Machine Learning Models for Customer Churn Risk Prediction

Oguzhan Akan
Department of Computer Science
California State University
Northridge, CA, US
oguzhan.akan.95@my.csun.edu

Abhishek Verma
Department of Computer Science
California State University
Northridge, CA, US
abhishek.verma@csun.edu

Abstract— **Customer churn analysis in the industry is an important area of research due to its effect on profitability of business, measuring customer satisfaction, figuring out product promotions, and creating marketing strategies. In this paper we predict the possibility of churn of a given customer by advanced machine learning pipelines. In particular we perform a comprehensive comparison of feature selection methods, oversampling methods, and three machine learning methods the Logistic Regression, Random Forest and Deep Neural Networks on customer churn dataset. The conclusions we reach include (1) Information Value features performed significantly better than Principal Component Analysis features, (2) data oversampling provides model consistency, and (3) Random Forest is superior to Logistic Regression and Deep Neural Networks in the datasets where the number of instances is few.**

Keywords*— *deep neural networks, random forest, logistic regression, smote, adasyn, information value analysis

I. Introduction

One of the primary concerns of the corporations (e.g., telecommunications, banking, or in general service sector) include customer churn and is reported directly to managing board and CEOs in most companies. Formerly expert-knowledge rulesets were used for deciding customers who are likely to churn, currently Machine Learning (ML) algorithms have taken its place. Logistic Regression (LR) and Random Forests (RFs) are being used to able to measure the probability of churn while more recently Deep Neural Networks have increasingly become popular.

This research aims to provide an end-to-end machine learning modeling pipeline and analyze model performance based on:

- **Feature selection techniques.** Reducing the dimensionality of the dataset with a predefined procedure. Dimensionality reduction is used for purposes such as increasing time efficiency for running ML algorithms or overcoming possible overfitting issues pre-emptively during modeling stage. This research applies dimensionality reduction with Information Value (IV) technique and Principal Component Analysis (PCA and reports the resulting ML models' performance.
- **Data upsampling methods.** When dataset is highly unbalanced, it is generally a good idea to synthetically balance the target class so that the ML algorithms fit their models to the feature space more precisely. This research investigates (i) whether upsampling is superior compared to no-upsampling, and (ii) between Synthetic Minority Oversampling Technique (SMOTE) and Adaptive Synthetic Sampling (ADASYN), which one produces better results in terms of performance metric, Area Under the Receiver Operating Characteristics Curve (AUROC) score.
- **Machine Learning algorithms.** As the main part of study, we evaluate how Deep Neural Networks (DNN) perform relative to LR and RF. The algorithms used in this research are described in section IV.

Rest of the paper is organized as follows: section II presents related work, section III describes dataset used in this research, section IV describes methodology, i.e., various preprocessing and machine learning algorithms used in this paper, experimental results are discussed in section V followed by conclusion in section VI.

II. Related Work

Although, there are several studies conducted for predicting churn risk, there are only a few formal research studies done on DNNs performance on churn problems. Mena et al. (2019) [1] used LSTM to predict churn on sequential data. Since the data is sequential, Recursive Neural Networks (RNNs) were used in the problem. They conclude that LSTM generates a greater increase in performance compared to the model built with Logistic Regression with a set of static features by 25%. This research provides a benchmark for this paper regarding the performance of DNN over Logistic Regression.

Vafeiadis et al. (2015) [2] compares various Machine Learning algorithms in their study, aiming to predict customer churn risk. The algorithms used in the study includes Naïve Bayes, Support Vector Machines, Decision Trees, Logistic Regression, Adaptive Boosting, and Neural Networks. This research is comprehensive for its time as it compares numerous Machine Learning algorithms, however, due the rapid new developments in the ML area several new approaches have emerged since then. For instance the XGBoost algorithm was introduced in 2016. Additionally, the Neural Networks were limited to simple back propagation network or BPN, and did not use improved Sequential Neural

Networks. They concluded that Support Vector Machine (SVM) with polynomial kernel, boosted with AdaBoost, produces the best result in terms of F-1 score.

