Bankruptcy Prediction: Mining the Polish Bankruptcy Data

May 1, 2024

Aakash Kamuju
AI21BTECH11005

Lokesh Badisa
AI21BTECH11005

Abstract

Bankruptcy forecasting involves predicting bankruptcy and various indicators of financial distress in companies. This constitutes a substantial domain within finance and accounting research. The significance of this field stems partly from its importance to creditors and investors, who need to assess the likelihood of a company going bankrupt. The objective of forecasting financial distress is to devise a predictive model that amalgamates diverse econometric parameters, enabling the anticipation of a company's financial health. Within this realm, numerous methodologies have been proposed, ranging from statistical hypothesis testing and modeling (e.g., generalized linear models) to more recent approaches like artificial intelligence (e.g., neural networks, Support Vector Machines, decision trees). In this document, we present our findings as we explore, construct, and contrast several widely utilized classification models relevant to bankruptcy prediction: Extreme Gradient Boosting for Decision Trees, Random Forests, Naïve Bayes, Balanced Bagging, and Logistic Regression. Our analysis is based on a dataset of bankruptcies among Polish companies, where synthetic features have been incorporated to capture higher-order statistics. A synthetic feature is formed through the combination of econometric measures using arithmetic operations such as addition, subtraction, multiplication, and division. Our process commences with data preprocessing and exploratory analysis, involving the utilization of popular data imputation techniques such as Mean, k-Nearest Neighbors, Expectation-Maximization, and Multivariate Imputation by Chained Equations (MICE) to handle missing data. To tackle the issue of data imbalance, we employ the Synthetic Minority Oversampling Technique (SMOTE) to oversample the minority class labels. Subsequently, we proceed to model the data using K-Fold Cross Validation on the aforementioned models, as well as on the imputed and resampled datasets. Finally, we assess and evaluate the performance of the models on validation datasets using various metrics such as accuracy, precision, and recall, thereby ranking the models accordingly. In conclusion, we discuss the encountered challenges and propose avenues for enhancing prediction accuracy, along with potential directions for future research.

1 Introduction

Enterprise bankruptcy prediction holds significant weight in economic decision-making processes. The financial state of businesses, whether small or large, impacts not only the local community and industry stakeholders but also investors, policymakers, and the global economy. Consequently, the considerable social and economic ramifications of corporate bankruptcies have spurred researchers to delve deeper into understanding bankruptcy causes and improving distress prediction. The volume of research in this domain is closely tied to data availability. For publicly traded firms, both bankrupt and solvent, a plethora of accounting ratios indicative of financial risk can be computed, alongside numerous other potential explanatory variables. Thus, this area lends itself well to the testing of increasingly sophisticated, data-driven forecasting methodologies. The evolution of bankruptcy prediction methods traces back to the application of various statistical tools, with a gradual refinement over time as researchers became more cognizant of inherent pitfalls in early analyses. Notably, many published studies have fallen prey to pitfalls that have long been recognized. Bankruptcy prediction has been under formal scrutiny since at least 1932, with FitzPatrick's seminal study comparing pairs of firms, one failed and one surviving, matched by date, size, and industry. While lacking the statistical rigor common today, FitzPatrick's interpretation of ratios and their trends effectively constituted a rudimentary form of multivariable analysis. The primary objective of bankruptcy prediction is to evaluate a company's financial health and its future viability in the market over the long term. It encompasses a broad spectrum of finance and econometrics, amalgamating expert insights into the phenomenon

with historical data from both successful and failed enterprises. Typically, businesses are characterized by numerous indicators reflecting their financial state, which are then utilized to construct mathematical models based on past observations. Several challenges are associated with bankruptcy prediction, with two prominent issues standing out. Firstly, domain experts may propose various econometric indicators describing a firm's condition, but the optimal method of combining them into an effective model remains ambiguous. Secondly, historical observations used for model training often suffer from imbalanced data, as successful companies significantly outnumber bankrupt ones. Consequently, trained models tend to disproportionately predict success (the majority class), even when some companies are distressed firms. These issues substantially impact the predictive efficacy of the model. Regarding contemporary approaches to bankruptcy prediction, survival methods have gained traction, along with option valuation approaches incorporating stock price volatility. Structural models define a default event for a firm when its assets depreciate sufficiently relative to its liabilities. Advanced techniques such as neural networks have been tested for bankruptcy prediction, while modern methods adopted by business information firms extend beyond annual accounts, incorporating current events such as company age, legal judgments, negative publicity, payment defaults, and creditor experiences.

2. Methodology

In the previous section, we formally introduced the problem statement of bankruptcy prediction. In this section, we explain our step-by-step solution of how we achieved benchmark results for bankruptcy prediction. Firstly, we introduce the Polish bankruptcy dataset and explain the details of the dataset like features, instances, data organization, etc. Next, we delve into data preprocessing steps, where we state the problems present with the data like missing data and data imbalance, and explain how we dealt with them. Next, we introduce the classification models we have considered and explain how we train our data using these models. Later, we analyze and evaluate the performance of these models using certain metrics like accuracy, precision and recall.

2.1 Data

The dataset we have considered for addressing the bankruptcy prediction problem is the Polish bankruptcy data, hosted by the University of California Irvine (UCI) Machine Learning Repository - a huge repository of freely accessible datasets for research and learning purposes intended for the Machine Learning/Data Science community. The dataset is about bankruptcy prediction of Polish companies. The data was collected from

Emerging Markets Information Service (EMIS), which is a database containing information on emerging markets around the world. The bankrupt companies were analyzed in the period 2000-2012, while the still operating companies were evaluated from 2007 to 2013. The dataset is very apt for our research about bankruptcy prediction because it has highly useful econometric indicators as attributes (features) and comes with a huge number of samples of Polish companies that were analyzed in 5 different timeframes: Based on the collected data five classification cases were distinguished, that depends on the forecasting period:

1. **1**$^{\text{st}}$ year: The data contains financial rates from 1$^{\text{st}}$ year of the forecasting period and corresponding class label that indicates bankruptcy status after 5 years.
2. **2**$^{\text{nd}}$ year: The data contains financial rates from 2$^{\text{nd}}$ year of the forecasting period and corresponding class label that indicates bankruptcy status after 4 years.
3. **3**$^{\text{rd}}$ year: The data contains financial rates from 3$^{\text{rd}}$ year of the forecasting period and corresponding class label that indicates bankruptcy status after 3 years.
4. **4**$^{\text{th}}$ year: The data contains financial rates from 4$^{\text{th}}$ year of the forecasting period and corresponding class label that indicates bankruptcy status after 2 years.
5. **5**$^{\text{th}}$ year: The data contains financial rates from 5$^{\text{th}}$ year of the forecasting period and corresponding class label that indicates bankruptcy status after 1 years.

The dataset is summarized in Table 1 below.

Dataset characteristic	Multivariate			
	Data	Total Instances	Bankrupt instances	Non-bankrupt instances
Number of Instances	1^{st}	7027	271	6756
	2^{st} year	10173	400	9773
	3^{nd} year	10503	495	10008
	4^{rd} year	9792	515	9227
	5^{th} year	5910	410	5500
Feature characteristics	Real values			
Has Missing Values ?	Yes			
Associated tasks	Classification			
Date donated	$04-11-2016$			

Table 1: Summary of the Polish bankruptcy dataset.

ID	Description	ID	Description
X1	net profit / total assets	X33	operating expenses / short-term liabilities
X2	total liabilities / total assets	X34	operating expenses / total liabilities
X3	working capital / total assets	X35	profit on sales / total assets
X4	current assets / short-term liabilities	X36	total sales / total assets
X5	[(cash + short-term securities + receivables - short-term liabilities)/ (operating expenses - depreciation)] $*$ 365	X37	(current assets - inventories) / long-term liabilities
X6	retained earnings / total assets	X38	constant capital / total assets
X7	EBIT / total assets	X39	profit on sales / sales
X8	book value of equity / total liabilities	X40	(current assets - inventory - receivables) / short-term liabilities
X9	sales / total assets	X41	total liabilities /((profit on operating activities + depreciation) $*$ (12/365))
X10	equity / total assets	X42	profit on operating activities / sales
X11	(gross profit + extraordinary items + financial expenses) / total assets	X43	rotation receivables + inventory turnover in days
X12	gross profit / short-term liabilities	X44	(receivables $*$365)/ sales
X13	(gross profit + depreciation) / sales	X45	net profit / inventory
X14	(gross profit + interest) / total assets	X46	(current assets - inventory) / short-term liabilities
X15	(total liabilities * 365) / (gross profit + depreciation)	X47	(inventory $*$365)/ cost of products sold
X16	(gross profit + depreciation) / total liabilities	X48	EBITDA (profit on operating activities - depreciation) / total assets
X17	total assets / total liabilities	X49	EBITDA (profit on operating activities - depreciation) / sales
X18	gross profit / total assets	X50	current assets / total liabilities
X19	gross profit / sales	X51	short-term liabilities / total assets
X20	(inventory $*$365)/ sales	X52	(short-term liabilities * 365)/ cost of products sold)
X21	sales (n) / sales (n-1)	X53	equity / fixed assets
X22	profit on operating activities / total assets	X54	constant capital / fixed assets
X23	net profit / sales	X55	working capital
X24	gross profit (in 3 years) / total assets	X56	(sales - cost of products sold) / sales
X25	(equity - share capital) / total assets	X57	(current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X26	(net profit + depreciation) / total liabilities	X58	total costs / total sales
X27	profit on operating activities / financial expenses	X59	long-term liabilities / equity
X28	working capital / fixed assets	X60	sales / inventory
X29	logarithm of total assets	X61	sales / receivables
X30	(total liabilities - cash) / sales	X62	(short-term liabilities $*$365)/ sales
X31	(gross profit + interest) / sales		sales / short-term liabilities
X32	(current liabilities $*$365)/ cost of products sold	X64	sales / fixed assets

Table 2: Summary of feature in the Polish bankruptcy data

Table 1 shows the total number of features and instances in the dataset, and the number of samples in each class (bankrupt or not-bankrupt) of all the 5 datasets. The features are explained in Table 2 above. As shown in the table, there are 64 features labelled X1 through X64, and each feature is a synthetic feature. A synthetic feature is a combination of the econometric measures using arithmetic operations (addition, subtraction, multiplication, division). Each synthetic feature is as a single regression model that is developed in an evolutionary manner. The purpose of the synthetic features is to combine the econometric indicators proposed by the domain experts into

complex features. The synthetic features can be seen as hidden features extracted by the neural networks but the fashion they are extracted is different.

