Final Report

Training Best Model of Predicting Depressive Disorder

Sun Yidan

Department of Computer Science, Boston University

CS566 Data Mining

Professor: Dr. Jae Young Lee

March 30, 2024

1. Introduction

This report is based on the raw 5000 personal data records attributes from date of birth, income, health test, race, marriage status, children, education, smoke, and each person has labeled as whether he or she has depressive order or not.

Since the raw data is incomplete and has lots of missing values, so the first step is to do preprocessing and later apply models training and evaluate result.

Our goal is to find the best model from 36 models and 6 algorithms are used in the training, in 36 models performance are evaluated from diverse metrics based on the predicted labels compared to actual labels.

Processes are divided for 9 parts: preprocessing, algorithms chosen, data partition, data balance, attribute selection, 36 models parameter tuning and training, generate performance metrics result, evaluate data visualization, finalize conclusion.

This data mining process is trained and coded in R.

2. Data Mining Tools

Followings are R package tools usage illustration.

Data Manipulation and Exploration

- dplyr: Part of the tidyverse, dplyr is a powerful package for data manipulation in R. It provides a set of functions for summarizing, rearranging, and modifying data frames with an emphasis on readability and usability.
- tidyverse: An ecosystem of packages designed for data science, which includes dplyr, ggplot2 (for data visualization), tidyr (for tidying data), readr (for reading data), and more. It promotes a coherent data analysis workflow in R.

Feature Selection and Engineering

- FSelector: Provides functions for selecting attributes (features) in your dataset based on their relevance to the outcome variable. It includes methods like entropy-based ranking, chi-squared test, and more.
- Boruta: An all-relevant feature selection method. Boruta iteratively removes features deemed less relevant until all remaining features are proven to be relevant or tentatively relevant, offering a robust approach to feature selection.

Machine Learning Models and Evaluation

- RWeka: An interface to Weka, which is a collection of machine learning algorithms for data mining tasks. It allows R users to access Weka methods for classification, regression, clustering, and feature selection directly from R.
- e1071: Houses functions for statistical learning, including SVM (Support Vector Machines), Fourier Transform, and more. It's commonly used for classification and regression tasks.
- caret: A comprehensive package for creating machine learning models. It simplifies the process of training, tuning, and evaluating models across a wide variety of algorithms.
- randomForest: Implements the Random Forest algorithm for classification and regression. It's known for its performance and ease of use for complex datasets.
- C50: Provides an interface to C5.0, a decision tree algorithm that is an extension of C4.5. It's used for classification and generating predictive models.
- rpart and rpart.plot: rpart is used for creating recursive partitioning models, such as decision trees, for classification and regression tasks. rpart.plot helps in visualizing these models.

Sampling and Data Preparation

- rsample: A part of the tidyverse, focused on providing functions for resampling statistical models, which is crucial in estimating model performance.
- ROSE: Aims at dealing with imbalanced dataset problems by generating synthetic samples. It's useful in improving model performance on datasets where one class is significantly underrepresented.

Statistical Analysis

- MASS: Provides functions and datasets to support Venables and Ripley's book "Modern Applied Statistics with S". It includes a variety of statistical methods, including linear and nonlinear modelling, classification, clustering, and more.
- stats: Part of R's base packages, it includes a wide range of statistical methods for linear and nonlinear modeling, tests, time series analysis, classification, and clustering.
- psych: Primarily aimed at psychologists, psychometricians, and social scientists, it provides tools for psychometric analysis and data visualization. It's useful for personality, psychological research, and psychometrics.

Visualization

- ggplot2: Part of the tidyverse, ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. It allows users to create complex, multi-layered graphics using a comprehensive and coherent system.

Other

mltools: Provides a set of tools for working with machine learning datasets, especially those related to multi-label classification.
pROC: A tool for visualizing, comparing, and analyzing Receiver Operating Characteristic (ROC) curves. It's particularly useful for evaluating the performance of binary classifiers.

3. Prepocessing Methods

3.1 Cleaning tool

Python, Excel spreadsheet

3.2 Related Files

Prepocessed_data, 2 python files.

3.3 Data exploration

1) There are in total 197 columns we found out in dataset that has missing values.
2) Some data has exact the same value, and their names are pretty close, it is due to the repeated data column input.
3) Most data are categorical, and some are continuous.
4) Some data are highly skewed, or only has one value, rest are remained N/A
5) Some columns have exact no value, there is no meaning to keep it in dataset.

3.4 Methodology

For missing value columns 1~99, Approached several methods here, using median/KNN/deletion for different data types. Used median for most categorical data, even some data are numerical, but they are only integer exists in the dataset, and also data is highly skewed for most of them, so it is better to use median to replace the N/A data. For continuous data, we used KNN method to fill in the N/A values. For those are completely N/A value in column, we just deleted it. For those we similar name and exact same value in column, also we deleted it.

For the remaining columns we sorted into categories based on how empty they were and whether the data was categorical or continuous. For the continuous data Knn method was used again to fill in values. Columns were identified that were primarily ones or empty and assumed to be single value indicators, so they were filled with zeros or ones. Some columns were primarily ones or twos so those were also filled in with ones, twos or removed entirely. An example of that would Male or Female was encoded as 1 or 2 so empty values were removed. The remaining categorical values were categorized based on how

many nulls there were. For columns that were more than 95% empty they were deleted. For columns that were more than 95% full, the values were filled in with the most common value in the spreadsheet because we assumed that they were missing values in error. Finally for categorical values with missing values between 5% and 95% full they mostly appeared to be just rare cases where the row values could be zeros if they were declined to be answered.

Further data cleaning will be required when the modeling begins but this was a great first step to understanding the dataset and doing some bulk cleaning early. Once significant columns are identified for model performance then we can perform more thorough cleaning on those areas in detail.

4. Classification Algorithms

Total of 6 algorithms have been used.

- Decision Tree-J48
- Decision Tree-C5.0
- Logistic Regression (optimized and regular)
- Naïve Bayes
- SVM
- Random Forest

5. Detailed Procedure

The Detailed Procedure split to two parts, data preprocessing, and data mining (data split, data balanced, data 36 models training, parameter tuning, performance measures, result comparison, etc.)

