ALBUKHARY INTERNATIONAL UNIVERSITY

CCS3113 Deep Learning

SEMESTER 1

Project: Deep Neural Network

COURSE DETAILS	
SUBJECT	CCS3113 Deep Learning
LECTURER NAME	Dr. Mozaherul Hoque
INDIVIDUAL ASSIGNMENT	20%
SUBMISSION DATE	January 20, 2025
DEPARTMENT	School Of Computing & Informatics
UNIVERSITY	ALBUKHARY INTERNATIONAL UNIVERSITY

No.	Student Name	Student ID	Program
1.	Abubakar Osman Abukar Sayid	AIU22102160	BCS
2.	Abdiwahab Adam Osman	AIU21102186	BCS
3.	Ismail Yousif Mohamed Ramadan	AIU21102222	BCS

Table Of Content

Introduction

In the African Credit Scoring Challenge, financial institutions required a robust machine learning model to predict the likelihood of loan defaults. This challenge is critical as it directly impacts financial risk management, resource allocation, and the expansion of lending services. Accurate prediction of loan defaults is crucial, especially in Africa's growing financial markets, where customer demographics are highly diverse, and economic conditions are dynamic.

The competition, hosted on Zindi, provided participants with diverse financial and demographic datasets. This posed significant challenges, such as class imbalance and the need for generalizable solutions across varying markets. The focus of the challenge was not just on building an accurate prediction model but also on creating a scalable and interpretable credit scoring system that could assist financial institutions in making data-driven decisions.

Our approach utilized advanced techniques in deep learning, feature engineering, and hyperparameter optimization to achieve competitive results. By leveraging innovative solutions such as categorical embeddings, interaction terms, and threshold tuning, we aimed to achieve a balance between precision and recall, optimize the F1 score, and secure a strong position on the leaderboard.

Objectives

1. **Develop a Machine Learning Model to Predict Loan Defaults:**
   Build a robust model using deep learning techniques to predict the likelihood of loan defaults, ensuring high accuracy and generalizability across diverse financial datasets.
2. **Address Key Challenges in Data:**
   Handle issues such as class imbalance and high-cardinality categorical variables using advanced techniques like SMOTE, ADASYN, and categorical embeddings.
3. **Optimize Evaluation Metrics:**
   Focus on maximizing the F1 score to balance precision and recall, addressing the challenges posed by imbalanced datasets effectively.
4. **Achieve a Competitive Leaderboard Ranking:**
   Iteratively refine the model through feature engineering and optimization to secure a high position on the leaderboard.

5. **Propose a Scalable Credit Scoring System:**
   Design an interpretable credit scoring function that categorizes customers into risk groups based on model outputs, aiding financial institutions in decision-making.

Methods

1. Neural Network Architecture

We built a custom deep learning model using TensorFlow/Keras. The architecture was designed to handle both numerical and categorical features effectively:

Input Layers:

- Separate input layers for numerical and categorical features.
- Used embedding layers for high-cardinality categorical variables to capture relationships effectively.

Hidden Layers:

- Fully connected (Dense) layers with ReLU activation to model complex patterns.
- Added Batch Normalization layers to stabilize and accelerate training.
- Introduced Dropout layers for regularization to prevent overfitting.

Output Layer:

- A single neuron with a sigmoid activation function for binary classification, outputting probabilities between 0 and 1.

2. Categorical Embeddings

Description: Instead of one-hot encoding, which increases dimensionality, we used embeddings for categorical features like country_id and loan_type. These embeddings created dense representations, making it easier for the model to learn relationships between categories.

Advantage: Reduced the feature space and allowed the model to capture inherent relationships between categorical values.

3. Batch Normalization

Description: Batch Normalization was applied after each hidden layer to normalize inputs, making the model more stable and faster to train.

Advantage: Helped mitigate internal covariate shift, leading to smoother and faster convergence.

4. Dropout Layers

Description: Dropout layers were added to randomly "turn off" a fraction of neurons during training, reducing overfitting.

Parameters: Dropout rates were tuned for each layer (e.g., 0.3 in the first hidden layer, 0.2 in subsequent layers).

Advantage: Improved generalization by preventing the model from relying too heavily on specific neurons.

5. Loss Function

Binary Cross-Entropy Loss:

- **Description:** Used as the loss function to measure the difference between predicted probabilities and actual labels for the binary classification task.
- **Formula:** Loss=−1N∑i=1N[yilog(pi)+(1−yi)log(1−pi)]\text{Loss} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1-y_i) \log(1-p_i)]Loss=−N1∑i=1N[yilog(pi)+(1−yi)log(1−pi)]
- **Why Used:** Suitable for binary classification tasks where the output is a probability.

