

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-anonymity*. 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-anonymity* 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-anonymity* 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.

BAB 2

DASAR TEORI

2.1 Privasi

Privasi adalah suatu keadaan dimana kehidupan pribadi seseorang atau sekelompok orang terbebas dari pengawasan atau gangguan orang lain. Privasi juga dapat berarti kemampuan satu atau sekelompok individu untuk menutupi atau melindungi kehidupan dan urusan personalnya dari publik dengan mengontrol sumber-sumber informasi mengenai diri mereka. Untuk melakukan publikasi data dari satu perusahaan ke perusahaan lain, digunakan teknik anonimisasi data untuk melindungi dan menyamarkan atribut sensitif untuk setiap data.

Personally Identifiable Information (PII) adalah standar yang digunakan untuk menentukan apakah informasi yang ada dapat melakukan identifikasi entitas individu secara langsung 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. PII adalah atribut yang biasanya terjadi pelanggaran data dan pencurian identitas. Jika data perusahaan atau organisasi terungkap, maka data pribadi pelanggan sangat mungkin terungkap. PII yang diketahui dapat dijual dan digunakan untuk melakukan pencurian identitas, menempatkan korban dalam risiko.

Berikut adalah contoh informasi yang bersifat sensitif menurut standar PII:

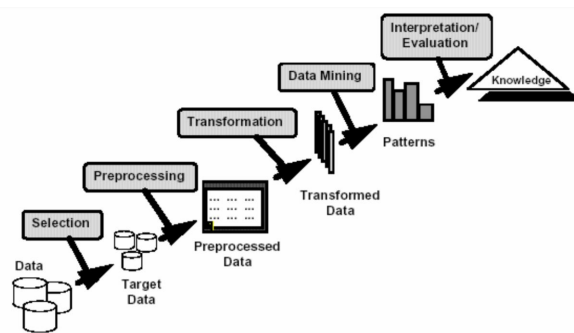
- Identitas diri
Nama lengkap, tempat tanggal lahir, alamat rumah, alamat email.
- Nomor identitas diri
NIK, nomor passport, nomor SIM, nomor wajib pajak, nomor rekening, nomor telepon, dan nomor kartu kredit.
- Karakteristik pribadi
Foto diri, sidik jari, dan tulisan tangan.
- Data biometrik
Pemindaian retina, jenis suara, dan geometri wajah.
- Aset informasi lainnya
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 finansial
- Letak geografis

2.2 Data Mining

Data yang dikumpulkan bertambah banyak, sehingga 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 2.1: Tahapan pada KDD

Berikut ini adalah penjelasan tahapan pada KDD pada Gambar 2.1 sebagai berikut:

1. Selection: proses mengambil data yang relevan terhadap analisis.
2. Preprocessing: proses pembersihan data dari data yang tidak konsisten dan integrasi data saat penggabungan data.
3. Transformation: proses manipulasi data menggunakan konsep agregasi, generalisasi, normalisasi, dan reduksi untuk kebutuhan analisis.
4. Data Mining: proses ekstraksi informasi menggunakan metode pengenalan pola seperti klasifikasi, pengelompokan/clustering.
5. Interpretation/Evaluation: proses interpretasi hasil pengolahan data menjadi sebuah grafik yang dapat dimengerti.

Berikut adalah beberapa jenis tipe data terkait teknik data mining:

- Binary: tipe data alphabet/numerik yang hanya memiliki 2 kemungkinan nilai.
Contoh: diadakan survei evaluasi beberapa produk pakaian untuk mengetahui produk yang diminati dan tidak diminati. Penilaian produk dapat diwakilkan nilai True atau False. True atau False termasuk jenis binary.
- Nominal: tipe data alphabet/numerik yang memiliki lebih dari 2 kemungkinan nilai.
Contoh: seseorang memilih beberapa bahan dari warna yang berbeda. Warna yang mungkin adalah kuning, hijau, hitam, merah. Warna termasuk jenis nominal.

Tujuan dari penggunaan teknik data mining adalah sebagai berikut:

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

2.2.1 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.

Berikut adalah tahapan klasifikasi secara umum:

1. Pelatihan: proses konstruksi model klasifikasi menggunakan algoritma tertentu. Algoritma digunakan untuk membuat model belajar menggunakan set pelatihan data yang tersedia. Model dilatih untuk menghasilkan prediksi yang akurat.
2. Klasifikasi: model yang digunakan untuk memprediksi label kelas dan menguji model yang dibangun pada data uji dan karenanya memperkirakan akurasi aturan klasifikasi.

Berikut adalah kategori pemodelan klasifikasi:

- Discriminative: pemodelan paling mendasar untuk menentukan satu kelas untuk setiap baris data. Pemodelan ini bergantung pada data yang diamati dan sangat bergantung pada kualitas data daripada distribusi data.

```
..
Student 1 : Test Score: 9/10, Grades: 8/10  Result: Accepted
Student 2 : Test Score: 3/10, Grades: 4/10, Result: Rejected
Student 3 : Test Score: 7/10, Grades: 6/10, Result: to be tested
```

Gambar 2.2: Contoh soal

Contoh: Logistic Regression

Penerimaan siswa pada sebuah Universitas, untuk mempertimbangkan test score dan grades terhadap keputusan seorang siswa diterima/tidak diterima.

- Generative: pemodelan ini memodelkan distribusi kelas individu dan mencoba mempelajari model yang menghasilkan data dengan memperkirakan asumsi dan distribusi model. Digunakan untuk memprediksi nilai data yang belum diketahui.

Contoh: Naive Bayes

Mendeteksi email spam dengan melihat data sebelumnya. Misalkan dari 100 email yang ada dibagi menjadi kategori Kelas A: 25% (Email spam) dan Kelas B: 75% (Email Non-Spam). Ingin diperiksa apakah email berisi spam atau bukan. Pada Kelas A, 20 dari 25 email adalah spam dan sisanya bukan spam. Pada Kelas B, 70 dari 75 email bukan spam dan sisanya adalah spam. Probabilitas email yang berisi spam termasuk pemodelan naive bayes.

Berikut adalah contoh pemodelan yang umum digunakan:

- Decision Trees
- Naive Bayes
- Neural Networks
- K-Nearest Neighbour
- Linear Regression

2.2.2 Naive bayes

Naive bayes menerapkan klasifikasi dengan menggunakan metode probabilitas dan statistik. Pe-modelan ini mencari nilai probabilitas tertinggi pada masing-masing kelas menggunakan teorema Bayes. Kelas dengan probabilitas tertinggi akan dipilih sebagai hasil akhir. Naive bayes mudah untuk dibangun dan memiliki komputasi yang lebih cepat daripada model klasifikasi lainnya.

Teorema Bayes menemukan probabilitas suatu peristiwa terjadi mengingat probabilitas peristiwa lain yang telah terjadi. Teorema Bayes dinyatakan secara matematis melalui persamaan berikut:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)} \quad (2.1)$$

Dari perhitungan probabilitas teorema Bayes, akan dicari kelas dengan probabilitas maksimum. Probabilitas maksimum dapat dinyatakan secara matematis melalui persamaan berikut:

$$MAP(H) = \max(P(H|D)) \quad (2.2)$$

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.

Dataset diberikan untuk menggambarkan kondisi cuaca saat bermain golf. Masing-masing data dikategorikan berdasarkan nilai atribut PlayGolf, yaitu cocok ("Yes") atau tidak cocok ("No").

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Gambar 2.3: Dataset untuk kondisi cuaca saat bermain golf

Berikut adalah pengelompokan nilai berdasarkan dataset yang telah diberikan:

- **Vektor Fitur**
Vektor Fitur adalah vektor yang mewakili nilai fitur untuk setiap baris dataset. Vektor Fitur dalam dataset ini tersusun dari nilai atribut Outlook, Temperature, Humidity, dan Windy.
- **Vektor Respon**
Vektor Respon adalah nilai prediksi kelas untuk setiap vektor fitur. Vektor Respon dalam dataset ini diwakili oleh nilai atribut PlayGolf.

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

1. Merepresentasikan teorema Bayes terhadap vektor fitur.
Berdasarkan dataset, teorema Bayes dapat diubah seperti berikut:

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)} \quad (2.3)$$

Dimana y adalah variabel kelas dan X adalah vektor fitur (dengan ukuran n), dinyatakan melalui persamaan berikut:

$$X = (x_1, x_2, x_3, \dots, x_n) \quad (2.4)$$

Contoh: $X = (\text{Rainy}, \text{Hot}, \text{High}, \text{False})$, $y = \text{No}$

Diasumsikan teorema Bayes saling independen terhadap fitur-fiturnya. Berikut adalah persamaan teorema Bayes baru, jika memakai lebih dari satu nilai atribut:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)} \quad (2.5)$$

2. Menghitung probabilitas masing-masing atribut.

Outlook				
	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature				
	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity				
	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind				
	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Gambar 2.4: Menghitung Probabilitas

Contoh: menghitung $P(No)$ untuk nilai "Sunny" pada atribut "Outlook"

$$P(No) = \frac{frekuensi(Sunny \cap No)}{frekuensi(No)} \quad (2.6)$$

Contoh: menghitung $P(Yes)$ untuk nilai "Sunny" pada atribut "Outlook"

$$P(Yes) = \frac{frekuensi(Sunny \cap Yes)}{frekuensi(Yes)} \quad (2.7)$$

3. Menghitung probabilitas bersyarat jika diketahui nilai dari data baru.

Contoh: today = (Sunny, Hot, Normal, False)

$$P(Yes|today) = \frac{P(Sunny|Outlook|Yes)P(Hot|Temperature|Yes)P(Normal|Humidity|Yes)P(False|Wind|Yes)P(Yes)}{P(today)}$$

$$P(Yes|today) = \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{5}{14} = 0.0068 \quad (2.8)$$

$$P(No|today) = \frac{P(Sunny|Outlook|No)P(Hot|Temperature|No)P(Normal|Humidity|No)P(False|Wind|No)P(No)}{P(today)}$$

$$P(No|today) = \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} = 0.0068 \quad (2.9)$$

4. Melakukan normalisasi terhadap probabilitas bersyarat.

Setelah probabilitas bersyarat dinormalisasi, akan menjadi seperti berikut:

$$P(Yes|today) = \frac{0.0141}{0.0141 + 0.0068} = 0.67 \quad (2.10)$$

$$P(No|today) = \frac{0.0068}{0.0141 + 0.0068} = 0.33 \quad (2.11)$$

Sehingga memiliki probabilitas total seperti berikut:

$$P(Yes|today) + P(No|today) = 1 \quad (2.12)$$

5. Mencari probabilitas tertinggi.

Berdasarkan pernyataan berikut:

$$P(Yes|today) > P(No|today) \quad (2.13)$$

Dapat disimpulkan bahwa, jika diberikan data dengan nilai (Sunny, Hot, Normal, False) klasifikasi yang tepat untuk atribut PlayGolf adalah Yes.

2.2.3 Pengelompokan/Clustering

Clustering adalah salah satu teknik analisis data yang paling umum digunakan untuk mendapatkan kemiripan antar data. Clustering dapat didefinisikan sebagai sebuah tugas untuk mengidentifikasi subkelompok dalam data sedemikian rupa sehingga titik data dalam subkelompok yang sama (cluster) sangat mirip sedangkan titik data dalam kelompok berbeda sangat berbeda. Contoh pemodelan clustering adalah K-Means.

2.2.4 K-Means

K-Means adalah algoritma pembelajaran mesin unsupervised learning untuk menentukan kelompok objek tertentu benar-benar milik. Unsupervised learning artinya tidak ada label yang ditentukan dalam data. Gagasan utama K-Means adalah menetapkan setiap data ke dalam cluster dengan mean terdekat (centroid). Mencari titik terdekat dilakukan dengan cara menghitung distance antara dua data menggunakan Euclidean distance, lalu membandingkan titik yang memiliki jarak paling dekat dengan titik lainnya.

Berikut adalah persamaan untuk menghitung Euclidean distance:

$$EuclidDist(p_i, C_i) = \sqrt{(p_1 - C_1)^2 + (p_2 - C_2)^2 + \dots + (p_n - C_n)^2} \quad (2.14)$$

Diberikan dataset skor A dan B untuk masing-masing individu:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Gambar 2.5: Contoh Dataset

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

1. Kumpulan data ini akan dibuat menjadi dua kelompok dengan menginisialisasi $k = 2$. Untuk menentukan titik centroid awal, akan dicari nilai A dan B terjauh dengan data lainnya menggunakan Euclidean distance.

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Gambar 2.6: Hasil Pengelompokan Awal

2. Data yang tersisa akan diperiksa secara berurutan dan dialokasikan pada cluster yang paling dekat dengan centroid awal menggunakan Euclidean distance. Vektor rata-rata (centroid) akan dihitung ulang setiap kali anggota baru ditambahkan.

Step	Cluster 1		Cluster 2	
	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Gambar 2.7: Mencari Centroid Kelompok

3. Menentukan titik centroid baru pada cluster yang baru terbentuk dari tahap sebelumnya.

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

Gambar 2.8: Hasil Pengelompokan Baru

4. Belum bisa dipastikan bahwa setiap individu telah dialokasikan pada cluster yang tepat. Oleh karena itu, perlu membandingkan distance masing-masing data dengan centroid baru pada masing-masing kelompok.

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Gambar 2.9: Euclidean Distance Cluster 1, Cluster 2

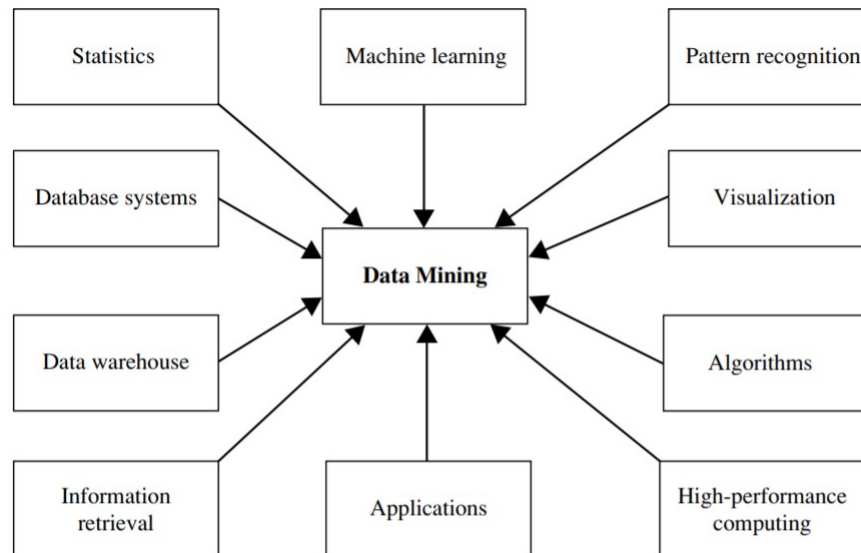
5. Dapat disimpulkan bahwa, hanya individu 3 yang jaraknya lebih dekat dengan centroid Cluster 2 dari pada centroid Cluster 1. Dengan kata lain, distance masing-masing individu ke centroid kelompoknya sendiri harus lebih kecil daripada rata-rata kelompok lain. Dengan demikian, individu 3 harus dialokasikan ke Cluster 2.

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

Gambar 2.10: Hasil Pengelompokan Akhir

2.3 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 2.11: Jenis bidang terkait data mining

2.3.1 Visualisasi Data

Visualisasi adalah penggunaan representasi grafik pada komputer. Visualisasi data melibatkan penyajian data dalam bentuk grafik atau gambar agar membuat informasi mudah dimengerti, membantu menjelaskan fakta, dan menentukan arah tindakan. 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.

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

- Line chart adalah grafik pembandingan perubahan nilai atribut pada periode waktu tertentu.
- Bar chart adalah grafik pembandingan 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.

2.3.2 Statistika

Statistika adalah pengetahuan yang berkaitan dengan pengumpulan angka-angka, pengolahan, dan analisis, penarikan kesimpulan, dan membuat keputusan berdasarkan data dan fakta yang sudah dianalisis. 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 \bar{x} .

Berikut adalah persamaan untuk menghitung mean populasi (μ):

$$\mu = \frac{x_1 + x_2 + x_3 + \dots + x_n}{n} \quad (2.15)$$

Berikut adalah persamaan untuk menghitung mean sampel (\bar{x}):

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \dots + x_n}{n} \quad (2.16)$$

Median

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

Berikut adalah persamaan median untuk jumlah data ganjil dan genap:

$$\tilde{x} = \begin{cases} X_{\frac{n+1}{2}} & , n = \text{ganjil} \\ \frac{X_{\frac{n}{2}} + X_{\frac{n}{2}+1}}{2} & , n = \text{genap} \end{cases} \quad (2.17)$$

Modus

Modus adalah nilai yang sering muncul dalam kelompok data tersebut. Modus berarti nilai data yang memiliki frekuensi terbanyak dalam sekelompok data.

2.3.3 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 pada machine learning adalah membuat program komputer bekerja secara otomatis untuk belajar mengenali pola yang kompleks dan membuat keputusan cerdas berdasarkan data.

Berikut adalah contoh pembelajaran pada Machine Learning:

- Supervised Learning adalah pembelajaran dengan menggunakan data yang diberi label dengan baik yang berarti beberapa data sudah ditandai dengan jawaban yang benar. Supervised Learning menganalisis data pelatihan dan menghasilkan hasil pelatihan yang benar dari data yang diberi label. Contoh penerapannya adalah clustering.
- Unsupervised Learning adalah pembelajaran menggunakan informasi yang tidak diklasifikasi atau diberi label dan memungkinkan algoritma bertindak pada informasi tersebut tanpa bimbingan. Unsupervised Learning bertujuan untuk mengelompokkan informasi yang tidak disortir menurut persamaan, pola dan perbedaan tanpa pelatihan data sebelumnya. Contoh penerapannya adalah klasifikasi.

2.4 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.

2.4.1 Pengumpulan Data

Untuk memastikan privasi tetap terjaga 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.

2.4.2 Proses Data Mining

Penambahan 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.

2.4.3 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.

2.4.4 Distribusi Data

Ada kondisi di mana seseorang berusaha untuk melakukan teknik data mining melalui data yang bersifat publik, berdasarkan ketahanan 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.

2.5 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.

