Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Customer Churn Prediction Using a Multilayer Feedforward Neural Network with Behavioral and Service-Usage Features

Shabista Khan[1], Dipti Sharma[2]

[1] M.Tech Scholar, Department of Computer Science and Engineering, Patel College of Science and Technology, Indore, India
[2] Assistant Professor, Patel College of Science and Technology, Indore, India

Corrected and extended technical manuscript based on a previously reported experiment [20]. The executable evidence supports a feedforward neural network, not an implemented PSO hybrid. The stored 95.77% result is not claimed as an independent rerun.

Abstract

This paper reconstructs a customer-churn classification experiment from a supplied 64,374-record CSV, MATLAB source file, and stored training figures. Ten demographic, relationship, usage, support, payment, subscription, contract, spend, and interaction variables are used; CustomerID is excluded. The executable script trains a feedforward fitting network on records 1–10,000 and evaluates rounded outputs on records 10,001–13,001, a 3,001-record external holdout. The network has five hidden layers with ten neurons each. Stored MATLAB artefacts identify Levenberg–Marquardt backpropagation, internal random train/validation/test division, termination at epoch 24, best validation mean-squared error of 0.032864 at epoch 18, final training performance of 0.00902, and an all-development regression coefficient of 0.93989. The external figure reports 95.77% accuracy, corresponding to 4.23% binary classification error under the script's calculation. A source variable labels this percentage as MAPE, but the code computes no percentage-error denominator; the correct interpretation is classification error. No active particle-swarm-optimization routine is present. The work contributes an implementation-aligned model description, a full-file data audit, separation of internal validation from external holdout evaluation, and an explicit statement of unavailable evidence. Because the prediction vector, random seed, MAT-file, and trained object are absent, confusion-matrix metrics, AUC, calibration, and exact rerun claims are not fabricated. The manuscript closes with a protocol for a reproducible comparative experiment.

Keywords: customer churn; feedforward neural network; multilayer perceptron; Levenberg–Marquardt; behavioral features; reproducibility

1. Introduction

Customer churn classification estimates whether a customer will discontinue a service relationship within a defined horizon. Accurate ranking can help focus retention resources, but a model is useful only when its inputs are available before the decision and its errors are evaluated against operational costs. Churn studies have used random forests, support vector machines, boosting, neural networks, and composite deep architectures [1]–[10]. Comparisons across papers remain difficult because datasets, label definitions, class prevalence, and split protocols differ.

This paper addresses a narrower but important issue: consistency between a reported model and its executable evidence. A prior paper by the same authors described the experiment as a hybrid swarm-intelligence/deep-neural-network model [20]. Audit of the supplied MATLAB file shows that its active statements create and train a feedforward fitting network. The only block containing 'PSO' is commented out, incomplete, and never connected to the network. The present manuscript corrects the technical characterization, retains the stored result with explicit provenance, and does not present textual revision as a new experiment.

The contributions are fourfold: (1) audit of the complete CSV and the row ranges actually used; (2) implementation-aligned description of a five-hidden-layer network trained with Levenberg–Marquardt backpropagation; (3) correction of the script's MAPE terminology to binary classification error; and (4) a traceability boundary separating observable results from metrics that cannot be derived without the prediction vector.

2. Source Package and Reproducibility Status

The source package contains churn_data.csv, a 132-line MATLAB text file, and screenshots of the network, training, regression, performance, state, and external prediction plots. The exact churndata.mat loaded by the source is absent. The trained network object, TrainingRecord object, internal split indices, random seed, and holdout scores are also absent. Consequently, descriptive statistics are recomputed from the CSV, whereas neural-network outcomes are transcribed from the stored figures.

The local CSV structure is consistent with the publicly listed Customer Churn Dataset [19], but the package does not include a download record or checksum from the original source. The audited local file has SHA-256 digest 29cdc0c919dd0d5c2b5e3010db8a6eb1aea77218fc4d46ae7b568474112ddca4. This digest identifies the analysed file; it does not establish collection provenance or prove equivalence with the missing MAT-file.

3. Dataset Audit

3.1 Schema and Data Quality

The CSV contains 64,374 rows and 12 integer-valued columns. CustomerID is unique, no exact duplicate rows are present, and no value is missing. Churn is binary. The ten predictors are Age, Gender, Tenure, Usage Frequency, Support Calls, Payment Delay, Subscription Type, Contract Length, Total Spend, and Last Interaction. Gender has observed codes {0,1}; Subscription Type and Contract Length have codes {1,2,3}. Because semantic category labels are not supplied, the codes are not assigned names in this paper.

Partition	Rows	n	Non-churn	Churn	Churn rate
Development	1–10,000	10,000	7,373	2,627	26.27%
External holdout	10,001–13,001	3,001	2,225	776	25.86%
Unused	13,002–64,374	51,373	24,283	27,090	52.73%
Complete CSV	1–64,374	64,374	33,881	30,493	47.37%

The target distribution changes across row ranges. The full CSV is nearly balanced, but both partitions used by the script contain approximately 26% churn, while the unused remainder contains 52.73% churn. This pattern makes the row-selection rule scientifically relevant and precludes describing the active experiment as a random split of the whole file.

