UNIVERSITY of BIRMINGHAM
SCHOOL of COMPUTER SCIENCE

Effect of Various Data Imputation Methods on Class Imbalance Problem

by

Darshitkumar M. Patel
Student ID: 2008468

Supervisor: **Dr. Leandro Minku**

A thesis submitted to the University of Birmingham for the degree of
MSc in Data Science

September 2022

Abstract

Beyond the realms of database management systems (DBMSs) and data engineering, the significance of data quality has been acknowledged. High data quality standards are also essential for machine learning (ML) applications to guarantee reliable prediction performance and responsible use of automated decision making. Missing values are one of the most prevalent data quality issues. When hidden, incomplete datasets can destroy data pipelines and have a catastrophic effect on downstream ML applications. The performance of most common classifier learning methods, which assume a comparatively balanced class distribution and equal misclassification costs, has been severely hindered when classifying data with uneven class distribution. The seriousness and recurrence of the class imbalance issue point to the need for additional study endeavours. The objective of this paper is to tackle both these problems together and provide a solution which can solve both the problems at once. We conduct a thorough study on a number of datasets with realistic missingness conditions, comparing classical ML imputation methods and class imbalance method together for training the model on these sets. Each method is evaluated on the predictive performance of ML model. Our findings shed important light on how various imputation techniques along with class imbalance method perform in practical settings. Our findings should advise academics and engineers in choosing a data pre-processing strategy for automatically enhancing data quality.

Keywords: Data imputation, class imbalance, SMOTE, MCAR, classification, missing data

Acknowledgement

I would like to thank my supervisor Dr. Leandro Minku for his constant guidance and support during the project and for encouraging me to explore the world of data imputation and class imbalance. During the implementation of project, he helped me to clarify my all the problem and gave me a lot of useful suggestion.

I would like to thank my inpsector Dr. Alexander Krull for giving feedbacks at every stage of this project. The feedbacks were really helpful.

Finally, I would like to thank my family for providing me with support and continuous encouragement throughout this year. They are the main source of inspiration and motivation in my life.

Table of Contents

Lists of Figures

Lists of Tables

1. Introduction

Our daily lives are incorporating machine learning more and more. The expansion of computational power and the ability to store vast amounts of data have transformed this field of study. This has made it possible for machine learning algorithms to perform well across a range of activities, sometimes even outperforming humans.

Modern business and institutions rely on the machine learning algorithms which in turns solely based on data. Data is used by online merchants, for instance, to estimate demand, schedule deliveries, and more generally to inform every single business operation and decision. Any decision-making process that follows is gravely jeopardised by incomplete or inaccurate information, which eventually harms the organization's overall effectiveness and efficiency.

Data can be lost due to application or communication failures, mistakenly not captured, purposefully left blank by users, or as a result of data integration mistakes, among other reasons. Researchers from several areas have been adding to the arsenal of techniques used to impute missing data throughout the course of the last few decades. Statisticians created the theoretical basis for missing value imputation (Donald, December 1976) by characterising different missingness patterns (further information are given in Section 4).

Simple solutions include discarding partial observations or substituting constant, mathematically sound values for missing variables. The amount of data available for downstream tasks is frequently reduced by such approaches, and depending on the missingness pattern, they may also bias downstream applications (Yang, et al., 2020), which further reduces data quality (Donald & Little, 2019). Nevertheless, this may be a reasonable solution to ensure the robust functioning of data pipelines.

Additionally, ML methods have been employed for imputation more and more recently. A few well-known techniques are k-nearest neighbours (k-NNs) (Monard & Batista, 2003), matrix factorization (Koren, et al., Aug. 2009) (Mazumder, et al., 2010) (Troyanskaya, et al., June 2001), random-forest-based methods (Stekhovan & Bühlmann, January 2012), discriminative deep learning methods (Biessman, et al., 2018), and generative deep learning methods (Shang, et al., 2017).

Researchers working on machine learning and data mining have recently become interested in the subject of class imbalance. When one class's instances outweigh those of other classes, it can happen. While the other class used the term minority class, the overcrowded class used the term majority class. The class's lesser-known cases, however, are frequently the more intriguing and significant ones in many applications. Every time the class of interest is somewhat uncommon and has a low number of occurrences in comparison to the majority class, the imbalance problem gets worse. In addition, the cost of misclassifying the minority class is far higher than the cost of misclassifying the majority class. For instance, compare the costs of identifying something as fraud or as cancer (Satuluri & Kuppa, September 2012).

Unbalanced classes present a variety of challenges and impair the effectiveness of machine learning and data mining approaches. Standard classifiers like decision trees and neural networks assume that the training samples are evenly distributed among the classes for calculating the class distribution. However, the percentage of the minority class is quite modest in many practical situations (1:100, 1:1000 or may exceeded to 1:10000). Due to a lack of data, the classifiers are more likely to mistakenly identify members of minority classes and set decision boundaries that are distant from the actual ones.

Concept complexity or overlapping, which refers to the degree of separability across data classes, sometimes causes problems. Higher complexity was achieved by more classes that overlapped and more

noise. Furthermore, the samples of each class may overlap in a feature space at various levels, making it challenging to infer discriminating rules (Sun, et al., 2007). The challenge is further complicated by the presence of minor disjuncts in a dataset. Furthermore, the cost of errors varies unevenly and is frequently unknown in the majority of imbalance problems.

In this project, we are dealing with both data imputation and class imbalance problem together. We are using three data imputation methods to fill the missing values such as Mean/ Mode Imputation, KNN imputation and Random Forest Imputation. For class imbalance problem, we are using SMOTE (Synthetic Minority Over Sampling Technique) (Chawla, et al., 2002). The aim of this project is to find best combination of data imputation method and class imbalance technique. To be precise, to find which data imputation method gives better performance with class imbalance technique SMOTE.

The project will be composed of two parts. The first part will be focusing on stimulating missing values and imputing the stimulated missing values. In other words, we will create the missing values in the datasets with varying missingness fractions. Since the datasets we are using in this project are complete datasets, we need to stimulate missing values. And then fill those missing values using data imputation methods: Mean/Mode imputation, KNN imputation and Random Forest imputation over the training set.

The second part will consist of class imbalance and modelling. After applying the data imputation methods, class imbalance technique called SMOTE will be applied. This technique handles the imbalance in the classification classes by oversampling the minority class by creating synthetic samples. The resulting sets are then modelled and evaluated using Random Forest Classifier.

Our intention is to get the best result after modelling such that if the dataset have missing values as well as class imbalance problem then our project will provide the best combination of both these problems altogether.

The format of this study is as follows. Section 2 will provide all the background information needed for the topic of this project. The work done in the fields of data imputation and class imbalance will be the main topic of Section 3. The dataset and performance measures utilised for evaluation will be described in Section 4. Section 5 elaborates the experimental design used for the experiments. We will present the outcomes of the experiments we conducted in section 6. Our study will be concluded in section 7. Finally, in section 8, suggestions for additional work will be provided.

2. Background

2.1. Data Imputation

The proper inclusion of missing data is one of the primary issues with data analysis. It is crucial to understand the distinction between empty and missing values. Empty values indicate that no value may be assigned, whereas missing values indicate that the variable's actual value exists but isn't available or recorded in the dataset for a variety of reasons. The data miner needs to distinguish between missing values and empty values. If not, both values will be considered to be missing. Missing data may be the result of equipment failure, inconsistency with other data that led to deletion, misunderstanding, or data that was not entered because it was not deemed necessary at the time the data was collected. Some data mining methods do not require the replacement of missing values since they were created and designed to manage missing values, but others are unable to do so.

It's crucial to comprehend the cause of missing data before attempting to deal with it. Three potential missing data mechanisms—Missing Completely at Random (MCAR), Missing at Random (MAR), and Not Missing at Random (NMAR)—were developed by (Donald & Little, 2019) and (Donald, December 1976).

Missing Data Mechanisms

The definition of Missing Completely at Random (MCAR) is that the pattern of missing values is completely random and independent of any variables that may or may not be included in the analysis. MCAR is the highest level of randomness. Therefore, if missingness does not depend on any dataset information, data is missing entirely at random. The MCAR model assumes that the probability of missingness is independent of both the observed values in the dataset's variables and the dataset's unobserved portions.

Missing at Random (MAR): In this scenario, the likelihood of missing data depends on the dataset's observed data. Meaning that the probability of missingness depends on the information that has been observed but not on the information that has not been seen. Because there is some association between the attribute holding the missing value and other attributes in the dataset, any variable in the dataset that has a missing value depends on the observed values of other variables in the dataset. The observed values in the dataset may allow one to identify the pattern of missing data.

Missing Not at Random (MNAR): In this situation, missingness depends more on unobserved than on observed data. Because the response variable is too sensitive to be answered, missingness depends on the missing data or item itself. Data that are MNAR have a relationship between the probability of missing data and the actual value of the missing data. When compared to the observed values of the other variables in the dataset, the pattern of missing data is unpredictable and not random.

These various kinds of missing data are significant because they dictate which statistical method can be successfully applied to the missing data. MNAR is frequently regarded as the worst type of missing data since it may provide biased results, whereas MCAR and MAR may result in a reduction in statistical power (Graham, 2009). It is always advised to gather as much data as you can about the causes of missing data. There are some techniques that could be utilised to tell MCAR from non-MCAR. comparing the traits of a group with missing values with the observed values for a certain variable using a t-test. The two groups will have distinct traits when the missing data are not MCAR. Because the sample size of the data is always a determining factor, this test is just informative. No mechanism exists to determine whether missingness is brought on by MAR or MNAR (Donald & Little, 2019).

Methods of Handling Missing Data

There are two distinct methods for dealing with missing data (Han, et al., 2022). The first tactic is to wilfully ignore missing data, and the second is to consider imputation of missing values.

Ignoring Missing Values: The technique for missing data ignoring omits the cases that have missing values. They are popular and frequently the go-to choice for handling missing data. This method's significant drawback is that it shrinks the dataset. When your dataset includes a modest number of missing values, this is fine. Listwise deletion (case deletion or complete case analysis) and pairwise deletion (available case analysis) are the two common methods for discarding missing data. The full case analysis method eliminates all observations with missing values for any relevant variable. As a result, this method restricts the analysis to instances for which all values are observed, which frequently yields skewed estimates and a reduction in precision. We analyse all instances in which the variables of interest are present while using pairwise deletion. The maximum amount of data from each unit is used rather than completely excluding any one unit. The benefit of this approach is that it preserves the greatest amount of data for analysis, even when some of its variables have missing values. This method's drawback is that it requires various sample sizes for various variables. Each individual analysis has a larger sample size than the overall case study.

