**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**Bài tập lớn: Dự đoán tỉ lệ sai thải của nhân viên**

**Mã số sinh viên: 1957012049 – Hoàng Dương**

**2051012085 – Nguyễn Thái Phát**

**Giảng viên hướng dẫn: Nguyễn Văn Bảy**

**TP. HỒ CHÍ MINH, 2023**

**MỤC LỤC**

[Chương 1. Mô tả bài toán và tiền xử lý dữ liệu 4](#_Toc143724079)

[1.1. Mô tả dữ liệu 4](#_Toc143724080)

[- Thuộc tính 4](#_Toc143724081)

[- Tiền xử lý dữ liệu 5](#_Toc143724082)

[- Ma trận tương quan (correlation matrix) 8](#_Toc143724083)

[- Phân tích khám phá dữ liệu (Exploratory Data Analysis EDA) 9](#_Toc143724084)

[- Mã hóa dữ liệu 11](#_Toc143724085)

[Chương 2. Bài toán Unsupervised Machine Learning 11](#_Toc143724086)

[2.1. Tổng quan về SSE và Phương pháp Elbow 11](#_Toc143724087)

[2.2. Thực hiện gom cụm 12](#_Toc143724088)

[- Kết quả gom cụm 13](#_Toc143724089)

[Chương 3. Bài toán Supervised Machine Learning 13](#_Toc143724090)

[3.1. Yêu cầu bài toán 13](#_Toc143724091)

[3.2. Quá trình rời rạc hóa 13](#_Toc143724092)

[3.3. Tiến hành training dữ liệu bằng KNN 16](#_Toc143724093)

[- Đọc và quan sát dữ liệu 16](#_Toc143724094)

[- Training và thử nghiệm tập test bằng thuật toán KNN 16](#_Toc143724095)

[- Thực hiện lại với bins = 3 19](#_Toc143724096)

[- Đánh giá kết quả và lựa chọn. 20](#_Toc143724097)

[- Đánh giá thêm về mô hình bằng Cross-Validation 21](#_Toc143724098)

[3.4. Tiến hành training dữ liệu và phân lớp bằng Naïve-Bayes 23](#_Toc143724099)

[- Kết quả: 23](#_Toc143724100)

[- Nhận xét: 25](#_Toc143724101)

[3.5. Tiến hành training và phân lớp bằng cây quyết định 26](#_Toc143724102)

[- Sử dụng cây quyết định 26](#_Toc143724103)

[- Kết quả: 28](#_Toc143724104)

[- Nhận xét 29](#_Toc143724105)

[- Tìm tham số tốt nhất 29](#_Toc143724106)

[- Một số tham số quan trọng: 29](#_Toc143724107)

[- Thông tin thêm đối với bài toán có nhiều hơn 2 lớp, cần phải có quy tắc tính trung bình. 30](#_Toc143724108)

[TÀI LIỆU THAM KHẢO 32](#_Toc143724109)

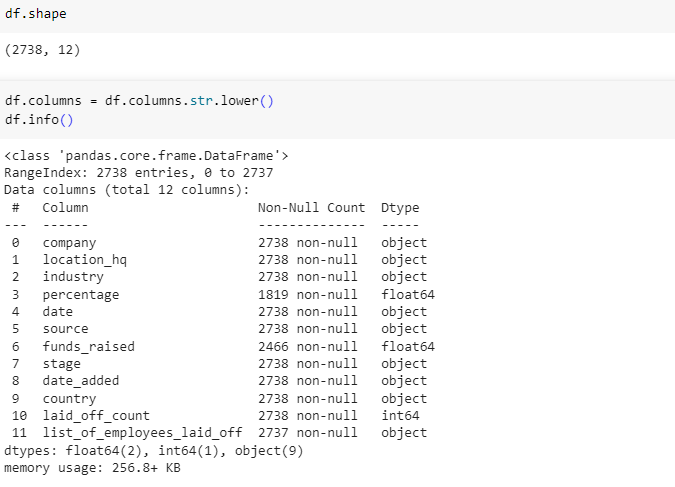
# Mô tả bài toán và tiền xử lý dữ liệu

Bộ dữ liệu layoff theo dõi tình trạng sa thải nhân viên của các công ty trên thế giới.

Tên file: layoff.csv

Số records: 2738

Số thuộc tính: 12



## Mô tả dữ liệu

### Thuộc tính

Các thuộc tính bao gồm:

Company: Tên của công ty

Location\_HQ: Trụ sở chính của công ty (Location headquarters).

Industry: Lĩnh vực hoạt động của công ty (VD: Thực phẩm, sức khỏe, …).

Percentage : Tổng phần trăm nhân viên bị sa thải trong công ty.

Date : Ngày công ty thực hiện sa thải.

Source : Nguồn thu thập dữ liệu

Funds\_Raised (mm) : Tổng số tiền gây quỹ mà một tổ chức, doanh nghiệp, dự án hoặc cá nhân đã thu thập được từ nhiều nguồn khác nhau (đơn vị là million).

Stage : Giai đoạn. Mọi tổ chức và doanh nghiệp đều có giai đoạn. Các công ty có thể được phân thành các giai đoạn khác nhau **có thứ tự** như Seed Stage, Series-A, Series-B, Initial Public Offering (IPO), Private Equity, Subsidiary, Acquired.

Date\_Added: Ngày dữ liệu được thêm vào dataset.

Country: Quốc gia mà công ty (Cơ sở chính hoặc phụ) này thực hiện sa thải nhân viên.

Laid\_Off\_Count: Tổng số lượng layoff (sa thải).

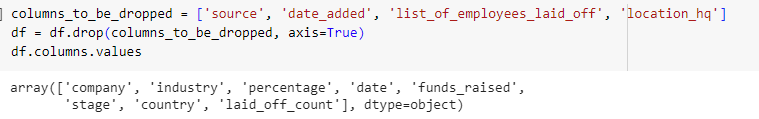
List\_of\_Employees\_Laid\_Off: Danh sách nhân viên bị sa thải.

### Tiền xử lý dữ liệu

* Lọc bỏ thuộc tính dư thừa.

Dựa trên mô tả thuộc tính, ta nhận thấy source, date\_added và list\_of\_employees\_laid\_off không cần thiết trong việc phục vụ bài toán sắp tới của đề tài. Ngoài ra thuộc tính location\_hq chỉ định vị trí công ty đặt trụ sở chính, không phải nơi xảy ra layoff.

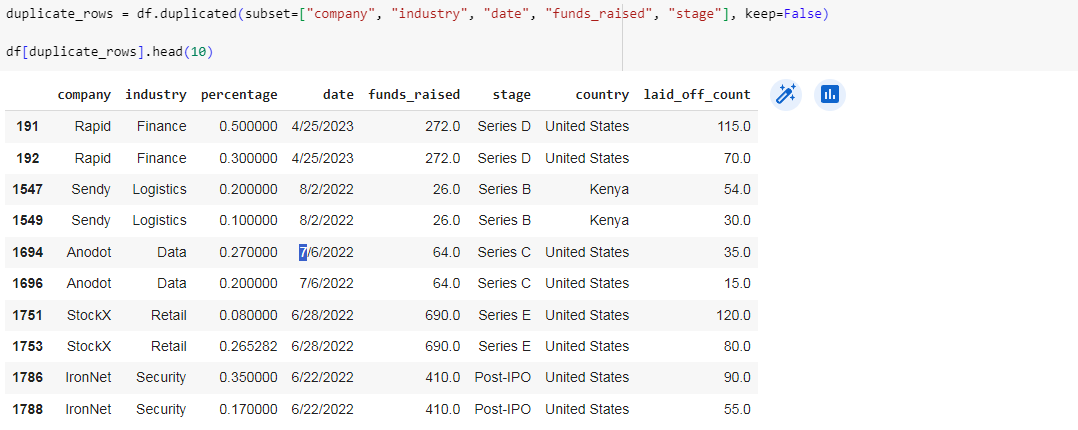
Thực hiện loại bỏ 4 thuộc tính nói trên



Lúc này chỉ còn 8 thuộc tính: company, industry, percentage, date, funds\_raised, stage, country, laid\_off\_count.

* Xoá dữ liệu trùng lắp.

Thực hiện xóa nếu trùng thông tin company, industry, date, funds\_raised và stage.





* Chuyển đổi dữ liệu thuộc tính

Với thuộc tính date có dạng **object**(mm/dd/yyyy), ta thực hiện phân tách thành 3 cột tương ứng: day, month, year dạng **numberic**. 

