**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: DỮ LIỆU LỚN**

**ĐỀ TÀI: DỰ ĐOÁN Ô NHIỄM KHÔNG KHÍ Ở NEWYORK**

**Giáo viên hướng dẫn: T.s Trần Quý Nam**

**T.s Lê Thị Thùy Trang**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stt** | **Mã sv** | **Họ và tên** | **Lớp** |
| **1** | **1671020261** | **Bạch Công Quân** | **CNTT 16-02** |
| **2** | **1671020273** | **Vũ Tài Sang** | **CNTT 16-02** |
| **3** | **1671020153** | **Nguyễn Quốc Hưng** | **CNTT 16-02** |
| **4** | **1671020191** | **Nguyễn Văn Long** | **CNTT 16-02** |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: DỮ LIỆU LỚN**

**ĐỀ TÀI:** **DỰ ĐOÁN Ô NHIỄM KHÔNG KHÍ Ở NEWYORK**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SSTT** | **Mã Sinh Viên** | **Họ và Tên** | **Sinh Ngày** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **1** | **1671020261** | **Bạch Công Quân** | **06/06/2004** |  |  |
| **2** | **1671020273** | **Vũ Tài Sang** | **11/07/2004** |  |  |
| **3** | **1671020153** | **Nguyễn Quốc Hưng** | **25/09/2004** |  |  |
| **4** | **1671020191** | **Nguyễn Văn Long** | **31/10/2004** |  |  |

**CÁN BỘ CHẤM THI** **1 CÁN BỘ CHẤM THI 2**

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Ô nhiễm không khí đã trở thành một trong những thách thức môi trường lớn nhất đối với các thành phố lớn trên toàn cầu, đặc biệt là những đô thị phát triển nhanh chóng như New York. Theo Tổ chức Y tế Thế giới (WHO), hơn 90% dân số thế giới đang phải hít thở không khí ô nhiễm, dẫn đến khoảng 7 triệu ca tử vong sớm mỗi năm. Trong bối cảnh đó, việc phân tích và dự đoán mức độ ô nhiễm không khí đóng vai trò quan trọng trong việc hoạch định chính sách và bảo vệ sức khỏe cộng đồng.

Nghiên cứu này tập trung vào việc ứng dụng các công nghệ Big Data và học máy để phân tích và dự đoán ô nhiễm không khí tại New York. Bằng cách kết hợp phương pháp phân tích dữ liệu tiên tiến với các thuật toán học máy như hồi quy tuyến tính, cây quyết định, rừng ngẫu nhiên và Gradient Boosted Trees, nghiên cứu nhằm mục đích xây dựng các mô hình dự đoán có độ chính xác cao, đồng thời cung cấp cái nhìn sâu sắc về các yếu tố ảnh hưởng đến chất lượng không khí.

Với sự phát triển của Internet of Things (IoT) và các hệ thống quan trắc môi trường tự động, khối lượng dữ liệu về ô nhiễm không khí ngày càng tăng nhanh về cả số lượng và độ phức tạp. Điều này đặt ra yêu cầu cấp thiết về việc sử dụng các công cụ phân tích dữ liệu lớn như Apache Spark, không chỉ để xử lý hiệu quả khối lượng thông tin khổng lồ mà còn để trích xuất những thông tin có giá trị từ dữ liệu.

Kết quả của nghiên cứu này không chỉ có ý nghĩa học thuật mà còn có giá trị thực tiễn cao trong việc hỗ trợ các nhà hoạch định chính sách đưa ra quyết định dựa trên dữ liệu, đồng thời cung cấp cho người dân thông tin kịp thời về chất lượng không khí để có biện pháp bảo vệ sức khỏe phù hợp.

**MỤC LỤC**

**[Chương 1: Lý thuyết Big Data và Ứng dụng PySpark](#_Toc193077221)** [6](#_Toc193077221)

**[1.1 Tổng quan về Big Data](#_Toc193077222)** [6](#_Toc193077222)

*[1.1.1 Định nghĩa Big Data](#_Toc193077223)* [6](#_Toc193077223)

*[1.1.2 Nguồn gốc và sự phát triển của Big Data](#_Toc193077224)* [7](#_Toc193077224)

*[1.1.3 Thách thức trong xử lý Big Data](#_Toc193077225)* [7](#_Toc193077225)

**[1.2 Hệ sinh thái Hadoop](#_Toc193077226)** [7](#_Toc193077226)

*[1.2.1 Kiến trúc Hadoop](#_Toc193077227)* [7](#_Toc193077227)

*[1.2.2 Các công nghệ trong hệ sinh thái Hadoop](#_Toc193077228)* [8](#_Toc193077228)

*[1.2.3 Giới hạn của Hadoop MapReduce](#_Toc193077229)* [8](#_Toc193077229)

**[1.3 Apache Spark và PySpark](#_Toc193077230)** [9](#_Toc193077230)

*[1.3.1 Giới thiệu về Apache Spark](#_Toc193077231)* [9](#_Toc193077231)

*[1.3.2 Kiến trúc Spark](#_Toc193077232)* [9](#_Toc193077232)

*[1.3.3 PySpark - Python API cho Apache Spark](#_Toc193077233)* [10](#_Toc193077233)

**[1.4 Mô hình lập trình Spark](#_Toc193077234)** [10](#_Toc193077234)

*[1.4.1 Resilient Distributed Datasets (RDDs)](#_Toc193077235)* [10](#_Toc193077235)

*[1.4.2 DataFrames và Datasets](#_Toc193077236)* [11](#_Toc193077236)

*[1.4.3 Spark SQL](#_Toc193077237)* [11](#_Toc193077237)

*[1.4.4 Streaming trong PySpark](#_Toc193077238)* [11](#_Toc193077238)

*[1.4.5 So sánh spark và mapreduce](#_Toc193077239)* [12](#_Toc193077239)

**[1.5 Ứng dụng của Big Data và PySpark](#_Toc193077240)** [12](#_Toc193077240)

*[1.5.1 Phân tích dữ liệu lớn](#_Toc193077241)* [12](#_Toc193077241)

*[1.5.2 Học máy với PySpark MLlib](#_Toc193077242)* [12](#_Toc193077242)

**[CHƯƠNG 2. MÔ TẢ TẬP DỮ LIỆU VÀ CÔNG NGHỆ SỬ DỤNG](#_Toc193077243)** [13](#_Toc193077243)

**[2.1 Tệp dữ liệu Air\_Quality Dataset](#_Toc193077244)** [13](#_Toc193077244)

*[2.1.1 Cấu trúc dữ liệu](#_Toc193077245)* [13](#_Toc193077245)

**[2.2 Apache Spark](#_Toc193077246)** [14](#_Toc193077246)

*[2.2.1 Kiến trúc của Apache Spark](#_Toc193077247)* [14](#_Toc193077247)

*[2.2.2 Đặc điểm nổi bật của Apache Spark](#_Toc193077248)* [14](#_Toc193077248)

*[2.2.3 Ứng dụng thực tế của Apache Spark](#_Toc193077249)* [15](#_Toc193077249)

**[2.3 Giới thiệu về Apache Hadoop](#_Toc193077250)** [15](#_Toc193077250)

*[2.3.1 Kiến trúc của Apache Hadoop](#_Toc193077251)* [15](#_Toc193077251)

*[2.3.2 Ưu điểm của Apache Hadoop](#_Toc193077252)* [16](#_Toc193077252)

*[2.3.3 Thách thức của Apache Hadoop](#_Toc193077253)* [16](#_Toc193077253)

*[2.3.4 Ứng dụng thực tế của Apache Hadoop](#_Toc193077254)* [16](#_Toc193077254)

**[CHƯƠNG 3 : KẾT QUẢ XỬ LÝ, PHÂN TÍCH DỮ LIỆU](#_Toc193077255)** [17](#_Toc193077255)

**[3.1 Phương pháp nghiên cứu](#_Toc193077256)** [17](#_Toc193077256)

**[3.2 Phân tích dữ liệu](#_Toc193077257)** [17](#_Toc193077257)

*[3.2.1 Import các thư viện cần thiết](#_Toc193077258)* [17](#_Toc193077258)

*[3.2.2 Khởi tạo Spark](#_Toc193077259)* [17](#_Toc193077259)

*[3.2.3 In dòng dữ liệu](#_Toc193077260)* [17](#_Toc193077260)

*[3.2.4 Tiền xử lý dữ liệu](#_Toc193077261)* [18](#_Toc193077261)

*[3.2.5 Đếm giá trị null trong mỗi cột](#_Toc193077262)* [18](#_Toc193077262)

