ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN


# BÁO CÁO

## Lab 01: Trực quan hóa dữ liệu với Python
## Bộ môn: Trực quan hóa dữ liệu

### NHÓM 22

| | |
|---|---|
| **Nhóm trưởng** | 20120128 – Nguyễn Thị Cẩm Lai |
| **Thành viên** | 20120037 – Trần Thị Minh Anh |
| **Thành Viên** | 20120125 – Bùi Anh Kiệt |
| **Thành viên** | 20120232 – Võ Duy Trường |
| **Thành viên** | 20120547 – Võ Thành Phong |

**Giáo viên hướng dẫn:** Nguyễn Thị Thu Hằng
Nguyễn Bảo Long
Lê Nhựt Nam

# MỤC LỤC

# I. Thông tin chung

## 1. Thông tin thành viên

| Họ và tên | MSSV | Email |
|---|---|---|
| Nguyễn Thị Cẩm Lai (nhóm trưởng) | 20120128 | 20120128@student.hcmus.edu.vn |
| Trần Thị Minh Anh | 20120037 | 20120073@student.hcmus.edu.vn |
| Bùi Anh Kiệt | 20120125 | 20120125@student.hcmus.edu.vn |
| Võ Duy Trường | 20120232 | 20120232@student.hcmus.edu.vn |
| Võ Thành Phong | 20120547 | 20120547@student.hcmus.edu.vn |

- **Github nhóm:** https://github.com/DataVisualizationGroup22/DataVisualization_Lab01

## 2. Phân công công việc

| Phần | Yêu cầu | Người thực hiện | Mức độ hoàn thành |
|---|---|---|---|
| A.Thu thập dữ liệu | Lựa chọn dataset | Cả nhóm | 100% |
| | Trình bày ngữ cảnh và động cơ lựa chọn | Cẩm Lai | 100% |
| B. Khám phá dữ liệu (thường đan xen với pha tiền xử lý dữ liệu) | 1. Đọc dữ liệu và tính số dòng và cột | Duy Trường | 100% |
| | 2. Mỗi dòng có ý nghĩa gì? Có vấn đề các dòng có ý nghĩa khác nhau không? | | |
| | 3. Dữ liệu có các dòng bị lặp không? | | |
| | 4. Tỷ lệ giá trị thiếu và thống kê mô tả của từng cột | | |
| | 5. Kiểu dữ liệu của mỗi cột | | |
| | 6. Xem xét tập giá trị của các thuộc tính phân loại | Anh Kiệt | 100% |
| | 7. Xem xét sự phân bố giá trị của các cột dữ liệu dạng số | | |
| | 8. Xem xét sự phân bố giá trị của các cột dữ liệu không phải dạng số | | |
| C. Khám phá mối quan hệ trong dữ liệu | Khám phá mối quan hệ đơn biến (độc lập) | Yêu cầu tối thiểu:<br>• Minh Anh (3 khám phá)<br>• Cẩm Lai (3 khám phá)<br>• Anh Kiệt (2 khám phá)<br>• Duy Trường (2 khám phá)<br>• Thành Phong (1 khám phá) | 100% |
| | Khám phá mối quan hệ đa biến/nhân quả (tích hợp) | | |
| D. Mô hình học máy | Sử dụng thuật toán học máy đơn giản để hiểu thêm về dữ liệu | Thành Phong | 100% |
| E. Viết báo cáo | | Cẩm Lai, Thành Phong | 100% |

### 3. Đánh giá mức độ hoàn thành

| Phần | Yêu cầu | Mức độ hoàn thành |
|---|---|---|
| A. Thu thập dữ liệu | Lựa chọn dataset | 100% |
| | Trình bày ngữ cảnh và động cơ lựa chọn | 100% |
| B. Khám phá dữ liệu | Khám phá dữ liệu | 100% |
| | Tiền xử lý dữ liệu | 100% |
| C. Khám phá mối quan hệ trong dữ liệu | Khám phá mối quan hệ đơn biến | 100% |
| | Khám phá mối quan hệ đa biến/nhân quả | 100% |
| | Trực quan hóa trên nhiều loại biểu đồ | 100% |
| D. Mô hình học máy | Chạy một số thuật toán học máy đơn giản để hiểu thêm về dữ liệu | 100% |
| | Trình bày chi tiết thuật toán | 100% |

# II. Chi tiết thuật toán

## 1. Bài toán đặt ra

**a) Bài toán**

Dự đoán liệu rằng nhân viên có nguy cơ nghỉ việc ở công ty đang làm việc hay không?

