TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

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

ĐỀ TÀI: SMARTPHONE PRICE PREDICTION IN BIG DATA ENVIRONMENT

Nhóm sinh viên thực hiện:

1. Nguyễn Minh Hiếu
2. Nguyễn Tiến Khôi
3. Tống Duy Tân
4. Phạm Quốc Tuấn
5. Quản Xuân Sơn

Giảng viên hướng dẫn: TS. Trần Hồng Việt

CN. Đỗ Thu Uyên

MỞ ĐẦU

Trong thời đại cách mạng công nghiệp 4.0, dữ liệu lớn (Big Data) đã và đang trở thành một nguồn tài nguyên quan trọng giúp các tổ chức, doanh nghiệp và cá nhân đưa ra các quyết định chiến lược. Việc khai thác, xử lý và phân tích dữ liệu lớn không chỉ đòi hỏi những công cụ và kỹ thuật tiên tiến mà còn yêu cầu sự hiểu biết sâu sắc về các mô hình và thuật toán liên quan.

Dự án *"Smartphone Price Prediction in Big Data Environment"* được nhóm chúng em thực hiện nhằm giải quyết bài toán dự đoán giá smartphone dựa trên các thông tin đầu vào, như thông số kỹ thuật, thương hiệu và các yếu tố liên quan khác. Trong môi trường dữ liệu lớn, việc xử lý khối lượng dữ liệu khổng lồ, đa dạng và có tốc độ tăng trưởng nhanh chóng đặt ra những thách thức nhưng đồng thời cũng mang lại cơ hội để cải thiện độ chính xác và hiệu quả của các mô hình dự đoán.

Báo cáo này sẽ trình bày chi tiết các bước thực hiện dự án, bao gồm thu thập và tiền xử lý dữ liệu, thiết kế và triển khai mô hình dự đoán, cũng như phân tích và đánh giá kết quả. Qua đó, nhóm thực hiện hy vọng mang lại những góc nhìn thiết thực về việc ứng dụng công nghệ dữ liệu lớn trong lĩnh vực thương mại và quản trị. Cụ thể, báo cáo gồm 4 chương:

Chương 1: Giới thiệu chung

Chương 2: Thuật toán XGBoost trong bài toán dự đoán

Chương 3: Thực nghiệm

Chương 4: Kết luận và hướng phát triển

PHÂN CÔNG NHIỆM VỤ

Thành viên	Nhiệm vụ
Nguyễn Tiến Khôi + Nguyễn Quốc Tuấn	thu thập, xử lý dữ liệu và xây dựng mô hình dự đoán
Quản Xuân Sơn + Tống Duy Tân	xây dựng quy trình xử lý dữ liệu theo kiến trúc Lambda
Nguyễn Minh Hiếu	tìm hiểu về các nền tảng công nghệ dữ liệu lớn, viết báo cáo

MỤC LỤC

Chương 1. GIỚI THIỆU CHUNG

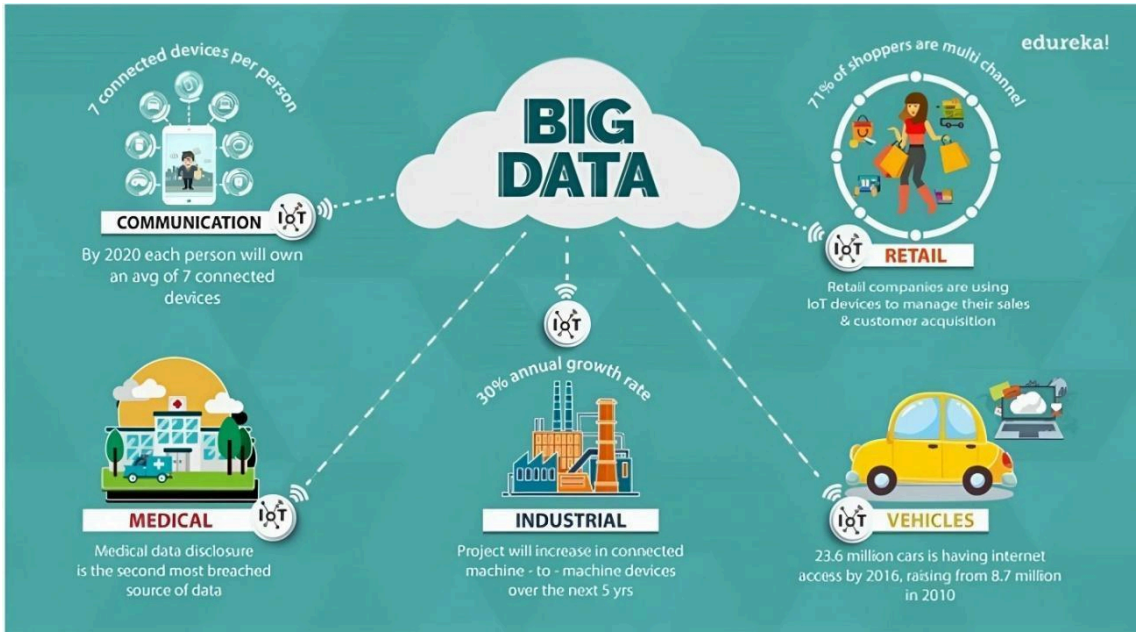
1.1 Tổng quan về dữ liệu lớn

Theo wikipedia: Dữ liệu lớn (Big data) là một thuật ngữ chỉ bộ dữ liệu lớn hoặc phức tạp mà các phương pháp truyền thống không đủ các ứng dụng để xử lý dữ liệu này

Theo Gartner: Dữ liệu lớn là những nguồn thông tin có đặc điểm chung khối lượng lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau, do đó muốn khai thác được đòi hỏi phải có hình thức xử lý mới để đưa ra quyết định, khám phá và tối ưu hóa quy trình.

Dữ liệu lớn đến từ đâu?

1. Dữ liệu hành chính (phát sinh từ chương trình của một tổ chức, có thể là chính phủ hay phi chính phủ). Ví dụ: hồ sơ y tế điện tử ở bệnh viện, hồ sơ bảo hiểm, hồ sơ ngân hàng...;
2. Dữ liệu từ hoạt động thương mại (phát sinh từ các giao dịch giữa hai thực thể) Ví dụ: các giao dịch thẻ tín dụng, giao dịch trên mạng, bao gồm cả các giao dịch từ các thiết bị di động
3. Dữ liệu từ các thiết bị cảm biến như thiết bị chụp hình ảnh vệ tinh, cảm biến đường, cảm biến khí hậu;

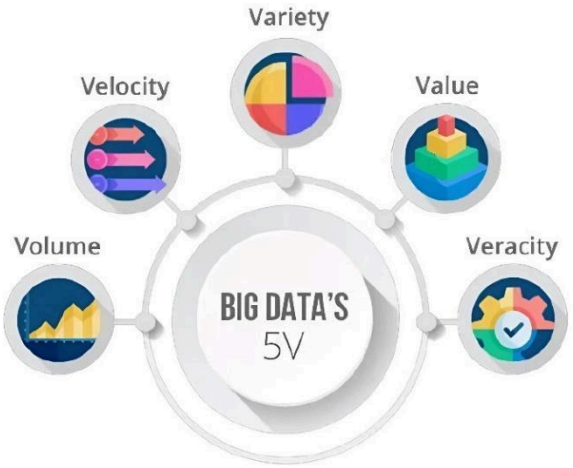


Hình 1.1 Minh họa về nguồn gốc dữ liệu lớn

4. Dữ liệu từ các thiết bị theo dõi. Ví dụ theo dõi dữ liệu từ điện thoại di động, GPS;
5. Dữ liệu từ các hành vi Ví dụ như tìm kiếm trực tuyến (tìm kiếm sản phẩm, dịch vụ hay thông tin khác), đọc các trang mạng trực tuyến...;
6. Dữ liệu từ các thông tin về ý kiến, quan điểm của các cá nhân, tổ chức, trên các phương tiện thông tin xã hội

