BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkembangan penggunaan internet dan teknologi informasi mengakibatkan pertumbuhan data yang sangat besar dan terjadi secara terus-menerus sehingga data sulit untuk dikelola, diproses, maupun dianalisis menggunakan teknologi pengolahan data biasa. Data yang terus bertumbuh menyebabkan basis data konvensional menjadi kurang efektif untuk mengolah data. Teknologi saat ini telah menemukan sebuah cara untuk mengurangi biaya penyimpanan dan komputasi data, sehingga kapasitas data dapat ditingkatkan dan data menjadi lebih mudah diolah.

Big data adalah data dalam jumlah sangat besar dikumpulkan, disimpan, diolah, dan dianalisis agar menghasilkan informasi yang bermanfaat sebagai dasar pengambilan keputusan atau kebijakan. *Data mining* adalah teknik ekstraksi informasi terhadap sekumpulan data dalam jumlah besar. *Data mining* efektif menggantikan pemrosesan kueri pada basis data dengan data berukuran besar. Masalah yang umum terjadi adalah data yang tersimpan banyak mengandung data yang bersifat privasi sehingga perlu adanya perlindungan privasi pada data yang akan diolah.

Perlindungan privasi dicapai dengan metode enkripsi dan anonimisasi. Enkripsi adalah metode yang memanfaatkan pola atau kunci tertentu. Anonimisasi adalah metode yang menyamarkan satu atau lebih nilai atribut data. Pada kasus tertentu, keamanan enkripsi dapat ditembus melalui penalaran nilai atribut. Penalaran ini sangat berbahaya karena menghubungkan nilai atribut data secara tidak langsung, dapat mengungkapkan entitas pemilik data. Dengan menerapkan konsep anonimisasi diharapkan nilai keterhubungan antar atribut data dapat diperkecil.

Dengan melakukan anonimisasi pada sebagian nilai atribut, bobot informasi yang diperoleh akan semakin kecil. Permasalahan *K-anonymity* adalah pencarian solusi untuk menyeimbangkan nilai informasi yang diperoleh dengan nilai informasi yang disamarkan. Permasalahan *K-anonymity* diuji dengan pendekatan generalisasi dan supresi. Hasilnya dinilai kurang efektif karena tingginya jumlah informasi yang hilang. Berdasarkan penelitian, permasalahan *K-anonymity* tercapai melalui penerapan *K-member clustering*. Penerapan *K-member clustering* pada algoritma *Greedy K-member clustering* dinilai baik karena dapat meminimalkan jumlah informasi yang hilang.

Spark adalah *framework* yang tepat untuk memproses data dengan ukuran yang relatif besar seperti *big data*, dengan membagi data tersebut ke sistem terdistribusi. Penggunaan Spark menggeser penggunaan Map Reduce pada Hadoop yang dinilai cukup lambat. Kelebihannya adalah Spark memiliki proses komputasi yang lebih cepat karena sebagian besar pemrosesan Spark berada pada RAM. Selain itu, Spark mampu melakukan pemrosesan *data mining* menggunakan *library* tambahan Spark MLlib. Kekurangannya adalah Spark masih tetap bergantung pada mekanisme penyimpanan Hadoop, agar hasil pemrosesan data dapat tersimpan di dalam *hardisk* komputer.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat memproses data semi terstruktur menjadi data anonimisasi menggunakan konsep *K-anonimity*. Perangkat lunak ini berjalan di atas Spark untuk memudahkan proses anonimisasi pada lingkungan *big data*. Algoritma *Greedy K-member clustering* dinilai tepat untuk melakukan anonimisasi data karena meminimalkan jumlah informasi yang hilang saat proses *data mining* di penelitian sebelumnya. Penelitian ini bertujuan membandingkan hasil *data mining* sebelum dan setelah dilakukan anonimisasi.

1.2 Rumusan Masalah

Berdasarkan deskripsi diatas, rumusan masalah pada skripsi ini adalah sebagai berikut:

1. Bagaimana cara kerja algoritma *Greedy K-member clustering* ?
2. Bagaimana implementasi algoritma *Greedy K-member clustering* pada Spark?
3. Bagaimana hasil *data mining* sebelum dan setelah dilakukan anonimisasi?

1.3 Tujuan

Berdasarkan rumusan masalah di atas, tujuan dari skripsi ini adalah sebagai berikut:

1. Mempelajari cara kerja algoritma *Greedy K-member clustering.*
2. Mengimplementasikan algoritma *Greedy K-member clustering* pada Spark.
3. Menganalisis hasil *data mining* sebelum dan setelah dilakukan anonimisasi.

1.4 Batasan Masalah

Batasan masalah pada pengerjaan skripsi ini adalah sebagai berikut:

1. Perangkat lunak dapat berjalan diatas Spark.
2. Perangkat lunak dapat menerapkan algoritma *Greedy K-member clustering.*
3. Perangkat lunak dapat diimplementasikan menggunakan *library* Scala-swing.
4. Perangkat lunak hanya menerima input data semi terstruktur CSV dan XML.
5. Menggunakan teknik *data mining* yang tersedia pada *library* Spark MLlib.
6. Membandingkan hasil *data mining* sebelum dan setelah dilakukan anonimisasi.

1.5 Metodologi

Bagian-bagian pengerjaan skripsi ini adalah sebagai berikut:

1. Mempelajari dasar-dasar privasi data.
2. Mempelajari konsep *K-anonimity* pada algoritma *Greedy K-member clustering.*
3. Mempelajari teknik-teknik dasar *data mining.*
4. Mempelajari konsep Hadoop, Spark, dan Spark MLlib.
5. Mempelajari bahasa pemrograman Scala pada Spark.
6. Melakukan analisis masalah dan mengumpulkan data studi kasus.
7. Mengimplementasikan algoritma *Greedy K-member clustering* pada Spark.
8. Mengimplementasikan tampilan perangkat lunak menggunakan *library* Scala-swing.
9. Mengimplementasikan teknik *data mining* menggunakan *library* Spark MLlib.
10. Melakukan pengujian fungsional dan experimental.
11. Melakukan analisis hasil *data mining* sebelum dan setelah dilakukan anonimisasi.
12. Menarik kesimpulan berdasarkan hasil eksperimen yang telah dilakukan.

1.6 Sistematika Pembahasan

Pengerjaan skripsi ini tersusun atas enam bab sebagai berikut:

- Bab 1 Pendahuluan
  Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
- Bab 2 Dasar Teori
  Berisi dasar teori tentang dasar-dasar dari privasi, *K-anonimity* berbasis *clustering*, dan metode pada teknik *data mining*.
- Bab 3 Analisis Masalah
  Berisi analisis masalah, studi kasus, diagram alian proses.
- Bab 4 Perancangan
  Berisi perancangan antarmuka dan diagram kelas.
- Bab 5 Implementasi dan Pengujian
  Berisi implementasi perangkat lunak, pengujian fungsional, pengujian eksperimental, dan melakukan analisis terhadap hasil pengujian.
- Bab 6 Kesimpulan dan Saran
  Berisi kesimpulan penelitian dan saran untuk penelitian selanjutnya.

1.7 Privasi

Privasi adalah adalah kemampuan satu atau sekelompok individu untuk menutupi atau melindungi kehidupan dan urusan personalnya dari publik dengan mengontrol sumber-sumber informasi mengenai diri mereka. Sebelum melindungi privasi sebuah data, diperlukan pemahaman untuk membedakan jenis atribut yang bersifat sensitif dan non-sensitif.

Personally Identifiable Information (PII) adalah standar yang digunakan untuk menentukan apakah informasi yang ada dapat melakukan identifikasi entitas individu secara lansung atau tidak langsung. PII menjelaskan bahwa identifikasi entitas secara langsung dapat dilakukan menggunakan atribut sensitif. Sedangkan identifikasi entitas secara tidak langsung dapat dilakukan menggunakan penggabungan beberapa atribut non-sensitif.

Berikut adalah contoh informasi yang bersifat sensitif menurut standar PII:

- Identitas diri seperti nama lengkap, tempat tanggal lahir, alamat rumah, alamat email.
- Nomor identitas diri seperti NIK, nomor passport, nomor SIM, nomor wajib pajak, nomor rekening, nomor telepon, dan nomor kartu kredit.
- Karakteristik pribadi seperti foto diri, sidik jari, dan tulisan tangan.
- Data biometrik seperti pemindaian retina, jenis suara, dan geometri wajah.
- Aset informasi lainnya seperti IP Address dan Media Access Control (MAC).

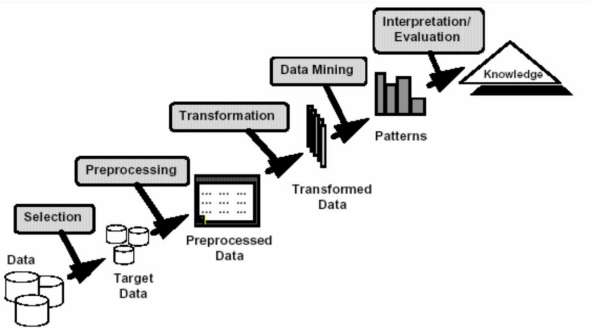
Berikut adalah contoh informasi yang bersifat non-sensitif menurut standar PII:

- Rekaman medis
- Riwayat pendidikan
- Riwayat pekerjaan
- Informasi finasial
- Letak geografis

Untuk melakukan publikasi data dari satu perusahaan ke perusahaan lain digunakan teknik anonimisasi untuk melindungi dan menyamarkan atribut sensitif pada masing-masing data, sehingga data yang diterima oleh perusahaan lain tidak dapat dilakukan identifikasi. Sebagai contoh, perusahaan asuransi ingin membagikan informasi dengan perusahaan pemasaran. Maka perusahaan ansuransi akan menyamarkan data yang bersifat sensitif dan menyediakan data terkait dengan kebutuhan perusahaan pemasaran.

1.8 Data Mining

Data yang dikumpulkan oleh sistem informasi semakin bertambah banyak. Oleh karena itu, perlu adanya cara untuk melakukan proses ekstraksi informasi pada sekumpulan data yang sangat banyak. Menurut Gartner, data mining adalah proses menemukan korelasi, pola, dan tren baru yang bermakna dengan menyaring sejumlah besar data yang disimpan menggunakan teknologi pengenalan pola serta teknik statistik dan matematika. Data mining merupakan bagian dari Knowledge Discovery in Databases (KDD). KDD adalah proses transformasi sekumpulan data yang disimpan pada basis data menjadi informasi yang berguna.



Gambar 1.1: Tahapan pada KDD

Berikut ini adalah penjelasan tahapan pada KDD pada Gambar 1.1 sebagai berikut:

1. Selection adalah proses mengambil data yang relevan terhadap analisis.
2. Preprocessing adalah proses pembersihan data dengan menghilangkan data yang tidak konsisten dan integrasi data dengan menggabungkan beberapa data dari sumber yang berbeda.
3. Transformation adalah proses manipulasi data menggunakan konsep agregasi, generalisasi, normalisasi, dan reduksi agar data tersebut dapat dipakai sesuai kebutuhan analisis.
4. Data Mining adalah proses ekstraksi informasi dari sekumpulan data yang telah diolah pada tahap sebelumnya, menggunakan metode yang umum digunakan untuk mengenali pola seperti klasifikasi, pengelompokan/clustering.
5. Interpretation/Evaluation adalah proses interpretasi hasil pengolahan data menjadi sebuah grafik yang dapat dimengerti, sehingga dapat disimpulkan menjadi sebuah pengetahuan.

Tantangan yang sering muncul pada penggunaan teknik data mining adalah sebagai berikut:

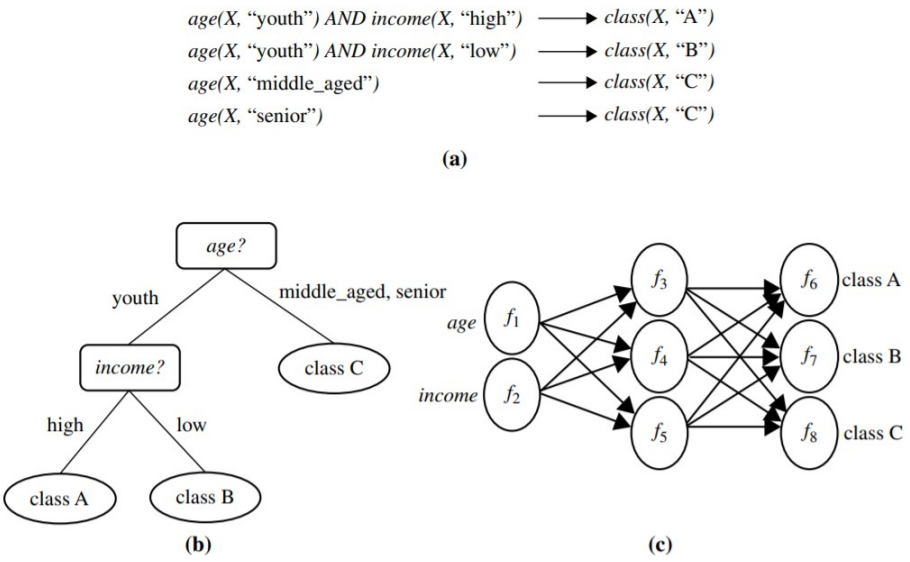
- Skalabilitas, artinya data yang ingin diolah berukuran sangat besar.
- Dimensionalitas yang tinggi, artinya data memiliki jumlah jenis atribut yang sangat banyak.
- Data yang heterogen, artinya munculnya jenis data terstruktur, semi-terstruktur, dan tidak struktur.