**b) Giới thiệu chung**

- Trong học máy, học có giám sát là một nhóm các thuật toán phổ biến trong lĩnh vực này và một trong những vấn đề quan trọng của học có giám sát là phân loại/phân lớp(classification problem).
- Có 2 dạng classification thường gặp là: binary classification và multiclassification, và trong bài toán mà nhóm đặt ra thì đây là một vấn đề binary classification: từ những thuộc tính đầu vào của một nhân viên như tổng số năm làm việc, mức lương, sự hài lòng về môi trường làm việc, ... dự đoán rằng nhân viên đó nguy cơ nghỉ làm ở công ty hiện tại hay không(Công ty có bị mất mát nhân viên hay không) (0: Không/ 1: Có).
- Nhóm sẽ tạo một mô hình logistic regression cho bài toán phân loại nhị phân.

## 2. Tiền xử lý dữ liệu

**a) Mã hóa các thuộc tính dạng danh mục về dạng số**

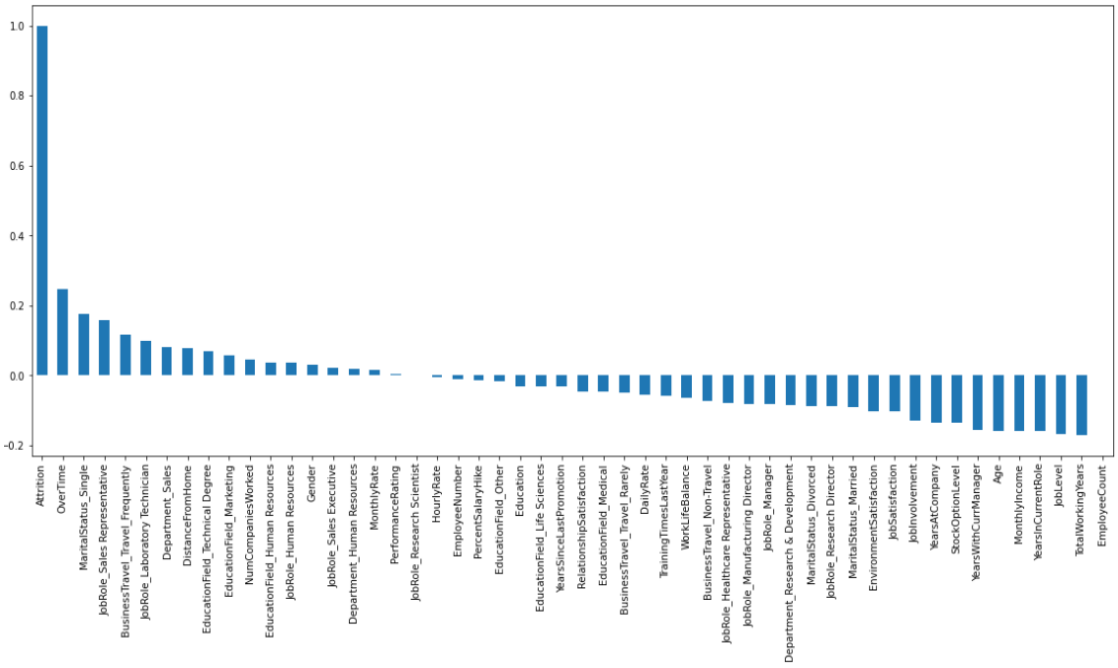
- Đầu vào cho các mô hình học máy phải là các vector, ma trận, hay tensor dạng số, do đó việc chuyển các cột dạng danh mục về dạng số là bước quan trọng trước khi tiếp tục tin chỉnh các thuộc tính cho việc làm đầu vào của mô hình.
- Tuy được chuyển về dạng số nhưng về ý nghĩa thật sự các cột này vẫn có thể được hiểu theo ý nghĩa phân loại như khi còn là giá trị dạng danh mục.
- Đầu tiên lọc ra các thuộc tính là dạng danh mục, sau đó dựa vào số lượng giá trị khác nhau trong mỗi cột mà có cách mã hóa khác nhau:

- Cột Over18 chỉ chứa 1 loại giá trị duy nhất là 'Y' chứng tỏ cột này sẽ không ảnh hưởng đến kết quả của việc huấn luyện mô hình.
- Có 3 thuộc tính là Attrition, Gender, và OverTime là có 2 loại giá trị khác nhau, do đó với các thuộc tính này chúng ta có thể mã hóa bằng cách gán nhãn 0/1 cho chúng.
- Đối với các thuộc tính có nhiều hơn 2 loại giá trị, chúng ta sẽ má hõa bằng one-hot vector, lý do sử dụng one-hot mà không dùng ordinal hay label để tránh xảy ra hiện tượng bias do mã hóa thành các giá trị lớn nếu số lượng giá trị lớn.

