

## SKRIPSI

### PENERAPAN ALGORITMA ANONIMISASI DATA PADA LINGKUNGAN BIG DATA



Stephen Jordan

NPM: 2016730018

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2020



**UNDERGRADUATE THESIS**

**APPLICATION OF DATA ANONYMIZATION ALGORITHM  
IN BIG DATA ENVIRONMENT**



**Stephen Jordan**

**NPM: 2016730018**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
2020**



## **LEMBAR PENGESAHAN**

### **PENERAPAN ALGORITMA ANONIMISASI DATA PADA LINGKUNGAN BIG DATA**

**Stephen Jordan**

**NPM: 2016730018**

**Bandung, «tanggal» «bulan» 2020**

**Menyetujui,**

**Pembimbing Utama**

**Pembimbing Pendamping**

**Mariskha Tri Adithia, P.D.Eng**

**Dr. Veronica Sri Moertini**

**Ketua Tim Penguji**

**Anggota Tim Penguji**

**«penguji 1»**

**«penguji 2»**

**Mengetahui,**

**Ketua Program Studi**

**Mariskha Tri Adithia, P.D.Eng**



## **PERNYATAAN**

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

### **PENERAPAN ALGORITMA ANONIMISASI DATA PADA LINGKUNGAN BIG DATA**

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal «tanggal» «bulan» 2020

Meterai Rp. 6000
---------------------

Stephen Jordan  
NPM: 2016730018



## ABSTRAK

Untuk menghasilkan keputusan bisnis yang berkualitas, diperlukan proses analisis sekumpulan data yang sangat banyak untuk mendapatkan hasil yang valid. Big data hadir sebagai teknologi penyimpanan dan pengolahan data yang hemat dan efisien untuk mengolah data yang berukuran besar. Beberapa teknologi big data yang umum untuk dipakai adalah Spark dan Hadoop. Data mining dapat membantu analisis pada big data untuk mencari karakteristik atau pola data. Sayangnya, proses data mining dapat menimbulkan masalah privasi. Konsep PPDM dipakai untuk melindungi privasi individu saat dilakukan proses data mining. Salah satu metode PPDM yang dapat dipakai adalah metode anonimisasi. Metode k-anonymity merupakan salah satu contoh metode anonimisasi. Karena metode anonimisasi menderita kehilangan informasi yang besar maka data akan dikelompokan terlebih dahulu menggunakan algoritma greedy k-member clustering. Tujuan dari penelitian ini adalah melakukan implementasi algoritma greedy k-member clustering dan k-anonymity pada lingkungan big data dan menguji model data mining klasifikasi dan clustering sebelum dan setelah data dilakukan anonimisasi.

Pada penelitian ini, dibangun tiga buah perangkat lunak dengan framework Spark. Pertama, perangkat lunak eksplorasi yang bertujuan mencari nilai unik sebuah kolom untuk dipakai dalam membuat pohon generalisasi. Kedua, perangkat lunak anonimisasi yang berisi implementasi algoritma greedy k-member clustering dan k-anonymity. Ketiga, perangkat lunak pengujian untuk mengamati hasil pemodelan data mining sebelum dan setelah data dilakukan anonimisasi. Hasil dari perangkat lunak anonimisasi pengujian dipakai untuk analisis. Analisis dilakukan dengan pengujian eksperimental. Pengujian eksperimental dilakukan untuk mendapatkan performa algoritma dan model data mining, menghitung total information loss, evaluasi hasil data mining, dan mencari perbedaan hasil prediksi model data mining terbaik.

Hasil pengujian kualitas informasi menunjukkan bahwa *total information loss* terendah ditemukan jika memakai kolom campuran, mengurangi jumlah quasi-identifier bertipe numerik, memakai ukuran data yang relatif kecil. Untuk waktu komputasinya, algoritma greedy k-member clustering membutuhkan waktu yang sangat lama untuk melakukan pengelompokan data, sedangkan algoritma k-anonymity dapat dilakukan komputasi dengan cepat. Saat dilakukan pengujian teknik data mining, dapat dilihat bahwa metode clustering memiliki perbedaan hasil pengelompokan yang cukup jauh antara sebelum dan setelah data dianonimisasi, sedangkan metode klasifikasi memiliki perbedaan hasil klasifikasi yang cukup dekat. Sehingga, pemodelan data mining yang cocok untuk anonimisasi data adalah klasifikasi.

**Kata-kata kunci:** Big Data, Data Mining, Privasi, Privacy Preserving Data Mining, K-Anonymity, Greedy K-Member Clustering



## ABSTRACT

In order to produce quality business decisions, it is necessary to process a large collection of data to obtain valid results. Big data exists as an efficient and efficient data storage and processing technology for processing large data. Some of the common big data technologies to use are Spark and Hadoop. Data mining can help analyze big data to look for data characteristics or patterns. Unfortunately, the data mining process can create privacy concerns. The PPDM concept is used to protect individual privacy during the data mining process. One of the PPDM methods that can be used is the anonymization method. The k-anonymity method is an example of an anonymization method. Because the anonymization method suffers from a large loss of information, the data will be grouped first using the greedy k-member clustering algorithm. The purpose of this study is to implement the greedy k-member clustering and k-anonymity algorithms in the big data environment and test the classification and clustering data mining model before and after the data is anonymized.

In this research, three software with the Spark framework were built. First, exploration software that aims to find the unique value of a column to be used in creating a generalization tree. Second, anonymization software which contains the implementation of the greedy k-member clustering and k-anonymity algorithms. Third, testing software to observe the results of data mining modeling before and after the data is anonymized. The results of the test anonymization software are used for analysis. The analysis was carried out by experimental testing. Experimental testing is done to get the performance of algorithms and data mining models, calculate total information loss, evaluate data mining results, and look for differences in the results of predictions of the best data mining models.

The results of the information quality test show that the lowest total information loss is found when using mixed columns, reducing the number of quasi-identifiers of numeric type, using a relatively small data size. For computation time, the greedy k-member clustering algorithm takes a very long time to group data, while the k-anonymity algorithm can be computed quickly. When testing data mining techniques, it can be seen that the clustering method has quite a large difference in the results of the grouping between before and after the data is anonymized, while the classification method has quite close differences in classification results. Thus, data mining modeling suitable for data anonymization is classification.

**Keywords:** Big Data, Data Mining, Privacy, Privacy Preserving Data Mining, K-Anonymity, Greedy K-Member Clustering



*«kepada siapa anda mempersembahkan skripsi ini. . . ?»*



## **KATA PENGANTAR**

«Tuliskan kata pengantar dari anda di sini . . . »

Bandung, «bulan» 2020

Penulis



# DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xxi</b>
<b>DAFTAR TABEL</b>	<b>xxv</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	3
1.3 Tujuan . . . . .	3
1.4 Batasan Masalah . . . . .	4
1.5 Metodologi . . . . .	4
1.6 Sistematika Pembahasan . . . . .	4
<b>2 LANDASAN TEORI</b>	<b>7</b>
2.1 Privasi . . . . .	7
2.2 <i>Data Mining</i> . . . . .	8
2.2.1 Klasifikasi . . . . .	9
2.2.2 <i>Naive Bayes</i> . . . . .	10
2.2.3 <i>Clustering</i> . . . . .	14
2.2.4 <i>K-Means</i> . . . . .	15
2.3 Tahapan Evaluasi <i>Data Mining</i> . . . . .	17
2.3.1 Menghitung Tingkat Akurasi untuk Model Klasifikasi . . . . .	17
2.3.2 Menghitung Koefisien <i>Silhouette</i> untuk Model <i>Clustering</i> . . . . .	18
2.4 <i>Privacy-Preserving Data Mining</i> (PPDM) . . . . .	18
2.4.1 Metode pada PPDM . . . . .	19
2.4.2 Metrik pada PPDM . . . . .	20
2.4.3 Model Serangan pada PPDM . . . . .	20
2.5 Anonimisasi . . . . .	20
2.6 <i>K-Anonymity</i> . . . . .	21
2.7 <i>Hierarchy Based Generalization</i> . . . . .	22
2.8 <i>Greedy K-Member Clustering</i> . . . . .	23
2.9 Metrik <i>Distance</i> dan <i>Information Loss</i> . . . . .	27
2.9.1 <i>Distance</i> . . . . .	27
2.9.2 <i>Information Loss</i> . . . . .	28
2.10 Sistem Terdistribusi . . . . .	29
2.10.1 Manfaat Sistem Terdistribusi . . . . .	29
2.10.2 Tantangan Sistem Terdistribusi . . . . .	29
2.11 Big Data . . . . .	30
2.12 Hadoop . . . . .	30
2.12.1 HDFS . . . . .	30

2.12.2	MapReduce . . . . .	31
2.13	Spark . . . . .	32
2.13.1	Ekosistem Spark . . . . .	32
2.13.2	Arsitektur Spark . . . . .	33
2.13.3	Jenis Instalasi pada Spark . . . . .	33
2.13.4	Resilient Distributed Datasets (RDD) . . . . .	34
2.13.5	<i>Dataframe</i> . . . . .	34
2.13.6	Komponen Spark . . . . .	35
2.14	Spark MLlib . . . . .	35
2.14.1	Machine Learning pada Spark MLlib . . . . .	35
2.14.2	Tipe Data pada Spark MLlib . . . . .	36
2.14.3	<i>Data Mining</i> pada Spark MLlib . . . . .	37
2.15	Scala . . . . .	37
2.16	Scala Swing . . . . .	37
2.16.1	Panel dan Layout . . . . .	38
2.16.2	Handling Event . . . . .	38
2.17	Format Penyimpanan Data . . . . .	38
2.17.1	CSV . . . . .	38
2.17.2	JSON . . . . .	39
<b>3</b>	<b>ANALISIS</b>	<b>41</b>
3.1	Analisis Masalah . . . . .	41
3.1.1	Dataset Eksperimen . . . . .	41
3.1.2	<i>Personally Identifiable Information</i> . . . . .	42
3.1.3	Perhitungan <i>Distance</i> dan <i>Information Loss</i> . . . . .	43
3.1.4	<i>Greedy K-Member Clustering</i> . . . . .	44
3.1.5	<i>Domain Generalization Hierarchy</i> . . . . .	46
3.1.6	<i>K-Anonymity</i> . . . . .	48
3.2	Eksplorasi Spark . . . . .	49
3.2.1	Instalasi Spark . . . . .	49
3.2.2	Membuat <i>Project</i> Spark pada IntelliJ . . . . .	53
3.2.3	Membuat <i>File JAR</i> pada Command Prompt . . . . .	55
3.2.4	Menjalankan Program Spark pada Komputer Lokal . . . . .	55
3.2.5	Menjalankan Program Spark pada Hadoop Cluster . . . . .	55
3.3	Studi Kasus . . . . .	56
3.3.1	Eksperimen Scala . . . . .	56
3.3.2	Eksperimen Spark . . . . .	59
3.3.3	Eksperimen Komponen Spark . . . . .	63
3.3.4	Eksperimen Spark MLIB . . . . .	68
3.4	Gambaran Umum Perangkat Lunak . . . . .	70
3.4.1	Diagram Aktifitas . . . . .	71
3.4.2	Diagram Kelas . . . . .	73
<b>4</b>	<b>PERANCANGAN</b>	<b>77</b>
4.1	Diagram Kelas Lengkap . . . . .	78
4.1.1	Diagram Package . . . . .	78
4.1.2	Diagram Kelas pada Package ExploratoryModel . . . . .	79
4.1.3	Diagram Kelas pada Package AnonymizationModel . . . . .	79
4.1.4	Diagram Kelas pada Package ExaminationModel . . . . .	91
4.2	Masukan Perangkat Lunak . . . . .	95
4.2.1	Masukan Perangkat Lunak Eksplorasi . . . . .	95
4.2.2	Masukan Perangkat Lunak Anonimisasi . . . . .	96

4.2.3	Masukan Perangkat Lunak Pengujian . . . . .	98
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>101</b>
5.1	Implementasi Antarmuka . . . . .	101
5.1.1	Komputer Lokal dengan IntelliJ . . . . .	101
5.1.2	Hadoop Cluster dengan Terminal Ubuntu . . . . .	114
5.2	Pengujian . . . . .	124
5.2.1	Pengujian Fungsional . . . . .	124
5.2.2	Pengujian Eksperimental . . . . .	129
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>141</b>
6.0.1	Kesimpulan . . . . .	141
6.0.2	Saran . . . . .	142
<b>A</b>	<b>KODE PROGRAM</b>	<b>143</b>
<b>B</b>	<b>HASIL EKSPERIMEN</b>	<b>145</b>



## DAFTAR GAMBAR

2.1	Tahapan pada KDD . . . . .	8
2.2	Contoh <i>Logistic Regression</i> . . . . .	9
2.3	Contoh <i>Hierarchical Clustering</i> . . . . .	14
2.4	Contoh <i>Partitional Clustering</i> . . . . .	15
2.5	<i>Privacy Preserving Data Mining</i> (PPDM) . . . . .	19
2.6	DGH dan VGH pada Atribut ZIP . . . . .	23
2.7	DGH dan VGH pada Atribut Race . . . . .	23
2.8	Sistem Terdistribusi . . . . .	29
2.9	Proses Komputasi pada MapReduce . . . . .	31
2.10	Ekosistem Spark . . . . .	32
2.11	Arsitektur Spark . . . . .	33
2.12	Arsitektur Spark . . . . .	33
2.13	Tahapan Pembelajaran Machine Learning . . . . .	35
2.14	Contoh Vektor Dense dan Sparse . . . . .	36
2.15	GUI Sederhana pada Scala Swing . . . . .	38
3.1	Taxonomy Tree (Workclass) . . . . .	43
3.2	DGH dan VGH pada atribut ZIP . . . . .	46
3.3	Environment Variables . . . . .	50
3.4	Penambahan Variable Value . . . . .	50
3.5	Perintah java -version . . . . .	50
3.6	Environment Variable . . . . .	51
3.7	Penambahan Variable Value . . . . .	51
3.8	Spark 2.4.5 . . . . .	51
3.9	Instalasi IntelliJ . . . . .	52
3.10	Plugins Scala . . . . .	52
3.11	Memilih Bahasa Scala Berbasis sbt . . . . .	53
3.12	Melakukan Konfigurasi Project Spark . . . . .	53
3.13	Menambahkan Scala Class pada Project Spark . . . . .	54
3.14	Memilih Tipe Object pada Scala Class . . . . .	54
3.15	Hasil Naive Bayes Spark MLlib . . . . .	69
3.16	Hasil K-Means Spark MLlib . . . . .	70
3.17	Flow Chart Penggunaan Perangkat Lunak . . . . .	71
3.18	Diagram Kelas Anonimisasi Data . . . . .	73
3.19	Diagram Aktifitas Anonimisasi Data . . . . .	75
3.20	Diagram Aktifitas Analisis Data . . . . .	76
4.1	Flow Chart Penggunaan Perangkat Lunak . . . . .	77
4.2	Flow Chart Penggunaan Perangkat Lunak . . . . .	77
4.3	Diagram Kelas pada Package . . . . .	78
4.4	Diagram Kelas pada Package ExploratoryModel . . . . .	79
4.5	Diagram Kelas pada Package AnonymizationModel . . . . .	80
4.6	Diagram Kelas pada ExaminationModel . . . . .	91

5.1	IntelliJ . . . . .	101
5.2	Main Class Perangkat Lunak Ekplorasi . . . . .	101
5.3	Konfigurasi Parameter Perangkat Lunak Ekplorasi . . . . .	102
5.4	Menjalankan Perangkat Lunak Ekplorasi . . . . .	102
5.5	Log Perangkat Lunak Ekplorasi . . . . .	103
5.6	Folder Output Perangkat Lunak Ekplorasi . . . . .	103
5.7	Tabel Nilai Unik (Age) . . . . .	103
5.8	Main Class Perangkat Lunak Ekplorasi . . . . .	104
5.9	Main Class Perangkat Lunak Ekplorasi . . . . .	104
5.10	Konfigurasi Parameter Perangkat Lunak Ekplorasi . . . . .	105
5.11	Menjalankan Perangkat Lunak Ekplorasi . . . . .	105
5.12	Log Perangkat Lunak Ekplorasi . . . . .	106
5.13	Folder Output Perangkat Lunak Anonimisasi . . . . .	106
5.14	Tabel Pengelompokan Data . . . . .	106
5.15	Main Class Perangkat Lunak Ekplorasi . . . . .	111
5.16	Main Class Perangkat Lunak Ekplorasi . . . . .	111
5.17	Konfigurasi Parameter Perangkat Lunak Ekplorasi . . . . .	111
5.18	Menjalankan Perangkat Lunak Ekplorasi . . . . .	112
5.19	Log Perangkat Lunak Ekplorasi . . . . .	112
5.20	Folder Output Perangkat Lunak Ekplorasi . . . . .	113
5.21	Tabel Nilai Unik (Age) . . . . .	113
5.22	Terminal Ubuntu . . . . .	114
5.23	Spesifikasi Slaves Node . . . . .	115
5.24	File Input Eksplorasi HDFS . . . . .	116
5.25	Log Perangkat Lunak Ekplorasi . . . . .	116
5.26	Folder HDFS Hasil Ekplorasi . . . . .	117
5.27	Folder HDFS Hasil Ekplorasi Atribut Race . . . . .	117
5.28	Hasil Eksplorasi pada Atribut Race . . . . .	117
5.29	File Input Anonimisasi HDFS . . . . .	118
5.30	Log Perangkat Lunak Anonimisasi . . . . .	119
5.31	Folder HDFS Hasil Pengelompokan Data . . . . .	119
5.32	Folder HDFS Hasil Anonimisasi Data . . . . .	120
5.33	Hasil Pengelompokan Greedy k-member clustering . . . . .	120
5.34	Hasil Anonimisasi K-Anonymity . . . . .	120
5.35	File Input Pengujian HDFS . . . . .	121
5.36	Log Perangkat Lunak Pengujian . . . . .	122
5.37	Folder HDFS Hasil Pengelompokan K-Means . . . . .	122
5.38	Folder HDFS Hasil Klasifikasi Naive Bayes . . . . .	123
5.39	Hasil Pengelompokan K-Means . . . . .	123
5.40	Hasil Klasifikasi Naive Bayes . . . . .	123
5.41	Hasil 1 . . . . .	132
5.42	Hasil 2 . . . . .	132
5.43	Hasil 1 . . . . .	132
5.44	Hasil 1 . . . . .	134
5.45	Hasil 2 . . . . .	134
5.46	Hasil 1 . . . . .	134
5.47	Hasil 1 . . . . .	136
5.48	Hasil 2 . . . . .	136
5.49	Hasil 1 . . . . .	136
5.50	Hasil 1 . . . . .	137
5.51	Hasil 2 . . . . .	137

5.52 Hasil 1 . . . . .	138
5.53 Hasil 2 . . . . .	138
5.54 Hasil 1 . . . . .	139
5.55 Hasil 2 . . . . .	139
5.56 Hasil 2 . . . . .	139
B.1 Hasil 1 . . . . .	145
B.2 Hasil 2 . . . . .	145
B.3 Hasil 3 . . . . .	145
B.4 Hasil 4 . . . . .	145



## DAFTAR TABEL

2.1	Contoh Kasus <i>PlayGolf</i> . . . . .	11
2.2	Tabel Probabilitas pada Atribut <i>Outlook</i> . . . . .	11
2.3	Tabel Probabilitas dari Atribut Temperature . . . . .	13
2.4	Tabel Probabilitas dari Atribut Humidity . . . . .	13
2.5	Tabel Probabilitas dari Atribut Wind . . . . .	13
2.6	Tabel Dataset Mata Pelajaran . . . . .	15
2.7	Hasil Pengelompokan Awal . . . . .	16
2.8	Mencari Centroid Kelompok . . . . .	16
2.9	Hasil Cluster Baru . . . . .	17
2.10	Euclidean Distance Cluster 1, Cluster 2 . . . . .	17
2.11	Hasil Pengelompokan Akhir . . . . .	17
3.1	Tabel Hasil Clustering Data pada Cluster 1 . . . . .	43
3.2	Dataset Adults . . . . .	44
3.3	Tabel Hasil Greedy K-Member Clustering . . . . .	46
3.4	Tabel Hasil Clustering . . . . .	48
3.5	Tabel Hasil Anonimisasi . . . . .	48
5.1	Pengujian Kualitas Informasi . . . . .	129
5.2	Pengujian Hasil Data Mining . . . . .	129
5.3	Konfigurasi Pengujian . . . . .	130
5.4	Konfigurasi Pengujian . . . . .	130
5.5	Sampel Data Credit Score(Numerik) . . . . .	131
5.6	Sampel Data Credit Score(Kategorikal) . . . . .	131
5.7	Sampel Data Credit Score(Campuran) . . . . .	131
5.8	Sampel Data Credit score ( $ QID =2$ ) . . . . .	133
5.9	Sampel Data Credit score ( $ QID =3$ ) . . . . .	133
5.10	Sampel Data Credit score ( $ QID =4$ ) . . . . .	133
5.11	Sampel Data Credit score (10k dan 30k) . . . . .	135
5.12	Kesimpulan Pengujian Eksperimental . . . . .	140



# BAB 1

## PENDAHULUAN

### 1.1 Latar Belakang

Untuk menghasilkan keputusan bisnis yang berkualitas, perusahaan memerlukan pengetahuan yang diperoleh melalui proses analisis data yang sangat banyak agar mendapatkan hasil yang valid. Biasanya data dikumpulkan dari berbagai platform seperti e-commerce, sosial media, maupun produk transaksi digital lainnya. Data yang disimpan dapat berupa data terstruktur, semi terstruktur, dan tidak terstruktur. Data terstruktur adalah data yang disimpan dalam bentuk tabel, baris dan kolom, contohnya *file excel* atau *spreadsheet*. Data semi-terstruktur adalah data yang disimpan dengan format tertentu, contohnya *file CSV*, *JSON*. Data tidak terstruktur adalah data yang tidak memiliki format penyimpanan nilai, contohnya *file text*. Data-data ini nantinya disimpan menggunakan teknologi basis data untuk di analisis lebih lanjut. Data yang dikumpulkan dapat memiliki kapasitas penyimpanan yang besar, sehingga proses analisis data menjadi sangat lambat. Dampak yang ditimbulkan dari pertumbuhan data menyebabkan basis data konvensional menjadi kurang efektif untuk pengolahan data. Oleh karena itu, teknologi *big data* digunakan untuk mengurangi biaya yang dikeluarkan untuk penyimpanan dan komputasi data, sehingga kapasitas data dapat ditambah dan data berukuran besar dapat lebih mudah untuk diolah.

Menurut Gartner, salah satu perusahaan riset teknologi informasi di Amerika Serikat, *big data* adalah adalah aset informasi bervolume tinggi, cepat, dan beragam yang menuntut bentuk pemrosesan informasi yang hemat biaya dan inovatif untuk meningkatkan wawasan dan pengambilan keputusan<sup>1</sup>. *Big data* disimpan, diolah, dan dilakukan analisis agar menghasilkan informasi yang bermanfaat sebagai dasar pengambilan keputusan atau kebijakan yang lebih tepat berdasarkan data sebenarnya. Karena *Big data* memiliki ukuran data yang besar, maka proses analisis *big data* harus dilakukan secara paralel. Caranya adalah dengan membagi data ke beberapa komputer untuk diolah masing-masing komputer tersebut. Konsep ini disebut dengan sistem terdistribusi. Sistem terdistribusi adalah solusi pengolahan *big data* karena terbukti dapat mengurangi biaya penyimpanan dan komputasi data dari pemrosesan data secara paralel.

Untuk melakukan proses analisis data, diperlukan teknik untuk mencari tahu kesamaan sifat yang dimiliki oleh sekumpulan data. Salah satu teknik yang dapat digunakan adalah *data mining*. Menurut Gartner, *Data mining* adalah proses menemukan korelasi, pola, dan tren yang bermakna menggunakan teknologi pengenalan pola, teknik statistik dan matematika dengan memilah-milah data dalam jumlah besar<sup>2</sup>. Teknik *data mining* dapat membantu proses analisis data pada lingkungan *big data*. Pemodelan data mining untuk *big data* dijalankan pada sistem terdistribusi, sehingga waktu komputasi dapat diminimalkan. Hasil *data mining* dapat menimbulkan masalah privasi. Privasi adalah hak seseorang untuk memiliki kendali atas bagaimana informasi pribadi dikumpulkan dan digunakan<sup>3</sup>. Privasi seseorang dapat terlanggar apabila sebuah perusahaan meminta data transaksi dari perusahaan lain dan secara tidak sengaja mengetahui informasi pribadi seseorang setelah dilakukan teknik *data mining* pada sekumpulan data masih banyak mengandung data privat.

---

<sup>1</sup><https://www.gartner.com/en/information-technology/glossary/big-data>

<sup>2</sup><https://www.gartner.com/en/information-technology/glossary/data-mining>

<sup>3</sup><https://iapp.org/about/what-is-privacy/>

PPDM merupakan bagian dari data mining yang bertanggung jawab atas perlindungan privasi dalam proses data mining. Oleh karena itu, Privacy Preserving Data Mining (PPDM) berperan penting untuk memberi perlindungan privasi dalam proses *data mining*.

Konsep PPDM dapat dicapai dengan metode enkripsi dan anonimisasi. Menurut Gartner, enkripsi adalah proses pengkodean aliran bit secara sistematis sebelum dilakukan transmisi sehingga pihak yang tidak berwenang tidak dapat mengetahui arti pesan sebenarnya<sup>4</sup>. Anonimisasi adalah metode yang menyamarkan satu atau lebih nilai atribut data agar data seseorang tidak dapat saling dibedakan dengan data lainnya. Kekurangan metode enkripsi adalah waktu komputasi yang lebih lama untuk membuat kunci dan biasanya data hasil enkripsi memiliki ukuran lebih besar dari data asli. Di satu sisi metode anonimisasi lebih unggul karena tidak perlu membuat kunci untuk menjaga privasi data. Metode anonimisasi dapat diterapkan pada kasus analisis data kartu kredit untuk mencari tahu karakteristik klien yang berpotensi untuk diberi kartu pinjaman kredit. Data kartu kredit dipilih karena masih memiliki kolom-kolom yang mengandung informasi pribadi seperti gaji, jenis pekerjaan, jenis pendidikan, kategori tempat tinggal, dan status keluarga.

Umumnya data yang disamarkan dengan metode anonimisasi menderita kehilangan informasi yang cukup banyak. Istilah ini lebih sering dikenal dengan *total information loss*. Hal ini mengakibatkan data yang telah dianonimisasi memiliki hasil prediksi yang lebih rendah ketika menggunakan model data mining klasifikasi/clustering. Salah satu cara untuk mencegah hal tersebut adalah dengan melakukan pengelompokan data terlebih dahulu sebelum dilakukan anonimisasi. Contoh algoritma pengelompokan data yang ingin diuji adalah greedy k-member clustering. Algoritma ini dipilih karena mencari solusi paling optimal dari seluruh kemungkinan pengelompokan yang ada. Metode anonimisasi yang digunakan adalah k-anonymity. Metode ini menjaga sebuah data tidak dapat dibedakan dengan  $k - 1$  data lainnya. Karena penelitian berkaitan dengan big data dan membutuhkan implementasi algoritma secara iteratif maka diperlukan teknologi untuk menangani komputasi secara paralel untuk kinerja yang lebih efisien. Spark merupakan pilihan yang tepat karena dapat memanfaatkan komputasi memori sehingga memiliki komputasi yang lebih cepat dalam implementasi iteratif maupun menganalisis big data.

Framework Spark dan Hadoop sering digunakan untuk melakukan komputasi pada lingkungan big data. Hadoop adalah framework yang memungkinkan pemrosesan terdistribusi dari kumpulan data besar di seluruh cluster komputer. Hadoop memiliki sistem penyimpanan yang tersebar di beberapa komputer dengan nama HDFS. Spark adalah mesin analitik terpadu untuk pemrosesan data skala besar. Spark cocok digunakan jika membutuhkan implementasi pemrograman iteratif. Penggunaan Spark dipilih karena Hadoop memiliki waktu pemrosesan *big data* yang lebih lama dari Spark karena melakukan komputasi pada *hardisk*, sedangkan Spark dapat 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 menfasilitasi pemodelan *data mining* yaitu klasifikasi dan pengelompokan/*clustering*. Kekurangan dari Spark adalah tidak memiliki penyimpanan yang tetap, sehingga membutuhkan mekanisme penyimpanan Hadoop, agar hasil pemrosesan data dapat tersimpan dalam *hardisk* komputer.

Pada skripsi ini, akan dibuat tiga jenis perangkat lunak yaitu perangkat lunak eksplorasi, perangkat lunak anonimisasi, dan perangkat lunak pengujian. Perangkat lunak eksplorasi bertujuan mencari nilai unik untuk pembuatan pohon binary tree yang digunakan ketika menghitung jarak terdekat antar data kategorikal. Perangkat lunak anonimisasi bertujuan mengimplementasikan proses pengelompokan data dengan algoritma *Greedy k-member clustering* dan proses anonimisasi data dengan metode *k-anonymity*. Perangkat lunak pengujian bertujuan membandingkan kualitas hasil data mining sebelum dan setelah data dianonimisasi. Ketiga perangkat lunak ini dibuat dengan bahasa Scala, berjalan di atas Spark, dan menggunakan penyimpanan HDFS untuk menyimpan hasil komputasi sementara dari algoritma greedy k-member clustering. Algoritma *Greedy k-member clustering* dinilai tepat melakukan pengelompokan data karena terbukti pada penelitian sebelumnya

---

<sup>4</sup><https://www.gartner.com/en/information-technology/glossary/encryption>

dapat meminimalkan *total information loss*. Kedua jenis perangkat lunak ini menerima data input dalam format CSV. Penelitian ini memiliki tujuan utama yaitu membandingkan silhouette score untuk model clustering, tingkat akurasi untuk model klasifikasi, mencari parameter terbaik pada model clustering dan klasifikasi, dan mencari perbedaan hasil clustering dan klasifikasi sebelum dan setelah dilakukan proses anonimisasi data.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah pada skripsi ini adalah sebagai berikut:

1. Bagaimana cara kerja algoritma *Greedy k-member clustering* untuk pengelompokan data?
2. Bagaimana cara kerja algoritma *k-anonymity* untuk anonimisasi data?
3. Bagaimana implementasi algoritma *Greedy k-member clustering* pada Spark?
4. Bagaimana performa algoritma *Greedy k-member clustering* dan *k-anonymity* untuk lingkungan big data pada komputer cluster?
5. Bagaimana performa pemodelan data mining clustering (k-means) dan klasifikasi (naive bayes) untuk lingkungan big data pada komputer lokal?
6. Bagaimana kualitas informasi pada data yang telah dikelompokan dengan algoritma *Greedy k-member clustering* berdasarkan total information loss?
7. Bagaimana analisis kualitas hasil data mining terhadap pemodelan clustering sebelum dan setelah dilakukan anonimisasi?
8. Bagaimana analisis kualitas hasil data mining terhadap pemodelan klasifikasi sebelum dan setelah dilakukan anonimisasi?

## 1.3 Tujuan

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

1. Mempelajari cara kerja algoritma *Greedy k-member clustering* untuk pengelompokan data.
2. Mempelajari cara kerja algoritma *k-anonymity* untuk anonimisasi data.
3. Mengimplementasi algoritma *Greedy k-member clustering* dan *k-anonymity* pada Spark.
4. Menganalisis performa algoritma *Greedy k-member clustering* dan *k-anonymity* untuk lingkungan big data pada komputer cluster.
5. Menganalisis performa pemodelan data mining clustering (k-means) dan klasifikasi (naive bayes) untuk lingkungan big data pada komputer lokal.
6. Menganalisis kualitas informasi pada data yang telah dikelompokan dengan algoritma *Greedy k-member clustering* berdasarkan total information loss.
7. Menganalisis kualitas hasil metode data mining clustering berdasarkan silhouette score, mencari parameter clustering terbaik, dan mencari persentase perbedaan hasil clustering terbaik sebelum dan setelah dilakukan anonimisasi.
8. Menganalisis kualitas hasil metode data mining klasifikasi berdasarkan tingkat akurasi, mencari parameter klasifikasi terbaik, dan mencari persentase perbedaan hasil klasifikasi terbaik sebelum dan setelah dilakukan anonimisasi.

## 1.4 Batasan Masalah

Batasan masalah pada penggerjaan skripsi ini adalah sebagai berikut:

1. Perangkat lunak hanya menerima masukan dalam format JSON. Format masukan JSON untuk masing-masing perangkat lunak berbeda dan dapat diisi sesuai kebutuhan. Contoh format JSON untuk masing-masing perangkat lunak telah dicatatumkan pada Subbab 4.3
2. Perangkat lunak hanya dapat mengeluarkan output dalam forma CSV. Folder penyimpanan output perangkat lunak dapat diubah pada file input JSON.
3. Perangkat lunak anonimisasi dapat melakukan pengelompokan dan anonimisasi data hingga 100.000 data dengan waktu komputasi selama 2 hari. Disarankan untuk menggunakan jumlah data dibawah 10.000 data untuk mendapat estimasi waktu komputasi kurang dari 3 jam.

## 1.5 Metodologi

Bagian-bagian penggerjaan skripsi ini adalah sebagai berikut:

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

## 1.6 Sistematika Pembahasan

Pengerjaan skripsi ini tersusun atas enam bab sebagai berikut:

- Bab 1 Pendahuluan  
Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
- Bab 2 Landasan Teori  
Berisi landasan teori mengenai konsep privasi, teknik *data mining*, *privacy-preserving data mining*, *k-anonymity*, algoritma *greedy k-member clustering*, metrik *distance* dan *information loss*, teknologi *big data*, pemrograman scala, dan format penyimpanan data.

- Bab 3 Analisis

Berisi analisis penelitian mengenai analisis masalah (dataset eksperimen, *personally identifiable information*, perhitungan *distance* dan *information loss*, algoritma *greedy k-member clustering*, *k-anonymity*, *domain generalization hierarchy*), eksplorasi spark (instalasi spark, pembuatan *project spark*, menjalankan program spark), studi kasus (eksperimen scala, eksperimen spark), dan gambaran umum perangkat lunak (diagram kelas dan diagram aktivitas).

- Bab 4 Perancangan

Berisi perancangan antarmuka perangkat lunak anonimisasi data dan analisis data, diagram kelas lengkap, masukan perangkat lunak anonimisasi data dan analisis data.

- Bab 5 Implementasi dan Pengujian

Berisi implementasi perangkat lunak anonimisasi data dan analisis data, pengujian fungsional, pengujian eksperimental, dan melakukan analisis terhadap hasil pengujian.

- Bab 6 Kesimpulan dan Saran

Berisi kesimpulan penelitian dan saran untuk penelitian selanjutnya.



## BAB 2

### LANDASAN TEORI

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

#### 2.1 Privasi

Privasi adalah hak untuk dibiarkan sendiri, atau bebas dari gangguan atau gangguan<sup>1</sup>. Informasi privasi adalah hak untuk memiliki kendali atas bagaimana informasi pribadi Anda dikumpulkan dan digunakan. Privasi dapat berarti kemampuan satu atau sekelompok individu untuk menutupi atau melindungi kehidupan dan urusan personalnya dari publik dengan mengontrol sumber-sumber informasi mengenai diri mereka. Untuk melakukan publikasi data dari satu perusahaan ke perusahaan lain, digunakan teknik anonimisasi data untuk melindungi dan menyamarkan atribut sensitif untuk setiap data.

*Personally Identifiable Information* (PII) adalah standar yang digunakan untuk menentukan apakah informasi yang ada dapat melakukan identifikasi entitas individu secara langsung atau tidak langsung. PII menjelaskan bahwa identifikasi entitas secara langsung dapat dilakukan menggunakan atribut sensitif. Sedangkan identifikasi entitas secara tidak langsung dapat dilakukan menggunakan penggabungan beberapa atribut non-sensitif. PII adalah atribut yang biasanya terjadi pelanggaran data dan pencurian identitas. Jika data perusahaan atau organisasi terungkap, maka sangat mungkin data pribadi seseorang akan terungkap. Informasi yang diketahui dapat dijual dan digunakan untuk melakukan pencurian identitas, menempatkan korban dalam risiko.

Berikut adalah contoh informasi yang bersifat sensitif menurut standar PII:

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

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

- Rekaman medis
- Riwayat pendidikan
- Riwayat pekerjaan

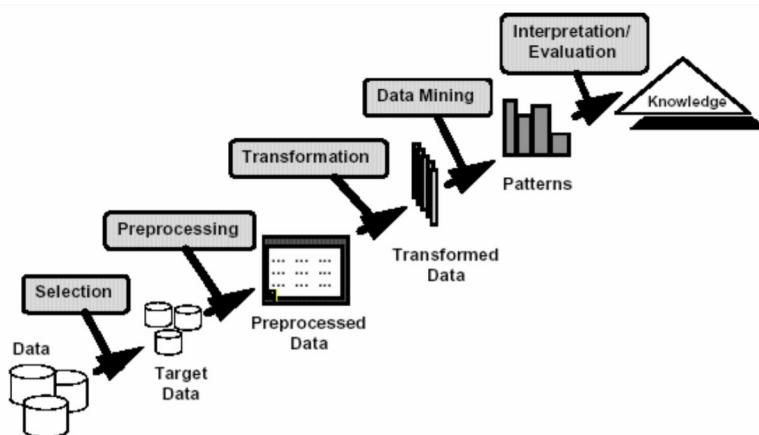
---

<sup>1</sup><https://iapp.org/about/what-is-privacy/>

- Informasi finasial
- Letak geografis

## 2.2 Data Mining

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



Gambar 2.1: Tahapan pada KDD

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

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

Berikut adalah beberapa jenis tipe data terkait teknik data mining:

- *Binary*: tipe data alphabet/numerik yang hanya memiliki 2 kemungkinan nilai.  
Contoh: nilai true/false dan 0/1.
- *Nominal*: tipe data alphabet/numerik yang memiliki lebih dari 2 kemungkinan nilai.  
Contoh: warna kuning, hijau, hitam, merah.

Tujuan dari penggunaan teknik *data mining* adalah sebagai berikut:

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

### 2.2.1 Klasifikasi

Klasifikasi adalah proses menemukan model (atau fungsi) yang menggambarkan dan membedakan kelas data. Model klasifikasi diturunkan berdasarkan analisis terhadap sekumpulan data pelatihan, yaitu objek data yang label kelasnya diketahui. Model klasifikasi digunakan untuk memprediksi label kelas dari objek yang label kelasnya tidak diketahui.

Tahapan klasifikasi perlu diawali dengan analisis relevansi. Analisis relevansi bertujuan untuk mengidentifikasi atribut yang secara signifikan relevan dengan klasifikasi. Atribut tersebut akan dipilih untuk pembuatan model klasifikasi. Atribut lain yang tidak relevan tidak perlu disertakan dalam pembuatan model klasifikasi.[?]

Berikut adalah tahapan klasifikasi secara umum:

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

Berikut adalah kategori pemodelan klasifikasi:

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

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

Gambar 2.2: Contoh *Logistic Regression*

Contoh: *Logistic Regression*

Gambar 2.2 adalah penerimaan siswa pada sebuah Universitas, untuk mempertimbangkan *test score* dan *grades* terhadap keputusan seorang siswa diterima/tidak diterima.

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

Contoh: *Naive Bayes*

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

Berikut adalah contoh pemodelan klasifikasi yang umum digunakan:

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

### 2.2.2 *Naive Bayes*

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

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

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

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

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

Keterangan:

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

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

- Vektor fitur  
Vektor fitur adalah vektor yang mewakili nilai fitur untuk setiap baris dataset. Vektor fitur pada contoh kasus ini terdiri dari atribut *Outlook*, *Temperature*, *Humidity*, dan *Windy*.
- Vektor respon  
Vektor respon adalah nilai prediksi dalam bentuk label kelas untuk setiap baris data. Vektor respon pada contoh kasus ini diwakili oleh atribut *PlayGolf*.

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

Tabel 2.1: Contoh Kasus *PlayGolf*

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

1. Merepresentasikan teorema *Bayes* terhadap vektor fitur.

Berdasarkan dataset, teorema *Bayes* dapat diubah seperti berikut:

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

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

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

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

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

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

2. Menghitung probabilitas nilai dari sebuah atribut.

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

Tabel 2.2: Tabel Probabilitas pada Atribut *Outlook*

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

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

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

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

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

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

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

Tabel 2.3: Tabel Probabilitas dari Atribut Temperature

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

Tabel 2.4: Tabel Probabilitas dari Atribut Humidity

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

Tabel 2.5: Tabel Probabilitas dari Atribut Wind

	Yes	No	P(Yes)	P(No)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100	100

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

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

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

5. Melakukan normalisasi terhadap probabilitas bersyarat.

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

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

Ketika probabilitas tersebut dijumlahkan, maka hasilnya akan menjadi 1.

$$P(Yes|today) + P(No|today) = 1 \quad (2.6)$$

6. Mencari probabilitas tertinggi.

Berdasarkan pernyataan berikut:

$$P(Yes|today) > P(No|today) \quad (2.7)$$

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

### 2.2.3 Clustering

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

Berikut adalah tahapan *clustering* secara umum:

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

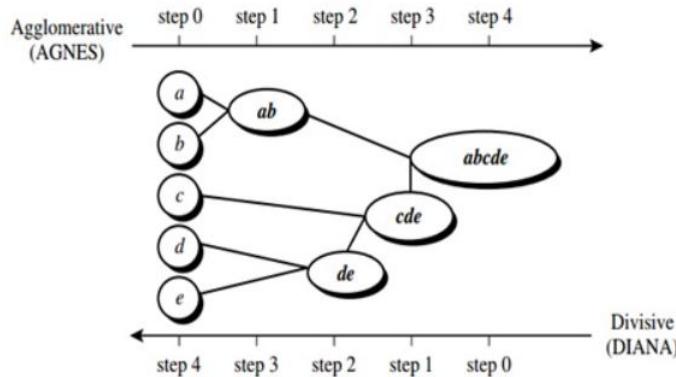
Berikut adalah persamaan untuk menghitung *Euclidean distance*:

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

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

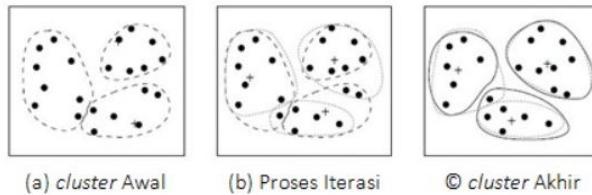
Berikut adalah kategori pemodelan *clustering*:

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



Gambar 2.3: Contoh *Hierarchical Clustering*

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



Gambar 2.4: Contoh *Partitional Clustering*

Berikut adalah contoh pemodelan *clustering* yang umum digunakan:

- *Agglomerative*
- *K-Means*

#### 2.2.4 K-Means

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

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

Tabel 2.6: Tabel Dataset Mata Pelajaran

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

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

1. Inisialisasi *k* dan titik *centroid* awal.

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

Tabel 2.7: Hasil Pengelompokan Awal

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

2. Melakukan perhitungan *distance*.

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

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

3. Menempatkan setiap titik data ke titik *centroid* terdekat.

Titik data akan dikelompokan ke *centroid* terdekat dengan memilih nilai Euclidian *distance* paling kecil. Sebagai contoh, Tabel 5.4 pada *Step 2* menyatakan data ke-1 lebih dekat dengan data ke-2, karena memiliki nilai Euclidean distance paling kecil. Karena itu, data ke-2 bergabung pada titik *centroid* data ke-1. Hal ini berlaku pada setiap langkah.

