TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

---□&□---

BÁO CÁO ĐỒ ÁN MÁY HỌC

Phân Loại Chất Lượng Không Khí bằng Softmax Regression

Giảng viên hướng dẫn: ThS. Vũ Đình Ái

Thành viên thực hiện:

Lê Nguyễn Nhật Tân	22DH114725
Nguyễn Trần Nhật Tân	22DH113258
Quang Công Văn	22DH114809

Thành phố Hồ Chí Minh, 2024

Danh mục ảnh

I. Giới thiệu

Ô nhiễm không khí là sự ô nhiễm môi trường trong nhà hoặc ngoài trời bởi bất kỳ tác nhân hóa học, vật lý hoặc sinh học nào làm thay đổi các đặc tính tự nhiên của khí quyển. Các thiết bị đốt trong gia đình, xe cơ giới, cơ sở công nghiệp và cháy rừng là những nguồn gây ô nhiễm không khí phổ biến. Các chất gây ô nhiễm gây lo ngại lớn cho sức khỏe cộng đồng bao gồm các hạt vật chất, carbon monoxide, ozone, nitơ dioxide và sulfur dioxide. Ô nhiễm không khí ngoài trời và trong nhà gây ra các bệnh về đường hô hấp và các bệnh khác, đồng thời là nguyên nhân quan trọng gây bệnh tật và tử vong.

Hiện nay, tình trạng ô nhiễm không khí đang là vấn đề cấp bách trên toàn cầu, đặc biệt tại các thành phố lớn và khu công nghiệp phát triển. Nhiều nghiên cứu đã chỉ ra rằng ô nhiễm không khí là nguyên nhân chính gây ra các bệnh về hô hấp, tim mạch, và thậm chí là ung thư. Theo Tổ chức Y tế Thế giới (WHO), khoảng 7 triệu người chết mỗi năm [1] do các bệnh liên quan đến không khí ô nhiễm, và 99% [2] dân số thế giới hít thở không khí vượt quá giới hạn hướng dẫn của WHO và chứa mức độ ô nhiễm cao, với các quốc gia có thu nhập thấp và trung bình chịu tác động nặng nề nhất.

Những chất gây ô nhiễm như khí CO, Ozone, NO2, và bụi mịn PM2.5 đang ngày càng gia tăng do sự phát triển công nghiệp, phương tiện giao thông, và đốt nhiên liệu hóa thạch. Đặc biệt, bụi mịn PM2.5 có khả năng xâm nhập sâu vào phổi, gây hại nghiêm trọng cho sức khỏe con người. Các thành phố như New Delhi, Bắc Kinh, và Jakarta là những ví dụ điển hình về mức độ ô nhiễm không khí ở mức báo động (151 – 200 theo AQI US) [3]

Giải pháp cấp bách hiện nay là phải có những biện pháp kiểm soát chặt chẽ về môi trường, kết hợp với việc ứng dụng công nghệ và khoa học dữ liệu để phân tích, dự báo và cảnh báo tình trạng ô nhiễm không khí. Việc sử dụng các thuật toán máy học, chúng ta có thể thu thập và phân tích dữ liệu không khí từ nhiều nguồn, dựa trên các chỉ số như CO, Ozone, NO2, và PM2.5 để đưa ra các mô hình dự đoán và phân loại mức độ ô nhiễm dựa trên dữ liệu thành các nhóm như: Tốt, Trung bình, Kém, Xấu, Rất xấu và Nguy hiểm. Qua đó, có thể cảnh báo sớm và hỗ trợ các cơ quan chức năng trong việc triển khai các biện pháp giảm thiểu ô nhiễm như kiểm soát rác thải, quy hoạch đô thị và nâng cao nhận thức của cộng đồng.

II. Cơ sở lý thuyết

II.1. Thuật toán Softmax Regression

Softmax Regression xuất phát từ lĩnh vực *hồi quy logistic (Logistic Regression),* vốn được sử dụng cho các bài toán phân loại nhị phân. Khi các nhà nghiên cứu muốn mở rộng hồi quy logistic cho các bài toán phân loại với nhiều lớp hơn (đa lớp), *Softmax Regression* được phát triển và trở thành một công cụ hiệu quả cho việc phân loại trong các tình huống này.

Thuật toán này được sử dụng phổ biến trong nhiều lĩnh vực như thống kê, xử lý ngôn ngữ tự nhiên (NLP), thị giác máy tính (Computer Vision), và đặc biệt là trong các mạng nơ-ron nhân tạo (Artificial Neural Networks, ANN), nơi softmax thường được sử dụng ở lớp cuối cùng để dự đoán xác suất cho các lớp.

Softmax Regression đã được phát triển và ứng dụng trong nhiều nghiên cứu liên quan đến trí tuệ nhân tạo (AI) và học máy (Machine Learning). Một số nghiên cứu quan trọng bao gồm:

- *Andrew Ng* và cộng sự tại Stanford đã đưa Softmax Regression vào giáo trình của các khóa học học máy như *Coursera Machine Learning*, giúp thuật toán này trở nên phổ biến trong cộng đồng học thuật và các ứng dụng thực tiễn [8]
- *Geoffrey Hinton* (1986) và *Yann LeCun* (1998) đã nghiên cứu và phát triển các mô hình mạng nơ-ron, trong đó Softmax Regression đóng vai trò là lớp phân loại ở tầng cuối cùng trong mạng nơ-ron. Công trình nghiên cứu về mạng nơ-ron như *LeNet-5* của LeCun đã sử dụng Softmax cho các bài toán nhận dạng hình ảnh, đặc biệt là nhận diện chữ viết tay [6]
- *Goodfellow, I., Bengio, Y., & Courville, A. (2016)* trong cuốn sách *Deep Learning* đã mô tả chi tiết về việc sử dụng Softmax trong học sâu, giúp nâng cao khả năng phân loại trong các bài toán có nhiều nhãn (multiclass classification) [9]

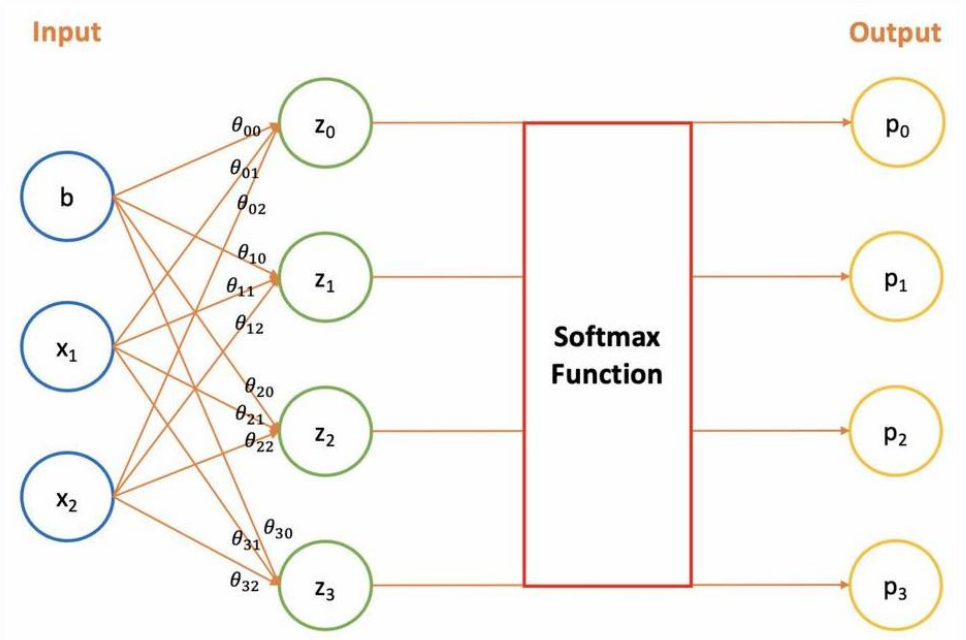
Ưu điểm:

- Softmax Regression là một phương pháp đơn giản để giải quyết bài toán phân loại đa lớp. Nó dễ hiểu và dễ triển khai, làm cho nó trở thành lựa chọn phổ biến trong giáo dục và ứng dụng thực tiễn.

- Cung cấp xác suất: *Softmax* chuyển đổi đầu ra của mô hình thành xác suất cho mỗi lớp, cho phép người dùng hiểu được độ tin cậy của dự đoán. Điều này rất hữu ích trong các ứng dụng yêu cầu ra quyết định dựa trên xác suất, như trong nhận diện hình ảnh hoặc xử lý ngôn ngữ tự nhiên.
- Tính khả thi với dữ liệu Lớn: *Softmax Regression* hoạt động tốt trên tập dữ liệu lớn và có thể mở rộng khi số lượng lớp tăng lên. Điều này làm cho nó phù hợp với nhiều bài toán phân loại thực tế.
- Hỗ trợ huấn luyện bằng *Gradient Descent*: *Softmax Regression* có thể được huấn luyện hiệu quả bằng các phương pháp tối ưu hóa như *Stochastic Gradient Descent (SGD)*, *Mini-Batch Gradient Descent* giúp thực hiện huyến luyện nhanh chóng và hiệu quả trong việc cải thiện mô hình.