Với thuộc tính laid\_off\_count, percentage, fund\_raised dạng numberic có giá trị null, thực hiện thay thế null thành các giá trị **trung bình**, **trung vị** hoặc **mode**.

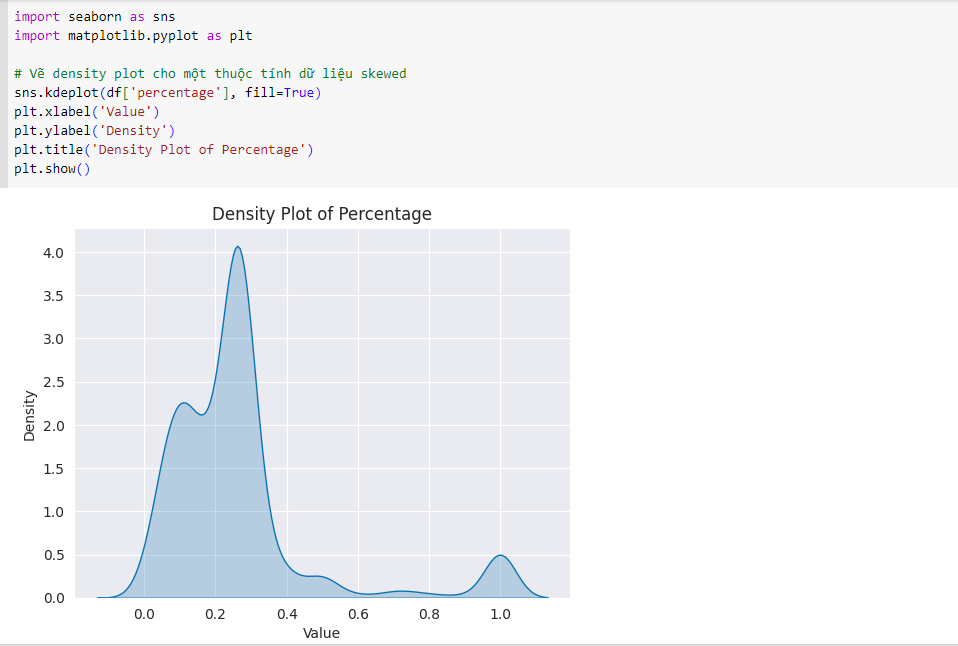
*Vấn đề:* Nên lựa chọn giá trị nàolàm giá trị thay thế cho missing value. [3]

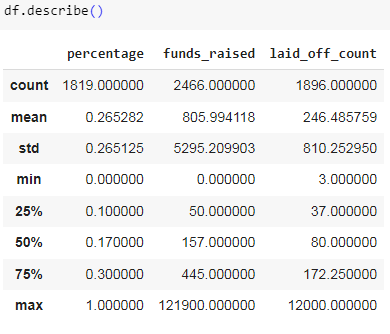
**Trung vị (median)**: Nếu bạn chọn sử dụng trung vị, điều này thường thích hợp khi dữ liệu bị lệch hoặc có nhiễu.

**Trung bình (mean)**: Sử dụng giá trị trung bình thường là lựa chọn có thể sử dụng khi dữ liệu có phân phối là số và không bị skewed.

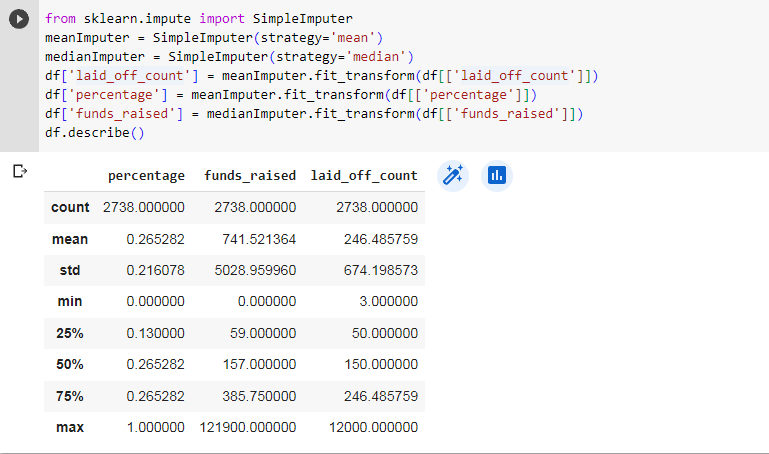
**Mode**: Sử dụng cho dữ liệu số và object.

Tuy nhiên cũng phải xem xét những yếu tố thực tế và sự hiểu biết về dữ liệu để lựa chọn cách thực hiện.

 Mô tả trước khi thay thế giá trị bị thiếu:



Mô tả sau khi thực hiện thay thế giá trị bị thiếu:

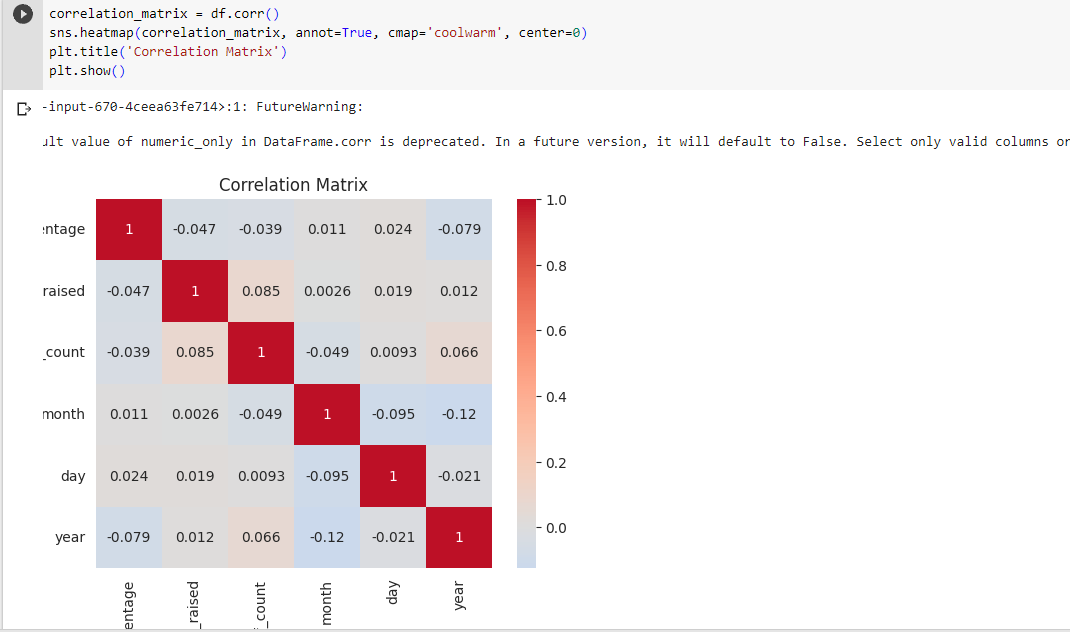


### Ma trận tương quan (correlation matrix)

Là một ma trận số liệu thể hiện mối quan hệ tương quan giữa các cặp thuộc tính trong tập dữ liệu. Giá trị trong ma trận tương quan nằm trong khoảng từ -1 đến 1 và thể hiện mức độ tương quan giữa hai thuộc tính với nhau.

* **Giá trị tương quan dương**: Nếu giá trị tương quan gần 1, điều này thể hiện mối quan hệ tương quan thuận giữa hai thuộc tính. Khi một thuộc tính tăng, thuộc tính còn lại cũng tăng và ngược lại.
* **Giá trị tương quan âm**: Nếu giá trị tương quan gần -1, thì có mối quan hệ tương quan nghịch giữa hai thuộc tính. Khi một thuộc tính tăng, thuộc tính còn lại giảm và ngược lại.
* **Giá trị tương quan gần 0**: Một giá trị tương quan gần 0 thể hiện rằng không có mối quan hệ tương quan rõ ràng giữa các thuộc tính.