Tabel 2.8: Mencari Centroid Kelompok

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

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

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

4. Menetapkan kelompok data baru pada masing-masing *cluster*.

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

5. Mencari titik *centroid* baru untuk setiap *cluster*.

Tabel 2.9: Hasil Cluster Baru

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

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

Tabel 2.10: Euclidean Distance Cluster 1, Cluster 2

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

#### 6. Menetapkan kelompok data akhir.

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

Tabel 2.11: Hasil Pengelompokan Akhir

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

## 2.3 Tahapan Evaluasi *Data Mining*

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

### 2.3.1 Menghitung Tingkat Akurasi untuk Model Klasifikasi

Hasil pemodelan klasifikasi dapat dievaluasi menggunakan perhitungan tingkat akurasi. Semakin tinggi tingkat akurasi yang diperoleh, maka hasil pemodelan klasifikasi menjadi semakin baik.

Akurasi dihitung dari rasio jumlah prediksi yang benar dengan jumlah total sampel input.

Berikut adalah persamaan untuk menghitung tingkat akurasi:

$$\text{Tingkat Akurasi} = \frac{\text{jumlah prediksi yang benar}}{\text{total sampel input}} \times 100\% \quad (2.9)$$

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

### 2.3.2 Menghitung Koefisien *Silhouette* untuk Model *Clustering*

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

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

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

Berikut adalah persamaan untuk menghitung koefisien *silhouette*:

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

Keterangan:

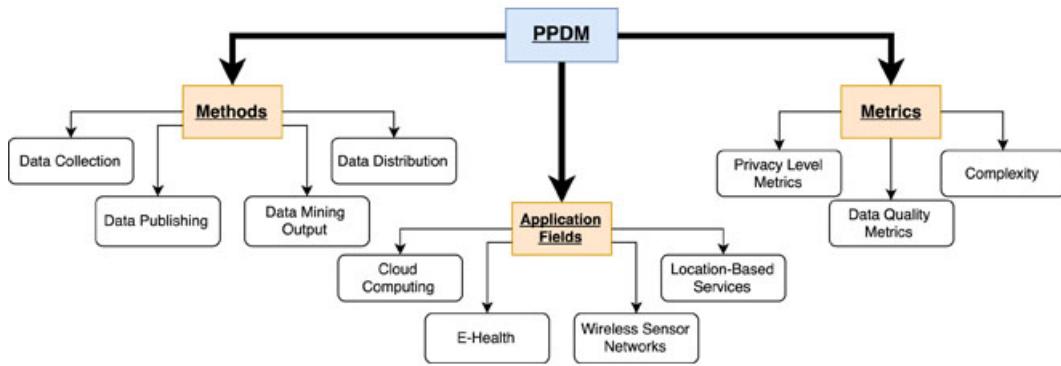
- $p$  = rata-rata *distance* setiap titik *intra-cluster* terhadap titik *centroid* pada *cluster* terdekat.
- $q$  = rata-rata *distance* setiap titik *intra-cluster* terhadap titik *centroid intra-cluster*.
- $\max(p, q)$  = nilai maksimum dari  $p$  dan  $q$ .

## 2.4 *Privacy-Preserving Data Mining* (PPDM)

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

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

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



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

#### 2.4.1 Metode pada PPDM

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

Berikut adalah metode perlindungan privasi pada PPDM:

- Pengumpulan data  
Pada fase ini, entitas yang bertugas mengumpulkan data tidak dapat dipercaya karena dapat mengumpulkan dan menggunakan data privasi individu secara tidak benar. Metode yang dapat digunakan untuk melindungi privasi saat pengumpulan data adalah *randomisation*.
- Publikasi data  
Pada fase ini, entitas ingin mempublikasikan datanya untuk keperluan penelitian atau analisis. Akan tetapi, ada beberapa individu yang mencoba melakukan tindakan deanonimisasi pada data-data privasi milik seseorang untuk tindakan kejahatan. Metode yang dapat digunakan untuk melindungi privasi saat publikasi data adalah *k-anonymity*.
- Hasil data mining  
Hasil data mining biasanya dibuat mudah diakses melalui aplikasi atau *interface*. Pada fase ini, entitas dapat melakukan kueri pada sistem untuk mencari informasi sensitif seseorang. Metode yang dapat digunakan untuk melindungi privasi pada hasil data mining adalah *association rule hiding*, *downgrading classifier effectiveness*, dan *query auditing and inference control*.
- Distribusi data  
Pada fase ini, beberapa entitas akan mencari wawasan umum dalam bentuk data statistik tanpa mengungkap informasi entitas lain. Akan tetapi ada beberapa entitas yang sengaja untuk menemukan celah keamanan privasi melalui pencocokan data dengan informasi umum yang diberikan secara publik. Hal ini dapat dicegah dengan memberlakukan *multiparty protocol* pada sistem distribusi data.

### 2.4.2 Metrik pada PPDM

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

Berikut adalah metrik perlindungan privasi pada PPDM:

- Tingkatan Privasi

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

- Kualitas Data

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

### 2.4.3 Model Serangan pada PPDM

Menurut Dalenius (1977), perlindungan privasi tidak memberikan kesempatan bagi orang lain untuk mendapatkan informasi sensitif individu meskipun orang lain mengetahui informasi umum yang berhubungan dengan informasi sensitif individu tersebut. Secara umum, orang lain dapat menemukan sebuah cara untuk memetakan sebuah data ke dalam tabel yang telah dianonimisasi ketika data tersebut telah dipublikasi. Serangan ini dikenal dengan istilah *linkage attack*.

Berikut adalah jenis serangan dari *linkage attack*:

- *Record Linkage*

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

- *Attribute Linkage*

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

## 2.5 Anonimisasi

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

Anonimisasi berupaya melindungi data pribadi atau atribut sensitif dengan menghapus atau mengenkripsi informasi yang dapat diidentifikasi secara pribadi dari database. Anonimisasi data

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

## 2.6 K-Anonymity

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

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

Berikut algoritma yang dapat diterapkan pada pemodelan *k-anonymity*:

- Algoritma *k-means clustering* akan melakukan beberapa iterasi sampai *centroid* dari semua data tidak lagi berubah atau perubahannya kecil. Algoritma *k-means clustering* tidak mampu untuk menyelesaikan masalah pada atribut yang bernilai kategorikal. Kelebihan dari algoritma *k-means clustering* adalah memiliki hasil pengelompokan yang sudah baik. Kekurangan dari algoritma *k-means clustering* adalah pemilihan *centroid* awal *k-means* secara acak, sehingga setelah digeneralisasi hasil pengelompokannya mengakibatkan hilangnya informasi yang besar.
- Algoritma *k-member* dapat melakukan generalisasi atribut kategorikal dengan memperoleh kualitas informasi yang lebih baik daripada algoritma *k-means clustering*. Namun algoritma *k-member* masih memiliki masalah ketika melakukan pengelompokan data. Kekurangan dari algoritma *k-member* adalah hanya mempertimbangkan pengelompokan data pada baris yang terakhir tanpa memperhatikan pengelompokan yang dihasilkan pada proses sebelumnya sehingga menyebabkan distribusi kelompok data pada beberapa bagian menjadi kurang tepat.
- Untuk menghindari kekurangan pada algoritma *k-means* dan algoritma *k-member*, maka kedua pendekatan ini digabung menjadi algoritma baru yaitu *Greedy k-member clustering*. Algoritma *Greedy k-member clustering* mendapatkan hasil pengelompokan yang lebih tepat dan memiliki nilai informasi yang lebih baik meskipun dilakukan generalisasi. Hasil akhir dari algoritma *Greedy k-member clustering* adalah data-data yang sejenis telah dikelompokan pada kelompok data yang sama. Untuk melakukan anonimisasi, digunakan konsep *Hierarchy Based Generalization*. *K-anonymity* menyamarkan data pada nilai quasi-identifier yang sama.

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

- *Identifier* (ID) adalah atribut yang unik dan secara langsung dapat mengidentifikasi seseorang seperti nama, nomor ID, dan nomor ponsel.

- *Quasi-identifier* (QID) adalah kombinasi atribut yang mungkin terjadi untuk mengidentifikasi individu berdasarkan penggabungan informasi lain dari luar. Seluruh atribut data terkecuali atribut identifier dapat dianggap sebagai atribut quasi-identifier.
- *Sensitive Attribute* (SA) adalah atribut yang menyangkut informasi sensitif seseorang, biasanya nilai atribut ini akan dirahasiakan saat distribusi data.
- *Non-sensitive Attribute* (NSA) adalah atribut yang tidak menyangkut informasi sensitif seseorang, biasanya nilai atribut ini langsung ditampilkan saat distribusi data.

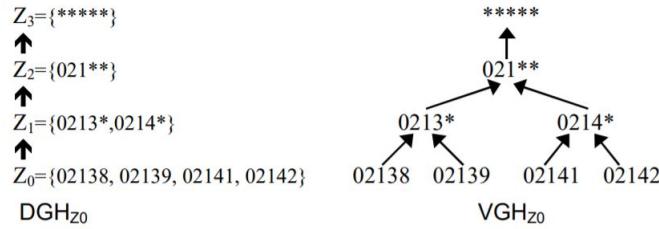
## 2.7 *Hierarchy Based Generalization*

*Hierarchy-based generalization* adalah tahapan anonimisasi pada pemodelan k-anonymity dimana *quasi-identifier* yang sama dikelompokan ke dalam cluster yang sama. *Hierarchy-based generalization* menggunakan konsep generalisasi dan supresi dalam melakukan anonimisasi. *Hierarchy-based generalization* termasuk metode *full-domain generalization*. *Full-domain generalization* diusulkan oleh Samarati dan Sweeney untuk memetakan seluruh domain untuk masing-masing atribut *quasi-identifier* pada tabel ke domain yang lebih umum berdasarkan kategori tertentu.

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

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

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



Gambar 2.6: DGH dan VGH pada Atribut ZIP



Gambar 2.7: DGH dan VGH pada Atribut Race

## 2.8 Greedy K-Member Clustering

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

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

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

**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 dikelompokan 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

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

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

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

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

Berikut adalah rumus menghitung *distance* antar data numerik:

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

Berikut adalah rumus menghitung distance antar data kategorikal:

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

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

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

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

- ii. Membuang himpunan data dari variabel r pada variabel S
- iii. Menambahkan himpunan data dari variabel r pada variabel c.
- iv. Menambahkan himpunan data dari variabel c pada variabel result

3. Pada kondisi  $|S| \neq 0$ , artinya jika masih terdapat data yang belum dimasukkan pada sebuah *cluster* dari tabel S, lakukan perulangan sebagai berikut:
- 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:

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

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

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

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

---

**Algorithm 1** Find Best Record

---

```

1: Function find_best_record(S,c)
2: Input: a set of records S and a cluster c.
3: Output: a record r ∈ S such that  $IL(c \cup \{r\})$  is minimal
4:
5:  $n = |S|$ 
6:  $min = \infty$ 
7:  $best = null$ 
8: for  $i = 1 \dots n$  do
9:   r = i-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 ∈ 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-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:  $result = \emptyset$ 
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  $S \neq \emptyset$  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.

## 2.9 Metrik *Distance* dan *Information Loss*

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

### 2.9.1 *Distance*

*Distance* adalah salah satu perhitungan untuk menyatakan akurasi terhadap utilitas sebuah data. *Distance* merupakan faktor yang paling penting untuk menentukan hasil pengelompokan data. Pemilihan *distance* yang baik dapat mencapai hasil klasifikasi dengan lebih optimal. Perhitungan *distance* dilakukan berdasarkan pengelompokan tipe data numerik atau kategorikal. Karena masalah *k-anonymity* menggunakan atribut numerik dan kategorikal, maka membutuhkan cara khusus untuk mengitung *distance* dari kedua jenis data pada saat yang sama.

### **Distance Data Numerik**

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

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

### **Distance Data Kategorikal**

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

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

### **Distance Record**

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

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

#### **2.9.2 Information Loss**

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

*Information loss* dihitung dengan rumus sebagai berikut:

$$IL(e) = |e| \cdot D(e) \quad (2.15)$$

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

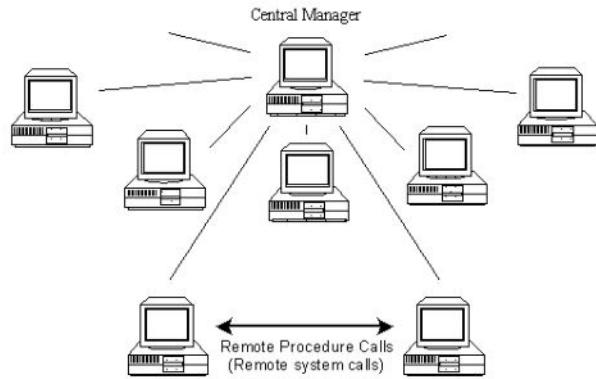
Total *Information Loss* dihitung dengan rumus sebagai berikut:

$$Total - IL(AT) = \sum_{e \in \epsilon} IL(e) \quad (2.17)$$

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

## 2.10 Sistem Terdistribusi

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



Gambar 2.8: Sistem Terdistribusi

### 2.10.1 Manfaat Sistem Terdistribusi

Berikut adalah manfaat penggunaan sistem terdistribusi:

- *Horizontal scalability*

Sistem terdistribusi menawarkan kemampuan untuk melakukan pemrosesan komputasi skala besar pada big data dengan harga yang murah.

- *Reliability*

Sistem terdistribusi dapat diandalkan karena proses komputasi pada sistem terdistribusi bergantung pada banyaknya komputer yang saling berkomunikasi satu sama lain untuk mencapai tujuan yang sama.

- *Performance*

Sistem terdistribusi dapat menangani proses komputasi tugas secara efisien karena beban kerja sesungguhnya dibagi menjadi beberapa bagian dan tersebar di beberapa komputer.

### 2.10.2 Tantangan Sistem Terdistribusi

Berikut adalah tantangan pada sistem terdistribusi:

- Penjadwalan

Kekuatan komputasi ada batasnya, sehingga sistem terdistribusi harus dapat memutuskan pekerjaan mana yang harus dikerjakan lebih dulu.

- Latensi

Dengan pertukaran data antara perangkat keras dan perangkat lunak menggunakan jalur komunikasi jaringan, sehingga nilai latensi menjadi sangat tinggi.

- Observasi

Ketika sistem terdistribusi menjadi kompleks, kemampuan pengamatan untuk memahami kegagalan pada sistem terdistribusi merupakan tantangan besar komputer.

## 2.11 Big Data

*Big data* adalah data yang besar dan kompleks sehingga tidak mungkin sistem tradisional dapat memproses dan bekerja pada lingkungan data yang besar secara maksimal. Data dapat dikategorikan sebagai data besar berdasarkan berbagai faktor. Konsep utama yang umum dalam semua faktor adalah jumlah data.

Berikut adalah karakteristik 5V pada *big data*:

- *Volume*

*Volume* mengacu pada jumlah data yang sangat besar. Data tumbuh begitu besar sehingga sistem komputasi tradisional tidak lagi dapat menanganinya seperti yang kita inginkan.

- *Velocity*

*Velocity* mengacu pada kecepatan di mana data dihasilkan. Setiap hari, sejumlah besar data dihasilkan, disimpan, dan dianalisis. Data dihasilkan dengan kecepatan kilat di seluruh dunia. Teknologi *big data* memungkinkan untuk mengeksplorasi data, saat data itu dihasilkan.

- *Variety*

*Variety* mengacu pada berbagai jenis data. Data terutama dikategorikan ke dalam data terstruktur dan tidak terstruktur. Faktanya, lebih dari 75 persen data dunia ada dalam bentuk yang tidak terstruktur.

- *Veracity*

*Veracity* mengacu pada kualitas data. Ketika menyimpan beberapa data yang besar, apabila tidak ada gunanya di masa depan, maka membuang-buang sumber daya untuk menyimpan data tersebut. Jadi, kita harus memeriksa kepercayaan data sebelum menyimpannya.

- *Value*

*Value* adalah bagian terpenting dari *big data*. Organisasi menggunakan data besar untuk menemukan nilai informasi baru. Menyimpan sejumlah besar data sampai pada ekstraksi informasi yang bermakna dari sekumpulan data tersebut.

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

## 2.12 Hadoop

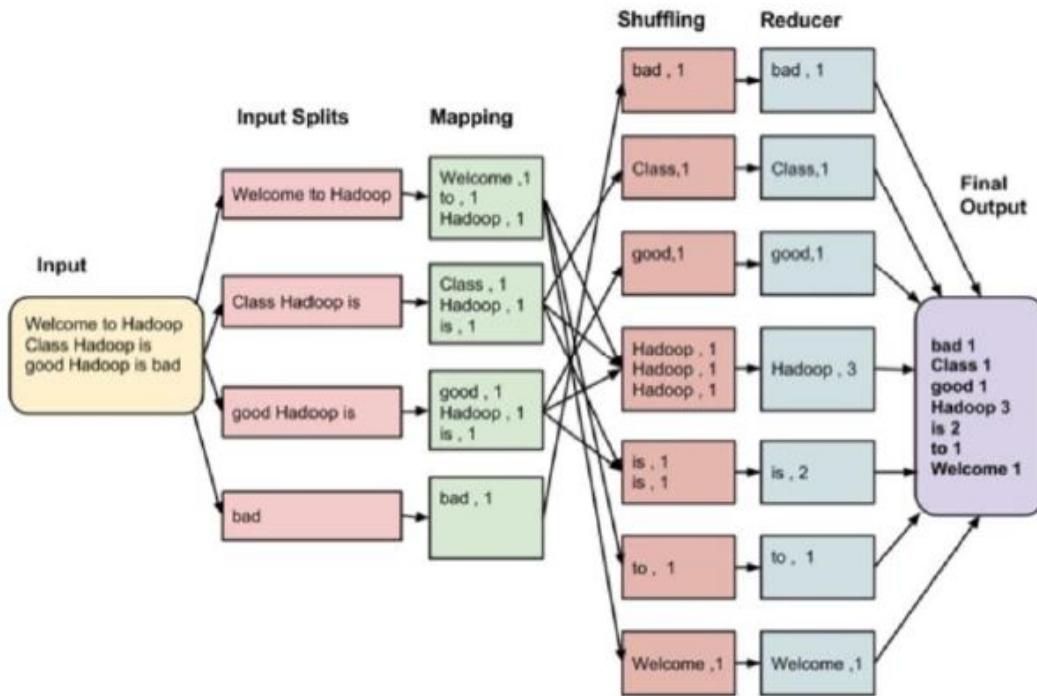
Hadoop adalah *framework* yang memanfaatkan beberapa komputer untuk menyelesaikan masalah yang melibatkan volume data sangat besar. Hadoop memecah input dari pengguna menjadi beberapa blok data dan masing-masing blok data diproses menggunakan konsep *MapReduce* di mana data akan diproses secara paralel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

### 2.12.1 HDFS

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

### 2.12.2 MapReduce

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



Gambar 2.9: Proses Komputasi pada MapReduce

Berikut adalah penjelasan masing-masing tahapan pada MapReduce:

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

## 2.13 Spark

Spark adalah teknologi komputasi *cluster* yang dirancang untuk komputasi cepat. Spark adalah paradigma pemrosesan data berukuran besar yang dikembangkan oleh para peneliti *University of California di Berkeley*. Spark adalah alternatif dari Hadoop MapReduce untuk mengatasi keterbatasan pemrosesan input output yang tidak efisien pada disk, dengan menggunakan memori. Fitur utama Spark adalah melakukan komputasi di dalam memori sehingga waktu komputasi menjadi lebih singkat dibandingkan waktu komputasi di dalam *disk*.

Berikut adalah karakteristik dari Spark:

- Kecepatan

Spark adalah alat komputasi *cluster* tujuan umum. Ini menjalankan aplikasi hingga 100 kali lebih cepat dalam memori dan 10 kali lebih cepat pada *disk* daripada Hadoop. Spark mengurangi jumlah operasi baca/tulis pada *disk* dan menyimpan data dalam memori.

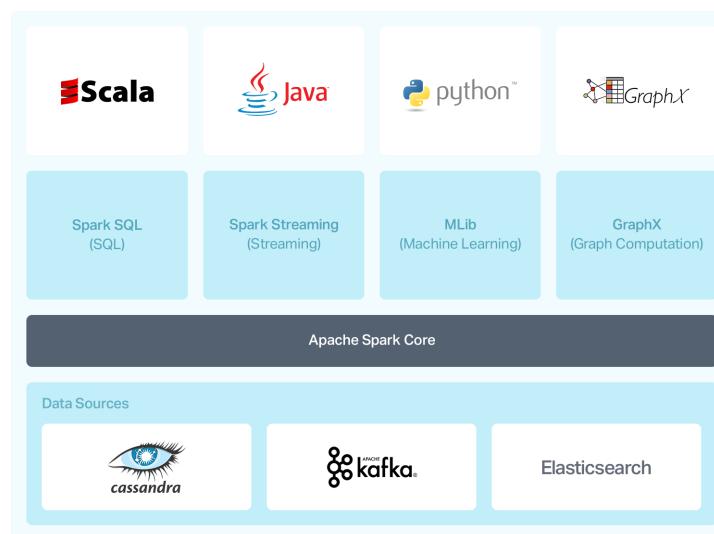
- Mudah untuk diatur

Spark dapat melakukan pemrosesan *batch*, analisis data secara interaktif, *machine learning*, dan *streaming data*. Semuanya pemrosesan tersebut dikerjakan pada satu komputer yang sama. Fungsi ini menjadikan Apache Spark sebagai mesin analisis data yang lengkap.

- Analisis secara *real-time*

Spark dapat dengan mudah memproses data *real-time*, misalnya *streaming data* secara *real-time* untuk ribuan peristiwa/detik. Contoh dari sumber *streaming data* adalah Twitter, Facebook, Instagram. *Streaming data* dapat diproses secara efisien oleh Spark.

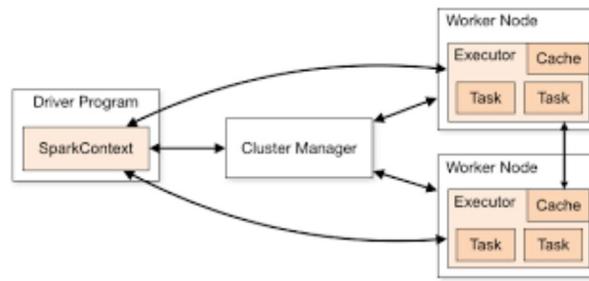
### 2.13.1 Ekosistem Spark



Gambar 2.10: Ekosistem Spark

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

### 2.13.2 Arsitektur Spark

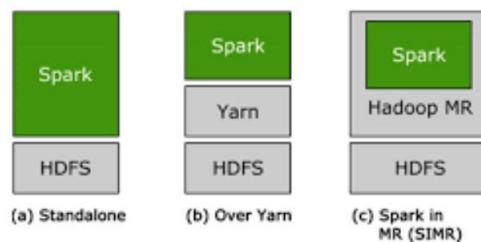


Gambar 2.11: Arsitektur Spark

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

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

### 2.13.3 Jenis Instalasi pada Spark



Gambar 2.12: Arsitektur Spark

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

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

### 2.13.4 Resilient Distributed Datasets (RDD)

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

Berikut adalah beberapa karakteristik yang dimiliki RDD:

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

Berikut adalah jenis operasi pada RDD:

- Fungsi *Transformation*

Fungsi *transformation* dilakukan secara *lazy*, sehingga hanya akan dikerjakan apabila dipanggil pada fungsi *action*. Fungsi *transformation* pada RDD akan dijelaskan pada tabel dibawah ini.

Fungsi	Deskripsi
map()	Mengembalikan RDD baru dengan menerapkan fungsi pada setiap elemen data
filter()	Mengembalikan RDD baru yang dibentuk dengan memilih elemen-elemen sumber di mana fungsi mengembalikan true
reduceByKey()	Menggabungkan nilai-nilai kunci menggunakan fungsi

- Fungsi *Action*

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

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

### 2.13.5 Dataframe

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

Berikut adalah beberapa karakteristik yang dimiliki *Dataframe*:

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

### 2.13.6 Komponen Spark

Komponen Spark adalah library tambahan pada Spark untuk melakukan proses komputasi pada lingkungan big data berdasarkan jenis-jenis kebutuhan pengolahan data. Berikut adalah penjelasan singkat mengenai komponen pada Spark:

- **Spark Core**

Spark Core adalah *library* Spark yang berisi fungsionalitas dasar Spark, termasuk komponen untuk penjadwalan tugas, manajemen memori, pemulihan kesalahan, dan berinteraksi dengan sistem penyimpanan. Spark Core menyediakan komputasi pada memori, fungsi *action* dan *transformation* untuk mengolah RDD.

- **Spark SQL**

Spark SQL memungkinkan pemrosesan kueri SQL pada lingkungan big data. Spark SQL menyediakan fungsi untuk menghitung nilai statistik dasar seperti *mean*, *median*, *modus*, nilai maksimum dan nilai minimum.

- **Spark Streaming**

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

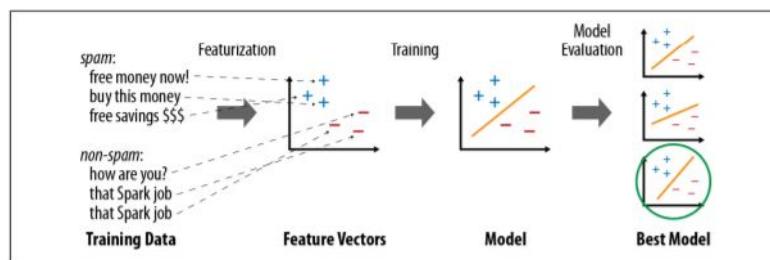
- **Spark MLLib**

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

## 2.14 Spark MLLib

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

### 2.14.1 Machine Learning pada Spark MLLib



Gambar 2.13: Tahapan Pembelajaran Machine Learning

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

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

### 1. Featurization

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

### 2. Training

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

### 3. Model Evaluation

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

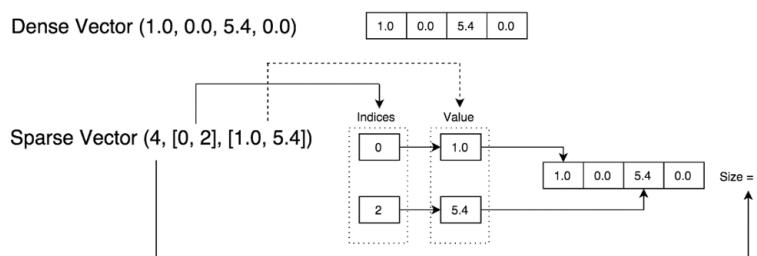
## 2.14.2 Tipe Data pada Spark MLlib

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

Berikut adalah beberapa jenis tipe data pada Spark MLlib:

- Vektor

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



Gambar 2.14: Contoh Vektor Dense dan Sparse

- Vektor *dense*

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

- Vektor *sparse*

Vektor *sparse* adalah vektor yang menyimpan setiap nilai fitur yang bukan nol pada dataset, sehingga jumlah elemen yang disimpan pada vektor *sparse* lebih sedikit dibandingkan dengan jumlah elemen yang disimpan pada vektor *dense*.

- *LabeledPoint*

*LabeledPoint* digunakan pada algoritma *supervised learning* yaitu klasifikasi dan regresi. Kelas *LabeledPoint* terletak pada package `mllib.regress`.

- *Various Model class*

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

### 2.14.3 Data Mining pada Spark MLlib

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

#### *Naive Bayes*

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

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

#### *K-Means*

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

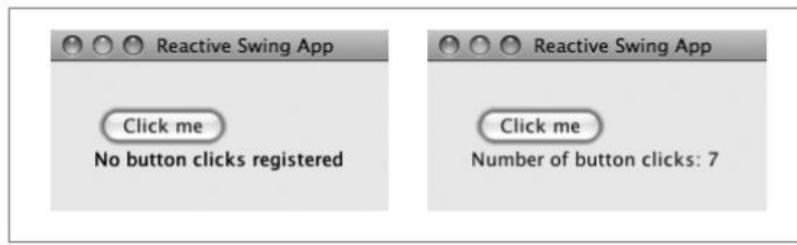
- *k* adalah jumlah cluster yang diinginkan.
- *maxIterations* adalah jumlah iterasi maksimum yang harus dijalankan.
- *initializationMode* menentukan inisialisasi centroid secara acak.
- *initializationSteps* menentukan jumlah langkah dalam algoritma *k-means*.
- *initialModel* adalah menentukan nilai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

## 2.15 Scala

Scala adalah bahasa pemrograman berbasis open source, dibuat oleh Profesor Martin Odersky. Scala adalah bahasa pemrograman multi-paradigma dan mendukung paradigma fungsional serta berorientasi objek. Untuk pengembangan Spark, penulisan sintaks Scala dianggap produktif untuk mengimplementasikan kode program. Pemrograman pada Scala mempertahankan prinsip keseimbangan antara produktivitas pengembangan program dan kinerja program. Pemrograman pada Scala tidak serumit pemrograman pada Java. Satu baris kode program pada Scala dapat menggantikan 20 hingga 25 baris kode Java. Karena alasan terbut, Scala menjadi bahasa pemrograman yang sangat diminati untuk melakukan pemrosesan *big data* pada Spark.

## 2.16 Scala Swing

Scala Swing adalah program berbasis *Graphical User Interface* (GUI) sehingga memiliki perbedaan dengan program Spark yang dieksekusi dengan terminal. Scala Swing bertujuan untuk memberi tampilan program sehingga hasil program diharapkan menjadi lebih interaktif. Scala menyediakan akses langsung terhadap kelas GUI pada Java menggunakan *library* Scala Swing. Dengan menggunakan Scala, penggunaan Scala Swing dapat memenuhi kebutuhan perancangan *User Interface* melalui berbagai macam komponen GUI pada umumnya. Gambar 2.15 adalah contoh implementasi GUI sederhana pada Scala Swing.



Gambar 2.15: GUI Sederhana pada Scala Swing

### 2.16.1 Panel dan Layout

*Panel* adalah tempat untuk menampilkan semua komponen GUI dengan beberapa aturan tata letak yang harus dipenuhi. Salah satu bagian tersulit pada perancangan aplikasi berbasis GUI adalah mengatur penempatan layout dengan benar. *Layout* terdiri dari beberapa komponen GUI seperti *Frame*, *Panel*, *Label* atau *Button*. Masing-masing komponen GUI pada *layout* memiliki nilai properti sendiri (warna, ukuran, posisi) yang dapat diatur secara manual.

### 2.16.2 Handling Event

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

## 2.17 Format Penyimpanan Data

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

### 2.17.1 CSV

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

Listing 2.1: Format Penyimpanan CSV

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

### 2.17.2 JSON

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

Listing 2.2: Format Penyimpanan JSON

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



## BAB 3

# ANALISIS

Pada bab ini akan dijelaskan analisis masalah penelitian ini. Analisis ini meliputi analisis masalah, eksplorasi spark, studi kasus, dan gambaran umum perangkat lunak.

### 3.1 Analisis Masalah

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

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

#### 3.1.1 Dataset Eksperimen

Dataset yang dipakai adalah *Adult*. Dataset ini diperoleh dari website Kaggle. Dataset ini disimpan dalam format CSV seperti penjelasan pada bagian 2.17. 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.

Listing 3.1: Dataset Adult

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

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:  $\leq 50K, > 50K$

### 3.1.2 *Personally Identifiable Information*

Pada bagian 2.1, telah dijelaskan mengenai konsep *Personally Identifiable Information* (PII). PII digunakan untuk mengelompokkan nilai atribut berdasarkan kategori atribut yang digunakan pada proses anonimisasi data. Berdasarkan bagian 2.5, 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.

### 3.1.3 Perhitungan *Distance* dan *Information Loss*

Pada bagian 2.9, 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.

#### *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:

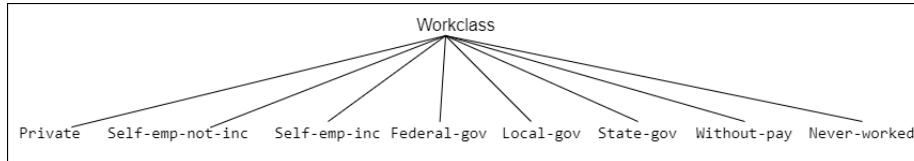
1. 39, State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K
2. 50, Self-emp-not-inc, 83311, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 13, United-States, <=50K

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

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

*Distance* atribut kategorikal dapat dihitung sebagai berikut berdasarkan *workclass* data pertama ( $v_1$ )= *State-gov*, *workclass* data kedua ( $v_2$ )= *Self-emp-not-inc*, jumlah subtree ( $H(\Lambda(v_i, v_j))$ )= 1, dan tinggi taxonomy tree ( $H(T_D)$ )= 1 seperti pada Gambar 3.1.

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



Gambar 3.1: Taxonomy Tree (Workclass)

#### *Information Loss*

*Information Loss* (IL) bertujuan untuk mengevaluasi seberapa baik kinerja algoritma *k-anonymity* terhadap nilai informasi sebuah data. Tabel 3.1 adalah contoh hasil pengelompokan data pada dataset *Adult*:

Tabel 3.1: 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  $H(\Lambda(\cup C_j))=1$ ,  $H(T_{C_j})=1$ .

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

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

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

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

### 3.1.4 Greedy K-Member Clustering

Algoritma *Greedy k-member clustering* telah dijelaskan pada bagian 2.8. 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.

Melalui sampel data pada Tabel 3.2, 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 ini siap ditelusuri lebih lanjut.

Tabel 3.2: 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

Berikut adalah tahapan yang terjadi pada algoritma *Greedy k-member clustering*:

1. Nilai awal result =  $\emptyset$ ,  $r = \{t1\}$ ,  $|S| = 5$ ,  $k = 2$
2. Karena kondisi  $|S| \geq k$  terpenuhi, maka dilakukan perulangan sebagai berikut:
  - (a) Nilai r diubah menjadi  $r = \{t3\}$ , karena terbukti data t3 memiliki  $\Delta(t1, t3) = 1.7189$

yang paling tinggi dari seluruh *distance* lain. Berikut adalah contoh perhitungannya:

$$\begin{aligned}\Delta(t_1, t_2) &= 1.715 \\ \Delta(t_1, t_3) &= 2.431 \\ \Delta(t_1, t_4) &= 2.122 \\ \Delta(t_1, t_5) &= 1.621\end{aligned}$$

- (b) Nilai awal  $S = \{t_1, t_2, t_4, t_5\}$
- (c) Nilai awal  $c = \{t_3\}$ ,  $|c| = 1$
- (d) Karena kondisi  $|c| < k$  terpenuhi, maka dilakukan perulangan sebagai berikut:
  - i. Nilai  $r$  diubah menjadi  $r = \{t_3, t_4\}$ , karena terbukti data  $t_4$  memiliki  $IL(t_3 \cup t_4) = 0.330$  yang paling rendah dari seluruh data lain. Berikut adalah contoh perhitungannya:

$$\begin{aligned}IL(t_3 \cup t_1) &= 0.479 \\ IL(t_3 \cup t_2) &= 0.515 \\ IL(t_3 \cup t_4) &= 0.330 \\ IL(t_3 \cup t_5) &= 0.367\end{aligned}$$

- ii. Nilai  $S$  diubah menjadi  $S = \{t_1, t_2, t_5\}$ ,  $|S| = 4$
- iii. Nilai  $c$  ditambahkan menjadi  $c = \{t_3, t_4\}$ ,  $|c| = 2$
- (e) Karena kondisi  $|c| < k$  sudah tidak terpenuhi lagi, maka perulangan ini akan berhenti
- (f) Nilai *result* akan ditambahkan menjadi  $result = \{t_3, t_4\}$
- (g) Karena kondisi  $|S| \geq k$  masih terpenuhi, maka perulangan akan tetap berlanjut sampai pada kondisi dimana  $|S| < k$  sehingga hasil akhirnya adalah  $result = \{\{t_3, t_4\}, \{t_2, t_5\}\}$ ,  $S = \{t_1\}$ ,  $|S| = 1$

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

- (a) Nilai  $r$  diubah menjadi  $r = \{t_1\}$
- (b) Nilai  $S$  diubah menjadi  $S = \{\phi\}$ ,  $|S| = 0$
- (c) Nilai  $c$  diubah menjadi  $c = \{t_3, t_4\}$  karena terbukti *cluster*  $c$  memiliki  $IL(\{t_3, t_4\} \cup t_1) = 0.279$  yang paling rendah dari seluruh *cluster* lain. Berikut adalah contoh perhitungannya:

$$\begin{aligned}IL(\{t_3, t_4\} \cup t_1) &= 0.279 \\ IL(\{t_2, t_5\} \cup t_1) &= 0.515\end{aligned}$$

- (d) Nilai  $c$  ditambahkan menjadi  $c = \{t_1, t_3, t_4\}$
  - (e) Nilai  $c$  pada perulangan ini tidak akan ditambahkan pada *result*, karena telah ditetapkan  $k = 2$  sedangkan jumlah datanya ganjil, sehingga sisa data tersebut tidak akan dicatat pada variabel *result* agar menjaga masing-masing *cluster* hanya memiliki 2 anggota saja.
  - (f) Karena kondisi  $S \neq 0$  sudah tidak terpenuhi lagi, maka perulangan ini akan berhenti.
4. Hasil akhirnya adalah  $result = \{\{t_3, t_4\}, \{t_2, t_5\}\}$  dikembalikan sebagai output untuk algoritma *Greedy k-member clustering* seperti pada Tabel 3.3 sebagai berikut:

Tabel 3.3: 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

### 3.1.5 Domain Generalization Hierarchy

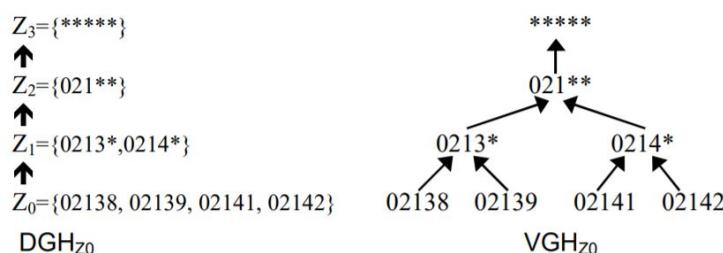
Pada bagian 2.7, telah dijelaskan konsep mengenai *Hierarchy Based Generalization*. DGH adalah contoh penerapan dari *Hierarchy Based Generalization*. DGH bertujuan untuk melindungi data dengan cara menerapkan metode generalisasi terhadap nilai atribut data yang bersifat unik, agar menjadi nilai yang 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}

Nilai 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 {\*\*\*\*\*}



Gambar 3.2: DGH dan VGH pada atribut ZIP

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

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

### 3.1.6 *K-Anonymity*

Pada bagian 2.5, dijelaskan konsep anonimisasi. *K-anonymity* 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 3.4 adalah kelompok data yang dihasilkan oleh algoritma *Greedy k-member clustering*.

Tabel 3.4: 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 proses anonimisasi dengan model *k-anonymity*:

1. Diketahui *quasi-identifier* sebagai berikut QI = {Age, ZIP, Sex, Hours/week} dan *sensitive attribute* sebagai berikut SA = {Workclass, Education}
2. Mencari nilai *quasi-identifier* yang unik pada kelompok *cluster* yang sama. Sebagai contoh, *cluster 2* memiliki nilai *quasi-identifier* yang unik sebagai berikut QI = {Age, Hours/week}
3. Melakukan generalisasi DGH pada nilai *quasi-identifier* yang unik menjadi bentuk. Sebagai contoh, QI = {Age, Hours/week} memiliki nilai yang unik, sehingga diubah menjadi Age = {[40-50]}, Hours/week = {[33-37]}
4. *Sensitive attribute* tidak akan dilakukan generalisasi, karena *quasi-identifier* sudah dilakukan generalisasi sehingga seseorang akan sulit untuk menebak kepemilikan dari *sensitive attribute*.
5. Ulangi hal yang sama pada langkah sebelumnya untuk setiap *cluster*. Hasil akhir dari proses anonimisasi ada pada Tabel 3.5 sebagai berikut:

Tabel 3.5: 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

## 3.2 Eksplorasi Spark

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

Berikut adalah beberapa hal penting terkait Spark:

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

### 3.2.1 Instalasi Spark

Spark 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 dijelaskan pada bagian [2.13.3](#). 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 ke atas.
- Scala 2.10 sudah usang apabila dipakai pada Spark 2.4.1 ke atas.
- Hadoop 2.6.5 telah dihilangkan pada implementasi Spark 2.2.0 ke atas.

Berikut adalah beberapa hal yang harus dipenuhi:

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

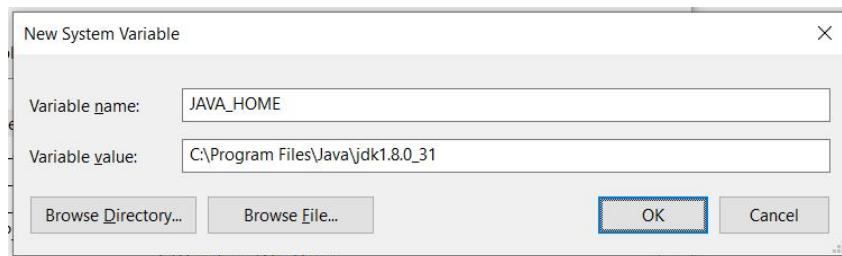
Berikut adalah tahapan instalasi Spark 2.4.5 secara umum:

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

## Instalasi Java 8

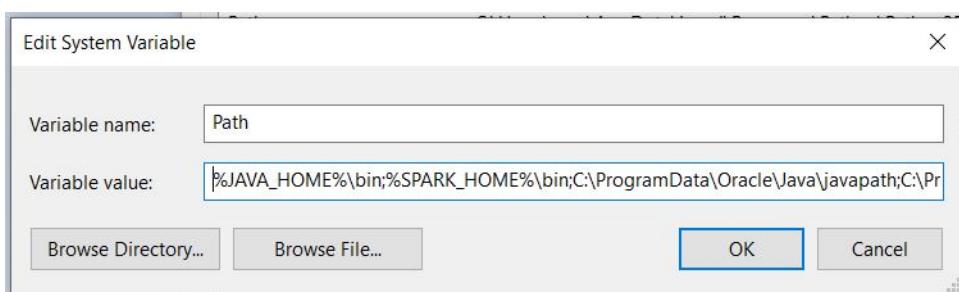
Berikut adalah tahapan instalasi Java 8 secara lengkap:

1. Download Java SE Development Kit 8u31 pada link berikut <https://www.oracle.com/technetwork/java/javase/downloads/java-archive-javase8-2177648.html>
2. Lakukan instalasi Java SE Development Kit 8u31 seperti biasa.
3. Pilih menu *Edit the system environment variables*.
4. Buat *environment variables* baru seperti Gambar 3.3.



Gambar 3.3: Environment Variables

5. Tambahkan %JAVA\_HOME%\bin; pada Path di System variables seperti Gambar 3.7.



Gambar 3.4: Penambahan Variable Value

Berikut adalah tahapan verifikasi terhadap instalasi Java 8:

1. Pilih menu *command prompt*.
2. Jalankan perintah java -version pada Command Prompt.