Imputation of missing values: Imputation of missing values is a process that substitutes some plausible values for the missing value (Donald, December 1976). The numerous imputation methods are designed to give precise population parameter estimation, preserving the power of data mining and data analysis methods. The amount of missing data will determine the best course of action. If more than 25% of the data is missing, it is always preferable to compare the findings before and after imputation, even if there is no hard-and-fast rule about what percentage of missing data is undesirable.

Missing Data Imputation Methods

Data Imputation is the process of estimating missing data for an observation based on the valid values of other variables (Donald, December 1976). The statistical literature has published a lot about missing data (Donald & Little, 2019) (Schafer, 1997). The two categories of data imputation methods are single imputation method and multiple imputation method.

Single Imputation

It means performing analysis as if all data were initially observed while reproducing one reasonable value for each variable in the dataset that has a missing value. Here are a few common single data imputation techniques:

Imputation using the constant: Using this technique, the constant is used to impute the missing values. If the variable is a categorical one, "Missing," "0," or "999" could be used to represent all missing values.

Mean Imputation: This is the most popular technique for replacing missing data. It substitutes the missing value with the sample mean, median, or mode, depending on the data distribution. Although this method is straightforward and simple to use, it also has limitations. When there are several missing values, the mean of all those values is used as the imputation value, which changes the distribution's form. When you contrast the standard deviation before and after imputation, it becomes less significant. The standard deviation will decrease as the number of values is missing increases. By grouping data into subgroups, this approach can be somewhat enhanced.

Studies in simulations demonstrate that mean imputation does, in fact, produce highly biased parameter estimates (Graham, et al., 1994). The constraints of mean imputation, according to some studies, are essentially non-existent, nevertheless, if less than 10% of the data is missing and there is little connection between the variables (Tsikriktsis, 2005). Median and modus imputation are two methods comparable to mean imputation. Although those techniques were created to handle the imputation of data that is not normally distributed, they share the same drawbacks as mean imputation and are consequently not very well-liked data imputation techniques.

Imputation using distributions: In this method, random values from a known distribution are used to impute missing values. The distribution's shape is unaltered by the imputed value.

Regression Imputation: This single imputation method is a little more advanced. This approach replaces missing values with predicted data through regression using non-missing data from other variables. This method is based on assumption of linear relationship between the attributes. However, as relationships are rarely linear, applying regression to fill in the missing value will bias the model. Regression imputation has an advantage over mean imputation in that it can keep the distribution's shape. Results from this method could be skewed, especially when used with MNAR and MAR.

KNN Imputation: This technique copies values from related records in the same dataset to fill in the blanks for missing values. A distance function is used to gauge how similar the two attributes are. Although it is not necessary, building a prediction model for each feature has drawbacks as well. Analysing a big dataset takes a lot of time. It's crucial to pick the right k value.

Multiple Imputation

In single imputation approaches, precision is overestimated, and it is presumed that the single imputation value is the accurate one. However, the accuracy of imputed data can never be guaranteed in full. As a result, uncertainty around these imputed values must be considered in the missing data approaches. A technique for averaging the results over various imputed datasets was devised by (Donald, 2004). As a result, multiple imputation replaces many possible values for each missing observation rather than a single value to indicate ambiguity about the appropriate values to impute. Multiple Imputation generates "m" distinct complete datasets, each containing observed and imputed values, as a result. Three steps make up the multiple imputation method: (1) Imputation, which is similar to single imputation in that it generates missing values "m" times rather than just once. Therefore, following imputation, "m" distinct complete datasets may exist. (2) Analysis of each dataset: Each of the "m" datasets is examined after imputation and the generation of "m" distinct datasets. (3) Pooling: The results from each dataset that has been studied are finally combined.

2.2. Class Imbalance

When instances of one or more classes, known as the majority classes, outnumber instances of the other classes, known as the minority classes, a classification problem is said to be imbalanced. In contrast to the related problem of “within-class imbalance” (Japkowicz, 2001), such an imbalance in the data is known as the "between-class imbalance" (Japkowicz & Stephen, 2002). A within-class imbalance occurs when a class is made up of several different subclusters and these subclusters do not contain the same number of examples. A between-class imbalance occurs when the number of examples representing the positive class differs from the number of examples representing the negative class. In

the areas of medical diagnosis, science, and engineering, imbalanced issues are very common. Some examples include monitoring nosocomial infections (Cohen, et al., 2006), providing cardiac care, elucidating protein-protein interactions (Yu, et al., 2010), fraud detection, network intrusion detection and telecommunication management [10].

Consider the fact that the minority classes are typically the more significant classes when there is an imbalance problem. One or more nosocomial infections, for instance, affect 11% of patients (Cohen, et al., 2006). The majority class's occurrences are referred to as negative in the study of two-class imbalanced situations, whereas the minority class's instances are referred to as positive. The minority class is more significant in practice; thus, one should be more concerned with the positive examples. There are two approaches to tackling the imbalanced problem: re-sampling methods and imbalanced learning algorithms.

Re-sampling Methods

The re-sampling method actually involves re-balancing the given unbalanced data set. Although some other studies (Japkowicz & Stephen, 2002) have suggested that imbalanced data sets are not necessarily to blame for the poor performance of particular classifiers, studies (Weiss & Provost, 2003), (Estabrooks, et al., 2004) on class distribution have demonstrated that balanced data sets yield higher classification performance than imbalanced ones. Re-sampling methods are desirable in the majority of imbalanced situations. This is due to the fact that re-sampling merely modifies the initial training data set in place of changing the learning method. As a result, this method is portable and external (Estabrooks, et al., 2004), and it offers a simple solution for dealing with imbalanced learning issues when utilising common classifiers.

Under sampling: A non-heuristic technique called random under-sampling (Kotsiantis & Pintelas, 2004) seeks to balance class distribution by arbitrarily removing examples from the majority class. It is done with the intention of balancing the dataset in an effort to go over the limitations of the machine learning algorithm Random under-primary sampling's flaw is its tendency to omit potentially valuable information that can be crucial for the induction procedure. The fact that the goal of machine learning is for the classifier to estimate the probability distribution of the target population presents another issue with this strategy. We attempt to estimate the population distribution using a sample distribution because the distribution is unknown. According to statistics, the sample distribution can be used to estimate the population distribution from the location where it was drawn as long as the sample was chosen at random. Therefore, we can learn to approximate the target distribution by learning the sample distribution. However, the sample is no longer regarded as random after we under sample the majority class. As an alternative, there are guided under-sampling, where the decision to omit is made with knowledge rather than at random.

Over sampling: A non-heuristic technique called random over-sampling replicates minority class examples at random in an effort to equalize the distribution of social classes. Random over-sampling might enhance the risk of over-fitting since it creates precise replicas of the minority class samples, according to several authors (Chawla, et al., 2002). By doing this, a symbolic classifier, for example, may create rules that appear to be accurate but actually only apply to one reproduced sample. Additionally, if the data set is already reasonably big but sampling techniques mainly based on SMOTE.

Figure 1 Difference between Undersampling & Oversampling[1]

SMOTE: Synthetic Minority Oversampling Technique, known as SMOTE, generates synthetic minority examples to over-sample the minority class. The fundamental concept is to interpolate between a number of nearby minority class examples to create new minority class examples. All minority example's k-nearest neighbours of the same class are determined (in SMOTE, this value is set to 5), and then some examples are randomly chosen from them based on the oversampling rate. The next step is to create additional synthetic examples along the line between the minority example and its chosen closest neighbours. As a result, the overfitting issue is avoided, and the decision boundaries for the minority class move farther into the space used by the majority class.

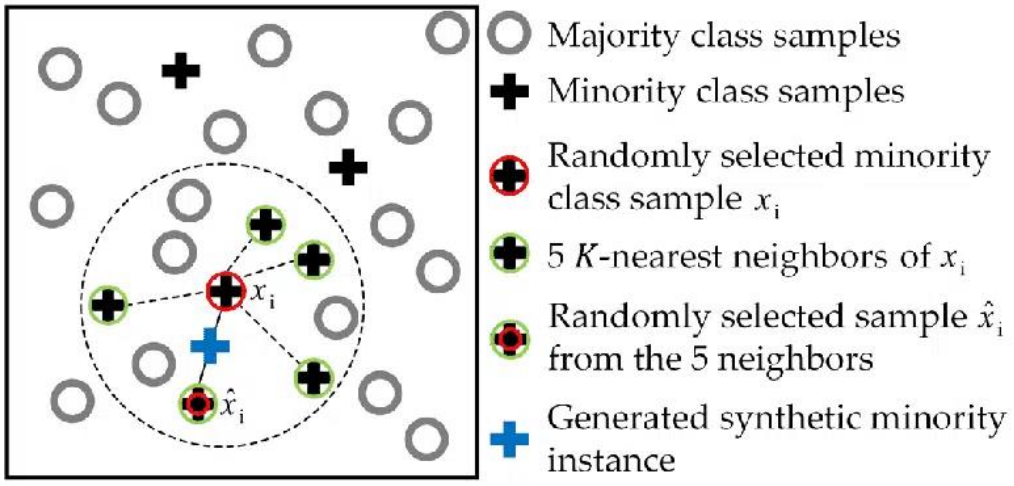

Figure 2 SMOTE working explained[2]

Imbalanced Learning Algorithms

The imbalanced learning algorithms approach can be thought of as a method to adjust or rebalance the current learning algorithms in order to make them more capable of handling imbalanced challenges. The cost-sensitive algorithm (Ting, May-June 2002), the probabilistic estimate at the tree leaf (when working with decision trees), modifying the decision threshold, and recognition-based (i.e., learning from one class) rather than discrimination-based (two class) learning (Japkowicz, 2001) are some algorithmic strategies that address the class imbalance.

Cost-Sensitive Method: Adding costs to decision-making is another method for enhancing classifier performance while learning from imbalanced datasets, in addition to altering the class distributions. The

cost of converting a sample from a true class j to class I corresponds to the matrix entry ij in the cost model, which is shown as a cost matrix in Table 1. This matrix is typically stated in terms of the problem's average misclassification costs. Typically, the diagonal elements are set to zero, indicating that accurate categorization is free. By selecting the class with the lowest conditional risk, the cost of misclassification can be reduced, which is the aim of cost-sensitive classification (Guo, et al., 2008).

Table 1 Cost Matrix

		Prediction	
		Class i	Class j
True	Class i	0	λ_{ij}
	Class j	λ_{ji}	0

Threshold Method: Various classifiers, including the Naive Bayes classifier and some Neural Networks, produce a score that indicates how much an example belongs to a given class. By changing the threshold of an example relevant to a class, such ranking can be utilised to create a number of classifiers (Kotsiantis, et al., 2006).

