Paper OS03 – PHUSE US Connect 2025

Python's Future in Clinical Trials: Innovations and Collaborative Advancements

Sohan Lal, Geninvo, India
Shweta Shukla, Geninvo, USA

ABSTRACT

In the Field of clinical trials, effective collaboration among study teams is crucial for ensuring the success and integrity of research undertakings. This abstract deep dive into the pivotal role of Python programming language in study team collaboration within clinical trials, particularly through its power in handling PDF files and leveraging AL/ML techniques and python has the ideal package to update Word documents with the data automatically. Python, known for its versatility and robustness, offers plenty of tools and libraries tailored to the needs of clinical researchers. Leveraging Python's capabilities, study teams can streamline data analysis visualization, uncover the data hidden patterns, predict outcomes, optimize the clinical trial protocols, and improve the communication processes among study teams. Through illustrative examples and case studies, this abstract talk about Python's future role for transformative and innovative collaboration in clinical trials to improved patient care outcomes

INTRODUCTION

As data has become the driving force of the 21st century and transforming the business operations across industries. With vast digital data at hand and rapidly advancing technology, companies are progressively exploring methods to leverage the connection between data and technology to maximize their benefits. While traditional, validated tools have the backbone of data analysis, there is a growing shift towards adopting the open-source tools. These tools have the potential to unlock valuable insights hidden within the data.

Clinical trials are the cornerstone of medical research, aiming to evaluate new treatments, drugs, or medical devices. The success of these trials depends on a variety of factors, including the integrity of the data collected, the efficiency of team collaboration, and the proper analysis of trial results. Study teams, consisting of physicians, researchers, statisticians, and support staff, work together to ensure that these trials are run smoothly and that the data generated is accurate and actionable.

However, managing the vast amount of data generated during a clinical trial can be a challenging task. Traditional methods of data processing, analysis, and reporting are often inefficient and time-consuming. Furthermore, maintaining effective communication and collaboration among the various stakeholders can also be challenging. This is where Python programming comes into role, offering a range of tools and techniques that can significantly improve both the efficiency of the research process and the collaboration between study teams

Open-source Python has become a pivotal tool in the development and training of Artificial Intelligence (AI) and Machine Learning (ML) models. Libraries such as TensorFlow, Keras, and Scikit-learn provide extensive frameworks that enable researchers and data scientists to design, train, and deploy complex AI/ML models efficiently. The flexibility and accessibility of Python allow for quick experimentation with different algorithms, hyperparameters, and model architecture, making it an ideal choice for AI/ML development. Additionally, its vibrant community contributes to a vast repository of resources, tutorials, and pre-trained models, further accelerating innovation.

This paper demonstrates how open-source technology python can be leveraged and emphasizing the various ways Python proves to be valuable. We begin by examining the advantages and drawbacks of using Python. Then, we will introduce **Pandas**, a robust library for data manipulation. Lastly, we will illustrate some fundamental examples of how Python can be used to create individual plots, basic dashboards, word documents and PDF outputs.

ADVANTAGES OF USING PYTHON

- **Easy to Learn and Readable Syntex:** Python is known for its simple and clean syntax, which is designed to be human-readable. This makes an ideal choice for beginners and reduces the complexity of writing and maintaining code.
- **Large and Active Community:** Python has a vast and diverse community of developers, contributors and users who provide valuable resources, framework and libraries. This large community ensures that support is

readily available and solutions to various programming challenges are often easily found in forums, documentation and tutorials.

- **Powerful AI/ML Capabilities:** Python offers robust libraries like TensorFlow, Keras, and Scikit-learn, making it an ideal choice for developing and training AI/ML models. Its flexibility and support for rapid experimentation empower developers to optimize algorithms and model architecture, facilitating advancements in AI/ML research, including applications in clinical trials for data analysis and prediction.
- **Cross-Platform Compatibility:** Python is platform-independent, meaning that code written in python can run on multiple OS(Windows, MacOS, Linux) with little to no changes. This flexibility is beneficial for developers who want to ensure that their applications are widely accessible without needing to write code for different environments

