

# PENERAPAN ALGORITMA ANONIMISASI PADA LINGKUNGAN BIG DATA

STEPHEN JORDAN—2016730018

## 1 Data Skripsi

Pembimbing utama/tunggal: **Mariskha Tri Adithia, P.D.Eng**

Pembimbing pendamping: **Dr. Veronica Sri Moertini, Ir., M.T.**

Kode Topik : **MTA4801**

Topik ini sudah dikerjakan selama : **1 semester**

Pengambilan pertama kali topik ini pada : **Semester 48 - Genap 19/20**

Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**

Tipe Laporan : **B -** Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

## 2 Latar Belakang

Berkembangnya penggunaan sistem informasi di jaman sekarang mengakibatkan data dihasilkan dalam jumlah yang sangat banyak. Data yang jumlahnya sangat banyak ini dikumpulkan dan disimpan dalam tabel basis data untuk keperluan analisis data di masa yang akan datang. Data yang dikumpulkan secara terus-menerus apabila tidak dilakukan analisis secara berkala, maka suatu saat nanti jika proses analisis data dilakukan, proses analisis tersebut berlangsung sangat lama karena ukuran datanya sudah terlanjur besar. Data yang terus bertumbuh menyebabkan basis data konvensional menjadi kurang efektif untuk mengolah data. Teknologi *big data* digunakan untuk mengurangi biaya penyimpanan dan komputasi data, sehingga kapasitas data dapat ditingkatkan dan data berukuran besar menjadi lebih mudah untuk diolah.

*Big data* adalah kumpulan data dalam jumlah data yang sangat besar disimpan, diolah, dan dianalisis untuk menghasilkan informasi yang bermanfaat sebagai dasar pengambilan keputusan atau kebijakan. Karena *big data* memiliki ukuran data yang besar, sehingga untuk melakukan analisis pada *big data*, data yang sudah terkumpul akan dibagi ke beberapa komputer untuk diolah secara paralel. Konsep ini disebut sistem terdistribusi. Sistem terdistribusi adalah solusi dari pengolahan *big data* karena terbukti dapat mengurangi biaya penyimpanan dan komputasi data karena dilakukan secara paralel. Untuk melakukan analisis data, diperlukan teknik khusus untuk mencari tahu pola apa saja yang terbentuk dari sekumpulan data yang telah dikumpulkan. Oleh karena itu, diperlukan teknik *data mining* untuk melakukan analisis data.

*Data mining* adalah teknik yang diciptakan untuk melihat pola yang terbentuk dari sekumpulan data yang telah terkumpul dalam jumlah yang besar. *Data mining* terbukti efektif untuk menggantikan pemrosesan kueri pada basis data konvensional dalam kasus analisis *big data*, karena basis data konvensional tidak menerapkan konsep sistem terdistribusi sehingga waktu komputasinya sangat lambat. Pemodelan data mining nantinya akan dijalankan pada sistem terdistribusi, sehingga waktu komputasi untuk analisis *big data* dapat diminimalkan. Hasil *data mining* nantinya akan dipakai untuk berbagai macam kebutuhan. Biasanya sebuah perusahaan akan meminta data dari perusahaan lain untuk kebutuhan analisis. Masalah yang umum terjadi adalah data hasil pengolahan *data mining* banyak mengandung data yang bersifat privasi sehingga perlu adanya cara untuk menjamin perlindungan privasi pada data yang akan didistribusikan.

Perlindungan privasi untuk distribusi data dapat dicapai dengan menggunakan metode enkripsi dan anonimisasi pada hasil pengolahan data mining. Enkripsi adalah metode yang memanfaatkan pola atau kunci tertentu untuk melindungi data yang sifatnya sensitif. Anonimisasi adalah metode yang menyamarkan satu atau lebih nilai atribut data agar data seseorang tidak dapat saling dibedakan dengan data lainnya. Salah satu kekurangan dari metode enkripsi dibandingkan metode anonimisasi adalah keamanan enkripsi dapat

diretas melalui penalaran hubungan nilai atribut yang unik untuk setiap baris data. Penalaran ini dicapai dengan menggabungkan seluruh nilai atribut yang unik pada masing-masing baris data untuk membentuk sebuah pola kelompok data. Penalaran ini sangat berbahaya karena menghubungkan nilai atribut data yang secara tidak langsung dapat mengungkapkan entitas pemilik data. Dengan menerapkan konsep anonimisasi diharapkan nilai keterhubungan antar atribut data diperkecil sehingga privasi dapat terlindungi

Dengan melakukan anonimisasi pada sebagian nilai atribut, bobot informasi yang diperoleh akan semakin kecil. Semakin kecil bobot informasi yang diperoleh maka pola untuk membentuk kelompok entitas data semakin kecil sehingga perlindungan privasi akan semakin aman. Akan tetapi dengan semakin kecil bobot informasi yang diperoleh maka nilai akurasi yang dihasilkan oleh metode anonimisasi akan semakin kecil. Oleh karena itu diperlukan cara untuk menyeimbangkan keamanan dan nilai akurasi informasi. Permasalahan *k-anonymity* adalah pencarian solusi untuk menyeimbangkan nilai akurasi informasi yang diperoleh dengan nilai informasi yang dilindungi. 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. Agar algoritma *Greedy k-member clustering* dapat berjalan dengan waktu pemrosesan yang tidak terlalu lama, maka akan digunakan *framework* Spark untuk mengatasi permasalahan *big data*.

Spark adalah *framework* yang tepat untuk melakukan proses anonimisasi data pada lingkungan *big data*, karena pekerjaan pengolahan data yang besar dapat dibagi ke beberapa komputer pada sistem terdistribusi. Penggunaan Spark dipilih karena Hadoop memiliki waktu pemrosesan *big data* yang lebih lama dari Spark dengan melakukan komputasi pada *hardisk*, sedangkan Spark melakukan komputasi pada memori. Selain itu Spark memiliki jenis *library* yang lebih beragam dibandingkan dengan Hadoop. Spark mampu melakukan pemrosesan teknik *data mining* pada lingkungan *big data* menggunakan *library* tambahan yaitu Spark MLlib. Spark MLlib memfasilitasi pemodelan *data mining* yaitu klasifikasi dan pengelompokan/*clustering*. Kekurangan dari Spark adalah tidak mempunyai penyimpanan yang tetap, sehingga membutuhkan mekanisme penyimpanan Hadoop, agar hasil pemrosesan data dapat tersimpan di dalam *hardisk* komputer.

Pada skripsi ini, akan dibuat dua jenis perangkat lunak yaitu perangkat lunak untuk anonimisasi data dan perangkat lunak untuk analisis data. Perangkat lunak anonimisasi data menggunakan konsep *k-anonymity* dengan implementasi algoritma *Greedy k-member clustering* agar sebuah data tidak dapat dibedakan dengan  $k - 1$  data lainnya. Perangkat lunak anonimisasi data dibuat dengan bahasa Scala dan berjalan di atas Spark untuk meminimalkan waktu komputasi pada proses anonimisasi di lingkungan *big data*. Algoritma *Greedy k-member clustering* dinilai tepat untuk melakukan pengelompokan data karena meminimalkan jumlah informasi yang hilang saat proses *data mining* yang terbukti pada penelitian sebelumnya. Kedua jenis perangkat lunak ini menerima data input dalam format CSV. Untuk tampilannya, kedua jenis perangkat lunak akan dibuat menggunakan GUI dari *library* Scala-swing. Penelitian ini memiliki tujuan utama yaitu membandingkan nilai akurasi dari hasil *data mining* sebelum dan setelah dilakukan anonimisasi.

### 3 Rumusan Masalah

Berdasarkan latar belakang di atas, 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?

## 4 Tujuan

Berdasarkan rumusan masalah di atas, tujuan pada 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.

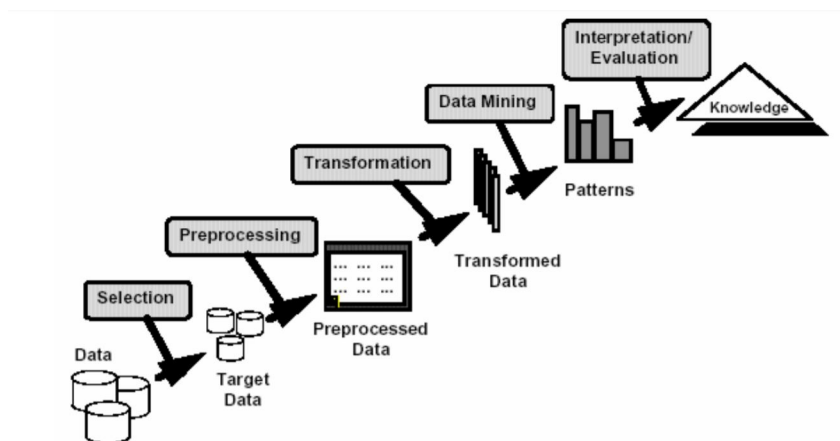
## 5 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencana kerja/laporan perkembangan terakhir:

1. Melakukan studi literatur mengenai teknik-teknik dasar data mining

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** 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 1: Tahapan pada KDD

Berikut ini adalah penjelasan tahapan pada KDD pada Gambar 1 sebagai berikut:

- (a) *Selection*: proses mengambil data yang relevan terhadap analisis.
- (b) *Preprocessing*: proses pembersihan data dari data yang tidak konsisten dan integrasi data saat penggabungan data.
- (c) *Transformation*: proses manipulasi data menggunakan konsep agregasi, generalisasi, normalisasi, dan reduksi untuk kebutuhan analisis.
- (d) *Data mining*: proses ekstraksi informasi menggunakan metode pengenalan pola seperti klasifikasi, pengelompokan/*clustering*.
- (e) *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*.