One-Class Learning: One-class learning is a recognition-based strategy that offers an alternative to discrimination by allowing the model to be built just from instances belonging to the target class. By putting a threshold on the similarity value (Japkowicz, 2001) between a query item and the target class, classification is achieved in this case. SVMs and auto-encoders were the two main classes of learners that were previously investigated in the context of the recognition-based one-class approach, and it was found that they were competitive (Manevitz & Yousef, 2001).

2.3 Modelling Algorithm

The random forest classifier is made up of several different tree classifiers, each of which is created using a random vector sampled separately from the input vector. Each tree then casts one unit of vote for the most prevalent class to categorise an input vector (Breiman, 1999). The Gini Index, which measures an attribute's impureness in relation to the classes, is used by the random forest classifier as an attribute selection metric. The Gini index can be stated as follows for a given training set T, selecting one case (pixel) at random and declaring that it belongs to some class C_i:

$$\sum\sum_{j\neq i}(f(C_i,\mathrm{T})/|\mathrm{T}|)(f(C_j,\mathrm{T})/|\mathrm{T}|)$$

where $f(C_i, \mathrm{T})/|\mathrm{T}|$ represents the likelihood that the chosen example belongs to the class C_i.

Each time a tree is developed to its maximum depth utilising a combination of features and fresh training data. These mature trees don't get pruned. This is one of the primary benefits of the random forest classifier over other decision tree techniques, such as (Quinlan, 2014). Studies indicate that the performance of tree-based classifiers is affected by the selection of the pruning methods rather than the attribute selection procedures (Pal & Mather, 2003). Two user-defined parameters are necessary to create a random forest classifier: the number of features used at each node to produce a tree and the number of trees to be developed. Only a few features are looked for at each node to find the best split. N trees make up the random forest classifier, where N is the number of trees to be generated and can be any value specified by the user. Each case of the data sets is handed down to each of the N trees in order

to classify a new data set. In that instance, the forest chooses the class with the most N votes. Below figures explains the structure and the working of random forest classifier.

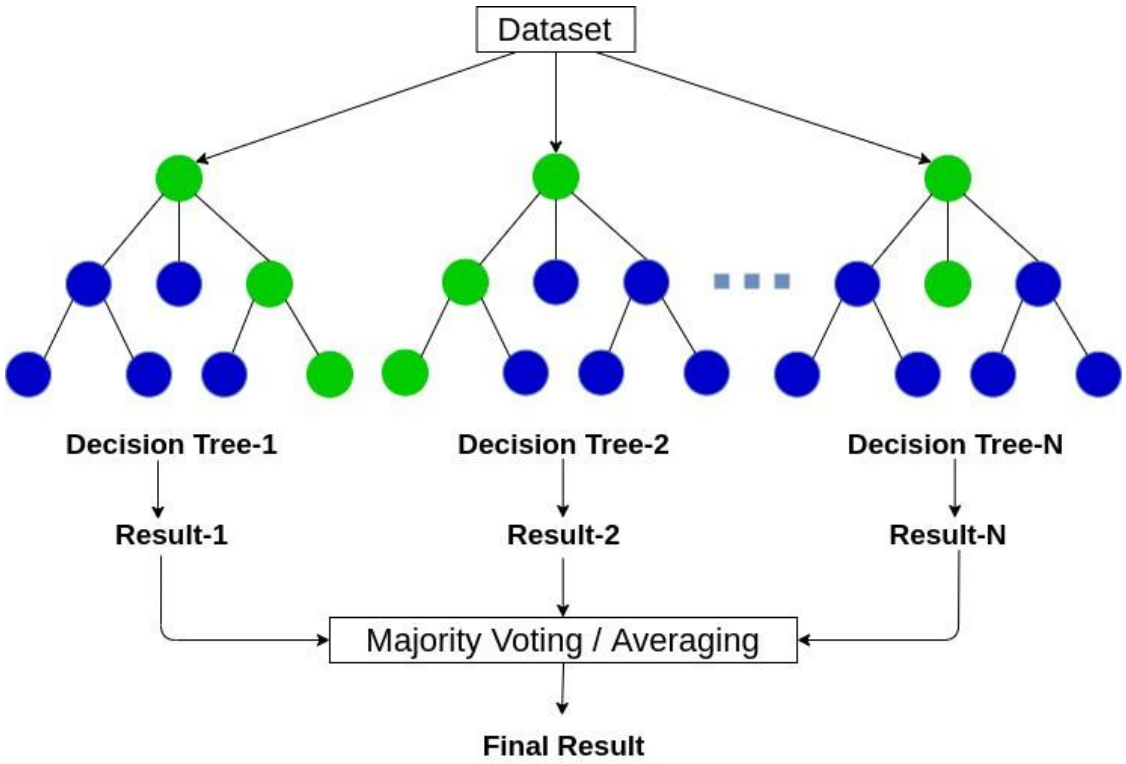

Figure 3 Structure of Random Forest[3]

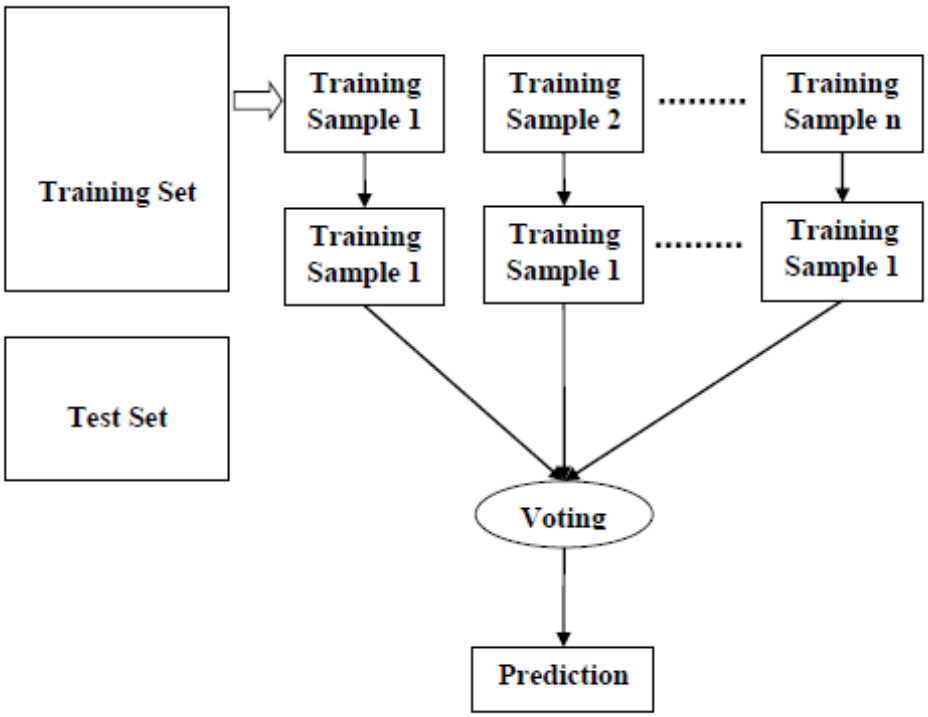

Figure 4 Working of Random Forest Classification Algorithm[3]

3. Related Work

Numerous research has reported various facets of data imputation techniques. The study by (Kyureghian, et al., 2011) evaluates imputation techniques by evaluating the accuracy of parameter estimates from subsequent regression analysis and the error of estimating the missing values. The study's findings demonstrate that multiple imputation techniques offer the best protection for both parameter estimates and dependent variable prediction. In their study, (Mishra & Khare, 2014) used a small dataset with variable levels of missingness to examine the effectiveness and suitability of several imputation techniques. The effects of missing data on several data mining methods, such as decision trees, k-nearest neighbours, association rules, and neural networks, were discussed by (Brown & Kros, 2003).

According to a study by (Geert, et al., 2006), multiple imputation produces accurately predicted standard errors and confidence intervals while the single imputation method produces estimates of standard errors that are too small. The performance of four approaches—kNN, mice, missForest, and Phylopars—for estimating missing values in a characteristics database is evaluated by (Penone, et al., 2014). (Schmitt, et al., 2015) contrast six data imputation techniques. Four genuine datasets of varying sizes were compared using the missing totally at random assumption. Their findings imply that methods like fuzzy k-means imputation and Bayesian principle component analysis should be given more practical consideration.

(Donald, 2004) talks about a thorough handling of multiple imputation. In (Tutz & Ramzan, 2015) their publication, suggested improved approaches for nearest neighbour imputation of missing data. They discovered that the proposed, improved nearest neighbour algorithm outperformed the alternatives. (Troyanskaya, et al., June 2001) examined the singular-value decomposition (SVD), mean imputation, and k-nearest neighbour imputation (KNNimpute) methods for gene expression data. Their simulation analysis shown that, when compared to mean imputation and SVD techniques, the KNN impute method performs well. (Malarvizhi & Thanamani, 2012) discovered that median or standard deviation substitution outperform mean substitution in a comparative evaluation of single imputation approaches. In (Nguyen, et al., 2004) their study, compared the KNNimpute approach to the mean, ordinary least squares (OLS), and partial least squares (PLS) imputation methods in microarray data and found that it performed well overall.

(Poulos & Valle, 2018) used two datasets to test different approaches for handling missing categorical data in supervised learning tasks. As an imputation technique, the k-nearest method was examined by (Monard & Batista, 2003). Their investigation revealed that the kNN approach performs better than the C4.5 and CN2 methods to handle missing data. The effects of missForest (MF), Multiple Imputation based on Expectation Maximization (MIEM), Sequential Hot-Deck, and Multiple Imputation based on Logistic Regression (MILR) on prediction accuracy over binary data were examined by (Ghorbani & Desmarais, 2017). Four distinct models—Tree Augmented Naive Bayes, Naive Bayes, Logistic Regression, and Support Vector Machine—are used to evaluate the effect (SVM). The outcome demonstrates that the MIEM approach produces the best outcomes for all classifiers across various percentages of missing data.

The issue of class imbalance has been addressed by the machine learning community in two different approaches. One is to give training examples different costs (Pazzani, et al., 1994) (Domingos, 1999). The alternative is to resample the original dataset, either by under- or oversampling the majority class (Ling & Li, 1998). The method used by (Chawla, et al., 2002) combines special oversampling of the minority class with undersampling of the majority class.

SMOTE was used by (Chawla, et al., 2002) in data that was skewed and imbalanced due to significant class skew and high sparsity. They employed the decision tree and Naive Bayes algorithms. Their findings highlighted how SMOTE performed well in sparse datasets. Four models were used by (Kamei, et al., 2007) to assess the effects of four sampling techniques (random oversampling, SMOTE, random oversampling, and one-sided selection) (linear discrimination analysis, logistic regression analysis, neural network and classification tree). The performance of the linear and logistic models, however, was enhanced by the sampling techniques, but the performance of the classification tree or neural network was unaffected.

