

Leakage-Controlled Benchmarking of Deep and Classical Models for Telecom Churn Prediction

¹Ritu, ²Manni Kumar, ³Priyanshu Sharma, ⁴Shraddha Patel

^{1,2,3,4}Department of Computer Science and Engineering,

Chandigarh University, Mohali, Punjab, India

¹ritugahlawat02@gmail.com, ²mannikumar55@gmail.com, ³priyanshzmanu123@gmail.com,

⁴shraddhapatel.282005@gmail.com

Abstract—Five predictive models (Logistic Regression, Random Forest, XGBoost, LSTM-only, and CNN-LSTM) are tested for the churn-prediction task. In this study, one of the modeling strategies used for predicting telecom customer churn is the LSTM architecture. The data set used for this work is the popular IBM Telco Customer Churn data set, which has 7,043 customer records. A leakage-controlled protocol removes identifier, geographic, and target-derived fields before an 80/20 stratified split, with learned preprocessing, class-imbalance handling, and feature-selection decisions restricted to the training partition. Five-fold cross-validation selects SMOTENC over class weighting by a mean macro-F1 difference of 0.0123, while recursive feature elimination is rejected because its macro-F1 change is only +0.0003. The locked test partition is evaluated once. Random Forest obtains the highest accuracy (0.7779) and macro-F1 (0.7245); Logistic Regression obtains the highest churn F1-score (0.6124), recall (0.7647), and ROC-AUC (0.8403); and XGBoost obtains the highest PR-AUC (0.6366). CNN-LSTM does not lead on any reported test metric. For the locked Random Forest model, SHAP values are aggregated from one-hot indicators to raw feature families, and constrained counterfactual search is limited to actionable service attributes. Two permitted service changes reduce the predicted churn probability below 0.50 for each of three examined high-risk cases. Because the evaluation uses one test split without significance testing, model orderings are descriptive rather than inferential.

Keywords—Class Imbalance, CNN-LSTM, Counterfactual Explanations, Explainable AI, Model Benchmarking, SHAP, SMOTENC, Telecom Churn Prediction.

I. INTRODUCTION

Subscriber acquisition in a saturated telecommunications market costs more than retention, which makes early identification of customers likely to leave a standing commercial requirement. Two properties of the problem keep it methodologically awkward: churners are a minority, so accuracy is a poor summary of performance; and a retention team needs to know which attributes to act on, not only which customers are at risk. Three gaps recur in the literature. Comparisons between deep and classical models are often reported without a common preprocessing and evaluation protocol, so it is unclear whether an architectural advantage or a pipeline difference is being measured. Imbalance correction and feature selection are frequently applied before partitioning, or chosen against the data used for reporting, which inflates the resulting figures. And explanations are usually given over encoded model inputs

and over attributes no operator can change, so they explain the model without supporting a decision [1]–[3].

To address these gaps, this work introduces a uniform protocol, with leakage prevention, for all five benchmarked models. Its contributions are a benchmark in which thirteen identifier, geographic and target-related columns are removed and an 80/20 stratified split precedes every fitted transformation; imbalance and feature-selection decisions taken by cross-validation on the training partition alone against rules fixed in advance; an Explain-Act lock in which the explained model is chosen from training-only evidence before the test partition is read; SHAP attributions aggregated from one-hot indicators back to raw feature families; and a counterfactual search restricted to attributes a provider can change. The benchmark returns a partly negative answer on the architectural question, reported as such rather than recast.

II. RELATED WORK

Churn prediction on telecom account data has been studied with most standard supervised families. Earlier empirical work reported the predictive value of behavioural and account attributes over demographic ones [1], [2], and social-network features add signal where call records are available [3]. Hybrid and ensemble designs dominate the recent literature: a hybrid of logistic regression with decision-tree partitioning [4], a KNN-LR hybrid [5], boosted trees on large-scale operator data [6], and comparative studies of classical learners on the public benchmarks [7], [8], [9], [10]. Deep architectures have been applied more recently, including convolutional churn networks [11], recurrent and hybrid models [12], [13], [14], and boundary-sampling variants for the imbalanced case [15]; comparable pipelines are reported in adjacent sectors.