2.5.1 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 milik seseorang dapat diketahui oleh orang lain. Menggunakan pemodelan k-anonymity, terbukti dapat menghindari jenis serangan record linkage.

2.5.2 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.

2.6 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 beberapa 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.

2.6.1 Anonimisasi berdasarkan generalisasi dan supresi

Idenya adalah meningkatkan jumlah equivalence class pada tabel dengan mengurangi nilai akurasi data dari atribut quasi-identifier. Quasi-identifier dikategorikan menjadi atribut numerik dan kategorial. Atribut numerik digeneralisasi secara interval, misalnya usia 16 dapat digeneralisasi pada interval [10-20]. Atribut kategorial mengubah nilai asli ke nilai yang lebih umum, misalnya negara "China" digeneralisasi menjadi "Asia". Supresi dipandang sebagai bentuk generalisasi yang ekstrem, karena menghilangkan beberapa nilai atribut.

2.6.2 Anonimisasi berdasarkan generalisasi

Saat ini, banyak jenis algoritma yang dipakai untuk implementasi anonimisasi. Berdasarkan perspektif, metode generalisasi dapat dibagi menjadi global generalization dan local generalization.

2.6.3 Anonimisasi berdasarkan clustering

Anonimisasi dapat diimplementasikan dengan konsep clustering. Ide dasarnya adalah menghasilkan setidaknya k buah data. Data dalam kelompok yang sama harus semirip mungkin agar kehilangan informasi dapat diminimalkan setelah dilakukan generalisasi. Konsep ini menjadi ide dasar untuk perancangan model K -anonymity.

2.7 Hierarchy Based Generalization

Hierarchy-based generalization adalah tahapan anonimisasi setelah data yang memiliki quasi-identifier yang sama dikelompokkan 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.

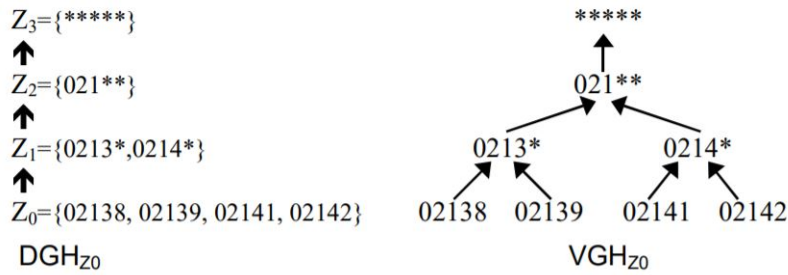
Full-domain generalization menetapkan bahwa metode generalisasi dapat dilakukan jika quasi-identifier telah ditentukan sebelumnya. Sebagai contoh $QI = \{A_1, \dots, A_n\}$, dimana A adalah atribut dari tabel dataset. Full-domain generalization digunakan oleh K -Anonymity untuk menentukan generalisasi dan supresi dari sebuah nilai. Terdapat dua jenis hierarki pada Full-domain generalization yaitu Domain Generalization Hierarchy (DGH) dan Value Generalization Hierarchy (VGH). Jenis hierarki yang paling umum digunakan adalah DGH. Tidak terdapat aturan khusus untuk memodelkan hierarki DGH. Semua nilai atribut dalam tabel harus bisa digeneralisasi menggunakan DGH. DGH adalah konsep sederhana dari penggantian nilai berdasarkan nilai yang kurang spesifik menjadi nilai lebih umum terhadap nilai aslinya.

Pada Gambar 2.12, nilai ZIP {02138, 02139} dapat digeneralisasi menjadi {0213*} dengan menghilangkan digit paling kanan. Secara semantik nilai yang telah digeneralisasi akan memiliki lingkup nilai yang lebih besar. Pada basis data relasional, domain digunakan untuk merepresentasikan nilai dari masing-masing atribut. Gagasan tentang domain akan diperluas agar lebih mudah dalam memahami cara kerja generalisasi. Dalam basis data, nilai telah dicatat secara spesifik sehingga dapat diasumsikan nilai atribut masih berada pada domain dasar. Sebagai contoh, nilai ZIP adalah {02138, 02139} berada pada domain dasar yaitu Z_0 . Untuk mencapai perlindungan pada K -Anonymity, maka kode ZIP yang sebelumnya representatif harus diubah menjadi kurang informatif. Sebuah nilai dapat digeneralisasi jika memiliki domain yang lebih umum. Sedangkan domain yang kurang spesifik dapat digunakan untuk mendeskripsikan garis besar nilai ZIP. Sebagai contoh dari domain kurang spesifik adalah Z_1 , di mana digit terakhir diganti dengan simbol (*). Berikut adalah contoh pemetaan dari Z_0 ke Z_1 seperti berikut $02139 \rightarrow 0213^*$.

Jika diberikan atribut A , maka fungsi generalisasi untuk atribut A adalah $f: A \rightarrow B$. Gambar 2.1 menyatakan urutan generalisasi secara fungsional dengan definisi fungsi sebagai berikut $f_h: h = \{0, \dots, n-1\}$, sehingga $A = A_0$ and $|A_n| = 1$. Fungsi f_h menerapkan urutan linier pada A_h di mana elemen minimal harus berada pada domain dasar yaitu A_0 dan elemen maksimalnya harus berada paling atas yaitu A_n . Persyaratannya adalah A_n memastikan bahwa semua nilai yang terkait dengan suatu atribut bisa digeneralisasikan ke nilai tunggal. $A_h, h = 0, \dots, n$ akan diuraikan jika pada implementasi nilainya bertentangan dengan memiliki nilai yang sama. Hubungan seperti itu menyiratkan adanya hierarki generalisasi nilai VGHA untuk atribut A .

$$A_0 \xrightarrow{f_0} A_1 \xrightarrow{f_1} \dots \xrightarrow{f_{n-1}} A_n$$

Representasi generalisasi diperluas agar metode supresi dapat diterapkan dengan cara membuat hirarki generalisasi untuk elemen dengan nilai maksimal yang baru berada di atas elemen dengan nilai maksimal yang lama. Elemen maksimal baru adalah nilai supresi atribut. Ketinggian setiap hierarki generalisasi akan bertambah seiring munculnya elemen maksimal yang baru. Setelah elemen mencapai nilai maksimal yang dapat digeneralisasi maka tidak akan ada lagi perubahan yang diperlukan untuk memasukkan nilai supresi (*). Gambar 2.12 dan Gambar 2.13 memberikan contoh hierarki generalisasi dari nilai dan domain yang diperluas untuk memasukkan elemen maksimal yang disupresi (*****). Pada contoh ini, domain Z_0 mewakili kode ZIP untuk Cambridge, MA, dan E_0 mewakili ras. Mulai sekarang, semua referensi untuk generalisasi termasuk elemen maksimal baru dan hierarki mengacu pada hierarki generalisasi domain.



Gambar 2.12: DGH dan VGH pada atribut ZIP



Gambar 2.13: DGH dan VGH pada atribut Race

2.8 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 mengidentifikasi 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 buah record yang 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 pemodelan 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. Hasil akhir dari algoritma Greedy K-member Clustering adalah data-data yang sejenis sudah dikelompokkan pada kelompok data yang sama. Untuk melakukan anonimisasi pada pemodelan k-anonymity, digunakan konsep Hierarchy Based Generalization. Konsep ini menyamakan data yang memiliki nilai quasi-identifier yang sama. Beberapa pemodelan yang dapat digunakan adalah Domain Generalization Hierarchy (DGH) dan Value Generalization Hierarchy (VGH). Pemodelan VGH akan diterapkan pada data yang nilainya sama dengan data lain dalam satu kelompok.

2.9 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 melalui algoritma K-Member Clustering. Akan tetapi algoritma K-Member Clustering berpotensi memiliki kompleksitas yang eksponensial. Untuk menurunkan kompleksitas tersebut, maka permasalahan algoritma K-Member Clustering dapat didefinisikan sebagai permasalahan algoritma Greedy. Algoritma Greedy K-Member Clustering bertujuan untuk membagi seluruh tuple pada dataset ke masing-masing cluster dengan kompleksitas 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. \square

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 $2k - 1$.*

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 dikelompokkan pada kelompok kluster yang sudah ada. Oleh karena itu, ukuran maksimum sebuah cluster adalah $2k - 1$. \square

Teorema 3. *N adalah jumlah tuple dan k menjadi parameter anonimitas yang ditentukan. Jika $n > k$, 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 $O(n^2)$. \square

Beberapa hal penting terkait algoritma Greedy K-Means Clustering:

- Menetapkan tabel S
- Menetapkan nilai k
- Menetapkan jumlah cluster (m) yang ingin dibuat

$$m = \left\lfloor \frac{n}{k} \right\rfloor - 1 \quad (2.18)$$

Berikut adalah langkah kerja algoritma Greedy K-Means Clustering secara lengkap:

1. Melakukan inisialisasi variabel result dengan himpunan kosong dan variabel r dengan memilih data secara acak dari tabel S
2. Pada kondisi $|S| \geq k$, lakukan perulangan sebagai berikut:
 - (a) Memilih data baru pada variabel r berdasarkan perbedaan distance tertinggi dari nilai r sebelumnya. Perbedaan distance dapat dicari menggunakan rumus berikut:

$$\Delta(r_1, r_2) = \sum_{i=1}^m \delta_N(r_1[N_i], r_2[N_i]) + \sum_{j=1}^n \delta_C(r_1[C_j], r_2[C_j])$$

Berikut adalah rumus menghitung distance antar data numerik:

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

Berikut adalah rumus menghitung distance antar data kategorikal:

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

- (b) Membuang himpunan data variabel r pada variabel S
- (c) Mengisi data dari variabel r pada variabel c.
- (d) Pada kondisi $|c| \geq k$, lakukan perulangan sebagai berikut:
 - i. Memilih data baru terbaik untuk variabel r berdasarkan nilai Information Loss (IL) yang paling rendah. Information Loss (IL) dapat dicari menggunakan rumus berikut:

$$IL(e) = |e| \cdot D(e)$$

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

- ii. Membuang himpunan data dari variabel r pada variabel S
 - iii. Menambahkan himpunan data dari variabel r pada variabel c.
 - iv. Menambahkan himpunan data dari variabel c pada variabel result
3. Pada kondisi $|S| \neq 0$, artinya jika masih terdapat data yang belum dimasukkan pada sebuah cluster dari tabel S, lakukan perulangan sebagai berikut:
- (a) Memilih data secara acak dari tabel S untuk disimpan pada variabel r
 - (b) Membuang himpunan data dari variabel r pada variabel S
 - (c) Memilih cluster terbaik untuk variabel c berdasarkan nilai Information Loss (IL) yang paling rendah. Information Loss (IL) dapat dicari menggunakan rumus berikut:

$$IL(e) = |e| \cdot D(e)$$

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

- (d) Menambahkan himpunan data dari variabel r pada variabel c.
4. Algoritma ini mengembalikan himpunan data berdasarkan jenis cluster yang berbeda-beda melalui variabel result.

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  $r \in S$  such that  $IL(c \cup \{r\})$  is minimal
4:
5:  $n = |S|$ 
6:  $min = \infty$ 
7:  $best = null$ 
8: for  $i = 1 \dots n$  do
9:    $r = i\text{-th record in } S$ 
10:   $diff = IL(c \cup \{r\}) - IL(c)$ 
11:  if  $diff < min$  then
12:     $min = diff$ 
13:     $best = r$ 
14:  end if
15: end for
16: return  $best$ 

```

Algoritma 1 menerima input himpunan data dataset dan sebuah data dengan nilai distance tertinggi dari data terpilih acak. Algoritma ini menghitung selisih distance dari dua jenis data yang berbeda. Variabel "diff" pada algoritma ini adalah perbedaan distance, dicari dengan penjumlahan information loss pada sebuah kluster dengan information loss pada data ke-i, lalu hasil penjumlahan tersebut dikurangi dengan information loss dari kluster. Output algoritma ini adalah sebuah data dengan nilai terbaik, yaitu data ke-i dari dataset S dengan nilai distance paling kecil.

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  $c \in C$  such that  $IL(c \cup \{r\})$  is minimal
4:
5:  $n = |C|$ 
6:  $min = \infty$ 
7:  $best = null$ 
8: for  $i = 1 \dots n$  do
9:    $c = i\text{-th cluster in } C$ 
10:   $diff = IL(c \cup \{r\}) - IL(c)$ 
11:  if  $diff < min$  then
12:     $min = diff$ 
13:     $best = c$ 
14:  end if
15: end for
16: return  $best$ 

```

Algoritma 2 menerima input himpunan data kluster dan sebuah data dengan nilai distance tertinggi dari data terpilih acak. Algoritma ini menghitung selisih distance dari dua jenis data yang berbeda. Variabel "diff" pada algoritma ini adalah perbedaan distance, dicari dengan penjumlahan information loss pada sebuah kluster dengan information loss pada data ke-i, lalu hasil penjumlahan tersebut dikurangi dengan information loss dari kluster. Output algoritma ini adalah data dengan nilai kluster terbaik, yaitu data ke-i dari dataset S dengan nilai distance paling kecil.

