6501 Progress Report Group 4

Improving Credit Card Fraud Detection using Generative Adversarial Networks

Group 4 Team Member: Hao Ning, Jun Ying

Working Schedule

Time	Milestone
09/21/2020	Exploratory Data Analysis (EDA): Jun Base Model: Hao
09/28/2020	Original data + GAN
10/05/2020	Network & Framework Development WGAN: Hao BEGAN: Jun
10/12/2020	WGAN & BEGAN Evaluation & Analysis
10/19/2020	Network & Framework Development BAGAN: Hao SNGAN: Jun
10/26/2020	Preliminary Presentation
11/02/2020	BAGAN, SNGAN Evaluation & Analysis
11/09/2020 & 11/16/2020	Summary of Results
11/23/2020	Manuscript
11/30/2020 & 12/07/2020	Mock Presentation & Presentation and Journal Submission

6501 Progress Report Group 4

09/21/2020

EDA

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
```

About the dataset, there are 30 features and 1 class (normal:0, fraud:1)

	Time	V1	V2	V3	...	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	...	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	...	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	...	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	...	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	...	0.219422	0.215153	69.99	0

There is no null value in the dataset.

Total null values in the dataset
0

As we know, the dataset is extremely imbalanced(0.173%).

The amounts of normal transactions (class 0) & fraud transactions (class 1)
0 284315
1 492

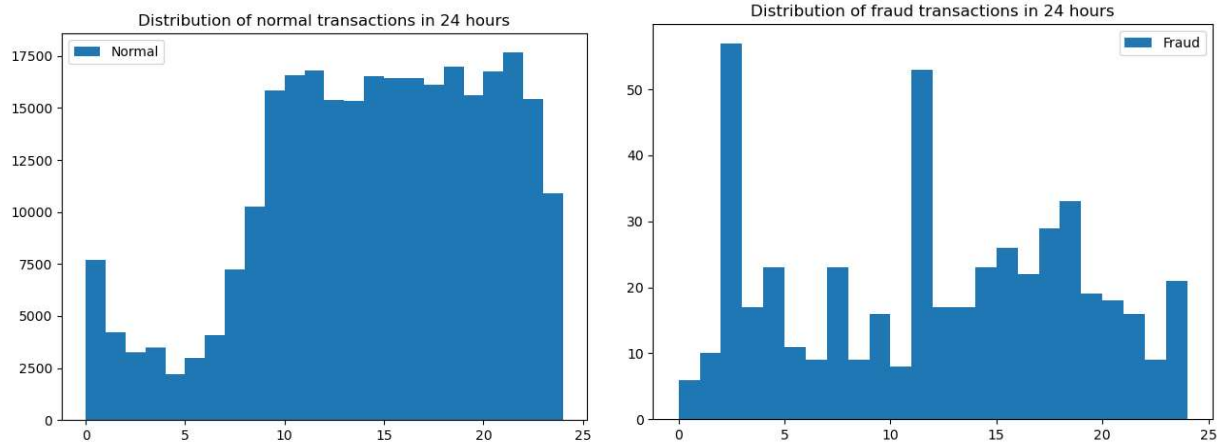
We have observed that there are some transactions which are 0.

	Time	V1	...	Amount	Class
count	284807.000000	2.848070e+05	...	284807.000000	284807.000000
mean	14.537951	3.919560e-15	...	88.349619	0.001727
std	5.847061	1.958696e+00	...	250.120109	0.041527
min	0.000000	-5.640751e+01	...	0.000000	0.000000
25%	10.598194	-9.203734e-01	...	5.600000	0.000000
50%	15.010833	1.810880e-02	...	22.000000	0.000000
75%	19.329722	1.315642e+00	...	77.165000	0.000000
max	23.999444	2.454930e+00	...	25691.160000	1.000000

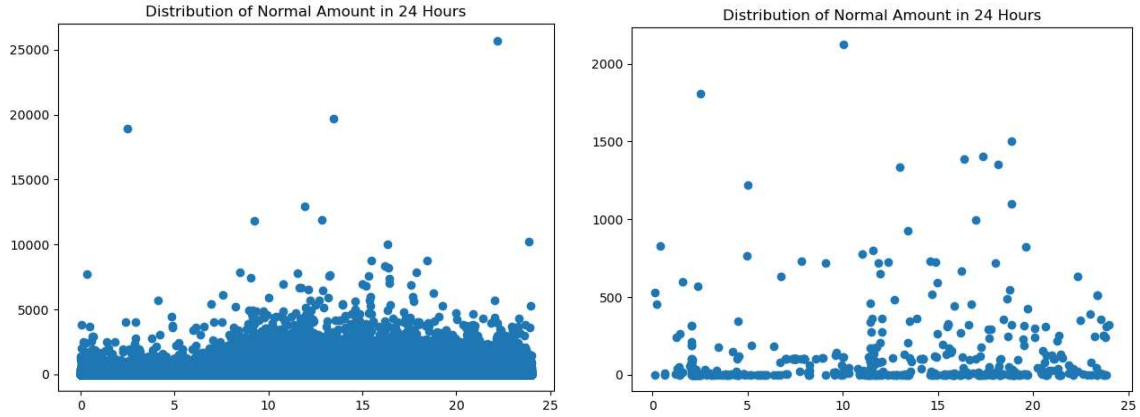
The total number of 0 amount: 1825 (1.479% fraud)