CHALLENGES OF USING PYTHON

- **Integration Challenges:** In clinical research, it's common to work with various legacy systems, databases, and software that may not be easily compatible with Python. For example, clinical data management systems (CDMS) and **laboratory information management systems (LIMS)** often use proprietary software or databases that might require custom connectors or wrappers to integrate seamlessly with Python. While Python has robust support for many libraries and APIs, integrating it with legacy systems in highly regulated environments could introduce challenges.
- **Regulatory and Compliance Issues:** Clinical research and drug discovery are heavily regulated and ensuring compliance with standards such as 21 CFR Part 11 (FDA regulations) or Good Clinical Practice (GCP) is critical. Python, while flexible and powerful, may not always offer built-in tools or certifications for compliance with these standards. The absence of certain built-in controls in Python may require additional steps to ensure compliance, which can complicate the development process in highly regulated industries.
- **Lack of Real-Time Processing:** Many clinical and drug discovery tasks require real-time data processing and analysis (e.g., processing live patient data, continuous monitoring, or drug response simulations). Python is not optimized for real-time systems, making it less suitable for such tasks compared to languages designed for real-time computing or low-level system integration.

In this paper, we will examine Python's role in clinical trials, with a particular focus on three key areas:

1. **Data Management and PDF Handling**: Leveraging Python's capabilities to automate the extraction, manipulation, and analysis of data stored in PDF documents, which are commonly used in clinical research.

2. **Artificial Intelligence and Machine Learning**: Exploring how AI and ML techniques, powered by Python, can be applied to clinical trial data to uncover hidden patterns, predict outcomes, and optimize protocols using NLP libraries.

3. **Automating Reporting and Document Updates**: Demonstrating how Python can be used to automatically update Word documents with real-time trial data, ensuring that study teams stay up to date with the latest information without manual intervention.

DATA MANAGEMENT AND PDF HANDLING

Clinical trials often involve a significant amount of documentation, including patient data, trial protocols, informed consent forms, and regulatory filings. Many of these documents are stored in PDF format, which, while widely used, can be difficult to manipulate programmatically.

Python is extensively used for handling different types of data formats and performing integration between multiple technologies. We utilize Python libraries to efficiently read and manipulate data from various sources, store it in databases, and manage the flow of information. By using its versatile libraries, Python enables smooth interaction between different data systems, ensuring data is processed and stored in an optimal manner.

Python offers several powerful libraries for working with PDFs, such as **PyPDF2**, **pdfminer.six**, and **PyMuPDF**. These libraries allow researchers to extract text, tables, and images from PDF files, enabling the automation of data extraction tasks.

1.1 Extracting Data from PDF Forms

Many clinical trials involve the collection of patient data through standardized forms, which are often submitted in PDF format. Python can be used to automate the process of extracting this data, significantly reducing the time and effort involved in data entry and validation.

For example, using **pdfminer.six**, researchers can extract tabular data from PDF documents and convert it into structured formats such as CSV or Excel, where it can then be further processed and analyzed.

1.2 Data Transformation and Integration

After extracting data from PDFs, Python's powerful data manipulation libraries like pandas can be used to clean, transform, and integrate data into a usable format. Researchers can merge data from multiple PDFs or other sources, clean missing or erroneous data, and prepare it for analysis.
This transformation process not only saves time but also ensures that data is standardized and ready for use in further analysis or reporting.

