Analisis Perbandingan CPU dan GPU (CUDA) Pada Klasifikasi Data Mining dengan Menggunakan Metode K-Nearest Neighbor Kernel Algorthm

Faris Muhammad[1], Ibnu Asror, S.T.,MT [2],Indra Lukmana Sardi. S.T.,MT [3],

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]farismuhammad@students.telkomuniversity.ac.id, [2]iasror@telkomuniversity.ac.id,
[3]indraluk@telkomuniversity.ac.id

Abstrak

Data mining merupakan proses semi-otomatis untuk pengekplorasian data yang berjumlah besar gunanya untuk mendapatkan pola yang berguna. Data mining ini merupakan proses gabungan antar bidang-bidang terutama adalah machine learning, analisis statistik dan basis data. Data mining berusaha untuk menemukan kaidah dan pola dari data.Salah satu task yang penting dalam data mining adalah classification (klasifikasi). Klasifikasi ini dapat dideskripsikan sebagai berikut: terdiri dari data *input* yang disebut juga sebagai *training set* terdiri dari sejumlah *examples* (record) yang masing-masing memilki sejumlah atribut atau disebut juga fitur. Adapun tujuan klasifikasi ini adalah untuk menganalisa data *input* dan mengembangkan sebuah model yang akurat untuk setiap kelas berdasarkan beberapa *variabel prediktor*. Untuk menghasilkan informasi saat melakukan proses data *mining* kendala yang dihadapi adalah banyaknya jumlah data sehingga proses yang dilakukan oleh CPU akan berjalan sangat lambat apabila dirasakan. Untuk menanggulangi masalah ini maka proses data mining menggunakan GPU menjadi salah satu solusi dalam menangani *running time* yang lambat dan akurasi yang kurang baik. Melalu tugas akhir ini penulis akan mencoba menganalisis sebuah algoritma KNN Kernel, Metode ini merupakan perkembangan dari metode KNN *Standard.* Dimana pada metode KNN *Standard* proses klasifikasi dilakukan dengan melihat sejumlah k tetangga terdekat, dan akan diklasifikasikan berdasarkan jumlah kelas terbanyak pada sejumlah k tetangga terdekatnya. *Classifier* tersebut diuji menggunakan 3 fungsi Kernel. Hasil yang didapat dari percobaan penulis yaitu pada pembagian *5 fold* total waktu CPU1: 1,68 s,CPU2: 15,63 s,GPU1: 12,29 s,GPU2: 4,61 s. dan pada pembagian *10 fold* total waktu CPU1: 1,53 s,CPU2: 15,27 s,GPU1: 12,05 s,GPU2: 4,55. Akurasi yang didapatkan pada pembagian *5 fold* 63,87% dan pembagian *10 fold* 64,30% pada semua perangkat.

Kata Kunci : data mining,klasifikasi,CPU, GPU,KNN Kernel

Abstract

Data mining is a semi-automatic process for exploring and analyzing large amounts of data to get useful patterns. Data mining is a joint process between fields, especially machine learning, statistical analysis and database. Data mining tries to find the rules and patterns of data. One important task in data mining is classification (classification). This classification can be described as follows: consists of input data which is also called training set consisting of a number of examples (records) which each have a number of attributes or also called features. The purpose of this classification is to analyze input data and develop an accurate model for each class based on several predictor variables. To produce information when doing data mining process, the obstacles faced are the large amount of data so that the process carried out by the CPU will run very slowly when felt. To overcome this problem, the data mining process uses GPU to be one of the solutions in handling slow running time and poor accuracy. Through this final project the author will try to analyze a KNN Kernel algorithm, this method is a development of the KNN Standard method. Where in the KNN Standard method the classification process is carried out by looking at a number of the closest neighbors, and will be classified based on the number of classes in the number of the closest neighbors. The classifier is tested using 3 Kernel functions. The results obtained from the authors' experiments are that the division of 5 fold total CPU time1: 1.68 s, CPU2: 15.63 s, GPU1: 12.29 s, GPU2: 4.61 s. and in dividing the 10 fold total CPU time1: 1.53 s, CPU2: 15.27 s, GPU1: 12.05 s, GPU2: 4.55. Accuracy obtained at 5 fold division is 63.87% and division of 10 fold is 64.30% on all devices.

Keywords: data mining,classication,CPU,GPU,KNN Kernel

1. Pendahuluan

1.1 Latar Belakang

Seiring dengan perkembangan teknologi dalam hal pengumpulan data dan penyimpanan data dapat menyebabkan tumpukan data yang banyak. Dengan adanya kumpulan data yang banyak, maka timbulah suatu kebutuhan untuk bisa memanfaatkan data tersebut. Pemanfaatan data tersebut tentunya bertujuan untuk mendapatkan informasi penting dari pola-pola data yang terbentuk. Proses untuk mendapatkan informasi atau pola-pola berharga dari sekumpulan data tersebut lah dinamakan Data mining[1]. Klasifikasi merupakan suatu metode dari data mining. Ini merupakan metode prediktif yang melakukan pembelajaran terhadap data-data yang sudah ada sehingga menghasilkan suatu model yang digunakan untuk memprediksi data-data baru. Salah satu alogritma klasifikasi yang terkenal adalah K-nearest neighbor (KNN)[3].

K-nerest neighbor (KNN) adalah suatu metode yang menggunakan alogritma supervised dimana hasil query instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada KNN. Tujuan dari algoritma ini adalah mengklasifikasikan obyek baru berdasarkan atribut dan traning sample[3]. Algoritma KNN sangatlah sederhana, berkerja berdasarkan jarak terpendek dari test sample dan training sample. Penulis melanjutkan penelitian sebelumnya yang hanya menggunakan KNN dalam pemrosesan *data mining.* Namun, pendekatan KNN berdasarkan jarak saja mempunyai akurasi yang cukup kecil sehingga diterapkan pembobotan terhadap jarak tersebut menggunakan Kernel[12][13].

Pada proses data mining kendala berupa banyaknya jumlah data sehingga menyebabkan running time untuk melakukan analisis berjalan sangat lambat sering dialami. Belum lagi akurasi yang kurang tinggi menyebabkan proses analisis kurang sempurna dan mendapatkan kredibilitas rendah. Hal ini biasa disebabkan oleh kemampuan CPU dalam melakukan proses data masih sangat terbatas. Untuk menanggulangi hal ini maka dilakukan lah pemprosesan data pada GPU dimana pemprosesan data di GPU akan meningkatkan running time dan akurasi dari sebuah data itu sendiri[11][12][13].Pada karya tulis ini penulis akan menggunakan algoritma K-Nearest neighbor Kernel untuk melakukan perbandingan pemprosesan data pada CPU dan GPU menggunakan Dataset Diabetes Retinopathy.

1.2 Topik dan Batasannya

Perumusan masalah dalam tugas akhir ini dibagi kedalam beberapa poin, yaitu:

1.Bagaimana cara melakukan proses klasifikasi *data mining* pada CPU dan GPU?

2.Bagaimana cara algoritma *K-Nearest Neighbor Kernel* dapat melakukan pemrosesan data melalui GPU ?

