

## SKRIPSI

**ANALISIS TEKNIK *RANDOM ROTATION PERTURBATION*  
DAN *RANDOM PROJECTION PERTURBATION* DALAM  
MENGACAK DATA UNTUK PENAMBANGAN DATA**



Chris Eldon

NPM: 2016730073

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



**UNDERGRADUATE THESIS**

**ANALYSIS OF RANDOM ROTATION PERTURBATION AND  
RANDOM PROJECTION PERTURBATION TECHNIQUES IN  
RANDOMIZING DATA FOR DATA MINING**



**Chris Eldon**

**NPM: 2016730073**

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



## ABSTRAK

Pada proses penambangan data, data yang digunakan seringkali diberikan kepada pihak lain dan ada kemungkinan privasi di dalam data tersebut tersebar kepada pihak yang tidak berhak. Data privasi tersebut dapat tersebar kepada pihak yang tidak bertanggung jawab dan disalahgunakan. Dalam menghindari hal tersebut, *privacy-preserving data mining* perlu dilakukan. Privasi dapat diartikan sebagai sebuah informasi personal seseorang yang dapat mengidentifikasi sesuatu hal pada orang tersebut. Salah satu cara untuk melakukan *privacy-preserving data mining* adalah mengacak data menggunakan metode *Randomization*. Metode *Randomization* bekerja dengan cara mengacak data tetapi data tersebut masih dapat digunakan untuk penambangan data. Pada penelitian ini akan dibangun sebuah perangkat lunak yang mengimplementasikan 2 buah teknik yang menggunakan metode *Randomization* yaitu teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*.

Pengujian akan dilakukan dengan menerapkan penambangan data klasifikasi dengan algoritma *k-nearest neighbors* dan penambangan data *clustering* dengan algoritma *k-means* masing-masing untuk menhitung akurasi model dan kemiripan hasil *cluster*. Berdasarkan hasil pengujian, model penambangan data yang dilatih dengan *dataset* asli dan *dataset* yang telah diacak dengan teknik *Random Rotation Perturbation* atau *Random Projection Perturbation* memiliki kualitas yang sama atau sangat mirip. Kedua teknik tersebut hanya dapat digunakan untuk data yang bersifat numerik dan khususnya untuk teknik *Random Projection Perturbation* hanya dapat digunakan untuk data yang memenuhi syarat teknik tersebut yaitu jumlah fitur pada data harus cukup banyak.

**Kata-kata kunci:** Privasi, PII, *privacy-preserving data mining*, *Randomization*, *Random Rotation Perturbation*, *Random Projection Perturbation*, penambangan data, klasifikasi, *clustering*, *k-nearest neighbors*, *k-means*



## ABSTRACT

In the data mining process, the data used is often given to other parties and there is the possibility of privacy in the data being spread to unauthorized parties. The privacy data can be spread to irresponsible and misused parties. To avoid this, privacy-preserving data mining needs to be done. Privacy can be interpreted as a person's personal information that can identify something about that person. One way to do privacy-preserving data mining is to randomize data using the Randomization method. The Randomization method works by randomizing data but the data can still be used for data mining. In this research, a software will be built that implements 2 techniques that use the Randomization method, namely the Random Rotation Perturbation technique and the Random Projection Perturbation.

The test will be carried out by applying classification data mining with the k-nearest neighbors algorithm and clustering data mining with the k-means algorithm respectively to calculate the accuracy of the model and the similarity of the cluster results. Based on the test results, the data mining model that is trained with the original dataset and the dataset that has been randomized with the Random Rotation Perturbation technique or Random Projection Perturbation has the same or very similar quality. Both of these techniques can only be used for numerical data and specifically for the Random Projection Perturbation technique can only be used for data that meet the technical requirement that is the number of features in the data must be quite a lot.

**Keywords:** Privacy, PII, privacy-preserving data mining, Randomization, Random Rotation Perturbation, Random Projection Perturbation, data mining, classification, clustering, k-nearest neighbors, k-means



# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>ix</b>
<b>DAFTAR GAMBAR</b>	<b>xi</b>
<b>DAFTAR TABEL</b>	<b>xiii</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan . . . . .	3
1.4 Batasan Masalah . . . . .	3
1.5 Metodologi . . . . .	3
1.6 Sistematika Pembahasan . . . . .	4
<b>2 DASAR TEORI</b>	<b>5</b>
2.1 Privasi Data . . . . .	5
2.1.1 Privasi . . . . .	5
2.1.2 <i>Personally Identifiable Information</i> . . . . .	6
2.2 Penambangan Data . . . . .	7
2.2.1 Klasifikasi . . . . .	7
2.2.2 <i>Clustering</i> . . . . .	8
2.3 <i>Privacy Preserving Data Mining</i> . . . . .	11
2.4 Metode <i>Randomization</i> . . . . .	12
2.4.1 <i>Random Noise Addition</i> . . . . .	12
2.4.2 <i>Random Rotation Perturbation</i> . . . . .	13
2.4.3 <i>Random Projection Perturbation</i> . . . . .	15
2.5 Bahasa Pemograman Python . . . . .	17
<b>3 ANALISIS</b>	<b>19</b>
3.1 Analisis Masalah . . . . .	19
3.1.1 Implementasi Metode <i>Randomization</i> . . . . .	19
3.1.2 Pengujian Dengan Penambangan Data . . . . .	22
3.2 Studi Kasus . . . . .	24
3.2.1 <i>Random Rotation Perturbation</i> . . . . .	24
3.2.2 <i>Random Projection Perturbation</i> . . . . .	27
3.3 Gambaran Umum Perangkat Lunak . . . . .	29
3.3.1 Diagram Aktivitas . . . . .	29
3.3.2 Diagram Kelas . . . . .	32
<b>4 PERANCANGAN PERANGKAT LUNAK</b>	<b>37</b>
4.1 Perancangan Antarmuka . . . . .	37
4.2 Perancangan Kelas . . . . .	39
4.2.1 Diagram <i>Package</i> . . . . .	41

4.2.2	Diagram Kelas pada <i>Package Perturbation</i>	42
4.2.3	Diagram Kelas pada <i>Package Matrix</i>	46
4.2.4	Diagram Kelas pada <i>Package Preprocessor</i>	50
4.3	Masukan Perangkat Lunak	53
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>55</b>
5.1	Implementasi Antarmuka	55
5.1.1	Masukan dan Pengaturan	55
5.1.2	Deskripsi <i>Dataset</i>	66
5.1.3	Deskripsi Hasil Pengacakan	67
5.2	Pengujian Fungsional	68
5.2.1	Teknik <i>Random Rotation Perturbation</i>	68
5.2.2	Teknik <i>Random Projection Perturbation</i>	72
5.3	Pengujian Eksperimental	75
5.3.1	Properti Data	76
5.3.2	Penambangan Data Klasifikasi	79
5.3.3	Penambangan Data <i>Clustering</i>	84
5.3.4	Kesimpulan Akhir	92
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>95</b>
6.1	Kesimpulan	95
6.2	Saran	96
<b>DAFTAR REFERENSI</b>		<b>99</b>
<b>A</b>	<b>KODE PERANGKAT LUNAK <i>Randomization</i></b>	<b>101</b>
A.1	<i>Package Perturbation</i>	101
A.1.1	Kelas <i>Perturbation</i>	101
A.1.2	Kelas <i>RandomRotationPerturbation</i>	101
A.1.3	Kelas <i>RandomProjectionPerturbation</i>	102
A.2	<i>Package Matrix</i>	102
A.2.1	Kelas <i>Matrix</i>	102
A.2.2	Kelas <i>RandomRotationMatrix</i>	103
A.2.3	Kelas <i>RandomTranslationMatrix</i>	103
A.2.4	Kelas <i>RandomProjectionMatrix</i>	103
A.3	<i>Package Preprocessor</i>	103
A.3.1	Kelas <i>CSVPreprocessor</i>	103
A.3.2	Kelas <i>ProjectionMatrixPreprocessor</i>	104
A.3.3	Kelas <i>RotationMatrixPreprocessor</i>	104
A.4	<i>Package View</i>	105
A.4.1	Dokumen <i>main_menu.py</i>	105
A.4.2	Dokumen <i>randomization_app.py</i>	110
A.4.3	Dokumen <i>randomization.kv</i>	111
<b>B</b>	<b>KODE PERANGKAT LUNAK PENGUJIAN</b>	<b>115</b>
B.1	Pengujian Penambangan Data Klasifikasi	115
B.1.1	<i>Random Rotation Perturbation</i>	115
B.1.2	<i>Random Projection Perturbation</i>	116
B.2	Pengujian Penambangan Data <i>Clustering</i>	118
B.2.1	<i>Random Rotation Perturbation</i>	118
B.2.2	<i>Random Projection Perturbation</i>	120
B.3	Pengujian Lainnya	121

## DAFTAR GAMBAR

1.1	Berbagai macam teknik modifikasi data untuk <i>privacy preserving data mining</i> . . . . .	2
2.1	Ilustrasi penjelasan teknik <i>Random Rotation Perturbation</i> pada bidang Euclidean . . . . .	14
3.1	Diagram aktivitas perangkat lunak <i>Randomization</i> . . . . .	30
3.2	Diagram kelas perangkat lunak <i>Randomization</i> . . . . .	33
4.1	Halaman saat memilih teknik <i>Random Rotation Perturbation</i> . . . . .	38
4.2	Halaman saat memilih teknik <i>Random Projection Perturbation</i> . . . . .	38
4.3	<i>Popup</i> yang ditampilkan apabila pengacakan sukses dilakukan . . . . .	39
4.4	<i>Popup</i> yang ditampilkan apabila pengacakan gagal dilakukan . . . . .	40
4.5	<i>Popup</i> yang akan ditampilkan apabila persyaratan pada <i>dataset</i> tidak terpenuhi . . . . .	40
4.6	Diagram <i>package</i> perangkat lunak . . . . .	41
4.7	Diagram kelas pada <i>package Perturbation</i> . . . . .	43
4.8	Diagram kelas pada <i>package Matrix</i> . . . . .	47
4.9	Diagram kelas pada <i>package Preprocessor</i> . . . . .	50
5.1	Tampilan perangkat lunak yang pertama ditampilkan saat perangkat lunak baru dibuka . . . . .	56
5.2	Bagian-bagian pada antarmuka perangkat lunak . . . . .	56
5.3	Bagian antarmuka masukan dan pengaturan perangkat lunak . . . . .	57
5.4	Jendela untuk memilih <i>dataset</i> yang berupa dokumen CSV . . . . .	58
5.5	Tampilan <i>popup</i> yang ditampilkan saat proses memuat dokumen berlangsung . . . . .	59
5.6	Tampilan antarmuka setelah sebuah dokumen dipilih . . . . .	59
5.7	Tampilan antarmuka saat pengguna memilih teknik . . . . .	60
5.8	Tampilan antarmuka saat perangkat lunak membuat dan menyimpan matriks . . . . .	61
5.9	Tampilan <i>popup</i> yang ditampilkan apabila matriks yang ingin diimpor tidak sesuai dengan <i>dataset</i> . . . . .	62
5.10	Tampilan <i>popup</i> peringatan apabila pengguna belum memilih <i>dataset</i> yang diinginkan untuk diacak . . . . .	62
5.11	Tampilan antarmuka parameter teknik <i>Random Projection Perturbation</i> . . . . .	63
5.12	Tampilan <i>popup</i> peringatan tombol “Hitung Nilai Min K” . . . . .	64
5.13	Tampilan saat perangkat lunak sedang melakukan proses pengacakan . . . . .	65
5.14	Tampilan <i>popup</i> untuk memberitahukan pengguna bahwa pengacakan berhasil dilakukan . . . . .	65
5.15	Tampilan <i>popup</i> peringatan apabila pengguna belum memilih atau membuat matriks rotasi/proyeksi . . . . .	66
5.16	Tampilan antarmuka deskripsi <i>dataset</i> setelah pengguna memilih <i>dataset</i> yang ingin diacak . . . . .	67
5.17	Tampilan antarmuka deskripsi hasil pengacakan setelah perangkat berhasil melakukan pengacakan . . . . .	68
5.18	Matriks rotasi dan translasi acak yang dibuat perangkat lunak dan disimpan pada satu dokumen <i>comma-separated values</i> . . . . .	69

5.19 Dua puluh baris pertama <i>dataset mall_customers</i> asli . . . . .	70
5.20 Dua puluh baris pertama <i>dataset mall_customers</i> setelah diacak . . . . .	70
5.21 Matriks proyeksi acak yang dibuat perangkat lunak dan disimpan pada sebuah dokumen <i>comma-separated values</i> . . . . .	73
5.22 Dua puluh baris terakhir <i>dataset mobile_sensor</i> yang asli . . . . .	74
5.23 Dua puluh baris terakhir <i>dataset mobile_sensor</i> setelah diacak . . . . .	74
5.24 Dua puluh baris pertama <i>dataset diabetes</i> asli . . . . .	77
5.25 Dua puluh baris pertama <i>dataset diabetes</i> setelah diacak . . . . .	77
5.26 Grafik akurasi model klasifikasi pada <i>training set</i> dan <i>test set</i> <i>dataset diabetes</i> . . . . .	80
5.27 Grafik akurasi model klasifikasi pada <i>training set</i> dan <i>test set</i> <i>dataset mobile_sensor</i> . . . . .	82
5.28 Grafik <i>Sum of Squared Error</i> model <i>clustering</i> pada <i>dataset mall_customers</i> . . . . .	85
5.29 Grafik <i>Silhouette Score</i> model <i>clustering</i> pada <i>dataset mall_customers</i> . . . . .	85
5.30 Visualisasi <i>cluster</i> pada <i>dataset mall_customers</i> yang asli . . . . .	87
5.31 Visualisasi <i>cluster</i> pada <i>dataset mall_customers</i> yang telah diacak . . . . .	88
5.32 Grafik <i>Sum of Squared Error</i> model <i>clustering</i> pada <i>dataset mobile_sensor</i> . . . . .	89
5.33 Grafik <i>Silhouette Score</i> model <i>clustering</i> pada <i>dataset mobile_sensor</i> . . . . .	89
5.34 Visualisasi <i>cluster</i> pada <i>dataset mobile_sensor</i> . . . . .	91

## DAFTAR TABEL

3.1	Tabel <i>dataset iris</i> yang digunakan sebagai contoh kasus . . . . .	25
5.1	Contoh perbedaan jarak Euclidean antara <i>dataset diabetes</i> asli dan yang telah diacak dengan <i>Random Rotation Perturbation</i> . . . . .	71
5.2	Contoh perbedaan jarak Euclidean antara <i>dataset mobile_sensor</i> asli dan yang telah diacak dengan <i>Random Projection Perturbation</i> . . . . .	75
5.3	Properti-properti pada <i>dataset diabetes</i> asli . . . . .	78
5.4	Properti-properti pada <i>dataset diabetes</i> yang telah diacak . . . . .	78
5.5	Properti-properti pada <i>dataset mobile_sensor</i> asli . . . . .	78
5.6	Properti-properti pada <i>dataset mobile_sensor</i> yang telah diacak . . . . .	79
5.7	Perbandingan Properti Data . . . . .	79
5.8	Perbandingan Model <i>k-nearest neighbors</i> antara model yang dilatih dengan <i>dataset</i> asli dan yang telah diacak . . . . .	83
5.9	Perbandingan model <i>k-means</i> antara model yang dilatih dengan <i>dataset</i> asli dan yang telah diacak . . . . .	92



1

## BAB 1

2

### PENDAHULUAN

#### 3 1.1 Latar Belakang

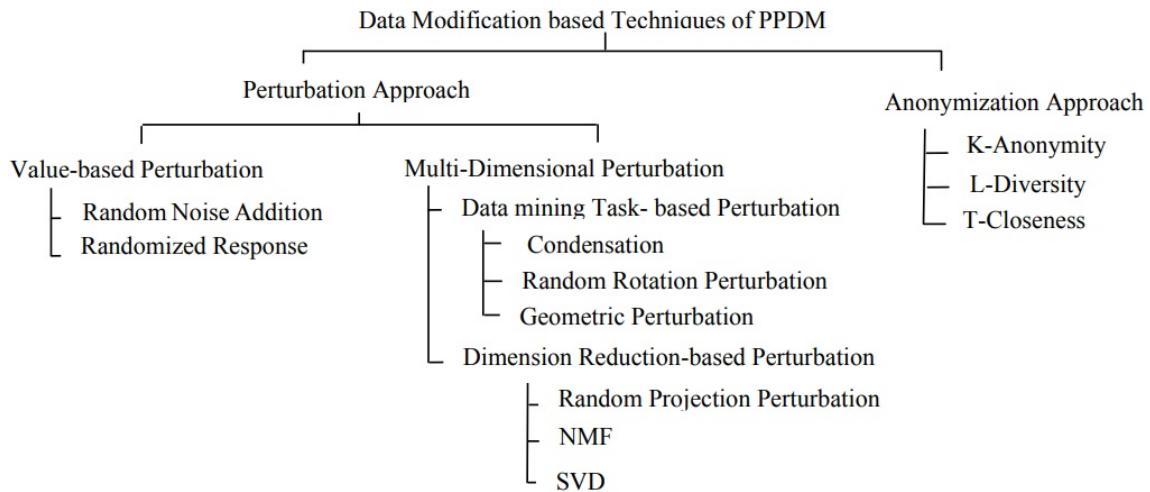
4 Dengan semakin banyaknya penambangan data yang dilakukan dan data yang digunakan juga  
5 semakin banyak, semakin banyak juga privasi di dalam data tersebut yang tersebar kepada pihak  
6 yang melakukan penambangan data. Data privasi tersebut dapat tersebar kepada pihak yang tidak  
7 bertanggung jawab dan disalahgunakan. Oleh karena itu perlu adanya suatu cara untuk mencegah  
8 privasi tersebar pada proses penambangan data, menjaga privasi pada data tersebut. Istilah untuk  
9 hal tersebut adalah *privacy preserving data mining* [1].

10 Ada kesulitan dalam menentukan data seperti apa yang dapat disebut sebagai privasi. Privasi  
11 dapat dikatakan adalah sebuah informasi personal seseorang yang dapat mengidentifikasi suatu hal  
12 pada orang tersebut. Konsep yang sering kali digunakan untuk mendeskripsikan informasi personal  
13 adalah *Personally Identifiable Information* yang disingkat PII. PII [2] adalah segala informasi  
14 mengenai individu yang dikelola oleh sebuah instansi, termasuk segala informasi yang dapat  
15 digunakan untuk membedakan atau mengusut identitas seseorang dan juga segala informasi yang  
16 berhubungan atau dapat dihubungkan kepada suatu individu, seperti informasi medis, pendidikan,  
17 finansial, dan pekerjaan seseorang.

18 Salah satu cara untuk melakukan *privacy preserving data mining* adalah dengan melakukan  
19 modifikasi data yang ada sebelum diberikan kepada pihak lain. Ada macam-macam teknik dan  
20 algoritma yang bertujuan modifikasi data untuk *privacy preserving data mining*, dibagi menjadi  
21 dua jenis yaitu *Perturbation Approach* dan *Anonymization Approach*. *Perturbation Approach* adalah  
22 pendekatan untuk *privacy preserving data mining* dengan cara mengacaukan data yang ada, tetapi  
23 hasil data yang dikacaukan masih tetap dapat ditambah. *Perturbation Approach* dapat dibagi  
24 menjadi dua jenis yaitu *Value-based Perturbation Techniques* dan *Multi-Dimensional Perturbation*.

25 *Value-based Perturbation Techniques* adalah teknik yang bekerja dengan cara menyisipkan  
26 *random noise* pada data. Sedangkan terdapat dua jenis teknik *Multi-Dimensional Perturbation*  
27 yaitu *Data mining Task-based Perturbation* dan *Dimension Reduction-based Perturbation*. *Data*  
28 *mining Task-based Perturbation* adalah teknik yang bekerja dengan cara modifikasi data sehingga  
29 properti yang bertahan pada data yang telah dimodifikasi spesifik hanya properti yang digunakan  
30 oleh suatu teknik penambangan data tertentu. Sedangkan *Dimension Reduction-based Perturbation*  
31 adalah teknik yang bekerja dengan cara modifikasi data sekaligus mengurangi dimensi dari data  
32 asli.

33 Dari berbagai macam teknik modifikasi data untuk *privacy preserving data mining* yang dapat  
34 dilihat pada Gambar 1.1, terdapat empat teknik yang menggunakan metode *Randomization* yaitu



Gambar 1.1: Berbagai macam teknik modifikasi data untuk *privacy preserving data mining*

<sup>1</sup> *Random Noise Addition, Randomized Response, Random Rotation Perturbation, dan Random*  
<sup>2</sup> *Projection Perturbation.* Pada penelitian ini, akan dibuat sebuah perangkat lunak yang dapat  
<sup>3</sup> memproses data yang akan ditambah menjadi data yang telah dimodifikasi dengan metode  
<sup>4</sup> *Randomization* sehingga privasi pada data tersebut terlindungi, tetapi masih dapat ditambah. Dari  
<sup>5</sup> berbagai macam teknik dengan metode *Randomization* yang ada, dipilih dua buah teknik yaitu  
<sup>6</sup> *Random Rotation Perturbation* dan *Random Projection Perturbation* untuk diimplementasikan pada  
<sup>7</sup> perangkat lunak serta membandingkan hasil dari kedua teknik tersebut.

<sup>8</sup> Pengujian akan dilakukan dengan menerapkan penambangan data klasifikasi dengan algoritma *k-*  
<sup>9</sup> *nearest neighbors* dan penambangan data *clustering* dengan algoritma *k-means*. Pada penambangan  
<sup>10</sup> data klasifikasi akan dihitung akurasinya dan dibandingkan antara model yang dilatih dengan  
<sup>11</sup> *dataset* asli dan *dataset* yang telah diacak dengan teknik *Random Rotation Perturbation* dan *Random*  
<sup>12</sup> *Projection Perturbation*. Tujuannya untuk melihat apakah antara kedua model tersebut memiliki  
<sup>13</sup> akurasi yang sama. Pada penambangan data *clustering*, *dataset* asli dan *dataset* yang telah diacak  
<sup>14</sup> akan digunakan untuk membuat model *clustering*. Kemudian visualisasi hasil *clustering*-nya akan  
<sup>15</sup> dibandingkan untuk melihat hasil *cluster* tersebut apakah sama. Kemiripan model *clustering* juga  
<sup>16</sup> akan dihitung dengan metode *Adjusted Rand Index*. Tujuannya untuk melihat apakah antara kedua  
<sup>17</sup> model tersebut memiliki hasil *cluster* yang sama.

## <sup>18</sup> 1.2 Rumusan Masalah

<sup>19</sup> Berdasarkan latar belakang, rumusan masalah pada penelitian ini adalah sebagai berikut.

- <sup>20</sup> 1. Bagaimana cara kerja teknik *Random Rotation Perturbation* dan *Random Projection Pertur-*  
<sup>21</sup> *bation* untuk *privacy preserving data mining*?
- <sup>22</sup> 2. Bagaimana implementasi dari teknik *Random Rotation Perturbation* dan *Random Projection*  
<sup>23</sup> *Perturbation* pada perangkat lunak?
- <sup>24</sup> 3. Bagaimana perbandingan antara hasil dari teknik *Random Rotation Perturbation* dan *Random*  
<sup>25</sup> *Projection Perturbation*?

### 1 1.3 Tujuan

- 2 Berdasarkan rumusan masalah, maka tujuan dari penelitian ini adalah sebagai berikut.
- 3 1. Mempelajari cara kerja dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* untuk *privacy preserving data mining*.
- 5 2. Mengimplementasikan teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* pada perangkat lunak.
- 7 3. Melakukan analisis dan pengujian untuk membandingkan hasil dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* dengan menggunakan penambangan data.

### 9 1.4 Batasan Masalah

- 10 Batasan-batasan masalah untuk penelitian ini adalah sebagai berikut.
- 11 1. Penelitian ini hanya menguji dengan teknik penambangan data klasifikasi dengan algoritma *k-nearest neighbors* dan *clustering* dengan algoritma *k-means*.
- 13 2. Perangkat lunak yang dibuat hanya dapat menerima masukan yang valid yaitu seluruh datanya bersifat numerik.

### 15 1.5 Metodologi

- 16 Metodologi yang digunakan dalam penelitian ini adalah sebagai berikut.
- 17 1. Melakukan studi literatur dasar-dasar privasi data.
- 18 2. Melakukan studi literatur teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* untuk *privacy preserving data mining*.
- 20 3. Melakukan studi literatur teknik penambangan data yang akan digunakan.
- 21 4. Melakukan analisis terhadap teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* serta bagaimana penerapannya dengan teknik penambangan data yang akan digunakan.
- 24 5. Melakukan perancangan perangkat lunak yang mengimplementasikan teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*.
- 26 6. Membangun perangkat lunak yang mengimplementasikan teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*.
- 28 7. Menguji perangkat lunak secara fungsional dan eksperimental dengan menggunakan *real data*.
- 29 8. Menerapkan teknik penambangan data terhadap data yang telah diacak untuk menganalisis hasil dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*.

- 1      9. Melakukan analisis dan pengujian untuk membandingkan hasil penambangan data menggunakan data yang telah diacak dengan teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*.
- 2
- 3
- 4      10. Menarik kesimpulan berdasarkan hasil eksperimen yang telah dilakukan.

## 5    1.6 Sistematika Pembahasan

6    Laporan penelitian tersusun ke dalam enam bab secara sistematis sebagai berikut.

- 7      • Bab 1 Pendahuluan  
8       Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan  
9       sistematika pembahasan.
- 10     • Bab 2 Dasar Teori  
11      Berisi dasar teori tentang dasar-dasar privasi data, *Random Rotation Perturbation*, *Random Projection Perturbation*, *Random Rotation Perturbation*, dan teknik penambangan data.
- 13     • Bab 3 Analisis  
14      Berisi analisis masalah, studi kasus, dan diagram aliran proses.
- 15     • Bab 4 Perancangan  
16      Berisi perancangan perangkat lunak yang dibangun meliputi perancangan antarmuka dan  
17      diagram kelas yang lengkap.
- 18     • Bab 5 Implementasi dan Pengujian  
19      Berisi implementasi antarmuka perangkat lunak, pengujian fungsional, pengujian eksperimental, dan kesimpulan dari pengujian.
- 21     • Bab 6 Kesimpulan dan Saran  
22      Berisi kesimpulan dari awal hingga akhir penelitian dan saran untuk pengembangan selanjutnya.
- 23

<sup>1</sup>

## BAB 2

<sup>2</sup>

### DASAR TEORI

<sup>3</sup> Dalam menjaga privasi data, perlu adanya definisi privasi yang konkret untuk menentukan data  
<sup>4</sup> seperti apa yang menjadi privasi. Pada penambangan data, perlu ada teknik yang baik untuk  
<sup>5</sup> menjaga privasi tidak tersebar kepada orang yang tidak berhak. Ada beberapa teknik untuk menjaga  
<sup>6</sup> privasi pada penambangan data antara lain modifikasi data dengan metode *Randomization* yaitu  
<sup>7</sup> teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*. Pada bab ini akan  
<sup>8</sup> dijelaskan dasar teori yang melandaskan penelitian ini.

#### <sup>9</sup> 2.1 Privasi Data

<sup>10</sup> Pada umumnya sebuah data dapat dikatakan privasi apabila data tersebut dapat dikaitkan dengan  
<sup>11</sup> identitas seseorang. Tetapi setiap orang memiliki kepentingan privasi yang berbeda-beda sehingga  
<sup>12</sup> definisi dari privasi sulit untuk dijelaskan secara eksak. Oleh karena itu, perlu adanya konsep privasi  
<sup>13</sup> yang dapat menjadi acuan untuk menentukan data seperti apa yang termasuk privasi atau bukan.

##### <sup>14</sup> 2.1.1 Privasi

<sup>15</sup> Dalam mendefinisikan privasi, sulit untuk mendapatkan definisi yang tepat untuk privasi karena  
<sup>16</sup> setiap individu memiliki kepentingan yang berbeda-beda sehingga privasi pada setiap individu dapat  
<sup>17</sup> berbeda-beda juga. Beberapa definisi privasi telah dikemukakan dan definisi tersebut bermacam-  
<sup>18</sup> macam berdasarkan konteks, budaya, dan lingkungan [3]. Menurut Warren dan Brandeis pada  
<sup>19</sup> papernya, mereka mendefinisikan privasi sebagai “*the right to be alone.*”, hak untuk menyendiri.  
<sup>20</sup> Lalu pada papernya, Westin mendefinisikan privasi sebagai “*the desire of people to choose freely*  
<sup>21</sup> *under what circumstances and to what extent they will expose themselves, their attitude, and their*  
<sup>22</sup> *behavior to others*”, keinginan orang untuk memilih secara bebas dalam segala situasi dan dalam  
<sup>23</sup> hal mengemukakan diri mereka, sikap mereka, dan tingkah laku mereka pada orang lain.

<sup>24</sup> Schoeman mendefinisikan privasi sebagai “*the right to determine what (personal) information is*  
<sup>25</sup> *communicated to others*”, hak untuk menentukan informasi pribadi apa saja yang dikomunikasikan  
<sup>26</sup> kepada yang lain, atau “*the control an individual has over information about himself or herself.*”  
<sup>27</sup> kendali seorang individu terhadap informasi tentang dirinya sendiri. Lalu baru-baru ini, Garfinkel  
<sup>28</sup> menyatakan bahwa “*privacy is about self-possession, autonomy, and integrity.*”, privasi adalah  
<sup>29</sup> tentang penguasaan diri sendiri, otonomi, dan integritas. Di samping itu, Rosenberg berpendapat  
<sup>30</sup> bahwa privasi sebenarnya bukan sebuah hak tetapi sebuah rasa: “*If privacy is in the end a matter of*  
<sup>31</sup> *individual taste, then seeking a moral foundation for it – beyond its role in making social institutions*

1 possible that we happen to prize – will be no more fruitful than seeking a moral foundation for the  
2 taste for truffles.”, intinya setiap orang memiliki perhatian yang berbeda-beda terhadap privasi  
3 mereka sendiri sehingga hal tersebut tergantung apa yang dirasakan oleh setiap individu.

4 Dari definisi-definisi privasi yang telah disebutkan di atas, dapat disimpulkan bahwa privasi  
5 dilihat sebagai konsep sosial dan budaya [3]. Konsep privasi pada suatu lingkungan dapat berbeda  
6 dari lingkungan lainnya dan hal ini menyebabkan sulitnya menentukan apakah sebuah data termasuk  
7 privasi atau bukan. Oleh karena itu, perlu adanya sebuah standar privasi untuk menentukan data  
8 mana yang dapat disebut sebuah privasi. Organisasi National Institute of Standards and Technology  
9 dari Amerika Serikat, membuat standar mereka sendiri untuk menentukan informasi seperti apa yang  
10 dapat disebut sebagai privasi. Mereka mengemukakan konsep *Personally Identifiable Information*  
11 sebagai informasi yang dapat dikatakan personal untuk setiap individu.