5.1 Data Preprocessing

Except for the intermediate report, we also cleaned data based on its collinearity and get rid of columns that has high similarity.

Also, we change N as 0 and Y as 1 in “Class”

The cleaned dataset is: “cleaned_trimmed_data.csv”

5.2 Data Mining

We used cleaned_trimmed_data.csv as our preprocessed dataset. The Class distribution: number of 3926 in Class “N”, number of 925 in Class “Y” , where we classify “N” as 0, “Y” as 1 in dataset.

5.2.1 Partition data

- The original processed dataset has Y of 3926 and N of 925.
- Used set.seed(123) for reproductivity, then used 8:2 ratio to partition data into two datasets: train_data and test_data.
- In train_data, we found out there is 3143 of “0” and 738 of “1”.

5.2.2 Balance Data

Used two methods to balance data since class is imbalanced in train_data, that is, the number of “0” and number of “1” is not equally distributed. This can lead to biased models that perform poorly in predicting the minority class.

To solve the problem, the usage of oversampling and undersampling is necessary.

- Oversampling

  To oversampling, the method generates more minority class to achieve the number of majority class to get balanced data. Implemented Oversample-SMOTE Technique, and library (ROSE).

  Before

  Count of “0” : 3143

  Count of “1” : 738

  After

  Count of “0” : 3143

  Count of “1” : 3124

  Now, Balanced Oversampling data is completed.

- Undersampling

  To undersampling, the method truncates more majority class to achieve the number of minority class to get balanced data.
  The location of code is after attributes selection of oversampling data.

Split data into data_N and data_Y
Set.seed(123) and create the balanced data sample stored as data_undersample, through table( )function

Before
Count of "0" : 3143
Count of "1" : 738

After
The count of "0":738
The count of "1":738

Now, Balanced Undersampling data is completed.

5.2.3 Attributes Selection

Three attributes' selections method: Chi-Square, Anova F-test, Random Forest Variable Importance.

The order of coding is first implementing on attribute selections on oversampling dataset, then later all selections on undersampling dataset.

Training Datasets Name
Oversample : Anov_F_1, Chi_Sq_1, RF_1
Undersample: Anov_F_2, Chi_Sq_2, RF_2

- Chi-Square

Apply on oversampling

Classify Class to factor, find approach to apply chi-square test and extract p-value, then filter dataset based on threshold p-value <0.05.

Filter out the selected attribute dataset and get oversampling chi-squared data: Chi_Sq_1 and based on attribute selection, get test data Chi_Sq_1_test

Apply on undersampling

Same method to generate p-value and filter the selected attribute dataset to get Chi_Sq_2, Chi_Sq_2_test

- Anova F-test

Apply on oversampling

Conduct ANOVA, extract the summary and then the p-value, threshold of p-value is 0.05.
Select features based on a p-value threshold $p < 0.05$ and generated under_selected_features_2 as selected features.

Generated oversampling selected attributes training data as Anov_F_1, and test data for Anov_F_1_test

Apply on undersampling

Same method, select features based on a p-value threshold $p < 0.05$, and generated under_selected_features_2 as selected features.
Generated under sampling selected attributes training data as Anov_F_2, and test data for Anov_F_2_test

- Random Forest Variable Importance

Apply on oversampling

Perform random forest variable importance, and extract the column corresponding to MeanDecreaseGini, Sort the importance measures and pick out the top 100 features. Choose the dataset with the 100 features and get selected data RF_1; trained dataset as RF_1_test

Apply on undersampling

Same method picked out top 100 features and generated the selected train data RF_2 and test data as RF_2_test.

5.2.4 Combinations

Usage of 36 combinations. Followed by 2*3*6 rule, where 2 datasets (undersample and oversample), under each dataset has 3 different attributes selection datasets, under each feature selection dataset, 6 models have been implemented and trained.

For each balanced and attribute selected data: Anov_F_1, Chi_Sq_1, RF_1, Anov_F_2, Chi_Sq_2, RF_2, 6 models have been implemented on each.

By using the parameter tuning, we get the best value for each model, and created the Performance measures and confusion matrix functions to get results.

	Balance	Algorithm	Models Name	Tuning Best Value
1			OS_Best_DecisionTree_Anov_F_1	cp= 0.004321383

2	Over Sampling	Decision Tree (rpart)	OS_Best_DecisionTree_Chi_Sq_1	cp= 0.004321383
3			OS_Best_DecisionTree_RF_1	cp= 0.004321383
4		Decision Tree (J48)	OS_Best_J48_Anov_F_1	C= 0.5, M=1
5			OS_Best_J48_Chi_Sq_1	C= 0.5, M=1
6			OS_Best_J48_RF_1	C= 0.5, M=1
7		Logistics Regression	OS_Best_LogR_Anov_F_1	alpha = 0.5 and lambda = 0.001519911
8			OS_Best_LogR_Chi_Sq_1	alpha = 0.5 and lambda = 0.002656088
9			OS_Best_LogR_RF_1	alpha = 0.5 and lambda = 0.002656088
10		Naïve Bayes	OS_Best_NB_Anov_F_1	usekernel = FALSE and adjust = 1.
11			OS_Best_NB_Chi_Sq_1	usekernel = FALSE and adjust = 1.
12			OS_Best_NB_RF_1	usekernel = FALSE and adjust = 1.
13		Support Vector Machine	OS_Best_SVM_Anov_F_1	0.4 and C = 1
14			OS_Best_SVM_Chi_Sq_1	0.4 and C = 1
15			OS_Best_SVM_RF_1	0.4 and C = 1
16		Random Forest	OS_Best_RF_Anov_F_1	mtry = 5
17			OS_Best_RF_Chi_Sq_1	mtry = 3
18			OS_Best_RF_RF_1	mtry = 5
19	Under Sampling	Decision Tree (rpart)	US_Best_DecisionTree_Anov_F_2	cp= 0.004742547
20			US_Best_DecisionTree_Chi_Sq_2	cp= 0.003387534
21			US_Best_DecisionTree_RF_2	cp= 0.006775068
22		Decision Tree (J48)	US_Best_J48_Anov_F_2	C=0.01, M=4
23			US_Best_J48_Chi_Sq_2	C=0.01, M=1
24			US_Best_J48_RF_2	C=0.01, M=2
25		Logistics Regression	US_Best_LogR_Anov_F_2	alpha = 0.5 and lambda = 0.01072267
26			US_Best_LogR_Chi_Sq_2	alpha = 0.5 and lambda = 0.00231013
27			US_Best_LogR_RF_2	alpha = 0.5 and lambda = 0.002656088
28		Naïve Bayes	US_Best_NB_Anov_F_2	Laplace=0,usekernel = FALSE and adjust = 1.
29			US_Best_NB_Chi_Sq_2	Laplace=0,usekernel = TRUE and adjust = 1.
30			US_Best_NB_RF_2	Laplace=0,usekernel = FALSE and adjust = 1.
31		Support Vector Machine	US_Best_SVM_Anov_F_2	0.4 and C = 1
32			US_Best_SVM_Chi_Sq_2	0.15 and C = 1.6
33			US_Best_SVM_RF_2	0.4 and C = 1
34		Random Forest	US_Best_RF_Anov_F_2	mtry = 5
35			US_Best_RF_Chi_Sq_2	mtry = 3
36			US_Best_RF_RF_2	mtry = 7