Đặc trưng cơ bản của dữ liệu lớn

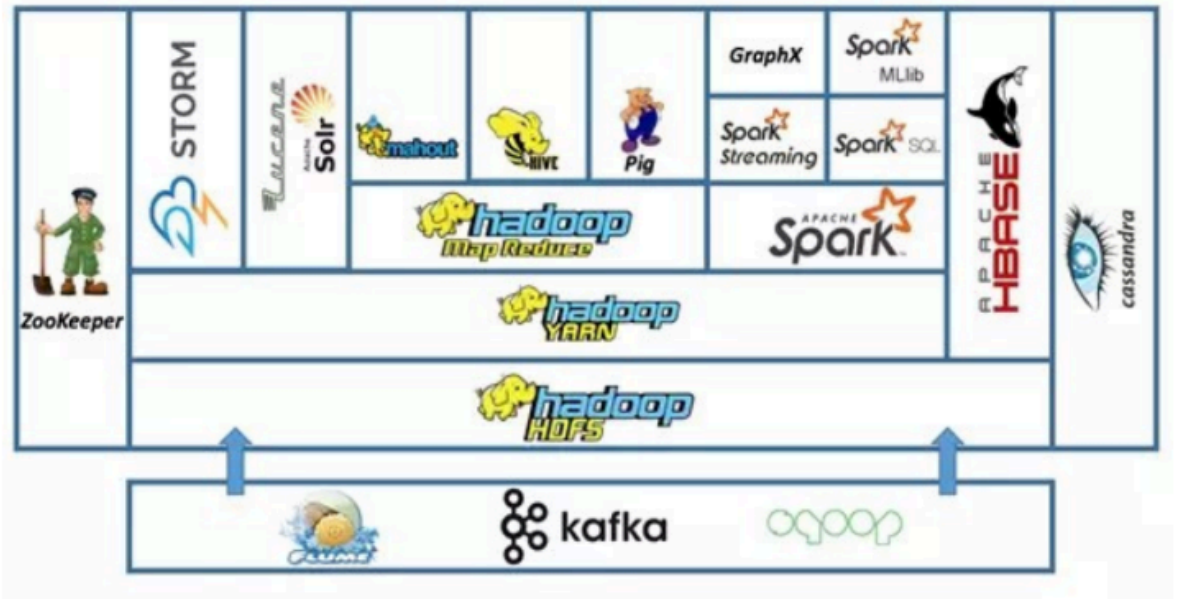
- Volume: Khối lượng dữ liệu: Là đặc điểm tiêu biểu nhất, kích cỡ của big data ngày càng tăng, sử dụng công nghệ đám mây mới có khả năng lưu trữ.
- Velocity: Tốc độ: Hiểu theo 2 hướng: Khối lượng dữ liệu gia tăng rất nhanh / Xử lý dữ liệu nhanh ở mức thời gian thực (real-time)
- Đa dạng (Variety): Đối với dữ liệu truyền thống chúng ta hay nói đến dữ liệu có cấu trúc. Ngày nay hơn 80% dữ liệu được sinh ra là phi cấu trúc (tài liệu, blog, hình ảnh, video). Big Data cho phép liên kết và phân tích nhiều dạng dữ liệu khác nhau
- Veracity: Độ tin cậy/Chính xác: Bài toán phân tích và loại bỏ dữ liệu thiếu chính xác và nhiễu đang là tính chất quan trọng của bigdata.
- Value: Giá trị: Giá trị là đặc điểm quan trọng nhất của dữ liệu lớn. Khi bắt đầu triển khai xây dựng dữ liệu lớn thì việc đầu tiên chúng ta cần phải làm đó là xác định được giá trị của thông tin mang lại như thế nào, khi đó chúng ta mới có quyết định nên triển khai dữ liệu lớn hay không



Hình 1.2 Đặc trưng 5V của dữ liệu lớn

1.2 Tổng quan về Hadoop

Hadoop là một công nghệ phân tán và mã nguồn mở được sử dụng phổ biến để xử lý và lưu trữ khối dữ liệu lớn trên các cụm máy tính phân tán, được thiết kế để xử lý và lưu trữ dữ liệu lớn một cách hiệu quả.



Hình 1.3 Hệ sinh thái Hadoop

- Hadoop sử dụng mô hình phân tán để lưu trữ và xử lý dữ liệu trên các máy tính thông thường.
- Hadoop phân chia dữ liệu thành các khối và lưu chúng trên nhiều máy tính, giúp tăng khả năng mở rộng, độ tin cậy và hiệu năng.
- Hadoop phân tán dữ liệu trên nhiều nút máy tính và xử lý nó song song trên các nút, giúp giảm thời gian xử lý và tăng hiệu suất

Hadoop được xây dựng dựa trên ba phần chính là Hadoop Distributes FileSystem (HDFS), YARN và MapReduce. YARN (Yet Another Resource Negotiator) là một thành phần quan trọng trong hệ sinh thái Hadoop, chịu trách nhiệm quản lý và phân phối tài nguyên cho các ứng dụng chạy trên Hadoop Cluster.

1.3 Tổng quan về các công nghệ sử dụng

Kafka

Apache Kafka là một kho dữ liệu phân tán được tối ưu hóa để thu nạp và xử lý dữ liệu truyền phát theo thời gian thực. Dữ liệu truyền phát là dữ liệu được tạo ra liên tục

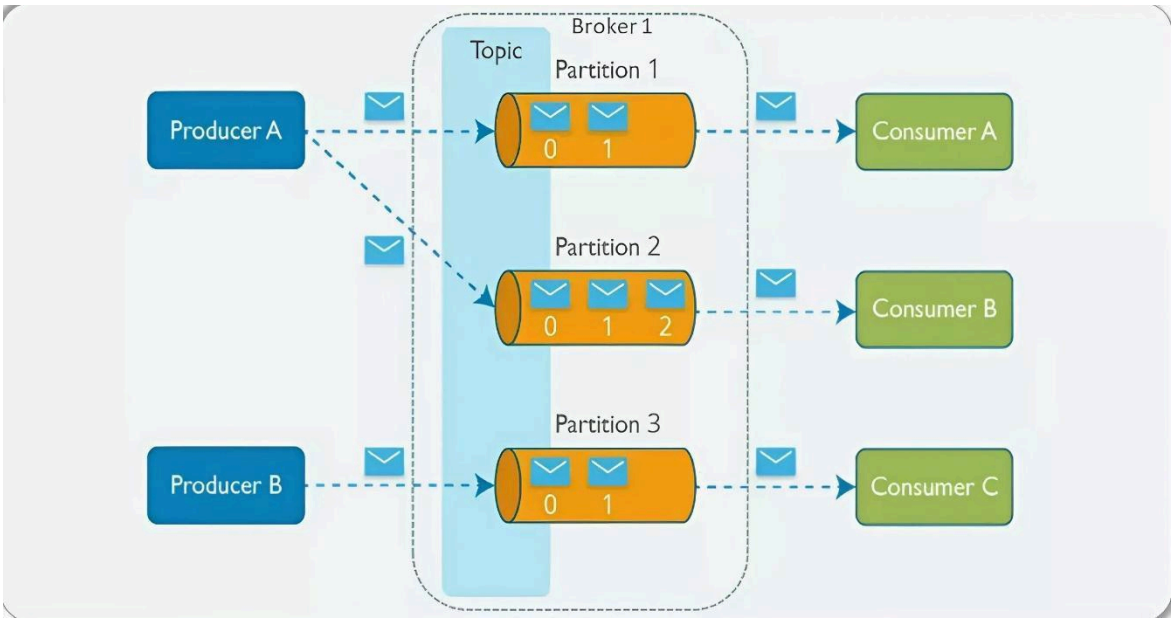
từ hàng nghìn nguồn dữ liệu khác nhau, các nguồn này thường gửi các bản ghi dữ liệu đồng thời. Nền tảng truyền phát cần phải xử lý luồng dữ liệu liên tục này và xử lý dữ liệu theo trình tự và tăng dần.

Kafka cung cấp ba chức năng chính cho người dùng:

- Xuất bản và đăng ký các luồng bản ghi
- Lưu trữ hiệu quả các luồng bản ghi theo thứ tự tạo bản ghi
- Xử lý các luồng bản ghi trong thời gian thực

Kafka chủ yếu được dùng để xây dựng các quy trình dữ liệu truyền phát trong thời gian thực và các ứng dụng thích ứng với luồng dữ liệu đó. Kafka kết hợp nhắn tin, lưu trữ và xử lý luồng nhằm hỗ trợ hoạt động lưu trữ, phân tích cả dữ liệu lịch sử lẫn dữ liệu trong thời gian thực.

