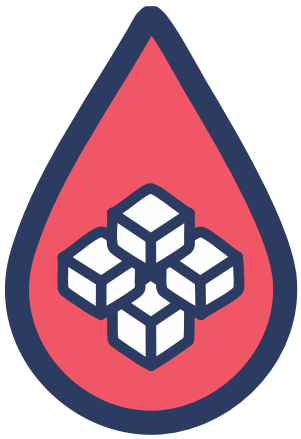
Diabetes
Data Preprocessing Report

Prepared by

[Team 10 – Mellitus]



MELLITUS

Date: 22/11/24

Table of Contents

1. INTRODUCTION 3

1.1. Background 3
1.2. Objective 3
1.3. Scope 3
1.4. RAID 4

2. DETAILED PROCESS 6

2.1. Data Flow Diagram 6
2.2. Preprocessing Architecture 6
2.3 High Level Workflow 8
2.4. File upload to Snowflake 8
2.5. Pipeline Stages 8
2.5.1. Wrapper (diabetes_preprocessing_main.py) 9
2.5.2. Data Cleansing (diabetes_stage_one.py) 9
2.5.3. Data Transformation (diabetes_stage_two.py) 13
2.5.4. Data Validation (diabetes_stage_three.py) 14

3. DESIGN DECISIONS SUMMARY 20

4. CONCLUSION 23

5. APPENDIX 24

1. Introduction

1.1. Background

Diabetes is a complex, chronic endocrine disorder characterized by elevated blood glucose levels resulting from insulin deficiency or resistance. It affects millions of people worldwide and poses significant health risks, including cardiovascular disease, neuropathy, nephropathy, and retinopathy. November is recognized as Diabetes Awareness Month, emphasizing the importance of understanding and combating this pervasive condition.

Considering the growing prevalence of diabetes and in honour of Diabetes Awareness Month, a leading pharmaceutical company has secured funding to develop a novel therapeutic product targeting diabetes management. To support this initiative, they have commissioned Mellitus, a team of data enthusiasts, to analyse the impact of diabetes on the human body. The company has provided a comprehensive diabetes dataset for this purpose.

The first major step in this project is to develop a robust preprocessing method for the data. Effective data preprocessing is crucial for producing relevant insights that can inform the client's product development strategy and enhance the understanding of the disease.

1.2. Objective

The primary objective of this project is to develop an effective data preprocessing pipeline tailored to the provided diabetes dataset. This pipeline aims to clean, prepare, and validate the data to ensure it is suitable for analysis and can produce meaningful insights about the impact of diabetes on the human body.

1.3. Scope

The scope of this project focuses on the data preprocessing phase, which includes:

1. Data Loading and Exploration:
    - Importing the raw diabetes dataset.
    - Understanding the structure and content of the data.
2. Data Cleaning:
    - Handling duplicates to ensure data uniqueness.
    - Addressing missing values using appropriate imputation techniques.
    - Enforcing consistent data types for all variables.

3. Data Preparation:
    - Performing feature engineering to create new informative variables.
    - Encoding categorical variables into numerical formats.
    - Scaling numerical features for consistency.
    - Adding identification and metadata columns for data management.
4. Data Validation:
    - Applying validation rules to ensure data integrity.
    - Identifying and handling invalid data entries.
5. Decision Justifications:
    - Providing a detailed report outlining the preprocessing steps taken.
    - Justify each key decision made during the preprocessing.

1.4. RAID

Risks

1. Data Quality:
    - The dataset may have significant missing or inconsistent data that could complicate preprocessing.
    - Potential biases in the data might affect the reliability of insights.
    - Data imbalance.
2. Technical Challenges: Computational limitations for modelling could hinder the preprocessing process, or software compatibility issues.

Assumptions

1. Data Completeness:
    - The provided dataset is sufficient and representative for the intended analysis.
    - Reflects real life patterns despite being synthetic data.
2. Tool Availability:
    - Necessary tools and libraries (e.g., Python, pandas, scikit-learn) are accessible and environment is setup for use.
3. Client Support:
    - The client will be available to provide clarifications if needed.

Issues

1. Data Privacy:
    - Ensuring compliance with data protection regulations during data handling.
2. Resource Constraints:

- Limited computational resources may affect processing speed and efficiency.

3. Limited 2-week delivery window:
    - Limited time to deliver full E2E solution.

Dependencies

1. Data Access:
    - The project relies on timely access to the diabetes dataset provided by the client.
    - The project relies on the definitions of the attributes in the Datathon Problem Statement.
2. Technological Tools:
    - Dependence on IT team to ensure specific programming languages and libraries for data preprocessing tasks are available in Mellitus suite.

By focusing on developing a robust data preprocessing method, this project sets the foundation for meaningful data analysis and insight generation. The preprocessing scripts are meticulously designed to address the unique challenges presented by the diabetes dataset, ensuring that the data is clean, consistent, and ready for exploration. The following sections of this report will delve into the specific preprocessing steps taken, including detailed justifications for each decision, to provide transparency and support for the methodologies employed.