3. Bagaimana hasil *running time* dan *accuracy* yang terjadi di CPU dibandingkan dengan di GPU?

Adapun batasan masalah pada tugas akhir ini adalah sebagai berikut:

1. Menggunakan *K-nearest neighbor kernel Algorithym.*
2. Pemprosesan data diharuskan menggunakan aplikasi CPU dan GPU.
3. Pemprosesan data difokuskan pada peningkatan akurasi dan *running time*.
4. Hardware yang digunakan harus menggunakan *NVIDIA Graphic card*.

1.3 Tujuan

Tujuan dari penulisan tugas akhir ini adalah:

1. Melakukan integrasi sistem agar dapat menjalankan aplikasi CUDA yang memiliki fungsi untuk menjalankan pemorsesan data pada GPU.
2. Melakukan pemrosesan data menggunakan *K-nearest neighbor Kernel Algorithm* dengan menggunakan CPU dan GPU.
3. Menganalisis perbandingan pemprosesan data yang dilakukan di CPU dan GPU.

1.4 Organisasi Tulisan

Adapun bagian-bagian selanjutnya pada TA ini adalah :

1. Landasan Teori
   Pada bagian ini menjelaskan apa saja teori yang mendukung dengan topik TA yang dikerjakan seperti pengertian Data Mining, Pengertian KNN-Kernel dan lainnya.
2. Perancangan Sistem
   Pada bagian ini penulis menjelaskan bagaimana alur jalannya program penulis dari awal pemrosesan data hingga data mining tersebut berhasil diproses.
3. Pengujian dan Analsis
   Pada bagian ini penulis menjelaskan bagaimana pengujian dari pemrosesan data ini dan akan menganalisis hasil yang ada dari hasil pengujian yang penulis buat.
4. Kesimpulan dan Saran
   Pada bagian ini penulis memberi kesimpulan mengenai hasil yang penulis teliti dan memberi saran atas hasil yang penulis dapatkan

2. Studi Terkait

2.1 Definisi Data Mining

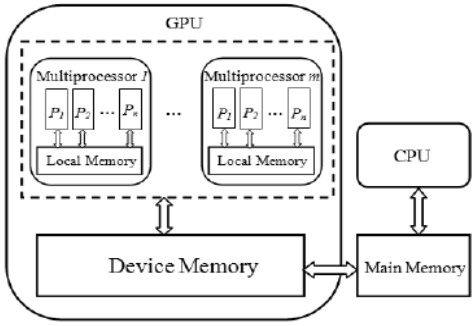
Data mining adalah suatu istilah yang digunakan untuk menemukan pengetahuan yang tersembunyi di dalam *database*. Data mining merupakan proses semi otomatik yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan *machine learning* untuk mengekstrasi dan mengidentifikasi informasi pengetahuan potensial dan juga bermafaat yang tersimpan didalam database besar[1].

Data *mining* adalah kegiatan menemukan pola yang menarik dari data dalam jumlah besar, data dapat disimpan dalam *database*, data warehouse atau penyimpanan informasi lainnya. Data *mining* berkaitan dengan bidang ilmu – ilmu lain seperti, database system, ata warehousing, statistik, *maching learning* dan sebagainya[1].

Data mining didefinisikan sebagai proses menemukan pola-pola di dalam data. Proses ini otomatis atau seringnya semiotomatis. Pola yang ditemukan harus penuh arti dan pola tersebut memberikan keuntungan , biasanya keuntungan secara ekonomi. Data yang dibutuhkanpun dalam jumlah besar[2].

2.2 Parallel Data Mining

Parallel data mining telah banyak dipelajari dalam *distributed system* dimana didistribusikan di cluster computer yang berbeda dan lingkungan grid dengan menggunakan manajemen beban kerja yang dinamik untuk mengatasi masalah dalam memori sehingga mencapai beban kerja yang seimbang dan mengurangi biaya komunikasi[2]. Arsitektur proses *parallel data mining* yang terjadi pada GPU dapat dilihat dari gambar 2.1 dibawah:



Gambar 2.1 Arsitektur Parallel Data Mining

Parallel data mining dapat meningkatkan kemampuan metode *data mining* dalam menghindari *sequential bottleneck*, memiliki kemampuan untuk menangani data dalam jumlah besar dan meningkatkan waktu respon sehingga tercapainya kinerja yang baik[1].

2.3 Klasifikasi

Klasifikasi adalah masalah klasik pada *machine learning* dan data mining untuk meramalkan suatu nilai pada sekumpulan data. Klasifikasi sendiri merupakan suatu proses menemukan kumpulan pola atau fungsi yang mendeskripsikan serta memisahkan kelas data yang satu dengan yang lainnya untuk menyatakan objek tersebut masuk pada kategori tertentu yang sudah ditentukan[4]. Klasifikasi juga dapat diartikan sebagai proses menggeneralisasi struktur data yang sudah dikenal untuk diterapkan kedalam data baru[2,3].

Algoritma yang terdapat dalam klasifikasi berupa *Decision Tree*, *Nearest Neighbor*, *Naïve Bayesian Classification, Neural Network* dan *Support Vector Machines*[3]. Algoritma diatas merupakan beberapa cara untuk melakukan proses klasifikasi dimana memiliki fungsi sebagai pembelajaran yang mengklasifikasi sebuah unsur data ke dalam salah satu dari beberapa kelas yang sudah didefinisikan. Klasifikasi melihat dari pola-polat historis suatu data untuk dipelajari dan bertujuan untuk menempatkan objek-objek baru kedalam kelompok kelasnya tersendiri[3].

2.4 *K-Nearst Neighbor* (KNN)

K-Nearst Neighbor (KNN) adalah salah satu yang paling metode yang umum digunakan untuk pengenalan pola [3], dan telah diterapkan dalam berbagai kasus [5]. Kesederhanaan dan kecepatan konvergensi yang relatif tinggi membuat pilihan yang populer. Namun, dalam beberapa aplikasi, mungkin gagal untuk menghasilkan hasil [5] yang memadai, dan juga membuat Operasi menjadi praktis [5,16,17]. Namun, fakta bahwa itu hanya memiliki satu parameter, jumlah tetangga yang digunakan (K), membuatnya mudah untuk menyempurnakan untuk berbagai situasi. Proses utama terdiri dari langkah-langkah berikut: diberikan satu set N poin (training set), yang label kelas diketahui, mengklasifikasikan satu set n poin (pengujian set) ke set yang sama kelas dengan memeriksa k titik terdekat di sekitar masing-masing titik pengujian ditetapkan dan dengan menerapkan suara mayoritas skema.

Meskipun metode KNN bila digunakan dalam klasifikasi masalah yang cukup cepat, sering terhambat oleh ukuran dataset tertentu, yang mengapa beberapa peneliti memiliki berfokus pada peningkatan kecepatan. SMART-TV misalnya [6,14,15] dirancang untuk menangani dataset tinggi dimensi dengan mengubah mereka menjadi

singledimensional ruang fitur. Pendekatan serupa dibagi dalam [6] untuk data spasial, dimana metode ini bekerja terbaik [6]. Namun, pendekatan ini berkonsentrasi pada kecepatan tinggi terutama dan sering gagal mencapai tingkat akurasi yang sangat baik kecuali mereka diterapkan pada masalah tertentu, seperti dataset spasial [6].