A more recent example of predicting churn risk using ML algorithms was conducted by Ahmad et al. in 2019 [3], where a telecommunication company's dataset was used to build churn prediction models with Decision Tree, Random Forest, Gradient Boosting, and XGBoost algorithms. The paper concludes that XGBoost's performance is 2.5% better in terms of AUROC score than the traditional Gradient Boosting algorithm, which is the closest to best algorithm after XGBoost [3]. Their work provides a good example of algorithm comparison albeit it only investigates tree-based algorithms.

One of the latest research projects by Lalwani et al. (2022) [4] on predicting churn risk compares several ML algorithms for data from a telecommunications company. Their research generates ML models based on statistical algorithms such as Naïve Bayes, likelihood-based algorithms such as Logistic Regression, tree-based algorithms, and ensemble algorithms such as Random Forest, XGBoost and Adaboost with some variations – for example, using different kernels for Support Vector Machines. They conclude that XGBoost and Adaboost produces the best performance with no significant difference [4].

III. Dataset Description

HackerEarth customer churn risk dataset [16] of a business that provide products and services online was chosen for this research. The dataset includes a variety of demographic and aggregated transaction information for training purposes and a target feature indicating whether the customer is at risk of churn or not churn. The complete dataset consists of 36,992 data points together with 19 training features and a target feature. Since the dataset's target is churn risk of a given customer, it is inherently imbalanced. There are only 5,393 true labels (churn) assigned for the binary target feature which translates to a small subset of full dataset. This property of the dataset impacted our decision on reshaping training data and selecting model evaluation metrics. The dataset required a preprocessing step that consists of cleaning missing or incorrectly populated data, encoding categorical features into ready-to-use features and up sample training data to overcome class imbalance problem. The preprocessing steps are described later in section IV of this paper. The overview of variables is shown in Table I to provide some insight into the dataset.

TABLE I. HACKEREARTH CUSTOMER CHURN RISK DATASET OVERVIEW

Feature	Mean	Min	25%	75%	Max
age	37.1	10.0	23.0	51.0	64.0
days_since_last_log	-43.0	-999.0	8.0	16.0	26.0
avg_time_spent	292.6	1.8	71.6	371.2	3040.4
avg_transaction	29.3	0.8	14.2	40.8	99.9
avg_frequency_login	16.0	-43.7	9.0	23.0	73.1
points_in_wallet	687.0	-760.7	615.9	763.9	2069.1
joining_year	2016.0	2015.0	2015.0	2017.0	2017.0

IV. Methodology

A. Logistic Regression (LR)

Logistic Regression is a linear discriminant model that attempts to maximize the likelihood function, which is binary cross-entropy using the sigmoid function. The name regression comes from the update function where weights of coefficients are updated by the derivative function. It is designed to classify categories. Since sigmoid function is inherently nonlinear, gradient descent is used to minimize cross-entropy, which is the equivalent of maximizing likelihood or log likelihood [5]. Let $X = \{x^t,\ r^t\}$ where $r^t = 1$ if $x \in C_1$ meaning the customer has churned and $r^t = 0$ if $x \in C_0$, or in other words, no churn. The r^t is assumed to be Bernoulli with probability $y^t \equiv P(C_1|x^t)$, which is the sigmoid function:

$$P(C_1|x) = \frac{1}{1+e^{-w^T+w_0}} \quad (1)$$

The aim is to learn w and w_0 . Since $(r^t|x^t) \sim Bernoulli(y^t)$, the likelihood function becomes $l(w, w_0|X) = \prod_t (y^t)^{r^t}(1-y^t)^{(1-r^t)}$. Then, instead of trying to maximize this function, it is converted to an error function that is to be minimized. The error function can be defined as $E = -log l$, the binary cross-entropy given as:

$$E(w, w_0|X) = -\sum_t r^t \, log \, y^t + (1-r^t) log(1-y^t) \quad (2)$$

The updated equations, Δw_j and Δw_0 are calculated by taking derivative of cross-entropy function with respect to w_j and w_0. The logistic regression algorithm starts with a small value assigned to w_j, usually a random number drawn from uniform distribution $[-0.01, 0.01]$ and repeats calculating cross-entropy and updates functions until convergence [5]. Convergence may be determined by a predefined number of iterations or a minimum required amount of improvement on the error. This research uses a maximum of 1,000 iterations or stopping early if there is no improvement toward convergence.

B. Random Forest (RF)

Random Forest is an ensemble model that uses bagging to generate a prespecified number, say L, of base estimators, in this case, decision trees, and take mean of the predictions made by each base estimator to produce final predictions [5]. Bagging is a method where L slightly different samples are generated using bootstrapping that are drawn from the initial dataset [5]. The bootstrapping sampling is done by randomly selecting data points from the initial dataset with a size of N with replacement [5].

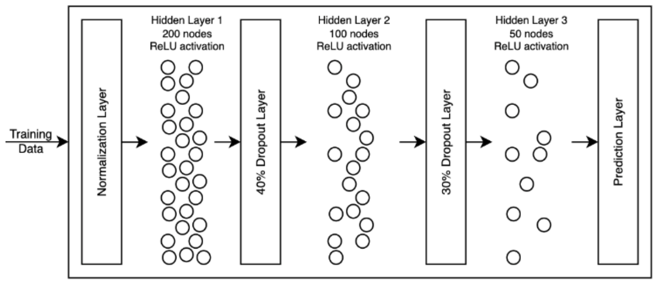

Figure 1. The visualization of DNN-Model structure used in this research. The 40% and 30% dropout layers are added after hidden layer to overcome overfitting problem. Learning decay is applied over epoch for a better loss convergence.

C. *Deep Neural Network (DNN)*

DNNs are complex Machine Learning algorithms that incorporate multiple hidden layers where each layer becomes a input for the subsequent layer, ultimately producing classification probabilities or regression values in the last prediction layer. One aspect of DNNs is that successive hidden layers learn to understand local and global features that are present in the raw data, enabling the neural network to come up with complex decision boundaries to be able to make accurate predictions. Thus, the main idea behind DNNs is minimizing human contribution by learning feature at different levels of abstraction [5]. An example of a DNN is shown in Figure 1. As figure suggests, a DNN may have multiple hidden layers with each having different number of neurons. Each layer's activation function determines how a single neuron will be activated for a given input instance and lastly the prediction layer.

A common issue with DNNs is the convergence problem. The DNNs are more susceptible to overfitting or divergence. Overcoming the overfitting problem may be achieved by decreasing model complexity by discarding some connections by adding a dropout layer between dense layers so that the connections causing overfitting are eliminated. As can be seen from Figure 1, the dropout layers are introduced in DNN model. Dropout helped our model to converge in 50 epochs. Another way we used is to use learning rate decay by 0.1 factor, it helped to find the local optima more precisely. The reduction factor could be set to higher levels to keep learning rate more conservative.

D. *Data Preprocessing and Feature Selection*

1) *Handling Missing or Incorrect Values*

Firstly, the values for $avg\_frequency\_login\_days$ that are less than zero are replaced by zero. The reasoning behind this approach is that these values are somehow populated but with incorrect numbers, so the number zero is picked as the closest non-negative integer. On the other hand, the data points for $days\_since\_last\_login$ where the value was less than zero were discarded because it was indicated in data description that these customers are already churned. Including those data points would cause noise in the model. Secondly, 9.48% of the values for $avg\_frequency\_login\_days$ feature was missing. Since Downey and King (1998) [6] suggest in their study that populating missing values by category means is a good approach in features, which have less than 20% missing values. Thus, this approach was taken to finalize the processing of this feature. After this step, the dataset is cleaned and it has 30,977 data points.

