1. Problem Statement

Our project is focused around helping business owners find potential competitors if they choose to open a branch in a new location. Using this information the business owners can not only find potential competitors, but also get an idea of how well their business will be received in the new locality.

We say that a “competitor” is any similar business, with the idea that a similar business caters to a similar clientele and is therefore a competitor. Therefore our task involves finding similar businesses in other regions. We do this by first clustering similar businesses. Once we have the clusters, to find similar businesses in a location, we look into the cluster to which a given restaurant belongs to, and then display restaurants in that cluster that are close to the location.

We use the k-medoids algorithm to cluster the businesses as not all attributes are numeric in nature and we need a richer set of dissimilarity scores. We run experiments with various parameter settings, dissimilarity scores, and features and report the rand index on a “gold standard” data set.

2. Algorithm Formulation

We use the K-medoid clustering algorithm to cluster businesses. K-medoids is a clustering technique that tries to minimize the pairwise dissimilarity between data points that are assigned to a cluster and the medoid of that cluster. The medoid is a data point in data set that best represent the cluster center. The medoid is analogous to the centroid in the K-Means algorithm.

The K-Medoids algorithm has the following advantage over K-Means.

• Since we are only using pairwise comparisons, we do not need to be able to calculate the average of a feature. This is better for non-numeric features like strings or sets.

• Since we are only using pairwise comparisons, once we compute the dissimilarity between the data points, we can do away with the actual data points, and this leads to a reduction in the memory usage and makes the computation more efficient.

• Since our medoid is a real data point (as opposed to a centroid), we can use this data point as a “representative” when visualizing our data (or perhaps even sampling it).

Our stopping condition is either when we have run through 10 iterations of clustering (experimentally found to be sufficient with this dataset), or when the current set of clusters is the same as either the last run or the run before it. We compare not just the most recent run so that we can prevent unnecessary runs when the clusters are jittering, or when outlier points are constantly shifting their membership back and forth between two or more clusters.

3. Distance Metrics

To compare our features, we use several different distance metrics. In this section, we will discuss only metrics that were used in our final experiments. However, we implemented and explored several different metrics that will be covered in Appendix B.

Our distance metrics fall into two categories: numeric and set-based. For numeric, we implemented L1 and L2 distances. For set-based, we implemented Jaccard and Dice.
To make combining the distance of different features together easier, we also attempt to normalize the output of the distance to ideally be in the range 0 to 1.

3.1. L1 (Manhattan) Distance
The Manhattan distance of two numbers \( x \) and \( y \) is calculated by:

\[
d = \sum_{i=1}^{n} |x_i - y_i|
\]

3.2. L2 (Euclidean) Distance
The Euclidean distance of two numbers \( x \) and \( y \) is calculated by:

\[
d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
\]

3.3. Jaccard Distance
The Jaccard distance of two set \( A \) and \( B \) is calculated by:

\[
d_j(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}
\]

3.4. Dice distance
The Dice distance of two set \( A \) and \( B \) is calculated by:

\[
d_d(A, B) = 1 - \frac{2|A \cap B|}{|A| + |B|}
\]

3.5. Normalizations
Because different distance metrics work with different output ranges, we need to normalize the distances metrics to something consistent. For example, the range of Jaccard Distance is between 0 and 1 while the range of Manhattan Distance is 0 to infinity.

To figure out what works best, we will try three different normalization methods:

1. **Raw** - No normalization is applied.

2. **Logarithmic** - Use a logarithm to squash down the value closer to zero. Since we did not want to let values less than 1 grow, we put a hinge at 1 and just let all values less than 1 go to zero.

\[
distance(d) = \ln(\max(1, d))
\]

3. **Logistic** - Use the Sigmoid function to squash the distance down to a 0-1 range. Since all our distance metrics return non-negative values, we can ensure that the following returns a value in the range of 0 to 1:

\[
distance(d) = \left( \frac{1}{1 - \exp(-d)} - 0.5 \right) \times 2
\]

4. Features
Our set of features can be divided into three types of features:

- **Numeric Features**
  - **Star Rating** - The star ratings of a business, rounded to half-stars.
  - **Total Review Count** - The total number of reviews present for the business.
  - **Available Review Count** - The number of reviews available for the business in the data set.
  - **Mean Review Length** - Average length of a review for the business.
  - **Mean Word Length** - Average length of the words in the reviews for the business.
  - **Number of Words** - Total number of words in the reviews of the business.
  - **Mean Words per Review** - Average number of words per review available for the business.