With the use of decision tree induction, regression analysis, neural networks, and SVM, (Kerdprasop & Kerdprasop, March 2012) were able to enhance the performance of the learnt model. Random over sampling produces the model with the highest sensitivity, while SMOTE produces the model with the highest specificity. Additionally, they used a cluster-based feature selection technique, which significantly increased the learnt models' ability to predict outcomes. A new hybrid sampling technique called SMOTE with fuzzy rough set theory was introduced by (Ramentol, et al., 2012) (SMOTE-FRST). By removing the synthetic minority class samples that had lesser degrees of proximity to the fuzzy zone, they increased the performance of SMOTE. C4.5 was employed as a classifier to assess the proposed approach. Performance of SMOTE-FRST outperformed those of other SMOTE methods.

The related work demonstrates the variety of literature describing various approaches to data imputation techniques and class imbalance problems separately. None have described both together. This gives us an opportunity to work on this problem. Understanding and assessing the efficacy of various imputation techniques and class imbalance methods is crucial from the implementation standpoint so that the best of both techniques can be selected while carrying out data mining tasks. Even though there are some publications that have examined the effectiveness of various imputation techniques and class imbalance methods, in this paper we aim to examine the effectiveness of various imputation techniques along with class imbalance method for three different datasets that use single and multiple imputation techniques and a fixed class imbalance method for all imputation work. Specifically, we are using Mean/Mode Imputation, KNN Imputation, Random Forest Imputation for imputation task and SMOTE for class imbalance task. To sum up the contributions of our work, we provide a broad and inclusive study of data imputation technique and class imbalance method with respect to the following dimensions, which complements prior research.

1. The quantity and variety of datasets: 3 datasets with numerical and categorical columns are utilised.
2. Realistic missingness distributions and the number of missing values: We employ 1%, 10%, 30%, and 50% missing values with MCAR missingness patterns.
3. Imputation techniques: We use 3 techniques from single imputation to multiple imputation approach.
4. Class imbalance method: We use one method to tackle class imbalance.
5. Modelling and Evaluation: We use random forest classifier model to evaluate the best combination that performs well.

4. Datasets

We used three real-world imbalanced datasets from various sources to validate the new approach that was suggested.

4.1. Datasets

In ascending order of the positive-to-negative dataset ratio, Table 2 provides a summary of these datasets specifics. This includes the name of the dataset, the total number of examples (Total), attribute, the number of target classes for each dataset, the number of examples from minority and majority classes (#min.), and the attribute. These datasets show a wide range of fields, levels of complexity, and imbalance ratios.

On three real-world datasets, including those for heart disease, stroke prediction, and software defects, we assess the proposed method. The three datasets are obtained, two from Kaggle and one from zenodo repository. Table 2 provides a full description of the datasets.

Table 2 Datasets and their Properties

Dataset	Total	Class	#min/#maj
Heart Disease	918	2	410/508
Stroke Prediction	3426	2	180/3246
Software Defect	48989	2	12430/36559

Since the sampling of subsets involves unpredictability, the classification technique is repeated three times. These three runs are used to calculate the average AUC and F1-Score. The final values from this procedure are the averages of these three runs.

4.2. Performance Metrics

The most typical evaluation metric for the majority of traditional applications is accuracy. However, accuracy is not appropriate to assess imbalanced data sets because, as many practitioners have noted, for severely skewed class distributions, the recall of the minority class is frequently zero, indicating that no classification rules are formed for the minority class. The minority class, to use language from information retrieval, has substantially worse precision and recall than the majority class. It is challenging for a classifier to perform well on the rare classes since accuracy gives more weight to the common classes than to the rare classes. This has led to the broad adoption of new metrics.

As a result, the confusion matrix analysis is used to assess the performance of classifiers. A confusion matrix for a two-class problem with positive and negative class values is shown in Table 3. The classifier's results are shown in the column. The row represents a real value for the class label. The acronyms TP, FN, FP, and TN are the distinct names for the numbers that appeared in each cell of the matrix. The explanations for each acronym are:

TP stands for true positive, or the proportion of positive instances that are accurately classified as positive.

FP stands for false positive, or the proportion of negative instances that are mistakenly classified as positive ones.

FN stands for false negative, or the proportion of positive instances that are misclassified as negative cases. And

TN stands for true negative, or the proportion of negative instances that are accurately classified as negative.

Table 3 Confusion Matrix for two-class problem

	Positive Prediction	Negative Prediction
Positive Class	TP	FP
Negative Class	FN	TN

A number of widely used metrics for evaluating the performance of learning systems can be extracted from above matrix. In this project we will be using F1-Score and AUC score to evaluate our model.

F1-Score: The harmonic mean of recall and precision is used to obtain the F1 score. Remember that the harmonic mean is a substitute measure for the more often used arithmetic mean. It frequently comes in handy when calculating an average rate. We calculate the average of precision and recall for the F1 score. It is sensible to use the harmonic mean since they are both rates. The F1-Score formula is as follows:

$$\text{F1-Score} = 2 * \frac{Precison * Recall}{Precision + Recall}$$

It is important to note that F1-score accounts for both FPs and FNs because it considers precision and recall. The F1 score increases with increased precision and recall. The F1-score is a number between 0 and 1. The model is better the closer it is to 1.

AUC: Area Under the ROC Curve is referred to as AUC. The ROC curve is a two-dimensional graph with the TP rate and FP rate shown on the y-axis and x-axis, respectively. ROC curves, similar to precision-recall curves, can be utilised to evaluate various trade-offs. The ROC curve shows relative trade-offs between profits (TP rate) and costs (FP rate), meaning that while more positive instances can be accurately identified, more false positives must also be introduced. The performance is calculated using all categorization levels in this performance metric. The likelihood that the model will rank a random positive sample higher than a random negative sample can also be used to explain AUC. This performance statistic has a value between 0 and 1. The AUC would be 1 if the model's predictions were all accurate. The AUC, however, will be zero if the model completely mispredicted the data. This assessment measure is highly desired because it only calculates absolute values and is scale-invariant, which means it determines how well predictions are scored. Moreover, because it assesses the accuracy of the model's prediction, it is classification threshold invariant.

In this project, we are using the standard libraries to obtain the F1-Score and AUC value. For F1-Score, we use ‘sklearn.metrics.f1_score()’ and ‘sklearn.metrics.roc_auc_score()’ for AUC.

5. Experimental Setting

5.1 Data Pre-processing

To get decent results, data pre-processing is frequently a crucial component of ML workflows. In our tests, we use the following pre-processing procedures:

5.1.1 Target Column Distribution: We examine the target column distribution. Target column of each datasets have categorical binary values. Heart disease dataset used to predict whether the patient have a heart attack or not based on their age, sex, chest pain type, resting BP, cholesterol, etc. Stroke prediction dataset used to predict whether the patient have a stroke or not based on gender, age, work type, residence type, either they have hypertension, heart disease, bmi, smoking status, etc. Software defect dataset used to predict whether or not the software behaviour has defect based on entropy, age, etc. Heart disease dataset have total of 918 entries and the dataset is imbalance as 508 patients have heart disease and 410 do not have heart disease. Similarly, stroke prediction and software defect dataset are also imbalance. But as compared to heart disease dataset, stroke prediction and software defect datasets have high percentage of unbalanced data. About 94.74% of patients in stroke dataset do not have stroke and only 5.26% of all patients have stroke. In software defect dataset, the percentage difference of software does have defect and software does not have defect is 9.3%. Below Fig. 3, 4, 5 visualizes the same.

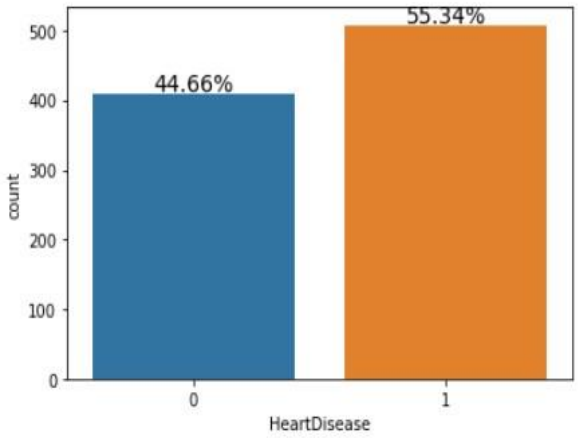

Figure 5 Heart Disease Target Column Distribution

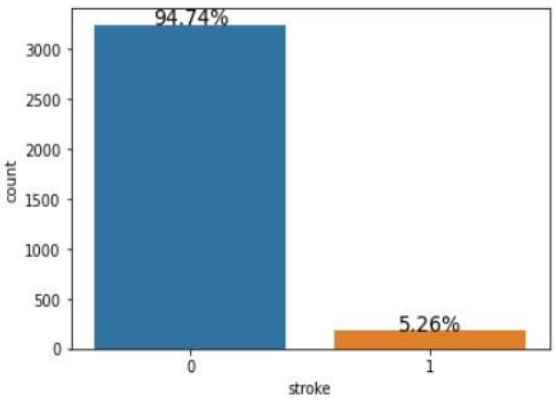

Figure 6 Stroke Prediction Target Column Distribution

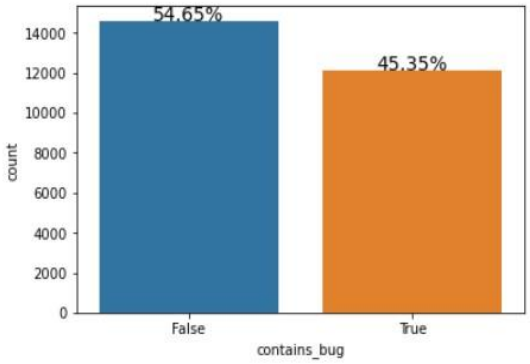

Figure 7 Software Defect Target Column Distribution

5.1.2 Splitting Datasets: All the datasets are split into training and testing sets with train size of 0.8. that means 80% of the data from dataset is allocated as training samples. Empirical studies shows that the best result is obtained if we use 80% of the samples as a training set and the remaining samples as testing set. As it gives more information for the model to learn from it. In this project, we are randomly splitting the datasets into training and testing sets. We created a function so that it can be used for all the dataset and which in turn reduce the redundancy of the code. We are split the dataset first because our datasets are complete that means datasets does not have any missing values in it. Because of this, we need to split the data first so that we are able to stimulate missing values (as explained **Section 5.2**) in it and apply imputation methods.

5.1.3 Encode Categorical Columns: Here we will encode the training and testing sets. A function is created in which we use one hot encoding with the help of pandas "get_dummies()" function and dropping the first column from dummies. This function is further used for all the datasets. In this part, we are only encoding the training set in which there are no missing values in it. Here, we used one hot encoding as it creates a new column for each unique value of a particular categorical column and it assigns 0 or 1 if that values is present in that row.

