Data Wrangling/Preprocessing: (Data Munging)

Notes by Mannan Ul Haq (BDS-3C)

Data munging, often referred to as data wrangling or data preprocessing, is the process of transforming, and organizing raw data into a format that is suitable for analysis or further processing.

Data Munging Tasks:

- Renaming Variables
- Data Type Conversion
- Encoding, Decoding, Recoding data.
- Merging Datasets
- Transforming Data

Renaming Variable

Renaming variables in data munging involves changing the names of columns or variables in a dataset. This is done for several reasons, including making the variable names more meaningful, clear, or consistent with your analysis goals. Here are some common scenarios where you might need to rename variables:

1. **Clarity**: Variable names may be abbreviated in the original dataset, making it difficult to understand their meaning. Renaming them to more descriptive names can improve clarity.
2. **Consistency**: Ensuring that variable names follow a consistent naming convention across your dataset or with other datasets can make it easier to work with and merge data.
3. **Standardization**: In some cases, you might need to standardize variable names to match industry or organizational standards.

Here's an example of how you can rename variables using Python's pandas library:

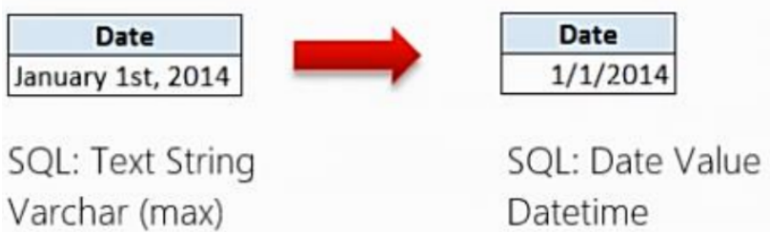
```
import pandas as pd

# Create a sample DataFrame
data = {'Old_Variable_Name1': [1, 2, 3],
        'Old_Variable_Name2': [4, 5, 6]}

df = pd.DataFrame(data)

# To get Columns Name
print(df.columns)

# Rename variables
df.rename(columns={'Old_Variable_Name1': 'New_Variable_Name1',
                   'Old_Variable_Name2': 'New_Variable_Name2'}, inplace = True)

# Now, the DataFrame has variables with new names
```

Data Type Conversion

Data type conversion, also known as data type casting, is the process of changing the type of data in a column to ensure it's compatible with your analysis or processing needs. This is a crucial data munging task because datasets often contain a mix of different data types, and converting them to the right type is essential for accurate analysis. Here are some common scenarios where data type conversion is necessary:

1. **String to Numeric**: Converting string values (text) to numeric types (integers or floating-point numbers) is common when working with numerical data. For example, you might need to convert "123" to the integer 123.
2. **Numeric to String**: Sometimes, you may need to convert numeric values to strings, especially when you want to include them in text output or concatenate them with other strings.

3. **Date and Time Formats**: Converting date and time data to a consistent format (e.g., from text to datetime objects) allows for date-based analysis and calculations.
4. **Categorical to Numerical**: Converting categorical data (e.g., "low," "medium," "high") to numerical values (e.g., 0, 1, 2) for use in machine learning algorithms.
5. **Boolean Conversion**: Converting binary values (e.g., "yes" or "no") to boolean values (True or False).



Here's an example in Python using pandas to convert a column's data type from string to integer:

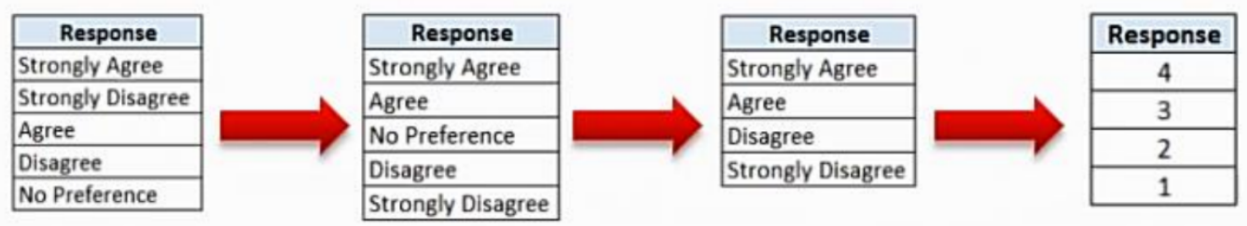
```
import pandas as pd

# Create a sample DataFrame with a column containing string values
data = {'Column1': ['123', '456', '789']}

df = pd.DataFrame(data)

column = df['Column1']

# To check Data Type of first index value
print(type(column[0]))

# Convert the 'Column1' data type from string to integer
df['Column1'] = df['Column1'].astype(int)

# Now, 'Column1' contains integer values
```

In this example, the `astype` method is used to convert the 'Column1' from a string data type to an integer data type. This makes the data suitable for numerical analysis.

Encoding, Decoding, and Recoding Data

Encoding, decoding, and recoding data are data munging tasks that deal with converting and transforming categorical variables, text, or other non-numeric data into a format that can be used for analysis or modeling.

Encoding Data: Encoding involves converting categorical or textual data into numerical representations. This is essential when working with machine learning algorithms that require numerical inputs. For example, assign a unique integer to each category. It is suitable for ordinal categorical data (categories with a specific order).