2.2 Dataset Quality Assessment

Now we move on to assessing the quality of the dataset. As we have mentioned earlier, the dataset suffers from missing values and data imbalance.

2.2.1 Missing Data

First, we look at some statistics of missing values. As an example, we plot of the nullity matrix for the 1st year dataset that explains the sparsity of 1st Year data. This plot shown in Figure 1 was achieved using the library missingno. The nullity matrix gives us a data-dense display which lets us visually pick out the missing data patterns in the dataset. We notice that the features **X21** and **X37** have the highest number of missing values.

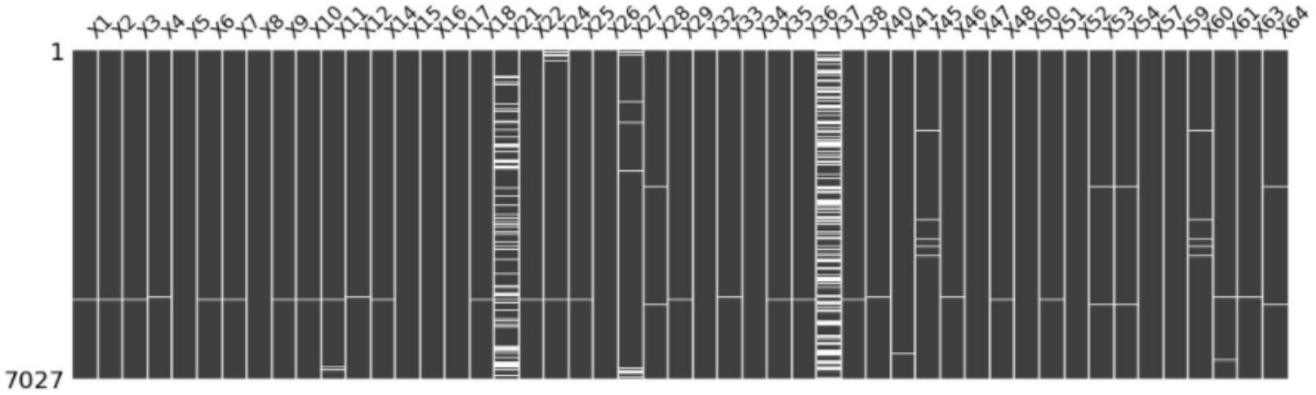

Figure 1: Sparsity matrix for dataset 'Year 1'. The white spaces indicate missing data values for the feature in the corresponding column.

Now, we explore to see the correlation among various features in the 1st Year data as an example. Shown in Figure 2 is a correlation heatmap for the 1st Year data that describes the degree of nullity relationship between the different features. The range of this nullity correlation is from -1 to 1($-1 \leq R \leq 1$). Features with no missing value are excluded in the heatmap. If the nullity correlation is very close to zero ($-0.05 < \mathrm{R} < 0.05$), no value will be displayed. Also, a perfect positive nullity correlation ($R = 1$) indicates when the first feature and the second feature both have corresponding missing values while a perfect negative nullity correlation ($R = -1$) means that one of the features is missing and the second is not missing.

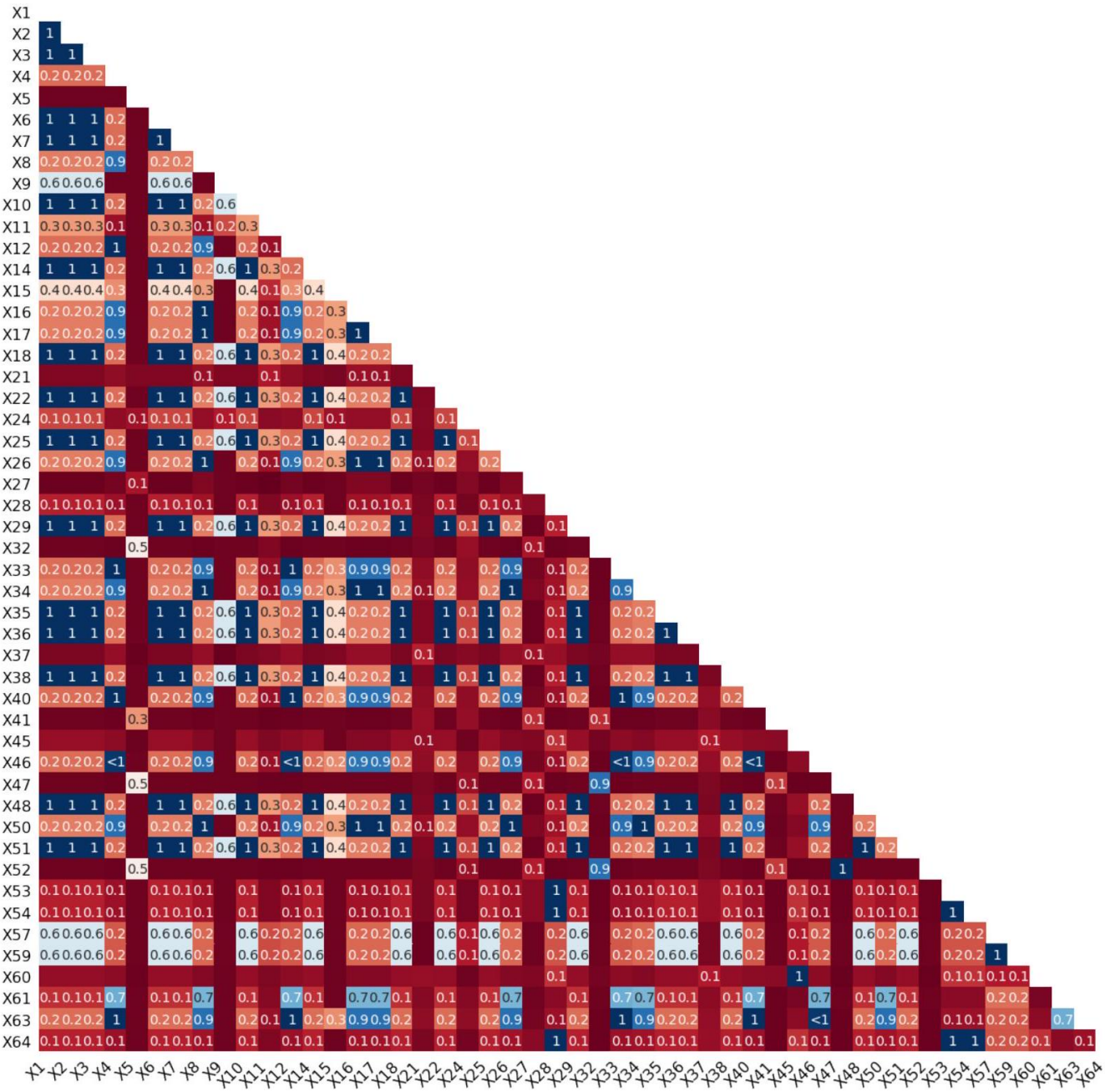

Figure 2: Feature correlation heatmap for the 1st Year dataset.

We have visually seen the sparsity in the data, as well as correlation among the features with respect to missing values. Now, let us see how much of data is actually missing. In Table 3 shown below, the second column shows the total number of instances in each dataset, and third column shows the number of instances or rows with missing values for at least one of the features. A naive approach of dealing with missing values would be to drop all such rows as in Listwise deletion. But dropping all such rows leads to a tremendous data loss. Column 4 shows the number of instances that would remain in each dataset if all rows with missing values were dropped. Column 5 shows the percent of data loss if all the rows with missing data values were indeed dropped. As the data loss in most of the datasets is over 50%, it is now clear that we cannot simply drop the rows with missing values, as it leads to severe loss in the representativeness of data.

Data Set	#Total Instances	# Instances with missing values	# Instances that would remain if all rows with missing values were dropped	% Data loss if rows with missing values were dropped
Year 1	7027	3833	3194	54.54%
Year 2	10173	6085	4088	59.81%
Year 3	10503	5618	4885	53.48%
Year 4	9792	5023	4769	51.29%
Year 5	5910	2879	3031	48.71%

Table 3: Assessing the Missing Data for all the datasets.

2.2.2 Data Imbalance

We have covered the missing data aspect of the data quality assessment, let us now see the Data Imbalance aspect. Table 4 shown below summarizes the populations of class labels in each dataset. Column 2 shows the total instances, while Column 3 and Column 4 show the number of instances with class label as Bankrupt and Non-Bankrupt respectively. Looking at the numbers of Bankrupt class label, we can figure out that they are a minority when compare with the non-bankrupt class label. But Column 5 clearly shows the population percentage of the minority class, i.e., the Bankruptcy class label, among the total population of the dataset. These numbers in column 5 tell us that there is a huge data imbalance. If this imbalance is not cured, in the modeling stage that follows, the models will not have seen enough data from the minority class label and they train and hence perform poorly.

2.3 Dealing with Missing Data

Missing data causes 3 problems:

1. Missing data can introduce a substantial amount of bias.
2. Makes the handling and analysis of the data more difficult.
3. Create reductions in efficiency.

Data Set	# Total Instances	# Bankrupt instances in this forecasting period	# Non-Bankrupt instances in this forecasting period	Percentage of minority class samples
Year 1	7027	271	6756	3.85%
Year 2	10173	400	9773	3.93%
Year 3	10503	495	10008	4.71%
Year 4	9792	515	9277	5.25%
Year 5	5910	410	5500	6.93%

Table 4: Assessing the Data Imbalance for all the datasets.

Dropping all the rows with missing values or Listwise deletion, introduces bias and affects representativeness of the results. The only viable alternative to Listwise deletion of missing data is Imputation. Imputation is the process of replacing missing data with substituted values and it preserves all the cases by replacing missing data with an estimated value, based on other available information. In our project we explored 4 techniques of imputation, and we will see them in the subsequent sections.

1. Mean Imputation
2. k-Nearest Neighbors Imputation
3. Expectation-Maximization Imputation
4. Multivariate Imputation Using Chained Equations

2.3.1 Mean Imputation

Mean imputation involves replacing any missing value within the dataset with the mean of the corresponding variable. In our dataset, missing values for a feature were substituted with the mean of the remaining non-missing values for that feature. This method mitigates correlations involving the imputed variable(s) since there is no relationship between the imputed variable and any other measured variables. While mean imputation exhibits favorable properties for univariate analysis, it poses challenges for multivariate analysis. As a result, we selected Mean Imputation as a baseline method. The implementation of mean imputation was carried out utilizing scikit-learn's Imputer class.

