Data & AI: Datathon

Diabetes

Data Preprocessing Report

Prepared by

[Team 10 – Mellitus]



Date: 25/11/24

Table of Contents

[1. Introduction 4](#_Toc182386583)

[1.1. Background 4](#_Toc182386584)

[1.2. Objective 4](#_Toc182386585)

[1.3. Scope 4](#_Toc182386586)

[1.4. RAID 4](#_Toc182386587)

[2. Data Loading 4](#_Toc182386588)

[2.1. Purpose 4](#_Toc182386589)

[2.2. Implementation 4](#_Toc182386590)

[2.3. Justification 4](#_Toc182386591)

[3. Data Validation 4](#_Toc182386592)

[3.1. Purpose 4](#_Toc182386593)

[3.2. Implementation 4](#_Toc182386594)

[3.3. Justification 4](#_Toc182386595)

[4. Handling Missing Values with Predictive Modeling 4](#_Toc182386596)

[4.1. Purpose 4](#_Toc182386597)

[4.2. Implementation 4](#_Toc182386598)

[4.3. Justification 4](#_Toc182386599)

[5. Handling Inconsistencies and Duplicates 4](#_Toc182386600)

[5.1. Purpose 4](#_Toc182386601)

[5.2. Implementation 4](#_Toc182386602)

[5.3. Justification 5](#_Toc182386603)

[6. Handling Outliers 5](#_Toc182386604)

[6.1. Purpose 5](#_Toc182386605)

[6.2. Implementation 5](#_Toc182386606)

[6.3. Justification 5](#_Toc182386607)

[7. Encoding Categorical Variables 5](#_Toc182386608)

[7.1. Purpose 5](#_Toc182386609)

[7.2. Implementation 5](#_Toc182386610)

[7.3. Justification 5](#_Toc182386611)

[8. Feature Scaling 5](#_Toc182386612)

[8.1. Purpose 5](#_Toc182386613)

[8.2. Implementation 5](#_Toc182386614)

[8.3. Justification 5](#_Toc182386615)

[9. Feature Engineering with Interaction Metrics 5](#_Toc182386616)

[9.1. Purpose 5](#_Toc182386617)

[9.2. Implementation 5](#_Toc182386618)

[9.3. Justification 5](#_Toc182386619)

[10. Data Anonymization 5](#_Toc182386620)

[10.1. Purpose 5](#_Toc182386621)

[10.2. Implementation 5](#_Toc182386622)

[10.3. Justification 6](#_Toc182386623)

[11. Saving the Cleaned Dataset 6](#_Toc182386624)

[11.1. Purpose 6](#_Toc182386625)

[11.2. Implementation 6](#_Toc182386626)

[11.3. Justification 6](#_Toc182386627)

[12. Conclusion 6](#_Toc182386628)

[12.1. Summary of Steps 6](#_Toc182386629)

[12.2. Impact on Data Quality 6](#_Toc182386630)

[12.3. Next Steps 6](#_Toc182386631)

[References 6](#_Toc182386632)

[Appendices 6](#_Toc182386633)

[A. Code Snippets 6](#_Toc182386634)

[B. Additional Tables or Figures 6](#_Toc182386635)

# 1. **Introduction**

## 1.1. Background

## 1.2. Objective

## 1.3. Scope

## 1.4. RAID

# **2. Detailed Process**

## 2.1. Data Flow Diagram

A screenshot of a computer screen

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| 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)
   1. Loads data file from Snowflake Landing into DataFrame
   2. Delivers Data Preprocessing Pipeline with 3 sub-stages:
      1. Data Cleansing
      2. Data Transformation
      3. Data Validation
   3. Exports cleaned data as CSV
   4. 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)
   1. External Datasets for enrichment of data and other variables
   2. Preprocessing DB for the following data:
      1. Configuration
      2. Audit Log
      3. Alert
   3. Snowflake Store Database for storing raw data files after processing is completed

A diagram of data processing

Description automatically generated

**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.
3. The pipeline performs three stages:
   1. Cleansing (handles nulls, duplicates, outliers, etc.).
   2. Transformation (encoding, scaling, and feature engineering).
   3. 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

**File upload to Snowflake**

A close-up of a computer

Description automatically generated

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.

**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.

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.