- **Descriptive Features**
  - **Attribute Features** - The dataset contains information about attributes of businesses which describe the operations of the business. They exist as key-value pairs in the data, for example - (WiFi, no), (Delivery, false), (Attire, casual). We squash the key-value pairs together and add these together to create a set of attributes that a business has, which we then use as a feature.
  - **Category Features** - The dataset also contains some categorical information about the business, for example, whether the business is a restaurant, cafe, food place, burgers place, etc. We construct the feature which is the set of categories that the business has assigned to it.
Key Words - These are words that Yelp has defined to help users in filtering out the businesses that appear in the search results. They’re words that delineate businesses as they’re mostly categorical words such as restaurant, cafes, etc. We use these key words and look for their occurrences in the reviews of the businesses and return a set of key words that they contain.

Top Words - This set contains the most frequently occurring words in the reviews of the text, after taking care of the stop words. We used a general English language stop words list containing 562 stop words.

Temporal Features
We have two time-related features pertaining to the functioning hours of businesses

Total Hours - Total number of hours the business is open during the week

Open Hours - This is a feature encoded information about the functioning hours of the business over the week. We divided the hours in a day in the following way to help us attribute the functioning times to a business.

- Open between 6AM - 12PM: The restaurant functions in the morning, or serves breakfast.
- Open between 12PM - 3PM: The restaurant functions in the afternoon, or serves lunch.
- Open between 5PM - 9PM: The restaurant functions in the evening, or serves dinner.
- Open between 9PM - 2AM: The restaurant functions post dinner, or late night.

However, to make our feature more robust, we specify that to encode that a business functions during a specific time-span specified above, it must be open for at least 4 days in a week during those hours. We again return a set of the time-spans that a business operates during as a set.

5. Evaluation

In order to evaluate the correctness of the generated clusters, we create sets of restaurants that are similar. Using these sets as the “gold standard” clusters, we report the Rand Index.

5.1. Rand Index

Rand Index is used in data clustering to measure the similarity between two cluster assignments. Given a set of elements \( S \), and two partitions of \( S \), \( X = \{x_1, x_2, \ldots, x_n\} \) and \( Y = \{y_1, y_2, \ldots, y_m\} \), we compute the following:

- \( a \) - No. of pairs that are assigned to same cluster in both \( X \) and \( Y \)
- \( b \) - No. of pairs that are assigned to different clusters in both \( X \) and \( Y \)
- \( c \) - No. of pairs that are assigned to same cluster in \( X \) but to different clusters in \( Y \)
- \( d \) - No. of pairs that are assigned to different clusters in \( X \) but to same cluster in \( Y \)

The rand index is then given by

\[
R = \frac{a + b}{a + b + c + d}
\]

5.2. Generating Gold Standard Clusters

Since we do not have the gold standard clusters, we cluster a subset of the data and evaluate the algorithm on these data points.

We looks at various restaurant chains such as “Taco Bell”, “Starbucks” etc. that are present in the data and assign all the stores belonging to the same chain as a new cluster. We extracted the top 15 restaurant chains and their details are given in Table 1. We only generate pairs within a restaurant chain. Since it is not clear if a McDonald’s and a Burger King should be in the same or different cluster, we do not look at pairs across various chains.

However if we only have positive pairing, then the rand index can be trivially maximized by assigning all data points to the same cluster. To counteract this, we also need pairs of restaurants that should not be in the same cluster. We collected a list of 285 “fine-dining” restaurants (restaurants which have the highest price range) and create pairs such that, the first restaurant comes from the “fast food” list like “McDonald’s” and “Taco Bell” and the other from the fine-dining list. These pairs should not be in the same cluster.

