

BÁO CÁO TIẾN ĐỘ TUẦN 1

Thử thách tính kỷ luật Business AI Lab

Tóm tắt nhiệm vụ:

- Học Machine Learning, Deep Learning .
- Làm bài tập thầy giao trong nhiệm vụ thử thách.

Tóm tắt tiến độ:

- Hoàn thành tất cả nhiệm vụ thầy giao.

Kế hoạch tuần tới:

- Đón nhận nhiệm vụ mới từ thầy.

Kết luận/ Đánh giá:

- Em đánh giá thời gian vừa rồi em làm việc chưa hiệu quả nhưng có thời gian lên Lab thường xuyên .
- Em thấy môi trường Labs làm việc cực kì chuyên nghiệp. Mọi người trong Lab vui vẻ, hòa đồng, sẵn sàng giúp đỡ em cộng thêm thái độ làm việc nghiêm túc chăm chỉ đã thúc đẩy em cố gắng hơn trong quá trình thử thách.

Github: <https://github.com/ngoctien1710/Th-th-ch-BAI-Labs.git>

Báo cáo kết quả

Bài 1 : Giữ nguyên các feature, train (không scaler) với các thông số mặc định

Model	Accuracy	Precision	Recall	F1	Hyperparameter
Logistic Regression	0.89	0.91	0.75	0.82	Mặc định
Decision Tree Classifier	0.85	0.81	0.75	0.78	Mặc định
Random Forest Classifier	0.90	0.83	0.89	0.86	Mặc định
GaussianNB	0.94	0.93	0.89	0.91	Mặc định
K Neighbors Classifier	0.82	0.77	0.71	0.74	Mặc định

⇒ Các mô hình phi tuyến có hiệu quả tốt hơn so với các mô hình tuyến tính đặc biệt là GaussianNB.

Bài 1 : Loại bỏ feature Gender, tiếp tục train (không scaler) với các thông số mặc định

Model	Accuracy	Precision	Recall	F1	Hyperparameter
Logistic Regression	0.89	0.91	0.75	0.82	Mặc định
Decision Tree Classifier	0.84	0.78	0.75	0.76	Mặc định
Random Forest Classifier	0.90	0.81	0.93	0.87	Mặc định
GaussianNB	0.93	0.92	0.86	0.89	Mặc định
K Neighbors Classifier	0.82	0.77	0.71	0.74	Mặc định

⇒ Nhìn chung feature Gender có ảnh hưởng đến quyết định mua xe nhưng không quá lớn.

Bài 1 : Loại bỏ feature Age, tiếp tục train (không scaler) với các thông số mặc định

Model	Accuracy	Precision	Recall	F1	Hyperparameter
Logistic Regression	0.75	0.79	0.39	0.52	Mặc định
Decision Tree Classifier	0.70	0.59	0.46	0.52	Mặc định
Random Forest Classifier	0.72	0.62	0.54	0.58	Mặc định
GaussianNB	0.74	0.77	0.36	0.49	Mặc định
K Neighbors Classifier	0.70	0.59	0.46	0.52	Mặc định

⇒ Feature Age có tác động đáng kể, khi loại bỏ feature Age đã làm giảm rõ rệt hiệu quả của mô hình.

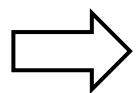
Bài 1 : Loại bỏ feature EstimatedSalary, tiếp tục train (không scaler) với các thông số mặc định

Model	Accuracy	Precision	Recall	F1	Hyperparameter
Logistic Regression	0.91	0.92	0.82	0.87	Mặc định
Decision Tree Classifier	0.84	0.94	0.57	0.71	Mặc định
Random Forest Classifier	0.85	0.94	0.61	0.74	Mặc định
GaussianNB	0.91	0.92	0.82	0.87	Mặc định
K Neighbors Classifier	0.91	0.89	0.86	0.87	Mặc định

➡ Feature EstimatedSalary có tác động tương đối đáng kể, khi loại bỏ feature EstimatedSalary đã làm giảm hiệu quả của mô hình.

Bài 2 Triển khai mô hình DNN trên tập dữ liệu MNIST với 4 lớp layers:
Flatten -> Dense(32, relu) -> Dense(16,relu) -> Dense(softmax)

Mô hình	Thông số	Accuracy – loss (tập test)
DNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Batch_size = 10 Epochs = 5	accuracy: 0.9704 - loss: 0.1029
DNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Batch_size = 16 Epochs = 8	accuracy: 0.9672 - loss: 0.1230
DNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Batch_size = 32 Epochs = 16	accuracy: 0.9677 - loss: 0.1454
DNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Batch_size = 128 Epochs = 16	accuracy: 0.9687 - loss: 0.1122



Nhìn chung mô hình có hiệu suất tốt, accuracy cao

Bài 2 Triển khai mô hình DNN trên tập dữ liệu MNIST với 4 lớp layers xen kẽ dropout(0.2):
Flatten -> Dense(512, relu) -> Dense(256, relu) -> Dense(128, relu) -> Dense(softmax)

Mô hình	Thông số	Accuracy – loss (tập test)
DNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Batch_size = 128 Epochs = 25	accuracy: 0.9803 - loss: 0.1064










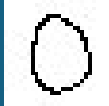
⇒ Sau khi điều chỉnh các hyperparameter trên mô hình DNN, accuracy của mô hình tăng đáng kể

Bài 2 Triển khai mô hình CNN trên tập dữ liệu MNIST

Mô hình	Thông số	Accuracy – loss (tập test)
CNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Epochs = 10 Batch_size = 64 validation_data = test	accuracy: 0.9916 - loss: 0.0460

⇒ Mô hình CNN đạt **độ chính xác (accuracy)** cao hơn đáng kể so với DNN.

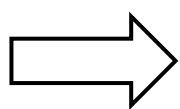
Bài 2 Sử dụng DNN và CNN nhận diện chữ viết tay

Mô hình											Kết quả đúng
DNN	1	7	3	6	1	3	3	4	5	0	4/10
CNN	1	4	8	4	7	8	5	4	5	0	7/10

⇒ CNN cho kết quả vượt trội rõ rệt so với DNN nhờ khả năng trích xuất đặc trưng không gian hiệu quả hơn.

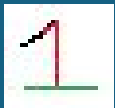


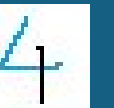






Bài 3 : xây dựng CNN và DNN với ảnh màu

Mô hình	Thông số	Accuracy – loss (tess)
DNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Batch_size = 128 Epochs = 25	accuracy: 0.9802 - loss: 0.0731
CNN	Optimizer = 'adam' Loss = 'sparse_categorical_crossentropy' Epochs = 10 Batch_size = 64 validation_data = test	accuracy: 0.9910 - loss: 0.0341



Khi áp dụng với ảnh màu (MNIST thêm cột màu giả), accuracy của hai mô hình có giảm đôi chút nhưng không đáng kể
Mô hình CNN vẫn cho hiệu suất tốt hơn so với DNN

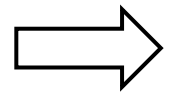
Bài 3 : xây dựng CNN và DNN với ảnh màu

Mô hình											Kết quả
DNN	3	2	3	4	5	3	3	3	1	5	4/10
CNN	1	2	3	4	5	5	7	8	7	6	7/10

➡ CNN vẫn cho kết quả vượt trội rõ rệt so hơn với DNN khi áp dụng xử lí màu

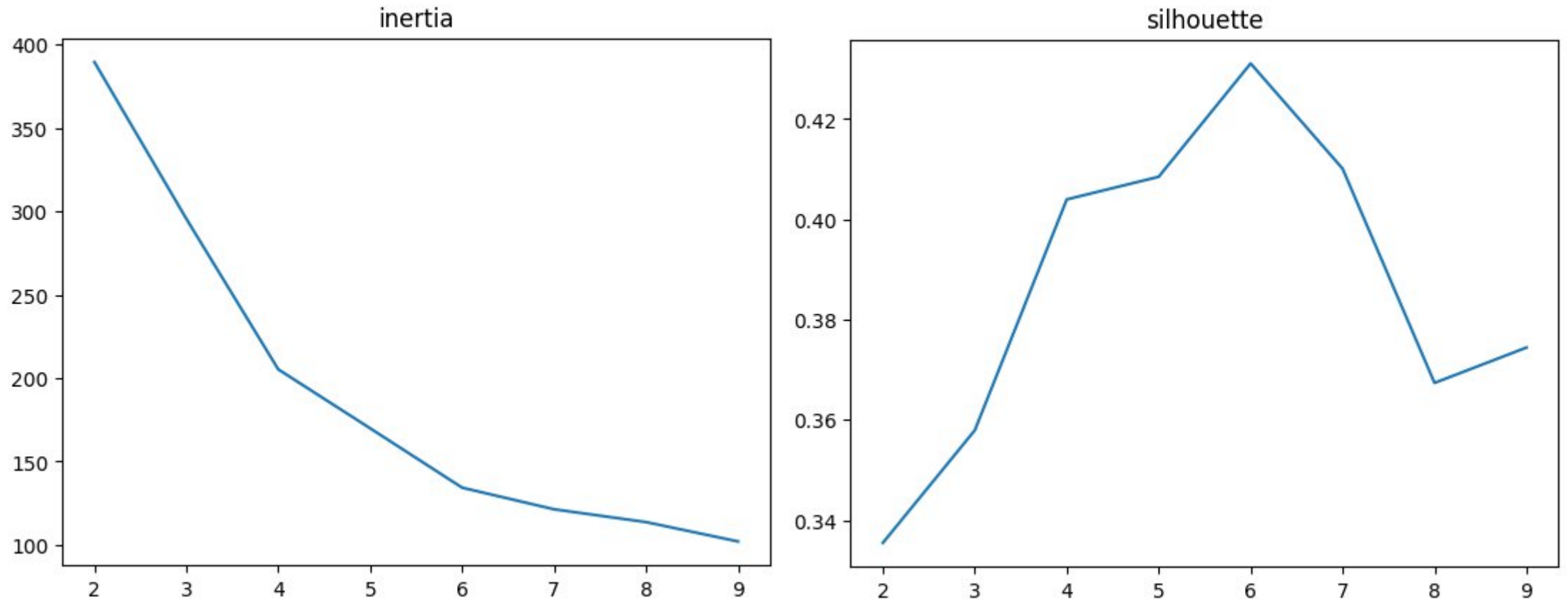
Bài 4: Bài toán Customer Segmentation (phân khúc khách hàng) “KMEAN”

Columns name	Outliers
Age	0
Annual Income (k\$)	0
Spending Score (1-100)	0



Bộ dữ liệu không tồn tại outliers, ta sẽ sử dụng 3 thuộc tính này để phân cụm

Bài 4: Bài toán Customer Segmentation (phân khúc khách hàng) “KMEAN”