2.3.2 k-Nearest Neighbors Imputation

The k-nearest neighbors algorithm, or k-NN, is a non-parametric technique employed for both classification and regression tasks. In either scenario, the algorithm selects the k nearest training examples in the feature space as input. Additionally, k-NN can serve as a data imputation method, where it replaces NaN (missing) values in the dataset with values from the nearest-neighbor row or column, depending on the specific requirement. The nearest neighbor, whether row or column, is determined based on Euclidean distance. In cases where the corresponding value from the nearest neighbor is also NaN, the next nearest neighbor is considered. For our implementation, we utilized the fancyimpute library to execute k-NN data imputation, employing 100 nearest neighbors for the process.

2.3.3 Expectation-Maximization Imputation

In statistics, the EM (expectation-maximization) algorithm is an iterative technique employed to derive maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models. This method is particularly useful when the model relies on unobserved latent variables. The EM iteration entails alternating between an expectation (E) step, where a function for the expectation of the log-likelihood is generated using the current parameter estimate, and a maximization (M) step, where parameters maximizing the expected log-likelihood obtained in the E step are computed. These parameter estimates are then utilized to ascertain the distribution of the latent variables in the subsequent E step. EM Imputation involves the process of imputing missing values utilizing the Expectation-Maximization algorithm. For quantitative variables, missing values are replaced with their expected values computed using EM. Typically, a Multivariate Gaussian distribution is assumed in practice. Overall, EM imputation tends to outperform mean imputation as it preserves relationships with other variables. For our implementation, we utilized the impyute library to execute EM Imputation.

2.3.4 Multivariate Imputation using Chained Equations

Multiple imputation using chained equations (MICE) is an imputation technique that generates multiple imputations instead of a single one. It is recognized as a fully conditional specification or sequential regression multiple imputation method, which has emerged as a prominent approach for handling missing data. By creating multiple imputations, MICE acknowledges the statistical uncertainty inherent in the imputed values. The chained equations method within MICE is highly adaptable, capable of accommodating variables of different types (e.g., continuous or binary) and addressing complexities such as bounds or survey skip patterns. This flexibility is particularly advantageous when dealing with extensive missing data. Since multiple imputation entails generating multiple predictions for each missing value, analyses conducted on multiply imputed data incorporate uncertainty from the imputations, resulting in more accurate standard errors. In the MICE procedure, a sequence of regression models is executed, wherein each variable with missing data is modeled conditional on the other variables in the dataset. Consequently, each variable can be modeled according to its distribution, with binary variables often modeled using logistic regression and continuous variables using linear regression. For our implementation, we utilized the fancyimpute library to perform MICE imputation.

2.4 Dealing with Data Imbalance

Moving on to the other shortcoming of the Polish bankruptcy dataset, we now explain how we dealt with the Data Imbalance. Data Imbalance can be treated with Oversampling and/or Undersampling. In data analysis, Oversampling and Undersampling are opposite and roughly equivalent techniques of dealing with Data Imbalance, where

they adjust the class distribution of a data set (i.e. the ratio between the different classes/categories represented). Oversampling is increasing the class distribution of the minority class label whereas Undersampling is decreasing the class distribution of the majority class label. In our project, we explored Synthetic Minority Oversampling Technique or SMOTE.

2.4.1 Synthetic Minority Oversampling Technique (SMOTE)

The Synthetic Minority Oversampling Technique (SMOTE) is a commonly employed method for oversampling data. To demonstrate its operation, let's consider a training dataset with s samples and f features in the feature space, assuming the features are continuous. For clarity, let's use the example of a dataset containing information about birds. The feature space for the minority class we wish to oversample could include attributes like beak length, wingspan, and weight. To perform oversampling with SMOTE, we select a sample from the dataset and identify its k nearest neighbors in the feature space. Then, to create a synthetic data point, we calculate the vector between one of these k neighbors and the current data point. Next, we multiply this vector by a random number x, where x is in the range of 0 to 1. Adding this scaled vector to the current data point results in the generation of a new synthetic data point. SMOTE offers a robust technique for addressing class imbalance in datasets, particularly when the minority class is underrepresented. This approach helps to enhance the representation of minority class instances, thereby improving the performance of machine learning models trained on imbalanced datasets. SMOTE implementation can be found in the imbalanced-learn library.

2.5 Data Modeling

In this section, we will look at the various classification models that we have considered for training on the Polish bankruptcy datasets to achieve the task of coming up with a predictive model that would predict the bankruptcy status of a given (unseen) company with an appreciable accuracy. We have considered the following 6 models:

1. Gaussian Naïve Bayes
2. Logistic Regression
3. Decision Tree
4. Random Forests
5. Extreme Gradient Boosting
6. Balanced Bagging

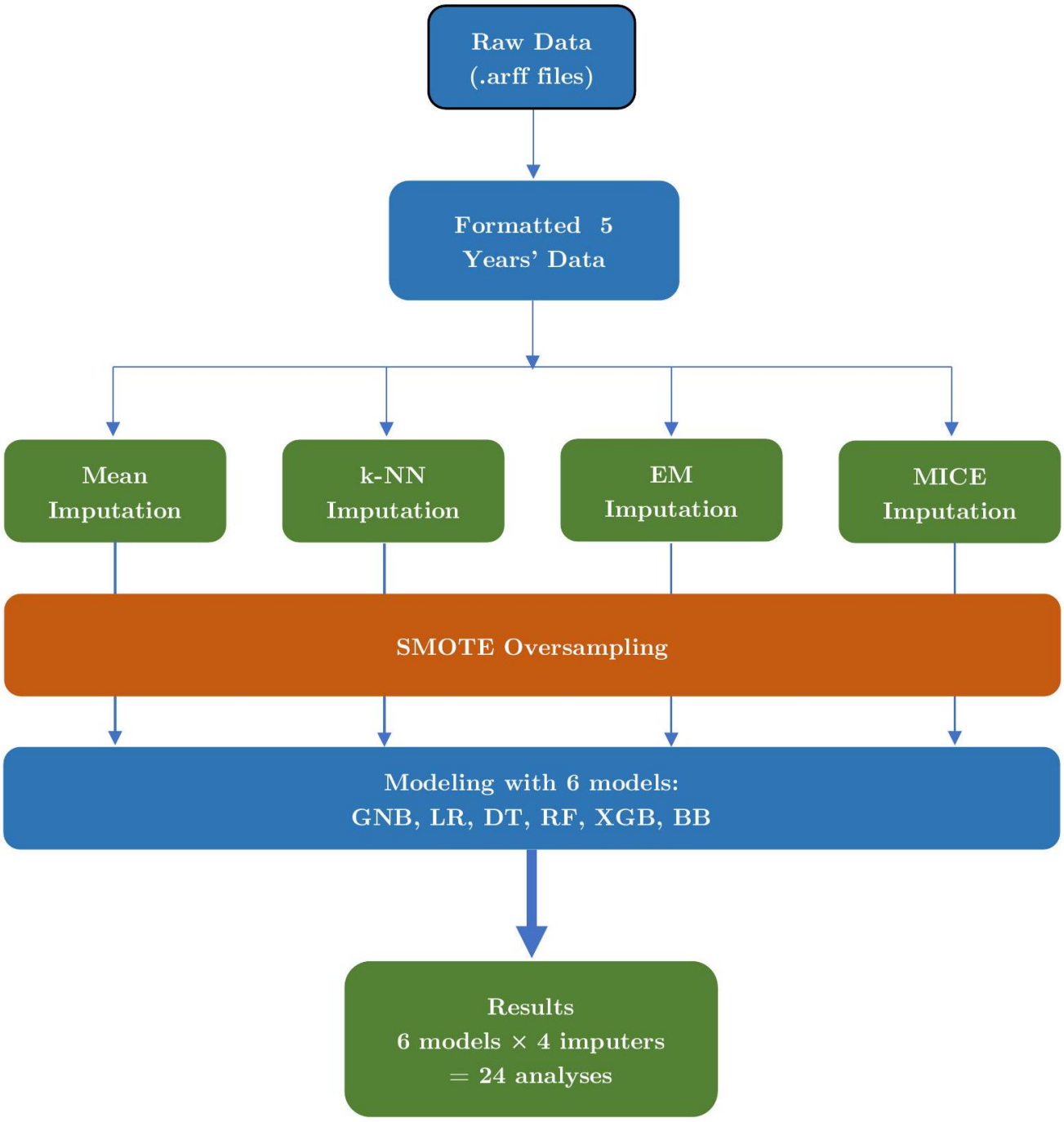

Figure 3: Pipeline for data modeling

Figure 3 shows the pipeline of data modeling for our project. After having obtained the formatted datasets from the raw data (.arff files), we have imputed the missing values via 4 different independent imputer methods (mean, k-NN, EM and MICE). Later, we have oversampled all these 4 imputed datasets with SMOTE oversampling technique and thus obtained 4 imputed and oversampled datasets. They datasets are ready for the data modeling step. We model each of these 4 datasets with the 6 models listed above. While modeling, we use the K-Fold Cross Validation technique for validation.

K-Fold Cross Validation

Since the Polish bankruptcy dataset lacks a distinct 'unlabeled' test dataset, it's evident that we must partition the training data to acquire a validation dataset (for each year's data). A simple split would yield only one validation dataset, and the inherent disparity in class label distributions between training and validation datasets would likely result in subpar model performance on both sets. Alternatively, K-Fold Cross Validation involves partitioning the training dataset into K bins. During each iteration (totaling K iterations), one bin is retained as the validation dataset while the remaining bins are utilized for training the model. Performance metrics such as accuracy, precision, and recall are recorded for each validation set. After all iterations, each bin will have served as the validation dataset at least once (depending on K). The metrics are then averaged across all iterations to yield final performance metrics. Consequently, at the conclusion of the modeling stage, we obtain 24 distinct results (6 models $\times$ 4 imputer datasets). In each subsequent subsection, we provide a brief overview of the model, elucidate the experiment by specifying the model's hyperparameters, and subsequently, in the Results section, we report the (Cross-Validation-Average) performance of the model on the validation data.

