Implement KNN Classifier for Sequence Prediction

Dr.Muhammad Haris
Information Technology (M.Eng.)
Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
Email: here

Zaka Ahmed
Information Technology (M.Eng.)
Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
zaka.ahmed@stud.fra-uas.de

Abstract—**Machine Learning (ML) has become ubiquitous in various industries and evolving in our daily life. The sectors in which machine learning is involved include education, Telecommunication, retail, research and development, finance, healthcare and transportation. As, the development is going on various algorithms are used for prediction of text, images, videos and pictures.**

There are various classifier aviliable including support vector machines, decision trees, naive Bayes, and K-Nearest Neighbours for data analysis. In this paper, our main focus is on KNN Implementation for sequence prediction with Hierarchical Temporal Memory (HTM). Supervised learning enables the prediction of output variables based on the input. Hierarchical Temporal Memory offers capabilities to learn complex temporal patterns, making it well-suited for processing sequential data. The Neocortex API helps in the integration of KNN model with HTM, which enables efficient classification of the sequences.

We are using three inputs odd numbers, even numbers or neither odd or even. These sequences are encoded and mapped on the cells which gives us SDR values. These SDR active cells are our data inputs. Which are further disguinsh by 0.3 in testing data and 0.7 in training data. Meaning splitting the dataset into two parts. Where 80% belongs to training data and other 20% to test data. Remember Training and testing data is labeled but while testing last index is not considered. In the further steps, using the Input SDR we will find the distance matrix between the training and test data. And then using K and voting method. We decide whether the unlabed data is belongs to that specific class. In KNN classification, the object is classified based on the votes of its nearest neighbors. If k=1, then the object will be in the class of that single nearest neighbor.The simple function of the kNN model is to predict the target class label. This is further explained in the paper.

The results achieved has shown a 80% accuracy rate in classifying the input sequences. In this paper we covers the introduction and important parameters of K-Nearest, Distance Matrics, KNN impelementation, and the challenges regarding KNN.

Keywords— Machine Learning, Hierarchical Temporal Memory, K-Nearest Neighbors, Sequence Classification, Integration, Neocortex API, Accuracy Enhancement

I. INTRODUCTION

A. Background

KNN was first developed by Joseph Hodges and Evelyn Fix in the year 1951[1], in statistics the concept of k-nearest neighbors algorithm(k-NN) is involved in the non-parametric supervised learning method. Further developments in KNN are proceeded by Thomas Cover.[2] KNN is commonly used for regression and classification. Both in regression and classification the input consists of k-closest training examples in the data set. Remember the output depends on whether the use case is either regression or classification of K-NN.

B. Regression

The main difference between classification and regression is that in regression, the output is the property value for the object. The value is the total average of the neighbor's nearest values. If k=1, the output is assigned from that particular single nearest neighbor.

C. Classification

The main variation in the output of classifier and regression is that in classification, the output is the class membership. In classification, the object is classified based on the votes of its nearest neighbors. If k = 1, then the object will be in the class of that single nearest neighbor. The simple function of the kNN model is to predict the target class label. In other words, the class label is often described as a majority voting. The most common terms are technically considered "plurality voting" and "majority vote" The term "majority voting" means the majority needs to be greater the 50% for making decisions. The classification problems with only two classes, like binary predictions, there is always a majority. A majority vote is also automatically a plurality vote. We don't require multi-class settings to make predictions via kNN in multi-class settings.

II. METHODS

This section includes multiple subsections including description of KNN in detail, second section try to focus on Theoretical background of including how the values determined accuracy and complexity of the classifier. In the last section of this section is regarding Hierarchical temporal Memory(HTM).

A. Literature Review

The K-Nearest Neighbors (KNN) algorithm is a widely-used non-parametric classification method in machine learning. KNN depend on instance based learning where prediction is made based on the similarity of new data points to existing labeled data points. In finance, KNN is used for credit scoring, stock analysis and fraud detection. In healthcare sector, KNN is used for patient health monitoring and drug suggestions. If, I will talk about marketing professionally KNN is used as product suggestion, market trend analysis and customer ads suggestions. In most cases KNN is used for determining the specific patterns and tasks like face recognition or image classification.

Researchers have proposed various enhancements to the traditional KNN algorithm which can improve the scalability and performance. The weight adjustment algorithm is proposed by Han EH., proposed assigning weights to the nearest neighbors based on their distance from the respective point.[11][8]. The assigned weights desgunish, how much the weights influence the classification method. In this way, high weights will be assigned to the one who are closer neigbors,

so it give more priority to the similar instances while performing classification. This technique is usefull where the dataset has many feature, some of which can be considered as unneccessary but it has high cost in context of computational cost.

