codeDataTransformation

October 8, 2024

3.1 Introduction to Data transformation

Data transformation is a crucial step in the data preparation process that involves converting raw data into a format suitable for analysis. It plays a vital role in ensuring that data is structured, consistent, and ready for various analytical tasks, including machine learning, reporting, and visualization. By applying transformation techniques, you can enhance the quality of data, making it more meaningful and easier to work with.

Definition Data transformation refers to the process of changing the format, structure, or values of data to make it more suitable for specific applications. This can involve a wide range of operations, such as scaling, encoding, aggregating, or normalizing data. These transformations help in creating a uniform and comparable dataset that aligns with the requirements of the analytical models or business objectives.

Objective The primary objective of data transformation is to improve the quality and usability of data by applying various techniques that address inconsistencies, normalize scales, encode categorical variables, and enhance the overall data structure. By transforming data, we can uncover hidden patterns, reduce the complexity of data analysis, and ensure that the data is in a suitable format for effective decision-making.

Importance Data transformation is essential for several reasons: 1. Enhancing Data Quality: It ensures that data is clean, consistent, and free from errors or inconsistencies, which is critical for reliable analysis. 2. Improving Model Performance: Properly transformed data can lead to better model accuracy and performance, especially in machine learning. 3. Facilitating Data Integration: It allows for the seamless integration of data from different sources, enabling comprehensive analysis across various datasets. 4. Supporting Business Goals: By aligning data with specific business objectives, transformation techniques help in extracting actionable insights that drive informed decisions.

3.2 Techniques 1. Normalization 2. Standardization 3. Encoding Categorical Variables 1. One-Hot Encoding 2. Label Encoding 4. Binning 5. Log Transformation 6. Polynomial Transformation 7. Box-Cox Transformation 8. Feature Scaling 9. Text Data Transformation 10. Tokenization 11. Stemming/Lemmatization 12. Handling Dates and Times 13. Aggregations and Rolling Calculations 14. Discretization

3.2.1 Normalization

Normalization is the process of scaling individual data points to a common range, usually [0, 1]. It is useful for ensuring that features contribute equally to the analysis and prevents any single feature from dominating due to its scale.

Techniques Min-Max Normalization: Rescales the data to a fixed range (typically [0, 1]).

[29]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Read the data from the specified location
df = pd.read_csv('D:/Projects/Data-cleaning-series/Chapter03 Data␣
 ↪Transformation/Products.csv')

# Display the initial DataFrame
print("Initial DataFrame:")
print(df.to_string(index=False))

# Initialize the MinMaxScaler
scaler = MinMaxScaler()

# Apply Min-Max Normalization to the 'Price' column
df['Price_Normalized'] = scaler.fit_transform(df[['Price']])

# Display the DataFrame after normalization
print("\nDataFrame After Min-Max Normalization:")
print(df.to_string(index=False))
```

```
Initial DataFrame:
 Product ID Product Name  Price    Category  Stock              Description
          1     Widget A  19.99 Electronics  100.0    A high-quality widget
          2     Widget B  29.99 Electronics    NaN                     NaN
          3          NaN  15.00  Home Goods   50.0      Durable and stylish
          4     Widget D    NaN  Home Goods  200.0       A versatile widget
          5     Widget E   9.99         NaN   10.0    Compact and efficient
          6     Widget F  25.00 Electronics    0.0 Latest technology widget
          7     Widget G    NaN     Kitchen  150.0     Multi-purpose widget
          8     Widget H  39.99     Kitchen   75.0          Premium quality
          9     Widget I    NaN Electronics    NaN        Advanced features
         10     Widget J  49.99 Electronics   60.0            Best in class

DataFrame After Min-Max Normalization:
 Product ID Product Name  Price    Category  Stock              Description
Price_Normalized
          1     Widget A  19.99 Electronics  100.0    A high-quality widget
0.25000
          2     Widget B  29.99 Electronics    NaN                     NaN
0.50000
          3          NaN  15.00  Home Goods   50.0      Durable and stylish
0.12525
          4     Widget D    NaN  Home Goods  200.0       A versatile widget
NaN
          5     Widget E   9.99         NaN   10.0    Compact and efficient
0.00000
          6     Widget F  25.00 Electronics    0.0 Latest technology widget
```

