JSPM's
Rajarshi Shahu College of Engineering, Tathwade Pune-33
(An Autonomous Institute Affiliated to Savitribai Phule Pune University, Pune)
Department of Computer Engineering
Academic Year -2025-2026 (Semester –V)

T. Y. B. Tech (Computer Engineering)
Professional Elective – I: [CS3203T-A]:- Data Mining and Analytics

Data Preparation **06 Hours**

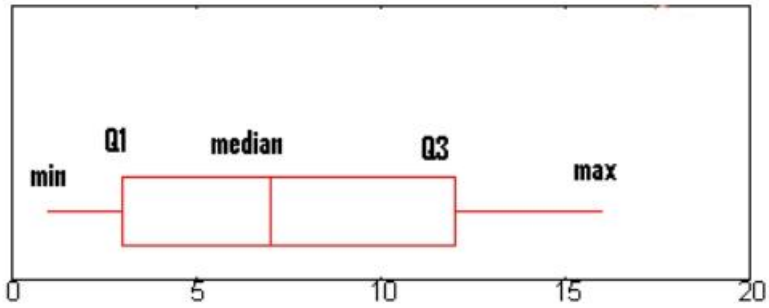
Data Cleaning: Handling missing values, outliers, and duplicates Data Transformation: Normalization, standardization, and scaling, Data encoding (e.g., one-hot encoding) , Datareduction, Discretization and generating concept hierarchies Feature Engineering: Types of Feature Selection Methods: Filter Methods, Select features based on Statistical measures. Techniques: Correlation Analysis ,Hypothesis Testing : Ttest, Ztest, Chi-Square Test, Mutual Information, ANOVA (Analysis of Variance), Wrapper Methods: Evaluate subsets of features using a machine learning model, Techniques: Forward Selection, Backward Elimination, Recursive Feature Elimination (RFE) Embedded Methods: LASSO (L1 Regularization), Ridge Regression (L2 Regularization) Tree based Models like Random Forest and XGBoost feature importance.

Tools and Framework: Installing Weka 3 Data Mining System, Experiments with Weka and Rapid miner

Outlier Detection"

How to Find a Five-Number Summary: Steps

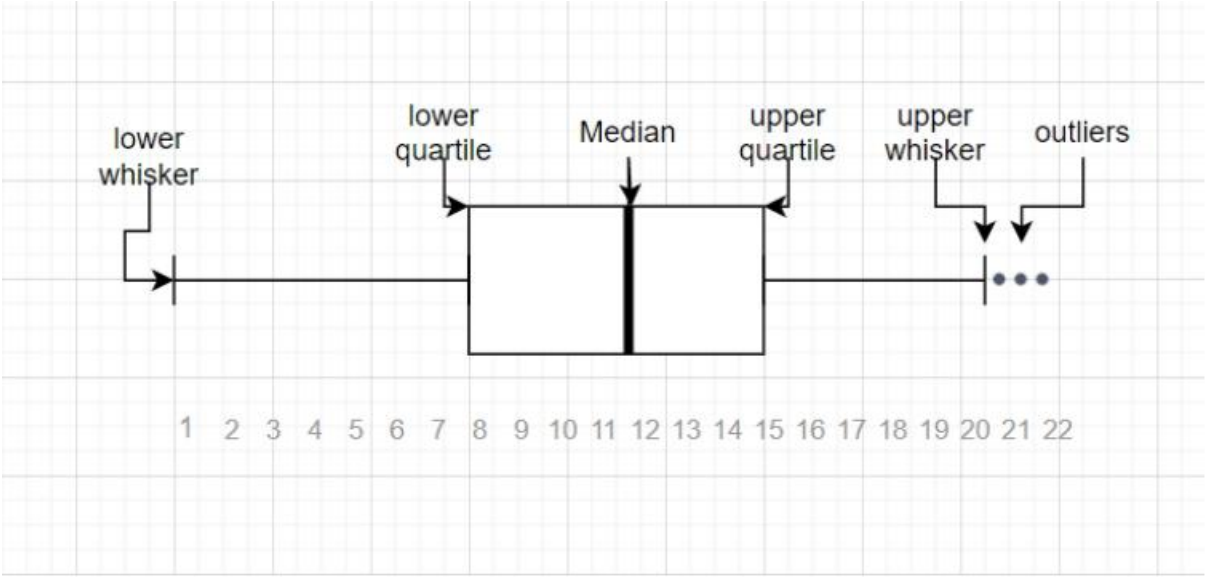
- Step 1: *Put your numbers in ascending order* (from smallest to largest). For this particular data set, the order is:
  Example: 1, 2, 5, 6, 7, 9, 12, 15, 18, 19, 27.
- Step 2: *Find the minimum and maximum* for your data set. Now that your numbers are in order, this should be easy to spot.
  In the example in step 1, the minimum (the smallest number) is 1 and the maximum (the largest number) is 27.
- Step 3: *Find the median*. The median is the middle number. If you aren't sure how to find the median,
- Step 4: *Place parentheses around the numbers above and below the median.*
  (This is not *technically* necessary, but it makes Q1 and Q3 easier to find).
  (1, 2, 5, 6, 7), 9, (12, 15, 18, 19, 27).
- Step 5: *Find Q1 and Q3*. Q1 can be thought of as a median in the lower half of the data, and Q3 can be thought of as a median for the upper half of data.
  (1, 2, 5, 6, 7), 9, (12, 15,18,19,27).
- Step 6: *Write down your summary found in the above steps*.
  minimum = 1, Q1 = 5, median = 9, Q3 = 18, and maximum = 27.



