Model Selection for Customer Churn Prediction in the Telecommunications Industry

Adam Cooke, Anthony Gheen, Nitin Shirsat
School of Data Science and Society
University of North Carolina
Chapel Hill, NC 27514

Abstract

This report aims to identify an improved methodology for implementing predictive models to identify customers at risk of churn. We perform an extensive comparison of six models used heavily in churn prediction. For each model, we identify the sampling method and scaling method that provided the highest F_1 score. Once the optimal sampling and scaling methods were identified, we further increase F_1 scores by optimizing hyperparameters for each of the models. Finally, we train and test model performance on three new datasets to determine if the identified methods can be used as a baseline for predicting customer churn in similar or different industries.

1 Introduction

Customer churn, defined as the termination of customer subscriptions to services, represents one of the critical challenges faced by businesses operating subscription-based models, especially in industries like telecommunications, banking, and online services. Churn directly impacts revenue streams, customer lifetime value, and overall market competitiveness. For telecommunications companies, particularly, customer acquisition costs (CAC) are substantially high, driven by significant investments in network infrastructure, marketing campaigns, and regulatory compliance. Thus, retention strategies driven by predictive analytics offer high returns on investment.

Effective churn prediction enables organizations not only to proactively engage and retain customers at risk but also to optimize marketing strategies by identifying and investing resources in high-value customers. Moreover, accurate churn prediction models help reduce unnecessary customer retention expenditure on customers unlikely to churn, improving overall operational efficiency.

Previous research in churn prediction spans a variety of analytical techniques, including logistic regression, decision trees, ensemble methods, and neural networks. Each technique provides unique advantages regarding interpretability, computational efficiency, and predictive accuracy. However, real-world datasets in churn scenarios are typically characterized by severe class imbalance, where the number of churn events is significantly smaller than non-churn events. This imbalance makes traditional accuracy metrics inadequate and demands specialized sampling strategies (undersampling, oversampling, synthetic generation) and robust performance metrics such as the F_1 score and ROC-AUC score.

In our work, we rigorously explore multiple state-of-the-art sampling and scaling methodologies combined with leading classification algorithms to identify an optimal approach for churn prediction in telecommunications. Unlike prior works that often rely solely on accuracy, our evaluation emphasizes business-critical metrics such as F_1 score and ROC-AUC, providing balanced insights into the practical utility of models. Furthermore, we perform comprehensive hyperparameter optimization and propose future experiments involving model explainability techniques (e.g., SHAP analysis), feature engineering enhancements, and cross-industry validation to increase the robustness and generalizability of our findings.

2 Related Work

J. Burez et. al. examined the effects of undersampling and oversampling when training classification models to account for scenarios where the predicted event is rare. If the true proportion of customers who churned was 5% in the original data set, under-sampling was used to artificially inflate the proportion of churned customers to up to 50%. They determined that this under-sampling technique increased the AUC for a logistic regression model but had no significant impact when using Random Forests.

O. Adwan et. al. used a Multilayer Perceptron Neural Network (MLP) to predict whether a customer was at risk of churn. They used change-on-error (CoE) analysis to identify which of the features had a large impact on the probability of a customer churning, and looked at the weights of each layer of the neural network to identify which features had the most impact.

Florian and Damien used recurrent neural networks to predict customer churn in the online gambling industry. This method uses a time series element to predict churn, and specifically predicted whether or not a player would churn within the next 30 days of a current time t. Predicting churn in this way is needed in the online gambling industry because there is not a simple way of identifying whether or not a customer has churned, compared to telecommunications where a customer has to manually opt out or cancel their services. Using an RNN could be useful in predicting churn in the telecommunications industry but seems unnecessarily complex for this task.