Tujuan dari penggunaan teknik data mining adalah sebagai berikut:

- Prediksi adalah menggunakan nilai dari beberapa atribut yang sudah ada untuk memprediksi nilai atribut di masa yang akan datang, contohnya adalah klasifikasi.
- Deskripsi adalah menemukan pola-pola yang dapat merepresentasikan kelompok dari sebuah data, contohnya adalah pengelompokan/clustering.

1.9 Klasifikasi

Klasifikasi adalah proses menemukan model (atau fungsi) yang cocok untuk mendeskripsikan dan membedakan sebuah kelas data dengan kelas data lain. Dalam pembelajaran mesin, klasifikasi sering dianggap sebagai contoh dari metode pembelajaran yang diawasi, yaitu menyimpulkan fungsi dari data pelatihan berlabel. Tugas klasifikasi adalah menganalisis data pelatihan dan menghasilkan fungsi yang tepat untuk digunakan dalam menentukan kelas dari data baru dengan menetapkan label kelas yang benar untuk masing-masing data. Metode klasifikasi memerlukan analisis relevansi antar atribut untuk mengidentifikasi atribut yang akan dipakai pada proses klasifikasi. Atribut tersebut dipilih sebelum proses klasifikasi dimulai. Atribut lain yang tidak relevan terhadap proses klasifikasi dapat diabaikan.



Gambar 1.2: Jenis Pemodelan Klasifikasi

Pada Gambar 1.2, klasifikasi dapat dimodelkan dengan 3 cara yaitu (a) IF-THEN, (b) Decision Tree, dan (c) Neural Network. Setiap model memiliki caranya masing-masing untuk melakukan proses klasifikasi pada data pelatihan. Algoritma klasifikasi biasanya menggunakan salah satu model dari ketiga jenis pemodelan tersebut. Berikut adalah penjelasan singkat dari ketiga jenis pemodelan klasifikasi pada Gambar 1.2 yang umum digunakan:

- IF-THEN adalah pemodelan klasifikasi data dengan diberikan kondisi-kondisi tertentu yang bersifat unik untuk merepresentasikan hasil dari klasifikasi kelas.
- Decision Tree adalah pemodelan klasifikasi data dengan menggunakan analogi struktur pohon, di mana setiap cabang mewakili jenis nilai atribut pada sebuah node dan setiap daun mewakili hasil dari klasifikasi kelas.
- Neural Network adalah pemodelan klasifikasi data menggunakan analogi neuron pada jaringan syaraf, di mana neuron adalah node yang saling terhubung dan memiliki bobot antar node untuk klasifikasi kelas.

1.10 Naive bayes

Naive bayes menerapkan metode klasifikasi probabilistik. Jenis klasifikasi ini menghasilkan probabilitas yang sesuai untuk masing-masing kelas yang mungkin. Kelas dengan probabilitas tertinggi akan dipilih sebagai kelas terbaik. Naive bayes menggunakan teorema bayes. Naive bayes mengasumsikan bahwa fitur tertentu tidak selalu berhubungan dengan fitur lainnya. Naive bayes mudah untuk dibangun dan memiliki komputasi yang lebih cepat daripada model klasifikasi lainnya untuk dataset berukuran sangat besar.

Kelebihan dari Naive bayes adalah sebagai berikut:

- Memiliki kinerja yang cepat untuk melakukan klasifikasi data.
- Membutuhkan data pelatihan yang lebih sedikit dibandingkan model lain.
- Implementasi lebih mudah dibandingkan model lain.

Kekurangan dari Naive bayes adalah sebagai berikut:

- Tidak dapat membuat prediksi untuk probabilitas yang bernilai nol.
- Hanya bergantung pada nilai probabilitas saja sehingga hasil prediksi menjadi kurang baik.

Teorema Bayes dapat dinyatakan dalam rumus sebagai berikut:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)} \tag{1.1}$$

Probabilitas hipotesis tertinggi dapat dinyatakan dalam rumus sebagai berikut:

$$MAP(H) = max(P(H|D)) \tag{1.2}$$

Keterangan:

- P(H|D) adalah probabilitas posterior apabila diberika hipotesis H dan diketahui data D.
- P(D|H) adalah probabilitas posterior data d jika hipotesis h adalah benar.
- P(H) adalah probabilitas hipotesis h adalah benar
- P(D) adalah probabilitas data.

Secara singkat, langkah kerja algoritma Naive Bayes dapat dijelaskan sebagai berikut:

1. Menghitung probabilitas masing-masing kelas.
   Probabilitas kelas dapat dihitung berdasarkan f data dari masing-masing kelas dibagi dengan jumlah f data dari seluruh kelas. Dalam kasus paling sederhana, setiap kelas akan memiliki probabilitas sebesar 0,5. Sebagai contoh dalam klasifikasi biner, probabilitas untuk atribut kelas “class” dengan nilai 1 dapat dihitung sebagai berikut:

$$P(class = 1) = \frac{f(class = 1)}{f(class = 0) + f(class = 1)}$$

2. Menghitung probabilitas bersyarat berdasarkan nilai atribut data yang diketahui.
   Probabilitas bersyarat dapat dihitung berdasarkan f data dari setiap nilai atribut untuk nilai kelas tertentu dibagi dengan f nilai kelas tertentu. Misalnya untuk atribut data yang diketahui

"weather" dengan nilai "sunny" dan "rainy" dan atribut kelas dengan nilai "goingout " dan "stayhome", maka probabilitas bersyarat dari seluruh nilai atribut "weather" jika diketahui seluruh nilai atribut "class" dapat dihitung sebagai berikut:

$$\begin{aligned}
P(weather = sunny|class = goingout) &= \frac{f(weather = sunny\ dan\ class = goingout)}{f(class = goingout)} \\
P(weather = sunny|class = stayhome) &= \frac{f(weather = sunny\ dan\ class = stayhome)}{f(class = stayhome)} \\
P(weather = rainy|class = goingout) &= \frac{f(weather = rainy\ dan\ class = goingout)}{f(class = goingout)} \\
P(weather = rainy|class = stayhome) &= \frac{f(weather = rainy\ dan\ class = stayhome)}{f(class = stayhome)}
\end{aligned}$$

3. Membuat prediksi berdasarkan probabilitas.
   Model Naive bayes dapat membuat prediksi untuk data baru menggunakan teorema bayes berdasarkan rumus berikut ini:

$$MAP(H) = max(P(D|H) * P(H)) \tag{1.3}$$

Apabila ingin menghitung probabilitas bersyarat untuk setiap nilai atribut kelas, dengan diketahui nilai atribut “weather = sunny “, maka dapat dihitung sebagai berikut:

$$\begin{aligned}
goingout &= P(weather = sunnyclass = goingout) * P(class = goingout) \\
stayhome &= P(weather = sunny|class = stayhome) * P(class = stayhome)
\end{aligned}$$

Hasil dari probabilitas bersyarat akan dinormalisasi sebagai berikut:

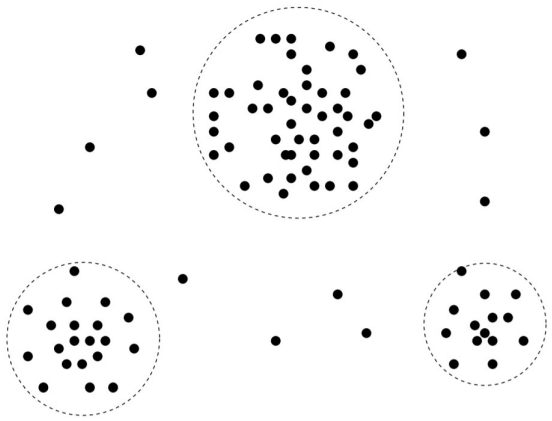
$$\begin{aligned}
P(goingout|weather = sunny) &= \frac{goingout}{(goingout + stayhome)} \\
P(stayhome|weather = sunny) &= \frac{stayhome}{(goingout + stayhome)}
\end{aligned}$$

Apabila ditambahkan atribut baru “car” dengan nilai “working” dan “broken”, maka probabilitas bersyarat setelah atribut baru ditambahkan adalah sebagai berikut:

$$\begin{aligned}
goingout = \quad & P(weather = sunny\ dan\ class = goingout) * \\
& P(car = working|class = goingout) * P(class = goingout)
\end{aligned}$$

1.11 Pengelompokan/Clustering

Clustering adalah proses menganalisis set data tanpa perlu mengetahui label kelas. Pada beberapa kasus, label kelas tidak selalu tersedia pada set pelatihan data. Clustering dapat digunakan untuk menghasilkan label kelas pada sekelompok tuple. Tuple baru dikelompokan berdasarkan prinsip memaksimalkan kesamaan satu kelas dan meminimalkan kesamaan antar kelas. Sebuah kluster terbentuk karena memiliki banyak kesamaan sifat antar sesama tuple di kluster tersebut, dan memiliki perbedaan sifat terhadap tuple di kluster lain. Setiap kluster yang terbentuk dapat dilihat sebagai kelas.



Gambar 1.3: Hasil dari clustering

1.12 K-Means Clustering

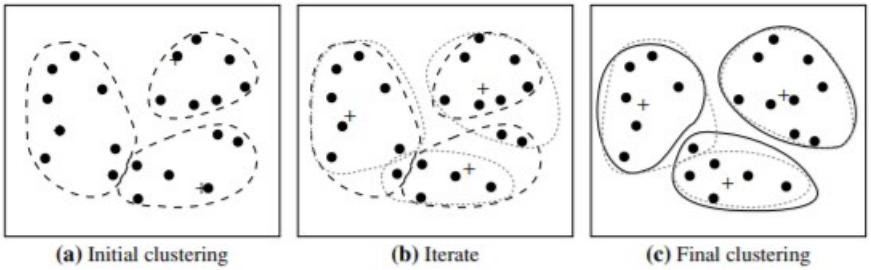
K-Means Clustering akan mempartisi data pada dataset D ke dalam k buah cluster $(C_1, \ldots, C_k)$, dengan syarat $C_i \subset D$ dan $C_i \cap C_j = \emptyset$ untuk $(1 \leq i, j \leq k)$. K-Means Clustering memiliki fungsi obyektif untuk menilai kualitas partisi sehingga tuple-tuple di dalam sebuah cluster akan mirip satu sama lain dan berbeda dengan tuple-tuple di kluster lain. K-Means Clustering bertujuan untuk mencari tingkat kesamaan dalam satu cluster yang tinggi, dengan tingkat kesamaan dengan kluster lain yang rendah.

K-Means Clustering menggunakan centroid dari cluster (C_i) untuk mewakili kluster tersebut. Centroid adalah pusat massa sebuah cluster sebagai titik pusatnya. Centroid dapat didefinisikan dengan berbagai cara seperti dengan rata-rata nilai atribut pada sebuah cluster. Perbedaan antar objek diukur berdasarkan jarak antar objek p dengan centroidnya, diwakili dengan notasi Dist(p, C_i). Notasi Dist(p, C_i) adalah Euclidean distance antara dua titik p dan C_i. Untuk setiap tuple di setiap cluster, jarak dari tuple ke pusat clusternya akan dikuadratkan dan dijumlah. Kualitas sebuah cluster dapat diukur berdasarkan jumlah variasi di dalam kluster itu sendiri. Semakin banyak variasi dalam sebuah kluster, maka akan semakin buruk kualitas kluster tersebut.

Mengoptimalkan jumlah variasi pada kluster merupakan tantangan komputasi. Dalam kasus terburuk, perlu mencoba sejumlah kemungkinan partisi (p) yang eksponensial terhadap jumlah cluster (C) dengan kompleksitas (p^C) dan memeriksa nilai variasi di dalam sebuah kelompok data. Telah terbukti bahwa K-Means Clustering termasuk masalah eksponensial walaupun memiliki jumlah kluster yang sedikit. Jika jumlah kluster dan dimensi ruang pada kluster diperbaiki, masalah dapat diselesaikan dalam waktu O (n dk + 1 log n), di mana n adalah jumlah tuple,dk adalah jumlah dimensi ruang pada kluster. Untuk mengatasi biaya komputasi yang mahal, solusi yang tepat adalah menggunakan pendekatan algoritma greedy. Salah satu contoh penerapan algoritma greedy adalah, Greedy K-Member Clustering.

Secara singkat, langkah kerja algoritma K-Means Clustering dapat dijelaskan sebagai berikut:

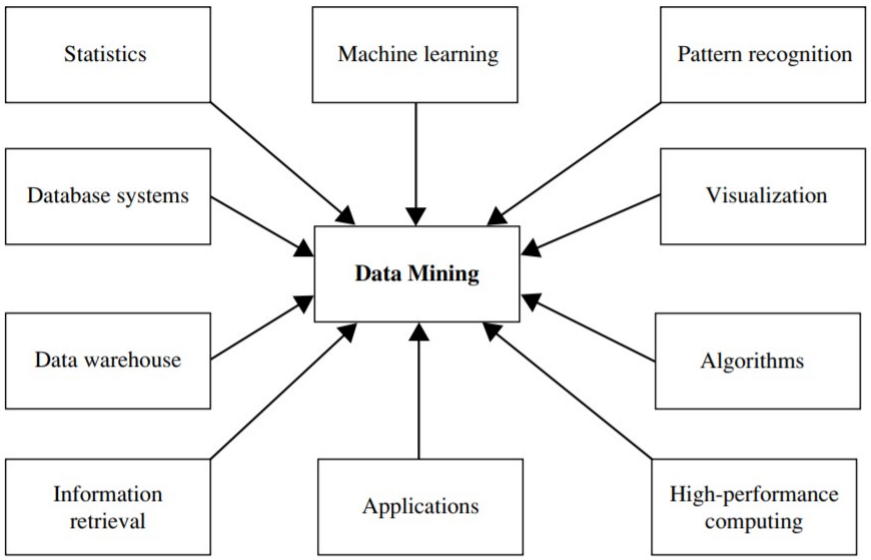
1. Memilih jumlah cluster (K).
2. Inisialisasi K-buah nilai centroid secara random.
3. Menghitung kemiripan antar objek dengan menghitung jarak terdekat antar suatu objek dengan centroid. Jika suatu objek memiliki jarak terpendek dengan nilai centroid tertentu maka objek tersebut akan dikelompokkan pada kelompok centroid tersebut.
4. Menentukan nilai centroid baru dengan cara menghitung nilai rata-rata dari semua objek pada kelompok tertentu.
5. Kumpulan objek pada setiap cluster dikelompokkan kembali berdasarkan nilai centroid baru.
6. Ulangi langkah 3 dan 4 sampai nilai centroid tidak mengalami perubahan atau Thresshold.