2.5.1 Gaussian Naïve Bayes Classifier

Naive Bayes classifier is one of the supervised learning algorithms which is based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P(y) P\left(x_1, \ldots, x_n \mid y\right)}{P\left(x_1, \ldots, x_n\right)}$$

Using the naive independence assumption that:

$$P\left(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n\right) = P\left(x_i \mid y\right)$$

for all i, this relationship is simplified to:

$$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)}{P\left(x_1, \ldots, x_n\right)}$$

Since $P\left(x_1, \ldots, x_n\right)$ is constant given the input, we can use the following classification rule:

$$P\left(y \mid x_1, \ldots, x_n\right) \propto P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)$$

$$\widehat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)$$

Gaussian Naïve Bayes implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P\left(x_i \mid y\right) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{\left(x_i - \mu_y\right)^2}{2\sigma_y^2}\right)$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

2.5.2 Logistic Regression Classifier

Logistic regression is a linear model for classification. It is also known as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

$$\text{LogitClassifier}(x) = \min_{w,c} \|w\|_1 + C \sum_{i=1}^{n} \left(\log\left(\exp\left(-y\left(X_i^T w + c\right)\right) + 1\right)\right)$$

Introduced L1 penalty (Lasso) into the model, which calculates LogitClassifier value for data points x_i as follows:

$$\text{LogitClassifier}(x) = \underset{\text{LogitClassifier}}{} (x) - \lambda J(\theta)$$

$$J(\theta) = \sum_{i=1}^{n} \|x_i\|$$

We implemented the model with $\lambda = 1$ and equal weights are given for all the features, using L1 regularization.

2.5.3 Decision Trees Classifier

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. For our classification task, we create a model that predicts the value of a target variable (y = will a firm go bankrupt?) by learning simple decision rules inferred from the data features ($x_1, x_2 \ldots . x_{64}$ - all the financial distress variables of a firm). While building decision tree, the data comes in records in the form:

$$(\text{x}, \text{Y}) = (x_1, x_2, \ldots\ldots\ldots, x_{64}, \text{Y})$$

Our model considers all features and gives equal weights to each of them while looking for best split during construction of a decision tree. We have considered 'Gini' index as a measure the quality of a split.

2.5.4 Random Forests Classifier

A random forest is a meta estimator that employs a collection of decision tree classifiers fitted on various sub-samples of the dataset, utilizing averaging to enhance predictive accuracy while mitigating overfitting. In a random forest, each tree within the ensemble is constructed from a sample drawn with replacement from the training set. Moreover, during the tree construction process, the split chosen for a node is not necessarily the best split among all features. Instead, it is selected from the best split among a random subset of the features. This inherent randomness in feature selection and sampling contributes to a slight increase in the forest's bias. However, the variance decreases due to averaging, often outweighing the increase in bias and resulting in an overall improved model. In our model, we utilize 5 estimators and adopt 'Entropy' as the criterion for evaluating the quality of a split.

2.5.5 Extreme Gradient Boosting Classifier

Extreme Gradient Boosting (XGBoost) is rooted in the principles of the gradient boosting framework, a powerful machine learning technique applied to regression and classification problems. Gradient boosting constructs a prediction model as an ensemble of weak prediction models, typically decision trees. It iteratively builds the model in a stage-wise fashion, similar to other boosting methods, while offering the flexibility to optimize an arbitrary differentiable loss function. XGBoost enhances the gradient boosting framework by incorporating a more regularized model formulation to combat overfitting, leading to superior performance. In our model, we utilize 100 estimators. Internally, XGBoost employs a log-linear classifier for model regularization, with a regularization parameter (λ) set to 1.

2.5.6 Balanced Bagging Classifier

Balanced Bagging is an application of the Bootstrap procedure to a high-variance machine learning algorithm, often decision trees, specifically tailored for classification tasks. Decision trees are known to be sensitive to the training data they are exposed to, resulting in potentially significant variations in the resulting tree structure and predictions when the training data changes. Balancing the dataset before training the classifier can enhance classification performance and mitigate the tendency of the ensemble to prioritize the majority class, a common limitation of decision tree classifiers. In our model, we utilize Random Forests as the base estimator for Balanced Bagging, with 5 estimators employed. We employ 'Entropy' as the criterion for evaluating the quality of a split. This approach aims to reduce the impact of class imbalance and improve overall model performance.

3. Results

Our results are organized as follow: Firstly, we report the accuracy score of the 6 models we have experimented with, using a plot of the accuracy score against each of the imputation method (Mean, k-NN, EM and MICE), and internally, on each of the 5 datasets (Year 1 - Year 5). Later, we also report the accuracy scores by years' datasets, i.e., plot of accuracy scores for each year's dataset, plotted against the 4 imputation techniques, and internally, the 6 models.

3.1 Accuracy plots of models

3.1.1 Gaussian Naïve Bayes (GNB)

The accuracy plot of Gaussian Naïve Bayes classifier model is shown in Figure 4. It was not surprising to us that the performance of GNB model was poor across all the imputation techniques. The overall accuracy scores remained below 55% and it only slightly better than a naïve random guess with a probability of 0.5 . The highest accuracy was obtained for 3rd year dataset under the MICE imputation. However, if we averaged the accuracy across all the years (1-5), the highest accuracy of 51.77% was output by EM imputation.

3.1.2 Logistic Regression (LR)

The accuracy plot of Logistic Regression classifier model is shown in Figure 5. This experiment was conducted using L1 regularization and the regularization parameter used was 1. We expected LR to perform better than GNB classifier model but it was quite surprising to see the overall (averaged) accuracy score of LR lower than GNB.

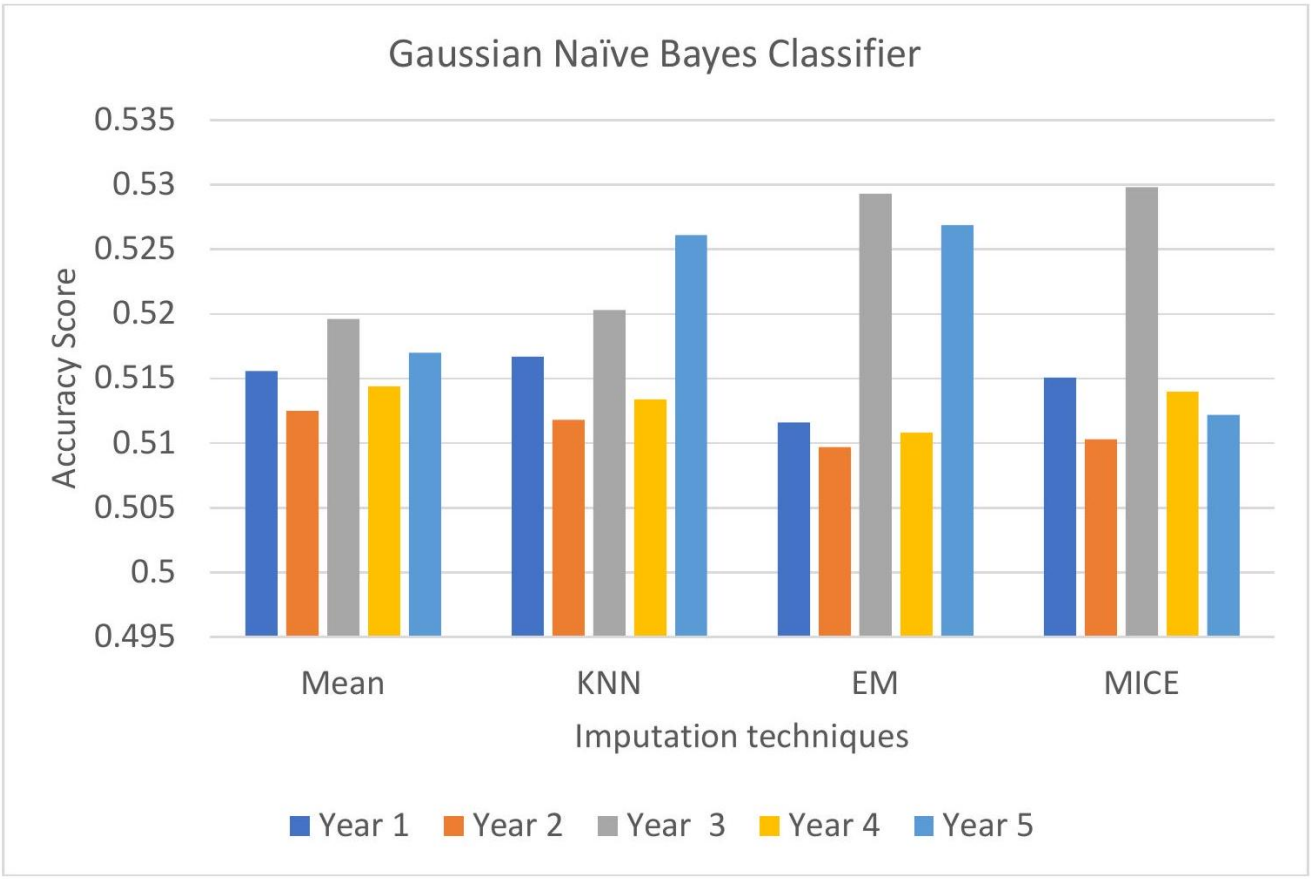

Figure 4: Accuracy evaluation of Gaussian Naïve Bayes classifier.