Setelah itu diambil dari k data latih teratas untuk menentukan kelas klasifikasi untuk kelas yang dominan dari k data latih yang diambil. Dekat atau jauhnya tetangga biasanya dihitung dari Euclidean Distance yang direpresentasikan dengan rumus sebagai berikut:

$$D(a,b) = \sqrt{\sum_{k-1}^{d}(a_k - b_k)^2} \quad \text{.....(1)}$$

Untuk D adalah ED, a data training dan b data testing, dan k itu adalah k dari knn.

2.4.1 *K-Nearst Neighbor Kernel*

Inovasi ini didasarkan pada gagasan bahwa sampel di dataset training yang dekat dengan sampel uji harus memiliki berat badan lebih tinggi dalam menentukan jenis sampel uji harus mendapatkan. Namun, hanya k tetangga terdekat memiliki dampak pada titik berlabel. Jadi itu perlu untuk mengubah jarak ke berat badan. Transisi dari jarak ke bobot dihitung dengan fungsi kepadatan kernel K (). fungsi kepadatan kernel yang khas dan umum digunakan adalah diberikan sebagai berikut[7,8]:

Triangular Kernel : $K(u) = (1 - |u|), |u| \leq 1$ .....(2)

Epanechnikov Kernel: $K(u) = \frac{3}{4}(1 - u^2), |u| \leq 1$ .....(3)

Gaussian Kernel : $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}, |u| \leq 1$ .....(4)

Berikut u singkatan jarak standar antara dua poin di ruang vektor, jelas bahwa hanya domain positif K () digunakan. Setiap fungsi kernel memiliki lebar jendela yang dapat mempengaruhi nilai dari kernel yang diberikan fungsi kepadatan. Dalam tulisan ini, kita tidak mempertimbangkan lebar jendela dan hanya peduli tentang pengaruh pengurangan metode kernel klasifikasi KNN. Kemudian yang (k + 1) th tetangga digunakan untuk standarisasi k tetangga terdekat dari rumus berikut[7,8,14,15,16] :

$$u_i = D(x, x_i) = \frac{d(x,x_i)}{d(x,x_{k+1}),}, i = 1, \dots, k \quad \text{.....(5)}$$

Berikut x adalah sampel uji dan x adalah sampel di dataset pelatihan. bahwa interval i u adalah [0,1]. Selanjutnya kita dapat menempatkan nilai yang dihitung i u ke setiap fungsi kernel K () untuk mendapatkan berat badan () i i w K u. Kemudian menghitung berat kategori j C (di sini j C singkatan kategori dalam training set, j 1,, r) menurut k tetangga terdekat dan mengetahui kelas menunjukkan Berat mayoritas bahwa sampel baru harus diklasifikasikan ke dalam dengan fungsi keputusan sebagai berikut[7,8,14,15] :

$$max_c\left(\sum_{i=1}^{k} w_i I(x_i, C_j)\right),\ j = 1, \dots, r \quad \text{.....(6)}$$

$I(x_i, C_j)$ adalah katagori yang bertujuan jika sample x_i merupakan bagian dari kategori C_j kemudian output dari fungsi ini 1 atau 0.

2.5 NVDIA GPUs

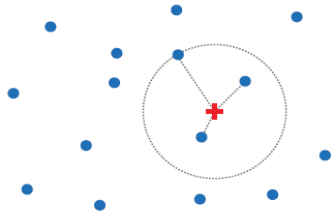
Pada tahun 2006 NVIDIA memperkenalkan Compute Arsitektur Perangkat Terpadu. Hari ini semua GPU NVIDIA adalah CUDA GPU. CUDA bukan arsitektur komputer dalam arti definisi dari set instruksi dan satu set register arsitektur; binari dikompilasi untuk satu CUDA GPU tidak selalu dijalankan pada semua CUDA GPU. Lebih khusus, NVIDIA mendefinisikan kemampuan CUDA komputasi yang berbeda untuk menggambarkan fitur yang didukung oleh CUDA perangkat keras. Pertama CUDA GPU memiliki kemampuan komputasi 1.0. Pada tahun 2011 NVIDIA merilis GPU dengan menghitung kemampuan 2.1, yang dikenal sebagai arsitektur "Fermi". Rincian tentang kemampuan komputasi yang berbeda dijelaskan dalam [9].

2.6 Implementasi CUDA Pada KNN Kernel

Dalam pengimplementasian CUDA menggunakan metode Kernel yang dibagi menjadi dua kernel yaitu[7,8] :

1. kernel pertama dihitung matriks jarak ukuran m × n yang berisi jarak antara titik-titik n permintaan dan titik referensi m. Perhitungan matriks ini sepenuhnya diparalelkan sejak jarak antara pasangan poin independen: setiap thread dihitung jarak antara diberikan titik qi permintaan dan rj titik referensi yang diberikan.
2. kernel kedua diurutkan matriks jarak. N penyortiran proses (satu untuk setiap titik query) yang diparalelkan karena mereka independen: setiap *thread*.

Berikut adalah contoh gambaran dari kernel yang dijelaskan pada Gambar 2.5 dibawah ini :



Gambar 2.5 Kernel GPU

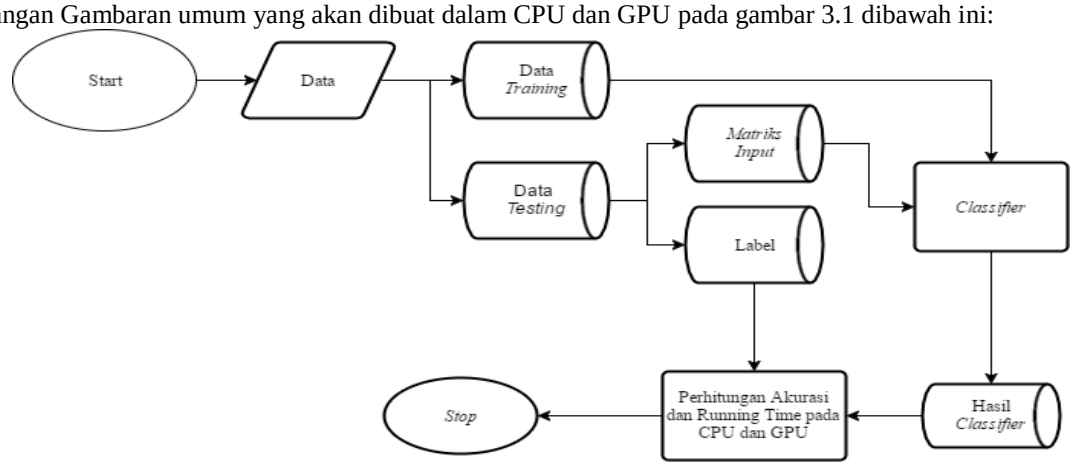
Algoritma sorting digunakan Adalah versi modifikasi Dari penyisipan menyortir. Mari kita berasumsi bahwa elemen k Pertama Dari berbagai D Sudah diurutkan, versi diusulkan memasukkan Unsur (Katakanlah elemen l-th) Ke posisi yang Benar Hanya jika D [l] <D [k]. untuk review Kecil Nilai k, algoritma pengurutan Yang diusulkan tampaknya lebih Cepat Dari algoritma semacamnya pengerjaan efisien[7].

