Data Mining CS 699 A2

Final Report

Dataset: 2021 BRFSS Asthma Call-back Survey (ACBS)

Student 1: Yash Rao

BUID: U91351678

Student 2: Sanjana Prasad

BUID: U35186791

Table of Contents

1. Introduction and Dataset Overview

The dataset used in this project is the 2021 BRFSS Asthma Call-back Survey (ACBS), which is part of the Behavioral Risk Factor Surveillance System (BRFSS) maintained by the Centers for Disease Control and Prevention (CDC). The data was collected from individuals who participated in the BRFSS, a national survey that aims to assess the health behaviors and outcomes of the U.S. population. The specific dataset is focused on individuals with asthma, with the purpose of understanding the patterns of healthcare utilization related to asthma, such as emergency room visits and urgent care treatments.

This dataset contains a total of 7,473 tuples (rows), with each tuple representing an individual. The dataset includes 359 attributes (columns), which consist of demographic, health, and lifestyle information.

The class attribute in the dataset is labeled as Class, with two possible values:

Y: The individual has visited an emergency room or urgent care center due to asthma at least once in the past 12 months.

N: The individual has not visited an emergency room or urgent care center for asthma-related reasons in the past 12 months.

The goal of this analysis is to predict whether an individual will visit the emergency room or urgent care center due to asthma, based on the available features. This is a binary classification problem, where the target variable (Class) can be used to assess the predictive performance of various machine learning models.

Through further preprocessing and feature engineering, we explore the relationships between the individuals' health profiles and their likelihood of utilizing emergency care services due to asthma. The analysis will involve applying various machine learning algorithms to predict the Class attribute, and evaluating the model performance based on metrics like accuracy, precision, recall, F1 score, and AUC-ROC, MCC, Kappa.

2. Initial Data Preprocessing

2. 1. Data Exploration

The dataset consisted of both categorical and numeric variables. Using sapply(df, class), the data types of columns were identified. Categorical variables were transformed into factors (if they had ≤ 33 unique values), ensuring proper handling during model training.

2. 2. Handling Redundant and Irrelevant Columns

Columns deemed irrelevant or redundant were removed. IDATE and SEQNO were removed because they were sequence numbers used for tracking. Employment-related Columns: EMP_STAT, UNEMP_R, and EMP_EVER1 were dropped as they didn't contribute directly to the classification task. HISPAC3 was removed due to lack of clarity and relevance based on dataset documentation. Columns with low variability (e.g., those with single unique values) were removed using select_if(~n_distinct(.) > 1). This helped streamline the dataset for modeling.

2. 3. Factor Conversion

Columns with ≤ 33 unique values were converted to factors using mutate(across(where(~n_distinct(.) <= 33), as.factor)), ensuring proper treatment of categorical data.

2. 4. Target Label Transformation

The class label Class was re-coded to binary values. Y (Visited ER) was mapped to 1. N (Did not visit) was mapped to 0. This conversion enabled easier processing during model training.

2. 5. Correlation and Collinearity

A correlation plot was generated to examine the relationships between numeric variables using Pearson's correlation. Variables with high correlations (≥ 0.7) were reviewed. WEIGHT_IN and LLCPWT_F were highly correlated; WEIGHT_IN was dropped in favor of LLCPWT_F. Other redundant variables like NRECSTR and HTIN4 were retained, while similar variables were removed to avoid multicollinearity.

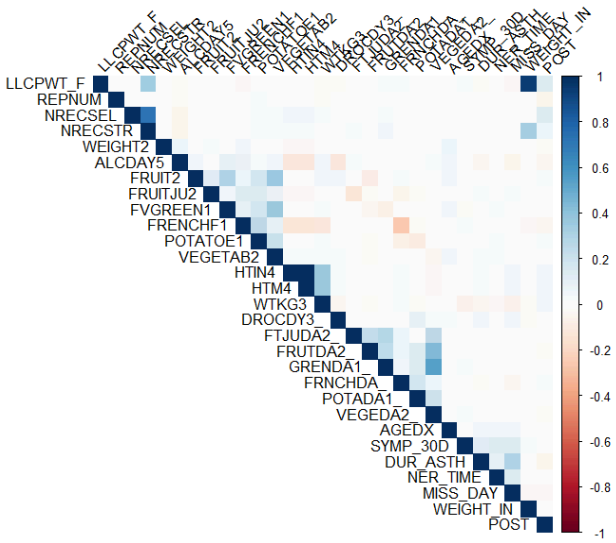

Figure 1: Correlation Plot

2. 6. Missing Value Imputation

Columns with missing values < 33% were imputed using mode imputation for categorical variables (e.g., Yes/No responses), and median imputation for numeric variables, ensuring that outliers didn't skew the imputation process.

2. 7. Handling Refused/Invalid Responses

Columns with refused responses (e.g., 9, 99, 999) were analyzed. Given their low occurrence, these responses were imputed as the mode or removed if they didn't carry significant information.

2. 8. Outlier Detection

Outliers in numeric variables were detected using boxplots, showing extreme values, particularly for right-skewed distributions. For factor variables, outliers represented refused answers or missing responses, which were retained as they provided context.

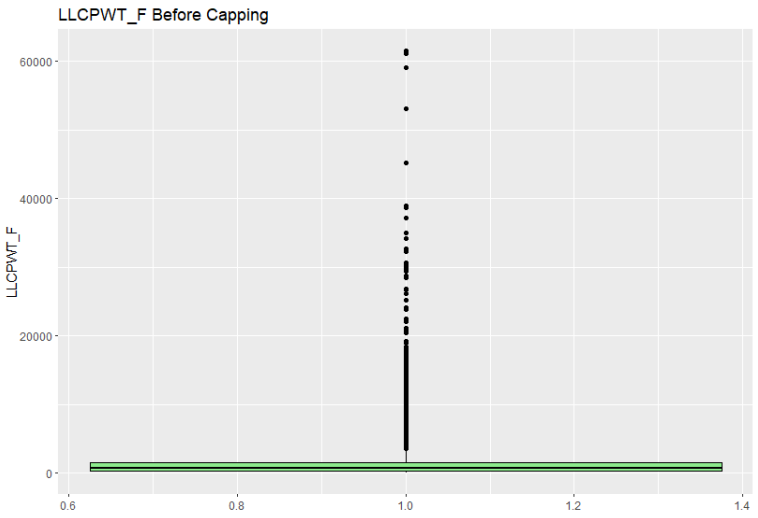

Figure 2: Boxplot of LLCPWT_F

2. 9. Outlier Handling

Outliers were handled by capping the values at the 1st and 99th percentiles, ensuring that extreme values did not unduly affect the model training. Log transformations were applied to right-skewed variables, normalizing their distributions. Variables such as MISS_DAY and POTATOE1 were excluded from transformations as they had minimal impact on the overall distribution.

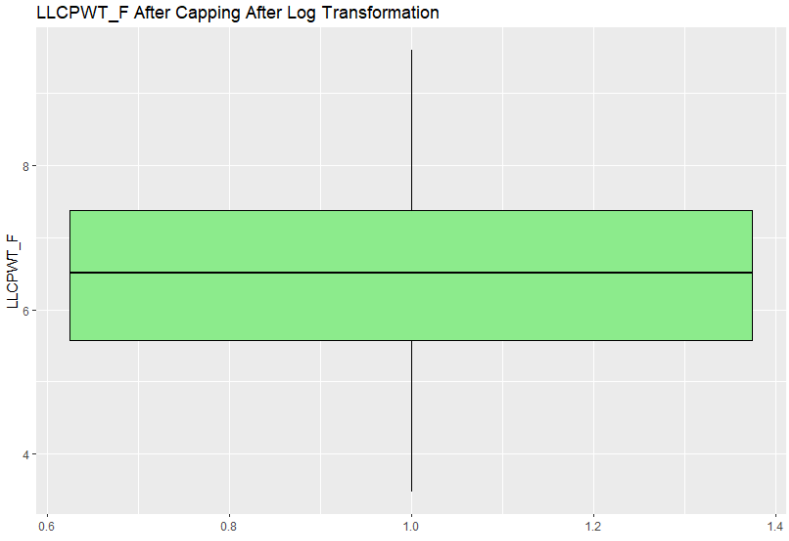

Figure 3: Boxplot of LLCPWT_F after capping and after log transformation

2. 10. Standardization

After addressing missing data and outliers, all numeric variables were standardized using z-score normalization. This ensured that variables had a mean of 0 and a standard deviation of 1, enabling more efficient model training without scale-related bias.

The final dataset contains 301 columns after dropping 58 irrelevant or redundant features. It is now ready for classification model training. The preprocessed dataset and the R code used for preprocessing (including the handling of missing values, outliers, and transformations) are provided in the submission.

3. Data Mining Tools Used

The project was implemented in R using specialized libraries for data balancing, feature selection, and classification. Below is a description of the tools and techniques applied:

3.1. Data Balancing Techniques

To address class imbalance, the project applied Random Under-Sampling (RUS), which reduces the size of the majority class to match the minority class. This was achieved using the caTools package for splitting the data and custom sampling functions.

Random Under-Sampling (RUS): The majority class was undersampled to match the count of the minority class using the sample.split function from the caTools package. The undersampling was done manually by selecting an equal number of samples from both classes. A total of 713 class counts for each class was recorded.

Clustered Under-Sampling: K-Means clustering was used to perform clustered under-sampling. A total of 1200 (600 of majority class and 600 of minority class) was chosen as the desired sample number. A total of 600 clusters for the majority class were created and a sample from each of these clusters was taken, effectively reducing the majority class count. 600 random samples for minority class were taken. caret package was used to create the KMeans cluster with the sample() function to get minority class counts.

3.2. Feature Selection Algorithms

Feature selection was crucial in improving model performance by removing irrelevant or redundant variables. Several feature selection methods were used over the balanced datasets:

Boruta: This algorithm was employed using the Boruta package for feature selection. Boruta performs an all-relevant feature selection by comparing the importance of each feature to a random shadow feature. Selected features were then used to create the final training set (train_boruta).

Random Forest Feature Importance: The caret package's train function with the "rf" method was used to assess feature importance using Random Forest. The most important features, based on the variable importance measure, were selected for training the model. The resulting dataset (train_info_gain) was saved for further use.

Linear Discriminant Analysis (LDA): The MASS package's lda function was used for feature reduction. LDA selects features that best separate the classes by maximizing the between-class variance while minimizing the within-class variance. Features selected by LDA were used to create the final training dataset (train_lda).

3.3. Classification Algorithms

The following classification algorithms were tested to predict the target variable:

K-Nearest Neighbors (KNN): Implemented using the kknn package, KNN is a simple yet effective algorithm for classification. Hyperparameters such as the number of neighbors (k) and the distance metric (Manhattan distance) were tuned using grid search and repeated cross-validation (trainControl from caret). The model was trained and evaluated on the Boruta-selected feature set.

AdaBoost: The AdaBoost algorithm, implemented via the ada package, was used for classification by combining multiple weak learners (decision trees). Hyperparameters like iterations, maxdepth, and nu (learning rate) were tuned for better performance using grid search. The AdaBoost model was evaluated on the Boruta-selected feature set.

Decision Trees (rpart): The rpart method from the caret package was used to build decision trees. The complexity parameter (cp) was tuned for optimal performance using repeated cross-validation. This method was applied to the Boruta-selected feature set.

Support Vector Machines (SVM): The svm function from the e1071 package with the radial basis function (RBF) kernel was also included. C, which controls the trade-off between maximizing the margin andand sigma (also called gamma), which determines the influence of a single training example, effectively controlling the "spread" of the RBF kernel were tuned.

Random Forest: The random forest model was implemented using the randomForest package. The base model used 500 trees and the Mtry parameter, which specifies the number of features (predictors) randomly selected and considered for splitting at each node in the decision trees, was included in the parameter tuning.

XGBoost: XGBoost model was taken from the xgboost package in R. In XGBoost, nrounds, max_depth, eta, gamma, colsample_bytree, min_child_weight, and subsample control boosting iterations, tree complexity, learning rate, regularization, feature sampling, leaf constraints, and sample diversity respectively. All of these tuned to get optimal results

3. 4. Model Evaluation and Metrics

The models were evaluated using several key metrics:

Confusion Matrix: Used to compute the class-wise metrics accuracy, sensitivity, specificity, precision, recall, F1 score, MCC, and Kappa for each class (Class0 and Class1).

ROC-AUC: The AUC (Area Under the Curve) was calculated using the pROC package to assess the model's ability to discriminate between the classes. The ROC curve and AUC were plotted for visual inspection of model performance.

A custom function was created to calculate weighted metrics across both classes, considering class imbalances and providing a balanced view of model performance.

4. Classification Algorithms

4. 1. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm that classifies data points based on the majority class of the k closest training examples in the feature space. KNN works by calculating the distance between data points using metrics such as Euclidean distance and assigning the label based on the nearest neighbors. The algorithm is simple to implement and does not require a training phase, making it particularly useful in cases where the relationship between data points is not necessarily linear or predefined. The relevance of KNN to the problem lies in its ability to classify instances based on proximity, which is particularly useful when dealing with datasets where the classes are clearly separated by features but may not follow traditional linear patterns.

