BFRSS-DEPRESSION CLASSIFIER REPORT

Table of Contents

Introduction

This data mining project embarks on an exploratory and predictive journey through a meticulously curated dataset, where each tuple represents an individual, encapsulating a variety of attributes pertinent to their experiences, behaviours, and characteristics. At the heart of this exploration is the 'Class' attribute, a binary marker distinguishing individuals based on their encounter with depressive disorders: 'Y' signifies those who have experienced a depressive disorder, while 'N' denotes those who have not. The primary objective of this endeavour is to harness the potential of this dataset to develop, evaluate, and compare multiple classification models. Through rigorous analysis and model assessment, the project aims to identify the most effective model capable of predicting the likelihood of a person experiencing a depressive disorder. This initiative not only seeks to push the boundaries of predictive analytics in mental health but also aims to contribute valuable insights into the identification and understanding of depressive disorders, thereby offering a foundation for informed decision-making and targeted interventions.

For the Data Mining project, the data given is provided from 2020 Behavioural Risk Factor Surveillance System (BRFSS) Survey Data.

The dataset contains 5000 tuples and 276 attributes. To accomplish the goal of the project is to build multiple classification models, which would predict a person with a depressive disorder, compare their performance, and select the "best" model, the main task will be to perform data cleaning and preprocessing.

Data Mining Tools

- **Caret (Classification And Regression Training):** This package provides a unified interface for training and tuning machine learning models. It supports numerous predictive modelling techniques and automates many tasks like model selection, performance evaluation, and parameter tuning.

- **Random Forest:** Implements the random forest algorithm for classification and regression. It's a powerful tool for predictive modelling, capable of handling large datasets and providing estimates of variable importance.

- **e1071:** This package contains functions for statistical learning, including support vector machines (SVM), short-time Fourier transform, fuzzy clustering, and more. It's widely used for classification and regression tasks.

- **xgboost :** An efficient implementation of the gradient boosting framework, **xgboost** is a powerful library for building predictive models, especially in competitions and practical applications due to its speed and performance.

- **pROC:** The pROC package in R is a versatile tool for visualizing and analyzing the performance of classification algorithms through Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) metrics. It provides an easy-to-use interface for creating ROC curves, calculating AUC values, and comparing the performance of different models.

- **rsample:** The rsample package in R is designed to assist in the process of resampling and cross-validation for model evaluation and selection. It provides a suite of tools to create and manage different types of resampling procedures, such as bootstrapping, k-fold cross-validation, and leave-one-out cross-validation.

- **Boruta:** The Boruta package in R is designed for feature selection, aiming to identify all relevant features in a dataset for building robust and accurate predictive models. It is based on a random forest classification method but extends it by creating a shadow feature set from the original features by random permutation.

- **kknn:** The kknn package in R is used for k-nearest neighbors (k-NN) classification and regression. K-nearest neighbors is a simple, instance-based learning algorithm where the response of an observation is determined by the majority vote (in classification) or average (in regression) of the k closest training examples.

Classification Algorithms

Classification algorithms are used in machine learning to categorize data into predefined classes or categories. They work by learning patterns from labelled training data and then applying this learning to classify new, unseen data points into the appropriate classes.

Logistic Regression: It's a linear model used for binary classification, estimating probabilities using a logistic function and making predictions based on a threshold.

Random Forest: A collection of decision trees that aggregate predictions, reducing overfitting and improving accuracy through ensemble learning.

Support Vector Machines (SVM): This algorithm finds the best hyperplane that separates classes in a high-dimensional space, maximizing the margin between data points of different classes.

Naive Bayes: Based on Bayes' theorem, it assumes independence between features and calculates the probability of each class given the input data, often used for text classification tasks.

K-Nearest Neighbours (KNN): It classifies data points based on the majority class among their k nearest neighbours, making it simple but sensitive to noisy data and the choice of k.

XGBoost (XGBoost): XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm widely used for structured data classification and regression tasks, renowned for its performance and speed in handling large datasets. It employs an ensemble of decision trees, optimized using gradient boosting techniques, to deliver highly accurate models, making it a popular choice in data science competitions and industry applications.

Data Mining Procedure

Data Pre-Processing:-

Data preprocessing is a crucial step in data analysis and machine learning, serving as the foundation for generating reliable, accurate results. It involves cleaning, transforming, and organizing raw data into a suitable format for analysis, which is essential for several reasons. Firstly, it helps in handling missing values, outliers, and noise in the data, which can significantly distort the outcomes of the analysis if not addressed properly. Secondly, it allows for feature extraction and selection, helping to identify the most relevant information and reduce the dimensionality of the data, which in turn improves computational efficiency and model accuracy.

Using the dataset given, we followed the below given steps:

- <u>Data Cleaning</u> : Data cleaning involves the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset, enhancing its quality. This step may include handling missing values, correcting errors, and removing duplicates, ensuring the data is consistent and suitable for analysis.
- <u>Data reduction</u> : It aims to decrease the volume of data, making it easier to analyse without significantly losing important information. Techniques such as dimensionality reduction, aggregation, and feature selection help to simplify the data, focusing on relevant information while reducing storage and processing requirements.
    1. Correlation Analysis
    2. Zero covariance

Data Cleaning:

The data cleaning is the process of finding out missing values and to remove duplicates. Using R programming language, this is the summary of data :

Number of tuples : 5000
Number of attributes : 276
Number of missing values in whole dataset : 639886

Attributes	Count	Status of the attributes
Attributes with no values	**7**	Drop
Attributes with no missing values	**79**	Analysis not Required
Attributes with missing values	**190**	Analysis Required

Attributes with missing values	Count	Status of Attributes
Attributes with >= 40% of missing values	**64**	Drop
Attributes unrelated to "Class"	**4**	Drop
Attributes with <= 40% of missing values	**122**	Analysis Required

Let us understand in detail for each condition of attributes :

Attributes with no values :
Dropping columns with no values from a dataset is often justified because such columns do not contribute any information that can be used for analysis or modelling. These columns represent features that, across all observations, have no data, meaning they offer no variance or insight that could improve the understanding of the dataset's underlying patterns or the performance of predictive models. The total number of attributes which have no values in the dataset is 7 and they are:

Attributes - "COLGHOUS" "COLGSEX" "X" "TOLDCFS" "HAVECFS" "WORKCFS" "MEDSHEPB"

In total 7 attributes are dropped from the given dataset and stored in a new data frame "df". The data frame now consists of 269 attributes.