Kafka kết hợp hai mô hình nhắn tin, hàng đợi và xuất bản-đăng ký, để cung cấp những lợi ích chính của mỗi mô hình cho người tiêu dùng. Hàng đợi cho phép xử lý dữ liệu được phân phối trên nhiều phiên bản người tiêu dùng, giúp dữ liệu có khả năng mở rộng cao. Tuy nhiên, hàng đợi truyền thống không dành cho nhiều bên đăng ký nhận. Cách tiếp cận xuất bản-đăng ký dành cho nhiều bên đăng ký nhận, nhưng vì mọi tin nhắn đi đến mọi bên đăng ký nhận nên không thể được sử dụng để phân phối công việc trên nhiều quy trình lao động. Kafka sử dụng mô hình bản ghi phân vùng để ghép hai giải pháp này lại với nhau. Bản ghi là một chuỗi các hồ sơ có thứ tự và các bản ghi này được chia thành các phân đoạn, hoặc phân vùng, tương ứng với các bên đăng ký nhận khác nhau. Điều này có nghĩa là có thể có nhiều bên đăng ký nhận cho cùng một chủ đề và mỗi bên được gán một phân vùng để cho phép khả năng điều chỉnh quy mô cao hơn. Cuối cùng, mô hình của Kafka cung cấp khả năng phát lại, cho phép nhiều ứng dụng độc lập đọc từ các luồng dữ liệu hoạt động độc lập với tốc độ riêng của luồng dữ liệu



Hình 1.4 Kiến trúc minh họa Kafka

Apache Hbase

Apache HBase là kho dữ liệu lớn phân tán, NoSQL, nguồn mở. cho phép truy cập ngẫu nhiên, hoàn toàn nhất quán, theo thời gian thực vào hàng petabyte dữ liệu. HBase rất hiệu quả trong việc xử lý các tập dữ liệu lớn, thưa thớt. HBase tích hợp liền mạch với Apache Hadoop và hệ sinh thái Hadoop, đồng thời hoạt động dựa trên Hệ thống tệp phân tán Hadoop (HDFS) hoặc Amazon S3 sử dụng hệ thống tệp Amazon Elastic MapReduce (EMR) hay EMRFS. HBase đóng vai trò là đầu vào và đầu ra trực tiếp đối với khung Apache MapReduce cho Hadoop và hoạt động với Apache Phoenix để cho phép truy vấn tương tự SQL trên các bảng HBase.

HBase là cơ sở dữ liệu phi quan hệ, hướng cột. Như vậy có nghĩa là dữ liệu được lưu trữ trong các cột riêng lẻ và được lập chỉ mục bằng một khóa hàng đơn nhất. Kiến trúc này cho phép truy xuất nhanh chóng các hàng và cột riêng lẻ, cũng như quét hiệu quả qua các cột riêng lẻ trong một bảng. Cả dữ liệu và yêu cầu đều được phân phối trên tất cả các máy chủ trong một cụm HBase. Nhờ đó, bạn có thể truy vấn kết quả trên hàng petabyte dữ liệu trong vài mili giây. HBase phát huy hiệu quả tốt nhất khi lưu trữ dữ liệu phi quan hệ, được truy cập thông qua API HBase. Apache Phoenix thường được dùng làm lớp SQL trên HBase nhằm giúp bạn có thể sử dụng cú pháp SQL quen thuộc để chèn, xóa và truy vấn dữ liệu lưu trữ trong HBase.

Ưu điểm của Hbase

- Có khả năng mở rộng: HBase được thiết kế để xử lý việc điều chỉnh quy mô trên hàng nghìn máy chủ và quản lý quyền truy cập vào hàng petabyte dữ liệu. Với độ linh hoạt của Amazon EC2 và khả năng điều chỉnh quy mô của Amazon S3, HBase có thể xử lý hoạt động truy cập trực tuyến vào các tập dữ liệu cực lớn.
- Nhanh: HBase cho phép truy cập đọc và ghi ngẫu nhiên với độ trễ thấp vào hàng petabyte dữ liệu bằng cách phân phối yêu cầu từ các ứng dụng trên một cụm máy chủ. Mỗi máy chủ đều có quyền truy cập vào dữ liệu trong HDFS và S3, cũng như phân phối các yêu cầu đọc và ghi trong vài mili giây.
- Có khả năng chịu lỗi: HBase tách dữ liệu lưu trữ trong các bảng trên nhiều máy chủ trong cụm và được xây dựng để chịu được các lỗi máy chủ riêng lẻ. Vì dữ liệu được lưu trữ trên HDFS hoặc S3 nên HBase sẽ tự động chọn máy chủ hoạt động tốt để lưu trữ dữ liệu từng được máy chủ bị lỗi phân phối. Dữ liệu này sẽ tự động chuyển sang trạng thái trực tuyến.

Apache Spark

Apache Spark là một framework xử lý dữ liệu nguồn mở được thiết kế để xử lý các tác vụ tính toán lớn và phức tạp trên các tập dữ liệu lớn. Spark cung cấp một một giao diện để lập trình các cụm máy tính song song với khả năng chịu lỗi. Khả năng tính toán phân tán của Apache Spark làm cho nó phù hợp với dữ liệu lớn và máy học, đòi hỏi sức mạnh tính toán lớn để hoạt động trên kho dữ liệu lớn.

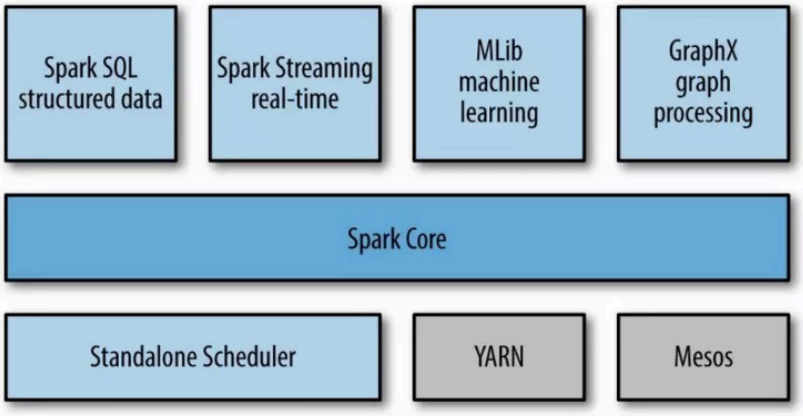
Đặc điểm của Apache Spark:

- Tốc độ: Spark tăng tốc quá trình thực thi ứng dụng trong cụm Hadoop lên tới 100 lần trong bộ nhớ và 10 lần trên đĩa. Điều này được thực hiện bằng cách giảm số lượng thao tác đọc/ghi đĩa. Dữ liệu xử lý trung gian được lưu trong bộ nhớ.

- Hỗ trợ nhiều ngôn ngữ: Spark có các API tích hợp trong Java, Scala và Python và hỗ trợ nhiều ngôn ngữ. Nhờ đó, bạn có thể viết các ứng dụng bằng nhiều ngôn ngữ khác nhau. Đối với truy vấn tương tác, Spark cung cấp 80 toán tử cấp cao.

- Phân tích nâng cao: Spark không chỉ hỗ trợ ‘Map’ và ‘reduce’ mà truy vấn SQL, truyền dữ liệu, máy học (machine learning) và thuật toán đồ thị cũng được hỗ trợ.

Các thành phần trong Apache Spark



Hình 1.5. Các thành phần của Apache Spark

- Spark Core: Cung cấp những chức năng cơ bản nhất của Spark như lập lịch cho các tác vụ, quản lý bộ nhớ, fault recovery, tương tác với các hệ thống lưu trữ... Spark Core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet) là tập hợp của các item được phân tán trên các node của cluster và có thể được xử lý song song.
- Spark SQL cho phép truy vấn dữ liệu cấu trúc qua các câu lệnh SQL. Spark SQL có thể thao tác với nhiều nguồn dữ liệu như Hive tables, Parquet, và JSON.
- Spark Streaming cung cấp API để dễ dàng xử lý dữ liệu stream,
- MLlib Cung cấp rất nhiều thuật toán của học máy như: classification, regression, clustering, collaborative filtering...
- GraphX là thư viện để xử lý đồ thị.

PostgreSQL

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, mạnh mẽ và đáng tin cậy. Nó hỗ trợ SQL (Structured Query Language) và được sử dụng rộng rãi trong các ứng dụng từ nhỏ đến lớn nhờ tính linh hoạt và hiệu suất cao.