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  $S \leq k$  then
6:   return  $S$ 
7: end if
8:
9:  $result = \phi$ 
10:  $r =$  a randomly picked record from  $S$ 
11: while  $|S| \geq k$  do
12:    $r =$  the furthest record from  $r$ 
13:    $S = S - \{r\}$ 
14:    $c = \{r\}$ 
15:   while  $|c| < k$  do
16:      $r = \text{find\_best\_record}(S, c)$ 
17:      $S = S - \{r\}$ 
18:      $c = c \cup \{r\}$ 
19:   end while
20:    $result = result \cup \{c\}$ 
21: end while
22: while  $S \neq 0$  do
23:    $r =$  a randomly picked record from  $S$ 
24:    $S = S - \{r\}$ 
25:    $c = \text{find\_best\_cluster}(result, r)$ 
26:    $c = c \cup \{r\}$ 
27: end while
28: return  $result$ 

```

Algoritma 3 menerima input himpunan data S dan nilai k . Algoritma ini mengeksekusi dua jenis fungsi yang berbeda yaitu fungsi `find_best_cluster` untuk mencari kluster dengan distance terkecil dan fungsi `find_best_record` untuk mencari data dengan distance terkecil. Output dari algoritma ini adalah himpunan data dari berbagai jenis kluster dengan nilai distance terkecil.

2.10 Distance, Information Loss, Cost Function

Konsep PPDM memberikan solusi untuk mengukur tingkat keamanan, fungsionalitas, dan utilitas data menggunakan beberapa jenis metrik. Beberapa metrik yang umum digunakan pada pengujian kualitas data yang telah dianonimisasi adalah distance, information loss, dan cost function. Secara umum, pengukuran metrik dilakukan dengan membandingkan hasil anonimisasi dengan dataset sesungguhnya.

2.10.1 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. Perhitungan distance dilakukan berdasarkan pengelompokan tipe data numerik atau kategorikal. Karena masalah k-anonimitas menggunakan atribut numerik dan kategorikal, maka membutuhkan cara khusus untuk menghitung distance dari kedua jenis data pada saat yang sama.

Distance Data Numerik

Distance data numerik direpresentasikan sebagai nilai rentang. Beberapa atribut pada distance numerik yaitu $|D|$ adalah jumlah data pada sebuah domain berdasarkan satu atribut numerik, v_1 , v_2 adalah nilai atribut numerik. Distance data numerik dihitung menggunakan rumus berikut:

$$\delta_n(v_1, v_2) = \frac{|v_1 - v_2|}{|D|} \quad (2.19)$$

Distance Data Kategorikal

Distance data kategorikal direpresentasikan sebagai pohon taksonomi. Beberapa atribut pada distance kategorikal yaitu $|D|$ adalah jumlah data pada domain kategorikal, TD adalah taxonomy tree untuk domain D, $H(\Lambda(v_i, v_j))$ adalah jarak dari satu subtree ke subtree lain, $H(T_D)$ adalah tinggi dari taxonomy tree. Distance data kategorikal dihitung menggunakan rumus berikut:

$$\delta_C(v_1, v_2) = \frac{H(\Lambda(v_i, v_j))}{H(T_D)} \quad (2.20)$$

Distance Record

Beberapa atribut pada distance record yaitu $r_1[N_i]$, $r_2[N_i]$ adalah nilai dari atribut numerik, $r_1[C_j]$, $r_2[C_j]$ adalah nilai dari atribut kategorikal, δ_N adalah distance data numerik, δ_C adalah distance data kategorikal. Distance record dihitung menggunakan rumus berikut:

$$\Delta(r_1, r_2) = \sum_{i=1}^m \delta_N(r_1[N_i], r_2[N_i]) + \sum_{j=1}^n \delta_C(r_1[C_j], r_2[C_j]) \quad (2.21)$$

2.10.2 Information Loss

Information Loss (IL) digunakan untuk mengevaluasi seberapa baik kinerja algoritma K-Anonymity terhadap utilitas sebuah data. Dalam menghitung Information Loss (IL), perlu mendefinisikan beberapa atribut seperti cluster $e = r_1, \dots, r_k$ untuk quasi-identifier yang terdiri dari atribut numerik N_1, \dots, N_m dan atribut kategorikal C_1, \dots, C_n , T_{C_i} adalah taxonomy tree untuk domain kategorikal C_i , MIN_{N_i} dan MAX_{N_i} adalah nilai minimum dan maksimum pada cluster e untuk atribut N_i , \cup_{C_i} adalah sekumpulan nilai pada cluster e berdasarkan atribut C_i .

Information loss dihitung dengan rumus sebagai berikut:

$$IL(e) = |e| \cdot D(e) \quad (2.22)$$

$$D(e) = \sum_{i=1}^m \frac{(MAX_{N_i} - MIN_{N_i})}{|N_i|} + \sum_{j=1}^n \frac{H(\Lambda(\cup_{C_j}))}{H(T_{C_j})} \quad (2.23)$$

Total Information Loss dihitung dengan rumus sebagai berikut:

$$Total - IL(AT) = \sum_{e \in \epsilon} IL(e) \quad (2.24)$$

2.10.3 Cost Function

Cost function menyangkut efisiensi dan skalabilitas algoritma. Efisiensi algoritma dapat dinilai berdasarkan faktor waktu dan ruang. Waktu dapat diukur berdasarkan biaya komputasi algoritma. Ruang dapat diukur berdasarkan jumlah pemakaian memori.

2.11 Sistem Terdistribusi

Sistem terdistribusi adalah kumpulan komputer berjalan secara independen, yang saling terhubung dan saling bekerja sama untuk mencapai satu tujuan yang sama. 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.

2.11.1 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.

2.11.2 Tantangan Sistem Terdistribusi

Tantangan penggunaan sistem terdistribusi adalah pemrosesan data skala besar dengan handal. 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, sehingga sistem terdistribusi harus dapat memutuskan pekerjaan mana yang harus dikerjakan lebih dulu.
- **Latensi**
Dengan pertukaran data antara perangkat keras dan perangkat lunak menggunakan jalur komunikasi jaringan, sehingga nilai latensi menjadi sangat tinggi.
- **Observasi**
Ketika sistem terdistribusi menjadi kompleks, kemampuan pengamatan untuk memahami kegagalan pada sistem terdistribusi merupakan tantangan besar. komputer.

2.12 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. Data dapat dikategorikan sebagai data besar berdasarkan berbagai faktor. Konsep utama yang umum dalam semua faktor adalah jumlah data.

Berikut adalah karakteristik 5V untuk Big data:

- **Volume**
Volume mengacu pada jumlah data yang sangat besar. Data tumbuh begitu besar sehingga sistem komputasi tradisional tidak lagi dapat menanganinya seperti yang kita inginkan.
- **Velocity**
Velocity mengacu pada kecepatan di mana data dihasilkan. Setiap hari, sejumlah besar data dihasilkan, disimpan, dan dianalisis. Data dihasilkan dengan kecepatan kilat di seluruh dunia. Teknologi Big Data memungkinkan untuk mengeksplorasi data, saat data itu dihasilkan.
- **Variety**
Variety mengacu pada berbagai jenis data. Data terutama dikategorikan ke dalam data terstruktur dan tidak terstruktur. Faktanya, lebih dari 75 persen data dunia ada dalam bentuk yang tidak terstruktur.
- **Veracity**
Veracity mengacu pada kualitas data. Ketika menyimpan beberapa data yang besar, apabila tidak ada gunanya di masa depan, maka membuang-buang sumber daya untuk menyimpan data tersebut. Jadi, kita harus memeriksa kepercayaan data sebelum menyimpannya.
- **Value**
Value adalah bagian terpenting dari big data. Organisasi menggunakan data besar untuk menemukan nilai informasi baru. Menyimpan sejumlah besar data sampai pada ekstraksi informasi yang bermakna dari sekumpulan data tersebut.

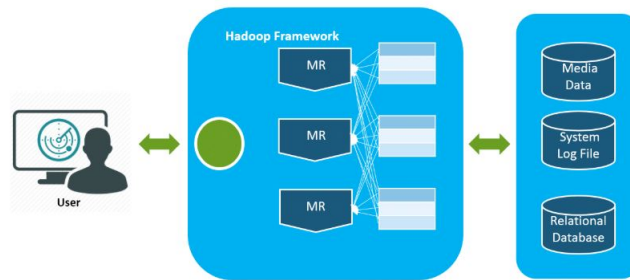
Analisis big data adalah proses menggunakan algoritma analisis yang berjalan pada platform pendukung yang kuat untuk mengungkap potensi yang disembunyikan dalam data besar, seperti pola tersembunyi atau korelasi yang tidak diketahui. Analisis big data dapat dikategorikan ke dalam dua jenis pemrosesan.

Berikut adalah jenis analisis pada big data:

- **Batch Processing**
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.
- **Streaming Processing**
Streaming processing mengasumsikan 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. Paradigma streaming processing biasanya digunakan untuk analisis aplikasi online dalam tingkat satuan detik maupun milidetik.

2.13 Hadoop

Apache Hadoop bertujuan untuk memecahkan permasalahan pada data besar. Google menemukan metodologi baru untuk memproses data yang dikenal sebagai MapReduce. Doug Cutting dan anggota timnya mengembangkan proyek open-source dikenal sebagai Hadoop untuk menangani jumlah data yang sangat besar. Kekuatan terbesar Hadoop adalah skalabilitas. Artinya Hadoop bekerja pada satu node ke ribuan node. Hadoop menjalankan MapReduce di mana data akan diproses secara paralel. Hadoop adalah framework yang didasarkan pada bahasa pemrograman Java. Hadoop bekerja dari satu server ke ribuan komputer dengan menawarkan perhitungan dan penyimpanan lokal untuk setiap komputer.

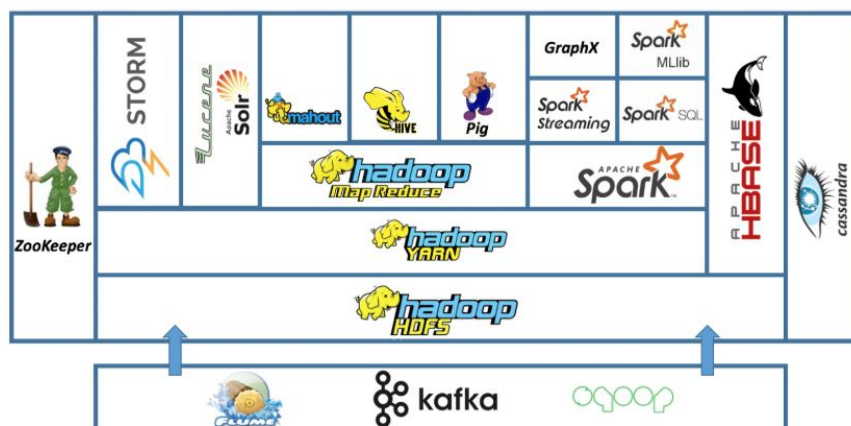


Gambar 2.14: Gambaran Umum Cara Kerja Hadoop

Hadoop memiliki karakteristik sebagai berikut:

- Hadoop menyediakan penyimpanan HDFS dan menggunakan pemodelan MapReduce.
- Hadoop melakukan replikasi data sejenis pada beberapa komponen berbeda.
- Hadoop fleksibel terhadap pengaturan jumlah komponen yang bekerja.
- Hadoop dioptimalkan untuk bekerja pada lingkungan big data.
- Hadoop menulis data sekali dan membaca data berulang kali.

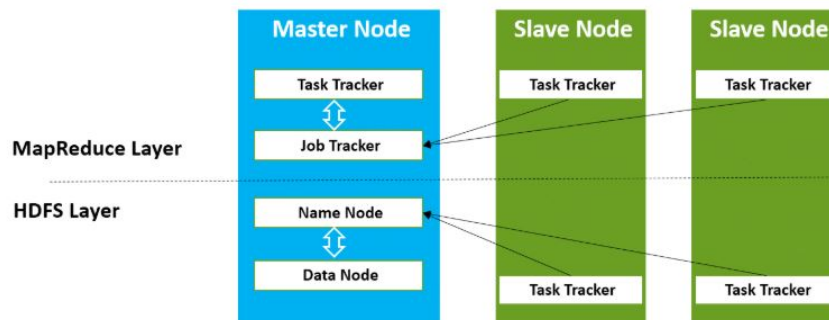
2.13.1 Ekosistem Hadoop



Gambar 2.15: Ekosistem Hadoop

Gambar 2.15 menunjukkan bahwa Hadoop dapat bekerja secara bersamaan dengan teknologi big data lainnya seperti Spark, HBase, Cassandra, STORM, dan lain-lain untuk memenuhi berbagai macam kebutuhan dalam pengolahan dan analisis pada big data.

2.13.2 Arsitektur Hadoop



Gambar 2.16: Arsitektur Hadoop

Gambar 2.16 menunjukkan arsitektur Hadoop yang tersusun atas 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.

2.13.3 HDFS

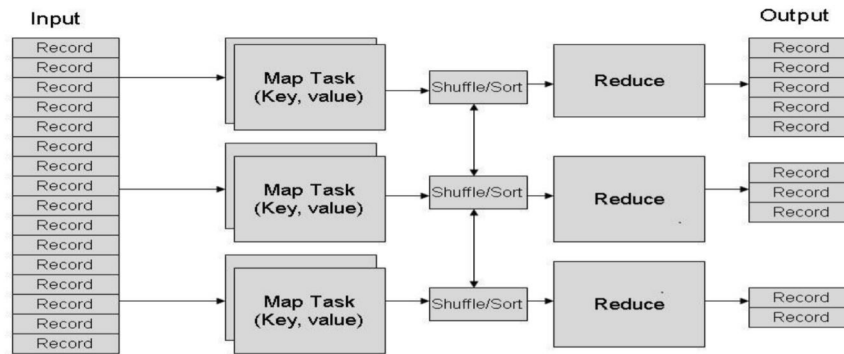
HDFS adalah sistem file terdistribusi pada Hadoop dengan menyediakan penyimpanan data yang handal, mendukung partisi, dan toleran terhadap kesalahan pada hardware. HDFS bekerja erat dengan MapReduce dengan mendistribusikan penyimpanan dan perhitungan di seluruh cluster dengan menggabungkan sumber daya penyimpanan yang dapat dipartisi tergantung kebutuhan.

Lapisan penyimpanan HDFS terdiri dari dua elemen:

- **NameNode**
NameNode adalah sebuah komputer yang bertindak sebagai master node, sedangkan. 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.
- **DataNode**
DataNode 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.

2.13.4 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.



Gambar 2.17: Tahapan pada MapReduce

Pada Gambar 2.17, tahapan MapReduce dapat dibagi menjadi dua fungsi utama:

- **Mapper**
Tugas fungsi Mapper adalah memetakan blok data kedalam pasangan $\langle \text{key}, \text{value} \rangle$. Key,value pada Mapper tidak harus memiliki tipe data yang sama satu sama lain. Pasangan $\langle \text{key}, \text{value} \rangle$ 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 kunci yang sama.
 2. **Sort**
Pasangan $\langle \text{key}, \text{value} \rangle$ pada satu node secara otomatis diurutkan oleh Hadoop sebelum diberikan kepada Reducer. Penyortiran dilakukan berdasarkan keterurutan nilai key.
 3. **Reducer**
Data output mapper yang diacak dan diurutkan disediakan untuk Reducer. Tahap ini membuat pasangan $\langle \text{key}, (\text{list of value}) \rangle$ baru berdasarkan pengelompokan key.

Berikut adalah komponen yang mengatur pekerjaan pada MapReduce:

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

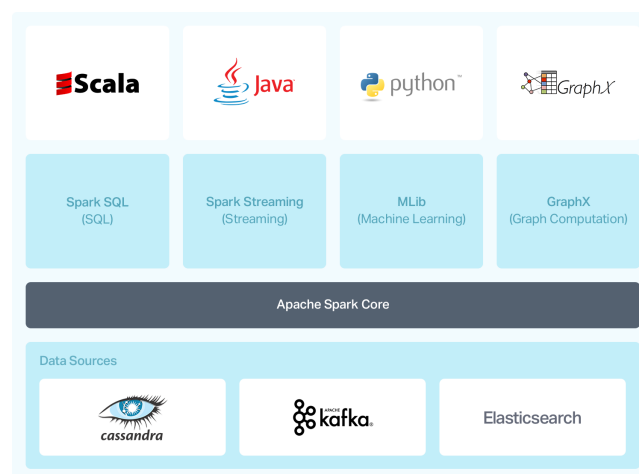
2.14 Spark

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 dari Hadoop MapReduce untuk mengatasi keterbatasan pemrosesan input output yang tidak efisien pada disk, dengan menggunakan memori. Fitur utama Spark adalah melakukan komputasi di dalam memori sehingga waktu komputasi menjadi lebih singkat dibandingkan waktu komputasi di dalam disk.

Berikut adalah karakteristik dari Spark:

- **Kecepatan**
Spark adalah alat komputasi kluster tujuan umum. Ini menjalankan aplikasi hingga 100 kali lebih cepat dalam memori dan 10 kali lebih cepat pada disk daripada Hadoop. Spark mengurangi jumlah operasi baca/tulis pada disk dan menyimpan data dalam memori.
- **Mudah untuk diatur**
Spark dapat melakukan pemrosesan batch, analisis data secara interaktif, machine learning, dan streaming data. Semuanya pemrosesan tersebut dikerjakan pada satu komputer yang sama. Fungsi ini menjadikan Apache Spark sebagai mesin analisis data yang lengkap.
- **Analisis secara real-time**
Spark dapat dengan mudah memproses data waktu-nyata, misalnya streaming data secara real-time untuk ribuan peristiwa/detik. Contoh dari sumber streaming data adalah Twitter, Facebook, Instagram. Streaming data dapat diproses secara efisien oleh Spark.

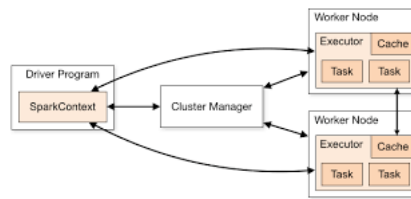
2.14.1 Ekosistem Spark



Gambar 2.18: Ekosistem Spark

Gambar 2.18 menunjukkan bahwa Spark bekerja sama dengan teknologi big data lain untuk memenuhi berbagai macam kebutuhan dalam pengolahan big data. Masing-masing warna pada Gambar 2.18 mewakili jenis teknologi yang dipakai pada Spark. Spark SQL, Spark Streaming, Spark MLib 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.

2.14.2 Arsitektur Spark

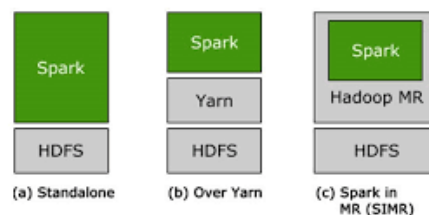


Gambar 2.19: Arsitektur Spark

Berikut adalah cara kerja pada arsitektur Spark:

1. Program Driver memanggil program utama aplikasi dan membuat SparkContext dan Spark Driver. SparkContext terdiri dari semua fungsi dasar. Spark Driver berisi DAG Scheduler, Task Scheduler, Backend Scheduler, dan Block Manager.
2. Spark Driver dan SparkContext secara kolektif mengawasi pelaksanaan pekerjaan di dalam cluster. Spark Driver bekerja sama dengan Cluster Manager untuk pembagian pekerjaan ke setiap Worker Node. Setiap kali RDD dibuat dalam SparkContext, RDD akan didistribusikan di setiap Worker Node.
3. Worker Node menjalankan tugas yang diberikan oleh Cluster Manager dan mengembalikannya ke Konteks Spark. Worker Node bertanggung jawab atas pelaksanaan tugas-tugas ini. Masa pakai pelaksana sama dengan masa hidup Aplikasi Spark. Jika ingin meningkatkan kinerja sistem, maka dapat meningkatkan jumlah pekerja sehingga pekerjaan dapat dibagi menjadi bagian kecil.

2.14.3 Jenis Instalasi pada Spark



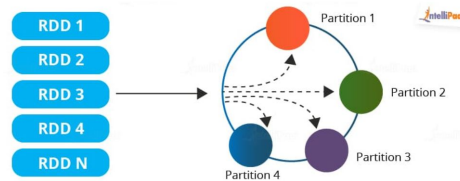
Gambar 2.20: Arsitektur Spark

Berikut adalah penjelasan jenis instalasi Spark pada Gambar 2.20 sebagai berikut:

- Standalone
Spark berdiri diatas HDFS Hadoop. Spark memungkinkan untuk mengakses data pada HDFS Hadoop untuk membaca input dan menulis output.
- Hadoop Yarn
Spark dapat berjalan pada Hadoop Yarn tanpa memerlukan instalasi atau meminta hak akses root apapun. Hadoop Yarn membantu integrasi Spark pada ekosistem Hadoop.
- Spark In MapReduce (SIMR)
SIMR digunakan untuk menjalankan pekerjaan Spark secara independen. Jenis instalasi ini sudah tidak lagi berlaku untuk Spark versi 2.0

2.14.4 Resilient Distibuted Datasets (RDD)

RDD adalah unit data logis utama di Spark. Mereka adalah kumpulan objek terdistribusi, yang disimpan dalam memori atau pada disk dari berbagai mesin cluster. RDD dapat dibagi menjadi beberapa partisi data sehingga partisi ini dapat disimpan dan diproses pada komputer yang berbeda. RDD adalah struktur data yang immutable (hanya baca). Anda tidak dapat mengubah RDD asli, tetapi Anda dapat membuat RDD baru dengan melakukan operasi butiran kasar, seperti transformasi, pada RDD yang ada. RDD dalam Spark dapat di simpan dalam cache dan digunakan lagi untuk transformasi di masa depan. RDD dievaluasi secara lazy, artinya Spark menunda evaluasi sampai benar-benar diperlukan. Ini menghemat banyak waktu dan meningkatkan efisiensi.

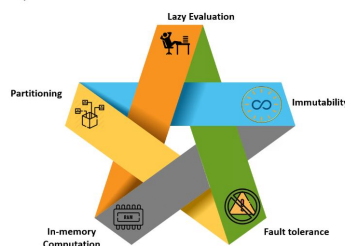


Gambar 2.21: Cara Kerja RDD

Fitur pada RDD

Berikut adalah fitur RDD pada Spark:

- **Resilience:** RDD melacak informasi garis keturunan data untuk memulihkan data yang hilang, secara otomatis jika gagal. Ini juga disebut toleransi kesalahan.
- **Distributed:** Data yang ada dalam RDD berada di beberapa node. Ini didistribusikan di berbagai node cluster.
