Final Report on

SPEEDS (Sport and Exercise Excellence Through Data Science)

Data Curation and Synthetic Data Generation

Industry Placement (STM5IPL)

Student ID: 21314174

Name: Rajesh Dhungana

Abstract

The emerging data science field has shown immense potential in the sports domain. The application of statistics on sports data can leverage complex decision-making and eventually increase the revenue of the overall spots sector. Sports analytics is such a field that synergizes statistics with sports. The availability of high quality and open-source sports data is the major hindrance in the field of sports analytics. This project aims to collect and curate high-quality sports data using cutting-edge tools and techniques. In this project, we use powerful tools like Python and R to curate and synthesize open-source and experimental sports data. During the first phase, we pre-processed and organized open-source sports data collected through various resources into an online data repository. In the second phase, we applied machine learning and deep learning techniques to create a model that can produce synthetic data. We have trained and optimized three different models using the experimental sports data obtained from the Australian Institute of Sport. The performance of each model was evaluated using statistical methods like distribution and correlation. The optimization of each of these models was done by fine-tuning various parameters. We found that the choice of model depends upon the complexity of the dataset. The GaussianCopulas based synthesizer was suitable for datasets with mild complexity whereas the CTGAN, which is a deep learning model, was suitable for more complex datasets. The PARSynthesizer was suitable for sequential datasets. We concluded that training the model with only numerical columns can enhance the data quality and reduce the computation time. This report demonstrates the application of Machine Learning and Deep Learning in the production of sports data and methods to solve the data availability challenges that are keeping sports analytics in the shade before realizing the gold it has to offer.

1 Introduction

The exponential growth of digital technology has given rise to the production of large volumes of data. The research by EARTHWEB (Wise, 2023) shows that about 3.5 quintillion bytes of data are generated per day from multiple sources. Similarly, the advancement in computing power has made it possible to harness the hidden patterns from collected data. Using various data science techniques like data analytics, data mining, machine learning, deep learning, artificial intelligence and so on we can extract meaningful insight.

Sports analytics is one such domain that uses statistics to make informed decisions based on available data. It not only covers the decision-making step but also the collection, management, production, cleaning, and many other aspects of data science. Hence, sports analytics can be coined as a branch of the broad data science field. The use of statistics in sports can be dated back to the 1980s when Bill James developed a mathematical equation to predict the runs that a team would likely earn in baseball. In 2002, Oakland Athletics general Billy Beane used statistical analysis to optimize the team, famously known as "Moneyball", which nearly won a World Series (Michael Lewis, 2016). It is considered as the breakthrough of statistics in sports. Today the use of machine learning, deep learning, and artificial intelligence has grown significantly due to their ability to process large volumes of data and make complex decisions. In 2006, Roger Bartlett reviewed the development of Artificial Intelligence (AI) over the decades and described the application of Artificial Neural Networks (ANN) for optimizing sports techniques and learning (Bartlett, 2006). Recently, many researchers are interested to see the effect of data and statistics in different sports disciplines. The research by John Warmenhoven titled "Statistics in High Performance Sport" showed the connection between sports analytics and data. Some of the notable findings of this survey are, among 90 participants 97.7% used data within their research (Waemehoven, 2022). Likewise, 88.9% used different statistical methods like exploratory data analysis and visualization (Waemehoven, 2022).

Many sports practitioners and researchers believe that statistical methods are useful to track athlete performance, improve team strategy, predict future outcomes, and take necessary actions. The machine learning algorithm can be used to identify the strengths and weaknesses of a player or whole team so that a customized training program can be developed to improve performance. The same analysis could be used by coaches to develop effective game strategies to counter specific opponents. Similarly, with the application of machine learning on historical data collected through GPS and other wearables, analysts can analyse the movement during the training session or actual match. The historical injury data could be used to predict the injury risk for an individual player so that early precautions could be considered. Off the field, sports analytics are being used to research and develop tools, wearables, and sports materials which eventually aid in performance. Likewise, sports analytics could be used in identifying and recruiting talented athletes through the examination of diverse attributes such as performance, physical strength, and mental well-being.

Despite being the fastest growing market with an annual growth rate of 5.2% and a net worth of 512.14 billion USD in 2023 (The Business Research Company, 2023), the sports domain has not been able to harness the optimum potential of the data. Although sports team can benefit heavily through data-driven decisions, many challenges are dragging the sports analytics behind before realizing its true possibility. The research conducted by John Warmenhoven and published by the Australian Institute of Sports titled "Statistics in High Performance Sports" (Waemehoven, 2022) showed that the reliability of sports analytics is only 5 out of 10. This questionable reliability is mainly due to factors

such as the quality of available data, limited knowledge of sports practitioners in the field of statistics, lack of enough time to study efficient tools and methods, and lack of open-source data. By addressing these challenges, we can bring sports analytics to the spotlight and harness the gold it has to offer.

The SPEEDS which stands for Sport and Exercise Science Excellence Through Data Science is aimed at addressing these challenges. This project is funded by the Australian Institute of Sport and managed by the Sports Data Analytics team at La Trobe University. The major aim of this project is to bridge the gap between sports data science knowledge among different sports and exercise practitioners (SPEEDS, 2023). This project is expected to play a pivotal role in the Brisban Olympics 2032 by creating an online learning platform with a high-quality sport-specific data repository, tools, and educational materials. This project is divided into three phases namely Curation, Creation, and Review. The major focus of our internship was on the first phase which is data curation. This phase played a foundation in the preparation of open-source data. It addresses the problem related to data availability and quality. With the application of tools like Python and R, we have curated open-sourced data to maintain the required quality. Different methods like data preprocessing, data wrangling, and analysis were applied in this process. Similarly, machine learning and deep learning algorithms were used to generate synthetic data which helped in addressing data availability issues.

Data Curation is the process of collecting, organizing, and managing data to ensure the required quality, usability, and replicability. It involves the process of collecting data from data sources, databases, research papers as well as many data generation methods. In this project, we have used Python and R for data formatting. We have applied statistical methods like linear regression, and random forest method during data cleaning. Linear regression is the method by which an unknown or targeted variable is predicted using a known variable or dependent variable. The linear regression aims to fit a straight line through the known variable which has the equation of form $Y = mX + c$. Using this method we have predicted numerous unknown variables in our dataset. Random forest is a machine learning algorithm based on a decision tree that combines multiple outputs of a decision tree to give a single output. In this project, we have used this algorithm to impute missing values during data preprocessing.

Machine learning is a branch of Artificial Intelligence that uses different statistical methods like linear regression, multiple regression, logistic regression, classification, and so on to learn from the data and predict unknown values. The classical machine learning algorithm uses pure statistical approaches and functions to predict the value. Whereas, in modern machine learning Artificial Neural Networks (ANN) are used to adjust the weight and make predictions. Modern machine learning techniques like Generative Adversarial Neural Networks are powerful methods in data generation. In this network structure, two distinct models namely generative and discriminative compete in generating realistic and highly accurate data. The generator which is based on ANN learns the patterns from real data and produces fake data whereas, the discriminator which is also based on ANN differentiates between real and fake data. This process of generation and assessment is continued through a feedback mechanism until the discriminator can no longer differentiate between real and fake data. At this point, we obtain high-quality synthetic data which have similar statistical properties as that of real data.

Synthetic data are fabricated data that possess similar statistical properties such as distribution, correlation, skewness, and so on as original data. The limited data availability issues have been addressed by this synthetic data generation method to some extent within in and beyond this project. The article by Conor Hassan titled “Deep Generative Models, Synthetic Tabular Data, and Differential

Privacy" provides a development of synthetic data generation using deep learning methods like GAN (Conor Hassan, 2023). Another research (James Jordon, 2018) explores the current trends in synthetic data, challenges, and methods. All these works showed that synthetic data generation is evolving rapidly and fields like sports with limited data can benefit highly.

2 Methodology

In this section, we will present the workflow of the entire data curation process and synthetic data generation. The data curation process can be broken down into four major phases. During the initial period, we were focused on data collection, data cleaning, and organization. In the latter phase, we were focused on developing, training, and optimizing machine learning models using the Synthetic Data Vault (SDV), a Python library. During the model development phase, we have used both classical machine learning algorithm based on the Copula function and modern machine learning algorithm based on deep learning (GAN). The details of each of the models will be discussed below. The high-level overview of the entire workflow is shown in the following diagram.