```
0.37525
          7     Widget G    NaN     Kitchen  150.0     Multi-purpose widget
NaN
          8     Widget H  39.99     Kitchen   75.0          Premium quality
0.75000
          9     Widget I    NaN Electronics    NaN        Advanced features
NaN
         10     Widget J  49.99 Electronics   60.0            Best in class
1.00000
```

Explanation:

Read the Data: Load the dataset from the specified location using pd.read_csv().

Initial Display: Display the DataFrame to see the data before applying normalization.

Initialize Scaler: Initialize the MinMaxScaler from sklearn.preprocessing.

Apply Normalization: Apply Min-Max normalization to the 'Price' column and add the normalized values as a new column.

Final Display: Display the DataFrame after applying normalization.

3.2.2 Standardization

Standardization transforms data to have a mean of 0 and a standard deviation of 1. This technique is useful when the data follows a Gaussian distribution and is required for many machine learning algorithms.

Techniques Z-Score Standardization: Subtracts the mean and divides by the standard deviation of each feature.

[30]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Read the data from the specified location
df = pd.read_csv('D:/Projects/Data-cleaning-series/Chapter03 Data␣
 ↪Transformation/Products.csv')

# Display the initial DataFrame
print("Initial DataFrame:")
print(df.to_string(index=False))

# Initialize the StandardScaler
scaler = StandardScaler()

# Apply Z-Score Standardization to the 'Price' column
df['Price_Standardized'] = scaler.fit_transform(df[['Price']])

# Display the DataFrame after standardization
print("\nDataFrame After Z-Score Standardization:")
print(df.to_string(index=False))
```

```
Initial DataFrame:
 Product ID Product Name  Price    Category  Stock              Description
          1     Widget A  19.99 Electronics  100.0    A high-quality widget
          2     Widget B  29.99 Electronics    NaN                      NaN
          3          NaN  15.00  Home Goods   50.0      Durable and stylish
          4     Widget D    NaN  Home Goods  200.0       A versatile widget
          5     Widget E   9.99         NaN   10.0    Compact and efficient
          6     Widget F  25.00 Electronics    0.0 Latest technology widget
          7     Widget G    NaN     Kitchen  150.0     Multi-purpose widget
          8     Widget H  39.99     Kitchen   75.0          Premium quality
          9     Widget I    NaN Electronics    NaN        Advanced features
         10     Widget J  49.99 Electronics   60.0            Best in class

DataFrame After Z-Score Standardization:
 Product ID Product Name  Price    Category  Stock              Description
Price_Standardized
          1     Widget A  19.99 Electronics  100.0    A high-quality widget
-0.547460
          2     Widget B  29.99 Electronics    NaN                      NaN
0.218678
          3          NaN  15.00  Home Goods   50.0      Durable and stylish
-0.929763
          4     Widget D    NaN  Home Goods  200.0       A versatile widget
NaN
          5     Widget E   9.99         NaN   10.0    Compact and efficient
-1.313598
          6     Widget F  25.00 Electronics    0.0 Latest technology widget
-0.163625
          7     Widget G    NaN     Kitchen  150.0     Multi-purpose widget
NaN
          8     Widget H  39.99     Kitchen   75.0          Premium quality
0.984815
          9     Widget I    NaN Electronics    NaN        Advanced features
NaN
         10     Widget J  49.99 Electronics   60.0            Best in class
1.750953
```

Explanation:

Read the Data: Load the dataset from the specified location using pd.read_csv().

Initial Display: Display the DataFrame to see the data before applying standardization.

Initialize Scaler: Initialize the StandardScaler from sklearn.preprocessing.

Apply Standardization: Apply Z-Score standardization to the 'Price' column and add the standardized values as a new column.

Final Display: Display the DataFrame after applying standardization.

3.2.3 Encoding Categorical Variables

Encoding categorical variables involves converting categorical data into numerical format so that it can be used in machine learning algorithms. Common techniques include one-hot encoding and label encoding.

Techniques 1. One-Hot Encoding: Converts each category into a separate binary column.

[31]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Read the data from the specified location
df = pd.read_csv('D:/Projects/Data-cleaning-series/Chapter03 Data␣
 ↪Transformation/Products.csv')

# Display the initial DataFrame
print("Initial DataFrame:")
print(df.to_string(index=False))

# Initialize the OneHotEncoder
encoder = OneHotEncoder(sparse_output=False)

# Apply One-Hot Encoding to the 'Category' column
category_encoded = encoder.fit_transform(df[['Category']])
category_encoded_df = pd.DataFrame(category_encoded, columns=encoder.
 ↪get_feature_names_out(['Category']))

# Concatenate the one-hot encoded columns to the original DataFrame
df = pd.concat([df, category_encoded_df], axis=1)

# Drop the original 'Category' column
df.drop('Category', axis=1, inplace=True)

# Display the DataFrame after one-hot encoding
print("\nDataFrame After One-Hot Encoding:")
print(df.to_string(index=False))
```

```
Initial DataFrame:
 Product ID Product Name  Price    Category  Stock              Description
          1     Widget A  19.99 Electronics  100.0    A high-quality widget
          2     Widget B  29.99 Electronics    NaN                     NaN
          3          NaN  15.00  Home Goods   50.0      Durable and stylish
          4     Widget D    NaN  Home Goods  200.0       A versatile widget
          5     Widget E   9.99         NaN   10.0    Compact and efficient
          6     Widget F  25.00 Electronics    0.0 Latest technology widget
          7     Widget G    NaN     Kitchen  150.0     Multi-purpose widget
          8     Widget H  39.99     Kitchen   75.0          Premium quality
          9     Widget I    NaN Electronics    NaN        Advanced features
         10     Widget J  49.99 Electronics   60.0            Best in class
```

