

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA HÀ NỘI



TIỂU LUẬN CUỐI KỲ
MÔN HỌC: HỌC MÁY

ĐỀ TÀI

Phân tích cảm xúc (Sentiment Analysis)

GV hướng dẫn: HOÀNG TUẤN ANH

Nhóm sinh viên thực hiện:

Họ và tên

Nguyễn Văn Long

Trương Đan Vi

Nguyễn Quang Việt

MSSV

22001609

22001655

22001659

Hà Nội, 2025

Mục lục

| | | |
|----------|--|-----------|
| 1 | Giới thiệu | 2 |
| 1.1 | Đặt vấn đề | 2 |
| 1.2 | Mục tiêu dự án | 2 |
| 1.3 | Phạm vi dự án | 3 |
| 2 | Thu thập dữ liệu | 4 |
| 2.1 | Nguồn dữ liệu | 4 |
| 2.2 | Mô tả dữ liệu | 4 |
| 3 | Tiền xử lí dữ liệu | 5 |
| 3.1 | Làm sạch dữ liệu | 5 |
| 3.2 | Feature Engineering | 7 |
| 4 | Lựa chọn mô hình và huấn luyện | 9 |
| 4.1 | Lựa chọn mô hình | 9 |
| 4.2 | Huấn luyện mô hình | 15 |
| 5 | Kết quả | 18 |
| 5.1 | Các thông số dùng để đánh giá mô hình | 18 |
| 5.2 | Kết quả | 19 |
| 5.3 | Bàn luận | 22 |
| 6 | Kết luận và bài học kinh nghiệm | 25 |
| 6.1 | Tóm tắt kết quả | 25 |
| 6.2 | Hạn chế | 25 |
| 6.3 | Bài học | 26 |
| 6.4 | Tiềm năng trong tương lai | 26 |
| 7 | Phân công công việc | 28 |
| 7.1 | Đóng góp của từng thành viên | 28 |
| 7.2 | Sự hợp tác trong quá trình thực hiện dự án | 28 |
| 8 | Tài liệu tham khảo | 29 |
| A | Apendices | 30 |

1 Giới thiệu

1.1 Đặt vấn đề

Phân tích quan điểm (Sentiment Analysis) là một ứng dụng của trí tuệ nhân tạo, nó sử dụng các thuật toán phức tạp để xử lý ngôn ngữ tự nhiên của con người (NLP) và xác định các đặc điểm cảm xúc tiêu cực/tích cực tại một thời điểm thông qua văn bản hoặc lời nói. Các nguồn dữ liệu được phân tích phổ biến như Social media, Blog, Website đánh giá sản phẩm, tổng đài Contact center,...

Trong những năm gần đây, với sự phát triển mạnh mẽ và vượt bậc của Internet, nhu cầu tham khảo của các khách hàng trước khi mua sắm trực tuyến ngày càng gia tăng. Vậy nên những trang mạng hiện nay được phát triển cho phép người dùng có thể chia sẻ những đánh giá, nhận xét và phản hồi về các loại hình sản phẩm hay dịch vụ của những công ty, doanh nghiệp khác nhau.

Các công ty hiện nay đang bắt đầu hướng tới việc nghiên cứu các nền tảng truyền thông xã hội bằng các công cụ để hiểu được mong muốn của khách hàng, nhằm mục đích cải thiện và phát triển sản phẩm hay dịch vụ họ cung cấp. Là một phần quan trọng trong hoạt động này, phân tích văn bản đã và đang trở thành một mảng nghiên cứu sôi nổi trong ngôn ngữ học tính toán (Computational Linguistics) và xử lý ngôn ngữ tự nhiên (Natural Language Processing).

Một trong những vấn đề phổ biến nhất trong hoạt động nói trên đó là phân loại văn bản (TextClassification), một công việc nhằm để chia các tài liệu thành một hoặc nhiều các nhóm bằng cách thủ công hoặc tính toán. Với hướng nghiên cứu này, trong vài năm gần đây chúng ta có thể thấy xu hướng được ưa chuộng đó là phân loại cảm xúc (Sentiments) từ các bình luận trên các nền tảng Social Media, các trang đánh giá (Review Sites) và các nhóm thảo luận. Công việc này gọi là phân tích cảm xúc (Sentiment Analysis), một quá trình tính toán mà sử dụng số liệu thống kê và kỹ thuật xử lý ngôn ngữ tự nhiên để nhận dạng và phân loại ý kiến được bày tỏ trong văn bản, cụ thể hơn là xác định các phân cực thái độ (tích cực, tiêu cực và trung lập) của người viết đối với chủ đề hoặc sản phẩm được nhắc tới. Phân loại thái độ đã và đang được sử dụng rộng rãi ở nhiều doanh nghiệp để doanh nghiệp hiểu hơn về khách hàng của họ thông qua các hệ thống hỗ trợ khách hàng hoặc thông qua các bảng đánh giá online của khách hàng đối với sản phẩm hoặc dịch vụ của họ.

1.2 Mục tiêu dự án

Xây dựng một số mô hình có thể giải quyết bài toán phân lớp cảm xúc của người dùng thông qua các bình luận, đánh giá (các tweet) dưới dạng xác định tính tích cực - tiêu cực - bình thường (hoặc không liên quan) của văn bản. Việc phân loại này sẽ giúp doanh nghiệp nắm bắt nhanh chóng tâm trạng người tiêu dùng và còn có thể

làm cơ sở cho việc cải thiện chất lượng dịch vụ, sản phẩm và chiến lược marketing. Các bình luận và đánh giá từ người dùng, dù là tích cực hay tiêu cực, đều mang đến những thông tin quý giá để các tổ chức hiểu và đáp ứng nhu cầu của khách hàng một cách hiệu quả hơn.

Hơn hết mục tiêu của dự án là xây dựng một hệ thống ứng dụng có thể triển khai thực tế, giúp các doanh nghiệp, tổ chức hiểu rõ hơn về cảm xúc của người tiêu dùng đối với sản phẩm, dịch vụ hoặc các vấn đề xã hội đang được bàn luận. Hệ thống này không chỉ giúp tăng cường trải nghiệm khách hàng mà còn hỗ trợ doanh nghiệp trong việc đưa ra các quyết định chiến lược dựa trên phân tích cảm xúc từ dữ liệu lớn

1.3 Phạm vi dự án

Phạm vi của dự án sẽ bao gồm việc thu thập dữ liệu từ các nền tảng trực tuyến phổ biến như Twitter (nay là X) Các dữ liệu này chủ yếu bao gồm các bình luận, tweet, bài viết hoặc nhận xét của người dùng liên quan đến các sản phẩm, dịch vụ, sự kiện hoặc chủ đề cụ thể. Dữ liệu thu thập sẽ được tiền xử lý để loại bỏ các yếu tố không cần thiết, như thông tin không liên quan, từ viết tắt hoặc ký hiệu đặc biệt.

Dự án tập trung vào việc phát triển các mô hình học máy để phân loại cảm xúc trong văn bản. Các mô hình sẽ xác định thái độ của người viết đối với sản phẩm hoặc chủ đề cụ thể thông qua ba lớp cảm xúc chính: tích cực, tiêu cực và trung lập.