Khuyết điểm:

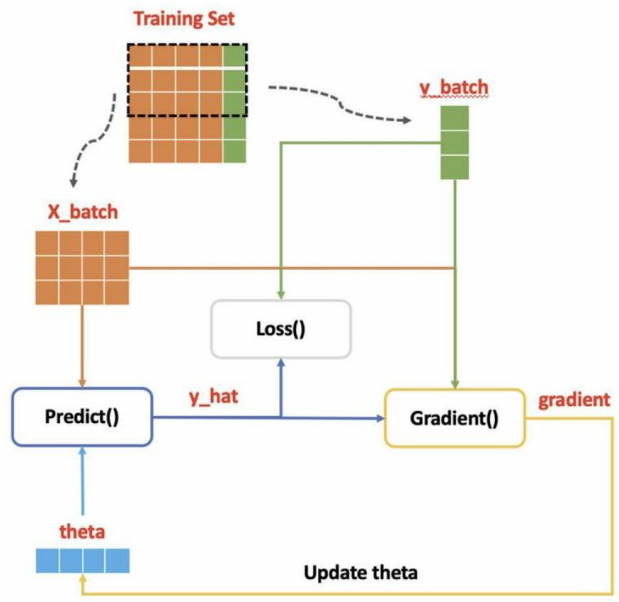
- Khi một trong các giá trị đầu vào lớn hơn nhiều so với các giá trị khác, thuật toán Softmax có thể dẫn đến các xác suất không chính xác.
- **Không ổn định với lớp đầu ra lớn**: Khi số lớp đầu ra lớn, tính toán của hàm Softmax có thể trở nên không ổn định, có thể dẫn đến tràn số (overflow) trong quá trình tính toán.
- **Cần thêm kỹ thuật regularization**: Khi sử dụng Softmax trong các mô hình lớn, cần áp dụng thêm các kỹ thuật regularization để tránh overfitting.



Hình II.1 Mô phỏng đơn giản về hàm Softmax

Các bước thực thi hàm Softmax đơn giản bằng mã giả:

```
# 1. Khởi tạo tham số
W, b = random_init()
for epoch in range(max_epochs):
   shuffle(X, Y)  # Trộn dữ liệu để đảm bảo tính ngẫu nhiên

   # 2. Chia dữ liệu thành mini-batch
   for mini_batch in split(X, Y, batch_size):
      X_batch, Y_batch = mini_batch  # Lấy dữ liệu của từng batch

      # 3. Tính dự đoán
      Z = X_batch . W + b
      P = softmax(Z)

      # 4. Tính mất mát
      loss = -sum(Y_batch * log(P)) / batch_size

      # 5. Gradient
      dW = X_batch^T . (P - Y_batch) / batch_size
      db = sum(P - Y_batch) / batch_size

      # 6. Cập nhật tham số
      W = W - learning_rate * dW
      b = b - learning_rate * db

   # In mất mát mỗi epoch
   print("Epoch:", epoch, "Loss:", loss)
```



Hình II.2 Sơ đồ hoạt động thuật toán với Mini-Batch Gradient Descent

II.2. Phương pháp đánh giá

Accuracy (Độ chính xác): Độ chính xác là thước đo cơ bản, xuất phát từ tỷ lệ dự đoán đúng so với tổng số mẫu, phổ biến trong các bài toán phân lớp nhị phân và đa lớp. Phù hợp khi dữ liệu được phân bố đều giữa các lớp, không có lớp nào chiếm ưu thế.

$$Precision = \frac{Số\ dự\ đoán\ đúng}{Tổng\ số\ mẫu}$$

Precision, Recall, và F1-Score

- **Precision (Độ chính xác):** Đánh giá tỷ lệ mẫu dự đoán đúng lớp dương so với tổng số mẫu được dự đoán là dương. Tốt khi bài toán ưu tiên giảm kết quả sai dương. Hữu ích trong các bài toán như phân loại email spam, nơi sai dương (email bình thường bị gắn là spam) cần giảm thiểu.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall (Độ bao phủ):** Đánh giá khả năng tìm đúng các mẫu thuộc lớp dương trong tất cả các mẫu thật sự thuộc lớp dương. Tốt khi cần giảm thiểu sai âm. Phù hợp cho các bài toán y tế (chẩn đoán bệnh), nơi việc bỏ sót các trường hợp thực sự dương là nghiêm trọng.

$$Recall = \frac{TP}{TP + FN}$$

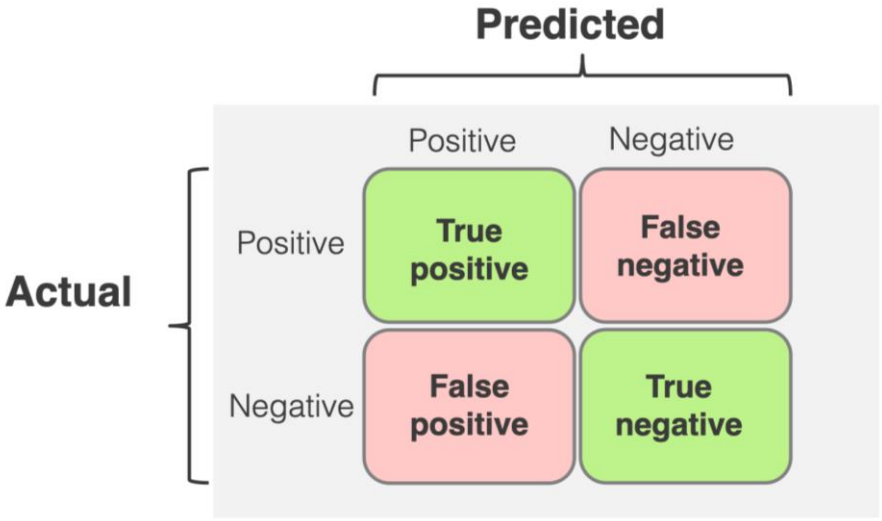
- **F1-Score:** Trung bình điều hòa của Precision và Recall. F1-Score hữu ích trong các tình huống mà cần cân bằng giữa precision và recall.

$$\text{F1Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Cross-Entropy Loss: Đây là một hàm mất mát phổ biến trong các bài toán phân loại đa lớp, đo lường sự khác biệt giữa nhãn thực tế và xác suất dự đoán.

$$L(\mathrm{y}, \hat{y}) = -\sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \log(\hat{y}_{ij})$$

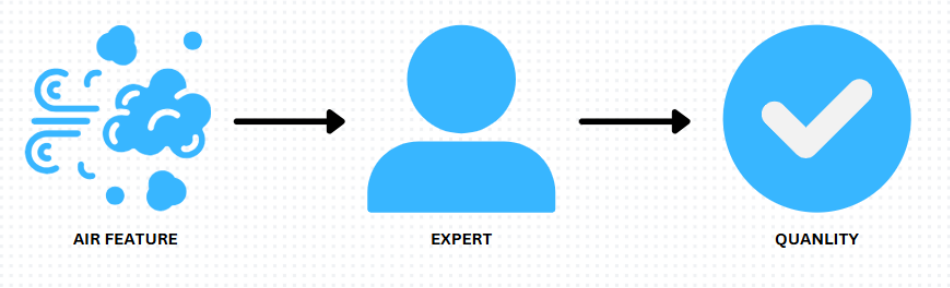
Confusion Matrix (Ma trận nhầm lẫn): Ma trận này giúp kiểm tra chất lượng phân loại bằng cách cho biết số lượng dự đoán đúng và sai cho từng lớp. Nó cung cấp thông tin về *True Positives (TP), False Positives (FP), True Negatives (TN), và False Negatives (FN).*



Hình II.3 Confusion Matrix

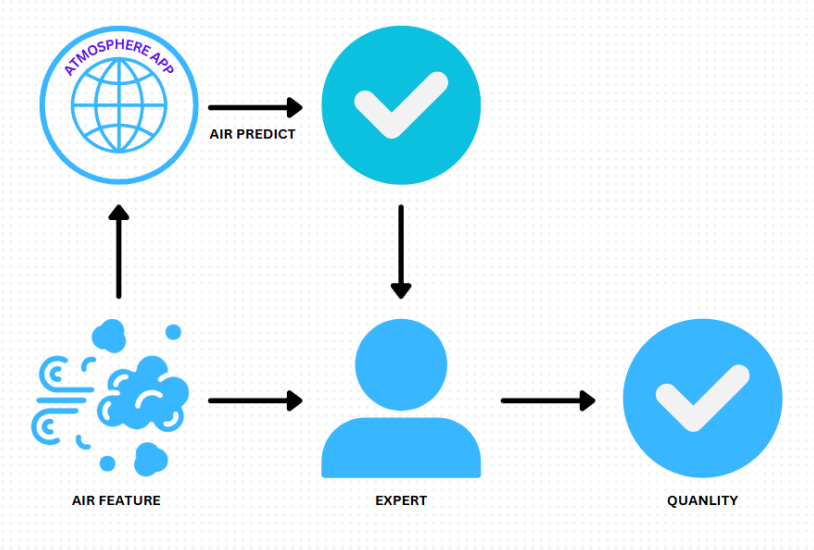
III. Đề xuất giải pháp

Quy trình đánh giá



Hình III.1 Quy trình đánh giá chất lượng không khí của chuyên gia

Đề xuất một quy trình đánh giá



Hình III.2 Quy trình đánh giá chất lượng không khí dựa vào ứng dụng dự đoán

Phương pháp tiếp cận:

Phương pháp tiếp cận trong nghiên cứu này bắt đầu bằng việc thu thập dữ liệu về chất lượng không khí từ Kaggle, nơi cung cấp các bộ dữ liệu liên quan đến các chỉ số ô nhiễm như PM2.5, PM10, CO, NO2 và các yếu tố môi trường khác. Sau khi dữ liệu được thu thập, quá trình tiền xử lý sẽ được thực hiện để làm sạch và chuẩn hoá dữ liệu. Điều này bao gồm việc xử lý các giá trị thiếu, loại bỏ các giá trị ngoại lai và chuẩn hoá các đặc trưng của dữ liệu để đưa về cùng một tỷ lệ, giúp đảm bảo mô hình học máy hoạt động hiệu quả.

