VIETNAM NATIONAL UNIVERSITY – HOCHIMINH CITY

THE INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DATA MINING

IT160IU

Topic: Heart Disease Analysis

By Group D – Member List

NUMBER	STUDENT NAME	STUDENT ID	ROLE
1	Phạm Nguyễn Công Danh	ITDSIU23035	Member
2	Phạm Huỳnh Nhật Tân	ITDSIU23022	Member
3	Nguyễn Văn Ngọc Hải	ITCSIU23007	Member

Instructor: Dr. Nguyen Thi Thanh Sang

Dr. Nguyen Dinh Phu

GitHub

Table of contents

Chapter 1: Introduction

1.1 Overview

Provide an overview of the project, its objectives, and the methodologies employed. Include a brief explanation of data mining concepts, machine learning algorithms, and their significance in the context of this assignment.

GitHub

- **Objective**: To build a data mining framework incorporating a classification/prediction model and a sequence mining algorithm.
- **Dataset Used**: Heart Disease Dataset

1.2 Goal

- Analyze the impact of various health indicators (attributes) on heart disease status.
- Compare the performance of Naive Bayes and Random Forest classifiers.
- Address the issue of class imbalance in the dataset to improve the model's sensitivity (Recall) for detecting positive cases.
- Implement a reproducible data mining pipeline using Java and Weka, covering data preprocessing, model training, and evaluation.

1.3 Techniques & Tools Used

- **GitHub:** Used for version control, progress tracking, and collaborative work among team members.
- **Java:** The primary programming language used for implementing custom data processing and analysis logic.
- **Weka:** A suite of machine learning software used for implementing and evaluating various data mining algorithms like Naive Bayes and Random Forest.

CHAPTER 2: Data Pre-Processing

2.1 Raw data overview

- Purpose: Explore dataset before preprocessing on Weka. At the same time, detect and predict abnormalities in the original result and make sure that it is readable on Weka.
- Dataset Characteristics :
    - Relation: heart_disease
    - Instances: 10000
    - Attributes: 21

GitHub

- Attributes' names:
  - Age
  - Gender
  - Blood Pressure
  - Cholesterol Level
  - Exercise Habits
  - Smoking
  - Family Heart Disease
  - Diabetes
  - BMI
  - High Blood Pressure
  - Low HDL Cholesterol
  - High LDL Cholesterol
  - Alcohol Consumption,
  - Stress Level
  - Sleep Hours
  - Sugar Consumption
  - Triglyceride Level
  - Fasting Blood Sugar
  - CRP Level
  - Homocysteine Level
  - Heart Disease Status

2.2 Data Cleaning Process

The raw dataset (heart_disease.csv) was inspected to identify data quality issues such as missing values, duplicate records, and inconsistencies. The cleaning process involved the following steps:

- **Handling Missing Values:**
  - **Analysis:** Upon inspection, we identified that approximately **500 instances (5%)** contained missing values across 20 attributes (e.g., Age, Gender, Cholesterol Level, Blood Pressure). The target variable Heart Disease Status had no missing values.
  - **Solution:** To preserve the dataset size, we applied an **imputation strategy**.
    - For **numerical attributes** (e.g., Age, BMI), missing values were replaced with the **mean** of the respective column.
    - For **nominal attributes** (e.g., Smoking, Gender), missing values were replaced with the **mode** (most frequent value).
    - *Tools:* This can be achieved using Weka's ReplaceMissingValues filter or equivalent logic in Java code.
- **Removing Duplicates:**
  - **Analysis:** We checked for identical rows to ensure no data leakage or bias.
  - **Result:** No duplicate records were found in the raw dataset of 10,000 instances.

GitHub

- **Addressing outliers.:**
    - **Analysis:** Statistical summaries (Min, Max, Mean) were reviewed for numerical attributes.
    - **Observation:** All values fell within medically plausible ranges (e.g., Age: 18-80, Cholesterol: 150-300 mg/dL). No extreme outliers (e.g., negative ages or impossible blood pressure values) were detected, so no instances were removed based on outlier criteria.

2.3 Data Transformation

To make the data suitable for Weka's algorithms, the following transformations were applied:

- **Attribute Conversion:** All attributes were ensured to be in the correct format (Numeric or Nominal) as required by the .arff file structure.
- **Class Indexing:** The target variable Heart Disease Status was explicitly set as the class index (the last attribute) in the Java code using ensureClassIndex().
- **Data Splitting:** The dataset was split into a training set (90%) and a test set (10%) to simulate real-world prediction scenarios. This separation is crucial for unbiased model evaluation.