Dự án sẽ áp dụng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP) và các phương pháp học máy để xây dựng mô hình phân loại cảm xúc. Cụ thể, các mô hình học máy như Naive Bayes, Support Vector Machines (SVM) , Cây láng giềng gần nhất (KNN) , Decision Tree , Random Forest , Logistic Regression

2 Thu thập dữ liệu

2.1 Nguồn dữ liệu

Bộ dữ liệu "Twitter Entity Sentiment Analysis" trên Kaggle là một tập dữ liệu phân tích cảm xúc từ các tweet trên Twitter. Mục tiêu của bộ dữ liệu này là xác định cảm xúc của toàn bộ câu đối với một thực thể cụ thể. Ví dụ, câu "A outperforms B" sẽ được coi là tích cực đối với thực thể A nhưng tiêu cực đối với thực thể B.

Bộ dữ liệu này được thiết kế để huấn luyện và đánh giá các mô hình phân tích cảm xúc, giúp xác định cảm xúc của người dùng đối với các thực thể cụ thể trong văn bản. Điều này đặc biệt hữu ích trong việc theo dõi phản hồi của công chúng đối với các thương hiệu, sản phẩm hoặc cá nhân trên mạng xã hội.

2.2 Mô tả dữ liệu

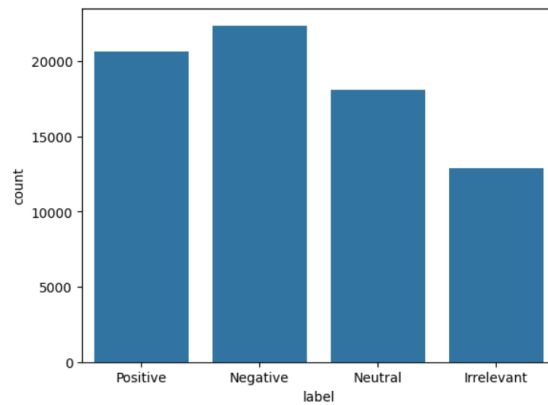
Bộ dữ liệu bao gồm 2 tệp tin `twitter_training.csv` và `twitter_validation.csv` trong đó tệp `twitter_training.csv` có khoảng 70 000 tweet dùng để huấn luyện mô hình và tệp `twitter_validation.csv` có khoảng 1000 tweet được dùng để kiểm tra mô hình sau khi đã được huấn luyện.

Bộ dữ liệu bao gồm những thuộc tính sau:

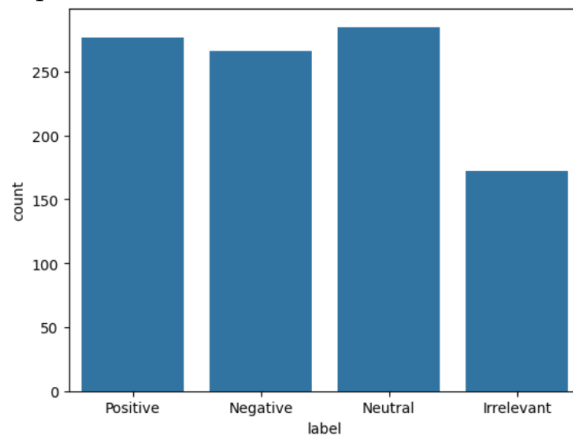
- **Tweet_ID**: ID duy nhất của mỗi tweet.
- **Topic** (Thực thể): Thực thể hoặc chủ đề mà tweet đề cập đến.
- **Sentiment** (Cảm xúc): Nhãn cảm xúc của tweet đối với thực thể, bao gồm các lớp: Tích cực, Tiêu cực, Trung lập và Không liên quan.
- **Tweet**: Nội dung văn bản của tweet.

Ta có biểu đồ mô tả phân bố của thuộc tính Sentiment trong tập huấn luyện và tập kiểm tra như sau:

Từ đồ thị trên, có thể thấy bộ dữ liệu có sự cân bằng, sự chênh lệch giữa các đặc trưng trong thuộc tính là không quá lớn.



Hình 1: Biểu đồ phân bố của thuộc tính Sentiment trong tập huấn luyện



Hình 2: Biểu đồ phân bố của thuộc tính Sentiment trong tập kiểm tra

3 Tiền xử lí dữ liệu

3.1 Làm sạch dữ liệu

Dữ liệu được tải từ các tệp CSV cho tập huấn luyện và tập kiểm tra bằng cách sử dụng thư viện `pandas`. Quá trình làm sạch dữ liệu gồm 2 công đoạn chính, lần lượt là:

- Làm sạch dữ liệu trống
- Mã hoá nhãn mục tiêu đầu ra

Làm sạch dữ liệu trống

Sau khi tải dữ liệu, chúng ta sử dụng phương thức `dropna()` để loại bỏ các dòng có giá trị bị thiếu (NaN). Sau khi làm sạch dữ liệu, thông tin về tập huấn luyện và tập kiểm tra được in ra bằng phương thức `info()` của pandas để kiểm tra lại tính đầy đủ của dữ liệu.

```
df_train = load_df('/content/twitter_training.csv')
df_valid = load_df('/content/twitter_validation.csv')
df_train.dropna(inplace=True)
df_valid.dropna(inplace=True)
```

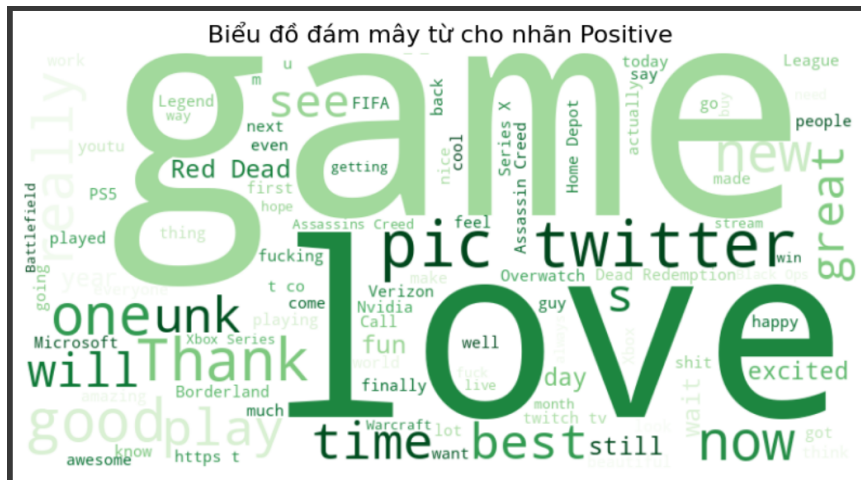
Mã hóa Nhãn

Do các mô hình học máy không thể xử lý trực tiếp các nhãn dạng chuỗi, chúng ta sử dụng `LabelEncoder` từ thư viện `scikit-learn` để mã hóa các nhãn thành dạng số. Các nhãn được mã hóa thành các giá trị số nguyên từ 0 đến 2 ứng với các nhãn {Positive, Negative, Neutral, Irrelevant}. (Ta quy ước nhãn Neutral và Irrelevant cùng ứng với giá trị là 2)

```
enc = LabelEncoder()
y_train = enc.fit_transform(df_train['label'])
y_test = enc.transform(df_valid['label'])
```

Sau khi làm sạch dữ liệu, ta tạo đám mây từ (word clouds) cho từng nhãn cảm xúc trên các dòng tweet đã được làm sạch.

