BAB 2

LANDASAN TEORI

Pada bab ini, akan dijelaskan konsep mengenai privasi, teknik *data mining*, *privacy-preserving data mining*, *k-anonymity*, algoritma *greedy k-member clustering*, metrik *distance* dan *information loss*, teknologi *big data*, pemrograman scala, dan format penyimpanan data sebagai landasan penelitian.

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 lansung atau tidak langsung. PII menjelaskan bahwa identifikasi entitas secara langsung dapat dilakukan menggunakan atribut sensitif. Sedangkan identifikasi entitas secara tidak langsung dapat dilakukan menggunakan penggabungan beberapa atribut non-sensitif. PII adalah atribut yang biasanya terjadi pelanggaran data dan pencurian identitas. Jika data perusahaan atau organisasi terungkap, maka sangat mungkin data pribadi seseorang akan terungkap. Informasi yang diketahui dapat dijual dan digunakan untuk melakukan pencurian identitas, menempatkan korban dalam risiko.

Berikut adalah contoh informasi yang bersifat sensitif menurut standar PII:

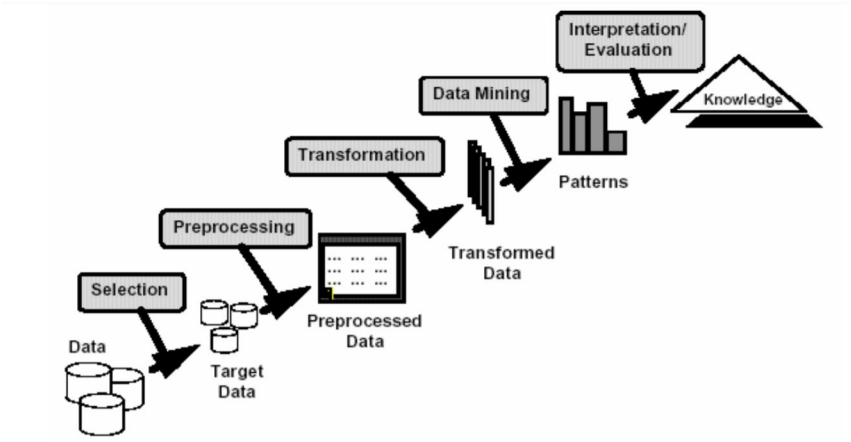
- Identitas diri
  Nama lengkap, tempat tanggal lahir, alamat rumah, alamat email.
- Nomor identitas diri
  NIK, nomor passport, nomor SIM, nomor wajib pajak, nomor rekening, nomor telepon, dan nomor kartu kredit.
- Data biometrik
  Pemindaian retina, jenis suara, dan geometri wajah.

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

- Rekaman medis
- Riwayat pendidikan
- Riwayat pekerjaan
- Informasi finasial
- 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: nilai true/false dan 0/1.
- *Nominal*: tipe data alphabet/numerik yang memiliki lebih dari 2 kemungkinan nilai.
  Contoh: warna kuning, hijau, hitam, merah.

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: *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 *Logistic Regression*

Contoh: *Logistic Regression*
Gambar 2.2 adalah 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 klasifikasi 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. Pemodelan 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:

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

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

$$MAP(H) = max(P(H|D)) \tag{2.2}$$

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.

Tabel 2.1 diberikan untuk menggambarkan kondisi cuaca saat bermain golf. Masing-masing data dikategorikan berdasarkan nilai atribut *PlayGolf*, yaitu cocok (*Yes*) atau tidak cocok (*No*).

Tabel 2.1: Contoh Kasus *PlayGolf*

Outlook	Temperature	Humidity	Windy	PlayGolf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

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 pada contoh kasus ini terdiri dari atribut *Outlook, Temperature, Humidity, dan Windy.*

- Vektor respon
  Vektor respon adalah nilai prediksi dalam bentuk label kelas untuk setiap baris data. Vektor respon pada contoh kasus ini diwakili oleh 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:

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

   Di mana y adalah label kelas dan X adalah vektor fitur (dengan ukuran n), dinyatakan melalui persamaan berikut:

$$X = (x_1, x_2, x_3, \ldots, x_n) \tag{2.4}$$

   Contoh: $X = (Rainy, Hot, High, False), y = No$

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

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

2. Menghitung probabilitas nilai dari sebuah atribut.
   Sebagai contoh, nilai yang dimiliki atribut *Outlook* adalah *{Sunny,Overcast,Rainy}*. Tabel 2.2 berfungsi untuk mencatat frekuensi dan menghitung probabilitas nilai dari atribut *Outlook*.

Tabel 2.2: Tabel Probabilitas pada Atribut *Outlook*

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

   Berikut adalah contoh perhitungan $P(No)$ untuk nilai *Sunny* pada atribut *Outlook*

$$P(No) = \frac{frekuensi(Sunny \cap No)}{frekuensi(No)} = \frac{2}{5}$$

   Berikut adalah contoh perhitungan $P(Yes)$ untuk nilai *Sunny* pada atribut *Outlook*

$$P(Yes) = \frac{frekuensi(Sunny \cap Yes)}{frekuensi(Yes)} = \frac{3}{9}$$

   Perhitungan probabilitas $P(Yes)$ dan $P(No)$ berlaku untuk nilai lainnya pada atribut *Outcast* yaitu *{Overcast,Rainy}*, sehingga hasil akhirnya dapat dilihat pada Tabel 2.2.

3. Membuat tabel probabilitas untuk atribut lainnya *{Temperature,Humidity, Windy}* dengan cara yang sama seperti langkah 2. Hasilnya akan menjadi Tabel 2.3, 2.4, 2.5

Tabel 2.3: Tabel Probabilitas dari Atribut 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%

Tabel 2.4: Tabel Probabilitas dari Atribut Humidity

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

Tabel 2.5: Tabel Probabilitas dari Atribut 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

4. Menghitung probabilitas bersyarat terhadap data baru yang diberikan.
Contoh: *today = (Sunny, Hot, Normal, NoWind)*

$$P(Yes|today) = \frac{P(Sunny|Yes)P(Hot|Yes)P(Normal|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$
$$= \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{5}{14} = 0.0068$$

$$P(No|today) = \frac{P(Sunny|No)P(Hot|No)P(Normal|No)P(NoWind|No)P(No)}{P(today)}$$
$$= \frac{2}{9} \cdot \frac{2}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} \cdot \frac{9}{14} = 0.0068$$

5. Melakukan normalisasi terhadap probabilitas besyarat.

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

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

Ketika probabilitas tersebut dijumlahkan, maka hasilnya akan menjadi 1.

$$P(Yes|today) + P(No|today) = 1 \tag{2.6}$$

6. Mencari probabilitas tertinggi.

   Berdasarkan pernyataan berikut:

$$P(Yes|today) > P(No|today) \tag{2.7}$$

   Dapat disimpulkan bahwa data dengan nilai *(Sunny, Hot, Normal, NoWind)* dapat diklasifikasikan terhadap atribut $PlayGolf$ dengan nilai label kelas *Yes*.

2.2.3 *Clustering*

Clustering merupakan proses mengelompokkan satu data ke dalam himpunan data yang disebut *cluster*. Objek di dalam *cluster* memiliki kemiripan karakteristik satu sama lain yang berbeda dengan *cluster* lainnya. *Clustering* sangat berguna untuk menemukan kelompok data yang tidak dikenal. *Clustering* sering disebut juga sebagai *outlier detection*.

Berikut adalah tahapan *clustering* secara umum:

1. Perhitungan *distance*: proses untuk mengukur kesamaan antar objek dengan cara menghitung *distance* antar 2 titik. Salah satu contoh metode pengukuran *distance* yang umum digunakan adalah *Euclidean distance*. Metode ini terdiri atas variabel p_i, menyatakan kordinat titik data dan variabel C_i, menyatakan kordinat titik *centroid* sebuah *cluster*.