### 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:

- 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.
- 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: Contoh Logistic Regression

Contoh: *Logistic Regression*

Gambar 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 yang umum digunakan:

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

### 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:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)} \quad (1)$$

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

$$MAP(H) = \max(P(H|D)) \quad (2)$$

Keterangan:

- $P(H|D)$  adalah probabilitas posterior apabila diberikan 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.

Gambar 3 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 3: Dataset Kondisi Cuaca 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:

- (a) Merepresentasikan teorema Bayes terhadap vektor fitur.

Berdasarkan dataset, teorema Bayes dapat diubah seperti berikut:

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)} \quad (3)$$

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

$$X = (x_1, x_2, x_3, \dots, x_n) \quad (4)$$

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:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)P(y)}{P(x_1)P(x_2) \dots P(x_n)} \quad (5)$$

(b) Gambar 4 adalah contoh menghitung probabilitas masing-masing atribut.

Outlook					Temperature				
	Yes	No	P(yes)	P(no)		Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5	Hot	2	2	2/9	2/5
Overcast	4	0	4/9	0/5	Mild	4	2	4/9	2/5
Rainy	3	2	3/9	2/5	Cool	3	1	3/9	1/5
Total	9	5	100%	100%	Total	9	5	100%	100%

Humidity					Wind				
	Yes	No	P(yes)	P(no)		Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5	False	6	2	6/9	2/5
Normal	6	1	6/9	1/5	True	3	3	3/9	3/5
Total	9	5	100%	100%	Total	9	5	100%	100%

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

Gambar 4: Menghitung Probabilitas

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

$$P(No) = \frac{\text{frekuensi}(Sunny \cap No)}{\text{frekuensi}(No)} \quad (6)$$

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

$$P(Yes) = \frac{\text{frekuensi}(Sunny \cap Yes)}{\text{frekuensi}(Yes)} \quad (7)$$

(c) Menghitung probabilitas bersyarat jika diketahui nilai dari data baru.

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

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

$$P(Yes|today) = \frac{3}{9} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \cdot \frac{5}{14} = 0.0068 \quad (8)$$

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

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

(d) Melakukan normalisasi terhadap probabilitas bersyarat.

Setelah probabilitas bersyarat dinormalisasi, akan menjadi seperti berikut:

$$P(Yes|today) = \frac{0.0141}{0.0141 + 0.0068} = 0.67 \quad (10)$$

$$P(No|today) = \frac{0.0068}{0.0141 + 0.0068} = 0.33 \quad (11)$$

Sehingga memiliki probabilitas total seperti berikut:

$$P(Yes|today) + P(No|today) = 1 \quad (12)$$

(e) Mencari probabilitas tertinggi.

Berdasarkan pernyataan berikut:

$$P(Yes|today) > P(No|today) \quad (13)$$

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

### 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/*cluster* yang sama sangat mirip sedangkan titik data dalam kelompok berbeda sangat berbeda. Contoh pemodelan *clustering* adalah *K-Means*.

#### K-Means

Algoritma *k-means* adalah algoritma pembelajaran mesin *unsupervised learning* untuk menentukan objek tersebut benar-benar milik kelompok data tertentu. *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*:

$$EuclidDist(p_i, C_i) = \sqrt{(p_1 - C_1)^2 + (p_2 - C_2)^2 + \dots + (p_n - C_n)^2} \quad (14)$$

Gambar 5 adalah 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 5: Contoh Dataset K-Means



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

- (a) Gambar 6 adalah hasil pengelompokan awal untuk  $k = 2$ . 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 6: Hasil Pengelompokan Awal

- (b) Data yang tersisa akan diperiksa secara berurutan dan dialokasikan pada cluster yang paling dekat dengan *centroid* awal menggunakan *Euclidean distance*. Gambar 7 menunjukkan vektor rata-rata (centroid) akan dihitung ulang setiap kali anggota baru ditambahkan.

	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)

Gambar 7: Mencari Centroid Kelompok

- (c) 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 8: Hasil Pengelompokan Baru

- (d) 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. Gambar 9 adalah tabel hasil perbandingan *distance* yang dihitung menggunakan rumus *Euclidian distance*.

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 9: Euclidean Distance Cluster 1, Cluster 2

- (e) 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*. Gambar 10 adalah hasil pengelompokan akhir yang dihasilkan oleh pemodelan *k-means*.

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

Gambar 10: Hasil Pengelompokan Akhir

## 2. Melakukan studi literatur mengenai algoritma Greedy k-member clustering

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** 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-Hard, artinya memiliki kompleksitas yang eksponensial.*

*Bukti.* Melalui pengamatan Aggarwal et al, permasalahan *k-member clustering* dapat diselesaikan dengan kompleksitas polinomial.  $\square$

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

**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  $O(n^2)$ .  $\square$

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

- Menetapkan tabel  $S$
- Menetapkan nilai  $k$

- Menetapkan jumlah cluster ( $m$ ) yang ingin dibuat

$$m = \left\lfloor \frac{n}{k} \right\rfloor - 1 \quad (15)$$

Berikut adalah langkah kerja algoritma *Greedy k-means clustering* secara lengkap:

- Melakukan inisialisasi variabel result dengan himpunan kosong dan variabel  $r$  dengan memilih data secara acak dari tabel  $S$
- Pada kondisi  $|S| \geq k$ , lakukan perulangan sebagai berikut:
  - 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:

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

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

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

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

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

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

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

- iv. Menambahkan himpunan data dari variabel  $r$  pada variabel  $c$ .
- (d) 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 *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  $c \in C$  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 *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  $|S| \leq k$  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 = 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 = 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 *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.

### 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 menghitung *distance* dari kedua jenis data pada saat yang sama.

*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:

$$\delta_n(v_1, v_2) = \frac{|v_1 - v_2|}{|D|} \quad (16)$$

*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:

$$\delta_C(v_1, v_2) = \frac{H(\Lambda(v_i, v_j))}{H(T_D)} \quad (17)$$

*Distance* antar *record* adalah gabungan dari *distance* numerik dan kategorikal. Beberapa atribut *distance* antar *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,  $\delta_N$  adalah *distance* data numerik,  $\delta_C$  adalah *distance* data kategorikal. *Distance* record dihitung menggunakan rumus berikut:

$$\Delta(r_1, r_2) = \sum_{i=1}^m \delta_N(r_1[N_i], r_2[N_i]) + \sum_{j=1}^n \delta_C(r_1[C_j], r_2[C_j]) \quad (18)$$

### 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:

$$IL(e) = |e| \cdot D(e) \quad (19)$$

$$D(e) = \sum_{i=1}^m \frac{(MAX_{N_i} - MIN_{N_i})}{|N_i|} + \sum_{j=1}^n \frac{H(\Lambda(\cup_{C_j}))}{H(T_{C_j})} \quad (20)$$

Total *Information Loss* dihitung dengan rumus sebagai berikut:

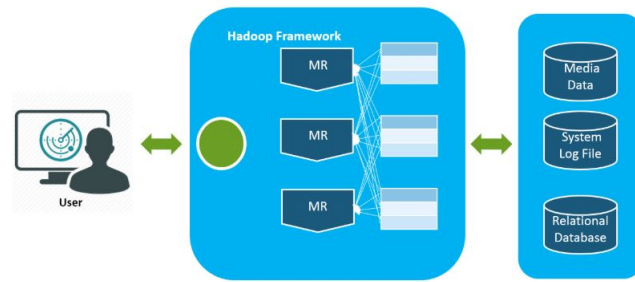
$$Total - IL(AT) = \sum_{e \in \epsilon} IL(e) \quad (21)$$

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.

### 3. Melakukan studi literatur mengenai konsep Hadoop, Spark, dan Spark MLlib

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** Hadoop adalah *framework* yang memanfaatkan beberapa komputer untuk menyelesaikan masalah yang melibatkan volume data sangat besar. Pada Gambar 11, 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.

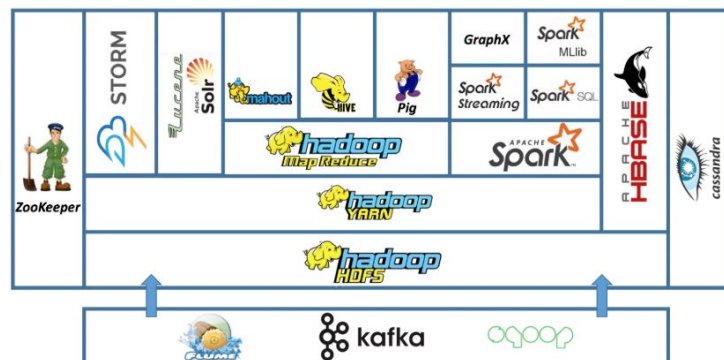


Gambar 11: Gambaran Umum Cara Kerja Hadoop

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.

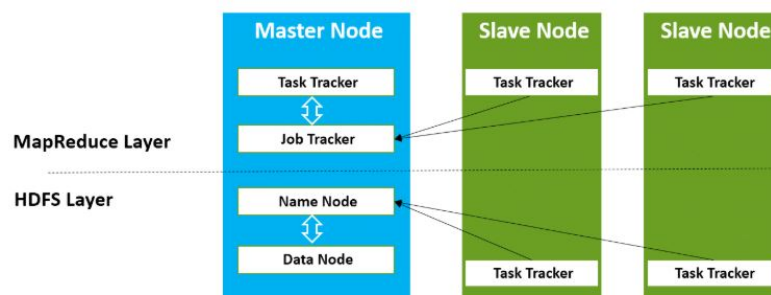
### Ekosistem Hadoop



Gambar 12: Ekosistem Hadoop

Gambar 12 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*.

