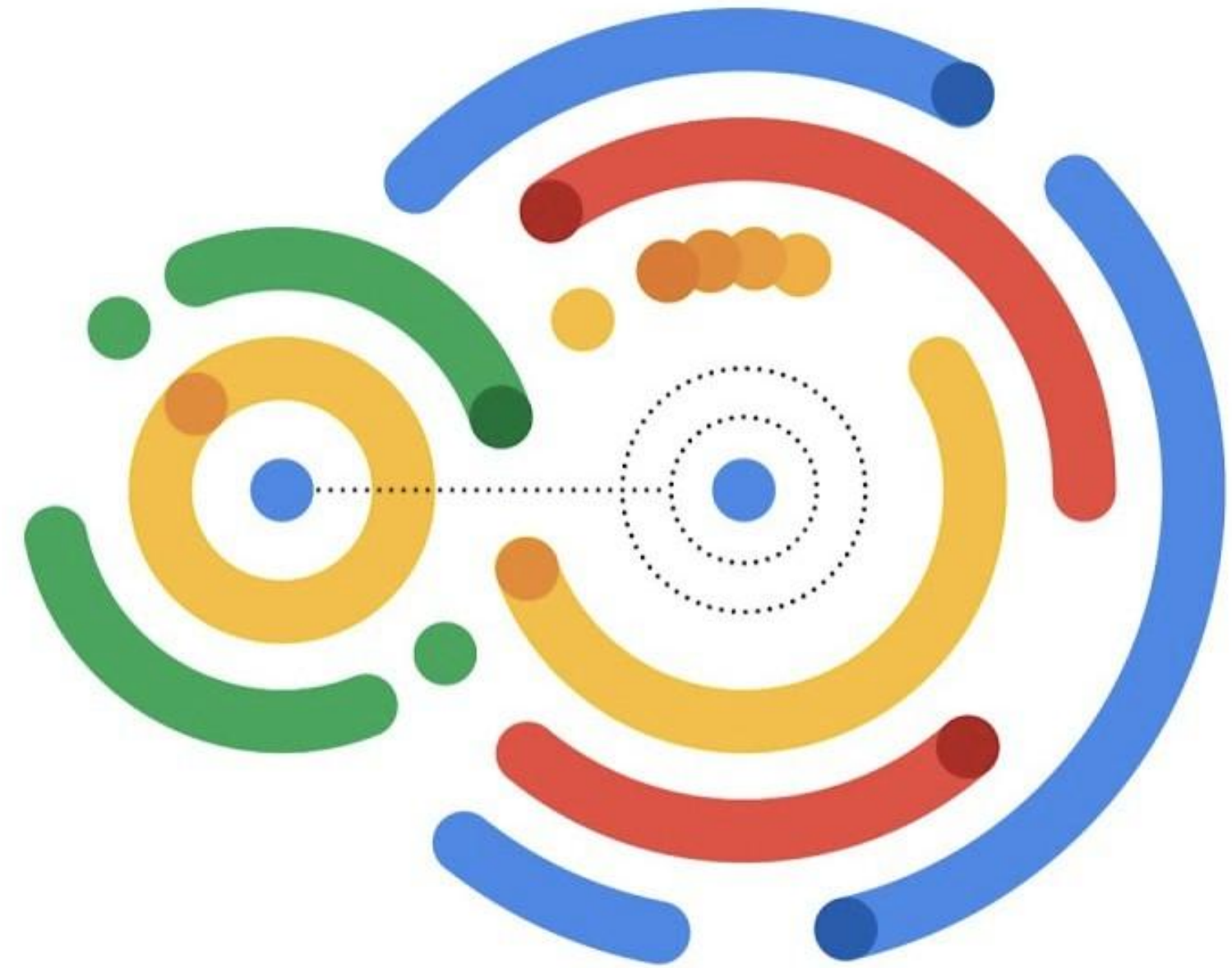


# Text Sentiment Analysis in Hadoop and Spark



## **Nhóm 10:**

**Bùi Hoàng Anh**

**MSSV: 20225154**

**Lê Hồng Sáng**

**MSSV: 20215298**

**Lê Đức Phương**

**MSSV: 20225380**

**Hoàng Trung Hải**

**MSSV: 20225307**

**Đỗ Dương Nghiêm**

**MSSV: 20225370**

**GVHN : Thầy Trần Việt Trung**

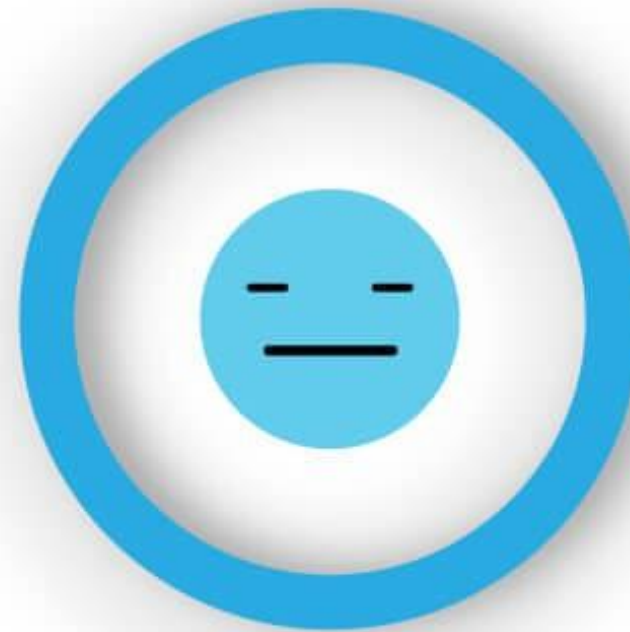


# SENTIMENT ANALYSIS



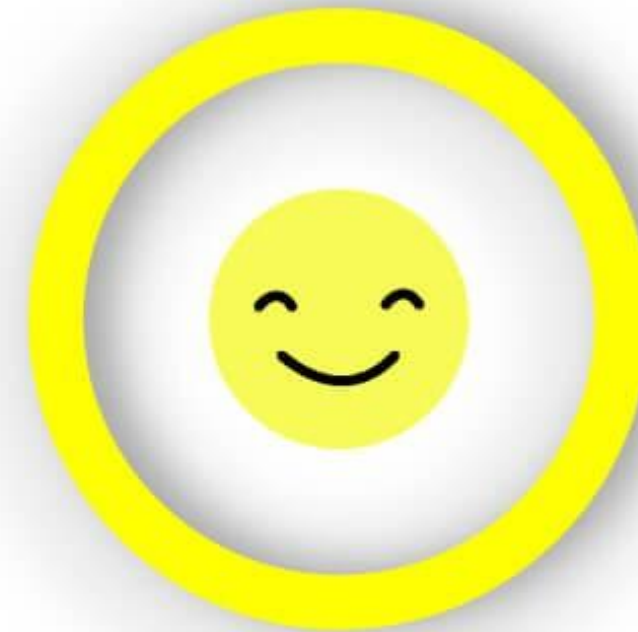
## NEGATIVE

Totally dissatisfied with the service. Worst customers care ever.



## NEUTRAL

Good Job but i will expect a lot more in future.



## POSITIVE

Brilliant effort guys! loved Your work.

# Content

01

**Giới thiệu**

02

**Tổng quan lý  
thuyết**

03

**Dữ liệu và tiền xử lý  
dữ liệu**

04

**Triển khai mô  
hình**

05

**Kết quả và đánh  
giá**

06

**Spark NLP và Deep  
Learning**

07

**Hadoop  
MapReduce**

08

**Kết luận**

# Content

01

**Giới thiệu**



# INTRODUCTION

## Bối cảnh:

- Mỗi ngày, lượng lớn dữ liệu văn bản được tạo ra trên mạng xã hội, diễn đàn, và trang thương mại điện tử, chứa nhiều ý kiến và cảm xúc của người dùng.
- Phân tích cảm xúc (Sentiment Analysis), một nhánh của NLP, giúp doanh nghiệp thấu hiểu khách hàng và đưa ra quyết định chiến lược.

## Thách thức:

- Dữ liệu văn bản khổng lồ khiến các phương pháp truyền thống kém hiệu quả và tốn kém.
- Cần giải pháp mạnh mẽ, mở rộng để xử lý, lưu trữ và phân tích dữ liệu lớn.

## Giải pháp:

- Hadoop: Cung cấp lưu trữ phân tán qua HDFS và xử lý dữ liệu với MapReduce.
- Spark: Xử lý in-memory nhanh chóng, triển khai mô hình học máy với MLlib, và nâng cao phân tích ngôn ngữ bằng Spark NLP.

## Lợi ích:

- Hệ thống hiệu quả, có khả năng mở rộng, giúp xử lý dữ liệu văn bản lớn nhanh chóng và chính xác.



# Mục Tiêu

Dự án này hướng tới các mục tiêu chính sau:

- Xây dựng hệ thống phân tích cảm xúc mở rộng với Hadoop và Spark
- Tiền xử lý và làm sạch dữ liệu văn bản thô từ nguồn dữ liệu để chuẩn bị cho quá trình phân tích.
- Triển khai và đánh giá hiệu suất của bốn mô hình phân loại phổ biến: Logistic Regression, Naive Bayes, Random Forest và Support Vector Machine (SVM) sử dụng thư viện Spark MLlib.
- Áp dụng thư viện Spark NLP để tiền xử lý và trích xuất các đặc trưng ngôn ngữ cho việc phân tích cảm xúc.
- Phân tích và so sánh kết quả của các mô hình, từ đó rút ra những đánh giá và đề xuất cải tiến.
- Khám phá và làm rõ những lợi ích của việc áp dụng Hadoop và Spark (bao gồm cả MapReduce) trong việc lưu trữ và phân tích cảm xúc trên dữ liệu lớn.



