Dean&Francis
ISSN 2959-6157

Different Performance of Transformer and Logistic Regression Models in Credit Risk Prediction

Yanran Lu[1*],

Jingxuan Lyu[2],

Minghong Ma[3],

Shenxin Yi[4]

[1]Qingdao No.2 Middle School, Qingdao, 266061, China, yanran.lu-alice@outlook.com
[2]Haidian Kaiwen Academy, Beijing, 100195, China, john.lyu.bj@hotmail.com
[3]Chengdu Foreign Languages School, Chengdu, 611731, China, minghongma2007@gmail.com
[4]Wuhan Britain-China School, Wuhan, 430030, China, yishenxin2025a@163.com

*Corresponding author email: yanran.lu-alice@outlook.com

Abstract:

Nowadays, people's use of credit cards is increasing significantly, and the importance of predicting credit default for banks has become higher. This paper aims to compare the different performances of two binary classification methods--logistic regression and transformer--in credit risk performance to determine which one bank should use wider when making loans. In the experiment, we selected 600,000 pieces of data randomly and then applied them in both logistic regression and transformer models. As a result, we evaluated the performance of each model in various aspects, and we eventually found that logistic regression is more accurate than transformer. So we can conclude that although there are diverse novel credit scoring models appearing in the world, logistic regression is still one of the most practical, useful and precise ones, which not only saves time but also performs well. However, in the future, if the data features get more complicated, people might discover more uses of transformers in the economic field, especially in credit risk prediction.

Keywords:-credit risk, economics, logistic regression, transformer, binary classification

1 Introduction

Personal loans generally feature a set interest rate, loan term, and monthly payments, offering borrowers the flexibility to cover various expenses. In the United States, personal loans have been gaining popularity, with the outstanding balance increasing by almost 59% over the past two years, rising from $146 billion to $232 billion. The interest rates for personal loans are usually higher than those for other types of loans because they are often unsecured [1].

Given the significant volume of personal loans, it is important to consider measures to mitigate lending risks. Credit scoring is the process of evaluating the risk associated with lending to individuals or organizations. In the US and the UK, most adults undergo monthly scoring to assist lenders in determining whether to lend [2]. The risk is evaluated based on many factors, such as credit history, capacity to repay, capital and so on. It is essential to depend on

ISSN 2959-6157

models and algorithms instead of human judgment due to the large amount of data available. Scores generated by models can provide an overview of consumers' creditworthiness, and lending institutions may develop their own unique risk models using confidential borrower information.

This paper aims to use "machine learning model", which refers to a set of algorithms specifically designed to predict the outcome by analyzing extremely large datasets, to analyze consumer credit risks. Machine learning models are able to perform more accurately than traditional credit scoring models in predicting default risk because a wider range of data is allowed when using machine learning models, and they also have the ability to learn from past data.

In section one, several papers written by other people on this topic will be summarized and compared, given their commonness and arguments. Section two focuses on methodology. Two methods of machine learning on credit risks will be suggested and the corresponding pseudocode will be shown. Section three is data description. This includes the description of how we gain the dataset and data preprocessing, such as label balancing and feature engineering. Section four is the experiment setting and experiment results. Section five will discuss whether there is any problems or limitations and some comparisons between our methods and others' methods will be done. Finally, section six is the conclusion, which includes the summary and possible solutions to the limitations.

2 Literature Review

Predicting credit scores has been achieved through the use of many machine learning models. Johannes Kriebel relies on six deep learning architectures and other techniques to get credit-related information from text generated by users on Lending Club, which is considered an essential peer-to-peer lending platform in the U.S. The deep learning methods include convolutional neural networks, recurrent neural networks, average embedding neural networks, bidirectional encoder representations from transformers (BERT), and a robustly optimized BERT pretraining approach (RoBERTa) [3]. His research indicates that deep learning methods are typically more effective than other machine learning approaches that utilize word frequencies, pre-trained word embeddings, or topic models in almost all scenarios.

