Requirements Specification Assignment 3: KNIME Cleaning Data

1. Start up the KNIME tool on your personal computer.

   Downloaded the Cleaning Data workflow from KNIME Hub using the below link and imported the workflow in KNIME:

   https://hub.knime.com/adm/spaces/Public/Workflows/Data%20Analytics%20Made%20Easy/Chapter%202/Cleaning%20data~DLJtVEX0f9gQTCS7/current-state

2. Execute and inspect the Cleaning Data workflow on your personal computer.

   We have executed each node of Cleaning Data workflow and inspected the input and output. Attaching the below screenshot of workflow.

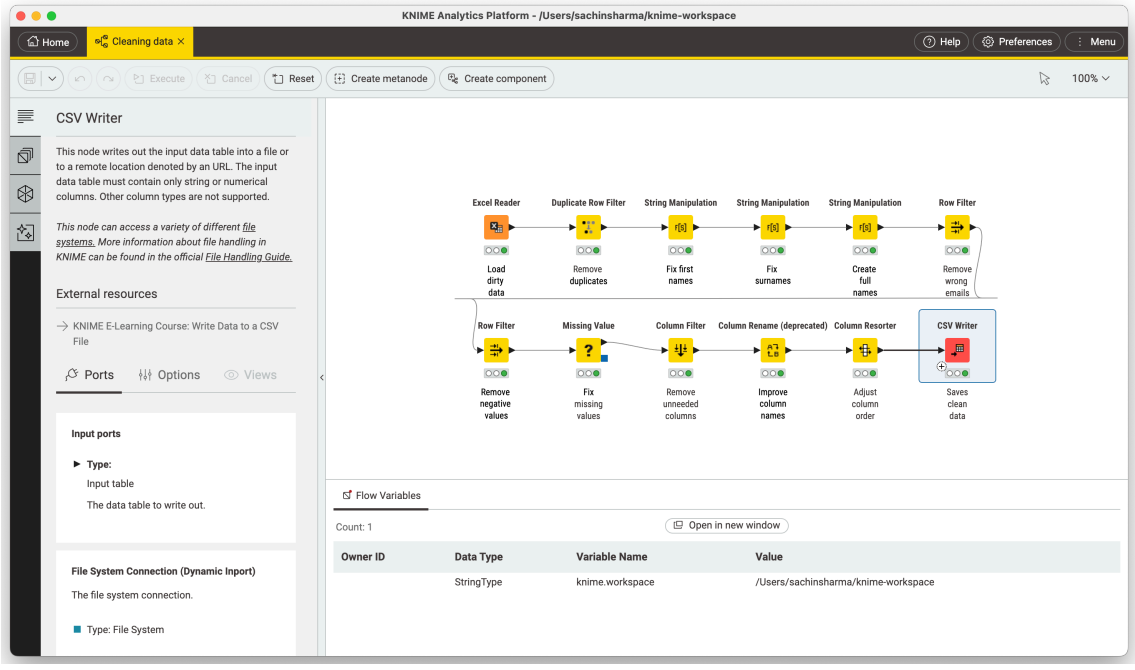

Fig 1

The workflow is loading excel file which contains data people data including name, email, age, credit and IP addresses. The whole data is considered as PII (Personal Identification Information).

The workflow is loading the data and pre-processing it with steps including removing duplicate data, removing unnecessary columns and filling empty columns. Finally we are storing the clean data using csv writer.

3. What are the data files used in this workflow? Describe the content of the datasets provided in these files.

Here we are using single xlsx file named DirtyData.xlsx. The file contains single sheet with 8 columns and 20 rows.

The below data contains columns with their details:

First Name:

Description: Represents the first name of the individual.

Data Patterns: No specific patterns, but names are in title case.

Observations: This column contains Users first name.

The data is inconsistent as it contains duplicate first name, it has white spaces which needs to be handled by trim() or strip() function of string and the words are not properly capitalized.

Analysing the first name data helps in understanding the diversity of names in the dataset and identifying any potential data quality issues, such as duplicates or inconsistent formatting.

