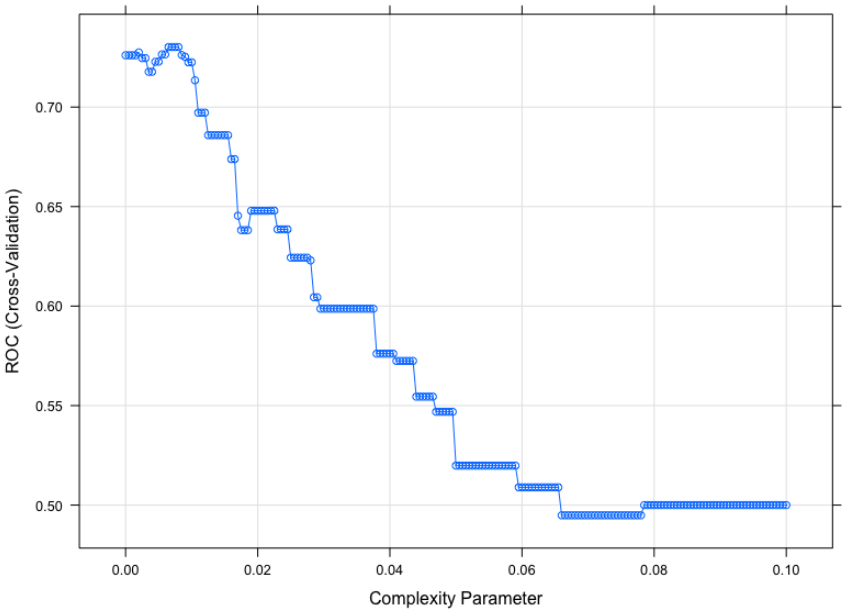


Figure 9: tree

透過圖 9 利用 caret 套計算，我們可以看到模型在不同的 cp 值下會對應到不同的 AUC 值。當 cp=0.008 時，AUC 的表現會達到最好，因此我們的運用 caret 套件所建立的 Tree model 的 cp 會選擇 0.008。