02

## **Tổng quan lý thuyết**



# Tổng quan lý thuyết

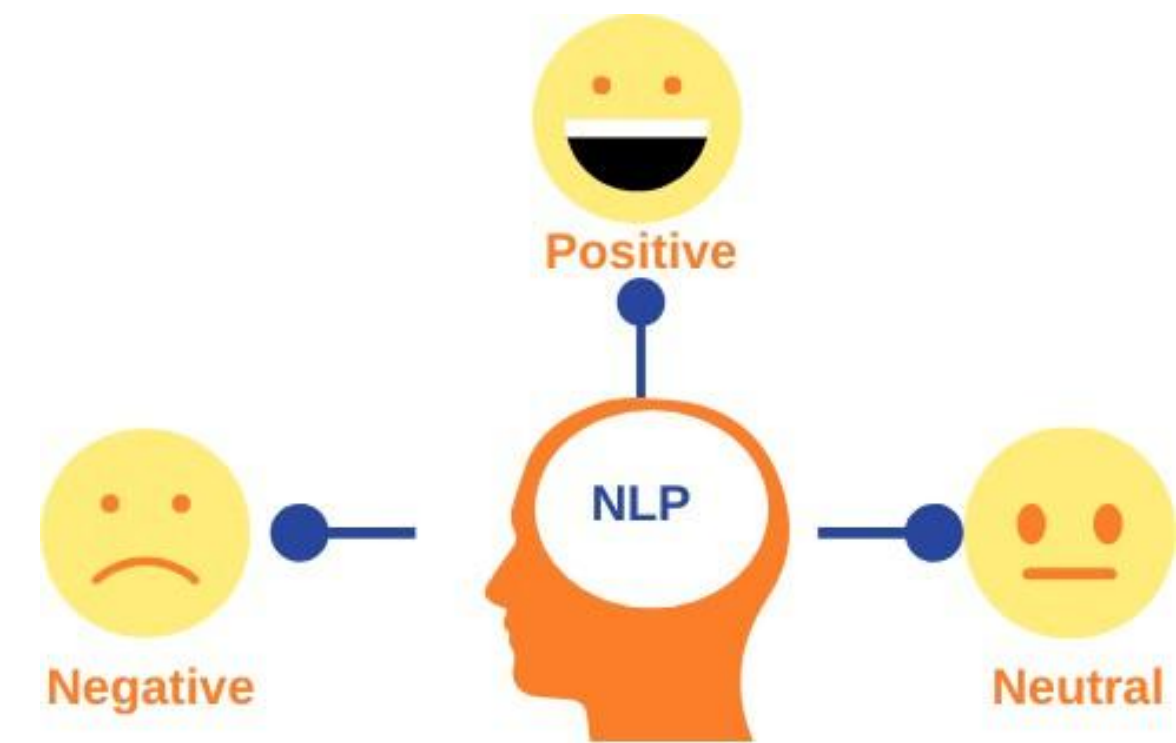
## Phân tích cảm xúc

- Là một lĩnh vực trong Xử lý ngôn ngữ tự nhiên (NLP), tập trung vào việc xác định và trích xuất thái độ, quan điểm, cảm xúc, hoặc đánh giá chủ quan của người viết/nói về các chủ thể, sự kiện, sản phẩm, hay dịch vụ từ dữ liệu văn bản.

## Ứng dụng:

Quản lý danh tiếng thương hiệu Nghiên

- cứu thị trường
- Dịch vụ khách hàng Phân
- tích chính trị Giám sát
- mạng xã hội
- 

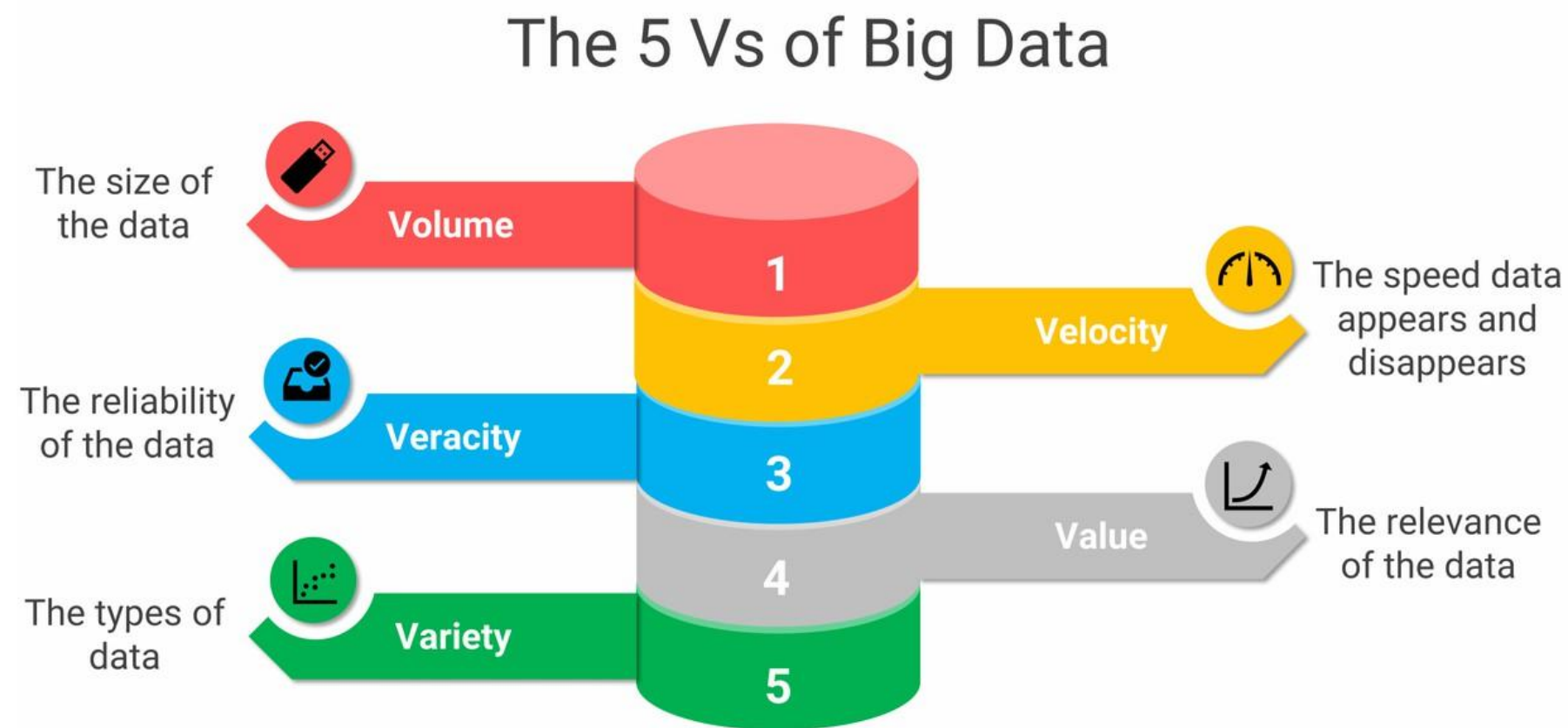


**Sentiment Analysis**

# Tổng quan lý thuyết

## Dữ liệu lớn (bigdata):

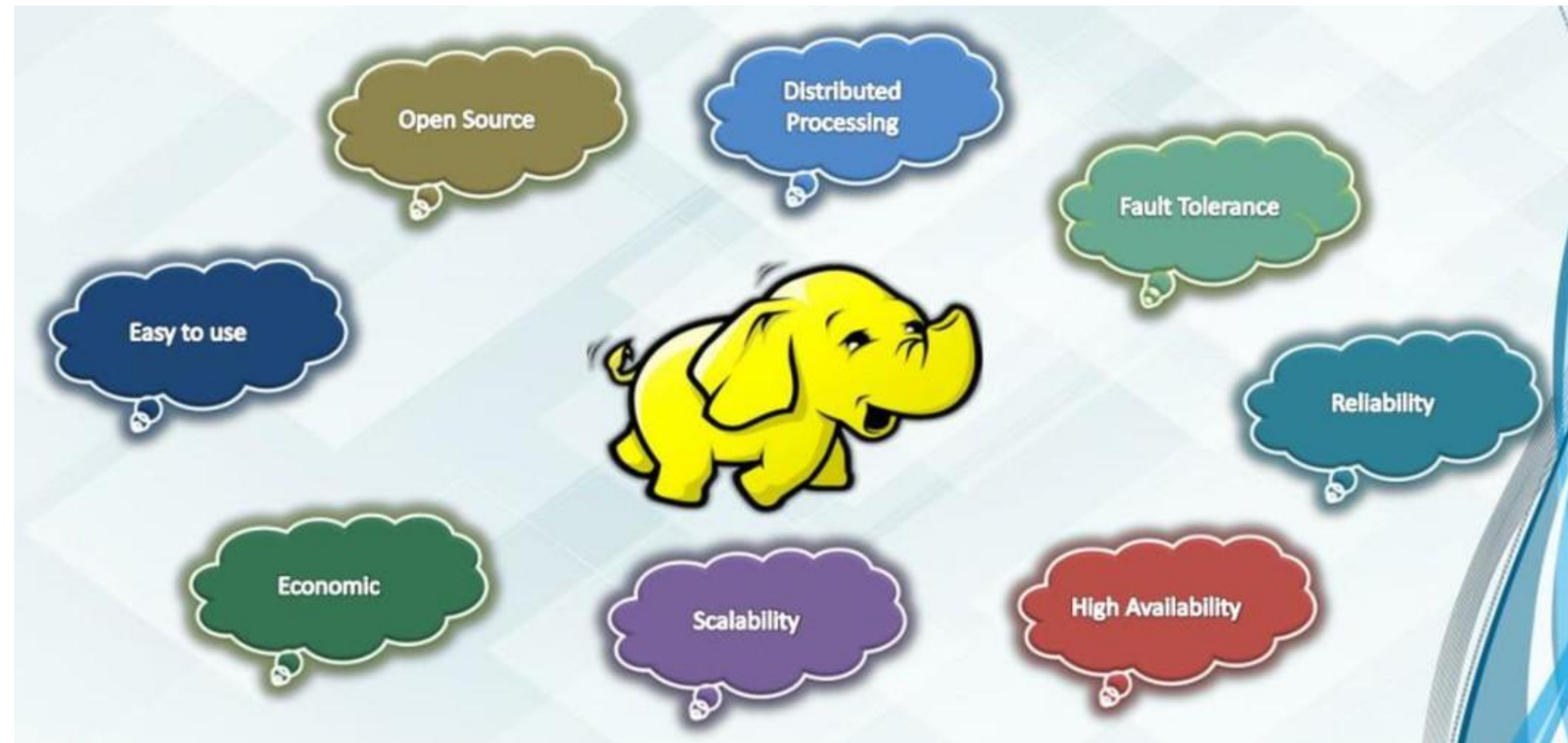
- Dữ liệu lớn (Big Data) là thuật ngữ dùng để chỉ các tập dữ liệu có khối lượng lớn, tốc độ tăng trưởng nhanh, và đa dạng về định dạng, vượt quá khả năng xử lý của các hệ thống quản trị cơ sở dữ liệu truyền thống. Dữ liệu lớn thường được đặc trưng bởi 5V:



# Hadoop

Hadoop là một framework mã nguồn mở được thiết kế để lưu trữ và xử lý dữ liệu lớn một cách phân tán trên các cụm máy tính thông thường. Hadoop bao gồm các thành phần chính sau:

- Hadoop Distributed File System (HDFS)
- MapReduce
- YARN (Yet Another Resource Negotiator)
- Các dự án khác: HBase, Hive, Pig, ...

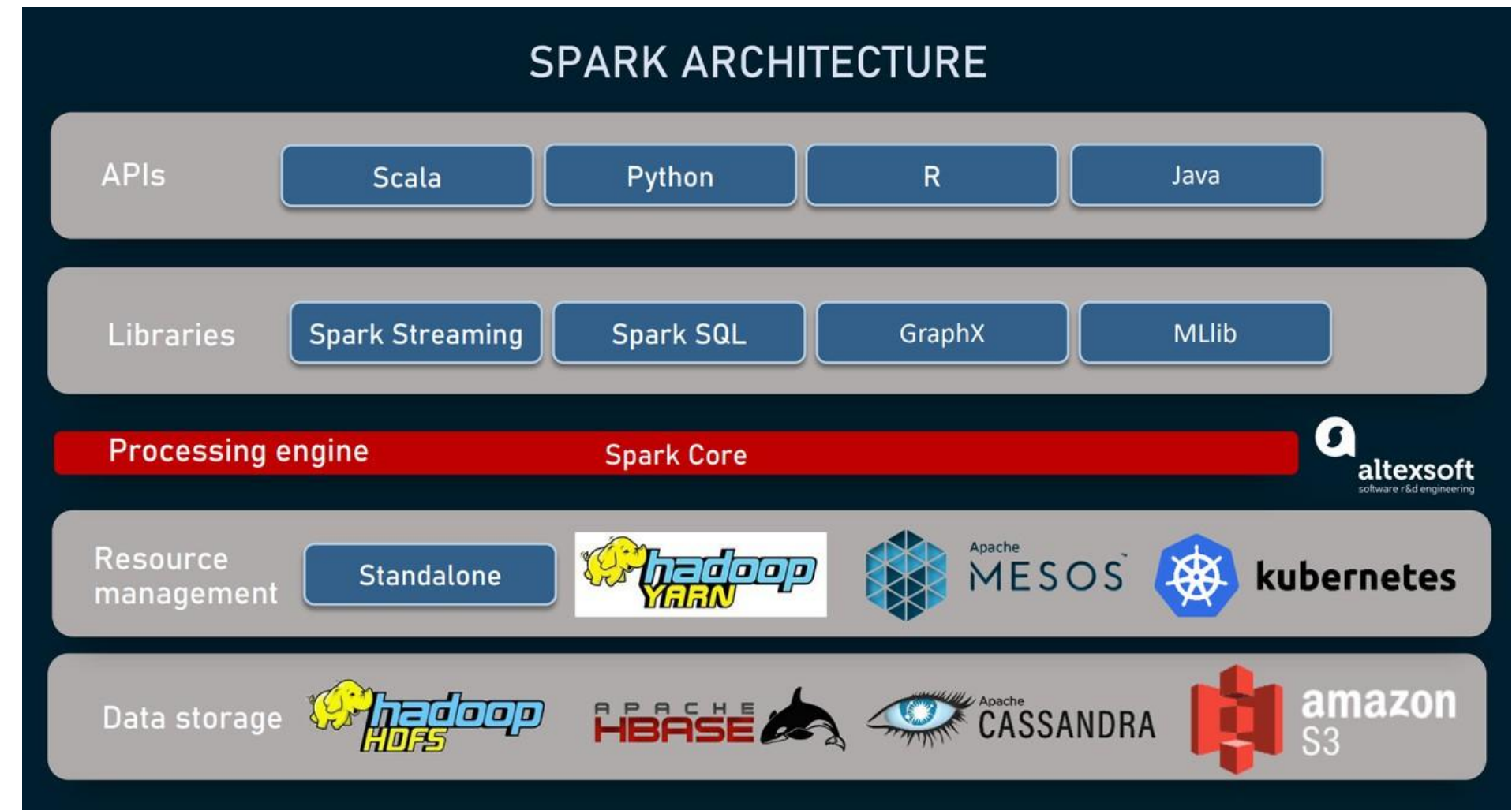




# Spark

Apache Spark là một framework mã nguồn mở, cung cấp một nền tảng thống nhất cho việc xử lý dữ liệu lớn với tốc độ nhanh và dễ sử dụng. Spark khắc phục một số hạn chế của Hadoop MapReduce, đặc biệt là về độ trễ và tính phức tạp. Các thành phần chính của Spark bao gồm:

- Spark Core
- Spark SQL
- Spark Streaming
- Spark MLlib
- Spark NLP
- GraphX



# Các mô hình phân loại trong Spark MLlib

## Logistic Regression:

- Logistic Regression là một thuật toán học máy được sử dụng cho các bài toán phân loại nhị phân (binary classification). Mô hình này ước tính xác suất của một đối tượng thuộc về một lớp cụ thể dựa trên các biến đầu vào.
- Cách hoạt động: Sử dụng hàm sigmoid để ánh xạ đầu ra của một hàm tuyến tính vào khoảng (0, 1), biểu diễn xác suất
- Ưu điểm: Đơn giản, dễ hiểu, và hiệu quả.
- Hạn chế: Giả định mối quan hệ tuyến tính giữa các biến đầu vào và đầu ra.

## Naive Bayes

- Naive Bayes là một thuật toán phân loại dựa trên định lý Bayes, với giả định "ngây thơ" (naive) rằng các đặc trưng là độc lập với nhau.
- Cách hoạt động: Tính toán xác suất của một đối tượng thuộc về từng lớp dựa trên xác suất xuất hiện của các đặc trưng trong từng lớp.
- Ưu điểm: Nhanh, hiệu quả, và yêu cầu ít dữ liệu huấn luyện.
- Hạn chế: Giả định về tính độc lập giữa các đặc trưng thường không đúng trong thực tế.

# Các mô hình phân loại trong Spark MLlib

## Random Forest

- Random Forest là một thuật toán học máy thuộc nhóm ensemble learning, xây dựng nhiều cây quyết định (Decision tree) và kết hợp kết quả dự đoán của các cây này để đưa ra kết quả cuối cùng.
- Cách hoạt động: Mỗi cây quyết định được huấn luyện trên một tập con ngẫu nhiên của dữ liệu huấn luyện và sử dụng một tập con ngẫu nhiên của các đặc trưng. Kết quả dự đoán của các cây được kết hợp bằng cách lấy trung bình (đối với bài toán hồi quy) hoặc bỏ phiếu (đối với bài toán phân loại).

## Support Vector Machine (SVM)

- Support Vector Machine (SVM) là một thuật toán học máy hiệu quả cho cả bài toán phân loại và hồi quy. Trong phân loại nhị phân, SVM tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân tách hai lớp dữ liệu, sao cho khoảng cách từ siêu phẳng đó đến các điểm dữ liệu gần nhất (support vectors) của mỗi lớp là lớn nhất (maximum margin).
- Cách hoạt động: SVM tìm kiếm siêu phẳng tối ưu trong không gian đặc trưng để phân tách hai lớp dữ liệu. Trong trường hợp dữ liệu không phân tách tuyến tính, SVM sử dụng kỹ thuật kernel để ánh xạ dữ liệu sang không gian đặc trưng mới, nơi dữ liệu có thể được phân tách tuyến tính.

