ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC & KỸ THUẬT THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

MÔN: HỌC MÁY THỐNG KÊ

LỚP: DS102.L21

BÀI TOÁN NHẬN DIỆN TIN GIẢ
FAKE NEWS DETECTION PROJECT

Sinh viên thực hiện

Thái Minh Triết - 19522397

Chu Hà Thảo Ngân - 19521882

Thành phố Hồ Chí Minh - 07/2020

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC & KỸ THUẬT THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

MÔN: HỌC MÁY THỐNG KÊ

LỚP: DS102.L21

BÀI TOÁN NHẬN DIỆN TIN GIẢ
FAKE NEWS DETECTION PROJECT

Giảng viên hướng dẫn

TS. Nguyễn Tấn Trần Minh Khang

ThS. Võ Duy Nguyên

Thành phố Hồ Chí Minh - 07/2020

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

LỜI CẢM ƠN

Lời đầu tiên, chúng tôi xin trân trọng cảm ơn hai Thầy giảng viên hướng dẫn là TS. Nguyễn Tấn Trần Minh Khang và ThS. Võ Duy Nguyên, các thầy đã tận tình giảng dạy chúng tôi trong suốt quá trình học tập cũng như đã hướng dẫn chúng tôi hoàn thành đồ án của môn học này.

Xin cảm ơn anh Hồ Thái Ngọc (PMCL2016) đã có sự giúp đỡ tận tình trong các tiết học lý thuyết và thực nghiệm tại lớp.

Do giới hạn về kiến thức và khả năng lý luận còn nhiều hạn chế, kính mong sự chỉ dẫn và đóng góp của các quý Thầy để bài báo cáo của chúng tôi được hoàn thiện hơn. Xin chân thành cảm ơn!

Nhóm sinh viên

Thái Minh Triết

Chu Hà Thảo Ngân

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

TÓM TẮT

Trong kỷ nguyên của Internet và mạng xã hội, cứ mỗi giây trôi qua, thế giới có gần 55.000 bài đăng trên Facebook, 456.000 tweet được gửi đi trên Twitter và có 86 bài blog xuất hiện trên Internet. Có thể thấy lượng thông tin được sản sinh ra và lan truyền nhanh chóng như thế nào trên không gian mạng. Tuy nhiên, những nguồn tin này luôn tiềm ẩn những thông tin giả mạo mà nếu không nhận diện và ngăn chặn kịp thời sẽ gây tổn hại đến đến các cá nhân, tổ chức kinh tế, chính trị và xã hội.

Các phương pháp nhận diện tin giả trước đây thường mang tính thủ công khi phải cần đến các chuyên gia để có thể thẩm định nguồn tin là đúng hay sai sự thật. Bên cạnh đó, các phương pháp nhận diện tự động ứng dụng lý thuyết Học máy và Xử lý ngôn ngữ tự nhiên là những hướng tiếp cận mới, trở thành xu hướng nghiên cứu nóng hổi vì mang lại hiệu quả cao, tiết kiệm thời gian và chi phí, vốn là những mặt còn nhiều hạn chế của các phương pháp cũ.

Trong đồ án này, chúng tôi tiến hành thực nghiệm những phương pháp Học máy phổ biến cùng các mô hình trích xuất đặc trưng từ dữ liệu văn bản, bao gồm 2 mô hình Bag-of-Words là TFIDFVectorizer và CountVectorizer cùng mô hình Word2Vec, với mục tiêu tìm hiểu khả năng trích xuất thông tin từ bộ dữ liệu cũng như độ hiệu quả của từng phương pháp trong việc phân loại một mẩu tin là đáng tin cậy hay là sai sự thật.

Từ khóa: Machine Learning, Fake News Detection, Natural Language Processing, Classification, Word Embedding

CHƯƠNG 1
GIỚI THIỆU CHUNG

1.1 Mục tiêu đề tài

Mục tiêu đặt ra của bài toán nhằm tìm hiểu và xây dựng các mô hình máy học để nhận diện tin tức giả bằng cách phân loại các tin tức từ bộ dữ liệu theo 2 nhãn: REAL (tin thật) và FAKE (tin giả).

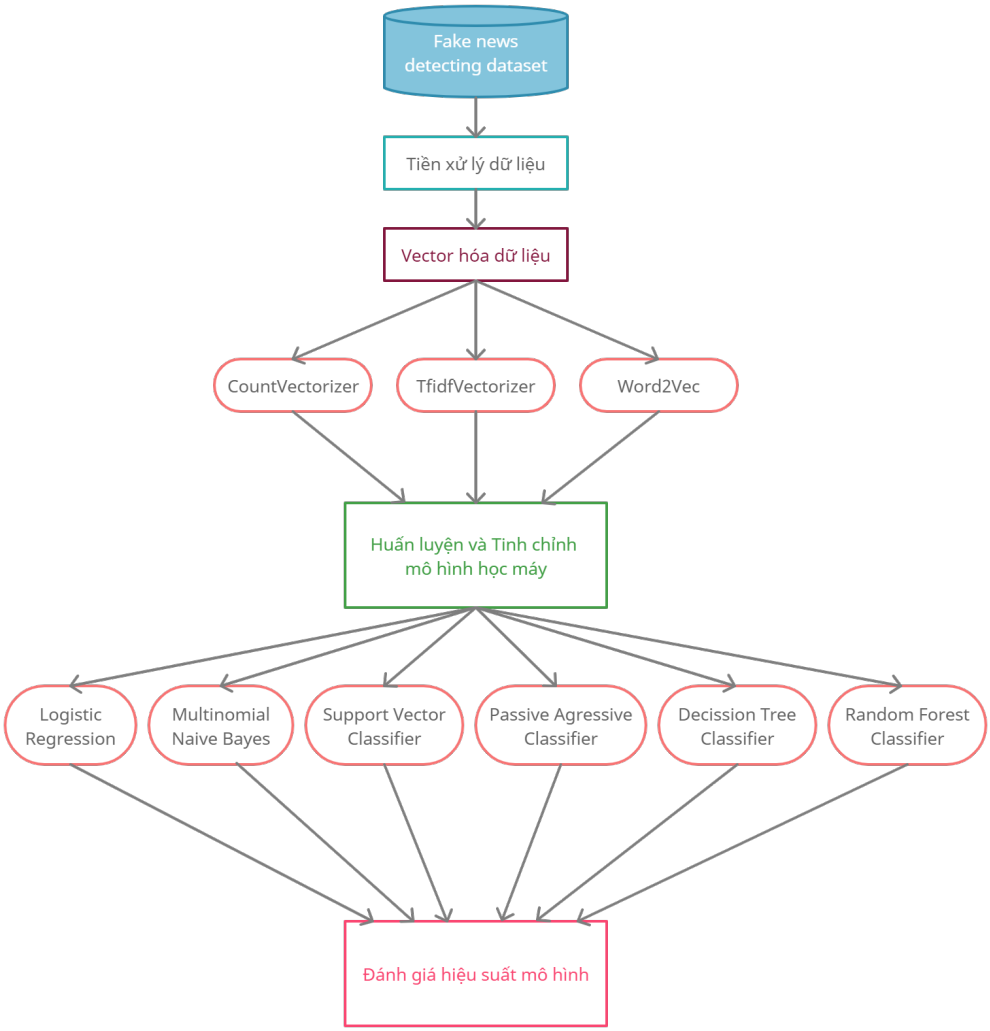
1.2 Bài toán học máy

Bài toán này thuộc lớp bài toán phân loại, có tổng 2 lớp đại diện: tin giả và tin thật.

Đầu vào của bài toán là tiêu đề tin tức hoặc nội dung đoạn tin tức.

Đầu ra là kết quả dự đoán: bằng 0 cho tin thật, bằng 1 cho tin giả.