Zhang et al. has proposed to adjust the value of K based on the local density of data points [3]. This adaptive KNN algorithm dynamically selects the optimal value of K for each query point, leading to more robust predictions. And shows that the adaptive algorithm outperforms many other traditional KNN algorithms. The other approach can be locally adoptive KNN algorithm. It choices the optimal value of K for classifying an input by analyzing the outcomes of cross-validation computations within the local neighborhood of the unlabeled data point.[12] The approach defined by Song Yang, is to introduce two input parameters. As we know determining the correct value of K depends on the characteristics of the dataset, the selection of correct parmeter for various applications is a challenge. Song Yang suggest to introduce a novel metric that assesses the informativeness of objects to be classified, with informativeness quantifying the significance of data points. Two parameter will be K and I as a input. The class is determined based on the majority class of the most informative training examples.[10]

Whereas on the some has proposed to integrated KNN with dimensionality reduction methods such as principal component analysis (PCA) to improve computational efficiency [5]. This a combination is a example of hybrid approach. Whereas other have combined KNN with ensemble methods such as random forests to enhance predictive accuracy [6]. These hybrid approaches enhance the performance as compared to standalone KNN.

B. K-Nearest Neighbors Parameters and Matrix

The K-nearest was first used by US Air force to execute characteristics analysis. There are differnet paramenters in KNN classsifier which plays a important role in algorithm designs including distance matrics, K-Value selection, and voting.

1) Computing Distance Matrix: If we try to recap, the main objective of the k-nearest neighbor algorithm is the identification of the nearest neighbors of a given input point. So, What we can do is assign a class label to that specific point. The first thing is determining the distance metrics. In order to find which class(data point) is nearest to the input data, to do so we will calculate the distance between the data points and query point. We get assistance to decide in which regions the input point belongs. The distance metrics can be either Manhattan distance or any other approach. The first thing is to identify the k-nearest neighbors and then amount of its k-neatest neighbors.The most famous techniques are discussed below:

a) Euclidean distance (p=2): In the early 300 B.C.E, the Greek mathematician introduced Euclid while finding the difference between distance and angle. Still Euclid is most common used of distance. From that early, till yet Euclid is widely and applies in two or three dimensions space. The main method, is the root of square distances between two coordinates of a pair of objects. Then there is a square root of the sum of squares of the differences between the corresponding values.

As from the fig.1 we have the $(x_1, y_1), and (x_2, y_2)$ from we can see that $(x_1, x_2), and (y_1, y_2)$ are allocated into the two dimensional space. And If we try to build a right angled triangle and draw a hypotenuse straight line between two points(d). The other two sides of a right angle triangle are base and altitude which will be $|x_1 - y_1|, and |x_1 - y_1|$. So there the hypotenuse(d) is our Euclidean distance which is the between (x1, x2) and (y1, y2). As this is only a straight line so we will use Pythagorean theorem. The distance between $(x_1, x_2), and (y_1, y_2)$ would be $(x_1, y_1)^2, and (x_2, y_2)^2$. Taking this further, to calculate the Euclidean distance between, $x_n, and y_n$ the Euclidean distance is $(x_n - y_n)^2$ in n-dimensional space.
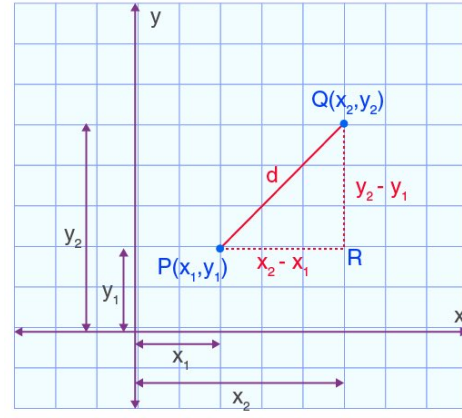

Fig. 1. Euclidean Distance

$$Euclidean and distance = d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

- y_i are the coordinates of one point.
- x_i are the coordinates of another point.
- d is the distance between $(x_1, y_1), and, (x_2, y_2)$

Euclidean is limited to real-valued vectors and is most commonly used for distance measures. The above expression is used to determine the straight line between the input point and the other point being measured.

Euclidean distance is the shortest distance between two points means the straight line between source and destination whereas the Manhattan distance is the sum of all the real distances between the source and destination. And each always be a straight line. The reason behind Euclidean distance used in KNN is that it provides a simple and intuitive measure of similarity between data points in a multi-dimensional space. One of the main reason is that Euclidean distance in KNN, is its computational efficiency and simplicity. Euclidean distance metric works well for continues features and is suitable for data represented in a high-dimensional space.

