Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 4, No. 7, Juli 2020, hlm. 2043-2052 http://j-ptiik.ub.ac.id

Implementasi Algoritma *K-Nearest Neighbor* Pada *Database* Menggunakan Bahasa SQL

Shafira Margaretta[1], Issa Arwani[2], Dian Eka Ratnawati[3]

Program Studi Teknologi Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]shaf.margaret@student.ub.ac.id, issa.arwani@ub.ac.id [2], dian_ilkom@ub.ac.id [3]

Abstrak

Meningkatnya jumlah data yang disimpan dalam DBMS, mengakibatkan proses *data mining* kurang efisien diterapkan melalui aplikasi *machine learning*. Secara umum, proses *data mining* dilakukan dengan menerapkan beberapa langkah kerja dimulai dari *data cleaning, data integration, data selection, data transformation* selanjutnya baru dilakukan *data mining.* Pada serangkaian langkah tersebut, terdapat proses dimana data yang disimpan dalam basis data diambil kemudian diekspor untuk dianalisis. Ekspor data dari *database* memungkinkan data menjadi tidak aman. Implementasi algoritma *data mining* pada *database*, merupakan salah satu solusi untuk mengatasi permasalahan tersebut. Selain itu, menerapkan proses *data mining* pada DBMS dapat menjamin keamanan data. Fakultas Ilmu Komputer, Universitas Brawijaya telah memanfaatkan DBMS untuk menyimpan dan mengelola data mahasiswa. Penerapan proses *data mining* untuk prediksi keberhasilan studi pada data akademik Fakultas Ilmu Komputer dilakukan dengan mereplikasi *database.* Dalam mengimplementasikan algoritma *K-Nearest Neighbor* pada *database,* tahap perancangan tabel dilakukan dengan memetakan setiap algoritma menggunakan operasi SQL. Pada tahap perancangan, operasi SQL disimpan kedalam *stored procedure*. Hasil eksekusi *stored procedure* diuji dengan membandingkan hasil sistem DBMS MySQL dan hasil aplikasi WEKA, mendapatkan akurasi hasil klasifikasi sebesar 96,94% menggunakan DBMS MySQL dan 96,84% menggunakan WEKA.

Kata kunci: *database, data mining,* KNN *(K-Nearest Neighbor),* DBMS *(Database Management System),* SQL *(Structured Query Language), stored procedure.*

Abstract

Increasing the amount of data stored in the DBMS, resulting in data mining processes less efficiently implemented through machine learning applications. In general, the data mining process is carried out by implementing a number of work steps starting from data cleaning, data integration, data selection, data transformation and then data mining. In this series of steps, there is a process whereby data stored in a database is taken then exported for analysis. Exporting data from a database allows data to be insecure. Implementation of data mining algorithms in the database, is one solution to overcome these problems. In addition, implementing a data mining process on a DBMS can guarantee data security. The Faculty of Computer Science, Universitas Brawijaya has used the DBMS to store and manage student data. The application of data mining processes to predict the success of studies in the academic data of the Faculty of Computer Science is done by replicating the database. In implementing the K-Nearest Neighbor algorithm in the database, the table design stage is carried out by mapping each algorithm using SQL operations. At the design stage, SQL operations are stored in stored procedures. The results of stored procedure execution were tested by comparing the results of the MySQL DBMS system and the results of the WEKA application, obtaining an accuracy of the classification results of 96,94% using a MySQL DBMS and 96,84% using a WEKA.

Keywords: *database, data mining, KNN (K-Nearest Neighbor), DBMS (Database Management System), SQL (Structured Query Language), stored procedure.*

1. PENDAHULUAN

. *K-Nearest Neighbor* (KNN) adalah salah satu algoritma paling populer dalam teknik *data mining*. Algoritma *K-Nearest Neighbor* (KNN) merupakan algoritma yang sederhana dan mudah diterapkan (Agrawal, 2014). Meskipun sederhana, *K-Nearest Neighbor* (KNN) mampu melakukan tugas secara efektif, salah satunya untuk mengatasi masalah klasifikasi. Cara kerja dari *K-Nearest Neighbor* (KNN) adalah melakukan klasifikasi dengan melihat kemiripan objek dengan kategori tertentu berdasarkan kelas terdekat. Hasil klasifikasi yang didapatkan, selanjutnya akan diambil kesimpulan untuk memprediksi suatu nilai. Secara umum, teknik klasifikasi dapat dilakukan dengan menggunakan aplikasi *machine learning*. Klasifikasi dilakukan dengan cara menjalankan *library* tertentu terlebih dahulu pada aplikasi *machine learning*, selanjutnya dilakukan penggalian data dengan memasukan *file dataset* untuk analisis data. Namun, hal ini dapat mengakibatkan proses klasifikasi membutuhkan waktu dan *resource* lebih, jika jumlah data yang diolah berskala besar.