Examples of PymuPDF in python

```
import fitz  # PyMuPDF

# Open the PDF file
doc = fitz.open("clinical_trial_report.pdf")

# Extract text from the first page
page = doc.load_page(0)  # 0-based index for the first page
text = page.get_text("text")

print(text)
```

```
In [366]: text
Out[366]: 'ALEPRO trial \nENGOT-OV70/BGOG-OV70 \n
\nVersion 1.0 (8 Feb 2023) \n \n \nPROTOCOL SYNOPSIS \n
\nTitle of clinical Trial («Trial») \nPhase II, open
label, multicenter study of abemaciclib and letrozole in
\nEstrogen receptor positive rare ovarian cancer
\nProtocol \nShort \nTitle \nAcronym \nALEPRO \nTrial
Phase (I, II, III, IV) \n2 \nSponsor name \nUniversity
Hospitals Leuven (UZ Leuven) \nCoordinating Investigator
\nEls Van Nieuwenhusyen \n            Contact Address CI
\nHerestraat 49, 3000 Leuven \n            Contact Email CI
\nEls.vannieuwenhuysen@uzleuven.be \n            Contact
Phone CI \n0032/16342531 \nEudraCT number
```

```
# Open the PDF file
doc = fitz.open("clinical_trial_report.pdf")

# Extract text from the first page
page = doc.load_page(0)  # 0-based index for the first page
text_blocks = page.get_text("blocks")  # Extract text in blocks
```

```
(76.58399963378906,
 142.09996032714844,
 518.1669311523438,
 165.49998474121094,
 'Title of clinical Trial («Trial») \nPhase II, open
abel, multicenter study of abemaciclib and letrozole in
nEstrogen receptor positive rare ovarian cancer \n',
 3,
 0),
(76.58399963378906,
 173.0599822998047,
 201.63734436035156,
 183.61997985839844,
 'Protocol \nShort \nTitle \n',
 4,
 0),
```

The above code and output snippets demonstrate how to read PDF files in two different ways using **PyMuPDF** (also known as **fitz**). The first approach extracts the entire PDF content as plain text, while the second extracts the text in blocks, also capturing the coordinates of each block. These extracted texts can then be preprocessed using libraries such as **spaCy**, **re**, or others to facilitate further analysis.**1.3 Automating Word Document Updates using python-docx**

Python-docx is a powerful Python's library that enables the automation of creating, editing, and updating Microsoft Word documents. It simplifies the process of managing large volumes of documents, ensuring that updates are consistent and efficient.

Python-docx can be a valuable tool when it comes to automating the comparison and extraction of data from multiple documents, such as **Word documents (docx)**, **PDFs**, and other formats in clinical trials or pharmaceutical documentation.When you have multiple **Word documents** (such as updated protocol documents, reports, or study logs), you may need to compare content between them to ensure consistency or track changes. **Python-docx** allows you to extract and compare the text and tables in **docx** files easily, helping identify discrepancies or ensuring that certain sections (e.g., tables or references) match across different versions.

```python
from docx import Document

# Function to read and extract text from a .docx file
def read_docx(file_path): # Load the .docx file
    doc = Document(file_path)
    text = ""
    for para in doc.paragraphs:
        text += para.text + "\n"  # Add a newline after each paragraph

    return text

file_path = 'SAP.docx'  # Replace with your file path
doc_text = read_docx(file_path)

print(doc_text)
```

```
In [12]: print(doc_text)
title page      1
TABLE OF CONTENTS       2
List Of Abbreviations   4
1.      INTRODUCTION    5
2.      OBJECTIVES      5
3.      STUDY OVERVIEW  5
3.1     Study Design    5
3.2     Sample Size     7
3.3     Randomization and Unblinding Procedures 8
4.      STUDY ENDPOINTS/Outcomes        8
5.      HYPOTHESES TESTING      9
6.      ANALYSIS SUBSETS        10
6.1     Safety Population       10
6.2     Modified Intent to Treat Population (mITT Population)   10
```

```python
from docx import Document

 # Function to read tables from a .docx file
def read_docx_tables(file_path):
    doc = Document(file_path)
    tables_data = []
    for table in doc.tables:
        table_data = [] # Loop through each row in the table
        for row in table.rows:
            row_data = [] # Loop through each cell in the row
            for cell in row.cells:
                row_data.append(cell.text.strip())
            table_data.append(row_data)
        tables_data.append(table_data)
    return tables_data

tables = read_docx_tables(file_path)
print(tables)
```

```
[[['Abbreviation or special term', 'Explanation'],
  ['AE', 'Adverse Event'],
  ['ANOVA', 'Analysis of Variance'],
  ['BMI', 'Body Mass Index'],
  ['CFR', 'Code of Federal Regulations'],
  ['eCRF', 'Electronic Case Report Form'],
  ['CRO', 'Contract Research Organization'],
  ['DCF', 'Data Clarification Forms'],
  ['EC', 'Ethics Committee'],
  ['ED', 'Early Discontinuation'],
  ['EDC', 'Electronic Data Capture'],
  ['EOT', 'End of Treatment'],
  ['FDA', 'Food and Drug Administration'],
  ['GCP', 'Good Clinical Practice'],
  ['hCG', 'Human Chorionic Gonadotropin'],
  ['HEENT', 'Head, Eyes, Ears, Nose and Throat'],
```