Bekerja dengan m awal × matriks n-index merupakan pemborosan ingatan. Sebuah cara sederhana memungkinkan kita untuk bekerja dengan k × n-index matriks dari awal, sehingga menghindari (m - k) × n memori atas. Proses pemilahan berkaitan dengan setiap kolom array yang monoton dari pertama sampai elemen terakhir. Akibatnya, pada setiap iterasi, indeks titik referensi dianggap diketahui. Sementara menyortir, jika elemen l-th perlu dimasukkan ke dalam matriks jarak, nilai indeks l juga dimasukkan ke dalam k × matriks n-indeks di posisi yang sama persis, iteratif mengisi seluruh matriks. Hasil utama [8] adalah bahwa implementasi CUDA yang diusulkan adalah sampai dengan 300X lebih cepat dari implementasi C sama dan up untuk 150X lebih cepat dari *library* C ++ yang sangat optimal [8,14,15,16,17].

3. Perancangan Sistem

3.1 Gambaran Umum

Dalam proses pembuatan tugas akhir ini dibutuhkan perancangan sistem yang akan dibuat. Berikut adalah Perancangan Gambaran umum yang akan dibuat dalam CPU dan GPU pada gambar 3.1 dibawah ini:



Gambar 3.1 Gambaran Umum Sistem CPU

Berikut adalah gambaran umum aktifitas yang dilakukan adalah sebagai berikut :

Data : Disini penulis menggunakan data dalam bentuk numeric.
Data *Training* : Data latih untuk KNN yang diambil 80% dari jumlah total data.
Data *Testing* : Data yang akan dicari kelasnya yang diambil dari data sisa dari data training.
Matriks Input : *Input* untuk classifier yang diambil dari data testing.
Label : Class data testing.
Classifier : Disini penulis melakukan pengkelompokan data pada setiap keadaan yang dicapai.
Hasil *Classifier* : Hasil prediksi dari *matriks input*.
Perhitungan Akurasi dan Running Time : Di sini akan dilakukan perhitungan akurasi dan *running time* pada CPU dan GPU

3.2 DataSet

Berikut adalah contoh dataset yang akan digunakan, dapat dilihat pada gambar 3.2 dibawah ini :

G	H	I	J	K	L	M	N	O
1.80E+01	1.40E+01	4.99E+01	1.78E+01	5.27E+00	7.72E-01	1.86E-02	6.86E-03	3.92E-03
1.60E+01	1.30E+01	5.77E+01	2.38E+01	3.33E+00	2.34E-01	3.90E-03	3.90E-03	3.90E-03
4.70E+01	3.30E+01	5.58E+01	2.80E+01	1.27E+01	4.85E+00	1.39E+00	3.73E-01	4.18E-02
4.30E+01	3.10E+01	4.05E+01	1.84E+01	9.12E+00	3.08E+00	8.40E-01	2.72E-01	7.65E-03
3.90E+01	2.70E+01	1.80E+01	8.57E+00	4.10E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00
3.70E+01	2.90E+01	2.84E+01	6.94E+00	2.31E+00	3.24E-01	0.00E+00	0.00E+00	0.00E+00
2.50E+01	1.60E+01	1.54E+01	9.11E+00	1.63E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
2.00E+00	1.00E+00	2.07E+01	9.50E+00	1.22E+00	1.50E-01	0.00E+00	0.00E+00	0.00E+00
1.30E+01	1.00E+01	6.67E+01	2.35E+01	6.15E+00	4.96E-01	0.00E+00	0.00E+00	0.00E+00
6.40E+01	4.70E+01	2.21E+01	1.01E+01	8.75E-01	9.98E-02	2.34E-02	0.00E+00	0.00E+00
4.00E+01	3.20E+01	8.44E+01	5.10E+01	1.73E+01	1.97E+00	0.00E+00	0.00E+00	0.00E+00
2.20E+01	1.80E+01	2.25E+01	1.39E+01	4.36E-01	1.16E-01	0.00E+00	0.00E+00	0.00E+00
6.30E+01	5.00E+01	1.06E+01	3.11E+00	6.26E-01	2.88E-01	1.04E-01	4.80E-03	0.00E+00
2.70E+01	1.80E+01	2.30E+01	6.74E+00	2.40E+00	1.89E-01	1.14E-02	0.00E+00	0.00E+00
8.60E+01	7.70E+01	8.61E+00	1.98E+00	4.01E-01	6.61E-02	0.00E+00	0.00E+00	0.00E+00
1.30E+01	9.00E+00	6.51E+01	3.31E+01	8.79E+00	6.74E-01	5.18E-02	2.93E-03	9.78E-04
4.60E+01	3.20E+01	1.23E+02	7.06E+01	3.74E+01	1.99E+01	1.48E+01	6.11E+00	2.35E+00
2.20E+01	1.90E+01	1.70E+01	9.98E+00	1.07E+00	4.85E-01	4.68E-01	3.07E-01	1.89E-01
5.50E+01	4.00E+01	1.97E+01	6.06E+00	9.07E-01	8.01E-02	0.00E+00	0.00E+00	0.00E+00
2.30E+01	1.50E+01	1.16E+02	2.13E+01	9.67E+00	2.28E+00	3.29E-01	1.86E-01	1.18E-01
3.10E+01	2.30E+01	6.14E+01	3.52E+01	8.11E+00	1.20E+00	1.78E-01	1.08E-02	0.00E+00
1.20E+01	7.00E+00	1.80E+02	3.47E+01	1.30E+01	1.05E+00	2.30E-02	5.00E-03	2.00E-03
2.80E+01	2.40E+01	8.82E+00	3.16E+00	1.90E+00	1.52E+00	1.29E+00	1.66E-01	0.00E+00
9.00E+00	6.00E+00	7.29E+01	2.03E+01	9.79E+00	9.16E-01	4.08E-02	0.00E+00	0.00E+00
4.00E+00	3.00E+00	1.33E+02	6.89E+00	2.72E+00	1.70E-01	1.79E-02	1.16E-02	3.17E-03
2.00E+01	1.20E+01	7.31E+01	2.31E+01	1.31E+01	5.44E+00	3.85E+00	2.03E+00	5.19E-01
3.90E+01	2.60E+01	7.13E+01	3.94E+01	2.11E+01	3.96E+00	6.37E-01	3.29E-02	0.00E+00

Gambar 3.2 Dataset

Dataset yang digunakan adalah dataset *Diabetes Retinopathy* yaitu gambar jaringan pada tubuh manusia yang sudah di *preprocessing* menjadi data *numeric* yang didapat dari
http://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set

3.2.1 *Dataset Atribut*

Berikut adalah contoh dataset yang akan digunakan dibawah ini:

@relation dr

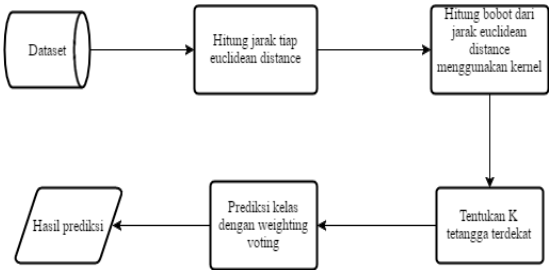
@attribute 0 numeric, @attribut 1 numeric, @attribut 2 numeric, @attribut 3 numeric, @attribute 4 numeric, @attribute 5 numeric, @attribute 6 numeric, @attribute 7 numeric, @attribute 8 numeric, @attribute 9 numeric, @attribute 10 numeric, @attribute 11 numeric, @attribute 12 numeric, @attribute 13 numeric, @attribute 13 numeric, @attribute 14 numeric, @attribute 15 numeric, @attribute 16 numeric, @attribute 17 numeric, @attribute 18 numeric, @attribute class {0,1}

3.2.2 Klasifikasi

Klasifikasi didalam data ini dibagi menjadi dua kelas yaitu kelas yang terjangkit penyakit diabetes dan kelas yang tidak terjangkit diabetes. KNN menggunakan *supervised* dimana didalam data sudah terdapat kelasnya yaitu angka 1 untuk yang terjangkit diabetes dan angka 0 untuk yang tidak terjangkit diabetes.