### Arsitektur Hadoop



Gambar 13: Arsitektur Hadoop

Gambar 13 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*.

### Hadoop HDFS

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

HDFS terdiri dari beberapa komponen, yaitu:

- *NameNode*

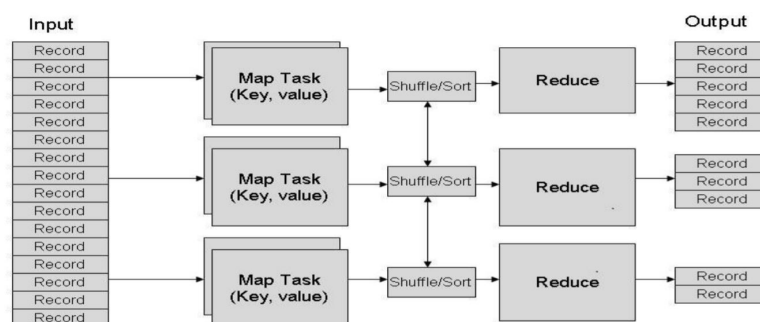
*NameNode* adalah sebuah komputer yang bertindak sebagai *master node*, sedangkan. *NameNode* bertanggungjawab menyimpan informasi tentang penempatan blok-blok data dalam *Hadoop cluster*. Ia bertanggungjawab mengorganisir dan mengontrol blok-blok data yang disimpan tersebar dalam komputer-komputer yang 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 *DataNode* direplikasi sesuai konfigurasi untuk memberikan keandalan dan ketersediaan tinggi. Blok yang direplikasi ini didistribusikan di seluruh cluster untuk memberikan perhitungan yang cepat.

### Hadoop 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. Pemodelan *MapReduce* terdiri dari dua fungsi, yaitu *Mapper* dan *Reducer*.



Gambar 14: Tahapan pada MapReduce

Pada Gambar 14, tahapan *MapReduce* dapat dibagi menjadi dua fungsi utama:

- *Mapper*

Tugas fungsi *Mapper* adalah memetakan blok data kedalam pasangan  $\langle key, value \rangle$ . *Key, value* pada *Mapper* tidak harus memiliki tipe data yang sama satu sama lain. Pasangan  $\langle key, value \rangle$  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*:

- (a) *Shuffle*

*Shuffle* adalah fase pada data antara untuk menggabungkan semua nilai menjadi koleksi yang terkait dengan kunci yang sama.

- (b) *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*.

- (c) *Reducer*

Hasil *mapper* yang diacak dan diurutkan disediakan untuk *Reducer*. Tahap ini membuat pasangan  $\langle \text{key}, (\text{list of value}) \rangle$  baru berdasarkan pengelompokan *key*.

*MapReduce* memiliki beberapa komponen, yaitu:

- *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*.

## 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 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.

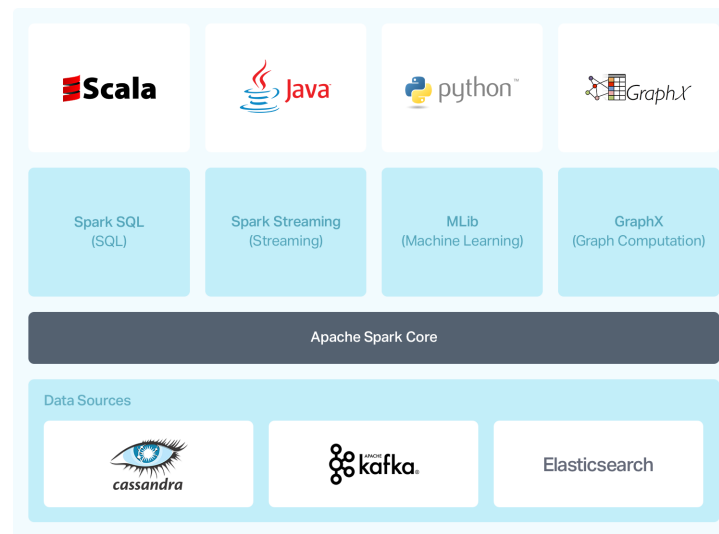
- Mudah untuk diatur

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 *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.

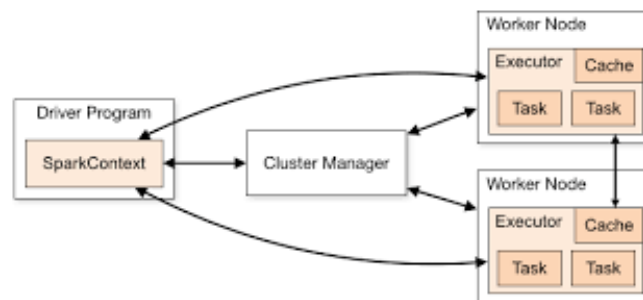
## Ekosistem Spark



Gambar 15: Ekosistem Spark

Gambar 15 menunjukkan bahwa Spark bekerja sama dengan teknologi *big data* lain untuk memenuhi berbagai macam kebutuhan dalam pengolahan *big data*. Masing-masing warna pada Gambar 15 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.

## Arsitektur Spark

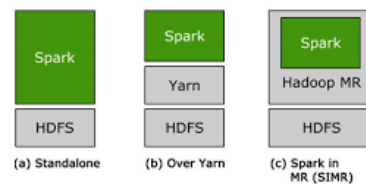


Gambar 16: Arsitektur Spark

Berdasarkan Gambar 16, berikut adalah tahapan kerja pada arsitektur Spark:

- 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*.
- Spark Driver* dan *SparkContext* secara kolektif mengawasi pelaksanaan pekerjaan di dalam *cluster*. *Spark Driver* bekerja sama dengan *Cluster Manager* untuk membagikan pekerjaan ke setiap *Worker Node*.
- Worker Node* menjalankan tugas yang diberikan oleh *Cluster Manager* dan mengembalikannya ke *SparkContext*. *Worker Node* bertanggung jawab atas pelaksanaan tugas yang diberikan.

## Jenis Instalasi Spark



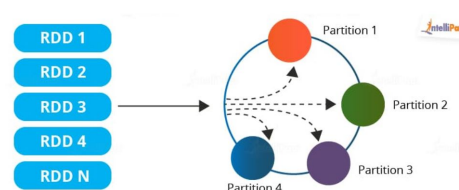
Gambar 17: Arsitektur Spark

Berdasarkan Gambar 17, 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

## Spark RDD

RDD adalah kumpulan objek terdistribusi yang disimpan dalam memori atau *disk* pada beberapa *cluster*. Gambar 18 menunjukan bahwa RDD tersebar menjadi beberapa partisi sehingga partisi tersebut dapat disimpan dan diproses pada komputer yang berbeda. RDD adalah struktur data yang *immutable*, artinya data tersebut hanya dapat dibaca dan tidak dapat diubah nilainya. RDD dalam Spark dapat di simpan dalam *cache* dan digunakan lagi untuk kebutuhan di masa depan.



Gambar 18: Cara Kerja RDD

## Spark 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. *DataFrame* adalah basis data relasional dengan teknik optimisasi yang lebih baik. Dengan *DataFrame*, kueri SQL dapat diimplementasi pada lingkungan *big data*.

## 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.

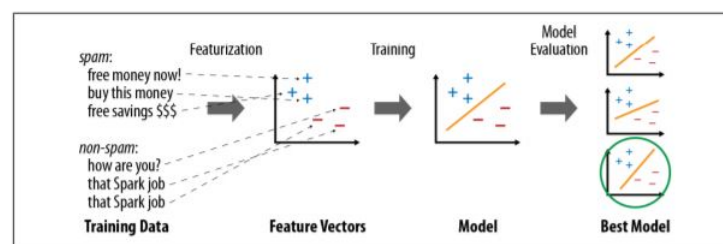
Berikut adalah tahapan yang terjadi pada Spark MLlib:

- (a) Membuat RDD baru untuk merepresentasikan jenis fitur dalam dataset.
- (b) Melakukan ekstraksi fitur pada sebuah vektor dan konversi menjadi data numerik.
- (c) Menjalankan algoritma *machine learning* untuk menghasilkan pemodelan baru.
- (d) Melakukan evaluasi model dengan memanggil fungsi evaluasi milik Spark MLlib.

Berikut adalah contoh pemodelan pada Spark MLlib:

- *Naive Bayes* (klasifikasi)
- *K-Means* (pengelompokan/*clustering*)

### Machine Learning pada Spark MLlib



Gambar 19: Tahapan Pembelajaran Machine Learning

*Machine learning* berupaya untuk membuat prediksi label/kelompok data berdasarkan jenis data pelatihan yang diberikan terhadap model yang dipakai. Pemodelan *machine learning* termasuk ke dalam pemodelan *data mining*. Jenis pemodelan machine learning terdiri dari klasifikasi, regresi, dan pengelompokan/*clustering*. Seluruh jenis pemodelan *machine learning* membutuhkan input berupa vektor fitur. Vektor fitur adalah nilai masing-masing atribut yang digunakan pada pelatihan data. Sebagian besar pemodelan *machine learning* hanya menerima input vektor berupa data numerik untuk mewakili vektor fitur pada setiap baris data.

