# Optical Digital Recognition Data set:

In this assignment, the ORC dataset classification problem will be tackled using KNN approach. The main challenge in KNN is to define a best combination of number of neighbors, weight distribution and the distance metric. In this analysis, the following combination will be considered to select the KNN model that fit the data.

|  |  |  |
| --- | --- | --- |
| Weight | Number of Nodes | Metric |
| Uniform, Distance | 1-20 | Manhattan, Euclidean |

Table 1: Different Compination To Study

In this analysis, ‘uniform’ means that all points have the same weight regarding how far from the target point. ‘Distance’ means that the points weight is proportion to the inverse of the distance.

# Feature Engineering

There will be no extra pre-processing for the data as it almost uniformly distributed and any scaling will result in a high dense vector which will increase the computation and decision time. While keeping sparse vectors will cancel out terms when calculating distances which result in short execution time. As in the following example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Normalized | Metric | Weight Distribution | Accuracy | Number of Neighbors | Time(sec) |
| No | Manhattan | Uniform | 100% | 1 | 0.99 |
| Yes | Manhattan | Uniform | 100% | 1 | 6.62 |

Table 2: Time comparison between Normalized vs Non-normalized data

# Analysis

As noted in table 1 different combination will be tested and analysis will be based on the results.

|  |  |
| --- | --- |
| Figure 1: Manhattan, Uniform | Figure 2: Euclidean, Uniform |
| Figure 3: Manhattan, Distance | Figure 4: Euclidean, Distance |

Looking at the accuracy in the figures above, it’s obvious that both metrics Manhattan and Euclidean score the same accuracy and shows same pattern in training and testing set while number of neighbors increases using the same neighbor distribution metric. On the other hand, the weight distribution for the neighbors has significant impact. The uniform distribution scores were less than the distance weight distribution. So regardless the metric used, the accuracy relies on weight distribution for the neighbors. Another point to mention, that all figures with one neighbor achieve 100% accuracy on the training set and that’s due to the fact that the data is well spreaded among different classes. And the drop-in accuracy appears while the number of nodes increases in the case of uniform is that more points from different classes will be considered in voting which reduce the accuracy because all points count the same in uniform distribution.

In terms of execution time, the more neighbors to consider the higher the execution time. For example, using Euclidean distance with inverse proportion of neighbors weight the execution trend will be:

Figure 5: Execution time vs number of neighbors

Given that in practice the test set is not available during the development the decision for which combination to be used will be based on: execution time, accuracy and simplicity with 1% margin of error. So more simple form with lower execution time will be favored on more complex ones with higher execution time among margin of 1% less accuracy. The most interesting results to select are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | Distribution | Accuracy | Number of Neighbors | Execution time (Seconds) |
| Manhattan | Uniform | 1 | 1 | 0.99 |
| Manhattan | Distance | 1 | 1 | 0.90 |

Table 3: Best results among different runs

To validate the results, cross validation conducted on the selected model and achieve accuracy of 0.98%. Testing the model on test set score 97% accuracy.

# Results

|  |  |
| --- | --- |
| Number of Neighbor | 1 |
| Weight Distribution | Inverse proportion with distance |
| Distance Metric | Manhattan |
| Training Time | 0.90 seconds |
| Accuracy on Training Set | 100% |
| Cross Validation | 98% |
| Accuracy on Testing Set | 97% |

# Amazon Baby product review Data set:

The main challenge in the Amazon dataset is to make the process of finding a nearest neighbor fast enough and responsive which is the challenge in any case-based reasoning system. In this experiment, the number of instants (rows) will be reduce along and a set of features will be created to measure the similarity between the source and target data points. As before, analysis for different combination will be conducted. i.e Weight, Number of nodes and distance metric.

# Feature Engineering

The following steps applied to the testing data set rows:

1. The title and product columns combined together
2. Drop stop words
3. Take the stem for each word
4. Combine the text for each target variable value (category) into one row. This operation reduces the number of instances from thousands to 5.
5. Compute the tf-idf score for each row and use the scores as a feature. This will result in a very dense vector with high dimension. Even though, the vector is huge and dense but still the comparison will be conducted between 5 points not thousands of instances represented with highly dimensional sparse vector.

The following steps result in a matrix with 5 rows and 52,061 column (feature).

# Analysis

The following combination tested to measure the performance.

|  |  |  |
| --- | --- | --- |
| Weight | Number of Nodes | Metric |
| Uniform, Distance | 1-5 | Manhattan, Euclidean |

Table 4: DIFFERENT COMPINATION TO STUDY

As noted in table 4 different combination will be tested and analysis will be based on the results.

|  |  |
| --- | --- |
| Figure 6: Manhattan, Uniform | Figure 7: Euclidean, Uniform |
| Figure 8: Manhattan, Distance | Figure 9: Euclidean, Distance |

The results and analysis for this approach was similar to the previous analysis.

“Looking at the accuracy in the figures above, it’s obvious that both metrics Manhattan and Euclidean score the same accuracy and shows same pattern in training and testing set while number of neighbors increases using the same neighbor distribution metric. On the other hand, the weight distribution for the neighbors has significant impact. The uniform distribution scores were less than the distance weight distribution. So regardless the metric used, the accuracy relies on weight distribution for the neighbors. Another point to mention, that all figures with one neighbor achieve 100% accuracy on the training set and that’s due to the fact that the data is well spreaded among different classes. And the drop-in accuracy appears while the number of nodes increases in the case of uniform is that more points from different classes will be considered in voting which reduce the accuracy because all points count the same in uniform distribution.”

In terms of execution time, the more neighbors to consider the higher the execution time. For example, using Euclidean distance with inverse proportion of neighbors weight the execution trend will be:

Figure 10: Execution Time (Euclidean, Distance)

Looking at the results, KNN with Manhattan distance along with weighted pointed based on the inverse proportion for the distance will be used. Note that, the number of cases are small after the reduction so cross validation will not be feasible to conduct in this case.

# Results

|  |  |
| --- | --- |
| Number of Neighbor | 1 |
| Weight Distribution | Inverse proportion with distance |
| Distance Metric | Manhattan |
| Training Time | 18 seconds |
| Accuracy on Training Set | 100% |
| Cross Validation | NA |
| Accuracy on Testing Set | 58% |