2. Detailed Process

2.1. Data Flow Diagram (DFD)

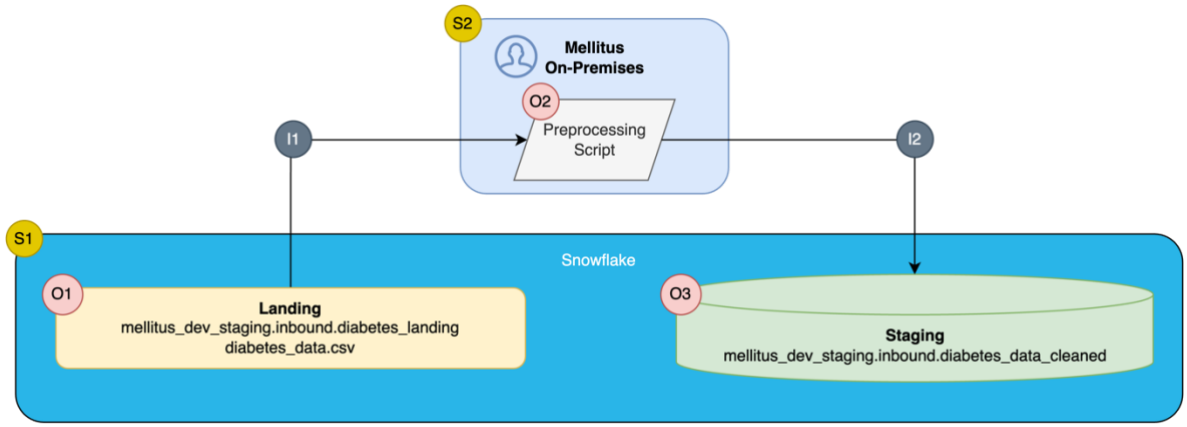


Figure 1: Data Flow Diagram of Preprocessing from Landing to Staging

Table 1: System, object and interface descriptions of DFD

ID	Type	Name	Description
S1	System	Snowflake Data Warehouse	Stores all raw/cleaned/curated data for Mellitus (Diabetes Data)
S2	System	Mellitus On-Premises	Runs Preprocessing Application
O1	Object	Landing	File staging for inbound files
O2	Object	Preprocessing Script	Set of python scripts for preprocessing
O3	Object	Staging	Database staging for inbound data
I1	Interface	Landing to Preprocessing	Data load from diabetes data file into DataFrame within preprocessing
I2	Interface	Preprocessing to Staging	Data write into staging from preprocessing script

2.2. Preprocessing Architecture

The preprocessing architecture is built as a framework to be reuseable for future data sources, with building to handle varying data structures, data types, file formats, etc through configuration of parameters within the framework. As such, future feeds can be setup quickly and the design patterns within the frameworks ensures appropriate standardisation is maintained.

The solution is split into two parts:

1. **Phase 1: Transition State** – built and deployed in this initial preprocessing discovery (design, build, and test phases completed)
    a. Loads data file from Snowflake Landing into DataFrame

b. Delivers Data Preprocessing Pipeline with 3 sub-stages:
    i. Data Cleansing
    ii. Data Transformation
    iii. Data Validation
c. Exports cleaned data as CSV
d. Loads cleaned data into Snowflake Staging Database

2. **Phase 2: End State** – to be deployed as part of core data delivery post discovery (design completed, build/deployment pending)
    a. External Datasets for enrichment of data and other variables
    b. Preprocessing DB for the following data:
        i. Configuration
        ii. Audit Log
        iii. Alert
    c. Snowflake Store Database for storing raw data files after processing is completed

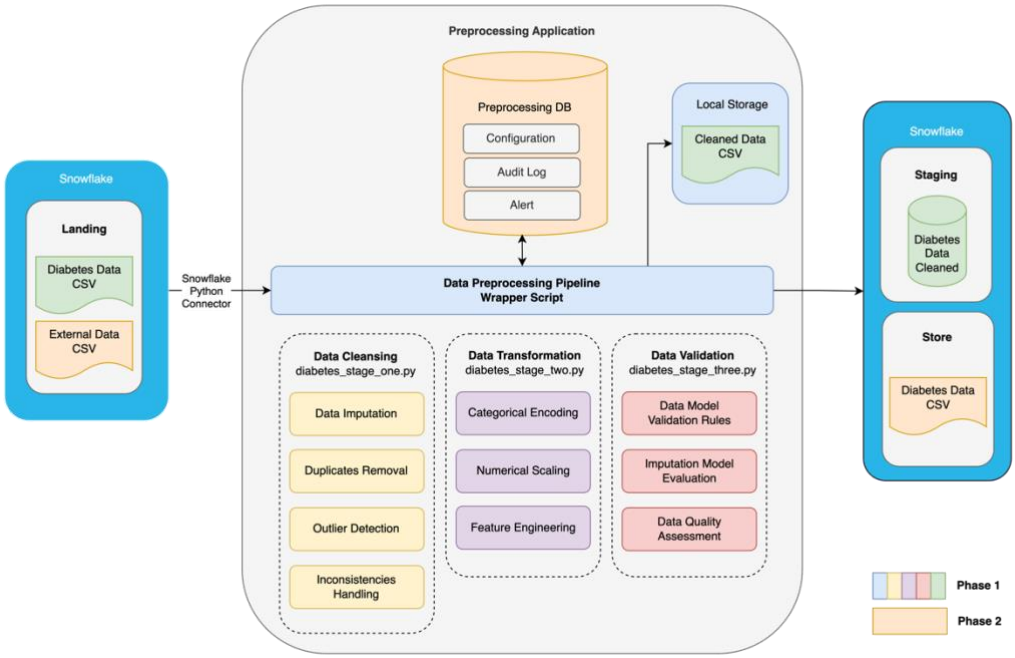


Figure 2: High Level Preprocessing Framework Architecture

2.3 High Level Workflow

1. Raw data is uploaded into the Landing area in Snowflake.
2. The Data Preprocessing Pipeline pulls data from Snowflake using the Snowflake Python Connector based on the feed configuration.
3. The pipeline performs three stages:
    a. Cleansing (handles nulls, duplicates, outliers, etc.).
    b. Transformation (encoding, scaling, and feature engineering).
    c. Validation (quality checks and rule validation).
4. Preprocessed cleaned data is exported in Local Storage.
5. Cleaned data is uploaded to Staging in Snowflake for further processing and reporting.
6. Once preprocessing successfully completes, raw data file is moved to the Store in Snowflake for archival

2.4. File upload to Snowflake

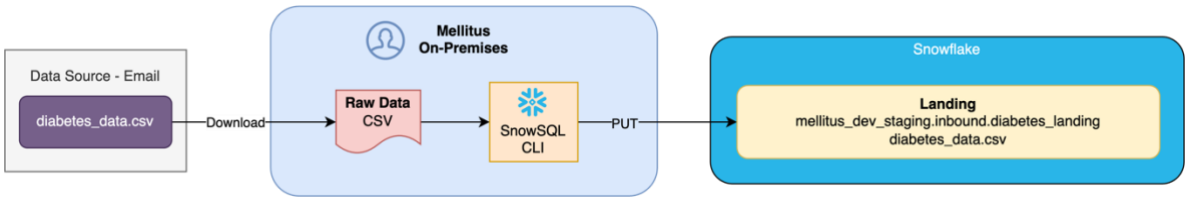


Figure 3: Architecture pattern for file upload to Snowflake landing

This data flow diagram illustrates the process of transferring diabetes-related data securely. It begins with a data source sending the raw file diabetes_data.csv via email. The file is then downloaded and processed on the Mellitus On-Premises system. To ensure security, credentials for accessing Snowflake are stored as environment variables in a configuration file. Using the SnowSQL CLI (a command-line tool for Snowflake), the diabetes_data.csv file is uploaded (via a PUT command) into the Landing Zone in Snowflake, specifically into the table mellitus_dev_staging.inbound.diabetes_landing. This workflow ensures that sensitive data is securely handled and transferred for further processing in Snowflake.