### 12 **2.1.2 *Personally Identifiable Information***

13 Privasi dapat dikatakan adalah sebuah informasi personal seseorang yang dapat mengidentifikasi  
14 suatu hal pada orang tersebut. Konsep yang sering kali digunakan untuk mendeskripsikan informasi  
15 personal adalah *Personally Identifiable Information* yang disingkat PII. PII adalah segala informasi  
16 mengenai individu yang dikelola oleh sebuah instansi, termasuk segala informasi yang dapat  
17 digunakan untuk membedakan atau mengusut identitas seseorang dan juga segala informasi yang  
18 berhubungan atau dapat dihubungkan kepada suatu individu, seperti informasi medis, pendidikan,  
19 finansial, dan pekerjaan seseorang [2].

20 Informasi yang termasuk membedakan individu adalah informasi yang dapat mengidentifikasi  
21 seorang individu. Informasi seperti ini adalah data privasi yang secara langsung bisa didapatkan.  
22 Beberapa contoh informasi yang mengidentifikasi seorang individu adalah nama, nomor KTP,  
23 tempat tanggal lahir, nama ibu kandung, atau catatan medis. Sedangkan, data yang hanya berisi  
24 misalkan saldo tabungan tanpa ada informasi lain mengenai identitas seseorang yang berkaitan  
25 tidak menyediakan informasi yang cukup untuk mengidentifikasi seorang individu.

26 Dari sebuah data, bisa saja data tersebut secara tidak langsung mengandung privasi, identitas  
27 seseorang bisa didapatkan tanpa data tersebut memberikan langsung identitas orang tersebut.  
28 Mengusut identitas seseorang adalah proses dari membuat perkiraan tentang aspek spesifik dari  
29 aktivitas atau status seseorang. Jika sebuah data dapat dianalisis datanya sampai identitas seseorang  
30 dapat diakses, berarti data tersebut secara tidak langsung mengandung privasi. Contohnya adalah  
31 sebuah catatan finansial seseorang dapat digunakan untuk memperkirakan aktivitas dari individu  
32 tersebut.

33 Informasi yang berhubungan dapat didefinisikan sebagai informasi yang berkaitan dengan seorang  
34 individu yang mana terkait secara logis dengan informasi lain tentang individu tersebut. Informasi  
35 tersebut secara tidak langsung mengandung privasi dan dapat diolah agar identitas seseorang bisa  
36 didapatkan. Contohnya adalah apabila ada dua buah basis data yang memiliki data berbeda dari  
37 seorang individu, maka seseorang yang memiliki akses pada 2 basis data tersebut berpotensi dapat  
38 mengaitkan data-data tersebut lalu mengidentifikasi individu yang ada pada data tersebut.

## 1 **2.2 Penambangan Data**

- 2 Pada era teknologi informasi, sangat banyak data terkumpul pada basis data. Data yang masif ini  
3 dapat dimanfaatkan untuk menggali informasi penting yang berguna untuk pembuatan keputusan.  
4 Proses pada aktivitas ini secara kasar dapat disebut dengan penambangan data.

5 Penambangan data adalah proses mengekstrak sebuah pola atau sebuah pengetahuan dari  
6 kumpulan data yang besar, yang mana dapat direpresentasikan dan diinterpretasikan [4]. Pada  
7 penambangan data, teknik *machine learning* dan *pattern recognition* intensif digunakan untuk  
8 mendapatkan pola maupun pengetahuan baru dari data. Tujuan utama dari penambangan data  
9 adalah untuk membentuk model deskriptif dan prediktif dari suatu data. Model deskriptif berusaha  
10 untuk mengubah pola-pola yang ada pada data menjadi deskripsi yang dapat dimengerti oleh orang  
11 awam. Sedangkan model prediktif digunakan untuk memprediksi data yang tidak diketahui atau  
12 data yang berpotensi muncul di kemudian hari.

13 Model tersebut biasanya dibuat dengan menggunakan teknik *machine learning*, yang mana  
14 terdapat dua teknik *machine learning* yang paling sering digunakan yaitu klasifikasi dan *clustering*.  
15 Subbab berikutnya akan menjelaskan secara singkat kedua teknik tersebut dan contoh algoritmanya.

### 16 **2.2.1 Klasifikasi**

17 Tujuan utama klasifikasi adalah membuat model yang dalam kasus ini disebut *classifier* yang mana  
18 dapat mengidentifikasi nilai kelas dari suatu data [4]. Dalam kata lain, sebuah *classifier* dibuat dari  
19 sebuah *training set* dan model ini digunakan untuk mengklasifikasi data tidak diketahui ke dalam  
20 salah satu kelas.

21 Ada dua tahap dalam proses klasifikasi yaitu tahap latihan dan tahap klasifikasi. Pada tahap  
22 latihan, model akan dibuat dengan menggunakan *training set*. *Training set* yang dimaksud adalah  
23 data yang sudah diketahui kelasnya sehingga model yang ada melatih dirinya. Setelah *classifier*  
24 terbentuk, barulah tahap klasifikasi dapat dilakukan dengan menggunakan *classifier* yang tadi  
25 sudah dibuat. *Classifier* akan memprediksi data yang kelasnya tidak diketahui. *Classifier* akan  
26 semakin baik performanya seiring dengan banyaknya tahap latihan yang dilakukan.

27 Teknik *machine learning* yang paling dikenal untuk klasifikasi antara lain *K-nearest Neighbors*,  
28 *Decision Tree*, dan *Naive Bayes*. Dalam penelitian ini, hanya teknik *K-nearest Neighbors* yang  
29 digunakan untuk pengujian sehingga berikutnya hanya akan dijelaskan teknik *K-nearest Neighbors*  
30 saja. Teknik *K-nearest Neighbors* adalah teknik penambangan data klasifikasi yang mencari label  
31 terbanyak pada sejumlah tetangga terdekatnya. Teknik ini bergantung pada jarak Euclidean antara  
32 titik yang merepresentasikan sebuah objek data yang akan diprediksi dengan titik-titik lain yang  
33 terdekat (tetangga-tetangganya). Setiap objek data pada *dataset* dipetakan ke bidang Euclidean  
34 dengan nilai dari beberapa atribut yang menentukan letaknya pada bidang Euclidean [5].

35 Berikut langkah kerja dari teknik *K-nearest Neighbors*.

- 36 1. Tentukan nilai *k* yang menentukan seberapa banyak tetangga yang digunakan.  
37 2. Lakukan perulangan dengan iterasi sebanyak rekord yang ada selain rekord yang ingin  
38 diprediksi labelnya.

- 1       (a) Hitung jarak Euclidean antara rekord iterasi sekarang dengan rekord yang ingin diprediksi  
2       labelnya.
- 3       (b) Catat jarak Euclidean dari rekord yang ingin diprediksi dan indeks rekord iterasi sekarang.
- 4       3. Urutkan jarak Euclidean titik-titik yang sudah dihitung pada perulangan pada langkah  
5       sebelumnya secara menaik.
- 6       4. Pilih rekord teratas (jarak Euclidean yang paling kecil) sebanyak  $k$  dari urutan pada langkah  
7       sebelumnya.
- 8       5. Ambil label dari semua rekord yang terpilih pada langkah sebelumnya. Label terbanyak  
9       adalah hasil prediksi label pada rekord yang ingin diprediksi.

10      Dalam mengevaluasi model klasifikasi dapat dihitung akurasi model dalam memprediksi label  
11     suatu data. Akurasi pada model klasifikasi berupa persentase jumlah prediksi yang benar dengan  
12     jumlah data yang diprediksi [5]. Misalkan apabila model klasifikasi menebak dengan benar 8 buah  
13     objek data dari total berjumlah 10 objek data, berarti model tersebut memiliki akurasi sebesar 80%.  
14      Berikut akan dijelaskan rumus untuk menghitung akurasi model klasifikasi.

Misalkan sebuah *dataset* yang sudah berlabel dan hanya memiliki 2 buah kelas label yang masing-masing akan disebut dengan *positive tuples* dan *negative tuples*. *Dataset* tersebut dibagi menjadi 2 bagian masing-masing yaitu *training set* dan *test set*. Model klasifikasi akan dilatih dengan *training set*. Kemudian model tersebut akan dipakai untuk memprediksi label pada *test set*. Rumus untuk menghitung akurasi model tersebut memprediksi *test set* dapat dilihat pada Persamaan 2.1.

$$\text{akurasi} = \frac{TP + TN}{P + N} \quad (2.1)$$

15      Berikut akan dijelaskan variabel-variabel yang ada pada Persamaan 2.1.

- 16      •  $P$  adalah jumlah objek data yang memiliki label *positive tuples*.
- 17      •  $N$  adalah jumlah objek data yang memiliki label *negative tuples*.
- 18      •  $TP$  (*true positives*) adalah jumlah objek data yang diprediksi dengan benar oleh model  
19       klasifikasi bahwa labelnya adalah *positive tuples*.
- 20      •  $TN$  (*true negatives*) adalah jumlah objek data yang diprediksi dengan benar oleh model  
21       klasifikasi bahwa labelnya adalah *negative tuples*.

### 22      2.2.2 *Clustering*

23      *Clustering* adalah proses mengelompokan kumpulan objek ke dalam sebuah kelompok (*cluster*)  
24      sedemikian rupa sehingga objek-objek dari suatu *cluster* memiliki lebih banyak kemiripan dari pada  
25      objek-objek dari *cluster* lainnya [4].

26      Salah satu contoh teknik *clustering* adalah *K-means*. Teknik *k-means* adalah teknik  
27      penambangan data *clustering* yang memanfaatkan jarak Euclidean antara titik-titik yang ada untuk  
28      menentukan titik mana saja yang masuk ke kluster mana.

29      Berikut langkah kerja dari teknik *K-means*. [5]

- 1 1. Tentukan nilai variabel  $k$  yang menentukan seberapa banyak kluster yang diinginkan dan  
2 sebuah *threshold* untuk menentukan batas perubahan nilai *centroid*.
- 3 2. Tentukan secara acak sebuah *centroid* sebanyak  $k$  untuk setiap kluster.
- 4 3. Lakukan perulangan sampai nilai fitur-fitur semua *centroid* (titik tengah kluster) relatif tidak  
5 berubah atau dengan kata lain perubahannya kurang dari *threshold*.
  - 6 (a) Menghitung jarak Euclidean tiap titik dari *centroid* ke titik tersebut dengan menggunakan  
7 beberapa fitur yang dipilih.
  - 8 (b) Kluster yang memiliki jarak Euclidean paling kecil dengan sebuah titik adalah kluster  
9 titik tersebut.
  - 10 (c) Tentukan kembali *centroid* setiap kluster dengan cara menghitung rata-rata tiap fitur  
11 pada seluruh objek data pada kluster tersebut.

12 Dalam menentukan nilai variabel  $k$  (jumlah *cluster*) terbaik ada 2 metode yang dapat dipakai  
13 yaitu menghitung nilai *Silhouette Score* tertinggi dan metode Elbow. Metode Elbow bekerja dengan  
14 menggunakan *Sum of Squared Error* untuk membuat sebuah grafik hubungan antara nilai variabel  $k$   
15 dengan nilai *Sum of Squared Error*. Penentuan nilai variabel  $k$  dapat dilihat pada grafik tersebut nilai  
16 variabel  $k$  mana yang perbedaan nilai *Sum of Squared Error*-nya dengan nilai variabel  $k$  sebelumnya  
17 memiliki perbedaan yang paling besar atau apabila dilihat pada grafiknya akan berbentuk seperti  
18 siku atau bengkokan.

*Sum of Squared Error* adalah jumlah dari kuadrat jarak Euclidean setiap objek data terhadap  
*centroid* pada masing-masing *clusternya* [5]. Rumus untuk menghitung *Sum of Squared Error* dapat  
dilihat pada Persamaan 2.2.

$$E = \sum_{i=1}^k \sum_{p \in C_i} dist(\mathbf{p}, \mathbf{c}_i)^2 \quad (2.2)$$

19 Variabel  $\mathbf{p}$  adalah titik pada bidang Euclidean yang merepresentasikan sebuah objek data. Variabel  
20  $\mathbf{c}_i$  adalah *centroid* pada *cluster*  $C_i$ .

*Silhouette Score* adalah rata-rata dari koefisien *Silhouette* setiap objek data yang ada. Sementara  
koefisien *Silhouette* adalah ukuran seberapa mirip suatu objek data dengan *cluster*-nya sendiri  
dibandingkan dengan *cluster* lain. Nilai dari koefisien *Silhouette* berada di antara -1 dan 1. Rumus  
untuk menghitung koefisien *Silhouette* dapat dilihat pada Persamaan 2.3. [5]

$$s(\mathbf{o}) = \frac{b(\mathbf{o}) - a(\mathbf{o})}{\max \{a(\mathbf{o}), b(\mathbf{o})\}} \quad (2.3)$$

Variabel  $\mathbf{o}$  adalah sebuah objek data. Fungsi  $a(\mathbf{o})$  adalah rata-rata jarak Euclidean antara sebuah  
objek data dan seluruh objek data pada *cluster*-nya sendiri. Rumus dari fungsi ini dapat dilihat  
pada Persamaan 2.4.

$$a(\mathbf{o}) = \frac{\sum_{\mathbf{o}' \in C_i, \mathbf{o} \neq \mathbf{o}'} dist(\mathbf{o}, \mathbf{o}')}{|C_i| - 1} \quad (2.4)$$

Sementara fungsi  $b(\mathbf{o})$  adalah rata-rata jarak Euclidean paling kecil antara sebuah objek data dan  
seluruh objek data pada *cluster* lain yang bukan *cluster*-nya sendiri. Rumus dari fungsi ini dapat

dilihat pada Persamaan 2.5.

$$b(\mathbf{o}) = \min_{C_j: 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{\mathbf{o}' \in C_j} dist(\mathbf{o}, \mathbf{o}')}{|C_j|} \right\} \quad (2.5)$$

Dalam membandingkan kedua model *clustering* terdapat suatu metode yang baik untuk digunakan yaitu metode *Adjusted Rand Index*. *Adjusted Rand Index* [6] adalah metode untuk menghitung kemiripan dari sebuah model *clustering* dengan model *clustering* lainnya. Nilai dari *Adjusted Rand Index* menunjukkan kemiripan anggota *cluster-cluster* yang ada pada dua model *clustering* yang ingin dibandingkan. Rumus untuk menghitung nilai *Adjusted Rand Index* dapat dilihat pada Persamaan 2.6.

$$ARI(\Pi, \Pi') = \frac{r - exp}{max - exp} \quad (2.6)$$

Dimana  $\Pi$  dan  $\Pi'$  masing-masing merepresentasikan sebuah model *clustering* yang berbeda dari *dataset* yang sama.  $r$  merepresentasikan banyaknya dua buah objek data yang berbeda berada di *cluster* yang sama pada  $\Pi$  maupun  $\Pi'$ . Rumus untuk menghitung variabel *exp* dapat dilihat pada Persamaan 2.7. Rumus untuk menghitung variabel *max* dapat dilihat pada Persamaan 2.8.

$$exp = (np(\Pi)np(\Pi'))/(n(n - 1)/2) \quad (2.7)$$

Dimana  $n$  adalah jumlah objek data pada *dataset*.  $np(\Pi)$  adalah  $r + s$  dimana  $s$  merepresentasikan banyaknya dua buah objek data yang berbeda berada di *cluster* yang sama pada  $\Pi$  tetapi tidak pada  $\Pi'$ .  $np(\Pi')$  adalah  $r + t$  dimana  $t$  merepresentasikan banyaknya dua buah objek data yang berbeda berada di *cluster* yang sama pada  $\Pi'$  tetapi tidak pada  $\Pi$ .

$$max = 0.5(np(\Pi) + np(\Pi')) \quad (2.8)$$

1 Apabila model *clustering* dibuat dengan data yang berdimensi sangat besar, ada kesulitan untuk  
 2 melakukan visualisasi. Oleh karena itu, ada suatu cara untuk mempermudah visualisasi model  
 3 yaitu mereduksi dimensi dengan teknik *Principal Component Analysis* tanpa merusak data secara  
 4 signifikan. *Principal Component Analysis* [5] yang disingkat menjadi PCA bekerja dengan cara  
 5 “menggabungkan” intisari dari atribut-atribut yang ada dengan membuat *dataset* alternatif yang  
 6 lebih kecil. PCA mencari  $k$   $n$ -dimensi vektor *orthogonal* yang terbaik untuk merepresentasikan  
 7 data yang ingin direduksi dimensinya. Berikut adalah gambaran kasar cara kerja teknik *Principal  
 8 Component Analysis*.

- 9 1. *Dataset* dinormalisasi sehingga nilai pada setiap atribut yang ada jatuh pada rentang yang  
 10 sama.
- 11 2. PCA menghitung  $k$  vektor *orthogonal* yang menyediakan dasar untuk *dataset* yang telah  
 12 dinormalisasi.
- 13 3. *Pricipal components* diurutkan secara menurun menurut “kepentingan” atau kekuatannya.
- 14 4. Oleh karena komponen yang ada diurutkan secara menurun menurut “kepentingannya” maka  
 15 ukuran data dapat direduksi dengan mengeliminasi komponen yang lebih lemah.

### **2.3 Privacy Preserving Data Mining**

Aktivitas penambangan data melibatkan jumlah data yang sangat masif. Data-data yang digunakan memiliki privasi banyak individu di dalamnya. Hal ini berpotensi menyebabkan pelanggaran privasi dalam kasus tidak adanya proteksi yang cukup dan penyalahgunaan privasi data untuk tujuan lain [1]. Faktor utama pelanggaran privasi pada penambangan data adalah penyalahgunaan data sehingga hal ini dapat merugikan seorang individu maupun sebuah organisasi. Oleh karena itu, ada kebutuhan untuk menghindari penyebaran informasi pribadi yang rahasia maupun pengetahuan lainnya yang dapat diambil dari data yang digunakan untuk aktivitas penambangan data.

Konsep privasi sering kali lebih kompleks dari pada yang dibayangkan. Dalam kasus penambangan data, definisi dari menjaga privasi masih tidak jelas. Ada sebuah paper yang mendefinisikan *privacy preserving data mining* sebagai “getting valid data mining results without learning the underlying data values”, mendapatkan hasil penambangan data yang valid tanpa nilai pada data. Tetapi pada saat ini setiap teknik *privacy preserving data mining* yang ada memiliki definisi privasinya masing-masing.

Salah satu cara untuk melakukan *privacy preserving data mining* adalah dengan melakukan modifikasi data yang ada sebelum diberikan kepada pihak lain. Berbagai macam pendekatan modifikasi data untuk *privacy preserving data mining* telah dikembangkan antara lain *Perturbation Approach* dan *Anonymization Approach*, selengkapnya dapat dilihat pada Gambar 1.1 [1]. *Perturbation Approach* adalah pendekatan untuk *privacy preserving data mining* dengan cara mengacaukan data yang ada, tetapi hasil data yang dikacaukan masih tetap dapat ditambah. Sedangkan pada *Anonymization Approach*, data diterapkan de-identifikasi di mana *dataset* mentah disebarluaskan setelah menghapus inti dari identitas setiap rekord [1].

*Perturbation Approach* dapat dibagi menjadi dua jenis lagi yaitu *Value-based Perturbation Techniques* dan *Multi-Dimensional Perturbation*. *Value-based Perturbation Techniques* adalah teknik yang bekerja dengan cara menyisipkan *random noise* pada data. Sedangkan terdapat dua jenis teknik *Multi-Dimensional Perturbation* yaitu *Data mining Task-based Perturbation* dan *Dimension Reduction-based Perturbation*. *Data mining Task-based Perturbation* adalah teknik yang bekerja dengan cara modifikasi data sehingga properti yang bertahan pada data yang telah dimodifikasi spesifik hanya properti yang digunakan oleh suatu teknik penambangan data tertentu. Sedangkan *Dimension Reduction-based Perturbation* adalah teknik yang bekerja dengan cara modifikasi data sekaligus mengurangi dimensi dari data asli.

Hal yang sering kali diperhatikan pada teknik-teknik *Perturbation Approach* adalah perbandingan antara jumlah privasi yang hilang dan jumlah informasi yang hilang. Idealnya teknik *Perturbation Approach* yang baik adalah teknik yang fokus meminimalkan jumlah privasi yang hilang dan jumlah informasi yang hilang sehingga hasil penambangan dan akurasinya sama baiknya dengan tanpa menerapkan teknik *Perturbation Approach*. Setiap teknik penambangan data memakai properti yang berbeda-beda pada data yang ditambah. Oleh karena itu, properti yang terjaga juga sebaiknya berdasarkan properti yang digunakan pada teknik penambangan data yang digunakan [7]. Pada saat ini, teknik modifikasi data yang ada sering kali memiliki perbedaan pada properti-properti yang terjaga. Teknik-teknik modifikasi data tertentu sering kali memiliki fungsi yang berbeda atau teknik penambangan data yang dapat digunakan berbeda karena properti yang terjaga pada teknik-teknik tersebut berbeda juga.

## **1 2.4 Metode *Randomization***

**2** Dari berbagai macam teknik modifikasi data untuk *privacy preserving data mining* yang dapat dilihat  
**3** pada Gambar 1.1, terdapat empat teknik yang menggunakan metode *Randomization* yaitu *Random*  
**4** *Noise Addition*, *Randomized Response*, *Random Rotation Perturbation*, dan *Random Projection*  
**5** *Perturbation*.

**6** Berbagai macam teknik dengan metode *Randomization* umumnya menerapkan perusakan nilai  
**7** pada data. Salah satu teknik yang pertama kali menggunakan metode *Randomization* untuk  
**8** *privacy preserving data mining* adalah teknik *Random Noise Addition* yang dikemukakan oleh  
**9** Agrawal dan Srikant pada paper berikut [8]. Teknik *Random Noise Addition* ini dilakukan dengan  
**10** cara menambahkan nilai random (*noise*) pada data. Nilai random tersebut diambil dari sebuah  
**11** distribusi. Untuk menambah data yang telah ditambahkan *noise* ini perlu dilakukan rekonstruksi  
**12** distribusi untuk mendapatkan distribusi yang asli. Oleh karena itu, teknik *Random Noise Addition*  
**13** ini hanya menjaga distribusi data asli sehingga hanya teknik penambangan data yang bergantung  
**14** pada distribusi data saja yang dapat digunakan. Penyesuaian pada algoritma penambangan data  
**15** yang digunakan juga perlu dilakukan agar teknik *Random Noise Addition* ini dapat digunakan  
**16** dan mendapatkan hasil penambangan data yang hampir sama dengan penambangan data tanpa  
**17** menggunakan teknik *Random Noise Addition*.

**18** Setelah teknik *Random Noise Addition* ditemukan, berbagai macam teknik lain juga dikembangkan  
**19** terinspirasi dari teknik *Random Noise Addition* ini. Teknik *Random Rotation Perturbation* dan  
**20** *Random Projection Perturbation* adalah teknik adalah salah satunya, tetapi teknik tersebut tidak  
**21** dilakukan dengan cara menambahkan *noise* melainkan mengkalikan data asli dengan nilai random.  
**22** Bagaimanapun juga, inti dari metode *Randomization* yang telah disebutkan di atas masih sama  
**23** yaitu merusak data sehingga data yang dirilis bukanlah data asli melainkan data yang sudah rusak  
**24** sehingga data yang dirilis tidak mengandung privasi sehingga privasinya terjaga. Masing-masing  
**25** dari dua teknik tersebut akan dijelaskan lebih detil pada subbab-subbab berikutnya.

### **26 2.4.1 *Random Noise Addition***

**27** Ide utama dari teknik *Random Noise Addition* [8] adalah mendistorsi nilai pada data dengan cara  
**28** menambahkan *random noise* yang diambil dari distribusi *Uniform* atau *Gaussian* dan memiliki  
**29** rata-rata bernilai 0. Tetapi menurut penelitian yang telah dilakukan, distribusi *Gaussian* lebih  
**30** baik digunakan untuk teknik ini. *Random noise* yang digunakan memiliki nilai yang berbeda untuk  
**31** setiap nilai pada data.

**32** Dengan teknik *Random Noise Addition*, dari data yang sudah didistorsi bisa didapatkan kembali  
**33** distribusi data asli dengan merekonstruksi distribusinya tanpa mendapatkan setiap nilai-nilai yang  
**34** ada pada data asli. Metode rekonstruksi yang digunakan berdasarkan pada aturan *Bayes*. Algoritma  
**35** rekonstruksi untuk mendapatkan distribusi dari data asli dapat dilihat pada Algoritma 1. Algoritma  
**36** ini berhenti sampai kriteria berhentinya terpenuhi. Kriteria tersebut adalah perbedaan estimasi  
**37** distribusi iterasi sekarang dengan yang sebelumnya sangat kecil. Algoritma ini akan menghasilkan  
**38** estimasi distribusi data asli dengan menggunakan data yang telah terdistorsi tanpa menggunakan  
**39** nilai-nilai pada data asli, sehingga nilai-nilai pada data asli tidak tersebar. Oleh karena teknik  
**40** *Random Noise Addition* hanya menjaga distribusi pada data maka teknik penambangan data yang

- 1 dapat digunakan hanya teknik-teknik yang bergantung pada distribusi data saja.

**Algorithm 1:** Merekonstruksi distribusi data yang asli dari data yang telah diacak

**Function** rekonstruksiDistribusi(*data*):

Masukan : Data yang telah diacak

Keluaran : Distribusi data yang asli hasil rekonstruksi

```

1   begin
2        $f_X^0 :=$ Uniform distribution
3        $j := 0$ 
4       repeat
5            $f_X^{j+1}(a) := \frac{1}{n} \sum_{i=1}^n \frac{f_Y(w_i-a)f_X^j(a)}{\int_{-\infty}^{\infty} f_Y(w_i-z)f_X^j(z)dz}$ 
6            $j := j + 1$ 
7       until stopping criterion met
8   end

```

- 2 Modifikasi pada algoritma penambangan data yang digunakan juga perlu dilakukan. Contohnya  
 3 apabila algoritma pohon keputusan digunakan, maka perlu modifikasi pada algoritma pohon  
 4 keputusan tersebut. Hal ini menimbulkan masalah pada aplikasi pada dunia nyata karena tidak  
 5 efisien dan memakan waktu untuk memodifikasi setiap algoritma yang ingin digunakan untuk  
 6 menyesuaikan dengan teknik *Random Noise Addition*. Masalah mengenai algoritma yang dapat  
 7 digunakan juga menjadi perhatian karena teknik *Random Noise Addition* hanya dapat digunakan  
 8 untuk algoritma yang bergantung pada distribusi saja sedangkan teknik lain tidak menjaga distribusi  
 9 pada data sehingga sulit untuk membandingkan dengan teknik lain pada penelitian ini. Ada juga  
 10 penelitian yang mengatakan bahwa teknik *Random Noise Addition* ini memiliki kualitas yang kurang  
 11 baik dalam menjaga privasi data karena banyaknya celah yang dapat diserang pada teknik ini. Oleh  
 12 karena masalah-masalah tersebut, akhirnya teknik ini juga tidak akan digunakan untuk penelitian  
 13 ini. Teknik *Random Projection Perturbation* akan digunakan untuk menggantikan teknik *Random*  
 14 *Noise Addition*.

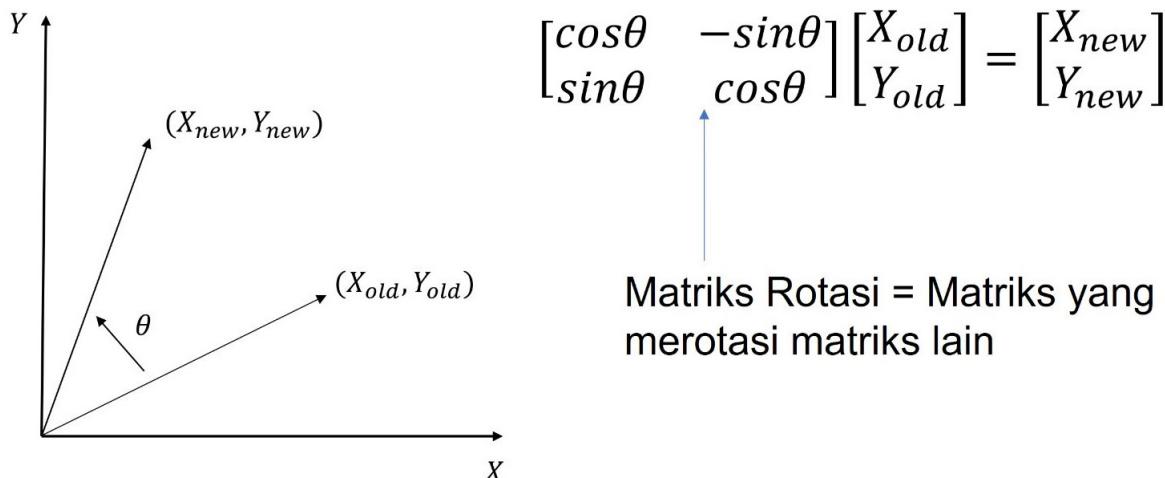
#### 15 2.4.2 Random Rotation Perturbation

- 16 Ide utama dari teknik *Random Rotation Perturbation* adalah melakukan transformasi rotasi pada  
 17 data. Dalam rangka memahami teknik ini, dapat dilihat ilustrasi penjelasannya pada Gambar 2.1.  
 18 Misalkan sebuah data yang memiliki 2 buah fitur dipetakan kepada bidang Euclidean 2 dimensi. Ada  
 19 sebuah objek data yang direpresentasikan sebagai titik pada bidang Euclidean dengan nilai  $X_{old}$  dan  
 20  $Y_{old}$ . Teknik *Random Rotation Perturbation* melakukan rotasi terhadap bidang Euclidean tersebut  
 21 sebesar  $\theta$  derajat sehingga nilai objek data tersebut berubah menjadi  $X_{new}$  dan  $Y_{new}$ . Apabila  
 22 rotasi tersebut dilakukan pada data yang direpresentasikan sebagai matriks maka diperlukan sebuah  
 23 matriks rotasi. Pada teknik *Random Rotation Perturbation*, matriks rotasi akan dibuat secara acak.

Jika data direpresentasikan sebagai matriks  $X_{n \times d}$ , *rotation perturbation* dari dataset X didefinisikan sebagai berikut.

$$G(X) = X_{n \times d} R_{d \times d} \quad (2.9)$$

- 24 Dimana  $R_{d \times d}$  adalah matriks rotasi yang dibuat secara acak. Matriks rotasi berukuran  $d$  dimensi  
 25 dapat dibuat dengan cara membuat matriks *special orthogonal* secara acak karena sebuah matriks



Gambar 2.1: Ilustrasi penjelasan teknik *Random Rotation Perturbation* pada bidang Euclidean

1 special orthogonal pasti merupakan matriks rotasi. Matriks *special orthogonal* adalah matriks  
 2 yang memiliki sifat *orthogonal* dan determinannya bernilai +1, yang mana matriks *orthogonal*  
 3 adalah matriks yang menghasilkan matriks identitas apabila dikalikan dengan transposenya sendiri.  
 4 Matriks rotasi ini dapat dibuat secara efisien mengikuti distribusi Haar [9]. Dari definisi di atas  
 5 dapat disimpulkan transformasi rotasi tersebut menjaga jarak Euclidean [7].