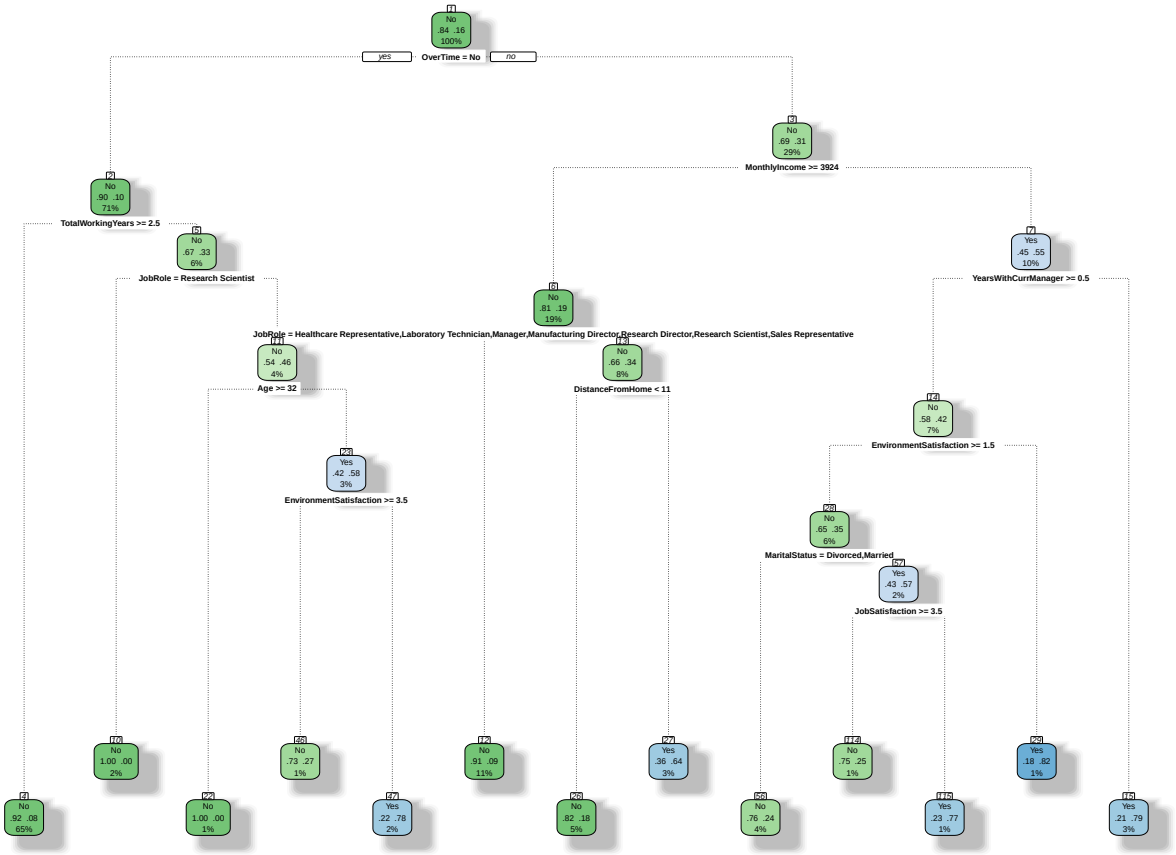


Figure 10: Decision Tree

透過分類樹結果我們可以看到潛在的離職族群如下：

- 滿足工作超時、月收入小於 3924（美元）、以及與現任主管共事未超過半年的員工，分類樹會預測此族群的員工會離職。
- 滿足工作超時、月收入小於 3924（美元）、與現任主管共事超過半年、對工作環境滿意讀為 "Low" 的員工，分類樹會預測此族群的員工會離職。
- 滿足工作超時、月收入小於 3924（美元）、與現任主管共事超過半年、對工作環境滿意度為 "Medium"、"High"、"Very High"、單身族群、工作滿意度為 "Low"、"Medium"、"High" 的員工，分類樹會預測此族群的員工會離職。

- 滿足工作超時、月收入大於3924（美元）、職位為 "Human Resource"、"Sales Executive"、公司離家距離超過 11 （英哩）的員工，分類樹會預測此族群的員工會離職。

- 滿足工作未超時、總工作年資未超過 2.5 年、職位不為 "Research Scientist"、年齡不超過 32 歲、對工作環境滿意度為"Low"、"Medium"、"High"的員工，分類數會預測此族群的員工會離職。