- **Lazy evaluation:** Data tidak dimuat dalam RDD bahkan jika Anda mendefinisikannya. Transformasi sebenarnya dihitung ketika Anda memanggil suatu tindakan, seperti menghitung atau mengumpulkan, atau menyimpan output ke sistem file
- **Immutability:** Data yang disimpan dalam RDD berada dalam mode read-only. RDD tidak dapat mengubah data didalamnya. Namun, RDD baru dapat dibuat dengan memanggil operasi transformasi pada RDD yang ada.
- **In-memory computation:** RDD menyimpan data langsung apa pun yang dihasilkan dalam memori (RAM) selain pada disk sehingga memberikan akses yang lebih cepat.
- **Partitioning:** Partisi dapat dilakukan pada RDD yang ada untuk membuat bagian logis yang bisa berubah. Anda dapat mencapai ini dengan menerapkan transformasi pada partisi yang tersedia.



Gambar 2.22: Fitur pada RDD

Operasi pada RDD

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

- Transformation

Transformation adalah operasi pada RDD untuk mengembalikan output berupa RDD baru. Transformasi pada RDD dilakukan secara lazy, sehingga transformasi hanya akan dikerjakan apabila dipanggil pada sebuah action. Transformasi pada RDD akan dijelaskan pada tabel dibawah ini.

Fungsi	Deskripsi
map()	Mengembalikan RDD baru dengan menerapkan fungsi pada setiap elemen data
filter()	Mengembalikan RDD baru yang dibentuk dengan memilih elemen-elemen sumber di mana fungsi mengembalikan true
reduceByKey()	Menggabungkan nilai-nilai kunci menggunakan fungsi
groupByKey()	Mengonversi pasangan (kunci, nilai) menjadi pasangan (kunci, <iterable value>)
union()	Mengembalikan RDD baru yang berisi semua elemen dan argumen dari RDD sumber
intersection()	Mengembalikan RDD baru yang berisi persimpangan elemen dalam dataset

- 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. Action pada RDD akan dijelaskan pada tabel dibawah ini.

Fungsi	Deskripsi
count()	Mendapat jumlah elemen data dalam RDD
collect()	Mendapat semua elemen data dalam RDD sebagai array
reduce()	Agregat elemen data ke dalam RDD dengan mengambil dua argumen dan mengembalikan satu
take(n)	Mengambil n elemen pertama dari RDD
foreach(operation)	Menjalankan operasi untuk setiap elemen data dalam RDD
first()	Mengambil elemen data pertama dari RDD

2.14.5 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 komponen 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 terdiri dari mesin eksekusi umum untuk platform Spark

yang dibangun sesuai kebutuhan. Spark Core memberikan komputasi pada memori dan set data disimpan dalam sistem penyimpanan eksternal. Spark Core juga menyediakan operasi action dan transformation pada RDD.

- **Spark SQL**
Spark SQL adalah komponen di atas Spark Core yang memperkenalkan serangkaian abstraksi data baru yang disebut SchemaRDD. SchemaRDD menyediakan dukungan untuk data terstruktur dan semi-terstruktur. Spark SQL memungkinkan pemrosesan kueri SQL pada lingkungan big data. Spark SQL menyediakan fungsi untuk menghitung nilai statistik dasar seperti mean, median, modus, nilai maksimum dan nilai minimum.
- **Spark Streaming**
Spark Streaming adalah salah satu komponen Apache Spark, yang memungkinkan Spark dapat memproses data streaming secara real-time. Spark Streaming menyediakan API untuk memanipulasi aliran data yang cocok dengan RDD. Hal ini memungkinkan analisis data untuk beralih melalui sumber aplikasi yang memberikan data secara real time. Mirip dengan Spark Core, Spark Streaming berusaha untuk membuat sistem toleran dan scalable
- **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. Spark MLlib juga menyediakan fungsi untuk menghitung nilai statistik dasar seperti nilai maksimum, minimum, mean, dan variansi.

2.15 Spark MLlib

Spark MLlib diciptakan untuk skalabilitas, dan integrasi yang mudah dengan komponen pada Spark. Dengan adanya kemudahan tersebut diharapkan penelitian dapat difokuskan pada masalah pada pemodelan data daripada menyelesaikan masalah kompleksitas data. Spark MLlib adalah library pembelajaran mesin berdasarkan komputasi secara paralel. MLlib terdiri dari algoritma pembelajaran umum seperti klasifikasi, pengelompokan/clustering. Spark MLlib dapat terintegrasi dengan komponen Spark lainnya seperti Spark SQL dan Spark Streaming. Spark MLlib dapat digunakan pada bahasa pemrograman Java, Scala, dan Python. Secara garis besar, MLlib melakukan data preprocessing, pelatihan model, dan membuat prediksi.

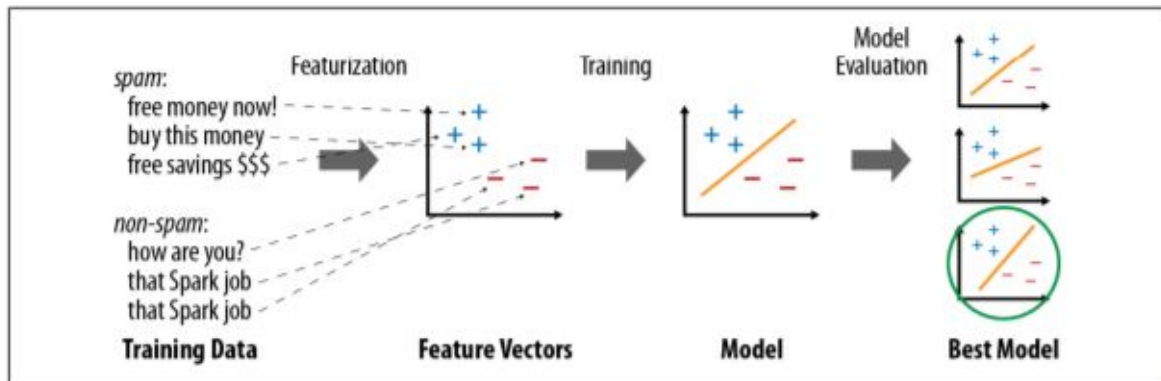
Berikut adalah garis besar pekerjaan yang dilakukan oleh Spark MLlib:

1. Membuat RDD baru untuk merepresentasikan jenis fitur dalam dataset.
2. Melakukan ekstraksi fitur pada sebuah vektor dan konversi menjadi data numerik.
3. Menjalankan algoritma machine learning untuk menghasilkan pemodelan baru.
4. Melakukan evaluasi model dengan memanggil fungsi evaluasi milik Spark MLlib.

Berikut adalah contoh pemodelan Spark MLlib yang umum digunakan untuk prediksi:

- Naive Bayes (klasifikasi)
- K-Means (pengelompokan/clustering)
- Statistika dasar (mean, variansi, nilai maksimum dan minimum)
- Grafik (visualisasi data)

2.15.1 Machine Learning pada Spark MLlib



Gambar 2.23: Tahapan Pembelajaran Machine Learning

Algoritma pembelajaran mesin berupaya membuat prediksi atau keputusan berdasarkan data pelatihan. Ada beberapa jenis masalah pembelajaran termasuk klasifikasi, regresi, atau pengelompokan, yang memiliki tujuan yang berbeda. Semua algoritma pembelajaran membutuhkan pendefinisian sekumpulan fitur untuk setiap item, yang akan dimasukkan ke dalam fungsi pembelajaran. Sebagian besar algoritma didefinisikan hanya untuk fitur numerik (khususnya, vektor angka yang mewakili nilai untuk setiap fitur).

Berikut adalah tahapan pembelajaran machine learning pada Spark MLlib sebagai berikut:

1. Featurization

Setelah data telah dibersihkan dan dipilih untuk pelatihan, perlu diubah menjadi representasi numerik dalam bentuk vektor sebagai input untuk model. Proses ini disebut vektorisasi. Proses pemilihan fitur yang tepat berpengaruh terhadap hasil pelatihan.

2. Training

Pelatihan model merupakan salah satu tahapan yang paling memakan waktu dan padat karya dalam setiap alur kerja pembelajaran mesin. Terlebih lagi, perangkat keras dan infrastruktur yang digunakan untuk melatih model sangat bergantung pada jumlah parameter dalam model, ukuran dataset, metode pengoptimalan yang digunakan, dan pertimbangan lainnya.

3. Model Evaluation

Setiap model harus menjalani evaluasi kualitatif dan kuantitatif. Banyak data pelatihan yang memiliki hasil evaluasi yang sesuai terhadap kinerja model. Model tersebut harus diterapkan pada data baru. Seringkali, pemeriksaan kualitatif tentang kinerja model, diperoleh dengan referensi silang prediksi model dengan apa yang diharapkan secara intuitif, dapat berfungsi sebagai panduan apakah model bekerja sesuai harapan.

2.15.2 Tipe Data pada Spark MLlib

Spark MLlib terdiri dari beberapa jenis tipe data berdasarkan library bahasa pemrograman yang digunakan. Pada Java/Scala, library MLlib menggunakan istilah package. Pada Python, library MLlib menggunakan istilah pyspark.mllib. Tipe data Spark MLlib dibuat dengan cara menyimpan objek data RDD berdasarkan operasi transformasi RDD pada dataset.

Berikut adalah beberapa jenis tipe data pada Spark MLlib:

- Vektor

Vektor terdiri dari dua jenis yaitu vektor dense dan vektor sparse. Perlu digaris bawahi bahwa elemen vektor dense dan vektor sparse disimpan dalam struktur data array bertipe double. Gambar ?? adalah contoh implementasi vektor dense dan vektor sparse pada Spark untuk bahasa pemrograman Scala. Kelas vektor berada pada package `mllib.linalg.Vectors`.

Berikut adalah penjelasan singkat dari vektor dense dan vektor sparse:

- Vektor dense

Vektor dense adalah vektor yang menyimpan setiap nilai fitur dataset. Jumlah elemen pada vektor dense akan memiliki jumlah yang sama dengan jumlah fitur pada dataset.

- Vektor sparse

Vektor sparse biasanya lebih disukai karena memiliki kelebihan dalam hal penggunaan memori dan kecepatan. Vektor sparse adalah vektor yang menyimpan setiap nilai fitur yang bukan nol pada dataset, sehingga jumlah elemen yang disimpan pada vektor sparse lebih sedikit dibandingkan dengan jumlah elemen yang disimpan pada vektor dense.

- LabeledPoint

LabeledPoint digunakan pada algoritma supervised learning yaitu klasifikasi dan regresi. LabeledPoint terdiri dari vektor fitur dan label yang direpresentasikan dengan tipe data Double. LabeledPoint terletak pada package `"mllib.regress"`.

- Various Model classes

Various Model classes adalah tipe data yang dihasilkan dari penerapan algoritma machine learning. Tipe data ini memiliki fungsi `predict()` untuk menerapkan pemodelan yang ada ke dalam data baru.

2.15.3 Jenis Pemodelan pada Spark MLlib

Naive Bayes

Naive Bayes adalah algoritma klasifikasi multiclass sederhana dengan asumsi independensi antara setiap pasangan fitur. Naive Bayes dapat dilatih dengan sangat efisien. Naive Bayes menghitung distribusi probabilitas bersyarat dari setiap fitur yang diberikan label, kemudian menerapkan teorema Bayes untuk menghitung distribusi probabilitas bersyarat label yang diberikan pengamatan dan menggunakannya untuk prediksi. Naive Bayes membutuhkan RDD dari LabeledPoint. Pemodelan Naive Bayes dapat digunakan untuk evaluasi dan prediksi. Gambar ?? adalah contoh implementasi dari pemodelan naive bayes.

K-Means

K-Means adalah salah satu algoritma pengelompokan yang paling umum digunakan untuk mengelompokkan titik-titik data menjadi sejumlah kelompok yang telah ditentukan. Gambar ?? adalah contoh implementasi dari pemodelan K-Means. K-Means memiliki parameter berikut:

- *k* adalah jumlah cluster yang diinginkan.
- *maxIterations* adalah jumlah iterasi maksimum yang harus dijalankan.
- *initializationMode* menentukan inisialisasi centroid secara acak.
- *initializationSteps* menentukan jumlah langkah dalam algoritma k-means.
- *initialModel* adalah menentukan nilai centroid saat inisialisasi.

Statistika dasar

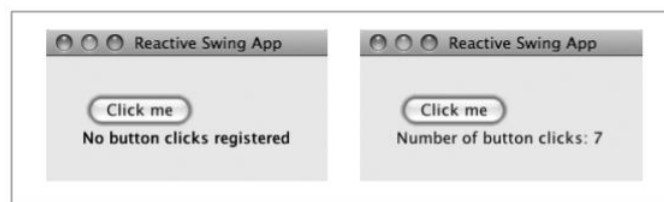
Statistik dasar adalah bagian penting dari analisis data, baik dalam eksplorasi ad hoc maupun memahami data untuk pembelajaran mesin. MLlib menawarkan beberapa statistik yang banyak digunakan fungsi yang bekerja langsung pada RDD, melalui metode di kelas `mllib.stat.Statistics`. Fungsi `colStats()` akan mengembalikan nilai maksimum, minimum, rata-rata, variansi. Gambar ?? adalah contoh implementasi statistika dasar pada Spark MLlib.

2.16 Scala

Scala adalah bahasa pemrograman berbasis open source, dibuat oleh Profesor Martin Odersky. Scala adalah bahasa pemrograman multi-paradigma dan mendukung paradigma fungsional serta berorientasi objek. 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. 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 Spark.

2.17 Scala Swing

Scala Swing adalah program berbasis Graphical User Interface (GUI) sehingga memiliki perbedaan dengan program Spark yang dieksekusi dengan terminal. Scala Swing bertujuan untuk memberi tampilan program sehingga hasil program diharapkan menjadi lebih interaktif. Scala menyediakan akses langsung terhadap kelas GUI pada Java menggunakan library Scala Swing. Dengan menggunakan Scala, penggunaan Scala Swing dapat memenuhi kebutuhan perancangan User Interface melalui berbagai macam komponen GUI pada umumnya. Gambar 2.24 adalah contoh implementasi GUI sederhana pada Scala Swing.



Gambar 2.24: GUI Sederhana pada Scala Swing

2.17.1 Panel dan Layout

Panel adalah tempat untuk menampilkan semua komponen GUI dengan beberapa aturan tata letak yang harus dipenuhi. Salah satu bagian tersulit pada perancangan aplikasi berbasis GUI adalah mengatur penempatan layout dengan benar. Gambar ?? diberikan sebagai contoh penerapan konsep dasar untuk menyusun aplikasi GUI. Layout dibangun atas beberapa komponen GUI seperti Frame, Panel, Label atau Button. Komponen ini memiliki nilai properti (warna, ukuran, posisi) yang dapat diatur secara manual.

2.17.2 Handling Event

Handling event adalah pekerjaan yang dilakukan masing-masing komponen. Komponen akan menerima aksi langsung dari pengguna aplikasi. Mekanisme ini dikenal sebagai handling event, yang dieksekusi ketika suatu peristiwa terjadi. Handling event memiliki listener. Listener adalah sebuah komponen memberi tahu sebuah aksi kepada komponen tertentu. Listener harus dibuat untuk masing-masing objek handling event. Gambar ?? adalah contoh implementasi handling event pada Scala Swing.

2.18 Jenis Data Input

Spark menerima berbagai jenis data input dan memfasilitasi penulisan berbagai jenis data output. Beberapa jenis data yang akan diolah memiliki format penulisan masing-masing, sehingga memiliki cara yang berbeda untuk mengambil dan menulis sekumpulan data dalam format tertentu.

2.18.1 CSV

Comma Separated Value (CSV) berisi sejumlah jumlah atribut data untuk setiap baris dan dipisahkan oleh koma. CSV tidak memiliki format yang konsisten. CSV tidak dapat mengakses nilai atribut data secara langsung. Untuk mendapatkan nilai atribut secara spesifik, perlu dilakukan pemisahan koma untuk setiap baris data. Spark mampu melakukan proses manipulasi dan transformasi CSV tanpa menggunakan library tambahan.

2.18.2 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. Spark mampu melakukan proses manipulasi dan transformasi data XML dengan menggunakan library tambahan.

BAB 3

ANALISIS MASALAH, EKSPLORASI, DAN STUDI KASUS

3.1 Analisis Masalah

Berkembangnya penggunaan teknologi informasi menyebabkan data bertumbuh dengan sangat pesat. Istilah data yang memiliki ukuran yang besar dikenal sebagai big data. Data mining adalah cara untuk mengekstraksi sebuah informasi dari sekumpulan data untuk mendukung pengambilan keputusan atau pernyataan tertentu. Hasil data mining yang mengandung data konsumen apabila disebarkan kepada pihak lain untuk kebutuhan tertentu tanpa dilakukan perlindungan privasi terlebih dahulu maka dapat melanggar hak privasi seseorang. Apabila informasi pribadi seseorang dapat diketahui oleh orang lain, maka mengakibatkan munculnya tindak kejahatan yang mengatasnamakan privasi orang bersangkutan. Oleh karena itu, perlu adanya sebuah cara untuk melindungi privasi seseorang sebelum dilakukan distribusi data.

Solusi yang tepat untuk menjamin perlindungan data sebelum dilakukan distribusi data adalah anonimisasi. Anonimisasi bertujuan untuk menyamarkan sebagian nilai atribut data yang unik terhadap atribut data lain, khususnya untuk atribut yang termasuk dalam kategori atribut privasi menurut PII. Privacy-preserving data mining adalah sebuah cara untuk melindungi data sebelum dilakukan data mining agar privasi dari hasil data mining dapat terlindungi. K-Anonymity adalah salah satu metode agar privacy-preserving data mining dapat dicapai dengan menyamarkan beberapa nilai atribut data. Tujuan utama dari penelitian ini adalah mempelajari, menganalisis, melakukan eksperimen, membuat perangkat lunak terkait anonimisasi pada lingkungan big data, dan menguji hasilnya agar privasi data dapat terjaga. Berikut beberapa kajian yang akan dianalisis terkait teknik anonimisasi pada lingkungan big data.

3.1.1 Dataset

Dataset yang dipakai pada penelitian ini adalah dataset Adults. Dataset ini diperoleh dari website Kaggle. Dataset ini telah disimpan dalam format CSV. Format CSV memisahkan nilai atribut data melalui simbol koma. Dataset Adults dipilih, karena pernah digunakan sebelumnya untuk eksperimen algoritma k-anonymity. Dataset ini berisi sampel sensus penduduk di Amerika Serikat pada tahun 1990. Penelitian ini melibatkan 10 juta baris data dengan ukuran data sebesar 1.2 GB.