Attributes with no missing values :
Columns with no missing values are highly valuable in data analysis and modelling for several reasons. Firstly, they ensure the integrity and completeness of the dataset, allowing for more straightforward and robust statistical analyses and modelling. When every observation in a column is available, it eliminates the need for imputation techniques or the handling of missing data, which can sometimes introduce bias or inaccuracies. Secondly, columns with no missing values can contribute to more reliable and consistent results since they provide a complete set of data points for analysis.
As per the given dataset, there are 79 attributes which contain no missing values and does not require any further analysis which can be used for modelling.

Attributes with missing values :
Dealing with attributes that have missing values is a critical aspect of the data pre-processing phase in data analysis and machine learning projects. Missing data can arise due to various reasons, such as errors in data collection,

non-responses in surveys, or system faults. Several strategies can be employed to manage missing data, including deleting records or features with missing values, imputing missing values using statistical methods (mean, median, mode imputation for numerical data, or the most frequent category for categorical data), and model-based methods (like using k-nearest neighbours or regression models for imputation).

Analysing the data , it is visible that there are 190 attributes which contain missing values.

- Attributes with >= 40 % missing values :
  Considering attributes which have more than or equal to 40% of missing values for each attribute, a total of 68 are dropped from the dataset.
  Reason : Dropping attributes with 40% or more missing values is a pragmatic approach to data cleaning, often adopted in the pre-processing phase of data analysis and machine learning projects. This threshold-based rule is applied because attributes with such a high level of missingness can introduce significant uncertainty and bias into the analysis or predictive modelling.

  Moreover, retaining these attributes might not add meaningful information due to the sparse nature of the data and could unnecessarily increase the complexity of the analysis or model. By removing these attributes early in the data pre-processing stage, analysts can focus on more reliable predictors, ensuring the robustness and validity of their findings.

- Attributes with <=40% missing values:

  A total number of 122 attributes contain 40% or less missing values.

  For attributes with 40% or less missing values, conducting a thorough analysis to understand the nature of the missing data and deciding on an appropriate method for handling these missing values is crucial. This threshold suggests that a significant portion of the data is present, potentially carrying valuable information that could influence the analysis or model outcomes. Retaining and properly imputing these attributes can enhance the dataset's integrity, allowing for a more comprehensive analysis. Various imputation methods can be applied, depending on the attribute's type and the missingness pattern. Common techniques include mean or median imputation for numerical attributes, mode imputation for categorical attributes, or more sophisticated approaches like using regression models, k-nearest neighbours (KNN), or multiple imputation methods that consider correlations between attributes.

The next step is to deal with the attributes which have missing values and to decide which method to use to handle the missing values.

Demonstrating three attributes to understand which method is used and why the specific method is used for the particular attribute. The remaining number of attributes(including the below three) and the methods used are shown in the R file.

Attribute 1 : SMOKE100
Description (from Codebook) : Smoked at Least 100 Cigarettes
Method : Mode
R snippet :

```
> #SMOKE100
> sum(is.na(df$SMOKE100))
[1] 221
> df$SMOKE100[is.na(df$SMOKE100)]=mfv(df$SMOKE100[!is.na(df$SMOKE100)])
> sum(is.na(df$SMOKE100))
[1] 0
```

Reason : The values are labelled as 1, 2, 7 and 9. There are around 221 missing values , using mode the missing values are replaced by the most frequent value in the whole attribute for all 5000 tuples.
Analysis :
Before the imputation – Missing values is 221
After the imputation – Missing values is 0

Attribute 2 : CHILDREN
Description (from Codebook) : Number of Children in Household
Method : Median
R snippet :

```
> #CHILDREN
> #Data Distribution: If the variable 'CHILDREN' is skewed or has a non-normal
> #distribution, median imputation might be preferred over mean imputation to avoid introducing bias.
> #
> sum(is.na(df$CHILDREN))
[1] 53
> #CHILDREN
> #Data Distribution: If the variable 'CHILDREN' is skewed or has a non-normal
> #distribution, median imputation might be preferred over mean imputation to avoid introducing bias.
> #
> sum(is.na(df$CHILDREN))
[1] 53
> summary(df$CHILDREN)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
   1.00    3.00   88.00   64.45   88.00   99.00      53 
> # Boxplot to inspect for outliers
> boxplot(df$CHILDREN, main = "Boxplot of Number of Children")
> #
> df$CHILDREN[is.na(df$CHILDREN)]=median(!is.na(df$CHILDREN))
> sum(is.na(df$CHILDREN))
[1] 0
```

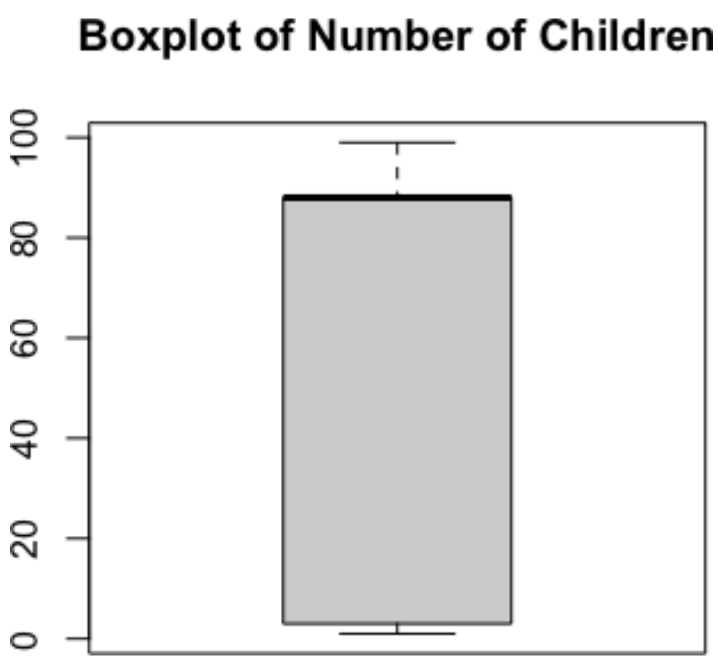
Reason : Median imputation might be preferred over mean imputation to avoid introducing bias. A box plot analysis is also performed to check for outliers for this particular attribute.

