IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

MODIFIED K-NEAREST NEIGHBOURS ALGORITHM FOR CUSTOMER CLASSIFICATION

SHYAMAL GOEL

Vellore Institute of Technology

ABSTRACT :*The original KNN algorithm takes into account the K nearest neighbours of a testing point, and that class or label is assigned to the testing point which holds a majority among the selected K points. But the algorithm does not take into account the classes or labels of the closest neighbours of each of the K selected points. If we also take this parameter into account, we can increase the accuracy of the KNN algorithm. The modified algorithm, has been implemented in [1]. We have taken the same algorithm and applied it on a customer classification dataset, and varied the parameters to increase accuracy and determine their most optimum values.*

Keywords *: K – nearest neighbours, clustering, validity, Euclidean distance*

INTRODUCTION

INTUITION BEHIND MODIFIED KNN

Let us say we want to classify a point using KNN algorithm using the 5 nearest neighbours. Out of these 5 nearest neighbours 3 belong to say class A and the other 2 belong to class B. So the testing point will be assigned to class A. But consider a scenario where the 2 points belonging to class B are significantly close to a large cluster of points belonging to class B. Hence , it will be better to assign the point to B. In the ordinary KNN algorithm this fact and the properties of the K nearest neighbours are overlooked.

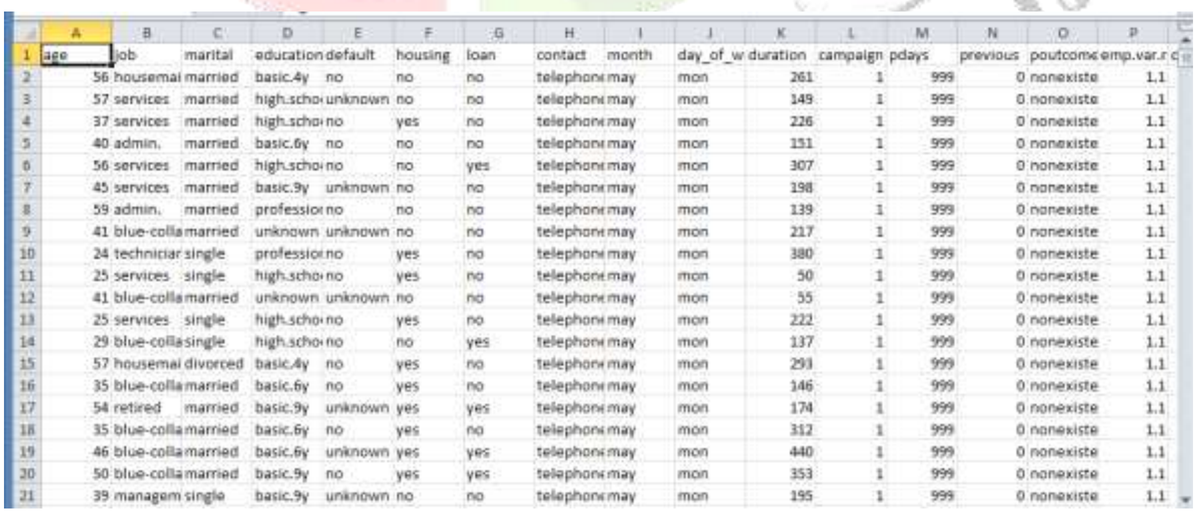
In the modified KNN algorithm [1], the validity assigned to each point is the measure of connectivity of that particular point to points belonging to the same class. Higher the validity higher is the points connection to similar points. When running the original KNN after assigning validity, we divide the validity of the K closest neighbours by the Euclidean distance to take into account the distance of the testing point from the neighbours. The class with the maximum weight out of the K closest neighbours is assigned to the testing point.

Understanding the dataset

In this dataset, we aim to classify whether the customer will respond to an organisation's marketing strategies or not. For any organisation or company, it is a very difficult task to figure out whether the target audience, will be interested in the product, and if not, what are the characteristics of the audience from whom we can reap maximum profit. At this juncture, artificial intelligence techniques can be applied for filtering out the target customers out of the pool. This would increase the overall effectiveness of the marketing campaign.

It has three kinds of attributes; numerical, which are in range type for all of them like (Age, Balance, Day, Duration, Campaign, PDays, and Previous), categorical are in set type as the attributes (Job, Marital, Education, Contact, Month, POutcome) and binary categorical (Default, Housing, Loan, Output)

INDEX	**ATTRIBUTES**	**ATTRIBUTE KIND**
1	Age	Numeric
2	Job	Categorical
3	Marital	Categorical
4	Education	Categorical
5	Default	Categorical
6	Housing	Categorical
7	Loan	Categorical
8	Contact	Categorical (Binary)
9	Month	Categorical
10	Day_of_week	Categorical
11	Duration	Numeric
12	Campaign	Numeric
13	Pdays	Numeric
14	Previous	Numeric
15	Poutcome	Categorical
16	Emp.var.rate	Numeric
17	Cons.price.idx	Numeric
18	Cons.conf.idx	Numeric
19	Euribor3m	Numeric
20	Nr.employed	Numeric
21	Y	Categorical(Binary)