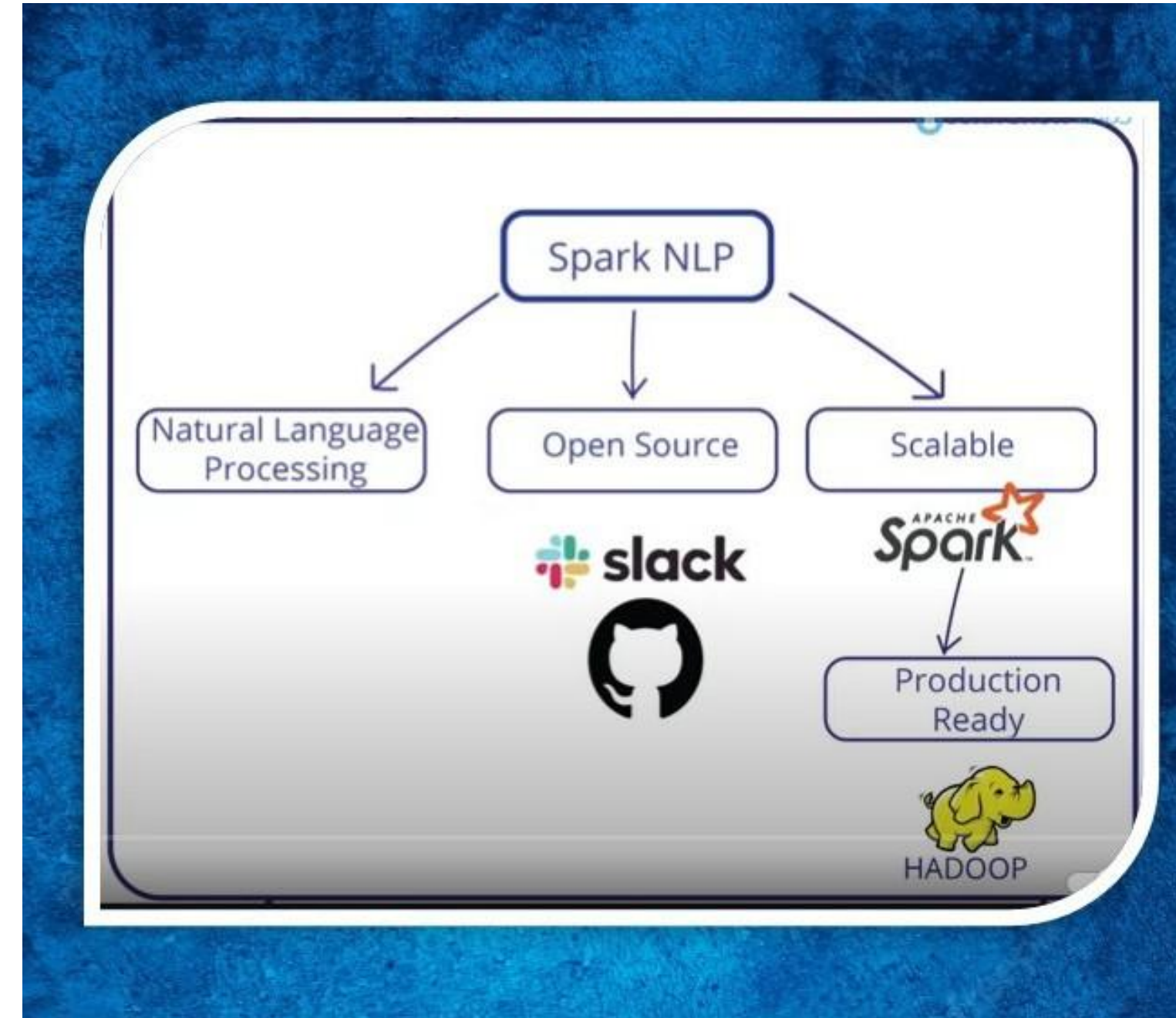


# Spark NLP

Spark NLP là một thư viện xử lý ngôn ngữ tự nhiên (NLP) mã nguồn mở, được xây dựng trên nền tảng Apache Spark, cung cấp các API đơn giản, hiệu suất cao và có khả năng mở rộng cho các tác vụ NLP. Spark NLP hỗ trợ nhiều mô hình NLP hiện đại, bao gồm các mô hình Deep Learning, cho phép thực hiện các nhiệm vụ như phân tích cảm xúc, nhận dạng thực thể, phân loại văn bản, v.v. với độ chính xác cao.

Một số ưu điểm nổi bật của Spark NLP bao gồm:

- Hiệu suất cao
- Độ chính xác cao
- Khả năng mở rộng
- Hỗ trợ nhiều ngôn ngữ
- Dễ sử dụng



# Thành phần chính trong Spark NLP

Spark NLP cung cấp nhiều thành phần quan trọng cho xử lý ngôn ngữ tự nhiên. Trong dự án này, chúng ta đặc biệt quan tâm đến 'ClassifierDLApproach'

- ClassifierDLApproach: Đây là một thành phần cho phép huấn luyện mô hình phân loại văn bản sử dụng mạng nơ-ron sâu (Deep Learning). Nó nhận đầu vào là văn bản đã được mã hóa thành các vector đặc trưng và nhãn tương ứng, sau đó huấn luyện một mô hình deep learning để phân loại văn bản vào các lớp khác nhau. ClassifierDLApproach cung cấp các tùy chọn để tinh chỉnh kiến trúc mạng nơ-ron, hàm mất mát, và thuật toán tối ưu để đạt được hiệu suất tốt nhất trên tập dữ liệu cụ thể.

# Naive Bayes trong Hadoop MapReduce

Triển khai Naive Bayes trong Hadoop MapReduce thường bao gồm các bước sau:

## 1. Map (Giai đoạn ánh xạ):

- Input: Mỗi mapper nhận một phần của tập dữ liệu huấn luyện, trong đó mỗi mẫu dữ liệu bao gồm các đặc trưng (ví dụ: các từ trong một tweet) và nhãn cảm xúc tương ứng.
- Output: Các mapper sẽ xuất ra các cặp key-value. Key là tổ hợp của (nhãn, từ) và value là 1. Điều này cho phép thống kê số lần xuất hiện của mỗi từ trong từng lớp cảm xúc.

## 2. Shuffle and Sort (Giai đoạn xáo trộn và sắp xếp):

- Hadoop tự động xáo trộn và sắp xếp các cặp key-value theo key, đưa các cặp có cùng key đến cùng một reducer.

## 3. Reduce (Giai đoạn rút gọn):

- Input: Các reducer nhận các cặp key-value có cùng key.
- Output: Mỗi reducer tính tổng số lần xuất hiện của mỗi từ trong mỗi lớp (tần suất xuất hiện của từ cho mỗi nhãn) và xuất ra một cặp key-value mới. Key là (nhãn, từ) và value là số lần xuất hiện của từ trong nhãn đó.

# Naive Bayes trong Hadoop MapReduce

## 4. Tính toán xác suất (Sau giai đoạn MapReduce):

- Từ output của các reducer, tính toán các xác suất cần thiết cho thuật toán Naive Bayes:
  - $P(\text{label})$ : Xác suất của mỗi lớp cảm xúc.
  - $P(\text{word}|\text{label})$ : Xác suất của mỗi từ xuất hiện trong mỗi lớp cảm xúc.
- Các xác suất này có thể được lưu trữ để sử dụng cho việc phân loại.

## 5. Phân loại (Sử dụng các xác suất đã tính toán):

- Cho một tweet mới, sử dụng các xác suất đã tính toán để dự đoán lớp cảm xúc của tweet đó.

### Ưu điểm:

- Hiểu rõ cơ chế hoạt động: Giúp hiểu rõ hơn về cơ chế hoạt động của Naive Bayes trên môi trường phân tán.
- Khả năng tùy chỉnh: Cung cấp nhiều tùy chỉnh hơn so với việc sử dụng các thư viện có sẵn.

### Hạn chế:

- Phức tạp: Cần nhiều công sức để triển khai so với việc sử dụng thư viện SparkMLlib.
- Tổn thời gian: Quá trình xử lý có thể tốn thời gian hơn so với các triển khai được tối ưu hóa, đặc biệt là khi hoạt động trên máy ảo (dùng trong dự án) bị hạn chế về tài nguyên và cấu hình.

# Content

03

**Dữ liệu và tiên  
xử lý dữ liệu**



# Dữ liệu và tiền xử lý dữ liệu

## 3.1 Mô tả dữ liệu

- Nguồn gốc dữ liệu: Dữ liệu sử dụng trong project này là tập dữ liệu Sentiment 140+Kaggle đã được tiền xử lý qua, bao gồm 1,6 triệu tweet được thu thập thông qua Twitter API và đã được gán nhãn cảm xúc 0 (tiêu cực) hoặc 1 (tích cực).
- Kích thước dữ liệu: Tập dữ liệu gần 1,6 triệu tweets, với tổng dung lượng khoảng 150MB.
- Cấu trúc dữ liệu: Được lưu dưới định dạng csv. Mỗi bản ghi trong tập dữ liệu bao gồm 4 trường thông tin: ItemID (mã định danh tweet), Sentiment (nhãn cảm xúc), SentimentSource (nguồn lấy dữ liệu), SentimentText (nội dung tweet), ví dụ:

ItemID	Sentiment	SentimentSource	SentimentText
1	0	Sentiment140	is so sad for my APL friend.....
2	0	Sentiment140	I missed the New Moon trailer...
3	1	Sentiment140	omg its already 7:30 :O
...	...	...	...