6 Teknik ini menjaga beberapa properti pada data antara lain yaitu jarak Euclidean, *inner*  
 7 *product*, dan *geometric shape hyper* pada bidang multi-dimensi [1]. Oleh karena itu, beberapa  
 8 teknik penambangan data tidak berpengaruh (dapat digunakan) terhadap teknik *Random Rotation*  
 9 *Perturbation* antara lain yaitu *K-nearest Neighbors*, *Support Vector Machines*, dan *Perceptrons* [7].  
 10 Teknik ini dipercaya dapat memberikan hasil penambangan yang maksimal, hasil penambangan  
 11 data yang telah dirusak persis sama dengan hasil penambangan data aslinya. Sehingga jumlah  
 12 informasi yang hilang tidak ada, tetapi jumlah privasi yang hilangnya tinggi. Walaupun demikian  
 13 ada beberapa penelitian yang mengatakan bahwa teknik *Random Rotation Perturbation*  
 14 ini memiliki sifat demikian sehingga teknik ini dikatakan tidak aman dan dapat diserang dengan  
 15 beberapa teknik untuk mendapatkan data asli yang lengkap.

16 Transformasi translasi juga perlu dilakukan agar rotasi yang dilakukan merusak data secara  
 17 menyeluruh. Apabila tidak dilakukan translasi, nilai pada data yang mendekati nilai nol akan  
 18 menghasilkan nilai yang mendekati nol juga setelah dirotasi. Implikasi dari hal tersebut adalah  
 19 lemahnya dalam menjaga privasi. Translasi dapat dilakukan dengan cara membuat matriks translasi  
 20 yang acak lalu kalikan dengan matriks data asli. Translasi dapat dilakukan karena translasi tidak  
 21 mengubah properti geometris dari matriks yang ditranslasi sehingga jarak Euclidean terjaga dan  
 22 hasil penambangan data yang bergantung pada jarak Euclidean tetap sama.

### 23 Algoritma

24 Algoritma *Random Rotation Perturbation* memiliki beberapa langkah yaitu sebagai berikut.

- 25 1. *Dataset* yang memiliki atribut sebanyak  $d$  dan rekord sebanyak  $n$  direpresentasikan dalam  
 26 bentuk matriks berukuran  $n \times d$ .
- 27 2. Buatlah matriks translasi acak yang diambil mengikuti distribusi *uniform* dengan rentang [0,

- 1        100] berdimensi  $(d + 1) \times (d + 1)$ .
- 2        3. Untuk keperluan transformasi translasi, matriks *dataset* perlu ditambahkan sebuah kolom  
3        dengan nilai 1 pada seluruh barisnya.
- 4        4. Lakukan transformasi translasi dengan cara mengkalikan matriks *dataset* dengan matriks  
5        translasi yang telah dibuat pada langkah kedua.
- 6        5. Oleh karena keperluan transformasi translasi, hasil translasi akan berupa matriks berdimensi  
7         $n \times (d + 1)$  dengan kolom terakhir berisi nilai 1 pada setiap barisnya. Oleh karena itu, kolom  
8        tersebut perlu dibuang agar dimensi matriks *dataset* kembali sesuai aslinya  $n \times d$ .
- 9        6. Buatlah matriks rotasi berukuran  $d \times d$  dengan membuat matriks *special orthogonal* secara  
10      acak. Matriks *special orthogonal* memiliki sifat yaitu determinannya sebesar 1 dan hasil  
11      perkalian matriks tersebut dengan transposenya adalah matriks identitas.
- 12      7. Lakukan transformasi rotasi dengan cara mengkalikan matriks *dataset* dengan matriks rotasi  
13      yang telah dibuat pada langkah keenam.
- 14      8. Hasil matriks yang telah dirotasi sudah dapat langsung digunakan untuk penambangan data.

#### 15      2.4.3 Random Projection Perturbation

16      Ide utama dari teknik *Random Projection Perturbation* adalah mereduksi dimensi dari representasi  
17      matriks data asli dengan syarat dimensi matriks tersebut cukup besar. Dasar dari teknik *Random*  
18      *Projection Perturbation* berdiri pada *Johnson-Lindenstrauss Lemma*. [10]

19      **Lemma 1 (JOHNSON-LINDENSTRAUSS LEMMA).** *For any  $0 < \epsilon < 1$  and any integer  $s$ , let  
20       $k$  be a positive integer such that  $k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n$ . Then, for any set  $S$  of  $s = |S|$  data  
21      points in  $\mathbb{R}^m$ , there is a map  $f : \mathbb{R}^m \rightarrow \mathbb{R}^k$  such that, for all  $x, y \in S$ ,  $(1 - \epsilon)s||u - v||^2 <  
22      ||p(u) - p(v)||^2 < (1 + \epsilon)s||u - v||^2$ , where  $||\cdot||$  denotes the vector 2-norm.*

23      Inti dari Lemma ini menunjukkan bahwa titik-titik pada bidang Euclidean  $d$ -dimensi dapat  
24      diproyeksikan ke bidang Euclidean yang berdimensi lebih kecil dari  $d$  tetapi jarak Euclidean antara  
25      setiap titik tetap terjaga dengan distorsi yang terkontrol tetapi dengan syarat  $d$  harus cukup besar.  
26      Oleh karena adanya distorsi yang muncul, akurasi pada model yang dibuat dengan data tersebut  
27      akan berkurang dibandingkan data aslinya. [11]

28      *Projection perturbation* dari *dataset X* didefinisikan sebagai berikut.

$$G(X) = X_{n \times d} R_{d \times k} \quad (2.10)$$

28      Dimana  $R_{d \times k}$  adalah *random projection matrix* yang dihasilkan mengikuti distribusi normal, dengan  
29      rata-rata bernilai 0 dan standar deviasi bernilai  $1/\sqrt{k}$ . Ukuran matriks  $R_{d \times k}$  disesuaikan dengan  
30      matriks  $X_{n \times d}$  yang mana *dataset* asli dengan jumlah rekord  $n$  dan jumlah atribut  $d$ , yang mana  $d$   
31      adalah jumlah fitur pada *dataset* atau dimensi matriks tersebut. Tentunya dalam mereduksi dimensi  
32      nilai  $k$  harus lebih kecil dari pada  $d$ , yang mana  $k$  adalah dimensi dari matriks baru yang dihasilkan  
33      dari *Random Projection Perturbation* ini.

1      Jika *random projection matrix* yang digunakan dihasilkan secara acak saja, hasil dari *random*  
 2 *projection perturbation* akan terlalu merusak nilai pada data sehingga akurasi pada model yang akan  
 3 dibuat kemungkinan berkurang drastis. Cara menanggulangi hal tersebut adalah menggunakan  
 4 matriks *orthogonal* sebagai *random projection matrix*. Tetapi membuat matriks *orthogonal* yang  
 5 berdimensi tinggi memiliki kompleksitas yang tinggi sehingga memerlukan *cost* yang besar. Pada  
 6 observasi yang dilakukan Hecht-Neilsen menunjukkan bahwa “*that in a high-dimensional space,*  
 7 *vectors with random directions are almost orthogonal*” [12]. Dapat disimpulkan bahwa dalam kasus  
 8 matriks berdimensi tinggi apabila sebuah matriks dihasilkan secara acak mengikuti suatu distribusi,  
 9 matriks tersebut akan kurang lebih hampir *orthogonal*. Oleh karena itu, matriks yang dibuat untuk  
 10 *Random Projection Perturbation* cukup matriks acak yang mengikuti suatu distribusi saja.

Menurut *Johnson-Lindenstrauss Lemma*, reduksi dimensi pada matriks berdimensi tinggi minimal berdimensi  $k$ , yang mana  $k$  didefinisikan sebagai berikut.

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \quad (2.11)$$

11 Sebuah matriks yang akan diproyeksikan ke dimensi yang lebih kecil akan memiliki distorsi pada  
 12 jarak Euclidean yang dimiliki oleh titik-titik (setiap elemen dari matriks) pada bidang Euclidean  
 13 tersebut. Distorsi tersebut ditentukan oleh variabel  $\epsilon$ , yang mana  $\epsilon$  menjadi ukuran seberapa baik  
 14 proyeksi dilakukan. Semakin kecil nilai  $\epsilon$  maka semakin besar  $k$ , yang mana  $k$  adalah dimensi  
 15 minimal matriks yang dihasilkan. Semakin titik-titik pada bidang Euclidean diproyeksikan ke  
 16 dimensi lebih kecil, semakin besar kerusakan yang timbul pada jarak Euclidean titik-titik tersebut.

Persamaan berikut menyatakan rentang error yang terjadi pada *Random Projection Perturbation* dengan  $\epsilon$  (eps) yang ditentukan berada pada rentang  $(0, 1)$ .

$$(1 - \text{eps})\|u - v\|^2 < \|p(u) - p(v)\|^2 < (1 + \text{eps})\|u - v\|^2 \quad (2.12)$$

17 Pada hasil proyeksi, jarak Euclidean antara suatu titik dengan suatu titik lainnya dapat dipastikan  
 18 berada pada rentang tersebut dan tidak akan melebihi distorsi yang ditentukan.

## 19 Algoritma

20 Algoritma *Random Projection Perturbation* memiliki beberapa langkah yaitu sebagai berikut.

- 21 1. *Dataset* yang memiliki atribut sebanyak  $d$  dan rekord sebanyak  $n$  direpresentasikan dalam  
 22 bentuk matriks berukuran  $n \times d$ .
- 23 2. Tentukan nilai variabel  $\epsilon$  (epsilon) yang diinginkan dan berada pada rentang  $(0, 1)$ .
- 24 3. Hitung nilai minimal variabel  $k$  (dimensi minimal) dengan rumus berikut  $k \geq 4(\epsilon^2/2 -$   
 25  $\epsilon^3/3)^{-1} \ln n$ .
- 26 4. Tentukan nilai variabel  $k$  yang diinginkan dengan memenuhi persyaratan pada langkah ketiga  
 27 dan nilai variabel  $k$  yang dipilih harus lebih kecil dari  $d$  (dimensi *dataset* aslinya).
- 28 5. Buatlah matriks proyeksi berukuran  $d \times k$  dengan cara membuat matriks acak yang diambil  
 29 mengikuti distribusi normal dengan rata-rata bernilai 0 dan standar deviasi bernilai  $1/\sqrt{k}$ .

- 1    6. Lakukan proyeksi dengan cara mengkalikan matriks *dataset* dengan matriks proyeksi yang  
2    telah dibuat pada langkah kelima.
- 3    7. Hasil matriks yang telah diproyeksi sudah dapat langsung digunakan untuk penambangan  
4    data.

## 5    2.5 Bahasa Pemograman Python

6    Bahasa pemograman Python [13] adalah bahasa pemograman yang sudah umum digunakan untuk  
7    melakukan penambangan data. Oleh karena itu, komunitas pada bahasa pemograman Python  
8    sudah sangat besar dan banyak pihak yang membuat *library* dan *tools* yang mendukung dalam  
9    melakukan penambangan data pada bahasa pemograman ini. Tetapi bahasa pemograman Python  
10   tidak semata-mata hanya dapat digunakan untuk penambangan data, Python merupakan bahasa  
11   pemograman yang multifungsi dan banyak juga digunakan pada bidang lain selain penambangan  
12   data. Bahasa pemograman Python dapat juga digunakan untuk membuat antarmuka dengan  
13   dukungan *framework* tertentu. Selain itu, Python juga merupakan bahasa pemograman yang  
14   berorientasi objek. Berikut akan dijelaskan beberapa *library*, *framework*, dan *tools* yang berguna  
15   untuk membuat antarmuka perangkat lunak, mendukung implementasi metode *Randomization*, dan  
16   mendukung proses penambangan data.

- 17   • Kivy<sup>1</sup> adalah *framework* yang berfungsi untuk membuat antarmuka perangkat lunak. Kivy  
18   dapat membuat antarmuka perangkat lunak pada berbagai *platform* seperti Windows, OS X,  
19   Android, iOS, dan Raspberry Pi.
- 20   • Numpy dan Scipy [14] adalah *library* yang berfungsi untuk memanipulasi struktur data  
21   matriks atau melakukan operasi matematis yang berkaitan struktur data *array* atau matriks.  
22   Contohnya yang dipakai pada penelitian ini adalah membuat matriks *orthogonal* secara acak.
- 23   • Pandas [15] adalah *library* yang berfungsi untuk menganalisis dan memanipulasi data berbentuk  
24   struktur data yang lebih kompleks seperti *dataframe*. Pandas juga menyediakan fungsi untuk  
25   membaca dokumen dan mengkonversinya ke struktur data *dataframe* dan sebaliknya.
- 26   • Scikit-learn [16] adalah *library* yang berfungsi untuk melakukan proses pembelajaran mesin  
27   dan memiliki berbagai macam fungsi pembelajaran mesin yang dapat digunakan untuk  
28   penambangan data seperti teknik *k-nearest neighbors* dan *k-means*.
- 29   • Matplotlib [17], Plotly<sup>2</sup>, dan Seaborn<sup>3</sup> adalah *library* yang berfungsi untuk melakukan berbagai  
30   macam visualisasi dengan mudah. Contohnya yang dipakai pada penelitian ini adalah  
31   visualisasi dengan *scatter plot* dan *line plot*.
- 32   • Spyder<sup>4</sup> adalah sebuah perangkat lunak *integrated development environment* (IDE) yang  
33   berfungsi sebagai editor dan *compiler* bahasa pemograman Python serta berfungsi untuk

<sup>1</sup><https://kivy.org/>

<sup>2</sup><https://plotly.com/>

<sup>3</sup><https://seaborn.pydata.org/>

<sup>4</sup><https://www.spyder-ide.org/>

menampilkan berbagai macam visualisasi. Perangkat lunak ini sudah umum digunakan untuk penambangan data karena kemudahannya.

- Anaconda<sup>5</sup> adalah *tools* yang berfungsi sebagai *platform data science* dengan bahasa pemrograman Python yang memudahkan pengguna dalam manajemen *package*, *setup* seluruh perangkat lunak, *library*, IDE, dan berbagai macam keperluan lainnya yang berkaitan dengan bahasa pemrograman Python serta berbagai hal yang mendukung proses penambangan data.

---

<sup>5</sup><https://www.anaconda.com/>

1

## BAB 3

2

### ANALISIS

3 Pada bab ini akan dijelaskan analisis yang telah dilakukan terhadap *privacy preserving data mining*,  
4 metode *Randomization*, studi kasus, teknik penambangan data, dan gambaran umum perangkat  
5 lunak. Perancangan perangkat lunak akan dijelaskan melalui diagram aktivitas dan diagram kelas  
6 yang telah dibuat berdasarkan analisis yang telah dilakukan.

7 **3.1 Analisis Masalah**

8 Pada penelitian ini masalah yang ingin dicoba untuk diselesaikan adalah privasi yang terlanggar  
9 tidak sebanding dengan hasil yang didapatkan pada penambangan data. Privasi pada data yang  
10 digunakan untuk proses penambangan data mungkin saja didapatkan oleh pihak yang tidak berhak.  
11 Oleh karena itu, penelitian ini menguji salah satu solusi terhadap masalah tersebut dengan cara  
12 mengacak data yang akan ditambah dengan metode *Randomization* sehingga privasinya hilang  
13 tetapi data tersebut tetap dapat digunakan untuk penambangan data. Privasi yang perlu dijaga  
14 antara lain mengenai identitas seseorang atau hal yang dapat dikaitkan terhadap identitas seseorang.  
15 Dua buah teknik yang digunakan yaitu *Random Rotation Perturbation* dan *Random Projection*  
16 *Perturbation* akan diimplementasikan menjadi perangkat lunak dan akan diuji kualitas dari hasil  
17 teknik tersebut dengan melakukan penambangan data.

18 **3.1.1 Implementasi Metode *Randomization***

19 Teknik yang dipilih untuk diimplementasikan adalah teknik *Random Rotation Perturbation* dan  
20 *Random Projection Perturbation*. Tetapi ada kekurangan pada kedua teknik tersebut yaitu nilai  
21 setiap data yang ingin diacak harus bersifat numerik dan kedua teknik tersebut hanya menjaga  
22 jarak Euclidean antara setiap titik (sebuah baris pada *dataset*) dengan titik lainnya sehingga hanya  
23 teknik penambangan data yang bergantung pada jarak Euclidean saja yang dapat digunakan. Oleh  
24 karena itu, kedua teknik *Randomization* akan diuji dengan melakukan penambangan data dengan  
25 teknik klasifikasi yaitu *k-nearest neighbors* dan teknik *clustering* yaitu *k-means* untuk menguji coba  
26 keberhasilan kedua teknik *Randomization* dan untuk membandingkan kualitas hasil dari kedua  
27 teknik *Randomization* tersebut.

28 Perangkat lunak diharapkan dapat berfungsi untuk menerapkan teknik *Random Rotation Perturbation*  
29 terhadap *dataset* apapun dengan syarat seluruh fiturnya bersifat numerik dan menerapkan  
30 teknik *Random Projection Perturbation* terhadap *dataset* yang memenuhi persyaratan yaitu dimensi  
31 sinya cukup besar dan seluruh fiturnya bersifat numerik. Hasil akhir dari perangkat lunak akan

berbentuk sebuah dokumen berjenis *comma-separated values*. Pada penambangan data, sebuah model yang telah dibuat sangat mungkin untuk dilatih lagi dengan data yang baru. Oleh karena itu, perangkat lunak juga diharapkan dapat menyimpan matriks rotasi/proyeksi yang telah dibuat agar dapat digunakan kembali apabila ada penambahan data di waktu yang akan datang.

Metode *Randomization* akan diimplementasikan dengan bantuan bahasa pemrograman Python beserta berbagai *library* seperti Pandas, Numpy, Scikit-learn, dan Scipy. Perangkat lunak *Randomization* akan dibangun beserta antarmukanya dengan bantuan *framework* antarmuka yaitu Kivy.

### **9 Teknik *Random Rotation Perturbation***

Teknik *Random Rotation Perturbation* menggunakan matriks *special orthogonal* untuk mengacak *dataset* tanpa mengubah jarak Euclidean *dataset* tersebut. Matriks *special orthogonal* relatif sulit dibuat pada dimensi yang tinggi dan kompleksitasnya tidak kecil yaitu  $O(n^2)$  [9]. Untuk *dataset* yang sangat besar, masih dapat dilakukan pengacakan dengan teknik ini walaupun mungkin akan sedikit memakan waktu.

Walaupun seperti itu, teknik ini dapat menjamin bahwa tidak akan ada distorsi yang terjadi pada jarak Euclidean pada seluruh data atau dengan kata lain jarak Euclidean tidak berubah sama sekali. Hal ini disebabkan oleh metode yang dipakai hanya dengan merotasi seluruh titik yang merepresentasikan sebuah objek data dalam bidang Euclidean. Transformasi rotasi akan merubah nilai setiap titik pada bidang tersebut tetapi karena pada dasarnya setiap titik bergerak dengan gerakan yang sama yaitu sebuah rotasi yang sama sehingga tidak akan ada perubahan pada jarak Euclidean antara seluruh titik-titik yang ada.

Transformasi rotasi mempunyai kelemahan yaitu apabila ada titik pada bidang Euclidean yang nilainya kecil mendekati 0 maka walaupun dirotasi dengan rotasi yang berbeda-beda, bagaimanapun juga titik tersebut akan tetap bernilai kecil atau berada dekat dengan nilai 0. Hal ini berpotensi menjadi sebuah faktor yang lemah untuk melindungi privasi [7]. Oleh karena itu transformasi translasi perlu dilakukan sebelum melakukan rotasi agar titik tersebut tidak selalu mendekati nilai 0 saat dirotasi dengan berbagai macam rotasi. Transformasi translasi akan dilakukan dengan melakukan translasi yang nilainya diambil secara acak pada rentang [0, 100] sehingga translasi yang dilakukan tidak bisa diketahui oleh orang yang tidak berhak.

### **30 Teknik *Random Projection Perturbation***

Teknik *Random Projection Perturbation* didasarkan pada lemma *Johnson-Lindenstrauss* yang mengatakan bahwa sejumlah titik pada bidang Euclidean berdimensi tertentu dapat diproyeksikan ke dimensi yang lebih kecil tanpa mengubah secara signifikan jarak Euclidean antara titik-titik tersebut. Distorsi terburuk yang dapat terjadi pada jarak Euclidean setelah proyeksi dilakukan ditentukan oleh nilai variabel *epsilon* dengan Pertidaksamaan 3.1 yang menunjukkan rentang jarak Euclidean sebuah objek data dengan sebuah objek data yang lain setelah diproyeksi.

$$(1 - \text{eps})\|u - v\|^2 < \|p(u) - p(v)\|^2 < (1 + \text{eps})\|u - v\|^2 \quad (3.1)$$

1 Pertidaksamaan di atas menyatakan bahwa kuadrat dari jarak Euclidean antara dua buah titik  
 2 setelah diproyeksi tidak akan kurang dari kuadrat jarak Euclidean aslinya dikalikan  $(1 - \epsilon)$   
 3 dan tidak akan lebih dari kuadrat jarak Euclidean aslinya dikalikan  $(1 + \epsilon)$ . Lemma *Johnson-*  
 4 *Lindenstrauss* menjamin bahwa jarak Euclidean pada data setelah diproyeksi hanya akan berada  
 5 pada rentang tersebut dan bisa saja jarak Euclidean setelah diproyeksi sangat dekat dengan jarak  
 6 Euclidean aslinya karena belum tentu jarak Euclidean data setelah diproyeksi menyentuh batas  
 7 pertidaksamaan tersebut. Tetapi pertidaksamaan ini juga dapat menyatakan bahwa jarak Euclidean  
 8 setelah diproyeksi dengan aslinya tidak mungkin sama persis.

Agar pertidaksamaan sebelumnya berlaku ada persyaratan yang harus dipenuhi yaitu nilai minimal variabel  $k$  atau dengan kata lain target dimensi terkecilnya proyeksi dilakukan harus memenuhi Pertidaksamaan 3.2.

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \quad (3.2)$$

9 Variabel  $n$  pada persamaan tersebut adalah jumlah titik pada bidang Euclidean atau dengan kata  
 10 lain jumlah baris pada *dataset* yang ingin diacak. Dengan melihat persamaan tersebut maka bisa  
 11 disimpulkan bahwa dimensi minimal sebuah *dataset* diproyeksikan berbanding lurus dengan jumlah  
 12 baris pada *dataset* tersebut. Apabila sebuah *dataset* diproyeksikan ke dimensi paling minimal dan  
 13 diterapkan teknik penambangan data sehingga menghasilkan sebuah model penambangan data  
 14 dengan data latihannya adalah *dataset* tersebut maka model tersebut akan melanggar persamaan  
 15 di atas jika model tersebut dilatih kembali dengan data yang baru karena artinya bidang Euclidean  
 16 yang disebutkan tadi akan memiliki titik yang lebih banyak, nilai  $n$  yang lebih besar.

17 Oleh karena itu, berdasarkan analisis di atas teknik *Random Projection Perturbation* tidak  
 18 relevan untuk model penambangan data yang memiliki sangat banyak data tetapi berdimensi relatif  
 19 kecil dibandingkan dengan jumlah datanya. Tetapi masalah ini dapat dihindari dengan tidak  
 20 mereduksi dimensi ke dimensi yang sangat kecil atau paling minimal sehingga tidak terlalu cepat  
 21 nilai minimal variabel  $k$  meningkat sampai menyulut dimensi *dataset* setelah proyeksi. Selain itu,  
 22 jarak Euclidean setelah diproyeksi juga belum tentu menyentuh batas pertidaksamaan yang di  
 23 atas sehingga belum tentu hasil proyeksi memiliki distorsi yang sangat besar apabila nilai minimal  
 24 variabel  $k$  melebihi besar dimensi *dataset* yang telah diacak.

25 Teknik *Random Projection Perturbation* memiliki kompleksitas yang relatif kecil yaitu  $O(dkn)$   
 26 [12]. Dengan kompleksitas yang cukup rendah maka teknik ini dapat bekerja secara cepat untuk  
 27 penambangan data dengan *dataset* yang besar sekalipun. Oleh karena itu, teknik ini berpotensi  
 28 dapat berkerja dengan baik dalam lingkungan *big data* karena kompleksitas yang relatif kecil dan  
 29 dapat bekerja pada *dataset* yang sangat besar. Hal ini disebabkan oleh metode yang digunakan  
 30 oleh teknik ini adalah melakukan proyeksi dengan cara mengkalikan *dataset* yang sudah berbentuk  
 31 matriks dengan matriks proyeksi yang berupa matriks acak saja, tidak ada sifat spesial pada  
 32 matriks acak tersebut seperti teknik *Random Rotation Perturbation* sehingga pembuatan matriks  
 33 proyeksinya dapat dilakukan dengan cepat.

### 1 3.1.2 Pengujian Dengan Penambangan Data

2 Pengujian fungsional dan eksperimental akan dilakukan masing-masing untuk menguji apakah  
3 perangkat lunak berfungsi secara benar menerapkan metode *Randomization* dan menguji hasil  
4 perangkat lunak. Pengujian fungsional dilakukan dengan membandingkan nilai pada *dataset* asli  
5 dan *dataset* yang telah diacak apakah berbeda. Pengujian eksperimental dilakukan untuk melihat  
6 perbedaan apa saja yang terjadi pada *dataset*. Pengujian tersebut membandingkan beberapa  
7 informasi pada *dataset* yaitu rata-rata, standar deviasi, nilai terkecil, nilai terbesar, kuartil bawah,  
8 kuartil tengah, dan kuartil atas setiap kolom pada *dataset*. Selain itu pada pengujian eksperimental  
9 akan membandingkan model penambangan data klasifikasi dan *clustering* antara model yang dilatih  
10 menggunakan *dataset* asli dan *dataset* yang telah diacak.

11 Teknik penambangan data akan diimplementasikan dengan bantuan bahasa pemrograman Python  
12 beserta berbagai *library* seperti Pandas, Numpy, Scikit-learn, dan Scipy. Perangkat lunak untuk  
13 menerapkan penambangan data akan dijalankan dengan bantuan perangkat lunak Spyder dan Ana-  
14 conda untuk menampilkan visualisasi dan teks yang dihasilkan oleh perangkat lunak penambangan  
15 data yang berbahasa pemrograman Python.

### 16 Pengujian Dengan Teknik Penambangan Data Klasifikasi *K-nearest Neighbors*

17 Teknik penambangan data klasifikasi yang digunakan untuk menguji kualitas hasil dari metode  
18 *Randomization* adalah teknik *k-nearest neighbors*. Hasil dari metode *Randomization* diharapkan  
19 masih dapat digunakan untuk penambangan data klasifikasi dengan kualitas yang sama seperti  
20 memakai *dataset* (asli) yang tidak diacak. Pada pengujian yang akan dilakukan, perbandingan  
21 antara *dataset* asli dengan *dataset* yang telah diacak akan dilakukan dengan melakukan proses  
22 dari awal sampai akhir pembuatan model klasifikasi hingga evaluasi model tersebut. Tujuan utama  
23 pengujian adalah membandingkan akurasi (dalam memprediksi *test set*) terbaik model yang dibuat  
24 dengan *dataset* asli dan *dataset* yang telah diacak. Apabila akurasi tersebut sama atau mirip, maka  
25 teknik *Randomization* yang digunakan dapat dinyatakan berhasil dan efektif digunakan untuk  
26 *privacy preserving data mining* dengan teknik penambangan data klasifikasi menggunakan algoritma  
27 *k-nearest neighbors*.

28 *Dataset* asli dan *dataset* yang telah diacak masing-masing akan digunakan untuk membuat  
29 model *k-nearest neighbors*. Proses penambangan data *k-nearest neighbors* dilakukan dengan langkah-  
30 langkah berikut ini.

- 31 1. Membuat 10 (dapat berubah disesuaikan dengan pengujian yang dilakukan) model *k-nearest*  
32 *neighbors* dengan nilai variabel *k* sebesar 1 sampai 10 untuk mengetahui model dengan nilai  
33 variabel *k* mana yang memiliki akurasi tertinggi.
- 34 2. Melatih 10 model tersebut dengan *training set* (60 persen baris pada *dataset* diambil secara  
35 acak) yang sama.
- 36 3. Melakukan prediksi dengan 10 model tersebut terhadap *training set* yang sama.
- 37 4. Melakukan prediksi dengan 10 model tersebut terhadap *test set* (40 persen baris pada *dataset*  
38 diambil secara acak) yang sama.

- 1     5. Menghitung akurasi 10 model tersebut pada hasil prediksi *training set* dan *test set*.
- 2     6. Membuat grafik hubungan antara nilai variabel  $k$  dengan akurasi pada *training set* dan *test set*.
- 3     7. Menampilkan seluruh nilai akurasi pada setiap nilai variabel  $k$ .
- 4     8. Membuat model dengan nilai variabel  $k$  yang memiliki akurasi tertinggi lalu menghitung akurasinya pada *test set* dan menghitung waktu eksekusi saat melatih model dan melakukan prediksi.
- 5     9. Menampilkan akurasi model, waktu eksekusi untuk melatih model, dan waktu eksekusi dalam melakukan prediksi dengan model tersebut.
- 10    Proses penambangan data tersebut dilakukan dua kali masing-masing menggunakan *dataset* asli dan *dataset* yang telah diacak agar dapat dibandingkan nilai akurasi tertinggi, nilai variabel  $k$  yang memiliki akurasi tertinggi, dan waktu eksekusi pelatihan model dan prediksi dengan model tersebut. Dengan membandingkan akurasi model dapat diketahui apakah metode *Randomization* yang digunakan mempunyai kualitas yang baik atau tidak untuk digunakan dalam penambangan data klasifikasi dengan teknik *k-nearest neighbors*. Kedua model klasifikasi yang dibandingkan dapat dinyatakan sama atau mirip apabila akurasi masing-masing pada kedua model tersebut mempunyai nilai yang sama atau tidak jauh perbedaannya.

#### 18 Pengujian Dengan Teknik Penambangan Data *Clustering K-means*

19    Teknik penambangan data *clustering* yang digunakan untuk menguji kualitas hasil dari metode *Randomization* adalah teknik *k-means*. Hasil dari metode *Randomization* diharapkan masih dapat digunakan untuk penambangan data *clustering* dengan kualitas yang sama seperti memakai *dataset* (asli) yang tidak diacak. Pada pengujian yang akan dilakukan, perbandingan antara *dataset* asli dengan *dataset* yang telah diacak akan dilakukan dengan melakukan proses dari awal sampai akhir pembuatan model *clustering* hingga evaluasi model tersebut. Tujuan utama pengujian adalah membandingkan jumlah *cluster* yang ada dan anggota pada tiap *cluster* apakah sama atau mirip pada *dataset* asli dan *dataset* yang telah diacak. Apabila jumlah *cluster*-nya sama dan anggota anggota tiap *cluster* sama atau mirip, maka teknik *Randomization* yang digunakan dapat dinyatakan berhasil dan efektif digunakan untuk *privacy preserving data mining* dengan teknik penambangan data *clustering* menggunakan algoritma *k-means*.