Sun and Vasarhelyi applied several popular machine learning models, such as logistic regression, deep neural networks, traditional artificial neural networks, and decision trees, to predict credit risk. The dataset includes 711,397 credit card holders from a prominent Brazilian bank [4]. The research showed that the Deep neural network had the best performance within the models, showed by the study.

Arram et al. introduce a new credit card default dataset for an American bank and investigate various machine learning models for the enhancement of defaulting credit card prediction. They compare the prediction performance using the following machine learning models: logistic regression, LightGBM, neural network, MLP, and so on. [4] The main goal is to identify the machine learning model that performs best and extract the most important features from the proposed dataset. The dataset consists of 500 entries, each containing 36 features. The features have been described in 12 of them, and the remaining 24 have been anonymized and labeled as var0 to var12. Handling missing values, outliers, and imbalanced datasets are the three main steps in their data preprocessing process. LightGBM is the most accurate model in their experiments, but MLP is the most effective model for effectively identifying potential credit card fraud customers.

Yu Cheng et al. have done research on the credit risk early warning model of commercial banks with the help of a neural network algorithm. The paper first introduces several financial risk early warning models: ARCH (Autoregressive Conditional Heteroskedasticity) model, logistic regression modeling and ARMA (Autoregressive Moving Average) model. In addition to these models, the paper outlines the benefits of employing BP neural networks for early warning of operational risks in commercial banks: not only does it streamline risk control processes, but it also improves the ability of banks to navigate future risk dynamics efficiently. The study consists of 135 samples that come from four diverse of industries: papermaking, electronics, chemicals, and brewing. The samples, which have been carefully selected and processed, are divided into two groups for training and testing purposes. The training set is made up of 112 samples, with 103 coming from non-ST companies and 9 from ST companies, and the testing set has 23 samples, 18 coming from non-ST companies and 5 from ST companies. [5] The effectiveness and reliability of the neutral network algorithm in practical applications are demonstrated in this paper.

Based on previous research results, we can divide the methods into two parts: the traditional statistic model and deep learning methods. Previous research shows that deep learning methods have higher accuracy than traditional statistic model. But traditional statistic model has higher interpretability and can better fit the problem of credit risk. Therefore, this paper decides to combine these two methods to enhance both accuracy and interpretability. The methods we choose to apply are logistic regression and transformer, including gradient boosting.

3 Methodology

The paper utilizes two methods: logistic regression and transformer. Each method is used to predict credit risk outcomes and is compared to determine the most efficient approach.

3.1 Logistic regression

Logistic regression, a machine learning algorithm, is supervised and employed to deal with classification tasks, which aim to predict the likelihood of an instance belonging to a given class or not. A sigmoid function is employed for binary classification to generate a probability value between 0 and 1 from input variables. [6] In logistic regression, the sigmoid function is frequently employed as an activation function. The horizontal axis is used to represent the input value in a sigmoid function graph, while the vertical axis represents the function value. The logistic regression value is restricted to between 0 and 1, resulting in a curve that resembles the 'S' form. The mathematical expression is:

$$f(x)=\frac{1}{1+e^{-x}}$$

In the case of credit risk, the independent input features are factors that are taken into consideration when judging a consumers' credit.

Let the independent input features be:

$$X=\begin{bmatrix} x_{11} & \cdots & x_{1m} \\ ? & \ddots & ? \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

and the dependent variable is Y, which has only binary value i.e. 0 or 1.

$$Y=\begin{cases} 0 if class 1 \\ 1 if class 2 \end{cases}$$

then, the multi-linear function is applied to the input variables X.

$$z=\left(\sum_{i=1}^{n} w_i x_i\right)+b$$

Here x_i is the ith observation of X. $w_i=[w_1,w_2,w_3,\cdots,w_m]$ is the coefficient. b is the intercept.