Finding the outlier points from Matplotlib

Outliers are the data points that differ from other observations or those which lie at a distance from the other data. They are mainly generated due to some *experimental error* which may cause several problems in statistical analysis. While in a big dataset it is quite obvious that some data will be further from the sample mean. These outliers need to be found and handle wisely.

We can use boxplots for the necessary.



Above is a diagram of boxplot created to display the summary of data values along with its *median, first quartile, third quartile, minimum and maximum*. And the data points out of the lower and upper whiskers are outliers. In between the first and third quartile of whisker lies the *interquartile region* above which a vertical line passes known as the median.

Following are the methods to find outliers from a boxplot :

1.Visualizing through matplotlib boxplot using *plt.boxplot().*

2.Using 1.5 IQR rule.

Example:

- Python3

```
# Adding libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
# random integers between 1 to 20
arr = np.random.randint(1, 20, size=30)

# two outliers taken
arr1 = np.append(arr, [27, 30])

print('Thus the array becomes{}'.format(arr1))
```

Output:

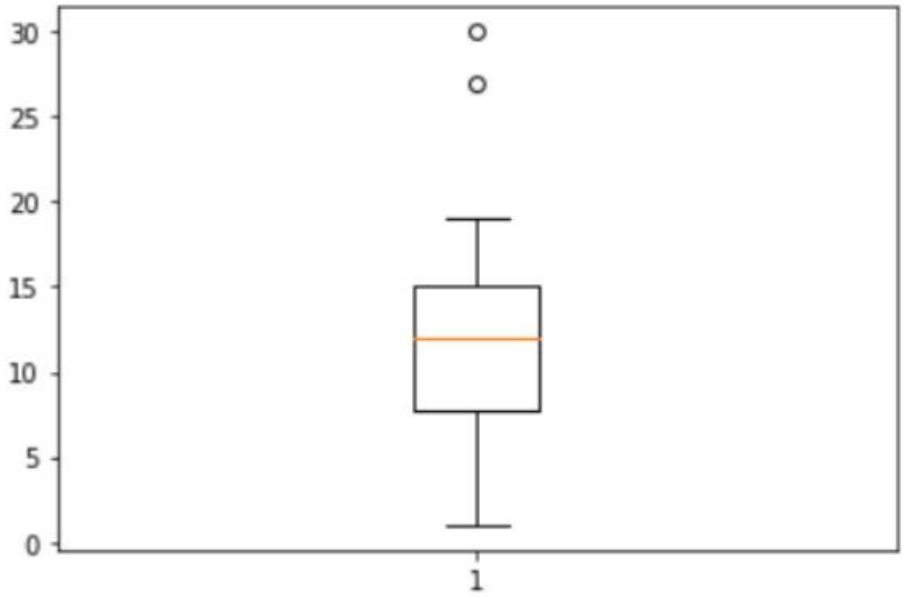
array([4, 12, 15, 7, 13, 2, 12, 11, 10, 12, 15, 5, 9, 16, 17, 2, 10, 15, 4, 16, 14, 19, 12, 8, 13, 3, 16, 10, 1, 13, 27, 30])