**Bảng 3.1:** Tổng quan dữ liệu



# Dữ liệu và tiền xử lý dữ liệu

## 3.2 Tiền xử lý dữ liệu

- Làm sạch dữ liệu:
  - Kiểm tra giá trị thiếu và trùng lặp để xử lý.
  - Loại bỏ những tweet quá dài (>180 từ, được tính là ngoại lệ, không đáng kể số lượng).
  - Loại bỏ các kí tự đặc biệt, số, hashtag, username, URL, khoảng trắng ở đầu và cuối câu.
- Chuẩn hóa văn bản:
  - Chuyển văn bản từ chữ hoa về chữ thường.
  - Stemming/Lemmatization: Ví dụ stemming là chuyển "running" thành "run" hoặc Lemmatization là chuyển "better" thành "good".
- Loại bỏ stop words: Loại bỏ các từ phổ biến nhưng ít mang ý nghĩa như các mạo từ 'a', 'an', 'the', ...
- Tách văn bản thành các từ (tokens).

# Lưu trữ dữ liệu trên HDFS

Sau khi tiền xử lý, dữ liệu được lưu trữ trên Hệ thống tệp phân tán Hadoop (HDFS) dưới định dạng Parquet bằng cách sử dụng Spark.

Lý do lựa chọn HDFS:

- Lưu trữ dữ liệu lớn, phân tán: HDFS được thiết kế để lưu trữ dữ liệu lớn, phân tán trên nhiều máy.
- Chịu lỗi tốt: HDFS có cơ chế sao lưu dữ liệu, đảm bảo an toàn dữ liệu khi có lỗi xảy ra.
- Tích hợp tốt với Spark: HDFS và Spark kết hợp tốt, cho phép xử lý dữ liệu hiệu quả.

Lý do lựa chọn định dạng Parquet:

- Truy vấn nhanh: Parquet lưu trữ dạng cột, giúp Spark chỉ đọc những cột cần thiết, tăng tốc độ truy vấn.
- Nén dữ liệu tốt: Parquet hỗ trợ nhiều thuật toán nén, giảm dung lượng lưu trữ.
- Tối ưu với Spark: Parquet được thiết kế tối ưu cho Spark, giúp tăng hiệu suất truy vấn.

# Content

04

**Triển khai mô hình**

# Triển khai mô hình bằng Spark MLlib

## 4.1 Biểu diễn đặc trưng

- Để các mô hình học máy có thể làm việc với dữ liệu văn bản, chúng ta cần chuyển đổi văn bản thành dạng vector số học. Quá trình này được gọi là biểu diễn đặc trưng hay trích xuất đặc trưng. Trong nghiên cứu này, kỹ thuật CountVectorizer kết hợp với IDF (Inverse Document Frequency) được sử dụng để biểu diễn đặc trưng cho các tweets.
- Các bước thực hiện:
  - CountVectorizer: Sử dụng 'CountVectorizer' của Spark MLlib để đếm tần suất xuất hiện của mỗi từ (đã được lemmatized- cột 'lemmatized\_words') trong từng tweet. 'CountVectorizer' tạo ra một từ điển (vocabulary) từ toàn bộ tập dữ liệu và biểu diễn mỗi tweet dưới dạng một vector, trong đó mỗi phần tử của vector tương ứng với số lần xuất hiện của một từ trong từ điển.
  - IDF: Sử dụng 'IDF' của Spark MLlib để tính toán trọng số IDF cho mỗi từ dựa trên cột 'raw\_features'. IDF giúp giảm trọng số của các từ xuất hiện phổ biến trong toàn bộ tập dữ liệu và tăng trọng số của các từ hiếm, có khả năng phân biệt cao. Kết quả được lưu trong cột 'features', là cột vector đặc trưng cuối cùng được sử dụng cho các mô hình phân loại.
- Sau khi thu được cột vector đặc trưng 'features' ta giữ lại cột này và cột 'Sentiment' (cột label) và tiến hành loại bỏ các cột không cần thiết

# RRN Encoder-Decoder

## 4.2 Chia dữ liệu

- Tập dữ liệu được chia thành hai tập con: tập huấn luyện và tập kiểm tra theo tỷ lệ 9:1. Tập huấn luyện được sử dụng để huấn luyện các mô hình phân loại, trong khi tập kiểm tra được sử dụng để đánh giá hiệu suất của các mô hình trên dữ liệu chưa từng thấy. Trong Spark, hàm 'randomSplit()' được sử dụng để chia dữ liệu một cách ngẫu nhiên

## 4.3 Xây dựng mô hình với Spark MLlib

- Để huấn luyện các mô hình phân loại, chúng ta sử dụng pipeline của Spark MLlib. 'Pipeline' cho phép kết hợp các bước tiền xử lý dữ liệu, trích xuất đặc trưng và huấn luyện mô hình thành một quy trình thống nhất. Trong nghiên cứu này, pipeline bao gồm các bước: 'CountVectorizer', 'IDF', và mô hình phân loại (Logistic Regression, Naive Bayes, Random Forest, hoặc SVM). Đầu vào của pipeline là cột 'lemmatized\_words' và 'Sentiment', và đầu ra là mô hình đã được huấn luyện.

Bốn mô hình phân loại được triển khai:

- Logistic Regression
- Naive Bayes
- RandomForest
- Support Vector Machine

# Logistic Regression

**Logistic Regression** là một mô hình phân loại tuyến tính, ước tính xác suất của một đầu vào thuộc về một lớp cụ thể. Trong Spark MLlib, 'LogisticRegression' được sử dụng để huấn luyện mô hình Logistic Regression. Có tham số như sau:

- 'maxIter = 20': Số lần lặp tối đa.
- 'regParam = 0.3': Tham số regularization (chính quy hóa), giúp tránh overfitting bằng cách thêm một số hạng phạt vào hàm mất mát (loss function). Giá trị 'regParam' càng lớn, mức độ chính quy hóa càng mạnh.
- 'elasticNetParam = 0': Tham số Elastic Net, là sự kết hợp giữa hai phương pháp regularization L1 (Lasso) và L2 (Ridge). 'elasticNetParam' kiểm soát tỷ lệ giữa L1 và L2. Giá trị 0 tương đương với L2 regularization, giá trị 1 tương đương với L1 regularization, và giá trị giữa 0 và 1 là sự kết hợp của cả hai.



# Naive Bayes và Random Forest

**Naive Bayes** là một mô hình phân loại dựa trên định lý Bayes, với giả định ngây thơ (naive) rằng các đặc trưng là độc lập với nhau. Trong Spark MLlib, 'NaiveBayes' được sử dụng để huấn luyện mô hình Naive Bayes. Có tham số như sau:

- 'smoothing = 1.0': Tham số làm mịn (smoothing parameter), tránh trường hợp xác suất bằng 0.
- 'modelType = "multinomial"': Loại mô hình Naive Bayes.

**Random Forest** là một mô hình học máy thuộc nhóm ensemble learning, xây dựng nhiều cây quyết định (decision tree) và kết hợp kết quả dự đoán của các cây này. Trong Spark MLlib, 'RandomForestClassifier' được sử dụng để huấn luyện mô hình Random Forest. Có tham số như sau:

- 'numTrees = 40': Số lượng cây quyết định).
- 'maxDepth = 5': Độ sâu tối đa của cây.
- 'maxBins = 64': Số lượng bins tối đa được sử dụng khi chia nhỏ các giá trị thuộc tính liên tục.

# Support Vector Machine (SVM)

Support Vector Machine (SVM) là một mô hình phân loại tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân tách hai lớp dữ liệu trong không gian đặc trưng, sao cho margin (khoảng cách từ siêu phẳng tới các điểm dữ liệu gần nhất của mỗi lớp) là lớn nhất. Trong Spark MLlib, LinearSVC được sử dụng để huấn luyện mô hình SVM tuyến tính. Có tham số như sau:

- 'maxIter = 100': Số lần lặp tối đa.
- 'regParam = 0.1': Tham số regularization, kiểm soát độ lớn của margin. Giá trị regParam càng lớn, mô hình càng cố gắng phân loại đúng tất cả các điểm dữ liệu huấn luyện, dẫn đến margin nhỏ hơn và có thể gây ra overfitting.

# Đánh giá mô hình

- Hiệu suất của các mô hình được đánh giá trên tập kiểm tra bằng các độ đo sau:
  - Accuracy (Độ chính xác): Tỷ lệ phần trăm các dự đoán đúng trên tổng số các dự đoán.
  - F1-score: Trung bình điều hòa của Precision và Recall, kết hợp cả hai độ đo.
- Trong Spark MLlib, 'MulticlassClassificationEvaluator' được sử dụng để tính toán các độ đo trên. Ngoài ra, Confusion Matrix cũng được sử dụng để cung cấp cái nhìn chi tiết hơn về hiệu suất của từng mô hình. Ta sẽ biết biết số lượng các điểm dữ liệu được dự đoán đúng và sai cho từng lớp.
- Sau khi huấn luyện và đánh giá, các mô hình được lưu trữ trên HDFS để có thể tái sử dụng trong tương lai mà không cần phải huấn luyện lại. Việc này giúp tiết kiệm thời gian và tài nguyên tính toán, đặc biệt là đối với các mô hình lớn và phức tạp.