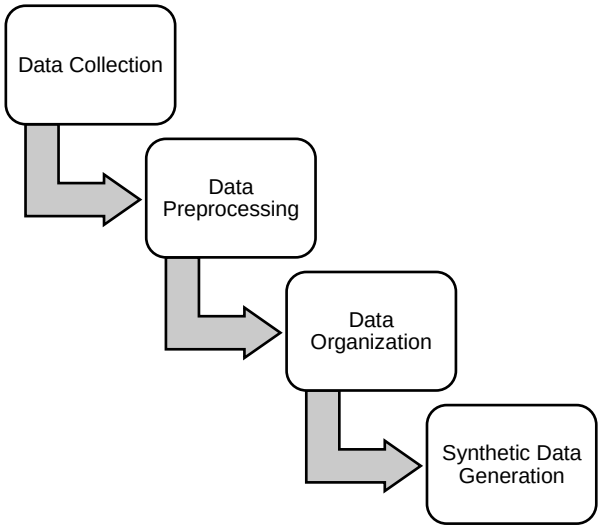
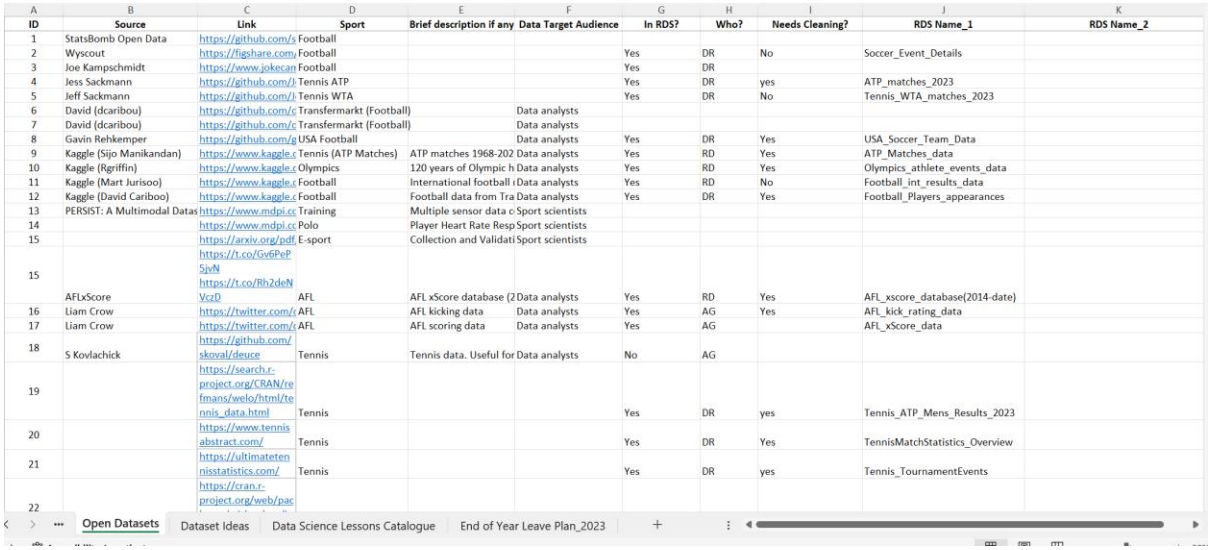


Figure 1 Data Curation

2.1 Data Collection

In this step, we referred to different online data repositories like Kaggle, Github, IBM SPSS Statistics, research websites, journals, and so on to collect open-sourced sports data. To organize the data collection task, an Excel file containing the link to a specific data source was created. The data sources were further divided based on the type of sports they belong to such as Football, Olympics, Polo, AFL, Basketball, and so on. This helped us to collect and organize open-source data into our local RDS data repository more easily. Moreover, the data file was available in different formats such as CSV, JSON, SAV, HTML, and XLSX. The raw file was then imported in Python or R to convert it into CSV format. The Pandas library in Python was used to import and change the format of these data. In Pandas either inbuilt functions like read_excel(), json.load(), or user-defined functions were used to read and convert datasets. Similarly, the dplyr and tidyr package in R was used to read and convert data into CSV format.

The CSV stands for Comma Separated Values which is a text-based format. This is a widely used format due to its simplicity, lightweightness, and compatibility. Figure 2 shows the master Excel sheet used to keep a record of all the data sources used.



ID	Source	Link	Sport	Brief description if any	Data Target Audience	In RDS?	Who?	Needs Cleaning?	RDS Name_1	RDS Name_2
1	StatsBomb Open Data	https://github.com/s	Football							
2	Wyscout	https://figshare.com/	Football			Yes	DR	No	Soccer_Event_Details	
3	Joe Kampschmidt	https://www.jokecan	Football			Yes	DR			
4	Jess Sackmann	https://github.com/j	Tennis ATP			Yes	DR	yes	ATP_matches_2023	
5	Jeff Sackmann	https://github.com/j	Tennis WTA			Yes	DR	No	Tennis_WTA_matches_2023	
6	David (dcaribou)	https://github.com/c	Transfermarkt (Football)		Data analysts					
7	David (dcaribou)	https://github.com/c	Transfermarkt (Football)		Data analysts					
8	Gavin Rehkemper	https://github.com/g	USA Football		Data analysts	Yes	DR	Yes	USA_Soccer_Team_Data	
9	Kaggle (Sijo Manikandan)	https://www.kaggle.c	Tennis (ATP Matches)	ATP matches 1968-202	Data analysts	Yes	RD	Yes	ATP_Matches_data	
10	Kaggle (Rgriffin)	https://www.kaggle.c	Olympics	120 years of Olympic h	Data analysts	Yes	RD	Yes	Olympics_athlete_events_data	
11	Kaggle (Mart Jurisoo)	https://www.kaggle.c	Football	International football i	Data analysts	Yes	RD	No	Football_int_results_data	
12	Kaggle (David Cariboo)	https://www.kaggle.c	Football	Football data from Tra	Data analysts	Yes	DR	Yes	Football_Players_appearances	
13	PERSIST: A Multimodal Datas	https://www.mdpi.cc	Training	Multiple sensor data c	Sport scientists					
14		https://www.mdpi.cc	Polo	Player Heart Rate Resp	Sport scientists					
15		https://arxiv.org/pdf	E-sport	Collection and Validati	Sport scientists					
15	AFLxScore	https://t.co/Gv6PeP5jvN https://t.co/Rh2deNVczD	AFL	AFL xScore database (2	Data analysts	Yes	RD	Yes	AFL_xscore_database(2014-date)	
16	Liam Crow	https://twitter.com/t	AFL	AFL kicking data	Data analysts	Yes	AG	Yes	AFL_kick_rating_data	
17	Liam Crow	https://twitter.com/t	AFL	AFL scoring data	Data analysts	Yes	AG		AFL_xScore_data	
18	S Kovlachick	https://github.com/skoval/deuce	Tennis	Tennis data. Useful for	Data analysts	No	AG			
19		https://search.r-project.org/CRAN/refmans/welo/html/tennis_data.html	Tennis			Yes	DR	yes	Tennis_ATP_Mens_Results_2023	
20		https://www.tennisabstract.com/	Tennis			Yes	DR	Yes	TennisMatchStatistics_Overview	
21		https://ultimatetennisstatistics.com/	Tennis			Yes	DR	yes	Tennis_TournamentEvents	
22		https://cran.r-project.org/web/pac								

Open Datasets | Dataset Ideas | Data Science Lessons Catalogue | End of Year Leave Plan_2023

Figure 2 Open-Source Data

The restricted data which was provided by the Australian Institute of Sport were collected and stored in separate folders. These restricted data are recorded by sports practitioners or researchers in a controlled environment for their research purposes. These were crucial data for this project because the synthetic data generation was performed in these restricted data. Most of the restricted data were obtained in XLSX format.

2.2 Data Preprocessing

Data preprocessing is a foundational step in data analysis in which multiple steps are taken to transform raw data into a clean and organized form. During this phase, most of the missing values and duplicated values were removed, columns were split and the data format was changed.

2.2.1 Loading and Diagnosing

In this step, the Pandas library was used to load the dataset in Python. The commonly used function is read_csv() which converts CSV into a data frame. The head() and summary() functions were used to explore the data distribution, data types of each column, structure of the dataset, and count the missing values.