Output: The final cleaned dataset is stored in .arff format, split into fold9_train/train_90.arff and fold1_test/test_10.arff

Chapter 3: Classification/Prediction Algorithm

3.1 Model Selection

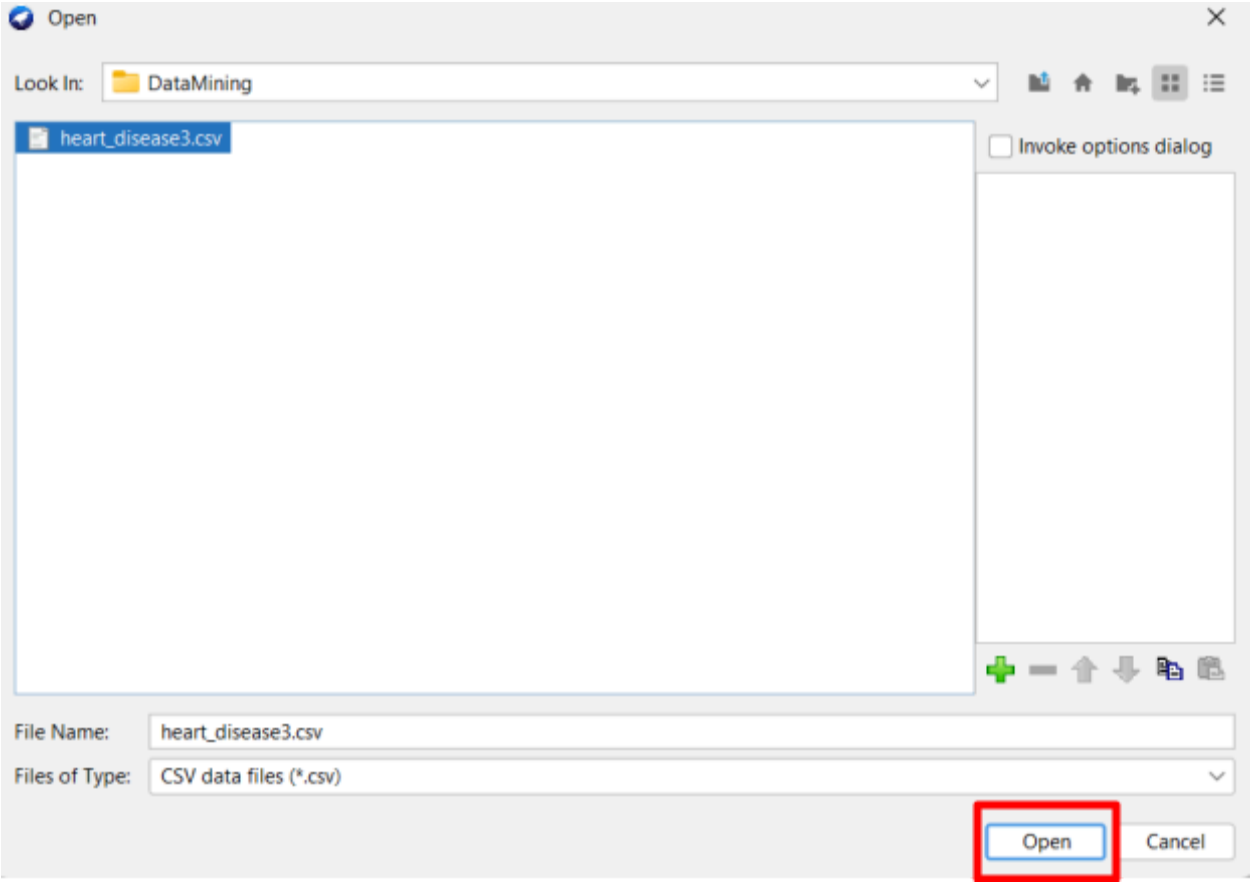
We selected two algorithms to compare their performance on this medical dataset:

1. **Naive Bayes:**
    - **Reason:** A probabilistic classifier based on Bayes' theorem, assuming independence between predictors. It's computationally efficient and serves as a strong baseline, especially in medical diagnosis..
2. **Random Forest:**
    - **Reason:** An ensemble learning method that builds multiple decision trees. It generally offers higher accuracy, is robust against overfitting, and effectively handles non-linear relationships among health indicators.