Analysis :

Before the imputation – Missing values is 53

After the imputation – Missing values is 0

Boxplot for CHILDREN:



Attribute 3 : HEIGHT3
Description (from Codebook) : Reported Height in Feet and Inches
Method : Mean
R snippet :

Reason : The HEIGHT3 mentioned in the data contains values of different magnitudes (inches and feet) . In order to convert all the HEIGHT3 values to inches and use it for further analysis is a crucial step.

Data Reduction:

Data reduction is a crucial process in data pre-processing that involves reducing the volume of data or the dimensionality of the feature space to make the data set more manageable for analysis. This process is required for several reasons. Firstly, it helps in significantly reducing storage requirements and computational costs, making data processing and analysis more efficient, especially for large datasets. Secondly, data reduction enhances the performance of machine learning models by eliminating redundant, irrelevant, or highly correlated features, thereby improving the model's accuracy and reducing the risk of overfitting.

Let's perform below steps for the given dataset:

- Zero covariance: Zero covariance between two variables indicates that there is no linear relationship between them. Covariance measures how two variables change together; it can be positive, negative, or zero.

```
> #HEIGHT3
> sum(is.na(df$HEIGHT3))
[1] 135
> max(df$HEIGHT3[df$HEIGHT3<=9900& df$HEIGHT3>7777],na.rm = TRUE)
[1] 9191
> df$HEIGHT3[1192]
[1] 510
>
>
> for (i in 1:nrow(df)) {
+   # Check the value at the current index, not the entire vector
+   if (!is.na(df$HEIGHT3[i])) {
+     if (df$HEIGHT3[i] >= 200 && df$HEIGHT3[i] <= 711) {
+       # Extract feet and inches and convert to inches
+       feet <- as.integer(substr(df$HEIGHT3[i], 1, 1))
+       inches <- as.integer(substr(df$HEIGHT3[i], 2, 3))
+       df$HEIGHT3[i] <- (feet * 12) + inches
+     } else if (startsWith(as.character(df$HEIGHT3[i]), "9")) {
+       # Convert from centimeters to inches
+       cm <- as.numeric(substr(as.character(df$HEIGHT3[i]), 2, nchar(as.character(df$HEIGHT3[i]))))
+       df$HEIGHT3[i] <- cm * 0.393701
+     }
+   }
+ }
> mean_HEIGHT3=mean(df$HEIGHT3[df$HEIGHT3!=9999 & df$HEIGHT3!=7777],na.rm = TRUE)
> mean_HEIGHT3
[1] 73.02431
> df$HEIGHT3[is.na(df$HEIGHT3)]=floor(mean_HEIGHT3)
```

- Correlation Analysis: Perform correlation analysis to identify highly correlated features. Highly correlated features with the target variable are good candidates for inclusion, while features highly correlated with each other but not with the target may be redundant.

	row	col
X_STATE	1	1
X_STSTR	59	1
FMONTH	2	2
IDATE	3	3
IMONTH	4	3
IDATE	3	4
IMONTH	4	4
IDAY	5	5
DISPCODE	6	6
SEQNO	7	7
X_PSU	8	7
SEQNO	7	8
X_PSU	8	8
CELLSEX	9	9
LANDLINE	10	10
HHADULT	11	11
SEXVAR	12	12
X_SEX	86	12
GENHLTH	13	13
PHYSHLTH	14	14
HLTHPLN1	15	15
PERSDOC2	16	16
MEDCOST	17	17

Zero covariance:

Zero covariance between two variables indicates that there is no linear relationship between them. Covariance measures how two variables change together; it can be positive, negative, or zero.

- Positive covariance means that as one variable increases, the other variable also tends to increase, indicating a positive linear relationship.
- Negative covariance means that as one variable increases, the other tends to decrease, indicating a negative linear relationship.
- Zero covariance means that there is no discernible linear trend between the variables. If the covariance is zero, changes in one variable do not predict changes in the other variable in a linear sense.

Correlation Analysis:

Performed Correlation Analysis for all the 122 attributes where "Class" attribute is the target variable and all other attributes are dependent variables. After performing Correlation Analysis, these were the insights which we got:

Considering a threshold for significance to be 0.8, the first few correlation pairs is shown in the image right

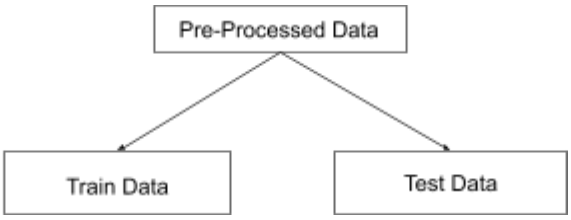
Next step is to remove the self-correlation and NA values from the data frame. Creating a data frame for significant pairs and finding out the highly correlated pairs. The image on the right shows the first few of high correlated pairs above 0.8 threshold.

Filtering out constant columns before performing a correlation analysis is essential because a column with a constant value does not vary and, therefore, has no linear relationship with other variables.

Split the Pre-processed dataset:-

Splitting a pre-processed dataset into training and testing sets is a fundamental practice in machine learning to evaluate the performance of a model.
The primary reason for this split is to ensure that the model can generalize well to unseen data, not just fit well to the data it was trained on.

Flowchart:




```
> high_cor_df
   Variable1 Variable2 Correlation
1    X_STSTR   X_STATE   0.9999956
2     IMONTH     IDATE   0.9996994
3      IDATE    IMONTH   0.9996994
4      X_PSU     SEQNO   1.0000000
5      SEQNO     X_PSU   1.0000000
6      X_SEX    SEXVAR   0.9991947
7  X_TOTINDA  EXERANY2   0.9926109
8  X_DRDXAR2  HAVARTH4   0.8340959
9   X_EDUCAG     EDUCA   0.9818362
10 X_DRNKDRV   ALCDAY5   0.8483267
```

Code Snippet:

```
##Splitting the dataset into test and train 66% training and 34% testing
set.seed(7)
split = initial_split(df,prop = 0.66,strata = "Class")
train = training(split)
test = testing(split)
write.csv(test,"initial_test.csv",row.names = FALSE)
write.csv(train,"initial_train.csv",row.names = FALSE)
```

We have split the preprocessed dataset into train and test dataset (66% training and 34% testing).Two csv files are generated after executing the commands in R by splitting the pre-processed dataset into train and test datasets and written as initial_test.csv and initial_train.csv which will be used for further analysis.