```
The null amounts of normal transactions (class 0) & fraud transactions (class 1)
0   1798
1     27
Name: Class, dtype: int64
```

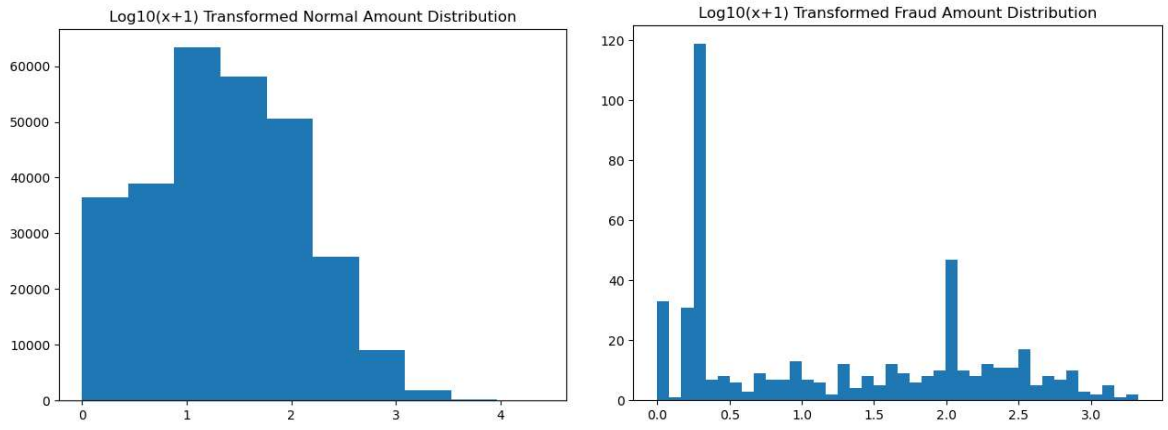
6501 Progress Report Group 4



From the histogram, we can observe that normal transactions generally occur from 9 am to 0 am. However, the fraud transactions occur particularly frequently at 2 am and 12 pm.

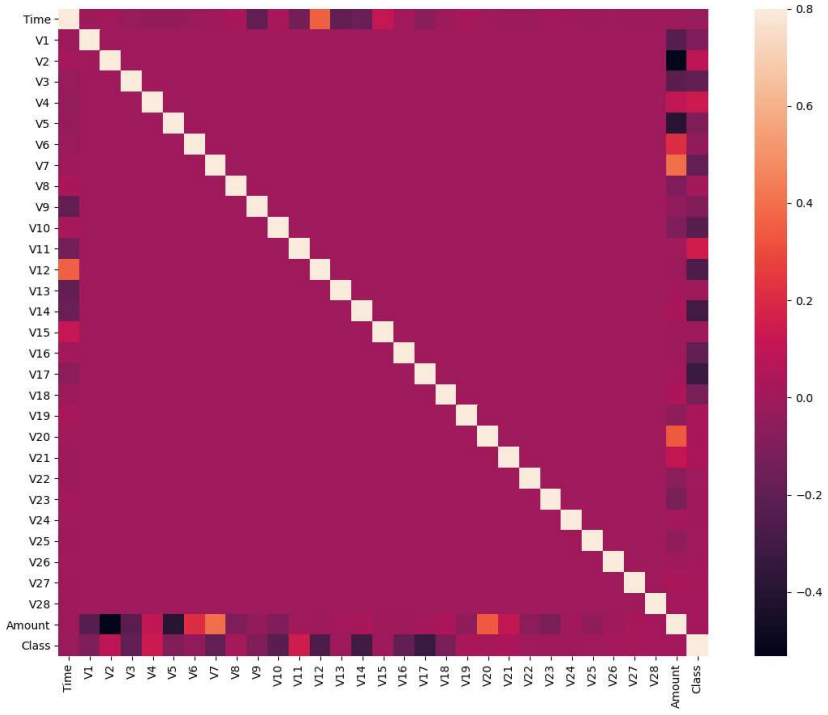


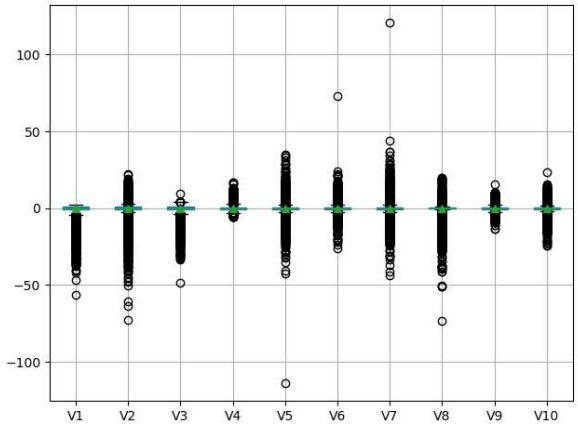
We can find from this scatter plot that the amount of super large transactions is very small. In comparison, the largest amount of nurmal transactions is over €25,000. However, the largest amount of fraud transactions is only €2,000.

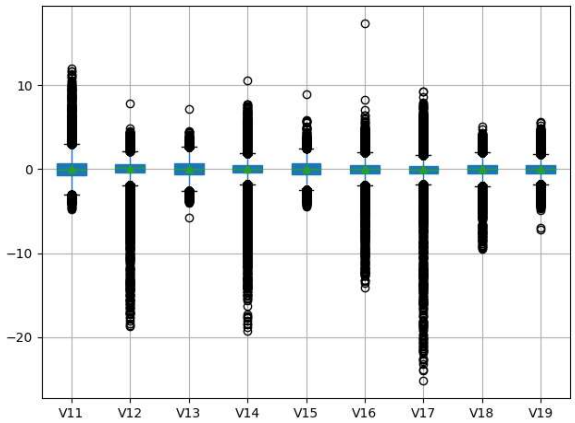


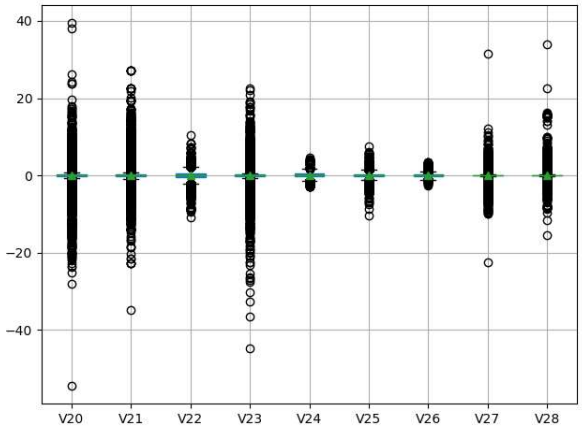
Normal amount was from ten to hundred. Fraud Amount distributed in less than €1.

6501 Progress Report Group 4







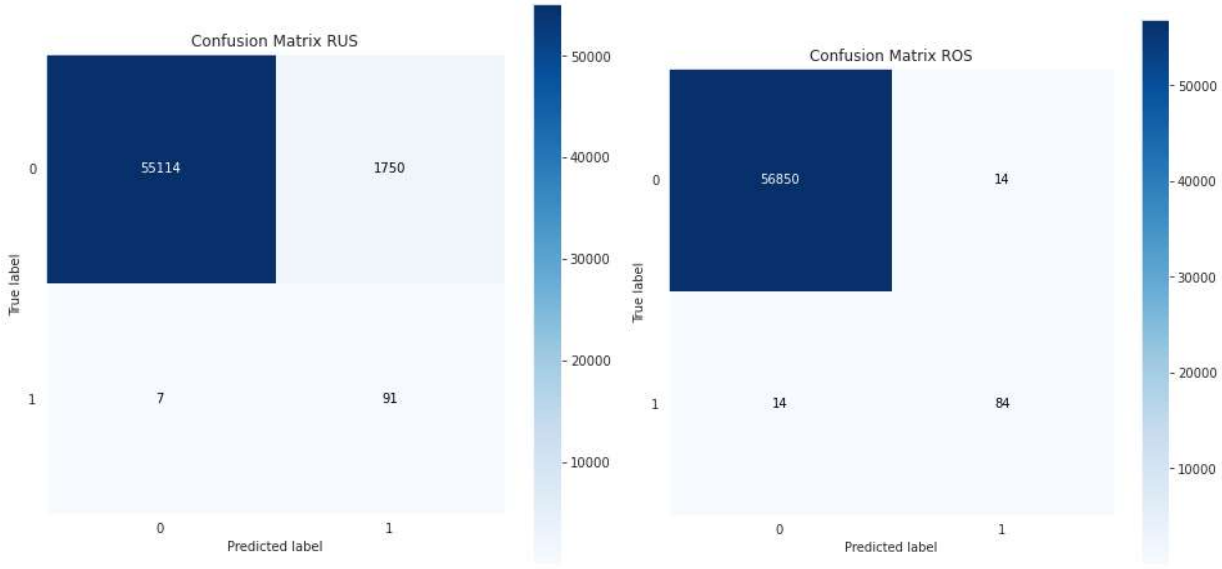


6501 Progress Report Group 4

Base Model:

1. Train Test Split & Stratified: 80% 227845 (**394 fraud**), 20% 56962 (**98 fraud**)