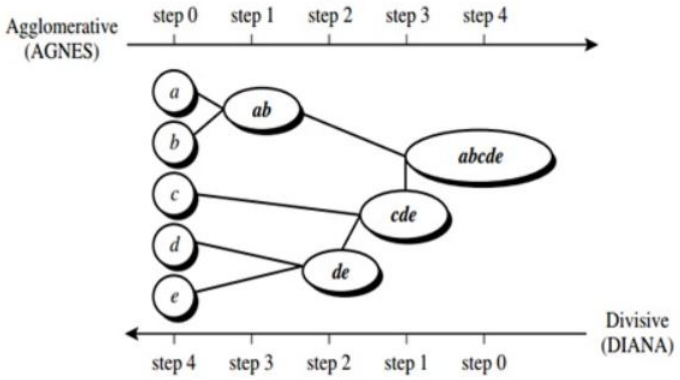
   Berikut adalah persamaan untuk menghitung *Euclidean distance*:

$$EuclidDist(p_i, C_i) = \sqrt{(p_1 - C_1)^2 + (p_2 - C_2)^2 + \ldots + (p_n - C_n)^2} \tag{2.8}$$

2. Pengelompokan data: proses pencarian anggota *cluster* dengan menghitung *distance* minimum antara masing-masing titik data dengan titik *centroid cluster* tersebut.

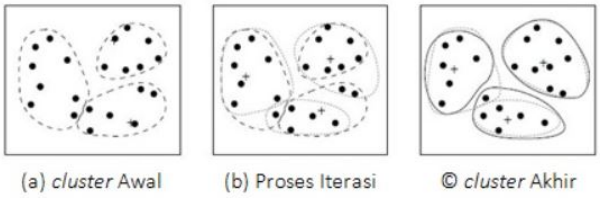
Berikut adalah kategori pemodelan *clustering*:

- *Hierarchical clustering*: pengelompokan data pada sebuah hirarki dengan cara menggabung dua kelompok data terdekat maupun membagi data ke dalam cluster tertentu. Contoh pemodelannya adalah *single linkage, complete linkage, average linkage, average group linkage*.



Gambar 2.3: Contoh *Hierarchical Clustering*

- *Partitional clustering*: pengelompokan data ke dalam sejumlah *cluster* tanpa adanya struktur hierarki. Pada metode ini, setiap *cluster* memiliki titik pusat cluster (centroid) dengan tujuan untuk meminimumkan (*dissimilarity distance*) seluruh data ke *centroid cluster* masing-masing. Contoh pemodelannya adalah *k-means, fuzzy k-means,* dan *mixture modeling.*



Gambar 2.4: Contoh *Partitional Clustering*

Berikut adalah contoh pemodelan *clustering* yang umum digunakan:

- *Agglomerative*
- *K-Means*

2.2.4 *K-Means*

K-means merupakan metode clustering yang paling sederhana dan umum. *K-means* merupakan salah satu algoritma *clustering* dengan metode *partitional clustering* menggunakan titik *centroid* sebagai pusat kelompok data. *K-means* memerlukan tiga jenis paramater yaitu menentukan jumlah kelompok data (k), inisialisasi titik *centroid* awal, dan mengetahui *distance* antar titik.

Tabel 2.6 diberikan untuk mengelompokan mata pelajaran yang sejenis berdasarkan data skor yang diperoleh dari individu A dan B:

Tabel 2.6: Tabel Dataset Mata Pelajaran

Subject	Person A	Person 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

Secara singkat, langkah kerja algoritma *k-means* dapat dijelaskan sebagai berikut:

1. Inisialisasi k dan titik *centroid* awal.
   Pada contoh ini, jumlah *cluster* yang ingin dibentuk adalah dua kelompok data atau $k = 2$ dengan inisialisasi titik *centroid* awal adalah $(1.0, 1.0)$ dan $(5.0, 7.0)$. Kedua titik ini dipilih secara acak. Data ini direpresentasikan pada Tabel 2.7 seperti berikut.

Tabel 2.7: Hasil Pengelompokan Awal

	Individual	Centroid
Cluster 1	1	(1.0, 1.0)
Cluster 2	4	(5.0, 7.0)

2. Melakukan perhitungan *distance*.
   Distance digunakan untuk mencari jarak antara sebuah titik data dengan titik *centroid* dari *cluster* tertentu. Sebagai contoh, *distance* antara data ke-1 $(1.5, 2.0)$ dengan titik *centroid* dari *Cluster 1* $(1.0, 1.0)$ dihitung menggunakan *Euclidean distance* sebagai berikut.

$$\begin{aligned} EuclidDist(p_i, C_i) &= \sqrt{(p_1 - C_1)^2 + (p_2 - C_2)^2 + \ldots + (p_n - C_n)^2} \\ &= \sqrt{(1.5 - 1.0)^2 + (2.0 - 1.0)^2} = 1.1180 \end{aligned}$$

3. Menempatkan setiap titik data ke titik *centroid* terdekat.
   Titik data akan dikelompokan ke *centroid* terdekat dengan memilih nilai Euclidian *distance* paling kecil. Sebagai contoh, Tabel 5.1 pada *Step 2* menyatakan data ke-1 lebih dekat dengan data ke-2, karena memiliki nilai Euclidean distance paling kecil. Karena itu, data ke-2 bergabung pada titik *centroid* data ke-1. Hal ini berlaku pada setiap langkah.

Tabel 2.8: Mencari Centroid Kelompok

	Cluster 1		Cluster 2	
Step	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)

Setelah pengelompokan data selesai dilakukan, selanjutnya perlu menghitung *Mean Vector (centroid)* sebagai titik *centroid* baru sebuah *cluster*. Sebagai contoh, *Mean Vector* pada *Cluster 1* untuk *Step 2* dapat dihitung sebagai berikut.

$$\begin{aligned} MeanVector(PersonA) &= \frac{1.0 + 1.5}{2} = 1.2 \\ MeanVector(PersonB) &= \frac{1.0 + 2.0}{2} = 1.5 \end{aligned}$$

4. Menetapkan kelompok data baru pada masing-masing *cluster*.
   Iterasi paling terakhir di langkah sebelumnya akan dijadikan sebagai anggota dari *cluster* baru. Tabel 2.9 menunjukan kelompok data sementara yang terbentuk pada masing-masing *cluster*. *Cluster 1* terdiri dari anggota data $\{1, 2, 3\}$, sedangkan *Cluster 2* terdiri dari anggota data $\{4, 5, 6, 7\}$. Kelompok data yang terbentuk saat ini masih sementara dan dapat berubah-ubah.

Tabel 2.9: Hasil Cluster Baru

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

5. Mencari titik *centroid* baru untuk setiap *cluster*.
   Belum dipastikan bahwa setiap individu telah dialokasikan dengan tepat. Oleh karena itu, langkah 1 sampai 4 perlu kembali diulang dengan menghitung kembali *Euclidean distance*. Tabel 2.10 merupakan hasil perhitungan *distance* untuk setiap titik data.

Tabel 2.10: Euclidean Distance Cluster 1, Cluster 2

Individual	Distance centroid of Cluster 1	Distance 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

6. Menetapkan kelompok data akhir.
   Apabila *cluster* tidak mengalami perubahan secara signifikan pada anggotanya selama periode iterasi tertentu, maka *cluster* yang terbentuk sudah sesuai. Tabel 2.11 menunjukan anggota dari *cluster* yang sudah tetap, sehingga proses *k-means* dapat dihentikan.

