Bank Marketing Campaign Analysis Using Deep Learning Models

Group 3 Project Team

Introduction

This report presents an analysis of a bank marketing campaign dataset using various deep learning models. The project aims to predict whether a client will subscribe to a term deposit (binary outcome: 'yes' or 'no') based on client information and campaign details. Understanding which customers are likely to subscribe to bank products is crucial for financial institutions to optimize their marketing strategies, reduce costs, and increase conversion rates. This analysis explores the effectiveness of different neural network architectures in addressing this classification problem, which is characterized by significant class imbalance.

Dataset Description and Preprocessing

Dataset Overview

The dataset contains information about direct marketing campaigns conducted by a Portuguese banking institution. It includes:

- 29,271 records in the training set and 11,917 records in the test set
- 15 features including demographic information (age, job, marital status, education), financial indicators (housing loan, personal loan), and campaign-related information (contact type, month, duration)
- A highly imbalanced target variable "Subscribed" with approximately 89% negative cases (no subscription) and 11% positive cases (subscription)

Preprocessing Steps

1. **Handling Missing Values**:
    - 'Unknown' values in categorical features were replaced with the mode (most frequent value) of each column
    - 'pdays=999' (indicating clients not previously contacted) was converted to -1 to better represent this special case
2. **Categorical Feature Encoding**:
    - All categorical variables were transformed using LabelEncoder to convert them into numerical format
    - The target variable 'Subscribed' was also encoded (no=0, yes=1)
3. **Feature Scaling**:

   - Numerical features were standardized using StandardScaler to ensure all features contribute equally to model training
4. **Class Imbalance Handling**:
   - Class weights were computed and applied during model training to address the significant imbalance between subscription classes
   - This approach penalizes misclassification of the minority class more heavily
5. **Data Reshaping**:
   - For CNN and RNN models, the data was reshaped to include an additional dimension required by these architectures

Model Descriptions

Four different neural network architectures were implemented and compared:

1. Multilayer Perceptron (MLP)

- `Dense layer (64 neurons, ReLU activation)`
- `Dropout (30%)`
- `Dense layer (32 neurons, ReLU activation)`
- `Dropout (20%)`
- `Output layer (1 neuron, sigmoid activation)`

The MLP model uses a simple feedforward architecture with dropout layers to prevent overfitting.

2. Convolutional Neural Network (CNN)

- `Conv1D layer (64 filters, kernel size 3, ReLU activation)`
- `MaxPooling1D (pool size 2)`
- `Flatten layer`
- `Dense layer (32 neurons, ReLU activation)`
- `Dropout (20%)`
- `Output layer (1 neuron, sigmoid activation)`

The CNN model treats the features as a sequence and applies convolutional operations to extract patterns.

3. Long Short-Term Memory (LSTM)

- `LSTM layer (64 units, return sequences=True)`
- `Dropout (30%)`
- `LSTM layer (32 units)`
- `Dropout (20%)`
- `Output layer (1 neuron, sigmoid activation)`

The LSTM model leverages recurrent connections with memory cells to capture potential sequential patterns in the data.

4. Simple Recurrent Neural Network (RNN)

- `SimpleRNN layer (64 units)`
- `Dropout (30%)`
- `Dense layer (32 neurons, ReLU activation)`
- `Dropout (20%)`
- `Output layer (1 neuron, sigmoid activation)`

The RNN model uses a basic recurrent architecture to process the features as a sequence.

Training Configuration

All models were trained with:

- Binary cross-entropy loss function
- Adam optimizer with learning rate of 0.001
- Early stopping with patience of 5 epochs to prevent overfitting
- Class weights to address class imbalance
- 50 maximum epochs with batch size of 64
- 20% validation split

Results and Findings

Model Performance Comparison

Model	Accuracy	Precision	Recall	F1 Score
MLP	0.1385	0.1233	1.0000	0.2196
CNN	0.1384	0.1233	1.0000	0.2195
LSTM	0.1355	0.1229	1.0000	0.2190
RNN	0.1354	0.1229	1.0000	0.2189
Linear	0.1386	0.1233	1.0000	0.2196