2. Random Under Sampling (RUS) and Random Over Sampling (ROS)
3. GridsearchCV for XGBoostClassifier
4. Predict with best_params
5. Test result comparison

RUS 1 394 0 394	ROS 1 227451 0 227451
Accuracy: 0.9691548751799445 Precision: 0.049429657794676805 Recall: 0.9285714285714286 F1 score: 0.09386281588447654 ROC AUC score: 0.9488981228394566	Accuracy: 0.9995084442259752 Precision: 0.8571428571428571 Recall: 0.8571428571428571 F1 score: 0.8571428571428571 ROC AUC score: 0.9284483278398585



6501 Progress Report Group 4

09/28/2020

Implemented with Keras

A bit change from from original gan:
Vanilla gan deals with images, but we are dealing with tabular data, so tanh is removed.
Also, we removed batch normalization since the training results are bad.

Noise = 32
len(features) = 30

Generator

Input	Output size
(32, 1)	(64,1)
(64, 1)	(128, 1)
(128, 1)	(256, 1)
(256, 1)	(30, 1)

LeakyReLU(0.2)

Discriminator

Input	Output size
(30, 1)	(256,1)
(256, 1)	(128, 1)
(128, 1)	(64, 1)
(64, 1)	(1, 1)

LeakyReLU(0.2)

Train
epoch=10, batch_size=128, steps_per_epoch=100

