NEST REPORT

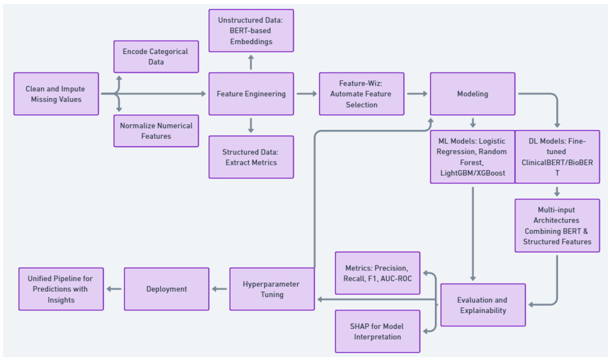
Problem Statement : Predicting Completion of Clinical Trials with Explainability

Authors :

- Yuvika Mishra (yuvikasfs@gmail.com)
- Siddhant Nijhawan (siddhantnijhawan111@gmail.com)

Date: 26-01-2025

Vision that was planned :



Clean and Impute Missing Values
Encode Categorical Data
Normalize Numerical Features
Unstructured Data: BERT-based Embeddings
Feature Engineering
Structured Data: Extract Metrics
Feature-Wiz: Automate Feature Selection
Modeling
ML Models: Logistic Regression, Random Forest, LightGBM/XGBoost
DL Models: Fine-tuned ClinicalBERT/BioBERT
Multi-input Architectures Combining BERT & Structured Features
Evaluation and Explainability
Metrics: Precision, Recall, F1, AUC-ROC
SHAP for Model Interpretation
Hyperparameter Tuning
Deployment
Unified Pipeline for Predictions with Insights

INTRODUCTION

Business Context

While designing and writing a clinical trial protocol document, multiple empirical, scientific and medical references are used. Like any other hypothesis testing, it is helpful in gaining insights from similar historical research / experiments / clinical studies from the past to minimize chances of failure and improve predictability, quality & execution. Authors of protocol documents would benefit from accurately identifying the right criteria for patients and accelerating patient recruitment for the clinical trial. Time taken for actual enrollment prediction during protocol development will provide valuable insights into whether the criteria should be more restrictive or broad, given past clinical trials data (e.g., avoid overly restrictive criteria that hinder recruitment, but also avoid criteria so broad that they fail to adequately test drug efficacy).

Problem Statement

Predict trial completion status ("Completed" vs "Not Completed") based on structured and unstructured data

Objectives

- Build a predictive model using structured and unstructured data.
- Apply causal inference techniques to identify key factors influencing trial outcomes.

Methodology

Data Handling

To prepare the dataset for analysis, we undertook the following structured approach to merge, clean, and transform the data from multiple sources:

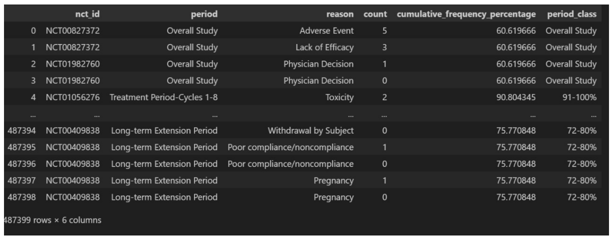
1. **Data Sources**:

   We worked with one CSVfile and four txt files containing clinical trial data. The key identifier across all files was the `NCT ID`, which uniquely identifies each clinical trial.

2. Column Handling:

   - Based on our research and domain intuition, we carefully selected relevant columns from each file, focusing on those that were most likely to impact the `Study Status` (our target variable).
   - We generated pivot tables to analyze relationships between features and `Study Status`, ranking features by their contribution to the target variable.
   - To focus on the most significant contributors, we retained the top 90% of features based on their relevance and grouped the remaining low-impact columns into 2–3 broad categories for simplicity and coherence. These grouped columns were subsequently encoded.
   - Also while dealing with numerical column and if same `NCT ID` have different numerical value we took maximum out of it to deal with worst case scenarios.

