TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
VIỆN TRÍ TUỆ NHÂN TẠO

_____***_____

BÁO CÁO MÔN HỌC KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

Đề tài:

Time-Series Forecasting with Spark and Kafka for Weather Data

Giảng viên: TS. Trần Hồng Việt

Ths. Đỗ Thu Uyên

Nhóm thực hiện:

1. **Triệu Vũ Hoàn - 22022654**
2. **Hoàng Đình Hưng - 22022662**
3. **Lê Thành Đạt - 22022627**
4. **Hoàng Bùi Tuấn Anh - 22022611**
5. **Lê Hoàng Anh - 22022563**

Mục lục:

I. Giới thiệu

1. Tổng quan và lý do thực hiện:

Sự biến đổi khí hậu và các hiện thời tiết cực đoan đang đặt ra thách thức lớn trong nhiều lĩnh vực như nông nghiệp, vận tải và quản lý khủng hoảng. Dự báo thời tiết chính xác và kịp thời có vai trò then chốt trong việc giảm thiểu rủi ro và tối ưu hóa nguồn lực. Tuy nhiên, việc dự báo thời tiết đòi hỏi khả năng xử lý và phân tích lượng dữ liệu lớn theo thời gian thực.

Dự án này nhằm xây dựng hệ thống thu thập, phân tích và dự báo dữ liệu chuỗi thời gian thực (time-series forecasting) kết hợp với công nghệ xử lý thời gian thực giúp cải thiện độ chính xác và khả năng ứng phó nhanh chóng với các tình huống bất ngờ. Với các công cụ Apache Spark và Kafka có khả năng xử lý dữ liệu lớn, phân tán và thời gian thực. Spark, với sự hỗ trợ từ **Spark Streaming**, cung cấp khả năng xử lý dữ liệu thời gian thực, tích hợp liền mạch với Kafka để đảm bảo luồng dữ liệu liên tục, đồng thời triển khai các mô hình học máy mạnh mẽ trên dữ liệu này.

2. Mục tiêu dự án

Xây dựng hệ thống thu thập, phân tích và dự báo dữ liệu thời tiết thời gian thực, tích hợp các công cụ mạnh mẽ như Spark, Kafka, Power BI, Delta Lake, Docker. Tăng cường độ chính xác của dự báo bằng cách áp dụng các mô hình học máy tiên tiến. Đảm bảo khả năng mở rộng và hiệu suất cao khi xử lý dữ liệu thời tiết lớn.

Mong muốn giảm thiểu thiệt hại trong các tình huống thời tiết bất lợi, hỗ trợ ra quyết định trong các lĩnh vực yêu cầu thời gian thực như quản lý giao thông hoặc ứng phó thiên tai.

3. Tầm quan trọng và ứng dụng thực tế:

Lợi ích của dự án: Dự báo thời tiết phục vụ cho nhiều lĩnh vực, giúp cải thiện khả năng ra quyết định dựa trên dữ liệu thời tiết, giảm thiểu tác động của thiên tai, tối ưu hoá sản xuất và nâng cao chất lượng cuộc sống:

- Nông nghiệp: Giúp nông dân lên kế hoạch gieo trồng, tưới tiêu và thu hoạch chính xác hơn, giảm tổn thất do thời tiết không thuận lợi.
- Vận tải: Dự báo thời tiết chính xác giúp điều hướng giao thông hiệu quả, giảm tai nạn và ùn tắc.
- Quản lý ứng phó thiên tai: Hỗ trợ các tổ chức cứu hộ và chính quyền địa phương trong việc chuẩn bị và ứng phó kịp thời với thiên tai.

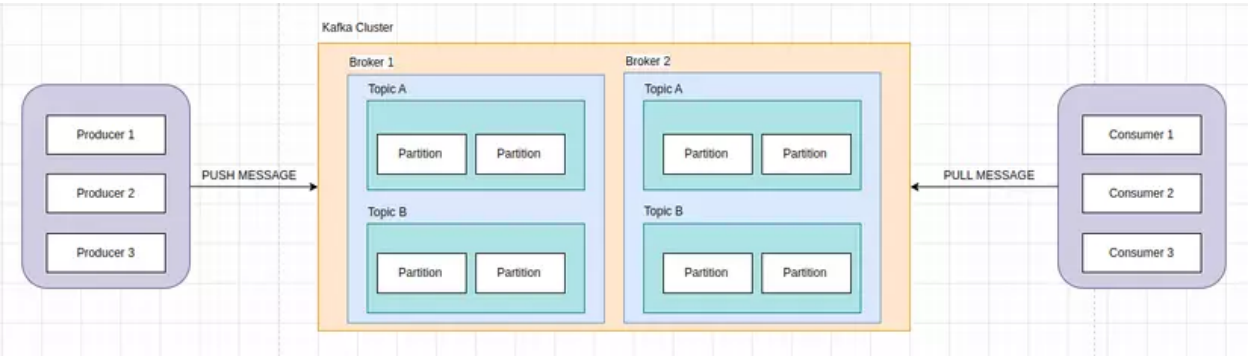
II. Dữ liệu và công nghệ sử dụng:

1. Dữ liệu

Gồm quá khứ (hơn 130k bản ghi) và thời gian thực:

- Nguồn: sử dụng dữ liệu thu thập tại vị trí Hà Nội, Việt Nam qua API open-meteo
- Dữ liệu thu thập gồm: thời gian, nhiệt độ , lượng mưa, tốc độ gió và tình trạng thời tiết.

2. Apache Kafka



2.1 Định nghĩa và lịch sử phát triển:

- Apache Kafka là một nền tảng phân tán dùng để truyền tải, lưu trữ, và xử lý dữ liệu theo thời gian thực. Kafka được thiết kế để hoạt động như một hệ thống log phân tán (distributed log) hoặc hàng đợi tin nhắn (message queue) có khả năng mở rộng và chịu lỗi cao. Kafka phù hợp với các ứng dụng yêu cầu xử lý dữ liệu lớn với độ trễ thấp, đặc biệt là trong các hệ thống phân tích dữ liệu, truyền tải sự kiện (event streaming), và các hệ thống tích hợp dữ liệu.
- Kafka được phát triển lần đầu bởi LinkedIn vào năm 2011 để xử lý dữ liệu log lớn trong nội bộ. Sau đó được chuyển sang dự án mã nguồn mở thuộc tổ chức Apache Software Foundation vào năm 2012.
- Kafka ngày nay được các công ty lớn như Netflix, Uber, và Airbnb sử dụng để xử lý hàng triệu sự kiện mỗi giây.

2.2 Kiến trúc: Kiến trúc Kafka được xây dựng xung quanh các thành phần:

a, Broker

- Broker là các máy chủ trung gian chịu trách nhiệm lưu trữ và phân phối dữ liệu.
- Một cụm Kafka (Kafka Cluster) có thể bao gồm nhiều broker, đảm bảo khả năng chịu lỗi và mở rộng dễ dàng.
- Dữ liệu được lưu trữ trong các broker dưới dạng log phân tán.

b, Topic và Partition

- Topic: Là kênh dữ liệu mà các producer gửi thông điệp vào và consumer nhận dữ liệu ra.
- Partition: Mỗi topic được chia thành nhiều phân vùng (partition) để tăng khả năng xử lý song song. Mỗi partition được lưu trữ trên một hoặc nhiều broker và có thứ tự tin nhắn riêng.

c, Producer và Consumer

- Producer: Là các ứng dụng hoặc dịch vụ gửi dữ liệu vào topic.
- Consumer: Là các ứng dụng đọc dữ liệu từ topic. Consumer có thể đọc từ nhiều partition và duy trì offset để biết đã xử lý đến đâu.

d, Zookeeper (hoặc Kafka Raft)

- Zookeeper (trước đây) quản lý metadata và điều phối giữa các broker.
- Từ Kafka 2.8 trở đi, Kafka chuyển sang sử dụng giao thức Kafka Raft (KRaft) thay thế Zookeeper để đơn giản hóa quản trị và tăng hiệu suất.

2.3 Hoạt động của Apache Kafka

a, Producer gửi dữ liệu vào topic:

- Dữ liệu có thể là log ứng dụng, sự kiện người dùng, hoặc bất kỳ thông điệp nào.
- Kafka đảm bảo tin nhắn được ghi vào topic theo thứ tự (tại từng partition).

b, Lưu trữ dữ liệu trong broker:

- Dữ liệu được lưu trữ trên ổ đĩa của broker, đảm bảo tính bền vững (durability).
- Mỗi broker có thể chịu trách nhiệm cho nhiều partition khác nhau.

c, Consumer đọc dữ liệu từ topic:

- Consumer có thể đăng ký với một hoặc nhiều topic.
- Consumer duy trì offset để đảm bảo không đọc lặp dữ liệu.

d, Replication và tính chịu lỗi:

- Kafka hỗ trợ replication (sao chép partition) để đảm bảo dữ liệu không bị mất khi một broker gặp sự cố.

2.4 Ưu điểm và nhược điểm của Apache Kafka

a, Ưu điểm:

- Hiệu suất cao: Kafka xử lý hàng triệu tin nhắn mỗi giây với độ trễ thấp.
- Khả năng mở rộng: Có thể dễ dàng thêm broker vào cụm mà không ảnh hưởng đến hiệu suất.
- Đảm bảo tính bền vững: Dữ liệu được lưu trữ trên đĩa và có thể sao chép giữa các broker.
- Hỗ trợ thứ tự: Tin nhắn trong từng partition được duy trì theo thứ tự.
- Tính tương thích: Dễ dàng tích hợp với nhiều công nghệ dữ liệu lớn như Spark, Flink, Hadoop.

b, Nhược điểm:

- Yêu cầu quản lý phức tạp.

2.5 Ứng dụng thực tế của Apache Kafka

a, Xử lý log và giám sát hệ thống:

- Kafka thường được sử dụng để truyền tải log ứng dụng đến các hệ thống phân tích như Elasticsearch.

b, Stream Processing:

- Tích hợp với Spark Streaming, Flink, hoặc Kafka Streams để phân tích dữ liệu thời gian thực.

c, Hệ thống thông báo:

- Ứng dụng thương mại điện tử, ngân hàng.Kafka được sử dụng làm nền tảng truyền tải thông báo trong các ứng

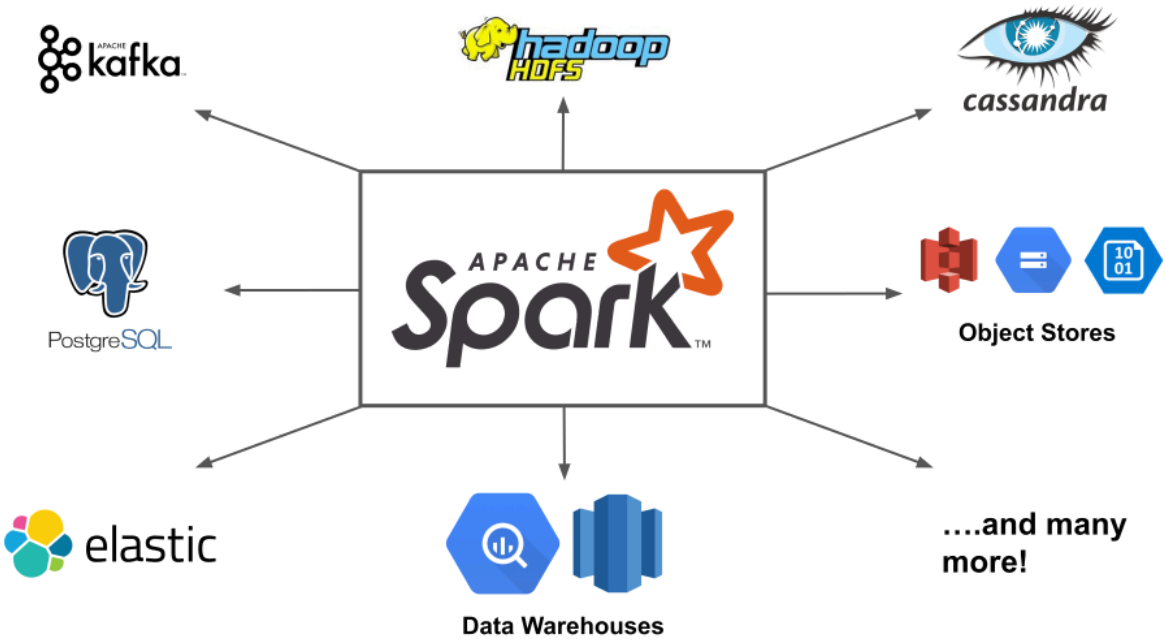
d, Tích hợp dữ liệu (ETL):

- Kafka làm trung gian để đồng bộ dữ liệu từ nhiều nguồn (database, hệ thống IoT) vào các kho dữ liệu.

e, Quản lý sự kiện trong microservices:

- Trong kiến trúc microservices, Kafka thường được sử dụng để truyền tải sự kiện giữa các dịch vụ.

