Artificial Intelligence

Program – Machine Learning

EECS 26' You-Ting Liu 111060013

Method

Data Preprocessing

In the beginning, I trained the models with the raw data. However, I found that the F1 score counted from the validation set is much different from the score on Kaggle. The F1 score of the validation set is about 0.79 while the score on Kaggle may be only 0.25 or worse than that. This confused me a lot. After checking the data description on Kaggle again, I noticed that there are a lot of outliers in the training data. Therefore, I wrote data_proccessor.py to delete outliers from the training data and convert them into .npy files to accelerate data loading in the following model training stage. After doing so, the F1 score counted from validation set is very close to the score on Kaggle and the performance of each model is improved a lot.

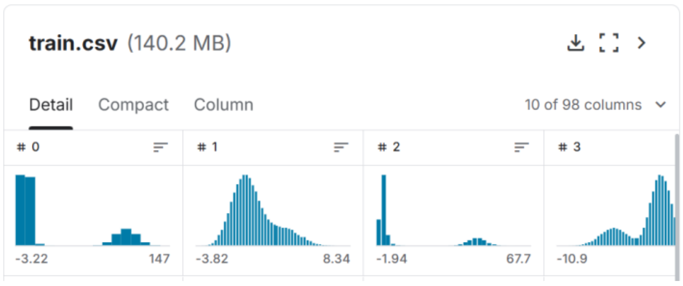

Figure 1

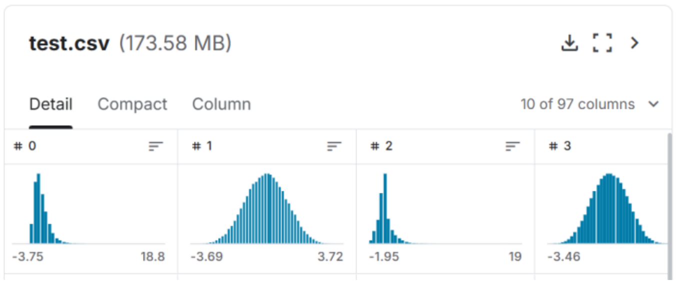

Figure 2

Model

I've implemented two kinds of models. The first one is Extreme Gradient Boost (XGBoost), which is an improved method from Decision Tree. The second one is a 1D neural network. Both have achieved a good performance after some improvements. Let's start with the basic version of them.

Model – XGBoost

For the XGBoost model, I use XGBClassifier from xgboost module with customized scale_pos_weight while all the other arguments are default because labels are unbalance in the training data. Here is how I count scale_pos_weight:

$$scale\_pos\_weight = \frac{\#lable0}{\#lable1} \quad (1)$$

Model – Neural Network

In the neural network model, I used three fully connected layers with ReLU activation. Each layer is followed by batch normalization and a dropout layer with a rate of 0.2. During training, I employed the Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) with a customized pos_weight, the same as the scale_pos_weight used in the XGBoost model, for the same class imbalance reason described in the Model – XGBoost section. To optimize the model, I used the Adam optimizer along with a scheduled learning rate decay to help the model converge to better weights.

Comparation

Although the performance difference between the two models is relatively small in terms of validation F1 score, XGBoost demonstrates significantly higher efficiency in both training and inference. As shown in Table 1, XGBoost achieved a validation F1 score of 0.4671, slightly outperforming the neural network's score of 0.4401. However, the most notable distinction lies in their computational efficiency: XGBoost completed training in just 0.81 seconds and inference in 0.08 seconds, whereas the neural network required 860.77 seconds for training and 4.51 seconds for inference. This highlights XGBoost as a much more practical choice when computational resources or time constraints are a concern.

TABLE 1. MODEL TYPE COMPARISON

MODEL TYPE	XGBoost	Neural Network
VALIDATION F1 SCORE	0.4671	0.4401
TRAINING TIME	0.81 sec	860.77 sec
INFERRING TIME	0.08 sec	4.51 sec

Improvement

I noticed that some features contributed little to the model based on feature importance analysis during decision tree training. Therefore, I removed features with column indices greater than or equal to 36 to improve training and inference speed, while maintaining the original F1 score. Additionally, by analyzing feature importance from the XGBoost model, I selected the 12 most important features to further reduce overhead without sacrificing performance.