Euclidean Distance is largely used in machine learning learning algorithms such as K-Nearest(KNN) where it is used to measure the similarity between two data points.Spatial Analysis for measuring distance between geographical locations. As well as, in robotics for obstacle avoidance and path planning. It is also used in image process for comparing images based on pixel values. Due to coordinate system, making it robust to transformations of the data. It is most widely adopted approach in machine learning applications.

b) Manhattan distance (p=1): In Manhattan distance, the absolute value is measured between two points. Manhattan is widely used as for resolving the problems related to geometry. We can say it as a ordinary distance between two points. Manhattan distance is also one of the popular and dominant distance metrics. The most common example is Uber app visualization with a grid and navigation via streets from one address to another illustration. Manhattan is widely used in cluster analysis. K-Means clustering algorithm is the common example where Manhattan distance is used. The second name of Manhattan Distance is city Block Distance. The simplest way to calculate the distance is to go horizontally and then vertically until you get from one point to the other between two points and go vertically instead of going in a straight line.

The Manhattan Distance or City Block Distance determines the absolute difference among the pair of the coordinates. It might be surprising, but the simplest way of calculating the distance between two points is to go horizontally and then vertically until you get from one point to the other, instead of just going in a straight line. It is
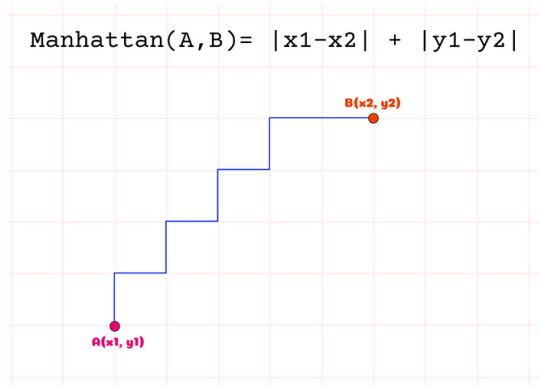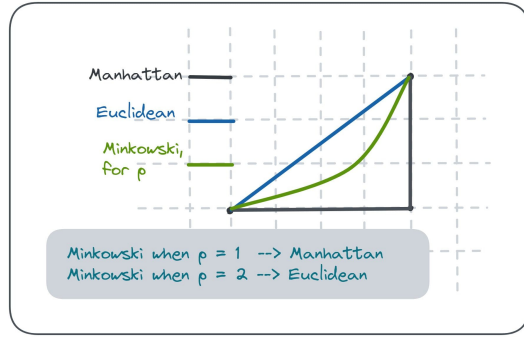
Fig. 2. Manhattan Distance

Fig. 3. Difference Between Distance Methods

Fig. 4. Process Diagram

a simpler technique since it only needs you to subtract instead of performing more complicated calculations.

$Manhattan distance = d(x,y) = \sqrt{(\sum_{i=1}^{m} |X_i - Y_i|)}$

As shown, it is the root of squared difference between the coordinates of two objects. We only need to subtract two points instead of performing complex tasks.

*c) **Minkowski distance**:* It is the generalized form of Manhattan and Euclidean distance matrics. Manhattan distance is denoted by p equal to one whereas the Euclidean distance is represented by p equal to two. Parameter p allows the creation of other distance metrics as shown below.

$Minkowski distance = \sqrt{(\sum_{i=1}^{n} |X_i - Y_i|)^{1}/p}$

*d) **Hamming distance**:* This technique used with string vectors or boolean identifying the points where the vectors do not match. Overlap metrics are also referred as represented below:

$Hamming Distance = D_H = (\sum_{i=1}^{k} |X_i - Y_i|)$

2) Defining k selection: In simple words, k is the number of neighbors used for making prediction. The k value indicates how many neighbors will be compared or we can say checked to determine the resultant class in KNN Algorithm. For-example: By changing the value of k the classification can lead to under-fitting or over-fitting. If k=1, the instance will be assigned to the same class because we have a single neighbor. By using lower value of k can low bias, but high variance. As well as, larger value of k may lead to lower variance and high bias. So, we can define k as the balancing act as different values impact the variance on under-fitting or over-fitting. For avoiding ties in classification, k is recommended to be a odd number. The best approach to get a optimal k for you dataset is cross-validation tactics.
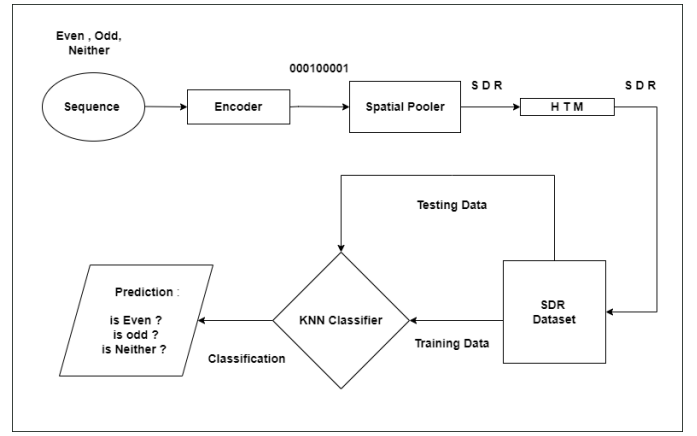
3) Voting Prinicple: In the KNN algorithm, the prediction is based on the number of K nearest neighbours is considered. The predicted or winning class is determined using the majority voting principle, the class which has high numbers of vote is the class from where our element belongs to. The highest frequency among the K nearest neighbors is chosen. However, the important thing that it is essential to recognize that the choice of K can impact the voting outcome. The higher value of K may leads to less confident predictions. The scenarios where we have imbalanced data, where one class is higher or outweighs the other class, the majority voting principle can be biased. To resolve such kind of issue techniques like distance-weighted voting and weighted voting us introduced which can help to balance the influence of each class.