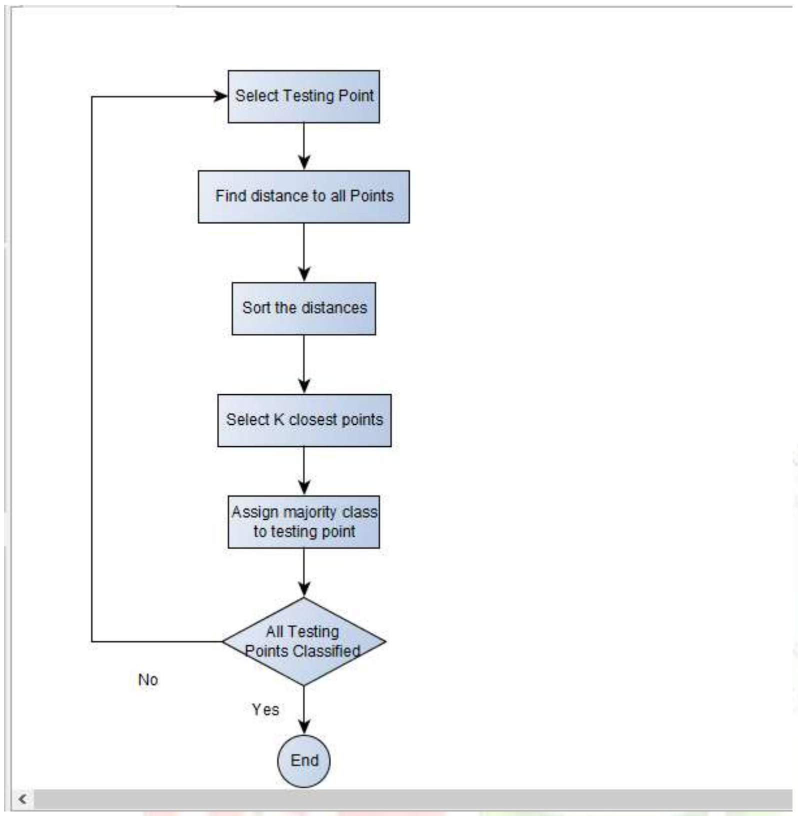


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	age	job	marital	education	default	housing	loan	contact	month	day_of_w	duration	campaign	pdays	previous	poutcome	emp.var.r
2	56	housemai	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexiste	1.1
3	57	services	married	high.scho	unknown	no	no	telephone	may	mon	149	1	999	0	nonexiste	1.1
4	37	services	married	high.scho	no	yes	no	telephone	may	mon	226	1	999	0	nonexiste	1.1
5	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexiste	1.1
6	56	services	married	high.scho	no	no	yes	telephone	may	mon	307	1	999	0	nonexiste	1.1
7	45	services	married	basic.9y	unknown	no	no	telephone	may	mon	198	1	999	0	nonexiste	1.1
8	59	admin.	married	professio	no	no	no	telephone	may	mon	139	1	999	0	nonexiste	1.1
9	41	blue-colla	married	unknown	unknown	no	no	telephone	may	mon	217	1	999	0	nonexiste	1.1
10	24	techniciar	single	professio	no	yes	no	telephone	may	mon	380	1	999	0	nonexiste	1.1
11	25	services	single	high.scho	no	yes	no	telephone	may	mon	50	1	999	0	nonexiste	1.1
12	41	blue-colla	married	unknown	unknown	no	no	telephone	may	mon	55	1	999	0	nonexiste	1.1
13	25	services	single	high.scho	no	yes	no	telephone	may	mon	222	1	999	0	nonexiste	1.1
14	29	blue-colla	single	high.scho	no	no	yes	telephone	may	mon	137	1	999	0	nonexiste	1.1
15	57	housemai	divorced	basic.4y	no	yes	no	telephone	may	mon	293	1	999	0	nonexiste	1.1
16	35	blue-colla	married	basic.6y	no	yes	no	telephone	may	mon	146	1	999	0	nonexiste	1.1
17	54	retired	married	basic.9y	unknown	yes	yes	telephone	may	mon	174	1	999	0	nonexiste	1.1
18	35	blue-colla	married	basic.6y	no	yes	no	telephone	may	mon	312	1	999	0	nonexiste	1.1
19	46	blue-colla	married	basic.6y	unknown	yes	yes	telephone	may	mon	440	1	999	0	nonexiste	1.1
20	50	blue-colla	married	basic.9y	no	yes	yes	telephone	may	mon	353	1	999	0	nonexiste	1.1
21	39	managem	single	basic.9y	unknown	no	no	telephone	may	mon	195	1	999	0	nonexiste	1.1

METHODOLOGY AND FLOWCHARTS:

ORIGINAL KNN

- For each instance in the testing set, find the distances to all the points in the dataset
- Sort the distances and select the K closest points
- Assign the testing point to that class which is a majority in the K closest points.
- Repeat the procedure for all the points in the testing set


Select Testing Point
Find distance to all Points
Sort the distances
Select K closest points
Assign majority class to testing point
All Testing Points Classified
No
Yes
End