2) *Encoding of Categorical Features*

Three widely known coding algorithms were used to convert our categorical features into numerical features, namely (1) Ordinal Encoding, (2) Binary Encoding, and (3) One-Hot Encoding. Both Binary Encoding and One-Hot Encoding algorithms are optimized in Python's Category Encoders module written by McGinnis in a way that it automatically excludes the newly generated features which have no variation [7]. This feature is extremely helpful for RAM processing and enabled us to run hundreds of models without high memory footprint. Lastly, Category Encoders provide various encoding algorithms, even some with statistical approaches, albeit since our primary goal is to evaluate and compare performances of different Machine Learning algorithms and Deep Neural Networks, they are considered as out of scope for this research.

a) *Ordinal Encoding*

Ordinal Encoding is the process of converting non-numerical data into numerical data by assigning an integer value to each category. The order can be prespecified or not, depending on the existence of ordinality. A drawback of this approach is that it assumes an order and is blind to new categories [8]. On the other hand, Ordinal Encoding does not change data cardinality, meaning that only one feature can capture the whole category information - which is beneficial for highly volatile features. The encoded features are represented with an "OE_" prefix in the rest of this research. Ordinal Encoding produced 12 new features as expected.

b) *Binary Encoding*

Binary Encoding is another encoding method that represents categories in binary format. The categories are initially converted to integers (starting from zero) then converted to their binary formats. For example, a categorical feature having 4 unique categories would be first assigned to integers 0, 1, 2, and 3. Then, the data points that are category 0 are given 00, 1 are given 01, 2 are given 10 and 3 are given 11. Therefore, Binary Encoding produces $\log_2(d)$ new columns for d unique values in a categorical feature [9]. The encoded features are represented with a "BE_" prefix in the rest of this research. Ultimately, Binary Encoding produced 41 new features for the customer churn dataset.

c) *One-Hot Encoding*

One-Hot Encoding is an encoding method that is somewhat like Binary Encoding. The main difference in One-Hot Encoding is that it solely instantiates an extra feature per category rather than incorporating features into binary versions. Therefore, a major drawback arises in One-Hot Encoding as it produces as many features as the number of unique values for a categorical variable [10]. The One-Hot encoded features are still included in our training set as it may have more informative value regarding other encoding algorithms. One-Hot Encoding produced 38 new features for out dataset, making the total number of features generated from OE, BE, and OH 91.

3) *Splitting Training and Test Data*

One of the crucial steps for building a successful algorithm is having a training and test set that is representative of the entire dataset. Stratified data splitting balances the true class label ratio in both sets without harming randomness [10]. The Scikit-learn's [17] stratified splitting method was used to keep the true class samples close to each other in training and test sets (15.08% true class samples in training and 15.07% in test). Also, 20% data was chosen as test split and rest for training. After splitting training and test sets, the training set consists of 24,781 data points and test consists of 6,196 data points, which we found in our experiments as adequate for evaluating ML model performance.

4) *Feature Selection Approaches*

After splitting the cleaned data into training and test, training data is further processed to create datasets consisting of the features selected by (1) Information Value technique, and (2) Principal Component Analysis. Finally, the rest of the experimental study is conducted on those two preprocessed datasets from two feature selection methods and results are compared to understand the effect of feature selection.

a) Information Value Analysis

Information Value (IV) is one of the approaches for exploratory data analysis that enables one to assess the individual impact of a given factor on the target feature and is applicable to both categorical and continuous features [11]. Weight of Evidence (WOE) is used to calculate IV, which is the natural logarithm of ratio of distributions of non-events to events. In our research, events correspond to churned customers, which are represented as integer 1 in the dataset and non-events are the retained customers represented as integer 0. In other words, WOE is calculated as:

$$\mathrm{WOE}_i = \lg\left(\frac{\%non-events_i}{\%events_i}\right) \quad (3)$$

where i represents each category (or bin) in the independent variable. Once we calculate all WOE's, the IV, which is the overall assessment metric, is calculated by:

$$\mathrm{IV} = \sum_{i=1}^{N}\left((\%\mathrm{non-events_i}) - (\%\mathrm{events_i})\right) \cdot W\mathrm{OE_i} \quad (4)$$

where N is the number of categories (or bins) that exist in the given variable [11]. Conventionally, an IV with less than 0.02 is considered non-informative and therefore should not be included in the modeling stage. On the other hand, an IV greater than 0.5 could cause overfitting - or is too good to be true. Out of 91 categorical features generated with category encoders, only 33 features had an IV score greater than 0.02. For Categorical Features, the IV formula was applied using encoded features. Since, Ordinal Encoding captures the categorical features in their raw form, we performed comprehensive analysis, for each categorical feature, a bubble graph that shows the population size and target distribution per category was drawn.

For continuous features, IV calculation requires an extra step called *binning*. This step can be understood as converting continuous features to categorical features and it is done only to calculate the IV and is useful toward feature selection. In the modeling stage, the raw forms of continuous features are used. The binning process can be summarized as follows:

- Selecting the number of bins. The number of bins were set to 10 to be able to have enough data points in each bin.
- Capping values within the feature that is greater than 95th percentile to the 95th percentile, and values less than 5th percentile to 5th percentile.
- Sorting the values in an ascending order and assigning bin number to each data point.
- Treating each bin number as a category and applying IV calculation.

Out of 7 continuous features, 5 features have an IV value that is greater than 0.02, making the total number of features as 38.

Dropping Highly Correlated Features. As the last step of IV feature selection, the features are first ranked according to their IV values in descending order. Then, for every feature pair (f_i, f_j), the correlation between pairs is calculated. If the correlation is greater than 95%, the feature with less significance is dropped from the dataset. After dropping highly correlated features, the final training set from the IV approach has 31 features in total (26 categorical and 5 continuous).

b) Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised method meaning that it has no interest in predicting a target variable, but rather its main goal is to maximize the variance with a reduced number of features. PCA algorithm takes the eigenvectors calculated from the normalized dataset and selects the eigenvector with the highest eigenvalue, that is the direction on which the data is most spread out [5]. As Rencher [12] exemplifies the superiority of PCA over plain mean or median calculations by considering a class of students with five courses. Who is the best student? One may decide that the student with the highest average grade is the best, but in fact, he/she may not be [12]. PCA takes the variances and correlations into account, and ultimately choses the student who is on the most divergent point of the eigenvector [5]. Thus, in our research the PCA method was applied to 91 features generated from category encoding. Scikit-learn's PCA module allows us to choose a minimum variance required in the transformed dataset. A minimum of 95% required variance was chosen in the resulting dataset so that the loss in informative components is minimized. After applying PCA, the final features in the dataset reduced to 40. Notice that the number of features is still greater than the dataset generated with IV approach.

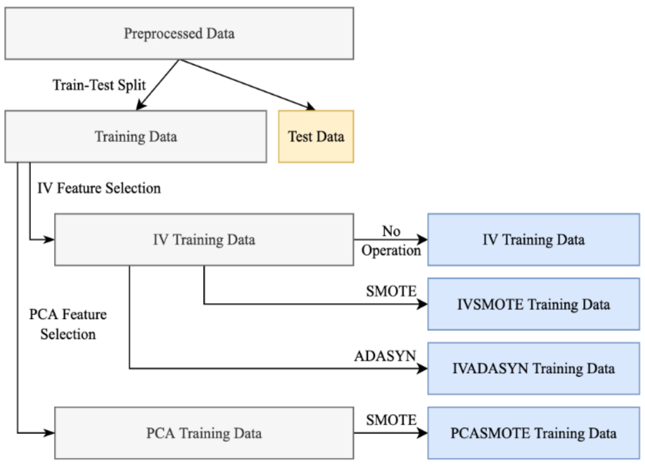

Figure 2. Training data generation from preprocessed data, described in sections IV.D.3) and IV.D.4).

