**TRƯỜNG ĐẠI HỌC THỦY LỢI**

**KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO BÀI TẬP LỚN HỌC PHẦN TIỀN XỬ LÝ DỮ LIỆU

ĐỀ TÀI:

Tiền xử lý dữ liệu đánh giá các loại ramen

Người hướng dẫn: Thầy Tạ Quang Chiểu

Nhóm thực hiện: Nhóm 1

1. Nguyễn Chí Minh, lớp 63TTNT

2. Nguyễn Tuấn Anh, lớp 63TTNT

3. Nguyễn Quang Khải, lớp 63TTNT

4. Nguyễn Tuấn Khang, lớp 63TTNT

Hà Nội, năm 2024

Mục lục

[PHẦN 1: GIỚI THIỆU BÀI TOÁN 3](#_Toc163044448)

[PHẦN 2: TIỀN XỬ LÝ DỮ LIỆU 4](#_Toc163044449)

[2.1. Khai phá dữ liệu 4](#_Toc163044450)

[2.2. Trực quan hóa dữ liệu 7](#_Toc163044451)

[2.3. Xử lý dữ liệu thiếu: 11](#_Toc163044452)

[2.4. Bổ sung thuộc tính mới từ thuộc tính đã có: 12](#_Toc163044453)

[2.5. Xử lý ngoại lệ 13](#_Toc163044454)

[2.6. Mã hoá dữ liệu categorical: 14](#_Toc163044455)

[2.7. Dữ liệu sau khi đã được tiền xử lý: 15](#_Toc163044456)

[PHẦN 3: ỨNG DỤNG MÔ HÌNH HỌC MÁY ĐỂ XỬ LÝ VỚI BÀI TOÁN 16](#_Toc163044457)

[3.1. Tổng quát chung về các mô hình học máy sử dụng trong bài toán 16](#_Toc163044458)

[3.1.1. Linear Regression: 16](#_Toc163044459)

[3.1.2. Lasso: 16](#_Toc163044460)

[3.1.3. Ridge: 17](#_Toc163044461)

[3.1.4. Neural Network (Multi-Layer Perceptron): 17](#_Toc163044462)

[3.2. Các công cụ, thư viện dùng xây dựng: 18](#_Toc163044463)

[3.3. Kết quả khi sử dụng mô hình 18](#_Toc163044464)

[PHẦN 4: TỔNG KẾT BÀI TẬP LỚN 19](#_Toc163044465)

[TÀI LIỆU THAM KHẢO 20](#_Toc163044466)

# 

# PHẦN 1: GIỚI THIỆU BÀI TOÁN

Mì Ramen là một loại thức ăn phổ biến, ngon nức tiếng ở Nhật Bản. Đây cũng là món ăn truyền thống mà người Nhật rất ưa chuộng và tự hào, nó cũng nhận được sự ủng hộ của các bạn bè quốc tế ở mọi lứa tuổi. Dần dần, ramen có mặt khắp thế giới với nhiều chủng loại, mẫu mã cùng thương hiệu khác nhau.

Ở bài tập lớn này, chúng em sẽ làm về dự án dự đoán mức độ đánh giá của từng loại ramen. Mục tiêu của bài tập lớn là tiền xử lý dữ liệu về đánh giá các loại ramen, sau đó đưa dữ liệu đã được xử lý vào một số mô hình học máy để thực hiện dự đoán.

Tổng quan tập dữ liệu của bài toán:

* Tập dữ liệu bao gồm 2580 mẫu dữ liệu
* Mỗi mẫu dữ liệu bao gồm các thông tin:
* Ma trận dữ liệu X:
  + Review #: Số lượt đánh giá.
  + Brand: Thương hiệu.
  + Variety: Tên sản phẩm.
  + Style: Sản phẩm được bày trí trên gì (bát, cốc, đĩa).
  + Country: Quốc gia tạo ra sản phẩm.
  + Top Ten: Sản phẩm nằm trong top 10 sản phẩm được đánh giá cao nhất..
* Nhãn Y:
  + Stars: Số điểm đánh giá sản phẩm

Một số dữ liệu mẫu



# PHẦN 2: TIỀN XỬ LÝ DỮ LIỆU

## 2.1. Khai phá dữ liệu

**2.1.1. Xem thông tin tổng quan của dữ liệu**

*print(data.info())*

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2580 entries, 0 to 2579

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Review # 2580 non-null int64

1 Brand 2580 non-null object

2 Variety 2580 non-null object

3 Style 2578 non-null object

4 Country 2580 non-null object

5 Stars 2580 non-null object

6 Top Ten 37 non-null object

dtypes: int64(1), object(6)

memory usage: 141.2+ KB

None

**2.1.2. Xác định kích cỡ và đếm số dòng dữ liệu không chứa giá trị Null theo từng cột của Dataframe**

*print(data.shape)*

*print(data.count())*

(2580, 7)

