Evaluation of Data Augmentation for Tabular Data

Fabiana Fazio[1], Kate Huisinga[2], Ace Mejía-Sánchez[3], Ellee Mortensen[2], and Marilyn Vazquez[2]

[1]Department of Science and Technology, Kean University, Union, New Jersey

[2]Department of Mathematics, Simpson College, Indianola, Iowa

[3]Department of Statistics, Harvard University, Cambridge, Massachusetts

July 26, 2024

This work is part of the Bryan Summer Research Program and the SIAM-Simons Undergraduate Summer Research Program. The Bryan program is generously supported by Dr. Albert H. and Greta A. Bryan, and the SIAM-Simons Undergraduate Summer Research Program is funded by the Society for Industrial and Applied Mathematics (SIAM) through award 1036702 of the Simons Foundation.

Abstract

This project aimed to determine whether current data augmentation methods effectively enhance tabular data while preserving the original patterns of the data. Data augmentation is "the process of artificially generating new data from existing data, primarily to train new machine learning (ML) models" [1]. Our project focused on the use of tabular data, which refers to data organized into rows and columns. Researchers and data scientists often use data augmentation because collecting new data is costly, time-consuming, or simply not an option. Prior research in this field has mainly focused on the creation of methods for data augmentation, whereas we establish a framework that measures if patterns are kept throughout the data augmentation process. By maintaining data patterns, one ensures that the patterns in the augmented data are not false or misrepresented. We measured three types of relationships: feature-to-label dependency, feature-to-feature dependency, and feature distributions. To do so, we utilized Chi-Squared tests, correlation matrices, subcategorical proportions, and the Mann-Whitney U test. For all of these tests, we used the Support Vector Machines (SVM) classifier. Our augmentation methods included modified $+/-$ one (modPMOne), random swap (randSwap), and the histogram augmentation technique (HAT). We found that these three augmentation methods were able to sufficiently maintain patterns throughout the augmentation process, though the results varied with each test we conducted. Generally, HAT performed the best.

Contents

1 Introduction

The Machine Learning phenomenon is evolving rapidly, enabling computers to learn from data and make decisions or predictions without explicitly being programmed to do so. The primary challenge researchers face when using Machine Learning techniques is that the efficiency of these models are heavily dependent on the size of the data sets used for training. Most of these models require adherence to the "10 Times Rule," [2] which suggests that a data set should have at least ten times the number of rows as feature columns. Furthermore, larger and more complex data sets may require even more observations to have truly accurate classification. Although adherence to this rule helps ensure that the model has sufficient data to learn from, it is not realistic to always obtain such large amounts of data.

An example of this issue being faced by researchers is a pediatric sleep apnea study [3]. After trying and failing to use a variety of cutting-edge methods, these researchers found the performance of predictive models for managing sleep apnea and its associated complications unsatisfactory.

To address this issue, Data Augmentation has emerged as a critical technique. Data Augmentation is the process of creating synthetic training data from an existing data set. This is done without the necessity of collecting more data, which can be time-consuming and costly. Synthetic data points are artificially created rather than obtained through direct measurement or collection. While data augmentation is typically used for computer vision, this project is focused on analyzing data augmentation with tabular data. This is a more sophisticated process, and there has been much less work done in the past on this topic. Tabular data refers to data organized in a table format consisting of rows and columns. The significance of this research lies in ensuring that the data augmentation for distinct data sets remains realistic and maintains the underlying patterns and relationships present originally.

This project began in 2022 with another group of researchers [4]; their work laid the foundation by utilizing data augmentation to address the challenges of limited data. Throughout our project, we tested various methods for measuring data patterns alongside different classifiers. Leveraging models that have proven successful in past research, we evaluated different data augmentation methods to assess their effectiveness. The primary focus of this project was to determine how closely the synthetic data resembled the original data set, as it is essential that the augmented data resembles the original data, or it cannot be used to represent the original data. Our most

significant contribution to this research was the development of a comprehensive function called `Fake_Measures`. We designed this powerful function that leverages different methods to ensure that the main patterns within the original data set are maintained. Alongside `Fake_Measures`, we created an subsequent function called `Fake_Differences` to measure the differences between the original and augmented patterns. By leveraging these functions, we can systematically evaluate and compare the effectiveness of various data augmentation techniques across multiple data sets.

1.1 Background

One of the most common applications of data augmentation is to computer vision. In Figure 1, an image of a cat is shown [5]. On the right side, the cat undergoes various data augmentation transformations where the original image is rotated and cropped. These transformations create new observations for a machine to learn from, each retaining the fundamental characteristics of the original image. This principle is crucial to our project; it is imperative to maintain the original data patterns while performing data augmentation. Adhering to this standard ensures that the augmented data is an accurate representation of the original data and valuable for model training.

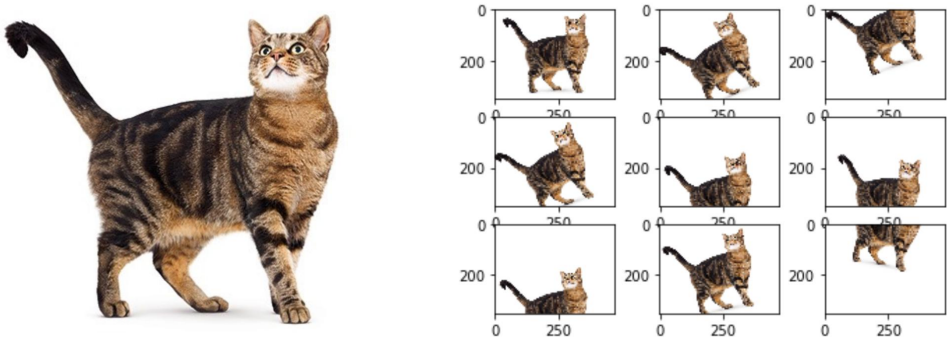


Figure 1: Data Augmentation with Computer Vision

Synthetic data plays a crucial role in enhancing the performance of data augmentation models. For example, consider Figure 2 below, depicting two classes represented by red and blue. Initially, these classes are not very distinctive, making it challenging for a model to classify them accurately. With the addition of synthetic data, the boundaries between the classes become better defined. As shown in Figure 2, the incorporation of synthetic data allows the boundaries between the classes to become better defined, thereby enhancing model performance. Data augmentation techniques help

ensure that the underlying patterns in the data are preserved. Overall, this process helps maintain high accuracy scores by providing the model with a more comprehensive data set from which to learn.

(a) Pre-Augmentation (b) Post-Augmentation

Figure 2: Scatter plots of Data Before and After Augmentation

1.2 Contributions

To date, there is no comprehensive methodology for testing whether patterns are maintained after augmentation. Our contributions to this project include:

- The creation of a framework that allows researchers and data scientists to evaluate if various types of relationships are maintained following augmentation.
- The development of two functions, `Fake_Measures` and `Fake_Differences`, which measure the patterns in a data set and compare patterns between data sets, respectively.
- Experimentation with synthetic data to test pattern retention after augmentation.

Coming next, we examine into the core topics of this report. Section 2 covers the classifier methods, exploring various techniques used to categorize data. Section 3 presents the literature review, summarizing key research findings and developments in the field and providing context and background for our research. In section 4, we discuss pattern methods, which involve identifying and analyzing patterns within data sets to extract meaningful information. Section 5 outlines the preliminary results, showcasing the initial findings and insights gained from our experiments. Section 6 addresses the challenges and limitations encountered during the research, highlighting potential obstacles. Finally, we provide an overall summary and conclusion in Section 7 describing

how we are able to maintain patterns in data augmentation for tabular data, reflecting on the research's contributions and proposing potential directions for future work.

2 Classifier Methods

To discuss classifier methods effectively, it is essential to first understand what classification is. Classification is the process of organizing data into distinct classes, enabling efficient categorization and analysis. The key to performing classification is using a classifier, an algorithm that automatically sorts or categorizes data into one or more of these predefined classes. By learning from labeled training data, classifiers can accurately predict the class of new, unseen data, making them essential tools in various fields [6].

A visual explanation of how F1 scores are calculated can be seen in Figure 3. An F1 Score is a common way to measure the performance of classifiers, as it uses both precision and recall in its measurements. To compute the F1 score, we need to be familiar with four key terms: true positives (TP) are instances that were correctly identified as positive by the classifier. For example, in the context of COVID-19 testing, a true positive would be a test result that correctly identifies an infected patient. False positives (FP) occur when the classifier incorrectly labels a negative instance as positive, such as a COVID-19 test that incorrectly identifies a healthy person as infected. True negatives (TN) are instances that were accurately classified as negative, like a test that correctly identifies a healthy person as not infected. While false negatives (FN) are those that were incorrectly classified as negative when they should have been positive, such as a test that fails to detect the COVID-19 virus in an infected patient.

From these terms, we derive two important metrics: precision and recall. Precision measures the accuracy of the classifier when it predicts a positive outcome, meanwhile, recall measures the classifier's ability to identify all relevant positive instances, as shown in Figure 3. Therefore, it is important to acknowledge that F1 scores combine both precision and recall into a single metric, providing a balanced view of a classifier's performance, as illustrated in the mathematical equation in 4. These scores are on a scale from 0 to 1. The closer to 1 the score is, the better the classifier is performing.

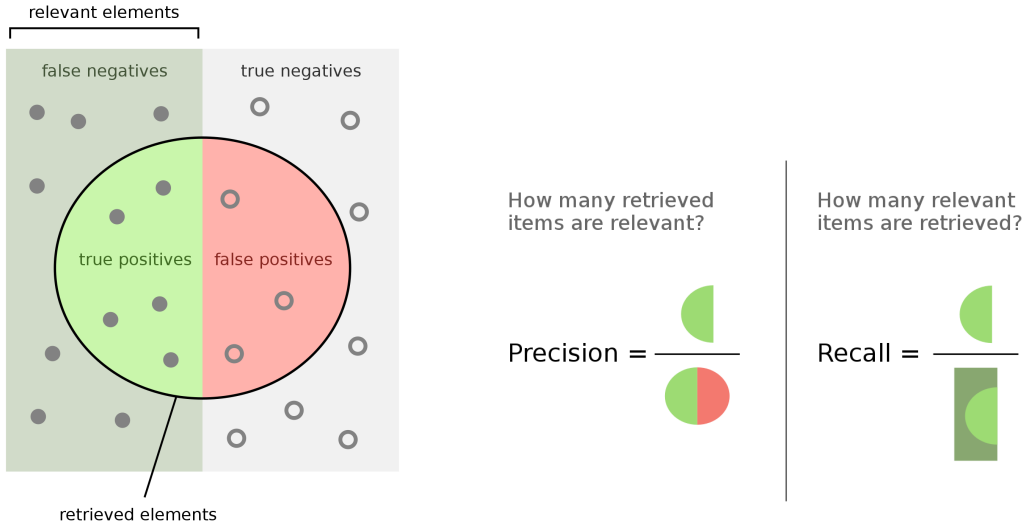


Figure 3: F1 Score Diagram

$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}}.$$

Figure 4: F1 Score Equation

2.1 Support Vector Machines (SVM)

Support vector machines (SVM) "are a set of supervised learning methods used for classification, regression and outliers detection" [7]. SVM can be used for either binary or multi-class classification. For our SVM classification, we imported the `svm` module from the `sklearn` library. This supervised-learning model finds the best boundary (hyperplane) to separate different classes.