Reported accuracies span roughly the high seventies to the high nineties, but the figures are not directly comparable; Section V-B sets out why. Class imbalance is usually handled by synthetic oversampling [17], for which categorical-aware variants and reference implementations exist [18], but the choice between oversampling and cost-sensitive weighting is rarely made against a rule fixed in advance.

Interpretability is increasingly reported alongside accuracy. SHAP provides additive attributions with a consistency guarantee [19] and a tree-specific estimator supporting global aggregation [20], and has been applied to churn models directly [21]. Counterfactual explanation offers the complementary view of what would have to differ for a decision to change [22],

formalised for tabular classifiers as diverse feasible alternatives [23]. What is rarely enforced is that the permitted changes be ones the provider can actually make. The gap addressed here is therefore methodological: under one protocol in which every decision is taken on training data alone, does a convolutional-recurrent model offer a consistent advantage over classical and ensemble baselines on static tabular customer records, and can the explanation of the selected model be made actionable rather than merely faithful?

III. METHODOLOGY

Fig. 1 summarizes the protocol. Each stage states the decision taken, the evidence it was taken on, and the point at which it was fixed.

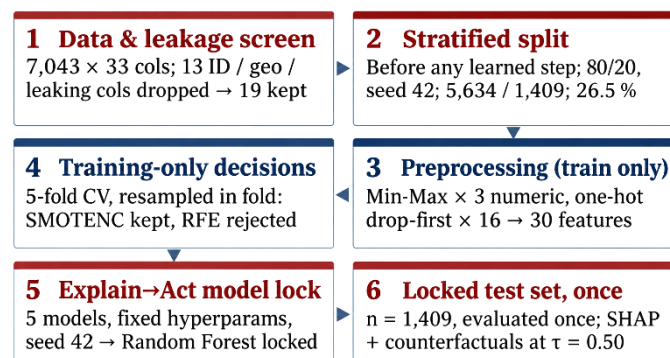


Fig. 1. Leakage-controlled benchmarking and Explain→Act protocol. Red headers mark the four leakage-control points; every learned transformation is fitted on the training partition only.

A. Data, Leakage Control and Partitioning

The data set is the publicly available IBM Telco customer churn sample [24], in its 33-column IBM Cognos Analytics form: 7,043 records, one row per customer, with a binary Churn Value target. The class distribution is 5,174 non-churners to 1,869 churners, a churn rate of 26.54 %. The widely circulated 21-column variant does not carry the columns discussed next.

Thirteen columns are removed before anything is fitted. CustomerID and Count carry no predictive content, and seven geographic columns are dropped because the sample is drawn from a single state and encodes location at a granularity that would fit noise. Four leak the target: Churn Label redundantly encodes Churn Value; Churn Score and CLTV are model-derived scores whose relationship to the target is undocumented in the source; and Churn Reason is populated only for customers who churned, so its presence alone identifies the positive class. Nineteen raw predictors remain, three numeric (Tenure Months, Monthly Charges, Total Charges) and sixteen categorical. Eleven records carry a blank Total Charges value, all at Tenure Months = 0; these accounts have not yet been billed, so the blank is set to zero rather than imputed.

The data are split once, before any transformation is fitted: an 80/20 split stratified on the target with `random_state = 42`, giving 5,634 training and 1,409 test records at unchanged prevalence. A ColumnTransformer,

fitted on the training partition only, applies min-max scaling to the three numeric columns and one-hot encoding with the first level dropped to the sixteen categorical columns, producing 30 encoded features. For the neural models a further 85/15 stratified split of that partition supplies a validation subset for early stopping; the subset receives a separate transformer fitted on the neural-training portion alone and is never resampled, so it retains the natural 26.54 % prevalence. The test partition is evaluated once, after every decision below has been fixed.

B. Class Imbalance and Feature Selection

Five-fold stratified cross validation was used on the training partition for two decisions, both of which were against a rule set before the comparison was performed. The first is imbalance handling: SMOTENC, the categorical-aware variant of SMOTE [17] implemented in imbalanced-learn [18], against `class_weight = 'balanced'`. Resampling is applied inside each fold, so no synthetic record derived from a fold's training portion can appear in its validation portion. The rule was to adopt SMOTENC only if it exceeded class weighting by at least 0.01 mean macro-F1. Table I shows that it does, by 0.0123 (0.7369 vs. 0.7246), so SMOTENC was applied to the training partition throughout.