2.2.2 Handling Missing Values

First, a function like isna().sum() was used to count the total number of missing values in a specific column. The beauty of Pandas library is that it automatically replaces missing values with NA in a data frame. The missing values were then plotted to select the appropriate imputing method. This was done using the vis_miss() function from the visdat package in R. Figure 3 shows a sample visualization of the missing value.

```
# Visualise missing data in the dataset
vis_miss(cycling_data)
```

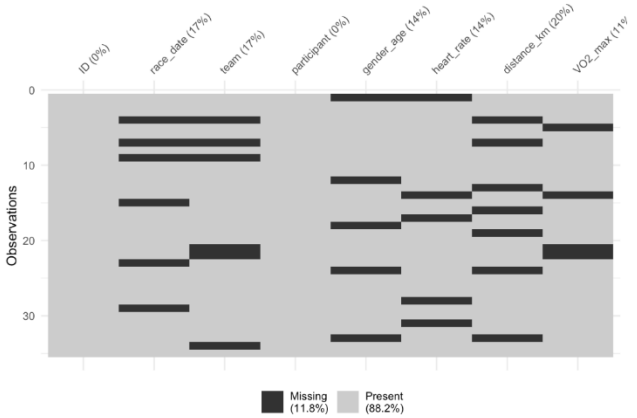


Figure 3 Missing values plot

For categorical columns mode was calculated, and NA values were replaced using mode. For numeric columns either mean or median values were used to replace the columns which have less complex distributions such as age or distance. For more complex columns with multiple correlations, the random forest method in R was used. As per our research, the random forest method was the most accepted algorithm for imputing the NA's because it not only considers the particular columns to calculate the value but also other columns in a data frame. Hence, the correlation between variables is maintained even after imputing. The mice() function offered by the MICE package in R was used to calculate the missing values. To select the best values, we calculated the range and mean for both the original column and the calculated column. Those columns whose value falls under the range of the original column and have less deviation from the mean were selected for imputing. Figure 4 shows the data distribution of column before and after imputing NA's.

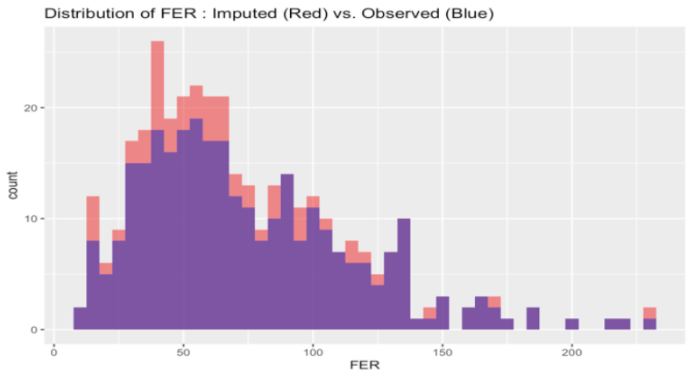


Figure 4 Distribution of reals vs imputed values

2.2.3 Data Wrangling

Data Wrangling is the process of restructuring the data frame as per the requirement of data analysis. Some of the columns in our raw dataset were complex and required splitting for better analysis. For example, the score columns consist of the scores separated by "-", which makes it difficult to analyse the actual performance. First, the white spaces were removed from the particular column, and the

split() function in Pandas was used to separate it into two or more columns as per our requirements. The split columns were first stored in a separate data frame and later concatenated in the original data frame using the concat() function. After concatenation, the original complex columns were removed. Similarly, in the restricted dataset some of the categorical columns were not important for synthetic data generation which were also removed to train our model. Figure 5 shows the data wrangling process for a particular dataset.

```
#Lets store some of the less intrested columns in a separate df data_1
data_1 = data_real[['ID', 'group', 'age', 'height', 'time', 'Crit1', 'Crit2', 'Crit1and2',
                    'Crit1or2', 'n_episodes']]
```

```
data_1.head()
```

	ID	group	age	height	time	Crit1	Crit2	Crit1and2	Crit1or2	n_episodes
0	1	4x(12x40/20s)	26	184.5	post	False	False	False	False	0
1	2	4x(12x40/20s)	24	182.0	post	True	True	True	True	3
2	4	4x(12x40/20s)	32	176.0	post	False	False	False	False	0
3	10	4x(12x40/20s)	17	171.0	post	True	True	True	True	2
4	14	4x(12x40/20s)	43	174.5	post	True	False	False	True	1

```
#Removing the categorical and other columns from the dataset which are of less interest
data = data_real.drop(columns = ['group', 'age', 'height', 'time', 'Crit1', 'Crit2', 'Crit1and2', 'Crit1or2', 'n_episodes
```

```
data.head()
```

	ID	y_train_h	y_sport	y_cyc	mass	bmi	p_fat	ffm	hr_rest	vo2_rel	...	s_fr	siga_c	siga_sr	sour_wor	symp_wor	sour_bet	symp_bet	day
0	1	409	5	5.0	73.1	21.6	17.4	61.301	52	61.2	...	105.0	244.79	25.702950	5	9	2	3	
1	2	289	3	3.0	63.2	19.1	6.3	60.400	39	71.5	...	268.6	141.37	37.971982	2	5	2	7	
2	4	401	6	6.0	68.5	22.1	12.9	60.975	45	67.3	...	104.6	160.97	18.937483	0	10	6	5	

Figure 5 Data wrangling

2.2.4 Duplicate Values

Duplicate values are an inevitable part of real-world datasets. It is one of the mandatory operations in data preprocessing. But Pandas library in Python offers a powerful and easy function duplicated () to check for any redundant values. For a row to be duplicated, values in every column must match it previous row. We have used this function to check and remove any duplicated values.

2.2.5 Dates Formatting

Date or timestamp are considered as one of the complex data structures in data analysis. There are many formats in which date or timestamp can be represented. It depends upon the location where the data was collected. Hence, we must change the dates and timestamps in our dataset to match our project standard. During the data preprocessing we used the to_datetime() function in Pandas to handle these values. The standard date format that we have used in our data set is YYYY-MM-DD HH:MM:SS.

2.3 Data Organization

After completing the data preprocessing method, it was time to organize our dataset into a separate folder for future use. All pre-processed datasets were converted into spreadsheets and a data dictionary was added to each of the files. The data dictionary is the description of dataset columns, values, ranges, and units. We want to make our data understandable and useful for non-technical users where a data dictionary will play a key role. Figure 6 shows a data dictionary for one of the datasets.

A	B	C	D	E
Variable	**Data_Type**	**Full_Name**	**Levels**	**Unit**
ID	Integer	Participant ID	1-29	N/A
group	String (categorical)	Training group	SI1: 4x(12x40/20s); SI2: 4x(8x40/20s); LI: 4x8min	N/A
age	Integer	Age	N/A	Years (y)
height	Float	Height	N/A	Centimetres (cm)
y_train_h	Integer	Training history	N/A	Years (y)
y_sport	Integer	Year sporting experience	N/A	Years (y)
y_cyc	Integer	Years cycling experience	N/A	Years (y)
time	String (categorical)	Measurement time	pre = Pre-exercise; post = post-exercise	N/A
mass	Float	Body mass	N/A	Kilograms (kg)
bmi	Float	Boyd mass index	N/A	Kilogram/metres^2 (kg/m^2)
p_fat	Float	Fat percentage	N/A	Percentage (%)
ffm	Float	Fat free mass	N/A	Kilograms (kg)
hr_rest	Integer	Resting heart rate	N/A	Beats per minute (beats/min)
vo2_rel	Float	Relative VO2peak	N/A	Litres per minute (L/min)
vo2_peak	Float	Absolute VO2peak	N/A	Millilitres per kilogram body mass per minute (mL/kg/min)
fer	Float	Serum ferritin	N/A	Micrograms per litre (ug/L)
iron	Float	Serum iron	N/A	Micromoles per litre (umol/L)
tf	Float	Serum transferrin	N/A	Grams per litre (g/L)
vitD	Float	Vitamin D	N/A	Nanomoles per litre (nmol/L)
cort	Float	Cortisol	N/A	Nanomoles per litre (nmol/L)
testo	Float	Testosterone	N/A	Nanomoles per litre (nmol/L)
e_exp	Float	Energy expenditure	N/A	Kilocalories (kcal)
e_int	Float	Energy intake	N/A	Kilojoules (kcal)

Figure 6 Data dictionary

2.4 Synthetic Data Generation

We used Synthetic Data Vault (SDV), a Python library for generating synthetic data in this project. The available dataset was divided into three different types namely Single table, Multi table, and Sequential. The Single table data are the most common type of dataset with only one table. It comes in a single spreadsheet file and doesn't contain multiple sheets or tables linked using the primary key. Unlike the Single table dataset Multi table dataset has more than two tables that are linked together using the primary key. These types of datasets are found in large databases. The sequential dataset is a complex dataset that has one or more columns of sequential type. The common type of sequential data is time series data where observation is recorded after continuous time intervals. After dividing the dataset, we selected the GausianCopulas or CTGAN for single table data and PARSynthesizer for sequential data.