Untuk meningkatkan efisisensi waktu dan *resource* maka proses klasifikasi sebaiknya dapat dilakukan dalam basis data. Proses klasifikasi pada sistem basis data dapat menangani *volume* data secara efisien. Implementasi algoritma penggalian set data pada *Relational Database Management System (RDBMS)* menggunakan *stored procedure* dan menggabungkan algoritma dengan tingkat efisiensi tinggi menjadikan teknik penggalian data lebih cepat (Yong, 2007). Selain efisien dan menyediakan ruang penyimpanan, *DBMS* memberikan kemudahan mengakses data dalam jumlah besar secara cepat. Faktor keamanan data juga menjadi pertimbangan untuk melakukan proses klasifikasi dalam *DBMS.*

Jumlah data akademik pada Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya yang semakin meningkat, dapat dimanfaatkan sebagai pengambil keputusan untuk menentukan keberhasilan studi mahasiswa. Dalam menentukan keberhasilan studi, klasifikasi data akan dibagi menjadi kelas evaluasi dan kelas tidak terkena evaluasi, diambil berdasarkan pengelompokan nilai indeks prestasi kumulatif (IPK) dan satuan kredit semester (SKS). Data akademik FILKOM Universitas Brawijaya disimpan dalam basis data, sehingga proses klasifikasi dilakukan dalam *Database Management System (DBMS).*

Pada penelitian ini difokuskan perancangan dan pengimplementasian algoritma *K-Nearest Neighbor* pada basis data MySQL menggunakan operasi *stored procedure*. Klasifikasi diujicobakan dengan mereplikasi basis data akademik mahasiswa FILKOM Universitas Brawijaya untuk menentukan keberhasilan studi mahasiswa berdasarkan IPK dan sks. Uji coba dilakukan dengan membandingkan kesesuaian hasil klasifikasi algoritma KNN pada MySQL dan aplikasi *machine learning* berdasarkan perhitungan akurasi terhadap variasi nilai *k* yang ditentukan.

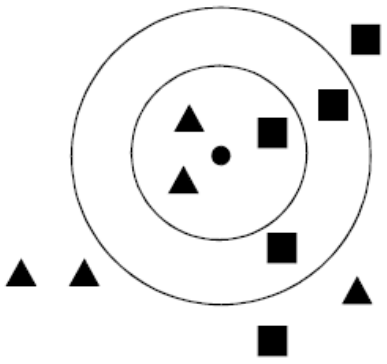
2. DASAR TEORI

2.1 Evaluasi Akademik

Evaluasi keberhasilan studi mahasiswa dilakukan untuk mendapatkan informasi mengenai seberapa jauh mahasiswa telah mencapai tujuan yang dirumuskan dalam kurikulum melalui penyelenggaraan ujian dan penilaian tugas. Menurut buku Pedoman Pendidikan Universitas Brawijaya tahun 2016-2017, dijelaskan bahwa evaluasi keberhasilan studi dilakukan pada setiap akhir semester, berdasarkan mata kuliah yang diambil oleh mahasiswa.

2.2 *K-Nearest Neighbor* (KNN)

K- Nearest Neighbor merupakan metode klasifikasi objek berdasarkan tingkat kemiripan atau kesamaan pada sampel data *training* (data latih) melalui klasifikasi jarak maupun jumlah tetangga terdekat. Tetangga terdekat ditentukan oleh fungsi matrik, yaitu berdasarkan nilai *k* yang didefinisikan dari titik pada sampel data (Phyu T. N., 2009).



Gambar 1. Contoh Klasifikasi *K-Nearest Neighbor*.

Klasifikasi *k nearest neighbor* merupakan algoritma paling sederhana, klasifikasi dilakukan dengan memberikan

setiap vektor fitur input ke kelas yang ditunjukkan oleh label vektor terdekat dalam sampel sumber. Secara umum, *k nearest neighbor* memetakan vektor fitur ke kelas pola yang paling sering muncul diantara tetangga terdekat *k* (Kulkarni, Lugosi, & Venkatesh, 1998). Nilai k dapat dipengaruhui oleh besar atau kecil suatu fitur yang diberikan. Secara sederhana, penentuan nilai k dapat dilakukan dengan menggunakan persamaan k=√n, dimana nilai n merupakan jumlah sampel data pada data latih.

2.3 *Database Management System* (DBMS)

Database Management System (DBMS) merupakan sistem perangkat lunak yang memungkinkan akses untuk data pada *database*. DBMS juga menyediakan kemudahan dan metode efektif untuk melakukan pendefinisian, penyimpanan dan pengambilan informasi pada basis data. Selain itu, DBMS memberikan pengelolaan terpusat terhadap *database*, sehingga mencegah pengguna tidak sah untuk mengakses data dan memastikan privasi data (Gunjai 2003).