5.1.4 Data Normalisation: Here we are using "StandardScalar()" as it follows standard normal distribution. Therefore, it makes mean = 0 and scales the data to unit variance. Same as encode, a function is created to be able to use it for further application.

5.2. Stimulating Missing Values

Most missing value imputation research considers three different kinds of missingness patterns:

- Missing completely at random (MCAR): Values are eliminated without regard to any other values.
- Missing at random (MAR): Based on values in another column k c, values in column c are rejected.
- Missing not at random (MNAR): Depending on their value in column c, values in that column are rejected.

In the research on missing value imputation, MCAR is the missingness pattern that is most frequently utilised. The missing values in this case are picked at random on their own. The implementations of this condition typically select a random sample at random from a uniform distribution and discard the value. Here, we strive for realistic modelling of these missingness patterns, which is motivated by findings in sizable real-world datasets as explored in the work of (Biessman, et al., 2018).

We implemented this by using for loop in which we use ".sample()" function of pandas dataframe. This function has attribute "frac" which selects the fraction of sample provided. Here we are using 1%, 10%, 30% and 50% missingness fraction (MF) that means this value will be assigned to the above attribute. With the help of for loop and ".sample()" function, a function "missing_values()" is created to stimulate missing values with different missingness fractions in the training set only. Below Fig. 9 explain the application of the function "missing_values()".

This function creates 4 different training sets of various missingness fraction. These sets are not encoded and not normalized. They will be encoded and normalized in or after the imputation of missing value is done. The heatmap of for all the missingness fraction has been plotted to visualize. Below figures shows the graphs for the heart disease datasets. Similar graphs are plotted for the other two datasets.

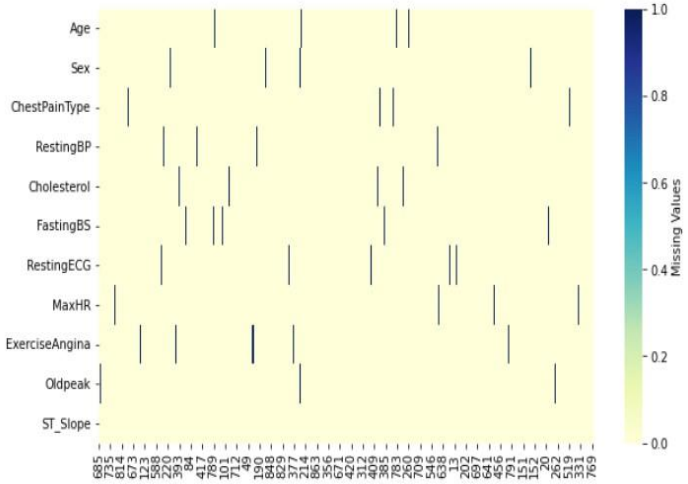

Figure 8 Heart Disease Training Set Heatmap with 1% MF

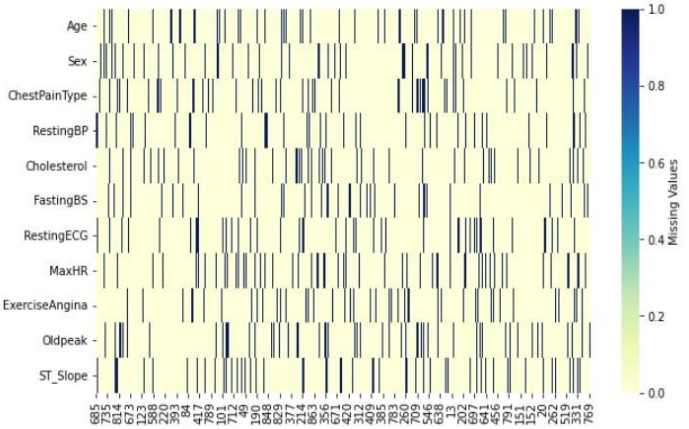

Figure 9 Heart Disease Training Set Heatmap with 10% MF

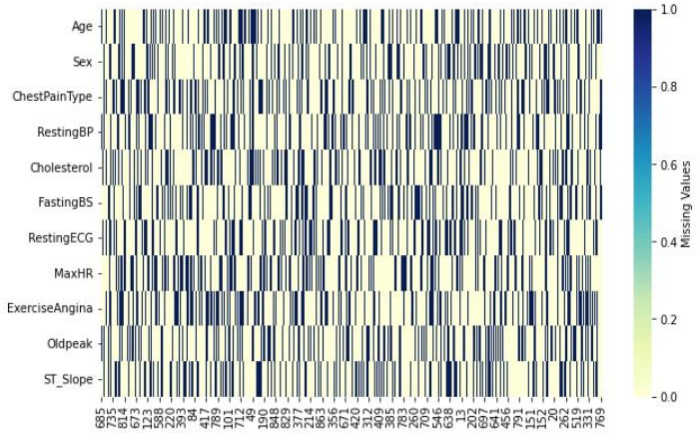

Figure 10 Heart Disease Training Set Heatmap with 30% MF

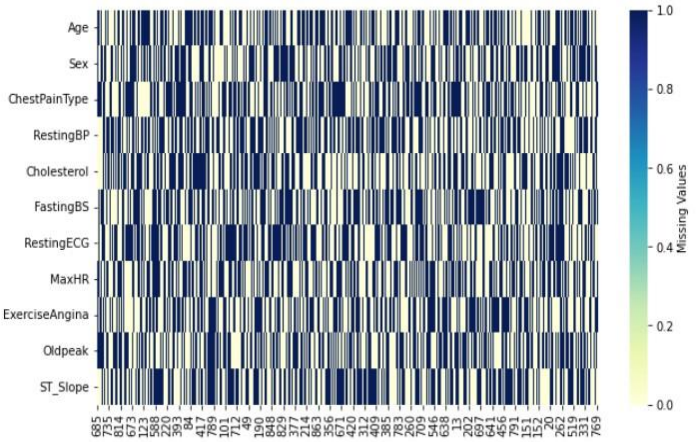

Figure 11 Heart Disease Training Set Heatmap with 50% MF

5.3 Data Imputation Techniques

In this section, we describe our three different data imputation techniques. We define a framework implemented by all the following imputation approaches. All these approaches are applied only on the training sets in which missing values are stimulated.

5.3.1 Mean/Mode Imputation: This method falls under the single imputation category (see **Section 2.1**). The most widely used method for replacing missing data is this approach. Depending on the distribution of the data, it replaces the missing value with the sample mean, median, or mode. In order to fill in missing values, we use the column-wise mean for numerical columns and the mode, or the most prevalent value, for categorical columns. This is achieved by creating a function which handle both the operations simultaneously. After the imputation is done, this function also encodes and normalizes the resulting imputed set. It uses the encode function (explained in **Section 5.1.3**) to encode and the normalize function (explained in **Section 5.1.4**) to normalize the resulting set.

5.3.2 KNN Imputation: K-NN imputation is a widely used ML imputation baseline. It falls under the single imputation category (see **Section 2.1**). Using a model to anticipate the missing values is an efficient method of data imputation. Input values from maybe all other input characteristics are used to develop a model for each feature that lacks values. A new sample is imputed by identifying the samples

that are "closest" to it in the given set and averaging those points to fill in the value. KNNImputer of sklearn's is used to apply this method. Choosing the distance metric (such as Euclidean) and the number of contributing neighbours for each prediction is a common step in KNN configuration. The training dataset's members' distances are calculated using a Euclidean distance measure that is NaN aware, which excludes NaN values from the calculation. The "metric" option is used to set this, and the value "nan_euclidean" is used by default. The "n_neighbors" argument can be used to change the default setting of five neighbours. Finally, although this is set to a uniform weighting by default, the distance measure can be weighed proportionally to the distance between instances (rows). This is done by controlling the "weights" option. In our project, we set "n_neighbors" value to 2 and all other configurations are set to default. KNNImputer cannot handle categorical data; hence, data transformation is required, and scaling the data is necessary for improved performance. That's why first of all we need to encode those categorical columns into a numeric value. For this purpose, we are using label encoder to encode and use the encoded data for imputation task. After the imputation task, we decode the data to get the original categorical values. As label encoder assigns serial number for each unique value of a particular categorical column. While training a model with those values in place, it creates some ranking relation in that column which leads to an incorrect prediction. Either way we are using one hot encoding to encode the categorical columns for our project. Encoding and normalisation of the resulting data has been done by using encode function and normalize function (see **Section 5.1.3** and **Section 5.1.4**).

5.3.3 Random Forest Imputation: A fun and effective method of imputation, Random Forest almost meets all the criteria for becoming the best imputation method. The Random Forests are fairly good at scaling to large data settings, and they can tolerate outliers and non-linearity in the data. In our implementation, we use IterativeImputer to apply random forest imputation. A procedure known as iterative imputation models each feature as a function of all other features, such as in a regression issue when missing values are forecasted. Since each characteristic is imputed one after the other, previous imputed values can be included into a model to predict new features. It is iterative because several repetitions of this method allow for the calculation of ever-improving estimates of missing values as missing values for all features are approximated. To estimate the missing values for each feature, a variety of regression algorithms can be utilised, albeit for simplicity, linear methods are frequently chosen. Iterations of the process are frequently limited to a few, like 10 which is default value. Finally, the initial strategy like mean (default), median, most frequent, or constant to be employed so that the missing value gets filled initially. To fill categorical column in our project, we are using RandomForestClassifier as an estimator to use in each step as round-robin imputation and initial strategy as most frequent. For numerical columns, estimator will be RandomForestRegressor and initial strategy to be mean. Maximum iterations for both the estimators are set to be 5. Similar to KNN imputation, this imputation also does not support string values. So, we need to set up same encoding and decoding mechanism as set up in KNN imputation. At last, we will encode the resulting data and normalize it. Below (Fig. 16) line of code better explains this.

5.4. Class Imbalance using SMOTE

After the imputation task is completed, now is the time to handle class imbalance problem. To handle class imbalance, here we are using only one method which is mostly used by many researchers called SMOTE. It is an oversampling method in which synthetic minority samples are created to oversample the minority class. The basic idea is to build new minority class samples by interpolating between a

number of neighbouring minority class examples. In SMOTE, this value is set to 5, and after determining all minority instances' k-nearest neighbours in the same class, some examples are randomly selected from them based on the oversampling rate. Creating more synthetic instances along the axis between the minority example and its predetermined closest neighbours is the next phase. As a result, the overfitting problem is avoided, and the minority class's decision-making boundaries extend further into the domain used by the majority class.