From the `sklearn.svm` module, we imported `svc` for use. Support Vector Classification (SVC) asks for a kernel input to determine what type of kernel to use in the algorithm. The algorithm transforms data into the third dimension to make separation with a hyperplane easier. We used a `linear` kernel input, which creates a straight line in two dimensions or a flat plane in three dimensions as the decision boundary between classes.

2.2 eXtreme Gradient Boosting (XGBoost)

eXtreme Gradient Boosting (XGBoost) implements the gradient boosting algorithm. Using this technique, models are built sequentially, with each new model correcting the errors of the previous one. XGBoost is able to handle large data sets and is known for its high accuracy and speed. To use this classification method, we imported the `xgb` and `XGBClassifier` modules from the `xgboost` library.

The parameters that we specified when we used XGBoost were `n_estimators`, `max_depth`, `learning_rate`, `objective`, and `enable_categorical`. We dictated that `n_estimators` should create two trees (estimators) for use and that `max_depth` should limit the depth of these trees to two. Increasing the number of trees or depth of these trees may cause the algorithm to perform slower and therefore be less practical. For objective, we specified `binary:logistic` in order to output probabilities. The `enable_categorical` parameter, which we set equal to `true`, allows the model to handle categorical features without requiring the user to manually preprocess it into a numerical format.

2.3 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric machine learning algorithm, meaning that it does not make any assumptions about the underlying data distribution. It relies on a distance metric to find the `k` closest neighbors to a given point, where `k` represents an arbitrary number. These distance metrics can vary, but the `KNeighborsClassifier` module that we used from the `sklearn.neighbors` library uses Euclidean distance by default. `KNeighborsClassifier` has a `n_neighbors` parameter, which we set equal to five. This means that the label of a given data point is predicted by examining the five nearest neighbors and getting them to "vote" on what the label in question should be.

2.4 Applications

In our research, four data sets were selected to test the proposed methods. First, a vehicular insurance claim data set from the Oracle Database was chosen to predict whether an individual conspires to make fraudulent claims following an accident. Another data set, from the World Values Survey Project, collected between 2017 and 2020, was used to predict whether an individual

considers themselves 'happy' based on their social values and demographic information. Additionally, a 2022 national survey on drug use and mental health from the Substance Abuse and Mental Health Services Administration was utilized to determine whether an individual has been arrested based on various health variables. Finally, a Division III softball data set was created by compiling team statistics from the 2024 softball season, sourced from the National Collegiate Athletic Association, to predict whether a team will make it to the World Series based on team statistics.

2.4.1 Classification with Data Sets

We processed each of these four data sets using the three classifiers mentioned earlier to evaluate how effectively they could categorize the original data sets into classes. The results of these tests can be seen in Table 1. The numbers in the table are F1 Scores. In Table 1, it can be seen that the F1 Scores for the data sets range from 0 all the way up to 0.92. When we are looking at data sets that we potentially want to augment, we are looking for F1 Scores around 0.80 or below. For our purposes, if the scores are higher than this, then there is not much point in going through the data augmentation process because there is less room for improvement. In real-world cases, 0.80 is not a very good F1 Score, such as in the medical field, where accuracy needs to be very high. However, in the case of our experiments, this is what we are looking for. The Vehicles data set performed the worst with all three classifiers, with the classifiers giving an F1 Score of almost 0. On the other hand, the WVS (World Values Survey) data set performed the best with the classifiers, and the lowest F1 Score being 0.78. Both the Softball and Arrests data sets performed somewhere in the middle. The Softball data set performed better, with its F1 Scores ranging from 0.67 to 0.75. The Arrests data set performed worse, with F1 Scores between 0.40 and 0.49.

	Softball	**Arrests**	**Vehicles**	**WVS**
XGBoost	0.75	0.40	0.00	0.88
KNN	0.67	0.49	0.01	0.92
SVM	0.67	0.47	0.07	0.78

Table 1: F1 Scores of the Classifiers on Real-World Data Sets

Looking at Table 1, specifically for the data sets with low F1 Scores, we can understand the motivation behind data augmentation. The goal with augmenting data is to maintain the patterns

of the original data, but provide the model more information to learn from. Then, the hope is that the classifiers will be better at classifying the data sets, meaning the F1 Scores will increase. There are numerous augmentation methods that have been developed by researchers and data scientists in the past, and the methods we explored will be discussed further in the Literature Review.

3 Literature Review

To determine what augmentation methods to use, we conducted a literature review in which we read about previously researched augmentation methods for tabular data. Through this literature review, we gained valuable insights that informed our approach. We implemented the highest performing methods from our research and evaluated their performance and effectiveness in augmenting tabular data. Our aim was to determine how well each of these methods initially performed, as well as how they maintained patterns after augmentation.

3.1 HAT and FIC

In the project "Feature-based augmentation and classification for tabular data" [8], the authors developed two methods, the Histogram Augmentation Technique (HAT) and the Feature Importance Classifier (FIC), to improve data augmentation and classification for tabular data. The purpose of HAT is to generate synthetic data that has a similar distribution to the original data set. This generation is achieved through different processes for continuous and discrete columns. HAT uses statistical tests to select values for continuous columns based on the original distribution. For discrete columns, proportional sampling from each subcategory is employed to maintain the original distribution. Unlike other methods, HAT explicitly ensures that the generated data retains the properties of the original data. This rule makes it more useful for training Machine Learning models without losing essential patterns, which is crucial for maintaining the integrity and usefulness of the augmented data.

FIC aims to improve classification accuracy by focusing on the importance of individual features. Instead of treating all features equally, FIC ranks them based on their importance to the classification label using a Chi-squared test. It then creates weak learner models for each feature and combines their predictions using weighted voting, where the weights are determined by how much each feature depends on the classification label. Weak learner models are models that only

perform slightly better than random chance. In the literature, this approach improved classification accuracy by 5.54% on average compared to other classifiers tested [8]. By concentrating on the most relevant features and nullifying uncorrelated ones, FIC avoids issues like the 'Curse of Dimensionality,' which can affect models with many features. The 'Curse of Dimensionality' refers to "the various challenges and complications that arise when analyzing and organizing data in high-dimensional spaces" [9]. This method ensures the model's performance is not hampered by unnecessary or poorly predictive information.

Without a doubt, the research on the HAT augmentation method and FIC classifier is one of the key studies that inspired our work, as it emphasizes feature-based augmentation, a critical aspect of our evaluation metrics for maintaining data patterns during the augmentation process. HAT's strategy of generating synthetic data that preserves the original distribution and FIC's approach to enhancing classification accuracy by focusing on feature importance both align closely with our objectives. In our presentation, we will delve deeper into which of these two methods we ultimately utilized and which one we did not, but it is clear that both techniques offered valuable insights for our research.

3.2 Bryan Research Program 2022

Further data augmentation methods were pulled from the 2022 Bryan Summer Research program, which primarily worked on developing data augmentation methods for tabular data [10]. They created one large function to implement their data augmentation process: `bestAndNewestSuperFunction`. This function allows for a user to choose a data augmentation method from the following: pmOne, modPMOne, gausNoise, modGausNoise, randSwap, or corrAugmentation. In conjunction with these methods, there are the following available classifiers: Logistic Regression, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). We incorporated KNN and SVM as classifiers in our experiments, as well as three of their data augmentation methods, which are further explained below.

A significant achievement of the previous group was the development of a comprehensive data augmentation framework, as illustrated in Figure 5 [4].

This framework provides a visual comparison of the typical classification process versus the data augmentation process. The traditional path involves fitting the raw data to a classifier and then obtaining an accuracy score. However, these researchers created a new framework where they

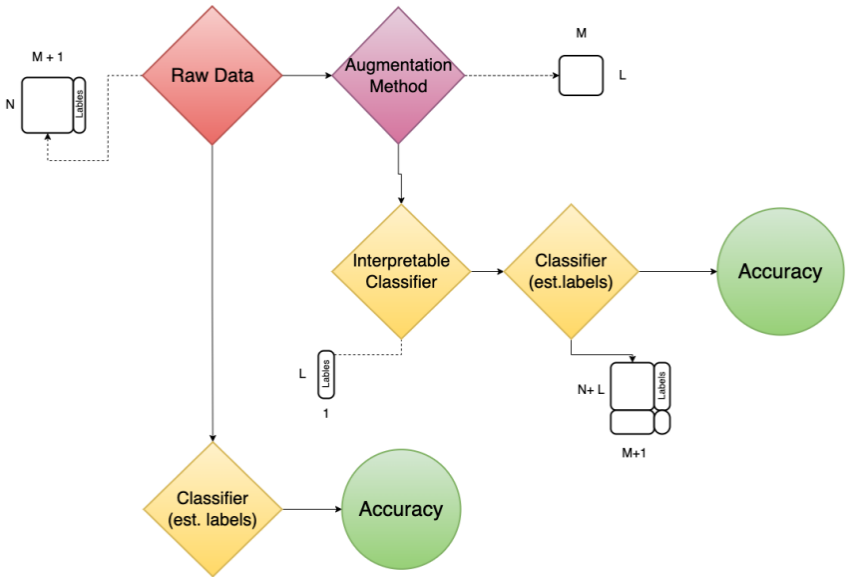


Figure 5: Augmentation Framework

first passed the raw data through an augmentation method. Once augmented, they processed the data through an interpretable classifier, followed by a standard classifier, and finally, they obtained the accuracy score. A classifier is interpretable "when humans can readily understand the reasoning behind predictions and decisions made by the model" [11]. These models ensure transparency by allowing users to see how decisions are made. This innovative approach aims to enhance the accuracy of the classifiers by leveraging augmented data. It is crucial to obtain and improve the accuracy score after the raw data is passed through the longer path of the framework to ensure the effectiveness of data augmentation techniques. Subsequently, we will delve deeper into the augmentation methods from the Bryan Research Program 2022 that our team utilized. These methods are highly effective and serve as a strong foundation for our own contributions, allowing us to build upon their work. Moreover, their flowchart served as a guide in the development of our own evaluation framework

3.2.1 Modified +/− One (modPMOne)

In this project, we utilize modPMOne as the first augmentation method for our experiments, as detailed in subsections 5.3 and 5.4. in simple words, modPMOne is a modification of pmone, a simpler precursor to this method, that augments data by adding or subtracting a unit across

multiple columns randomly while taking into account their minimum and maximum values. Users are able to manually input the size of the unit while remaining within the statistical range of each respective column.