2.4 *Structured Query Language (SQL)*

Structured Query Language (SQL) merupakan bahasa standar terdiri dari kumpulan instruksi, digunakan untuk berinteraksi dengan *relational database*. SQL memiliki tiga komponen utama yaitu *Database Definition Language* (DDL), *Database Manipulation Language* (DML), *Data Control Language* (DCL). SQL terdiri dari kumpulan pernyataan dimana semua program dan *users* dapat mengakses data melalui basis data.

2.5 *Stored Procedure*

Hampir semua *Database Management System* (DBMS) *mendukung stored procedure. Storde procedure* merupakan sub program yang dapat disusun secara terpisah dan disimpan secara permanen dalam database, dan siap dijalankan. Penyusunan dilakukan sekali dam disimpan melalui *data dictionary*, yang disebut sebagai *schema object*, yang dapat di refrensikan pada beberapa jumlah aplikasi yang terhubung dengan database (Yong, 2007).

2.6 Pengujian Akurasi

Pengujian akurasi merupakan tahapan untuk mengetahui tingkat keberhasilan dalam melakukan klasifikasi. Untuk menentukan tingkat akurasi, dapat dilakukan dengan cara menghitung hasil pembagian dari jumlah klasifikasi yang benar terhadap jumlah seluruh data (Han & Kamber, 2000). Berikut merupakan persamaan dalam menentukan akurasi, ditunjukkan pada persamaan (1)

$$\text{Akurasi } (\%) = \frac{\sum data\ uji\ benar}{\sum total\ data\ uji} \times 100\% \quad (1)$$

3. PEMETAAN ALGORITMA KNN di DBMS

Dalam sub bab ini ditunjukkan mengenai pemetaan algoritma *K-Nearest Neighbor* terhadap kebutuhan tabel dalam DBMS yang dijabarkan melalui perhitungan manual. Pemetaan algoritma KNN terhadap tabel terdiri dari 7 tahapan.

Tabel 3.1. Pemetaan Algoritma Terhadap Kebutuhan Tabel

Tahap	Algoritma	Kebutuhan Tabel	
		Nama	Kolom
1	Membaca set data latih horizontal	dtTrainH	i,x_1, x_2, x_3, x_4, x_5, x_6
2	Mengubah format tabel data latih horizontal menjadi vertikal	dtTrainV	i_x, d_x, v_x
3	Membaca set data uji vertikal	dtTestH	i,x_1, x_2, x_3, x_4, x_5
4	Mengubah format tabel data uji horizontal menjadi vertikal	dtTestV	i_y, d_y, v_y
5	Menghitung jarak	dtHitung	i, d_x, v_x, d_y, v_y, h
6	Mengurutkan data secara ascending	dtTestV	i, h, u
7	Mengambil nilai data berdasarkan nilai k	dtHitung	i, h, u

3.1. Pemetaan Algoritma Dalam Perhitungan Manual

Tabel 3.2. Contoh Tabel Data Akademik

i	x1	x2	x3	x4	x5	x6
1	201637	1	19	19	3,21	T
2	201732	1	18	18	2,81	T
3	201612	1	19	18	1,65	E

Dalam perhitungan manual ini akan diberikan 3 sampel data akademik mahasiswa sebagai contoh yang mana akan dipecah menjadi 2 jenis data yaitu data latih dan data uji dengan perbandingan jumlah 70% : 30%. Pada Tabel 3.2 ditunjukkan data dengan kolom i bernilai 1-2 merupakan data latih, dan i bernilai 3 merupakan data uji. Data yang digunakan dalam perhitungan manual ini adalah data evaluasi mahasiswa pada tahun pertama.

3.2. Tahap 1 Membaca Set Data Latih

Pada tahap 1 merupakan algoritma membaca set data latih, Tabel dtTrainH merupakan tabel data horizontal menyimpan data latih. Berikut merupakan contoh struktur tabel dtTrainH.

Dimana kolom i = indeks, x1= id mahasiswa, x3=sks beban, x4=sks lulus, x5=ipk, x6=klasifiasi.

Tabel 3.3. Contoh Tabel dtTrainH

i	x1	x2	x3	x4	x5	x6
1	201637	1	19	19	3,21	T
2	201732	1	18	18	2,81	T

3.3. Tahap 2 Membaca Set Data Latih Vertikal

Pada tahap 2 merupakan algoritma membaca set data latih, Tabel dtTrainH merupakan tabel data horizontal menyimpan data latih dirubah strukturnya menjadi tabel vertikal diberi dengan nama dtTrainV. Berikut merupakan contoh struktur tabel dtTrainV.