**b) Loại những thuộc tính không có ý nghĩa cho bài toán**

Có nhiều cách để có thể chọn lọc thuộc tính (đặc trưng) phù hợp nhất cho bài toán, trong bài lab này nhóm sẽ thực hiện các cách sau:

- **Cách 1**: Cách thô sơ nhất có thể nghĩ đến là cách loại các thuộc tính có thể thấy ngay về mặt ý nghĩa là không cần thiết cho bài toán phân loại: Ví dụ thuộc tính 'Over18' được loại bỏ ở trên do chúng chỉ mang một giá trị 'yes', dễ hiểu khi bộ dữ liệu được cung cấp bởi IBM, có trụ sở ở Mỹ và nước Mỹ chỉ cho phép đi làm khi đủ 18 tuổi trở lên. Ngoài ra khi nhìn vào bảng ý nghĩa các thuộc tính mà nhóm đã cung cấp ở phần thu thập dữ liệu thì cột 'StandardHours' cũng sẽ không có ý nghĩa cho bài toán vì cột này mang cùng một giá trị cho tất cả nhân viên là giờ làm chuẩn mà họ phải đi làm, có thể xóa thuộc tính này đi.
- **Cách 2:** Sử dụng giá trị correlations giữa từng biến biến độc lập với biến phụ thuộc:
  - Correlation là một thuật ngữ thống kê được sử dụng phổ biến đề cập đến mức độ liên quan của hai biến để có mối quan hệ tuyến tính với nhau hay không.
  - Correlation cao nhất có giá trị là 1 (hai biến hoàn toàn có quan hệ tuyến tính) và thấp nhất dần nếu hai biến càng không có quan hệ tuyến tính.
  - Việc trực quan điểm correlation giữa cột mục tiêu ‘Attrition’ với từng cột thuộc tính sẽ giúp việc đánh giá được dễ dàng hơn.




Dễ nhận thấy cột 'EmployeeCount' không có quan hệ nào với 'Attrition' nên có thể xóa đi.

- **Cách 3:** Sử dụng correlations giữa các biến độc lập với nhau:
  - Hai biến độc lập khi có giá trị correlation càng cao thì chứng tỏ chúng càng mang thông tin giống nhau cho ngữ cảnh của bài toán.
  - Do đó khi hai biến độc lập có correlation cao thì có thể chọn một trong hai để áp dụng vào việc huấn luyện mô hình.

| | feature1 | feature2 | correlation |
|---|---|---|---|
| **0** | JobLevel | MonthlyIncome | 0.950300 |
| **1** | JobLevel | TotalWorkingYears | 0.782208 |
| **2** | MonthlyIncome | TotalWorkingYears | 0.772893 |
| **3** | PercentSalaryHike | PerformanceRating | 0.773550 |
| **4** | YearsAtCompany | YearsInCurrentRole | 0.758754 |
| **5** | YearsAtCompany | YearsWithCurrManager | 0.769212 |
| **6** | YearsInCurrentRole | YearsWithCurrManager | 0.714365 |
| **7** | Department_Human Resources | JobRole_Human Resources | 0.904983 |
| **8** | Department_Sales | JobRole_Sales Executive | 0.808869 |

- **TotalWorkingYears, JobLevel và MonthlyIncome:** Có giá trị correlation rất cao. Chọn giữ lại MonthlyIncome.
- **PercentSalaryHike và PerformanceRating:** Có giá trị correlation là 0.77. Chọn giữ lại PerformanceRating.
- **YearsAtCompany, YearsInCurrentRole, và YearsWithCurrManager:** Có giá trị correlation cao. Chọn giữ lại YearsAtCompany.
- **Department_Human Resources và JobRole_Human Resources:** Có giá trị correlation là 0.9. Chọn giữ lại JobRole_Human Resources.
- **Department_Sales và JobRole_Sales Executive:** Có giá trị correlation là 0.8. Chọn giữ lại JobRole_Sales Executive.

### c) Xử lý các giá trị thiếu

- Nếu số lượng các giá trị thiếu là lớn sẽ có thể ảnh hưởng đến kết quả của mô hình. Do đó cần phải kiểm tra và thực hiện xử lý các giá trị thiếu.
- Bộ dữ liệu không tồn tại bất kì giá trị thiếu nào trong tất cả các cột thuộc tính. Vậy có thể bỏ qua bước này.

**d) Feature Scaling**

- Khi khoảng giá trị giữa 2 thuộc tính quá cách xa nhau thì việc mô hình hóa cũng như trực quan mối quan hệ có thể gặp khó khăn, do đó phải thực hiện kĩ thuật 'Feature Scaling' hay việt hóa là 'Co giãn thuộc tính'.
- Có 3 phương pháp feature scaling chính là:
  - Standardisation (Chính quy hóa): Làm cho tập dữ liệu có trung bình là 0 và độ lệch chuẩn là 1 và được áp dụng cho hầu hết các trường hợp cần feature scaling.
  - Normalisation (Tiêu chuẩn hóa): Làm cho các giá trị trong tập dữ liệu thuộc đoạn [0, 1] và được áp dụng nếu tập dữ liệu tuân theo phân phối chuẩn.
  - MinMax Scaler: Đưa các giá trị về khoảng giữa 2 giá trị min và max trong miền giá trị của thuộc tính, có thể là đoạn [-1, 0], [0, 1], [-1, 1],...
- Trong bài này nhóm chọn phương pháp Standardisation để scaling khoảng giá trị của thuộc tính về khoảng gần hơn với giá trị của tập y.

