Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.201962085
Vol. 6, No. 6, Desember 2019, hlm. 665-670 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

EFISIENSI KLASIFIKASI BIG DATA MENGGUNAKAN IMPROVED NEAREST NEIGHBOR

Aditya Hari Bawono[1], Ahmad Afif Supianto[2]

[1,2] Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]ndaimen@ub.ac.id, [2]afif.supianto@ub.ac.id

(Naskah masuk: 13 Juni 2019, diterima untuk diterbitkan: 07 November 2019)

Abstrak

Klasifikasi adalah salah satu metode penting dalam kajian data mining. Salah satu metode klasifikasi yang populer dan mendasar adalah k-*nearest neighbor* (kNN). Pada kNN, hubungan antar sampel diukur berdasarkan tingkat kesamaan yang direpresentasikan sebagai jarak.. Pada kasus mayoritas terutama pada data berukuran besar, akan terdapat beberapa sampel yang memiliki jarak yang sama namun amat mungkin tidak terpilih menjadi tetangga, maka pemilihan parameter k akan sangat mempengaruhi hasil klasifikasi kNN. Selain itu, pengurutan pada kNN menjadi masalah komputasi ketika dilakukan pada data berukuran besar. Dalam usaha mengatasi klasifikasi data berukuran besar dibutuhkan metode yang lebih akurat dan efisien. Dependent Nearest Neighbor (dNN) sebagai metode yang diajukan dalam penelitian ini tidak menggunakan parameter k dan tidak ada proses pengurutan sampel. Hasil percobaan menunjukkan bahwa dNN dapat menghasilkan efisiensi waktu sebesar 3 kali lipat lebih cepat daripada kNN. Perbandingan akurasi dNN adalah 13% lebih baik daripada kNN.

Kata kunci: *klasifikasi, big data, nearest neighbor, data mining*

BIG DATA CLASSIFICATION EFFICIENCY USING IMPROVED NEAREST NEIGHBOR

Abstract

Classification is one of the important methods of data mining. One of the most popular and basic classification methods is k-nearest neighbor (kNN). In kNN, the relationships between samples are measured by the degree of similarity represented as distance. In major cases, especially on big data, there will be some samples that have the same distance but may not be selected as neighbors, then the selection of k parameters will greatly affect the results of kNN classification. Sorting phase of kNN becomes a computation problem when it is done on big data. In the effort to overcome the classification of big data a more accurate and efficient method is required. Dependent Nearest Neighbor (dNN) as method proposed in this study did not use the k parameters and no sample at the sorting phase. The proposed method resulted in 3 times faster than kNN. The accuracy of the proposed method is13% better results than kNN.

Keywords: *classification, big data, nearest neighbor, data mining*

1. PENDAHULUAN

Big data atau data berukuran besar adalah data yang didefinisikan menggunakan 5 karakteristik yaitu : volume, variety, velocity, veracity, dan value (Minelli, Chambers, & Dhiraj, 2013). Seiring dengan bertambahnya karakteristik tersebut, maka beberapa metode tidak dapat menyimpan dan memproses data. Pada titik tersebut maka data tersebut dapat didefinisikan sebagai big data. Analisis big data didefinisikan sebagai proses pengamatan dan pemahaman terhadap karakteristik data dengan ukuran besar tersebut dengan tujuan mengekstrak pola-pola geometris dan statistik. Banyak aplikasi yang bermasalah dengan big data seperti lalu lintas, geospatial, bisnis, dan game.

Terdapat beberapa cara untuk melakukan analisis big data. Salah satu cara yang populer adalah data mining. Pada data mining terdapat banyak macam teknik seperti unsupervised dan supervised learning. Unsupervised learning digunakan untuk data yang tidak memperhatikan kelas didalamnya, sedangkan supervised learning memperhatikan kelas yang terlampir pada data. Pada supervised learning terdapat beberapa metode yaitu bayesian, neural network, support vector machine, decision tree, dan nearest neighbor. Salah satu metode yang populer adalah nearest neighbor classifier. Nearest neighbor