Surname:

Description: Represents the surname or last name of the individual.

Data Patterns: No specific patterns, but surnames are in title case.

Observations: This column contains Users last name.

The data is inconsistent as it contains duplicate last names, it has white spaces which needs to be handled by trim() or strip() function of string and the words are not properly capitalized.

Analysing the surname data helps in understanding the diversity of surnames in the dataset and identifying any potential data quality issues, such as duplicates or inconsistent formatting.

__Email_Entered:

Description: Represents the email address entered by the individual.

Data Patterns: Follows the standard email format (username@domain).

Observations: The "__Email_Entered" column contains a variety of email addresses.

There is a case where the email address "bakshi.a@unesco.org" is repeated, possibly representing the same individual (duplicated entry).

There are few entries which are improper formats like ("jacekwlochem.com") appears to be missing the "@" symbol, and rmammatt16@prweb which is missing .domain entry.

Gender:

Description: Represents the gender of the individual..

Data Patterns: Categories include "Male," "Female," and "Rather not say.".

Observations: The "Gender" column contains three distinct categories: "Male," "Female," and "Rather not say."

The categories are well-defined and appear to be consistent in terms of spelling and capitalization.

The presence of "Rather not say" suggests that individuals had the option to withhold their gender information.

The data does not seem to have missing or invalid entries in the "Gender" column.

Age:

Description: Represents the age of the individual.

Data Patterns: Contains the age of individuals.

Observations: The "Age" column in this subset of data contains a mix of numeric values and null entries, where null represents missing data.

There are several instances of missing data in the "Age" column (e.g., rows 2, 4, 10, 17).

There are repeated values (e.g., 79, 47), suggesting individuals of the same age.

The ages vary, covering a range from teenagers (19) to individuals in their late 70s (79).

We can handle missing values using mean, median or mode imputations.

Logins:

Description: Represents the number of logins associated with the individuals.

Data Patterns: Contains number of users logins, some entries are missing.

Observations: The "Logins" column in this subset of data contains a mix of numeric values and null entries, where null represents missing data.

There are several instances of missing data in the "Logins" column (e.g., rows 3, 5, 7, 15).

The logins values vary, covering a range from 1 to 5.

There are repeated values, such as "3," suggesting individuals with the same number of logins.

__CREDITS:

Description: Represents the credit-related information.

Data Patterns: Contains the credit-related information for users.

Observations: The "_CREDIT" column contains a mix of numeric values, including positive and negative values.

There are some repeated values (e.g., 20.48, -100, 257.4), suggesting individuals with the same credit amounts.

Negative values (e.g., -100) may represent missing or placeholder values or could be valid data points depending on the context.

The credit values vary, covering a range from negative values to positive values.

Depending on the context, negative values might represent missing data or be valid entries. Consider investigating the meaning of negative values in the context of the dataset.

If negative values are not meaningful, you may choose to treat them as missing and replace them with an appropriate imputation method (e.g., mean, median).

IP_Address:

Description: Represents the IP address associated with the individual.

Data Patterns: Represents the IP address of individuals..

Observations: The "IP_Address" column contains IPv4 addresses, each consisting of four octets separated by periods.

Some IP addresses appear to be repeated (e.g., rows 8 and 9), suggesting the same individual or device.

IP addresses vary in terms of format and distribution.

If IP addresses are intended for analysis, consider using them for geolocation or network-related insights.

If the goal is to identify unique individuals or devices, investigate and address repeated IP addresses.

If IP addresses are not needed for your analysis, they can be used as unique identifiers or excluded from further processing. If geolocation information is relevant, external services can be used to obtain additional details based on IP addresses.

These columns provide information about the personal details (name, email, gender, age) and activities (logins, credit) of individuals, as well as their IP addresses. Analysing this data could involve tasks such as demographic analysis, login behaviour analysis, and identifying patterns in credit-related information

Analysing data characters and patterns helps in understanding the nature of the data, identifying anomalies, and preparing for further data processing or analysis.

4. Document what every node in the workflow does.

These are below nodes which are used in Cleaning Data workflow. Let us discuss in detail what each node is doing.