ARTIFICAL INTELLEGENCE AND MACHINE LEARNING

Artificial Intelligence (AI) and Machine Learning (ML) are revolutionizing clinical trials by providing tools that can analyze large volumes of data, identify hidden patterns, and predict outcomes. Python, with its rich ecosystem of libraries such as TensorFlow, Keras, scikit-learn, and PyTorch, is well-equipped to handle the complexities of AI and ML in clinical research.

1. SDTM data domain prediction

Python's robust machine learning ecosystem allowed us to implement models for predictive analysis. Python's machine learning libraries allow researchers to build predictive models that can forecast the likelihood of success or failure for a treatment, based on factors such as patient demographics, medical history, and different domains data (AE, LB, VS etc.)

For example, we used the **Random Forest** algorithm, **NLTK**, **re** and **word2Vec**, a powerful ensemble learning method, to predict and automate certain data mappings. Used Python's **scikit-learn** library to develop, train, and evaluate the Random Forest model, which allowed us to make data-driven predictions based on the patterns observed in the dataset.

To predict the best possible **SDTM variable mappings**, we have developed four specialized machine learning models:

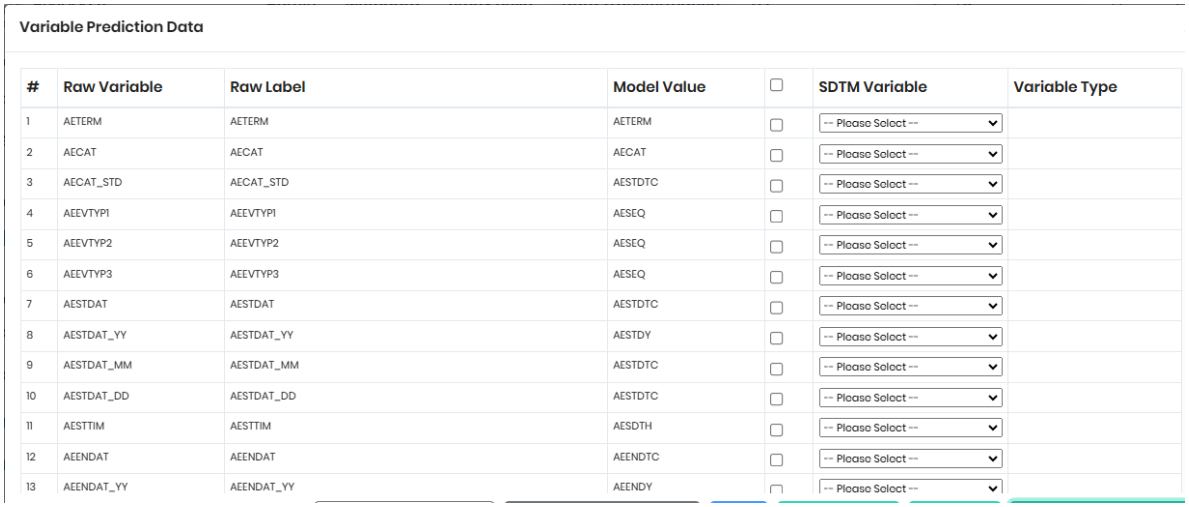
- **Character Model**
- **Label Model**
- **Datetime Model**

- **Numeric Model**

Each of these models is trained on historical clinical trial data, and work by analysing the data and providing the most probable mappings based on statistical probabilities. This approach significantly reduces manual intervention and enhances the consistency and accuracy of the mappings.