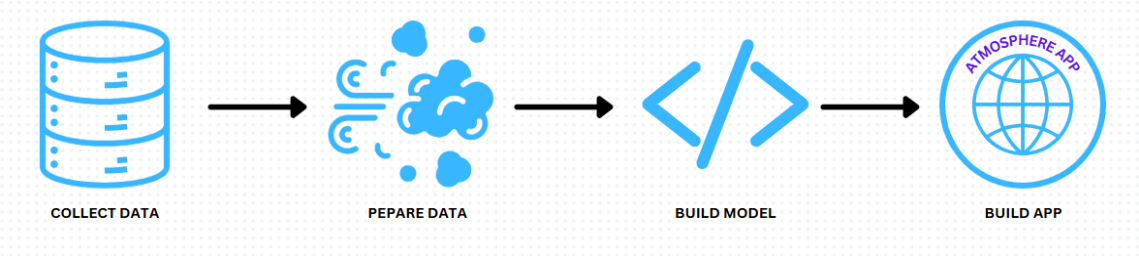
Tiếp theo, thuật toán *Softmax Regression* sẽ được áp dụng để xây dựng mô hình phân loại chất lượng không khí. Softmax Regression là một thuật toán phân loại đa lớp,

giúp phân loại dữ liệu ô nhiễm không khí thành các nhóm khác nhau, chẳng hạn như chất lượng không khí tốt, trung bình hoặc xấu, dựa trên các chỉ số ô nhiễm và các yếu tố môi trường. Mô hình sẽ được huấn luyện với dữ liệu đã chuẩn hoá, và trọng số của mô hình sẽ được tối ưu hoá trong quá trình này.

Sau khi xây dựng xong mô hình, các chỉ số đánh giá hiệu suất như độ chính xác (accuracy), độ nhạy (recall), độ chính xác chuẩn hóa (precision), và điểm F1 (F1-score) sẽ được sử dụng để đánh giá khả năng phân loại của mô hình trên bộ dữ liệu kiểm tra. Những chỉ số này sẽ giúp đánh giá mức độ chính xác và hiệu quả của mô hình trong việc phân loại các mức chất lượng không khí.

Cuối cùng, ứng dụng sẽ được phát triển trên nền tảng *Streamlit*, cho phép người dùng, đặc biệt là các chuyên gia về môi trường, dễ dàng nhập vào các chỉ số ô nhiễm và nhận được kết quả phân loại về chất lượng không khí. Ứng dụng này sẽ cung cấp giao diện người dùng đơn giản, dễ sử dụng và trực quan, giúp người dùng có thể xem được kết quả phân tích cũng như các biểu đồ trực quan để hỗ trợ trong việc đưa ra các quyết định quản lý môi trường.

Nhờ vào phương pháp này, hệ thống phân loại chất lượng không khí sẽ giúp cảnh báo kịp thời về tình trạng ô nhiễm, từ đó cung cấp thông tin quý giá cho các nhà quản lý môi trường và cộng đồng trong việc bảo vệ sức khỏe.



Hình III.3 Quy trình xây dựng mô hình dự đoán

III.1. Giới thiệu tập dữ liệu

Tập dữ liệu chất lượng không khí bao gồm nhiều quan sát về các chỉ số ô nhiễm không khí tại các thành phố khác nhau, cùng với các thông tin liên quan đến chất lượng không khí. Tập dữ liệu được thi thập thông qua trang Kaggle[ ] và cung cấp cái nhìn tổng quan về tình trạng ô nhiễm không khí trên toàn thế giới. Dưới đây là các thông tin dữ liệu

- **country_name**: Tên quốc gia.
- **city_name**: Tên thành của thành phố.
- **aqi_value**: Giá trị AQI chung của thành phố.
- **aqi_category**: Hạng mục AQI tổng thể của thành phố.
- **co_aqi_value**: Giá trị AQI của CO của thành phố.
- **co_aqi_category**: Hạng mục AQI của CO của thành phố.
- **ozone_aqi_value**: Giá trị AQI của Ozone của thành phố
- **ozone_aqi_category**: Hạng mục AQI của Ozone của thành phố.
- **no2_aqi_value**: AQI value of Nitrogen Dioxide of the city.
- **no2_aqi_category**: Hạng mục AQI của Nitrogen Dioxide của thành phố.
- **pm2.5_aqi_value**: Giá trị AQI của Vật chất hạt có đường kính từ 2,5 micromet trở xuống của thành phố.
- **pm2.5_aqi_category**: Loại AQI của Vật chất hạt có đường kính từ 2,5 micromet trở xuống của thành phố.
-

Trực quan hóa dữ liệu

Để đọc dữ liệu từ tệp CSV chúng ta sẽ sử dụng thư viện pandas và sử dụng thư viện matplotlib để trực quan hoá dữ liệu bằng cách vẽ biểu đồ phân phối cho các biến. Điều này giúp chúng ta có cái nhìn tổng quan về dữ liệu trước khi bắt đầu xây dựng mô hình dự đoán bệnh tiểu đường.

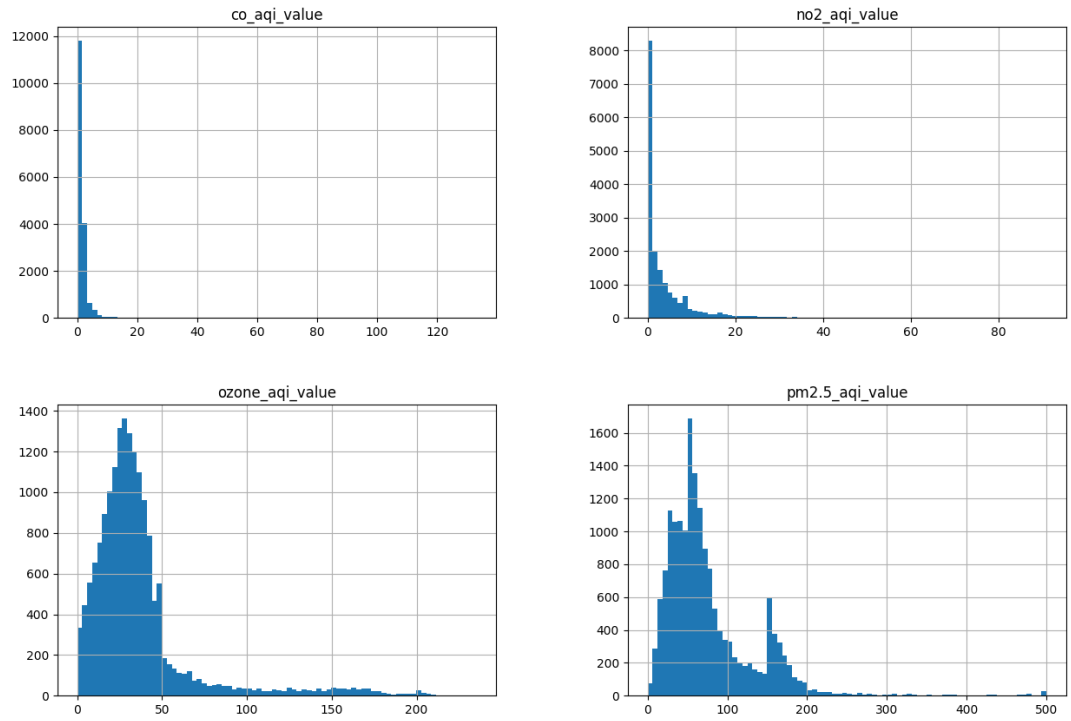
```
df = pd.read_csv('global_air_pollution_data.csv')
df.head()
```

	country_name	city_name	aqi_value	aqi_category	co_aqi_value	co_aqi_category	ozone_aqi_value	ozone_aqi_category	no2_aqi_value	no2_aqi_category	pm2.5_aqi_value	pm2.5_aqi_category
0	Russian Federation	Praskoveya	51	Moderate	1	Good	36	Good	0	Good	51	Moderate
1	Brazil	Presidente Dutra	41	Good	1	Good	5	Good	1	Good	41	Good
2	Italy	Priolo Gargallo	66	Moderate	1	Good	39	Good	2	Good	66	Moderate
3	Poland	Przasnysz	34	Good	1	Good	34	Good	0	Good	20	Good
4	France	Punaauia	22	Good	0	Good	22	Good	0	Good	6	Good

Hinh III.4: Tập dữ liệu đầy đủ

Giá trị nồng độ các chất:

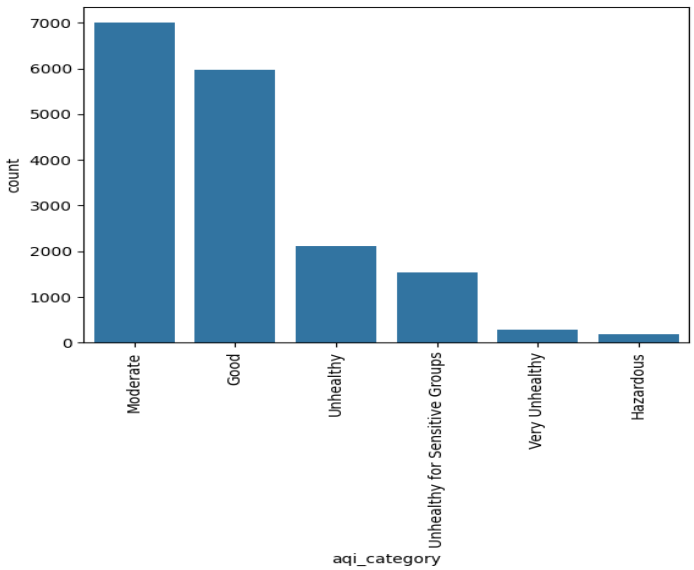
```
# histogram cho numberic
df.hist(bins=80, figsize=(15, 10))
plt.show()
```



Hình III.5 Phân phối các dữ liệu số

Số lượng mẫu tại mỗi lớp:

```
sns.barplot(df.iloc[:,0].value_counts())
plt.xticks(rotation = 90)
plt.show()
```