2.4.1 Metadata Creation

The metadata is the dictionary of data type of each column which is used by the model to learn and generate the distribution. In SDV the numerical columns are referred to as KSComplement and categorical columns are referred to as TVEComplement. The metadata object was created using the detect_from_csv() function which automatically detects the data types from a provided CSV file. During the validation process, the auto-detected metadata were inspected and any incorrect columns were changed to the correct type using the update_colum() function. The following figure shows the metadata creation process.

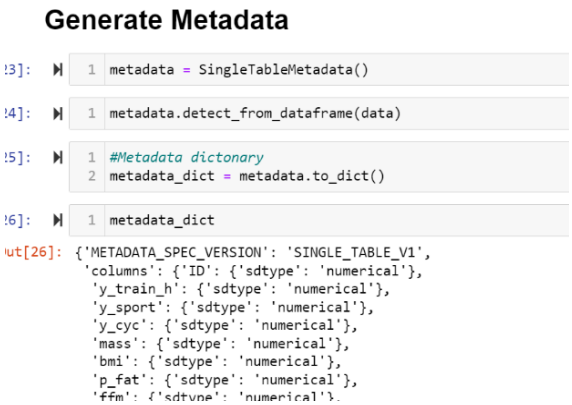


Figure 7 Meta Data generation

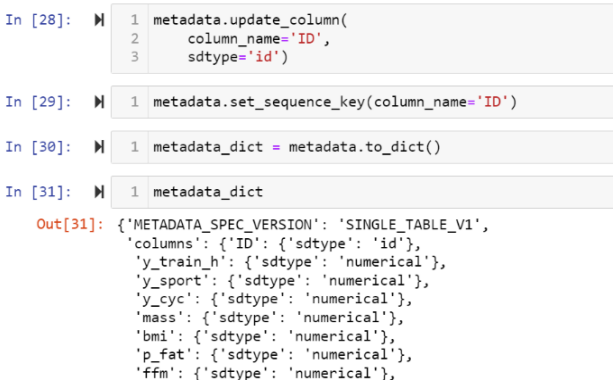


Figure 8 Metadata validation

2.4.2 Model Development

Different type of models was developed based on the machine learning algorithm that suits the dataset. The first model is the GaussianCopula Synthesizer which uses the Copula function. This function works on the principle that marginal distribution can be modelled by joint distribution (DataCebo, Inc., 2023). The second model is the CTGAN Synthesizer based on deep generative models (Lei Xu, 2019) and the third model is the PARSynthesizer based on the Conditional Probabilistic Auto-Regressive Model (Zhang, 2022).

The SDV library was used to develop each of these models. The GaussianCopula Synthesizer was developed using the GaussianCopulaSynthesizer() function with default parameters. This function takes metadata, enforce_min_max_values, enfore_rounding, and numerical_distribution as parameters. The previously created metadata object was saved into a JSON file and passed as a parameter. The enforce_min_max_values is used to generate synthetic data within the range of real data. The enforce_rounding argument is used to produce synthetic data with the same decimal places as that of real data. The default numerical distribution is beta distribution because other complex distributions could be modelled using the Copulas function. The below figure 9 shows the developed model.

```
#Creating a synthesizer model with GaussianCopulaSynthesizer
from sdv.single_table import GaussianCopulaSynthesizer

synthesizer = GaussianCopulaSynthesizer(metadata,
                                        enforce_min_max_values=True,
                                        enforce_rounding=True,
                                        numerical_distributions={
                                         'MASS': 'norm'
                                                               })
```

Figure 9 Base GaussianCopula Synthesizer

The second model was developed using the CTGANSynthesizer() function. This model takes metadata argument, which is a previously created JSON file, enforce_min_max_values set to True, enforce_rounding set to True, epochs, batch size, and so on. At first, the epochs which is the number

of iterations the model is trained, was set to 2500, and batch size, the number of rows for one iteration of training, was set to 250. The below figure shows the CTGAN Synthesizer model development.

```
# %%capture output

epochs = 2500

synthesizer = CTGANSynthesizer(
    metadata,
    enforce_min_max_values=True,
    enforce_rounding=True,
    verbose=True,
    epochs = epochs,
    batch_size=250,
    discriminator_lr=1e-5,
    generator_lr=1e-5,
    discriminator_dim=[256, 256],
    generator_dim=[256, 256]
)

```

Figure 10 Basse CTGAN Synthesizer

The third model was developed using the PARSynthesizer() function with all default parameters. The following figure shows the model development for this synthesizer.

```
#Creating a synthesizer model with PARSynthesizer
from sdv.sequential import PARSynthesizer

synthesizer = PARSynthesizer(metadata)
```

```
synthesizer.get_parameters()
```

```
t[37]: {'enforce_min_max_values': True,
 'enforce_rounding': False,
 'locales': None,
 'context_columns': [],
 'segment_size': None,
 'epochs': 128,
 'sample_size': 1,
 'cuda': True,
 'verbose': False}
```

Figure 11 Base PARSynthesizer

2.4.3 Model Training and Sampling

For training each of these models we use the synthesizer.fit() function which takes real data as an argument. During the model training process, each of these models learns the distribution of individual columns and updates its weight using the specific loss function for example GaussianCopulas in GaussianCopulas Synthesizer. After the computation is completed, the synthesizer.sample() function with one argument num_rows containing integer value was passed to generate the synthetic data. After sampling, the synthetic data was stored in a dataframe. The following figure shows the training and sampling process for sample model.

```
[ ] #Training the model with our data
    synthesizer.fit(data)
```

```
[ ] #Sampling synthetic data using the trained model
    synthetic_data = synthesizer.sample(
        num_rows=64
    )
```

```
[ ] synthetic_data.head(20)
```

	ID	GROUP	DAY	AGE	HGT	MASS	FAT	VO2_ABS	VO2_REL	PPO_ABS	PPO_REL	TRAIN_HIST	P4	E2	TEMP	FER	FE	TRANS	IL6	CRP
0	13	0	3	33	177	75	27	4.1	46	331	3.9	7	3.76	14.3	37.5	51.3	10.8	3.1	0.7	3.7
1	13	1	18	35	159	61	19	2.7	44	233	4.6	12	9.03	36.3	37.4	55.4	14.6	3.6	0.3	4.3
2	8	0	21	20	160	55	20	2.9	59	279	4.5	4	9.11	51.0	37.5	14.8	13.2	3.4	0.5	0.7
3	13	0	3	25	158	72	31	3.0	44	250	3.0	14	0.01	57.5	37.1	79.1	26.1	2.4	0.5	1.3
4	14	1	3	20	175	78	29	4.8	60	333	4.3	6	0.16	3.2	37.1	66.5	16.3	2.6	4.8	0.9

Figure 12 Model training and sampling

2.4.4 Model Evaluation

This is the final stage of our synthetic data generation process where we use different approaches to evaluate the model. There are three primary metrics which evaluate how well our model generates the synthetic data. The first approach is diagnostic where we checked the validity of synthetic data. The data structure and type of each column are evaluated by comparing it with metadata variables. For example, if we have an age column of numeric type in real data then the synthesized data must have an age column with the same numeric type.

Another approach is evaluating the statistical properties such as distribution and correlation for each of the columns. We first evaluate the correlation similarity between two columns in synthetic data. For each column, a score between 0 to 1 was produced which measures how accurately the model has maintained the correlation among different columns and compare it with the real data. Finally, we calculated the aggregate score using both distribution and correlation similarity metrics.The figure below shows the density plot and heat map to compare the data distribution and correlation between real and synthetic data.