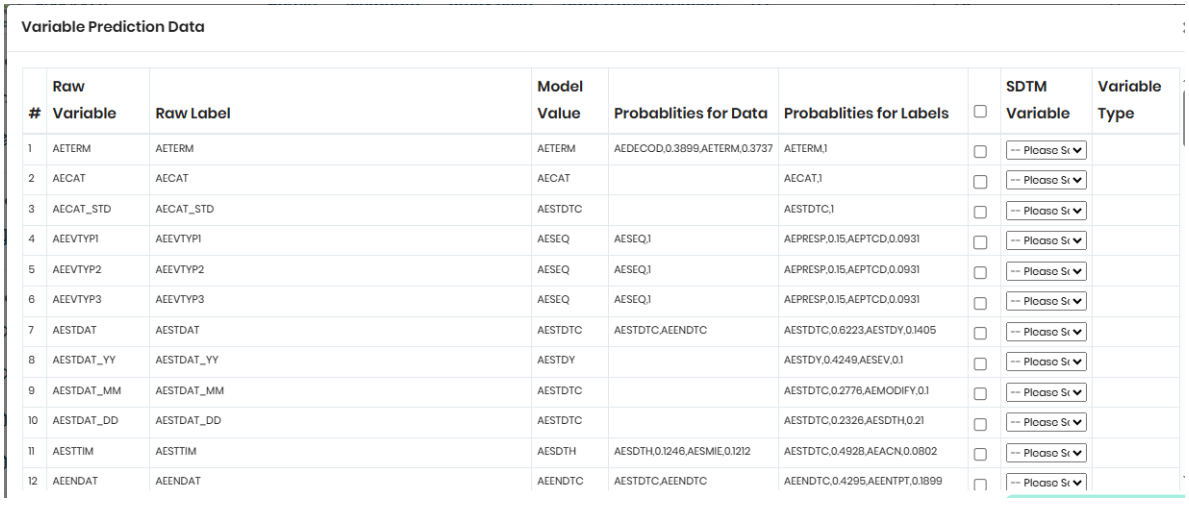
The models predict the appropriate method based on user selections and variable patterns. For example, if a variable contains “DAT,” “DT,” or “DTC,” we associate it with a date-related method. Once the final dataset is created, we apply SDTM terminology to ensure compliance with standard definitions. Additionally, we can arrange variables in the correct order as per the sequence defined in the Implementation Guide (IG). We also evaluate that the dataset includes all required and expected variables to meet SDTM standards. Our predictive models—Character Model, Label Model, Datetime Model, and Numeric Model—analyze the data and determine the best possible SDTM variable mappings based on probability. These models help automate the mapping process, improving accuracy and efficiency.

Once the model completes the mapping predictions, **Python** handles storage tasks, ensuring that SDTM terminology is consistently applied throughout the dataset. Also include logic to flag discrepancies in the mapped values, particularly when a value does not conform to SDTM standards. In such cases, a list of discrepancies is generated, allowing users to review and select the correct mappings, improving overall accuracy.



Variable Prediction Data

#	Raw Variable	Raw Label	Model Value		SDTM Variable	Variable Type
1	AETERM	AETERM	AETERM	☐	-- Please Select --	
2	AECAT	AECAT	AECAT	☐	-- Please Select --	
3	AECAT_STD	AECAT_STD	AESTDTC	☐	-- Please Select --	
4	AEEVTYP1	AEEVTYP1	AESEQ	☐	-- Please Select --	
5	AEEVTYP2	AEEVTYP2	AESEQ	☐	-- Please Select --	
6	AEEVTYP3	AEEVTYP3	AESEQ	☐	-- Please Select --	
7	AESTDAT	AESTDAT	AESTDTC	☐	-- Please Select --	
8	AESTDAT_YY	AESTDAT_YY	AESTDY	☐	-- Please Select --	
9	AESTDAT_MM	AESTDAT_MM	AESTDTC	☐	-- Please Select --	
10	AESTDAT_DD	AESTDAT_DD	AESTDTC	☐	-- Please Select --	
11	AESTTIM	AESTTIM	AESDTH	☐	-- Please Select --	
12	AEENDAT	AEENDAT	AEENDTC	☐	-- Please Select --	
13	AEENDAT_YY	AEENDAT_YY	AEENDY	☐	-- Please Select --	