4. 2. Support Vector Machines (SVM)

Support Vector Machines (SVM) are a class of supervised learning models that find an optimal hyperplane that best separates data points of different classes in high-dimensional feature spaces. SVM can use kernel functions to transform the data into a higher dimension, enabling it to handle non-linear decision boundaries. SVMs are particularly effective in high-dimensional spaces and for problems with clear margin of separation between classes. The relevance of SVM to the problem lies in its ability to classify data that is not linearly separable, using kernel tricks to create flexible decision boundaries. It is particularly useful when dealing with high-dimensional datasets and when a clear, robust classification boundary is required.

4. 3. Decision Trees (rpart)

Decision Trees (implemented by rpart in R) are supervised learning models that recursively split the dataset into subsets based on feature values, resulting in a tree-like structure of decision rules. Each node in the tree represents a decision based on a feature, and each leaf node corresponds to a class label. Decision Trees are easy to interpret and visualize, providing a clear understanding of how decisions are made. They are well-suited for handling both categorical and continuous data. The relevance of Decision Trees to the problem lies in their interpretability and ability to model complex, non-linear relationships in the data. This makes them useful for understanding which features contribute most to class predictions and for developing transparent models.

4. 4. AdaBoost (Adaptive Boosting)

AdaBoost is an ensemble learning algorithm that combines multiple weak learners, typically decision trees, to create a strong classifier. In AdaBoost, each subsequent model is trained to

correct the errors made by the previous models, focusing on the misclassified instances. This iterative process helps to reduce bias and improve model performance, particularly in noisy datasets or when there is class imbalance. AdaBoost's relevance to the problem lies in its ability to improve accuracy by focusing on difficult-to-classify examples. In cases where the dataset has noisy or complex patterns, AdaBoost can boost the performance of simpler models, making it a robust choice for enhancing classification accuracy.

4. 5. Random Forest

Random Forest is an ensemble learning method that builds multiple decision trees during training and combines their predictions to improve accuracy and robustness. Each tree is trained on a random subset of the data with a random selection of features, which helps reduce overfitting and increases generalization. For classification tasks, Random Forest aggregates the majority vote across trees, making it effective in handling complex datasets, reducing variance, and managing high-dimensional data. Its use of randomness in both data and features leads to diverse, less correlated models, which collectively enhance prediction accuracy and robustness, making it highly suitable for reliable classification outcomes.

4. 6. XGBoost (eXtreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting) is an optimized gradient boosting algorithm designed for speed and performance, building an ensemble of decision trees where each tree sequentially corrects errors from the previous ones. Known for handling missing data, regularizing to prevent overfitting, and efficiently processing large datasets with parallelism, XGBoost excels in classification tasks due to its high accuracy, ability to capture complex patterns, and robustness in handling imbalanced classes and high-dimensional data. Its iterative boosting approach refines predictions, making XGBoost a preferred choice for tasks requiring precision and adaptability.

5. Detailed Data Mining Procedure

A detailed data mining procedure for reproducing results is presented here -

1. The data mining process began by loading the dataset from a CSV file into a DataFrame, then exploring the structure and unique values across each column to identify any constant or low-variance columns. Columns with only a single unique value were removed to streamline the dataset, as these do not contribute meaningful variation for analysis. DATE columns ("IDATE" and "IDATE_F") were excluded, as well as columns related to employment status ("EMP_STAT," "UNEMP_R," and "EMP_EVER1") and sequence numbers ("SEQNO" and "SEQNO_FINL") that were deemed irrelevant. For columns with fewer than 33 distinct values, a conversion to categorical (factor) data type was applied. The binary target variable "Class" was converted to a binary factor, with "Y" mapped to 1 and "N" to 0. An initial data distribution analysis was performed by creating histograms for age (AGEDX) and bar plots for gender (SEXVAR) and S_INSIDE variables. To handle missing data, the percentage of missing values was calculated for each column, with factor columns imputed by mode and numeric columns by median. A correlation analysis on numeric columns identified highly correlated pairs, which were documented for further inspection. Highly collinear columns were removed to reduce redundancy (e.g., "REPNUM," "NRECSEL," and several columns with overlapping information). Outliers were detected and capped within the 1st and 99th percentiles, with additional log transformations applied to selected numeric columns to improve skewness. Standardization of numeric columns followed, while rare categories within factor variables were identified and documented. Factor variables were reviewed to decide whether to retain or create dummy variables, given the potential importance of categorical responses. Finally, the cleaned and preprocessed data was saved to a new CSV file, "preprocessed_data.csv," for further analysis.
2. To prepare the data for model training, an initial 80-20 split was performed on the dataset to create separate training and testing sets, ensuring that 80% of the data was allocated for training and 20% for testing. This split was done with a set random seed for reproducibility, and the resulting sets were saved as "initial_train.csv" and "initial_test.csv." For training, the target variable was converted to a factor type. Given class imbalance, Random Under-Sampling (RUS) was applied to balance the class distribution in the training data by randomly selecting an equal number of samples from each class. This balanced subset, "train_undersampled," was analyzed to confirm an equal class distribution. To identify key features, the Boruta feature selection algorithm was applied to the undersampled training data, highlighting important features with their tentative attributes. These selected features were saved as "rus_boruta_train.csv" for future analysis. Next, a Random Forest model was used to further assess variable importance, enabling selection of the top 20 features with high predictive power; this data subset was saved as "rus_info_gain_train.csv." Additionally, Linear Discriminant

Analysis (LDA) was employed for feature selection, where non-informative (constant) features were removed before extracting the coefficients of the linear discriminants. The features were sorted by the magnitude of their coefficients, and the top 20 were selected for the final training dataset saved as "rus_lda_train.csv." This multi-method feature selection approach provided varied perspectives on important features, preparing multiple feature-engineered datasets for subsequent modeling.

3. The clustered data sampling approach was initiated by specifying desired sample sizes for the majority and minority classes (600 samples each). The majority and minority classes were separated, and k-means clustering was performed on the majority class to achieve the specified undersampling while maintaining representative clusters. Duplicate cluster samples were removed, and a balanced sample for the minority class was created using random sampling with a set seed for consistency. The resulting balanced dataset was combined into "train_undersampled" for feature selection. For feature selection, the Boruta algorithm was applied to identify important features. Selected attributes were documented and saved as "cluster_boruta_train.csv" after creating a dataset with only these features. To further explore feature importance, a Random Forest model was trained using the caret package, allowing extraction of the top 20 influential features based on variable importance, saved as "cluster_info_gain.csv."Linear Discriminant Analysis (LDA) was then used to refine feature selection by identifying discriminative features. Before applying LDA, constant features were removed to ensure effective feature extraction. The coefficients of linear discriminants were sorted, and the top 20 features were selected, resulting in a dataset saved as "cluster_lda_train.csv." This clustered sampling method, followed by multi-step feature selection, aimed to improve class balance and highlight the most informative features for model training.
4. The evaluation process begins with defining metrics to assess model performance. The Matthews Correlation Coefficient (MCC) is calculated from the confusion matrix, considering True Positives, True Negatives, False Positives, and False Negatives. Additionally, an AUC-ROC calculation function is implemented to evaluate the model's ability to distinguish between the target classes. Both MCC and AUC-ROC values provide robust insights into model quality. For model performance metrics, weighted calculations are applied using class distributions, considering metrics like True Positive Rate, False Positive Rate, Precision, Recall, F-measure, MCC, and Kappa Statistic for each class (Class 0 and Class 1). The results are presented separately for each class, and an overall weighted average across classes is also provided. The evaluate_model function, designed to streamline this process, computes these metrics for a given model using the test dataset. Predictions are made on the test data, with separate confusion matrices generated for each class to facilitate detailed metric extraction. This function also ensures that both classes are represented in the test set, essential for AUC-ROC calculations.

5. The function reads multiple preprocessed training datasets, including those from Random Under-Sampling with Boruta ("rus_boruta_train.csv"), Linear Discriminant Analysis ("rus_lda_train.csv"), and Random Forest feature selection ("rus_info_gain_train.csv"). Clustered versions of these datasets (e.g., "cluster_boruta_train.csv," "cluster_lda_train.csv," "cluster_info_gain.csv") are also loaded. After loading, the target variable in each dataset is converted to a factor with consistent labeling for compatibility with the evaluation functions. Finally, the class distributions in key training datasets, including RUS and clustered data with feature selection, are printed to verify class balance.
6. Classification procedures included a common framework use, wherein a classifier was tuned using their respective grids, trained on all the subsets of data and then tested on the testing dataset. In addition to the same, repeated cross validation was used.
    a. In this K-Nearest Neighbors (KNN) model setup, hyperparameter tuning was conducted through a grid search focusing on the number of neighbors (kmax), evaluated from 1 to 20 in steps of 2. The Manhattan distance metric was selected for measuring proximity (distance = 1), as it calculates distances based on absolute differences, which can be effective in high-dimensional data. A rectangular kernel was applied, treating all neighbors within the specified range equally. This configuration allowed the model to explore a range of neighbor counts to find the optimal balance between model bias and variance.
    b. A rpart decision tree model was implemented and hyperparameter tuning was conducted through a grid search across a range of complexity parameter (cp) values. The cp parameter controls the minimum improvement in the model's accuracy needed to justify further splits in the tree. Values for cp were evaluated from 0.001 to 0.05 in increments of 0.01, allowing the model to explore a range of pruning intensities. Lower values of cp encourage deeper trees by allowing more splits, while higher values lead to simpler trees by pruning more aggressively.
    c. In this AdaBoost (ada) model setup, hyperparameter tuning was performed using a grid search across three essential parameters: the number of boosting iterations (iter), the maximum depth of the base decision trees (maxdepth), and the learning rate (nu). The iter parameter, which defines the number of boosting rounds, was tested with values of 20, 50, and 100 to explore different levels of model boosting, allowing for varying degrees of ensemble strength. For the maximum depth (maxdepth) of the base trees, values of 3, 5, and 7 were evaluated to identify the optimal level of tree complexity, balancing between simple, shallow trees and more complex ones. The learning rate (nu), controlling the contribution of each tree in the boosting process, was tuned with values of 0.1, 0.3, and 0.5. Lower values for nu allow for more gradual learning and can improve model stability, while higher values speed up convergence.

d. In the Random Forest classification procedure, a repeated cross-validation control setup is used with 10 folds and 5 repetitions. This configuration is complemented by class probabilities and a summary function focusing on ROC (Receiver Operating Characteristic) to optimize the model's performance for binary classification. A grid search is performed over various values of mtry (from 2 to 11) to determine the optimal number of variables considered at each split. The model is set with a class weighting parameter to account for class imbalance, assigning a 0.9 weight to the majority class and 0.1 to the minority class. Each Random Forest model is trained with 500 trees (ntree = 500).
e. A support vector machine (SVM) using Radial Basis Function model with hyperparameter tuning is prepared as well. A cross-validation control object (train_control) is created, specifying repeated cross-validation as the validation method, with 5 folds repeated 3 times to enhance the model's reliability through multiple training and validation cycles. The twoClassSummary function is used as the summary metric, focusing on binary classification evaluation metrics like ROC, sensitivity, and specificity, while setting classProbs = TRUE to ensure probability estimates are included in the results. A grid search is defined over a range of C and sigma values for the SVM model. These parameters control the model's regularization and the Gaussian kernel's spread, respectively. By setting C and sigma to powers of 2 between -2 and 2, the grid allows the model to test both lower and higher regularization and kernel bandwidth values.
f. An XGBoost model was also prepared for hyperparameter tuning, utilizing a cross-validation strategy. The train_control object specifies repeated cross-validation with 5 folds repeated 3 times to increase reliability through multiple validation rounds. The model's performance will be evaluated using the twoClassSummary function, which calculates key metrics for binary classification. The grid search defined for hyperparameter tuning includes a range of values for key XGBoost parameters. For boosting iterations (nrounds), values of 100 and 200 are specified, allowing evaluation of model performance across different training durations. The max_depth parameter, set to 3, 6, and 9, explores models of varying complexity, while the learning rate (eta) is tested with values of 0.01, 0.1, and 0.3, balancing between faster convergence and more gradual, robust learning. Regularization strength is controlled by gamma, with values of 0 and 1, and the model samples features per tree (colsample_bytree) at fractions of 0.5 and 0.7 to assess model robustness across feature subsets. Other parameters include min_child_weight, which controls the minimum sample requirements in a child node with values of 1 and 5, and subsample, tested at 0.7 and 1, to evaluate model stability under varying sample proportions.

6. Results and Evaluation

Models were trained to emphasize on ROC, which was used to select the optimal model using the largest value. Results are presented below with a table describing the True Positive Rate, False Positive Rate, Precision, Recall, F-Score, AUC-ROC, MCC and Kappa statistic. This is calculated for each class and for a weighted average based on class counts of the test set is used. A ROC curve is generated and presented below each table.

