Traditional vs Neural Approaches for Bank Marketing Prediction

Sam Phillips

Department of Computer Science,
University of North Carolina at Charlotte

ITSC 3156: Introduction to Machine Learning
Dr. Hongfei Xue

3 December 2025

1 Introduction

Problem Statement. This project addresses the task of predicting whether a banking client will subscribe to a term deposit during a marketing campaign. The problem is formulated as a binary classification task, where the target variable indicates whether the customer responded "yes." The input features include demographic information, economic indicators, and details about prior campaign contact, spanning both numerical and categorical attributes.

Motivation and Challenges. Accurately identifying which customers are likely to respond is important because targeted marketing reduces cost and improves campaign efficiency. However, the dataset presents several challenges: the predictors show weak and sometimes nonlinear relationships with the target, and the classes are heavily imbalanced, with far more "no" responses than "yes." These issues make naïve accuracy misleading and require models and evaluation metrics that handle imbalance and subtle signal patterns appropriately.

Summary of Approach. The analysis follows a complete machine learning pipeline, beginning with exploratory data analysis and preprocessing, including normalization and class-imbalance handling. I implement two types of models: traditional baseline algorithms (such as logistic regression) and a more flexible feed-forward neural network. A secondary motivation for this project was to gain hands-on experience using PyTorch, as I had not previously built a neural network framework from scratch. Model performance is evaluated using precision, recall, and F1-score to provide a fair comparison under class imbalance.

2 Data

2.1 Dataset Description

The dataset used in this project is the Bank Marketing dataset from the UCI Machine Learning Repository. It contains information on 41,188 customers contacted as part of a direct marketing campaign for term deposit subscriptions. The dataset includes 16 input features and one binary target variable indicating whether the client subscribed to a term deposit. The features span demographic attributes (such as age and job), financial indicators (such as account balance and loan status), and campaign interaction details (such as contact method, month, and number of prior contacts). The majority of features are categorical, with a smaller set of numerical fields.

A summary of the dataset structure is shown in Table 1.

Table 1: Overview of the Bank Marketing Dataset

Feature Type	Count	Examples
Numerical	7	age, balance, duration, campaign
Categorical	9	job, marital, education, housing, loan, contact
Target	1	y (yes/no)

2.2 Exploratory Data Analysis

Target Distribution. A bar plot of the target classes reveals a significant class imbalance: the vast majority of customers did not subscribe to a term deposit. Only a small minority responded "yes," making accuracy an unreliable metric for evaluation and motivating the use of precision, recall, and F1-score later in the analysis.