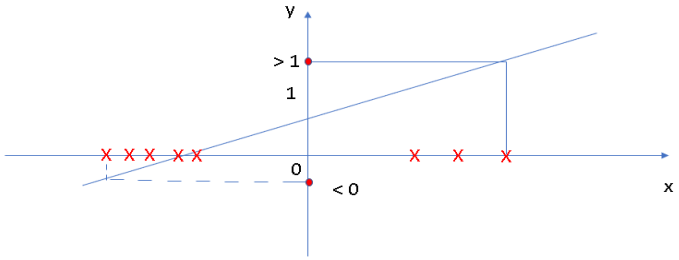
## 3. Cơ sở xây dựng mô hình học máy

**a) Mô tả bài toán**

- Mục tiêu của bài toán phân loại nhị phân là dự đoán xác suất thuộc về một trong hai lớp cần phân lớp của một biến phụ thuộc dựa vào các biến độc lập(hay còn gọi là thuộc tính).
- Trong bộ dữ liệu này chúng ta sẽ thử dự đoán **xác suất thuộc về lớp 0 hoặc 1 của biến Attrition dựa vào các thuộc tính độc lập khác của một người nhân viên.**

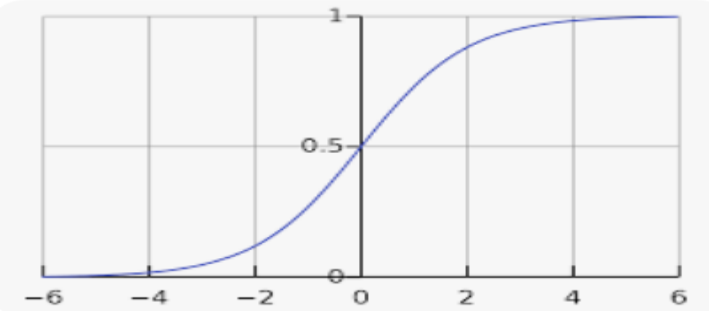
**b) Cost Function**

- Bắt đầu với việc xây dựng một hypothesis tương tự như bài toán linear regression: $h_\theta(x)$: $\theta_0 + \theta_1 x$ với $\theta_i$ (i=0,1) là các tham số (parameters) của công thức hồi quy và $\theta_0$ còn đươc gọi là hệ số tự do.
- Tuy nhiên sử dụng đường thẳng tuyến tính là không phù hợp cho mục tiêu của bài toán khi giá trị dự đoán cần thuộc vào một trong hai lớp là 0 hoặc 1, nhưng với một đường thẳng tuyến tính thì $h_\theta$(x) có thể lớn hơn 1 và nhỏ hơn 0.




- Lúc này cần 1 giải pháp để giá trị dự đoán có thể nằm trong khoảng 0 đến 1 => truyền hypothesis qua một hàm sigmoid mà giá trị trả về của một hàm sigmoid nằm trong khoảng 0 đến 1. Đặt z= $h_\theta(x)$

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

i. 


- Lúc này, ta có giá trị dự đoán y_hat=sigmoid(z) nằm trong khoảng từ 0 đến 1 hay nói cách khác đây chính là giá trị xác suất mà y_hat thuộc về một trong hai lớp với điều kiện cho trước là các thuộc tính đầu vào khác.
- Thông thường, quy ước giá trị trả về của hàm sigmoid biểu thị cho xác suất thuộc lớp 1 của y_hat: y_hat=sigmoid(z)=P(y=1|x) => P(y=0|x) = 1 - y_hat = 1 - P(y=1|x).
- Vậy làm thế nào có thể quy định lớp mà y_hat thuộc về?
  - Chúng ta sẽ sử dụng một giá trị ngưỡng mà nếu y_hat lớn hơn hoặc bằng ngưỡng này sẽ thuộc về lớp 1, ngược lại thuộc về lớp 0.
  - Ngưỡng giá trị thông thường sẽ là 0.5, điều này tương ứng với việc nếu z>=0 thì y_hat thuộc về lớp 1 (do sigmoid(z>=0)>=0.5).
- Sau khi có được tập các giá trị dự đoán của các mẫu đầu vào, để đánh giá xem hypothesis đã tốt hay chưa chúng ta xây dựng một hàm chi phí để tính toán độ sai lệch giữa giá trị dự đoán và giá trị thực tế.

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - yi)\log(1 - \hat{y}_i))$$

với $\hat{y}_i$ = y_hat$_i$

- Do càng nhiều giá trị dự đoán giống với giá trị thực tế càng tốt nên $J(\theta)$ có giá trị càng bé càng tốt.