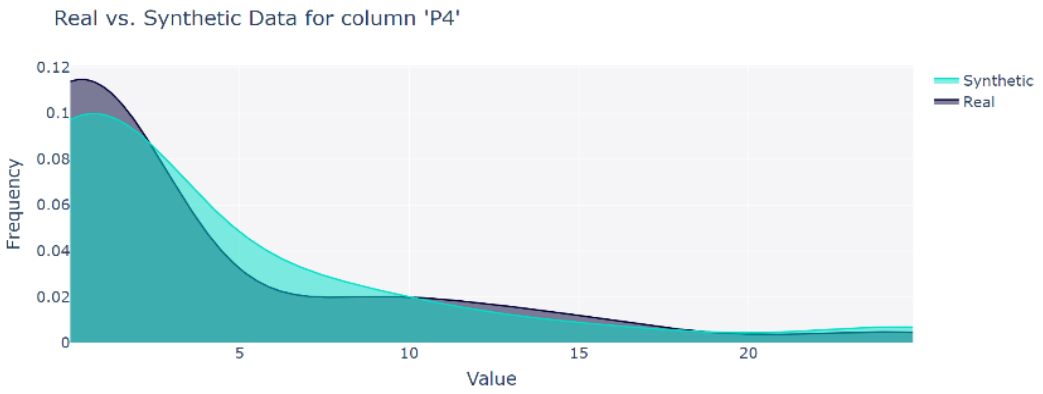


Figure 13 Distribution comparison

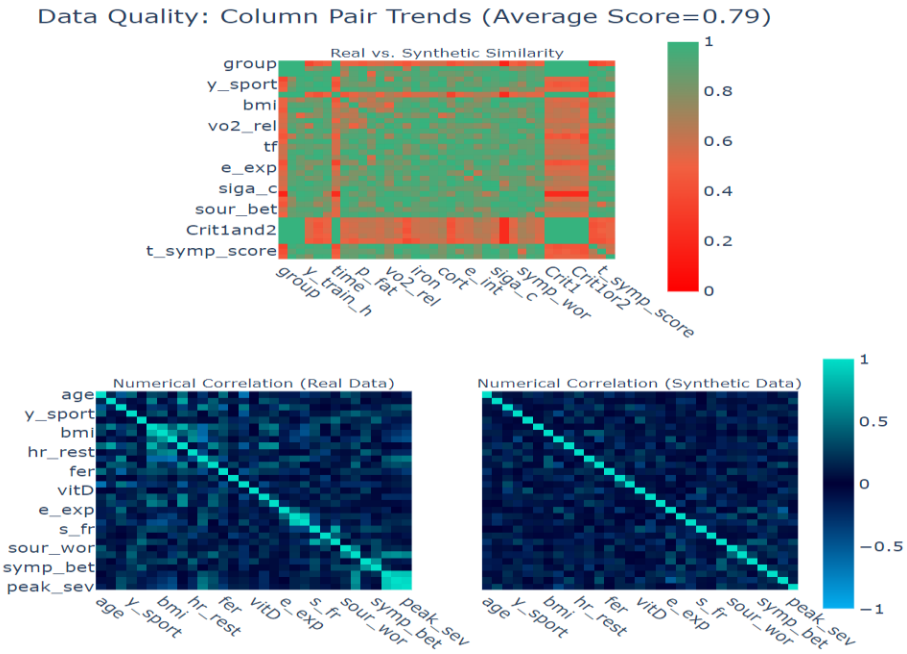


Figure 14 Correlation similarity test

3 Result and Discussion

As indicated in the methodology section, we trained three different models namely GaussianCopulas, CTGAN, and PAR Synthesizer in their default parameter settings to observe the quality of produced data. We then optimized each of these models based on their output to get better results. In the following section, we will discuss our findings for both default and optimized models in detail.

3.1 GaussianCopulas Synthesizer

First, we trained the GaussianCopulas synthesizer using the 'Badenhorst_2023_Iron.csv' data which contains 20 columns and 64 rows. Our diagnostic evaluation on synthetic data showed that all 20 columns have valid data, meaning the ID column is the key with numeric type and the generated data

falls under the min-max range of real data. Also, the synthetic data has the same column name as that of real data. The result is shown in Figure 16.

```
Generating report ...
(1/2) Evaluating Data Validity: : 100%|██████████| 20/20 [00:00<00:00, 595.12it/s]
(2/2) Evaluating Data Structure: : 100%|██████████| 1/1 [00:00<00:00, 126.40it/s]

Overall Score: 100.0%

Properties:
- Data Validity: 100.0%
- Data Structure: 100.0%
```

Figure 16 Diagnostic Result on GaussianCopulas

We investigated the marginal distribution of each column on synthetic data and compared it with the real data. The result showed that most of the distribution was in the acceptable range above 0.8 except for the 'IL6' column which was only 0.5. The density plot below (figure 17) shows the comparison of distribution between real and synthetic for this column. The real data has the truncated normal distribution, and the model has failed to capture the min-max range.

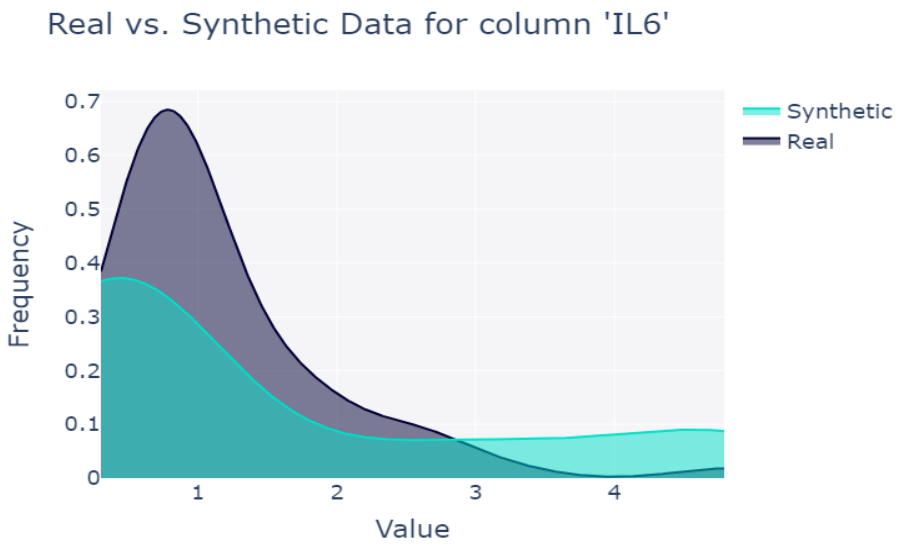


Figure 17 Density Plot for IL6

Similarly, we compared the correlation similarity between the MASS and PPO_REL columns using the scatter plot. Figure 18 compares the correlation between real and synthetic data. This model has preserved a strong negative correlation for the MASS column in synthetic data.

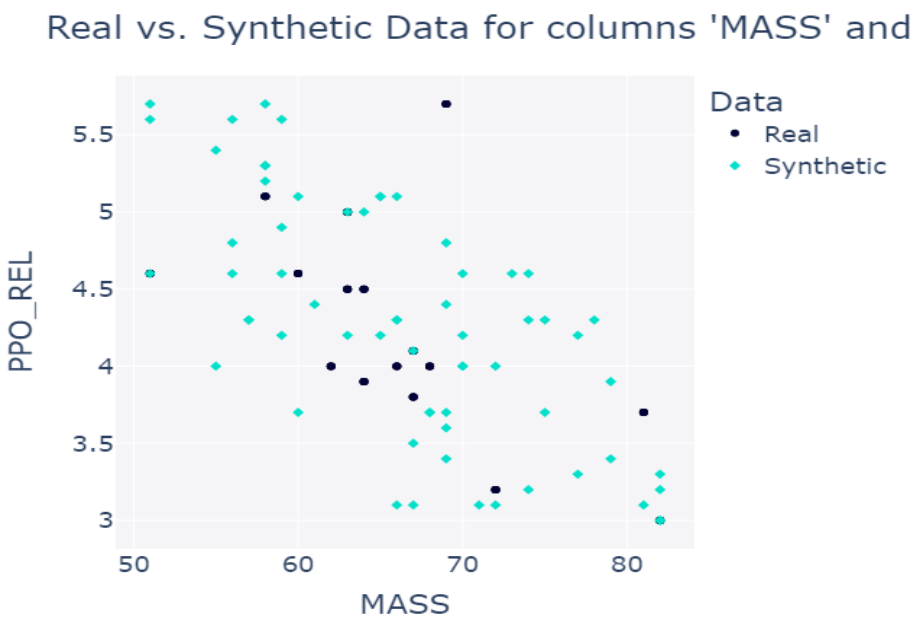


Figure 18 Scatter Plot Mass vs PPO_REL

The overall distribution score was 0.83 which falls under the acceptable range. But we cannot accept the fact that some of the columns like IL6 have extremely low shape. Hence, we decided to optimize this model by passing the numerical_distribution parameter as 'normal' for the MASS and truncated normal for IL6 manually during the training process. After optimization, the overall score increased to 0.84 but the distribution for IL6 was not captured properly as shown in Figure 19.