1.3 Quy trình giải quyết bài toán



Hình 1.1: Quy trình thực hiện bài toán

1.4 Ứng dụng

CHƯƠNG 2
TỔNG QUAN VỀ BỘ DỮ LIỆU

2.1 Giới thiệu chung về bộ dữ liệu

Bộ dữ liệu Detecting Fake News Dataset của tác giả Hassan Amin được chia sẻ trên nền tảng Kaggle (?)

Rahul Patodi (link drive thầy gửi) <— ông này chỉ viết blog tổng hợp các dataset lên drive ổng (?)

Bảng 2.1: Thông tin bộ dữ liệu.

Thông tin	**Nội dung**
Tên bộ dữ liệu	Detecting Fake News Dataset
Nguồn dữ liệu	`https://www.kaggle.com/hassanamin/textdb3`
Kích thước bộ dữ liệu	6335 điểm dữ liệu
Thông tin thuộc tính	4 thuộc tính. Trong đó: • Unnamed: ID dạng số ngẫu nhiên • title: Tiêu đề tin tức • text: Nội dung tin tức • label: Nhãn phân loại tin tức
Ý nghĩa các nhãn	Có 2 nhãn "REAL" và "FAKE": • REAL: Đây là tin tức thật • REAL: Đây là tin tức giả
Thông tin Tác giả	Hassan Amin

2.2 Phân tích và trực quan bộ dữ liệu

CHƯƠNG 3
TIỀN XỬ LÝ VÀ VECTƠ HÓA DỮ LIỆU

3.1 Tiền xử lý dữ liệu

3.2 Trích xuất đặc trưng

3.2.1 CountVectorizer

3.2.2 TFIDF

3.2.3 Word2Vec

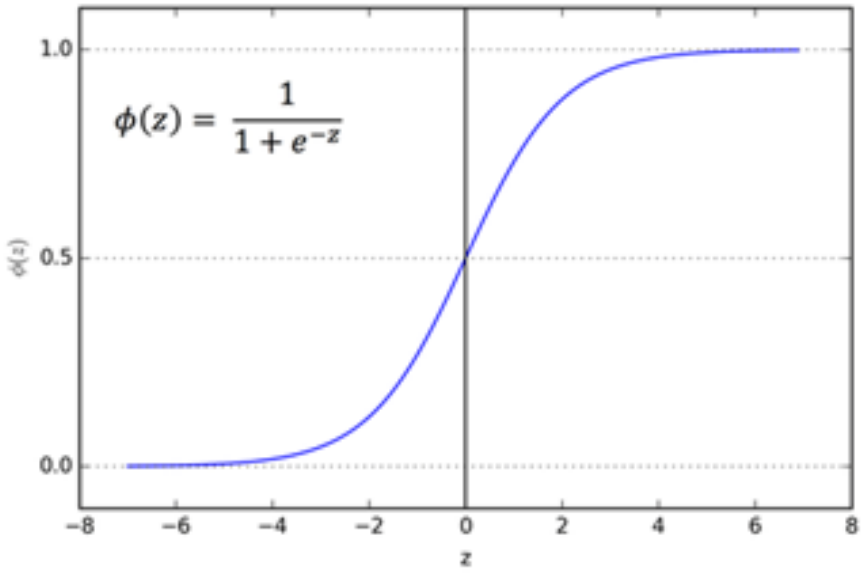
CHƯƠNG 4
HUẤN LUYỆN VÀ TINH CHỈNH CÁC MÔ HÌNH HỌC MÁY

4.1 Cơ sở lý thuyết

4.1.1 Logistic Regression

Phương pháp Logistic Regression là một mô hình hồi quy được sử dụng phổ biến cho bài toán phân lớp nhị phân.

Mô hình hoạt động dựa trên hàm sigmoid với đường cong chữ S đặc trưng, dùng để dự đoán xác suất xảy ra hay không xảy ra của một nhãn nào đó.



Hình 4.1: Đồ thị hàm sigmoid

Đầu ra của bài toán logistic regression với dữ liệu đầu vào **x** và vector

hệ số $\mathbf{w}$ thường viết chung dưới dạng:

$$f(x) = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

Sau khi tìm được mô hình, việc xác định class $\mathbf{y}$ cho một điểm dữ liệu $\mathbf{x}$ được xác định bằng việc so sánh hai biểu thức xác suất:

$$P(y = 1|\mathbf{x}; \mathbf{w}); P(y = 0|\mathbf{x}; \mathbf{w})$$

Vì tổng hai biểu thức này luôn bằng 1 nên một cách gọn hơn, ta cần xác định xem $P(y = 1|\mathbf{x}; \mathbf{w})$ có lớn hơn 0.5 hay không. Nếu lớn hơn 0.5, ta kết luận điểm dữ liệu thuộc class 1, ngược lại thì điểm dữ liệu thuộc class 0.

4.1.2 Multinomial Naive Bayes

Bộ phân lớp Naive Bayes (NBC) thường được sử dụng trong các bài toán phân loại văn bản, với thời gian train và test rất nhanh. Điều này có được do giả sử về tính độc lập giữa các thành phần, nếu biết class. Nếu giả sử về tính độc lập được thoả mãn (dựa vào bản chất của dữ liệu), NBC được cho là cho kết quả tốt hơn so với SVM và Logistic Regression khi có ít dữ liệu training.

NBC có thể hoạt động với các feature vector mà một phần là liên tục (sử dụng Gaussian Naive Bayes), phần còn lại ở dạng rời rạc (sử dụng Multinomial hoặc Bernoulli).

Mô hình Multinomial Naive Bayes chủ yếu được sử dụng trong bài toán phân loại văn bản mà vector đặc trưng được xây dựng dựa trên ý tưởng Bag of words (BoW). Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d (là số từ trong văn bản). Giá trị của thành phần thứ i trong mỗi vector là số lần từ thứ i xuất hiện trong văn bản đó.

Khi đó $p(\mathbf{x}_i|c)$ tỉ lệ với tần suất từ thứ i (hay đặc trưng thứ i trong trường hợp tổng quát) xuất hiện trong các văn bản có class c. Giá trị này được tính bởi công thức:

$$\lambda_{ci} = p(\mathbf{x}_i|c) = \frac{N_{ci}}{N_c} \ (*)$$

Trong đó:

- N_{ci} là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c.
- N_c là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c.

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong nhãn c thì biểu thức $(*)$ sẽ bằng 0. Để giải quyết việc này, một kỹ thuật được gọi là làm mềm Laplace (Laplace Smoothing) được áp dụng:

$$\hat{\lambda} = \frac{N_{ci} + \alpha}{N_c + d\alpha}$$

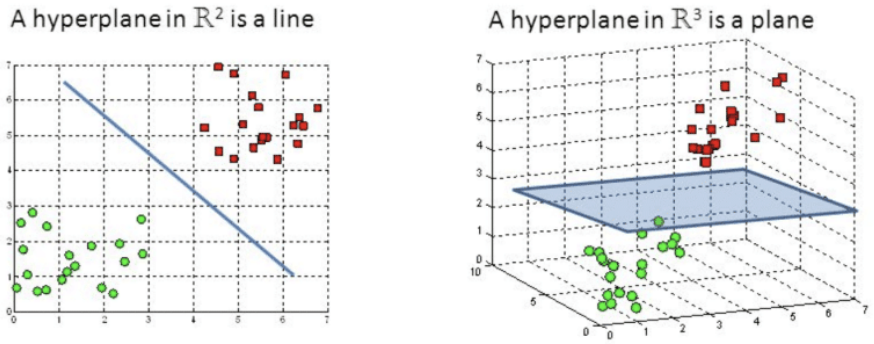
Với α là một số dương (thường bằng 1) để tránh trường hợp tử số bằng 0. Mẫu số được cộng với $d\alpha$ để đảm bảo tổng xác suất bằng 1.