Ta có biểu đồ đám mây từ cho nhãn Positive và Negative như sau:



Từ biểu đồ trên ta thấy một số từ đặc trưng cho các nhãn là:

- Nhân Positive: "love", "great", "best"
- Nhân Negative: "shit", "fucking", "fuck"

3.2 Feature Engineering

Văn bản trong cột `text` của dữ liệu được xử lý qua một pipeline bao gồm hai bước chính:

1. **CountVectorizer**: Biến đổi văn bản thành ma trận tần suất từ vựng, trong đó mỗi từ trong văn bản được biểu diễn bằng một chỉ số. Tham số `stop_words='english'` giúp loại bỏ các từ dừng (stop words) trong tiếng Anh, như "and", "the", "is", v.v.
2. **TfidfTransformer**: Chuyển đổi ma trận tần suất từ thành ma trận TF-IDF (Term Frequency-Inverse Document Frequency). Điều này giúp giảm trọng số của các từ xuất hiện phổ biến và tăng trọng số của các từ đặc trưng trong từng tài liệu.

Pipeline này giúp tối ưu hóa việc xử lý dữ liệu văn bản, đồng thời làm cho mô hình học máy có thể làm việc hiệu quả hơn với dữ liệu văn bản.

Dữ liệu huấn luyện và kiểm tra sau khi xử lý được lưu trữ trong các biến `X_train_transformed` và `X_test_transformed`, và thông tin về kích thước của ma trận đầu vào được in ra để kiểm tra.

```
text_pipeline = make_pipeline(  
    CountVectorizer(stop_words='english'),  
    TfidfTransformer()  
)  
X_train_transformed = text_pipeline.fit_transform(df_train['text'])  
X_test_transformed = text_pipeline.transform(df_valid['text'])  
  
print(f"Shape of the transformed training data: {X_train_transformed.shape}")  
print(f"Shape of the transformed test data: {X_test_transformed.shape}")
```

Kết luận

Sau khi hoàn thành quá trình tiền xử lý, dữ liệu đã sẵn sàng để huấn luyện các mô hình học máy. Các bước bao gồm việc làm sạch dữ liệu và xử lý văn bản. Các phương pháp này giúp chuẩn bị dữ liệu một cách đầy đủ và hợp lý để có thể xây dựng các mô hình phân loại hiệu quả hơn.

4 Lựa chọn mô hình và huấn luyện

4.1 Lựa chọn mô hình

Mô hình Naive Bayes

Naive Bayes Classification(NBC) là 1 thuật toán dựa trên định lí Bayes về lý thuyết xác suất để đưa ra các phán đoán cũng như phân loại dữ liệu dựa trên các dữ liệu được quan sát và thống kê .

Định lý Bayes được biểu diễn dưới dạng công thức như sau:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Trong đó:

- $P(C|X)$: Xác suất hậu nghiệm (*posterior probability*) – xác suất để X thuộc lớp C .
- $P(X|C)$: Xác suất có điều kiện (*likelihood*) – xác suất để X xảy ra khi biết lớp C .
- $P(C)$: Xác suất tiên nghiệm (*prior probability*) – xác suất xảy ra của lớp C .
- $P(X)$: Xác suất xảy ra của X – đóng vai trò là hằng số trong bài toán phân loại.

Ta gọi định lí này là "Naïve" (ngây thơ) vì nó giả định rằng các yếu tố dự báo trong mô hình Naïve Bayes là độc lập có điều kiện hoặc không liên quan đến bất kỳ đặc trưng nào khác trong mô hình. Nó cũng giả định rằng tất cả các đặc trưng đều đóng góp như nhau vào kết quả. Mặc dù các giả định này thường bị vi phạm trong các tình huống thực tế (ví dụ: một từ tiếp theo trong email phụ thuộc vào từ trước đó), nhưng nó đơn giản hóa vấn đề phân loại bằng cách làm cho nó dễ tính toán hơn. Nghĩa là, bây giờ chỉ cần một xác suất duy nhất cho mỗi biến, điều này giúp cho việc tính toán mô hình dễ dàng hơn. Bất chấp giả định độc lập không thực tế này, thuật toán phân loại hoạt động tốt, đặc biệt là với kích thước mẫu nhỏ.

Nhờ giả định này, xác suất $P(X|C)$ có thể được tính như sau:

$$P(X|C) = P(X_1|C) \cdot P(X_2|C) \cdot \dots \cdot P(X_n|C)$$

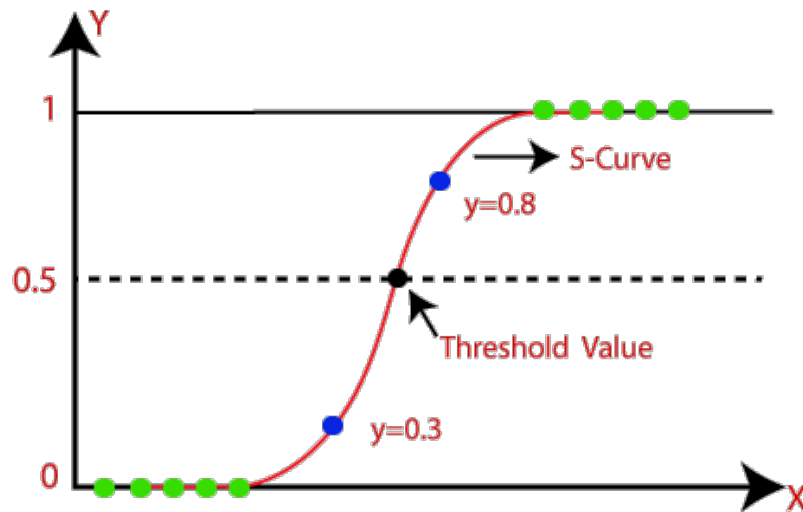
Các phân phối $p(C)$ và $p(X_i|C), i = 1, \dots, d$ sẽ được xác định dựa vào dữ liệu huấn

luyện. Việc xác định các giá trị này có thể dựa vào MLE hoặc MAP.

Việc tính toán $p(x_i|c)$ phụ thuộc vào loại dữ liệu. Có ba loại phân bố xác suất thường được dùng: *Gaussian Naive Bayes*, *Multinomial Naive Bayes*, và *Bernoulli Naive Bayes*.

Mô hình Logistic Regression

Hồi quy Logistic là một mô hình thống kê được sử dụng để phân loại nhị phân, tức dự đoán một đối tượng thuộc vào một trong hai nhóm. Hồi quy Logistic làm việc dựa trên nguyên tắc của hàm sigmoid – một hàm phi tuyến tự chuyển đầu vào của nó thành xác suất thuộc về một trong hai lớp nhị phân.



Hình 5: Mô hình hồi quy Logistic

Hồi quy Logistic hoạt động dựa trên hàm Sigmoid, được biểu diễn như sau:

$$S(z) = \frac{1}{1 + e^{-z}}$$

Hàm Sigmoid nhận đầu vào là một giá trị z bất kỳ, và trả về đầu ra là một giá trị xác suất nằm trong khoảng $[0; 1]$. Khi áp dụng vào mô hình Hồi quy Logistic với đầu vào là ma trận dữ liệu X và trọng số w , ta có $z = Xw$

Việc huấn luyện của mô hình là tìm ra bộ trọng số w sao cho đầu ra dự đoán của hàm Sigmoid gần với kết quả thực tế nhất. Để làm được điều này, ta sử dụng hàm mất mát (Loss Function) để đánh giá hiệu năng của mô hình. Mô hình càng tốt khi hàm mất mát càng nhỏ.