3 Methods

3.1 Data

Table 1: Features and Descriptions

Feature	**Description**
Gender	Gender: The customer's gender: Male, Female
Age	Age: The customer's current age, in years, at the time the fiscal quarter ended.
Under 30	Under 30: Indicates if the customer is under 30 years old: Yes, No
Senior Citizen	Senior Citizen: Indicates if the customer is 65 or older: Yes, No
Partner	Married: Indicates if the customer is married: Yes, No
Dependents	Dependents: Indicates if the customer lives with any dependents: Yes, No. Dependents could be children, parents, grandparents, etc.
Number of Dependents	Number of Dependents: Indicates the number of dependents that live with the customer.
Referred a Friend	Referred a Friend: Indicates if the customer has ever referred a friend or family member to this company: Yes, No
Number of Referrals	Number of Referrals: Indicates the number of referrals to date that the customer has made.
Tenure	Tenure in Months: Indicates the total amount of months that the customer has been with the company by the end of the quarter specified above.
Phone Service	Phone Service: Indicates if the customer subscribes to home phone service with the company: Yes, No
Avg Monthly Long Distance Charges	Avg Monthly Long Distance Charges: Indicates the customer's average long distance charges, calculated to the end of the quarter specified above.
Multiple Lines	Multiple Lines: Indicates if the customer subscribes to multiple telephone lines with the company: Yes, No
Internet Service	Internet Service: Indicates if the customer subscribes to home internet service with the company: Yes, No
Internet Type	Internet Type: Indicates what type of Internet service the customer subscribes to with the company: No, DSL, Fiber Optic, Cable.
Avg Monthly GB Download	Avg Monthly GB Download: Indicates the customer's average download volume in gigabytes, calculated to the end of the quarter specified above.
Online Security	Online Security: Indicates if the customer subscribes to an additional online security service provided by the company: Yes, No
Online Backup	Online Backup: Indicates if the customer subscribes to an additional online backup service provided by the company: Yes, No
Device Protection Plan	Device Protection Plan: Indicates if the customer subscribes to an additional device protection plan for their Internet equipment provided by the company: Yes, No
Premium Tech Support	Premium Tech Support: Indicates if the customer subscribes to an additional technical support plan from the company with reduced wait times: Yes, No
Streaming TV	Streaming TV: Indicates if the customer uses their Internet service to stream television programing from a third party provider: Yes, No. The company does not charge an additional fee for this service.
Streaming Movies	Streaming Movies: Indicates if the customer uses their Internet service to stream movies from a third party provider: Yes, No. The company does not charge an additional fee for this service.
Streaming Music	Streaming Music: Indicates if the customer uses their Internet service to stream music from a third party provider: Yes, No. The company does not charge an additional fee for this service.
Unlimited Data	Unlimited Data: Indicates if the customer has paid an additional monthly fee to have unlimited data downloads/uploads: Yes, No
Contract	Contract: Indicates the customer's current contract type: Month-to-Month, One Year, Two Year.
Paperless Billing	Paperless Billing: Indicates if the customer has chosen paperless billing: Yes, No
Payment Method	Payment Method: Indicates how the customer pays their bill: Bank Withdrawal, Credit Card, Mailed Check
Monthly Charges	Monthly Charge: Indicates the customer's current total monthly charge for all their services from the company.
Total Charges	Total Charges: Indicates the customer's total charges, calculated to the end of the quarter specified above.
Total Refunds	Total Refunds: Indicates the customer's total refunds, calculated to the end of the quarter specified above.
Total Extra Data Charges	Total Extra Data Charges: Indicates the customer's total charges for extra data downloads above those specified in their plan, by the end of the quarter specified above.
Total Long Distance Charges	Total Long Distance Charges: Indicates the customer's total charges for long distance above those specified in their plan, by the end of the quarter specified above.
Total Revenue	Total Revenue: Indicates the cusomer's total revenue generated, calculated to the end of the quarter specified above.
Churn	Churn Label: Yes = the customer left the company this quarter. No = the customer remained with the company. Directly related to Churn Value.

The data used in to train the below models is the 'Telco' dataset published by IBM. This data set consists of simulated data in a fictional telecommunication company. The full list of features and their descriptions are in Table 1. All categorical variables were one-hot encoded and numerical variables will be scaled using various scaling techniques detailed below. The full dataset has a total of 7032 rows, of which 1869 (27%) are customers who have churned. The distributions of the numerical and categorical features are listed in Figures 1 and 2.