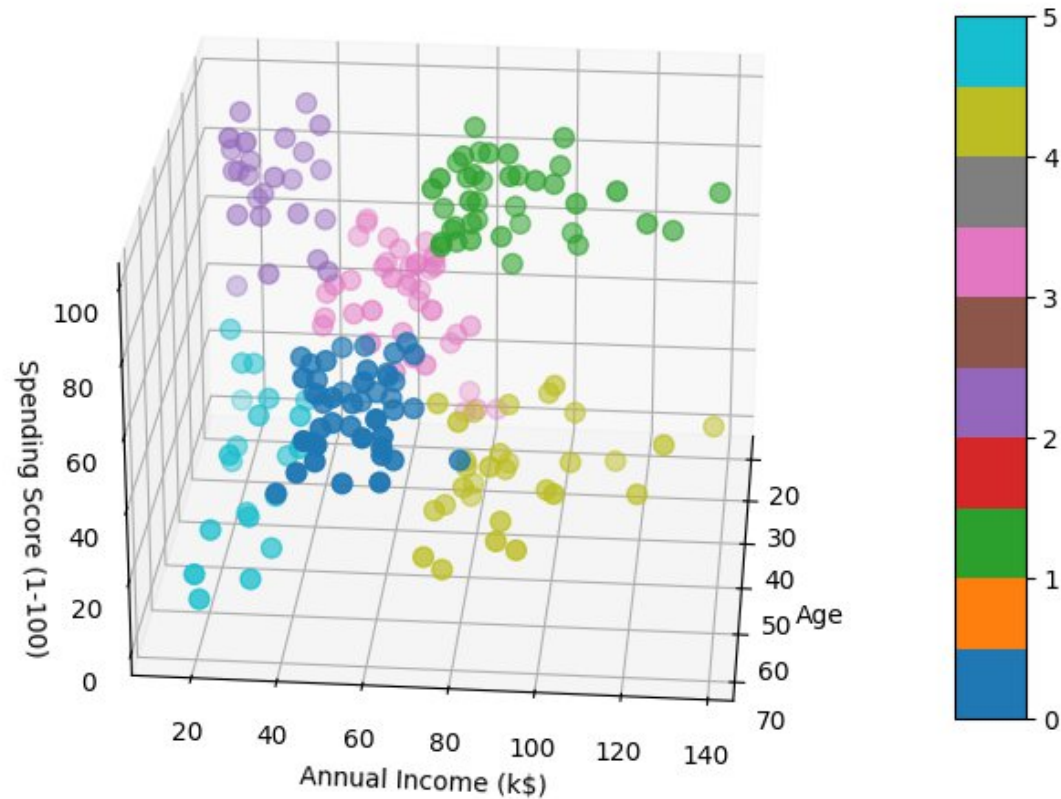


Xây dựng biểu đồ 2 phương pháp chọn số cụm trong thuật toán Kmean cho thấy số cụm bằng 6 là tốt nhất

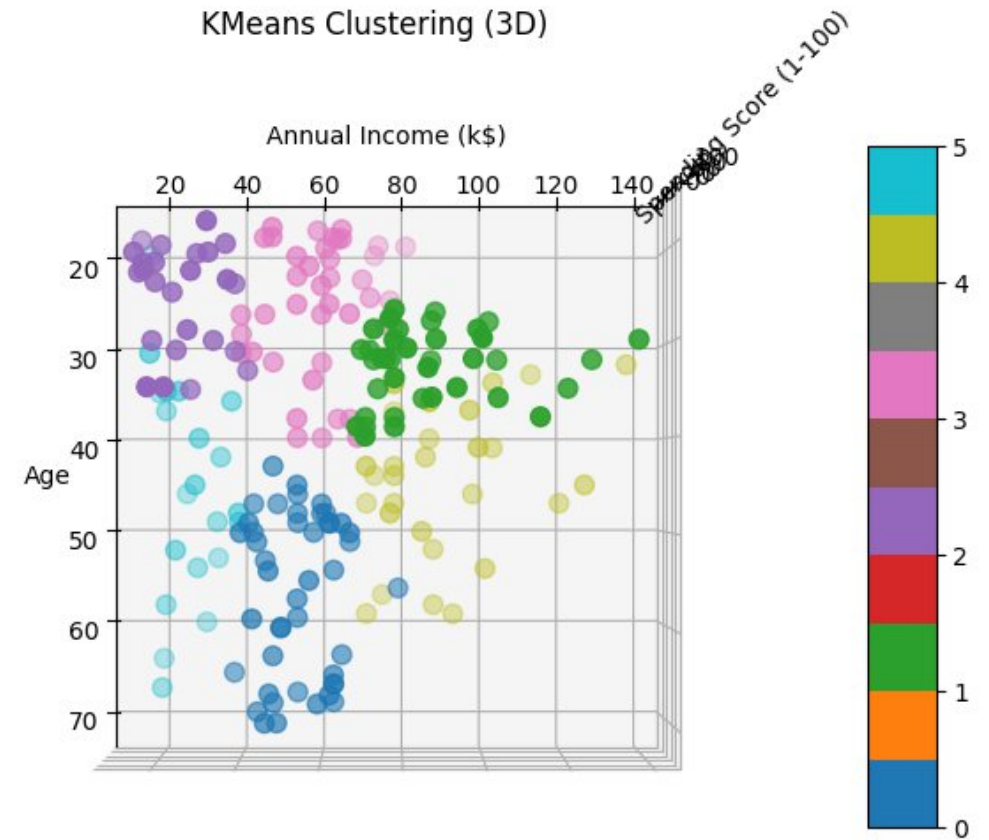
⇒ Nên ta sẽ chọn số cụm là 6

Bài 4: Bài toán Customer Segmentation (phân khúc khách hàng) “KMEAN”

KMeans Clustering (3D)

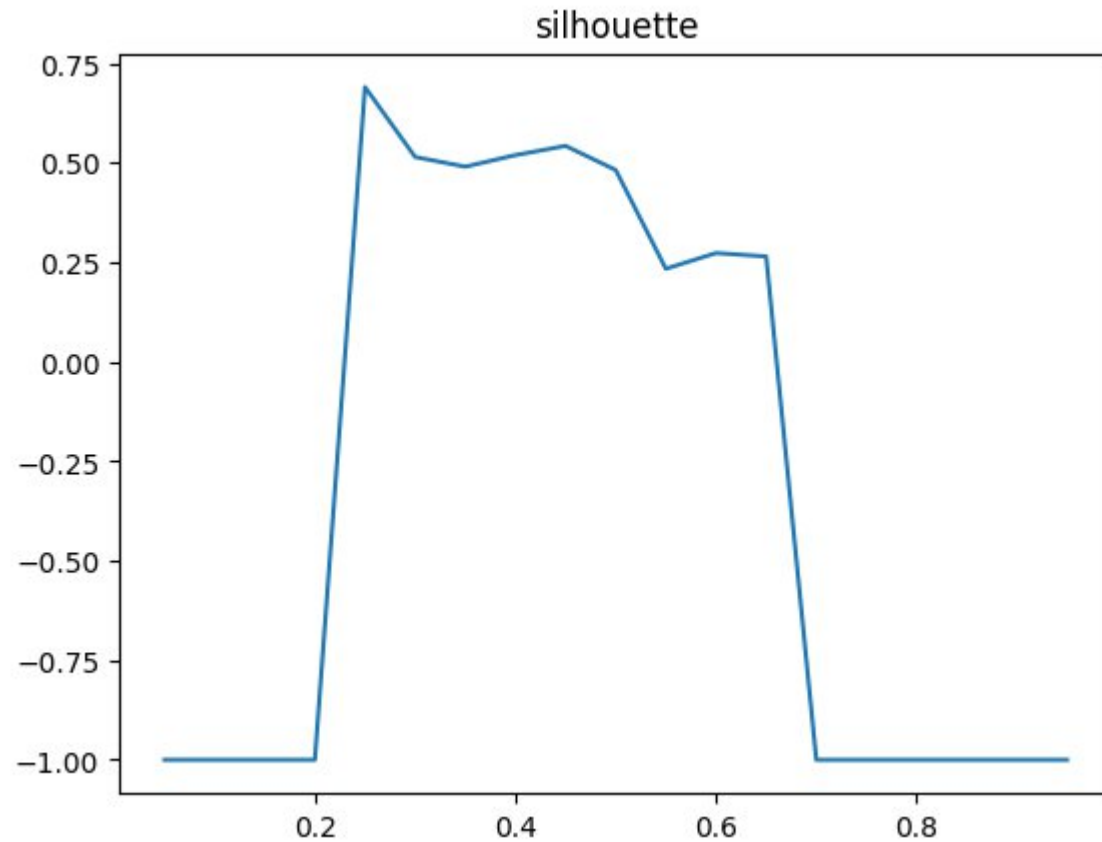


KMeans Clustering (3D)



➡ Với số cụm bằng 6 cho kết quả phân cụm rất tốt

Bài 4: Bài toán Customer Segmentation (phân khúc khách hàng) “DBSCAN”