Như vậy, mỗi nhãn c được mô tả bằng một bộ các số dương có tổng bằng 1:

$$\hat{\lambda_c} = \hat{\lambda}c1, ..., \hat{\lambda}cd$$

4.1.3 Support Vector Machine

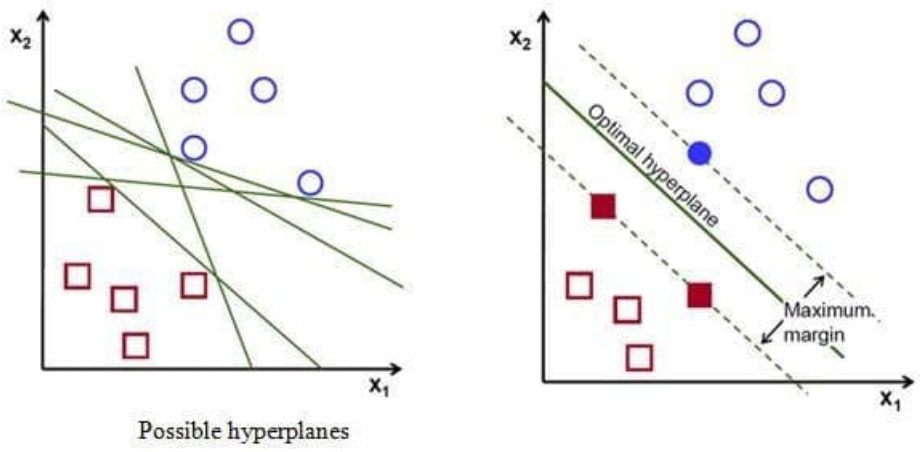
Mục tiêu của SVM là tìm ra một siêu phẳng trong không gian N chiều (ứng với N đặc trưng) chia dữ liệu thành hai phần tương ứng với lớp của chúng. Nói theo ngôn ngữ của đại số tuyển tính, siêu phẳng này phải có lề cực đại và phân chia hai bao lồi và cách đều chúng.



Hình 4.2: Siêu phẳng tạo ra một biên giới phân chia 2 lớp của dữ liệu

Biên (hay lề - Margin)

Là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với 2 phân lớp. Một cách trực quan, đường thẳng phân tách 2 tập nhãn là tốt nhất nếu như nó không quá gần một tập nào đó, nghĩa là độ rộng biên M là cực đại.



Hình 4.3: Siêu phẳng tối ưu có lề cực đại

Vector hỗ trợ

Phía trên chính là ý tưởng của phương pháp SVM. Bên cạnh đó, các vector hỗ trợ - là các điểm dữ liệu nằm trên hoặc gần nhất với siêu phẳng chúng ảnh hưởng đến vị trí và hướng của siêu phẳng. Các vector này được sử dụng để tối ưu hóa lề và nếu xóa các điểm này, vị trí của siêu phẳng sẽ thay đổi. Một điểm lưu ý nữa đó là các vector hỗ trợ phải cách đều siêu phẳng.

SVM giải quyết vấn đề overfitting rất tốt, là phương pháp phân lớp nhanh, có hiệu suất tổng hợp tốt và có hiệu suất tính toán cao.

Tham số C

Tham số C được gọi là "siêu tham số" (hyperparameter). Tham số C cho biết mức độ tối ưu hoá SVM để giảm thiểu phân lớp sai (misclassification) đối với điểm dữ liệu mới đi vào. Nếu mô hình SVM quá khớp (overfitting), ta có thể điều chỉnh giảm tham số C (khi đó độ rộng biên sẽ tăng lên) và ngược lại, tham số C càng lớn thì độ rộng biên càng hẹp.

Tham số gamma

Gamma là một siêu tham số được sử dụng với SVM phi tuyến tính. Một trong những nhân phi tuyến tính được sử dụng phổ biến nhất là hàm cơ sở xuyên tâm (RBF). Tham số gamma của RBF kiểm soát khoảng cách ảnh hưởng của một điểm đào tạo duy nhất.

Giá trị thấp của gamma cho thấy bán kính tương tự lớn dẫn đến nhiều điểm được nhóm lại với nhau. Đối với các giá trị cao của gamma, các điểm cần phải rất gần nhau để được xem xét trong cùng một nhóm (hoặc lớp). Do đó, các mô hình có giá trị gamma rất lớn có xu hướng quá mức.

Thủ thuật Kernel Một kernel là một hàm ánh xạ dữ liệu từ không gian ít nhiều hơn sang không gian nhiều chiều hơn, từ đó ta tìm được siêu phẳng phân tách dữ liệu. Các kiểu kernel:

- Tuyến tính
- Đa thức
- RBF
- sigmoid

4.1.4 Thuật toán Passive Agressive

4.1.5 Decission Tree

Decision Tree (cây quyết định) là mô hình thuộc nhóm thuật toán học máy học có giám sát (Supervised Learning). Mô hình cây quyết định cho biết giá trị của một biến mục tiêu thuộc lớp nào bằng cách dựa vào các giá trị của một tập các biến dự đoán, hay tập các câu hỏi.

Cây quyết định có dạng cấu trúc cây, bao gồm: nút gốc (root node), nút quyết định (decision node) và nút lá (leaf node).

- Nút gốc: Là nút đầu tiên trong cấu trúc cây, thường nằm trên cùng, các nút khác trong cấu trúc cây bắt nguồn từ nút gốc.
- Nút quyết định: Là nút có một hoặc nhiều phân nhánh tới nút con, nút này tương ứng với biến dự đoán.
- Nút lá: Là giá trị dự đoán của biến mục tiêu.
- Nhánh nối giữa một nút và nút con thể hiện giá trị cụ thể cho biến đó.

Quy tắc để có được giá trị dự đoán cụ thể là biểu diễn đường đi từ nút gốc đi xuyên qua cây theo các nhánh đến nút lá. Từ đó, cây quyết định sẽ sinh ra các luật để dự đoán lớp của các dữ liệu chưa biết. Giải bài toán phân lớp dựa trên mô hình cây quyết định chính là xây dựng một cây quyết định.

Hai thuật toán xây dựng cây quyết định phổ biến là CART và ID3.

Thuật toán CART sử dụng độ đo Gini

Gini index là chỉ số mức độ nhiễu loạn thông tin hay sự khác biệt về giá trị lớp của mỗi điểm dữ liệu trong một tập hợp con. Gini index cho phép đánh giá sự tối ưu của từng cách phân nhánh thông qua xác định mức độ purity của từng nút trong mô hình cây quyết định.

Công thức tổng quát hệ số Gini:

$$GINI(t) = 1 - \sum_{j}[p(j|t)]^2 \tag{4.1}$$

Trong đó, t là một nút bất kỳ có chứa các điểm dữ liệu mang lớp j của biến mục tiêu và p là xác suất để một điểm dữ liệu trong t có lớp j, nếu

tất cả các điểm dữ liệu đều mang lớp j, lúc này p = 1 và hệ số Gini sẽ bằng 0, khi đó nút được công nhận là pure node

Thuật toán ID3 sử dụng độ đo Entropy

Công thức tổng quát Entropy:

$$Entropy(t) = -\sum_{j}[p(j|t)]^2 * log_2[p(j|t)] \tag{4.2}$$

Khi xác suất p càng lớn thì $log_2[p(j|t)]$ sẽ mang giá trị âm tiến tới 0, nhân với giá trị âm xác suất thì công thức Entropy luôn bé hơn 1. Giống như Gini index, giá trị Entropy càng nhỏ thì nút càng chứa nhiều điểm dữ liệu có cùng lớp j bất kỳ.

Hai thuật toán như nhau trong đa số trường hợp. Gini thường chạy nhanh hơn nên được mặc định trong Scikit-Learn. Entropy thường cho cây cân bằng hơn.

4.1.6 Random Forest

Random Forest là thuật toán học có giám sát nên có thể giải quyết bài toán classification.