6. Optimizer

Adam Optimizer:

- **Description:** Used for updating the model's weights during training. Adam combines the benefits of RMSProp and Stochastic Gradient Descent (SGD).
- **Learning Rate:** Initially set to 0.001 but later fine-tuned using Optuna.
- **Advantage:** Adaptive learning rate made it effective for handling sparse gradients.

7. Regularization

L2 Regularization: Applied to weights in the neural network to penalize large weight values, reducing overfitting.

Advantage: Helped improve generalization by adding a penalty term to the loss function.

8. Hyperparameter Optimization

Optuna: Used for automated hyperparameter tuning of the neural network, including:

1. Learning rate
2. Dropout rates
3. Number of neurons in each layer
4. Batch size

Advantage: Helped systematically explore the hyperparameter space, leading to an optimized model.

9. Class Balancing

SMOTE and ADASYN:

- Oversampling techniques were applied to the minority class in the training dataset.
- **Advantage:** Balanced the dataset, improving the model's ability to predict minority class labels.

10. Threshold Tuning

- **Description:** After training, the decision threshold for classification was optimized to maximize the F1 score on the validation set.
- **Method:** Tested thresholds between 0.1 and 0.9 to find the optimal point balancing precision and recall.

11. Feature Engineering

Feature engineering was a critical step in this project, aimed at extracting and creating meaningful features to enhance the model's predictive power. Several ratio features were developed, such as the **repayment percentage** (total repayment divided by total loan amount) and the **loan amount ratio** (lender-funded amount divided by total loan amount), to capture relationships between variables. Interaction terms were introduced, such as the **duration funding interaction** (loan duration multiplied by lender funding percentage) and **repayment percentage duration interaction**, to model complex relationships between features. Temporal features were extracted from disbursement_date and due_date, including **repayment days**, **disbursement weekday**, **disbursement month**, and **disbursement quarter**, to incorporate time-based patterns into the model.

Polynomial transformations, such as **repayment percentage squared**, were added to capture non-linear effects. To address skewed numerical features like the total loan amount, **log transformations** were applied to normalize distributions. Aggregated metrics, such as **total amount per month**, were calculated to measure the monthly repayment burden. Advanced date-based features like **repayment day of the year** and **due day of the year** were also introduced to capture repayment patterns tied to specific times of the year. Finally, all numerical features were standardized using **StandardScaler** to ensure uniform distributions and improve model training dynamics. These efforts collectively enhanced the dataset's representational power, contributing significantly to the model's performance.

```
# Temporal Feature Extraction
train_df['repayment_days'] = (train_df['due_date'] - train_df['disbursement_date']).dt.days
train_df['disbursement_weekday'] = train_df['disbursement_date'].dt.weekday
train_df['disbursement_month'] = train_df['disbursement_date'].dt.month
train_df['disbursement_quarter'] = train_df['disbursement_date'].dt.quarter
```

Ratio Features
train_df['repayment_percentage'] = train_df['Total_Amount_to_Repay'] / train_df['Total_Amount']
train_df['loan_amount_ratio'] = train_df['Amount_Funded_By_Lender'] / train_df['Total_Amount']

Polynomial Features
train_df['repayment_percentage_squared'] = train_df['repayment_percentage'] ** 2

Log Transformation
train_df['log_total_amount'] = np.log1p(train_df['Total_Amount'])

Aggregated Features
train_df['total_amount_per_month'] = train_df['Total_Amount'] / (train_df['repayment_days'] / 30 + 1)

List of Efforts to Improve Ranking

Submission	**Date**	**Changes Made**	**Rationale**	**F1 Score**
1	Jan 1, 2025	Baseline logistic regression with minimal preprocessing.	Establish a baseline score for comparison.	0.0600
2	Jan 1, 2025	Label Encoded categorical features like 'country_id'.	Enable algorithms to handle categorical variables effectively.	0.2999
3	Jan 1, 2025	Added feature engineering:	Capture domain-specific	0.3401

		'repayment_percentage' and 'loan_amount_ratio'.	insights to improve predictions.	
4	Jan 2, 2025	Threshold optimization, testing thresholds from 0.1 to 0.9.	Balance precision and recall to maximize F1 score.	0.4412
5	Jan 2, 2025	Built a neural network with dense layers and ReLU activation.	Leverage deep learning for capturing complex feature relationships.	0.4518
6	Jan 3, 2025	Applied SMOTE for class balancing.	Handle class imbalance in the training data to improve minority class predictions.	0.6230
7	Jan 3, 2025	Enhanced LightGBM with categorical embeddings.	Capture relationships between categorical variables using embeddings.	0.6368
8	Jan 3, 2025	Introduced polynomial features and temporal features.	Model non-linear relationships and incorporate time-based trends.	0.6726