classifier adalah metode yang menentukan kelas data berdasarkan kedekatan karakteristik dengan data lain. K-nearest neighbor (kNN) adalah nearest neighbor classifier yang melakukan voting penentuan kelas dengan tetangga sebanyak k. Kelemahan kNN adalah ketergantungan pada parameter k yang menyebabkan hasil yang berbeda pada setiap percobaan. Terdapat beberapa penelitian untuk meningkatkan performa nearest neighbor classifier. (Mullick, Datta, Das, & Member, 2018) dan (Bohacik & Zabovsky, 2017) melakukan penelitian terhadap metode learning kNN. Sedangkan studi pada perhitungan jarak telah dilakukan oleh (Pan, Wang, & Ku, 2017) (Neo & Ventura, 2012) (Song, Liang, Lu, & Zhao, 2017). Sedangkan penelitian untuk menggantikan parameter k sebagai tetangga terpilih dilakukan oleh (Ertuğrul & Tağluk, 2017) dengan nama dependent nearest neighbor. Dependent nearest neighbor (dNN) adalah metode dengan menggunakan teori densitas untuk menentukan tetangga. Setiap data pada radius yang telah ditentukan akan langsung menjadi tetangga. Kelebihan dari metode dNN adalah menemukan hasil yang lebih baik dengan cara yang hampir sesederhana kNN namun menghindari kompleksitas perhitungan kNN agar dapat diimplementasikan pada big data.

Penelitian kNN pada big data telah dilakukan oleh (Cattral & Oppacher, 2007) dengan poker hand dataset. Hasil yang diperoleh menggunakan tools Waikato Environment for Knowledge Analysis (WEKA) adalah 50.121% dengan akurasi sebagai metode perhitungan performa. Tujuan dari penelitian ini adalah mengetahui performa metode yang diajukan dNN terhadap metode terdahulu yaitu kNN untuk memproses big data.

2. METODOLOGI

Langkah-langkah algoritma dNN adalah perhitungan dissimilarity, deteksi tetangga, pelabelan, dan evaluasi.

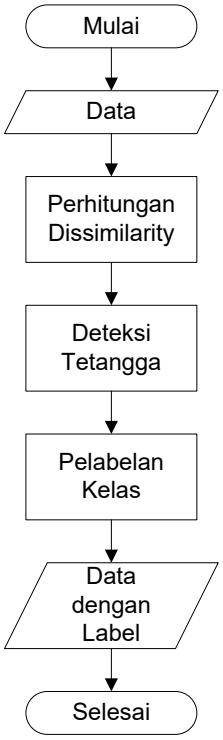
2.1 Perhitungan Dissimilarity

Jarak yang akan dihitung adalah jarak antar obyek data uji dengan seluruh data latih yang ada. Menurut (Han, Kamber, & Pei, 2012), similarity adalah kuantitas yang mencerminkan hubungan antara dua obyek dengan fitur-fitur yang dimiliki. Sedangkan dissimilarity mengukur ketidaksesuaian obyek pada fitur yang dimiliki obyek tersebut. Semakin besar dissimilarity maka semakin besar pula jarak antar kedua obyek. Hubungan antara similarity dan dissimilarity akan ditunjukkan oleh persamaan 1.

$$Dissimilarity(x,y)\begin{cases}0, (x=y)\\1, (x\neq y)\end{cases} \quad (1)$$

$$Similarity = 1 - Dissimilarity$$

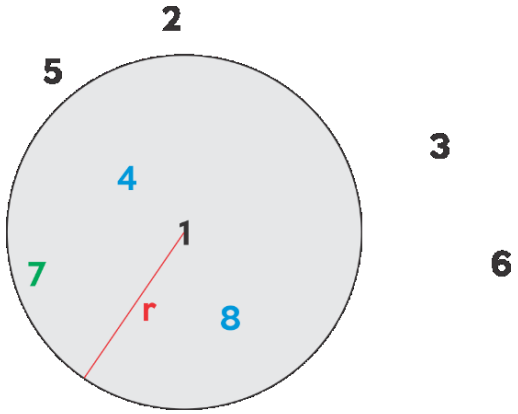
Jika suatu fitur obyek x dan y bernilai sama, maka dissimilarity bernilai 0, sebaliknya jika berbeda, maka dissimilarity bernilai 1. semakin berbeda nilai fitur-fitur antar obyek, maka dapat disimpulkan makin jauh jarak antar kedua obyek tersebut. Semakin jauh jarak sebuah onyek, semakin kecil pula kemungkinan sebuah obyek menjadi tetangga.