Mathematically, if $(C_1, C_2, C_3, ...C_n)$ represent the unique class labels among the K nearest neighbors, and $Count(C_i)$ represents the count of occurrences of class C_i the predicted class y for the new observation is determined by:

$y = argmax Count(C_i)$

Where y is the predicted class Label.This voting equation ensures that the predicted class is the one that is most prevalent among the K nearest neighbors.Techniques like normalization, standardization, and scaling play a crucial role in preparing the data for modeling in pre-processing. Normalization involves scaling the features to a similar range, typically between 0 and 1, the main advantage of normalization is that it prevents larger scales from dominating the model. Scaling adjusts the range of features to a desired range, which can enhance the performance of algorithms sensitive to feature magnitudes.

As the process diagram (Fig:4) shown the step-by-step procedure, we have adopted in the implementation. In the first step, we feed the input sequences from the text file (3.1) and that specific sequence get encoded. Then after spatial pooler we get sequence SDRs. Then it will be stored in HTM. The next step will be implementation of KNN classifier in which the test data will be mapped, the KNN helps to predict the class from where the sequence blogs to that specific class. The sudo-code of voting is also discussed in the implementation part of paper.

C. Overview of HTM

As the objective behind the HTM CLA is to make a progress towards making a program that functions cognitive tasks as like a simple human being brain. The prediction is done by making a system which can memorize as well as learn from the information executions which are fed before. HTM to anticipate and memorize, it requires user Input.

As the overall HTM have multiple sections which includes data, Encoder, HTM spatial Pooler, HTM temporal Memory, and HTM

Classifier. The data or which is also know as input is a scalar value, data or time, or a picture. Then the next element is encoder which is responsible for changing the data into SDR which can further be used with HTM classifier. SDR is in the cluster of binary input either '0' or '1'. As discussed earlier, input of encoder can be anything a scalar value. It includes locations, weeks, months, time or days in a week etc.

The next part of the HTM is a spatial pooler it, is an algorithm which learns spatial patterns. The spatial pooler gets an input of bits cluster and converts it into SDR. Next Parts is Temporal memory, it a part which learns the arrangements of SDRs shaped by the spatial pooler algorithm. [5]

D. Spatial Pooler

Spatial Poolar is the second phase of HTM learning. It uses the output of the encoder to learn the SDR of the given input binary array. The idea for the spatial poolar SDR is to generate SDRs of the input which is the output of the encoder. Once the input SDR are learned, if the same input is given again, it tries to match already learned SDRs and then generates a similar matching SDR.In this method, it will disgunish, is it a same input which is already memorized or a different one.

Fig. 5. General Flow

III. IMPLEMENTATION DETAILS

The KNN method is divided into two classes first and main class is classifier which is consist of distnace, vote and classify. Second class for indexanddistance. Distance method which is used to calculate the distance of a new data point that has unlabeled data. The voting method is used to prepare the voting table from the distance matrix and the Classify method is used to classify the class of the unknown label data as an output. Remember we are using text file as our dataset which is slit into two parts training and test dataset. As these details are discussed in the further section in details.

A. Training and Test Dataset

In the start, we have started with intilizing a array for the data matrix. But later-on we have created a text file from where we read the training data and test dataset. We have divided the file in such a way that 70% of data act as a training data and other 30% data in text file act as a test dataset. The large data sets, the technique is not scale able, due to fact that k-NN classifier store all the training data into memory. The below is example of our text file data.

```
04, 9697, 9772, 9841, 9851, 9922,........, 0
06, 9732, 9753, 9854, 9955, 10107,......., 0
10295, 10353, 10461, 10598, 10612,......., 1
06, 9854, 9881, 9955, 10107, 10165,......, 1
10792, 10812, 10880, 11007, 11060,......., 2
10418, 10662, 10777, 10846, 11008,......., 2
```