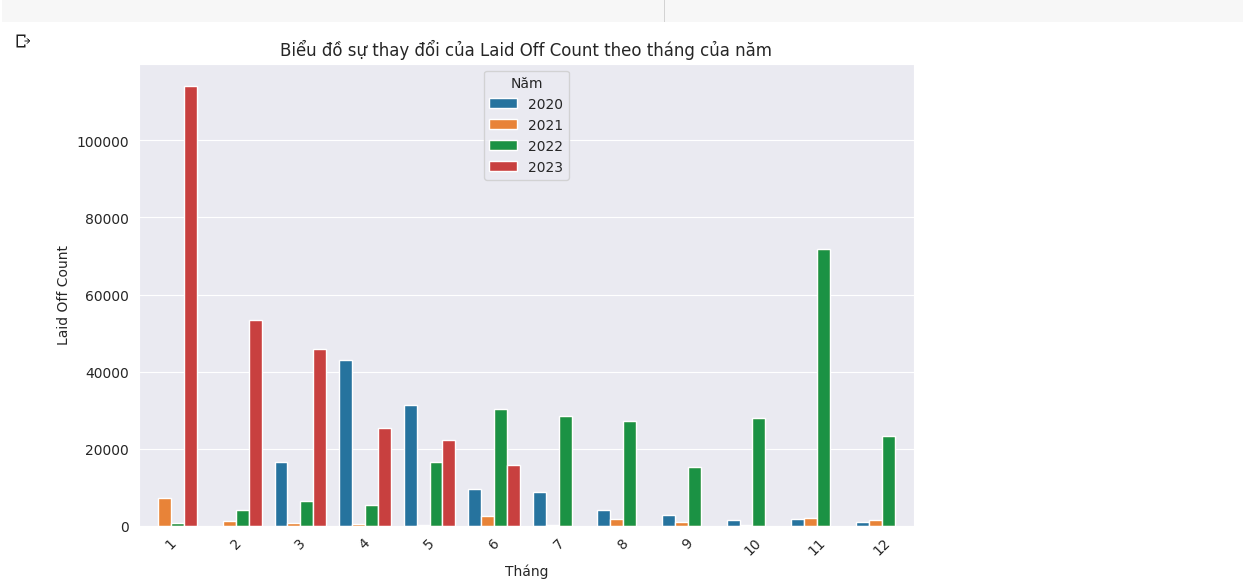
Ma trận tương quan cung cấp thông tin về mối quan hệ tương quan cao giữa các cặp thuộc tính. Từ đó quyết định về việc giữ lại, loại bỏ hoặc chuyển đổi dữ liệu các thuộc tính. Tuy nhiên, việc thực hiện cũng phụ thuộc vào ngữ cảnh dữ liệu và sự hiểu biết về mô hình máy học.



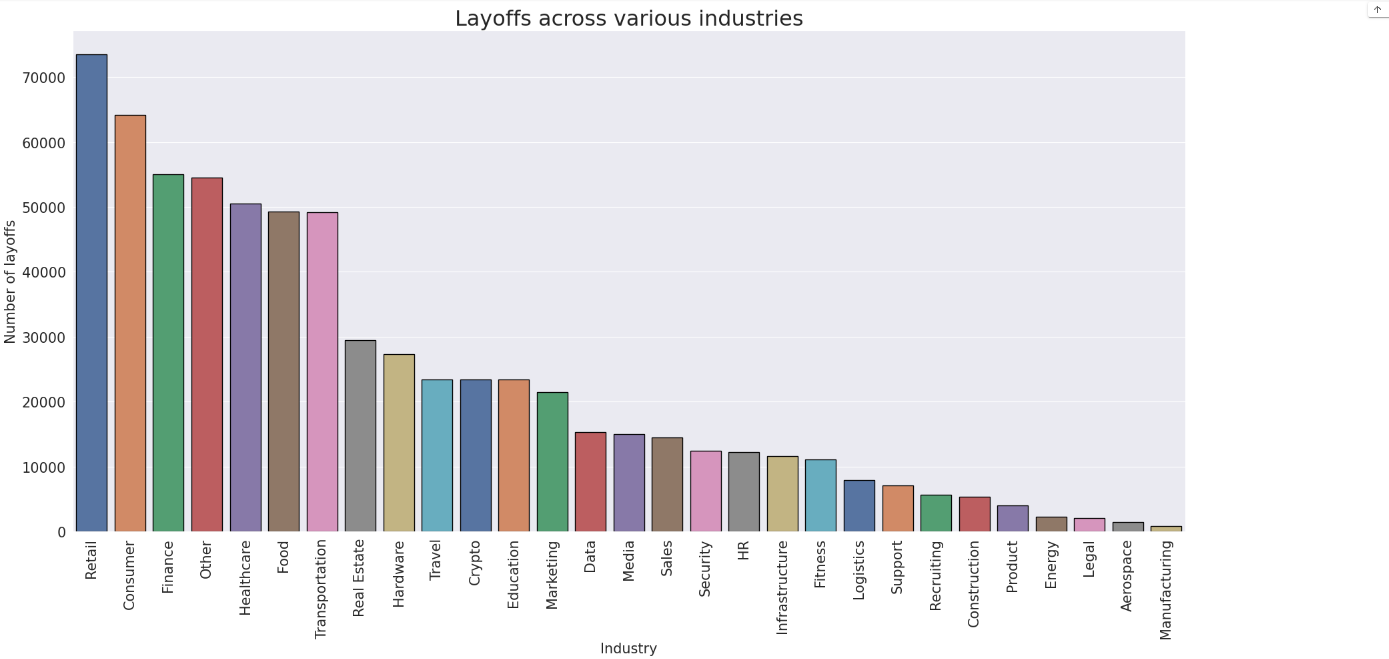
Từ bảng Ma trận tương quan ở trên, ta nhận thấy các thuộc tính không có quá nhiều mối quan hệ với nhau nên em quyết định không chuẩn hóa các dữ liệu.

### Phân tích khám phá dữ liệu (Exploratory Data Analysis EDA)

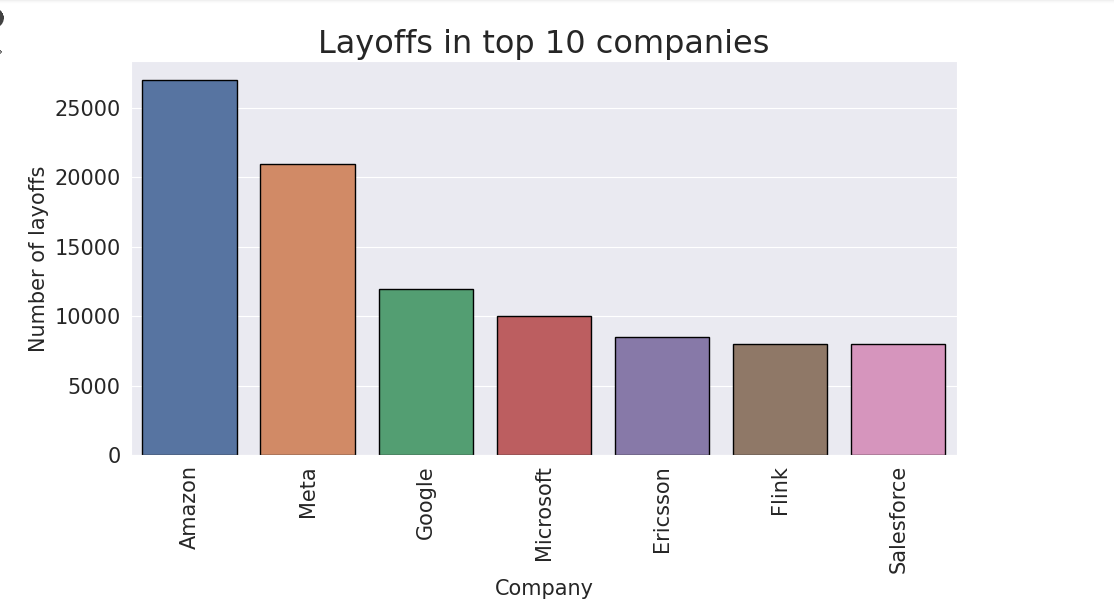
* Biểu đồ về sự thay đổi Laid off count theo tháng của năm



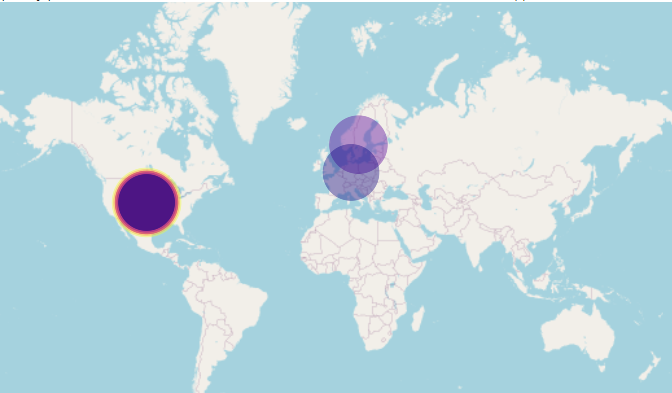
* Biểu đồ về laid off count theo ngành công nghiệp



* Top 10 công ty thực hiện sa thải nhân viên trong tập dữ liệu



* Vị trí theo country của 10 công ty trên



### Mã hóa dữ liệu

Thuộc tính Stage (giai đoạn của công ty) là thuộc tính nominal **có thứ tự**. Thực hiện mã hóa dạng Ordinal Encoding.