Now the sigmoid function is used where the input will be z and the probability is found between 0 and 1. Figure 1 shows the image of the sigmoid function.

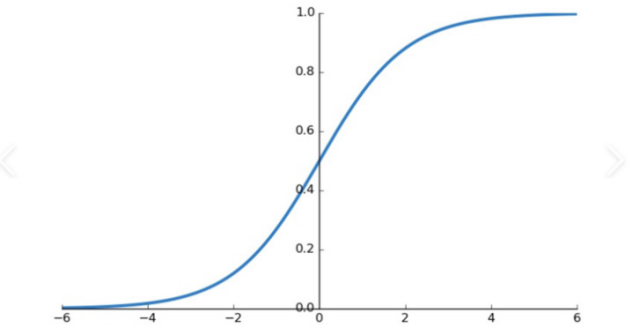

Figure 1 graph of sigmoid function

l $\sigma(z)$ tends towards 1 as $z\to\infty$

l $\sigma(z)$ tends towards 0 as $z\to-\infty$

l $\sigma(z)$ is always bounded between 0 and 1

$$p(y=1)=\sigma(z)$$

$$p(y=0)=1-\sigma(z)$$

The odd is determined by the ratio of something occurring to something not occurring.

$$\frac{p(x)}{1-p(x)}=e^{x}$$

Applying natural log on odd.

$$log\left[\frac{p(x)}{1-p(x)}\right]=z$$

$$log\left[\frac{p(x)}{1-p(x)}\right]=w\bullet X+b$$

$$\frac{p(x)}{1-p(x)}=e^{w\bullet X+b}$$

$$p(x)=e^{w\bullet X+b}\bullet(1-p(x))$$

$$p(x)=e^{w\bullet X+b}-e^{w\bullet X+b}\bullet(p(x))$$

$$p(x)+e^{w\bullet X+b}\bullet(p(x))=e^{w\bullet X+b}$$

$$p(x)\left(1+e^{w\bullet X+b}\right)=e^{w\bullet X+b}$$

$$p(x)=\frac{e^{w\bullet X+b}}{1+e^{w\bullet X+b}}$$

then the final logistic regression equation will be:

$$p(X;b,w)=\frac{e^{w\bullet X+b}}{1+e^{w\bullet X+b}}=\frac{1}{1+e^{-w\bullet X+b}}$$

The likelihood function for Logistic Regression is:

$$L(b,w)=\prod_{i=1}^{n} p(x_i)^{y_i}\left(1-p(x_i)\right)^{1-y_i}$$

Taking natural logs on both sides we can get:

$$log\left(L(b,w)\right)=\sum_{i=1}^{n}-log1+e^{w\bullet X+b}+\sum_{i=1}^{n} y_i(w\bullet x_i+b)$$

In order to locate the most probable estimates, we differ-

ISSN 2959-6157

entiate with respect to w,

$$\frac{\partial J\left(L(b,w)\right)}{\partial w_j} = \sum_{i=n}^{n}(y_i - p(x_i;b,w))x_{ij}$$

Figure 2 exhibits the python codes of binomial logistic regression:

```
# import the necessary libraries
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True)

# split the train and test dataset
X_train, X_test,\
    y_train, y_test = train_test_split(X, y,
                                       test_size=0.20,
                                       random_state=23)
# LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)

# Prediction
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Logistic Regression model accuracy (in %):", acc*100)
```

Figure 2 basic codes of logistic regression

3.2 Transformer

Transformer is a neural network architecture that is utilized for carrying out machine learning tasks. Transformer models work by processing input data through a series of layers.[7] Figure 3 is the framework of transformer model.