3.3 Rancangan KNN Kernel

KNN kernel merupakan pengembangan dari KNN berikut adalah alur kerja dari KNN Kernel pada gambae 3.4 dibawah ini :



Gambar 3.4 KNN-Kernel

Alur pengerjaan KNN Kernel dimana pertama kita menghitung jarak tiap euclidean distance kemudian hitung bobot dari jarak yang telah didapatkan menggunakan kernel. Lalu, baru kita tentukan K tetangga terdekat untuk mencari insialnya tersebut. Setelah itu lakukan prediksi kelas dengan cara *weighting voting*. Lalu didapatkan hasil dari prediksi tersebut.

3.4 Data *Training* dan Data *Testing*

Data *Training* merupakan proses untuk melatih suatu dataset dengan mengirimkan data training kepada dataset untuk dipelajari, sehingga algoritma pada data mining dapat merubah parameter dirinya untuk menyesuaikan dengan data yang diberikan saat latihan. Data Testing merupakan proses untuk mengambil parameter untuk membagi

data menjadi Matriks dan juga label. Untuk proses data *training* dan data *testing* yang dilakukan pada aplikasi ini adalah sebagai berikut:

1. Memuat data training ke dalam matrix x dan vector y
2. Deklarasi variable theano berupa
    a. Vector val
    b. Rata-rata = mean(val)
    c. Standar deviasi = std(val)
    d. Total = sum(val)
3. Deklarasi probabilitas prior dan parameter
4. Melakukan perulangan untuk setiap label *class*
    a. Melakukan akumulasi jumlah label class bernilai tertentu pada data training
    b. Melakukan sortir data training berdasarkan nilai label class
    c. Melakukan perhitungan rata-rata dan standar deviasi pada masing-masing atribut berdasarkan nilai label *class*, yang dimasukkan sebagai parameter
5. Menghitung probabilitas prior berdasarkan akumulasi nilai terhadap jumlah dataset
6. Prior dan parameter sudah ditentukan.

3.5 Klasifikasi Data dan *Output* Parameter

Klasifikasi data merupakan suatu proses yang bertujuan untuk melakukan testing setelah data di training sehingga menghasilkan parameter-parameter untuk dijadikan model klasifikasi. Hal yang dilakukan aplikasi ini saat melakukan *testing* data adalah sebagai berikut:

1. Memuat data testing ke dalam matrix x dan vector "truth" sebagai penilaian
2. Melakukan prediksi untuk sejumlah dataset *testing*, dan menghasilkan vector "prediction"
3. Melakukan evaluasi *vector prediction* terhadap *vector truth*

Dalam hal ini vector merupakan sejumlah class yang memiliki urutan tertentu. Vector prediction dan vector truth memiliki panjang yang sama.

3.6 Evaluasi

Evaluasi merupakan proses akhir dari suatu proses data mining yang bertujuan untuk menghasilkan output berupada prediksi, performansi dan *running time*. Hasil dari evaluasi kemudian dapat dilakukan analisis sehingga dapat dihasilkan kesimpulan dari keseluruhan pengujian pada tugas akhir ini. Hasil evaluasi dapat dilakukan dengan cara membandingkan *vector prediction* terhadap *vector truth* yang menghasilkan *accuracy, precision* dan *recall*. Berikut adalah detai proses evaluasi untuk menghasilkan *accuracy, precision* dan *recall* :

- *Accuracy = (true positive + true negative)* / jumlah data
- *Precision = true positive / (true positive + false positive)*
- *Recall = true positive / (true positive + false negative)*

Dengan asumsi bahwa class 1 bernilai positif, ada beberapa istilah sebagai berikut.

1. *True positive* merupakan ekspresi dimana prediksi bernilai positif dan truth positif;
2. *True negative* merupakan ekspresi dimana prediksi bernilai negatif dan truth negatif;
3. *False positive* merupakan ekspresi dimana prediksi positif dan truth negatif; dan
4. *False negative* merupakan ekspresi dimana prediksi negatif dan truth positif.

3.6 Implementasi KNN-Kernel

Berikut adalah implementasi dari hitung manual ***knn kernel***.

1.Berikut adalah gambaran data yang digunakan pada perhitungan manual, dapat dilihat pada gambar 3.5 dibawah ini :

	A	B	C	D	E	F	G	H	I	J	K
1	1	1	22	22	22	19	18	14	49,89576	17,77599	5,27092
2	1	1	24	24	22	18	16	13	57,70994	23,79999	3,325423
3	1	1	62	60	59	54	47	33	55,83144	27,99393	12,68749
4	1	1	55	53	53	50	43	31	40,46723	18,44595	9,118901
5	1	1	44	44	44	41	39	27	18,02625	8,570709	0,410381
6	1	1	44	43	41	41	37	29	28,3564	6,935636	2,305771
7											

L	M	N	O	P	Q	R	S	T	U	V
0,771761	0,018632	0,006864	0,003923	0,003923	0,486903	0,100025	1	0		Data Testing
0,234185	0,003903	0,003903	0,003903	0,003903	0,520908	0,144414	0	0		Data Training
4,852282	1,393889	0,373252	0,041817	0,007744	0,530904	0,128548	0	1		Data Training
3,079428	0,840261	0,272434	0,007653	0,001531	0,483284	0,11479	0	0		Data Training
0	0	0	0	0	0,475935	0,123572	0	1		Data Training
0,323724	0	0	0	0	0,502831	0,126741	0	1		Data Training

Gambar 3.5 Data

2.Berikut adalah gambaran Perhitungan Data yang digunakan, dapat dilihat pada gambar 3.6 dibawah ini :

Class K	Weight		class		k =	3
0	1,222927		0	<=	kelas yang telah di klasifikasi	
1	0					
weight untuk masing masing kelas						

Euclidean Distance		sorted ED		Kernel Weight	
10,74371		10,74371		0,848004	
83,98407		54,28123		0,232058	
70,68397		60,58567		0,142865	
60,58567		70,68397			
54,28123		83,98407			

Gambar 3.6 Perhitungan Data

***Euclidean Distance* :** Jarak antara data testing dengan masing-masig data training.
Sorted Euclidean Distance : Pengurutan *ascending* dari semua jarak yang ada.
Kernel weight : Perhitungan *weight* dari semua jarak yang ada.

