Deep Learning Assignment: Neural Networks for Tabular Data Classification

Overview

In this assignment, you will build a deep neural network to predict customer churn based on tabular data. Customer churn prediction is a critical business problem where companies try to identify customers likely to discontinue their service, allowing for targeted retention strategies.

Learning Objectives

By completing this assignment, you will:

- Gain practical experience preprocessing tabular data for deep learning
- Design and implement deep neural network architectures appropriate for structured data
- Apply techniques to handle class imbalance in classification problems
- Implement regularization and optimization strategies for neural networks
- Evaluate model performance using appropriate metrics for imbalanced classification
- Compare the performance of deep learning against traditional machine learning approaches

Dataset

You will work with the Bank Customer Churn Prediction dataset. The dataset contains information about bank customers and whether they left the bank (churned) or remained customers.

Dataset Description

- **Source**: Bank Customer Churn Dataset
- **Features**:
  - CustomerID: Unique identifier for the customer
  - Surname: Customer's surname
  - CreditScore: Credit score of the customer

- Geography: Customer's location (country)
- Gender: Customer's gender
- Age: Customer's age
- Tenure: Number of years the customer has been with the bank
- Balance: Account balance
- NumOfProducts: Number of bank products the customer uses
- HasCrCard: Whether the customer has a credit card
- IsActiveMember: Whether the customer is an active member
- EstimatedSalary: Estimated salary of the customer
- Exited: Whether the customer left the bank (1) or not (0) - this is our target variable

Assignment Tasks

Part 1: Data Preprocessing and Exploration (25 points)

1. Load and inspect the dataset
2. Perform exploratory data analysis (EDA) to understand the distribution of features and target
3. Handle any missing values if present
4. Preprocess categorical features using appropriate encoding techniques
5. Normalize/standardize numerical features
6. Split the data into training, validation, and test sets (60%/20%/20%)
7. Analyze and address the imbalance issue
8. Apply feature engineering to create new features

Part 2: Baseline Model (15 points)

1. Implement a baseline machine learning model (e.g., Random Forest or Logistic Regression)
2. Train the model on the preprocessed data

3. Evaluate the baseline model using appropriate metrics (accuracy, precision, recall, F1-score, ROC-AUC)
4. Analyze the feature importance of the baseline model

Part 3: Neural Network Design and Implementation (30 points)

1. Design a deep neural network architecture suitable for tabular data classification
    - Implement at least two hidden layers
    - Experiment with different activation functions
    - Implement dropout for regularization
    - Consider batch normalization layers

2. Implement the model using TensorFlow or PyTorch
3. Train the model using an appropriate optimization algorithm
4. Implement techniques to handle the class imbalance:
    - Class weighting
    - Sampling techniques (oversampling/undersampling)
    - Custom loss functions

5. Implement callbacks for early stopping and learning rate scheduling
6. Visualize the training and validation loss/accuracy curves

Part 4: Model Evaluation and Hyperparameter Tuning (30 points)

1. Evaluate the neural network on the validation set
2. Perform hyperparameter tuning using keras tuner if you are using tensorflow or Ray tune if you are using PyTorch:
    - Learning rate
    - Batch size
    - Number of neurons in each layer

    - Dropout rate
    - Regularization strength

3. Evaluate the final and the best model on the test set
4. Compare the performance with the baseline model

Deliverables

1. Python code (Jupyter notebook or Python scripts) with clear comments
2. In your notebook:
    - Description of the preprocessing steps
    - Details of the neural network architecture and design choices
    - Analysis of the results, including comparison with the baseline
    - Discussion of challenges faced, and solutions implemented
    - Recommendations for further improvements

Evaluation Criteria

- Correctness and completeness of the implementation (40%)
- Quality of the neural network design and implementation (30%)
- Depth of analysis and interpretation of results (20%)
- Code quality and documentation (10%)

Resources

- TensorFlow documentation: https://www.tensorflow.org/api_docs
- PyTorch documentation: https://pytorch.org/docs/stable/index.html
- Scikit-learn documentation: https://scikit-learn.org/stable/
- Imbalanced-learn documentation: https://imbalanced-learn.org/stable/
- Feature engineering techniques: https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

Submission Instructions

Submit your code as a IPYNB file through LMS by 4/25/2025.

Notes

- You are encouraged to use GPU acceleration if available
- You can implement your model either in tensorflow or pytorch.
- You may use any Python libraries relevant to the task
- Consult with the instructor if you encounter significant computational limitations