Hàm mất mát (Loss Function) là một hàm số được sử dụng để đo lường mức độ lỗi mà mô hình của chúng ta tạo ra khi dự đoán các kết quả từ dữ liệu đầu vào. Trong bài toán Hồi quy Logistic, chúng ta sử dụng hàm mất mát Cross-Entropy (còn gọi là Log Loss) để đánh giá hiệu năng của mô hình.

Hàm mất mát Cross-Entropy được định nghĩa như sau:

$$L(w) = \frac{1}{n} \sum_{i=1}^n y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

Trong đó:

- n : số lượng mẫu dữ liệu trong tập huấn luyện.
- y_i : giá trị thực tế của đầu ra thứ i .
- p_i : xác suất dự đoán thuộc lớp 1 của mô hình cho đầu vào thứ i .

Hàm Cross-Entropy đo lường khoảng cách giữa hai phân phối xác suất y_i và p_i . Khi mô hình dự đoán chính xác, tức là nếu $y_i = 1$ thì p_i càng gần 1, và nếu $y_i = 0$ thì p_i càng gần 0, sau đó hàm mất mát sẽ tiến gần về 0.

Trong quá trình huấn luyện, chúng ta tìm cách cập nhật bộ trọng số w sao cho giá trị hàm mất mát Cross-Entropy đạt giá trị nhỏ nhất, dẫn đến một mô hình dự đoán tốt nhất.

Để tìm giá trị tối ưu cho bộ trọng số w , chúng ta có thể sử dụng kỹ thuật Gradient Descent. Tại mỗi bước lặp, chúng ta cập nhật w theo phương hướng ứng với đạo hàm của hàm mất mát $L(w)$ theo w .

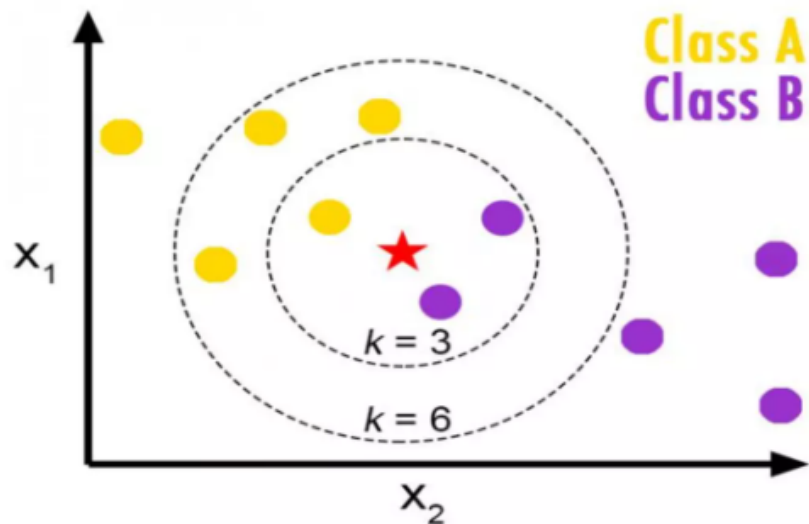
Mô hình K-hàng xóm gần nhất KNN

K-Nearest Neighbor (KNN) là một thuật toán phổ biến để phân loại văn bản. Thuật toán KNN với phương pháp TF-IDF và framework để phân loại văn bản, dựa vào khoảng cách gần nhất giữa đối tượng cần xếp lớp (Query point) và tất cả các đối tượng trong bộ dữ liệu huấn luyện. Nó cho phép phân loại theo các tham số khác nhau, đo lường, phân tích kết quả.

Với KNN, trong bài toán Classification, nhãn của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Nhãn của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho các điểm gần nhất đó rồi suy ra nhãn.

Một đối tượng được phân lớp dựa vào K láng giềng của nó. K là số nguyên dương được xác định trước khi thực hiện thuật toán. Người ta thường dùng khoảng cách Euclidean để tính khoảng cách giữa các đối tượng. Trong hình dưới đây, training data

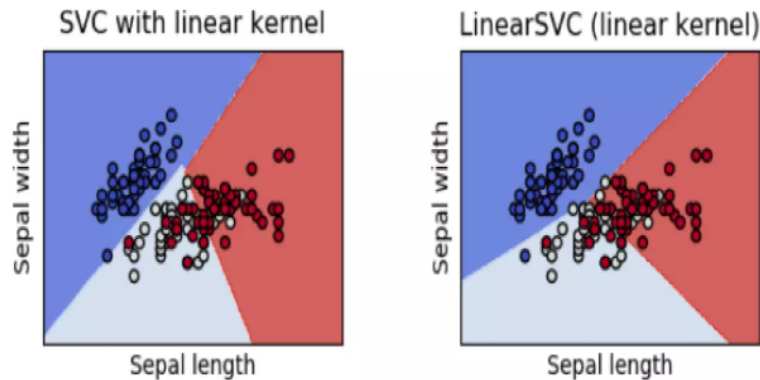
được mô tả bởi dấu (+) và dấu (-), đối tượng cần được xác định lớp cho nó (Query point) là hình tròn đỏ. Nhiệm vụ của chúng ta là ước lượng (hay dự đoán) lớp của Query point dựa vào việc lựa chọn số láng giềng gần nhất với nó. Nói cách khác chúng ta muốn biết liệu Query Point sẽ được phân vào lớp (+) hay lớp (-).



Hình 6: Mô hình K-Hàng xóm gần nhất KNN

Mô hình Support Vector Machine (SVM)

Support Vector Machine (SVM) là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồ thị dữ liệu là các điểm trong n chiều (ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó thuật toán SVM tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân chia dữ liệu thành các lớp. Siêu phẳng này có thể là một đường thẳng trong không gian 2D, một mặt phẳng trong không gian 3D hoặc một siêu phẳng trong không gian đa chiều. Mục tiêu của SVM là tìm ra siêu phẳng với margin lớn nhất, tức là khoảng cách từ các điểm dữ liệu gần nhất (gọi là các "Support Vectors") đến siêu phẳng này.



Hình 7: Mô hình Support Vector Machine - Linear

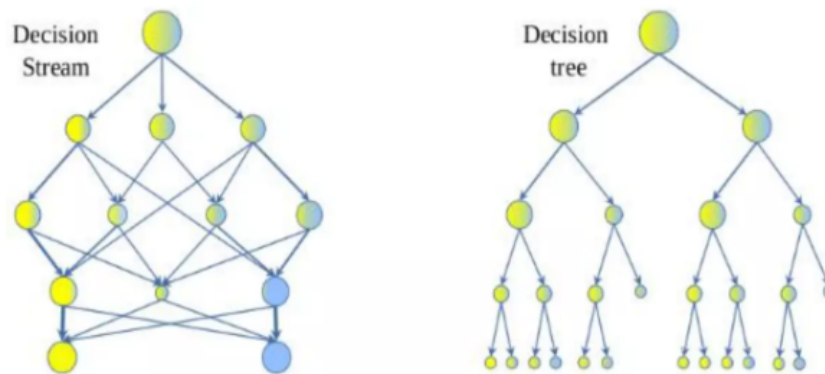
Ưu điểm của nó là: xử lý trên không gian có số chiều cao, tiết kiệm bộ nhớ, tính linh hoạt. Bên cạnh đó cũng có một số nhược điểm nhỏ như: gặp khó khăn với bài toán số chiều cao, chưa thể hiện rõ tính xác xuất.