Table I also records a cost the rule does not capture: mean PR-AUC falls from 0.6592 to 0.6474, and Logistic Regression's churn recall from 0.8114 to 0.7773. The decision was taken on the criterion fixed in advance, and its cost is stated rather than suppressed.

TABLE I
CLASS-IMBALANCE STRATEGY COMPARISON (FIVE-FOLD CROSS-
VALIDATION, TRAINING PARTITION ONLY)

Strategy	Model	F1 churn	Recall churn	Macro- F1	PR- AUC
Class weight	LR	0.6421	0.8114	0.7308	0.6780
Class weight	RF	0.5700	0.5077	0.7185	0.6405
Class weight	Mean	0.6061	0.6595	0.7246	0.6592
SMOTENC	LR	0.6439	0.7773	0.7380	0.6696
SMOTENC	RF	0.6178	0.6448	0.7358	0.6252
SMOTENC	Mean	0.6308	0.7110	0.7369	0.6474

Fold means. The selection rule uses the Mean rows of the Macro-F1 column.

The second decision is feature selection: recursive feature elimination [25] to 15 of the 30 encoded features, to be retained only if it improved mean macro-F1 by at least 0.005 without reducing mean PR-AUC by more than 0.005. It gained $\Delta_{\text{macro-F1}} = +0.0003$, an order of magnitude below the margin, and lost $\Delta_{\text{PR-AUC}} = -0.0037$, so it was rejected and all 30 features retained. This comparison ran on the resampled training partition, so its absolute values are optimistic and the decision rests on the difference between the two conditions.

C. Models and Configurations

Five models are trained on the identical resampled and encoded training partition: Logistic Regression as linear baseline,

Random Forest [26] and XGBoost [27] as bagged and boosted tree ensembles, and an LSTM-only network [28] and a CNN-LSTM as the recurrent and convolutional-recurrent families. Table II gives every configuration. Hyperparameters were fixed in advance with no search, so the comparison is between default-configured rather than optimised models, a limitation restated in Section V-C. Both networks use Adam [29] at learning rate 1e-3 and dropout [30]; the CNN-LSTM adds batch normalisation [31].

TABLE II MODEL CONFIGURATIONS

Model	Configuration
Logistic Regression	solver lbfgs, max_iter 2000, class_weight None, seed 42
Random Forest	n_estimators 300, max_depth None, class_weight None, seed 42
XGBoost	n_estimators 300, max_depth 4, learning_rate 0.1, subsample 0.9, colsample_bytree 0.9, scale_pos_weight 1.0, seed 42
LSTM-only	LSTM(32) → Dropout(0.3) → Dense(16, ReLU) → Dense(1, sigmoid); 4,897 parameters; binary cross-entropy, batch 32, ≤ 60 epochs, early stopping on validation AUC, patience 8, best weights restored (40 epochs run)
CNN-LSTM	Conv1D(32, kernel 3, same, ReLU) → BatchNorm → LSTM(32) → Dropout(0.3) → Dense(16, ReLU) → Dense(1, sigmoid); 9,121 parameters; training configuration identical to LSTM-only (21 epochs run)

class_weight is None and scale_pos_weight is 1.0 because SMOTENC has already balanced the training partition; applying both would double-correct the imbalance.

The neural models require a three-dimensional input, so each customer's 30 encoded features are presented as a tensor of shape (30, 1). This is an ordered feature vector, not a time series: there is one row per customer, and the ordering is the arbitrary column order produced by the ColumnTransformer, so the recurrent layer has no genuine sequence to model. To test whether the ordering matters, the CNN-LSTM was retrained on two random permutations of the feature axis under a shortened budget of 20 epochs with patience 5; validation churn F1-scores were 0.6151 and 0.6337 (standard deviation 0.0093). That is consistent with little order-dependent structure being available for the recurrent layer to exploit.