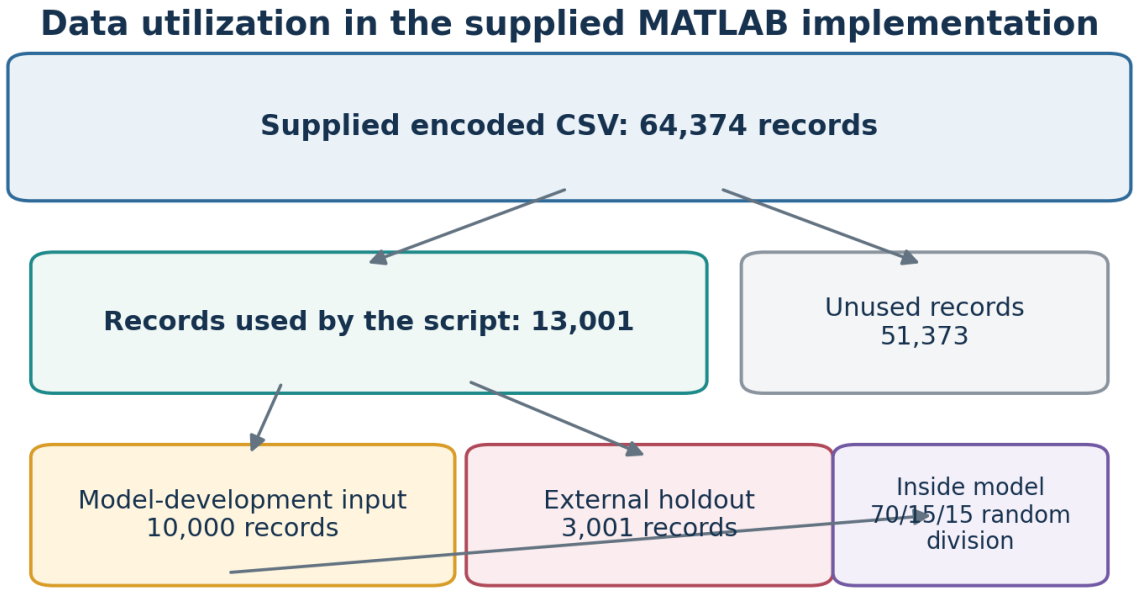

Figure 1. Index-defined outer development, external holdout, and unused partitions.

3.2 Descriptive Associations

On the complete CSV, the strongest Pearson correlations between eligible predictors and Churn are Payment Delay (0.557), Support Calls (0.305), and Tenure (0.195). Usage Frequency (−0.115), Total Spend (−0.079), and Gender code (−0.165) are negatively correlated. These are univariate descriptive associations, not neural-network feature attributions and not causal effects.

CustomerID has a correlation of approximately 0.530 with Churn, despite being an identifier. Its exclusion by the source code is therefore important; otherwise a model could exploit row-order or construction artefacts. The finding also motivates testing models across later cohorts instead of assuming independent and identically distributed rows.

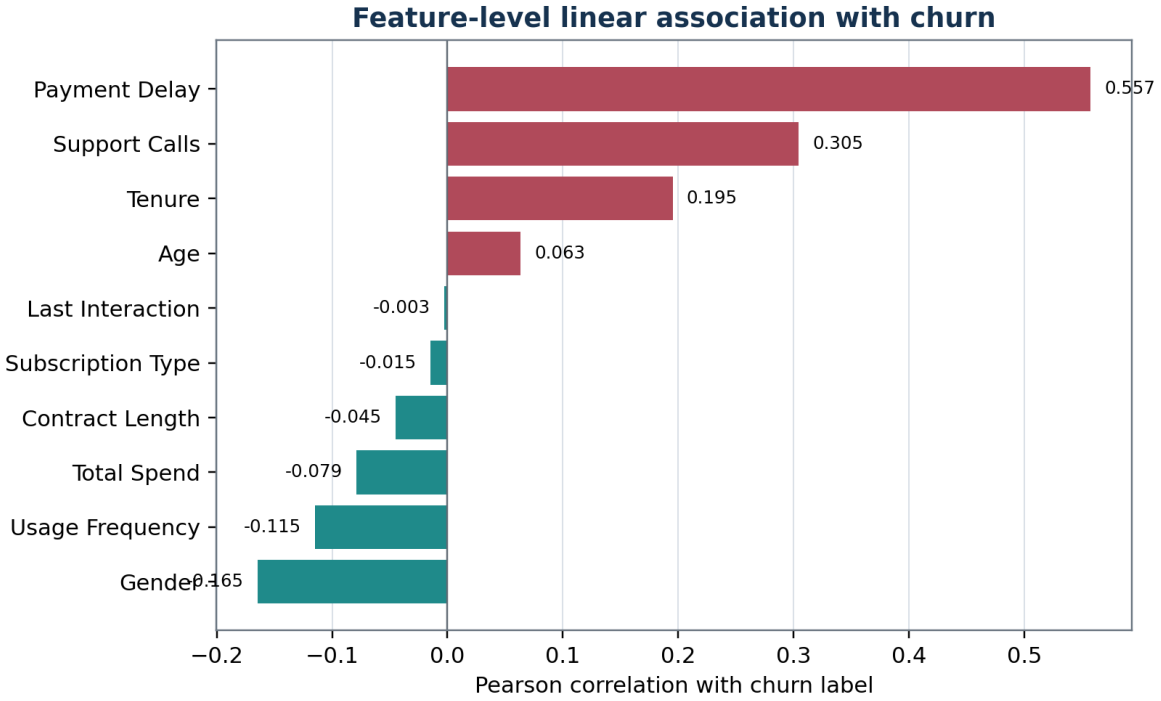

Figure 2. Pearson correlation of the ten model predictors with Churn in the complete CSV.