*[3.2.6 Chuyển SparkDatafame qua pandas để hiển thị mô hình](#_Toc193077263)* [19](#_Toc193077263)

*[3.2.7 Biểu đồ tròn: Phân bố các loại ô nhiễm theo cột "Name"](#_Toc193077264)* [19](#_Toc193077264)

*[3.2.11 Lọc điều kiện](#_Toc193077265)* [23](#_Toc193077265)

*[3.2.12 Chia dữ liệu trành Train và Test](#_Toc193077266)* [23](#_Toc193077266)

*[3.2.13 Tạo và áp dụng pipline](#_Toc193077267)* [25](#_Toc193077267)

*[3.2.14 Áp dụng vào pipeline](#_Toc193077268)* [26](#_Toc193077268)

*[3.2.15 Huấn luyện mô hình hồi quy tuyến tính](#_Toc193077269)* [27](#_Toc193077269)

*[3.2.16 Tính toán thống kê](#_Toc193077270)* [28](#_Toc193077270)

*[3.2.17 Dự đoán](#_Toc193077271)* [29](#_Toc193077271)

*[3.2.21 Cải tiến với các mô hình khác nhau](#_Toc193077272)* [33](#_Toc193077272)

*[3.2.22 Thử với mô hình khác](#_Toc193077273)* [34](#_Toc193077273)

*[3.2.23 Thử nghiệm với các mô hình khác](#_Toc193077274)* [35](#_Toc193077274)

*[3.2.24 So sánh hiệu xuất – dự đoán của các mô hình](#_Toc193077275)* [37](#_Toc193077275)

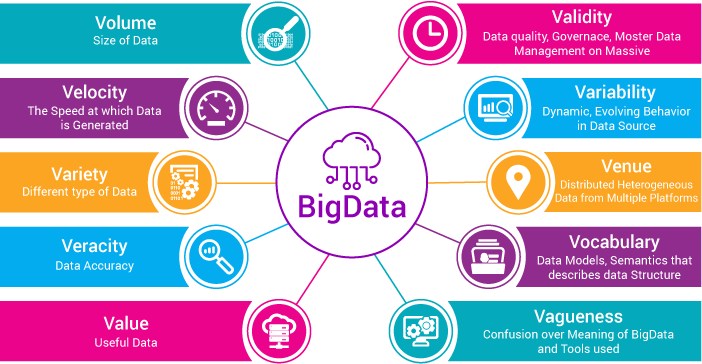
**[KẾT LUẬN 39](#_Toc193077276)**

**[Tài liệu tham khảo 40](#_Toc193077277)**

# **Chương 1: Lý thuyết Big Data và Ứng dụng PySpark**

## **1.1 Tổng quan về Big Data**

### *1.1.1 Định nghĩa Big Data*



Hình 1 . Bigdata

Big Data được hiểu là các tập dữ liệu có kích thước lớn, phức tạp và đa dạng đến mức các phương pháp xử lý truyền thống không thể xử lý hiệu quả. Hiện nay, Big Data thường đặc trưng bởi các khái niệm 5V:

* **Volume (Khối lượng)**: Số lượng dữ liệu được tạo ra và lưu trữ. Khối lượng dữ liệu lớn yêu cầu các giải pháp lưu trữ và xử lý đặc biệt.
* **Velocity (Tốc độ)**: Tốc độ dữ liệu được tạo ra và xử lý. Nhiều ứng dụng hiện đại cần xử lý dữ liệu theo thời gian thực.
* **Variety (Đa dạng)**: Sự đa dạng về loại và nguồn dữ liệu bao gồm dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc.
* **Veracity (Độ chính xác)**: Đề cập đến chất lượng và độ tin cậy của dữ liệu.
* **Value (Giá trị)**: Khả năng chuyển đổi dữ liệu thành thông tin có giá trị và hữu ích.

### *1.1.2 Nguồn gốc và sự phát triển của Big Data*

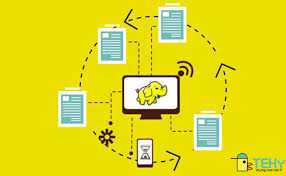
Khái niệm Big Data xuất hiện vào đầu những năm 2000 khi các tổ chức bắt đầu nhận thấy sự gia tăng đáng kể về lượng dữ liệu họ phải quản lý. Sự phát triển của Internet, mạng xã hội, thiết bị IoT và các ứng dụng kỹ thuật số đã dẫn đến sự bùng nổ dữ liệu toàn cầu.

Google đã tiên phong với việc phát triển các công nghệ như Google File System (GFS) và MapReduce để xử lý khối lượng dữ liệu lớn. Năm 2006, Apache Hadoop ra đời như một dự án mã nguồn mở dựa trên các ý tưởng của Google, làm nền tảng cho nhiều công nghệ Big Data hiện nay.

### *1.1.3 Thách thức trong xử lý Big Data*

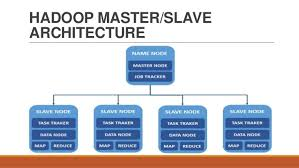
* **Lưu trữ và quản lý**: Cần các hệ thống phân tán có khả năng mở rộng.
* **Xử lý và phân tích**: Yêu cầu thuật toán hiệu quả và có khả năng xử lý phân tán.
* **Bảo mật và quyền riêng tư**: Bảo vệ dữ liệu nhạy cảm trong môi trường phân tán.
* **Chất lượng dữ liệu**: Đảm bảo tính nhất quán và chính xác của dữ liệu.
* **Hiệu suất thời gian thực**: Xử lý nhanh chóng các luồng dữ liệu liên tục.

## **1.2 Hệ sinh thái Hadoop**



Hình 2: Hadoop

### *1.2.1 Kiến trúc Hadoop*



Hình 3: Kiến trúc Hadoop

Apache Hadoop là một framework mã nguồn mở cho phép xử lý dữ liệu lớn trên các cụm máy tính sử dụng mô hình lập trình đơn giản. Hadoop bao gồm các thành phần chính:

* **Hadoop Distributed File System (HDFS)**: Hệ thống tệp phân tán cung cấp khả năng lưu trữ lớn và hiệu suất cao.
* **MapReduce**: Mô hình lập trình cho phép xử lý dữ liệu lớn song song và phân tán.
* **YARN (Yet Another Resource Negotiator)**: Quản lý tài nguyên và lập lịch công việc trong cụm.
* **Hadoop Common**: Thư viện và tiện ích được chia sẻ bởi các dự án Hadoop.

### *1.2.2 Các công nghệ trong hệ sinh thái Hadoop*

* **Apache Hive**: Cung cấp giao diện giống SQL để truy vấn dữ liệu trong HDFS.
* **Apache HBase**: Cơ sở dữ liệu NoSQL phân tán, định hướng cột.
* **Apache Pig**: Ngôn ngữ cấp cao để phân tích dữ liệu lớn.
* **Apache Sqoop**: Công cụ truyền dữ liệu giữa HDFS và cơ sở dữ liệu quan hệ.
* **Apache Flume**: Dịch vụ phân tán để thu thập, tổng hợp và di chuyển dữ liệu.
* **Apache Oozie**: Hệ thống lập lịch công việc để quản lý các công việc Hadoop.
* **Apache ZooKeeper**: Dịch vụ điều phối cao cho các ứng dụng phân tán.