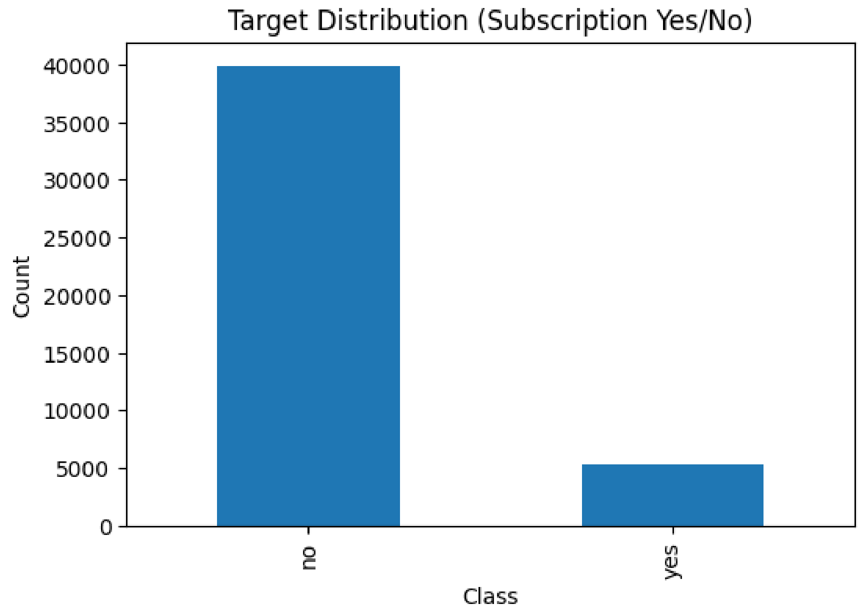
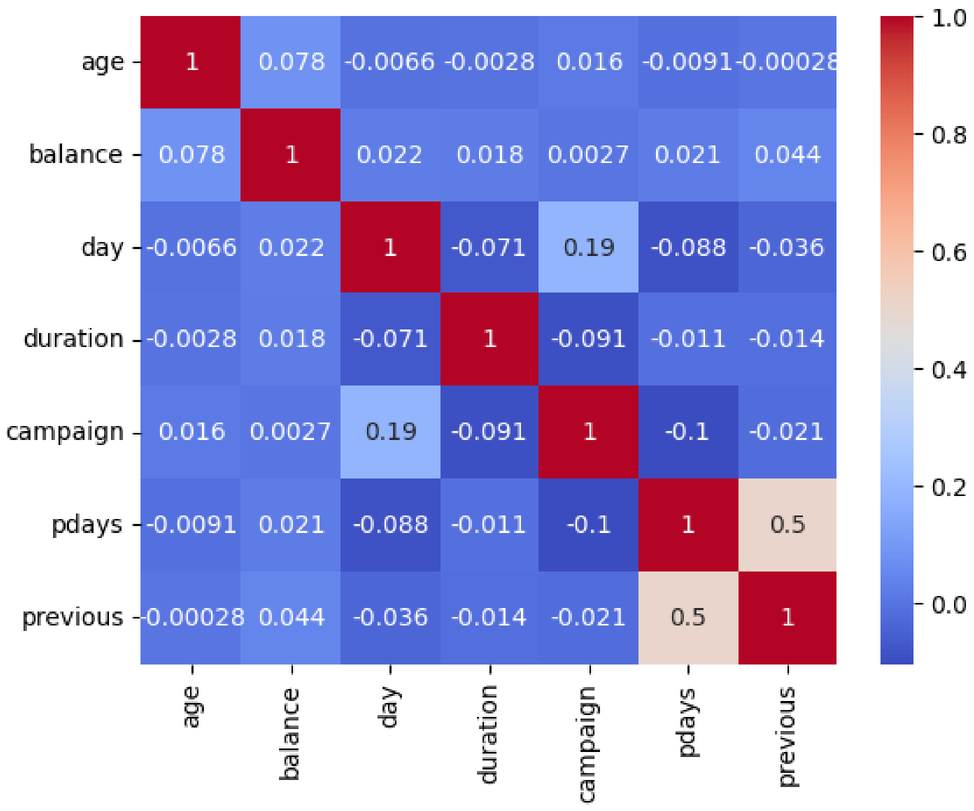


Figure 1: Distribution of the target variable showing substantial class imbalance.

I noted at this time that balancing techniques might be necessary in order to properly approximate the function.

Numerical Feature Distributions. Histograms for the numerical features (such as age, campaign, and balance) show strong right-skewed behavior. For example, most clients were contacted only once during the campaign, but a small number experienced many repeated contacts. Account balances also exhibit a long-tailed distribution. These characteristics highlight the presence of outliers and non-uniform spreads that models must account for. Another issue that exploration revealed is the significant lack of a linear relationship, see Figure 2 below. Due to the nonlinear nature of these relationships, more flexible models



	age	balance	day	duration	campaign	pdays	previous
age	1	0.078	-0.0066	-0.0028	0.016	-0.0091	-0.00028
balance	0.078	1	0.022	0.018	0.0027	0.021	0.044
day	-0.0066	0.022	1	-0.071	0.19	-0.088	-0.036
duration	-0.0028	0.018	-0.071	1	-0.091	-0.011	-0.014
campaign	0.016	0.0027	0.19	-0.091	1	-0.1	-0.021
pdays	-0.0091	0.021	-0.088	-0.011	-0.1	1	0.5
previous	-0.00028	0.044	-0.036	-0.014	-0.021	0.5	1

Figure 2: Correlation heatmap of numerical features. Most correlations are near zero, indicating weak linear relationships.

are required to capture meaningful patterns that linear methods may miss. In addition, the feature 'duration' was noted to be correlated with the target variable, however, since a call's duration wouldn't be known until after a client was contacted, I flagged it for feature leakage.

Categorical Variables. The dataset contains several categorical features such as job, marital status, education, contact type, and loan indicators. These variables exhibit highly uneven category frequencies: for example, occupations such as “management,” “technician,” and “blue-collar” appear far more often than categories like “student” or “unknown.” Similarly, most clients are married, and the majority are contacted through cellular channels. These imbalances introduce sparsity in certain categories, which meant that, appropriate preprocessing—such as one-hot encoding and later normalization—is essential to prevent dominant categories from disproportionately influencing the model. In addition, it was noted that none of the features besides education had a natural hiearchy, so it was the only one label-encoded.