PostgreSQL là một hệ quản trị cơ sở dữ liệu mạnh mẽ với khả năng hỗ trợ ACID, đảm bảo tính toàn vẹn dữ liệu trong các giao dịch. Nó cung cấp một loạt kiểu dữ liệu phong phú, bao gồm JSON, XML, Arrays và hỗ trợ cả kiểu dữ liệu do người dùng định nghĩa. PostgreSQL còn nổi bật với tính năng xử lý dữ liệu địa lý thông qua PostGIS, phù hợp cho các ứng dụng GIS. Ngoài ra, hệ thống giao dịch tiên tiến với các

chức năng như rollback, khóa đa cấp giúp quản lý dữ liệu hiệu quả trong môi trường phức tạp.

PostgreSQL có khả năng mở rộng vượt trội nhờ tích hợp các tiện ích mở rộng như TimescaleDB hay pg_cron. Nó hỗ trợ viết hàm bằng nhiều ngôn ngữ lập trình như Python, JavaScript, và PL/pgSQL, giúp tùy chỉnh linh hoạt theo nhu cầu. Với giấy phép mã nguồn mở, PostgreSQL được cộng đồng phát triển mạnh mẽ, mang lại các cải tiến liên tục. Đây là hệ quản trị lý tưởng cho cả ứng dụng nhỏ và các hệ thống lớn, từ web, di động đến phân tích dữ liệu lớn.

Apache Airflow

Apache Airflow là một công cụ mã nguồn mở được sử dụng để lập lịch, quản lý, và giám sát các quy trình xử lý dữ liệu. Nó được sử dụng rộng rãi trong các hệ thống xử lý dữ liệu lớn để tự động hóa các quy trình xử lý dữ liệu phức tạp

Airflow phát triển dựa trên AirBnb vào 10/2014 và quản lý các công việc dựa trên một đồ thị DAG (đồ thị có hướng và không có chu trình) trong đó mỗi node trong đồ thị là một task nhỏ

Lợi ích của Apache Airflow:

- Quản lý và lập lịch công việc: Cung cấp một giao diện đồ họa dễ sử dụng để quản lý và lập lịch các công việc.
- Tích hợp dễ dàng: Tích hợp tốt với nhiều công cụ và dịch vụ phổ biến như Hadoop, Spark, Hive, BigQuery
- Mở rộng linh hoạt: Cho phép mở rộng hệ thống một cách linh hoạt. Người dùng có thể thêm các công việc mới, mở rộng máy chủ và tăng cường khả năng xử lý dữ liệu một cách dễ dàng
- Xác thực và phân quyền: Airflow cung cấp tính năng xác thực và phân quyền mạnh mẽ. Người dùng có thể xác định quyền truy cập cho từng cá nhân và nhóm người, giúp bảo mật dữ liệu của người dùng
- Quản lý lỗi và theo dõi: Airflow cung cấp các công cụ để quản lý lỗi và theo dõi quá trình thực thi công việc. Người dùng có thể dễ dàng xem các báo cáo lỗi và theo dõi tiến trình của các công việc

Flask

Flask là một microweb framework được viết bằng Python . Nó được phân loại là microframework vì nó không yêu cầu các công cụ hoặc thư viện cụ thể. Nó không có lớp trừu tượng hóa cơ sở dữ liệu, xác thực biểu mẫu hoặc bất kỳ thành phần nào khác mà các thư viện của bên thứ ba hiện có cung cấp các chức năng chung. Tuy nhiên, Flask hỗ trợ các tiện ích mở rộng có thể thêm các tính năng ứng dụng như thể chúng được triển khai trong chính Flask. Các tiện ích mở rộng tồn tại cho các trình ánh xạ quan hệ đối tượng , xác thực biểu mẫu, xử lý tải lên, nhiều công nghệ xác thực mở và một số công cụ liên quan đến khuôn khổ chung. Một số ứng dụng sử dụng khung Flask bao gồm: Pinterest và LinkedIn.

Microsoft Power BI

Microsoft Power BI là một sản phẩm phần mềm trực quan hóa dữ liệu tương tác được phát triển bởi Microsoft với trọng tâm chính là kinh doanh thông minh (BI). Power BI là một phần của Microsoft Power Platform. Power BI là tập hợp các dịch vụ phần mềm, ứng dụng và trình kết nối, hoạt động cùng nhau để biến các nguồn dữ liệu không liên quan thành thông tin chi tiết, mạch lạc, trực quan. Dữ liệu có thể được nhập bằng cách đọc trực tiếp từ cơ sở dữ liệu, trang web hoặc các tệp có cấu trúc như bảng tính, CSV, XML và JSON.

Hình 1.6 Web Traffic Power BI Dashboard Example

Chương 2: THUẬT TOÁN XGBOOST TRONG BÀI TOÁN DỰ ĐOÁN

2.1 Tổng quan về học kết hợp

Các phương pháp học kết hợp (ensemble learning) khai thác nhiều thuật toán học máy khác nhau để tạo ra các kết quả dự đoán yếu dựa trên các đặc trưng được xuất thông qua các cách tiếp cận đa dạng bộ dữ liệu, và hợp nhất các kết quả bằng nhiều cơ chế bỏ phiếu khác nhau nhằm đạt được hiệu suất tốt hơn so với việc chỉ sử dụng bất kỳ thuật toán nào một cách riêng lẻ [2].

Phương pháp học kết hợp được chia thành 3 loại sau đây:

- Đóng bao là quá trình xây dựng một số lượng lớn các mô hình (thường là cùng loại) trên các tập dữ liệu con khác nhau được lấy mẫu từ tập huấn luyện (lấy mẫu ngẫu nhiên từ một tập dữ liệu để tạo ra một tập dữ liệu mới). Các mô hình này sẽ được huấn luyện độc lập và song song với nhau, sau đó đầu ra của chúng sẽ được kết hợp bằng cách tính trung bình để tạo ra kết quả cuối cùng.
- Tăng cường là quá trình xây dựng một chuỗi các mô hình (thường là cùng loại). Mỗi mô hình sau sẽ học cách sửa những lỗi của mô hình trước đó, tức là dữ liệu mà mô hình trước đó dự đoán sai. Điều này tạo ra một chuỗi các mô hình mà mô hình sau sẽ tốt hơn mô hình trước đó bằng cách cập nhật trọng số của dữ liệu qua mỗi mô hình. Cụ thể, trọng số của những dữ liệu mà mô hình dự đoán đúng sẽ không thay đổi, trong khi trọng số của những dữ liệu mà mô hình dự đoán sai sẽ được tăng thêm. Chúng ta sẽ sử dụng kết quả của mô hình cuối cùng làm kết quả trả về, vì mô hình càng về sau lại càng tốt hơn
- Xếp chồng kết hợp nhiều mô hình khác nhau, thường là các loại mô hình khác
- Xếp chồng kết hợp nhiều mô hình khác nhau, thường là các loại mô hình khác nhau. Mỗi mô hình được huấn luyện độc lập và sau đó một mô hình giám sát (meta model) sẽ học cách kết hợp những dự đoán từ các mô hình này để đưa ra dự đoán chung tốt nhất.

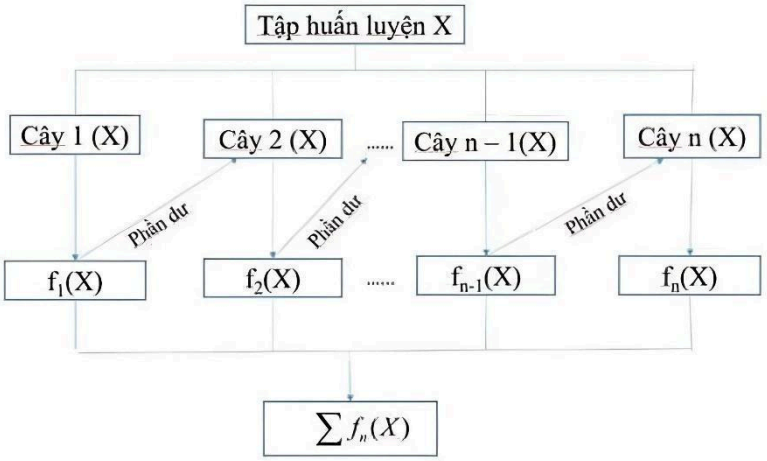
Khi kết hợp dự đoán từ các mô hình con, phương pháp đóng bao giúp giảm phương sai bằng cách "làm mềm" các nhiễu và biến động có thể xuất hiện từ một mô hình đơn lẻ. Còn phương pháp tăng cường thường tập trung vào việc giảm độ chệch

của mô hình do mỗi mô hình trong quá trình Boosting sẽ cố gắng tối ưu hóa phần dữ liệu mà mô hình trước đó dự đoán sai. Điều này dẫn đến việc mỗi mô hình mới được tạo ra sẽ cố gắng sửa chữa các lỗi mà các mô hình trước đó đã mắc phải. Cuối cùng, phương pháp xếp chồng có thể được sử dụng để giảm cả độ chệch (bias) và phương sai (variance) của mô hình.