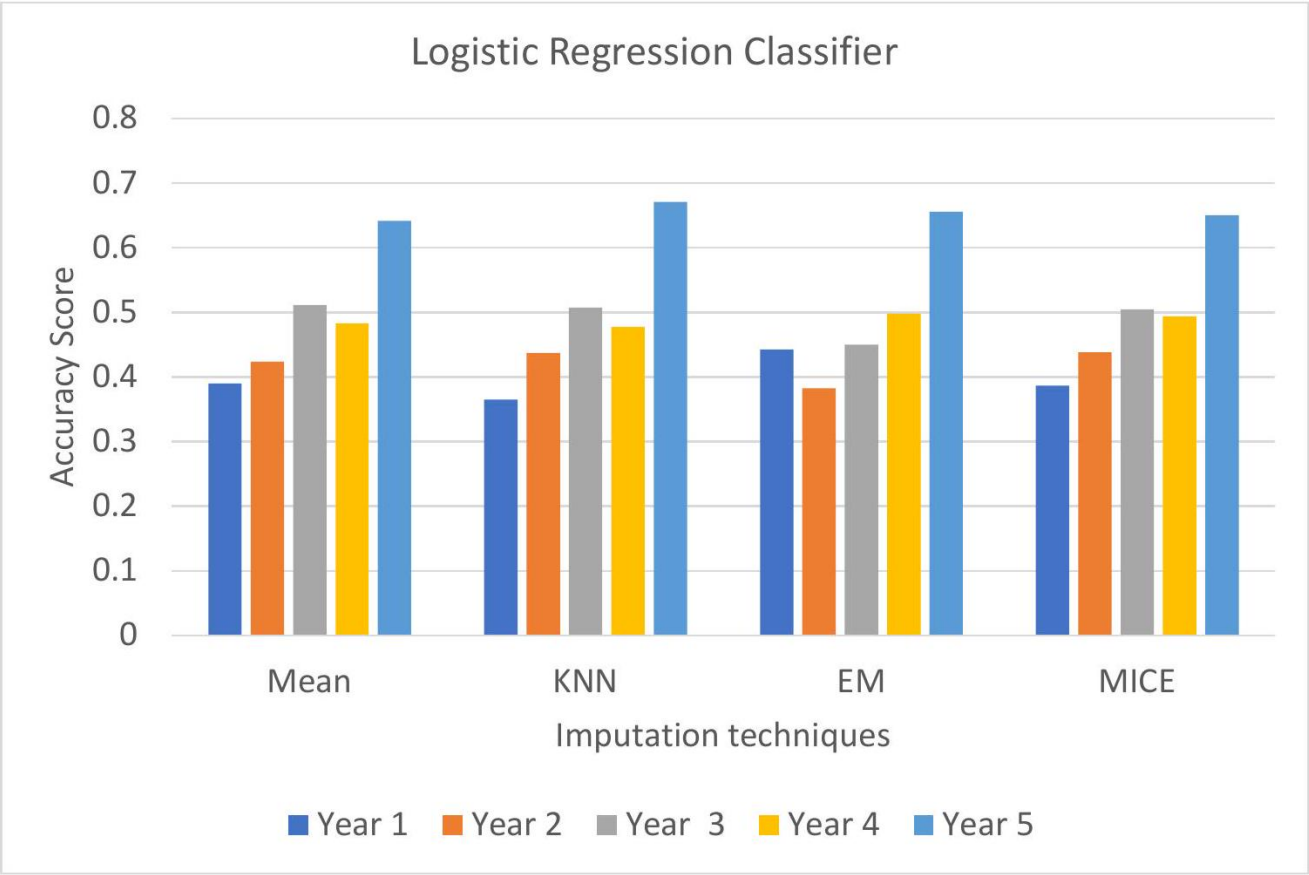

Figure 5: Accuracy evaluation of Logistic Regression classifier.

Although the Year 5 dataset for all the imputation techniques obtained an accuracy of $> 60\%$, the average accuracy of all the years is lower than the GNB, when compared with the corresponding imputation techniques.

3.1.3 Decision Tree (DT)

The accuracy plot of Decision Tree Classifier model is shown in Figure 6. Decision Tree classifier model performed slightly better than what we expected. It performed much better than the LR and GNB models we saw previously. The highest accuracy of 93.58% was obtained by the dataset Year 1 for Mean imputation technique. The highest average accuracy was also for the Mean imputed datasets. All the datasets under k-NN imputation performed poorly somehow, with an average accuracy of only 87.48%. Year 4 datasets consistently performed poorly for all the imputation methods.

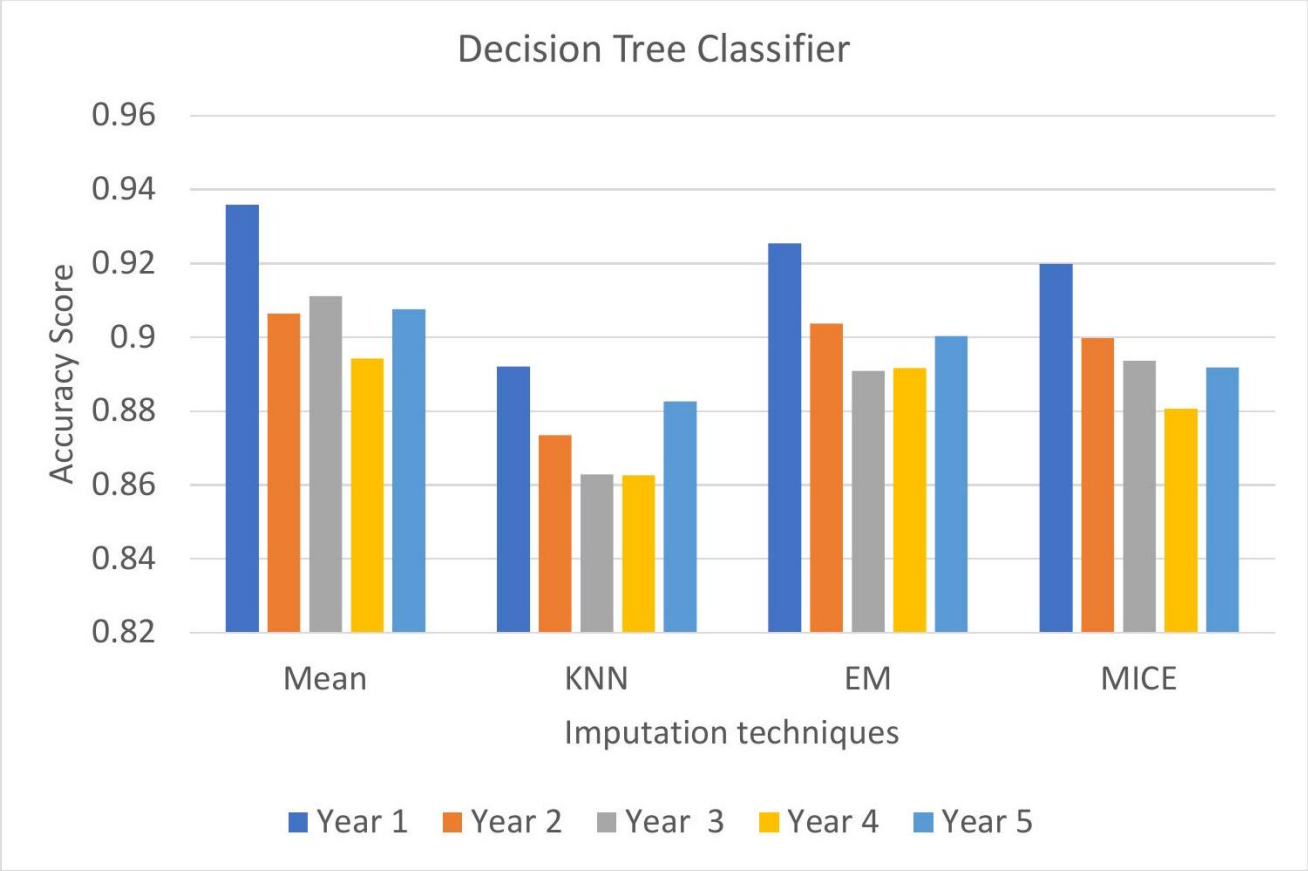

Figure 6: Accuracy evaluation of Decision Tree classifier.

3.1.4 Random Forests (RF)

The accuracy plot of Random Forest Classifier model is shown in Figure 7. As we expected, RF model performed better than the Decision Tree model. RF model obtained its highest accuracy of 92.89% for Mean imputation method, where Decision Tree model
only could obtain 91.10%. The highest difference between the performance of RF(90.28%) and DT (87.48%) models was noted for the k-NN imputation method.

Random Forest Classifier

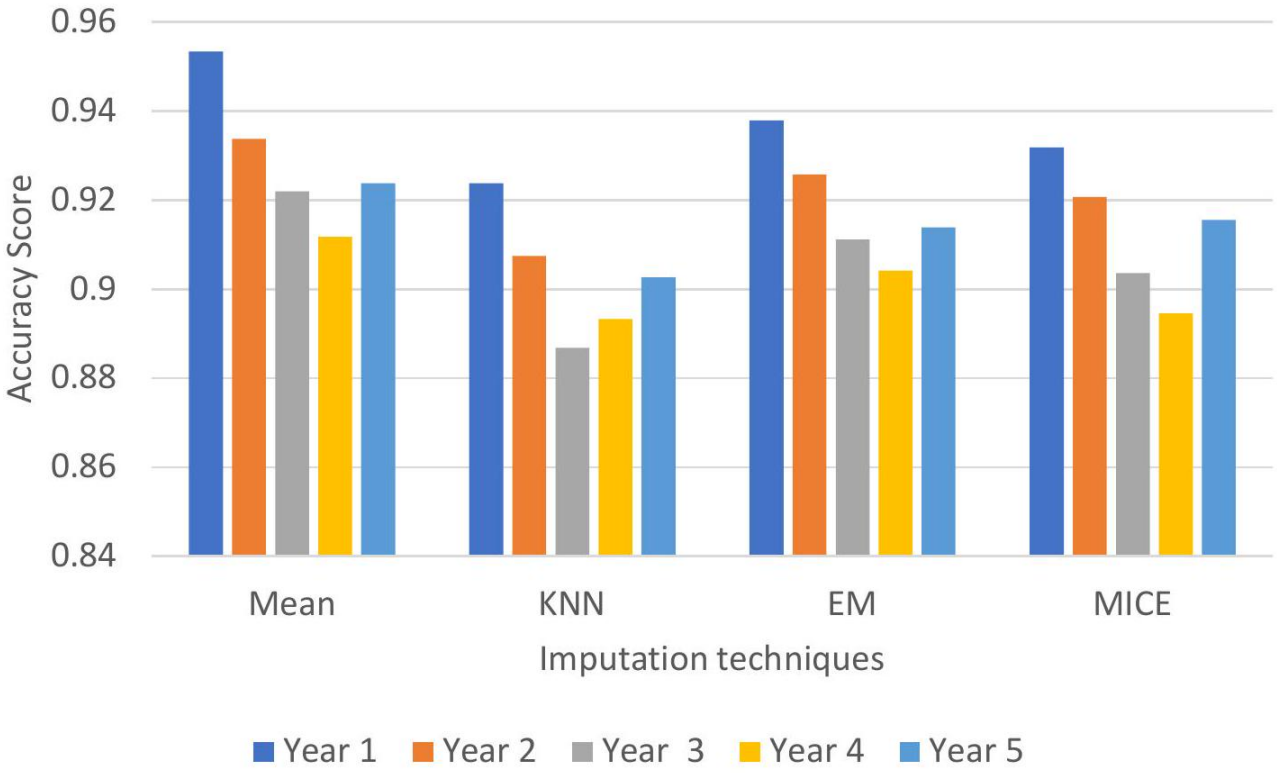

Figure 7: Accuracy evaluation of Random Forest classifier.