Gambar 19 adalah tahapan *machine learning* pada Spark MLlib, berikut adalah penjelasan singkat dari masing-masing tahapan:

- (a) *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.
- (b) *Training*  
Pelatihan model merupakan salah satu tahapan yang paling memakan waktu dan padat karya dalam setiap alur kerja *machine learning*. 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.
- (c) *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.

### Struktur Data pada Spark MLlib

Seperti yang sudah dijelaskan pada bagian sebelumnya, pemodelan *machine learning* menerima input berupa vektor fitur. Struktur data yang disediakan pada Spark MLlib terdiri dari beberapa jenis yaitu vektor, *labeledpoint*, dan *various model class*. Masing-masing tipe data pada Spark MLlib ditentukan berdasarkan jenis pemodelan yang dipilih. Contoh apabila menggunakan pemodelan *clustering* akan digunakan tipe data vektor, sedangkan untuk pemodelan klasifikasi akan digunakan struktur data *labeledpoint*. Berikut adalah struktur data pada Spark MLlib:

- Vektor

Vektor terdiri dari dua jenis yaitu vektor dense dan vektor sparse. Kelas vektor berada pada package `mllib.linalg.Vectors`. Berikut penjelasan singkat 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 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`. Kelas *LabeledPoint* terletak pada package `mllib.regress`.

### Data Mining pada Spark MLlib

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

(a) *Naive Bayes*

*Naive Bayes* menjadi pemodelan klasifikasi yang umum digunakan. *Naive Bayes* dapat dilatih dengan sangat efisien karena prosesnya hanya menghitung probabilitas bersyarat masing-masing atribut dan mencari probabilitas tertinggi. *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.

(b) *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 pemodelan *k-means*.
- *initialModel* adalah menentukan nilai *centroid* saat dilakukan inisialisasi.

#### 4. Melakukan instalasi dan konfigurasi Spark pada cluster Hadoop

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** Spark berjalan pada sistem operasi Windows, Linux, dan Mac OS. Spark dapat dijalankan secara lokal menggunakan satu komputer, meskipun Spark tetap membutuhkan beberapa komputer untuk pemrosesan data yang besar. Jenis instalasi Spark telah dijelaskan pada bagian sebelumnya. Pada penelitian ini digunakan jenis instalasi *Standalone* untuk Spark versi 2.4.5 pada sistem operasi Windows. 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

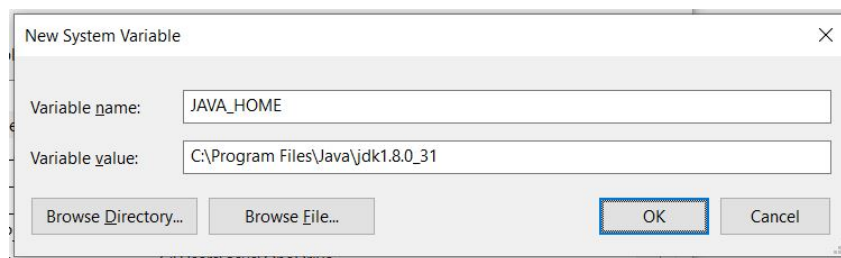
Berikut adalah tahapan instalasi Spark 2.4.5 secara umum:

- Melakukan instalasi Java 8.
- Melakukan instalasi Spark 2.4.5
- Melakukan instalasi IntelliJ untuk Scala sbt.

##### Instalasi Java 8

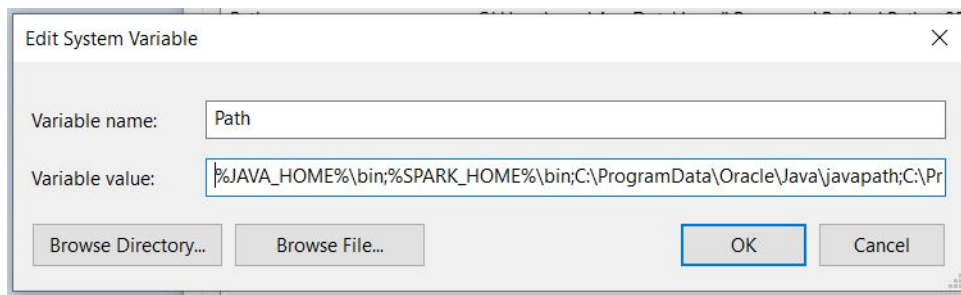
Berikut adalah tahapan instalasi Java 8 secara lengkap:

- Download Java SE Development Kit 8u31 pada link berikut  
<https://www.oracle.com/technetwork/java/javase/downloads/java-archive-javase8-2177648.html>
- Lakukan instalasi Java SE Development Kit 8u31 seperti biasa.
- Pilih menu *Edit the system environment variables*.
- Buat *environment variables* baru seperti Gambar 20.



Gambar 20: Environment Variables

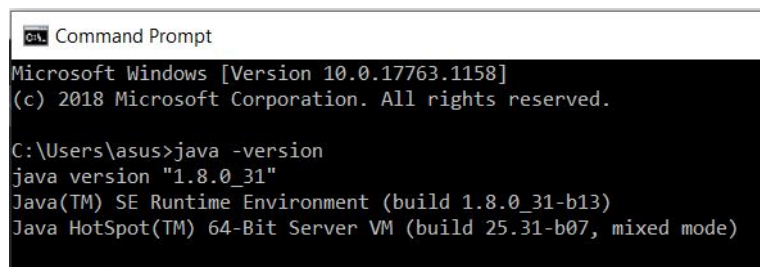
- Tambahkan `%JAVA_HOME%\bin` pada Path di System variables seperti Gambar 24.



Gambar 21: Penambahan Variable Value

Berikut adalah tahapan verifikasi terhadap instalasi Java 8:

- (a) Pilih menu Command Prompt.
- (b) Jalankan perintah `java -version` pada Command Prompt.

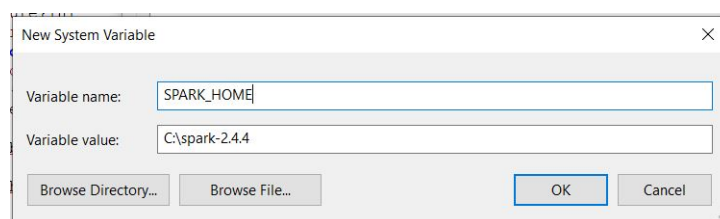
Gambar 22: Perintah `java -version`

- (c) Apabila sistem tidak menampilkan pesan error, maka Java 8 sudah terpasang dengan baik.

### Instalasi Spark 2.4.5

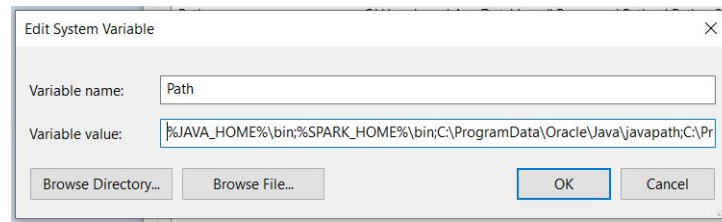
Berikut adalah tahapan instalasi Spark 2.4.5 secara lengkap:

- (a) Download `winutils.exe` dari link <https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin>, tempatkan `winutils.exe` pada `C:/winutils/bin`
- (b) 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>
- (c) Buat folder sebagai berikut `C:\spark-2.4.4` dan ekstraksi *file* `spark-2.4.5-bin-hadoop2.7.tgz` di dalam folder tersebut.
- (d) Buat *environment variables* baru seperti Gambar 23.



Gambar 23: Environment Variable

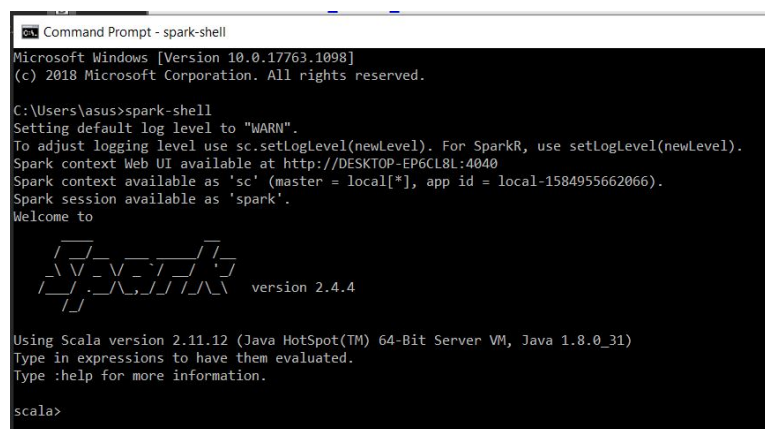
- (e) Tambahkan `%SPARK_HOME%\bin` pada Path di System variables seperti Gambar 24



Gambar 24: Penambahan Variable Value

Berikut adalah tahapan verifikasi terhadap instalasi Spark 2.4.5:

- Jalankan perintah `spark-shell` pada Command Prompt.
- Apabila terminal menampilkan tampilan seperti pada Gambar 25, artinya Spark 2.4.5 sudah dapat berjalan dengan baik pada komputer tersebut.