2.2 Thuật toán XGBoost

XGBoost viết tắt của Extreme Gradient Boosting, là một mô hình được phát triển bởi Tianqi Chen trong khuôn khổ của cuộc thi "Kaggle" vào năm 2014. Sau đó vào năm 2016, XGBoost đã được công nhận rộng rãi trong cộng đồng học máy khi Tianqi Chen và Carlos Guestrin đã xuất bản bài báo "XGBoost: A Scalable Tree Boosting System" trên hội nghị KDD (Knowledge Discovery and Data Mining) [3].

Mô hình XGBoost sử dụng thuật toán tăng cường độ dốc cấp cao, thuộc loại Boosting method và là sự mở rộng của thuật toán Gradient Tree Boosting (GTB) được đề xuất bởi Friedman [4]. Nguyên lý cơ bản được sử dụng trong mô hình XGBoost là việc kết hợp các cây mô hình học tập có độ sai số cao thành một cây mô hình học tập mạnh hơn theo kiểu tuần tự, nghĩa là đào tạo các mô hình mới tốt hơn từ việc kết hợp các mô hình yếu trước đó để bù đắp các thiếu sót trong các mô hình trước, với sơ đồ trên hình:



Hình 2.1: Quy trình huấn luyện mô hình XGBoost

Mô hình XGBoost nhằm mục đích xử lý bài toán học máy có giám sát và cho độ tin cậy cao. Một trong những ưu điểm lớn của XGBoost là tốc độ huấn luyện nhanh chóng. Điều này được đạt được thông qua việc sử dụng tất cả các lõi CPU trong máy

tính, cho phép tính toán song song trong quá trình huấn luyện. Điều này làm cho việc huấn luyện mô hình trở nên hiệu quả, đặc biệt là đối với các bộ dữ liệu lớn.

XGBoost có nguồn gốc từ thuật toán Gradient Tree Boosting (GTB), nhưng được cải thiện với nhiều tối ưu hóa và điều chỉnh. Các cải tiến này bao gồm tối ưu hóa thuật toán, quản lý bộ nhớ hiệu quả hơn, và tận dụng tối đa hiệu suất tính toán từ cả phần cứng và phần mềm.

Thuật toán tăng cường độ dốc là trung tâm của XGBoost, nơi các mô hình được tạo ra tuần tự để dự đoán phần dư hoặc sai số của các mô hình trước đó. XGBoost sử dụng thuật toán giảm độ dốc để điều chỉnh mô hình mới nhằm giảm thiểu tổn thất toàn cục.

Với khả năng xử lý dữ liệu đa dạng, tốc độ huấn luyện nhanh và hiệu suất cao, XGBoost không chỉ là một công cụ mạnh mẽ cho các bài toán học máy mà còn là một lựa chọn phù hợp cho các dự án thực tiễn với dữ liệu lớn.

Cơ sở toán học

Với bộ dữ liệu đã cho $D = \{(x_i, y_i)\}(|D| = n, x_i \in R^m, y_i \in R)$, mô hình tổ hợp cây thường bao gồm nhiều cây, mỗi cây dự đoán đầu ra bằng cách kết hợp các dự đoán của tất cả các cây.

Hàm dự đoán

Hàm dự đoán cho một mẫu i được tính như sau:

$$\hat{y}_i = F(x_i) = \sum_{d=1}^{D} f_d(x_i),$$

Trong đó:

- $\hat{y}_i$ là giá trị dự đoán của mẫu i
- $F(x_i)$ là hàm dự đoán cuối cùng, là tổng của dự đoán từ tất cả D cây
- $f_d(x_i)$ là dự đoán từ cây thứ d cho mẫu i

Hàm mục tiêu: Hàm mục tiêu là sự kết hợp giữa hàm mất mát và một phần thưởng chính quy. Nó được định nghĩa tại lần lặp d như sau:

$$\mathcal{L}^{(d)} = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i^{(d-1)} + f_d(x_i)) + \sum_{j=1}^{T} \Omega(f_d),$$

Trong đó:

- $\ell(y_i, \hat{y}_i)$ là hàm mất mát đo lường sự khác biệt giữa giá trị thực y_i và giá trị dự đoán $\hat{y}_i$.
- $\Omega(f_d)$ là phần thưởng chính quy cho cây thứ d.

Hàm mất mát

Trong bài toán hồi quy, hàm mất mát thường được sử dụng là hàm mất mát bình phương:

$$\ell(y_i, \hat{y}_i) = \frac{1}{2}(y_i - \hat{y}_i)^2$$

Phần thưởng chính quy

Phần thưởng chính quy giúp ngăn chặn mô hình overfitting bằng cách phạt các mô hình quá phức tạp. Nó được định nghĩa như sau:

$$\Omega(f_d) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2,$$

Trong đó:

- γ là tham số phạt số lượng lá T.
- λ là hệ số phạt cho chuẩn ℓ_2 của trọng số lá w_j.

Cấu trúc cây

Cấu trúc q của cây có thể được định nghĩa như sau:

$$q : \mathbb{R}^p \to \{1, 2, \ldots, T\},$$

trong đó q chỉ định một quan sát vào một lá cụ thể j trong cây. Mỗi lá j có liên kết với một trọng số w_j, và dự đoán cho một quan sát x_i trong cây d được đưa ra bởi:

$$f_d(X_i) = w_{q(X_i)}$$

Gradient và Hessian

Gradient và Hessian của hàm mất mát được tính như sau:

$$g_i = \frac{\partial \ell(y_i, \hat{y}_i)}{\partial \hat{y}_i}$$

$$h_i = \frac{\partial^2 \ell(y_i, \hat{y}_i)}{\partial \hat{y}_i^2}$$

Hàm mục tiêu được xấp xỉ bằng khai triển Taylor bậc hai:

$$\mathcal{L}^{(d)} \approx \sum_{i=1}^{n} \left[\ell(y_i, \hat{y}_i^{(d-1)}) + g_i f_d(x_i) + \frac{1}{2} h_i f_d(x_i)^2 \right] + \Omega(f_d)$$

Trọng số lá tối ưu

Trọng số lá tối ưu w_j được tính như sau:

$$w_j = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

Trong đó: I_j là tập hợp các chỉ số cho các quan sát rơi vào lá j.

Cắt tỉa: XGBoost sử dụng kỹ thuật cắt tỉa cây để kiểm soát độ phức tạp của cây và tránh overfitting. Sau khi các cây được xây dựng, các nút không cung cấp lợi ích đáng kể sẽ được cắt tỉa bằng cách sử dụng tham số ngưỡng.

Giải thuật

Algorithm 1 Thuật toán Gradient Boosting với Xấp Xỉ Bậc Hai.

Require: Input: Tập huấn luyện $\{(x_i, y_i)\}_{i=1}^N$, hàm mất mát khả vi $L(y, F(x))$, số lượng bộ học yếu M, tốc độ học α

Ensure: Output: Mô hình được huấn luyện $\hat{f}(x)$

1: Khởi tạo mô hình với giá trị hằng số:

$$\hat{f}^{(0)}(x) = \arg\min_{\theta} \sum_{i=1}^{N} L(y_i, \theta)$$

2: **for** $m = 1$ đến M **do**

3: Tính Gradient và Hessian:

$$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$$

$$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\right]_{f(x)=\hat{f}^{(m-1)}(x)}$$

4: Huấn luyện một bộ học cơ bản ϕ_m sử dụng tập huấn luyện $\{x_i, \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\}_{i=1}^N$ bằng cách giải:

$$\hat{\phi}_m = \arg\min_{\phi \in \mathbf{\Phi}} \sum_{i=1}^{N} \frac{1}{2} \hat{h}_m(x_i) \left[\phi(x_i) - \frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\right]^2$$

5: Cập nhật mô hình:

$$\hat{f}_m(x) = \alpha \hat{\phi}_m(x)$$

$$\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \hat{f}_m(x)$$

6: **end for**

7: Đầu ra mô hình cuối cùng:

$$\hat{f}(x) = \hat{f}^{(M)}(x) = \sum_{m=0}^{M} \hat{f}_m(x)$$

Chương 3: THỰC NGHIỆM

3.1 Thu thập dữ liệu

Quá trình thu thập dữ liệu là bước đầu tiên trong đề tài *Smartphone Price Prediction in Big Data Environment*, bởi trong big data, chất lượng và khối lượng dữ liệu sẽ quyết định đến độ chính xác của mô hình dự đoán.