4. Implemented Neural-Network Method

4.1 Outer Split and Internal Division

The script constructs a 10-column input matrix and a binary target vector, assigns rows 1–10,000 to trinp/trout, and assigns rows 10,001–13,001 to testinp/testout. The network training routine internally divides the development block using dividerand. MATLAB documentation describes this as random division and the stored interface identifies separate training, validation, and internal-test subsets [17]. The 3,001-row block is therefore an outer holdout, not the internal test partition shown in the regression panel.

4.2 Architecture

The statement newfit(trinp', trout', [10 10 10 10 10]) creates five hidden layers with ten neurons per layer and a scalar output. For hidden layer ℓ, the forward calculation is $a^{(\ell)}=\varphi(W^{(\ell)}a^{(\ell-1)}+b^{(\ell)})$. A final affine/transfer mapping produces $f(x;\theta)$. Under dense connectivity, the 10–10–10–10–10–10–1 structure contains 551 weights and biases.

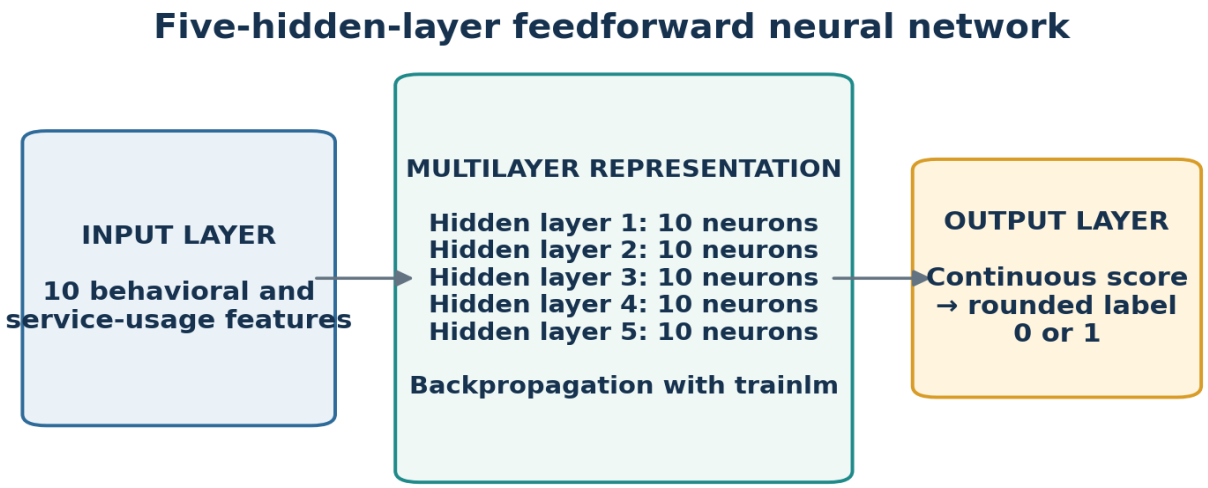

Figure 3. Implemented five-hidden-layer feedforward architecture.

4.3 Training Objective and Optimizer

The fitting network uses mean squared error as its performance function [18]. The stored training interface identifies trainlm, MATLAB's Levenberg–Marquardt backpropagation implementation [16]. For residual vector e and Jacobian J, a standard LM update has the form $\theta_{k+1}=\theta_k-(J^TJ+\mu I)^{-1}J^Te$. The damping parameter mediates between Gauss–Newton-like and more conservative updates.

$$MSE = (1/n)\,\Sigma_{i=1}^{n}\,[y_i - f(x_i;\theta)]^2 \quad (1)$$

4.4 Decision Rule

The network produces continuous holdout outputs. The script applies round, calculates mean absolute error between the binary targets and rounded labels, multiplies by 100, and subtracts the result from 100. For binary y and rounded ŷ, this mean absolute error equals the misclassification proportion.

$$Error\ (\%) = 100\cdot(1/n)\Sigma|y_i-\hat{y}_i|; \quad Accuracy\ (\%) = 100-Error\ (\%) \quad (2)$$

Although the source variable is named MAPE, conventional MAPE requires a denominator involving the actual value and is undefined for zero targets. The active calculation contains no denominator. 'Classification error' is therefore the accurate term.

4.5 Absence of Active PSO

An implemented PSO routine would require particles, velocities, personal and global best states, a fitness function tied to network parameters, and iterative swarm updates. None appears in active code. A multiline comment contains the label PSO but is ignored by MATLAB and does not invoke or modify NNmodel. The experimental result can only be attributed to the feedforward network and its configured training function.