Trong trường hợp dữ liệu không thể phân chia bằng một siêu phẳng tuyến tính, SVM sử dụng kỹ thuật kernel trick để chuyển đổi không gian đặc trưng thành không gian có chiều cao hơn, nơi mà phân chia tuyến tính có thể thực hiện được. Các hàm kernel phổ biến bao gồm:

- **Linear kernel:** Không áp dụng biến đổi không gian, phân chia tuyến tính trong không gian gốc.
- **Polynomial kernel:** Chuyển dữ liệu về không gian đặc trưng bậc cao.
- **Radial Basis Function (RBF) kernel:** Là một trong các kernel phổ biến nhất, đặc biệt hiệu quả trong việc xử lý dữ liệu không tuyến tính.

Mô hình Cây quyết định

Cây quyết định (Decision Tree) là một cây phân cấp có cấu trúc được dùng để phân lớp các đối tượng dựa vào dãy các luật (series of rules). Các thuộc tính của đối tượng (ngoại trừ thuộc tính phân lớp - Category attribute) có thể thuộc các kiểu dữ liệu khác nhau (binary, nominal, ordinal, quantitative values) trong khi đó thuộc tính phân lớp phải có kiểu dữ liệu là Binary hoặc Ordinal. Tóm lại, cho dữ liệu về các đối tượng gồm các thuộc tính cùng với lớp (classes) của nó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các đối tượng chưa biết (unseen data).



Hình 8: Mô hình cây quyết định

Decision Tree là 1 sơ đồ khối đơn giản, chọn nhãn cho giá trị đầu vào. Nó gồm các nốt quyết định, kiểm tra giá trị đặc trưng. Decision Tree là một mô hình supervised learning, có thể được áp dụng vào cả hai bài toán classification và regression. Việc xây dựng một Decision Tree trên dữ liệu huấn luyện cho trước là việc đi xác định các câu hỏi và thứ tự của chúng. Một điểm đáng lưu ý của Decision Tree là nó có thể làm việc với các đặc trưng, thường là rời rạc và không có thứ tự. Decision Tree cũng làm việc với dữ liệu có vector đặc trưng bao gồm cả thuộc tính dạng phân loại và liên tục.

Cây quyết định là một cấu trúc được sử dụng để chia liên tiếp một tập các bản ghi lớn thành các tập con nhỏ hơn bằng cách áp dụng một chuỗi các luật đơn giản. Với mỗi phép chia liên tiếp, các tập con thu được trong tập kết quả sẽ ngày càng giống nhau. Nó có cấu trúc như sau:

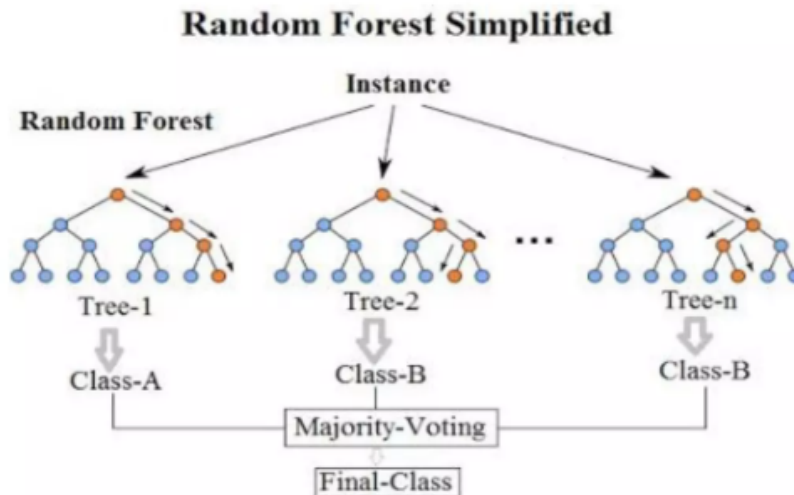
- Mỗi nút mang một thuộc tính (biến độc lập).
- Mỗi nhánh tương ứng với một giá trị của thuộc tính.
- Mỗi nút lá là một lớp (biến phụ thuộc).

Đối với cây quyết định, tại mỗi nút, một thuộc tính sẽ được chọn ra để phân tách tập mẫu thành những lớp khác nhau nhiều nhất có thể. Tiến hành lặp lại bước này đến khi kết thúc ta sẽ có được một tập các lớp đã được định nghĩa trước. Một trường hợp mới sẽ được phân loại dựa vào việc tìm một đường dẫn phù hợp tới nút lá.

Mô hình Random Forest

Random forest là thuật toán supervised learning, có thể giải quyết cả bài toán regression và classification.

Random là ngẫu nhiên, Forest là rừng, nên ở thuật toán Random Forest mình sẽ xây dựng nhiều cây quyết định bằng thuật toán Decision Tree, tuy nhiên mỗi cây quyết định sẽ khác nhau (có yếu tố random). Sau đó kết quả dự đoán được tổng hợp từ các cây quyết định.



Hình 9: Mô hình Random Forest

4.2 Huấn luyện mô hình

Các mô hình học máy được huấn luyện trên tập dữ liệu huấn luyện sau khi dữ liệu đã được xử lý và chuyển đổi thành dạng số.

Naive Bayes

Ta dùng `MultinomialNB()` để huấn luyện. Đây là một biến thể của mô hình Naive Bayes.

```
nb_model = MultinomialNB()
nb_model.fit(X_train_transformed, y_train)
y_pred = nb_model.predict(X_test_transformed)
```

Logistic Regression

Ta dùng `LogisticRegression` (Scikit-learn) để huấn luyện. Với tham số `max_iter=1000` (số lần lặp tối đa)


```
lg_model = LogisticRegression(max_iter=10000)
lg_model.fit(X_train_transformed, y_train)
y_pred = lg_model.predict(X_test_transformed)
```

KNN

Ta dùng KNeighborsClassifier (Scikit-learn) để huấn luyện. Với các tham số *neighbors=5*, *metric = cosine*

```
knn = KNeighborsClassifier(n_neighbors=5, metric='cosine')
knn.fit(X_train_transformed, y_train)
y_pred = knn.predict(X_test_transformed)
```

SVM

Ta dùng thư viện SVC để huấn luyện. Dự án có sử dụng 2 hàm kernel là linear kernel và Radial Basis Function (RBF) kernel

Linear Kernal

```
model = SVC(kernel='linear', random_state=42)
model.fit(X_train_transformed, y_train)
y_pred = model.predict(X_test_transformed)
```

RBF Kernal

```
model = SVC(kernel='rbf', random_state=42)
model.fit(X_train_transformed, y_train)
y_pred = model.predict(X_test_transformed)
```