30    Perlu diperhatikan bahwa teknik *Random Projection Perturbation* memiliki persyaratan yaitu *dataset* yang digunakan harus berdimensi cukup besar. Hal ini membuat visualisasi sulit untuk dilakukan, oleh karena itu teknik *Principal Component Analysis* digunakan untuk mereduksi dimensi *dataset* yang sangat besar dalam pengujian ini menjadi 2 dimensi saja. Teknik ini sudah terbukti relatif baik dan sudah umum digunakan untuk teknik penambangan data *clustering* dengan algoritma *k-means*.

36    *Dataset* asli dan *dataset* yang telah diacak masing-masing akan digunakan untuk membuat model *k-means*. Proses penambangan data *k-means* dilakukan dengan langkah-langkah berikut ini.

- 38    1. Membuat 9 model *k-means* dengan nilai variabel  $k$  sebesar 2 sampai 10 untuk mengetahui model dengan nilai variabel  $k$  mana yang memiliki *cluster* terbaik.

- 1      2. Melatih 9 model tersebut dengan *dataset* yang sama.
  - 2      3. Menghitung nilai *Sum of Squared Error* dan *Silhouette Score* pada 9 model tersebut.
  - 3      4. Membuat grafik hubungan antara nilai variabel *k* dengan nilai *Sum of Squared Error*.
  - 4      5. Membuat grafik hubungan antara nilai variabel *k* dengan nilai *Silhouette Score*.
  - 5      6. Menampilkan seluruh nilai *Sum of Squared Error* pada setiap nilai variabel *k*.
  - 6      7. Menampilkan seluruh nilai *Silhouette Score* pada setiap nilai variabel *k*.
  - 7      8. Menampilkan nilai *Silhouette Score* tertinggi dan nilai variabel *k*-nya.
  - 8      9. Membuat model dengan nilai variabel *k* yang memiliki *Silhouette Score* tertinggi lalu menghitung *Silhouette Score*-nya dan menghitung waktu eksekusi saat melatih model.
  - 10     10. Menampilkan visualisasi *cluster* pada model dengan *Scatter Plot*, *Silhouette Score* model, dan waktu eksekusi untuk melatih model tersebut.
- 12 Proses penambangan data tersebut dilakukan dua kali masing-masing menggunakan *dataset* asli dan  
13 *dataset* yang telah diacak agar dapat dibandingkan visualisasi *cluster* pada model yang terbaik, nilai  
14 variabel *k* yang dimiliki oleh model yang terbaik, dan informasi lainnya yaitu *Sum of Squared Error*,  
15 *Silhouette Score*, dan waktu pelatihan model. Dalam membandingkan antara model *clustering* yang  
16 dilatih dengan *dataset* asli dan *dataset* yang telah diacak, metode *Adjusted Rand Index* digunakan  
17 untuk mengukur kemiripan antara kedua model *clustering* tersebut. Dengan menggunakan metode  
18 *Adjusted Rand Index* dapat diketahui apakah metode *Randomization* yang digunakan mempunyai  
19 kualitas yang baik atau tidak untuk digunakan dalam penambangan data *clustering* dengan teknik  
20 *k-means*. Kedua model *clustering* yang dibandingkan dapat dinyatakan sama atau mirip apabila  
21 nilai *Adjusted Rand Index* pada kedua model *clustering* tersebut mendekati angka 1.

## 22    3.2 Studi Kasus

23 Dalam rangka untuk lebih memahami bagaimana cara kerja teknik *Random Rotation Perturbation*  
24 dan *Random Projection Perturbation*, studi kasus akan dilakukan dan akan menggunakan *dataset*  
25 *iris*. Tetapi untuk memudahkan perhitungan pada studi kasus ini data yang dipakai hanya sebagian  
26 kecil saja dari seluruh data pada *dataset iris* yaitu 9 baris pertama. *Dataset* tersebut dapat dilihat  
27 pada Tabel 3.1. *Dataset iris* adalah *dataset* yang berisi data tentang korelasi antara ukuran bunga  
28 dengan spesiesnya. *Dataset* ini memiliki empat buah fitur dan satu buah label. Fitur-fitur pada  
29 *dataset iris* adalah kolom *sepal\_length*, *sepal\_width*, *petal\_length*, dan *petal\_width*. Label pada  
30 *dataset iris* adalah kolom *species*.

### 31    3.2.1 *Random Rotation Perturbation*

32 Berikut langkah-langkah teknik *Random Rotation Perturbation* yang diaplikasikan pada *dataset*  
33 *iris* pada Tabel 3.1.

Tabel 3.1: Tabel *dataset iris* yang digunakan sebagai contoh kasus

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

1. Fitur-fitur pada *dataset* tersebut yang berbentuk tabel akan direpresentasikan sebagai matriks. Labelnya tidak diikutsertakan. Matriks tersebut dapat dilihat pada Matriks 3.3 dan selanjutnya akan disebut sebagai matriks *dataset*.

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & 3.1 & 1.5 & 0.2 \\ 5 & 3.6 & 1.4 & 0.2 \\ 5.4 & 3.9 & 1.7 & 0.4 \\ 4.6 & 3.4 & 1.4 & 0.3 \\ 5 & 3.4 & 1.5 & 0.2 \\ 4.4 & 2.9 & 1.4 & 0.2 \end{bmatrix}_{9 \times 4} \quad (3.3)$$

2. Membuat matriks translasi yang diambil mengikuti distribusi *uniform* dengan rentang [0,100] dengan dimensi sesuai dimensi matriks *dataset*. Hasilnya dapat dilihat pada Matriks 3.4.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 71 & 93 & 77 & 27 & 1 \end{bmatrix}_{5 \times 5} \quad (3.4)$$

3. Untuk keperluan translasi, matriks *dataset* yang dapat dilihat pada Matriks 3.3 ditambahkan

sebuah kolom dengan nilai 1 pada setiap barisnya. Hasilnya dapat dilihat pada Matriks 3.5.

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 & 1 \\ 4.9 & 3 & 1.4 & 0.2 & 1 \\ 4.7 & 3.2 & 1.3 & 0.2 & 1 \\ 4.6 & 3.1 & 1.5 & 0.2 & 1 \\ 5 & 3.6 & 1.4 & 0.2 & 1 \\ 5.4 & 3.9 & 1.7 & 0.4 & 1 \\ 4.6 & 3.4 & 1.4 & 0.3 & 1 \\ 5 & 3.4 & 1.5 & 0.2 & 1 \\ 4.4 & 2.9 & 1.4 & 0.2 & 1 \end{bmatrix}_{9 \times 5} \quad (3.5)$$

- Dilakukan transformasi translasi terhadap matriks *dataset* dengan matriks translasi yang telah dibuat pada langkah sebelumnya dengan cara mengkalikan matriks *dataset* yang dapat dilihat pada Matriks 3.5 dengan matriks translasi yang dapat dilihat pada Matriks 3.4. Hasil translasi pada matriks *dataset* dapat dilihat pada Matriks 3.6. Dapat dilihat kolom terakhir adalah kolom yang harus dibuang karena kolom tersebut ada hanya untuk melakukan transformasi translasi yang telah dilakukan.

$$\begin{bmatrix} 76.1 & 96.5 & 78.4 & 27.2 & 1 \\ 75.9 & 96 & 78.4 & 27.2 & 1 \\ 75.7 & 96.2 & 78.3 & 27.2 & 1 \\ 75.6 & 96.1 & 78.5 & 27.2 & 1 \\ 76 & 96.6 & 78.4 & 27.2 & 1 \\ 76.4 & 96.9 & 78.7 & 27.4 & 1 \\ 75.6 & 96.4 & 78.4 & 27.3 & 1 \\ 76 & 96.4 & 78.5 & 27.2 & 1 \\ 75.4 & 95.9 & 78.4 & 27.2 & 1 \end{bmatrix}_{9 \times 5} \quad (3.6)$$

- Berikutnya matriks rotasi dibuat dengan cara membuat matriks *special orthogonal* yang berdimensi sesuai dimensi matriks *dataset*. Matriks rotasi berikut dibuat dengan menggunakan *library Scipy*<sup>1</sup> pada bahasa pemrograman Python. Hasilnya dapat dilihat pada Matriks 3.7.

$$\begin{bmatrix} -0.45126938 & -0.70425922 & 0.32389616 & 0.44211556 \\ -0.43989334 & 0.70728617 & 0.39249528 & 0.39011226 \\ -0.17797534 & 0.06110969 & -0.83056872 & 0.52416218 \\ 0.75576092 & 0.00555185 & 0.22626167 & 0.61449187 \end{bmatrix}_{4 \times 4} \quad (3.7)$$

- Dilakukan transformasi rotasi terhadap matriks *dataset* dengan matriks rotasi yang telah dibuat pada langkah sebelumnya dengan cara mengkalikan matriks *dataset* yang dapat dilihat pada Matriks 3.6 dengan matriks rotasi yang dapat dilihat pada Matriks 3.7. Hasil rotasi dapat dilihat pada Matriks 3.8. Hasil dari penerapan teknik *Random Rotation Perturbation*

---

<sup>1</sup>[http://scipy.github.io/devdocs/generated/scipy.stats.special\\_ortho\\_group.html](http://scipy.github.io/devdocs/generated/scipy.stats.special_ortho_group.html)

pada *dataset iris* ini sudah dapat langsung digunakan untuk penambangan data.

$$\begin{bmatrix} -70.18787676 & 19.60099878 & 3.56202207 & 129.09932098 \\ -69.87767621 & 19.38820754 & 3.3009952 & 128.81584174 \\ -69.85760347 & 19.66440565 & 3.3977719 & 128.75302486 \\ -69.80408227 & 19.67632489 & 3.16001901 & 128.77463452 \\ -70.18673916 & 19.74215332 & 3.56888198 & 129.09412065 \\ -70.40145533 & 19.69207876 & 3.61227075 & 129.66814758 \\ -69.84267664 & 19.88295496 & 3.38345063 & 128.90070116 \\ -70.11655802 & 19.60680705 & 3.40732606 & 129.06851442 \\ -69.60805219 & 19.66960853 & 3.09979759 & 128.55577273 \end{bmatrix}_{9 \times 4} \quad (3.8)$$

### ***3.2.2 Random Projection Perturbation***

- 2 Teknik *Random Projection Perturbation* memiliki persyaratan pada *dataset* agar teknik ini menghasilkan hasil yang baik yaitu *dataset* tersebut harus memiliki dimensi yang cukup besar. Sebetulnya *dataset iris* yang dapat dilihat pada Tabel 3.1 tidak memenuhi persyaratan untuk mendapatkan hasil yang baik, tetapi untuk memudahkan perhitungan pada studi kasus ini data yang dipakai adalah *dataset* tersebut saja yang memiliki dimensi yang kecil dan hanya sebagian kecil saja data yang dipakai. Dalam menghitung nilai minimal variabel  $k$  juga, pada studi kasus ini menggunakan jumlah rekord dan atribut yang tidak sesuai dengan *dataset iris* untuk keperluan kemudahan dalam melakukan studi kasus dan juga agar memenuhi persyaratan teknik *Random Projection Perturbation*.
- 10 Jumlah rekordnya adalah 1000 dan jumlah atributnya adalah 500.
- 11 Berikut langkah-langkah teknik *Random Projection Perturbation* yang diaplikasikan pada *dataset iris* pada Tabel 3.1.

1. Fitur-fitur pada *dataset* tersebut yang berbentuk tabel akan direpresentasikan sebagai matriks. Labelnya tidak diikutsertakan. Matriks tersebut dapat dilihat pada Matriks 3.9 dan selanjutnya akan disebut sebagai matriks *dataset*.

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & 3.1 & 1.5 & 0.2 \\ 5 & 3.6 & 1.4 & 0.2 \\ 5.4 & 3.9 & 1.7 & 0.4 \\ 4.6 & 3.4 & 1.4 & 0.3 \\ 5 & 3.4 & 1.5 & 0.2 \\ 4.4 & 2.9 & 1.4 & 0.2 \end{bmatrix}_{9 \times 4} \quad (3.9)$$

2. Ditentukan nilai  $\epsilon$  (*epsilon*) yang diinginkan adalah 0.5. Dengan menggunakan Pertidaksamaan 3.1 dapat dihitung untuk melihat rentang jarak Euclidean antara sebuah titik yang merepresentasikan sebuah objek data dan titik lainnya dari *dataset* yang telah diproyeksi. Contoh yang akan diberikan berikut menguji rentang jarak Euclidean antara baris ke-6 dan

ke-9 setelah diproyeksi.

$$\begin{aligned} \|u - v\|^2 &= (5.4 - 4.4)^2 + (3.9 - 2.9)^2 + (1.7 - 1.4)^2 + (0.4 - 0.2)^2 \\ &= 2.13 \end{aligned}$$

$$\begin{aligned} (1 - \text{eps})\|u - v\|^2 &< \|p(u) - p(v)\|^2 < (1 + \text{eps})\|u - v\|^2 \\ (1 - 0.5)2.13 &< \|p(u) - p(v)\|^2 < (1 + 0.5)2.13 \\ 1.065 &< \|p(u) - p(v)\|^2 < 3.195 \end{aligned}$$

3. Nilai minimal variabel  $k$  (dimensi minimal) dihitung dengan Pertidaksamaan 3.2.

$$\begin{aligned} k &\geq \frac{4 \ln n}{\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}} \\ &\geq \frac{4 \ln 1000}{\frac{0.5^2}{2} - \frac{0.5^3}{3}} \\ &\geq \frac{27.63}{0.125 - 0.041666} \\ &\geq 331.57 \end{aligned}$$

- <sup>1</sup> 4. Nilai variabel  $k$  dipilih sesuai keinginan, dalam kasus ini dipilih nilai  $k$  sebesar 332. Tetapi  
<sup>2</sup> untuk keperluan kemudahan dalam melakukan studi kasus, nilai  $k$  yang digunakan adalah 3  
<sup>3</sup> saja.
5. Membuat matriks proyeksi berukuran  $d \times k$  dengan cara membuat matriks acak yang diambil mengikuti distribusi normal dengan rata-rata bernilai 0 dan standar deviasi bernilai  $1/\sqrt{k}$ . Untuk keperluan kemudahan dalam melakukan studi kasus, *dataset iris* akan direduksi dimensinya menjadi 3 dimensi. Hasilnya dapat dilihat pada Matriks 3.10.

$$\left[ \begin{array}{ccc} 0.11483014 & -0.10167359 & 0.06652355 \\ 0.0638684 & -0.1499892 & 0.10146435 \\ -0.10429573 & 0.03839861 & 0.04955419 \\ -0.0315941 & -0.06905021 & -0.17782438 \end{array} \right]_{4 \times 3} \quad (3.10)$$

6. Dilakukan proyeksi dengan cara mengkalikan matriks *dataset* yang dapat dilihat pada Matriks 3.9 dengan matriks proyeksi yang telah dibuat pada langkah sebelumnya dan dapat dilihat pada Matriks 3.10. Hasil proyeksi dapat dilihat pada Matriks 3.11. Hasil dari penerapan teknik *Random Projection Perturbation* pada *dataset iris* ini sudah dapat langsung digunakan

untuk penambangan data.

$$\begin{bmatrix} 0.65684027 & -1.0035495 & 0.72820632 \\ 0.60194004 & -0.90822018 & 0.66416944 \\ 0.60217727 & -0.92172316 & 0.66620218 \\ 0.56344827 & -0.88887716 & 0.65931422 \\ 0.6517441 & -1.00838106 & 0.7317004 \\ 0.67922914 & -1.09633771 & 0.76805051 \\ 0.58987895 & -0.9446188 & 0.66701567 \\ 0.62854085 & -0.97454336 & 0.71636295 \\ 0.53813813 & -0.84238446 & 0.62076123 \end{bmatrix}_{9 \times 3} \quad (3.11)$$

### **1 3.3 Gambaran Umum Perangkat Lunak**

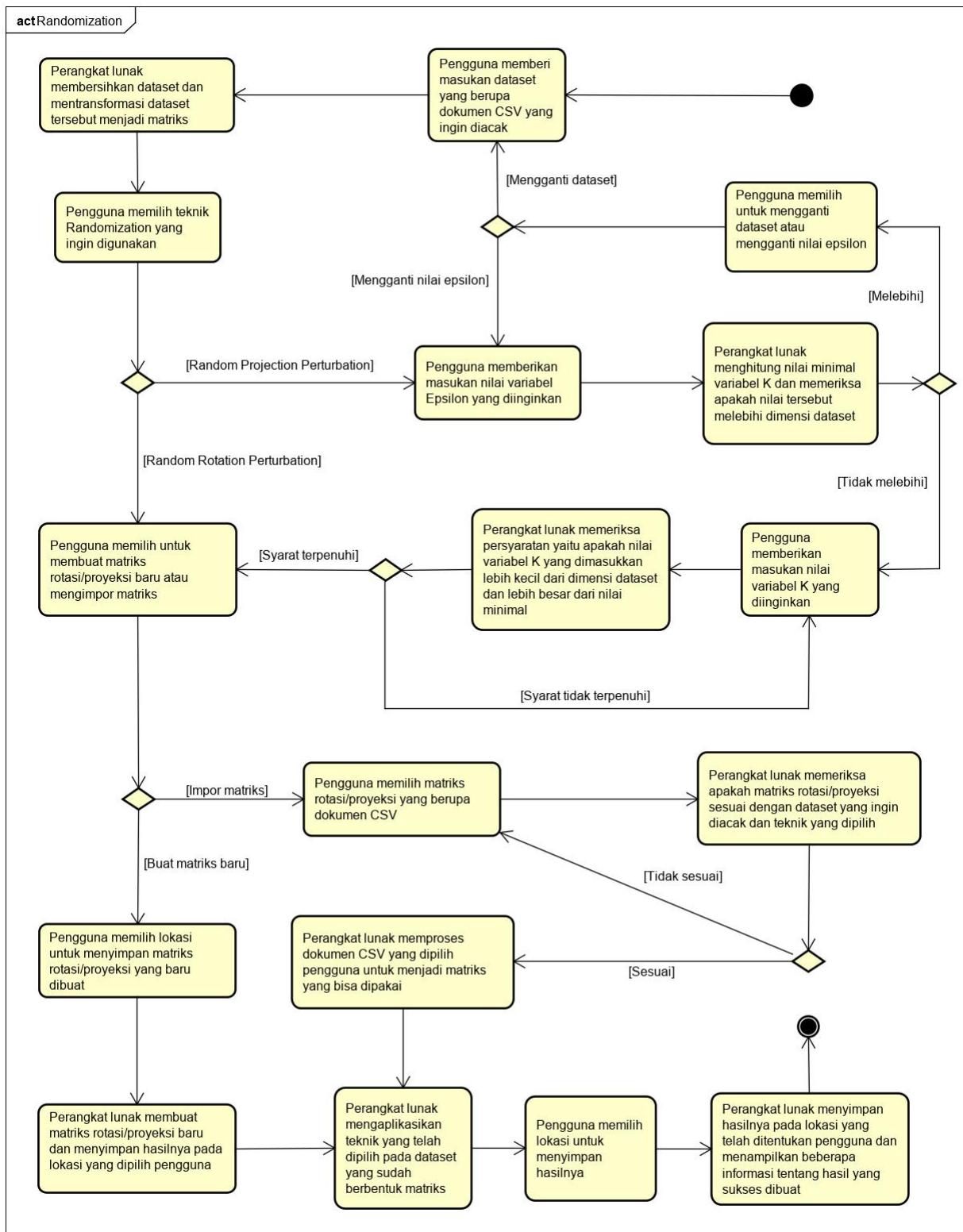
2 Ada beberapa persyaratan yang harus dipenuhi pada implementasi perangkat lunak seperti alur,  
 3 masukan, keluaran, dan implementasi fungsi perangkat lunak. Hal-hal tersebut harus disesuaikan  
 4 dengan kebutuhan dan tujuan dibuatnya perangkat lunak ini. Oleh karena itu, perlu adanya  
 5 perancangan terlebih dahulu untuk menjadi gambaran bagaimana perangkat lunak yang akan  
 6 dibuat akan berfungsi. Diagram aktivitas dibuat untuk menunjukkan bagaimana perangkat lunak  
 7 bekerja dengan alur yang baik serta masukan dan keluaran yang tepat. Perancangan kelas dibuat  
 8 untuk menunjukkan bagaimana perangkat lunak diimplementasikan dan berfungsi dengan baik.

9 Perangkat lunak yang mengimplementasikan metode *Randomization* akan mempunyai sebuah  
 10 masukan yaitu *dataset* berbentuk dokumen berjenis *comma-separated values*. Lalu perangkat  
 11 lunak akan mengimplementasikan metode *Randomization* terhadap *dataset* tersebut dan hasil  
 12 keluarannya adalah *dataset* yang telah diacak berbentuk dokumen berjenis *comma-separated values*.  
 13 Pada subbab ini akan ditunjukkan gambaran umum perangkat lunak yang akan dijelaskan dengan  
 14 diagram aktivitas dan diagram kelas.

#### **15 3.3.1 Diagram Aktivitas**

16 Perangkat lunak *Randomization* adalah perangkat lunak yang digunakan untuk memodifikasi data  
 17 dengan metode *Randomization*. Perangkat lunak ini akan memiliki fungsi untuk mengubah nilai  
 18 setiap data yang dimasukan agar privasinya terjaga tetapi masih dapat dilakukan penambangan  
 19 data. Algoritma *Random Rotation Perturbation* dan *Random Projection Perturbation* akan diimple-  
 20 mentasikan pada perangkat lunak ini untuk fungsi utama yaitu mengubah nilai pada setiap data.  
 21 Dengan mempertimbangkan studi literatur, analisis masalah, dan studi kasus yang telah dilakukan  
 22 pada kedua teknik tersebut, perangkat lunak akan memiliki berbagai pilihan dan parameter yang  
 23 pengguna harus masukan dan juga ada beberapa persyaratan atau batasan agar perangkat lunak ini  
 24 berjalan dengan semestinya. Diagram aktivitas untuk perangkat lunak *Randomization* dapat dilihat  
 25 pada Gambar 3.1. Berikut adalah penjelasan langkah-langkah pada diagram aktivitas tersebut.

- 26 1. Pengguna memberikan masukan berupa *dataset* yang berupa dokumen berjenis *comma-*  
 27 *separated values* (CSV). Dokumen ini harus berisi tiap rekord pada barisnya dan tiap fitur  
 28 pada kolomnya. Selain itu, pada baris pertama dalam dokumen tersebut harus berupa nama

Gambar 3.1: Diagram aktivitas perangkat lunak *Randomization*

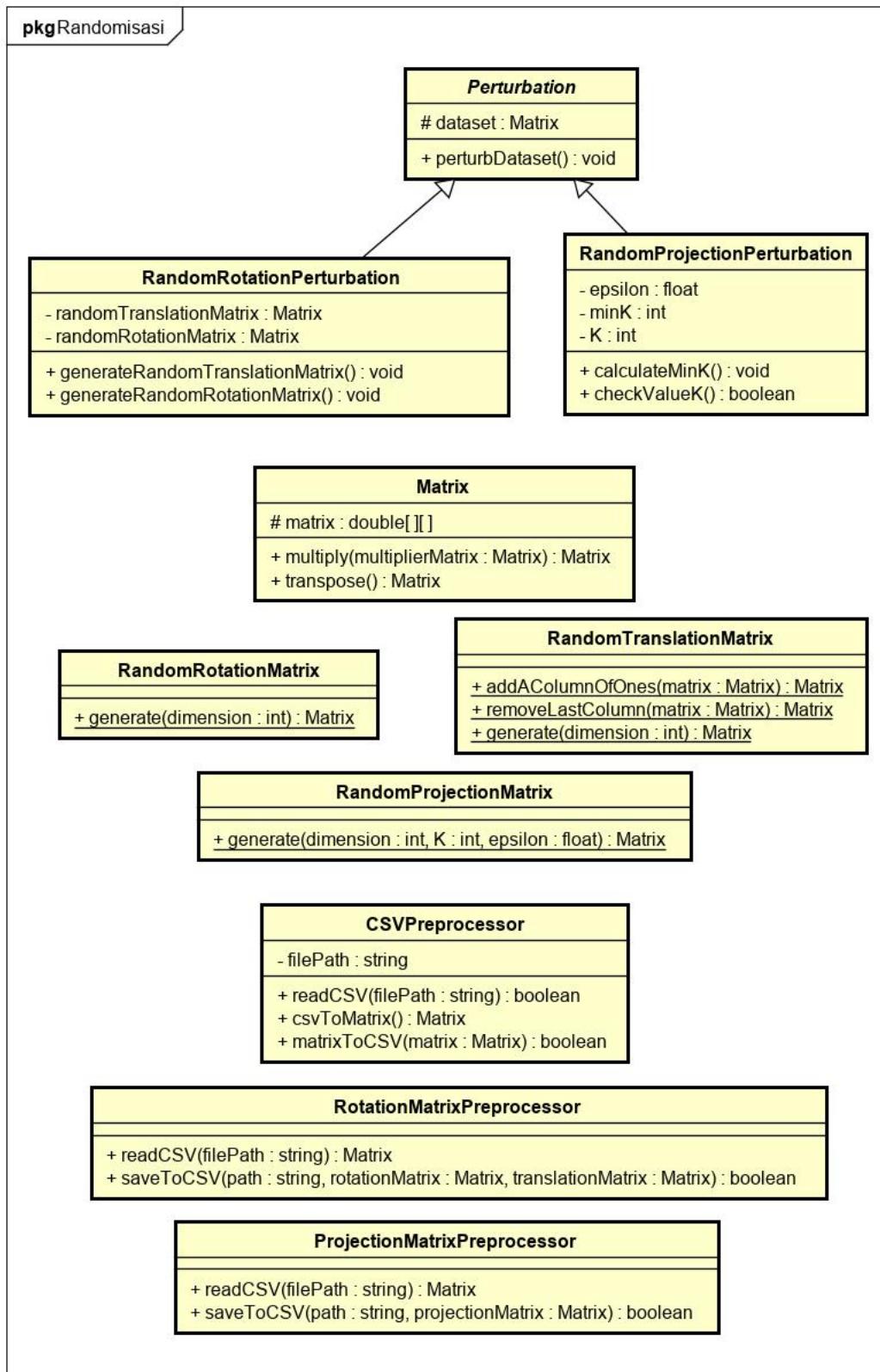
- 1 setiap fitur pada setiap kolomnya.
- 2 2. Perangkat lunak akan membersihkan dokumen yang berisi *dataset* tersebut dan mentransfor-  
3 masi *dataset* tersebut menjadi sebuah matriks. Matriks tersebut akan berisi nilai-nilai pada  
4 *dataset* saja tanpa nama kolom.
- 5 3. Pengguna memilih teknik *Randomization* yang ingin digunakan antara *Random Rotation*  
6 *Perturbation* dan *Random Projection Perturbation*. Jika *Random Projection Perturbation*  
7 yang dipilih maka akan ada beberapa langkah yang harus dipenuhi yaitu sebagai berikut.
- 8 (a) Pengguna memberi masukan nilai variabel *epsilon* yang diinginkan. Variabel ini akan  
9 menentukan distorsi terburuk yang dapat terjadi pada hasil proyeksi dan menentukan  
10 nilai minimal variabel *k*.
- 11 (b) Perangkat lunak menghitung nilai minimal variabel *k* lalu menampilkannya kepada  
12 pengguna dan memeriksa sebuah persyaratan yaitu apakah nilai minimal variabel *k*  
13 tersebut tidak melebihi dimensi *dataset*.
- 14 (c) Jika persyaratan tidak terpenuhi maka pengguna harus memilih untuk mengganti *dataset*  
15 atau mengganti nilai variabel *epsilon*.
- 16 (d) Jika persyaratan terpenuhi maka berikutnya pengguna memberikan masukan nilai  
17 variabel *k* yang diinginkan. Nilai yang diberikan pengguna harus lebih dari nilai minimal  
18 variabel *k* yang sudah dihitung oleh perangkat lunak pada langkah sebelumnya.
- 19 (e) Perangkat lunak memeriksa persyaratan teknik yaitu nilai variabel *k* yang diberikan  
20 harus lebih kecil dari dimensi *dataset* dan lebih besar dari nilai minimal variabel *k*.
- 21 (f) Jika persyaratan tidak terpenuhi maka pengguna harus mengganti nilai variabel *k*  
22 kembali.
- 23 4. Pengguna memilih untuk membuat matriks rotasi/proyeksi baru sesuai teknik yang dipilih  
24 atau pengguna dapat mengimpor matriks yang telah dibuat dan disimpan sebelumnya.
- 25 5. Berikut ini langkah-langkah tambahan yang harus dilakukan pengguna apabila pengguna  
26 memilih untuk membuat matriks rotasi/proyeksi baru.
- 27 (a) Pengguna memilih lokasi direktori pada komputer pengguna sebagai lokasi untuk me-  
28 nyimpan matriks rotasi/proyeksi yang akan dibuat.
- 29 (b) Perangkat lunak membuat matriks rotasi/proyeksi baru dan menyimpan hasilnya pada  
30 lokasi yang telah dipilih pengguna dalam bentuk dokumen berjenis CSV.
- 31 6. Berikut ini langkah-langkah tambahan yang harus dilakukan pengguna apabila pengguna  
32 memilih untuk mengimpor matriks rotasi/proyeksi yang pernah dibuat sebelumnya.
- 33 (a) Pengguna memilih matriks rotasi atau proyeksi sesuai teknik yang dipilih berupa dokumen  
34 berjenis CSV pada sebuah direktori komputer pengguna yang dipilih.
- 35 (b) Perangkat lunak memeriksa dokumen CSV yang dipilih berisi matriks rotasi/proyeksi  
36 yang sesuai dengan *dataset* dan teknik *Randomization* yang dipilih.

- 1       (c) Jika tidak sesuai maka pengguna harus memilih kembali matriks rotasi/proyeksi dengan  
2       benar.
- 3       (d) Jika sudah sesuai dan memenuhi persyaratan maka perangkat lunak akan memproses  
4       dokumen CSV yang dipilih pengguna untuk menjadi matriks rotasi/proyeksi yang dapat  
5       dipakai untuk teknik *Randomization* yang dipilih.
- 6      7. Perangkat lunak mengaplikasikan teknik yang telah dipilih pada *dataset* yang sudah berbentuk  
7       matriks dengan memakai matriks rotasi/proyeksi yang telah dipilih.
- 8      8. Pengguna memilih lokasi direktori pada komputer pengguna untuk menyimpan hasil pengacak-  
9       an yang sudah dilakukan menggunakan teknik yang dipilih. Hasilnya adalah sebuah dokumen  
10       *comma-separated values* yang berisi *dataset* yang telah diacak.
- 11     9. Perangkat lunak menyimpan hasilnya pada lokasi yang telah ditentukan oleh pengguna dalam  
12       bentuk dokumen berjenis *comma-separated values* dan menampilkan beberapa informasi  
13       tentang hasil yang sukses dibuat.

14     **3.3.2 Diagram Kelas**

15   Perancangan diagram kelas didasarkan pada analisis terhadap algoritma yang ingin diimplementa-  
16   sikan yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation*, serta berdasarkan  
17   pada diagram aktivitas yang telah dibuat, dan dengan mempertimbangkan studi literatur, analisis  
18   masalah, dan studi kasus yang telah dilakukan pada kedua algoritma *Randomization* yang ingin  
19   diimplementasikan. Detail dari diagram kelas perangkat lunak *Randomization* pada Gambar 3.2  
20   adalah sebagai berikut.