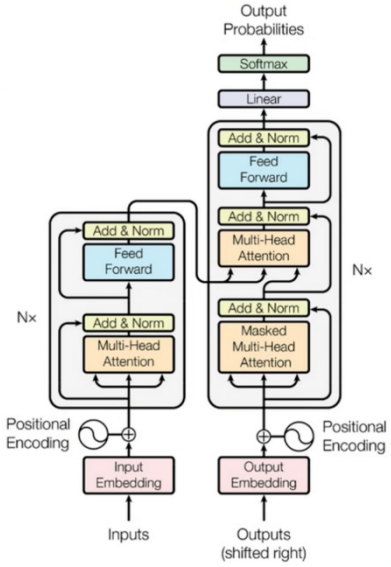

Figure 3 main structure of transformer

Imagine that you are required to translate an English sentence into Chinese. Here are the steps involved by using a transformer model.

l Input embedding

The English sentence to be translated enters the input embedding, while the Chinese annotation enters the output embedding. First, the input sentence is converted into embedding, which captures the semantic meaning of the tokens in the input sequence. Word vectors are created by converting a word sequence.

l Positional encoding

Once the word vectors for input and output are acquired, the transformer will encode these vectors positionally. The model doesn't know the order of various words. Positional encoding is therefore a set of extra values or vectors that are included in the token embedding before being fed into the transformer model. Cosine and sine are used in the calculated.

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

pos represents the position in the sequence. For different dimensions of the same word, the dimension in the even digit corresponds to $2i$ and the dimension in the odd digit corresponds to $2i+1$. d represents the total number of dimensions.

l Encoder and decoder

After positional encoding, the English sentence to be translated and the Chinese annotation will enter the encoder on the left and the decoder on the right, respectively. The data goes through three calculation processes. The encoder encodes the data of English sentence to be translated based on the multi-head attention and feedforward neural network. The decoder encodes the data of Chinese annotation based on the masked multi-head attention.

l Multi-head self attention

In 'attention heads', self-attention is utilized to document various relationships between objects. People use softmax function to calculate attention weights in the self-attention mechanism.

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Through the multi-head attention mechanism, the transformer can calculate and attach the general information in the input data to the decoded result. Multi-head self attention will use Q, K, V linear layers respectively to perform feature transformations on the input data.

l Feedforward neural networks

Feedforward layers process the self-attention layer's output. By transforming the token representations in these networks in a non-linear way, the model is able to identify intricate the relationship and the patterns of the data.

l Stacked layers
Each stacked layer processes the output of the previous layer to improve the representations. Hierarchical and abstract features in the data can be captured by the model.
l Output layer
A separate decoder module can be built on top of the encoder in sequence-to-sequence applications, such as neural machine translation, in order to produce the output sequence.
l Training
Transformer models are taught through supervised learning to minimize a loss function that measures the variance between the model's predictions and the actual data for the task.
l Inference
Once the model has been trained, it can be used for inference to deal with data that has not been encountered. [8]