Tabel 2.11: Hasil Pengelompokan Akhir

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

2.3 Tahapan Evaluasi *Data Mining*

Setelah melakukan implementasi model *data mining*, langkah selanjutnya adalah mengevaluasi hasil dari model *data mining* yang telah dibuat. Tahapan evaluasi sangat penting untuk mengukur seberapa tinggi tingkat akurasi model yang dipilih untuk menyelesaikan sebuah permasalahan data. Metode evaluasi untuk setiap model *data mining* berbeda satu sama lain. Oleh karena itu, perlu untuk mendefinisikan perhitungan evaluasi untuk masing-masing model.

2.3.1 Menghitung Tingkat Akurasi untuk Model Klasifikasi

Hasil pemodelan klasifikasi dapat dievaluasi menggunakan perhitungan tingkat akurasi. Semakin tinggi tingkat akurasi yang diperoleh, maka hasil pemodelan klasifikasi menjadi semakin baik. Akurasi dihitung dari rasio jumlah prediksi yang benar dengan jumlah total sampel input.

Berikut adalah persamaan untuk menghitung tingkat akurasi:

$$Tingkat\ Akurasi = \frac{jumlah\ prediksi\ yang\ benar}{total\ sampel\ input} \times 100\% \tag{2.9}$$

Kisaran skor untuk tingkat akurasi adalah $[\,0\%, 100\%\,]$.

2.3.2 Menghitung Koefisien *Silhouette* untuk Model *Clustering*

Hasil pemodelan *clustering* dapat dievaluasi menggunakan perhitungan koefisien *silhouette*. Koefisien *silhouette* bertujuan untuk menghitung seberapa dekat sebuah objek dengan *intra-cluster* dan mengukur seberapa jauh objek yang sama dengan *cluster* terdekat lainnya. Semakin tinggi nilai koefisien *silhouette*, maka hasil pengelompokannya akan semakin baik. Koefisien *silhouette* dihitung berdasarkan rata-rata *distance* antara setiap titik data dengan titik *centroid intra-cluster* dan rata-rata *distance* antara setiap titik data dengan titik *centroid cluster* lainnya.

Kisaran skor koefisien *silhouette* adalah $[-1, 1]$, berikut adalah analisisnya:

- Skor 1, artinya sampel berada jauh dari *cluster* tetangganya.
- Skor 0, artinya bahwa sampel sangat dekat dengan *cluster* tetangganya
- Skor -1, artinya sampel telah dikelompokan pada *cluster* yang salah.

Berikut adalah persamaan untuk menghitung koefisien *silhouette*:

$$Silhouette\ Score = \frac{(p - q)}{max(p, q)} \tag{2.10}$$

Keterangan:

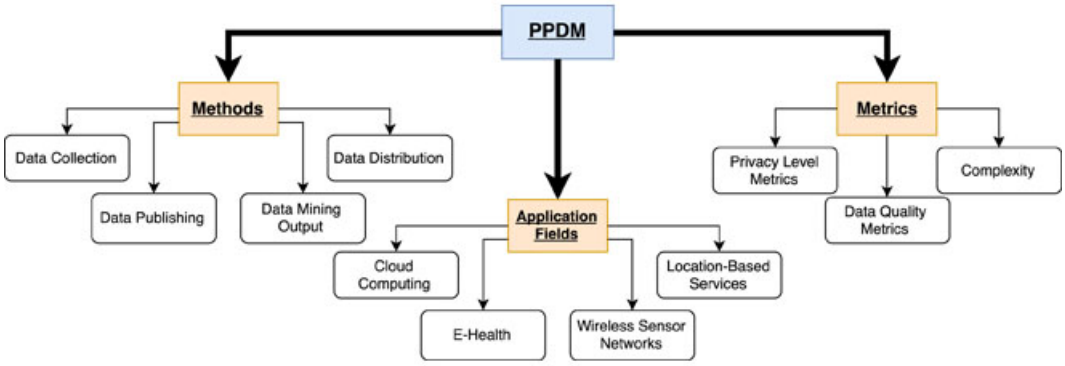
- p = rata-rata *distance* setiap titik *intra-cluster* terhadap titik *centroid* pada *cluster* terdekat.
- q = rata-rata *distance* setiap titik *intra-cluster* terhadap titik *centroid intra-cluster*.
- $max(p, q)$ = nilai maksimum dari p dan q.

2.4 *Privacy-Preserving Data Mining* (PPDM)

Di era informasi saat ini, sistem informasi terus menghasilkan banyak informasi. Namun, banyak dari informasi yang dikumpulkan bersifat sensitif seperti data pribadi yang dapat menimbulkan masalah privasi. Untuk melindungi kebocoran informasi, metode preservasi privasi telah dikembangkan untuk melindungi entitas pemilik data dengan memodifikasi data asli menjadi data yang disamarkan atau data anonim. Akan tetapi metode ini memiliki kekurangan karena secara tidak langsung mengubah isi data, sehingga informasi yang didapat menjadi berkurang. Hal ini mengakibatkan ekstraksi informasi melalui proses *data mining* menjadi tidak akurat.

Privacy Preserving Data Mining (PPDM) adalah metodologi yang dirancang khusus untuk menjamin tingkat privasi pada level tertentu, sekaligus memaksimalkan utilitas data saat dilakukan proses data mining. PPDM mencakup semua teknik yang dapat digunakan untuk mengekstrak pengetahuan dari data dan secara bersama menjaga privasi dari data tersebut. Menurut IEEE, metodologi ini dapat dibagi menjadi beberapa bagian yaitu metode perlindungan privasi dan metrik perlindungan privasi yang akan dijelaskan pada bagian selanjutnya.

*G*ambar 2.5 menjelaskan garis besar pokok pembahasan dari PPDM yaitu metode, metrik, dan bidang penerapannya. Metode PPDM dapat diterapkan fase siklus data mulai dari pengumpulan data, publikasi data, hasil *data mining*, sampai kepada distribusi data. Selain itu, PPDM juga memiliki metrik untuk mengevaluasi dan membandingkan teknik-teknik yang digunakan agar mencapai tingkat privasi dan kualitas data tertentu. PPDM umumnya diaplikasikan pada bidang *cloud computing, e-health, wireless sensor,* dan *location-based service.*



Gambar 2.5: *Privacy Preserving Data Mining* (PPDM)

2.4.1 Metode pada PPDM

Beberapa metode PPDM telah diusulkan untuk menjamin perlindungan privasi pada masing-masing fase siklus hidup sebuah data di mana fase siklus tersebut sering terjadi pelanggaran privasi saat pengumpulan data, penerbitan data, distribusi data dan terhadap hasil *data mining*.

Berikut adalah metode perlindungan privasi pada PPDM:

- Pengumpulan data
  Pada fase ini, entitas yang bertugas mengumpulkan data tidak dapat dipercaya karena dapat mengumpulkan dan menggunakan data privasi individu secara tidak benar. Metode yang dapat digunakan untuk melindungi privasi saat pengumpulan data adalah *randomisation*.

- Publikasi data
  Pada fase ini, entitas ingin mempublikasikan datanya untuk keperluan penelitian atau analisis. Akan tetapi, ada beberapa individu yang mencoba melakukan tindakan deanonimisasi pada data-data privasi milik seseorang untuk tindakan kejahatan. Metode yang dapat digunakan untuk melindungi privasi saat publikasi data adalah *k-anonymity*.

- Hasil data mining
  Hasil data mining biasanya dibuat mudah diakses melalui aplikasi atau *interface*. Pada fase ini, entitas dapat melakukan kueri pada sistem untuk mencari informasi sensitif seseorang. Metode yang dapat digunakan untuk melindungi privasi pada hasil data mining adalah *association rule hiding, downgrading classifier effectiveness,* dan *query auditing and inference control*.

- Distribusi data
  Pada fase ini, beberapa entitas akan mencari wawasan umum dalam bentuk data statistik tanpa mengungkap informasi entitas lain. Akan tetapi ada beberapa entitas yang sengaja untuk menemukan celah keamanan privasi melalui pencocokan data dengan informasi umum yang diberikan secara publik. Hal ini dapat dicegah dengan memberlakukan *multiparty protocol* pada sistem distribusi data.