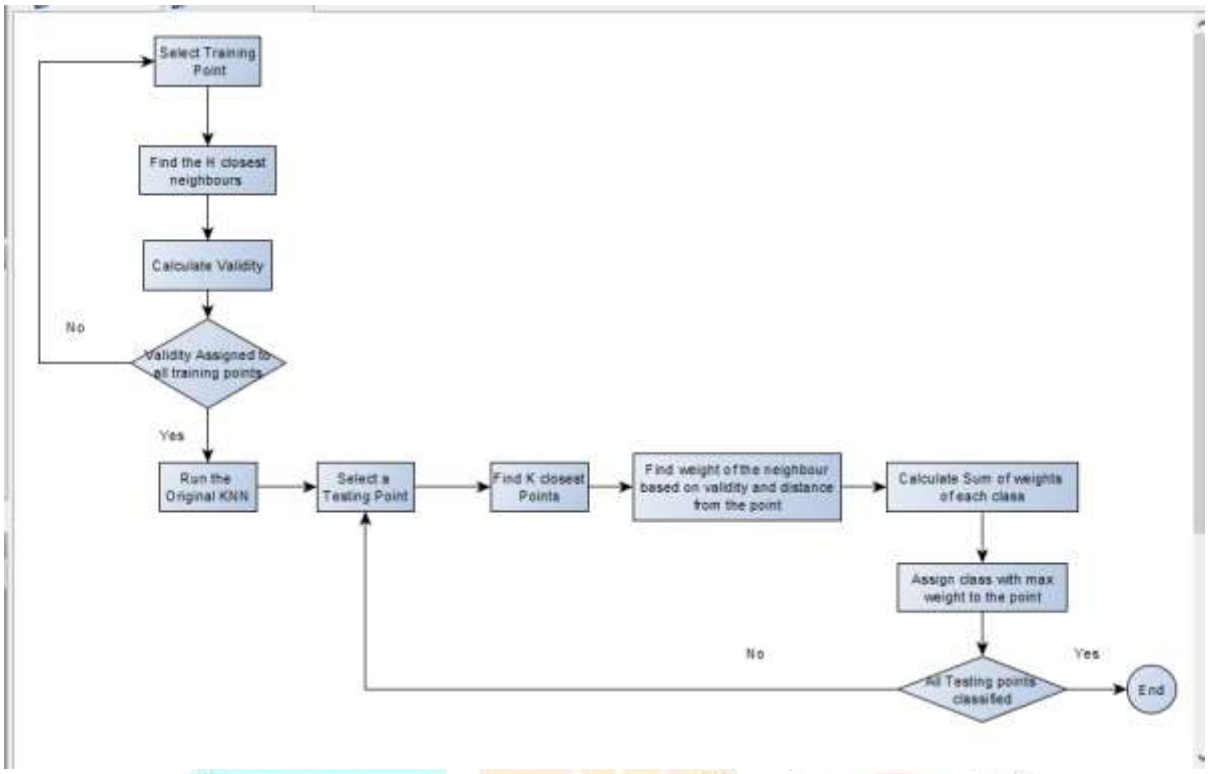
MODIFIED KNN

- We take constants H and K.
- For each point in the training dataset, calculate the H closest neighbours and assign a validity to the data point.
  - Validity(data point) = no. of neighbours of same class as of data point / H
- Run the original KNN algorithm.
- For each point in the testing dataset, find the K closest neighbours. For each neighbour, assign the value of weight given by
  - Weight(neighbour)= validity(neighbour)*(1/(0.5+Euclid_dist(testing point ,neighbour))

- Calculate the sum of weights of the each class in the K nearest neighbours
- Assign the class to the testing point with the maximum sum of the weights.
- Repeat the procedure for all the testing points.

PSEUDOCODE =>

```
X: training data, Y: labels
For each instance(xi), xi ∈ X {
Classify_Validity(X,Y,xi){
For t = 1 to m do
Compute distance d(Xt, xi)
End For
Compute set I1 containing indices for the H smallest distances d(Xt,xi)
Validity(xi) = (no. of labels of same class (yi ∈ Y) as of xi )/H
}
End For
Classify (X,Y,P) {   //P – point to be classified
For t = 1 to m do
Compute distance d(Xt,P)
End For
Compute set I2 containing indices for the K smallest distance d(Xi,P)
For each point pi, pi ∈ I2
Weight(pi) = validity(pi)*[1/(0.5+Euclid_distance(P,pi))]
End For
For each yi, yi ∈ Y
Sum(yi) = sum of weights of pi of class yi     // pi ∈ I2
End For
Assign yi with maximum sum of weights to point P.
}
```


Select Training Point
Find the H closest neighbours
Calculate Validity
Validity Assigned to all training points
No
Yes
Run the Original KNN
Select a Testing Point
Find K closest Points
Find weight of the neighbour based on validity and distance from the point
Calculate Sum of weights of each class
Assign class with max weight to the point
All Testing points classified
No
Yes
End

IMPLEMENTATION

CALCULATE THE OPTIMUM K,H VALUES OF THE FOLLOWING 3 CASES:

NOW WE VARY THE DENOMINATOR , TO GIVE VARIED IMPORTANCE TO EUCLIDEAN DISTANCE.

CASE 1:

Weight(neighbour)= validity(neighbour)*(1/(**0.5**+Euclid_dist(testing point ,neighbour))

K	ORIGINAL (%)	H	MODIFIED KNN(%)
5	72.058	3	73.529
5	72.058	5	75.157
5	72.058	8	75.254
5	72.058	9	75.339
5	72.058	10	76.691
5	72.058	11	76.694
5	72.058	13	75.157
5	72.058	15	76.691
5	72.058	20	75.758
3	70.588	3	73.413
3	70.588	5	73.523
3	70.588	8	73.623
3	70.588	10	74.325
3	70.588	13	74.127
3	70.588	15	73.834
10	76.725	3	77.145
10	76.725	5	77.524
10	76.725	8	77.326
10	76.725	10	78.234
10	76.725	13	78.431
10	76.725	15	78.318
10	76.725	18	77.763
10	76.725	20	77.213

CASE 2:

Weight(neighbour)= validity(neighbour)*(1/(**0.3**+Euclid_dist(testing point ,neighbour))

K	ORIGINAL (%)	H	MODIFIED KNN(%)
5	72.041	3	73.145
5	72.041	5	74.347
5	72.041	8	74.419
5	72.041	9	75.537
5	72.041	10	76.721
5	72.041	11	76.811
5	72.041	13	76.743
5	72.041	15	76.528
5	72.041	20	76.788
3	70.489	3	71.215
3	70.489	5	72.478
3	70.489	8	72.784
3	70.489	10	73.632
3	70.489	13	72.614
3	70.489	15	72.513
10	76.738	3	77.761
10	76.738	5	77.891
10	76.738	8	78.147

10	76.738	10	78.435
10	76.738	13	78.519
10	76.738	15	77.738
10	76.738	18	77.224
10	76.738	20	77.221

CASE 3:

Weight(neighbour)= validity(neighbour)*(1/(**0.7**+Euclid_dist(testing point ,neighbour))

K	ORIGINAL (%)	H	MODIFIED KNN(%)
5	72.073	3	74.437
5	72.073	5	74.892
5	72.073	8	75.326
5	72.073	9	76.553
5	72.073	10	76.773
5	72.073	11	76.789
5	72.073	13	75.843
5	72.073	15	74.574
5	72.073	20	74.496
3	70.613	3	71.435
3	70.613	5	72.746
3	70.613	8	72.861
3	70.613	10	73.632
3	70.613	13	73.591
3	70.613	15	72.747
10	76.705	3	77.428
10	76.705	5	77.549
10	76.705	8	77.844
10	76.705	10	78.134
10	76.705	13	78. 215
10	76.705	15	77.996
10	76.705	18	77.819
10	76.705	20	77.593