Gambar 1 Diagram Alir proses dNN

2.2 Deteksi tetangga

Setelah diketahui jarak antar obyek, maka tentukan tetangga dalam radius r. Ilustrasi penentuan tetangga akan ditunjukkan pada gambar 1.



Gambar 2. Tetangga terpilih

Jika diketahui data 1 adalah data uji, maka data 2 hingga 8 adalah data latih. Dalam radius r, data yang masuk adalah data 4, 8, dan 8, maka hanya ketiga data tersebut yang akan masuk pertimbangan pelabelan. Berbeda dengan kNN, pada kNN jika dipilih k=3, maka yang terpilih adalah 4, 7, dan 8, namun jika k=4, maka 4, 8, 7, dan 5 akan terpilih meskipun 5 berada cukup jauh dari 1. hal ini adalah perbedaan mendasar dari kNN dan dNN.

2.3 Pelabelan

Metode pelabelan yang akan digunakan pada dNN sama dengan kNN yaitu dengan modus kelas tetangga. Kelas yang paling banyak muncul dari tetangga yang terpilih akan ditetapkan sebagai predicted class data uji tersebut. Predicted class akan dibandingkan dengan actual class pada fase evaluasi. Berdasarkan gambar 1, contoh pelabelan akan ditunjukkan oleh tabel 1.

Tabel 1. Distribusi tetangga

Data	Kelas
4	Biru
8	Biru
7	Hijau

Pada Tabel 1 dapat diketahui bahwa kelas biru adalah kelas yang paling sering muncul, maka dari itu predicted class untuk data 1 adalah biru. Jika terdapat jumlah kelas yang sama pada distribusi tetangga, maka pilih salah satu dari kelas-kelas tersebut.

Tabel 2 Distribusi tetangga dengan jumlah anggota kelas yang sama

Data	Kelas
4	Biru
8	Biru
7	Hijau
5	Hijau
2	Merah

Sebagai contoh pada Tabel 2, jika terdapat dua kelas biru dan dua kelas hijau. Pilih salah satu diantara keduanya dengan cara acak.

2.4 Evaluasi

Pada fase evaluasi, metode yang digunakan adalah f-measure. F-measure adalah metode untuk mengukur relevansi suatu kelas dengan hasil prediksi. Sebelum dilakukan perhitungan f-measure, data akan dibentuk dengan confusion matrix terlebih dahulu. Contoh tabel confusion matrix akan ditunjukkan oleh tabel 3.

Tabel 3. Confusion Matrix

		Actual Class	
		Biru	**Lain**
Predicted Class	**Biru**	5 TP	2 FP
	Lain	3 FP	17 TN

Dengan persamaan sebagai berikut :

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F - Measure = \frac{2*Precision*Recall}{Precision+Recall} \quad (4)$$

2.5 Dataset

Dataset yang digunakan adalah Poker hand dataset yang disediakan oleh UCI Machine Learning Repository. Dataset poker hand adalah salah satu dataset yang menantang untuk klasifikasi . Setiap kartu diambil dari dek standar berisikan 52 kartu. Setiap kartu direpresentasikan oleh dua atribut (suit dan rank), untuk total 10 atribut prediktif. Ada satu atribut kelas yang menggambarkan "Poker Hand". Urutan kartu penting dan diperhitungkan. Salah satu pertimbangan dalam penggunaan dataset ini adalah pada penelitian sebelumnya oleh Ertuğrul (2017) tidak terdapat data yang berukuran besar. Berikut adalah informasi tentang Poker Hand dataset.

Tabel 4. Informasi Dataset Poker Hand

Instance	25010 data latih 1000000 data uji
Atribut	10 prediktif 1 goal (kelas)
Missing value	0
Distribusi kelas data latih	
0: Nothing in hand, 12493 instance	
1: One pair, 10599 instance	
2: Two pairs, 1206 instance	
3: Three of a kind, 513 instance	
4: Straight, 93 instance	
5: Flush, 54 instance	
6: Full house, 36 instance	
7: Four of a kind, 6 instance	
8: Straight flush, 5 instance	
9: Royal flush, 5 instance	
Distribusi kelas data uji	
0: Nothing in hand, 501209 instance	
1: One pair, 422498 instance	
2: Two pairs, 47622 instance	
3: Three of a kind, 21121 instance	
4: Straight, 3885 instance	
5: Flush, 1996 instance	
6: Full house, 1424 instance	
7: Four of a kind, 230 instance	
8: Straight flush, 12 instances	
9: Royal flush, 3 instances	

