Classification Rule Discovery for Software Bug Severity using KNN with different distance metric

Raj Kumar

Research Scholar, Dept. of CSE, I.K. Gujral Punjab Technical University, Kapurthala, Jalandhar, Punjab, PIN-144603, India
rajshira@gmail.com

Sanjay Singla

Professor, Dept. of CSE, GGS College of Modern Technology, Kharar, Mohali, PIN-140301, India
dr.ssinglacs@gmail.com

Abstract
KNN is one of the most popular machine learning algorithms. There are different parameters on which the performance of an algorithm depends. The performance of KNN algorithm depends on value of K, and distance metric. So while implementing the KNN algorithm for the classification rule discovery, a proper care must be taken to choose the value of 'K'. Second parameter which is also very important for the KNN algorithm is the distance metric. As there are different types of the distance function which can be used with KNN, selection of distance function is also an effective task. In this paper value of 'K' is selected on the basis of mean error and it is found that best values of 'K' are 8, 12 and 15. Then using K=8, 12 and 15, KNN algorithm is implemented for bug severity classification. A comparative analysis of three distance metrics Euclidean, Manhattan and Hamming distance functions is done.

***Keywords*: Bug; Distance Metric; KNN, Classification.**

1. Introduction

Day by day, there is rapid growth in the use of software. Deviation of software from expected result may be due to a bug/defect. Bugs are generally mistakes which start because of human cooperation. 57% bug starts from mistake made by human, which could be either because of remissness or inattentiveness [Nakamura, K. et al(2002), Kumar, R. et al.(2019)Kumar, R et al.(2012]. In case of software failure, these bugs can cost organizations a large measure of cash and for some situation loss of human lives for example programming bug in the a Royal Air Force Chinook airplane's motor control PC made it crash in the year 1994 and 29 individuals were killed[Rogerson, S. (2002)].

So, bug/defect should be resolved as early as possible. Bug repository contains the data of the software bugs on which different machine learning techniques have been applied by the different researchers. First of all bug detection using machine learning was done in 2004[Murphy G. et al. (2004)]. Second attempt was made in 2006; SVM and Naïve Bayes algorithms are used for bug classification [Anvik J. et al.(2006)]. Then an automatic software bug triaging system was proposed in 2009[Ahsan, S. N (2009)]. A approach to reduce the bug triaging effort represented by J. Anvik and G. Murphy[Anvik J(2011)]. In this way different approaches have been proposed by the number of researches.

So using the machine learning approach for software bug classification is really a challenging task. Almost all the users applied the different machine learning techniques on the dataset of the Mozilla and Eclipses. In this paper implementation of software bug classification is done for the bug data of the Jira tool which is very latest tool being used by the software developers/testers.

2. K-Nearest Neighbour

KNN is one of the very famous classification algorithms. Its prominence springs from the way that it is straightforward and deciphers yet ordinarily [Cover T. & Hart P. (1967). Guo G. et al.(2003), Kumar R., & Verma R. (2012)]. The accuracy of KNN is analogous or far and away superior to other complicated and complex algorithms [Franz P. (2005)]. KNN is a lazy type algorithm, it means it doesn't expressly learn the model; however it spares all the training tuples and utilize the entire training set for prediction or classification. However eager approaches like e.g. SVM, some tuples (i.e non- support vectors) can be easily rejected [Han J. (2006), Weinberger K. Q., & Saul L. K. (2009)].Because entire data is used in case of the KNN algorithm the

main drawback of using the KNN is the storage problem. But sometimes depending on the situation, it is advantageous to use KNN because of its execution speed. For the small data set, KNN is a very useful algorithm [Yong Z et al. (2009)].

Other significant supposition that will be that KNN algorithm necessitates that the data is in metric space. This implies that on the basis of the separation between, a metric can be defined [Weinberger, K. Q., & Saul, L. K. (2009)].