# Content

05

**Kết quả và đánh giá**

# Kết quả

- Bảng 5.1 là tổng hợp kết quả đánh giá các mô hình trên tập kiểm tra, sử dụng hai độ đo chính là Accuracy và F1-score:

Mô hình	Accuracy	F1-score
Naive Bayes	0.8015	0.8015
Logistic Regression	0.8187	0.8187
Random Forest	0.6535	0.6461
Support Vector Machine	0.7919	0.7915

**Bảng 5.1:** Kết quả đánh giá các mô hình



# Kết quả

**Bảng 5.2:** Kết quả đánh giá các mô hình

Mô hình	Điểm Mạnh	Điểm Yếu
Logistic Regression	Đơn giản, dễ hiểu, hiệu quả, nhanh chóng	Giả định mối quan hệ tuyến tính, khó biểu diễn mối quan hệ phức tạp
Naive Bayes	Nhanh, hiệu quả, cần ít dữ liệu huấn luyện	Giả định các đặc trưng độc lập, có thể kém chính xác hơn
Random Forest	Độ chính xác cao, ít bị overfitting	Khó diễn giải, tốn nhiều tài nguyên tính toán, tham số chưa tối ưu
Support Vector Machine	Hiệu quả với dữ liệu chiều cao, linh hoạt với nhiều kernel	Cần chọn kernel và tham số cẩn thận, không có ước tính xác suất trực tiếp

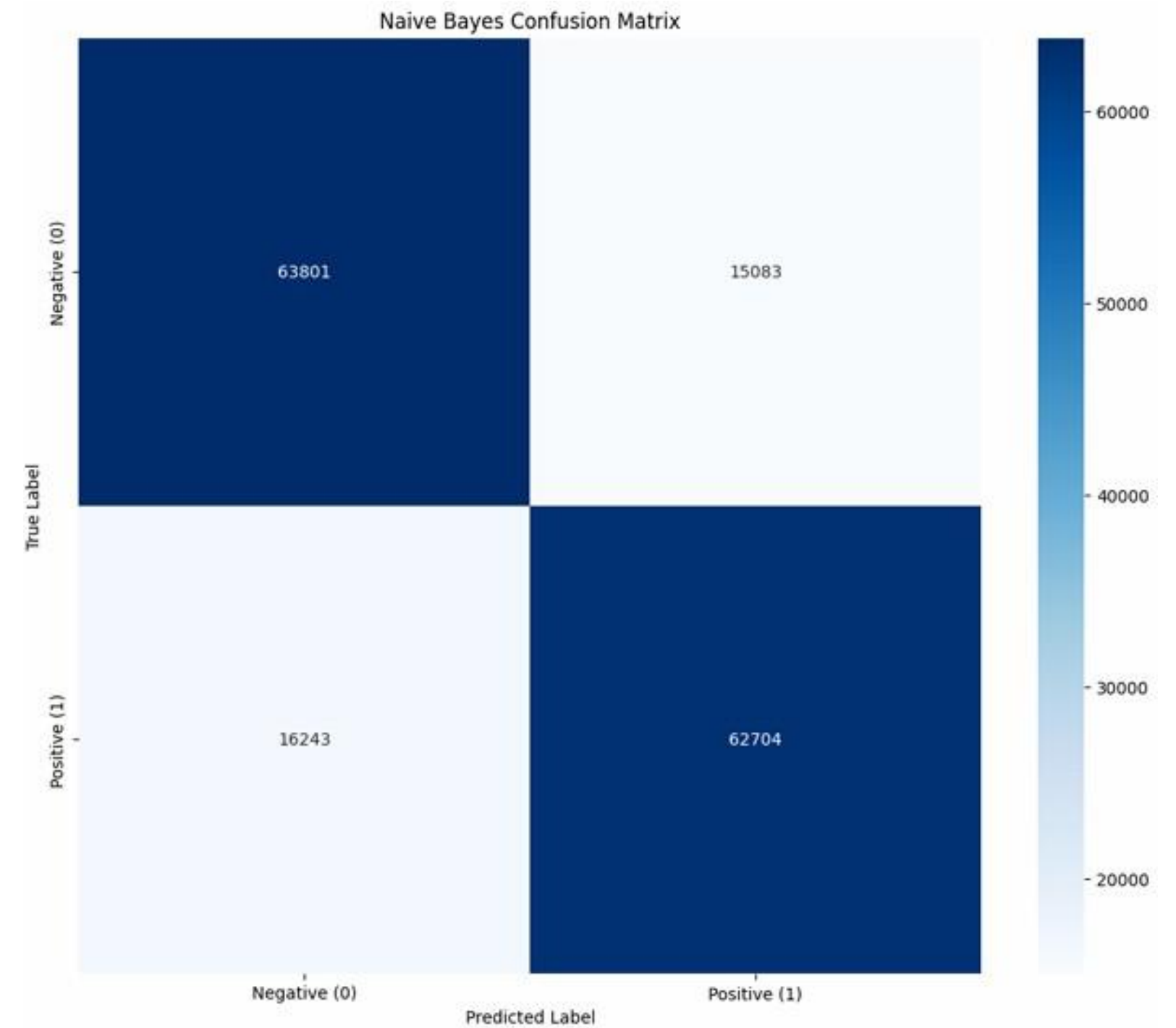


# Phân tích Confusion Matrix

Hình 5.1 hiển thị Confusion Matrix của mô hình Naive Bayes. Các giá trị trên đường chéo chính (từ trái sang phải) thể hiện số lượng các dự đoán đúng (True Positives và True Negatives), trong khi các giá trị ngoài đường chéo chính thể hiện số lượng các dự đoán sai (False Positives và False Negatives).

Nhận xét:

- Mô hình dự đoán đúng 63,801 tweets Negative (TN) và 62.704 tweets Positive (TP).
- Mô hình dự đoán sai 16.243 tweets Positive thành Negative (FN) và 15.083 tweets Negative thành Positive (FP).
- Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 79.43%.
- Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 80.88%.

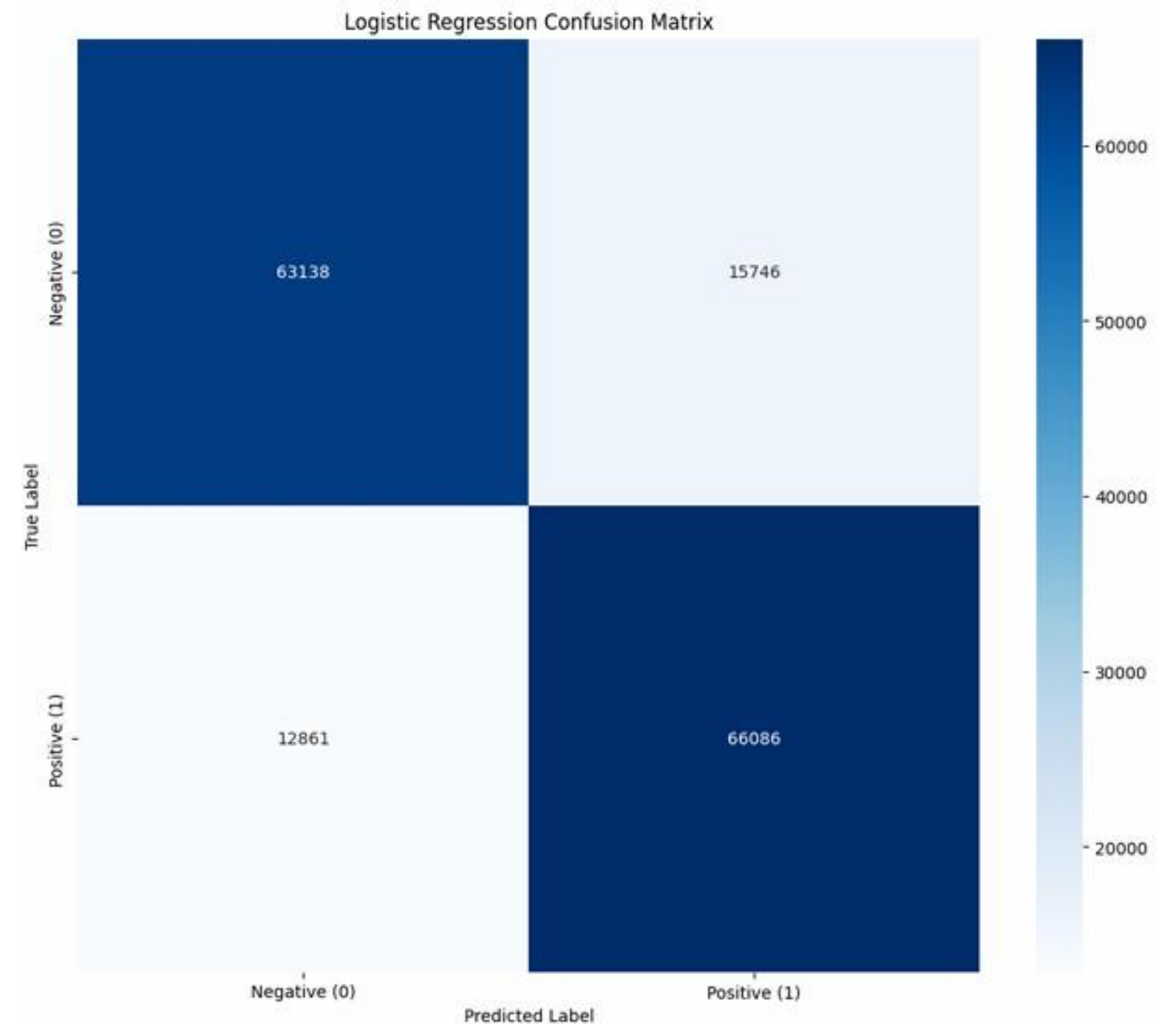


Hình 5.1: Confusion Matrix cho mô hình Naive Bayes

# Phân tích Confusion Matrix

Nhận xét:

- Mô hình dự đoán đúng 63,138 tweets Negative (TN) và 66,086 tweets Positive (TP).
- Mô hình dự đoán sai 12,861 tweets Positive thành Negative (FN) và 15,746 tweets Negative thành Positive (FP).
- Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 83.71%
- Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 80.04%.
- So với Naive Bayes, Logistic Regression dự đoán đúng ít trường hợp Negative hơn (TN thấp hơn) và đúng nhiều trường hợp Positive hơn (TP cao hơn). Tuy nhiên, tổng thể số lượng dự đoán đúng của Logistic Regression cao hơn Naive Bayes, dẫn đến Accuracy và F1-score cao hơn.