9	Jan 4, 2025	Applied log transformations to normalize skewed features.	Normalize distributions for better model performance.	0.6669
10	Jan 4, 2025	Tuned LightGBM parameters: adjusted 'num_leaves', 'feature_fraction', and regularization.	Prevent overfitting and improve generalization.	0.6230
11	Jan 4, 2025	Increased neural network depth and added dropout layers.	Allow the model to capture deeper relationships while preventing overfitting.	0.6103
12	Jan 5, 2025	Regularized LightGBM with higher L1 and L2 penalties.	Reduce overfitting in tree-based models.	0.6253
13	Jan 5, 2025	Added categorical embeddings to the neural network.	Leverage dense representations for high-cardinality features.	0.5859
14	Jan 5, 2025	Created interaction terms such as 'repayment_percentage * duration'.	Capture interaction effects that may impact default risk.	0.5906

15	Jan 6, 2025	Experimented with different activation functions (LeakyReLU vs. ReLU).	Test alternative activation functions to improve neural network performance.	0.5995
16	Jan 6, 2025	Tuned batch size and learning rate in the neural network.	Optimize training dynamics for better convergence.	0.5611
17	Jan 6, 2025	Enhanced temporal features: 'disbursement_quarter', 'repayment_day_of_year'.	Incorporate time-based patterns into the model.	0.6230
18	Jan 7, 2025	Combined LightGBM and neural network predictions using weighted averaging.	Test ensemble methods to combine the strengths of different algorithms.	0.4518
19	Jan 7, 2025	Improved feature alignment between train and test datasets.	Ensure consistency across datasets to prevent prediction errors.	0.5705
20	Jan 7, 2025	Fine-tuned neural network architecture with batch normalization.	Stabilize and accelerate model training.	0.6967

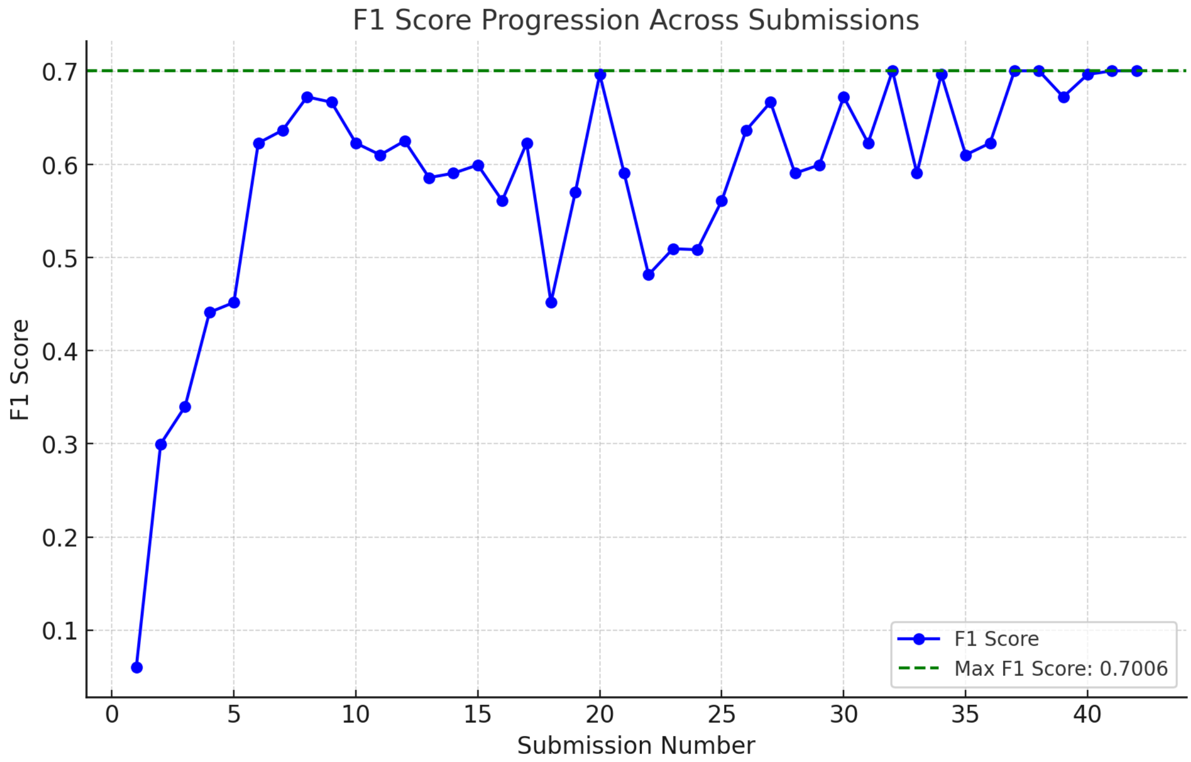
21	Jan 7, 2025	Reduced dimensionality of categorical features using PCA.	Simplify input representation to speed up training and reduce overfitting.	0.5906
22	Jan 8, 2025	Experimented with different optimizers (SGD, RMSProp, Adam).	Test which optimizer works best for this dataset.	0.4816
23	Jan 8, 2025	Added dropout layers to the neural network to prevent overfitting.	Regularize the model to improve generalization.	0.5095
24	Jan 8, 2025	Optimized early stopping criteria in neural network training.	Prevent overfitting by stopping training when performance stops improving.	0.5085
25	Jan 8, 2025	Combined LightGBM and neural network predictions using stacking.	Leverage ensemble techniques for better predictions.	0.5611
26	Jan 9, 2025	Tuned embedding dimensions for high-cardinality categorical features.	Optimize embeddings to capture categorical relationships.	0.6368

