Information Technology Course
Module Software Engineering
by Damir Dobric / Andreas Pech

Investigate and Implement KNN Classifier

1st Ankush Laxman Patil
ankush.patil.stud@fra-uas.de

2nd Ayan Borthakur
ayan.borthakur@stud.fra-uas.de

3rd Nasir Ishaq
Nasir.ishaq@stud.fra-uas.de

Abstract*— Machine learning (ML) has grown over a wide range of sectors such as finance, education, communication, transportation, retail, and healthcare. In response to this, researchers have developed a variety of algorithms to analyze a large amount of data and derive quantifiable insights from that data. Subsequently, ML has a different range of algorithms to support text, image, audio, video, and numeric formats, and based on the application ML can predict the outcomes. Supervised learning, which is a type of machine learning, uses classification and regression to analyze the data. In general, there are several types of classifiers present under supervised machine learning, such as decision trees, support vector machines, naive Bayes, and K-nearest neighbors (KNN) designed under this section. In this paper, we have proposed the implementation of the most widely used k-nearest neighbor (KNN) machine learning algorithm used to predict the outcome variable or dependent variable based on the input variables or independent variables. KNN classifier can be used with HTM by incorporating it as a sub-module for classification tasks. Precisely, HTM models can learn to represent the input data in a high-dimensional feature space and then use the KNN classifier to classify the data based on the closest neighbors in this feature space. This can be useful in situations where the input data has a complex temporal structure and requires a more sophisticated approach to classification than simple threshold-based methods. So, the developed kNN model is also integrated with the Neocortex API in order to get the input dataset from Hierarchical temporal memory (HTM). The designed KNN model gets a stream of sequences as input from HTM and then it classifies whether the predicated sequence has a match or mismatch with the input data sequence with 80% accuracy. Additionally, the design procedure, challenges, and enhancements to improve model accuracy are discussed in the paper.

Keywords—Machine Learning; Supervised Learning, K-Nearest Neighbors; Neocortex API; Hierarchical temporal memory.

I. INTRODUCTION

The aim of machine learning is to create statistical models that allow computers to learn from data without being to be explicitly programmed. More precisely, these statistical models and algorithms generate the "learn" method, which helps the computer in predictions about the future based on the trained data. It is also observed that some of the algorithms predict with better accuracy in the specific type of dataset. Supervised, unsupervised, and reinforcement learning are the three main types of machine learning based on the learning patterns of the model.

The supervised machine learning algorithms learn from input variables and a pre-labeled dataset to predict an output variable. supervised learning can be used for a wide variety of applications, such as image and speech rec. cognition, natural language processing, fraud detection, and recommendation systems based on it regression and classifiers algorithm.

A. Regression

Regression is a type of supervised learning which predicts a continuous numerical output for a given input data point. The input data is a set of features or attributes, but the output is a continuous value rather than a discrete class label. In regression, the algorithm learns from labeled training data to predict the value of a target variable based on the values of other variables. The best example of a regression algorithm could be trained to predict the price of a house based on features such as the number of bedrooms, location, age, and area.

B. Classification

Classification is also a type of supervised learning where the goal is to predict a discrete class label for a given input data point. In classification, the algorithm learns from labeled training data to classify new, unseen data points into one of the predefined classes. For example, a classification algorithm could be trained to classify if the person is diabetic or non-diabetic based on certain input features. The algorithm would learn from a set of labeled characteristics of a large dataset and their corresponding classifications, and then it could predict whether an unknown person is diabetic or not.

In this paper, we have specifically focused on designing the k-nearest neighbor (*KNN*) classifier which comes under the supervised machine learning algorithm. It is mainly used for pattern recognition and classification problems. The basic idea is to develop the *KNN* classifier algorithm as a prototype and then integrate it with the Neocortex *API*. The focus of Neocortex API is to implement Hierarchical Temporal Memory Cortical Learning Algorithm. HTM is a type of machine learning model that is inspired by the structure and function of the neocortex in the brain. HTM models are designed to learn and recognize patterns in time-varying data, such as audio, video, and sensor data. So, the HTM will act as an input to the KNN classifier and this integration of KNN with Neocortix API can be useful in situations where the input data has a complex temporal structure and requires a more sophisticated approach to classification than traditional threshold-based methods.