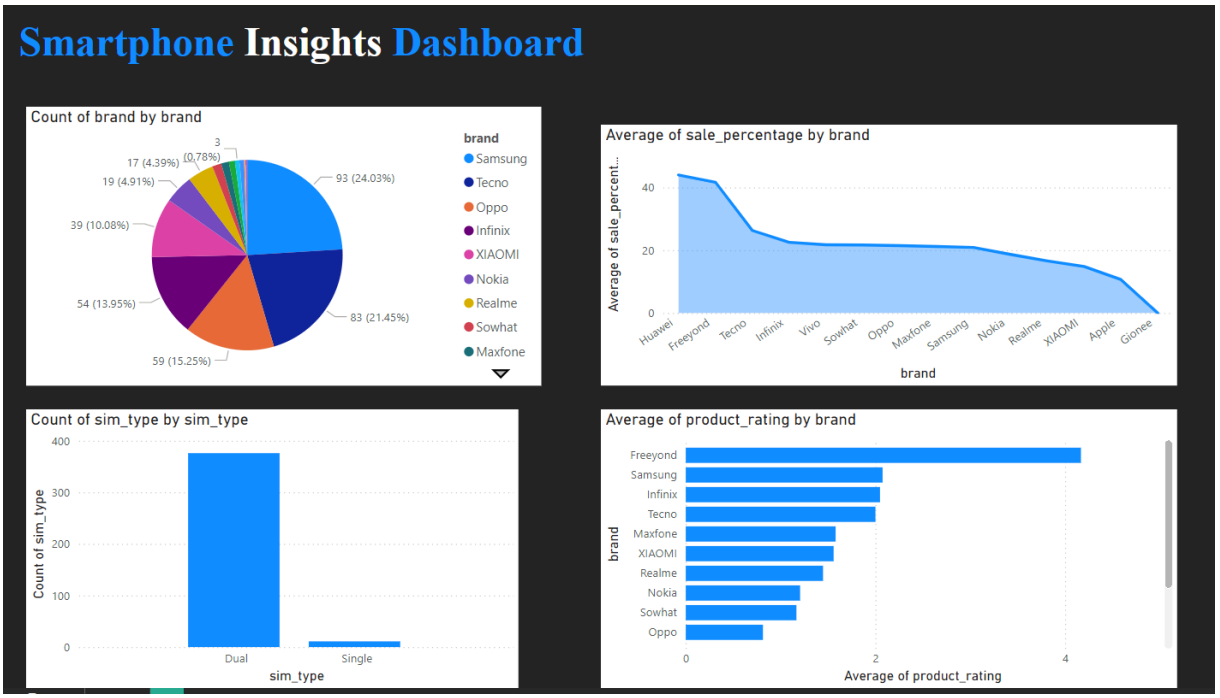
Dữ liệu chính trong dự án này:

- Nguồn dữ liệu Kaggle: Kaggle là một nền tảng phổ biến cung cấp các bộ dữ liệu phong phú cho nhiều lĩnh vực khác nhau, bao gồm thông tin chi tiết về các dòng smartphone, thông số kỹ thuật, và giá bán,... Việc sử dụng những dữ liệu từ Kaggle giúp đảm bảo rằng nhóm em có được nguồn dữ liệu chất lượng, đa dạng và có cấu trúc rõ ràng.
- Dữ liệu tự crawl: Trong đề tài này, để đám bảo tính thực tiễn và phù hợp với thị trường hiện tại, nhóm chúng em cũng đã tự tiến hành thu thập thêm dữ liệu từ các nguồn trực tuyến, trong đó bao gồm giá smartphone từ trang thương mại điện tử Jumia, một nền tảng nổi tiếng tại Nigeria. Dữ liệu từ Jumia cung cấp cái nhìn trực quan về thị trường smartphone tại một khu vực cụ thể, từ đó giúp dự đoán giá bán có sự khác biệt giữa các khu vực địa lý.

Tổng cộng, sau khi tích hợp cả hai nguồn, tập dữ liệu có kích thước khoảng 10,000,000 rows, đáp ứng tốt các yêu cầu về xử lý và phân tích dữ liệu lớn trong môi trường Big Data. Bước thu thập dữ liệu này đã đặt nền móng vững chắc cho các giai đoạn tiếp theo, từ tiền xử lý dữ liệu, xây dựng mô hình đến đánh giá kết quả. Dữ liệu phong phú và đa dạng từ các nguồn giúp đảm bảo tính chính xác và khả năng áp dụng thực tiễn của mô hình dự đoán.

3.2 Trực quan hóa dữ liệu

Sau khi thu thập dữ liệu, nhóm chúng em đã sử dụng công cụ Power BI để phân tích và trực quan hóa dữ liệu. Công cụ này giúp chuyển đổi các bảng dữ liệu phức tạp thành các biểu đồ, đồ thị và báo cáo dễ hiểu.



Hình 3.1 Trực quan hóa dữ liệu

Biểu đồ tròn - Count of brand by brand:

- Biểu đồ này hiển thị tỷ lệ số lượng smartphone của từng thương hiệu trong tổng số dữ liệu.
- Samsung chiếm tỷ lệ lớn nhất (24.03%), tiếp theo là Tecno (21.45%) và Oppo (15.25%). Các thương hiệu như Nokia, XIAOMI, và Infinix cũng đóng góp đáng kể, trong khi các thương hiệu nhỏ hơn như MaxPhone và SoWhat có tỷ lệ rất thấp.

Biểu đồ đường - Average of sale_percentage by brand:

- Biểu đồ đường minh họa tỷ lệ phần trăm doanh số trung bình theo từng thương hiệu.
- Huawei và Freeyond có doanh số trung bình cao nhất, trong khi các thương hiệu như Apple và Gionee nằm ở mức thấp nhất. Điều này cho thấy sự khác biệt lớn về khả năng tiêu thụ sản phẩm giữa các thương hiệu.

Biểu đồ cột - Count of sim_type by sim_type:

- Biểu đồ cột cho thấy sự phân bổ số lượng smartphone theo loại SIM (Dual hoặc Single).

- Loại SIM Dual áp đảo, chiếm số lượng lớn hơn nhiều so với SIM Single. Điều này có thể phản ánh nhu cầu của người dùng đối với tính năng sử dụng 2 SIM trên smartphone.

Biểu đồ cột ngang - Average of product_rating by brand:

- Biểu đồ này trực quan hóa điểm đánh giá trung bình của sản phẩm theo từng thương hiệu.
- Freeyond dẫn đầu với điểm đánh giá trung bình cao nhất, tiếp theo là Samsung và Infinix. Các thương hiệu như Oppo, Nokia, và SoWhat có điểm đánh giá thấp hơn, có thể cho thấy chất lượng sản phẩm hoặc mức độ hài lòng của người dùng không cao.

3.3 Huấn luyện model XGBoost

1. Dữ liệu huấn luyện được lấy từ dữ liệu.
2. Model huấn luyện sử dụng thư viện XGBoost, với khởi tạo tham số như sau:
    - số lượng cây: 100
    - độ sâu cây: 3
    - tốc độ học 0.1
    - tỉ lệ lấy mẫu mỗi cây : 0.8
    - tỉ lệ các đặc trưng được chọn cho một cây là:0.8
3. Đánh giá kết quả của mô hình XGBoost:
    - Giá trị MSE phản ánh độ chênh lệch bình phương trung bình giữa giá trị dự đoán và giá trị thực tế. Kết quả MSE lớn là do biến mục tiêu (giá điện thoại) nằm trong khoảng giá trị cao (hàng chục nghìn). Để cải thiện hiệu quả dự đoán, dữ liệu đã được scale từ đơn vị **VND** sang một thang đo thấp hơn.
    - R^2 (R-squared) là một chỉ số trong thống kê dùng để đánh giá mức độ phù hợp của một mô hình hồi quy với dữ liệu thực tế. Nó cho biết tỷ lệ phần trăm của biến thiên trong biến mục tiêu (biến phụ thuộc) được giải thích bởi các biến độc lập trong mô hình. Kết quả R^2 của mô hình đạt được là khoảng 0.9942 cho thấy mô hình có độ chính xác rất cao.

```
print("Mean Squared Error:", mse)
print("R^2 Score:", r2)
```

```
Mean Squared Error: 137734.10231770505
R^2 Score: 0.9942023382200806
```