```
Command Prompt
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\asus>java -version
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
```

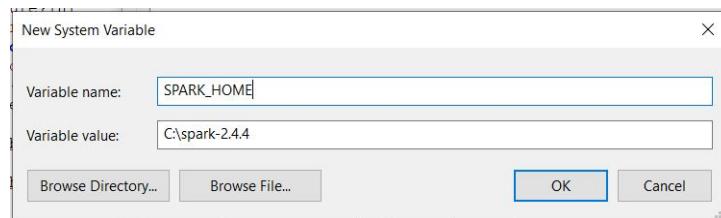
Gambar 3.5: Perintah java -version

3. Apabila sistem tidak menampilkan pesan error, maka Java 8 sudah terpasang dengan baik.

## Instalasi Spark 2.4.5

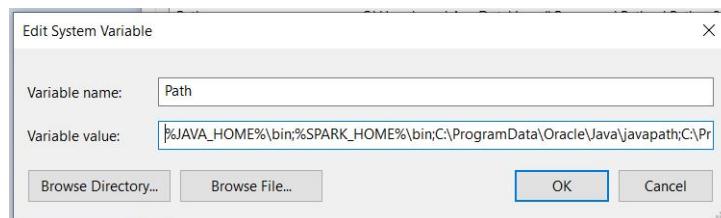
Berikut adalah tahapan instalasi Spark 2.4.5 secara lengkap:

1. Download winutils.exe dari link <https://github.com/steveloughran/winutils/tree/master/hadoop-2.7.1/bin>, tempatkan winutils.exe pada C:\winutils\bin
  2. 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>
  3. Buat folder sebagai berikut C:\spark-2.4.4 dan ekstraksi file spark-2.4.5-bin-hadoop2.7.tgz di dalam folder tersebut.
  4. Buat *environment variables* baru seperti Gambar 3.6.



Gambar 3.6: Environment Variable

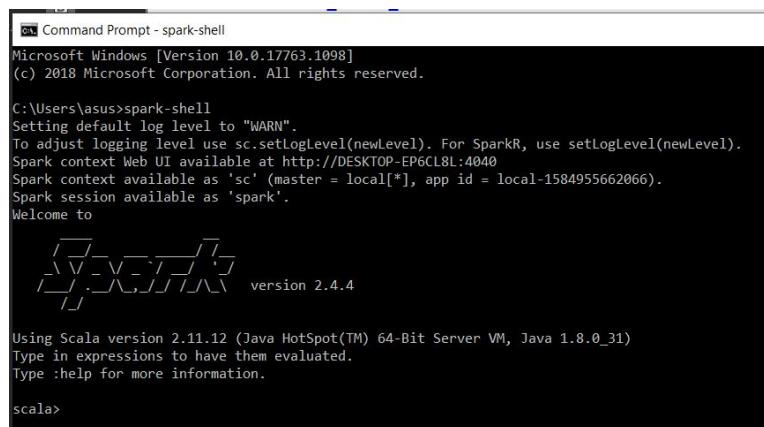
5. Tambahkan %SPARK\_HOME%\bin; pada Path di System variables seperti Gambar 3.7



Gambar 3.7: Penambahan Variable Value

Berikut adalah tahapan verifikasi terhadap instalasi Spark 2.4.5:

1. Jalankan perintah `spark-shell` pada *command prompt*.
  2. Apabila terminal menampilkan tampilan seperti pada Gambar 3.8, artinya Spark 2.4.5 sudah dapat berjalan dengan baik pada komputer tersebut.



Gambar 3.8: Spark 2.4.5

## Instalasi IntelliJ untuk Scala SBT

Berikut adalah tahapan instalasi IntelliJ:

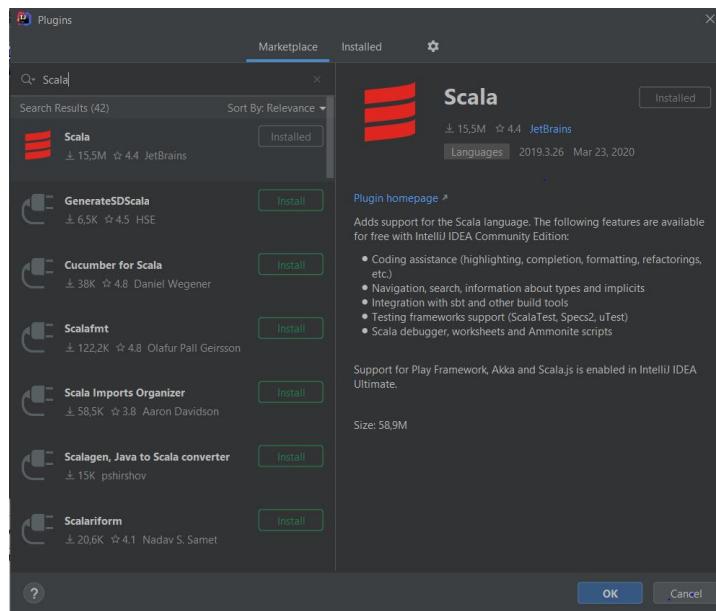
1. Download IntelliJ melalui link berikut  
<https://www.jetbrains.com/idea/download/#section=windows>
2. Lakukan instalasi IntelliJ seperti biasa.



Gambar 3.9: Instalasi IntelliJ

Berikut adalah tahapan pemasangan *plugin* Scala pada IntelliJ.

1. Pilih menu *Configure* pada IntelliJ, lalu pilih menu *Plugins*.
2. Telusuri *plugin* Scala pada kolom pencarian seperti Gambar 3.10.



Gambar 3.10: Plugins Scala

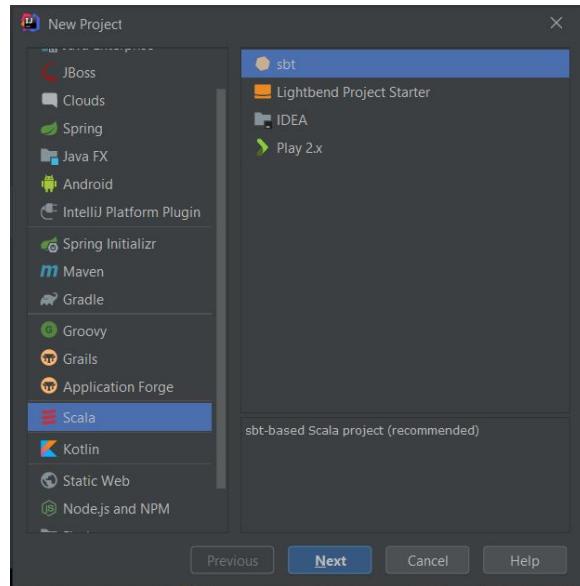
3. Klik tombol *install*

### 3.2.2 Membuat *Project Spark* pada IntelliJ

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

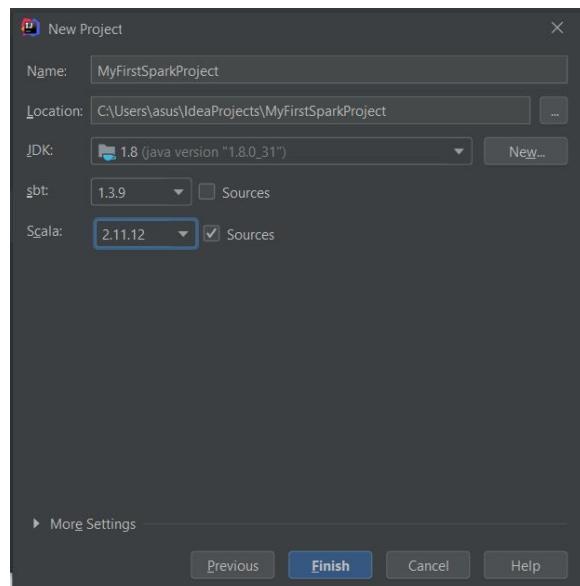
Berikut adalah tahapan pembuatan *project Spark* pada IntelliJ:

1. Memilih menu *Create New Project*
2. Menggunakan bahasa pemrograman Scala berbasis sbt seperti Gambar 3.11.



Gambar 3.11: Memilih Bahasa Scala Berbasis sbt

3. Melakukan konfigurasi pada project Spark baru seperti Gambar 3.12.



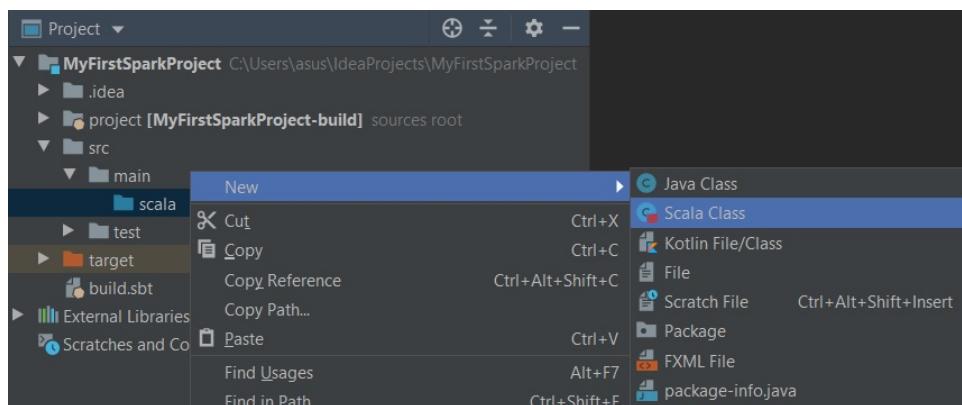
Gambar 3.12: Melakukan Konfigurasi Project Spark

4. Listing 3.2 adalah perintah *import libraryDependencies* pada file *build.sbt*  
Contoh: spark-core, spark-sql, spark-mllib.

Listing 3.2: Melakukan Import Library Spark

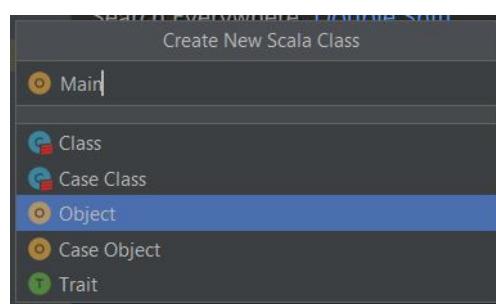
```
name := "NamaProject"
version := "0.1"
scalaVersion := "2.11.12"
// https://mvnrepository.com/artifact/org.apache.spark/spark-core
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.2.0"
// https://mvnrepository.com/artifact/org.apache.spark/spark-sql
libraryDependencies += "org.apache.spark"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

5. Menambahkan *Scala class* pada *src/main/scala* seperti Gambar 3.13.



Gambar 3.13: Menambahkan Scala Class pada Project Spark

6. Memilih tipe *Scala class* sebagai *Object* seperti Gambar 3.14.



Gambar 3.14: Memilih Tipe Object pada Scala Class

7. Listing 3.3 adalah perintah untuk menambahkan *main method* pada *Scala class*.

Listing 3.3: Menambahkan Main method pada Scala Class

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

### 3.2.3 Membuat *File JAR* pada Command Prompt

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

Berikut adalah tahapan untuk membuat File JAR:

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

### 3.2.4 Menjalankan Program Spark pada Komputer Lokal

Apabila ukuran data input eksperimen kecil, maka program Spark dapat dijalankan pada komputer lokal menggunakan perintah dari Command Prompt. Waktu komputasi pada komputer lokal akan jauh lebih lama dibandingkan dijalankan pada server Hadoop Cluster.

Berikut adalah tahapan menjalankan program Spark pada komputer lokal:

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

### 3.2.5 Menjalankan Program Spark pada Hadoop Cluster

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

Berikut adalah tahapan menjalankan program Spark pada Hadoop cluster:

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

### 3.3 Studi Kasus

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

#### 3.3.1 Eksperimen Scala

Pada bagian 2.15 telah dijelaskan tujuan dari penggunaan bahasa 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.

##### 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 3.4 mengenai jenis variabel pada Scala.

Listing 3.4: 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 3.5 mengenai jenis tipe data pada Scala.

Listing 3.5: Menentukan Jenis Tipe Data pada Scala

```
val donutsBought: Int = 5
val bigNumberOfDonuts: Long = 10000000
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 3.6 mengenai *immutable collection* pada Scala dan Listing 3.7 mengenai *mutable collection* pada Scala.

Listing 3.6: Membuat immutable collection pada Scala

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

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

Listing 3.7: 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 pada Scala

Kelas pada Scala memiliki fungsi kelas yang sama pada Java yaitu untuk menyimpan variabel dan method. Kode program dapat dilihat pada Listing 3.8 mengenai cara membuat kelas pada Scala.

Listing 3.8: Membuat Kelas Object pada Scala

```
class AreaOfRectangle
{
    var length = 20; // Variables
    var height = 40;

    def area() // Method which gives the area of the rectangle
    {
        var ar = length * height;
        println("Area of the rectangle is :" + ar);
    }
}
```

## Membuat *Singleton Object* pada Scala

Scala tidak memiliki variabel statik seperti pada Java, sehingga fungsinya digantikan oleh *singleton object*. *Singleton object* adalah objek yang mendefinisikan method main dari kelas-kelas pada Scala. Kode program dapat dilihat pada Listing 3.9 mengenai cara membuat *singletion object* pada Scala.

Listing 3.9: Membuat Kelas Object pada Scala

```
object Main
{
    def main(args: Array[String])
    {
        // Creating object of AreaOfRectangle class
        var obj = new AreaOfRectangle();
        obj.area();
    }
}
```

## 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 3.10 mengenai pembuatan fungsi pada Scala.

Listing 3.10: Membuat Fungsi Sedehana pada Scala

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

    // make some calculations ...
    2.50 * quantity
}
```

## 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 3.11 mengenai percabangan pada Scala.

Listing 3.11: 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 3.12 mengenai perulangan pada Scala.

Listing 3.12: Membuat Fungsi Perulangan pada Scala

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

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

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

### 3.3.2 Eksperimen Spark

Spark adalah teknologi yang digunakan untuk mengolah *big data* berdasarkan konsep dari bagian 2.13. Spark membagi satu pekerjaan pada masing-masing *Worker Node* seperti pada bagian 2.13.2. Oleh karena itu Spark memecah data partisi data agar data yang besar dapat distribusikan ke masing-masing komputer. Berikut adalah beberapa fungsi dasar Spark untuk mengolah partisi data.

#### Melakukan Konfigurasi Spark

Berikut adalah tahapan konfigurasi Spark pada Main Class:

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

Listing 3.13: Konfigurasi Spark

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

## Membuat RDD

Pada bagian 2.13.4, sudah dijelaskan konsep RDD. Pada bagian ini, akan dilakukan eksperimen mengenai jenis-jenis cara untuk membuat RDD pada Spark.

Berikut adalah beberapa cara untuk membuat RDD:

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

Listing 3.14: Cara Pembuatan RDD

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

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

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

## Membuat Dataframe

Pada bagian 2.13.5, sudah dijelaskan konsep *DataFrame*. Pada bagian ini, akan dilakukan eksperimen mengenai jenis-jenis cara untuk membuat *DataFrame* pada Spark.

Berikut adalah beberapa cara untuk membuat *DataFrame*:

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

Listing 3.15: Cara Pembuatan Dataframe

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

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

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

### Memanggil Fungsi *Transformation*

Pada bagian ?? dijelaskan jenis-jenis fungsi *Transformation* pada RDD. Listing 3.16 adalah contoh penerapan jenis-jenis fungsi *Transformation* pada Spark, berikut adalah penjelasan singkat dari masing-masing fungsi:

- `select()`: menampilkan isi RDD berdasarkan nama variabel yang menyimpan RDD.
- `filter()/where()`: menyeleksi isi RDD berdasarkan kondisi tertentu
- `sort()/orderBy()`: mengurutkan isi RDD berdasarkan keterurutan atribut tertentu
- `groupBy() dan agg()`: mengelompokan isi RDD dan melakukan agregasi.
- `join()`: menggabungkan dua RDD yang berbeda berdasarkan kesamaan nilai atribut.

Listing 3.16: Contoh Fungsi Transformation

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

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

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

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

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

### Memanggil Fungsi *Action*

Pada bagian ?? dijelaskan jenis-jenis fungsi *Action* pada RDD. Listing 3.17 adalah contoh penerapan jenis-jenis fungsi *Action* pada Spark, berikut adalah penjelasan singkat dari masing-masing fungsi:

- `show()`: menampilkan n baris pertama dari *DataFrame* atau RDD
- `take()`: menampilkan beberapa baris dari *DataFrame* atau RDD
- `collect()`: mengumpulkan seluruh data dari *DataFrame* atau RDD
- `count()`: menghitung jumlah baris
- `printSchema()`: menampilkan nama kolom dan tipe data

Listing 3.17: Contoh Fungsi Action

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

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

# 3. collect()
df.collect()

# 4. count()
df.count()

# 6. printSchema()
df.printSchema()

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

### Memanggil Fungsi RDD

Listing 3.18 adalah contoh penerapan jenis-jenis fungsi RDD pada Spark, berikut adalah penjelasan singkat dari masing-masing fungsi RDD:

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

Listing 3.18: Contoh Fungsi RDD

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

### Membuat Variabel Global

Listing 3.19 adalah perintah untuk membuat variabel global pada Spark.

Listing 3.19: Membuat Variabel Global

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

### 3.3.3 Eksperimen Komponen Spark

Pada bagian 2.13.6 dijelaskan mengenai jenis-jenis komponen Spark dan tujuan penggunaannya. Pada bagian ini akan dilakukan eksperimen berdasarkan masing-masing jenis komponen Spark.

#### Spark Core

Berikut adalah langkah-langkah eksperimen dari Spark Core:

1. Listing 3.20 adalah membuat perintah SparkSession untuk inisialisasi project Spark

Listing 3.20: Membuat SparkSession

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

2. Listing 3.21 adalah melihat dan mengatur partisi RDD berdasarkan data input CSV

Listing 3.21: Melihat dan Mengatur Partisi RDD

```
val sc = spark.sparkContext

val rdd: RDD[String] = sc.textFile("input/adult100k.csv")
println("initial partition count:" + rdd.getNumPartitions)

val reparRdd = rdd.repartition(4)
println("re-partition count:" + reparRdd.getNumPartitions)
```

3. Listing 3.22 adalah membuat jenis-jenis fungsi *transformation*.

Listing 3.22: Membuat Fungsi Transformation

```
//Transformation - flatMap
val rdd2 = rdd.flatMap(f => f.split(","))
rdd2.foreach(f => println(f))

//Transformation - map
val rdd3: RDD[(String, Int)] = rdd2.map(key => (key, 1))
rdd3.foreach(println)

//Transformation - filter
val rdd4 = rdd3.filter(a => a._1.startsWith("State-gov"))
rdd4.foreach(println)

//Transformation - reduceByKey
val rdd5 = rdd3.reduceByKey((x,y)=> x + y)
rdd5.foreach(println)

//Transformation - sortByKey
val rdd6 = rdd5.map(a => (a._2, a._1)).sortByKey()
```

4. Listing 3.23 adalah membuat jenis-jenis fungsi *action*.

Listing 3.23: Membuat Fungsi Action

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

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

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

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

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

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

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

## Spark SQL

Berikut adalah langkah-langkah eksperimen dari Spark Core:

1. Listing 3.24 adalah perintah untuk membuat SparkSession saat inisialisasi *project* Spark

Listing 3.24: Membuat Perintah SparkSession

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

2. Listing 3.25 adalah perintah untuk membuat *DataFrame* dari data input CSV.

Listing 3.25: Membuat Dataframe

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

peopleDF.printSchema()
```

3. Listing 3.26 adalah perintah untuk membuat tabel sementara, melakukan kueri, dan menyimpan hasil kueri pada file CSV.

Listing 3.26: Membuat Tabel Sementara

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

4. Listing 3.27 adalah perintah untuk mencari nilai statistik seperti jumlah data, *mean*, standar deviasi, nilai minimum dan maksimum.

Listing 3.27: Mencari Nilai Statistik

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

5. Listing 3.28 adalah perintah untuk mencari nilai *median*.

Listing 3.28: Mencari Nilai Median

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

6. Listing 3.29 adalah perintah untuk mencari nilai *modus*.

Listing 3.29: Mencari Nilai Modus

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

## Spark MLlib

Berikut adalah langkah-langkah eksperimen dari Spark Core:

1. Listing 3.30 adalah perintah untuk membuat SparkSession saat inisialisasi *project* Spark

Listing 3.30: Membuat Perintah SparkSession

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

2. Listing 3.31 adalah perintah untuk membuat skema untuk *DataFrame*.

Listing 3.31: Membuat Skema Dataframe

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

3. Listing 3.32 adalah perintah untuk mengubah data input CSV menjadi *DataFrame* berdasarkan skema.

Listing 3.32: Mengubah CSV Menjadi Dataframe

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

adult100k_df.show()
```

4. Listing 3.33 adalah perintah untuk menggunakan fungsi *stringIndexer* untuk membuat kolom indeks dan fungsi *oneHotEncoder* untuk membuat kolom vektor.

Listing 3.33: Membuat Kolom Index

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

5. Listing 3.34 adalah perintah untuk menambahkan kolom vektor dan kolom indeks pada *DataFrame*.

Listing 3.34: Membuat Kolom Vektor

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

6. Listing 3.35 adalah perintah untuk memilih jenis implementasi vektor yang akan digunakan.

Listing 3.35: Memilih Jenis Vektor

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

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

7. Listing 3.36 adalah perintah untuk membuat vektor fitur dari setiap baris data pada *DataFrame*.

Listing 3.36: Membuat Vektor Fitur

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

### 3.3.4 Eksperimen Spark MLIB

Pada bagian 2.14 telah dijelaskan mengenai konsep dan contoh pemodelan pada Spark MLlib. Pada penelitian ini akan digunakan pemodelan *Naive Bayes* untuk permasalahan klasifikasi pada bagian 2.2.2 dan *k-means* untuk permasalahan clustering pada bagian 2.2.4.

#### *Naive Bayes*

Pada bagian 2.14.3 menjelaskan parameter pemodelan *Naive Bayes* pada Spark MLlib. Berikut adalah tahapan eksperimen pada Listing 3.37 untuk pemodelan *Naive Bayes*:

1. Membagi data input CSV menjadi training data dan test data.
2. Melakukan pelatihan data pada pemodelan Naive Bayes.
3. Mengembalikan hasil klasifikasi dalam bentuk tabel.
4. Menghitung akurasi dari klasifikasi label kelas.

Listing 3.37: 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)
```

Dataset yang dipakai untuk eksperimen *naive bayes* ini adalah sampel data dari dataset Adult yang sudah dijelaskan pada bagian. Data ini berjumlah 100.000 baris data dengan ukuran 11.000 KB. Dataset ini akan dibagi menjadi data test dan data training. Jumlah data test adalah 30.000 baris data, sedangkan jumlah data training adalah 70.000 data. Jumlah data training lebih banyak karena akan dipakai untuk pelatihan model *naive bayes*.

Gambar 3.15: Hasil Naive Bayes Spark MLlib

Hasil pemodelan *naive bayes* dari eksperimen ini adalah label data. Cluster ini terbentuk berdasarkan distance terdekat antara anggota data dengan titik centroid pada cluster tersebut. Sehingga data-data yang tergabung pada sebuah cluster, sudah dipastikan bahwa data-data tersebut lebih dekat dengan titik centroid pada cluster tersebut dibanding dengan titik centroid pada cluster lainnya. Gambar 3.15 menunjukan bahwa data dengan nilai umur yang sama ( $age = 17$ ) memiliki kelompok cluster yang sama ( $prediction = 3.0$ ).

## *K-Means*

Pada bagian 2.14.3 menjelaskan parameter pemodelan *k-means* pada Spark MLlib. Berikut adalah tahapan eksperimen pada Listing 3.38 untuk pemodelan *k-means*:

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

Listing 3.38: 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)
```

Dataset yang dipakai untuk eksperimen *k-means* ini adalah sampel data dari dataset Adult yang sudah dijelaskan pada bagian. Data ini berjumlah 100.000 baris data dengan ukuran 11.000 KB. Batas maksimum iterasi untuk fungsi k-means telah diatur sedemikian rupa oleh library Spark MLlib itu sendiri, sehingga parameter ini tidak perlu lagi diset manual oleh pengguna. Jumlah cluster yang akan dibentuk pada eksperimen ini berjumlah 8 cluster. Jumlah cluster pada fungsi k-means nantinya dapat diatur kembali sesuai kebutuhan eksperimen.

age	prediction
null	0
39	5
50	1
38	5
53	1
28	5
37	5
49	5
52	1
31	5

only showing top 10 rows

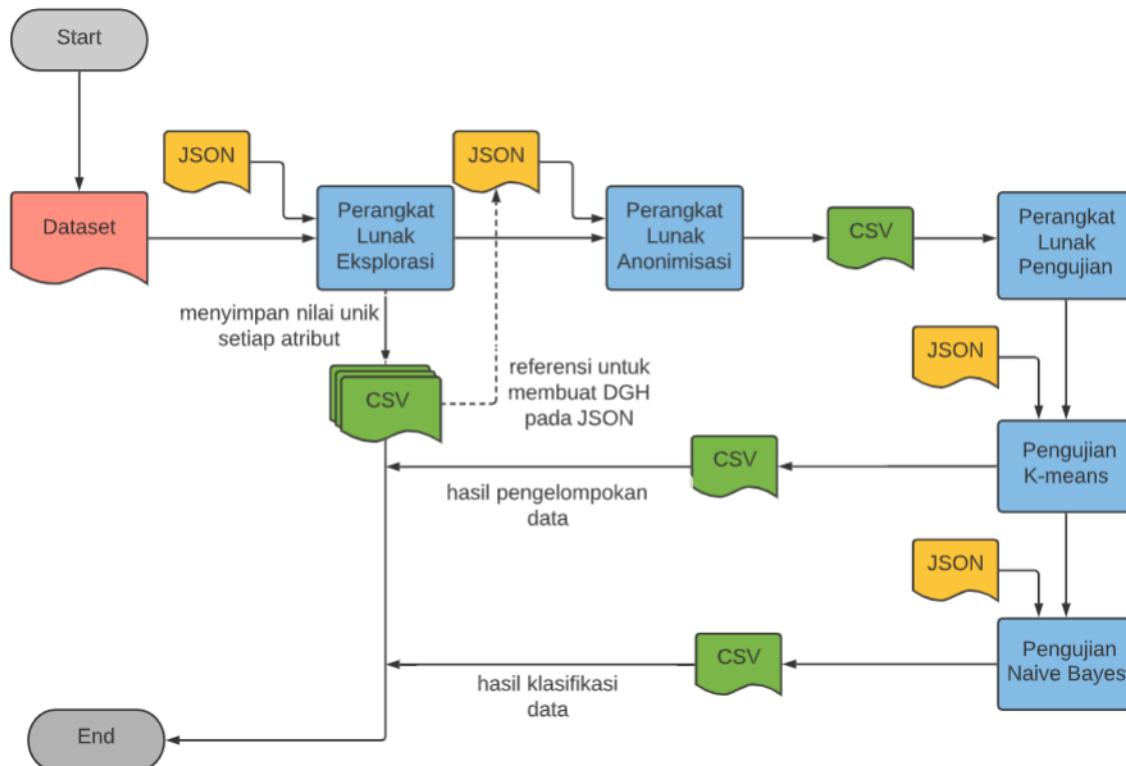
Gambar 3.16: Hasil K-Means Spark MLlib

Hasil pemodelan *k-means* dari eksperimen ini adalah *cluster* data. Cluster ini terbentuk berdasarkan distance terdekat antara anggota data dengan titik centroid pada cluster tersebut. Sehingga data-data yang tergabung pada sebuah cluster, sudah dipastikan bahwa data-data tersebut lebih dekat dengan titik centroid pada cluster tersebut dibanding dengan titik centroid pada cluster lainnya. Gambar 3.16 menunjukkan bahwa data dengan atribut (*age* = 39) memiliki kelompok *cluster* yang sama dengan dengan data lain dengan atribut (*age* = 38), karena berada pada kelompok data yang sama dengan representasi nama kelompok (*prediction* = 5).

### 3.4 Gambaran Umum Perangkat Lunak

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

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



Gambar 3.17: Flow Chart Penggunaan Perangkat Lunak

Pada Gambar 4.2 telah ditampilkan input dan output untuk masing-masing perangkat lunak. Persegi panjang berwarna biru diartikan sebagai program utama dan persegi panjang berwarna krem diartikan sebagai program pengujian. Bentuk dengan warna merah diartikan sebagai masukan perangkat lunak dalam bentuk CSV. Bentuk dengan warna hijau diartikan sebagai keluaran perangkat lunak dalam bentuk CSV. Bentuk dengan warna oranye diartikan sebagai input perangkat lunak dalam bentuk JSON. Pada bagian selanjutnya akan dijelaskan perancangan perangkat lunak dengan lebih detil. Penjelasan akan dibagi menjadi tiga bagian yaitu perancangan perangkat lunak eksplorasi, perancangan perangkat lunak anonimisasi, dan perancangan perangkat lunak pengujian.

### 3.4.1 Diagram Aktifitas

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

#### Perangkat Lunak Anonimisasi Data

Perangkat lunak ini bertujuan untuk melakukan proses anonimisasi pada dataset *Adult* menggunakan algoritma *k-anonymity*. Diagram aktifitas dapat dilihat pada Gambar 3.19, berikut adalah tahapan yang terjadi pada perangkat lunak saat melakukan proses anonimisasi data:

1. Pengguna memberi masukan dalam format file CSV dan beberapa jenis atribut *quasi-identifier* untuk menjadi tabel input pada proses anonimisasi.

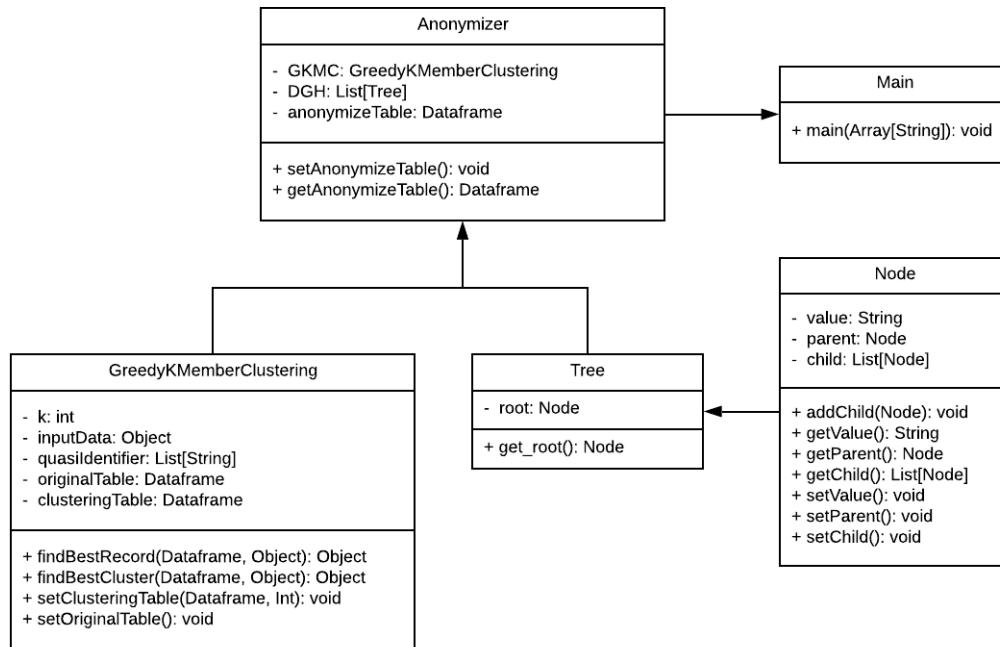
2. Perangkat lunak menampilkan sebagian baris data dari tabel input karena baris data yang akan digunakan pada eksperimen akan berjumlah sangat banyak .
3. Pengguna akan meninjau ulang apakah jumlah kolom yang ditampilkan sudah sesuai dengan jumlah atribut *quasi-identifier* yang akan dipakai.
4. Pengguna memberikan parameter tambahan seperti rentang nilai  $k$  untuk menentukan jumlah anggota *cluster* dan objek DGH untuk proses anonimisasi.
5. Perangkat lunak akan melakukan proses anonimisasi dengan bantuan Spark pada tabel input berdasarkan paramater tambahan yang diberikan sebelumnya.
6. Perangkat lunak mengembalikan seluruh isi *log* yang dihasilkan selama proses eksekusi Spark berlangsung kepada pengguna untuk deteksi *error*.
7. Perangkat lunak hanya menampilkan baris data yang berubah akibat proses anonimisasi pada GUI dan hasil keseluruhannya dalam format *file CSV*.
8. Perangkat lunak mengembalikan nilai *information loss* pada masing-masing *cluster* yang terbentuk agar pengguna dapat mencari hasil yang optimal.
9. Pengguna dapat membandingkan hasil anonimisasi antara baris data yang berubah akibat proses anonimisasi dengan baris data yang ada pada tabel asli.
10. Pengguna dapat mengulangi eksperimen untuk mencari nilai  $k$  terbaik agar dihasilkan *information loss* seminimal mungkin pada proses anonimisasi.

### Perangkat Lunak Analisis Data

Perangkat lunak ini bertujuan untuk mencari perbandingan hasil sebelum dan setelah data dilakukan proses anonimisasi dengan metode *data mining*. Diagram aktifitas dapat dilihat pada Gambar 3.20, berikut adalah tahapan yang terjadi pada perangkat lunak saat melakukan pemodelan *data mining*:

1. Pengguna memberi dua jenis masukan yaitu data asli dan data hasil anonimisasi dalam format *file CSV* untuk menjadi tabel input pada proses analisis data.
2. Perangkat lunak hanya menampilkan sebagian baris data dari dua jenis tabel input karena input baris data pada eksperimen berjumlah sangat banyak
3. Pengguna meninjau kembali apakah jumlah kolom yang ditampilkan pada kedua jenis tabel memiliki jumlah kolom atribut yang sama.
4. Pengguna memilih jenis pemodelan data mining yang tersedia pada eksperimen, yaitu klasifikasi dengan *Naive Bayes* atau pengelompokan/*clustering* dengan *k-means*.
5. 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.
6. Perangkat lunak akan melakukan proses pelatihan data pada Spark untuk menemukan klasifikasi/pengelompokan yang sesuai berdasarkan jenis pemodelan yang dipilih.
7. Perangkat lunak mengembalikan seluruh isi *log* yang dihasilkan selama proses eksekusi Spark berlangsung kepada pengguna untuk deteksi *error*.
8. Perangkat lunak menampilkan sebagian hasil prediksi *cluster* untuk masing-masing data dan menyimpan hasil keseluruhannya dalam format *file CSV*.
9. Pengguna melakukan analisis lebih lanjut terkait pengelompokan dan klasifikasi kelompok data yang terbentuk dari proses pemodelan *data mining*.

### 3.4.2 Diagram Kelas



Gambar 3.18: Diagram Kelas Anonimisasi Data

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

- Kelas *Anonymizer* bertujuan untuk melakukan proses anonimisasi setelah data dikelompokan 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.

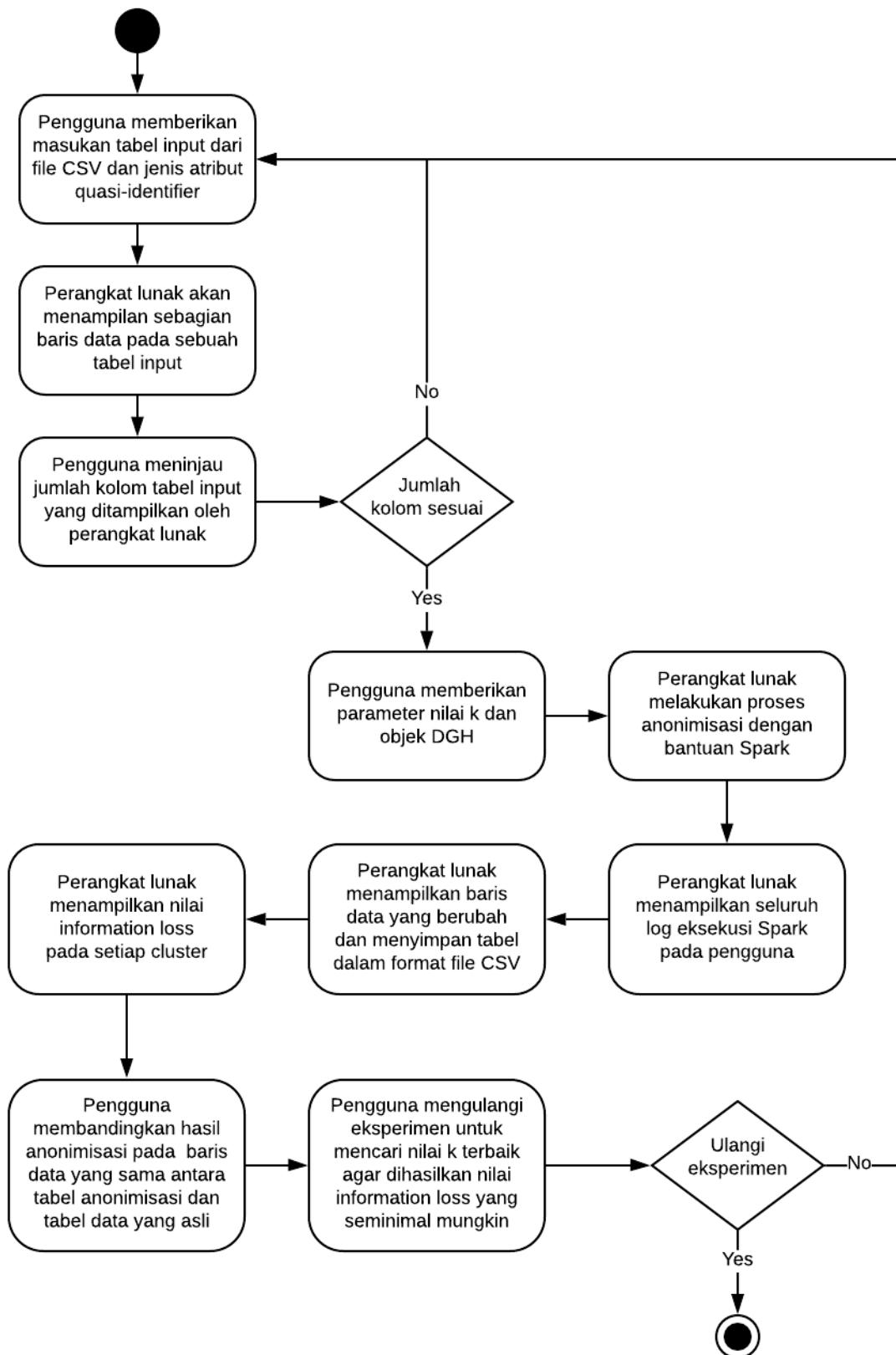
Kelas *Anonymizer* memiliki 2 jenis method, yaitu:

- *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:

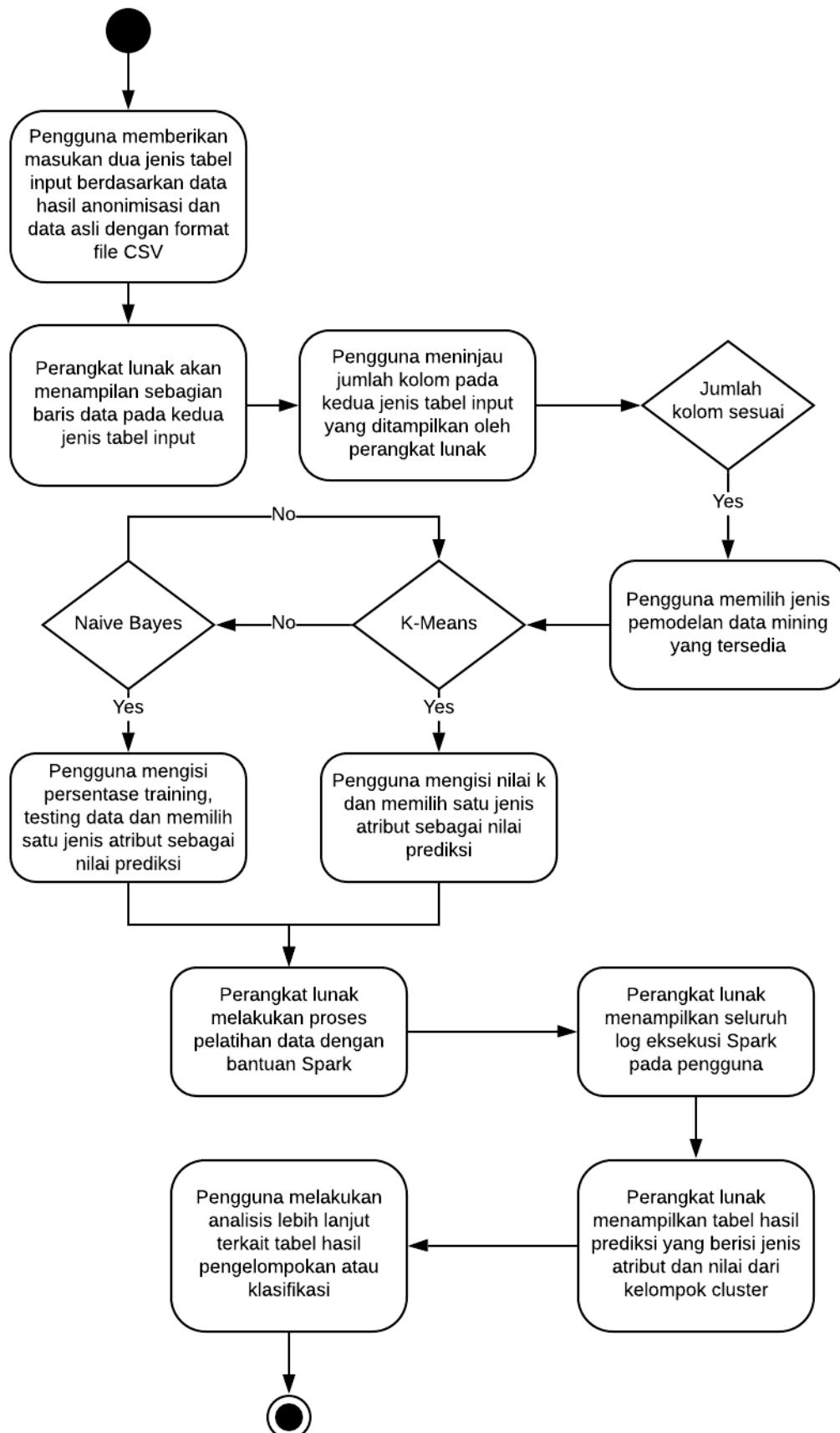
- $k$  adalah variabel bertipe *Integer* untuk membatasi jumlah anggota pada sebuah *cluster* agar memiliki jumlah yang tetap sebanyak jumlah tertentu.
- *inputData* adalah variabel untuk menyimpan seluruh baris data *file CSV*.
- *quasiIdentifier* adalah daftar dari nama-nama kolom yang akan dipilih 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.