Dimana kolom ix = indeks tabel dtTrainV, dx= dimensi, vx=value

Tabel 3.4. Contoh Tabel dtTrainV

ix	dx	vx
1	x1	201637
1	x2	1
1	x3	19
1	x4	19
1	x5	3,21
2	x1	201732
2	x2	1
2	x3	18
2	x4	18
2	x5	2,81

3.4. Tahap 3 Membaca Set Data Uji

Pada tahap 3 merupakan algoritma membaca set data uji, Tabel dtTestH merupakan tabel data horizontal menyimpan data uji . Berikut merupakan contoh struktur tabel dtTestH.

Dimana kolom i = indeks, x1= id mahasiswa, x3=sks beban, x4=sks lulus, x5=ipk.

Tabel 3.5. Contoh Tabel dtTrainV

i	x1	x2	x3	x4	x5
3	201612	1	19	18	1,65

3.5. Tahap 4 Membaca Set Data Uji Vertikal

Pada tahap 4 merupakan algoritma membaca set data uji, Tabel dtTestH merupakan tabel data horizontal menyimpan data uji dirubah strukturnya menjadi tabel vertikal, diberi dengan nama dtTestV. Berikut merupakan contoh struktur tabel dtTestV.

Dimana kolom ix = indeks tabel dtTestV, dx= dimensi, vx=value.

Tabel 3.6. Contoh Tabel dtTestV

iy	dy	vy
3	x1	201612
3	x2	1
3	x3	19
3	x4	18
3	x5	1,65

3.6. Tahap 5 Menghitung Jarak Terdekat

Pada tahap 5 terdapat tabel dtHitung. Tabel dtHitung merupakan tabel yang menyimpan nilai dari tabel dtTrainV dan dtTestV. Gabungan tabel tersebut digunakan untuk menyimpan hasil nilai jarak berdasarkan data yang diuji.

Hasil nilai jarak menggunakan perhitungan *Euclidean*, pada tahap ini perhitungan dilakukan dengan menemukan nilai kuadrat terlebih dahulu. Nilai hasil perhitungan kuadrat berdasarkan persamaan (2) disimpan kedalam kolom h.

$$h = (vx - vy)^2 \qquad (2)$$

Tabel 3.6. Contoh Tabel dtHitung

i	dx	vx	iy	dy	xy	h
1	x2	1	3	x2	1	0
1	x3	19	3	x3	19	0
1	x4	19	3	x4	18	1
1	x5	3,21	3	x5	1,65	2,4336
2	x2	1	3	x2	1	0
2	x3	18	3	x3	19	1
2	x4	18	3	x4	18	0
2	x5	2,81	3	x5	1,65	1,3456

3.7. Tahap 6 Mengurutkan Data Secara Ascending

Pada tahap 6 merupakan algoritma mengurutkan data secara *ascending.* Pada kolom h hasil perhitungan jarak yang diperoleh masing-masing indeks dijumlahkan seluruhnya kemudian diakar. Setelah didapatkan hasil jumlah perhitungan diurutkan berdasarkan nilai terkecil. Persamaan untuk menjumlahkan semua nilai dengan indeks menggunakan persamaan (3).

$$h = \sqrt{(h(dx)_{x2} + h(dx)_{x3} + h(dx)_{x4} + h(dx)_{x5})} \quad (3)$$

Tabel 3.6. Contoh Tabel dtHitung Jarak

i	h	u
2	1,531	1
1	1,852	2

3.8. Tahap 7 Mengambil Data Berdasarkan Nilai K

Pada tahap 7 merupakan algoritma untuk mengambil data berdasarkan nilai k. Jika dalam algortima ini diberikan nilai k=2 maka nilai tabel yang muncul adalah i ke 2 dan 1 dengan hasil x6 atau klasifikasi bernilai T, artinya tidak evaluasi.

Tabel 3.6. Contoh Tabel dtHitung Hasil Klasifikasi

i	h	u	x6
2	1,531	1	T
1	1,852	2	T

4. PERANCANGAN SQL PEMETAAN ALGORITMA KNN PADA DBMS

Perancangan pada penelitian ini menggunakan serangkaian kumpulan perintah operasi pada SQL, meliputi perintah *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML).

4.1. Perancangan DDL dtTrainH

Perancangan SQL untuk membuat tabel dengan nama dtTrainH. Tabel dtTrainH merupakan tabel yang menyimpan informasi mengenai data latih.

```
CREATE TABLE dtTrainH (
i INT (10),
x1 INT (10),
x2 INT (10),
x3 INT (20),
x4 INT (20),
x5 FLOAT (3,2),
x6 VARCHAR (20)
);
```

4.2. Perancangan DDL dtTrainV

Perancangan SQL untuk membuat tabel dengan nama dtTrainV. Tabel dtTrainV merupakan tabel yang menyimpan informasi mengenai data latih.

```
CREATE TABLE dtTrainV (
ix INT (10),
dx VARCHAR (10),
vx DOUBLE (22,2)
);
```

