PuckTrick: A Library for Making Synthetic Data More Realistic

Alessandra Agostini[1], Andrea Maurino[1] and Blerina Spahiu[1]

[1]*Department of Informatics, Systems and Communication*
University of Milan -Bicocca
Viale Sarca 336, 20126 Milan, Italy

Abstract

The increasing reliance on machine learning (ML) models for decision-making requires high-quality training data. However, access to real-world datasets is often restricted due to privacy concerns, proprietary restrictions, and incomplete data availability. As a result, synthetic data generation (SDG) has emerged as a viable alternative, enabling the creation of artificial datasets that preserve the statistical properties of real data while ensuring privacy compliance. Despite its advantages, synthetic data is often overly clean and lacks real-world imperfections, such as missing values, noise, outliers, and misclassified labels, which can significantly impact model generalization and robustness. To address this limitation, we introduce Pucktrick, a Python library designed to systematically contaminate synthetic datasets by introducing controlled errors. The library supports multiple error types, including missing data, noisy values, outliers, label misclassification, duplication, and class imbalance, offering a structured approach to evaluating ML model resilience under real-world data imperfections. Pucktrick provides two contamination modes: one for injecting errors into clean datasets and another for further corrupting already contaminated datasets. Through extensive experiments on real-world financial datasets, we evaluate the impact of systematic data contamination on model performance. Our findings demonstrate that ML models trained on contaminated synthetic data outperform those trained on purely synthetic, error-free data, particularly for tree-based and linear models such as SVMs and Extra Trees.

1. Introduction

The rapid advancement of technology in recent years has facilitated the efficient and large-scale gathering, processing, and analysis of data for informed decision-making across various domains. As of 2024, approximately 402.74 million terabytes (or 0.4 zettabytes) of data are created each day, totaling around 147 zettabytes annually[1]. This daily data generation is projected to increase to about 181 zettabytes per year by 2025[2]. The rapid evolution of statistical analysis and pattern recognition techniques has revolutionized the ability to extract meaningful insights from such data. However, the effectiveness of these methods depends on the quality of the data being analyzed. When data is incomplete, inconsistent, duplicated, or lacks proper security measures, the accuracy and reliability of the results are significantly compromised.

One of the major challenges in machine learning is the availability of real dataset for training

SEBD 2025: 33st Symposium on Advanced Database Systems. June 16-19 Ischia, Campania, Italy

alessandra.agostini@unimib.it (A. Agostini); andrea.maurino@unimib.it (A. Maurino); Blerina.Spahiu@unimib.it (B. Spahiu)

[1] https://explodingtopics.com/blog/data-generated-per-day
[2] https://www.statista.com/statistics/871513/worldwide-data-created/

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-03-13

the model [1]. In many applied contexts, it is not always possible to use real data due to privacy concerns or the unwillingness of data owners to share their proprietary information. To address both data quality and privacy concerns, synthetic data has emerged as a trade-off solution [2]. Synthetic data replicates the statistical properties of real data while eliminating direct exposure of sensitive information. It also helps mitigate the issue of missing values by generating artificial replacements, balancing datasets by creating synthetic examples for underrepresented groups, and providing an added layer of security since the generated data does not correspond to real-world individuals or entities. Existing models focus on creating synthetic data by leveraging techniques such as generative adversarial networks (GANs)[3], variational autoencoders (VAEs)[4], and statistical sampling methods [5] to replicate the properties of real-world data distributions.

Once a synthetic dataset is generated, its quality is typically assessed by training a preliminary machine learning model and evaluating performance metrics such as accuracy, F1-score, precision, and recall. These metrics help determine whether the synthetic data preserves meaningful patterns from real-world data. By performing this initial assessment with a lightweight model or proxy task, researchers can estimate the dataset's utility before committing to the expensive and time-consuming process of full-scale model training [6]. However, this approach overlooks a critical aspect: real-world data is inherently noisy and imperfect. For example, a medical sensor may malfunction and provide incorrect measurements. A change in the IT system that feeds the dataset could result in missing data or values on a different scale than the original ones. Finally, experts might misclassify certain data points. Synthetic datasets, typically designed to be clean and well-structured, may fail to capture these real-world inconsistencies. As a result, models trained on idealized synthetic data risk poor generalization when applied to real-world scenarios, where unpredictable data variations and errors are common [7]. This gap highlights the importance of incorporating realistic noise and imperfections into synthetic data to improve its applicability in practical machine-learning applications.

To address this limitation, we developed a python library called Pucktrick designed to systematically and controllably contaminate datasets, thereby generating more realistic training data. The Pucktrick library simulates errors that can be encountered in a dataset used for training models. The most common types of errors encountered in real-world datasets for machine learning training include:

- *Missing Data*: Instances where values are absent, leading to biases and potential inaccuracies in predictions.
- *Noisy Data*: Random errors or fluctuations in the data that obscure underlying patterns and degrade model performance.
- *Outliers:* Data points that significantly deviate from the norm, potentially skewing model predictions.
- *Imbalanced Data*: Unequal representation of classes, leading to bias in favor of the majority class.
- *Duplicate Data*: Repeated or nearly identical entries that can lead to overfitting and inefficient training.
- *Incorrect Labels*: Mislabelled instances that negatively impact classification performance and model reliability.