Characterizing distance metric can be a genuine test. A fascinating thought is to discover the separation measurements utilizing AI (basically by changing over the data to vector space, speak to the contrasts between objects as separations among vectors and become familiar with those distinctions. Different types of the distance metrics are [Hu L. Y(2016) 14, Prasath V. B (2017)]:

Minkowski Distance:
Minkowski distance separation is characterized as the comparability metric between two points in the normed vector space. It is donated as:

$$D(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

The above metric is a generalized metric which represents different distance metric, depending on the value of 'p' (Lp form)

- Manhattan Distance (when value of p = 1)
- Euclidean Distance (when value of p = 2)
- Chebychev Distance(when value of p = ∞)

a) Euclidean Distance:
This is the geometrical separation that is being utilized in day by day life.

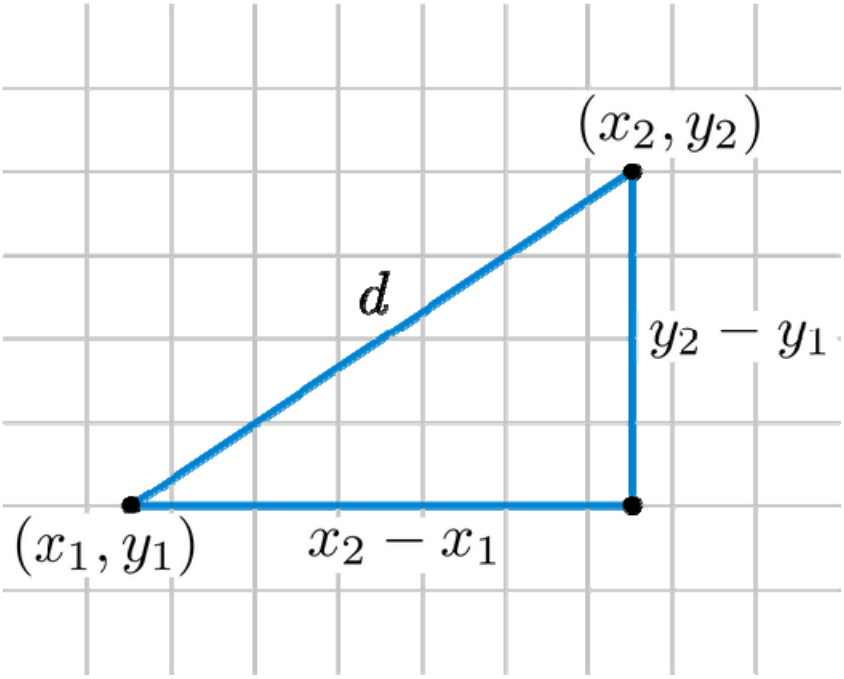

Figure 2.1: Difference between to data points

In case of 2D space

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

According to Pythagorus Theorem:

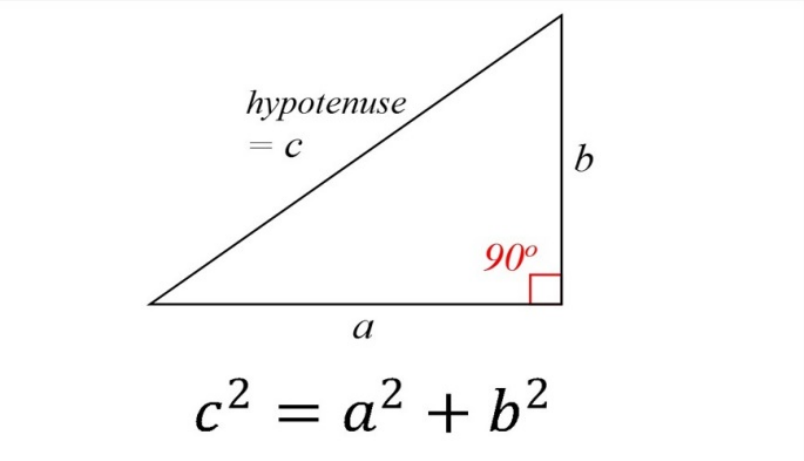

Figure 2.2: Pythagorous Theorm

So the Euclidian distance can be calculated as:

$$D(x,y) = \sqrt{\left(\sum_{i=1}^{n} |x_i - y_i|^2\right)}$$

b) Manhattan Distance:

In the Manhattan distance function the absolute difference between two points is calculated. Manhattan distance also known as the city block distance. For example, a city may be represented using the matrix and this can be walk through (or going through a Manhattan). The edges of small squares in the matrix donates the streets. So to compute the distance between squares A and B, it is required to go through the edges of the number of small squares.