2.5. Pipeline Stages

Data preprocessing pipeline is architected into three main stages, each encapsulated within its own script, and orchestrated by a wrapper script. Below is a summary of the architectural components with a focus on the three scripts and the wrapper.

2.5.1. Wrapper (diabetes_preprocessing_main.py)

Purpose:

Acts as the orchestrator for the entire data preprocessing pipeline.

Key Functions:

1. Data Loading & Inspection:
   - Loads the raw dataset (diabetes_data.csv) into a pandas DataFrame.
   - Displays the first five rows and summarizes missing data for initial exploration.
2. Pipeline Execution:
   - Calls functions from the three stage scripts in sequence:
     a. Data Cleansing (Stage One): Cleanses and imputes missing data.
     b. Data Preparation (Stage Two): Encodes, scales, and engineer's features.
     c. Data Validation and Model Evaluation (Stage Three): Validates data and evaluates the machine learning model.
3. Data Saving:
   - Saves intermediate cleaned data after each stage (cleaned_data_stage_one.csv, cleaned_data_stage_two.csv).
   - Saves the final cleaned data (cleaned_data_final.csv).
   - Saves the trained model (random_forest_classifier.joblib) and scaler (scaler.joblib).
4. Error Handling:
   - Checks for file existence and handles exceptions like FileNotFoundError.

2.5.2. Data Cleansing (diabetes_stage_one.py)

Purpose:

- Cleanses the raw data by handling duplicates and missing values using various imputation techniques.

Key Functions:

1. Duplicate Removal:
   - Eliminates duplicate rows to ensure data uniqueness and reports the number of duplicates removed.
2. Data Imputation Strategy Applied:
   - Applies hybrid approach, combination of following techniques:

    - random imputation,
    - K-NN imputation,
    - and Random Forest imputation.
3. Imputation Tracking:
    - Initializes an imputed_columns list for each row to track which columns have been imputed.
4. Data Type Enforcement:
    - Ensures all columns have consistent and appropriate data types after imputation.
5. Inconsistencies Handling (No inconsistencies e.g. capitation, spelling, etc found)
6. Outlier Detection (No outliers found)

Detailed Imputation Strategy

Each of the attributes has null values, as such we cannot just remove nulls entirely as we would lose large amount of data.

Table 2: Attribute null count analysis

#	Attribute	Null Count
1	gender	20046
2	age	19855
3	hypertension	19831
4	diabetes_pedigree_function	19880
5	diet_type	20061
6	star_sign	20194
7	BMI	20066
8	weight	19874
9	family_diabetes_history	20137
10	social_media_usage	20032
11	physical_activity_level	19968
12	sleep_duration	19937
13	stress_level	19976
14	pregnancies	19967
15	alcohol_consumption	20104
16	diabetes	19758

There is an option to remove the nulls from diabetes column as it is the primary target variable, however, instead we will follow an approach where all other variables will be imputed, and then diabetes null values will be predicted using the trained model on the valid data. Once predicted, the predicted rows will be flagged to ensure tracking.

For high number of nulls (more than 7), we can remove these from the dataset for consideration as these have significant missing data and would require a lot of imputation. This may lead to insights shifting because of incomplete data, and overfitting based on the model responses. On the other hand, this data can still be imputed, provided the imputed columns are tracked and can be filtered if required at a later stage.

Table 3: Count of rows with specified number of nulls

Number of Nulls	Number of Rows
0	2893
1	11224
2	21221
3	24621
4	19879
5	11930
6	5528
7	1980
8	574
9	130
10	19
11	1

In addition, we can handle all the null values using imputation methods.

Table 4: Detailed Imputation Strategy and Justifications

Attribute	Data Type	Imputation Strategy	Justification
Gender	Categorical Nominal	RandomForestClassifier	Captures complex relationships; better than mode or KNN for categorical data; handles non-linear interactions effectively.
Age	Numerical Continuous	KNN Imputer	Non-parametric; captures local patterns; better than mean/median or regression for complex relationships without assuming a specific functional form.
Hypertension	Numerical Binary	RandomForestClassifier	Models complex, non-linear interactions; superior to mode imputation and logistic regression; handles binary variables effectively.
Diabetes Pedigree Function	Numerical Continuous	KNN Imputer	Captures correlations with other features; better than mean/median or regression; adapts to data without parametric assumptions.
Diet Type	Categorical Nominal	RandomForestClassifier	Handles complex interactions between diet and other variables; better than mode or KNN imputation for categorical data.

