Tìm hiểu các thuật toán xác định giao dịch bất thường

Đại học Sư Phạm Kỹ Thuật hành phố Hồ Chí Minh

1. Đặt vấn đề

Ngày nay, thực hiện các giao dịch online thông qua ngân hàng điện tử ngày càng được hướng đến nhiều hơn. Kèm với sự tiện lợi của việc này thì vẫn tồn tại những rủi ro gây ra thiệt hại lớn cho người dùng. Gần đây hàng loạt bài báo về vấn đề này xuất hiện ngày càng nhiều như: "Tài khoản bỗng chốc bốc hơi 400 triệu đồng sau một đêm", "Tài khoản bốc hơi trong vài phút", "Điện thoại liên tục báo trừ tiền tài khoản vào nửa đêm",… cùng hàng loạt bài báo với tiêu đề tương tự. Vậy biện pháp để hạn chế vấn đề này là gì? Đây chính là câu hỏi đặt ra cho các hệ thống ngân hàng điện tử. Những năm gần đây, học máy (một nhành của trí tuệ nhân tạo) ngày càng phổ biến và phát triển nhanh chóng do có tính ứng dụng thực tiễn cao. Vì vậy, việc ứng dụng học máy và áp dụng các thuật toán để xác định các giao dịch bất thường là vô cùng cần thiết và cấp bách. Nhằm giúp các hệ thống ngân hàng điện tử có thể tự phân tích các giao dịch từ đó xác định ra các giao dịch bất thường. Từ đó góp phần ngăn chặn và hạn chế tổn hại đến mức tối thiểu. Nghiên cứu này ứng dụng một số thuật toán phổ biến để xác định giao dịch bất thường. Nhóm tác giả sẽ tiến hành tìm hiểu, tổng hợp thông tin và đồng thời thực hiện demo đơn giản cho các thuật toán này.

2. Giới thiệu

2.1. Machine learning là gì?

Machine learning là khoa học (và nghệ thuật) lập trình máy tính để chúng có thể học từ dữ liệu.

2.2. Các loại hệ thống Machine learning

Có rất nhiều loại hệ thống Machine learning khác nhau có thể được phân loại dựa trên:

- Có được đào tạo với sự giám sát của con người hay không:
  + Supervised learning (Học có giám sát)
  + Unsupervised learning (Học không giám sát)
  + Semi-supervised learning (Học bán giám sát)
  + Reinforcement learning (Học tăng cường)

- Có thể học tăng dần một cách nhanh chóng hay không (online versus batch learning).
- Có hoạt động đơn giản bằng cách so sánh các điểm dữ liệu mới với các điểm dữ liệu đã biết hay thay vào đó là phát hiện các mẫu trong dữ liệu đào tạo và xây dựng mô hình dự đoán, giống như các nhà khoa học làm (instance-based versus model-based learning).

3. Giải pháp

- Sau khi tìm hiểu các dạng Machine learning khác nhau, nhóm nghiên cứu quyết định tìm hiểu sâu về hai dạng của Machine learning là Supervised learning và Unsupervised learning để tìm ra giải pháp cho bài toán. Từ đó sẽ tìm hiểu một số thuật toán phù hợp để giải quyết vấn đề đặt ra ứng với mỗi dạng.
- Cụ thể với Supervised learning nhóm sẽ tìm hiểu về hai thuật toán là Random Forests và k-Nearest Neighbors. Với Unsupervised learning nhóm sẽ tìm hiểu về thuật toán k-Means.

3.1 Supervised learning

- Trong học tập có giám sát, dữ liệu training bạn cung cấp cho thuật toán bao gồm các giải pháp mong muốn, được gọi là nhãn.
- Một ví dụ về học tập có giám sát điển hình là phân loại. Bộ lọc thư rác là một ví dụ điển hình về điều này: nó được đào tạo với nhiều email mẫu cùng với lớp của chúng (spam hoặc ham) và nó phải học cách để phân loại các email mới.
- Dưới đây là một số thuật toán học có giám sát quan trọng:
  + k-Nearest Neighbors
  + Linear Regression
  + Logistic Regression
  + Support Vector Machines (SVMs)
  + Decision Trees and Random Forests
  + Neural networks

3.2 Unsupervised learning

- Theo như tên gọi của thuật toán, việc học máy máy sẽ không có sự hướng dẫn rõ ràng và dữ liệu được đưa vào sẽ không được tiền xử lý bởi con người.
- Một số thuật toán không giám sát quan trọng:

+ Clustering
    - k-Means
    - Hierarchical Cluster Analysis (HCA)
    - Expectation Maximization
+ Visualization and dimensionality reduction
    - Principal Component Analysis (PCA)
    - Kernel PCA
    - Locally-Linear Embedding (LLE)
    - t-distributed Stochastic Neighbor Embedding (t-SNE)
+ Association rule learning
    - Apriori
    - Eclat

- Một ví dụ điển hình học tập không giám sát là giả sử bạn có nhiều dữ liệu về khách truy cập trang web đọc sách của mình. Bạn có thể muốn chạy một thuật toán phân cụm để cố gắng phát hiện các nhóm đọc giả truy cập tương tự (như nhóm yêu thích thể loại khoa học viễn tưởng, hay nhóm thích sách về kinh tế,…). Bạn không cho thuật toán biết trước là khách truy cập thuộc nhóm nào: nó sẽ phải tìm thấy các kết nối đó mà không cần sự trợ giúp của bạn.

4. Các thuật toán

4.1. Thuật toán Random Forest

4.1.1. Khái quát về Random Forest

Định nghĩa:

- Đây là một phương pháp xây dựng dựa trên một ý tưởng đơn giản: "trí tuệ đám đông" để tổng hợp một tập hợp rất nhiều decision trees và sử dụng phương pháp “voting” để đưa quyết định về biến “target” cần được dự báo. Vì vậy kĩ thuật này được gọi là "Ensemble Learning" - học tập theo cụm.