5. Results

5.1 Training and Internal Diagnostics

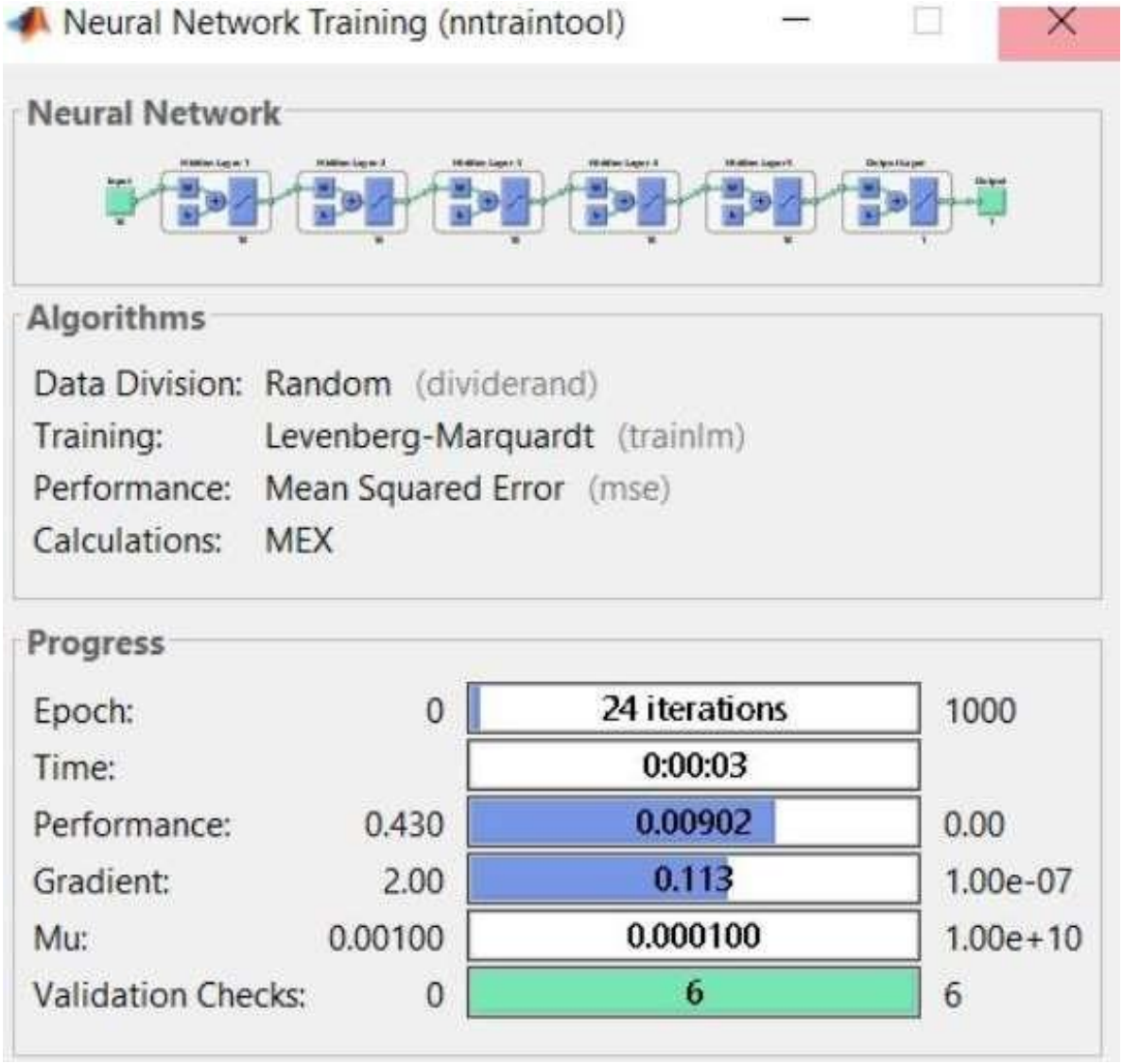

Figure 4. Stored MATLAB training summary; source artefact reproduced for traceability.

Training terminates at epoch 24. The interface shows performance 0.00902, gradient approximately 0.113, μ=0.000100, six validation checks, and elapsed time 00:00:03 in the recorded environment. The architecture panel confirms five ten-unit hidden layers. The elapsed time is not generalized beyond that environment.