II. Literature Review

In this section, we will understand the KNN in detail and trace the design procedure of different researchers. The selection of K is an important parameter of the algorithm, whose value determines the complexity and accuracy of the classifier. The goal is to understand KNN from the perspective of different researchers which will eventually help in the model-designing procedure.

The study [2] proposes Zhang et al, an adaptive KNN algorithm based on the density of data points. The algorithm dynamically adjusts the value of K based on the density of data points in the local region. They showed that the adaptive KNN algorithm outperformed the traditional KNN algorithm on several benchmark datasets.

A weighted KNN algorithm that assigns different weights to the K nearest neighbors based on their distance from the query point. The weights are calculated using a Gaussian kernel function, and the classifier is trained using a cross-validation technique. They showed that the weighted KNN algorithm outperformed the traditional KNN algorithm on several benchmark datasets [3].

Liu et al [4] propose a hybrid KNN algorithm that combines KNN with the random subspace method. The algorithm randomly selects a subset of features and applies KNN to the reduced feature space. The process is repeated several times, and the results are combined using an ensemble technique. They showed that the hybrid KNN algorithm outperformed both the traditional KNN algorithm and the random subspace method on several benchmark datasets.

The study [8] explains the survey paper and explains the important definitions, principles, and methodologies of the supervised machine learning algorithm.

Reddy, and Ravi Babu [9], explain in the paper that supervised machine learning is mainly used for predictive modeling, data mining technique, image classification, etc.

In today's world, many newspapers provide us with a variety of information that can be classified into many categories such as sports, politics, world, technology, and so on. Text classification has become very important nowadays as large-scale data classification has become a problem for everyone everywhere. Convolutional Neural Networks (CNN) are a very important approach for Deep Learning, which is currently proving to be a strong competitor for other classification algorithms such as decision trees and K-NN [11].

In summary, the literature suggests that KNN can be improved by adjusting the value of K based on the density of data points, assigning different weights to K-nearest neighbors based on their distance, and combining KNN with other algorithms using ensemble techniques. These techniques can improve the accuracy and robustness of the KNN classifier for various classification problems.

III. Theoretical background KNN and HTM

A. K-Nearest Neighbors Parameters and Design Impact

The k-nearest-neighbor classification algorithm was first used in an unpublished US Air Force School of Aviation Proceedings of the International Conference on Intelligent Computing and Control Systems to execute characteristic analysis when clear parametric approximations of probability densities were unknown or difficult to determine[2]. In the KNN classifier algorithm, there are several parameters that plays important role in algorithm designs, and those parameters are discussed below,

1. Distance calculations

The KNN algorithm works by finding the k-nearest neighbors to a new input sample in the training dataset based on a distance metric. The distance metric can be Euclidean distance, Manhattan distance, or any other appropriate distance measure. Once the k-nearest neighbors are identified, the algorithm assigns the new sample to the class that is most common among its k-nearest neighbors.[7] The most common distance calculations techniques used in kNN are described below:

A. Euclidean Distance: Euclidean distance is the straight-line distance between two points in a Euclidean space. In other words, it is the distance between two points in a 2D or 3D space. The Euclidean distance between two points (a1, b1) and (a2, b2) can be calculated as follows:

$$distance = \sqrt{(a_2 - a_1)^2 + (b_2 - b_1)^2}$$

B. Manhattan Distance: Manhattan distance is the distance between two points measured along the axes at right angles. It is named after the grid-like layout of the streets in Manhattan. The Manhattan distance between two points (a1, b1) and (a2, b2) can be calculated as follows:

$$distance = |a_2 - a_1| + |b_2 + b_1|$$

2. K value selection:

The value of K is a hyperparameter that needs to be selected before making predictions. The value of K controls the number of neighbors that are used to make predictions. A larger value of K results in a smoother decision boundary but may result in more misclassifications of the training data. A smaller value of K results in a more complex decision boundary but may result in overfitting the training data. The choice of K is typically made using a validation set or cross-validation. The algorithm is trained on a portion of the data, and the remaining data is used to validate the performance of the algorithm for different values of K. The value of K that results in the highest validation accuracy is selected as the final value of K.