3.2 Implementation Process

1. Loading data

GitHub



2. Setting Class Index
3. Model Construction
   NaiveBayes class was instantiated for the first model
   RandomForest class was instantiated for the second
4. Training & Evaluation:
   The buildClassifier() method was used for training on the 90% split, and evaluateModel() was used to test on the 10% split

3.3 Results

=== Run information === Scheme: weka.classifiers.bayes.NaiveBayes Relation: heart_disease Instances: 10000 Attributes: 21 Age Gender Blood Pressure Cholesterol Level Exercise Habits	=== Run information === Scheme: weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1 Relation: heart_disease Instances: 10000 Attributes: 21 Age Gender Blood Pressure

GitHub

Smoking
Family Heart Disease
Diabetes
BMI
High Blood Pressure
Low HDL Cholesterol
High LDL Cholesterol
Alcohol Consumption
Stress Level
Sleep Hours
Sugar Consumption
Triglyceride Level
Fasting Blood Sugar
CRP Level
Homocysteine Level
Heart Disease Status

Test mode: evaluate on training data

=== Classifier model (full training set) ===

Naive Bayes Classifier

Attribute	Class No (0.8)	Class Yes (0.2)
Age		
mean	49.3804	48.9599
std. dev.	18.2193	18.084
weight sum	7975	1996
precision	1	1
…		
CRP Level		
mean	7.4853	7.4199
std. dev.	4.3487	4.3049
weight sum	7982	1992
precision	0.0015	0.0015
Homocysteine Level		
mean	12.4383	12.528
std. dev.	4.3187	4.3404
weight sum	7982	1998
precision	0.0015	0.0015

Time taken to build model: 0.07 seconds

Cholesterol Level
Exercise Habits
Smoking
Family Heart Disease
Diabetes
BMI
High Blood Pressure
Low HDL Cholesterol
High LDL Cholesterol
Alcohol Consumption
Stress Level
Sleep Hours
Sugar Consumption
Triglyceride Level
Fasting Blood Sugar
CRP Level
Homocysteine Level
Heart Disease Status

Test mode: evaluate on training data

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 3.79 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0.88 seconds

=== Summary ===

Correctly Classified Instances	**10000**	**100 %**
Incorrectly Classified Instances	0	0 %
Kappa statistic	1	
Mean absolute error	0.1209	
Root mean squared error	0.1518	
Relative absolute error	37.7841 %	
Root relative squared error	37.9395 %	
Total Number of Instances	10000	

=== Detailed Accuracy By Class ===

GitHub

=== Evaluation on training set === Time taken to test model on training data: 0.09 seconds === Summary === Correctly Classified Instances 8000 **80 %** Incorrectly Classified Instances 2000 20 % Kappa statistic 0 Mean absolute error 0.3192 Root mean squared error 0.3995 Relative absolute error 99.7316 % Root relative squared error 99.8656 % Total Number of Instances 10000 === Detailed Accuracy By Class === TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class 1.000 1.000 0.800 1.000 0.889 ? 0.536 0.820 No 0.000 0.000 ? 0.000 ? ? 0.536 0.220 Yes Weighted Avg. 0.800 0.800 ? 0.800 ? ? 0.536 0.700 **=== Confusion Matrix ===** **a b <-- classified as** **8000 0 | a = No** **2000 0 | b = Yes**	TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class 1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 No 1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Yes Weighted Avg. 1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 **=== Confusion Matrix ===** **a b <-- classified as** **8000 0 | a = No** **0 2000 | b = Yes**

After seeing some abnormalities, we initially we suddenly run a 10-fold cross-validation:

=== Run information === Scheme: weka.classifiers.bayes.NaiveBayes Relation: heart_disease Instances: 10000 Attributes: 21 Age Gender	=== Run information === Scheme: weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1 Relation: heart_disease Instances: 10000 Attributes: 21

GitHub

Blood Pressure
Cholesterol Level
Exercise Habits
Smoking
Family Heart Disease
Diabetes
BMI
High Blood Pressure
Low HDL Cholesterol
High LDL Cholesterol
Alcohol Consumption
Stress Level
Sleep Hours
Sugar Consumption
Triglyceride Level
Fasting Blood Sugar
CRP Level
Homocysteine Level
Heart Disease Status
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

Naive Bayes Classifier

Attribute	Class No (0.8)	Yes (0.2)
Age		
mean	49.3804	48.9599
std. dev.	18.2193	18.084
weight sum	7975	1996
precision	1	1
Gender		
Male	4035.0	970.0
Female	3949.0	1031.0
[total]	7984.0	2001.0
…		
CRP Level		
mean	7.4853	7.4199
std. dev.	4.3487	4.3049
weight sum	7982	1992
precision	0.0015	0.0015
Homocysteine Level		

