Project's Report

1) Problem Statement :-

1.1) Using GANs to fix imbalanced data problems in bank Marketing :-

I worked with bank marketing data that tries to predict if customers will subscribe to term deposit . The problem is that 88.3% of people refuse and only 11.7% accept , making it difficult for machine learning models to learn properly as the dataset is imbalanced , where when we train traditional models we get bad results for minority class .

1.2) what I did to solve this :

I used vanilla GAN (as the basic version) , and WGAN (more advanced than vanilla) to create fake "yes" examples to balance the dataset . Then I used the generated data to train the machine learning model that I chose (which is MLP) .

1.3) Considerations :
- which GAN works better , Vanilla or WGAN ?
- how much better do models get with balanced data ?

1.4) Importance :
Banks waste a lot of money contacting people who won't sign up. If we can better predict who will sign up, they can save resources and time .

2) Description of Dataset & Imbalance Analysis :-

Dataset characteristics :

Name : Bank Marketing Dataset from UCI Machine Learning Repository
Type: Tabular data (customer banking information)
Purpose: Predict if customers will subscribe to term deposits
Features: 16 different customer attributes (age, job, marital status, education , default , balance , housing , loan , contact , day , month , duration , campaign , pdays , previous , poutcome) .

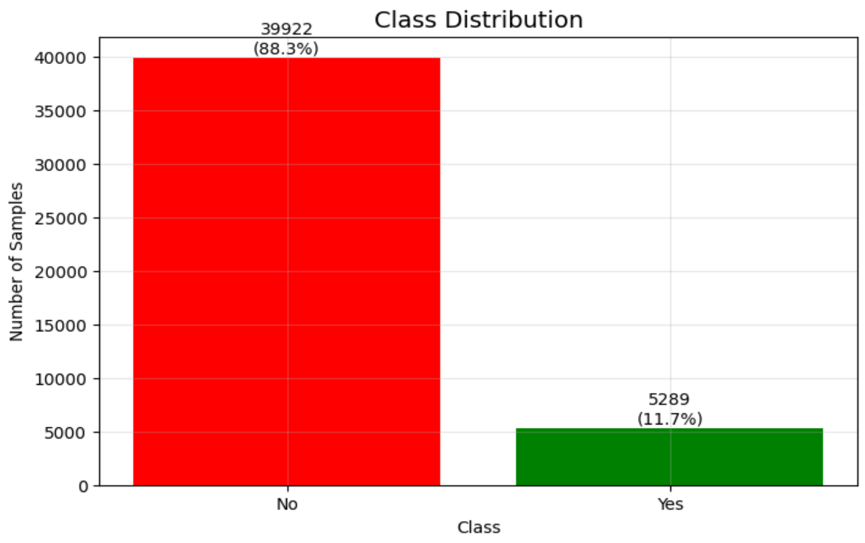
Target: Binary classification , "yes" (will subscribe) or "no" (won't subscribe).

Imbalance Analysis :

Dataset size: 45,211 customer records

Class distribution :

- **Class "no":** 39,922 customers (88.3%) , majority class
- **Class "yes":** 5,289 customers (11.7%) , minority class
- **Imbalance ratio:** 7.55:1



3) Details of GAN Architectures & Training :-

a) Vanilla Gan architecture:

- generator: 100 (random noise) → 256 (ReLU) → 512 (ReLU) → 256 (ReLU) → 16 (Tanh)

- discriminator : 16 (input data) → 256 (LeakyReLU 0.2) → 128 (LeakyReLU 0.2) → 1 (Sigmoid)

b) WGAN architecture :

- generator : 100 (random noise) → 256 (ReLU) → 512 (ReLU) → 256 (ReLU) → 16 (Tanh)

- critic : 16 (input data) → 256 (LeakyReLU 0.2) → 128 (LeakyReLU 0.2) → 1 (Linear)
* Training details :-

(same for both GANS)

Training epochs: 150 , initially started with less number of epochs but increased them to get better results (which happened) .