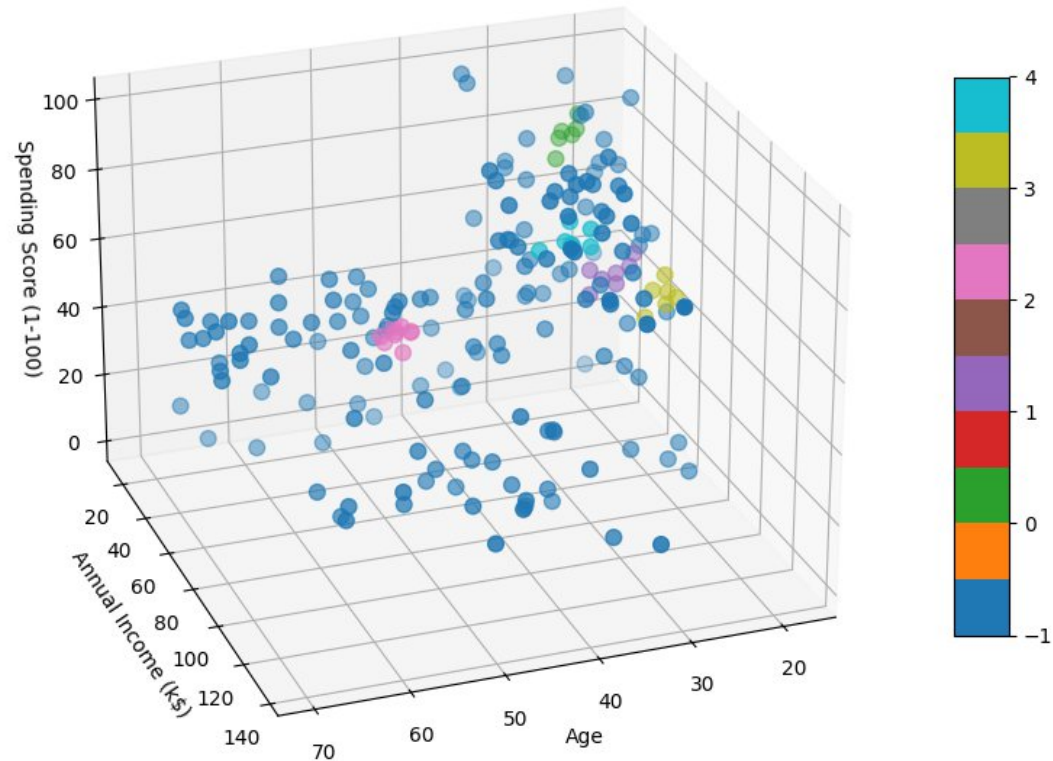
Chọn min_sample = 5

Ta tiến hành dò eps dựa vào silhouette score

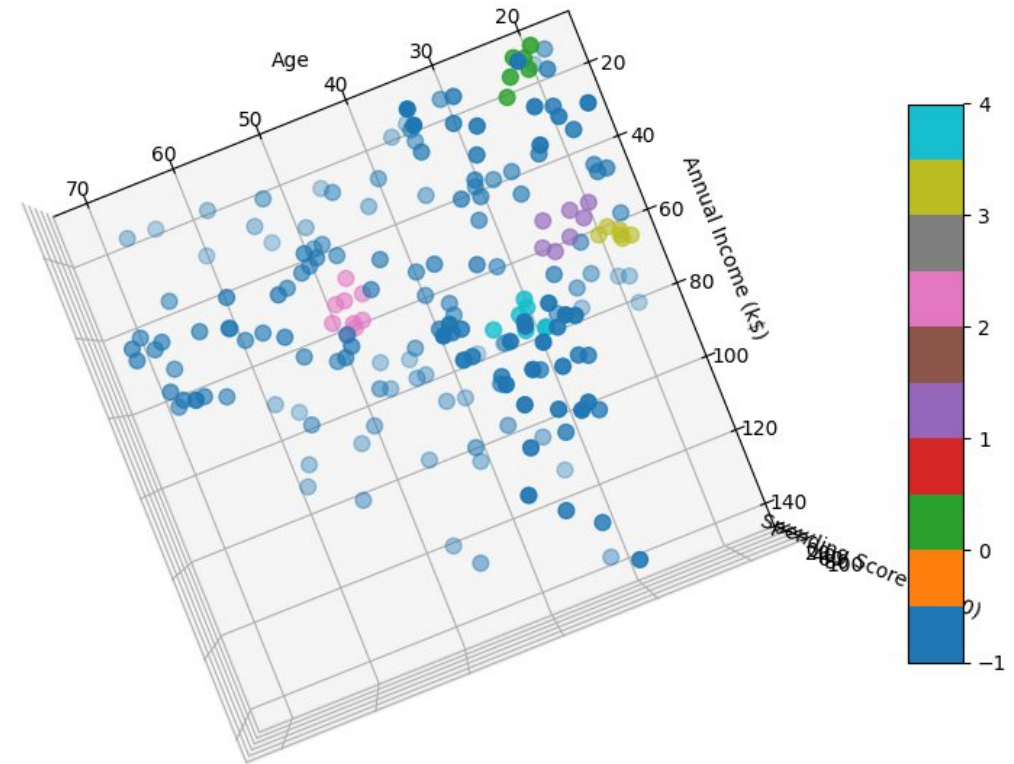
⇒ Ta chọn eps = 0.25

Bài 4: Bài toán Customer Segmentation (phân khúc khách hàng) “DBSCAN”

KMeans Clustering (3D)



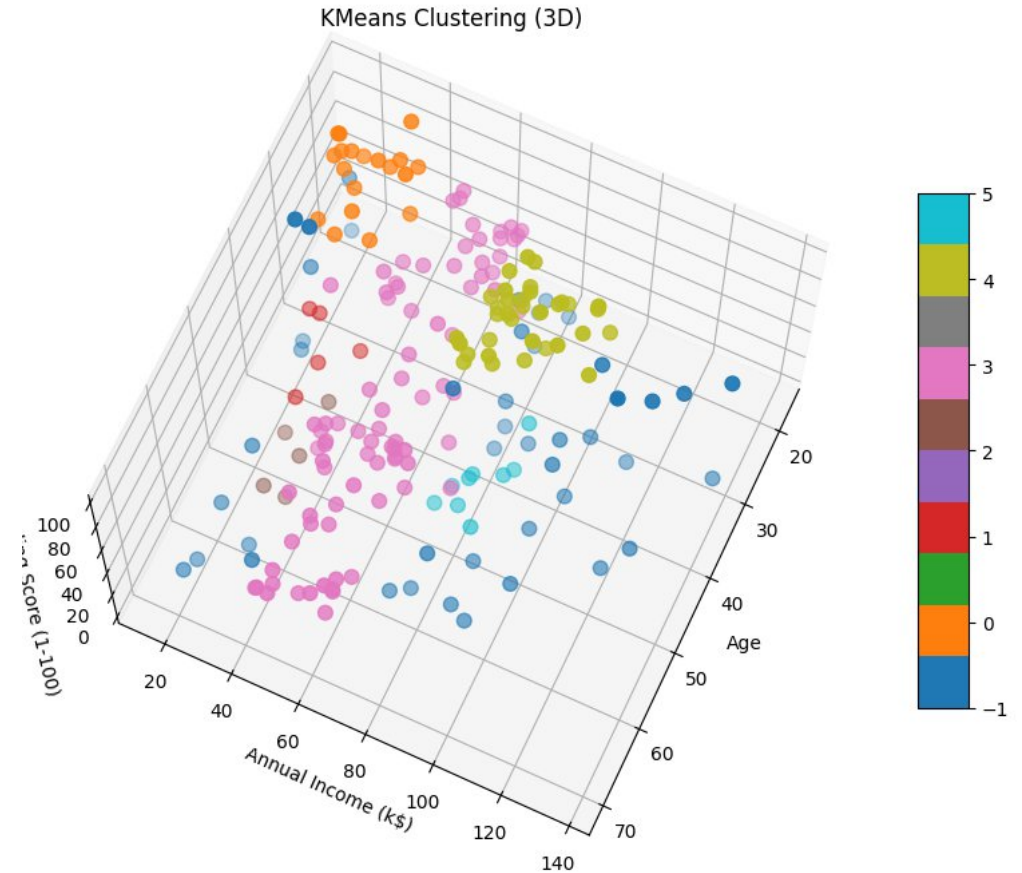
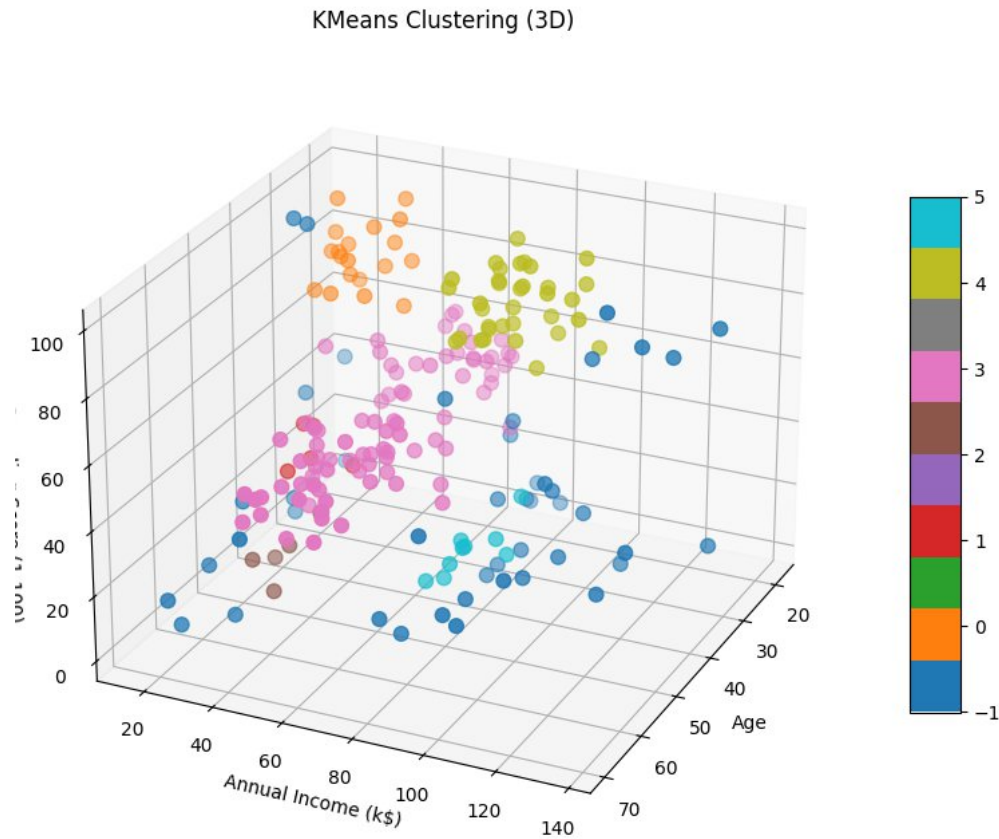
KMeans Clustering (3D)



➡ Với $\text{eps} = 0.25$ kết quả phân cụm không được tốt lắm

Bài 4: Bài toán Customer Segmentation (phân khúc khách hàng) “DBSCAN”

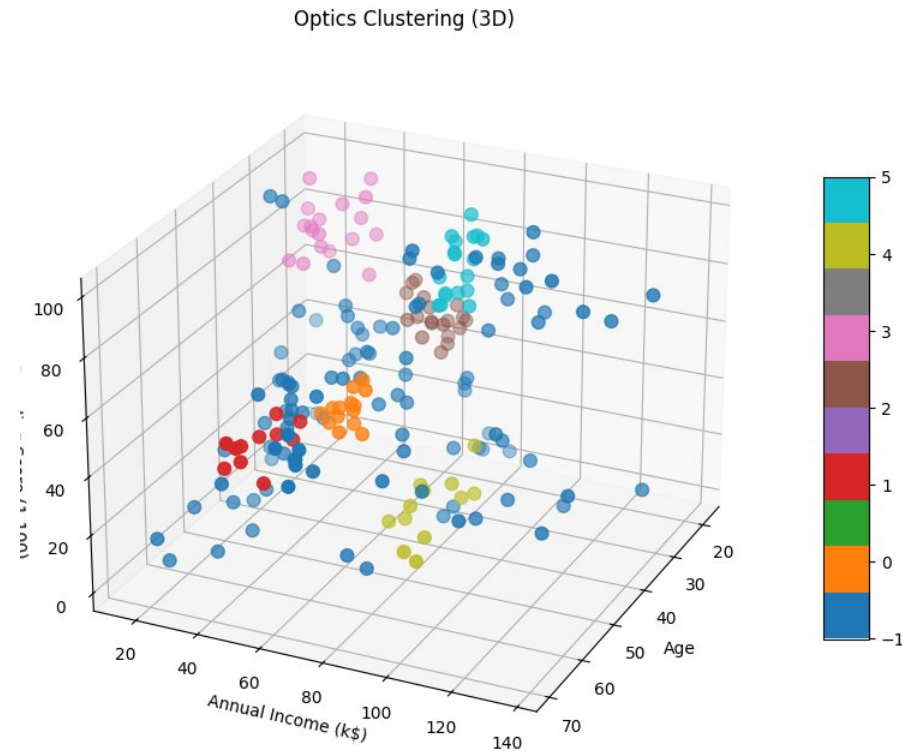
Chọn ngẫu nhiên $\text{eps} = 0.52$



➡ Với $\text{eps} = 0.52$ kết quả phân cụm tương đối ổn định hơn so với $\text{eps} = 0.25$. Nhưng vẫn kém so với Kmea

Bài 4: Bài toán Customer Segmentation (phân khúc khách hàng) “OPTICS”