Decision Tree

Ta dùng Decision Tree (Scikit-learn) để huấn luyện . Với các tham số *criterion = gini*, *max_depth=None* và *random_state=42*

```
clf = DecisionTreeClassifier(criterion='gini', max_depth=None, random_state=42)
clf.fit(X_train_transformed, y_train)
y_pred = clf.predict(X_test_transformed)
```

Random Forest

Ta dùng thư viện RandomForestClassifier, BaggingClassifier, ExtraTreesClassifier, AdaBoostClassifier để huấn luyện với tham số `n_estimators=100`

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_transformed, y_train)
y_pred = rf_model.predict(X_test_transformed)
```

5 Kết quả

5.1 Các thông số dùng để đánh giá mô hình

Classification Report là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán. Các chỉ số phổ biến trong một **Classification Report** bao gồm:

1. **Precision (Độ chính xác)**: Precision đo lường tỷ lệ giữa số lượng dự đoán đúng của một lớp so với tất cả các dự đoán mà mô hình đưa ra cho lớp đó. Nó cho ta biết mô hình có bao nhiêu dự đoán đúng trên tổng số dự đoán.

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

Trong đó:

- TP (True Positive): Số lượng dự đoán đúng là lớp đó.
 - FP (False Positive): Số lượng dự đoán sai là lớp đó.
2. **Recall (Độ bao phủ)**: Recall đo lường tỷ lệ giữa số lượng dự đoán đúng của một lớp so với tổng số thực tế của lớp đó. Nó cho ta biết mô hình có thể phát hiện bao nhiêu mẫu của lớp đó.

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

Trong đó:

- FN (False Negative): Số lượng mẫu của lớp đó bị dự đoán sai.
3. **F1-Score**: F1-Score là một chỉ số kết hợp giữa Precision và Recall, được tính bằng trung bình điều hòa. Nó là một chỉ số quan trọng khi bạn cần sự cân bằng giữa Precision và Recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

4. **Accuracy (Độ chính xác chung)**: Accuracy là tỷ lệ giữa số lượng dự đoán đúng trên tổng số mẫu.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

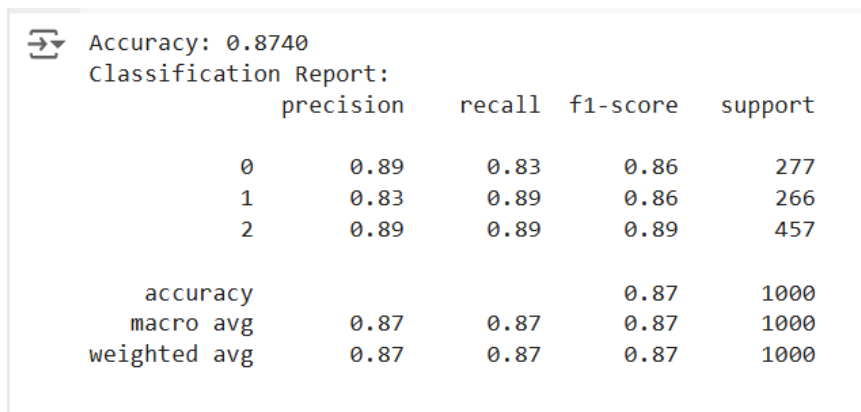
Trong đó:

- TN (True Negative): Số lượng dự đoán đúng không thuộc lớp.

5.2 Kết quả

Mô hình Naive Bayes

- Sau khi thực thi mô hình, ta có báo cáo phân tích của mô hình Naive Bayes như sau:



```
Accuracy: 0.8740
Classification Report:
      precision    recall  f1-score   support


     0       0.89      0.83      0.86       277
     1       0.83      0.89      0.86       266
     2       0.89      0.89      0.89       457

 accuracy          0.87       1000
 macro avg         0.87      0.87      0.87       1000
 weighted avg      0.87      0.87      0.87       1000
```

Hình 10: Mô hình Naive Bayes

Mô hình Logistic Regression

- Sau khi thực thi mô hình, ta có báo cáo phân tích của mô hình Logistic Regression như sau:



```

Accuracy: 0.9280
Classification Report:
              precision    recall  f1-score   support

     0           0.92       0.91       0.92         277
     1           0.91       0.95       0.93         266
     2           0.95       0.93       0.94         457


 accuracy          0.93         1000
 macro avg         0.92         1000
 weighted avg      0.93         1000

```

Hình 11: Mô hình hồi quy Logistic Regression

Mô hình KNN

- Sau khi thực thi mô hình, ta có báo cáo phân tích của mô hình KNN như sau:



```

Accuracy: 0.9770
Classification Report:
              precision    recall  f1-score   support

     0           0.97       0.97       0.97         277
     1           0.96       0.97       0.97         266
     2           0.99       0.98       0.98         457