Visualizing by matplotlib boxplot using plt.boxplot()

- Python3

```
plt.boxplot(arr1)
fig = plt.figure(figsize =(10, 7))
plt.show()
```

Output:




```
<Figure size 720x504 with 0 Axes>
```

So from the above figure, we can witness the two outliers.

1.5 IQR Rule

Steps in 1.5IQR rule:-

- Finding the median, quartile, and interquartile regions
- Calculate 1.5*IQR below the first quartile and check for low outliers.

- Calculate 1.5*IQR above the third quartile and check for outliers.

- Python

```
# finding the 1st quartile
q1 = np.quantile(arr1, 0.25)

# finding the 3rd quartile
q3 = np.quantile(arr1, 0.75)
med = np.median(arr1)

# finding the iqr region
iqr = q3-q1

# finding upper and lower whiskers
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)
print(iqr, upper_bound, lower_bound)
```

Output:
8.25 26.375 -6.625

- Python3

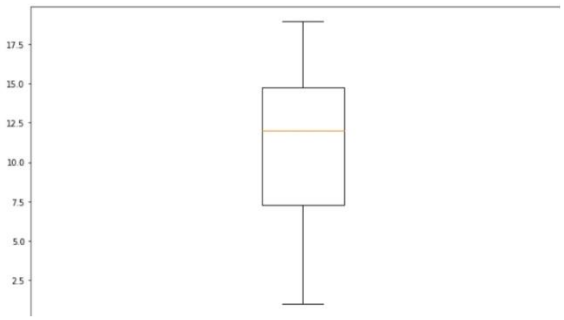
```
outliers = arr1[(arr1 <= lower_bound) | (arr1 >= upper_bound)]
print('The following are the outliers in the boxplot:{}'.format(outliers))
```

Output:
The following are the outliers in the boxplot:[27 30]
Thus, the outliers have been detected using the rule. Now eliminating them and plotting a graph with the data points-

- Python3

```
# boxplot of data within the whisker
arr2 = arr1[(arr1 >= lower_bound) & (arr1 <= upper_bound)]
plt.figure(figsize=(12, 7))
plt.boxplot(arr2)
plt.show()
```

Output :



Z-score

Z- Score is also called a standard score. This value/score helps to understand that how far is the data point from the mean. And after setting up a threshold value one can utilize z score values of data points to define the outliers.

Zscore = (data_point -mean) / std. deviation

In this example, we are calculating the Z scores for the 'age' column in the DataFrame df_diabetics using the zscore function from the SciPy stats module. The resulting array z contains the absolute Z scores for each data point in the 'age' column, indicating how many standard deviations each value is from the mean.

Python

```
from scipy import stats
import numpy as np
z = np.abs(stats.zscore(df_diabetics['age']))
print(z)
```

Output:

```
0     0.800500
1     0.039567
2     1.793307
3     1.872441
4     0.113172
      ...
437   0.876870
438   0.115937
439   0.876870
440   0.956004
441   0.956004
Name: age, Length: 442, dtype: float64
```

Now to define an outlier threshold value is chosen which is generally 3.0. As 99.7% of the data points lie between +/- 3 standard deviation (using Gaussian Distribution approach).

Removal of Outliers with Z-Score

Let's remove rows where Z value is greater than 2.

In this example, we sets a threshold value of 2 and then uses NumPy's np.where() to identify the positions (indices) in the Z-score array z where the absolute Z score is greater than the specified threshold (2). It prints the positions of the outliers in the 'age' column based on the Z-score criterion.