5) Oversampling the Training Data

Different training sets are created with oversampling methods. These sets are abbreviated as:

(1) IVR: The dataset created by IV feature selection and no oversampling
(2) IVS: The dataset created by IV feature selection and SMOTE oversampling
(3) IVA: The dataset created by IV feature selection and ADASYN oversampling
(4) PCAS: The dataset created by PCA feature selection and SMOTE oversampling

a) Synthetic Minority Oversampling Technique (SMOTE)

SMOTE has been a widely used up sampling method since it was introduced by Chawla et al. [14]. The algorithm aims to generate pseudo-random data points in the feature space by repeating the following steps:

1. Select a random instance of the minority class

TABLE II: CLASSIFICATION PERFORMANCE OF VARIOUS MACHINE LEARNING METHODS ON DIFFERENT FEATURE SELECTION AND SAMPLING TECHNIQUES ON THE CHURN PREDICTION DATASET

	IVR			**IVS**			**IVA**			**PCAS**		
Metric	***LR***	***RF***	***DNN***	***LR***	***RF***	***DNN***	***LR***	***RF***	***DNN***	***LR***	***RF***	***DNN***
Accuracy	89.0%	87.1%	85.1%	83.4%	86.3%	84.8%	80.9%	85.7%	84.9%	82.5%	82.4%	84.8%
Precision	71.2%	54.4%	50.3%	46.5%	52.8%	49.8%	42.0%	51.6%	50.0%	45.7%	45.5%	49.8%
FPR	3.3%	13.1%	15.1%	13.8%	13.7%	15.5%	17.3%	14.5%	15.6%	18.0%	17.9%	15.3%
TPR	45.6%	88.0%	85.7%	67.5%	86.5%	86.7%	70.4%	86.7%	87.7%	85.5%	84.2%	85.4%
F-1	55.6%	67.2%	63.3%	55.1%	65.6%	63.3%	52.6%	64.7%	63.7%	59.6%	59.1%	62.9%
AUROC	71.2%	87.5%	85.3%	76.8%	86.4%	85.6%	76.6%	86.1%	86.1%	83.8%	83.1%	85.1%
FPR is the false positive rate and TPR is the true positive rate.												

2. Create a synthetic instance by:
3. Selecting K neighbors (K=5 in this research)
4. Choosing a random neighbor among K neighbors
5. Randomly select a point between the linear distance from source point to the neighbor
6. Repeat steps until $n_{minority} = n_{majority}$

b) Adaptive Synthetic Sampling (ADASYN)

ADASYN algorithm aims to balance the minority class by generating synthetic instances by assigning a weight to each minority class instance, which represents their level of difficulty to learn. Although, the step to create the instance is same as in the SMOTE algorithm, ADASYN assigns the number of synthesized instances generated for each data point, which is calculated from the given weights [13]. The weights are calculated by a density distribution:

$$\hat{r_i} = r_i / \sum_{i=1}^{m_s} r_i \qquad (5)$$

where $\hat{r_i}$ is the density distribution for minority class member i, and

$$r_i = \Delta_i / K \qquad (6)$$

where K is the number of neighbors (constant) and Δ_i is the number of majority class members among K neighbors. Lastly, the number of instances required for a minority class member i is:

$$g_i = \hat{r_i} \times G \qquad (7)$$

where $G = (m_l - m_s) \times \beta$. Here, the beta is a balancing coefficient that ranges between 0 and 1. 1 means full balance and 0 is no balance. m_l and m_s are the number of majority and minority class members, respectively. He et al. [15] concludes that full balance ($\beta = 1$) produces minimum error in classification.

V. Experimental Setup, Results and Model Evaluation

The modeling approach followed in this research is shown in Figure 2. The datasets using various feature selection approaches and oversampling techniques are generated while the test set is always kept separate. This approach guarantees that the scores generated for the test set are unaffected by the feature selection or oversampling steps.