Star Sign	Categorical Nominal	Random Imputation Based on Distribution	Preserves original distribution; suitable when low correlation with other variables; avoids bias introduced by mode or model-based imputations.
BMI	Numerical Continuous	KNN Imputer	Captures individual variations and relationships; better than mean/median or regression; non-parametric and flexible.
Weight	Numerical Continuous	KNN Imputer	Utilizes relationships with other variables; better than mean/median or regression; captures non-linearities without model specification.
Family Diabetes History	Numerical Binary	RandomForestClassifier	Models complex factors influencing family history; better than mode or logistic regression; handles binary outcomes effectively.
Social Media Usage	Categorical Nominal	Random Imputation Based on Distribution	Maintains category proportions; appropriate when weak correlations exist; avoids bias from mode or model-based imputations.
Physical Activity Level	Categorical Ordinal	RandomForestClassifier	Captures complex interactions; better than mode or ordinal regression; handles non-linear relationships in ordinal data effectively.
Sleep Duration	Numerical Continuous	KNN Imputer	Captures associations with other features like stress and activity; better than mean/median or regression; flexible for non-linear patterns.
Stress Level	Categorical Ordinal	RandomForestClassifier	Models complex relationships; superior to mode or ordinal regression; effectively handles non-linear interactions in ordinal data.
Pregnancies	Numerical Count	RandomForestRegressor	Handles non-linear relationships; better than mean/median or Poisson regression; does not assume specific distribution for count data.
Alcohol Consumption	Categorical Nominal	RandomForestClassifier	Captures complex interactions influencing consumption habits; better than mode or KNN; handles categorical variables effectively.
Diabetes	Numerical Binary	RandomForestClassifier after all other imputations	Utilizes all available information post-imputation; better than mode or logistic regression; handles complex, non-linear relationships in binary outcomes effectively.

In summary, the selected imputation strategies are tailored to each attribute's data type and the nature of their relationships with other variables. RandomForestClassifier and RandomForestRegressor are chosen for their ability to handle complex, non-linear relationships and interactions without requiring explicit modeling of these relationships.

They are robust to overfitting and can handle mixed data types, making them superior to simpler models like logistic regression, mean/mode imputation, or KNN in many cases.

KNN Imputer is selected for continuous variables where local patterns and similarities between instances are essential for accurate imputation. It is non-parametric and flexible, making it suitable when the data does not conform to specific distributions or when relationships are too complex for parametric models.

Random Imputation Based on Distribution is used for attributes with weak correlations to other variables, ensuring the original data distribution is preserved without introducing bias from more aggressive imputation models.

By carefully selecting the imputation strategy for each attribute, we ensure that missing data is filled in a way that maintains the integrity of the dataset, improves the accuracy of downstream analyses, and leverages the strengths of different imputation methods where they are most effective.

2.5.3. Data Transformation (diabetes_stage_two.py)

Purpose:

Prepares the cleansed data for modeling by encoding categorical variables, scaling features, and performing feature engineering.

Key Functions:

1. Feature Engineering:

   Creates new features:

   - age_group: Categorizes age into groups like young_adult, middle_aged, senior.
   - bmi_group: Categorizes BMI into underweight, normal, overweight, obese.
   - sleep_group: Categorizes sleep_duration into meaningful groups.
   - height: Calculated column using BMI and Weight

2. Categorical Encoding:
   - Encodes categorical variables using LabelEncoder.

3. Scaling Features:
   - Applies StandardScaler to numerical and encoded categorical columns.
   - Scaled columns are added as new features with _scaled suffix.

4. Adding Supplementing Columns and Rearranging:
    - Adds identification columns: file_id, row_id, and calculates height from BMI and weight.
    - Adds metadata columns: created_user, created_dttm, modified_user, modified_dttm.
    - Stores preprocessing failed flag and reason in preprocessing database
    - Rearranges columns to the desired order.

2.5.4. Data Validation (diabetes_stage_three.py)

This stage contains Data Validation, Data Quality Assessment, and Model Evaluation.

Purpose:

Validates the prepared data against defined rules and evaluates the performance of the machine learning model.

Key Components and Functions:

1. Defining Validation Rules:
    - Sets up rules for each column, specifying allowed values, data types, and whether the field is mandatory.

2. Handling Invalid Rows:
    - Removes rows that fail validation to prevent them from affecting the model.
    - Reports the number of invalid rows removed.

3. Model Training:
    - Splits data into training and testing sets using train_test_split, ensuring no data leakage.
    - Trains a RandomForestClassifier with balanced class weights.

4. Model Testing and Evaluation:
    - Makes predictions on the test set.
    - Calculates evaluation metrics:
        - Accuracy
        - ROC-AUC Score
        - Classification Report
        - Confusion Matrix

Validation Rules:

Validation rules were informed by medical knowledge (e.g., realistic ranges for BMI, age). This ensures that the data aligns with real-world expectations and that outliers or improbable values are identified and handled appropriately.

Table 5: Validation Rules for data based on type and domain

Field	Type	Allowed Values	Min	Max
gender	categorical	['male', 'female']		
age	numerical		0	120
hypertension	binary	[0, 1]		
diabetes_pedigree_function	numerical		0	2.42
diet_type	categorical	['vegetarian', 'vegan', 'low carb', 'mediterranean', 'standard american diet', 'gluten free', 'pescatarian', 'carnivore', 'free', 'paleo', 'raw food', 'ketogenic', 'atkins', 'weight watchers']		
star_sign	categorical	['aries', 'taurus', 'gemini', 'cancer', 'leo', 'virgo', 'libra', 'scorpio', 'sagittarius', 'capricorn', 'aquarius', 'pisces']		
BMI	numerical		10	70
weight	numerical		30	300
family_diabetes_history	binary	[0, 1]		
social_media_usage	categorical	['never', 'rarely', 'occasionally', 'moderate', 'excessive']		
physical_activity_level	categorical	['sedentary', 'lightly active', 'moderately active', 'very active', 'extremely active']		
sleep_duration	numerical		4	12
stress_level	categorical	['low', 'moderate', 'elevated', 'high', 'extreme']		

pregnancies	binary	[0, 1]		
alcohol_consumption	categorical	['none', 'light', 'moderate', 'heavy']		
diabetes	binary	[0, 1]		

Data Validation Results

1. BMI
    a. Sample invalid entries in 'BMI': [7.6 6.6 9.1 9.2 8.8]
    b. These BMIs are not physically possible. BMI has total of 188 invalid entries
    c. Preprocessing failed for these entries and flagged in preprocessing DB
2. Sleep duration
    a. Using 4–12-hour range validation window, 26368 invalid entries were found which is too much. It is possible below 4 hours sleep may not be invalid necessarily.
    b. As such, sleep duration validation rule was reduced to 0 to 14. It is physically possible to get lower than 4 hours sleep.