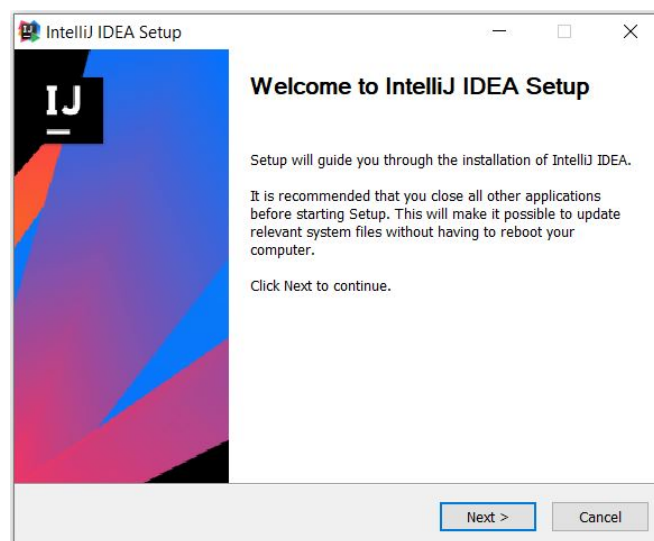


Gambar 25: Spark 2.4.5

### Instalasi IntelliJ untuk Scala SBT

Berikut adalah tahapan instalasi IntelliJ:

- Download IntelliJ melalui link berikut  
<https://www.jetbrains.com/idea/download/#section=windows>
- Lakukan instalasi IntelliJ seperti biasa.

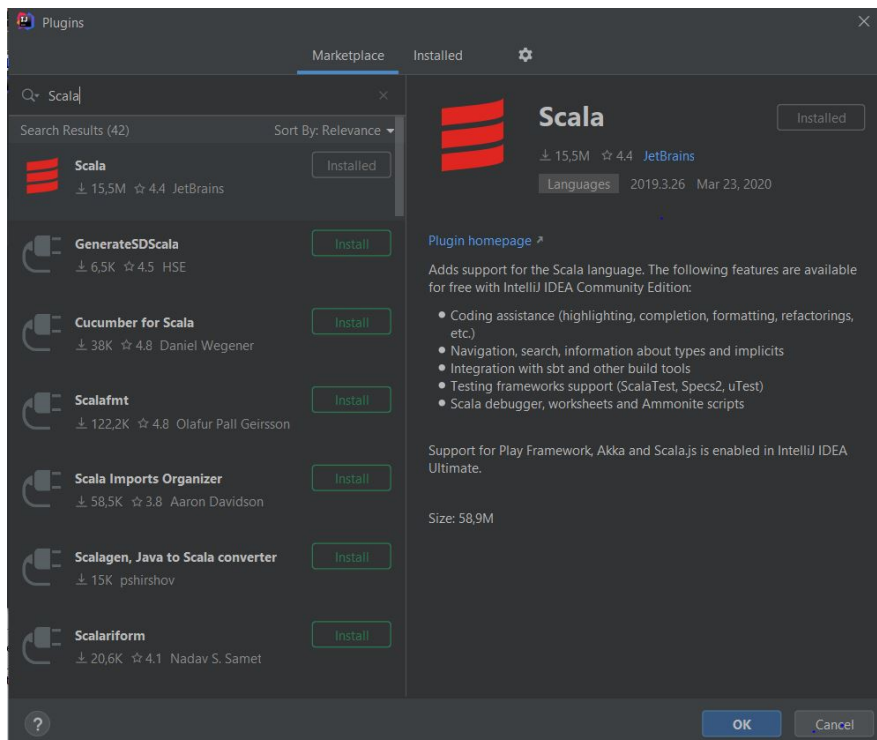


Gambar 26: Instalasi IntelliJ



Berikut adalah tahapan pemasangan *plugin* Scala pada IntelliJ.

- (a) Pilih menu *Configure* pada IntelliJ, lalu pilih menu *Plugins*.
- (b) Telusuri *plugin* Scala pada kolom pencarian seperti Gambar 27.



Gambar 27: Plugins Scala

- (c) Klik tombol *install*

## 5. Melakukan studi literatur mengenai bahasa pemrograman Scala pada *framework* Spark

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** 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 tersebut, Scala menjadi bahasa pemrograman yang sangat diminati untuk melakukan pemrosesan *big data* pada Spark.

### Eksperimen Scala

Scala digunakan pada penelitian ini karena sintaks yang sederhana untuk mengimplementasi beberapa baris kode pada bahasa pemrograman Java. Berikut adalah beberapa contoh eksperimen yang dilakukan pada bahasa Scala. Berikut adalah jenis eksperimen yang dilakukan pada 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 dapat dilihat pada Listing 1 mengenai jenis variabel pada Scala.

Listing 1: 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 dapat dilihat pada Listing 2 mengenai jenis tipe data pada Scala.

Listing 2: 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 dapat dilihat pada Listing 3 mengenai *immutable collection* pada Scala dan Listing 4 mengenai *mutable collection* pada Scala.

Listing 3: 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 4: 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 dapat dilihat pada Listing 5 mengenai kelas *object* pada Scala.

Listing 5: Membuat Kelas Object pada Scala

```
object DonutShoppingCartCalculator {
  val discount: Double = 0.01
  def calculateTotalCost(donuts: List[String]): Double = {
    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 dapat dilihat pada Listing 6 mengenai pembuatan fungsi pada Scala.

Listing 6: Membuat Fungsi Sederhana pada Scala

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

### Membuat Main Method

Scala menggunakan main method untuk melakukan eksekusi program. Kode program dapat dilihat pada Listing 7 mengenai pembuatan main method pada Scala.

Listing 7: Membuat Main Method pada Scala

```
object HelloWorld {
  def main(args: Array[String]) {
    println("Hellow, 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 dapat dilihat pada Listing 8 mengenai percabangan pada Scala.

Listing 8: 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 dapat dilihat pada Listing 9 mengenai perulangan pada Scala.

Listing 9: 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)
```

## 6. Melakukan studi dan eksplorasi teknik-teknik dasar data mining pada Spark MLlib.

**Status :** Ada sejak rencana kerja skripsi.

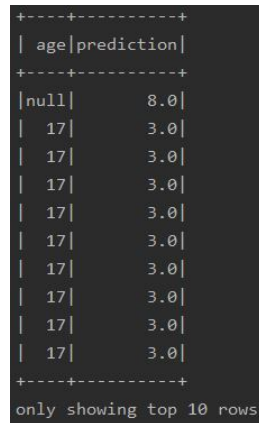
**Hasil :** Pada bagian sebelumnya telah dijelaskan mengenai konsep dan contoh pemodelan pada Spark MLlib. Pada penelitian ini akan digunakan pemodelan *Naive Bayes* untuk permasalahan klasifikasi dan *k-means* untuk permasalahan clustering.

### Naive Bayes

Berikut adalah tahapan eksperimen pada Listing 10 untuk pemodelan *Naive Bayes*:

- (a) Membagi data input CSV menjadi training data dan test data.
- (b) Melakukan pelatihan data pada pemodelan Naive Bayes.
- (c) Mengembalikan hasil klasifikasi dalam bentuk tabel.
- (d) Menghitung akurasi dari klasifikasi label kelas.

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 28, diketahui bahwa data dengan nilai umur yang sama yaitu (*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 28: Hasil Naive Bayes Spark MLlib

Listing 10: Eksperimen Naive Bayes Spark MLlib

```
// 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()

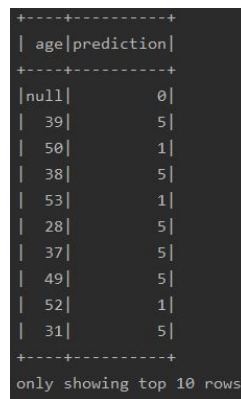
// Accuracy
val evaluator = new MulticlassClassificationEvaluator()
               .setLabelCol("workclass_Index")
               .setPredictionCol("prediction")
               .setMetricName("accuracy")
val accuracy = evaluator.evaluate(predictions)
```

## K-Means

Berikut adalah tahapan eksperimen pada Listing 11 untuk pemodelan *k-means*:

- Membuat model *k-means* menggunakan Spark MLlib
- Menentukan jumlah cluster (*k*) untuk pemodelan *k-means*.
- Melakukan pelatihan data pada pemodelan *k-means*.
- Mencari nilai *centroid* dari masing-masing *cluster*.
- Mengembalikan hasil *clustering* dalam bentuk tabel.

Hasil dari pemodelan *k-means* adalah prediksi jenis *cluster* berdasarkan sifat dari masing-masing data. Umumnya hasil pemodelan *k-means*, data-data yang memiliki perbedaan nilai atribut terkecil akan menjadi kelompok *cluster* yang sama. Pada Gambar 29, 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

Gambar 29: Hasil K-Means Spark MLlib

Listing 11: Eksperimen K-Means Spark MLlib