透過上方分類樹預測離職員工族群，我們會發現某些可能是造成離職的原因，並給予人資部門一些建議：

建議各部門主管不要讓員工有工作超時的問題產生、盡量讓工作環境達到員工的滿意。

最後，透過 caret 套件計算出的準確率為 0.8365。透過圖 11 的 ROC curve，我們可以看出此分類樹的 AUC 為 0.6446。

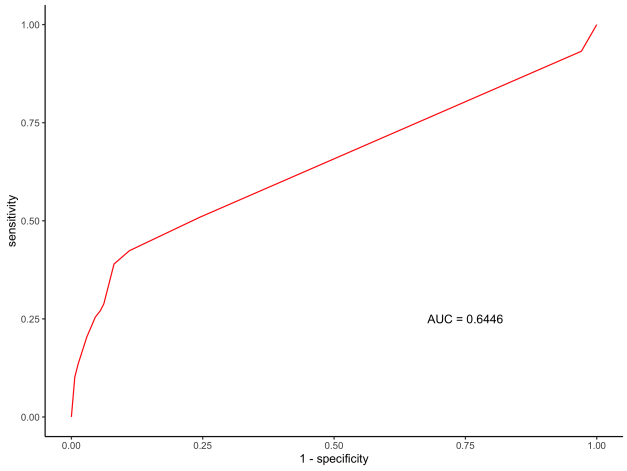


Figure 11: Tree: ROC and AUC

4.4 Random Forest

caret 的套件預設 sampsize 為0.6、nodesize 為 1。

接著，分別對 ntree 和 mtry 做 grid search，此處以 Out-of-Bag Error 作為標準進行參數調控。

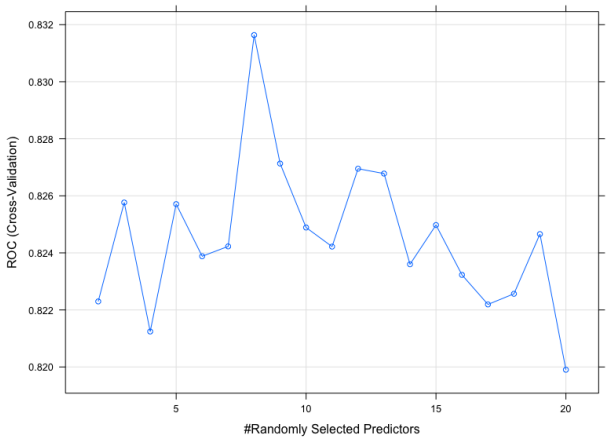


Figure 12: Parameter

表 6 為做完 grid search 後，選出表現最好的參數組合。

表 6: Parameters

模型參數（Parameters）			
ntree	1000	mtry	8
sampsize	0.6	nodesize	1

模型配適完後，將 testing data 帶入模型進行預測，得到結果如圖13：

我們得到準確率為 0.8501，而透過 ROC curve，我們得到 Random Forest 的 AUC 為 0.7814。

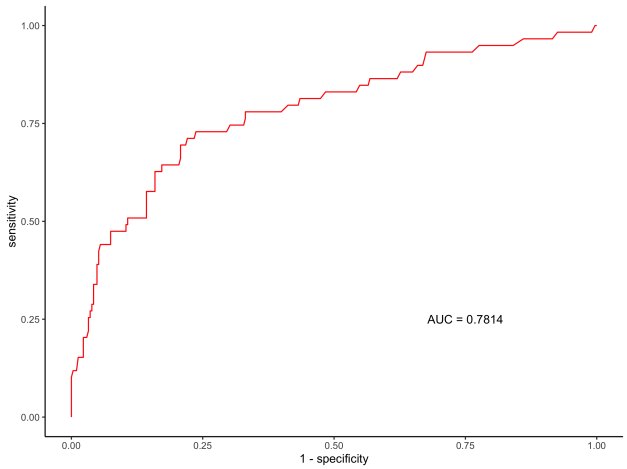


Figure 13: Random Forest: ROC and AUC

4.5 XGBOOST

利用 caret 套件，XGboost 模型中參數 nround 設為 30 或 50，並調控對於樹的模型結構影響較大的兩個參數為 eta 以及 max_depth ，更進一步調控 min_child_weight、subsample、colsample bytree 等參數；此處以 Accuracy 當作選模準則，進而決定模型中參數的選擇。

此處利用 Cross-Validation 來進行 grid search，並挑選出表現較好的參數組合，將這些參數組合繪製出 AUC 的比較圖，如圖14

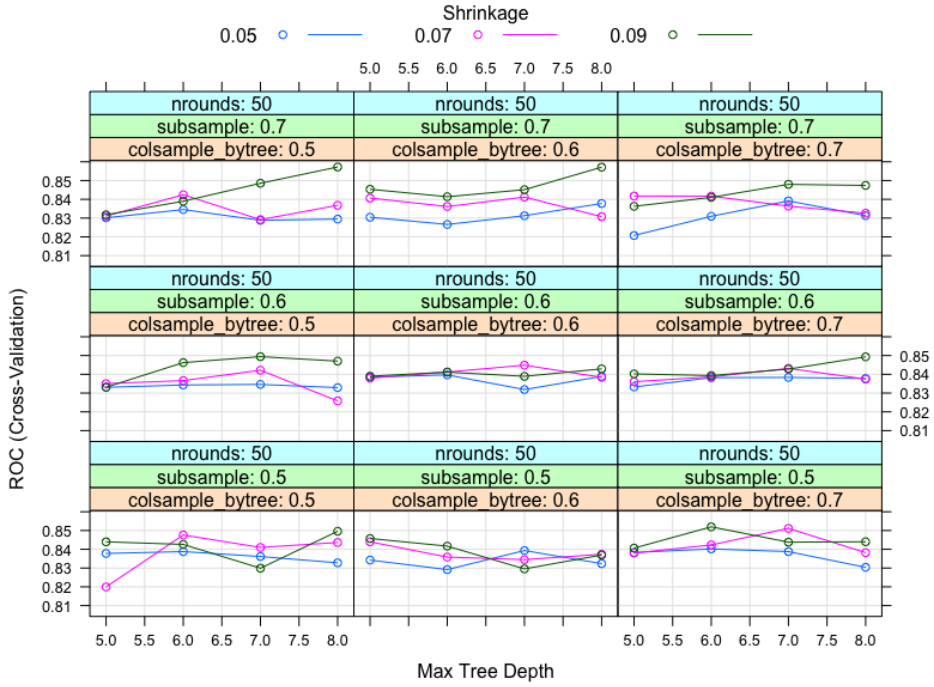


Figure 14: XGboost

根據上圖做完 grid search 後，我們選出表現最好的參數組合。如表7

表 7: Parameters

模型參數（Parameters）			
eta	0.21	max_depth	2
subsample	0.5	colsample_bytree	0.7
gamma	0.05	min_child_weight	1

模型配適完後，將 testing data 放入模型進行預測後我們得到 ROC curve，則 AUC 值根據圖 15 其值為 0.811；且其 model 的準確率為 0.8583。