Python

```
import numpy as np
 threshold_z = 2
 outlier_indices = np.where(z > threshold_z)[0]
no_outliers = df_diabetics.drop(outlier_indices)
 print("Original DataFrame Shape:", df_diabetics.shape)
print("DataFrame Shape after Removing Outliers:", no_outliers.shape)
```

Output:

Original DataFrame Shape: (442, 10)
DataFrame Shape after Removing Outliers: (426, 10)
IQR (Inter Quartile Range)
IQR (Inter Quartile Range) Inter Quartile Range approach to finding the outliers is the most commonly used and most trusted approach used in the research field.
IQR = Quartile3 – Quartile1
Syntax : numpy.percentile(arr, n, axis=None, out=None)

Parameters :

- *arr :input array.*
- *n : percentile value.*

In this example, we are calculating the interquartile range (IQR) for the 'bmi' column in the DataFrame df_diabetics . It first computes the first quartile (Q1) and third quartile (Q3) using the midpoint method, then calculates the IQR as the difference between Q3 and Q1, providing a measure of the spread of the middle 50% of the data in the 'bmi' column.

Python

```
# IQR
Q1 = np.percentile(df_diabetics['bmi'], 25, method='midpoint')
Q3 = np.percentile(df_diabetics['bmi'], 75, method='midpoint')
IQR = Q3 - Q1
print(IQR)
```

Output

```
0.06520763046978838
```

To define the outlier base value is defined above and below dataset's normal range namely Upper and Lower bounds, define the upper and the lower bound (1.5*IQR value is considered) :
*upper = Q3 +1.5*IQR*
*lower = Q1 – 1.5*IQR*
In the above formula as according to statistics, the 0.5 scale-up of IQR (new_IQR = IQR + 0.5*IQR) is taken, to consider all the data between 2.7 standard deviations in the Gaussian Distribution.
Python

```
# Above Upper bound
upper = Q3+1.5*IQR
upper_array = np.array(df_diabetics['bmi'] >= upper)
print("Upper Bound:", upper)
print(upper_array.sum())

# Below Lower bound
lower = Q1-1.5*IQR
lower_array = np.array(df_diabetics['bmi'] <= lower)
print("Lower Bound:", lower)
print(lower_array.sum())
```

Output:
Upper Bound: 0.128790008117763063
Lower Bound: -0.132040513761390450
Outlier Removal in Dataset using IQR
In this example, we are using the interquartile range (IQR) method to detect and remove outliers in the 'bmi' column of the diabetes dataset. It calculates the upper and lower limits based on the IQR, identifies outlier indices using Boolean arrays, and then removes the corresponding rows from the DataFrame, resulting in a new DataFrame with outliers excluded. The before and after shapes of the DataFrame are printed for comparison.

Data Transformation

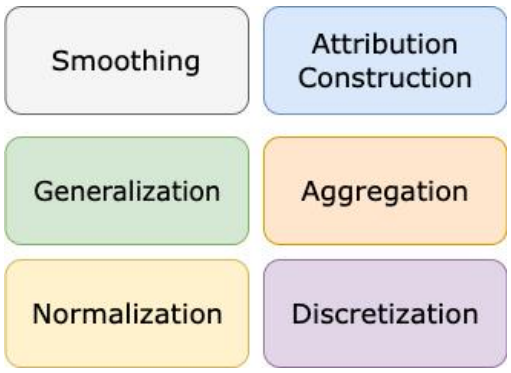
Analyzing data is a critical task for decision-making. So, your dataset must have no inconsistencies otherwise you may end up with wrong insights. To make your data clean and ready for analysis, you need to organize and transform it. For this, you can use different data transformation techniques.

What are data transformation techniques?

Data transformation techniques refer to all the actions that help you transform your raw data into a clean and ready-to-use dataset.

There are different types of data transformation techniques that offer a unique way of transforming your data and there is a chance that you won't need all of these techniques on every project. Nevertheless, it's important to understand what each technique can offer so you can choose what's best for your case.