6. 1. KNN

Sampling Method: Random Under Sampling

Feature Selection Method: LDA

The final values used for the model were kmax = 19, distance = 1 and kernel = rectangular.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.986	0.983	0.881	0.986	0.930	0.501	0.008	0.005
Class Y	0.016	0.013	0.142	0.016	0.030	0.501	0.008	0.005
Wt. avg	0.871	0.868	0.793	0.871	0.823	0.501	0.008	0.005

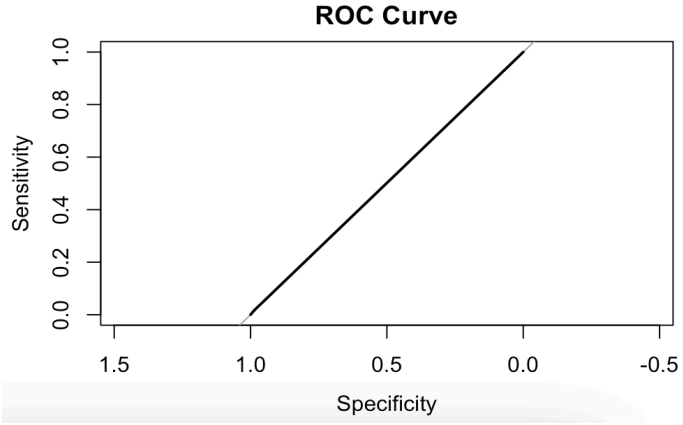

Figure 4: ROC Curve for KNN on LDA for Random Undersampling

Feature Selection Method: Info Gain

The final values used for the model were kmax = 19, distance = 1 and kernel = rectangular.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.761	0.286	0.951	0.761	0.845	0.816	0.337	0.289
Class Y	0.713	0.238	0.287	0.713	0.410	0.816	0.337	0.289
Wt. avg	0.756	0.281	0.873	0.756	0.794	0.816	0.337	0.29

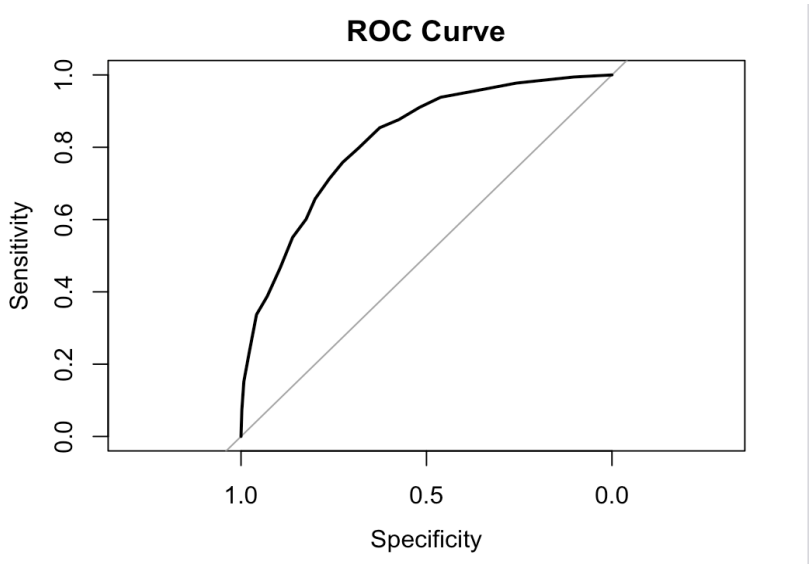

Figure 5: ROC Curve for KNN on Information Gain for Random Undersampling

Feature Selection Method: Boruta

The final values used for the model were kmax = 13, distance = 1 and kernel = rectangular.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.749	0.303	0.948	0.749	0.837	0.811	0.314	0.266
Class Y	0.696	0.250	0.273	0.696	0.392	0.811	0.314	0.266
Wt. avg	0.743	0.297	0.868	0.743	0.784	0.811	0.314	0.267

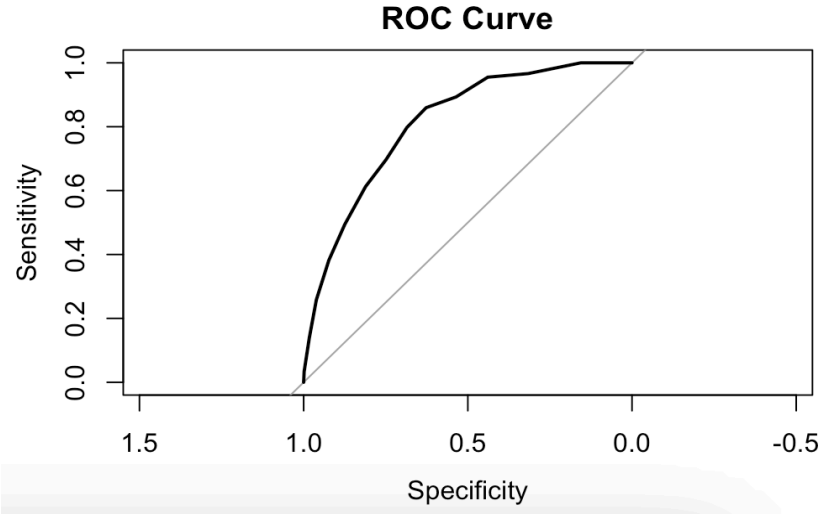

Figure 6: ROC Curve for KNN on Boruta for Random Undersampling

Sampling Method: Clustering

Feature Selection Method: LDA

The final values used for the model were kmax = 15, distance = 1 and kernel = rectangular.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.986	0.960	0.883	0.986	0.932	0.513	0.064	0.040
Class Y	0.039	0.013	0.28	0.039	0.068	0.513	0.064	0.040
Wt. avg	0.873	0.848	0.812	0.873	0.829	0.513	0.064	0.040

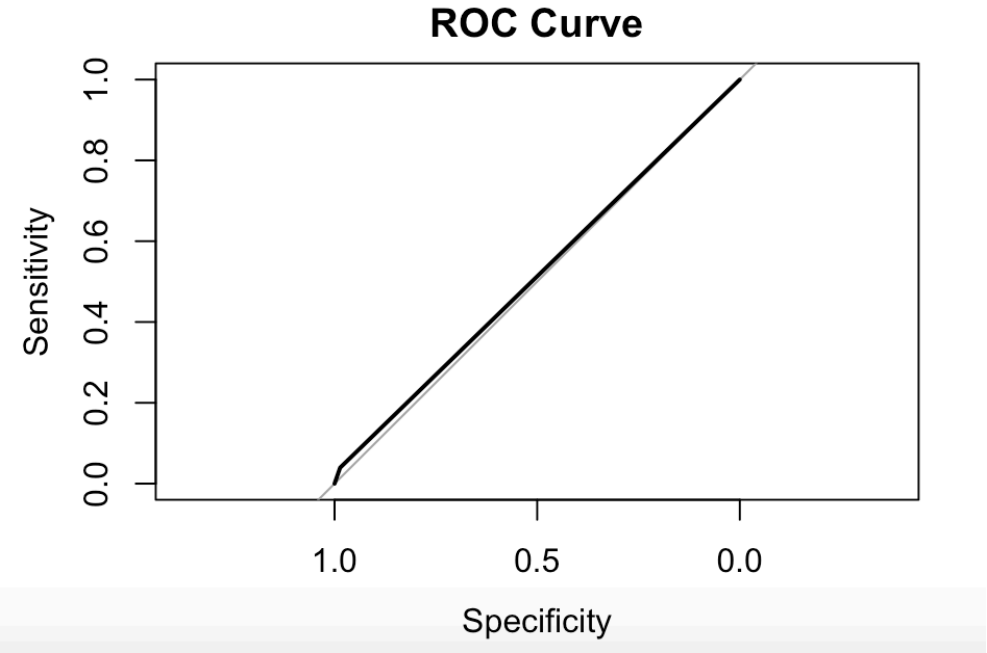

Figure 7: ROC Curve for KNN on LDA for Clustered Dataset

Feature Selection Method: Info Gain

The final values used for the model were kmax = 17, distance = 1 and kernel = rectangular.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.760	0.32	0.946	0.760	0.843	0.787	0.314	0.271
Class Y	0.679	0.239	0.277	0.679	0.394	0.787	0.314	0.271
Wt. avg	0.751	0.311	0.866	0.751	0.79	0.787	0.314	0.271

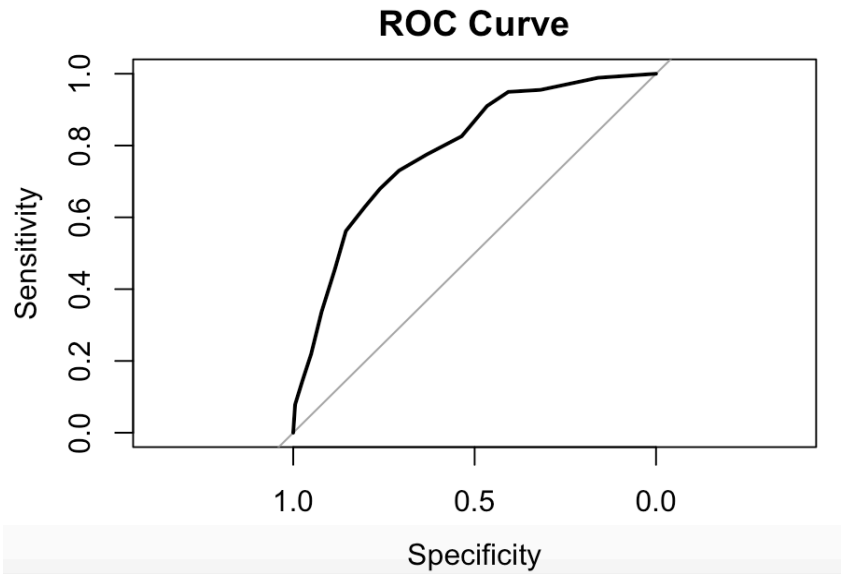

Figure 8: ROC Curve for KNN on Information Gain for Clustered Dataset

Feature Selection Method: Boruta

The final values used for the model were kmax = 13, distance = 1 and kernel = rectangular.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.766	0.353	0.941	0.766	0.845	0.790	0.297	0.259
Class Y	0.646	0.233	0.272	0.646	0.383	0.790	0.297	0.259
Wt. avg	0.752	0.34	0.862	0.752	0.79	0.790	0.297	0.259

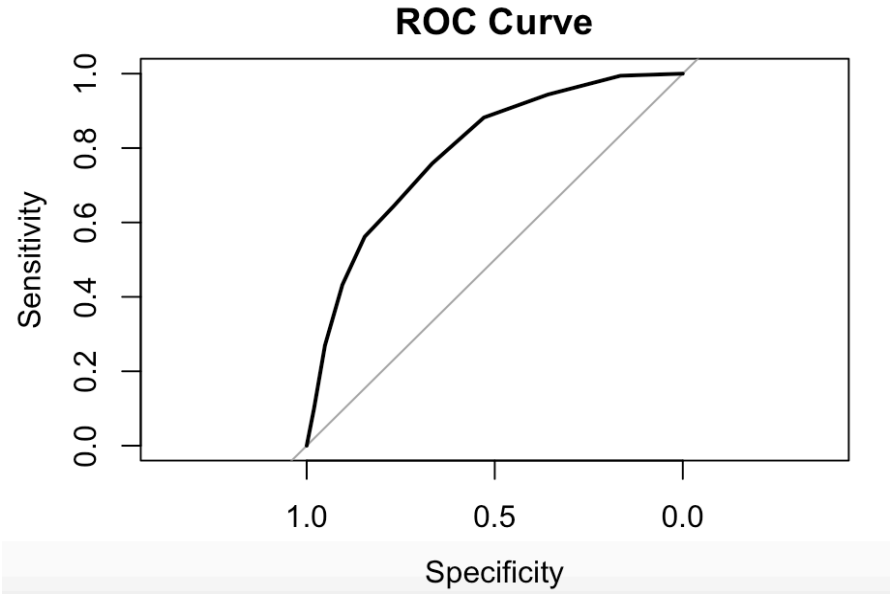

Figure 9: ROC Curve for KNN on Boruta for Clustered Dataset

6. 2. SVM

Sampling Method: Random Under Sampling

Feature Selection Method: LDA
Final model parameters were sigma = 4 and c = 0.25

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.934	0.916	0.883	0.934	0.908	0.509	0.023	0.022
Class Y	0.084	0.066	0.147	0.084	0.107	0.509	0.023	0.022
Wt. avg	0.833	0.815	0.795	0.833	0.812	0.509	0.023	0.022

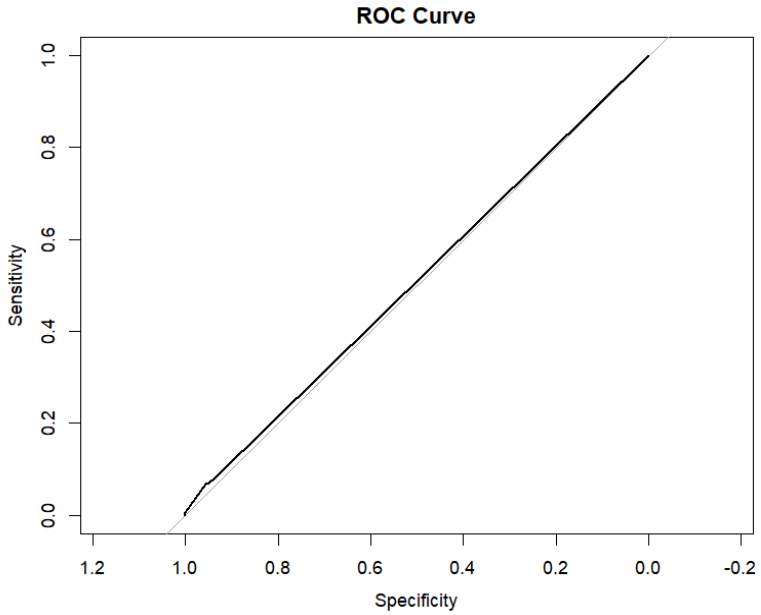

Figure 10: ROC Curve for SVM on LDA for Random Undersampling

Feature Selection Method: Information Gain
Final model parameters were sigma = 0.25 and c = 5