   For example :



```
             nct_id                         period                         reason  count  cumulative_frequency_percentage  period_class
     0  NCT00827372                  Overall Study                  Adverse Event      5                        60.619666  Overall Study
     1  NCT00827372                  Overall Study               Lack of Efficacy      3                        60.619666  Overall Study
     2  NCT01982760                  Overall Study             Physician Decision      1                        60.619666  Overall Study
     3  NCT01982760                  Overall Study             Physician Decision      0                        60.619666  Overall Study
     4  NCT01056276  Treatment Period-Cycles 1-8                        Toxicity      2                        90.804345       91-100%
   ...          ...                            ...                            ...    ...                              ...           ...
487394  NCT00409838   Long-term Extension Period          Withdrawal by Subject      0                        75.770848        72-80%
487395  NCT00409838   Long-term Extension Period  Poor compliance/noncompliance      1                        75.770848        72-80%
487396  NCT00409838   Long-term Extension Period  Poor compliance/noncompliance      0                        75.770848        72-80%
487397  NCT00409838   Long-term Extension Period                      Pregnancy      1                        75.770848        72-80%
487398  NCT00409838   Long-term Extension Period                      Pregnancy      0                        75.770848        72-80%

487399 rows × 6 columns
```

An Anecdote: Tackling the Adverse Events Column

While clustering, we faced a major challenge: the **adverse_event_term** column had over 100,000 unique categorical values. Traditional methods like one-hot encoding or probability distributions were infeasible. After hours of research, we devised a solution leveraging **pre-trained Transformers** like BERT to encode these terms into numerical representations.

Our approach involved two stages:

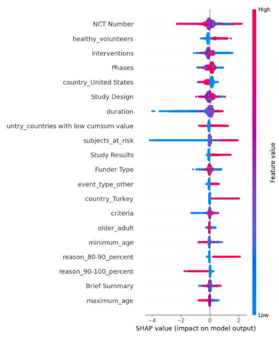
1. **Clustering Similar Entries**: Using Transformers, we generated embeddings for the adverse events, reduced their dimensions using **UMAP**, and applied **HDBSCAN** for clustering. This allowed us to preserve both global and local relationships in the data.
2. **Consolidating Clusters**: We validated the clusters manually, ensuring diseases like migraines or cancers were grouped logically. Combining results from multiple Transformers with a voting mechanism further improved accuracy.

3. **Data Transformation**:

   - We transformed our **long-format data** into a **wide format**, ensuring that each text file was aggregated to a unique level with respect to the `NCT ID`. This ensured that all features were aligned and deduplicated.
   - Once all text files were normalized to a unique `NCT ID` level, we used the main CSV file as the **base dataset** and merged the text files **to the right**, using `NCT ID` as the key for the joins.

This preprocessing approach not only ensured a consistent and complete dataset but also reduced noise and dimensionality, making it suitable for further analysis.

4. Exploratory Data Analysis
    - **Probability Distributions and Histograms**:
        - For numerical features, we plotted probability distributions and histograms to identify skewness, outliers, and patterns within the data.
        - These visualizations helped us assess variability and determine appropriate transformations, such as log-scaling, to handle skewed distributions effectively.
    - **Univariate Analysis**:
        - We analyzed each feature individually to evaluate its relationship with the target variable (`Study Status`).
    - SHAP is used in EDA to explain model predictions by visualizing feature importance.



5. Handling Missing Data
    - **Numerical Columns**:
        - Missing values in numerical columns were filled with the **median** of the respective feature.The median is robust to outliers and provides a representative central value.
    - For categorical columns, missing values were filled with a placeholder label `"Unknown_<column_name>"`.
    - **Encoded Columns**:
        - Missing values in already encoded columns were filled with `-1`.
6. Achieving 1,500 Encoded Columns: A Structured Approach

    To transform our dataset into a fully encoded format with **1,500 columns**, we followed a methodical and data-driven process, guided by domain requirements and best practices for handling mixed cardinality features. Here's how we approached it:

    Cardinality-Based Strategy

    To handle the remaining columns, we categorized them based on cardinality and applied targeted encoding techniques:

    - **Low Cardinality**
        - Used **Label Encoding** and **Multi-Label Encoding**, converting categories into numerical values while retaining their inherent order (where applicable).
    - **Medium Cardinality (e.g., 10–100 unique values)**:
        - Applied **Feature Hashing**, an efficient method to convert categorical data into fixed-length numerical vectors .
    - **High Cardinality (e.g., >100 unique values)**:

    - Used **TF-IDF (Term Frequency-Inverse Document Frequency)** to encode text-heavy or highly diverse categorical columns as it captures the significance of terms relative to the dataset

7. Causal Inference
    a. We have created DAG before doing feature engineering to extract causation for feature against the target variable.