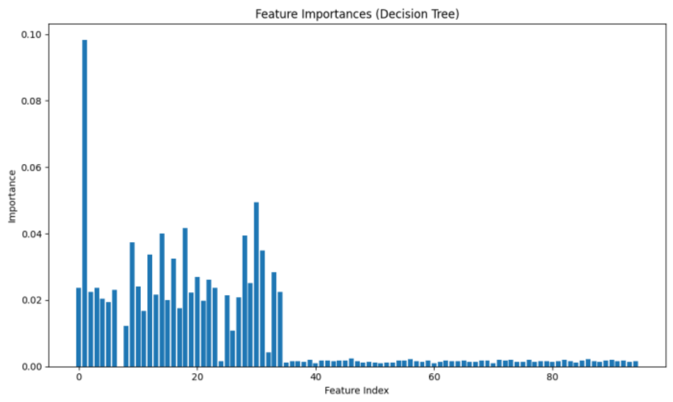

Figure 3. Feature importances (decision tree)

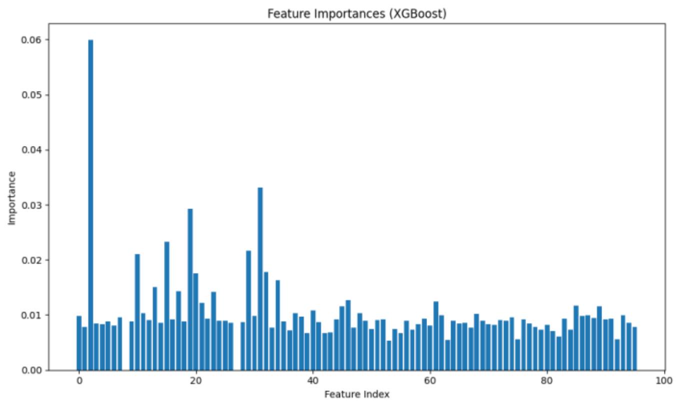

Figure 4. Feature importances (XGBoost)

Test Result

Reducing the number of features led to significant improvements in training and inference time, while maintaining or even enhancing model performance. For XGBoost, decreasing the feature count from 96 to 12 reduced training time by nearly 70%, with only a slight drop in F1 score. In the case of the neural network, fewer features resulted in both faster computation and improved validation F1 scores. These findings suggest that appropriate feature selection can effectively reduce computational overhead without compromising model effectiveness.

TABLE 2. XGBOOST

NUMBERS OF FEATURES	96	36	12
VALIDATION F1 SCORE	0.4671	0.4599	0.4540
TRAINING TIME	0.81 sec	0.62 sec	0.26 sec
INFERRING TIME	0.08 sec	0.07 sec	0.05 sec

TABLE 3. NEURAL NETWORK

NUMBERS OF FEATURES	96	36	12
VALIDATION F1 SCORE	0.4401	0.4688	0.4782
TRAINING TIME	860.77 sec	549.71 sec	545.25 sec
INFERRING TIME	4.51 sec	9.01 sec	3.24 sec

AI Tool Usage Statement

Tool Used

ChatGPT-4o. (2025, May 2). https://chat.openai.com/
Prompt: "{my content} 幫我順一下"

ChatGPT-4o. (2025, April 29). https://chat.openai.com/
Prompt: "<image>{my files structures} {my instruction} 幫我寫 README"

Scope of Use

The AI tool was used solely to improve the clarity, grammar, and flow of written English in certain sections, such as the explanation of feature selection and model performance. The technical content, code, analysis, data interpretation, and all original ideas presented in this homework are entirely my own work. Except the README files, no content was directly generated by the AI in place of original work.

Academic Integrity Declaration

I affirm that the use of AI complied with academic integrity guidelines and did not involve plagiarism, unauthorized content generation, or infringement of copyright.