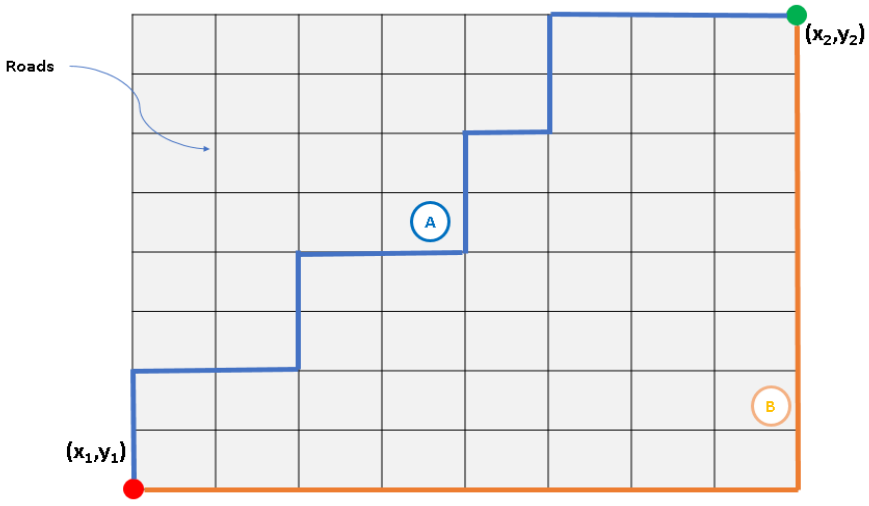

Figure 2.3: Computation of Manhattan distance between to squares.

Manhattan distance in 2D space:

$$\left|(x_2 - x_1)\right| + \left|(y_2 - y_1)\right|$$

By putting p=1 in Minkowski distance equation, the manhatten distance can be computed as:

$$D(x,y) = \sum_{i=1}^{n} |x_i - y_i|$$

City Block Distance, Taxicab Geometry are the names by which Manhattan distance is also known as.

c) Hamming Distance:

Hamming distance is a very important metric to compute the difference between two binary strings. The difference between two binary strings of equal length can be computed by computing the number of bit positions which are different in both the stings say p and k. The hamming distance is represented as: d(p,k)

Cosine, Mahalanobis, Chi square, Correlation etc. are some other distance functions.In this paper Euclidian, Manhattan and Hamming distance metrics are used.

3. Result and Discussion:

A multiclass classification of bug severity is done in this paper. The bug severity can be classified as S2, S3, S4 and S5 according to the impact of the bug. A bug dataset of Jira tool is taken for the implementation of the KNN algorithm with different value of 'K' and by using different type of distance function. First of all question arises: What should be the value of K.?

To decide the value of the 'K', the mean error is computed for k=1 to 40. As depicted in the figure 3.1, it has been witnessed that the error near to zero is found for K=8, 12 and 15. So it is desirable to take the value of K=8, 12 or 15. A comparative analysis of different distance functions is done by taking K=8, 12 and 15.