Cấu tạo:

- Là một tập các decision trees.

Ưu điểm:

- Hoạt động tốt hơn decision trees.
- Kế thừa ưu điểm decision trees.

Nhược điểm:

- Độ chính xác không bằng decision trees được tăng cường độ dốc.
- Đặc điểm dữ liệu ảnh hưởng đến hiệu suất.

Nguyên lý hoạt động:

- Sử dụng kỹ thuật tổng hợp bootsting hoặc đóng gói chung. Cho một Tập huấn luyện X và các phản hồi Y đóng gói lặp lại (B lần).
- Sau huấn luyện các dự đoán cho các mẫu chưa nhìn thấy x' có thể thực hiện bằng cách lấy trung bình các dự đoán từ tất cả các cây hồi quy riêng lẻ trên x'.

Các xây dựng Random Forest:

- B1: Import the data
- B2: Train the model
- B3: Search the best maxnodes
- B4: Search the best ntrees
- B5: Evaluate the model
- B6: Visualize Result

4.1.2. Khái quát về Decision Tree

Định nghĩa:

- Là thuật toán học máy đa năng có thể thực hiện cả nhiệm vụ phân loại và hồi quy.
- Là một thuật toán rất mạnh và được tích hợp để giải quyết nhiều vấn đề phức tạp.
- Ngoài ra còn là thành phần cơ bản của Random Forest - một trong những thuật toán Machine Learning mạnh nhất hiện nay.

Cấu tạo decision tree gồm 3 loại nút:

- Các nút quyết định - Hình vuông.
- Các nút cơ hội - Vòng tròn.
- Các nút kết thúc - Tam giác.

Ưu điểm:

- Rất đơn giản để hiểu và giải thích thích.
- Giúp xác định giá trị xấu nhất, tốt nhất và giá trị mong đợi cho các tình huống khác nhau.
- Là thành phần cơ bản trong Machine Learning, có thể dễ dàng kết hợp với các thuật toán khác nhau.

Nhược điểm:

- Không ổn định, một sự thay đổi nhỏ có thể dẫn đến sự thay đổi lớn trong cấu trúc quyết định tối ưu của cây.
- Chỉ mang tính tương đối và phụ thuộc rất nhiều vào thông tin thu được.
- Các phép tính có thể trở nên rất phức tạp, đặc biệt là khi có rất nhiều giá trị không chắc chắn và nhiều kết quả được liên kết lại với nhau.

Nguyên lý hoạt động:

- Cây quyết định nếu thỏa điều kiện 1 và điều kiện 2 và điều kiện 3 thì cho ra kết quả.

Cách cài đặt 1 cây quyết định:

- B1: Import the data.
- B2: Clean the dataset.
- B3: Create train/test set.
- B4: Build the model.
- B5: Make a prediction.
- B6: Measure performance.
- B7: Tune the hyper-parameters.

4.2 Thuật toán K-Nearest Neighbors

Định nghĩa:

- K-Nearest Neighbors là thuật toán kết hợp giữa học giám sát và không giám sát.
- Ở phương pháp học không giám sát thuật toán là nền tảng của nhiều phương pháp học tập khác nhau, nổi bật nhất vẫn là phương pháp Notably Manifold (Đa tạp) và Spectral Clustering (Phân cụm quang phổ).
- Ở phương pháp học giám sát, k-Nearest Neighbors có thể áp dụng được vào cả hai loại của bài toán là Classification (Phân loại) và Regression (Hồi quy).
- K-Nearest Neighbors là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu.

Nguyên tắc hoạt động:

- Là thuật toán tìm một số lượng mẫu huấn luyện được xác định trước trong khoảng cách gần nhất với điểm mới và dự đoán nhãn từ chúng.
- Số lượng mẫu có thể là một hằng số do người dùng xác định hoặc sẽ thay đổi dựa theo mật độ cục bộ của điểm.

- Khoảng cách có thể là bất kỳ thước đo nào. Nhưng khoảng cách Euclide là tiêu chuẩn lựa chọn phổ biến nhất.
- Đây là phương pháp không tổng quát hóa, vì chỉ hoạt động dựa trên hoạt động "ghi nhớ" tất cả dữ liệu huấn luyện.
- Mặc dù chỉ "ghi nhớ" nhưng thuật toán đã thành công trong giải bài toán có số lượng lớn phân loại và hồi quy. Và là công cụ để giải quyết phân loại ranh giới xác định điểm bất thường.

Ưu điểm:

- Thuật toán đơn giản, dễ dàng triển khai.
- Độ phức tạp tính toán nhỏ.
- Xử lý tốt với tập dữ liệu nhiễu.

Nhược điểm:

- Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác.
- Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
- Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.

4.3 Thuật toán K-Means

- K-mean là một thuật toán thuộc nhóm Clustering.
- Clustering là cách nhóm 1 tập hợp các Objects dựa trên các đặc điểm, thuộc tính, tổng hợp chúng lại theo từng nhóm dựa trên độ giống nhau (similarities)
- Thuật toán liên quan đến data mining - phân chia data bằng thuật toán join được chỉnh định, thích hợp nhất cho các phân tích thông tin mong muốn.
- Yếu tố ảnh hưởng quan trọng đến thuật toán nhóm clustering là việc định nghĩa thế nào là giống nhau, không giống nhau, bởi vì nó sẽ ảnh hưởng đến toàn bộ cấu trúc của cả nhóm.
- Do đó, K-mean có thể hiểu là thuật toán phân nhóm 1 tập Object dựa trên các thuộc tính giống nhau và khác nhau thuộc tính của chúng.