The Pucktrick library offers two modes of operation. The first mode introduces errors into a dataset that is initially considered clean, allowing users to simulate real-world imperfections systematically. The second mode is designed for cases where a dataset is already contaminated but requires further corruption with specific types of errors. In such situations, the extended methods of Pucktrick can be used to incrementally introduce additional errors until the desired error percentage is reached. Beyond dataset contamination, Pucktrick also serves other purposes. For instance, researchers developing data-cleaning techniques can leverage Pucktrick to inject specific errors into a dataset and then assess whether their method effectively detects and corrects these errors. This makes Pucktrick a versatile tool for both generating realistic training data and benchmarking data-cleaning algorithms.

Experiments show that training machine learning models on synthetic data with controlled errors (introduced using Pucktrick) results in better accuracy compared to training on purely synthetic, error-free data. This confirms that exposing models to realistic imperfections during training makes them more robust and adaptable, ultimately improving their performance in real-world scenarios.

Given the importance of training machine learning models with realistic, imperfect data, in this study, we investigate the impact of controlled dataset contamination on machine learning model performance by analyzing:

- The effect of training on contaminated synthetic data versus purely synthetic, error-free data,
- The influence of different types of data errors on model generalization and real-world applicability,
- The extent to which systematic error introduction enhances a model's ability to handle naturally occurring data imperfections, and
- The potential of Pucktrick as a benchmarking tool for evaluating data-cleaning techniques and error-handling strategies in machine learning workflows.

The paper is organized as follows: section 2 reviews the state of the art in both synthetic data generation and data contamination. Section 3 introduces the Pucktrick library and its functionalities while section 4 outlines the experimental setup and discusses the preliminary results. Finally, section 5 provides the conclusions and explores potential directions for future work.

2. Related Work

This section explores the current state of the art in SDG techniques, approaches for introducing controlled data imperfections, and their implications for model robustness and real-world applicability.

2.1. Synthetic data generation algorithms

Synthetic data generation (SDG) involves creating artificial datasets that replicate the statistical properties of real-world data. SDG is relevant in various scenarios where real data cannot be used for training machine learning models. In the medical domain, for instance, utilizing real patient data is challenging due to confidentiality concerns. Similarly, in the financial sector, datasets for fraud detection are often highly imbalanced, with a predominance of non-fraudulent transactions. Synthetic data is obtained from a generative process based on properties of real data. A comprehensive survey [8] analyzed 417 SDG models developed over the past decade, highlighting the evolution of model performance and complexity. The study found that neural network-based approaches, particularly Generative Adversarial Networks (GANs), dominate the landscape, especially in computer vision applications. Emerging models like diffusion models and transformers are also gaining traction, offering promising avenues for future research.

The most common format for synthetic data generation is the tabular structure, which consists of columns (also referred to as features) and rows (also called observations). According to [9] a synthetic data generation process can be described along four different dimensions: *Architecture* (it represents the type of data augmentation technique used), *Application level* (it refers to the phase of machine learning pipeline where the process is included), *Scope* (it is related to the usage of existing dataset properties), and *Data Space* (it is related to the representation model used in the process). The primary metric for assessing the quality of synthetic data is its ability to enhance machine learning model performance, typically measured through accuracy or F1-score. Evaluating synthetic data quality before full-scale model training is crucial, as training an ML classifier can be computationally expensive and time-consuming. By anticipating data quality early, researchers can optimize resources and improve model efficiency.

Generative Adversarial Networks (GANs) have been extensively explored for generating synthetic tabular data. These models consist of a generator and a discriminator that engage in a minimax game, leading to the production of data that closely resembles real datasets. Recent studies have adapted GANs to handle the unique challenges of tabular data, such as mixed data types and complex feature dependencies. Among others frequently used synthetic data generation algorithm are CTGAN [10], GReaT [11], SDV [12]. Conditional Tabular GAN (CTGAN) is a deep generative model specifically designed for the synthesis of tabular data. Unlike traditional GANs, which struggle with the complex statistical properties of tabular datasets, CTGAN introduces several innovations to enhance data fidelity. To address the issue of managing at the same time continuous and categorical features, CTGAN employs a mode specific normalization technique based on a Variational Gaussian Mixture Model (VGM) [13], which effectively encodes continuous variables while preserving their original distribution. CTGAN also mitigates the underrepresented class problem by incorporating a conditional generator, ensuring that all categories are sufficiently represented during training. Furthermore, the model employs a training-by-sampling approach, where data instances are selected based on the log-frequency of categorical values rather than uniformly. This strategy improves the generator's ability to produce balanced and representative synthetic samples. The network architecture of CTGAN consists of fully connected neural networks for both the generator and the critic. To stabilize training and improve the quality of the generated data, the model utilizes the Wasserstein GAN with Gradient Penalty (WGAN-GP) framework [14, 15]. The generator takes as input a random noise vector along with a conditional variable and produces synthetic tabular records. The critic then evaluates these records, helping the model refine its ability to generate realistic samples.

Variational Autoencoders (VAEs) have also been applied to tabular data synthesis, focusing on learning latent representations that capture the underlying data distribution. By sampling from the latent space, VAEs can generate new data points that maintain the statistical properties of the original dataset. Additionally, models like TVAE (Tabular Variational Autoencoder) [9] have been developed to specifically address the challenges of tabular data, incorporating mechanisms to handle diverse data types and complex relationships between features.