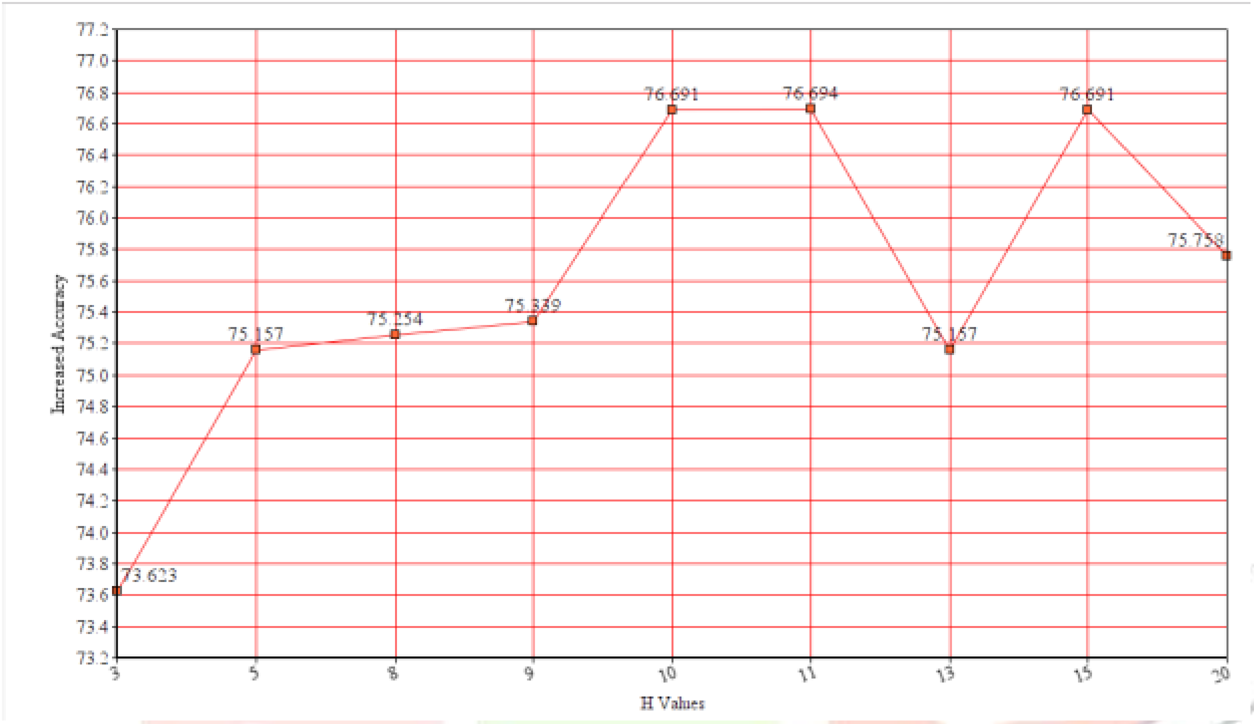
GRAPHICAL ANALYSIS:

CASE 1 :

Weight(neighbour)= validity(neighbour)*(1/(0.5+Euclid_dist(testing point ,neighbour))