2.3 Preprocessing

After completing the initial exploratory analysis, several preprocessing steps were applied to prepare the dataset for modeling. Numerical and categorical features were first separated to allow for appropriate transformations. Categorical variables were primarily one-hot encoded to avoid imposing artificial ordinal structure on non-ordered categories. An exception was the `education` feature, which was label-encoded instead of one-hot encoded because it possesses a meaningful hierarchy (primary ¡ secondary ¡ tertiary). The small number of samples labeled as “unknown” in this feature were removed to preserve the integrity of this ordering.

The numerical features were normalized to give them comparable scales, which is important for distance-based models and neural networks. During this stage, the `duration` variable was also removed. Although highly predictive, it represents the length of the last phone call and is only known after contacting the client; thus as mentioned previously including it would therefore introduce unrealistic information leakage into the model.

A central challenge in preprocessing was the substantial class imbalance. Only a small fraction of customers subscribed to a term deposit, and the majority-class dominance significantly affects many learning algorithms. To address this, a balanced version of the training set was created using random undersampling of the majority class. This allowed for controlled experiments comparing unbalanced and balanced training regimes and was essential for evaluating how different models respond to the limited signal available in the minority class.

3 Methods

This project evaluates two baseline machine learning models—Logistic Regression and Random Forest—followed by a more flexible feed-forward neural network implemented in Py-

Torch. These models were chosen to provide a progression from simple linear structure to models capable of capturing nonlinear interactions that were suggested by the exploratory analysis.

3.1 Logistic Regression

Logistic Regression is a linear classifier that models the log-odds of the positive class as a weighted sum of the input features. It serves as a natural baseline because it is interpretable, computationally efficient, and effective when the underlying relationships are approximately linear. In this dataset, no strong multicollinearity was present, making Logistic Regression a suitable starting point even though the weak, nonlinear signal limits its performance.

Key hyperparameters included the regularization strength C, L2 regularization, and the `class_weight` parameter to mitigate the strong imbalance between "yes" and "no" responses. Minimal tuning was performed beyond selecting reasonable defaults, as the goal was primarily to establish a meaningful linear reference model.

3.2 Random Forest

A Random Forest is an ensemble of decision trees trained on bootstrapped samples, with each split considering only a random subset of features. This structure allows the model to capture nonlinear interactions and feature hierarchies that linear models cannot represent. It provides a stronger baseline for this dataset, where individual features have weak marginal predictive power but can contribute meaningful signal when combined.

The primary hyperparameters were the number of trees, the maximum depth of each tree, and the minimum number of samples required to split. Only light tuning was applied, with parameter choices guided by standard defaults rather than exhaustive search.

3.3 Neural Network (PyTorch)

The final model is a feed-forward neural network composed of an input layer, two hidden layers with ReLU activations, and a sigmoid output neuron for binary classification. Neural networks are well-suited for learning distributed, nonlinear patterns, making them an appropriate choice given the weak linear correlations observed in the exploratory analysis.

The main hyperparameters were the batch size, and number of epochs. Training used the Adam optimizer and binary cross-entropy loss. As with the tree-based model, tuning was kept minimal, with hyperparameter values selected to provide stable learning rather than to optimize performance aggressively. Number of epochs, for instance, was increased to 48, but afterwards showed diminishing returns. This model also served as an opportunity to gain practical familiarity with PyTorch.

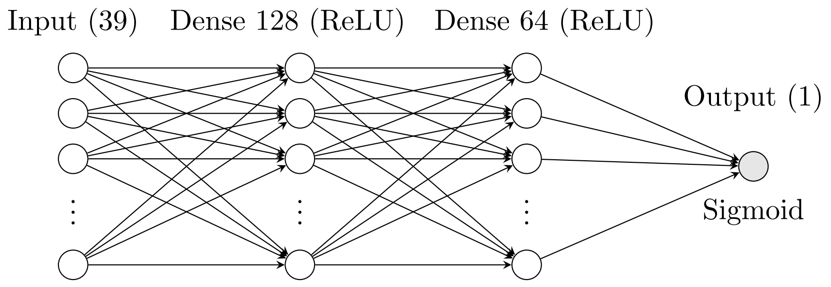


Figure 3: Neural network architecture with 39 input features, two hidden layers (128 and 64 neurons), and a single sigmoid-activated output neuron.