Chọn ngẫu nhiên $\text{min_samples} = 5$, $\text{xi} = 0.05$, $\text{min_cluster_size} = 0.05$



➡ Các cụm được phân chia không được tốt lắm

Bài 5: Xây dựng mô hình Decision Tree và Random Forest cho bài toán Insurance Anti-Fraud Detection

- + Xóa các cột ít có khả năng ảnh hưởng đến kết quả
- + Xóa hàng 578 vì ngày xảy ra tai nạn trước ngày kí hợp đồng

```
▼ dropc_list = ['policy_number', 'policy_bind_date', 'policy_state', 'insured_zip', 'incident_location', 'incident_date',  
               'incident_state', 'incident_city', 'insured_hobbies', 'auto_make', 'auto_model', 'auto_year', '_c39',  
               'age', 'total_claim_amount']  
dropr_list = [578]  
✓ 0.0s
```

Python

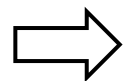
- + Tiến hành thay dấu '?' bằng 'NaN' -> thay thế các giá trị 'NaN' bằng giá trị xuất hiện nhiều trong cột
- + Tiến hành encoder các cột có kiểu dữ liệu phân loại
- + Dùng GridSearchCV và dò tay, tìm bộ thông số tốt nhất của 2 mô hình

```
param_grid = {  
    'criterion': ['gini', 'entropy'],  
    'max_depth': list(range(1, 10, 2)) + [None],  
    'min_samples_leaf': range(1, 10, 2),  
    'min_samples_split': range(1, 10, 2),  
}
```

✓ 0.0s

Bài 5: Xây dựng mô hình Decision Tree và Random Forest cho bài toán Insurance Anti-Fraud Detection

Mô hình	Thông số	F1-score
DecisionTreeClassifier	max_depth = 1 min_samples_leaf= 3, min_samples_split = 3 random_state = 42	Tập train: 0.64 Tập test: 0.63
RandomForestClassifier	criterion= 'entropy' min_samples_leaf= 1, min_samples_split = 10, n_estimators= 82, random_state = 42 class_weight = 'balanced_subsample'	Tập train: 0.91 Tập test: 0.62



- Decision Tree train F1 = 0.64, test F1 = 0.63
- Random Forest train F1 = 0.91, test F1 = 0.62 → overfit mạnh, test F1 thấp.

Bài 5: Xây dựng mô hình Decision Tree và Random Forest cho bài toán Insurance Anti-Fraud Detection

Mô hình	5 thuộc tính quan trọng
DecisionTreeClassifier	+ incident_severity_Major Damage + policy_deductable + months_as_customer + capital-gains + capital-loss
RandomForestClassifier	+ incident_severity_Major Damage + property_claim + injury_claim + vehicle_claim + incident_severity_Minor Damage

- ⇒
- + Hai mô hình chỉ có 1 feature chung trong top 5 cho thấy cơ chế đánh giá quan trọng khác nhau.
 - + Feature chung duy nhất là yếu tố quan trọng nhất, các feature còn lại khác nhau do cách học riêng của từng mô hình.

Bài 6: Credit Scoring in Banking: chấm điểm tín dụng ngân hàng

Mô hình	Thông số	Metrics
LogisticRegression	Mặc định	Recall score: 0.8 Precision score: 0.88 Accuracy score: 0.76 F1 score: 0.84
GaussianNB	Mặc định	Recall score: 0.83 Precision score: 0.74 Accuracy score: 0.71 F1 score: 0.78
RandomForestClassifier	n_estimators = 100	Recall score: 0.8 Precision score: 0.94 Accuracy score: 0.79 F1 score: 0.86
Support Vector Machine(không scaler)	kernel = 'rbf' C = 1, gamma = 'scale' class_weight = 'balanced'	Recall score: 0.78 Precision score: 0.67 Accuracy score: 0.63 F1 score: 0.72
KNeighborsClassifier(không scaler)	n_neighbors = 2	Recall score: 0.72 Precision score: 0.9 Accuracy score: 0.69 F1 score: 0.8

Bài 6: Credit Scoring in Banking: chấm điểm tín dụng ngân hàng

Mô hình	Thông số	Metrics
LogisticRegression	C=10 max_iter=1000 penalty='l1' solver='saga'	Recall score: 0.79 Precision score: 0.9 Accuracy score: 0.76 F1 score: 0.84
GaussianNB	var_smoothing=1e-05	Recall score: 0.83 Precision score: 0.79 Accuracy score: 0.73 F1 score: 0.81
RandomForestClassifier	n_estimators = 150 criterion='entropy' max_depth=10, min_samples_split=4	Recall score: 0.8 Precision score: 0.94 Accuracy score: 0.79 F1 score: 0.86
Support Vector Machine(có scaler)	kernel = 'rbf' C = 1 gamma = 'scale' class_weight = 'balanced'	Recall score: 0.7 Precision score: 1.0 Accuracy score: 0.7 F1 score: 0.83
KNeighborsClassifier	n_neighbors=14 p=1 weights='distance'	Recall score: 0.7 Precision score: 1.0 Accuracy score: 0.7 F1 score: 0.83

Bài 6: Credit Scoring in Banking: chấm điểm tín dụng ngân hàng

Mô hình	Thông số	Metrics
LogisticRegression(SMOTE)	C=10 max_iter=1000 penalty='l1' solver='saga'	Recall score: 0.8 Precision score: 0.91 Accuracy score: 0.77 F1 score: 0.85
GaussianNB(SMOTE)	var_smoothing=1e-05	Recall score: 0.81 Precision score: 0.82 Accuracy score: 0.74 F1 score: 0.82
RandomForestClassifier(SMOTE)	n_estimators = 150 criterion='entropy' max_depth=10, min_samples_split=4	Recall score: 0.79 Precision score: 0.95 Accuracy score: 0.79 F1 score: 0.86
Support Vector Machine(có scaler) (SMOTE)	kernel = 'rbf' C = 1 gamma = 'scale' class_weight = 'balanced'	Recall score: 0.7 Precision score: 1.0 Accuracy score: 0.7 F1 score: 0.83
KNeighborsClassifier(SMOTE)	n_neighbors=14 p=1 weights='distance'	Recall score: 0.7 Precision score: 1.0 Accuracy score: 0.7 F1 score: 0.83

Bài 6: Credit Scoring in Banking: chấm điểm tín dụng ngân hàng

Mô hình	Thông số	Metrics
LogisticRegression (TomekLinks)	C=10 max_iter=1000 penalty='l1' solver='saga'	Recall score: 0.79 Precision score: 0.91 Accuracy score: 0.77 F1 score: 0.84
GaussianNB(TomekLinks)	var_smoothing=1e-05	Recall score: 0.81 Precision score: 0.82 Accuracy score: 0.74 F1 score: 0.82
RandomForestClassifier(TomekLinks)	n_estimators = 150 criterion='entropy' max_depth=10, min_samples_split=4	Recall score: 0.79 Precision score: 0.95 Accuracy score: 0.79 F1 score: 0.86
Support Vector Machine(có scaler) (TomekLinks)	kernel = 'rbf' C = 1 gamma = 'scale' class_weight = 'balanced'	Recall score: 0.7 Precision score: 1.0 Accuracy score: 0.7 F1 score: 0.83
KNeighborsClassifier(TomekLinks)	n_neighbors=14 p=1 weights='distance'	Recall score: 0.7 Precision score: 1.0 Accuracy score: 0.7 F1 score: 0.83

Bài 7: Giảm chiều dữ liệu với PCA

Khi dùng PCA để trích xuất đặc trưng của dữ liệu n_components càng cao thì điểm f1 càng giảm

Mô hình	Thông số	F1 - score
RandomForestClassifier	Mặc định	0.268
XGBClassifier	Mặc định	0.1194
GradientBoostingClassifier	Mặc định	0.0967
GaussianNB	Mặc định	0.0
KNeighborsClassifier	Mặc định	0.166

Bảng điểm f1 khi n_components = 1

Bài 7: Giảm chiều dữ liệu với PCA

Áp dụng PCA với 2 mô hình được xây dựng trong bài 5

Mô hình	Thông số	F1-score
DecisionTreeClassifier	max_depth = 1 min_samples_leaf= 3, min_samples_split = 3 random_state = 42	Tập train: 0.0 Tập test: 0.0
RandomForestClassifier	criterion= 'entropy' min_samples_leaf= 1, min_samples_split = 10, n_estimators= 82, random_state = 42 class_weight = 'balanced_subsample'	Tập train: 0.77 Tập test: 0.31

⇒ Kết quả khi áp dụng PCA cho 2 mô hình trên với bộ dữ liệu bài 5, hiệu suất của mô hình giảm đáng kể

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ tiền xử lý dữ liệu:

*Chỉ xử lý scaler, encoder và xóa 2 cột 'id', 'attack_cat' cho 2 phương pháp feature selection và feature extracti

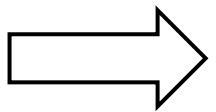
Xóa cột	'id', 'rate', 'spkts', 'dpkts', 'swin', 'sbytes', 'dloss', 'is_sm_ips_ports', 'ct_srv_dst', 'is_ftp_login', 'ct_srv_src', 'ct_src_dport_ltm', 'tcp_rtt', 'attack_cat'
Scaler	MinMaxScaler()
Encoder	OneHotEncoder()
Cân bằng dữ liệu	SMOTE

⇒ Sau bước này bộ dữ liệu có 188 cột và tương đối sạch để đưa vào mô hình.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ RandomForestClassifier

Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	Random_state = 42	Precision score: 0.84 Recall score: 0.98 Accuracy score: 0.89 F1 score: 0.91 AUC score: 0.98	73.716 giây
Chỉ feature extraction (PCA n = 4)	Random_state = 42	Precision score: 0.79 Recall score: 0.97 Accuracy score: 0.85 F1 score: 0.87 AUC score: 0.97	65.279 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	Random_state = 42	Precision score: 0.82 Recall score: 0.99 Accuracy score: 0.87 F1 score: 0.89 AUC score: 0.98	37.776 giây



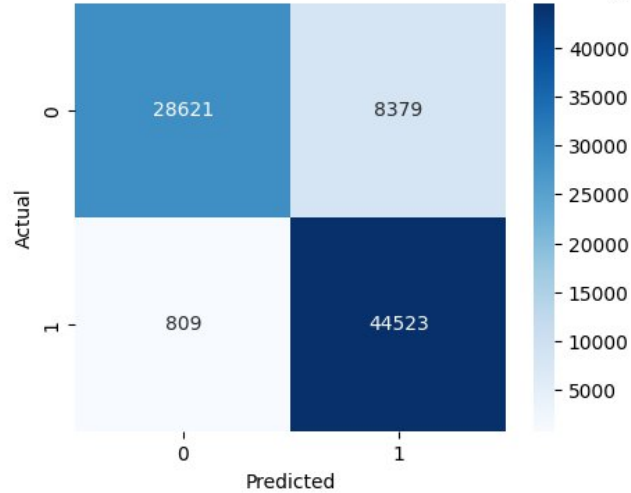
+ Chỉ preprocessing cho kết quả metrics cao nhất.