2.4.2 Metrik pada PPDM

Karena privasi tidak memiliki standar definisi yang jelas, maka diperlukan metrik untuk mengukur keamanan privasi. Melalui metode PPDM, beberapa metrik perlindungan privasi telah diusulkan. Metrik PPDM terbagi menjadi dua kategori utama yaitu privasi data dan kualitas data.

Berikut adalah metrik perlindungan privasi pada PPDM:

- Tingkatan Privasi
  Tingkatan privasi memberikan gambaran seberapa aman data diolah agar terhindar dari kasus pelanggaran privasi . Metrik ini dapat ditemukan pada model *k-anonymity*. Model k-anonymity berpengaruh terhadap tingkat privasi, dimana nilai k dapat diubah sesuai keinginan dalam menentukan tingkat keamanan data.

- Kualitas Data
  Kualitas data dinilai dari tiga parameter penting. *Accuracy*, untuk mengukur seberapa dekat nilai data yang disamarkan dengan data asli. *Completeness*, untuk mengevaluasi banyaknya data yang hilang. *Consistency*, untuk mengukur hilangnya korelasi data pada data yang disamarkan. Pada pemodelan *k-anonymity*, metrik yang dapat diukur untuk menyatakan kualitas data adalah *Information Loss* (IL).

2.4.3 Model Serangan pada PPDM

Menurut Dalenius (1977), perlindungan privasi tidak memberikan kesempatan bagi orang lain untuk mendapatkan informasi sensitif 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 dipublikasi. Serangan ini dikenal dengan istilah *linkage attack*.

Berikut adalah jenis serangan dari *linkage attack*:

- *Record Linkage*
  Record Linkage mengacu serangan mencocokan antara rekord korban yang dirilis dengan record korban yang telah diketahui untuk menebak atribut sensitif milik korban. Jika record tersebut cocok dengan record lain, artinya kedua data saling berkaitan. Serangan ini dapat dicegah dengan penggunaan model *k-anonymity*.

- *Attribute Linkage*
  Attribute Linkage mengacu serangan untuk mendapatkan beberapa informasi nilai atribut sensitif korban dengan mencari keterhubungan atribut. Tipe serangan ini mencari keterhubungan nilai atribut non sensitif untuk menebak atribut sensitif milik korban. Serangan ini dapat dicegah dengan penggunaan model *l-diversity*.

2.5 Anonimisasi

Anonimisasi data mengacu pada penghapusan atau enkripsi informasi pribadi atau data sensitif. Karena bisnis, pemerintah, sistem perawatan kesehatan, dan organisasi lain semakin banyak menyimpan informasi individu di server lokal atau cloud, anonimisasi data sangat penting untuk menjaga integritas data dan mencegah pelanggaran keamanan.Di sektor kesehatan dan keuangan, data dapat menjadi hal yang sangat sensitif. Data pasien atau entitas lain harus disembunyikan untuk memenuhi persyaratan perlindungan privasi.

Anonimisasi berupaya melindungi data pribadi atau atribut sensitif dengan menghapus atau mengenkripsi informasi yang dapat diidentifikasi secara pribadi dari database. Anonimisasi data dilakukan dengan tujuan melindungi aktivitas pribadi individu atau perusahaan sambil menjaga integritas data yang dikumpulkan dan dibagikan. Teknik anonimisasi data juga dikenal sebagai *"obfuscation data"*, *"data masking"*. Sedangkan *data de-anonymization* merupakan teknik yang digunakan pada proses *data mining* dengan tujuan untuk mengidentifikasi kembali informasi yang hilang pada data yang dienkripsi atau dianonimisasi.

2.6 *K-Anonymity*

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

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 nilai yang rendah. Beberapa algoritma telah diuji pada pemodelan *k-anonymity.*

Berikut algoritma yang dapat diterapkan pada 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 data pada baris yang terakhir tanpa memperhatikan pengelompokan yang dihasilkan pada proses sebelumnya sehingga menyebabkan distribusi kelompok data pada beberapa bagian menjadi kurang tepat.
- Untuk menghindari kekurangan pada algoritma *k-means* dan algoritma *k-member*, maka kedua pendekatan ini digabung menjadi algoritma baru yaitu *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. Hasil akhir dari algoritma *Greedy k-member clustering* adalah data-data yang sejenis telah dikelompokan pada kelompok data yang sama. Untuk melakukan anonimisasi, digunakan konsep *Hierarchy Based Generalization. K-anonymity* menyamarkan data pada nilai quasi-identifier yang sama.

Berikut adalah atribut dari pemodelan *k-anonymity*, yaitu:

- *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.
- *Sensitive Attribute* (SA) adalah atribut yang menyangkut informasi sensitif seseorang, biasanya nilai atribut ini akan dirahasiakan saat distribusi data.
- *Non-sensitive Attribute* (NSA) adalah atribut yang tidak menyangkut informasi sensitif seseorang, biasanya nilai atribut ini langsung ditampilkan saat distribusi data.

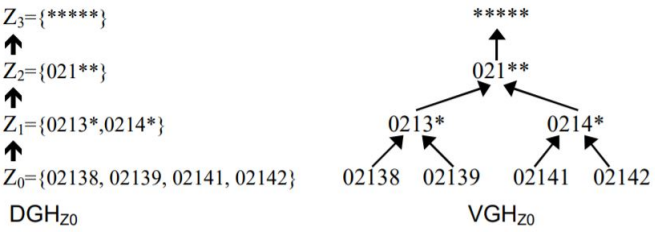
2.7 *Hierarchy Based Generalization*

Hierarchy-based generalization adalah tahapan anonimisasi pada pemodelan k-anonymity dimana *quasi-identifier* yang sama dikelompokan ke dalam cluster 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 berdasarkan kategori tertentu.

Full-domain generalization dapat digunakan oleh model *k-anonymity* untuk menentukan generalisasi dan supresi dari sebuah nilai. *Full-domain generalization* menetapkan bahwa proses anonimisasi data dapat dilakukan dengan metode generalisasi apabila atribut *quasi-identifier* telah dipilih sejak awal. Sebagai contoh, dipilih atribut quasi-identifier sebagai berikut $QI = \{A_1, \ldots, A_n\}$, dimana A adalah atribut pada tabel dataset. 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. DGH merupakan konsep sederhana dari penggantian nilai berdasarkan nilai yang kurang spesifik menjadi nilai lebih umum terhadap nilai aslinya. Tidak terdapat aturan khusus untuk memodelkan hierarki DGH. Beberapa nilai atribut harus digeneralisasi menggunakan DGH untuk mengakhiri proses anonimisasi data.

Pada Gambar 3.2, nilai ZIP {02138, 02139} dapat digeneralisasi menjadi {0213∗} dengan menghilangkan digit paling kanan. Nilai yang telah digeneralisasi akan memiliki lingkup nilai yang lebih besar. Sebagai contoh, nilai ZIP adalah $\{02138, 02139\}$ berada pada domain dasar yaitu Z_0. Untuk mencapai perlindungan data pada *k-anonymity*, maka kode ZIP yang sebelumnya unik harus diubah menjadi bentuk umum. Sebuah nilai dapat digeneralisasi jika memiliki domain yang lebih umum. Sedangkan domain yang kurang spesifik digunakan untuk mendeskripsikan garis besar nilai ZIP. Sebagai contoh dari domain kurang spesifik adalah Z_1, di mana digit terakhir diganti dengan simbol (*). Berikut contoh pemetaan dari $Z0$ ke $Z1$ seperti berikut $02139 \rightarrow 0213*$.

