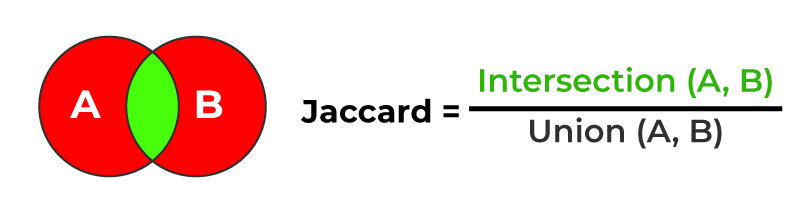
## **What is Jaccard Similarity?**

**Jaccard Similarity** also known as Jaccard index, is a statistic to measure the similarity between two data sets. It is measured as the size of the intersection of two sets divided by the size of their union.

For example: Given two sets A and B, their Jaccard Similarity is provided by,



accard Similarity *J*(*A*,*B*)=

∣*A*∪*B*∣

∣*A*∩*B*∣

​

Where:

* ∣A∩B∣
* ∣*A*∩*B*∣ is the cardinality (size) of the intersection of sets A and B.
* ∣A∪B∣
* ∣*A*∪*B*∣ is the cardinality (size) of the union of sets A and B.

Jaccard Similarity is also known as the Jaccard index or Jaccard coefficient, its values lie between 0 and 1. where 0 means no similarity and the values get closer to 1 means increasing similarity 1 means the same datasets

Example:

A = {1,2,3,4,6}

B = {1,2,5,8,9}

# Intersection and Union of two sets can also be done using & and | operators.

C = A.intersection(B)

D = A.union(B)

print('AnB = ', C)

print('AUB = ', D)

print('J(A,B) = ', float(len(C))/float(len(D)))

Output:

AnB = {1, 2}

AUB = {1, 2, 3, 4, 5, 6, 8, 9}

J(A,B) = 0.25

#### **EXAMPLE: 2**

The Jaccard similarity can be used to compare the similarity of two sets of words, which are frequently represented as sets of unique terms.

**def** jaccard\_similarity(set1, set2):

*# intersection of two sets*

intersection = len(set1.intersection(set2))

*# Unions of two sets*

union = len(set1.union(set2))

**return** intersection / union

set\_a = {"Mumbai", "Pune", "Satara", "Gujrat", "Punjab"}

set\_b = {"Mumbai", "Pune", "Satara", "Sangli", 'Nashik', "Haryana"}

similarity = jaccard\_similarity(set\_a, set\_b)

print("Jaccard Similarity:", similarity)

**Output**:

Jaccard Similarity: 0.25

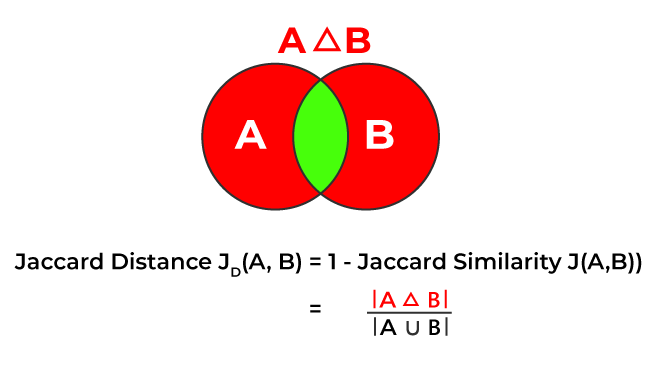
### **Significance of Jaccard Similarity**

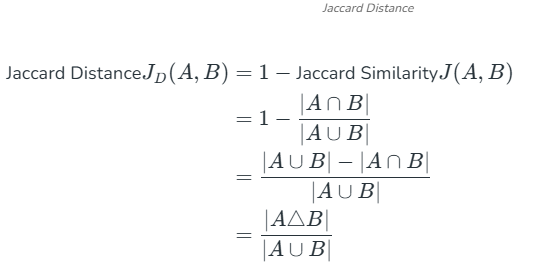
The Jaccard similarity is especially effective when the order of items is irrelevant and only the presence or absence of elements is examined. It is extensively used in:

1. Text Analysis: Jaccard similarity can be used in natural language processing to compare texts, text samples, or even individual words.
2. Recommendation Systems: Jaccard similarity can help in finding similar items or products based on user behavior.
3. Data Deduplication: Jaccard similarity can be used to find duplicate or near-duplicate records in a dataset.
4. Social Network Analysis: Jaccard similarity can be used in social networks to detect similarities between user profiles or groups.
5. Genomics: Jaccard similarity is employed to compare gene sets in biological studies.

## **Jaccard Distance**

The Jaccard distance is a measure of how different two sets are i.e Unlike the Jaccard coefficient, which determines the similarity of two sets. The Jaccard distance is computed by subtracting the Jaccard coefficient from one, or by dividing the difference in the sizes of the union and the intersection of two sets by the size of the union.





Where:

* ∣A∩B∣
* ∣*A*∩*B*∣ is the cardinality (size) of the intersection of sets A and B.
* ∣A∪B∣
* ∣*A*∪*B*∣ is the cardinality (size) of the union of sets A and B.
* ∣A△B∣
* ∣*A*△*B*∣ represents the cardinality (size) of symmetric difference of sets (A) and (B), containing elements that are in either set but not in their intersection.

The Jaccard distance is often used to calculate a nxn matrix For clustering and multidimensional scaling of n sample sets. This distance is a collection metric for all finite sets.

def jaccard\_distance(set1, set2):

#Symmetric difference of two sets

Symmetric\_difference = set1.symmetric\_difference(set2)

# Unions of two sets

union = set1.union(set2)

return len(Symmetric\_difference)/len(union)

set\_a = {"Mumbai", "Pune", "Satara", "Gujrat", "Punjab"}

set\_b = {"Mumbai", "Pune", "Satara", "Sangli", 'Nashik', "Haryana"}

distance = jaccard\_distance(set\_a, set\_b)

print("Jaccard distance:", distance)

Output:

Jaccard distance: 0.75

Program to Implement the Jaccard\_Distance:

def jaccard\_distance(set1, set2):

"""

Calculates the Jaccard distance between two sets.

Args:

set1: The first set.

set2: The second set.

Returns:

The Jaccard distance as a float.

"""

intersection = len(set1.intersection(set2))

union = len(set1.union(set2))

# Handle the case where both sets are empty to avoid division by zero

if union == 0:

return 0.0 # Jaccard distance is 0 if both sets are empty

jaccard\_similarity = intersection / union

jaccard\_dist = 1 - jaccard\_similarity

return jaccard\_dist

# Example usage:

set\_a = {"apple", "banana", "orange", "kiwi"}

set\_b = {"banana", "grape", "kiwi", "pineapple"}

distance = jaccard\_distance(set\_a, set\_b)

print(f"Jaccard distance between set\_a and set\_b: {distance}")

set\_empty1 = set()

set\_empty2 = set()

distance\_empty = jaccard\_distance(set\_empty1, set\_empty2)

print(f"Jaccard distance between two empty sets: {distance\_empty}")