3.3 Summary of the whole process:

Firstly, the English sentence to be translated and the Chinese annotation go through the embedding layer to get word vectors and go through positional encoding to encode these vectors positionally. Then, English word vectors will go though N encoders, which include multi-head self attention and feedforward neural networks. Chinese word vectors will go though N decoders that include masked multi-head attention and feedforward neural networks. Then, the output value of decoder can be calculated. The final translated value will be calculated by processing the output of the decoder through a linear and softmax layer. Figure 4 and figure 5 shows the fundamental codes of transformer model.

```
import torch
import torch.nn as nn

from timm.models.layers import PatchEmbed, Mlp, DropPath
from timm.models.registry import register_model


class Attention(nn.Module):
    def __init__(self, dim, num_heads=1, qkv_bias=False, qk_scale=None, attn_drop=0., proj_drop=0.):
        super().__init__()
        self.num_heads = num_heads
        head_dim = dim // num_heads
        self.scale = qk_scale or head_dim ** -0.5

        self.qkv = nn.Linear(dim, dim * 3, bias=qkv_bias)
        self.attn_drop = nn.Dropout(attn_drop)
        self.proj = nn.Linear(dim, dim)
        self.proj_drop = nn.Dropout(proj_drop)

    def forward(self, x):
        B, N, C = x.shape
        qkv = self.qkv(x).reshape(B, N, 3, self.num_heads, C // self.num_heads).permute(2, 0, 3, 1, 4)
        q, k, v = qkv[0], qkv[1], qkv[2]  # make torchscript happy (cannot use tensor as tuple)

        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = attn.softmax(dim=-1)
        attn = self.attn_drop(attn)

        x = (attn @ v).transpose(1, 2).reshape(B, N, C)
        x = self.proj(x)
        x = self.proj_drop(x)
        return x


class Block(nn.Module):

    def __init__(self, dim, num_heads=1, mlp_ratio=4., qkv_bias=False, qk_scale=None, drop=0., attn_drop=0.,
                 drop_path=0., act_layer=nn.GELU, norm_layer=nn.LayerNorm):
        super().__init__()
        self.norm1 = norm_layer(dim)
        self.attn = Attention(
            dim, num_heads=num_heads, qkv_bias=qkv_bias, qk_scale=qk_scale, attn_drop=attn_drop, proj_drop=drop)
        # NOTE: drop path for stochastic depth, we shall see if this is better than dropout here
        self.drop_path = DropPath(drop_path) if drop_path > 0. else nn.Identity()
        self.norm2 = norm_layer(dim)
        mlp_hidden_dim = int(dim * mlp_ratio)
        self.mlp = Mlp(in_features=dim, hidden_features=mlp_hidden_dim, act_layer=act_layer, drop=drop)

    def forward(self, x):
        x = x + self.drop_path(self.attn(self.norm1(x)))
        x = x + self.drop_path(self.mlp(self.norm2(x)))
        return x


class RTransformer(nn.Module):
    def __init__(self):
        super().__init__()
        self.dim = 29
```

Figure 4. basic codes of transformer

ISSN 2959-6157

```
        self.decoder_depth = 1
        self.blocks = nn.Sequential(*[
            Block(
                dim=self.dim,
                drop=0.2,
                attn_drop=0.2,
            )
            for i in range(self.decoder_depth)])
        self.relu = nn.ReLU()
        # self.softmax = nn.Softmax()
        self.cls_token = nn.Parameter(torch.zeros(1, 1, self.dim))  # 1, 1, 8000

        # self.fc1 = nn.Linear(60, 512)
        self.fc1 = nn.Linear(self.dim, 512)

        self.fc2 = nn.Linear(512, 2)
        # self.fc2 = nn.Linear(512, 5)

        torch.nn.init.normal_(self.cls_token, std=.02)

    def forward(self, x):
        # x -> bz x 59 x 8000
        # print(x.shape)
        cls_token = self.cls_token.repeat(x.shape[0], 1, 1)  # bz x 1 x 8000
        x = torch.cat((x, cls_token), 1)  # bz x 60 x 8000
        x = self.blocks(x)

        # cls_token = x[:, -1, :].squeeze(1).unsqueeze(-1) # bz x 8000 x 1
        # attn_map = torch.bmm(x, cls_token).squeeze(-1) # bz x 60
        # attn_map = self.relu(self.fc1(attn_map))
        # attn_map = self.fc2(attn_map)
        # return attn_map

        cls_token = x[:, -1, :].squeeze(1)
        cls_token = self.relu(self.fc1(cls_token))
        cls_token = self.fc2(cls_token)
        # print(cls_token)
        return cls_token
```

Figure 5. basic codes of transformer

4 Data

4.1 Data Source

The data set is called the Credit Card Fraud Detection Dataset 2023, coming from the Kaggle website (https://www.kaggle.com). In this dataset you will find data about the credit card transactions made by European cardholders in 2023. The data, including more than 560,000 records in total, has been anonymized to protect cardholders' identities. The primary goal of the dataset is to make it easier to create models and algorithms for fraud detection that can be used to spot potential fraudulent transactions.

