BayesBank: A Bayesian Neural Network for Uncertainty-Aware Marketing-Campaign Response Prediction

Bianco Blanco
Department of Data Science
New Jersey Institute of Technology
bmb45@njit.edu

May 11, 2025

Abstract

Conventional classifiers for predicting customer response in direct-marketing campaigns treat model outputs as point estimates, making it impossible to flag predictions that the model itself deems uncertain. We present **BayesBank**, a Bayesian neural network (BNN) trained with Bayes-by-Backprop on the UCI *Bank Marketing* dataset (45,211 records, **5** categorical, **6** numeric, **3** binary, and **2** date features). BayesBank (i) attains competitive accuracy ($\approx$ 91%) and AUC ($\approx$ 0.923), (ii) produces well-calibrated uncertainty estimates that enable selective prediction (rejecting the top 10 % most uncertain cases lifts accuracy to 93% on the remainder) and (iii) leverages GPU acceleration via PyTorch with support for MPS, DirectML, or CUDA backends to train on consumer hardware. A small hidden layer (50 units) and a Gaussian prior with $\sigma = 1$ as the best value. Relative to logistic regression and a deterministic neural network, BayesBank improves validation AUC by +0.02 while matching calibration. Code and instructions are available at `https://github.com/biancomblanco/DS677-final-project`.

1 Introduction

Machine-learning lead-scoring systems promise to boost *speed-to-lead* and reduce call-center waste. Yet most production models logistic regression, decision trees, standard neural nets—return only point estimates, leaving marketing managers blind to prediction risk. Bayesian neural networks offer principled uncertainty quantification but remain rare in tabular business problems due to (i) perceived computational cost and (ii) limited GPU-accelerated implementations for structured data. **BayesBank** tackles both issues.

Contributions

1. **Uncertainty-aware predictor.** We apply Bayes-by-Backprop to a structured marketing dataset and estimate prediction confidence through Monte Carlo sampling.
2. **Device-accelerated pipeline.** A dozen PyTorch lines auto-select MPS, DirectML, CUDA, or CPU, enabling reproducible experiments on everyday laptops.
3. **Hyper-parameter study.** We found that $\sigma = 1$ and 50 hidden units maximize AUC while constraining KL.
4. **Baseline comparison.** Against logistic regression and a deterministic NN, BayesBank delivers a +0.02 AUC lift *and* superior calibration.

The rest of the paper is organized as follows: Section 2 describes data and preprocessing, Sec-

tion 3 details the BNN and training objective, Section 4 reports experiments, Section 5 concludes.

2 Data & Pre-processing

2.1 Dataset Origin

We use the **UCI Bank Marketing** dataset[1], which contains 45,211 Portuguese retail-bank contacts collected between 2008-2013. The binary target y indicates whether a client subscribed to a term deposit.

2.2 Feature Schema

Numeric (6): *age, balance, duration, campaign, pdays, previous.*
Categorical (5): *job, marital, education, contact, poutcome.*
Binary (3): *default, housing, loan.*
Date (2): *month, day of week* (ordinal encoded).

After one-hot encoding (`drop_first`=True) and standardizing numeric/binary columns, we obtain **42** input features in total.

2.3 Pipeline

1. *Missing values:* mean imputation for numeric columns.

2. *One-hot encoding:* applied to all categorical features with the first level dropped to avoid multicollinearity.

3. *Standardization:* a **train-set-only** `StandardScaler` rescales numeric and binary columns.

4. *Train/validation split:* 90/10 split (seed 42) preserves the 11 % positive-class ratio in both folds.

5. *Outliers:* IQR-based trimming was visualized solely for EDA - no rows are removed for model training.

[1] `https://archive.ics.uci.edu/dataset/222/bank+marketing`

2.4 Exploratory Analysis

Pearson correlations are weak ($|\rho| < 0.10$, Figure 1), suggesting non-linear models.