The role of data-centric AI in improving synthetic tabular data generation and evaluation is investigated in [16]. Traditional approaches rely on statistical fidelity metrics to assess synthetic data quality, but the authors argue that these methods alone are insufficient. Instead, they propose incorporating data profiling techniques that categorize samples based on learning difficulty to guide synthetic data generation. Through extensive benchmarking across eleven datasets and five state-of-the-art synthetic data generators, the study demonstrates that considering data profiles enhances model performance, model selection reliability, and feature selection effectiveness. The findings suggest that different generative models excel in different tasks, emphasizing the need for task-specific synthetic data evaluation.

Emerging approaches have explored the use of language models for generating synthetic tabular data. By treating rows as sequences, these models can capture dependencies between features effectively. The GReaT model [11] utilizes large language models (LLMs) for synthetic tabular data generation. Instead of encoding tabular data numerically, GReaT transforms each row into a structured textual representation using a subject-predicate-object format, maintaining semantic coherence. A pretrained transformer-based LLM is then fine-tuned on this transformed dataset, enabling it to generate synthetic tabular samples while preserving statistical properties. To ensure flexibility, a random feature order permutation step is introduced, preventing the model from learning an artificial ordering among features. The model supports arbitrary conditioning, allowing data generation based on selected feature constraints. During inference, GReaT generates synthetic records by iteratively sampling feature values in an autoregressive manner. The generated data is then transformed back into tabular form through pattern-matching algorithms. A python library[3] is also available.

The development of automated platforms for synthetic data generation has streamlined the process of creating high-quality synthetic datasets. For example, the Synthetic Tabular Neural Generator (STNG) [17] integrates multiple generation methods with an AutoML module, facilitating the automatic generation of synthetic data tailored to specific tasks. This platform addresses the need for user-friendly tools that can adapt to various data characteristics and requirements.

The Synthetic Data Vault (SDV) [12] is a synthetic data generation model designed to automatically generate synthetic data, enabling data science applications while preserving privacy. SDV builds generative models of relational databases, allowing the synthesis of artificial data that retains the statistical and structural properties of real datasets. The system employs a recursive modeling technique called Recursive Conditional Parameter Aggregation (RCPA), which models the relationships between database tables to generate realistic synthetic data. SDV has been evaluated on multiple publicly available datasets, demonstrating that synthetic data can effectively replace real data in predictive modeling tasks. Its generative process incorporates multivariate statistical techniques, including Gaussian Copulas, to model data distributions

[3] https://github.com/kathrinse/be_great

accurately. SDV is now implemented in a python library called SDV[4] and it includes other synthetic data generation models such as CTGAN, and Copulas, allowing users to choose the most appropriate approach for their specific data characteristics.

Assessing the quality and utility of synthetic tabular data is crucial for its adoption in practical applications [18]. Structured evaluation frameworks have been proposed to provide a comprehensive assessment of synthetic data generators. These frameworks consider various metrics, including resemblance, utility, and privacy preservation, offering a standardized approach to evaluate and compare different synthetic data generation methods.

2.2. Data contamination

To the best of the authors' knowledge, no study has attempted to define a systematic approach for introducing errors into the dataset to realistically simulate data. In some papers, errors are introduced to test the proposed solution only. For example, in [12] authors introduced two types of noise: table and key noise. The table noise method alters the covariance structure between variables in the dataset. To introduce noise, the SDV modifies the covariance values σ_{ij} (where $i \neq j$) by halving them, effectively reducing the strength of correlations between features. This perturbation ensures that the synthetic data maintains its general statistical properties but with weaker dependencies, making it more distinct from the original dataset. The key noise method affects the integrity of foreign key relationships in relational datasets. Instead of using the inferred relationships when synthesizing child tables, the SDV randomly assigns foreign keys to synthetic records. This disrupts the original structural dependencies within the dataset, introducing additional randomness while preserving the overall schema. Such methods are only used to test the quality of the proposed synthetic data generation algorithms and there are no available noise functions in the python library.

Numerous studies have investigated the impact of the most common types of data errors on machine learning models, and an equally large number of approaches have been proposed to detect and correct these errors. In [19], authors address the problem of binary classification in the presence of random label noise, where training labels may be independently flipped with a certain probability. The authors propose two approaches to adapt surrogate loss functions for robust learning. These methods lead to a key result: widely used techniques such as weighted SVMs and weighted logistic regression are provably noise-tolerant. Authors of [20] discuss a survey of existing methods for managing the missing data problem in machine learning. In [21] a survey of outlier detection techniques reports seven different classes of outlier detection techniques for a total of 47 different proposed techniques.

3. The Pucktrick Library

To systematically introduce artificial errors into datasets and simulate real-world data imperfections, we developed Pucktrick[5][6]. This tool enables users to contaminate datasets with controlled

[4] https://github.com/sdv-dev/SDV

[5] Puck is the name of the elf in "A Midsummer Night's Dream" by William Shakespeare, who is famous for enjoying causing trouble and playing tricks on mortals and other fairies alike

[6] https://andreamaurino.github.io/pucktrick-ui-docs/

levels of errors as the ones described in Section 1, ensuring a structured and reproducible approach to studying the effects of data corruption.

The Pucktrick library is designed to introduce errors at a specified percentage, offering two operational modes:

- *New mode*: This mode introduces errors into a clean dataset, allowing users to inject a predefined level of corruption.
- *Extended mode*: This mode further contaminates a dataset that has already been modified, ensuring that additional noise is selectively introduced into previously untouched portions while maintaining the desired overall error distribution.