Gambar 1.4: K-Means Clustering

1.13 Bidang Terkait Data Mining

Data mining terhubung dengan beberapa bidang lain seperti statistika, machine learning, pengenalan pola, sistem basisdata, sistem data warehouse, information retrieval, visualisasi data, dan bidang lainnya. Jenis bidang ini memberikan berkontribusi yang signifikan terhadap keberhasilan dalam pengolahan data menggunakan teknik data mining.



Gambar 1.5: Jenis bidang terkait data mining

1.13.1 Visualisasi Data

Visualisasi adalah penggunaan representasi data pada komputer. Visualisasi data melibatkan penyajian data dalam bentuk grafik atau gambar agar membuat informasi mudah dimengerti, membantu menjelaskan fakta, dan menentukan arah tindakan. Sulit untuk membuat visualisasi pada ukuran data yang besar. Kemampuan untuk memvisualisasikan data sangat penting untuk melakukan analisis data. Saat ini, komputer dapat memproses data dalam jumlah besar. Visualisasi data berkaitan dengan tata cara untuk menyatakan representasi data secara informatif agar mudah untuk dipahami. Dengan adanya representasi data dalam bentuk visual, seseorang dapat memahami sekumpulan data dengan mudah. Hal ini membantu individu menemukan pola, memahami informasi, dan membentuk opini berdasarkan hasil analisis. Visualisasi menguntungkan bidang studi yang membutuhkan cara-cara inovatif untuk menyajikan informasi yang besar dan kompleks. Dalam visualisasi data, data diabstraksi dan diringkas. Visualisasi harus melakukan reduksi data, melakukan ransformasi data, dan menampilkan hasil pengolahan data dalam bentuk grafik. Grafik dapat menyajikan hasil analisis data agar mudah untuk dimengerti.

Beberapa teknik visualisasi yang umum digunakan untuk menggambarkan informasi dari sekumpulan data adalah berikut:

- Line chart adalah grafik pembanding perubahan nilai atribut pada periode waktu tertentu.
- Bar chart adalah grafik pembanding nilai atribut untuk jenis kategori yang berbeda.
- Scatter plot adalah grafik dengan plot dua dimensi yang menunjukkan variasi dua item.
- Pie chart adalah grafik untuk membandingkan persentase nilai atribut secara keseluruhan.

1.13.2 Statistika

Statistika adalah suatu metode yang digunakan dalam pengumpulan dan analisis data dapat diperoleh informasi yang bermamfaat. Statistika menyedikan prinsip dan metodologi untuk merancang proses pengumpulan data , meringkas dan menyajikan data yang telah diperoleh , menganalisis dan pengambilan keputusan secara ringkas. Secara ringkas pengertian statistika adalah pengetahuan yang berkaitan dengan pengumpulan angka-angka , pengolahan , dan penganalisisan , penarikan kesimpulan , serta pembuatan keputusan berdasarkan data dan fakta yang sudah dianalisis.

Analisis statistika secara deskriptif merupakan metode penyajian data sehingga memberikan informasi yang berguna. Upaya penyajian ini dimaksudkan untuk mengungkapkan informasi penting yang terdapat dalam data ke dalam berntuk yang lebih ringkas dan sederhana yang pada akhirnya mengarah pada keperluan adanya penjelasan dan penafsiran (Aunudin, 1989). Analisis yang dilakukan dalam statistika meliputi ukuran pemusatan dan penyebaran data. Ukuran pemusatan data meliputi nilai rata-rata (mean), modus, dan median. Sedangkan ukuran penyebaran data meliputi variansi dan standar deviasi

Mean

Mean adalah nilai rata-rata dari sekumpulan data. Nilai dari mean dapat ditentukan dengan cara membagi jumlah data yang ada dengan total data. Mean terdiri dari dua jenis yaitu: populasi dan sampel. Mean populasi memiliki simbol μ, sedangkan mean untuk sampel memiliki simbol $\overline{x}$.

Berikut adalah rumus untuk menghitung mean populasi (μ):

$$\mu = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{N} \tag{1.4}$$

Berikut adalah rumus untuk menghitung mean sampel ($\overline{x}$):

$$\overline{x} = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{N} \tag{1.5}$$

Median

Median menentukan letak dari titik tengah data setelah data disusun menurut urutan nilainya. Median disimbolkan dengan $\widetilde{x}$. Median dibedakan berdasarkan jumlah datanya, yaitu jumlah data yang bernilai ganjil dan jumlah data yang bernilai genap.

Berikut adalah rumus untuk menghitung median dengan jumlah data yang bernilai ganjil dan genap:

$$\widetilde{x} = \begin{cases} X_{\frac{n+1}{2}} & , n = ganjil \\ \frac{X_{\frac{n}{2}} + X_{\frac{n}{2}+1}}{2} & , n = genap \end{cases} \tag{1.6}$$

Modus

Modus adalah nilai yang sering muncul dalam kelompok data tersebut. Modus berarti nilai data yang memiliki frekuensi terbanyak dalam sekelompok data. Jika dalam suatu kelompok data memiliki lebih dari satu nilai data yang sering muncul maka sekumpulan data tersebut memiliki lebih dari satu modus. Sekelompok data yang memiliki dua modus disebut dengan bimodal, sedangkan jika lebih dari dua modus disebut multimodal. Jika dalam sekelompok data tidak terdapat satu pun nilai data yang sering muncul, maka sekelompok data tersebut dianggap tidak memiliki modus. Modus biasanya dilambangkan dengan M_o.

1.13.3 Machine Learning

Machine learning mempelajari metode pembelajaran pada sebuah komputer, agar dapat belajar secara mandiri atau meningkatkan kinerjanya berdasarkan data pelatihan yang pernah diberi. Penelitian terutama pada machine learning adalah membuat program komputer secara otomatis belajar mengenali pola yang kompleks dan membuat keputusan cerdas berdasarkan data. Machine leaning adalah disiplin ilmu yang berkembang dengan cepat. Berikut adalah jenis pembelajaran machine learning untuk mengenali sebuah pola:

Contoh penerapan metode machine Learning adalah supervised learning. Supervised learning adalah permasalahan klasifikasi. Beberapa algoritma klasifikasi yang umum digunakan adalah regression, , artificial neural networks, and random forests. Tujuan klasifikasi adalah untuk mengenali hubungan yang spesifik dalam data input agar secara efektif dapat menghasilkan data output yang benar apabila diberikan data baru. Menentukan output yang benar pada klasifikasi ditentukan dari banyaknya data pelatihan yang diberi. Noisy data sering ditemui pada banyak kasus sehingga perlu ditangani secara khusus dengan metode machine learning tertentu.

Contoh penerapan lain dari metode machine learning adalah unsupervised learning. Unsupervised learning adalah permasalahan pengelompokan/clustering. Unsupervised learning mempelajari sifat data tanpa menggunakan label yang disediakan secara eksplisit. Beberapa algoritma clustering yang umum digunakan adalah k-means clustering. Karena tidak ada label yang disediakan, perlu adanya cara khusus untuk mencari kesamaan antar nilai atribut data menggunakan metode unsupervised learning.

1.14 Privacy Preserving Data Mining (PPDM)

Privacy Preserving Data Mining (PPDM) adalah teknik yang telah dikembangkan untuk mencegah pengungkapan informasi sensitif seseorang saat dilakukan data mining dari sekumpulan data yang berukuran besar. PPDM dapat melakukan perubahan dan menghilangkan sebagian data untuk menjaga privasi. Semakin banyak terjadinya perubahan pada data, maka perlindungan privasi akan meningkat dan kualitas informasi akan menurun. Utilitas adalah kondisi untuk meningkatkan kualitas informasi yang diperoleh dengan mempertimbangkan jumlah data yang dimodifikasi. Metode PPDM menjamin perlindungan data pada tingkat privasi tertentu bersamaan dengan memaksimalkan utilitas data, sehingga memungkinkan pengolahan data mining menghasilkan informasi yang efektif. Klasifikasi PPDM dibuat berdasarkan siklus pengolahan data mulai dari pengumpulan data, proses data mining, penerbitan data, dan distribusi data.

1.14.1 Pengumpulan Data

Untuk memastikan privasi tetap terjada pada saat pengumpulan data, perangkat sensor sebagai reseptor input dapat mengacak nilai data yang ditangkap sebelum data dikirimkan kepada kolektor (perangkat lain). Entitas yang mengumpulkan data diasumsikan menjadi entitas yang tidak dapat dipercaya. Oleh karena itu, nilai-nilai data sesungguhnya tidak pernah disimpan dalam basis data, dan nilai-nilai tersebut hanya dapat digunakan setelah melalui tahap transformasi (saat proses KDD). Metode yang dapat digunakan untuk melindungi atribut sensitif saat pengumpulan data adalah randomisation.

1.14.2 Proses Data Mining

Penambangan data sangat memungkinkan terjadinya identifikasi terhadap atribut sensitif. Berikut adalah beberapa metode yang dapat digunakan untuk melindungi atribut sensitif seseorang saat proses data mining yaitu: association rule hiding adalah aturan untuk mengekstraksi seluruh atribut non-sensitif, downgrading classifier effectiveness adalah teknik untuk menurunkan keakuratan dari classifier yang sering digunakan, query auditing and inference control adalah aturan yang membatasi lingkup penggunaan kueri agregasi berdasarkan dataset, bukan terhadap catatan individu atau kelompok tertentu.

1.14.3 Publikasi Data

Ada kondisi di mana sebuah perusahaan ingin melakukan publikasi koleksi data baik secara publik atau kepada pihak ketiga untuk analisis data tanpa mengungkapkan kepemilikan data sensitif. Dalam situasi ini, PPDM dapat dicapai dengan melakukan anonimisasi pada atribut data yang bersifat sensitif sebelum data tersebut dipublikasikan. PPDM pada publikasi data dikenal sebagai Privacy Preserving Data Publishing (PPDP). Berikut adalah beberapa metode yang dapat digunakan untuk melindungi atribut sensitif saat publikasi data yaitu: k-anonymity, l-diversity, t-closeness, e-differential privacy.

1.14.4 Distribusi Data

Ada kondisi di mana seseorang berusaha untuk melakukan teknik data mining melalui data yang bersifat publik, berdasarkan ketehubungan nilai atribut non-sensitif. Beberapa pendekatan dapat digunakan untuk melindungi atribut sensitif saat distribusi data. Pendekatan pertama adalah menggunakan protokol yang aman untuk mencegah pengungkapan atau perhitungan informasi antar entitas seperti: oblivious transfer protocol dan homomorphic encryption. Pendekatan kedua adalah mempertimbangkan sekumpulan operasi primitif yang sering digunakan pada algoritma data mining seperti: secure sum, secure set union, secure size of intersection, scalar product and the set intersection.

1.15 Jenis Serangan Publikasi Data

Menurut Dalenius (1977), perlindungan privasi tidak akan memberikan kesempatan bagi orang lain untuk mendapatkan informasi sensitif spesifik dari seseorang atau individu meskipun orang lain mengetahui informasi umum yang berhubungan dengan informasi sensitif individu tersebut. Secara umum, orang lain dapat menemukan sebuah cara untuk memetakan sebuah data ke dalam tabel yang telah dianonimisasi ketika data tersebut telah dipublikasikan. Serangan ini dikenal dengan nama linkage attack. Model serangan ini dapat dikategorikan menjadi dua macam, berdasarkan jenis serangannya.

1.15.1 Record Linkage

Record linkage mengacu pada pemetaan beberapa data korban yang ditargetkan ke dalam sebuah tabel yang dirilis secara publik berdasarkan prinsip anonimisasi. Jika pada proses identifikasi salah satu nilai tupel cocok dengan nilai tupel lainnya pada tabel yang sudah dipublikasi, maka memungkinkan atribut sensitif miliki seseorang dapat diketahui oleh orang lain. Menggunakan pemodelan k-anonimity, terbukti dapat menghindari jenis serangan record linkage.

1.15.2 Attribute Linkage

Dalam serangan ini, penyerang mendapatkan beberapa informasi terkait atribut sensitifnya, meskipun penyerang tidak dapat menghubungkan satu tupel dengan tupel lain dari data yang telah dipublikasi. Pemodelan l-diversity dapat mencegah serangan attribute linkage. Kondisi yang diperlukan adalah kesetaraan setidaknya sebuah tupel memiliki l nilai atribut yang berbeda dengan tupel lainnya pada data yang telah dipublikasi. Konsep dasarnya adalah menghindari hubungan atribut jika akan ada nilai sensitif yang bernilai unik pada tupel-tuper tertentu dalam sebuah tabel.

1.16 Anonimisasi

Anonimisasi adalah proses menghilangkan pengidentifikasi pribadi, baik langsung maupun tidak langsung, yang dapat menyebabkan seseorang diidentifikasi. Seseorang dapat diidentifikasi secara langsung dari nama, alamat, kode pos, nomor telepon, foto atau gambar, atau beberapa karakteristik pribadi unik lainnya. Seseorang dapat diidentifikasi secara tidak langsung ketika informasi tertentu dihubungkan bersama dengan sumber informasi lain.