Review # 2580

Brand 2580

Variety 2580

Style 2578

Country 2580

Stars 2580

Top Ten 37

dtype: int64

**2.1.3. Thống kê dữ liệu thiếu**

*print(data.isnull().sum())*

Review # 0

Brand 0

Variety 0

Style 2

Country 0

Stars 0

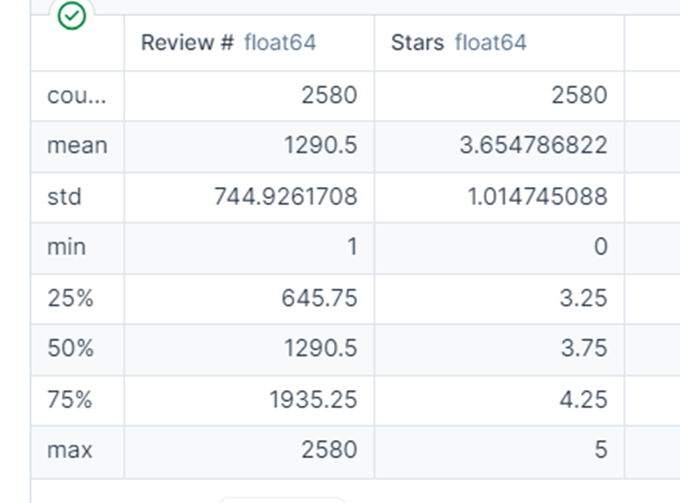
Top Ten 2543

dtype: int64

**2.1.4. Thông kê các thuộc tính trong tập dữ liệu**

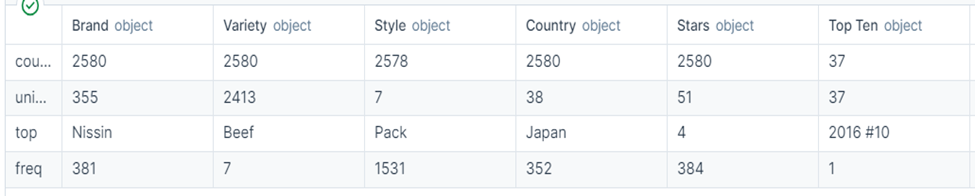
**#Thông kê các thuộc tính số**

*data.describe()*

****

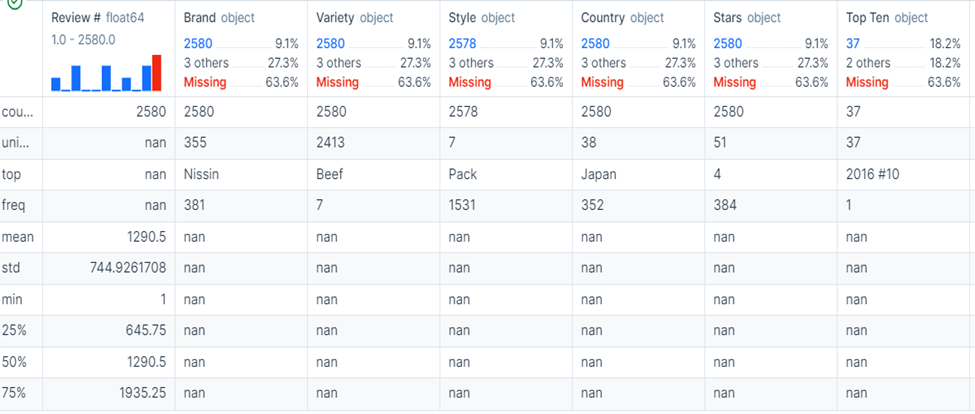
**#Thống kê các thuộc tính có kiểu dữ liệu object:**

*data.describe(include=['O'])*

****

**#Thông kê tập dữ liệu các thuộc tính**

*data.describe(include='all')*

****

**2.1.5. Phân tích các thuộc tính chủ chốt của dữ liệu ảnh hướng đến kết quả đánh giá**

* **Cột Style**

*data['Stars'] = pd.to\_numeric(data['Stars'], errors='coerce')*

*data[['Style','Stars']].groupby(['Style'],as\_index=False).mean().sort\_values(by='Stars',ascending=False)*

* **Cột Brand**

*data[['Brand','Stars']].groupby(['Brand'],as\_index=False).mean().sort\_values(by='Stars',ascending=False)*

* **Cột Variety**

*data[['Variety','Stars']].groupby(['Variety'],as\_index=False).mean().sort\_values(by='Stars',ascending=False)*

* **Cột Country**

*data[['Country','Stars']].groupby(['Country'],as\_index=False).mean().sort\_values(by='Stars',ascending=False)*

**2.1.6. Phân loại các thuộc tính**

* Categorical (Hạng mục): Đây là các thuộc tính mà giá trị của chúng không thể được sắp xếp theo thứ tự nào cả. Các giá trị chỉ đơn thuần là các nhãn khác nhau. Trong trường hợp này, ‘Brand’, ‘Variety’, ‘Style’, và ‘Country’ có thể được xem là thuộc tính hạng mục.
* Ordinal (Thứ tự): Đây là các thuộc tính mà giá trị của chúng có thể được sắp xếp theo một thứ tự cụ thể. Trong trường hợp này, không có thuộc tính nào rõ ràng là thuộc loại này dựa trên thông tin bạn cung cấp.
* Discrete (Rời rạc): Đây là các thuộc tính mà giá trị của chúng là các số nguyên hoặc giá trị rời rạc. ‘Review #’ và ‘Top Ten’ có thể được xem là thuộc tính rời rạc. Tuy nhiên, ‘Top Ten’ có thể cũng được xem là thuộc tính thứ tự nếu nó biểu thị xếp hạng. Continuous (Liên tục): Đây là các thuộc tính mà giá trị của chúng có thể nhận bất kỳ giá trị nào trong một khoảng cụ thể. ‘Stars’ có thể được xem là thuộc tính liên tục nếu nó biểu thị điểm số từ 0 đến 5.

