

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 available 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 distinguished 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 unlabeled data 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 Matrices, KNN implementation, 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 distinguish, how much the weights influence the classification method. In this way, high weights will be assigned to the one who are closer neighbors,

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 unnecessary 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 Materics

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

1) *Computing Distance Matrics:* 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 matrices. 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 indentify 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 $|x_1 - x_2|$, and $|y_1 - y_2|$. So there the hypotenuse(d) is our Euclidean distance which is the between (x_1, x_2) and (y_1, y_2) . 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.

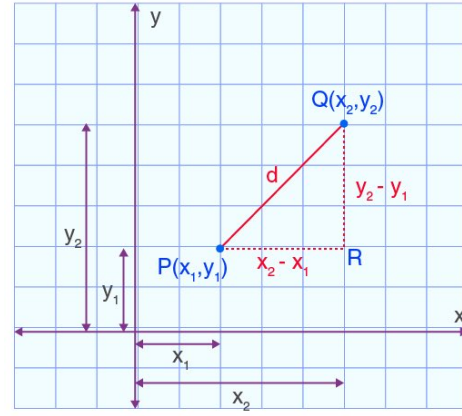


Fig. 1. Euclidean Distance

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

- 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 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.

It is easy to compute and widely understood, making it a popular choice for measuring distances between points in machine learning algorithms.

Additionally, the Euclidean distance metric works well for continuous features and is suitable for data represented in a high-dimensional space. It captures the straight-line distance between points in space, which aligns with the intuitive notion of distance.

Furthermore, the Euclidean distance metric is invariant to the scale and orientation of the coordinate system, making it robust to transformations of the data.

Overall, the Euclidean distance metric is a versatile and effective choice for measuring distances between data points in KNN, making it a 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 resloving 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

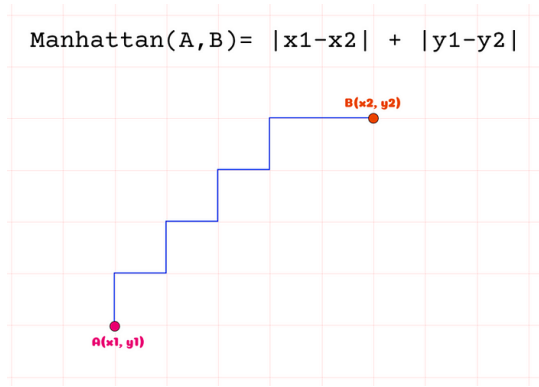


Fig. 2. Manhattan Distance

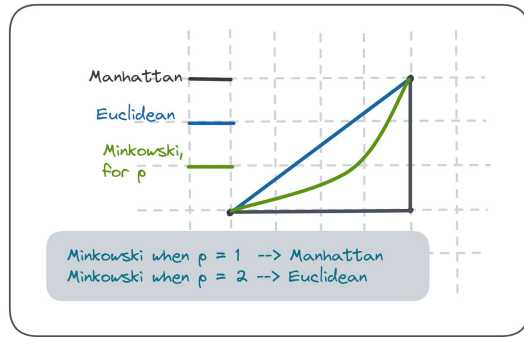


Fig. 3. Enter Caption

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

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

$$Minkowskidistance = \sqrt{(\sum_{i=1}^m |X_i - Y_i|)^{1/p}}$$

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:

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

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

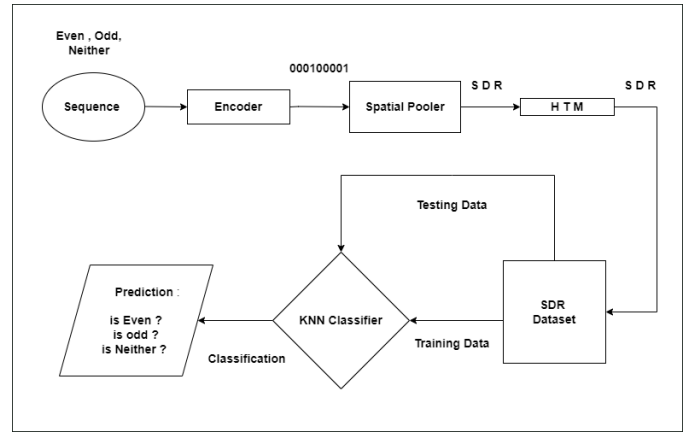


Fig. 4. Process Diagram

value of k the classification can lead to underfitting or overfitting. 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 underfitting or overfitting. 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 Principle:

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.