3. PySpark



3.1 Định nghĩa và Lịch sử phát triển

a. Định nghĩa:

PySpark là một API dành cho Python, được phát triển trên nền tảng Apache Spark. PySpark cho phép lập trình viên sử dụng ngôn ngữ Python để xử lý và phân tích dữ liệu lớn (big data) trên các cụm máy tính (clusters). Apache Spark, cốt lõi của PySpark, là một hệ thống tính toán phân tán mã nguồn mở với khả năng xử lý dữ liệu nhanh chóng nhờ vào việc sử dụng bộ nhớ (in-memory computing). Ngoài khả năng xử lý dữ liệu theo lô truyền thống, Spark còn hỗ trợ xử lý thời gian thực thông qua module Spark Streaming, cho phép xử lý luồng dữ liệu liên tục từ các nguồn như Kafka, Flume, hay socket.

b. Lịch sử phát triển:

Apache Spark được phát triển tại UC Berkeley AMPLab và ra mắt lần đầu vào năm 2014. PySpark được phát triển để hỗ trợ các nhà khoa học dữ liệu và lập trình viên Python dễ dàng làm việc với Spark, thay thế MapReduce của Hadoop. Module Spark Streaming được giới thiệu nhằm đáp ứng nhu cầu xử lý dữ liệu thời gian thực, trở thành một thành phần không thể thiếu trong hệ sinh thái của Spark.

3.2 Kiến trúc

Kiến trúc PySpark dựa trên Apache Spark và bao gồm các thành phần chính sau:

a. Driver Program

- Chịu trách nhiệm khởi tạo ứng dụng PySpark và điều phối các tác vụ trên các worker node.
- Chương trình PySpark mà bạn viết (bằng Python) chính là driver program.

b. Cluster Manager

- Quản lý tài nguyên của cụm (cluster) để phân phối các tác vụ đến worker node.
- PySpark hỗ trợ các cluster manager phổ biến như Standalone, YARN, và Kubernetes.

c. Worker Node

- Là nơi thực hiện các tác vụ (tasks) do driver chỉ định.
- Worker chứa các tiến trình executor, chịu trách nhiệm thực thi mã lệnh và lưu trữ dữ liệu.

d. Resilient Distributed Dataset (RDD)

- Là cấu trúc dữ liệu cốt lõi trong Spark, cho phép lưu trữ và xử lý dữ liệu trên nhiều node.
- RDD hỗ trợ tính bền vững (resilience) bằng cách ghi lại lịch sử các phép biến đổi, đảm bảo khả năng phục hồi khi xảy ra lỗi.

e. Streaming Layer

- Trong Spark Streaming, dữ liệu thời gian thực được chia thành các lô nhỏ (micro-batches) để xử lý theo từng khung thời gian.
- Spark Streaming tích hợp chặt chẽ với các thành phần còn lại của Spark, giúp tận dụng hiệu quả các công cụ như RDD, DataFrame, và MLlib.

3.3 Các thành phần chính của PySpark

a. Resilient Distributed Dataset (RDD)

- Định nghĩa: RDD là một tập hợp dữ liệu được phân tán trên nhiều node, cho phép xử lý song song.
- Đặc điểm chính:

    - *Immutability*: Một khi đã tạo, RDD không thể thay đổi mà chỉ có thể tạo RDD mới từ nó.
    - *Partitioning*: Dữ liệu trong RDD được tự động phân vùng.
    - *Fault Tolerance*: RDD có khả năng phục hồi tự động khi một node bị lỗi.
- Phép biến đổi:
    - *Transformation*: Tạo ra RDD mới từ RDD cũ (ví dụ: map, filter, groupBy).
    - *Action*: Thực thi tính toán trên RDD và trả về kết quả (ví dụ: collect, count).

b. DataFrame

- Định nghĩa: DataFrame là một cấu trúc dữ liệu dạng bảng, giống như bảng trong cơ sở dữ liệu hoặc DataFrame trong pandas.
- Ưu điểm:
    - Tích hợp mạnh mẽ với SQL (hỗ trợ Spark SQL).
    - Tối ưu hóa tự động nhờ công cụ Catalyst Optimizer.
- Các thao tác phổ biến:
    - Tạo DataFrame từ file CSV, JSON, Parquet hoặc từ RDD.
    - Truy vấn dữ liệu bằng các hàm như select, filter, groupBy, hoặc câu lệnh SQL.

c. Spark SQL

- Định nghĩa: Spark SQL là một module của Spark hỗ trợ truy vấn dữ liệu bằng ngôn ngữ SQL.
- Đặc điểm nổi bật:
    - Có thể chạy các truy vấn SQL trực tiếp trong PySpark.
    - Hỗ trợ tích hợp với các công cụ BI (Business Intelligence) như Tableau, Power BI.

d. Spark Streaming

- Định nghĩa: Spark Streaming là module hỗ trợ xử lý dữ liệu thời gian thực trong Apache Spark.
- Đặc điểm nổi bật:
    - Phân chia luồng dữ liệu thành các lô nhỏ để xử lý (micro-batch).
    - Tích hợp với các nguồn dữ liệu phổ biến như Kafka, Flume, socket và các hệ thống file.
- Ứng dụng:
    - Xử lý luồng dữ liệu từ cảm biến IoT, mạng xã hội, hoặc nhật ký hệ thống (log).

    - Phát hiện các sự kiện bất thường trong thời gian thực, như giao dịch gian lận hoặc giám sát hiệu suất hệ thống.

e. Machine Learning với MLlib

- PySpark MLlib là thư viện máy học tích hợp sẵn của Spark.
- Hỗ trợ:
    - Các thuật toán học máy phổ biến: hồi quy (regression), phân cụm (clustering), cây quyết định (decision tree),...
    - Tiền xử lý dữ liệu: chuẩn hóa, chuyển đổi đặc trưng (feature transformation).

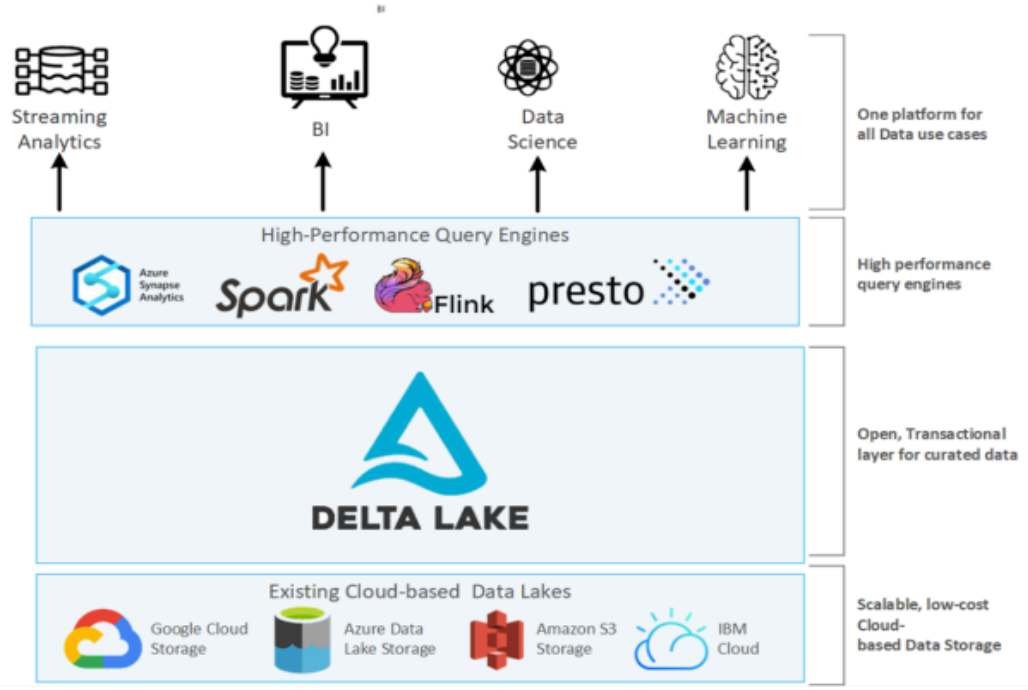
3.4 Ưu điểm của PySpark

- Xử lý song song: Tận dụng sức mạnh của cluster để xử lý dữ liệu nhanh hơn so với các thư viện Python truyền thống.
- Tương thích: Hỗ trợ nhiều định dạng dữ liệu lớn như HDFS, Cassandra, Amazon S3, và MongoDB.
- Khả năng mở rộng: Hoạt động tốt trên cả cụm nhỏ và cụm lớn với hàng ngàn node.
- Tích hợp dễ dàng: Kết hợp tốt với các công cụ dữ liệu lớn khác như Hadoop, Kafka, và Delta Lake.

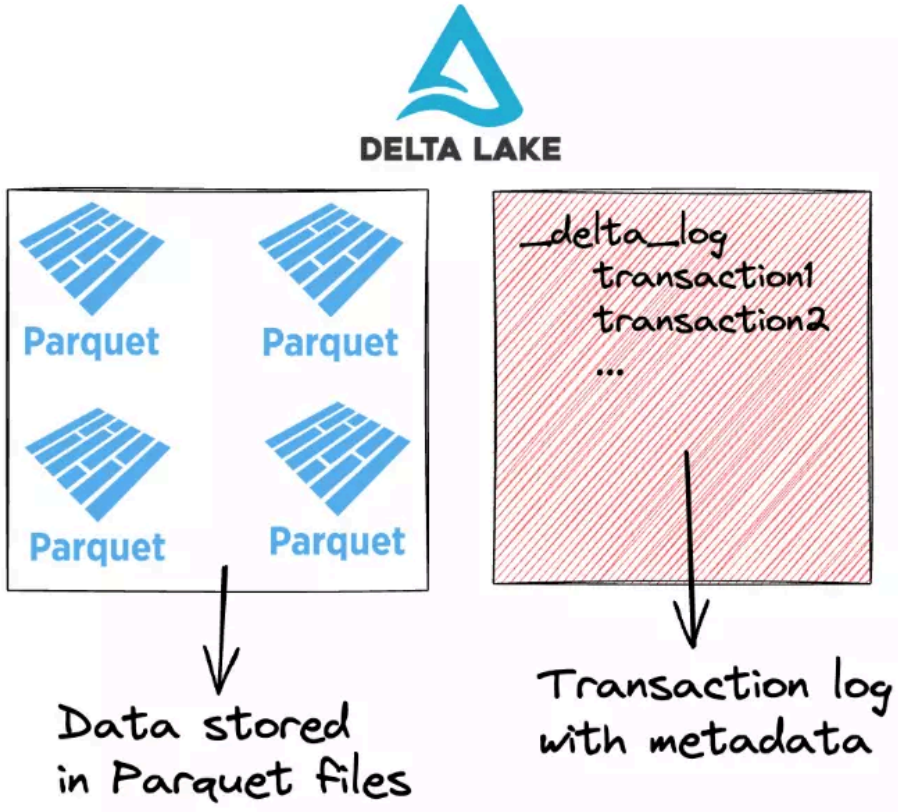
3.5 Ứng dụng thực tế của PySpark

- Phân tích dữ liệu lớn: Xử lý dữ liệu log, giao dịch khách hàng, hoặc thông tin từ hệ thống IoT.
- Xây dựng mô hình máy học: Sử dụng PySpark MLlib để huấn luyện mô hình trên dữ liệu lớn, chẳng hạn dự đoán doanh thu hoặc phân cụm khách hàng.
- Xử lý luồng dữ liệu thời gian thực: Kết hợp PySpark Streaming với Kafka hoặc Flume để xử lý dữ liệu trực tiếp từ các nguồn luồng.
- Tích hợp với hệ thống lưu trữ dữ liệu: PySpark hỗ trợ ETL (Extract, Transform, Load) để đồng bộ dữ liệu từ nhiều nguồn vào kho dữ liệu tập trung.

4. Delta Lake



4.1 Định nghĩa và lịch sử phát triển:



a, Định nghĩa: Delta Lake là một lớp lưu trữ mã nguồn mở được xây dựng trên Apache Spark. Nó cung cấp khả năng lưu trữ dữ liệu đáng tin cậy, hiệu quả, và nhất quán bằng cách giải quyết các vấn đề phổ biến trong hệ thống lưu trữ dữ liệu truyền thống như lỗi ghi đè (overwrite), dữ liệu trùng lặp, hoặc không đồng bộ.

Delta Lake kết hợp các ưu điểm của lakes (data lakes) và warehouses (data warehouses):

- Linh hoạt: Như data lake.
- Đồng nhất và đáng tin cậy: Như data warehouse.

b, Lịch sử phát triển:Delta Lake được phát triển bởi Databricks, công ty sáng lập Apache Spark.

- Ra mắt lần đầu vào năm 2019.
- Hiện tại, Delta Lake là một dự án mã nguồn mở thuộc Linux Foundation.

4.2 Kiến trúc:

Delta Lake được xây dựng trên Apache Parquet và sử dụng định dạng Delta file format để lưu trữ metadata và dữ liệu. Kiến trúc của nó bao gồm các lớp:

a, Dữ liệu thô (Raw Data) Là dữ liệu nguồn được lưu trữ dưới dạng file Parquet trong hệ thống file phân tán (HDFS, Amazon S3, Azure Blob).

b, Metadata Layer

- Delta Lake thêm một lớp metadata để ghi lại lịch sử các phiên làm việc (transactions) và phiên bản của dữ liệu.
- Sử dụng Apache Spark để quản lý metadata trong các file JSON.

c, Delta Table: Là bảng dữ liệu được xây dựng trên định dạng Delta Lake, hỗ trợ tính năng ACID, versioning và indexing.