Representasi generalisasi dapat diperluas, agar metode supresi dapat diterapkan dalam hirarki generalisasi. Elemen baru dengan nilai supresi maksimal $(*****)$ memiliki posisi lebih tinggi dibandingkan dengan elemen generalisasi $(021**)$. Ketinggian setiap hierarki generalisasi akan bertambah seiring bertambahnya kategori elemen generalisasi. Setelah elemen mencapai nilai supresi maksimal yang dapat digeneralisasi $(*****)$, maka tidak akan ada lagi perubahan yang diperlukan untuk memasukkan elemen generalisasi yang baru. Gambar 3.2 dan 2.7 adalah contoh hierarki generalisasi DGH dan VGH pada atribut ZIP (kode pos) dan *Race* (warna kulit).



Gambar 2.6: DGH dan VGH pada Atribut ZIP

Gambar 2.7: DGH dan VGH pada Atribut Race

2.8 *Greedy K-Member Clustering*

Sebagian besar metode *k-anonymity* memakai metode generalisasi dan supresi sehingga data menderita kehilangan informasi yang signifikan. Masalah pengelompokan dipercaya dapat meminimalkan kehilangan informasi melalui implementasi algoritma *k-member clustering*. Akan tetapi algoritma tersebut memiliki kompleksitas eksponensial $O(2^n)$, sehingga perlu ditransformasi dengan algoritma *Greedy* dengan kompleksitas $O(nlogn)$. Algoritma *Greedy k-member clustering* bertujuan melakukan pengelompokan data ke masing-masing *cluster* dengan kompleksitas algoritma yang lebih baik dan dapat meminimalkan nilai informasi yang hilang saat dilakukan anonimisasi data.

Teorema 1. *Masalah pengambilan keputusan pada k-member clustering adalah NP-Hard. NP-Hard adalah suatu kelompok masalah dimana tidak ada algoritma yang dapat menemukan solusi optimal dengan kompleksitas lebih kecil dari polinomial.*

Bukti. Melalui percobaan Aggarwal et al dengan model *3-anonymity*, telah dibuktikan bahwa satu-satunya cara untuk melakukan anonimisasi atribut, yaitu dengan cara melakukan iterasi dari node paling atas (*root node*) ke node paling bawah (*leaf node*). □

Teorema 2. *N adalah total data dan k adalah parameter untuk anonimisasi. Setiap cluster 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 data. Algoritma ini menemukan *cluster* selama jumlah data yang tersisa sama dengan atau lebih besar dari k, setiap *cluster* berisi k data. Jika total data pada S kurang dari k, maka sisa data akan dikelompokan pada *cluster* yang sudah ada. Oleh karena itu, ukuran maksimum sebuah *cluster* adalah $2k - 1$. □

Teorema 3. *N adalah jumlah data dan k menjadi parameter anonimisasi 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 data 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). □

Beberapa hal penting terkait algoritma *Greedy k-means clustering*:

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

$$m = \left\lfloor \frac{n}{k} \right\rfloor - 1 \tag{2.11}$$

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 perbedaaan 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 ∈ S such that IL(c ∪ {r}) is minimal
4:
5:  n = |S|
6:  min = ∞
7:  best = null
8:  for i = 1 ... n do
9:      r = i-th record in S
10:     diff = IL(c ∪ {r}) − IL(c)
11:     if diff < min then
12:         min = diff
13:         best = r
14:     end if
15: end for
16: return best
```

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 *cluster* 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 ∈ C such that IL(c ∪ {r}) is minimal
4:
5:  n = |C|
6:  min = ∞
7:  best = null
8:  for i = 1 ... n do
9:      c = i-th cluster in C
10:     diff = IL(c ∪ {r}) − IL(c)
11:     if diff < min then
12:         min = diff
13:         best = c
14:     end if
15: end for
16: return best
```

Algoritma 2 menerima input himpunan data *cluster* 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 *cluster* dengan *information loss* pada data ke-i, lalu hasil penjumlahan tersebut dikurangi dengan *information loss* dari cluster. Output algoritma ini adalah data dengan nilai *cluster* 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 ≤ k then
6:     return S
7: end if
8:
9: result = φ
10: r = a randomly picked record from S
11: while |S| ≥ k do
12:     r = the furthest record from r
13:     S = S - {r}
14:     c = {r}
15:     while |c| < k do
16:         r = find_best_record(S,c)
17:         S = S - {r}
18:         c = c ∪ {r}
19:     end while
20:     result = result ∪ {c}
21: end while
22: while S ≠ 0 do
23:     r = a randomly picked record from S
24:     S = S - {r}
25:     c = find_best_cluster(result,r)
26:     c = c ∪ {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 *cluster* dengan *distance* terkecil dan fungsi *find_best_record* untuk mencari data dengan *distance* terkecil. Output dari algoritma ini adalah himpunan data dari berbagai jenis *cluster* dengan nilai *distance* terkecil.

2.9 Metrik *Distance* dan *Information Loss*

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* dan *information loss*. Secara umum, pengukuran metrik dilakukan dengan membandingkan seberapa baik akurasi hasil data yang telah dianonimisasi dengan akurasi pada dataset sesungguhnya.

2.9.1 *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-anonymity* menggunakan atribut numerik dan kategorikal, maka membutuhkan cara khusus untuk mengitung *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:

$$\delta_n(v_1, v_2) = \frac{|v_1 - v_2|}{|D|} \tag{2.12}$$

Distance Data Kategorikal

Distance data kategorikal direpresentasikan sebagai *taxonomy tree*. 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:

$$\delta_C(v_1, v_2) = \frac{H(\Lambda(v_i, v_j))}{H(T_D)} \tag{2.13}$$

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:

$$\Delta(r_1, r_2) = \sum_{i=1}^{m} \delta_N(r_1[N_i], r_2[N_i]) + \sum_{j=1}^{n} \delta_C(r_1[C_j], r_2[C_j]) \tag{2.14}$$

2.9.2 *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* $e = r_1, \ldots, r_k$ untuk *quasi-identifier* yang terdiri dari atribut numerik $N1, \ldots, Nm$ dan atribut kategorikal $C_1, \ldots, C_n$, 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 Ni, $\cup_{C_i}$ adalah sekumpulan nilai pada *cluster* e berdasarkan atribut C_i.

Information loss dihitung dengan rumus sebagai berikut:

$$IL(e) = |e| \cdot D(e) \tag{2.15}$$

$$D(e) = \sum_{i=1}^{m} \frac{(MAX_{N_i} - MIN_{N_i})}{|N_i|} + \sum_{j=1}^{n} \frac{H(\Lambda(\cup_{C_j}))}{H(T_{C_j})} \tag{2.16}$$

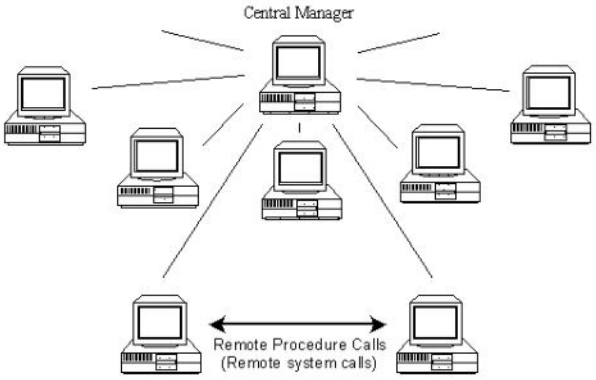
Total Information Loss dihitung dengan rumus sebagai berikut:

$$Total - IL(AT) = \sum_{e \in \varepsilon} IL(e) \tag{2.17}$$

Semakin besar *total information loss* yang dihasilkan maka informasi yang dihasilkan semakin kurang akurat. Oleh karena itu perlu dilakukan beberapa eksperimen terhadap penentuan nilai k pada algoritma *Greedy k-member clustering* agar dihasilkan *total information loss* seminimal mungkin sehingga hasil *clustering* dan klasifikasi dengan nilai akurasi yang tinggi.