```
EPOCH #  10 --------------------------------------------------
Steps (5 / 50): [Loss_D_real: 0.101874, Loss_D_fake: 0.804704, acc: 55.47%] [Loss_G: 0.577402]
Steps (10 / 50): [Loss_D_real: 0.034651, Loss_D_fake: 0.794337, acc: 58.59%] [Loss_G: 0.583686]
Steps (15 / 50): [Loss_D_real: 0.043936, Loss_D_fake: 0.802042, acc: 64.84%] [Loss_G: 0.606348]
Steps (20 / 50): [Loss_D_real: 0.034166, Loss_D_fake: 0.831742, acc: 57.81%] [Loss_G: 0.573465]
Steps (25 / 50): [Loss_D_real: 0.045282, Loss_D_fake: 0.852394, acc: 56.25%] [Loss_G: 0.597851]
Steps (30 / 50): [Loss_D_real: 0.072814, Loss_D_fake: 0.801378, acc: 60.94%] [Loss_G: 0.593632]
Steps (35 / 50): [Loss_D_real: 0.062091, Loss_D_fake: 0.837749, acc: 55.47%] [Loss_G: 0.597265]
Steps (40 / 50): [Loss_D_real: 0.099103, Loss_D_fake: 0.853835, acc: 54.69%] [Loss_G: 0.578026]
Steps (45 / 50): [Loss_D_real: 0.088608, Loss_D_fake: 0.799784, acc: 60.16%] [Loss_G: 0.573147]
Steps (50 / 50): [Loss_D_real: 0.043118, Loss_D_fake: 0.808880, acc: 61.72%] [Loss_G: 0.589081]
```

epoch=20, batch_size=128, steps_per_epoch=100