Test dataset Variables

We picked out all the attributes selections from each unit of train dataset and selected the data with same attributes from test_data, which is partitioned already at the beginning of code.

The test dataset ready for calculating performance metrics are as followings:

Oversampling test data:
Chi_Sq_1_test
Anov_F_1_test
RF_1_test

Undersampling test data:
Anov_F_2_test
Chi_Sq_2_test
RF_2_test

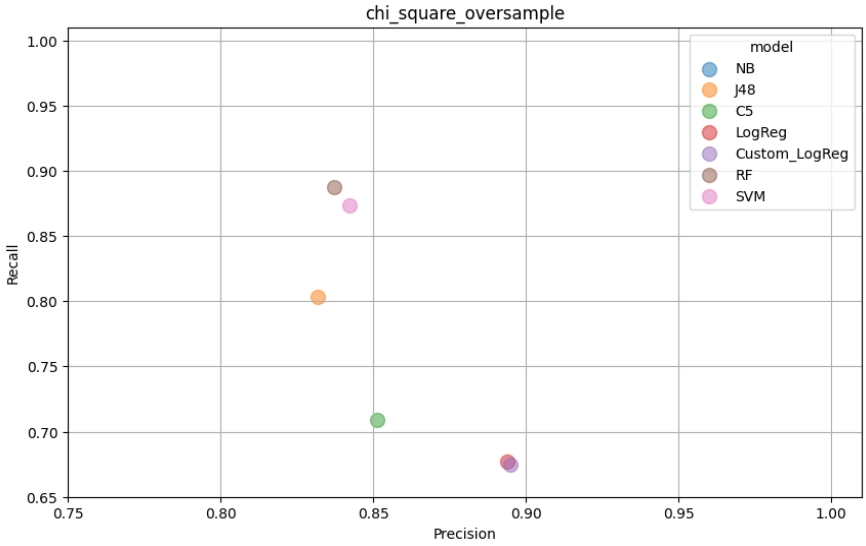
6. Result and Evaluation

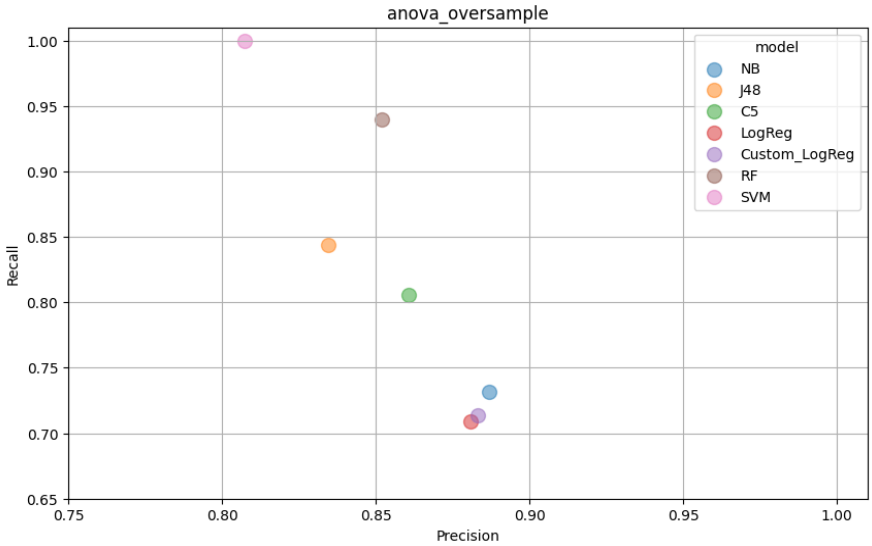
Model Results:

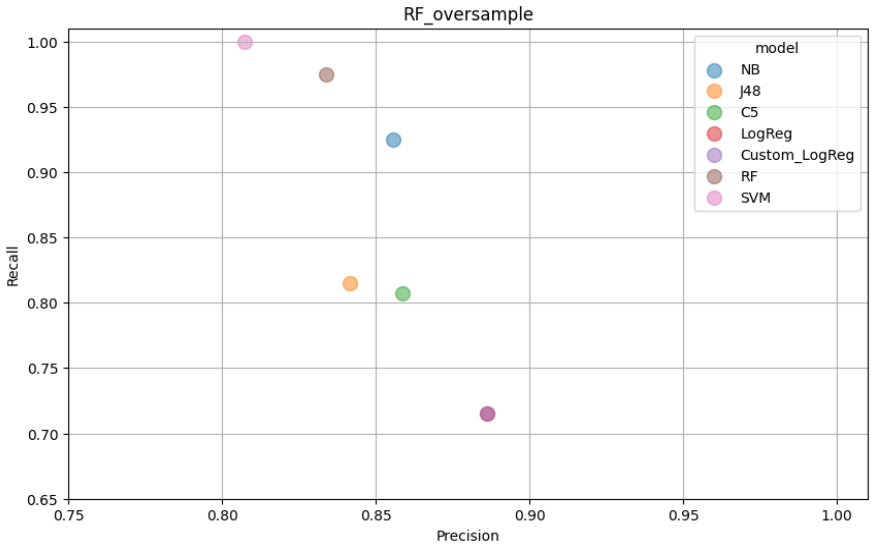
Over Sample Training Results:

model	data	Precision	Recall	F1	Accuracy	Kappa	Sensitivity	Specificity	mcc
LogReg	all_data	0.849	0.962	0.902	0.831	0.311	0.962	0.283	0.346
NB	Chi Square	0.909	0.446	0.598	0.516	0.143	0.446	0.813	0.209
J48	Chi Square	0.832	0.803	0.817	0.710	0.118	0.803	0.321	0.118
C5	Chi Square	0.851	0.709	0.774	0.665	0.150	0.709	0.481	0.160
LogReg	Chi Square	0.894	0.677	0.770	0.674	0.245	0.677	0.663	0.275
Custom_LogReg	Chi Square	0.895	0.674	0.769	0.673	0.246	0.674	0.668	0.277
RF	Chi Square	0.837	0.888	0.862	0.770	0.183	0.888	0.278	0.186
SVM	Chi Square	0.842	0.874	0.858	0.766	0.201	0.874	0.316	0.202

NB	ANOVA	0.887	0.732	0.802	0.708	0.267	0.732	0.610	0.286
J48	ANOVA	0.835	0.844	0.839	0.739	0.146	0.844	0.299	0.146
C5	ANOVA	0.861	0.806	0.832	0.738	0.236	0.806	0.455	0.239
LogReg	ANOVA	0.881	0.709	0.786	0.688	0.235	0.709	0.599	0.254
Custom_ LogReg	ANOVA	0.883	0.714	0.790	0.693	0.244	0.714	0.604	0.264
RF	ANOVA	0.852	0.940	0.894	0.820	0.306	0.940	0.316	0.323
SVM	ANOVA	0.807	1.000	0.893	0.807	0.000	1.000	0.000	0.000
NB	RF	0.856	0.925	0.889	0.813	0.312	0.925	0.348	0.322
J48	RF	0.842	0.815	0.828	0.727	0.165	0.815	0.358	0.165
C5	RF	0.859	0.807	0.832	0.737	0.229	0.807	0.444	0.231
LogReg	RF	0.886	0.715	0.792	0.696	0.253	0.715	0.615	0.273
Custom_ LogReg	RF	0.886	0.715	0.792	0.696	0.253	0.715	0.615	0.273
RF	RF	0.834	**0.974**	0.899	0.823	0.221	0.974	0.187	0.276
SVM	RF	0.807	**1.000**	0.893	0.807	0.000	1.000	0.000	0.000

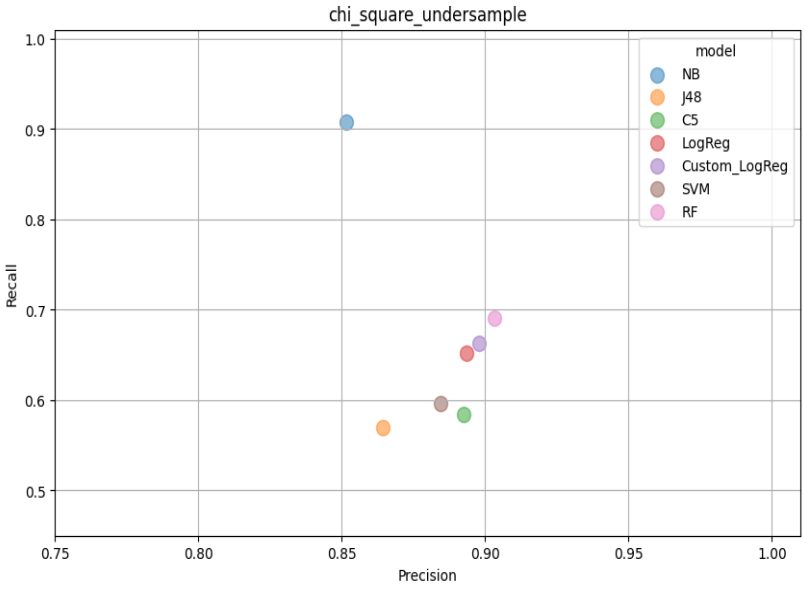

chi_square_oversample
model
NB
J48
C5
LogReg
Custom_LogReg
RF
SVM
Recall
Precision
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.75
0.80
0.85
0.90
0.95
1.00

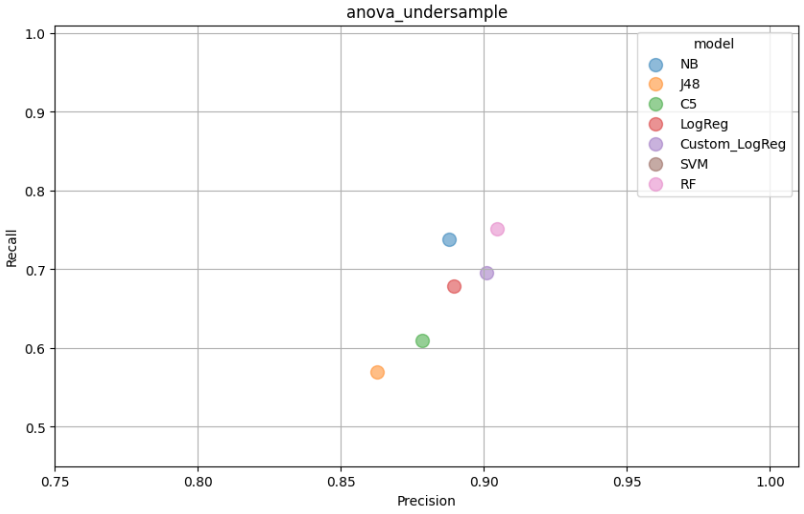

anova_oversample
model
NB
J48
C5
LogReg
Custom_LogReg
RF
SVM
Recall
Precision
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.75
0.80
0.85
0.90
0.95
1.00