Gambar 3.19: Diagram Aktifitas Anonimisasi Data



Gambar 3.20: Diagram Aktifitas Analisis Data

## BAB 4

# PERANCANGAN

Perancangan yang dibuat pada bab ini meliputi tiga perangkat lunak yaitu eksplorasi, anonimisasi, dan pengujian. Pada perangkat lunak eksplorasi akan dilakukan pencarian nilai unik untuk setiap atribut. Pada perangkat lunak anonimisasi akan dilakukan pemodelan algoritma greedy k-member clustering dan k-anonymity. Pada perangkat lunak pengujian akan dilakukan pemodelan k-means dan naive bayes. Pada tahap ini tidak dilakukan perancangan terhadap tampilan antarmuka karena program yang dihasilkan akan dijalankan dengan menggunakan command prompt.

Gambar 4.1: Flow Chart Penggunaan Perangkat Lunak

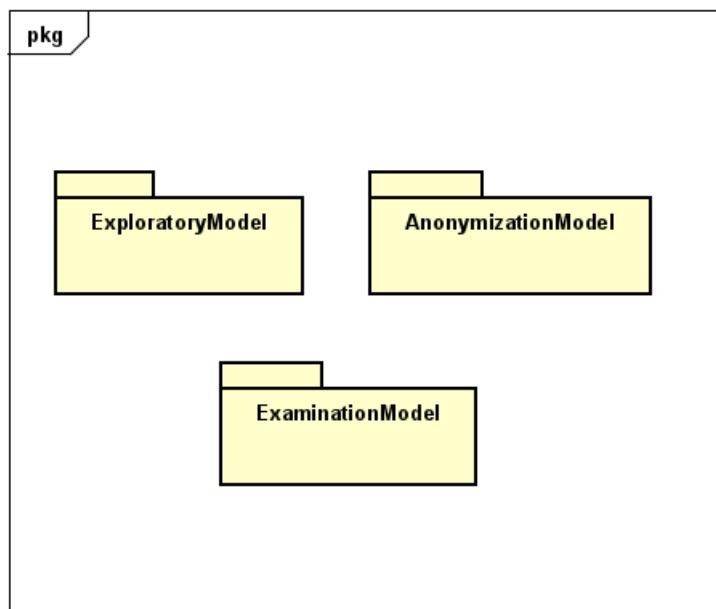
Gambar 4.2: Flow Chart Penggunaan Perangkat Lunak

## 4.1 Diagram Kelas Lengkap

Pada bagian ini, akan dibuat empat jenis diagram kelas, yaitu diagram kelas untuk package, diagram kelas untuk perangkat lunak eksplorasi, diagram kelas untuk perangkat lunak anonimisasi, dan diagram kelas untuk perangkat lunak pengujian.

### 4.1.1 Diagram Package

Perangkat lunak ini mempunyai 3 buah package yang tidak saling berhubungan satu sama lain yaitu ExploratoryModel, AnonymizationModel, ExaminationModel. Ketika package ini memiliki fungsinya masing-masing. Diagram Package dapat dilihat pada Gambar 4.3



Gambar 4.3: Diagram Kelas pada Package

#### Package ExploratoryModel

Package ExploratoryModel merupakan package yang menangani pencarian nilai unik pada masing-masing atribut. Package ini hanya memiliki implementasi kelas Main untuk mencari nilai atribut yang unik dan mengembalikan nilai unik tersebut ke dalam tabel data yang disimpan dalam format CSV. Jumlah CSV yang dihasilkan bergantung pada jumlah atribut yang dicantumkan pada JSON.

#### Package AnonymizationModel

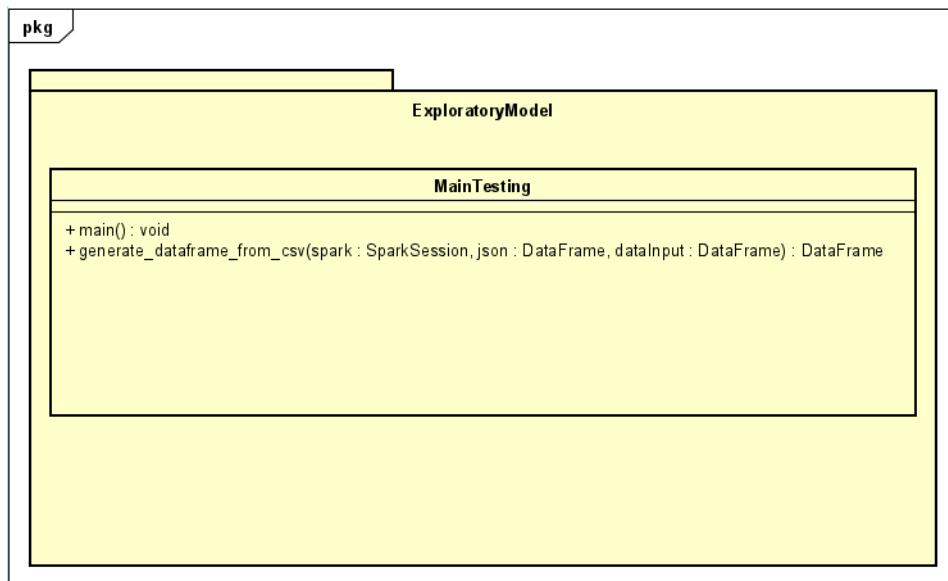
Package AnonymizationModel merupakan package yang menangani segala jenis fungsi yang berkaitan dengan masalah pengelompokan data dan anonimisasi data. Fungsi tersebut antara lain pengelompokan data dengan algoritma Greedy k-member clustering, pencarian record dan cluster terbaik, perhitungan distance numerik dan kategorikal, dan perhitungan information loss. Package ini juga mengimplementasikan berbagai macam operasi untuk membaca atribut DGH pada JSON dan menangani pembuatan binary tree berdasarkan atribut DGH. Operasi-operasi ini diimplementasikan karena adanya kebutuhan untuk membantu proses implementasi pengelompokan data dan anonimisasi data. Selain itu, package ini memiliki dua jenis kelas Main yaitu MainTesting dan MainLCATesting. Kelas MainTesting digunakan untuk melakukan proses pengelompokan dan anonimisasi data dan mengembalikan hasilnya ke dalam format CSV. Kelas MainLCATesting digunakan untuk menampilkan root terdekat dari kedua node.

## Package ExaminationModel

Package ExaminationModel merupakan package yang menangani segala jenis fungsi yang berkaitan dengan masalah pengujian data. Fungsi tersebut antara lain pengelompokan data dengan pemodelan k-means beserta evaluasi modelnya dengan silhouette score dan pemodelan naive bayes beserta evaluasi modelnya dengan accuracy. Package ini juga mengimplementasikan berbagai macam operasi untuk membuat DataFrame berdasarkan atribut yang telah dipilih pada JSON dan mendapatkan nilai parameter k-means dan naive bayes dari JSON. Operasi-operasi ini diimplementasikan karena adanya kebutuhan untuk membantu proses pengujian data berdasarkan pemodelan yang dipilih (k-means/naive bayes). Selain itu, package ini memiliki satu jenis kelas Main yaitu MainTesting. Kelas MainTesting digunakan untuk melakukan pengujian data dengan pemodelan k-means/naive bayes dan mengembalikan hasilnya ke dalam format CSV.

### 4.1.2 Diagram Kelas pada Package ExploratoryModel

Perangkat lunak eksplorasi hanya memiliki satu jenis *package* dengan nama *ExploratoryModel*. *Package ExploratoryModel* terdiri dari satu kelas yaitu MainTesting. Kelas ini memiliki fungsi penting untuk menyelesaikan permasalahan pencarian nilai atribut yang unik.



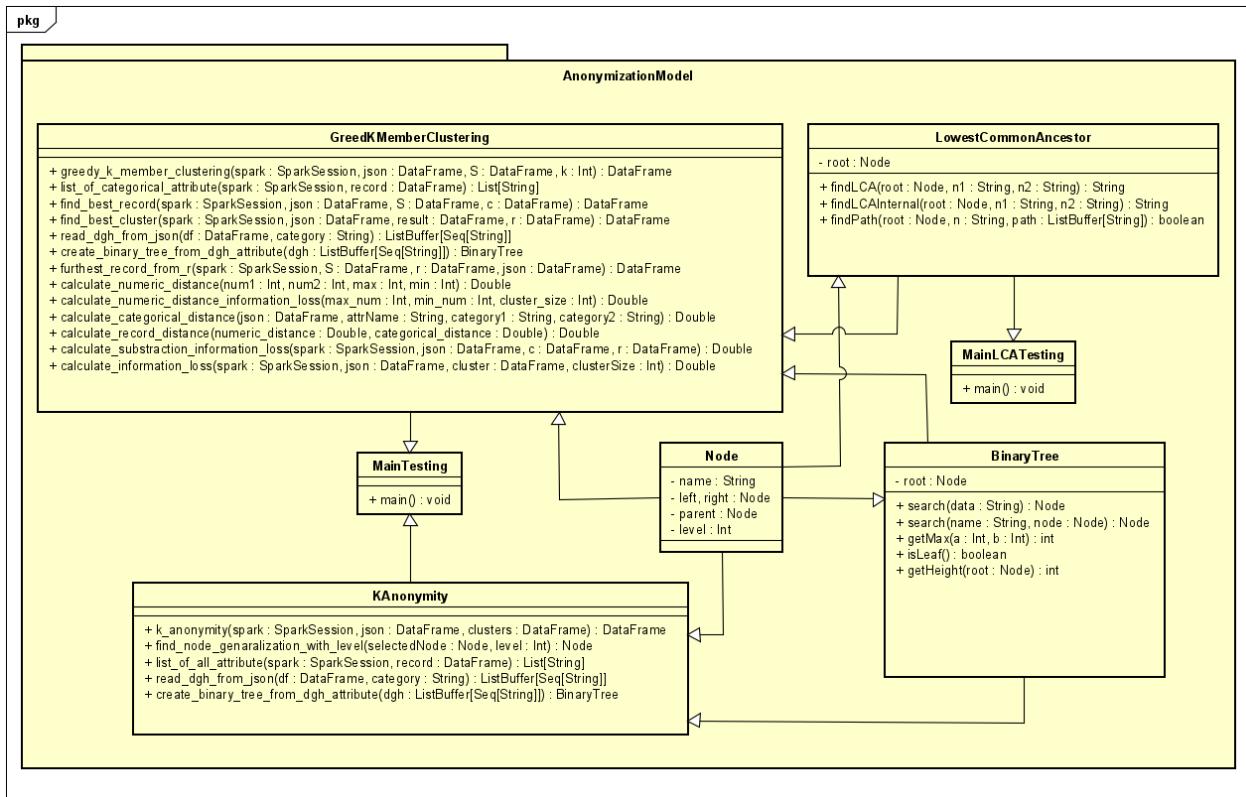
Gambar 4.4: Diagram Kelas pada Package ExploratoryModel

## Kelas *MainTesting*

Kelas *MainTesting* merupakan kelas dengan tipe *object*, karena kelas ini memiliki method Main untuk menjalankan eksekusi program. Kelas ini berperan penting untuk mencari nilai unik setiap atribut. Hasil eksekusi kelas ini akan menghasilkan output berupa beberapa jenis CSV yang bergantung kepada jumlah atribut yang ingin diketahui nilai uniknya dan atribut tersebut telah dicantumkan pada format JSON. Perlu diketahui bahwa kelas ini tidak memiliki atribut dan hanya memiliki sebuah method dengan nama *generate\_dataframe\_from\_csv* untuk membuat DataFrame berdasarkan atribut yang dipilih dan telah dicantumkan dalam format JSON.

### 4.1.3 Diagram Kelas pada Package AnonymizationModel

Perangkat lunak anonimisasi hanya memiliki satu jenis *package* bernama *AnonymizationModel*. *Package AnonymizationModel* terdiri dari beberapa kelas. Masing-masing kelas memiliki fungsi penting menyelesaikan permasalahan pengelompokan dan anonimisasi data.



Gambar 4.5: Diagram Kelas pada Package AnonymizationModel

### Kelas Node

Kelas *Node* merupakan kelas dengan tipe *class*, karena kelas ini berfungsi sebagai model untuk membuat atribut dan *method* pada objek *Node*. Kelas ini bertujuan untuk menyimpan informasi seperti nama *node*, menyatakan *node* kiri dan *node* kanan dari *node* tersebut, menyatakan *parent* dari *node* tersebut, dan *level* yang menyatakan posisi ketinggian *node* pada objek *BinaryTree*. Kelas *Node* nantinya akan dipakai untuk pembuatan objek *BinaryTree*.

Berikut adalah penjelasan masing-masing atribut pada kelas *Node*:

- **name** adalah variabel yang berfungsi untuk menyimpan nilai atribut tertentu pada sebuah tabel data dengan tipe *String* dan termasuk jenis variabel *var* sehingga atribut ini nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.
- **left** adalah variabel yang berfungsi untuk menyimpan node kiri dari node tersebut dengan tipe *Node* dan termasuk jenis variabel *var* sehingga atribut ini nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.
- **right** adalah variabel yang berfungsi untuk menyimpan node kanan dari node tersebut dengan tipe *Node* dan termasuk jenis variabel *var* sehingga atribut ini nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.
- **parent** adalah variabel yang berfungsi untuk menyimpan node *parent* dari node tersebut dengan tipe *Node* dan termasuk jenis variabel *var* sehingga atribut ini nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.
- **level** adalah variabel yang berfungsi untuk menyimpan posisi ketinggian *BinaryTree* untuk node tersebut dengan tipe *Integer* dan termasuk jenis variabel *var* sehingga atribut ini nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.

### Kelas *BinaryTree*

Kelas *BinaryTree* merupakan kelas dengan tipe *class*, karena kelas ini berfungsi sebagai model untuk membuat atribut dan method pada objek *BinaryTree*. Kelas ini nantinya akan dipakai untuk pembuatan objek *BinaryTree* berdasarkan *Domain Generalization Hierarchy* (DGH).

Berikut deskripsi atribut pada kelas *BinaryTree*:

- *root* adalah variabel yang berfungsi untuk menyimpan node yang menjadi root pada objek *BinaryTree* dengan tipe *Node* dan termasuk jenis variabel *var* sehingga atribut *root* nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.

Berikut deskripsi atribut pada kelas *BinaryTree*:

- *search* dengan parameter *data(String)* adalah fungsi yang akan memanggil fungsi *search* lain dengan parameter *name(String)*, *node(Node)* untuk mengembalikan objek *Node* yang ingin dicari pada objek *BinaryTree*.
- *search* dengan parameter *name(String)*, *node(Node)* adalah fungsi yang bertujuan untuk mengembalikan objek *Node* yang ingin dicari pada objek *BinaryTree*.
- *getMax* dengan parameter *a(Integer)*, *b(Integer)* adalah fungsi yang bertujuan untuk mengembalikan nilai *Integer* paling besar antara parameter *a* dan *b*.
- *isLeaf* tanpa parameter adalah fungsi yang bertujuan untuk mengembalikan nilai *Boolean* untuk menyatakan apakah root dari node kiri dan kanan bernilai *null*. Jika kondisi terpenuhi maka nilainya *true*, apabila tidak terpenuhi maka nilainya *false*.
- *getHeight* dengan parameter *root(Node)* adalah fungsi yang bertujuan untuk mengembalikan nilai *Integer* untuk menyatakan ketinggian dari objek *BinaryTree*.

Berikut implementasi method *search* pada kelas *BinaryTree*:

---

#### **Algorithm 4** Mencari Node dengan Nama Tertentu

---

```

1: Function search(name,node)
2: Input: a name of node (name) and a node.
3: Output: a node with selected name.
4:
5: if node != null then
6:   if node.name == name then
7:     return node
8:   else
9:     foundNode = search(name,node.left)
10:    if foundNode == null then
11:      foundNode = search(name,node.right)
12:    end if
13:    return foundNode
14:  end if
15: else
16:   return null
17: end if

```

---

- Baris 6-8: baris ini mengembalikan sebuah node, jika nama sebuah node sudah sesuai.
- Baris 9-14: baris ini melakukan proses rekursif untuk mencari nama node yang sesuai.

### Kelas *LowestCommonAncestor*

Kelas *LowestCommonAncestor* merupakan kelas dengan tipe *class*. Hal ini dikarenakan kelas *LowestCommonAncestor* berfungsi sebagai model untuk membuat atribut dan method-method pada objek *LowestCommonAncestor*. Kelas *LowestCommonAncestor* bertujuan untuk melakukan pencarian node root terdekat dari kedua node. Kelas *LowestCommonAncestor* nantinya akan dipakai untuk mencari level node root terdekat untuk menghitung distance kategorikal.

Berikut deskripsi atribut pada kelas *LowestCommonAncestor*:

- **path1** adalah variabel yang berfungsi untuk menyimpan list node yang pernah dipilih sebelumnya dengan tipe *ListBuffer[String]* dan termasuk jenis variabel *var* sehingga atribut *path1* nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.
- **path2** adalah variabel yang berfungsi untuk menyimpan list node yang pernah dipilih sebelumnya dengan tipe *ListBuffer[String]* dan termasuk jenis variabel *var* sehingga atribut *path1* nilainya bersifat *mutable*/dapat diubah sesuai kebutuhan.

Berikut deskripsi method pada kelas *LowestCommonAncestor*:

- **findLCA** dengan parameter *root (Node)*, *n1 (String)*, *n2 (String)* adalah fungsi yang bertujuan untuk mereset nilai *path1* dan *path2* dan memanggil fungsi *findLCAInternal* untuk mencari nama root paling bawah yang mengandung kedua node.
- **findLCAInternal** dengan parameter *root (Node)*, *n1 (String)*, *n2 (String)* adalah fungsi yang bertujuan untuk mencari nama root paling bawah yang mengandung kedua node. Caranya dengan menambahkan indeks pada setiap iterasi, jika nama node pada *path1* sama dengan nama node pada *path2*. Indeks ini nantinya dipakai untuk mengambil nama node pada *path1* sebagai hasil output dari algoritma Lowest Common Ancestor.
- **findPath** dengan parameter *root (Node)*, *n (String)*, *path (ListBuffer[String])* adalah fungsi yang bertujuan untuk menambahkan node yang pernah diproses sebelumnya pada method ini sebagai elemen untuk array *path 1* dan *path 2*. Method ini akan mengembalikan nilai true, jika nama node yang diberikan sama dengan nama yang dicari.

Berikut implementasi method *findLCAInternal* pada kelas *LowestCommonAncestor*:

---

#### **Algorithm 5** Mencari Nama Root Terdekat dengan Kedua Node

---

```

1: Function findLCAInternal(root,namenode1,namenode2)
2: Input: root node (root), name of node 1 (n1), name of node 2 (n2).
3: Output: a name of root node.
4:
5: if (!findPath(root,namenode1,path1) or !findPath(root,namenode2,path2))
   then
6:     return "not found"
7: end if
8:
9: i = 0
10:
11: while (i < path1.size and i < path2.size) do
12:     if !path1(i).equals(path2(i)) then
13:         i += 1
14:     end if
15: end while
16: return path1(i-1)

```

---

- Baris 5-7: baris ini memeriksa apakah nama node 1 dan nama node 2 ada pada objek Binary Tree, jika tersedia maka method akan mengembalikan nilai true.
- Baris 9: baris ini digunakan untuk mengambil nama node yang pernah dikunjungi sebelumnya pada indeks ke-i dari elemen array pada `path1` dan elemen array pada `path2`.
- Baris 11-15: baris ini melakukan perulangan untuk mencari nama node root paling rendah, dengan objek node 1 dan node 2 sebagai anak dari node root tersebut.
- Baris 12-14: baris ini mencari nama root terdekat dengan menambahkan indeks pada setiap iterasi, jika node 1 memiliki nama yang sama dengan node 2.
- Baris 16: pada baris ini, iterasi yang diperoleh akan dipakai untuk menyatakan nama dari node root terdekat antara node 1 dan node 2.

Berikut implementasi method `findPath` pada kelas *LowestCommonAncestor*:

---

**Algorithm 6** Mencari Node yang Pernah Dilalui Sebelumnya

---

```

1: Function findPath(root, namenode, path)
2: Input: root node (root), name of node (n), path of node.
3: Output: true/false.
4:
5: if (root == null) then
6:     return false
7: end if
8: path += root.name
9:
10: if (root.name == namenode) then
11:     return true
12: end if
13:
14: if (root.left != null and findPath(root.left, n, path)) then
15:     return true
16: end if
17:
18: if (root.right != null and findPath(root.right, n, path)) then
19:     return true
20: end if
21: path.remove(path.size - 1)
22: return false

```

---

- Baris 5-7: baris ini akan mengembalikan nilai *false* jika *node* sudah kosong.
- Baris 9: baris ini akan mencatat setiap *node* yang pernah dikunjungi ke dalam variabel *path*
- Baris 11-13: baris ini akan mengembalikan nilai *true* jika node yang dikunjungi memiliki nama yang sama dengan nama yang ingin dicari
- Baris 15-17: baris ini akan memeriksa apakah setiap *node* kiri dari sebuah node memiliki nama yang dicari, jika benar maka akan mengembalikan nilai *true*.
- Baris 19-21: baris ini akan memeriksa apakah setiap *node* kanan dari sebuah *node* memiliki nama yang dicari, jika benar maka akan mengembalikan nilai *true*.
- Baris 23: setiap kali method ini dipanggil, maka *node* paling terakhir akan dihapus dari kumpulan node yang sudah pernah dicatat sebelumnya pada variabel *path*.

### Kelas *GreedyKMemberClustering*

Kelas *GreedyKMemberClustering* merupakan kelas dengan tipe *class*. Hal ini dikarenakan kelas *GreedyKMemberClustering* berfungsi sebagai model untuk membuat atribut dan method pada objek *GreedyKMemberClustering*. Kelas *GreedyKMemberClustering* bertujuan untuk melakukan fungsi pengelompokan data. Kelas *GreedyKMemberClustering* tidak memiliki atribut.

Berikut deskripsi method pada kelas *GreedyKMemberClustering*:

- **greedy\_k\_member\_clustering** dengan parameter `spark(SparkSession)`, `json(DataFrame)`, `S(DataFrame)`, `k(Integer)` adalah fungsi yang bertujuan untuk melakukan pengelompokan data dengan algoritma Greedy k-member clustering sebelum data dilakukan anonimisasi. Kelompok data yang terbentuk akan didasari pada selisih information loss paling minimum antar masing-masing data berdasarkan fungsi `find_best_record` dan fungsi `find_best_cluster`.
- **list\_of\_categorical\_attribute** dengan parameter `spark(SparkSession, record (DataFrame))` adalah fungsi yang bertujuan untuk mendapatkan seluruh nama atribut dari tabel data. Fungsi lain seperti `furthest_record_from_r`, `calculate_information_loss` akan memanggil fungsi ini untuk mendapatkan tabel data yang berisi kolom-kolom bertipe kategorikal, sebelum tabel data dipakai untuk mencari record paling jauh dengan record r dan sebelum tabel data dipakai untuk menghitung nilai information loss pada kelompok data.
- **find\_best\_record** dengan parameter `spark(SparkSession)`, `json(DataFrame)`, `S(DataFrame)`, `c(DataFrame)` adalah fungsi yang bertujuan untuk mencari setiap record dari tabel data yang memiliki nilai information loss paling minimum terhadap record c.
- **find\_best\_cluster** dengan parameter `spark(SparkSession)`, `json(DataFrame)`, `S (DataFrame)`, `r(DataFrame)` adalah fungsi yang bertujuan untuk mencari setiap record dari tabel data yang telah dikelompokan dengan nilai information loss paling rendah terhadap sebuah record r, dimana r adalah record yang belum masuk kelompok data tertentu.
- **read\_dgh\_from\_json** dengan parameter `df(DataFrame)`, `category(String)` adalah fungsi yang bertujuan untuk mengembalikan nilai `value`, `parent`, `level`, `position` untuk setiap nama atribut tabel data pada `domain_generaliation_hierarchy` JSON.
- **create\_binary\_tree\_from\_dgh\_attribute** dengan parameter `dgh(ListBuffer[Seq[String]])`, `category(String)` adalah fungsi yang bertujuan untuk membuat pohon DGH melalui representasi objek `BinaryTree` pada `domain_generaliation_hierarchy` JSON.
- **furthest\_record\_from\_r** dengan parameter `spark(SparkSession)`, `S(DataFrame)`, `r (DataFrame)`, `json(DataFrame)` adalah fungsi yang bertujuan untuk menghitung distance record antara record r dengan masing-masing record pada tabel data. Setelah dilakukan perhitungan, record dengan nilai distance record tertinggi akan dikembalikan sebagai output dari fungsi ini.
- **calculate\_numeric\_distance** dengan parameter `num1(Int)`, `num2(Int)`, `max(Int)`, `min(Int)` adalah fungsi yang bertujuan untuk menghitung distance numerik antara 2 data numerik dengan atribut yang sejenis. Fungsi ini dipanggil oleh fungsi lain yaitu `furthest_record_from_r`, agar hasil perhitungan distance numerik dapat digunakan untuk menghitung distance record.
- **calculate\_numeric\_distance\_information\_loss** dengan parameter `max_num(Integer)`, `min_num(Integer)`, `cluster_size(Integer)` adalah fungsi yang bertujuan untuk menghitung distance numerik pada kasus information loss antar 2 data numerik di atribut yang sejenis. Fungsi ini dipanggil oleh fungsi lain yaitu `calculate_information_loss`, agar hasil perhitungan distance numerik dapat digunakan untuk menghitung nilai information loss pada sebuah kelompok data untuk mendapatkan hasil pengelompokan data yang terbaik.

- `calculate_categorical_distance` dengan parameter `json(DataFrame)`, `attrName(String)`, `category1(String)`, `category2(String)` adalah fungsi yang bertujuan untuk menghitung distance kategorikal antara 2 data kategori dengan atribut yang sejenis. Fungsi ini dipanggil oleh fungsi `furthest_record_from_r` untuk menghitung distance record.
- `calculate_record_distance` dengan parameter `numeric_distance(Double)`, `categorical_distance (Double)` adalah fungsi yang bertujuan untuk menghitung distance record antara 2 record. Fungsi ini dipanggil oleh fungsi `furthest_record_from_r` untuk mencari record dengan distance record paling kecil. Distance record didapat dengan menjumlahkan total distance numerik dan total distance kategorikal.
- `calculate_subtraction_information_loss` dengan parameter `spark (SparkSession)`, `json(DataFrame)`, `c(DataFrame)`, `r(DataFrame)` adalah fungsi yang bertujuan untuk mencari selisih information loss antara dua kelompok tabel data yang berbeda. Fungsi ini dipanggil oleh fungsi lain seperti `find_best_record` dan `find_best_cluster` untuk mencari record terbaik berdasarkan selisih information loss paling kecil.
- `calculate_information_loss` dengan parameter `spark(SparkSession)`, `json(DataFrame)`, `cluster(DataFrame)`, `clusterSize(Integer)` adalah fungsi yang bertujuan untuk menghitung nilai information loss pada kelompok tabel data tertentu. Fungsi ini dipanggil oleh fungsi lain seperti `calculate_subtraction_information_loss` untuk mencari selisih information loss antara dua kelompok tabel data yang berbeda.

Berikut implementasi method `calculate_categorical_distance`:

---

#### **Algorithm 7** Menghitung Distance Kategorikal

---

```

1: Function calculate_categorical_distance(json, attrName, cat1, cat2)
2: Input: numeric 1, numeric 2, maximum number, maximum number.
3: Output: numeric distance.
4:
5: result = 0.0
6: dgh = read_dgh_from_json(json, attrName)
7:
8: if (dgh == null) then
9:     return result
10: end if
11:
12: binaryTree = create_binary_tree_from_dgh_attribute(dgh)
13: node1 = binaryTree.search(category1)
14: node2 = binaryTree.search(category2)
15:
16: if (node1 != null and node2 != null) then
17:     LCA = new LowestCommonAncestor()
18:     resultLCA = LCA.findLCA(binaryTree.root, node1.name, node2.name)
19:     H_subtree = binaryTree.search(resultLCA).level
20:     H_TD = binaryTree.getHeight(binaryTree.root).toDouble
21:     result = H_subtree / H_TD
22: end if
23:
24: result = BigDecimal(result)
25: result = result.setScale(2, BigDecimal.RoundingMode.HALF_UP)
26: result = result.toDouble
27: return result

```

---

- Baris 6: baris ini mendapatkan nilai `value, parent, level, position` untuk setiap nama atribut pada `domain_generalization_hierarchy` JSON.
- Baris 8-10: baris ini memeriksa jika nilai pada `domain_generalization_hierarch` JSON kosong, maka perhitungan distance kategorikal tidak perlu dilakukan.
- Baris 12: baris ini membuat objek `BinaryTree` berdasarkan atribut `dgh` pada baris 6.
- Baris 13: baris ini mendapatkan objek `Node` pertama dengan parameter nama tertentu.
- Baris 14: baris ini mendapatkan objek `Node` kedua dengan parameter nama tertentu.
- Baris 16-22: baris ini menghitung *distance* kategorikal jika kedua node tidak kosong.
- Baris 19: baris ini menyimpan ketinggian `node root` dari objek `LowestCommonAncestor`
- Baris 20: baris ini menyimpan ketinggian maksimal dari objek `BinaryTree`
- Baris 24-26: baris ini mengembalikan hasil perhitungan *distance*kategorikal dalam format desimal, dengan ketelitian 2 angka dibelakang koma.

Berikut implementasi method `calculate_numeric_distance`:

---

#### **Algorithm 8** Menghitung Distance Numerik

---

```

1: Function calculate_numeric_distance(num1, num2, max, min)
2: Input: numeric 1, numeric 2, maximum number, maximum number.
3: Output: numeric distance.
4:
5: v1 = num1
6: v2 = num1
7: diff = Math.abs(v1-v2)
8: range = Math.abs(max-min)
9: result = diff/range
10: return result

```

---

- Baris 5: baris ini melakukan inisialisasi `v1` dengan nilai `num1`.
- Baris 6: baris ini melakukan inisialisasi `v2` dengan nilai `num2`.
- Baris 7: baris ini menghitung selisih perbedaan nilai antara `v1` dan `v2`.
- Baris 8: baris ini menghitung selisih range nilai antara `v1` dan `v2`.
- Baris 9: baris ini membagi selisih perbedaan nilai dengan selisih *range* nilai.

Berikut implementasi method `calculate_record_distance`:

---

#### **Algorithm 9** Menghitung Distance Record

---

```

1: Function calculate_record_distance(num_dist, cat_dist)
2: Input: numeric 1, numeric 2, maximum number, maximum number.
3: Output: numeric distance.
4:
5: result = numeric_distance+categorical_distance
6: return result

```

---

- Baris 5: baris ini menjumlahkan total *distance* numerik masing-masing atribut dan total *distance* kategorikal masing-masing atribut.
- Baris 6: baris ini mengembalikan hasil penjumlahannya sebagai *distance record*.

Berikut implementasi method `furthest_record_from_r`:

---

**Algorithm 10** Mencari Record Paling Jauh dari Record Tertentu
 

---

```

1: Function furthest_record_from_r(spark,S,r,json)
2: Input: sparkSession(spark), all records(S), a record(r), json.
3: Output: furthest record from record r.
4:
5: list_record_distance = ()
6: domain = S.cache()
7: r_values = r.first().toSeq
8: categorical_name = list_of_categorical_attribute(spark,r)
9: index = 0
10:
11: while (!domain.isEmpty) do
12:   selected_record = domain.limit(1).cache()
13:   record_values = selected_record.first().toSeq
14:   column_size = record_values.length-1
15:   record_distance = 0
16:   record_id = selected_record.select("id").first().getInt(0)
17:   for (i = 0 to column_size) do
18:     if (record_values(i).isInstanceOf[Int]) then
19:       num1 = record_values(i).toString.toInt
20:       num2 = r_values(i).toString.toInt
21:       max = domain.groupBy().max(domain.columns(i)).first().getInt(0)
22:       min = domain.groupBy().min(domain.columns(i)).first().getInt(0)
23:       record_distance += calculate_numeric_distance(num1,num2,max,min)
24:     else
25:       cat1 = record_values(i).toString
26:       cat2 = r_values(i).toString
27:       attrName = categorical_name(index)
28:       record_distance += calculate_categorical_distance(json,attrName,cat1,cat2)
29:       index += 1
30:     end if
31:   end for
32:   list_record_distance +=((record_id,record_distance))
33:   domain.unpersist()
34:   domain = domain.except(selected_record)
35:   index = 0
36: end while
37: return false
  
```

---

- Baris 8: baris ini mendapatkan nilai masing-masing kolom untuk *record r* yang dipilih.
- Baris 12-38: baris ini melakukan perulangan setiap baris data sampai domain kosong.
- Baris 19-33: baris ini melakukan perulangan setiap kolom dengan menghitung distance *record*.
- Baris 20-25: baris ini melakukan perhitungan distance numerik, lalu hasil perhitungannya akan ditambahkan pada nilai atribut `record_distance` sebelumnya.
- Baris 26-32: baris ini melakukan perhitungan *distance* kategorikal, lalu hasil perhitungannya akan ditambahkan pada nilai atribut `record_distance` sebelumnya.

Berikut implementasi method `calculate_information_loss`:

---

**Algorithm 11** Menghitung Information Loss

---

```

1: Function calculate_information_loss(spark,json,cluster,clusterSize)
2: Input: numeric 1, numeric 2, maximum number, maximum number.
3: Output: numeric distance.
4:
5: cluster_temp = cluster.cache()
6: informationLoss = 0
7: categorical_name = list_of_categorical_attribute(spark,cluster_temp)
8: index = 0
9:
10: record_values = cluster_temp.first().toSeq
11: column_size = record_values.length-1
12: for (i <- 0 to column_size) do
13:   if (record_values(i).isInstanceOf[Int]) then
14:     max = cluster_temp.groupBy().max(cluster_temp.columns(i)).first().getInt(0)
15:     min = cluster_temp.groupBy().min(cluster_temp.columns(i)).first().getInt(0)
16:     informationLoss += calculate_numeric_distance_IL(max,min,clusterSize)
17:   else
18:     attrName = categorical_name(index)
19:     distinctValues = cluster_temp.select(attrName).distinct().cache()
20:     countDistinctValues = distinctValues.count()
21:     index += 1
22:     if (countDistinctValues == 2) then
23:       record1 = distinctValues.limit(1).cache()
24:       record2 = distinctValues.except(record1)
25:       cat1 = record1.first().getString(0)
26:       cat2 = record2.first().getString(0)
27:       informationLoss += calculate_categorical_distance(json,attrName,cat1,cat2)
28:       distinctValues.unpersist()
29:       record1.unpersist()
30:     else
31:       informationLoss += 0
32:     end if
33:   end if
34: end for
35: cluster_temp.unpersist()
36: return result

```

---

- Baris 7: baris ini mendapatkan seluruh nama atribut dari tabel data.
- Baris 10: baris ini mendapatkan nilai dari masing-masing kolom tabel data.
- Baris 12-34: baris ini melakukan perulangan untuk setiap kolom tabel data.
- Baris 13-16: baris ini menghitung *distance* numerik apabila kolom termasuk nilai numerik.
- Baris 17-33: baris ini menghitung *distance* kategorikal apabila kolom termasuk nilai kategori.
- Baris 22-29: baris ini memeriksa jika nilai unik sebuah kolom kategorikal terdiri dari 2 nilai unik, maka *distance* kategorikal dihitung menggunakan fungsi *Lowest Common Ancestor*.

### Kelas *KAnonymity*

Kelas *KAnonymity* merupakan kelas dengan tipe *class*. Hal ini dikarenakan kelas *GreedyKMember-Clustering* berfungsi sebagai model untuk membuat atribut dan method pada objek *KAnonymity*. Kelas *KAnonymity* bertujuan untuk melakukan anonimasasi k-anonymity pada data-data yang sudah dikelompokan. Kelas *KAnonymity* tidak memiliki atribut.

Berikut deskripsi method pada kelas *KAnonymity*:

- **k\_anonymity** dengan parameter *spark(SparkSession)*, *json(DataFrame)*, *clusters(DataFrame)* adalah fungsi yang bertujuan untuk melakukan anonimisasi data dengan algoritma k-anonymity setelah tabel data berhasil dilakukan pengelompokan data.
- **find\_node\_genaralization\_with\_level** dengan parameter *selectedNode(Node)*, *level(Int)* adalah fungsi yang bertujuan untuk mencari node dengan tingkatan generalisasi tertentu pada objek *BinaryTree*. Tingkatan generalisasi ditentukan dari ketinggian node pada objek *BinaryTree*. Semakin kecil ketinggian node, maka nilainya semakin umum.
- **list\_of\_all\_attribute** dengan parameter *spark(SparkSession)*, *record(DataFrame)* adalah fungsi yang bertujuan untuk mendapatkan seluruh nama kolom pada sebuah tabel data. Fungsi ini akan dipanggil pada fungsi lain, yaitu *k\_anonymity* untuk mengganti nilai sesungguhnya dengan nilai yang lebih umum pada proses anonimisasi.
- **read\_dgh\_from\_json** dengan parameter *df(DataFrame)*, *category(String)* adalah fungsi yang bertujuan untuk mengembalikan nilai *value*, *parent*, *level*, *position* untuk beberapa nama kolom pada domain\_generaliation\_hierarchy JSON.
- **create\_binary\_tree\_from\_dgh\_attribute** dengan parameter *dgh(ListBuffer[Seq[String]])* adalah fungsi yang bertujuan mengambil domain\_generaliation\_hierarchy pada JSON untuk membuat objek *BinaryTree*. Fungsi ini akan dipanggil pada fungsi lain, yaitu *k\_anonymity* untuk mengambil nilai yang lebih umum pada pohon DGH.

Berikut implementasi method *k\_anonymity*:

- Baris 7: baris ini mendapatkan seluruh nama atribut dari tabel data.
- Baris 9-48: baris ini melakukan perulangan selama *cluster\_temp* tidak kosong.
- Baris 11-12: baris ini mendapatkan data berdasarkan nama cluster tertentu.
- Baris 16-40: baris ini melakukan perulangan untuk untuk setiap kolom tabel cluster tertentu.
- Baris 17: baris ini mendapatkan nilai unik dari sebuah kolom data pada nama cluster tertentu.
- Baris 21-25: melakukan anonimisasi kolom numerik dengan rentang nilai kolom pada cluster.
- Baris 26-38: melakukan anonimisasi kolom kategorikal dengan nama root pada pohon DGH.
- Baris 28-31: baris ini memeriksa jika nilai unik kategorikal pada sebuah kolom melebihi 1 nilai unik, maka hasil anonimisasinya adalah nama root dari pohon DGH.
- Baris 32-37: baris ini memeriksa jika nilai unik kategorikal pada sebuah kolom hanya 1 nilai unik, maka hasil anonimisasinya adalah nilai kategorikal tersebut.
- Baris 41-42: baris ini menginisialisasi baris data hasil anonimisasi untuk pertama kali.
- Baris 43-44: baris ini menggabungkan hasil anonimisasi pada baris-baris data sebelumnya dengan hasil anonimisasi pada baris data yang baru.

---

**Algorithm 12** Melakukan Anonimisasi Data dengan K-Anonymity

---

```

1: Function k_anonymity()
2: Input: spark, json, clusters.
3: Output: table with anonymization data.
4:
5: result = spark.emptyDataFrame
6: clusters_temp = clusters
7: columnName = list_of_all_attribute(spark,clusters_temp)
8: column_size = columnName.length-1
9: while (!clusters_temp.isEmpty) do
10:   clusterName = clusters_temp.select("Cluster").first().getString(0)
11:   clusterDF = clusters_temp.where(clusters_temp("Cluster"))
12:   clusterDF = clusterDF.contains(clusterName).cache()
13:   clusterAnonymization = clusterDF.select("id")
14:   recordDistinctValues = spark.emptyDataFrame
15:   numDistValues = 0
16:   for (i <- 0 to column_size) do
17:     recordDistinctValues = clusterDF.select(columnName(i)).distinct().cache()
18:     numDistinctValues = recordDistinctValues.count().toInt
19:     colName = columnName(i)
20:     colValues = lit(generalizationNumeric))
21:     if (numDistValues > 1 and recordDistinctValues.isInstanceOf[Int] then
22:       maxValue = recordDistinctValues.groupBy().max()
23:       minValue = recordDistinctValues.groupBy().min()
24:       generalizationNumeric = "["+minValue+"-"+maxValue+"]"
25:       clusterAnonym = clusterAnonym.withColumn(columnName, columnValues
26:     else
27:       dgh = read_dgh_from_json(json, columnName(i))
28:       if (numDistValues > 1 and recordDistinctValues.isInstanceOf[String] then
29:         binaryTree = create_binary_tree_from_dgh_attribute(dgh)
30:         generalizationCategorical = binaryTree.root.name
31:         clusterAnonym = clusterAnonym.withColumn(colName, colValues
32:       else
33:         column = clusterDF.select("id", columnName(i))
34:         column = column.withColumnRenamed("id", "id_temp")
35:         clusterAnonym = clusterAnonym.join(column)
36:         clusterAnonym = clusterAnonym.drop("id_temp")
37:       end if
38:     end if
39:     recordDistinctValues.unpersist()
40:   end for
41:   if (result.isEmpty) then
42:     result = clusterAnonym
43:   else
44:     result = result.union(clusterAnonym)
45:   end if
46:   clusters_temp.unpersist()
47:   clusters_temp = clusters_temp.except(clusterDF)
48: end while
49: result = result.drop("Cluster")
50: result = result.orderBy(asc("id"))
51: return result

```

---

### Kelas *MainTesting*

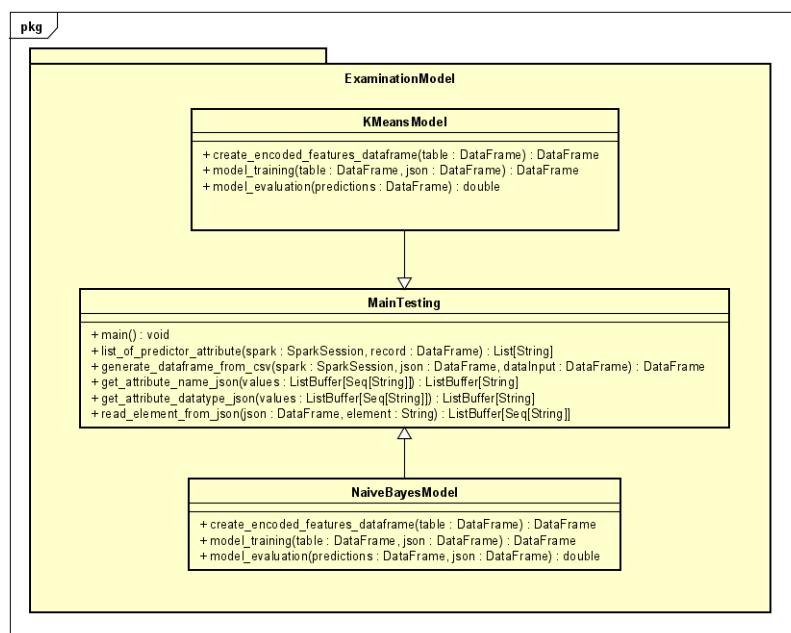
Kelas *MainTesting* merupakan kelas dengan tipe *object*. Hal ini dikarenakan kelas *MainTesting* berperan penting untuk melakukan eksekusi pengelompokan data dengan algoritma Greedy k-member clustering dan anonimisasi data dengan algoritma k-anonymity.