```
EPOCH #  20 --------------------------------------------------
Steps (10 / 100): [Loss_D_real: 0.041530, Loss_D_fake: 0.459832, acc: 100.00%] [Loss_G: 1.017299]
Steps (20 / 100): [Loss_D_real: 0.049112, Loss_D_fake: 0.453462, acc: 100.00%] [Loss_G: 1.008852]
Steps (30 / 100): [Loss_D_real: 0.065196, Loss_D_fake: 0.458509, acc: 100.00%] [Loss_G: 1.018385]
Steps (40 / 100): [Loss_D_real: 0.061704, Loss_D_fake: 0.459229, acc: 100.00%] [Loss_G: 1.009706]
Steps (50 / 100): [Loss_D_real: 0.051204, Loss_D_fake: 0.460033, acc: 100.00%] [Loss_G: 1.005355]
Steps (60 / 100): [Loss_D_real: 0.059828, Loss_D_fake: 0.457545, acc: 100.00%] [Loss_G: 1.008903]
Steps (70 / 100): [Loss_D_real: 0.056420, Loss_D_fake: 0.468705, acc: 100.00%] [Loss_G: 1.013539]
Steps (80 / 100): [Loss_D_real: 0.052308, Loss_D_fake: 0.452640, acc: 100.00%] [Loss_G: 1.017379]
Steps (90 / 100): [Loss_D_real: 0.053092, Loss_D_fake: 0.457065, acc: 100.00%] [Loss_G: 0.990622]
Steps (100 / 100): [Loss_D_real: 0.052301, Loss_D_fake: 0.467294, acc: 100.00%] [Loss_G: 1.017609]
```

Original data + GAN

Original x_train has total of 227451 transactions, 227451 normal & 394 Fraud

1000

	Time	V1	V2	V3	V4	V5	V6	V7
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000
mean	1.705863	-0.389961	0.402280	-0.679441	0.026302	-0.807352	-0.231370	-0.42902
std	0.469486	0.196144	0.227128	0.295721	0.242404	0.291625	0.233276	0.244944
min	0.602538	-1.098236	-0.274687	-2.079900	-0.865937	-1.734678	-1.088720	-1.45883
25%	1.365533	-0.506325	0.248550	-0.854513	-0.133793	-0.995170	-0.377086	-0.59152
50%	1.651262	-0.377364	0.400120	-0.658193	0.027595	-0.780409	-0.215505	-0.41083
75%	1.977582	-0.258227	0.534954	-0.478576	0.182285	-0.586832	-0.076296	-0.26430
max	3.685767	0.283151	1.292993	0.067721	0.973806	-0.147021	0.420937	0.264664

227451

6501 Progress Report Group 4

	Time	V1	V2	V3	V4	V5	V6
count	227057.000000	227057.000000	227057.000000	227057.000000	227057.000000	227057.000000	227057.00
mean	1.708374	-0.393808	0.398990	-0.670787	0.026841	-0.810465	-0.232531
std	0.462913	0.189808	0.225360	0.281548	0.237714	0.292137	0.232690