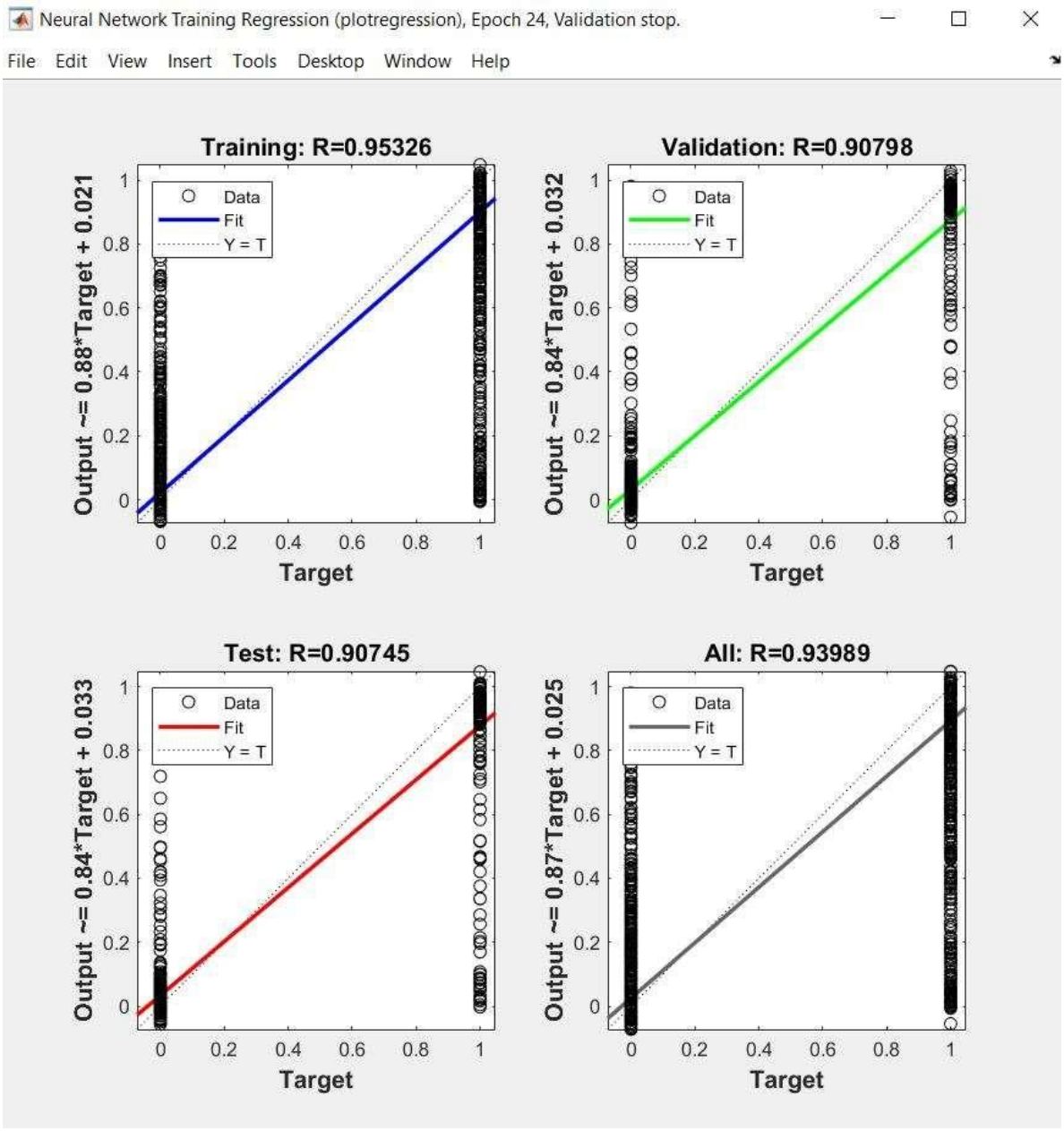

Figure 5. Stored MATLAB regression diagnostics for internal development partitions.

The displayed regression coefficients are R=0.95326 for internal training, 0.90798 for validation, 0.90745 for the internal test subset, and 0.93989 for all development samples. These coefficients describe linear association between continuous outputs and binary targets; they are not classification accuracies.