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



Fig. 5. General Flow

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 split 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 initializing an 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.

```

static int[] ExtractLabels(double[][] testData)
{
    // Extract actual labels from the dataset
    // Assuming labels are stored in the last
    // Column of each row
    int[] actualLabels = new int[testData.Length];
    for (int i = 0; i < testData.Length; i++)
    {
        actualLabels[i] = (int)testData[i].Last();
    }
    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 accuracy of our implementation. The classify function accepts multiple parameters from 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 Implementation

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.

```

double sum = 0.0;
for (int i = 0; i < testData.Length; ++i)
{
    double difference = testData[i]-trainData[i];
    sum += difference*difference;
}
return Math.Sqrt(sum);

```

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.

```

static int Vote(IndexAndDistance[] info, double
[] trainData, int numofclass, int k)
{
    int[] votes = new int[numofclass];

    for (int i = 0; i < numofclass; ++i)
    {
        votes[i] = 0;
    }

    for (int i = 0; i < k; ++i)
    {
        int idx = info[i].idx;
        int c = (int)trainData[idx][20];
        ++votes[c];
    }
    int mostVotes = 0;
    int classWithMostVotes = 0;
}

```

```

// Loop through each class
for (int j = 0; j < numofclass; ++j)
{
    if (votes[j] > mostVotes)
    {
        mostVotes = votes[j];
        classWithMostVotes = j;
    }
}

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 initialized 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 and 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 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

Overview of unit testing methodologies used. Description of `UnitTest1` class and its purpose. Explanation of how test datasets are generated and utilized for testing. Discussion of exception handling strategies implemented. Explanation of how errors and exceptions are handled during file reading and data parsing.

IV. RESULTS

ALL the results should be here.

V. DISCUSSION

A. Design Challenges

B. Applications of k-NN in machine learning

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

1) *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.

2) *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.

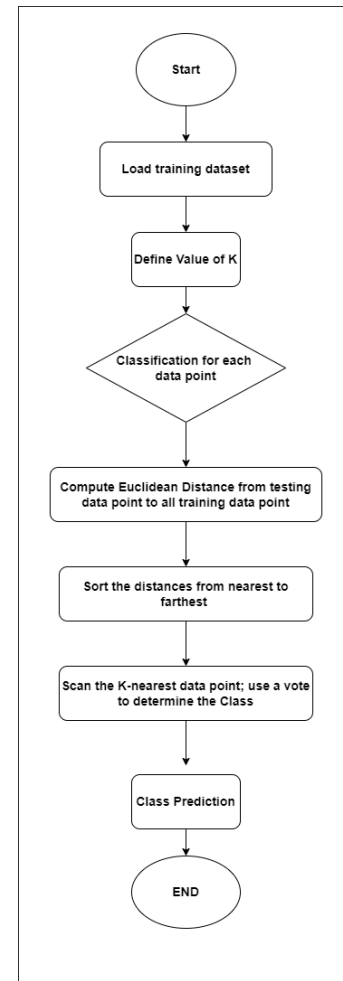


Fig. 6. KNN Flow Diagram according to steps

3) *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.

4) *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.

5) *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

ACKNOWLEDGMENT

The authors would like to thank...

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/ACCESS.2019.2926611, 2019.

- [3] S. A. a. J. H. Y. Cui, "Continuous Online Sequence Learning with an Unsupervised Neural Network Model," *Neural Comput.*, vol. 28, pp. 2474-2504, 2016.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] D. Wettschereck and D. Thomas G., "Locally adaptive nearest neighbour algorithms," *Adv. Neural Inf. Process. Syst.*, pg. 184–186, 1994.