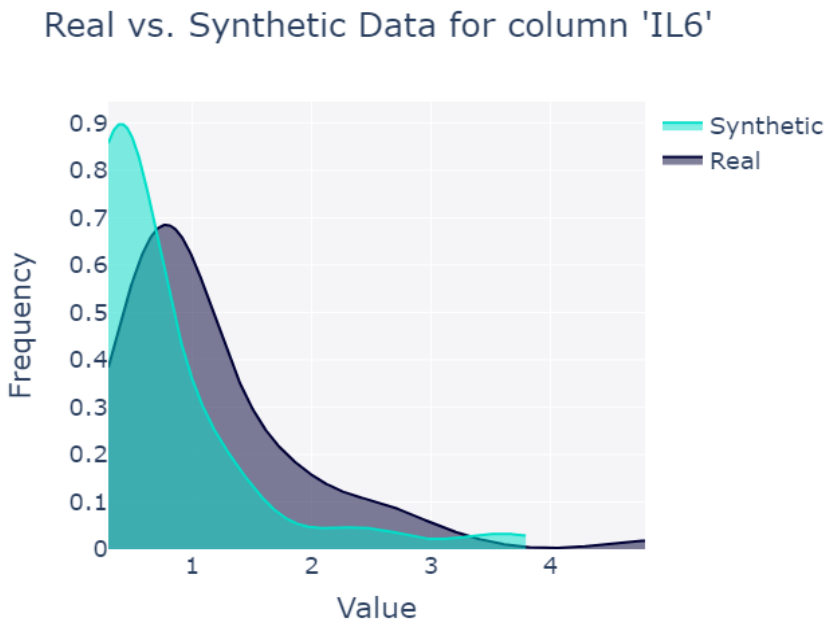


Figure 19 IL6 density plot for optimized model

In the CTGAN model, we choose different data ‘Hanstock_2016_Illness.csv’ which contains 15 columns and 50 rows with the complex distribution. The base model with all default parameters including epochs of 2500 and batch size of 250 produced an overall score of 81.06%. But the column shape was only 0.74 and the column pair trend was 0.87. We further investigated the stability of the model by plotting its loss value over epochs as shown in figure 20. This result showed that the model was highly unstable with fluctuating loss values.

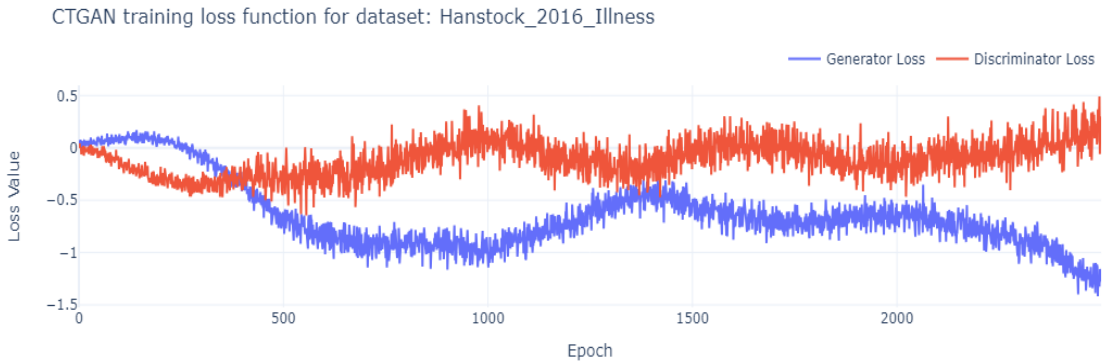


Figure 20 Training loss for base CTGAN over epochs

The quality report further showed that columns like ‘bmi’ and ‘vitD’ have a score of 0.64 which indicates the distribution of synthetic data was not captured correctly. The density plot for each of these columns is shown below in Figure 21 and Figure 22 respectively.

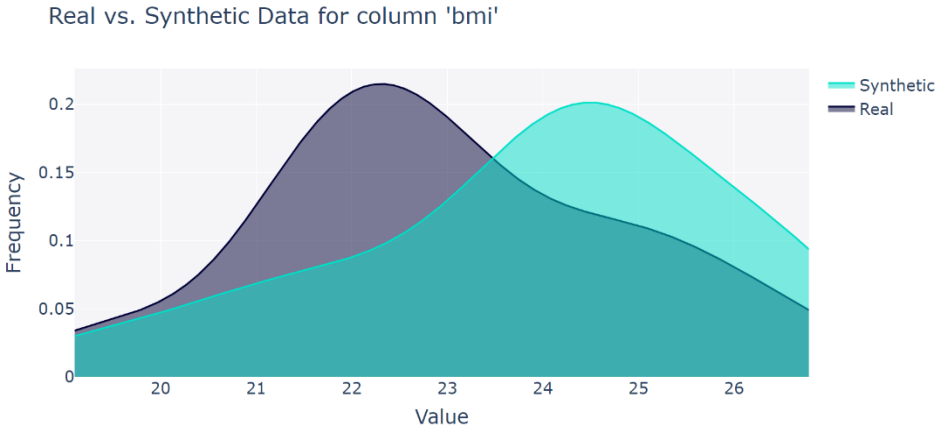


Figure 21 Distribution comparison of bmi, real vs synthetic data

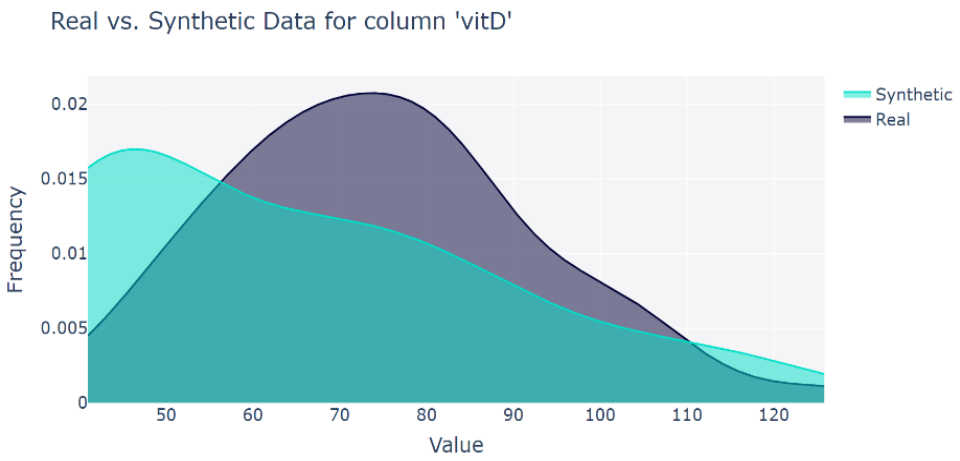


Figure 22 Distribution comparison of vitD, real vs synthetic data

To increase the performance and select the best model we trained the model in phase. As per our research, the optimized CTGAN must have a stabilized negative loss value for the generator and a stabilized loss value around 0 for the discriminator (DataCebo, Inc., 2023). In phase training, we iterated the epoch value starting from 100 to 2000 and selected the model which has minimum loss along with the highest KSComplement value for the 'bmi' column. During our model training process, the model with 700 epochs meets these criteria. This model also produces the KSComplement score of 0.8 for 'bmi' which our base model has struggled on. Figure 23 and Figure 24 show the training loss and 'bmi' per epoch.

