AIM: Apply data cleaning techniques (e.g. Data Imputation).

THEORY:

Data cleaning is identifying and correcting errors, inconsistencies, and missing values in a dataset. This is crucial for ensuring the accuracy and reliability of analysis and modelling.

Data imputation specifically addresses missing values in a dataset. Missing values can occur for various reasons, such as data collection errors, participant non-response, or measurement limitations. Imputation aims to estimate and replace these missing values with reasonable values.

There are several data imputation techniques, each with its advantages and disadvantages. The choice of technique depends on the data type, the nature of missingness (random vs. non-random), and the intended use of the data.

Some common imputation techniques:

1. **Next or Previous Value:** Replaces missing values with the value from the previous or next non-missing observation.
2. **K Nearest Neighbors (KNN):** Uses the values of similar observations (k nearest neighbours) to estimate the missing value.
3. **Maximum or Minimum Value:** Replaces missing values with the maximum or minimum value in the dataset (be cautious as it can introduce bias).
4. **Missing Value Prediction:** Trains a model (e.g., linear regression) to predict missing values based on other variables in the dataset.
5. **Most Frequent Value:** Replaces missing values with the most frequently occurring value in the variable.
6. **Average or Linear Interpolation:** Estimates missing values by averaging or interpolating between surrounding non-missing values.
7. **(Rounded) Mean or Moving Average or Median Value:** Replaces missing values with the overall mean, moving average, or median of the variable.
8. **Fixed Value:** Replaces missing values with a predetermined constant value.

Code Analysis

The code demonstrates various data-cleaning techniques using Python libraries like Pandas and Scikit-learn. Here's a breakdown of the key functionalities:

1. **Data Preprocessing:**
    a. Reads the CSV data into a Pandas DataFrame.
    b. Drops irrelevant columns from the analysis.
    c. Selects columns containing numerical data (int and float) for imputation.
2. **Imputation Techniques:**
    a. **Next or Previous Value:** Implements a custom function to fill missing values with the previous or next non-missing value.
    b. **KNN Imputation:** Uses KNNImputer from sci-kit-learn to impute missing values based on k nearest neighbours.
    c. **Maximum or Minimum Value:** Iterates through each column and replaces missing values with the maximum or minimum value.
    d. **Missing Value Prediction:** Performs the following steps:
        i. Identifies features with missing values.
        ii. Temporarily imputes missing values with mean values.
        iii. For each feature with missing values, train a linear regression model using other features as predictors.
        iv. Uses the trained model to predict missing values in the original data frame.
    e. **Most Frequent Value:** Iterates through each column and replaces missing values with the most frequent value.
    f. **Average or Linear Interpolation:** Uses Pandas interpolate method to fill missing values using linear interpolation.
    g. **(Rounded) Mean or Moving Average or Median Value:** Iterates through each column and replaces missing values with the mean, moving average, or median of the variable (not implemented).
    h. **Fixed Value:** Prompts the user to enter a fixed value and replaces missing values with that value.
3. **User Interaction:** Presents a menu with various imputation techniques and allows the user to choose the desired method.
4. **Evaluation:** After applying the chosen technique, display the remaining number of missing values in the dataset.

CONCLUSION:

Data cleaning and imputation are crucial steps in data analysis, ensuring accuracy and reliability. Addressing inconsistencies, errors, and missing values is essential for comprehensive analysis. Imputation offers a solution by estimating and replacing missing values with reasonable alternatives. Careful selection of imputation techniques depends on data characteristics and analysis goals. Understanding missingness patterns, incorporating domain-specific knowledge, and evaluating and validating techniques are also essential. By understanding these steps, well-cleaned data forms the foundation for robust analysis and informed decision-making.

```
import pandas as pd
from sklearn.impute import KNNImputer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
df = pd.read_csv("food_coded.csv")
df.head()
```

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	com
0	2.4	2	1	430	NaN	315.0	1	
1	3.654	1	1	610	3.0	420.0	2	
2	3.3	1	1	720	4.0	420.0	2	fro
3	3.2	1	1	430	3.0	420.0	2	a
4	3.5	1	1	720	2.0	420.0	2	