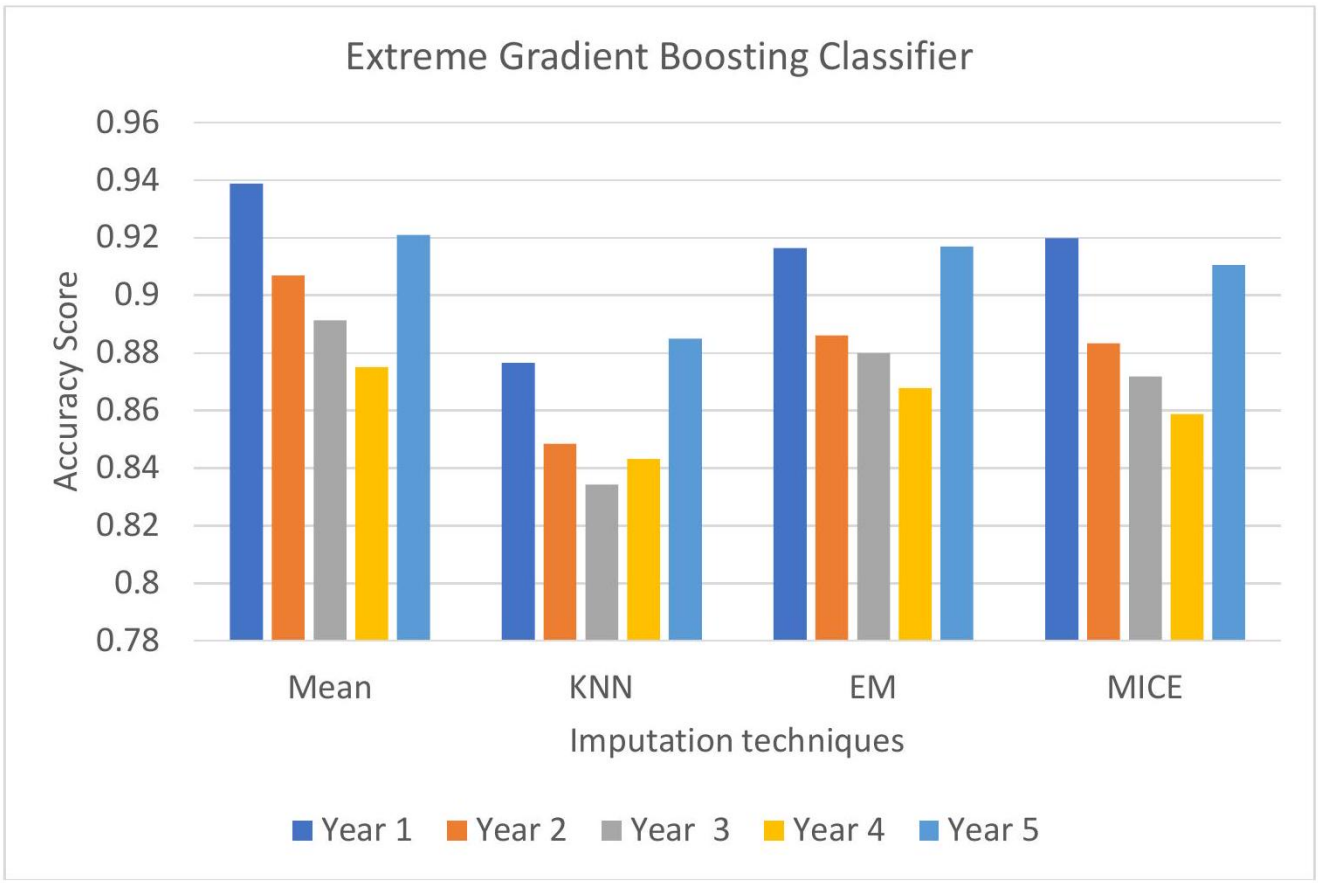

Figure 8: Accuracy evaluation of Extreme Gradient Boosting classifier.

3.1.5 Extreme Gradient Boosting (XGB)

The accuracy plot of Extreme Gradient Boosting Classifier model is shown in Figure 8 above. We expected XGB model would perform better than the Decision Tree and Random Forest methods, but it performed slightly worse than the Decision Tree model. The highest accuracy was obtained by the XGB model was for the Year 1 dataset and the Mean imputation method. All the datasets performed poorly for the k-NN imputation method.

3.1.6 Balanced Bagging (BB)

The accuracy plot of Balanced Bagging Classifier model is shown in Figure 9 below. Balanced Bagging was the best model we have experimented with so far. It has given the highest accuracy scores which were higher than all other models for each of the data imputation methods. The highest accuracy of 98.19% was obtained for the Year 1 dataset and Mean imputation method. The least accuracy of 94.2% was obtained for Year 5 dataset for k-NN imputation. Even this least accuracy score was better than the accuracy scores reported by the other models. The highest average accuracy across all years' datasets of 96.59% was obtained for the mean imputation method.

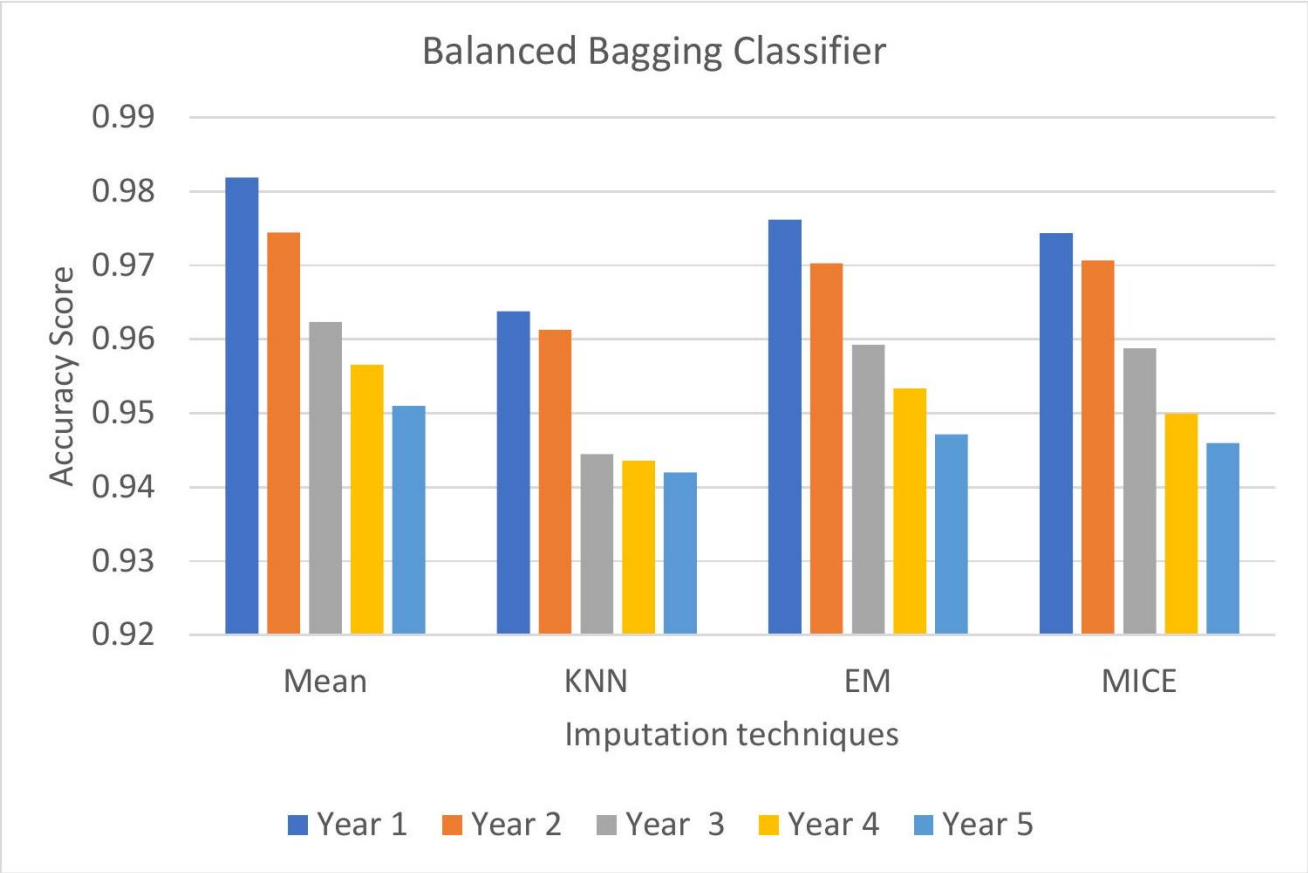

Figure 9: Accuracy evaluation of Balanced Bagging classifier.

3.2 Accuracy plots by Years' datasets

3.2.1 Year 1 dataset

The accuracy plot for the Year 1 dataset is shown in Figure 10. We observed that for Year 1, Balanced Bagging classifier model with Mean imputation methods proven to be the most successful once with an accuracy of 98.19%. In general, more or less, all the imputation techniques performed uniformly. The next best combination would be Random Forest model with Mean Imputation with an accuracy of 95.37%.