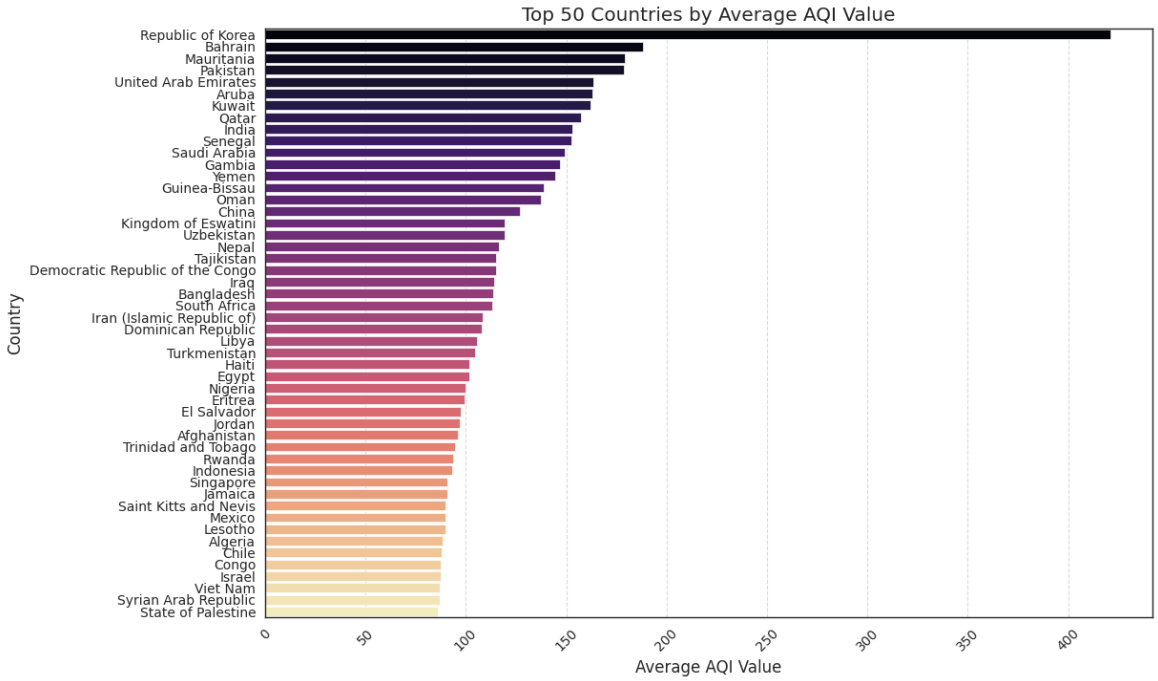


Hình III.6 Phân bố các lớp dữ liệu

Giá trị AQI các nước (50 nước lớn nhất):

```
# Set seaborn style to white
sns.set_style("white")
```

```
# Calculate average AQI value by country
avg_aqi_by_country                                                    =
df.groupby('country_name')['aqi_value'].mean().sort_values(ascending=False)

# Select top 50 countries
top_50_countries = avg_aqi_by_country.head(50)

# Plotting
plt.figure(figsize=(12, 8))
sns.barplot(x=top_50_countries.values, y=top_50_countries.index, palette='magma')
plt.title('Top 50 Countries by Average AQI Value')
plt.xlabel('Average AQI Value')
plt.ylabel('Country')
plt.xticks(rotation=45)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



Hình III.7 Biểu đồ cột ngang 50 nước lớn có AQI lớn nhất

III.2. Tiền xử lý dữ liệu

III.2.1 Trích chọn đặc trưng

Dữ liệu về chất lượng không khí là một tập hợp thông tin đa dạng về các yếu tố như CO, Ozone… Mỗi đặc trưng này cung cấp thông tin quan để đánh giá chất lượng không khí.

Trong trường hợp với bộ dữ liệu này chúng ta cần bỏ qua những đặc trưng không cần thiết để tránh nhiễu cho mô hình:

Với các đặc trưng địa lý **"country_name"** và **"city_name":**

Để mô hình phân loại một cách tổng quá cần xóa đi các đặc trưng địa lý để tập trung vào phân lọa chất lượng không khí thì các cột này sẽ được bỏ đi để giảm kích thước dữ liệu. Ngoài ra, **city_name** thường chứa nhiều giá trị duy nhất (unique) cụ thể 23035 giá trị và có thể phức tạp khi mã hóa thành các biến số.

aqi_value: Mặc dù aqi_value cung cấp một giá trị đo lường chính xác về chất lượng không khí, nhưng vì **aqi_category** đã đại diện cho phân loại chất lượng không khí (ví dụ: tốt, trung bình, xấu), việc giữ lại cả hai sẽ gây dư thừa. Mô hình có thể dựa vào *aqi_category* để phân loại mà không cần sử dụng *aqi_value* vì *aqi_category* đã cung cấp thông tin phân loại chính xác.

Các đặc trưng phân loại khí như co_aqi_category, ozone_aqi_category, no2_aqi_category, pm2.5_aqi_category: Vì chúng chỉ là phân loại của các giá trị AQI đã có (ví dụ: tốt, trung bình, xấu), và chúng được biểu diễn bởi các giá trị AQI riêng lẻ (như co_aqi_value, ozone_aqi_value), việc giữ lại các đặc trưng này có thể gây dư thừa và làm cho mô hình trở nên phức tạp hơn mà không mang lại giá trị gia tăng nào. Mô hình có thể dựa vào các giá trị AQI cụ thể để phân loại mà không cần các danh mục phân loại.

Để tối ưu hóa mô hình, nên loại bỏ các đặc trưng không cần thiết và chỉ giữ lại các biến số liệu chi tiết như *_aqi_value để giúp mô hình tập trung vào các yếu tố quan trọng, cải thiện khả năng dự đoán chất lượng không khí một cách chính xác và hiệu quả.

```
df = df[['aqi_category', 'co_aqi_value','no2_aqi_value','ozone_aqi_value',
'pm2.5_aqi_value']]
```

	aqi_category	co_aqi_value	no2_aqi_value	ozone_aqi_value	pm2.5_aqi_value
0	Moderate	1	0	36	51
1	Good	1	1	5	41
2	Moderate	1	2	39	66
3	Good	1	0	34	20
4	Good	0	0	22	6

Hình III.8 Dataframe đã được lượt đi những đặc trưng không cần thiết cho mô hình

III.2.2 Xử lý dữ liệu

Bước tiếp theo cần thực hiện để chuẩn bị dữ liệu cho máy học là làm sạch dữ liệu đó. Làm sạch dữ liệu liên quan đến việc tìm và sửa lỗi, sự không nhất quán và các giá trị bị thiếu. Có một số cách tiếp cận để làm điều đó:

Xử lý dữ liệu bị thiếu: Thiếu giá trị là một vấn đề phổ biến trong học máy. Nó có thể được xử lý bằng cách quy nạp (điền các giá trị còn thiếu bằng dữ liệu dự đoán hoặc ước tính), nội suy (lấy các giá trị còn thiếu từ các điểm dữ liệu xung quanh) hoặc xóa (chỉ cần xóa các hàng hoặc cột có giá trị bị thiếu khỏi tập dữ liệu.)

```
count_null = df.isna().sum()
print(count_null)
print('trước khi xóa null: ', df.shape)
df = df.dropna()
print('sau khi xóa null: ', df.shape)
```

```
aqi_category       0
co_aqi_value       0
no2_aqi_value      0
ozone_aqi_value    0
pm2.5_aqi_value    0
dtype: int64
trước khi xóa null:  (23463, 5)
sau khi xóa null:  (23463, 5)
```

Hình III.9 Dữ liệu trước và sau khi xóa null

Xử lý các dòng dữ liệu trùng lặp: Một bước khác trong quá trình chuẩn bị dữ liệu cho máy học là loại bỏ các dữ liệu trùng lặp. Các bản sao không chỉ làm sai lệch các dự đoán ML mà còn lãng phí dung lượng lưu trữ và tăng thời gian xử lý, đặc biệt là trong các bộ

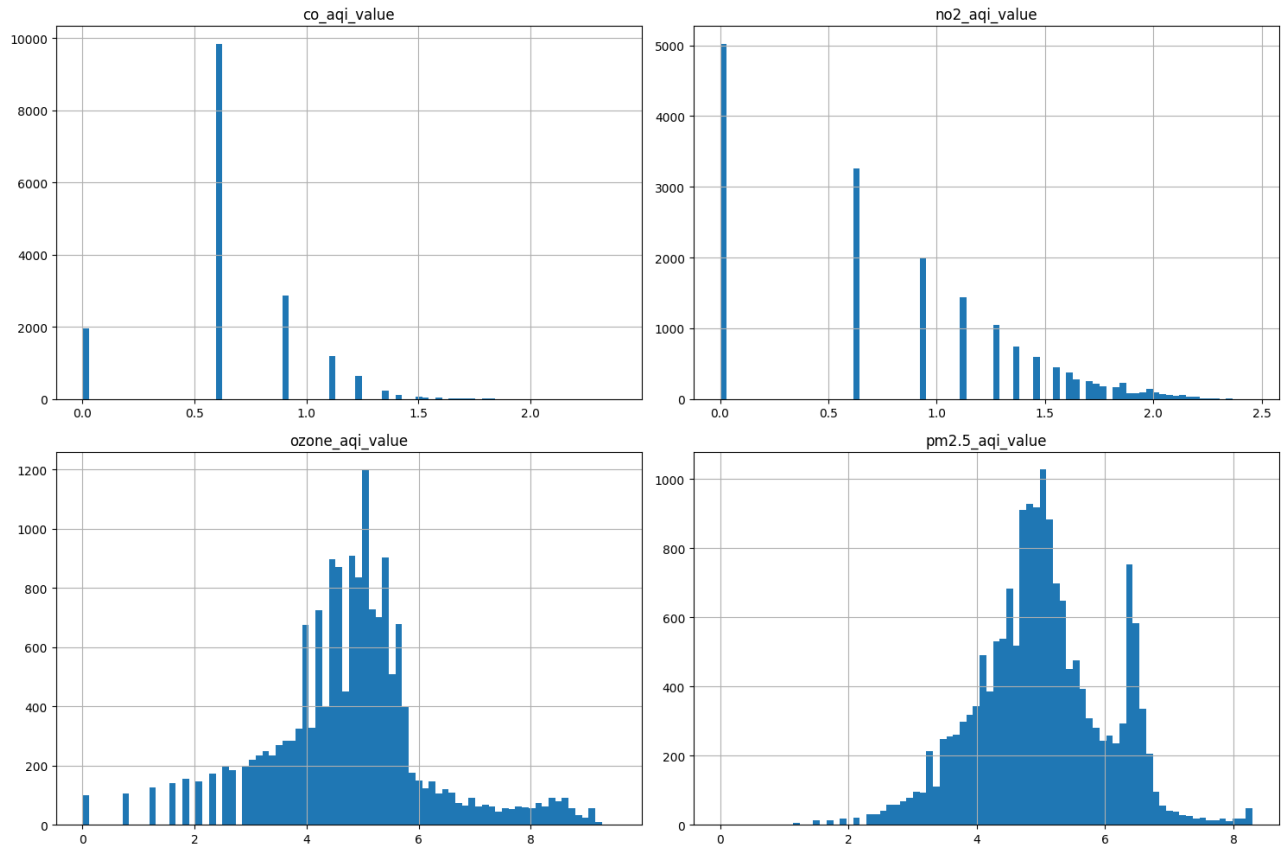
dữ liệu lớn. Để loại bỏ các bản sao, các nhà khoa học dữ liệu sử dụng nhiều kỹ thuật nhận dạng trùng lặp (như khớp chính xác, khớp mờ, băm hoặc liên kết bản ghi). Sau khi được xác định, chúng có thể được loại bỏ hoặc hợp nhất.