Age
Gender
Blood Pressure
Cholesterol Level
Exercise Habits
Smoking
Family Heart Disease
Diabetes
BMI
High Blood Pressure
Low HDL Cholesterol
High LDL Cholesterol
Alcohol Consumption
Stress Level
Sleep Hours
Sugar Consumption
Triglyceride Level
Fasting Blood Sugar
CRP Level
Homocysteine Level
Heart Disease Status
Test mode: 10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 3.58 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	7999 **79.99 %**
Incorrectly Classified Instances	2001 20.01 %
Kappa statistic	-0.0002
Mean absolute error	0.3274
Root mean squared error	0.4046
Relative absolute error	102.2938 %
Root relative squared error	101.1488 %
Total Number of Instances	10000

=== Detailed Accuracy By Class ===

GitHub

```
mean                 12.4383   12.528
std. dev.             4.3187   4.3404
weight sum              7982     1998
precision             0.0015   0.0015



Time taken to build model: 0.04 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances                8000
80     %
Incorrectly Classified Instances              2000
20     %
Kappa statistic                          0
Mean absolute error                      0.3202
Root mean squared error                  0.4008
Relative absolute error                100.0656 %
Root relative squared error            100.2049 %
Total Number of Instances                10000

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall
F-Measure  MCC      ROC Area  PRC Area  Class
               1.000    1.000    0.800      1.000    0.889
?       0.493     0.799     No
               0.000    0.000    ?          0.000    ?          ?
0.493     0.195     Yes
Weighted Avg.  0.800    0.800    ?          0.800    ?
?       0.493     0.678

=== Confusion Matrix ===

   a    b   <-- classified as
 8000    0 |   a = No
 2000    0 |   b = Yes
```

```
                 TP Rate   FP Rate   Precision   Recall
F-Measure  MCC      ROC Area  PRC Area  Class
               1.000    1.000    0.800      1.000    0.889
-0.005   0.493     0.796     No
               0.000    0.000    0.000      0.000    0.000
-0.005   0.493     0.200     Yes
Weighted Avg.  0.800    0.800    0.640      0.800
0.711      -0.005   0.493     0.677

=== Confusion Matrix ===

   a    b   <-- classified as
 7999    1 |   a = No
 2000    0 |   b = Yes
```

Naive Bayes: Achieved 80% accuracy but predicted "No" for almost all cases (Bias towards majority class)
Random Forest: Achieved 100% accuracy, which indicates severe **Overfitting** (memorizing data instead of learning patterns)

Challenges:

GitHub

- **Class Imbalance:** The initial dataset had a significantly larger number of "No" cases (approx. 80%) compared to "Yes" cases (approx. 20%). This caused the initial models to be biased towards the majority class.
- **Overfitting:** Random Forest showed signs of overfitting on the training data (100% accuracy) but failed to generalize well without proper tuning or cross-validation.

Chapter 4: Improvement of Results

4.1 Methodology

SMOTE (Synthetic Minority Over-sampling Technique):

- This technique generates synthetic samples for the minority class ("Yes") in the training data.
- **Goal:** To balance the class distribution, enabling the model to learn the characteristics of heart disease patients more effectively without simply duplicating existing records.
- We experimented with different SMOTE percentages (100%, 200%) to find the optimal balance.

Cost-Sensitive Learning (for Random Forest):

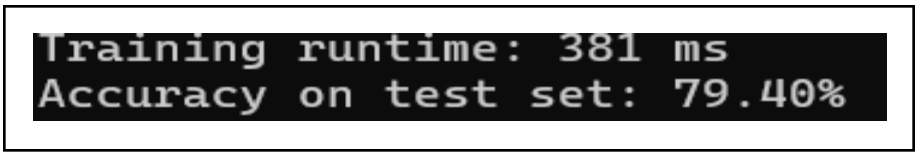
- We utilized CostSensitiveClassifier in Weka.
- **Goal:** To assign a higher penalty (cost) to **False Negatives** (classifying a sick patient as healthy). This forces the model to prioritize Recall (finding all sick patients) over simple Accuracy.