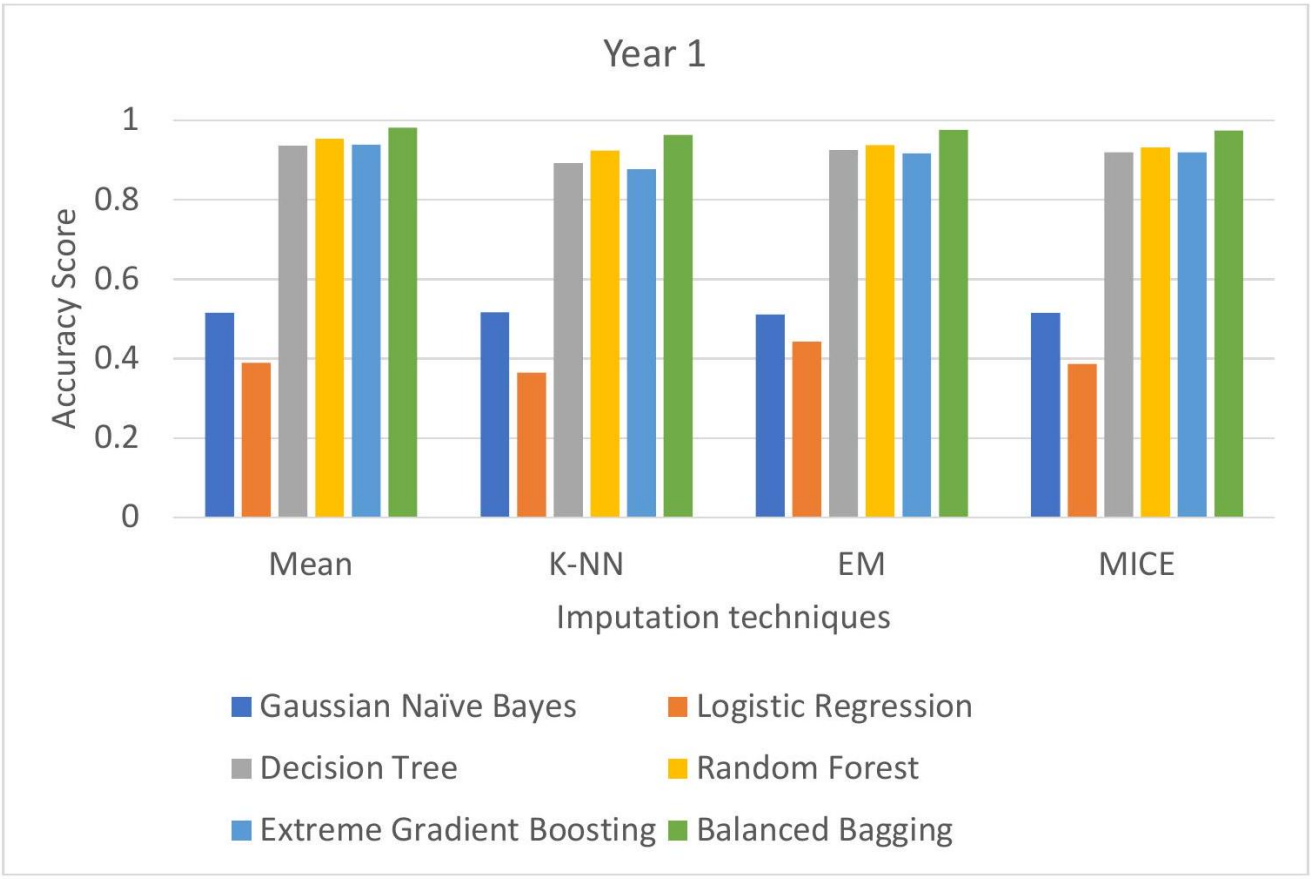

Figure 10: Accuracy evaluation of Year 1 dataset

3.2.2 Year 2 dataset

The accuracy plot for the Year 2 dataset is shown in Figure 11. Like the Year 1 dataset, Year 2 dataset also obtained its best performance for Balanced Bagging classifier model and Mean Imputation method with an accuracy of 97.45%. The next best model is Random Forest with Mean imputation method with an accuracy of 93.38%

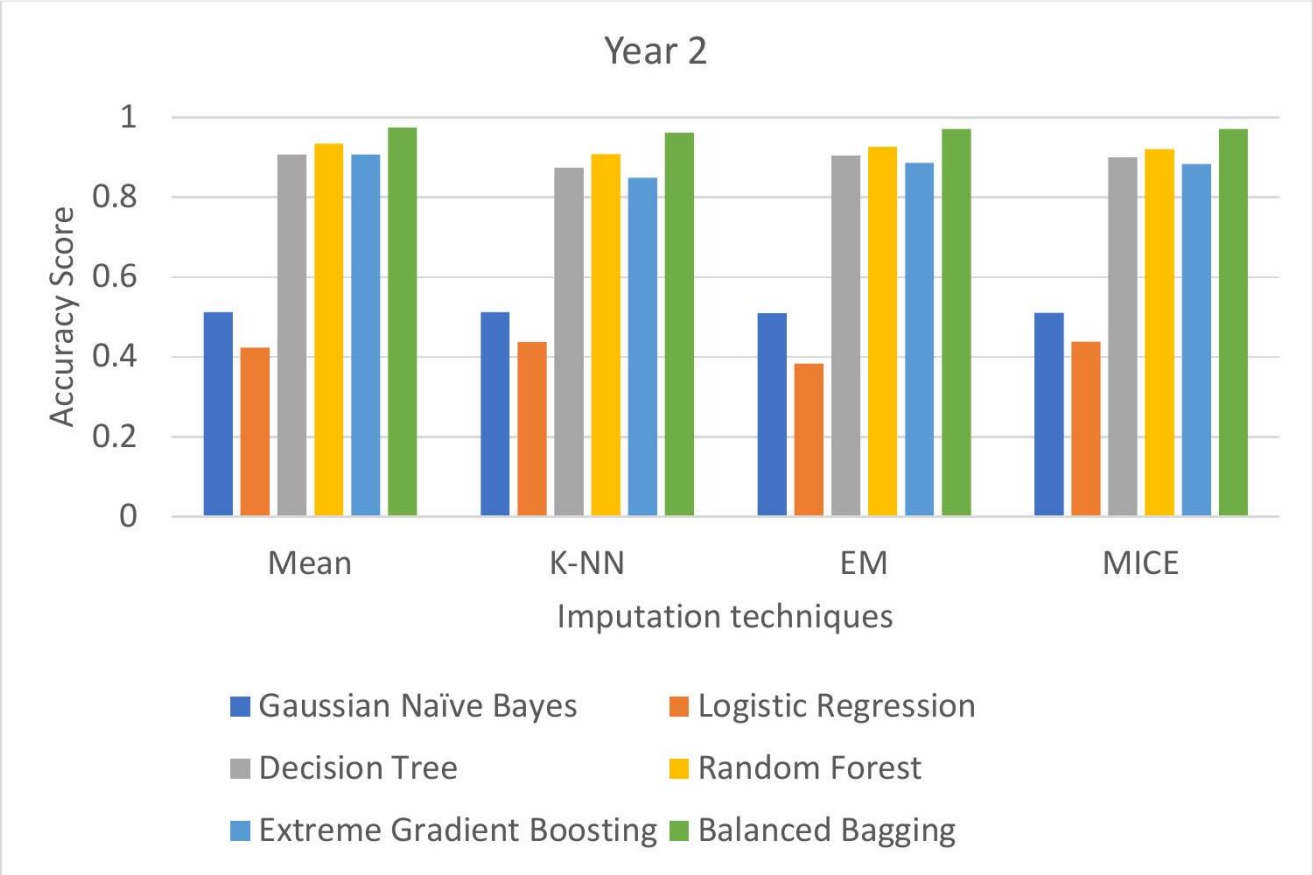

Figure 11: Accuracy evaluation of Year 2 dataset.

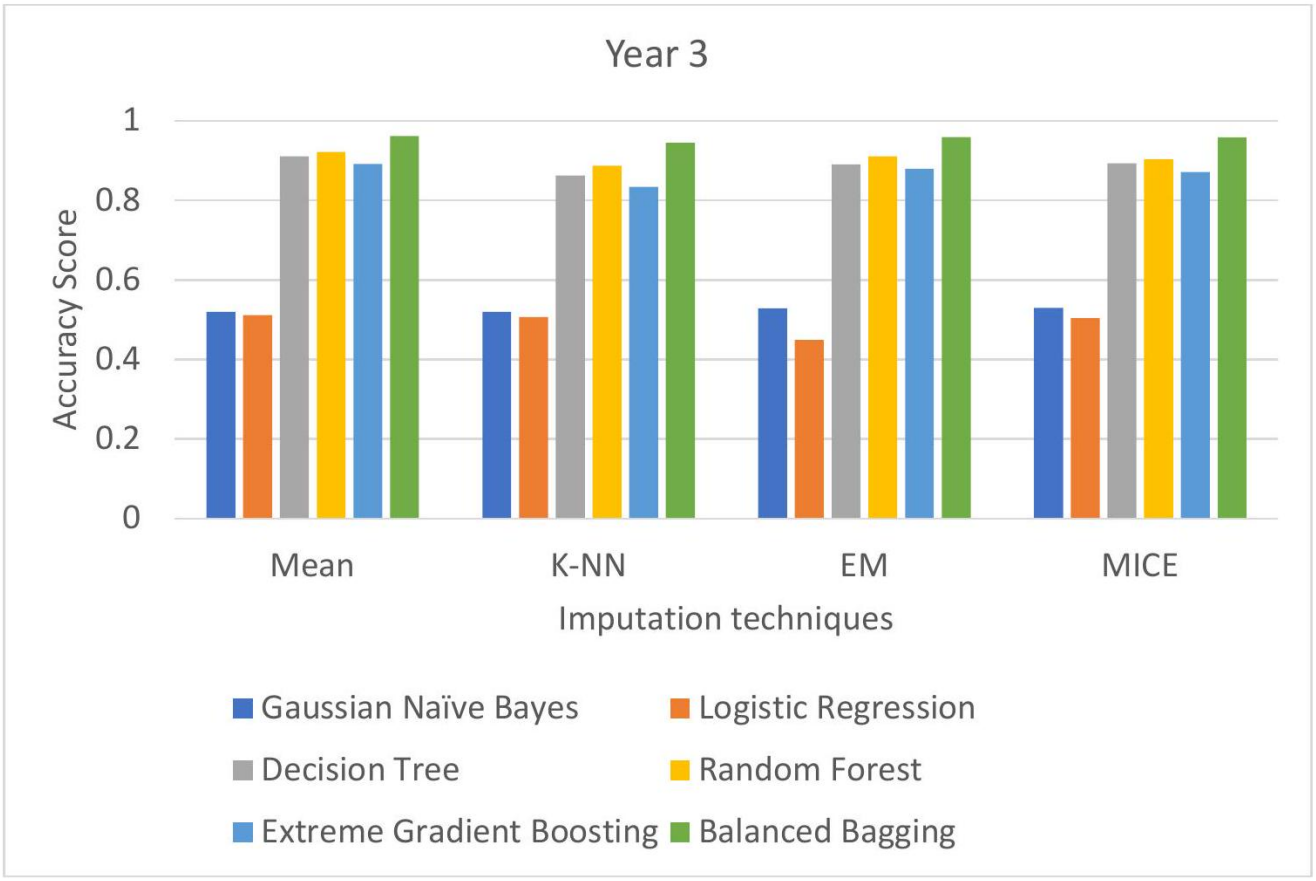

Figure 12: Accuracy evaluation of Year 3 dataset.