```
39, State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K
50, Self-emp-not-inc, 83311, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 13, United-States, <=50K
38, Private, 215646, HS-grad, 9, Divorced, Handlers-cleaners, Not-in-family, White, Male, 0, 0, 40, United-States, <=50K
53, Private, 234721, 11th, 7, Married-civ-spouse, Handlers-cleaners, Husband, Black, Male, 0, 0, 40, United-States, <=50K
28, Private, 338409, Bachelors, 13, Married-civ-spouse, Prof-specialty, Wife, Black, Female, 0, 0, 40, United-States, <=50K
37, Private, 284582, Masters, 14, Married-civ-spouse, Exec-managerial, Wife, White, Female, 0, 0, 40, United-States, <=50K
49, Private, 160187, 9th, 5, Married-spouse-absent, Other-service, Not-in-family, Black, Female, 0, 0, 16, Jamaica, <=50K
52, Self-emp-not-inc, 209642, HS-grad, 9, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 45, United-States, >50K
31, Private, 45781, Masters, 14, Never-married, Prof-specialty, Not-in-family, White, Female, 14084, 0, 50, United-States, >50K
42, Private, 159449, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 5178, 0, 40, United-States, >50K
37, Private, 280464, Some-college, 10, Married-civ-spouse, Exec-managerial, Husband, Black, Male, 0, 0, 80, United-States, >50K
30, State-gov, 141297, Bachelors, 13, Married-civ-spouse, Prof-specialty, Husband, Asian-Pac-Islander, Male, 0, 0, 40, India, >50K
23, Private, 122272, Bachelors, 13, Never-married, Adm-clerical, Own-child, White, Female, 0, 0, 30, United-States, <=50K
32, Private, 205019, Assoc-acdm, 12, Never-married, Sales, Not-in-family, Black, Male, 0, 0, 50, United-States, <=50K
40, Private, 121772, Assoc-voc, 11, Married-civ-spouse, Craft-repair, Husband, Asian-Pac-Islander, Male, 0, 0, 40, ?, >50K
34, Private, 245487, 7th-8th, 4, Married-civ-spouse, Transport-moving, Husband, Amer-Indian-Eskimo, Male, 0, 0, 45, Mexico, <=50K
25, Self-emp-not-inc, 176756, HS-grad, 9, Never-married, Farming-fishing, Own-child, White, Male, 0, 0, 35, United-States, <=50K
32, Private, 186824, HS-grad, 9, Never-married, Machine-op-inspct, Unmarried, White, Male, 0, 0, 40, United-States, <=50K
```

Gambar 3.1: Dataset Adults

Berikut adalah kemungkinan nilai untuk masing-masing jenis atribut dalam dataset:

- Age: numerik
- Workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- Education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- Years of education: numerik
- Marital status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, MarriedAF-spouse
- Occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspect, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, ArmedForces.
- Relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
- Race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
- Sex: Male, Female
- Capital gain: numerik
- Capital loss: numerik
- Hours per week: numerik
- Native country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVltc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinidad and Tobago, Peru, Hong, HollandNetherlands
- Income: $\leq 50K$, $> 50K$

3.1.2 Personally Identifiable Information

Pada bagian 2.1, telah dijelaskan mengenai konsep Personally Identifiable Information (PII). PII digunakan untuk mengelompokkan nilai atribut berdasarkan kategori atribut yang digunakan pada proses anonimisasi data. Berdasarkan bagian 2.6, atribut pada proses anonimisasi dapat dikategorikan sebagai identifier, quasi-identifier, dan sensitive attribute.

Berdasarkan jenis atribut pada dataset Adult, diperoleh kategori sebagai berikut:

- Identifier (ID): atribut yang nilainya secara langsung dapat mengidentifikasi seseorang
Contoh: name (sudah dihilangkan pada dataset)
- Quasi-identifier (QID): atribut yang nilainya dapat digabungkan dengan atribut lain untuk mengidentifikasi seseorang
Contoh: age, zip, education, years of education, occupation, race, sex, native country, income
- Sensitive Attribute (SA): atribut yang nilainya bersifat privasi atau rahasia
Contoh: workclass, marital status, relationship

3.1.3 Distance, Information Loss, Cost Function

Distance dan Information Loss digunakan oleh algoritma Greedy K-Member Clustering untuk mencari kelompok data terbaik sehingga menghasilkan pengelompokan data yang tepat. Cost Function akan dianalisis untuk mencari tahu seberapa baik kinerja algoritma Greedy K-Member Clustering. Cost Function penting untuk dianalisis mengingat data input yang akan diproses berukuran sangat besar.

Distance

Distance bertujuan untuk menentukan hasil pengelompokan data pada algoritma Greedy K-Member Clustering. Pemilihan distance yang baik dapat mencapai hasil klasifikasi yang lebih optimal.

Akan diambil 2 sampel data dari dataset Adults sebagai berikut:

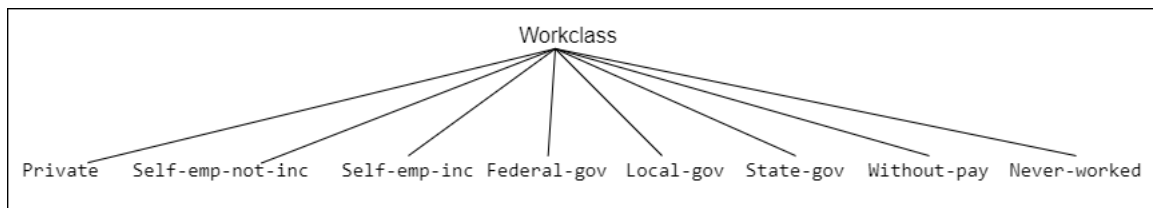
1. 39, State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K
2. 50, Self-emp-not-inc, 83311, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 13, United-States, <=50K

Distance atribut numerik dapat dihitung sebagai berikut berdasarkan umur data pertama (v_1)= 39, umur data kedua (v_2)= 50, dan jumlah data (D)= 10.000.000 data.

$$\delta_n(v_1, v_2) = \frac{|v_1 - v_2|}{|D|} = \frac{|39 - 50|}{10.000.000} = \frac{11}{10.000.000} = 0.0000011$$

Distance atribut kategorikal dapat dihitung sebagai berikut berdasarkan workclass data pertama (v_1)= State-gov, workclass data kedua (v_2)= Self-emp-not-inc, jumlah subtree ($H(\Lambda(v_i, v_j))$)= 1, dan tinggi taxonomy tree ($H(T_D)$)= 1.

$$\delta_C(v_1, v_2) = \frac{H(\Lambda(v_i, v_j))}{H(T_D)} = \frac{1}{1} = 1$$



Gambar 3.2: Taxonomy Tree (Workclass)

Information Loss

Information Loss (IL) bertujuan untuk mengevaluasi seberapa baik kinerja algoritma K-Anonymity terhadap utilitas sebuah data. Information Loss (IL) juga dipakai untuk menentukan hasil pengelompokan data pada algoritma Greedy K-Member Clustering. Jika diberikan tabel data yang sudah dikelompokkan berdasarkan cluster, maka nilai Information Loss (IL) dapat dihitung.

Berikut adalah contoh hasil pengelompokan data pada cluster 1 untuk dataset Adults:

Age	Workclass	Education	Occupation	Sex	Income	Cluster Name
39	State-gov	Bachelors	Adm-clerical	Male	<=50K	Cluster 1
50	Self-emp-not-inc	Bachelors	Exec-managerial	Male	<=50K	Cluster 1
38	Private	HS-grad	Handlers-cleaners	Male	<=50K	Cluster 1
53	Private	11th	Handlers-cleaners	Male	<=50K	Cluster 1
28	Private	Bachelors	Prof-specialty	Female	<=50K	Cluster 1

Information Loss (IL) dapat dihitung sebagai berikut berdasarkan atribut numerik yaitu jumlah anggota cluster (e) = 5, MAX_{Age} = 53, MIN_{Age} = 28, N_{Age} = 5 mencakup atribut Age dan atribut kategorikal yaitu $H(\Lambda(\cup_{C_j}))$ = 1, $H(T_{C_j})$ = 1 mencakup atribut Workclass, Education, Occupation, Sex, dan Income.

$$\begin{aligned}
 D(e) &= \sum_{i=1}^m \frac{(MAX_{N_i} - MIN_{N_i})}{|N_i|} + \sum_{j=1}^n \frac{H(\Lambda(\cup_{C_j}))}{H(T_{C_j})} \\
 &= \frac{(53 - 28)}{5} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} + \frac{1}{1} = 10
 \end{aligned}$$

$$\begin{aligned}
 IL(e) &= |e| \cdot D(e) \\
 &= 5 \cdot 10 = 50
 \end{aligned}$$

Total Information Loss dihitung dari jumlah Information Loss masing-masing cluster.

$$\begin{aligned}
 Total - IL(AT) &= \sum_{e \in \varepsilon} IL(e) \\
 &= IL(cluster1) + IL(cluster2) + \dots + IL(clusterN)
 \end{aligned}$$

Cost Function

Cost Function bertujuan untuk menghitung kompleksitas algoritma Greedy K-Member Clustering yang sudah dijelaskan pada bagian 2.9. Kompleksitas akan dianalisis untuk menilai apakah kinerja algoritma sudah sesuai atau belum.

Berikut adalah tahapan kompleksitas dari Algoritma 1 yaitu Find Best Record:

1. Baris 5 sampai dengan baris 7, kompleksitasnya 1
2. Baris 8 sampai dengan baris 15, kompleksitasnya n
3. Kompleksitas totalnya adalah $T(n) = 1 + n$
4. Algoritma Finding Best Record memiliki kompleksitas $O(n)$

Berikut adalah tahapan kompleksitas dari Algoritma 2 yaitu Find Best Cluster:

1. Baris 5 sampai dengan baris 7, kompleksitasnya 1
2. Baris 8 sampai dengan baris 15, kompleksitasnya n
3. Kompleksitas totalnya adalah $T(n) = 1 + n$
4. Algoritma Finding Best Cluster memiliki kompleksitas $O(n)$

Berikut adalah tahapan kompleksitas dari Algoritma 3 yaitu Greedy K-Member Clustering:

1. Baris 5 sampai dengan baris 10, kompleksitasnya 1
2. Baris 11 sampai dengan baris 21, kompleksitasnya n^2
3. Baris 22 sampai dengan baris 27, kompleksitasnya n
4. Baris 28, kompleksitasnya 1
5. Kompleksitas totalnya adalah $T(n) = 1 + n^2 + n + 1$
6. Algoritma Finding Best Cluster memiliki kompleksitas $O(n^2)$

3.1.4 Greedy K-Member Clustering

Algoritma Greedy K-Member Clustering bertujuan untuk membagi seluruh data pada tabel terhadap masing-masing cluster untuk kompleksitas yang lebih baik dan mendukung nilai utilitas informasi yang lebih baik dibandingkan algoritma clustering lain. Pada bagian ini, akan dilakukan eksperimen sederhana mengenai algoritma Greedy K-Member Clustering terhadap dataset Adults untuk mencari tahu langkah kerja algoritma Greedy K-Member Clustering secara konseptual.

Diberikan contoh sampel dataset Adults sebagai berikut:

ID	Age	Workclass	Education	Occupation	Sex	Income
t1	39	State-gov	Bachelors	Adm-clerical	Male	<=50K
t2	50	Self-emp-not-inc	Bachelors	Exec-managerial	Male	<=50K
t3	38	Private	HS-grad	Handlers-cleaners	Male	<=50K
t4	53	Private	11th	Handlers-cleaners	Male	<=50K
t5	28	Private	Bachelors	Prof-specialty	Female	<=50K

Melalui sampel data pada tabel 2.1, akan diputuskan nilai dari setiap atribut anonimisasi. Jenis atribut anonimisasi yang pertama adalah Quasi-identifier, dengan nilai $QI = \{Age, Education, Occupation, Sex, Income\}$. Jenis atribut anonimisasi yang kedua adalah Sensitive Attribute, dengan nilai $SA = \{Workclass\}$. Jika telah diketahui tabel data seperti diatas, $k = 2$, dan jumlah cluster (m) = 2, maka algoritma Greedy K-Member Clustering siap ditelusuri lebih lanjut.

Berikut adalah tahapan yang terjadi pada algoritma Greedy K-Member Clustering:

1. Nilai awal result = \emptyset
2. Nilai awal $r = \{t1\}$
3. Nilai awal $|S| = 5$, $k = 2$
4. Karena kondisi $|S| \geq k$ terpenuhi, maka dilakukan perulangan sebagai berikut:
 - (a) Nilai r diubah menjadi $r = \{t3\}$, karena terbukti data $t3$ memiliki $\Delta(t1, t3) = 1.7189$ yang paling tinggi dari seluruh distance lain. Berikut adalah contoh perhitungannya:

$$\Delta(t_1, t_2) = 1.715$$

$$\Delta(t_1, t_3) = 2.431$$

$$\Delta(t_1, t_4) = 2.122$$

$$\Delta(t_1, t_5) = 1.621$$

- (b) Nilai awal $S = \{t1, t2, t4, t5\}$
- (c) Nilai awal $c = \{t3\}$, $|c| = 1$
- (d) Karena kondisi $|c| < k$ terpenuhi, maka dilakukan perulangan sebagai berikut:
 - i. Nilai r diubah menjadi $r = \{t3, t4\}$, karena terbukti data $t4$ memiliki $IL(t3 \cup t4) = 0.330$ yang paling rendah dari seluruh data lain. Berikut adalah contoh perhitungannya:

$$IL(t3 \cup t1) = 0.479$$

$$IL(t3 \cup t2) = 0.515$$

$$IL(t3 \cup t4) = 0.330$$

$$IL(t3 \cup t5) = 0.367$$

- ii. Nilai S diubah menjadi $S = \{t1, t2, t5\}$, $|S| = 4$
 - iii. Nilai c ditambahkan menjadi $c = \{t3, t4\}$, $|c| = 2$
- (e) Karena kondisi $|c| < k$ sudah tidak terpenuhi lagi, maka perulangan ini akan berhenti
- (f) Nilai result akan ditambahkan menjadi $result = \{t3, t4\}$
- (g) Karena kondisi $|S| \geq k$ masih terpenuhi, maka perulangan akan tetap berlanjut sampai pada kondisi dimana $|S| < k$ sehingga hasil akhirnya adalah $result = \{\{t3, t4\}, \{t2, t5\}\}$, $S = \{t1\}$, $|S| = 1$

5. Karena kondisi $S \neq 0$ terpenuhi, maka dilakukan perulangan sebagai berikut:

- (a) Nilai r diubah menjadi $r = \{t1\}$
- (b) Nilai S diubah menjadi $S = \{\phi\}$, $|S| = 0$
- (c) Nilai c diubah menjadi $c = \{t3, t4\}$ karena terbukti cluster c memiliki $IL(\{t3, t4\} \cup t1) = 0.279$ yang paling rendah dari seluruh cluster lain. Berikut adalah contoh perhitungannya:

$$IL(\{t3, t4\} \cup t1) = 0.279$$

$$IL(\{t2, t5\} \cup t1) = 0.515$$

- (d) Nilai c ditambahkan menjadi $c = \{t1, t3, t4\}$
 - (e) Nilai c pada perulangan ini tidak akan ditambahkan pada result, karena telah ditetapkan $k = 2$ sedangkan jumlah datanya ganjil, sehingga sisa data tersebut tidak akan dicatat pada result agar menjaga masing-masing cluster hanya memiliki 2 anggota saja.
 - (f) Karena kondisi $S \neq 0$ sudah tidak terpenuhi lagi, maka perulangan ini akan berhenti
6. Hasil akhirnya adalah $result = \{\{t3, t4\}, \{t2, t5\}\}$ dikembalikan sebagai output untuk algoritma Greedy K-Member Clustering seperti pada tabel berikut:

ID	Age	Workclass	Education	Occupation	Sex	Income
t3	38	Private	HS-grad	Handlers-cleaners	Male	<=50K
t4	53	Private	11th	Handlers-cleaners	Male	<=50K
t2	50	Self-emp-not-inc	Bachelors	Exec-managerial	Male	<=50K
t5	28	Private	Bachelors	Prof-specialty	Female	<=50K

3.1.5 Domain Generalization Hierarchy

Domain Generalization Hierarchy (DGH) bertujuan untuk melindungi data dengan cara menerapkan metode generalisasi terhadap nilai atribut data yang bersifat unik, agar menjadi nilai yang lebih umum. Nilai yang lebih umum biasa sulit untuk ditebak karena mengandung beberapa arti dalam satu nilai, berbeda dengan nilai yang bersifat unik yang memiliki satu arti dalam satu nilai. Pada bagian ini, akan dilakukan eksperimen sederhana mengenai penerapan DGH terhadap masing-masing atribut pada dataset Adults, untuk mencari tahu langkah kerja DGH secara konseptual.

Diketahui kemungkinan nilai atribut numerik pada dataset Adults adalah sebagai berikut:

- Age = {33,36,38,40,42,43,46,49}
- ZIP = {77516,77517,77526,77527}
- Sex = {Male,Female}
- Hours/week = {12,18,33,37,53,54}

Berikut adalah tahapan yang terjadi pada Domain Generalization Hierarchy:

1. Dipilih atribut quasi-identifier sebagai berikut $QI = \{\text{Age, ZIP, Sex, Hours per week}\}$
2. Diketahui domain dasar dari atribut ZIP yaitu $Z_0 = \{77516,77517,77526,77527\}$
3. Membuat domain yang kurang spesifik berdasarkan tiga cara:
 - Untuk atribut numerik dengan jumlah digit < 2 , maka hasilnya sebagai berikut jika atributnya Age, $Z_1 = \{[33-38],[40-49]\}$
 - Untuk atribut numerik dengan jumlah digit > 2 , maka hasilnya sebagai berikut jika atributnya ZIP, $Z_1 = \{7751*,7752*\}$
 - Untuk atribut kategorikal, maka hasilnya adalah nilai yang lebih umum sebagai berikut jika atributnya Sex, $Z_1 = \{\text{Person}\}$
4. Membuat domain yang lebih umum dari atribut ZIP yaitu $Z_2 = \{775**\}$
5. Membuat domain yang sangat umum dari atribut ZIP yaitu $Z_3 = \{*****\}$
6. Memilih domain sesuai kebutuhan, biasanya dipilih domain yang kurang spesifik seperti $Z_1 = \{7751*,7752*\}$ atau domain yang lebih umum seperti $Z_2 = \{775**\}$
7. Langkah ini diulangi sampai seluruh atribut quasi-identifier telah selesai digeneralisasi.
8. Hasil akhir dari DGH adalah sebagai berikut:

$$\begin{aligned}
 \text{Age} &= \{[33 - 38], [40 - 49]\} \\
 \text{ZIP} &= \{7751*, 7752*\} \\
 \text{Sex} &= \{\text{Person}\} \\
 \text{Hours/Week} &= \{[12 - 18], [33 - 37], [53 - 54]\}
 \end{aligned}$$

9. Proses generalisasi DGH telah selesai dilakukan. Tahap selanjutnya adalah proses anonimisasi data pada tabel dataset.

3.1.6 Anonimisasi

Anonimisasi bertujuan untuk menyamarkan nilai dari masing quasi-identifier yang unik pada kelompok cluster yang sama. Kata kuncinya adalah nilai unik pada kelompok cluster yang sama. Setelah dataset dilakukan anonimisasi, maka data privasi secara konseptual sudah terlindungi sehingga publikasi data dapat dilakukan dengan aman.

Diketahui kelompok data berdasarkan algoritma Greedy K-Member Clustering sebagai berikut:

ID	Age	Workclass	Education	ZIP	Sex	Hours/week	Cluster Name
t3	32	Private	HS-grad	77516	Male	30	Cluster 1
t4	32	Private	11th	77541	Female	30	Cluster 1
t2	34	Self-emp-not-inc	Bachelors	77526	Male	34	Cluster 2
t5	50	Private	Bachelors	77526	Male	37	Cluster 2
t1	47	Local-gov	Bachelors	77581	Male	54	Cluster 3
t6	50	Federal-gov	HS-grad	77532	Male	57	Cluster 3

Diketahui bentuk generalisasi berdasarkan Domain Generalization Hierarchy sebagai berikut:

$$Age = \{[20 - 30], [40 - 50]\}$$

$$ZIP = \{775 * *\}$$

$$Sex = \{Person\}$$

$$Hours/week = \{[12 - 18], [33 - 37], [53 - 61]\}$$

Berikut adalah tahapan yang terjadi pada proses anonimisasi:

1. Diketahui Quasi-Identifier sebagai berikut $QI = \{Age, ZIP, Sex, Hours/week\}$ dan sensitive attribute sebagai berikut $SA = \{Workclass, Education\}$
2. Mencari nilai Quasi-Identifier yang unik pada kelompok cluster yang sama. Sebagai contoh, cluster 2 memiliki nilai quasi-identifier yang unik sebagai berikut $QI = \{Age, Hours/week\}$
3. Melakukan generalisasi DGH pada nilai quasi-identifier yang unik menjadi bentuk . Sebagai contoh, $QI = \{Age, Hours/week\}$ memiliki nilai yang unik, sehingga diubah menjadi $Age = \{[40-50]\}$, $Hours/week = \{[33-37]\}$
4. Sensitive Attribute tidak akan dilakukan generalisasi, karena Quasi-Identifier sudah dilakukan generalisasi sehingga seseorang akan sulit untuk menebak kepemilikan dari Sensitive Attribute.
5. Ulangi hal yang sama pada langkah sebelumnya untuk setiap cluster. Hasil akhir dari tabel dataset setelah dianonimisasi adalah sebagai berikut:

ID	Age	Workclass	Education	ZIP	Sex	Hours/week	Cluster Name
t3	32	Private	HS-grad	775**	Person	30	Cluster 1
t4	32	Private	11th	775**	Person	30	Cluster 1
t2	[40-50]	Self-emp-not-inc	Bachelors	77526	Male	[33-37]	Cluster 2
t5	[40-50]	Private	Bachelors	77526	Male	[33-37]	Cluster 2
t1	[40-50]	Local-gov	Bachelors	775**	Male	[53-61]	Cluster 3
t6	[40-50]	Federal-gov	HS-grad	775**	Male	[53-61]	Cluster 3

6. Proses anonimisasi data telah selesai dilakukan. Pada tahap ini dataset telah aman untuk dilakukan publikasi data.

3.2 Eksplorasi Spark

Pada bagian ini akan dilakukan penelusuran lebih lanjut mengenai beberapa hal penting terkait Spark sebelum melakukan eksperimen metode anonimisasi pada Spark.

Berikut adalah beberapa hal penting terkait Spark:

- Spark bekerja sama dengan komponen lain seperti JDK, SBT, HDFS sehingga instalasi Spark untuk masing-masing sistem operasi dapat berbeda. Pada penelitian ini, akan dilakukan instalasi Spark melalui sistem operasi Windows.
- Spark dapat bekerja dengan bahasa pemrograman Scala. Scala dipilih karena memiliki efektivitas yang baik pada penulisan kode program. Scala dapat menyederhanakan perintah pada Spark menjadi baris yang lebih sedikit.
- Program Spark dijalankan dengan cara membuat jar sebelum perintah eksekusi dijalankan. Hal ini menghambat pekerjaan pada tahap implementasi perangkat lunak. Intel IJ adalah sebuah Integrated Development Environment (IDE) yang memfasilitasi pemrograman Scala pada Spark dan menampilkan hasil pemrosesan Spark secara langsung.
- Spark menyediakan konfigurasi untuk mengatur jumlah resource yang dibutuhkan (jumlah pemakaian RAM, core CPU) pada pemrosesan data. Konfigurasi ini bertujuan agar Spark dapat mengolah data yang besar secara maksimal dengan menggunakan jumlah resource yang tersedia. Konfigurasi ini ditulis pada perintah eksekusi Spark.

3.2.1 Instalasi Spark

Spark dapat berjalan pada sistem operasi Windows, Linux, dan Mac OS. Spark dapat dijalankan secara lokal menggunakan satu komputer, meskipun nyatanya Spark tetap membutuhkan beberapa komputer agar pemrosesan data yang besar dapat menjadi lebih cepat. Pada bagian ini akan dijelaskan tahapan instalasi Spark untuk versi 2.4.5 pada sistem operasi Windows dari awal hingga akhir. Sebelum melakukan instalasi Spark, ada beberapa hal yang harus diperhatikan dan dipenuhi.

Berikut adalah beberapa hal yang harus diperhatikan:

- Java 7, Python 2.6 telah dihilangkan pada implementasi Spark 2.2.0
- Scala 2.10 sudah tidak dapat dipakai, Scala 2.11 sudah terlalu usang untuk dipakai pada Spark 2.4.1 dan akan dihilangkan pada Spark 3.0
- Hadoop 2.6.5 telah dihilangkan pada implementasi Spark 2.2.0

Berikut adalah beberapa hal yang harus dipenuhi:

- Spark 2.4.5 berjalan di Java 8, Python 2.7+/3.4+ dan R 3.1+
- Spark 2.4.5 menggunakan Scala 2.12
- Spark 2.4.5 menggunakan Hadoop 2.7

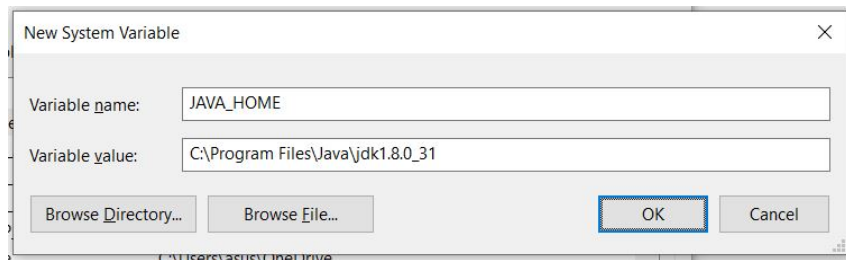
Berikut adalah tahapan instalasi Spark 2.4.5 secara umum:

1. Melakukan instalasi Java 8.
2. Melakukan instalasi Spark 2.4.5
3. Melakukan instalasi IntelliJ untuk Scala sbt.

Instalasi Java 8

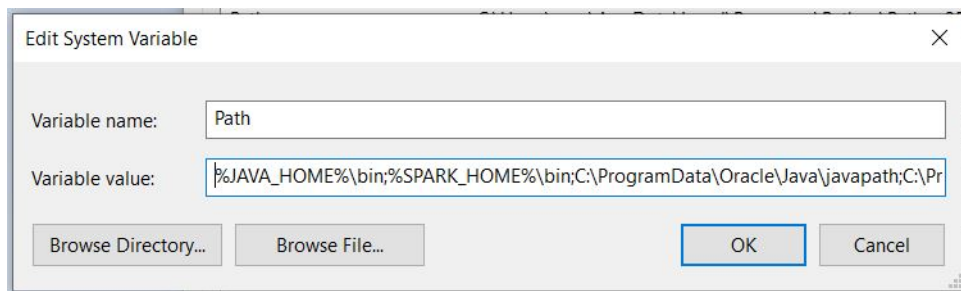
Berikut adalah tahapan instalasi Java 8 secara lengkap:

1. Download Java SE Development Kit 8u31 pada link berikut <https://www.oracle.com/technetwork/java/javase/downloads/java-archive-javase8-2177648.html>
2. Lakukan instalasi Java SE Development Kit 8u31 seperti biasa.
3. Pilih menu Edit the system environment variables.
4. Buat environment variables baru seperti gambar berikut.



Gambar 3.3: Environment Variables

5. Tambahkan %JAVA_HOME%\bin; pada Path di System variables



Gambar 3.4: Penambahan Variable Value

6. Pada tahap ini Java 8 sudah terpasang, langkah selanjutnya adalah melakukan verifikasi instalasi untuk memeriksa apakah instalasi tersebut dapat berjalan dengan baik.

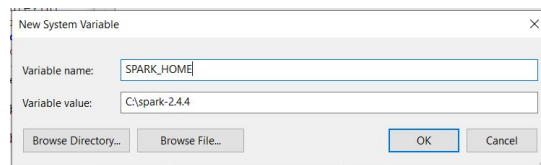
Berikut adalah tahapan verifikasi terhadap instalasi Java 8:

1. Pilih menu Command Prompt.
2. Jalankan perintah `java -version` pada Command Prompt.
3. Apabila sistem tidak menampilkan pesan error, maka Java 8 sudah terpasang dengan baik.

Instalasi Spark 2.4.5

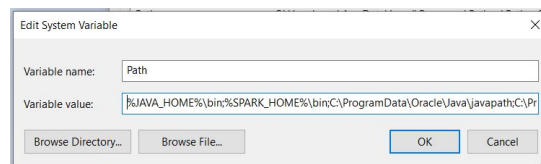
Berikut adalah tahapan instalasi Spark 2.4.5 secara lengkap:

1. Download winutils.exe dari link <https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin>
2. Buat folder seperti berikut C:\winutils\bin dan tempatkan winutils.exe di dalam folder tersebut.
3. Download Spark 2.4.5 dari link <https://downloads.apache.org/spark/spark-3.0.0-preview2/spark-3.0.0-preview2-bin-hadoop2.7.tgz>
4. Buat folder sebagai berikut C:\spark-2.4.4 dan ekstraksi file spark-2.4.5-bin-hadoop2.7.tgz di dalam folder tersebut.
5. Pilih menu Edit the system environment variables
6. Buat environment variables baru seperti gambar berikut.



Gambar 3.5: Environment Variable

7. Tambahkan %SPARK_HOME%\bin; pada Path di System variables

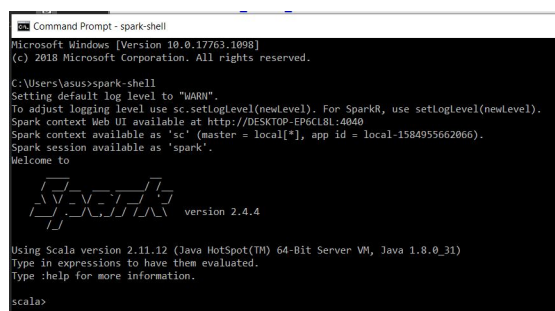


Gambar 3.6: Penambahan Variable Value

8. Pada tahap ini Spark 2.4.5 sudah terpasang, langkah selanjutnya adalah melakukan verifikasi instalasi untuk memeriksa apakah instalasi tersebut dapat berjalan dengan baik.

Berikut adalah tahapan verifikasi terhadap instalasi Spark 2.4.5:

1. Jalankan perintah **spark-shell** pada Command Prompt.
2. Apabila tampilan berubah seperti gambar berikut, artinya Spark 2.4.5 sudah dapat berjalan dengan baik.



Gambar 3.7: Spark 2.4.5

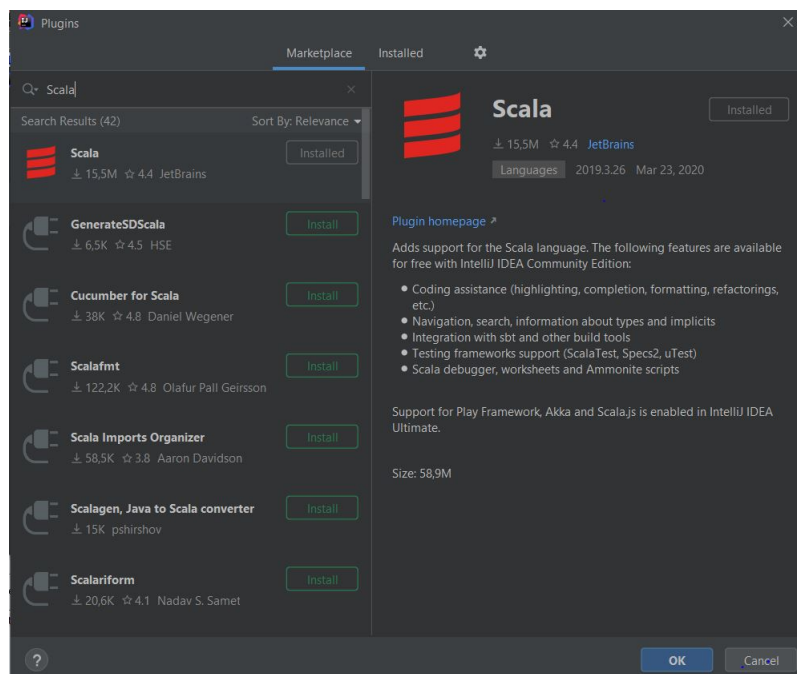
Instalasi IntelliJ untuk Scala

Berikut adalah tahapan instalasi IntelliJ:

1. Download IntelliJ melalui link berikut <https://www.jetbrains.com/idea/download/#section=windows>
2. Lakukan instalasi IntelliJ seperti biasa.
3. Pada tahap ini IntelliJ sudah terpasang dengan baik pada komputer, langkah selanjutnya adalah melakukan instalasi plugin Scala pada IntelliJ.

Berikut adalah tahapan pemasangan plugin Scala pada IntelliJ.

1. Pilih menu Configure pada IntelliJ, lalu pilih menu Plugins.
2. Telusuri plugins Scala pada kolom pencarian.



Gambar 3.8: Plugins Scala

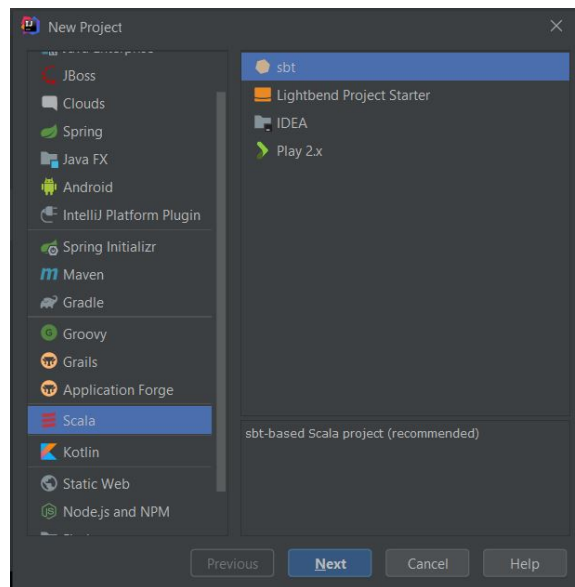
3. Klik tombol install
4. Pada tahap ini program Spark sudah dapat berjalan dan dikompilasi dengan IntelliJ, langkah selanjutnya adalah membuat project Spark baru pada IntelliJ.

3.2.2 Membuat Project Spark pada IntelliJ

Untuk membuat program Spark, pertama-tam perlu membuat project Spark baru untuk merancang kelas-kelas yang dibutuhkan pada eksekusi Spark. Beberapa hal yang perlu diperhatikan adalah menggunakan versi Scala sbt, memilih versi sbt 1.3.9, memilih versi Scala 2.11.12, dan melakukan import libraryDependencies Spark sesuai kebutuhan.

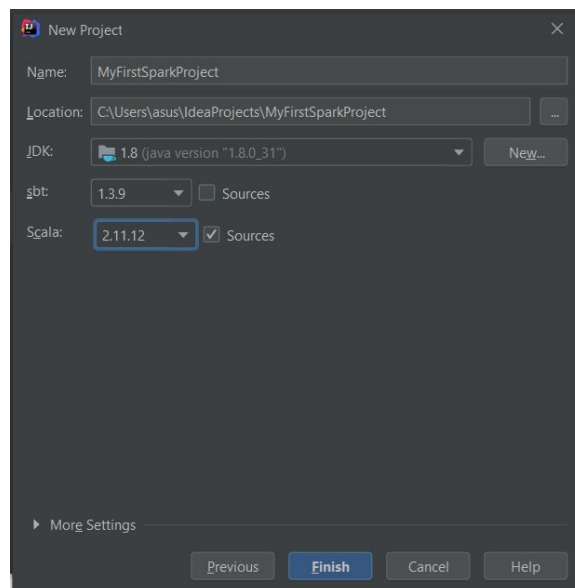
Berikut adalah tahapan pembuatan project Spark pada IntelliJ:

1. Memilih menu Create New Project
2. Menggunakan bahasa pemrograman Scala dengan berbasis sbt.



Gambar 3.9: Plugins Scala

3. Melakukan konfigurasi pada project Spark baru



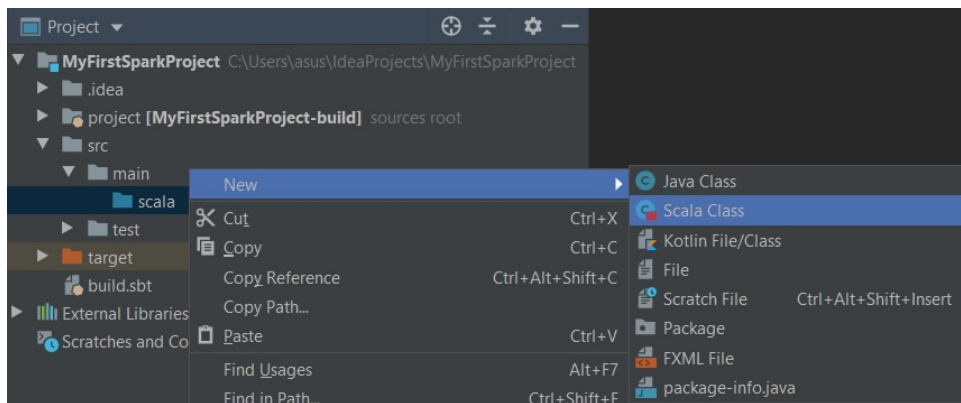
Gambar 3.10: Plugins Scala

4. Melakukan import libraryDependencies sesuai kebutuhan pada file `build.sbt`
Contoh: `spark-core`, `spark-sql`, `spark-mllib`.

Listing 3.1: Main method

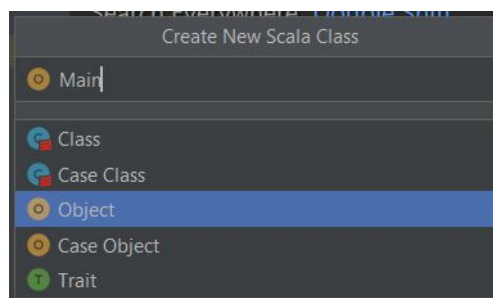
```
name := "NamaProject"
version := "0.1"
scalaVersion := "2.11.12"
// https://mvnrepository.com/artifact/org.apache.spark/spark-core
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.2.0"
// https://mvnrepository.com/artifact/org.apache.spark/spark-sql
libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.4.0"
// https://mvnrepository.com/artifact/org.apache.spark/spark-mllib
libraryDependencies += "org.apache.spark" %% "spark-mllib" % "2.4.3"
```

5. Menambahkan Scala class pada `src/main/scala`.



Gambar 3.11: Plugins Scala

6. Memilih tipe Scala class sebagai Object.



Gambar 3.12: Plugins Scala

7. Menambahkan main method pada Scala class.

Listing 3.2: Main method