Feature Engineering

To process our large dataset of **250,000 rows and 1,500 columns**, we leveraged a systematic approach focused on reducing dimensionality while maintaining data integrity, especially given the sensitivity of healthcare data.

1. **Initial Feature Reduction with LightGBM**:
    - We trained a **vanilla LightGBM model** to identify features contributing to **95% of the data's variance**. LightGBM is well-suited for high-dimensional data due to its speed, scalability, and ability to handle encoded categorical features effectively.
    - Hyperparameters chosen:
        - **n_estimators=100**: To prevent overfitting while maintaining model performance.
        - **learning_rate=0.05**: A conservative rate to ensure stable updates for sensitive healthcare data.
        - **random_state=42**: To maintain reproducibility.
    - Results: The number of features was reduced from **1,500 to ~300**, making the dataset much more manageable for downstream tasks.
2. **Decision Not to Use Featurewiz**:
    - Initially, we planned to apply the **Featurewiz library** for further optimization. However, the feature selection achieved by LightGBM was already optimal, as validated by model performance and explainability metrics. Hence, Featurewiz was deemed unnecessary.
3. **Explainability with Feature Importance**:
    - Using **LightGBM's feature importance**, we identified the top contributors to `Study Status`.
    - The top 20 features were visualized and verified against domain knowledge to ensure alignment with healthcare context.

MODELLING

The dataset exhibited a significant class imbalance, with class 0 (Not Completed) being underrepresented compared to class 1 (Completed). To address this, we implemented a combination of techniques and models to improve the model's performance and ensure explainability.

We used multiple classifiers, including Logistic Regression, Random Forest, LightGBM, XGBoost, and more. Additionally, we used powerful tools like LIME and SHAP to explain model predictions and ensure transparency. Finally, we tuned models using both traditional hyperparameter tuning (RandomizedSearchCV) and advanced methods like Optuna and Bayesian optimization.

Instead of using techniques like SMOTE (Synthetic Minority Over-sampling Technique), we opted to assign **class weights** to handle the class imbalance. Specifically, we assigned higher weights to class 0 (Not Completed), which was underrepresented. The computed class weights were:

- Class 0: 0.582
- Class 1: 3.544

This approach was chosen to minimize the risk of overfitting to the majority class and improve model performance on the minority class.

We implemented and evaluated several machine learning models to tackle this classification task:

1. Logistic Regression

The Logistic Regression model was the first classifier we trained. We used the **class_weight** parameter to account for the imbalance in the data.

```
       precision    recall  f1-score   support
    0    0.95       0.82     0.88      44249
    1    0.41       0.76     0.54      7267
```

While the Logistic Regression model performed well overall, the recall for the minority class (class 1) was relatively high, showcasing the effectiveness of the class weight adjustment.

2. Random Forest Classifier

Next, we trained a Random Forest model with **balanced class weights** to account for the class imbalance. We used 100 trees as the initial baseline configuration.

```
       precision    recall  f1-score   support
    0    0.91       0.99     0.95      44249
    1    0.91       0.42     0.57      7267
```

Random Forest displayed a significant improvement in terms of accuracy, but the recall for class 1 remained suboptimal. Therefore, we decided to hyperparameter tune the Random Forest to further enhance performance.

3. Hyperparameter Tuning for Random Forest

To fine-tune the Random Forest model, we used **RandomizedSearchCV**. We explored various combinations of hyperparameters to find the best set of values.

{'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_depth': None, 'bootstrap': False}

Optimized Random Forest Performance:

- **Accuracy**: 91.18%
- **Precision**: 83.22%
- **Recall**: 49.21%
- **F1 Score**: 62.02%
- **AUC-ROC**: 89.12%

We observed significant improvements in **precision** and **recall** after hyperparameter tuning. The model performed much better, especially in terms of predicting the minority class.