Alternatively, these invalid values can also be imputed and replaced.

Imputation Model Evaluation

Based on the initial exploratory analysis done, 6 important features were selected to evaluate the imputation model performance. These features were split and evaluated across K-NN (BMI, weight, sleep) and Random Forest Models (physical activity, alcohol, stress).

K-NN Evaluation

The KNN imputation appears to have preserved the overall distribution of the BMI variable, as the original and imputed densities closely overlap, particularly in the central range. However, the slight increase in peak density for the imputed dataset suggests that KNN might have introduced some minor smoothing or overestimation in central BMI values.

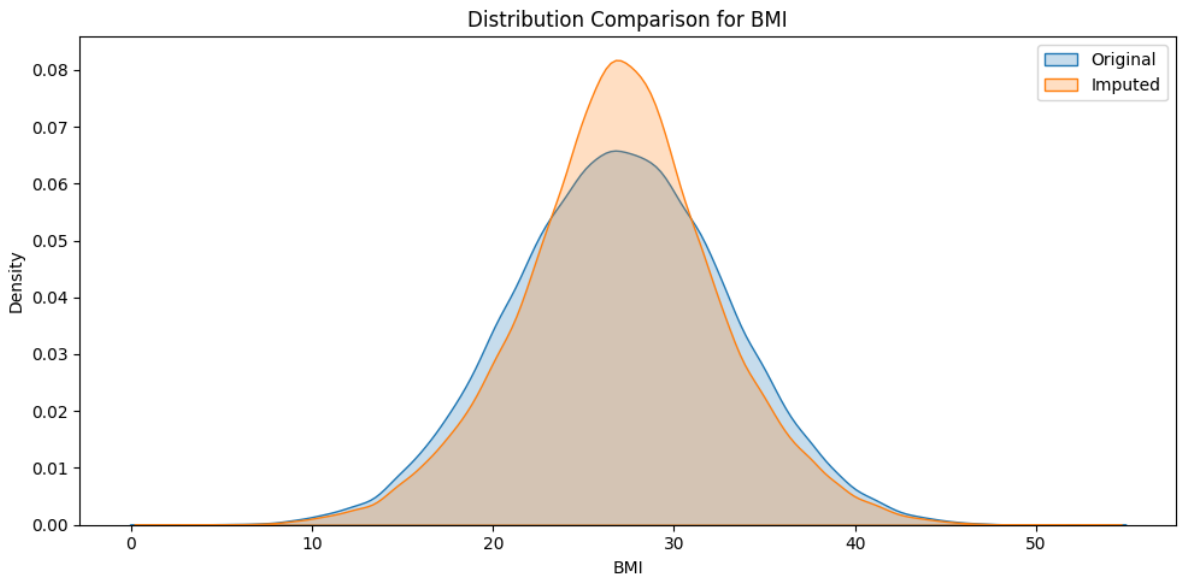


Figure 4: BMI distribution comparison for imputed and original data.

The KNN imputation for the weight variable shows good preservation of the overall distribution shape. However, the imputed dataset displays a slightly smoother and more centralized distribution, as indicated by a higher density near the mean (around 150) and reduced densities at the tails. This suggests that KNN imputation might have reduced variability by clustering values closer to the mean, potentially underrepresenting extreme weights.

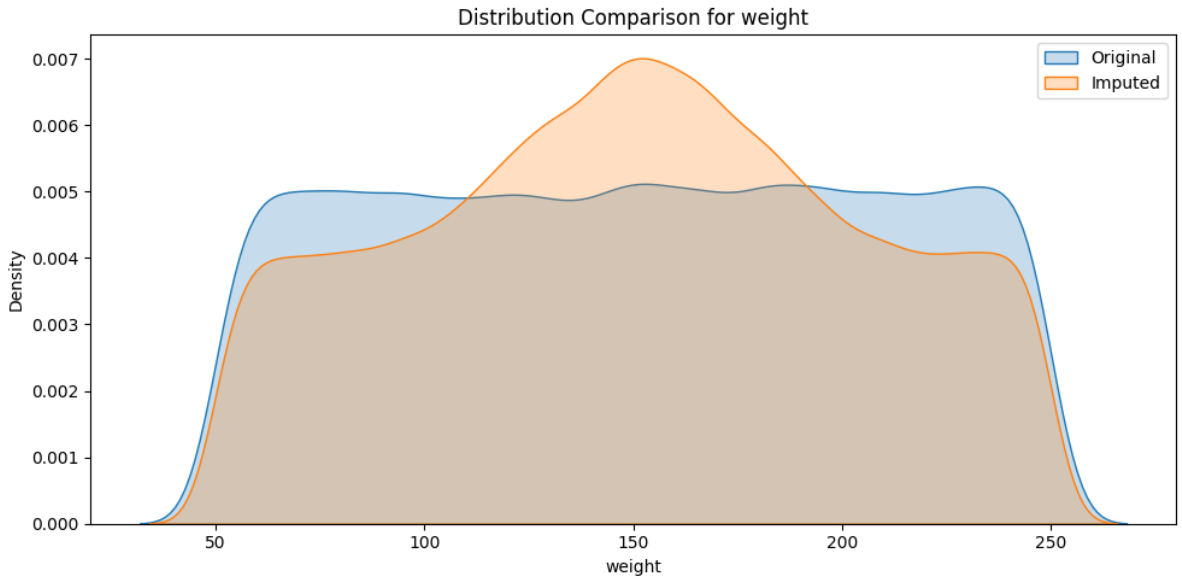


Figure 5: Weight distribution comparison for imputed and original data.

The KNN imputation for the sleep duration variable captures the general shape of the original distribution, preserving key features such as the peaks around 6-8 hours and a smaller peak near 12 hours. However, the imputed dataset shows a slightly smoother distribution, with higher density in the main peak (6-8 hours) and reduced variability at the tails. This indicates that KNN imputation may have slightly overemphasized common values while underrepresenting rare or extreme values, potentially losing some of the distribution's granularity.