**c) Gradient Descent**

- Các tham số $\theta$ sẽ là những giá trị mà chúng ta cần phải thay đổi để tối ưu hóa Cost Function, và một trong những cách để thực hiện việc này là thuật toán Gradient Descent.
- Thuật toán được thực hiện như sau:
  - **Trong mỗi lần lặp cập nhật một cách đồng thời các tham số $\theta_j$ theo công thức như sau:**

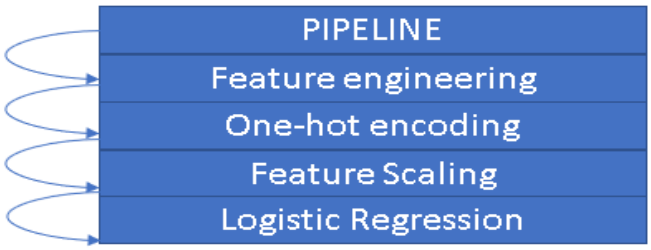
$$\theta_j = \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

Trong đó: alpha là 'learning rate' giúp việc học được tối ưu hơn.

**d) Sử dụng Pipeline và Các độ đo được dùng để đánh giá mô hình**

- **Pipeline**
  - Pipeline là một công cụ giúp kết hợp nhiều bước xử lý dữ liệu và huấn luyện mô hình thành một quy trình hoàn chỉnh.

- Các bước để thực hiện xây dựng một mô hình học máy sẽ được xếp tuần tự trong một đối tượng Pipeline (có thể xem như một đường ống để dẫn lần lượt đi qua các bước).




- **Các độ đo cần thiết cho việc đánh giá mô hình phân loại nhị phân.**
  - Các độ đo được nhóm sử dụng trong bài này sẽ là: Accuracy, Precision, F1-score, Recall.
  - **Giải thích các độ đo:**

    Giả sử chúng ta có bài toán phân loại các thư điện tử đến hộp thư là thư spam hay không spam.

    Nhãn:

    - 1 (positive): là thư spam, tương ứng với đây là nhãn xấu (bad label).
    - 0 (negative): là thư không spam, tương ứng với đây là nhãn tốt (good label).

    Quy ước các chỉ số như sau:
    - TP (True Positive): Tổng số trường hợp dự đoán đúng mẫu dương.
    - FP (False Positive): Tổng số trường hợp dự đoán sai mẫu dương.
    - TN (True Negative): Tổng số trường hợp dự đoán đúng mẫu âm.
    - FN (False Negative): Tổng số trường hợp dự đoán sai mẫu âm.

    Giả sử ta có các số liệu như sau: tổng số mẫu khảo sát là 1000 mẫu trong đó có 900 mẫu là thư không spam và 100 mẫu là thư spam. Kết quả của mô hình phân loại cho ra như sau:

| | Real Values | |
|---|---|---|
| Predictions | Positive | Negative |
| Positive | TP = 50 | FP = 50 |
| Negative | FN = 25 | TN = 875 |

  - **Accuracy:** Tỉ lệ của tổng số trường hợp dự đoán đúng trên tổng số trường hợp. Nhìn vào mô hình này Accuracy của mô hình sẽ là (50+875)/1000 = 0.925, tức là mô hình có độ chính xác đến 92.5%. Tuy nhiên đối với trường hợp phát hiện thư spam thì chỉ đúng được phân nửa số thư spam mà trong bài toán phân loại thư điện tử ta lại quan tâm phần nhiều đến việc phát hiện một thư có phải là thư spam hay không hơn, do đó có thể thấy được Accuracy chưa phù hợp để đánh giá tổng thể của mô hình khi mà bộ dữ liệu có hiện

tượng skewed classes và mức quan trọng của các lớp là khác nhau. Lúc này vai trò của F1 score, Precision và Recall sẽ được thể hiện.