The library consists of five distinct modules, each designed to introduce a specific category of data errors. These errors can be applied either at the dataset level (affecting all records) or at the feature level (targeting specific attributes). The supported data types include:

- *Categorical data*: Errors such as label flipping, misspellings, and misclassifications.
- *Numerical data*: Noise injection, incorrect scaling, and value swapping.
- *Boolean data*: Random inversions of true/false values.
- *Date and time data*: Temporal shifts, incorrect formatting, and missing timestamps.

By allowing users to precisely control the level and type of errors introduced, Pucktrick serves as a powerful tool for evaluating machine learning robustness, benchmarking data-cleaning algorithms, and understanding the impact of real-world noise on analytical models. The library provides a structured, repeatable, and scalable approach to data contamination, bridging the gap between synthetic data generation and real-world dataset imperfections.

In the following subsection, we introduce the Pucktrick modules. Readers seeking more details can refer to the online documentation for further information.

3.1. Duplicate data

The *duplicate* module offers functionalities for duplicating rows within a dataset, available in both 'new' and 'extended' modes. This can be done either randomly across the entire dataset or in a targeted manner, where specific rows are duplicated based on a chosen attribute value. This feature is particularly useful for handling imbalanced datasets, where oversampling a particular class can improve the performance of multi-class classifiers. This module allows users to test how well machine learning models handle redundant information and potential data leakage scenarios.

3.2. Label misclassification

The *label* module introduces errors in target variables, simulating real-world misclassification errors that commonly occur in practical settings [22]. In binary classification tasks, the module allows direct label swapping (e.g., converting 0 to 1 and vice versa). For multi-class classification, labels are randomly reassigned to a different class within the dataset. This feature enables researchers to study the impact of label noise on classification performance and evaluate the effectiveness of error correction algorithms.

3.3. Missing data

The *missing* module includes methods for inserting "null" values in a specific column with a predefined percentage. Missing values are a frequent challenge in real-world datasets, often caused by sensor failures, human errors, or incomplete data collection. This module enables controlled experimentation with various missing data scenarios, helping to assess the performance of imputation techniques and machine learning models under different levels of data incompleteness.

3.4. Noisy data

The *noise* is one of the most complex components of the library, allowing users to introduce random distortions into dataset features. Different methods are implemented depending on the data type: For each data type outlined in Section 3, dedicated methods have been implemented.

- Continuous and discrete features: Noise is generated from a normal distribution within the range defined by the minimum and maximum values of the feature. This simulates real-world measurement errors or rounding imprecisions.
- Categorical integer features: Original values are replaced with alternative values randomly sampled from the existing set, following a normal distribution to ensure realistic variation.
- Categorical string features: The module generates synthetic categories by introducing new, normally distributed string values. This simulates scenarios where new categories emerge in real-world data (e.g., evolving customer preferences, new product categories).

3.5. Outlier data

The *outlier* module generates extreme data points to simulate anomalies and rare events in a dataset. The approach differs based on the data type:

- Continuous features: The module employs the 3-sigma method, a statistical technique for detecting and generating outliers under the assumption of a normal distribution. The method calculates the mean ($\bar{X}$) and standard deviation (σ) of the feature define the outlier boundaries as:

$$\text{Upper Boundary} = \bar{X} + 3\sigma, \quad \text{Lower Boundary} = \bar{X} - 3\sigma$$

- Integer features: A similar 3-sigma methodology is applied to discrete numerical data.
- Categorical features: To introduce categorical outliers, the module simulates an unknown category by adding values that do not exist in the dataset. These values can take the form of:
  - A new string category labeled "Puck was here", mimicking unexpected input values.
  - An integer category not previously present in the dataset, representing unseen class labels in classification tasks.

4. Pucktrick evaluation

4.1. Evaluation pipeline

To assess the effectiveness of the Pucktrick library, we construct a dedicated pipeline that highlights the advantages of introducing errors into synthetically generated data. Figure 1 provides an overview of the pipeline. The provided pipeline illustrates the process of generating, contaminating, and utilizing synthetic data for machine learning training. It begins with an original dataset, which is split into two parts: one used for generating synthetic data and another reserved as a test set. The synthetic data generator produces a synthetic dataset by applying one of the synthetic data generation algorithms referenced in Section 2 to produce a synthetic dataset that retains the statistical properties of the original data. Subsequently, this synthetic dataset is passed through the Pucktrick library to introduce controlled noise and data contamination as outlined in Section 3.

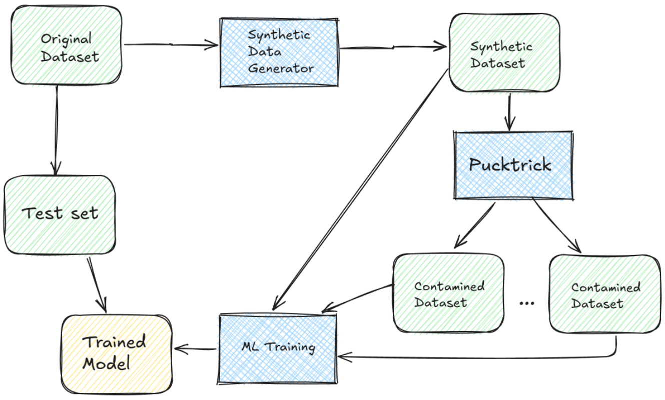

Figure 1: The proposed pipeline

To highlight the contribution of each type of error produced, the pipeline constructs a separate contaminated dataset for each introduced error type (e.g., labels misclassification, outliers, etc.). The contaminated datasets, along with the synthetic dataset, are used to train one or more machine learning models. Once training is completed, the resulting models are then employed to predict the correct class using the test set of the original dataset, which serves as a benchmark to compare the model's performance under different training conditions. The obtained results are subsequently compared with the available labels in the test set to compute standard performance metrics, such as accuracy, F1-score, precision, recall, and AUC. Finally, the different performance outcomes are compared to assess the effectiveness of the proposed approach. This approach ensures a systematic evaluation of data contamination effects on model generalization, providing insights into how well machine learning models handle real-world noisy and imperfect data.