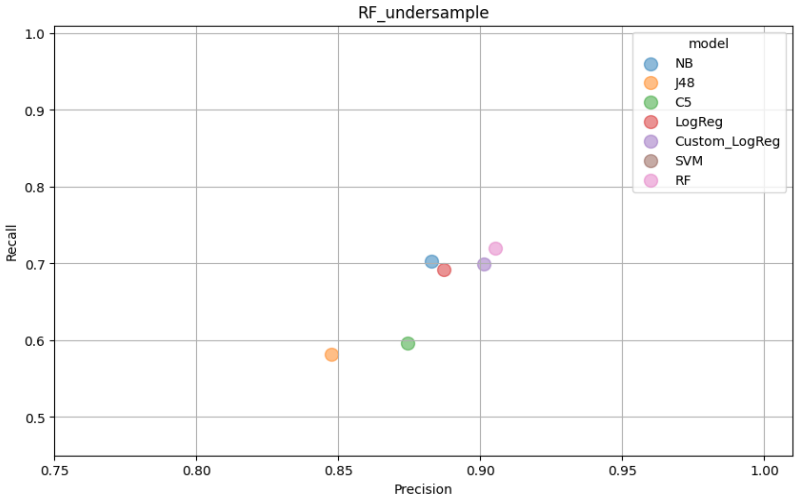

RF_oversample
model
NB
J48
C5
LogReg
Custom_LogReg
RF
SVM
Recall
Precision
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.75
0.80
0.85
0.90
0.95
1.00

Under Sample Training Results:

model	data	Precision	Recall	F1	Accuracy	Kappa	Sensitivity	Specificity	mcc
NB	Chi Square	0.851	**0.908**	**0.879**	**0.798**	0.274	0.908	0.337	0.279
J48	Chi Square	0.864	0.570	0.687	0.580	0.127	0.570	0.626	0.154
C5	Chi Square	0.893	0.584	0.706	0.607	0.187	0.584	0.706	0.229
LogReg	Chi Square	0.893	0.653	0.754	0.657	0.228	0.653	0.674	0.262
Custom_LogReg	Chi Square	0.898	0.663	0.763	0.667	0.245	0.663	0.684	0.279
SVM	Chi Square	0.884	0.596	0.712	0.611	0.178	0.596	0.674	0.214
RF	Chi Square	0.903	0.691	0.783	0.691	0.277	0.691	0.690	0.309
NB	ANOVA	0.888	0.738	0.806	0.713	0.274	0.738	0.610	0.292
J48	ANOVA	0.863	0.570	0.686	0.579	0.123	0.570	0.620	0.150
C5	ANOVA	0.878	0.609	0.719	0.616	0.172	0.609	0.647	0.204
LogReg	ANOVA	0.889	0.678	0.770	0.672	0.236	0.678	0.647	0.264
Custom_LogReg	ANOVA	0.901	0.696	0.785	0.693	0.275	0.696	0.679	0.306
SVM	ANOVA	**0.914**	0.326	0.480	0.431	0.097	0.326	0.872	0.172
RF	ANOVA	0.905	0.751	0.821	0.735	0.330	0.751	0.668	**0.316**
NB	RF Select	0.883	0.702	0.782	0.685	0.235	0.702	0.610	0.257
J48	RF Select	0.848	0.582	0.690	0.578	0.096	0.582	0.561	0.114
C5	RF Select	0.875	0.596	0.709	0.605	0.158	0.596	0.642	0.189
LogReg	RF Select	0.887	0.692	0.778	0.680	0.239	0.692	0.631	0.264


chi_square_undersample
model
NB
J48
C5
LogReg
Custom_LogReg
SVM
RF
Recall
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.75
0.80
0.85
0.90
0.95
1.00

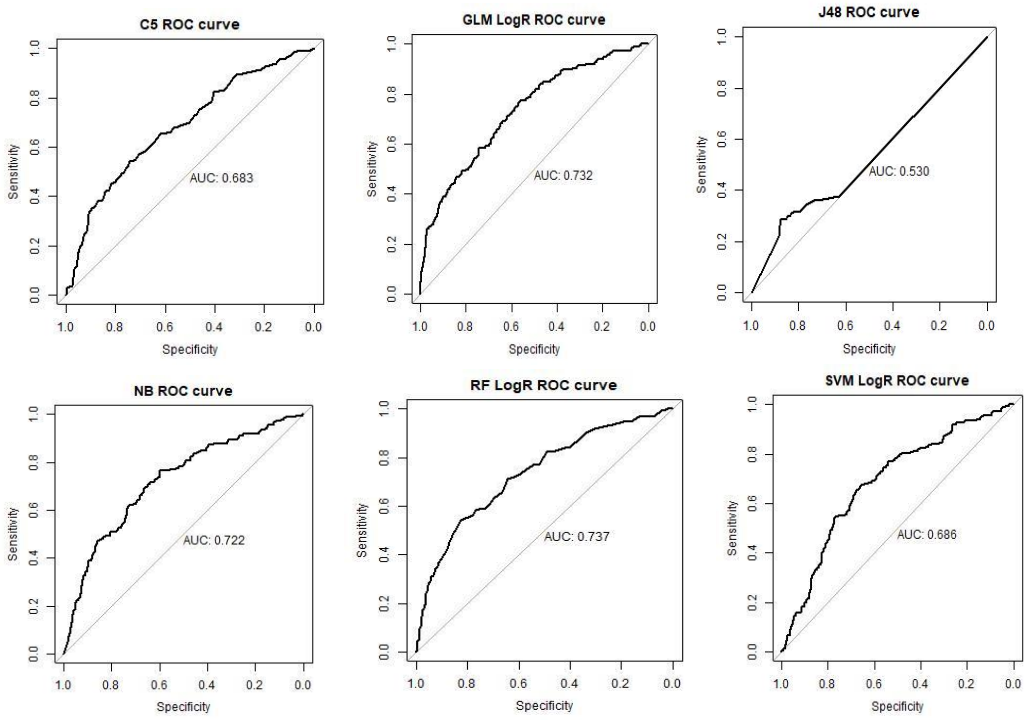

anova_undersample
model
NB
J48
C5
LogReg
Custom_LogReg
SVM
RF
Recall
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.75
0.80
0.85
0.90
0.95
1.00