3.7 K-Fold Cross Validation

Membagi data menjadi bagian sebanyak K. Misal kalo 5-fold *cross validation*, data dibagi menjadi 5 bagian pada table 3.5 dan 10 bagian pada tabel 3.6 dibawah ini:

***Tabel 3.5* K-Fold Cross Validation 5 - Fold**

Sample Data	Range Data	Data Training	Data Testing
A	1 - 936	BCDE	A
B	937 - 1872	ACDE	B
C	1872 - 2808	ABDE	C
D	2809 - 3744	ABCE	D
E	3745 - 4680	ABCD	E

Tabel 3.*6 K-Fold Cross Validation 10-Fold*

Sample Data	Range Data	Data Training	Data Testing
A	1 - 468	BCDEFGHIJ	A
B	469 - 936	ACDEFGHIJ	B
C	937 - 1404	ABDEFGHIJ	C
D	1405 - 1872	ABCEFGHIJ	D
E	1873 - 2340	ABCDFGHIJ	E
F	2341 - 2808	ABCDEGHIJ	F
G	2809 - 3276	ABCDEFHIJ	G
H	3277 - 3744	ABCDEFGIJ	H
I	3745 - 4212	ABCDEFGHJ	I
J	4213 - 4680	ABCDEFGHI	J

Setelah dibagi, kemuadian setiap bagian akan menjadi data testing dan menjadi data training namun pada tahap yang berbeda. Contohnya jika bagian 1 menjadi data *testing*, maka bagian 2-5 akan menjadi data training, jika bagian 2 menjadi data *testing* maka yang akan menjadi data training yaitu bagian 1, 3,4 dan 5. Begitu seterusnya hingga semua bagian data dicoba menjadi data *testing*. Setelah mendapat 5 hasil akurasi dari 5 percobaan data *testing* yang berbeda, maka dicari rata rata akurasinya.

3.6 Spesifikasi Perangkat Lunak

Beberapa perangkat lunak yang akan digunakan adalah sebagai berikut :
1. Sistem Operasi Windows.
2. Bahasa Pemograman Python.
3. Aplikasi CUDA pada GPU.
4. *Library Theano*.

3.7 Spesifikasi Perangkat Keras

Adapun daftar spesifikasi perangkat keras yang digunakan untuk menjalankan sistem adalah sebagai berikut :
Perangkat Keras 1 pada lampiran A dan B:

A. Processor Intel® Core™ i7-3610 CPU @ 2.30 GHz
B. Memory 8 GB RAM
C. VGA NVIDIA GeForce GT 650M

Perangkat Keras 2 pada lampiran C dan D:

A. Processor Intel® Core™ i5-3440HQ CPU @ 2.60 GHz ~2.66 GHz
B. Memory 8GB RAM
C. VGA NVIDIA GeForce GTX 960

4. Pengujian dan Analisis

4.1 Skenario Pengujian

Skenario pengujian sistem yang dilakukan menggunakan *K-fold cross validation.* Parameter-parameter yang diuji untuk mengukur performansi dari sistem yaitu :
1. CPU dan GPU

Pada sistem ini waktu eksekusi metode terhadap cpu dan gpu akan di kalkulasikan lalu dibandingkan mana yang lebih cepat waktu pengerjaannya antara cpu dan gpu.

2. K pada KNN-Kernel

K yang digunakan pada sistem bermacam-macam, Untuk melihat tingkat akurasi dari sistem.

3. Perangkat keras 1 dan perangkat keras 2

Pada pengujian ini kita melakukan perbandingan antara perangkat keras 1 dan perangkat keras 2, dimana kita membanding waktu pemrosesan oleh metode KNN-Kernel terhadap CPU dan GPU perangkat keras 1 maupun perangkat keras 2.

4. 5-fold cross validation dan 10-fold cross validation

Pada pengujian ini akan dilihat akurasi tertinggi antara 5-fold cross validation dan 10-fold cross validation. Untuk menghasilkan pengujian yang lebih *reliabel* maka dilakukan pengujian dengan menggunakan 2 perangkat keras yang berbeda baik di CPU maupun GPU yang digunakan sehingga memudahkan analisis yang nanti dilakukan,

Semua skenario akan dilakukan berdasarkan pembagian K-*fold* cross validation yang dibuat sebelumnya, sehingga untuk setiap skenario dengan pembagian 5 – *fold* pada tabel 4.1 dan 10-*fold* pada tabel 4.2 adalah sebagai berikut:

Tabel 4.1 Skenario Dengan Pembagian Data 5 – Fold

Data Training	***Data Testing***
BCDE	A
ACDE	B
ABDE	C
ABCE	D
ABCD	E

Tabel 4.2 Skenario Dengan Pembagian Data 10 – Fold

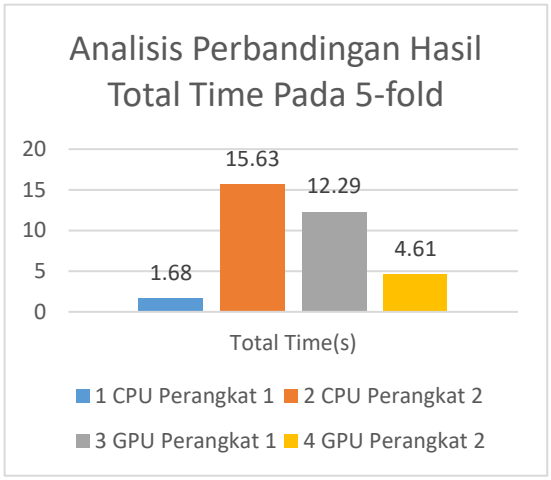
Data Training	***Data Testing***
BCDEFGHIJ	A
ACDEFGHIJ	B
ABDEFGHIJ	C
ABCEFGHIJ	D
ABCDFGHIJ	E
ABCDEGHIJ	F
ABCDEFHIJ	G
ABCDEFGIJ	H
ABCDEFGHJ	I
ABCDEFGHI	J

Kemudian untuk pengukuran performansi disesuaikan dengan perancangan sistem, maka setiap skenario akan dicari poin poin sebagai berikut:

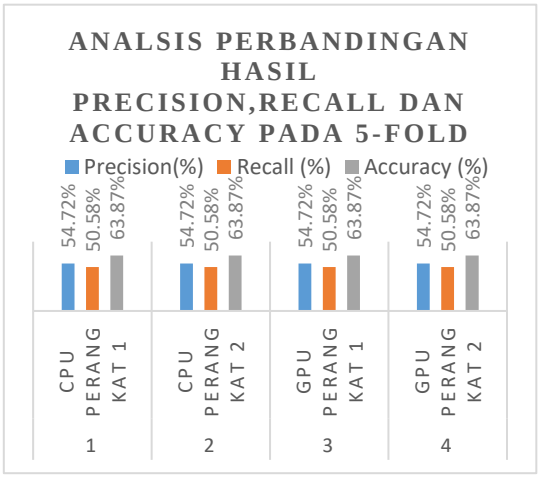
- K : Jumlah tetangga pada K-Nearest Neighbor
- *Total Time :* jumlah waktu yang didapatkan dari penjumlahan *initiation time, training time* dan *testing time.*
- *Precision* : tingkat ketepatan informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem.
- *Recall* : tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi.
- *Accuracy* : tingkat kedekatan antara nilai prediksi dengan nilai aktual.