D. Locking the Explain→Act Model

Explanations are properties of one fitted model, and choosing that model by test-set performance would make the test partition part of model selection. The choice was therefore made from training-derived evidence alone, before any test metric was computed: five-fold cross-validation on the resampled training partition for the classical models, and the held-out neural-validation subset for the networks. Random Forest scored highest on every column of that evidence (churn F1 0.8606, macro-F1 0.8567, PR-AUC 0.9165) and was locked as the Explain→Act model; Sections IV-C and IV-D explain that model and no other.

Two properties of this evidence must be stated. It is asymmetric by construction — cross-validation for the classical models against a single held-out subset for the

neural models — so the two evidence types are not comparable. And here the resampling is applied before cross-validation rather than inside each fold, so the absolute values are optimistic and only the ordering among the classical models is interpretable. The lock is a procedural device against test-set contamination, not a performance claim, and no value from it is reported as a result.

E. Explainability and Counterfactual Search

The locked model is explained with SHAP TreeExplainer [20] on the test partition. Because one-hot encoding splits a raw attribute across several indicator columns, attributions are reported twice: over the 30 encoded inputs, and aggregated by summing the mean absolute contributions of all indicators belonging to the same raw attribute. For action, each of the 19 raw predictors was classified before the search as actionable if the provider can change it directly, influenceable if it can be encouraged, conditional if it depends on other choices, or immutable otherwise. Five are actionable — Contract, Tech Support, Online Security, Online Backup and Device Protection — and the counterfactual search [22], [23] may alter only those, at most two per customer, requiring the predicted probability to fall below the fixed threshold $\tau = 0.50$. Immutable attributes such as Tenure Months are excluded by construction, which separates a feasible counterfactual from an arithmetically valid one.

F. Metrics, Reproducibility and Availability

Because churn prevalence is 26.54 %, accuracy alone is inadequate: predicting the majority class for every customer reaches 73.46 %. Results are reported as accuracy, precision, recall and F1-score for the churn class, macro-averaged F1, ROC-AUC and PR-AUC, with macro-F1 as the primary comparison metric because it weights both classes equally, and PR-AUC as the more informative area under imbalance because it ignores true negatives. All threshold-dependent metrics use $\tau = 0.50$, and no significance test was performed, so every ordering below is descriptive rather than inferential. Every stochastic component is seeded with random_state = 42: the split, the resamplers, the tree ensembles and the neural initialisation. The pipeline was executed in Python with scikit-learn [32], imbalanced-learn, XGBoost and TensorFlow/Keras [33]. The analysis code and experiment configuration are available from the authors upon reasonable request.

IV. RESULTS

TABLE III
PERFORMANCE ON THE LOCKED TEST PARTITION (N = 1,409), EVALUATED ONCE

Model	Acc.	Prec.	Rec.	F1	Macro F1	ROC-AUC	PR-AUC
Log. Regression	0.7431	0.5107	0.7647	0.6124	0.7101	0.8403	0.6331
Random Forest	0.7779	0.5735	0.6364	0.6033	0.7245	0.8280	0.6116
XGBoost	0.7651	0.5458	0.6845	0.6074	0.7199	0.8353	0.6366
LSTM-only	0.7608	0.5433	0.6203	0.5793	0.7061	0.8249	0.6098
CNN-LSTM	0.7672	0.5540	0.6310	0.5900	0.7137	0.8233	0.6071

Precision, recall and F1 are for the churn class. No value is marked as best, because no significance test was performed. All values use $\tau = 0.50$; confusion

counts are given in Section IV-B.

A. Test-Set Performance

Table III reports the single evaluation of the locked test partition. No model leads on every metric, and which one leads depends on what is being asked. Random Forest attains the highest accuracy (0.7779) and macro-F1 (0.7245); Logistic Regression the highest churn recall (0.7647), churn F1-score (0.6124) and ROC-AUC (0.8403); and XGBoost the highest PR-AUC (0.6366), the metric least distorted by the majority class. The CNN-LSTM leads on none of them, reaching accuracy 0.7672, churn F1-score 0.5900, macro-F1 0.7137, ROC-AUC 0.8233 and PR-AUC 0.6071, below all three classical models on the churn class while carrying the highest parameter count. The convolutional front end does help the recurrent model, raising accuracy from 0.7608 to 0.7672 and churn F1-score from 0.5793 to 0.5900, but neither neural model reaches the classical baselines. The spread across the five architectures is 0.0348 in accuracy and 0.0169 in ROC-AUC, small enough that without a significance test the ordering is descriptive of this single split rather than a ranking of architectures.