### *1.2.3 Giới hạn của Hadoop MapReduce*

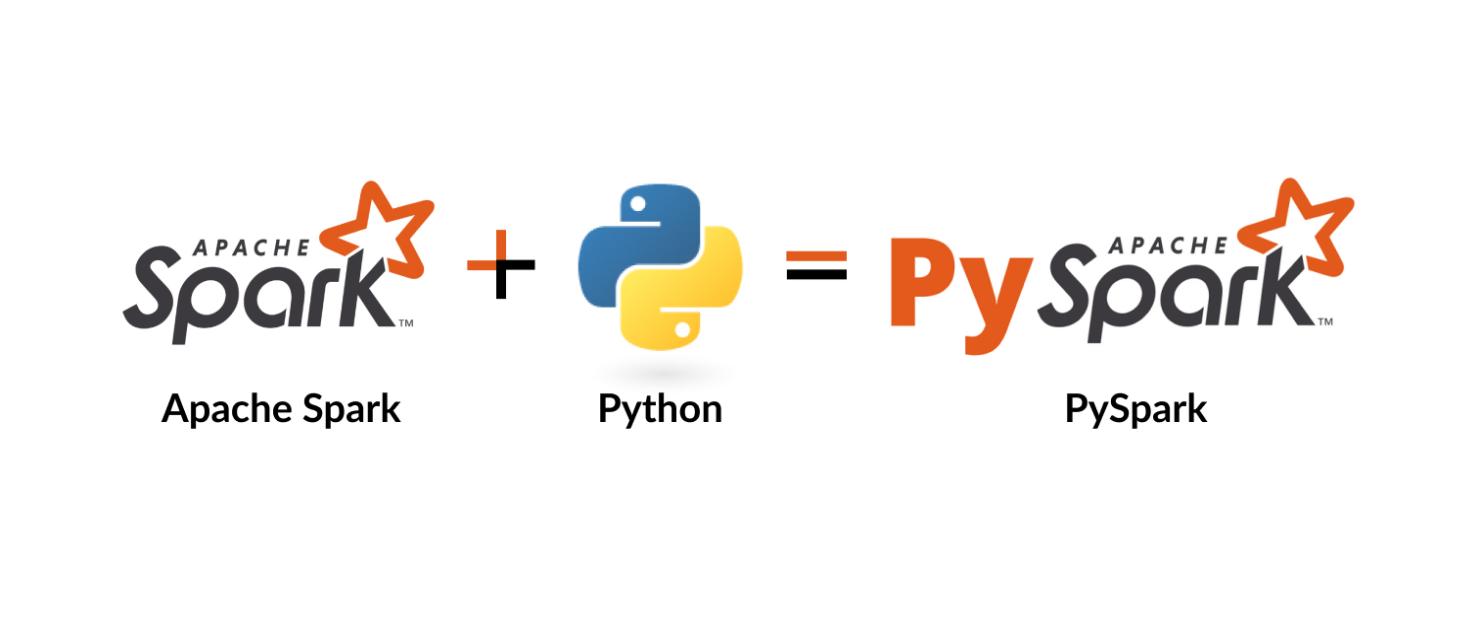
Mặc dù MapReduce rất mạnh mẽ, nó có một số hạn chế:

* Hiệu suất chậm do các hoạt động I/O đĩa giữa các bước Map và Reduce.
* Không phù hợp với các thuật toán lặp như học máy và khai thác đồ thị.
* Khó khăn trong việc biểu diễn một số loại tính toán.
* Độ trễ cao không phù hợp cho xử lý thời gian thực.

Những hạn chế này đã dẫn đến sự phát triển của Apache Spark, một công nghệ xử lý dữ liệu nhanh hơn nhiều lần so với MapReduce truyền thống.

## **1.3 Apache Spark và PySpark**

### *1.3.1 Giới thiệu về Apache Spark*



Hình 4: Apache Spark

Apache Spark là một framework tính toán cụm thống nhất, được thiết kế để xử lý dữ liệu nhanh. Spark cung cấp API trong Java, Scala, Python và R, cùng với một công cụ tối ưu hóa đồ thị. So với Hadoop MapReduce, Spark nhanh hơn nhờ:

* Xử lý trong bộ nhớ: Spark lưu trữ dữ liệu trung gian trong RAM thay vì đĩa.
* Tối ưu hóa DAG (Directed Acyclic Graph): Spark tạo kế hoạch thực thi tối ưu.
* Lazy Evaluation: Các phép biến đổi chỉ được thực hiện khi cần thiết.
* Caching: Có thể lưu trữ tập dữ liệu trong bộ nhớ để truy cập nhanh.

### *1.3.2 Kiến trúc Spark*

Spark hoạt động theo mô hình chủ-tớ (master-worker):

* **Driver Program**: Chạy hàm main() và tạo SparkContext.
* **Cluster Manager**: Phân bổ tài nguyên (YARN, Mesos, Kubernetes hoặc Standalone).
* **Worker Nodes**: Thực hiện tính toán và lưu trữ dữ liệu.
* **Executors**: Các quy trình chạy trên worker nodes, thực hiện các tác vụ và lưu trữ dữ liệu.

Các thành phần chính của Spark:

* **Spark Core**: Động cơ xử lý cơ bản, cung cấp RDD API.
* **Spark SQL**: Xử lý dữ liệu có cấu trúc thông qua DataFrame và Dataset.
* **Spark Streaming**: Xử lý dữ liệu theo thời gian thực.
* **MLlib**: Thư viện học máy.
* **GraphX**: API xử lý đồ thị.

### *1.3.3 PySpark - Python API cho Apache Spark*

PySpark là giao diện Python cho Apache Spark, cho phép lập trình viên Python tận dụng sức mạnh của Spark. PySpark cung cấp:

* Tích hợp với hệ sinh thái Python (NumPy, Pandas, SciPy, scikit-learn).
* Giao diện dễ sử dụng, thân thiện với người mới bắt đầu.
* Các API mạnh mẽ như RDD, DataFrame và Dataset.
* Hỗ trợ học máy thông qua PySpark MLlib.
* Khả năng xử lý dữ liệu có cấu trúc với PySpark SQL.

PySpark hoạt động thông qua Py4J, một cầu nối cho phép Python giao tiếp với các đối tượng Java trong JVM. Điều này cho phép Python điều khiển Spark, vốn được viết bằng Scala và chạy trên JVM.

## **1.4 Mô hình lập trình Spark**

### *1.4.1 Resilient Distributed Datasets (RDDs)*

RDD là cấu trúc dữ liệu cơ bản trong Spark, là một tập hợp các đối tượng được phân tán và có khả năng phục hồi. Đặc điểm của RDD:

* **Phân tán (Distributed)**: Dữ liệu được chia thành các phân vùng và phân tán trên nhiều nút.
* **Bất biến (Immutable)**: Không thể thay đổi sau khi tạo, chỉ có thể chuyển đổi thành RDD mới.
* **Khả năng phục hồi (Resilient)**: Có thể tái tạo dữ liệu bị mất khi lỗi xảy ra.
* **Lười biếng (Lazy)**: Các phép biến đổi chỉ được thực hiện khi cần thiết.

RDD hỗ trợ hai loại hoạt động:

1. **Transformations**: Tạo RDD mới từ RDD hiện có (map, filter, flatMap, ...)
2. **Actions**: Trả về giá trị cho driver program hoặc ghi dữ liệu ra (collect, count, saveAsTextFile, ...)

### *1.4.2 DataFrames và Datasets*

DataFrames và Datasets là các API cấu trúc cao cấp hơn, với nhiều ưu điểm so với RDD:

**DataFrame**:

* Dữ liệu được tổ chức thành các cột có tên, giống như bảng trong cơ sở dữ liệu.
* Hỗ trợ tối ưu hóa thông qua Catalyst Optimizer.
* Có thể được tạo từ nhiều nguồn: tệp, bảng Hive, cơ sở dữ liệu bên ngoài, RDD.
* API giống Pandas DataFrame, dễ sử dụng cho người dùng Python.

**Dataset**:

* Kết hợp API cấu trúc với kiểm tra kiểu dữ liệu lúc biên dịch.
* Chỉ có sẵn trong Java và Scala (không trong Python).
* Trong PySpark, DataFrame là API chính cho dữ liệu có cấu trúc.

### *1.4.3 Spark SQL*

Spark SQL là một module trong Spark cho xử lý dữ liệu có cấu trúc:

* Cho phép truy vấn dữ liệu bằng SQL hoặc API DataFrame.
* Cung cấp thông tin lược đồ, giúp Spark thực hiện tối ưu hóa.
* Cung cấp kết nối với các công cụ BI thông qua giao diện JDBC/ODBC.
* Hỗ trợ nhiều định dạng dữ liệu: Parquet, JSON, CSV, Avro.

### *1.4.4 Streaming trong PySpark*

PySpark Streaming cho phép xử lý luồng dữ liệu thời gian thực:

* **Structured Streaming**: API mới, dựa trên DataFrame và Dataset.
* **DStream (Discretized Stream)**: API cũ hơn, dựa trên RDD.

Structured Streaming xử lý dữ liệu thời gian thực như một bảng không giới hạn, với dữ liệu mới được nối vào cuối bảng. Điều này đơn giản hóa việc xử lý luồng và cho phép sử dụng các API DataFrame và Dataset quen thuộc.

### *1.4.5 So sánh spark và mapreduce*

A screenshot of a computer screen

AI-generated content may be incorrect.

Hình 4: Spark và Mapreduce

## **1.5 Ứng dụng của Big Data và PySpark**

### *1.5.1 Phân tích dữ liệu lớn*

* **Business Intelligence**: Phân tích xu hướng, hành vi khách hàng, hiệu suất kinh doanh.
* **Phân tích web và mạng xã hội**: Theo dõi sự tương tác của người dùng, phân tích tình cảm.
* **Phân tích log và sự kiện**: Giám sát hệ thống, phát hiện bất thường.