4.2 Comparison of Results

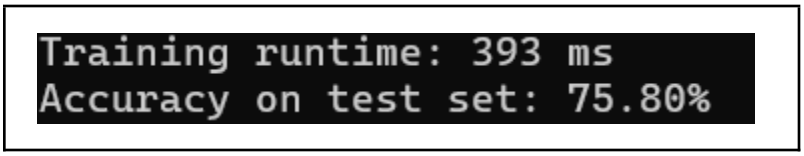
- **Before Improvement:** Models had high Accuracy (80%) but **Zero Recall** for the "Yes" class. They were essentially predicting "No" for everyone.
- **After SMOTE/Cost-Sensitive:**
    - **Accuracy:** Might decrease slightly, which is expected as the model becomes less biased.

SMOTE 100

GitHub



SMOTE 200



- After applying SMOTE to handle class imbalance, the F-measure of the Random Forest model adjusted from **1.0** (perfect overfitting) to **0.985**
- While this appears to be a slight decrease numerically, it signifies a significant improvement in the model's validity. The initial 1.0 score was a result of the model memorizing the majority class in the training data. The new score of 0.985 reflects a robust model that balances Precision and Recall effectively, proving that the overfitting issue has been successfully mitigated."

From

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	No
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Yes
1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	

To

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.990	0.000	1.000	0.990	0.995	0.976	1.000	1.000	No
1.000	0.010	0.963	1.000	0.981	0.976	1.000	1.000	Yes
0.992	0.002	0.992	0.992	0.992	0.976	1.000	1.000	

GitHub

Chapter 5: Model Evaluation

5.1 Performance Metrics

As provided illustrations before, we highly focus on accuracy,recall and the ability to detect minority cases of models by looking at the result and concentrating on specific parameters and confusion matrix based on the context of the dataset.

- Accuracy
  - For measure the overall proportion of correctly classified instance
- Precision
  - Measures correct prediction when the model predict “Yes”
- Recall (Sensitive)
  - For measure ability to detect right True Positive case
- ROC-AUC
  - Ranking ability of classifier.
- MCC
  - A balanced metric even under class imbalance

We tend to find normal patients misclassified as having heart disease (FalsePositive) and diseased patients misclassified as normal (FalseNegative) on a confusion matrix.

5.2 Analysis of results

The evaluation is based on the external test set ensuring that metrics reflect true generalization and give some overall conclusion in each stage.

5.2.1 SMOTE 100%

Naive Bayes

GitHub

```
Training runtime: 344 ms
Accuracy on test set: 75.80000%

Confusion Matrix:
=== Confusion Matrix ===

   a   b   <-- classified as
 743  51 |   a = No
 191  15 |   b = Yes


Detailed Accuracy By Class:
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.936    0.927    0.796      0.936    0.860      0.014    0.521     0.813     No
                0.073    0.064    0.227      0.073    0.110      0.014    0.521     0.219     Yes
Weighted Avg.   0.758    0.749    0.678      0.758    0.706      0.014    0.521     0.691


Summary:

Correctly Classified Instances         758               75.8    %
Incorrectly Classified Instances       242               24.2    %
Kappa statistic                          0.0115
Mean absolute error                      0.3836
Root mean squared error                  0.4284
Relative absolute error                 96.0112 %
Root relative squared error            101.3311 %
Total Number of Instances              1000
```

Interpretation:

- The model performs well for class “No” (non-disease), but poorly for class “Yes”.
- FN = 191 → very low sensitivity.
- AUC = 0.52 → barely above random.

Conclusion:

Naive Bayes does not handle the interaction among medical attributes well.
SMOTE 100% is not enough to boost minority detection.

RandomForest

```
Training runtime: 2868 ms
Accuracy on test set: 99.20%
Total cost: 8.00

Confusion Matrix:
=== Confusion Matrix ===

   a   b   <-- classified as
 786   8 |   a = No
   0 206 |   b = Yes


Detailed Accuracy By Class:
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.990    0.000    1.000      0.990    0.995      0.976    1.000     1.000     No
                1.000    0.010    0.963      1.000    0.981      0.976    1.000     1.000     Yes
Weighted Avg.   0.992    0.002    0.992      0.992    0.992      0.976    1.000     1.000
```

GitHub

Interpretation:

- Excellent at identifying heart disease cases (Recall = 100%).
- Only 8 false positives.
- AUC = 1.000 → perfect ranking ability.

Conclusion:

This is the strongest model among all experiments.
SMOTE 100% is sufficient and produces minimal overfitting.