```
object Main {
  def main(args: Array[String]) {
    //statement
  }
}
```

3.2.3 Membuat File JAR pada Command Prompt

Sebelum menjalankan program Spark pada Hadoop Cluster, program Spark yang sudah jadi harus dibuat menjadi file JAR terlebih dahulu. Hal ini disebabkan karena perintah Spark hanya menerima input berupa kode program dalam format (.jar).

Berikut adalah tahapan untuk membuat JAR:

1. Membuka folder pengerjaan project Spark `\IdeaProjects\NamaProject`
2. Membuka command prompt pada folder project.
3. Mengeksekusi perintah `sbt package` pada command prompt
4. Menunggu proses pembuatan file JAR oleh sistem, apabila terminal tidak menampilkan pesan error maka file JAR telah berhasil dibuat dan tersimpan pada folder tertentu.
5. File JAR yang telah dibuat akan tersimpan pada `NamaProject\target\scala-2.11`

3.2.4 Menjalankan Program Spark pada Command Prompt

Apabila ukuran data input eksperimen kecil, maka program Spark dapat dijalankan pada komputer lokal menggunakan perintah dari Command Prompt.

Berikut adalah tahapan menjalankan program Spark pada Command Prompt:

1. Membuka command prompt pada komputer lokal.
2. Menjalankan perintah eksekusi Spark sebagai berikut `spark-submit --class NamaMainClass --master local[*] lokasi_jar\nama_jar.jar` pada command prompt
3. Menunggu proses eksekusi file JAR oleh komputer lokal, apabila terminal tidak menampilkan pesan error maka program Spark berhasil dijalankan dengan baik.

3.2.5 Menjalankan Program Spark pada Hadoop Cluster

Karena ukuran data input eksperimen terbilang besar yaitu mencapai 1GB, maka akan lebih efektif apabila komputasi dilakukan secara paralel melalui Hadoop cluster. Hadoop cluster terdiri dari beberapa perangkat komputer yang dapat saling bekerja sama, sehingga proses komputasi dapat dilakukan lebih cepat.

Berikut adalah tahapan menjalankan program Spark pada Hadoop cluster:

1. Membuka command prompt pada komputer lokal.
2. Menyambungkan jaringan komputer lokal dengan server Hadoop cluster menggunakan perintah `ssh hduser@10.100.69.101` pada command prompt.
3. Melakukan upload file JAR dari komputer lokal ke folder Hadoop cluster menggunakan perintah `scp nama_jar.jar hduser@10.100.69.101:nama_folder` pada command prompt.
4. Menjalankan perintah eksekusi Spark sebagai berikut `spark-submit --class NamaMainClass --master yarn lokasi_jar\nama_jar.jar` pada command prompt
5. Menunggu proses eksekusi file JAR oleh Hadoop cluster, apabila terminal tidak menampilkan pesan error maka program Spark berhasil dijalankan dengan baik.

3.3 Studi Kasus

Untuk memahami implementasi algoritma anonimisasi pada Spark, maka dilakukan studi kasus terhadap fungsi Spark yang umum digunakan, seperti fungsi dasar pada Spark, fungsi dasar pada komponen Spark, dan fungsi dasar pada Spark MLlib. Bentuk dari studi kasus yang akan dilakukan adalah memberikan contoh kode program berikut penjelasan singkat mengenai tujuan pemanggilan fungsi, parameter input, dan contoh output yang dikeluarkan oleh fungsi tersebut.

3.3.1 Eksperimen Scala

Menentukan Jenis Variabel pada Scala

Scala memiliki dua jenis variabel yaitu immutable variabel dan mutable variabel. Immutable variabel adalah variabel yang nilainya tidak dapat diubah, sedangkan mutable variabel adalah variabel yang nilainya dapat diubah. Implementasi immutable dan mutable memiliki implementasi sintaks yang berbeda. Immutable variabel menggunakan sintaks `val`, sedangkan mutable variabel menggunakan sintaks `var`. Kode program pada bagian ini dapat dilihat pada Listing 3.3 mengenai jenis variabel pada Scala.

Listing 3.3: Menentukan Jenis Variabel pada Scala