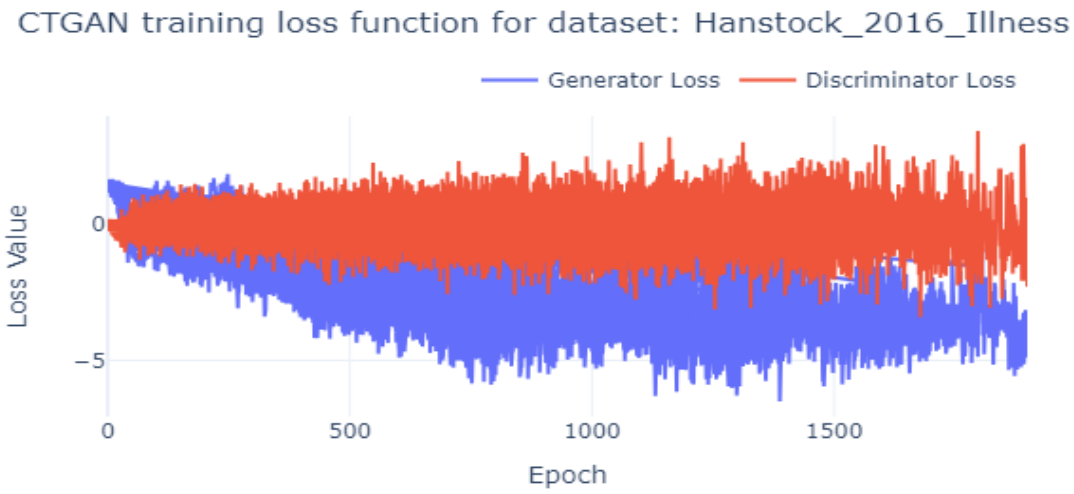


Figure 23 Training loss over epochs for optimized CTGAN model

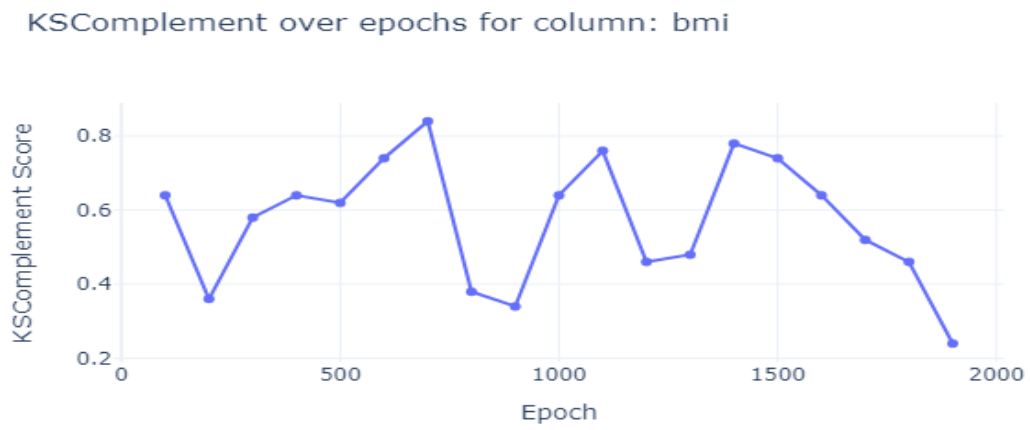


Figure 24 Column score for 'bmi' over epochs

We further compared the distribution of real and synthetic data for ‘bmi’ and ‘vitD’ to see the performance of our select model. The Figure 25 shows that our model has failed to capture the maximum range for the bmi column, but it has been able to mimic the data distribution to some extent.

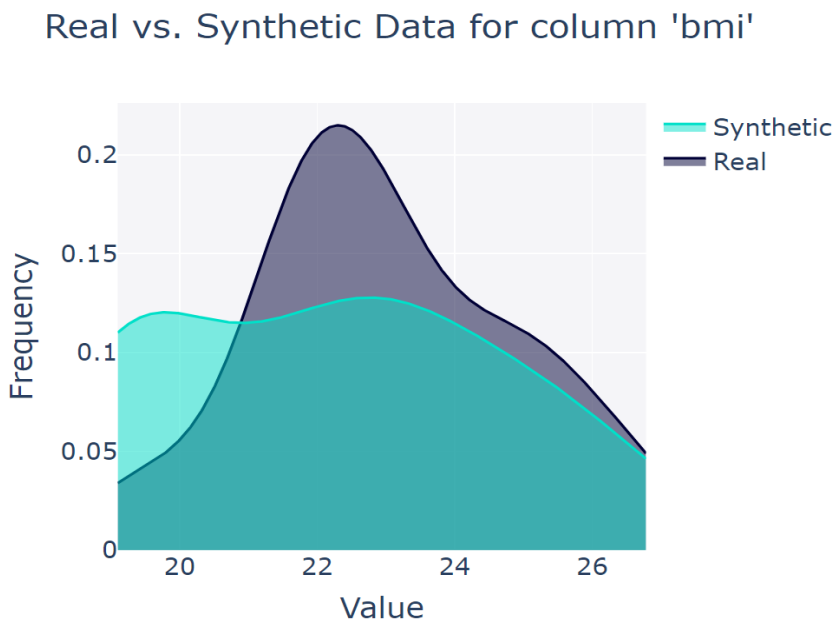


Figure 25 Distribution comparison of bmi by optimized CTGAN model

We further investigated the correlation similarity between real and synthetic data for every column. The average score was 0.84 which was acceptable considering the complexity of the dataset. The below figure 26 is the heat map which compares the correlation between each column in real and synthetic data. The result showed that the synthetic data has struggled to maintain correlation among each of the columns.

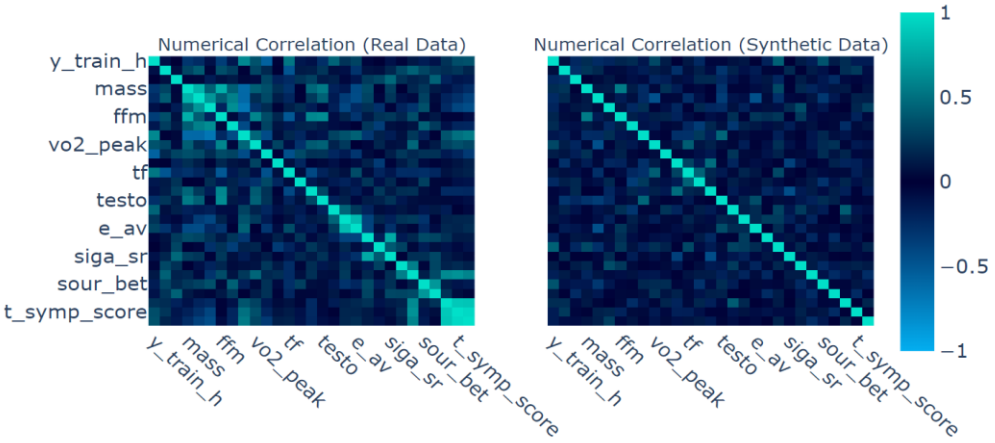


Figure 26 Correlation comparison for real and synthetic data

We use the same 'Hanstock_2016_Illness.csv' dataset in PARSynthesizer which produces the overall data quality report of 0.83 with an average distribution score of 0.79 and correlation similarity score of 0.87. Figure 27 depicts the dissimilarity between real and synthetic data. The score 0 implies both are similar whereas the score 1 implies both data are completely different. The correlational similarity between real and synthetic data as shown in Figure 28 demonstrates the correlation between columns in synthetic data is not preserved.