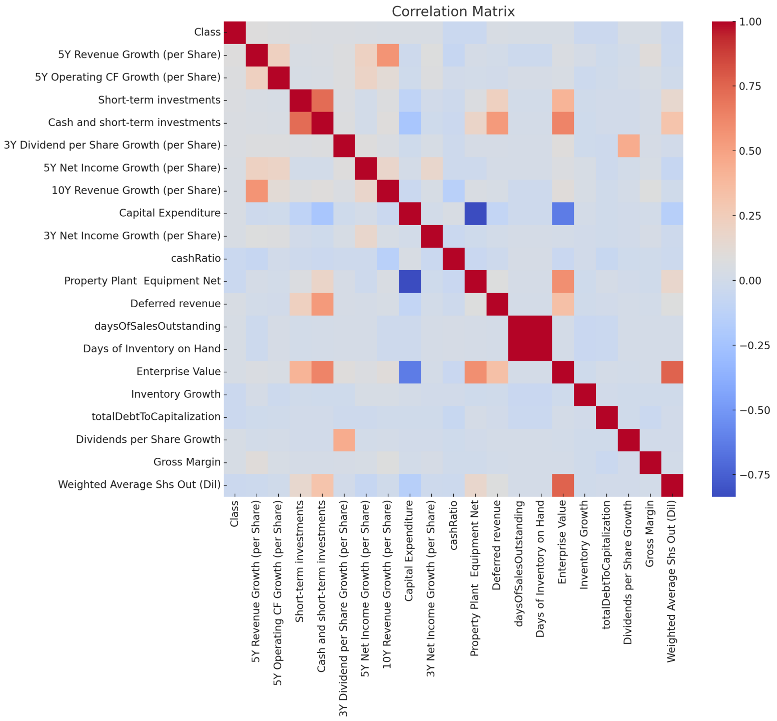

Figure 2: Correlation matrix of selected dataset (year 2014)

4.2. Dataset and experimental setup

To evaluate the effectiveness of Pucktrick, we selected a diverse set of datasets related to stock market activities spanning the years 2014 to 2018[7]. These datasets were chosen due to their highly dynamic nature, real-world complexity, and susceptibility to various types of data errors, such as missing values, noise, outliers, and label misclassifications.

The dataset collection consists of five different datasets, each containing over 200 financial indicators commonly found in publicly traded company reports. These indicators encompass key financial metrics such as revenue, profit margins, asset values, stock price fluctuations, and trading volumes, offering a comprehensive view of financial market trends. Each dataset includes information on more than 4,000 publicly traded US stocks per year, ensuring a broad representation of market dynamics. However, these datasets are not free from imperfections. Certain financial indicators contain missing values, which can result from incomplete reporting or data collection issues. Additionally, outliers are present, representing extreme values that are likely caused by data entry errors, mistypings, or unusual market behaviors. These underlying issues make the datasets particularly suitable for testing Puck's ability to simulate realistic data contamination scenarios.

The dataset also includes key financial performance indicators relevant to stock trading decisions. Among these, the second-to-last column, PRICE VAR [%], represents the percentage

[7] https://www.kaggle.com/datasets/cnic92/200-financial-indicators-of-us-stocks-20142018

price variation of each stock over the course of a given year. This metric is crucial for evaluating a stock's annual performance. The last column, class, provides a binary classification for each stock based on its price variation. If the PRICE VAR [%] value is positive, the stock is assigned CLASS = 1, indicating that it has increased in value. From a trading perspective, this signifies that an idealized trader would have benefited from buying the stock at the beginning of the year and selling it at the end for a profit. Conversely, if the PRICE VAR [%] value is negative, the stock is labeled CLASS = 0, meaning its value declined over the year. In this case, a rational trader would avoid purchasing the stock to prevent capital loss. This dataset presents a highly realistic financial scenario, where market fluctuations introduce inconsistencies and imperfections in the data. Stock price variations depend on numerous external factors such as market sentiment, economic conditions, corporate performance, and investor behavior, all of which contribute to data irregularities. Moreover, the presence of missing values, outliers, and misclassified stock performance further enhances the dataset's suitability for testing Pucktrick 's ability to introduce and manage controlled data contamination.