Berikut adalah istilah yang umum digunakan pada proses anonimisasi data:

- Identifier (ID) adalah atribut yang unik dan secara langsung dapat mengidentifikasi seseorang seperti nama, nomor ID, dan nomor ponsel.
- Quasi-Identifier (QID) adalah kombinasi atribut yang mungkin terjadi untuk mengidentifikasi individu berdasarkan penggabungan informasi lain dari luar. Seluruh atribut data terkecuali atribut identifier dapat dianggap sebagai atribut quasi-identifier.
- Equivalence class (EQ) adalah himpunan tupel dengan jenis nilai atribut yang identik satu sama lain.
- Sensitive Attribute (SA) adalah atribut yang menyangkut informasi privasi sensitif seseorang.
- Non-sensitive Attribute (NSA) adalah atribut yang tidak menyangkut informasi privasi sensitif seseorang.

1.16.1 Anonimisasi berdasarkan generalisasi dan supresi

Idenya adalah meningkatkan jumlah equivalence class pada tabel dengan mengurangi nilai akurasi data dari atribut quasi-identifier. Secara umum, quasi-identifier dapat dibagi menjadi dua jenis yaitu atribut numerik dan atribut kategori. Atribut numerik biasanya dilakukan generalisasi secara interval, misalnya, usia 16 dapat digeneralisasi ke interval [10-20]. Dan untuk atribut kategori, nilai-nilai asli akan digantikan oleh yang nilai lebih umum menurut VGH. Sebagai contoh, atribut negara yang nilainya "China", maka dapat digeneralisasi menjadi "Asia". Supresi dapat dipandang sebagai bentuk generalisasi yang ekstrem, di mana atribut yang digeneralisasi tidak dapat digeneralisasi lebih lanjut.

1.16.2 Anonimisasi berdasarkan generalisasi

Saat ini, banyak jenis algoritma yang dipakai untuj implementasi algoritma anonimisasi, dan sebagian besar algoritma menggunakan pendekatan generalisasi dan supresi untuk proses anonimisasi. Berdasarkan perspektif, metode generalisasi dapat dibagi menjadi 2 kategori yaitu global generalization dan local generalization.

Global generalization

Algoritma ini memungkinkan seluruh domain atribut identifier dipetakan ke dalam domain generalisasi, artinya nilai dalam tabel hanya akan memiliki satu nilai. Secara umum, algoritma global generalization dapat diimplementasi secara sederhana dan efisien. Akan tetapi perlu diketahui bahwa generalisasi yang berlebihan dapat menyebabkan hilangnya informasi yang tinggi.

Local generalization

Algoritma ini memetakan seluruh nilai atribut ke dalam nilai generalisasi berdasarkan pengelompokan. Pengelompokan nilai atribut yang sama dapat digeneralisasikan ke nilai atribut yang berbeda jika berada dalam kelompok yang berbeda. Pengelompokan data biasanya mengadopsi beberapa prinsip heuristik, seperti pembagian, pengelompokan, dan sebagainya. Jumlah kehilangan informasi pada jenis algoritma ini lebih kecil dibandingkan dengan algoritma global generalization. Akan tetapi algoritma ini memiliki kompleksitas yang lebih tinggi dibandingkan dengan algoritma global generalization. Dalam kasus sejumlah besar data, kompleksitas adalah masalah yang harus diperhatikan.

Melalui pernyataan diatas, dapat disimpulkan bahwa secara umum algoritma generalisasi global paling efisien, tetapi kualitas informasinya rendah. Sebaliknya, algoritma generalisasi lokal dapat sangat meningkatkan kualitas informasi, tetapi memiliki kompleksitas yang tinggi sehingga tidak efisien.

1.16.3 Anonimisasi berdasarkan clustering

Anonimisasi dapat diimplementasikan dengan pengelompokan/clustering. Clustering merupakan metode yang paling umum digunakan. Ide dasarnya adalah untuk menghasilkan setidaknya k buah tupel kelas sebagai kelas ekivalensi. Tupel dalam kelas yang sama harus sama mungkin untuk membuat kehilangan informasi seminimal mungkin setelah generalisasi. Konsep anonimisasi berdasarkan clustering ada ide dasar untuk perancangan model K-anonimity.

1.17 K-Anonymity

K-anonymity merupakan model yang paling efektif untuk melindungi privasi saat melakukan publikasi data. K-anonymity adalah pemodelan untuk mengurangi perbedaan antara satu data dengan data lain, agar sebuah data tidak dapat dibedakan setidaknya dengan k-1 data lainnya. Dengan kata lain, penyerang tidak dapat mengidentifikasi identitas dari satu data karena k-1 data yang lain memiliki sifat yang sama. Dalam pemodelan k-anonymity, nilai k dapat digunakan sebagai tingkat keamanan privasi. Semakin tinggi nilai k, semakin sulit untuk mengindentifikasi sebuah data. Secara teori, probabilitas identifikasi sebuah data adalah 1 / k. Namun, meningkatkan nilai k juga berpengaruh terhadap nilai informasi yang diperoleh dari sekumpulan data.

K- anonymity pertama kali diusulkan pada tahun 1998 oleh Sweeney et al. K- anonymity tergantung pada menganonimkan kumpulan data asli untuk memenuhi persyaratan anonimisasi, yang dapat digunakan untuk penerbitan data. Teknik anonimisasi yang umum adalah generalisasi dan disembunyikan. Gagasan dasar K- anonymity adalah menganonimkan data penerbitan untuk memenuhi persyaratan bahwa setidaknya K tupel tidak dapat dibedakan satu sama lain. Yaitu, untuk masing-masing tuple terdapat setidaknya K tuple dengan nilai yang sama dari quasi-identifiers. Para peneliti telah membuktikan bahwa kompleksitas K- anonymity adalah NP-hard. NP-hard adalah istilah yang sering digunakan untuk menyatakan kompleksitas sebuah algoritma sangat tinggi (eksponensial).

Penelitian menunjukkan bahwa sebagian besar metode k-anonymity menggunakan metode generalisasi dan supresi. Pendekatan tersebut menderita kehilangan informasi yang signifikan karena mereka sangat bergantung terhadap hubungan relasi antar atribut. Oleh karena itu, hasil anonimisasi menghasilkan nilai kehilangan informasi yang cukup tinggi. Selain itu, algoritma anonimisasi yang ada hanya berfokus pada perlindungan informasi pribadi dan mengabaikan utilitas data yang sebenarnya. Akibatnya, nilai utilitas pada data yang telah dianonimisasi memiliki bernilai rendah. Beberapa algoritma telah dicoba untuk mengimplementasikan pemodelan k-anonymity.

Algoritma k-means clustering akan melakukan beberapa iterasi sampai centroid dari semua data tidak lagi berubah atau perubahannya kecil. Algoritma k-means clustering tidak mampu untuk menyelesaikan masalah pada atribut yang bernilai kategorikal. Kelebihan dari algoritma k-means clustering adalah memiliki hasil pengelompokan yang sudah baik. Kekurangan dari algoritma k-means clustering adalah pemilihan centroid awal k-means secara acak, sehingga setelah digeneralisasi hasil pengelompokannya mengakibatkan hilangnya informasi yang besar.

Algoritma k-member dapat melakukan generalisasi atribut kategorikal dengan memperoleh kualitas informasi yang lebih baik daripada algoritma k-means clustering. Namun algoritma k-member masih memiliki masalah ketika melakukan pengelompokan data. Kekurangan dari algoritma K-member adalah hanya mempertimbangkan pengelompokan terakhir tanpa memperhatikan pengelompokan yang dihasilkan pada proses sebelumnya sehingga menyebabkan distribusi kelompok pada beberapa bagian menjadi kurang tepat.

Untuk menghindari kekurangan pada algoritma k-means dan algoritma k-member maka kedua konsep ini perlu digabung menjadi sebuah algoritma baru dengan nama algoritma Greedy K-member Clustering. Algoritma Greedy K-member Clustering mendapatkan hasil pengelompokan yang lebih tepat dan memiliki nilai informasi yang lebih baik meskipun dilakukan generalisasi. Menggunakan algoritma Greedy K-Member Clustering, pengelompokan data dapat dilakukan satu kali sehingga dapat menurunkan kompleksitas algoritma dan hasil pemilihan centroid dapat dioptimalkan sehingga hasil pengelompokan dapat ditingkatkan secara signifikan.

1.18 Greedy K-Member Clustering

Penelitian menunjukkan bahwa sebagian besar metode k-anonymity didasarkan pada generalisasi dan teknik penekanan sehingga menderita dari kehilangan informasi yang signifikan. Masalah pengelompokan dapat meminimalkan kehilangan informasi. Akan tetapi k-member clustering berpotensi memiliki kompleksitas eksponensial. Untuk menurunkan kompleksitas, maka permasalahan pengelompokan pada algoritma k-member dapat didefinisikan sebagai permasalahan greedy. Algoritma Greedy K-Clustering anggota bertujuan untuk membagi seluruh tuple pada dataset ke masing-masing cluster dengan kerumitan yang lebih baik dan mendukung informasi yang lebih banyak dibandingkan algoritma clustering yang lain.

Teorema 1. *Masalah pengambilan keputusan pada k-member clustering adalah NP-complete.*

Bukti. Melalui pengamatan Aggarwal et al, permasalahan k-member clustering dapat diselesaikan dengan kompleksitas polinomial. □

Teorema 2. *N adalah total tuple dan k adalah parameter anonimisasi. Setiap kluster yang ditemukan oleh algoritma greedy k-member clustering memiliki jumlah tuple minimal sebanyak k, dan jumlah tuple tidak melebihi* $2k\breve{}1$.

Bukti. S adalah himpunan tuple. Algoritma ini menemukan cluster selama jumlah tuple yang tersisa sama dengan atau lebih besar dari k, setiap cluster berisi k tuple. Jika total tuple pada S kurang dari k, maka sisa tuple akan dikelompokan pada kelompok kluster yang sudah ada. Oleh karena itu, ukuran maksimum sebuah cluster adalah $2k - 1$. □

Teorema 3. *N adalah jumlah tuple dan k menjadi parameter anonimitas yang ditentukan. Jika* $n \gg k$*, kompleksitas algoritma greedy k-member clustering adalah* $O(n^2)$.

Bukti. Algoritma greedy k-member clustering menghabiskan sebagian besar waktunya untuk memilih tuple dari S satu per satu hingga mencapai $|S| = k$. Karena ukuran set input berkurang satu pada setiap iterasi, total waktu eksekusi adalah O (n^2). □

Secara singkat, langkah kerja algoritma Greedy K-Means Clustering dapat dijelaskan sebagai berikut:

1. Menghitung jumlah kluster (m) berdasarkan m = pagardalam(n/k)-1
2. Memilih sebuah tupel r secara acak, menjadikan r sebagai centroid dari kluster ci.
3. Setiap (1-r) tupel dalam dataset S, akan dihitung distance antara (1-r) tupel lain dengan centroid r, lalu dipilih sebuah tupel dengan nilai distance terbesar pada centroid r, sehingga centroid r menjadi nilai centroid baru bagi kluster ci.
4. Memilih k-1 tupel lain yang dekat dengan nilai ri, artinya mencari nilai $IL(e1 \cup rj)$ yang minimal.
5. Mengulangi langkaH 4, sampai pada kondisi dimana $|ci| < k$.
6. Mengulangi proses clustering dari tahap 3 sampai dengan tahap 5, sampai pada kondisi dimana $|S| < k$.
7. Melakukan iterasi $(1 - r)$ tupel yang tersisa dalam dataset dan masukan setiap tupel ke dalam sebuah kluster dengan mempertimbangkan nilai information loss seminimal mungkin.