B. Error Structure and Training Behaviour

The two error types carry different costs: a false negative is a churning who receives no retention offer, a false positive an incentive extended to a customer who would have stayed. Test-partition confusion counts, as TN/FP/FN/TP with churn positive, are Log. Regression 761/274/88/286; Random Forest 858/177/136/238; XGBoost 822/213/118/256; LSTM-only 840/195/142/232; CNN-LSTM 845/190/138/236; every row reconciles with the partition at 1,035 non-churners and 374 churners. The CNN-LSTM misses 138 of the 374 actual churners, which is the practical meaning of its churn recall of 0.6310, while Logistic Regression recovers 286 churners at the cost of 84 additional false positives. Which trade-off is preferable depends on the relative cost of a missed churning and a wasted incentive, a ratio this study does not estimate.

The training histories are consistent with capacity exceeding the available signal. Early stopping on validation AUC halted the CNN-LSTM after 21 epochs and restored the best weights; its training loss continues to fall while validation loss stops improving and begins to rise after roughly the twelfth epoch. The LSTM-only model, trained identically, ran for 40 epochs. With 9,121 trainable parameters fitted to 30 features per customer, the CNN-LSTM has ample capacity to begin memorising the training partition before it exhausts the generalisable signal in static tabular attributes.

TABLE IV
GLOBAL SHAP IMPORTANCE: ENCODED INPUTS VERSUS AGGREGATED RAW FEATURES

	Encoded input	SHAP	Raw feature family	SHAP
1	Tenure Months	0.07089	Contract	0.08747
2	Dependents_Yes	0.06195	Tenure Months	0.07089
3	Contract_Two year	0.05891	Payment Method	0.06694
4	Payment Method_Electronic check	0.05249	Dependents	0.06195

5	Internet Service_Fiber optic	0.05095	Internet Service	0.05873
6	Total Charges	0.03519	Online Security	0.03898

Mean absolute SHAP values, top six of each ranking. The right-hand ranking sums the contributions of all one-hot indicators belonging to the same raw feature.

A. Global Attribution Under the Locked Model

SHAP TreeExplainer is used to determine the key drivers for the test partition, while Random Forest gets locked without reference to test metrics. Table IV reports importance at both a single indicator level and a level of the raw feature family represented by the indicators.

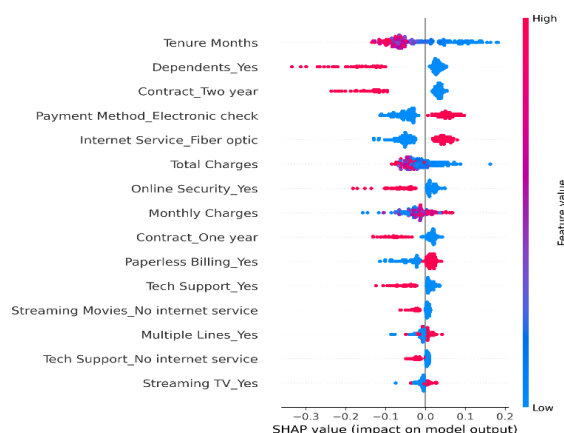


Fig.2. SHAP summary plot for the locked Random Forest model, showing the direction and magnitude of each feature's contribution to predicted churn probability.

B. Local Explanations and Feasible Counterfactuals

Three high-risk test customers were examined, at predicted churn probabilities of 0.9933, 0.9867, 0.9867; all three are actual churners, so the model is correct in each case. Tenure Months is the largest positive contributor for all three, at 0.1348, 0.1077 and 0.1786 respectively, and Total Charges is second for two of them, with electronic-check payment and fibre-optic internet service also contributing positively. Negative contributions are an order of magnitude smaller than the leading positive ones. These explanations are diagnostic but not prescriptive: the dominant driver in every case is an attribute nobody can change.