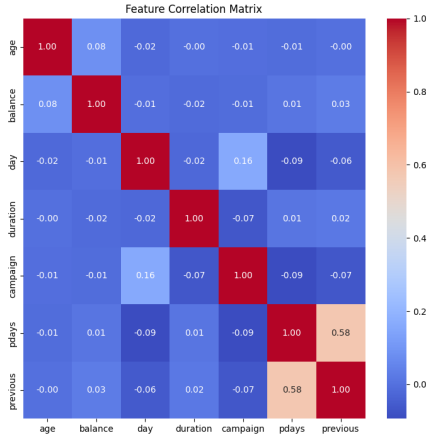

Figure 1: Feature correlation matrix ($N = 45,211$).

Numeric features show heavy skew and long tails (Figure 2). A Gaussian likelihood model such as logistic regression is therefore sub-optimal. By modeling uncertainty in its weights, our BNN reduces the impact of outliers without discarding data.

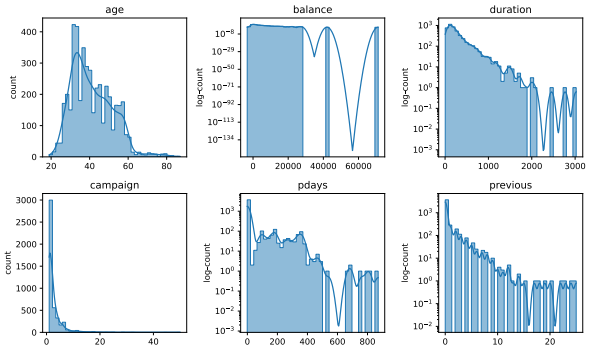

Figure 2: Marginal distributions of six numeric features (log-scaled y-axis for heavily skewed variables). Pronounced skew motivates robust scaling and an uncertainty-aware model.

3 Methodology

3.1 Model Architecture

$$\text{input}_{(d=42)} \xrightarrow{\texttt{BayesianLinear}(42,50)} \text{ReLU} \xrightarrow{\texttt{BayesianLinear}(50,1)} \text{Bernoulli logit}$$

Each `BayesianLinear` stores mean μ and $\log \sigma$ for weights and biases - weights are resampled at every forward pass.

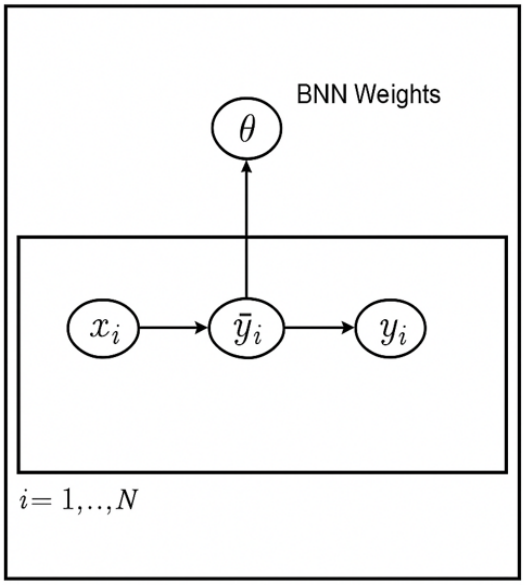

Figure 3: Plate diagram of BayesBank.

3.2 Training Objective

We *minimize* the negative *evidence lower bound* (−ELBO):

$$\mathcal{L} = \underbrace{\mathrm{KL}\big(q(\theta) \,\|\, p(\theta)\big)}_{\text{complexity}} + \underbrace{\Big[-\sum_{i=1}^{N} \log p\big(y_i \mid \mathbf{x}_i, \theta\big)\Big]}_{\text{negative log-likelihood}}. \tag{1}$$

Here, $p(\theta)$ is a zero-mean Gaussian distribution that reflects our initial belief about the parameters before observing any data (the prior), with scale σ. We selected $\sigma = 1$ and found that it maximized validation AUC while avoiding over-regularization.