Berikut adalah pseudocode secara lengkap dari algoritma greedy k-member clustering:

Algorithm 1 Find Best Record

```
1: Function find_best_record(S,c)
2: Input: a set of records S and a cluster c.
3: Output: a record r ∈ S such that IL(c ∪ {r}) is minimal
4:
5: n = |S|
6: min = ∞
7: best = null
8: for i = 1 ... n do
9:     r = i-th record in S
10:    diff = IL(c ∪ {r}) − IL(c)
11:    if diff < min then
12:        min = diff
13:        best = r
14:    end if
15: end for
16: return best
```

Algorithm 2 Find Best Cluster

```
1: Function find_best_cluster(C,r)
2: Input: a set of cluster C and a record r.
3: Output: a cluster c ∈ C such that IL(c ∪ {r}) is minimal
4:
5: n = |C|
6: min = ∞
7: best = null
8: for i = 1 ... n do
9:     r = i-th record in S
10:    diff = IL(c ∪ {r}) − IL(c)
11:    if diff < min then
12:        min = diff
13:        best = c
14:    end if
15: end for
16: return best
```

Algorithm 3 Greedy K-Member Clustering

```
1: Function greedy_k_member_clustering(S,k)
2: Input: a set of records S and a threshold value k
3: Output: a set of clusters each of which contains at least k records.
4:
5: if S ≤ k then
6:     return S
7: end if
8:
9: result = φ
10: r = a randomly picked record from S
11: while |S| ≥ k do
12:     r = the furthest record from r
13:     S = S - {r}
14:     c = {r}
15:     while |c| < k do
16:         find_best_record(S, c)
17:         S = S - {r}
18:         c = c ∪ {r}
19:     end while
20:     result = result ∪ {c}
21: end while
22: while S ≠ 0 do
23:     r = a randomly picked record from S
24:     S = S - {r}
25:     find_best_cluster(result, r)
26:     c = c ∪ {r}
27: end while
28: return result
```

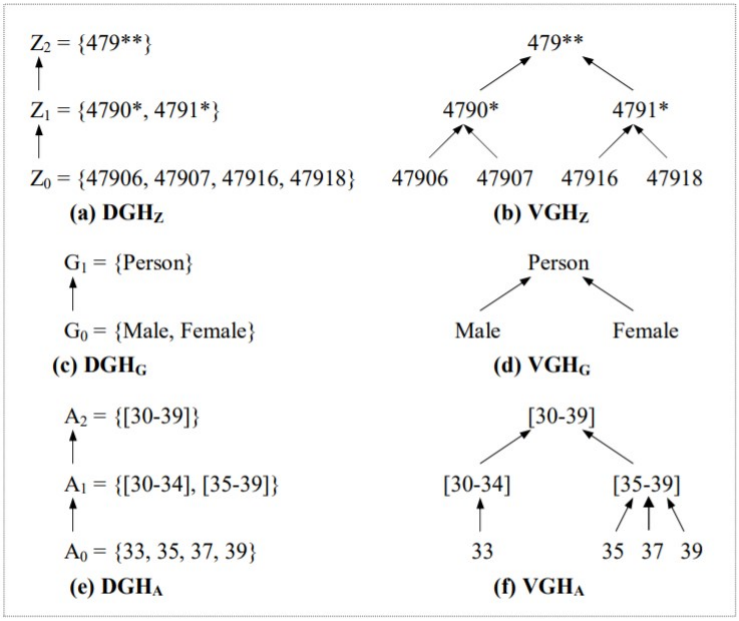
1.19 Hierarchy Based Generalization

Hierarchy-based generalization adalah tahapan anonimisasi setelah data yang memiliki quasi-identifier yang sama dikelompokan ke dalam kelas yang sama. Hierarchy-based generalization menggunakan konsep generalisasi dan supresi dalam melakukan anonimisasi. Hierarchy-based generalization termasuk metode full-domain generalization. Full-domain generalization diusulkan oleh Samarati dan Sweeney untuk memetakan seluruh domain untuk masing-masing atribut quasi-identifier pada tabel ke domain yang lebih umum. Skema ini menjamin bahwa semua nilai atribut tertentu dalam V adalah milik domain yang sama.

Pendekatan dalam kelas ini mensyaratkan bahwa strategi generalisasi dari setiap atribut dalam quasi-identifier harus ditentukan sebelumnya oleh Domain Generalization Hierarchy (DGH), merupakan kumpulan domain yang sepenuhnya menunjukkan atribut yang sesuai harus digeneralisasi dalam beberapa langkah. Berdasarkan DGH, seseorang dapat memperoleh Value Generalization Hierarchy (VGH), merupakan partisi dari urutan himpunan nilai. Pada Gambar 1.6, diberikan contoh perbedaan generalisasi menggunakan DGH dan VGH untuk atribut kode pos pada gambar (a) dan (b), jenis kelamin pada gambar (c) dan (d), dan usia pada gambar (e) dan (f).

Masalah k-anonimisasi adalah menemukan cara generalisasi yang tepat untuk seluruh quasi-identifier, dengan memaksimalkan kualitas data (meminimalkan nilai Information Loss) yang diperoleh saat dilakukan proses data mining. Meskipun solusi optimal k-anonimisasi adalah eksponensial, penelitian sebelumnya telah menghasilkan algoritma yang efisien agar konsep k-anonimisasi dapat diterima karenan memiliki kompleksitas yang cukup baik. Pendekatan Hierarchy-based

generalization memiliki beberapa batasan yang signifikan. Pertama, Hierarchy-based generalization menderita kehilangan informasi yang relatif tinggi karena melakukan generalisasi pada nilai atribut yang berbeda. Keterbatasan lain pada hierarchy-based generalization adalah bahwa utilitas data yang dianonimisasi sangat bergantung pada pemilihan kata. Hierarchy-based generalization bergantung pada pengetahuan pengguna terhadap pengertian kata dari domain tertentu, sehingga sulit untuk mendapat nilai utilitas data dengan hasil yang diharapkan.



Gambar 1.6: Perbedaan DGH dan VGH

1.20 Distance, Information Loss, Cost Function

Konsep PPDM memberikan solusi untuk mengukur tingkat keamanan dan utilitas data berdasarkan data yang telah dianonimisasi menggunakan beberapa jenis metrik. Sebuah metrik tidak dapat mengukur performa algoritma, tingkat keamanan, dan utilitas data secara bersamaan. Untuk mengukur performa algoritma, tingkat keamanan, dan utilitas data secara bersamaan perlu menggunakan beberapa metrik. Beberapa metrik yang umum digunakan pada pengujian kualitas data yang telah dianonimisasi adalah distance,information loss, dan cost function.

Perlindungan privasi seringkali menurunkan utilitas informasi dari sebuah data. Metrik dipakai untuk mengukur banyaknya utilitas data yang hilang terhadap data yang telah dianonimisasi. Secara umum, pengukuran dilakukan dengan membandingkan hasil anonimisasi dengan dataset sesungguhnya. Beberapa atribut yang dapat diukur untuk menyatakan utilitas sebuah data adalah akurasi, kelengkapan, dan konsistensi. Akurasi dapat mengukur seberapa dekat data setelah dilakukan transformasi terhadap data asli. Kelengkapan dapat mencari data yang hilang dalam dataset setelah dilakukan transformasi. Konsistensi dapat menghitung jumlah korelasi yang hilang dalam dataset setelah dilakukan anonimisasi.

1.20.1 Distance

Distance adalah salah satu perhitungan untuk menyatakan akurasi terhadap utilitas sebuah data. Distance merupakan faktor yang paling penting untuk menentukan hasil pengelompokan data. Pemilihan distance yang baik dapat mencapai hasil klasifikasi dengan lebih optimal. Atribut numerik dapat dihitung secara langsung menggunakan Euclidean distance dan Manhattan distance. Distance tidak mampu menangani atribut kategorikal yang bertipe String.

Perhitungan distance dipakai untuk mengukur tingkat perbedaan nilai antara dua jenis data yang berbeda. Perhitungan distance dilakukan berdasarkan pengelompokan tipe data numerik atau kategorikal. Karena masalah k-anonimitas menggunakan atribut numerik dan kategorikal, maka membutuhkan cara khusus untuk mengitung distance dari kedua jenis data pada saat yang sama. Distance ini dihitung berdasarkan representasi nilai numerik pada kelas ekivalensi yang sama. Dengan menggunakan metrik distance, seseorang dapat dengan mudah melakukan pembobotan dengan menambahkan bobot non-negatif untuk setiap atribut.

Distance Data Numerik

Untuk atribut numerik, perbedaan antara dua jenis data menggambarkan ketidaksamaan data. Ketika tupel dalam kelas ekivalensi yang sama digeneralisasi, quasi-identifier yang digeneralisasi harus merangkum semua nilai atribut dalam kelas ekivalensi. Generalisasi dua nilai x dan y dalam atribut numerik biasanya direpresentasikan sebagai rentang $[x, y]$ dimana $x < y$. Dengan demikian, perbedaan tersebut menangkap jumlah perbedaan yang disebabkan oleh proses generalisasi dari atribut masing-masing. Untuk menghitung distance pada atribut numerik, perlu mendefinisikan atribut |D|, dimana |D| adalah jumlah data pada sebuah domain berdasarkan satu atribut numerik. Distance atribut numerik yang sudah dinormalisasi berdasarkan nilai $vi, vj \in D$, dapat dihitung menggunakan rumus dibawah ini:

$$\delta_n(v_1, v_2) = \frac{|v_1 - v_2|}{|D|} \tag{1.7}$$

Distance Data Kategorial

Untuk atribut kategorikal, distance tidak lagi berlaku karena sebagian besar domain kategorikal tidak dapat disebutkan dalam urutan tertentu. Solusi paling mudah adalah mengasumsikan setiap nilai dalam domain yang sama saling berbeda satu sama lain, misalkan jarak antara dua nilai adalah 0 jika kedua nilai bernilai sama, dan 1 jika kedua nilai bernilai beda. Beberapa domain mungkin memiliki beberapa hubungan semantik di antara nilai-nilai tersebut. Dalam domain seperti itu, diinginkan untuk mendefinisikan fungsi jarak berdasarkan hubungan yang ada. Hubungan seperti itu dapat dengan mudah ditangkap di pohon taksonomi. Kami berasumsi bahwa pohon taksonomi suatu domain adalah pohon seimbang yang simpul daunnya mewakili semua nilai berbeda dalam domain tersebut. Untuk beberapa jenis atribut, tidak ada hubungan semantik yang dapat membantu proses klasifikasi nilai domain. Asumsikan bahwa setiap nilai dalam domain yang sama, nilainya saling berbeda satu sama lain. Untuk mengakomodasi semua kasus ini, kami mendefinisikan fungsi jarak untuk nilai kategori sebagai berikut. Untuk menghitung distance pada atribut kategorikal, perlu mendefinisikan atribut |D| dan TD.|D| adalah jumlah data pada domain kategorikal. TD adalah taxonomy tree untuk domain D. Distance atribut kategorial yang sudah dinormalisasi berdasarkan nilai $vi, vj \in D$, dapat dihitung menggunakan rumus dibawah ini:

$$\delta_C(v_1, v_2) = \frac{H(\Lambda(v_i, v_j))}{H(T_D)} \tag{1.8}$$

1.20.2 Information Loss

Information Loss (IL) digunakan untuk mengevaluasi seberapa baik kinerja algoritma k- anonimisasi terhadap utilitas sebuah data. Algoritma K-anonimisasi menyebabkan hilangnya sebagian besar informasi pada dataset. Information Loss (IL) menghitung ketersediaan data setelah melalui tahap anonimisasi secara akurat.

Untuk menghitung Information Loss (IL), perlu mendefinisikan atribut: $e = r_1, \ldots, r_k$ adalah sebuah kluster dengan quasi-identifier yang terdiri dari atribut numerik N1,. . . , Nm dan atribut kategorikal $C_1, \ldots, C_n$. TCi adalah pohon taksonomi yang ditentukan untuk domain atribut

kategorikal Ci. MINNi dan MAXNi adalah nilai minimum dan maksimum pada kluster e berdasarkan atribut Ni. $\cup C_i$ adalah sekumpulan nilai dalam kluster e berdasarkan atribut C_i. Perhitungan nilai Information Loss (IL) terhadap generalisasi kluster e, dilambangkan dengan IL (e). Information loss didefinisikan dengan rumus sebagai berikut:

$$IL(e) = |e| \cdot D(e) \tag{1.9}$$

$$D(e) = \sum_{i=1}^{m} \frac{(MAX_{N_i} - MIN_{N_i})}{|N_i|} + \sum_{j=1}^{n} \frac{H(\Lambda(\cup_{C_j}))}{H(T_{C_j})} \tag{1.10}$$

Untuk menghitung total Information Loss, E adalah set semua kelas ekivalensi kluster e pada tabel anonimisasi AT. Total Information Loss pada tabel AT dilambangkan dengan Total-IL(AT). Total Information Loss didefinisikan dengan rumus sebagai berikut:

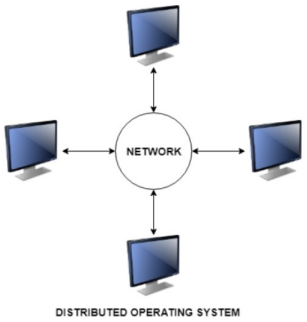
$$Total - IL(AT) = \sum_{e \in \varepsilon} IL(e) \tag{1.11}$$

1.20.3 Cost Function

Cost function menyangkut efisiensi dan skalabilitas dari algoritma yang diterapkan. Metrik ini umum digunakan untuk menyatakan kualitas sebuah algoritma. Untuk mengukur efisiensi, seseorang dapat menggunakan standar untuk menentukan kompleksitas sebuah algoritma berdasarkan faktor waktu dan ruang. Waktu dapat diukur berdasarkan biaya komputasi algoritma. Ruang dapat diukur berdasarkan jumlah pemakaian memori selama menjalankan algoritma dari awal sampai dengan akhir. Kualitas sistem terdistribusi dapat diukur melalui biaya komunikasi antar komputer berdasarkan waktu, jumlah pesan yang dipertukarkan, dan konsumsi bandwidth.

1.21 Sistem Terdistribusi

Sistem terdistribusi adalah kumpulan komputer berjalan secara independen, yang saling terhubung dan saling bekerja sama untuk mencapai satu tujuan yang sama. Menurut Nora Jones (Netflix), sistem terdistribusi berbicara tentang sistem yang sangat rumit, yang terdiri dari banyak bagian yang saling berkomunikasi untuk menghasilkan produk tunggal. Menurut Brendan Burns (Microsoft), "Meningkatnya kekritisan pada sistem kini mengakibatkan munculnya ide pembangunan sistem distribusi untuk mengatasi masalah redundansi, memberikan toleransi kesalahan, dan menjamin tingkat ketersediaan yang tinggi". Beberapa konsep penting terkait sistem terdistribusi adalah setiap komputer bekerja untuk satu tujuan yang sama, setiap komputer mengerjakan tugas yang berbeda dengan komputer lainnya, pembagian jumlah tugas dilakukan secara adil, setiap komputer memiliki jalur komunikasi yang sama untuk saling bertukar informasi.



Gambar 1.7: Sistem Terdistribusi

1.21.1 Manfaat Sistem Terdistribusi

Permasalahan sistem terdistribusi bukan difokuskan terhadap penggunaan sistem terdistribusi, melainkan alasan yang tepat untuk menggunakan sistem terdistribusi. Sistem terdistribusi dibutuhkan karena adanya peningkatan skala dalam pengolahan data dan munculnya sistem yang lebih handal untuk melakukan pemrosesan data.