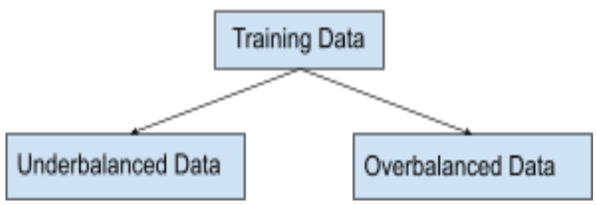
Data Balancing:

Data balancing refers to the process of adjusting the distribution of classes in a dataset to prevent biases in machine learning models. In an imbalanced dataset, one or more classes significantly outnumber the others, which can lead to a model that performs well on the majority class but poorly on the minority class.

Data balancing techniques, such as oversampling the minority class and under sampling the majority class are used to create a more balanced class distribution. This helps ensure that the model learns to recognize patterns associated with all classes, not just the dominant one. By balancing the dataset, we promote a more equitable representation of all classes, enabling the model to learn a more general and robust representation of the data.

We have performed oversampling and undersampling for the pre-processed dataset obtained after performing pre-processing techniques as mentioned above. In R, we have used “srswor” (Simple Random Sampling Without Replacement) for under sampling and “srswr” (Simple Random Sampling Without Replacement) for over sampling.

Flowchart:



Code Snippet:

```
#undersampling using simple random sampling without replacement
set.seed(7)
srswor=sample(index_0s,number_1,replace = FALSE)


undersample_index=c(srswor,index_1s)
undersample_index

undersample=train[undersample_index,]
prop.table(table(undersample$Class))

#oversampling using simple random sampling with replacement
set.seed(7)
srswr=sample(index_1s,number_0,replace = TRUE)


oversample_index=c(srswr,index_0s)
oversample_index

oversample=train[oversample_index,]
prop.table(table(oversample$Class))

#saving the underbalanced and overbalanced data sets

#saving the underbalanced dataset
write.csv(undersample,"underbalanced.csv",row.names=FALSE)

#saving the overbalanced dataset
write.csv(oversample,"overbalanced.csv",row.names=FALSE)
```

Two csv files are generated after executing the commands in R by implementing oversampling and undersampling code to the train dataset and saved as overbalanced.csv and underbalanced.csv which will be used for further analysis.

Attribute Selection Methods:-

Attribute selection, also known as feature selection, is a process used in machine learning to identify and select a subset of relevant features (attributes) for use in model construction. The goal of attribute selection is to improve model performance by eliminating redundant, irrelevant, or noisy data, thus simplifying the model, reducing training times, and enhancing generalization by reducing overfitting.

The three methods we used for our project are listed below:

Boruta: Boruta is a feature selection algorithm that identifies all relevant features in a dataset for predictive modelling. It is based on a random forest approach and operates by creating shadow features (random copies of original

features) and then iteratively comparing the importance of actual features with these shadows.
Below are the codes for underbalanced and overbalanced using Boruta method:

```
#boruta
set.seed(1001)
underbalanced.boruta=Boruta(Class~.,data=underbalanced)
underbalanced.boruta

features.boruta.underbalanced= getSelectedAttributes(underbalanced.boruta, withTentative = FALSE)
underbalanced_boruta= underbalanced[,c(features.boruta.underbalanced,"Class")]
write.csv(underbalanced_boruta,"underbalanced_boruta.csv",row.names = FALSE)

test_underbalanced_boruta= test[,c(features.boruta.underbalanced,"Class")]
write.csv(test_underbalanced_boruta,"test_underbalanced_boruta.csv",row.names = FALSE)
```

```
#boruta
set.seed(1001)
overbalanced.boruta=Boruta(Class~.,data=overbalanced)
overbalanced.boruta

features.boruta.overbalanced= getSelectedAttributes(overbalanced.boruta, withTentative = FALSE)
overbalanced_boruta= overbalanced[,c(features.boruta.overbalanced,"Class")]
write.csv(overbalanced_boruta,"overbalanced_boruta.csv",row.names = FALSE)

test_overbalanced_boruta= test[,c(features.boruta.overbalanced,"Class")]
write.csv(test_overbalanced_boruta,"test_overbalanced_boruta.csv",row.names = FALSE)
```

Information Gain (InfoGain): Information Gain is a feature selection method used primarily in the context of decision tree algorithms. It measures the reduction in entropy or surprise from transforming a dataset in some way. In feature selection, it evaluates the worth of an attribute by calculating the difference between the entropy of the dataset before and after a split on that attribute.
Below are the codes for underbalanced and overbalanced using InfoGain method:

```
#information gain

underbalanced.infogain=information.gain(Class~.,data=underbalanced)
underbalanced.infogain=cbind(rownames(underbalanced.infogain),data.frame(underbalanced.infogain,row.names = NULL))
names(underbalanced.infogain)=c("Attribute","Info Gain")
sorted.underbalanced.infogain=underbalanced.infogain[order(-underbalanced.infogain$`Info Gain`),]
sorted.underbalanced.infogain$Attribute[1:5]

features.infogain.underbalanced=c(sorted.underbalanced.infogain$Attribute[1:5])
underbalanced_infogain=underbalanced[,c(features.infogain.underbalanced,"Class")]
write.csv(underbalanced_infogain,"underbalanced_infogain.csv",row.names = FALSE)

test_underbalanced_infogain=test[,c(features.infogain.underbalanced,"Class")]
write.csv(test_underbalanced_infogain,"test_underbalanced_infogain.csv",row.names = FALSE)
```

```
#information gain

overbalanced.infogain=information.gain(Class~.,data=overbalanced)
overbalanced.infogain=cbind(rownames(overbalanced.infogain),data.frame(overbalanced.infogain,row.names = NULL))
names(overbalanced.infogain)=c("Attribute","Info Gain")
sorted.overbalanced.infogain=overbalanced.infogain[order(-overbalanced.infogain$`Info Gain`),]
sorted.overbalanced.infogain[1:5,]

features.infogain.overbalanced=c(sorted.overbalanced.infogain$Attribute[1:5])
overbalanced_infogain=overbalanced[,c(features.infogain.overbalanced,"Class")]
write.csv(overbalanced_infogain,"overbalanced_infogain.csv",row.names = FALSE)

test_overbalanced_infogain=test[,c(features.infogain.overbalanced,"Class")]
write.csv(test_overbalanced_infogain,"test_overbalanced_infogain.csv",row.names = FALSE)
```