6. Experiments

We use all 3069 restaurants present in the gold standard dataset for our experiments.

There are four different parameters that we can tune in our algorithm:

1. \( D \) - the set of features
2. \( K \) - the number of clusters
3. \( F \) - the function used to normalize various distance metrics
4. \( S \) - the distance measure used to measure set similarity

We run multiple experiments, keeping a few of these parameters fixed, and altering others to better understand the sensitivity of each parameter.
In the first experiment, we modify the feature set keeping other parameters fixed. The possible features are \(N\) (Numeric features), \(A\) (Attribute descriptive features (attributes and categories)), \(W\) (Word descriptive features (key words and top words))\(^1\). We set \(K = 10\), \(F\) to logistic normalization, and \(S\) to Dice. The results are shown in table 2.

We observe that the combination of Attribute and Word gives the best performance, followed by Attribute alone. We also observe that numeric features lead to deterioration of performance.

In the second experiment, we keep the number of clusters \(K\), the feature set fixed and alter the normalization function \(F\) and the set distance. We use the setting \(K = 10\), and use all the features (Numeric, Attribute, and Word). The results are shown in table 3.

We observe that the best performance is achieved when we use Jaccard similarity. We also observe that using normalization does not make any difference to the metrics.

In our third experiment, we modify the number of clusters \(K\), keeping the other parameters fixed. We set the set distance to Jaccard and the normalization to logistic and use all the features (Numeric, Attribute, and Word). The metrics are given in table 4.

We observe that the best setting for \(K\) is 10. We also observe that as we increase the number of clusters from 10, the rand index decreases. This could be due to the good clusters being split into many clusters resulting in decreasing rand index. Another reason could be that the number of “same clusters” pairs in the gold standard dataset is much higher than the number of “across cluster” pairs. This could result in rand index preferring fewer and larger clusters over many small clusters.

The full results can be seen in Appendix D.

### 7. Conclusion

Clustering works fairly well for this dataset. The signals that most strongly indicate that two businesses are the same come from the reviews rather than the more structured data like number of stars. Our work with the textual features was limited to just a few instances of “low-hanging fruit” But if the success of those features is any indication of the richness of the reviews, then the reviews should be the focus of future work.

---

\(^1\)Because of time constraints, temporal features were not included in these experiments. Partial results for temporal features can be found in Appendix A.
Appendices

A. Temporal Feature Experiments

Time constraints lead us to not being able to complete the experiments with Temporal features. However, we do have partial results.

It does not look like Temporal features are stronger than Attribute or Word features. However, we see Temporal features performing better than most combinations which include Numeric features. We already know that the Numeric features are detrimental, but it is very interesting that Temporal beats it when we consider what is represented by each feature. Numeric features contain information such as the restaurants star rating, but Temporal features just have information about when a place is open.

<table>
<thead>
<tr>
<th>Feature Set</th>
<th>K</th>
<th>Scalar Normalization</th>
<th>Set Distance</th>
<th>Rand Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>NWT</td>
<td>8</td>
<td>Log</td>
<td>Dice</td>
<td>0.694153</td>
</tr>
<tr>
<td>NWT</td>
<td>8</td>
<td>Log</td>
<td>Jaccard</td>
<td>0.686108</td>
</tr>
<tr>
<td>NWT</td>
<td>8</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.774644</td>
</tr>
<tr>
<td>NWT</td>
<td>8</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.764851</td>
</tr>
<tr>
<td>NWT</td>
<td>10</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.700142</td>
</tr>
<tr>
<td>NWT</td>
<td>10</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.695933</td>
</tr>
<tr>
<td>NWT</td>
<td>10</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.761750</td>
</tr>
<tr>
<td>NWT</td>
<td>12</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.708275</td>
</tr>
<tr>
<td>NWT</td>
<td>12</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.787857</td>
</tr>
<tr>
<td>NWT</td>
<td>14</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.749919</td>
</tr>
<tr>
<td>NWT</td>
<td>14</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.704690</td>
</tr>
<tr>
<td>NWT</td>
<td>14</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.708275</td>
</tr>
<tr>
<td>NWT</td>
<td>14</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.765841</td>
</tr>
<tr>
<td>NWT</td>
<td>14</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.747906</td>
</tr>
<tr>
<td>NWT</td>
<td>10</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.718515</td>
</tr>
<tr>
<td>NWT</td>
<td>10</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.742525</td>
</tr>
<tr>
<td>NWT</td>
<td>10</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.703385</td>
</tr>
<tr>
<td>T</td>
<td>10</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.779835</td>
</tr>
</tbody>
</table>