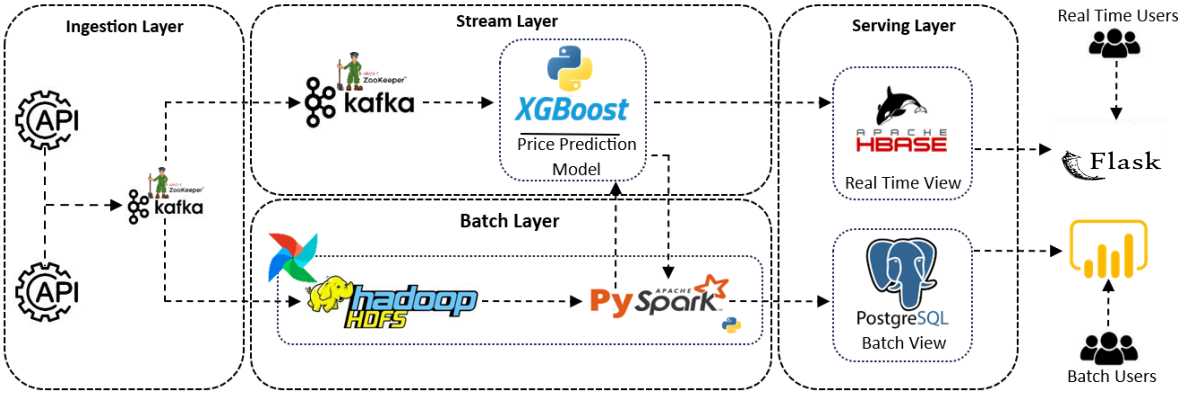
Hình 3.2 Hiệu suất mô hình XGBoost

3.4 Pipeline quá trình dự đoán giá điện thoại dựa trên kiến trúc lambda

Kiến trúc Lambda trong hệ thống dự đoán giá điện thoại giúp cân bằng giữa xử lý dữ liệu theo thời gian thực và theo lô:

- Phần luồng (Streaming): Đảm bảo dự đoán và lưu trữ dữ liệu thời gian thực, phù hợp với các ứng dụng yêu cầu cập nhật liên tục.
- Phần lô (Batch): Xử lý khối lượng lớn dữ liệu lịch sử để đưa ra dự đoán chính xác và lưu trữ dữ liệu có cấu trúc, hỗ trợ phân tích sâu hơn.

Với sự kết hợp của Kafka, PySpark, XGBoost, và các công cụ lưu trữ như HBase và PostgreSQL, hệ thống đảm bảo tính linh hoạt, hiệu quả và khả năng mở rộng trong việc dự đoán giá điện thoại.



Hình 3.3 Kiến trúc sử dụng

Cụ thể, hệ thống gồm hai phần chính :

1. Dự đoán giá điện thoại theo luồng với Kafka và XGBoost

Dữ liệu đầu vào:

- Dữ liệu được đọc từ API (hoặc minh họa qua file CSV trong dự án).
- Sau đó, Kafka Producer sẽ gửi dữ liệu đến Kafka Consumer.

Xử lý dữ liệu và dự đoán:

- Dữ liệu nhận được từ Kafka Consumer được chuyển đổi thành các đặc trưng phù hợp cho mô hình dự đoán.
- Mô hình XGBoost được sử dụng để dự đoán giá điện thoại theo thời gian thực.

Lưu trữ và hiển thị dữ liệu:

- Sau khi dự đoán, kết quả sẽ được lưu vào HBase.
- Việc sử dụng HBase đảm bảo rằng người dùng có thể truy cập và xem dữ liệu đã được dự đoán gần như ngay lập tức trong thời gian thực (Realtime View).

2. Dự đoán giá điện thoại theo lô và lưu trữ vào cơ sở dữ liệu.

Xử lý dữ liệu theo lô:

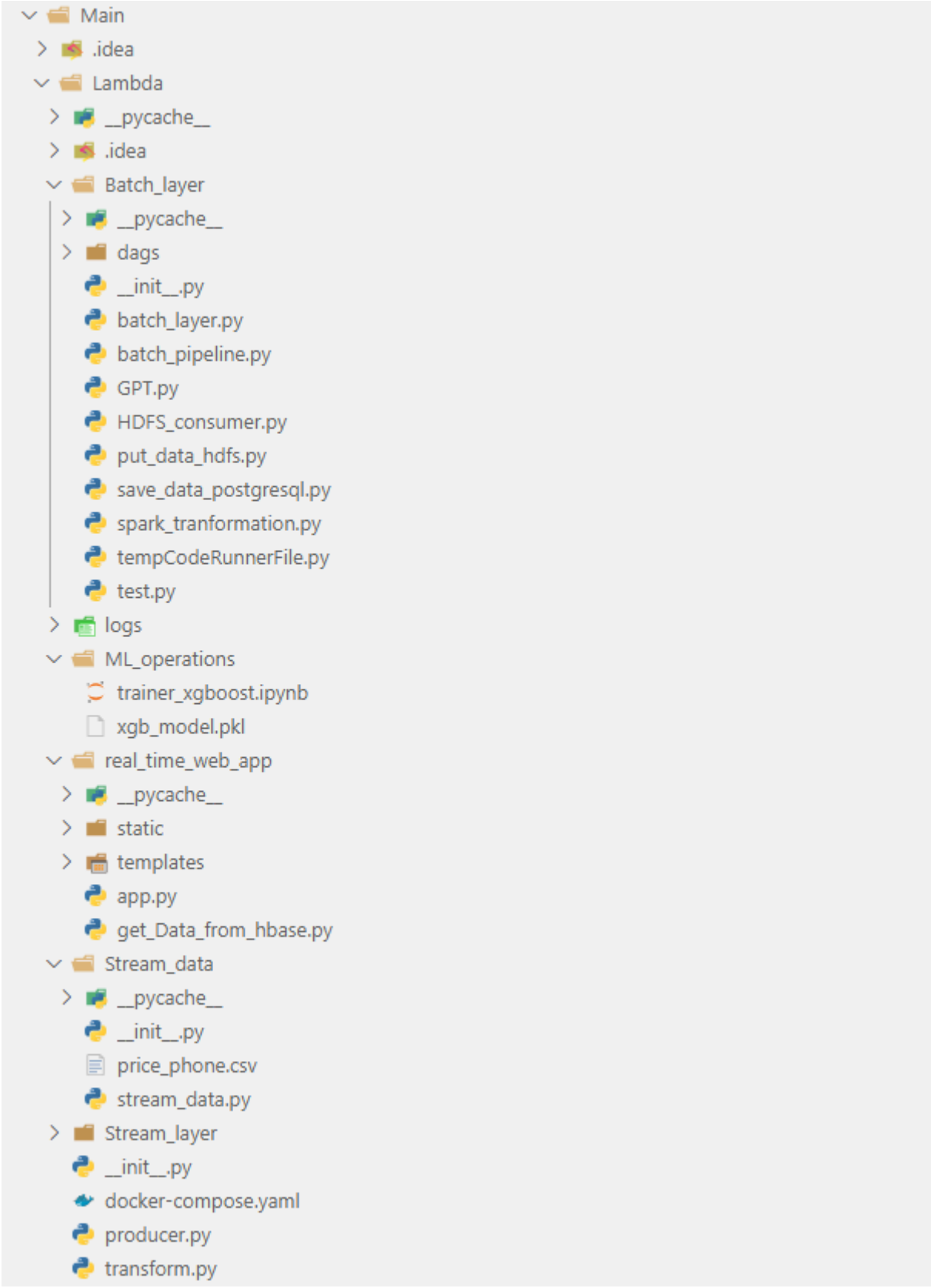
- Dữ liệu được đọc từ file CSV được lưu trên hệ thống file phân tán Hadoop HDFS. Cụ thể dữ liệu từ API được lưu trữ trong Hadoop HDFS như một phần của giải pháp hồ dữ liệu (Data Lake). Sau đó, PySpark thực hiện các bước chuyển đổi dữ liệu (data transformation) trên dữ liệu đã lưu trữ, đảm bảo chuẩn bị tốt cho các bước dự đoán và phân tích tiếp theo. Apache Airflow điều phối các luồng công việc xử lý theo lô, đảm bảo quá trình xử lý diễn ra đúng lịch trình và hiệu quả.
- Việc xử lý dữ liệu được thực hiện bằng PySpark, đảm bảo khả năng xử lý song song và hiệu quả với khối lượng dữ liệu lớn.