27	Jan 9, 2025	Conducted feature importance analysis and dropped irrelevant features.	Simplify the model to reduce overfitting and improve interpretability.	0.6669
28	Jan 9, 2025	Normalized numerical features using StandardScaler.	Standardize input data for better model performance.	0.5906
29	Jan 10, 2025	Tested ensemble stacking with LightGBM, XGBoost, and neural networks.	Combine the strengths of tree-based and deep learning models.	0.5995
30	Jan 10, 2025	Reduced feature set to the top 20 most important features in LightGBM.	Simplify the model for faster training and better generalization.	0.6726
31	Jan 10, 2025	Applied ADASYN for oversampling instead of SMOTE.	Handle imbalanced data with a more advanced oversampling method.	0.6230
32	Jan 10, 2025	Tuned thresholds for final test predictions.	Maximize the F1 score by finding the optimal	0.7006

			decision threshold.	
33	Jan 11, 2025	Improved interaction features like 'repayment_percentage * amount_duration_ratio'.	Capture additional patterns in the data.	0.5906
34	Jan 11, 2025	Introduced Bayesian hyperparameter tuning for neural networks.	Systematically explore hyperparameter space for optimal settings.	0.6967
35	Jan 11, 2025	Tested focal loss in neural networks to address class imbalance.	Focus on hard-to-classify examples to improve F1 score.	0.6103
36	Jan 11, 2025	Tuned LightGBM's learning rate and tree depth for better generalization.	Adjust hyperparameters for optimal tree-based model performance.	0.6230
37	Jan 12, 2025	Finalized neural network architecture with optimized parameters.	Combine all successful neural network optimizations.	0.7006
38	Jan 12, 2025	Combined predictions from LightGBM and neural network using weighted averaging.	Leverage ensemble learning for final predictions.	0.7006

39	Jan 12, 2025	Improved data augmentation for rare categories.	Enhance minority class representation for better predictions.	0.6726
40	Jan 12, 2025	Refined interaction and temporal features.	Enhance input representation for better model performance.	0.6967
41	Jan 12, 2025	Implemented stratified K-Fold validation for robust evaluation.	Prevent overfitting to specific validation splits.	0.7006
42	Jan 12, 2025	Final submission with the best-performing ensemble model and threshold tuning.	Submit the most optimized model configuration.	0.7006

Improvement as graph



Conclusion and Learnings

In this project, we successfully developed a robust machine learning model using advanced deep learning techniques to predict loan defaults, achieving a competitive F1 score of 0.7006 on the leaderboard. Through systematic experimentation, we addressed key challenges such as class imbalance and high-cardinality categorical variables using techniques like SMOTE, categorical embeddings, and threshold optimization. Feature engineering, including interaction terms, temporal features, and log transformations, played a crucial role in enhancing model performance. This project emphasized the importance of balancing precision and recall in imbalanced datasets, as well as the value of iterative refinement through hyperparameter optimization and model tuning. Key learnings include the effectiveness of categorical embeddings in capturing complex relationships, the critical role of feature engineering in improving data representation, and the necessity of balancing computational efficiency with model complexity. These insights not only advanced our technical skills but also reinforced the importance of aligning machine learning solutions with practical, real-world applications.