Thuộc tính Industry và country là thuộc tính nominal **không có thứ tự**. Có thể thực hiện Label encode và One-hot encode [2].

* + **Label encode :** khi thực hiện sẽ vô tình khiến xảy ra nhầm lẫn về thứ tự cho giá trị. Do label sẽ đánh số cho mỗi giá trị duy nhất. Tuy nhiên sẽ giúp tránh việc tạo ra nhiều cột, giảm chiều dữ liệu.
  + **One-Hot Encoding:** One-Hot Encoding chuyển đổi các danh mục thành các biến nhị phân (0 hoặc 1), tạo ra một cột mới cho mỗi giá trị duy nhất trong cột gốc. One-Hot Encoding không ánh xạ thông tin về thứ tự, vì tất cả các cột mới là độc lập với nhau.

# Bài toán Unsupervised Machine Learning

## Tổng quan về SSE và Phương pháp Elbow

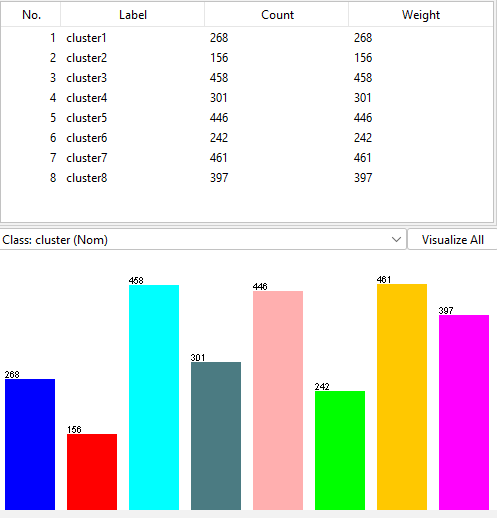
**Sum of Squared Errors (SSE)** là một phép đo độ biến động, tức là độ lệch giữa các điểm dữ liệu và trung tâm của cụm nơi chúng thuộc về trong thuật toán K-Means. SSE tính toán tổng của bình phương khoảng cách giữa mỗi điểm dữ liệu và trung tâm cụm của nó.

Bước cơ bản đối với bất kỳ thuật toán không giám sát nào là xác định số lượng các cụm tối ưu mà dữ liệu có thể được gom vào. Trong đó Phương pháp **Elbow** là một trong những phương pháp phổ biến nhất để xác định giá trị tối ưu của k. Qua đó lựa chọn được số lượng các cụm phù hợp dựa vào đồ thị trực quan hoá bằng cách nhìn vào sự suy giảm của *hàm biến dạng* và lựa chọn ra điểm *khuỷ tay* (*elbow point*). Ngoài Elbow ra chúng ta còn có thể kể đến là **Silhouette**

## Thực hiện gom cụm

Qua biểu đổ Elbow, sử dụng thuật toán k-mean để tính toán SSE, chúng em quyết định lựa chọn k = 8 cho bài toán này.

### Kết quả gom cụm



# Bài toán Supervised Machine Learning

Có 2 dạng unsupervised machine learning bao gồm phân lớp và hồi quy. Trong đề tài này chúng em chọn thực hiện phân lớp.

## Yêu cầu bài toán

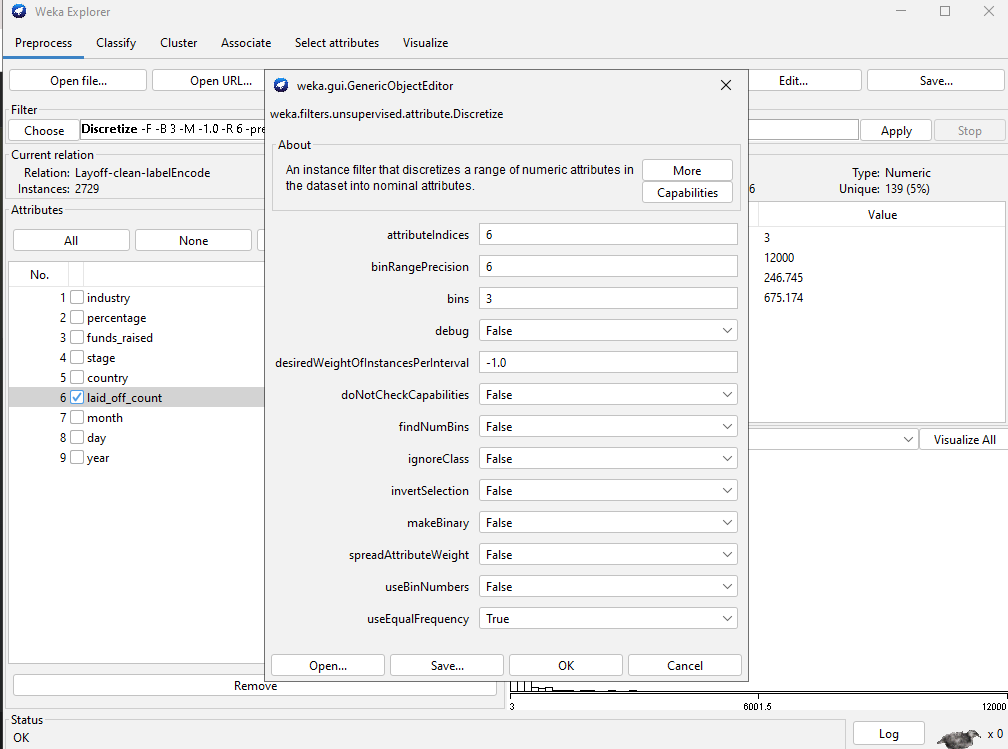
Dựa vào các thuộc tính **industry**, **percentage**, **funds\_raised**, **stage**, **country**, **month**, **day**, **year** xây dựng mô hình máy học dự đoán tỉ lệ **laid\_off\_count**.

Trước đó em thấy **laid\_off\_count** là thuộc tính numberic, nên chúng ta cần rời rạc hóa thuộc tính đó thành nominal **theo thứ tự**.

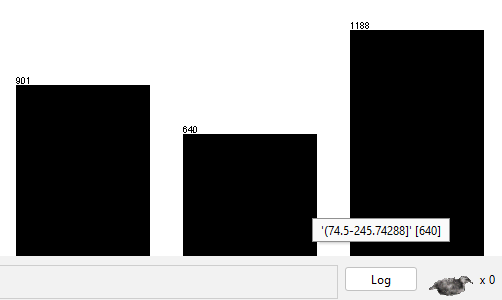
## Quá trình rời rạc hóa

Quá trình được thực hiện trong weka, với các tham số:

* bins = n để chia thành n khoảng giá trị tương ứng
* useEqualFrequency: Để đảm bảo mỗi bins có số lượng phần tử gần với nhau. Tránh việc có một bins có số lượng chiếm đa số.

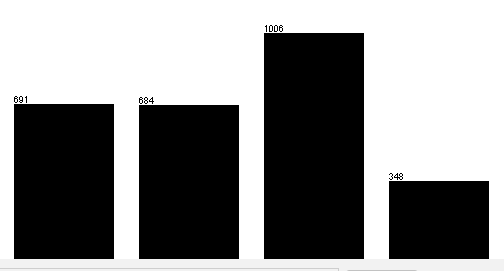


* Với bins = 3



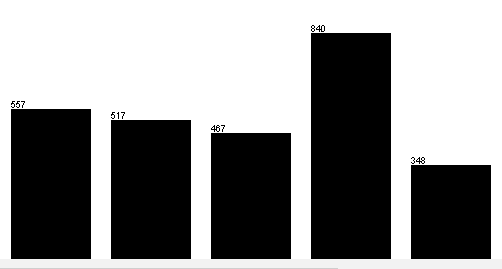
Nhận xét: Giá trị các bins trông có vẻ không đồng đều nhưng có thể chấp nhận được. Nhìn hình ta thấy rằng bin thứ 3 có giá trị gấp hai lần bin thứ 2.

* Với bins = 4



Nhận xét: Bin thứ 4 có số lượng khá ít so với các bin còn lại.