## 2.2. Trực quan hóa dữ liệu

*plt.figure(figsize=(13, 3))*

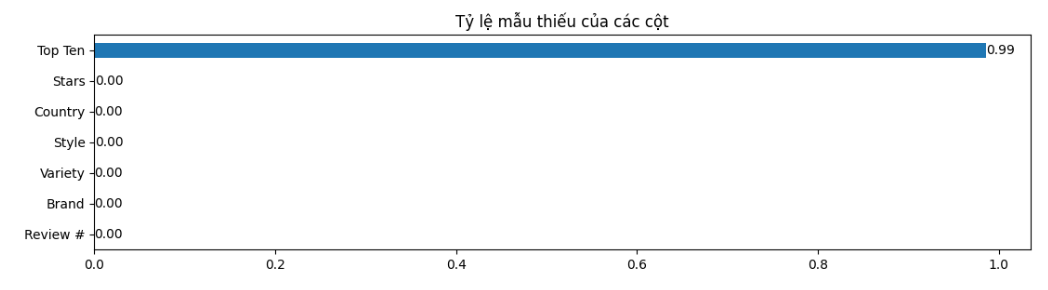
*missing\_ratio = data.isnull().mean(axis=0)*

*ax = missing\_ratio.plot.barh()*

*plt.title("Tỷ lệ mẫu thiếu của các cột")*

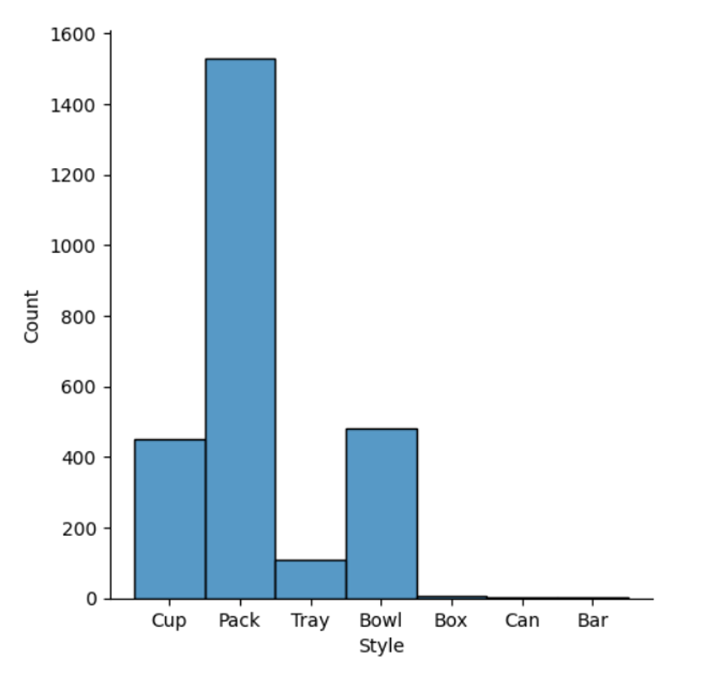
*for i, v in enumerate(missing\_ratio):*

*ax.text(v, i, f'{v:.2f}', ha='left', va='center')*

****

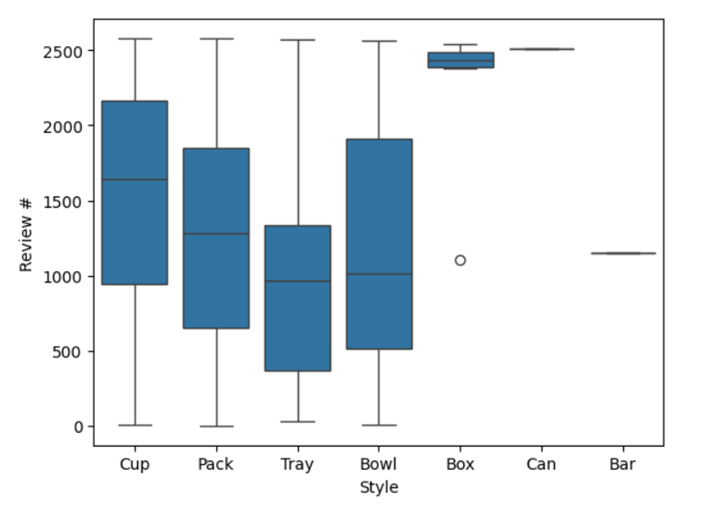
* Biểu đồ phân phối của dữ liệu trong cột 'Style'

*sns.displot(data['Style'])*

**

* Biểu đồ boxplot, hiển thị phân phối của số lượng đánh giá (Review #) cho mỗi kiểu ở cột Style

*sns.boxplot(x=data['Style'], y=data['Review #'])\*

**

Nhận xét:

*Kiểu ramen được sử dụng nhiều nhất trong tập dữ liệu là Pack.*

*Kiểu ramen được review nhiều nhất trong tập dữ liệu là Cup.*

**-**Biểu đồ cột hiển thị số lượng Country trong tập dữ liệu

*custom\_aggregation = {"Country": "count"}*

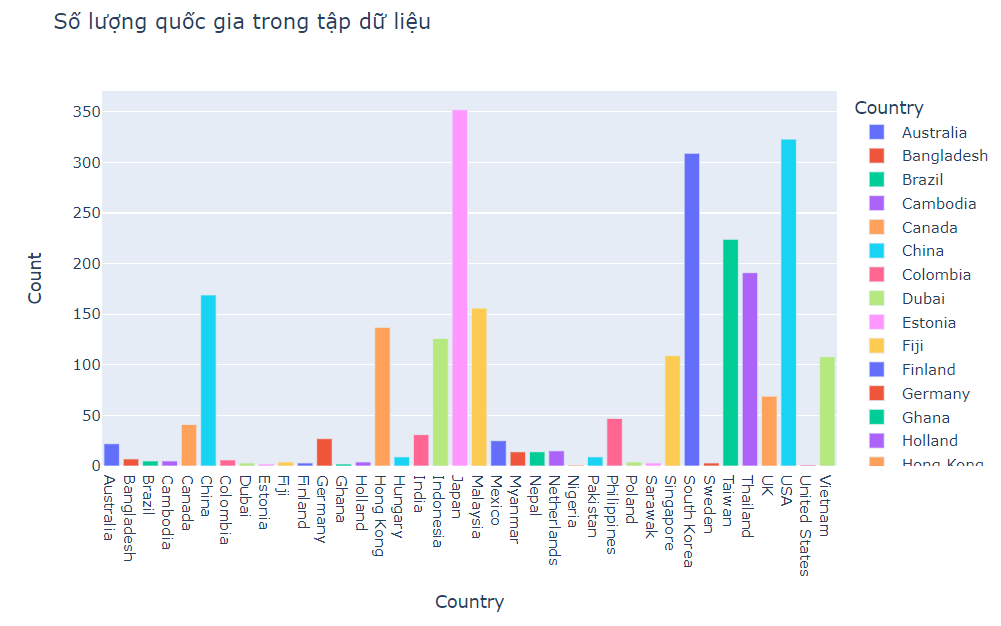
*data1 = data.groupby("Country").agg(custom\_aggregation)*