5 rows × 61 columns

```
df.columns
```

```
Index(['GPA', 'Gender', 'breakfast', 'calories_chicken', 'calories_day',
       'calories_scone', 'coffee', 'comfort_food', 'comfort_food_reasons',
       'comfort_food_reasons_coded', 'cook', 'comfort_food_reasons_coded.1',
       'cuisine', 'diet_current', 'diet_current_coded', 'drink',
       'eating_changes', 'eating_changes_coded', 'eating_changes_coded1',
       'eating_out', 'employment', 'ethnic_food', 'exercise',
       'father_education', 'father_profession', 'fav_cuisine',
       'fav_cuisine_coded', 'fav_food', 'food_childhood', 'fries', 'fruit_day',
       'grade_level', 'greek_food', 'healthy_feeling', 'healthy_meal',
       'ideal_diet', 'ideal_diet_coded', 'income', 'indian_food',
       'italian_food', 'life_rewarding', 'marital_status',
       'meals_dinner_friend', 'mother_education', 'mother_profession',
       'nutritional_check', 'on_off_campus', 'parents_cook', 'pay_meal_out',
       'persian_food', 'self_perception_weight', 'soup', 'sports', 'thai_food',
       'tortilla_calories', 'turkey_calories', 'type_sports', 'veggies_day',
       'vitamins', 'waffle_calories', 'weight'],
      dtype='object')
```

```
df.shape
```

```
(125, 61)
```

```
df.drop(columns=["employment","comfort_food_reasons","comfort_food_reasons_coded","comfort_food_reasons_coded.1","diet_current_coded","eati
                 "fav_cuisine_coded","ideal_diet_coded","income","life_rewarding","marital_status","meals_dinner_friend","mother_education"
       'food_childhood', 'healthy_meal', 'ideal_diet', 'type_sports',
       'weight'],axis=1,inplace=True)
df.columns
```

```
Index(['Gender', 'breakfast', 'calories_chicken', 'calories_day',
       'calories_scone', 'coffee', 'cook', 'cuisine', 'drink', 'eating_out',
       'ethnic_food', 'exercise', 'fav_food', 'fries', 'fruit_day',
       'grade_level', 'greek_food', 'healthy_feeling', 'indian_food',
       'italian_food', 'nutritional_check', 'persian_food', 'soup', 'sports',
       'thai_food', 'tortilla_calories', 'turkey_calories', 'veggies_day',
       'vitamins', 'waffle_calories'],
      dtype='object')
```

```
df.shape
```

```
(125, 30)
```

```
print(df.isnull().sum())
```

```
Gender                0
breakfast             0
calories_chicken      0
calories_day         19
calories_scone        1
coffee                0
cook                  3
cuisine              17
drink                 2
eating_out            0
ethnic_food           0
exercise             13
fav_food              2
fries                 0
fruit_day             0
grade_level           0
greek_food            0
healthy_feeling       0
indian_food           0
italian_food          0
nutritional_check     0
persian_food          1
soup                  1
sports                2
thai_food             0
tortilla_calories     1
turkey_calories       0
veggies_day           0
vitamins              0
waffle_calories       0
dtype: int64
```

```
# Assuming df is your DataFrame
# Use select_dtypes() to select columns of types 'int', 'float'
selected_columns = df.select_dtypes(include=['int', 'float'])

# Display the list of selected columns
print("Selected columns:")
print(selected_columns.columns)
```

```
Selected columns:
Index(['Gender', 'breakfast', 'calories_chicken', 'calories_day',
       'calories_scone', 'coffee', 'cook', 'cuisine', 'drink', 'eating_out',
       'ethnic_food', 'exercise', 'fav_food', 'fries', 'fruit_day',
       'grade_level', 'greek_food', 'healthy_feeling', 'indian_food',
       'italian_food', 'nutritional_check', 'persian_food', 'soup', 'sports',
       'thai_food', 'tortilla_calories', 'turkey_calories', 'veggies_day',
       'vitamins', 'waffle_calories'],
      dtype='object')
```

```
def fill_missing_values(df, columns):
  for col in columns:
    for i in range(len(df)):
      if pd.isna(df.loc[i, col]):
        # Check for previous non-missing value
        prev_value = df.loc[i - 1, col] if i > 0 and not pd.isna(df.loc[i - 1, col]) else None
        # Check for next non-missing value (if previous not found)
        next_value = df.loc[i + 1, col] if i < len(df) - 1 and not pd.isna(df.loc[i + 1, col]) else None
        # Use the first available non-missing value (previous or next)
        df.loc[i, col] = prev_value or next_value
  return df

def fill_missing_values_knn(df, columns):
    imputer = KNNImputer(n_neighbors=5)
    df_filled = pd.DataFrame(imputer.fit_transform(df), columns=df.columns, index=df.index)
    return df_filled


def fill_missing_values_regression(df):

  # Step 1: Identify Features with Missing Values
  features_with_missing_values = df.columns[df.isnull().any()].tolist()