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.677	0.084	0.983	0.677	0.802	0.907	0.393	0.297
Class Y	0.916	0.323	0.277	0.916	0.426	0.907	0.393	0.297
Wt. avg	0.705	0.113	0.899	0.705	0.757	0.907	0.393	0.297

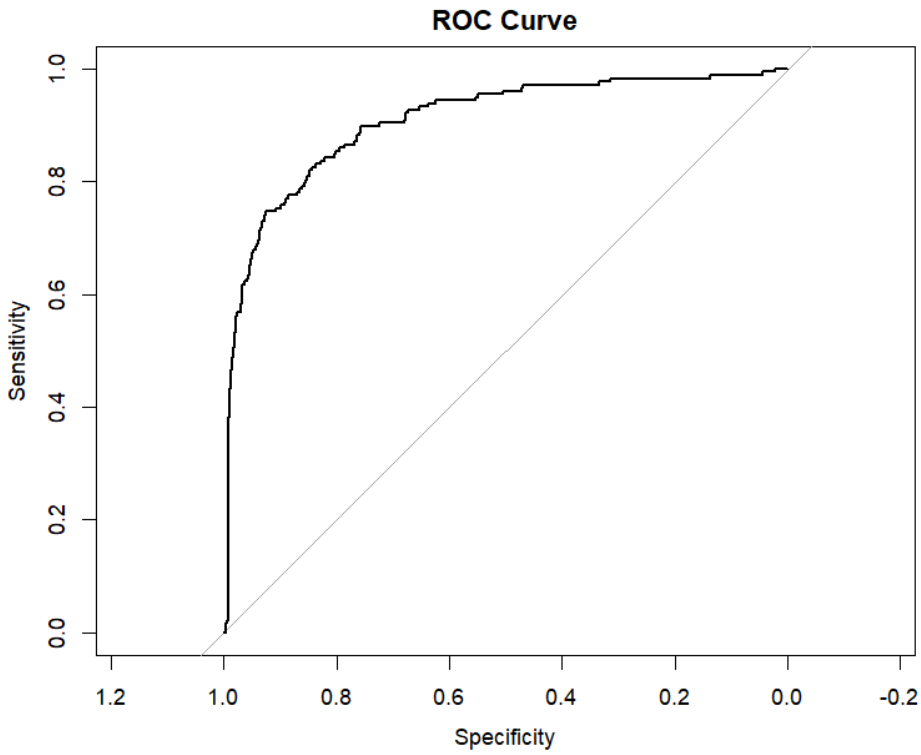

Figure 11: ROC Curve for SVM on Information Gain for Random Undersampling

Feature Selection Method: Boruta
Final model parameters were sigma = 0.25 and c = 0.5

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.549	0.056	0.986	0.549	0.706	0.933	0.32	0.204
Class Y	0.944	0.451	0.221	0.944	0.358	0.933	0.32	0.204
Wt. avg	0.596	0.103	0.895	0.596	0.664	0.933	0.32	0.204

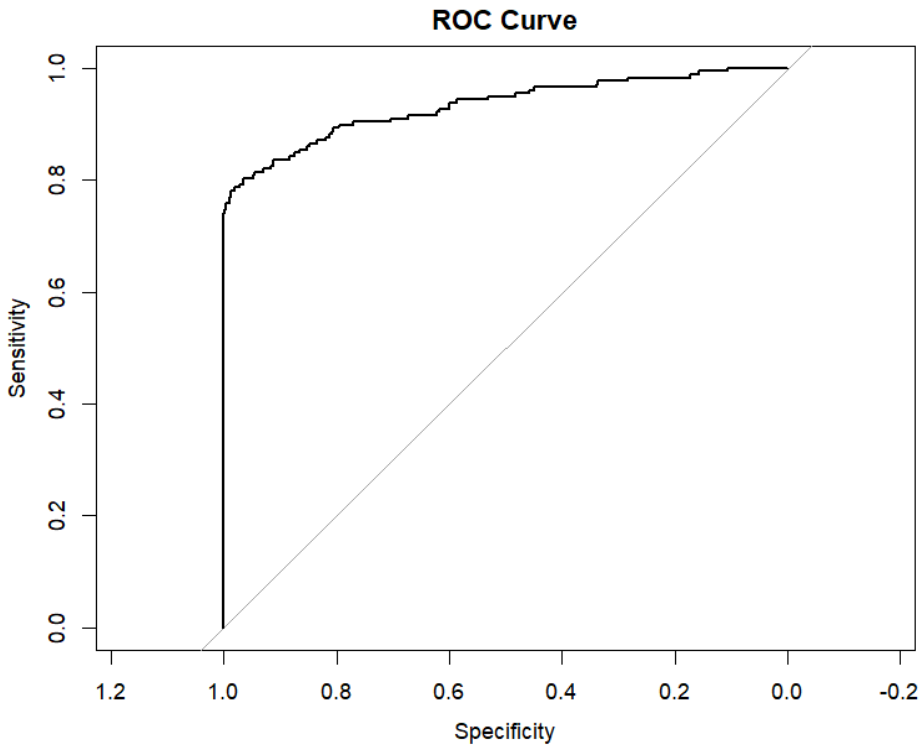

Figure 12: ROC Curve for SVM on Boruta for Random Undersampling

Sampling Method: Clustering

Feature Selection Method: LDA
Final parameter values were c = 2 and sigma = 0.5

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.944	0.91	0.885	0.944	0.913	0.531	0.046	0.043
Class Y	0.09	0.056	0.178	0.09	0.119	0.531	0.046	0.043
Wt. avg	0.842	0.808	0.8	0.842	0.819	0.531	0.046	0.043

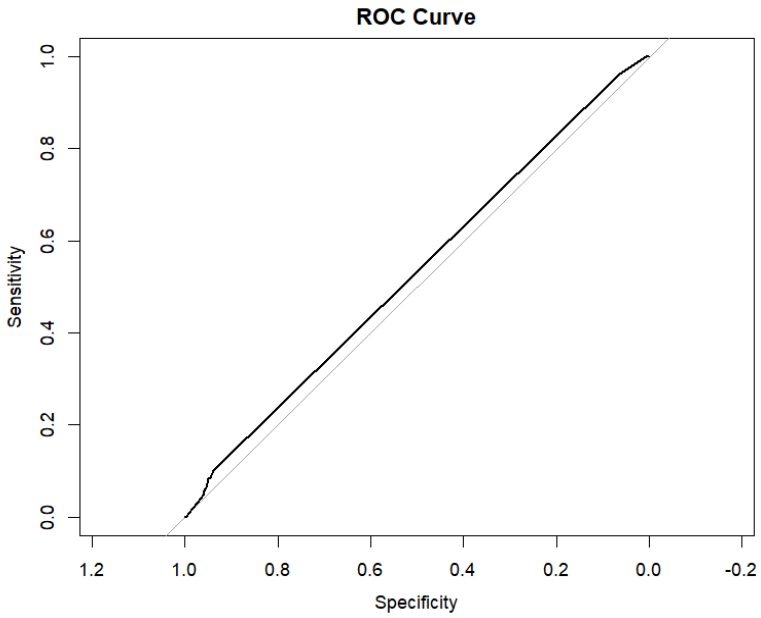

Figure 13: ROC Curve for SVM on LDA for Clustered Dataset

Feature Selection Method: Information Gain
Final parameter values are c = 0.25 and sigma = 0.25

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.696	0.146	0.972	0.696	0.811	0.842	0.369	0.288
Class Y	0.854	0.304	0.275	0.854	0.416	0.842	0.369	0.288
Wt. avg	0.715	0.165	0.889	0.715	0.746	0.842	0.369	0.288

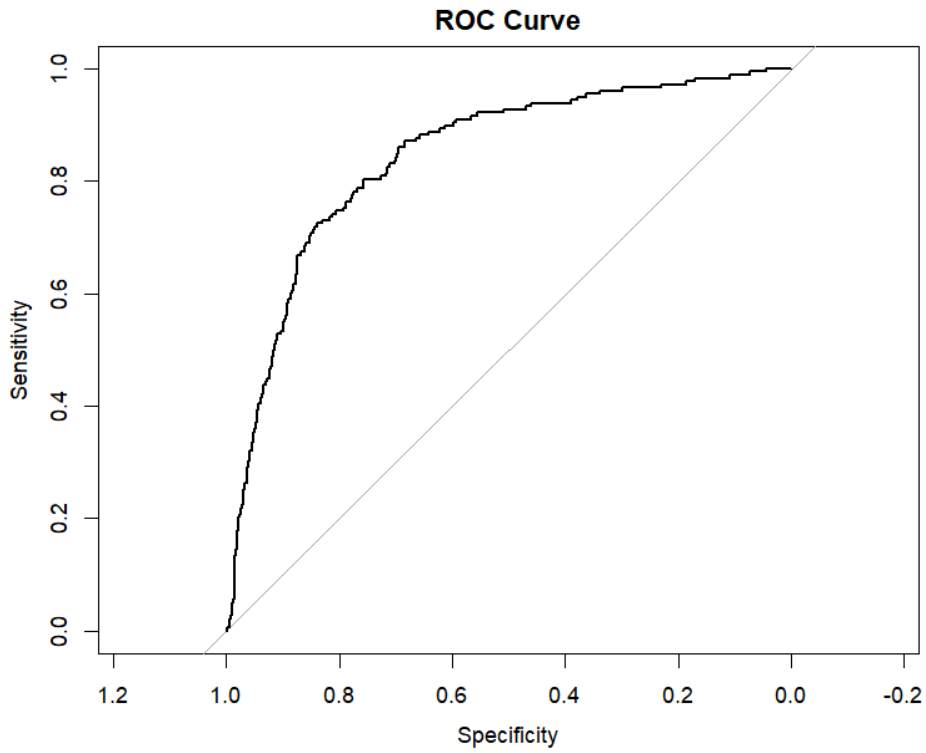

Figure 14: ROC Curve for SVM on Information Gain for Clustered Dataset

Feature Selection Method: Boruta
Final model parameters were c = 0.25 and sigma = 0.5

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.47	0.079	0.978	0.47	0.635	0.894	0.257	0.147
Class Y	0.921	0.53	0.19	0.921	0.316	0.894	0.257	0.147
Wt. avg	0.524	0.132	0.884	0.524	0.597	0.894	0.257	0.147

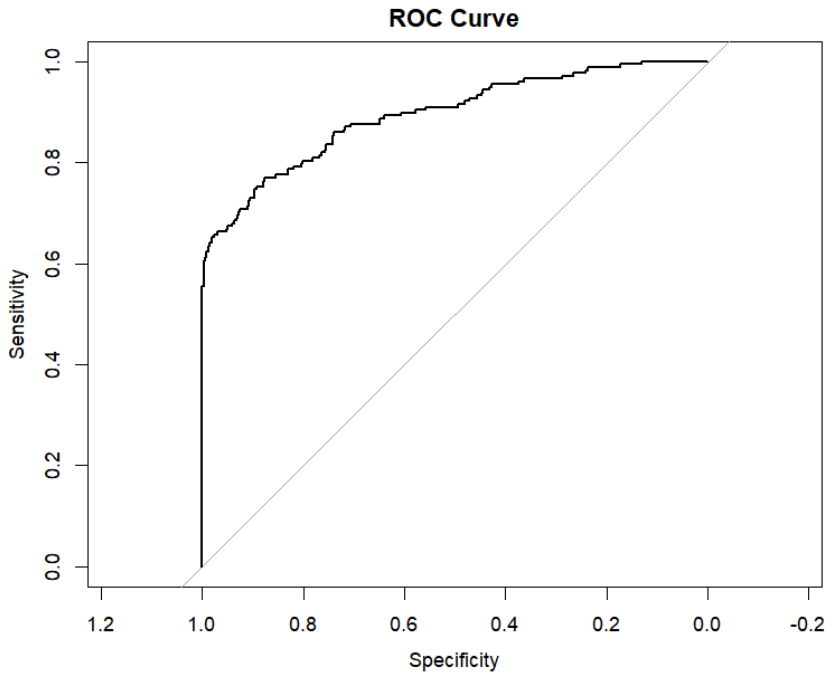

Figure 15: ROC Curve for SVM on Boruta for Clustered Dataset

6. 3. Decision Tree

Sampling Method: Random Under Sampling

Feature Selection Method: LDA
The final value used for the model was cp = 0.001.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.952	0.938	0.882	0.952	0.915	0.507	0.021	0.019
Class Y	0.061	0.047	0.148	0.061	0.087	0.507	0.021	0.019
Wt. avg	0.846	0.832	0.795	0.846	0.817	0.507	0.021	0.019