3.2.2 Gaussian Noise (gausNoise and modGausNoise)

Two additional data augmentation methods that the previous Bryan Program research team developed included the use of Gaussian noise across random rows, as opposed to a single value like modPMOne, to augment the data. gausNoise, adds Gaussian noise with a mean of zero, $\mu = 0$, and a standard deviation, δ, of the user's choosing. It is important to note that these augmented values are floats as opposed to integers, and may provide inconsistent or illogical values for some data sets.

modGausNoise accomplishes the same task as gausNoise by creating a Gaussian distribution from each column's mean, μ, and standard deviation, δ. From this Gaussian distribution, a random value is set aside and swapped with another value from the same column for each augmented row. This data augmentation method assumes that each column is a Gaussian distribution and requires no user input.

3.2.3 Random Swap (randSwap)

Additionally, randSwap, a second data augmentation method from Bryan Research Program 2022 article [10], will also be employed as detailed in subsections 5.3 and 5.4. randSwap requires no user input and randomly grabs values for the augmented rows from the original data set's columns. However, there is no control over randomness, therefore this method can change the original variables relationships post-augmentation [10].

4 Pattern Methods

Our primary contribution to this project was creating a framework for measuring data patterns within data sets before and after data augmentation. This advancement allows us to better understand how the data augmentation method changes the original data set and assess the quality of the augmented data. This process was done in two steps, through the creation of two Fabiana-Ace-Kate-Ellee (F.A.K.E.) functions. First, we created a function called `Fake_Measures`, which takes in one data set at a time and measures the patterns found in the data set, which will

be discussed in Section 4.2. Then we created a function called `Fake_Differences`, which takes in two data sets at a time; this function allows the user to input both the original data set and the augmented data set to compare the patterns between the two and ensure that they remain consistent. This function will be discussed in-depth in Section 4.3.

4.1 Assumptions

Before discussing the function that we created, we want to address the assumptions that we are making during this process. In the endeavors of this project, our primary assumption is that it is possible to maintain data patterns from the raw tabular data post-augmentation. Suppose we have a data from the World Values Survey and the United Nations Development Programme, where each row is a survey participant coupled with the characteristics of the country they are from. An example of a feature-to-feature relationship that we want to keep is portrayed in Figure 6, where one feature, life expectancy at birth, and another feature, gross national income per capita, are positively associated. In this case, we want to ensure that any synthetic data generated by the data augmentations methods reflect this same pattern, as well as all of the other patterns found in the data set. An example of a feature-to-label relationship from this same data set would be income index and education index, given that either variable is our 'label' or the column we are asking our classifier to predict. Last, evaluating feature distributions in this data set would mean analyzing the patterns in each column of the dataset, including the survey participant's individual demographics, their country's demographics, and survey responses. It would be highly inappropriate if the augmentation methods were to create new or illogical patterns, or create values that lie beyond the original data's maximum and minimum values.

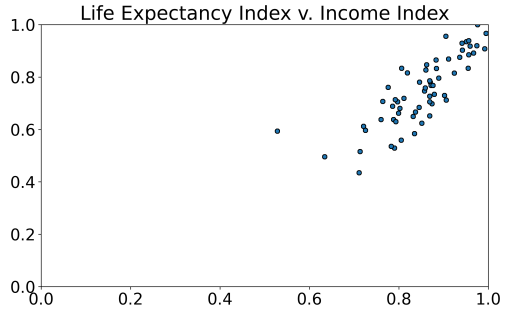


Figure 6: Scatter plot of country life expectancy index and income index

Lastly, we also assume that the initial data sets utilized for our project are high-quality and representative of our target population. The augmented data set will always reflect the original data set. Thus, we must assume that the original data represents our target population and is fairly unbiased in order to use the augmented data to train classifiers and obtain a high accuracy ethically.

4.2 Measuring Patterns: F.A.K.E. Measures

First, we created the `Fake_Measures` function to measure all of the patterns within a given data set through a series of tests. Initially, the function takes in a raw data set, including the label column. Then, the data is split into categorical and numerical data, because the patterns are measured by different tests depending on the type of each variable. We created a framework to visualize how the function works, shown in Figure 7. `Fake_Measures` synthesizes the different tests we developed to measure the patterns within our data sets. Depending on the type of variable each column contains, the function applies the appropriate tests to examine patterns, including feature-to-feature relationships and feature-to-label relationships as well as individual feature distributions. The function and each of these tests will be described in further depth below.

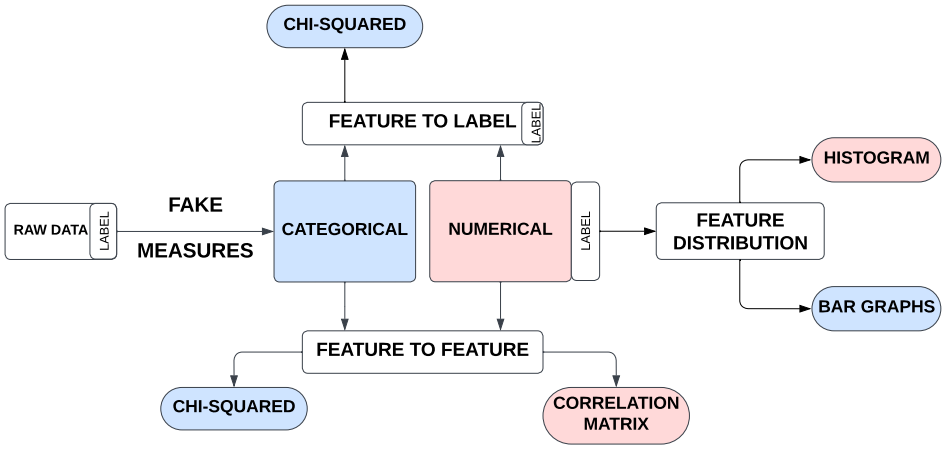


Figure 7: Measuring Patterns Framework

Currently, the function allows for three arguments: `features`, `label`, and `optional`. `Features` and `label` represent all of the data in the data set, with `features` representing all of the feature columns (every column besides the label column) and `label` representing the label column. The `optional` argument guides how the function will differentiate between categorical and

numerical variables. Our function, by default, treats all floats as numerical variables and every other data type as categorical. Therefore, if there is a variable made up of integers, it will treat the column as a categorical variable. Since integers are often used for both categorical and numerical data, `optional` takes a boolean list where True represents categorical columns so that the user has greater control of how the function analyzes their data set. This allows the user to specify which of their integer variables should be treated as truly categorical or numerical.

4.2.1 Feature-to-Feature Comparison

Feature-to-feature comparison refers to when two features from a data set are compared to examine the relationship between them, if any at all. A feature is one of the columns of the data set, or an independent variable. Feature-to-feature comparison can be used for numerical and categorical data, but the methods differ depending on which one is used. For feature-to-feature comparison of numerical data, we find the correlation between all numerical columns. For categorical data, the Chi-square test of independence is applied. These two methods are discussed further below.

First, we analyze the correlation between numerical columns. Before diving into the method we use to examine the correlation between numerical columns, we must define correlation. Correlation is "the degree to which two or more quantities are linearly associated" [12]. Correlation measurements are on a scale from -1 to 1, where -1 represents a perfect negative relationship, and 1 represents a perfect positive relationship. The closer the correlation is to -1 or 1, the stronger the relationship. The closer to 0, the weaker the relationship is.

Fortunately, there is an easy way to find correlations between columns using the Pandas package in Python. Using our data frame, which we will call "df", we can call the correlation function via the command `df.corr()`. We adjust the method parameter to use the Pearson correlation coefficient, `method=pearson`, but this method can be changed depending on the user's needs. In this case, the Pearson correlation coefficient measures linear correlation between two data sets.

Using this on our data frame will output a correlation matrix, shown in Figure 8. A correlation matrix is "a table in which every cell contains a correlation coefficient" [13]. This specific correlation matrix comes from a uniform synthetic data set with a slight separation between classes. In this correlation matrix, the correlations between all the columns (except for the

correlations between each column and itself, which are always 1) range from approximately 0.7 to 0.8, indicating strong positive relationships between all variables.

	1	2	3	4	5	6	7
1	1.000000	0.767573	0.738057	0.752376	0.725562	0.768269	0.766072
2	0.767573	1.000000	0.750275	0.745639	0.766979	0.742926	0.758110
3	0.738057	0.750275	1.000000	0.745440	0.728711	0.771735	0.724196
4	0.752376	0.745639	0.745440	1.000000	0.757583	0.737913	0.741724
5	0.725562	0.766979	0.728711	0.757583	1.000000	0.746112	0.777179
..	...	...	...	...	...	...	...
145	0.748385	0.753294	0.771767	0.740646	0.742525	0.763958	0.734994
146	0.754933	0.766213	0.748129	0.772422	0.755325	0.766268	0.754908
147	0.759659	0.744549	0.761166	0.750440	0.731195	0.752059	0.739595
148	0.738154	0.746596	0.739993	0.707117	0.726581	0.722616	0.724181
149	0.771622	0.758519	0.766564	0.754624	0.733481	0.740484	0.752919

Figure 8: Example Output for the df.corr() Method

The `Fake_Measures` function also runs the Chi-square test of independence to conduct feature-to-feature comparisons between categorical columns. As implied by its name, it looks at the dependence of one variable on another and is a non-parametric hypothesis test [14]. In other words, the Chi-square test of independence looks at whether two variables are likely to be related or not. The null hypothesis of this test is that there is no dependence between the two variables. When the test is run, there are 2 outputs: a Chi-square statistic and a p-value. The Chi-square statistic measures the difference between the observed and expected values, although its meaning is only fully interpretable given the critical value [14]. The p-value determines whether or not the null hypothesis should be rejected. If the p-value is significant, meaning it is less than 0.05, then the null hypothesis can be rejected. In other words, there seems to be a dependency between the variables. Conversely, if the p-value is greater than 0.05, the null hypothesis cannot be rejected, meaning there is not a statistically significant relationship between the variables.

4.2.2 Feature-to-Label Comparison

Feature-to-label comparison is when one of the feature columns in the data set is compared to the label column to examine what the relationship between the feature and the label is, if any at all. As a reminder, the features are all of the independent variables, which include every column except the label column, which can also be referred to as the dependent variable. In the `Fake_Measures` function, there is only one test that conducts a feature-to-label comparison for categorical columns and the label. We tested several methods for measuring relationships between numerical columns and the label column, but none of these were included in the final function.

This is explained in-depth in Section 6.1.1.

In Figure 7, one can see that the Chi-square test of independence is also used for conducting feature-to-label comparisons with categorical columns. The test works in the same way as previously described, but instead of measuring each categorical column against one another, each categorical variables is only compared to the label column, so the output of this function is much more condensed. However, the function still outputs a Chi-square statistic and a p-value, which have the same meaning as previously described.