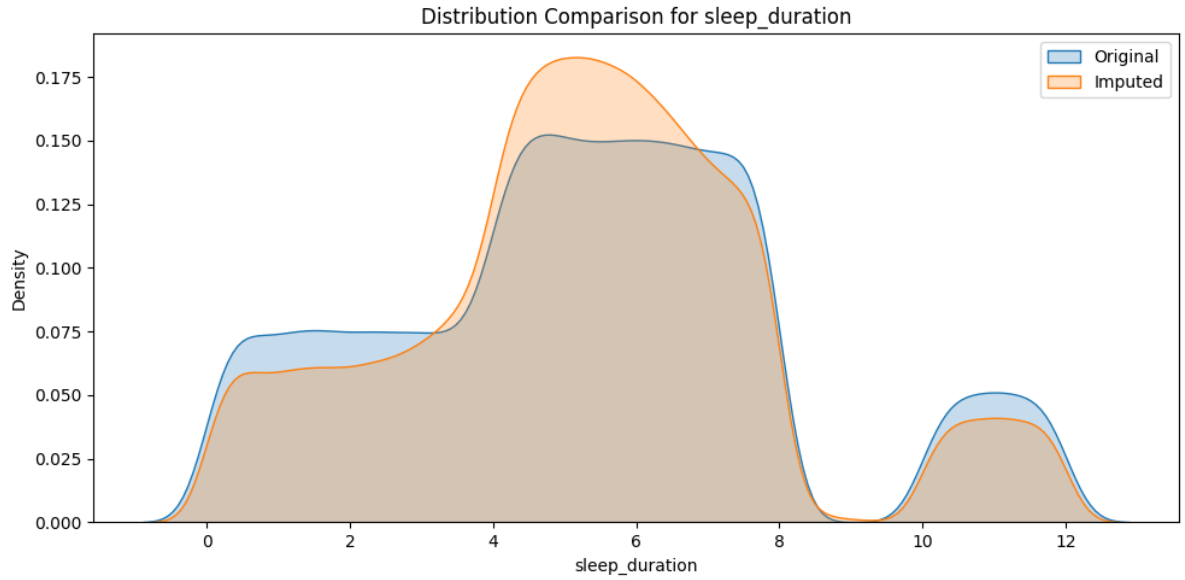


Figure 6: Sleep distribution comparison for imputed and original data

Random Forest Evaluation

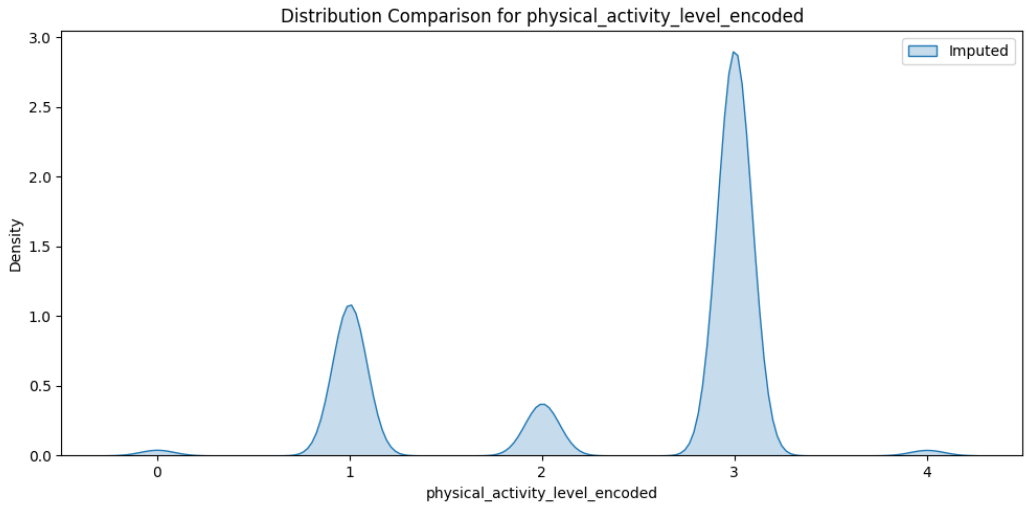


Figure 7: Physical Activity distribution for imputed data

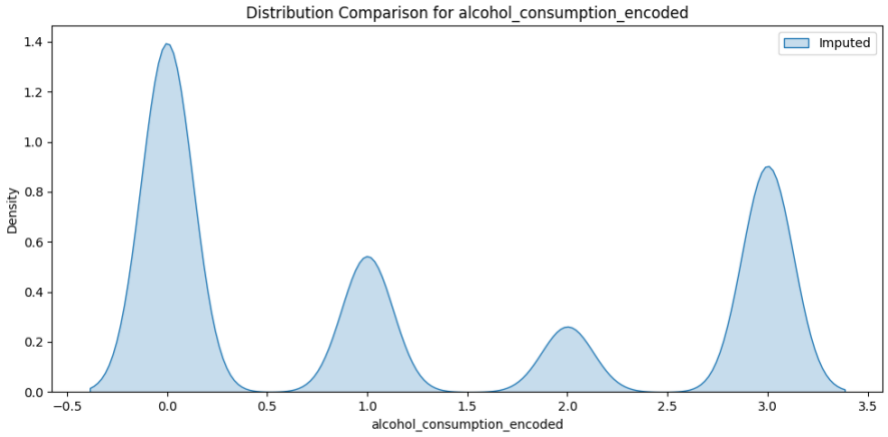


Figure 8: Alcohol consumption distribution for imputed data

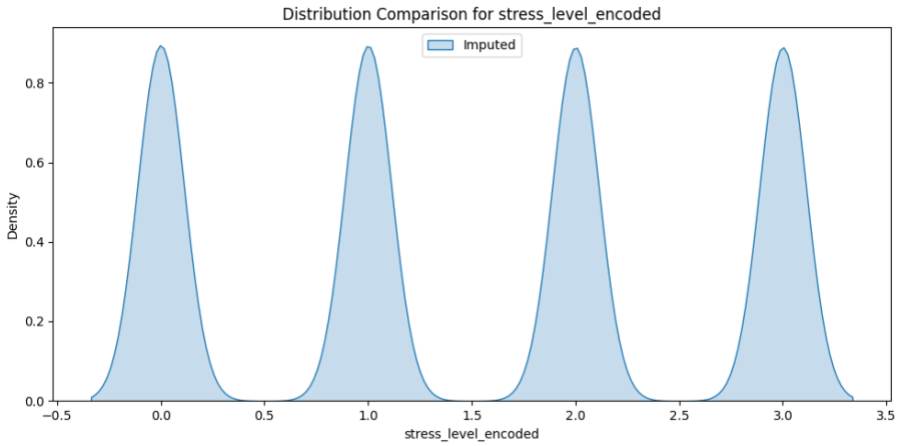


Figure 9: Stress level distribution for imputed data

As shown above, stress level is evenly distributed across each stress level. This aligns with the initial dataset. As such, we can have high confidence in this imputed attribute.