3. Voting principle:

In the KNN algorithm, the prediction for a new observation is made based on the class labels of its K nearest neighbors. The class with the most number of occurrences among the K nearest neighbors is selected as the predicted class for the new observation. This is known as the majority voting principle[13].

The majority voting principle is simple yet effective and has been shown to perform well in many applications. However, it is important to note that the choice of K can affect the voting outcome, and a larger value of K may result in a less confident prediction. Additionally, in the case of imbalanced data, where one class has significantly more instances than the other, the majority voting principle may result in biased predictions. In such cases, techniques such as weighted voting or distance-weighted voting can be used to balance the influence of each class[10].

Apart from these parameters Feature selection and Preprocessing techniques such as normalization, standardization, and scaling can be considered while designing the model and increasing its accuracy of the model.

B. Hierarchical temporal memory

The HTM algorithm is designed to learn and recognize the temporal patterns in the data. It builds the hierarchy of processing layers. In this process, each layer consists of a set of interconnected neurons. There are two types so layers in this hierarchy where the lower layers of the hierarchy learn to recognize simple patterns in the data and the higher layers learn to recognize more complex and abstract patterns. This HTM algorithm is based on the concepts of “sparsity" and this makes this algorithm unique. The sparsity means that only a small subset of neurons in each layer are active at a given time. Also due to this, it is very efficient in memory management and computation of any parameter related to the data. [8]

HTM has a variety of applications such as prediction, classification, and anomaly detection. So In this paper, the HTM algorithm is used as an input to the KNN Classifier. The KNN classifier processes the data and predicts the unlabeled data class as an output [11].

IV. Model Explanation

In this section, the design and working methodology of the KNN classifier is explained in detail with the help of examples. As described in Figure. 1 the categorical data is plotted on X and Y axis and data has two primary categories i.e. Category A and Category B. The goal is to predict the new data points given in the blue column which do not belong to any category. This is a classic example of a classification-based problem and in order to predict the outcome we have to follow the below-mentioned steps,

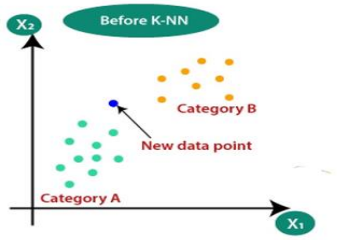

Figure. 1: Plotted KNN Data for prediction

Step-1: Select the number K of the neighbors for computation and it is recommended that select the odd value of K for better predictions. In this case, let's choose K=5. so, the model will select the 5 nearest neighbors.

Step-2: In the second step, the algorithm will calculate the Euclidean distance as shown in Figure 2 from all the 5 nearest neighbors from the new data points and store the distance in the distance table.

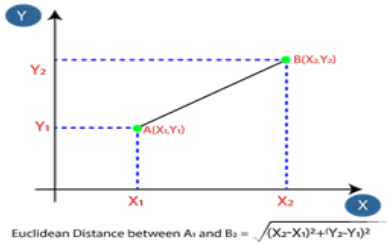

Figure. 2: Euclidean Distance Calculation

Step-3: The distance table will have all the values along with their known category label.

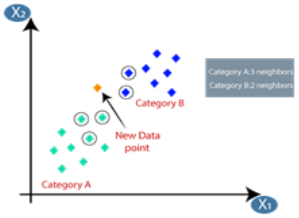

Figure. 3: Euclidean Distance Calculation

Step 4: In step 4, As Figure 3 shows the voting principle will calculate the vote for each category and then the highest number of vote categories is assigned to new data points. For example, the Category of the New Data Point is A and the final outcome is given in Figure.4 below,

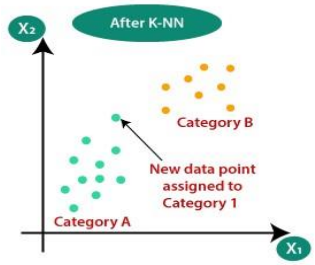

Figure. 4: Final KNN Classifier Outcome

As discussed in the above example the KNN Classifier is an effective machine-learning algorithm for classification problems and works well with all kinds of categorical datasets.