4.2 Data Features

There are 31 columns in total and can be divided into 4 categories. The first column represents the unique identifier for each transaction, numbered from 1 to 568629, and was denoted as . Then from the 2nd to 29th column, these 28 columns are anonymized features representing various transaction attributes. These attributes may include time, location, etc. They are denoted as . The penultimate column means the transaction amount, denoted as .

Finally, the last column is denoted as . Through the binary label, this column can indicate whether the transaction is fraudulent (1) or not (0). There are a total of 568631 data. Label 0 has 284315 data, and label 1 has 284316 data.

5 Model

5.1 Data processing

We randomly selected 30,000 pieces of data labeled 0 and 30,000 pieces labeled 1, and dropped the id column. Then we combined them in one dataset and normalized it, which is called data after processing. After concatenating these two subsets, we forged a unified structure that served as the foundation for our analysis.

5.2 Data splitting

We divided the data after processing into two parts, the training data, which includes 80% of the 60,000 pieces of data, and the test data, which includes 20% of the 60,000 pieces of data.

5.3 Model training and testing

5.3.1 Logistic Regression

In order to predict credit risks, we employ logistic regression as one of our model, which is a fundamental technique used in machine learning.

We served the data after process as the foundation for our logistic regression analysis.

It took only few seconds to produce the result of logistic regression model.

5.3.2 Transformer

We defined the parameters of the model in order to get the best result. The input size of the dataset is 29, the hidden size is 128, the number of layers is 2, the output size is 2, and the learning rate we defined is 5e-6. Simultaneously, the number of epochs is 1000, the L2 regression is 0.001, and the batch size is 128.

It took about 50 minutes to train the transformer model, much longer than logistic regression.

5.4 Experiment result

5.4.1 Logistic Regression

The model is trained, which yields an impressive accuracy of 99.7%. This figure, while seemingly indicative of a accurate model, also raised the possibility of overfitting, which we were keen to avoid.

To address this concern, we apply other evaluations to our logistic model, including confusion matrix and cross validation. When used in conjunction with cross-validation, the confusion matrix can show how consistent the model's performance is across different subsets of the data. If the confusion matrix results are similar across folds, it suggests that the model is generalizing well and is less likely to be overfitting. By dividing our dataset into multiple subsets and iteratively training and validating the model, we can ensure that our findings were not an character of the particular way the data was split. The confusion matrix is shown in figure 6, and the cross validation result were

displayed in figure 7:

$$\begin{bmatrix} 5989 & 8 \\ 19 & 5984 \end{bmatrix}$$

figure 6. Confusion matrix of logistic regression

```
Cross-Validation Scores: [0.99758333 0.998      0.99766667 0.9985     0.998     ]
Average Score: 0.99795
```

figure 7. Cross validation result of logistic regression

Gratifyingly, the cross-validation results corroborated the absence of overfitting. Each score in the Cross-Validation Scores array represents the accuracy of the model on a different fold of the data. The scores are all very close to 1, which suggests that the model is consistently accurate across different subsets of the data. In addition, the scores are clustered around the average score of 0.99795, which is a good sign as it suggests that the model is stable and not sensitive to the particular way the data is split.

5.4.2 Transformer

We use accuracy, recall, and precision to evaluate the performance of the transformer. Figure 8, figure 9, figure 10 and figure 11 respectively show the accuracy, recall, precision and loss image of transformer training process. Then we start testing the model, and we find that the accuracy is 95.13%, the recall value is 95.13%, and the precision is 95.35%.