We are using imblearn.over_sampling's SMOTE object to implement this method. A function is created such that an instance is created, and that instance is fitted and resampled to overcome class imbalance in the dataset. Below explains the function of code.

5.5. Modelling

This section discusses the use of a classification model to model the data processed by the data imputation techniques and class imbalance method. The processed data resulting from the data imputation techniques and class imbalance handling is ready to be taken as input for the modelling, which is next part of this project. For this part of project, we are using RandomForestClassifier to build a classification model on the processed data. A function to perform the modelling task is created. So, this defined function not only build a model but also fits the model, predict the outcomes and evaluate the model by using F1-Score and AUC score. The results are stored as a dictionary and will be assign to a variable upon each iteration of calling. This function takes training and testing set as an input. For output, it gives a confusion matrix, F1-Score and AUC which is printed on calling for each iteration as well as F1-Score and AUC are stored for further analysis. A sample of the function is shown below.

```
def model(mf, im, X_train,y_train,X_test, y_test):

    rf_cls =  RandomForestClassifier(random_state=0)
    rf_cls.fit(X_train, y_train)
    print("Model trained.")


    result = {}
    print("\nModel Performance\n-----------------")
    y_pred = rf_cls.predict(X_test)

    #use model to predict probability that given y value is 1
    y_pred_proba = rf_cls.predict_proba(X_test)[::,1]

    print(
        "\n" + "\n\t\t\t\t\tF1-Score: {:.5f}\n\t\t\t\t\t     AUC: {:.5f}\n\t\t\t\t"\
        .format(f1_score(y_test, y_pred), metrics.roc_auc_score(y_test, y_pred_proba))
    )

    result[lst[0]] = [mf,im,"%.4f" % f1_score(y_test, y_pred), "%.4f" % metrics.roc_auc_score(y_test, y_pred_proba)]
    lst.pop(0)

    # Print the confusion matrix using Matplotlib
    conf_matrix = confusion_matrix(y_true=y_test, y_pred=y_pred)
    fig, ax = plt.subplots(figsize=(5, 5))
    ax.matshow(conf_matrix, cmap=plt.cm.Oranges, alpha=0.3)
    for i in range(conf_matrix.shape[0]):
        for j in range(conf_matrix.shape[1]):
            ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

    plt.xlabel('Predictions', fontsize=18)
    plt.ylabel('Actuals', fontsize=18)
    plt.title('Confusion Matrix', fontsize=18)
    plt.show()

    return result
```

Figure 12 Model Function

After handling missing values and imbalance classes, we get thirteen different training sets. In those training sets, we have varying missing values which are filled with by using three imputation techniques and these sets are further fetched to balance the class. Table 4 lists all the variables according to the missingness fraction. The testing set is same for all the training sets. Same number of variables are obtained for each datasets.

Table 4 Lists of variables

Missingness Fraction	X_train	y_train
0%	X_train_heart_smote	y_train_heart_smote
1%	X_train_heart_1_mmi_smote	y_train_heart_1_mmi_smote
	X_train_heart_1_knn_smote	y_train_heart_1_knn_smote
	X_train_heart_1_rf_smote	y_train_heart_1_rf_smote
10%	X_train_heart_10_mmi_smote	y_train_heart_10_mmi_smote
	X_train_heart_10_knn_smote	y_train_heart_10_knn_smote
	X_train_heart_10_rf_smote	y_train_heart_10_rf_smote
30%	X_train_heart_30_mmi_smote	y_train_heart_30_mmi_smote
	X_train_heart_30_knn_smote	y_train_heart_30_knn_smote
	X_train_heart_30_rf_smote	y_train_heart_30_rf_smote
50%	X_train_heart_50_mmi_smote	y_train_heart_50_mmi_smote
	X_train_heart_50_knn_smote	y_train_heart_50_knn_smote
	X_train_heart_50_rf_smote	y_train_heart_50_rf_smote

5.6. Hyperparameters Optimization

Gaining knowledge about a model's performance, resilience, and training duration requires optimising hyperparameters. The “n_neighbors” is set to 2 for knn imputation and “weights” is set to “uniform”. For random forest imputation, we are using iterative imputer. So, in iterative imputer we set “estimator” to “RandomForestClassifier” for categorical columns and “RandomForesrRegressor” for numerical column and “max_iter” is set to 10 for both the columns. All other parameters are kept default like random state is not defined for train test split , random forest imputation and classification model.

Table 5 A summary of all imputation methods, the class imbalance method, and the hyperparameters we optimised for it. Iterative imputer is for implementing Random Forest Imputation

Method	Hyperparameter	
	Name	Value
Mean/Mode Imputation	-	-
KNN Imputation	n_neighbors	2
Iterative Imputer	estimator	RandomForestClassifier, RandomForestRegressor
	initial_strategy	most_frequent, mean
	max_iter	10
SMOTE (Class Imbalance)	-	-
Random Forest Classifier Model	-	-

6. Results and Performance Evaluation

This section displays the outcomes of all potential pairings of each data imputation techniques with class imbalance method including the original datasets where missing value is not stimulated, along with model's performance on its own. Here, to evaluate the performance of our approach, we are using F1-Score and AUC.

6.1 Role of F1-Score: By calculating the harmonic mean of a classifier's precision and recall, the F1-score integrates both into a single metric. Therefore, both false positives and false negatives are considered while calculating this score. Assume classifiers A and B have higher recall and precision, respectively. The F1-scores for both classifiers in this situation can be used to assess which one yields superior results. The smaller of the two numbers is typically closer to the harmonic mean of two numbers. Therefore, a high F1-Score value guarantees that both recall and precision are at a respectable level.

6.2 Role of AUC: An indicator of performance for classification issues at different threshold levels is the AUC-ROC curve. An easy way to determine which classifier performs better overall is to look at the area under a ROC curve (AUC). AUC stands for the degree or measure of separability, and ROC is a probability curve. It reveals how well the model can differentiate across classes. The strong closeness between AUC and well-known Wilcoxon statistics has been demonstrated in (Hanley & McNeil, 1982).

6.3 Result Comparison: This subsection covers the comparison of the performance of each data imputation with class imbalance along with classification model. The result is evaluated with three different datasets with the help of F1-Score and AUC. All the result we are going to discuss are average of results from 3 runs.

6.3.1 Heart Disease Dataset:

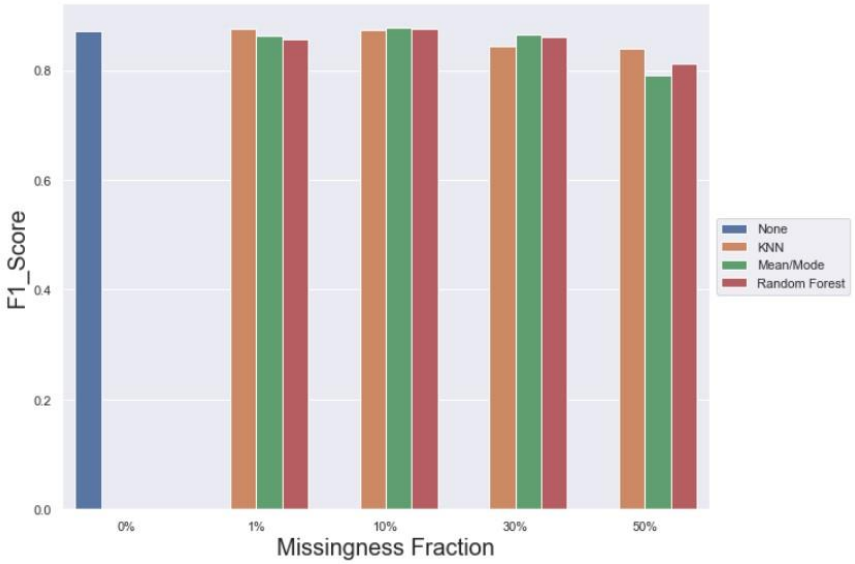

Figure 13 Average F1-Score of Heart Disease Dataset

From the above graph, we can clearly say that the F1-Score of Mean/Mode Imputation with 10% and 30% stimulated missing value performs better than the other two techniques. But with 1% and 50% stimulated missing values, KNN Imputation gives better results than the Random Forest Imputation and Mean/Mode Imputation. The reason behind mean/mode imputation beats other two new techniques with 10% and 30% stimulated missing values is

as you remove original distribution of data mean/mode create its own distribution of data. But this fails if you have more than 30% missing values. In that case, KNN imputation becomes more robust. From the above graph, we can say that random forest imputation is also performs very competitive with mean/mode imputation.

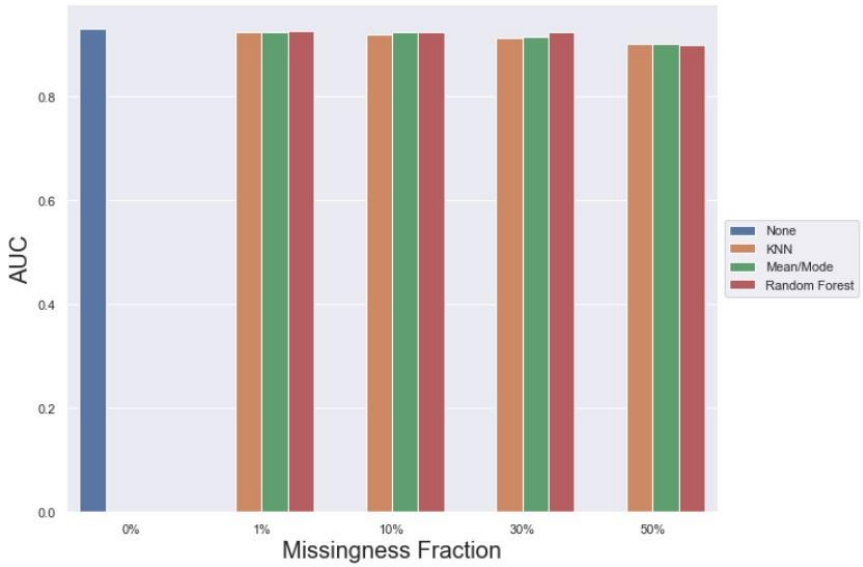

Figure 14 Average AUC of Heart Disease Dataset

As opposed to the F1-Score, random forest imputation gives better AUC value. This is because F1-Score is calculated from the precision and recall or by using confusion matrix and AUC is calculated by varying the threshold of the probability distribution and report the best results at a best threshold. So, random forest imputation along with SMOTE gives better classification results in almost all missingness fraction.
The results of F1-Score and AUC are tabulated in below figure.