```
DataFrame After One-Hot Encoding:
 Product ID Product Name  Price  Stock              Description
Category_Electronics  Category_Home Goods  Category_Kitchen  Category_nan
          1     Widget A  19.99  100.0    A high-quality widget
1.0                  0.0               0.0           0.0
          2     Widget B  29.99    NaN                     NaN
1.0                  0.0               0.0           0.0
          3          NaN  15.00   50.0      Durable and stylish
0.0                  1.0               0.0           0.0
          4     Widget D    NaN  200.0       A versatile widget
0.0                  1.0               0.0           0.0
          5     Widget E   9.99   10.0    Compact and efficient
0.0                  0.0               0.0           1.0
          6     Widget F  25.00    0.0 Latest technology widget
1.0                  0.0               0.0           0.0
          7     Widget G    NaN  150.0     Multi-purpose widget
0.0                  0.0               1.0           0.0
          8     Widget H  39.99   75.0          Premium quality
0.0                  0.0               1.0           0.0
          9     Widget I    NaN    NaN        Advanced features
1.0                  0.0               0.0           0.0
         10     Widget J  49.99   60.0            Best in class
1.0                  0.0               0.0           0.0
```

Explanation:

Read the Data: Load the dataset from the specified location using pd.read_csv().

Initial Display: Display the DataFrame to see the data before applying encoding.

Initialize Encoder: Initialize the OneHotEncoder from sklearn.preprocessing with sparse_output=False.

Apply One-Hot Encoding: Apply one-hot encoding to the 'Category' column, creating new binary columns for each category.

Concatenate and Drop: Concatenate the new columns to the original DataFrame and drop the original 'Category' column.

Final Display: Display the DataFrame after applying one-hot encoding.

This updated code should work correctly with the current version of scikit-learn.

2. Label Encoding:

Label Encoding involves converting categorical data into numerical labels. Each unique category is assigned an integer value. This technique is straightforward and can be useful when the categorical variable has an ordinal relationship.

[32]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

```python
# Read the data from the specified location
df = pd.read_csv('D:/Projects/Data-cleaning-series/Chapter03 Data␣
 ↪Transformation/Products.csv')

# Display the initial DataFrame
print("Initial DataFrame:")
print(df.to_string(index=False))

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Apply Label Encoding to the 'Category' column
df['Category_Encoded'] = label_encoder.fit_transform(df['Category'].astype(str))

# Display the DataFrame after label encoding
print("\nDataFrame After Label Encoding:")
print(df.to_string(index=False))
```

```
Initial DataFrame:
 Product ID Product Name  Price    Category  Stock              Description
          1     Widget A  19.99 Electronics  100.0    A high-quality widget
          2     Widget B  29.99 Electronics    NaN                     NaN
          3          NaN  15.00  Home Goods   50.0      Durable and stylish
          4     Widget D    NaN  Home Goods  200.0       A versatile widget
          5     Widget E   9.99         NaN   10.0    Compact and efficient
          6     Widget F  25.00 Electronics    0.0 Latest technology widget
          7     Widget G    NaN     Kitchen  150.0     Multi-purpose widget
          8     Widget H  39.99     Kitchen   75.0          Premium quality
          9     Widget I    NaN Electronics    NaN        Advanced features
         10     Widget J  49.99 Electronics   60.0            Best in class

DataFrame After Label Encoding:
 Product ID Product Name  Price    Category  Stock              Description
Category_Encoded
          1     Widget A  19.99 Electronics  100.0    A high-quality widget
0
          2     Widget B  29.99 Electronics    NaN                     NaN
0
          3          NaN  15.00  Home Goods   50.0      Durable and stylish
1
          4     Widget D    NaN  Home Goods  200.0       A versatile widget
1
          5     Widget E   9.99         NaN   10.0    Compact and efficient
3
          6     Widget F  25.00 Electronics    0.0 Latest technology widget
0
          7     Widget G    NaN     Kitchen  150.0     Multi-purpose widget
2
```

```
          8     Widget H  39.99     Kitchen   75.0          Premium quality
2
          9     Widget I    NaN Electronics    NaN        Advanced features
0
         10     Widget J  49.99 Electronics   60.0            Best in class
0
```

Explanation:

Read the Data: Load the dataset from the specified location using pd.read_csv().

Initial Display: Display the DataFrame to see the data before applying encoding.

Initialize LabelEncoder: Initialize the LabelEncoder from sklearn.preprocessing.

Apply Label Encoding: Convert the 'Category' column into numerical labels using fit_transform(). The new encoded labels are stored in a new column called 'Category_Encoded'.

Final Display: Display the DataFrame after applying label encoding, showing the original categories and their corresponding numerical labels.

This code will replace the categories in the 'Category' column with integer values that represent each unique category. If a category has missing values, they will be treated as a separate category or encoded as -1 depending on how missing values are handled.