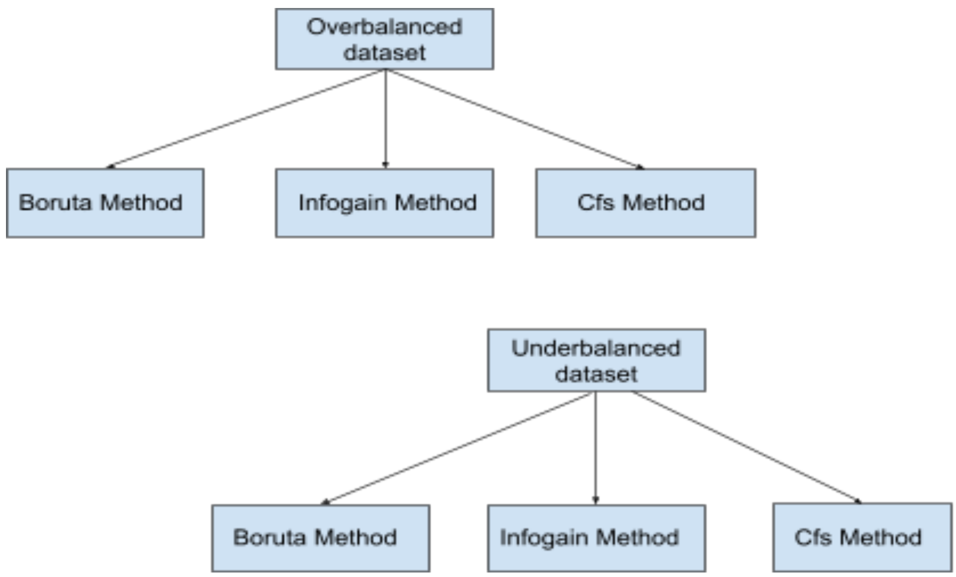
Correlation-based Feature Selection (CFS): Correlation-based Feature Selection (CFS) is a heuristic method for feature selection that evaluates the usefulness of individual features and subsets of features based on their correlation with the target variable and the absence of correlation with each other.
Below are the codes for underbalanced and overbalanced using CFS method:

```
#CFS
underbalanced.cfs=cfs(Class~.,data=underbalanced)
underbalanced.cfs
underbalanced_cfs=underbalanced[,c(underbalanced.cfs,"Class")]
write.csv(underbalanced_cfs,"underbalanced_cfs.csv",row.names = FALSE)
```

```
#CFS
overbalanced.cfs=cfs(Class~.,data=overbalanced)
overbalanced.cfs
overbalanced_cfs=overbalanced[,c(overbalanced.cfs,"Class")]
write.csv(overbalanced_boruta,"overbalanced_cfs.csv",row.names = FALSE)
```

We have created 6 datasets based on the three attribute selection methods using overbalanced and underbalanced datasets created after data balancing.

Flowchart



Classification Algorithms:

Classification algorithms used are mentioned below:

Logistic Regression: The logistic regression model is used in all 6 dataset that was created and saved after doing the preprocessing, initial splitting , balancing and feature selection. Since logistic regression model does not have any parameters that can be tuned we didn't do parameter tuning

Random Forest: The Random forest model which is an ensemble model is used on all the 6 dataset as mentioned above. The random forest model was tuned for its parameter using tune grid and train control.

Support Vector Machines (SVM): The SVM model was also run for all 6 datasets that were saved as mentioned above.This model was run using only base parameters.

Naive Bayes: Since this model is based on Bayes' theorem, it assumes independence between features and calculates the probability of each class given the input data, often used for classification tasks.This model was tuned using its parameters such as Train Control and Tune Grid.

K-Nearest Neighbours (KNN): Since it classifies data points based on the majority class among their k nearest neighbours.This model was also used to run the 6 datasets that were saved as mentioned above. This model was tuned for the K value and also the distance metric using Tune Grid and Train Control

XGBoost (XGBoost): XGBoost (Extreme Gradient Boosting) is a powerful machine learning algorithm widely used for structured data classification and regression tasks, renowned for its performance and speed in handling large datasets. It employs an ensemble of decision trees, optimized using gradient boosting techniques, to deliver highly accurate models, making it a popular choice in data science competitions and industry applications.

Data Mining Result and Evaluation

1).LOGISTIC REGRESSION :

Confusion matrix:

Table 1: Underbalanced Boruta

Prediction \ Reference	N	Y
N	931	118
Y	446	206

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.6358025	0.3238925	0.3159509	0.6358025	0.4221311	0.6997	0.2519168	0.2248691
Class N	0.6761075	0.3641975	0.8875119	0.6761075	0.7675185	0.6997	0.2519168	0.2248691
Wt.Average	0.655955	0.344045	0.6017314	0.655955	0.5948248	0.6997062	0.2519168	0.2248691

Table 2: Underbalanced_Cfs

Prediction \ Reference	N	Y
N	935	125
Y	442	119

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.6141975	0.3209877	0.3104524	0.6141975	0.4124352	0.692	0.2375944	0.2133844
Class N	0.6790123	0.3858025	0.8820755	0.6790123	0.7673369	0.692	0.2375944	0.2133844
Wt.Average	0.6466049	0.3533951	0.5962639	0.6466049	0.5898861	0.6919643	0.2375944	0.2133844

Table 3: Underbalanced Infogain

Prediction \ Reference	N	Y
N	968	143
Y	409	181

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.558642	0.2970225	0.3067797	0.558642	0.3960613	0.672	0.2158373	0.1991157
Class N	0.7029775	0.441358	0.8712871	0.7029775	0.778135	0.672	0.2158373	0.1991157
Wt.Average	0.6308097	0.3691903	0.5890334	0.6308097	0.5870982	0.6720337	0.2158373	0.1991157

Table 4: Overbalanced Boruta

Prediction \ Reference	N	Y
N	984	115
Y	383	209

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.6450617	0.2854031	0.3471761	0.6450617	0.4514039	0.7234	0.2953472	0.2708134
Class N	0.7145969	0.3549383	0.8953594	0.7145969	0.7948304	0.7234	0.2953472	0.2708134
Wt.Average	0.6798293	0.3201707	0.6212677	0.6798293	0.6231171	0.7233564	0.2953472	0.2708134