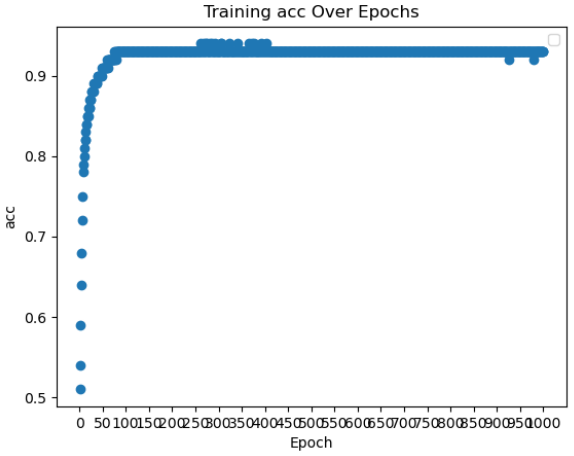

Figure 8. Accuracy image of transformer training

Figure 9. recall image of transformer training

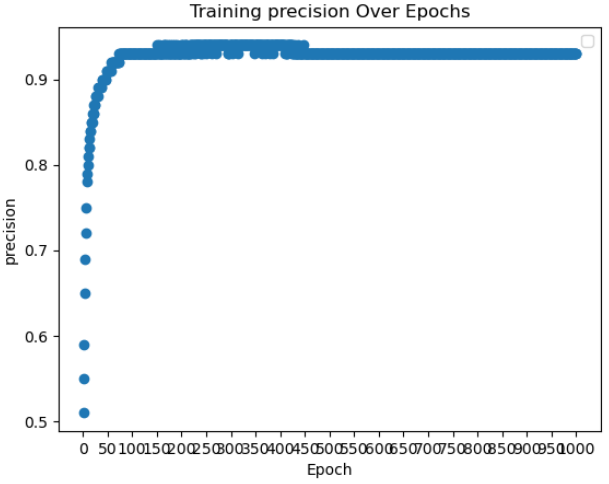

Figure 10. precision image of transformer training

Figure 11. loss image of transformer training

ISSN 2959-6157

6 Conclusion

Although we guessed that transformer may be more sensitive to the classification of credit default before the research, the experiment result shows that for credit risk area, logistic regression is still more effective and practical. However, if the features are in non-linear relationship, transformer might be able to predict credit risk more successfully. Besides, if there are features with time included in the dataset, transformer may also be an ideal model to deal with the problem. Therefore, through more rigorous and accurate experiment, people could discover a wider use of transformer in economics.

Acknowledgement

Yanran Lu, Jingxuan Lyu, Minghong Ma and Shenxin Yi contributed equally to this work and should be considered co-first authors.

References

[1] Horan, S. (2024a) Personal Loan Data and statistics (2024), MarketWatch. Available at: https://www.marketwatch.com/guides/personal-loans/personal-loan-statistics/#:~:text=Key%20Statistics%201%20Nearly%2023%20million%20Americans%20have,and%20lowest%20among%20Generation%20Z%20%28%247%2C684%29.%20More%20items (Accessed: 11 July 2024).

[2] Author links open overlay panelJonathan N. Crook a *et al*. (2011) *Recent developments in Consumer Credit Risk Assessment, European Journal of Operational Research*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0377221706011866 (Accessed: 10 July 2024).

[3] Author links open overlay panelJohannes Kriebel *et al*. (2021) *Credit default prediction from user-generated text in peer-to-peer lending using Deep Learning, European Journal of Operational Research*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S037722172101078X (Accessed: 10 July 2024).

[4] Arram, A. *et al.* (2023) *Credit card score prediction using Machine Learning Models: A new dataset, arXiv.org*. Available at: https://arxiv.org/abs/2310.02956 (Accessed: 10 July 2024).

[5] Cheng, Y. *et al.* (2024) *Research on credit risk early warning model of commercial banks based on neural network algorithm, arXiv.org*. Available at: https://arxiv.org/abs/2405.10762 (Accessed: 10 July 2024).

[6] *Logistic regression in machine learning* (2024) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/understanding-logistic-regression/ (Accessed: 10 July 2024).

[7] *Transformers in machine learning* (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/getting-started-with-transformers/ (Accessed: 10 July 2024).

[8] *What is a transformer model?* (2023) *IBM*. Available at: https://www.ibm.com/topics/transformer-model (Accessed: 10 July 2024).