+ Thay vào đó feature selection lại có thời gian huấn luyện mô hình nhanh hơn và kết quả metric cũng rất t

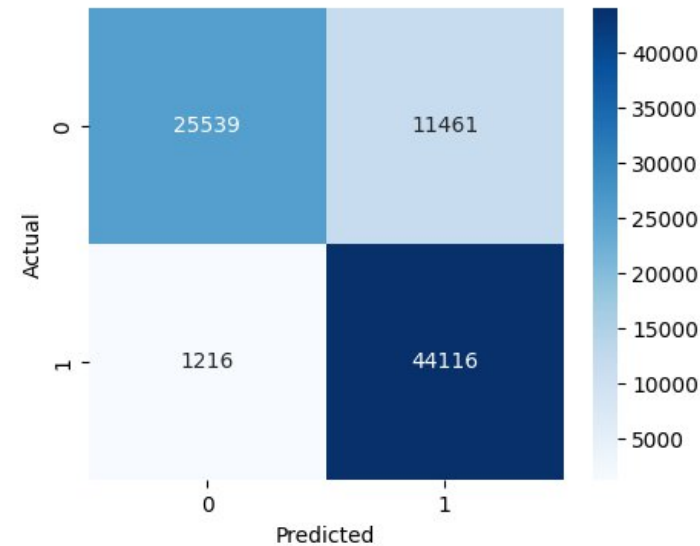
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ RandomForestClassifier

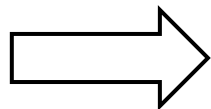
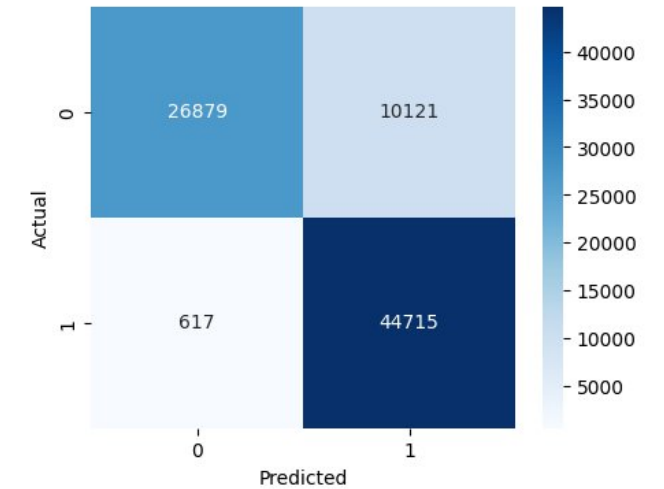
Confusion Matrix - RandomForestClassifier - Preprocessing



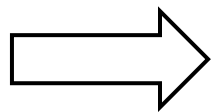
Confusion Matrix - RandomForestClassifier - PCA



Confusion Matrix - RandomForestClassifier - VarianceThreshold



Phương pháp feature select có ít trường hợp mắc sai lầm loại 2 nhất.



+ Trong mô hình RandomForest áp dụng lên bộ dữ liệu UNSW-NB15 phương pháp feature selection có hiệu suất tốt nhất

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ GradientBoostingClassifier

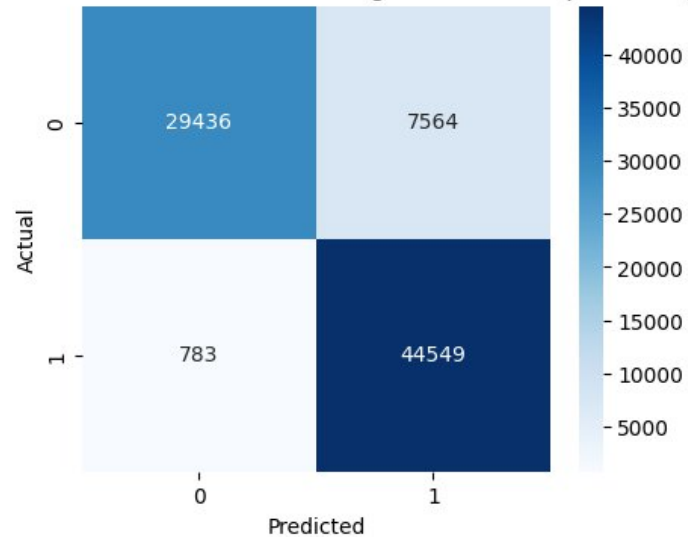
Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	Random_state = 42	Precision score: 0.85 Recall score: 0.98 Accuracy score: 0.90 F1 score: 0.91 AUC score: 0.99	291.387 giây
Chỉ feature extraction (PCA n = 4)	Random_state = 42	Precision score: 0.74 Recall score: 1.00 Accuracy score: 0.81 F1 score: 0.85 AUC score: 0.96	59.791 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	Random_state = 42	Precision score: 0.79 Recall score: 0.98 Accuracy score: 0.85 F1 score: 0.88 AUC score: 0.98	84.345 giây

+ chỉ preprocessing có hiệu suất cao nhất nhưng quá trình huấn luyện mô hình mất rất nhiều thời gian.

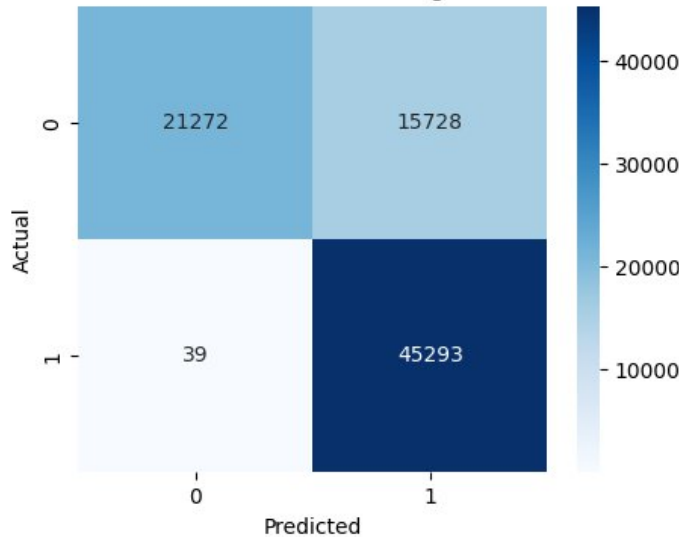
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ GradientBoostingClassifier

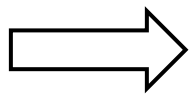
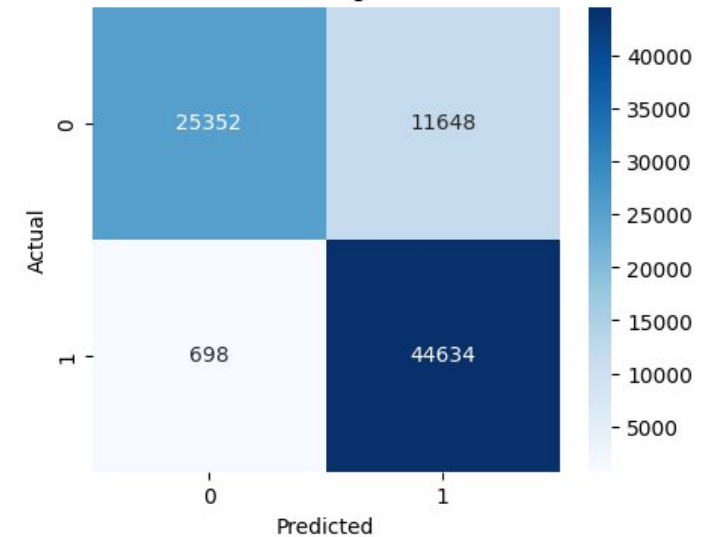
Confusion Matrix - GradientBoostingClassifier - Preprocessing



Confusion Matrix - GradientBoostingClassifier - PCA



Confusion Matrix - GradientBoostingClassifier - VarianceThreshold

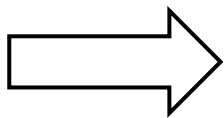


- + phương pháp feature extraction có ít trường hợp mắc sai lầm loại 2 nhưng mắc rất nhiều sai lầm loại 1.
- + mặc dù có thời gian huấn luyện lâu nhưng phương pháp preprocessing có độ tin cậy cao hơn.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ AdaBoostClassifier

Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	Random_state = 42	Precision score: 0.82 Recall score: 0.98 Accuracy score: 0.87 F1 score: 0.89 AUC score: 0.97	77.434 giây
Chỉ feature extraction (PCA n = 4)	Random_state = 42	Precision score: 0.73 Recall score: 0.99 Accuracy score: 0.79 F1 score: 0.84 AUC score: 0.93	13.159 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	Random_state = 42	Precision score: 0.77 Recall score: 0.97 Accuracy score: 0.82 F1 score: 0.86 AUC score: 0.95	21.038 giây

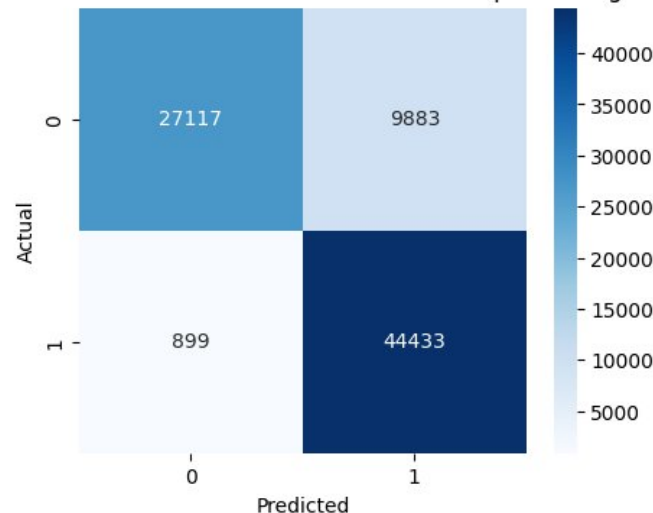


+ phương pháp feature extraction cho thời gian huấn luyện nhanh nhất và recall score tốt nhất nhưng các metrics khác khá thấp khi so với các phương pháp khác.