Random Forest (hay còn gọi là Rừng Ngẫu Nhiên) được phát triển bởi Leo Breiman tại đại học California, Berkeley; là một trong những thuật toán thuộc kỹ thuật Bagging của phương pháp Ensemble Learning với mục đích dùng để phân loại bằng cách xây dựng vô số các cây quyết định (decision tree) trong thời gian huấn luyện. Đầu ra là tập hợp mô hình phân lớp của những cây riêng biệt: $h(X, \Theta n)$ trong đó Θn là những vector ngẫu nhiên phân phối độc lập như nhau và mỗi cây sẽ chọn ra được một đơn vị chọn lớp phổ biến nhất thuộc dữ liệu đầu vào X.

Mỗi cây quyết định đều có những yếu tố ngẫu nhiên: Lấy ngẫu nhiên

dữ liệu để xây dựng cây quyết định hoặc lấy ngẫu nhiên các thuộc tính để xây dựng cây quyết định. Ở bước huấn luyện Random Forest sẽ xây dựng nhiều cây quyết định, các cây quyết định có thể khác nhau. Sau đó ở bước dự đoán, với một dữ liệu mới, thì ở mỗi cây quyết định, đi từ trên xuống theo các nút điều kiện để được các dự đoán, sau đó kết quả cuối cùng được tổng hợp từ kết quả của các cây quyết định.

Trong những năm gần đây, Random Forest được sử dụng khá phổ biến bởi những điểm vượt trội của nó so với các thuật toán khác: xử lý được với dữ liệu có số lượng các thuộc tính lớn, làm việc với tập dữ liệu thiếu giá trị, thường có độ chính xác cao trong phân loại, tránh được overfitting với tập dữ liệu và quá trình học nhanh.

4.2 Huấn luyện và tinh chỉnh mô hình

Sau khi đã nắm vững cơ sở lý thuyết của các phương pháp Học máy, chúng tôi tiến hành huấn luyện và thẩm định các bộ giá trị siêu tham số ứng với từng mô hình nhằm tìm ra bộ tham số tối ưu.

Do hạn chế về mặt tài nguyên và thời gian, chúng tôi chỉ thử nghiệm một số lượng nhất định giá trị siêu tham số của các mô hình, do đó các mô hình sau khi tinh chỉnh có thể chưa là tốt nhất. Sau cùng, chúng tôi mong muốn có thể đưa ra bộ tham số tốt nhất của từng mô hình trên từng bộ vector hóa cụ thể, nhằm thực hiện việc so sánh và đánh giá một cách khách quan nhất về hiệu suất giữa các mô hình trên bộ dữ liệu.

Việc tinh chỉnh mô hình Học máy được thực hiện bằng cách sử dụng công cụ GridsearchCV hỗ trợ bởi thư viện sklearn, đánh giá qua phương pháp k-Fold Cross Validation với k=3 trên tập huấn luyện đã được tiền

xử lý và vector hóa, cùng độ đo F1_macro.

Sau đây là bảng kết quả tinh chỉnh mô hình với GridSearchCV.

Kết quả tinh chỉnh mô hình với GridSearchCV			
	CountVectorizer	**TfidfVectorizer**	**Word2Vec**
LogisticRegression	solver = sag C = 1	solver = saga C = 1000	solver = newton-cg C = 10
MultinomialNB	alpha = 0.1 fit_prior = True	alpha = 0.1 fit_prior = False	alpha = 1.1 fit_prior = True
SVC	kernel = linear C = 0.1 gama = 1	kernel = rbf C = 100 gamma = 0.1	kernel = rbf C = 10 gamma = 0.1
PassiveAgressive Classifier	C = 0.001 early_stopping = True	C = 0.1 early_stopping = False	C = 0.001 early_stopping = False
DecissionTree Classifier	criterion = entropy max_depth = None min_sample_leaf = 2 min_sample_split = 5	criterion = gini max_depth = 16 min_sample_leaf = 1 min_sample_split = 2	criterion = entropy max_depth = 6 min_sample_leaf = 3 min_sample_split = 2
RadomForest Classifier	criterion = entropy max_depth = None min_sample_split = 5 min_sample_leaf = 1 max_features = auto n_estimators = 400	criterion = gini max_depth = None min_sample_split = 10 min_sample_leaf = 1 max_features = auto n_estimators = 400	criterion = entropy max_depth = None min_sample_split = 5 min_sample_leaf = 1 max_features = auto n_estimators = 300

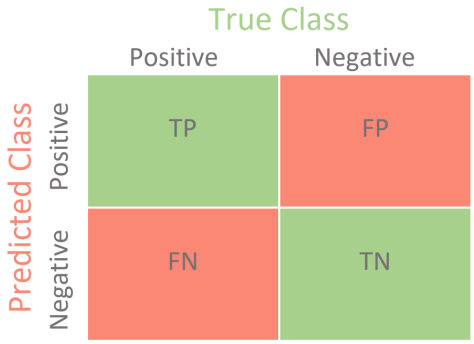
Bảng 4.1: Bảng kết quả GridSearchCV

CHƯƠNG 5
ĐÁNH GIÁ HIỆU SUẤT MÔ HÌNH

5.1 Các độ đo đánh giá mô hình

5.1.1 Confusion Matrix

Khi đánh giá một mô hình phân lớp, để biết được cụ thể các lớp được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất, dữ liệu lớp nào thường bị phân nhầm vào lớp khác,... chúng ta thường sử dụng một ma trận được gọi là Confusion Matrix (hay còn gọi là Ma trận nhầm lẫn).



Hình 5.1: Cấu trúc cơ bản của một Confusion Matrix

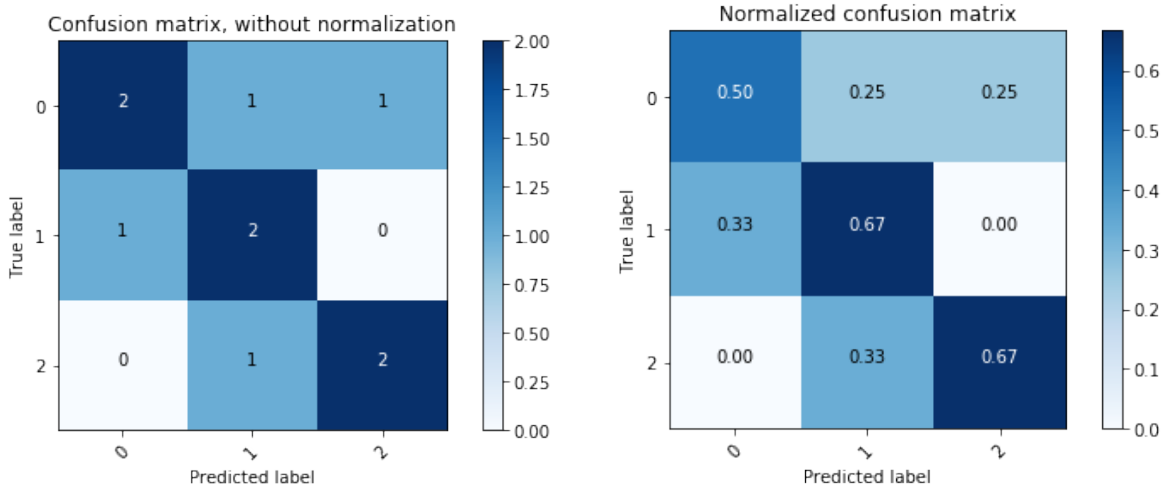
Trong bài toán phân lớp nhị phân, Confusion Matrix là một ma trận có kích thước là 2x2 (như hình 5.1), trong đó:

- **True Positive (TP):** số điểm dữ liệu có nhãn là Positive được dự đoán là Positive.
- **False Positive (FP):** số điểm dữ liệu có nhãn là Positive được dự đoán là Negative.

- **False Negative (TN):** số điểm dữ liệu có nhãn là Negative được dự đoán là Positive.
- **True Negative (FN):** số điểm dữ liệu có nhãn là Negative được dự đoán là Negative.