The starting value till the last index of each item are the predictor values and the last index is our class label. As we have there classes: the first one class is even numbers, second class is odd numbers elements and third class is decimal class which is also known as neither odd or even. The even class is represented by 0, odd class is represented by 1 and decimal class dataset is represented by 2. The second way to store class label is to introduce a separate array in which the class label will be stored. In our code we have assumed the class labels are numeric and the number starts from 0.

```
function ExtractLabels(testData):
    actualLabels = array of size length of
    testData

    // Loop through each row of the testData
    for i from 0 to length of testData - 1:
        // Extract the last element of the row as
    the label
        actualLabels[i] = testData[i][length of
    row - 1]

    return actualLabels
```

The above code is used for extracting label from the dataset. As we have already discussed that the label of class is the last index. And for extraction of that specific last index, the above piece of code is used. The next point is test data. In test data, which is 30% of our file, is used to determine the accuraccy of our implementation. The classify function accepts multiple parameters fro the item to predict, it includes the number of classes in the training data as we have 3 classes, the number of nearest neighbors to evaluate and the matrix of training data, so the classifier is the function which determine or classify our data. Further discussed in the KNN implementation section.

B. KNN Impelementation

In the high level, If I summarize the pseudo-code, there are only three major steps which includes:

```
1. Compute distances from unknown
2. Sort the distances (nearest, farthest)
3. Use a vote to determine the result
```

These sections try to discuss the methods defined within the KNN classifier Class, these methods are discussed below:

1) Distance(double[] vector1, double[] vector2): As we have discussed in section 2(A) regarding the Euclidean distance. The distance function calculates the Euclidean distance between two vectors which in our case are vector1 and vector2. As, the formula of Euclidean distance is to go through each element of a vector and compute the squared difference and then find the accumulative sum. And then finally finding the square root of the whole sum, represents the Euclidean distance between two vectors.

```
function Distance(testData, trainData):
    sum = 0.0

    // Loop through each element of the data
    arrays
    for i from 0 to length of testData - 1:
        // Calculate the squared difference
    between corresponding elements
        difference = testData[i] - trainData[i]
        sum += difference * difference

    // Calculate the square root of the sum
    distance = squareRoot(sum)

    return distance
```

This is a simple implementation of distance finding by Euclidean. We can also use other alternatives including Mahalanobis distance, Mahattan distance. K-NN is not good for mixed numerical and non-numerical data(categorical data), as K-NN needs a two notion "nearest" and most distance metrics to work so it strictly works on either numerical data or non-numeric data.

2) Vote(IndexAndDistance[] info, double[][] trainData, int numClasses, int k): This method performs the voting mechanism between the k nearest neighbors. The arguments of this method are IndexAndDistance which represents distance and Indices of nearest neighbors in other we can say the information of nearest neighbors, then our training dataset (trainData), the total number of classes (numClasses), and the value of K. The step of Vote method is to initialize an array whose responsibility is to store the total number of votes against each class means every nearest k-neighbour has a vote. In the starting the array is initialized as zero. Then in the second step, it goes through the first k neighbors and retrieves its class label from the training data, and the respective array index is counted as one positive vote. In our case, the training data have either an odd number or an even number respective voting has been counted for each of them. Then the counted voting with the highest vote returns its label as the predicted class.

```
function Vote(info, trainData, numofclass, k):
    // Initialize an array to store the votes for
    each class
    votes = new Array of length numofclass,
    initialized with zeros

    // Count the votes for each class
    for i from 0 to numofclass - 1:
        votes[i] = 0

    // Loop through the first k neighbors
    for i from 0 to k - 1:
        idx = info[i].idx
        c = trainData[idx][20]
        votes[c] += 1

    // Initialize variables to track the class
    with the most votes
    mostVotes = 0
    classWithMostVotes = 0

    // Find the class with the most votes
    for j from 0 to numofclass - 1:
        if votes[j] > mostVotes:
            mostVotes = votes[j]
            classWithMostVotes = j

    // Return the class with the most votes
    return classWithMostVotes
```

As, we know that finding a class label is little bit trickier from the k items. As above code shows every k-nearest training gets one vote for its class label. In the start votes is intilized as zero. Then it add if class is nearest. As currently we also have to count the votes. So, the class which have higher number of vote is the class from where our test data belongs from that specific class. It can be either odd, even or decimal class.