A 2 layer neural network was chosen due to the tabular nature of the data. A deeper architecture was unnecessary, as the dataset does not exhibit the high-dimensional structure typically requiring large or complex networks.

4 Results

4.1 Experimental Setup

The dataset was split into an 80% training set and a 20% test set. All models were trained on the preprocessed feature matrix described earlier, and evaluation was performed exclusively on the held-out test set. Because of the strong class imbalance, accuracy alone is not informative; therefore precision, recall, and F1-score were used as the primary evaluation metrics, with a particular emphasis on the minority “yes” class. No cross-validation was applied, as the goal was to compare the relative behavior of the three models under identical conditions rather than to optimize absolute performance. To better understand the impact of imbalance on model behavior, each model was trained and evaluated twice: once on the original unbalanced dataset and once on a version rebalanced through random undersampling.

All experiments were run using Python, PyTorch, and scikit-learn, with the neural network trained on an NVIDIA GPU through CUDA acceleration. As mnetioedn above, the model used the Adam optimizer and binary cross-entropy loss with logits, applying the sigmoid transformation only during evaluation when converting raw outputs to probabilities.

4.2 Quantitative Results

Table 2 summarizes the test-set results for all models under both the original unbalanced data and a version rebalanced through undersampling. Each model captures the dominant "no" class effectively, but differs significantly in its ability to recover the minority "yes" class.

Table 2: Performance Comparison of Models on Test Set

Model	Accuracy	Precision (1)	Recall (1)	F1 (1)
Logistic Regression (Unbalanced)	0.89	0.63	0.18	0.27
Logistic Regression (Balanced)	0.89	0.63	0.18	0.27
Random Forest (Unbalanced)	0.89	0.60	0.21	0.31
Random Forest (Balanced)	0.89	0.62	0.22	0.32
Neural Network (Unbalanced)	0.88	0.46	0.31	0.37
Neural Network (Balanced)	0.79	0.32	0.74	0.44

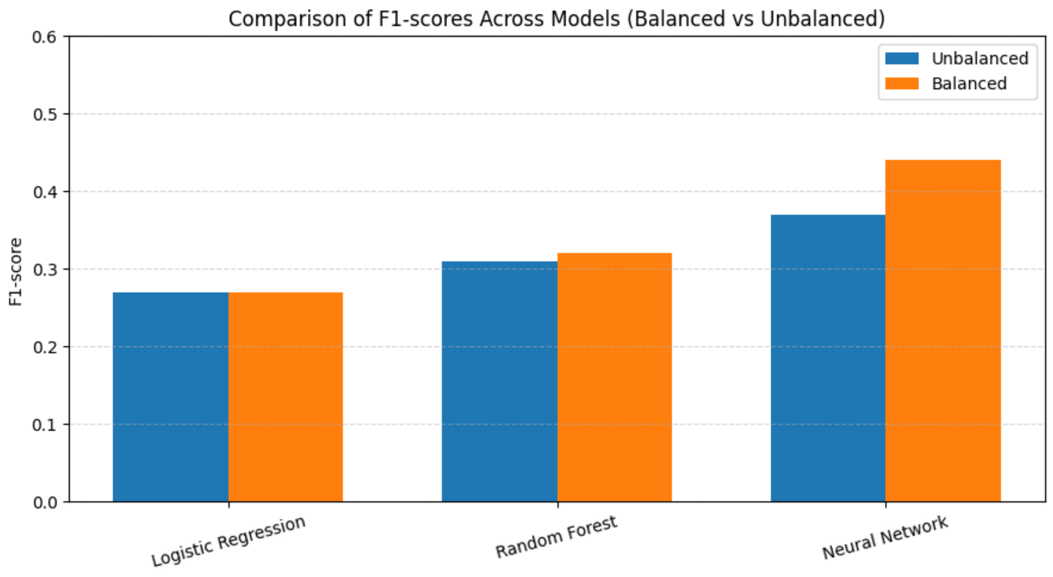


Figure 4: F1-score comparison across Logistic Regression, Random Forest, and Neural Network models under both unbalanced and balanced training conditions. Balancing significantly improves the neural network while having minimal effect on the linear and tree-based models.