Một cách tổng quát, Confusion Matrix là một ma trận vuông có kích thước mỗi chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ i, cột thứ j là số lượng điểm lẽ ra thuộc vào class i nhưng lại được dự đoán là thuộc vào class j. Bên cạnh cách đinh nghĩa trên thì vẫn có một số tài liệu định nghĩa ngược lại.

Để có cái nhìn rõ hơn thì Confusion Matrix còn được biểu diễn bằng cách chuẩn hóa (normalize). Việc chuẩn hóa Confusion Matrix được thực hiện bằng cách lấy từng phần tử trong mỗi hàng chia cho tổng phần tử trên hàng đó. Như vậy tổng của các phần tử trên một hàng luôn bằng 1.



(a) Trước khi chuẩn hóa **(b)** Sau khi chuẩn hóa

Hình 5.2: Ví dụ về Confusion Matrix trước và sau khi được chuẩn hóa

5.1.2 Accuracy

Độ chính xác Accuracy là độ đo đơn giản dùng để đánh giá hiệu suất phân loại của một mô hình Học máy. Cách đánh giá này đơn giản tính tỉ lệ số lượng điểm dữ liệu được dự đoán đúng trên tổng số điểm dữ liệu trong tập kiểm thử:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Độ đo Accuracry thường không phù hợp đối với bài toán phân loại có số lượng điểm dữ liệu giữa các lớp quá mất cân bằng, vì mô hình có xu hướng đánh giá dựa trên lớp có nhiều điểm dữ liệu hơn. Tuy nhiên đối với bộ dữ liệu sử dụng trong bài toán này thì độ đo Accuracy vẫn mang lại hiệu quả do phân bố điểm dữ liệu trên hai lớp là cân bằng.

5.1.3 Precision - Recall - F1

Độ đo Precison được tính bằng tỉ lệ giữa số điểm dữ liệu có nhãn là Positive được dự đoán đúng trên tổng số dự đoán cho nhãn Positive đưa ra bởi mô hình:

$$precision = \frac{TP}{TP + FP}$$

Độ đo Recall (còn gọi là Sensitivity hay độ phủ) được tính bằng tỉ lệ giữa số điểm dữ liệu có nhãn là Positive được dự đoán đúng trên tổng số điểm dữ liệu có nhãn là Positive:

$$recall = \frac{TP}{TP + FN}$$

Nếu **Precision cao** trong khi **Recall thấp**: Mô hình có độ chính xác cao trên tổng số dự đoán Positive được mô hình đưa ra, tuy nhiên nó lại

có xu hướng bỏ sót nhiều điểm dữ liệu có nhãn là Positive trong thực tế.

Nếu **Recall cao** trong khi **Precision thấp**: Mô hình có độ phủ tốt trên các nhãn Positive, tuy nhiên độ chính xác của kết quả dự đoán các điểm dữ liệu này lại không cao.

Để thể hiện sự tổng hòa giữa Precision và Recall, một độ đo khác được sử dụng để đánh giá mô hình phân lớp, đó là độ đo F1 và được tính bằng trung bình điều hòa của Precision và Recall:

$$F_1 = 2 \times \frac{precision \cdot recall}{precision + recall}$$

Độ đo F_1 thường được dùng khi chúng ta cần sự đồng đều giữa precision và recall hoặc khi bộ dữ liệu quá mất cân bằng. Vì vậy nó được sử dụng rất phổ biến trong các bài toán thực tế và trong nghiên cứu khoa học.

5.1.4 Đường cong ROC và chỉ số AUC

Đường cong ROC (Receiver Operating Characteristic Curve) là một đồ thị đánh giá hiệu suất thường được sử dụng trong bài toán phân lớp nhị phân. Ứng dụng đầu tiên của nó là cho việc nghiên cứu các hệ thống nhận diện trong việc phát hiện các tín hiệu radio khi có sự hiện diện của nhiễu vào thập niên 1940, sau sự kiện cuộc tấn công Trân Châu Cảng.

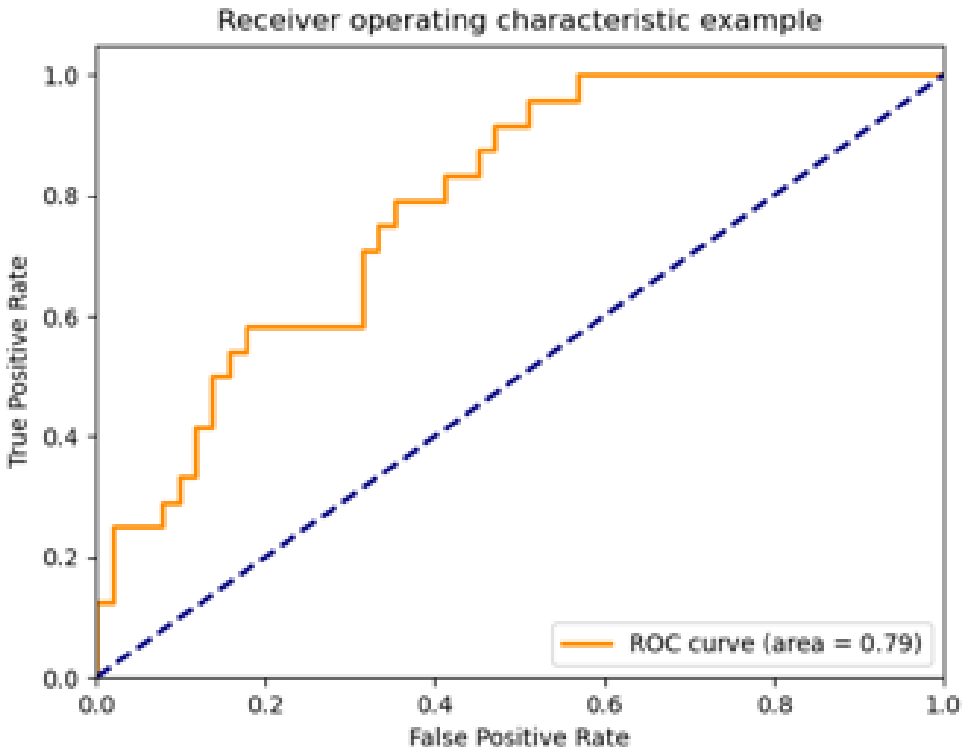
Đường cong ROC thường được biểu diễn bằng cặp chỉ số (TPR, FPR) tại mỗi ngưỡng với TPR là trục tung và FPR là trục hoành. Trong đó:

- $TPR = Recall = \dfrac{TP}{TP + FN}$
- $FPR = \dfrac{FP}{TN + FP}$: tỉ lệ giữa số điểm dữ liệu có nhãn Negative bị dự đoán sai trên tổng số điểm dữ liệu có nhãn là Negative.

Đây chính là các chỉ số dùng để tính toán hiệu suất phân loại của mô hình. Để hợp chúng lại thành 1 chỉ số duy nhất, ta sử dụng đường cong ROC để hiển thị từng cặp (TPR, FPR) cho các ngưỡng khác nhau với mỗi điểm trên đường cong biểu diễn 1 cặp (TPR, FPR) cho 1 ngưỡng, sau đó tính chỉ số AUC cho đường cong này.

Chỉ số AUC (Area under the curve) chính là diện tích không gian bên dưới đường cong ROC, là con số thể hiện hiệu suất phân loại của mô hình.