B. Additional Distance Metrics

We implemented and explored using string similarity metrics to compute distance. However, we were not able to find suitable features to use string distance with.

B.1. Levenshtein Distance

The distance of two strings a and b is evaluated by Levenshtein distance:

\[
\text{lev}_{a,b}(i,j) = \begin{cases} 
\max(i,j), & \text{if } i = 0 \text{ or } j = 0 \\
\min( \begin{cases} 
\text{lev}_{a,b}(i-1,j) + 1 \\
\text{lev}_{a,b}(i,j-1) + 1 \\
\text{lev}_{a,b}(i-1,j-1) + 1(a_i \neq b_j) 
\end{cases} ) & \text{otherwise}
\end{cases}
\]

Then we normalize the Levenshtein distance \(\text{lev}_{a,b}\) by transaction function to the range of [0, 1]. \(l_a\) and \(l_b\) are the length of string ’a’ and ’b’.

\[
dis = \frac{\text{lev}_{a,b}}{\max(l_a, l_b)}
\]

B.2. Needleman-Wunsch Distance

The distance of two string ’a’ and ’b’ is calculated by Needleman-Wunsch algorithm. We calculate the distance by comparing the letters in two strings. The scoring way is as follows:

- Match = 1: two letters \(a_i\) and \(b_j\) are the same
- Mismatch = -1: two letters \(a_i\) and \(b_j\) are the different
- Indel = -1:
  - Delete: one letter in the string ’a’ aligns to a gap in the string ’b’.
  - Insert: one letter in the string ’b’ aligns to a gap in the string ’a’.

The pseudo-code for the Needleman-Wunsch algorithm is as follows:

```plaintext
for i = 0 to length(a) do
  F(i, 0) ← i
end for

for j = 0 to length(b) do
  F(0, j) ← j
end for

for i = 1 to length(a) do
  for j = 1 to length(b) do
    if a[i] == a[j] then
      Match ← F(i−1, j−1) + 1
    else
      Mismatch ← F(i−1, j−1) − 1
    end if
    Insert ← F(i, j−1) − 1
    Delete ← F(i−1, j) − 1
    F(i, j) ← max(Match, Mismatch, Insert, Delete)
  end for
end for
```

Then we need to normalize the Needleman-Wunsch distance \(NW_a, b\) by transaction function to the range of [0, 1]. \(l_a\) and \(l_b\) are the length of string ’a’ and ’b’.

\[
dis = \frac{NW_a, b - \max(l_a, l_b)}{2 * \max(l_a, l_b)}
\]

C. Implementation Details

C.1. Code

Our code is hosted on GitHub: https://github.com/eriq-augustine/242-2016. Since we have implemented all our methods ourselves, numpy is the only dependency required to run our code.
C.2. Data

To better handle and analyze the Yelp data, we first put it into a relational database. We are using PostgreSQL. Our resulting schema was 12 tables in Boyce-Codd normal form. The create table statements can be found in our repository at data/sql/create.sql.