Different types of data transformation techniques



There are 6 basic data transformation techniques that you can use in your analysis project or data pipeline:

- Data Smoothing
- Attribution Construction
- Data Generalization
- Data Aggregation
- Data Discretization
- Data Normalization

Data Smoothing

Smoothing is a technique where you apply an algorithm in order to remove noise from your dataset when trying to identify a trend. Noise can have a bad effect on your data and by eliminating or reducing it you can extract better insights or identify patterns that you wouldn't see otherwise.

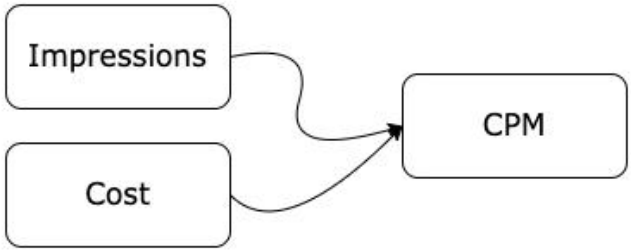
There are 3 algorithm types that help with data smoothing:

- **Clustering**: Where you can group similar values together to form a cluster while labeling any value out of the cluster as an outlier.
- **Binning**: Using an algorithm for binning will help you split the data into bins and smooth the data value within each bin.
- **Regression**: Regression algorithms are used to identify the relation between two dependent attributes and help you predict an attribute based on the value of the other.

Attribution Construction

Attribution construction is one of the most common techniques in data transformation pipelines. Attribution construction or feature construction is the process of creating new features from a set of existing features/attributes in the dataset.

Imagine working in marketing and trying to analyze the performance of a campaign. You have all the impressions that your campaign generated and the total cost for the given time frame. Instead of trying to compare these two metrics across all of your campaigns, you can construct another metric to calculate the cost per million impressions or CPM.



This will make your data mining and analysis process a lot easier, as you'll be able to compare the campaign performance on a single metric rather than two separate metrics.

Data Generalization

Data generalization refers to the process of transforming low-level attributes into high-level ones by using the concept of hierarchy. Data generalization is applied to categorical data where they have a finite but large number of distinct values.

This is something that we, as people, are already doing without noticing and it helps us get a clearer picture of the data. Let's say we have 4 categorical attributes in our database:

1. City
2. Street
3. Country
4. State/province.

We can define a hierarchy between these attributes by specifying the total ordering among them at the schema level, for example:

street < city < state/province < country.

Data Aggregation

Data aggregation is possibly one of the most popular techniques in data transformation. When you're applying data aggregation to your raw data you are essentially storing and presenting data in a summary format.

This is ideal when you want to perform statistical analysis of your data as you might want to aggregate your data over a specific time period and provide statistics such as average, sum, minimum, and maximum.

	A	B
1	**Date**	**Temperature, C**
2	1981-01-01	20.7
3	1981-01-02	17.9
4	1981-01-03	18.8
5	1981-01-04	14.6
6	1981-01-05	15.8
7	1981-01-06	15.8
8	1981-01-07	15.8
9	1981-01-08	17.4
10	1981-01-09	21.8
11	1981-01-10	20
12	1981-01-11	16.2
13	1981-01-12	13.3
14	1981-01-13	16.7
15	1981-01-14	21.5
16	1981-01-15	25
17	1981-01-16	20.7

Date - Month	AVERAGE of Temprature
Jan	17.14032258
Feb	16.8
Mar	14.21774194
Apr	11.91333333
May	9.537096774
Jun	6.456666667
Jul	6.109677419
Aug	7.290566038
Sep	10.14333333
Oct	10.08709677
Nov	11.89
Dec	13.68064516

In the above example, we aggregated our daily temperature data to see the average temperature for each particular month. As a result, you can find which month has the minimum average temperature in the glance of an eye.

Data Discretization

Data discretization refers to the process of transforming continuous data into a set of data intervals. This is an especially useful technique that can help you make the data easier to study and analyze and improve the efficiency of any applied algorithm.

Imagine having tens of thousands of rows representing people in a survey providing their first name, last name, age, and gender. Age is a numerical attribute that can have a lot of different values. To make our life easier we can divide the range of this continuous attribute into intervals.

Mapping this attribute to a higher-level concept, like youth, middle-aged, and senior, can help a lot with the efficiency of the task and improve the speed of the algorithms applied.