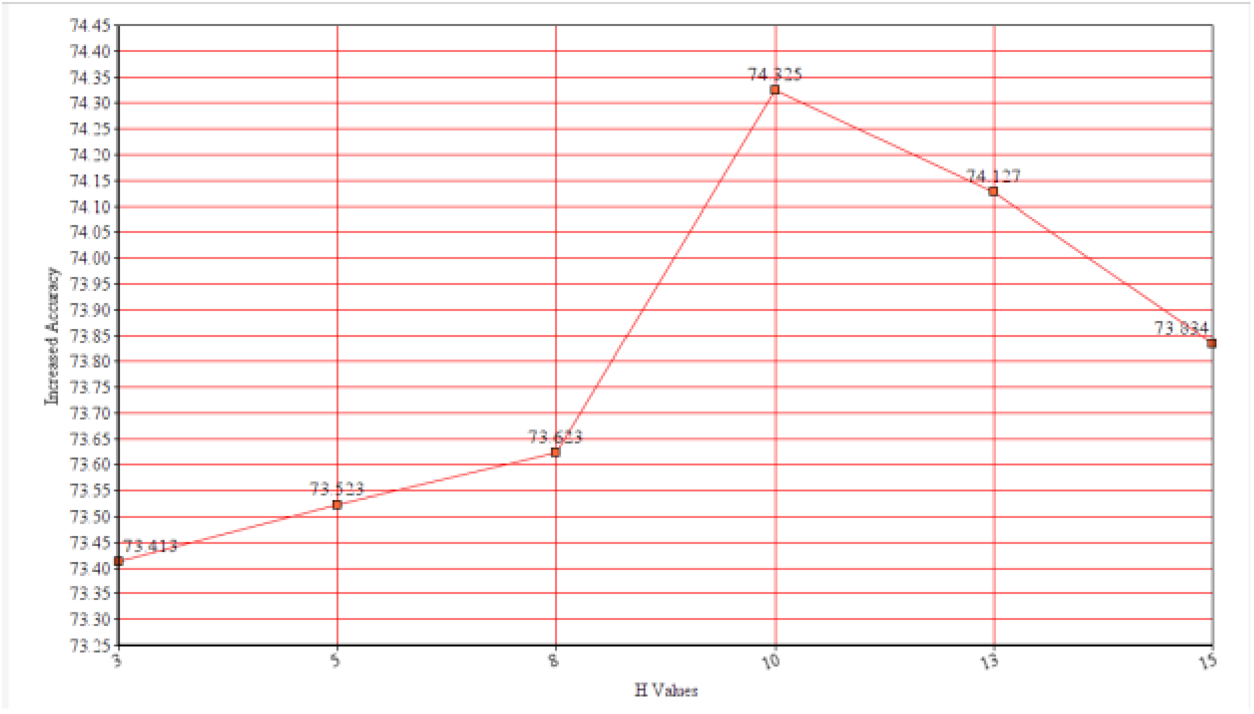
K = 5,

Original Accuracy : 72.058%



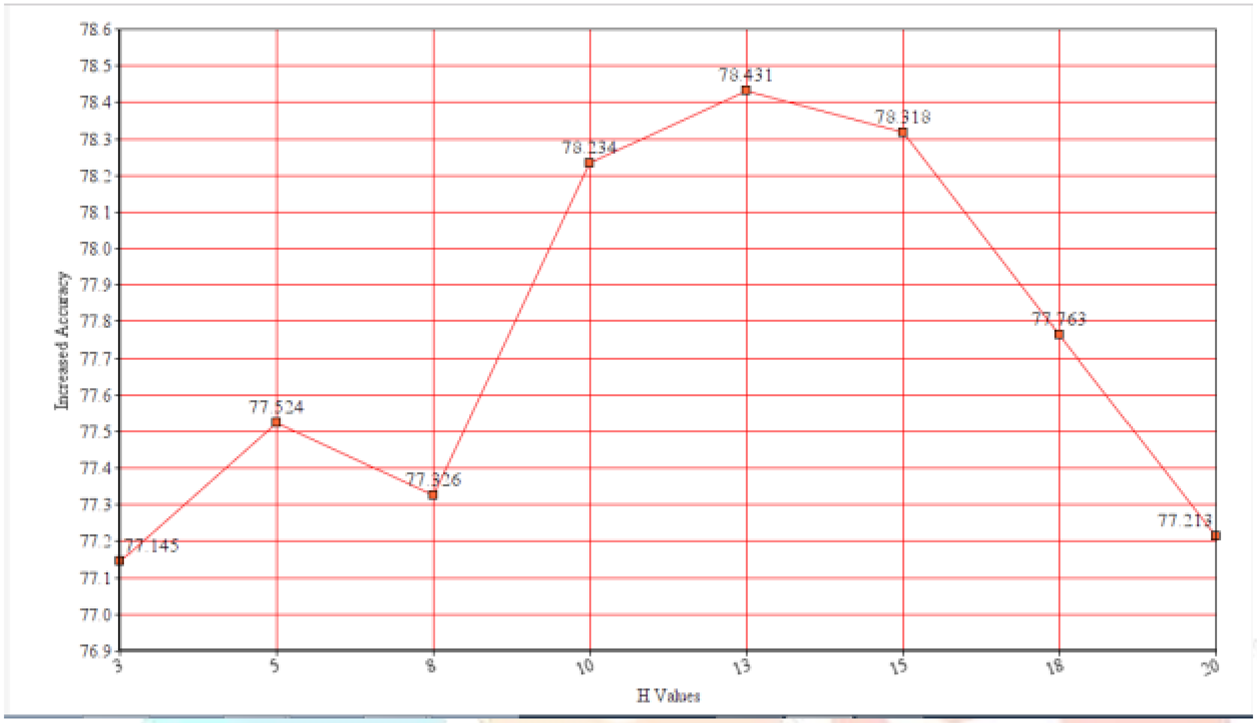
K = 3,

Original Accuracy : 70.588%



K = 10,

Original Accuracy : 76.725%

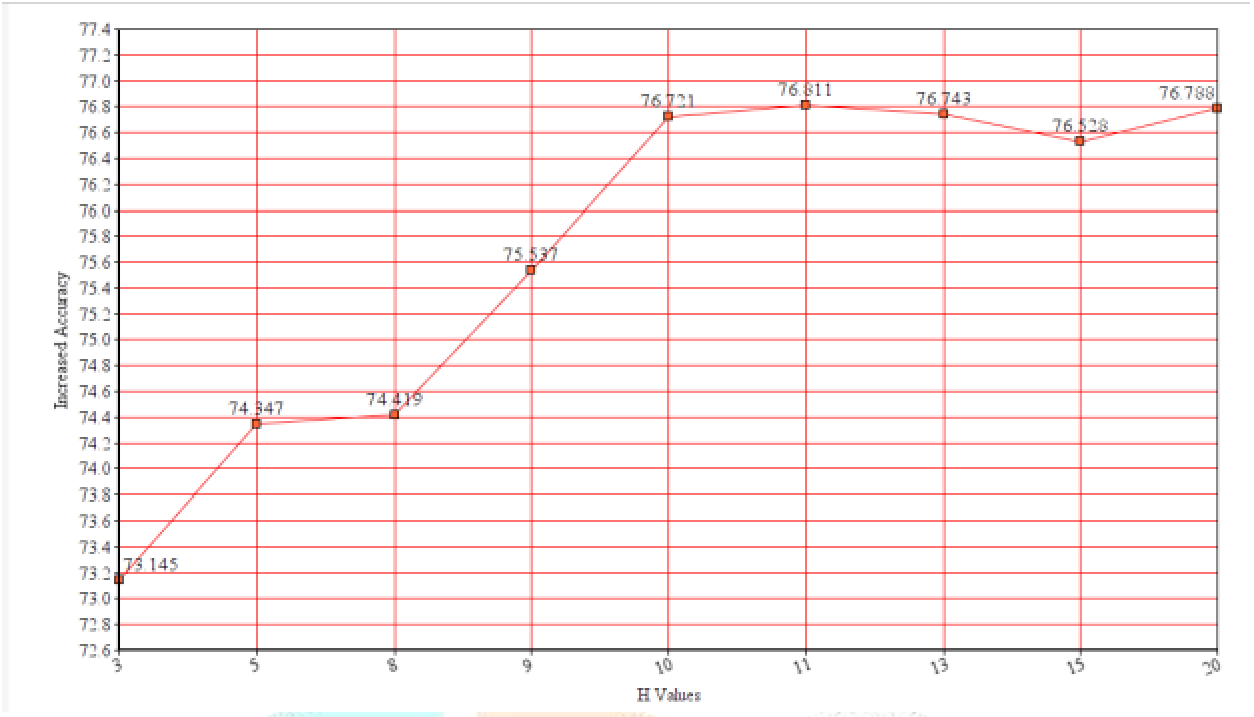


<u>CASE 2:</u>

Weight(neighbour)= validity(neighbour)*(1/(**0.3**+Euclid_dist(testing point ,neighbour))