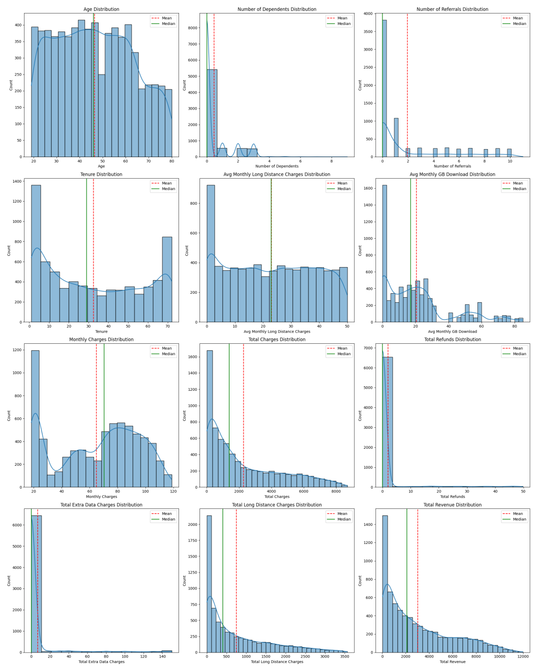


Figure 1: Distribution of Numerical Features

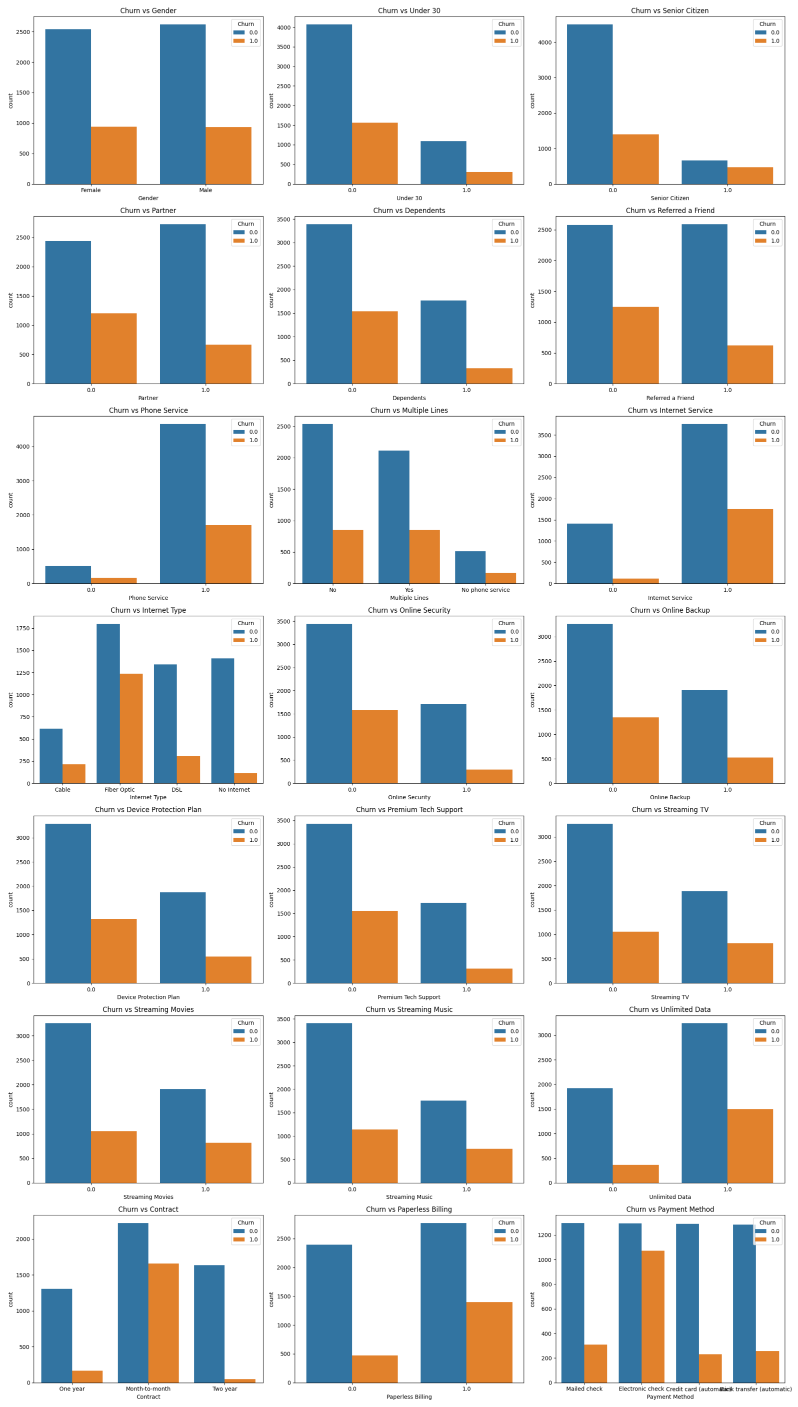


Figure 2: Distribution of Categorical Features

To identify the optimal method of training a classification model to predict customer churn, multiple sampling methods and scaling methods will be employed. For each model we train, we will build train datasets using the following steps:

1. Split the original dataframe into training dataset and testing dataset using a 5-fold randomly shuffled cross validation
2. Determine scaling technique to be used and scale continuous variables
3. Determine sampling method to be used and resample from training dataset

4. Fit model using the training set and score based on test dataset

The combination of model, scaling method, and sampling method will be reported on by taking the average of the evaluation metrics across all 5 folds of the cross validation splits.

3.2 Sampling Methods

J. Burez et. al described the potential need to use different sampling methods when predicting customer churn, due to the relative rarity of a customer disconnecting services. In this report, we will train each model using four sampling techniques: no sampling at all, random under sampling, random over-sampling, and Synthetic Minority Over-sampling Technique (SMOTE). We are including a technique that does not use any sampling method at all to establish a baseline of model performance. Random under-sampling and random over-sampling both artificially increase the proportion of churners in the data, but they are not the same. Under-sampling removes instances from the majority class until the desired proportion is met, and oversampling will duplicate instances of the minority class to increase the proportion of churners to non-churners. Lastly, we will be using SMOTE. SMOTE is another oversampling technique, but differs from the oversampling technique above in that it generates synthetic examples to balance out the minority class. SMOTE generates new synthetic instances by taking one real instance x_i of the minority class, taking several nearest neighbors of the same class, and performing random interpolation to obtain new instances.

3.3 Scaling Methods

For scaling, we will use 4 approaches: we will train the models without any scaling of variables, using ScikitLearn's StandardScaler, ScikitLearn's MinMax Scaler and ScikitLearn's RobustScaler. The only columns that we are scaling are the columns that contain continuous numerical values. Binary indicator feature values and encoded values will not be scaled.

Scikit Learn's StandardScaler scales each observation by subtracting the mean of that feature and dividing by the standard deviation. RobustScaler scales each observation by subtracting the median from each observation and dividing it by the interquartile range, which is the 75th percentile minus the 25th percentile. MinMax scaler scales each observation using the below formula:

$$x_{scaled} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

3.4 Models Used

We are comparing six different models and comparing their ability to predict churn. The models we will use are the following.