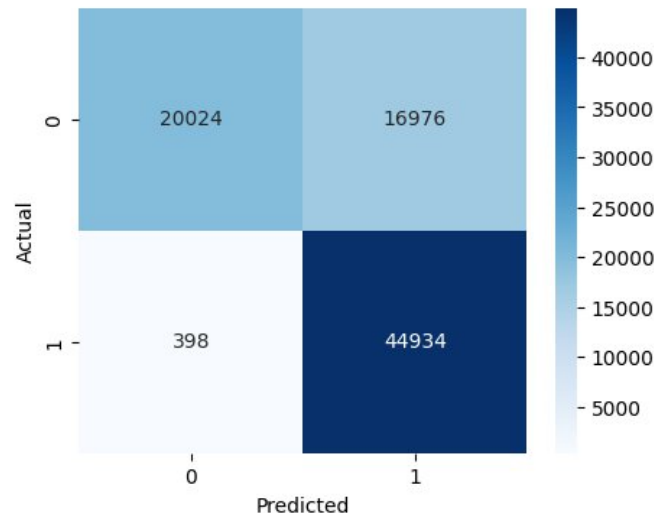
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ AdaBoostClassifier

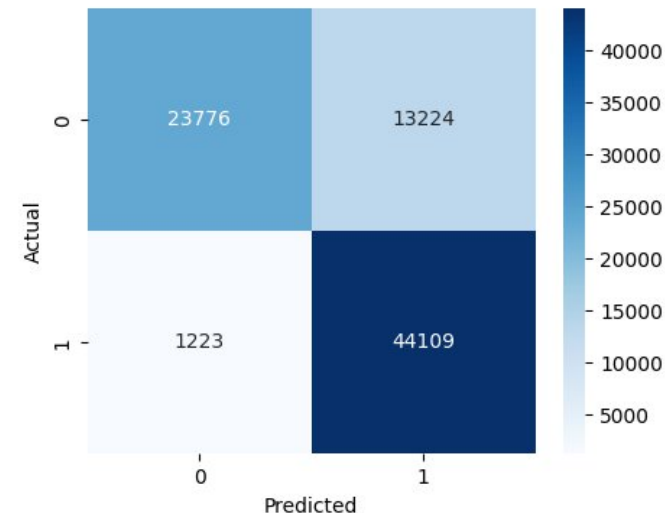
Confusion Matrix - AdaBoostClassifier - Preprocessing



Confusion Matrix - AdaBoostClassifier - PCA



Confusion Matrix - AdaBoostClassifier - VarianceThreshold



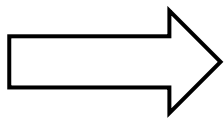
+ phương pháp feature extraction có ít trường hợp mắc sai lầm loại 2 nhưng mắc rất nhiều sai lầm loại 1.

+ mặc dù có thời gian huấn luyện lâu nhưng phương pháp preprocessing có độ tin cậy cao hơn.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ LogisticRegression

Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	Random_state = 42	Precision score: 0.80 Recall score: 0.96 Accuracy score: 0.85 F1 score: 0.87 AUC score: 0.96	15.057 giây
Chỉ feature extraction (PCA n = 4)	Random_state = 42	Precision score: 0.68 Recall score: 0.89 Accuracy score: 0.71 F1 score: 0.77 AUC score: 0.85	1.683 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	Random_state = 42	Precision score: 0.75 Recall score: 0.98 Accuracy score: 0.81 F1 score: 0.85 AUC score: 0.95	3.036 giây

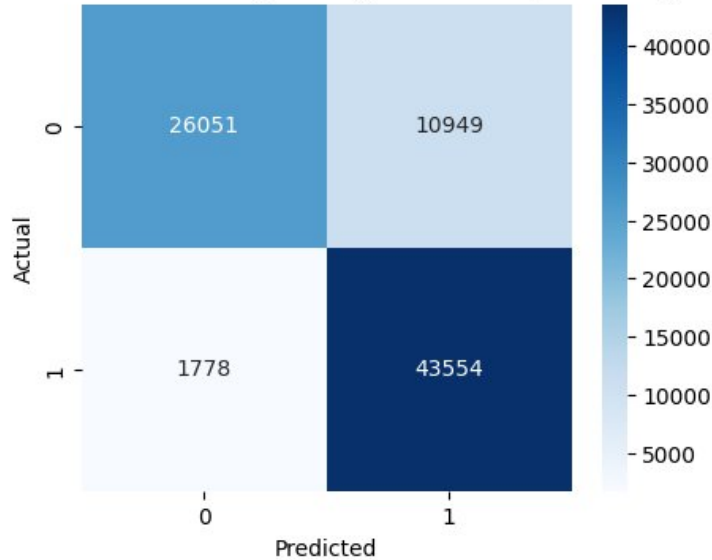


+ phương pháp feature extraction cho thời gian huấn luyện nhanh nhất nhưng các chỉ số metrics khá thấp khi so với các phương pháp khác.

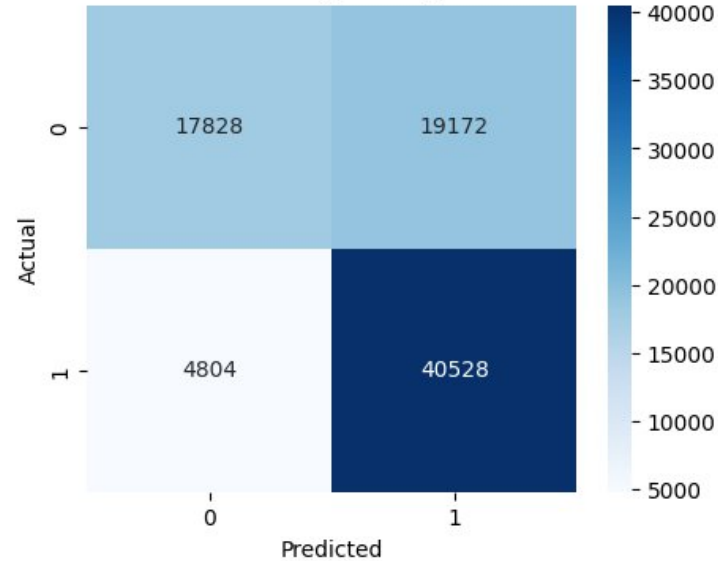
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ LogisticRegression

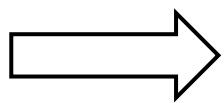
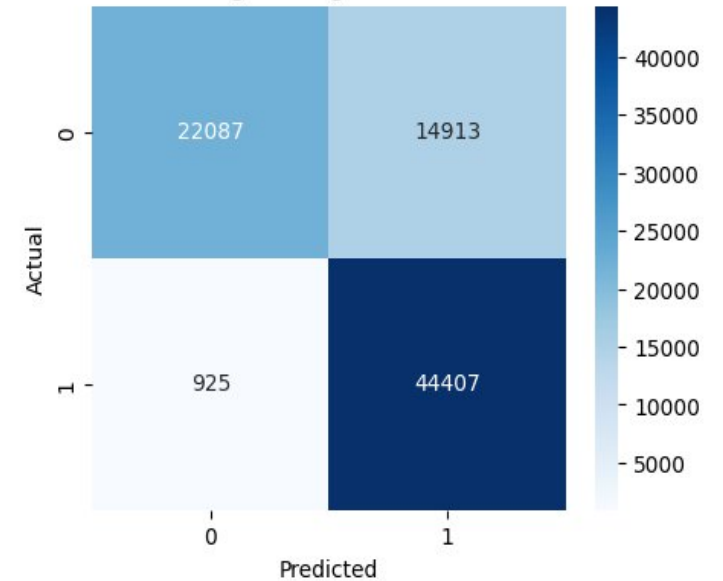
Confusion Matrix - LogisticRegression - Preprocessing



Confusion Matrix - LogisticRegression - PCA



Confusion Matrix - LogisticRegression - VarianceThreshold

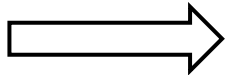


+ mặc dù preprocessing có thông số metrics tốt nhất nhưng mô hình khi áp dụng phương pháp feature selection lại đáng tin hơn.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ KNeighborsClassifier

Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	n_neighbors = 2	Precision score: 0.87 Recall score: 0.92 Accuracy score: 0.88 F1 score: 0.90 AUC score: 0.89	103.087 giây
Chỉ feature extraction (PCA n = 4)	n_neighbors = 2	Precision score: 0.86 Recall score: 0.92 Accuracy score: 0.87 F1 score: 0.89 AUC score: 0.88	2.280 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	n_neighbors = 2	Precision score: 0.85 Recall score: 0.92 Accuracy score: 0.87 F1 score: 0.89 AUC score: 0.88	26.469 giây



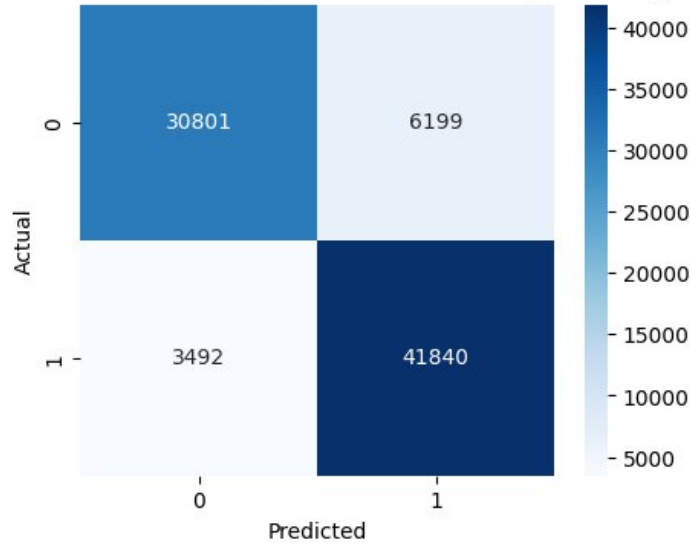
+ khi áp dụng phương pháp preprocessing mô hình huấn luyện với thời gian rất lâu nhưng hiệu suất không quá khác biệt với các phương pháp khác.

+ phương pháp feature extraction có thời gian huấn luyện ít và có hiệu suất cao nhất.

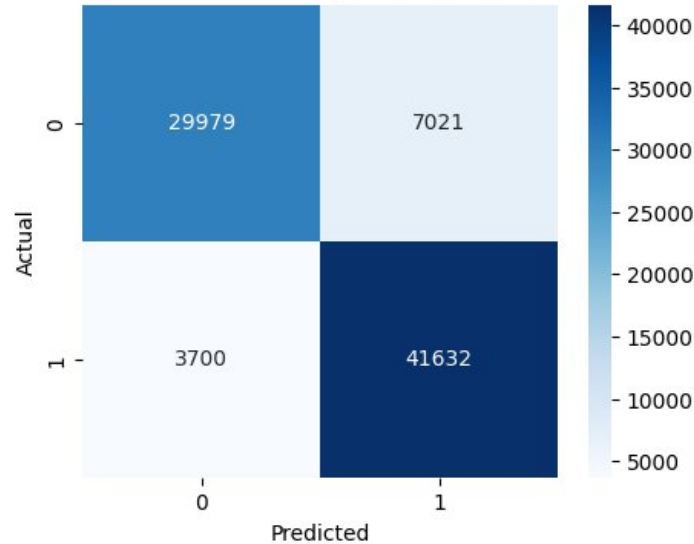
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ KNeighborsClassifier

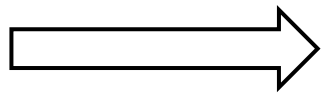
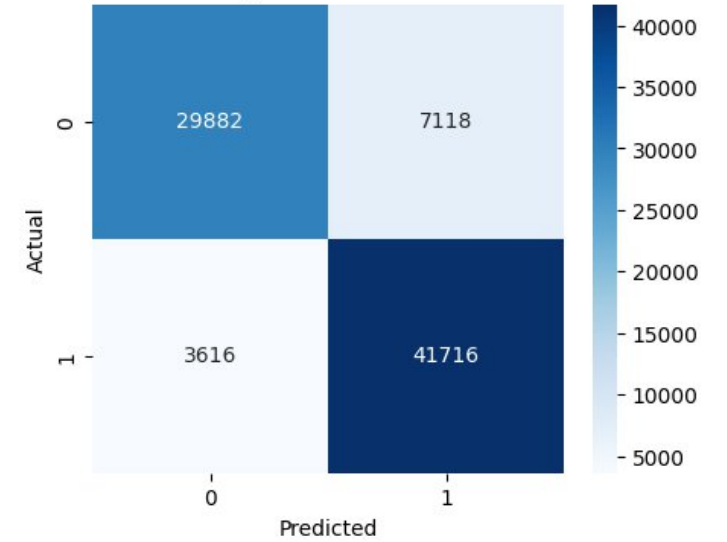
Confusion Matrix - KNeighborsClassifier - Preprocessing