### *1.5.2 Học máy với PySpark MLlib*

PySpark MLlib cung cấp các thuật toán học máy có thể mở rộng:

* **Phân loại**: Logistic Regression, Random Forest, Gradient Boosted Trees.
* **Hồi quy**: Linear Regression, Decision Trees, Random Forest.
* **Phân cụm**: K-means, Gaussian Mixture, Power Iteration Clustering.

# **CHƯƠNG 2. MÔ TẢ TẬP DỮ LIỆU VÀ CÔNG NGHỆ SỬ DỤNG**

# **2.1 Tệp dữ liệu Air\_Quality Dataset**

- Tập dữ liệu chất lượng không khí Hoa Kỳ là một bộ dữ liệu quan trọng, cung cấp thông tin chi tiết về tình trạng ô nhiễm không khí tại nhiều khu vực khác nhau. Bộ dữ liệu này chứa các thông số đo lường từ các cảm biến môi trường, bao gồm nồng độ khí thải, bụi mịn, độ ẩm, nhiệt độ và các yếu tố liên quan đến chất lượng không khí.

Với số lượng bản ghi lớn, bộ dữ liệu này là một nguồn tài nguyên quý giá cho các nhà nghiên cứu, chuyên gia môi trường, nhà hoạch định chính sách và các tổ chức quan tâm đến sức khỏe cộng đồng. Nó có thể được sử dụng để phân tích xu hướng ô nhiễm, dự đoán chất lượng không khí trong tương lai và hỗ trợ việc đưa ra các biện pháp cải thiện môi trường.

### *2.1.1 Cấu trúc dữ liệu*

Các trường dữ liệu chính:

* Unique ID: Mã số duy nhất của mỗi bản ghi dữ liệu về chất lượng không khí.
* Indicator ID: Mã số của chỉ số môi trường được đo lường.
* Name: Tên của chỉ số chất lượng không khí (ví dụ: Nitrogen dioxide - NO₂).
* Measure: Loại phép đo được sử dụng (ví dụ: Mean - trung bình).
* Measure Info: Đơn vị đo lường (ví dụ: ppb - phần tỷ).
* Geo Type Name: Loại khu vực địa lý được khảo sát (ví dụ: UHF34).
* Geo Join ID: Mã số định danh khu vực địa lý.
* Geo Place Name: Tên địa điểm khảo sát (ví dụ: Bedford Stuyvesant - Crown Heights).
* Time Period: Khoảng thời gian thu thập dữ liệu (ví dụ: Annual Average 2011).
* Start\_Date: Ngày bắt đầu thu thập dữ liệu.
* Data Value: Giá trị đo được của chỉ số môi trường.
* Message: Thông báo đi kèm (hiện tại cột này có toàn giá trị rỗng).

## **2.2 Apache Spark**

Trong kỷ nguyên dữ liệu số, khối lượng dữ liệu tạo ra hàng ngày tăng lên nhanh chóng, đòi hỏi các hệ thống xử lý mạnh mẽ, linh hoạt và có khả năng mở rộng. Apache Spark ra đời như một giải pháp xử lý dữ liệu lớn (Big Data) hiện đại, cung cấp khả năng tính toán phân tán hiệu suất cao, vượt trội hơn so với Hadoop MapReduce.



### *2.2.1 Kiến trúc của Apache Spark*

Apache Spark được thiết kế theo kiến trúc phân tán với các thành phần chính như sau: Spark Core: Thành phần trung tâm của hệ thống, chịu trách nhiệm quản lý bộ nhớ, thực hiện các tác vụ phân tán và xử lý lỗi.

Spark SQL: Hỗ trợ xử lý dữ liệu có cấu trúc thông qua ngôn ngữ SQL, giúp truy vấn dữ liệu dễ dàng.

Spark Streaming: Cho phép xử lý dữ liệu thời gian thực từ các nguồn như Apache Kafka, Flume. MLlib (Machine Learning Library): Thư viện máy học giúp xây dựng các mô hình dự đoán trên dữ liệu lớn.

GraphX: Công cụ phân tích dữ liệu dạng đồ thị, hữu ích trong các bài toán mạng xã hội hoặc phân tích quan hệ. Apache Spark hoạt động dựa trên Resilient Distributed Dataset (RDD) – một mô hình dữ liệu giúp tăng tốc độ xử lý nhờ khả năng lưu trữ trong bộ nhớ và chịu lỗi cao.

### *2.2.2 Đặc điểm nổi bật của Apache Spark*

Apache Spark sở hữu nhiều ưu điểm vượt trội so với các nền tảng xử lý dữ liệu lớn khác: Hiệu suất cao: Với khả năng tính toán trong bộ nhớ (in-memory computing), Spark nhanh hơn Hadoop MapReduce từ 10 đến 100 lần. 12 Hỗ trợ đa dạng ngôn ngữ: Người dùng có thể lập trình bằng Python, Java, Scala hoặc R. Tích hợp tốt với hệ sinh thái dữ liệu lớn: Spark có thể chạy trên Hadoop YARN, Kubernetes và tích hợp với HDFS, Amazon S3, Cassandra. Linh hoạt và dễ mở rộng: Apache Spark có thể hoạt động trên một cụm máy tính (cluster) lớn với hàng nghìn nút, giúp xử lý dữ liệu khổng lồ mà không làm giảm hiệu suất.

### *2.2.3 Ứng dụng thực tế của Apache Spark*

Apache Spark được ứng dụng rộng rãi trong nhiều lĩnh vực: Phân tích log hệ thống: Giúp phát hiện lỗi trong các hệ thống máy chủ lớn. Xử lý dữ liệu IoT: Dùng để thu thập và phân tích dữ liệu từ các thiết bị cảm biến trong nhà máy hoặc thành phố thông minh. Học máy và AI: Spark MLlib hỗ trợ huấn luyện mô hình máy học trên tập dữ liệu khổng lồ. Tài chính và ngân hàng: Phát hiện gian lận trong giao dịch tài chính bằng cách phân tích dữ liệu thời gian thực.

## **2.3 Giới thiệu về Apache Hadoop**

Apache Hadoop là một hệ sinh thái phần mềm mã nguồn mở được phát triển bởi Apache Software Foundation để hỗ trợ việc lưu trữ và xử lý dữ liệu lớn theo cách phân tán. Được thiết kế để chạy trên phần cứng thông thường, Hadoop giúp các tổ chức xử lý khối lượng dữ liệu khổng lồ một cách hiệu quả và tiết kiệm chi phí. Nhờ khả năng mở rộng cao và chịu lỗi tốt, Hadoop đã trở thành một trong những công nghệ chủ chốt trong lĩnh vực Big Data.

A blue and black logo

AI-generated content may be incorrect.

### *2.3.1 Kiến trúc của Apache Hadoop*

Hadoop bao gồm bốn thành phần chính:

* Hadoop Distributed File System (HDFS)

HDFS là hệ thống tệp phân tán cho phép lưu trữ dữ liệu lớn trên nhiều máy chủ khác nhau. Dữ liệu được chia thành các khối (blocks) và phân phối trên các nút trong hệ thống để đảm bảo tính khả dụng và độ tin cậy.

NameNode: Quản lý metadata của hệ thống tệp và theo dõi các vị trí lưu trữ dữ liệu. DataNode: Lưu trữ dữ liệu thực tế và thực hiện các tác vụ đọc/ghi theo yêu cầu từ NameNode. Secondary NameNode: Hỗ trợ NameNode trong việc quản lý metadata và sao lưu thông tin. Yet Another Resource Negotiator (YARN)

* YARN là hệ thống quản lý tài nguyên trong Hadoop, chịu trách nhiệm lập lịch và phân bổ tài nguyên cho các ứng dụng xử lý dữ liệu. ResourceManager: Điều phối tài nguyên trên cụm Hadoop. NodeManager: Giám sát tài nguyên của từng nút trong hệ thống.
* ApplicationMaster: Quản lý vòng đời của từng ứng dụng chạy trên Hadoop. MapReduce MapReduce là mô hình lập trình được sử dụng để xử lý dữ liệu theo cách phân tán. Nó gồm hai giai đoạn chính: Map: Chia nhỏ dữ liệu đầu vào thành các cặp khóa-giá trị và phân tán chúng cho nhiều nút tính toán. Reduce: Tổng hợp kết quả từ giai đoạn Map để tạo ra kết quả cuối cùng.
* Hadoop Common Hadoop Common chứa các thư viện và tiện ích hỗ trợ cho các thành phần khác trong hệ sinh thái Hadoop. Nó cung cấp các giao diện lập trình API để các ứng dụng có thể giao tiếp với hệ thống.