* Với bins = 5

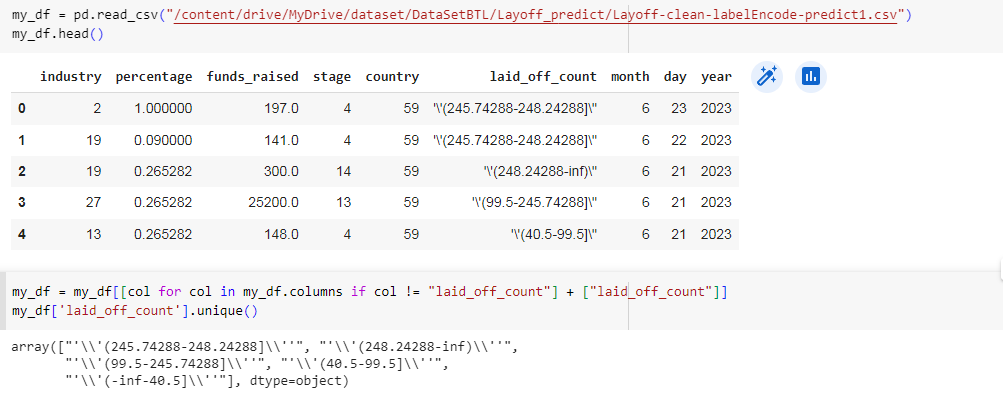


Nhận xét: Bin thứ 5 có số lượng ít nhưng chênh không quá nhiều với các bins gần số lượng với chính nó.

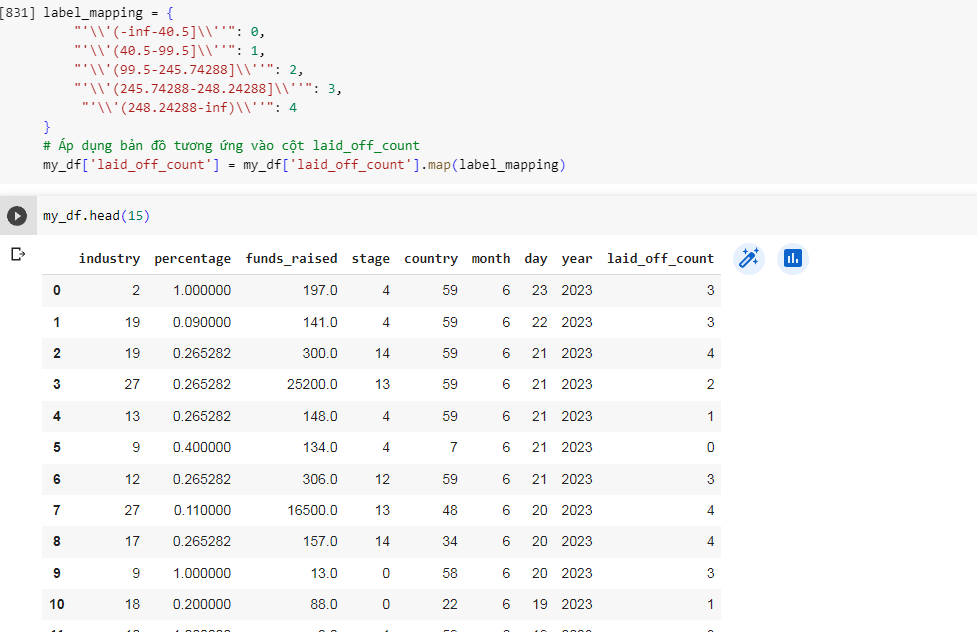
=> Lựa chọn tham số bins = 5 cho bài toán

## Tiến hành training dữ liệu bằng KNN

### Đọc và quan sát dữ liệu



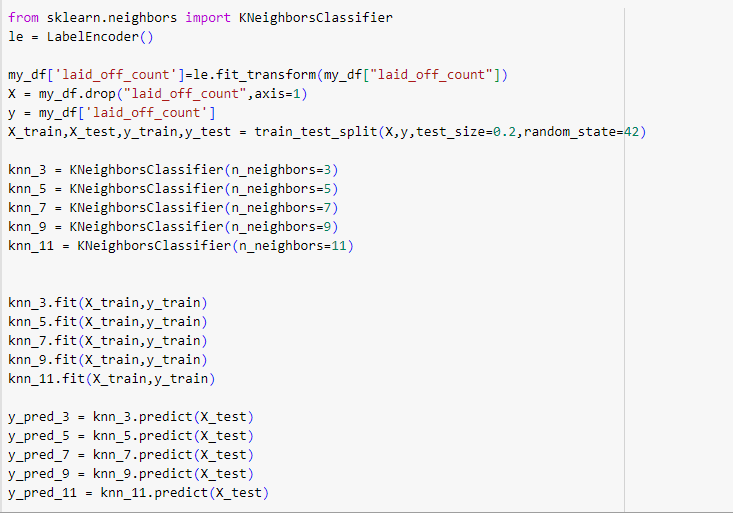
Chuyển đổi thành numberic có thứ tự tương ứng từ thấp đến cao (0-4)



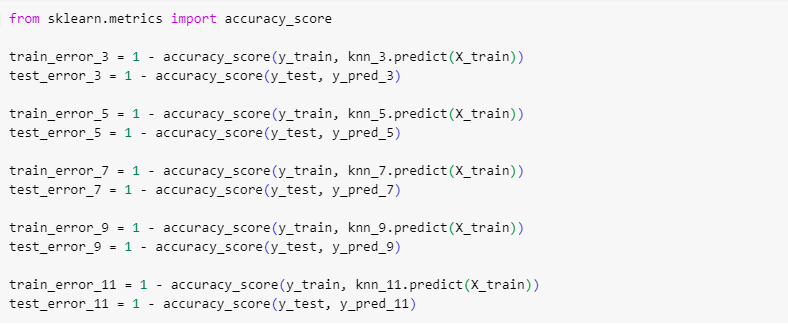
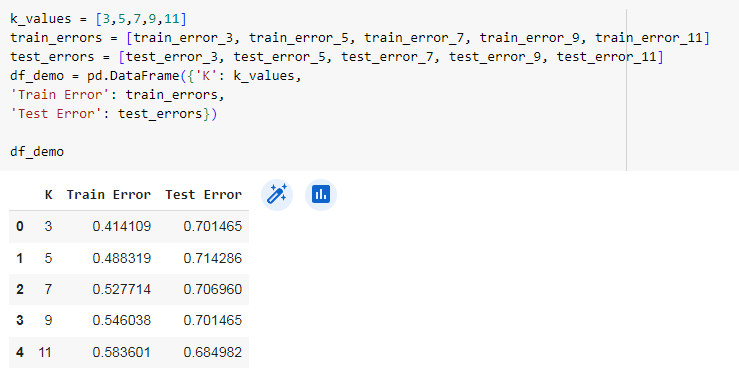
Giá trị của cột laid\_off\_count lúc này là numberic với 5 giá trị.

### Training và thử nghiệm tập test bằng thuật toán KNN

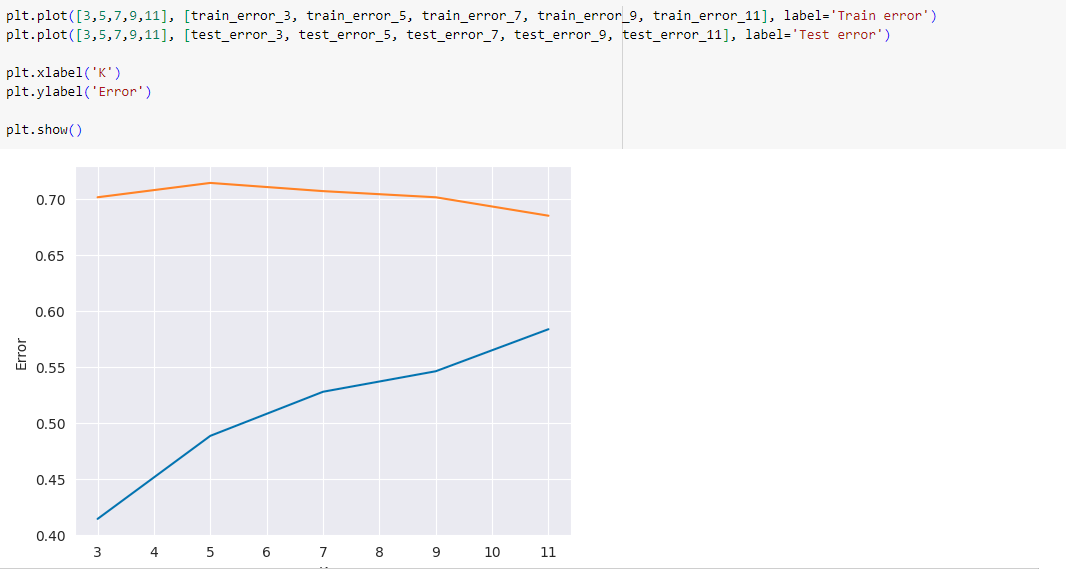
Khởi tạo training và thực hiện



Kết quả:

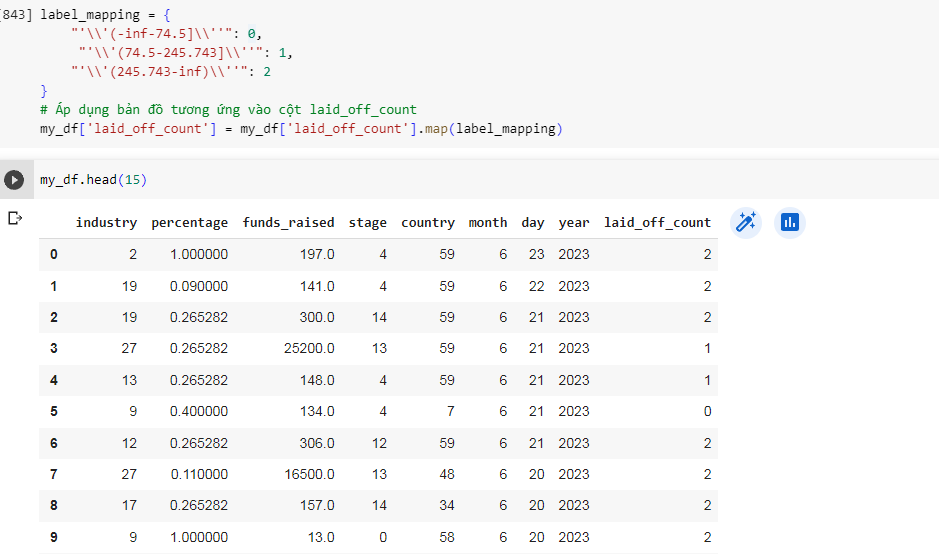
 

Biểu đồ miêu tả tỉ lệ lỗi trong KNN với k = 3, 5, 7, 9, 11

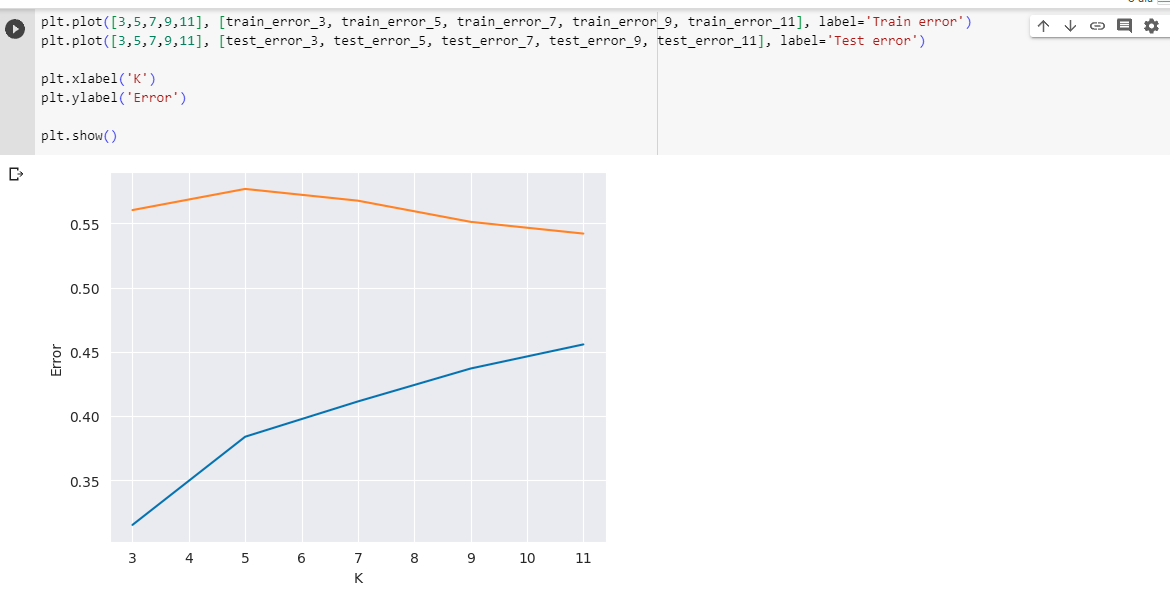


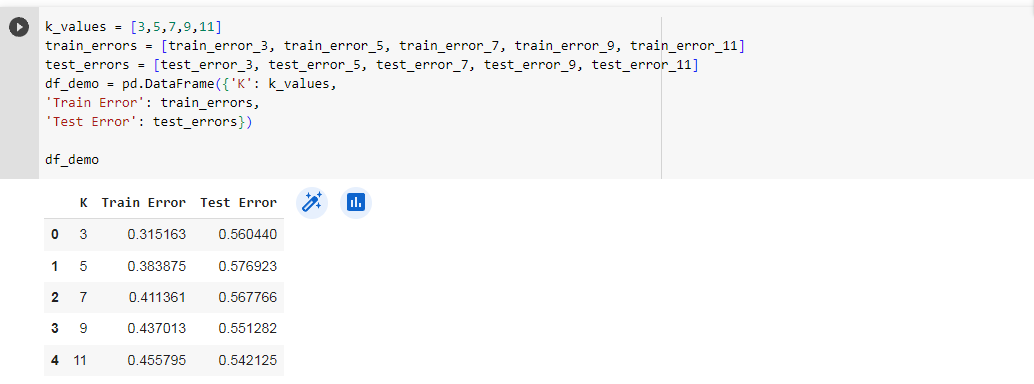
Nhận xét: Thông qua KNN với số k = 3,5,7,9,11. Có vẻ như tỉ lệ dự đoán sai cao ở cả tập test và tập train khá cao.

### Thực hiện lại với bins = 3



Biểu đồ miêu tả tỉ lệ lỗi trong KNN với k = 3, 5, 7, 9, 11

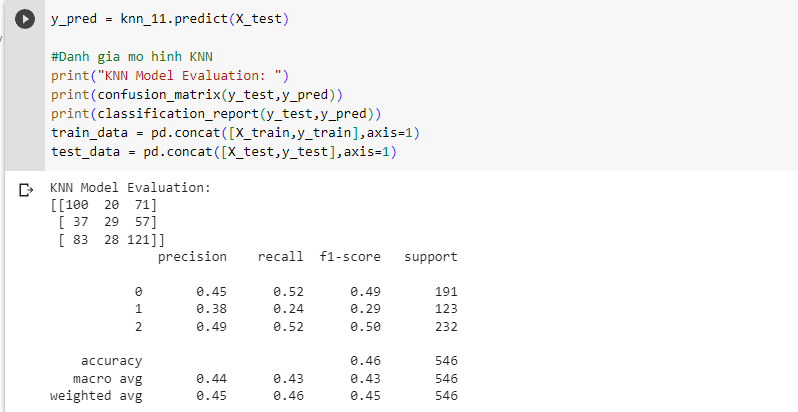




Nhận xét: Ta thấy kết quả cho thấy độ chính xác đã tăng lên và tỉ lệ dự đoán lỗi giảm xuống. Tuy nhiên vẫn còn ở trên mức 50%.

### Đánh giá kết quả và lựa chọn.

Sau 2 lần thử nghiệm với 3 bins và 5 bins cũng như quan sát lỗi khi thay đổi số k trong thuật toán KNN. Nhóm quyết định chọn chia thuộc tính laid\_off\_count thành 3 bins và lựa chọn k = 11 cho thuật toán KNN. Kết quả của việc đánh giá mô hình:



Với các giá trị:

**Recall (Độ phục hồi):** Là tỷ lệ giữa số lượng dự đoán đúng cho một lớp cụ thể và tổng số điểm thực sự thuộc lớp đó (TP / (TP + FN)). Recall đo lường khả năng của mô hình dự đoán đúng những điểm thực sự thuộc lớp đó.

**Support:** Số lượng điểm dữ liệu thực sự thuộc lớp đó trong tập kiểm tra.

**Precision (Độ chính xác):** Là tỷ lệ giữa số lượng dự đoán đúng cho một lớp cụ thể và tổng số dự đoán cho lớp đó (TP / (TP + FP)). Precision đo lường khả năng của mô hình dự đoán đúng những điểm thực sự thuộc lớp đó.

**Accuracy (Độ chính xác toàn bộ):** Là tỷ lệ giữa số lượng dự đoán đúng trên tổng số dự đoán (TP + TN) / Tổng số điểm. Accuracy đo lường khả năng của mô hình dự đoán đúng toàn bộ dữ liệu.