*data1.columns = ["Count"]*

*data1['Country'] = data1.index*

*fig = px.bar(data1, x='Country', y="Count", color="Country", title="Số lượng quốc gia trong tập dữ liệu")*

*fig.show()*

****

Nhìn biểu đồ, ta thấy Japan là nươc chiếm nhiều nhất trong tập dữ liệu, có 1 nước là United States ta sẽ gộp vào USA.

*data['Country'] = data['Country'].replace({'United States':'USA'})*

* Biểu đồ kết hợp gồm một biểu đồ cột và một biểu đồ tròn để thể hiện thông tin về số lượng mẫu ở cột Brand

*fig = plt.figure(figsize = (14,6))*

*fig.suptitle('Thương hiệu và số lượng mẫu mã')*

*ax1 = fig.add\_subplot(121)*

*sns.barplot(data = vairety\_data.head(10),y='Brand' ,x= 'Number\_Of\_Variety',orient='h',ax =ax1)*

*plt.xlabel('Số lượng mẫu mã')*

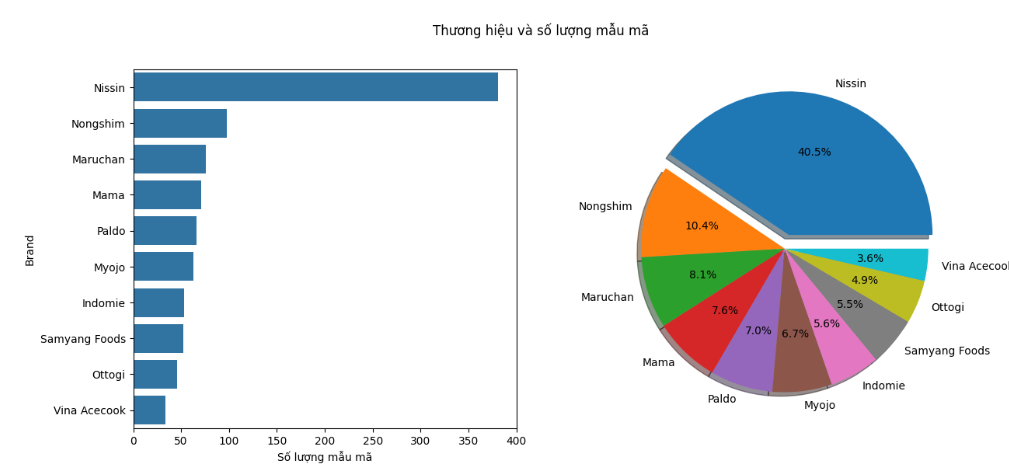
*ax2 = fig.add\_subplot(122)*

*plt.pie(vairety\_data.head(10)['Number\_Of\_Variety'],*

*explode = (0.1,0,0,0,0,0,0,0,0,0),labels =vairety\_data.head(10)['Brand'],*

*autopct='%1.1f%%',*

*shadow =True)*

****

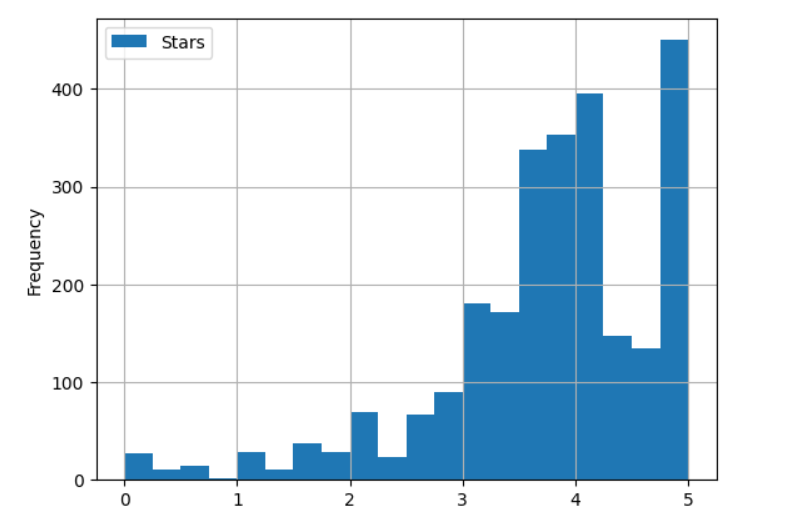
Nhìn biểu đồ ta thấy, Nissin là thương hiệu được đánh giá nhiều nhất và có số lượng mẫu mã nhiều nhất.