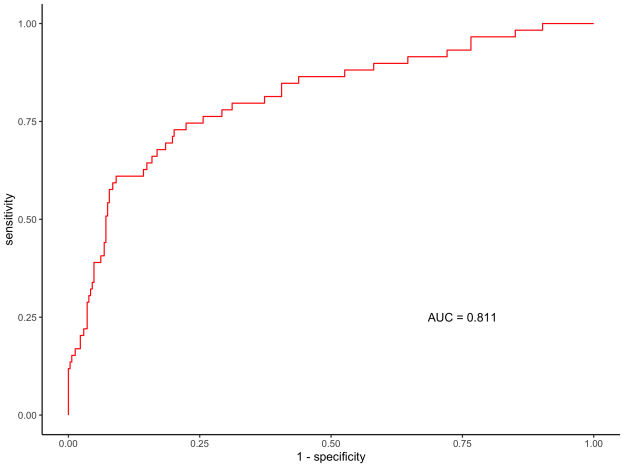


Figure 15: XGboost: ROC and AUC

5 結論

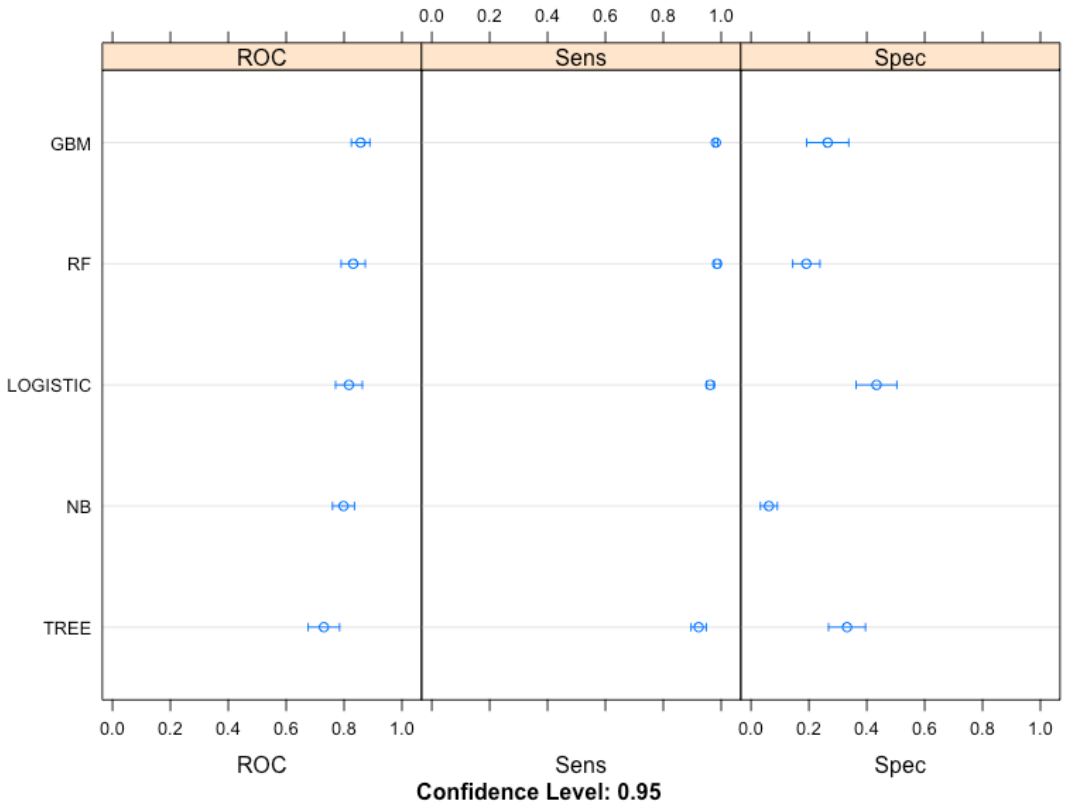


Figure 16: ROC&Sens&Spec

透過圖 16，在 ROC 之下我們發現 Tree 的表現相對不佳，並且可以觀察到每個 model 下的 Sensitivity 都十分接近 1，我們猜想此資料可能是不平衡資料（Imbalance data）。