V. Model Construction and Classification Process

A. Base model construction

The base KNN model is developed to understand the basic functionality and it has three main methods that are Distance, Vote, and Classify. This is designed to predict the outcome for a static dataset. The implementation of the prototype is given as a Prototype.

The Distance method is used to calculate the distance of a new data point that has unlabeled data. The vote is used to prepare the voting table and from the distance matrix and the Classify method is used to classify the class of the unknown label data as an output.

B. Integration of Model with Neocortex API

From the base model, the new KNN model is designed in integration with the Neocortex API. In this combination, Neocortex API will provide the stream of sequences as input and KNN will predict the predefined labels for the output data sequence. Later, the unit test cases are implemented to analyze the Integrity of the model. Also, NuPIC's kNN Classifier implementation used for the Sparsely Distributed Represented dataset has influenced the development of this integrated KNN model. As discussed in the theoretical background these algorithms compute the distance computation and voting calculation, therefore, Class-based generic containers are used to design the algorithm. These containers store the distance, classification parameters, and SDR number of classifications. For Example, If a classification contains 2 SDR's the SDR number shall store 1 for the first and 2 for the latter.

In order to meet the base requirement of the Neocortex API, two mandatory methods from the **IClassifier** are implemented. The first method helps the model to learn the dataset and the second returns the N values chosen by the classifier. Both methods are described below in detail.

1. Learn: public void Learn(TIN input, Cell[] cells)

This method takes labels and the sequence of cells which have already been labeled. The new entry is inserted into the _model variable which is a dictionary mapping of Dictionary<string, List<int[]>>, this keeps multiple representations of these sequences called SDRs as the learning process continues. There is an upper threshold of values each classification can hold. This is controlled by the _sdr value. The SDR values are dynamically adjusted if the threshold is exceeded.

2. Assigning the value of K

Public List<ClassifierResult<TIN>>
GetPredictedInputValues(Cell[] unclassifiedCells, short howMany

The implemented GetPredictedInputvalues method returns the N values that have been chosen by the classifier in the format of the ClassifierResult<TIN>.

GetPredictedInputValues

This method is the heart of the prediction process. It runs through all the sequences saved in the model and compares it with the inserted sequence to be classified. It does this by comparing the distance between the values in the sequences and considering the least difference between individual values. Which then outputs a Dictionary holding the value of the unclassified sequence as the keys and the relative distances.

```
{
 "23": [0, 1, 2, …],
 "36": [0, 2, 8, …],
 …
}
```

Also, the conversion of the data to Dictionary<int, List<ClassificationAndDistance>, the Keys being a value of the unclassified sequence and the Values being a List of ClassificationAndDistance objects as showed below,

```
{
 "23": ["<object Classification & Distance 1>", "<object Classification & Distance 2>", …],
 "36": ["<object Classification & Distance 1>", "<object Classification & Distance 2>", …],
 …
}
```

The object representation of ClassificationAndDistance which stores the classification information and the distance for the voting process looks similar to the code snippet below,

```
public class
ClassificationAndDistance:IComparable<ClassificationAnd
Distance>
{
 public string Classification { get; }
 public int Distance { get; }
 public int ClassificationNo { get; }
} …
```

This implementation is used because it simplifies the voting process as it holds the label information and the distances compared to each value of the unlabeled sequence.