Berikut adalah penjelasan masing-masing method pada kelas *MainTesting*:

- **main** adalah fungsi yang bertujuan untuk melakukan eksekusi perangkat lunak anonimisasi dengan objek GreedyKMemberClustering dan objek KAnonymity.
- **generate\_dataframe\_from\_csv** dengan parameter `spark(SparkSession)`, `json(DataFrame)`, `dataInput(DataFrame)` adalah fungsi yang bertujuan untuk mengambil data `quasi_identifier` dan `sensitive_attribute` pada JSON. Output dari fungsi ini adalah array 2 dimensi yang berisi nama atribut dan tipe atribut (*category/numeric*).
- **get\_attribute\_name\_json** dengan parameter `values(ListBuffer[Seq[String]])` adalah fungsi yang bertujuan untuk mendapatkan nama-nama atribut berdasarkan output array 2 dimensi dari fungsi `generate_dataframe_from_csv`.
- **get\_attribute\_datatype\_json** dengan parameter `values(ListBuffer[Seq[String]])` adalah fungsi yang bertujuan untuk mengubah atribut dengan tipe "category" menjadi String, sedangkan atribut dengan tipe "numeric" menjadi Integer.
- **read\_element\_from\_json** dengan parameter `json(DataFrame)`, `elemen(String)` adalah fungsi yang bertujuan untuk mendapatkan nilai atribut tertentu pada JSON. Fungsi ini dipanggil pada fungsi lain yaitu `generate_dataframe_from_csv` untuk mengambil nilai dari atribut `quasi_identifier` dan `sensitive_identifier`.

#### 4.1.4 Diagram Kelas pada Package ExaminationModel

Perangkat lunak anonimisasi hanya memiliki satu jenis *package* dengan nama *model\_anonimisasi*. *Package model\_anonimisasi* terdiri dari beberapa kelas. Masing-masing kelas memiliki fungsi penting untuk menyelesaikan permasalahan pengelompokan dan anonimisasi data.



Gambar 4.6: Diagram Kelas pada ExaminationModel

### Kelas *KMeansModel*

Kelas *KMeansModel* merupakan kelas dengan tipe *class*. Hal ini dikarenakan kelas *KMeansModel* berfungsi sebagai model untuk membuat atribut dan method pada objek *KMeansModel*. Kelas *KMeansModel* bertujuan membuat pemodelan *k-means* dan menghitung *silhouette score*.

Berikut deskripsi method pada kelas *KMeansModel*:

- `create_encoded_features_dataframe` dengan parameter `table(DataFrame)` adalah fungsi yang bertujuan untuk membuat tabel data baru yang berisi nilai aktual, label index, vektor masing-masing atribut, dan vektor dari masing-masing label index.
- `model_training` dengan parameter `table(DataFrame), json(DataFrame)` adalah fungsi yang bertujuan untuk membuat model pelatihan k-means dan menghasilkan tabel hasil pengelompokan data berdasarkan input vektor fitur pada setiap baris data.
- `model_evaluation` dengan parameter `predictions(DataFrame)` adalah fungsi yang bertujuan untuk mencari tahu seberapa baik model k-means yang dibuat dengan menghitung silhouette score. Jika silhouette score mendekati nilai 1, maka hasil pengelompokan sudah baik.

Berikut implementasi method `model_training`:

---

#### Algorithm 13 Membuat Pemodelan K-Means

---

```

1: Function model_training(table, json)
2: Input: table data, JSON
3: Output: table of cluster data.
4:
5: k = json.select("k_means.k").first().getLong(0).toInt
6: kmeans = new KMeans().setK(k).setFeaturesCol("features").setPredictionCol("prediction")
7: model = kmeans.fit(table)
8: predictions = model.transform(table)
9: return predictions

```

---

- Baris 5: baris ini mendapatkan nilai `k` dari atribut `k_means` JSON.
- Baris 6: baris ini membuat model KMeans menggunakan parameter `k` pada baris sebelumnya.
- Baris 7: baris ini melakukan pelatihan model k-means dengan parameter tabel data.
- Baris 8: baris ini melakukan prediksi model k-means terhadap parameter tabel data.

Berikut implementasi method `model_evaluation`:

---

#### Algorithm 14 Menghitung Silhouette Score

---

```

1: Function calculate_numeric_distance(num1, num2, max, min)
2: Input: numeric 1, numeric 2, maximum number, maximum number.
3: Output: numeric distance.
4:
5: evaluator = new ClusteringEvaluator()
6: silhouette_score = evaluator.evaluate(predictions)
7: return silhouette_score

```

---

- Baris 5: baris ini membuat model `ClusteringEvaluator` untuk evaluasi k-means.
- Baris 6: baris ini menghitung nilai *silhouette score* untuk pemodelan k-means.

### Kelas *NaiveBayesModel*

Kelas *NaiveBayesModel* merupakan kelas dengan tipe *class*, karena kelas *NaiveBayesModel* hanya berfungsi sebagai model untuk membuat atribut dan fungsi pada objek *NaiveBayesModel*. Kelas ini bertujuan untuk melakukan pemodelan *naive bayes* dan menghitung *accuracy*.

Berikut deskripsi method pada kelas *NaiveBayesModel*:

- `create_encoded_features_dataframe` dengan parameter `table(DataFrame)` adalah fungsi yang bertujuan untuk membuat tabel data baru yang berisi nilai aktual, label index, vektor untuk masing-masing atribut, dan ditambah satu kolom baru untuk menyimpan vektor fitur dari label index. Output fungsi ini menjadi input pada fungsi `model_training`
- `model_training` dengan parameter `table(DataFrame), json(DataFrame)` adalah fungsi yang bertujuan untuk membuat model pelatihan naive bayes dan menghasilkan tabel klasifikasi data berdasarkan input vektor fitur untuk setiap baris data.
- `model_evaluation` dengan parameter `predictions(DataFrame), json(DataFrame)` adalah fungsi yang bertujuan untuk mencari tahu seberapa baik model naive bayes yang dibuat dengan menghitung accuracy score. Apabila nilainya mendekati 1, maka klasifikasi data yang dibuat dengan model naive bayes sudah mendekati benar.

Berikut implementasi method `model_training`:

---

#### Algorithm 15 Membuat Pemodelan Naive Bayes

---

```

1: Function model_training(table,json)
2: Input: table data, JSON
3: Output: table of classification data.
4:
5: attrName = json.select("naive_bayes.label").first().getString(0)
6:
7: trainingSet = json.select("naive_bayes.training_set").first().getDouble(0)
8: testSet = json.select("naive_bayes.test_set").first().getDouble(0)
9:
10: Array(training, test) = table.randomSplit(Array(trainingSet, testSet))
11:
12: model = new NaiveBayes().setModelType("multinomial")
13: model = model.setLabelCol(attrName+"_Index").fit(training)
14:
15: predictions = model.transform(test)
16: return predictions

```

---

- Baris 5: baris ini mendapatkan nilai `label` dari atribut `naive_bayes` JSON.
- Baris 7: baris ini mendapatkan persentase `training_set` dari atribut `naive_bayes` JSON.
- Baris 8: baris ini mendapatkan persentase `test_set` dari atribut `naive_bayes` JSON.
- Baris 10: baris ini membagi data menjadi data training dan data test berdasarkan persentase dari atribut `trainingSet` dan `testSet` pada baris sebelumnya.
- Baris 12: baris ini digunakan untuk membuat model NaiveBayes bertipe "multinomial".
- Baris 13: baris ini melakukan pelatihan model naive bayes dengan data training.
- Baris 15: baris ini melakukan prediksi model naive bayes dengan data test.

Berikut implementasi method `model_evaluation`:

---

**Algorithm 16** Menghitung Silhouette Score

---

```

1: Function calculate_numeric_distance(num1, num2, max, min)
2: Input: numeric 1, numeric 2, maximum number, maximum number.
3: Output: numeric distance.
4:
5: attrName = json.select("naive_bayes.label").first().getString(0)
6: evaluator = new MulticlassClassificationEvaluator()
7: evaluator = evaluator.setLabelCol(attrName + "_Index")
8: evaluator = evaluator.setPredictionCol("prediction")
9: evaluator = evaluator.setMetricName("accuracy")
10: accuracy = evaluator.evaluate(predictions)
11: return accuracy

```

---

- Baris 5: baris ini mendapatkan nilai `label` dari atribut `naive_bayes` JSON.
- Baris 6-9: baris ini membuat model evaluasi `MulticlassClassificationEvaluator` untuk mengetahui nilai akurasi dari hasil klasifikasi pemodelan naive bayes.
- Baris 10: baris ini mengembalikan hasil akurasi untuk pemodelan naive bayes.

### Kelas *MainTesting*

Kelas *MainTesting* merupakan kelas dengan tipe *object*. Hal ini dikarenakan kelas *MainTesting* berperan penting untuk melakukan eksekusi perangkat lunak pengujian terhadap masalah pengelempokan data dengan k-means dan masalah klasifikasi data dengan naive bayes. Kelas ini juga menangani eksekusi kedua model tersebut.

Berikut deskripsi method pada kelas *MainTesting*:

- `main` adalah fungsi yang bertujuan untuk melakukan proses eksekusi perangkat lunak pengujian dengan membuat objek `NaiveBayesModel` dan objek `KMeansModel`.
- `list_of_predictor_attribute` dengan parameter `spark(SparkSession)`, `record(DataFrame)` adalah fungsi yang bertujuan untuk mendapatkan nilai kolom-kolom prediktor dari tabel data.
- `generate_dataframe_from_csv` dengan parameter `spark(SparkSession)`, `json(DataFrame)`, `dataInput(DataFrame)` adalah fungsi yang bertujuan untuk mengambil data `quasi_identifier` dan `sensitive_attribute` pada JSON. Output dari fungsi ini adalah array 2 dimensi berisi informasi mengenai nama atribut dan kategori atribut (*category/numeric*).
- `get_attribute_name_json` dengan parameter `values(ListBuffer[Seq[String]])` adalah fungsi yang bertujuan untuk mendapatkan nama-nama atribut berdasarkan output array 2 dimensi dari fungsi `generate_dataframe_from_csv`.
- `get_attribute_datatype_json` dengan parameter `values(ListBuffer[Seq[String]])` adalah fungsi yang bertujuan untuk mengubah atribut dengan tipe "category" menjadi String, sedangkan atribut dengan tipe "numeric" menjadi Integer.
- `read_element_from_json` dengan parameter `json(DataFrame)`, `element(String)` adalah fungsi yang bertujuan untuk mendapatkan nilai atribut tertentu pada JSON. Fungsi ini dipanggil pada fungsi lain yaitu `generate_dataframe_from_csv` untuk mengambil nilai dari atribut `quasi_identifier` dan `sensitive_identifier`.

## 4.2 Masukan Perangkat Lunak

Perangkat lunak membutuhkan masukan berupa data input .csv yang berisi tabel privat beserta nama atributnya. Pada file .csv, baris pertama merupakan nama atribut dan baris berikutnya merupakan data. Setiap atribut dipisahkan dengan tanda koma (","), sedangkan data baru dipisahkan dengan baris. Format file .csv dapat dilihat pada Listing 5.9.

Listing 4.1: Dataset Adult

```
age,workclass,zip,education,year_of_education,marital_status,occupation
39,State-gov,77516,Bachelors,13,Never-married,Adm-clerical
50,Self-emp-not-inc,83311,Bachelors,13,Married-civ-spouse,Exec-managerial
38,Private,215646,HS-grad,9,Divorced,Handlers-cleaners
53,Private,234721,11th,7,Married-civ-spouse,Handlers-cleaners
```

### 4.2.1 Masukan Perangkat Lunak Eksplorasi

Perangkat lunak eksplorasi membutuhkan data masukan tambahan yaitu file .json yang berisi pohon klasifikasi serta tipe dan jenis atribut. File .json menerapkan format penyimpanan data dengan diawali oleh tanda ("{ }") dan diisi oleh nilai *key,value* yang dipisahkan oleh tanda titik dua (:). Format file .json dapat dilihat pada Listing 4.2.

Berikut spesifikasi input .json untuk perangkat lunak eksplorasi:

- **input\_path** adalah lokasi penyimpanan data input perangkat lunak. Atribut ini akan memberi tahu Spark untuk mengambil data input pada lokasi ini.
- **output\_path** adalah lokasi penyimpanan hasil output perangkat lunak. Atribut ini akan memberi tahu Spark untuk menyimpan hasil output pada lokasi ini.
- **selected\_column** adalah array yang menyimpan informasi pemilihan atribut. Atribut ini menyimpan 2 jenis informasi penting dalam pemilihan atribut, yaitu
  - **attrName** adalah nama atribut yang ingin dipilih pada tabel data. Atribut ini akan memberi tahu Spark untuk mengambil atribut berdasarkan nama atribut.
  - **dataType** adalah jenis tipe data atribut yang telah dipilih. Atribut ini akan memberi tahu Spark untuk menyimpan atribut yang dipilih dalam format tertentu.

Listing 4.2: Input JSON untuk Eksplorasi Data

```
{
  "input_path": "<path data input>",
  "output_path": "<path data output>",
  "selected_column": [
    {
      "attrName": "<nama atribut>",
      "dataType": "<tipe data atribut>"
    },
    {
      "attrName": "<nama atribut>",
      "dataType": "<tipe data atribut>"
    }
  ]
}
```

#### 4.2.2 Masukan Perangkat Lunak Anonimisasi

Perangkat lunak anonimisasi membutuhkan data masukan tambahan yaitu file .json yang berisi pohon klasifikasi serta tipe dan jenis atribut. File .json menerapkan format penyimpanan data dengan diawali oleh tanda ("{ }") dan diisi oleh nilai *key,value* yang dipisahkan oleh tanda titik dua (":"). Format file .json dapat dilihat pada Listing 4.3.

Berikut spesifikasi input .json untuk perangkat lunak anonimisasi:

- **k** adalah nilai konstanta k-anonymity dan greedy k-member clustering. Atribut ini memiliki tipe data Integer sehingga nilainya harus bilangan bulat.
- **num\_sample\_datas** adalah jumlah sample data. Atribut ini memiliki tipe data Integer sehingga nilainya harus dinyatakan dalam bilangan bulat.
- **input\_path** adalah lokasi penyimpanan data input perangkat lunak. Atribut ini akan memberi tahu Spark untuk mengambil data input pada lokasi ini.
- **output\_path** adalah lokasi penyimpanan hasil output perangkat lunak. Atribut ini akan memberi tahu Spark untuk menyimpan hasil output pada lokasi ini.
- **identifiers** adalah array yang berisi nama-nama atribut yang dapat mengungkapkan identitas sebuah data. Atribut ini akan dihilangkan pada tabel anonimisasi.
- **sensitive\_identifiers** adalah array yang berisi nama-nama atribut yang nilainya bersifat sensitif. Atribut ini akan tetap ada pada tabel dianonimisasi.
- **quasi\_identifiers** adalah atribut yang nilainya dapat dipakai untuk mengungkap entitas data. Atribut ini berisi pasangan nilai berupa nama atribut dan jenis atribut. Jenis atribut diisi dengan nilai "category" untuk menyatakan atribut kategori, sedangkan untuk menyatakan atribut numerik jenis atribut akan diisi dengan nilai "numeric".
- **domain\_generalization\_hierarchy** adalah array yang menyimpan informasi mengenai pembentukan pohon DGH. Atribut ini menyimpan 5 jenis informasi penting dalam pembentukan pohon DGH, yaitu:
  - **attrName** adalah nama atribut yang ingin dipilih pada tabel data. Atribut ini akan memberi tahu Spark untuk mengambil atribut berdasarkan nama atribut.
  - **value** adalah nilai-nilai yang mungkin muncul untuk atribut yang terpilih. Atribut ini akan digunakan untuk memberi nama pada sebuah node di pohon DGH.
  - **parent** adalah nilai yang menyatakan nama dari sebuah node root. Atribut ini akan digunakan untuk menyatakan parent dari sebuah node di pohon DGH.
  - **level** adalah nilai yang menyatakan ketinggian node di pohon DGH. Atribut ini digunakan untuk menempatkan node di ketinggian tertentu di pohon DGH.
  - **position** adalah nilai yang menyatakan posisi node di pohon DGH. Atribut ini digunakan untuk menempatkan node di posisi (kiri/kanan) dari node parent.

Berikut adalah hal penting yang perlu diperhatikan terkait input (.json):

- Nilai atribut yang dicantumkan pada **domain\_generalization\_hierarchy** di salah satu atribut harus mencakup seluruh kemungkinan nilai untuk atribut tersebut. Apabila tidak terpenuhi, maka beberapa nilai tidak dapat dianonimisasi.
- Semakin banyak atribut yang dicantumkan pada **quasi\_identifier**, maka waktu komputasi akan semakin lama untuk dijalankan pada komputer lokal.

Listing 4.3: Input JSON untuk Anonimisasi Data

```
{  
    "k": <konstanta untuk k-anonymity dan greedy k-member clustering>,  
    "num_sample_datas": <jumlah sampel data>,  
    "input_path": "<path data input>",  
    "output_path": "<path data output>",  
    "identifier": [  
        {  
            "attrName": "<nama atribut>",  
            "dataType": "<tipe data atribut>"  
        }  
    ],  
    "sensitive_identifier": [  
        {  
            "attrName": "<nama atribut>",  
            "dataType": "<jenis atribut>"  
        }  
    ],  
    "quasi_identifier": [  
        {  
            "attrName": "<nama atribut>",  
            "dataType": "<jenis atribut>"  
        },  
        {  
            "attrName": "<nama atribut>",  
            "dataType": "<jenis atribut>"  
        }  
    ],  
    "domain_generalization_hierarchy": {  
        "<nama atribut>": [  
            {  
                "value": "<nilai atribut>",  
                "parent": "<nilai atribut parent>",  
                "level": "1",  
                "position": "null"  
            },  
            {  
                "value": "<nilai atribut>",  
                "parent": "<nilai atribut parent>",  
                "level": "2",  
                "position": "left"  
            },  
            {  
                "value": "<nilai atribut>",  
                "parent": "<nilai atribut parent>",  
                "level": "2",  
                "position": "right"  
            }  
        ]  
    }  
}
```

#### 4.2.3 Masukan Perangkat Lunak Pengujian

Perangkat lunak anonimisasi membutuhkan data masukan tambahan yaitu file .json yang berisi pohon klasifikasi serta tipe dan jenis atribut. File .json menerapkan format penyimpanan data dengan diawali oleh tanda ("{ }") dan diisi oleh nilai *key,value* yang dipisahkan oleh tanda titik dua (":"). Format file .json dapat dilihat pada Listing ??.

Berikut spesifikasi input .json untuk perangkat lunak pengujian:

- **input\_path** adalah lokasi penyimpanan data input perangkat lunak. Atribut ini akan memberi tahu Spark untuk mengambil data input pada lokasi ini.
- **output\_path** adalah lokasi penyimpanan hasil output perangkat lunak. Atribut ini akan memberi tahu Spark untuk menyimpan hasil output pada lokasi ini.
- **model\_name** adalah jenis model yang akan dipakai untuk pengujian. Atribut ini terdiri dari 2 jenis nilai yaitu **k\_means**,**naive\_bayes**.
- **selected\_column** adalah array yang menyimpan informasi pemilihan atribut. Atribut ini menyimpan 2 jenis informasi penting dalam pemilihan atribut, yaitu:
  - **attrName** adalah nama atribut yang ingin dipilih pada tabel data. Atribut ini akan memberi tahu Spark untuk mengambil kolom berdasarkan nama atribut.
  - **dataType** adalah jenis tipe data atribut yang telah dipilih. Atribut ini akan memberi tahu Spark untuk menyimpan atribut yang dipilih dalam format tertentu.
- **k\_means** adalah parameter untuk pemodelan k-means. Atribut ini terdiri dari 1 parameter untuk pemodelan k-means, yaitu:
  - **k** adalah konstanta k pada pemodelan k-means. Atribut ini digunakan untuk menentukan jumlah kelompok data yang ingin dibentuk.
- **naive\_bayes** adalah parameter untuk pemodelan naive bayes. Atribut ini terdiri dari 3 parameter untuk pemodelan naive bayes, yaitu:
  - **label** adalah nama atribut yang akan dianggap sebagai label untuk pemodelan naive bayes. Atribut ini digunakan untuk mengklasifikasikan data berdasarkan nilai pada atribut label.
  - **training\_set** adalah persentase pembagian data training. Umumnya, persentase yang dipilih untuk pembagian data training adalah 0.7.
  - **test\_set** adalah persentase pembagian data test. Umumnya, persentase yang dipilih untuk pembagian data test adalah 0.3.

Berikut adalah hal penting yang perlu diperhatikan terkait input (.json):

- Atribut **input\_path** untuk dapat diisi menggunakan lokasi data setelah anonimisasi maupun data sebelum anonimisasi untuk membandingkan hasil keduanya.
- Atribut **model\_name** hanya boleh diisi oleh satu jenis model, tidak boleh lebih. Jika diisi lebih dari satu nilai, maka program akan mengeluarkan pesan error.
- Atribut **training\_set** dan **test\_set** harus berada pada rentang 0 sampai dengan 1. Lalu ketika dijumlahkan **training\_set** dan **test\_set** harus benilai 1.

Listing 4.4: Input JSON untuk Pengujian Data

```
{  
    "input_path": "<path data input>",  
    "output_path": "<path data output>",  
    "model_name": "<model pengujian (k_means/naive_bayes)>",  
    "selected_column": [  
        {  
            "attrName": "<nama atribut>",  
            "dataType": "<jenis atribut>"  
        },  
        {  
            "attrName": "<nama atribut>",  
            "dataType": "<jenis atribut>"  
        }  
    ],  
    "k_means": {  
        "k": <nilai k untuk k_means>  
    },  
    "naive_bayes": {  
        "label": "<nama atribut>",  
        "training_set": <persentase training_set (0.7)>,  
        "test_set": <persentase training_set (0.3)>  
    }  
}
```



## BAB 5

# IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dijelaskan lebih detil mengenai hasil implementasi perangkat lunak eksplorasi, perangkat lunak anonimisasi, dan perangkat lunak pengujian.

### 5.1 Implementasi Antarmuka

Implementasi antarmuka pada lingkungan big data terbagi menjadi dua jenis, yaitu menggunakan terminal command line jika eksperimen ingin dieksekusi pada lab Hadoop cluster dan menggunakan IntelliJ jika eksperimen ingin dieksekusi pada komputer lokal (Standalone). Penjelasan implementasi antarmuka akan dijelaskan lebih detil pada bagian selanjutnya.

#### 5.1.1 Komputer Lokal dengan IntelliJ

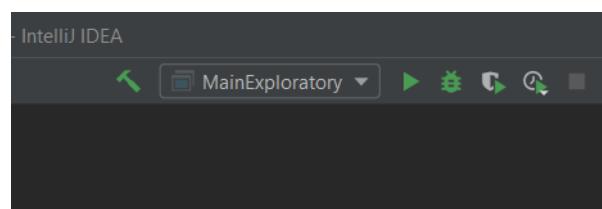
Pengujian ini dilakukan menggunakan komputer dengan spesifikasi sebagai berikut Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz (8 CPUs) dan 8 GB RAM. Hasil pengujian ini dapat menghasilkan yang valid jika dijalankan pada komputer lokal dengan hadoop cluster. Perbedaanya terletak pada jumlah komputer yang melakukan komputasi. Apabila menggunakan komputer lokal, waktu komputasinya menjadi sangat lambat jika menggunakan ukuran data yang besar. Oleh karena itu ukuran data perlu dibatasi pada komputer lokal.



Gambar 5.1: IntelliJ

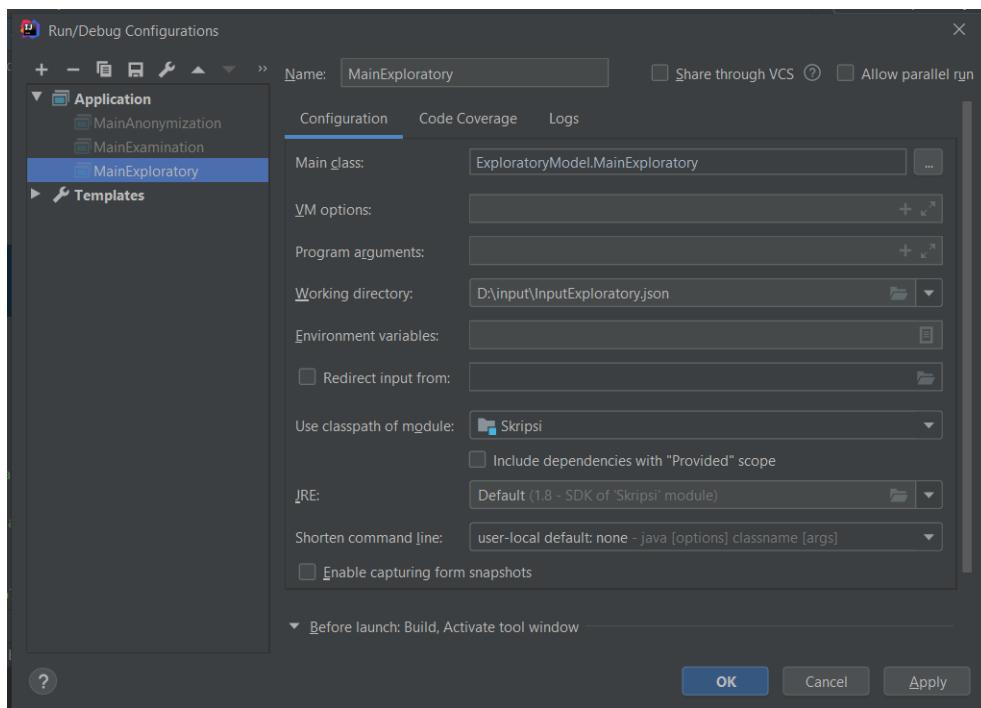
#### Perangkat Lunak Eksplorasi

Langkah pertama sebelum menjalankan perangkat lunak eksplorasi pada IntelliJ adalah mengisi parameter input. Caranya dengan menekan tombol selector Main class pada sisi layar kanan atas, contohnya pada 5.16 tombol selector Main class yang ditekan bernama MainExploratory.



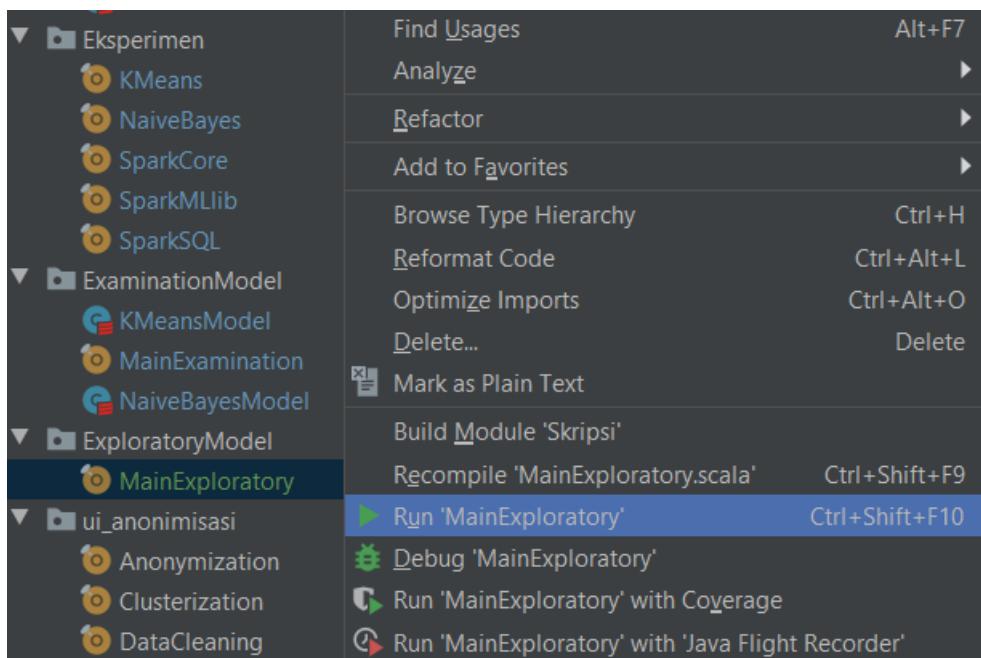
Gambar 5.2: Main Class Perangkat Lunak Ekplorasi

Setelah itu pilih menu **Edit Configurations...**, lalu isi parameter **Working directory** dengan lokasi **InputAnonymization.json**, contohnya pada Gambar 5.40 **Working directory** diisi dengan nilai **D:\input\InputAnonymization.json**. Pada tahap ini langkah pertama telah selesai.



Gambar 5.3: Konfigurasi Parameter Perangkat Lunak Ekplorasi

Langkah kedua adalah menjalankan perangkat lunak ekplorasi pada IntelIJ dengan memilih Main class pada sisi layar kiri layar, contohnya **MainExploratory**. Kemudian klik kanan pada Main class tersebut dan pilih menu Run '**MainExploratory**'. Perangkat lunak akan menghasilkan output berupa tabel unik berdasarkan pemilihan **selected\_column** pada **InputExploratory.json**



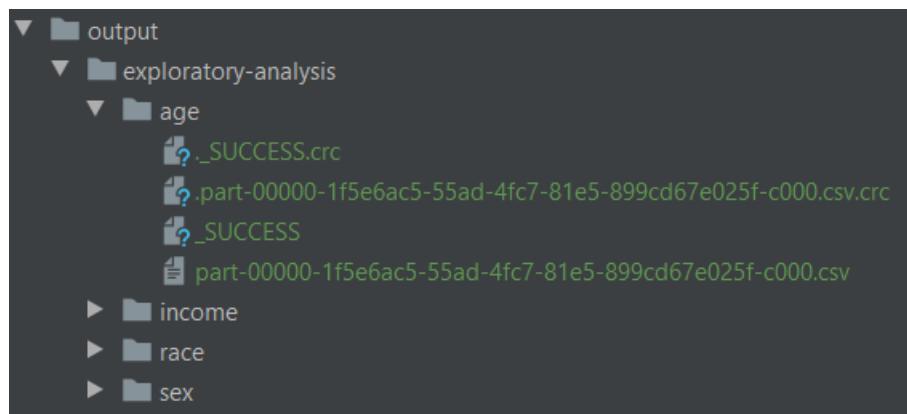
Gambar 5.4: Menjalankan Perangkat Lunak Ekplorasi

Langkah terakhir adalah menunggu proses komputasi sampai dengan selesai. Sebagai catatan semakin banyak data yang diolah, maka waktu komputasi akan semakin lama. Perangkat lunak eksplorasi yang telah berhasil menyelesaikan proses komputasinya ditandai dengan baris log seperti berikut 'Process finished with exit code 0. Karena proses komputasi sudah selesai, output perangkat lunak eksplorasi sudah muncul berdasarkan `output_path` pada `InputExploratory.json`

```
20/10/25 12:45:54 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-EP6CL8L:4040
20/10/25 12:45:54 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
20/10/25 12:45:54 INFO MemoryStore: MemoryStore cleared
20/10/25 12:45:54 INFO BlockManager: BlockManager stopped
20/10/25 12:45:54 INFO BlockManagerMaster: BlockManagerMaster stopped
20/10/25 12:45:54 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
20/10/25 12:45:54 INFO SparkContext: Successfully stopped SparkContext
20/10/25 12:45:54 INFO ShutdownHookManager: Shutdown hook called
20/10/25 12:45:54 INFO ShutdownHookManager: Deleting directory C:\Users\asus\AppData\Local\Temp\spark-1533322222\part-00000
Process finished with exit code 0
```

Gambar 5.5: Log Perangkat Lunak Ekplorasi

Output yang dihasilkan dari perangkat lunak eksplorasi dapat dilihat pada lokasi yang telah dicantumkan sebelumnya pada nilai `output_path` JSON. Sehingga ketika lokasi `output_path` dibuka, maka akan tampak seperti Gamabar 1.1. Hasil eksplorasi nilai unik akan digunakan sebagai referensi mengisi nilai `domain_generalization_hierarchy` sebuah atribut tabel data pada `InputAnonymization.json`. Eksplorasi perlu dilakukan mengingat seluruh nilai atribut harus dapat diubah menjadi nilai anonimisasi. Output dapat dilihat ketika membuka file `part-000X-YYYYY.csv`.



Gambar 5.6: Folder Output Perangkat Lunak Ekplorasi

Contoh output yang dihasilkan adalah tabel nilai unik dari atribut `Age`. Karena output disimpan dalam format CSV, baris pertama menyatakan nama kolom, sedangkan baris selanjutnya menyatakan nilai unik pada kolom tersebut. Diketahui bahwa nilai unik atribut `Age` memiliki 5 jenis nilai unik.

1	<code>race</code>
2	<code>Other</code>
3	<code>Amer-Indian-Eskimo</code>
4	<code>White</code>
5	<code>Asian-Pac-Islander</code>
6	<code>Black</code>

Gambar 5.7: Tabel Nilai Unik (Age)

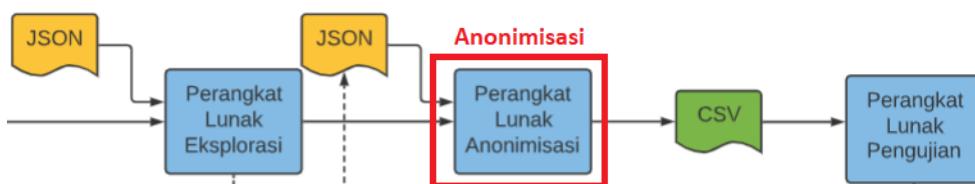
Listing 4 adalah contoh input JSON dengan nama `InputExploratory.json`:

Listing 5.1: Data JSON untuk Perangkat Lunak Eksplorasi

```
{
  "input_path": "D:/input/adult100k.csv",
  "output_path": "D:/output/exploratory-analysis/",
  "selected_column": [
    {
      "attrName": "age",
      "dataType": "numeric"
    },
    {
      "attrName": "race",
      "dataType": "category"
    },
    {
      "attrName": "sex",
      "dataType": "category"
    },
    {
      "attrName": "income",
      "dataType": "category"
    }
  ]
}
```

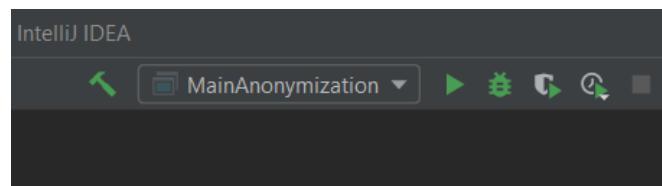
### Perangkat Lunak Anonimisasi

Setelah mengeksekusi perangkat lunak eksplorasi untuk mencari tahu nilai unik pada setiap kolom quasi-identifier, pengujian akan dilanjutkan pada perangkat lunak anonimisasi untuk mencari tahu mengenai hasil pengelompokan data dengan algoritma Greedy k-member clustering.



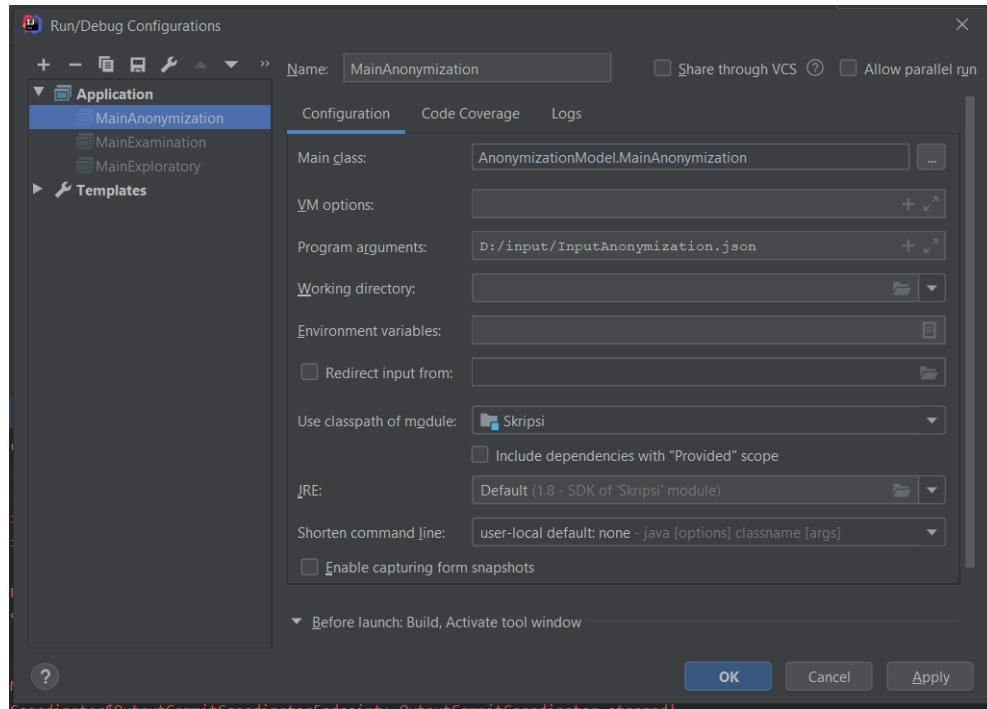
Gambar 5.8: Main Class Perangkat Lunak Eksplorasi

Langkah pertama sebelum menjalankan perangkat lunak eksplorasi pada IntelliJ adalah mengisi parameter input. Caranya dengan menekan tombol selector Main class pada sisi layar kanan atas, contohnya pada [5.16](#) tombol selector Main class yang ditekan bernama `MainExploratory`.



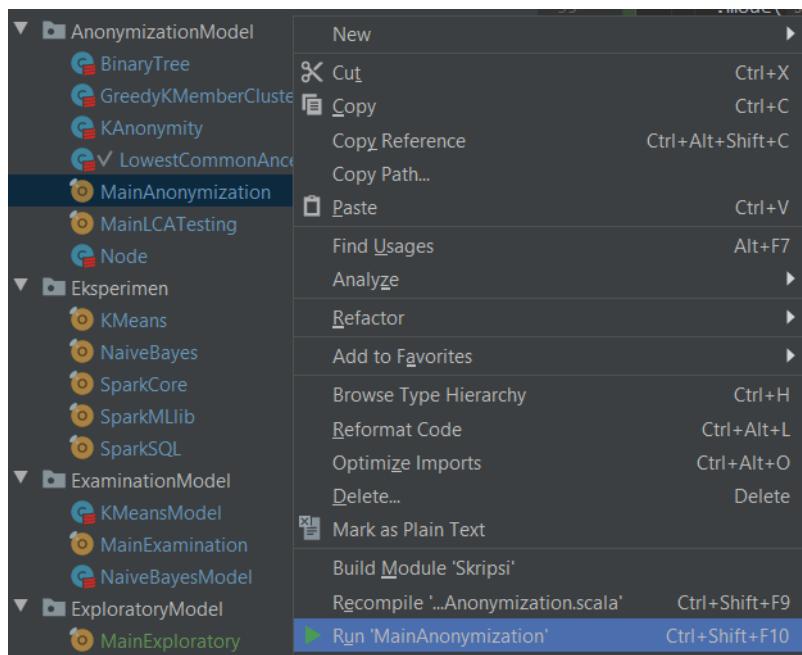
Gambar 5.9: Main Class Perangkat Lunak Eksplorasi

Setelah itu pilih menu **Edit Configurations...**, lalu isi parameter **Working directory** dengan lokasi **InputAnonymization.json**, contohnya pada Gambar 5.40 **Working directory** diisi dengan nilai **D:\input\InputAnonymization.json**. Pada tahap ini langkah pertama telah selesai.



Gambar 5.10: Konfigurasi Parameter Perangkat Lunak Ekplorasi

Langkah kedua adalah menjalankan perangkat lunak ekplorasi pada IntelliJ dengan memilih Main class pada sisi kiri layar, contohnya **MainExploratory**. Kemudian klik kanan pada Main class tersebut dan pilih menu **'MainExploratory'**. Perangkat lunak akan menghasilkan output berupa tabel unik berdasarkan pemilihan **selected\_column** pada **InputExploratory.json**



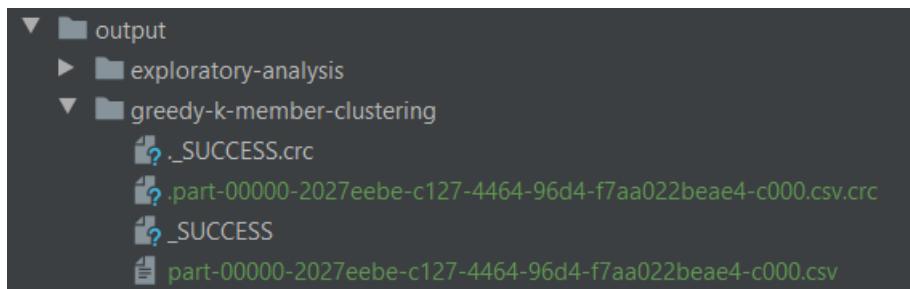
Gambar 5.11: Menjalankan Perangkat Lunak Ekplorasi

Langkah terakhir adalah menunggu proses komputasi sampai dengan selesai. Sebagai catatan semakin banyak data yang diolah, maka waktu komputasi akan semakin lama. Perangkat lunak eksplorasi yang telah berhasil menyelesaikan proses komputasinya ditandai dengan baris log seperti berikut 'Process finished with exit code 0. Karena proses komputasi sudah selesai, output perangkat lunak eksplorasi sudah muncul berdasarkan `output_path` pada `InputExploratory.json`

```
20/10/25 12:45:54 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-EP6CL8L:4040
20/10/25 12:45:54 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
20/10/25 12:45:54 INFO MemoryStore: MemoryStore cleared
20/10/25 12:45:54 INFO BlockManager: BlockManager stopped
20/10/25 12:45:54 INFO BlockManagerMaster: BlockManagerMaster stopped
20/10/25 12:45:54 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
20/10/25 12:45:54 INFO SparkContext: Successfully stopped SparkContext
20/10/25 12:45:54 INFO ShutdownHookManager: Shutdown hook called
20/10/25 12:45:54 INFO ShutdownHookManager: Deleting directory C:\Users\asus\AppData\Local\Temp\spark-1a2c3d\part-00000
Process finished with exit code 0
```

Gambar 5.12: Log Perangkat Lunak Ekplorasi

Output yang dihasilkan dari perangkat lunak eksplorasi dapat dilihat pada lokasi yang telah dicantumkan sebelumnya pada nilai `output_path` JSON. Sehingga ketika lokasi `output_path` dibuka, maka akan tampak seperti Gamabar 1.1. Hasil eksplorasi nilai unik akan digunakan sebagai referensi mengisi nilai `domain_generalization_hierarchy` sebuah atribut tabel data pada `InputAnonymization.json`. Eksplorasi perlu dilakukan mengingat seluruh nilai atribut harus dapat diubah menjadi nilai anonimisasi. Output dapat dilihat ketika membuka file `part-000X-YYYYYY.csv`.



