Abstract:

Machine learning comes out as an advanced and significant technology, evolving in our daily life. The focus of Machine Learning is to enable computers to learn from the data and make predictions or decisions based on that data, enabling informed decision making without the need of extra instruction or explicit programming for the system to perform task better. The sectors in which machine learning is involved include education, telecommunication, retail, research and development, finance, healthcare, and transportation through data-driven insights. Machine leaning has three basic types: Supervised Leaning, Unsupervised Learning and Reinforcement leaning. In the domain of supervised machine learning, a variety of classifiers abound, including decision trees, support vector machines, Naive Bayes, and K-nearest neighbors (KNN), each tailored to address specific data analysis challenges. Among the numerous machines leaning algorithms, K-Nearest Neighbors (KNN) classifier is simplest and most effective for classification problem. This paper presents the implementation of the k-nearest neighbor (KNN) machine learning algorithm for predicting outcome variables based on input variables. Leveraging the capabilities of Hierarchical Temporal Memory (HTM) for learning complex temporal patterns, our study focuses on predicting types of sequences: even number sequences, odd number sequences, and decimal number sequences. We integrate the KNN model with the Neocortex API to efficiently classify sequences. The KNN model receives input data in the form of Sparse Distributed Representations (SDR) from HTM. We construct a dataset comprising multiple sequence SDRs, each with varying values within a defined threshold. The KNN model processes a stream of sequence SDRs, with the dataset split into 70% training data and 30% testing data. During testing, the model accurately classifies sequences, achieving a 90.9% accuracy rate with some SDR testing data, and consistently predicts matches with 100% accuracy in most cases. The paper discusses the KNN design procedure, challenges encountered, and potential enhancements to further improve model accuracy.

Result: -

In our scenario, we utilize a data splitting method that randomly shuffles the dataset's rows, allocating a specified ratio for training and the remainder for testing. Specifically, 70% of the data is allocated for training, while 30% is reserved for testing. The method returns the training and testing datasets as separate arrays.

The reported accuracy values represent the model's prediction accuracy at various values of k when tested with randomly generated test data. These accuracy percentages provide insights into the performance of our model under different configurations of the k-Nearest Neighbors algorithm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value of K | Random Generated Test Data  Accuracy  in %  (1) | Random Generated Test Data  Accuracy  in %  (1) | Random Generated Test Data  Accuracy  in %  (3) | Random Generated Test Data  Accuracy  in %  (4) | Random Generated Test Data  Accuracy  in %  (5) | Random Generated Test Data  Accuracy  in % (6) | Random Generated Test Data  Accuracy  in %  (7) | Random Generated Test Data  Accuracy  in %  (8) |
| 1 | 100 | 90.9 | 90.9 | 90.9 | 90.9 | 90.9 | 90.9 | 100 |
| 2 | 100 | 100 | 90.9 | 90.9 | 90.9 | 90.9 | 90.9 | 100 |
| 3 | 100 | 100 | 100 | 100 | 100 | 90.9 | 90.9 | 100 |
| 4 | 90.9 | 100 | 100 | 90.9 | 90.9 | 90.9 | 90.9 | 100 |
| 5 | 100 | 100 | 100 | 90.9 | 100 | 90.9 | 90.9 | 100 |

Figure 4.1. Accuracy of the KNN Classifier for different value of K for different testing Data

The analysis presented in Figure 4.1 suggests that the optimal value of k for the k-Nearest Neighbors (k-NN) classifier, when applied to a dataset with three classes, is k=3. Here's a breakdown of the findings:

1. **K=1, 2, and 4**: When the model relies solely on the single nearest neighbor (k=1) or a small number of nearest neighbors (k=2, 4), the accuracy tends to hover around 90.9%. This indicates that the classifier may be prone to misclassifications or inconsistencies when considering a small number of neighbors, leading to accuracy fluctuations.
2. **K=3**: At k=3, the accuracy consistently remains around 100% for most randomly generated testing data splits from the dataset. This suggests that considering three nearest neighbors leads to more stable and reliable predictions, resulting in higher accuracy across various testing scenarios.
3. **K=5**: While the accuracy at k=5 is also reported as 100%, it is noted that there are instances where the accuracy drops to 90.9%. This indicates some variability in performance compared to k=3, where the accuracy remains consistently high.

Based on these observations, it's reasonable to conclude that k=3 appears to be the optimal choice for the k-NN classifier with this dataset.

A computer screen shot of a black screen

Description automatically generated

Figure 4.2.a Predicted Result displayed on output window.

A screenshot of a computer

Description automatically generated

Figure 4.2.b Predicted Result displayed on output window

Figures 4.2.a and 4.2.b display the output window presenting the predicted results. They showcase the nearest neighbors for the test data point with a value of k set to 3, along with the calculated distances and the classification of the class for that specific test data. Subsequently, the predicted class for the test data is indicated, determining whether it is even, odd, or neither based on the voting method. This process repeats for each test data point. Finally, the model's predicted accuracy is displayed at the end.

.A screenshot of a computer

Description automatically generated

Figure 4.4. Unit test For KNN Classifier

Figure 4.4 depicts the unit test conducted on the KNN classifier. A random SDR is selected from the dataset, serving as the test data array to evaluate the classifier's performance. The test ensures that the predicted value by the classifier aligns with the actual class value of the test data. The KNN classifier successfully passes the unit test, as evident from the figure. Additionally, we've incorporated an exception in the unit test to accommodate varying values of K. If the value of K surpasses the length of the SDR data, the test gracefully handles this scenario.