4.2.3 Feature Distributions

The last objective of the function is to measure the distributions for each feature in the data set. For every feature in our data sets, we want to look at their distributions and measure them in some way. For numerical columns, we used histograms to understand the distribution of the observations in each column. An example of a numerical column would be age, where we can see the distribution of the ages for all of the observations. The `Fake_Measures` function sends all of the numerical columns through the histogram function, which outputs the histogram information for each numerical column. Each histogram has 10 bins, so the output includes the bin edges, the bin heights, and, most importantly, the bin probabilities, which are the proportion of observations that lie within each bin. This gives us an idea of what the distribution of each numerical variable looks like.

For the categorical columns, we used bar plots to analyze what proportion of observations are in each subcategory within every categorical variable. An example of subcategories for a categorical variable would be subcategories such as male, female, etc. for a gender variable. In the `Fake_Measures` function, all of the categorical columns in the data set are passed through this bar plot function. Similar to the bin probabilities, this function outputs the subcategory probabilities, which are the proportions of observations within each subcategory. Again, this gives us an idea of what the distribution of categorical variables looks like. The main difference is that the order of the bars does not have significance, whereas with the histograms, the variables are continuous, and the order of bins matters. This is important to note, because it is the reason we need separate functions for measuring numerical column distributions versus categorical columns.

4.3 Measuring Differences: F.A.K.E. Differences

As mentioned above, we also created a second function called `Fake_Differences`. The `Fake_Differences` function is designed to compare two data sets: a data set before augmentation and an augmented version of it. The function evaluates differences between the data sets through several measuring tools used in Figure 7, and outputs results that help understand the impact of the augmentation process . As a reminder, the goal with augmentation is to ensure that the augmented data set has the same patterns as the original data set, meaning it accurately represents the original data set.

The function takes three parameters. The first parameter, `data1`, is the data set before any augmentation. The second parameter, `data2`, is the augmented data set. The third parameter, `optional`, is a boolean list indicating whether the columns are categorical (`True`) or numerical (`False`). If the `optional` parameter is not provided, the function assumes all columns containing integer values are categorical.

The `Fake_Differences` function performs a detailed comparison between the two data sets (`data1` and `data2`). Although it does not actually use the `Fake_Measures` function, it carries out the primary types of measuring tools from the `Fake_Measures` function such as the Chi-square test for feature-to-feature and feature-to-label comparison, and the correlation test for feature-to-feature comparison. Additionally, there is the implementation of the Mann-Whitney U Test, which measures the feature distribution differences. The function also compares the subcategory proportions in each categorical variable using Euclidean Norms for bar graphs.

4.3.1 Feature-to-Feature Comparison Differences

From the `Fake_Measures` function, the `Fake_Differences`function evaluates the correlation matrices of the numerical features in both data sets. This test, called the Correlation between Columns (feature-to-feature) test, uses the Frobenius norm to compute the difference between the correlation matrices [15]. This norm is found within the `numpy` package and can be utilized as `np.linalg.norm( , ord = 'fro')`. A Frobenius norm of a matrix is "the sum of the squares of all elements of the matrix" [16]. The norm outputs the absolute and relative errors between the correlation matrices, providing a measure of how the relationships between numerical features have changed due to augmentation, as shown below.

Absolute Error: |Correlation 1−Correlation 2|

Relative Error:

$$\frac{|\text{Correlation 1} - \text{Correlation 2}|}{|\text{Correlation 1}|}$$

Additionally, the feature-to-feature Chi-squared test compares the categorical columns (excluding the label column) between the two data sets. It calculates the Chi-squared significance values and p-values, and outputs them in separate data frames. These p-values are then converted into a boolean data frame, which is a table of true and false values. Where `True` indicates a p-value less than 0.05, suggesting a significant relationship, corresponding to the alternative hypothesis (H_a). Conversely, a `False` value indicates a p-value greater than 0.05, suggesting no significant relationship, which corresponds to the null hypothesis (H_0). The final output is the count of changes in the boolean data frame, highlighting how many relationships have changed between the categorical features in both the original and augmented data sets.

4.3.2 Feature-to-Label Comparison Differences

When it comes to comparing feature-to-label differences, the Chi-squared test evaluates the relationship between each categorical feature and the label column. Unlike the Chi-square for feature-to-feature, which presents a table, the feature-to-label Chi-squared test outputs an array of p-values. These p-values are then converted into boolean values (True or False) based on their significance. This approach simplifies the interpretation by providing a straightforward count of significant changes. The interpretation of `True` and `False` remains the same as in the feature-to-feature test, indicating whether the relationship is statistically significant or not.

4.3.3 Feature Distribution Differences

While our `Fake_Measures` function found the distributions for each feature in our synthetic data sets, we want to know the distribution of the observations in each numerical column and the proportions of observations in each subcategory of all of the categorical columns before and after augmentation so we can evaluate the data augmentation method's overall performance. This is done to ensure that the patterns within each column (variable) remain consistent after we augment the data and that the patterns have not been significantly altered. Two of the many tests contained in the `Fake_Differences` function address this type of analysis. For numerical

variables, the Mann-Whitney U Test is applied to compare the histogram distributions of each numerical column before and after the augmentation process. For categorical columns, we put the proportions of observations in each subcategory into an array and found the Euclidean norm of the relative error between the two arrays (before and after augmentation). These methods are further discussed below.

The Mann-Whitney U Test is a test that compares the distributions of numerical data. The test does not assume anything about the distribution of the data (e.g. it does not assume the data has a uniform or Gaussian distribution, for example), meaning it works for any type of distribution [17]. This is ideal given that working with certain types of distributions is not a guarantee. In this case, our end goal is to use it to compare the numerical distribution of a column before and after augmentation to see how similar the distributions are. The null hypothesis of this test states that two distributions are identical. Because we want the distributions to be similar, we do not want to reject the null hypothesis. To determine whether to reject the null hypothesis or not, we have to look at the p-value output from the test. If the p-value is below 0.05 , this means that the difference in the distributions is statistically significant. In other words, a p-value lower than 0.05 indicates that the distributions are different, and thus the patterns would not have been kept during the augmentation process. As discussed previously, this is not what we want. We want the p-value to be greater than 0.05, because the greater the p-value, the less likely it is that the difference in distributions happened by random chance, which is what the goal is when augmenting our data, since we want to ensure patterns are kept and not just by chance.

In Figure 9, we can see an example of the output for a numerical column for a data set before versus after augmentation. The green distribution represents the distribution pre-augmentation and the purple represents it post-augmentation. The blue is where the distributions overlap, so it is ideal to have a copious amount of blue. Furthermore, a p-value is also output. In this example, we can see that the p-value is 0.21. This is greater than 0.05, meaning the null hypothesis cannot be rejected, so the distributions are not significantly different and therefore demonstrates that patterns have been maintained.

Just like with the numerical column distributions, we also need to be able to compare categorical column distributions pre- and post- augmentation. Specifically, we can look at the proportions of observations in each subcategory. To compare the subcategory proportions, we are looking at the relative error using the Euclidean norm of the differences between the proportions of

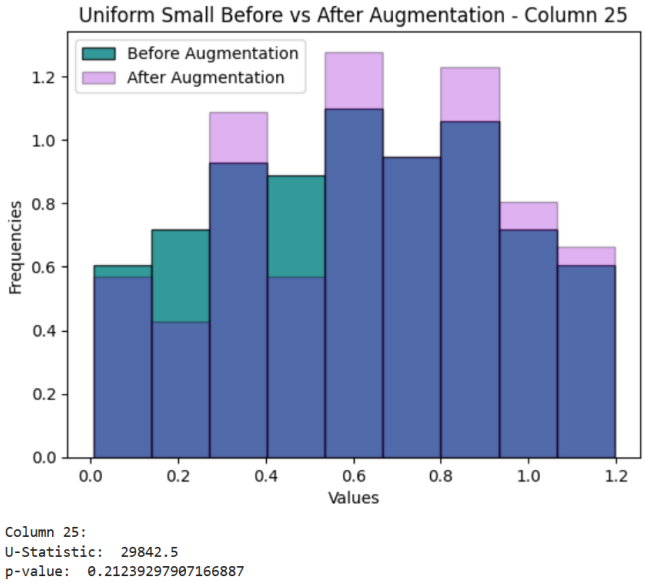


Figure 9: Example of Output from Mann-Whitney U Test

observations in subcategories. Put simply, a Euclidean norm is a measurement of distance between vectors [18]. Like the Frobenius norm, the Euclidean norm is found within the `numpy` package, but is called by `np.linalg.norm( , ord = 2)`. We want the Euclidean norm to be a small number, because this means that the proportions of observations in each subcategory does not differ significantly pre- and post- augmentation.

In Figure 10, we can see a visual example of our comparison. Just like with the Mann-Whitney U Test histogram output, the green represents the categorical proportions from the original data set and the purple represents the augmented data set, with the blue representing the overlap. Again, overlap is ideal. While the code in `Fake_Differences` does not output this visual, it is helpful in understanding conceptually how we conducted this comparison. However, it does output the subcategorical proportions for each categorical column in arrays and calculates the relative error of the Euclidean norm of the difference between the arrays. This output can be see above the bar plot in Figure 10. This is what is displayed for each categorical column when the `Fake_Differences` function is run.

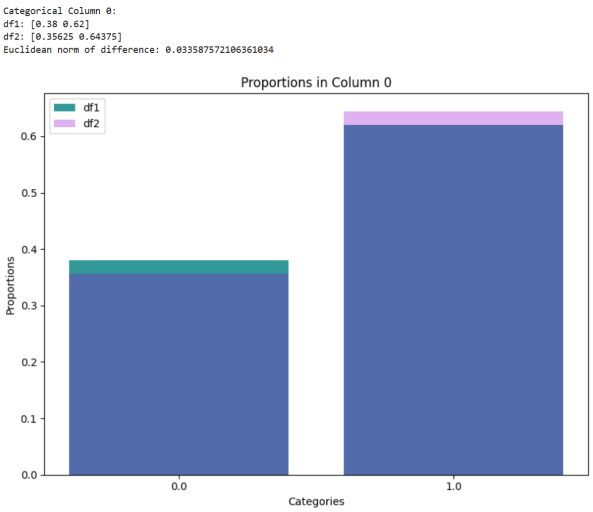


Figure 10: Example Output for Subcategorical Differences

5 Preliminary Results

5.1 Synthetic Data Sets for Experiments

We created three synthetic data sets, each following a different distribution: Gaussian, uniform, and mixed. The Gaussian data set has 172 rows and 13 columns. A scatter plot of this data set is shown on the left side of Figure 11. It can be seen that there are two classes in this data set, one represented by the red points and the other represented by the blue points. Next, there is the uniform distribution which has 343 rows by 25 columns. The other scatter plot in Figure 11 displays the uniform distribution. Again, there are two classes in this distribution, represented by the red and blue points. Lastly, there is the mixed distribution, which was created using a combination of Gaussian, uniform, and Poisson distributions. In total, it is comprised of five smaller data sets: two following Gaussian distributions, two following uniform distributions, and one Poisson distribution.

