Feed-Forward Neural Network for Binary Classification: Customer Churn Prediction

Seunghyun Park

260686853

Introduction

Customer churn prediction is a critical component of modern business strategy, enabling companies to identify and retain valued clients. Accurate risk assessment and effective churn prevention can significantly impact revenue and customer loyalty. While traditional machine learning methods have been employed for prediction, this study will investigate the application of Feed-Forward Neural Networks (FFNs) to enhance customer churn prediction. It will be consisted of developing, training, and evaluating Feed-Forward Neural Network model for the binary classification of customer churn with different network architectures, activation functions, and learning rates and compare their performance. The model leverages a dataset containing multiple features related to customer demographics, account details, and banking behavior.

Data

The dataset used for the analysis includes data of 10,000 customers, with 18 features. The features include demographic information, account details, and behavioral metrics.

Column	Data Type	Description	Missing Value
RowNumber	int64	Identifier for the record number	0
CustomerId	int64	Unique identifier for the customer	0
Surname	object	Last name of the customer	0
CreditScore	int64	A score representing the customer's creditworthiness	0
Geography	object	Customer's location	0
Gender	object	Customer’s gender	0
Age	int64	Customer’s age	0
Tenure	int64	Number of years the customer has been with the bank	0
Balance	float64	Account balance	0
NumOfProducts	int64	Number of bank products used by the customer	0
HasCrCard	int64	Indicates if the customer has a bank-issued credit card	0
IsActiveMember	int64	Customer’s status	0
EstimatedSalary	float64	Customer’s estimated salary	0
Exited	int64	Whether the customer has left the bank	0
Complain	int64	Indicates if the customer has filed complaints	0
Satisfaction Score	int64	Customer's satisfaction rating	0
Card Type	object	Type of card held (e.g., standard, gold)	0
Point Earned	int64	Points accumulated by the customer	0

Table 1: Data Description

Data Preprocessing

Irrelevant Variables

Columns such as RowNumber, CustomerId, and Surname were removed from the data, as they are not relevant to customer churn.

Categorical Variables

Categorical features like Geography, Gender, and Card Type were encoded using One-Hot encoding and converted into a numerical format.

Numerical Features

Numerical features such as Credit Score, Age, and Estimated Salary were normalized using MinMaxScaler, ensuring that all features contributed equally to the model's predictive capabilities, without bias towards a particular feature's scale.

The data was then split into training (60%), validation (20%), and testing (20%) sets to validate and evaluate the model.

Feed-Forward Neural Network Model Overview

Input Layer

The input layer comprises 20 neurons, each representing a key feature extracted from the dataset such as demographics, account information, and behavioral data.

Hidden Layers

The model experiments with two different numbers of hidden layers: one with a single hidden layer and another with two hidden layers, both evaluated for their performance metrics. Each hidden layer has 200 neurons and the activation functions ReLU (Rectified Linear Unit) and Leaky ReLU are used and compared in terms of performance.

Output Layer

The output layer consists of a single neuron with sigmoid activation function, which is ideal for binary classification.

Dropout Regularization:

The dropout regularization with rate of 0.2 is employed in the hidden layer to prevent overfitting, dropping out about 20% of the neurons randomly during training iteration.

Initialization

He initialization is used for hidden layers with ReLU and Leaky ReLU activation functions, while Xavier initialization is utilized for the output layer with a sigmoid function.

Loss Function

For the loss function, Binary Cross-Entropy Loss is utilized across all models, as it is well-suited for binary classification tasks.

Optimizers

Two different optimizers, Momentum SGD and ADAM, are experimented in this study to avoid overfitting. The Momentum SGD is configured with a momentum of 0.9 and a learning rate of 0.01, with L2 regularization (weight decay of 0.001). The Adam optimizer with a learning rate of 0.001 is employed for the comparison with the Momentum SGD optimizer.