The Voting is done using two main comparison types Votes and Overlaps. The resultant mapped dictionary Dictionary<string, int>; key holds the label information, and the int holds the occurrence count of label data .There might be some instances where the sequences are close to one another without any overlaps and there are cases where multiple overlaps occur but the distances of some elements in the sequence could be very large. Hence, we consider these two parameters essential. Thereafter these Comparison variables are ordered according to the integer value from the highest to the lowest. Priority is given to the overlaps if it happens to have 50% overlaps in the sequence otherwise the voting is considered.

Also, the important thing is the votes counted are in the length of the number of Classification present & the overlaps counted are in the length of the total number of SDRs (_sdr) under all the classifications.

C. Unit Testing:

In order to test the designed KNN algorithm the test case is implemented by taking the reference of HTM Classifier of Neocortex API. In the unit testing the model showed promising results with 100% accuracy mismatching 1 value at most when ran under the Multisequence learning experiment.

VI. RESULTS

As described in the previous section KNN model gets the stream of data sequences to predict the outcome. The below figure shows the output window and the KNN classifier is classifying whether the output sequence has a match or

mismatch with the input sequence. The model gives an accuracy of 100% in most of the input data sequences. However, further tuning the model is required to make the model reliable.

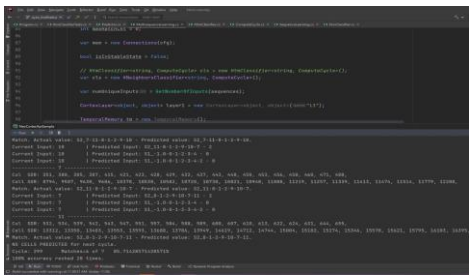

Figure. 5: KNN Classification Output

The expected results are seen for the Unit Testing of the model and the reference of the same is shown in Figure. 6. Unit tests have also been implemented to handle special cases. All the unit tests that have been implemented for the HTM Classifier were implemented for this classifier, and in the testing phase of the model we have not observed any occurrence of failed tests. As the development of this continues many other special Conditions need to be considered.

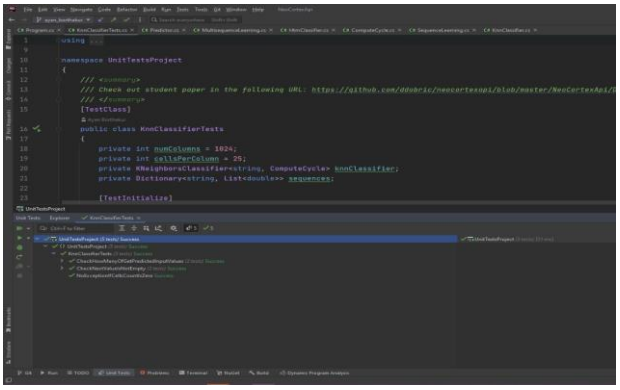

Figure. 6: Unit Testing Of The KNN Algorithm

VII. Desigen Challenges and Enhancement

In this section, the KNN model designing challenges and some of the improvements that can be made to get the in care the accuracy of the model are discussed.

A. Design Challenges

1. Selection of K-Value: The selection of the K value is based on the KNN algorithm so in order model needs to evaluate by selecting the different values of K and then it needs to be cross-validation for better performance or reliable accuracy.

2. Distance Metric: In the KNN algorithm different distance calculation matrices can be used to calculate the distance such as Euclidean distance, Manhattan distance, Minkowski distance, and cosine distance. So again, we have to try this matrix to build a stable model.

B. Methods to Enhance Accuracy

There are standing processes that help machine learning models to improve their performance such as normalization, standardization, and scaling. Apart from this, the below two methods can help the KNN algorithm to enhance its performance.

1. Algorithm Variants: Implementation of KNN algorithm variants of the KNN algorithm such as weighted KNN, instance-based KNN, and kernel-based KNN can help to evaluate their performance on your dataset and get better predictions.

2. Ensemble Techniques: Combination of multiple KNN models using ensemble techniques such as Bagging or Boosting to improve the accuracy of the model.