To improve the modelling, non-important features can be removed from the imputation modelling based on the advanced insights attained from the exploratory data analysis. In addition, various other models can be explored to identify which model is performing best.

3. Design Decisions Summary

In assessing the comprehensive strategy of the data preprocessing pipeline for the diabetes dataset, several key design decisions were made to ensure the effectiveness, reliability, and scalability of the preprocessing steps. The following table outlines these decisions along with their justifications.

Design Decision	Justification
Choice of Python Programming Language	Python was selected due to its extensive libraries for data manipulation and machine learning (e.g., pandas, scikit-learn), ease of use, readability, and strong community support. Python's versatility makes it ideal for developing complex data preprocessing pipelines efficiently.
Modular Structure with Wrapper Script and Three Stage Scripts	The pipeline was organized into a wrapper script (diabetes_preprocessing_main.py) and three stage scripts (diabetes_stage_one.py, diabetes_stage_two.py, diabetes_stage_three.py) to enhance modularity and maintainability. This separation of concerns allows for easier debugging, testing, and potential reuse of individual components.
Splitting the Pipeline into Three Stages	Dividing the pipeline into Data Cleansing (Stage One), Data Preparation (Stage Two), and Data Validation and Model Evaluation (Stage Three) aligns with standard data preprocessing workflows. This structure facilitates focused processing at each stage and provides clear checkpoints for data quality assessment.
Adoption of a Hybrid Imputation Strategy	A combination of imputation methods (KNN Imputer, Random Forest models, random imputation) was used to address different types of missing data effectively. This hybrid approach allows for tailored imputation based on the nature of each variable, improving the accuracy and reliability of the imputed values.
Tracking Imputed Columns	An imputed_columns list was maintained for each row to record which columns were imputed. This practice enhances transparency, allowing analysts to identify which data points were originally missing and imputed, which is crucial for interpretability and potential bias assessment.
Inclusion of a Diabetes Prediction Flag	The predicted_diabetes_flag column indicates rows where the diabetes outcome was imputed. This flag is important for downstream analysis to differentiate between actual and imputed outcomes, ensuring that insights derived are based on reliable data.

Retaining All Data Rows While Tracking Imputations	Keeping all data rows, including those with missing values, maximizes data utilization. By tracking imputations, the pipeline maintains data integrity and allows for comprehensive analysis without discarding potentially valuable information.
Training the Model Only on Clean Data	The machine learning model was trained exclusively on data that passed validation rules (preprocessed_flag == 'Y'). This ensures that the model learns from high-quality data, enhancing its predictive performance and reducing the risk of learning from erroneous or invalid entries.
Selection and Application of Validation Rules	Validation rules were defined based on domain knowledge and data characteristics to enforce data integrity. Applying these rules helps identify and handle invalid or inconsistent data, ensuring that the dataset used for modeling is accurate. BMI and sleep rules which failed had entries into the Preprocessing DB with their file and row id.
Use of Random Seed for Reproducibility	Setting a random seed (RANDOM_STATE = 42) ensures that all random processes (e.g., data splitting, random imputation) produce the same results every time the script is run. This reproducibility is essential for verifying results and debugging.
Feature Engineering Choices	New features such as age_group, bmi_group, and sleep_group were engineered to capture additional insights. These categorizations can reveal patterns and relationships that are not apparent from raw numerical values, enhancing the analytical value of the dataset.
Scaling and Encoding Decisions	Numerical features were scaled using StandardScaler, and categorical variables were encoded using LabelEncoder. Scaling ensures that features contribute equally to the model, while encoding converts categorical variables into a numerical format suitable for modeling algorithms.
Saving Intermediate Data Stages	Saving the data after each stage (cleaned_data_stage_one.csv, cleaned_data_stage_two.csv) allows for checkpointing and facilitates debugging. It enables the team to inspect the data at various processing stages, ensuring that each step performs as intended.
Use of Random Forest Models for Imputation	Random Forest algorithms were chosen for imputing missing values in both regression and classification contexts due to their robustness, ability to handle non-linear relationships, and effectiveness with mixed data types. This improves the accuracy of imputed values.

Error Handling	Implementing try-except blocks ensures that the pipeline runs smoothly and that any issues are caught and reported without halting execution. This user-friendly approach aids in maintaining the flow of processing.
Adding Metadata Columns	Columns like file_id, row_id, created_user, created_dttm, modified_user, and modified_dttm were added to enhance data management and traceability. These metadata columns support data governance practices and facilitate data lineage tracking.
Enforcing Data Types Post-Imputation	Data types were enforced after imputation to ensure consistency (e.g., converting ages to integers). This prevents type-related errors in subsequent analysis and ensures that statistical computations are accurate.
Incorporating Domain Knowledge in Validation Rules	Validation rules were informed by medical knowledge (e.g., realistic ranges for BMI, age). This ensures that the data aligns with real-world expectations and that outliers or improbable values are identified and handled appropriately.
Use of Advanced Imputation Techniques	Advanced imputation methods were employed instead of simple mean or median imputation. Techniques like KNN and Random Forest consider relationships between variables, resulting in more accurate and contextually appropriate imputations.