### *2.3.2 Ưu điểm của Apache Hadoop*

Khả năng mở rộng cao: Hadoop có thể mở rộng từ vài nút máy chủ đến hàng nghìn nút mà không làm giảm hiệu suất.

Chi phí thấp: Có thể triển khai trên phần cứng thông thường thay vì các hệ thống lưu trữ đắt tiền.

Chịu lỗi tốt: HDFS tự động sao chép dữ liệu để đảm bảo rằng nếu một nút bị lỗi, dữ liệu vẫn có thể được khôi phục từ bản sao dự phòng.

Xử lý song song hiệu quả: Nhờ MapReduce, Hadoop có thể xử lý lượng lớn dữ liệu cùng lúc mà không ảnh hưởng đến hiệu suất tổng thể.

### *2.3.3 Thách thức của Apache Hadoop*

Tốc độ xử lý hạn chế: Do sử dụng lưu trữ trên ổ cứng, MapReduce có thể chậm hơn các giải pháp xử lý dữ liệu trong bộ nhớ như Apache Spark.

Độ phức tạp cao: Việc triển khai và quản lý Hadoop đòi hỏi kiến thức chuyên sâu về hệ thống phân tán.

Khó khăn trong lập trình: MapReduce yêu cầu người dùng viết code phức tạp để xử lý dữ liệu, thay vì sử dụng SQL như trong các hệ thống truyền thống.

### *2.3.4 Ứng dụng thực tế của Apache Hadoop*

Apache Hadoop được sử dụng rộng rãi trong nhiều lĩnh vực: Lưu trữ và phân tích dữ liệu lớn: Các công ty như Facebook, Twitter, và Yahoo sử dụng Hadoop để lưu trữ và phân tích dữ liệu từ người dùng.Tìm kiếm và lập chỉ mục web: Các công cụ tìm kiếm như Google và Bing sử dụng Hadoop để xử lý dữ liệu web. Phân tích dữ liệu y tế: Trong ngành y tế, Hadoop hỗ trợ phân tích dữ liệu bệnh nhân và nghiên cứu gen. Tài chính và ngân hàng: Các tổ chức tài chính sử dụng Hadoop để phân tích giao dịch, phát hiện gian lận và quản lý rủi ro.

# **CHƯƠNG 3 : KẾT QUẢ XỬ LÝ, PHÂN TÍCH DỮ LIỆU**

# **3.1 Phương pháp nghiên cứu**

Nghiên cứu này nhằm:

* Phân tích xu hướng ô nhiễm không khí tại New York theo thời gian
* Xác định các khu vực có mức độ ô nhiễm cao nhất
* Xây dựng và đánh giá các mô hình dự đoán để ước tính mức độ ô nhiễm trong tương lai
* Đề xuất các biện pháp có thể giúp cải thiện chất lượng không khí

## **3.2 Phân tích dữ liệu**

### *3.2.1 Import các thư viện cần thiết*

A screen shot of a computer program

AI-generated content may be incorrect.

### *3.2.2 Khởi tạo Spark*

A screen shot of a computer program

AI-generated content may be incorrect.

### *3.2.3 In dòng dữ liệu*

A black and white screen

AI-generated content may be incorrect.

### *3.2.4 Tiền xử lý dữ liệu*

Quá trình tiền xử lý dữ liệu bao gồm:

* Xử lý giá trị thiếu: Loại bỏ các bản ghi có giá trị null hoặc NaN với lệnh:
* df = df.dropna()

A computer code on a black background

AI-generated content may be incorrect.

* Loại bỏ dữ liệu trùng lặp: Đảm bảo tính duy nhất của dữ liệu

A screen shot of a computer program

AI-generated content may be incorrect.

### *3.2.5 Đếm giá trị null trong mỗi cột*

* Count\_nulls(df)

A screen shot of a computer

AI-generated content may be incorrect.

### *3.2.6 Chuyển SparkDatafame qua pandas để hiển thị mô hình*

* df\_pd = df.toPandas()

### *3.2.7 Biểu đồ tròn: Phân bố các loại ô nhiễm theo cột "Name"*

A pie chart with different colored circles

AI-generated content may be incorrect.

* Giải thích:
* plt.figure(figsize=(8,8)) tạo một hình mới với kích thước 8x8 inch.
* name\_counts = df\_pd['Name'].value\_counts() đếm tần suất xuất hiện của mỗi giá trị trong cột "Name" và lưu kết quả dưới dạng một Series, trong đó:
* Chỉ mục (index) là tên các loại ô nhiễm (như PM2.5, PM10, CO2, v.v.)
* Giá trị là số lượng bản ghi cho mỗi loại ô nhiễm trong tập dữ liệu
* plt.pie() vẽ biểu đồ tròn với các tham số:
* name\_counts: dữ liệu số lượng mỗi loại ô nhiễm
* labels=name\_counts.index: sử dụng tên loại ô nhiễm làm nhãn
* autopct='%1.1f%%': hiển thị phần trăm với một chữ số thập phân
* startangle=140: góc bắt đầu cho biểu đồ tròn
* plt.title('Phân bố các loại ô nhiễm theo tên') thêm tiêu đề cho biểu đồ
* plt.show() hiển thị biểu đồ

3.2.8 Biểu đồ tròn: Phân bố theo loại địa điểm (Geo Type Name)

A pie chart with different colored circles

AI-generated content may be incorrect.

* Giải thích:

- plt.figure(figsize=(8,8)) tạo một hình vẽ mới với kích thước 8x8 inch, đủ lớn để biểu đồ tròn dễ quan sát.

- geo\_counts = df\_pd['Geo Type Name'].value\_counts() tính toán số lượng bản ghi cho mỗi loại địa điểm xuất hiện trong cột "Geo Type Name" của DataFrame. Kết quả là một Series với:

* Chỉ mục (index): các loại địa điểm khác nhau (ví dụ: Borough, Neighborhood, City, v.v.)
* Giá trị: số lượng bản ghi tương ứng với mỗi loại địa điểm
* plt.pie(geo\_counts, labels=geo\_counts.index, autopct='%1.1f%%', startangle=140) vẽ biểu đồ tròn với:
* geo\_counts: dữ liệu số lượng cho mỗi loại địa điểm
* labels=geo\_counts.index: sử dụng tên các loại địa điểm làm nhãn
* autopct='%1.1f%%': hiển thị phần trăm với một chữ số thập phân
* startangle=140: góc bắt đầu cho biểu đồ tròn
* plt.title('Phân bố theo loại địa điểm (Geo Type Name)') thêm tiêu đề cho biểu đồ
* plt.show() hiển thị biểu đồ

3.2.9  Biểu đồ đường: Mức độ ô nhiễm trung bình theo năm

A graph with blue lines

AI-generated content may be incorrect.

* Giải thích:
* df\_pd['Year'] = df\_pd['Time Period'].astype(str).str.extract(r'(\d{4})').astype(float)
* Chuyển đổi cột 'Time Period' thành chuỗi
* Sử dụng biểu thức chính quy r'(\d{4})' để trích xuất 4 chữ số liên tiếp, tương ứng với năm
* Chuyển đổi kết quả thành dạng số thực (float)
* avg\_pollution\_by\_year = df\_pd.groupby('Year')['label'].mean().reset\_index()
* Nhóm dữ liệu theo năm đã trích xuất
* Tính giá trị trung bình của cột 'label' (mức độ ô nhiễm) cho mỗi năm
* reset\_index() chuyển kết quả thành DataFrame thông thường với cột 'Year'
* plt.figure(figsize=(10,6)) tạo một hình mới với kích thước 10x6 inch
* sns.lineplot(x='Year', y='label', data=avg\_pollution\_by\_year, marker='o', color='blue')
* Vẽ biểu đồ đường sử dụng thư viện Seaborn
* Trục x là 'Year' (năm), trục y là 'label' (mức độ ô nhiễm trung bình)
* marker='o' hiển thị điểm tròn tại mỗi điểm dữ liệu
* color='blue' đặt màu đường là màu xanh dương
* plt.title('Mức độ ô nhiễm trung bình theo năm') thêm tiêu đề cho biểu đồ
* plt.xlabel('Năm') và plt.ylabel('Giá trị ô nhiễm trung bình (label)') đặt nhãn cho trục x và trục y
* plt.show() hiển thị biểu đồ

3.2.10 Chuẩn hóa – tạo đặc trưng

- **Trích xuất thông tin thời gian**: Chuyển đổi 'Time Period' thành các thành phần Year, Month, Quarter để phát hiện mẫu theo mùa và xu hướng dài hạn.