```
print('before:', df.shape)
df = df.drop_duplicates().reset_index(drop=True)
print('after:', df.shape)
```

```
before: (23463, 5)
after: (17100, 5)
```

Xử lý dữ liệu phân phối không chuẩn: Dữ liệu phân phối không chuẩn là dữ liệu không tuân theo phân phối chuẩn (phân phối Gauss), với đặc điểm có thể bị lệch (skewed) hoặc có đỉnh (kurtosis) quá cao hoặc quá thấp. Điều này làm cho phân phối dữ liệu không đồng đều và có thể ảnh hưởng đến các phân tích và dự đoán. Các mô hình học máy và thống kê, đặc biệt là những mô hình yêu cầu giả định phân phối chuẩn, như hồi quy tuyến tính, sẽ gặp khó khăn khi làm việc với dữ liệu không chuẩn. Các thuật toán này có thể không học được mối quan hệ chính xác giữa các biến, dẫn đến kết quả sai lệch hoặc không chính xác.

```
# Chọn các biến số để chuẩn hóa (loại bỏ nhãn 'aqi_category')
input_data = df[['co_aqi_value', 'no2_aqi_value', 'ozone_aqi_value',
'pm2.5_aqi_value']]

# Kiểm tra và điều chỉnh dữ liệu nếu có giá trị <= 0
input_data = input_data.apply(lambda x: x + 1 if (x <= 0).any() else x)
#Để lưu giá trị lambda sử dụng cho streamlit
lambda_values = []
for column in input_data.columns:
    _, lambda_value = stats.boxcox(input_data[column])
    lambda_values.append(lambda_value)

# Áp dụng Box-Cox transformation cho từng cột và lưu kết quả
normalized_data = input_data.apply(lambda x: stats.boxcox(x)[0])
# Gán lại các cột đã chuẩn hóa vào DataFrame ban đầu
df[['co_aqi_value', 'no2_aqi_value', 'ozone_aqi_value', 'pm2.5_aqi_value']]
normalized_data
```



Hinh III.10: Phân phối dữ liệu sau khi xử lý lệch phải

Xử lý dữ liệu chuỗi: Vì biến *"aqi_category"* là biến mục tiêu nhưng đặc trưng dạng chuỗi sẽ gây khó khăn trong việc xây dựng mô hình. Chuyển đổi dữ liệu dạng chuỗi thành dạng số bằng cách sử dụng *LabelEncoder* để mô hình học máy có thể hiểu được

```
label_encoder = LabelEncoder()

df['aqi_category'] = label_encoder.fit_transform(df['aqi_category'])

# Hiển thị các nhãn và giá trị tương ứng sau khi chuyển đổi
label_mapping = dict(zip(label_encoder.classes_,
label_encoder.transform(label_encoder.classes_)))
print("Mapping của các nhãn chữ sang số:", label_mapping)
```

III.2.3 Phân chia và chuẩn bị dữ liệu

Phân chia dữ liệu thành tập huấn luyện (train set), và tập xác thực (validate set) và tập kiểm tra (test set) là một bước quan trọng trong quá trình xây dựng mô hình học máy. Mục tiêu của việc phân chia này là để đánh giá khả năng tổng quát của mô hình đối với

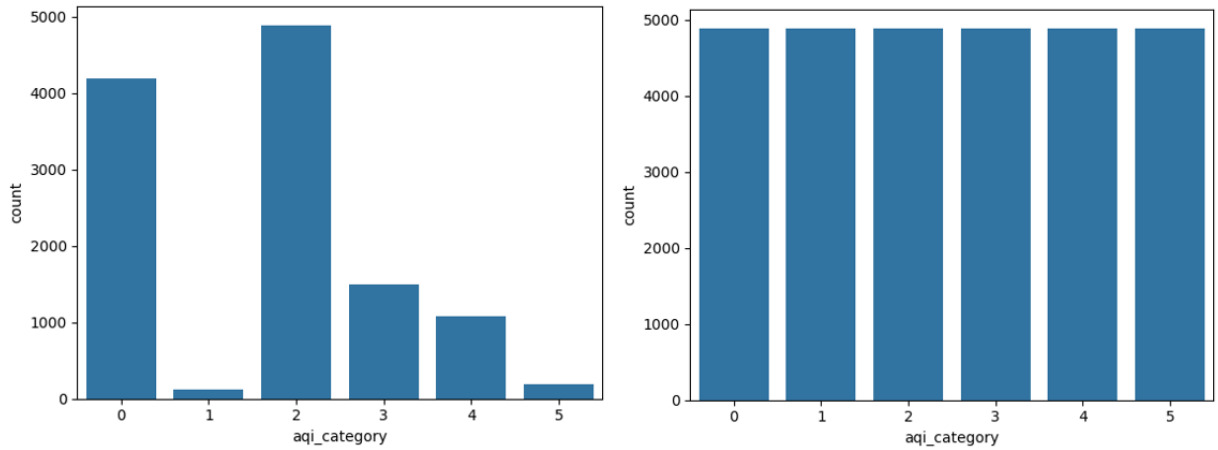
dữ liệu chưa thấy, giúp xác định hiệu quả của mô hình trong các tình huống thực tế. Bộ dữ liệu được thành 3 tập với tỉ lệ: 70% dữ liệu training, 15% validation, 15% testing

```
# Bước 1: Chia dữ liệu thành tập training và temporary (validation + test)
X_train, X_temp, y_train, y_temp = train_test_split(df.iloc[:,1:],df.iloc[:,0],
test_size=1 - train_size, random_state=42)

# Bước 2: Chia temporary thành validation và test
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)
```

Xử lý dữ liệu mất cân bằng: Tập dữ liệu không cân bằng là tập dữ liệu trong đó số điểm dữ liệu trong một lớp thấp hơn đáng kể so với số điểm dữ liệu trong lớp khác. Điều này có thể dẫn đến một mô hình thiên lệch ưu tiên cho tầng lớp đa số, trong khi bỏ qua tầng lớp thiểu số. Để giải quyết vấn đề, các nhóm dữ liệu có thể sử dụng các kỹ thuật như lấy mẫu lại (lấy mẫu quá mức của lớp thiểu số hoặc lấy mẫu dưới lớp đa số để cân bằng việc phân phối dữ liệu), tạo dữ liệu tổng hợp (tạo điểm dữ liệu bổ sung cho lớp thiểu số một cách tổng hợp), chi phí - học tập nhạy cảm (gán trọng số cao hơn cho lớp thiểu số trong quá trình đào tạo), học tập đồng bộ (kết hợp nhiều mô hình được đào tạo trên các tập hợp con dữ liệu khác nhau bằng các thuật toán khác nhau) và các mô hình khác.

Áp dụng kỹ thuật SMOTE để tăng cường mẫu cho các lớp thiểu số trong tập training.

```
# Áp dụng SMOTE để cân bằng dữ liệu trên tập training
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)
```



Hình III.11 Biểu đồ trước và sau khi được cân bằng dữ liệu bằng SMOTE

Phải: Trước khi được cân bằng; Trái: Sau khi được cân bằng

Chú thích: 0: Good, 1: Hazardous, 2: Moderate, 3: Unhealthy, 4: Unhealthy for sensitive group, 5: Very Unhealthy

Chuẩn hóa dữ liệu: là bước quan trọng trong tiền xử lý dữ liệu, đặc biệt khi các đặc trưng có đơn vị và phạm vi khác nhau (ví dụ, chiều cao và thu nhập). Việc này giúp cải thiện hiệu suất của các mô hình học máy, đặc biệt là với những thuật toán như hồi quy tuyến tính, k-NN, và SVM, vốn nhạy cảm với sự khác biệt về quy mô. Chuẩn hóa giúp mô hình học được mối quan hệ chính xác giữa các đặc trưng, giảm thiểu thiên lệch và cải thiện độ chính xác, đồng thời giúp thuật toán hội tụ nhanh hơn và hiệu quả hơn

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_val = scaler.transform(X_val)
```