4.2 Analisis Hasil Pengujian

Berikut adalah hasil rata-rata dari analisis pada lampiran A dan Lampiran C yang bisa dilihat pada gambar 4.3 dan 4.4 dibawah ini :



Gambar 4.3 Grafik Perbandingan Hasil Total Time dengan Pembagian Data 5-Fold

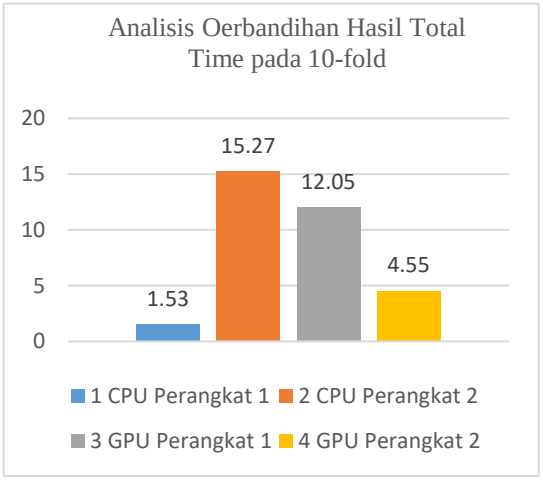


Gambar 4.4 Grafik Perbandingan *Precision,Recall* dan *Accuracy* dengan pembagian data 10-fold

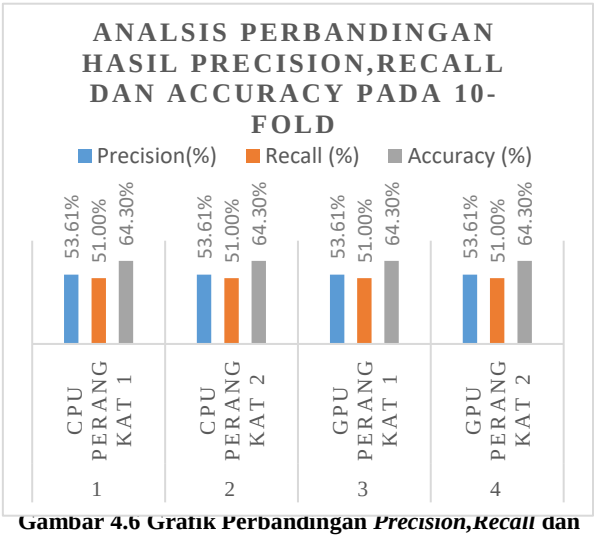
Pada Gambar 4.3 dan 4.4 terdapat perbandingan rata-rata dari tiap skenario pada pembagian data 5-fold *total time*, *precision*, *recall*, dan *accuracy*. Pada *total time* terdapat perbedaan yang signifikan antara perbandingan CPU perangkat 1 dengan GPU perangkat 1 begitu juga CPU perangkat 2 dengan GPU perangkat 2. Pada perangkat 1 bisa dilihat bahwa CPU perangkat 1 jauh lebih cepat dibandingkan dengan GPU perangkat 1. Sebaliknya, Pada perangkat 2 GPU perangkat 2 lah yang Jauh lebih cepat dari pada CPU perangkat 2. Jika diberi peringkat maka dapat dihasilkan bahwa CPU perangkat 1 lah yang lebih cepat lalu disusul dengan GPU perangkat 2 kemudian GPU perangkat 3 dan di tempat terakhir yang paling lama yaitu CPU perangkat 2.

Untuk *percision* selalu menghasilkan angka 54,72. Untuk *recall* pada selalu menghasilkan angka 50,58. Untuk *accuracy* selalu menghasilkan angka 63,87%.Ini membuktikan bahwa proses data mining pada CPU maupun GPU tidak menghasilkan perbedaan pada ketiga poin diatas.Dan untuk *Accuracy* dataset yang menyentuh angka 63,87% membuktikan bahwa dataset yang digunakan cukup baik.

Berikut adalah hasil rata-rata dari analisis data pada lampiran B dan lampiran C yang bisa dilihat pada gambar 4.5 dan 4.6 dibawah ini :



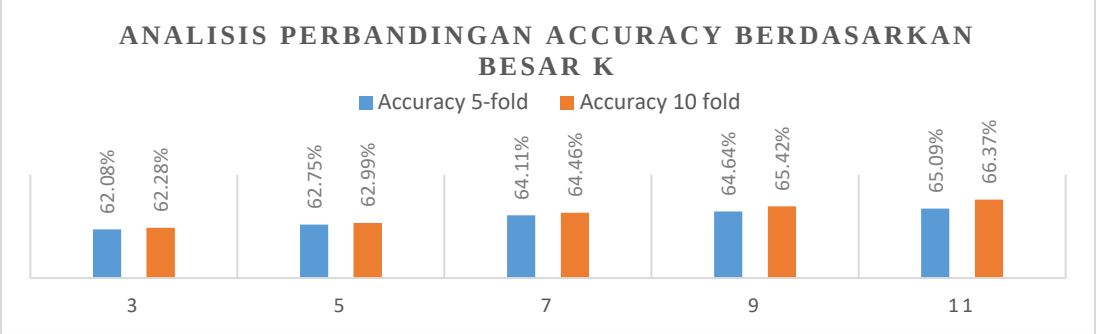
Gambar 4.5 Grafik Perbandingan Hasil Total time dengan pembagian data 10-fold



Gambar 4.6 Grafik Perbandingan *Precision,Recall* dan *Accuracy* dengan pembagian data 10-fold

Pada gambar 4.5 dan 4.6 terdapat perbandingan rata-rata dari tiap skenario pada pembagian data 10-fold *total time*, *precison*, *recall*, dan a*ccuracy*. Pada *total time* terdapat perbedaan yang signifikan antara perbandingan CPU perangkat 1 dengan GPU perangkat 1 begitu juga CPU perangkat 2 dengan GPU perangkat 2. Pada perangkat 1 bisa dilihat bahwa CPU perangkat 1 jauh lebih cepat dibandingkan dengan GPU perangkat 1. Sebaliknya, Pada perangkat 2 GPU perangkat 2 lah yang Jauh lebih cepat dari pada CPU perangkat 2. Jika diberi peringkat maka dapat dihasilkan bahwa CPU perangkat 1 lah yang lebih cepat lalu disusul dengan GPU perangkat 2 kemudian GPU perangkat 3 dan di tempat terakhir yang paling lama yaitu CPU perangkat 2.

Untuk *percision* selalu menghasilkan angka 53,61%. Untuk *recall* selalu menghasilkan 51,00%. Untuk *accuracy* selalu menghasilkan angka 64,30%. Ini membuktikan bahwa proses data mining pada CPU maupun GPU tidak menghasilkan perbedaan untuk ketiga poin diatas, dan untuk *Accuracy* dataset yang menyentuh angka 64,30% membuktikan bahwa dataset yang digunakan cukup baik.