Variable Prediction Data

#	Raw Variable	Raw Label	Model Value	Probablities for Data	Probablities for Labels		SDTM Variable	Variable Type
1	AETERM	AETERM	AETERM	AEDECOD,0.3899,AETERM,0.3737	AETERM,1	☐	-- Please Se	
2	AECAT	AECAT	AECAT		AECAT,1	☐	-- Please Se	
3	AECAT_STD	AECAT_STD	AESTDTC		AESTDTC,1	☐	-- Please Se	
4	AEEVTYP1	AEEVTYP1	AESEQ	AESEQ,1	AEPRESP,0.15,AEPTCD,0.0931	☐	-- Please Se	
5	AEEVTYP2	AEEVTYP2	AESEQ	AESEQ,1	AEPRESP,0.15,AEPTCD,0.0931	☐	-- Please Se	
6	AEEVTYP3	AEEVTYP3	AESEQ	AESEQ,1	AEPRESP,0.15,AEPTCD,0.0931	☐	-- Please Se	
7	AESTDAT	AESTDAT	AESTDTC	AESTDTC,AEENDTC	AESTDTC,0.6223,AESTDY,0.1405	☐	-- Please Se	
8	AESTDAT_YY	AESTDAT_YY	AESTDY		AESTDY,0.4249,AESEV,0.1	☐	-- Please Se	
9	AESTDAT_MM	AESTDAT_MM	AESTDTC		AESTDTC,0.2776,AEMODIFY,0.1	☐	-- Please Se	
10	AESTDAT_DD	AESTDAT_DD	AESTDTC		AESTDTC,0.2326,AESDTH,0.21	☐	-- Please Se	
11	AESTTIM	AESTTIM	AESDTH	AESDTH,0.1246,AESMIE,0.1212	AESTDTC,0.4928,AEACN,0.0802	☐	-- Please Se	
12	AEENDAT	AEENDAT	AEENDTC	AESTDTC,AEENDTC	AEENDTC,0.4295,AEENTPT,0.1899	☐	-- Please Se	

The above snippets demonstrate the output of a machine learning (ML) prediction model that predicts adverse event variables based on adverse event label data. Initially, the snippets display the **raw labels** and **raw variables** of the adverse event data. Then, the model's **predicted values** are presented, including the predicted labels and the associated probability values for each prediction. This enables the user to evaluate multiple predictions for the same variable and make an informed decision about which prediction is most likely correct, based on the probabilities.

2. Natural Language Processing (NLP) for Redaction

Natural Language Processing (NLP) is a branch of **Artificial Intelligence (AI)** aimed at enabling machines to understand, interpret, and generate human language. The integration of **machine learning (ML)** models has significantly advanced NLP, providing powerful tools for tasks such as language translation, sentiment analysis, chatbots, text summarization, and information retrieval.

In the pharmaceutical industry, NLP is transforming processes like **clinical data extraction**, **medical literature review**, and **adverse event detection**. It helps automate and improve various tasks, including the **generation of regulatory documents** and data **anonymization** to ensure compliance with privacy regulations such as **GDPR** and **HIPAA**.

NLP is also playing a crucial role in **pharma literature mining**, allowing researchers to efficiently extract insights from vast amounts of published scientific literature. In **drug discovery**, NLP tools can sift through clinical trial reports, patents, and medical texts to identify promising drug candidates, biomarkers, and new therapeutic targets.

Key Components of NLP:

- **Named Entity Recognition (NER):**

  NER is a critical part of NLP that helps automate the process of redacting sensitive information. It identifies specific entities in text, such as names, addresses, medical conditions, and dates, which need to be protected for privacy and compliance.
  For instance, in a clinical document, a sentence like “John Smith, a 45-year-old male with a history of heart disease, was admitted” contains private details that need to be redacted. NLP models automatically detect and redact entities like "John Smith" and "heart disease" to protect patient privacy.

  By categorizing sensitive data, NER ensures compliance with privacy laws like HIPAA or GDPR, while maintaining the integrity of the document. It is essential in clinical settings where data protection is a top priority.

- **Semantic Analysis:**

  Traditional redaction methods often rely on matching specific keywords, which can be inaccurate—either removing non-sensitive content or missing critical sensitive data. The limitations of keyword-based methods are especially evident in complex texts where context is important for determining what is sensitive.
  NLP, along with semantic analysis, addresses these issues by understanding the meaning behind the words, not just the words themselves. This understanding helps the system accurately redact sensitive information, considering the surrounding context.
  For example, in clinical trial documents, terms like "heart disease" or "sensitive medical history" should be flagged for redaction. While a keyword-based system might miss these terms, NLP systems can evaluate the text's context and redact only sensitive information, leaving non-sensitive terms untouched.