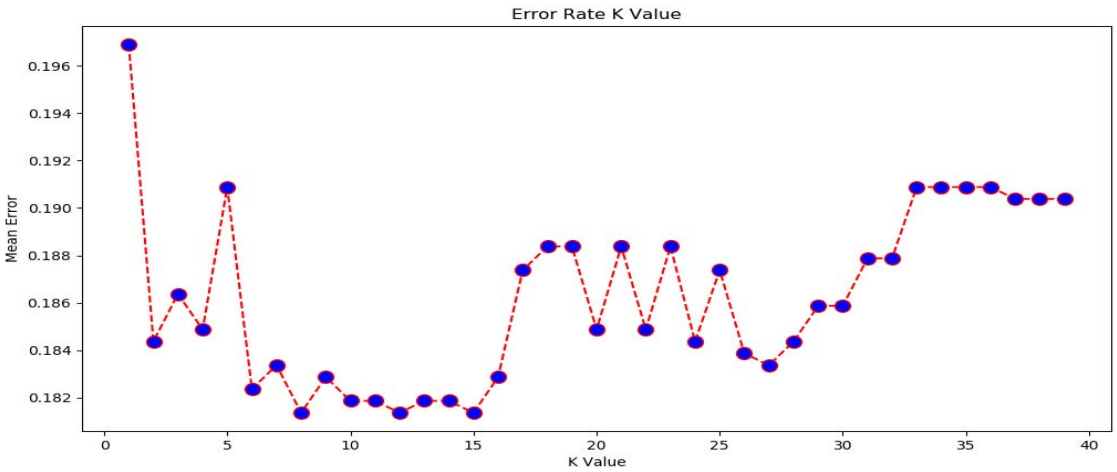

Figure 3.1 Mean error for K=1 to 40

Implementation results are shown in Table 3.1. Comparative analysis can be done with respect the value of K or distance metric. After calculating average accuracy for K=8, 12 and 15, it has been observed that classification accuracy K=8 is 79.99%, for K=12 is 80.73% and for K=15 is 81.66%. So it can be concluded that for K=15 is giving best accuracy however these values are very close to each other. Another comparison can be done with respect to the Distance function as shown in the figure 3.3. For the different distance functions average (for all K's) of Accuracy, Precision and Recall are computed. Following observations are made:

- **Euclidean distance:** average Accuracy, Precision and Recall are is 79.99%, 78.67% and 65% respectively
- **Manhattan distance:** average Accuracy, Precision and Recall are is 81.03%, 79.33% and 64.33% respectively
- **Hamming distance:** average Accuracy, Precision and Recall are is 81.36%, 81% and 67% respectively

	K=8			K=12			K=15		
Distance Function	Accuracy %age	Precision %age	Recall %age	Accuracy %age	Precision %age	Recall %age	Accuracy %age	Precision %age	Recall %age
Euclidian	79.91	70	67	80.11	83	62	79.96	83	66
Manhattan	79.26	68	63	81.76	84	66	82.06	86	64
Hamming	80.81	80	68	80.31	83	64	82.97	80	69

Table 3.1 Accuracy, Precision and Recall for Euclidian, Manhattan and Hamming distance metric