Berikut adalah deskripsi dari manfaat penggunaan sistem terdistribusi bagi efektivitas pengolahan data:

- Horizontal Scalability
  Sistem terdistribusi menawarkan kemampuan untuk pemrosesan komputasi skala besar dengan murah
- Reliability
  Sistem terdistribusi dapat diandalkan karena proses komputasi pada sistem terdistribusi bergantung pada banyaknya komputer yang saling berkomunikasi satu sama lain untuk mencapai tujuan yang sama.
- Performance
  Sistem terdistribusi dapat menangani proses komputasi tugas secara efisien karena beban kerja sesungguhnya dibagi menjadi beberapa bagian dan tersebar di beberapa komputer.

1.21.2 Tantangan Sistem Terdistribusi

Manfaat penggunaan sistem terdistribusi adalah handal untuk pemrosesan data skala besar. Dalam perancangan sistem terdistribusi, sering ditemui kesulitan pada pemecahan masalah untuk kasus-kasus tertentu.

Berikut adalah tiga tantangan paling umum ditemui ketika merancang sistem terdistribusi:

- Penjadwalan
  Kekuatan komputasi ada batasnya, sistem terdistribusi harus dapat memutuskan pekerjaan mana yang harus dijadwalkan untuk dijalankan dan menentukan prioritas pekerjaan yang dikerjakan lebih dulu.
- Latensi
  Dengan pertukaran data yang semakin kompleks antara perangkat keras, perangkat lunak menggunakan jalur komunikasi jaringan, maka nilai latensi pada sistem tersebut menjadi sangat tinggi. Ketika volume data pada jaringan semakin meningkat, maka perlu dibuat partisi sementara untuk menyimpan data-data yang akan diolah.
- Observasi
  Ketika sistem terdistribusi menjadi kompleks, kemampuan pengamatan untuk memahami kegagalan pada sistem terdistribusi merupakan tantangan besar. Pemantauan terhadap proses partisi data bermanfaat bagi sistem terdistribusi untuk menelusuri tingkat kegagalan proses pada setiap komputer.

1.22 Big Data

Big data adalah data yang besar dan kompleks sehingga tidak mungkin sistem tradisional dapat memproses dan bekerja pada lingkungan data yang besar secara maksimal. Big data dihasilkan oleh mesin dan manusia. Big data dapat dikategorikan menjadi data terstruktur, semi terstruktur dan tidak terstruktur. Big data tidak memungkinkan untuk dianalisis menggunakan kueri pada sistem manajemen basis data tradisional, karena waktu pemrosesan data terlalu lama.

Sekumpulan data dapat dikatakan big data jika memenuhi kelima sifat di bawah ini. Berikut adalah penjelasan kelima sifat yang dimiliki oleh big data:

- Volume
  Volume menyajikan tantangan terbesar untuk melakukan analisis pada data yang besar. Banyak perusahaan memiliki sejumlah besar data yang diarsipkan, tetapi tidak memiliki kapasitas yang memadai untuk memproses data berukuran besar. Kemampuan memproses informasi pada data berukuran besar adalah daya tarik utama untuk menganalisis informasi untuk keperluan tertentu.

- Velocity
  Velocity mengacu pada peningkatan kecepatan ketika data dalam ukuran besar dihasilkan, sehingga membutuhkan peningkatan kecepatan pemrosesan, penyimpanan, dan analisis. Velocity selalu dikaitkan pada kecepatan dimana data baru selalui dihasilkan sepanjang waktu dan kecepatan perpindahan data.

- Variety
  Big data tidak selalu merupakan data terstruktur dan tidak mudah untuk mengolah data besar dengan format file yang beragan. Hal ini menunjukan bahwa keberagaman big data termasuk fakta yang sangat penting dan perlu diketahui untuk sebelum proses analisis data lebih lanjut. Berurusan dengan berbagai data terstruktur dan tidak terstruktur meningkatkan kompleksitas dalam menyimpan dan menganalisis big data. Sebagian besar data yang dihasilkan berbentuk data tidak terstruktur.

- Veracity
  Big data diperoleh dari berbagai macam sumber sehingga kebenaran sebuah data harus diuji. Kebenaran nilai sebuah data dapat teruji apabila mengambil sekumpulan data pada sumber yang terpecaya. Kualitas data yang ditangkap dapat sangat bervariasi. Sehingga keakuratan data yang dianalisis bergantung pada kebenaran nilai data berdasarkan sumber yang beragam.

- Value
  Value adalah aspek terpenting dalam big data. Potensial nilai yang terkandung dalam big data sangat besar. Sangat baik apabila dapat mengakses data-data pada big data, tetapi semuanya itu menjadi tidak berguna apabila data yang diperoleh tidak dapat diubah menjadi sebuah informasi terhadap masalah yang ingin diselesaikan. Value menjadi informasi yang sangat mahal karena memerlukan modal yang cukup besar untuk mengolah dan menyimpan data yang berukuran besar.

1.23 Analisis Big Data

Analisis Big Data adalah proses menggunakan algoritma analisis yang berjalan pada platform pendukung yang kuat untuk mengungkap potensi yang disembunyikan dalam data besar, seperti pola tersembunyi atau korelasi yang tidak diketahui. Menurut pemrosesan persyaratan waktu, analisis big data dapat dikategorikan ke dalam dua paradigma alternatif.

Berikut adalah jenis analisis pada big data:

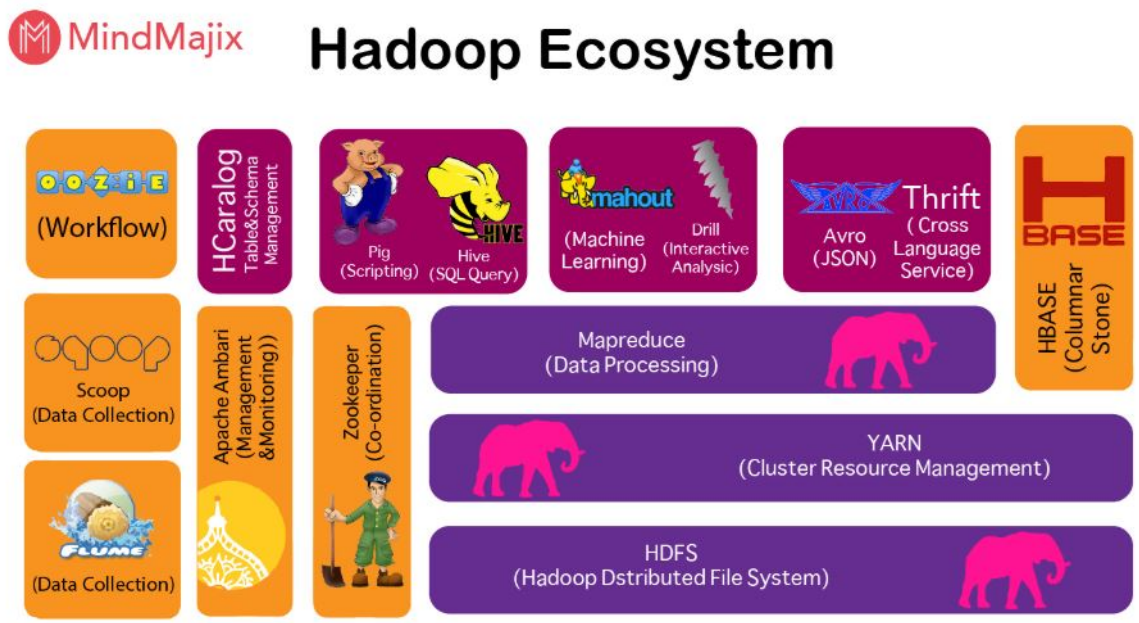
- Batch Processing
  Paradigma batch processing adalah sekumpulan data disimpan terlebih dahulu, lalu pada waktu tertentu data yang telah terkumpul akan dilakukan analisis. MapReduce telah menjadi model penerapan batch processing pada umumnya. Ide dari MapReduce adalah data yang besar dibagi menjadi potongan data yang lebih kecil. Selanjutnya, potongan-potongan data ini diproses secara paralel dan secara terdistribusi untuk dianalisis lebih lanjut. Hasil akhir dari MapReduce diperoleh dengan menggabungkan semua hasil dari setiap node. MapReduce menjadwalkan sumber daya komputasi secara dekat dengan lokasi data itu disimpan, sehingga dapat menghindari overhead dalam transmisi data. Konsep MapReduce banyak diterapkan dalam bioinformatika, data mining website, dan machine learning.

- Streaming Processing
  Paradigma streaming processing adalah diasumsikan bahwa nilai informasi data yang diperoleh bergantung kepada seberapa cepat data dapat diolah secara real time. Pengertian dari real time adalah data diolah secara langsung ketika data tersebut diperoleh. Dalam paradigma ini, data diambil melalui aliran data. Karena aliran data yang masuk sangat cepat dan membawa volume yang cukup besar, maka hanya sebagian kecil data yang dapat disimpan dan diolah dalam memori yang terbatas. Teknologi streaming processing telah dipelajari selama beberapa dekade ini. Sistem pengolahan big data secara open source yaitu Spark dapat menerapkan paradigma ini. Paradigma streaming processing biasanya digunakan untuk analisis aplikasi online dalam tingkat satuan detik maupun milidetik.

1.24 Hadoop

Hadoop adalah framework yang melakukan pemrosesan data dengan memanfaatkan sistem terdistribusi dari kumpulan data besar yang terbagi pada seluruh komputer. Hadoop dikembangkan oleh Google untuk penelitian pada sekumpulan data berukuran besar. Hadoop menggunakan bahasa pemrograman Java. Hadoop dirancang untuk memproses data dalam jumlah yang sangat besar dengan memanfaatkan penyimpanan pada hardisk. Karena pemrosesan pada Hadoop dilakukan berdasarkan paradigma batch processing, sehingga memerlukan waktu pemrosesan data tidak langsung. Hadoop melakukan replikasi file pada masing-masing node untuk menjamin reliability, availability, and fault tolerance.

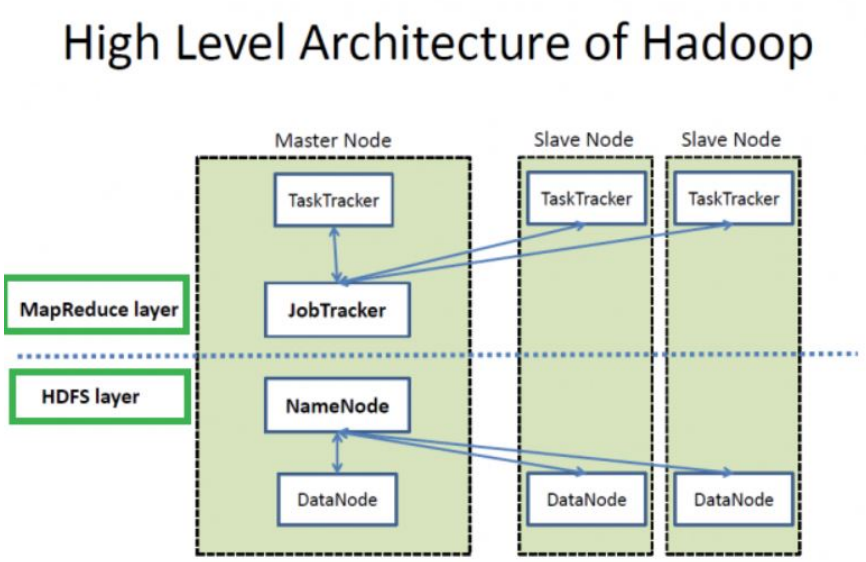
1.24.1 Ekosistem Hadoop



Gambar 1.8: Ekosistem Hadoop

Melalui Gambar 1.8, hadoop bekerja sama dengan teknologi lain untuk memenuhi berbagai macam kebutuhan dalam pengolahan big data. Masing-masing warna pada Gambar 1.8 mewakili tingkat kebutuhan yang berbeda-beda. Warna ungu mewakili komponen pada Hadoop. Warna oranye mewakili teknologi tambahan yang umum digunakan secara bersamaan dengan hadoop. Warna ungu mewakili teknologi khusus yang digunakan secara bersamaan dengan hadoop untuk kasus-kasus tertentu.

1.24.2 Arsitektur Hadoop



Gambar 1.9: Arsitektur Hadoop

Melalui Gambar 1.9, arsitektur Hadoop tersusun dari dua bagian utama yaitu MapReduce dan Hadoop Distributed File System (HDFS). Masing-masing bagian memiliki dua jenis node yaitu master node dan slave node. Master node mengatur jumlah pekerjaan yang diberikan kepada dirinya sendiri dan slave node. Slave node mengerjakan pekerjaan yang diberikan oleh master node.

Hadoop terdiri dari empat elemen penting, yaitu:

1. NameNode
2. DataNode
3. JobTracker
4. TaskTracker

1.24.3 HDFS

HDFS adalah sistem file terdistribusi pada Hadoop dengan menyediakan penyimpanan data yang handal, mendukung partisi, dan toleran terhadap kesalahan pada hardware. HDFS bekerja erat dengan MapReduce dengan mendistribusikan penyimpanan dan perhitungan di seluruh cluster dengan menggabungkan sumber daya penyimpanan yang dapat dipartisi tergantung kebutuhan. HDFS menerima data dalam format apa pun seperti teks, gambar, dan video. Keuntungan utama HDFS adalah memiliki toleransi terhadap kesalahan hardware. Dengan menyediakan transfer data cepat antara node dan memungkinkan Hadoop layanan yang cepat apabila terjadi kegagalan node, sehingga mengurangi risiko terhadao kegagalan komputasi.