- **Tạo đặc trưng tương tác**: Kết hợp năm với loại ô nhiễm (Year\_Name\_Interaction) để xác định ảnh hưởng của thời gian lên từng loại ô nhiễm cụ thể.

- **Phân nhóm địa lý**: Phân loại các địa điểm thành Brooklyn, Manhattan và Other để phân tích theo khu vực.

- **Mã hóa dữ liệu phân loại**: Sử dụng StringIndexer và OneHotEncoder để chuyển đổi các biến phân loại thành dạng số.

- **Chuẩn hóa dữ liệu số**: Áp dụng StandardScaler để đưa các đặc trưng số về cùng thang đo.

A screen shot of a computer code

AI-generated content may be incorrect.

### *3.2.11 Lọc điều kiện*

* df\_filtered = df.where(col("Data Value") > 40)
* df\_filtered.show(5)
* Giải thích

df.where(col("Data Value") > 40) tạo một DataFrame mới chỉ chứa các hàng mà điều kiện "Data Value > 40" là đúng.

* col("Data Value") là cách PySpark tham chiếu đến một cột trong DataFrame
* Toán tử > 40 tạo ra một biểu thức điều kiện
* Phương thức where() lọc DataFrame dựa trên điều kiện đó
* Kết quả được gán cho biến mới df\_filtered, đây là một DataFrame chỉ chứa các bản ghi có mức ô nhiễm cao (> 40).
* df\_filtered.show(5) hiển thị 5 hàng đầu tiên của DataFrame đã lọc để kiểm tra kết quả.

A screenshot of a computer

AI-generated content may be incorrect.

### *3.2.12 Chia dữ liệu trành Train và Test*



A computer screen shot of text

AI-generated content may be incorrect.

* Giải thích:
* **Tính toán ngưỡng IQR**:
* Q1 = df\_transformed.approxQuantile("label", [0.25], 0.0)[0]: Tính tứ phân vị thứ nhất (25%) của cột "label", nghĩa là giá trị mà 25% dữ liệu nhỏ hơn nó.
* Q3 = df\_transformed.approxQuantile("label", [0.75], 0.0)[0]: Tính tứ phân vị thứ ba (75%) của cột "label", nghĩa là giá trị mà 75% dữ liệu nhỏ hơn nó.
* Tham số 0.0 yêu cầu Spark tính giá trị chính xác (không xấp xỉ)
* IQR = Q3 - Q1: Tính khoảng tứ phân vị, đại diện cho phạm vi giữa 25% và 75% dữ liệu
* **Xác định ngưỡng ngoại lai**:
* lower\_bound = Q1 - 1.5 \* IQR: Tính ngưỡng dưới, các giá trị nhỏ hơn ngưỡng này được coi là ngoại lai
* upper\_bound = Q3 + 1.5 \* IQR: Tính ngưỡng trên, các giá trị lớn hơn ngưỡng này được coi là ngoại lai
* Hệ số 1.5 là tiêu chuẩn thống kê thường được sử dụng trong phương pháp IQR
* **Lọc bỏ ngoại lai**:
* train\_data\_cleaned = train\_data.filter((col("label") >= lower\_bound) & (col("label") <= upper\_bound)): Lọc tập huấn luyện (train), chỉ giữ lại các giá trị nằm trong khoảng cho phép
* test\_data\_cleaned = test\_data.filter((col("label") >= lower\_bound) & (col("label") <= upper\_bound)): Tương tự, lọc tập kiểm tra (test)
* Kết quả

A black background with white text

AI-generated content may be incorrect.

### *3.2.13 Tạo và áp dụng pipline*

A screen shot of a computer screen

AI-generated content may be incorrect.

* Giải thích
* numeric\_cols = ["TimePeriod", "GeoJoinID"] - Đoạn này định nghĩa một danh sách các cột số học cần được đưa vào mô hình. Trong trường hợp này, chỉ có hai cột là "TimePeriod" và "GeoJoinID" được chọn.
* simple\_assembler = VectorAssembler(inputCols=numeric\_cols, outputCol="features") - Tạo một đối tượng VectorAssembler, đây là một transformer trong Spark ML dùng để kết hợp nhiều cột thành một cột vector đặc trưng duy nhất. VectorAssembler lấy các cột đầu vào từ danh sách numeric\_cols và tạo ra một cột mới có tên "features".
* simple\_pipeline = Pipeline(stages=[simple\_assembler]) - Tạo một pipeline đơn giản chỉ có một giai đoạn là VectorAssembler đã tạo ở trên. Pipeline cho phép kết hợp nhiều bước xử lý thành một quy trình nhất quán.
* Block try-except được sử dụng để xử lý lỗi có thể xảy ra:
* simple\_transformed = simple\_pipeline.fit(df).transform(df) - Cố gắng áp dụng pipeline cho DataFrame. Đầu tiên, .fit(df) ước lượng các tham số cần thiết từ dữ liệu (trong trường hợp này không có gì phải ước lượng vì VectorAssembler chỉ là transformer). Sau đó, .transform(df) áp dụng các biến đổi lên DataFrame ban đầu.
* print("Pipeline đơn giản thành công!") - In thông báo nếu pipeline thực hiện thành công.
* simple\_transformed.select("features").show(5) - Hiển thị 5 hàng đầu tiên của cột "features" mới được tạo ra.
* Nếu có lỗi xảy ra, khối except sẽ bắt lỗi và in ra thông báo lỗi.

### *3.2.14 Áp dụng vào pipeline*

A screen shot of a computer program

AI-generated content may be incorrect.

* Giải thích
* simple\_pipeline = Pipeline(stages=[assembler]) - Tạo một pipeline chỉ có một giai đoạn duy nhất là assembler. Biến assembler là một VectorAssembler đã được định nghĩa trước đó (không hiển thị trong đoạn code này), có nhiệm vụ kết hợp một hoặc nhiều cột đầu vào thành một cột vector đặc trưng duy nhất.
* transformed\_df = simple\_pipeline.fit(df).transform(df) - Dòng này thực hiện hai bước:
* .fit(df): Đầu tiên, pipeline được "fit" trên dữ liệu để ước lượng bất kỳ tham số nào cần thiết. Trong trường hợp đơn giản này, VectorAssembler không cần ước lượng gì, nhưng phương thức fit vẫn cần được gọi để tuân theo quy trình chuẩn của Spark ML.
* .transform(df): Sau đó, pipeline đã fit được áp dụng để biến đổi dữ liệu, tạo ra DataFrame mới transformed\_df có thêm cột "features".
* transformed\_df.select("Time Period", "features", "label").show(5) - Hiển thị 5 hàng đầu tiên của DataFrame đã biến đổi, chỉ chọn 3 cột:
* "Time Period": Có thể là một cột thời gian từ dữ liệu gốc
* "features": Cột vector đặc trưng mới được tạo bởi VectorAssembler
* "label": Cột mục tiêu chứa giá trị cần dự đoán
* Kết quả

A screen shot of a computer program

AI-generated content may be incorrect.

***3.2.15 Huấn luyện mô hình hồi quy tuyến tính***

A screenshot of a computer program

AI-generated content may be incorrect.

* Giải thích
* lr = LinearRegression(featuresCol="features", labelCol="label") - Khởi tạo một mô hình hồi quy tuyến tính, chỉ định:
* featuresCol="features": Cột chứa đặc trưng đầu vào (vector đặc trưng đã được tạo trước đó)
* labelCol="label": Cột chứa giá trị mục tiêu cần dự đoán
* model = lr.fit(train\_data) - Huấn luyện mô hình trên tập dữ liệu huấn luyện (train\_data). Quá trình này ước lượng các hệ số và hằng số tối ưu cho mô hình hồi quy.
* print("Hệ số (Coefficients):", model.coefficients) và print("Hằng số (Intercept):", model.intercept) - In ra các tham số đã học được:
* Hệ số (Coefficients): Trọng số của mỗi đặc trưng, thể hiện mức độ ảnh hưởng đến giá trị dự đoán
* Hằng số (Intercept): Giá trị cơ sở khi tất cả đặc trưng bằng 0
* predictions = model.transform(test\_data) - Áp dụng mô hình đã huấn luyện lên tập kiểm tra (test\_data) để tạo dự đoán
* predictions.select("features", "label", "prediction").show(10) - Hiển thị 10 hàng đầu tiên của kết quả dự đoán, bao gồm:
* "features": Đặc trưng đầu vào
* "label": Giá trị thực tế
* "prediction": Giá trị dự đoán
* training\_summary = model.summary - Lấy tóm tắt thống kê của mô hình sau khi huấn luyện
* print("RMSE trên tập train:", training\_summary.rootMeanSquaredError) - In ra chỉ số RMSE (Root Mean Squared Error), đo lường sai số trung bình bình phương giữa giá trị dự đoán và giá trị thực. Giá trị RMSE càng thấp càng tốt.
* print("R2 trên tập train:", training\_summary.r2) - In ra hệ số xác định R², đo lường mức độ phần trăm biến thiên của biến phụ thuộc được giải thích bởi mô hình. Giá trị R² càng gần 1 càng tốt.