4.3 Các tính năng chính của Delta Lake

a, Hỗ trợ ACID Transactions

- Delta Lake đảm bảo tính atomic (tính nguyên tử), consistency (nhất quán), isolation (cách ly), và durability (bền vững) trong quá trình xử lý dữ liệu.
- Ví dụ: Nếu một tác vụ ghi dữ liệu thất bại, Delta Lake sẽ tự động hoàn nguyên trạng thái về trước khi tác vụ xảy ra.

b, Lưu phiên bản dữ liệu (Time Travel)

- Delta Lake lưu lại mọi thay đổi trong lịch sử dữ liệu, cho phép bạn quay lại phiên bản trước đó để kiểm tra hoặc phục hồi dữ liệu.

c, Dữ liệu không trùng lặp (Data Deduplication): Hỗ trợ tích hợp cơ chế Merge và Upsert để loại bỏ dữ liệu trùng lặp một cách hiệu quả.

d, Quản lý dữ liệu bị thiếu hoặc lỗi (Schema Enforcement & Evolution)

- Schema Enforcement: Bảo vệ dữ liệu bằng cách xác minh rằng schema (cấu trúc) của dữ liệu đầu vào phải khớp với schema định nghĩa trước.
- Schema Evolution: Tự động cập nhật schema khi có thêm cột hoặc thuộc tính mới.

e, Hỗ trợ dữ liệu quy mô lớn (Scalable Metadata Management)

- Metadata trong Delta Lake được tối ưu hóa để quản lý hàng tỷ file và partition mà vẫn đảm bảo hiệu năng cao.

f, Streaming và Batch Integration:

- Delta Lake hỗ trợ cả xử lý batch và streaming trong một hệ thống duy nhất. Điều này giúp đồng bộ luồng dữ liệu trực tiếp và dữ liệu lịch sử một cách dễ dàng.

4.4 Cách hoạt động của Delta Lake

a, Đọc dữ liệu từ Delta Table: Delta Lake sử dụng Spark để truy vấn và xử lý dữ liệu giống như các hệ thống SQL thông thường.

b, Ghi dữ liệu vào Delta Table: Mọi thay đổi dữ liệu đều được ghi lại dưới dạng transaction (phiên làm việc), và mỗi transaction đi kèm một phiên bản mới của bảng dữ liệu.

c, Quản lý lịch sử phiên bản: Các file log ghi lại thông tin chi tiết về từng phiên làm việc, giúp dễ dàng truy xuất lại dữ liệu cũ.

4.5 Ưu điểm của Delta Lake

- Đảm bảo độ tin cậy: Tính năng ACID đảm bảo dữ liệu không bị lỗi hoặc trùng lặp.
- Lưu trữ tối ưu: Sử dụng định dạng Parquet giúp nén dữ liệu tốt hơn.
- Dễ tích hợp: Làm việc mượt mà với Spark, Kafka, Hive, và các hệ thống big data khác.
- Đơn giản hóa quy trình ETL: Hỗ trợ các tác vụ như merge, upsert, và quản lý schema một cách dễ dàng.

4.6 Ứng dụng thực tế của Delta Lake

a, Phân tích dữ liệu lịch sử và thời gian thực:

- Kết hợp batch và streaming để phân tích cả dữ liệu cũ và luồng dữ liệu mới.
- Ví dụ: Theo dõi hành vi khách hàng trong thương mại điện tử.

b, Xây dựng kho dữ liệu (Data Warehouse):

- Delta Lake cải thiện hiệu suất và độ tin cậy của kho dữ liệu truyền thống, đồng thời giảm chi phí lưu trữ.

c, Machine Learning và AI:

- Hỗ trợ việc lưu trữ và xử lý dữ liệu lớn cho các dự án học máy.
- Ví dụ: Tạo pipeline xử lý dữ liệu để huấn luyện mô hình AI.

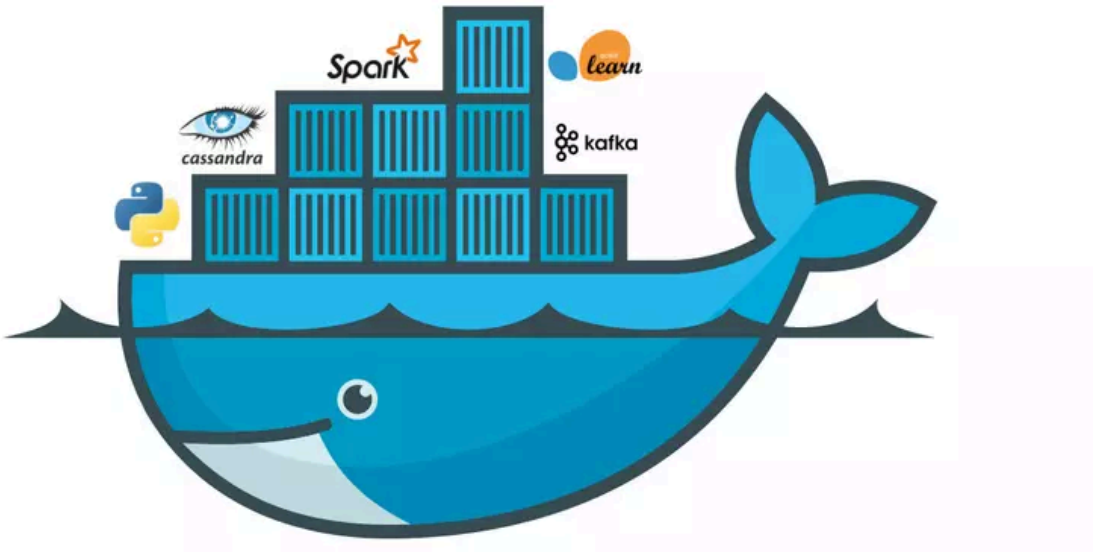
d, Quản lý dữ liệu doanh nghiệp (Enterprise Data Management):

- Delta Lake giúp doanh nghiệp quản lý hiệu quả dữ liệu phân tán trên nhiều nguồn và định dạng.

e, Tích hợp Delta Lake với Apache Spark:

- Delta Lake thường được tích hợp với PySpark để xử lý dữ liệu.

5. Docker



5.1 Giới thiệu

Hệ thống thu thập và xử lý dữ liệu thời tiết được triển khai với sự hỗ trợ của Docker nhằm đảm bảo tính linh hoạt, khả năng mở rộng và quản lý dễ dàng. Việc sử dụng Docker giúp đóng gói các thành phần của hệ thống (như Zookeeper, Kafka Broker) thành các container độc lập, dễ dàng triển khai trên nhiều nền tảng và môi trường khác nhau.

Hệ thống được cấu hình thông qua tệp Docker Compose với phiên bản 3, cho phép quản lý đồng thời nhiều dịch vụ trong một môi trường tích hợp.

5.2 Kiến trúc tổng quan:

Hệ thống Docker được thiết kế gồm hai dịch vụ chính:

a, Zookeeper:

- Đóng vai trò là trung tâm điều phối, giúp quản lý các broker trong Apache Kafka.
- Zookeeper đảm bảo tính nhất quán của hệ thống Kafka khi làm việc với nhiều broker, đồng thời lưu trữ thông tin cấu hình cần thiết.

b, Kafka Broker:

- Là thành phần chính trong Apache Kafka, chịu trách nhiệm lưu trữ và truyền tải dữ liệu giữa các producer và consumer.
- Broker quản lý các topic, xử lý các thông điệp từ producer và phân phối chúng tới consumer.
- Trong cấu hình này, chúng tôi triển khai một broker đơn lẻ, tuy nhiên, hệ thống hoàn toàn có thể mở rộng để thêm nhiều broker khi cần.
- Các container được kết nối với nhau trong cùng một mạng Docker, đảm bảo việc giao tiếp giữa các thành phần thông qua địa chỉ nội bộ.

5.3 Tính năng chính:

a, Môi trường nhất quán:

- Containerization: Đảm bảo Spark, Kafka và các công cụ hoạt động đồng nhất trên nhiều nền tảng, tránh sự khác biệt giữa các môi trường phát triển, kiểm thử và triển khai thực tế.
- Dependency Management: Đóng gói các thư viện, công cụ cần thiết trong container giúp dễ quản lý và tái sử dụng.

b, Dễ dàng triển khai và mở rộng

- Đa nền tảng: Docker containers có thể chạy trên nhiều nền tảng khác nhau (Ubuntu, Windows, macOS), giúp dễ triển khai hệ thống.
- Khả năng mở rộng: Docker Swarm hoặc Kuburneter cho phép dễ dàng mở rộng khi khối lượng dữ liệu tăng.

c, Phân tách các thành phần hệ thống

d, Tích hợp CI/CD:

- Docker tích hợp tốt với các pipeline CI/CD để tự động hoá việc build, kiểm thử và triển khai các container.

e, Tăng cường khả năng tái sử dụng, giảm chi phí tài nguyên

f, Hỗ trợ công cụ bổ trợ:

- Docker Compose: Tạo và quản lý nhiều container (Kafka, Spark Master, Spark Worker, Jupyter Notebook) dễ dàng bằng cách định nghĩa trong 1 file docker-compose.yml.

- Docker Hub: Hỗ trợ tải xuống các image chuẩn của Dpark, Kafka, hoặc cả Python ML runtime từ Docker Hub.

6. Power BI

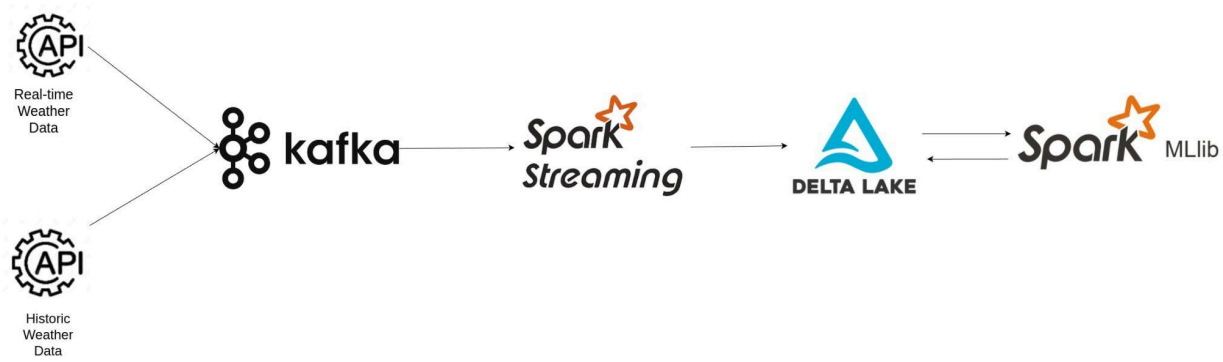
Power BI là một công cụ phân tích dữ liệu mạnh mẽ được phát triển bởi Microsoft, cho phép người dùng trực quan hóa và chia sẻ dữ liệu dưới dạng biểu đồ, báo cáo và dashboard. Với giao diện thân thiện, Power BI giúp kết nối, xử lý, và tích hợp dữ liệu từ nhiều nguồn khác nhau như Excel, SQL Server, Azure, hoặc các dịch vụ trực tuyến như Google Analytics. Công cụ này không chỉ hỗ trợ ra quyết định nhanh chóng mà còn giúp doanh nghiệp theo dõi hiệu suất và dự đoán xu hướng.

III. Thiết kế và triển khai hệ thống:

1. Mục tiêu:

Hệ thống dự báo thời tiết được thiết kế nhằm thu thập, xử lý và lưu trữ dữ liệu thời tiết theo thời gian thực từ API thời tiết với delta lake, với khả năng lưu trữ và phân tích dữ liệu lịch sử phục vụ các mục đích dự đoán hoặc báo cáo.

2. Mô tả hệ thống



2.1 Docker:

Việc sử dụng Docker giúp hệ thống thu thập và xử lý dữ liệu thời tiết trở nên linh hoạt, dễ dàng triển khai và quản lý. Với cấu hình hiện tại, hệ thống không chỉ đáp ứng tốt các yêu cầu xử lý dữ liệu thời gian thực mà còn có khả năng mở rộng và tích hợp với các công nghệ khác trong tương lai.

Cấu hình Docker Compose:

- Tệp cấu hình: Hệ thống được định nghĩa trong tệp docker-compose.yml với nội dung chính như sau:

  a, Zookeeper

  - Image: confluentinc/cp-zookeeper:7.0.1
  - Đây là phiên bản Zookeeper do Confluent cung cấp, được tối ưu hóa cho Apache Kafka.
  - Cổng kết nối:
    - Cổng 2181 được ánh xạ từ container ra môi trường host, giúp Kafka Broker và các ứng dụng khác có thể giao tiếp với Zookeeper.
  - Biến môi trường:
    - ZOOKEEPER_CLIENT_PORT: Đặt cổng giao tiếp cho client (mặc định là 2181).
    - ZOOKEEPER_TICK_TIME: Thiết lập thời gian giao tiếp giữa các node Zookeeper, đảm bảo đồng bộ.

  b, Kafka Broker

  - Image: confluentinc/cp-kafka:7.0.1

        - Là phiên bản Kafka broker do Confluent phát hành, tích hợp sẵn các công cụ cần thiết để quản lý Kafka.
    - Cổng kết nối:
        - Các cổng sau được sử dụng:
            - 29092: Cổng nội bộ để giao tiếp giữa các container trong mạng Docker.
            - 9092: Cổng dành cho client bên ngoài, như producer hoặc consumer.
            - 9101: Cổng JMX để giám sát Kafka Broker.
    - Biến môi trường:
        - KAFKA_BROKER_ID: ID duy nhất của broker trong hệ thống (mặc định là 1).
        - KAFKA_ZOOKEEPER_CONNECT: Địa chỉ kết nối tới Zookeeper (zookeeper:2181), sử dụng tên container Zookeeper.
        - KAFKA_ADVERTISED_LISTENERS:
            - Định nghĩa các giao thức và cổng mà broker có thể sử dụng.
            - PLAINTEXT dành cho các container trong mạng Docker.
            - PLAINTEXT_HOST cho phép client bên ngoài kết nối thông qua localhost:9092.
        - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: Đảm bảo các topic offset có ít nhất một bản sao.