Ultimately, in our research we perform three main comparisons and analyze model performance:

1. The effect of IV and PCA feature selection
2. Compare SMOTE and ADASYN oversampling over raw data
3. Compare Machine Learning algorithm: LR, RF and DNN.

A. *Selecting Optimal Cutoff Point for Class Assignment*

The optimal threshold for deciding the minority class (churned customers) is selected by calculating TP-Rate and FP-Rate for every possible threshold. Then, for each threshold, the AUROC is calculated. The threshold that maximizes AUROC is concluded as optimal. In other words, the intersection of TP-Rate and FP-Rate is the optimal threshold for a given model. Figure 3 is a plot to visualize optimal threshold via inverse ROC curve.

B. *Information Value Features and No Oversampling Models (IVR)*

The IVR dataset is generated from features selected by IV method with no oversampling and the ML algorithms described in Section III were run on this dataset. After optimizing the parameters, a 2% threshold has been put in place for the difference between train and test AUROC scores to prevent overfitting. Finally, the confusion matrix metrices are calculated and shared in Table II. As the table suggests, the RF-IVR model has the best AUROC score with 87.5%. The DNN model for IVR was inferior as it could not converge with the given parameter set. This also shows that the tree-based models, especially RFs require less parameter tuning and are more flexible with the input data.

It is also worthwhile to discuss other metrics produced. For example, LR seems to have the best precision and specificity, however, since recall is very low, the resulting F-1 score and AUROC is also lower than rest of the methods. This is a great example of how performance metrics could mislead where only one aspect of the results is considered in isolation.

C. *Information Value Features and SMOTE Oversampling Models (IVS)*

The SMOTE models with features selected from IV method is shared in Table II. Like the IVR model set, RF produced best results albeit they are lower than their

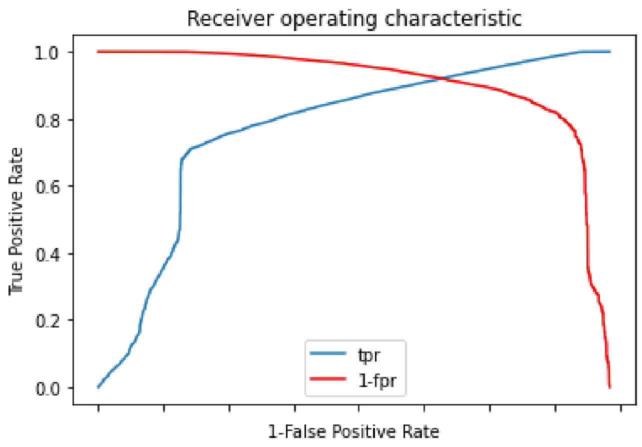

Figure 3. The inverse ROC curve plot to decide optimum threshold in order to maximize AUROC score. This example was plotted for IVR-DNN. TPR is the true positive rate and FPR is the false positive rate.

counterparts in IVR models with a slight delta. For example, AUROC for RF-IVS is 1.1% less than RF-IVR.

D. *Information Value Features and ADASYN Oversampling Models (IVA)*

The models built on ADASYN dataset were slightly better than SMOTE counterparts, but they are still inferior to IVR dataset. The difference is low with a 1.4% between AUROC scores of RF-IVA and RF-IVR. A side note for DNNs would be the lack of data points cause a lower activation in network layers, therefore resulting in a weaker model than the ensemble models we presented.

E. *Principal Component Analysis Features and SMOTE Oversampling Models (PCAS)*

Test set results of PCAS model set are shared in Table II. Compared to other experiments presented in section 4, the results are inferior across all models. The primary reason of the decrease in the performance is the 5% loss in total variance during PCA dimensionality reduction. The 5% reduction in variance translated into an almost 4% decrease in AUROC score for the best model, which is DNN. This experiment shows that DNN is helpful in extracting information when there are more features – an increase in the number of features by 9 for PCA dataset compared to IV dataset made DNN superior over other algorithms.