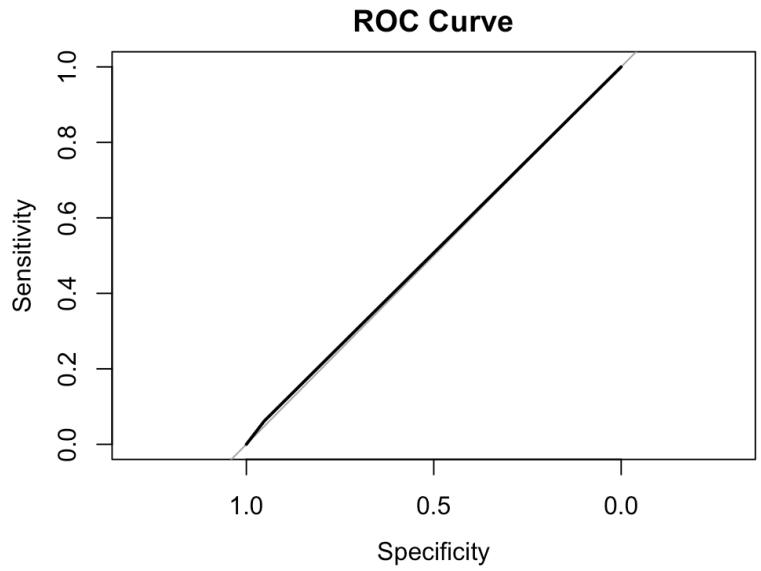

Figure 16: ROC Curve for Decision Tree on LDA for Random Undersampling

Feature Selection Method: Info Gain

The final value used for the model was cp = 0.001.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.733	0.151	0.972	0.733	0.836	0.853	0.398	0.325
Class Y	0.848	0.266	0.300	0.848	0.444	0.853	0.398	0.325
Wt. avg	0.747	0.165	0.893	0.747	0.79	0.853	0.398	0.325

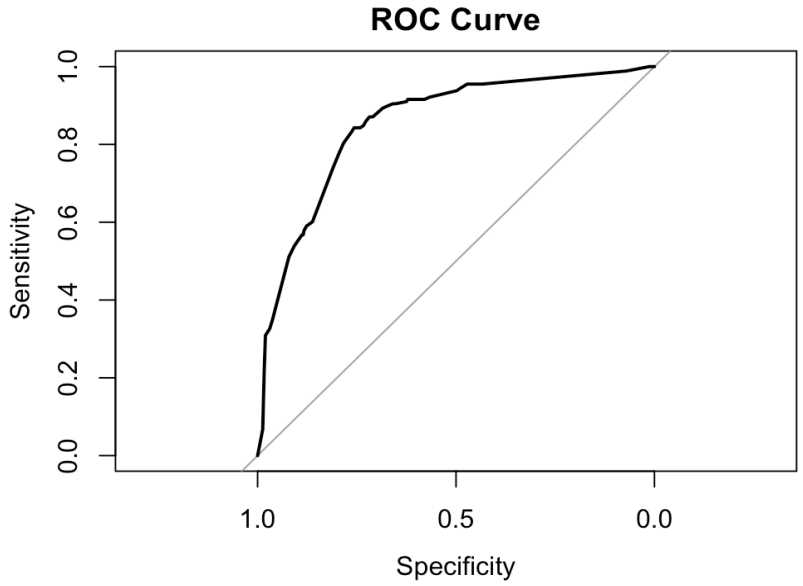

Figure 17: ROC Curve for Decision Tree on Information Gain for Random Undersampling

Feature Selection Method: Boruta
The final value used for the model was cp = 0.001.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.735	0.213	0.962	0.735	0.833	0.832	0.360	0.297
Class Y	0.786	0.264	0.286	0.286	0.786	0.832	0.360	0.297
Wt. avg	0.742	0.22	0.882	0.742	0.785	0.832	0.361	0.297

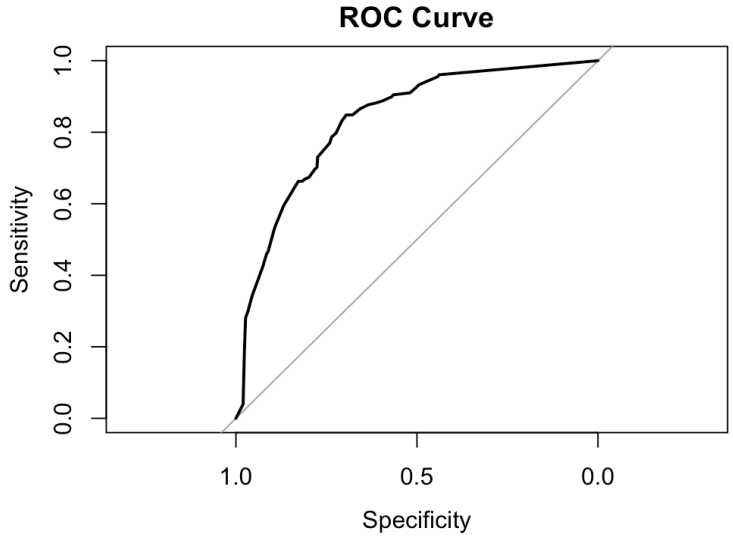

Figure 18: ROC Curve for Decision Tree on Boruta for Random Undersampling

Sampling Method: Clustering

Feature Selection Method: LDA
The final value used for the model was cp = 0.011.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.948	0.932	0.882	0.948	0.914	0.507	0.023	0.021
Class Y	0.067	0.051	0.15	0.067	0.093	0.507	0.023	0.021
Wt. avg	0.843	0.828	0.795	0.843	0.795	0.507	0.023	0.021

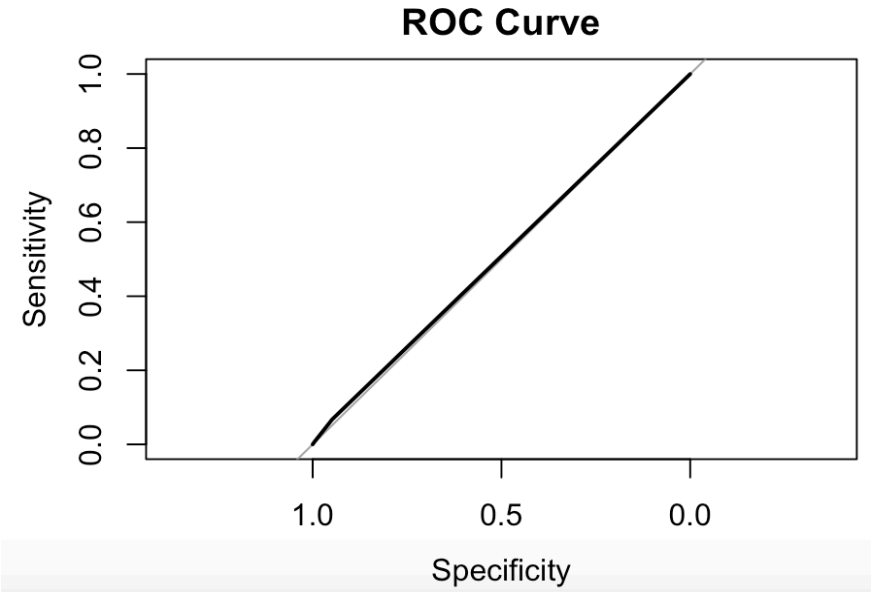

Figure 19: ROC Curve for Decision Tree on LDA for Clustered Dataset

Feature Selection Method: Info Gain

The final value used for the model was cp = 0.001.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.707	0.185	0.965	0.707	0.816	0.819	0.353	0.281
Class Y	0.814	0.292	0.273	0.814	0.409	0.819	0.353	0.281
Wt. avg	0.72	0.198	0.883	0.72	0.768	0.819	0.353	0.281

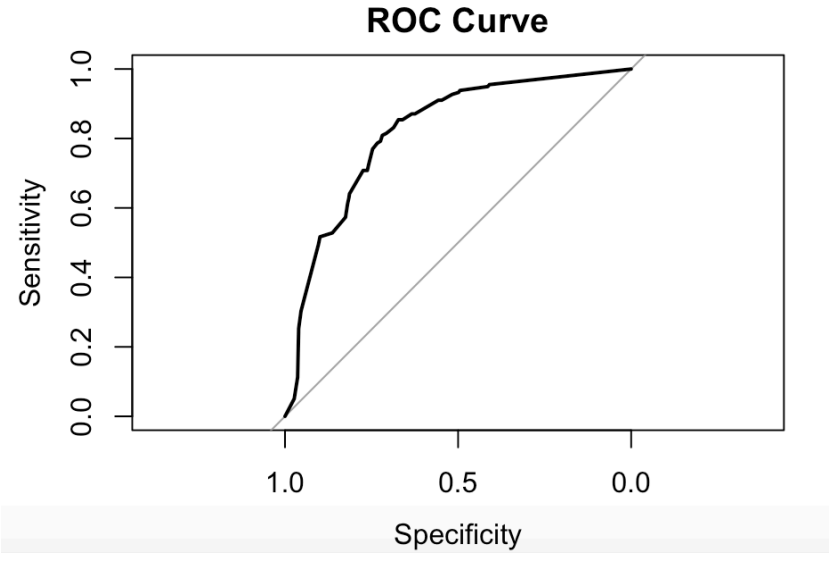

Figure 20: ROC Curve for Decision Tree on Information Gain for Clustered Dataset

Feature Selection Method: Boruta
The final value used for the model was cp = 0.001.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.689	0.179	0.965	0.689	0.804	0.783	0.342	0.267
Class Y	0.820	0.310	0.263	0.82	0.398	0.783	0.342	0.267
Wt. avg	0.705	0.195	0.882	0.705	0.757	0.783	0.342	0.267

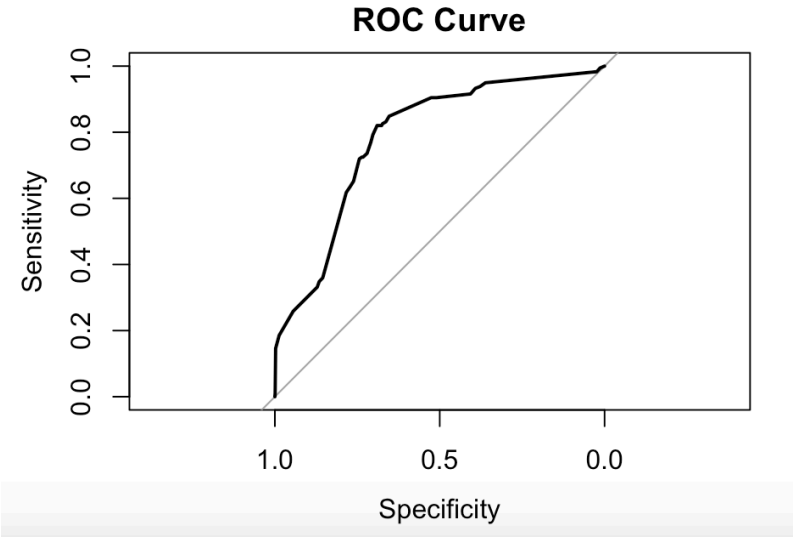

Figure 21: ROC Curve for Decision Tree on Boruta for Clustered Dataset

6. 4. AdaBoost

Sampling Method: Random Under Sampling

Feature Selection Method: LDA
The final values used for the model were iter = 20, maxdepth = 7 and nu = 0.5.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.942	0.926	0.882	0.942	0.911	0.513	0.021	0.019
Class Y	0.073	0.057	0.146	0.073	0.097	0.513	0.021	0.019
Wt. avg	0.839	0.823	0.795	0.839	0.814	0.513	0.021	0.019

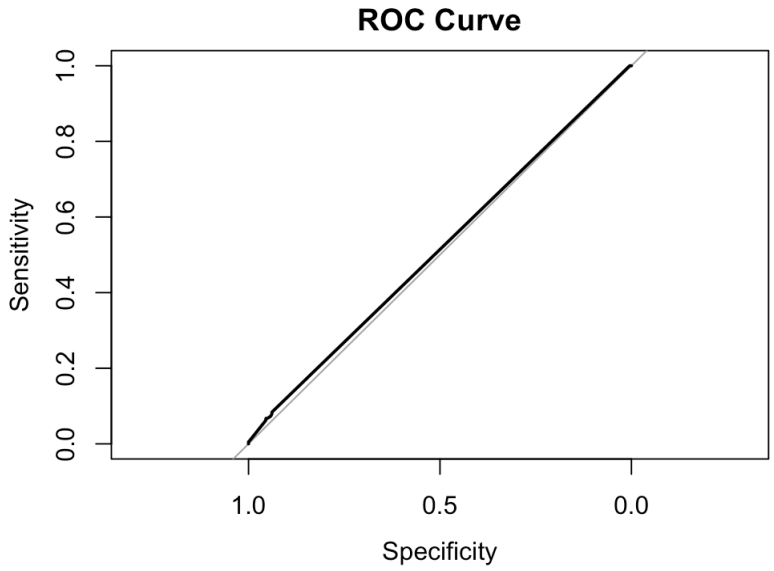

Figure 22: ROC Curve for Adaboost on LDA for Random Undersampling

Feature Selection Method: Info Gain

The final values used for the model were iter = 50, maxdepth = 3 and nu = 0.1.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.724	0.219	0.960	0.724	0.826	0.843	0.347	0.283
Class Y	0.780	0.275	0.277	0.780	0.409	0.843	0.347	0.283
Wt. avg	0.732	0.226	0.879	0.732	0.777	0.843	0.347	0.283