		F1_Score	AUC
Missingness Fraction	Imputation method		
0%	None	0.87065	0.92970
1%	KNN	0.87485	0.92360
	Mean/Mode	0.86205	0.92315
	Random Forest	0.85715	0.92445
10%	KNN	0.87355	0.91940
	Mean/Mode	0.87720	0.92260
	Random Forest	0.87560	0.92375
30%	KNN	0.84450	0.91195
	Mean/Mode	0.86415	0.91460
	Random Forest	0.86000	0.92420
50%	KNN	0.83900	0.90030
	Mean/Mode	0.78975	0.89970
	Random Forest	0.81130	0.89955

Figure 15 Heart Disease Final Results

6.3.2 Stroke Prediction Dataset:

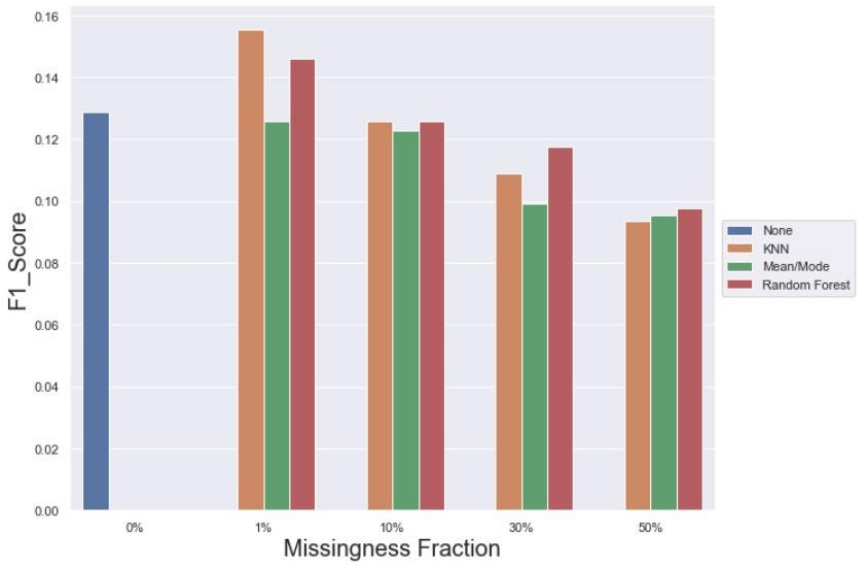

Figure 16 Average F1-Score of Stroke Prediction Dataset

Random forest imputation with SMOTE gives very compelling result for the stimulated missing values more than 1%. With 1% missingness fraction, knn imputation outperformed the best performing random forest imputation. Also, it is interesting to see that the results of original dataset which does not have any missing values gives poor result as compared to 1% stimulated missing value. We can say that there is significant improvement the prediction of stroke if we create 1% of missing value in the complete dataset. As the missingness fraction increases the prediction performance of the classifier does not show any significant increase.

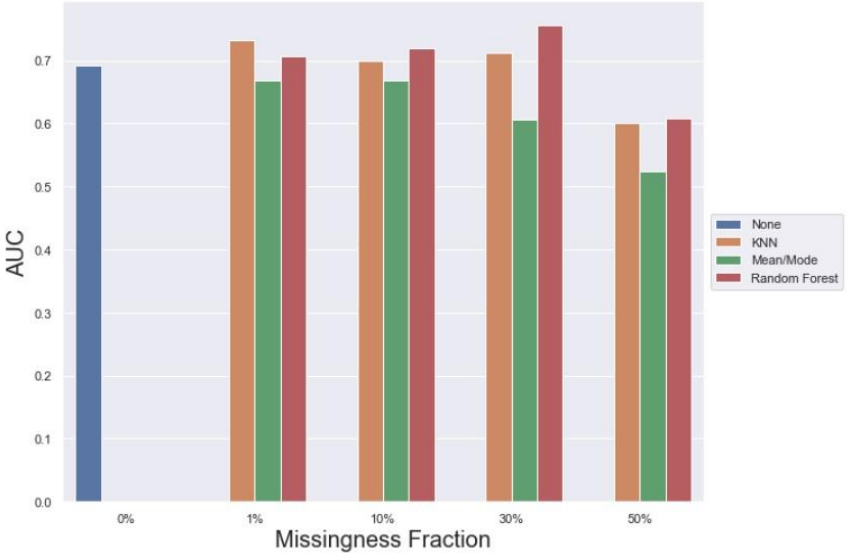

Figure 17 Average AUC of Stroke Prediction Dataset

Similar to the F1-Score, AUC also gives the same result in predicting stroke. Random forest imputation along with SMOTE is recommended for missingness fraction of more than 1% and for 1% missing value, knn imputation along with SMOTE is recommended.
The results of F1-Score and AUC are tabulated in below figure.

Missingness Fraction	Imputation method	F1_Score	AUC
0%	None	0.12890	0.69165
1%	KNN	0.15535	0.73205
	Mean/Mode	0.12590	0.66775
	Random Forest	0.14620	0.70580
10%	KNN	0.12560	0.69885
	Mean/Mode	0.12280	0.66840
	Random Forest	0.12590	0.71865
30%	KNN	0.10900	0.71125
	Mean/Mode	0.09900	0.60665
	Random Forest	0.11735	0.75540
50%	KNN	0.09365	0.60145
	Mean/Mode	0.09550	0.52315
	Random Forest	0.09760	0.60720

Figure 18 Stroke Prediction Final Results

6.3.3 Software Defect Dataset:

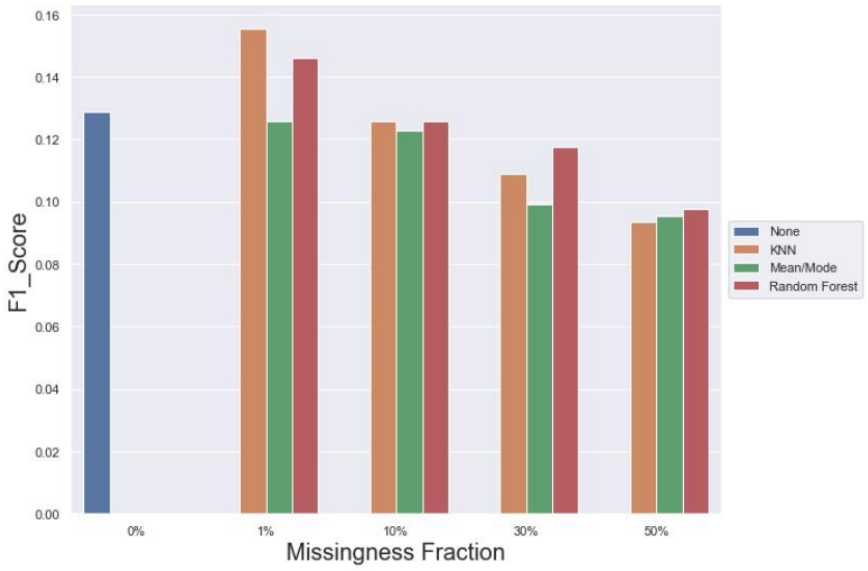

Figure 19 Average F1-Score of Software Defect Dataset

Results for software defect dataset is very similar to that of stroke prediction dataset. Here also the random forest imputation with SMOTE outperforms other techniques with missingness fraction more than 1%. KNN imputation is best for 1% missing values along with SMOTE. Same increase in performance in detecting the software defect has seen if we

create 1% missing value in the complete dataset as compared to original dataset. But afterwards the performance gets decreases with increase in missingness fraction as compared to the original dataset.

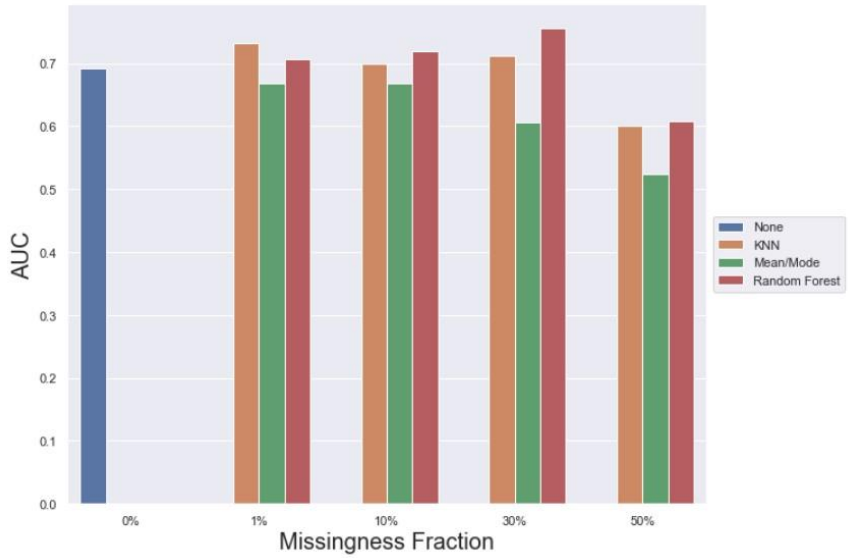

Figure 20 Average AUC of Software Defect Dataset

The AUC of random forest imputation with SMOTE at 10% and 30% missingness fraction shows outstanding increase in the performance of detecting software defect as compared to the original dataset which does not have any missing value. In case of 1% missingness fraction, knn imputation with SMOTE shows promising results than the other two techniques. Overall, random forest imputation with SMOTE shows best outcome.
The results of F1-Score and AUC are tabulated in below fig.

		F1_Score	AUC
Missingness Fraction	Imputation method		
0%	None	0.12890	0.69165
1%	KNN	0.15535	0.73205
	Mean/Mode	0.12590	0.66775
	Random Forest	0.14620	0.70580
10%	KNN	0.12560	0.69885
	Mean/Mode	0.12280	0.66840
	Random Forest	0.12590	0.71865
30%	KNN	0.10900	0.71125
	Mean/Mode	0.09900	0.60665
	Random Forest	0.11735	0.75540
50%	KNN	0.09365	0.60145
	Mean/Mode	0.09550	0.52315
	Random Forest	0.09760	0.60720

Figure 21 Software Defect Final Results

7. Conclusion

In this study, we carried out a thorough study for imputation techniques and class imbalance method, comparing traditional and contemporary approaches on a number of datasets under realistic missingness conditions with regard to their impact on performance of the model.

The issue with the class imbalanced dataset is that, in the case of a little imbalance, a classification issue may be slightly prejudiced. In contrast, the classification problem may have a significantly biased, where, for a given training dataset, there may be hundreds or thousands of examples in one class and only a handful in another. In our study, we work with both type of class imbalance in which the heart disease dataset is slightly imbalance while other two, stroke prediction and software defect datasets, are largely imbalanced.

From the final result comparison, we conclude that random forest imputation works well in most of the datasets with higher percentage of missingness fraction of upto 50% along with class imbalance method SMOTE. KNN imputation is suggested from our study for missing value is no more than 1%. It is also seen that the predictive performance of ML model trained on knn imputed set is better as compared to that trained on complete set (which does not have any missing values stimulated). But as we increase the missingness fraction in the training set, the predictive performance of ML model decreases as compared to that complete training set.