A general conclusion on PCAS model set would be that the models have tendency to predict true more than the conservative IV model sets. Of course, the threshold is set to optimize FP-Rate and TP-Rate simultaneously, but the flexibility on predicting true results in a lower AUROC score across all models. The DNN-PCAS model predicts 1577 data points as true while the corresponding DNN-IVR model only predicts 1374 data points as true. Therefore, the PCA method could result in lower TP-Rate.

VI. Conclusion

We provided a comprehensive comparison of feature selection methods, oversampling methods, and several machine learning models. First conclusion drawn from the experiments is that the oversampling methods such as SMOTE and ADASYN helped stabilize the model performance by providing consistent results across various metrics. As Table II suggests, most of the ML algorithms performed best in no-oversampling dataset. The only significant improvement on model performance was for LR in PCAS dataset where AUROC score has improved 12.6% compared to IVR dataset. Second conclusion can be drawn from Table II, which is that the IV feature selection method outperforms PCA feature selection. Since PCA is mainly a dimensionality reduction method, it is expected that there may be some information loss. We plan to perform a wider study of IV feature selection method along with various deep neural network approaches in our future research.

References

[1] C. Mena, A. Caigny, K. Coussement, K. De Bock and S. Lessmann, "Churn Prediction with Sequential Data and Deep Neural Networks," School of Business and Economics, Humboldt University of Berlin. IÉSEG School of Management, 2019.

[2] T. Vafeiadis, D. KI, G. Sarigiannidis and K. Chatzisavvas, "A comparison of machine learning techniques for customer churn prediction," Simulation Modelling Practice and Theory, pp. 1-9, 2015.

[3] A. Ahmad, A. Jafar and K. Aljoumaa, "Customer churn prediction in telecom using machine learning in big data platform," Journal of Big Data, vol. 6, no. 28, 2019.

[4] P. Lalwani, M. Mishra, J. Chadha and P. Sethi, "Customer churn prediction system: a machine learning approach," Computing, vol. 104, pp. 1-24, 2022.

[5] E. Alpaydin, Introduction to Machine Learning, Cambridge, Massachussets: The MIT Press, 2014.

[6] D. RG and K. C, "Missing data in Likert ratings: A comparison of replacement methods," J Gen Psychol Apr;125(2), pp. 175-91, 1998.

[7] W. McGinnis, T. W. Hbghhy, Andrethrill, C. Siu, C. Davison and N. Bollweg, "Scikit-learn-contrib/categorical-encoding: Release for Zenodo," Zenodo, 2018.

[8] K. Potdar, T. Pardawala and C. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," International Journal of Computer Applications 175(4), pp. 7-9, 2017.

[9] C. Seger, "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing," 2018.

[10] F. Vilarino, P. Spyridonos, J. Vitria and P. Radeva, "Experiments with SVM and stratified sampling with an imbalanced problem: Detection of intestinal contractions," Pattern Recognition and Image Analysis, pp. 783-791, 2005.

[11] P. Verma, "Churn Prediction for Savings Bank Customers: A Machine Learning Approach," Journal of Statistics Applications & Probability vol. 9, no. 3, pp. 535-547, 2020.

[12] A. Rencher, Methods of Multivariate Analysis, New York: John Wiley & Sons, Inc., 1995.

[13] H. He and Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications, Wiley-IEEE Press, 2013.

[14] N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002.

[15] H. He, Y. Bai, E. Garcia and S. Li, "ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning," in IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008.

[16] HackerEarth's churn risk rate dataset. Accessed on 11/01/2021. https://www.kaggle.com/datasets/imsparsh/churn-risk-rate-hackerearth-ml

[17] Scikit-Learn. Accessed on 11/01/2021. https://scikit-learn.org/stable/