TABLE V
MINIMAL FEASIBLE COUNTERFACTUALS UNDER THE LOCKED MODEL

Case	p before	Minimal feasible change	n	p after	Δp
CF 1	1.000	Contract $\rightarrow$ Two year; Tech Support $\rightarrow$ Yes	2	0.473	-0.527
CF 2	1.000	Contract $\rightarrow$ Two year; Online Security $\rightarrow$ Yes	2	0.393	-0.607
CF 3	1.000	Contract $\rightarrow$ Two year; Online Security $\rightarrow$ Yes	2	0.473	-0.527

Probabilities are Random Forest outputs. Only the five features classified as actionable were permitted to change, with a maximum of two changes per case.

Table V reports the constrained search. Each case begins at a predicted probability of 1.000 and from an identical actionable profile: a month-to-month contract with tech support, online security, online backup and device protection all absent. In every case a change to exactly two actionable features moves the predicted probability below $\tau = 0.50$, the largest single reduction being 0.607. Three qualifications are necessary. The three cases share that profile and are therefore not independent evidence; the search reports only whether a feasible counterfactual exists for these customers, not what fraction of high-risk customers admit one. The narrowest crossing lands at 0.473, only 0.027 below the threshold, so that flip is not robust to a small shift in τ or to retraining under a different seed. And a counterfactual is a statement about the model's decision surface, not a prediction about the customer: establishing that a move to a two-year contract would actually reduce a customer's probability of leaving would require a controlled or uplift study.

V. DISCUSSION

A. Does the Convolutional-Recurrent Model Help?

On this data set, under this protocol, it does not (as shown in Fig.3.). The CNN-LSTM leads on none of accuracy, macro-F1, churn F1-score, ROC-AUC or PR-AUC, and it is the most expensive model in the comparison. The explanation is a property of the data rather than of the implementation: convolutional and recurrent layers earn their parameters by exploiting local and sequential structure, and a customer record has neither, being a fixed-length vector of unrelated attributes whose order is an artefact of the encoder — which the permutation check of Section III-C supports.

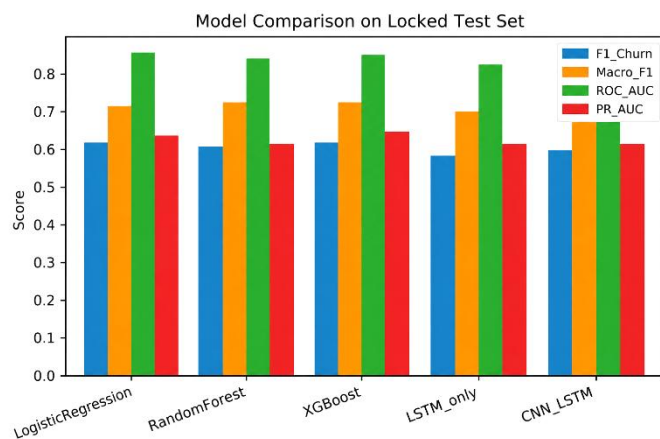


Fig.3. Bars are not the tallest for any of the five models benchmarked for the locked test partition: Churn-class F1, macro-F1, ROC-AUC, and PR-AUC for CNN-LSTM.

The finding is not that convolutional-recurrent architectures are weak, but that this representation gives them nothing to exploit while still charging for the capacity; a genuine test would require longitudinal usage, billing or support-contact sequences, which this data set does not contain. Two qualifications bound the claim: the comparison uses default configurations with no hyperparameter search, so a tuned

CNN-LSTM might close part of the gap; and the spread across the five models is small and untested, so the correct summary is the absence of a demonstrated advantage rather than a demonstrated deficit.

B. Relation to Published Accuracies

The accuracies here, in the 0.76 to 0.78 range, sit below many published figures for telecom churn, and the difference is informative. Several reported results resample the entire data set before partitioning, which places synthetic neighbours of test records into the training set and inflates every subsequent metric; others report accuracy on an artificially balanced evaluation set, where the majority-class baseline is 50 % rather than the 73.46 % that applies here; others select the reported configuration by test-set performance. This study excludes all three, and also removes Churn Score and CLTV, model-derived columns that a pipeline retaining them would find highly predictive of a target they may partly encode. The figures here should be read as what this protocol yields, not as a ceiling for the data set.