Pada tabel 4 dapat diketahui bahwa dataset poker hand memiliki distribusi yang tidak merata pada setiap kelas. Jeda antara jumlah kelas terkecil dan terbesar terpaut 5-12493 instance untuk data latih dan 3-501209 instance untuk data uji. Perbedaan jumlah instance tiap kelas yang tidak merata amat memungkinkan kesalahan klasifikasi. Sedangkan jumlah data yang besar akan menguji efisiensi metode dalam hal komputasi.

3. ANALISIS HASIL

Skenario pengujian yang dilakukan adalah menjalankan dNN dan kNN dengan parameter masing-masing. Pada kNN, k yang akan digunakan adalah 1 hingga 10 dan pada dNN radius yang akan digunakan adalah 0,1 hingga 1. Ketika pengujian parameter sudah dilakukan, kemudian dilanjutkan dengan pengujian persentase data latih. Data latih yang akan diujikan adalah 25% dan 50% menggunakan parameter terbaik dari pengujian sebelumnya. Dengan demikian hasil yang akan

didapat adalah f-measure dan *running time* dari aplikasi tersebut.

Aplikasi dibuat dengan Visual Studio 2017, dengan bahasa pemrograman C#. Aplikasi dijalankan pada PC dengan spesifikasi i7 haswell, 8GB Ram, VGA R9 285, dan harddisk 1TB.

Berikut ini adalah hasil dari pengujian :

Tabel 5. Perbandingan f-measure data latih 100%

kNN		dNN	
k	**f-measure**	**r%**	**f-measure**
1	0.158342	**1**	0.166176
2	0.158342	**2**	0.166176
3	0.159833	**3**	0.166176
4	0.155109	**4**	0.166176
5	0.155667	**5**	0.166176
6	0.150426	**6**	0.166176
7	0.144385	**7**	0.166176
8	0.141193	**8**	0.166176
9	0.138241	**9**	0.166176
10	0.134343	**10**	0.166176

Tabel 6. Perbandingan f-measure data latih 50%

kNN		dNN	
k	**f-measure**	**r%**	**f-measure**
1	0.129337465	1	0.1475516
2	0.129337465	2	0.1534084
3	0.139181782	3	0.1534084
4	0.147634358	4	0.1534084
5	0.148077823	5	0.1534084
6	0.145030803	6	0.1534084
7	0.138620978	7	0.1534084
8	0.131786414	8	0.1534084
9	0.126279548	9	0.1534084
10	0.122452683	10	0.1534084

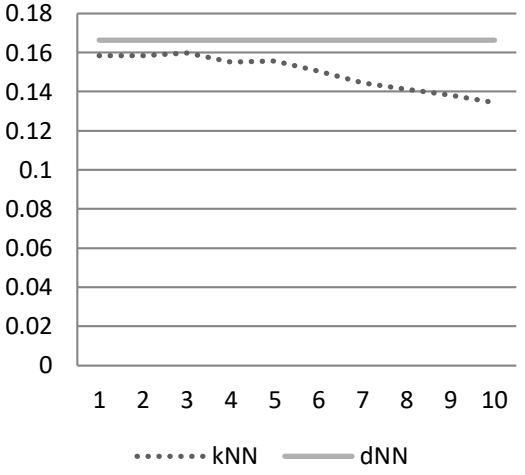
Tabel 7. Perbandingan f-measure data latih 25%

kNN		dNN	
k	**f-measure**	**r%**	**f-measure**
1	0.122727878	1	0.153408
2	0.122727878	2	0.147552
3	0.129692838	3	0.147552
4	0.136140314	4	0.147552
5	0.135835685	5	0.147552
6	0.134581623	6	0.147552
7	0.135206896	7	0.147552
8	0.139272028	8	0.147552
9	0.145190208	9	0.147552
10	0.149933762	10	0.147552

Pada perbandingan performa klasifikasi, dNN lebih unggul pada semua ukuran data latih. Pada Tabel 5 dataset 100% dNN lebih unggul 11,4% dibandingkan kNN, pada Tabel 6 dataset 50% dNN lebih unggul 13%, dan pada Tabel 7 dataset 25% dNN lebih unggul 10%. Hasil tersebut membuktikan bahwa dNN lebih unggul dari segi performa untuk berbagai macam ukuran data. Performa dNN cenderung lebih stabil, namun performa dNN memburuk pada parameter r yang besar, ditunjukkan pada Tabel 8.