Excel Reader: This node is designed to read Excel files in various formats such as xlsx, xlsm, xlsb, and xls. It has the capability to read either a single file or multiple files simultaneously, though it processes only one sheet per file. In this specific instance, the node is employed to read the "DirtyData.xlsx" file from the local file system. The resulting data is then displayed as the output of the execution step. The significance of this node lies in its role of facilitating the loading of data, which is subsequently utilized in the workflow for further processing.

Duplicate Row Filter: This node is dedicated to detecting duplicate rows in a dataset. For each set of duplicates, the node selects a single row as the "chosen" one. Users have the flexibility to decide whether to eliminate all duplicate rows from the input

table, retaining only the unique and chosen rows, or to annotate the rows with additional information indicating their duplication status. In the specified configuration, the node has been set to remove duplicates based on email columns while excluding all other columns from this deduplication process. This approach is chosen because the email values serve as valuable input features when identifying unique values across the dataset.

String Manipulation: This node specializes in manipulating strings, offering functionalities such as search and replace, capitalization, and removal of leading and trailing white spaces. In our data pre-processing tasks, we employ this node for the following purposes:

- Addressing the first name column by eliminating white spaces around the strings and subsequently applying the capitalize function to enhance uniformity.
- Utilizing the same node in the subsequent step to refine the values in the surname column. This involves the process of stripping white spaces around the strings and capitalizing the values.
- Following these two stages of processing to rectify the first name and surname columns, we employ the node to concatenate these columns. This is achieved by using the join function to merge the values from the first name and surname columns, resulting in the creation of a new column representing the full name.

Row Filter: This node facilitates row filtering based on specified criteria. In our workflow, we leverage this node for two distinct purposes:

Email Format Validation:

- The node is employed to eliminate rows with email entries that do not adhere to the proper email format. Rows containing incorrect email formats are excluded from the dataset, ensuring data cleanliness for subsequent processing in the workflow.

Credit Column Filtering:

- Another application of the Row Filter node involves checking the credit column to remove rows with negative values. The configuration is set to validate against a minimum lower value of 0.0, ensuring that rows with credit values below this threshold are filtered out. This step is crucial for maintaining data integrity, particularly when negative credit values are deemed invalid for further processing.

Missing Values: This node is instrumental in addressing missing values within the cells of the input table. It offers a dual-tab configuration dialog for comprehensive handling:

Default Handling (First Tab):

- The "Default" tab presents default handling options for all columns of a specific type.
- These settings are applied universally to columns not explicitly specified in the second tab labelled "Individual."

Individual Column Settings (Second Tab):

- The "Individual" tab allows tailored settings for each available column, thereby superseding the default configurations.

In our specific use case:

- We are utilizing this node to manage missing values in the integer columns "Age" and "Logins."
- For the "Age" column, the strategy involves filling the missing values using the median.
- For the "Logins" column, a distinct approach is taken, filling the missing values with a default of 0.

This strategy ensures a nuanced handling of missing data in the specified columns, contributing to a more robust and accurate dataset for subsequent analysis or processing in the workflow.

Column Filter: This node facilitates the selective filtration of columns from the input table, allowing only the designated columns to pass through to the output table. The configuration dialog permits dynamic movement of columns between the "Include" and "Exclude" lists. In this particular instance:

- Excluded Columns:
  - Columns such as "First name," "Surname," and "Ip_Address" have been deliberately excluded.
  - The decision to remove "First name" and "Surname" is grounded in the creation of a consolidated column named "Full name," rendering the individual name columns redundant.

- The "Ip_Address" column is excluded as its values are deemed non-essential for the current analysis. While the "Ip_Address" may contribute to understanding user demographics, its inclusion is considered unnecessary for the specific objectives of our workflow. Consequently, we opt to exclude it from the output table to streamline the dataset for enhanced clarity and focus.

Column Rename: The Column Rename node provides functionality to rename column names or change their types. Within the dialog, users can either change the name of individual columns by editing the text field or modify the column type by selecting one from the combo box.