- 21     • Kelas *Perturbation* adalah kelas abstrak yang akan di-*extend* oleh kelas yang mengimplemen-  
22       tasikan algoritma *Random Rotation Perturbation* dan *Random Projection Perturbation*. Kelas  
23       ini mempunyai sebuah atribut yaitu *dataset* yang berfungsi untuk menyimpan *dataset* yang  
24       ingin diacak. Ada sebuah fungsi pada kelas ini yaitu *perturbDataset* yang berfungsi untuk  
25       menerapkan teknik *Randomization* pada *dataset* yang ingin diacak dengan kata lain atribut  
26       *dataset*.
- 27     • Kelas *RandomRotationPerturbation* adalah kelas turunan dari kelas *Perturbation* yang ber-  
28       tujuan untuk mengimplementasikan algoritma *Random Rotation Perturbation*. Kelas ini  
29       memiliki dua tambahan atribut yaitu *randomTranslationMatrix* dan *randomRotationMatrix*  
30       yang masing-masing memiliki fungsi untuk menyimpan matriks translasi dan matriks rotasi.  
31       Ada dua buah fungsi tambahan juga pada kelas ini yaitu *generateRandomTranslationMatrix*  
32       dan *generateRandomRotationMatrix* yang masing-masing memiliki fungsi untuk membuat  
33       matriks translasi dan matriks rotasi baru.
- 34     • Kelas *RandomProjectionPerturbation* adalah kelas turunan dari kelas *Perturbation* yang  
35       bertujuan untuk mengimplementasikan algoritma *Random Projection Perturbation*. Kelas ini  
36       memiliki tiga tambahan atribut yaitu *epsilon*, *minK*, dan *K* yang masing-masing memiliki  
37       fungsi untuk menyimpan nilai variabel *epsilon*, nilai minimal variabel *k*, dan nilai variabel  
38       *k* yang akan dipakai untuk menerapkan algoritma *Random Projection Perturbation*. Ada

Gambar 3.2: Diagram kelas perangkat lunak *Randomization*

1 dua buah fungsi tambahan juga pada kelas ini yaitu *calculateMinK* dan *checkValueK* yang  
2 masing-masing memiliki fungsi untuk menghitung nilai minimal variabel *k* dan memeriksa  
3 persyaratan pada nilai atribut *K*.

- 4 • Kelas *Matrix* adalah kelas untuk merepresentasikan matriks dan menyimpan nilai-nilai pada  
5 setiap elemen matriks. Kelas ini memiliki sebuah atribut yaitu *matrix* yang berfungsi untuk  
6 menyimpan setiap elemen matriks yang ada. Kelas ini juga memiliki dua buah fungsi yaitu  
7 *multiply* dan *transpose* yang masing-masing berfungsi untuk melakukan operasi perkalian  
8 dan transpose matriks yang akan digunakan untuk implementasi kedua algoritma teknik  
9 *Randomization*.
- 10 • Kelas *RandomTranslationMatrix* adalah kelas yang mempunyai tujuan utama untuk membuat  
11 matriks translasi secara acak. Kelas ini adalah kelas statis sehingga tidak bisa diinstansiasi  
12 dan hanya memiliki atribut atau fungsi statis saja. Ada tiga buah fungsi statis yaitu *addACo-*  
13 *lumnOfOnes*, *removeLastColumn*, dan *generate*. Fungsi *addAColumnOfOnes* memiliki fungsi  
14 untuk menambahkan sebuah kolom yang pada setiap baris memiliki nilai berupa angka 1  
15 kepada sebuah matriks. Fungsi *removeLastColumn* memiliki fungsi untuk menghapus kolom  
16 terakhir pada suatu matriks. Fungsi *generate* adalah fungsi yang bertujuan untuk membuat  
17 matriks translasi baru secara acak.
- 18 • Kelas *RandomRotationMatrix* adalah kelas yang mempunyai tujuan utama untuk membuat  
19 matriks rotasi secara acak. Kelas ini adalah kelas statis sehingga tidak bisa diinstansiasi dan  
20 hanya memiliki atribut atau fungsi statis saja. Hanya ada sebuah fungsi pada kelas ini yaitu  
21 *generate* yang memiliki fungsi untuk membuat matriks rotasi baru secara acak.
- 22 • Kelas *RandomProjectionMatrix* adalah kelas yang mempunyai tujuan utama untuk membuat  
23 matriks proyeksi secara acak. Kelas ini adalah kelas statis sehingga tidak bisa diinstansiasi dan  
24 hanya memiliki atribut atau fungsi statis saja. Hanya ada sebuah fungsi pada kelas ini yaitu  
25 *generate* yang memiliki fungsi untuk membuat matriks proyeksi baru secara acak.
- 26 • Kelas *CSVPreprocessor* adalah kelas untuk menangani masukan *dataset* berupa dokumen  
27 *comma-separated values* yang akan direpresentasikan menjadi matriks. Kelas ini berguna untuk  
28 mengkonversi dokumen berjenis *comma-separated values* menjadi matriks dan sebaliknya.  
29 Hanya ada sebuah atribut pada kelas ini yaitu *filePath* yang berfungsi untuk menyimpan lokasi  
30 dokumen yang ingin diproses menjadi matriks. Ada tiga buah fungsi pada kelas ini yaitu  
31 *readCSV*, *csvToMatrix*, dan *matrixToCSV* yang masing-masing berfungsi untuk membaca  
32 dokumen berjenis *comma-separated values*, mengkonversi dokumen *comma-separated values*  
33 menjadi matriks, dan mengkonversi matriks menjadi dokumen *comma-separated values*.
- 34 • Kelas *RotationMatrixPreprocessor* adalah kelas untuk menangani masukan matriks rotasi  
35 yang digabung dengan matriks translasi berupa sebuah dokumen *comma-separated values*  
36 yang akan dikonversi menjadi dua buah matriks (rotasi dan translasi) dan sebaliknya. Ada  
37 dua buah fungsi pada kelas ini yaitu *readCSV* dan *saveToCSV*. Fungsi *readCSV* berfungsi  
38 untuk membaca dokumen berjenis *comma-separated values* untuk dikonversi menjadi matriks  
39 rotasi dan matriks translasi. Fungsi *SaveToCSV* berfungsi untuk mengkonversi matriks rotasi  
40 dan matriks translasi menjadi sebuah dokumen berjenis *comma-separated values*.

- 1     ● Kelas *ProjectionMatrixPreprocessor* adalah kelas untuk menangani masukan matriks proyeksi  
2       berupa sebuah dokumen *comma-separated values* yang akan dikonversi menjadi sebuah matriks.  
3       Ada dua buah fungsi pada kelas ini yaitu *readCSV* dan *saveToCSV*. Fungsi *readCSV* berfungsi  
4       untuk membaca dokumen berjenis *comma-separated values* untuk dikonversi menjadi matriks  
5       proyeksi. Fungsi *SaveToCSV* berfungsi untuk mengkonversi matriks proyeksi menjadi sebuah  
6       dokumen berjenis *comma-separated values*.



1

## BAB 4

2

### PERANCANGAN PERANGKAT LUNAK

3 Pada bab ini akan dijabarkan perancangan perangkat lunak untuk penelitian ini. Perancangan  
4 perangkat lunak tersebut meliputi perancangan antarmuka dan perancangan kelas untuk penelitian  
5 ini.

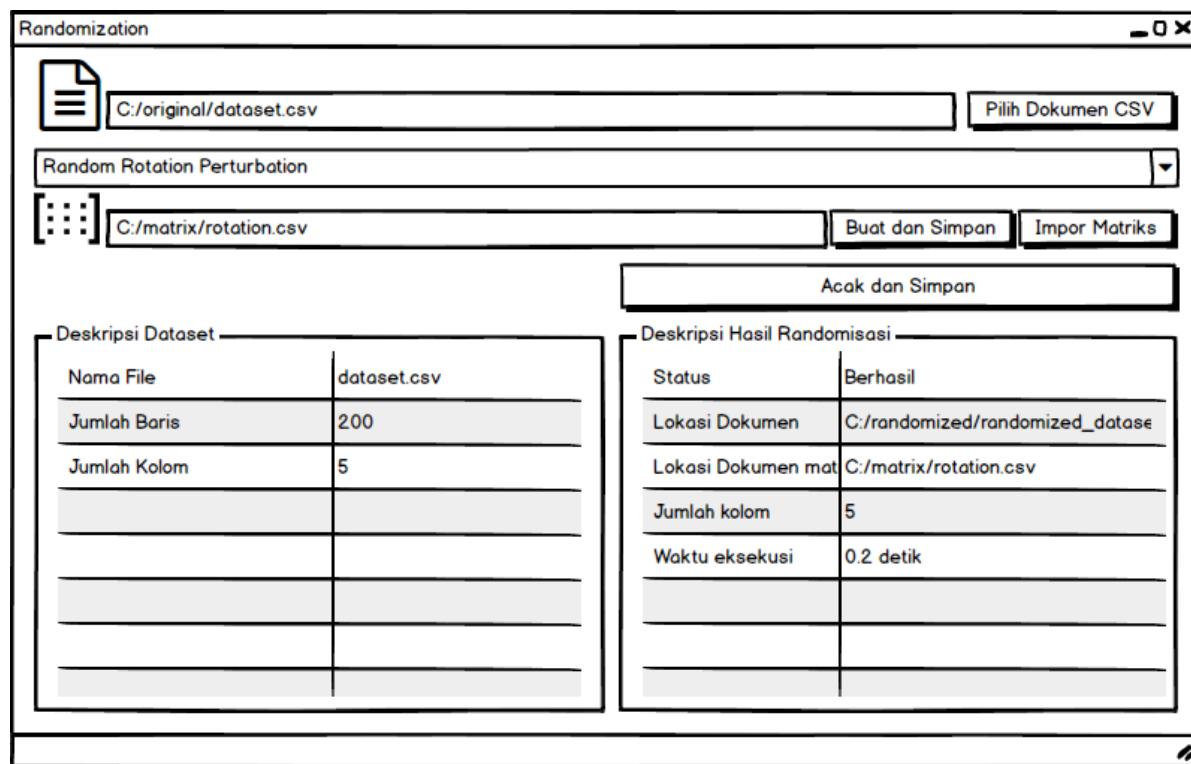
#### 6 4.1 Perancangan Antarmuka

7 Perancangan antarmuka yang dibuat disesuaikan dengan diagram kelas dan diagram aktivitas  
8 yang dibuat sesuai analisis perangkat lunak dilakukan. Rancangan antarmuka dapat dilihat pada  
9 Gambar 4.1. Pada bagian atas antarmuka terdapat sebuah kolom untuk memasukkan dokumen  
10 berjenis *comma-separated values* yang ingin diacak. Pertama-tama pengguna wajib untuk mengisi  
11 kolom tersebut apapun teknik *Randomization* yang akan digunakan nanti. Setelah pengguna  
12 memasukkan dokumen yang diinginkan, perangkat lunak akan menampilkan berbagai informasi  
13 mengenai *dataset* yang ada di dokumen tersebut seperti ukuran dokumen, nama dokumen, dan  
14 jumlah kolom. Deskripsi ini bertujuan untuk memberitahukan pengguna bahwa dokumen yang  
15 dimasukkan telah benar dan bagaimana sifat dari *dataset* yang dimasukkan. Pengguna juga sekarang  
16 harus memilih teknik mana yang ingin digunakan. Tampilan antarmuka akan menyesuaikan secara  
17 otomatis sesuai teknik yang dipilih dan dapat dilihat perubahannya pada Gambar 4.2 apabila teknik  
18 *Random Projection Perturbation* yang dipilih.

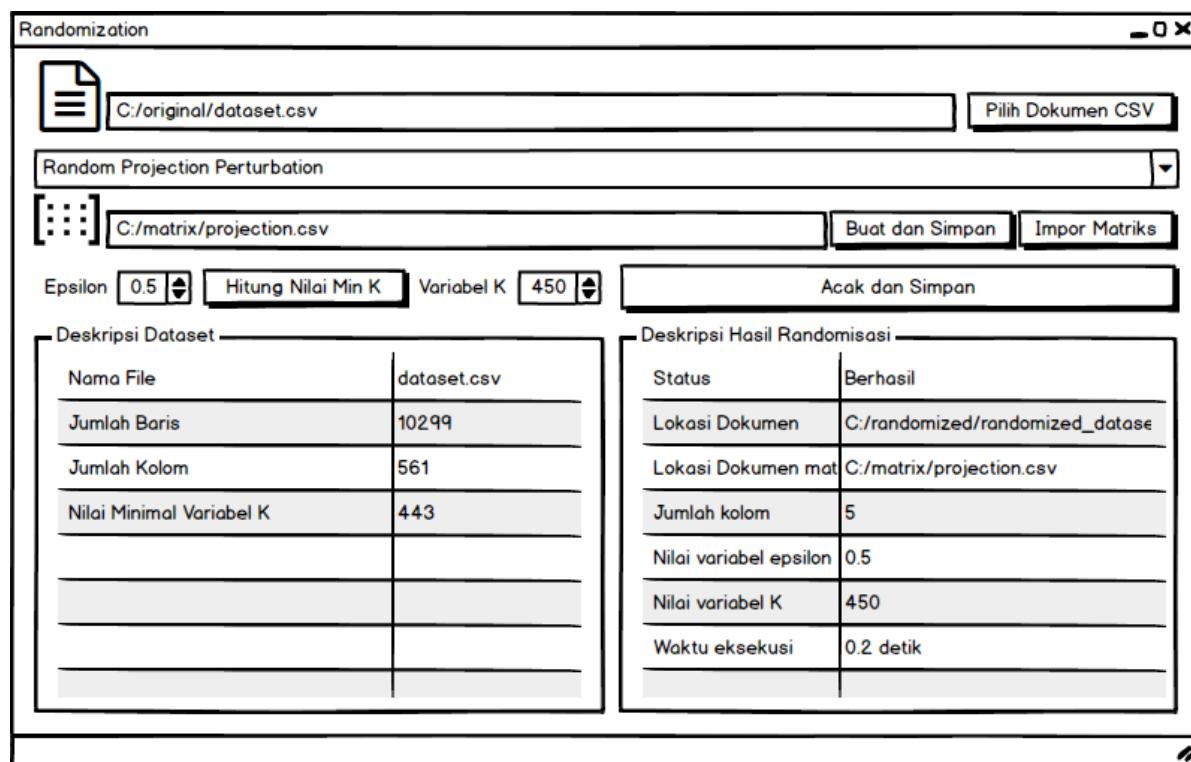
19 Sebelum melakukan metode *Randomization*, pengguna perlu memilih untuk membuat matriks  
20 rotasi/proyeksi baru atau mengimpor matriks rotasi/proyeksi yang sudah pernah dibuat masing-  
21 masing dengan menekan tombol “Buat dan Simpan Matriks” dan “Impor Matriks”. Pada teknik  
22 *Random Projection Perturbation*, ada persyaratan yang harus dipenuhi mengenai dimensi akhir yang  
23 diinginkan dan nilai Epsilon. Kedua nilai tersebut perlu sesuai dengan *dataset* yang ada sehingga  
24 pengguna tidak bisa sembarangan menentukan dimensi dan nilai Epsilon. Perangkat lunak akan  
25 membuat aturan mengenai minimal dimensi yang bisa digunakan sehingga pengguna tidak bisa  
26 memasukan dimensi yang lebih kecil dari minimal yang telah ditentukan.

27 Setelah pengguna melakukan berbagai pengaturan, pengguna dapat melakukan pengacakan  
28 dengan menekan tombol “Acak dan Simpan”. Apabila pengacakan berhasil dilakukan, perangkat  
29 lunak akan menampilkan *popup* yang memberitahukan bahwa pengacakan berhasil dilakukan.  
30 *Popup* tersebut dapat dilihat pada Gambar 4.3. Selain itu, perangkat lunak juga akan menampilkan  
31 berbagai informasi beserta deskripsi tentang hasil pengacakan yang berhasil dilakukan.

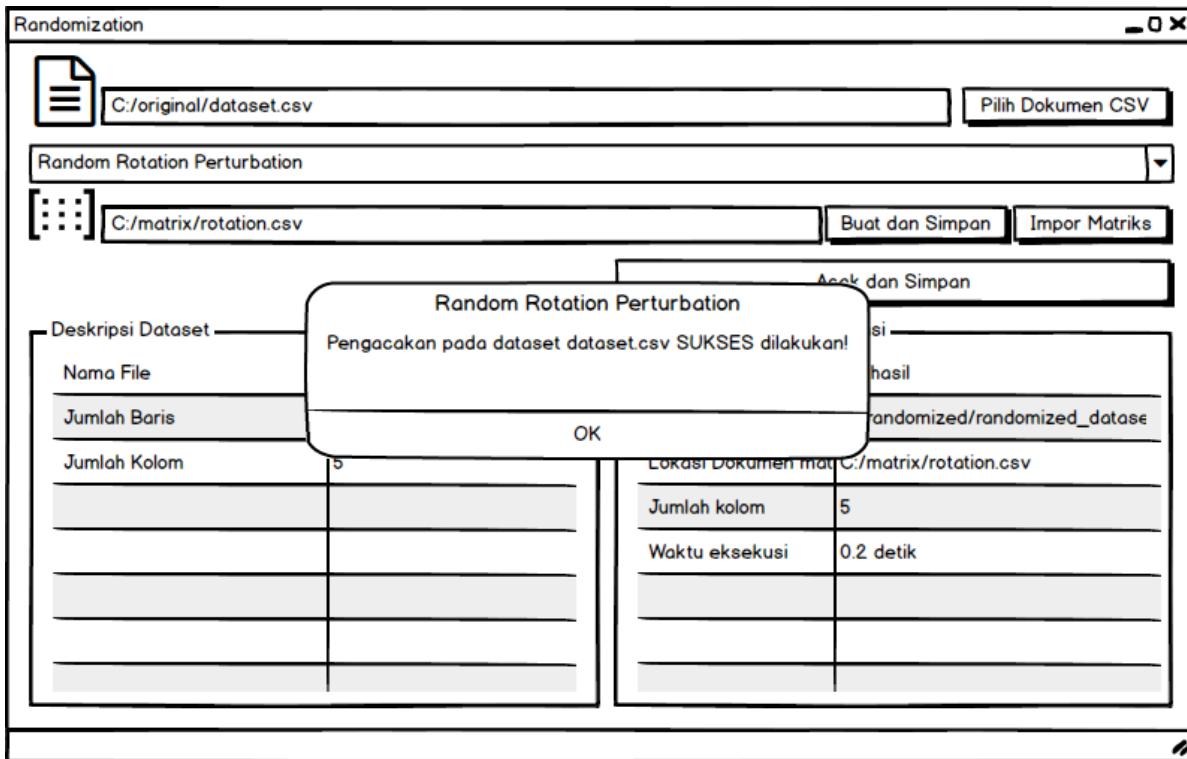
32 Perangkat lunak akan menampilkan deskripsi hasil dari pengacakan yang dilakukan. Informasi  
33 yang ditampilkan oleh perangkat lunak antara lain nama dokumen, ukuran dokumen, jumlah fitur,



Gambar 4.1: Halaman saat memilih teknik *Random Rotation Perturbation*



Gambar 4.2: Halaman saat memilih teknik *Random Projection Perturbation*



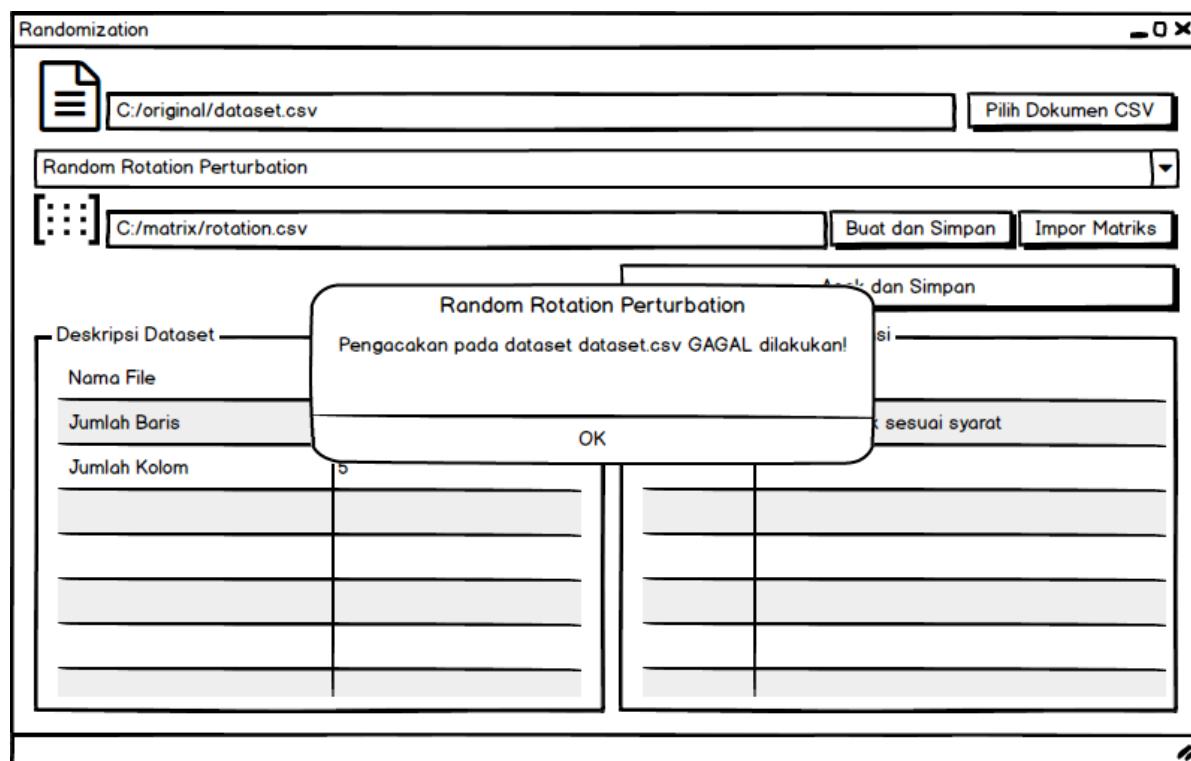
Gambar 4.3: *Popup* yang ditampilkan apabila pengacakan sukses dilakukan

- <sup>1</sup> nilai Epsilon yang dipakai, jumlah dimensi pada hasil pengacakan, dan kolom yang diabaikan.
- <sup>2</sup> Informasi ini ditampilkan oleh perangkat lunak bertujuan untuk memberitahukan pengguna properti properti *dataset* yang telah diacak dan pengguna dapat memeriksa hasil yang dihasilkan oleh
- <sup>3</sup> perangkat lunak apakah sesuai dengan keinginan pengguna.

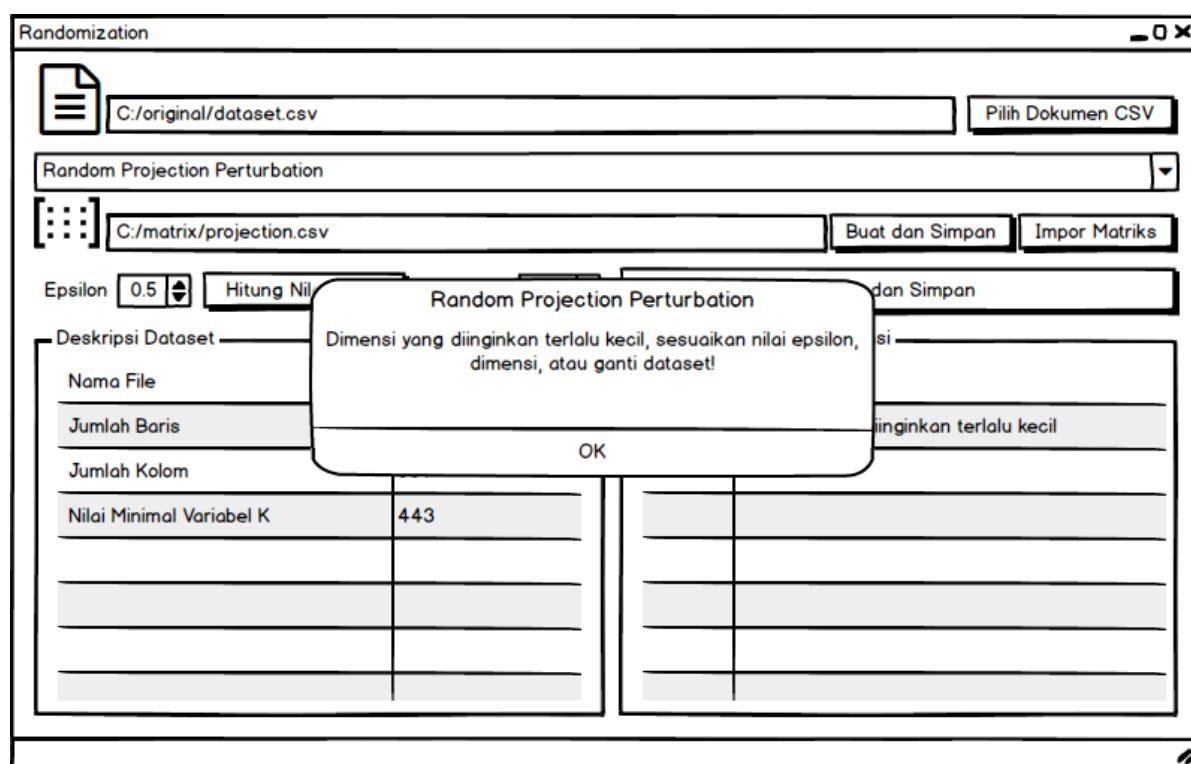
<sup>4</sup> Apabila ada masalah-masalah tertentu dan pengacakan gagal dilakukan, maka perangkat lunak akan memberitahukan bahwa pengacakan telah gagal dilakukan dengan menampilkan *popup* yang berisi peringatan bahwa pengacakan gagal dilakukan. Hasil pengacakan tidak akan terbuat dan perangkat lunak akan menampilkan informasi bahwa metode *Randomization* gagal dilakukan. *Popup* tersebut dapat dilihat pada Gambar 4.4. Oleh karena adanya persyaratan yang harus dipenuhi oleh pengguna untuk melakukan pengacakan dengan teknik *Random Projection Perturbation*, maka pengacakan bisa saja gagal dilakukan karena persyaratan yang ada tidak dipenuhi oleh pengguna. Persyaratan yang disebutkan adalah persyaratan jumlah dimensi dan nilai Epsilon yang telah dijelaskan di atas. Perangkat lunak akan menampilkan *popup* yang memberitahukan bahwa persyaratan tidak dipenuhi dan pengguna harus mengatur kembali pengaturan atau mengganti *dataset*. Rancangan ini dapat dilihat pada Gambar 4.5.

## <sup>16</sup> 4.2 Perancangan Kelas

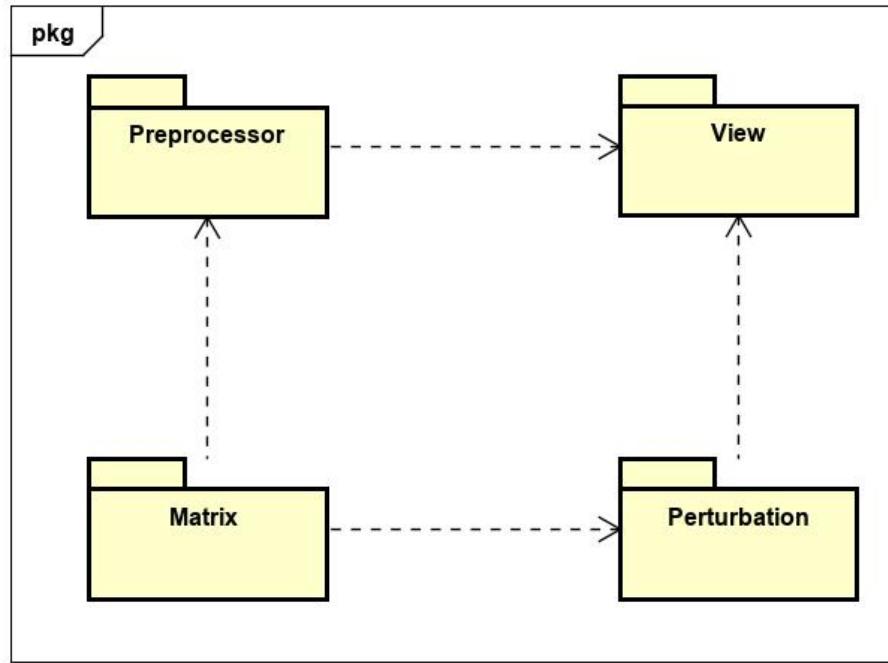
- <sup>17</sup> Dalam pengembangan perangkat lunak, perlu adanya perancangan perangkat lunak secara menyeluruh untuk menghindari kesulitan dan kebingungan pada waktu pengembangan. Perangkat lunak yang akan dibuat pada penelitian ini akan berorientasi objek sehingga perlu adanya perancangan kelas-kelas yang akan dibuat. Pada subbab ini akan dijelaskan perancangan kelas perangkat lunak
- <sup>18</sup>
- <sup>19</sup>
- <sup>20</sup>



Gambar 4.4: *Popup* yang ditampilkan apabila pengacakan gagal dilakukan



Gambar 4.5: *Popup* yang akan ditampilkan apabila persyaratan pada *dataset* tidak terpenuhi



Gambar 4.6: Diagram *package* perangkat lunak

<sup>1</sup> dan apa saja kegunaannya.

#### <sup>2</sup> 4.2.1 Diagram *Package*

<sup>3</sup> Perangkat lunak akan mempunyai 4 buah *package* yaitu *Perturbation*, *Matrix*, *Preprocessor*, dan  
<sup>4</sup> *View*. Keempat *package* ini mempunyai fungsinya masing-masing untuk mendukung perangkat lunak  
<sup>5</sup> berjalan yang akan dijelaskan pada subbab ini. Diagram *Package* perangkat lunak dapat dilihat  
<sup>6</sup> pada Gambar 4.6

#### <sup>7</sup> *Package Perturbation*

<sup>8</sup> *Package Perturbation* merupakan *package* yang menangani implementasi algoritma dari teknik  
<sup>9</sup> *Randomization* yang dipakai pada penelitian ini yaitu *Random Rotation Perturbation* dan *Random*  
<sup>10</sup> *Projection Perturbation*. Kedua algoritma ini masing-masing akan menjadi sebuah kelas terpisah  
<sup>11</sup> dan memiliki fungsinya masing-masing. Kedua kelas ini akan berada di dalam *package Perturbation*.  
<sup>12</sup> Satu kelas lagi akan ada di dalam *package* ini yang berperan sebagai kelas *super* bersifat abstrak  
<sup>13</sup> untuk kedua kelas lainnya.