Confusion Matrix - KNeighborsClassifier - PCA



Confusion Matrix - KNeighborsClassifier - VarianceThreshold



+ Phương pháp feature extraction có độ tin cậy cao nhất.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ GaussianNB

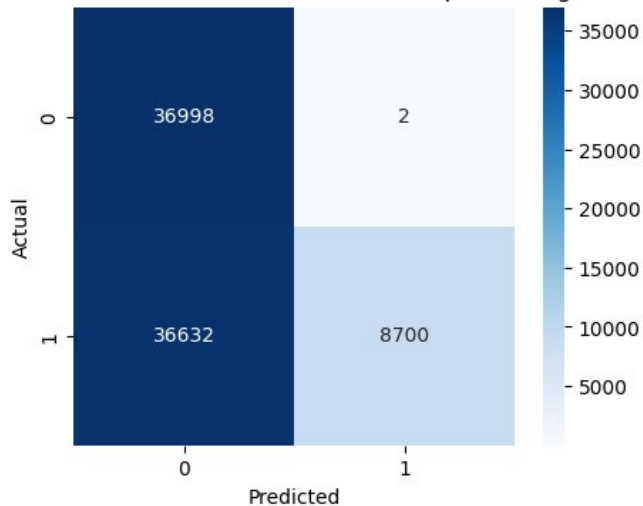
Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing		Precision score: 1.00 Recall score: 0.19 Accuracy score: 0.56 F1 score: 0.32 AUC score: 0.62	12.001 giây
Chỉ feature extraction (PCA n = 4)		Precision score: 0.70 Recall score: 0.92 Accuracy score: 0.74 F1 score: 0.80 AUC score: 0.86	1.403 giây
Chỉ feature selection(VarianceThreshold n = 0.011)		Precision score: 0.73 Recall score: 0.86 Accuracy score: 0.75 F1 score: 0.79 AUC score: 0.85	2.478 giây

➡ + các phương pháp xử lý dữ liệu khi áp dụng mô hình GaussianNB với bộ dữ liệu UNSW-NB15 có hiệu không t
Khi so với các mô hình khác.

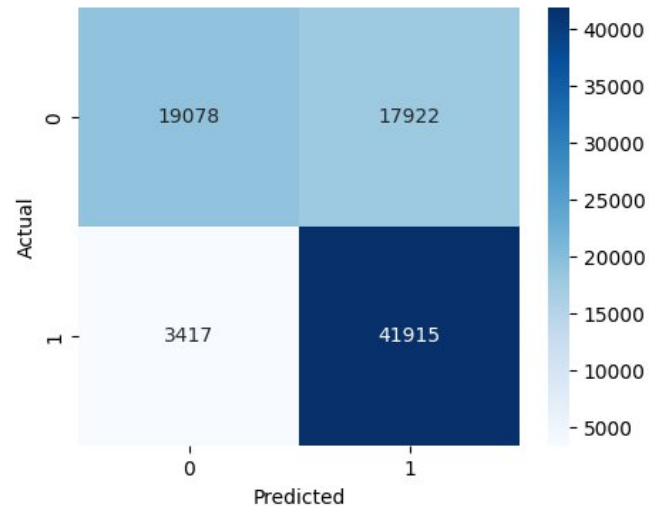
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ GaussianNB

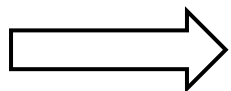
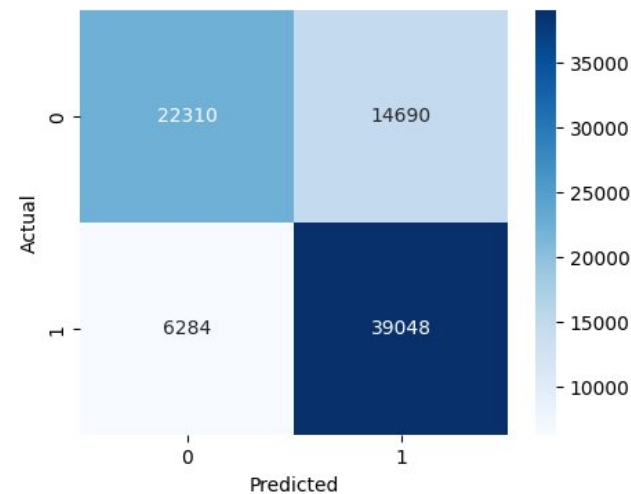
Confusion Matrix - GaussianNB - Preprocessing



Confusion Matrix - GaussianNB - PCA



Confusion Matrix - GaussianNB - VarianceThreshold



Phương pháp preprocessing có tỉ lệ mắc sai lầm loại 2 gần như tuyệt đối.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ DecisionTreeClassifier

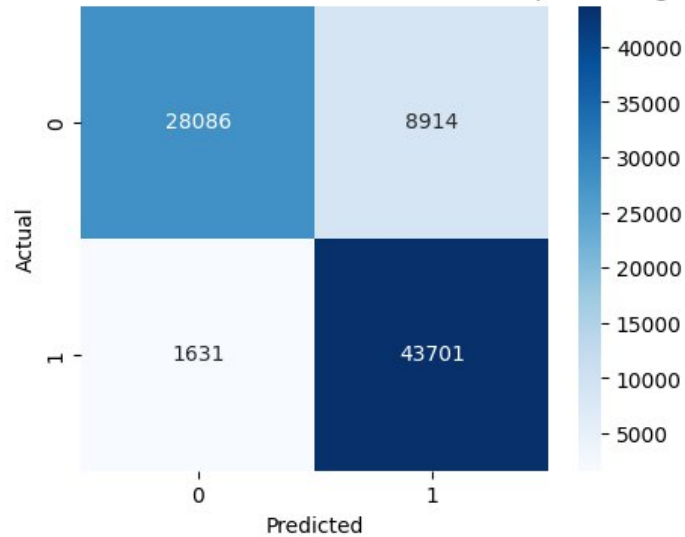
Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	Random_state = 42	Precision score: 0.83 Recall score: 0.96 Accuracy score: 0.87 F1 score: 0.89 AUC score: 0.87	17.099 giây
Chỉ feature extraction (PCA n = 4)	Random_state = 42	Precision score: 0.81 Recall score: 0.95 Accuracy score: 0.85 F1 score: 0.87 AUC score: 0.84	3.409 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	Random_state = 42	Precision score: 0.82 Recall score: 0.96 Accuracy score: 0.86 F1 score: 0.88 AUC score: 0.85	4.173 giây

⇒ + phương pháp preprocessing cho thời gian huấn luyện rất lâu nhưng hiệu suất không quá tốt so với feature selection

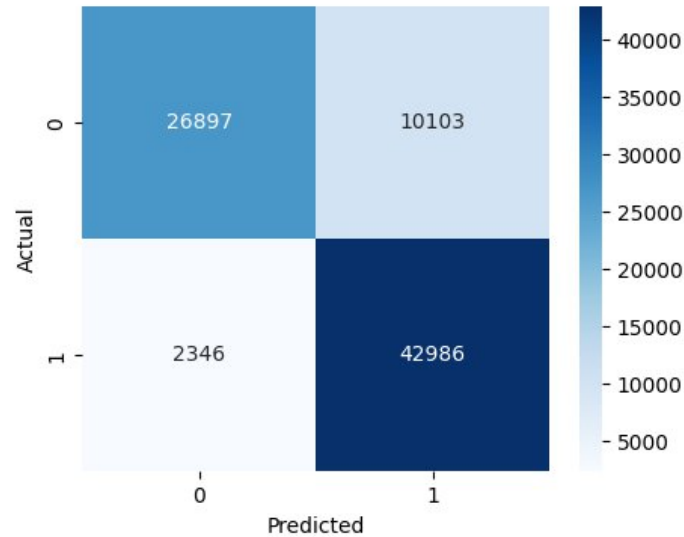
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ DecisionTreeClassifier

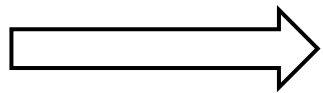
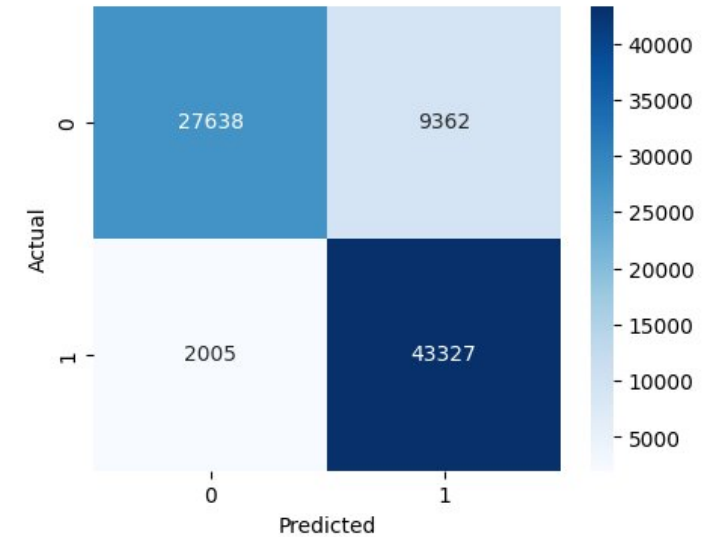
Confusion Matrix - DecisionTreeClassifier - Preprocessing



Confusion Matrix - DecisionTreeClassifier - PCA



Confusion Matrix - DecisionTreeClassifier - VarianceThreshold



+ Mô hình khi áp dụng phương pháp preprocessing có độ tin cậy cao nhất.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ XGBClassifier

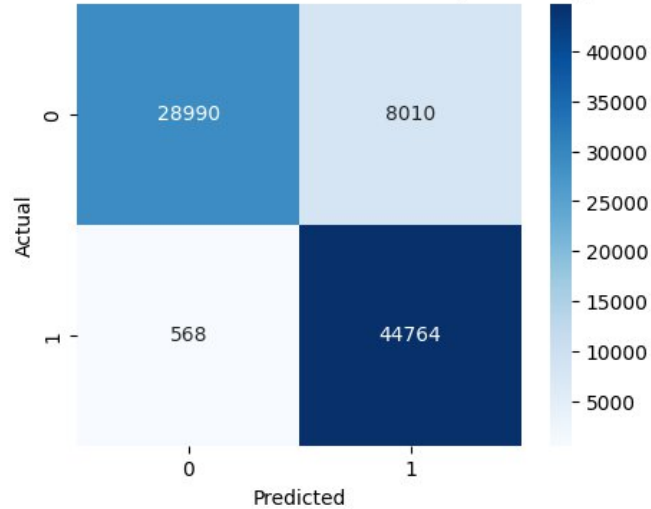
Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing		Precision score: 0.85 Recall score: 0.99 Accuracy score: 0.90 F1 score: 0.91 AUC score: 0.99	13.490 giây
Chỉ feature extraction (PCA n = 4)		Precision score: 0.77 Recall score: 0.99 Accuracy score: 0.83 F1 score: 0.86 AUC score: 0.97	1.454 giây
Chỉ feature selection(VarianceThreshold n = 0.011)		Precision score: 0.82 Recall score: 0.98 Accuracy score: 0.87 F1 score: 0.89 AUC score: 0.98	2.122 giây