RF_undersample
model
NB
J48
C5
LogReg
Custom_LogReg
SVM
RF
Recall
Precision
1.0
0.9
0.8
0.7
0.6
0.5
0.75
0.80
0.85
0.90
0.95
1.00

ROC plots:

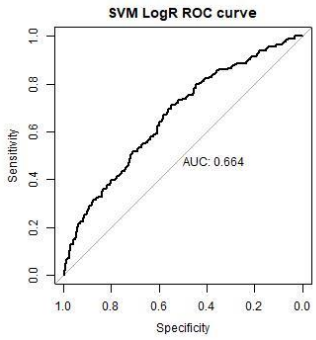
Over Sample Set

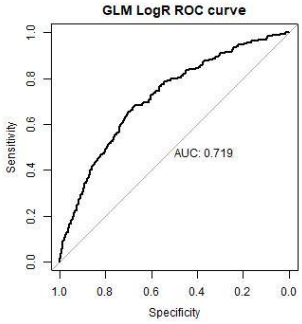
ANOVA

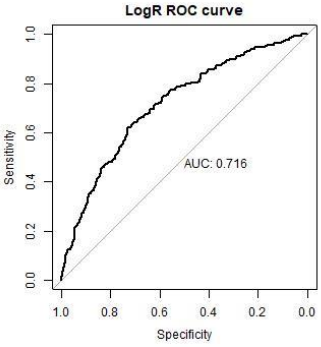


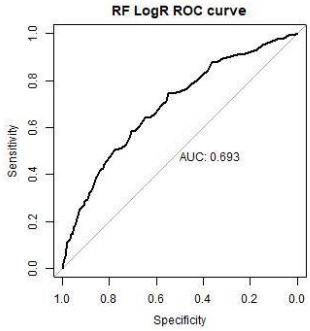
Over Sample Set

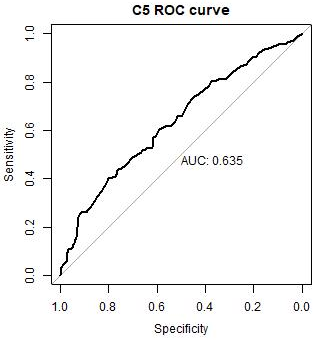
Chi-Square

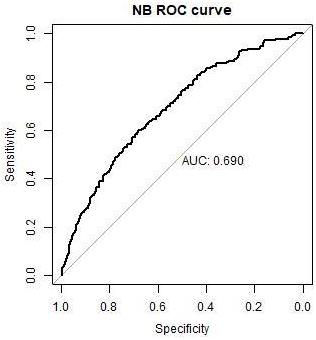


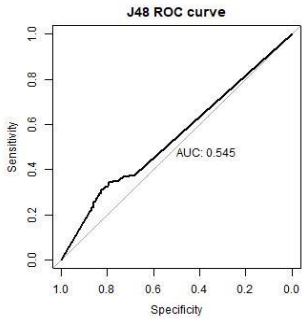






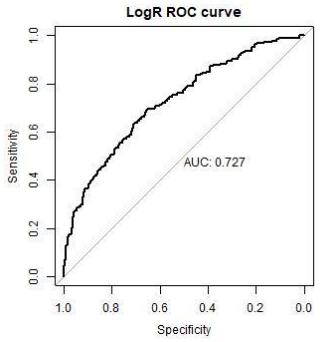


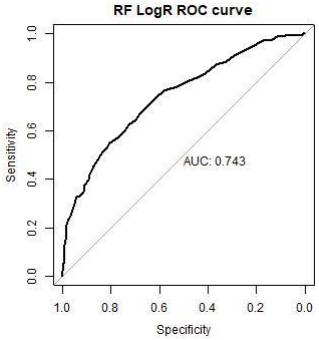


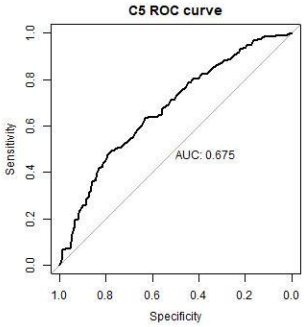


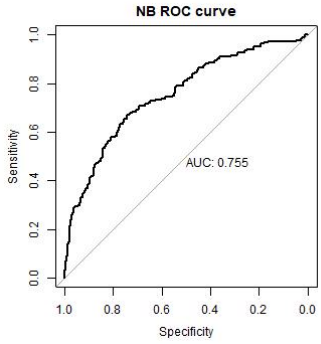
Over Sample Set

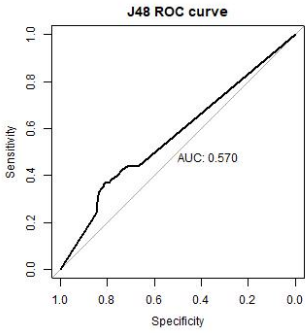
Random Forest Selection

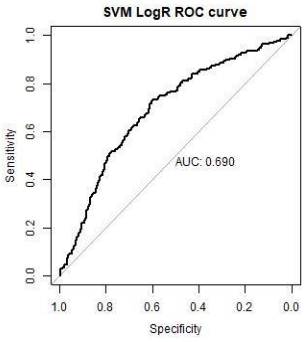


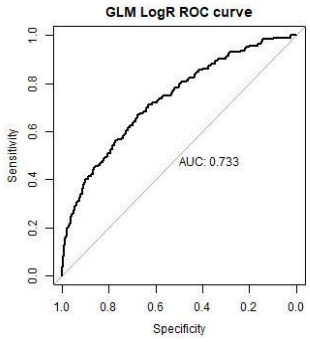






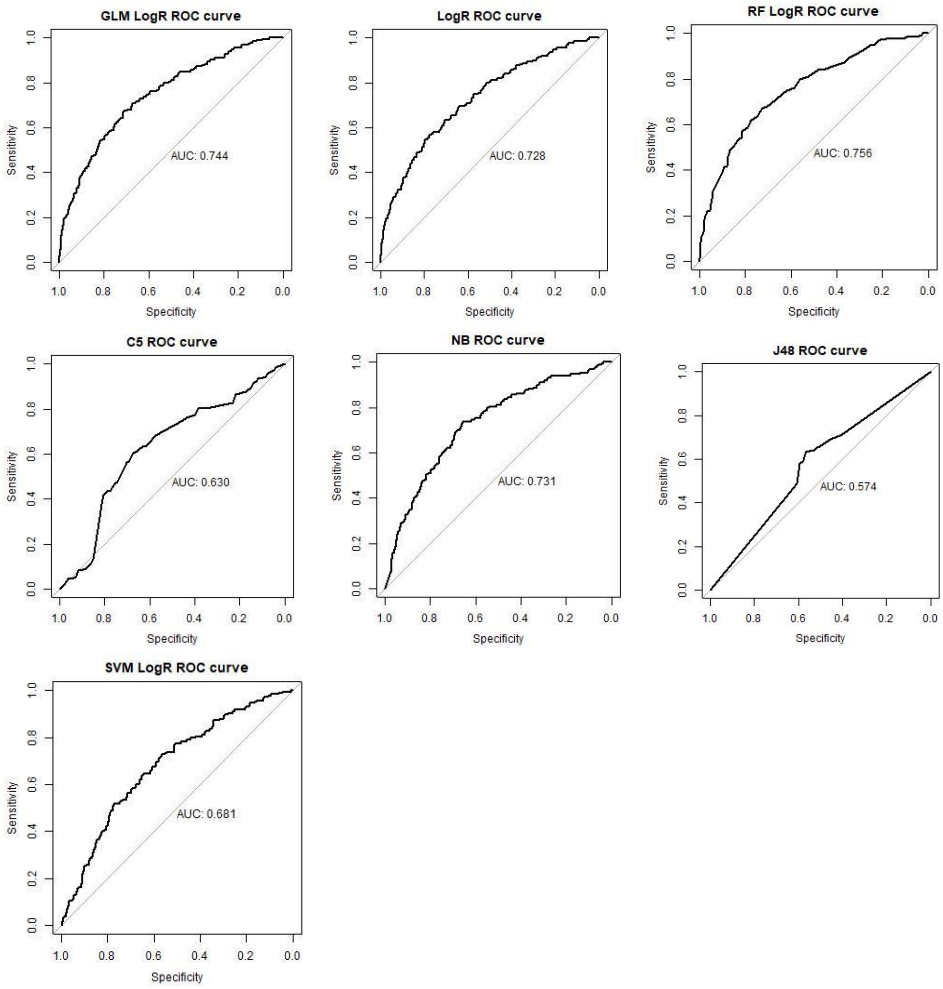






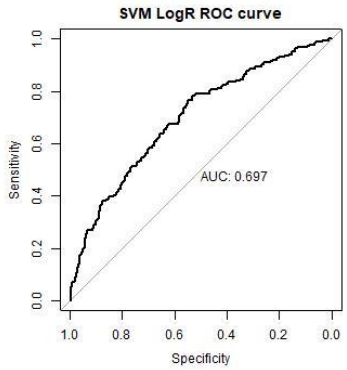
Under Sample Set

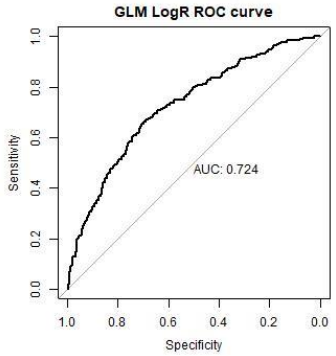
ANOVA

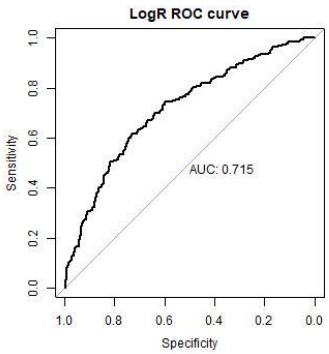


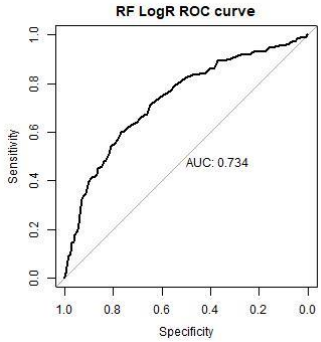
Under Sample Set

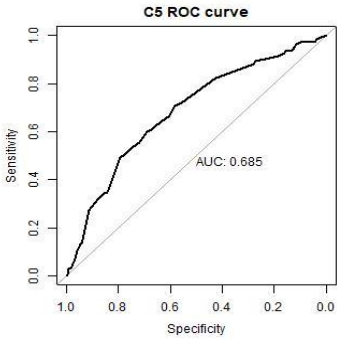
Chi–Square

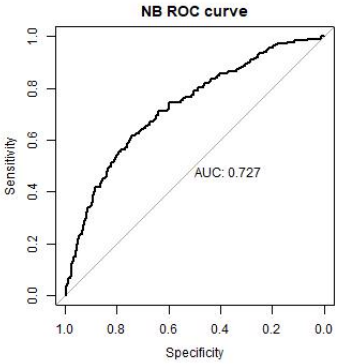


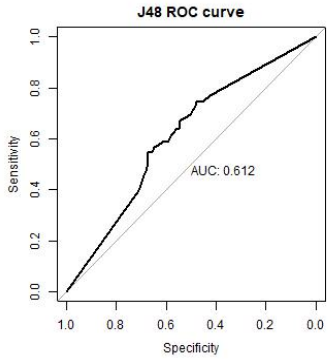






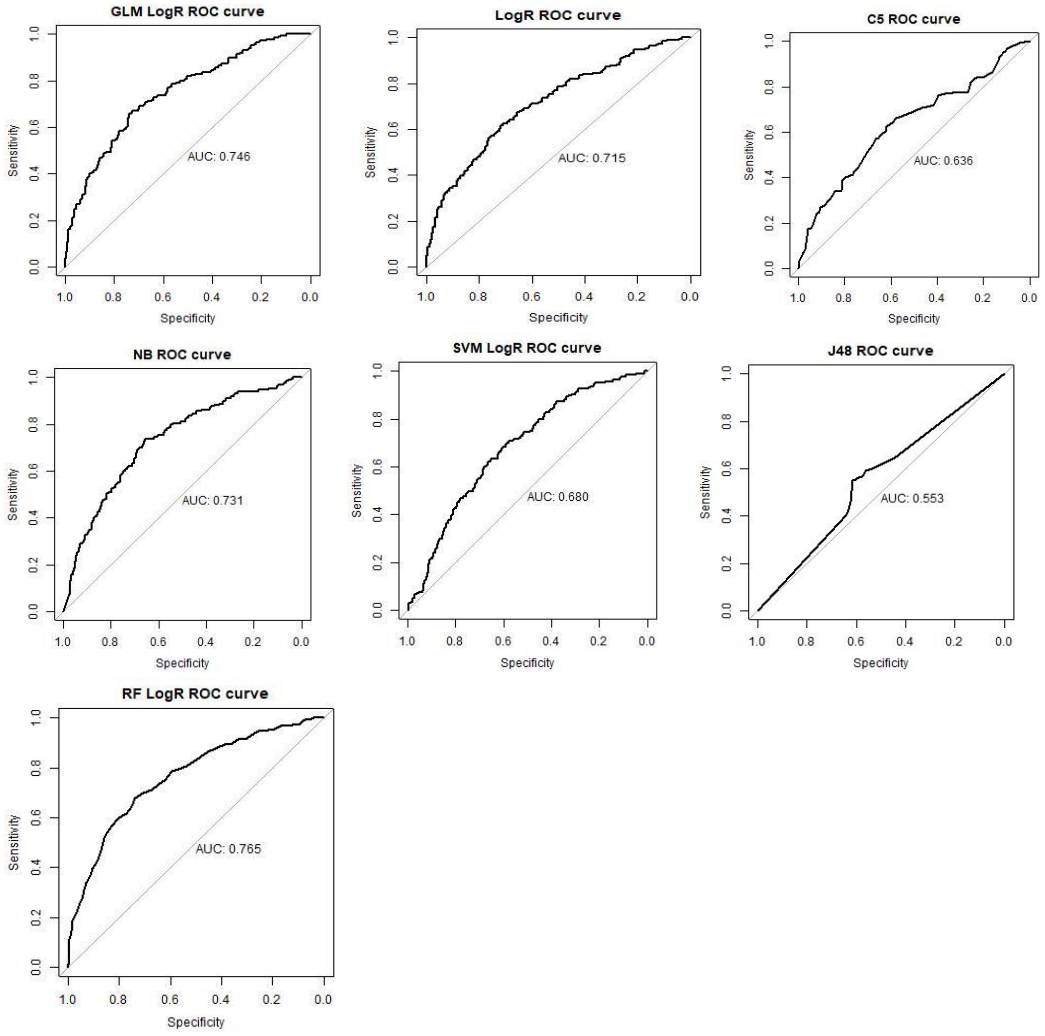






Under Sample Set

Random Forest



7. Conclusion

In this project, we investigated the performance of various machine learning models for classifying our dataset. Through rigorous experimentation and analysis, several key findings emerged.