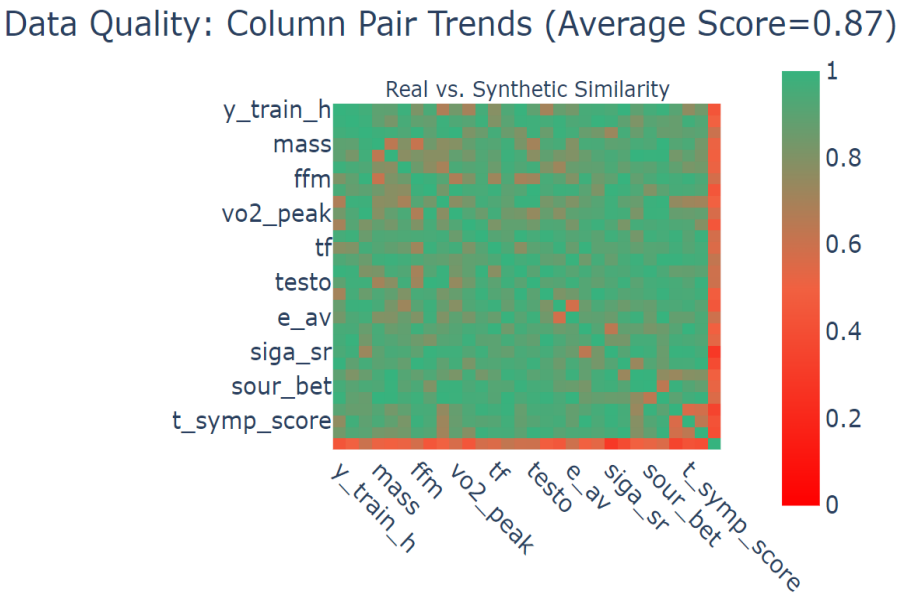


Figure 27 Data similarity comparison between real vs synthetic data

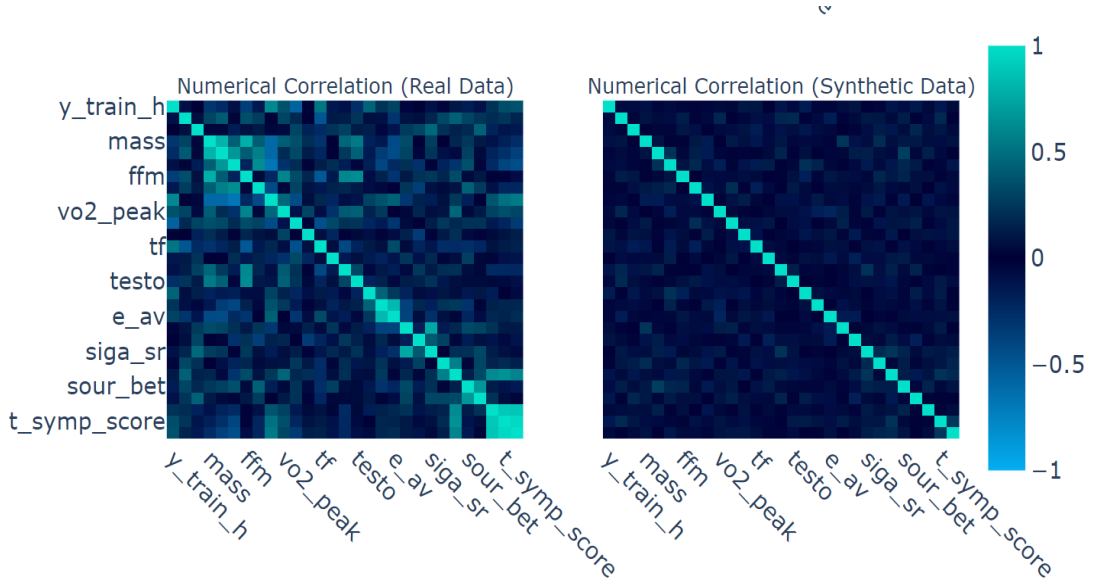


Figure 28 Correlation similarity between real vs synthetic data

After comparing the results of all three models, CTGAN Synthesizer was considered as the best model which can capture smaller details from the dataset. The only synthesizer based on a deep learning model was able to mimic the correlations between every column in synthetic data. Although the choice of synthesizer depends upon the type of dataset and complexity, one can consider the CTGAN model for single-table or multi-table synthetic data generation. However, this model comes with a computational cost. If the dataset is not that complex one can consider GaussianCopulas as it is fast and less complex, but for a complex dataset, it struggles to capture small details like correlations and similarity.

4 Further Work

In GaussianCopulas one can work on identifying the data distribution of each column before training the model. The result can be passed during the training process so that the model can use the Copulas function to produce similarly distributed data. This method will enhance the quality score as well as the computational time of the model.

In CTGAN one can iterate over the batch size of the model just like we did for epochs. We can also experiment with fine-tuning the generator and discriminator separately using the Scikit learn Grid Search. We can also stabilize the loss function by changing the learning rate for both the generator and the discriminator. This method could save a lot of computational power because we may get an optimized model at a lower value of epochs. Moreover, we can break our dataframe into smaller groups according to the complexity of columns and create customized models for each subgroup.

In PARSynthesizer we can discard categorical columns as this value can be easily replicated by any synthesizer. We can then use numerical columns with sequence keys to synthesize the data. One can

also divide the dataframe based on different sequence categories such as pre and post exercise and generate data individually for each of the categories.

There is the SDGym library that explores the best synthesizer according to the dataset. Due to time constraints, we were unable to implement this library but, in the future, it will be beneficial and save a lot of time in synthetic data generation.

5 Conclusion

After comparing all three models and their performance we found that the model behaves according to the complexity of the data. If we can organize data for a specific model, then the model can produce higher-quality synthetic data. Although the CTGAN model has a lower score among three, it is one of the most powerful models. Most of our work was focused on optimizing and tweaking the dataset for CTGAN. Moreover, the model's performance greatly varies based on parameters like data distribution in GaussianCopulas, epochs in CTGAN, and sequence_index in PAR Synthesizer. Hence, we must experiment with these values to optimize each of the models.

In this project, we have successfully applied different libraries in Python and R for the data curation process. We cleaned open-source datasets of different complexity and format. These clean datasets will be useful for the creation of learning materials. Similarly, the optimized machine learning model can be used in the future to produce synthetic data of any volume and complexity. Many sports practitioners and researchers can benefit from this data in making noble discoveries and predictions.

This project has extensively used modern and sophisticated methods to fulfill the objective. Ongoing buzzwords like Artificial Intelligence and GAN models have been successfully applied in sports dataset. It has also demonstrated the current research and progress of the overall sports analytics' domain. Further optimized model and highest quality synthetic data could revolutionize the sport analytics in near future.

References

Ahmed, S. J. (2023). Machine Learning in Sports Analytics and Performance Prediction. *Medium*.

Australian Institute of Sport. (2023, December 7). *Publication/Media Centre/AIS*. Retrieved from AIS: https://www.sportaus.gov.au/media-centre/publications

Bartlett, R. (2006). Artificial intelligence in sports biomechanics: new dawn or false hope? . *J Sports Sci Med*, 474-9.

Carlo Batini, C. C. (2009). Methodologies for data quality assessment and improvement. *ACM Computing Surveys*, 1-52.

Conor Hassan, R. S. (2023). Deep Generative Models, Synthetic Tabular Data, and Differential Privacy: An Overview and Synthesis. *arxiv.org*.

DataCebo, Inc. (2023, December 7). *Synthetic Data Vault*. Retrieved from Synthetic Data Vault: https://docs.sdv.dev/sdv/

FSTec. (2024, February 12). Retrieved from Fstec: https://fstec.com/profile/billy-beane

James Jordon, F. H. (2018). *Synthetic Data - what, why and how?* The Royal Society.

Lei Xu, M. S. (2019). Modeling Tabular Data using Conditional GAN. *NeurlPS*.

McGauran, N. W. (2010). Reporting bias in medical research - a narrative review. *BMC*.

Michael Lewis, Y. Y. (2016). An Empirical Examination of the Development and Impact of Star Power in Major League Baseball. *Journal of Sports Economics*, 155-187.

SPEEDS. (2023, July). *The SPEEDS Project*. Retrieved from Quarto: https://speeds.quarto.pub/speeds/about.html

The Business Research Company. (2023, January). *Sports Global Market Report*. Retrieved from The Business Research Company: https://www.thebusinessresearchcompany.com/report/sports-global-market-report

vanderSchaarLab. (2023, December 7). *synthcity*. Retrieved from github.com: https://github.com/vanderschaarlab/synthcity#--synthcity

Viechtbauer, W. (2005). Bias and Efficiency of Meta-Analytic Variance Estimators in the Random-Effects Model. *Journal of Educational and Behavioral Statistics*, 261-293.

Waemehoven, J. (2022). *Statistics in High Performance Sport.* Australian Institute of Sport.

Wise, J. (2023, April 7). *HOW MUCH DATA IS GENERATED EVERY DAY IN 2024?* Retrieved from EARTHWEB: https://earthweb.com/how-much-data-is-created-every-day/

Zhang, K. (2022). SequentialModelsintheSyntheticDataVault. *Datacebo*.

Appendices

A sample screenshot of the code is shown below for the reference. For the details code one can refer SDV documentation (DataCebo, Inc., 2023).