Decoding Data: Decoding is the reverse process of encoding. It converts numerical representations back into their original categorical or textual form. This is often done for interpretability or reporting purposes.

Recoding Data: Recoding refers to changing the values of a variable to make them more meaningful, manageable, or suitable for analysis. This can involve combining categories, creating new categories, or mapping specific values to others.

Merging Datasets

Merging datasets involves combining two or more separate datasets into a single, unified dataset. This is often done to enrich data for analysis or modeling. There are several common types of dataset merging:

1. **Concatenation:** This is used to combine datasets that have the same structure. Rows from one dataset are appended to another, either vertically (row-wise) or horizontally (column-wise).

2. **Joining (Merging):** Joining is used when you have datasets with a common key (e.g., an identifier or a shared column), and you want to combine them based on that key. Common join types include inner join, left join, right join, and outer join.

Here's an example in Python using pandas to demonstrate concatenation and joining:

Concatenation:

```
import pandas as pd

# Create two sample DataFrames
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                    'B': ['B0', 'B1', 'B2']})

df2 = pd.DataFrame({'A': ['A3', 'A4', 'A5'],
                    'B': ['B3', 'B4', 'B5']})

# Concatenate vertically (along rows)
concatenated_df = pd.concat([df1, df2], axis=0)

# Concatenate horizontally (along columns)
concatenated_df_horizontal = pd.concat([df1, df2], axis=1)
```

Joining:

```
# Create two sample DataFrames with a common key
left_df = pd.DataFrame({'Key': ['K0', 'K1', 'K2'],
                        'Value': ['V0', 'V1', 'V2']})

right_df = pd.DataFrame({'Key': ['K1', 'K2', 'K3'],
                         'Value': ['V3', 'V4', 'V5']})

# Perform an inner join based on the 'Key' column
inner_join_df = pd.merge(left_df, right_df, on='Key')

# The 'inner_join_df' now contains rows with matching 'Key' values from both DataFrames
```

Data Transformation

Data transformation is a crucial step in data munging and data preprocessing, and it involves various techniques to make your data more suitable for analysis and modeling. It is basically used for feature scaling.

Feature scaling aims to ensure features are on a similar scale, making each feature equally important and facilitating processing by most ML algorithms.

Here are the three data transformation techniques:

1. **Bucketing/Binning**:
    - **Definition**: Bucketing, also known as binning, involves dividing a continuous variable into discrete intervals or bins. Each bin represents a range of values, and this transformation is often used to simplify data or convert continuous variables into categorical variables.
    - **Use Cases**: Binning can be helpful when you want to reduce the impact of outliers, create categorical variables from continuous data, or make data more interpretable.
    - **Example**: You can bin age values into groups like "child," "adult," and "senior" based on specified age ranges.
2. **Normalization**:
    - **Definition**: Normalization is a data transformation that scales numeric variables to a common range, often [0, 1] or [-1, 1]. It is useful for ensuring that variables are on a consistent scale, which is important for machine learning algorithms that are sensitive to feature magnitudes.
    - **Use Cases**: Normalization is particularly important when working with models like k-nearest neighbors, support vector machines, and neural networks.
    - **Example**:

    Min-Max Normalization: Min-Max scaling is a common normalization technique that re-scales features to a distribution value between 0 and 1. The minimum value transforms into 0, and the maximum value transforms into 1. This implies the data is more concentrated around the mean with Max-Min Normalization. If there are outliers in your features, normalizing data scales most of it to a small interval, providing uniform scales but not handling outliers well. Standardization is more robust to outliers, making it preferable in many cases.

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Standardization (Z-Score Normalization): Rescales features to ensure mean and standard deviation are 0 and 1, respectively. The range is from -3 to +3.

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

This is a dataset that contains an independent variable (Purchased) and 3 dependent variables (Country, Age, and Salary). We can easily notice that the variables are not on the same scale because the range of ***Age* is from 27 to 50**, while the range of *Salary* going from 48 K to 83 K.

	Country	Age	Salary	Purchased
1	France	44	72000	No
2	Spain	27	48000	Yes
3	Germany	30	54000	No
4	Spain	38	61000	No
5	Germany	40		Yes
6	France	35	58000	Yes
7	Spain		52000	No
8	France	48	79000	Yes
9	Germany	50	83000	No
10	France	37	67000	Yes

Standardisation

	Age	Salary
0	0.758874	7.494733e-01
1	-1.711504	-1.438178e+00
2	-1.275555	-8.912655e-01
3	-0.113024	-2.532004e-01
4	0.177609	6.632192e-16
5	-0.548973	-5.266569e-01
6	0.000000	-1.073570e+00
7	1.340140	1.387538e+00
8	1.630773	1.752147e+00
9	-0.258340	2.937125e-01

Max-Min Normalization

	Age	Salary
0	0.739130	0.685714
1	0.000000	0.000000
2	0.130435	0.171429
3	0.478261	0.371429
4	0.565217	0.450794
5	0.347826	0.285714
6	0.512077	0.114286
7	0.913043	0.885714
8	1.000000	1.000000
9	0.434783	0.542857

3. **Other Mathematical Transformations**:
    - **Definition**: This category encompasses various mathematical operations that can be applied to data, such as logarithmic transformations, exponentiation,

square roots, or other mathematical functions.

- **Use Cases**: These transformations are used when you need to change the distribution of data, reduce the impact of outliers, or make data conform to certain statistical assumptions.