Gambar 5.13: Folder Output Perangkat Lunak Anonimisasi

Contoh output yang dihasilkan adalah tabel pengelompokan data berdasarkan atribut quasi-identifier `age`, `race`, `sex`, `workclass`, `income`. Karena output disimpan dalam format CSV, baris pertama menyatakan nama kolom, sedangkan baris selanjutnya menyatakan nilai unik pada kolom tersebut. Diketahui bahwa masing-masing data sudah dikelompokan ke masing-masing cluster. Sampel membentuk 2 kelompok data, karena dipilih parameter `k=2` pada `InputAnonymization.json`.

<code>id</code> , <code>age</code> , <code>race</code> , <code>sex</code> , <code>workclass</code> , <code>income</code> , <code>Cluster</code>
2,50,White,Male,Self-emp-not-inc,<=50K,Cluster 1
4,53,Black,Male,Private,<=50K,Cluster 1
3,38,White,Male,Private,<=50K,Cluster 2
1,39,White,Male,State-gov,<=50K,Cluster 2
5,28,Black,Female,Private,<=50K,Cluster 2

Gambar 5.14: Tabel Pengelompokan Data

Listing 5.2 adalah contoh input JSON dengan nama InputAnonymization.json:

Listing 5.2: Data JSON untuk Perangkat Lunak Eksplorasi

```
{  
    "k": 2,  
    "num_sample_datas": 5,  
    "input_path": "D:/input/adult100k.csv",  
    "output_path": "D:/output/",  
    "identifier": [  
        {  
            "attrName": "name",  
            "dataType": "category"  
        }  
    ],  
    "sensitive_identifier": [  
        {  
            "attrName": "income",  
            "dataType": "category"  
        }  
    ],  
    "quasi_identifier": [  
        {  
            "attrName": "age",  
            "dataType": "numeric"  
        },  
        {  
            "attrName": "race",  
            "dataType": "category"  
        },  
        {  
            "attrName": "sex",  
            "dataType": "category"  
        },  
        {  
            "attrName": "workclass",  
            "dataType": "category"  
        }  
    ],  
    "domain_generalization_hierarchy": {  
        "race": [  
            {  
                "value": "Person",  
                "parent": "Person",  
                "level": "1",  
                "position": "null"  
            },  
            {  
                "value": "Oriental",  
                "parent": "Person",  
                "level": "2",  
                "position": "left"  
            },  
        ]  
    }  
}
```

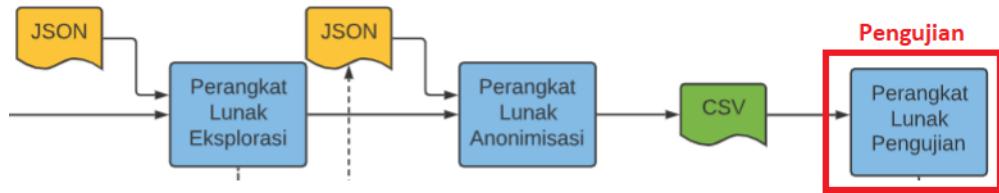
```
{  
    "value": "General",  
    "parent": "Person",  
    "level": "2",  
    "position": "right"  
},  
{  
    "value": "Amer-Indian-Eskimo",  
    "parent": "Oriental",  
    "level": "3",  
    "position": "left"  
},  
{  
    "value": "Asian",  
    "parent": "Oriental",  
    "level": "3",  
    "position": "right"  
},  
{  
    "value": "White",  
    "parent": "General",  
    "level": "3",  
    "position": "left"  
},  
{  
    "value": "Black",  
    "parent": "General",  
    "level": "3",  
    "position": "right"  
}  
],  
"sex": [  
    {  
        "value": "Adult",  
        "parent": "Adult",  
        "level": "1",  
        "position": "null"  
    },  
    {  
        "value": "Female",  
        "parent": "Adult",  
        "level": "2",  
        "position": "left"  
    },  
    {  
        "value": "Male",  
        "parent": "Adult",  
        "level": "2",  
        "position": "right"  
    }  
],
```

```
"workclass": [
  {
    "value": "Employee",
    "parent": "Employee",
    "level": "1",
    "position": "null"
  },
  {
    "value": "Gov-employee",
    "parent": "Employee",
    "level": "2",
    "position": "left"
  },
  {
    "value": "Local-employee",
    "parent": "Employee",
    "level": "2",
    "position": "right"
  },
  {
    "value": "Gov",
    "parent": "Gov-employee",
    "level": "3",
    "position": "left"
  },
  {
    "value": "Other-gov",
    "parent": "Gov-employee",
    "level": "3",
    "position": "right"
  },
  {
    "value": "Self-emp",
    "parent": "Local-employee",
    "level": "3",
    "position": "left"
  },
  {
    "value": "Others",
    "parent": "Local-employee",
    "level": "3",
    "position": "right"
  },
  {
    "value": "State-gov",
    "parent": "Government",
    "level": "4",
    "position": "left"
  },
  {
    "value": "Federal-gov",
```

```
        "parent": "Government",
        "level": "4",
        "position": "right"
    },
    {
        "value": "Self-emp-not-inc",
        "parent": "Self-emp",
        "level": "4",
        "position": "left"
    },
    {
        "value": "Self-emp-inc",
        "parent": "Self-emp",
        "level": "4",
        "position": "right"
    },
    {
        "value": "Private",
        "parent": "Others",
        "level": "4",
        "position": "left"
    },
    {
        "value": "Never-worked",
        "parent": "Others",
        "level": "4",
        "position": "right"
    },
    {
        "value": "Local-gov",
        "parent": "Others-gov",
        "level": "4",
        "position": "left"
    },
    {
        "value": "Without-pay",
        "parent": "Others-gov",
        "level": "4",
        "position": "right"
    }
]
}
```

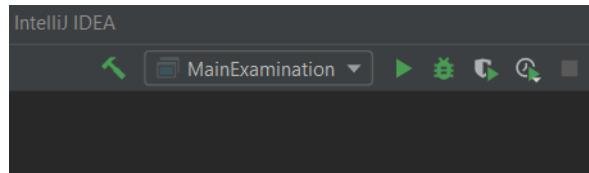
### Perangkat Lunak Pengujian

Setelah mengeksekusi perangkat lunak eksplorasi untuk mencari tahu nilai unik pada setiap kolom quasi-identifier, pengujian akan dilanjutkan pada perangkat lunak anonimisasi untuk mencari tahu mengenai hasil pengelompokan data dengan algoritma Greedy k-member clustering.



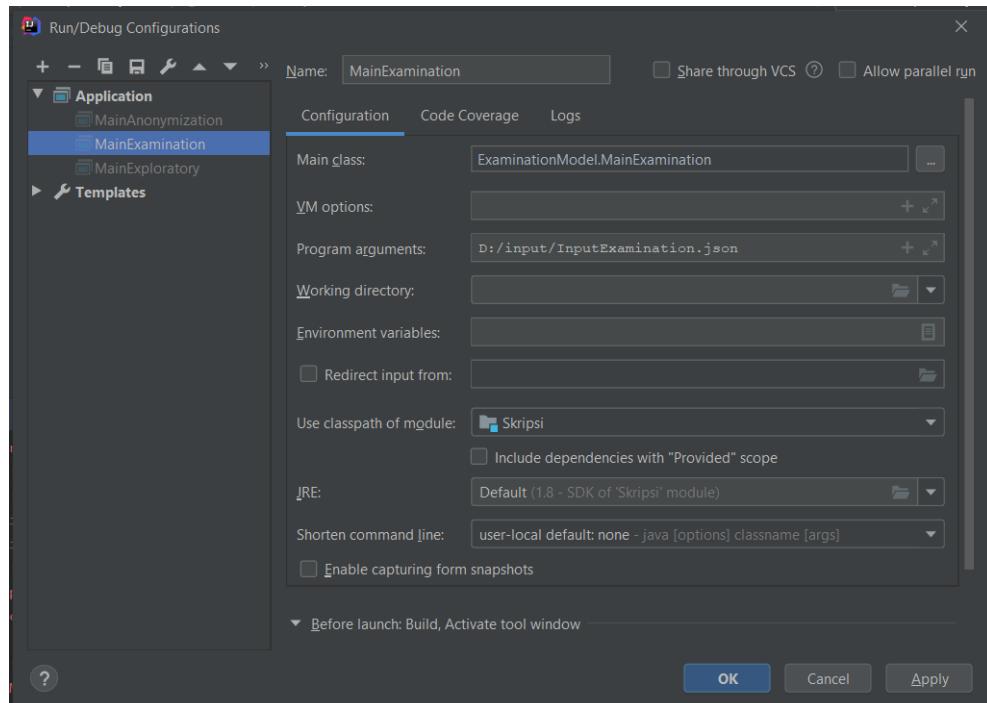
Gambar 5.15: Main Class Perangkat Lunak Eksplorasi

Langkah pertama sebelum menjalankan perangkat lunak eksplorasi pada IntelliJ adalah mengisi parameter input. Caranya dengan menekan tombol selector Main class pada sisi layar kanan atas, contohnya pada [5.16](#) tombol selector Main class yang ditekan bernama MainExploratory.



Gambar 5.16: Main Class Perangkat Lunak Eksplorasi

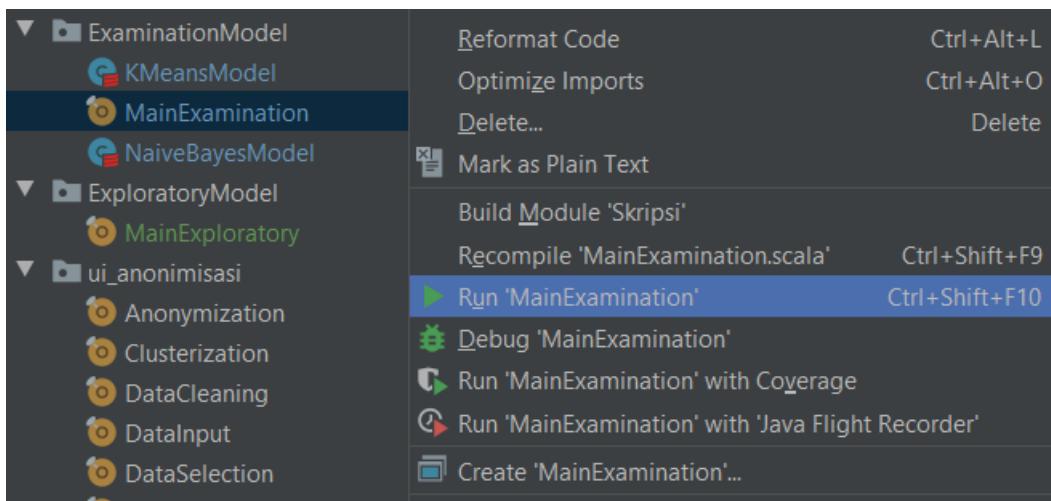
Setelah itu pilih menu **Edit Configurations...**, lalu isi parameter **Working directory** dengan lokasi `InputAnonymization.json`, contohnya pada Gambar [5.40](#) Working directory diisi dengan nilai `D:\input\InputAnonymization.json`. Pada tahap ini langkah pertama telah selesai.



Gambar 5.17: Konfigurasi Parameter Perangkat Lunak Eksplorasi

Langkah kedua adalah menjalankan perangkat lunak eksplorasi pada IntelliJ dengan memilih

Main class pada sisi layar kiri layar, contohnya MainExploratory. Kemudian klik kanan pada Main class tersebut dan pilih menu Run 'MainExploratory'. Perangkat lunak akan menghasilkan output berupa tabel unik berdasarkan pemilihan `selected_column` pada `InputExploratory.json`



Gambar 5.18: Menjalankan Perangkat Lunak Ekplorasi

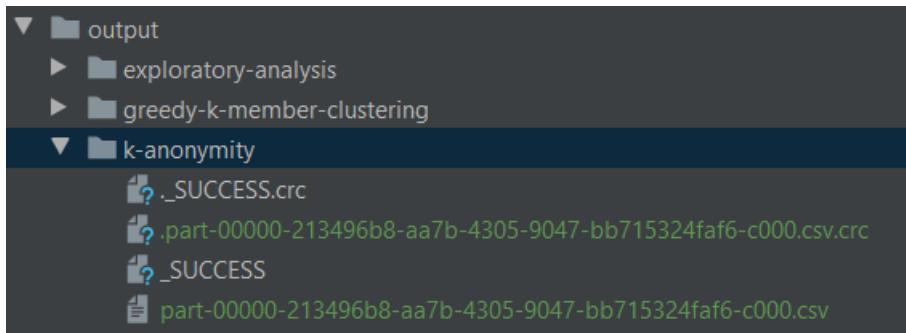
Langkah terakhir adalah menunggu proses komputasi sampai dengan selesai. Sebagai catatan semakin banyak data yang diolah, maka waktu komputasi akan semakin lama. Perangkat lunak eksplorasi yang telah berhasil menyelesaikan proses komputasinya ditandai dengan baris log seperti berikut 'Process finished with exit code 0'. Karena proses komputasi sudah selesai, output perangkat lunak eksplorasi sudah muncul berdasarkan `output_path` pada `InputExploratory.json`

```
20/10/25 12:45:54 INFO SparkUI: Stopped Spark web UI at http://DESKTOP-EP6CL8L:4040
20/10/25 12:45:54 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
20/10/25 12:45:54 INFO MemoryStore: MemoryStore cleared
20/10/25 12:45:54 INFO BlockManager: BlockManager stopped
20/10/25 12:45:54 INFO BlockManagerMaster: BlockManagerMaster stopped
20/10/25 12:45:54 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
20/10/25 12:45:54 INFO SparkContext: Successfully stopped SparkContext
20/10/25 12:45:54 INFO ShutdownHookManager: Shutdown hook called
20/10/25 12:45:54 INFO ShutdownHookManager: Deleting directory C:\Users\asus\AppData\Local\Temp\spark-1533322210\part-00000
```

Process finished with exit code 0

Gambar 5.19: Log Perangkat Lunak Ekplorasi

Output yang dihasilkan dari perangkat lunak eksplorasi dapat dilihat pada lokasi yang telah dicantumkan sebelumnya pada nilai `output_path` JSON. Sehingga ketika lokasi `output_path` dibuka, maka akan tampak seperti Gambar 1.1. Hasil eksplorasi nilai unik akan digunakan sebagai referensi mengisi nilai `domain_generalization_hierarchy` sebuah atribut tabel data pada `InputAnonymization.json`. Eksplorasi perlu dilakukan mengingat seluruh nilai atribut harus dapat diubah menjadi nilai anonimisasi. Output dapat dilihat ketika membuka file `part-000X-YYYYYY.csv`.



Gambar 5.20: Folder Output Perangkat Lunak Eksplorasi

Contoh output yang dihasilkan adalah tabel anonimisasi data berdasarkan atribut quasi-identifier `age`, `race`, `sex`, `workclass`, `income`. Karena output disimpan dalam format CSV, baris pertama menyatakan nama kolom, sedangkan baris selanjutnya menyatakan nilai unik pada kolom tersebut. Diketahui bahwa masing-masing data sudah dianonimisasi berdasarkan Domain Generalization Hierarchy. Sampel sudah membentuk equivalence class, dimana sebuah data menjadi sulit dibedakan dengan data lainnya. Hal ini membuktikan prinsip k-anonymity telah tercapai pada penelitian ini.

<code>id,age,race,sex,workclass,income</code>
1,[28-39],Person,Adult,Employee,<=50K
2,[50-53],Person,Male,Employee,<=50K
3,[28-39],Person,Adult,Employee,<=50K
4,[50-53],Person,Male,Employee,<=50K
5,[28-39],Person,Adult,Employee,<=50K

Gambar 5.21: Tabel Nilai Unik (Age)

Listing 4 adalah contoh input JSON dengan nama `InputExamination.json`:

Listing 5.3: Data JSON untuk Perangkat Lunak Eksplorasi

```
{
  "input_path": "D:/input/adult100k.csv",
  "output_path": "D:/output/",
  "selected_column": [
    {
      "attrName": "age",
      "dataType": "numeric"
    },
    {
      "attrName": "race",
      "dataType": "category"
    },
    {
      "attrName": "sex",
      "dataType": "category"
    },
    {
      "attrName": "income",
      "dataType": "category"
    }
  ]
}
```

```

}
{
  "input_path": "D:/input/adult100k.csv",
  "output_path": "D:/output/",
  "model_name": "k_means",
  "selected_column": [
    {
      "attrName": "age",
      "dataType": "numeric"
    },
    {
      "attrName": "race",
      "dataType": "category"
    },
    {
      "attrName": "sex",
      "dataType": "category"
    },
    {
      "attrName": "income",
      "dataType": "category"
    }
  ],
  "k_means": {
    "k": 2
  },
  "naive_bayes": {
    "label": "income",
    "training_set": 0.7,
    "test_set": 0.3
  }
}

```

### 5.1.2 Hadoop Cluster dengan Terminal Ubuntu

Pengujian ini dilakukan menggunakan Hadoop cluster dengan 10 slaves node yang dapat digunakan untuk proses komputasi. Hasil pengujian ini memiliki hasil output yang sama jika dijalankan pada komputer lokal. Perbedaanya terletak pada jumlah komputer yang melakukan komputasi. Apabila menggunakan Hadoop cluster, waktu komputasi untuk memproses ukuran data yang besar dapat dikurangi, karena komputasi dilakukan secara paralel terhadap 10 slaves node. Perintah untuk melakukan eksekusi Spark dapat ditulis menggunakan terminal Ubuntu.



Gambar 5.22: Terminal Ubuntu

Berikut adalah spesifikasi slaves node pada Hadoop cluster:

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
slave6:50010 (10.100.69.213:50010)	0	In Service	907.92 GB	282.77 MB	63.57 GB	844.08 GB	78	282.77 MB (0.03%)	0	2.7.1
slave9:50010 (10.100.69.216:50010)	2	In Service	224.02 GB	357.2 MB	28.91 GB	194.76 GB	76	357.2 MB (0.16%)	0	2.7.1
slave3:50010 (10.100.69.210:50010)	2	In Service	146.65 GB	394.67 MB	34.37 GB	111.89 GB	76	394.67 MB (0.26%)	0	2.7.1
slave7:50010 (10.100.69.214:50010)	0	In Service	907.92 GB	452.92 MB	298.94 GB	608.54 GB	63	452.92 MB (0.05%)	0	2.7.1
slave2:50010 (10.100.69.209:50010)	0	In Service	146.65 GB	1.09 GB	37.82 GB	107.74 GB	67	1.09 GB (0.74%)	0	2.7.1
slave4:50010 (10.100.69.211:50010)	2	In Service	907.92 GB	670.63 MB	64.06 GB	843.21 GB	73	670.63 MB (0.07%)	0	2.7.1
slave5:50010 (10.100.69.212:50010)	2	In Service	911.68 GB	498.01 MB	65.04 GB	846.15 GB	81	498.01 MB (0.05%)	0	2.7.1
slave1:50010 (10.100.69.207:50010)	0	In Service	146.65 GB	440.79 MB	57.85 GB	88.36 GB	75	440.79 MB (0.29%)	0	2.7.1
slave10:50010 (10.100.69.217:50010)	0	In Service	146.65 GB	473.57 MB	26.74 GB	119.44 GB	82	473.57 MB (0.32%)	0	2.7.1
slave8:50010 (10.100.69.215:50010)	0	In Service	146.65 GB	650.28 MB	29.53 GB	116.48 GB	79	650.28 MB (0.43%)	0	2.7.1

Gambar 5.23: Spesifikasi Slaves Node

- Node adalah nama dan alamat slaves node yang tersedia pada Hadoop cluster.
- Capacity adalah kapasitas penyimpanan data pada masing-masing slaves node.
- Used adalah jumlah kapasitas yang terpakai pada masing-masing slaves node.
- Remaining adalah sisa kapasitas penyimpanan pada masing-masing slaves node.
- Version adalah versi hadoop yang dipasang pada masing-masing slaves node.

### Perangkat Lunak Eksplorasi

Langkah pertama sebelum menjalankan perangkat lunak eksplorasi pada CLI adalah menyimpan data input pada sistem HDFS. Data input yang dibutuhkan antara lain dataset `adult100k.csv` dan parameter program `InputExploratory.json`. Listing 1 adalah perintah untuk membuat folder dan menyimpan data input pada HDFS. Hasil folder dan file yang disimpan pada HDFS dapat dilihat menggunakan browser dengan alamat `http://10.100.69.101:50070/`.

Listing 5.4: Perintah Spark untuk Perangkat Lunak Eksplorasi

```
// Perintah untuk membuat folder di HDFS
hadoop fs -mkdir /<nama folder HDFS>

// Perintah untuk menyimpan file adult100k.csv di HDFS
hadoop fs -put /home/hduser/<nama folder>/<nama file>.csv /<nama folder HDFS>

// Perintah untuk menyimpan file InputAnonymization.json di HDFS
hadoop fs -put /home/hduser/<nama folder>/<nama file>.json /<nama folder HDFS>

// Perintah untuk menghapus sebuah folder pada HDFS
hadoop fs -rm -r -f /<nama folder HDFS>
```

Langkah kedua adalah memastikan bahwa file sudah ditempatkan pada folder dengan nama yang sesuai. Hal ini perlu diperhatikan, karena jika terjadi kesalahan input pada nama file atau nama folder, program dapat menampilkan pesan error. Gambar 1 adalah contoh penempatan file pada folder HDFS dengan benar karena tidak ada kesalahan penulisan nama.

Browse Directory							
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	hduser	4.46 KB	10/25/2020, 7:39:35 PM	3	128 MB	InputAnonymization.json
-rw-r--r--	hduser	hduser	4.47 KB	10/27/2020, 5:22:43 PM	3	128 MB	InputAnonymization100k.json
-rw-r--r--	hduser	hduser	639 B	10/25/2020, 7:10:35 PM	3	128 MB	InputExamination.json
-rw-r--r--	hduser	hduser	486 B	10/25/2020, 7:10:35 PM	3	128 MB	InputExploratory.json
-rw-r--r--	hduser	hduser	10.71 MB	10/25/2020, 7:42:57 PM	3	128 MB	adult100k.csv
-rw-r--r--	hduser	hduser	108.88 MB	10/27/2020, 5:08:03 PM	3	128 MB	adult1m.csv
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 6:27:28 PM	0	0 B	output
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 5:23:35 PM	0	0 B	output100k

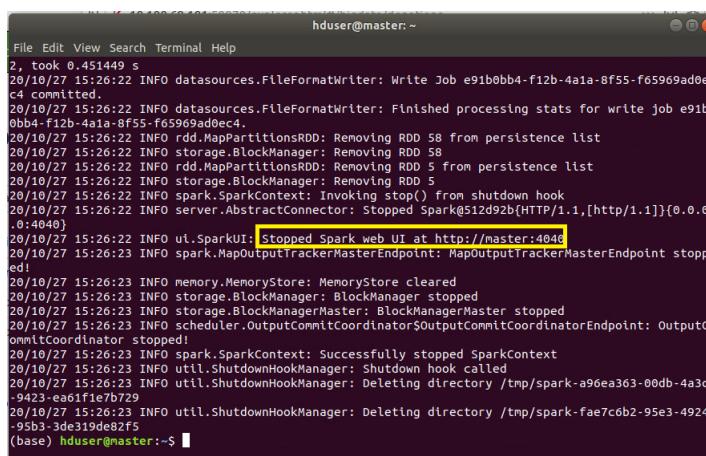
Gambar 5.24: File Input Eksplorasi HDFS

Langkah ketiga adalah melakukan eksekusi perangkat lunak eksplorasi menggunakan perintah Spark. Listing 1 adalah contoh perintah eksekusi pada Spark. Format `--class ExploratoryModel.MainExploratory` artinya menunjukkan kelas Main yang dieksekusi bernama `MainExploratory` pada package `ExploratoryModel`. Format `--master yarn` menunjukkan bahwa program dieksekusi pada sebuah cluster komputer. Format `/home/hduser/skripsi-stephen/skripsi.jar` menunjukkan lokasi JAR pada komputer. Format `/skripsi-jordan/InputExploratory.json` menunjukkan lokasi JSON pada HDFS. Pada tahap ini, perintah Spark siap untuk dieksekusi.

Listing 5.5: Perintah Eksekusi Spark

```
spark-submit --class ExploratoryModel.MainExploratory --master yarn /home/hduser/
skripsi-stephen/skripsi.jar /skripsi-jordan/InputExploratory.json
```

Langkah terakhir adalah menunggu proses komputasi sampai dengan selesai. Perangkat lunak eksplorasi yang telah berhasil menyelesaikan proses komputasinya pada command line ditandai dengan baris log seperti Gambar 5.25: 'Stopped Spark Web UI at http://master:4040'. Setelah proses komputasi selesai, maka output perangkat lunak eksplorasi sudah disimpan pada HDFS dengan lokasi sebagai berikut `/skripsi-jordan/output/exploratory-analysis`



```
hduser@master: ~
File Edit View Search Terminal Help
2, took 0.451449 s
20/10/27 15:26:22 INFO datasource.FileFormatWriter: Write Job e91b0bb4-f12b-4a1a-8f55-f65969ad0e
c4 committed.
20/10/27 15:26:22 INFO datasource.FileFormatWriter: Finished processing stats for write job e91b
0bb4-f12b-4a1a-8f55-f65969ad0ec4.
20/10/27 15:26:22 INFO rdd.MapPartitionsRDD: Removing RDD 58 from persistence list
20/10/27 15:26:22 INFO storage.BlockManager: Removing RDD 58
20/10/27 15:26:22 INFO rdd.MapPartitionsRDD: Removing RDD 5 from persistence list
20/10/27 15:26:22 INFO storage.BlockManager: Removing RDD 5
20/10/27 15:26:22 INFO spark.SparkContext: Invoking stop() from shutdown hook
20/10/27 15:26:22 INFO server.AbstractConnector: Stopped Spark@512d92b[HTTP/1.1]{0.0.0
.0:4040}
20/10/27 15:26:22 INFO ui.SparkUI: Stopped Spark web UI at http://master:4040
20/10/27 15:26:23 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stop
ped!
20/10/27 15:26:23 INFO memory.MemoryStore: MemoryStore cleared
20/10/27 15:26:23 INFO storage.BlockManager: BlockManager stopped
20/10/27 15:26:23 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
20/10/27 15:26:23 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputC
ommitCoordinator stopped!
20/10/27 15:26:23 INFO spark.SparkContext: Successfully stopped SparkContext
20/10/27 15:26:23 INFO util.ShutdownHookManager: Shutdown hook called
20/10/27 15:26:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-a96ea363-00db-4a3c
-9423-ea61fe7b729
20/10/27 15:26:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-fae7c6b2-95e3-4924
-95b3-3de319de82f5
(base) hduser@master:~
```

Gambar 5.25: Log Perangkat Lunak Eksplorasi

Ketika lokasi output pada HDFS dibuka, maka akan tampak seperti Gambar 1.1. Hasil eksplorasi nilai unik akan digunakan sebagai referensi mengisi nilai `domain_generalization_hierarchy` sebuah atribut tabel data pada `InputExploratory.json`. Eksplorasi perlu dilakukan mengingat seluruh nilai atribut harus dapat diubah menjadi nilai anonimisasi. Gambar 5.26 menunjukkan lokasi folder nilai unik untuk masing-masing atribut. Gambar 5.27 menunjukkan lokasi output disimpan `/skripsi-jordan/exploratory-analysis/workclass/part-000X-YYYYY.csv`.

Browse Directory							
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 9:57:39 PM	0	0 B	age
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 9:57:41 PM	0	0 B	income
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 9:57:40 PM	0	0 B	race
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 9:57:41 PM	0	0 B	sex

Gambar 5.26: Folder HDFS Hasil Ekplorasi

Browse Directory							
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rW-r--r-	hduser	hduser	0 B	10/27/2020, 9:57:40 PM	3	128 MB	_SUCCESS
-rw-r--r--	hduser	hduser	61 B	10/27/2020, 9:57:40 PM	3	128 MB	part-00000-clef8e3-d4dc-48ca-b9f0-574031e46224-c000.csv

Gambar 5.27: Folder HDFS Hasil Ekplorasi Atribut Race

Contoh output yang dihasilkan adalah tabel nilai unik dari atribut Race. Karena output disimpan dalam format CSV, baris pertama menyatakan nama kolom, sedangkan baris selanjutnya menyatakan nilai unik. Gambar 5.28 menunjukkan `Race` memiliki 5 jenis nilai unik.

exploratory-race.csv	
Open ▾	Save
1 race	
2 Other	
3 Amer-Indian-Eskimo	
4 White	
5 Asian-Pac-Islander	
6 Black	

Gambar 5.28: Hasil Eksplorasi pada Atribut Race

## Perangkat Lunak Anonimisasi

Langkah pertama sebelum menjalankan perangkat lunak eksplorasi pada CLI adalah menyimpan data input pada sistem HDFS. Data input yang dibutuhkan antara lain dataset `adult100k.csv` dan parameter program `InputExploratory.json`. Listing 1 adalah perintah untuk membuat folder dan menyimpan data input pada HDFS. Hasil folder dan file yang disimpan pada HDFS dapat dilihat menggunakan browser dengan alamat `http://10.100.69.101:50070/`.

Listing 5.6: Perintah Spark untuk Perangkat Lunak Anonimisasi

```
// Perintah untuk membuat folder di HDFS
hadoop fs -mkdir <nama folder HDFS>

// Perintah untuk menyimpan file adult100k.csv di HDFS
hadoop fs -put /home/hduser/<nama folder>/<nama file>.csv /<nama folder HDFS>/

// Perintah untuk menyimpan file InputAnonymization.json di HDFS
hadoop fs -put /home/hduser/<nama folder>/<nama file>.json /<nama folder HDFS>/

// Perintah untuk menghapus sebuah folder pada HDFS
hadoop fs -rm -r -f /<nama folder HDFS>
```

Langkah kedua adalah memastikan bahwa file sudah ditempatkan pada folder dengan nama yang sesuai. Hal ini perlu diperhatikan, karena jika terjadi kesalahan input pada nama file atau nama folder, program dapat menampilkan pesan error. Gambar 1 adalah contoh penempatan file pada folder HDFS dengan benar karena tidak ada kesalahan penulisan nama.

Browse Directory							
/skripsi-jordan							Go!
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	hduser	4.46 KB	10/25/2020, 7:39:35 PM	3	128 MB	InputAnonymization.json
-rw-r--r--	hduser	hduser	4.47 KB	10/27/2020, 5:22:43 PM	3	128 MB	InputAnonymization100k.json
-rw-r--r--	hduser	hduser	639 B	10/25/2020, 7:10:35 PM	3	128 MB	InputExamination.json
-rw-r--r--	hduser	hduser	486 B	10/25/2020, 7:10:35 PM	3	128 MB	InputExploratory.json
-rw-r--r--	hduser	hduser	10.71 MB	10/25/2020, 7:42:57 PM	3	128 MB	adult100k.csv
-rw-r--r--	hduser	hduser	108.88 MB	10/27/2020, 5:08:03 PM	3	128 MB	adult1m.csv
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 6:27:28 PM	0	0 B	output
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 5:23:35 PM	0	0 B	output100k

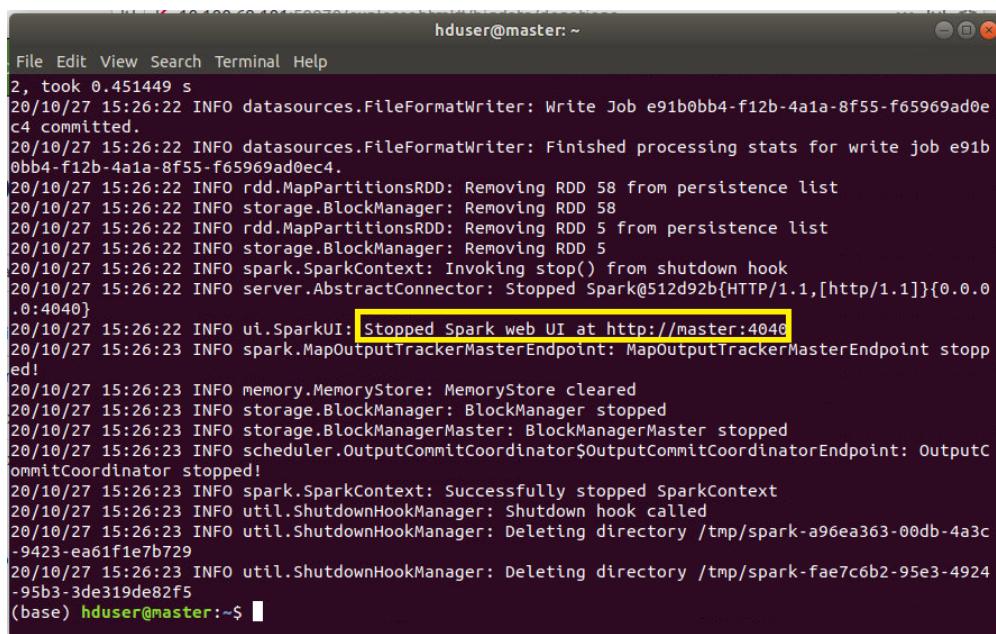
Gambar 5.29: File Input Anonimisasi HDFS

Langkah ketiga adalah melakukan eksekusi perangkat lunak eksplorasi menggunakan perintah Spark. Listing 1 adalah contoh perintah eksekusi pada Spark. Format `--class AnonymizationModel`. `MainAnonymization` artinya menunjukkan kelas Main yang dieksekusi bernama `MainAnonymization` pada package `AnonymizationModel`. Format `--master yarn` menunjukkan bahwa program dieksekusi pada sebuah cluster komputer. Format `/home/hduser/skripsi-stephen/skripsi.jar` menunjukkan lokasi JAR pada komputer. Format `/skripsi-jordan/InputAnonymization100k.json` menunjukkan lokasi JSON pada HDFS. Pada tahap ini, perintah siap dieksekusi.

Listing 5.7: Perintah Eksekusi Spark

```
spark-submit --class AnonymizationModel.MainAnonymization --master yarn /home/
hduser/skripsi-stephen/skripsi.jar /skripsi-jordan/InputAnonymization100k.json
```

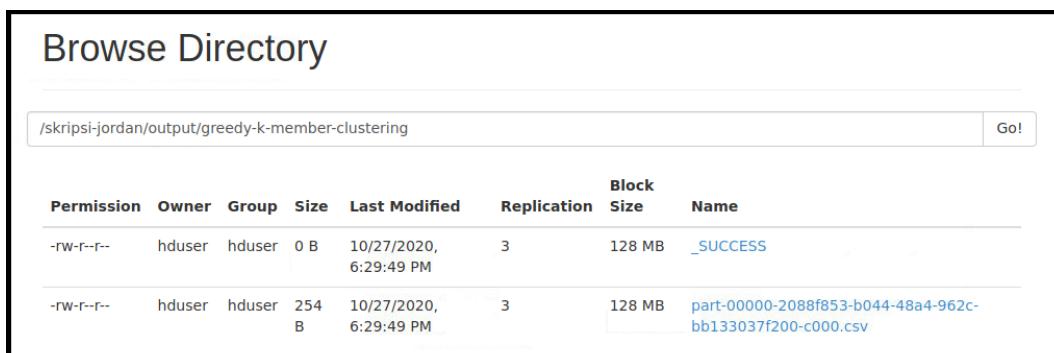
Langkah terakhir adalah menunggu proses komputasi sampai dengan selesai. Perangkat lunak eksplorasi yang telah berhasil menyelesaikan proses komputasinya pada command line ditandai dengan baris log seperti Gambar 5.25: 'Stopped Spark Web UI at http://master:4040'. Setelah proses komputasi selesai, maka output perangkat lunak anonimisasi telah tersimpan pada HDFS. Hasil pengelompokan data disimpan pada lokasi HDFS sebagai berikut /skripsi-jordan/output/greedy-k-member-clustering, sedangkan hasil anonimisasi data disimpan pada lokasi HDFS sebagai berikut /skripsi-jordan/output/k-anonymity.



```
File Edit View Search Terminal Help
2, took 0.451449 s
20/10/27 15:26:22 INFO datasources.FileFormatWriter: Write Job e91b0bb4-f12b-4a1a-8f55-f65969ad0e
c4 committed.
20/10/27 15:26:22 INFO datasources.FileFormatWriter: Finished processing stats for write job e91b
0bb4-f12b-4a1a-8f55-f65969ad0ec4.
20/10/27 15:26:22 INFO rdd.MapPartitionsRDD: Removing RDD 58 from persistence list
20/10/27 15:26:22 INFO storage.BlockManager: Removing RDD 58
20/10/27 15:26:22 INFO rdd.MapPartitionsRDD: Removing RDD 5 from persistence list
20/10/27 15:26:22 INFO storage.BlockManager: Removing RDD 5
20/10/27 15:26:22 INFO spark.SparkContext: Invoking stop() from shutdown hook
20/10/27 15:26:22 INFO server.AbstractConnector: Stopped Spark@512d92b[HTTP/1.1,[http/1.1]}{0.0.0
.0:4040}
20/10/27 15:26:22 INFO ui.SparkUI: Stopped Spark web UI at http://master:4040
20/10/27 15:26:23 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopp
ed!
20/10/27 15:26:23 INFO memory.MemoryStore: MemoryStore cleared
20/10/27 15:26:23 INFO storage.BlockManager: BlockManager stopped
20/10/27 15:26:23 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
20/10/27 15:26:23 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputC
ommitCoordinator stopped!
20/10/27 15:26:23 INFO spark.SparkContext: Successfully stopped SparkContext
20/10/27 15:26:23 INFO util.ShutdownHookManager: Shutdown hook called
20/10/27 15:26:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-a96ea363-00db-4a3c
-9423-ea61fie7b729
20/10/27 15:26:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-fae7c6b2-95e3-4924
-95b3-3de319de82f5
(base) hduser@master:~$
```

Gambar 5.30: Log Perangkat Lunak Anonimisasi

Ketika lokasi output pada HDFS dibuka, maka akan tampak seperti Gambar 1.1. Hasil eksplorasi nilai unik akan digunakan sebagai referensi mengisi nilai domain\_generalization\_hierarchy sebuah atribut tabel data pada InputAnonymization.json. Gambar 5.27 menunjukan hasil pengelompokan data pada /skripsi-jordan/greedy-k-member-clustering/part-000X-YYYYY.csv. Gambar 5.28 menunjukan hasil pengelompokan data pada /skripsi-jordan/k-anonymity/part-000X-YYYYY.csv. Untuk membuka hasil output, maka CSV harus diunduh terlebih dahulu.



Browse Directory							
/skripsi-jordan/output/greedy-k-member-clustering							
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	hduser	0 B	10/27/2020, 6:29:49 PM	3	128 MB	_SUCCESS
-rw-r--r--	hduser	hduser	254 B	10/27/2020, 6:29:49 PM	3	128 MB	part-00000-2088f853-b044-48a4-962c-bb133037f200-c000.csv

Gambar 5.31: Folder HDFS Hasil Pengelompokan Data

Browse Directory							
/skripsi-jordan/output/k-anonymity							Go!
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	hduser	0 B	10/27/2020, 6:30:44 PM	3	128 MB	_SUCCESS
-rw-r--r--	hduser	hduser	221 B	10/27/2020, 6:30:44 PM	3	128 MB	part-00000-35966531-d740-4874-b8b3-8af32707618-c000.csv

Gambar 5.32: Folder HDFS Hasil Anonimisasi Data

Contoh output yang dihasilkan adalah tabel pengelompokan data dan tabel anonimisasi data. Karena output disimpan dalam format CSV, baris pertama menyatakan nama kolom, sedangkan baris selanjutnya menyimpan data. Gambar 5.33 menunjukkan hasil pengelompokan data. Gambar 5.33 menunjukkan hasil anonimisasi data. Output disimpan dalam format CSV.

greedy-k-member-clustering.csv							
~/Downloads							
1	id,age,race,sex,workclass,income,Cluster						
2	1,39,White,Male,State-gov,<=50K,Cluster 1						
3	3,38,White,Male,Private,<=50K,Cluster 1						
4	2,50,White,Male,Self-emp-not-inc,<=50K,Cluster 2						
5	4,53,Black,Male,Private,<=50K,Cluster 2						
6	5,28,Black,Female,Private,<=50K,Cluster 1						

Gambar 5.33: Hasil Pengelompokan Greedy k-member clustering

k-anonymity.csv							
~/Downloads							
greedy-k-member-clustering.csv	x	k-anonymity.csv	x				
1	id,age,race,sex,workclass,income						
2	1,[28-39],Person,Adult,Employee,<=50K						
3	2,[50-53],Person,Male,Employee,<=50K						
4	3,[28-39],Person,Adult,Employee,<=50K						
5	4,[50-53],Person,Male,Employee,<=50K						
6	5,[28-39],Person,Adult,Employee,<=50K						

Gambar 5.34: Hasil Anonimisasi K-Anonymity

## Perangkat Lunak Pengujian

Langkah pertama sebelum menjalankan perangkat lunak pengujian pada CLI adalah menyimpan data input pada sistem HDFS. Data input yang dibutuhkan antara lain dataset `adult100k.csv` dan parameter program `InputExploratory.json`. Listing 1 adalah perintah untuk membuat folder dan menyimpan data input pada HDFS. Hasil folder dan file yang disimpan pada HDFS dapat dilihat menggunakan browser dengan alamat `http://10.100.69.101:50070/`.