2.2 Producer:

- Producer sử dụng Python để truy vấn dữ liệu thời tiết từ API thời tiết (WeatherAPI).
- Sau khi lấy được dữ liệu, các thông tin quan trọng như thời gian, nhiệt độ, độ ẩm, tốc độ gió... sẽ được lọc ra và gửi vào Kafka topic có tên là weatherkafkatopic.

2.3 Apache Kafka:

- Kafka đóng vai trò là hệ thống truyền tải trung gian.
- Các thông điệp được Producer gửi lên topic weatherkafkatopic sẽ được Kafka quản lý và chờ Consumer xử lý.

2.4 Consumer:

- Consumer được xây dựng bằng Spark Streaming, sử dụng cơ chế đọc dữ liệu streaming trực tiếp từ Kafka topic weatherkafkatopic.

- Dữ liệu được chuyển đổi từ định dạng JSON và áp dụng schema để đảm bảo cấu trúc nhất quán.

2.5 Delta Lake:

- Sau khi được xử lý, dữ liệu thời tiết sẽ được lưu trữ vào Delta Lake.
- Hệ thống sử dụng Delta Lake để lưu trữ dữ liệu dạng bảng với khả năng hỗ trợ lịch sử thay đổi và đảm bảo tính toàn vẹn dữ liệu (ACID).

2.6 Spark:

Sử dụng pyspark và spark mlib để phân tích, xử lý dữ liệu và triển khai mô hình dự đoán VD như random forest , linear regression, gradient boosted tree

2.7 Power BI

Quy trình phân tích trong Power BI:

- Sau khi kết nối với Delta Lake, dữ liệu thời tiết được nhập vào Power BI để xây dựng các dashboard và báo cáo tương tác.
    1. **Trực quan hóa**:
        - Tạo biểu đồ và đồ thị tương tác như biểu đồ đường (line chart) để thể hiện xu hướng nhiệt độ theo thời gian, hoặc biểu đồ thanh (bar chart) để so sánh lượng mưa giữa các khu vực.
        - Sử dụng bản đồ nhiệt (heatmap) để phân tích mức độ ảnh hưởng thời tiết tại từng địa điểm cụ thể.
    2. **Cập nhật dữ liệu thời gian thực**:
        - Power BI có thể được cấu hình để tự động làm mới dữ liệu từ Delta Lake ở các khoảng thời gian định kỳ, đảm bảo dashboard luôn hiển thị thông tin mới nhất.

3. Quy trình triển khai

3.1 Chuẩn bị Docker Compose

- Tạo tệp docker-compose.yml với cấu hình như trên trong thư mục làm việc.

3.2 Khởi chạy các container

Sử dụng lệnh: docker-compose up -d

- up: Khởi động các container được định nghĩa trong tệp docker-compose.yml.
- -d: Chạy các container ở chế độ nền (detached mode).

3.3 Kiểm tra trạng thái container

Kiểm tra các container đã khởi chạy: docker ps

- Đảm bảo hai container zookeeper và broker đang hoạt động bình thường.

3.4 Kết nối Kafka từ ứng dụng

- Producer và Consumer (Spark Streaming) kết nối tới Kafka thông qua địa chỉ localhost:9092.
- Nếu ứng dụng chạy trong container khác trong cùng mạng Docker, sử dụng địa chỉ broker:29092.

4. Quy trình thu thập dữ liệu

Dữ liệu cho dự án này được thu thập từ **Open-Meteo API**, một nền tảng cung cấp dữ liệu thời tiết miễn phí và phong phú, hỗ trợ truy xuất dữ liệu theo yêu cầu

Các loại thông tin đặc trưng được lấy từ nguồn dữ liệu này bao gồm:

- **Thời gian (Time):** Thời điểm ghi nhận dữ liệu, được lưu trữ theo định dạng ngày/giờ (ISO 8601), đảm bảo tính chính xác và thống nhất.
- **Nhiệt độ (Temperature):** Giá trị nhiệt độ được đo (tính bằng độ C), phản ánh tình trạng nhiệt độ tại từng thời điểm cụ thể.
- **Lượng mưa (Precipitation):** Lượng mưa ghi nhận (tính bằng mm), dùng để phân tích mức độ ẩm ướt hoặc các hiện tượng mưa lớn.
- **Tốc độ gió (Wind Speed):** Tốc độ gió trung bình (tính bằng km/h hoặc m/s), cung cấp thông tin về điều kiện gió trong khu vực.
- Tình trạng thời tiết (Condition): Nắng , mưa ,gió mạnh, gió vừa,..

Dữ liệu được thu thập thông qua API của Open-Meteo với khả năng hỗ trợ truy xuất linh hoạt theo nhiều khoảng thời gian và loại thông tin cụ thể. Phương pháp thu thập dữ liệu được thực hiện theo các bước sau:

1. **Kết nối tới Open-Meteo API:** Sử dụng thư viện Python phổ biến như requests để lấy dữ liệu từ API.

2. **Lựa chọn khung thời gian phù hợp:** Dữ liệu được thu thập theo các khoảng thời gian cố định (hàng giờ, hàng ngày hoặc hàng tháng) nhằm phục vụ phân tích đặc thù của dự án.
3. **Lưu trữ dữ liệu:** Dữ liệu sau khi thu thập được lưu trữ dưới định dạng **delta lake table** để thuận tiện cho việc xử lý và phân tích sau này.

Dữ liệu liên tục

a. Thu thập dữ liệu:
Producer thực hiện truy vấn dữ liệu thời tiết từ API theo khoảng thời gian định kỳ mỗi khi có cập nhật mới.

b. Gửi dữ liệu:
Dữ liệu được gửi tới Kafka topic weatherkafkatopic.

c. Xử lý dữ liệu:
Consumer (Spark Streaming) đọc luồng dữ liệu từ Kafka và chuyển đổi dữ liệu sang cấu trúc xác định (schema).

d. Lưu trữ dữ liệu:
Consumer lưu dữ liệu vào Delta Lake với chế độ ghi luồng (streaming). Dữ liệu được lưu trữ theo mô hình bảng để phục vụ truy vấn và phân tích.

e. Ưu điểm của hệ thống

- Thời gian thực: Hệ thống có khả năng thu thập và xử lý dữ liệu ngay khi được tạo ra từ API.
- Mở rộng: Sử dụng Apache Kafka và PySpark, hệ thống có thể dễ dàng mở rộng để xử lý dữ liệu lớn.
- Tính bền vững: Dữ liệu được lưu trữ trong Delta Lake với khả năng hỗ trợ giao dịch ACID và quản lý lịch sử thay đổi, đảm bảo tính chính xác và nhất quán.

Dữ liệu lịch sử:

Dữ liệu thu thập được bao gồm hơn **130.000 bản ghi** trong khoảng thời gian kéo dài **15 năm** (từ đầu năm 2010 đến cuối năm 2024) tại khu vực **Hà Nội, Việt Nam**.

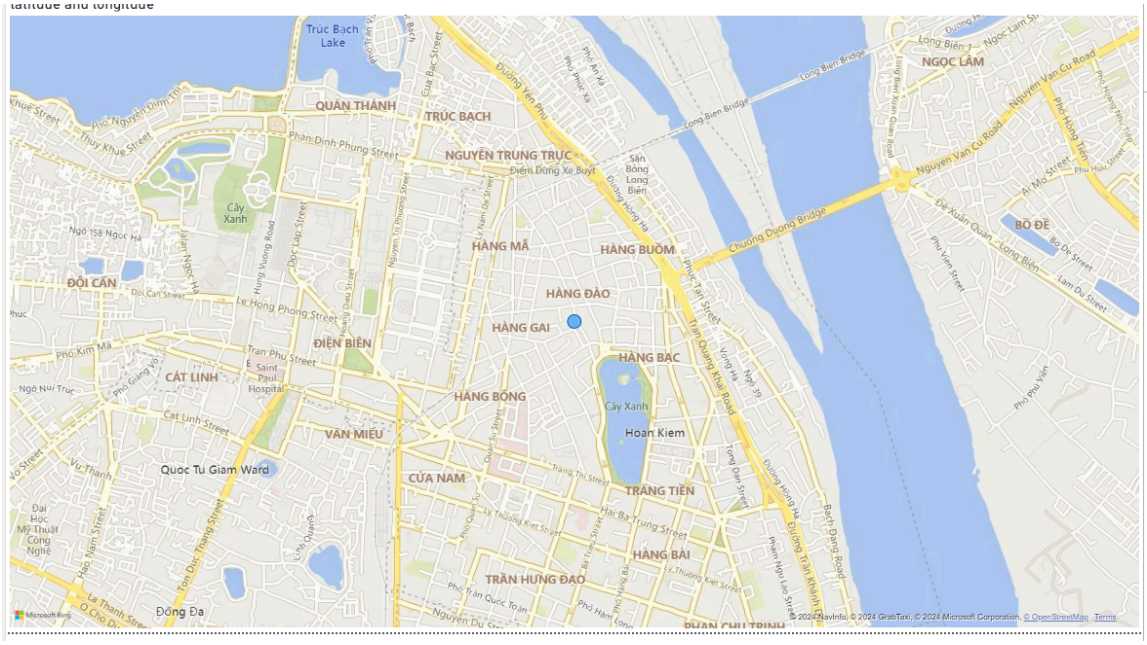
5. Phân tích dữ liệu với Power BI

Dữ liệu bao gồm các thông tin về:

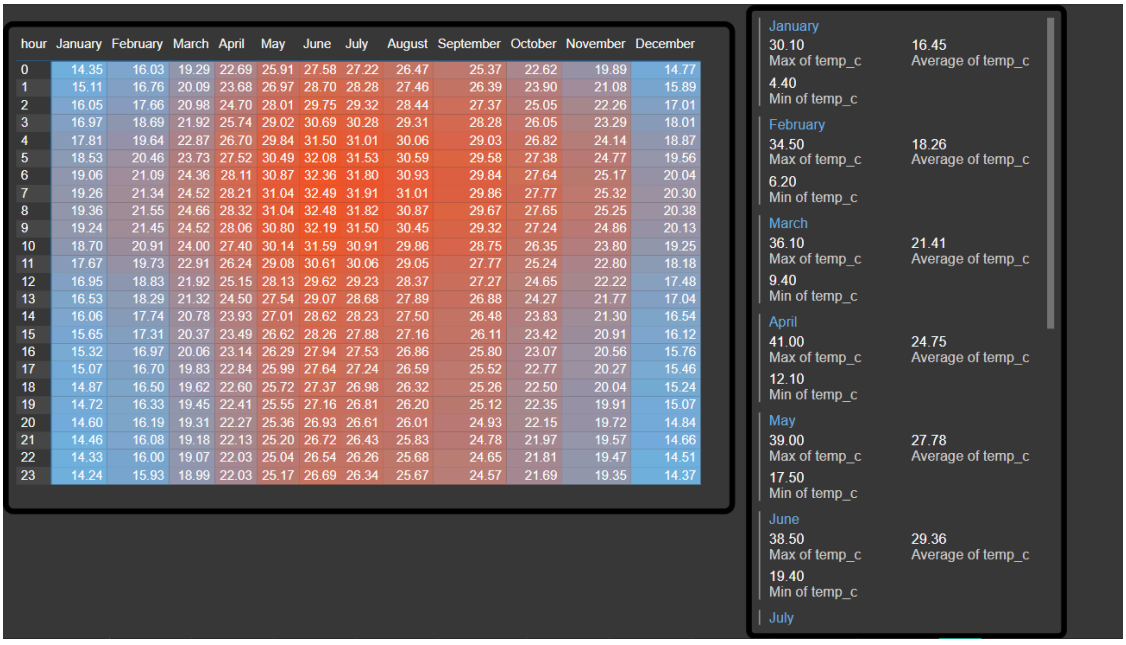
- Datetime: dữ liệu thời gian(ngày/tháng/năm giờ)
- Name: địa điểm thu thập dữ liệu(Hà Nội)
- Country: đất nước (Việt Nam)
- Latitude và longitude(vị trí trên bản đồ)
- Temp_c: nhiệt độ
- Wind_mph : tốc độ gió
- Precip_mm: lượng mưa
- Condition: trình trạng thời tiết tại thời điểm đó