  # Step 2: Temporary Imputation (Mean Imputation)
  df_temp_imputed = df.fillna(df.mean())
  # Step 3-6: Train a Model for Each Feature with Missing Values
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum '
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
```

```
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
 Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 1
 Gender               0
 breakfast            0
 calories_chicken     0
 calories_day         0
 calories_scone       0
 coffee               0
 cook                 0
 cuisine              0
 drink                0
 eating_out           0
 ethnic_food          0
 exercise             0
 fav_food             0
 fries                0
 fruit_day            0
 grade_level          0
 greek_food           0
 healthy_feeling      0
 indian_food          0
 italian_food         0
 nutritional_check    0
 persian_food         0
 soup                 0
 sports               0
 thai_food            0
```

```
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum Va
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 2
 Gender              0
 breakfast           0
 calories_chicken    0
 calories_day        0
```

```
calories_scone       0
coffee               0
cook                 0
cuisine              0
drink                0
eating_out           0
ethnic_food          0
exercise             0
fav_food             0
fries                0
fruit_day            0
grade_level          0
greek_food           0
healthy_feeling      0
indian_food          0
italian_food         0
nutritional_check    0
persian_food         0
soup                 0
sports               0
thai_food            0
tortilla_calories    0
turkey_calories      0
veggies_day          0
vitamins             0
waffle_calories      0
dtype: int64
```

```
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum Va
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 3
 Gender              0
 breakfast           0
 calories_chicken    0
 calories_day        0
```

```
calories_scone       0
coffee               0
cook                 0
cuisine              0
drink                0
eating_out           0
ethnic_food          0
exercise             0
fav_food             0
fries                0
fruit_day            0
grade_level          0
greek_food           0
healthy_feeling      0
indian_food          0
italian_food         0
nutritional_check    0
persian_food         0
soup                 0
sports               0
thai_food            0
tortilla_calories    0
turkey_calories      0
veggies_day          0
vitamins             0
waffle_calories      0
dtype: int64
```

```
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum Va
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 4
 Gender              0
 breakfast           0
 calories_chicken    0
 calories_day        0
```

```
calories_scone       0
coffee               0
cook                 0
cuisine              0
drink                0
eating_out           0
ethnic_food          0
exercise             0
fav_food             0
fries                0
fruit_day            0
grade_level          0
greek_food           0
healthy_feeling      0
indian_food          0
italian_food         0
nutritional_check    0
persian_food         0
soup                 0
sports               0
thai_food            0
tortilla_calories    0
turkey_calories      0
veggies_day          0
vitamins             0
waffle_calories      0
dtype: int64
```

```
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum Va
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 5
 Gender              0
 breakfast           0
 calories_chicken    0
 calories_day        0
```

```
calories_scone       0
coffee               0
cook                 0
cuisine              0
drink                0
eating_out           0
ethnic_food          0
exercise             0
fav_food             0
fries                0
fruit_day            0
grade_level          0
greek_food           0
healthy_feeling      0
indian_food          0
italian_food         0
nutritional_check    0
persian_food         0
soup                 0
sports               0
thai_food            0
tortilla_calories    0
turkey_calories      0
veggies_day          0
vitamins             0
waffle_calories      0
dtype: int64
```

```
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum Va
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 6
 Gender              0
 breakfast           0
 calories_chicken    0
 calories_day        1
```

```
calories_scone       0
coffee               0
cook                 0
cuisine              1