Tabel 8. Perbandingan performa dNN pada parameter r

r%	**match**	**f-measure**	**waktu (jam)**
10	527068	0.1661759	2.769603611
20	565175	0.1286764	2.9746616667
30	543550	0.0975356	2.7839280556
40	536494	0.0906148	2.8078872222
50	502045	0.0674232	2.9892316667



Gambar 3. Grafik Perbandingan f-Measure dNN dan kNN

Hasil pada Tabel 8 menunjukkan bahwa dNN baik digunakan pada parameter r kurang dari 10. Penggunaan parameter r lebih dari itu kurang direkomendasikan, karena makin banyak sampel yang tidak relevan masuk menjadi tetangga.

Tabel 9. Perbandingan waktu pada data latih 100%

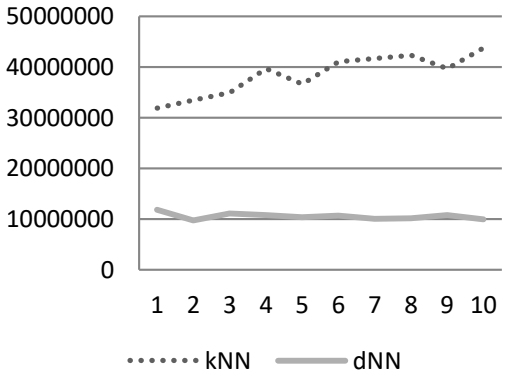
kNN		dNN	
k	**waktu (jam)**	**r%**	**waktu (jam)**
1	8.845525	1	3.277773
2	9.281101	2	2.697624
3	9.689663	3	3.077514
4	11.01999	4	2.997817
5	10.17192	5	2.886226
6	11.39882	6	2.974662
7	11.57956	7	2.783928
8	11.76181	8	2.807887
9	11.01212	9	2.989232
10	12.16621	10	2.769604

Tabel 10. Perbandingan waktu pada data latih 50%

kNN		dNN	
k	**waktu (jam)**	**r%**	**waktu (jam)**
1	2.390273	1	1.382309
2	2.405384	2	1.359527
3	2.493089	3	1.362739
4	2.680436	4	1.387331
5	2.743325	5	1.387698
6	2.766918	6	1.366829
7	2.910411	7	1.387539
8	3.0323	8	1.383196
9	3.075043	9	1.387854
10	3.156808	10	1.358761

Tabel 11. Perbandingan waktu pada data latih 25%

kNN		dNN	
k	**waktu (jam)**	**r%**	**waktu (jam)**
1	1.510839	1	1.521357
2	1.560573	2	1.420285
3	1.650828	3	1.321135
4	1.795513	4	1.657194
5	1.944818	5	1.52047
6	2.017786	6	1.325304
7	2.075959	7	1.529306
8	2.136557	8	1.451279
9	2.279071	9	1.393115
10	2.436634	10	1.466805



Gambar 4. Grafik Perbandingan *Running Time*

dNN cenderung berjalan dengan waktu yang lebih cepat. Pada Tabel 11 dataset 25% kNN menempuh waktu 169% lebih lama dari dNN, pada Tabel 10 kNN lebih 223% lebih lama daripada dNN, sedangkan pada Tabel 9 kNN lebih 339% lebih lama daripada dNN. Hal ini disebabkan karena dNN tidak memerlukan sorting seperti pada kNN. Proses sorting pada big data sebaiknya dihindari, karena sorting cenderung memiliki komputasi kuadratik. Salah satu cara untuk mengatasi masalah sorting pada kNN adalah partial sorting dan sudah diimplementasikan pada penelitian ini. Contoh perhitungan pada 1 data uji dengan 25100 data latih adalah sebagai berikut. Tabel 11 akan menunjukkan perbandingan komputasi dNN dan kNN.