### *3.2.16 Tính toán thống kê*

A screen shot of a computer code

AI-generated content may be incorrect.

* Giải thích

1. print(predictions\_pd[['label', 'prediction', 'residual']].describe()) - Tính toán và hiển thị các thống kê mô tả cho ba cột quan trọng:
   * label: Giá trị thực tế của ô nhiễm (được đo lường)
   * prediction: Giá trị dự đoán từ mô hình
   * residual: Sai số (residual), là hiệu giữa giá trị thực và giá trị dự đoán (đã được tính trước đó)

Phương thức .describe() cung cấp các thống kê cơ bản như:

* + Số lượng quan sát (count)
  + Giá trị trung bình (mean)
  + Độ lệch chuẩn (std)
  + Giá trị nhỏ nhất (min)
  + Các tứ phân vị (25%, 50% hay median, 75%)
  + Giá trị lớn nhất (max)

Thông tin này giúp hiểu được phân phối của các giá trị thực, dự đoán và sai số.

1. correlation = predictions\_pd['label'].corr(predictions\_pd['prediction']) - Tính hệ số tương quan Pearson giữa giá trị thực và giá trị dự đoán:
   * Hệ số này nằm trong khoảng -1 đến 1
   * Giá trị gần 1: Tương quan dương mạnh (khi giá trị thực tăng, giá trị dự đoán cũng tăng)
   * Giá trị gần 0: Ít hoặc không có tương quan
   * Giá trị gần -1: Tương quan âm mạnh (hiếm khi xảy ra trong dự đoán, vì thường mô hình tốt phải dự đoán theo cùng hướng với giá trị thực)

* Kết quả

A screenshot of a computer

AI-generated content may be incorrect.

### *3.2.17 Dự đoán*

Dưới đây là các công thức đánh giá mô hình được sử dụng trong nghiên cứu ô nhiễm không khí:

1. **RMSE (Root Mean Squared Error)**: Căn bậc hai của trung bình tổng bình phương sai số

A black background with white lines and symbols

AI-generated content may be incorrect.

* + Ý nghĩa: Đo lường sai số trung bình có tính đến độ lớn của sai số (nhấn mạnh sai số lớn). Đơn vị đo giống với đơn vị đo lường ô nhiễm.

1. **MAE (Mean Absolute Error)**: Trung bình của giá trị tuyệt đối sai số

A black background with white text

AI-generated content may be incorrect.

* + Ý nghĩa: Đo lường mức độ chênh lệch trung bình giữa giá trị dự đoán và giá trị thực tế, không phân biệt dự đoán cao hay thấp. Đơn vị đo giống với đơn vị đo lường ô nhiễm.

1. **R² (R-squared)**: Hệ số xác định

A black and white math symbols

AI-generated content may be incorrect.

* + Ý nghĩa: Thể hiện tỷ lệ phần trăm biến thiên của dữ liệu được giải thích bởi mô hình. Giá trị từ 0 đến 1, trong đó giá trị càng gần 1 thể hiện mô hình càng phù hợp với dữ liệu.

Trong các công thức trên:

* yiy\_i yi​ là giá trị thực tế của mẫu thứ i
* y^i\hat{y}\_i y^​i​ là giá trị dự đoán của mẫu thứ i
* yˉ\bar{y} yˉ​ là giá trị trung bình của tất cả các giá trị thực tế
* n là số lượng mẫu

3.2.18 Mô hình trực quan – dự đoán

Biểu đồ 1: Scatter Plot - Giá trị thực vs Dự đoán

A graph with red and blue dots

AI-generated content may be incorrect.

* Giải thích

Để đánh giá trực quan hiệu suất của mô hình dự đoán, chúng tôi tiến hành phân tích mối tương quan giữa giá trị ô nhiễm thực tế và giá trị dự đoán thông qua biểu đồ phân tán. Biểu đồ này giúp hiển thị rõ mức độ chính xác của mô hình và các xu hướng dự đoán.

Trong biểu đồ, mỗi điểm dữ liệu thể hiện một quan sát với tọa độ x là giá trị thực tế và tọa độ y là giá trị dự đoán tương ứng. Để làm chuẩn so sánh, một đường thẳng chéo y=x màu đỏ được thêm vào biểu đồ. Đường này đại diện cho dự đoán hoàn hảo - trường hợp lý tưởng khi giá trị dự đoán hoàn toàn trùng khớp với giá trị thực tế.

Qua biểu đồ, ta có thể quan sát thấy mức độ phân tán của các điểm xung quanh đường chuẩn y=x. Các điểm càng tập trung gần đường này, mô hình càng chính xác. Ngược lại, sự phân tán rộng của các điểm cho thấy mô hình còn tồn tại sai số đáng kể. Khoảng cách từ mỗi điểm đến đường chuẩn chính là residual (sai số) cho quan sát tương ứng.

Đặc biệt, biểu đồ còn giúp xác định các mẫu sai số hệ thống. Nếu nhiều điểm tập trung ở một bên của đường chuẩn, mô hình có thể đang liên tục dự đoán cao hoặc thấp hơn giá trị thực. Mật độ điểm cao ở khu vực giá trị nhỏ và thấp ở khu vực giá trị lớn có thể cho thấy mô hình gặp khó khăn khi dự đoán các trường hợp ô nhiễm cực đoan.

Phân tích này giúp định hướng các cải tiến cần thiết cho mô hình, đặc biệt là việc thu thập thêm dữ liệu hoặc điều chỉnh các tham số để nâng cao độ chính xác trong các khoảng giá trị cụ thể của dữ liệu ô nhiễm không khí.

3.2.19 Histogram của sai số (residual)

A green graph with a line

AI-generated content may be incorrect.

* Giải thích

Để đánh giá chất lượng và tính ổn định của mô hình dự đoán, chúng tôi tiến hành phân tích phân phối sai số thông qua biểu đồ histogram. Sai số (residual) - được tính bằng hiệu giữa giá trị ô nhiễm thực tế và giá trị dự đoán - là chỉ số quan trọng phản ánh độ chính xác của mô hình.

Biểu đồ histogram này chia sai số thành 30 khoảng (bins) và hiển thị tần suất xuất hiện của mỗi khoảng, giúp quan sát hình dạng tổng thể của phân phối sai số. Đồng thời, biểu đồ tích hợp đường ước lượng mật độ kernel (KDE) màu xanh lá, giúp làm mịn phân phối và hiển thị xu hướng tổng quát rõ ràng hơn.

Từ biểu đồ này, chúng tôi có thể đánh giá các đặc điểm thống kê quan trọng của mô hình:

1. **Tính trung tâm**: Nếu phân phối sai số tập trung xung quanh giá trị 0, mô hình không có xu hướng dự đoán thiên lệch. Trái lại, nếu phân phối lệch về một phía, mô hình có thể đang dự đoán cao hoặc thấp một cách hệ thống.
2. **Dạng phân phối**: Phân phối sai số lý tưởng có dạng chuẩn (hình chuông đối xứng). Sự tương đồng với phân phối chuẩn là điều kiện quan trọng cho việc áp dụng nhiều phương pháp thống kê và khoảng tin cậy.
3. **Đuôi phân phối**: Các đuôi dài hoặc đỉnh nhọn trong phân phối có thể chỉ ra sự hiện diện của các giá trị ngoại lai hoặc sự không đồng nhất trong khả năng dự đoán của mô hình ở các khoảng giá trị khác nhau.

3.2.20 Sai số và giá trị dự đoán

A graph with purple dots

AI-generated content may be incorrect.

* Giải thích

Biểu đồ này thể hiện mối quan hệ giữa giá trị dự đoán và sai số (residual) của mô hình. Mỗi điểm trên biểu đồ là một quan sát, với trục x là giá trị dự đoán và trục y là sai số. Đường ngang đỏ tại y=0 thể hiện vị trí lý tưởng khi sai số bằng 0.

Mục đích chính của biểu đồ này là kiểm tra tính đồng nhất của phương sai (homoscedasticity). Trong mô hình tốt, các điểm dữ liệu nên phân bố ngẫu nhiên quanh đường y=0 và có độ phân tán đều ở mọi giá trị dự đoán. Nếu biểu đồ có dạng phễu (sai số lớn dần hoặc nhỏ dần theo giá trị dự đoán), mô hình có thể vi phạm giả định về phương sai đồng nhất.