Model Details

XGBoost

TABLE 4. XGBOOST PARAMETERS

PARAMETER	VALUE
SCALE_POS_WEIGHT	scale_pos_weight
EVAL_METRIC	'auc'
N_JOBS	-1
USE_LABEL_ENCODER	TRUE
BOOSTER	'gbtree'
LEARNING_RATE	0.3
MAX_DEPTH	6
MIN_CHILD_WEIGHT	1
GAMMA	0
SUBSAMPLE	1
COLSAMPLE_BYTREE	1
REG_ALPHA	0
REG_LAMBDA	1

Neural Network

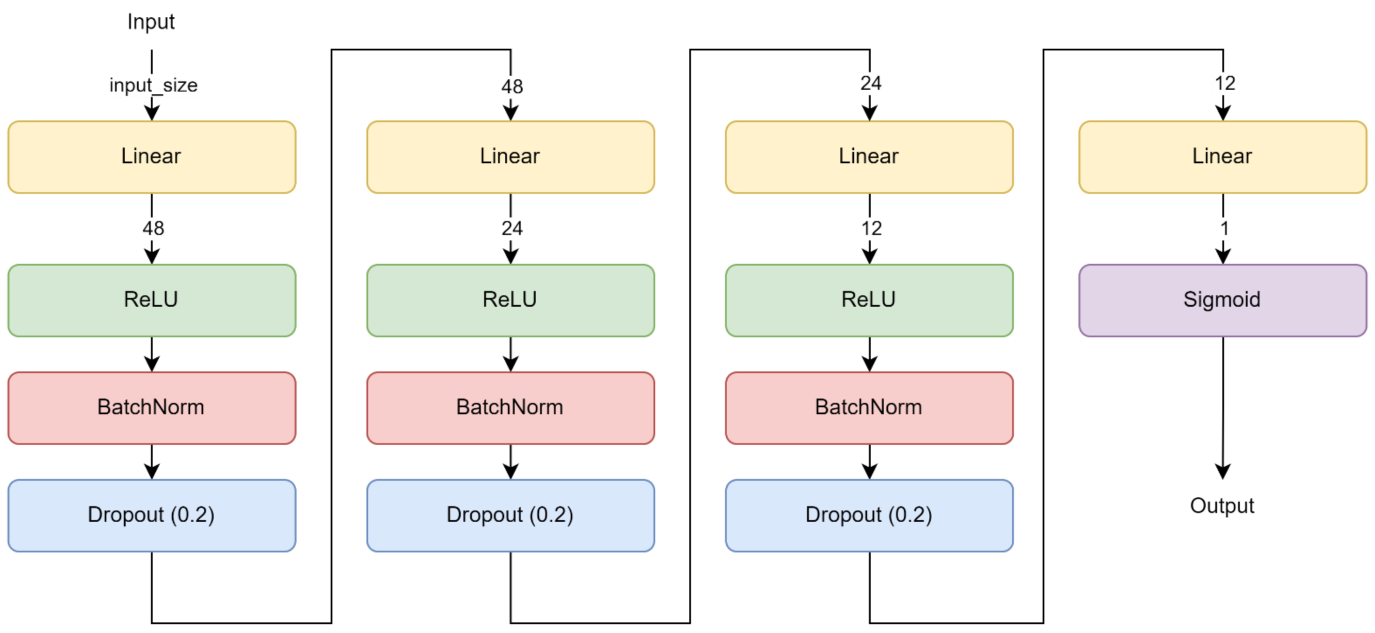

Figure 5. Neural network model architecture

TABLE 5. NEURAL NETWORK PARAMETERS

PARAMETER	VALUE
BATCH_SIZE	32
NUM_EPOCHS	50
LEARNING_RATE	0.001
SCHEDULER	StepLR
DECAY_STEP	12
DECAY_RATE	0.3
LOSS FUNCTION	BCEWithLogitsLoss(pos_weight)
OPTIMIZER	Adam
METRIC	F1 Score
TRAIN / VALIDATION SPLIT	80% / 20%