Two key conclusions may be drawn from our research as a whole. First, we show that, independent of the missingness conditions, imputation significantly improves downstream predictive performance. Our findings show that, when training data have 1% missing value, imputation improves the predictive performance of ML models by 10% to 20% in more than 75% of our studies. Second, we discover that, in nearly all studies, imputation based on random forests with SMOTE produces the greatest results in terms of the performance of ML model prediction with most of the missingness fraction. This result is consistent with earlier imputation benchmark studies.

We are confident that the experimental setup created in this study can improve the testing of imputation techniques and class imbalance method, in the end, assist in stress testing these techniques together under real-world circumstances in significant unified study with heterogeneous datasets. The best combination of data imputation technique and class imbalance method coming out to be Random Forest for imputation tasks and SMOTE to tackle class imbalance problem.

8. Future Work

We took several decisions that limit our results because one of the key objectives of this study is finding best combination of imputation techniques and class imbalance method on a wide range of datasets and missingness situations.

First, we restrict class imbalance to only one method which is SMOTE. Reason behind this is limited timeframe of project. And also, the method we used is considered as the best method to handle class imbalance problem. Second, limited use of imputation techniques. Our projects aim is to check the best imputation technique among all the techniques that includes traditional techniques too. Because of that we only choose three imputation techniques to handle missing values.

In future, more imputation techniques and more class imbalance methods can be included and studied further to get the best combination.

9. References

Biessman, F. et al., 2018. *"Deep" Learning for Missing Value Imputationin Tables with Non-Numerical Data.* s.l., 27th ACM International Conference on Information and Knowledge Management (CIKM '18).

Breiman, L., 1999. *RANDOM FORESTS--RANDOM FEATURES,* Statistics Department, University of California, Berkeley: Technical Report 567.

Brown, M. L. & Kros, J. F., 2003. Data mining and the impact of missing data.. *Industrial Management & Data Systems,* Volume 103, pp. 611-621.

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

Cohen, G. et al., 2006. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial Intelligence in Medicine,* 37(1), pp. 7-18.

Domingos, P., 1999. Metacost: A General Method for Making Classifiers Cost-sensitive.. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 155-164.

Donald, R. B., 2004. *Multiple Imputation for Nonresponse in Surveys.* s.l.:John Wiley & Sons.

Donald, R. B., December 1976. Inference and missing data. *Biometrika,* 63(3), p. 581–592.

Donald, R. B. & Little, R. J. A., 2019. *Statistical Analysis with Missing Data..* 3rd ed. s.l.:s.n.

Estabrooks, A., Jo, T. & Japkowicz, N., 2004. A Multiple Resampling Method for Learning from Imbalanced Data Sets.. *Computational Intelligence,* 20(1), pp. 18-36.

Geert, J. M. G. et al., 2006. Imputation of missing values is superior to complete case analysis and the missing-indicator method in multivariable diagnostic research: a clinical example.. *Journal of Clinical Epidemiology,* 59(10), pp. 1102-1109.

Ghorbani, S. & Desmarais, M. C., 2017. Performance Comparison of Recent Imputation Methods for Classification Tasks over Binary Data. *Applied Artificial Intelligence,* 31(1), pp. 1-22.

Graham, J. W., 2009. Missing data analysis: Making it work in the real world.. *Annual review of psychology,* 60(1), pp. 549-576.

Graham, J. W., Hofer, S. M. & Piccinin, A. M., 1994. Analysis with missing data in drug prevention research.. *NIDA research monograph,* Volume 142, pp. 13-13.

Guo, X. et al., 2008. On the Class Imbalance Problem. *2008 Fourth International Conference on Natural Computation,* pp. 192-201.

Han, J., Pei, J. & Tong, H., 2022. *Data mining: concepts and techniques.* 3rd ed. s.l.:Morgan Kaufmann.

Hanley, J. A. & McNeil, B. J., 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology,* 143(1).

Japkowicz, N., 2001. Concept-learning in the presence of between-class and within-class imbalances.. *Springer, Berlin*, pp. 67-77.

Japkowicz, N., 2001. Supervised versus unsupervised binary learning by feed forward neural networks.. *Machine Learning,* Volume 42, pp. 97-122.

Japkowicz, N. & Stephen, S., 2002. The class imbalance problem: A systematic study.. *Intelligent data analysis,* 6(5), pp. 429-449.

Kamei, Y. et al., 2007. The Effects of Over and Under Sampling on Fault-prone Module Detection. *IEEE,* pp. 196-204.

Kerdprasop, N. & Kerdprasop, K., March 2012. On the Generation of Accurate Predictive Model from Highly Imbalanced Data with Heuristics and Replication Techniques. *International Journal of Bio-Science and Bio-Technology,* 4(1), pp. 49-64.

Koren, Y., Bell, R. & Volinsky, C., Aug. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE,* 42(8), pp. 30-37.

Kotsiantis, S., Kanellopoulos, D. & Pintelas, P., 2006. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering,* Volume 30, pp. 25-36.

Kotsiantis, S. & Pintelas, P. E., 2004. Mixture of Expert Agents for HandlingImbalanced Data Sets. *Annals of Mathematics, Computing & Teleinformatics,* Volume 1, pp. 46-55.

Kyureghian, G., Capps, O. & Nayga, R. M., 2011. A missing variable imputation methodology with an empirical application.. *Missing data methods: Cross-sectional methods and applications.*.

Ling, C. X. & Li, C., 1998. Data Mining for Direct Marketing Problems and Solutions.. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data.*

Malarvizhi, R. & Thanamani, D. A. S., 2012. K-NN Classifier Performs Better Than K-Means Clustering in Missing Value Imputation. *IOSR Journal of Computer Engineering,* 6(5), pp. 12-15.

Manevitz, L. M. & Yousef, M., 2001. One-class SVMs for document classification.. *Journal of Machine Learning Research,* Volume 2, pp. 139-154.

Mazumder, R., Hastie, T. & Tibshirani, R., 2010. Spectral Regularization Algorithms for Learning Large Incomplete. *Journal of Machine Learning Research,* Volume 11, pp. 2287-2322.

Mishra, S. & Khare, D., 2014. On comparative performance of multiple imputation methods for moderate to large proportions of missing data in clinical trials: a simulation study.. *Journal of Medical Statistics and Informatics,* Volume 2.

Monard, M. C. & Batista, G. E. A. P. A., 2003. An analysis of four missing data treatment methods for supervised learning. pp. 519-533.

Nguyen, D. V., Wang, N. & Carroll, R. J., 2004. Evaluation of missing value estimation for microarray data. Journal of Data Science. *Journal of Data Science,* 2(4), pp. 347-370.

Pal, M. & Mather, P. M., 2003. An assessment of the effectiveness of decision tree methods for land cover classification. *Remote Sensing of Environment,* 86(4), pp. 554-565.

Pazzani, M. et al., 1994. Reducing Misclassification Costs. *Machine Learning Proceedings,* pp. 217-225.

Penone, C. et al., 2014. Imputation of missing data in life-history trait datasets: which approach performs the best?. *Methods in Ecology and Evolution,* Volume 5, pp. 961-970.

Poulos, J. & Valle, R., 2018. Missing Data Imputation for Supervised Learning. *Applied Artificial Intelligence,* 32(2), pp. 186-196.

Quinlan, J. R., 2014. *C4.5: Programs for Machine Learning*.. San Mateo: Morgan Kaufmann.: Elsevier.

Ramentol, E. et al., 2012. SMOTE-FRST: A NEW RESAMPLING METHOD USING FUZZY ROUGH SET THEORY. *Proc: WSPC.*

Satuluri, N. & Kuppa, M. R., September 2012. A Novel Class Imbalance Learning Using Intelligent Under-Sampling. *International Journal of Database Theory and Application*, 5(3), pp. 25-35.

Schafer, J. W., 1997. *Analysis of incomplete multivariate data*.. s.l.:CRC.

Schelter, S. B. F. J. T. S. D. S. S. a. S. G., 2018a. On Challenges in Machine Learning Model Management. *IEEE Data*, pp. 5-15.

Schmitt, P., Mandel, J. & Guedj, M., 2015. A comparison of six methods for missing data imputation.. *Journal of Biometrics & Biostatistics*, 6(1), pp. 1-6.

Shang, C. et al., 2017. VIGAN: Missing view imputation with generative adversarial networks. *IEEE International Conference on Big Data (Big Data)*, pp. 766-775.

Stekhovan, D. J. & Bühlmann, P., January 2012. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1), pp. 112-118.

Sun, Y., Kamel, M. S., Wang, A. K. C. & Wang, Y., 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12), pp. 3358-3378.

Ting, K. M., May-June 2002. An instance-weighting method to induce cost-sensitive trees.. *IEEE Transactions on Knowledge and Data Engineering*, 14(3), pp. 659-665.

Troyanskaya, O. et al., June 2001. Missing Value Estimation Methods for DNa Microarrays. *Bioinformatics*, 17(6), pp. 520-525.

Tsikriktsis, N., 2005. A review of techniques for treating missing data in OM survey research. *Journal of Operations Management*, 24(1), pp. 53-62.

Tutz, G. & Ramzan, S., 2015. Improved methods for the imputation of missing data by nearest neighbor methods. *Computational Statistics & Data Analysis*, Volume 90, pp. 84-89.

Weiss, G. M. & Provost, F., 2003. Learning when training data are costly: The effect of class distribution on tree induction.. *Journal of artificial intelligence research*, Volume 19, pp. 315-354.

Yang, K., Huang, B., Stoyanovich, J. & Schelter, S., 2020. Fairness-Aware Instrumentation of Preprocessing~Pipelines for Machine Learning.

Yu, C.-Y., Chou, L.-C. & Chang, D. T.-H., 2010. Predicting protein-protein interactions in unbalanced data using the primary structure of proteins.. *BMC Bioinformatics*, Volume 11.

1 https://www.researchgate.net/figure/Differences-between-undersampling-and-oversampling_fig1_341164819

2 https://rikunert.com/smote_explained

3 https://medium.com/analytics-vidhya/random-forest-classification-and-its-mathematical-implementation-1895a7bb743e

10. Appendices

All files for the project can be found on the git server of the University of Birmingham at the following URL:

https://git-teaching.cs.bham.ac.uk/mod-msc-proj-2021/dxp169

All code can be found in a Jupyter notebook named as project.ipynb.

All code in this file is my own work.

To run, Jupyter with Python 3.9 is needed with the following Python modules installed: numpy, pandas, matplotlib, seaborn, scikit-learn, imblearn.