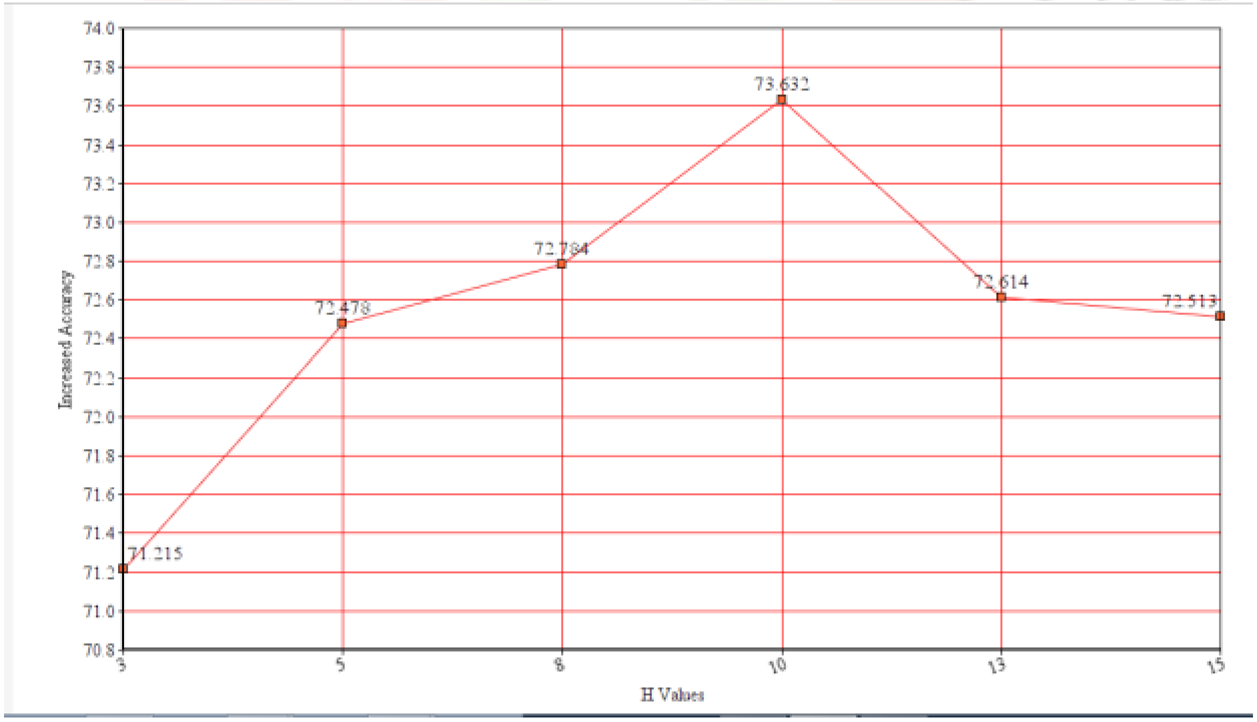
K = 5,

Original Accuracy : 72.041%



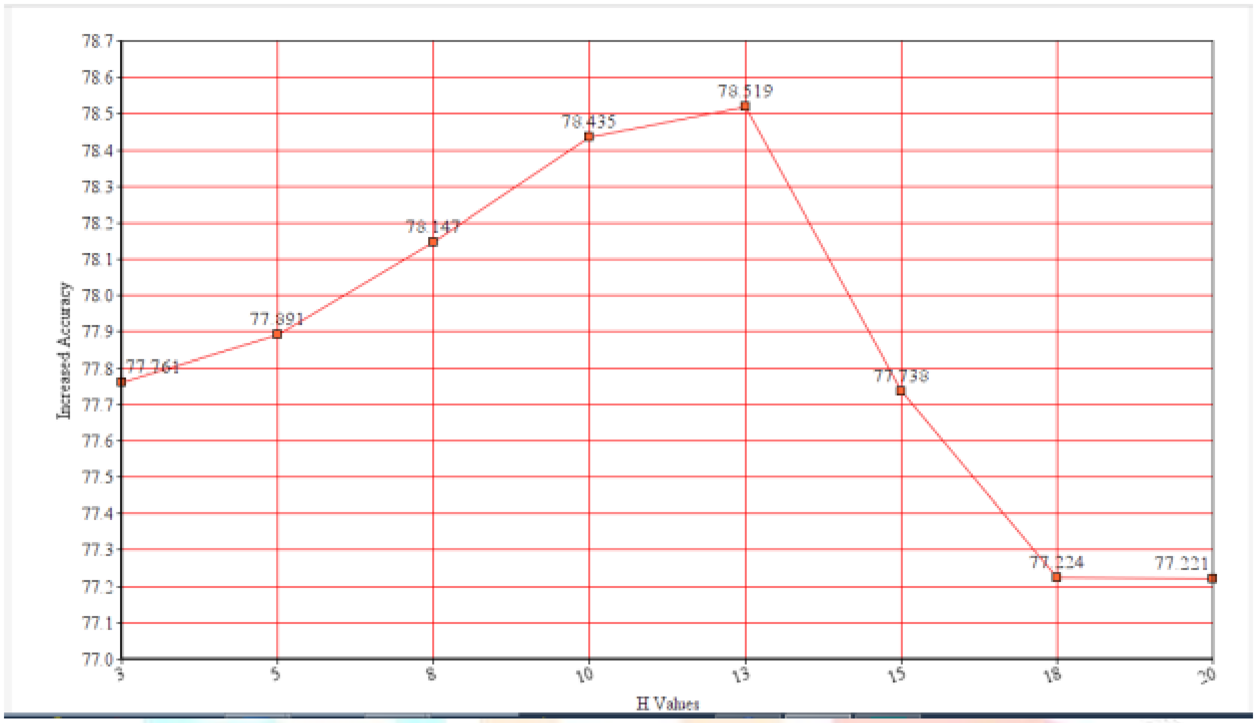
K = 3,

Original Accuracy : 70.489%



K = 10,

Original Accuracy : 76.738%



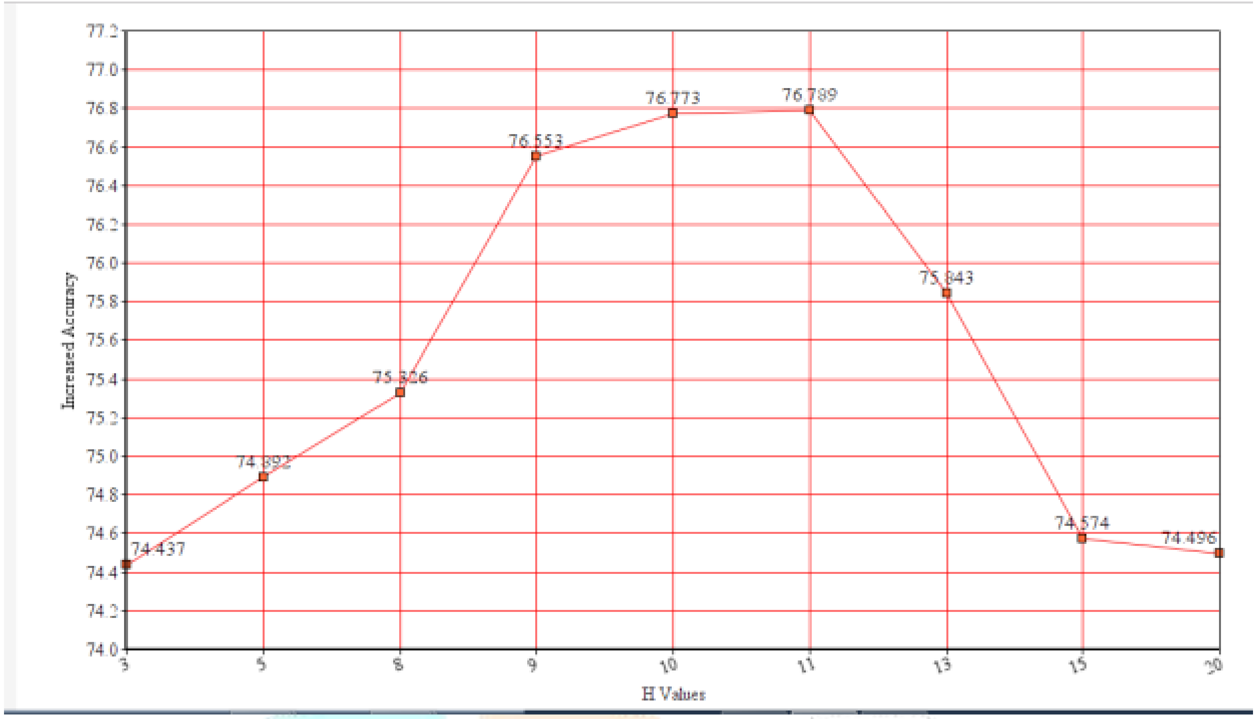
CASE 3:

Weight(neighbour)= validity(neighbour)*(1/(**0.7**+Euclid_dist(testing point ,neighbour))