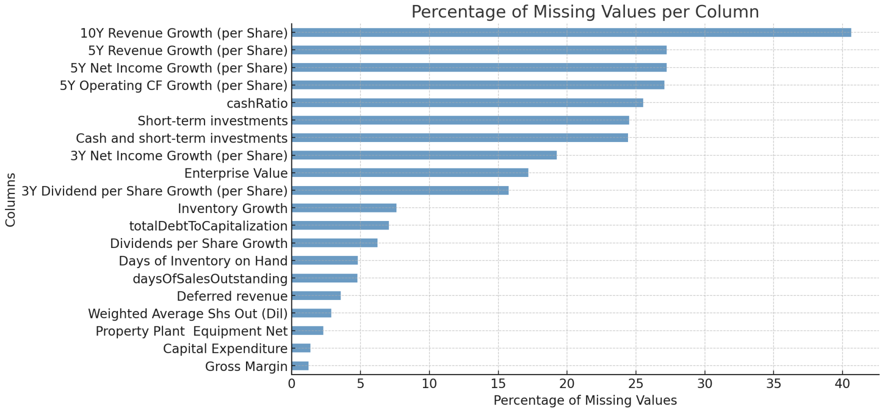

Figure 3: Missing date of the original dataset

For this experiment, we considered only the top 20 features with the highest correlation with the binary target variable. The correlation matrix of the new dataset is shown in Figure 2 highlighting the relationships between financial indicators and stock performance classification. Additionally, the dataset contains a significant number of missing values, as shown in Figure 3, which poses challenges for machine learning models and serves as a natural test case for assessing the impact of synthetic data contamination.

To generate the synthetic dataset, we used the GreaT algorithm [11] through the publicly available be_great Python library[8]. We employed the distilgpt2 large language model with a batch size of 32 for 50 epochs, ensuring a robust generative process capable of mimicking real-world financial data patterns.

To evaluate the effects of controlled data contamination, we created 61 synthetic datasets, categorized as follows:

- One dataset with only mislabeled errors (error rate fixed at 30%).

[8] https://github.com/kathrinse/be_great

Label	model Name	Python reference
SVM	SVM - Linear Kernel	sklearn.linear_model._stochastic_gradient.SGDC...
ET	Extra Trees Classifier	sklearn.ensemble._forest.ExtraTreesClassifier
RF	Random Forest Classifier	sklearn.ensemble._forest.RandomForestClassifier
KN	K Neighbors Classifier	sklearn.neighbors._classification.KNeighbors...
LDA	Linear Discriminant Analysis	sklearn.discriminant_analysis.LinearDiscriminant...
MLP	Multilayer Perception Classifier	sklearn.neural_network._multilayer_perceptron....
LR	Logistic Regression	sklearn.linear_model._logistic.LogisticRegression
NB	Naive Bayes	sklearn.naive_bayes.GaussianNB
DT	Decision Tree Classifier	sklearn.tree._classes.DecisionTreeClassifier
QDA	Quadratic Discriminant Analysis	sklearn.discriminant_analysis.QuadraticDiscri...

Table 1
Machine learning algorithms used in the experimentation.

- One dataset for each attribute containing only missing data (error rate fixed at 30%).
- One dataset for each attribute containing only noisy data (error rate fixed at 30%).
- One dataset for each attribute containing only outlier data (using the 3σ model with an error rate fixed at 30%).

To ensure a comprehensive evaluation, we utilized 10 machine learning models from diverse algorithmic categories, including tree-based models, linear classifiers, probabilistic models, neural networks, and nearest neighbors approaches. The specific models and their corresponding Python implementations are detailed in Table 1. To streamline the model selection, training, and evaluation process, we leveraged PyCaret[9], an open-source, low-code Python library that automates machine learning workflows. PyCaret simplifies the experimental pipeline by handling data preprocessing, hyperparameter tuning, and model validation, allowing for efficient large-scale model comparison.

4.3. Results

By comparing the performance of the 610 developed models (61 contaminated datasets for 10 algorithms), we obtained several key insights into the impact of synthetic data contamination on machine learning accuracy, showing that the use of the Pucktrick library significantly enhances the performance of machine learning algorithms compared to using only the synthetic dataset.

Firstly, we address the question: *Which machine learning algorithm benefits the most from the use of the Pucktrick library?*

To answer this question we analyzed the percentage of models trained on contaminated datasets that achieved a higher accuracy than their counterparts trained on purely synthetic data. The results, summarized in Table 2, reveal key insights into how different machine learning models respond to various types of data contamination. For *noisy data* the Support Vector Machine (SVM) model consistently showed the highest improvement when trained on noise-contaminated datasets across all three years (2014, 2015, and 2016). This suggests that SVM models benefit from controlled noise introduction, possibly due to their inherent ability to

[9] https://pycaret.org/

find optimal decision boundaries even in the presence of variations in the feature space. Noise may serve as a form of regularization, preventing overfitting and improving model robustness. Regarding *missing values*, Extra Trees (ET) classifier exhibited the most significant improvement when trained on datasets containing missing values. Across all years, ET was the best-performing model for handling missing data, indicating that ensemble tree-based methods are particularly resilient to incomplete datasets. This aligns with previous findings in the literature, as decision trees and ensemble models are capable of handling missing data by splitting based on available features without requiring imputation techniques. Similarly to missing values, ET consistently outperformed other models when trained on datasets containing *outliers*. Interestingly, in 2015, a linear model (ET) also showed competitive performance, suggesting that some models can effectively learn from extreme values when trained on outlier-containing data. This further confirms the robustness of tree-based models in handling anomalies, as they can partition the feature space in a way that minimizes the impact of extreme values. Unlike other types of contamination, no single model stood out as the best performer in handling *mislabeled data*. The results indicate that different models responded differently to label errors, with Linear Discriminant Analysis (LDA) performing well in 2014 dataset, while Random Forest (RF) and Extra Trees (ET) emerged as the strongest models for the 2015 and 2016 datasets, respectively. This suggests that the impact of label misclassification is highly model-dependent, and additional correction mechanisms such as semi-supervised learning or active learning strategies may be needed to mitigate its effects effectively.