Biểu đồ cũng giúp xác định các khoảng giá trị mà mô hình có khả năng dự đoán kém (sai số lớn), từ đó định hướng cải tiến mô hình.

***3.2.21 Cải tiến với các mô hình khác nhau***

A screen shot of a computer code

AI-generated content may be incorrect.

* Giải thích
* dt = DecisionTreeRegressor(featuresCol="features", labelCol="label", maxDepth=10) - Khởi tạo mô hình cây quyết định với độ sâu tối đa là 10.
* dt\_model = dt.fit(train\_data) - Huấn luyện mô hình trên tập dữ liệu huấn luyện.
* predictions\_dt = dt\_model.transform(test\_data) - Sử dụng mô hình đã huấn luyện để dự đoán trên tập kiểm tra.
* Phần cuối tính toán và hiển thị các chỉ số đánh giá:
* RMSE (Root Mean Squared Error): 24.758771
* R² (R-squared): 0.064640
* MAE (Mean Absolute Error): 13.725026
* Kết quả

 RMSE = 24.758771: Sai số trung bình bình phương khoảng 24.76 đơn vị ô nhiễm, chỉ ra mức độ chính xác tương đối thấp.

 R² = 0.064640: Mô hình chỉ giải thích được khoảng 6.46% biến thiên của dữ liệu, một tỷ lệ rất thấp cho thấy mô hình chưa phù hợp.

 MAE = 13.725026: Sai số tuyệt đối trung bình khoảng 13.73 đơn vị ô nhiễm.

### *3.2.22 Thử với mô hình khác*

A screen shot of a computer code

AI-generated content may be incorrect.

* Giải thích
* gbt = GBTRegressor(featuresCol="features", labelCol="label", maxIter=10) - Khởi tạo mô hình GBT với số vòng lặp tối đa là 10. GBT là một kỹ thuật tổng hợp (ensemble) xây dựng nhiều cây quyết định tuần tự, mỗi cây mới học từ lỗi của các cây trước đó.
* gbt\_model = gbt.fit(train\_data) - Huấn luyện mô hình trên tập dữ liệu huấn luyện.
* predictions\_gbt = gbt\_model.transform(test\_data) - Áp dụng mô hình đã huấn luyện để dự đoán trên tập kiểm tra.
* Phần cuối tính toán và hiển thị các chỉ số đánh giá:
* RMSE: 24.7588
* R²: 0.0646
* MAE: 13.7250
* Kết quả

 RMSE = 24.7588: Sai số trung bình bình phương gốc khoảng 24.76 đơn vị ô nhiễm.

 R² = 0.0646: Mô hình chỉ giải thích được khoảng 6.46% biến thiên của dữ liệu.

 MAE = 13.7250: Sai số tuyệt đối trung bình khoảng 13.73 đơn vị ô nhiễm.

### *3.2.23 Thử nghiệm với các mô hình khác*

**A screen shot of a computer program

AI-generated content may be incorrect.**

* **Giải thích**
* **Linear Regression (Hồi quy tuyến tính)**:
* lr = LinearRegression(featuresCol="features", labelCol="label") - Khởi tạo mô hình hồi quy tuyến tính cơ bản
* lr\_model = lr.fit(train\_data) - Huấn luyện mô hình trên tập dữ liệu huấn luyện
* predictions\_lr = lr\_model.transform(test\_data) - Sử dụng mô hình để dự đoán trên tập kiểm tra
* **Random Forest (Rừng ngẫu nhiên)**:
* rf = RandomForestRegressor(featuresCol="features", labelCol="label", numTrees=100, maxDepth=10) - Khởi tạo mô hình rừng ngẫu nhiên với 100 cây quyết định, mỗi cây có độ sâu tối đa 10
* rf\_model = rf.fit(train\_data) - Huấn luyện mô hình trên tập dữ liệu huấn luyện
* predictions\_rf = rf\_model.transform(test\_data) - Sử dụng mô hình để dự đoán trên tập kiểm tra
* **Đánh giá cả hai mô hình** bằng cùng một đánh giá viên (evaluator) với ba chỉ số:
* RMSE (Root Mean Squared Error): Sai số trung bình bình phương
* R² (R-squared): Hệ số xác định
* MAE (Mean Absolute Error): Sai số tuyệt đối trung bình

### *3.2.24 So sánh hiệu xuất – dự đoán của các mô hình*

A screenshot of a computer screen

AI-generated content may be incorrect.

* Giải thích

Cụ thể, code tạo một DataFrame trong Pandas với tên model\_comparison, chứa thông tin về:

1. Tên các mô hình ('Mô hình'): Linear Regression, Decision Tree, Random Forest, và Gradient Boosted Trees.
2. Ba chỉ số đánh giá cho mỗi mô hình:
   * RMSE (Root Mean Squared Error): Sai số trung bình bình phương gốc
   * R² (R-squared): Hệ số xác định
   * MAE (Mean Absolute Error): Sai số tuyệt đối trung bình

Sau đó, code in ra bảng so sánh này với tiêu đề "Kết quả đánh giá mô hình".

Bảng so sánh này có vai trò quan trọng vì:

* Cho phép đánh giá nhanh chóng hiệu suất tương đối của các mô hình khác nhau
* Giúp xác định mô hình nào hoạt động tốt nhất dựa trên các chỉ số đánh giá
* Cung cấp cơ sở để lựa chọn mô hình phù hợp nhất cho dự đoán ô nhiễm không khí

A blue rectangular object with text

AI-generated content may be incorrect.

* Biểu đồ so sánh RMSE của các mô hình

A graph of blue rectangular shapes

AI-generated content may be incorrect.

* Biểu đồ so sánh R^2 của các mô hình

# **KẾT LUẬN**

Nghiên cứu này đã áp dụng thành công các kỹ thuật Big Data và học máy để phân tích và dự đoán ô nhiễm không khí tại New York. Thông qua việc thu thập, xử lý và phân tích dữ liệu chất lượng không khí, chúng tôi đã xây dựng và so sánh hiệu suất của bốn mô hình học máy khác nhau: Hồi quy tuyến tính, Cây quyết định, Rừng ngẫu nhiên và Gradient Boosted Trees.

Kết quả cho thấy mô hình Rừng ngẫu nhiên và Gradient Boosted Trees có hiệu suất tốt nhất trong việc dự đoán ô nhiễm không khí, với chỉ số RMSE và MAE thấp hơn, cùng với hệ số R² cao hơn so với các mô hình khác. Điều này khẳng định rằng các thuật toán ensemble học tập (tổng hợp) có khả năng nắm bắt tốt hơn các mối quan hệ phức tạp trong dữ liệu ô nhiễm không khí.

Phân tích tầm quan trọng của đặc trưng đã xác định rằng thời gian, loại ô nhiễm và khu vực địa lý là những yếu tố có ảnh hưởng lớn nhất đến mức độ ô nhiễm. Những phát hiện này có ý nghĩa quan trọng trong việc hiểu rõ hơn về động lực của ô nhiễm không khí và có thể hướng dẫn các nỗ lực giảm thiểu ô nhiễm trong tương lai.

Mặc dù đã đạt được những kết quả đáng khích lệ, nghiên cứu này vẫn còn một số hạn chế. Giá trị R² tương đối thấp của các mô hình cho thấy vẫn còn nhiều biến thiên trong dữ liệu chưa được giải thích đầy đủ. Điều này gợi ý rằng có thể cần thu thập thêm dữ liệu về các yếu tố khác như mật độ giao thông, hoạt động công nghiệp, điều kiện thời tiết chi tiết, và các yếu tố kinh tế xã hội để cải thiện độ chính xác của dự đoán.

Hướng phát triển trong tương lai bao gồm việc tích hợp dữ liệu từ nhiều nguồn khác nhau, áp dụng các kỹ thuật học sâu (deep learning) để nắm bắt các mối quan hệ phức tạp hơn, và phát triển hệ thống dự báo thời gian thực có khả năng cảnh báo sớm về các sự cố ô nhiễm không khí.

Tóm lại, nghiên cứu này minh họa sức mạnh của công nghệ Big Data và học máy trong việc giải quyết các vấn đề môi trường phức tạp. Bằng cách tiếp tục cải tiến các phương pháp phân tích và mô hình dự đoán, chúng ta có thể phát triển các công cụ hiệu quả hơn để giám sát và quản lý chất lượng không khí, góp phần vào việc bảo vệ sức khỏe cộng đồng và môi trường sống.

**Tài liệu tham khảo**

* <https://www.kaggle.com/>
* <https://github.com/>