Early Stopping

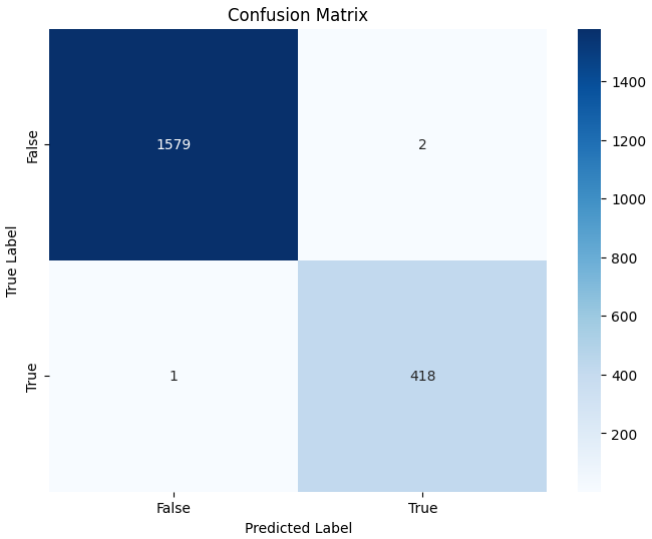
Early stopping is used to prevent overfitting. The iteration stops if no improvement in validation loss is observed for 5 consecutive epochs, minimizing overfitting by stopping the training when the model no longer improving its performance. The initial minimum value for loss is set to infinity.

Training Protocol

The maximum number of epochs is preset at 10,000, with early stopping criteria can terminate iterations if necessary.

Result

The baseline model, with a single hidden layer and ReLu as an activation function of the hidden layer, utilizes a momentum optimizer with learning rate of 0.01, momentum value of 0.9, and L2 regularization with a weight decay parameter of 0.001. This model has achieved accuracy of 99.85%, precision of 0.9952, recall of 0.9976 and F1 of 0.9964 after training over 4,714 epochs.



Plot 1: Confusion Matrix of Baseline Model

Model Comparison

All the models have achieved high accuracy across different configurations of hidden layers, learning rates, optimizers, and activation functions.

Learning Rate

When the learning rate increases from 0.01 to 0.05 for the same model configuration (single hidden layer, ReLU activation function, and Momentum Optimizer), there is a dramatic reduction in the number of epochs required to converge (from 4714 to 61) while the accuracy, precision, recall, and F1 scores remain consistent. This suggests that the higher learning rate accelerates convergence by allowing the model to make larger updates to the weights per iteration, with the risk of overshooting.

Hidden Layers

The models with different number of hidden layers, one hidden layer and two hidden layers, show different number of epochs required to converge, from 4714 for single hidden layer and 656 for two hidden layers, with same performance metrics. This suggests that an additional hidden layer may aid in faster learning but does not necessarily improve the model's performance.

Optimizers

Both momentum and ADAM optimizers perform well with all same performance metrics, but the Adam optimizer shows better efficiency in convergence even with small learning rate of 0.001, requiring only 430 epochs compared to the 4714 epochs needed with Momentum with the learning rate of 0.01.

Activation Functions

Switching from ReLU to Leaky ReLU in a single hidden layer model with Momentum optimizer and a learning rate of 0.01 shows reduction in training epochs from 4714 to 2678. This suggests that Leaky ReLU, which allows a small gradient when the unit is inactive and thus avoids dead neurons, might facilitate faster learning in some scenarios.

Hidden Layer	Learning Rate	Optimizer	Activation Function	Number of Epoch	Accuracy	Precision	Recall	F1
1	0.01	Momentum	ReLu	4714	0.9985	0.9952	0.9976	0.9964
1	0.05	Momentum	ReLu	61	0.9985	0.9952	0.9976	0.9964
2	0.01	Momentum	ReLu	656	0.9985	0.9952	0.9976	0.9964
1	0.01	Momentum	Leaky ReLu	2678	0.9985	0.9952	0.9976	0.9964
1	0.001	Adam	ReLu	430	0.9985	0.9952	0.9976	0.9964

Table 2: Model Comparison

Conclusion

The Feed-Forward neural network models demonstrated consistent performance across all configurations, including learning rates, optimizer, number of hidden layers, and activation function types. Each configuration achieved uniformly high scores in accuracy, precision, recall, and F1 measures, indicating robust and reliable performance. However, the different choice of configuration significantly influenced the convergence speed. This suggests that the main challenge lies in optimizing the training process rather than enhancing accuracy. Therefore, it would be beneficial to prioritize developing more efficient training strategies, considering the consistent model performance across all different settings.