1. Decision Tree Classifier
2. Logistic Regression
3. Gaussian Naive Bayes Classifier
4. Random Forest Classifier
5. Support Vector Machine
6. XGBoost Classifier

3.5 Validation Metrics

There are many metrics that can be used to validate a classification model, but we will primarily be interested in the F_1 score and area under the receiver operator characteristic curve (AUC-ROC). F_1 score is comprised of two other performance metrics, precision and recall.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

Where TP denotes 'true positive' (customers who had churned and the model predicted churn), FP denotes 'false positive' (customers who did not churn but model predicted churn), and FN denotes 'false negative' (customers who churned but model predicted not churn). F_1 score is defined as the harmonic mean of precision and recall:

$$F_1 score = 2 * \frac{precision * recall}{precision + recall} \tag{4}$$

F_1 score is one of the best ways we can evaluate these models' ability to predict churn, because this value will be low if either precision or recall is low. In customer churn, incorrectly classifying a customer as at risk of churn results in lowering prices for customers who were never at risk of leaving. Incorrect labeling of a customer as not being at risk of churn likely results in losing that customer. Both of these scenarios are costly to the business, and F_1 score accounts for both scenarios.

We will also use AUC-ROC, which is the area under the curve plotted with the true positive rate on the vertical axis and the true negative rate on the horizontal axis. Simply put, the closer this value is to 1, the better the performance of the model.

3.6 Hyper Parameter Tuning

After identifying the optimal combinations of scaling and sampling techniques for each classification model based on average F_1 scores (as shown in Table 2), we performed hyperparameter tuning to further enhance model performance. We used GridSearchCV, a technique that uses all possible combinations of specified hyper-parameters using cross validation to find the best combination of hyper-parameters. We predefined values for some of the more crucial hyper-parameters for each model, and only defined a select few values for each parameter to save on computational resources. The GridSearchCV was completed using 5-fold cross validation and the models were evaluated across multiple metrics including accuracy, precision, recall, F_1 score, and AUC scores. However, the F_1 score was the primary metric used to select the optimal hyper-parameters for each classification model. The following parameters are those for which we optimized for each classification model type.

Logistic Regression:

- Regularization strength (C): $[0.1, 1.0, 10]$ - The smaller the value the stronger the regularization (Default: 1.0)
- Penalty: [L2] - Type of regularization applied to the model (Default: L2). L1 is excluded due to lbfgs incompatibility
- Solver: [liblinear, lbfgs] - Algorithm used to optimize the parameters (Default: lbfgs)
- Class weight: [None, balanced] - Balanced automatically adjust the weights inversely proportional to class frequencies (Default: None)

Support Vector Classifier:

- Regularization strength (C): $[0.1, 1.0, 10]$ - The smaller the value the stronger the regularization (Default: 1.0)
- Kernel: [Linear, RBF, Sigmoid] - Kernel type for used in algorithm (Default: RBF)
- Gamma: [scale, auto, $0.1, 1.0$] - Kernel coefficient (Default: Scale)
- Class weight: [None, balanced] - Balanced automatically adjust the weights inversely proportional to class frequencies (Default: None)

Decision Tree Classifier:

- Criterion: [gini, entropy, log_loss] - The function used to measure the quality of a split (Default: gini)
- Max depth: [None, $5, 10, 15$] - Maximum depth of the tree (Default: None)
- Min samples split: $[2, 10, 20]$ - The minimum number of samples required to split an internal node (Default: 2)

- Class weight: $[\text{None}, \text{balanced}]$ - Balanced automatically adjust the weights inversely proportional to class frequencies (Default: None)

Random Forest Classifier:

- N estimators: $[100, 200, 300]$ - Number of trees in the forest (Default: 100)
- Criterion: $[\text{gini}, \text{entropy}, \text{log\_loss}]$ - The function used to measure the quality of a split (Default: gini)
- Max depth: $[\text{None}, 5, 10, 15]$ - Maximum depth of the tree (Default: None)
- Min samples split: $[2, 10, 20]$ - The minimum number of samples required to split an internal node (Default: 2)
- Max features: $[\text{None}, \text{sqrt}, \text{log2}]$ - The number of features to consider when looking for the best split (Default SQRT)
- Class weight: $[\text{None}, \text{balanced}]$ – class balancing mechanism

Gaussian Naive Bayes Classifier:

- Var smoothing: $[1 \times 10^{-9}, 1 \times 10^{-7}, 1 \times 10^{-5}]$ - Artificially adding a value to the variance of each feature, widening the distribution and accounting for more samples further from the mean (Default: 1×10^{-9})

XGBoost Classifier:

- N estimators: $[100, 200, 300]$ - Number of trees in the forest (Default: 100)
- Learning rate: $[0.1, 0.2, 0.3]$ - Step size shrinkage (Default: 0.3)
- Max depth: $[3, 6]$ - Maximum depth of the trees (Default: 6)
- Subsample: $[0.8, 1.0]$ - The fraction of samples to be randomly sampled to build each tree (Default: 1.0)
- Colsample bytree: $[0.8, 1.0]$ - The fraction of features to be randomly sampled for building each tree (Default: 1.0)