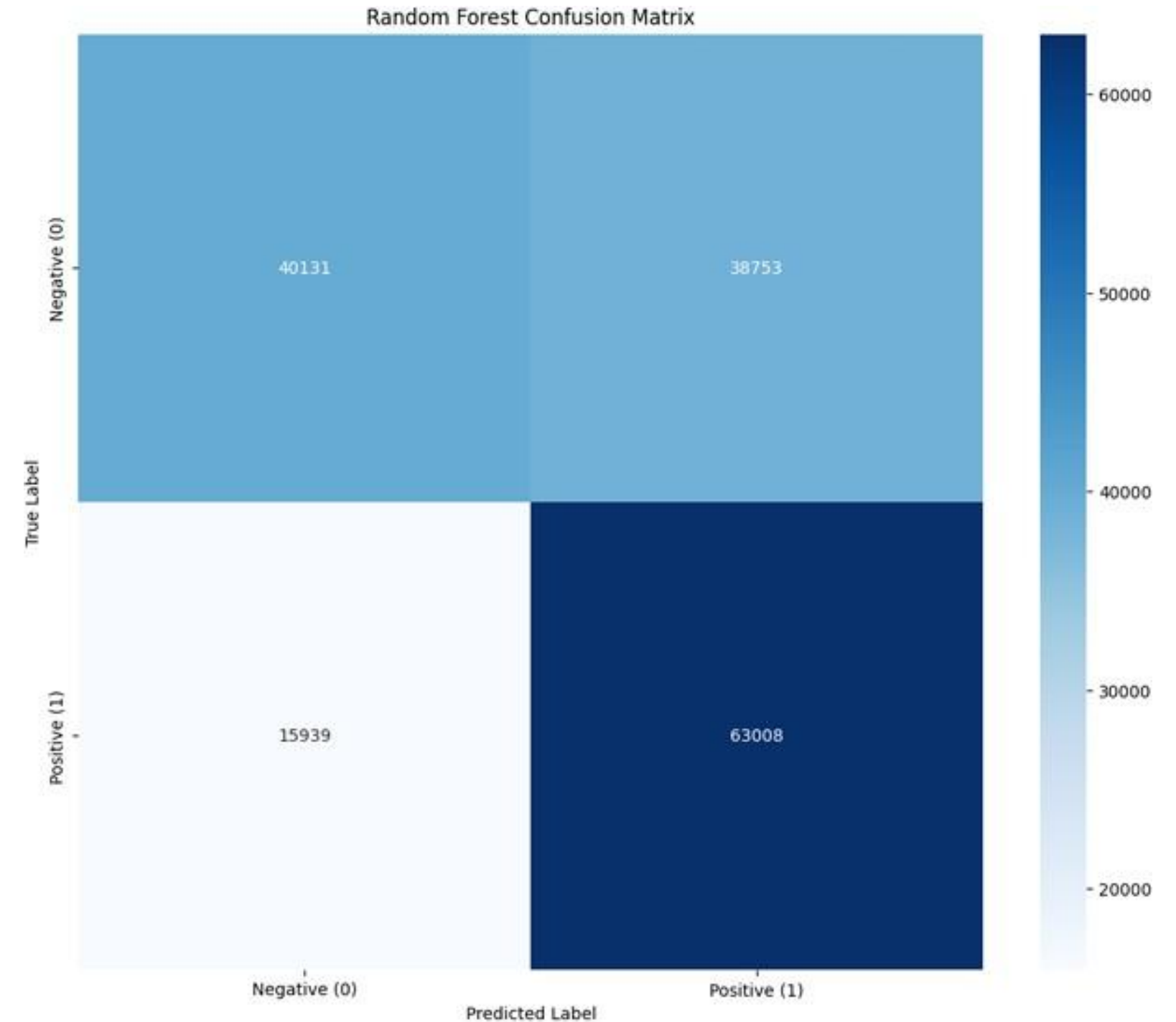


Hình 5.2: Confusion Matrix cho mô hình Logistic Regression

# Phân tích Confusion Matrix

Nhận xét:

- Mô hình dự đoán đúng 40,131 tweets Negative (TN) và 63,008 tweets Positive (TP).
- Mô hình dự đoán sai 15,939 tweets Positive thành Negative (FN) và 38,753 tweets Negative thành Positive (FP).
- Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 79.81%.
- Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 50.87%.
- Mô hình Random Forest có độ chính xác dự đoán các tweets Positive khá cao (TP cao), nhưng lại dự đoán các tweets Negative rất kém (TN thấp), dẫn đến hiệu suất tổng thể (Accuracy và F1-score) thấp hơn so với hai mô hình còn lại.

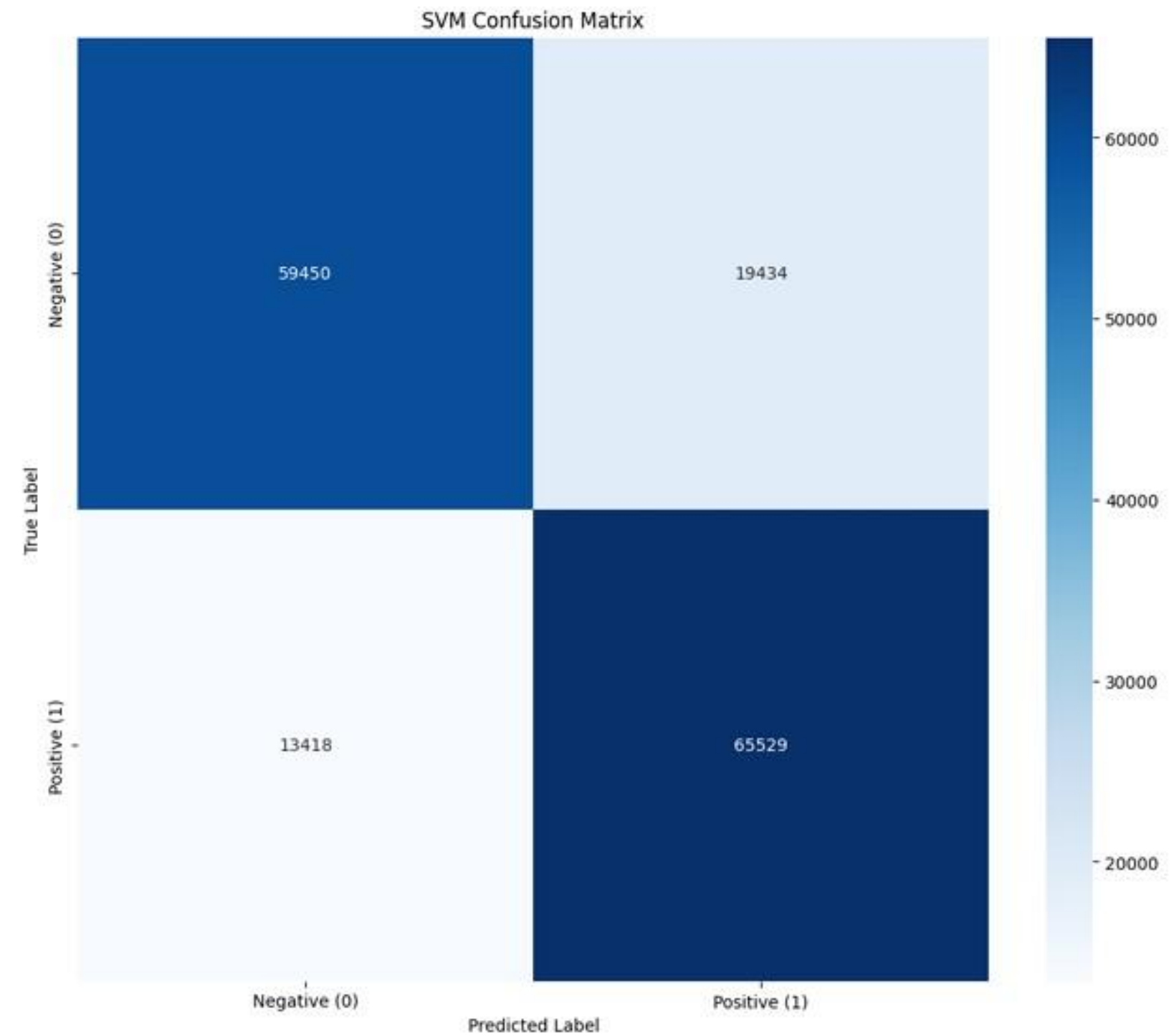


Hình 5.3: Confusion Matrix cho mô hình Random Forest

# Phân tích Confusion Matrix

Nhận xét:

- Mô hình dự đoán đúng 59,450 tweets Negative (TN) và 65,529 tweets Positive (TP).
- ■ Mô hình dự đoán sai 13,418 tweets Positive thành Negative (FN) và 19,434 tweets Negative thành Positive (FP).
- ■ Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 83.00%.
- ■ Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 75.36%.
- ■ Mô hình SVM có độ chính xác dự đoán các tweets Positive khá cao (TP cao) và chỉ kém Logistic Regression một chút mà thôi, dự đoán các tweets Negative (TN) không quá cao, thấp hơn so với Naive Bayes và Logistic Regression nhưng cao hơn Random Forest, nhìn chung thì hiệu suất tổng thể (Accuracy và F1-score) chỉ đứng sau Naive Bayes một chút thôi



Hình 5.4: Confusion Matrix cho mô hình SVM



## So sánh hiệu suất các mô hình

Mô hình	Accuracy	F1-score	TPR (Sensitivity)	TNR (Specificity)	Lưu ý
Logistic Regression	0.8187	0.8187	83.71%	80.04%	Cân bằng tốt giữa Positive và Negative.
Naive Bayes	0.8015	0.8015	79.43%	80.88%	Nhầm nhiều tweets Negative thành Positive (FP = 15,083).
SVM	0.7919	0.7915	83.00%	75.36%	Dự đoán Positive tốt nhưng kém ở Negative.
Random Forest	0.6535	0.6461	79.81%	50.87%	Dự đoán Negative kém, dẫn đến hiệu suất thấp.