2.10 Sistem Terdistribusi

Sistem terdistribusi adalah kumpulan komputer berjalan secara independen dan saling terhubung dan saling bekerja sama untuk mencapai satu tujuan yang sama. Gambar 2.8 adalah ilustrasi dari cara kerja sistem terdistribusi secara paralel.



Gambar 2.8: Sistem Terdistribusi

2.10.1 Manfaat Sistem Terdistribusi

Berikut adalah manfaat penggunaan sistem terdistribusi:

- *Horizontal scalability*
  Sistem terdistribusi menawarkan kemampuan untuk melakukan pemrosesan komputasi skala besar pada big data dengan harga yang 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.

2.10.2 Tantangan Sistem Terdistribusi

Berikut adalah tantangan pada 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.

2.11 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 pada *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.

Big data memerlukan teknologi tertentu untuk melakukan komputasi. Pada bagian selanjutnya akan dijelaskan konsep-konsep terkait penggunaan *framework* beserta komponen-komponen penting pada *framework* tersebut terkait komputasi pada lingkungan *big data*. *Framework* tersebut antara lain Hadoop dan Spark. Masing-masing *framework* akan diteliti lebih lanjut, untuk dipilih pada penelitian ini berdasarkan kecepatan waktu komputasi.

2.12 Hadoop

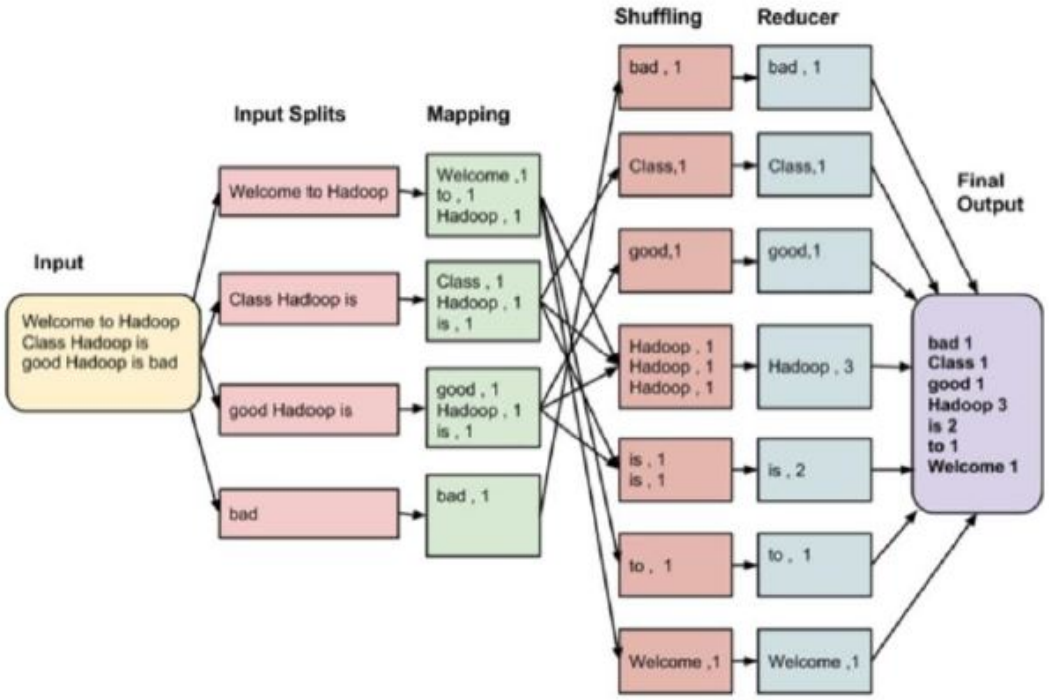
Hadoop adalah *framework* yang memanfaatkan beberapa komputer untuk menyelesaikan masalah yang melibatkan volume data sangat besar. Hadoop memecah input dari pengguna menjadi beberapa blok data dan masing-masing blok data diproses menggunakan konsep *MapReduce* di mana data akan diproses secara paralel pada sistem terdistribusi.

2.12.1 HDFS

HDFS adalah sistem *file* terdistribusi pada Hadoop yang menyediakan penyimpanan data yang handal dengan mendukung partisi data dan toleran terhadap kesalahan pada hardware. HDFS bekerja erat dengan MapReduce dengan mendistribusikan penyimpanan dan perhitungan di seluruh *cluster* dengan menggabungkan seluruh penyimpanan data menjadi terpusat.

2.12.2 MapReduce

MapReduce adalah model pemrograman untuk memproses data berukuran besar secara terdistribusi dan paralel pada cluster yang terdiri atas banyak komputer. Dalam memproses data, secara garis besar, MapReduce dibagi menjadi dua jenis proses yaitu map dan reduce. Setiap fase memiliki pasangan key-value sebagai input dan output. Kedua jenis proses ini didistribusikan ke setiap komputer dalam suatu cluster dan berjalan secara paralel tanpa saling bergantung satu sama lain.



Gambar 2.9: Proses Komputasi pada MapReduce

Berikut adalah penjelasan masing-masing tahapan pada MapReduce:

- Input
  Pada tahap ini, model MapReduce menerima input data secara utuh dari file text/CSV.
- Input Splits Pada tahap ini, model MapReduce akan memecah input data menjadi blok-blok data dan disebarkan ke seluruh cluster.
- *Mapping*
  Mapping bertujuan untuk memetakan blok-blok data ke dalam pasangan <*key*,*value*>. Key,Value pada contoh ini adalah jenis kata dan jumlah jenis kata pada sebuah blok data.
- Shuffling
  Shuffling bertujuan untuk mengirim data dari Mapping ke Reducer, agar data dengan key yang sama akan dikelompokan dan diolah oleh Reducer yang sama
- Reducer
  Reducer bertujuan sebagai proses penggabungan key,value dari proses shuffling untuk dihitung dan dikembalikan sebagai sebuah output
- Output
  Pada tahap ini, pemodelan MapReduce telah selesai. Output siap untuk ditulis pada file maupun ditampilkan pada console.

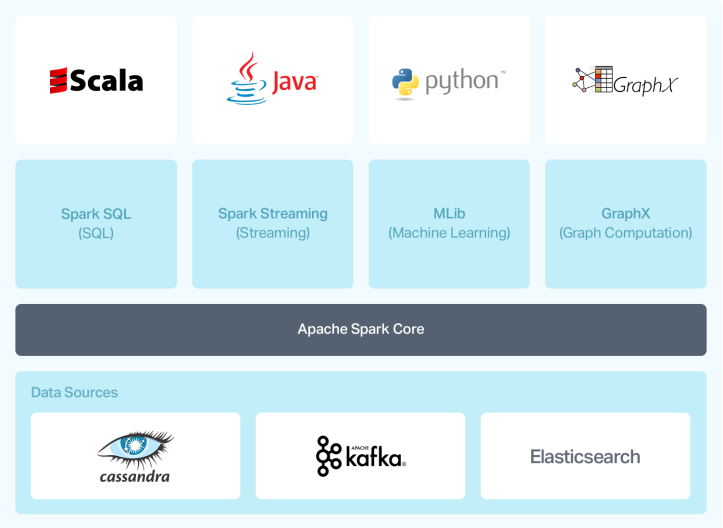
2.13 Spark

Spark adalah teknologi komputasi *cluster* 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 *cluster* 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 *real-time*, 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.