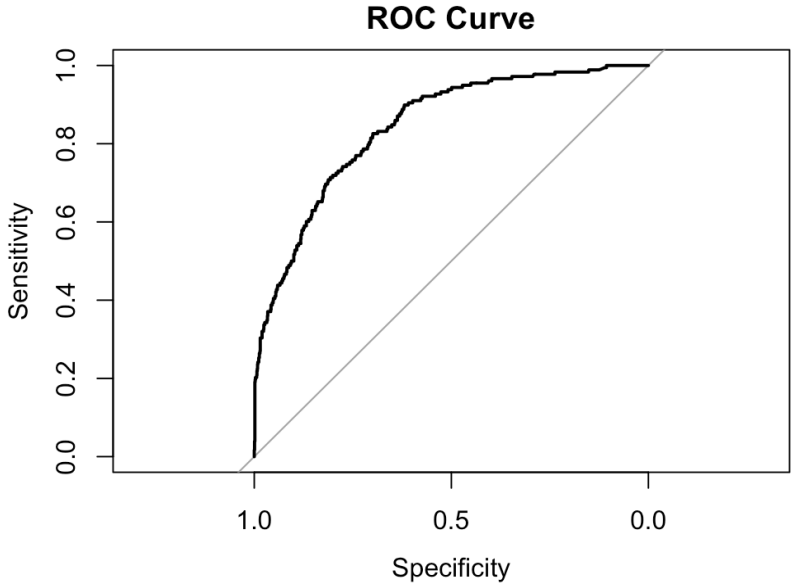

Figure 23: ROC Curve for Adaboost on Information Gain for Random Undersampling

Feature Selection Method: Boruta
The final values used for the model were iter = 100, maxdepth = 3 and nu = 0.1.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.746	0.213	0.962	0.746	0.840	0.858	0.37	0.309
Class Y	0.786	0.253	0.295	0.786	0.429	0.858	0.37	0.309
Wt. avg	0.751	0.218	0.883	0.751	0.792	0.858	0.37	0.309

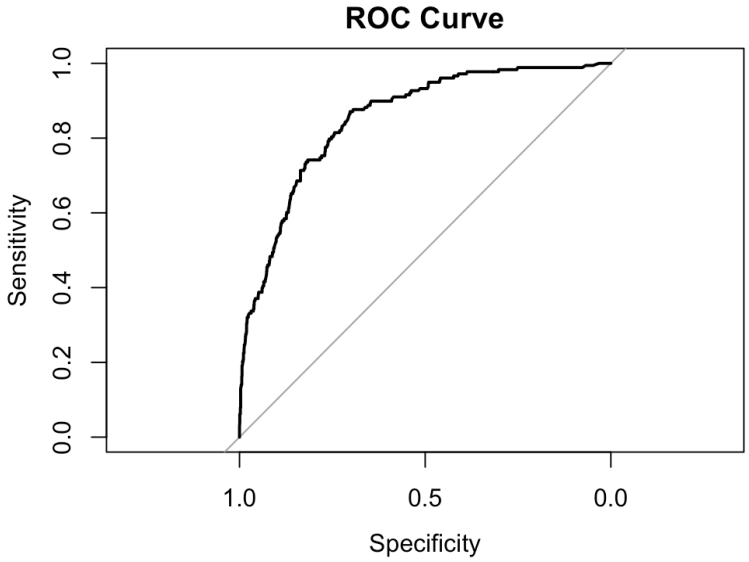

Figure 24: ROC Curve for Adaboost on Boruta for Random Undersampling

Sampling Method: Clustering

Feature Selection Method: LDA
ROC was used to select the optimal model using the largest value.
The final values used for the model were iter = 100, maxdepth = 7 and nu = 0.5.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.943	0.91	0.884	0.943	0.913	0.526	0.046	0.043
Class Y	0.089	0.056	0.177	0.089	0.119	0.526	0.046	0.043
Wt. avg	0.842	0.808	0.8	0.842	0.819	0.526	0.046	0.043

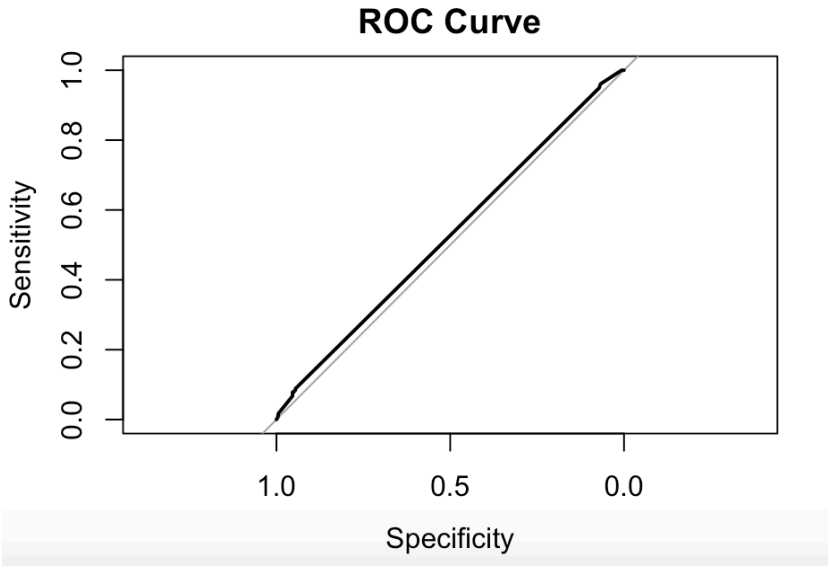

Figure 25: ROC curve for Adaboost on LDA for Clustered Dataset

Feature Selection Method: Info Gain

The final values used for the model were iter = 100, maxdepth = 3 and nu = 0.3.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.693	0.207	0.961	0.693	0.805	0.82	0.327	0.257
Class Y	0.792	0.306	0.259	0.792	0.39	0.82	0.327	0.257
Wt. avg	0.705	0.22	0.877	0.705	0.756	0.82	0.327	0.257

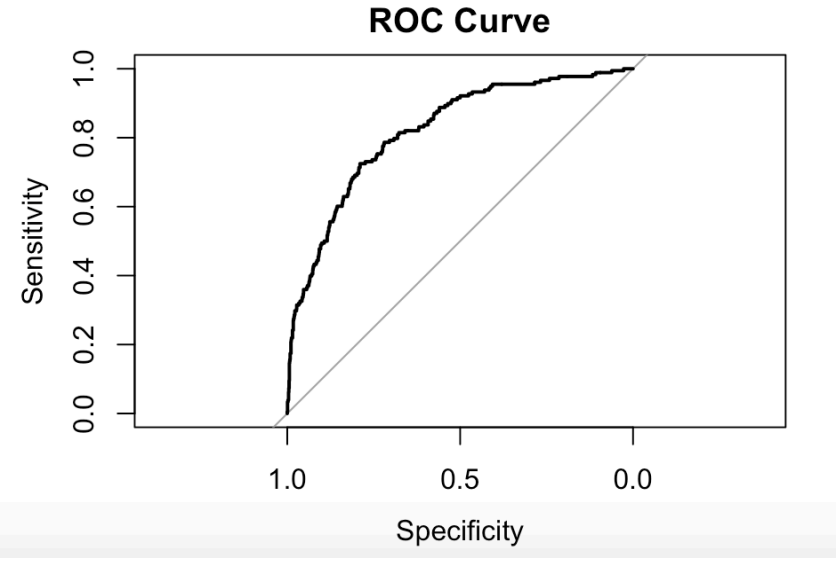

Figure 26: ROC curve for Adaboost on Information Gain for Clustered Dataset

Feature Selection Method: Boruta
The final values used for the model were iter = 100, maxdepth = 3 and nu = 0.5.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.680	0.191	0.963	0.68	0.797	0.825	0.327	0.252
Class Y	0.319	0.254	0.808	0.387	0.387	0.825	0.327	0.252
Wt. avg	0.695	0.206	0.879	0.695	0.749	0.825	0.327	0.252

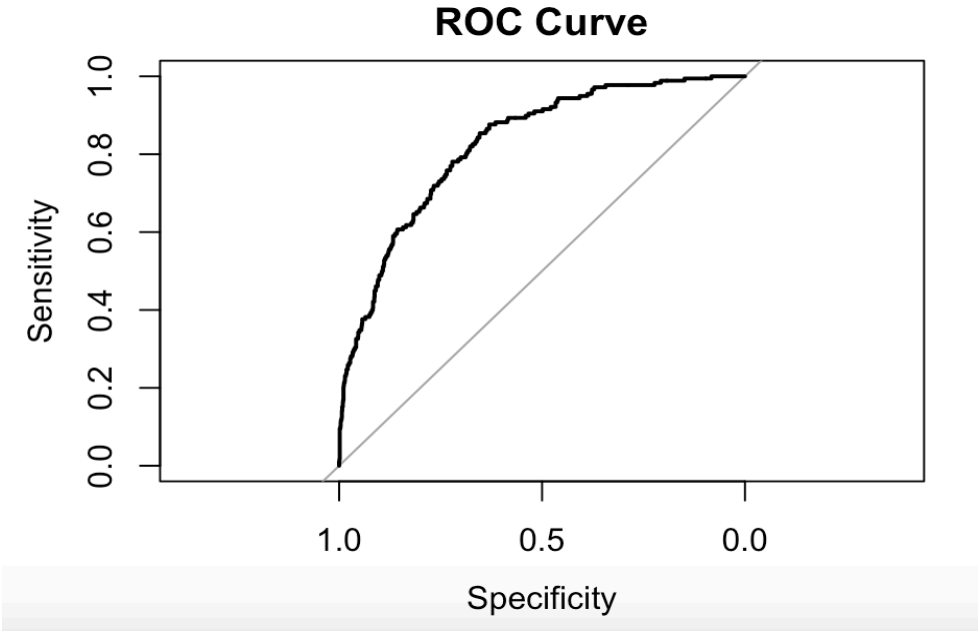

Figure 27: ROC curve for Adaboost on Boruta for Clustered Dataset

6. 5. XGBoost

Sampling Method: Random Under Sampling

Feature Selection Method: LDA
Final model parameters were nrounds = 100, max_depth = 3, eta = 0.01, gamma = 0, colsample_bytree = 0.5, min_child_weight = 5 and subsample = 0.7

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.952	0.938	0.882	0.952	0.916	0.506	0.021	0.019
Class Y	0.062	0.048	0.149	0.062	0.087	0.506	0.021	0.019
Wt. avg	0.846	0.832	0.795	0.846	0.817	0.506	0.021	0.019

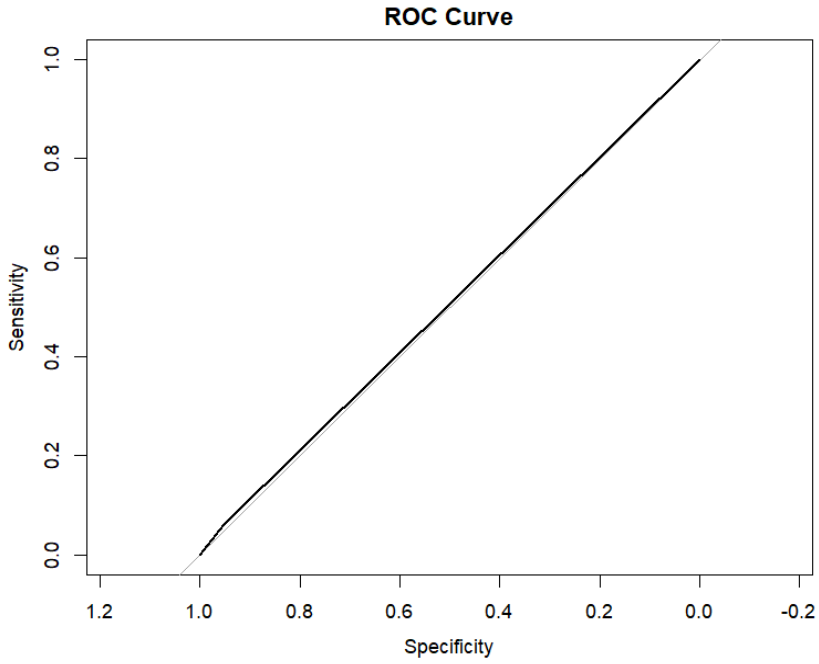

Figure 28: ROC curve for XGBoost on LDA for Random Undersampling

Feature Selection Method: Information Gain
Final model parameters were nrounds = 200, max_depth = 3, eta = 0.01, gamma = 1, colsample_bytree = 0.5, min_child_weight = 1 and subsample = 1.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.724	0.18	0.968	0.724	0.828	0.851	0.372	0.302
Class Y	0.82	0.276	0.287	0.82	0.425	0.851	0.372	0.302
Wt. avg	0.736	0.191	0.736	0.736	0.78	0.851	0.372	0.302