drink                0
eating_out           0
ethnic_food          0
exercise             0
fav_food             0
fries                0
fruit_day            0
grade_level          0
greek_food           0
healthy_feeling      0
indian_food          0
italian_food         0
nutritional_check    0
persian_food         0
soup                 0
sports               0
thai_food            0
tortilla_calories    0
turkey_calories      0
veggies_day          0
vitamins             0
waffle_calories      0
dtype: int64
```

```
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum Va
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 7
 Gender              0
 breakfast           0
 calories_chicken    0
 calories_day        0
```

```
calories_scone       0
coffee               0
cook                 0
cuisine              0
drink                0
eating_out           0
ethnic_food          0
exercise             0
fav_food             0
fries                0
fruit_day            0
grade_level          0
greek_food           0
healthy_feeling      0
indian_food          0
italian_food         0
nutritional_check    0
persian_food         0
soup                 0
sports               0
thai_food            0
tortilla_calories    0
turkey_calories      0
veggies_day          0
vitamins             0
waffle_calories      0
dtype: int64
```

```
  for feature in features_with_missing_values:
    # Split data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(
        df_temp_imputed.drop(columns=[feature]),
        df_temp_imputed[feature],
        test_size=0.2,
        random_state=42
    )

    # Train a regression model (you can use any other model suitable for your data)
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Impute missing values in the original DataFrame
    missing_indices = df[feature].isnull()
    df.loc[missing_indices, feature] = model.predict(df_temp_imputed.drop(columns=[feature])[missing_indices])

  return df


menu = '''Enter which technique you want to use as data cleaning\n1. Next or Previous Value\n2. K Nearest Neighbors\n3. Maximum or Minimum Va
5. Most Frequent Value\n6. Average or Linear Interpolation\n7. (Rounded) Mean or Moving Average or Median Value\n8. Fixed Value\n'''
print(menu)
choice = int(input("Enter the choice: "))

if choice == 1:
    filled_df = fill_missing_values(df.copy(), selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 2:
    filled_df = fill_missing_values_knn(df[selected_columns.columns], selected_columns.columns)
    print(filled_df.isnull().sum())
elif choice == 3:
    for col in selected_columns.columns:
        max_val = df[col].max()
        min_val = df[col].min()
        df[col].fillna(max_val, inplace=True)  # or df[col].fillna(min_val, inplace=True)
    print(df.isnull().sum())
elif choice == 4:
    filled_df = fill_missing_values_regression(df)
    print(filled_df.isnull().sum())
elif choice == 5:
    for col in selected_columns.columns:
        most_frequent = df[col].mode()[0]
        df[col].fillna(most_frequent, inplace=True)
    print(df.isnull().sum())
elif choice == 6:
    df.interpolate(method='linear', inplace=True)
    print(df.isnull().sum())
elif choice == 7:
    for column in selected_columns.columns:
        mean_value = df[column].mean()
        df[column].fillna(mean_value, inplace=True)
    print(df.isnull().sum())
elif choice == 8:
    fixed_value = float(input("Enter the fixed value: "))  # Assuming the fixed value is a float
    for col in selected_columns.columns:
        df[col].fillna(fixed_value, inplace=True)
    print(df.isnull().sum())
```

```
Enter which technique you want to use as data cleaning
 1. Next or Previous Value
 2. K Nearest Neighbors
 3. Maximum or Minimum Value
 4. Missing Value Prediction
 5. Most Frequent Value
 6. Average or Linear Interpolation
 7. (Rounded) Mean or Moving Average or Median Value
 8. Fixed Value

 Enter the choice: 8
 Enter the fixed value: 10
 Gender              0
 breakfast           0
 calories_chicken    0
```

```
calories_day         0
calories_scone       0
coffee               0
cook                 0
cuisine              0
drink                0
eating_out           0
ethnic_food          0
exercise             0
fav_food             0
fries                0
fruit_day            0
grade_level          0
greek_food           0
healthy_feeling      0
indian_food          0
italian_food         0
nutritional_check    0
persian_food         0
soup                 0
sports               0
thai_food            0
tortilla_calories    0
turkey_calories      0
veggies_day          0
vitamins             0
waffle_calories      0
dtype: int64
```