4.4 Thuật toán Support vector machine (SVM)

Định nghĩa:

- Là một khái niệm trong thống kê và khoa học máy tính cho một tập hợp các phương pháp học có giám sát liên quan đến nhau để phân loại và phân tích hồi quy.

- SVM dạng chuẩn nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau. Do đó SVM là một thuật toán phân loại nhị phân.
- Với các bộ ví dụ luyện tập thuộc hai thể loại cho trước, thuật toán luyện tập SVM xây dựng một mô hình SVM để phân loại các ví dụ khác vào hai thể loại đó. Một mô hình SVM là một cách biểu diễn các diểm trong không gian và lựa chọn ranh giới giữa hai thể loại sao cho khoảng cách từ các ví dụ luyện tập tới ranh giới là xa nhất có thể. Các ví dụ mới cũng được biểu diễn trong cùng một không gian và được thuật toán dự đoán thuộc một trong hai thể loại tùy vào ví dụ đó ở phía nào của ranh giới.

4.5 Thuật toán Neural Network (NN)

Định nghĩa:

- Neural là tính từ của neuron (nơ-ron), network chỉ cấu trúc đồ thị nên neural network (NN) là một hệ thống tính toán lấy cảm hứng từ sự hoạt động của các nơ-ron trong hệ thần kinh và cách nó hoạt động, chứ không phải bắt chước toàn bộ các chức năng của hệ thần kinh.
- Việc cần làm là dùng mô hình hệ thần kinh đi giải quyết các bài toán chúng ta cần.

Cách đào tạo:

- Bằng cách xử lý các ví dụ và mỗi ví dụ chứa “input” và “output” đã cho, để tạo thành các liên kết có trọng số xác suất giữa hai mạng này, được lưu trữ trong cấu trúc dữ liệu của chính mạng đó.
- Việc đào tạo NN từ một ví dụ nhất định thường được tiến hành bằng cách xác định sự khác biệt giữa đầu ra đã xử lý của mạng (thông thường là một dự đoán) và đầu ra mục tiêu. Điều này là sẽ gây ra lỗi trong dự đoán. Sau đó, mạng sẽ điều chỉnh các liên kết cá trọng số của nó theo một quy tắc học tập và sử dụng giá trị lỗi này.
- Các điều chỉnh liên tiếp sẽ khiến NN tạo ra output ngày càng giống với output target.
- Sau khi có đủ số điều chỉnh thích hợp, khóa đào tạo có thể được chấm dứt dựa trên những tiêu chí nhất định đã được đề ra trước đó.
- NN thực hiện các nhiệm vụ bằng cách xem xét các ví dụ mà không được lập trình các quy tắc cụ thể cho nhiệm vụ ví dụ như nhận diện hình ảnh.

Mô hình:

- Bắt đầu từ một nổ lực khai thác kiến trúc của bộ não con người để thực hiện các nhiệm vụ mà các thuật toán thông thường không mấy thành công.

- Được định hướng cải thiện các kết quả thực nghiệm, chủ yếu từ bỏ các nổ lực để duy trì đúng với tiền chất sinh học. Các tế bào thần kinh được kết nối với nhau theo nhiều kiểu khác nhau, để cho phép đầu ra của một số tế bào thàn kinh trở thành đầu vào của những tế bào khác. Mạng tạo thành một đồ thị có hướng, có trọng số.
- Một mạng nơ-ron nhân tạo bao gồm một tập hợp các nơ-ron mô phỏng. Mỗi nơ-ron là một nút được kết nốt với các nút khác thông qua các liên kết tương ứng với các kết nối sợi trục khớp thần kinh, sợi trục sinh học. Một liên kết có một trọng số, xác định sức mạnh ảnh hưởng của một nút đối với nút khác.

Các thành phần của NN:

- **Tế bào thần kinh:** bao gồm các tế bào thần kinh nhân tạo có nguồn gốc khái niệm từ các tế bào thần kinh sinh học.
- **Kết nối và trọng lượng:** mạng bao gồm các kết nối, mỗi kết nối cung cấp đầu ra của một tế bào thần kinh và làm đầu vào cho một tế bào thần kinh khác.
- **Tổ chức:** các tế bào thần kinh thường được tổ chức thành nhiều lớp, đặc biệt là trong học sâu. Các tế bào thần kinh của một lớp chỉ kết nối với các tế bào thần kinh của các lớp ngay trước và ngay sau đó.
- **Siêu tham số:** là một tham số không đối có giá trị được đặt trước khi quá trình học bắt đầu. Giá trị của các tham số được suy ra thông qua học tập.
- **Học tập:** là sự thích ứng của mạng để xử lý một nhiệm vụ tốt hơn bằng cách xem xét các quan sát mẫu. Việc học bao gồm việc điều chỉnh trọng số (và các ngưỡng tùy chọn) của mạng để cải thiện độ chính xác của kết quả.