Batch size: 64

Input noise: 100 dimensions , generator starts with 100 random number

Output features: 16 , Both produce synthetic data with 16 features (same as real data)

Real "yes" samples: 5,289

Generated "yes" samples: 34,633 each , created enough synthetic data to balance the dataset .

- both GANs were trained only on the minority class ("yes" customers) .
- training data was preprocessed: text converted to numbers, features normalized .
- Each epoch: Generator creates fake data, Discriminator/Critic tries to detect fake data / gives scores for data .
- training continued until models could produce realistic synthetic samples
- After training, generated synthetic data was added to balance the original dataset .

4) Classifier Setup and Evaluation:-

I used the MLP classifier :-

- Architecture :- 3 hidden layers with 128 -> 64 -> 32 nodes .
- Activation Function: ReLU for all hidden layers
- Optimizer: Adam with adaptive learning rate
- Training: 300 maximum iterations with early stopping

Training :

- Before training I transformed categorical features to numerical ones , converted labels to 0/1 , and finally used standard scaler to ensures that the model treats all variables with equal importance, preventing features with naturally larger units from overwhelming those with smaller ones.
- train/test split: 70% training, 30% testing
- stratified split: Maintains same class distribution in both sets
- random seed: 42 (for reproducible results)

- feature scaling: standardized all features for better MLP performance
- early stopping: stops training if validation score doesn't improve

I implemented the MLP on :-

-original imbalanced dataset : Trained on original 45,211 samples (88.3% "no", 11.7%"yes") , used for comparison to see how results improved with the GANS .

-Vanilla GAN balanced dataset : Added 34,633 synthetic "yes" samples from Vanilla GAN , with total of 79,844 samples and (50% "no", 50% "yes") .

- WGAN balanced dataset : same as Vanilla GAN , Added 34,633 synthetic "yes" samples from WGAN , with total of 79,844 samples and (50% "no", 50% "yes") .

For evaluation I used 5 metrics :
1) Accuracy
2) Precision
3) Recall
4) F1-Score
5) AUC-ROC

5) Results & Comparisons :-

Original Imbalanced Dataset :

- Accuracy: 0.8998
- Precision: 0.6058
- Recall: 0.4115 (poor - finds 41% of "yes" customers)
- F1-Score: 0.4901 (Poor overall performance)
- high accuracy but struggles to detect minority class

Vanilla GAN Balanced Dataset:

- Accuracy: 0.9383 (+4.3% improvement)
- Precision: 0.9713 (+60.3% improvement)
- Recall: 0.9033 (+119.5% improvement)
- F1-Score: 0.9361 (+91.0% improvement)
- massive improvement in finding "yes" customers

WGAN Balanced Dataset:

- Accuracy: 0.9382 (+4.3% improvement)
- Precision: 0.9682 (+59.8% improvement)
- Recall: 0.9062 (+120.2% improvement - even better!)
- F1-Score: 0.9362 (+91.0% improvement)
- Slightly better recall than Vanilla GAN

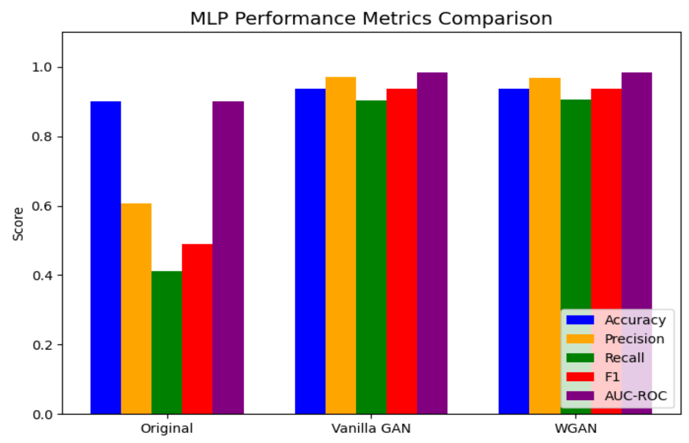
Comparisons :