* Biểu đồ tần xuất cột Star:

*plt.figure()*

*data[['Stars']].plot.hist(alpha=1, bins=20)*

*plt.grid(True)*

****

## 2.3. Xử lý dữ liệu thiếu:

**2.3.1. Xem số lượng giá trị thiếu (giá trị NaN) trong mỗi cột của DataFrame :**

*data.isnull().sum()*

Review # 0

Brand 0

Variety 0

Style 2

Country 0

Stars 3

Top Ten 2543

dtype: int64

*data.shape*

(2580, 7)

* Nhận thấy giá trị thiếu do lỗi trong quá trình thu thập dữ liệu : Cột Style thiếu 2 giá trị , cột Stars thiếu 3 giá trị, cột Top ten thiếu 2543 giá trị

**2.3.2. Xử lí dữ liệu thiếu cho từng cột :**

* Xóa cột dữ liệu Top Ten do thiếu quá nhiều giá trị gây ảnh hưởng đến kết quả mô hình

*data = data.drop(columns=["Top Ten"],axis =1)*

* Thay thế giá trị thiếu bằng giá trị xuất hiện nhiều nhất trong cột Style

*data['Style'].fillna(data['Style'].mode()[0], inplace=True)*

* Thay thế giá trị thiếu bằng giá trị trung vị trong cột Stars

*data['Stars'].fillna(data['Stars'].median(), inplace=True )*

**2.3.3. Kiểm tra lại :**

*data.isnull().sum()*

Review # 0

Brand 0

Variety 0

Style 0

Country 0

Stars 0

dtype: int64

*data.shape*

(2580, 6)

→ Như vậy, tất cả giá trị thiếu tại các cột đều đã được xử lý.

## 2.4. Bổ sung thuộc tính mới từ thuộc tính đã có:

Vì cột Variety có chứa nhiều giá trị duy nhất, ta sẽ tách những dữ liệu ở cột Variety thành các đặc trưng là Is\_Spicy (có cay hay không), Noodle\_Types (loại mì), Meat\_Types (loại thịt), Contain\_Flavor (có hương vị có sẵn không).

**2.4.1. Tạo thuộc tính Is\_Spicy từ cột Variety, cho biết sản phẩm có cay hay không:**

*def is\_spicy(word):*

*word\_lower = word.lower()*

*spc = ['spicy', 'spicies', 'chili', 'chilli', 'hot']*

*for i in spc:*

*if i in word\_lower:*

*return 1*

*return 0*

*data['Is\_Spicy'] = data['Variety'].apply(is\_spicy)*

. **2.4.2. Tạo thuộc tính Meat\_Types từ cột Variety, cho biết loại thịt được sử dụng:**

*meat\_types = ['chicken', 'beef', 'pork', 'shrimp', 'seafood', 'crab', 'fish', 'duck', 'turkey']*

*def meat\_scanner(word):*

*word\_lower = word.lower()*

*for meat in meat\_types:*

*if meat in word\_lower:*

*return meat*

*return 'undefined'*

*data['Meat\_Types'] = data['Variety'].apply(meat\_scanner)*

**2.4.3. Tạo thuộc tính Noodle\_Types từ cột Varirty, cho biết loại mì được sử dụng:**

*noodle\_types = ['noodle', 'udon', 'soba', 'egg', 'ramen']*

*def noodle\_scanner(word):*

*word\_lower = word.lower()*

*for noodle in noodle\_types:*

*if noodle in word\_lower:*

*return noodle*

*return 'undefined'*

*data['Noodle\_Types'] = data['Variety'].apply(noodle\_scanner)*

**2.4.4. Tạo thuộc tính Contain\_Flavor, cho biết sản phẩm có sẵn hương vị hay không:**

*def contain\_flavor(word):*

*word\_lower = word.lower()*

*return 1 if 'flavor' in word\_lower or 'flavour' in word\_lower else 0*

*data['Contain\_Flavor'] = data['Variety'].apply(contain\_flavor)*

Sau đó, loại bỏ cột Variety ra khỏi tập dữ liệu:

*data.drop[‘Variety’, axis=1]*

## 2.5. Xử lý ngoại lệ

* Đưa ra số lần xuất hiện của các giá trị trong cột Style

*data['Style'].value\_counts()*

Style

Pack 1533

Bowl 481

Cup 450

Tray 108

Box 6

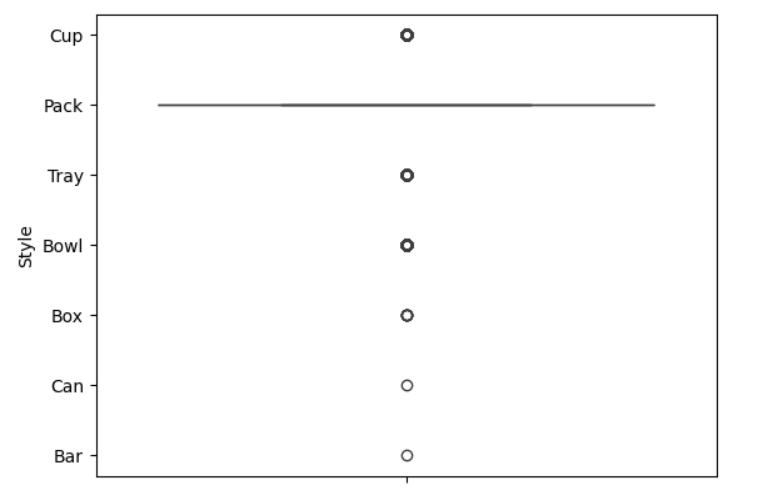
Can 1

Bar 1

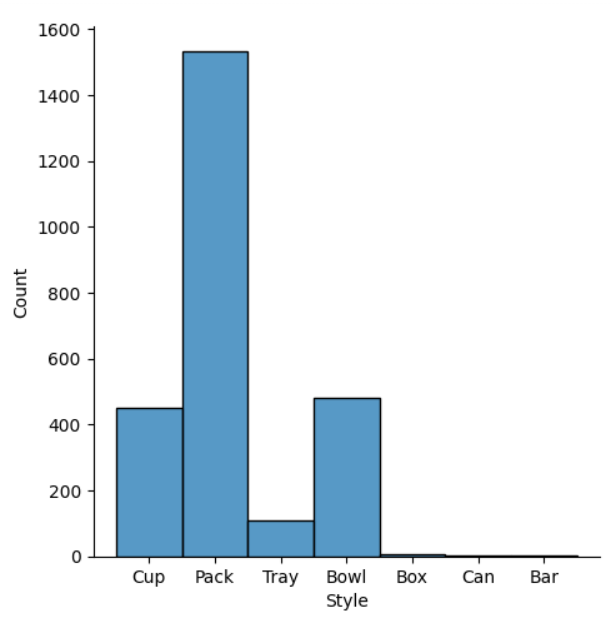
Name: count, dtype: int64

* Biểu đồ boxplot hiển thị các giá trị từ cột 'Style' của DataFrame

*sns.boxplot(y='Style', data=data)*



* Biểu đồ phân phối của dữ liệu trong cột 'Style'



Dựa vào biếu đồ, ta thấy dữ liệu thường tập trung ở 4 loại chính là Cup, Pack, Tray và Bowl, trong đó tập trung nhiều nhất là Pack. Những loại còn lại Box, Can, Bar xuất hiện với số lượng rất ít và chiếm 0,003% tập dữ liệu. Vì vậy, những giá trị này có thể gây nhiễu dữ liệu, nên ta sẽ loại bỏ những dữ liệu có chứa các loại này.

*filt = data['Style'].isin(['Bar', 'Box', 'Can'])*

*data[filt]*

*data = data[~filt]*

## 2.6. Mã hoá dữ liệu categorical:

* **Các cột cần mã hóa:**
  + Brand
  + Noodle\_Types
  + Meat\_Types
  + Style
  + Country

**2.6.1. Mã hóa cột Style**

*from sklearn.preprocessing import OneHotEncoder*

*encoder = OneHotEncoder()*

*encoder\_data = encoder.fit\_transform(data[['Style']]).toarray()*

*column\_names = [f"{value}" for value in encoder.get\_feature\_names\_out(['Style'])]*

*new\_data = pd.DataFrame(encoder\_data, columns=column\_names)*

*data=data.join(new\_data)*

*data.drop('Style',axis=1,inplace=True)*

**2.6.2. Mã hóa cột Brand, Noodle\_Types, Meat\_Types, Country**

*frequency\_brand = (data['Brand'].value\_counts() / len(data) ).to\_dict()*

*data['Brand'] = data['Brand'].map(frequency\_brand)*

*frequency\_meat\_types = (data['Meat\_Types'].value\_counts() / len(data) ).to\_dict()*

*data['Meat\_Types'] = data['Meat\_Types'].map(frequency\_meat\_types)*

*frequency\_noodle\_types = (data['Noodle\_Types'].value\_counts() / len(data) ).to\_dict()*

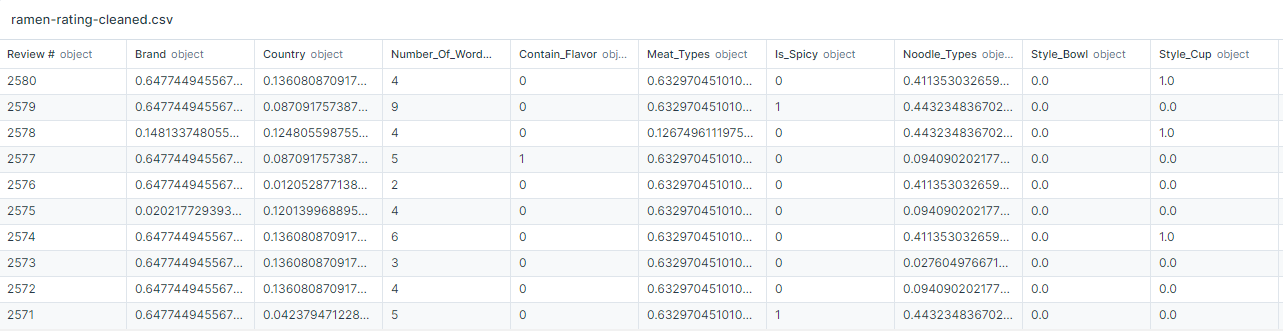
*data['Noodle\_Types'] = data['Noodle\_Types'].map(frequency\_noodle\_types)*

*frequency\_country = (data['Country'].value\_counts() / len(data) ).to\_dict()*

*data['Country'] = data['Country'].map(frequency\_country)*

* **Lí do sử dụng các phương pháp để mã hóa**
* One-Hot Encoding: Phương pháp này tạo ra một vectơ nhị phân cho mỗi lớp hạng mục. Điều này rất hữu ích khi không có thứ tự tự nhiên hoặc quan hệ giữa các lớp. Với số lượng giá trị duy nhất nhỏ (7 giá trị), bạn có thể sử dụng One-Hot Encoding. Phương pháp này sẽ tạo ra một vectơ nhị phân cho mỗi lớp hạng mục, giúp mô hình dễ dàng phân biệt giữa các lớp.
* Frequency Encoding: Việc sử dụng Frequency Encoding cho các cột có số lượng giá trị duy nhất lớn có thể mang lại kết quả tốt. Phương pháp này thay thế mỗi hạng mục bằng tần suất xuất hiện của nó trong dữ liệu, giúp giảm kích thước dữ liệu khi có nhiều giá trị duy nhất. Nó có thể giúp mô hình học được mối quan hệ giữa tần suất xuất hiện của một hạng mục và biến mục tiêu.