- **Precision (Độ chuẩn xác):** Là tỉ lệ của tổng số trường hợp dự đoán đúng mẫu **dương** trên tổng số trường hợp dư đoán là mẫu dương. Precision = TP / (TP + FP). Vậy độ chuẩn xác càng cao thì mô hình sẽ dự đoán càng tốt các mẫu thuộc lớp dương (hay dự đoán càng tốt các trường hợp nhãn xấu).
- **Recall (Độ phủ):** là tỉ lệ của tổng số trường hợp dự đoán đúng mẫu **dương** trên tổng số mẫu dương thật sự của dữ liệu đầu vào. Recall = TP / (TP + FN). Như tên gọi của nó, Recall cho biết mức độ bỏ sót các mẫu thuộc lớp positive của mô hình, nếu Recall càng cao chứng tỏ mô hình bỏ sót rất ít các mẫu thuộc lớp positive, chỉ khác với Precision là chỉ có sự tham gia của các mẫu đã được dự đoán là positive thì ở Recall có sự tham gia trên toàn bộ các mẫu của tập dữ liệu.
- **Trade off (đánh đổi) giữa Precision và Recall:**
  - Trong thực tế một mô hình binary classification lý tưởng phải đều thu được cả Precision và Recall cao, tuy nhiên điều này là rất khó xảy ra.
  - Xét ví dụ trên để tăng Precision thì có một cách đơn giản là tăng 'ngưỡng' để quyết định nhãn lên, khi đó sẽ đảm bảo việc dự đoán đúng các mẫu positive hơn nhưng sẽ làm giảm số lượng TP do đó làm giảm Recall.
  - Nếu nới lỏng ngưỡng để tăng Recall thì sẽ làm tăng nhanh số lượng mẫu được dự đoán là positive do đó lại làm giảm Precision.
  - Sự đánh đổi qua lại này thường xuyên diễn ra trong các bộ dữ liệu thực tế do đó cần một độ đo có thể kết hợp 2 độ đo trên, và đó chính là F1 Score.
- **F1-Score:** F1 Score = (2*precision*recall) / (precision + recall). Điều đặc biệt là F1 score luôn nằm trong khoảng của Precision và Recall. Do đó đối với những trường hợp mà precision và recall quá chênh lệch thì F1 score sẽ cân bằng được cả hai giá trị này và giúp ta đưa ra một đánh giá khách quan hơn.

## 4. Cài đặt mô hình học máy

### a) Tạo tập thuộc tính đầu vào X và tập mục tiêu y từ bộ dữ liệu ban đầu

- Tập X chứa tất cả các cột của bộ dữ liệu trừ cột 'Attrition'.
- Tập y chứa cột 'Attrition' của bộ dữ liệu.

### b) Chia tập dự liệu thành tập huấn luyện và tập kiểm tra

- Mục đích: Bất cứ khi nào chúng ta đào tạo một mô hình học máy, chúng ta không thể đào tạo mô hình đó trên một tập dữ liệu hoặc thậm chí chúng ta đào tạo nó trên một tập dữ liệu duy nhất thì chúng ta sẽ không thể đánh giá hiệu suất của mô hình của mình do tập dữ liệu đã được mô hình học thuộc lòng thì việc đánh giá sẽ luôn đạt tỉ lệ đúng rất cao và có thể lên đến 100% dẫn đến hiện tượng over fitting Không tổng quát khi áp dụng vào dữ liệu thực tế.
- Vì lý do đó, chúng ta chia dữ liệu nguồn của mình thành 2 tập training set và test set, nhiều bài viết cũng đề cập đến việc chia thành 3 tập là training, validation, và test set nhưng trong bài làm này nhóm sẽ chỉ chia thành 2 tập training và tets set.
- Kích thước mỗi tập như sau:
  - Size of Training set = 80% * (Size of Dataset).

- Size of Test set = 20% * (Size of Dataset).

**c) Cài đặt mô hình thông qua pipeline**

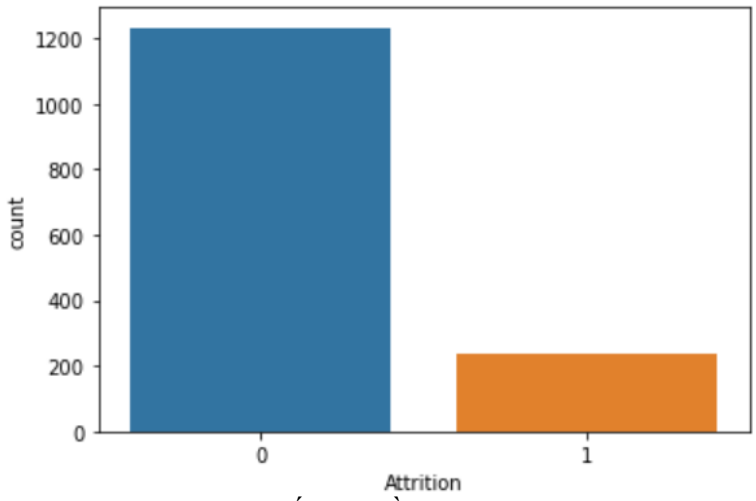
```
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', LogisticRegression(solver="liblinear",penalty="l2",
                                      class_weight={0:1,1:1},max_iter=10000))
])
```

Xây dựng một pipeline thực hiện các bước sau:

- Đầu tiên tạo một thành phần 'scaler' thực hiện việc chuẩn hóa dữ liệu như đã đề cập ở mục cuối của phần tiền xử lý dữ liệu. Thành phần này sử dụng lớp StandardScaler() được cung cấp bởi sklearn để thực hiện chuẩn hóa giá trị của từng cột trong X theo phương pháp Standardisation (Chính quy hóa).
- Tiếp theo là tạo một bộ phân lớp 'classifier' để thực hiện việc phân loại các mẫu đầu vào về một trong hai lớp của bài toán phân loại nhị phân. Bộ phân lớp này áp dụng thuật toán Logistic Regression mà nhóm đã giới thiệu trước đó.
  - 'solver' là tham số dùng để xác định thuật toán để tối ưu việc học của mô hình. Giá trị 'liblinear' sẽ chỉ định sử dụng thuật toán mini-batch gradient descent cho việc học của mô hình.
  - 'penalty' là tham số dùng để xác định công thức cho việc regularization mô hình. Giá trị 'l2' sẽ chỉ định regularization bằng cách cộng một lượng bằng tổng bình phương các tham số $\theta$ (trừ $\theta_0$).
  - 'class_weight' là tham số dùng để điều chỉnh trọng số cho các lớp nếu mất cân bằng dữ liệu xảy ra.
  - 'max_iter' là tham số dùng để chỉ số vòng lặp tối đa cho việc chạy thuật toán tối ưu được chỉ định trong tham số 'solver'.
- **Đánh giá mô hình:**
  - Trước tiên xem xét 'accuracy': Đạt 0.8775510204081632 (~ 87.76%). Accuracy score đạt khá cao nhưng liệu đã đủ tốt cho việc đánh giá mức độ hiệu quả của mô hình?
  - Hãy cùng xem xét thêm về các độ đo khác.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.89 | 0.93 | 267 |
| 1 | 0.41 | 0.74 | 0.53 | 27 |
| accuracy | | | 0.88 | 294 |
| macro avg | 0.69 | 0.82 | 0.73 | 294 |
| weighted avg | 0.92 | 0.88 | 0.89 | 294 |

  - Các độ đo khác như Precision, Recall và F1-score đều rất thấp ở lớp 1 hơn so với ở lớp 0, chứng tỏ mô hình dự đoán chưa tốt cho các trường dự đoán mẫu thuộc thực tế thuộc về lớp 1 (hay nói trong ngữ cảnh này là các mẫu thuộc về lớp tiêu cực).
  - Và điều này làm chúng ta có thể nghĩ đến trường hợp mất cân bằng dữ liệu (imbalanced data).
- **Kiểm tra tính cân bằng của dữ liệu:**




- Rõ ràng xảy ra hiện tượng mất cân bằng giữa hai lớp 0 và 1 khi số lượng các phần tử thuộc lớp 0 trong bộ dữ liệu này nhiều xấp xỉ gấp 6 lần số lượng các phần tử thuộc lớp 1.
- Sự mất cân bằng nghiêm trọng này ảnh hưởng rất lớn đến khả năng dự đoán chính xác của mô hình.

- **Mất cân bằng dữ liệu:**
  - Dữ liệu bị mất cân bằng hiểu sự phân bổ các mẫu trên các lớp chêch lệch nhau quá lớn dẫn tới việc mô hình chỉ tập trung học những đặc trưng của lớp có số lượng mẫu là chiếm đa số.
  - Tác hại: Làm cho việc dự đoán của mô hình xảy ra tình trạng thiên vị, mất đi tính tổng quát cho dữ liệu thực tế sau này.
  - Có nhiều cách xử lý việc mất căn bằng dữ liệu như: Thu thập thêm dữ liệu, các thuật toán tăng mẫu dữ liệu (Oversampling), các thuật toán giảm mẫu dữ liệu (Undersampling), sử dụng trọng số để phạt mô hình cho các giá trị dự đoán của các lớp,....
  - Nhóm sẽ sử dụng 2 phương pháp: sử dụng trọng số để phạt mô hình, Ứng dụng **thuật toán SMOTE** thuộc nhóm Oversampling.
- **Sử dụng trọng số để phạt mô hình:** Là việc thay đổi mức phạt khác nhau khi dự đoán sai các lớp của mô hình. Bằng cách nhân trọng số khác nhau cho các lớp vào cost function.

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} w_i(y_i \log(\hat{y}_i) + (1 - yi)\log(1 - \hat{y}_i))$$

  - w sẽ là vector trọng số tương ứng cho vector y ban đầu với giá trị trọng số là người huấn luyện quyết định trong quá trình huấn luyện. Nhóm đã thực nghiệm với nhiều cặp trọng số và thấy kết quả tốt nhất khi thay đổi trọng số trong bài như sau: các mẫu thuộc lớp 0 có trọng số là 1, các mẫu thuộc lớp 1 có trọng số là 2.
  - Accuracy: 0.8741496598639455.