C. Limitations

Seven limitations bound the conclusions. The evaluation rests on a single stratified split and a single seed, so no variance estimate accompanies any value in Table III. No significance test was performed — a paired test such as McNemar's [34] on the test predictions would be the appropriate next step — so the orderings are descriptive and the small spreads should not be read as differences between architectures. Hyperparameters were fixed in advance with no search, so the comparison is between default configurations. In the model-selection and feature-selection comparisons the resampling is applied before cross-validation rather than inside each fold, so those absolute values are optimistic and only the differences are interpretable; and the Explain→Act evidence is additionally asymmetric between the classical and neural models. The counterfactual analysis covers three non-independent cases sharing one actionable profile, one of which crosses the threshold by only 0.027, and it establishes existence rather than any feasibility rate; it also describes the model's decision surface and carries no causal claim about customer behaviour. Finally, the data set is a single static snapshot from one provider, so neither the ranking nor the attribution should be assumed to transfer to another operator or to a later period.

VI. CONCLUSION

This study benchmarked five models on the 33-column IBM Telco customer churn data set under one leakage-controlled protocol: thirteen identifier, geographic and target-related columns removed, an 80/20 stratified split before every fitted transformation, the imbalance and feature-selection decisions taken by cross-validation on the training partition against rules fixed in advance, and the test partition evaluated once. SMOTENC was selected over class weighting by 0.0123 mean macro-F1 with the accompanying PR-AUC cost recorded, recursive feature elimination was rejected at $\Delta\text{macro-F1} = +0.0003$, and Random Forest was locked as the Explain→Act model from training-only evidence. On the locked partition of

1,409 customers the CNN-LSTM led on no metric, so the benchmark found no consistent advantage for the convolutional-recurrent architecture on static tabular customer data, and the permutation check indicates why: an encoded feature vector holds no order-dependent structure for a recurrent layer to exploit. Because no significance test was performed, the orderings in Table III are descriptive of one split rather than inferential.

The procedural contribution is the part that transfers. A faithful explanation identifies what the model uses, an actionable one what a provider can change, and the two differ here because the dominant driver is immutable. Locking the explained model on training-only evidence, aggregating SHAP attributions from one-hot indicators back to raw feature families, and constraining counterfactual search to attributes a provider can actually change together turn the first into the second, and none of the three depends on the model family. Aggregation moved Contract ahead of Tenure Months in the global ranking, and two permitted service changes moved each of three high-risk customers below $\tau = 0.50$. Whether such a change would alter real behaviour is a causal question for a controlled or uplift study. The clearest extensions are repeated splits with paired significance testing, a cost-weighted rather than threshold-fixed decision rule, and evaluation on longitudinal per-customer sequences.