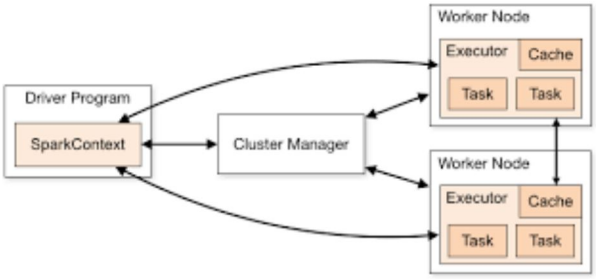
2.13.1 Ekosistem Spark



Gambar 2.10: Ekosistem Spark

Gambar 2.10 menunjukan bahwa Spark bekerja sama dengan teknologi *big data* lain untuk memenuhi berbagai macam kebutuhan dalam pengolahan *big data*. Masing-masing warna pada Gambar 2.10 mewakili jenis teknologi yang dipakai pada Spark. Spark SQL, Spark Streaming, Spark MLlib adalah *library* tambahan pada Spark. Cassandra, Kafka, dan ElasticSearch adalah *framework* untuk melakukan pengumpulan data secara *streaming*. Scala, Java, dan Python adalah bahasa pemrograman yang dapat digunakan pada Spark.

2.13.2 Arsitektur Spark

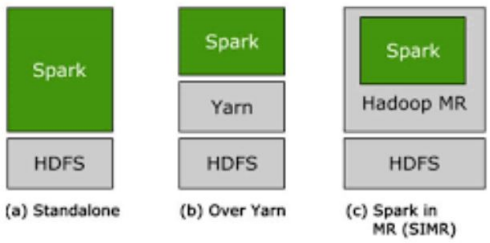


Gambar 2.11: Arsitektur Spark

Berdasarkan Gambar 2.11, berikut adalah beberapa hal penting terkait arsitektur Spark:

- *Driver Program* bertugas untuk menjalankan *Object main* pada *master node*. *Driver Program* adalah tempat dimana *Spark Context* dibuat.
- *Spark Context* bertugas untuk menghubungkan pengguna dengan *cluster*. *Spark Context* juga digunakan untuk membuat RDD, *accumulator*, dan *broadcast variable*.
- *Cluster Manager* bertugas untuk mengatur sumber daya pada sebuah *cluster*.
- *Executor* membantu memantau proses-proses yang berjalan pada *worker node* dan bertanggung jawab untuk mengerjakan *task* yang diberikan.
- *Task* adalah satuan kerja pada Spark yang berisi perintah-perintah fungsi yang diserialisasi.

2.13.3 Jenis Instalasi pada Spark



Gambar 2.12: Arsitektur Spark

Berdasarkan Gambar 2.12, berikut adalah jenis-jenis instalasi pada Spark:

- *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

2.13.4 Resilient Distibuted Datasets (RDD)

RDD adalah kumpulan partisi terdistribusi yang disimpan dalam memori atau *disk* pada beberapa *cluster*. RDD tersebar menjadi beberapa partisi, sehingga partisi tersebut dapat disimpan dan diproses pada komputer yang berbeda.

Berikut adalah beberapa karakteristik yang dimiliki RDD:

- *Lazy evaluation*: operasi pada Spark hanya akan dilakukan ketika memanggil fungsi *Action*.
- *Immutability*: data yang disimpan dalam RDD tidak dapat diubah nilainya.
- *In-memory computation*: RDD menyimpan data secara langsung dalam memori.
- *Partitioning*: dapat membagi pekerjaan RDD pada beberapa komputer.

Berikut adalah jenis operasi pada RDD:

- Fungsi *Transformation*
  Fungsi *transformation* dilakukan secara *lazy*, sehingga hanya akan dikerjakan apabila dipanggil pada fungsi *action*. Fungsi *transformation* 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

- Fungsi *Action*
  Fungsi *Action* adalah operasi yang mengembalikan nilai output ke dalam terminal atau melakukan penulisan data pada sistem penyimpanan eksternal. Fungsi *Action* memaksa evaluasi pada RDD yang akan dipanggil, untuk menghasilkan output. Fungsi *Action* pada RDD akan dijelaskan pada tabel dibawah ini.

Fungsi	Deskripsi
count()	Mendapat jumlah elemen data dalam RDD
reduce()	Agregat elemen data ke dalam RDD dengan mengambil dua argumen dan mengembalikan satu
foreach(operation)	Menjalankan operasi untuk setiap elemen data dalam RDD

2.13.5 *Dataframe*

Dataframe adalah kumpulan data yang didistribusikan, disusun dalam baris dan kolom. Setiap kolom dalam *Dataframe* memiliki nama dan tipe terkait. *Dataframe* mirip dengan tabel database tradisional, yang terstruktur dan ringkas. Dengan menggunakan *Dataframe*, kueri SQL dapat dengan mudah diimplementasi pada *big data*.

Berikut adalah beberapa karakteristik yang dimiliki *Dataframe*:

- Terdiri atas baris dan kolom seperti tabel.
- Memiliki skema untuk penyimpanan data
- Data yang dapat disimpan berupa numerik dan kategorikal.
- Dapat dilakukan pemrosesan kueri SQL.

2.13.6 Komponen Spark

Komponen Spark adalah library tambahan pada Spark untuk melakukan proses komputasi pada lingkungan big data berdasarkan jenis-jenis kebutuhan pengolahan data. 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 menyediakan komputasi pada memori, fungsi *action* dan *transformation* untuk mengolah RDD.

- Spark SQL
  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.

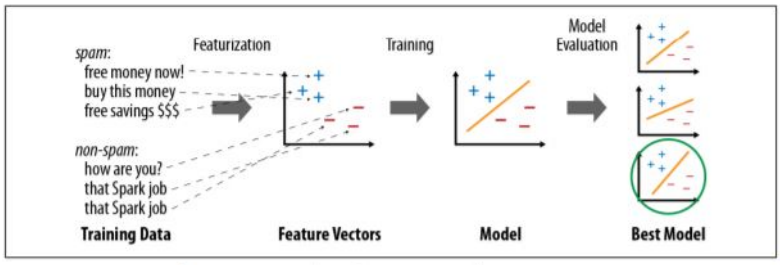
- Spark Streaming
  Spark Streaming adalah salah satu komponen Apache Spark, yang memungkinkan Spark dapat memproses data *streaming* secara *real-time*. Spark Streaming menyediakan API untuk memanipulasi aliran data yang cocok dengan RDD. Hal ini memungkinkan analisis data untuk beralih melalui sumber aplikasi yang memberikan data secara *real-time*.

- Spark MLlib
  Spark MLlib adalah *library* Spark yang berisi fungsionalitas yang umum digunakan pada *machine learning*. Untuk mengimplementasikan teknik *data mining* pada lingkungan *big data* dibutuhkan *library* Spark MLlib. Spark MLlib menyediakan berbagai jenis algoritma *machine learning* termasuk klasifikasi dan pengelompokan/*clustering*.

2.14 Spark MLlib

Spark MLlib adalah library pembelajaran mesin berdasarkan komputasi secara paralel. MLlib terdiri dari algoritma pembelajaran umum seperti klasifikasi, pengelompokan/*clustering*. Secara garis besar, MLlib melakukan data *preprocessing*, pelatihan model, dan membuat prediksi.

2.14.1 Machine Learning pada Spark MLlib



Gambar 2.13: Tahapan Pembelajaran Machine Learning

Machine learning bertujuan membuat prediksi label/kelompok data berdasarkan jenis model yang dipakai. Pemodelan *machine learning* mencakup model dari*data mining*. Pemodelan *machine learning* membutuhkan input berupa vektor fitur. Vektor fitur adalah nilai masing-masing atribut yang digunakan pada pelatihan data.

Gambar 2.13 adalah tahapan *machine learning* pada Spark MLlib, berikut adalah penjelasan singkat dari masing-masing tahapan:

1. *Featurization*
   Pemodelan *machine learning* hanya dapat menerima input berupa vektor. Oleh karena itu, nilai atribut pada tabel akan diubah ke representasi numerik dalam bentuk vektor.