AUC càng gần về 1 (ROC hội tụ về góc trên bên trái) thì mô hình càng phân loại chính xác, AUC càng gần về 0.5 (ROC hội tụ về đường chéo) thì hiệu suất phân loại càng tệ. Nếu AUC càng gần về 0 thì mô hình sẽ phân loại ngược kết quả (phân loại Positive thành Negative và ngược lại)



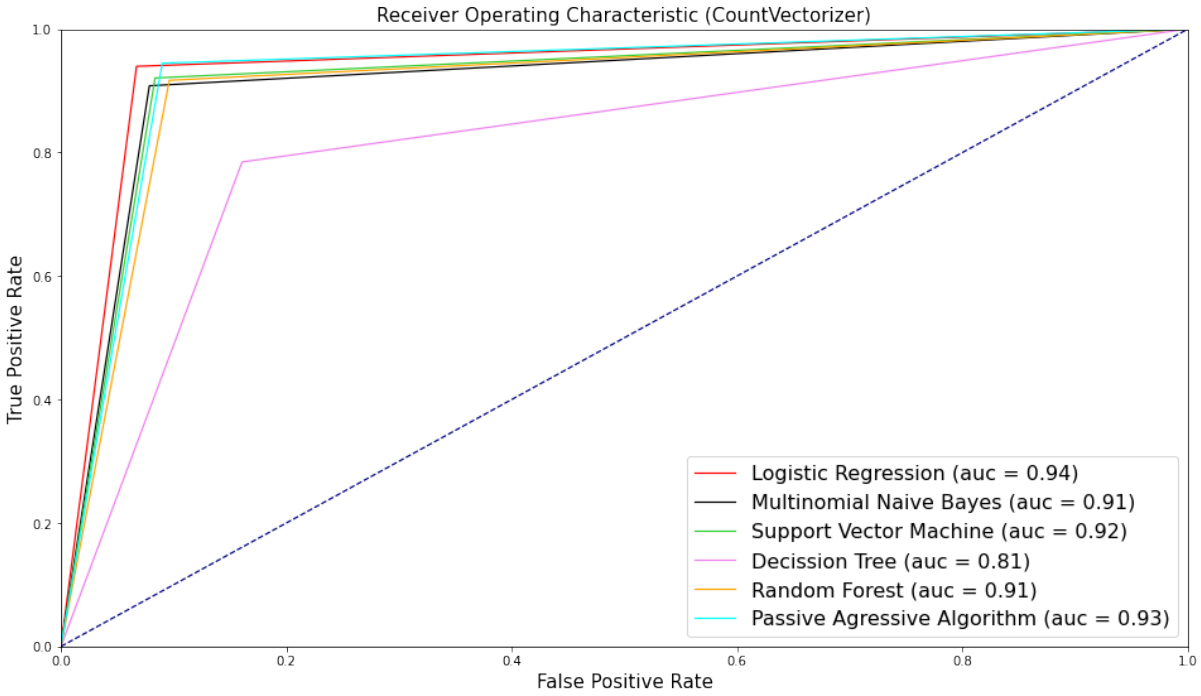
Hình 5.3: Ví dụ về đường cong ROC và AUC

5.2 Kết quả đánh giá hiệu suất mô hình

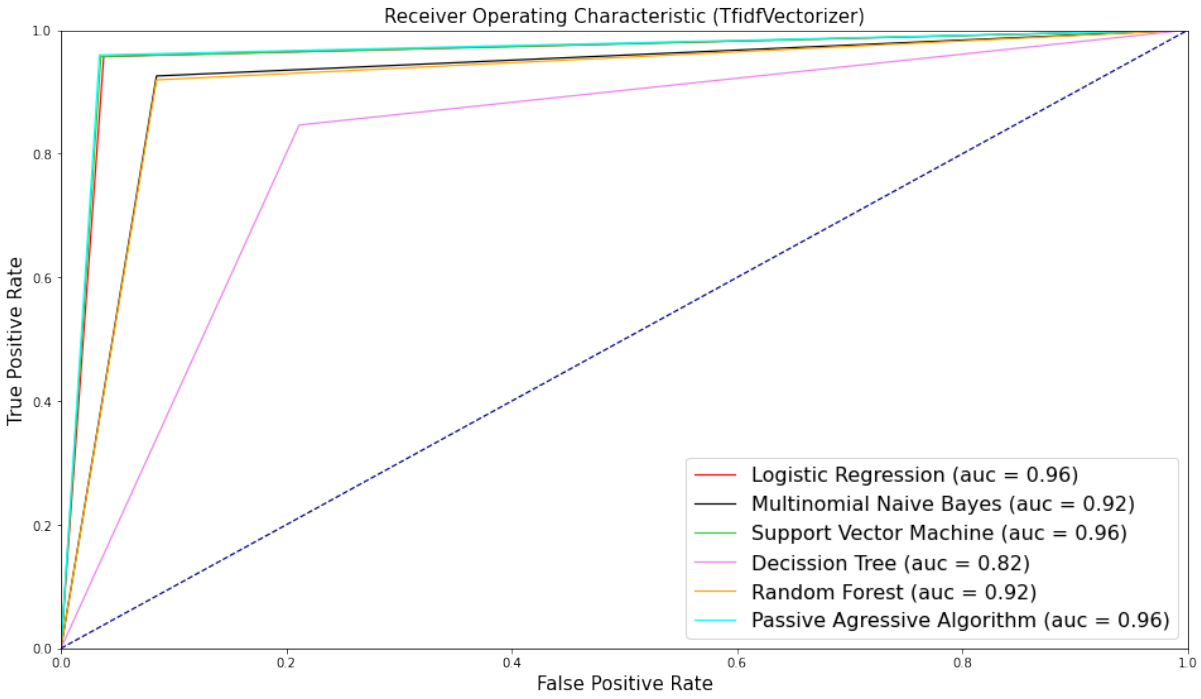
Khi đã tìm được những giá trị siêu tham số tốt nhất của từng mô hình trên từng bộ vector hóa, chúng tôi tiến hành dự đoán các điểm dữ liệu trong tập kiểm thử và thu được kết quả đánh giá dưới đây.

	CountVectorizer	**TfidfVectorizer**	**Word2Vec**
LogisticRegression	**0.9362**	0.9596	0.9135
MultinomialNB	0.9148	0.9205	0.8062
SVC	0.9186	0.9609	**0.9293**
PassiveAgressiveClassifier	0.9274	**0.9628**	0.9085
DecissionTreeClassifier	0.8119	0.8176	0.7898
RandomForestClassifer	0.9104	0.9173	0.9034

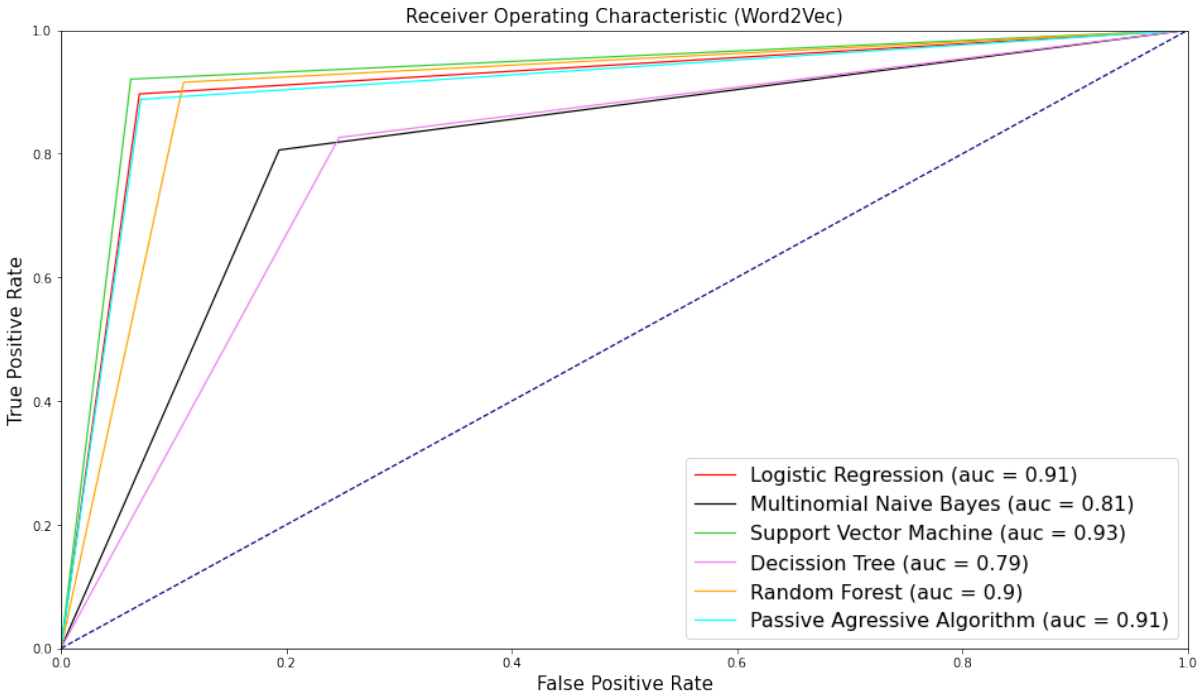
Bảng 5.1: Bảng tổng hợp kết quả đánh giá độ chính xác (Accuracy) của các mô hình