Dự đoán và lưu trữ:

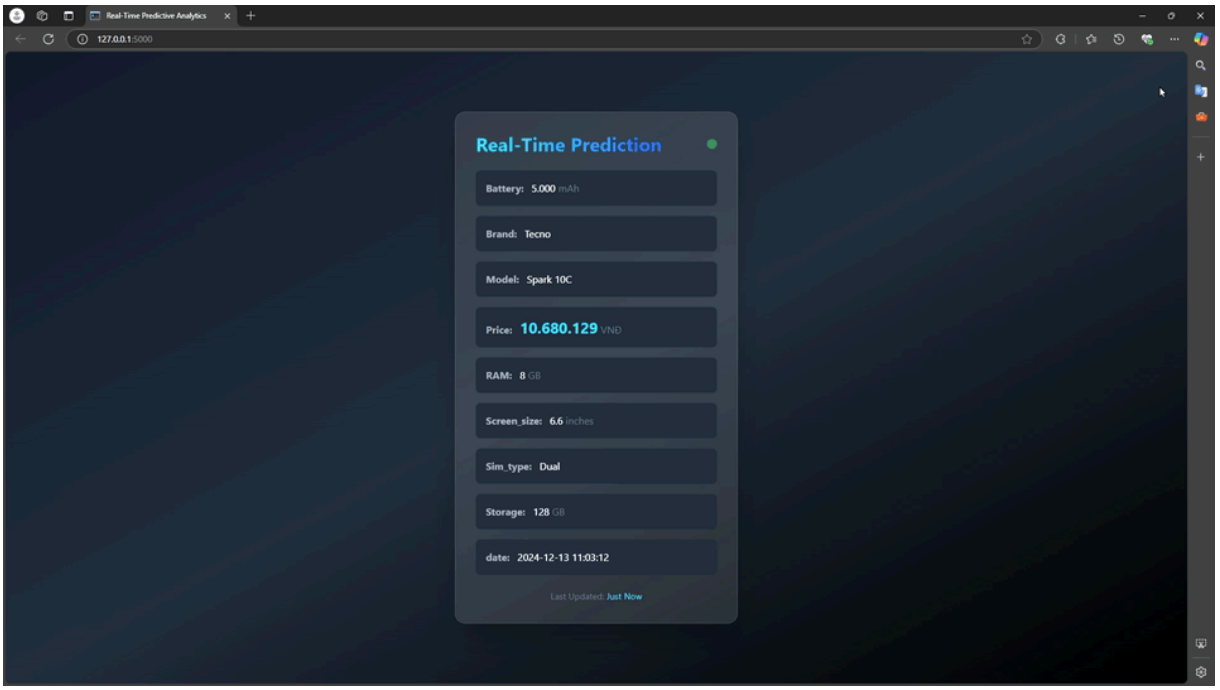
- Mô hình XGBoost sẽ được áp dụng để dự đoán giá điện thoại từ dữ liệu đã xử lý.
- Kết quả dự đoán sau đó được lưu vào cơ sở dữ liệu cục bộ, cụ thể là PostgreSQL, để phục vụ các ứng dụng phân tích hoặc truy xuất dữ liệu sau này.

3.5 Demo chương trình

Cấu trúc thư mục của project:



Kết quả chạy chương trình:


Real-Time Predictive Analytics
127.0.0.1:5000
Real-Time Prediction
Battery: 5.000 mAh
Brand: Tecno
Model: Spark 10C
Price: 10.680.129 VNĐ
RAM: 8 GB
Screen_size: 6.6 inches
Sim_type: Dual
Storage: 128 GB
date: 2024-12-13 11:03:12
Last Updated: Just Now

Chương 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 Kết luận

Dự án đã giúp nhóm chúng em tìm hiểu, học hỏi nhiều kiến thức trong việc xây dựng và triển khai kiến trúc xử lý dữ liệu Lambda, phục vụ cho bài toán dự đoán giá điện thoại cả ở chế độ thời gian thực và theo lô. Kiến trúc Lamda không chỉ cho thấy sự kết hợp hiệu quả của các công nghệ dữ liệu lớn mà còn làm nổi bật khả năng ứng dụng vào thực tiễn thông qua:

- Xử lý dữ liệu lớn: Apache Spark đảm nhận việc xử lý dữ liệu theo lô, đảm bảo hiệu quả và khả năng mở rộng. Kafka hỗ trợ tiếp nhận dữ liệu thời gian thực và giao tiếp giữa các thành phần trong hệ thống. HBase cung cấp khả năng truy cập nhanh chóng và lưu trữ dữ liệu thời gian thực. Hadoop HDFS lưu trữ dữ liệu thô như một phần của giải pháp data lake.
- Quản lý và trực quan hóa: Apache Airflow được sử dụng để điều phối luồng công việc xử lý dữ liệu. Power BI cung cấp giao diện trực quan hóa dữ liệu, hỗ trợ người dùng phân tích và đưa ra các quyết định dựa trên dữ liệu. Spring Boot giúp xây dựng giao diện web phục vụ người dùng thời gian thực.

Đề tài này đã tích hợp thành công các công cụ dữ liệu lớn, đồng thời cung cấp cách triển khai và vận hành một hệ thống xử lý dữ liệu phức tạp, mở ra hướng giải quyết nhiều bài toán thực tế khác**.**

4.2 Hướng phát triển

Kiến trúc lambda là một kiến trúc xử lý rất thú vị và hiệu quả cao nên dự án sẽ tiếp tục sử dụng kiến trúc này. Dưới đây là một số hướng phát triển tiếp theo cho dự án này:

Tăng cường năng lực mô hình dự đoán

Sử dụng các mô hình Deep Learning như LSTM, Transformer, hoặc Autoencoder để nâng cao độ chính xác trong dự đoán, đặc biệt với các bài toán yêu cầu phân tích dữ liệu phức tạp hơn hoặc có yếu tố thời gian. Tích hợp các công cụ hỗ trợ như TensorFlow Serving hoặc TorchServe để triển khai mô hình học sâu vào môi trường.

Cải thiện quy trình thu thập dữ liệu

Phát triển API tích hợp trực tiếp với các ứng dụng di động hoặc web để thu thập dữ liệu thời gian thực một cách liên tục và tự động, thay thế quy trình tải dữ liệu qua file CSV thủ công hiện tại. Áp dụng các công cụ thu thập dữ liệu nâng cao để mở rộng nguồn dữ liệu ví dụ: từ mạng xã hội, hệ thống thương mại điện tử hoặc các sàn giao dịch trực tuyến...

Mở rộng phạm vi bài toán

Từ bài toán dự đoán giá điện thoại, dự án có thể mở rộng sang các bài toán tương tự như: Dự đoán giá Bitcoin: Sử dụng dữ liệu thị trường tài chính theo thời gian thực để đưa ra dự đoán giá trị; Dự đoán doanh thu của sản phẩm: Phân tích doanh thu các sản phẩm khác như máy tính, quần áo, hoặc các mặt hàng tiêu dùng phổ biến; Dự đoán doanh thu của doanh nghiệp: Phục vụ các doanh nghiệp trong việc phân tích và dự báo doanh thu, hỗ trợ chiến lược kinh doanh.

Tăng cường hiệu suất hệ thống

Nâng cấp cơ sở hạ tầng bằng cách triển khai trên các nền tảng đám mây như AWS, Google Cloud, hoặc Azure, tận dụng các dịch vụ như EMR, BigQuery, hoặc Dataflow. Tối ưu hóa khả năng xử lý dữ liệu song song và theo luồng để giảm độ trễ, nâng cao trải nghiệm người dùng thời gian thực.

Tích hợp các bài toán khác trong hệ sinh thái dữ liệu lớn

Kết hợp các bài toán dự báo khác như phân tích hành vi người dùng, tối ưu hóa chuỗi cung ứng, hoặc dự đoán xu hướng tiêu dùng, từ đó tạo ra một hệ sinh thái dữ liệu lớn hoàn chỉnh và linh hoạt.

TÀI LIỆU THAM KHẢO

[1]. https://github.com/aymane-maghouti/Big-Data-Project/tree/main

[2]. https://www.databricks.com/glossary/hadoop-ecosystem

[3]https://atekco.io/1672307232989-cac-kien-truc-xu-ly-du-lieu-lambda-kappa-va-delta/

[4] https://spark.apache.org/docs/latest/api/python/index.html

[5] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html