Table 5: Overbalanced Infogain

Prediction \ Reference	N	Y
N	960	133
Y	417	191

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5895062	0.3028322	0.3141447	0.5895062	0.4098712	0.6965	0.2348907	0.2147131
Class N	0.6971678	0.4104938	0.8783166	0.6971678	0.7773279	0.6965	0.2348907	0.2147131
Wt.Average	0.643337	0.356663	0.5962306	0.643337	0.5935996	0.6965054	0.2348907	0.2147131

Table 6: Overbalanced Cfs

Prediction \ Reference	N	Y
N	974	136
Y	403	188

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5802469	0.2926652	0.3181049	0.5802469	0.410929	0.675	0.2371622	0.2186788
Class N	0.7073348	0.4197531	0.8774775	0.7073348	0.783273	0.675	0.2371622	0.2186788
Wt.Average	0.6437908	0.3562092	0.5977912	0.6437908	0.597101	0.6750282	0.2371622	0.2186788

2).NAIVE BAYES:

Table 1: Overbalanced Boruta

Prediction \ Reference	N	Y
N	1137	186
Y	240	138

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4259259	0.1742919	0.3650794	0.4259259	0.3931624	0.6737	0.2376747	0.2365591
Class N	0.8257081	0.5740741	0.8594104	0.8257081	0.8422222	0.6736	0.2376747	0.2365591
Wt.Average	0.625817	0.374183	0.6122449	0.625817	0.6176923	0.6736447	0.2376747	0.2365591

Table 2: Overbalanced Cfs

Prediction \ Reference	N	Y
N	1168	183
Y	209	141

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4351852	0.1517792	0.4028571	0.4351852	0.4183976	0.6913	0.2752877	0.2749692
Class N	0.8482208	0.5648148	0.8645448	0.8482208	0.856305	0.6913	0.2752877	0.2749692
Wt.Average	0.641703	0.358297	0.633701	0.641703	0.6373513	0.6912639	0.2752877	0.2749692

Table 3: Overbalanced Infogain

Prediction \ Reference	N	Y
N	1194	202
Y	183	122

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.3765432	0.1328976	0.4	0.3765432	0.3879173	0.689	0.2494052	0.2492339
Class N	0.8671024	0.6234568	0.8553009	0.8671024	0.8611612	0.689	0.2494052	0.2492339
Wt.Average	0.6218228	0.3781772	0.6276504	0.6218228	0.6245393	0.6890023	0.2494052	0.2492339

Table 4: Underbalanced Boruta

Prediction \ Reference	N	Y
N	1137	184
Y	240	140

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4320988	0.1742919	0.3684211	0.4320988	0.3977273	0.6995	0.2430472	0.2418256
Class N	0.8257081	0.5679012	0.8607116	0.8257081	0.8428466	0.6994	0.2430472	0.2418256
Wt.Average	0.6289034	0.3710966	0.6145663	0.6289034	0.6202869	0.6994383	0.2430472	0.2418256

Table 5: Underbalanced Cfs

Prediction \ Reference	N	Y
N	1126	162
Y	251	162

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5	0.1822803	0.3922518	0.5	0.4396201	0.7068	0.2909716	0.2875216
Class N	0.8177197	0.5	0.8742236	0.8177197	0.8450281	0.7068	0.2909716	0.2875216
Wt.Average	0.6588598	0.3411402	0.6332377	0.6588598	0.6423241	0.7068232	0.2909716	0.2875216

Table 6: Underbalanced Infogain

Prediction \ Reference	N	Y
N	1175	186
Y	202	138

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4259259	0.1466957	0.4058824	0.4259259	0.4156627	0.692	0.2741785	0.2740556
Class N	0.8533043	0.5740741	0.8633358	0.8533043	0.8582907	0.692	0.2741785	0.2740556
Wt.Average	0.6396151	0.3603849	0.6346091	0.6396151	0.6369767	0.6920002	0.2741785	0.2740556

3).RANDOM FOREST:

Table 1: Overbalanced Boruta

Prediction \ Reference	N	Y
N	1323	238
Y	54	86

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.2932099	0.04647785	0.5974843	0.2932099	0.3933747	0.7166	0.3328314	0.3063916
Class N	0.9535221	0.7067901	0.8514916	0.9535221	0.8996232	0.7166	0.3328314	0.3063916
Wt.Average	0.623366	0.376634	0.7244879	0.623366	0.6464989	0.7166109	0.3328314	0.3063916

Table 2: Overbalanced Cfs

Prediction \ Reference	N	Y
N	1163	213
Y	214	111

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.3487654	0.1633987	0.3343195	0.3487654	0.3413897	0.6649	0.1824169	0.182354
Class N	0.8366013	0.6512346	0.8451944	0.8366013	0.8408759	0.6649	0.1824169	0.182354
Wt.Average	0.5926834	0.4073166	0.589757	0.5926834	0.5911328	0.6649139	0.1824169	0.182354

Table 3: Overbalanced Infogain

Prediction \ Reference	N	Y
N	1072	154
Y	305	170

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5308642	0.221496	0.360587	0.5308642	0.4294632	0.6836	0.2704365	0.2620531
Class N	0.778504	0.4691358	0.875817	0.778504	0.8242983	0.6836	0.2704365	0.2620531
Wt.Average	0.6546841	0.3453159	0.618202	0.6546841	0.6268808	0.683633	0.2704365	0.2620531

Table 4: Underbalanced Boruta

Prediction \ Reference	N	Y
N	963	134
Y	414	190

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5925926	0.3093682	0.3106796	0.5925926	0.4076433	0.7045	0.2312393	0.2102709
Class N	0.6906318	0.4074074	0.8781163	0.6906318	0.7731707	0.7045	0.2312393	0.2102709
Wt.Average	0.6416122	0.3583878	0.594398	0.6416122	0.590407	0.704549 8	0.2312393	0.2102709

Table 5: Underbalanced Cfs

Prediction \ Reference	N	Y
N	998	137
Y	379	187

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5895062	0.2788671	0.3321739	0.5895062	0.4249166	0.7005	0.257865	0.2396526
Class N	0.7211329	0.4104938	0.8818828	0.7211329	0.7934479	0.7005	0.257865	0.2396526
Wt.Average	0.6553195	0.3446805	0.6070283	0.6553195	0.6091822	0.7004637	0.257865	0.2396526