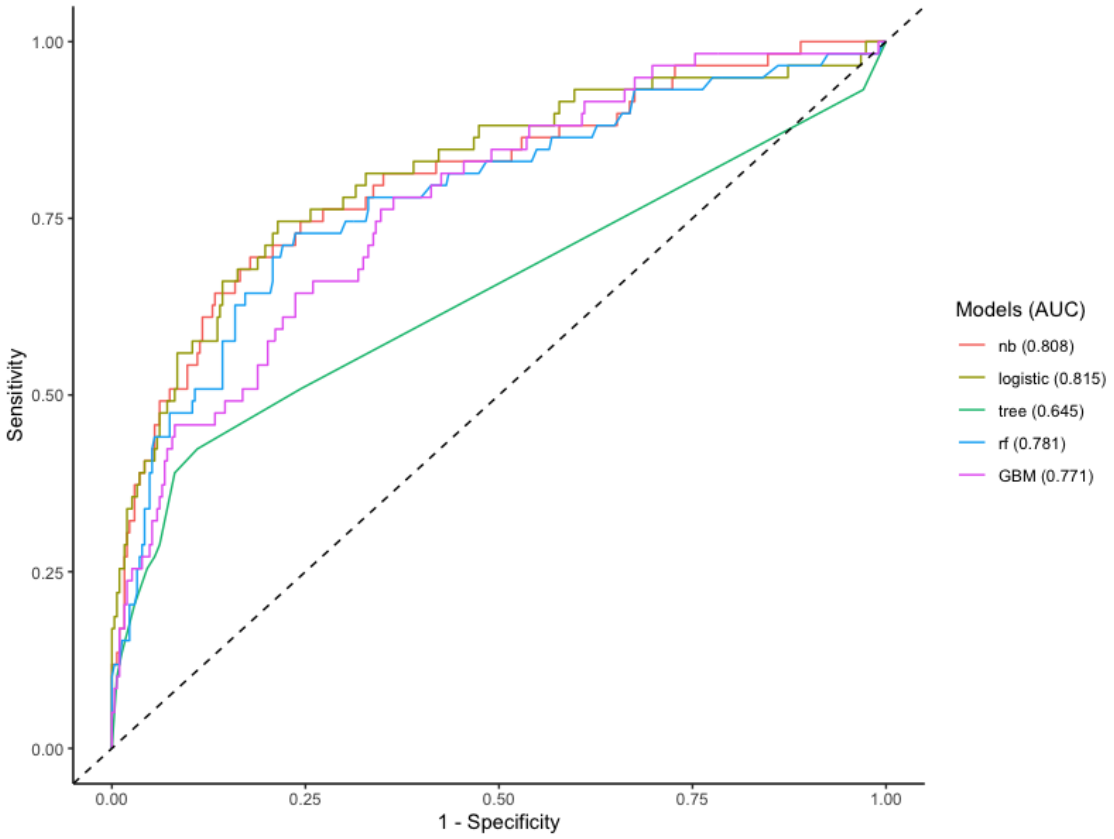


Figure 17: each model with ROC Curve

根據圖 17，可以看出每個 model 相對應的 ROC curve，若以 AUC 為選模標準時，可知 Logistic Regression Model 為最佳模型，而 Tree 則相對表現不佳。

最後，我們在此資料分析會選擇 Logistic Regression Model 做為模型以預測該名員工的離職傾向，並提供給人力資源部門作為參考依據。

A 工作分配表

組員	工作分配
王浚驊	EDA、Naives Bayes、簡報彙整
林鼎智	EDA、Logistic Regression、Code彙整
李泓緯	EDA、XGBOOST、簡報彙整
呂文翔	EDA、Random Forest、書面報告彙整
宋承恩	EDA、Classification Tree、書面報告彙整