Hình 5.4: Đường cong ROC cho bộ CountVectorizer

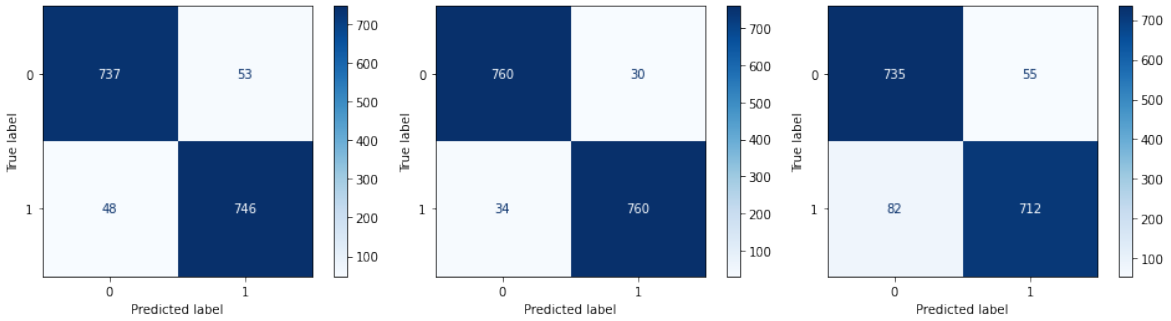


Hình 5.5: Đường cong ROC cho bộ TfidfVectorizer



Hình 5.6: Đường cong ROC cho bộ Word2Vec

5.2.1 Kết quả đánh giá LogisicRegression



(a) CountVectorizer **(b) TfidfVectorizer** **(c) Word2Vec**

Hình 5.7: Confusion Matrix LogisticRegression

	precision	recall	f1-score	support
0	0.94	0.93	0.94	790
1	0.93	0.94	0.94	794
macro avg	0.94	0.94	0.94	1584

(a) CountVectorizer

	precision	recall	f1-score	support
0	0.96	0.96	0.96	790
1	0.96	0.96	0.96	794
macro avg	0.96	0.96	0.96	1584

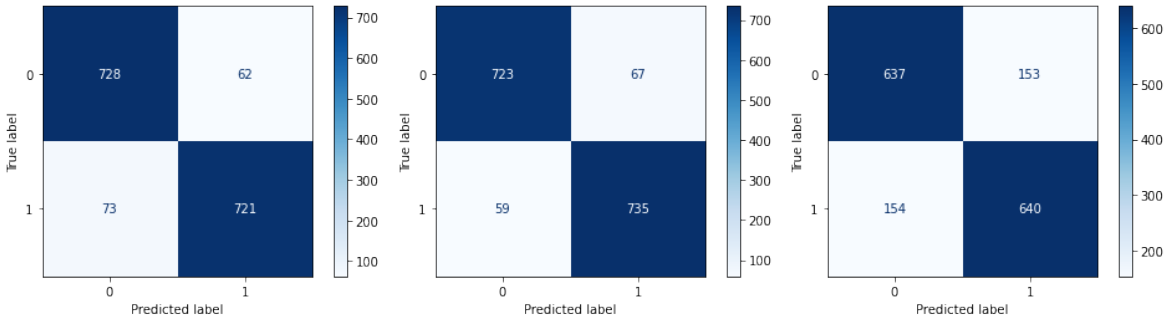
(b) TfidfVectorizer

	precision	recall	f1-score	support
0	0.90	0.93	0.91	790
1	0.93	0.90	0.91	794
macro avg	0.91	0.91	0.91	1584

(c) Word2Vec

Bảng 5.2: Classification report LogisticRegression

5.2.2 Kết quả đánh giá MultinomialNB



(a) CountVectorizer **(b) TfidfVectorizer** **(c) Word2Vec**

Hình 5.8: Confusion Matrix MultinomialNB

	precision	recall	f1-score	support
0	0.91	0.92	0.92	790
1	0.92	0.91	0.91	794
macro avg	0.91	0.91	0.91	1584

(a) CountVectorizer

	precision	recall	f1-score	support
0	0.92	0.92	0.92	790
1	0.92	0.93	0.92	794
macro avg	0.92	0.92	0.92	1584

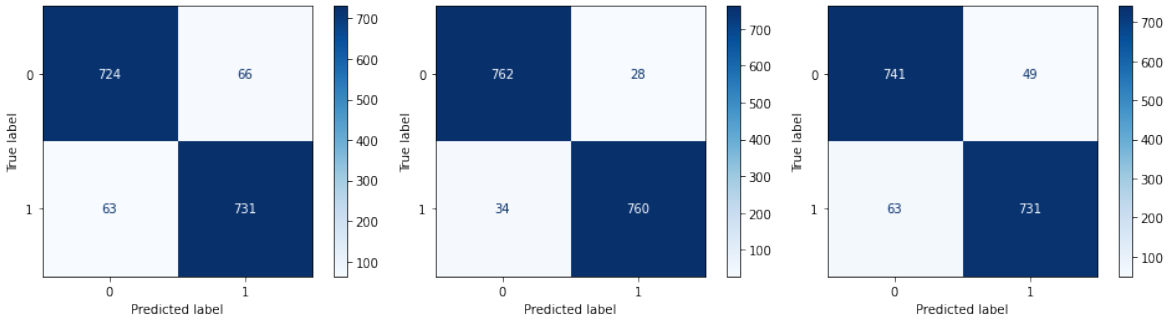
(b) TfidfVectorizer

	precision	recall	f1-score	support
0	0.81	0.81	0.81	790
1	0.81	0.81	0.81	794
macro avg	0.81	0.81	0.81	1584

(c) Word2Vec

Bảng 5.3: Classification report MultinomialNB

5.2.3 Kết quả dánh giá SVC



(a) CountVectorizer **(b) TfidfVectorizer** **(c) Word2Vec**

Hình 5.9: Confusion Matrix SVC

	precision	recall	f1-score	support
0	0.92	0.92	0.92	790
1	0.92	0.92	0.92	794
macro avg	0.92	0.92	0.92	1584

(a) CountVectorizer

	precision	recall	f1-score	support
0	0.96	0.96	0.96	790
1	0.96	0.96	0.96	794
macro avg	0.96	0.96	0.96	1584

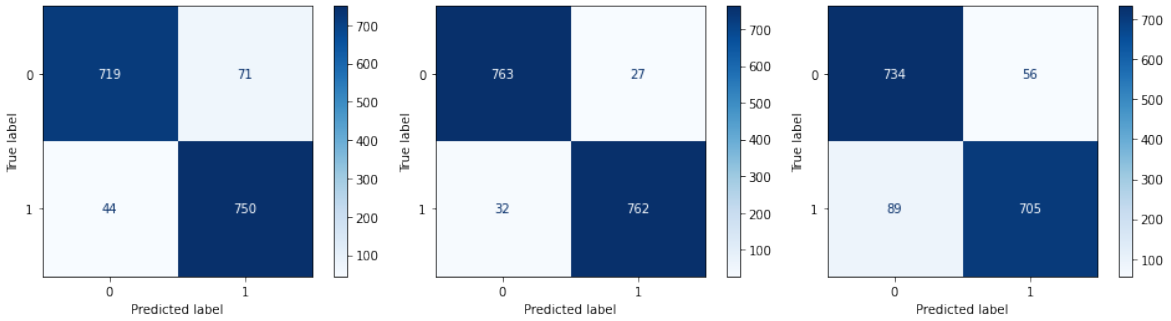
(b) TfidfVectorizer

	precision	recall	f1-score	support
0	0.92	0.94	0.93	790
1	0.94	0.92	0.93	794
macro avg	0.93	0.93	0.93	1584