3.7 SHAP Analysis

In order to effectively communicate model results to stakeholders, Shapley additive explanations (SHAP) will be employed. SHAP analysis provides a method for measuring each feature's contribution to the predicted value for a given sample. SHAP Analysis is rooted in game theory, and builds upon concepts used to determine a fair payout in a collaborative game where individual contributions are not equal. While we will not dive deep into the mathematical foundations of SHAP Analysis, the below formula illustrates the concept at a high level.

$$f_i = \phi_0 + \sum_{features} \phi_{i,j} \tag{5}$$

Where f_i indicates the prediction of the model for sample *i*, ϕ_0 indicates the average prediction of the model, and $\phi_{i,j}$ indicates the SHAP values for each sample *i*. Each individual SHAP value represents the impact that feature had on the model's prediction, relative to the average prediction of the model.

4 Results

While attempting to find the optimal combination of scaling and sampling techniques for each type of classification model, 96 models were trained. Tables 2 and 3 contain the combination of scaling and sampling techniques that lead to the highest F_1 and AUC scores, respectively. Interesting to note, the RobustScaler was not among any of the combinations of high performing models, 2 of the 12 listed in tables 2 and 3 scored better with no scaling, and another 2 of the 12 listed in tables 2 and 3 scored better with no sampling techniques applied. Most of the models performed best when using the StandardScaler and random oversampling techniques applied. XGBoost performed the best for maximizing both the F_1 and AUC scores. The combination of no scaling and random oversampling were optimal for F_1 score, and the combination of using the StandardScaler and SMOTE were optimal for AUC scores. Other models like Logistic Regression and Random Forest Classifiers Support Vector Classifiers performed comparatively to the F_1 scores of XGBoost having strong recall, but poor precision causes their F_1 scores to drop.

There was more variance in sampling techniques. When maximizing F_1 score, the Decision Tree and Random Forest classifiers performed best with undersampling, the Support Vector Machine, XGBoost and Logistic Regression performed better with oversampling, and only the Naive Bayes classifier did best with SMOTE.

Figure 3 contains the AUC curves for all combinations of the models trained for each type of classification model. In the decision tree classifier, logistic regression, and support vector machine ROC curves, there is a significant variance between the ROC curves depending on the combination of scaling technique and sampling method. For naive bayes, random forest, and XGBoost, there does not appear to be much change to the ROC curves when the sampling and scaling methods change.

We then took the 6 combinations of scaling and sampling techniques and utilized GridSearchCV to perform hyper parameter tuning, aiming to see if we could further enhance model performance. The metrics shown in table 4 show a substantial improvement in model performance across all models. Most notably, the Support Vector Classifier shown the most significant improvement in performance with the F_1 score increasing by .258 from 0.674 to 0.931, resulting in the highest F_1 score of the 6 models. XGBoost also improved similarly and held the highest AUC score. All models improved with hyper parameter tuning. These results show that initial scaling and sampling technique combination choices are critical, however, hyperparameter optimization is crucial for unlocking the potential performance of each model.

Table 2: Combination of Scaling and Sampling that maximizes F_1 Score

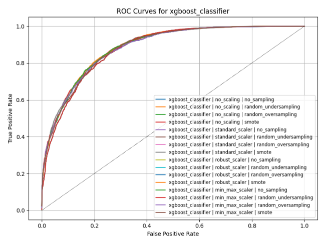
classification_model	**scaling_technique**	**sampling_technique**	**accuracy**	**precision**	**recall**	**f1_score**	**log_loss**	**roc_auc**
xgboost_classifier	no_scaling	random_oversampling	0.817974	0.642409	0.711859	0.675115	0.406087	0.886566
random_forest_classifier	standard_scaler	random_undersampling	0.787257	0.568583	0.826544	0.673595	0.458929	0.875245
support_vector_classifier	standard_scaler	random_oversampling	0.787684	0.56976	0.824689	0.673592	0.422451	0.88003
logistic_regression	standard_scaler	random_oversampling	0.780573	0.558455	0.834342	0.668893	0.431611	0.883752
gaussian_naive_bayes_classifier	standard_scaler	smote	0.769764	0.547335	0.780207	0.642949	2.185817	0.849863
decision_tree_classifier	standard_scaler	random_undersampling	0.730519	0.49491	0.725554	0.588345	9.713094	0.728828

Table 3: Combination of Scaling and Sampling that maximizes AUC

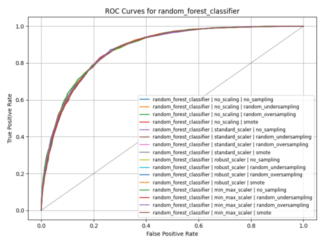
classification_model	**scaling_technique**	**sampling_technique**	**accuracy**	**precision**	**recall**	**f1_score**	**log_loss**	**roc_auc**
xgboost_classifier	standard_scaler	smote	0.829492	0.68649	0.661427	0.673444	0.38513	0.890087
logistic_regression	standard_scaler	no_sampling	0.827218	0.690538	0.63485	0.661133	0.366044	0.885327
support_vector_classifier	standard_scaler	random_oversampling	0.787684	0.56976	0.824689	0.673592	0.422451	0.88003
random_forest_classifier	standard_scaler	smote	0.822099	0.670228	0.650816	0.659916	0.416036	0.879037
gaussian_naive_bayes_classifier	no_scaling	no_sampling	0.766352	0.541913	0.786551	0.641472	1.763892	0.85059
decision_tree_classifier	standard_scaler	random_undersampling	0.730519	0.49491	0.725554	0.588345	9.713094	0.728828