Data Normalization

Last but not least, data normalization is the process of scaling the data to a much smaller range, without losing information to help minimize or exclude duplicated data and improve algorithm efficiency and data extraction performance.

There are three methods to normalize an attribute:

- **Min-max normalization:** Where you perform a linear transformation on the original data.
- **Z-score normalization:** In z-score normalization (or zero-mean normalization) you are normalizing the value for attribute *A* using the mean and standard deviation.
- **Decimal scaling:** Where you can normalize the value of attribute *A* by moving the decimal point in the value.

Normalization methods are frequently used when you have values that skew your dataset and you find it hard to extract valuable insights.

Other ways of transforming data

The techniques we went through in the previous sections are considered to be the standard data transformation techniques used in almost every analytics project.

In addition to the above, there are two other ways you can transform your data to be able to analyze it and extract valuable insights.

Data Integration

Data integration is not a data transformation technique but rather a critical step during the pre-processing phase.

Data integration is the process of bringing together information from different sources to create a unified view of the data. These sources can be:

- Business applications including marketing, accounting, CRM, and other software
- Traditional databases
- Data warehouses
- Simple CSV or Excel files
- Reports or dashboards

The destinations for the data integrated from sources include spreadsheets, data warehouses, and even BI tools.

For instance, Coupler.io provides data connectors to 60+ data sources with the following destinations:

- Spreadsheets (Google Sheets, Excel)
- Data warehouses (BigQuery, Amazon Redshift, PostgreSQL)
- BI tools (Looker Studio, Power BI, Tableau, Qlik)

Try it yourself by selecting the needed source and destination apps in the form below and click **Proceed**.

You can create a Coupler.io account for free and set up an integration with three simple steps:

- Collect data
- Transform data
- Schedule data refresh

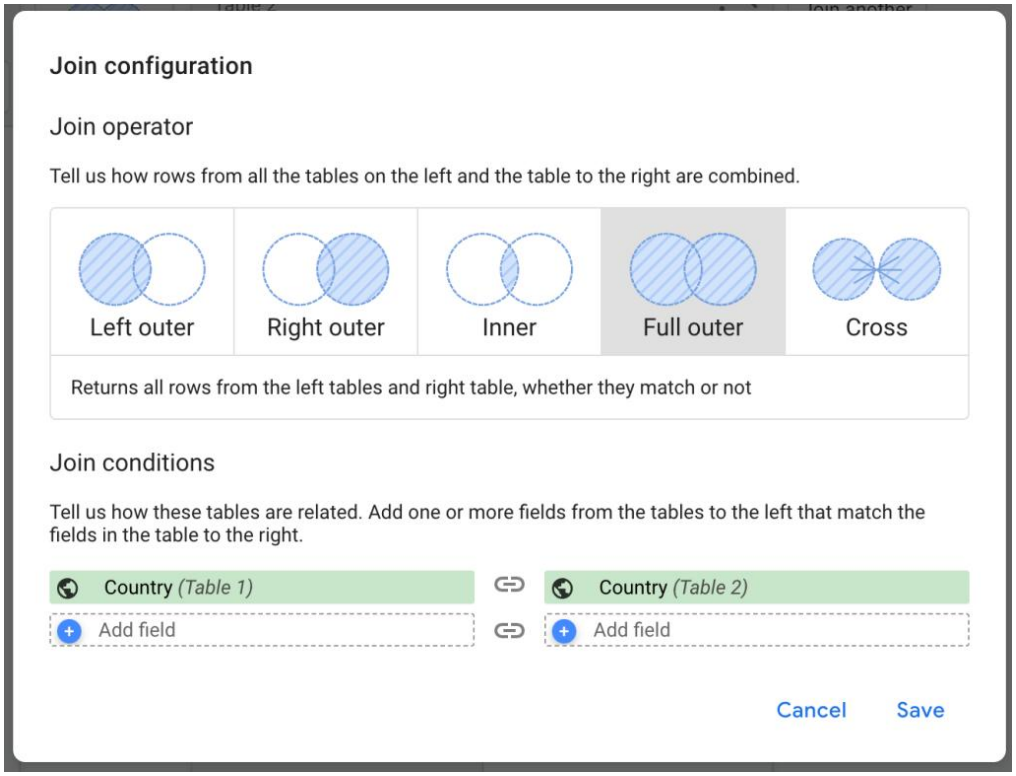
At the **Transform data** step, you can filter and sort data, add custom columns, hide unnecessary columns, and blend data from other sources…