4.3. Perancangan DDL dtTestH

Perancangan SQL untuk membuat tabel dengan nama dtTestH. Tabel dtTestH merupakan tabel yang menyimpan informasi mengenai data uji.

```
CREATE TABLE dtTestH (
i INT (10),
x1 INT (10),
x2 INT (10),
x3 INT (20),
x4 INT (20),
x5 FLOAT (3,2),
x6 VARCHAR (20)
);
```

4.4. Perancangan DDL dtTestV

Perancangan SQL untuk membuat tabel dengan nama dtTestV. Tabel dtTestV merupakan tabel yang menyimpan informasi mengenai data uji.

```
CREATE TABLE dtTestV (
ix INT (10),
dx VARCHAR (10),
vx DOUBLE (22,2)
);
```

4.5. Perancangan DML dtTrainH

Peracangan DML pada tabel dtTrainH bertujuan untuk memasukkan sumber data pada tabel yang telah didefinisiskan.

Terdapat dua pilihan untuk memasukan sumber data, pertama melalui *import file* berekstensi CSV atau menggunakan perintah SQL yaitu INSERT INTO.

```
INSERT INTO dtTrainH (i, x1, x2, x3,
x4, x5)
VALUES
(1, 201610443,1,39,39,3.46),
(2, 20171033,1,39,33,2.61),
// sampai data ke-n
;
```

4.6. Perancangan DML dtTrainV

Peracangan DML pada tabel dtTrainV bertujuan untuk memasukkan data kedalam tabel dtTrainV, data yang diambil berdasarkan tabel dtTrainH.

```
INSERT INTO dtTrainV
SELECT i AS ix, x1, 'x2' AS dx, x2 AS
vx FROM dtTrainH
UNION SELECT i AS ix, x1, 'x3' AS dx,
x3 AS vx FROM dtTrainH
UNION SELECT i AS ix, x1, 'x4' AS dx,
x4 AS vx FROM dtTrainH
UNION SELECT i AS ix, x1, 'x5' AS dx,
x5 AS vx FROM dtTrainH
GROUP BY i
ORDER BY 1;
```

4.5. Perancangan DML dtTestH

Peracangan DML pada tabel dtTestH bertujuan untuk memasukkan sumber data pada tabel yang telah didefinisiskan. Terdapat dua pilihan untuk memasukan sumber data, pertama melalui *import file* berekstensi CSV atau menggunakan perintah SQL yaitu INSERT INTO.

```
INSERT INTO dtTrainH (i, x1, x2, x3,
x4, x5)
VALUES
(1,2016101,1,39,39,3.73),
(2,2016401141,1,40,40,2.85),
// sampai data ke-n
;
```

4.6. Perancangan DML dtTestV

Peracangan DML pada tabel dtTestV bertujuan untuk memasukkan data kedalam tabel dtTrainV, data yang diambil berdasarkan tabel dtTestH.

```
INSERT INTO dtTestV
 SELECT i AS iy, y1, 'y2' AS dy, y2 AS
 vy FROM dtTestH
 UNION SELECT i AS iy, y1, 'y3' AS dy,
 y3 AS vy FROM dtTestH
 UNION SELECT i AS iy, y1, 'y4' AS dy,
 y4 AS vy FROM dtTestH
 UNION SELECT i AS iy, y1, 'y5' AS dy,
 y5 AS vy FROM dtTestH
  GROUP BY i
ORDER BY 1;
```

4.6. Perancangan DML Perhitungan Jarak

Peracangan DML bertujuan untuk melakukan operasi perhitungan jarak menggunakan persamaan *Euclidean* disimpan kedalam tabel bernama dtHitung. Data untuk perhitungan diambil melalui tabel dtTrainV dan dtTestV.

```
INSERT INTO dtHitung
SELECT ix, dx, vx, iy, dy, vy,
(POWER ((vx-vy),2)) AS h
FROM dtTrainV, dtTestV
WHERE dx = dy AND iy = 1;
```

4.6. Perancangan DML Menentukan Nilai K

Perancangan DML dalam menentukan nilai k bertujuan untuk mengetahui kategori klasifikasi pada data uji. Kategori klasifikasi terdiri dari kelas “EVALUASI” dan “TIDAK EVALUASI”.

```
SELECT ix, iy,SQRT (SUM(h)) dist,
x6 as class
FROM dtHitung INNER JOIN
dtTrainHK
ON dtHitung.`ix` = dtTrainHK.`i`
WHERE iy =1
GROUP BY ix
ORDER BY dist ASC
LIMIT 10;
```

5. IMPLEMENTASI DML DALAM *STORED PROCEDURE*

Pada implementasi DML dalam *stored procedure* akan dijelaskan mengenai pendefinisian dan pemanggilan prosedur menggunakan pernyataan SQL. Setiap *stored procedure* didefinisikan berdasarkan pemetaan tahapan algoritma *K-Nearest Neighbor*.