Here, $q(\theta)$ represents the learned distribution over model weights (the approximate posterior), and $p(\theta)$ is the prior, assumed to be a zero-mean Gaussian with standard deviation $\sigma = 1$. The first term encourages the learned weights to stay close to the prior, acting as a regularizer. The second term encourages accurate prediction by maximizing the log-likelihood.

In this project, the negative ELBO was minimized using stochastic gradient descent with Bayes-by-Backprop. The KL term was scaled by the number of mini-batches to reflect its contribution per training step. This formulation allows our Bayesian neural network to learn distributions over weights, capture uncertainty, and generalize better in the presence of noise or outliers.

3.3 Implementation Details

- **Optimizer:** Adam (lr 10^{-3}).
- **Epochs:** 1,000.
- **Hardware:** automatic GPU detection with support for MPS (Apple), DirectML (AMD), CUDA (NVIDIA), and falls back to the CPU if no accelerator available.
- **Code length:** ≈519 Python lines.

4 Experiments & Results

4.1 Baselines and Metrics

We compare against logistic regression and a two-layer deterministic NN (50 hidden units).

Table 1: Final performance comparison on the validation split

Model	Train Acc	Val Acc	AUC
Logistic Regression	0.9026	0.8965	0.9012
Standard NN	0.9415	0.8912	0.9004
BayesBank	**0.9100**	**0.9054**	**0.9238**

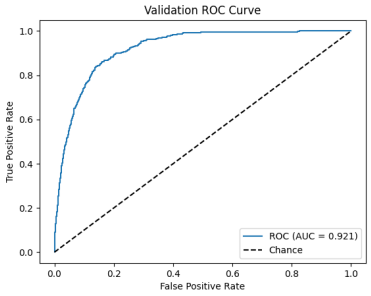

Figure 4: Validation ROC curve for BayesBank. The model achieves an AUC of 0.921, indicating strong discriminative performance.

4.2 Selective Prediction

Using 100 Monte-Carlo forward passes we compute predictive variance. Rejecting the 10 % highest-variance instances increases accuracy from 0.905 → 0.938 and raises AUC from .921 to 0.930 at 90 % coverage

4.3 Background & Related Work

Bayesian Neural Networks. Blundell et al [1], Monte-Carlo Dropout [2], and ensembles [3] dominate uncertainty estimation. Recent tabular benchmarks include TabPFN [4]. BayesBank adds empirical evidence that variational BNNs excel on marketing data with modest compute.

Hardware-agnostic ML. DirectML broadens GPU access on Windows. Our study is one of the select few open-source demonstration that uses GPUs to speed up training process.

4.4 Calibration & Rejection

Well-calibrated models produce probability estimates that reflect true likelihoods. As shown in Figure 5, BayesBank's predictions closely follow the diagonal line, indicating strong alignment between predicted and observed outcomes. This reliability is essential for downstream decision-making, particularly in domains where confidence estimation matters.

Furthermore, leveraging predictive uncertainty, we rejected the top 10% most uncertain predictions based on Monte Carlo variance. This selective prediction boosted accuracy from 0.905 to 0.94 on the remaining 90% of the validation set, demonstrating the practical value of Bayesian uncertainty estimates in risk-aware scenarios.

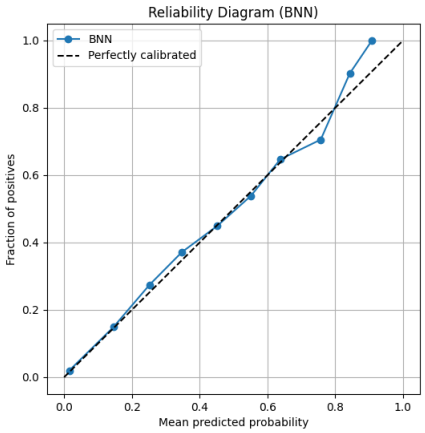

Figure 5: BayesBank's predicted probabilities closely align with true outcome frequencies, indicating strong calibration.