5. Thực hiện và kết quả

5.1 Mô tả tập dữ liệu

Tập dữ liệu mà nhóm nghiên cứu sử dụng là Credit Card Fraud Detection của Machine Learning Group – ULB. Tập dữ liệu có bối cảnh là các công ty thẻ tín dụng có thể nhận ra các giao dịch gian lận để khách hàng của họ không phải chi trả cho các mặt hàng mà họ không mua. Tập dữ liệu chứa các giao dịch được thực hiện bởi người dùng thẻ tín dụng ở châu Âu trong tháng 09/2013. Tập dữ liệu này trình bày các giao dịch xảy ra trong hai ngày, có 492 giao dịch gian lận trong số 284.807 giao dịch. Tập dữ liệu rất mất cân bằng, positive class (gian lận) chiếm 0,172% tổng số giao dịch. Dưới đây là 5 dòng dữ liệu mẫu của tập dữ liệu:

```
# Kích thước của dataset
print("Credit Card Fraud Detection data -  rows:",transactions.shape[0]," columns:", transactions.shape[1])
```

```
Credit Card Fraud Detection data -  rows: 284807  columns: 31
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
59374	48811.0	1.211467	0.165640	0.472042	1.032958	-0.099906	0.174685	-0.143792	0.087779	0.213612	-0.010794	0.452212	1.229415	0.435334
32027	36611.0	-0.840778	0.935760	1.503474	0.129457	-0.349554	-0.583203	0.709837	0.049221	0.310527	-0.154802	-1.015387	-0.524254	-1.372489
206800	136378.0	1.883602	-0.808253	-1.849872	-0.764916	1.758524	3.735088	-1.016969	0.989247	0.861279	-0.009335	0.001254	0.431651	-0.019172
221433	142569.0	2.002046	-0.070866	-1.845985	0.343959	0.602313	-0.773681	0.602403	-0.433046	0.148602	0.027594	-0.718873	1.059588	1.380692
42258	41016.0	-1.283617	0.788903	1.116795	1.530067	-0.144005	-0.364828	0.111039	0.531096	-0.242631	-0.172977	-0.627450	-0.076126	-1.072716

V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
9991	0.103901	-0.601382	0.111944	0.597100	-0.102969	-0.261379	-0.569661	-0.056388	-0.366827	0.575964	-0.520361	0.039664	0.010803	1.00	0
4116	-0.370140	0.115375	-0.391553	0.363592	0.144844	-0.233974	-0.398455	-0.095359	0.386643	0.001768	0.248674	0.428421	0.232806	40.63	0
3402	0.344230	-0.582177	-0.618850	-0.178475	0.001319	-0.230978	-0.806520	0.432593	0.692862	-0.630443	0.247344	-0.030556	-0.040340	63.93	0
9663	-0.624671	-0.167210	-1.027237	0.160834	-0.013085	0.004029	0.151261	0.041610	0.705762	0.230279	0.527307	-0.098780	-0.059568	55.95	0
3446	-1.138455	0.952393	-0.625531	0.640571	-0.119733	0.005222	0.269849	0.304351	0.392463	-0.065117	-0.236086	0.221314	0.095940	17.18	0

Các thuộc tính V1 đến V28 là các thành phần quan trọng, do vấn đề về bảo mật nên được chuyển đổi với PCA. Các thuộc tính duy nhất không được chuyển đổi với PCA là "Time" và "Amount". Thuộc tính "Time" là số giây trôi qua giữa mỗi giao dịch và giao dịch đầu tiên trong tập dữ liệu. Thuộc tính 'Số tiền' là số tiền giao dịch. Thuộc tính "Class" là biến phản hồi và nó nhận giá trị 1 trong trường hợp gian lận và 0 nếu không phải là giao dịch gian lận.

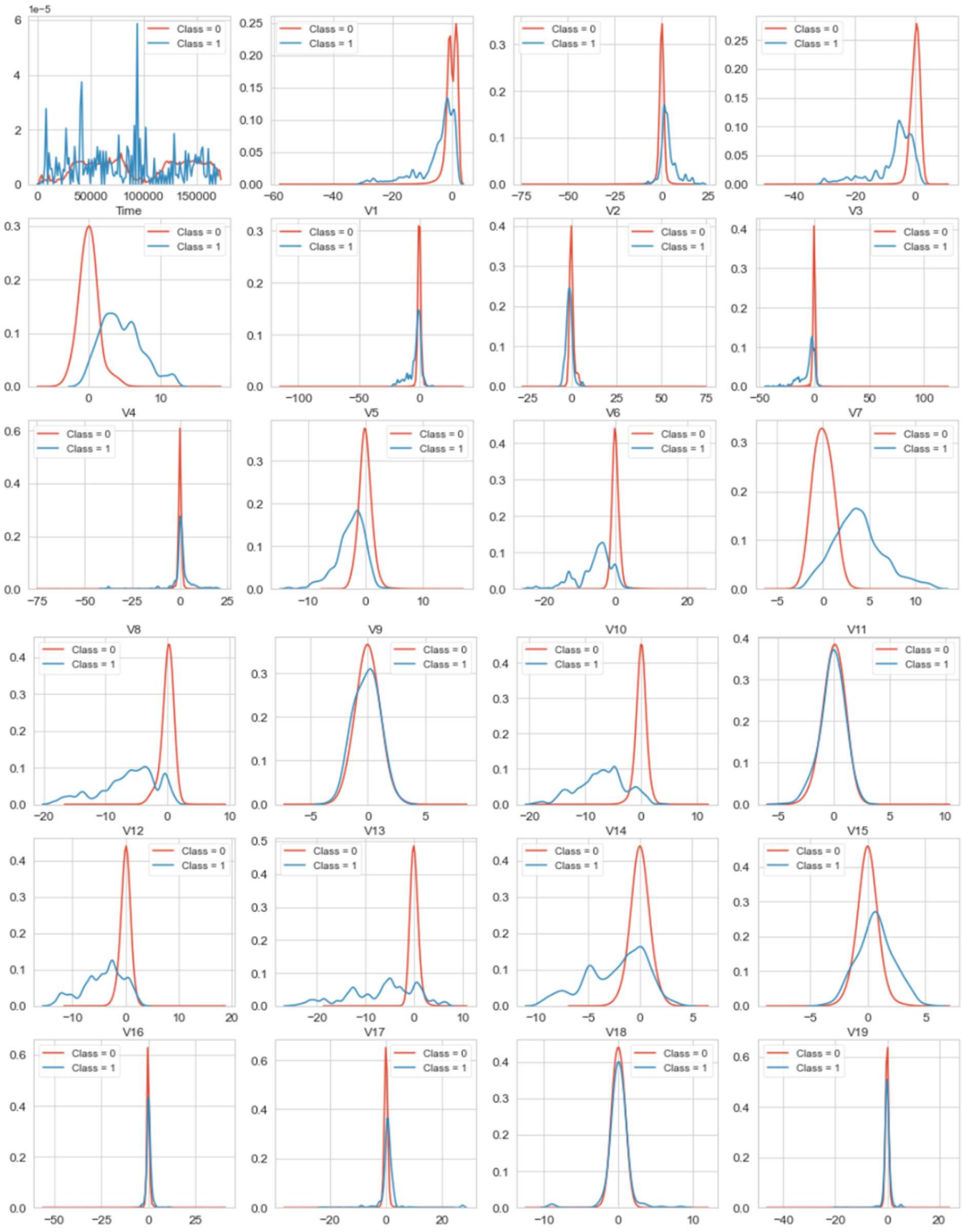
Độ tương quan giữa các thuộc tính dựa trên thuộc tính "Class":

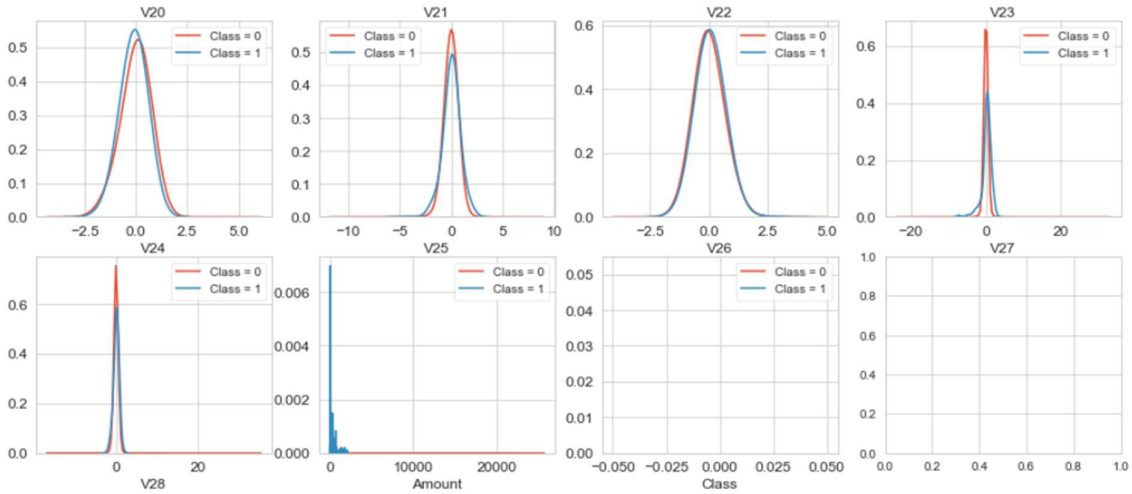
```
# Biểu đồ mật độ của các thuộc tính dựa trên thuộc tính Class
var = transactions.columns.values