During preliminary testing of these synthetic data sets, we classified them and determined the F1 scores of this classification process. F1 Scores are measurements of how accurately the classifiers are separating the data sets into the correct classes. Prior to augmentation, we wanted our F1 scores to be at or below 0.8 for many reasons. Of these reasons, our primary motivation was having room for the augmentation process to improve the classification, but not be so accurate that augmentation was not necessary.

We created Figures 13 and 14 by running hundreds of iterations on Gaussian and uniform data sets, respectively. Every 100 iterations, we added an additional column for the machine to learn from. Throughout this process, we kept track of the continuous average of the F1 scores up to that point. To achieve our goal of obtaining an F1 score of 0.8 or lower while retaining as much data as possible, we identified and saved the iteration and number of columns that provided the optimal balance of these parameters. For the Gaussian data set, we saved the 12th column, and for the uniform data set, we saved the 24th column.

The mixed data set has a total of 240 rows and 25 columns, though each different distribution represented has 240 rows and 8 columns, excluding the label column. The separate Gaussian data sets and the separate uniform data sets are each represented by orange and blue, as seen in the left scatter plot in Figure 12. Whereas the blue Gaussian distribution is tightly clustered, the orange Gaussian distribution is more scattered. Likewise, in the comparison of the uniform distributions in the middle scatter plot in Figure 12, the orange distribution is more tightly clustered and the blue distribution is more spread out. In the Poisson distribution, all points represented are integers. Where the color is darkest in the right scatter plot in Figure 12, more points are clustered. Similarly, where the color is lightest, less points are clustered.

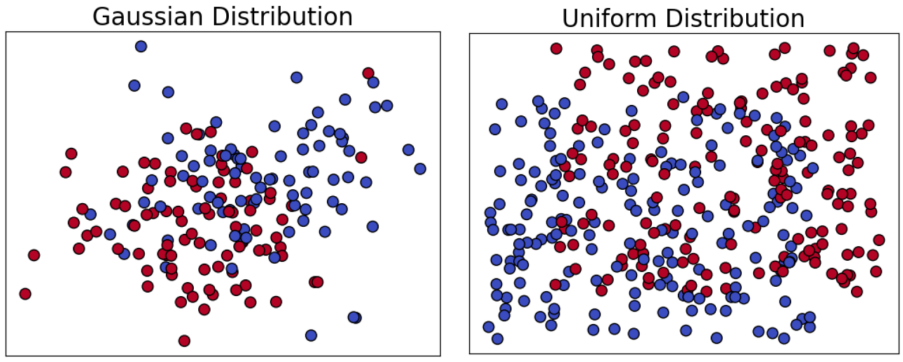


Figure 11: Plots of the Gaussian and Uniform Data Sets

Following the creation of the three original data sets, we used the same parameters to generate new random data sets. This method did not create duplicate data sets, but rather similarly distributed data sets. This newly collected data sets were concatenated to the original data sets to simulate "collecting" more data to add to the data sets. Additionally, we used the augmentation methods discussed above - modPMOne, randSwap, and HAT - to augment each of the three original data sets. Thus, in the end, we have nine total data sets we are working with in our

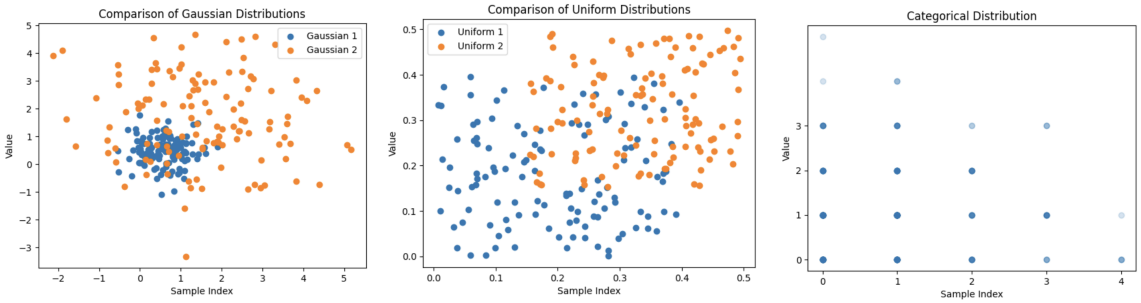


Figure 12: The Three Distributions Included in the Mixed Data Set

experiments - the three original data sets, the three original + newly collected data sets, and the three original + augmented data sets.

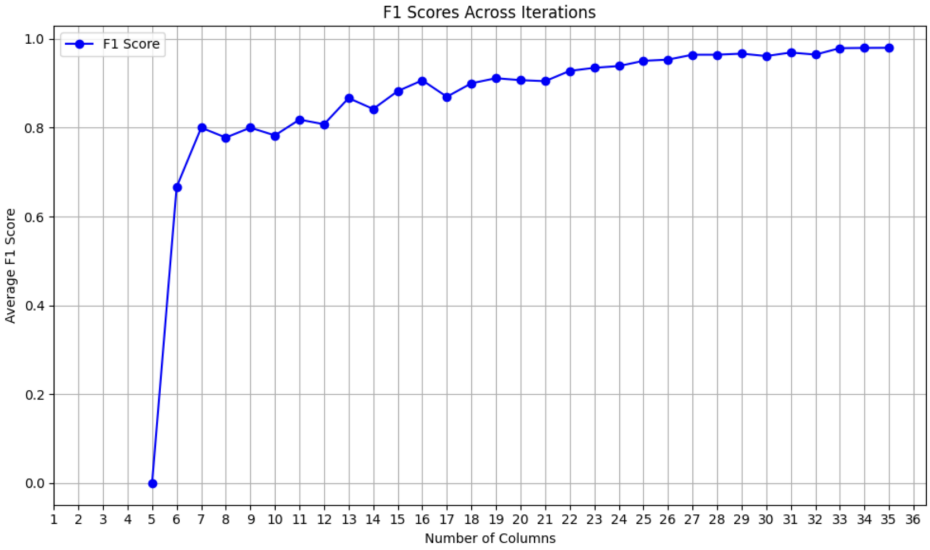


Figure 13: Gaussian Line Plot

5.2 Experimental Design

The purpose of our experiments utilizing synthetic data sets was to compare collecting more real world data versus simply applying a data augmentation method. This allowed us to see if augmenting a data set yields similar results to collecting new data. Simulating newly collected data creates baseline metrics for us to fully understand to what extent the data augmentation methods maintain patterns. Figure 15 depicts our experimental design, with the first row showcasing the different synthetic data sets we used for our experiments, as discussed previously.

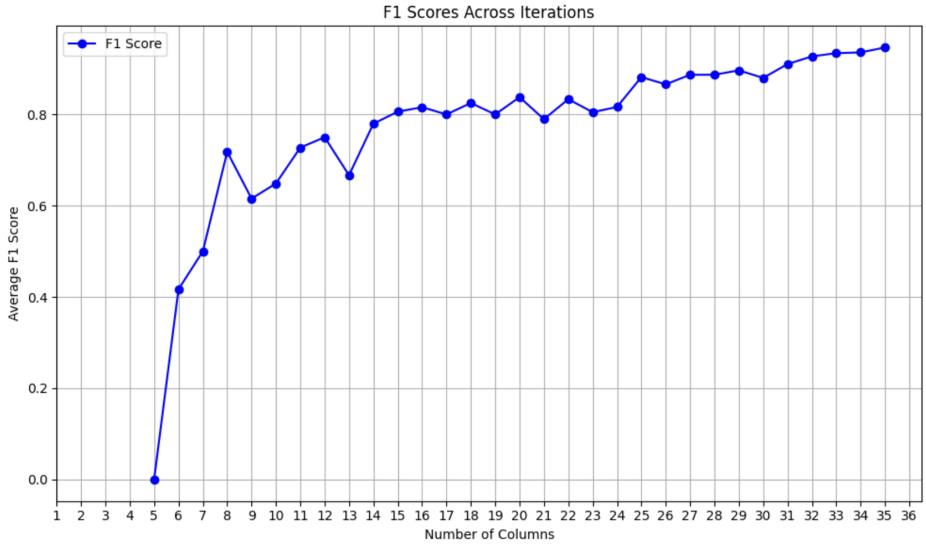


Figure 14: Uniform Line Plot

We begin with an original synthetic data set, as shown in yellow, and then sample from the same distributions to create our newly collected data set, shown in teal, as mentioned above. We then run the original data sets and the newly collected data sets through the function `Fake_Differences` to compare the patterns in the two data sets and determine baseline metrics for how much data patterns change when collected data is added to the original data sets.

Then, using the same original data set, we apply a data augmentation method to create a separate augmented data set shown in pink. Then, the original data set and the augmented data set are compared through `Fake_Differences` to see how much the patterns change when the original data is augmented. We can then compare these results to the baseline metrics. We can do this with all three of the original data sets. This allowed us to see how much we could expect patterns in the data sets to change when real data is collected versus when augmented data is added for the different distributions. Although not explicitly stated in the experimental design, we concatenated the original data set with the newly collected and augmented data sets respectively before fully evaluating their patterns.

5.3 Original vs Newly Collected Data

First, we will discuss the results for comparing the patterns in the original and newly generated data when running both data sets through the `Fake_Differences` function. The results

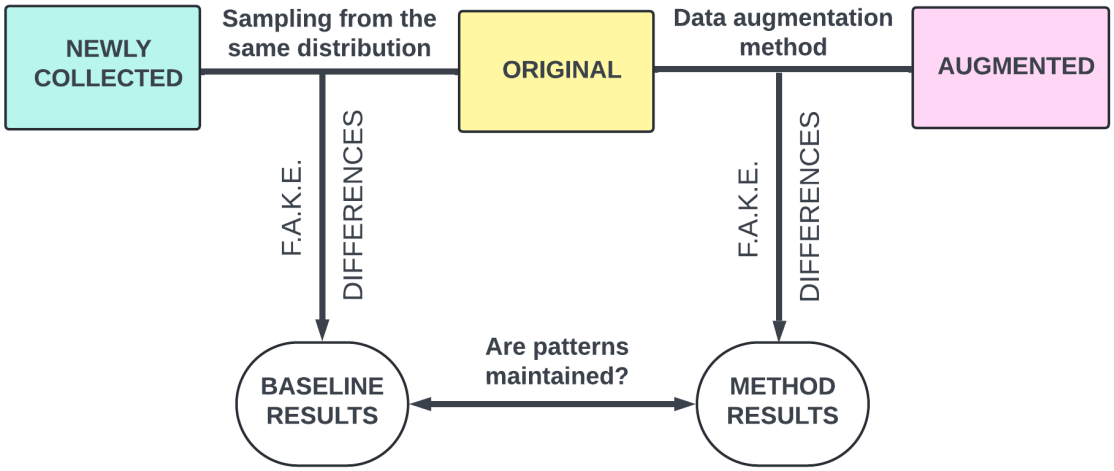


Figure 15: Experimental Design

for the tests are as follows for the feature-to-feature, feature-to-label, and feature distribution experiments. In comparing both data sets, we utilized 13 features for all data sets regardless of the distribution which have indexes ranging from 0 to 12, including the label column. The reason that we had to subset each of the data sets to include only 13 columns is because we compared each of the three original data sets to each of the three newly collected data sets. The purpose of this was to not only find baselines when comparing the original data sets to the newly collected data set of the same distribution type, but also to see baselines for "bad" changes in patterns. For example, we expect to see small changes when comparing the original Gaussian to the newly collected Gaussian, but we would expect to see "bad" results, or large changes, when comparing the original Gaussian to the newly collected uniform.