Table 4: Optimized hyper parameters selected based on the best average F_1 Score

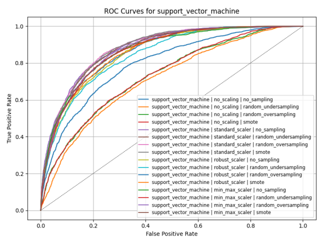
classification_model	**scaling_technique**	**sampling_technique**	**best_hyperparameters**	**accuracy**	**precision**	**recall**	**f1_score**	**roc_auc**
support_vector_classifier	standard_scaler	random_oversampling	{’C’: 10, ’class_weight’: None, ’gamma’: 1.0, ’kernel’: ’rbf’}	0.933082	0.960004	0.903858	0.931057	0.967525
xgboost_classifier	no_scaling	random_oversampling	{’colsample_bytree’: 0.8, ’learning_rate’: 0.2, ’max_depth’: 6, ’n_estimators’: 300, ’subsample’: 0.8}	0.914294	0.874552	0.96723	0.918555	0.971051
random_forest_classifier	standard_scaler	random_undersampling	{’class_weight’: None, ’criterion’: ’entropy’, ’max_depth’: 10, ’max_features’: None, ’min_samples_split’: 2, ’n_estimators’: 300}	0.796956	0.767085	0.854196	0.807968	0.878474
logistic_regression	standard_scaler	random_oversampling	{’C’: 1.0, ’class_weight’: ’balanced’, ’penalty’: ’l2’, ’solver’: ’lbfgs’}	0.799632	0.781263	0.832183	0.805904	0.887956
gaussian_naive_bayes_classifier	standard_scaler	smote	{’var_smoothing’: 1e-09}	0.787236	0.775824	0.808266	0.791667	0.8697
decision_tree_classifier	standard_scaler	random_undersampling	{’class_weight’: None, ’criterion’: ’gini’, ’max_depth’: 5, ’min_samples_split’: 2}	0.776082	0.744953	0.842526	0.789896	0.848311



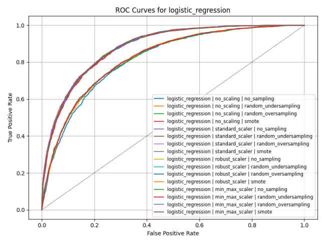
(a)



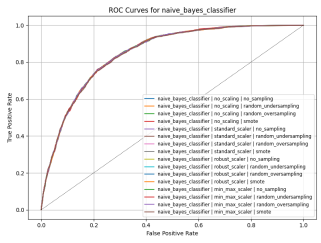
(b)



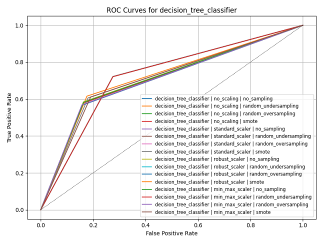
(c)



(d)

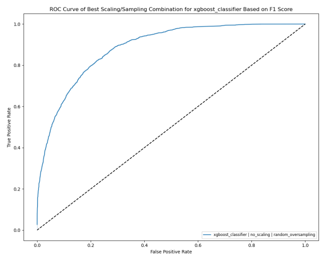


(e)

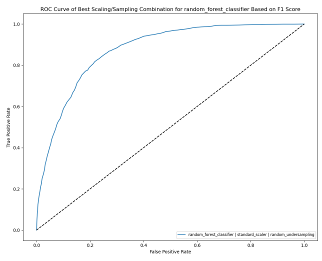


(f)

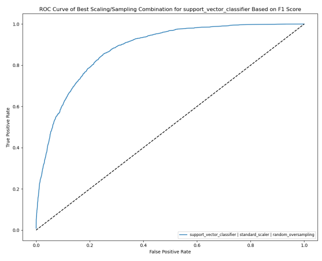
Figure 3: AUC curves for each type of classification model



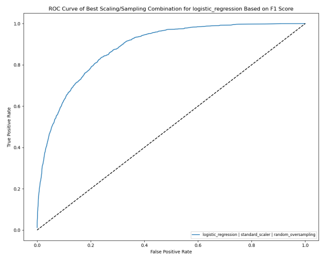
(a)



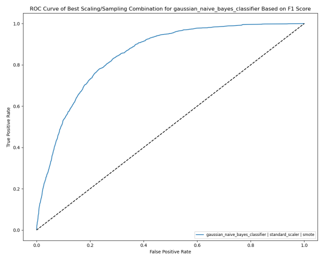
(b)



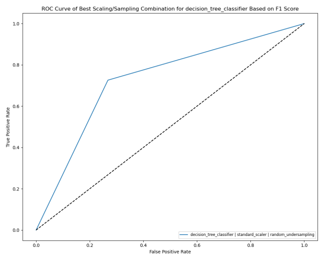
(c)