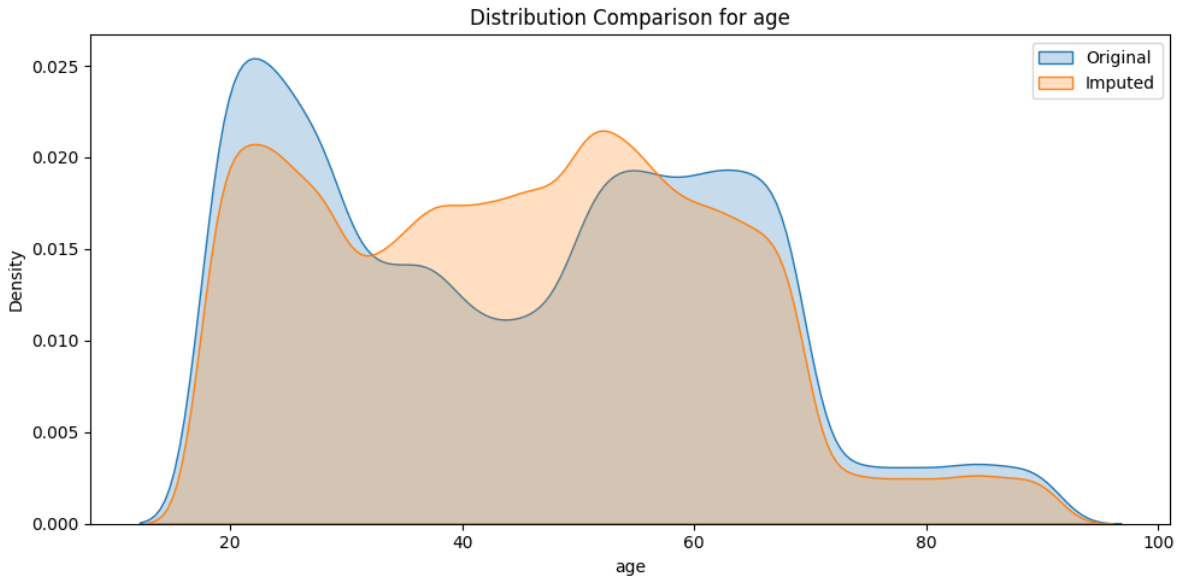
4. Conclusion

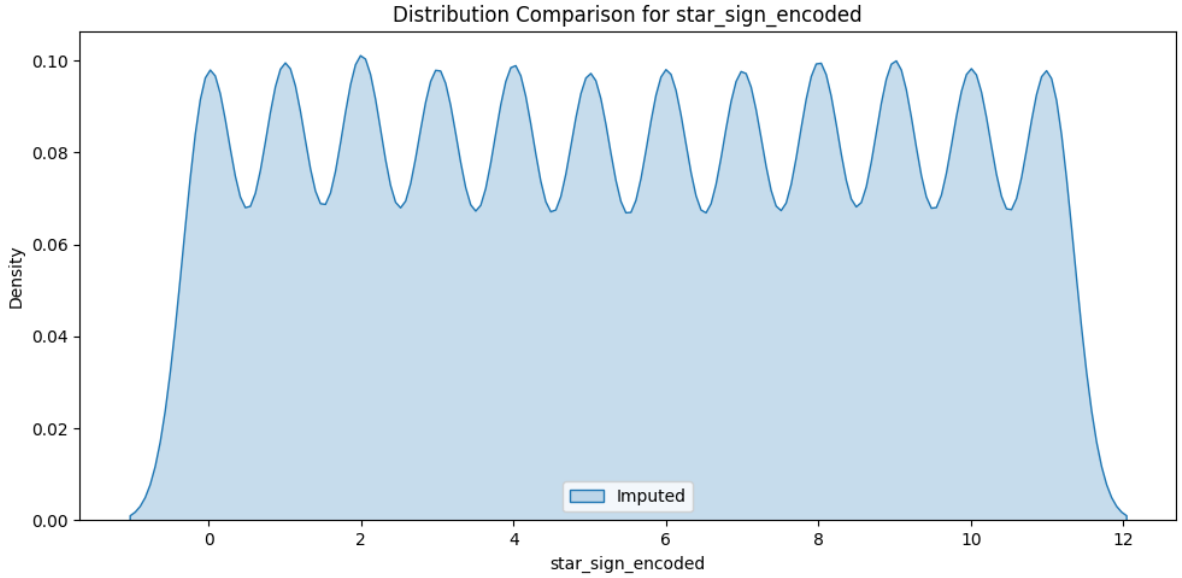
The preprocessing pipeline for the diabetes dataset has been carefully designed to ensure data quality, integrity, and optimal usability for downstream analysis and modelling. The selected imputation strategies—Random Forest models, KNN Imputer, and distribution-based random imputation—were chosen to address the specific challenges posed by different data types and relationships. Random Forest models excel at handling complex, non-linear interactions and mixed data types, making them ideal for imputing categorical and continuous variables in datasets with intricate dependencies. KNN Imputer complements this by capturing localized patterns in continuous variables, offering precision in scenarios where neighborhood relationships are critical. Distribution-based random imputation ensures that the dataset's original statistical properties are preserved, particularly for attributes with weaker correlations to other variables.

By integrating these advanced imputation methods, the pipeline maintains a high level of accuracy and interpretability, ensuring the reliability of subsequent analyses. The modular design of the pipeline facilitates efficient debugging, testing, and potential reuse in future projects, enhancing its scalability and adaptability. Validation rules based on domain knowledge reinforce data quality by filtering out improbable or inconsistent entries, while metadata tracking and imputed column flags ensure full traceability and transparency.

Finally, the decision to train the machine learning model exclusively on validated, high-quality data ensures that the model's predictive capabilities are optimized. By leveraging comprehensive feature engineering, robust scaling and encoding practices, and reproducibility measures like setting a random seed, the pipeline provides a strong foundation for reliable model evaluation and deployment. Together, these thoughtful design choices position the pipeline as a robust, scalable solution for data preprocessing and machine learning, contributing to meaningful insights and actionable outcomes in diabetes prediction.

5. Appendix


Distribution Comparison for age
Original
Imputed
Density
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
age


Distribution Comparison for star_sign_encoded
Imputed
Density
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12
star_sign_encoded