<sup>14</sup> Dalam penerapan algoritma teknik yang dipakai, *package* ini membutuhkan komponen atau  
<sup>15</sup> fungsi lain untuk membuat algoritma yang diimplementasikan bekerja. Fungsi yang dibutuhkan  
<sup>16</sup> antara lain adalah membuat matriks dengan sifat tertentu. Oleh karena itu *package Perturbation*  
<sup>17</sup> membutuhkan *package Matrix* untuk membantu dalam pembuatan matriks yang khusus digunakan  
<sup>18</sup> pada algoritma. *Package Matrix* akan dijelaskan setelah ini.

1 **Package Matrix**

2 *Package Matrix* merupakan *package* yang menangani segala jenis fungsi yang berkaitan dengan  
3 matriks. Semua fungsi yang terkait tentang matriks diimplementasikan pada *package* ini, fungsi  
4 tersebut antara lain adalah implementasi struktur data matriks yang diimplementasikan pada  
5 sebuah kelas dan pembuatan matriks khusus yang akan dipakai untuk implementasi algoritma  
6 pada *package Perturbation* yang diimplementasikan menjadi tiga buah kelas. *Package* ini juga  
7 mengimplementasikan berbagai macam operasi matriks seperti perkalian, transpose matriks, dan  
8 menghitung determinan. Operasi-operasi ini diimplementasikan karena adanya kebutuhan operasi-  
9 operasi tersebut untuk membantu implementasi dari algoritma pada *package Perturbation*.

10 **Package Preprocessor**

11 *Package Preprocessor* merupakan *package* yang berfungsi sebagai *preprocessor* untuk masukan dan  
12 keluaran perangkat lunak. Tentunya masukan yang diberikan pengguna kepada perangkat lunak  
13 tidak bisa langsung diolah begitu saja. Perlu adanya pengolahan terlebih dahulu sebelum masukan  
14 yang diterima dipakai pada perangkat lunak. Oleh karena itu, *package* ini mempunyai fungsi untuk  
15 menyelesaikan masalah tersebut yaitu mengolah masukan yang diterima menjadi struktur data  
16 yang sesuai untuk digunakan pada perangkat lunak.

17 Pada penelitian ini, masukan perangkat lunak yang dimaksud adalah *dataset* yang akan diacak.  
18 Pengguna perlu mengikuti persyaratan masukan seperti apa yang dapat menjadi masukan perangkat  
19 lunak pada penelitian ini. Pada penelitian ini, perangkat lunak dirancang untuk hanya menerima  
20 dokumen berjenis *comma-separated values*. Oleh karena itu, *package preprocessor* ini memiliki  
21 sebuah fungsi untuk mengolah masukan berupa dokumen berjenis *comma-separated values* menjadi  
22 struktur data matriks atau sebaliknya dengan menggunakan *package Matrix*.

23 **Package View**

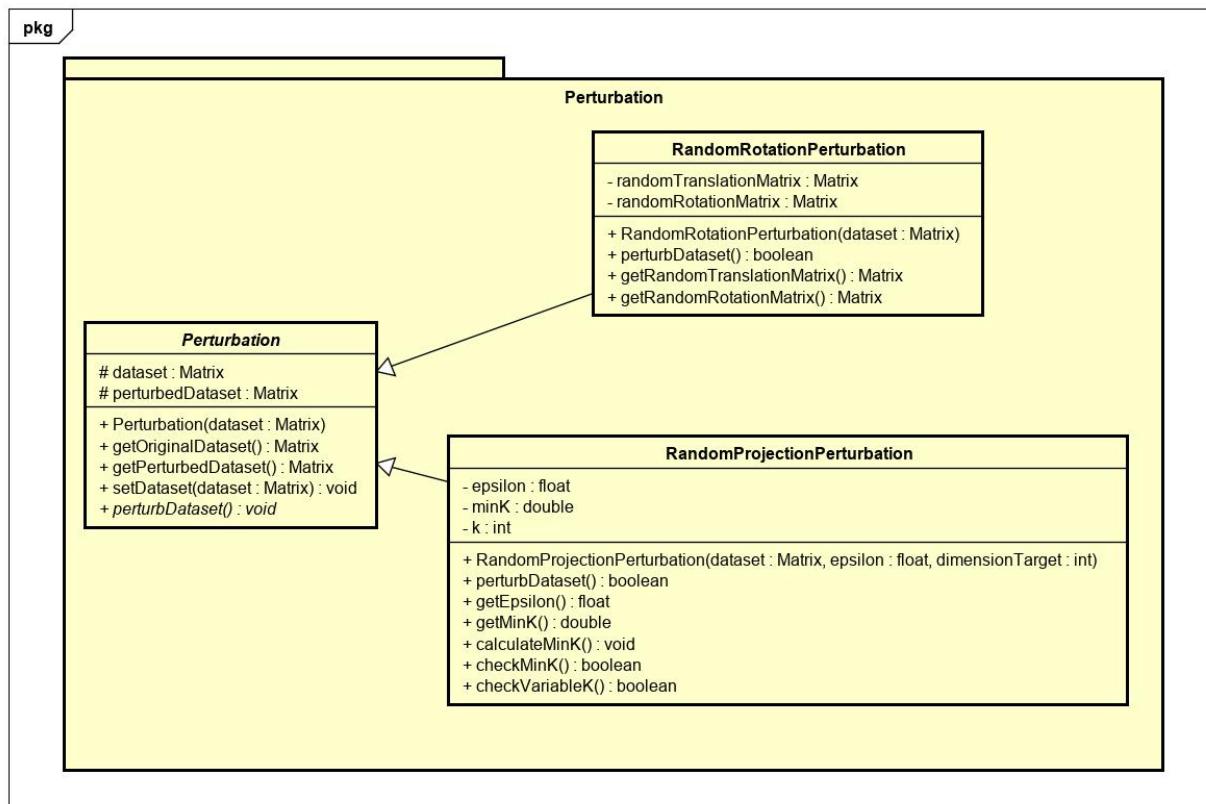
24 *Package View* merupakan *package* yang menangani bagian antarmuka pada perangkat lunak di  
25 penelitian ini. Antarmuka perangkat lunak akan diimplementasikan menggunakan *framework*  
26 antarmuka grafis berbasis bahasa pemrograman Python yang bernama Kivy. *Package* ini akan berisi  
27 kelas-kelas dan seluruh fungsi yang bertujuan untuk menampilkan antarmuka pada perangkat lunak.

28 **4.2.2 Diagram Kelas pada *Package Perturbation***

29 *Package Perturbation* memiliki tiga buah kelas yang bertujuan untuk mengimplementasikan algo-  
30 ritma teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*. Ketiga kelas  
31 tersebut adalah *Perturbation*, *RandomRotationPerturbation*, dan *RandomProjectionPerturbation*  
32 yang akan dijelaskan secara detail pada subbab ini. Diagram kelas pada *package Perturbation* dapat  
33 dilihat pada Gambar 4.7.

34 **Kelas Perturbation**

35 Kelas *Perturbation* berperan sebagai kelas abstrak yang akan menjadi kelas *super* dari kedua kelas  
36 lainnya dalam package *Perturbation* yaitu kedua kelas yang mengimplementasikan algoritma teknik  
37 *Random Rotation Perturbation* dan *Random Projection Perturbation*. Kelas ini mendeklarasikan

Gambar 4.7: Diagram kelas pada package *Perturbation*

1 atribut dan fungsi apa saja yang seharusnya diimplementasikan oleh kelas *RandomRotationPerturbation* dan *RandomProjectionPerturbation*. Beberapa atribut dan fungsi tersebut masih kosong pada 2 kelas *Perturbation* sehingga berbagai atribut dan fungsi tersebut perlu didefinisikan pada kelas-kelas 3 turunannya. Selanjutnya akan dijelaskan secara rinci tiap atribut dan fungsi pada kelas ini.

4 Berikut adalah deskripsi setiap atribut pada kelas *Perturbation*.

- 5
- 6 *dataset* adalah atribut untuk menampung *dataset* yang diambil dari masukan perangkat lunak  
7 yang ingin diacak dan sudah berbentuk matriks. Tipe data atribut ini adalah *Matrix*.
  - 8 *perturbedDataset* adalah atribut untuk menampung hasil dari *dataset* yang telah diacak. Tipe  
9 data atribut ini adalah *Matrix*.

10 Berikut adalah deskripsi setiap fungsi pada kelas *Perturbation*.

- 11
- 12 *Perturbation* adalah *constructor* dari kelas *Perturbation*. Tujuan utama fungsi ini adalah  
13 mendefinisikan atribut-atribut yang ada. Fungsi ini memiliki sebuah masukan yang dinamakan  
*dataset* yang bertipe data *Matrix* dan berfungsi untuk mendefinisikan atribut *dataset*.
  - 14 *getOriginalDataset* adalah fungsi untuk mendapatkan *dataset* asli yang belum diacak atau  
15 dengan kata lain mendapatkan atribut *dataset*. Fungsi ini tidak memiliki masukan apapun  
16 dan mempunyai tipe data kembalian berupa *Matrix*.
  - 17 *getPerturbedDataset* adalah fungsi untuk mendapatkan hasil dari *dataset* yang telah diacak  
18 atau dengan kata lain mendapatkan atribut *perturbedDataset*. Fungsi ini tidak memiliki  
19 parameter apapun. Tipe data kembalian pada fungsi ini berupa *Matrix*.

- 1     • *setDataset* adalah fungsi untuk mendefinisikan ulang atribut *dataset* dengan *dataset* yang  
2         baru. Fungsi ini memiliki sebuah masukan yang dinamakan *dataset* yang bertipe data *Matrix*  
3         dan berfungsi untuk mendefinisikan atribut *dataset*. Tidak ada kembalian pada fungsi ini.
- 4     • *perturbDataset* adalah fungsi untuk melakukan pengacakan pada atribut *dataset* dan menyimpan  
5         hasilnya pada atribut *perturbedDataset*. Fungsi ini bersifat abstrak yang berarti fungsi  
6         ini belum didefinisikan sehingga fungsi ini harus didefinisikan pada setiap kelas turunannya.  
7         Tidak ada kembalian ataupun masukan pada fungsi ini.

8     **Kelas *RandomRotationPerturbation***

9     Kelas *RandomRotationPerturbation* adalah kelas yang mempunyai tujuan utama melakukan penga-  
10    cakan dengan teknik *Random Rotation Perturbation* pada *dataset* yang menjadi masukan perangkat  
11    lunak. Kelas ini merupakan kelas turunan dari kelas *Perturbation* sehingga kelas ini pun mewarisi se-  
12    mua atribut dan fungsi yang dimiliki oleh kelas *Perturbation*. Oleh karena itu, fungsi *perturbDataset*  
13    yang diturunkan dari kelas *Perturbation* harus didefinisikan oleh kelas *RandomRotationPerturbation*.  
14    Kelas ini akan mengimplementasikan algoritma dari teknik *Random Rotation Perturbation* untuk  
15    melakukan pengacakan pada fungsi *perturbDataset*. Selanjutnya akan dijelaskan secara rinci tiap  
16    atribut dan fungsi pada kelas ini.

17    Berikut adalah deskripsi setiap atribut pada kelas *RandomRotationPerturbation*.

- 18     • *randomTranslationMatrix* adalah atribut untuk menampung matriks translasi yang akan  
19         dipergunakan untuk implementasi algoritma *Random Rotation Perturbation*. Atribut ini  
20         mempunyai tipe data *Matrix*.
- 21     • *randomRotationMatrix* adalah atribut untuk menampung matriks rotasi yang akan dipergu-  
22         nakan untuk implementasi algoritma *Random Rotation Perturbation*. Atribut ini mempunyai  
23         tipe data *Matrix*.

24    Berikut adalah deskripsi setiap fungsi pada kelas *RandomRotationPerturbation*.

- 25     • *RandomRotationPerturbation* adalah *constructor* untuk mendefinisikan atribut-atribut yang  
26         ada dan memanggil *constructor* dari kelas *super* milik kelas *RandomRotationPerturbation*  
27         yaitu *Perturbation*. Tujuan utama fungsi ini adalah memanggil *constructor* dari kelas *super*  
28         yaitu *Perturbation* sehingga fungsi ini berguna untuk mendefinisikan atribut *dataset*. Selain  
29         itu, fungsi ini juga akan mendefinisikan atribut *randomTranslationMatrix* dan *randomRotatio-*  
30         *nMatrix*.
- 31     • *perturbDataset* adalah fungsi turunan dari kelas *Perturbation* yang pada kelas *RandomRotati-*  
32         *onPerturbation* diimplementasikan algoritma teknik *Random Rotation Perturbation*. Fungsi  
33         ini akan mengimplementasikan algoritma teknik *Random Rotation Perturbation* kepada atribut  
34         *dataset* dan menyimpan hasilnya kepada atribut *perturbedDataset*. Tidak ada masukan pada  
35         fungsi ini tetapi memiliki kembalian berupa *boolean* yang menyatakan apakah pengacakan  
36         berhasil dilakukan.
- 37     • *getRandomTranslationMatrix* adalah fungsi untuk mendapatkan matriks translasi yang di-  
38         gunakan untuk implementasi algoritma teknik *Random Rotation Perturbation* atau dengan

1 kata lain mendapatkan atribut *randomTranslationMatrix*. Fungsi ini tidak memiliki masukan  
2 apapun tetapi mempunyai kembalian berupa matriks translasi yang bertipe data *Matrix*.

- 3 • *getRandomRotationMatrix* adalah fungsi untuk mendapatkan matriks rotasi yang digunakan  
4 untuk implementasi algoritma teknik *Random Rotation Perturbation* atau dengan kata lain  
5 mendapatkan atribut *randomRotationMatrix*. Fungsi ini tidak memiliki masukan apapun  
6 tetapi mempunyai kembalian berupa matriks rotasi yang bertipe data *Matrix*.

7 **Kelas *RandomProjectionPerturbation***

8 Kelas *RandomProjectionPerturbation* adalah kelas yang mempunyai tujuan utama melakukan  
9 pengacakan dengan teknik *Random Projection Perturbation* pada *dataset* yang menjadi masukan  
10 perangkat lunak. Kelas ini merupakan kelas turunan dari kelas *Perturbation* sehingga kelas  
11 inipun mewarisi semua atribut dan fungsi yang dimiliki oleh kelas *Perturbation*. Oleh karena  
12 itu, fungsi *perturbDataset* yang diturunkan dari kelas *Perturbation* harus didefinisikan oleh kelas  
13 *RandomProjectionPerturbation*. Kelas ini akan mengimplementasikan algoritma dari teknik *Random*  
14 *Projection Perturbation* untuk melakukan pengacakan pada fungsi *perturbDataset*. Selanjutnya  
15 akan dijelaskan secara rinci tiap atribut dan fungsi pada kelas ini.

16 Berikut adalah deskripsi setiap atribut pada kelas *RandomProjectionPerturbation*.

- 17 • *epsilon* adalah atribut yang berguna untuk menentukan seberapa besar batas maksimal  
18 distorsi yang dapat terjadi pada hasil pengacakan. Atribut *epsilon* ini akan digunakan untuk  
19 menghitung nilai dari atribut *minK* yang akan dijelaskan nanti. Tipe data atribut ini adalah  
20 *float*, bilangan riil yang nantinya hanya akan diisi dengan rentang nilai (0, 1). Pengguna akan  
21 menentukan seberapa besar batas maksimal distorsi yang dapat terjadi pada hasil pengacakan  
22 dengan cara mendefinisikan atribut *epsilon* ini.
- 23 • *minK* adalah atribut yang berguna untuk menentukan dimensi terkecil untuk sebuah *dataset*  
24 yang ingin diacak dapat direduksi dengan distorsi yang terkontrol sesuai atribut *epsilon*.  
25 Atribut ini juga menentukan apakah *dataset* memenuhi persyaratan untuk direduksi yaitu  
26 memiliki jumlah kolom yang lebih besar dari nilai *minK*. Tipe data atribut ini adalah *double*,  
27 bilangan riil. Atribut ini akan dihitung oleh perangkat lunak sesuai nilai *epsilon* yang telah  
28 ditentukan oleh pengguna.
- 29 • *k* adalah atribut untuk menyimpan besar dimensi akhir yang diinginkan pengguna dimiliki  
30 oleh hasil *dataset* yang telah diacak. Pengguna harus mendefinisikan atribut ini dengan nilai  
31 yang lebih besar dari atau sama dengan nilai *minK*, jika tidak maka distorsi yang terjadi pada  
32 hasil pengacakan akan lebih besar dari nilai *epsilon* yang diinginkan. Tipe data atribut ini  
33 adalah *integer*, bilangan bulat.

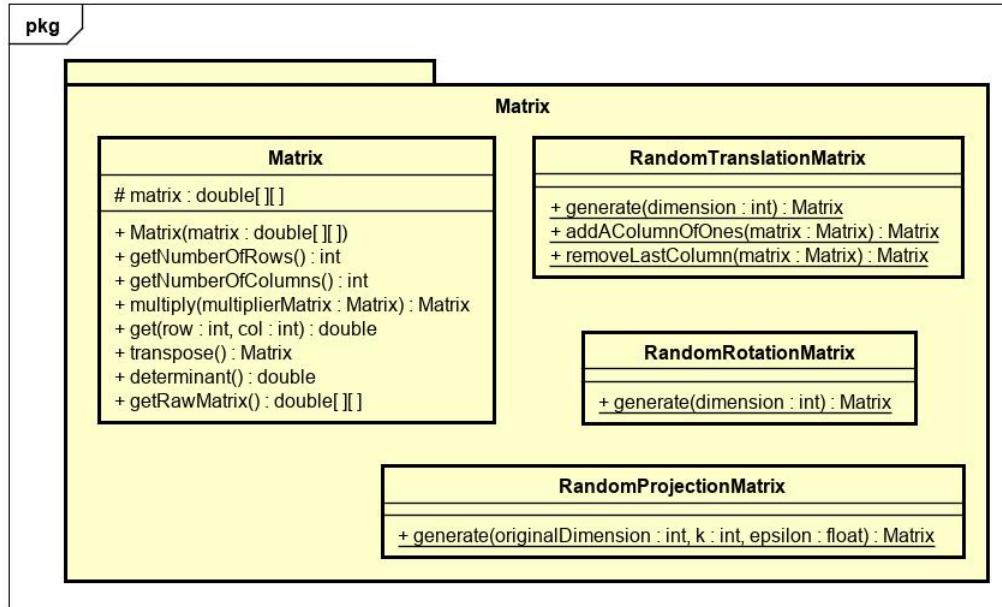
34 Berikut adalah deskripsi setiap fungsi pada kelas *RandomProjectionPerturbation*.

- 35 • *RandomProjectionPerturbation* adalah *constructor* untuk mendefinisikan atribut-atribut yang  
36 ada dan memanggil *constructor* dari kelas *super* milik kelas *RandomProjectionPerturbation*  
37 yaitu *Perturbation*. Tujuan utama fungsi ini adalah memanggil *constructor* dari kelas *super*  
38 yaitu *Perturbation* sehingga fungsi ini berguna untuk mendefinisikan atribut *dataset*. Selain  
39 itu, fungsi ini juga akan mendefinisikan atribut *epsilon*, *minK*, dan *k*.

- 1     • *perturbDataset* adalah fungsi turunan dari kelas *Perturbation* yang pada kelas *RandomP-*  
2         *rojectionPerturbation* diimplementasikan algoritma teknik *Random Projection Perturbation*.  
3         Fungsi ini akan mengimplementasikan algoritma teknik *Random Projection Perturbation*  
4         kepada atribut *dataset* dan menyimpan hasilnya kepada atribut *perturbedDataset*. Tidak ada  
5         masukan pada fungsi ini tetapi memiliki kembalian berupa *boolean* yang menyatakan apakah  
6         pengacakan berhasil dilakukan.
- 7     • *getEpsilon* adalah fungsi untuk mendapatkan nilai batas maksimal distorsi yang dapat terjadi  
8         pada hasil pengacakan atau dengan kata lain mendapatkan atribut *epsilon*. Fungsi ini tidak  
9         memiliki masukan apapun tetapi mempunyai kembalian berupa nilai dari atribut *epsilon* yang  
10         bertipe data *float*, bilangan riil.
- 11    • *getMinK* adalah fungsi untuk mendapatkan besar dimensi minimal hasil *dataset* yang telah  
12         diacak atau dengan kata lain mendapatkan atribut *minK*. Fungsi ini tidak memiliki masukan  
13         apapun tetapi mempunyai kembalian berupa nilai dari atribut *minK* yang bertipe data *double*,  
14         bilangan riil.
- 15    • *calculateMinK* adalah fungsi untuk menghitung nilai *minK* sesuai dengan atribut *epsilon* dan  
16         jumlah baris pada *dataset* yang ingin diacak. Fungsi ini akan menghitung nilai *minK* tersebut  
17         dan menyimpan hasilnya pada atribut *minK*. Tidak ada masukan ataupun kembalian pada  
18         fungsi ini.
- 19    • *checkMinK* adalah fungsi untuk memeriksa apakah dimensi dari *dataset* yang ingin diacak lebih  
20         besar dari nilai *minK*, besar dimensi minimal. Selain itu, fungsi ini memeriksa apakah nilai *k*  
21         lebih besar dari atau sama dengan nilai *minK*. Intinya fungsi ini menguji apakah *dataset* dan  
22         keinginan pengguna memenuhi persyaratan untuk mengaplikasikan teknik *Random Projection*  
23         *Perturbation*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa  
24         sebuah *boolean* yang menandakan apakah *dataset* dan *k* yang diberikan oleh pengguna  
25         memenuhi persyaratan untuk mengaplikasikan teknik *Random Projection Perturbation* dengan  
26         *dataset* dan *k* tersebut.
- 27    • *checkVariableK* adalah fungsi untuk memeriksa apakah nilai *k* lebih besar dari atau sama  
28         dengan nilai *minK*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian  
29         berupa sebuah *boolean* yang menandakan apakah nilai variabel *k* yang diberikan oleh pengguna  
30         memenuhi persyaratan untuk mengaplikasikan teknik *Random Projection Perturbation* dengan  
31         nilai variabel *k* tersebut.

32 **4.2.3 Diagram Kelas pada *Package Matrix***

33 *Package Matrix* memiliki empat buah kelas yang bertujuan untuk menangani segala jenis fungsi yang  
34         berkaitan dengan matriks maupun pembuatan matriks yang khusus digunakan untuk implementasi  
35         algoritma. Keempat kelas tersebut adalah *Matrix*, *RandomTranslationMatrix*, dan *RandomRotatio-*  
36         *nMatrix* yang akan dijelaskan secara detail pada subbab ini. Diagram kelas pada *package Matrix*  
37         dapat dilihat pada Gambar 4.8.

Gambar 4.8: Diagram kelas pada *package Matrix*

### <sup>1</sup> Kelas *Matrix*

<sup>2</sup> Kelas *Matrix* adalah kelas yang menangani struktur data matriks serta segala fungsi atau operasi  
<sup>3</sup> yang berkaitan dengan matriks. Kelas ini akan mempunyai semua kebutuhan terkait dengan urusan  
<sup>4</sup> struktur data matriks yang ada untuk implementasi algoritma *Random Rotation Perturbation* dan  
<sup>5</sup> *Random Projection Perturbation*, antara lain seperti perkalian, transpose matriks, dan mencari  
<sup>6</sup> determinan. Selanjutnya akan dijelaskan secara rinci tiap atribut dan fungsi pada kelas ini.

<sup>7</sup> Berikut adalah deskripsi setiap atribut pada kelas *Matrix*.

- <sup>8</sup> • *matrix* adalah atribut untuk menyimpan matriks dengan cara menyimpannya memakai *array*  
<sup>9</sup> 2 dimensi dengan tipe data *double*, bilangan riil.

<sup>10</sup> Berikut adalah deskripsi setiap fungsi pada kelas *Matrix*.

- <sup>11</sup> • *Matrix* adalah *constructor* dari kelas *PerturbMatrixxation*. Tujuan utama fungsi ini adalah  
<sup>12</sup> mendefinisikan atribut-atribut yang ada. Fungsi ini memiliki sebuah masukan berbentuk array  
<sup>13</sup> yang dinamakan *matrix* yang bertipe data *double* dan berfungsi untuk mendefinisikan atribut  
<sup>14</sup> *matrix*.

- <sup>15</sup> • *getNumberOfRows* adalah fungsi untuk mendapatkan jumlah baris yang ada pada matriks  
<sup>16</sup> kelas ini atau dengan kata lain mendapatkan ukuran dari atribut *matrix*. Fungsi ini tidak  
<sup>17</sup> memiliki masukan apapun tetapi mempunyai kembalian berupa jumlah baris yang bertipe  
<sup>18</sup> data *integer*, bilangan bulat.

- <sup>19</sup> • *getNumberOfColumns* adalah fungsi untuk mendapatkan jumlah kolom yang ada pada matriks  
<sup>20</sup> kelas ini atau dengan kata lain mendapatkan ukuran dari atribut *matrix*. Fungsi ini tidak  
<sup>21</sup> memiliki masukan apapun tetapi mempunyai kembalian berupa jumlah baris yang bertipe  
<sup>22</sup> data *integer*, bilangan bulat.

- 1     • *multiply* adalah fungsi yang berguna untuk mengkalikan matriks yang ada pada atribut *matrix*  
2       dengan suatu matriks lain. Tentu saja matriks lain tersebut harus sudah berbentuk objek  
3       kelas *Matrix* sehingga mempunyai tipe data yang sama dan dapat dikalikan. Fungsi ini  
4       mempunyai sebuah parameter yaitu *multiplierMatrix* bertipe data objek kelas *Matrix* yang  
5       berguna sebagai pengali matriks yang ingin dikalikan. Kembalian pada fungsi ini adalah hasil  
6       kali antara kedua matriks dan kembalian ini bertipe data objek kelas *Matrix*.
- 7     • *get* adalah fungsi untuk mendapatkan sebuah elemen pada matriks atau dengan kata lain  
8       sebuah nilai pada atribut *matrix*. Fungsi ini memiliki dua buah masukan yaitu *row* dan *col*  
9       yang masing-masing berguna sebagai penunjuk baris dan kolom mana yang ingin didapatkan.  
10      Kembalian pada fungsi ini adalah elemen matriks pada baris dan kolom yang diinginkan dan  
11       bertipe data *double*, bilangan riil.
- 12     • *transpose* adalah fungsi untuk mendapatkan transpose dari matriks kelas ini atau dengan  
13       kata lain transpose dari atribut *matrix*. Fungsi ini memiliki kembalian berupa matriks yang  
14       merupakan hasil transpose dan bertipe data objek kelas *Matrix*. Tidak ada masukan apapun  
15       pada fungsi ini.
- 16     • *determinant* adalah fungsi untuk mendapatkan determinan dari matriks kelas ini. Fungsi ini  
17       memiliki kembalian berupa determinan matriks yang bertipe data *double*, bilangan riil. Tidak  
18       ada masukan apapun pada fungsi ini.
- 19     • *getRawMatrix* adalah fungsi untuk mendapatkan matriks pada kelas ini atau dengan kata lain  
20       mendapatkan atribut *matrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai  
21       kembalian berupa *array* yang bertipe data *double*, bilangan riil.

## 22    **Kelas *RandomTranslationMatrix***

23    Kelas *RandomTranslationMatrix* adalah kelas statis sehingga tidak bisa diinstansiasi dan hanya  
24    memiliki atribut atau fungsi statis saja. Tujuan utama dari kelas ini adalah menangani pembuatan  
25    matriks translasi yang akan digunakan untuk melakukan operasi translasi yang akan diimplementa-  
26    sikan di kelas *RandomRotationPerturbation*. Kelas ini tidak memiliki atribut apapun tetapi memiliki  
27    tiga buah fungsi statis yang memiliki fungsi seputar pembuatan matriks translasi dan fungsi yang  
28    mendukung operasi translasi. Selanjutnya akan dijelaskan secara rinci tiap fungsi pada kelas ini.

29    Berikut adalah deskripsi setiap fungsi pada kelas *RandomTranslationMatrix*.

- 30     • *generate* adalah fungsi statis yang berguna untuk membuat matriks translasi. Fungsi ini  
31       memiliki sebuah parameter yaitu *dimension* yang akan menentukan ukuran atau dimensi  
32       matriks translasi yang ingin dibuat. Kembalian pada fungsi ini adalah sebuah matriks translasi  
33       yang bertipe data objek kelas *Matrix*.
- 34     • *addAColumnOfOnes* adalah fungsi statis yang berguna untuk menambahkan sebuah kolom  
35       pada suatu matriks di posisi terakhir (setelah kolom terakhir). Kolom tersebut akan berisi  
36       angka satu pada setiap barisnya. Alasan dibuatnya fungsi ini dikarenakan kebutuhan persya-  
37       ratan yang harus dipenuhi sebuah matriks apabila ingin diterapkan operasi translasi. Fungsi  
38       ini memiliki sebuah parameter yaitu matriks yang diinginkan bernama *matrix* dan bertipe

1 data objek kelas *Matrix*. Kembalian pada fungsi ini adalah matriks yang telah ditambahkan  
2 sebuah kolom dan bertipe data objek kelas *Matrix*.

- 3 • *removeLastColumn* adalah fungsi statis yang berguna untuk menghapus kolom terakhir pada  
4 suatu matriks. Alasan dibuatnya fungsi ini dikarenakan kebutuhan persyaratan yang harus  
5 dipenuhi sebuah matriks apabila ingin diterapkan operasi translasi. Setelah matriks ditam-  
6 bahkan dengan menggunakan fungsi *addAColumnOfOnes* dan diterapkan operasi translasi,  
7 kolom terakhir pada matriks tersebut perlu dibuang dikarenakan kolom tersebut adalah kolom  
8 hasil penambahan yang hanya digunakan untuk operasi translasi dan bukan kolom asli matriks  
9 tersebut.

10 **Kelas *RandomRotationMatrix***

11 Kelas *RandomRotationMatrix* adalah kelas statis sehingga tidak bisa diinstansiasi dan hanya memiliki  
12 atribut atau fungsi statis saja. Tujuan utama dari kelas ini adalah menangani pembuatan matriks  
13 rotasi yang akan digunakan untuk melakukan transformasi rotasi yang akan diimplementasikan di  
14 kelas *RandomRotationPerturbation*. Kelas ini tidak memiliki atribut apapun tetapi memiliki sebuah  
15 fungsi statis yang memiliki fungsi untuk pembuatan matriks rotasi. Selanjutnya akan dijelaskan  
16 secara rinci tiap fungsi pada kelas ini.

17 Berikut adalah deskripsi setiap fungsi pada kelas *RandomRotationMatrix*.

- 18 • *generate* adalah fungsi statis yang berguna untuk membuat matriks rotasi. Pembuatan matriks  
19 rotasi dilakukan dengan mengikuti distribusi Haar [9] dan akan dibantu oleh *library Scipy*<sup>1</sup>.  
20 Fungsi ini memiliki sebuah parameter yaitu *dimension* yang akan menentukan ukuran atau  
21 dimensi matriks rotasi yang ingin dibuat. Kembalian pada fungsi ini adalah sebuah matriks  
22 rotasi yang bertipe data objek kelas *Matrix*.