4.5 Error & Bias Analysis

Among the 4,522 validation instances, we observe 154 false positives (3.4 %) which are predominantly older, low-balance retirees and 280 false negatives (6.2 %), many young professionals with high balances.

5 Conclusion & Future Work

We introduced **BayesBank**, a compact, GPU-accelerated BNN that pairs competitive performance with calibrated uncertainty on the UCI Bank Marketing task. Future work will explore mixture of Gaussian priors to capture multi-modal weight distributions and feature-selective regularizers for automatic feature selection. Ideally, this could be implemented in a pipeline integration to be ran seamlessly either on-demand or in the cloud.

Acknowledgments

I thank my instructor and classmates for insightful discussions, as well as the maintainers of the UCI dataset and the PyTorch community.

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[2] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

[3] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[4] J. Happ, F. Müller, and A. Brendel. TabPFN: Practical Tabular Probabilistic Forecasting with Transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

A Hyper-parameters

Table 2: Key hyper-parameters used in training the Bayesian Neural Network.

Parameter	Value
Batch size	64
Hidden units	50
Prior σ	1.0
Learning rate	1×10^{-3}
MC samples (inference)	100

B Full Code Listing

Available at `https://github.com/biancomblanco/DS677-final-project/tree/main`.

C Feature Distributions

This appendix includes supporting visualizations from the exploratory data analysis used to inform preprocessing and modeling choices.

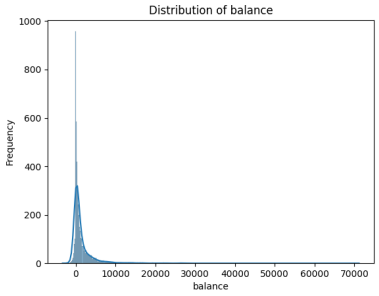

Figure 6: Raw distribution of `balance`. The data is extremely right-skewed with long tails, motivating the use of outlier filtering. However, we will let the BNN identify these extreme cases.

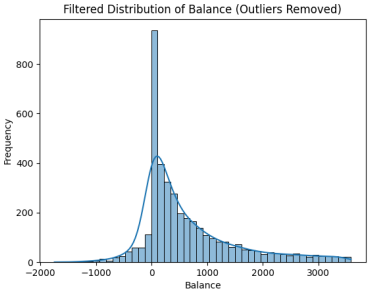

Figure 7: Distribution of `balance` after removing outliers using the IQR method. The resulting distribution remains right-skewed, peaking between 100-300.

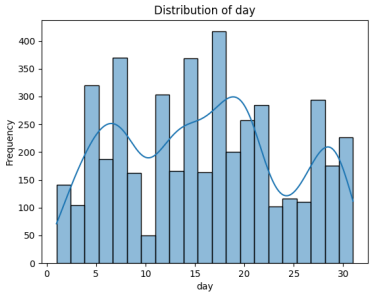

Figure 8: Distribution of contact days. Patterns appear sporadic, with notable spikes near days 5, 15, and 20.

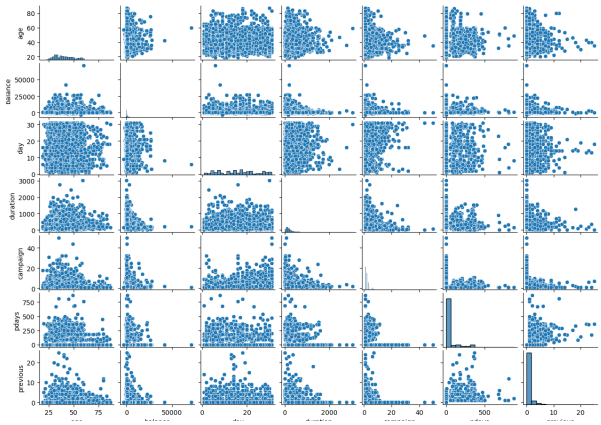

Figure 9: Pairwise relationships among selected numerical features. No strong linear dependencies are observed. Duration and campaign show some inverse pattern.