```
// 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)
```

## 7. Mencari dan mengumpulkan data studi kasus

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** Dataset yang dipakai adalah *Adult*. Dataset ini diperoleh dari website Kaggle. Dataset ini disimpan dalam format CSV. Format CSV memisahkan nilai atribut data melalui simbol koma. Dataset *Adult* 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, Cuba, <=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 30: 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, 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:  $\leq 50K$ ,  $> 50K$

### Studi Kasus Personally Identifiable Information

PII digunakan untuk mengelompokkan nilai atribut berdasarkan kategori atribut yang digunakan pada proses anonimisasi data. Berdasarkan penjelasan pada bagian sebelumnya, atribut pada proses anonimisasi dapat dikategorikan sebagai *identifier*, *quasi-identifier*, dan *sensitive attribute*.

Atribut *identifier* adalah atribut yang dapat mengidentifikasi individu secara langsung. Contoh dari atribut identifier pada dataset *Adult* adalah nama, tempat tanggal lahir, alamat rumah, nomor KTP. Atribut *quasi-identifier* adalah atribut yang dapat mengidentifikasi seseorang apabila nilai sebuah

atribut digabung dengan nilai atribut lain pada baris data yang sama. Contoh *quasi-identifier* pada dataset *Adult* adalah *age*, *zip*, *education*, *years of education*, *occupation*, *race*, *sex*, *native country*. *Sensitive attribute* adalah nilai yang ingin dirahasiakan. Contoh sensitive attribute pada dataset *Adult* adalah *workclass*, *marital status*, *relationship*, *income*.

Atribut *identifier* nantinya akan dihilangkan sebelum dilakukan proses anonimisasi, karena nilai dari atribut *identifier* dapat langsung mengidentifikasi seseorang. Sedangkan *sensitive attribute* nilainya tidak akan dihapus karena akan melalui proses anonimisasi bersamaan dengan nilai dari *quasi-identifier* sehingga *sensitive attribute* milik individu tidak dapat dibedakan satu sama lain pada hasil tabel akhir anonimisasi sehingga keamanan distribusi data terjamin.

### Studi Kasus Distance, Information Loss

Pada bagian sebelumnya, telah dijelaskan konsep mengenai penggunaan *distance* dan *information loss*. *Distance* dan *Information Loss* digunakan oleh algoritma *Greedy k-member clustering* untuk mencari kelompok data terbaik sehingga menghasilkan pengelompokan data yang tepat. Berikut adalah contoh dari masing-masing jenis perhitungan metrik:

(a) *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 *Adult* sebagai berikut:

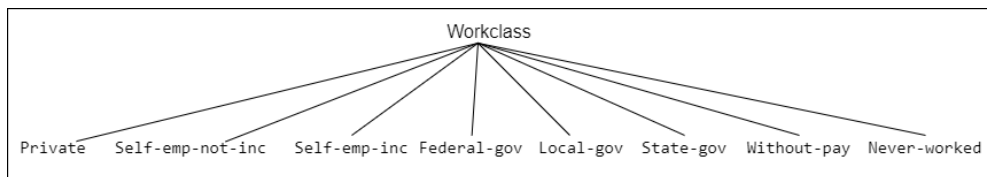
- i. 39, State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K
- ii. 50, Self-emp-not-inc, 83311, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 13, United-States, <=50K

*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 seperti pada Gambar 31.

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



Gambar 31: Taxonomy Tree (Workclass)



(b) *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. Tabel 1 adalah contoh hasil pengelompokan data pada *cluster 1* untuk dataset *Adult*:

Tabel 1: Tabel Hasil Clustering Data pada Cluster 1

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*.

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

**Studi Kasus Algoritma Greedy k-member clustering**

Algoritma *Greedy k-member clustering* telah dijelaskan pada bagian sebelumnya. Algoritma ini 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 untuk mencari tahu langkah kerja algoritma *Greedy k-member clustering* secara konseptual.

Tabel 2: Tabel Dataset Adults

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, 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*:

- (a) Nilai awal result =  $\emptyset$
- (b) Nilai awal  $r = \{t1\}$
- (c) Nilai awal  $|S| = 5$ ,  $k = 2$
- (d) Karena kondisi  $|S| \geq k$  terpenuhi, maka dilakukan perulangan sebagai berikut:
  - i. 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:

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

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

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

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

- ii. Nilai awal  $S = \{t1, t2, t4, t5\}$
- iii. Nilai awal  $c = \{t3\}$ ,  $|c| = 1$
- iv. Karena kondisi  $|c| < k$  terpenuhi, maka dilakukan perulangan sebagai berikut:
  - A. 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:

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

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

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

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

- B. Nilai  $S$  diubah menjadi  $S = \{t1, t2, t5\}$ ,  $|S| = 4$
- C. Nilai  $c$  ditambahkan menjadi  $c = \{t3, t4\}$ ,  $|c| = 2$
- v. Karena kondisi  $|c| < k$  sudah tidak terpenuhi lagi, maka perulangan ini akan berhenti
- vi. Nilai *result* akan ditambahkan menjadi  $result = \{t3, t4\}$
- vii. Karena kondisi  $|S| \geq k$  masih terpenuhi, maka perulangan akan tetap berlanjut sampai pada kondisi dimana  $|S| < k$  sehingga hasil akhirnya adalah  $result = \{\{t3, t4\}, \{t2, t5\}\}$ ,  $S = \{t1\}$ ,  $|S| = 1$
- (e) Karena kondisi  $S \neq 0$  terpenuhi, maka dilakukan perulangan sebagai berikut:
  - i. Nilai  $r$  diubah menjadi  $r = \{t1\}$
  - ii. Nilai  $S$  diubah menjadi  $S = \{\phi\}$ ,  $|S| = 0$
  - iii. 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:

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

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

- iv. Nilai  $c$  ditambahkan menjadi  $c = \{t1, t3, t4\}$
  - v. 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 variabel *result* agar menjaga masing-masing *cluster* hanya memiliki 2 anggota saja.
  - vi. Karena kondisi  $S \neq 0$  sudah tidak terpenuhi lagi, maka perulangan ini akan berhenti.
- (f) Hasil akhirnya adalah  $result = \{\{t3, t4\}, \{t2, t5\}\}$  dikembalikan sebagai output untuk algoritma *Greedy k-member clustering* seperti pada Tabel 3 sebagai berikut:

Tabel 3: Tabel Hasil Greedy K-Member Clustering

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

### Studi Kasus Domain Generalization Hierarchy

Pada bagian sebelumnya, telah dijelaskan konsep mengenai *Hierarchy Based Generalization*. DGH adalah contoh penerapan dari *Hierarchy Based Generalization*. DGH bertujuan untuk melindungi data dengan cara menerapkan metode generalisasi terhadap nilai atribut data yang bersifat unik, agar menjadi nilai yang lebih umum. Berikut adalah penerapan DGH terhadap dataset *Adult*.

Diketahui kemungkinan nilai unik atribut pada dataset *Adult* sebagai berikut:

- Age = {33,36,38,40,42,43,46,49}
- ZIP = {77516,77517,77526,77527}
- Sex = {Male,Female}

Untuk atribut ZIP, akan dibangun tiga jenis domain sebagai berikut:

- Domain dengan nilai kurang spesifik  
Domain ini dipilih apabila tujuannya adalah lebih mengutamakan hasil informasi yang diperoleh dengan cara melakukan sedikit anonimisasi pada nilai data. Contohnya atribut ZIP akan diubah menjadi {7751\*,7752\*} apabila satu digit terakhir memiliki nilai yang berbeda dan digit sisanya memiliki nilai yang sama.
- Domain dengan nilai yang lebih umum  
Domain ini dipilih apabila tujuannya adalah menyeimbangkan nilai informasi yang diperoleh dengan tingkat perlindungan data yang didapat dengan cara meningkatkan level anonimisasi nilai data. Contohnya atribut ZIP akan diubah menjadi {775\*\*} apabila kedua digit terakhir memiliki nilai yang berbeda dan digit sisanya memiliki nilai yang sama.
- Domain dengan nilai yang umum. Domain ini dipilih apabila tujuannya adalah mengutamakan perlindungan data. Biasanya jenis domain ini jarang dipilih, karena hasil anonimisasinya tidak dapat digunakan untuk proses data mining (memiliki nilai akurasi yang rendah apabila dilakukan pemodelan *data mining*). Contohnya atribut ZIP akan diubah menjadi {\*\*\*\*\*}

Untuk atribut *Age* dan *Sex* akan dibangun berdasarkan ketentuan berikut:

- Nilai atribut *Age* akan diubah menjadi rentang nilai. Contohnya nilai 33 diubah menjadi [30-39], karena 33 termasuk pada rentang nilai tersebut.

- Nilai atribut *Sex* termasuk nilai kategorikal, sehingga akan diubah menjadi nilai yang lebih umum. Contohnya nilai *Male/Female* diubah menjadi *Person* (bentuk umum).

### Studi Kasus Penerapan Anonimisasi

Pada bagian sebelumnya, telah dijelaskan konsep 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 sudah terlindungi sehingga publikasi data dapat dilakukan dengan aman. Tabel 4 adalah kelompok data yang dihasilkan oleh algoritma *Greedy k-member clustering*.

Tabel 4: Tabel Hasil Clustering

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:

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

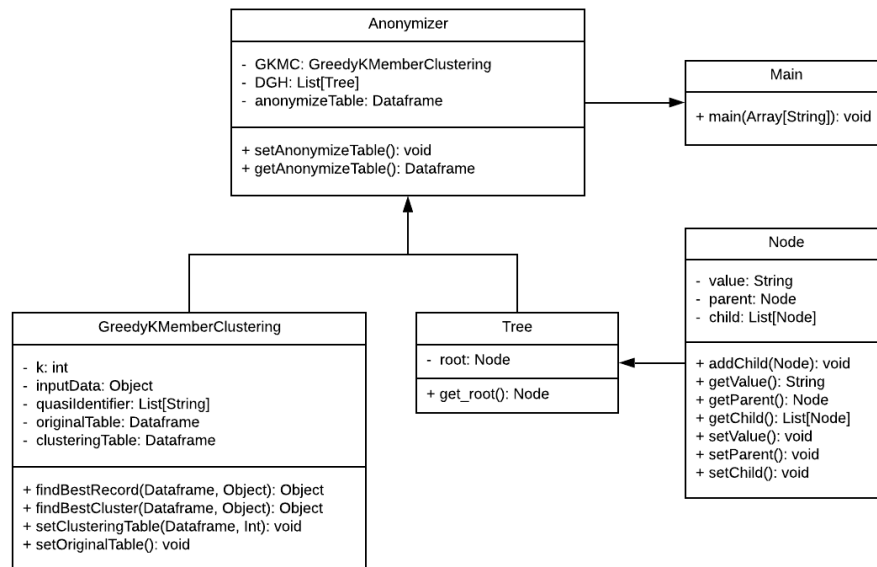
Tabel 5: Tabel Hasil Anonimisasi

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

## 8. Membuat diagram kelas perangkat lunak anonimisasi

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** 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 32 menggambarkan keterhubungan antar kelas pada perangkat lunak anonimisasi data. Berikut adalah penjelasan lengkap mengenai deskripsi kelas dan method pada perangkat lunak anonimisasi data:



Gambar 32: Diagram Kelas Anonimisasi Data

- Kelas *Anonymizer* bertujuan untuk melakukan proses anonimisasi setelah data dikelompokkan menjadi beberapa *cluster*. Kelas *Anonymizer* memiliki 2 jenis variabel, yaitu:
  - *GKMC* adalah objek dari kelas *GreedyKMemberClustering* yang berisi tabel hasil pengelompokan data berdasarkan algoritma *Greedy k-member clustering*.
  - *DGH* adalah array 1 dimensi dari objek *Tree* yang berisi hasil anonimisasi untuk nilai quasi-identifier yang unik agar menjadi nilai yang lebih umum.
  - *anonymizeTable* adalah array 2 dimensi dari kelas Object untuk menyimpan tabel hasil anonimisasi data.

Kelas *Anonymizer* memiliki 2 jenis method, yaitu:

- *setAnonymizeTable()* bertujuan untuk melakukan proses anonimisasi pada masing-masing baris data yang tergabung dalam sebuah *cluster*, berdasarkan perbedaan nilai dari beberapa *quasi-identifier*.
- *getAnonymizeTable()* bertujuan untuk mengambil nilai pada atribut *anonymizeTable*.
- Kelas *GreedyKMemberClustering* bertujuan untuk melakukan pengelompokan data menjadi beberapa *cluster* berdasarkan sifat/nilai atribut yang dimiliki oleh masing-masing baris data. Kelas *GreedyKMemberClustering* memiliki 5 jenis variabel, yaitu:
  - *k* adalah variabel bertipe *Integer* untuk membatasi jumlah anggota pada sebuah *cluster* agar memiliki jumlah yang tetap sebanyak jumlah tertentu.

- *inputData* adalah variabel untuk menyimpan seluruh baris data *file* CSV.
- *quasiIdentifier* adalah daftar dari nama-nama kolom yang akan dipilih untuk membuat tabel baru yang digunakan pada proses anonimisasi data
- *originalTable* adalah tabel yang menyimpan seluruh baris data pada *file* CSV berdasarkan jenis kolom yang terpilih pada variabel *quasiIdentifier*.
- *clusteringTable* adalah tabel yang menyimpan hasil pengelompokan baris data dari algoritma *Greedy k-member clustering*.

Kelas *GreedyKMemberClustering* memiliki 4 jenis method, yaitu:

- *findBestRecord()* bertujuan mencari sebuah baris data yang memiliki nilai *information loss* yang paling minimal dengan baris data lainnya.
  - *findBestCluster()* bertujuan mencari sebuah *cluster* data yang memiliki nilai *information loss* yang paling minimal dengan *cluster* lainnya.
  - *setClusteringTable()* bertujuan mengelompokkan data berdasarkan algoritma *Greedy k-member clustering* dan hasilnya disimpan pada variabel *clusteringTable*.
  - *setOriginalTable()* bertujuan mengubah hasil pembacaan data input CSV menjadi tabel baru dan hasilnya disimpan pada variabel *originalTable*.
- Kelas *Tree* bertujuan untuk membuat pohon generalisasi berdasarkan jenis atribut *quasi-identifier* yang dipilih.
  - Kelas *Node* bertujuan untuk menyimpan seluruh nilai *quasi-identifier* yang unik untuk masing-masing baris data.
  - Kelas *Main* bertujuan untuk membuat tahapan anonimisasi dari awal sampai akhir dengan memanfaatkan pemanggilan *method* dari masing-masing objek kelas.