23 **Kelas *RandomProjectionMatrix***

24 Kelas *RandomProjectionMatrix* adalah kelas statis sehingga tidak bisa diinstansiasi dan hanya  
25 memiliki atribut atau fungsi statis saja. Tujuan utama dari kelas ini adalah menangani pembuatan  
26 matriks proyeksi yang akan digunakan untuk melakukan proyeksi yang akan diimplementasikan  
27 di kelas *RandomProjectionPerturbation*. Kelas ini tidak memiliki atribut apapun tetapi memiliki  
28 sebuah fungsi statis yang memiliki fungsi untuk pembuatan matriks proyeksi. Selanjutnya akan  
29 dijelaskan secara rinci tiap fungsi pada kelas ini.

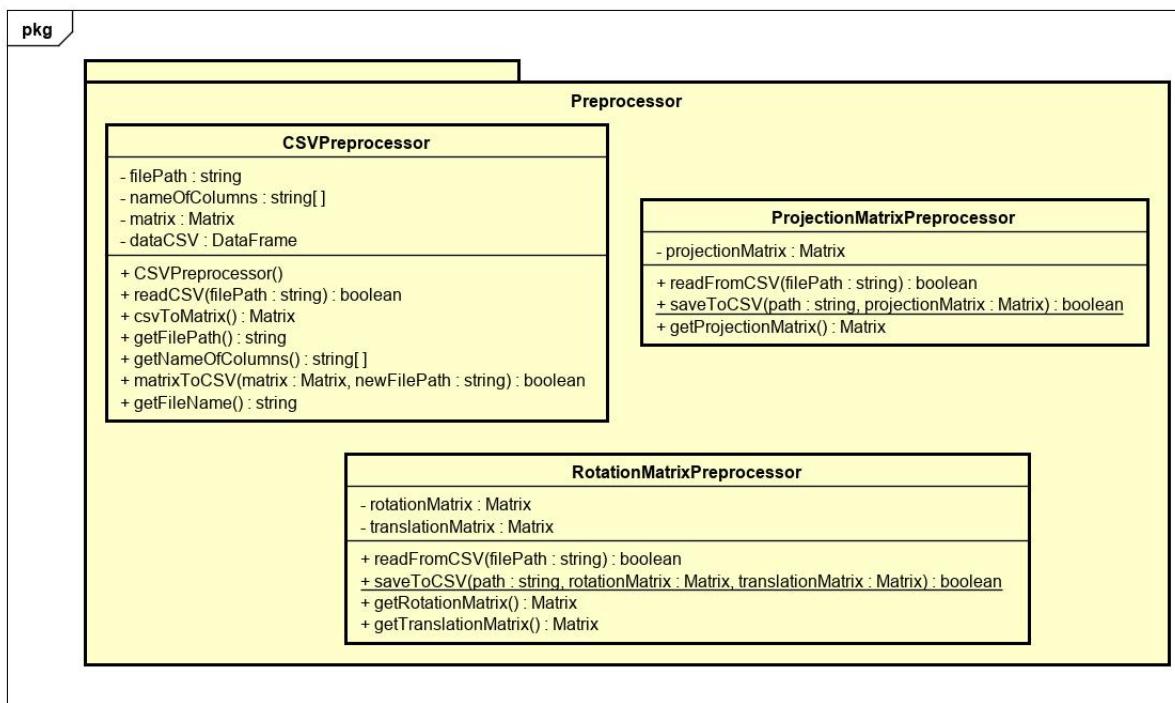
30 Berikut adalah deskripsi setiap fungsi pada kelas *RandomProjectionMatrix*.

- 31 • *generate* adalah fungsi statis yang berguna untuk membuat matriks proyeksi. Pembuatan  
32 matriks proyeksi akan dibantu oleh *library Scikit-learn*<sup>2</sup>. Fungsi ini memiliki 3 buah parameter  
33 yaitu *epsilon*, *k*, dan *originalDimension* yang akan menentukan ukuran matriks proyeksi yang  
34 ingin dibuat. Kembalian pada fungsi ini adalah sebuah matriks proyeksi yang bertipe data  
35 objek kelas *Matrix*.

---

<sup>1</sup>[http://scipy.github.io/devdocs/generated/scipy.stats.special\\_ortho\\_group.html](http://scipy.github.io/devdocs/generated/scipy.stats.special_ortho_group.html)

<sup>2</sup>[https://scikit-learn.org/stable/modules/random\\_projection.html](https://scikit-learn.org/stable/modules/random_projection.html)



Gambar 4.9: Diagram kelas pada *package Preprocessor*

#### 1 4.2.4 Diagram Kelas pada *Package Preprocessor*

2 *Package Preprocessor* memiliki 3 buah kelas yang bertujuan untuk mengolah masukan perangkat  
 3 lunak agar dapat diterapkan teknik *Randomization*. Kelas tersebut yaitu *CSVPreprocessor*, *Ro-  
 4 tationMatrixPreprocessor*, dan *ProjectionMatrixPreprocessor* yang akan dijelaskan secara detail pada  
 5 subbab ini. Diagram kelas pada *package Preprocessor* dapat dilihat pada Gambar 4.9.

#### 6 Kelas *CSVPreprocessor*

7 Kelas *CSVPreprocessor* berguna untuk mengolah masukan perangkat lunak yang berjenis *comma-  
 8 separated values* sebelum diterapkan teknik *Randomization* agar masukan tersebut sesuai dengan  
 9 persyaratan teknik *Randomization*. Tujuan utama dari kelas ini adalah mengubah masukan berjenis  
 10 *comma-separated values* menjadi sebuah objek kelas *Matrix* dan sebaliknya. Kelas ini memiliki tiga  
 11 buah atribut dan enam buah fungsi yang selanjutnya akan dijelaskan secara rinci.

12 Berikut adalah deskripsi setiap atribut pada kelas *CSVPreprocessor*.

- 13 • *filePath* adalah atribut untuk menyimpan lokasi masukan yang berjenis *comma-separated  
 14 values* yang ingin diolah pada komputer pengguna. Atribut ini memiliki tipe data *string*.
- 15 • *nameOfColumns* adalah atribut untuk menyimpan nama seluruh kolom pada *dataset* yang  
 16 telah diolah. Atribut ini berupa array yang bertipe data *string*.
- 17 • *matrix* adalah atribut untuk menyimpan objek kelas *Matrix* yang didapatkan dari hasil  
 18 pengolahan masukan yang berjenis *comma-separated values*. Atribut ini memiliki tipe data  
 19 objek kelas *Matrix*.

- 1     • *dataCSV* adalah atribut untuk menyimpan data dokumen *comma-separated values* yang telah  
2       dibaca dalam bentuk *DataFrame*<sup>3</sup>. Atribut ini memiliki tipe data objek kelas *DataFrame*.

3       Berikut adalah deskripsi setiap fungsi pada kelas *CSVPreprocessor*.

- 4     • *CSVPreprocessor* adalah *constructor* kelas ini yang berfungsi untuk membuat sebuah objek  
5       kelas *CSVPreprocessor* dan menginisialisasi semua atribut pada kelas ini. Fungsi ini tidak  
6       memiliki masukan apapun.
- 7     • *readCSV* adalah fungsi untuk membaca dokumen berjenis *comma-separated values* yang ingin  
8       diolah oleh kelas ini. Tujuan utama dari fungsi ini adalah menyimpan lokasi dokumen pada  
9       atribut *filePath* dan menyimpan data dokumen *comma-separated values* yang telah dibaca  
10      dalam bentuk *DataFrame* pada atribut *dataCSV*. Fungsi ini memiliki sebuah masukan bernama  
11      *filePath* yang berguna untuk menentukan lokasi dokumen yang ingin diolah. Kembalian pada  
12      fungsi ini adalah sebuah *boolean* yang menyatakan apakah fungsi ini berhasil membaca  
13      dokumen yang menjadi masukan fungsi *readCSV*.
- 14     • *csvToMatrix* adalah fungsi untuk mengolah dokumen *comma-separated values* yang telah  
15       dibaca menjadi sebuah objek kelas *Matrix*. Fungsi ini tidak memiliki masukan apapun tetapi  
16       mempunyai kembalian berupa objek kelas *Matrix* yang didapatkan dari hasil pengolahan  
17       dokumen *comma-separated values*.
- 18     • *getFilePath* adalah fungsi untuk mendapatkan lokasi dokumen *comma-separated values* yang  
19       telah dibaca atau dengan kata lain mendapatkan atribut *filePath*. Fungsi ini tidak memiliki  
20       masukan apapun tetapi mempunyai kembalian berupa lokasi dokumen yang bertipe data  
21       *string*.
- 22     • *getNameOfColumns* adalah fungsi untuk mendapatkan nama semua kolom yang ada pada  
23       dokumen *comma-separated values* yang telah dibaca atau dengan kata lain mendapatkan  
24       atribut *nameOfColumns*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai  
25       kembalian berupa *array* yang bertipe data *string*.
- 26     • *matrixToCSV* adalah fungsi untuk mengolah sebuah objek kelas *Matrix* menjadi sebuah  
27       dokumen *comma-separated values* atau dengan kata lain mengkonversi sebuah matriks menjadi  
28       dokumen berjenis *comma-separated values*. Fungsi ini bisa dikatakan kebalikan dari fungsi  
29       *csvToMatrix*. Ada dua buah masukan pada fungsi ini yaitu sebuah matriks dan lokasi  
30       penyimpanan dokumen yang baru. Kembalian pada fungsi ini adalah sebuah *boolean* yang  
31       menyatakan apakah konversi berhasil dilakukan.
- 32     • *getFileName* adalah fungsi untuk mendapatkan nama dokumen *comma-separated values* yang  
33       telah dibaca. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa  
34       nama dokumen yang bertipe data *string*.

35     **Kelas *ProjectionMatrixPreprocessor***

36     Kelas *ProjectionMatrixPreprocessor* berguna untuk mengolah masukan perangkat lunak berupa  
37       matriks proyeksi yang berjenis *comma-separated values* sebelum diterapkan teknik *Randomization*

---

<sup>3</sup><https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

1 agar masukan tersebut sesuai dengan persyaratan teknik *Randomization*. Tujuan utama dari kelas  
2 ini adalah mengubah masukan matriks proyeksi berjenis *comma-separated values* menjadi sebuah  
3 objek kelas *Matrix* dan sebaliknya. Kelas ini memiliki sebuah atribut dan tiga buah fungsi yang  
4 selanjutnya akan dijelaskan secara rinci.

5 Berikut adalah deskripsi setiap atribut pada kelas *ProjectionMatrixPreprocessor*.

- 6 • *projectionMatrix* adalah atribut untuk menyimpan matriks proyeksi yang berbentuk objek  
7 kelas *Matrix* yang didapatkan dari hasil pengolahan masukan yang berjenis *comma-separated*  
8 *values*. Atribut ini memiliki tipe data objek kelas *Matrix*.

9 Berikut adalah deskripsi setiap fungsi pada kelas *ProjectionMatrixPreprocessor*.

- 10 • *readFromCSV* adalah fungsi untuk membaca dokumen berjenis *comma-separated values*  
11 menjadi matriks proyeksi yang dapat dipakai perangkat lunak. Fungsi ini memiliki sebuah  
12 masukan bernama *filePath* yang berguna untuk menentukan lokasi dokumen yang ingin  
13 diolah. Kembalian pada fungsi ini adalah sebuah *boolean* yang menyatakan apakah pembacaan  
14 dokumen berhasil dilakukan.

- 15 • *saveToCSV* adalah fungsi statis untuk mengolah sebuah matriks proyeksi berbentuk objek  
16 kelas *Matrix* menjadi sebuah dokumen *comma-separated values* dan menyimpannya pada  
17 sebuah direktori pengguna. Fungsi ini bisa dikatakan kebalikan dari fungsi *readFromCSV*.  
18 Ada dua buah masukan pada fungsi ini yaitu sebuah matriks proyeksi dan lokasi penyimpanan  
19 dokumen yang baru. Kembalian pada fungsi ini adalah sebuah *boolean* yang menyatakan  
20 apakah penyimpanan berhasil dilakukan.

- 21 • *getProjectionMatrix* adalah fungsi untuk mendapatkan matriks proyeksi berbentuk objek  
22 kelas *Matrix* yang didapatkan dari fungsi *readFromCSV* atau dengan kata lain mendapatkan  
23 variabel *projectionMatrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai  
24 kembalian berupa matriks proyeksi yang bertipe data *Matrix*.

## 25 Kelas *RotationMatrixPreprocessor*

26 Kelas *RotationMatrixPreprocessor* berguna untuk mengolah masukan perangkat lunak berupa  
27 matriks rotasi dan translasi yang berjenis *comma-separated values* sebelum diterapkan teknik  
28 *Randomization* agar masukan tersebut sesuai dengan persyaratan teknik *Randomization*. Tujuan  
29 utama dari kelas ini adalah mengubah masukan matriks rotasi dan translasi berjenis *comma-*  
30 *separated values* menjadi dua buah objek kelas *Matrix* dan sebaliknya. Kelas ini memiliki dua buah  
31 atribut dan empat buah fungsi yang selanjutnya akan dijelaskan secara rinci.

32 Berikut adalah deskripsi setiap atribut pada kelas *RotationMatrixPreprocessor*.

- 33 • *rotationMatrix* adalah atribut untuk menyimpan matriks rotasi yang berbentuk objek kelas  
34 *Matrix* yang didapatkan dari hasil pengolahan masukan yang berjenis *comma-separated values*.  
35 Atribut ini memiliki tipe data objek kelas *Matrix*.

- 36 • *translationMatrix* adalah atribut untuk menyimpan matriks translasi yang berbentuk objek  
37 kelas *Matrix* yang didapatkan dari hasil pengolahan masukan yang berjenis *comma-separated*  
38 *values*. Atribut ini memiliki tipe data objek kelas *Matrix*.

1 Berikut adalah deskripsi setiap fungsi pada kelas *RotationMatrixPreprocessor*.

- 2 • *readFromCSV* adalah fungsi untuk membaca dokumen berjenis *comma-separated values*  
3 menjadi matriks rotasi dan translasi yang dapat dipakai perangkat lunak. Fungsi ini memiliki  
4 sebuah masukan bernama *filePath* yang berguna untuk menentukan lokasi dokumen yang  
5 ingin diolah. Kembalian pada fungsi ini adalah sebuah *boolean* yang menyatakan apakah  
6 pembacaan dokumen berhasil dilakukan.
- 7 • *saveToCSV* adalah fungsi statis untuk mengolah sebuah matriks rotasi dan translasi berbentuk  
8 objek kelas *Matrix* menjadi sebuah dokumen *comma-separated values* dan menyimpannya pada  
9 sebuah direktori pengguna. Fungsi ini bisa dikatakan kebalikan dari fungsi *readFromCSV*.  
10 Ada tiga buah masukan pada fungsi ini yaitu sebuah matriks rotasi, matriks translasi dan  
11 lokasi penyimpanan dokumen yang baru. Kembalian pada fungsi ini adalah sebuah *boolean*  
12 yang menyatakan apakah penyimpanan berhasil dilakukan.
- 13 • *getRotationMatrix* adalah fungsi untuk mendapatkan matriks rotasi berbentuk objek kelas  
14 *Matrix* yang didapatkan dari fungsi *readFromCSV* atau dengan kata lain mendapatkan variabel  
15 *rotationMatrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian  
16 berupa matriks rotasi yang bertipe data *Matrix*.
- 17 • *getTranslationMatrix* adalah fungsi untuk mendapatkan matriks translasi berbentuk objek  
18 kelas *Matrix* yang didapatkan dari fungsi *readFromCSV* atau dengan kata lain mendapatkan  
19 variabel *translationMatrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai  
20 kembalian berupa matriks translasi yang bertipe data *Matrix*.

## 21 4.3 Masukan Perangkat Lunak

22 Perangkat lunak akan mempunyai sebuah masukan yang berupa *dataset*. *Dataset* yang ingin diacak  
23 memiliki persyaratan yaitu harus berupa sebuah matriks. Tetapi mayoritas *dataset* yang ada  
24 didistribusikan bukan sebagai matriks melainkan dokumen berjenis tertentu antara lain seperti  
25 dokumen berjenis *comma-separated values*. Oleh karena itu, perangkat lunak ini hanya dapat  
26 menerima masukan berupa dokumen berjenis *comma-separated values*. Berikut akan dijelaskan  
27 secara rinci masukan yang dapat diterima oleh perangkat lunak.

28 Dokumen *comma-separated values* adalah dokumen berisi teks yang menggunakan koma untuk  
29 memisahkan antara tiap nilai dengan nilai lainnya. Berikut adalah contoh isi dari dokumen berjenis  
30 *comma-separated values*.

31 `sepal_length,sepal_width,petal_length,petal_width,species`  
32 `5.1,3.5,1.4,0.2,setosa`  
33 `4.9,3,1.4,0.2,setosa`  
34 `4.7,3.2,1.3,0.2,setosa`  
35 `4.6,3.1,1.5,0.2,setosa`  
36 `5,3.6,1.4,0.2,setosa`  
37 `5.4,3.9,1.7,0.4,setosa`  
38 `4.6,3.4,1.4,0.3,setosa`

1 5,3.4,1.5,0.2, setosa  
2 4.4,2.9,1.4,0.2, setosa  
3 4.9,3.1,1.5,0.1, setosa

4 Baris pertama pada contoh di atas merupakan nama semua kolom yang ada pada *dataset*. Baris  
5 lainnya adalah nilai-nilai setiap kolom pada suatu rekord. Pada *dataset* di atas, kolom terakhir  
6 adalah kolom label. Selain kolom tersebut adalah kolom fitur yang akan diacak. Kolom fitur tersebut  
7 untuk perangkat lunak *Randomization* ini mempunyai persyaratan yaitu nilai kolom tersebut harus  
8 berjenis numerik.

1

## BAB 5

2

### IMPLEMENTASI DAN PENGUJIAN

3 Pada bab ini akan ditunjukkan tampilan dari implementasi perangkat lunak dan juga bagaimana  
4 perangkat lunak diimplementasikan. Pengujian fungsional dan eksperimental perangkat lunak juga  
5 akan dilakukan. Hasil dari pengujian akan dijelaskan secara rinci dan sistematis serta akan dibuat  
6 kesimpulan untuk pengujian yang telah dilakukan.

7 **5.1 Implementasi Antarmuka**

8 Antarmuka perangkat lunak diimplementasikan dengan memakai *framework* antarmuka grafis  
9 berbasis bahasa pemrograman Python yang bernama Kivy. Implementasi antarmuka disesuaikan  
10 dengan rancangan antarmuka perangkat lunak yang telah dibuat pada bab 4. Gambar 5.1 adalah  
11 tampilan antarmuka dari implementasi perangkat lunak.

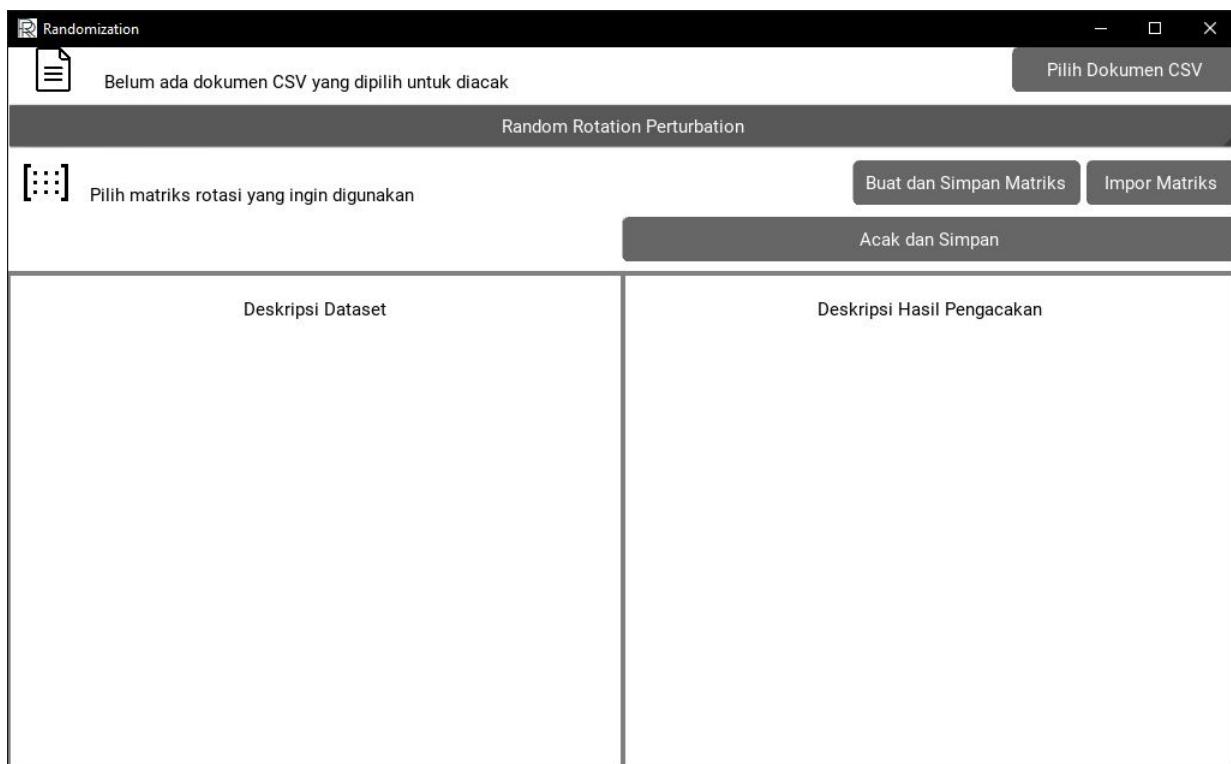
12 Antarmuka perangkat lunak mempunyai tiga buah bagian yang mempunyai fungsinya masing-  
13 masing. Ketiga bagian ini dapat dilihat pada Gambar 5.2 Pertama adalah bagian masukan dan  
14 pengaturan, terdapat pada bagian atas yang bernomor satu dan dikelilingi kotak merah. Kedua  
15 adalah bagian deskripsi *dataset*, terdapat pada bagian bawah sebelah kiri yang bernomor dua dan  
16 dikelilingi kotak biru. Terakhir adalah bagian deskripsi hasil pengacakan, terdapat pada bagian  
17 bawah sebelah kanan yang bernomor tiga dan dikelilingi kotak hijau. Ketiga bagian ini akan  
18 dijelaskan secara rinci pada subbab-subbab berikutnya.

19 Perangkat lunak *Randomization* mengimplementasikan dua buah teknik *Randomization* yang  
20 berbeda yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation*. Oleh karena  
21 itu, antarmuka perangkat lunak akan menyesuaikan dengan teknik yang dipilih oleh pengguna.  
22 Ketiga bagian antarmuka yang telah disebutkan tadi dengan otomatis akan berubah sesuai dengan  
23 teknik yang dipilih. Pada setiap subbab akan dijelaskan juga sekaligus perbedaan antarmuka teknik  
24 *Randomization* satu dengan yang lainnya.

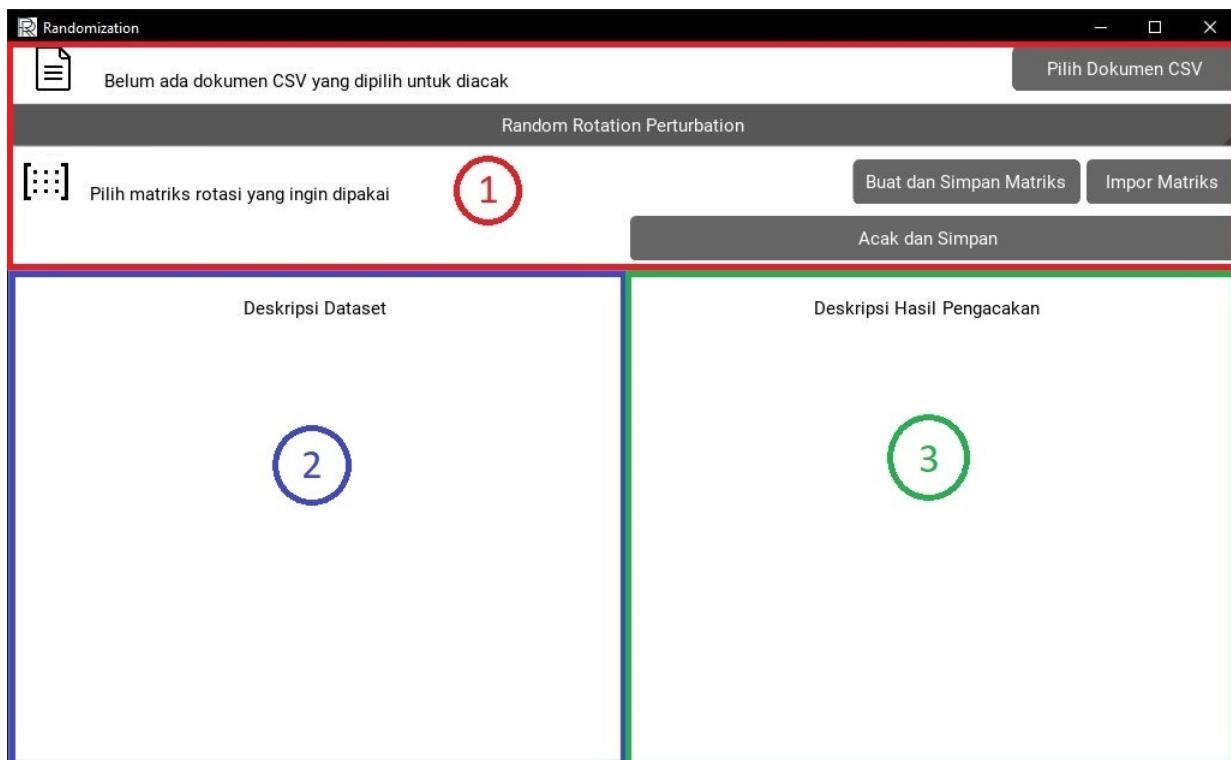
25 **5.1.1 Masukan dan Pengaturan**

26 Bagian masukan dan pengaturan menyediakan berbagai interaksi untuk pengguna dapat mengatur  
27 masukan yang perlu diberikan kepada perangkat lunak dan menerapkan teknik *Randomization*  
28 yang diinginkan. Ada beberapa fungsi inti pada bagian ini yaitu sebagai berikut.

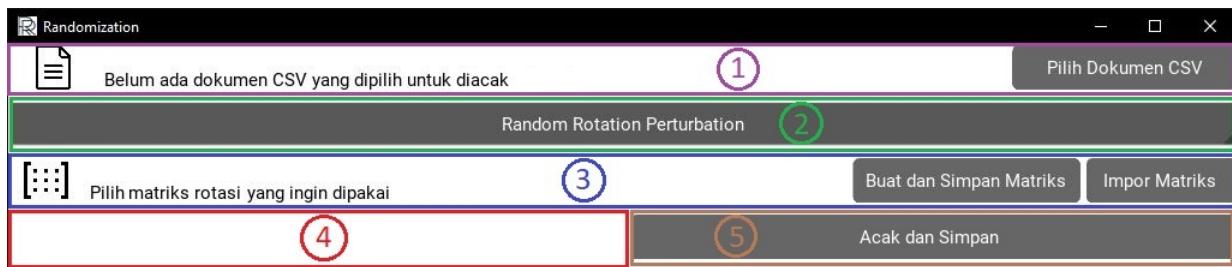
- 29
  - Masukan *dataset* berupa file *comma-separated values* yang ingin diacak.
  - 30 • Memilih teknik *Randomization* yang ingin digunakan.



Gambar 5.1: Tampilan perangkat lunak yang pertama ditampilkan saat perangkat lunak baru dibuka



Gambar 5.2: Bagian-bagian pada antarmuka perangkat lunak



Gambar 5.3: Bagian antarmuka masukan dan pengaturan perangkat lunak

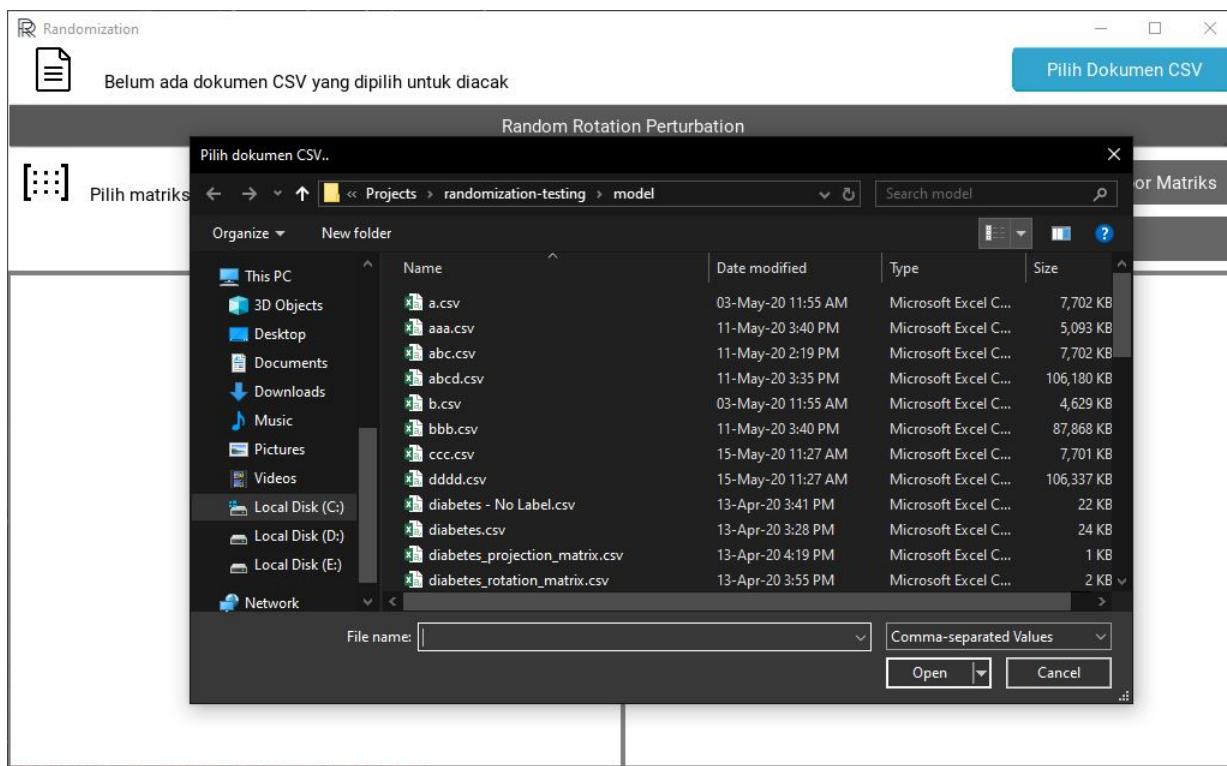
- 1 • Membuat baru dan memilih matriks rotasi atau proyeksi yang ingin digunakan.
- 2 • Masukan nilai variabel *epsilon* dan nilai variabel *k* untuk teknik *Random Projection Perturbation*.
- 3 • Sebuah tombol untuk menerapkan teknik *Randomization* dan menyimpan hasilnya.
- 4 Berikut akan dijelaskan secara rinci dengan gambar setiap fungsi tersebut yang dapat dilihat pada
- 5 Gambar 5.3 dan cara pemakaiannya yang benar secara berturut.