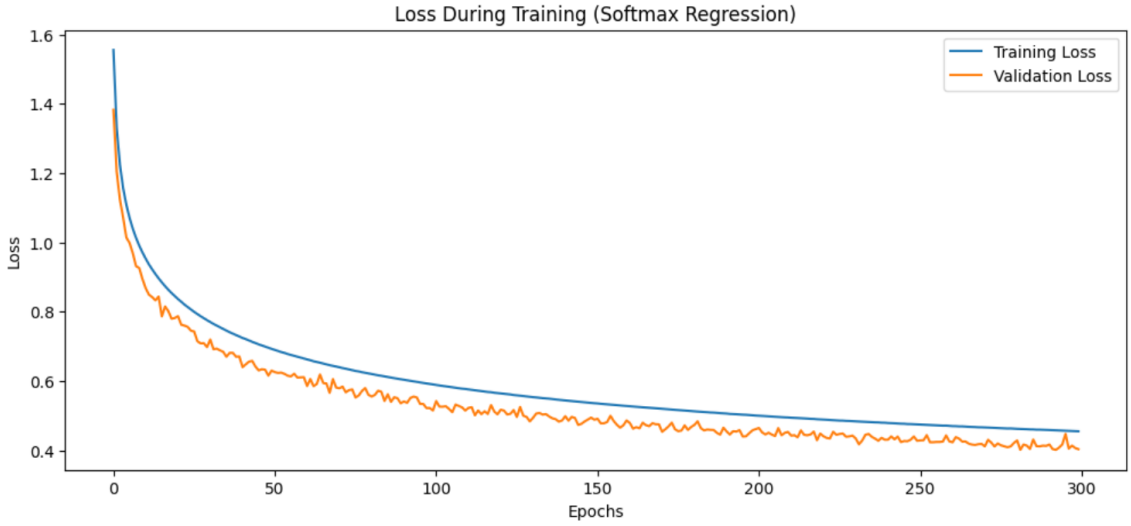
III.3. Xây dựng mô hình dựa trên thuật toán Softmax Regression

```
import numpy as np

class SoftmaxRegression:
    def __init__(self, learning_rate=0.01, n_iter=1000, batch_size=32):
        self.learning_rate = learning_rate
        self.n_iter = n_iter
        self.batch_size = batch_size
        self.weights = None
        self.bias = None
        self.training_losses = []
        self.validation_losses = []
        self.training_accuracies = []
        self.validation_accuracies = []


    def softmax(self, z):
        exp_z = np.exp(z - np.max(z, axis=1, keepdims=True))
        return exp_z / np.sum(exp_z, axis=1, keepdims=True)

    def one_hot_encode(self, y, n_classes):
        return np.eye(n_classes)[y]

    def fit(self, X, y, X_val, y_val):
        n_samples, n_features = X.shape
        n_classes = len(np.unique(y))

        self.weights = np.zeros((n_features, n_classes))
        self.bias = np.zeros(n_classes)

        for i in range(self.n_iter):
            # Shuffle the data to ensure randomization for each mini-batch
```

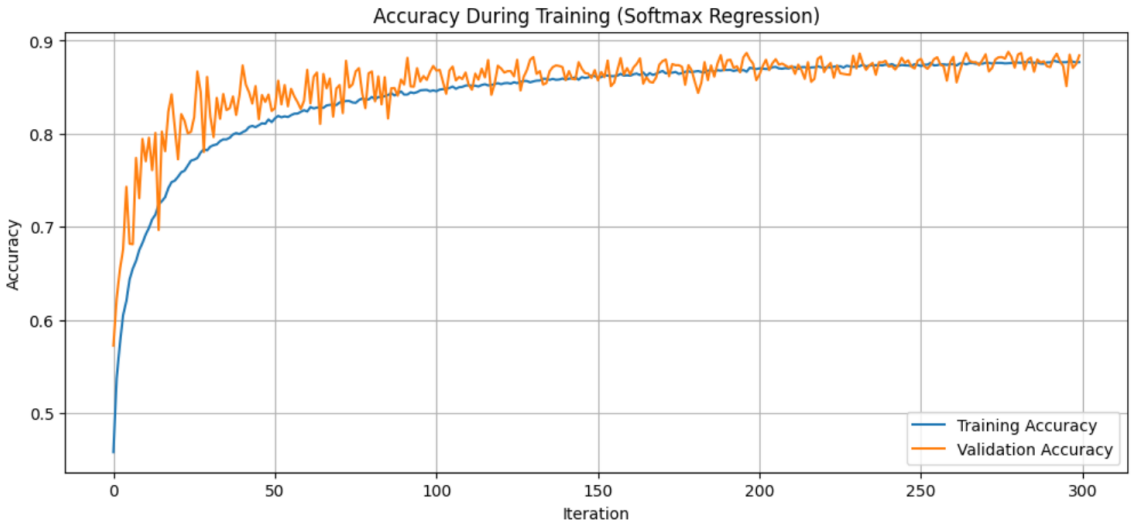
```
            indices = np.random.permutation(n_samples)
            X_shuffled = X[indices]
            y_shuffled = y[indices]

            epoch_training_loss = []
            epoch_training_accuracy = []

            # Loop over mini-batches
            for start_idx in range(0, n_samples, self.batch_size):
                end_idx = min(start_idx + self.batch_size, n_samples)
                X_batch = X_shuffled[start_idx:end_idx]
                y_batch = y_shuffled[start_idx:end_idx]

                # Compute predictions for the mini-batch
                linear_model = np.dot(X_batch, self.weights) + self.bias
                y_predicted = self.softmax(linear_model)


                # Update weights and bias
                dw = (1 / len(y_batch)) * np.dot(X_batch.T, (y_predicted -
self.one_hot_encode(y_batch, n_classes)))
                db = (1 / len(y_batch)) * np.sum(y_predicted -
self.one_hot_encode(y_batch, n_classes), axis=0)
                self.weights -= self.learning_rate * dw
                self.bias -= self.learning_rate * db

                # Compute and store mini-batch loss and accuracy
                batch_loss = self._compute_loss(X_batch, y_batch, n_classes)
                batch_accuracy = self._compute_accuracy(X_batch, y_batch)
                epoch_training_loss.append(batch_loss)
                epoch_training_accuracy.append(batch_accuracy)

            # Record average loss and accuracy for the epoch
            avg_training_loss = np.mean(epoch_training_loss)
            avg_training_accuracy = np.mean(epoch_training_accuracy)
            self.training_losses.append(avg_training_loss)
            self.training_accuracies.append(avg_training_accuracy)

            # Compute loss and accuracy on validation set
            val_loss = self._compute_loss(X_val, y_val, n_classes)
            val_accuracy = self._compute_accuracy(X_val, y_val)

            self.validation_losses.append(val_loss)
            self.validation_accuracies.append(val_accuracy)


    def _compute_loss(self, X, y, n_classes):
        linear_model = np.dot(X, self.weights) + self.bias
        y_predicted = self.softmax(linear_model)
```

```
        y_encoded = self.one_hot_encode(y, n_classes)
        return -np.mean(np.sum(y_encoded * np.log(y_predicted), axis=1))

    def _compute_accuracy(self, X, y):
        y_predicted = self.predict(X)
        return np.mean(y_predicted == y)

    def predict(self, X):
        linear_model = np.dot(X, self.weights) + self.bias
        y_predicted = np.argmax(self.softmax(linear_model), axis=1)
        return y_predicted

# Khởi tạo và huấn luyện mô hình
softmax_model = SoftmaxRegression(learning_rate=0.1, n_iter=300,batch_size= 16)
softmax_model.fit(X_train, y_train, X_val, y_val)
# Vẽ biểu đồ Loss
plt.figure(figsize=(12, 5))
plt.plot(range(softmax_model.n_iter), softmax_model.training_losses,
label='Training Loss')
plt.plot(range(softmax_model.n_iter), softmax_model.validation_losses,
label='Validation Loss')
plt.title('Loss During Training (Softmax Regression)')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
# Vẽ biểu đồ Accuracy
plt.figure(figsize=(12, 5))
plt.plot(range(softmax_model.n_iter), softmax_model.training_accuracies,
label='Training Accuracy')
plt.plot(range(softmax_model.n_iter), softmax_model.validation_accuracies,
label='Validation Accuracy')
plt.title('Accuracy During Training (Softmax Regression)')
plt.xlabel('Iteration')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()

plt.show()
```

Thực thi mô hình dử dụng bộ tham số như sau bộ tham số này đã qua chọn lọc như RanbomSearch hay GridSearch các tham số được chọn dưa trên tiêu chí phù hợp với bộ dữ liệu phản ánh đúng yêu cầu bài toán tốc độ học *(learning_rate):* 0.1, số lần lặp (n_iter): 300, kích thước gói (*batch_size*): 16.



Hinh III.12: Biểu đồ hàm loss của trainning set và validation set



Hinh III.13: Biểu đồ chỉ số accuracy của trainning set và validation set

III.4. Đánh giá mô hình

Accuracy, Classification Report:

```
y_pred_softmax = softmax_model.predict(X_test)
accuracy_softmax = accuracy_score(y_test, y_pred_softmax)
print(f"Softmax Regression Accuracy: {accuracy_softmax:.4f}")
print(classification_report(y_test, y_pred_softmax))
```

```
Softmax Regression Accuracy: 0.8722
              precision    recall  f1-score   support

           0       0.95      0.88      0.91       900
           1       0.94      0.94      0.94        31
           2       0.88      0.91      0.90      1021
           3       0.87      0.76      0.81       321
           4       0.73      0.82      0.77       245
           5       0.44      0.90      0.59        48

    accuracy                           0.87      2566
   macro avg       0.80      0.87      0.82      2566
weighted avg       0.88      0.87      0.88      2566
```

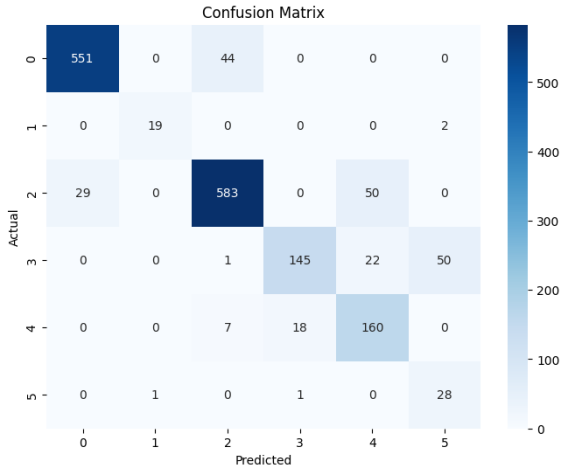
Hinh III.14: Chỉ số accuracy và classification report của mô hình

Tỷ lệ accuracy cho ra khoảng 87% cho thấy mô hình đang hoạt động khá tốt.

- **Lớp 0(Good)** và **Lớp 1(Hazardous)** có độ chính xác, độ bao phủ và điểm F1 cao, cho thấy mô hình hoạt động tốt với các lớp này.
- **Lớp 5(Very Unhealthy)** có độ chính xác thấp nhưng độ bao phủ, cho thấy mô hình nhận diện hầu hết các trường hợp thực sự của lớp 5 nhưng cũng có nhiều dự đoán sai.