(d)



(e)



(f)

Figure 4: AUC curves for the optimal scaling/sampling combinations for each type of classification model

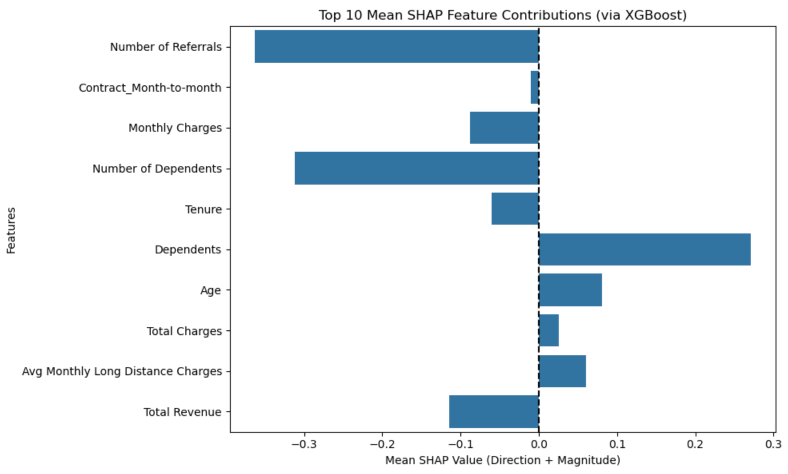


Figure 5: Top 10 Mean SHAP Feature Contributions (Via XGBoost

5 Applications beyond initial dataset

Now that we have identified the optimal scaling and sampling technique for each model and identified the hyperparameters that led to the highest F_1 scores using the original dataset, we want to apply those techniques and hyperparameters to different datasets to test how applicable these methods are. Before moving to a completely new dataset, we performed feature engineering on the original dataset before training the models in an attempt to further increase the F_1 scores and AUC. The feature engineering consisted of the following:

- Binning age, monthly charges, customer tenure, and average monthly GB download into 10 distinct buckets each
- Created a "Service Penetration Rate" column defined as the total number of services a customer is subscribed to divided by the number of services available
- A binary indicator to identify if a customer has any family
- A "Family Size" column indicating combining the partner indicator and the sum of all dependents
- A "Referral Rate" defined as the total number of referrals for a customer divided by the customer's tenure
- A "Refund Rate" defined as the total refund amount divided by the total charges
- A "Monthly Cost per GB" column defined as the monthly charges divided by the average monthly download in GB
- An "Extra Charges" ratio defined as the ratio of total extra data charges over the total charges

We then dropped the original features used for binning, one-hot encoded the categorical features, and trained the optimal model combinations on this new dataset. With respect to F_1 score the results were similar to the results while using the original dataset, expect for the Support Vector Classifier. The recall in the Support Vector Classifier dropped below 0.02, which severely impacted the F_1 score. The full results of this model are listed in Table 5.

Table 5: Model Perfomance using Engineered/Derived Features

Classification Model	**Scaling Technique**	**Sampling Technique**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Log Loss**	**ROC AUC**
Random Forest Classifier	Standard Scaler	Random Undersampling	0.777302	0.552262	0.860853	0.672624	0.434098	0.884096
Logistic Regression	Standard Scaler	Random Oversampling	0.777588	0.555098	0.821599	0.662332	0.433754	0.881479
XGBoost Classifier	No Scaling	Random Oversampling	0.817972	0.655546	0.667395	0.661047	0.453056	0.882279
Decision Tree Classifier	Standard Scaler	Random Undersampling	0.747298	0.516246	0.846182	0.640395	0.610188	0.839351
Gaussian Naive Bayes Classifier	Standard Scaler	SMOTE	0.778297	0.571152	0.687393	0.622214	2.851317	0.838566
Support Vector Classifier	Standard Scaler	Random Oversampling	0.733077	0.459865	0.019299	0.036977	0.753589	0.764371

We identified a new dataset also from the telecommunications industry containing 64374 records with 8 features and indicator of whether the customer churned or not. Three of these features were categorical and one hot encoded.

We trained and tested each model with their respective combination of scaling, sampling, and hyperparameters. Table 6 contains the average the model performance metrics across a 5-fold cross validation split.

Table 6: Model Performance using second Telecommunications Dataset

Classification Model	**Scaling Technique**	**SamplingTechnique**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Log Loss**	**ROC AUC**
XGBoost Classifier	No Scaling	Random Oversampling	0.999922	0.999869	0.999968	0.999918	0.000582	0.999992
Random Forest Classifier	Standard Scaler	Random Undersampling	0.998742	0.998261	0.999082	0.998671	0.004244	0.999978
Decision Tree Classifier	Standard Scaler	Random Undersampling	0.956613	0.930023	0.982346	0.955465	0.088584	0.99441
Support Vector Classifier	Standard Scaler	Random Oversampling	0.942228	0.934457	0.944284	0.939341	0.143388	0.987544
Gaussian Naïve Bayes Classifier	Standard Scaler	SMOTE	0.835151	0.800956	0.867575	0.832931	0.403325	0.908518
Logistic Regression	Standard Scaler	Random Oversampling	0.825504	0.800325	0.84158	0.820427	0.393823	0.904097

Similar to the original dataset, the Support Vector Classifier and XGBoost model had the highest F_1 score and AUC. All models improved significantly as compared to the original dataset, with all models except the Naive Bayes Classifier and Logistic Regression having an F_1 score of over 0.939. This appears to suggest that these model parameters and techniques could be implemented to predict churn in any industry with decent success.