(c) Word2Vec

Bảng 5.4: Classification report SVC

5.2.4 Kết quả đánh giá PassiveAgressiveClassifier



(a) CountVectorizer **(b) TfidfVectorizer** **(c) Word2Vec**

Hình 5.10: Confusion Matrix PassiveAgressiveClassifier

	precision	recall	f1-score	support
0	0.94	0.91	0.93	790
1	0.91	0.94	0.93	794
macro avg	0.93	0.93	0.93	1584

(a) CountVectorizer

	precision	recall	f1-score	support
0	0.96	0.97	0.96	790
1	0.97	0.96	0.96	794
macro avg	0.96	0.96	0.96	1584

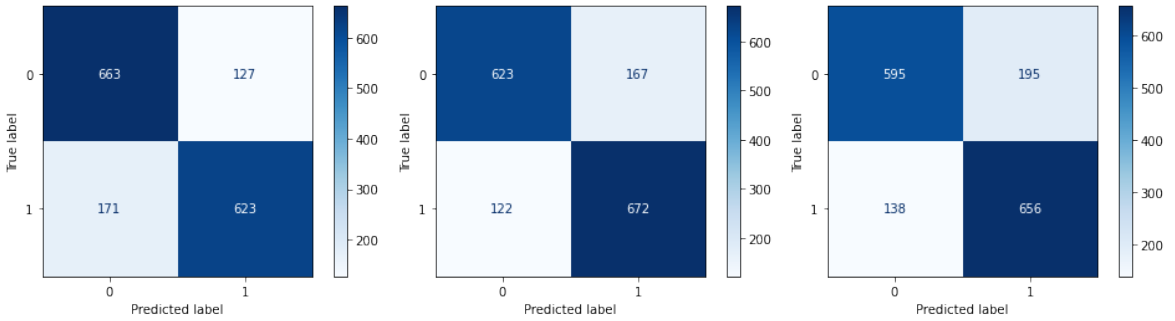
(b) TfidfVectorizer

	precision	recall	f1-score	support
0	0.89	0.93	0.91	790
1	0.93	0.89	0.91	794
macro avg	0.91	0.91	0.91	1584

(c) Word2Vec

Bảng 5.5: Classification report PassiveAgressiveClassifier

5.2.5 Kết quả đánh giá DecisionTreeClassifier



(a) CountVectorizer **(b) TfidfVectorizer** **(c) Word2Vec**

Hình 5.11: Confusion Matrix DecisionTreeClassifier

	precision	recall	f1-score	support
0	0.79	0.84	0.82	790
1	0.83	0.78	0.81	794
macro avg	0.81	0.81	0.81	1584

(a) CountVectorizer

	precision	recall	f1-score	support
0	0.84	0.79	0.81	790
1	0.80	0.85	0.82	794
macro avg	0.82	0.82	0.82	1584

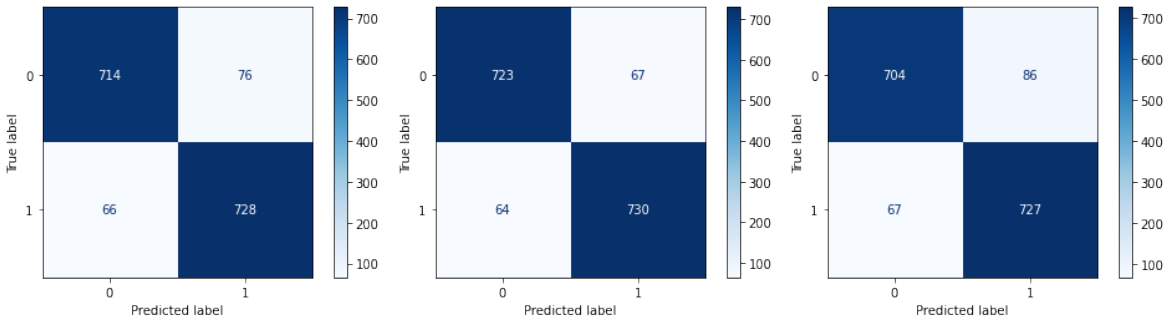
(b) TfidfVectorizer

	precision	recall	f1-score	support
0	0.81	0.75	0.78	790
1	0.77	0.83	0.80	794
macro avg	0.79	0.79	0.79	1584

(c) Word2Vec

Bảng 5.6: Classification report DecisionTreeClassifier

5.2.6 Kết quả đánh giá RandomForestClassifier



(a) CountVectorizer **(b) TfidfVectorizer** **(c) Word2Vec**

Hình 5.12: Confusion Matrix RandomForestClassifier

	precision	recall	f1-score	support
0	0.92	0.90	0.91	790
1	0.91	0.92	0.91	794
macro avg	0.91	0.91	0.91	1584

(a) CountVectorizer

	precision	recall	f1-score	support
0	0.92	0.92	0.92	790
1	0.92	0.92	0.92	794
macro avg	0.92	0.92	0.92	1584

(b) TfidfVectorizer

	precision	recall	f1-score	support
0	0.91	0.89	0.90	790
1	0.89	0.92	0.90	794
macro avg	0.90	0.90	0.90	1584

(c) Word2Vec

Bảng 5.7: Classification report RandomForestClassifier

CHƯƠNG 6
PHÂN TÍCH LỖI VÀ HƯỚNG PHÁT TRIỂN

6.1 Phân tích lỗi

6.2 Hướng phát triển

KẾT LUẬN

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Trần Thị Tuấn Anh , *Bài toán ứng dụng cực trị trong kinh tế*, Tài liệu giảng dạy, Trường Đại học Kinh tế Thành phố Hồ Chí Minh.

[2] Bùi Quang Hân, Trần Văn Bồi, Phạm Ngọc Tiến, Nguyễn Thành Tương (1998), *Giải toán Vật lí 10, Tập 1* , NXB Giáo dục.

[3] Bùi Quang Hân, Đào Văn Cư, Phạm Ngọc Tiến, Nguyễn Thành Tương (2000), *Giải toán Vật lí 11 Tập 1* , NXB Giáo dục.

[4] Trần Nam Dũng (2018), *Giải tích và các bài toán cực trị*, Tạp chí Thông tin Toán học, Tập 22 Số 4.

[5] Nguyễn Huy Đoan (Chủ biên) (2008), *Bài tập Giải tích 12 nâng cao*, NXB Giáo dục.

[6] Trần Văn Hạo (Tổng Chủ biên) (2008), *Giải tích 12* , NXB Giáo dục.

[7] Nguyễn Xuân Liêm, Nguyễn Mạnh Quý (2009), *Giáo trình phép tính vi phân và tích phân của hàm nhiều biến số*, NXB Đại học Sư phạm.

[8] Đoàn Quỳnh (Tổng Chủ biên) (2008), *Giải tích 12 nâng cao*, NXB Giáo dục.

[9] Vũ Tuấn (Chủ biên) (2008), *Bài tập Giải tích 12*, NXB Giáo dục.

[10] Lê Đình Thúy, Nguyễn Quỳnh Lan (2012), *Toán cao cấp cho các nhà kinh tế*, NXB Đại học Kinh tế Quốc dân.

[11] Lê Anh Vũ , *Bài giảng Toán cao cấp* , Đại học Kinh tế- Luật, Đại học Quốc gia Thành phố Hồ Chí Minh.