datetime	name	country	latitude	longitude	temp_c	wind_mph	humidity	precip_mm	condition
29/10/2024 5:00:00 PM	Hanoi	Vietnam	21.0333	105.85	22.5	4.9	72	0	Partly cloudy
29/10/2024 6:00:00 PM	Hanoi	Vietnam	21.0333	105.85	22.1	5.4	73	0	Partly cloudy
29/10/2024 7:00:00 PM	Hanoi	Vietnam	21.0333	105.85	21.6	5.1	75	0	Partly cloudy
29/10/2024 8:00:00 PM	Hanoi	Vietnam	21.0333	105.85	21.2	5.6	77	0	Partly cloudy
29/10/2024 9:00:00 PM	Hanoi	Vietnam	21.0333	105.85	21	5.6	78	0	Partly cloudy
29/10/2024 10:00:00 PM	Hanoi	Vietnam	21.0333	105.85	21	5.4	78	0	Partly cloudy
29/10/2024 11:00:00 PM	Hanoi	Vietnam	21.0333	105.85	20.9	5.4	78	0	Partly cloudy
30/10/2024 12:00:00 AM	Hanoi	Vietnam	21.0333	105.85	21.6	5.8	76	0	Partly cloudy
30/10/2024 4:00:00 AM	Hanoi	Vietnam	21.0333	105.85	26.1	4.3	59	0	Partly cloudy
30/10/2024 5:00:00 AM	Hanoi	Vietnam	21.0333	105.85	27	5.1	55	0	Partly cloudy
30/10/2024 6:00:00 AM	Hanoi	Vietnam	21.0333	105.85	27.6	6.5	54	0	Partly cloudy
30/10/2024 7:00:00 AM	Hanoi	Vietnam	21.0333	105.85	27.9	6.7	52	0	Partly cloudy
30/10/2024 8:00:00 AM	Hanoi	Vietnam	21.0333	105.85	27.7	6	52	0	Partly cloudy
30/10/2024 9:00:00 AM	Hanoi	Vietnam	21.0333	105.85	27.3	4.9	53	0	Partly cloudy
30/10/2024 10:00:00 AM	Hanoi	Vietnam	21.0333	105.85	26.2	3.4	59	0	Partly cloudy
30/10/2024 11:00:00 AM	Hanoi	Vietnam	21.0333	105.85	25.3	4	60	0	Partly cloudy
30/10/2024 12:00:00 PM	Hanoi	Vietnam	21.0333	105.85	24.9	3.6	62	0	Partly cloudy
30/10/2024 1:00:00 PM	Hanoi	Vietnam	21.0333	105.85	24.5	3.4	64	0	Partly cloudy
30/10/2024 2:00:00 PM	Hanoi	Vietnam	21.0333	105.85	24.2	3.4	66	0	Partly cloudy
30/10/2024 3:00:00 PM	Hanoi	Vietnam	21.0333	105.85	23.9	3.6	67	0	Partly cloudy
30/10/2024 4:00:00 PM	Hanoi	Vietnam	21.0333	105.85	23.6	2.7	68	0	Partly cloudy
30/10/2024 5:00:00 PM	Hanoi	Vietnam	21.0333	105.85	23.4	2.2	69	0	Partly cloudy
30/10/2024 6:00:00 PM	Hanoi	Vietnam	21.0333	105.85	23.1	1.3	70	0	Partly cloudy
30/10/2024 7:00:00 PM	Hanoi	Vietnam	21.0333	105.85	22.9	2.7	70	0	Partly cloudy
30/10/2024 8:00:00 PM	Hanoi	Vietnam	21.0333	105.85	22.6	2.7	71	0	Partly cloudy
30/10/2024 9:00:00 PM	Hanoi	Vietnam	21.0333	105.85	22.4	2.2	72	0	Partly cloudy
30/10/2024 10:00:00 PM	Hanoi	Vietnam	21.0333	105.85	22.2	2.5	72	0	Partly cloudy

- Khu vực thu thập dữ liệu(21.0333 105.05, gần trung tâm hà nội)

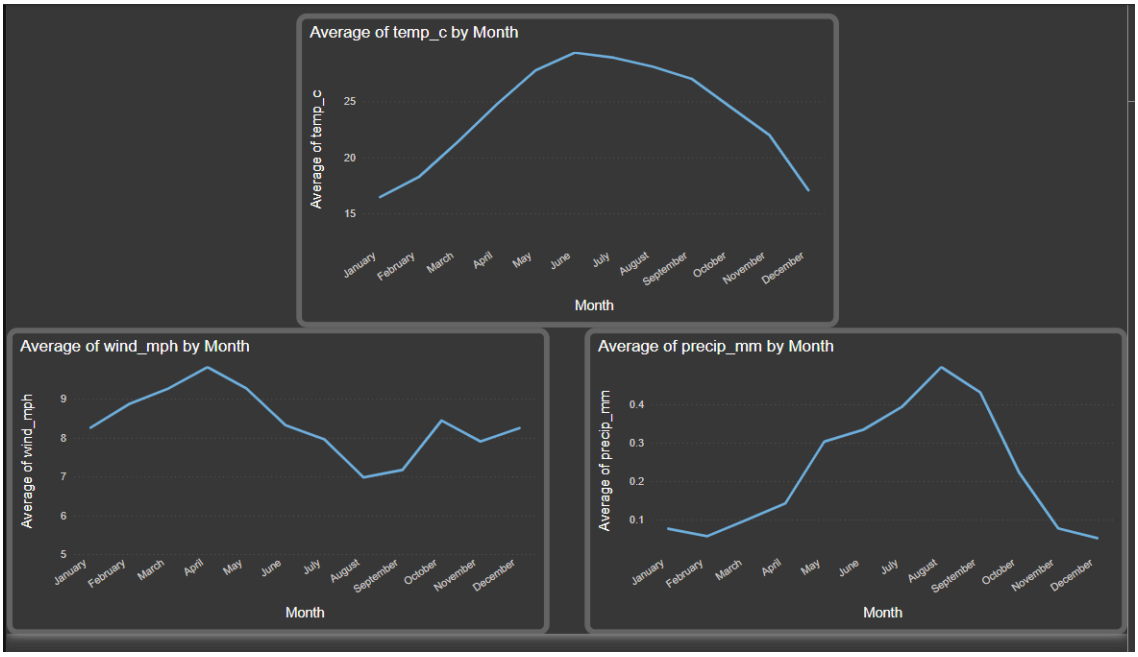


- Dữ liệu số (nhiệt độ, gió, độ ẩm, lượng mưa rơi)



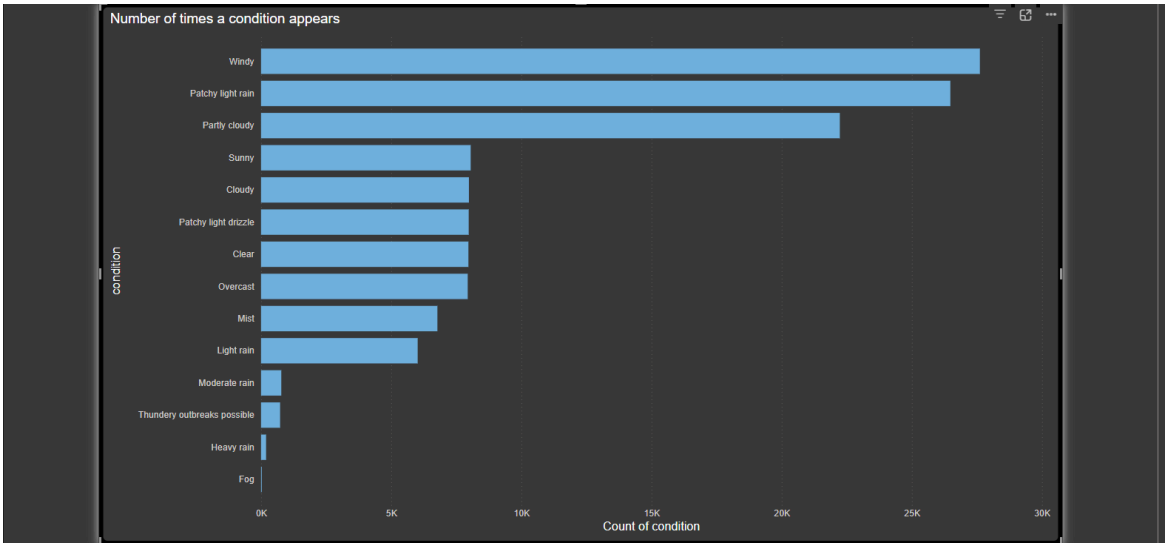
hour	January	February	March	April	May	June	July	August	September	October	November	December
0	14.35	16.03	19.29	22.69	25.91	27.58	27.22	26.47	25.37	22.62	19.89	14.77
1	15.11	16.76	20.09	23.68	26.97	28.70	28.28	27.46	26.39	23.90	21.08	15.89
2	16.05	17.66	20.98	24.70	28.01	29.75	29.32	28.44	27.37	25.05	22.26	17.01
3	16.97	18.69	21.92	25.74	29.02	30.69	30.28	29.31	28.28	26.05	23.29	18.01
4	17.81	19.64	22.87	26.70	29.84	31.50	31.01	30.06	29.03	26.82	24.14	18.87
5	18.53	20.46	23.73	27.52	30.49	32.08	31.53	30.59	29.58	27.38	24.77	19.56
6	19.06	21.09	24.36	28.11	30.87	32.36	31.80	30.93	29.84	27.64	25.17	20.04
7	19.26	21.34	24.52	28.21	31.04	32.49	31.91	31.01	29.86	27.77	25.32	20.30
8	19.36	21.55	24.66	28.32	31.04	32.48	31.82	30.87	29.67	27.65	25.25	20.38
9	19.24	21.45	24.52	28.06	30.80	32.19	31.50	30.45	29.32	27.24	24.86	20.13
10	18.70	20.91	24.00	27.40	30.14	31.59	30.91	29.86	28.75	26.35	23.80	19.25
11	17.67	19.73	22.91	26.24	29.08	30.61	30.06	29.05	27.77	25.24	22.80	18.18
12	16.95	18.83	21.92	25.15	28.13	29.62	29.23	28.37	27.27	24.65	22.22	17.48
13	16.53	18.29	21.32	24.50	27.54	29.07	28.68	27.89	26.88	24.27	21.77	17.04
14	16.06	17.74	20.78	23.93	27.01	28.62	28.23	27.50	26.48	23.83	21.30	16.54
15	15.65	17.31	20.37	23.49	26.62	28.26	27.88	27.16	26.11	23.42	20.91	16.12
16	15.32	16.97	20.06	23.14	26.29	27.94	27.53	26.86	25.80	23.07	20.56	15.76
17	15.07	16.70	19.83	22.84	25.99	27.64	27.24	26.59	25.52	22.77	20.27	15.46
18	14.87	16.50	19.62	22.60	25.72	27.37	26.98	26.32	25.26	22.50	20.04	15.24
19	14.72	16.33	19.45	22.41	25.55	27.16	26.81	26.20	25.12	22.35	19.91	15.07
20	14.60	16.19	19.31	22.27	25.36	26.93	26.61	26.01	24.93	22.15	19.72	14.84
21	14.46	16.08	19.18	22.13	25.20	26.72	26.43	25.83	24.78	21.97	19.57	14.66
22	14.33	16.00	19.07	22.03	25.04	26.54	26.26	25.68	24.65	21.81	19.47	14.51
23	14.24	15.93	18.99	22.03	25.17	26.69	26.34	25.67	24.57	21.69	19.35	14.37

- Biểu đồ độ ẩm, nhiệt độ, tốc độ gió, lượng mưa trung bình theo tháng
- Tháng có nhiệt độ thấp nhất là tháng 1(tới 8.9 C)
- Tháng có nhiệt độ cao nhất là tháng 4(tới 43.8 C)
- Các tháng có nhiệt độ trung bình là 28 C, standard deviation là 5.1



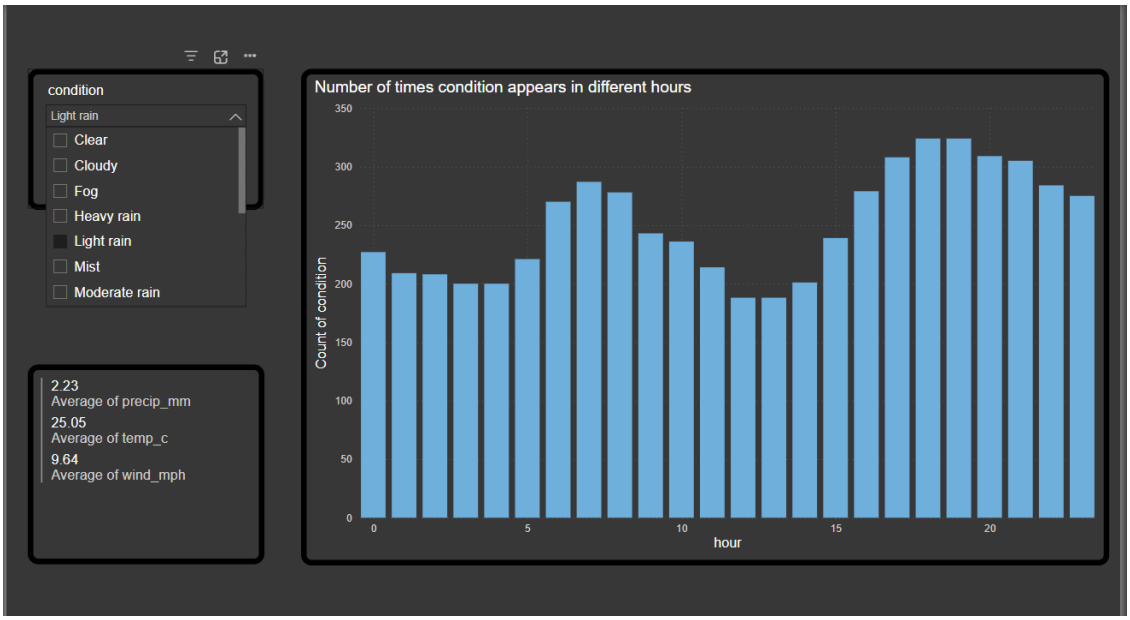
- Nhiệt độ: Tăng dần từ tháng 1 và đạt đỉnh vào tháng 6, sau đó giảm dần đến cuối năm.
- Tốc độ gió: Tốc độ trung bình giao động mạnh, đạt đỉnh vào tháng 4 và thấp nhất vào tháng 7, sau đó tăng nhẹ vào cuối năm.
- Lượng mưa: Thấp nhất vào đầu năm và tăng đáng kể vào tháng 7 và đạt đỉnh vào tháng 9