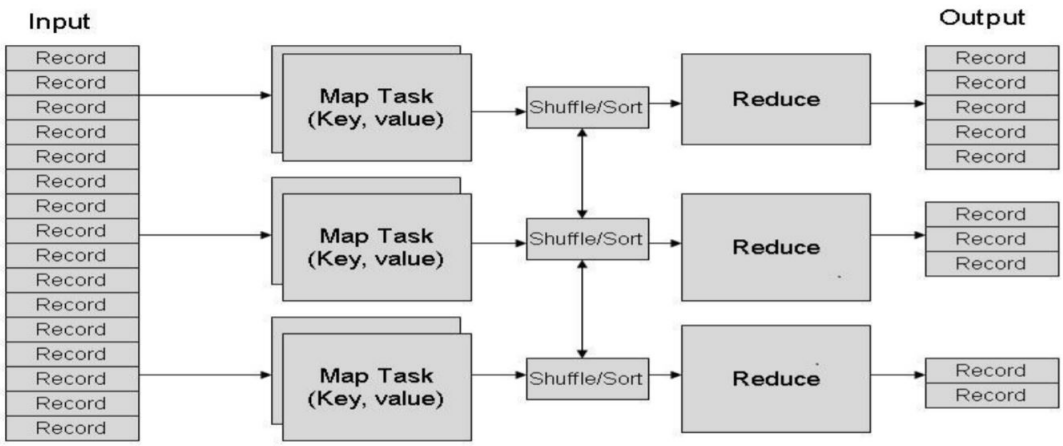
Lapisan penyimpanan HDFS terdiri dari dua elemen:

- NameNode adalah sebuah komputer yang bertindak sebagai master node, sedangkan DataNode adalah komputer-komputer dalam Hadoop Cluster yang bertugas sebagai slaves atau anak buah. NameNode bertanggungjawab menyimpan informasi tentang penempatan block-block data dalam Hadoop Cluster. Ia bertanggungjawab mengorganisir dan mengontrol block-block data yang disimpan tersebar dalam komputer-komputer yang menyusun Hadoop Cluster. Sedangkan DataNode bertugas menyimpan block-block data yang dialamatkan kepadanya, dan secara berkala melaporkan kondisinya kepada NameNode.
- DataNode bertugas menyimpan alamat block-block data, dan secara berkala melaporkan kondisinya kepada NameNode. DataNodes adalah elemen penyimpanan utama HDFS yang menyimpan blok data dan permintaan baca / tulis layanan pada file yang disimpan di HDFS. DataNode dikendalikan oleh NameNode. Blok yang disimpan dalam DataNodes direplikasi sesuai konfigurasi untuk memberikan keandalan dan ketersediaan tinggi. Blok yang direplikasi ini didistribusikan di seluruh cluster untuk memberikan perhitungan yang cepat.

1.24.4 MapReduce

MapReduce adalah kerangka kerja pemrograman untuk komputasi terdistribusi yang dibuat oleh Google menggunakan membagi dan menaklukkan metode untuk memecah masalah data besar yang kompleks menjadi unit-unit kecil pekerjaan dan memprosesnya sejajar. Model pemrograman Map Reduce terdiri dari dua fungsi, yaitu Mapper dan Reducer. Pengguna dapat mengimplementasikan logika pemrosesan mereka sendiri dengan menentukan jenis atribut pada Mapper dan jenis pemrosesan untuk Reducer. Fungsi Mapper dibutuhkan untuk menghasilkan daftar pasangan key,value. MapReduce akan mengelompokkan semua pasangan key,value dan meneruskannya kepada fungsi Reducer untuk mengolah data.

Fungsi Mapper menerima input data berupa blok data dan menghasilkan output data berupa pasangan key-value. Hadoop membuat fungsi mapper untuk setiap blok data HDFS. Jumlah pemanggilan fungsi mapper sama dengan jumlah blok. Fungsi Reducer menerima input data berupa pasangan key-list value dan menghasilkan output data berupa pasangan key-value baru. Pemanggilan fungsi mapper dan fungsi reducer tidak saling bergantung satu sama lain karena komputasi dilakukan secara paralel. Jumlah pemanggilan fungsi reducer dapat ditentukan sesuai kebutuhan.



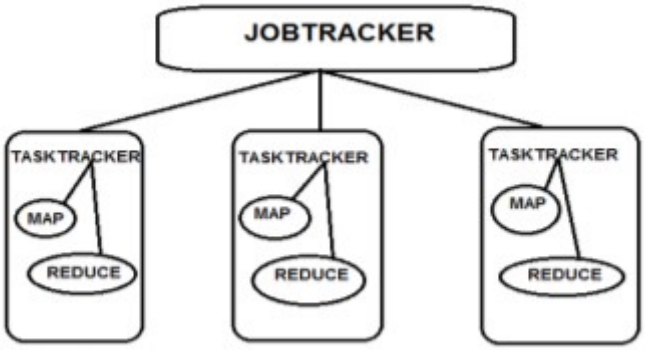
Gambar 1.10: Tahapan pada MapReduce

Pada Gambar 1.10, tahapan MapReduce dapat dibagi menjadi dua fungsi utama:

- Mapper
  Tugas fungsi Mapper adalah memetakan blok data kedalam pasangan <key,value>. Key,value pada Mapper tidak harus memiliki tipe data yang sama satu sama lain. Pasangan <key,value> yang mungkin terjadi pada fungsi Mapper adalah tidak memiliki pasangan atau memiliki banyak pasangan.

- Reducer
  Reducer memiliki 3 fase utama: shuffle, sort dan reducer. Berikut adalah penjelasan dari tahapan yang dilakukan oleh fungsi Reducer:

  1. Shuffle
     Shuffle adalah fase pada data antara untuk menggabungkan semua nilai menjadi koleksi yang terkait dengan yang sama kunci. Setelah ini tidak akan ada kunci duplikat dalam data antara.
  2. Sort
     Pasangan <key,value> pada satu node secara otomatis diurutkan oleh Hadoop sebelum diberikan kepada Reducer. Penyortiran dilakukan berdasarkan keterurutan nilai key. Fase shuffle dan sort terjadi secara bersamaan. Output dari shuffle and sort dari masing-masing node akan diambil dan digabung.
  3. Reducer
     Data output mapper yang diacak dan diurutkan disediakan untuk Reducer. Pada fase ini mengurangi (Writable Metode Sebanding, Iterator, Kolektor Keluaran, Pelapor) dipanggil untuk setiap pasangan <key, (list of value)> dalam input yang dikelompokkan.



Gambar 1.11: JobTracker dan TaskTracker

Pada Gambar 1.11, pekerjaan pada MapReduce dikerjakan oleh dua elemen:

- JobTracker
  JobTracker berjalan pada master node untuk memonitor tugas-tugas MapReduce yang telah dijalankan oleh TaskTracker pada node salve. Tugas JobTracker adalah mengalokasikan pekerjaan yang sesuai untuk TaskTracker tertentu tergantung pada berapa banyak slot tugas yang tersedia. TaskTracker mengatur pelaksanaan setiap operasi MapReduce pada setiap node slave.

- TaskTracker
  TaskTracker berjalan pada masing-masing node slave. TaskTracker menerima pekerjaan dari JobTracker dan menjalankan operasi MapReduce. Setiap TaskTracker memiliki jumlah slot pekerjaan yang terbatas. Tugas masing-masing TaskTracker adalah memulai JVM untuk menjalankan operasi MapReduce secara paralel.

1.25 Spark

Spark adalah teknologi komputasi cluster kilat-cepat, yang dirancang untuk komputasi cepat. Spark adalah paradigma pemrosesan data berukuran besar yang dikembangkan oleh para peneliti University of California di Berkeley. Spark adalah alternatif untuk Hadoop yang dirancang untuk mengatasi keterbatasan pemrosesan input output pada disk dan meningkatkan kinerja sistem sebelumnya. Fitur utama Spark yang membuatnya unik adalah kemampuannya untuk tampil dalam perhitungan memori. Ini memungkinkan data di-cache dalam memori, sehingga menghilangkan overhead disk Hadoop keterbatasan untuk tugas berulang.

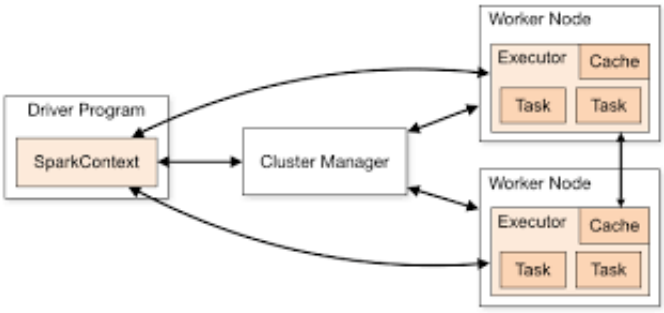
Spark dirancang untuk mencakup pembagian beban kerja secara adil dalam sistem terdistribusi yang terpisah. Dengan memanfaatkan sistem terdistribusi yang ada, Spark menjadi alternatif untuk melakukan komputasi secara paralel dalam analisis data sehingga proses yang besar dapat dilakukan dengan mudah dan murah. Spark bertanggung jawab untuk melakukan penjadwalan, distribusi, dan memonitor jalannya komputasi pada banyak komputer. Spark adalah mesin umum untuk pemrosesan data skala besar yang mendukung Java, Scala dan Python dan untuk tugas-tugas tertentu diuji hingga 100 kali lebih cepat dari Hadoop MapReduce ketika data dapat muat dalam memori dan hingga 10 kali lebih cepat ketika data berada pada disk.

1.25.1 Ekosistem Spark

Gambar 1.12: Ekosistem Spark

Melalui Gambar 1.12, Spark bekerja sama dengan teknologi lain untuk memenuhi berbagai macam kebutuhan dalam pengolahan big data. Masing-masing warna pada Gambar 1.12 mewakili jenis teknologi yang dipakai pada Spark. Spark SQL, Spark Streaming, Spark MLlib adalah library tambahan pada Spark. Cassandra, Kafka, dan ElasticSearch adalah framework untuk melakukan pengumpulan data secara streaming. Sedangkan scala, java, dan python adalah bahasa pemrograman yang dapat digunakan pada Spark.

1.25.2 Arsitektur Spark

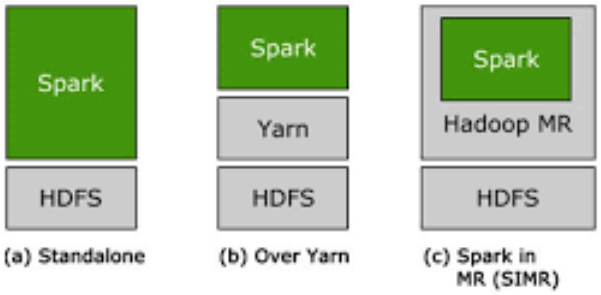


Gambar 1.13: Arsitektur Spark

Beberapa hal penting pada Gambar1.13 terkait arsitektur Spark adalah sebagai berikut:

- Driver Program
  Driver Program adalah komputer yang sedang menjalankan perintah-perintah pada Spark (perintah SparkContext).
- Cluster Manager
  Cluster Manager disebut juga master node, bertugas untuk mengatur jumlah pekerjaan yang diberikan untuk masing node "worker" dan jumlah node "worker" yang diperlukan.
- Worker Node
  Worker Node adalah node yang akan mengerjakan seluruh tugas yang telah diberikan oleh cluster manager.

1.25.3 Jenis Instalasi pada Spark



Gambar 1.14: Arsitektur Spark

Berdasarkan Gambar 1.14, berikut adalah tiga metode yang dapat dilakukan dalam instalasi Spark:

- Standalone
  Spark berdiri diatas HDFS Hadoop. Pada metode instalasi ini, Spark dan Hadoop akan berjalan bersamaan pada mesin yang sama. Selain itu, Spark memungkinkan untuk mengakses data pada HDFS Hadoop untuk membaca input dan menulis output.
- Hadoop Yarn adalah
  Hadoop Yarn berarti Spark bekerja pada Yarn tanpa memerlukan instalasi atau meminta hak akses root apapun. Hadoop Yarn membantu untuk mengintegrasikan Spark pada ekosistem Hadoop dan memungkinkan komponen lain berjalan di atas Spark.
- Spark In MapReduce (SIMR)
  SIMR digunakan untuk menjalankan pekerjaan Spark secara independen. Pada metode instalasi ini, pengguna dapat memulai Spark menggunakan perintah "spark-shell" tanpa perlu ijin dari administrator. Pada versi Spark 2.0 keatas, metode instalasi ini sudah tidak dapat lagi dipakai.

1.25.4 RDD pada Spark

Spark menawarkan abstraksi Resilient distributed Datasets (RDD) untuk mendukung pemrosesan data secara efisien. RDD adalah kumpulan data yang didistribusikan. Dalam Spark semua pekerjaan dinyatakan dengan membuat RDD baru, mengubah RDD yang ada, atau memanggil operasi pada RDD untuk menghitung hasilnya. RDD mendukung dua jenis operasi yaitu transformation dan action. Spark secara otomatis mendistribusikan data dalam RDD di seluruh cluster komputasi dan melakukan komputasi secara paralel. RDD dapat disimpan dalam memori antara banyak kueri tanpa membutuhkan replikasi. RDD dapat membangun kembali data yang hilang pada kegagalan menggunakan riwayat pembentukan setiap RDD mengingat RDD dibangun dari kumpulan data lain (melalui fungsi transformasi seperti map, join, dan group By). RDD memungkinkan Spark untuk mengungguli model yang ada hingga 100x dalam menganalisis data. RDD dapat mendukung berbagai macam algoritma machine learning untuk melakukan teknik data mining secara interaktif dan menyediakan pemrograman SQL yang sangat efisien.