i = 0
t0 = transactions.loc[transactions['Class'] == 0]
t1 = transactions.loc[transactions['Class'] == 1]

sns.set_style('whitegrid')
plt.figure()
fig, ax = plt.subplots(8,4,figsize=(16,28))

for feature in var:
    i += 1
    plt.subplot(8,4,i)
    sns.kdeplot(t0[feature], bw=0.5,label="Class = 0")
    sns.kdeplot(t1[feature], bw=0.5,label="Class = 1")
    plt.xlabel(feature, fontsize=12)
    locs, labels = plt.xticks()
    plt.tick_params(axis='both', which='major', labelsize=12)
plt.show();
```


1e−5
Class = 0
Class = 1
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19



Quan sát các đồ thị ứng với các thuộc tính, chúng ta có thể thấy độ chọn lọc tốt về mặt phân bố cho hai giá trị của thuộc tính "Class" như sau: V4, V11 có sự phân bố tách biệt rõ ràng cho các giá trị 0 và 1 của thuộc tính "Class", tiếp đó là V12, V14, V18. Với V1, V2, V3, V10 thì có phân bố khá khác biệt, còn với V25, V26, V28 thì có độ phân bố tương đối đồng đều cho hai giá trị của thuộc tính "Class".

Nhìn chung, chỉ với một số ngoại lệ (Thời gian và Số tiền), mật độ phân bố đối với các giao dịch hợp pháp (giá trị của "Class" = 0) tập trung xung quanh 0, đôi khi kéo dài ở một số các điểm cực trị. Ngược lại, các giao dịch gian lận (giá trị của "Class" = 1) có phân bố lệch (không đối xứng).

Mức độ ảnh hưởng của các thuộc tính đến kết quả dự đoán:

```
X = transactions.drop(labels='Class', axis=1) # Features
y = transactions.loc[:,'Class']               # Response
del transactions                              # Xóa dữ liệu lúc đầu

# Tách dataset thành 80% cho training và 20% cho test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
del X, y
```

```
from sklearn.feature_selection import mutual_info_classif

# Tính toán sự ảnh hưởng của các thuộc tính đến kết quả dự đoán
mutual_infos = pd.Series(data=mutual_info_classif(X_train, y_train, discrete_features=False, random_state=1), index=X_train.colum
```

```
# Kết quả của tương quan của mỗi thuộc tính với thuộc tính Class
mutual_infos.sort_values(ascending=False)
```