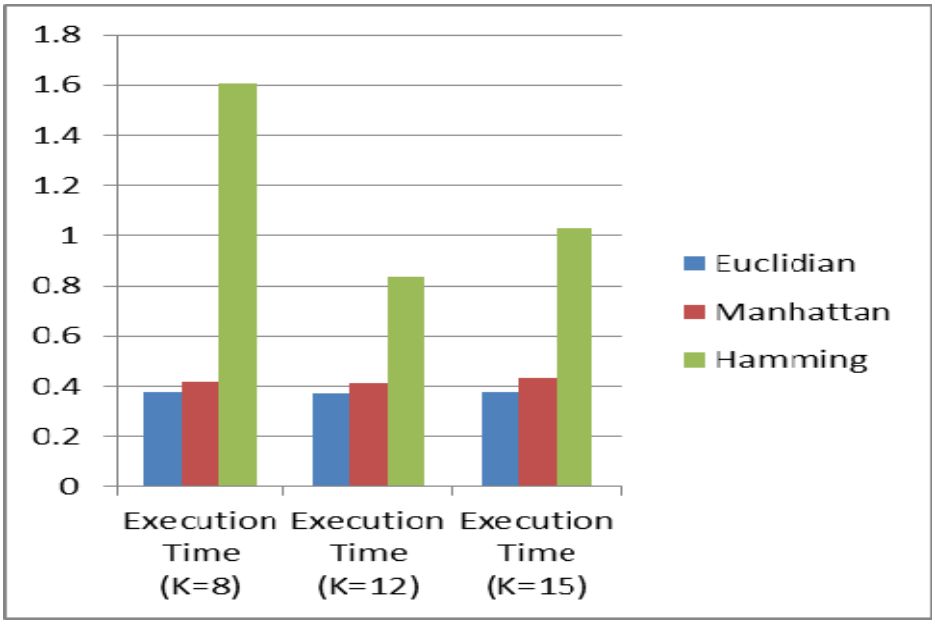

Figure 3.2 Execution time

So it can be said that hamming distance metric is giving best accuracy, however the accuracy is almost equal for all the three distance metric.

One another parameter of performance evaluation used in this study is the execution time. Results of the execution time for different values of K and different distance metric are shown in the table 3.2.. Average of execution time (for all K's) can be computed for the different distance functions and analysis of the different distance functions can be done. A comparative analysis of execution time is shown in figure 3.1. It is observed that for K=8 execution time is 1.6 seconds in case of hamming distance functions.

Not only for K=8, it is observed that execution time for K=12 and K=15 is more as comparison to the Euclidian and Manhattan distance metric. From the result, it can be concluded that with respect to the execution time it is desirable to use Euclidian function.

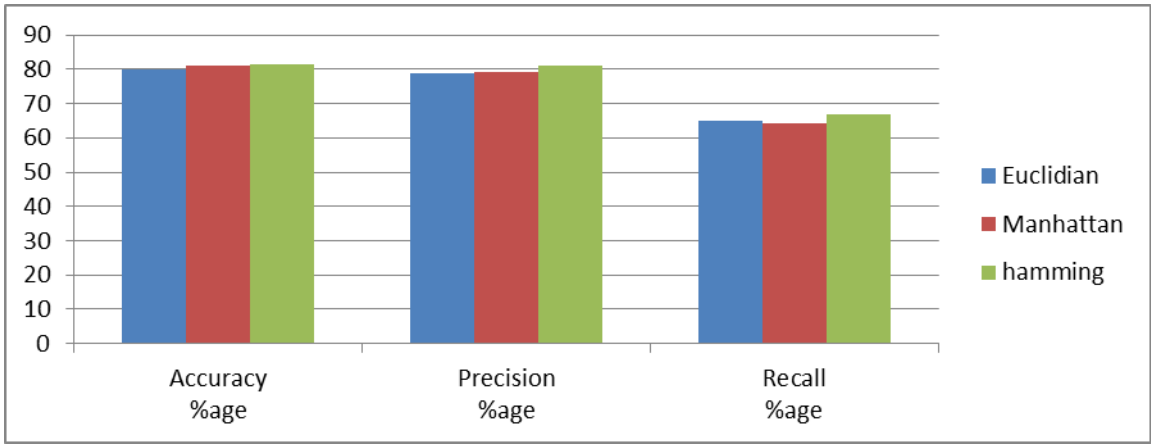

Figure 3.3: Accuracy, Precision and Recall for different distance functions.

Distance Function	Execution Time (K=8)	Execution Time (K=12)	Execution Time (K=15)
Euclidian	0.37400031089782715	0.3698689937591553	0.3779888153076172
Manhattan	0.41501355171203613	0.41440248489379883	0.43491196632385254
Hamming	1.6085307598114014	0.8361709117889404	1.029836654663086

Table 3.2 Execution time for Euclidian, Manhattan and Hamming distance metric

4. Conclusion

Euclidian, Manhattan and Hamming distance functions has been used for the KNN algorithm after implementation it has been observed that KNN algorithm is performing good as comparison Manhattan and Euclidian distance function. However performance all these three function is almost equal as shown in fig 3.3. But in case of execution time KNN algorithm with ecludian distance metric is taking very less time (Table 3.2). So overall, it has been observed that using the Euclidian distance metric is advantageous as comparison to Hamming and Manhattan metric