- Dữ liệu classification(các tình trạng thời tiết - weather conditions)



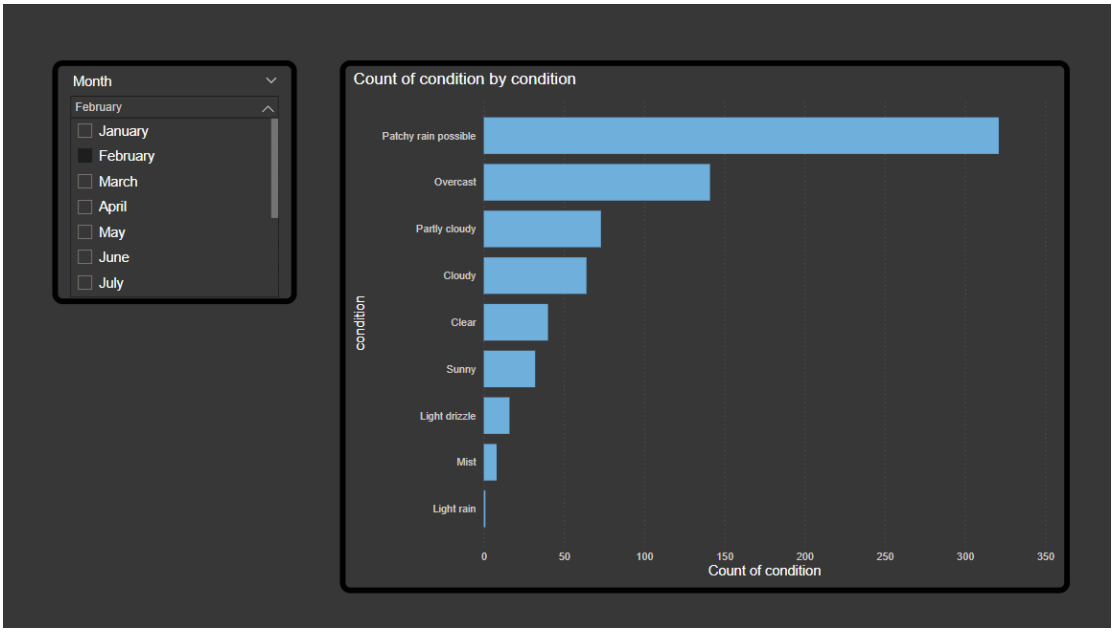
Ảnh: Tần suất của các weather conditions trong dữ liệu

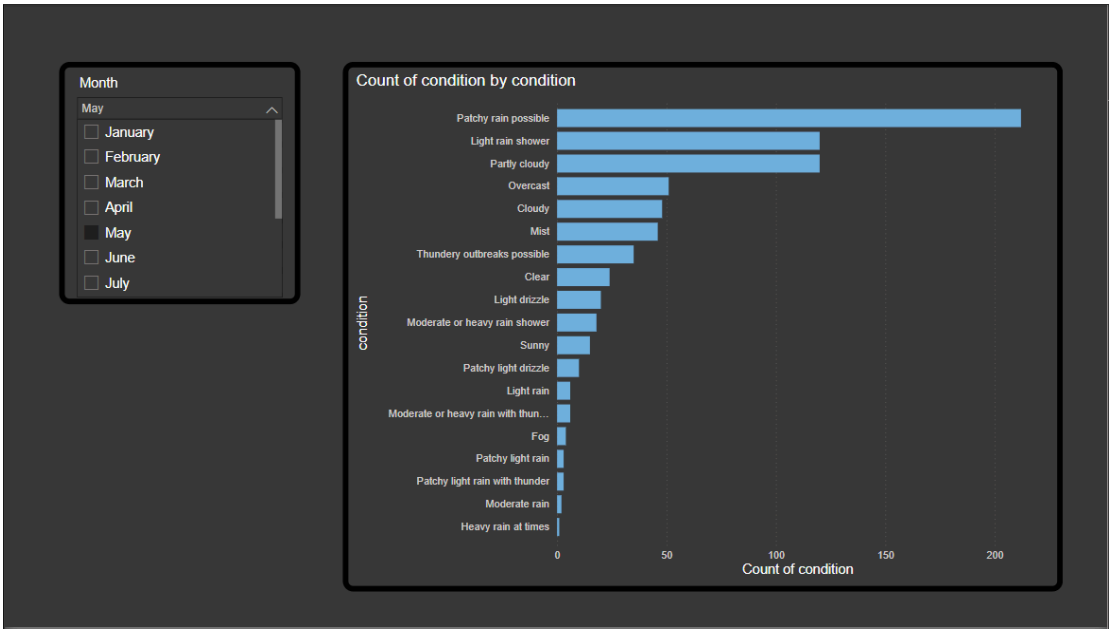
- Thời tiết chủ yếu ôn hoà với các điều kiện nhẹ như Windy, Patchy rain (có mưa rải rác), Partly cloudy (trời có mây), Overcast (âm u) và Clear(trời quang) chiếm đa số.
- Các điều kiện cực đoan như Heavy rain (mưa lớn) hay thunderstorms(sấm sét) xảy ra ít, cho thấy khu vực có thời tiết tương đối ổn định, không thường xuyên xảy ra hiện tượng khắc nghiệt.



Ảnh: Tần suất dữ liệu của các condition

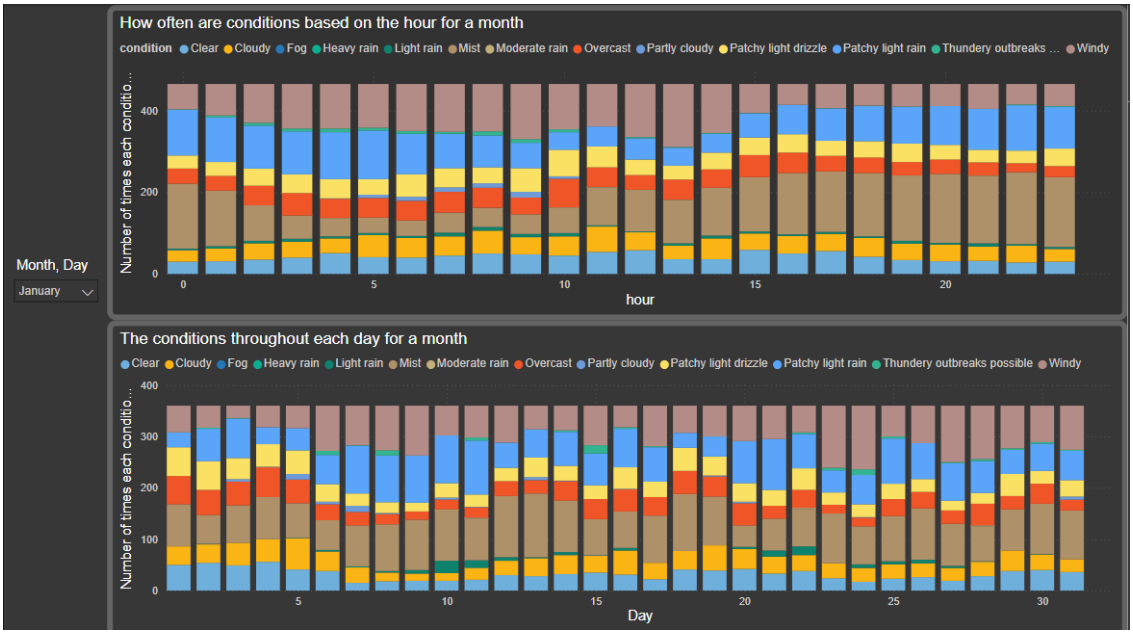
- Mỗi tình trạng thời tiết đều có các đặc điểm dữ liệu khác nhau
- Mô hình phân tích dữ liệu đặc trưng để phân loại các condition
- Ví dụ ở đây với condition là cloudy, thấy trung bình temp là 20.27 C, tb gió là 6.03, không có precipitation nào





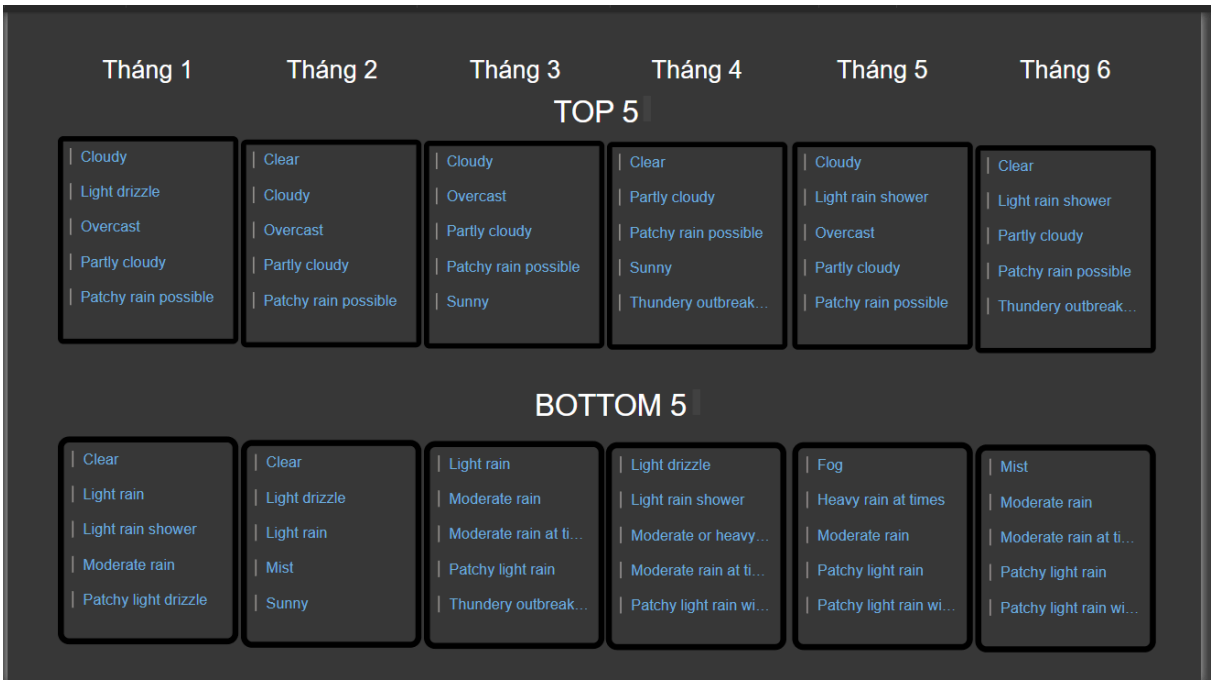
Ảnh: Số lần các weather conditions diễn ra trong 1 tháng cụ thể

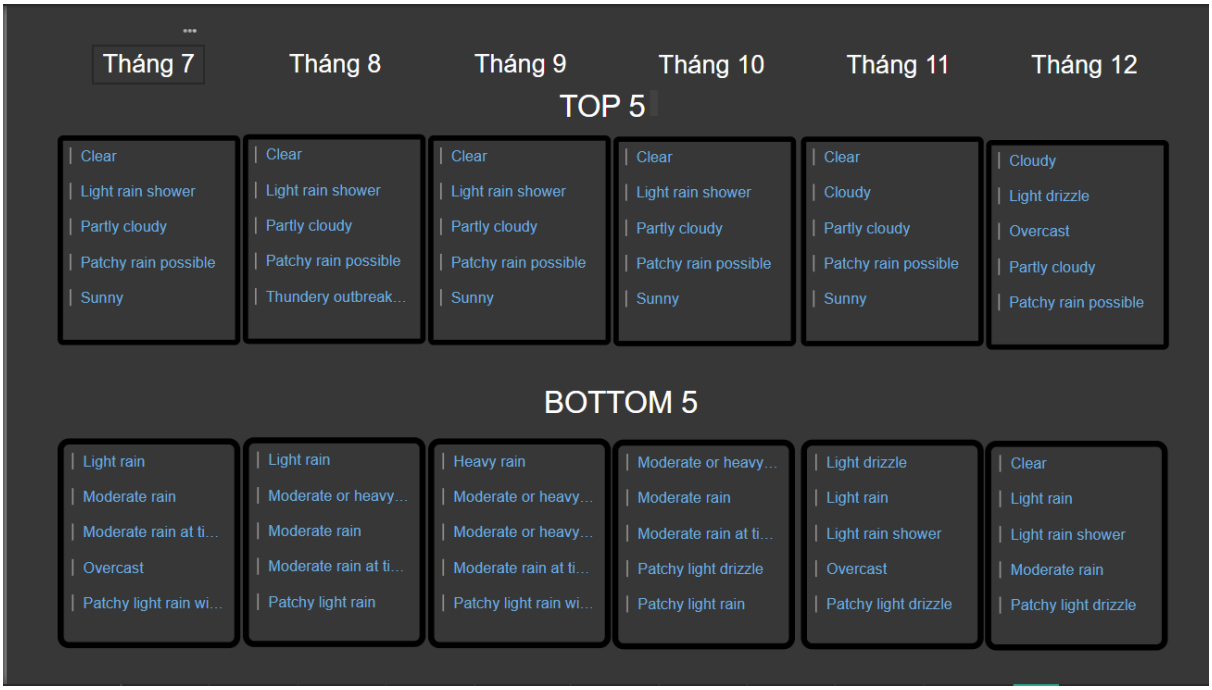
- Tùy vào từng tháng các weather conditions cụ thể có thể xảy ra thường xuyên hoặc ít thường xuyên hơn, ví dụ như ở trên ta có thể thấy tháng 2 và tháng 10 mưa nhiều



Hình trên mô tả tần suất của các tình trạng thời tiết theo giờ và ngày trong 1 tháng cụ thể

- Ta có thể thấy phân bố conditions của mỗi ngày đều khác nhau, và các conditions thường có khung giờ cụ thể xuất hiện





Ảnh: top và bottom 5 của tần suất các conditions xuất hiện trong từng tháng

- Ta có thể thấy từ tháng 5 đến tháng 10 là ‘light rain shower’ xuất hiện nhiều
- Clear và cloudy là 2 conditions xuất hiện thường xuyên nhất, tháng nào cũng top 3 (trừ tháng 12 và tháng 1).