5.1. *Stored Procedure* Membaca Set Data Latih

Stored procedure untuk membaca *dataset* data latih dibuat dengan nama prosedur getDataTrainH() dan getDataTrainV(). Tujuan dari implementasi prosedur adalah untuk memudahkan dalam membaca set data latih tanpa mengeksekusi seluruh *query*. Sehingga *stored procedure* dapat meningkatkan performa basis data.

Stored procedure juga memungkinkan untuk dipanggil setiap saat dibutuhkan.

```
DELIMITER //
CREATE PROCEDURE getDataTrainH()
BEGIN
 SELECT * FROM dtTrainH;
END //
DELIMITER ;

CALL getDataTrainH();
```

Berikut merupakan *stored procedure* untuk memanggil dtTrainV.

```
DELIMITER //
CREATE PROCEDURE getDataTrainV()
BEGIN
  SELECT i AS ix, x1, 'x2' AS dx, x2
AS vx FROM dtTrainH
  UNION SELECT i AS ix, x1, 'x3' AS
dx, x3 AS vx FROM dtTrainH
  UNION SELECT i AS ix, x1, 'x4' AS
dx, x4 AS vx FROM dtTrainH
  UNION SELECT i AS ix, x1, 'x5' AS
dx, x5 AS vx FROM dtTrainH
  GROUP BY i
  ORDER BY 1;
END //
DELIMITER ;

CALL getDataTrainV();
```

5.2. *Stored Procedure* Membaca Set Data Uji

Stored procedure untuk membaca *dataset* data uji dibuat dengan nama prosedur getDataTestH() dan getDataTestV(). Tujuan dari implementasi prosedur adalah untuk memudahkan dalam membaca set data latih tanpa mengeksekusi seluruh *query*. Sehingga *stored procedure* dapat meningkatkan performa basis data. *Stored procedure* juga memungkinkan untuk dipanggil setiap saat dibutuhkan.

```
DELIMITER //
CREATE PROCEDURE getDataTestH()
BEGIN
 SELECT * FROM dtTestH;
END //
DELIMITER ;
CALL getDataTestH();
```

Berikut merupakan *stored procedure* untuk memanggil dtTestV.

```
DELIMITER //
CREATE PROCEDURE getDataTestV()
BEGIN
 SELECT i AS iy, y1, 'y2' AS dy, y2 AS
 vy FROM dtTestH
 UNION SELECT i AS iy, y1, 'y3' AS dy,
 y3 AS vy FROM dtTestH
 UNION SELECT i AS iy, y1, 'y4' AS dy,
 y4 AS vy FROM dtTestH
 UNION SELECT i AS iy, y1, 'y5'
 AS dy, y5 AS vy FROM dtTestH
 GROUP BY i
 ORDER BY 1;
 END //
DELIMITER ;

CALL getDataTestV();
```

5.3. *Stored Procedure* Menampilkan Hasil Perhitungan Jarak *Euclidean*

Stored procedure untuk menampilkan hasil nilai perhitungan *Euclidean* dibuat dengan nama prosedur getHasilEuclidean().

```
DELIMITER //
CREATE                    PROCEDURE
getHasilEuclidean()
BEGIN
 SELECT ix, dx, vx, iy, dy, vy,
 (POWER ((vx-vy),2)) AS h
 FROM dtTrainV, dtTestV
 WHERE dx = dy
 AND iy = 1;
END //
DELIMITER ;

CALL getHasilEuclidean();
```

5.3. *Stored Procedure* Menentukan Nilai K

Stored procedure untuk memanggil hasil klasifikasi dibuat dengan nama prosedur getHasilKlasifikasi(). Menentukan nilai k dilakukan dengan menampilkan beberapa jumlah tetangga terdekat berdasarkan jarak terdekat dari perhitungan *Euclidean*.

```
DELIMITER //
CREATE PROCEDURE
getHasilKlasifikasi()
BEGIN
 SELECT ix, iy, SQRT(SUM(h))
 dist, x6 AS class
 FROM dtHitung INNER JOIN
 dtTrainHK
 ON dtHitung.`ix` = dtTrainHK.`i`
 WHERE iy = 1
 GROUP BY ix
 ORDER BY dist ASC
 LIMIT 10;
END //
DELIMITER ;

CALL getHasilKlasifikasi();
```

6. PENGUJIAN

Pengujian dilakukan dengan menerapkan Algoritma KNN menggunakan aplikasi WEKA dan aplikasi DBMS MySQL. Tujuan pengujian pada aplikasi WEKA diharapkan dapat menjadi

pertimbangan bahwa penerapan prediksi *K-Nearest Neighbor* (KNN) pada SQL di DBMS MySQL mendapatkan hasil yang *valid*.