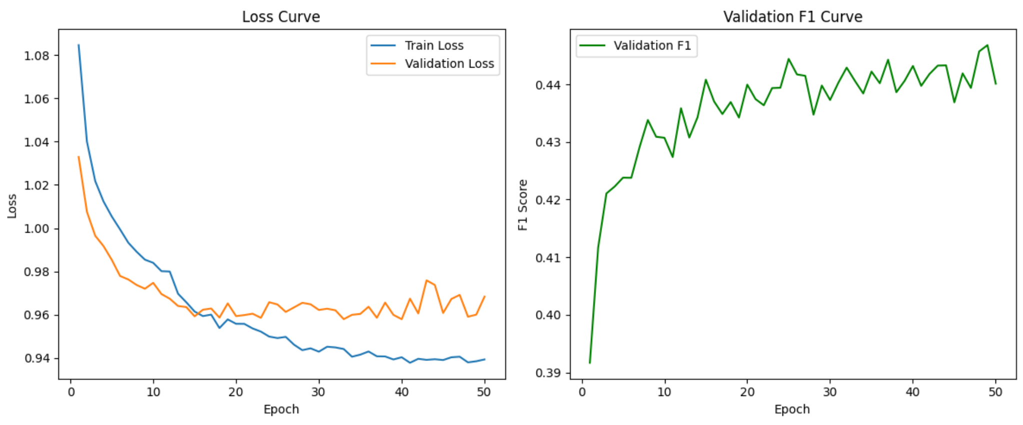

Figure 6. Training process with 96 features

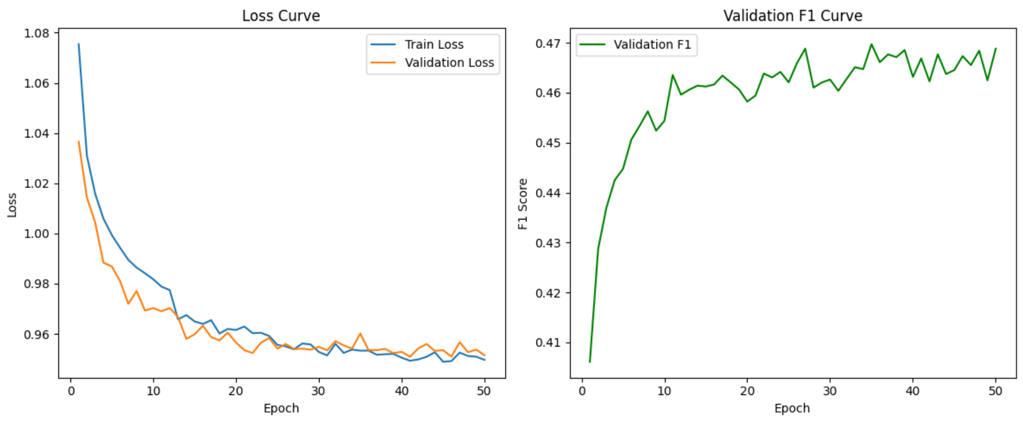

Figure 7. Training process with 36 features

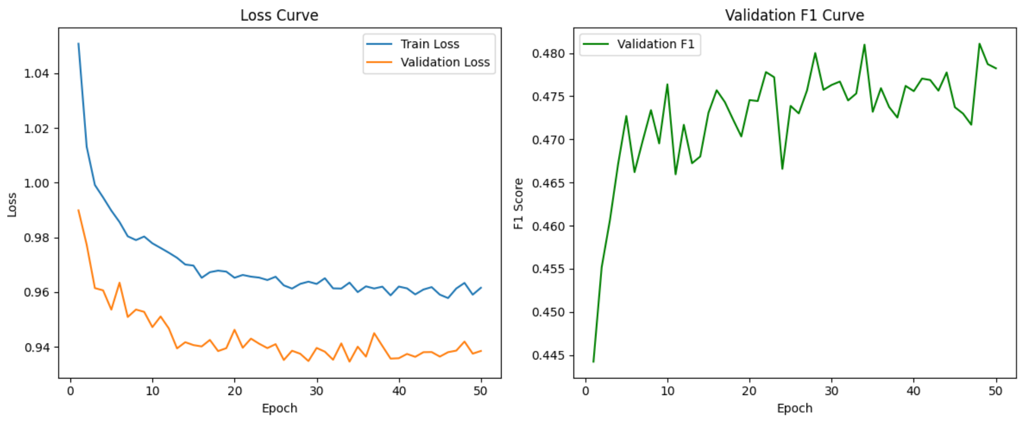

Figure 8. Training process with 12 features