**Nhận xét:**

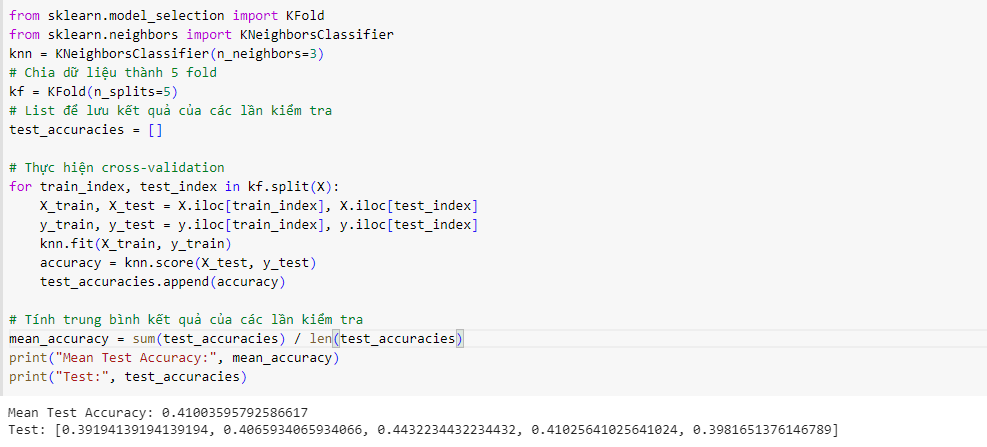
* Precision, Recall và F1-score của từng lớp được thể hiện trong bảng. Precision và Recall có giá trị trung bình tương đối cân đối, nhưng không đạt mức cao.
* Lớp 0 (Class 0):
* Precision: 0.45
* Recall: 0.52
* Precision khá gần với Recall, điều này cho thấy mô hình có khả năng dự đoán đúng những điểm thực sự thuộc lớp 0 với mức độ tương đối cao.
* Lớp 1 (Class 1):
* Precision: 0.38
* Recall: 0.24
* Precision thấp hơn Recall, có thể cho thấy mô hình dự đoán đúng một phần nhỏ các điểm thực sự thuộc lớp 1, nhưng không nhiều. Điều này có thể là do dữ liệu lớp 1 có sự phân bố không đều hoặc có tính chất khó phân loại.
* Lớp 2 (Class 2):
* Precision: 0.49
* Recall: 0.52
* Precision gần với Recall, cho thấy mô hình có khả năng dự đoán đúng những điểm thực sự thuộc lớp 2 ở mức độ khá cao.
* Accuracy của mô hình là khoảng 46%, có nghĩa là mô hình dự đoán đúng khoảng 46% tổng số điểm dữ liệu trong tập kiểm tra. Điều này chỉ ra rằng mô hình chưa có khả năng phân lớp tốt trên tập dữ liệu này.

**Kết luận:**

Tổng quan, mô hình cần cải thiện để có hiệu suất tốt hơn trên tập dữ liệu kiểm tra. Nên cân nhắc điều chỉnh tham số, thử nghiệm các biến thể của mô hình, tăng cường dữ liệu, hoặc thực hiện các biện pháp tiền xử lý khác để cải thiện kết quả.

### Đánh giá thêm về mô hình bằng Cross-Validation

Cross-validation là một kỹ thuật đánh giá mô hình bằng cách chia tập dữ liệu thành nhiều fold (phần chia) và thực hiện kiểm tra trên từng fold. Vì các fold này được chia ngẫu nhiên và thực hiện từng phần nên có thể giúp đảm bảo khả năng tổng quát hóa của mô hình trên các tập dữ liệu mới và kiểm tra giá trị dự đoán chính xác của nó cũng như tạo ra độ tin cậy cao hơn về khả năng dự đoán của mô hình.



**Kết quả Cross-Validation:**

* Mean Test Accuracy: 0.4100 (hoặc khoảng 41.00%)
* Kết quả Accuracy của từng lần kiểm tra (Test): [0.3919, 0.4066, 0.4432, 0.4103, 0.3982]

=> Kết quả Mean Test Accuracy cho thấy mô hình trên trung bình có độ chính xác khoảng 41%, tức là mức độ dự đoán đúng của mô hình trên các fold.

## Tiến hành training dữ liệu và phân lớp bằng Naïve-Bayes

#### nb\_model: Đây là biến mà bạn sử dụng để lưu mô hình Gaussian Naive Bayes sau khi bạn đã tạo và huấn luyện nó.

#### GaussianNB(): Đây là cách bạn tạo một đối tượng mô hình Gaussian Naive Bayes. Bạn có thể thay thế nó bằng các biến thể khác của mô hình Naive Bayes tùy thuộc vào loại dữ liệu của bạn (ví dụ: MultinomialNB() cho dữ liệu văn bản đa dạng).

#### nb\_model.fit(X\_train, y\_train): Đây là nơi bạn huấn luyện mô hình Gaussian Naive Bayes trên tập dữ liệu huấn luyện (X\_train, y\_train). Mô hình sẽ học cách dự đoán nhãn dựa trên các đặc trưng trong dữ liệu huấn luyện.

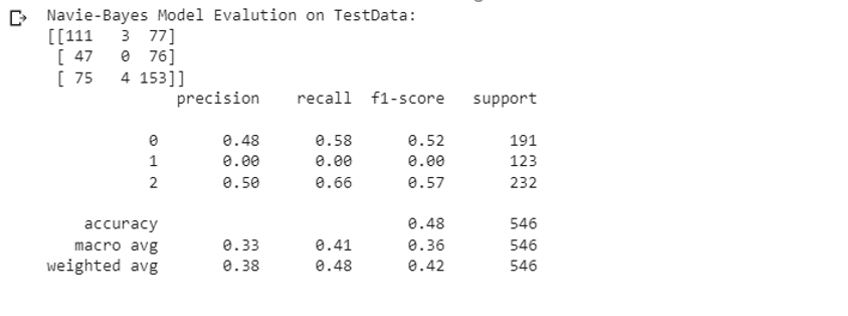
#### 

#### y\_pred\_nb = nb\_model.predict(X\_test): Đây là nơi bạn sử dụng mô hình Naive Bayes đã huấn luyện để dự đoán nhãn trên tập kiểm tra (X\_test). Kết quả dự đoán được lưu vào biến y\_pred\_nb.

#### print(confusion\_matrix(y\_test, y\_pred\_nb)): Đoạn mã này sử dụng hàm confusion\_matrix từ thư viện Scikit-Learn để tính và in ra ma trận nhầm lẫn. Ma trận này sẽ cho biết mô hình dự đoán đúng và sai các lớp nhãn như thế nào.

#### print(classification\_report(y\_test, y\_pred\_nb)): Đoạn mã này sử dụng hàm classification\_report từ thư viện Scikit-Learn để tính và in ra báo cáo phân loại. Báo cáo này bao gồm các chỉ số như độ chính xác, recall, precision và f1-score cho từng lớp nhãn.

### Kết quả:



### Nhận xét:

#### Precision, Recall và F1-score của từng lớp được thể hiện trong bảng. Tuy nhiên có thể thấy ở class 1 không có điểm nào thuộc nhóm 1 mà chỉ tập trung ở nhóm 0 và nhóm 2. Precision có mức trung bình tương đối nhưng chỉ chạm mức trung bình, nhưng recall thì vượt mức trung bình, đặc biệt là recall thuộc nhóm 2.

#### Lớp 0 (Class 0):

#### Precision: 0.48

#### Recall: 0.58

#### Precision khá gần với Recall, điều này cho thấy mô hình có khả năng dự đoán đúng những điểm thực sự thuộc lớp 0 với mức độ tương đối cao.

#### Lớp 1 (Class 1):

#### Precision: 0.00

#### Recall: 0.00

#### Precision bằng Recall, nhưng cả 2 thông số đều bằng 0.00 cho ta thấy Precision thấp đồng nghĩa độ chính xác của các mẫu đúng là thấp. Recall thấp đồng nghĩa với việc bỏ sót các mẫu thực sự thuộc nhóm 1 là cao.

#### Lớp 2 (Class 2):

#### Precision: 0.50

#### Recall: 0.66

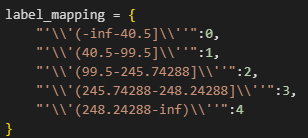
#### Precision gần với Recall, cho thấy mô hình có khả năng dự đoán đúng những điểm đúng và thực sự thuộc lớp 2 ở mức độ khá cao.