```
// Immutable Variabel
val donutsToBuy: Int = 5
donutsToBuy = 10

// Mutable Variabel
var favoriteDonut: String = "Glazed Donut"
favoriteDonut = "Vanilla Donut"
```

Menentukan Jenis Tipe Data pada Scala

Scala memiliki jenis tipe data yang mirip dengan tipe data pada bahasa pemrograman Java. Scala dapat menangani tipe data `Int`, `Long`, `Short`, `Double`, `Float`, `String`, `Byte`, `Char` dan `Unit`. Kode program pada bagian ini dapat dilihat pada Listing 3.4 mengenai jenis tipe data pada Scala.

Listing 3.4: Menentukan Jenis Tipe Data pada Scala

```
val donutsBought: Int = 5
val bigNumberOfDonuts: Long = 1000000000L
val smallNumberOfDonuts: Short = 1
val priceOfDonut: Double = 2.50
val donutPrice: Float = 2.50f
val donutStoreName: String = "allaboutscala Donut Store"
val donutByte: Byte = 0xa
val donutFirstLetter: Char = 'D'
val nothing: Unit = ()
```

Menentukan Struktur Data pada Scala

Scala memiliki dua jenis struktur data yaitu immutable dan mutable collection. Immutable collection adalah struktur data yang nilainya tidak dapat diubah, sedangkan mutable collection adalah struktur data yang nilainya dapat diubah. Implementasi immutable dan mutable collection memiliki jenis struktur data yang berbeda satu sama lain. Kode program pada bagian ini dapat dilihat pada Listing 3.5 mengenai immutable collection pada Scala dan Listing 3.6 mengenai mutable collection pada Scala.

Listing 3.5: Membuat immutable collection pada Scala

```
// List
val list1: List[String] = List("Plain Donut","Strawberry Donut","Chocolate Donut
    ")
println(s"Elements of list1 = $list1")

// Map
val map1: Map[String, String] = Map(("PD","Plain Donut"),("SD","Strawberry Donut
    "),("CD","Chocolate Donut"))
println(s"Elements of map1 = $map1")
```

Listing 3.6: Membuat mutable collection pada Scala

```
// Array
val array1: Array[String] = Array("Plain Donut","Strawberry Donut","Chocolate
    Donut")
println(s"Elements of array1 = ${array1.mkString(", ")}")

// Map
val map1: Map[String, String] = Map(("PD","Plain Donut"),("SD","Strawberry Donut
    "),("CD","Chocolate Donut"))
println(s"Elements of map1 = $map1")
```

Membuat Kelas Object pada Scala

Scala menggunakan kelas Object untuk menempatkan berbagai macam fungsi dan variabel yang saling berkaitan pada satu kelas yang sama. Kode program pada bagian ini dapat dilihat pada Listing 3.7 mengenai kelas object pada Scala.

Listing 3.7: Membuat Kelas Object pada Scala

```
object DonutShoppingCartCalculator {

    val discount: Double = 0.01

    def calculateTotalCost(donuts: List[String]): Double = {
        // calculate the cost of donuts
        return 1
    }

}
```

Membuat Fungsi Sederhana pada Scala

Scala menggunakan fungsi untuk menempatkan kode program berdasarkan tujuan masing-masing. Perlu diperhatikan bahwa hasil akhir dari fungsi langsung dikembalikan tanpa memanggil perintah `return`, seperti pada Java. Kode program pada bagian ini dapat dilihat pada Listing 3.8 mengenai pembuatan fungsi pada Scala.

Listing 3.8: Membuat Fungsi Sederhana pada Scala

```
def calculateDonutCost(donutName: String, quantity: Int): Double = {  
    println(s"Calculating cost for $donutName, quantity = $quantity")  
  
    // make some calculations ...  
    2.50 * quantity  
}
```

Membuat Main Method

Scala menggunakan main method untuk melakukan eksekusi program. Kode program pada bagian ini dapat dilihat pada Listing 3.9 mengenai pembuatan main method pada Scala.

Listing 3.9: Membuat Main Method pada Scala

```
object HelloWorld {  
    def main(args: Array[String]) {  
        println("Hello, world!")  
    }  
}
```

Membuat Fungsi Percabangan

Scala memiliki jenis implementasi percabangan yang sama dengan Java. Percabangan digunakan untuk melakukan eksekusi pada baris statement yang sesuai berdasarkan kondisi tertentu. Kode program pada bagian ini dapat dilihat pada Listing 3.10 mengenai percabangan pada Scala.

Listing 3.10: Membuat Fungsi Percabangan pada Scala

```
# If-Else statement  
if(numberOfPeople > 10) {  
    println(s"Number of donuts to buy = ${numberOfPeople * donutsPerPerson}")  
}  
else if (numberOfPeople == 0) {  
    println("Number of people is zero.")  
    println("No need to buy donuts.")  
}  
else {  
    println(s"Number of donuts to buy = $defaultDonutsToBuy")  
}
```


Membuat Fungsi Perulangan pada Scala

Scala memiliki jenis implementasi perulangan yang sama dengan Java. Perulangan digunakan untuk mengulangi eksekusi pada baris statement yang sama berdasarkan kondisi tertentu. Kode program pada bagian ini dapat dilihat pada Listing 3.11 mengenai perulangan pada Scala.

Listing 3.11: Membuat Fungsi Perulangan pada Scala

```
# For loop
for(numberOfDonuts <- 1 to 5){
  println(s"Number of donuts to buy = $numberOfDonuts")
}

# While loop
while (numberOfDonutsToBake > 0) {
  println(s"Remaining donuts to be baked = $numberOfDonutsToBake")
  numberOfDonutsToBake -= 1
}

# Do-while loop
do {
  numberOfDonutsBaked += 1
  println(s"Number of donuts baked = $numberOfDonutsBaked")
}
while (numberOfDonutsBaked < 5)
```

3.3.2 Eksperimen Spark

Fungsi dasar pada Spark terdiri dari membuat RDD, membuat dataframe, dan membuat fungsi transformation dan action.

Melakukan Konfigurasi Spark

Berikut adalah tahapan konfigurasi Spark pada Main Class:

- Membuat objek SparkConf untuk inisialisasi project Spark
- Menetapkan jumlah core CPU yang bekerja pada perintah setMaster()
- Menetapkan nama program Spark pada perintah setAppName()
- Membuat objek SparkContext untuk membuat RDD.

Listing 3.12: Konfigurasi Spark

```
val conf = new SparkConf()
conf.setMaster("local[2]")
conf.setAppName("Tutorial Spark")
val sc = new SparkContext(conf)
```

Membuat RDD

Berikut adalah beberapa cara untuk membuat RDD:

- Membaca data eksternal pada Spark sebagai RDD
- Membuat RDD dari struktur data list
- Merubah Dataframe menjadi RDD

Listing 3.13: Cara Pembuatan RDD

```
# 1. Membaca data eksternal pada Spark sebagai RD
rdd = sc.textFile("path")

# 2. Membuat RDD dari struktur data list
rdd = sc.parallelize(["id","name","3","5"])

# 3. Merubah Dataframe menjadi RDD
rdd = df.rdd
```

Membuat Dataframe

Berikut adalah beberapa cara untuk membuat Dataframe:

- Membaca data eksternal sebagai Dataframe
- Mengubah RDD menjadi Dataframe dengan nama kolom
- Mengubah RDD menjadi Dataframe dengan skema

Listing 3.14: Cara Pembuatan Dataframe

```
# 1. Membaca data eksternal sebagai Dataframe
# header and schema are optional
df = sqlContext.read.csv("path", header = True/False, schema=df_schema)

# 2.1 Mengubah RDD menjadi Dataframe dengan nama kolom
df = spark.createDataFrame(rdd,["name","age"])

# 2.2 Mengubah RDD menjadi Dataframe dengan skema
from pyspark.sql.types import *
df_schema = StructType([
...     StructField("name", StringType(), True),
...     StructField("age", IntegerType(), True)])
df = spark.createDataFrame(rdd,df_schema)
```

Memanggil Fungsi Transformation

Listing 3.15 adalah contoh penerapan jenis-jenis fungsi Transformation pada Spark, berikut adalah penjelasan singkat dari masing-masing fungsi:

- `select()`
- `filter()` dan `where()`
- `sort()` dan `orderBy()`
- `groupBy()` dan `agg()`
- `join()`

Listing 3.15: Contoh Fungsi Transformation

```
# 1. select
df.select(df.name)
df.select("name")

# 2. filter/where
df.filter(df.age>20)
df.filter("age>20")
df.where("age>20")
df.where(df.age>20)

# 3. sort/orderBy
df.sort("age",ascending=False)
df.sort(df.age.desc())

# 4. groupBy dan agg
df.groupBy("gender").agg(count("name"),avg("age"))

# 5. join
df1.join(df.2, (df1.x1 == df2.x1) & (df1.x2 == df2.x2),'left')
```

Memanggil Fungsi Action

Listing 3.16 adalah contoh penerapan jenis-jenis fungsi Action pada Spark, berikut adalah penjelasan singkat dari masing-masing fungsi:

- `show()`: menampilkan n baris pertama dari Dataframe atau RDD
- `take()`: menampilkan beberapa baris dari Dataframe atau RDD
- `collect()`: mengumpulkan seluruh data dari Dataframe atau RDD
- `count()`: menghitung jumlah baris
- `printSchema()`: menampilkan nama kolom dan tipe data
- `cache()`: menyimpan data ke dalam cache apabila data sering digunakan
- `unpersist()`: mengosongkan isi cache

Listing 3.16: Contoh Fungsi Action

```
# 1. show()
df.show(5)

# 2. take()
df.take(5)

# 3. collect()
df.collect()

# 4. count()
df.count()

# 6. printSchema()
df.printSchema()

# 7. transformation, action
df1.filter("age>10").join(df2,df1.x==df2.y).sort("age").show()
```

Memanggil Fungsi RDD

Listing 3.17 adalah contoh penerapan jenis-jenis fungsi RDD pada Spark, berikut adalah penjelasan singkat dari masing-masing fungsi RDD:

- `repartition(n)`: membagi RDD menjadi n buah partisi
- `cache()`: menyimpan RDD pada penyimpanan memori.
- `persist()`: menyimpan RDD pada penyimpanan memori atau disk.
- `unpersist()`: menghapus RDD pada memori atau disk.
- `foreach(println)`: melakukan print seluruh baris data pada RDD
- `saveAsTextFile(path)`: menyimpan RDD pada sebuah file

Listing 3.17: Contoh Fungsi RDD

```
rdd.repartition(4)
rdd.cache()
rdd.persist()
rdd.unpersist()
rdd.foreach(println)
rdd.saveAsTextFile(path)
```

Membuat Variabel Global

Listing 3.18 adalah contoh penerapan variabel global pada Spark.

Listing 3.18: Membuat Variabel Global

```
val broadcastVar = sc.broadcast(Array(1, 2, 3))
```

3.3.3 Eksperimen Komponen Spark

Spark Core

Berikut adalah langkah-langkah eksperimen dari Spark Core:

1. Membuat perintah SparkSession untuk inisialisasi project Spark

Listing 3.19: Main method

```
val spark: SparkSession = SparkSession.builder()  
    .master("local[3]")  
    .appName("SparkCoreAdults")  
    .getOrCreate()
```

2. Melihat dan mengatur partisi yang terbentuk pada RDD berdasarkan data input CSV

Listing 3.20: Main method

```
val sc = spark.sparkContext  
  
val rdd: RDD[String] = sc.textFile("input/adult100k.csv")  
println("initial partition count:" + rdd.getNumPartitions)  
  
val reparable = rdd.repartition(4)  
println("re-partition count:" + reparable.getNumPartitions)
```

3. Melakukan implementasi jenis-jenis fungsi transformation.

Listing 3.21: Main method

```
//Transformation - flatMap  
val rdd2 = rdd.flatMap(f => f.split(","))  
rdd2.foreach(f => println(f))  
  
//Transformation - map  
val rdd3: RDD[(String, Int)] = rdd2.map(key => (key, 1))  
rdd3.foreach(println)  
  
//Transformation - filter  
val rdd4 = rdd3.filter(a => a._1.startsWith("State-gov"))  
rdd4.foreach(println)  
  
//Transformation - reduceByKey  
val rdd5 = rdd3.reduceByKey((x,y)=> x + y)  
rdd5.foreach(println)  
  