4. LightGBM

We next employed **LightGBM**, a gradient boosting method known for its efficiency and effectiveness. We used **balanced class weights** and optimized hyperparameters to maximize performance.

```
       precision    recall  f1-score   support
    0    0.95       0.90     0.93      44249
    1    0.54       0.71     0.61      7267
```

LightGBM demonstrated a strong balance between **precision** and **recall**, making it an excellent candidate for this task.

5. XGBoost

We also explored **XGBoost**, another powerful boosting algorithm, and fine-tuned it using its class-weight parameter to deal with class imbalance.

```
       precision    recall  f1-score   support
    0    0.94       0.94     0.94      44249
    1    0.65       0.64     0.64      7267
```

XGBoost achieved high **accuracy** and **precision**, and its **AUC-ROC** score confirmed its robustness.

6. Hyperparameter Tuning with Optuna

Our final step involved applying **Optuna** and **Bayesian optimization** for hyperparameter tuning. Optuna offers an automatic way to optimize the hyperparameters, improving model performance beyond traditional grid searches. The reason **Optuna** often gives the best results lies in its use of advanced algorithms, such as **Tree-structured Parzen Estimator (TPE)** and **Gaussian Processes.**

We got he best accuracy with Trail 26.

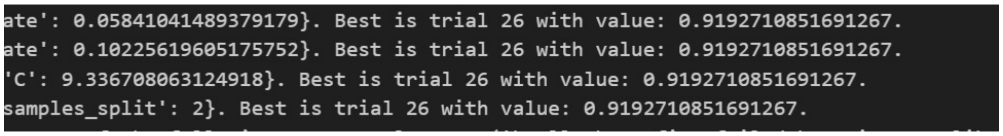
```
The Best was Trial 26
Trial 26 finished with value: 0.9192710851691267 and parameters:
{'classifier': 'xgboost', 'n_estimators': 238, 'max_depth': 8,
 'learning_rate': 0.05841041489379179}. Best is trial 26 with value: 0.9192710851691267.
```

The **XGBoost** model (OPTUNA Tuned) has shown an impressive performance after optimization. Here's a summary of the results and their implications:

Key Metrics:

- **Accuracy**: 91.88%
- **AUC-ROC Score**: 0.8909

These numbers indicate that the model is performing well overall. The **AUC-ROC** score of **0.8909** suggests that the model has a solid capability to distinguish between the two classes (Class 0 and Class 1), with a good trade-off between sensitivity and specificity.



```
ate': 0.05841041489379179}. Best is trial 26 with value: 0.9192710851691267.
ate': 0.10225619605175752}. Best is trial 26 with value: 0.9192710851691267.
'C': 9.336708063124918}. Best is trial 26 with value: 0.9192710851691267.
samples_split': 2}. Best is trial 26 with value: 0.9192710851691267.
```

Explainability

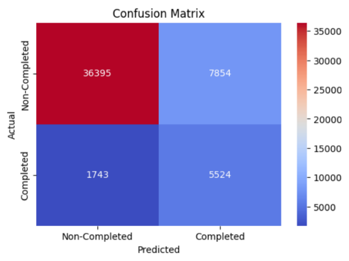
To ensure the models were interpretable, we used **LIME** (Local Interpretable Model-Agnostic Explanations) to explain the predictions of our models. LIME helped us understand the factors contributing to a given prediction, offering valuable insights into model behavior.

We have also tried using BERT, RNN,MLP AND TabNetbut at base level they werent providing with satisfactionable result but on hyperparameter tuning them , our computer wasnt supporting computationally.

Visualization

We have created confusion matrix, ROC-AUC curve , Lime graphs, correlation matrix and many more visualization techniques to make our project more understandable.

Example:



CONCLUSION

1. Key Takeaways

- **XGBoost** model (OPTUNA Tuned) gave us with the best results
- We never knew merging and handling data set was biggest task in life of data scientist
- Explainability ensures transparency and trust in predictions.

2. Limitations

- Imbalanced data may introduce biases despite oversampling.
- Our laptop needs some recovery after this project.

3. Future Work

- Deploying this project and making pipeline
- Explore alternative causal inference techniques like Difference-in-Differences (DID).

4. Impact

- Practical benefits for researchers and sponsors in improving clinical trial success rates and optimizing R&D investment