However, it's essential to note that the Column Rename node is deprecated, and its use in production workflows is not advisable. There is a potential for it to be completely removed in future releases. As an alternative, the recommended replacement is the Column Renamer (Dictionary) node.

In our specific application:

- We are rectifying column names, such as changing "__Email_Entered" to "Email" and "_CREDIT" to "Credit."
- The renaming process aids in enhancing user understanding by providing more intuitive and accurate column names. Additionally, the revised naming convention aligns with correct constraints, contributing to better clarity and consistency in the dataset.

By adopting the Column Renamer (Dictionary) node, we ensure compatibility with KNIME's evolving standards and reduce the risk associated with the deprecation of the Column Rename node in future releases.

Column Resorted: This node facilitates the adjustment of column order in the input table, according to user-defined preferences. Users can shift columns left or right in single steps, move them entirely to the beginning or end of the table, and even sort columns based on their names. The resulting re-sorted table is then made available at the output port.

In our specific use case:

- We are utilizing this node to correct the column orders, ensuring that columns are arranged in a way that aligns with our desired structure.
- The adjustments may involve shifting columns, moving them to specific positions, or sorting them based on names to achieve a more organized and meaningful arrangement.
- This process helps enhance the overall clarity and coherence of the dataset, making it more user-friendly and conducive to subsequent analyses or processing steps in the workflow.

CSV Writer: The CSV Writer node serves the purpose of storing the cleaned data, obtained after the execution of various nodes, to a CSV file in a local directory. This step becomes essential when there is a need to utilize the results with another program or if there is a requirement to persistently store the outcomes following the

completion of the workflow. By writing the data to a CSV file, it becomes easily accessible for external use, analysis, or archiving purposes.

5. Nodes, as computational units in the workflow, attempt to automate many of the manual data cleaning and data pre-processing tasks. Write a summary report describing your experience of running the Cleaning Data workflow. Identify nodes or missing information that you would add to the documentation.

Summary Report: Cleaning Data Workflow in KNIME

Objective: The Cleaning Data workflow in KNIME was designed to address data quality issues, missing values, and anomalies in the provided dataset. The goal was to prepare the data for further analysis or modeling.

Experience:

The workflow effectively handled missing values in numeric and categorical columns using appropriate imputation methods.

Duplicates were identified and managed, ensuring data integrity.

String manipulation nodes may have been employed to standardize entries, especially in the "Gender" column.

Consideration was given to potential insights derived from IP addresses, although further documentation may be needed on the specific analysis or use case for this information.

Documentation Recommendations:

Imputation Rationale:

Document the rationale behind the chosen imputation methods, especially if mean, median, or forward fill were used. Explain why these methods were suitable for the specific columns.

Handling Negative Values:

Elaborate on the decision to handle negative values in the "_CREDIT" column, providing justification for the chosen approach.

IP Address Insights:

If IP addresses were utilized for specific analyses, document the goals and outcomes of that analysis. Clarify any external services or libraries used for geolocation.

Workflow Structure:

Provide an overview of the workflow structure, outlining the logical flow of nodes and the purpose of each step.

Data Preview:

Include a data preview or sample output at key stages of the workflow to facilitate understanding and validation.

Next Steps:

Suggest next steps in the data analysis or modeling process after the data cleaning workflow. Specify any additional analyses or nodes that could be beneficial.

Conclusion: The Cleaning Data workflow in KNIME successfully addressed data quality issues, missing values, and duplicates. Documentation enhancements should focus on explaining the rationale behind decisions, providing insights from IP addresses, and ensuring clarity in the workflow structure for future reference.

Citations:

Detail about KNIME Nodes has been taken from below links:
https://hub.knime.com/adm/spaces/Public/Workflows/Data%20Analytics%20Made

%20Easy/Chapter%202/Cleaning%20data~DLJtVEX0f9gQTCS7/current-state

Detail about Column Renamer (Dictionary) has been taken from below link:

https://hub.knime.com/knime/extensions/org.knime.features.base/latest/org.knime.

base.node.preproc.columnheaderinsert.ColumnHeaderInsertNodeFactory