## 2.7. Dữ liệu sau khi đã được tiền xử lý:

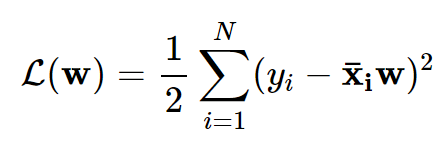
****

# PHẦN 3: ỨNG DỤNG MÔ HÌNH HỌC MÁY ĐỂ XỬ LÝ VỚI BÀI TOÁN

## 3.1. Tổng quát chung về các mô hình học máy sử dụng trong bài toán

### 3.1.1. Linear Regression:

* Giới thiệu: Linear Regression (hồi quy tuyến tính) là 1 thuật toán học máy có giám sát, đây là thuật toán cơ bản, dễ tiếp cận và đơn giản nhất trong học máy.
* Mục đích: Dùng để dự đoán giá trị của 1 biến (target) dựa trên giá trị của những biến còn lại.
* Mô tả chung về thuật toán:
* Input: Tập dữ liệu huấn luyện X\_train, y\_train.
* Output: Giá trị dự đoán đầu ra f(x) = xT.w + b sao cho sai khác giữa giá trị thực tế và giá trị dự đoán là nhỏ nhất.
* Hàm mất mát: là hàm sao cho giá trị dự đoán so với giá trị thực tế có sai số nhỏ nhất, được viết dưới dạng



trong đó: yi là giá trị thực tế.

xi là giá trị của các biến còn lại.

w là hệ số tối ưu.

xiw là giá trị dự đoán.

* Cách giải bài toán: Giải phương trình đạo hàm của hàm mất mát bằng 0, thu được phương trình sau



### 3.1.2. Lasso:

* Giới thiệu: Mô hình hồi quy Lasso (hay còn được biết đến là chuẩn hoá L1), là kĩ thuật phổ biến trong thống kê và học máy để ước tính mỗi quan hệ giữa các biến và tạo dự đoán. Nó hoạt động bằng cách vừa tìm mối quan hệ giữa các biến độc lập (biến dự đoán) với biến phụ thuộc (biến cần dự đoán) giống như hồi quy tuyến tính, vừa loại bỏ các biến không quan trọng ra khỏi mô hình.
* Mục đích: Dùng để dự đoán giá trị của 1 biến (target) dựa trên giá trị của những biến còn lại. Sử dụng mô hình Lasso khi muốn tránh overfitting, loại bỏ những thuộc tính không hữu ích cho mô hình.
* Mô tả chung về thuật toán:
* Cách hoạt động: Lasso sẽ tìm ra sự cân bằng giữa sự đơn giản hoá và độ chính xác của mô hình bằng cách thêm hàm phạt (L1) so với mô hình hồi quy Linear Regression để đóng phạt các hệ số hồi quy của các biến. Hàm phạt này khuyến khích các hệ số bằng 0, do đó loại bỏ thuộc tính không quan trọng ra khỏi mô hình, cải thiện khả năng tổng quát hoá, dự đoán trên dữ liệu mới tốt hơn.
* Input: Tập dữ liệu huấn luyện X\_train, y\_train.
* Output: Giá trị dự đoán đầu ra f(x) = xT.w + b sao cho sai khác giữa giá trị thực tế và giá trị dự đoán là nhỏ nhất.
* Hàm mất mát:



→ Mô hình Lasso chọn đặc trưng hữu ích cho mô hình, sẽ giảm thiểu khả năng overfitting.

### 3.1.3. Ridge:

* Giới thiệu: Mô hình học máy Ridge (hay còn được biết đến là chuẩn hoá L2) là 1 kỹ thuật thống kê được áp dụng trong mô hình hồi quy tuyến tính để giải quyết vấn đề đa cộng tuyến tính. Đa cộng tuyến tính xảy ra khi có mức độ tương quan giữa 2 hoặc nhiều biến độc lập trong dữ liệu.
* Mục đích: Dùng để dự đoán giá trị của 1 biến (target) dựa trên giá trị của những biến còn lại. Sử dụng mô hình Ridge khi dữ liệu có sự tương quan cao giữa 2 hoặc nhiều biến độc lập.
* Mô tả chung về thuật toán:
* Cách hoạt động: Ridge giải quyết đa cộng tuyến tính bằng cách thêm hàm phạt vào hàm mất mát của hồi quy tuyến tính thông thường, hàm này sẽ phạt các hệ số có giá trị lớn, khuyến khích chúng tiến gần về 0, do đó sẽ giảm thiểu độ lớn của các hệ số, làm mô hình ít nhạy cảm với đa cộng tuyến tính.
* Input: Tập dữ liệu huấn luyện X\_train, y\_train.
* Output: Giá trị dự đoán đầu ra f(x) = xT.w + b sao cho sai khác giữa giá trị thực tế và giá trị dự đoán là nhỏ nhất.
* Hàm mất mát:



→ Mô hình Ridge tìm ra các hệ số trong w không quá lớn, giúp tránh việc đầu ra phụ thuộc quá nhiều vào một đặc trưng nào đó.