Table 6: Underbalanced Infogain

Prediction \ Reference	N	Y
N	1042	146
Y	335	178

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5462963	0.2389252	0.3498024	0.5462963	0.426506	0.6812	0.2640239	0.2530265
Class N	0.7610748	0.4537037	0.8769874	0.7610748	0.81493	0.6812	0.2640239	0.2530265
Wt.Average	0.6536855	0.3463145	0.6133949	0.6536855	0.620718	0.6811809	0.2640239	0.2530265

4).XGBoost:
Table 1: Overbalanced Boruta

Prediction \ Reference	N	Y
N	1237	219
Y	140	105

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.3240741	0.1016703	0.4285714	0.3240741	0.3690685	0.6977	0.2487241	0.2452698
Class N	0.8983297	0.6759259	0.8495879	0.8983297	0.8732792	0.6977	0.2487241	0.2452698
Wt.Average	0.6112019	0.3887981	0.6390797	0.6112019	0.6211739	0.6977079	0.2487241	0.2452698

Table 2: Overbalanced Infogain

Prediction \ Reference	N	Y
N	1113	177
Y	264	147

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4537037	0.1917211	0.3576642	0.4537037	0.4	0.6483	0.2403234	0.2375895
Class N	0.8082789	0.5462963	0.8627907	0.8082789	0.8346457	0.6483	0.2403234	0.2375895
Wt.Average	0.6309913	0.3690087	0.6102275	0.6309913	0.6173228	0.6483219	0.2403234	0.2375895

Table 3: Overbalanced Cfs

Prediction \ Reference	N	Y
N	1157	201
Y	220	123

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.3796296	0.1597676	0.3586006	0.3796296	0.3688156	0.6236	0.2151755	0.2150404
Class N	0.8402324	0.6203704	0.8519882	0.8402324	0.8460695	0.6236	0.2151755	0.2150404
Wt.Average	0.609931	0.390069	0.6052944	0.609931	0.6074425	0.6236025	0.2151755	0.2150404

Table 4: Underbalanced Boruta

Prediction \ Reference	N	Y
N	975	134
Y	402	190

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5864198	0.291939	0.3209459	0.5864198	0.4148472	0.7162	0.2427559	0.2237241
Class N	0.708061	0.4135802	0.8791704	0.708061	0.7843926	0.7162	0.2427559	0.2237241
Wt.Average	0.6472404	0.3527596	0.6000582	0.6472404	0.5996199	0.7161615	0.2427559	0.2237241

Table 5: Underbalanced Infogain

Prediction \ Reference	N	Y
N	1015	139
Y	362	185

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5709877	0.2628903	0.3382084	0.5709877	0.4247991	0.692	0.2590185	0.2439095
Class N	0.7371097	0.4290123	0.8795494	0.7371097	0.8020545	0.692	0.2590185	0.2439095
Wt.Average	0.6540487	0.3459513	0.6088789	0.6540487	0.6134268	0.6919845	0.2590185	0.2439095

Table 6: Underbalanced Cfs

Prediction \ Reference	N	Y
N	973	132
Y	404	192

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.2439095	0.2933914	0.3221477	0.5925926	0.4173913	0.6979	0.2462626	0.2264981
Class N	0.7066086	0.4074074	0.880543	0.7066086	0.7840451	0.6979	0.2462626	0.2264981
Wt.Average	0.6496006	0.3503994	0.6013453	0.6496006	0.6007182	0.6978648	0.2462626	0.2264981

5).SVM:
Table 1: Overbalanced Boruta

Prediction \ Reference	N	Y
N	1152	161
Y	225	163

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5030864	0.1633987	0.4201031	0.5030864	0.4578652	0.7285	0.3178858	0.3158339
Class N	0.8366013	0.4969136	0.87738	0.8366013	0.8565056	0.7285	0.3178858	0.3158339
Wt.Average	0.6698439	0.3301561	0.6487416	0.6698439	0.6571854	0.7284578	0.3178858	0.3158339

Table 2: Overbalanced Infogain

Prediction \ Reference	N	Y
N	1026	166
Y	351	158

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4876543	0.254902	0.3104126	0.4876543	0.3793517	0.6608	0.199589	0.1910439
Class N	0.745098	0.5123457	0.8607383	0.745098	0.7987544	0.6608	0.199589	0.1910439
Wt.Average	0.6163762	0.3836238	0.5855754	0.6163762	0.5890531	0.6607774	0.199589	0.1910439

Table 3: Overbalanced Cfs

Prediction \ Reference	N	Y
N	953	133
Y	424	191

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5895062	0.3079158	0.3105691	0.5895062	0.4068158	0.6899	0.2301468	0.2096088
Class N	0.6920842	0.4104938	0.8775322	0.6920842	0.773853	0.6899	0.2301468	0.2096088
Wt.Average	0.6407952	0.3592048	0.5940507	0.6407952	0.5903344	0.6899426	0.2301468	0.2096088

Table 4: Underbalanced Boruta

Prediction \ Reference	N	Y
N	924	131
Y	453	193

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.595679	0.328976	0.2987616	0.595679	0.3979381	0.6879	0.2157868	0.193264
Class N	0.671024	0.404321	0.8758294	0.671024	0.7598684	0.6879	0.2157868	0.193264
Wt.Average	0.6333515	0.3666485	0.5872955	0.6333515	0.5789033	0.6878782	0.2157868	0.193264

Table 5: Underbalanced Infogain

Prediction \ Reference	N	Y
N	1026	166
Y	351	158

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4876543	0.254902	0.3104126	0.4876543	0.3793517	0.6564	0.199589	0.1910439
Class N	0.745098	0.5123457	0.8607383	0.745098	0.7987544	0.6564	0.199589	0.1910439
Wt.Average	0.6163762	0.3836238	0.5855754	0.6163762	0.5890531	0.6563752	0.199589	0.1910439

Table 6: Underbalanced Cfs

Prediction \ Reference	N	Y
N	949	136
Y	428	188

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5802469	0.3108206	0.3051948	0.5802469	0.4	0.6792	0.2201275	0.2003781
Class N	0.6891794	0.4197531	0.8746544	0.6891794	0.770918	0.6792	0.2201275	0.2003781
Wt.Average	0.6347131	0.3652869	0.5899246	0.6347131	0.585459	0.6791748	0.2201275	0.2003781