⇒ + phương pháp preprocessing cho thời gian huấn luyện rất lâu nhưng hiệu suất không quá tốt so với feature selection

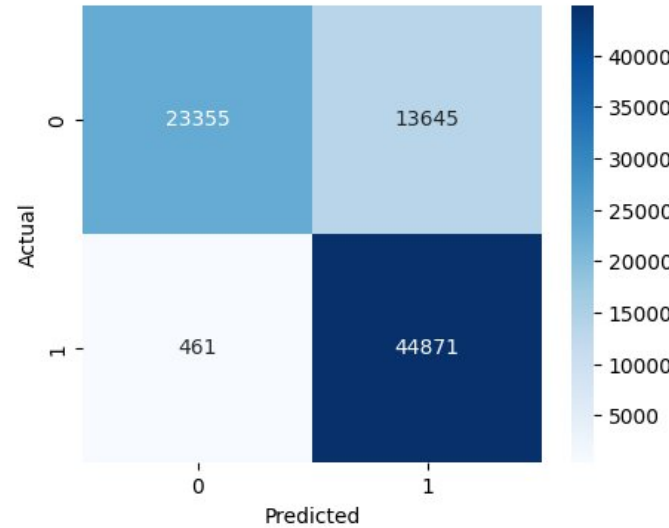
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ XGBClassifier

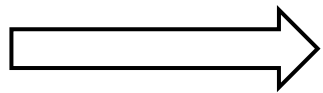
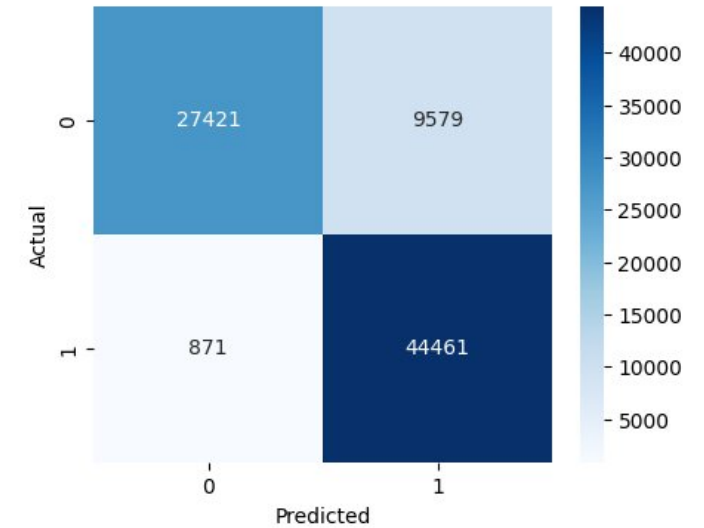
Confusion Matrix - XGBClassifier - Preprocessing



Confusion Matrix - XGBClassifier - PCA



Confusion Matrix - XGBClassifier - VarianceThreshold



+ Mô hình khi áp dụng phương pháp preprocessing có độ tin cậy cao nhất.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

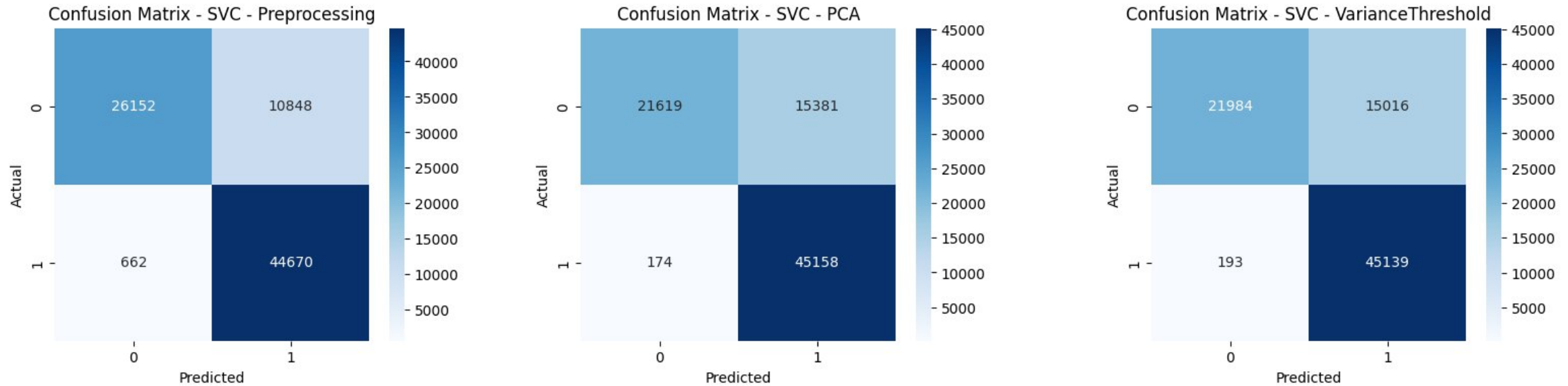
+ Support vector machine

Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	Random_state = 42	Precision score: 0.80 Recall score: 0.99 Accuracy score: 0.86 F1 score: 0.89	3678.511 giây
Chỉ feature extraction (PCA n = 4)	Random_state = 42	Precision score: 0.75 Recall score: 1.00 Accuracy score: 0.81 F1 score: 0.85	920.243 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	Random_state = 42	Precision score: 0.75 Recall score: 1.00 Accuracy score: 0.82 F1 score: 0.86	725.765 giây

+ Preprocessing cho hiệu suất tốt nhất nhưng thời gian huấn luyện rất lâu.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ Support vector machine



+ các phương pháp đều cho ra kết quả có hiệu quả không khác nhau lắm, nhưng nếu có thể chấp nhận nhiều cảnh báo giả, thì mô hình với phương pháp feature selection là tốt nhất.

Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ LightGBM

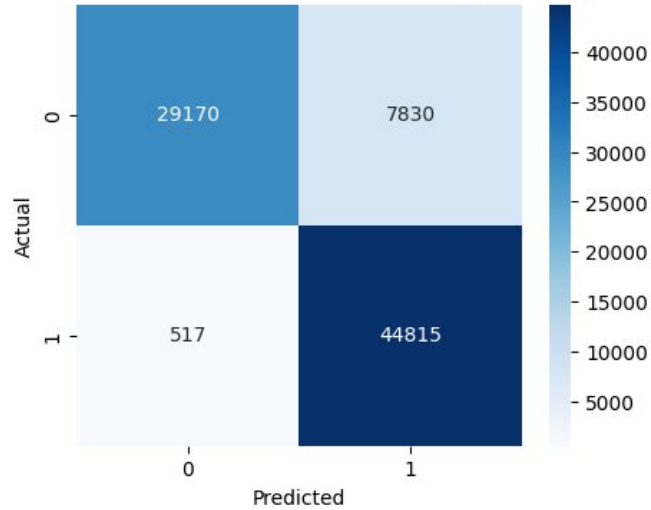
Phương pháp	Thông số mô hình	Metrics	Thời gian chạy chương trình
Preprocessing	Random_state = 42	Precision score: 0.85 Recall score: 0.99 Accuracy score: 0.90 F1 score: 0.91 AUC score: 0.99	11.377 giây
Chỉ feature extraction (PCA n = 4)	Random_state = 42	Precision score: 0.75 Recall score: 1.00 Accuracy score: 0.82 F1 score: 0.86 AUC score: 0.97	1.665 giây
Chỉ feature selection(VarianceThreshold n = 0.011)	Random_state = 42	Precision score: 0.82 Recall score: 0.99 Accuracy score: 0.88 F1 score: 0.90 AUC score: 0.99	2.164 giây

+ phương pháp feature cho hiệu suất tương đối tốt với thời gian huấn luyện mô hình nhanh nhất.

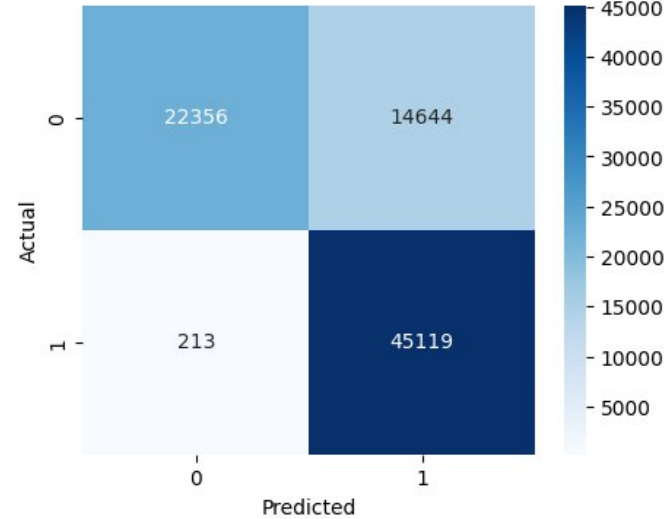
Bài 8: Network Intrusion Detection Systems - tổng hợp các kiến thức đã học

+ LightGBM

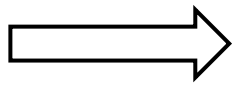
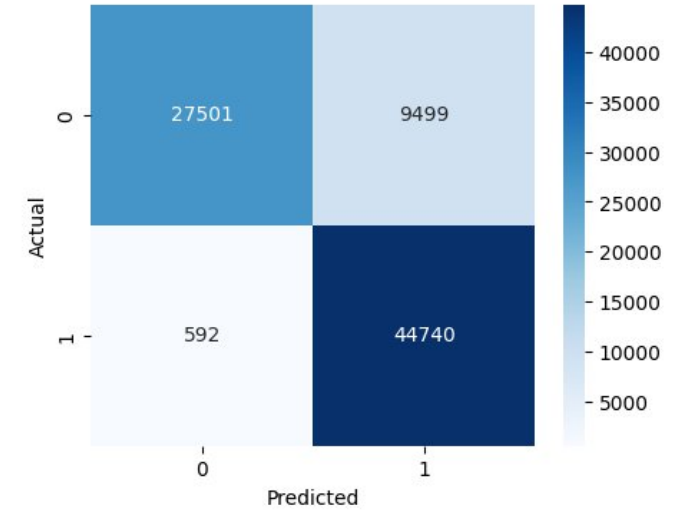
Confusion Matrix - LGBMClassifier - Preprocessing



Confusion Matrix - LGBMClassifier - PCA



Confusion Matrix - LGBMClassifier - VarianceThreshold



+ nếu có thể chấp nhận thời gian huấn luyện lâu thì phương pháp preprocessing cho ra kết quả có hiệu suất cao nhất.