Berikut adalah penjelasan singkat dan contoh dari jenis operasi pada RDD:

- Transformation
  Tranformation adalah operasi pada RDD untuk mengembalikan output berupa RDD baru. Transformatsi pada RDD dilakukan secara lazy, sehingga tranformasti hanya akan dikerjakan apabila dipanggil pada sebuah action. Contoh dari transformasi adalah map(), flatmap(), filter(), groupByKey(), lessByKey(), mapValues().

- Action
  Action adalah operasi yang mengembalikan nilai output ke dalam terminal atau melakukan penulisan data pada sistem penyimpanan eksternal. Action memaksa evaluasi pada RDD yang akan dipanggil, untuk menghasilkan output. Contoh dari action adalah take(), count(), collect(), reduce(), takeOrdered(), top().

1.25.5 Komponen Spark

Komponen Spark adalah library tambahan pada Spark untuk melakukan proses komputasi pada lingkungan big data berdasarkan jenis-jenis kebutuhan pengolahan data. Pada umumnya, Spark hanya membutuhkan library Spark Core saja untuk melakukan komputasi sederhana seperti pemrosesan pada MapReduce. Untuk kasus-kasus tambahan seperti implementasi kueri SQL dan teknik data mining digunakan library tambahan yaitu Spark Core dan Spark SQL.

Berikut adalah penjelasan singkat mengenai jenis library pada Spark:

- Spark Core
  Spark Core adalah library Spark yang berisi fungsionalitas dasar Spark, termasuk komponen untuk penjadwalan tugas, manajemen memori, pemulihan kesalahan, dan berinteraksi dengan sistem penyimpanan. Spark Core juga merupakan rumah bagi API yang mendefinisikan set data terdistribusi yaitu (RDD), yang merupakan abstraksi pemrograman utama Spark. RDD merupakan kumpulan item yang didistribusikan di banyak node komputasi yang dapat dimanipulasi secara paralel.

- Spark SQL
  Spark SQL adalah library Spark untuk bekerja dengan data terstruktur. Spark SQL memungkinkan pemrosesan kueri SQL pada lingkungan big data. Selain itu, Spark SQL menyediakan antarmuka SQL untuk Spark. Hasil pemrosesan kueri pada Spark SQL dapat digabungkan dengan hasil manipulasi data oleh RDD, sehingga memungkinkan untuk melakukan analisis data dengan lebih variatif.

- Spark MLlib
  Spark MLlib adalah library Spark yang berisi fungsionalitas yang umum digunakan pada Machine Learning (ML). Untuk mengimplementasikan teknik data mining pada lingkungan big data dibutuhkan library Spark MLlib. Spark MLlib menyediakan berbagai jenis algoritma Machine Learning termasuk klasifikasi, regresi, pengelompokan/clustering, dan filter kolaboratif, dan fungsionalitas pendukung seperti evaluasi model dan impor data. Semua metode pada Spark MLlib ini dirancang untuk pemrosesan Machine Learning pada lingkungan big data.

1.25.6 Perbandingan Hadoop dan Spark

Berikut ini adalah studi perbandingan analisis data menggunakan Hadoop dan Spark:

Hadoop	Spark
Hadoop tidak efisien untuk pengolahan data dengan berbagi data ke setiap kluster, karena memiliki latensi rendah.	Spark memungkinkan melakukan steream processing pada data yang besar dengan memotong-motong data menjadi ukuran yang lebih kecil yang dapat ditampung oleh memori
Lebih lambat karena hasil pengolahan data disimpan di hard disk.	Memiliki kecepatan 100 kali lebih cepat dibandingkan Hadoop, karena hasil pemrosesan data disimpan dalam memori.
Kompatibel melakukan stream processing, machine learning, dan kueri basisdata.	Kompatibel terhadap Spark streaming, Spark MLLib untuk machine learning, dan Spark SQL untuk melakukan kueri pada data terstruktur.
Pemakaian memori yang lebih rendah.	Pemakaian memori lebih tinggi. Dapat melakukan degradasi dalam membagi pekerjaan apabila data tidak muat di memori.
Output disimpan dalam disk.	Sistem operasi akan memutuskan apakah output data akan disimpan dalam memori atau disimpan dalam hard disk.

1.26 Scala

Scala adalah bahasa pemrograman berbasi open source, dibuat oleh Profesor Martin Odersky. Scala adalah bahasa pemrograman multi-paradigma dan mendukung paradigma fungsional serta berorientasi objek. Apache Spark ditulis dalam Scala dan karena skalabilitasnya di JVM. Pemrograman scala adalah bahasa pemrograman yang paling banyak digunakan, oleh pengembang big data untuk mengerjakan proyek Spark.

Scala digunakan untuk mengakses dan mengimplementasikan fitur-fitur Spark dengan mudah. Kesamaan antara Scala dengan Java adalah memakai konsep object oriented dengan bahasa pemrograman yang lebih disederhanakan. Scala menyediakan jalur terbaik untuk membangun aplikasi big data untuk menangani permasalahan pada kompleksitas program. Scala mendukung konsep parallel dan concurrency untuk pemodelan MapReduce dan sangat cocok untuk implementasi pada pemrograman fungsional. Scala memiliki kelebihan pada concurrency dan library bawaan yang sangat beragam.

Untuk pengembangan Spark, penulisan sintaks Scala dianggap produktif untuk mengimplementasikan kode program. Pemrograman pada Scala mempertahankan prinsip keseimbangan antara produktivitas pengembangan program dan kinerja program. Pemrograman pada Scala tidak serumit pemrograman pada Java. Satu baris kode program pada Scala dapat menggantikan 20 hingga 25 baris kode java. Karena alasan terbut, Scala menjadi bahasa pemrograman yang sangat diminati untuk melakukan pemrosesan big data pada Apache Spark.

1.27 Scala Swing

Scala swing mengimplementasikan program dengan Graphical User Interface (GUI), berbeda dengan program Spark pada umumnya yang hanya melakukan eksekusi dengan menggunakan terminal. Penggunaan terminal terbatas pada penerimaan input dan penampilan output. Dengan menggunakan program berbasis GUI, diharapkan interaksi pada program menjadi lebih beragam. Perbedaan utama antara terminal dan GUI adalah program berbasis terminal menjalankan prosedur

yang tetap, sedangkan program berbasis GUI menjalankan prosedur sesuai dengan kebutuhan penggunanya.

Scala menyediakan akses langsung terhadap kelas GUI Java menggunakan library Scala Swing. Secara konseptual, library Scala Swing menyerupai library Java Swing pada Java. Akan tetapi, Scala Swing lebih menyembunyikan kerumitan kode dibandingkan library Java Swing. Dengan menggunakan Scala, penggunaan library Scala Swing memenuhi kebutuhan perancangan User Interface, dengan tersedianya berbagai macam fitur GUI pada umumnya.

Dasar untuk menyusun aplikasi GUI adalah membangun komponen GUI. Library Scala Swing menyediakan komponen Frame, Panel, Label dan Button yang sama seperti komponen GUI pada umumnya. Komponen pada Scala Swing memiliki properti yang dapat diatur secara manual. Beberapa komponen pada Scala Swing dapat menangani handling event tertentu. Handling event adalah perkerjaan yang dilakukan sebuah komponen, ketika komponen tersebut menerima aksi secara langsung dari pengguna aplikasi.

1.28 Jenis Data Input

Spark menerima berbagai jenis data input dan menfasilitasi penulisan berbagai jenis data output, karena Spark dibangun di atas ekosistem Hadoop. Secara khusus, Spark dapat mengakses data melalui antarmuka format input dan output yang digunakan oleh Hadoop Map Reduce. Berberapa jenis data yang akan diolah memiliki format penulisan masing-masing, sehingga memiliki cara yang berbeda untuk mengambil sekumpulan data dalam format tersebut. Berikut adalah contoh format data yang akan digunakan pada proses pengolahan data.

1.28.1 CSV

Comma Separated Value (CSV) berisi sejumlah jumlah atribut data untuk setiap baris dan dipisahkan oleh koma. CSV tidak memiliki format yang konsisten. Beberapa hal yang sering terjadi terkait penulisan format yang salah pada CSV adalah penggunaan karakter non-ASCII. CSV tidak dapat mengakses nilai atribut data secara langsung. Untuk mendapatkan nilai atribut secara spesifik, perlu dilakukan pemisahan koma untuk setiap baris data. Dalam beberapa tahun terakhir, penggunaan struktur CSV untuk penyimpanan data semakin meningkat. Semakin banyaknya kebutuhan pengolahan data berdasarkan format CSV, maka perlu mencari tahu teknik penyimpanan dan pengambilan data CSV pada Spark. Spark mampu melakukan proses manipulasi dan transformasi data CSV tanpa menggunakan library tambahan.

1.28.2 XML

XML menyediakan format penulisan data menggunakan tag pembuka dan penutup. Tag pembuka dan penutup pada XML ditulis dengan tujuan memberi tahu data-data apa saja yang merupakan cakupan dari elemen tag tersebut. Dalam beberapa tahun terakhir, pengolahan data dengan format XML semakin meningkat seiring berkembangnya pemakaian XML sebagai format penyimpanan data. Oleh karena itu, perlu adanya cara untuk menyimpan dan mengambil nilai sebuah data pada file yang beformat XML. Berkembangan format penyimpanan XML menghadirkan tantangan baru bagi pengolahan big data, karena Spark tidak mampu melakukan proses manipulasi dan transformasi data XML tanpa menggunakan library tambahan.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

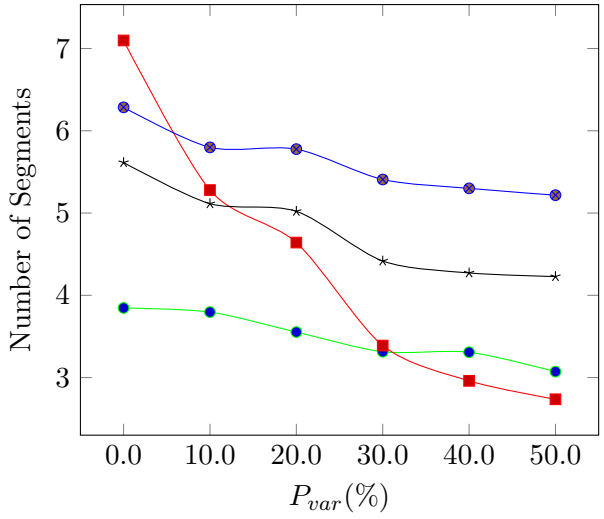
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

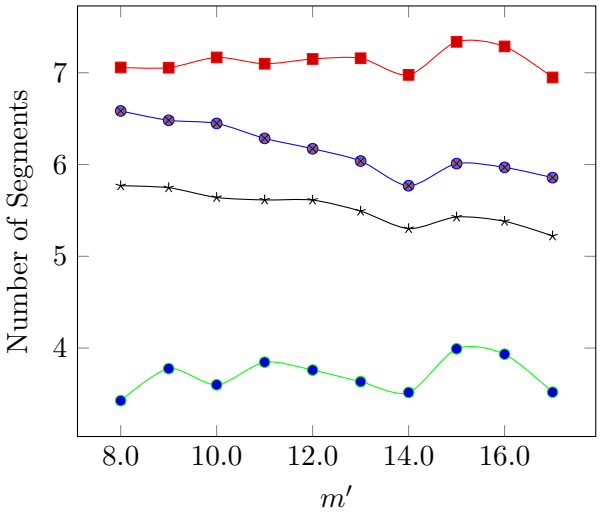
LAMPIRAN B

HASIL EKSPERIMEN

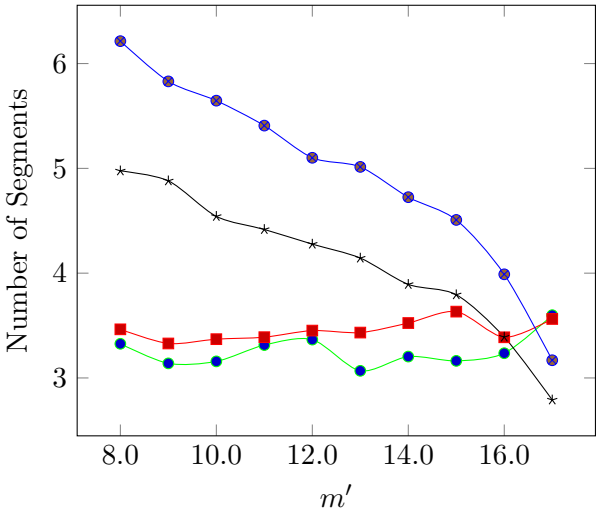
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



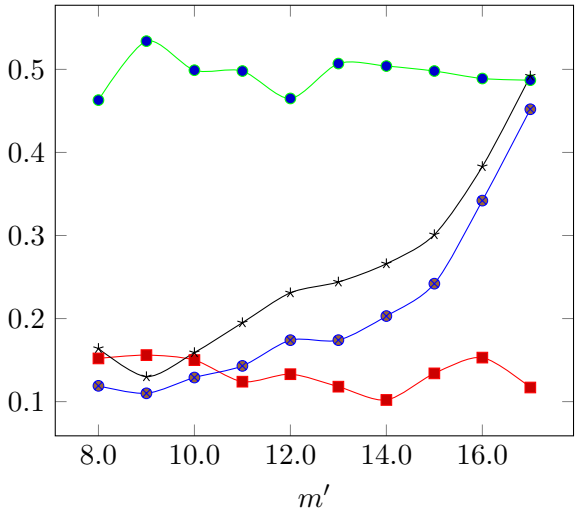
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4