IV. Triển khai mô hình

1. **Huấn luyện mô hình dự đoán nhiệt độ**:

1.1 Huấn luyện mô hình dự đoán nhiệt độ chỉ sử dụng thông tin về thời gian

- Mục tiêu của bài toán là xây dựng mô hình dự đoán nhiệt độ (°C) với 1 bước thời gian, do mỗi mẫu cách nhau 1 giờ.
- **Nguồn dữ liệu**: Dữ liệu thời tiết bao gồm các thông tin:
    - Nhiệt độ hiện tại (temp_c), độ ẩm (humidity), tốc độ gió (wind_mph), lượng mưa (precip_mm), và điều kiện thời tiết (condition) .
    - Các thuộc tính thời gian như giờ, ngày trong tuần, ngày trong tháng, tháng, quý, và tuần trong năm.
- Mô hình đầu tiên sẽ chỉ sử dụng thông tin về mặt thời gian để dự đoán.

1.2 Tiền xử lý dữ liệu

a. Chuyển đổi cột thời gian (datetime) thành các đặc trưng mới:
   - hour: Giờ trong ngày.
   - week: Tuần trong năm.
   - month: Tháng trong năm.
   - quarter: Quý trong năm.
   - day_of_week: Ngày trong tuần.
   - day_of_month: Ngày trong tháng.
   - day_of_year: Ngày trong năm.

b. Chuẩn hóa các đặc trưng thời gian:

$$\text{Normalized Value} = \frac{\text{Value} - \text{Min}}{\text{Max} - \text{Min}}$$

1.3 Phân chia dữ liệu

- **Dữ liệu huấn luyện**: Các mẫu trước thời điểm 2022-10-01 12:00:00.
- **Dữ liệu kiểm tra**: Các mẫu từ thời điểm 2022-10-01 12:00:00 trở đi.

1.4. Mô hình sử dụng

1.4.1 RandomForestRegressor:

Một mô hình cây quyết định ngẫu nhiên phù hợp cho bài toán hồi quy.

a, Huấn luyện mô hình

- Sử dụng Grid Search với các tham số:
  - numTrees: 10, 20, 50, 100.
  - maxDepth: 5, 7, 10.
- Áp dụng Cross-Validation với 3 lần gấp (3-fold) để tìm tham số tối ưu.

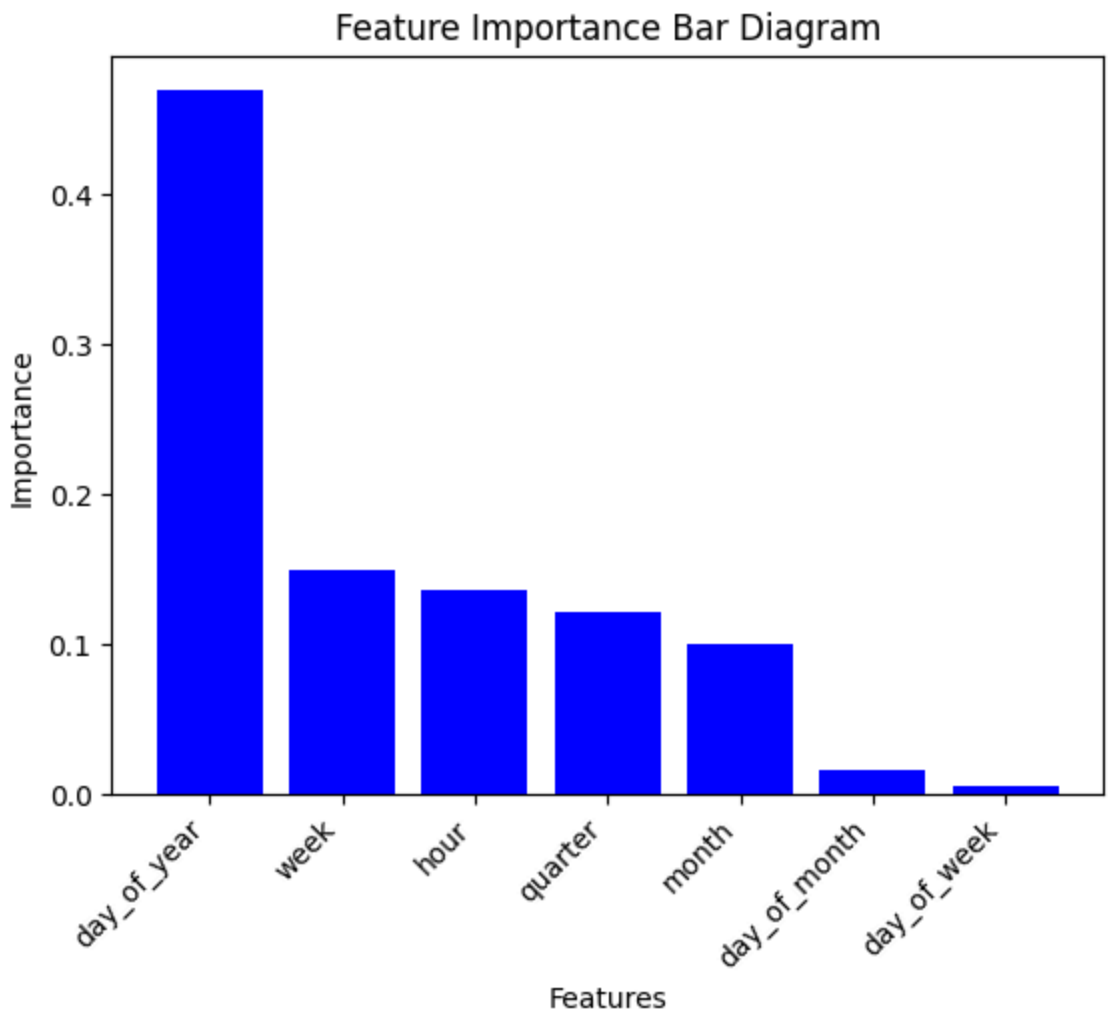
b. Hiệu suất mô hình

Kết quả đánh giá trên dữ liệu kiểm tra:

Chỉ số	Giá trị
Root Mean Squared Error (RMSE)	2.98
Mean Squared Error (MSE)	8.88
Mean Absolute Error (MAE)	2.4

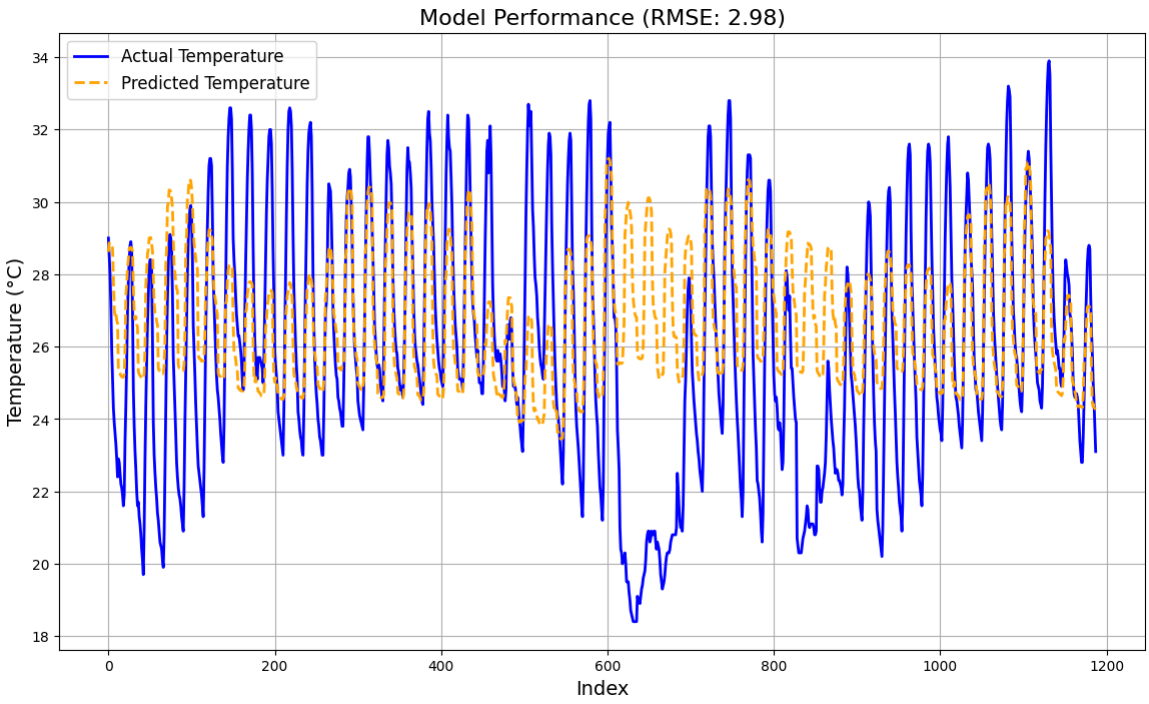
c. Độ quan trọng của các đặc trưng

Các đặc trưng quan trọng nhất trong dự đoán:


Feature Importance Bar Diagram
Importance
0.0
0.1
0.2
0.3
0.4
day_of_year
week
hour
quarter
month
day_of_month
day_of_week
Features

d. Đồ thị kết quả

Biểu đồ dự đoán so với giá trị thực:



- Mô hình dự đoán có số sai lệch không nhỏ. Do chỉ sử dụng đặc trưng về thời gian
- Các đặc trưng như weak và day_of_year đóng vai trò quan trọng nhất trong việc dự đoán nhiệt độ.

Để cải thiện khả năng dự đoán :

Huấn luyện sử dụng mọi thông tin đã có kể cả thông tin của bước thời gian trước một mô hình dự đoán nhiệt độ môi trường dựa trên dữ liệu thời tiết lịch sử bao gồm: nhiệt độ trước đó, độ ẩm, tốc độ gió, lượng mưa và các điều kiện thời tiết. Sử dụng hai mô hình chính:

- **Random Forest Regression**

- **Gradient Boosted Tree Regression**

Tiền xử lý dữ liệu:

1. **Mã hóa điều kiện thời tiết**: Các điều kiện thời tiết dạng chuỗi được ánh xạ sang giá trị số thông qua một bảng ánh xạ.
2. **Tạo đặc trưng thời gian**: Sử dụng các bước trễ (lag) để thêm các giá trị của nhiệt độ, độ ẩm, tốc độ gió và lượng mưa của 4 bước thời gian trước đó.
3. **Tích hợp các đặc trưng thời gian**: Bao gồm giờ, ngày, tháng, quý, và tuần.
4. **Loại bỏ các giá trị khuyết thiếu**: Sử dụng phương pháp dropna để bỏ các hàng không đủ dữ liệu.

1.4.2 Phương pháp

Sử dụng hai mô hình học máy:

1. **Random Forest Regression**:

   Sử dụng grid search với

   numTrees:10 - 100

   maxDepth: 5 - 15

   Tìm ra bộ tham số tối ưu là 70 cây (numTrees=70), độ sâu tối đa là 5 (maxDepth=5)

2. **Gradient Boosted Tree Regression**:

   Sử dụng grid search với

   maxIter: 10 - 50

   maxDepth: 5 - 15

   Ra được bộ tham số tối ưu là 50 vòng lặp (maxIter=50) và độ sâu tối đa là 5 (maxDepth=5).

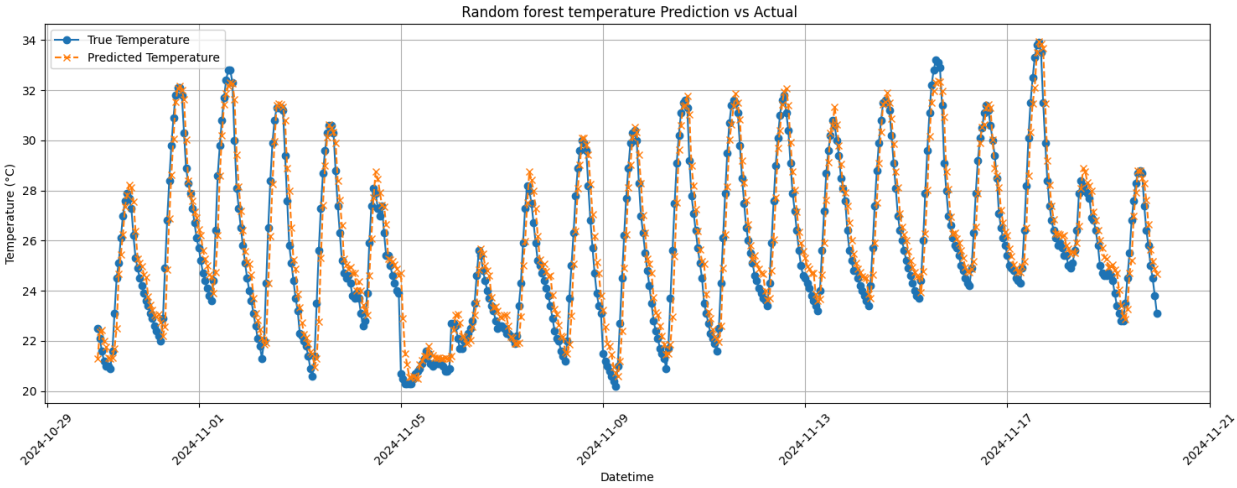
1.4.3 Đánh giá mô hình

Tiêu chí đánh giá **Root Mean Squared Error (RMSE):** Đánh giá độ chính xác của các mô hình dự đoán, thể hiện mức độ sai lệch trung bình của các giá trị dự đoán so với giá trị thực tế.