The script to parse the Yelp data and convert it to SQL insert statements can be found at data/sql/p Ultimate...
<table>
<thead>
<tr>
<th>Feature Set</th>
<th>K</th>
<th>Scalar Normalization</th>
<th>Set Distance</th>
<th>Rand Index</th>
<th>Feature Set</th>
<th>K</th>
<th>Scalar Normalization</th>
<th>Set Distance</th>
<th>Rand Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>NW</td>
<td>18</td>
<td>None</td>
<td>Jaccard</td>
<td>0.669458</td>
<td>NW</td>
<td>18</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.681717</td>
</tr>
<tr>
<td>AW</td>
<td>10</td>
<td>N/A</td>
<td>Dice</td>
<td>0.892555</td>
<td>A</td>
<td>10</td>
<td>N/A</td>
<td>Dice</td>
<td>0.907089</td>
</tr>
<tr>
<td>AW</td>
<td>10</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.855210</td>
<td>A</td>
<td>10</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.969376</td>
</tr>
<tr>
<td>AW</td>
<td>12</td>
<td>N/A</td>
<td>Dice</td>
<td>0.820459</td>
<td>A</td>
<td>12</td>
<td>N/A</td>
<td>Dice</td>
<td>0.898305</td>
</tr>
<tr>
<td>AW</td>
<td>12</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.864035</td>
<td>A</td>
<td>12</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.910638</td>
</tr>
<tr>
<td>AW</td>
<td>14</td>
<td>N/A</td>
<td>Dice</td>
<td>0.851381</td>
<td>A</td>
<td>14</td>
<td>N/A</td>
<td>Dice</td>
<td>0.869441</td>
</tr>
<tr>
<td>AW</td>
<td>14</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.842278</td>
<td>A</td>
<td>14</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.911598</td>
</tr>
<tr>
<td>AW</td>
<td>16</td>
<td>N/A</td>
<td>Dice</td>
<td>0.850255</td>
<td>A</td>
<td>16</td>
<td>N/A</td>
<td>Dice</td>
<td>0.826885</td>
</tr>
<tr>
<td>AW</td>
<td>16</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.844351</td>
<td>A</td>
<td>16</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.907550</td>
</tr>
<tr>
<td>AW</td>
<td>18</td>
<td>N/A</td>
<td>Dice</td>
<td>0.848773</td>
<td>N</td>
<td>8</td>
<td>Log</td>
<td>N/A</td>
<td>0.657850</td>
</tr>
<tr>
<td>NW</td>
<td>18</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.851001</td>
<td>N</td>
<td>8</td>
<td>Logistic</td>
<td>N/A</td>
<td>0.644067</td>
</tr>
<tr>
<td>NA</td>
<td>8</td>
<td>Log</td>
<td>Dice</td>
<td>0.659055</td>
<td>N</td>
<td>8</td>
<td>None</td>
<td>N/A</td>
<td>0.659098</td>
</tr>
<tr>
<td>NA</td>
<td>8</td>
<td>Log</td>
<td>Jaccard</td>
<td>0.658976</td>
<td>N</td>
<td>10</td>
<td>Log</td>
<td>N/A</td>
<td>0.657301</td>
</tr>
<tr>
<td>NA</td>
<td>8</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.742567</td>
<td>N</td>
<td>10</td>
<td>Logistic</td>
<td>N/A</td>
<td>0.649856</td>
</tr>
<tr>
<td>NA</td>
<td>8</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.770920</td>
<td>N</td>
<td>10</td>
<td>None</td>
<td>N/A</td>
<td>0.667856</td>
</tr>
<tr>
<td>NA</td>
<td>10</td>
<td>Log</td>
<td>Dice</td>
<td>0.670728</td>
<td>N</td>
<td>12</td>
<td>Log</td>
<td>N/A</td>
<td>0.669908</td>
</tr>
<tr>
<td>NA</td>
<td>10</td>
<td>Log</td>
<td>Jaccard</td>
<td>0.671137</td>
<td>N</td>
<td>12</td>
<td>Logistic</td>
<td>N/A</td>
<td>0.665688</td>
</tr>
<tr>
<td>NA</td>