	2014		2015		2016	
Error Type	Model Type	Model	Model Type	Model	Model Type	Model
Noisy	Linear	SVM	Linear	SVM	Linear	SVM
Missing	Tree	ET	Tree	ET	Tree	ET
Outlier	Tree	ET	Linear	ET	Tree	ET
Labels	Linear	LDA	Tree	RF	Tree	ET

Table 2
Errors and Model

A second question we address is: *Which type of error, when introduced into the synthetic dataset, improves or maintains accuracy across all classifiers and throughout all three years of evaluation?* To answer this question we calculated the percentage of models, trained with contained dataset, with an accuracy that is higher (best) or higher or equal (best or equal) to the same classifier trained with the synthetic dataset. The results are summarized in Table 3. Across datasets of all years, models trained on noisy data consistently outperformed or matched those trained on purely synthetic data, with 100% for the 2014 dataset maintaining accuracy and 90% improving. Although the effect weakened slightly for 2015 and 2016, 70% of models still maintained or improved performance, confirming that controlled noise acts as a form of regularization, preventing overfitting and enhancing generalization. Missing values also had a positive effect, with 82% of models in 2014 performing at least as well as those trained on synthetic data, though only 44% showed direct improvements. Similar trends were observed in 2015 and 2016, where 75% maintained accuracy, but only 27–35% improved. This suggests that while some models can adapt to missing data, others require feature engineering or imputation techniques to perform optimally. The impact of outliers was moderate, with 70–79% of models across all years maintaining or improving accuracy, while 27–39% experienced direct gains. Though not as influential as noise, outliers were better handled by tree-based models, reinforcing their resilience to high-variance data. Label misclassification had the most unpredictable effect. While 90% of models maintained or improved accuracy, only 10–20% saw actual performance gains, indicating that some models can tolerate label noise, while others are significantly affected. The inconsistency suggests that additional correction mechanisms, such as semi-supervised learning or label adjustment strategies, may be necessary to mitigate its impact.

	2014		2015		2016	
Error Type	Best or equal	Best	Best or equal	Best	Best or equal	Best
Labels	90	10	**90**	10	**90**	20
Outlier	79	39	76	29	70	27
Noisy	**100**	**90**	70	**40**	70	**50**
Missing	82	44	75	27	75	35

Table 3
Percentage of accuracy improvement wrt. errors

Table 4 presents the percentage of models that exhibited improved accuracy (Best) when trained on Pucktrick contaminated datasets compared to their counterparts trained on purely synthetic data. Additionally, it shows the percentage of models that either improved or maintained accuracy (Best or Equal) relative to the synthetic dataset baseline.

		2014		2015		2016	
Type	Model	Best or equal	Best	Best or equal	Best	Best or equal	Best
Tree	ET	**96.7**	95.1	**98.36**	85.25	**91.80**	55.74
	RF	95.1	91.8	88.52	9.84	44.26	8.20
	DT	85.2	70.5	44.26	24.59	72.13	54.10
Linear	LDA	93.4	83.6	**95.08**	9.84	86.89	11.48
	SVM	**100**	49.2	93.44	42.62	**96.72**	52.46
	LR	91,8	45.9	42.62	6.56	91.80	47.54
Neural	MLP	**93.4**	83.6	**85.25**	42.62	**95.08**	52.46
Probabilistic	QDA	65.6	63.9	**73.77**	65.57	3.28	3.28
	NB	**86.9**	1.6	63.93	19.67	**90.1**6	55.74
Nieghbors	KN	**95.1**	47.5	**50.82**	3.28	**45.90**	24.59

Table 4
Percentage of improvement of accuracy by model and type

As shown in Table 4, Puck enhances the performance of Extra Trees and Random Forests in over 90% of experiments, demonstrating its effectiveness in improving tree-based models. Furthermore, Pucktrick consistently improves or maintains the accuracy of models trained on the original synthetic dataset, ensuring that data contamination does not degrade performance. In general, the results confirm that Pucktrick positively impacts or preserves accuracy across all major machine learning algorithm categories, including tree-based, linear, neural, and neighbor-based models.

5. Conclusions

Training machine learning models with real-world data is often difficult due to privacy and confidentiality restrictions, limiting the ability to develop classifiers using actual datasets. To address this, various synthetic data generation algorithms have been developed over the years, aiming to replicate the statistical properties of real data. However, a critical limitation of synthetic datasets is their lack of real-world imperfections. Unlike actual datasets, synthetic data is often too clean, as it does not contain common data issues such as missing values, outliers, and label misclassification, which are inherent in real-world applications. In this paper we introduced Pucktrick a library designed to systematically introduce a controlled percentage of errors into datasets, either at the feature level or across the entire dataset. Experimental evaluations on three real-world datasets confirm the effectiveness of this approach. Notably, linear models such as SVMs and tree-based models demonstrate improved accuracy when trained on controlled-error datasets, suggesting that strategic contamination can enhance model robustness.

There are several directions for future research. One key focus is the development of an enhanced version of the Pucktrick library that supports a more generalized strategy for error insertion, allowing for greater flexibility in simulating real-world data imperfections. Additionally, we are exploring methods to modify dataset schemas rather than only altering data values. A common challenge arises when the schema of a dataset used in a machine learning model differs from that of the training dataset. For instance, a feature such as total living area in the training data may later be divided into interior area and exterior area in a real-world application, leading to a structural mismatch that could affect model performance. Addressing such discrepancies will be crucial for ensuring model adaptability across different data environments.