K = 5,

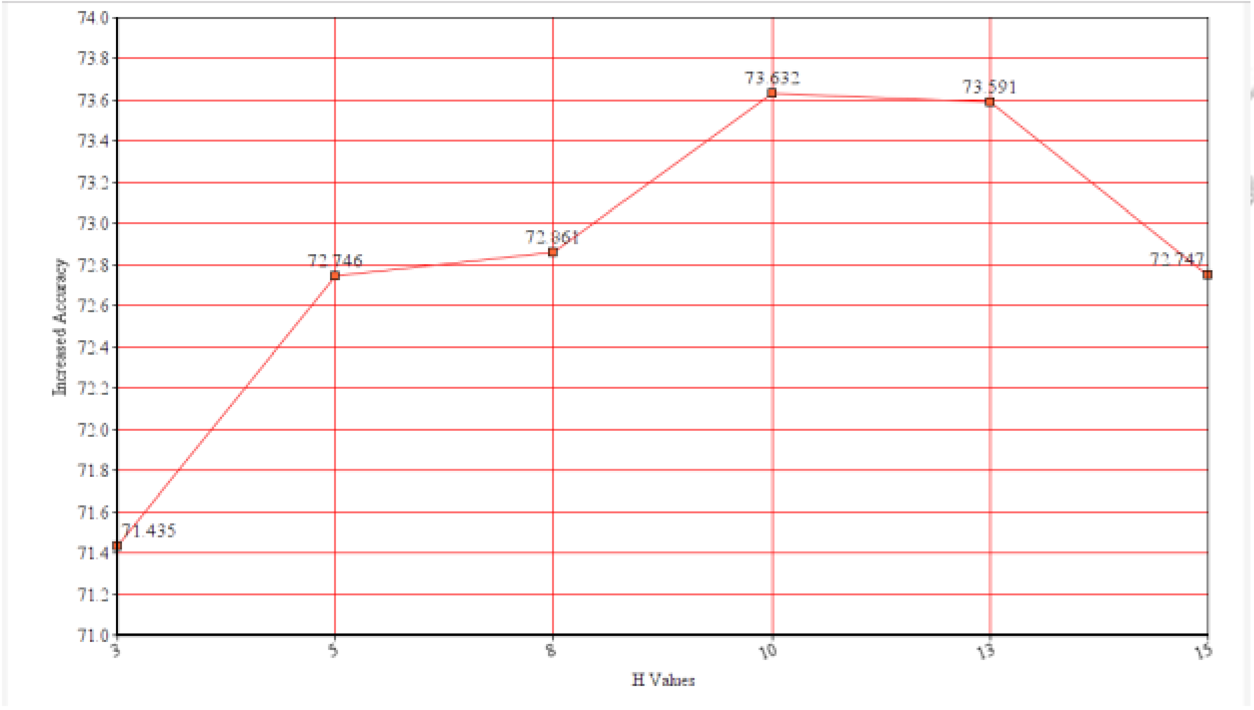
Original Accuracy : 72.073%

IJCRT



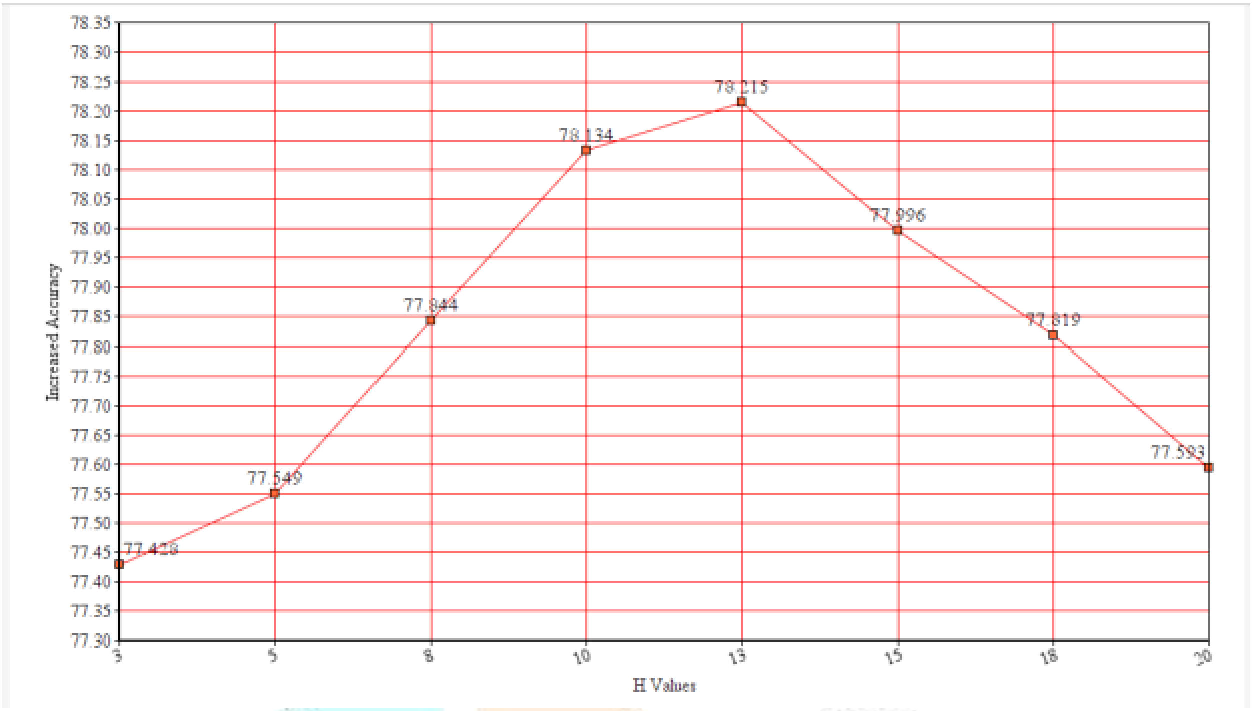
K = 3,

Original Accuracy : 70.613%



K = 10,

Original Accuracy : 76.705%



RESULTS

We can clearly see that the modified algorithm has increased the percentage of accuracy in all the 3 cases , when we vary K from 3 to 10, and H from 3 to 20. For a certain value of K, we can see that the accuracy increases when we vary H from a value lower than K to a value almost double than that of K, and then accuracy relatively decreases. But the accuracy is always greater than the accuracy of the original KNN algorithm.

In case 1 , we see that for the value of K as 5, the accuracy is 72.058%, and accuracy increases to 73.623% when we H as 3. As we cary H from 3 to 11, accuracy steadily increases to 76.694% and then steadily decreases to 75.758% when H is 20. Similarly for value of K as 3 , the accuracy is 70.588% and when we introduce H as 3 the accuracy becomes 73.413%. The accuracy increases till 74.325% when H is 10 and decreases to 73.834% when H becomes 15. When we take the value of K as 10, the accuracy is 76.725%, and when H is put as 3, the accuracy increases to 77.145. The accuracy steadily increases to 78.431% as H increases to 13, and then reduces steadily to 77.213% as H becomes 20.

In case 2, we have changed the denominator in to calculate the weight of the point. [0.3 + euclid_dist(test point, neighbour point)] is now used as denominator instead of [0.5 + euclid_dist(test point, neighbour point)] as in case 1. As we put the value of K =5, the accuracy is 72.041%. When we introduce H as 3, the accuracy increases to 73.145%. The accuracy steadily increases to 76.811% as H increases to 11 , and then slightly decreases to 76.788% as H approaches 20. When we put the value of K as 3, the accuracy is 70.489%. As we vary H from 3 to 10, accuracy increases from 71.215% to 73.632%. As H approaches 15, accuracy steadily decreases to 72.513%. When we put K as 10, accuracy is 76.738%. When we vary H form 3 to 13, accuracy steadily increases from 77.761% to 78.519%. As H approaches 20, accuracy decreases to 77.221%.