3.2.3 Year 3 dataset

The accuracy plot for the Year 3 dataset is shown in Figure 12. Like the Year 1 and Year 2 datasets, Year 3 dataset also obtained its best performance for Balanced Bagging classifier model and Mean Imputation method with an accuracy of 96.24%, closely followed by the EM imputation for the same model, with an accuracy of 95.93%. The next best model is Random Forest with Mean imputation method with an accuracy of 92.22%

3.2.4 Year 4 dataset

The accuracy plot for the Year 4 dataset is shown in Figure 13 below. Like the previous plots, Year 4 dataset also obtained its best performance for Balanced Bagging classifier model and Mean Imputation method with an accuracy of 95.66%, closely followed by the EM imputation for the same model, with an accuracy of 95.34%. The next best model is Random Forest with Mean imputation method with an accuracy of 91.18%.

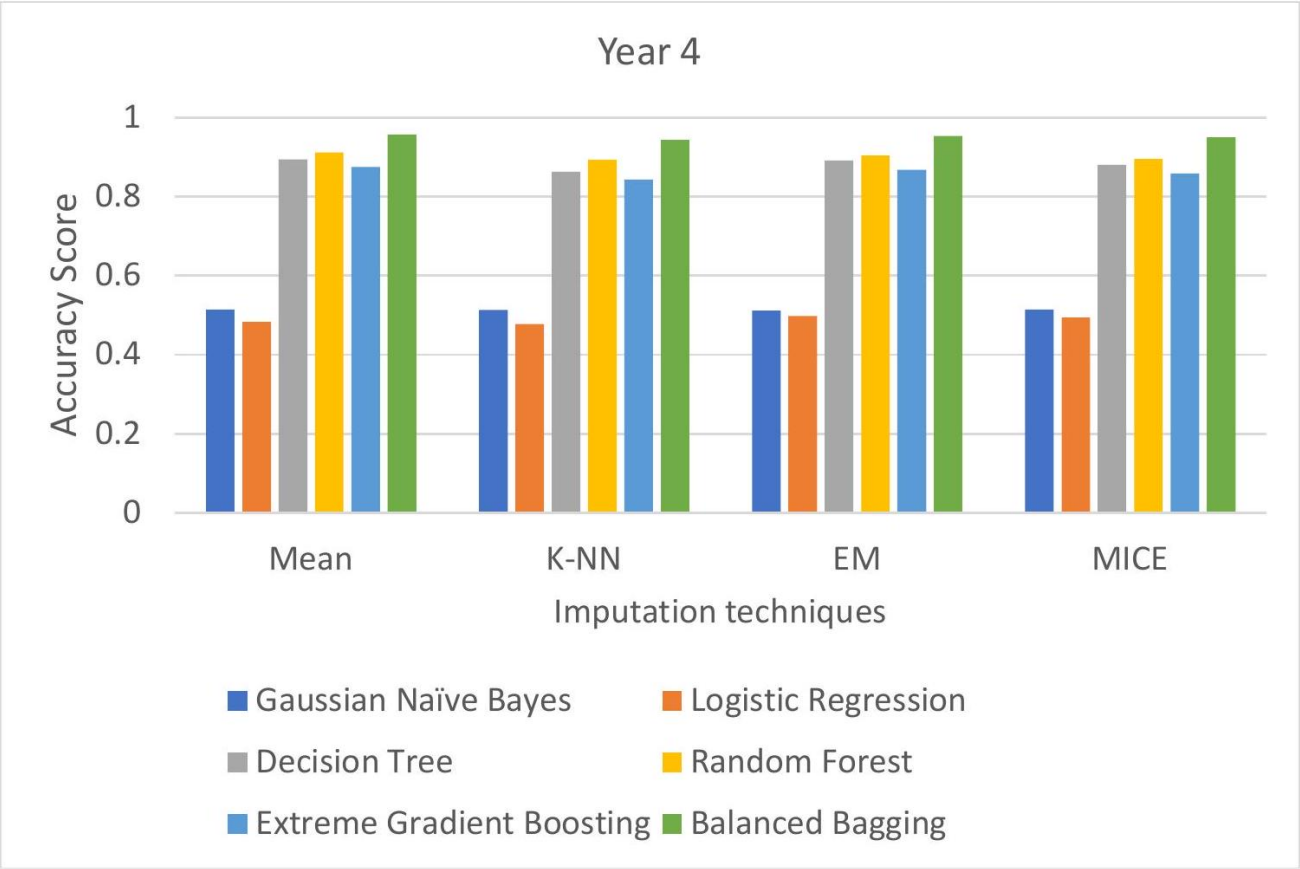

Figure 13: Accuracy evaluation of the Year 4 dataset.

3.2.5 Year 5 dataset

The accuracy plot for the Year 5 dataset is shown in Figure 14 below. Like the previous plots, the Year 5 dataset also obtained its best performance for the Balanced Bagging classifier model and Mean Imputation method with an accuracy of 95.10%, closely followed by the EM imputation for the same model, with an accuracy of 94.72%. The next best model is Extreme Gradient Boosting with Mean imputation method with an accuracy of 92.10%.

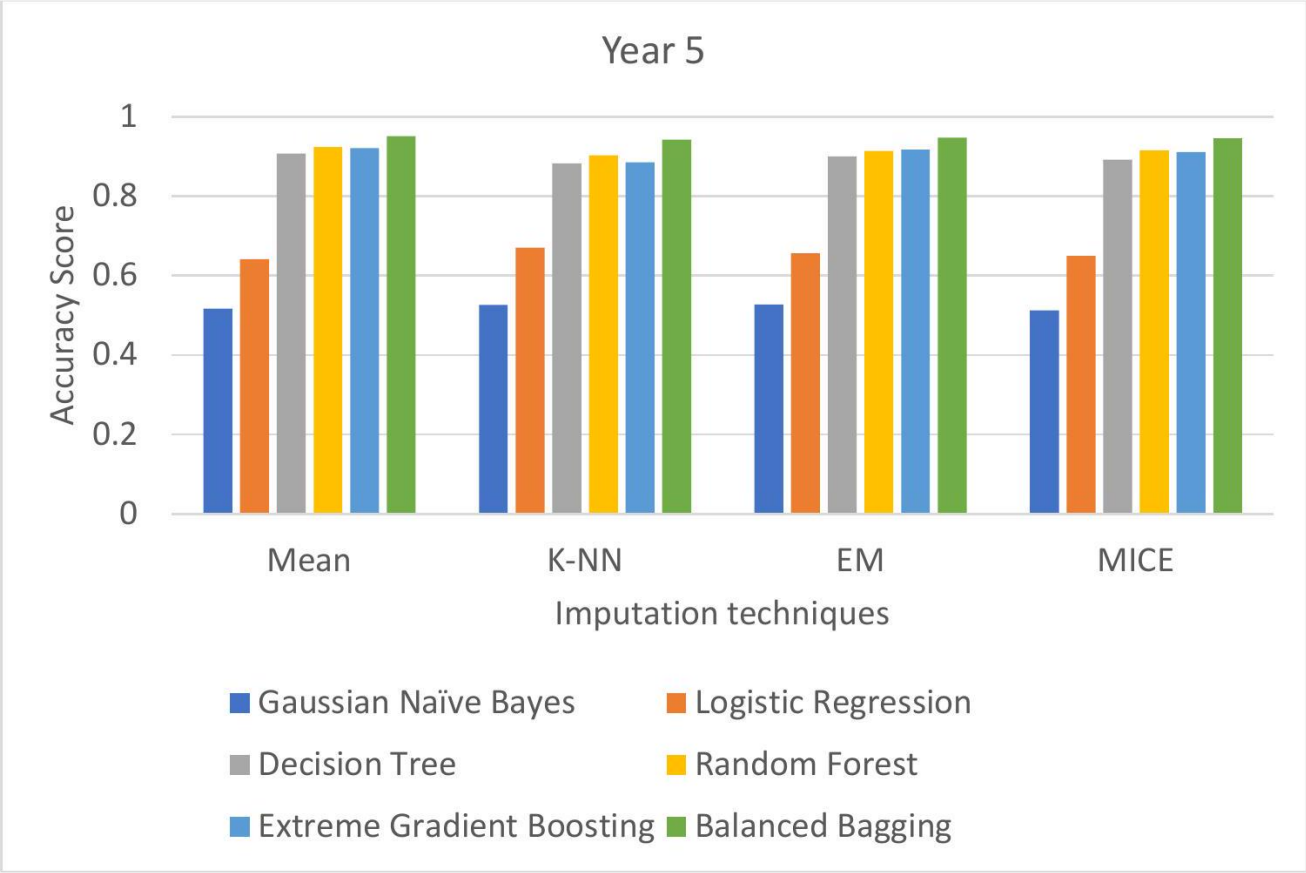

Figure 14: Accuracy evaluation of the Year 5 dataset.

	IMPUTATION TECHNIQUES				
		Mean	k-NN	EM	MICE
MO	Gaussian Naïve Bayes	51.58	51.77	51.77	51.63
	Logistic Regression	49.00	49.16	48.58	49.48
DELS	Decision Tree	91.10	87.48	90.24	89.72
	Random Forests	92.89	90.28	91.86	91.32
	Extreme Gradient Boosting	90.67	85.75	89.36	88.82
	Balanced Bagging	96.59	95.14	96.07	95.86

Table 6: Mean accuracies across all years' datasets for various models and imputation methods

4. Conclusion

In this section, we discuss the summary of the work we have done on this project thus far. We have successfully modeled 6 classification models: Gaussian Naïve Bayes, Logistic Regression, Decision Trees, Random Forests, Extreme Gradient Boosting, and Balanced Bagging classifiers. The training sets were made sure to have a balanced set of class labels by oversampling the minority class labels using the Synthetic Minority Oversampling technique. Also, we have imputed the missing values in the data using 4 imputer techniques: Mean, k-nearest Neighbors (k-NN), Expectation-Maximization (EM), and Multiple Imputation using Chained Equations (MICE). The biggest

challenge was to deal with the missing/sparse data. Since all the companies being evaluated for bankruptcy don't operate on the same timelines, it is difficult to gather meaningful data and organize it. The features on which the bankruptcy prediction is based are not as straightforward as the financial ratios found on the balance sheets of the companies and need to thoroughly be studied and validated. We have successfully documented our findings and suggested the best bankruptcy prediction model we have seen in our project.

To access the code, please visit: Bankruptcy Prediction