min	0.240171	-1.468923	-0.628445	-2.425254	-1.185536	-2.930725	-1.455245
25%	1.377550	-0.516910	0.244866	-0.845061	-0.131113	-0.991146	-0.382999
50%	1.670537	-0.387175	0.391337	-0.645978	0.025858	-0.785670	-0.227924
75%	1.998045	-0.263956	0.545245	-0.470083	0.184095	-0.602572	-0.076197
max	4.212597	0.412227	1.578318	0.269768	1.187458	0.064927	0.826037

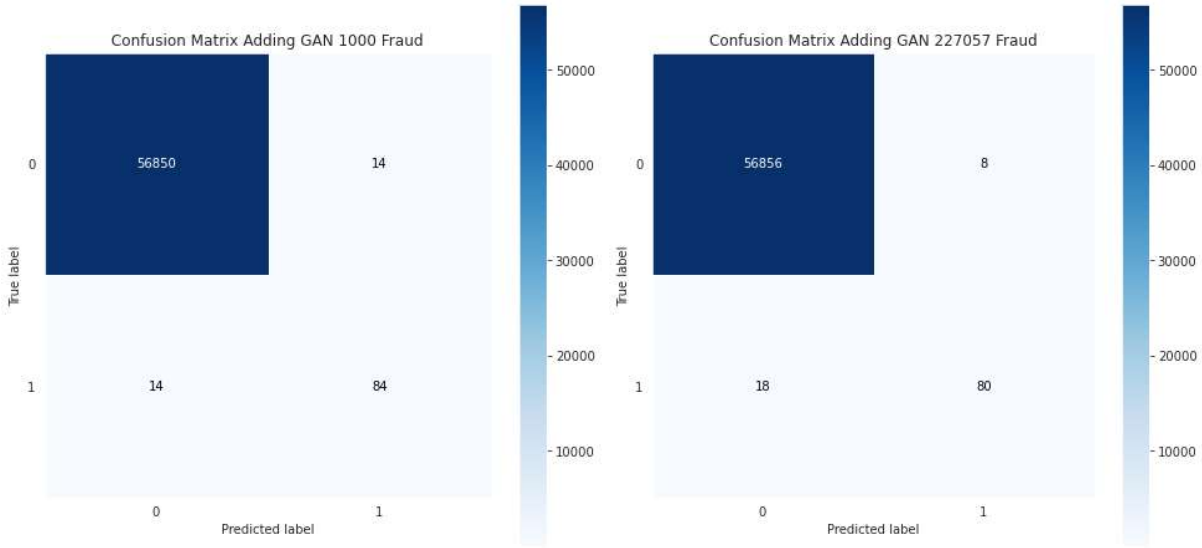
The fraud in x_train

	Time	V1	V2	V3	V4	V5	V6	V7
count	394.000000	394.000000	394.000000	394.000000	394.000000	394.000000	394.000000	394.000000
mean	23.008938	-4.707808	3.588729	-7.068378	4.592975	-3.101629	-1.387192	-5.539909
std	13.347935	6.841390	4.309436	7.166449	2.883467	5.406586	1.864770	7.316745
min	1.239444	-30.552380	-8.402154	-31.103685	-1.313275	-22.105532	-5.773192	-43.557242
25%	11.500278	-5.996596	1.229209	-8.436924	2.419178	-4.741036	-2.504633	-7.765017
50%	21.393056	-2.272114	2.662472	-5.133485	4.258196	-1.522962	-1.421577	-2.926216
75%	35.912917	-0.410418	4.737900	-2.302626	6.390866	0.240184	-0.361122	-0.900824
max	47.318889	2.132386	22.057729	2.250210	12.114672	11.095089	6.474115	5.802537

The little std observed in the GAN generated data indicates **mode collapse** in vanilla gan

Base ROS from Original data Normal: 227451 Fraud: 227451	Add GAN 1000 then ros.fit Normal: 227451 Fraud: 227451	GAN 227057 Normal: 227451 Fraud: 227451
Accuracy: 0.9995084442259752 Precision: 0.8571428571428571 Recall: 0.8571428571428571 F1 score: 0.8571428571428571 ROC AUC score: 0.9284483278398585	Accuracy: 0.9995084442259752 Precision: 0.8571428571428571 Recall: 0.8571428571428571 F1 score: 0.8571428571428571 ROC AUC score: 0.9284483278398585	Accuracy: 0.9995435553526912 Precision: 0.9090909090909091 Recall: 0.8163265306122449 F1 score: 0.8602150537634408 ROC AUC score: 0.9080929220309395

6501 Progress Report Group 4



GAN didn't improve the performance of the classification for now, we think this is because the generator is only producing low spectrum data. We will work on a few improved GAN algorithms to see if the problems are resolved.