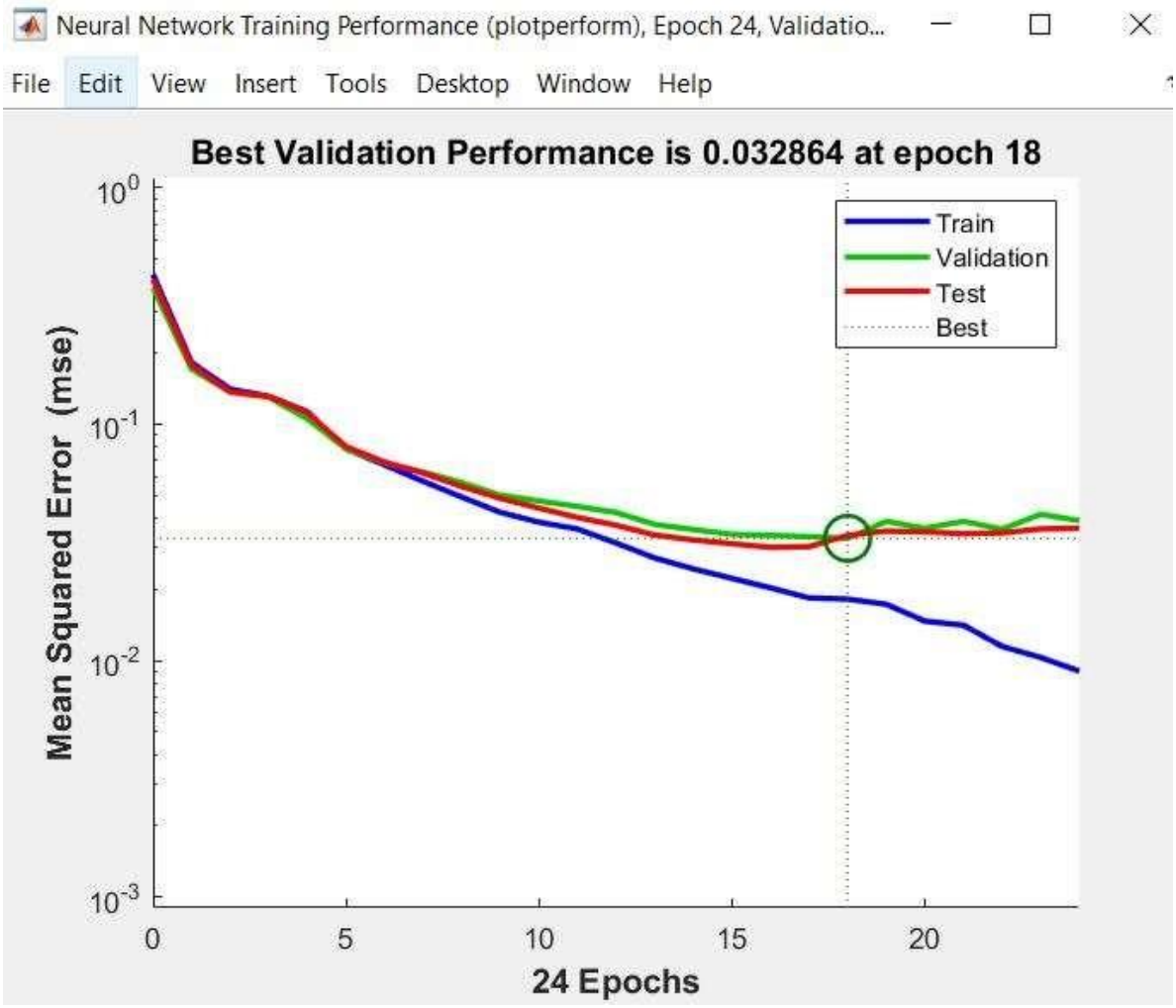

Figure 6. Stored performance curve with best validation MSE at epoch 18.

Best validation MSE is 0.032864 at epoch 18. Training stops at epoch 24 after six validation failures. Final training performance 0.00902 and best validation MSE 0.032864 refer to different partitions and should not be collapsed into one MSE value.

5.2 External-Holdout Accuracy

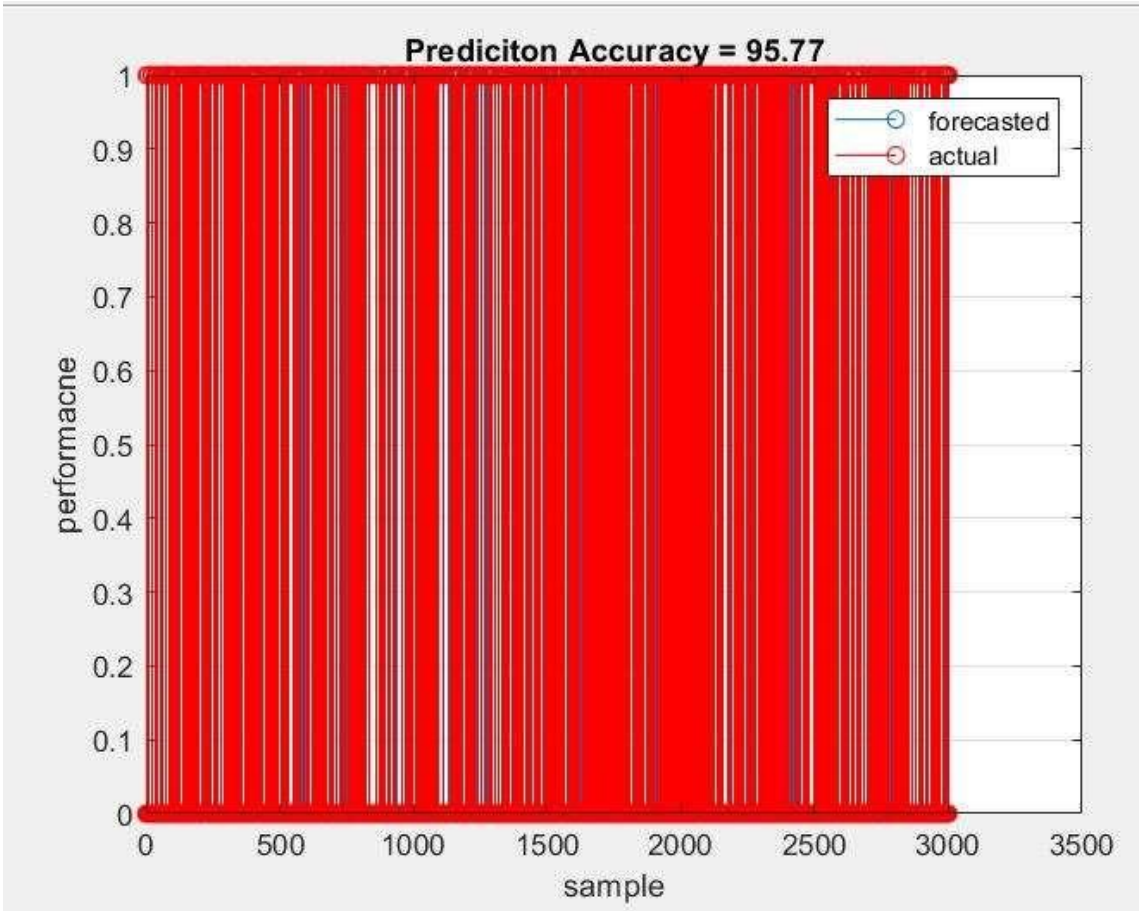

Figure 7. Stored rounded-forecast and actual-label plot for the external holdout.