## 9. Membuat diagram aktivitas perangkat lunak anonimisasi dan analisis

**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** 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 berdasarkan tujuan fungsionalitas yang berbeda.

### Perangkat Lunak Anonimisasi Data

Perangkat lunak anonimisasi adalah perangkat lunak yang digunakan untuk menyamarkan nilai data menggunakan metode anonimisasi. Perangkat lunak ini akan memiliki fungsi untuk menyamarkan beberapa nilai data agar sebuah data tidak dapat dibedakan dengan  $k - 1$  data lainnya berdasarkan konsep *k-anonymity*. Algoritma *Greedy k-member clustering* akan diimplementasikan pada Spark dengan fungsi utama yaitu menyamarkan nilai atribut data pada kelompok data yang sejenis. Berikut adalah tahapan yang terjadi pada perangkat lunak saat melakukan proses anonimisasi data:

- (a) Pengguna memberi masukan dalam format *file* CSV dan beberapa jenis atribut *quasi-identifier* untuk menjadi tabel input pada proses anonimisasi.
- (b) Perangkat lunak menampilkan sebagian baris data dari tabel input karena baris data yang akan digunakan pada eksperimen akan berjumlah sangat banyak .
- (c) Pengguna akan meninjau ulang apakah jumlah kolom yang ditampilkan sudah sesuai dengan jumlah atribut *quasi-identifier* yang akan dipakai.
- (d) Penggunaan memberikan parameter tambahan seperti rentang nilai  $k$  untuk menentukan jumlah anggota *cluster* dan objek DGH untuk proses anonimisasi.

- (e) Perangkat lunak akan melakukan proses anonimisasi dengan bantuan Spark pada tabel input berdasarkan paramater tambahan yang diberikan sebelumnya.
- (f) Perangkat lunak mengembalikan seluruh isi *log* yang dihasilkan selama proses eksekusi Spark berlangsung kepada pengguna untuk deteksi *error*.
- (g) Perangkat lunak hanya menampilkan baris data yang berubah akibat proses anonimisasi pada GUI dan hasil keseluruhannya dalam format *file* CSV.
- (h) Perangkat lunak mengembalikan nilai *information loss* pada masing-masing *cluster* yang terbentuk agar pengguna dapat mencari hasil yang optimal.
- (i) Pengguna dapat membandingkan hasil anonimisasi antara baris data yang berubah akibat proses anonimisasi dengan baris data yang ada pada tabel asli.
- (j) Pengguna dapat mengulangi eksperimen untuk mencari nilai *k* terbaik agar dihasilkan *information loss* seminimal mungkin pada proses anonimisasi.

### Perangkat Lunak Analisis Data

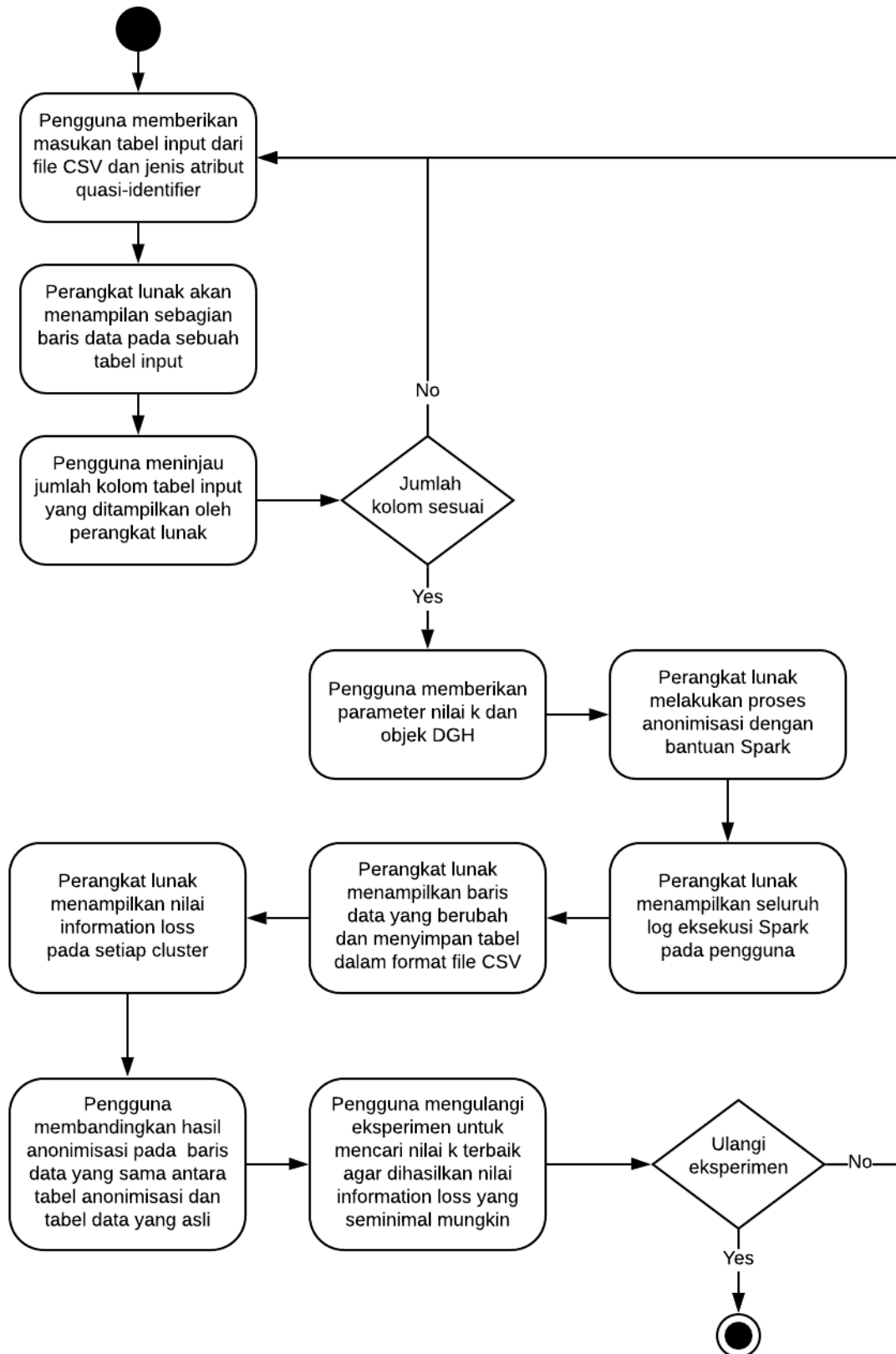
Perangkat lunak analisis data adalah perangkat lunak yang digunakan untuk melakukan pengujian *data mining* pada pemodelan klasifikasi dan *clustering*. Perangkat lunak ini akan memiliki fungsi untuk mencari label kelas berdasarkan pemodelan klasifikasi dan mengelompokkan data berdasarkan pemodelan *clustering*. Pemodelan *naive bayes* dan *k-means* akan diimplementasikan pada Spark dengan fungsi utama yaitu memprediksi kelompok data. Berikut adalah tahapan yang terjadi pada perangkat lunak saat melakukan pemodelan *data mining*:

- (a) Pengguna memberi dua jenis masukan yaitu data asli dan data hasil anonimisasi dalam format *file* CSV untuk menjadi tabel input pada proses analisis data.
- (b) Perangkat lunak hanya menampilkan sebagian baris data dari dua jenis tabel input karena input baris data pada eksperimen berjumlah sangat banyak
- (c) Pengguna meninjau kembali apakah jumlah kolom yang ditampilkan pada kedua jenis tabel memiliki jumlah kolom atribut yang sama.
- (d) Pengguna memilih jenis pemodelan data mining yang tersedia pada eksperimen, yaitu klasifikasi dengan *Naive Bayes* atau pengelompokan/*clustering* dengan *k-means*.
- (e) Pengguna mengisi parameter pada pemodelan yang dipilih. Contoh pada *k-means* adalah nilai *k* dan satu jenis atribut. Sedangkan pada *Naive Bayes* adalah persentase *training*, *testing* data dan satu jenis atribut.
- (f) Perangkat lunak akan melakukan proses pelatihan data pada Spark untuk menemukan klasifikasi/pengelompokan yang sesuai berdasarkan jenis pemodelan yang dipilih.
- (g) Perangkat lunak mengembalikan seluruh isi *log* yang dihasilkan selama proses eksekusi Spark berlangsung kepada pengguna untuk deteksi *error*.
- (h) Perangkat lunak menampilkan sebagian hasil prediksi *cluster* untuk masing-masing data dan menyimpan hasil keseluruhannya dalam format *file* CSV.
- (i) Pengguna melakukan analisis lebih lanjut terkait pengelompokan dan klasifikasi kelompok data yang terbentuk dari proses pemodelan *data mining*.

## 10. Menulis dokumen skripsi

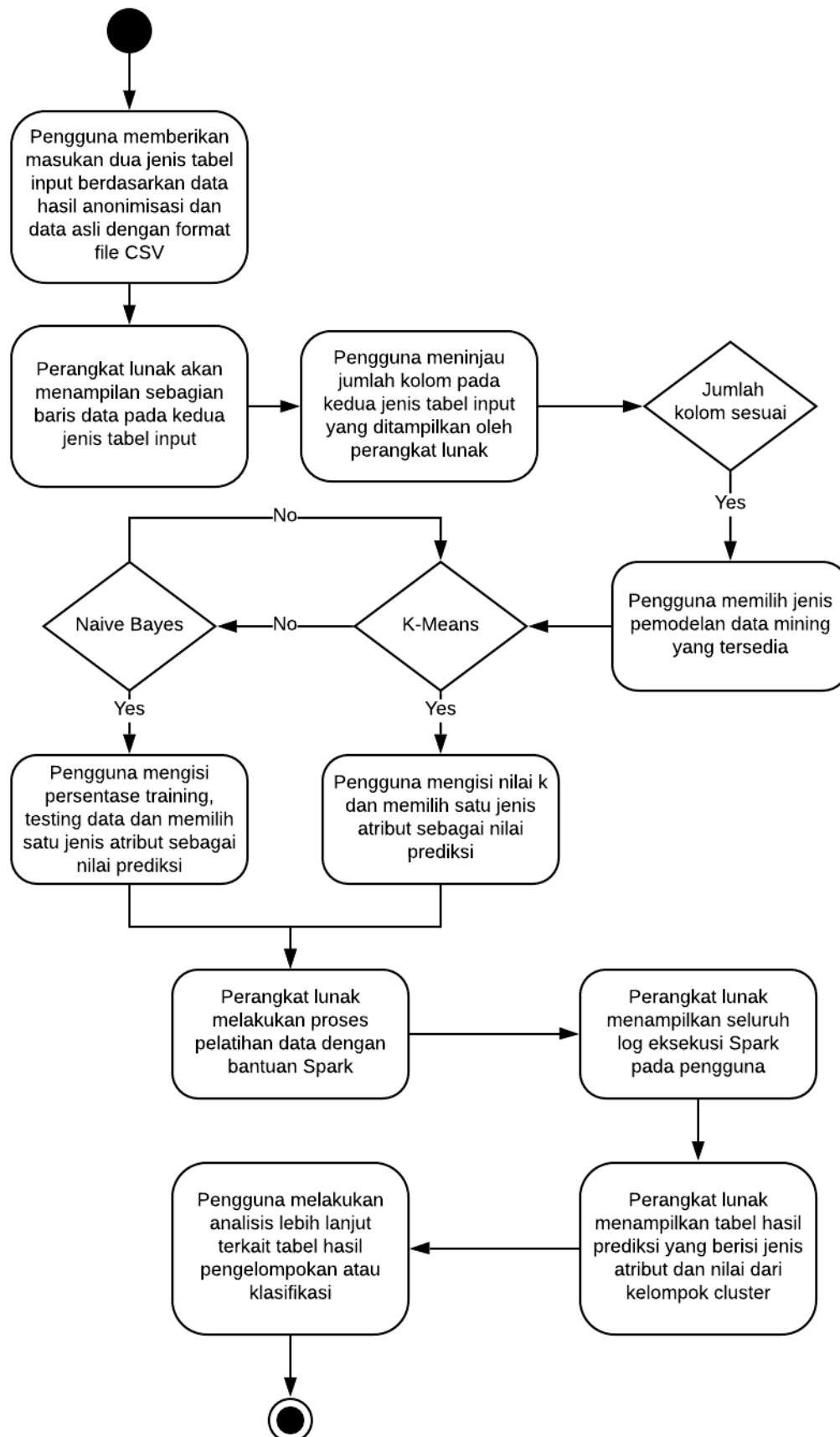
**Status :** Ada sejak rencana kerja skripsi.

**Hasil :** Penulisan dokumen skripsi sampai saat ini masih dalam tahap pengerjaan. Penyusunan bab yang telah dibuat pada skripsi ini tersusun dari bab 1 yang berisi pendahuluan, bab 2 yang berisi dasar teori, dan bab 3 yang berisi analisis masalah. Seluruh pengerjaan dari bab 1 sampai dengan bab 3 telah selesai dikerjakan dengan baik.



Gambar 33: Diagram Aktifitas Anonimisasi Data





Gambar 34: Diagram Aktifitas Analisis Data

## 6 Pencapaian Rencana Kerja

Langkah-langkah kerja yang berhasil diselesaikan dalam Skripsi 1 ini adalah sebagai berikut:

1. Mempelajari teknik-teknik dasar *data mining*.
2. Mempelajari algoritma *Greedy K-member clustering*.
3. Mempelajari konsep Hadoop, Spark, dan Spark MLlib.
4. Melakukan instalasi dan konfigurasi Spark pada *cluster* Hadoop.
5. Mempelajari bahasa pemrograman Scala pada *framework* Spark.
6. Melakukan studi dan eksplorasi teknik-teknik dasar *data mining* pada Spark MLlib.
7. Mencari dan mengumpulkan data studi kasus.
8. Menulis dokumen skripsi.

## 7 Kendala yang Dihadapi

Kendala - kendala yang dihadapi selama mengerjakan skripsi :

- Spark tidak dapat menangani data input XML.
- Skripsi ditempuh bersamaan dengan matakuliah lain.
- Pandemi COVID-19 di Indonesia mengganggu fokus pengerjaan skripsi.

Bandung, 02/05/2020

Stephen Jordan

Menyetujui,

Menyetujui,

Nama: Mariskha Tri Adithia, P.D.Eng  
Pembimbing Utama

Nama: Dr. Veronica Sri Moertini, Ir., M.T.  
Pembimbing Pendamping