1. Pipeline Execution:

* Calls functions from the three stage scripts in sequence:
  1. Data Cleansing (Stage One): Cleanses and imputes missing data.
  2. Data Preparation (Stage Two): Encodes, scales, and engineer’s features.
  3. Data Validation and Model Evaluation (Stage Three): Validates data and evaluates the machine learning model.

1. 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).

1. Error Handling:

* Checks for file existence and handles exceptions like FileNotFoundError.

1. **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.

1. Data Imputation Strategy Applied:

* Applies hybrid approach, combination of following techniques:
  + random imputation,
  + K-NN imputation,
  + and Random Forest imputation.

1. Imputation Tracking:

* Initializes an imputed\_columns list for each row to track which columns have been imputed.

1. Data Type Enforcement:

* Ensures all columns have consistent and appropriate data types after imputation.

1. Inconsistencies Handling
2. Outlier Detection

**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.

|  |  |  |
| --- | --- | --- |
| # | **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.

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.

|  |  |
| --- | --- |
| **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.

|  |  |  |  |
| --- | --- | --- | --- |
| **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.

1. **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

1. **Categorical Encoding:**

* Encodes categorical variables using LabelEncoder.

1. **Scaling Features:**

* Applies StandardScaler to numerical and encoded categorical columns.
* Scaled columns are added as new features with \_scaled suffix.

1. **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.
* Rearranges columns to the desired order.

1. **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.

1. **Handling Invalid Rows:**

* Removes rows that fail validation to prevent them from affecting the model.
* Reports the number of invalid rows removed.

1. **Model Training:**

* Splits data into training and testing sets using train\_test\_split, ensuring no data leakage.
* Trains a RandomForestClassifier with balanced class weights.

1. **Model Testing and Evaluation:**

* Makes predictions on the test set.
* Calculates evaluation metrics:
  + Accuracy
  + ROC-AUC Score
  + Classification Report
  + Confusion Matrix

Validation Rules:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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] |  |  |

# 

**Imputation Model Evaluation**

1. **Metrics**
2. **Key Visualisations**

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**

**A diagram of distribution of bmi

Description automatically generated**

A graph of a mountain range

Description automatically generated

**A graph of a distribution graph

Description automatically generated**

Random Forest

A graph of a diagram

Description automatically generated with medium confidence

A graph of a bar graph

Description automatically generated with medium confidence

A graph of a stress level

Description automatically generated

Conclusion

The selected imputation strategies are chosen based on their ability to handle the specific data types and capture the underlying relationships between variables more effectively than alternative methods. RandomForest models are preferred for their robustness and ability to model complex interactions without overfitting, while KNN Imputer is selected for its flexibility and effectiveness in capturing local data patterns in continuous variables. Random imputation based on distribution is used when preserving the original data distribution is crucial, particularly for attributes with weak correlations to other variables.

Appendix

**A graph of a distribution graph

Description automatically generated with medium confidence**

# **Data Cleansing**

## 3.1. Purpose

## 3.2. Implementation

## 3.3. Justification

# **3. Data Validation**

## 3.1. Purpose

## 3.2. Implementation

## 3.3. Justification

# **4. Handling Missing Values with Predictive Modeling**

## 4.1. Purpose

## 4.2. Implementation

## 4.3. Justification

# **5. Handling Inconsistencies and Duplicates**

## 5.1. Purpose

## 5.2. Implementation

## 5.3. Justification

# **6. Handling Outliers**

## 6.1. Purpose

## 6.2. Implementation

## 6.3. Justification

# **7. Encoding Categorical Variables**

## 7.1. Purpose

## 7.2. Implementation

## 7.3. Justification

# 8. Feature Scaling

## 8.1. Purpose

## 8.2. Implementation

## 8.3. Justification

# 9. Feature Engineering with Interaction Metrics

## 9.1. Purpose

## 9.2. Implementation

## 9.3. Justification

# 10. Data Anonymization

## 10.1. Purpose

## 10.2. Implementation

## 10.3. Justification

# 11. Saving the Cleaned Dataset

## 11.1. Purpose

## 11.2. Implementation

## 11.3. Justification

# 12. Performance Assessment

# 13. Conclusion

# References

# Appendices

## A. Code Snippets

## B. Additional Tables or Figures