As previously mentioned, the Gaussian data set has dimensions of [172 x 13] with categorical columns at indices [2, 3, 7, 9, 12]. The uniform data set has dimensions [343 x 25] where the categorical columns are at indices [2, 7, 10, 15, 24]. As this data set was too large, we subset it to 13 features while maintaining the specified categorical features and populating the rest with random numerical features in the data set. Additionally, for the mixed data set, the dimensions were [240 x 25] with categorical features in columns [16, 17, 18, 19, 20, 21, 22, 23, 24]. However, since we needed to subset, we only selected features [16, 17, 18, 19, 24] as categorical and populated the rest with numerical features similarly to the uniform data set.

An important aspect to mention is that our primary objective was to compare the original data and the newly generated data. These data sets form our baseline, which is the most crucial part of our experiments. The baseline ensures that we can determine whether we are maintaining the patterns in the data set. We aimed to evaluate their performance across different distributions by crisscrossing them in our comparisons, as mentioned previously. It is essential to note that we expected the smallest changes to appear in the diagonal rows, indicating minimal differences when comparing features within the same distribution. These diagonal, bold values represent our baseline in the preliminary results tables below.

In the feature-to-feature experiments, we conduct two test to examine the differences in patterns. The first measuring tool involves correlation matrices differences, which calculates the relative error of the Frobenius norm between two matrices as shown in Table 2. Meanwhile, the second measuring tool is the Chi-Squared which outputs the count of changes in significance. This allows us to identify and analyze the transitions between true (non-significant p-value) and false (significant p-value) outputs, and vice versa, and count the number of changes in data patterns for each experiment, as shown in Table 3. Therefore, for both measuring tools, the results exhibit the smallest changes in significance are found along the diagonal, as we expected.

	Gaussian Original	**Uniform Original**	**Mixed Original**
Gaussian Original + New	**0.099**	0.257	0.225
Uniform Original + New	0.215	**0.105**	0.252
Mixed Original + New	0.218	0.275	**0.114**

Table 2: Relative Error in Correlation Matrices using Frobenius Norm (Feature-to-Feature)

	Gaussian Original	**Uniform Original**	**Mixed Original**
Gaussian Original + New	**0**	4	12
Uniform Original + New	4	**0**	12
Mixed Original + New	10	10	**2**

Table 3: Count of Changes in Chi-Squared Significance (Feature-to-Feature)

For the feature-to-label comparison, we apply the same logic used in the feature-to-feature experiments. However, in this case, each feature in the data set is compared with the label column. This approach allows us to assess how individual features relate to the target variable, providing insights into the predictive power and relevance of each feature in the context of the label.

Table 4 shows the results of the count of changes in significance when performing the

Chi-squared test for feature-to-label. The same logic applies here: we expect the smallest count of changes to appear in the diagonal rows, indicating minimal differences when comparing data sets with the same distribution. Although the diagonal values are the smallest, there are also other rows with similarly small values. This could be due to the inherent similarities between certain features across different data sets or the presence of features that are less sensitive to changes.

	Gaussian Original	**Uniform Original**	**Mixed Original**
Gaussian Original + New	**0**	0	4
Uniform Original + New	0	**0**	4
Mixed Original + New	4	4	**0**

Table 4: Count of Changes in Chi-Squared Significance (Feature-to-Label)

Subsequently, the goal with feature distribution comparison is to compare the spread of values within specific columns of two data sets. To verify this, we utilize measuring tools such as the Mann-Whitney U Test for numerical columns, and the subcategory proportions differences measurements for categorical columns. Just as before, the first set of experiments for feature distribution was designed to find the baselines for these comparisons.

Table 5 shows the results from the Mann-Whitney U Test for the original vs newly collected data sets. The results show the count of numerical columns that output significant p-values. As a reminder, a significant p-value for the Mann-Whitney U Test means that the null hypothesis - which states that the distributions are the same - cannot be rejected. Because this is what we want, we hope to see small numbers in the results tables for this test. The numbers in Table 5 range from 0 to 8, which is the total number of numerical columns in each data set for this set of experiments. It can be seen that when we are comparing the original data sets to their same type of distribution, the numbers are zeros. Additionally, when comparing different distributions, all 8 of the numerical columns were significantly different. This is what we expected to see. Therefore, these results indicate that the Mann-Whitney U Test is performing correctly.

	Gaussian Original	**Uniform Original**	**Mixed Original**
Gaussian Original + New	**0**	8	8
Uniform Original + New	8	**0**	8
Mixed Original + New	8	8	**0**

Table 5: Significant p-values for Mann-Whitney U Test

Furthermore, the results of differences in subcategory proportions for categorical columns can

be seen in Table 6. In Table 6, the numbers that are displayed are the relative error of the Euclidean norm of the difference between the subcategory proportions. Specifically, in the table, the largest Euclidean norm out of all of the categorical columns is shown. This means that all of the categorical columns in the data set had Euclidean norms of differences that were that size or smaller. In this table, we want the numbers to be small, because that would mean there is a small difference in subcategory proportions between categorical columns in the original data set and the newly collected data set. Along the diagonal values in the table, which are in bold text, we can see that, just like with the Mann-Whitney U Test, comparing like distributions yields the smallest difference in subcategory proportions, which is what we expect to see. Likewise, we expect that we get higher Euclidean norms of differences when comparing different distribution types, which is also what we observed.

	Gaussian Original	**Uniform Original**	**Mixed Original**
Gaussian Original + New	**0.138**	0.411	0.557
Uniform Original + New	1.206	**0.029**	0.328
Mixed Original + New	0.887	0.186	**0.100**

Table 6: Relative Error of Categorical Proportions using Euclidean Norm

5.4 Original vs Augmented Data

After experimenting with the newly generated data, we decided to augment the original data set using different augmentation methods: HAT, modPMOne, and randSwap. It's important to note that these experiments were conducted differently from the previous ones. In these experiments, we compare the original and augmented data sets against the baseline. Consequently, when experimenting with the augmented data, we did not need to perform any subsetting and kept the dimensions of the data sets unchanged.

The most critical aspect is comparing the original vs augmented results to the baseline, which is the second now in the table. This comparison is essential because we want the values in the table to closely align with the baseline. If this alignment occurs, it indicates that the augmentation methods used produce similar results as collecting more data, so are thus successfully preserving the patterns from the original data set.

When performing feature-to-feature comparison using the original vs augmentation results and the baseline, we anticipate exciting findings. The first preliminary result table shows the relative

error of the correlation matrices measuring tool. As illustrated in Table 7, the HAT augmentation method maintained the patterns the best, while the Random Swap method preserved the patterns the least, as evidenced by the larger relative error values in that row. This key takeaway highlights the effectiveness of HAT in preserving data patterns between numerical variables compared to other augmentation methods.

	Gaussian Original	**Uniform Original**	**Mixed Original**
Original + New	**0.099**	**0.105**	**0.114**
Augmented with modPMOne	0.151	0.134	0.145
Augmented with randSwap	0.225	0.274	0.341
Augmented with HAT	0.082	0.124	0.116

Table 7: Relative Error in Correlation Matrices using Frobenius Norm (Feature-to-Feature)

The preliminary results for the Chi-squared test, which counts the number of changes in significance of dependence between categorical variables, reveal interesting findings as shown in Table 8. The baseline shows values of zero except for the mixed data set. When comparing the baseline with the original vs augmented data set results, the mixed data set exhibits the most changes, ranging from 2 to 60. Notably, the HAT method maintained the patterns the least, as it also shows a high count of changes, ranging from 10 to 60. This is the opposite of what we saw for relationships between numerical variables.

	Gaussian Original	**Uniform Original**	**Mixed Original**
Original + New	**0**	**0**	**2**
Augmented with modPMOne	8	6	28
Augmented with randSwap	0	6	8
Augmented with HAT	10	10	60

Table 8: Count of Changes in Chi-Squared Significance (Feature-to-Feature))

Moving on to the feature-to-label comparison, we expected smaller changes for the Chi-squared test. Table 9 shows no changes across the Gaussian distribution, indicating perfectly conserved patterns for all augmentation methods. In contrast, the mixed distribution shows the most pattern changes, with the modPMOne and HAT augmentation methods exhibiting the highest count of changes. Again, the HAT augmentation method performed the worst for maintaining relationships between categorical variables and the label column.

In addition, we want to compare the numerical distributions of the original data sets to the augmented data sets to see how much they changed. Again, the results for the Mann-Whitney U Test outputs the number of numerical columns that had significant p-values, so we want the

	Gaussian Original	Uniform Original	Mixed Original
Original + New	**0**	**0**	**0**
Augmented with modPMOne	0	5	9
Augmented with randSwap	0	0	2
Augmented with HAT	0	0	9

Table 9: Count of Changes in Chi-Squared Significance (Feature-to-Label)

numbers to be low. When looking at feature distribution changes for numerical variables for the original vs augmented data sets, we obtained precisely what was expected: no changes in patterns. As seen in Table 10, when we employ the Mann-Whitney U Test to compare the original data set to all three augmented data sets, all the values are 0. This is what we want to see, because it means that, when using all three augmentation methods on the three different distributions, none of the numerical columns' distributions were significantly altered. This means that the augmentation methods are successful at maintaining patterns in numerical distributions.

	Gaussian Original	Uniform Original	Mixed Original
Original + New	**0**	**0**	**0**
Augmented with modPMOne	0	0	0
Augmented with randSwap	0	0	0
Augmented with HAT	0	0	0

Table 10: Count of Numerical Columns with Significant p-values using Mann-Whitney U Test

In Table 11, the results of the subcategory proportion differences are shown. As a reminder, for subcategorical proportions, the values shown represent the highest Euclidean norm of the difference in the subcategorical proportions for that comparison of data sets. For the Gaussian and mixed distributions, it can be seen that modPMOne was the worst at maintaining the patterns, as it has the highest relative error. Conversely, in the uniform distribution, this is not the case, as it maintained the patterns the most. For the Gaussian distribution, both HAT and randSwap maintained patterns even better than when data was "collected," and HAT performed almost as well as "collecting" data for the mixed distribution. Overall, most relative error values were relatively close to the baseline, meaning the augmentation methods were mostly effective in maintaining patterns in categorical distributions.