Gambar 4.7 Analisis Perbandingan *Accuracy* Berdasarkan Besar K

Pada gambar 4.7 terdapat perbandingan rata-rata akurasi berdasarkan skenario besar k. Untuk k sama dengan 3 didapat akurasi sebesar 62,08% pada pembagian data 5-fold dan akurasi sebesar 62,28% pada pembagian data 10-fold. Untuk k sama dengan 5 didapat akurasi sebesar 62,75% pada pembagian data 5-fold dan akurasi sebesar 62,99% pada pembagian data 10-fold. Untuk k sama dengan 7 didapat kaurasi sebesar 64,11% pada pembagian data 5-fold dan akurasi sebesar 64,46% pada pembagian data 10-fold. Untuk k sama dengan 9 didapat akurasi sebesar 64,64% pada pembagian data 5-fold dan akurasi sebesar 65,42% pada pembagian data 10-fold. Untuk k sama dengan 11 didapat akurasi sebesar 65,09% pada pembagian data 5-fold dan akurasi sebesar 66,37% pada pembagian data 10-fold.

Jika melihat hasil diatas dapat disimpulkan bahwa semakin besar k yang kita masukan semakin besar pula akurasi yang akan kita dapatkan dan berlaku sebalikanya

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan pengujian serta analisis yang telah dilakukan pada tugas akhir ini, dapat diambil beberapa kesimpulan yaitu:

1. Sistem dapat terintegrasi dengan baik dengan GPU sehingga dapat melakukan pemrosesan *Data Mining* dengan menggunakan GPU.
2. Proses Data Mining pada CPU dan GPU untuk *total time* mendapatkan hasil yang mengejutkan dimana pemrosesan data pada CPU 1 lebih cepat dari pada yang lainnya dengan mendapatkan angka 1,68 detik pada pembagian data 5-fold dan 1,53 detik pada pembagian data 10-fold dibandingkan dengan GPU 1 hanya mendapatkan angka 12,29 detik pada pembagian data 5-fold dan 12,05 detik dengan pembagian data 10-fold. Ini disebabkan CPU 1 mempunyai perangkat yang lebih bagus dibandingkan GPU 1. Sedangkan sebaliknya pada perangkat 2 yang mempunyai *hardware* yang sama bagus, GPU 2 mendapatkan waktu lebih cepat dibandingkan dengan CPU 2 dengan mendapatkan angka 4,61 detik pada pembagian data 5-fold dan 4,55 detik untuk pembagian data 10-fold, ini membuktikan bahwa proses data mining pada GPU lebih cepat dibandingkan dengan CPU.
3. Klasifikasi pada sistem sudah cukup baik antara CPU dan GPU yang selalu memberikan hasil yang sama. Terbukti pada nilai *precision* yang mencapai angka 54,72%, *recall* yang mencapai 50,58% dan *accuracy* yang mencapai 63,81% pada pembagian data 5-fold, sama halnya dengan pembagian data 10-fold dengan *precision* yang mencapai angka 53,61%, *recall* yang mencapai angka 51% dan *accuracy* yang mencapai angka 64,30%. Ini membuktikan bahwa proses data mining pada GPU dapat memberikan proses klasifikasi yang cukup baik sama seperti pada CPU.

4. Dalam perhitungan *accuracy* didapatkan angka untuk k sama dengan 3 mendapatkan angka 62,08% pada pembagian data 5-fold dan 62,28% pada pembagian data 10-fold, untuk k sama dengan 5 mendapatkan angka 62,75% pada pembagian data 5-fold dan 62,99 pada pembagian data 10-fold, untuk k sama dengan 7 mendapatkan angka 64,11% pada pembagian data 5-fold dan 64,46 pada pembagian data 10-fold, untuk k sama dengan 9 mendapatkan angka 64,64% pada pembagian data 5-fold dan 65,42% pada pembagian data 10-fold, untuk k sama dengan 11 mendapatkan angka 65,09% pada pembagian data 5-fold dan 66,37 pada pembagian data 10-fold. Ini membuktikan bahwa semakin besar k yang digunakan makan akan semakin besar akurasi yang akan kita dapatkan.

5.2 Saran

Berikut merupakan saran dari penulis untuk penelitian kedepannya :

1. Menggunakan CPU dan GPU yang berbeda dari yang digunakan oleh penulis karena perbedaan hardware tersebut dapat membuat eksekusi program menjadi berbeda.
2. Memperlebar parameter K yang akan diuji karena parameter K yang diuji bisa saja lingkupnya masih kecil.
3. Menggunakan metode lain untuk pengklasifikasian menggunakan GPU dan CPU.
4. Menggunakan data yang lebih besar agar mendapatkan hasil pemrosesan yang lebih akurat.

Daftar Pustaka

[1] Gary M M Weis dan Brian D Davison, "Data Mining", 2010.
[2] Zaki, J. Mohammed, "Parallel and Distributed Data Mining An Introduction", 2012.
[3] Hemlata Sahu, Shalini Shrma dan Seema Gondhalakar, "A Brief Overview on Data Mining Survey", 2012.
[4] Neethu Baby dan Priyanka LT, "Customer Classification And Prediction Based On Data Mining Technique", 2012.
[5] Yun-lei Cai, Duo Ji dan Dong-feng Cai, "A KNN Research Paper Classification Method Based on Shared Nearest Neighbor", 2010.
[6] R.S. Ramadhan, Z. Junta dan Ardytha Luthfarta, "Penerapan Algoritma K-Nearest Neighbor pada Information Retrieval dalam Penentuan Topik Referensi Tugas Akhir" Journal of Applied Intelligent System, Vol. 1, No. 2, 2016.
[7] Hechenbichler, Schliep, "Weighted k-Nearest-Neighbor Techniques and Ordinal Classificatio" Sonderforschungsbereich 386, Paper 399, 2004.
[8] Schimidt-Thieme, Lars, "Nearest Neighbor and Kernel Methods" Information Systems and Machine Learning Lab (ISMLL), Institute BW/WI & Institute for Computer Science, University of Hildesheim Course on Machine Learning, 2007
[9] Patrick Harvey, Jesse, "GPU acceleration of object classification algorithms using NVIDIA CUDA",2009.
[10] NVIDIA Corporation. NVIDIA CUDA – NVIDIA CUDA C Programming Guide, Version 4.0, 2011.
[11] Aydin, Berkay, "Parallel Algorithms on Nearest Neighbor Search".
[12] S.G. Lade and Nikhil Vyawahare, "Document Classification Using KNN on GPU", 2014.
[13] Kankatala, Sriram, "Performance Analysis of kNN on large datasets using CUDA & Pthreads comparing between CPU & GPU", 2015.
[14] Xiaogang Han, Junfa Liu, Zhiqi Shen, dan Chunyan Miao, "An Optimized K-Nearest Neighbor Algorithm for Large Scale Hierarchical Text Classification", 2010.
[15] V.B.Nikam dan B.B.Meshram, "PARALLEL kNN ON GPU ARCHITECTURE USING OpenCL", 2011.
[16] L.Shengren dan N. Amenta, "Brute-Force k-Nearest Neighbors Search on the GPU"
[17] A.Tahta, S.Budi dan R.D Ali, "Analisa Perbandingan Metode Hierarchical Clustering, K-means dan Gabungan Keduanya dalam Cluster Data (Studi kasus : Problem Kerja Praktek Jurusan Teknik Industri ITS)", 2012.