```
V17       0.008037
V14       0.007977
V10       0.007354
V12       0.007354
V11       0.006607
V16       0.005793
V4        0.004843
V3        0.004755
V18       0.004025
V9        0.003996
V7        0.003941
V2        0.003085
V21       0.002304
V27       0.002271
V6        0.002265
V5        0.002254
V1        0.001990
V8        0.001843
V28       0.001757
Time      0.001722
Amount    0.001388
V19       0.001322
V20       0.001136
V23       0.000827
V24       0.000593
V26       0.000459
V22       0.000388
V25       0.000376
V15       0.000230
V13       0.000205
dtype: float64
```

Từ các đồ thị và kết quả trên chúng ta có thể thấy được 5 thuộc tính có ảnh hưởng lớn nhất đến kết quả dự đoán lần lượt là: V17, V14, V10, V12 và V11.

5.2 Các phương pháp đánh giá áp dụng

- Confusion matrix giúp có cái nhìn rõ hơn về việc các điểm dữ liệu được phân loại đúng/sai như thế nào.

	Predicted as Positive	Predicted as Negative
Actual: Positive	True Positive (TP)	False Negative (FN)
Actual: Negative	False Positive (FP)	True Negative (TN)

- True Positive (TP): số lượng điểm của lớp positive được phân loại đúng là positive.
- True Negative (TN): số lượng điểm của lớp negative được phân loại đúng là negative.
- False Positive (FP): số lượng điểm của lớp negative bị phân loại nhầm thành positive.
- False Negative (FN): số lượng điểm của lớp positive bị phân loại nhầm thành negative.
- Khi kích thước các lớp dữ liệu là chênh lệch (imbalanced data hay skew data), precision và recall thường được sử dụng:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

+ Precision được định nghĩa là tỉ lệ số điểm true positive trong số những điểm được phân loại là positive (TP + FP).

+ Recall được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive (TP + FN).

- F1 score: là harmonic mean của precision và recall, có giá trị nằm trong nửa khoảng (0,1]. F1 càng cao, bộ phân lớp càng tốt.

$$F_1 = 2\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2\frac{\text{precion} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Micro-average precision, micro-average recall:

$$\text{micro-average precision} = \frac{\sum_{c=1}^{C} \text{TP}c}{\sum_{c=1}^{C} (\text{TP}c + \text{FP}c)} \quad \text{micro-average recall} = \frac{\sum_{c=1}^{C} \text{TP}c}{\sum_{c=1}^{C} (\text{TP}c + \text{FN}c)}$$

với $\text{TP}c, \text{FP}c, \text{FN}c$ lần lượt là TP, FP, FN của class c.

- Micro-average precision, macro-average recall là trung bình cộng của các precision, recall cho từng lớp. Micro-average (macro-average) F1 scores cũng được tính dựa trên các micro-average (macro-average) precision, recall tương ứng.
- Accuracy là tỉ lệ giữa số điểm được phân loại đúng và tổng số điểm. Accuracy chỉ phù hợp với các bài toán mà kích thước các lớp dữ liệu là tương đối như nhau.

5.3 Thuật toán Random Forest

5.3.1 Thực hiện

```
# Phân tách dataset
X = transactions.drop(labels='Class', axis=1) # Features
y = transactions.loc[:,'Class']               # Response
del transactions                              # Xóa dữ liệu lúc đầu

# Tách dataset thành 80% cho training và 20% cho test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
del X, y
```

```
# Khai báo thư viện RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
# Training model
classifier = RandomForestClassifier(n_estimators=50)
classifier.fit(X_train, y_train)
```

```
# Dự đoán cho bộ dữ liệu test
y_pred = classifier.predict(X_test)
```

5.3.2 Kết quả

```
# Kết quả dự đoán
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[56858     6]
 [   14    84]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.93      0.86      0.89        98

    accuracy                           1.00     56962
   macro avg       0.97      0.93      0.95     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy: 0.9996488887328394
```

_ Ma trận nhầm lẫn (Confusion Matrix):

- Kết quả thu được sau khi dự đoán cho 20% (56962) dữ liệu còn lại của tập dữ liệu ta có kết quả của ma trận nhầm lẫn là:

  [[56858 6]

  [14 84]]

- Trong đó số các giao dịch bất thường được dự đoán đúng là 84, dự đoán sai là 14. Số giao dịch bình thường được dự đoán đúng là 56858, dự đoán sai là 6.

_ Classification report:

Chỉ số precision, recall, f1-score đối với các giao dịch bình thường (có nhãn là 0) đều là 1.0. Điều này có nghĩa là thuật toán dự đoán rất tốt và hầu như không có nhầm lẫn giao dịch bất thường thành giao dịch bình thường.

Chỉ số precision, recall, f1-score đối với các giao dịch bất thường (có nhãn là 1) lần lượt là 0.93, 0.86, 0.89. Điều này cho thấy là thuật toán dự đoán khá tốt các trường hợp giao dịch bất thường dù vẫn còn có dự đoán sai. Nhưng nhìn một cách tổng quan hơn thì các giao dịch bất thường chỉ chiếm 0.172% trong tập dữ liệu. Do đó thuật toán đạt các chỉ số đều trên 0.85 là một điều khá tốt.

Thuật toán có độ chính xác là 0.9996488887328394. Được tính dựa trên số điểm được phân loại đúng và tổng số điểm.

5.4 Thuật toán K-Nearest Neighbors (KKN)

5.4.1 Thực hiện

```
# Phân tách dataset
X = transactions.drop(labels='Class', axis=1) # Features
y = transactions.loc[:,'Class']               # Response
del transactions                              # Xóa dữ liệu lúc đầu

# Tách dataset thành 80% cho training và 20% cho test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
del X, y
```


```
# Phân tách dataset
X = transactions.drop(labels='Class', axis=1) # Features
y = transactions.loc[:,'Class']               # Response
del transactions                              # Xóa dữ liệu lúc đầu

# Tách dataset thành 80% cho training và 20% cho test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
del X, y
```


```
# Chuẩn hóa dữ liệu
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Training model
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 4)
classifier.fit(X_train, y_train)
```