	Gaussian Original	Uniform Original	Mixed Original
Original + New	**0.138**	**0.029**	**0.100**
Augmented with modPMOne	0.387	0.046	0.283
Augmented with randSwap	0.107	0.058	0.098
Augmented with HAT	0.106	0.060	0.102

Table 11: Relative Error of Categorical Proportions using Euclidean Norm

6 Challenges & Limitations

One of the first challenges we encountered during this project was the selection of four real-life data sets. To establish a robust and clear foundation, our data sets needed to be neither too easy nor too cumbersome to classify. This was to ensure that we could effectively compare the accuracy of classical Machine Learning (ML) approaches with data augmentation methods. If putting the data set through a classifier alone produced an accuracy of 98% and above, it would have been challenging to determine how well different data augmentation methods improved accuracy. Simultaneously, we aimed to avoid using complex classifiers that required substantial adjustment outside of data augmentation to produce a decent accuracy rate. Our goal was to find data sets that lent themselves to comparing the differences in accuracy between the classical ML approach and the application of different data augmentation methods. Although we did not end up using our data sets in our experimentation, they were used to determine baselines for what an F1 Score might look like before augmentation, as discussed in Section 2.4.1. We also ran our data sets through the `Fake_Measures` function to ensure that the function worked correctly.

Second, we needed to ensure that augmentation methods did not lead to over-fitting or under-fitting, as portrayed in Figure 16 [19]. In the example below, our model identifies two classes of circles and crosses. The first of the three images portrays an instance of under-fitting, where the model cannot distinguish between the classes in any way and has very low accuracy. The rightmost image displays over-fitting, wherein the model can correctly predict between the circles and exes in the training data. However, it performs poorly on other data sets because it is too closely tailored to the training data. Appropriate fitting is shown in the middle image, where, although the classes aren't perfectly separated, the model mostly separates them, and is not too tailored to this specific data set that it would not work with a different data set.

Overall, data augmentation performs only as well as the original data set. Any relationships within the initial data should also be reflected in the augmented data. Due to this, preserving any

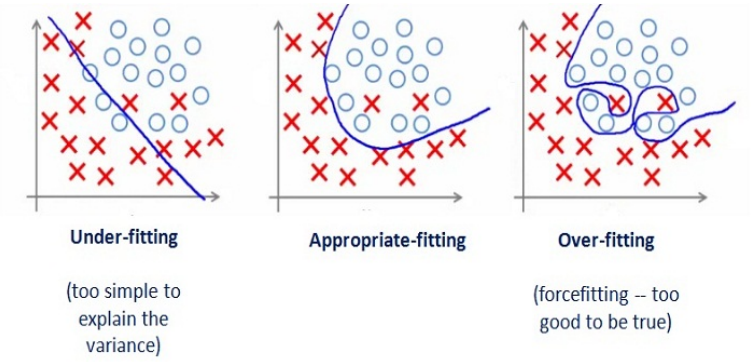


Figure 16: Over-Fitting vs. Under-Fitting

data patterns present in the original data set is imperative, which is easily done with image data but not so much tabular data. This is why preserving the patterns of tabular data was the main focus of this project.

6.1 Rejected Methods

6.2 Synthetic Minority Over-Sampling Technique (SMOTE)

Pedro Filipe Costa Machado's thesis on data augmentation methods for tabular data served as a starting point for us to explore additional data augmentation methods [20]. His thesis explores the use of different sampling methods and adversarial networks to augment tabular data, including Synthetic Minority Over-Sampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN), Gaussian Mixture Models, Clustering, Variational Autoencoders, and Generative Adversarial Networks (GANs). We got the idea to use SMOTE from his work and tested it with our evaluation framework.

SMOTE generates additional data points by creating new samples from the minority class to balance the data set. To begin, SMOTE selects a data point from the raw data's minority class, as depicted in step 1 of Figure 17 [20]. Next, the user selects the number of nearest neighbors to use for classification of the selected data point. This classification technique is known as K-Nearest Neighbors (KNN). In step 2 of the figure, the data point selected from the original data set is depicted in yellow and its three nearest neighbors are in red. Of those neighbors, one is randomly chosen and a synthetic data point is created between the selected data point, depicted in yellow, and the randomly chosen point, depicted in red.

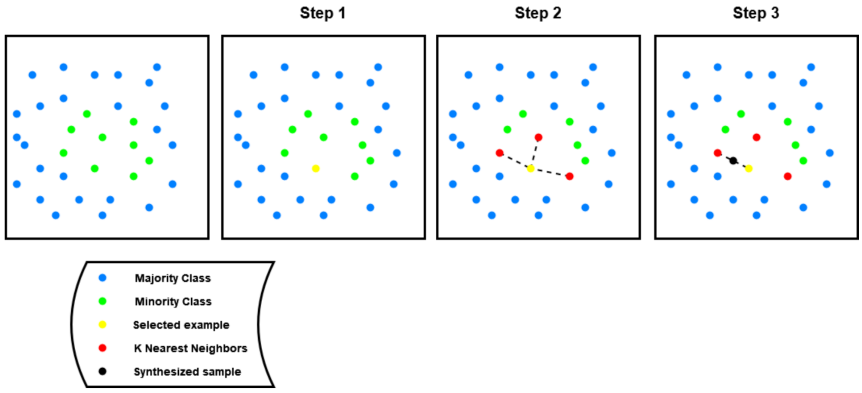


Figure 17: Synthetic Minority Over-sampling Technique (SMOTE)

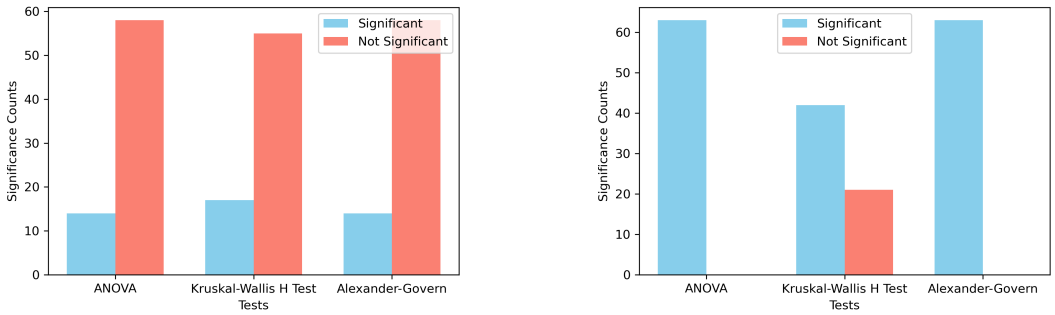
It is recommended to under-sample the majority class alongside applying SMOTE to more easily balance out the classes in a data set [20]. However, one analysis of SMOTE and other sampling techniques has revealed that this can sometimes present issues of over-fitting and go as far as misrepresenting the the minority class's original distribution [21]. Although we attempted to evaluate this method, due to time constraints and the need for code adjustment, we were unable to include it in our final experiments.

6.2.1 ANOVA, Kruskal-Wallis H Test, & Alexander Govern Test

There were several attempts to incorporate a diverse range of tests into the `Fake_Measures` function, ranging from parametric to non-parametric tests. For feature-to-feature and feature-to-label comparison, we reviewed analysis of variance (ANOVA), the Kruskal-Wallis H (K-WH) test, and the Alexander-Govern (A-G) test. Vargha and Delaney offer some insight on the interchangeability of the ANOVA and K-WH test in cases of stochastic homogeneity [22] while the Alexander-Govern test is a method that was developed as a form to cope with and outperform ANOVA in the cases of variance heterogeneity [23]. For initial experimentation, we ran the ANOVA on the mixed synthetic data we crafted to test our function. It provided an overwhelming turnout of statistical significance for our comparisons, which led us to examine it in greater detail and survey its robustness to parametric and non-parametric alternatives. We briefly incorporated the Friedman χ^2 test into our experiments, but discarded it early on due to our need for a two-sample test.

To begin our test comparison, we ran all three tests using each of the 8 Poisson columns as our

label which produced the following outputs found in Figure 18. We compared each Poisson 'label' in the data to individual Gaussian and uniform columns. As we can see in Figure 18(a), the tests determined the comparisons between the Gaussian columns and Poisson labels to be non-significant, as expected. However, when comparing our Poisson labels to our uniform columns, all tests except the K-WH test determined that there were statistically significant differences between them. Given that the Poisson labels are randomly generated, we expected non-significant determinations for the majority of columns across both the Gaussian and uniformly distributed data.



(a) Gaussian Columns & Poisson Label

(b) Uniform Columns & Poisson Label

Figure 18: Test comparisons of ANOVA, K-WH, & A-G using mixed data

We created a few scatter plots of the columns that the K-WH test determined to be statistically significant and non-significant to analyze its robustness. We plotted Gaussian columns along the axes of Figures 19 (a) and (b) and Uniform columns along the axes of Figures 19 (c) and (d). The K-WH test was unable to provide consistent results for the uniform columns and Poisson labels to the extent that it provided a 'significant' determination for a combination of uniform and Poisson labels that had no statistically significant differences as seen in Figure 19 (d) and vice versa.

In order to further investigate the issue, we decided to try and create binomial 'labels' to compare the tests. New synthetic data for individual Gaussian, uniform, Poisson and binomial samples, respectively n=500, were generated while each test was run for at least 150 loops. All tests determined statistically significant differences across all combinations of binomial and Poisson labels with the exceptions of the K-WH test in the case of uniform columns and Poisson labels, as seen in Figure 20.

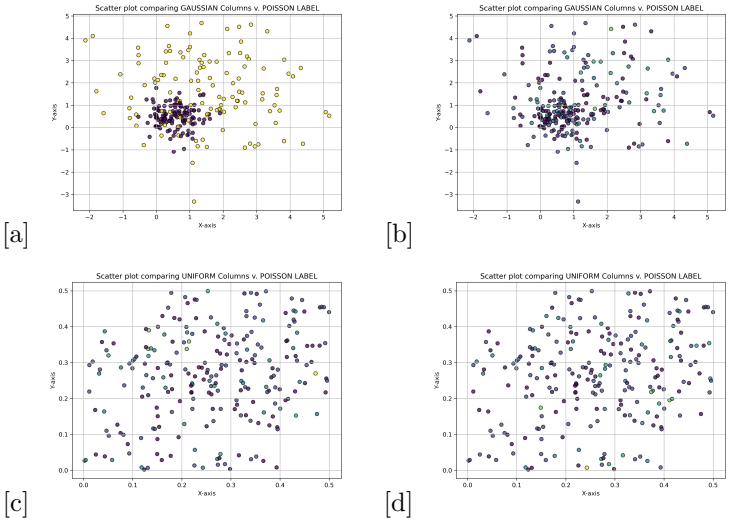


Figure 19: (a) Gaussian & Poisson: 'SIGNIFICANT' (b) Gaussian & Poisson: 'NOT SIGNIFICANT' (c) Uniform & Poisson: 'SIGNIFICANT' (d) Uniform & Poisson: 'NOT SIGNIFICANT'

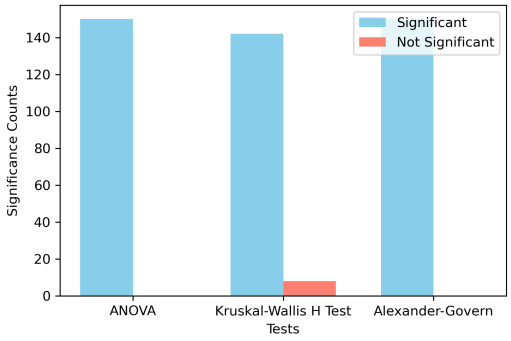


Figure 20: Test comparison using newly generated Uniform & Poisson columns

Next, we dropped our binomial and Poisson 'labels' altogether and generated one randomly. Random indices were pulled from [0, 249] and [250, 500] and were assigned a value of 1, leaving the remaining indices with a value of 0. We created a label column from these values alongside generating two additional columns sampling from the Gaussian and uniform distributions. We ran the ANOVA and K-WH test on this data up to 150 times, generating new data inside the loop each time. All of the tests returned statistically significant outputs for each iteration.