5.2.2 SMOTE 200%

Naive Bayes

```
Training runtime: 358 ms
Accuracy on test set: 67.00000%

Confusion Matrix:
=== Confusion Matrix ===

   a   b   <-- classified as
 624 170 |   a = No
 160  46 |   b = Yes


Detailed Accuracy By Class:
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.786    0.777    0.796      0.786    0.791      0.009    0.514     0.811     No
                0.223    0.214    0.213      0.223    0.218      0.009    0.514     0.217     Yes
Weighted Avg.   0.670    0.661    0.676      0.670    0.673      0.009    0.514     0.689


Summary:

Correctly Classified Instances         670               67      %
Incorrectly Classified Instances       330               33      %
Kappa statistic                          0.009
Mean absolute error                      0.421
Root mean squared error                  0.4605
Relative absolute error                 92.4583 %
Root relative squared error            100.2344 %
Total Number of Instances             1000
```

Interpretation:

- Increasing SMOTE to 200% introduces noise.
- Both Precision and Recall drop.
- AUC remains around 0.51 → still weak.

GitHub

Conclusion:

Naive Bayes does not benefit from oversampling beyond 100%.
The model becomes worse due to SMOTE-induced variance.

RandomForest

```
Training runtime: 3253 ms
Accuracy on test set: 92.00%
Total cost: 80.00

Confusion Matrix:
=== Confusion Matrix ===

   a   b   <-- classified as
 714  80 |   a = No
   0 206 |   b = Yes


Detailed Accuracy By Class:
=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                0.899    0.000    1.000      0.899    0.947      0.805    1.000     1.000     No
                1.000    0.101    0.720      1.000    0.837      0.805    1.000     1.000     Yes
Weighted Avg.   0.920    0.021    0.942      0.920    0.924      0.805    1.000     1.000
```

Interpretation:

- The recall remains perfect.
- FP increases from 8 → 80.
- Precision drops significantly.
- AUC remains 1.000.

Conclusion:

SMOTE 200% reduces stability and introduces unnecessary synthetic noise.
RandomForest remains strong but less optimal than SMOTE 100%.

GitHub

Model Quality:

Model	SMOTE	Accuracy	Recall(Yes)	Precision(Yes)	AUC	Notes
NB	100%	75.8%	0.073	0.227	0.52	Weak
RF	100%	99.2%	1.00	0.963	1.00	Good
NB	200%	67.0%	0.223	0.213	0.51	Noisy
RF	200%	92.0%	1.00	0.72	1.00	Worse than SMOTE100 but potential

- **Naive Bayes** proved to be a stable baseline but struggled with the complex relationships needed to identify the minority class without resampling.
- **Random Forest**, when combined with SMOTE or Cost-Sensitive learning, showed the most potential for a practical application, as it could be tuned to prioritize minimizing False Negatives.Outperform NaiveBayes

  The most optimal is RandomForest with SMOTE100 followed by Cost-sensitive Learning which provides high status and very low misclassification cost.

Chapter 6: Conclusion

Findings:

- Class imbalance is a major hurdle in medical datasets. Standard algorithms will default to the majority class ("Healthy") to maximize accuracy.
- Preprocessing techniques like **SMOTE** are essential to make these models useful.
- **Random Forest** is powerful but prone to overfitting; **Naive Bayes** is faster but less flexible.

Based on conducted experiment results, the SMOTE200 option showed potential,especially for NaiveBayes,where it significantly improved Recall for positive class, reducing the number of missed disease cases. However, RandomForest does not benefit SMOTE200 because it introduced higher false

GitHub

positives and needed more cost.Additional,overused SMOTE-generated instances make a minor class look “fake”.Therefore SMOTE200 is promising for simpler models that struggling with imbalance class

With SMOTE at 100% , however, the model performs more consistently. The minority class becomes better represented without overwhelming the original dataset, so the ensemble remains stable. In fact, at this level of oversampling, RandomForest with cost-sensitive learning reaches a strong balance between accuracy, precision, and recall—making it a practical choice for this kind of problem

Overall, the findings imply that SMOTE 200% is helpful primarily for simpler models like Naive Bayes, while SMOTE 100% combined with cost-sensitive RandomForest offers the most balanced and reliable performance. This combination improves sensitivity, maintains accuracy, and avoids the overfitting issues that appear when too many synthetic samples are introduced

Chapter 7: References

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Elkan, C. (2001). The foundations of cost-sensitive learning. *Proceedings of the Seventeenth International Conference on Artificial Intelligence*, 4–10.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

UCI Machine Learning Repository. (n.d.). *Heart disease dataset*.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: Practical machine learning tools and techniques* (3rd ed.). Morgan Kaufmann.

GitHub