4.3 Observations and Analysis

Several important trends emerge from the results. First, all models achieve broadly similar overall accuracy (approximately 88–89%) on the unbalanced dataset, but this metric conceals poor performance on the minority class. Logistic Regression achieves the highest precision among the linear methods but suffers from extremely low recall, identifying only 18% of positive samples. This aligns with expectations for a linear model applied to a dataset where the predictive signal is weak, nonlinear, and distributed across many features.

The Random Forest provides a slight improvement over Logistic Regression, particularly in recall and F1-score, reflecting its ability to capture limited nonlinear interactions. However, even the tree ensemble remains dominated by the majority class and struggles to identify positive examples reliably under the unbalanced distribution.

The Neural Network exhibits the most distinctive behavior. Under the unbalanced dataset, it achieves a substantially higher recall (31%) and the highest unbalanced F1-score among the models. More importantly, when trained on a balanced dataset, the Neural Network is the only model whose recall exceeds 70%, demonstrating that its nonlinear layers can meaningfully learn the minority-class decision boundary when given sufficient samples. This comes at the cost of reduced overall accuracy, but the trade-off reflects improved sensitivity to the positive class rather than degraded performance.

These results confirm the conclusions drawn from the exploratory analysis: the dataset contains no strong linear separators, and successful classification depends on combining many weak signals. Linear models struggle under these conditions, tree-based methods provide moderate improvements, and neural networks—though sensitive to class imbalance—are best able to exploit the subtle nonlinear structure when rebalanced appropriately. and show how the metrics move.

5 Conclusion

This project explored the problem of predicting term deposit subscriptions in the Bank Marketing dataset using logistic regression, random forest, and a PyTorch-based neural network. The analysis showed that the dataset contains weak individual predictors, low linear correlations, and a substantial class imbalance, all of which make the classification task difficult. Among the three models, the neural network achieved the highest F1-score, particularly when trained on a balanced dataset, indicating that its ability to learn nonlinear interactions offers an advantage over simpler models. In contrast, logistic regression and random forest showed only modest performance and minimal improvement under balancing.

These results suggest that more flexible models are better suited to datasets where predictive information is subtle and distributed across many features. However, performance remained limited overall due to the inherent structure of the data and the constraints of minimal hyperparameter tuning.

After completing the project, I also reviewed a paper showing that random forests can perform extremely well on this dataset. Seeing their approach helped me understand why their results were much stronger: they relied on a series of advanced preprocessing and feature-engineering steps that I would not have naturally devised at my current level. In contrast, the neural network provided a more intuitive and flexible modeling framework for me, and ultimately proved to be the most practical and effective method given my workflow and experience.

What I Learned

Through this project, I strengthened my understanding of the full machine learning pipeline—from exploratory analysis and preprocessing to model training and evaluation. Addressing class imbalance was a major challenge and highlighted how much metrics can shift depending on data distribution. Implementing the neural network in PyTorch was especially valuable, giving me practical experience with building models, training on GPU, and handling logits and activation functions.

The hardest part was figuring out how to downsample the dataset without damaging the underlying function present in the data. At first, I was far too aggressive. I believe the key to further optimization would be to review this - and use more advanced techniques to identify useless clusters. PCA on the numeric features revealed several dense groups, however, overall the features were not trivial.

Overall, this project reinforced the importance of data quality, careful preprocessing, and choosing models that match the structure of the problem.

6 References

"PyTorch Tutorials." *PyTorch*, https://pytorch.org/tutorials/. Accessed 6 Dec. 2025.

Moro, Sérgio, Paulo Cortez, and Paulo Rita. "Bank Marketing Data Set." *UCI Machine Learning Repository*, 2014, https://archive.ics.uci.edu/dataset/222/bank+marketing.

7 Acknowledgement

I used ChatGPT as a supplementary tool during the completion of this project. Its assistance included helping clarify concepts while I worked through the official PyTorch tutorials, providing example Matplotlib code for certain visualizations, and answering technical questions that arose during model development. ChatGPT also helped with drafting and refining the written report and with resolving various LaTeX formatting issues. All data preprocessing, model implementation, training decisions, and analytical interpretations were performed independently by me.

I additionally consulted the official PyTorch documentation for guidance on network architecture, training procedures, and CUDA usage. These materials supported my understanding but did not directly contribute code to the project. No other external tutorials, sample projects, or third-party implementations were used.

8 Source Code

The full source code for this project is publicly available at:

`https://github.com/SamKhan32/bank-marketing-ml`