We manually calculated the means for the generated columns individually, which provided a 'significant' determination for all tests, before running the tests several times in a loop with the generated data. All tests determined statistically significant differences between the Gaussian &

uniform columns and our random labels. Due to this, we decided to drop the ANOVA test altogether as we didn't find it ethical to incorporate it into our framework given the diversity of the data sets that may be input to our function.

6.2.2 FIC Classifier

We are rejecting the Feature Importance Classifier (FIC) as a classifier due to its consistently low accuracy scores, which were 0 for half of our data sets, as shown in Table 12. Despite its intended approach of improving classification accuracy by emphasizing the importance of individual features, the performance metrics indicate otherwise. FIC ranks features using a Chi-squared test and combines weak learner models through weighted voting based on feature importance [8]. However, in our evaluations, this method did not yield the expected improvements. The low accuracy scores suggest that FIC fails to effectively distinguish between relevant and irrelevant features, resulting in poor classification performance. Therefore, we have concluded that FIC is not a suitable classifier for our needs.

	Softball	**Arrests**	**Vehicles**	**WVS**
FIC	0.40	0	0	0.92

Table 12: F1 Scores of FIC on Real-World Data Sets

Additionally, using FIC, we encountered an error indicating that it could not process negative values from the columns. To address this issue, we attempted to drop the columns containing negative values. Despite this effort, the FIC classifier continued to malfunction and did not operate as expected. This persistent problem suggests that the root cause might not solely be the presence of negative values but potentially other underlying issues within the data set or the classifier's configuration. Further investigation into the classifier's requirements and the data set's characteristics is necessary to identify and resolve the problem effectively.

6.3 Ethics

6.3.1 Issues

We sought to ensure that our methods were ethical over the course of their development. We recognize that there are many considerations to keep in mind throughout each step of the data augmentation process. The usage of our data augmentation methods with some data sets may

present unintended challenges, such as the enhancement of bias or data privacy concerns. While it is possible to encounter such challenges, it is in no way the intended use of our methods. We strongly urge extensive vetting of the data sets used with our methods, as well as a close examination of how the data augmentation process may have changed the implications of one's conclusions.

In maintaining the patterns present in a data set throughout augmentation, any bias present initially may be reflected or even enhanced. Furthermore, as many data sets requiring augmentation are limited in their initial capabilities, there may be pressing data privacy concerns. We find it particularly important to ensure that vulnerable populations such as minors, differently abled persons, and people of color are not harmed by the employment of our methods.

6.3.2 Potential Solutions

In conducting our research, we endeavored not to distort any patterns in the data sets used. The data sets we chose to test our methodology with had ethical methods of data collection and no perceived unethical implications. In these data sets, particularly those whose data was collected through surveys, no personal identifiers were used in order to protect the anonymity and privacy of those involved.

When using partially augmented data sets, it is important to explicitly state that one is doing so. This complete transparency ensures one's methods are trusted and used with the right amount of caution. Augmented data and real data are not completely interchangeable, and should not be assumed to have the same level of validity and applicability.

7 Conclusion

Throughout this project, our goal was to answer the question: do current data augmentation methods effectively augment tabular data while maintaining the original data patterns? To do this, we developed comprehensive methodology for the measurement and comparison of patterns throughout the augmentation process for tabular data. This included the creation of two functions, `Fake_Measures` and `Fake_Differences`. As described in the names of the functions, `Fake_Measures` measures patterns in the data, whereas `Fake_Differences` compares the patterns of an original data set to an augmented data set to find the differences in them. Using these

functions, we are better able to answer our research question through experimentation.

There is not one simple answer to our research question that encapsulates all of the results that we found through our experiments. When it comes to maintaining patterns between numerical variables in a data set during augmentation, some of the augmentation methods were better than others. For our experiments, the HAT augmentation method performed the best out of the three that we used. This is to say that its results were closest to the baseline results from when we "collected" more data and added it to the original data set. On the other hand, the randSwap augmentation performed the worst for all data sets.

However, the interpretation of our results is different for the Chi-squared test, which was utilized for feature-to-feature and feature-to-label comparisons with categorical variables. For feature-to-feature comparison, HAT actually performed the worst and randSwap performed the best out of the three augmentation methods, which is the opposite of what we saw with the numerical feature-to-feature comparison. For feature-to-label comparison using the Chi-squared test, modPMOne performed the worst out of the three augmentation methods. In both of the Chi-squared tests, the mixed data set performed the worst out of the three data sets for maintaining patterns through the augmentation process.

Lastly, we can look at the results for comparing feature distributions - both numerical and categorical - before and after augmentation. Using the Mann-Whitney U Test to compare numerical distributions, we saw that for all three data sets, there were no significant changes in the distributions of the numerical columns before versus after the augmentation process. This was the case for the "collected" data as well as all three augmentation methods. We can also look at the results for subcategorical proportion differences for categorical distributions. These results are less straightforward, because while modPMOne performed the best for the uniform data set, it performed the worst for the other two. However, across the board, both randSwap and HAT performed well, meaning the categorical distributions remained similar during the augmentation process.

While this is not necessarily a clear answer, all of this shows that, depending on the types of patterns that are necessary to maintain in a data set, there are augmentation methods that may work better for that specific data set. We can also see, overall, that it is possible to maintain patterns through the augmentation process. While this was an assumption going into the process, we can now see from our results that it is the case. However, to make more accurate claims for

which augmentation methods are best at maintaining patterns, more experiments must be run.

8 Future Work

Future improvements to our methods could be achieved by enhancing the augmentation process to better maintain the patterns. Because it is the augmentation process that determines whether patterns in the data are kept or not, making improvements to augmentation methods would therefore improve their ability to keep the original data patterns. Additionally, it would be beneficial for a variety of real-life data sets to be tested with our methods in order to determine their efficacy across the board. We were able to test our methods on our synthetic data sets for our experimentation purposes, but it is important that these methods are all tested on real-life data sets as well. Furthermore, though we were unable in the span of our research to find a measuring tool that compares numerical and categorical variables, this would further the understanding of whether patterns were maintained. This would allow the `Fake_Measures` function to do feature-to-feature comparison between numerical and categorical variables, as well as feature-to-label comparison with numerical variables, since the label column is categorical. Adding a method that can do this would make `Fake_Measures` a more comprehensive function. In turn, this method could also be added to `Fake_Differences` to compare the changes in these relationships before and after augmentation.

References

[1] "What is data augmentation?" [Online]. Available: https://aws.amazon.com/what-is/data-augmentation/

[2] E. Dorfman, "How much data is required for machine learning?" https://postindustria.com/how-much-data-is-required-for-machine-learning/.

[3] E. T. Winn, M. Vazquez, P. Loliencar, K. Taipale, X. Wang, and G. Heo, *A Survey of Statistical Learning Techniques as Applied to Inexpensive Pediatric Obstructive Sleep Apnea Data*. Cham: Springer International Publishing, 2021, pp. 291–328. [Online]. Available: https://doi.org/10.1007/978-3-030-79891-8_12

[4] C. Dietrich, J. Roberts, J. White, and M. Vazquez, "Data augmentation for tabular data sets," 2022.

[5] Walidamamou, "Data augmentation: What are its difficulties?" 2019.

[6] Z. Keita, "Classification in machine learning: An introduction," 2022, https://www.datacamp.com/blog/classification-machine-learning.

[7] "Support vector machines." [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#support-vector-machines

[8] B. Sathianarayanan, Y. C. Singh Samant, P. S. Conjeepuram Guruprasad, V. B. Hariharan, and N. D. Manickam, "Feature-based augmentation and classification for tabular data," *CAAI Transactions on Intelligence Technology*, vol. 7, no. 3, pp. 481–491, 2022.

[9] A. A. Awan, "The curse of dimensionality in machine learning: Challenges, impacts, and solutions." [Online]. Available: https://www.datacamp.com/blog/curse-of-dimensionality-machine-learning#

[10] J. W. M. V. Christina Dietrich, Jeffrey Roberts, "Data augmentation for tabular data sets," 2022.

[11] "What is interpretability?" [Online]. Available: https://www.interpretable.ai/interpretability/what/

[12] E. W. Weisstein, "Correlation." [Online]. Available: https://mathworld.wolfram.com/Correlation.html

[13] S. Wagavkar, "Introduction to the correlation matrix," https://builtin.com/data-science/correlation-matrix: :text=A

[14] K. S. U. Libraries, "Spss tutorials: Chi-squared test of independence," 2024, https://libguides.library.kent.edu/SPSS.

[15] N. Developers, "numpy.linalg.norm," https://numpy.org/doc/stable/reference/generated/numpy.linalg.norm.html.

[16] H. Izadkhah, "Frobenius norm," *Deep Learning in Bioinformatics*, 2022.

[17] K. Perry, "Determine if two distributions are significantly different using the mann-whitney u test," 2019.

[18] T. V. Maliamanis and G. A. Papakostas, "Chapter 3 - machine learning vulnerability in medical imaging," *Machine Learning, Big Data, and IoT for Medical Informatics*, 2021.

[19] A. Kumar, "Machine learning – how to diagnose underfitting/overfitting of learning algorithm," 2015. [Online]. Available: https://vitalflux.com/machine-learning-diagnose-underfittingoverfitting-learning-algorithm/

[20] P. F. C. Machado, "Conception and evaluation of data augmentation techniques for tabular data," Ph.D. dissertation, Universidade do Minho, 2022.

[21] I. M. Alkhawaldeh, I. Albalkhi, and A. J. Naswhan, "Challenges and limitations of synthetic minority oversampling techniques in machine learning," *World Journal of Methodology*, vol. 13, no. 5, p. 373–378, 2023.

[22] A. Vargha and H. Delaney, "The kruskal-wallis test and stochastic homogeneity," *Journal of Educational and Behavioral Statistics - J EDUC BEHAV STAT*, vol. 23, 06 1998.

[23] P. J. Schneider and D. A. Penfield, "Alexander and govern's approximation: Providing an alternative to anova under variance heterogeneity," *The Journal of Experimental Education*, vol. 65, no. 3, pp. 271–286, 1997. [Online]. Available: http://www.jstor.org/stable/20152526