Data Blending

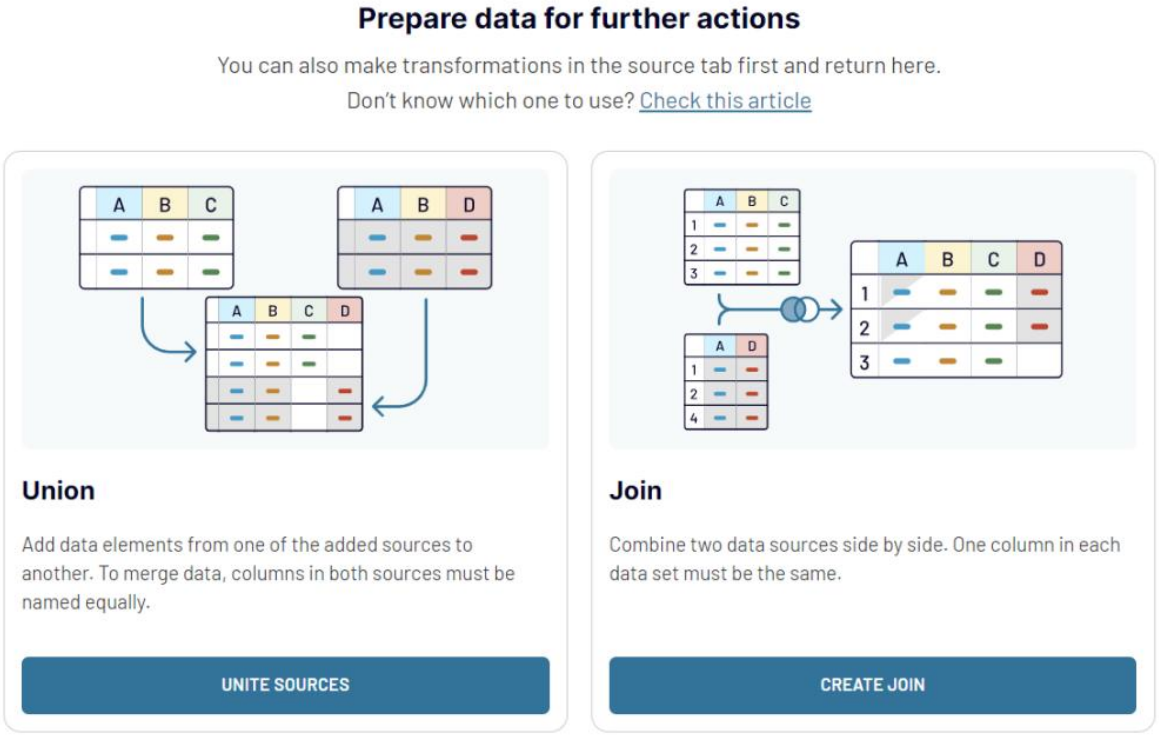
Data blending is the process of combining data from multiple sources. Sometimes this process is also referred to as data join. *Isn't data integration the same though?*

Not exactly, since when you integrate data, you can only extract data from one source or database. With data blending, you can expand your reach to multiple data sources.

Data integration software and automated reporting tools may have different approaches or techniques of data blending. For example, in Looker Studio, you can choose from 5 types of data blending.



Coupler.io allows you to combine data from multiple sources using the two most widespread approaches:



As a result, you can blend your data on the go and load it to the chosen destination in the analysis-ready format.

Data Manipulation

Data manipulation refers to the process of making your data more readable and organized. This can be achieved by changing or altering your raw datasets.

Data manipulation tools can help you identify patterns in your data and apply any data transformation technique (e.g. attribution creation, normalization, or aggregation) in an efficient and easy way.