```
# Dự đoán cho bộ dữ liệu test
y_pred = classifier.predict(X_test)
```

5.4.2 Kết quả

```
# Kết quả dự đoán
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[56859     5]
 [   22    76]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.94      0.78      0.85        98

    accuracy                           1.00     56962
   macro avg       0.97      0.89      0.92     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy: 0.9995259997893332
```

_ Ma trận nhầm lẫn (Confusion Matrix):

- Kết quả thu được sau khi dự đoán cho 20% (56962) dữ liệu còn lại của tập dữ liệu ta có kết quả của ma trận nhầm lẫn là:

  [[56859 5]

  [22 76]]

- Trong đó số các giao dịch bất thường được dự đoán đúng là 76, dự đoán sai là 22. Số giao dịch bình thường được dự đoán đúng là 56859, dự đoán sai là 5.

_ Classification report:

Chỉ số precision, recall, f1-score đối với các giao dịch bình thường (có nhãn là 0) đều là 1.0. Điều này có nghĩa là thuật toán dự đoán rất tốt và hầu như không có nhầm lẫn giao dịch bất thường thành giao dịch bình thường.

Chỉ số precision, recall, f1-score đối với các giao dịch bất thường (có nhãn là 1) lần lượt là 0.94, 0.78, 0.85. Điều này cho thấy là thuật toán dự đoán chỉ ở mức tạm ổn các trường hợp giao dịch bất thường vì có số dự đoán sai khá nhiều.

Thuật toán có độ chính xác là 0.9995259997893332. Được tính dựa trên số điểm được phân loại đúng và tổng số điểm.

5.5 Thuật toán Support Vector Machine (SVM)

5.5.1 Thực hiện

```
# Chọn lọc dữ liệu
transactions_1 = transactions_total[transactions_total['Class'] == 1] # Lấy ra các giao dịch bất thường
transactions_0 = transactions_total[transactions_total['Class'] == 0] # Lấy ra các giao dịch bình thường
print('Trong dataset, có ' + str(len(transactions_1)) +" giao dịch bất thường do đó chúng ta sẽ lấy số lượng tương tự đối với cá

transactions_temp = transactions_0.sample(500)
transactions = transactions_1.append(transactions_temp) # Gộp các giao dịch bất thường và bình thường với nhau
transactions = transactions.sample(frac=1) # Trộn thứ tự các record của dataset

transactions.info()
```

```
# Phân tách dataset
X = transactions.drop(labels='Class', axis=1) # Features
y = transactions.loc[:,'Class']               # Response
del transactions                              # Xóa dữ liệu lúc đầu

# Tách dataset thành 80% cho training và 20% cho test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
del X, y
```

```
from sklearn import svm

classifier = svm.SVC(kernel='linear') # Thiết lập SVM classifier
```

```
classifier.fit(X_train, y_train) # Train model
```

```
# Dự đoán cho bộ dữ liệu test
y_pred = classifier.predict(X_test)
```

5.5.2 Kết quả

```
# Kết quả dự đoán
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test,y_pred)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[98  2]
 [28 71]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.98      0.87       100
           1       0.97      0.72      0.83        99

    accuracy                           0.85       199
   macro avg       0.88      0.85      0.85       199
weighted avg       0.87      0.85      0.85       199

Accuracy: 0.8492462311557789
```

_ Do thiếu điều kiện về thiết bị và thời gian nên nhóm nghiên cứu thu gọn tập dữ liệu lại để có thể demo được thuật toán này. Do tập dữ liệu ban đầu có 492 giao dịch bất thường vì thế nhóm sẽ lấy số lượng tương tự đối với các giao dịch bình thường là 500. Vậy tập dữ liệu mới tổng cộng có 992 giao dịch.

_ Ma trận nhầm lẫn (Confusion Matrix):

- Kết quả thu được sau khi dự đoán cho 20% (199) dữ liệu còn lại của tập dữ liệu ta có kết quả của ma trận nhầm lẫn là:

  [[98 2]

  [28 71]]

- Trong đó số các giao dịch bất thường được dự đoán đúng là 71, dự đoán sai là 28. Số giao dịch bình thường được dự đoán đúng là 98, dự đoán sai là 2.

_ Classification report:

Chỉ số precision, recall, f1-score đối với các giao dịch bình thường (có nhãn là 0) lần lượt là 0.78, 0.98, 0.87.

Chỉ số precision, recall, f1-score đối với các giao dịch bất thường (có nhãn là 1) lần lượt là 0.97, 0.72, 0.83. Điều này cho thấy là thuật toán dự đoán chỉ ở mức tạm ổn các trường hợp giao dịch bất thường vì có số dự đoán sai khá nhiều.

Thuật toán có độ chính xác là 0.8492462311557789. Được tính dựa trên số điểm được phân loại đúng và tổng số điểm.

5.6 Thuật toán Neural Network (NN)

5.6.1 Thực hiện

```
# Phân tách dataset
X = transactions.drop(labels='Class', axis=1) # Features
y = transactions.loc[:,'Class']               # Response
del transactions                              # Xóa dữ liệu lúc đầu

# Tách dataset thành 80% cho training và 20% cho test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
del X, y
```

```
# Chuẩn hóa dữ liệu
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Khai báo thư viện cần thiết dùng cho Neural Network
from keras import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
```

```
NumerOfClasses = len(y_train.unique())
NumerOfClasses
```

```
RN = Sequential() # Tạo network structure
RN.add(Dense(10, input_shape = X_train.shape[1:], activation ='sigmoid'))
RN.add(Dense(NumerOfClasses, activation ='sigmoid'))
RN.summary()
```

```
# Train model
from keras.utils import to_categorical
sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9)
RN.compile(optimizer=sgd, loss='mean_squared_error', metrics=['accuracy'])
trainedRN = RN.fit(X_train, to_categorical(y_train), epochs=2, verbose=1)
```