The external plot reports Prediction Accuracy=95.77 on 3,001 holdout records, implying 4.23% classification error under Eq. (2). The displayed value is preserved as a result of the source experiment. Because the exact scores and rounded predictions are absent, the integer confusion matrix and all metrics derived from it are unavailable.

Quantity	Reported value	Evidence boundary
Epoch at stop	24	Stored training interface
Best validation MSE	0.032864 at epoch 18	Stored performance plot
Final training performance	0.00902	Stored training interface
All-development R	0.93989	Stored regression plot
External holdout n	3,001	Inclusive source indices
External accuracy	95.77%	Stored external plot
External error	4.23%	100−95.77

Quantity	Reported value	Evidence boundary
Precision/recall/F1/AUC	Not available	Predictions/scores not supplied

6. Discussion

6.1 Interpretation

The stored experiment indicates that a compact 551-parameter feedforward network can closely classify the selected holdout after rounding. The holdout slice of the supplied CSV has a 74.14% majority-class baseline, so the displayed 95.77% is materially above that descriptive comparator if the MAT-file preserves identical rows. A same-protocol rerun is still required to verify the result and compare trained alternatives.

The result does not establish that five hidden layers are necessary. No shallow-network ablation or equally tuned logistic, random-forest, or boosting baseline is available. Random forests [21] and XGBoost [22] should be included in a new tabular benchmark. Multiple seeds are necessary because both parameter initialization and dividerand are stochastic.

6.2 Metric Adequacy

Accuracy alone does not reveal churn recall or false-positive workload. Future reports should provide the confusion matrix, precision, recall, specificity, F1, and MCC [15]. ROC-AUC and PR-AUC should be calculated from continuous scores, with PR-AUC emphasized when prevalence is low [14]. If scores are interpreted as probabilities, Brier score and calibration curves are required. A campaign-oriented analysis should report lift or expected value at realistic contact capacities.

6.3 Validity and Generalization

The outer split follows row order without a documented timestamp. The sharp prevalence change in the unused remainder indicates dataset shift across row ranges. Testing on later or independently collected cohorts would provide stronger evidence. Category semantics, feature timing, and the CSV-to-MAT conversion should be documented before any claim of operational generalization.

Model-selection bias is another risk. If the holdout result influenced architecture or reporting choices, it is no longer a pristine final test. Nested validation and a locked holdout reduce this bias [13]. The source package contains no search trace, so the extent of prior experimentation cannot be determined.

6.4 Explainability and Responsible Use

Descriptive correlations suggest that payment delay and support calls are informative, but a model-specific analysis is needed. LIME [11] or SHAP [12] could be applied to saved scores, with sensitivity analysis for correlated features. Explanations should not be represented as causal effects. Age and gender codes require governance review, lawful purpose, and segment-level error analysis before operational use.

7. Limitations

- The neural network was not independently rerun; its results are transcribed from stored artefacts.

- The exact MAT-file, conversion process, random seed, network object, partitions, and predictions are unavailable.
- Only 13,001 of 64,374 rows enter the reported experiment, and the remainder has a different target distribution.
- The category-code meanings and temporal definition of churn are undocumented.
- Accuracy is the only stored external classification metric; class-wise performance and calibration cannot be recovered.
- No controlled baseline, ablation, uncertainty analysis, fairness assessment, or economic evaluation is available.

8. Reproducible Extension Protocol

A new experiment should import the CSV directly, record its checksum, define category semantics, and select a deployment-aligned split before training. All encoding, scaling, feature selection, and resampling must be fit within development folds. Seeds and partition indices must be saved. Regularized logistic regression, random forest, XGBoost, shallow neural networks, and the documented five-layer network should receive comparable tuning budgets.

The final package should archive continuous scores and labels for every evaluation row, confusion matrices, calibration data, model checkpoints, environment versions, and timing. Repeated seeds and confidence intervals should quantify uncertainty. An out-of-time cohort should test drift, and retention value should be evaluated at predeclared thresholds. If PSO is added, it must be implemented, parameterized, logged, and ablated against the same non-PSO network.

9. Conclusion

The executable evidence supports a five-hidden-layer feedforward neural network trained with Levenberg–Marquardt backpropagation. It does not support a particle-swarm-optimized model. On a sequential 3,001-record external holdout, the stored run reports 95.77% accuracy and 4.23% classification error. Correct terminology and traceability strengthen the result, while missing predictions, stochastic controls, baselines, and temporal context constrain its interpretation. The appropriate next contribution is a fresh, versioned, leakage-aware comparison—not another unverified hybrid label.