Furthermore, additional experiments are needed to gain deeper insights into the relationship between error types, model architectures, and feature selection. A key objective is to help users identify the minimum set of essential features required to improve classifier accuracy, even when working with imperfect datasets. By refining these aspects, we aim to make Pucktrick a more comprehensive tool for evaluating machine learning models under realistic data conditions.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. Paleyes, R.-G. Urma, N. D. Lawrence, Challenges in deploying machine learning: a survey of case studies, ACM computing surveys 55 (2022) 1–29.

[2] A. Figueira, B. Vaz, Survey on synthetic data generation, evaluation methods and gans, Mathematics 10 (2022) 2733.

[3] K. S, M. Durgadevi, Generative adversarial network (gan): a general review on different variants of gan and applications, in: 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021, pp. 1–8. doi:10.1109/ICCES51350.2021.9489160.

[4] R. Wei, C. Garcia, A. El-Sayed, V. Peterson, A. Mahmood, Variations in variational autoencoders - a comparative evaluation, IEEE Access 8 (2020) 153651–153670. doi:10.1109/ACCESS.2020.3018151.

[5] F. Liu, D. Panagiotakos, Real-world data: a brief review of the methods, applications, challenges and opportunities, BMC Medical Research Methodology 22 (2022) 287.

[6] S. Iskander, N. Cohen, Z. Karnin, O. Shapira, S. Tolmach, Quality matters: Evaluating synthetic data for tool-using llms, arXiv preprint arXiv:2409.16341 (2024).

[7] J. Chen, Y. Zhang, B. Wang, W. X. Zhao, J.-R. Wen, W. Chen, Unveiling the flaws: Exploring imperfections in synthetic data and mitigation strategies for large language models, arXiv preprint arXiv:2406.12397 (2024).

[8] A. Bauer, S. Trapp, M. Stenger, R. Leppich, S. Kounev, M. Leznik, K. Chard, I. Foster, Comprehensive exploration of synthetic data generation: A survey, arXiv preprint arXiv:2401.02524 (2024).

[9] J. Fonseca, F. Bacao, Tabular and latent space synthetic data generation: a literature review, Journal of Big Data 10 (2023) 115. URL: https://doi.org/10.1186/s40537-023-00792-7. doi:10.1186/s40537-023-00792-7.

[10] L. Xu, M. Skoularidou, A. Cuesta-Infante, K. Veeramachaneni, Modeling tabular data using conditional gan, in: Advances in Neural Information Processing Systems (NeurIPS), volume 32, 2019, p. 1049.

[11] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, G. Kasneci, Language models are realistic tabular data generators, in: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023, OpenReview.net, 2023. URL: https://openreview.net/forum?id=cEygmQNOeI.

[12] N. Patki, R. Wedge, K. Veeramachaneni, The synthetic data vault, in: 2016 IEEE international conference on data science and advanced analytics (DSAA), IEEE, 2016, pp. 399–410.

[13] N. Nasios, A. Bors, Variational learning for gaussian mixture models, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 36 (2006) 849–862. doi:10.1109/TSMCB.2006.872273.

[14] M. Walia, B. Tierney, S. Mckeever, Synthesising tabular data using wasserstein conditional gans with gradient penalty (wcgan-gp), in: Irish Conference on Artificial Intelligence and Cognitive Science, 2020. URL: https://api.semanticscholar.org/CorpusID:229345165.

[15] T. Milne, A. I. Nachman, Wasserstein gans with gradient penalty compute congested transport, in: P.-L. Loh, M. Raginsky (Eds.), Proceedings of Thirty Fifth Conference on Learning Theory, volume 178 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 103–129. URL: https://proceedings.mlr.press/v178/milne22a.html.

[16] L. Hansen, N. Seedat, M. van der Schaar, A. Petrovic, Reimagining synthetic tabular data generation through data-centric ai: A comprehensive benchmark, Advances in Neural Information Processing Systems 36 (2023) 33781–33823.

[17] H. H. Rashidi, S. Albahra, B. P. Rubin, B. Hu, A novel and fully automated platform for synthetic tabular data generation and validation, Scientific Reports 14 (2024) 23312.

[18] Q. Liu, M. Khalil, J. Jovanovic, R. Shakya, Scaling while privacy preserving: A comprehensive synthetic tabular data generation and evaluation in learning analytics, in: Proceedings of the 14th Learning Analytics and Knowledge Conference, 2024, pp. 620–631.

[19] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, A. Tewari, Learning with noisy labels, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 26, Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/3871bd64012152bfb53fdf04b401193f-Paper.pdf.

[20] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, E. Tabane, M. Velempini, A survey on missing data in machine learning, Journal of Big Data 8 (2021) 140. URL: https://doi.org/10.1186/s40537-021-00516-9. doi:10.1186/s40537-021-00516-9.

[21] A. Boukerche, L. Zheng, O. Alfandi, Outlier detection: Methods, models, and classification, ACM Computing Surveys (CSUR) 53 (2020) 55:1–55:37. URL: https://doi.org/10.1145/3381028. doi:10.1145/3381028.

[22] B. Frénay, A. Kabán, et al., A comprehensive introduction to label noise., in: ESANN, Citeseer, 2014.