VIII. Conclusion

We started with the designing of the KNN prototype which was more generic and got the desired outcomes. Then based on the same model the KNN model is developed and then it is integrated with the Neocortex API. The model is successfully integrated and gets the stream of data sequences to predict the outcome. KNN classifier is classifying the output sequence as a match or mismatch with the input sequence. The model gives an accuracy of 100% in most of the input data sequences. Unit test is implemented to handle special cases. All the unit tests that have been implemented with reference to HTM Classifier and satisfactory results are achieved.

IX. References

[1] A. A. A. Soofi, "Classification Techniques in Machine Learning: Application and Issues," International Journal of Basic & Applied Sciences, Vols. vol. 4, no. 13, pp. pp. 459-465, 2017.

[2] R. B. S. Aiman Moldagulova, "Using KNN Algorithm for Classification of Textual Documents," in 8th International Conference on Information Technology (ICIT) , 2017.

[3] D. Dobric, ""Influence of input sparsity to Hierarchical Temporal Memory Spatial Pooler Noise Robustness," 2019.

[4] J. Z. A. L. J. W. Q. &. L. S. Zhang, "An adaptive KNN algorithm based on data point density," IEEE Access, 7, 80876-80883. doi: 10.1109/ACCESS.2019.2926611, 2019.

[5] X. L. Y. &. C. Z. Yu, "Weighted K-nearest neighbor algorithm based on Gaussian kernel function," Journal of Ambient Intelligence and Humanized Computing, 11(5), 1985-1993. doi: 10.1007/s12652-019-01633-9, 2020.

[6] S. A. a. J. H. Y. Cui, "The HTM Spatial Pooler__ A Neocortical Algorithm for Online Sparse Distributed Coding," Front. Comput. Neurosci, vol. vol. 11, p. 111, 2017.

[7] S. A. a. J. H. Y. Cui, "Continuous Online Sequence Learning with an Unsupervised Neural Network Model," Neural Comut.,, vol. 28, pp. 2474-2504, 2016.

[8] Y. H. D. C. W. Xiaoyang Liu, "Memristor-Based HTM Spatial Pooler with On-Device Learning for Pattern Recognition," IEEE, vol. 52. NO. 3, 2022.

[9] D. N. Sang Nguyen, "Investigation of Hierarchical Temporal Memory Spatial Pooler's Noise Robustness against Gaussian Noise," Information Technology Course .

[10] M. B. Sadegh Bafandeh Imandoust, "Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoratical Background," S B Imandoust et al. Int. Journal of Engineering Research and Applications, vol. 3, pp. 605-610, 2013.

[11] F. H. N. Burkart, "A Survey on the Explain ability of Supervised Machine Learning," Journal of Artificial Intelligence Research, vol. vol 70, pp. pp. 245-317, 2019.

[12] X. X. H. G. Y. &. Y. X. Liu, "A hybrid k-nearest neighbor algorithm based on random subspace," IEEE Access, 9, 28792-28799. doi: 10.1109/ACCESS.2021.3052101., 2021.

[13] S. D. S. V. A. S. Kashvi Taunk, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," in Proceedings of the International Conference on Intelligent Computing and Control Systems, 2019.

[14] R. B. K.V. Reddy, "A Review on Classification Techniques in Machine Learning," International Journal of Advance Research in Science and Engineering, Vols. 7, no 3, pp. pp. 40-47, 2018.

[15] J. B. J. Sreedevi, "Newspaper Article Classification using MachineLearning Techniques," International Journal of Innovative Technology and Exploring Engineering, Vols. vol. 9, no. 5, pp. pp. 872-877, 2019.

[16] S. A. a. J. Hawkins, "Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory," p. Available: http://arxiv.org/abs/1503.07469, 2015.

[17] J. H. a. D. George, "“Hierarchical Temporal Memory Concepts, Theory and Terminology," Numenta, Redwood City, CA, USA, p. Available: http://www.mlanctot.info/files/papers/Numenta_HTM_Concepts.pdf, 2006.

[18] G. K. V. K. Eui-Hong (Sam) Han, "Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification," Department of Computer Science and Engineering University of Minnesota, USA, Minnesota, 1999.