3) Classifier(double[] unknownSDR, double[][] Sdrdata, int numofclass, int k): This method performs the KNN classification for a given unknown SDR which we are also saying as test-data. The arguments of this method include the testdata (unknownSDR) and the training data (Sdrdata) and the total number of classes which is 3 in our case. And the value of k as an input parameter. In this method, we have called all the methods including IndexandDistance objects to store indices and distances of all data points from the unknown SDR. The next, we have called the distance method, which will calculate the distances between the test data ad training data points that are stored in the array. Then we sorted an array in ascending order based on distances. Then in the final, we have called the vote method to determine the predicted class label based on k nearest neighbors. Then, the class with the highest vote is where the
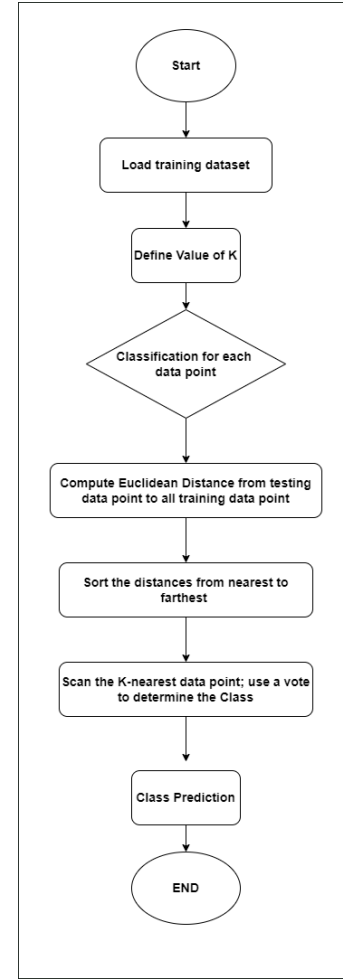

Fig. 6. KNN Flow Diagram according to steps

test data belongs to that specific class. And then this method returns the result of a specific class label.

4) IndexAndDistance Class: This class used to represent the index of a training item and its distance from the input (test data). It implements the IComparable¡IndexAndDistance¿ Interface to enable sorting based on distance. The CompareTo method is overridden to compare distances between instances of the IndexAndDistance class.

C. Unit Testing and Exception Handling

The outcomes of the unit test model are depicted in Figure.7. Unit test is developed to handle various scenarios, including handling exceptions. All the unit test which are designed and implemented for HTM classifier have been integrated into KNN classifier. During conducting the test, we have not noticed any instance failed tests. Indicating the robustness and reliability of the implemented algorithm. But there is one important thing is that the continuous testing and refinement of the model are imperative to ensure its accuracy and performance across diverse scenarios and datasets

IV. RESULTS

ALL the results should be here. ALL the results should be here. ALL the results should be here.
ALL the results should be here.
ALL the results should be here.
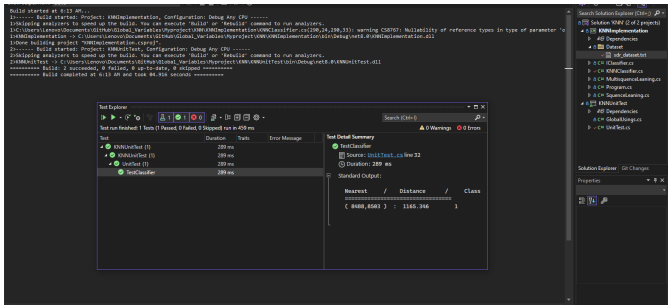ALL the results should be here.
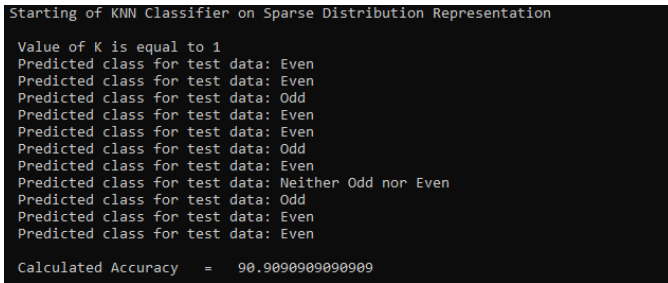
Fig. 7. Unit Test

Fig. 8. K=1

ALL the results should be here.

V. APPLICATION OF KNN

KNN Algorithm is utilized in different applications across different sectors, mostly in classification. The common cases includes:

A. Pattern Recognition

KNN is used for the identification of specific patterns, it can be in a text. Like it predict the missing fill-in-the-blanks. This also help in solving cache which are basically handwritten numbers. So, KNN can also to identify these patterns.
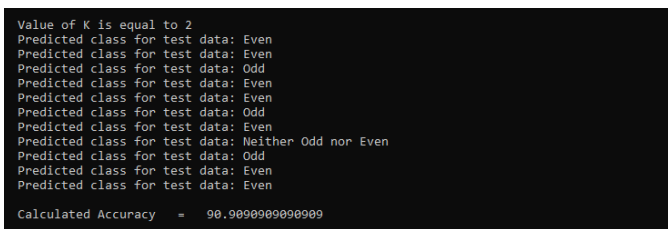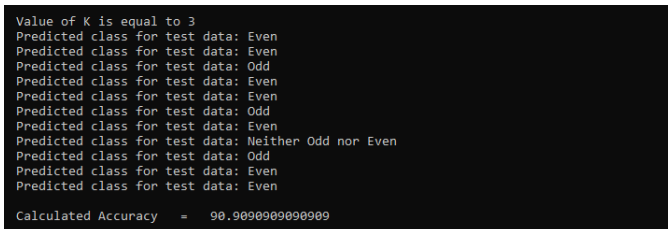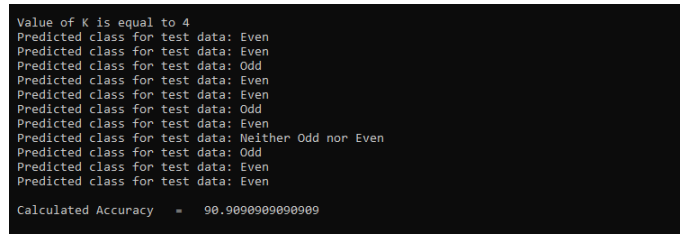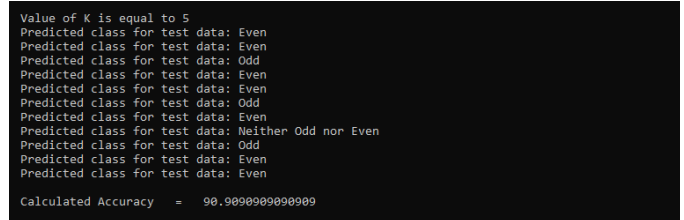

Fig. 9. K=2

Fig. 10. K=3

Fig. 11. K=4

Fig. 12. K=5

B. Healthcare

The common use of KNN is health department is prediction of chances of cancer as well as heart attacks risks. The algorithm try to learn from most likely gene expressions.

C. Recommendation Engines

When we surf on internet, the KNN algorithm can be used by the website to recommend us other additional content. This recommendation is based on user behaviour. But for larger datasets this approach is not optimal.

D. Data preprocessing

Mostly we have a missing values in our dataset, KNN algorithm can help to determine those values. Those estimated missing values are also known as missing data imputation.

E. Finance

The banks uses the credit card spending to predict the risk associated with the specific individual loan. As well as knn is used to determine the credit card worthiness of a loan application. So, KNN is used in a variety of economic and finance departments. As well as the other common use case is currency exchange forecasting, stock market as well as trading futures etc.

VI. CONCLUSION

In the conclusion, the journey began with the implementation of KNN Algorithm, which we have successfully achieved. We have tried to create a robust KNN model that can seamlessly integrate with the Neocortex API. This integration allows to receive a continuous sequences of data for prediction. and effectively classifying the input
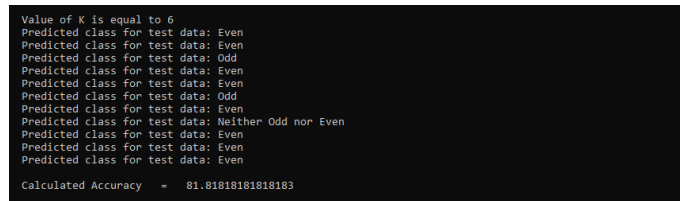

Fig. 13. K=6

sequences and predicting the output classes of receptive Input. We have achieved the accuracy rate of 80% for most of input data sequences. To ensure the robustness and reliablity of the model. In short if I can discribe our step then the first step, we feed the input sequences from the text file and that specific sequence get encoded. Then after spatial pooler we get sequence SDRs. Then it will be stored in HTM. The next step will be implementation of KNN classifier in which the test data will be mapped, the KNN helps to predict the class from where the sequence blogs to that specific class. We have tried to implement the Unit test with exception Handling. Moving forward, continued refinement and enhancement of the model will be essential to address evolving challenges and further elevate its performance across diverse applications and scenarios.

REFERENCES

[1] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.

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

[3] Guo, Gongde Wang, Hui Bell, David Bi, Yaxin. (2004). KNN Model-Based Approach in Classification.

[4] 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.

[5] Hastie, T., Tibshirani, R., and Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction.

[6] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46(3), 175-185.

[7] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). An introduction to statistical learning.

[8] X. L. Y. and 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.

[9] Taunk, Kashvi and De, Sanjukta and Verma, Srishti and Swetapadma, Aleena. (2019). A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. 1255-1260. 10.1109/ICCS45141.2019.9065747.

[10] Song Yang, Jian Huang, Ding Zhou, Hongyuan Zha, and C. Lee Giles, "Iknn: Informative k-nearest neighbour pattern classification," in Knowledge Discovery in Databases, (2007), pg. 248– 264.

[11] Han EH.., Karypis G., Kumar V.(2001) "Text Categorization Using Weight Adjusted k-Nearest Neighbour Classification". In: Cheung D., Williams G.J., Li Q. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2001. Lecture Notes in Computer Science, vol 2035. Springer, Berlin, Heidelberg.