Data yang digunakan dalam pengujian adalah data akademik mahasiswa Fakultas Ilmu Komputer untuk tahun akademik 2016/2017 sampai dengan 2018/2019. Terdapat 2 skenario pengujian yang dilakukan yaitu :

1. Perbandingan hasil klasifikasi algoritma KNN menggunakan aplikasi WEKA dan SQL-KNN pada DBMS MySQL.
2. Pengujian untuk menentukan nilai K yang terbaik untuk hasil klasifikasi pada data akademik tahun 2016/2017 sampai dengan 2018/2019

6.1. Perbandingan Hasil Klasifikasi Algoritma KNN MMenggunakan Aplikasi WEKA dan DBMS MySQL

Nilai akurasi pada setiap data akademik melalui aplikasi WEKA, diperoleh ketika memasukan set data uji dan set data latih pada aplikasi tersebut. Sedangkan nilai akurasi pada DBMS MySQL diperoleh dengan menghitung jumlah data benar per total data, dikali 100%. Hasil KNN ditunjukkan dengan hasil perhitungan jarak terdekat pada urutan nilai k dan klasifikasi kelas untuk nilai k. Nilai k yang digunakan adalah k=√(data latih). Terdapat dua kategori kelas "Tidak" dan kelas "Eval".

Adapun perbandingan prosentase akurasi hasil klasifikasi algoritma KNN menggunakan aplikasi WEKA dan SQL-KNN pada DBMS MySQL ditunjukkan pada Tabel 6.1.

Tabel 6.1 Perbandingan Hasil Klasifikasi Aplikasi WEKA dan DBMS MySQL

Perbandingan Hasil KNN Setiap Data Uji				
Data Akademik	**Jumlah Data**	**Nilai K**	**Prosentase Akurasi Hasil Klasifikasi (%)**	
			WEKA	**DBMS MySQL**
2016 /2017	1151	28	95,95	96,24
2017 /2018	694	22	98,08	98,08
2018 /2019	665	21	96,5	96,5
Rata-rata %			96,84	96,94

6.2. Pengujian Akurasi Masing-masing Nilai K

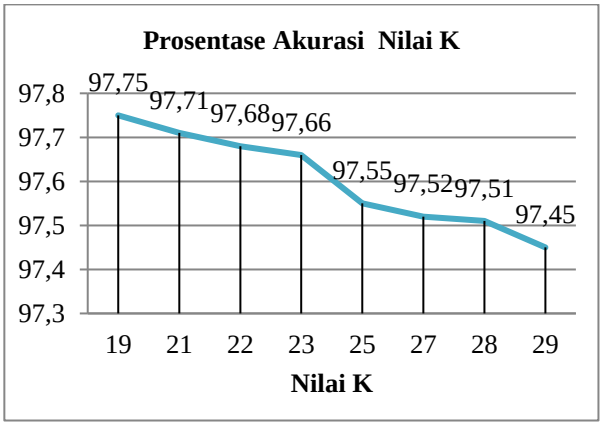
Pengujian akurasi ditentukan dengan menghitung prosentase jumlah klasifikasi benar terhadap seluruh data uji pada setiap nilai k. Dimana klasifikasi benar merupakan hasil klasifikasi kelas setiap indeks pada SQL bernilai sama dengan data sebenarnya. Adapun hasil pengujian akurasi yang diperoleh ditunjukan pada Tabel 6.2.

Tabel 6.2 Perbandingan Akurasi Hasil Klasifikasi Setiap Nilai K

Perbandingan Hasil KNN Setiap Data Uji				
Nilai K	**Prosentase Akurasi Hasil Klasifikasi Variasi Nilai K**			
	2016 /2017	**2017 /2016**	**2018 /2019**	**Rata – Rata (%)**
K=19	97,25	98,54	97,47	97,75
K=21	97,27	98,47	97,38	97,71
K=22	97,28	98,52	97,25	97,68
K=23	97,31	98,56	97,11	97,66
K=25	97,25	98,56	96,82	97,55
K=27	97,27	98,64	96,65	97,52
K=28	97,25	98,67	96,61	97,51
K=29	97,14	98,68	96,53	97,45

6.3. Analisis Hasil Pengujian Akurasi

Hasil grafik pengujian akurasi terhadap variasi nilai k, menunjukkan bahwa nilai k mempengaruhi akurasi. Berdasarkan grafik, prosentase nilai akurasi k mengalami perubahan yang ditunjukkan dengan pola menurun pada setiap nilai k.



Gambar 6.1 Grafik Prosentase Akurasi Nilai K

Tingkat akurasi tertinggi ditunjukkan dengan grafik pada nilai k = 19 dengan prosentase sebesar 97,75%. Sehingga pada nilai k = 19 menjadi nilai k terbaik dalam menentukan kelas sebagai prediksi keberhasilan studi mahasiswa untuk tahun akademik 2016/2017 sampai dengan tahun akademik 2018/2019 pada data akademik mahasiswa Fakultas Ilmu Komputer.