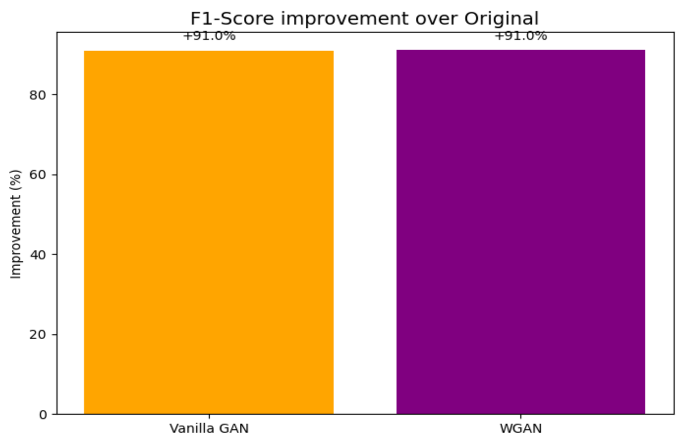
Dataset	Accuracy	Precision	Recall	F1-Score	AUC-ROC	Observation
Original	0.8998	0.6058	0.4115	0.4901	0.8995	High accuracy, poor recall (41%)
Vanilla GAN	0.9383 (+4.3%)	0.9713 (+60.3%)	0.9033 (+119.5%)	0.9361 (+91.0%)	0.9829 (+9.3%)	Finds 90% of cu stomers
WGAN	0.9382 (+4.3%)	0.9682 (+59.8%)	0.9062 (+120.2%)	0.9362 (+91.0%)	0.9831 (+9.3%)	Best recall (91%), best AUC

Aspect	Vanilla GAN	WGAN	Winner
Recall	90.33%	90.62%	WGAN
AUC-ROC	0.9829	0.9831	WGAN
Training Stability	Moderate	High	WGAN
Final Performance	Excellent	Slightly Better	WGAN

- Both GANs successfully balanced dataset from 7.55:1 to 1:1.

- "Training showed WGAN was more stable than Vanilla GAN





6) Observations and Conclusions :-

Observations :-

1) GANs successfully solved imbalance problem:

- both Vanilla GAN and WGAN successfully balanced the dataset from 7.55:1 to 1:1 ratio

- generated 34,633 realistic synthetic "yes" samples for each GAN

- visual plots confirmed synthetic samples closely resembled real minority class data

2) Remarkable performance improvement :

- recall improved by ~120% (from 41% to 90%+) - most important for business
- F1-Score improved by 91% (from 0.4901 to 0.9361-0.9362)
- models went from missing most "yes" customers to finding almost all of them

3. WGAN slightly outperformed Vanilla GAN:

- higher recall (90.62% vs 90.33%)
- higher AUC-ROC (0.9831 vs 0.9829)
- more stable training observed (smooth loss curves)

4. No Overfitting Issues:

- I check for overfitting to make sure the results are legit , and this is what I got :

- All models generalized well to unseen data
- GAN-augmented data actually improved generalization (smaller train-test gaps)
- MLP architecture was well-regularized

*Overfitting results

Original: Training accuracy = 0.907, Testing accuracy = 0.900, Difference = 0.0068

Vanilla GAN: Training accuracy = 0.941, Testing accuracy = 0.938, Difference = 0.0030

WGAN: Training accuracy = 0.941, Testing accuracy = 0.938, Difference = 0.0025

Conclusions :-

Both GANs fixed the imbalance problem extremely well.

- **91% improvement** in F1-Score with both GANs
- **Recall went from 41% to 90%+** (finds almost all "yes" customers)
- **WGAN slightly better** than Vanilla GAN (more stable, better recall)
- **No overfitting** - models generalize well

GANs work great for imbalanced banking data. WGAN is the better choice.