B 程式碼

```
pkgs <- c(
  "tidyverse", "rpart", "caret", "e1071", "xgboost", "doParallel", "doMC",
  "plotROC","pROC", "RColorBrewer", "rattle", "broom"
        )
sapply(pkgs, library, character.only = T)


HR <- read_rds("Datasets/HR_preTrain.rds")
debug_contr_error(HR)


idx <- as.vector(createDataPartition(HR_dt$Attrition, p = 0.75, list = F))
trn <- HR[idx, ]
tst <- HR[-idx, ]

# Define color for Attrition
```

```
twoClassColor <- brewer.pal(3,'Set1')[1:2]
names(twoClassColor) <- c('No','Yes')


############## 0. Self-defined functions ################


plot_conti <- function(var_name, target_name, dt) {
  p1 <- ggplot() +
    geom_histogram(aes(x = dt[[var_name]]), colour = "white") +
    theme_bw()+
    xlab(var_name)

  p2 <- ggplot() +
    geom_boxplot(aes(x = dt[[target_name]], y = dt[[var_name]])) +
    xlab(target_name) +
    ylab(var_name) +
    theme_bw()

  p <- grid.arrange(p1, p2, ncol = 2)
  return(p)
}


plot_discrete <- function(var_name, target_name, dt) {

  # rquire pacakge
  library(vcd)

  # build formula
  formula_f <- as.formula(paste(target_name, "~", var_name))
```

```
  # call mosaic() to create visualization of contigency table
  p <- mosaic(formula_f, data = dt)

  return(p)
}



# Draw ROC and calculate AUC
drawROC <- function(mod, test_dt){
  preds <- predict(mod, test_dt, type = "prob")
  ROC_obj <- roc(response = test_dt$Attrition, predictor = preds$No)
  ggroc(ROC_obj, color = "red", legacy.axes = TRUE) +
    annotate("text", x = 0.75, y = 0.25, label=paste("AUC=",
    round((auc(ROC_obj)), 4))) +
    theme_classic()
}

drawROC_multiple <- function(test_dt, modList) {
  len <- length(modList)

  ROC_obj_lst <- list()

  for (i in 1:len) {
    preds = predict(modList[[i]], test_dt, type = "prob")
    roc_obj = roc(response = test_dt$Attrition, predictor = preds$No)
    name <- paste0(names(modList)[i], "␣", "(", round(auc(roc_obj), 3),")")
    ROC_obj_lst[[name]] = roc(response = test_dt$Attrition,
    predictor = preds$No)
  }
```

```
  ggroc(ROC_obj_lst, legacy.axes = TRUE) +
    geom_abline(linetype = "dashed")+
    theme_classic() +
    labs(x = "1 - Specificity",
    y = "Sensitivity",
    color = "Models (AUC)")
}



############## 0. Pre-processing ############

for (name in names(debug_obj$levels)) {
  trn[[name]] <- factor(trn[[name]])
  tst[[name]] <- factor(tst[[name]])
}

# One-hot encoding
features <- setdiff(names(trn), "Attrition")
X <- trn[, features]
dummies_rule <- dummyVars(Attrition ~., data = trn)
X_oneHOT <- predict(dummies_rule, trn)


trnControl <- trainControl(
  method = "cv",
  number = 10,
  verboseIter = T,
  allowParallel= F,
  savePredictions = TRUE,
  summaryFunction = twoClassSummary,
```

```
  classProbs = T,
  )

# trnControl <- trainControl(
#   method = "cv",
#   number = 10,
#   verboseIter = T,
#   allowParallel= F,
#   savePredictions = TRUE,
#   summaryFunction = twoClassSummary,
#   classProbs = T
# )

# setting for doParallel
cl <- registerDoMC(4)
registerDoParallel(cl)


#################### 1. Naive Bayes #################

set.seed(123)
# tuning parameters for Naive Bayes
nb_searchGrid <- expand.grid(
  usekernel = c(T, F),
  fL = 0:5,
  adjust = seq(0, 5, 1)
)

# train NaiveBayes model
naiveBayes_mod <- train(
  x = X,
```

```
  y = Y,
  method = "nb",
  trControl = trnControl,
  tuneGrid = nb_searchGrid,
  allowParallel = F,
  metric = "ROC"
)

# table of result
nb_res <- naiveBayes_mod$results %>%
  top_n(5, wt = ROC) %>%
  arrange(desc(ROC))

write_csv(nb_res, "Results/nb_res.csv")

# Calculate testing error
preds_prob <- predict(naiveBayes_mod, tst[, -2], type = "prob")[["Yes"]]
preds <- ifelse(preds_prob > 0.5, 1, 0)
true_labels <- as.integer(as.character(factor(tst$Attrition,
labels = c(0, 1))))
mean(preds == true_labels) # 0.8583106

# Visualization for tuning results
plot(naiveBayes_mod)
ggsave("Plots/para_NB.png", width = 8, height = 6, dpi = 320)

# ROC
drawROC(mod = naiveBayes_mod, test_dt = tst)
ggsave("Plots/ROC_NB.png", width = 8, height = 6, dpi = 320)