REFERENCES

- [1] J.-H. Ahn, S.-P. Han, and Y.-S. Lee, "Customer Churn Analysis: Churn Determinants and Mediation Effects of Partial Defection in the Korean Mobile Telecommunications Service Industry," *Telecommunications Policy*, vol. 30, nos. 10–11, pp. 552–568, 2006, doi: 10.1016/j.telpol.2006.09.006.
- [2] S.-Y. Hung, D. C. Yen, and H.-Y. Wang, "Applying Data Mining to Telecom Churn Management," *Expert Systems with Applications*, vol. 31, no. 3, pp. 515–524, 2006.
- [3] W. Verbeke, D. Martens, and B. Baesens, "Social Network Analysis for Customer Churn Prediction," *Applied Soft Computing*, vol. 14, pp. 431–446, 2014, doi: 10.1016/j.asoc.2013.09.017.
- [4] De Caigny, K. Coussemont, and K. W. De Bock, "A New Hybrid Classification Algorithm for Customer Churn Prediction Based on Logistic Regression and Decision Trees," *European Journal of Operational Research*, vol. 269, no. 2, pp. 760–772, 2018, doi: 10.1016/j.ejor.2018.02.009.
- [5] Y. Zhang, J. Qi, H. Shu, and J. Cao, "A Hybrid KNN-LR Classifier and Its Application in Customer Churn Prediction," in *Proc. IEEE Int. Conf. Systems, Man and Cybernetics (SMC)*, Montreal, Canada, 2007, pp. 3265–3269, doi: 10.1109/ICSMC.2007.4414197.
- [6] K. Ahmad, A. Jafar, and K. Aljoumaa, "Customer Churn Prediction in Telecom Using Machine Learning in Big Data Platform," *Journal of Big Data*, vol. 6, art. 28, 2019, doi: 10.1186/s40537-019-0191-6.
- [7] T. Vafeiadis, K. I. Diamantaras, G. Sarigiannidis, and K. Ch. 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.
- [8] Y. Huang and T. Kechadi, "An Effective Hybrid Learning System for Telecommunication Churn Prediction," *Expert Systems with Applications*, vol. 40, no. 14, pp. 5635–5647, 2013, doi: 10.1016/j.eswa.2013.04.020.
- [9] L. Çallı and S. Kasım, "Using Machine Learning Algorithms to Analyze Customer Churn in the Software as a Service (SaaS) Industry," *Academic Platform Journal of Engineering and Smart Systems*, vol. 10, no. 3, pp. 115–123, 2022, doi: 10.21541/apjess.1139862.
- [10] H. Karaarslan, M. Baştuğ, C. Güngör Şen, and E. E. Işık, "A Comparative Study on Customer Churn Analysis Using Machine Learning and Data Enrichment Techniques," *Journal of Soft Computing and Decision Analytics*, vol. 2, no. 1, pp. 225–235, 2024, doi: 10.31181/jscda.21202441.
- [11] S. Saha, C. Saha, M. M. Haque, M. G. R. Alam, and A. Talukder, "ChurnNet: Deep Learning Enhanced Customer Churn Prediction in Telecommunication Industry," *IEEE Access*, vol. 12, pp. 4471–4484, 2024, doi: 10.1109/ACCESS.2024.3349950.
- [12] Thenmozhi, C. Jeyabharathi, and S. Vimala, "Customer Churn Prediction Analysis in the Banking Sector Using Machine Learning," *IAPQR Transactions*, vol. 49, no. 2, pp. 106–117, 2024.
- [13] Khattak, Z. Mehak, H. Ahmad, M. U. Asghar, M. Z. Asghar, and A. Khan, "Customer Churn Prediction Using Composite Deep Learning Technique," *Scientific Reports*, vol. 13, no. 1, art. 17294, 2023, doi: 10.1038/s41598-023-44396-w.
- [14] Y. Elyusufi and M. Ait Kbir, "Churn Prediction Analysis by Combining Machine Learning Algorithms and Best Features Exploration," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 7, pp. 615–622, 2022, doi: 10.14569/IJACSA.2022.0130773.
- [15] L. Feng, "Research on Customer Churn Intelligent Prediction Model Based on Borderline-SMOTE and Random Forest," in *Proc. IEEE 4th Int. Conf. Power, Intelligent Computing and Systems (ICPICS)*, 2022, pp. 803–807, doi: 10.1109/ICPICS55264.2022.9873702.
- [16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, doi: 10.1613/jair.953.
- [17] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017.
- [18] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4765–4774.
- [19] S. M. Lundberg et al., "From Local Explanations to Global Understanding with Explainable AI for Trees," *Nature Machine Intelligence*, vol. 2, no. 1, pp. 56–67, 2020, doi: 10.1038/s42256-019-0138-9.
- [20] 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.
- [21] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR," *Harvard Journal of Law and Technology*, vol. 31, no. 2, pp. 841–887, 2018.
- [22] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining Machine Learning Classifiers Through Diverse Counterfactual Explanations," in *Proc. ACM Conf. Fairness, Accountability, and Transparency*, 2020, pp. 607–617, doi: 10.1145/3351095.3372850.
- [23] IBM, "Telco Customer Churn," IBM Cognos Analytics sample data set, IBM Corporation.
- [24] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene Selection for Cancer Classification Using Support Vector Machines," *Machine Learning*, vol. 46, nos. 1–3, pp. 389–422, 2002.
- [25] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- [26] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 785–794.
- [27] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
- [28] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
- [30] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proc. 32nd Int. Conf. Machine Learning (ICML)*, 2015, pp. 448–456.

- [31] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [32] M. Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning,” in *Proc. 12th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.
- [33] Q. McNemar, “Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages,” *Psychometrika*, vol. 12, no. 2, pp. 153–157, 1947, doi: 10.1007/BF02295996.