6.4. Pengujian Performa Proses KNN Dalam DBMS MySQL

Pengujian performa proses KNN dalam DBMS MySQL pada Tabel 6.3 menunjukkan perolehan hasil pengujian performa secara keseluruhan berdasarkan jumlah waktu seluruh tahapan proses KNN untuk masing-masing data akademik. Dari hasil pegujian performa yang telah diperoleh, untuk data akademik tahun 2016/2017 memiliki total waktu rata-rata tertinggi dengan waktu 0,525 detik. Perolehan waktu tertinggi pada data akademik tahun 2016/2017 dikarenakan jumlah baris data yang dieksekusi paling memiliki jumlah baris data banyak dibandingkan dengan data akademik lainnya. Berdasarkan hasil tersebut, dapat dikatakan bahwa performa proses KNN dalam DBMS MySQL memerlukan waktu yang relatif tergantung dengan jumlah baris data yang dieksekusi.

Tabel 6.3 Hasil Pengujian Performa Proses KNN Dalam DBMS MySQL Secara Keseluruhan

Data Akademik Tahun	Proses KNN	Total Waktu (Detik)	Rata-rata (Detik)
2016/ 2017	Membaca Set Data Latih	0,085	0,525
	Membaca Set Data Latih Vertikal	0,133	
	Membaca Set Data Uji	0,165	
	Membaca Set Data Uji Vertikal	0,193	
	Menampilkan Hasil Perhitungan *Euclidean*	0,272	
	Menampilkan Hasil Klasifikasi	2,303	
2017/ 2018	Membaca Set Data Latih	0,028	0,268
	Membaca Set Data Latih Vertikal	0,086	
	Membaca Set Data Uji	0,088	
	Membaca Set Data Uji Vertikal	0,090	
	Menampilkan Hasil Perhitungan *Euclidean*	0,132	
	Menampilkan Hasil Klasifikasi	1,186	
2018/ 2019	Membaca Set Data Latih	0,082	0,286
	Membaca Set Data Latih Vertikal	0,117	
	Membaca Set Data Uji	0,122	
	Membaca Set Data Uji Vertikal	0,140	
	Menampilkan Hasil Perhitungan *Euclidean*	0,178	
	Menampilkan Hasil Klasifikasi	1,077	

7. KESIMPULAN

Berdasarkan hasil pengujian akurasi yang dilakukan dengan membagi data menjadi data latih dan data uji dengan perbandingan 70% : 30% pada data akademik tahun 2016/2017 sampai tahun 2018/2019, menghasilkan prosentase akurasi dengan rata-rata sebesar 96,8466% untuk hasil klasifikasi pada aplikasi WEKA dan 96,9429% untuk hasil klasifikasi pada DBMS MySQL. Pada hasil klasifikasi masing-masing aplikasi tersebut, selisih akurasi antara keduanya hanya sebesar 0,0963%. Sehingga, dapat disimpulkan

bahwa hasil klasifikasi yang didapat untuk implementasi KNN pada DBMS MySQL hampir mendekati valid dengan aplikasi *machine learning*, WEKA. Hasil pengujian akurasi nilai k, mendapatkan hasil nilai rata-rata akurasi maksimum pada k=19 sebesar 97,75. Sehingga, nilai k=19 adalah nilai k terbaik untuk KNN pada data akademik tahun 2016/2017 sampai tahun 2018/2019. Berdasarkan hasil pengujian performa keseluruhan menunjukkan bahwa jumlah waktu dengan rata-rata tertinggi yaitu 0,525 detik diperoleh data akademik tahun 2016/2017. Sehingga dapat dikatakan bahwa waktu eksekusi proses KNN dalam DBMS memerlukan waktu yang dapat ditolerir oleh pengguna.

8. DAFTAR PUSTAKA

Agrawal, K. R. (2014). K-Nearest Neighbor for Uncertain Data. *International Journal of Computer Applications (0975-8887). Vol. 105 No. 11,* 13-16.

Gunjai, B. (2003). Database Management: Concepts and Design.

Han, Jiawei, and Micheline Kamber. *Data Mining Concepts & Techniques.* Morgan Kaufmann Publishers, 2000.

Kulkarni, S. R., Lugosi, G., & Venkatesh, S. S. (1998). Learning pattern classification-A survey. *IEEE Transactions on Information Theory*.

Phyu, T. N. (2009). Survey of Classification Techniques in Data. *Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I.*

Yong, Q. (2007). Integrating Frequent Itemsets Mining with Relational Database. *The Eighth International Conference on Electronic Measurement and Instruments ICEMI'2007*.