Tabel 12. Perbandingan Komputasi dNN vs kNN

kNN dengan k = 3	**Komputasi**
Scan semua data latih, ambil data latih yang berjarak paling kecil sebanyak k.	25100 + 25099 + 25098 = 75297
Scan k tetangga untuk melihat kelas yang akan dipilih.	3
Jumlah	**75300**
dNN dengan r = 3	**Komputasi**
Scan semua data latih, ambil semua data latih yang berada pada r = 3	25100
Scan semua tetangga untuk melihat kelas mana yang akan dipilih (asumsi tetangga terpilih adalah 50)	50
Jumlah	**25150**

Dari perhitungan pada tabel 12 dapat disimpulkan bahwa dNN lebih efisien karena tidak menggunakan sorting pada waktu memproses. dNN terbukti dapat melakukan performa 75300/25150 = 299.4%, artinya 3 kali lebih cepat daripada kNN. Perbandingan tersebut dapat lebih mencolok ketika data uji yang diproses berjumlah besar. Pada dataset poker hand yang memiliki 1000000 instance, sangat mungkin kNN melakukan komputasi 1000000*75300 pada k = 3. Jika k yang dipilih lebih besar, sudah tentu akan berdampak pada komputasi yang harus dilakukan. dNN tidak memiliki beban sorting seperti kNN menyebabkan stabilitas performa dNN untuk data berukuran besar.

4. KESIMPULAN

Penelitian ini menunjukkan bahwa metode nearest neighbor dapat digunakan untuk melakukan analisis big data. kNN adalah metode nearest neighbor paling sederhana namun dapat menganalisa big data. kNN memiliki kelemahan menentukan parameter k yang terbaik namun untuk menemukan k terbaik adalah proses yang sangat memakan waktu. Untuk mengatasi hal tersebut digunakan metode dNN yang tidak menggunakan parameter k, melainkan menggunakan radius yang berhasil berjalan dengan performa yang amat baik. *Running time* yang dihasilkan metode dNN lebih cepat daripada kNN karena dNN tidak menggunakan sorting secara berlebihan. Meskipun dNN memiliki performa yang baik untuk klasifikasi dataset poker hand, namun hasil yang didapatkan masih bisa ditingkatkan lagi dengan perbaikan metode dan *running time* bisa diturunkan dengan metode lain.

DAFTAR PUSTAKA

BOHACIK, J., & ZABOVSKY, M. 2017. Nearest Neighbor Method Using Non-nested

Generalized Exemplars in Breast Cancer Diagnosis, 40–44.

CATTRAL, R., & OPPACHER, F. 2007. Discovering rules in the poker hand dataset. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation - GECCO '07*, (May), 1870. https://doi.org/10.1145/1276958.1277329

ERTUĞRUL, Ö. F., & TAĞLUK, M. E. 2017. A novel version of k nearest neighbor: Dependent nearest neighbor. *Applied Soft Computing Journal*, *55*, 480–490. https://doi.org/10.1016/j.asoc.2017.02.020

HAN, J., KAMBER, M., & PEI, J. 2012. *Data Mining: Concepts and Techniques*. *San Francisco, CA, itd: Morgan Kaufmann*. https://doi.org/10.1016/B978-0-12-381479-1.00001-0

MINELLI, M., CHAMBERS, M., & DHIRAJ, A. 2013. *Big Data Analytics - Emerging BI and Analitics trends for today's businesses*.

MULLICK, S. S., DATTA, S., DAS, S., & MEMBER, S. 2018. Adaptive Learning-Based k -Nearest Neighbor Classifiers With Resilience to Class Imbalance, 1–13.

NEO, T. K. C., & VENTURA, D. 2012. A direct boosting algorithm for the k-nearest neighbor classifier via local warping of the distance metric. *Pattern Recognition Letters*, *33*(1), 92–102. https://doi.org/10.1016/j.patrec.2011.09.028

PAN, Z., WANG, Y., & KU, W. 2017. A new general nearest neighbor classification based on the mutual neighborhood information. *Knowledge-Based Systems*, *121*, 142–152. https://doi.org/10.1016/j.knosys.2017.01.021

SONG, Y., LIANG, J., LU, J., & ZHAO, X. 2017. An efficient instance selection algorithm for k nearest neighbor regression. *Neurocomputing*, *251*, 26–34. https://doi.org/10.1016/j.neucom.2017.04.018