Trong bài toán dự đoán chất lượng không khí, ưu tiên Recall là chiến lược hợp lý nhằm đảm bảo không bỏ sót bất kỳ trường hợp không khí xấu nào. Việc phát hiện đầy đủ các nguy cơ tiềm tàng là tối quan trọng, bởi không khí xấu, ngay cả khi chỉ xuất hiện thoáng qua, cũng có thể gây hậu quả nghiêm trọng đến sức khỏe, đặc biệt đối với người già, trẻ em, hoặc người có bệnh nền. Nếu bỏ sót (False Negative) một trường hợp không khí xấu, hậu quả có thể là sự chậm trễ trong việc cảnh báo hoặc thực hiện các biện pháp bảo vệ, dẫn đến tổn thất không thể khắc phục. Mặc dù việc tập trung vào Recall có thể dẫn đến một số cảnh báo sai (False Positive), hậu quả của chúng chỉ dừng lại ở việc thực hiện những biện pháp an toàn không cần thiết, như đeo khẩu trang hay hạn chế ra ngoài. Đây là cái giá nhỏ phải trả so với nguy cơ bỏ qua một mối nguy thực sự. Vì vậy, với tiêu chí "*thà giết lầm còn hơn bỏ sót*", Recall trở thành lựa chọn ưu tiên để bảo vệ sức khỏe cộng đồng một cách hiệu quả nhất.

Confusion Matrix.



Hình III.15: Ma trận nhầm lẫn của mô hình Softmax Regression

Chú thích: 0: Good, 1: Hazardous, 2: Moderate, 3: Unhealthy, 4: Unhealthy for sensitive group, 5: Very Unhealthy

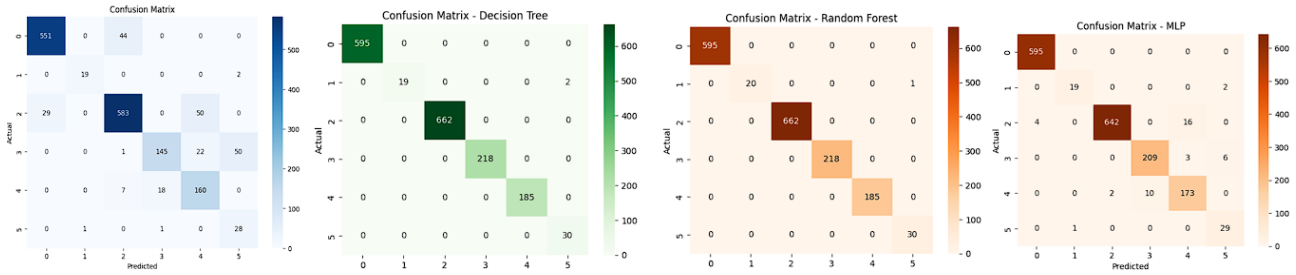
III.5. So sánh mô hình Softmax Regression tự xây dựng với các mô hình cùng loại của Sklearn

Decision Tree: Cây quyết định là một thuật toán học máy dễ hiểu và trực quan, hoạt động bằng cách chia dữ liệu thành các nhánh dựa trên các thuộc tính. Mỗi nút trong cây đại diện cho một thuộc tính hoặc điều kiện kiểm tra, trong khi các nhánh biểu diễn các kết quả của điều kiện đó. Đối với bài toán phân loại đa lớp, cây quyết định chia nhánh nhiều lần để bao quát tất cả các nhãn có trong tập dữ liệu. Một cách phổ biến để xây dựng cây là sử dụng các tiêu chí như thông tin thu được (information gain) hoặc chỉ số Gini để chọn thuộc tính tối ưu cho mỗi bước chia. Cây quyết định thường dễ bị overfitting trên các tập dữ liệu nhỏ hoặc phức tạp, nhưng lại đơn giản để triển khai và diễn giải. Nhược điểm là khi số lớp và thuộc tính tăng lên, cây có thể trở nên cồng kềnh, dẫn đến hiệu suất giảm trên dữ liệu mới.

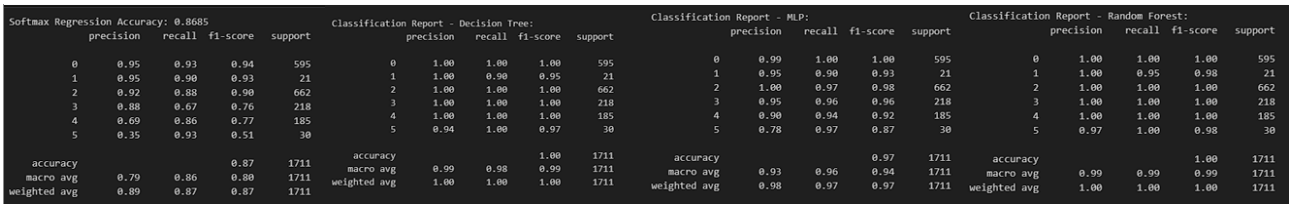
Random Forest: Rừng ngẫu nhiên là một tập hợp của nhiều cây quyết định, nơi mỗi cây được đào tạo trên một mẫu dữ liệu ngẫu nhiên và một tập con của các thuộc tính. Điều này tạo nên sự đa dạng và giúp giảm thiểu overfitting. Trong bài toán phân loại đa lớp, mỗi cây đưa ra dự đoán cho một nhãn cụ thể. Sau đó, các nhãn này được kết hợp lại bằng cách sử dụng nguyên tắc "đa số phiếu" hoặc tính toán xác suất trung bình để đưa ra dự

đoán cuối cùng. Random Forest có khả năng mở rộng tốt cho các bài toán đa lớp mà không cần thay đổi lớn trong cấu trúc. Đây là một thuật toán mạnh mẽ, hiệu quả và ít nhạy cảm với dữ liệu nhiễu. Tuy nhiên, nhược điểm của Random Forest là yêu cầu nhiều tài nguyên tính toán hơn so với cây quyết định đơn lẻ.

Multilayer Perceptron: Multilayer Perceptron là một dạng mạng nơ-ron sâu, gồm nhiều lớp kết nối đầy đủ giữa các nơ-ron. Đối với bài toán phân loại đa lớp, MLP sử dụng một lớp đầu ra với số nơ-ron bằng số nhãn trong tập dữ liệu và áp dụng hàm kích hoạt softmax để tính toán xác suất của từng nhãn. Hàm softmax đảm bảo tổng xác suất bằng 1, giúp mạng đưa ra dự đoán chính xác bằng cách chọn nhãn có xác suất cao nhất. MLP thường được đào tạo bằng phương pháp lan truyền ngược (backpropagation) và tối ưu hóa dựa trên thuật toán gradient descent. Thuật toán này có khả năng xử lý các bài toán phức tạp với dữ liệu phi tuyến tính, nhưng đòi hỏi thời gian đào tạo dài và có thể cần điều chỉnh siêu tham số cẩn thận để tránh overfitting hoặc underfitting.



Hình III.16: Confusion Matrix của 4 thuật toán



Softmax Regression Accuracy: 0.8685

	precision	recall	f1-score	support
0	0.95	0.93	0.94	595
1	0.95	0.90	0.93	21
2	0.92	0.88	0.90	662
3	0.88	0.67	0.76	218
4	0.69	0.86	0.77	185
5	0.35	0.93	0.51	30
accuracy			0.87	1711
macro avg	0.79	0.86	0.80	1711
weighted avg	0.89	0.87	0.87	1711

Classification Report - Decision Tree:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	595
1	1.00	0.90	0.95	21
2	1.00	1.00	1.00	662
3	1.00	1.00	1.00	218
4	1.00	1.00	1.00	185
5	0.94	1.00	0.97	30
accuracy			1.00	1711
macro avg	0.99	0.98	0.99	1711
weighted avg	1.00	1.00	1.00	1711

Classification Report - MLP:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	595
1	0.95	0.90	0.93	21
2	1.00	0.97	0.98	662
3	0.95	0.96	0.96	218
4	0.90	0.94	0.92	185
5	0.78	0.97	0.87	30
accuracy			0.97	1711
macro avg	0.93	0.96	0.94	1711
weighted avg	0.98	0.97	0.97	1711

Classification Report - Random Forest:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	595
1	1.00	0.95	0.98	21
2	1.00	1.00	1.00	662
3	1.00	1.00	1.00	218
4	1.00	1.00	1.00	185
5	0.97	1.00	0.98	30
accuracy			1.00	1711
macro avg	0.99	0.99	0.99	1711
weighted avg	1.00	1.00	1.00	1711

Hình III.17: Classification Report của 4 thuật toán

Chú thích: 0: Good, 1: Hazardous, 2: Moderate, 3: Unhealthy, 4: Unhealthy for sensitive group, 5: Very Unhealthy

IV. Xây dựng Demo

Hàm nhận giá trị đầu vào từ người dùng

```
def input_features():
    co = st.sidebar.slider('CO', 0, 500,1)
    no2 = st.sidebar.slider('NO2', 0, 500,2)
    ozone = st.sidebar.slider('Ozone', 0, 500,3)
    pm25 = st.sidebar.slider('PM2.5', 0, 500,4)
    data = {
        'co_aqi_value' : co,
        'no2_aqi_value' : no2,
        'ozone_aqi_value' : ozone,
        'pm2.5_aqi_value' : pm25
    }
    features = pd.DataFrame(data , index=[0])
    return features
```

Hàm xử lí giá trị đầu (khớp với mô hình)