################ 2. Logistic regression ########################
```

```
# Build the logistic regression model based on AIC
set.seed(123)
logistic_mod <- train(
  Attrition ~ .,
  data = trn,
  method = "glmStepAIC",
  family = "binomial",
  trControl = trnControl
)

# Calculate testing error
preds_prob <- predict(logistic_mod, tst, type = "prob")[["Yes"]]
preds <- ifelse(preds_prob > 0.5, 1, 0)
true_labels <- as.integer(as.character(factor(tst$Attrition,
labels = c(0, 1))))
mean(preds == true_labels) # 0.8692098

# Summary of this parametric model # AIC = 685.57
summary(logistic_mod)
glm_res <- tidy(logistic_mod$finalModel)
glm_res <- bind_cols(glm_res[,1], round(glm_res[,-1],5))
write_csv(glm_res, "Results/glm_res.csv")

# ROC
drawROC(mod = logistic_mod, test_dt = tst)
ggsave("Plots/ROC_logistic.png", width = 8, height = 6, dpi = 320)

confu <- confusionMatrix(
  data = logistic_mod,
  reference = tst$Attrition,
```

```
)

################## 3. tree model ####################

set.seed(123)
tree_searchGrid <- expand.grid(
  cp = seq(0, 0.1, by=0.0005)
  )


tree_mod <- train(
  Attrition ~ .,
  data = trn,
  method = "rpart",
  trControl = trnControl,
  tuneGrid = tree_searchGrid,
  metric = "ROC"
)

# table of result  # optimal cp = 0.008
tree_res <- tree_mod$results %>%
  top_n(5, wt = ROC) %>%
  arrange(desc(ROC))

write_csv(tree_res, "Results/tree_res.csv")


# Calculate testing error
preds_prob <- predict(tree_mod, tst, type = "prob")[["Yes"]]
preds <- ifelse(preds_prob > 0.5, 1, 0)
true_labels <- as.integer(as.character(factor(tst$Attrition,
```

```
labels = c(0, 1))))
mean(preds == true_labels) # 0.8365123

# ROC
drawROC(mod = tree_mod, test_dt = tst)
ggsave("Plots/ROC_tree.png", width = 8, height = 6, dpi = 320)

#
plot(tree_mod)
ggsave("Plots/para_tree.png", width = 8, height = 6, dpi = 320)
fancyRpartPlot(tree_mod$finalModel, sub = "")
savePlotToFile("tree_classifer.png")


################## 4. random forest ########################

# number of features
n_features <- length(setdiff(names(trn), "Attrition"))
mtry_optimal <- sqrt(n_features) # 5

rf_searchGrid <- expand.grid(
  mtry = seq(2, 20, by = 1)
  )

set.seed(123)
# Build random forest model
rf_mod <- train(
  Attrition ~ .,
  data = trn,
  method = "rf",
  trControl = trnControl,
```

```
  ntrees = 1000,
  tuneGrid = rf_searchGrid,
  metric = "ROC"
)

# table of result  # optimal mtry = 8
rf_res <- rf_mod$results %>%
  top_n(5, wt = ROC) %>%
  arrange(desc(ROC))

write_csv(rf_res, "Results/rf_res.csv")

# Calculate testing error
preds_prob <- predict(rf_mod, tst, type = "prob")[["Yes"]]
preds <- ifelse(preds_prob > 0.5, 1, 0)
true_labels <- as.integer(as.character(factor(tst$Attrition,
labels = c(0, 1))))
mean(preds == true_labels) # 0.8501362

# ROC
drawROC(mod = rf_mod, test_dt = tst)
ggsave("Plots/ROC_rf.png", width = 8, height = 6, dpi = 320)

#
plot(rf_mod)
plot(rf_mod$finalModel)


################## 5. xgboosting  ######################

xgboost_searchGrid <- expand.grid(
```

```
  nrounds = c(30, 50),
  max_depth = seq(5, 8, by = 1),
  eta = seq(0.05, 0.09, by=0.02),
  gamma = c(0),
  colsample_bytree = seq(0.5, 0.7, by=0.1),
  min_child_weight=c(1),
  subsample = seq(0.5, 0.7, by=0.1)
  )

xgboost_mod <- train(
  Attrition ~ .,
  data = trn,
  method = "xgbTree",
  metric = "ROC",
  trControl = trnControl,
  tuneGrid = xgboost_searchGrid,
  allowParallel = T
  )

plot(xgboost_mod)



# Fit the final model (optimal parameters)
xgboost_Grid <- expand.grid(
  nrounds = 180,
  max_depth = 2,
  eta = 0.21,
  gamma = 0.05,
  colsample_bytree = 0.7,
  min_child_weight = 1,
```

```
  subsample = 0.5
)

set.seed(123)
xgboost_mod_final <- train(
  Attrition ~ .,
  data = trn,
  method = "xgbTree",
  metric = "ROC",
  trControl = trnControl,
  tuneGrid = xgboost_Grid,
  allowParallel = T
)


# Calculate testing error
preds_prob <- predict(xgboost_mod_final, tst, type = "prob")[["Yes"]]
preds <- ifelse(preds_prob > 0.5, 1, 0)