6).KNN:

Table 1: Overbalanced Boruta

Prediction \ Reference	N	Y
N	1207	242
Y	170	82

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.2530864	0.1234568	0.3253968	0.2530864	0.2847222	0.5648	0.1432878	0.1416667
Class N	0.8765432	0.7469136	0.8329883	0.8765432	0.8542109	0.5648	0.1432878	0.1416667
Wt.Average	0.5648148	0.4351852	0.5791925	0.5648148	0.5694666	0.5648148	0.1432878	0.1416667

Table 2: Overbalanced Infogain

Prediction \ Reference	N	Y
N	88	33
Y	1289	291

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.8981481	0.936093	0.1841772	0.8981481	0.3056723	0.481	0.0579656 4	0.0152867 2
Class N	0.0639070 4	0.1018519	0.7272727	0.0639070 4	0.11749	0.481	0.0579656 4	0.0152867 2
Wt.Average	0.4810276	0.5189724	0.455725	0.4810276	0.2115811	0.4810276	0.0579656 4	0.0152867 2

Table 3: Overbalanced Cfs

Prediction \ Reference	N	Y
N	857	188
Y	520	136

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4197531	0.3776325	0.2073171	0.4197531	0.277551	0.5211	0.0339800 1	0.0302635
Class N	0.6223675	0.5802469	0.8200957	0.6223675	0.7076796	0.5211	0.0339800 1	0.0302635
Wt.Average	0.5210603	0.4789397	0.5137064	0.5210603	0.4926153	0.5210603	0.0339800 1	0.0302635

Table 4: Underbalanced Boruta

Prediction \ Reference	N	Y
N	1017	149
Y	360	175

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.5401235	0.2614379	0.3271028	0.5401235	0.4074505	0.6851	0.2356827	0.2231267
Class N	0.7385621	0.4598765	0.8722127	0.7385621	0.7998427	0.6851	0.2356827	0.2231267
Wt.Average	0.6393428	0.3606572	0.5996577	0.6393428	0.6036466	0.6851448	0.2356827	0.2231267

Table 5: Underbalanced Infogain

Prediction \ Reference	N	Y
N	1154	190
Y	223	134

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4135802	0.1619463	0.3753501	0.4135802	0.3935389	0.6815	0.2426471	0.2422018
Class N	0.8380537	0.5864198	0.858631	0.8380537	0.8482176	0.6815	0.2426471	0.2422018
Wt.Average	0.625817	0.374183	0.6169905	0.625817	0.6208782	0.6814611	0.2426471	0.2422018

Table 6: Underbalanced Cfs

Prediction \ Reference	N	Y
N	1148	180
Y	229	144

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.4444444	0.1663036	0.386059	0.4444444	0.4131994	0.6981	0.2639677	0.2629366
Class N	0.8336964	0.5555556	0.8644578	0.8336964	0.8487985	0.6981	0.2639677	0.2629366
Wt.Average	0.6390704	0.3609296	0.6252584	0.6390704	0.630999	0.6981181	0.2639677	0.2629366

Parameters of the Best Model Selected

Classification Model : Logistic Regression
The best model which we are getting is Logistic Regression.This combination satisfies the minimum criteria .Logistic Regression , one of the popular Machine Learning algorithms, which is a type of supervised machine learning model. The combination of the model which we are getting is for Overbalanced and Boruta.

Combination : Overbalanced Boruta
Overbalanced Boruta is a combination of the overbalanced dataset which is using the oversampling method to balance it and using the Boruta method to obtain attribute selection for the overbalanced dataset.

1. Overbalancing:
Simple Random Sampling with replacement was used to over-balance the class variable that had a lesser number of values compared to the other. The indexed of class 'Y' had lesser proportion than the class 'N' variable hence the indexes of class 'Y' variable was oversampled with replacement.The data was then saved as "Overbalanced.csv" which had 5342 rows which had equal proportion of Y and N classes .

2. Boruta:
After balancing the data with the help of oversampling using the function "srswr" Simple Random Sampling With Replacement. We had a dataset that had 5342 rows and 64 columns. The 64 columns had 1 Class column which would be used for the classification task, Hence there were 63 columns or features.Now the 'overbalanced.csv' dataset was then used on the 'boruta' function to calculate the most important features.
Which turned out to be that all 63 feature columns were important.

```
> overbalanced.boruta
Boruta performed 13 iterations in 29.47609 secs.
 63 attributes confirmed important: ASTHMA3, BLDSTOL1, BLIND, CELLSEX, CHCCOPD2 and 58 more;
 No attributes deemed unimportant.
```

Performance Metrics Of The Best Model

Confusion matrix:

Prediction \ Reference	N	Y
N	984	115
Y	383	209

Performance Measures

	TPR	FPR	Precision	Recall	F-measure	ROC	MCC	Kappa
Class Y	0.6450617	0.2854031	0.3471761	0.6450617	0.4514039	0.7234	0.2953472	0.2708134
Class N	0.7145969	0.3549383	0.8953594	0.7145969	0.7948304	0.7234	0.2953472	0.2708134
Wt.Average	0.6798293	0.3201707	0.6212677	0.6798293	0.6231171	0.7233564	0.2953472	0.2708134

The model exhibits higher performance rates as compared to other models & combinations. The performance metrics are mentioned above in the table.

Discussion and Conclusion

The project aims to predict depressive disorders using classification algorithms on a given dataset initially containing 5000 tuples and 276 attributes. Pre-processing addressed missing values and reduced the attributes to 66, ensuring data quality. The dataset was then split into training (66%) and testing (34%) sets to facilitate model learning and evaluation. To tackle class imbalance, undersampling and oversampling were performed, resulting in balanced training data. Attribute selection using Boruta, Information Gain, and CFS methods further refined the essential features for model training.

Six classification algorithms, including Logistic Regression, Naive Bayes, SVM, KNN, XGBoost, and Random Forest, were evaluated across 36 model-training combinations, with parameter tuning tailored to each algorithm. The Overbalanced Boruta approach, combined with Logistic Regression, emerged as the most effective, demonstrating superior performance metrics. This combination, notable for its robust feature selection and balancing techniques, proved to be a comprehensive and interpretable solution for predicting depressive disorders, highlighting the effectiveness of meticulous data preparation and strategic model selection in achieving accurate and reliable classification outcomes.