<td>10</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.726959</td>
<td>N</td>
<td>12</td>
<td>None</td>
<td>N/A</td>
<td>0.669810</td>
</tr>
<tr>
<td>NA</td>
<td>10</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.737731</td>
<td>N</td>
<td>14</td>
<td>Log</td>
<td>N/A</td>
<td>0.673658</td>
</tr>
<tr>
<td>NA</td>
<td>10</td>
<td>None</td>
<td>Dice</td>
<td>0.667858</td>
<td>N</td>
<td>14</td>
<td>Logistic</td>
<td>N/A</td>
<td>0.661820</td>
</tr>
<tr>
<td>NA</td>
<td>10</td>
<td>None</td>
<td>Jaccard</td>
<td>0.667858</td>
<td>N</td>
<td>14</td>
<td>None</td>
<td>N/A</td>
<td>0.670462</td>
</tr>
<tr>
<td>NA</td>
<td>12</td>
<td>Log</td>
<td>Dice</td>
<td>0.671808</td>
<td>N</td>
<td>16</td>
<td>Log</td>
<td>N/A</td>
<td>0.669040</td>
</tr>
<tr>
<td>NA</td>
<td>12</td>
<td>Log</td>
<td>Jaccard</td>
<td>0.673073</td>
<td>N</td>
<td>16</td>
<td>Logistic</td>
<td>N/A</td>
<td>0.666306</td>
</tr>
<tr>
<td>NA</td>
<td>12</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.723300</td>
<td>N</td>
<td>16</td>
<td>None</td>
<td>N/A</td>
<td>0.670721</td>
</tr>
<tr>
<td>NA</td>
<td>12</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.729915</td>
<td>N</td>
<td>18</td>
<td>Log</td>
<td>N/A</td>
<td>0.669534</td>
</tr>
<tr>
<td>NA</td>
<td>12</td>
<td>None</td>
<td>Dice</td>
<td>0.669829</td>
<td>N</td>
<td>18</td>
<td>Logistic</td>
<td>N/A</td>
<td>0.661045</td>
</tr>
<tr>
<td>NA</td>
<td>12</td>
<td>None</td>
<td>Jaccard</td>
<td>0.669829</td>
<td>N</td>
<td>18</td>
<td>None</td>
<td>N/A</td>
<td>0.669127</td>
</tr>
<tr>
<td>NA</td>
<td>14</td>
<td>Log</td>
<td>Dice</td>
<td>0.667990</td>
<td>W</td>
<td>10</td>
<td>N/A</td>
<td>Dice</td>
<td>0.852379</td>
</tr>
<tr>
<td>NA</td>
<td>14</td>
<td>Log</td>
<td>Jaccard</td>
<td>0.669195</td>
<td>W</td>
<td>10</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.816539</td>
</tr>
<tr>
<td>NA</td>
<td>14</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.714399</td>
<td>W</td>
<td>12</td>
<td>N/A</td>
<td>Dice</td>
<td>0.860230</td>
</tr>
<tr>
<td>NA</td>
<td>14</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.722061</td>
<td>W</td>
<td>12</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.815057</td>
</tr>
<tr>
<td>NA</td>
<td>14</td>
<td>None</td>
<td>Dice</td>
<td>0.670182</td>
<td>W</td>
<td>14</td>
<td>N/A</td>
<td>Dice</td>
<td>0.859987</td>
</tr>
<tr>
<td>NA</td>
<td>14</td>
<td>None</td>
<td>Jaccard</td>
<td>0.670232</td>
<td>W</td>
<td>14</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.814819</td>
</tr>
<tr>
<td>NA</td>
<td>16</td>
<td>Log</td>
<td>Dice</td>
<td>0.678210</td>
<td>W</td>
<td>16</td>
<td>N/A</td>
<td>Dice</td>
<td>0.819333</td>
</tr>
<tr>
<td>NA</td>
<td>16</td>
<td>Log</td>
<td>Jaccard</td>
<td>0.676306</td>
<td>W</td>
<td>16</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.798225</td>
</tr>
<tr>
<td>NA</td>
<td>16</td>
<td>Logistic</td>
<td>Dice</td>
<td>0.712950</td>
<td>W</td>
<td>18</td>
<td>N/A</td>
<td>Dice</td>
<td>0.773908</td>
</tr>
<tr>
<td>NA</td>
<td>16</td>
<td>Logistic</td>
<td>Jaccard</td>
<td>0.720035</td>
<td>W</td>
<td>18</td>
<td>N/A</td>
<td>Jaccard</td>
<td>0.769316</td>
</tr>
</tbody>
</table>

continued...