true_labels <- as.integer(as.character(factor(tst$Attrition,
labels = c(0, 1))))
mean(preds == true_labels) # 0.8583106

# ROC
drawROC(mod = xgboost_mod_final, test_dt = tst)
ggsave("Plots/ROC_xgboost.png", width = 8, height = 6, dpi = 320)

############## 6. xgboosting (another version) ###############

# using one hot encoding
labels <- trn$Attrition
tst_labels <- tst$Attrition
```

```
X_oneHOT <- model.matrix(~.+0,data = trn %>% dplyr::select(-Attrition))
X_oneHOT_tst <- model.matrix(~.+0,data = tst %>% dplyr::select(-Attrition))

# convert factor to numeric
labels <- as.integer(as.character(factor(labels, labels = c(0, 1))))
tst_labels <- as.integer(as.character(factor(tst_labels,
                          labels = c(0, 1))))

# create DMatrix for modeling
dtrain <- xgb.DMatrix(data = X_oneHOT, label = labels)
dtest <- xgb.DMatrix(data = X_oneHOT_tst, label = tst_labels)

# grid search
searchGridCol <- expand.grid(
  eta = seq(0.05, 0.09, by = 0.02),
  max_depth = seq(5, 8, by=1),
  subsample = seq(0.5, 0.7, by = 0.1),
  colsample_bytree = seq(0.5, 0.7, by = 0.1),
  lambda=seq(0.6, 1.0, by = 0.2),
  alpha=seq(0.6, 1.0, by = 0.2)
  )

ntrees = 30

system.time(AUCHyperparameters <- apply(searchGridCol, 1,
function(parameterList){

  currentEta <- parameterList[["eta"]]
  currentDepth <- parameterList[["max_depth"]]
  currentSubsampleRate <- parameterList[["subsample"]]
  currentColsampleRate <- parameterList[["colsample_bytree"]]
```

```
currentLambda <- parameterList[["lambda"]]
currentAlpha <- parameterList[["alpha"]]

xgboostModelCV <- xgb.cv(
  data = dtrain,
  nrounds = ntrees,
  nfold = 10,
  showsd = TRUE,
  metrics = "auc", booster = "gbtree",
  eval_metric = "auc",
  objective = "binary:logistic",
  "eta"=currentEta,
  "max_depth"=currentDepth,
  "subsample"=currentSubsampleRate,
  "colsample_bytree"=currentColsampleRate,
  "lambda"=currentLambda,
  "alpha"=currentAlpha,
  print_every_n=10,
  early_stopping_rounds=10
  )

xvalidationScores <- as.data.frame(xgboostModelCV$evaluation_log)

auc <-
xvalidationScores$test_auc_mean[xgboostModelCV$best_ntreelimit]
output <- return(
  c(auc, currentEta, currentDepth, currentSubsampleRate,
  currentColsampleRate, currentLambda, currentAlpha)
  )
}
)
```

```
)

output <- as.data.frame(t(AUCHyperparameters))
varnames <- c('TestAUC', 'eta', 'Depth', 'SubSampleRate',
              'ColSampRate', 'Lambda', 'Alpha')
names(output) <- varnames
output[which.max(output$TestAUC), ]

# run the above code from the remote server
xgboos_res <- read_csv("Datasets/xgboost.csv")
head(xgboos_res)

params <- list(
  booster = 'gbtree',
  objective = 'binary:logistic',
  eta = 0.05, # learning rate
  max_depth = 5, # tree depth
  subsample = 0.5, # percentage data use in every iteration
  colsample_bytree = 0.5, # percentage covariate use in every iteration
  lambda = 0.6,
  alpha = 0.6
)

xgb_model <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = 300,
  eval_metric = 'auc',
  print_every_n = 10,
  watchlist = list(val = dtest, train = dtrain),
  early_stopping_rounds = 10,
```

```
  maximize = T
  )

# Calculate testing error
prob_test <- predict(xgb_model, dtest)
preds <- ifelse(preds_prob > 0.5, 1, 0)
true_labels <- as.integer(as.character(factor(tst$Attrition,
                          labels = c(0, 1))))
mean(preds == true_labels) # 0.852


# ------------------------------------------------------------

# Compare the results

# collect resamples
results <- resamples(
  list(NB = naiveBayes_mod, LOGISTIC = logistic_mod,
  TREE = tree_mod, RF = rf_mod, GBM = xgboost_mod)
  )

# summarize the distributions
summary(results)

# dot plots of results
dotplot(results)

# Plot multiple ROC in single figure
drawROC_multiple(
  tst, modList = list(
    "nb" = naiveBayes_mod, "logistic" = logistic_mod, "tree" = tree_mod,
```

```
    "rf" = rf_mod, "GBM" = xgboost_mod)
)

ggsave("Plots/ROC_multiple.png", width = 800, height = 600, dpi = 320)

## When you are done:
stopCluster(cl)
```