Listing 5.8: Perintah Spark untuk Perangkat Lunak Pengujian

```
// Perintah untuk membuat folder di HDFS
hadoop fs -mkdir /<nama folder HDFS>

// Perintah untuk menyimpan file adult100k.csv di HDFS
hadoop fs -put /home/hduser/<nama folder>/<nama file>.csv /<nama folder HDFS>/

// Perintah untuk menyimpan file InputAnonymization.json di HDFS
hadoop fs -put /home/hduser/<nama folder>/<nama file>.json /<nama folder HDFS>/

// Perintah untuk menghapus sebuah folder pada HDFS
hadoop fs -rm -r -f /<nama folder HDFS>
```

Langkah kedua adalah memastikan bahwa file sudah ditempatkan pada folder dengan nama yang sesuai. Hal ini perlu diperhatikan, karena jika terjadi kesalahan input pada nama file atau nama folder, program dapat menampilkan pesan error. Gambar 1 adalah contoh penempatan file pada folder HDFS dengan benar karena tidak ada kesalahan penulisan nama.

### Browse Directory

Browse Directory							
/skripsi-jordan							
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	hduser	4.46 KB	10/25/2020, 7:39:35 PM	3	128 MB	InputAnonymization.json
-rw-r--r--	hduser	hduser	4.47 KB	10/27/2020, 5:22:43 PM	3	128 MB	InputAnonymization100k.json
-rw-r--r--	hduser	hduser	639 B	10/25/2020, 7:10:35 PM	3	128 MB	InputExamination.json
-rw-r--r--	hduser	hduser	486 B	10/25/2020, 7:10:35 PM	3	128 MB	InputExploratory.json
-rw-r--r--	hduser	hduser	10.71 MB	10/25/2020, 7:42:57 PM	3	128 MB	adult100k.csv
-rw-r--r--	hduser	hduser	108.88 MB	10/27/2020, 5:08:03 PM	3	128 MB	adult1m.csv
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 6:27:28 PM	0	0 B	output
drwxr-xr-x	hduser	hduser	0 B	10/27/2020, 5:23:35 PM	0	0 B	output100k

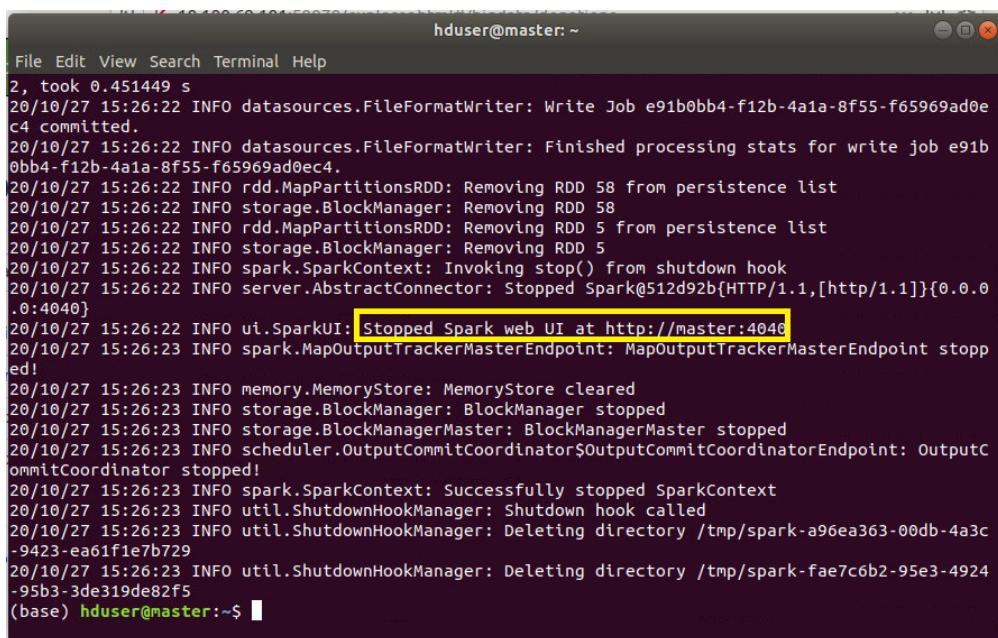
Gambar 5.35: File Input Pengujian HDFS

Langkah ketiga adalah melakukan eksekusi perangkat lunak pengujian menggunakan perintah Spark. Listing 1 adalah contoh perintah eksekusi pada Spark. Format `--class ExaminationModel.MainAnonymization` artinya menunjukkan kelas Main yang dieksekusi bernama `MainAnonymization` pada package `ExaminationModel`. Format `--master yarn` menunjukkan bahwa program dieksekusi pada sebuah cluster komputer. Format `/home/hduser/skripsi-stephen/skripsi.jar` menunjukkan lokasi JAR pada komputer. Format `/skripsi-jordan/InputKMeans.json` menunjukkan lokasi JSON pada HDFS untuk pemodelan k-means dan format `/skripsi-jordan/InputNaiveBayes.json` menunjukkan lokasi JSON pada HDFS untuk pemodelan naive bayes.

Listing 5.9: Perintah Eksekusi Spark

```
spark-submit --class ExaminationModel.MainExamination --master yarn /home/hduser/skripsi-stephen/skripsi.jar /skripsi-jordan/InputExamination.json
```

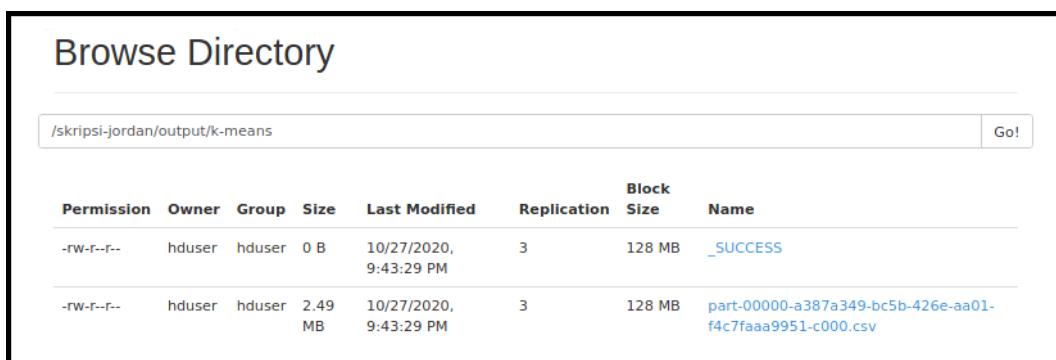
Langkah terakhir adalah menunggu proses komputasi sampai dengan selesai. Perangkat lunak pengujian yang telah berhasil menyelesaikan proses komputasinya pada command line ditandai dengan baris log seperti Gambar 5.25: 'Stopped Spark Web UI at http://master:4040'. Setelah proses komputasi selesai, maka hasil pengujian k-means disimpan sebagai format CSV pada lokasi HDFS berikut /skripsi-jordan/output/k-means, sedangkan hasil pemodelan naive bayes disimpan sebagai format CSV pada lokasi HDFS berikut /skripsi-jordan/output/naive-bayes.



```
hduser@master: ~
File Edit View Search Terminal Help
2, took 0.451449 s
20/10/27 15:26:22 INFO datasources.FileFormatWriter: Write Job e91b0bb4-f12b-4a1a-8f55-f65969ad0e
c4 committed.
20/10/27 15:26:22 INFO datasources.FileFormatWriter: Finished processing stats for write job e91b
0bb4-f12b-4a1a-8f55-f65969ad0ec4.
20/10/27 15:26:22 INFO rdd.MapPartitionsRDD: Removing RDD 58 from persistence list
20/10/27 15:26:22 INFO storage.BlockManager: Removing RDD 58
20/10/27 15:26:22 INFO rdd.MapPartitionsRDD: Removing RDD 5 from persistence list
20/10/27 15:26:22 INFO storage.BlockManager: Removing RDD 5
20/10/27 15:26:22 INFO spark.SparkContext: Invoking stop() from shutdown hook
20/10/27 15:26:22 INFO server.AbstractConnector: Stopped Spark@512d92b[HTTP/1.1]{0.0.0
.0:4040}
20/10/27 15:26:22 INFO ui.SparkUI: Stopped Spark web UI at http://master:4040
20/10/27 15:26:23 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopp
ed!
20/10/27 15:26:23 INFO memory.MemoryStore: MemoryStore cleared
20/10/27 15:26:23 INFO storage.BlockManager: BlockManager stopped
20/10/27 15:26:23 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
20/10/27 15:26:23 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputC
ommitCoordinator stopped!
20/10/27 15:26:23 INFO spark.SparkContext: Successfully stopped SparkContext
20/10/27 15:26:23 INFO util.ShutdownHookManager: Shutdown hook called
20/10/27 15:26:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-a96ea363-00db-4a3c
-9423-ea61f1e7b729
20/10/27 15:26:23 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-fae7c6b2-95e3-4924
-95b3-3de319de82f5
(base) hduser@master:~$
```

Gambar 5.36: Log Perangkat Lunak Pengujian

Ketika lokasi output pada HDFS dibuka, maka akan tampak seperti Gambar 1.1. Hasil eksplorasi nilai unik akan digunakan sebagai referensi mengisi nilai domain\_generalization\_hierarchy sebuah atribut tabel data pada InputAnonymization.json. Eksplorasi perlu dilakukan mengingat seluruh nilai atribut harus dapat diubah menjadi nilai anonimisasi. Gambar 5.26 menunjukan lokasi HDFS yang menyimpan hasil pengolompokan data pada CSV /skripsi-jordan/output/naive-bayes/part-000X-YYYYY.csv. Gambar 5.27 menunjukan lokasi HDFS yang menyimpan hasil klasifikasi data pada CSV /skripsi-jordan/output/naive-bayes/part-000X-YYYYY.csv.



Gambar 5.37: Folder HDFS Hasil Pengelompokan K-Means

Browse Directory								
/skripsi-jordan/output/naive-bayes								Go!
Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
-rw-r--r--	hduser	hduser	0 B	10/27/2020, 9:44:30 PM	3	128 MB	_SUCCESS	
-rw-r--r--	hduser	hduser	814.79 KB	10/27/2020, 9:44:30 PM	3	128 MB	part-00000-c76ed676-c4ba-4e82-95c3-ef00aaa3606c-c000.csv	

Gambar 5.38: Folder HDFS Hasil Klasifikasi Naive Bayes

Contoh output yang dihasilkan adalah tabel pengelompokan data k-means dan tabel klasifikasi data naive bayes. Karena output disimpan dalam format CSV, baris pertama menyatakan nama kolom, sedangkan baris selanjutnya menyatakan data. Gambar 5.28 menunjukkan tabel pengelompokan data k-means. Gambar 5.29 menunjukkan tabel klasifikasi data naive bayes.

k-means.csv								
~/Downloads								
1	age,race,sex,income,prediction							
2	39,White,Male,<=50K,1							
3	50,White,Male,<=50K,1							
4	38,White,Male,<=50K,1							
5	53,Black,Male,<=50K,0							
6	28,Black,Female,<=50K,1							
7	37,White,Female,<=50K,1							
8	49,Black,Female,<=50K,1							
9	52,White,Male,>50K,1							
10	31,White,Female,>50K,1							
11	42,White,Male,>50K,1							
12	37,Black,Male,>50K,1							
13	30,Asian-Pac-Islander,Male,>50K,1							
14	23,White,Female,<=50K,1							
15	32,Black,Male,<=50K,1							
16	40,Asian-Pac-Islander,Male,>50K,1							
17	34,Amer-Indian-Eskimo,Male,<=50K,1							
18	25,White,Male,<=50K,1							
19	22,White,Male,<=50K,1							

Gambar 5.39: Hasil Pengelompokan K-Means

naive-bayes.csv								
~/Downloads								
1	age,race,sex,income,prediction							
2	17,Amer-Indian-Eskimo,Female,<=50K,0.0							
3	17,Amer-Indian-Eskimo,Male,<=50K,0.0							
4	17,Amer-Indian-Eskimo,Male,>50K,1.0							
5	17,Amer-Indian-Eskimo,Male,>50K,1.0							
6	17,Asian-Pac-Islander,Female,<=50K,0.0							
7	17,Asian-Pac-Islander,Female,<=50K,0.0							
8	17,Asian-Pac-Islander,Female,<=50K,0.0							
9	17,Asian-Pac-Islander,Female,>50K,1.0							
10	17,Asian-Pac-Islander,Female,>50K,1.0							
11	17,Asian-Pac-Islander,Male,>50K,1.0							
12	17,Asian-Pac-Islander,Male,>50K,1.0							
13	17,Black,Female,<=50K,0.0							
14	17,Black,Female,<=50K,0.0							
15	17,Black,Female,<=50K,0.0							
16	17,Black,Female,<=50K,0.0							
17	17,Black,Female,<=50K,0.0							
18	17,Black,Female,<=50K,0.0							
19	17,Black,Female,<=50K,0.0							

Gambar 5.40: Hasil Klasifikasi Naive Bayes

## 5.2 Pengujian

Pengujian pada penelitian ini dibagi menjadi dua tahap, yaitu pengujian fungsional dan pengujian eksperimental. Pengujian fungsional bertujuan untuk melihat apakah fungsi-fungsi pada perangkat lunak sudah menghasilkan keluaran yang sesuai. Pengujian eksperimental bertujuan untuk melihat pengaruh data yang telah dianonimisasi pada proses data mining.

### 5.2.1 Pengujian Fungsional

Pengujian fungsional memastikan bahwa fungsi perangkat lunak anonimisasi berjalan dengan benar. Tahapan pengujian fungsional adalah melakukan pemeriksaan output pengelompokan data (greedy k-member clustering), anonimisasi data (k-anonymity), pemodelan data mining dengan model clustering (k-means) dan model klasifikasi (naive bayes).

#### Pengelompokan Data dengan Greedy k-member clustering

Pengujian pengelompokan data dilakukan pada perangkat lunak anonimisasi. Pengujian ini menggunakan data set credit score yang berisi data pribadi pemohon kartu kredit untuk memprediksi kemungkinan gagal bayar dan pinjaman kartu kredit di masa depan. Dataset ini terdiri dari atribut numerik dan atribut kategorikal. Hasil pengelompokan data sudah disimpan dengan format CSV untuk mempermudah observasi. Gambar ?? adalah contoh sampel data yang diambil dari data set Credit score yang terdiri dari 18 jenis atribut.

Listing 5.10: Sampel Data Credit Score

ID	DAY_BIRTH	DAY_EMPLOYED	OCCUPATION_TYPE	CODE_GENDER	NAME_INCOME_TYPE	AMT_INCOME_TOTAL
5008801	-19110	-3051	Sales staff	F	Commercial associate	270000
5008802	-19110	-3051	Sales staff	F	Commercial associate	270000
5008803	-21474	-1134	Security staff	M	Working	112500
5008804	-19110	-3051	Sales staff	F	Commercial associate	270000
5008805	-12005	-4542	""	M	Working	427500
5008806	-12005	-4542	""	M	Working	427500
5008807	-19110	-3051	Sales staff	Commercial associate	1	
5008808	-22464	365243	""	Pensioner	0	
5008809	-22464	365243	""	Pensioner	0	
5008810	-22464	365243	""	Pensioner	0	

Hasil yang ingin dicapai dari pengujian ini adalah perangkat lunak dapat mengeluarkan hasil pengelompokan data menggunakan algoritma greedy k-member clustering. Setiap baris pada data yang memiliki hubungan yang dekat dengan baris data lainnya telah dikelompokan ke dalam satu cluster dan jumlah anggota sebuah cluster bergantung pada nilai k yang ditetapkan sebelumnya. Selain itu, hasil pengelompokan data dan perhitungan distance record telah sesuai dengan perhitungan manual yang dilakukan pada studi kasus. Pada pengujian ini, nilai k yang digunakan adalah 2 sehingga masing-masing cluster yang terbentuk minimal terdiri dari 2 data . Gambar ?? adalah hasil pengelompokan data dengan algoritma greedy k-member clustering.

Listing 5.11: Hasil Pengelompokan Greedy K-Member Clustering

id	DAY_BIRTH	DAY_EMPLOYED	OCCUPATION_TYPE	CODE_GENDER	NAME_INCOME_TYPE	AMT_INCOME_TOTAL	Cluster
2	-12005	-4542	""	M	Working	427500	Cluster 3
8	-22464	365243	""	F	Pensioner	283500	Cluster 3
5	-19110	-3051	Sales staff	F	Commercial associate	270000	Cluster 1

```

1,-12005,-4542,"",M,Working,427500,Cluster 1
7,-19110,-3051,Sales staff,F,Commercial associate,270000,Cluster 1
9,-22464,365243,"",F,Pensioner,283500,Cluster 3
3,-21474,-1134,Security staff,M,Working,112500,Cluster 2
4,-19110,-3051,Sales staff,F,Commercial associate,270000,Cluster 2
10,-22464,365243,"",F,Pensioner,283500,Cluster 2
6,-19110,-3051,Sales staff,F,Commercial associate,270000,Cluster 2

```

Berdasarkan hasil pengujian yang didapat, dapat disimpulkan bahwa perangkat lunak anonimisasi dapat melakukan perhitungan infomation loss dengan benar. Hal ini didasari dari Subbab 2.1 bahwa pengelompokan data dengan information loss yang kecil berpotensi tinggi menjadi satu kelompok dan telah dibuktikan pada Gambar ?? bahwa data-data yang memiliki nilai numerik atau kategorikal yang berdekatan telah dikelompokan ke dalam cluster yang sama. Hal ini terjadi karena data-data dengan nilai yang berdekatan memiliki bobot information loss yang relatif lebih kecil. Hasil pengelompokan data yang didapat dari program anonimisasi telah sesuai dengan hasil pengelompokan data yang dilakukan sebelumnya pada studi kasus Subbab 3.2.

### Anonimisasi Data dengan Metode K-Anonymity

Pengujian ini dilakukan pada perangkat lunak anonimisasi. Pengujian anomisasi data dilakukan dengan memakai input dari hasil pengelompokan data yang berisi kumpulan data-data pada cluster tertentu. Dataset ini terdiri dari atribut numerik dan atribut kategorikal. Pengelompokan data telah dilakukan sebelum melakukan tahap anonimisasi, sehingga diharapkan nilai informasi yang didapat masih cukup baik karena proses anonimisasi data dilakukan per masing-masing kelompok data. Hasil anonimisasi data telah disimpan dalam CSV untuk dilakukan observasi.

Listing 5.12: Sampel Pengelompokan Data

```

id,DAYS_BIRTH,DAYS_EMPLOYED,OCCUPATION_TYPE,CODE_GENDER,NAME_INCOME_TYPE,
AMT_INCOME_TOTAL,Cluster
2,-12005,-4542,"",M,Working,427500,Cluster 3
8,-22464,365243,"",F,Pensioner,283500,Cluster 3
5,-19110,-3051,Sales staff,F,Commercial associate,270000,Cluster 1
1,-12005,-4542,"",M,Working,427500,Cluster 1
7,-19110,-3051,Sales staff,F,Commercial associate,270000,Cluster 1
9,-22464,365243,"",F,Pensioner,283500,Cluster 3
3,-21474,-1134,Security staff,M,Working,112500,Cluster 2
4,-19110,-3051,Sales staff,F,Commercial associate,270000,Cluster 2
10,-22464,365243,"",F,Pensioner,283500,Cluster 2
6,-19110,-3051,Sales staff,F,Commercial associate,270000,Cluster 2

```

Hasil yang ingin dicapai dari pengujian ini adalah perangkat lunak dapat mengeluarkan hasil anonimisasi data menggunakan metode k-anonymity. Dimana setiap baris pada data tidak dapat dibedakan dengan  $k - 1$  baris data lainnya, sehingga untuk mencapai pernyataan tersebut, maka proses anonimisasi data dilakukan per kelompok data/cluster. Oleh karena itu, hasil anonimisasi data antara kelompok data yang satu dengan yang kelompok data lainnya saling berbeda satu sama lain. Selain itu, hasil pengujian anonimisasi data dan perhitungan information loss harus sesuai dengan perhitungan manual yang dilakukan pada studi kasus di Subbab 3.12. Pada pengujian ini, nilai  $k$  yang digunakan adalah 2 sehingga masing-masing cluster yang terbentuk minimal terdiri dari 2 data . Gambar 1.1 adalah hasil anonimisasi data dengan metode k-anonymity.

Listing 5.13: Hasil Anonimisasi K-Anonymity

```

id,Anonym_DAYS_BIRTH,Anonym_DAYS_EMPLOYED,Anonym_OCCUPATION_TYPE,
Anonym_CODE_GENDER,Anonym_NAME_INCOME_TYPE,AMT_INCOME_TOTAL
1, [(-19110)-(-12005)], [(-4542)-(-3051)], Sales staff, Adult, Unique earnings, 427500
2, [(-21474)-(-12005)], [(-4542)-(-1134)], Employee, Adult, Unique earnings, 427500
3, [(-21474)-(-12005)], [(-4542)-(-1134)], Employee, Adult, Unique earnings, 112500
4, [(-21474)-(-12005)], [(-4542)-(-1134)], Employee, Adult, Unique earnings, 270000
5, [(-19110)-(-12005)], [(-4542)-(-3051)], Sales staff, Adult, Unique earnings, 270000
6, [(-21474)-(-12005)], [(-4542)-(-1134)], Employee, Adult, Unique earnings, 270000
7, [(-19110)-(-12005)], [(-4542)-(-3051)], Sales staff, Adult, Unique earnings, 270000
8, -22464, 365243, "", F, Pensioner, 283500
9, -22464, 365243, "", F, Pensioner, 283500
10, -22464, 365243, "", F, Pensioner, 283500

```

Berdasarkan hasil anonimisasi data yang didapat, dapat disimpulkan bahwa perangkat lunak anonimisasi sudah dapat melakukan proses anonimisasi data menggunakan metode k-anonymity. Hal ini dibuktikan bahwa masing-masing nilai atribut telah dilakukan generalisasi ke nilai yang lebih umum. Selain itu, hasil dari pengelompokan data juga sudah sesuai dengan konsep anonimisasi data, karena setiap baris pada data tidak dapat dibedakan dengan  $k - 1$  baris data lainnya. Jika dilihat lebih seksama, nilai-nilai data pada Gambar 5.14 terlihat mirip satu sama lain, sehingga tujuan metode k-anonymity dapat dikatakan tercapai untuk hasil anonimisasi data.

### Pembuatan Model Data Mining dengan K-Means

Pengujian ini dilakukan pada perangkat lunak pengujian. Pengujian pemodelan data k-means data dilakukan dengan memakai input sebelum dan setelah data dilakukan proses anonimisasi. Pengujian ini dilakukan untuk membandingkan hasil pengelompokan data sebelum dan setelah dilakukan anonimisasi. Selanjutnya, dilakukan perbandingan selisih silhouette score antara kedua model data tersebut, untuk mengetahui jumlah informasi yang hilang selama data dilakukan anonimisasi. Hasil pemodelan k-means disimpan dalam CSV untuk dilakukan observasi lebih lanjut.

Hasil yang ingin dicapai dari pengujian ini adalah perangkat lunak dapat mengeluarkan hasil pengelompokan data menggunakan model k-means dan menampilkan hasil evaluasi model menggunakan silhouette score. Dimana semakin kecil nilai selisih silhouette score antara kedua model maka model yang dibuat semakin mendekati informasi pada data yang belum dianonimisasi. Oleh karena itu, perlu dilakukan pengujian eksperimental untuk mencari model k-means terbaik dengan mencari nilai  $k$  terbaik. Pada pengujian ini, nilai  $k$  yang digunakan adalah 2 sehingga masing-masing cluster yang terbentuk, minimal terdiri dari 2 data. Gambar 1.1 adalah hasil pengelompokan data dengan model k-means. Gambar 1.2 adalah hasil silhouette score untuk model k-means.

Listing 5.14: Hasil Pengelompokan K-Means Sebelum Anonimisasi

```

DAYS_BIRTH,DAYS_EMPLOYED,OCCUPATION_TYPE,NAME_INCOME_TYPE,prediction
-12005,-4542,"",Working,0
-12005,-4542,"",Working,0
-21474,-1134,Security staff,Working,2
-19110,-3051,Sales staff,Commercial associate,1
-19110,-3051,Sales staff,Commercial associate,1
-19110,-3051,Sales staff,Commercial associate,1
-19110,-3051,Sales staff,Commercial associate,1
-22464,365243,"",Pensioner,0
-22464,365243,"",Pensioner,0
-22464,365243,"",Pensioner,0

```

Listing 5.15: Hasil Pengelompokan K-Means Setelah Anonimisasi

```
Anonym_DAYS_BIRTH,Anonym_DAYS_EMPLOYED,Anonym_OCCUPATION_TYPE,
Anonym_NAME_INCOME_TYPE,prediction
[(-22464)--(-12005)], [(-4542)-365243], Sales staff, Unique earnings, 0
[(-22464)--(-12005)], [(-4542)-365243], Sales staff, Unique earnings, 0
[(-22464)--(-19110)], [(-3051)-365243], Employee, Unique earnings, 1
[(-22464)--(-12005)], [(-4542)-365243], Sales staff, Unique earnings, 0
[(-22464)--(-19110)], [(-3051)-365243], Employee, Unique earnings, 1
[(-22464)--(-12005)], [(-4542)-365243], Sales staff, Unique earnings, 0
[(-22464)--(-19110)], [(-3051)-365243], Employee, Unique earnings, 1
```

Listing 5.16: Perbedaan Silhouette Score

```
Info,Silhouette score
Normal table,0.7857142857142856
Anonymize table,1.0
How much different is Silhouette score,0.2142857142857144
```

Listing 5.17: Perbedaan Persentase Hasil Pengelompokan (%)

```
Percentage of Clustering Difference
0.5
```

Berdasarkan hasil anonimisasi data yang didapat, dapat disimpulkan bahwa perangkat lunak anonimisasi sudah dapat melakukan proses anonimisasi data menggunakan metode k-anonymity. Hal ini dibuktikan bahwa masing-masing nilai atribut telah dilakukan generalisasi ke nilai yang lebih umum. Selain itu, hasil dari pengelompokan data juga sudah sesuai dengan konsep anonimisasi data, karena setiap baris pada data tidak dapat dibedakan dengan  $k - 1$  baris data lainnya. Jika dilihat lebih seksama, nilai-nilai data pada Gambar 5.14 terlihat mirip satu sama lain, sehingga tujuan metode k-anonymity dapat dikatakan tercapai untuk hasil anonimisasi data.

### Pencarian Model Data Mining Terbaik dengan Naive Bayes

Pengujian ini dilakukan pada perangkat lunak pengujian. Pengujian pemodelan data naive bayes dilakukan dengan memakai input sebelum dan setelah data dilakukan proses anonimisasi. Pengujian ini dilakukan untuk membandingkan hasil pengelompokan data sebelum dan setelah dilakukan anonimisasi. Selanjutnya, dilakukan perbandingan selisih silhouette score antara kedua model data tersebut, untuk mengetahui jumlah informasi yang hilang selama data dilakukan anonimisasi. Hasil pemodelan naive bayes disimpan dalam CSV untuk dilakukan observasi.

Hasil yang ingin dicapai dari pengujian ini adalah perangkat lunak dapat mengeluarkan hasil pengelompokan data menggunakan model naive bayes dan menampilkan hasil evaluasi model menggunakan accuracy. Dimana semakin kecil selisih nilai accuracy antara kedua model, maka model yang dibuat semakin mendekati informasi pada data yang belum dianonimisasi. Oleh karena itu, semakin banyak data yang digunakan untuk pembuatan model, maka diharapkan akurasinya semakin baik. Pada pengujian ini, data akan dibagi menjadi 70% data training dan 30% data pelatihan. Gambar 1.1 adalah hasil pengelompokan data dengan model k-means. Gambar 1.2 adalah selisih nilai akurasi untuk model naive bayes.

Listing 5.18: Hasil Klasifikasi Naive Bayes Sebelum Anonimisasi

```

id,DAYS_BIRTH,DAYS_EMPLOYED,OCCUPATION_TYPE,NAME_INCOME_TYPE,prediction
1,-12005,-4542,"",Working,0.0
2,-12005,-4542,"",Working,0.0
3,-21474,-1134,Security staff,Working,1.0
4,-19110,-3051,Sales staff,Commercial associate,0.0
5,-19110,-3051,Sales staff,Commercial associate,0.0
6,-19110,-3051,Sales staff,Commercial associate,0.0
7,-19110,-3051,Sales staff,Commercial associate,0.0
8,-22464,365243,"",Pensioner,0.0
9,-22464,365243,"",Pensioner,0.0
10,-22464,365243,"",Pensioner,0.0

```

Listing 5.19: Hasil Klasifikasi Naive Bayes Setelah Anonimisasi

```

id,Anonym_DAYS_BIRTH,Anonym_DAYS_EMPLOYED,Anonym_OCCUPATION_TYPE,
Anonym_NAME_INCOME_TYPE,prediction
1,[(--22464)--(-12005)],[(-4542)-365243],Sales staff,Unique earnings,0.0
2,[(--22464)--(-12005)],[(-4542)-365243],Sales staff,Unique earnings,0.0
3,[(--22464)--(-19110)],[(-3051)-365243],Employee,Unique earnings,1.0
4,[(--22464)--(-12005)],[(-4542)-365243],Sales staff,Unique earnings,0.0
5,[(--22464)--(-19110)],[(-3051)-365243],Employee,Unique earnings,1.0
6,[(--22464)--(-12005)],[(-4542)-365243],Sales staff,Unique earnings,0.0
7,[(--22464)--(-12005)],[(-4542)-365243],Sales staff,Unique earnings,0.0
8,[(--22464)--(-12005)],[(-4542)-365243],Sales staff,Unique earnings,0.0
9,[(--22464)--(-12005)],[(-4542)-365243],Sales staff,Unique earnings,0.0
10,[(--22464)--(-19110)],[(-3051)-365243],Employee,Unique earnings,1.0

```

Listing 5.20: Perbedaan Tingkat Akurasi

```

Info,Accuracy
Normal table,0.70
Anonymize table,0.50
How much different is accuracy,0.20

```

Listing 5.21: Perbedaan Persentase Hasil Klasifikasi (%)

Percentage of Classification Difference
0.52

Berdasarkan hasil anonimisasi data yang didapat, dapat disimpulkan bahwa perangkat lunak anonimisasi sudah dapat melakukan proses anonimisasi data menggunakan metode k-anonymity. Hal ini dibuktikan bahwa masing-masing nilai atribut telah dilakukan generalisasi ke nilai yang lebih umum. Selain itu, hasil dari pengelompokan data juga sudah sesuai dengan konsep anonimisasi data, karena setiap baris pada data tidak dapat dibedakan dengan k - 1 baris data lainnya. Jika dilihat lebih seksama, nilai-nilai data pada Gambar 5.14 terlihat mirip satu sama lain, sehingga tujuan metode k-anonymity dapat dikatakan tercapai untuk hasil anonimisasi data.

### 5.2.2 Pengujian Eksperimental

Pengujian eksperimental bertujuan untuk meningkatkan kualitas hasil anonimisasi dari metode k-anonymity, agar data yang dihasilkan memiliki information loss yang rendah sehingga utilitas data tetap terjaga. Hasil dari pengujian ini menghasilkan model greedy k-member clustering dan k-anonymity terbaik untuk digunakan pada kasus perlindungan privasi lainnya.

Lingkungan pengujian eksperimental untuk menguji perangkat lunak anonimisasi data adalah pada komputer dengan prosesor Intel(R) Core(TM) i7-4720HQ @ 2.6 GHz dan 4GB RAM, dengan Java(TM) JDK 8, Spark 2.4.3, Hadoop 3.1.2, dan Scala 2.11.12. Pada pengujian eksperimental, digunakan IntelliJ untuk menjalankan seluruh perangkat lunak agar lebih praktis.

Untuk melakukan pengujian eksperimental, digunakan data kartu skor kredit yang didapat dari Kaggle. Kartu skor kredit adalah metode pengendalian risiko yang umum di industri keuangan. Data ini menggunakan informasi pribadi dari pemohon kartu kredit untuk memprediksi kemungkinan gagal bayar dan pinjaman kartu kredit di masa depan. Tujuan dari data ini adalah sebagai histori agar bank dapat memutuskan apakah kartu kredit diberikan/tidak kepada pemohon. Skor kredit dapat mengukur secara objektif besarnya risiko bank memberikan pinjaman kredit.

Berikut adalah tahapan dari masing-masing pengujian eksperimental:

Tabel 5.1: Pengujian Kualitas Informasi

Tahapan Pengujian Kualitas Informasi	Kajian
1. Pengaruh jenis kolom terhadap information loss	
• Waktu anonimisasi	numerik,kategorikal,campuran
• Waktu pengelompokan	numerik,kategorikal,campuran
• Total Information Loss	numerik,kategorikal,campuran
2. Pengaruh jumlah quasi-identifier terhadap information loss	
• Waktu anonimisasi	1-QID,2-QID,3-QID
• Waktu pengelompokan	1-QID,2-QID,3-QID
• Total Information Loss	1-QID,2-QID,3-QID
3. Pengaruh ukuran data terhadap information loss	
• Waktu anonimisasi	10k,30k
• Waktu pengelompokan	10k,30k
• Total Information Loss	10k,30k

Tabel 5.2: Pengujian Hasil Data Mining

Tahapan Pengujian Data Mining	Kajian
1. Pencarian model k-means terbaik untuk pengelompokan	
• Silhouette score	sebelum, setelah anonimisasi
• Waktu komputasi	sebelum, setelah anonimisasi
• Perbedaan hasil clustering (%)	(k=25,n=1000),(k=100,n=1000)
• Perbedaan hasil clustering (%)	(k=100,n=100),(k=100,n=1000)
2. Pencarian model naive bayes terbaik untuk klasifikasi	
• Tingkat akurasi	sebelum, setelah anonimisasi
• Waktu komputasi	sebelum, setelah anonimisasi
• Perbedaan hasil klasifikasi (%)	(n=100),(n1000)

Berikut adalah konfigurasi yang dipakai pada pengujian total information loss:

Tabel 5.3: Konfigurasi Pengujian

#	Eksperimen	Parameter	Dataset
1	column	n = 100,  QIDs  = 2, column ∈ numerik, kategorikal, campuran, k-value ∈ 25, 50, 75, 100	Credit score
2	QIDs	n = 100,  QIDs  ∈ [1..5], k-value ∈ 25, 50, 75, 100	Credit score
4	size	n ∈ 10k, 50k,  QIDs  = 5, k-value ∈ 25, 50, 75, 100	Credit score

Keterangan:

- k-value: nilai k pada greedy k-member cluster dan k-anonymity
- column: jenis kolom (numerik, kategorikal, numerik & kategorikal)
- |QIDs|: jumlah atribut quasi-identifier.
- size: jumlah data yang dipakai.

Berikut adalah konfigurasi yang dipakai pada pengujian hasil data mining:

Tabel 5.4: Konfigurasi Pengujian

#	Eksperimen	Parameter	Dataset
1	k-means	n ∈ 100, 1k, k = 25,100, k-value ∈ 100, features ∈ age, race, sex, workclass	Credit score sebelum dan setelah anonimisasi
2	naive bayes	n ∈ 100, 1k, label = income, trainset = 0.7, testset = 0.3, k-value ∈ 100, features ∈ age, race, sex, workclass	Credit score sebelum dan setelah anonimisasi

Keterangan:

- k: nilai k pada model k-means.
- label: nilai prediksi pada model naive bayes.
- k-value: nilai k pada greedy k-member cluster dan k-anonymity
- k-means: model pengelompokan data, menggunakan silhouette score.
- naive bayes: model klasifikasi data, menggunakan akurasi.

### Pengujian Eksperimental Total Information Loss

Tabel 5.5,5.6,5.7 adalah hasil setiap jenis pengujian eksperimental terhadap total information loss:

#### Jenis kolom bervariasi

Berikut adalah contoh sampel pengujian berdasarkan pemilihan kolom numerik, kategorikal, kategorikal & numerik dari dataset Credit score.

Tabel 5.5: Sampel Data Credit Score(Numerik)

DAY_S_BIRTH	DAY_S_EMPLOYED
-12005	-4542
-22464	365243
-19110	-3051

Tabel 5.6: Sampel Data Credit Score(Kategorikal)

OCCUPATION	GENDER
Sales staff	M
Security staff	F
Security staff	M

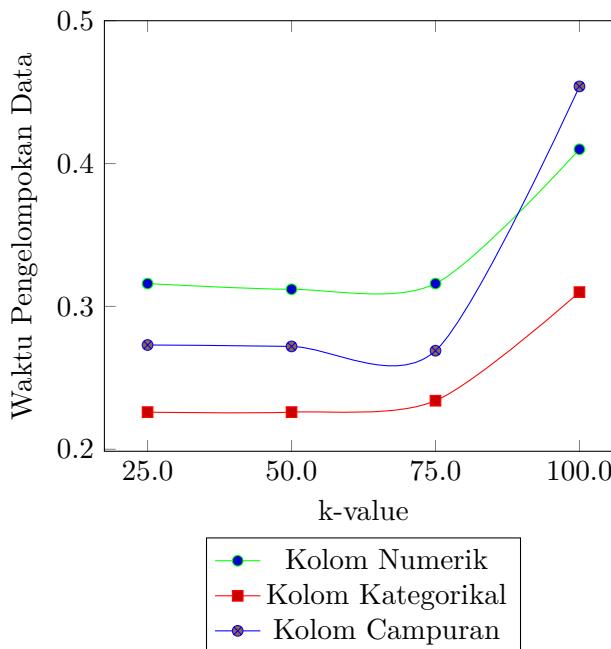
Tabel 5.7: Sampel Data Credit Score(Campuran)

DAY_S_BIRTH	OCCUPATION
-12005	Sales staff
-22464	Security staff
-19110	Security staff

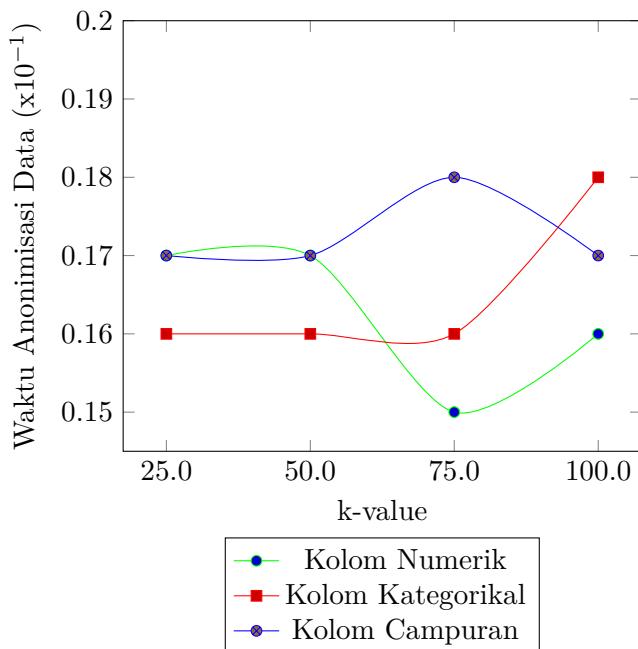
Pada pengujian ini, ingin dibuktikan apakah kualitas hasil pengelompokan greedy k-member clustering dan anonimisasi data k-anonymity dapat diterima jika jenis kolom bervariasi. Pengujian ini menggunakan  $|QIDs| \in [2]$  dengan pengujian pada 2 atribut numerik, 2 atribut kategorikal, dan 2 atribut campuran (numerik dan kategorikal),  $k \in [25,50,75,100]$ , dan  $n = 100$ . Berikut penjelasan jenis kajian yang diuji terhadap jumlah QID bervariasi:

- **Clusterization Time.** Pengamatan ini dilakukan pada algoritma greedy k-member clustering. Gambar 5.41 menunjukkan waktu pengelompokan data mengalami peningkatan signifikan pada bobot k-value = 100. Selain itu, kolom kategorikal menempati waktu pengelompokan data tercepat dibandingkan kolom campuran dan numerik. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa pengelompokan data dapat dilakukan lebih cepat jika menggunakan kolom kategorikal dengan pembobotan k-value kurang dari 100.
- **Anonymization Time.** Pengamatan ini dilakukan pada algoritma k-anonymity. Gambar 5.42 menunjukkan waktu anonimisasi yang hampir mirip untuk setiap jenis kolom. Hal ini dapat dilihat dari rentang waktu anonimisasi yang cukup sempit (0.15-0.18). Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa setiap kolom tidak memberikan pengaruh signifikan terhadap penambahan waktu anonimisasi. Proses anonimisasi juga memiliki waktu komputasi yang jauh lebih singkat dibandingkan pengelompokan data.

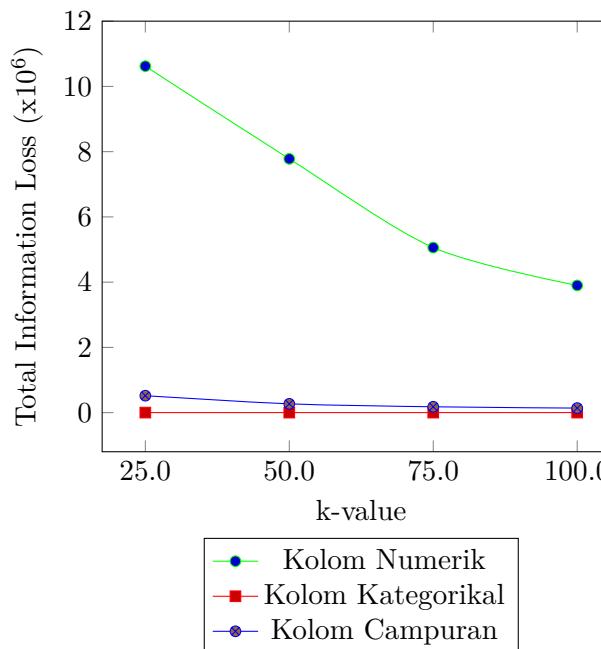
- **Total Information Loss.** Pengamatan ini dilakukan pada algoritma greedy k-member clustering. Gambar 5.43 menunjukkan total information loss yang dihasilkan oleh kolom numerik lebih besar dibandingkan kolom campuran dan kategorikal. Selain itu, total information loss mengalami penurunan seiring bertambahnya bobot k-value. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa pengelompokan data dapat menghasilkan kualitas pengelompokan yang lebih baik jika total information loss yang dihasilkan kecil dengan menggunakan kolom campuran dan pembobotan k-value yang lebih besar (k-value = 100).



Gambar 5.41: Perubahan Waktu Pengelompokan Data



Gambar 5.42: Perubahan Waktu Anonimisasi Data



Gambar 5.43: Perubahan Total Information Loss

### Kesimpulan:

Berdasarkan pengujian jumlah kolom bervariasi diatas, diketahui bahwa hasil pengelompokan dengan algoritma greedy k-member clustering cukup baik jika menggunakan kolom campuran dengan pembobotan k-value = 100, karena memiliki total information paling kecil dari pengujian lainnya. Dari segi kecepatan komputasi, waktu pengelompokan data jauh lebih lama dibandingkan waktu anonimisasi karena beberapa pekerjaan pengelompokan data pada algoritma greedy k-member clustering tidak dapat dipecah secara paralel.