```
# Dự đoán cho bộ dữ liệu test
import numpy as np
y_pred = RN.predict(X_test)
y_pred_rounded = np.argmax(y_pred.round(), axis=1)
```

5.6.1 Kết quả

```
# Kết quả dự đoán
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
result = confusion_matrix(y_test, y_pred_rounded)
print("Confusion Matrix:")
print(result)
result1 = classification_report(y_test, y_pred_rounded)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(y_test, y_pred_rounded)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[56853    11]
 [   16    82]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.88      0.84      0.86        98

    accuracy                           1.00     56962
   macro avg       0.94      0.92      0.93     56962
weighted avg       1.00      1.00      1.00     56962

Accuracy: 0.9995259997893332
```

_ Ma trận nhầm lẫn (Confusion Matrix):

- Kết quả thu được sau khi dự đoán cho 20% (56962) dữ liệu còn lại của tập dữ liệu ta có kết quả của ma trận nhầm lẫn là:

  [[56853 11]

  [16 82]]

- Trong đó số các giao dịch bất thường được dự đoán đúng là 82, dự đoán sai là 16. Số giao dịch bình thường được dự đoán đúng là 56853, dự đoán sai là 11.

_ Classification report:

Chỉ số precision, recall, f1-score đối với các giao dịch bình thường (có nhãn là 0) đều là 1.0. Điều này có nghĩa là thuật toán dự đoán rất tốt và hầu như không có nhầm lẫn giao dịch bất thường thành giao dịch bình thường.

Chỉ số precision, recall, f1-score đối với các giao dịch bất thường (có nhãn là 1) lần lượt là 0.88, 0.84, 0.86. Điều này cho thấy là thuật toán dự đoán khá tốt các trường hợp giao dịch bất thường dù vẫn còn có dự đoán sai. Nhưng nhìn một cách tổng quan hơn thì các giao dịch bất thường chỉ chiếm 0.172% trong tập dữ liệu. Do đó thuật toán đạt các chỉ số đều trên 0.84 là một điều khá tốt.

Thuật toán có độ chính xác là 0.9995259997893332. Được tính dựa trên số điểm được phân loại đúng và tổng số điểm.

6. So sánh thuật toán

Tổng số giao dịch = 56962	Random Forest	KNN	SVM (199 giao dịch)	NN
Số giao dịch bất thường dự đoán đúng	**84/98***	76/98	71/99	82/98
Số giao dịch bình thường dự đoán đúng	56858/56864	**56859/56864***	98/100	56853/56864
Precision (class = 1)	0.93	0.94	**0.97***	0.88
Recall (class = 1)	**0.86***	0.78	0.72	0.84
F1-score (class = 1)	**0.89***	0.85	0.83	0.86

7. Kết luận

Từ kết quả trên chúng ta có thể thấy số lượng kết quả dự đoán đúng các giao dịch bất thường của thuật toán Random Forest cao hơn so với các thuật toán còn lại. Hơn hết, đối với các tập dữ liệu có độ lệch cao thì thuật toán nào có f1-score cao hơn thì tốt hơn. Vì thế, mức độ hiệu quả trong việc dự đoán các giao dịch bất thường của các thuật toán lần lượt là **Random Forest (0.89*)** → Neural Network (0.86) → K-Nearest Neigbors (0.85)

→ Support Vector Machine (0.83). Vậy thuật toán **Random Forest** là thuật toán phù hợp nhất cho việc dự đoán các giao dịch bất thường.

Tài liệu tham khảo

1. Tin tức về giao dịch bất thường:
   https://vnexpress.net/tai-khoan-ngan-hang-boc-hoi-406-trieu-trong-vai-phut-4171383.html
   https://vnexpress.net/tai-khoan-ngan-hang-boc-hoi-100-trieu-dong-4064219.html
   https://vnexpress.net/chu-the-mat-29-trieu-dong-trong-dem-3993936.html
2. Tổng quan về các thuật toán xác định giao dịch bất thường:
   https://perfectial.com/blog/fraud-detection-machine-learning/
3. Thuật toán Decision Tree:
   https://www.guru99.com/r-decision-trees.html
   https://en.wikipedia.org/wiki/Decision_tree
4. Thuật toán Random Forest:
   https://medium.com/@thanhleo92/random-forest-và-ứng-dụng-b6965c1f0634
   https://www.guru99.com/r-random-forest-tutorial.html
   https://en.wikipedia.org/wiki/Random_forest
5. Sách Hands-On Machine Learning with Scikit-Learn and TensorFlow, Aurélien Géron
6. Thuật toán K-Nearest Neighbors:
   https://scikitlearn.org/stable/modules/neighbors.html?fbclid=IwAR3uPsg6MkUD71Kf8Qz-M20M8lbulOpslX89wsXr_hYETTueE9fd3SQ9OTs
   https://machinelearningcoban.com/2017/01/08/knn/
   https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm
7. Thuật toán Support vector machine (SVM):
   https://machinelearningcoban.com/2017/04/09/smv/
   https://www.tutorialspoint.com/machine_learning_with_python/classification_algorithms_support_vector_machine.htm
8. Thuật toán Neural Network (NN):
   https://nttuan8.com/bai-3-neural-network/
   https://scikit-learn.org/stable/modules/neural_networks_supervised.html
9. Các phương pháp đánh giá:
   https://machinelearningcoban.com/2017/08/31/evaluation/

Thông tin nhóm chịu trách nhiệm bài viết

1. Họ tên: Ôn Đức Khang

Mã số sinh viên: 17110310

Email: 17110310@student.hcmute.edu.vn

2. Họ tên: Lê Minh Tiến

Mã số sinh viên: 17110235

Email: 17110235@student.hcmute.edu.vn