#### Accuracy của mô hình là khoảng 48%, có nghĩa là mô hình dự đoán đúng khoảng 48% tổng số điểm dữ liệu trong tập kiểm tra. Điều này chỉ ra rằng mô hình chưa có khả năng phân lớp tốt trên tập dữ liệu này.

## Tiến hành training và phân lớp bằng cây quyết định

### Sử dụng cây quyết định

#### Đầu tiên ta tiến hành sử dụng label\_mapping như ở trên để dán nhãn cho cột Laid\_off\_count theo thứ tự từ 0 tới 4



#### Từ các nhãn ở trên ta tách dữ liệu thành 2 vùng là feature và label với feature là bao gồm các cột trừ cột laid\_off\_count. Label bao gồm laid\_off\_count



#### Tiếp đó ta tiến hành chia dữ liệu ra các tập train và test để tiến hành huấn luyện dữ liệu

#### Ta chọn max\_dept = 5 để giới hạn cây chỉ có tối đa là 5 tầng từ nhánh tới gốc(tránh việc cây bị quá nhiều nhánh). Sử dụng DecisionTreeClassifier để tạo cây và fit với các tập X\_train và y\_train

#### dot\_data: Biến này sẽ chứa đối tượng DOT data của cây quyết định, sẽ được sử dụng để tạo đồ thị.

#### export\_graphviz(clf, out\_file=None, feature\_names=X.columns.tolist(), class\_names=[str(c) for c in clf.classes\_], filled=True, rounded=True, special\_characters=True): Đây là cuộc gọi hàm để tạo DOT data từ mô hình cây quyết định.

#### clf: Mô hình cây quyết định đã được huấn luyện.

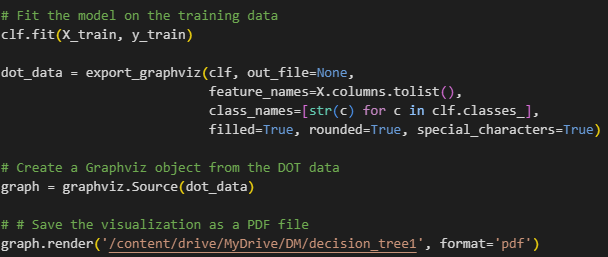
#### out\_file=None: Tham số này xác định tập tin đầu ra để lưu DOT data. Trong trường hợp này, bạn không muốn lưu thành tập tin nên đặt là None.

#### feature\_names=X.columns.tolist(): Tên của các đặc trưng trong tập dữ liệu.

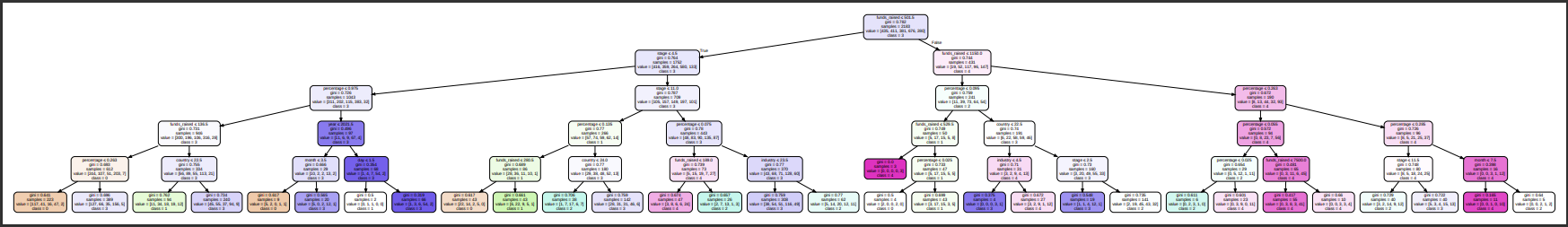
#### class\_names=[str(c) for c in clf.classes\_]: Tên của các lớp (classes) trong dữ liệu. Ở đây, bạn đang sử dụng str(c) để đảm bảo rằng các tên lớp được chuyển thành chuỗi.

#### filled=True, rounded=True, special\_characters=True: Các tham số này định dạng đồ thị bằng cách tạo các màu sắc cho các nhóm, làm tròn các góc, và xử lý các ký tự đặc biệt.

#### Cuối cùng ta dùng thư viện graphviz để tạo đồ thị và lưu ở dạng PDF



### Kết quả:

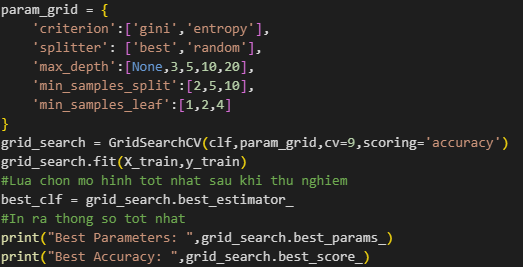


### Nhận xét

#### Cây được phân nhánh theo từng khoảng giá trị của từng thuộc tính và được chứa trong các class Laid\_off\_count, các khoảng giá trị bắt đầu từ funds\_raised cho ta thấy rõ sự phân hoá của các giá trị theo funds\_raised từ thấp tới cao với thấp nằm bên phải và cao nằm bên trái. Các nhánh sau khi được lọc bởi các thuộc tính cho ra số nhánh(tầng) tối đa là 5(như ta tìm hiểu ở phần dưới thì số nhánh tối đa có thể tạo tốt nhất là 5)

### Tìm tham số tốt nhất

#### Ngoài ra, ta còn có thể tìm kiếm tham số tốt nhất cho mô hình cây bằng việc sử dụng grid\_search



#### Kết quả:



#### Nhận xét: Dựa trên tham số tốt nhất ta có thể thấy max\_dept = 5 rất thích hợp để vẽ cây, ảnh hưởng tới việc dùng cây quyết định phía trên.

### Một số tham số quan trọng:

1. Confidence Factor (CF): Đây là ngưỡng độ tin cậy dùng để cắt tỉa cây. Một nút sẽ không được chia nếu mức độ tin cậy tại nút đó nhỏ hơn ngưỡng này. Giá trị mặc định thường là 0.25.
2. Min Num Obj (Minimum Number of Objects): Đây là số lượng tối thiểu các mẫu cần có trong mỗi nút để tiếp tục chia. Nếu số lượng mẫu thấp hơn ngưỡng này, nút sẽ không được chia. Điều này giúp tránh việc cây bị quá tập trung và quá phức tạp. Giá trị mặc định thường là 2.
3. Unpruned: Tham số này xác định xem cây có được tỉa cắt sau khi được xây dựng hay không. Nếu được đặt thành "True", cây sẽ không được tỉa cắt sau khi xây dựng.
4. Subtree Raising: Tham số này xác định xem có nên thực hiện cải thiện trình tự các nút con hay không.
5. Binary Splits: Xác định xem cây có sử dụng các phân chia nhị phân (binary splits) hay không.
6. Reduced Error Pruning: Xác định xem thuật toán có sử dụng phương pháp tỉa cắt theo lỗi giảm (reduced error pruning) hay không.
7. Confidence pruning: Xác định xem thuật toán có sử dụng tỉa cắt theo ngưỡng độ tin cậy (confidence pruning) hay không.
8. Use MDL Correction: Xác định xem thuật toán có sử dụng sửa đổi MDL (Minimum Description Length) hay không.
9. Collapse Tree: Xác định xem thuật toán có sử dụng cách thu nhỏ cây (collapse tree) hay không.

### Thông tin thêm đối với bài toán có nhiều hơn 2 lớp, cần phải có quy tắc tính trung bình.

* Các quy tắc tính bao gồm: **micro**, **macro**, **weighted** và **none**. [4]
* Với average=None, điểm precision/recall của mỗi lớp được trả về (không tính trung bình), vì vậy chúng ta nhận được một mảng điểm có độ dài bằng số lượng lớp.
* Với average='macro', độ chính xác/thu hồi được tính cho từng lớp và sau đó lấy mức trung bình. Công thức của nó như sau:

vĩ mô

* Với average='micro', sự đóng góp của tất cả các lớp được tổng hợp để tính độ chính xác/thu hồi trung bình. Công thức của nó như sau:

vi mô

* Với average='weighted' thực sự là một mức trung bình vĩ mô có trọng số trong đó các trọng số là các lớp dương thực tế. Công thức của nó như sau:

có trọng lượng

TÀI LIỆU THAM KHẢO

[1].<https://machinelearningcoban.com/2016/12/26/introduce/>

[2] <https://www.datacamp.com/tutorial/categorical-data>

[3] <https://www.shiksha.com/online-courses/articles/handling-missing-data-mean-median-mode/>

[4] https://stackoverflow.com/questions/52269187/facing-valueerror-target-is-multiclass-but-average-binary