References

1. M. Imani, M. Joudaki, A. Beikmohammadi, and H. R. Arabnia, “Customer churn prediction: A systematic review of recent advances, trends, and challenges in machine learning and deep learning,” Machine Learning and Knowledge Extraction, vol. 7, no. 3, art. 105, 2025. doi: 10.3390/make7030105. https://doi.org/10.3390/make7030105
2. M. A. Manzoor et al., “A review on machine learning methods for customer churn prediction and recommendations for business practitioners,” IEEE Access, vol. 12, pp. 70434–70463, 2024. doi: 10.1109/ACCESS.2024.3402092. https://doi.org/10.1109/ACCESS.2024.3402092
3. A. Ullah, B. Raza, A. K. Malik, M. Imran, S. U. Islam, and S. W. Kim, “A churn prediction model using random forest: Analysis of machine learning techniques for churn prediction and factor identification in telecom sector,” IEEE Access, vol. 7, pp. 60134–60149, 2019. doi: 10.1109/ACCESS.2019.2914999. https://doi.org/10.1109/ACCESS.2019.2914999

4. S. Wu, W.-C. Yau, T.-S. Ong, and S.-C. Chong, “Integrated churn prediction and customer segmentation framework for telco business,” IEEE Access, vol. 9, pp. 62118–62136, 2021. doi: 10.1109/ACCESS.2021.3073776. https://doi.org/10.1109/ACCESS.2021.3073776
5. S. S. Poudel, S. Pokharel, and M. Timilsina, “Explaining customer churn prediction in telecom industry using tabular machine learning models,” Machine Learning with Applications, vol. 17, art. 100567, 2024. doi: 10.1016/j.mlwa.2024.100567. https://doi.org/10.1016/j.mlwa.2024.100567
6. V. Chang et al., “Prediction of customer churn behavior in the telecommunication industry using machine learning models,” Algorithms, vol. 17, no. 6, art. 231, 2024. doi: 10.3390/a17060231. https://doi.org/10.3390/a17060231
7. S. K. Wagh, A. A. Andhale, K. S. Wagh, J. R. Pansare, S. P. Ambadekar, and S. H. Gawande, “Customer churn prediction in telecom sector using machine learning techniques,” Results in Control and Optimization, vol. 14, art. 100342, 2024. doi: 10.1016/j.rico.2023.100342. https://doi.org/10.1016/j.rico.2023.100342
8. A. Khattak et al., “Customer churn prediction using composite deep learning technique,” Scientific Reports, vol. 13, art. 17294, 2023. doi: 10.1038/s41598-023-44396-w. https://doi.org/10.1038/s41598-023-44396-w
9. X. Liu, G. Xia, X. Zhang, W. Ma, and C. Yu, “Customer churn prediction model based on hybrid neural networks,” Scientific Reports, vol. 14, art. 30707, 2024. doi: 10.1038/s41598-024-79603-9. https://doi.org/10.1038/s41598-024-79603-9
10. T. Vafeiadis, K. I. Diamantaras, G. Sarigiannidis, and K. C. Chatzisavvas, “A comparison of machine learning techniques for customer churn prediction,” Simulation Modelling Practice and Theory, vol. 55, pp. 1–9, 2015. doi: 10.1016/j.simpat.2015.03.003. https://doi.org/10.1016/j.simpat.2015.03.003
11. M. T. Ribeiro, S. Singh, and C. Guestrin, “Why should I trust you? Explaining the predictions of any classifier,” in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144, 2016. doi: 10.1145/2939672.2939778. https://doi.org/10.1145/2939672.2939778
12. S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in Advances in Neural Information Processing Systems 30, pp. 4765–4774, 2017. https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html
13. G. C. Cawley and N. L. C. Talbot, “On over-fitting in model selection and subsequent selection bias in performance evaluation,” Journal of Machine Learning Research, vol. 11, pp. 2079–2107, 2010. https://www.jmlr.org/papers/v11/cawley10a.html
14. T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” PLOS ONE, vol. 10, no. 3, e0118432, 2015. doi: 10.1371/journal.pone.0118432. https://doi.org/10.1371/journal.pone.0118432
15. D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient over F1 score and accuracy in binary classification evaluation,” BMC Genomics, vol. 21, art. 6, 2020. doi: 10.1186/s12864-019-6413-7. https://doi.org/10.1186/s12864-019-6413-7

16. MathWorks, “trainlm: Levenberg–Marquardt backpropagation,” Deep Learning Toolbox documentation, accessed Aug. 9, 2026. https://www.mathworks.com/help/deeplearning/ref/trainlm.html
17. MathWorks, “dividerand: Divide data randomly,” Deep Learning Toolbox documentation, accessed Aug. 9, 2026. https://www.mathworks.com/help/deeplearning/ref/dividerand.html
18. MathWorks, “fitnet: Function fitting neural network,” Deep Learning Toolbox documentation, accessed Aug. 9, 2026. https://www.mathworks.com/help/deeplearning/ref/fitnet.html
19. M. S. Azeem, “Customer churn dataset,” Kaggle, dataset page, accessed Aug. 9, 2026. https://www.kaggle.com/datasets/muhammadshahidazeem/customer-churn-dataset
20. P. Gehlot, C. K. Tiwari, and D. Nagar, “A hybrid swarm intelligence-deep neural network model for churn rate prediction,” International Journal of Scientific Research in Engineering and Management, vol. 9, no. 9, 2025. doi: 10.55041/IJSREM52624. https://doi.org/10.55041/IJSREM52624
21. L. Breiman, “Random forests,” Machine Learning, vol. 45, pp. 5–32, 2001. doi: 10.1023/A:1010933404324. https://doi.org/10.1023/A:1010933404324
22. T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016. doi: 10.1145/2939672.2939785. https://doi.org/10.1145/2939672.2939785