 accuracy          0.98         1000
 macro avg         0.97         1000
 weighted avg      0.98         1000

```

Hình 12: Mô hình KNN

Mô hình Decision Tree

- Sau khi thực thi mô hình, ta có báo cáo phân tích của mô hình Decision Tree như sau:




| | | | | |
|------------------------|-----------|--------|----------|---------|
| Accuracy: 0.9230 | | | | |
| Classification Report: | | | | |
| | precision | recall | f1-score | support |
| 0 | 0.92 | 0.90 | 0.91 | 277 |
| 1 | 0.90 | 0.95 | 0.93 | 266 |
| 2 | 0.94 | 0.92 | 0.93 | 457 |
| accuracy | | | 0.92 | 1000 |
| macro avg | 0.92 | 0.92 | 0.92 | 1000 |
| weighted avg | 0.92 | 0.92 | 0.92 | 1000 |

Hình 13: Mô hình Decision Tree

Mô hình Random Forest

- Sau khi thực thi mô hình, ta có báo cáo phân tích của mô hình Random Forest như sau:



| | | | | |
|------------------------|-----------|--------|----------|---------|
| Accuracy: 0.9560 | | | | |
| Classification Report: | | | | |
| | precision | recall | f1-score | support |
| 0 | 0.97 | 0.94 | 0.95 | 277 |
| 1 | 0.97 | 0.94 | 0.96 | 266 |
| 2 | 0.94 | 0.98 | 0.96 | 457 |
| accuracy | | | 0.96 | 1000 |
| macro avg | 0.96 | 0.95 | 0.96 | 1000 |
| weighted avg | 0.96 | 0.96 | 0.96 | 1000 |

Hình 14: Mô hình Random Forest

Mô hình SVM linear

- Sau khi thực thi mô hình, ta có báo cáo phân tích của mô hình SVM linear như sau:

| | | | | | |
|------------------------|-----------|--------|----------|---------|--|
| Accuracy: 0.9390 | | | | | |
| Classification Report: | | | | | |
| | precision | recall | f1-score | support | |
| 0 | 0.92 | 0.93 | 0.93 | 277 | |
| 1 | 0.93 | 0.95 | 0.94 | 266 | |
| 2 | 0.95 | 0.94 | 0.95 | 457 | |
| accuracy | | | 0.94 | 1000 | |
| macro avg | 0.94 | 0.94 | 0.94 | 1000 | |
| weighted avg | 0.94 | 0.94 | 0.94 | 1000 | |

Hình 15: Mô hình SVM linear

Mô hình SVM nhân rbf

- Sau khi thực thi mô hình, ta có báo cáo phân tích của mô hình SVM rbf như sau:

| | | | | | |
|------------------------|-----------|--------|----------|---------|--|
| Accuracy: 0.9760 | | | | | |
| Classification Report: | | | | | |
| | precision | recall | f1-score | support | |
| 0 | 0.97 | 0.96 | 0.97 | 277 | |
| 1 | 0.98 | 0.97 | 0.98 | 266 | |
| 2 | 0.97 | 0.98 | 0.98 | 457 | |
| accuracy | | | 0.98 | 1000 | |
| macro avg | 0.98 | 0.97 | 0.98 | 1000 | |
| weighted avg | 0.98 | 0.98 | 0.98 | 1000 | |

Hình 16: Mô hình SVM rbf

5.3 Bàn luận

Điểm mạnh và điểm yếu của từng mô hình

- Mô hình Naive Bayes:

- **Điểm mạnh:** Mô hình đơn giản, dễ triển khai, và hiệu quả với các bài toán phân loại có phân phối xác suất rõ ràng.
- **Điểm yếu:** Độ chính xác chỉ đạt 87.4%, thấp hơn so với các mô hình khác. Recall của lớp 0 là 83%, có thể gặp khó khăn trong việc nhận diện các mẫu

của lớp này.

- **Mô hình Logistic Regression:**

- **Điểm mạnh:** Độ chính xác khá cao (92.8%), các chỉ số precision và recall đồng đều, hiệu quả trong các bài toán phân loại tuyến tính.
- **Điểm yếu:** Không xử lý tốt với dữ liệu phi tuyến tính hoặc dữ liệu có sự tương tác phức tạp.

- **Mô hình KNN:**

- **Điểm mạnh:** Độ chính xác rất cao (97.7%), tất cả các chỉ số precision và recall đều rất tốt cho các lớp. Thích hợp cho dữ liệu không có mối quan hệ tuyến tính rõ ràng.
- **Điểm yếu:** Mô hình yêu cầu nhiều tài nguyên tính toán và thời gian tính toán cao, đặc biệt khi bộ dữ liệu lớn.

- **Mô hình Decision Tree:**

- **Điểm mạnh:** Mô hình dễ hiểu và trực quan. Độ chính xác đạt 92.3%, với các chỉ số precision và recall đồng đều.
- **Điểm yếu:** Quá trình huấn luyện có thể dẫn đến overfitting nếu không được cắt tỉa cây quyết định (pruning).

- **Mô hình Random Forest:**

- **Điểm mạnh:** Độ chính xác cao (95.6%), ít bị overfitting nhờ vào việc sử dụng nhiều cây quyết định. Các chỉ số precision và recall đồng đều cho tất cả các lớp.
- **Điểm yếu:** Mô hình phức tạp, khó giải thích trực quan và thời gian huấn luyện dài hơn so với các mô hình đơn giản.

- **Mô hình SVM với Linear Kernel:**

- **Điểm mạnh:** Độ chính xác cao (93.9%), các chỉ số precision và recall tốt cho tất cả các lớp. Thích hợp cho các bài toán phân loại tuyến tính.
- **Điểm yếu:** Thời gian huấn luyện lâu nếu số lượng dữ liệu lớn.

- **Mô hình SVM với RBF Kernel:**

- **Điểm mạnh:** Độ chính xác rất cao (97.6%), precision và recall tốt cho tất cả các lớp. Mô hình có thể xử lý các mối quan hệ phi tuyến tính.
- **Điểm yếu:** Thời gian huấn luyện và dự đoán có thể kéo dài, yêu cầu tài nguyên tính toán lớn khi bộ dữ liệu lớn.

Tổng hợp kết quả huấn luyện mô hình

| Mô Hình | Accuracy | Precision (0) | Recall (0) | F1-Score (0) |
|---------------------|----------|---------------|------------|--------------|
| Naive Bayes | 0.8740 | 0.89 | 0.83 | 0.86 |
| Logistic Regression | 0.9280 | 0.92 | 0.91 | 0.92 |
| KNN | 0.9770 | 0.97 | 0.97 | 0.97 |
| Decision Tree | 0.9230 | 0.92 | 0.90 | 0.91 |
| Random Forest | 0.9560 | 0.97 | 0.94 | 0.95 |
| SVM (Linear) | 0.9390 | 0.92 | 0.93 | 0.93 |
| SVM (RBF) | 0.9760 | 0.97 | 0.96 | 0.97 |

Bảng 1: Bảng tổng hợp kết quả huấn luyện mô hình phân loại

Mô hình phù hợp nhất

Mô hình KNN là mô hình phù hợp nhất. KNN đạt được độ chính xác cao nhất (97.7%) và các chỉ số precision, recall đều rất tốt cho tất cả các lớp.

Lí giải tại sao chọn mô hình KNN mà không phải mô hình SVM:

KNN được chọn thay vì SVM vì có độ chính xác cao (97.7%) và các chỉ số precision, recall đều tốt cho tất cả các lớp. Tuy nhiên, SVM với RBF kernel (có độ chính xác 97.6%) mặc dù cho kết quả cũng tốt không kém, nhưng lại có nhược điểm là thời gian huấn luyện và dự đoán có thể kéo dài, đặc biệt khi bộ dữ liệu lớn.

Vì vậy, KNN được chọn vì nó có độ chính xác tương đương với SVM với RBF kernel nhưng dễ dàng triển khai, yêu cầu ít tài nguyên tính toán hơn và có thể tối ưu hóa nhanh chóng với ít tham số cần điều chỉnh. SVM mặc dù có độ chính xác cao hơn một chút, nhưng nó yêu cầu tài nguyên tính toán lớn và thời gian huấn luyện lâu, điều này khiến cho KNN trở thành lựa chọn phù hợp hơn khi cân nhắc giữa hiệu quả và tài nguyên tính toán trong các ứng dụng thực tế.

6 Kết luận và bài học kinh nghiệm

6.1 Tóm tắt kết quả

Tóm lại, những mô hình được đưa ra đã có thể giải quyết được yêu cầu bài toán đã nêu ra ban đầu với độ chính xác khá cao.

6.2 Hạn chế

Mặc dù các mô hình phân loại đã được huấn luyện với độ chính xác khá cao, mỗi mô hình đều có một số hạn chế cần được lưu ý:

- **Naive Bayes:** Mô hình này có giả định về tính độc lập giữa các đặc trưng, điều này có thể không đúng trong thực tế, dẫn đến việc mô hình bị giảm hiệu quả trong các bài toán mà các đặc trưng có mối quan hệ phức tạp với nhau.
- **Logistic Regression:** Mô hình này chỉ hoạt động hiệu quả với các bài toán phân loại tuyến tính. Nếu dữ liệu có sự tương tác phi tuyến tính giữa các đặc trưng, mô hình này sẽ không cho kết quả tốt.
- **KNN:** KNN cần tính toán khoảng cách giữa điểm cần phân loại và tất cả các điểm trong tập huấn luyện, điều này có thể rất tốn tài nguyên và thời gian tính toán khi tập dữ liệu lớn. Ngoài ra, KNN dễ bị ảnh hưởng bởi các đặc trưng không quan trọng.
- **Decision Tree:** Mô hình này có thể dẫn đến hiện tượng overfitting nếu không được cắt tỉa đúng cách. Thêm vào đó, cây quyết định có thể trở nên phức tạp và khó giải thích nếu không được đơn giản hóa.
- **Random Forest:** Mặc dù rất mạnh mẽ và ít bị overfitting, mô hình Random Forest vẫn có thể gặp khó khăn khi yêu cầu giải thích mô hình hoặc khi cần thực hiện các dự đoán nhanh trong thời gian thực.
- **SVM với Linear Kernel:** Mô hình này hoạt động tốt với các bài toán phân loại tuyến tính nhưng lại gặp khó khăn khi dữ liệu có sự tương tác phi tuyến tính.
- **SVM với RBF Kernel:** Mặc dù đạt được độ chính xác cao, mô hình SVM với RBF yêu cầu tài nguyên tính toán lớn và thời gian huấn luyện/dự đoán kéo dài, đặc biệt khi tập dữ liệu lớn.

Trong quá trình thực hiện, dự án vẫn còn tồn tại một số hạn chế cần khắc phục như sau

- Bộ dữ liệu được chọn chưa có kích thước đủ lớn về mặt thực tế. Trên thực tế, những bộ dữ liệu về đánh giá, nhận xét của người dùng có thể chứa đến hàng triệu bản ghi.
- Quá trình xử lý làm sạch dữ liệu còn chưa triệt để, vẫn còn nhiều sai sót, có thể nói đến như trong biểu đồ đám mây của nhãn Positive và Negative, ta có thể thấy có những từ cùng xuất hiện như các từ "game", "will", "twitter"...
- Cần có thêm những quá trình tối ưu hoá mô hình để có thể cải thiện độ chính xác và giảm thời gian thực thi trong tương lai.

6.3 Bài học

Qua việc huấn luyện các mô hình trên, có thể rút ra một số bài học sau:

- **Xử lý dữ liệu:** Trước khi huấn luyện mô hình, việc xử lý dữ liệu (bao gồm việc làm sạch, chuẩn hóa và chọn lựa các đặc trưng) là rất quan trọng để tối ưu hóa hiệu suất mô hình.
- **Chọn mô hình phù hợp:** Mỗi mô hình có ưu điểm và hạn chế riêng, vì vậy cần lựa chọn mô hình phù hợp với loại dữ liệu và bài toán cụ thể. Ví dụ, KNN có thể là lựa chọn tốt khi không có mối quan hệ tuyến tính rõ ràng giữa các đặc trưng, trong khi Logistic Regression lại hiệu quả với dữ liệu tuyến tính.
- **Tối ưu hóa mô hình:** Cần thực hiện các bước tối ưu hóa như chọn tham số (hyperparameter tuning) và tránh overfitting bằng cách sử dụng các kỹ thuật như cross-validation.
- **Overfitting và Underfitting:** Decision Tree là một mô hình dễ bị overfit nếu không được cắt tỉa đúng cách. Trong khi đó, các mô hình như **Naive Bayes** và **Logistic Regression** có xu hướng ít bị overfit hơn nhưng có thể gặp phải vấn đề underfitting khi dữ liệu không hoàn toàn đáp ứng các giả định của mô hình.

6.4 Tiềm năng trong tương lai

Trong tương lai, các mô hình học máy có thể được cải thiện bằng cách:

- **Mở rộng tập dữ liệu:** Việc thu thập và sử dụng các bộ dữ liệu lớn và đa dạng hơn có thể giúp cải thiện độ chính xác của mô hình.
- **Sử dụng các mô hình phức tạp hơn:** Các mô hình như deep learning, neural networks có thể được xem xét trong các bài toán phức tạp với dữ liệu lớn.

- **Tăng cường khả năng giải thích mô hình:** Các mô hình phức tạp như Random Forest và SVM có thể được cải thiện về khả năng giải thích để dễ dàng phân tích kết quả.
- **Áp dụng mô hình vào các bài toán lớn và phức tạp hơn:** Các mô hình phân loại hiện tại có thể được mở rộng và áp dụng vào các bài toán phức tạp hơn với tập dữ liệu lớn hơn, chẳng hạn như phân tích dữ liệu y tế, phân tích văn bản, hay các bài toán nhận diện hình ảnh và tiếng nói.

7 Phân công công việc

7.1 Đóng góp của từng thành viên

1. Nguyễn Quang Việt

- Tham gia viết báo cáo phần Tiền xử lý dữ liệu, Lựa chọn mô hình và huấn luyện, Kết quả
- Tham gia code và xây dựng mô hình.

2. Trương Đan Vi

- Tham gia viết báo cáo phần Giới thiệu và Thu thập dữ liệu, Lựa chọn mô hình và huấn luyện, Kết quả
- Tham gia code và xây dựng mô hình.

3. Nguyễn Văn Long

- Tham gia viết báo cáo phần Tiền xử lý dữ liệu. Lựa chọn mô hình và huấn luyện, Kết quả
- Tham gia code và xây dựng mô hình.

7.2 Sự hợp tác trong quá trình thực hiện dự án

Trong quá trình dự án, mỗi thành viên luôn hỗ trợ nhau trong việc viết báo cáo và lập trình, cụ thể hơn mỗi thành viên đều hỗ trợ lẫn nhau trong các giai đoạn quan trọng của dự án nhằm đảm bảo hoàn thành công việc đúng tiến độ và đạt chất lượng tốt nhất.

8 Tài liệu tham khảo

References

- [1] Go, Alec, Richa Bhayani, and Lei Huang. *Twitter Sentiment Classification Using Distant Supervision.* 2009. Web. <https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf>
- [2] D. O. Computer, C. wei Hsu, C. chung Chang, and C. jen Lin. A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin. Technical report, 2003.
- [3] N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, March 2000
- [4] Omuya EO, Okeyo G, Kimwele M. Sentiment analysis on social media tweets using dimensionality reduction and natural language processing. Engineering Reports. 2023; 5(3):e12579. doi:10.1002/eng2.12579

A Apendices

Link to the code: Sentiment Analysis

Danh sách hình vẽ

| | | |
|----|---|----|
| 1 | Biểu đồ phân bố của thuộc tính Sentiment trong tập huấn luyện . . . | 5 |
| 2 | Biểu đồ phân bố của thuộc tính Sentiment trong tập kiểm tra | 5 |
| 3 | Biểu đồ đám mây từ cho nhãn Positive | 7 |
| 4 | Biểu đồ đám mây từ cho nhãn Negative | 7 |
| 5 | Mô hình hồi quy Logistic | 10 |
| 6 | Mô hình K-Hàng xóm gần nhất KNN | 12 |
| 7 | Mô hình Support Vector Machine - Linear | 13 |
| 8 | Mô hình cây quyết định | 14 |
| 9 | Mô hình Random Forest | 15 |
| 10 | Mô hình Naive Bayes | 19 |
| 11 | Mô hình hồi quy Logistic Regression | 20 |
| 12 | Mô hình KNN | 20 |
| 13 | Mô hình Decision Tree | 21 |
| 14 | Mô hình Random Forest | 21 |
| 15 | Mô hình SVM linear | 22 |
| 16 | Mô hình SVM rbf | 22 |

Danh sách bảng

| | | |
|---|--|----|
| 1 | Bảng tổng hợp kết quả huấn luyện mô hình phân loại | 24 |
|---|--|----|