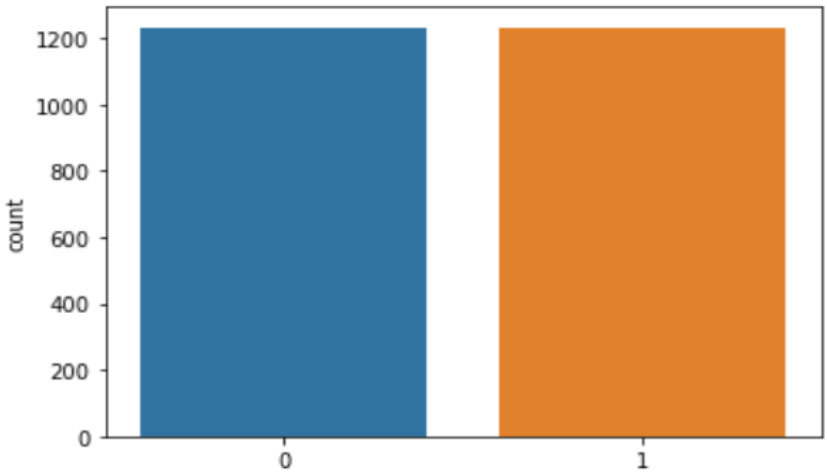
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.92 | 0.92 | 246 |
| 1 | 0.61 | 0.62 | 0.62 | 48 |
| accuracy | | | 0.87 | 294 |
| macro avg | 0.77 | 0.77 | 0.77 | 294 |
| weighted avg | 0.88 | 0.87 | 0.87 | 294 |

- Với việc sử dụng một trọng số lớn hơn cho lớp 1 đã giúp cải thiện precision và f1-score của lớp 1 một cách đáng kể, cho thấy tính tổng quá của mô hình đã được tăng lên.
- Tuy nhiên các độ đo này vẫn chưa đạt được đến giá trị thật sự tốt lắm cho một bài toán phân loại.

- **Sử dụng thuật toán SMOTE:**
  - Ý tưởng chung của SMOTE là tạo ra dữ liệu tổng hợp giữa mỗi mẫu của lớp thiểu số và 'n' hàng xóm gần nhất của nó. Có nghĩa là, đối với mỗi một trong các mẫu của lớp thiểu số, 'n' các láng giềng gần nhất của nó được chọn (n do người cài đặt chỉ định, và mặc định trong thư viện hỗ trợ là 5). Sau đó giữa các cặp điểm được tạo bởi mẫu và từng láng giềng của nó thì ta sẽ có được từng data mới.
  - Chọn gần nhất thường dựa vào khoảng cách Euclid giữa các điểm dữ liệu.
  - Giá trị của điểm dữ liệu mới được tính bằng cách lấy trung bình cộng giữa điểm mẫu và điểm hàng xóm của nó.
  - Thuật toán cứ như vậy duyệt qua hết các điểm mẫu trong lớp thiểu số.

```
from imblearn.over_sampling import SMOTE
X,y=SMOTE(sampling_strategy=1, random_state=0).fit_resample(X, y)
```

  - ‘sampling_strategy’ bằng 1 tức là tạo đến khi nào số lượng mẫu của lớp thiểu số bằng số lượng mẫu của lớp đa số thì dừng lại. Nếu là 0.5 thì đến khi số lượng mẫu của lớp thiểu số bằng phân nửa số lượng mẫu của lớp đa số thì dừng lại.



- Đã không còn lệch dữ liệu thì đặt lại trọng số cho hai lớp là bằng nhau và bằng một.
  Accuracy: 0.9048582995951417.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.98 | 0.84 | 0.91 | 268 |
| 1 | 0.84 | 0.98 | 0.90 | 226 |
| accuracy | | | 0.90 | 494 |
| macro avg | 0.91 | 0.91 | 0.90 | 494 |
| weighted avg | 0.92 | 0.90 | 0.90 | 494 |

- Tất cả các độ đo đều đạt ở mức 90% hoặc hơn, cho thấy sự hiệu quả của thuật toán SMOTE trên bộ dữ liệu này.

### 5. Tổng kết

- Thông qua mô hình học máy nhóm chú trọng vào các khía cạnh:
  - Tiền xử lý dữ liệu.
  - Rút trích đặc trưng.
  - Xây dựng mô hình học máy hiệu quả.
- Nhóm đã phát hiện vấn đề về mất cân bằng dữ liệu xảy ra trên bộ dữ liệu này, một điều gây nhiều khó khăn cho quá trình xây dựng các bộ phân lớp.
- Thực hiện các phương pháp để giải quyết vấn đề mất cân bằng dữ liệu và xây dựng mô hình học máy để sử dụng như bộ phân lớp với các độ đo đạt kết quả rất khả quan:
  - Accuracy score: 90%.

## III. Tài liệu tham khảo


**Tham khảo cách vẽ biểu đồ:**

https://www.kaggle.com/code/hasibalmuzdadid/monkeypox-analysis

https://www.kaggle.com/code/nguyenthicamlai/software-industry-india-analysis

https://www.kaggle.com/code/varunsaikanuri/employee-attrition-analysis

**Tham khảo phần xây dựng học máy:**