We finally applied the same methodologies to a third dataset, this time in the banking industry. The models performed significantly worse when trained and tested in this dataset, with the highest F_1 score slightly below 0.6. XGBoost was the best performing model in terms of F_1 score, which was the case with both the original dataset and the second telecommunications dataset. The significantly lower F_1 scores across the board could indicate that these combinations are not necessarily applicable to datasets or industries beyond the initial dataset.

Table 7: Model Performance using Banking Dataset

Classification Model	**Scaling Technique**	**Sampling Techinque**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Log Loss**	**ROC AUC**
XG Boost Classifier	No Scaling	Random Oversampling	0.8384	0.607759	0.584399	0.595701	0.422642	0.836589
Random Forest Classifier	Standard Scaler	Random Undersampling	0.7810	0.476573	0.756371	0.584504	0.474847	0.856955
Decision Tree Classifier	Standard Scaler	Random Undersampling	0.7449	0.430041	0.761398	0.548784	0.618796	0.830766
Gaussian Naive Bayes Classifier	Standard Scaler	SMOTE	0.7361	0.408822	0.661405	0.505227	0.549287	0.781643
Logistic Regression	Standard Scaler	Random Undersampling	0.7117	0.383714	0.687141	0.492409	0.577906	0.768511
Support Vector Classifier	Standard Scaler	Random Undersampling	0.7991	0.509257	0.361364	0.422674	0.698014	0.764247

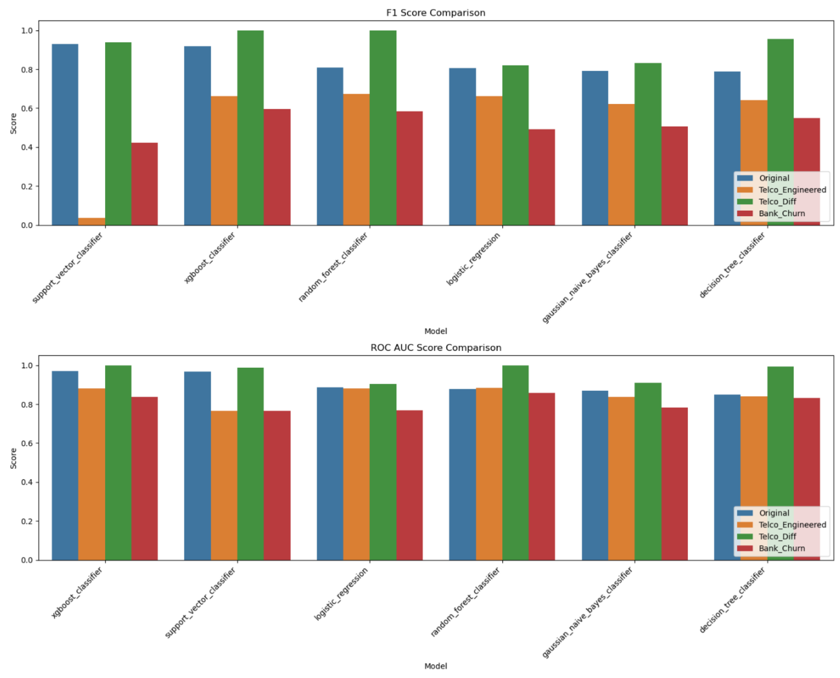


Figure 6: F_1 score and ROC AUC score Bar plots for each classification model on each dataset

6 Conclusion

Some of the key takeaways are that XGBoost, Random Forest, and Naive Bayes classifiers all had little variance between combinations of sampling and scaling techniques. Support Vector Classifier, Logistic Regression, and the Decision Tree Classifier all had a significantly higher variance in the ROC Curves given different combinations.

It appears that the model, sampling, and scaling combinations identified in this report may be applicable to any instance where predicting customer churn is of benefit. The model seemed to maintain high AUC values and F_1 scores beyond the original dataset where optimal scaling/sampling strategies and hyperparameters were identified. This could be used as a framework or starting point for any industry where minimizing churn is important.

To continue in this research, it would be critical to identify other verified, high quality datasets to apply the combinations of scaling and sampling techniques with the optimized hyperparameters. With more applications of these combinations, we would be able to better assess whether this methodology is applicable to churn prediction as a whole. It would also be interesting to assess the variances between each combination for each of the chosen models to see if the pattern of low variance for XGBoost, Random Forest and Naive Bayes and high variance for Support Vector Classifier, Logistic Regression, and Decision Trees follows. Regardless of the performance of these specific model combinations for predicting churn in future datasets or industries, this iterative process serves as a good framework in identifying the ideal data pre-processing techniques and hyperparameters that will apply to any dataset.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.