In case 3, the denominator is [0.7 + euclid_dist(test point, neighbour point)]. When we put K as 5, the accuracy is 72.073%. When we introduce the value of H as 3,accuracy is 74.437%. When we steadily increase H to 11, the accuracy steadily increases to 76.789%, and when we increase H from 11 to 20, the accuracy steadily decreases to 74.496%. When K is put as 3, the accuracy is 70.613%. When we increase H from 3 to 10, accuracy increases from 71.435% to 73.632%. When we increase H from 10 to 15, the accuracy steadily decreases to 72.747%. When K is put as 10, the accuracy is 76.705%. When we put H as 3, accuracy becomes 77.428%. When H steadily increases to 13, accuracy steadily increases to 78.215%. The accuracy steadily decreases to 77.593% when H increases to 20.

CONCLUSION

In this project, we have successfully implemented a modification of the K Nearest Neighbors Algorithm[1]. The original algorithm only considers the majority labels of the K closest points of the testing point, but this overlooks the fact that those K closest points may themselves be close to a large cluster of points belonging to the same label. This may lead to erroneous results. Therefore in the modified implementation, we first calculate the validity of each point in the dataset. Validity is the number of H closest points belonging to the same label divided by H. Then we run the original KNN algorithm, and select the K nearest neighbors. We assign weights to these K points depending on the validity of these points and the Euclidean distance from the testing point. The label with the maximum weight is assigned to the testing point. This successfully overcomes the drawback of the original KNN, and manages to increase the accuracy of the algorithm, leading to more accurate and reliable results.

REFERENCES

1. MKNN: Modified K-Nearest Neighbor Hamid Parvin, Hosein Alizadeh and Behrouz Minaei-Bidgoli, Proceedings of the World Congress on Engineering and Computer Science, San Francisco, USA
2. scikit-learn.org
3. r-project.org
4. Data Mining: Concepts and Techniques, Jiawei Han, Micheline Kamber and Jian Pei
5. Mining of Massive Datasets, Anand Rajaraman and Jeffrey Ullman
6. Analyzing Social Networks, Stephen P Borgatti,, Martin G. Everett, Jeffrey C. Johnson