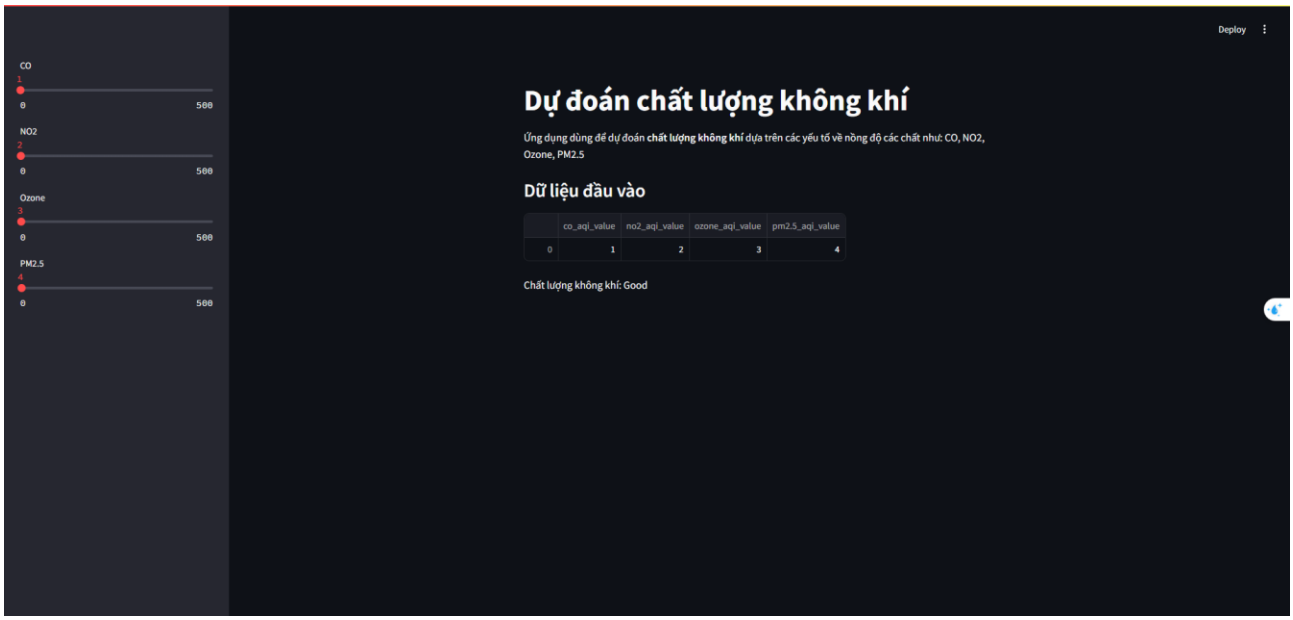
```
# Hàm xử lý dữ liệu trước khi dự đoán (Box-Cox Transformation)
def preprocess_input_data(data):
    # Tải giá trị lambda đã huấn luyện trước cho Box-Cox
    lambda_values = joblib.load('boxcox_lambda_values.pkl')

    # Thêm 1 vào dữ liệu để đảm bảo nó dương (nếu cần thiết)
    data_transformed = data.copy()
    for column in data.columns:
        if (data[column] <= 0).any():  # Kiểm tra nếu có giá trị <= 0
            data_transformed[column] += 1  # Thêm 1 chỉ vào các giá trị <= 0

    # Áp dụng Box-Cox transformation cho từng cột
    for i, column in enumerate(data.columns):
        data[column]= stats.boxcox(data_transformed[column],
lmbda=lambda_values[i])

    # Chuẩn hóa MinMax
    minmax_scaler = joblib.load('minmax_scaler.pkl')
    data_scaled = minmax_scaler.transform(data)

    return data_scaled
```

Tải model đã train

```
softmax_model = joblib.load('softmax_model.pkl')
```

Thực hiện dự đoán dựa trên model đã train

```
y_pred = softmax_model.predict(data)
predicted_classes_name = class_names[y_pred[0]]
st.write('Chất lượng không khí:',predicted_classes_name)
```



Hình IV.1: Giao diện chương trình

V. Kết luận

Thực hiện phân tích và dự đoán mức độ ô nhiễm không khí dựa trên các chỉ số AQI của các chất ô nhiễm khác nhau như CO, NO2, Ozone, và PM2.5. Các bước triển khai mô hình bao gồm chuẩn hóa dữ liệu, trích chọn đặc trưng, cân bằng dữ liệu, và áp dụng thuật toán hồi quy Softmax.

Các kết quả đạt được:

Đối với mô hình Softmax Regression

- **Độ chính xác** trên tập kiểm tra của mô hình Softmax Regression đạt **87.22%**, thể hiện khả năng phân loại tốt các mức độ ô nhiễm không khí.
- Các chỉ số như **Precision**, **Recall**, và **F1-score** cho các lớp ô nhiễm cho thấy mô hình hoạt động tốt đối với một số lớp (như lớp 0(Good), 1(Hazardous), 2(Moderate), nhưng hiệu suất giảm đối với các lớp ít mẫu hơn, ví dụ lớp 5.
- **Ma trận nhầm lẫn** (Confusion Matrix) đã giúp chỉ ra rằng mô hình phân loại chính xác các lớp ô nhiễm với độ chính xác cao đối với các lớp có số lượng mẫu lớn, nhưng còn gặp khó khăn trong việc phân loại các lớp ít mẫu (như lớp 5(Very Unhealthy)).

Đối với mô hình Decision Tree

- **Độ chính xác** trên tập kiểm tra của mô hình Softmax Regression đạt **gần như 100%** thể hiện khả năng phân loại tốt tất cả các mức độ ô nhiễm không khí.
- Các chỉ số như **Precision**, **Recall**, và **F1-score** cho ra kết quả gần 100% tuy nhiên chỉ số recall ở lớp 1 (Hazardous) 90% cho thấy lớp này vẫn có độ nhớ cao nhưng thấp hơn so với các lớp còn lại
- **Ma trận nhầm lẫn** (Confusion Matrix) đã giúp chỉ ra rằng mô hình phân loại chính xác các lớp ô nhiễm với độ chính xác cao (chỉ có 2 mẫu ở lớp 1(Hazardous) dự đoán sai) có thể vì 1 số lý do như số lượng mẫu thử ít

Đối với mô hình Random Forest

- **Độ chính xác** trên tập kiểm tra của mô hình Random Forest đạt **gần như 100%** thể hiện khả năng phân loại tốt tất cả các mức độ ô nhiễm không khí.

- Các chỉ số như **Precision**, **Recall**, và **F1-score** cho ra kết quả rất cao đối với tất cả các lớp thấp nhất là chỉ số recall ở lớp 1 (Hazardous) nhưng vẫn rất cao (95%), cho thấy mô hình không nhạy cảm để phát hiện ra lớp này hơn.

- **Ma trận nhầm lẫn** (Confusion Matrix) đã giúp chỉ ra rằng mô hình phân loại chính xác các lớp ô nhiễm với độ chính xác cao (chỉ có 1 mẫu ở lớp 1(Hazardous) dự đoán sai) có thể vì 1 số lý do như số lượng mẫu thử ít

Đối với mô hình MLP

- **Độ chính xác** trên tập kiểm tra của mô hình Random Forest đạt **97%** thể hiện khả năng phân loại tốt tất cả các mức độ ô nhiễm không khí.

- Các chỉ số như **Precision**, **Recall**, và **F1-score** cho ra kết quả rất cao đối với tất cả các lớp. Precision thấp nhất (0.78), nhưng Recall cao nhất (0.97). Điều này cho thấy mô hình nhận diện gần như tất cả các mẫu thuộc lớp 5(Very Unhealthy), nhưng có nhiều dự đoán nhầm từ các lớp khác.

- **Ma trận nhầm lẫn** (Confusion Matrix): Đặc biệt trên các lớp lớn như lớp 0 (Good) và lớp 2 (Moderate). Tuy nhiên, vẫn còn lỗi nhầm lẫn ở các lớp nhỏ và các lớp gần nhau (3 (Unhealthy), 4 (Unhealthy for sensitive group). Việc cải thiện cân bằng dữ liệu và tăng độ phân biệt giữa các lớp sẽ giúp nâng cao hiệu suất của mô hình.

Đánh giá và hướng phát triển:

- Mô hình hiện tại đạt được kết quả khá ổn định với độ chính xác cao, nhưng vẫn cần cải thiện trong việc phân loại chính xác các lớp có ít mẫu. Việc sử dụng các phương pháp cân bằng dữ liệu tốt hơn hoặc thử nghiệm với các thuật toán khác có thể giúp nâng cao hiệu suất của mô hình.

- Các kỹ thuật như **ensemble learning** hoặc **deep learning** có thể là những hướng tiếp theo để cải thiện mô hình, đặc biệt là đối với các lớp có ít mẫu và phân loại khó.

Hướng phát triển của đề tài:

- Tiếp tục nghiên cứu các mô hình học máy khác để cải thiện hiệu quả phân loại, đặc biệt là những lớp ô nhiễm có ít mẫu dữ liệu.
- Cải thiện dữ liệu ở mức ô nhiễm cao, từ đó dự đoán chính xác hơn không khí tệ cụ thể từng khu vực (đặc trưng không khí mỗi vùng) từ đó đưa ra cảnh báo cho cơ quan dự báo và có những biện pháp phòng tránh sớm.
- Cải thiện quá trình xử lý và chuẩn hóa dữ liệu, bao gồm việc sử dụng các kỹ thuật như **feature engineering** để trích xuất đặc trưng có giá trị hơn.
- Phát triển một ứng dụng trực quan để người dùng có thể dễ dàng theo dõi chất lượng không khí theo thời gian thực tại từng quốc gia và vị trí địa lý khác nhau

Tài liệu tham khảo (IEEE)

[1] WHO: Mỗi năm có 7 triệu người chết vì ô nhiễm không khí - Tuổi Trẻ Online (tuoitre.vn)

[2] WHO – Air Pollution - Air pollution (who.int)

[3] AQI - Chỉ số chất lượng không khí (AQI) và thông tin về ô nhiễm không khí | IQAir

[4] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6), 386-408.

[5] Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning Representations by Back-Propagating Errors. *Nature*, 323(6088), 533-536.

[6] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

[7] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice Hall.

[8] **Andrew Ng**: Coursera Machine Learning

[9] Ian Goodfellow, Yoshua Bengio, Aaron Courville. "Deep Learning." MIT Press, 2016.

[10] V. H. Tiệp, “Machine learning cơ bản,” 28 12 2016. [Trực tuyến]. Available: https://machinelearningcoban.com/2016/12/28/linearregression