//Transformation - sortByKey  
val rdd6 = rdd5.map(a => (a._2, a._1)).sortByKey()
```

4. Melakukan implementasi jenis-jenis fungsi action.

Listing 3.22: Main method

```
//Action - count
println("Count : " + rdd6.count())

//Action - first
val firstRec = rdd6.first()
println("First Record : " + firstRec._1 + "," + firstRec._2)

//Action - max
val datMax = rdd6.max()
println("Max Record : " + datMax._1 + "," + datMax._2)

//Action - reduce
val totalWordCount = rdd6.reduce((a, b) => (a._1 + b._1, a._2))
println("dataReduce Record : " + totalWordCount._1)

//Action - take
val data3 = rdd6.take(3)
data3.foreach(f => {
    println("data3 Key:" + f._1 + ", Value:" + f._2)
})

//Action - collect
val data = rdd6.collect()
data.foreach(f => {
    println("Key:" + f._1 + ", Value:" + f._2)
})

//Action - saveAsTextFile
rdd5.saveAsTextFile("c:/tmp/wordCount")
```

Spark SQL

Berikut adalah langkah-langkah eksperimen dari Spark Core:

1. Membuat perintah SparkSession untuk inisialisasi project Spark

Listing 3.23: Main method

```
val spark = SparkSession
    .builder.master("local[*]")
    .appName("SparkSQL")
    .getOrCreate()
```

2. Membuat dataframe dari data input CSV

Listing 3.24: Main method

```
val peopleDF = spark.read
    .format("csv")
    .option("header", "true")
    .option("delimiter", ",")
    .load("input/adult100k.csv")

peopleDF.printSchema()
```

3. Membuat tabel sementara, melakukan kueri, dan menyimpan hasil kueri pada file CSV.

Listing 3.25: Main method

```
// Query
peopleDF.createTempView("tAdults")
val query = spark.sql(
    "SELECT workclass,count(education) as count_people"+
    "FROM tAdults " +
    "WHERE lower(education) == 'bachelors'" +
    "GROUP BY workclass " +
    "ORDER BY count_people DESC " +
    "LIMIT 10"
)
query.write.csv("C:/Users/asus/Desktop/resultquery1.csv")
```

4. Mencari nilai statistik seperti jumlah data, mean, standar deviasi, nilai minimum dan maksimum.

Listing 3.26: Main method

```
// Statistika: count, mean, stddev, min, max
peopleDF.describe().show()
```

5. Mencari nilai median.

Listing 3.27: Main method

```
// Median
val median = spark.sql(
    "SELECT percentile_approx(age, 0.5) as Median " +
    "FROM tAdults"
).show()
```

6. Mencari nilai modus.

Listing 3.28: Main method

```
// Modus
val modus = spark.sql(
    "SELECT age as Modus " +
    "FROM tAdults " +
    "GROUP BY age " +
    "ORDER BY COUNT(age) DESC " +
    "LIMIT 1"
).show()
```

Spark MLlib

Berikut adalah langkah-langkah eksperimen dari Spark Core:

1. Membuat perintah SparkSession untuk inisialisasi project Spark

Listing 3.29: Main method

```
val spark = SparkSession
    .builder.master("local[*]")
    .appName("SparkMLlib")
    .getOrCreate()
```

2. Membuat skema untuk dataframe

Listing 3.30: Main method

```
val schema = StructType(
    List(
        StructField("age", IntegerType, true),
        StructField("workclass", StringType, true),
        StructField("fnlwgt", IntegerType, true),
        StructField("education", StringType, true),
        StructField("education-num", IntegerType, true),
        StructField("marital-status", StringType, true),
        StructField("occupation", StringType, true),
        StructField("relationship", StringType, true),
        StructField("race", StringType, true),
        StructField("sex", StringType, true),
        StructField("capital-gain", IntegerType, true),
        StructField("capital-loss", IntegerType, true),
        StructField("hours-per-week", IntegerType, true),
        StructField("native-country", StringType, true),
        StructField("salary", StringType, true)
    )
)
```


3. Mengubah data input CSV menjadi dataframe berdasarkan skema

Listing 3.31: Main method

```
val adult100k_df = spark.read
    .format("csv")
    .option("header", "false")
    .option("delimiter", ",")
    .schema(schema)
    .load("input/adult100k.csv")

adult100k_df.show()
```

4. Menggunakan fungsi stringIndexer untuk membuat kolom index dan fungsi oneHotEncoder untuk membuat kolom vektor.

Listing 3.32: Main method

```
// Create vector based on stringIndexer and oneHotEncoder
val cols = adult100k_df.columns
val encodedFeatures = cols.flatMap{columnName =>
    val stringIndexer = new StringIndexer()
        .setInputCol(columnName)
        .setOutputCol(columnName + "_Index")
    val oneHotEncoder = new OneHotEncoderEstimator()
        .setInputCols(Array(columnName + "_Index"))
        .setOutputCols(Array(columnName + "_vec"))
        .setDropLast(false)
    Array(stringIndexer.setHandleInvalid("keep"), oneHotEncoder)
}
```

5. Menambahkan kolom vektor dan kolom index pada dataframe

Listing 3.33: Main method

```
// Pipeline
val pipeline = new Pipeline().setStages(encodedFeatures)
val indexer_model = pipeline.fit(adult100k_df)
```

6. Memilih jenis implementasi vektor yang akan digunakan.

Listing 3.34: Main method

```
// Sparse Vector
val df_transformed = indexer_model.transform(adult100k_df)
df_transformed.show()

// Dense Vector
```

```
val sparseToDense = udf((v: Vector) => v.toDense)
val df_denseVectors = df_transformed
    .withColumn("dense_workclass_vec",
        sparseToDense(df_transformed("workclass_vec")))
df_denseVectors.show()
```

7. Membuat vektor fitur untuk setiap baris data pada dataframe

Listing 3.35: Main method

```
// Final Result: Feature Vector
val vecFeatures = df_transformed.columns.filter(_.contains("vec"))
val vectorAssembler = new VectorAssembler()
    .setInputCols(vecFeatures)
    .setOutputCol("features")
val pipelineVectorAssembler = new Pipeline()
    .setStages(Array(vectorAssembler))
val result_df = pipelineVectorAssembler
    .fit(df_transformed)
    .transform(df_transformed)
result_df.show()
```

3.3.4 Eksperimen Spark MLIB

Naive Bayes

Berikut adalah tahapan eksperimen pada pemodelan Naive Bayes:

1. Membagi data input CSV menjadi training data dan test data.
2. Membuat model Naive Bayes menggunakan Spark MLlib
3. Melakukan pelatihan data pada pemodelan Naive Bayes.
4. Mengembalikan hasil klasifikasi dalam bentuk tabel.
5. Menghitung akurasi dari klasifikasi kelompok cluster.

Listing 3.36: Main method

```
// Split data into training (70%) and test (30%).
val Array(training, test) = result_df.randomSplit(Array(0.7, 0.3))

// Naive Bayes
val model = new NaiveBayes().setModelType("multinomial")
    .setLabelCol("workclass_Index").fit(training)

// Predict model
val predictions = model.transform(test)
predictions.show()
```

```
// Accuracy
val evaluator = new MulticlassClassificationEvaluator()
    .setLabelCol("workclass_Index")
    .setPredictionCol("prediction")
    .setMetricName("accuracy")
val accuracy = evaluator.evaluate(predictions)
println("Test set accuracy = " + accuracy)
```

Hasil dari pemodelan Naive Bayes adalah prediksi jenis cluster berdasarkan sifat dari masing-masing data. Umumnya pada hasil pemodelan Naive Bayes data-data yang memiliki sifat yang sama letaknya berdekatan satu sama lain. Pada Gambar 3.14, diketahui bahwa data dengan nilai umur yang sama (age = 17) memiliki kelompok cluster yang sama (prediction = 3.0).

```
+-----+-----+
| age|prediction|
+-----+-----+
| null|      8.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
|  17|      3.0|
+-----+-----+
only showing top 10 rows
```

Gambar 3.13: Hasil Naive Bayes

K-Means

Berikut adalah tahapan eksperimen pada pemodelan K-Means:

1. Membuat model K-Means menggunakan Spark MLlib
2. Menentukan jumlah cluster (k) untuk pemodelan K-Means.
3. Melakukan pelatihan data pada pemodelan K-Means.
4. Mencari nilai centroid dari masing-masing cluster.
5. Mengembalikan hasil clustering dalam bentuk tabel.

Listing 3.37: Main method

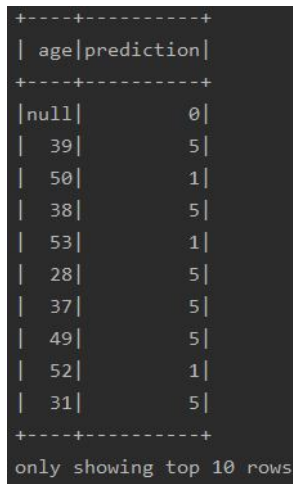
```
// KMeans with 8 clusters
val kmeans = new KMeans()
    .setK(8)
    .setFeaturesCol("features")
    .setPredictionCol("prediction")

val kmeansModel = kmeans.fit(result_df)
```

```
kmeansModel.clusterCenters.foreach(println)

// Predict model
val predictDf = kmeansModel.transform(result_df)
predictDf.show(10)
```

Hasil dari pemodelan K-Means adalah prediksi jenis cluster berdasarkan sifat dari masing-masing data. Umumnya pada hasil pemodelan K-Means, data-data yang memiliki perbedaan nilai atribut terkecil akan menjadi kelompok cluster yang sama. Pada Gambar ??, diketahui bahwa sebuah data yang bernilai (age = 39) memiliki kelompok cluster yang sama (prediction = 5) dengan data lain yang bernilai (age = 38).



```
+----+-----+
| age|prediction|
+----+-----+
| null|      0|
| 39|      5|
| 50|      1|
| 38|      5|
| 53|      1|
| 28|      5|
| 37|      5|
| 49|      5|
| 52|      1|
| 31|      5|
+----+-----+
only showing top 10 rows
```

Gambar 3.14: Hasil Naive Bayes

3.4 Gambaran Umum Perangkat Lunak

Penelitian ini menghasilkan dua jenis perangkat lunak dengan tujuan yang berbeda satu sama lain, untuk menyelesaikan permasalahan penerapan algoritma Greedy K-Member Clustering pada lingkungan big data. Berikut adalah deskripsi perangkat lunak yang akan dibuat:

1. Perangkat lunak dapat mengimplementasikan algoritma K-Anonymity. Algoritma K-Anonymity yang dimaksud adalah algoritma Greedy K-Member Clustering. Masukan dari perangkat lunak ini adalah dataset Adults dalam format file CSV, tabel atribut quasi-identifikasi (QID), dan parameter dari algoritma Greedy K-Member Clustering yaitu nilai k dan objek Domain Generalization Hierarchy (DGH). Keluaran dari perangkat lunak ini adalah hasil anonimisasi dari dataset Adults yang disimpan dalam format file CSV.
2. Perangkat lunak dapat membandingkan hasil anonimisasi algoritma K-Anonymity melalui metode data mining yang telah disediakan. Metode data mining yang akan disediakan adalah klasifikasi dengan pemodelan Naive Bayes dan pengelompokan/clustering dengan pemodelan K-Means. Masukan dari perangkat lunak ini adalah dataset Adults dalam format file CSV, baik dataset pernah dilakukan proses anonimisasi maupun dataset asli. Untuk pemodelan K-Means membutuhkan parameter tambahan seperti nilai k dan jenis atribut yang akan diprediksi nilainya. Sedangkan untuk pemodelan Naive Bayes membutuhkan parameter tambahan seperti persentase antara training dan testing data, jenis atribut yang akan diprediksi nilainya. Keluaran dari perangkat lunak ini untuk pemodelan Naive Bayes dan K-Means memiliki hasil yang sama, yaitu jenis nilai dari atribut yang telah dipilih dan jenis clusternya.

3.4.1 Diagram Aktifitas

Penelitian ini memiliki dua jenis diagram aktivitas, yaitu diagram aktivitas untuk perangkat lunak anonimisasi data dan diagram aktivitas untuk perangkat lunak analisis data. Tujuan dari membuat dua jenis perangkat lunak antara lain untuk memisahkan perangkat lunak dari fungsionalitas yang berbeda. Fungsionalitas tersebut antara lain melakukan proses anonimisasi data pada dataset dan membandingkan hasil antara dataset asli dengan dataset yang telah dilakukan proses anonimisasi untuk mencari tahu seberapa baik kinerja algoritma Greedy K-Member Clustering untuk mendapatkan hasil yang informatif.

Perangkat Lunak Anonimisasi Data

Perangkat lunak ini bertujuan untuk melakukan proses anonimisasi pada dataset Adults menggunakan algoritma K-Anonymity. Diagram aktifitas dapat dilihat pada Listing 3.3, berikut adalah tahapan yang terjadi pada perangkat lunak saat melakukan proses anonimisasi data:

1. Pengguna memberi masukan berupa data input CSV, lalu memilih beberapa atribut quasi-identifier pada data input tersebut, dan terakhir memberikan parameter masukan untuk algoritma Greedy K-Member Cluster seperti nilai k dan parameter masukan untuk objek DGH agar nilai quasi-identifier dapat dilakukan anonimisasi.
2. Perangkat lunak menampilkan data input pengguna pada tabel. Karena data input yang dipakai akan berukuran sangat besar maka data input hanya ditampilkan sebagian pada GUI. Tabel yang dibuat akan memiliki jumlah kolom yang sama dengan jumlah atribut dari dataset Adult. Atribut yang tidak memiliki nilai akan diisi dengan nilai null.
3. Pengguna meninjau kembali apakah nilai yang ada dalam tabel sudah sesuai dengan jenis kolomnya. Selanjutnya pengguna akan meninjau kembali apakah jumlah kolom yang ditampilkan pada tabel sudah sesuai dengan jumlah atribut pada dataset Adult.
4. Pengguna memberi masukan jumlah nilai k yang diinginkan dengan rentang $2 < k < n$. Nilai k merepresentasikan jumlah anggota maksimal pada sebuah cluster. Nilai k akan menjadi parameter masukan untuk menentukan pengelompokan data berdasarkan algoritma Greedy K-Member Clustering.
5. Perangkat lunak menerima parameter nilai k dari pengguna, selanjutnya perangkat lunak akan melakukan proses anonimisasi data dengan bantuan Spark.
6. Perangkat lunak mengembalikan isi log eksekusi Spark kepada pengguna sehingga pengguna dapat mengetahui apabila terjadi error/bug saat dilakukan proses anonimisasi data
7. Perangkat lunak menampilkan hasil anonimisasi data pada baris data yang hanya mengalami perubahan. Selain itu hasil anonimisasi data juga akan disimpan dalam format file CSV agar pengguna dapat membandingkan nilainya dengan data yang asli.
8. Perangkat lunak mengembalikan nilai information loss pada masing-masing cluster yang terbentuk dalam bentuk tabel. Nilai information loss dipakai untuk mencari nilai k yang tepat agar nilai information loss yang didapat seminimal mungkin
9. Pengguna dapat meninjau kembali hasil anonimisasi data yang tersimpan dalam format CSV dengan data yang asli. Caranya adalah membandingkan nilai pada baris data yang mengalami perubahan dengan nilai pada baris data asli.
10. Pengguna dapat mengulangi eksperimen untuk mencari nilai k yang tepat agar menghasilkan nilai information loss yang seminimal mungkin dari metode anonimisasi.

Perangkat Lunak Analisis Data

Perangkat lunak ini bertujuan untuk mencari perbandingan hasil sebelum dan setelah data dilakukan proses anonimisasi dengan metode data mining. Diagram aktifitas dapat dilihat pada Listing 3.3, berikut adalah tahapan yang terjadi pada perangkat lunak saat melakukan pemodelan data mining:

1. Pengguna memberi dua jenis masukan berupa data input CSV berdasarkan data asli dan data hasil anonimisasi, lalu dipilih beberapa atribut quasi-identifier pada data tersebut
2. Perangkat lunak menampilkan tabel dari data asli dan tabel dari data hasil anonimisasi. Karena data input yang dipakai berukuran sangat besar maka data input hanya ditampilkan sebagian saja pada GUI.
3. Pengguna meninjau kembali apakah nilai yang ada dalam tabel sudah sesuai dengan jenis kolomnya. Selanjutnya pengguna akan meninjau kembali apakah jumlah kolom yang ditampilkan pada tabel sudah sesuai dengan jumlah atribut pada hasil pengelompokan data.
4. Pengguna memilih jenis pemodelan data mining yang tersedia yaitu Naive Bayes dan K-Means.
5. Pengguna mengisi parameter pada pemodelan yang dipilih. Contohnya pada pemodelan K-Means, pengguna perlu mengisi nilai k dan jenis atribut yang dipilih. Sedangkan pada pemodelan Naive Bayes, pengguna perlu mengisi persentase antara training, testing data dan jenis atribut yang dipilih.
6. Perangkat lunak menerima parameter dari pengguna, selanjutnya perangkat lunak akan melakukan proses pelatihan data pada Spark berdasarkan jenis pemodelan yang dipilih.
7. Perangkat lunak mengembalikan isi log eksekusi Spark kepada pengguna sehingga pengguna dapat mengetahui apabila terjadi error/bug saat dilakukan proses pelatihan data
8. Perangkat lunak akan menampilkan sebagian dari hasil prediksi cluster untuk masing-masing data pada tabel. Selain itu perangkat lunak juga akan menyimpan hasil keseluruhan prediksi kelompok data pada format file CSV.
9. Pengguna melakukan analisis lebih lanjut terkait pengelompokan dan klasifikasi kelompok data yang sudah terbentuk dari proses pemodelan data mining.

3.4.2 Diagram Kelas

Diagram kelas bertujuan untuk menggambarkan keterhubungan antar kelas. Pada penelitian ini digambarkan diagram kelas untuk perangkat lunak anonimisasi data. Karena perangkat lunak analisis data hanya memiliki satu kelas saja, maka keterhubungan antar kelas tidak perlu digambarkan dalam diagram kelas. Gambar 1.1 menggambarkan keterhubungan antar kelas pada perangkat lunak anonimisasi data. Berikut adalah penjelasan lengkap mengenai deskripsi kelas dan method pada perangkat lunak anonimisasi data:

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```

1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf

```

Listing A.2: MyCode.java

```

1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }

```


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