2. *Training*
   Pemodelan *machine* learning melakukan pelatihan agar model yang dipakai memberikan hasil yang tepat untuk menentukan label atau kelompok data. Oleh karena itu, pemodelan *machine learning* memerlukan pelatihan model beberapa kali untuk mendapatkan model terbaik.

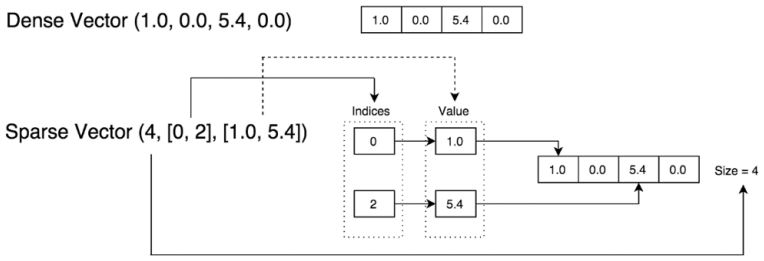
3. *Model Evaluation*
   Pada akhir pelatihan, model yang terbentuk dapat diputuskan baik atau tidak melalui perhitungan nilai akurasi. Semakin besar nilai akurasi, maka model dapat digunakan untuk memprediksi nilai label atau kelompok data secara tepat.

2.14.2 Tipe Data pada Spark MLlib

Seperti yang sudah dijelaskan pada bagian 2.14.1, pemodelan *machine learning* menerima input berupa vektor fitur. Tipe data yang disediakan pada Spark MLlib terdiri dari beberapa jenis yaitu vektor, *labeledpoint*, dan *various model class*.

Berikut adalah beberapa jenis tipe data pada Spark MLlib:

- Vektor
  Vektor terdiri dari dua jenis yaitu vektor dense dan vektor sparse. Kelas vektor berada pada *package* mllib.linalg.Vectors. Gambar 2.14 adalah contoh vektor dense dan vektor sparse:



Gambar 2.14: Contoh Vektor Dense dan 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* 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. Kelas *LabeledPoint* terletak pada *package* mllib.regress.

- *Various Model class*
  Various Model classes adalah tipe data yang dihasilkan dari pemodelan *machine learning*. Tipe data ini memiliki fungsi predict() untuk melakukan prediksi label dan kelompok data.

2.14.3 *Data Mining* pada Spark MLlib

Data mining pada Spark MLlib menggunakan tahapan pemodelan pada *machine learning* yang dijelaskan pada bagian 2.14.1 untuk menghasilkan tabel hasil pengelompokan dan klasifikasi. Pada bagian ini akan dijelaskan parameter dari pemodelan Spark MLlib.

Naive Bayes

Naive Bayes menjadi pemodelan klasifikasi yang umum digunakan. *Naive Bayes* dapat dilatih dengan sangat efisien karena prosesnya hanya menghitung probabilitas bersyarat. *Naive Bayes* memiliki parameter masukan sebagai berikut:

- *randomSplit* adalah membagi *training* dan *test* data berdasarkan persentase.
- *setModelType* adalah memilih model yang tersedia (*multinomial/bernoulli*)
- *setLabelCol* adalah memilih jenis atribut yang menjadi label kelas.

K-Means

K-means menjadi pemodelan pengelompokan/*clustering* yang paling umum digunakan untuk mengelompokkan titik-titik data menjadi sejumlah kelompok yang telah ditentukan. *K-means* memiliki parameter masukan sebagai berikut:

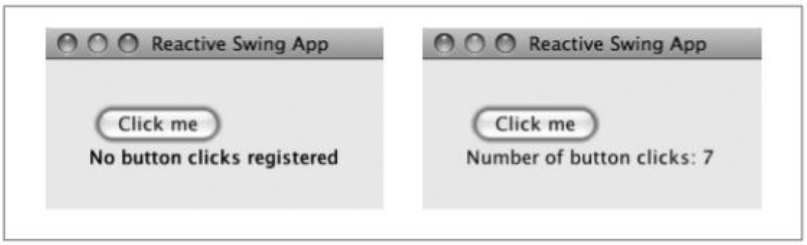
- k adalah jumlah cluster yang diinginkan.
- *maxIterations* adalah jumlah iterasi maksimum yang harus dijalankan.
- *initializationMode* menentukan inisialisasi centroid secara acak.
- *initializationSteps* menentukan jumlah langkah dalam algoritma *k-means*.
- *initialModel* adalah menentukan nilai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

2.15 Scala

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

2.16 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 2.15 adalah contoh implementasi GUI sederhana pada Scala Swing.



Gambar 2.15: GUI Sederhana pada Scala Swing

2.16.1 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. *Layout* terdiri dari beberapa komponen GUI seperti *Frame, Panel, Label* atau *Button*. Masing-masing komponen GUI pada *layout* memiliki nilai properti sendiri (warna, ukuran, posisi) yang dapat diatur secara manual.

2.16.2 Handling Event

Handling event adalah perkerjaan 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*.

2.17 Format Penyimpanan Data

Spark dapat melakukan aksi membaca dan menulis pada data terstruktur dan semi terstruktur. Contoh data terstruktur yang umum digunakan adalah CSV, sedangkan contoh data semi terstruktur yang umum digunakan adalah JSON. Berikut adalah penjelasan lengkap mengenai format penyimpanan data CSV dan JSON.

2.17.1 CSV

CSV *(Comma Separated Values)* menjadi format yang sangat umum digunakan untuk menyimpan nilai pada tabel data yang terstruktur. CSV menggunakan format ekstensi (.csv) saat berdiri sendiri. Hasil penyimpanan dengan format CSV umum digunakan untuk menyimpan data saat ingin menyimpan tabel dari basis data. CSV memisahkan nilai atribut yang satu dengan yang lainnya menggunakan tanda koma. CSV dapat memisahkan data yang satu dengan data lainnya berdasarkan penempatan data pada baris yang berbeda. Listing 3.1 adalah contoh format penyimpanan CSV.

Listing 2.1: Format Penyimpanan CSV

```
age,workclass,zip,education,year_of_education,marital_status,occupation
39,State-gov,77516,Bachelors,13,Never-married,Adm-clerical
50,Self-emp-not-inc,83311,Bachelors,13,Married-civ-spouse,Exec-managerial
38,Private,215646,HS-grad,9,Divorced,Handlers-cleaners
53,Private,234721,11th,7,Married-civ-spouse,Handlers-cleaners
28,Private,338409,Bachelors,13,Married-civ-spouse,Prof-specialty
37,Private,284582,Masters,14,Married-civ-spouse,Exec-managerial
49,Private,160187,9th,5,Married-spouse-absent,Other-service
52,Self-emp-not-inc,209642,HS-grad,9,Married-civ-spouse,Exec-managerial
```

2.17.2 JSON

JSON *(JavaScript Object Notation)* adalah format untuk pertukaran data. JSON menggunakan format ekstensi (.json) saat berdiri sendiri. JSON diturunkan dari bahasa pemrograman JavaScript. Walaupun diturunkan dari bahasa pemrograman lain, JSON tidak bergantung pada bahasa pemrograman apapun. Oleh karena itu, format JSON sangat mudah dipakai untuk pertukaran data antar bahasa pemrograman. JSON memiliki format penyimpanan *key-value* seperti pada Listing 2.2. JSON menyimpan enam jenis tipe data yaitu *string, number, object, array, boolean, null*. Menulis format JSON dalam beberapa baris akan lebih mudah dibaca terutama saat datanya sudah banyak.

Listing 2.2: Format Penyimpanan JSON

```
{
    "firstName": "Rack",
    "lastName": "Jackon",
    "gender": "man",
    "age": 24,
    "address": {
        "streetAddress": "126",
        "city": "San Jone",
        "state": "CA",
        "postalCode": "394221"
    },
    "phoneNumbers": [
        { "type": "home", "number": "7383627627" }
    ]
}
```