- **Multilingual NLP Models:**

  Models like BERT (Bidirectional Encoder Representations from Transformers) are designed to process multiple languages, handling the grammatical and syntactical differences of each language. Multilingual models ensure that sensitive data is detected and redacted accurately in both the original and translated versions of a document

1. SYNOPSIS

Name of Study Drug RVT-3101
Name of Active Ingredient Recombinant human monoclonal antibody (immunoglobulin gamma-1 with kappa light chains, IgG1 kappa) directed against Tumor necrosis factor-like Ligand 1A (TL1A)
Title of Study A Phase 2, Multicenter, Double-blind, Two-arm Study of Subcutaneous RVT-3101 for the Treatment of Subjects with Moderate to Severe Active Crohn's Disease
Study Center(s) This study will be conducted in up to approximately 150 sites worldwide. Worldwide Clinical Trials, a contract research organization, will oversee operational aspects of this study on behalf of Telavant, Inc., the Sponsor of the study.
Phase of Development

1. SINOPSIS

Nombre del fármaco del estudio RVT-3101
Nombre del principio activo Anticuerpo monoclonal humano recombinante (inmunoglobulina gamma-1 con cadenas ligeras kappa, IgG1 kappa) dirigido contra el ligando 1A (TL1A) similar al factor de necrosis tumoral
Título del estudio Estudio de fase II, multicéntrico, doble ciego y de dos grupos sobre RVT-3101 subcutáneo para el tratamiento de sujetos con enfermedad de Crohn activa moderada a grave
Centro(s) del estudio Este estudio se llevará a cabo en aproximadamente un máximo de 150 centros de todo el mundo. Worldwide Clinical Trials, una organización de investigación por contrato, supervisará los aspectos operativos de este estudio en nombre de Telavant, Inc., el promotor del estudio.

The following code snippets demonstrate how to perform redaction of sensitive information in clinical documents. In the first example, redaction is performed on a single word or string using logic and regular expressions. The second example expands on this by applying the same redaction technique to a document in a different language, utilizing **AI/ML** above mentioned **Multi-lingual BERT** models techniques for automatic identification and redaction of sensitive information across languages