[12] D. Wettschereck and D. Thomas G., "Locally adaptive nearest neighbour algorithms," Adv. Neural Inf. Process. Syst., pg. 184–186, 1994.

[13] Bishop, C. M. (2006). Pattern recognition and machine learning.

[14] Zhang, Y., and Kwok, J. T. (2003). Content-based image retrieval using edge-based color histogram and adaptive similarity measure. In Proceedings of the eleventh ACM international conference on Multimedia.

[15] Liu, H., Yu, L., and Liu, H. (2005). Discretization: An enabling technique. Data Mining and Knowledge Discovery, 6(4), 393-423.

[16] Brownlee, J. (2016). Machine learning mastery with Python: Understand your data, create accurate models and work projects end-to-end.

[17] Mueller, A. C., and Guido, S. (2016). Introduction to machine learning with Python: A guide for data scientists.

[18] McKinney, W., and others. (2017). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.

[19] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... and Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.

[20] Kulkarni, A., and others. (2018). Building intelligent systems: A guide to machine learning engineering.

[21] Sun, Jingwen and Du, Weixing & Shi, Niancai. (2018). A Survey of kNN Algorithm. Information Engineering and Applied Computing. 1. 10.18063/ieac.v1i1.770.

[22] K. Taunk, S. De, S. Verma and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[23] Komal Bharti, Prachi Shrivastava, Pradip K. Das, "Comparison of Multiple Classifiers for Audio-Visual Speaker Recognition System", 2023 International Conference on Computational Intelligence, Networks and Security (ICCINS), pp.1-6, 2023.

[24] Ozturk Kiyak, E.; Ghasemkhani, B.; Birant, D. High-Level K-Nearest Neighbors (HLKNN): A Supervised Machine Learning Model for Classification Analysis. Electronics 2023, 12, 3828. https://doi.org/10.3390/electronics12183828

[25] Zhang Z. Introduction to machine learning: k-nearest neighbors. Ann Transl Med. 2016 Jun;4(11):218. doi: 10.21037/atm.2016.03.37. PMID: 27386492; PMCID: PMC4916348.

[26] Cai, Y.L., Ji, D., Cai, D. (2010). A KNN Research Paper Classification Method Based on Shared Nearest Neighbor. NTCIR Conference on Evaluation of Information Access Technologies.

[27] Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)

[28] Goldberger, J., Roweis, S.T., Hinton, G.E., Salakhutdinov, R.: Neighbourhood components analysis. In: NIPS (2004)

[29] Qin, Y., Zhang, S., Zhu, X., Zhang, J., Zhang, C.: Semi-parametric optimization for missing data imputation. Applied Intelligence 27(1), 79–88 (2007)

[30] Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: NIPS, pp. 1473–1480 (2005)

[31] Wu, X., Zhang, C., Zhang, S.: Efficient mining of both positive and negative association rules. ACM Transactions on Information Systems (TOIS) 22(3), 381–405 (2004)

[32] Zhang, C., Zhu, X., Zhang, J., Qin, Y., Zhang, S.: GBKII: An imputation method for missing values. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 1080–1087. Springer, Heidelberg (2007)

[33] Zhang, S.: KNN-CF approach: Incorporating certainty factor to knn classification. IEEE Intelligent Informatics Bulletin 11(1), 24–33 (2010)

[34] Zhang, S.: Nearest neighbor selection for iteratively knn imputation. Journal of Systems and Software 85(11), 2541–2552 (2012)

[35] Zhu, X., Zhang, L., Huang, Z.: A sparse embedding and least variance encoding approach to hashing. IEEE Transactions on Image Processing 23(9), 3737–3750 (2014)

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

[37] 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.

[38] 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/NumentaHTMConcepts.pdf, 2006.

[39] 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.

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

[41] Zhu, X., Zhang, S., Jin, Z., Zhang, Z., Xu, Z.: Missing value estimation for mixed-attribute data sets. IEEE Transactions on Knowledge and Data Engineering 23(1), 110–121 (2011)

[42] T. M. COVER, MEMBER, IEEE, AND P. E. HART, MEMBER, IEEE, Nearest Neighbor Pattern Classification, IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-IS, NO. 1, JANUARY 1967

[43] Adeniyi, Adedayo and Wai, Zune and Yongquan, Y.. (2014). Automated Web Usage Data Mining and Recommendation System using K-Nearest Neighbor (KNN) Classification Method. Applied Computing and Informatics. 36. 10.1016/j.aci.2014.10.001.

[44] M A Mukid, T Widiharih, A Rusgiyono, A Prahutama, Credit scoring analysis using weighted k nearest neighbor, OP Conf. Series: Journal of Physics: Conf. doi:10.1088/1742-6596/1025/1/012114