### Jumlah QID bervariasi

Tabel 5.8, 5.9, 5.10 adalah contoh sampel pengujian berdasarkan pemilihan kolom numerik, kategorikal, kategorikal & numerik dari dataset Credit score.

Tabel 5.8: Sampel Data Credit score ( $|QID|=2$ )

DAY_S_BIRTH	OCCUPATION
-12005	Security staff
-22464	Sales staff
-19110	Sales staff

Tabel 5.9: Sampel Data Credit score ( $|QID|=3$ )

DAY_S_BIRTH	OCCUPATION	GENDER
-12005	Security staff	M
-22464	Sales staff	F
-19110	Sales staff	F

Tabel 5.10: Sampel Data Credit score ( $|QID|=4$ )

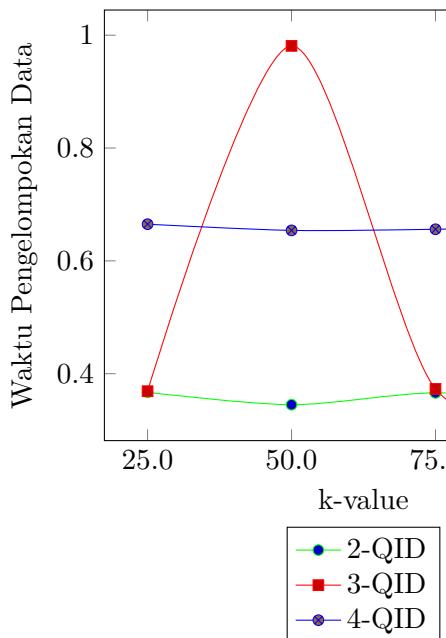
DAY_S_BIRTH	DAY_S_EMPLOYED	OCCUPATION	GENDER
-12005	-4542	Security staff	M
-22464	-4542	Sales staff	F
-19110	-769	Sales staff	F

Pada pengujian ini, ingin dibuktikan apakah kualitas pengelompokan dan anonimisasi data dari algoritma greedy k-member clustering dan k-anonymity dapat diterima jika jumlah quasi-identifier bervariasi . Pengujian ini menggunakan  $|QIDs| \in [2..4]$  dengan pengambilan 2 atribut numerik dan 2 atribut kategorikal,  $k \in [25,50,75,100]$  terhadap  $n = 100$ . Berikut penjelasan jenis kajian yang diuji:

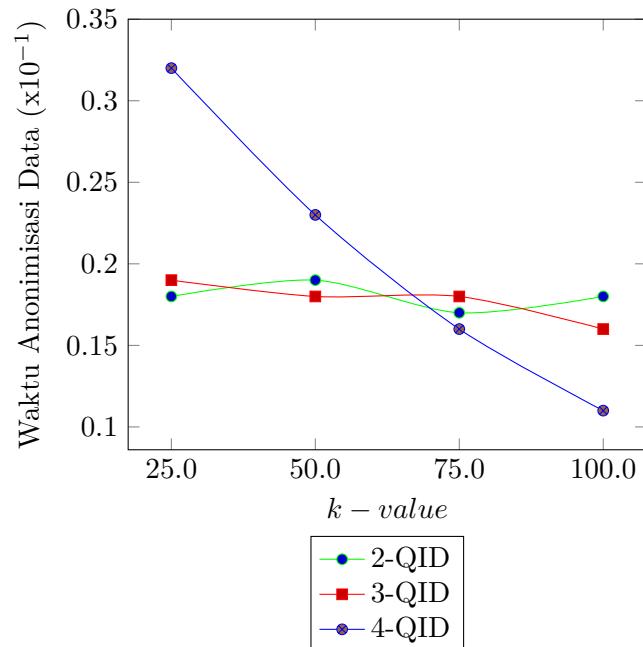
- **Clusterization Time.** Pengamatan ini dilakukan pada algoritma greedy k-member clustering. Gambar 5.45 menunjukkan waktu pengelompokan data mengalami peningkatan signifikan pada 3-QID dengan bobot k-value = 50. Selain itu, kolom 2-QID menempati waktu pengelompokan data tercepat dibandingkan 3-QID dan 4-QID. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa pengelompokan data dapat dilakukan lebih cepat jika menggunakan sampel data dengan 2 quasi-identifier (2-QID) untuk setiap pembobotan k-value.
- **Anonymization Time.** Pengamatan ini dilakukan pada algoritma k-anonymity. Gambar 5.45 menunjukkan waktu anonimisasi lebih lama pada penggunaan 4-QID. Hal ini dapat dilihat dari rentang waktu anonimisasi yang cukup besar antara 4-QID dengan 2-QID, 3-QID. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa dengan bertambahnya kolom

numerik pada sampel data dengan 4 quasi-idntifier (4-QID) memberikan pengaruh signifikan terhadap penambahan waktu anonimisasi. Proses anonimisasi juga memiliki waktu komputasi yang jauh lebih singkat dibandingkan pengelompokan data.

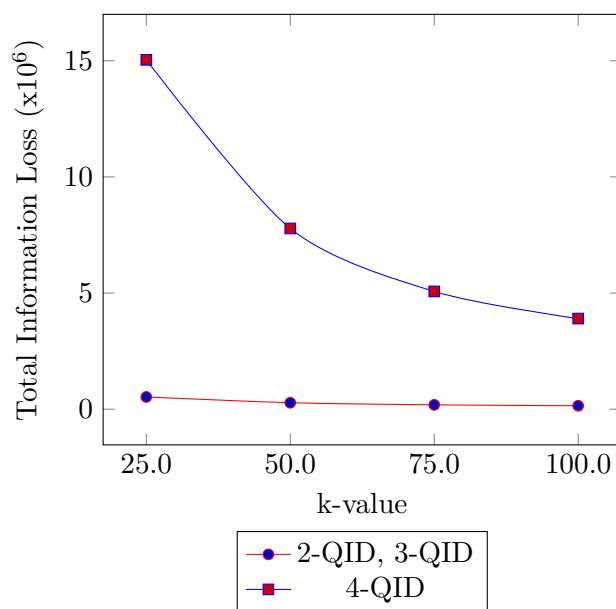
- **Total Information Loss.** Pengamatan ini dilakukan pada algoritma greedy k-member clustering. Gambar 5.46 menunjukkan total information loss yang dihasilkan oleh 4-QID jauh lebih besar dibandingkan 2-QID dan 3-QID. Selain itu, total information loss mengalami penurunan seiring bertambahnya bobot k-value. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa pengelompokan data dapat menghasilkan kualitas pengelompokan yang lebih baik jika total information loss yang dihasilkan kecil dengan menggunakan 2-QID, 3-QID dengan jumlah kolom numerik = 1 dan pembobotan k-value = 100.



Gambar 5.44: Perubahan Waktu Pengelompokan Data



Gambar 5.45: Perubahan Waktu Anonimisasi Data



Gambar 5.46: Perubahan Total Information Loss

### Kesimpulan:

Berdasarkan pengujian jumlah QID bervariasi diatas, diketahui bahwa hasil pengelompokan dengan algoritma greedy k-member clustering cukup baik jika menggunakan 2-QID/3-QID dengan pembobotan k-value = 100, karena memiliki total information loss paling kecil dari pengujian lainnya. Dari segi kecepatan komputasi, waktu pengelompokan data jauh lebih lama dibandingkan waktu anonimisasi karena beberapa pekerjaan pengelompokan data pada algoritma greedy k-member clustering tidak dapat dikerjakan secara paralel.

### Ukuran data bervariasi

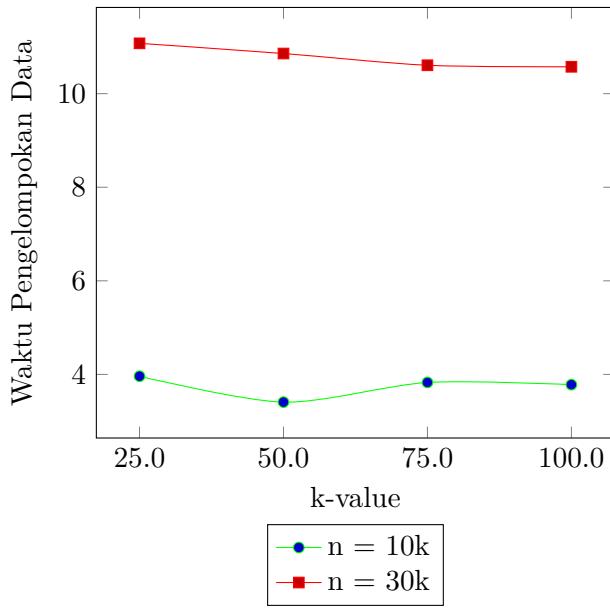
Tabel 5.11 adalah sampel pengujian berdasarkan pemilihan kolom numerik, kategorikal, campuran dari dataset Credit score, dimana QID adalah quasi-identifier dan SA adalah sensitive attribute.

Tabel 5.11: Sampel Data Credit score (10k dan 30k)

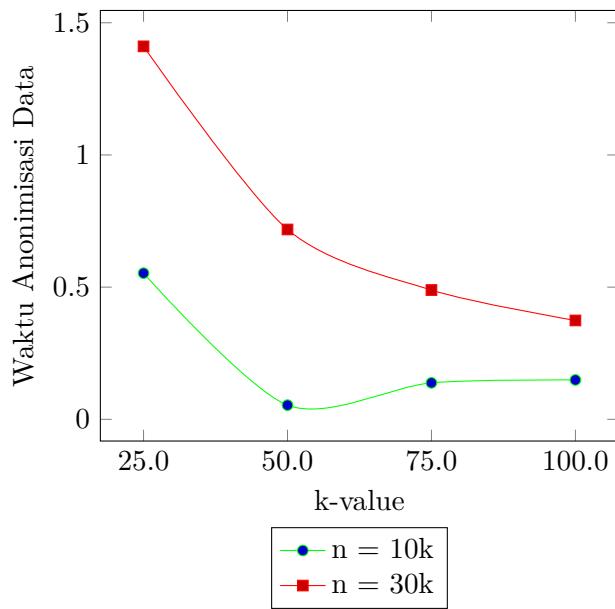
Nama Atribut	Tipe Data	Keterangan
DAYS_BIRTH	numerik	QID
DAYS_EMPLOYED	numerik	QID
OCCUPATION_TYPE	kategorikal	QID
CODE_GENDER	kategorikal	QID
NAME_INCOME_TYPE	kategorikal	SA
NAME_EDUCATION_TYPE	kategorikal	QID
NAME_FAMILY_STATUS	kategorikal	QID
NAME_HOUSING_TYPE	kategorikal	QID

Pada pengujian ini, ingin dibuktikan apakah kualitas pengelompokan dan anonimisasi data dari algoritma greedy k-member clustering dan k-anonymity dapat diterima jika ukuran data bervariasi . Pengujian ini menggunakan  $k \in [25,50,75,100]$  dengan pengambilan atribut secara acak terhadap  $n = 10k$  dan  $n = 30k$ . Berikut penjelasan jenis kajian yang diuji:

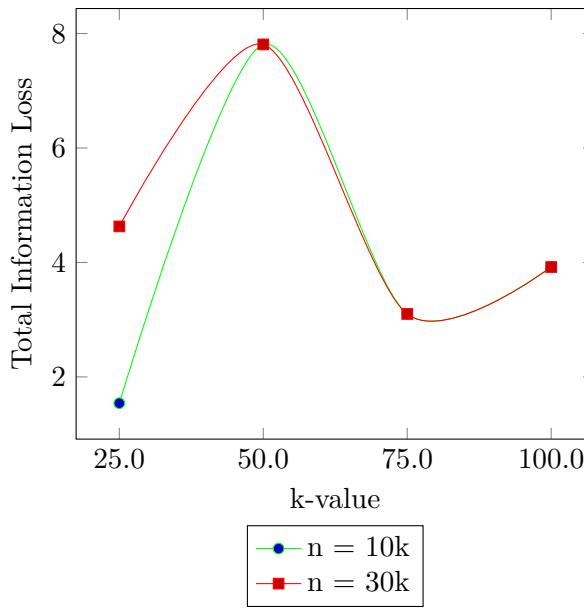
- **Clusterization Time.** Pengamatan ini dilakukan pada algoritma greedy k-member clustering. Gambar 5.47 menunjukkan waktu pengelompokan data dengan  $n = 30k$  membutuhkan waktu dua kali lebih lama dibandingkan pengelompokan data dengan  $n = 10k$ , dengan total waktu pengelompokan data sekitar 10 jam. Selain itu, waktu pengelompokan data juga cenderung stabil seiring bertambahnya nilai k-value. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa pengelompokan data dapat dilakukan lebih cepat jika menggunakan ukuran data yang relatif lebih kecil, yaitu  $n = 10k$  untuk setiap pembobotan k-value.
- **Anonymization Time.** Pengamatan ini dilakukan pada algoritma k-anonymity. Gambar 5.48 menunjukkan proses anonimisasi dilakukan lebih lama pada sampel data  $n = 10k$ , dengan total waktu pengelompokan data sekitar 1.5 jam. Selain itu, k-value = 25 menempati posisi waktu anonimisasi data lebih tinggi dibandingkan dengan nilai k-value lainnya. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa proses anonimisasi data dapat dilakukan lebih cepat jika menggunakan sampel ukuran data yang relatif kecil, yaitu  $n = 10k$ .
- **Total Information Loss.** Pengamatan ini dilakukan pada algoritma greedy k-member clustering. Gambar 5.49 menunjukkan total information loss yang dihasilkan oleh  $n = 10k$  dan  $n = 30k$  hampir sama. Perbedaan signifikan total information loss hanya ditunjukkan pada k-value = 25, dimana  $n = 10k$  menempati total information loss terkecil. Melalui hasil pengujian ini, dapat ditarik kesimpulan bahwa pengelompokan data dapat menghasilkan kualitas pengelompokan yang lebih baik jika total information loss yang dihasilkan kecil dengan menggunakan  $n = 10k$  maupun  $n = 30k$  untuk setiap pemilihan nilai k-value.



Gambar 5.47: Perubahan Waktu Pengelompokan Data



Gambar 5.48: Perubahan Waktu Anonimisasi Data



Gambar 5.49: Perubahan Total Information Loss

### Kesimpulan:

Berdasarkan pengujian jumlah QID bervariasi diatas, diketahui bahwa hasil pengelompokan dengan algoritma greedy k-member clustering cukup baik jika menggunakan ukuran data apapun dengan nilai k-value = 25, karena memiliki total information loss paling kecil dari pengujian lainnya. Dari segi kecepatan komputasi, waktu pengelompokan data jauh lebih lama dibandingkan waktu anonimisasi karena beberapa pekerjaan pengelompokan data pada algoritma greedy k-member clustering tidak dapat dikerjakan secara paralel. Hal penting yang perlu diketahui adalah komputasi pengelompokan data dengan  $n > 10k$  membutuhkan waktu lebih dari 3 jam sehingga diperlukan pertimbangan untuk menerapkan algortima greedy k-member clustering untuk lingkungan big data.

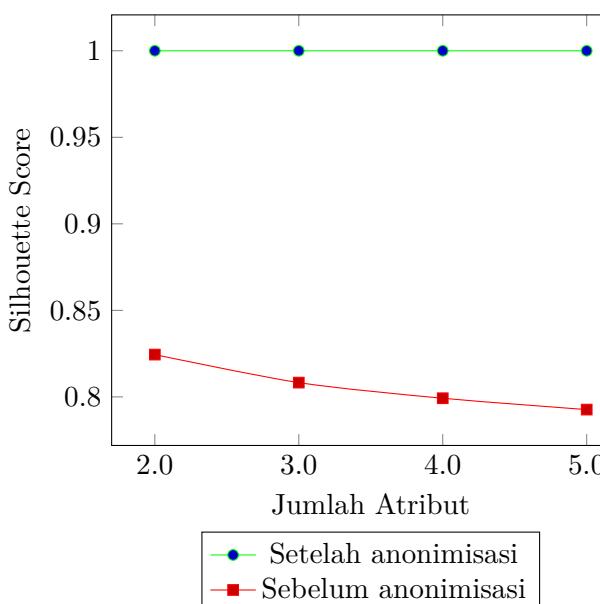
## Pengujian Eksperimental Hasil Data Mining

Berikut adalah hasil setiap jenis pengujian eksperimental terhadap total informasi yang hilang:

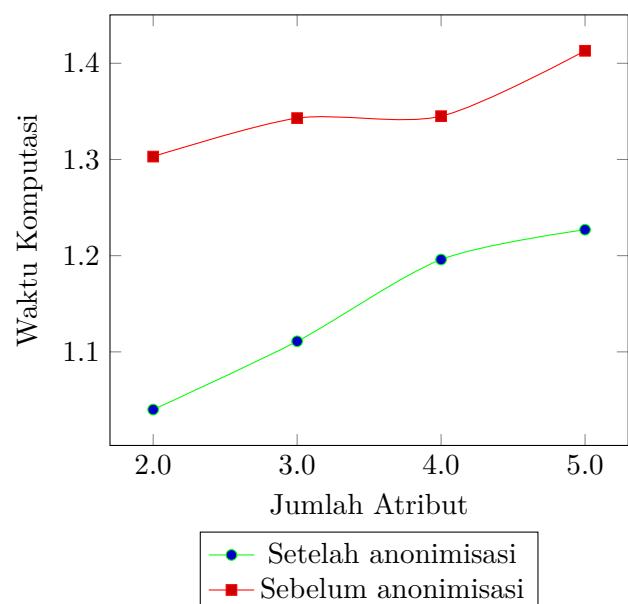
### K-means

Pada pengujian ini, ingin dibuktikan apakah algoritma greedy k-member clustering dan k-anonymity dapat diterima jika hasil anonimisasinya digunakan untuk clustering/pengelompokan data. Pengujian ini menggunakan  $|QIDs| \in [1..5]$  dengan pengambilan atribut secara acak terhadap  $n = 1000$  dan  $k\text{-value} = 100$ , karena pada pengujian sebelumnya, kondisi ini memiliki information loss paling kecil. Berikut penjelasan jenis kajian yang diuji:

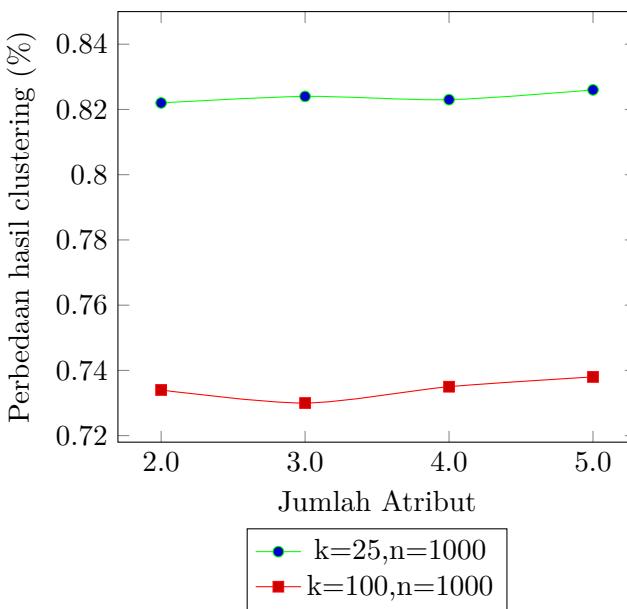
- **Silhouette score.** Pengamatan ini dilakukan dengan pemodelan k-means pada library Spark MLlib. Gambar 5.50 menunjukkan silhouette score setelah anonimisasi stabil pada nilai 1. Perilaku ini disebabkan karena hasil anonimisasi yang pada cluster yang sejenis bernilai sama dan tidak dapat dibedakan satu sama lain. Hasil pengujian ini memberikan kesimpulan bahwa hasil pengelompokan setelah dilakukan anonimisasi lebih baik dibandingkan sebelum dilakukan anonimisasi, karena memiliki silhouette score yang lebih tinggi.
- **Waktu komputasi.** Pengamatan ini dilakukan dengan komputer lokal (standalone). Gambar 5.51 menunjukkan waktu komputasi tercepat untuk pembuatan model k-means didapat oleh data yang telah dianonimisasi. Perilaku ini disebabkan karena data yang telah dianonimisasi umumnya memiliki lebih sedikit jumlah variasi nilai, sehingga dapat mempercepat klasifikasi data. Hasil pengujian ini memberikan kesimpulan bahwa jumlah variasi nilai dapat memberi pengaruh terhadap waktu komputasi.
- **Perbedaan hasil clustering (%)**. Pengamatan ini menghitung persentase perbedaan hasil clustering saat sebelum dan setelah data dianonimisasi pada kondisi jumlah data dan  $k$  yang beragam. Gambar 5.52 menunjukkan persentase perbedaan hasil klasifikasi tertinggi berdasarkan nilai  $k$  diraih oleh  $k = 25, n = 1000$ . Gambar 5.53 menunjukkan persentase perbedaan hasil klasifikasi tertinggi berdasarkan nilai  $k$  diraih oleh  $k = 10, n = 1000$ . Hasil dari pengujian ini memberikan kesimpulan bahwa nilai  $k$  yang semakin kecil dan ukuran data semakin besar dapat menyebabkan tingginya perbedaan hasil clustering.



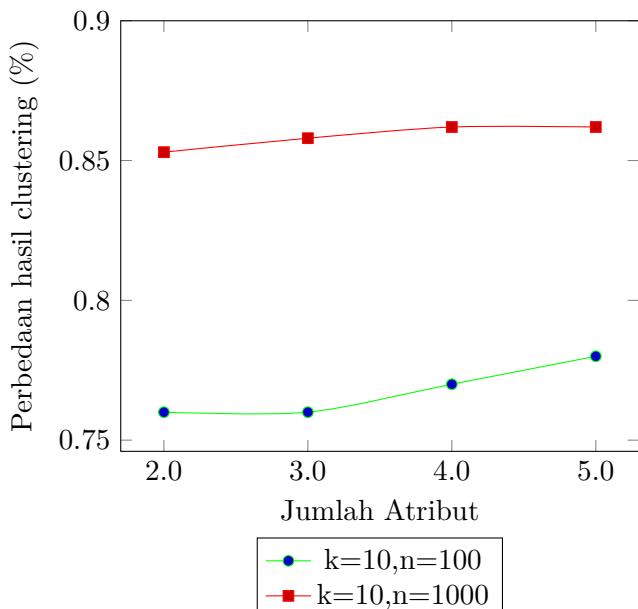
Gambar 5.50: Perbandingan Silhouette Score Model K-Means



Gambar 5.51: Perbandingan Waktu Komputasi Model K-Means



Gambar 5.52: Perbandingan Hasil Clustering terhadap k



Gambar 5.53: Perbandingan Hasil Clustering terhadap n

### Kesimpulan:

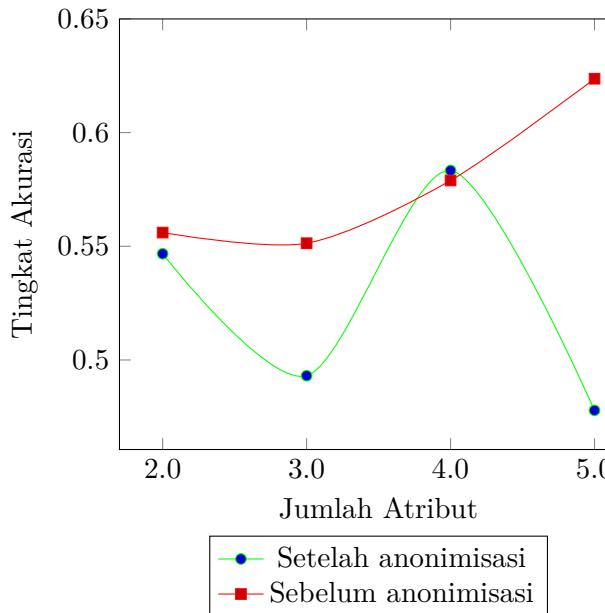
Pada percobaan ini dapat disimpulkan 3 hal penting. Pertama, penggunaan algoritma greedy k-member clustering dan k-anonymity pada data credit score dapat diterima, karena hasil pengelompokannya sudah baik dibuktikan dengan hasil anonimisasi memiliki silhouette score bernilai 1. Kedua, penggunaan model k-means pada library Spark MLlib dapat melakukan komputasi dengan cepat untuk ukuran 1000 data. Ketiga, perbedaan hasil clustering paling minimal dapat dicapai dengan memperkecil ukuran data dan memperbesar nilai k

### Naive bayes

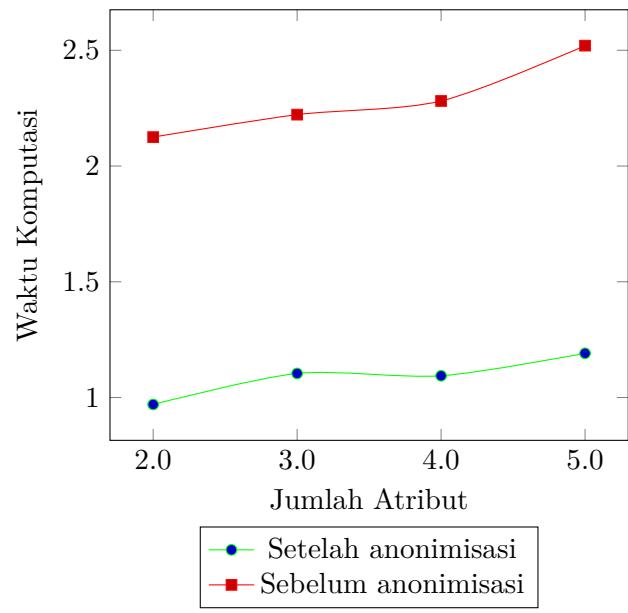
Pada pengujian ini, ingin dibuktikan apakah algoritma greedy k-member clustering dan k-anonymity dapat diterima jika hasil anonimisasinya digunakan untuk klasifikasi data. Pengujian ini menggunakan  $|QIDs| \in [1..5]$  dengan pengambilan atribut secara acak terhadap  $n = 1000$  dan  $k\text{-value} = 100$ , karena pada pengujian sebelumnya, kondisi ini memiliki information loss paling kecil. Berikut penjelasan jenis kajian yang diuji:

- **Tingkat akurasi.** Pengamatan ini dilakukan dengan pemodelan naive bayes pada library Spark MLlib. Gambar 5.54 menunjukkan perbedaan akurasi yang cukup tinggi (pada kondisi jumlah atribut berjumlah ganjil) dan perbedaan akurasi yang cukup dekat (pada kondisi jumlah atribut berjumlah genap). Perilaku ini disebabkan karena penyisipan jenis atribut bertipe numerik pada data dengan jumlah atribut genap. Hasil pengujian ini memberikan kesimpulan bahwa untuk mendapatkan tingkat akurasi yang baik, data yang digunakan perlu lebih banyak mengandung atribut numerik.
- **Waktu komputasi.** Pengamatan ini dilakukan dengan komputer lokal (standalone). Gambar 5.55 menunjukkan waktu komputasi tercepat untuk pembuatan model naive bayes diraih oleh data yang telah dianonimisasi. Perilaku ini disebabkan karena data yang telah dianonimisasi umumnya memiliki lebih sedikit jumlah variasi nilai, sehingga dapat mempercepat klasifikasi data. Hasil pengujian ini memberikan kesimpulan bahwa jumlah variasi nilai dapat memberi pengaruh terhadap waktu komputasi.

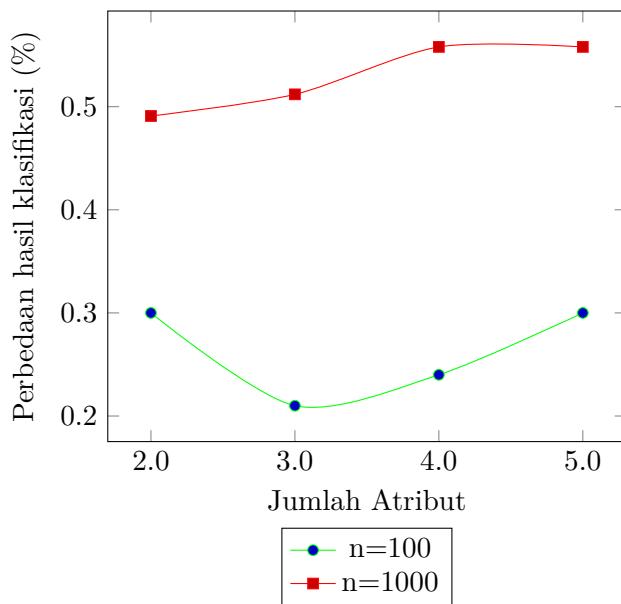
- **Perbedaan hasil klasifikasi (%)**. Pengamatan ini menghitung persentase perbedaan hasil klasifikasi saat sebelum dan setelah data dianonimisasi pada kondisi jumlah data yang beragam. Gambar 5.56 menunjukkan persentase perbedaan hasil klasifikasi tertinggi diraih oleh  $n = 1000$ . Hasil dari pengujian ini memberikan kesimpulan bahwa jumlah data dapat mempengaruhi persentase perbedaan hasil klasifikasi, sehingga untuk mendapatkan perbedaan hasil klasifikasi yang minimal perlu mengurangi jumlah data yang diuji.



Gambar 5.54: Perbandingan Tingkat Akurasi Model Naive Bayes



Gambar 5.55: Perbandingan Waktu Komputasi Model Naive Bayes



Gambar 5.56: Perbedaan Hasil Klasifikasi terhadap Jumlah Data

### Kesimpulan:

Pada percobaan ini dapat disimpulkan 3 hal penting. Pertama, penggunaan algoritma greedy k-member clustering dan k-anonymity pada data credit score dapat diterima, karena pada kasus tertentu tingkat akurasi sebelum dan setelah anonimisasi memiliki perbedaan cukup dekat. Kedua, penggunaan model naive bayes pada library Spark MLlib dapat melakukan komputasi dengan cepat untuk ukuran 1000 data. Ketiga, perbedaan hasil klasifikasi paling minimal dapat dicapai dengan memperkecil ukuran data pengujian.

### Kesimpulan Pengujian Eksperimental

Beberapa hal penting yang perlu dipertimbangkan untuk penelitian selanjutnya. Pertama, pengelompokan data dengan greedy k-member clustering sangat lama sehingga dapat diganti ke library KMeans milik Spark MLlib. Kedua, total information loss dapat diminimalisir dengan penggunaan nilai k-value yang cukup besar dan jumlah quasi-identifier secukupnya. Ketiga, metode data mining terbaik untuk hasil anonimisasi adalah klasifikasi karena menghasilkan perbedaan prediksi paling sedikit. Hasil pengujian dapat dilihat pada Tabel 5.12.

Tabel 5.12: Kesimpulan Pengujian Eksperimental

Hasil Pengujian	Kajian	Parameter Terbaik	Kesimpulan
Total Information Loss	column	kolom campuran, k-value = 100	Parameter ini dipilih karena memiliki total information loss paling kecil yaitu kurang dari $(2 \times 10^6)$ untuk 1000 data .
	$ QIDs $	2-QID/3-QID, 1 kolom numerik, 2 kolom kategori, k-value = 100	Parameter ini dipilih karena memiliki total information loss paling kecil yaitu kurang dari $(1 \times 10^6)$ untuk 1000 data.
	size	7-QID, 2 kolom numerik, 5 kolom kategori, k-value = 25	Parameter ini dipilih karena memiliki total information loss paling kecil yaitu kurang dari $(4 \times 10^7)$ untuk 10.000 data.
Hasil Data Mining	k-means	k = 100, n = 100	Parameter ini dipilih karena memiliki perbedaan hasil clustering terendah (0.74) terhadap data sebelum dan setelah anonimisasi.
	naive bayes	n = 100	Parameter ini dipilih karena memiliki perbedaan hasil klasifikasi terendah (0.30) terhadap data sebelum dan setelah anonimisasi

## **BAB 6**

### **KESIMPULAN DAN SARAN**

Pada bab ini akan dijelaskan kesimpulan dari awal hingga akhir penelitian beserta saran untuk penelitian selanjutnya.

#### **6.0.1 Kesimpulan**

Kesimpulan yang dapat diambil dari penelitian ini adalah sebagai berikut:

- Pada penelitian ini, telah dipelajari cara kerja algoritma pengelompokan data yaitu Greedy k-member clustering. Pengelompokan data perlu dilakukan untuk meminimalkan informasi yang hilang saat anonimisasi data. Secara singkat tahapan dari algoritma ini adalah mengambil sebuah data secara acak, mencari  $k-1$  data lainnya yang dekat dengan data acak tersebut, dan terakhir jika terdapat sisa data yang belum dikelompokan maka sisa data tersebut digabungkan pada cluster yang paling dekat.
- Perangkat lunak eksplorasi dan anonimisasi dibuat untuk memenuhi pengamatan terhadap anonimisasi data. Kedua perangkat lunak ini berhasil dibuat dengan menggunakan library Spark, antara lain Spark Core, Spark SQL. Masing-masing perangkat lunak hanya menerima sebuah file JSON dengan format yang berbeda satu sama lain. Masing-masing perangkat lunak dapat mengeluarkan output ke dalam format CSV pada lokasi folder yang telah dicantumkan pada file JSON.
- Perangkat lunak pengujian dibuat untuk memenuhi pengamatan terhadap pengujian fungsional dan eksperimental. Perangkat lunak ini berhasil dibuat menggunakan library Spark, yaitu Spark MLLib (library KMeans dan NaiveBayes). Perangkat lunak menerima input JSON yang dapat diisi sesuai jenis pengujian data mining (mengisi parameter untuk pengujian k-means/naive bayes). Perangkat lunak dapat mengeluarkan output dalam format CSV pada lokasi yang telah dicantumkan pada file JSON.
- Pada pengujian fungsional, telah dibandingkan hasil anonimisasi pada perangkat lunak anonimisasi dengan hasil perhitungan manual. Melalui pengamatan sebelumnya, telah diketahui bahwa pengujian memiliki hasil anonimisasi yang sama sehingga fungsionalitas perangkat lunak anonimisasi dapat dinyatakan sudah benar.
- Pengujian eksperimental berfungsi untuk mencari bukti apakah hasil anonimisasi dapat memiliki hasil data mining yang baik. Pada pengujian eksperimental yang dilakukan sebelumnya, telah dilakukan pengamatan terhadap kualitas hasil anonimisasi data berdasarkan total information loss dan kualitas hasil data mining sebelum dan setelah anonimisasi dengan metode k-means dan naive bayes.
- Terkait pengujian kualitas hasil anonimisasi, telah diketahui bahwa hasil terbaik diperoleh pemilihan  $k$ -value yang lebih besar, penggunaan kolom campuran dengan tidak memakai terlalu banyak kolom numerik, penggunaan jumlah quasi-identifier yang lebih sedikit, dan penggunaan ukuran data yang lebih kecil. Karena parameter ini memiliki total information

loss paling kecil dibandingkan dengan parameter lainnya, maka dapat dinyatakan bahwa kualitas anonimisasi data sudah baik untuk parameter tersebut.

- Terkait pengujian kualitas hasil data mining berdasarkan persentasi prediksi, telah diketahui bahwa hasil terbaik diperoleh dengan metode klasifikasi karena memiliki perbedaan persentase prediksi yang lebih kecil antara data sebelum dan setelah dilakukan anonimisasi. Hasil klasifikasi memiliki perbedaan persentase prediksi dibawah 0.5, sehingga dapat disimpulkan bahwa proses anonimisasi data dengan dilakukan pengelompokan greedy k-member clustering terlebih dahulu memiliki prediksi yang hampir sama dengan data sesungguhnya, yang berarti kualitas hasil data mining pada data yang telah dianonimisasi sudah baik untuk mendapatkan informasi.
- Terkait pengujian kualitas hasil data mining berdasarkan silhouette score (clustering) dan tingkat akurasi (klasifikasi), telah diketahui 2 jenis informasi penting. Pertama, silhouette score tertinggi diperoleh pada data yang telah dianonimisasi karena data-data tersebut memiliki nilai inter cluster yang sangat mirip dan intra cluster yang sangat berbeda. Hal ini yang membuat pengelompokan data yang telah dianonimisasi lebih baik dibandingkan data yang belum dianonimisasi. Kedua, tingkat akurasi tertinggi diperoleh pada data yang belum dianonimisasi karena data yang dianonimisasi banyak mengadung nilai yang mirip, sehingga kemungkinan besar prediksinya bisa salah akibat dari kurangnya data yang bersifat unik.
- Terkait kinerja algoritma k-anonymity dan greedy k-member clustering, telah diketahui bahwa waktu eksekusi algoritma greedy k-member clustering sangat lama walaupun telah diimplementasi pada framework Spark yang dapat membagi pekerjaan secara paralel. Hal ini disebabkan karena sebagian besar pekerjaan pada algoritma ini tidak dapat dilakukan proses paralel, contohnya mencari anggota sebuah cluster. Hal ini membuat pemrosesan hanya dapat dijalankan pada satu komputer saja. Berbanding terbalik dengan algoritma sebelumnya, algoritma k-anonymity memiliki waktu komputasi yang lebih cepat karena pekerjaannya dapat dilakukan paralel.

### 6.0.2 Saran

Saran untuk penelitian selanjutnya adalah sebagai berikut:

- Pada penelitian ini, diketahui bahwa algoritma greedy k-member clustering memiliki waktu komputasi yang sangat lama jika merancang algoritma pengelompokan secara mandiri. Solusi dari permasalahan pengelompokan data pada lingkungan big data adalah menggunakan library KMeans pada Spark MLlib agar waktu eksekusinya menjadi lebih efisien.
- Pada penelitian ini, pernah terjadi error java.lang.OutOfMemoryError terkait lazy evaluation pada Spark yang diterapkan pada algoritma yang iteratif. Dikutip dari Medium, masalah ini terjadi ketika fungsi transformation filter(),union() dipanggil pada setiap iterasi. Fungsi transformasi adalah fungsi dengan biaya komputasi yang mahal, terutama jika dipanggil beberapa kali dalam satu iterasi. Konsep lazy evaluation pada Spark mirip dengan konsep rekursif, dimana fungsi transformation akan dijalankan saat fungsi action dipanggil. Error ini terjadi ketika fungsi action dipanggil pada iterasi tertentu yang membuat fungsi transformasi pada iterasi sebelumnya juga ikut dijalankan. Solusi yang dapat diterima adalah menyimpan dan membaca hasil komputasi transformasi pada sistem penyimpanan HDFS.
- Struktur data pada pemrosesan big data perlu diperhatikan. Diusahakan untuk memilih struktur data Dataframe/RDD, karena hanya operasi tersebut yang dapat berjalan secara paralel. Selain itu diusahakan untuk tidak terlalu banyak menggunakan operasi looping. Pada kasus tertentu, operasi looping dapat diganti dengan operasi kueri SQL.

# LAMPIRAN A

## KODE PROGRAM

Listing A.1: MyCode.c

```

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

```

Listing A.2: MyCode.java

```

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

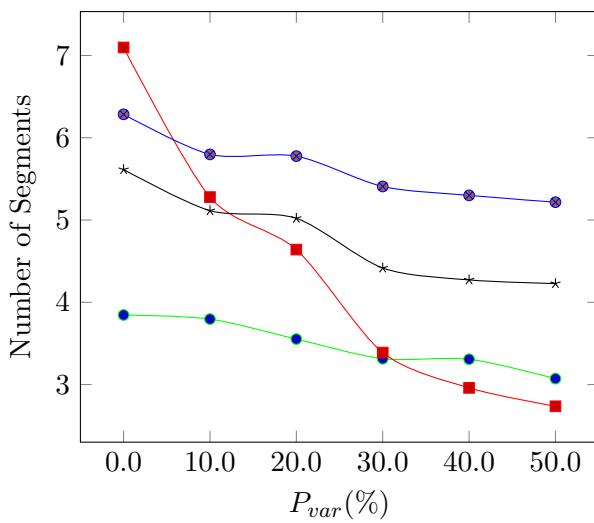
```



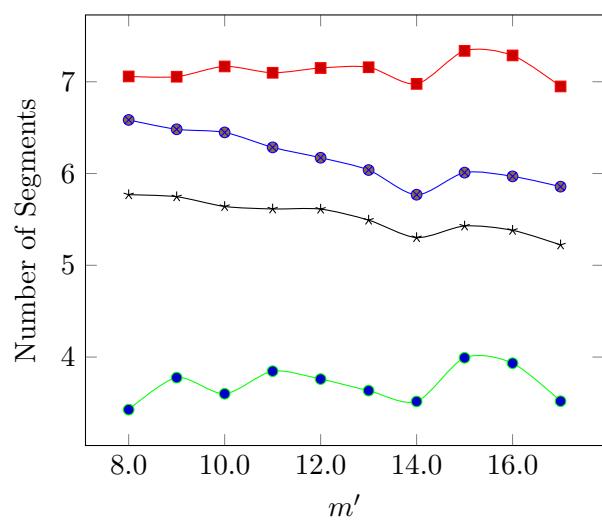
## LAMPIRAN B

### HASIL EKSPERIMENT

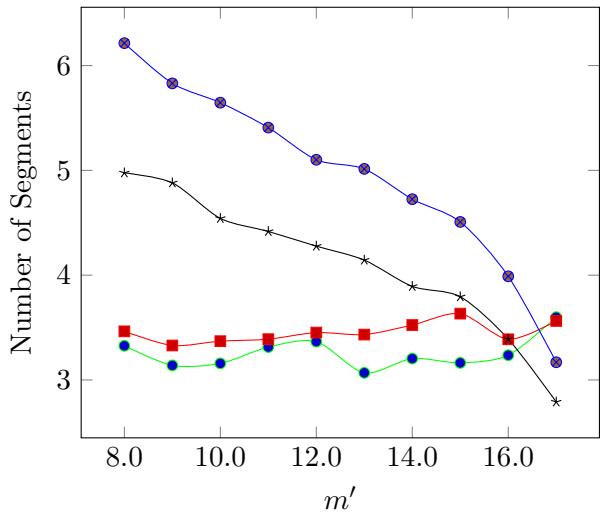
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



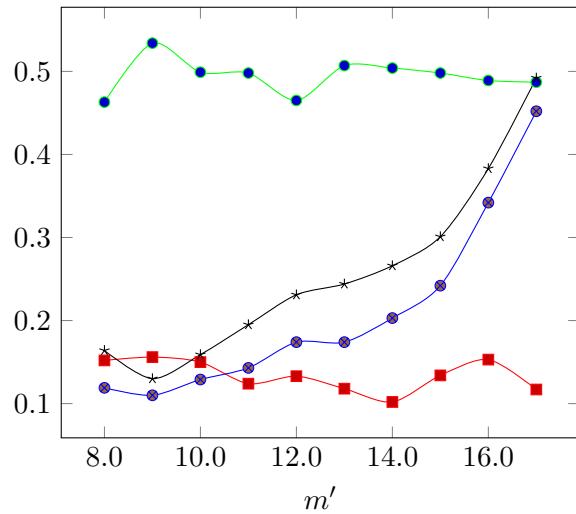
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4