# Content

06

**Spark NLP và Deep Learning**

# Triển khai mô hình

- Để tận dụng sức mạnh của Deep Learning trong bài toán phân tích cảm xúc, chúng ta sử dụng 'ClassifierDLApproach' của Spark NLP, kết hợp với 'BertEmbeddings' để biểu diễn từ.

## Chuẩn bị dữ liệu:

- Do hạn chế về tài nguyên tính toán, chúng ta tiến hành lấy mẫu từ tập dữ liệu gốc để thu được tập dữ liệu cân bằng hơn, bao gồm 200,000 tweets với tỷ lệ số lượng mẫu positive/negative là 1/1 (mỗi loại 100,000 tweets). Các bước thực hiện như sau:
  - 1. Lấy mẫu dữ liệu: Từ tập dữ liệu đã qua tiền xử lý, lấy mẫu 100,000 tweets tích cực ('Sentiment' = 1) và 100,000 tweets tiêu cực ('Sentiment' = 0).
  - 2. Kết hợp dữ liệu: Kết hợp hai tập dữ liệu positive và negative thành một tập dữ liệu cân bằng duy nhất.
  - 3. Chia dữ liệu: Chia tập dữ liệu cân bằng thành hai tập: tập huấn luyện (80%) và tập kiểm tra (20%) bằng cách sử dụng phương thức 'randomSplit()'.

# Xây dựng Pipeline

Mô hình 'ClassifierDLApproach' được triển khai trong pipeline của Spark MLlib để đảm bảo quy trình nhất quán trong tiền xử lý và trích xuất đặc trưng. Các bước chính trong pipeline gồm:

- DocumentAssembler: Chuyển đổi cột 'SentimentText' thành định dạng 'document', chuẩn bị cho các bước xử lý tiếp theo.
- Tokenizer: Tách các câu trong 'document' thành các token (từ).
- BertEmbeddings: Sử dụng mô hình 'small\_bert\_L4\_256' đã huấn luyện trước để tạo vector biểu diễn cho từng từ (token).
- SentenceEmbeddings: Kết hợp các word embeddings thành sentence embedding bằng phương pháp trung bình cộng.
- ClassifierDLApproach: Huấn luyện mô hình phân loại cảm xúc dựa trên sentence embeddings. Các tham số quan trọng gồm:
  - 'maxEpochs': Số lượng epochs tối đa.
  - 'lr': learning rate.
  - 'batchSize': Kích thước batch.
  - 'validationSplit': Tỷ lệ dữ liệu validation.
  - 'dropout': Tỷ lệ dropout, giúp tránh overfitting.
  - 'setEnableOutputLogs(True)': Bật in ra thông tin training log.
  - 'setOutputLogsPath('logs')': Thiết lập đường dẫn lưu log, thuận tiện cho việc theo dõi quá trình train.

# Kết quả và So sánh

Bảng 6.1 tổng hợp kết quả đánh giá các mô hình, bao gồm cả 4 mô hình đã triển khai ở phần trước (Naive Bayes, Logistic Regression, Random Forest, SVM) và mô hình 'ClassifierDLApproach' với 'BertEmbeddings' mới được huấn luyện.

	Precision	Recall	F1-score	Support
<b>0</b>	0.73	0.80	0.76	18171
<b>1</b>	0.82	0.75	0.79	22046
<b>Accuracy</b>			0.78	40217
<b>Macro avg</b>	0.78	0.78	0.78	40217
<b>Weighted avg</b>	0.78	0.78	0.78	40217

**Bảng 6.2:** Classification Report

Mô hình	Accuracy	F1-score
Naive Bayes	0.8015	0.8015
Logistic Regression	0.8187	0.8187
Random Forest	0.6535	0.6461
Support Vector Machine	0.7919	0.7915
ClassifierDL (Bert Embeddings)	0.78	0.78

**Bảng 6.1:** Kết quả đánh giá các mô hình

# Kết quả và So sánh

## Nhận xét

- Mô hình đạt precision 0.73 cho lớp 0 (Negative) và 0.82 cho lớp 1 (Positive), recall 0.80 cho lớp 0 và 0.75 cho lớp 1.
- F1-score cho lớp 0 là 0.76 và cho lớp 1 là 0.79.
- Kết quả cho thấy mô hình dự đoán lớp 1 (Positive) tốt hơn một chút so với lớp 0 (Negative).

## So sánh với các mô hình khác

- So với các mô hình truyền thống (Logistic Regression, Naive Bayes, SVM, Random Forest), mô hình ClassifierDLApproach với BertEmbeddings cho kết quả ở mức khá tốt, gần bằng SVM.
- Mặc dù chưa đạt được hiệu suất cao nhất, nhưng việc sử dụng BertEmbeddings đã cho thấy tiềm năng trong việc cải thiện độ chính xác của mô hình phân tích cảm xúc. BertEmbeddings có khả năng nắm bắt ngữ nghĩa và ngữ cảnh tốt hơn so với CountVectorizer và IDF, dẫn đến các vector biểu diễn câu có chất lượng cao hơn.



- Mô hình DeepLearning ClassifierDLApproach với BertEmbeddings cho kết quả khả quan trong bài toán phân tích cảm xúc trên tập dữ liệu tweets, đạt hiệu suất gần bằng với SVM.
- Việc sử dụng pre-trained BertEmbeddings giúp cải thiện khả năng biểu diễn đặc trưng cho văn bản, dẫn đến kết quả phân loại tốt hơn.
- Cần thử nghiệm thêm với việc tinh chỉnh tham số, thay đổi kiến trúc mạng, và sử dụng các pre-trained embeddings khác để có thể cải thiện hơn nữa hiệu suất của mô hình.
- Việc in ra các thông số loss và accuracy trong quá trình training giúp ích cho việc điều chỉnh các tham số.

# Content

07

**Hadoop MapReduce**

# Triển khai và Cài đặt

Mô hình Naive Bayes được triển khai trên Hadoop MapReduce, bao gồm hai giai đoạn chính: Training (huấn luyện) và Testing (kiểm thử):

- Giai đoạn Training: Giai đoạn này sử dụng một job MapReduce để tính toán tần suất xuất hiện của các từ trong từng loại cảm xúc (Positive và Negative).
- Giai đoạn Testing: Giai đoạn này sử dụng một job MapReduce để dự đoán cảm xúc của các tweets trong tập kiểm tra

# Kết quả

- Kết quả từ hai lần chạy thử nghiệm với 100,000 tweets và 200,000 tweets (75% train và 25% test) như sau:

Lần chạy	Confusion Matrix (TN, FP, FN, TP)	Accuracy	Thời gian thực thi (giây)
Lần 1	TN = 11,793, FP = 3,005, FN = 2,973, TP = 7,229	0.76088	258.7057
Lần 2	TN = 23,780, FP = 6,102, FN = 5,765, TP = 14,353	0.76266	768.74725

## Nhận xét:

- Mô hình NaiveBayes triển khai trên Hadoop MapReduce đạt được độ chính xác 0.76088 trong lần thử nghiệm đầu tiên, cho thấy tiềm năng trong việc xử lý dữ liệu lớn.
- Hadoop MapReduce cho phép xử lý tập dữ liệu lớn hiệu quả, tận dụng khả năng tính toán song song và phân tán.
- Thời gian thực thi cho thấy MapReduce chạy khá tốn thời gian, đặc biệt là với tập dữ liệu lớn (Với tập dữ liệu gồm 1,000,000 tweets có thể tốn thời gian gấp khoảng 80 lần tập dữ liệu 100,000 tweets)

# Content

08

**Kết luận**



# Kết luận và Hướng Phát Triển

1

## Kết luận

Logistic Regression đạt hiệu suất cao nhất, Naive Bayes và SVM cũng cho thấy kết quả khả quan. Mô hình Deep Learning với BertEmbeddings có tiềm năng, nhưng chưa đạt được hiệu suất tối ưu.

2

## Hạn chế

Dự án còn tồn tại một số hạn chế, bao gồm tập dữ liệu chưa đủ đa dạng, biểu diễn đặc trưng chưa tối ưu, và các tham số mô hình chưa được tinh chỉnh hoàn toàn.

3

## Hướng Phát Triển

Thử nghiệm trên các tập dữ liệu khác, cải tiến biểu diễn đặc trưng, tinh chỉnh tham số, nghiên cứu các mô hình Deep Learning tiên tiến hơn, và xây dựng hệ thống phân tích cảm xúc thời gian thực.

**Thank you  
very much!**

---