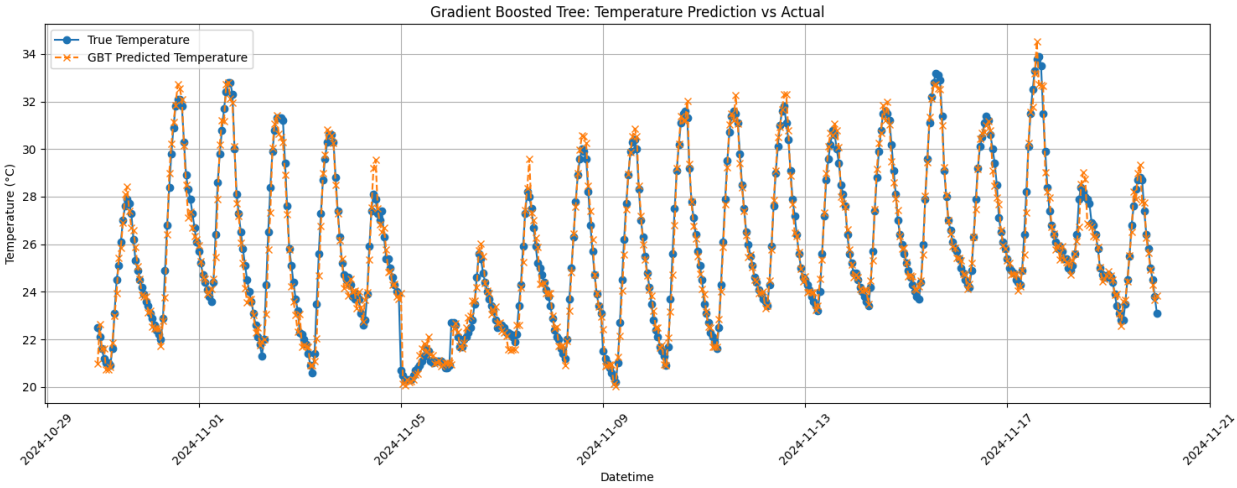
Công thức:

$$\text{RMSE} = \sqrt{\sum \frac{(y_{pred} - y_{ref})^2}{N}}$$

- Random Forest: RMSE = 0.89



- Gradient Boosted Tree: RMSE = 0.63



=> Gradient Boosted Tree cho kết quả tốt hơn Random Forest với RMSE nhỏ hơn, cho thấy dự đoán chính xác hơn, tốt hơn trong việc giảm bias bằng cách kết hợp các mô hình nhỏ.

1.4.4 Kết luận

- **Gradient Boosted Tree** là lựa chọn phù hợp hơn để dự đoán nhiệt độ trong bài toán này.
- Kết quả mô hình có thể được ứng dụng để dự báo thời tiết trong tương lai.

2. Huấn luyện mô hình dự đoán tình trạng thời tiết (condition)

2.1 Mục tiêu

Dự đoán điều kiện thời tiết (ví dụ: Nắng, Mưa, Sương mù, v.v.) trước 1 bước thời gian (1h do mỗi mẫu cách nhau 1h) dựa trên các dữ liệu lịch sử bao gồm điều kiện thời tiết, nhiệt độ, độ ẩm, tốc độ gió, lượng mưa và các đặc trưng thời gian (giờ, ngày, tháng).

2.2 Dữ liệu

Dữ liệu ban đầu bao gồm các cột sau:

- **Thời gian**: datetime
- **Nhiệt độ**: temp_c
- **Độ ẩm**: humidity
- **Tốc độ gió**: wind_mph
- **Lượng mưa**: precip_mm
- **Điều kiện thời tiết**: condition (được mã hóa thành condition_index)

2.3 Phương pháp

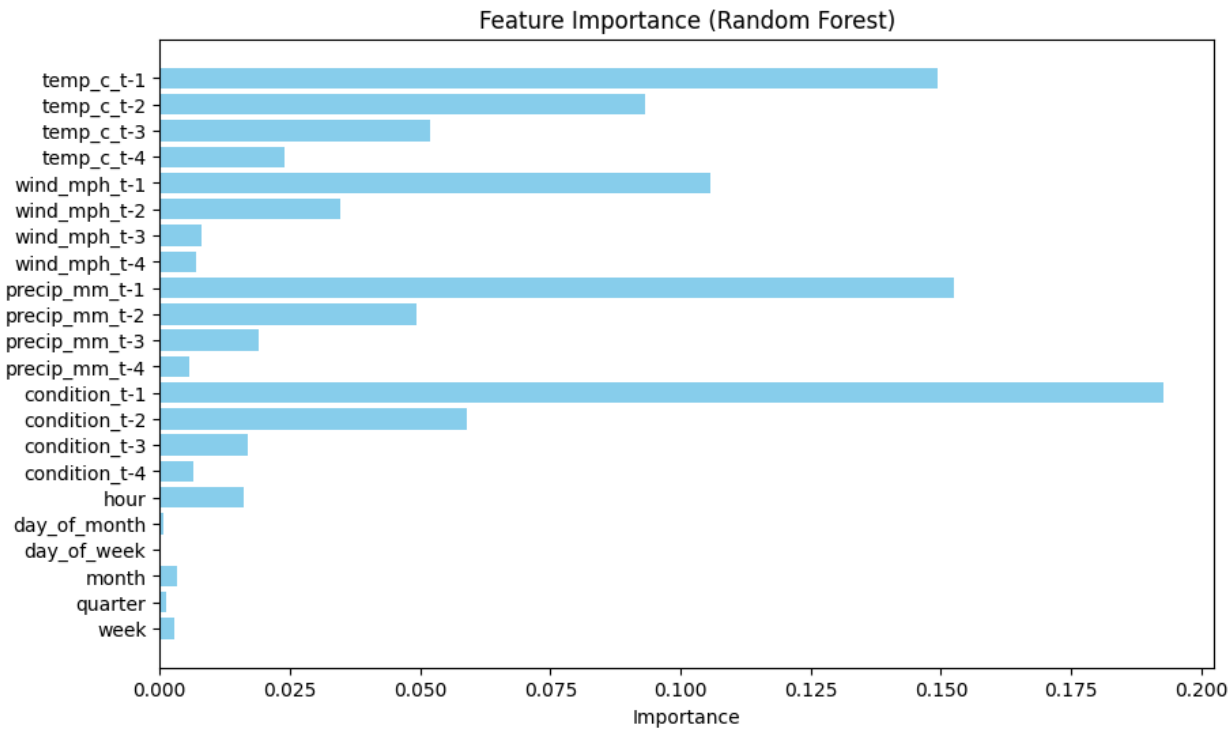
1. **Mã hóa nhãn:**
    - Sử dụng từ điển condition_to_index để chuyển đổi các giá trị điều kiện thời tiết (chuỗi) thành chỉ số số nguyên.
    - Sử dụng UDF (User Defined Function) trong PySpark để thực hiện quá trình này.
2. **Tạo đặc trưng:**
    - Tạo thêm các đặc trưng sử dụng dữ liệu của các bước thời gian trước đó (1 đến 4 bước lùi).

    - Thêm các đặc trưng thời gian như: hour, day_of_month, day_of_week, month, quarter, và week.
3. **Chia dữ liệu:**
    - Dữ liệu được chia thành hai phần: tập huấn luyện (trước ngày 01/01/2022) và tập kiểm tra (sau ngày 01/01/2022).
4. **Xây dựng mô hình:**
    - Sử dụng 3 mô hình:
        - **Random Forest Classifier**
        - **Logistic Regression**
        - **Naive Bayes**
5. **Đánh giá hiệu năng:**
    - Sử dụng MulticlassClassificationEvaluator với tiêu chí **độ chính xác (accuracy)** để đánh giá hiệu năng các mô hình trên tập kiểm tra.

2.4 Kết quả

- **Độ chính xác của mô hình trên tập test:**
    - Logistic Regression: 37.5%
    - Naive Bayes: 20%
    - Random Forest: 61,7%

Feature Importance (Random Forest): Các đặc trưng quan trọng nhất đối với mô hình Random Forest bao gồm:



2.5 Kết luận

- Random Forest cho kết quả tốt nhất trong số các mô hình được thử nghiệm.
- Logistic Regression và Naive Bayes có độ chính xác thấp hơn, do bản chất của bài toán không phù hợp với các giả định tuyến tính.
- Logistic Regression không phù hợp với bài toán phân loại đa lớp có nhiều lớp và tương tác phức tạp giữa các đặc trưng.
- Naive Bayes: Giả định đặc trưng độc lập với nhau, điều này không đúng với trong thực tế.

3. Thử nghiệm thực tế

3.1 Triển khai Pipeline Kafka và PySpark

1. Kafka:
    - Cấu hình Kafka Producer để liên tục thu thập dữ liệu thời tiết từ API .
    - Cấu hình Kafka Topic làm kênh trung gian cho dữ liệu thô..
2. PySpark:
    - Tạo Spark Session và Spark Streaming để nhận dữ liệu từ Kafka, xử lý và đưa vào delta lake.

    - Tiến hành tiền xử lý dữ liệu như chuẩn hóa các đặc trưng và xóa dữ liệu khuyết thiếu.
    - Đánh giá kết quả trên dữ liệu streaming và batch.

3.2 Kết quả thực tế

- Triển khai Kafka và PySpark hoạt động ổn định, nhận dữ liệu từ API thời tiết theo thời gian thực.
- Tốc độ xử lý: Kafka Producer ghi dữ liệu với tốc độ trung bình 10.000 bản ghi/phút,dữ liệu được toàn vẹn trong delta lake .
- Đánh giá hiệu suất thành công: Streaming latency < 2 giây.

3.3 Hạn chế trong thực tế

- Không đủ dữ liệu trong một số khung thời gian.
- Cần có kiểm soát tốt hơn khi Kafka Producer gửi số lượng dữ liệu lớn liên tục.
- Yêu cầu hạ tầng cao để đáp ứng dữ liệu lưu lượng cao.

4. Kết luận và đề xuất

4.1 Kết luận

- Sử dụng Apache Kafka,Delta lake, PySpark và Power BI để phân tích và xử lý dữ liệu thời gian thực là phương pháp hiệu quả.
- Gradient Boosted Tree Regression cho hiệu quả dự đoán nhiệt độ tốt nhất, trong khi sử dụng Random Forest để phân loại tình trạng thời tiết cho kết quả hơi thấp do thời tiết có yếu tố rất phức tạp.

4.2 Hạn chế của dự án

- Dữ liệu dành cho điều kiện thời tiết hiện tại chưa đủ phong phú, cần mở rộng nguồn thu thập trên nhiều địa điểm thế giới.
- Yêu cầu hạ tầng tính toán cao để xử lý dữ liệu lưu lượng lớn trong thời gian thực.

4.3 Đề xuất hướng phát triển tiếp theo

- Mở rộng dữ liệu: Thu thập dữ liệu thời tiết từ nhiều khu vực hơn, bao gồm dữ liệu phong phú hơn .
- Nâng cấp hạ tầng: Sử dụng hạ tầng Spark đám mây (để dàn trản lưu lượng lớn).

- Tích hợp Deep Learning: Đề xuất sử dụng các mô hình AI như LSTM hoặc Transformer cho bài toán chuỗi thời gian.

Tài liệu tham khảo

1. Zaharia, M., & Wendell, P. (2016). *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media.
2. Kreps, J. (2014). *I Heart Logs: Event Data, Stream Processing, and Data Integration*. O'Reilly Media.
3. Apache Kafka Documentation. (n.d.). Retrieved from https://kafka.apache.org/documentation/
4. Apache Spark Documentation. (n.d.). Retrieved from https://spark.apache.org/docs/latest/
5. Weather Data API Documentation. (n.d.). Retrieved from 🌤️ Free Open-Source Weather API | Open-Meteo.com

Công việc của từng thành viên

Họ và tên	Công việc
Triệu Vũ Hoàn	- Cài đặt và triển khai hệ thống docker, kafka, delta lake,spark . - Thu thập dữ liệu kết hợp spark streaming, - Xây dựng mô hình SparkMLib. - Viết báo cáo
Hoàng Đình Hưng	- Tìm hiểu đề tài. - Sử dụng power BI phân tích dữ liệu tự động. - Viết báo cáo

Lê Thành Đạt	- Tìm hiểu đề tài. - Tìm hiểu kiến trúc dự án. - Thu thập và tiền xử lý dữ liệu với spark. - Viết báo cáo. - Làm slide
Hoàng Bùi Tuấn Anh	- Tìm hiểu đề tài. - Tìm hiểu kiến trúc dự án. - Xây dựng mô hình với sparkMlib. - Viết báo cáo. - Làm slide.
Lê Hoàng Anh	- Tìm hiểu đề tài. - Tiền xử lý dữ liệu với spark - Viết báo cáo. - Làm slide.