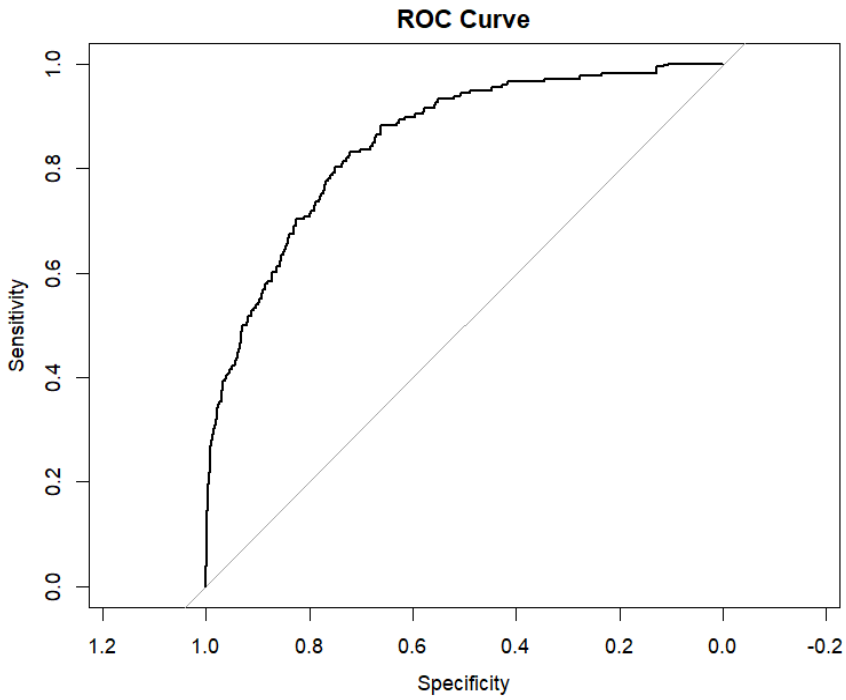

Figure 29: ROC curve for XGBoost on Information Gain for Random Undersampling

Feature Selection Method: Boruta
Final model parameter values are nrounds = 100, max_depth = 3, eta = 0.1, gamma = 1, colsample_bytree = 0.5, min_child_weight = 5 and subsample = 1

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.748	0.14	0.975	0.748	0.846	0.874	0.42	0.348
Class Y	0.86	0.252	0.315	0.86	0.462	0.874	0.42	0.348
Wt. avg	0.761	0.154	0.897	0.761	0.801	0.874	0.42	0.348

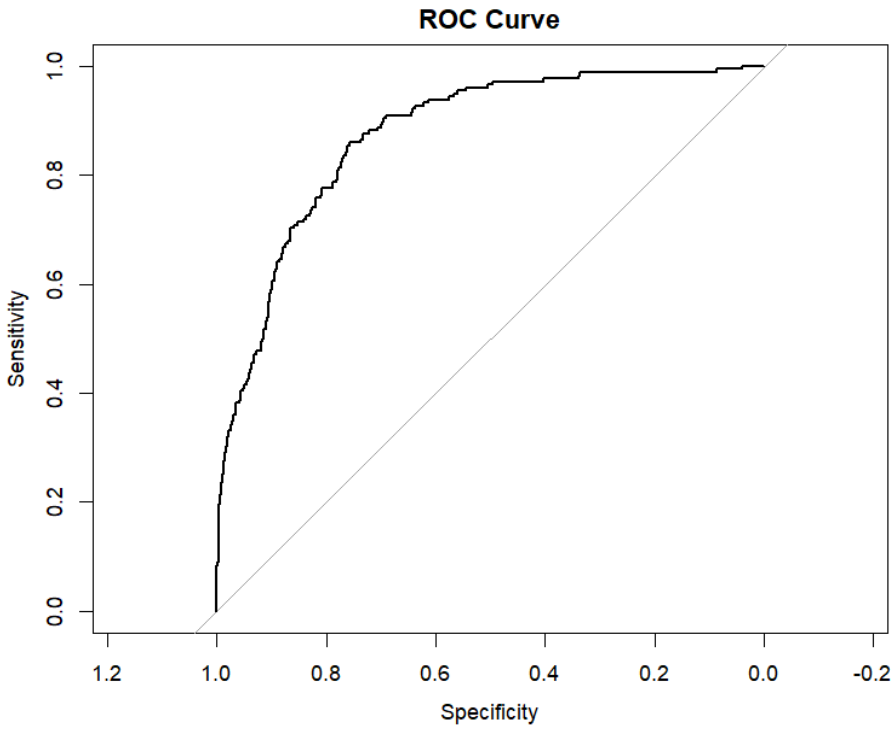

Figure 30: ROC curve for XGBoost on Boruta for Random Undersampling

Sampling Method: Clustering

Feature Selection Method: LDA
Final model parameters were nrounds = 100, max_depth = 3, eta = 0.01, gamma = 1, colsample_bytree = 0.7, min_child_weight = 1 and subsample = 0.7.

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.948	0.927	0.883	0.948	0.914	0.519	0.029	0.027
Class Y	0.073	0.052	0.159	0.073	0.1	0.519	0.029	0.027
Wt. avg	0.843	0.823	0.797	0.843	0.817	0.519	0.029	0.027

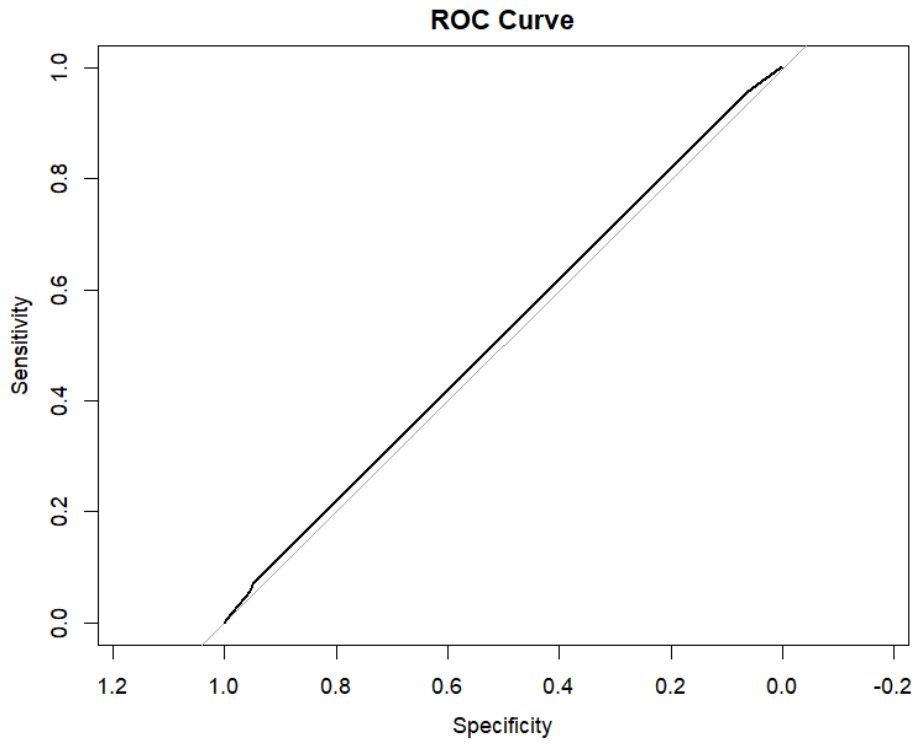

Figure 31: ROC curve for XGBoost on LDA for Clustered Dataset

Feature Selection Method: Information Gain
Final model parameter values were nrounds = 100, max_depth = 3, eta = 0.1, gamma = 0, colsample_bytree = 0.5, min_child_weight = 1 and subsample = 1

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.68	0.191	0.963	0.68	0.797	0.828	0.327	0.252
Class Y	0.809	0.32	0.255	0.809	0.388	0.828	0.327	0.252
Wt. avg	0.695	0.206	0.879	0.695	0.749	0.828	0.327	0.252

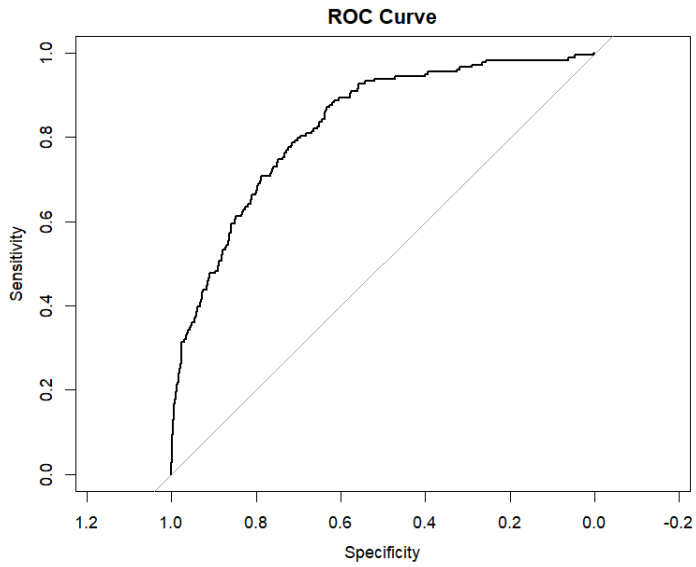

Figure 32: ROC curve for XGBoost on Information Gain for Clustered Dataset

Feature Selection Method: Boruta
Final model parameter values are nrounds = 100, max_depth = 3, eta = 0.1, gamma = 1, colsample_bytree = 0.5, min_child_weight = 5 and subsample = 1

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.682	0.146	0.972	0.682	0.801	0.846	0.357	0.274
Class Y	0.854	0.318	0.266	0.854	0.406	0.846	0.357	0.274
Wt. avg	0.702	0.167	0.888	0.702	0.754	0.846	0.357	0.274

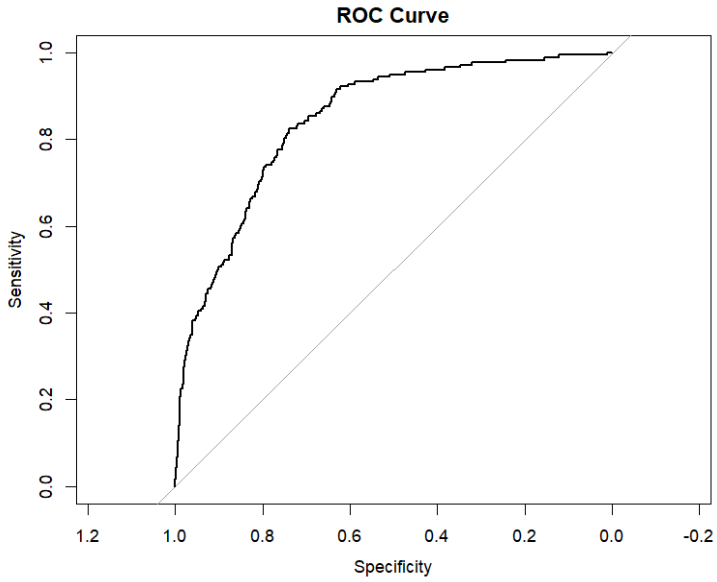

Figure 33: ROC curve for XGBoost on Boruta for Clustered Dataset

6. 6. Random Forest

Sampling Method: Random Under Sampling

Feature Selection Method: LDA
Final parameter values were mtry = 10

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.995	0.989	0.881	0.995	0.935	0.503	0.025	0.01
Class Y	0.011	0.005	0.222	0.011	0.021	0.503	0.025	0.01
Wt. avg	0.878	0.872	0.803	0.878	0.826	0.503	0.025	0.01

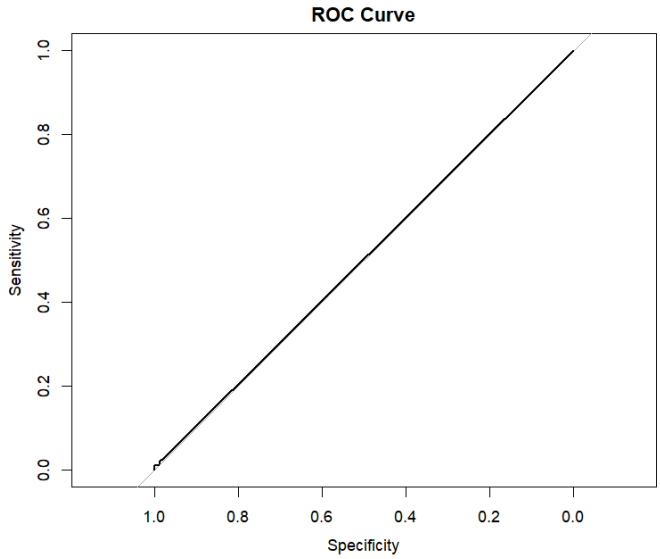

Figure 34: ROC curve for Random Forest on LDA for Random Undersampling

Feature Selection Method: Information Gain
Final parameter values were mtry = 2

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	**0.800**	0.084	0.986	0.8	0.883	0.932	0.514	0.447
Class Y	**0.916**	0.2	0.383	0.916	0.54	0.932	0.514	0.447
Wt. avg	0.814	0.098	0.914	0.814	0.842	0.932	0.514	0.447