Training and Validation Metrics

The training history plots revealed:

- All models showed signs of overfitting, with training accuracy continuing to improve while validation accuracy declined after initial epochs
- The LSTM model showed the most stable validation performance, though still with evidence of overfitting
- Early stopping typically triggered after 5-6 epochs for most models, indicating rapid overfitting

Confusion Matrices

The confusion matrices for all models showed similar patterns:

- Perfect recall (1.0) for the positive class (all actual subscriptions were correctly identified)
- Very poor precision (~0.12) indicating many false positives
- All models essentially predicted every instance as positive, resulting in high recall but low precision

Analysis and Discussion

Model Performance Analysis

1. **Similar Performance Across Models**: Despite architectural differences, all models performed nearly identically, suggesting that the problem may not benefit from complex architectures. The simple linear model performed as well as the more complex neural networks.

2. **Extreme Prediction Behavior**: All models essentially predicted every test instance as positive (subscription), resulting in:
    - Perfect recall (1.0) but very low precision (~0.12)
    - F1 scores around 0.22, which is poor for a binary classification task
    - Accuracy around 0.14, which is worse than a naive majority class predictor
3. **Class Imbalance Impact**: The class weights applied during training appear to have overcompensated for the imbalance, causing models to be overly biased toward the minority class.

4. **Feature Importance**: Correlation analysis revealed:
    - 'Duration' had the strongest positive correlation with subscription outcome
    - 'Previous outcome success' was a strong predictor of future subscription
    - Contact method and month of contact showed significant influence on subscription rates

Key Insights

1. **Data Characteristics Matter More Than Model Complexity**: The similar performance across different architectures suggests that the dataset's characteristics (class imbalance, feature distributions) have more impact than model architecture.

2. **Call Duration as a Key Predictor**: The strong correlation between call duration and subscription outcome suggests that longer customer interactions are associated with successful conversions.

3. **Previous Success Predicts Future Success**: Customers who subscribed in previous campaigns were much more likely to subscribe again, indicating the value of retargeting previous successful contacts.

4. **Seasonal Effects**: Subscription rates varied significantly by month, with certain months showing higher conversion rates.

Conclusion

This analysis of bank marketing campaign data using various deep learning models revealed several important findings:

1. **Model Selection**: For this particular classification problem, simpler models performed as well as complex ones. The linear model achieved the highest F1 score (0.2196), marginally outperforming the neural network architectures.

2. **Class Imbalance Challenges**: The extreme class imbalance in the dataset posed significant challenges for all models, resulting in poor overall performance despite attempts to address it through class weighting.

3. **Prediction Behavior**: All models exhibited the same behavior of predicting nearly all instances as positive, suggesting that the current approach to handling class imbalance needs refinement.

4. **Feature Insights**: The analysis identified several key predictors of subscription success, including call duration, previous campaign outcomes, and contact timing.

Future Improvements

1. **Alternative Sampling Techniques**: Explore techniques like SMOTE, ADASYN, or under-sampling to address class imbalance instead of relying solely on class weights.

2. **Feature Engineering**: Develop new features based on the identified important predictors, such as interaction terms or domain-specific transformations.

3. **Threshold Adjustment**: Implement probability threshold tuning to optimize the precision-recall tradeoff rather than using the default 0.5 threshold.

4. **Ensemble Methods**: Combine multiple models using ensemble techniques like stacking or boosting to improve overall performance.

5. **Hyperparameter Optimization**: Conduct more extensive hyperparameter tuning, particularly focusing on learning rate and regularization parameters.

References

1. TensorFlow and Keras documentation: https://www.tensorflow.org/api_docs/python/tf
2. Scikit-learn documentation: https://scikit-learn.org/stable/documentation.html
3. Moro, S., Cortez, P., & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, 62, 22-31.
4. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

Team Contributions

All five of the Group 3 team members contributed to the development, review, report, presentation and project delivery.

Jawwad Khalil Ahmed

Eric Efon

Daniel Mehta

Thomas Nash

Jeffrey Ng