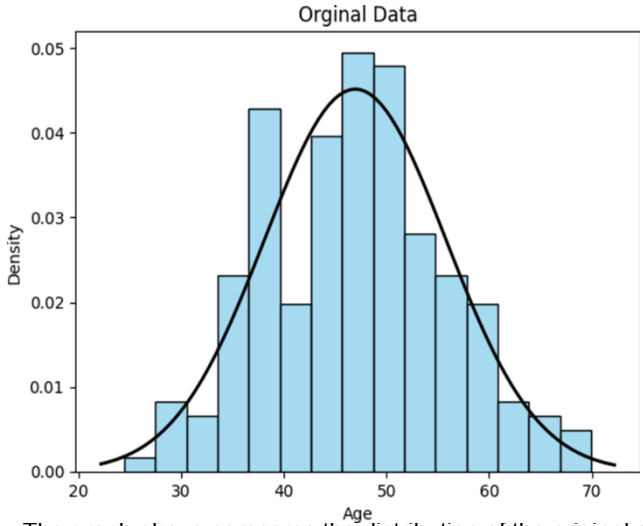
SYNTHETIC DATA GENERATION

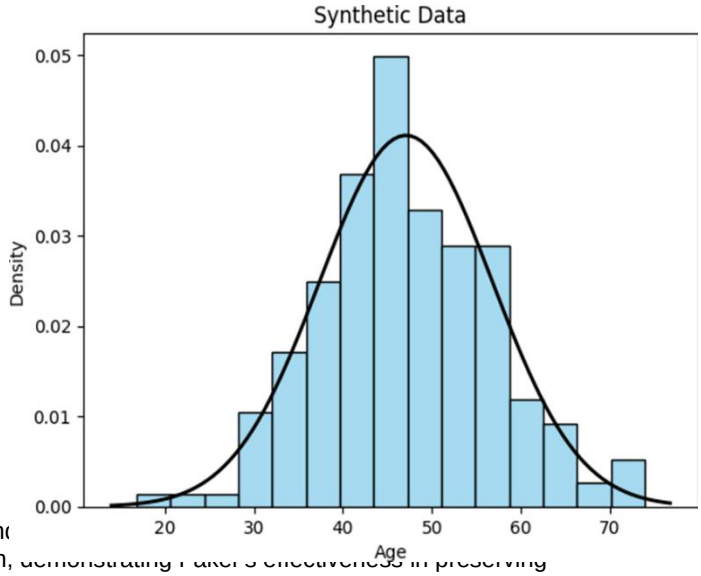
Synthetic data generation refers to the process of creating artificial data that mimics the characteristics of real-world data without revealing sensitive or personal information. This is especially important for tasks like testing, machine learning model training, and data privacy compliance.

In clinical trials and research, data plays a crucial role in driving informed decisions, improving healthcare outcomes, and ensuring the safety and efficacy of new treatments. However, real patient data often contains sensitive information that cannot be shared due to privacy regulations like HIPAA (Health Insurance Portability and Accountability Act) or GDPR (General Data Protection Regulation). This is where **synthetic data** generated using tools like the **Faker library** in Python can be incredibly valuable for simulating clinical trial

scenarios, training machine learning models, and testing clinical systems without compromising patient confidentiality.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm

# Generate 100 random age values between 18 and 75
#np.random.seed()
ages = np.random.normal(loc=47, scale=9, size=200)
#ages = np.clip(ages, 18, 75)

# Create the histogram
sns.histplot(ages, bins=15, kde=False, color='skyblue', stat='density')

# Add the normal distribution curve
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(ages), np.std(ages))
plt.plot(x, p, 'k', linewidth=2)

# Add labels and title
plt.xlabel('Age')
plt.ylabel('Density')
plt.title('Synthetic Data')

# Show the plot
plt.show()
```





The graph above compares the distribution of the original and synthetic data generated using the Faker library. It successfully matches the original data's distribution, demonstrating Faker's effectiveness in preserving statistical properties while generating synthetic data

CONCLUSION

In conclusion, Python has become a vital tool in clinical trials, improving collaboration, data management, and automating key tasks. With its strong libraries, Python helps research teams manage large amounts of data, analyse it quickly, and generate reports in real-time. The use of Artificial Intelligence and Machine Learning adds even more power to Python, enabling predictions, spotting patterns, and improving trial processes.
By automating tasks like clinical document updates and data extraction, Python reduces the need for manual work, boosts accuracy, and ensures compliance with regulations. As more clinical trials adopt Python, it will continue to speed up drug development and improve patient care. The future of clinical trials will benefit from Python's growing role in making research more efficient, scalable, and innovative.

REFERENCES

- **https://support.sas.com/en/software/saspy.html**
- **https://phdservices.org/python-in-clinical-research/**
- **Synthetic data generation methods in healthcare: A review on open-source tools and methods - ScienceDirect**

ACKNOWLEDGMENTS

We would like to express our heartfelt gratitude to Geninvo Technology Pvt Ltd and PHUSE for providing the opportunity to contribute to this paper, as well as for their continuous support and encouragement throughout the research process. We also extend our sincere appreciation to all those who contributed to the completion of this paper, with special thanks to our colleagues for their valuable feedback and support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged

Contact the author at: **sohan.l225@geninvo.com**
Author Name: Sohan Lal
Company: Geninvo technologies pvt. ltd
Address: 1408 East Empire Street 61701
Bloomington (Illinois), United States.
Work Phone: +1 309-807-5006
Email: sohan.l225@geninvo.com
Website: GENINVO is the go-to partner for life science industry.

Contact the Co-author at: **shweta.s1@geninvo.com**
Co-Author Name: Shweta Shukla
Company: Geninvo technologies pvt. ltd
Address: 1408 East Empire Street 61701
Bloomington (Illinois), United States.
Work Phone: +1 309-807-5006
Email: shweta.s1@geninvo.com
Website: GENINVO is the go-to partner for life science industry.