### 3.1.4. Neural Network (Multi-Layer Perceptron):

* Giới thiệu: Neural Network là 1 loại mạng nơron nhân tạo được sử dụng để giải quyết các vấn đề học máy khác nhau. Nó được cấu trúc bởi các lớp của perceptron - là đơn vị xử lý thông tin đơn giản mô phỏng các nơron thần kinh sinh học.
* Mục đích: Mô hình Neural Network được sử dụng khi muốn xử lý các vấn đề phức tạp mà các thuật toán truyền thống không thể xử lý được như xử lý ảnh, dự báo, xử lý ngôn ngữ tự nhiên,... Mô hình Multi-Layer Perceptron sử dụng ở đây chỉ phù hợp với dữ liệu có kích thước không quá lớn.
* Mô tả chung về thuật toán:
* Input: Tập dữ liệu huấn luyện X\_train, y\_train, hàm mất mát, hàm kích hoạt, kiến trúc mạng nơron.
* Output: Bộ vector trọng số của các liên kết giữa các nơron (W) để hàm mất mát đạt giá trị tối ưu.
* Hàm mất mát: Tuỳ theo bài toán là hồi quy hay phân lớp, hàm mất mát sẽ khác nhau.
* Phương thức: Gradient Descent
* Cách giải bài toán: Để áp dụng Gradient Descent, chúng ta cần tính được gradient của hàm mất mát theo từng ma trận trọng số W(*l) ,* và vector bias b(l) theo phương pháp lan truyền ngược (backpropagation).

## 

## 3.2. Các công cụ, thư viện dùng xây dựng:

* Trong bài toán này, ta sẽ sử dụng 4 mô hình Linear Regression, Lasso, Ridge và Neural Network cùng với các độ đo R2, NSE, MAE, RMSE.
* Với mô hình Linear Regression chúng em đã tự viết mô hình còn 3 mô hình còn lại sử dụng thư viện đã có sẵn của scikit-learn.
* Xây dựng mô hình, phân tích lựa chọn tham số:
* Linear Regression: Sử dụng tham số mặc định của mô hình
* Lasso + Ridge:
* alpha: Giá trị alpha kiểm soát hệ số regularization đồng thời còn giúp giảm thiểu overfitting, vậy nên cần thử nhiều lần để có thể tìm được 1 giá trị alpha tối ưu nằm trong khoảng [0,inf). Từ đó chọn ra alpha tối ưu nhất cho Lasso là 0.001 còn Ridge là 1.
* max\_iter: số lần lặp tối đa khi chạy mô hình. Cần phải chọn ra một max\_iter không quá nhỏ (không đạt được giải pháp tối ưu) hay quá lớn (tiêu tốn quá nhiều thời gian).
* Neural Network:
* activation: chọn hàm kích hoạt cho lớp ẩn. Chọn hàm ‘tanh’ do mang lại hiệu suất tốt nhất cho mô hình.
* hidden\_layer\_sizes: chọn số lượng nodes trong lớp ẩn, trong bài toán này chọn mô hình có 2 lớp ẩn và mỗi layer chứa 10 nodes.
* solver: chọn thuật giải. Thuật giải tối ưu nhất cho bài toán là ‘sgd’.
* alpha: hệ số điều chỉnh, tránh overfitting.
* learning\_rate: tốc độ học.
* max\_iter: số lần lặp tối đa khi chạy mô hình. Cần phải chọn ra một max\_iter không quá nhỏ (không đạt được giải pháp tối ưu) hay quá lớn (tiêu tốn quá nhiều thời gian).

## 3.3. Kết quả khi sử dụng mô hình

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Linear Regression | Lasso | Ridge | Neural Network |
| R2 | 0.039 | 0.040 | 0.040 | 0.035 |
| NSE | 0.039 | 0.040 | 0.040 | 0.035 |
| MAE | 0.729 | 0.728 | 0.728 | 0.732 |
| RMSE | 1.020 | 1.020 | 1.020 | 1.023 |

Dựa trên kết quả của các mô hình, ta thu được những nhận xét sau:

* Do dữ liệu gần như không có sự tương quan với nhau kèm theo số lượng dữ liệu là không nhiều khiến cho điểm R2 không được tốt, tuy nhiên độ đánh giá RMSE là phù hợp, nó cho thấy mô hình dự đoán khá chính xác, sai số trung bình giữa giá trị dự đoán và thực tế là 1.02 đơn vị.
* 2 mô hình cho ra kết quả tốt nhất là Lasso và Ridge với R2\_Score là 0.04 cùng với RMSE là 1.02

# PHẦN 4: TỔNG KẾT BÀI TẬP LỚN

Sau một khoảng thời gian làm bài tập lớn với đề tài “Dự đoán mức độ đánh giá của các loại Ramen” dưới sự hướng dẫn của thầy Tạ Quang Chiểu, nhóm chúng em đã thu được kết quả sau:

* Áp dụng các kĩ thuật tiền xử lý đã được học để tiền xử lý dữ liệu như xử lý giá trị thiếu, mã hoá các cột, xử lý ngoại lệ,... giúp cho dữ liệu trở nên sạch sẽ hơn, có thể đưa vào các mô hình học máy.
* Áp dụng dữ liệu đã được tiền xử lý vào các mô hình học máy Linear Regression, Ridge, Lasso, MLP Regressor để dự đoán.

Tuy nhiên, nhóm chúng em còn có một số nhược điểm:

* Dữ liệu đưa vào các mô hình học máy cho kết quả với độ đo R2 còn thấp, điều đó cho rằng sự tương quan giữa biến phụ thuộc với biến đầu ra của dữ liệu có sự tương quan thấp, còn sai số là 1.02 có thể chấp nhận được. Chúng em sẽ cải tiến các mô hình này trở nên tốt hơn và có thể đưa vào sử dụng thực tế.

# TÀI LIỆU THAM KHẢO

[1] Tiền xử lý dữ liệu:[**https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/**](https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/)

[2] eBook Data Preprocessing with Python for Absolute Beginners: First Printing, 2020. Edited by AI Publishing. Ebook Converted and Cover by Gazler Studio. Published by AI Publishing LLC