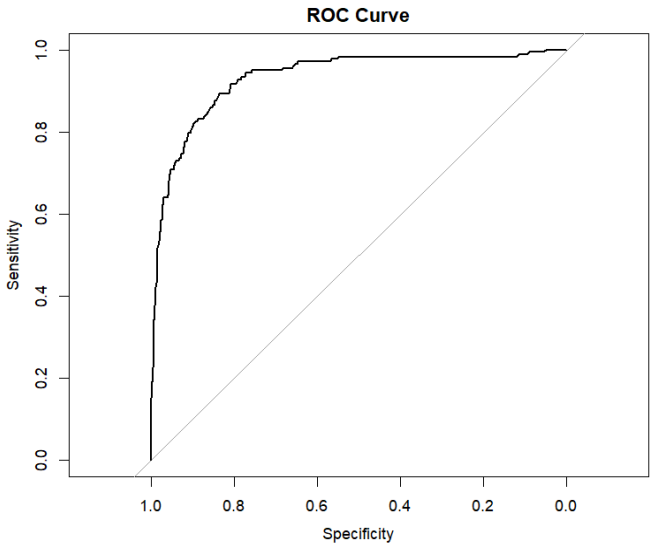

Figure 35: ROC curve for Random Forest on Information Gain for Random Undersampling

Feature Selection Method: Boruta
Final parameter values were mtry = 4

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.758	0.062	0.989	0.758	0.858	0.935	0.482	0.399
Class Y	0.938	0.242	0.344	0.938	0.504	0.935	0.482	0.399
Wt. avg	0.78	0.083	0.912	0.78	0.817	0.935	0.482	0.399

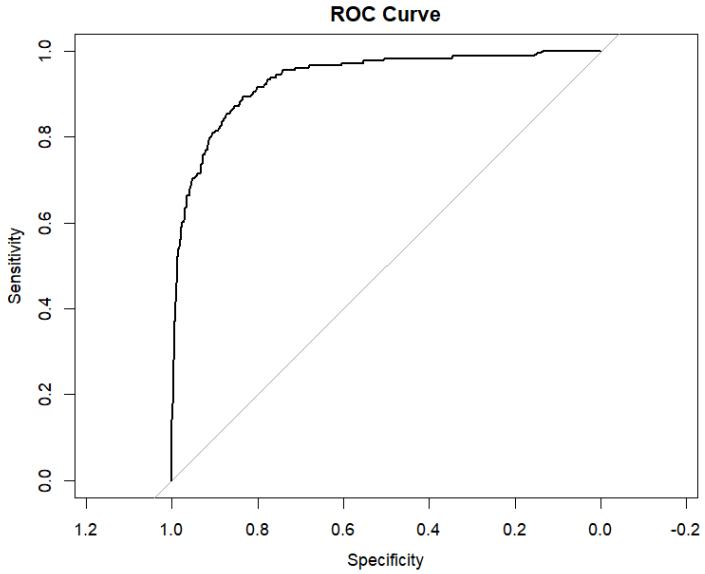

Figure 36: ROC curve for Random Forest on Boruta for Random Undersampling

Sampling Method: Clustering

Feature Selection Method: LDA
Final parameter values were mtry = 9

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.989	0.966	0.883	0.989	0.933	0.512	0.061	0.036
Class Y	0.034	0.011	0.286	0.034	0.06	0.512	0.061	0.036
Wt. avg	0.875	0.853	0.812	0.875	0.829	0.512	0.061	0.036

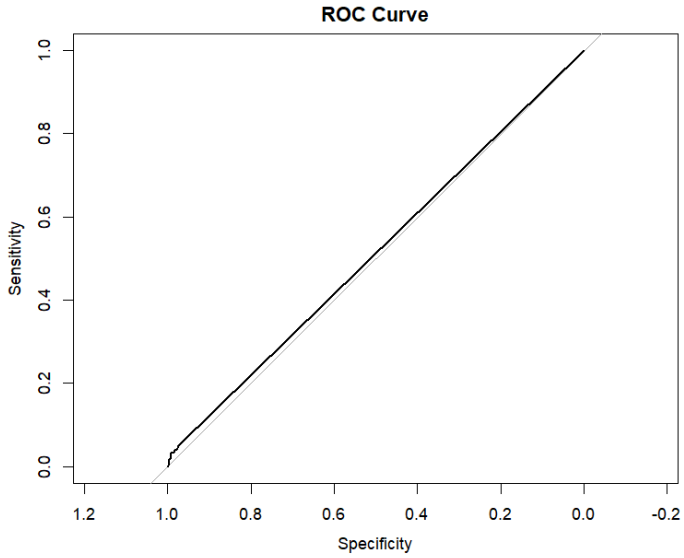

Figure 37: ROC curve for Random Forest on LDA for Clustered Dataset

Feature Selection Method: Information Gain
Final parameter values were mtry = 3

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.711	0.096	0.982	0.711	0.825	0.902	0.415	0.327
Class Y	0.902	0.289	0.298	0.902	0.448	0.902	0.415	0.327
Wt. avg	0.734	0.119	0.901	0.734	0.78	0.902	0.415	0.327

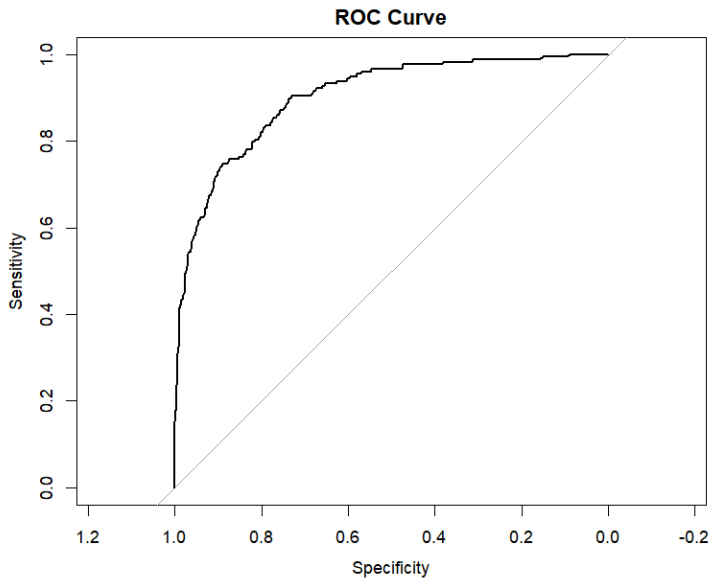

Figure 38: ROC curve for Random Forest on Information Gain for Clustered Dataset

Feature Selection Method: Boruta
Final parameter values were mtry =4

	TPR	FPR	Precision	Recall	F-score	ROC	MCC	Kappa
Class N	0.735	0.107	0.981	0.735	0.84	0.903	0.43	0.349
Class Y	0.893	0.265	0.313	0.893	0.464	0.903	0.43	0.349
Wt. avg	0.754	0.126	0.901	0.754	0.795	0.903	0.43	0.349

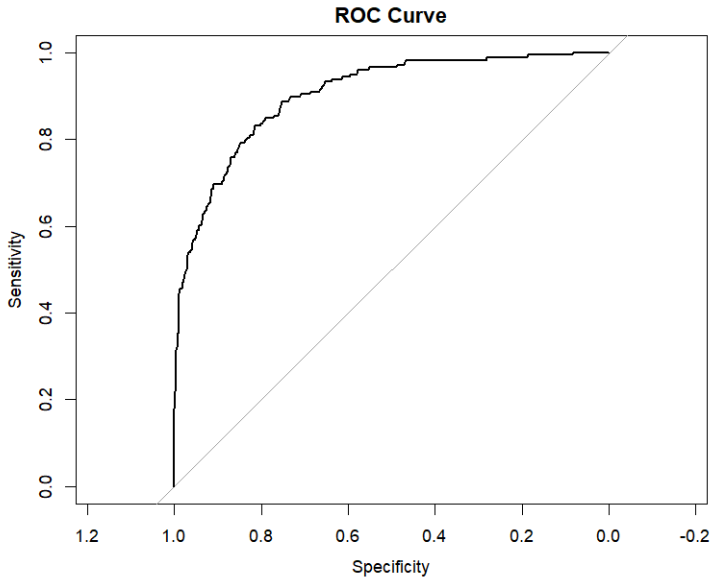

Figure 39: ROC curve for Random Forest on Boruta for Clustered Dataset

7. Best Model and Analysis

Random Undersampling consistently improved minority class recall while keeping weighted metrics like precision, F-score, and MCC high, especially in the Random Forest and AdaBoost models. **Clustering-based sampling**, while effective in retaining majority class patterns, had mixed results on minority recall and overall FPR, with models sometimes achieving high weighted averages but lacking balanced TPRs across classes. Thus, the combination of Random Forest with Information Gain and RUS emerged as the optimal approach, achieving high performance across metrics and underscoring the importance of combining targeted feature selection and robust balancing methods to address class imbalance effectively and improve overall classification performance.

Examining feature selection models, **LDA-based feature selection** in combination with Random Forest or KNN often led to high TPRs for Class 0 but struggled with lower TPRs and precision for Class 1. This imbalance led to lower MCC and Kappa values, highlighting potential limitations in capturing complex patterns needed for minority class classification. **Boruta** feature selection methods generally improved precision, especially for Class 0, due to its comprehensive approach to selecting relevant features. However, Boruta-based models occasionally retained redundant information, which could increase FPR (False Positive Rate) and reduce TPR for Class 1, especially in models where minority class sensitivity is challenging to maintain. These findings suggest that **Information Gain** feature selection, especially in Random Forest, better balances precision and recall, enhancing both TPR and reducing FPR for minority classes.

Analysing metric values across tables reveals that the **Random Forest model with Information Gain on Random Under-Sampling (RUS)** not only had the **highest TPRs for Class 0 (0.800) and Class 1 (0.916)** but also demonstrated strong performance across other metrics. This model achieved a weighted precision of 0.914 and an F-score of 0.842, indicating that it effectively balances precision and recall, capturing true positives while minimizing false positives. The ROC (0.932) and Matthews Correlation Coefficient (MCC) (0.514) further underscore its balanced and robust predictive power, making it a top choice for classification tasks where both class sensitivity and accuracy are crucial. In comparison, **models with LDA-based feature selection**, such as KNN and rpart, often showed strong TPR for Class 0 but lower TPR and precision for Class 1, resulting in lower MCC and Kappa values. This indicates limitations in handling the complexity of minority class patterns, likely due to LDA's linear assumptions. On the other hand, models using **Boruta feature selection** generally provided high precision for Class 0, as Boruta effectively captures a wide range of relevant features. However, Boruta models occasionally suffered from redundant information, which led to slightly higher False Positive Rates (FPR) and lower TPR for Class 1 in some cases, particularly with models sensitive to feature selection noise.

8. Conclusion and Division of Work

8. 1. Conclusion

In conclusion, the **Random Forest model with Information Gain on Random Under-Sampling (RUS)** demonstrated the highest performance across key metrics, including TPR for both classes, precision, F-score, ROC, and MCC. This model effectively balanced sensitivity across classes, capturing both minority and majority class patterns with minimal bias, making it particularly well-suited for classification tasks involving imbalanced data. The combination of Information Gain for targeted feature selection and RUS for class balancing proved essential in optimizing the model's predictive power by emphasizing relevant features and minimizing the impact of class imbalance. Although other approaches, such as LDA and Boruta-based feature selection with clustering, had their strengths—such as higher precision or preserved majority class patterns—they often lacked the same balanced TPRs and robustness. This analysis underscores that selecting the right feature selection and balancing techniques is crucial in enhancing model accuracy and stability. Overall, the Random Forest with Information Gain and RUS emerges as an optimal strategy, providing a reliable solution for handling complex and imbalanced classification tasks.

8. 2. Division of Work

Sanjana Prasad -

- Research about the dataset from the cookbook
- Initial Meta Data Analysis of Patients
- Data Exploration on understanding invariant and unnecessary columns
- Null Values imputation
- Random Under Sampling of the training data with LDA, Information Gain, and Boruta Feature Selection methodology
- Training (Cross Validation with Parameter tuning) and Testing KNN, Adaboost and Decision Tree models.

Yash Rao -

- Research about the dataset from the cookbook
- Data conversion (factor conversion and numeric conversion of columns)
- Outlier Detection and Handling
- Standardizing numeric columns
- Clustered Sampling of the training data with LDA, Information Gain, and Boruta Feature Selection methodology
- Training (Cross Validation with Parameter tuning) and Testing SVM (Radial Basis Function), Random Forest and XGBoost model.

9. References

- https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
- https://stat.ethz.ch/R-manual/R-patched/library/MASS/html/lda.html
- https://cran.r-project.org/web/packages/Boruta/Boruta.pdf
- https://cran.r-project.org/web/packages/caret/caret.pdf
- https://cran.r-project.org/web/packages/adabag/adabag.pdf
- https://cran.r-project.org/web/packages/e1071/e1071.pdf
- https://xgboost.readthedocs.io/
- Rahman, R. M., & Davis, D. N. (2013). Cluster based under-sampling for unbalanced cardiovascular data. *Computers in Biology and Medicine*, 43(10), 1407-1416. https://doi.org/10.1016/j.compbiomed.2013.07.008
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81-106. https://doi.org/10.1007/BF00116251
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794. https://doi.org/10.1145/2939672.2939785