The initial exploration of the dataset revealed a substantial class imbalance, with significantly more negative cases than positive instances. To mitigate this issue, extensive preprocessing was conducted, including resampling techniques such as oversampling of the minority class and undersampling of the majority class. Additionally, feature selection methods were employed to enhance the relevance of input features and mitigate the impact of noise. We elected to use three different feature selection techniques: Chi-square attribute selection method, ANOVA F-test and Random Forest Variable Importance. These different methods were combined together to form 6 different training sets with unique sampling and feature selection to evaluate with each model.

During the model evaluation, we observed that the dataset exhibits complex patterns that require sophisticated modeling techniques to accurately classify. Traditional algorithms such as Logistic Regression, Naive Bayes and Decision Trees showed limited performance, suggesting their incapability to capture the intricate relationships present in the data.

However, more advanced models, including Support Vector Machines (SVM) and Random Forest, showcased superior performance in terms of both accuracy and robustness. These models demonstrated the ability to handle non-linear relationships and high-dimensional feature spaces, thereby offering better predictive power. They especially were high performing models at recall and were able to identify even the most difficult edge example cases positively.

Moreover, we explored the impact of hyperparameter tuning on model performance. Fine-tuning hyperparameters significantly enhanced the classification accuracy of SVM and Random Forest models, underlining the importance of parameter optimization in machine learning tasks.

Furthermore, we investigated the interpretability of the models, recognizing the trade-off between model complexity and interpretability. While simpler models like Logistic Regression provide intuitive insights into feature importance, complex models such as Gradient Boosting offer superior predictive performance at the cost of interpretability.

Lastly, we conducted a comparative analysis of model performance metrics, including accuracy, precision, recall, F1-score, kappa and Matthews Correlation Coefficient (MCC).

Through this analysis, we identified the SVM model as the most suitable choice for our dataset, as it achieved the highest overall performance across multiple evaluation metrics. We elected to weight recall (TPR) as the most important metric because it was specifically mentioned in the project criteria. SVM really shined on this metric with a perfect scoring while maintaining high precision numbers.

We also found that oversampling was a consistently more successful method to overcome class imbalance and achieve better scores regardless of the model or feature selection used.

In conclusion, our study highlights the importance of selecting appropriate machine learning models and optimizing their parameters for effective classification tasks. Moving forward, further research could focus on ensemble techniques and deep learning architectures to explore their potential in enhancing classification performance for similar datasets.