References

[1] Nakamura, K., Sugawara, M., & Toyama, M. (1991, June). Defect prevention activities and tools. In ICC 91 International Conference on Communications Conference Record (pp. 360-363). IEEE.
[2] Rogerson, S. (2002). The chinook helicopter disaster. IMIS Journal, 12(2).
[3] Murphy, G., & Cubranic, D. (2004, June). Automatic bug triage using text categorization. In Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (pp. 1-6). Citeseer..
[4] Anvik, J., Hiew, L., & Murphy, G. C. (2006, May). Who should fix this bug?. In Proceedings of the 28th international conference on Software engineering (pp. 361-370).
[5] Ahsan, S. N., Ferzund, J., & Wotawa, F. (2009, September). Automatic software bug triage system (bts) based on latent semantic indexing and support vector machine. In 2009 Fourth International Conference on Software Engineering Advances (pp. 216-221). IEEE.
[6] Anvik, J., & Murphy, G. C. (2011). Reducing the effort of bug report triage: Recommenders for development-oriented decisions. ACM Transactions on Software Engineering and Methodology (TOSEM), 20(3), 1-35.
[7] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.
[8] Franz, P. (2005). Bayesian Network Classifiers Versus Selective Formula Not Shown-NN Classifier [J]. Pattern Recognition, 38(1), 1-10.
[9] Han, J. (2006). Jiawei Han & Micheline Kamber, Second Edi.
[10] Hand, D. J. (2007). Principles of data mining. Drug safety, 30(7), 621-622.
[11] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003, November). KNN model-based approach in classification. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 986-996). Springer, Berlin, Heidelberg.
[12] Yong, Z., Youwen, L., & Shixiong, X. (2009). An improved KNN text classification algorithm based on clustering. Journal of computers, 4(3), 230-237.
[13] Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. Journal of machine learning research, 10(2).
[14] Hu, L. Y., Huang, M. W., Ke, S. W., & Tsai, C. F. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. SpringerPlus, 5(1), 1-9.
[15] Prasath, V. B., Alfeilat, H. A. A., Hassanat, A., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., & Salman, H. S. E. (2017). Distance and Similarity Measures Effect on the Performance of K-Nearest Neighbor Classifier--A Review. arXiv preprint arXiv:1708.04321.
[16] Kumar, R., Singla, S., Yadav, R. K., & Kumar, D. (2019). An experimental analysis of various data mining techniques for software bug classification. International Journal of Innovative Technology and Exploring Engineering, 8(8 SpecialIssue 3), 108-113.

[17] Kumar, R., & Verma, R. (2012). Classification rule discovery for diabetes patients by using genetic programming. International Journal of Soft Computing and Engineering, 2(4), 183-185.
[18] Kumar, R., Kapil, A. K., & Bhatia, A. (2012). Modified Tree Classification In Data Mining. Global Journal of Computer Science and Technology.

Authors Profile

Raj Kumar is a PhD Research Scholar at Dept. of CSE, I.K. Gujral Punjab Technical University, Jalandhar (Punjab), India. He received his M.Tech degree in Computer Sc. & Engineering from Chaudhary Devi Lal University, Sirsa(Haryana), India in 2007. He obtained his M.Sc. and MCA degree from Kurukshetra University, Kurukshetra, India. He completed his graduation from Kurukshetra University Kurukshetra. His area of interest includes Data Mining and Machine Learning. He has more than 23 referred journal papers in his credit. He has 14 years of teaching experience. Currently he is working as Assistant Professor in the Dept. of CSE at Chandigarh University, Gharuan(Mohali), India

Dr Sanjay Singla completed Ph.D. from Mahrishi Dayanad University, Rohtak and currently working as Professor and Dean at GGS College of Modern Technology, Punjab. His main research work focuses on Vanets, Network Security, Cloud Security and Privacy, Data Mining and IoT. He has more than 20 years of teaching experience.