## 7 Masukan *Dataset*

Pertama pengguna perlu memberikan masukan *dataset* yang ingin diacak berupa dokumen berjenis *comma-separated values*. Perangkat lunak menyediakan fitur tersebut yang dapat dilihat pada Gambar 5.3 yang terdapat pada bagian yang dikelilingi kotak berwarna merah dan bermomor satu. Pengguna dapat menekan tombol “Pilih Dokumen CSV” yang terletak pada ujung sebelah kanan. Tombol ini bertujuan untuk memilih dokumen yang ingin diacak pada direktori pengguna. Ketika tombol ditekan, perangkat lunak akan membuka jendela baru untuk memilih dokumen yang dapat dilihat pada gambar 5.4.

Setelah pengguna memilih *dataset* yang diinginkan, perangkat lunak akan otomatis menuliskan lokasi dokumen yang dipilih berada. Perangkat lunak akan menampilkan lokasi dokumen tersebut pada bagian tengah sebelah kanan simbol dokumen dan sebelah kiri tombol “Pilih Dokumen CSV”. Jika belum ada *dataset* yang dipilih maka perangkat lunak akan menampilkan label yang berupa kalimat “Belum ada dokumen CSV yang dipilih untuk diacak” yang menunjukkan bahwa belum ada dokumen yang dipilih oleh pengguna. Jika pengguna memilih ulang dokumen, maka secara otomatis juga perangkat lunak akan memperbarui lokasi dokumen sesuai dokumen yang dipilih pengguna.

Apabila dokumen yang dipilih berukuran besar, maka perangkat lunak akan memakan sedikit waktu yang lebih lama. Dalam rangka memberitahukan kepada pengguna bahwa perangkat lunak sedang melakukan proses pemilihan dokumen, perangkat lunak akan menampilkan sebuah *popup* yang memberitahukan bahwa proses pemilihan sedang berjalan dan perangkat lunak tidak berhenti bekerja atau ada kesalahan sehingga pengguna tidak bingung apabila perangkat lunak memakan waktu yang lebih lama untuk memproses dokumen yang dipilih. Tampilan antarmuka *popup* tersebut dapat dilihat pada Gambar 5.5. Setelah dokumen dipilih pengguna dan perangkat lunak berhasil memproses dokumen tersebut, perangkat lunak akan memperbarui lokasi dokumen dan menampilkan beberapa informasi *dataset* yang dipilih pada bagian deskripsi *dataset* yang akan



Gambar 5.4: Jendela untuk memilih *dataset* yang berupa dokumen CSV

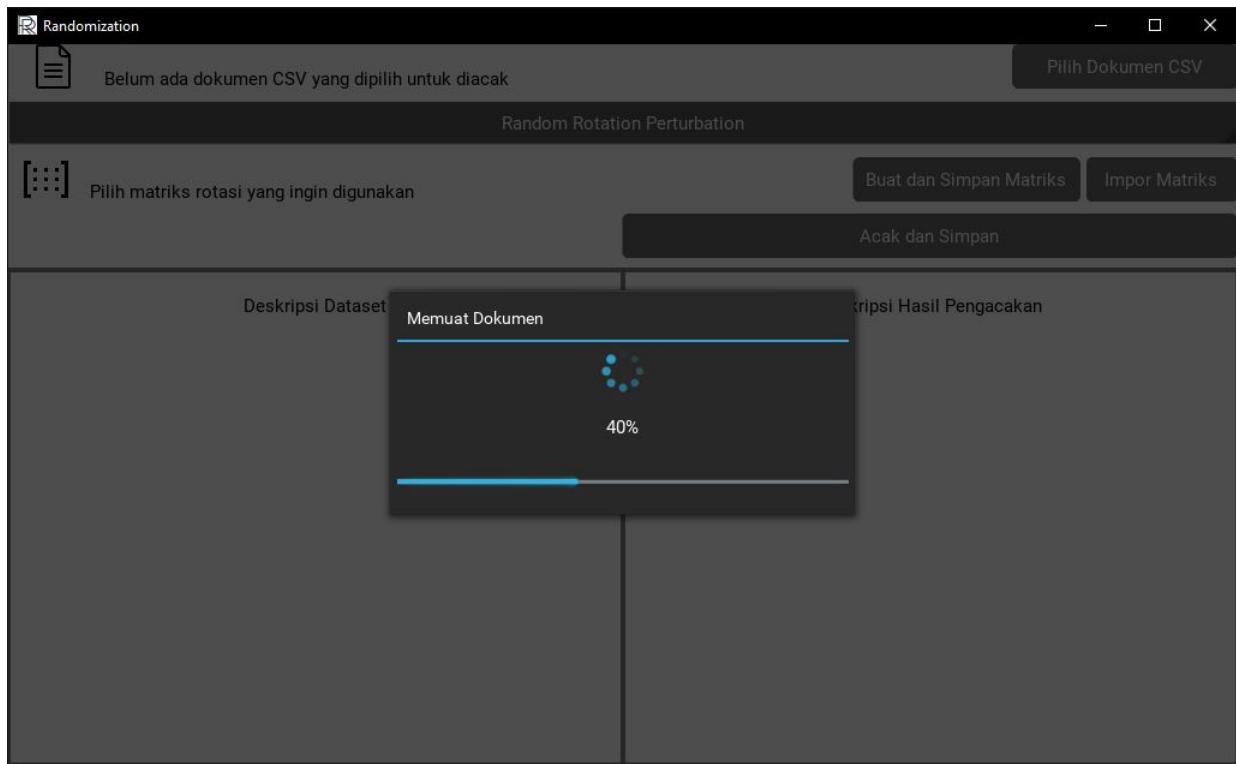
1 dijelaskan pada subbab berikutnya. Tampilan antarmuka setelah pengguna memilih dokumen dapat  
2 dilihat pada Gambar 5.6

3 Selain itu setelah pengguna memilih dokumen CSV, perangkat lunak akan membaca dokumen  
4 tersebut dan memproses isi dari dokumen tersebut menjadi *dataset* yang berupa matriks. Proses  
5 ini dilakukan sekali saja tepat setelah pengguna memilih dokumen dengan menekan tombol “Pilih  
6 Dokumen CSV”. Oleh karena itu, apabila sebuah dokumen CSV diubah isinya setelah dokumen  
7 tersebut dipilih oleh pengguna maka perangkat lunak tetap akan menggunakan isi dari dokumen  
8 tersebut yang belum diubah. Pengguna harus berhati-hati apabila isi dokumen diubah maka  
9 pengguna juga harus memilih kembali dokumen yang sama tersebut walaupun perangkat lunak  
10 sudah menunjukkan lokasi dokumen yang digunakan adalah dokumen yang pengguna inginkan.

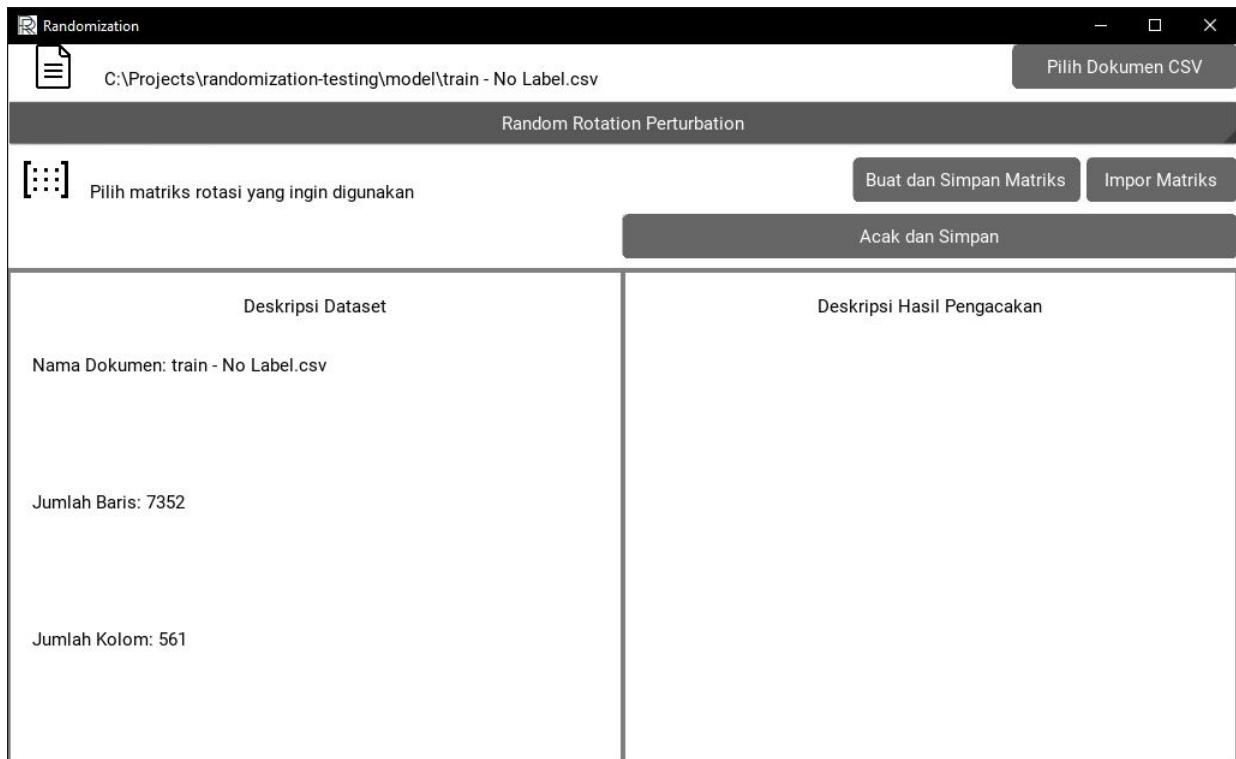
### 11 **Pemilihan teknik *Randomization***

12 Setelah pengguna memilih *dataset* yang ingin diacak, pengguna juga harus memilih teknik *Randomi-*  
13 *zation* apa yang ingin diterapkan terhadap *dataset* yang sudah dipilih. Pada awal perangkat lunak  
14 dibuka, secara otomatis teknik *Random Rotation Perturbation* yang dipilih. Apabila pengguna ingin  
15 mengganti teknik yang ingin diterapkan pada *dataset*, pengguna dapat menekan tombol *dropdown*  
16 yang bertuliskan nama teknik *Randomization*. Tombol ini dapat dilihat pada Gambar 5.3 yang  
17 dikelilingi kotak berwarna hijau dan bermotor dua.

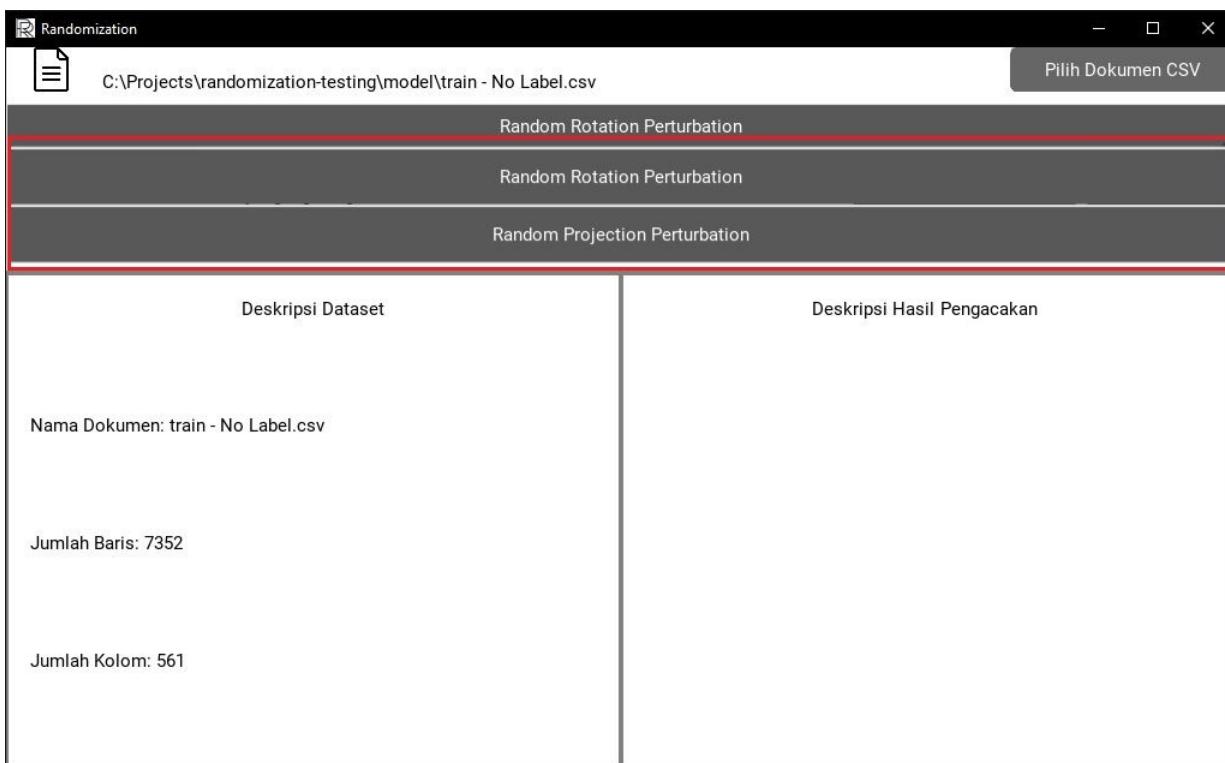
18 Apabila pengguna menekan tombol ini maka perangkat lunak akan menampilkan *dropdown*  
19 yang mempunyai dua buah opsi teknik *Randomization* yaitu “Random Rotation Perturbation” dan  
20 “Random Projection Perturbation”. Antarmuka tersebut dapat dilihat pada Gambar 5.7 yang  
21 dikelilingi oleh kotak merah. Pemilihan teknik ini juga akan memicu beberapa perubahan pada



Gambar 5.5: Tampilan *popup* yang ditampilkan saat proses memuat dokumen berlangsung



Gambar 5.6: Tampilan antarmuka setelah sebuah dokumen dipilih



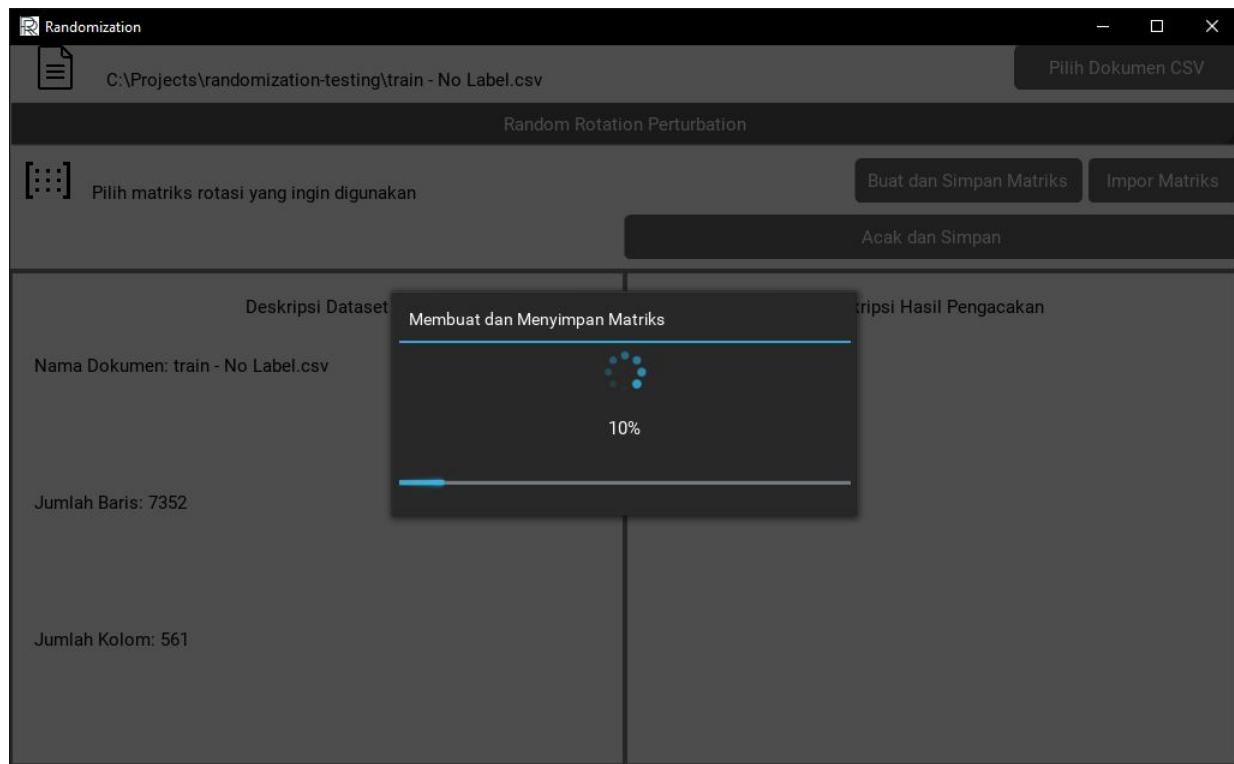
Gambar 5.7: Tampilan antarmuka saat pengguna memilih teknik

- 1 tampilan antarmuka perangkat lunak menyesuaikan dengan teknik yang dipilih. Beberapa perubahan
- 2 pada perangkat lunak tersebut melingkupi bagian pembuatan dan pemilihan matriks, parameter
- 3 teknik *Randomization*, dan bagian pengacakan dan simpan yang akan dijelaskan setiap perubahan
- 4 tersebut pada subbab berikutnya.

## 5 Pembuatan dan Pemilihan Matriks

- 6 Setelah pengguna memilih teknik yang ingin diterapkan, pengguna harus memilih matriks yang
- 7 diinginkan atau membuat baru. Matriks yang dimaksudkan adalah matriks rotasi atau matriks
- 8 proyeksi sesuai teknik *Randomization* yang dipilih. Apabila teknik *Random Rotation Perturbation*
- 9 yang dipilih maka perangkat lunak akan mengubah fungsi pembuatan dan pemilihan matriks
- 10 ini menjadi matriks rotasi. Apabila teknik *Random Projection Perturbation* yang dipilih maka
- 11 perangkat lunak akan mengubah fungsi pembuatan dan pemilihan matriks ini menjadi matriks
- 12 proyeksi. Perubahan ini dapat terlihat pada label yang berada di sebelah kanan simbol matriks
- 13 apabila belum memilih atau membuat matriks maka label tersebut akan menampilkan kalimat
- 14 “Pilih matriks rotasi yang ingin digunakan” atau “Pilih matriks proyeksi yang ingin digunakan”.
- 15 Bagian ini dapat dilihat pada Gambar 5.3 yang dikelilingi oleh kotak berwarna hijau dan bermotor
- 16 dua.

17 Ada dua buah tombol pada bagian ini yaitu “Buat dan Simpan Matriks” dan “Impor Matriks”.  
 18 Tombol “Buat dan Simpan Matriks” mempunyai fungsi untuk membuat matriks rotasi atau proyeksi  
 19 baru sesuai teknik *Randomization* yang dipilih dan menyimpan matriks tersebut pada sebuah  
 20 dokumen CSV baru yang dibuat oleh perangkat lunak pada direktori tertentu yang akan dipilih oleh  
 21 pengguna. Pada saat perangkat lunak sedang memproses matriks tersebut, perangkat lunak akan

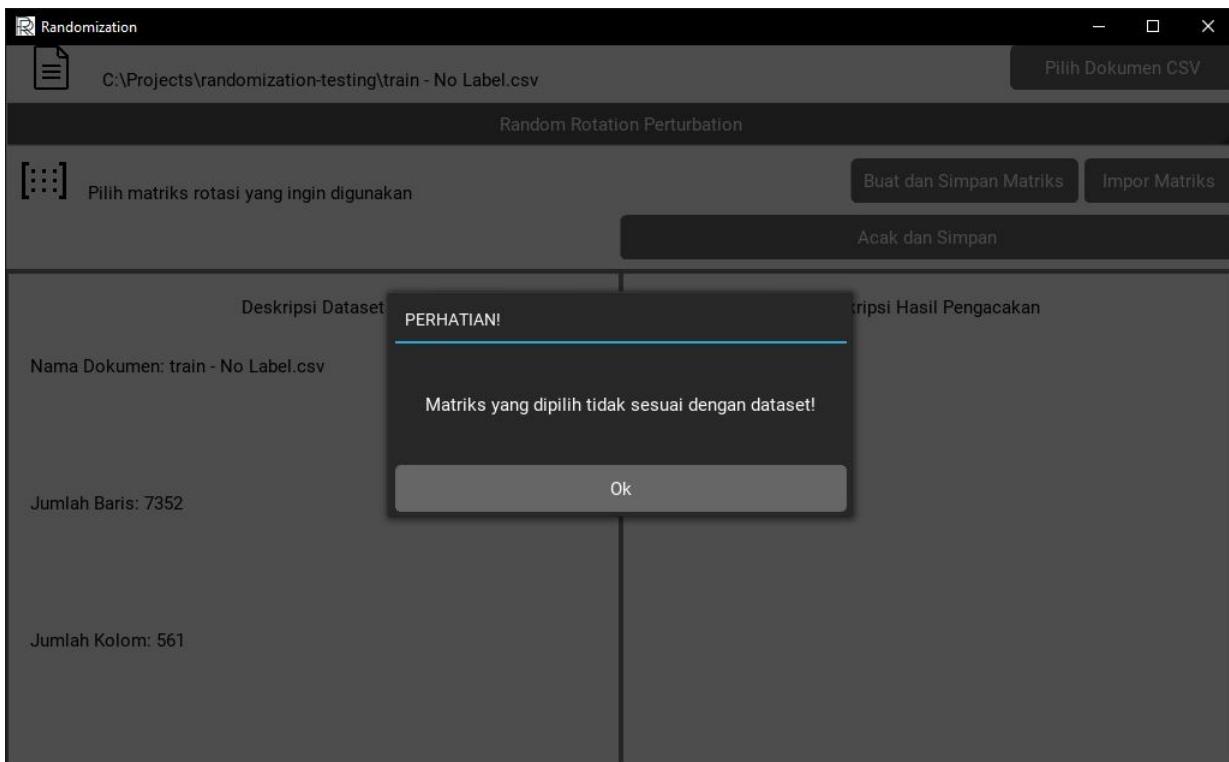


Gambar 5.8: Tampilan antarmuka saat perangkat lunak membuat dan menyimpan matriks

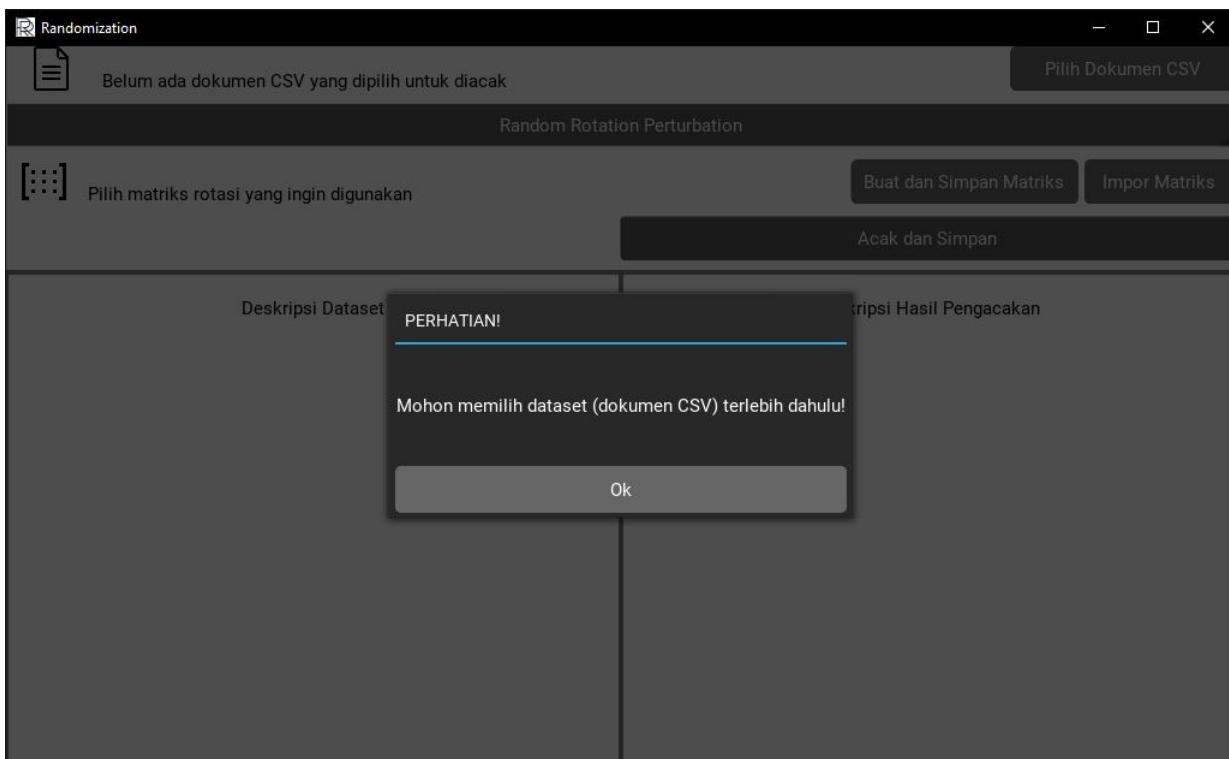
- 1 menampilkan *popup* memuat yang dapat dilihat pada Gambar 5.8. *Popup* ini juga akan tampil saat proses impor matriks dilakukan. Hasil matriks yang dibuat oleh perangkat lunak dapat digunakan kembali untuk lain kali sehingga rotasi atau proyeksi yang diterapkan akan sama dengan yang sebelumnya.

5 Pengguna dapat melakukan impor matriks dengan cara menekan tombol “Impor Matriks” untuk 6 memilih matriks rotasi atau proyeksi yang diinginkan untuk diterapkan pada *dataset*. Matriks 7 yang dipilih harus sesuai dengan *dataset* yang ingin diacak, misalnya apabila matriks rotasi yang 8 dipilih memiliki dimensi yang berbeda dengan *dataset* maka perangkat lunak akan melarang impor 9 matriks dilakukan karena pengacakan tidak dapat dilakukan. Perangkat lunak akan menampilkan 10 *popup* peringatan untuk pengguna yang dapat dilihat pada Gambar 5.9. Apabila pengguna memilih 11 teknik *Random Projection Perturbation* dan pengguna mengimpor matriks proyeksi maka parameter 12 variabel *k* akan terisi secara otomatis sesuai dengan matriks proyeksi yang diimporkan.

13 Apabila pengguna belum memilih *dataset* yang ingin diacak, pengguna tidak dapat membuat 14 maupun impor matriks terlebih dahulu. Hal ini dikarenakan perlu ada proses pengecekan terlebih 15 dahulu yang dilakukan perangkat lunak untuk memastikan *dataset* yang ingin diacak sudah sesuai 16 persyaratan dan sesuai dengan matriks yang akan dipilih. Perangkat lunak akan melarang pengguna 17 membuat maupun impor matriks dengan menampilkan sebuah *popup* peringatan yang dapat dilihat 18 pada Gambar 5.10. Pada teknik *Random Projection Perturbation*, pengguna baru bisa membuat 19 matriks proyeksi apabila sudah memenuhi persyaratan yang diminta yaitu mengisi parameter teknik 20 tersebut yang mana adalah variabel *epsilon* dan variabel *k*.



Gambar 5.9: Tampilan *popup* yang ditampilkan apabila matriks yang ingin diimpor tidak sesuai dengan *dataset*



Gambar 5.10: Tampilan *popup* peringatan apabila pengguna belum memilih *dataset* yang diinginkan untuk diacak



Gambar 5.11: Tampilan antarmuka parameter teknik *Random Projection Perturbation*

### <sup>1</sup> Parameter teknik *Randomization*

2 Perangkat lunak hanya meminta kepada pengguna parameter untuk teknik *Random Projection*  
 3 *Perturbation* saja apabila pengguna memilih teknik tersebut. Pada teknik *Random Rotation*  
 4 *Perturbation* tidak ada parameter yang perlu pengguna berikan. Ada dua buah parameter yang  
 5 perlu pengguna berikan yaitu variabel *epsilon* dan variabel *k*. Pengguna dapat memasukkan nilai  
 6 kedua variabel tersebut dengan menekan kolom variabel tersebut masing-masing. Kedua buah  
 7 parameter tersebut dapat dilihat antarmukanya pada Gambar 5.11

8 Seperti yang disinggung pada subbab sebelumnya antarmuka perangkat lunak akan menyesuaikan  
 9 secara otomatis sesuai teknik yang dipilih pengguna. Pada bagian parameter teknik *Randomization*,  
 10 perangkat lunak akan menyembunyikan antarmuka parameter *Random Projection Perturbation*  
 11 apabila pengguna memilih teknik *Random Rotation Perturbation*. Antarmuka tersebut dapat dilihat  
 12 pada Gambar 5.3 yang dikelilingi oleh kotak berwarna merah dan bermotor empat, dapat dilihat  
 13 tidak ada parameter apapun yang tampil apabila teknik *Random Rotation Perturbation* yang dipilih.

14 Selain dua buah parameter, pada bagian ini juga ada sebuah tombol yaitu "Hitung Nilai  
 15 Min K" yang memiliki fungsi untuk menghitung nilai minimal variabel *k* yang pengguna berikan.  
 16 Pada teknik *Random Projection Perturbation*, ada beberapa persyaratan yang harus dipenuhi oleh  
 17 pengguna dan salah satunya adalah variabel *k* yang diberikan harus melebihi sebuah nilai minimal  
 18 yang dihitung berdasarkan ukuran *dataset* dan nilai variabel *epsilon*. Oleh karena itu, sebelum  
 19 tombol ini dapat berfungsi, pengguna harus memilih terlebih dahulu *dataset* yang ingin diacak  
 20 dan memberikan masukan nilai variabel *epsilon* yang sesuai dengan persyaratan variabel *epsilon*  
 21 yaitu nilainya lebih besar dari 0 dan kurang dari 1. Apabila pengguna belum memenuhi kedua  
 22 persyaratan tersebut, tombol tidak akan berfungsi dan perangkat lunak akan menampilkan *popup*  
 23 peringatan yang dapat dilihat pada Gambar 5.12. Nilai minimal variabel *k* akan ditampilkan pada  
 24 bagian antarmuka deskripsi *dataset* yang akan dijelaskan pada subbab berikutnya.

25 Perangkat lunak juga akan memberikan peringatan apabila nilai minimal variabel *k* melebihi  
 26 besar dimensi *dataset* yang ingin diacak karena salah satu persyaratan dari teknik *Random Projection*  
 27 *Perturbation* adalah nilai variabel *k* harus lebih kecil daripada besar dimensi *dataset* yang ingin  
 28 diacak. Apabila pengguna melakukan impor matriks maka variabel *k* akan terisi secara otomatis  
 29 dan pengguna harus menyesuaikan nilai variabel *epsilon* dengan variabel *k* yang tidak boleh diubah  
 30 oleh pengguna.



Gambar 5.12: Tampilan *popup* peringatan tombol “Hitung Nilai Min K”

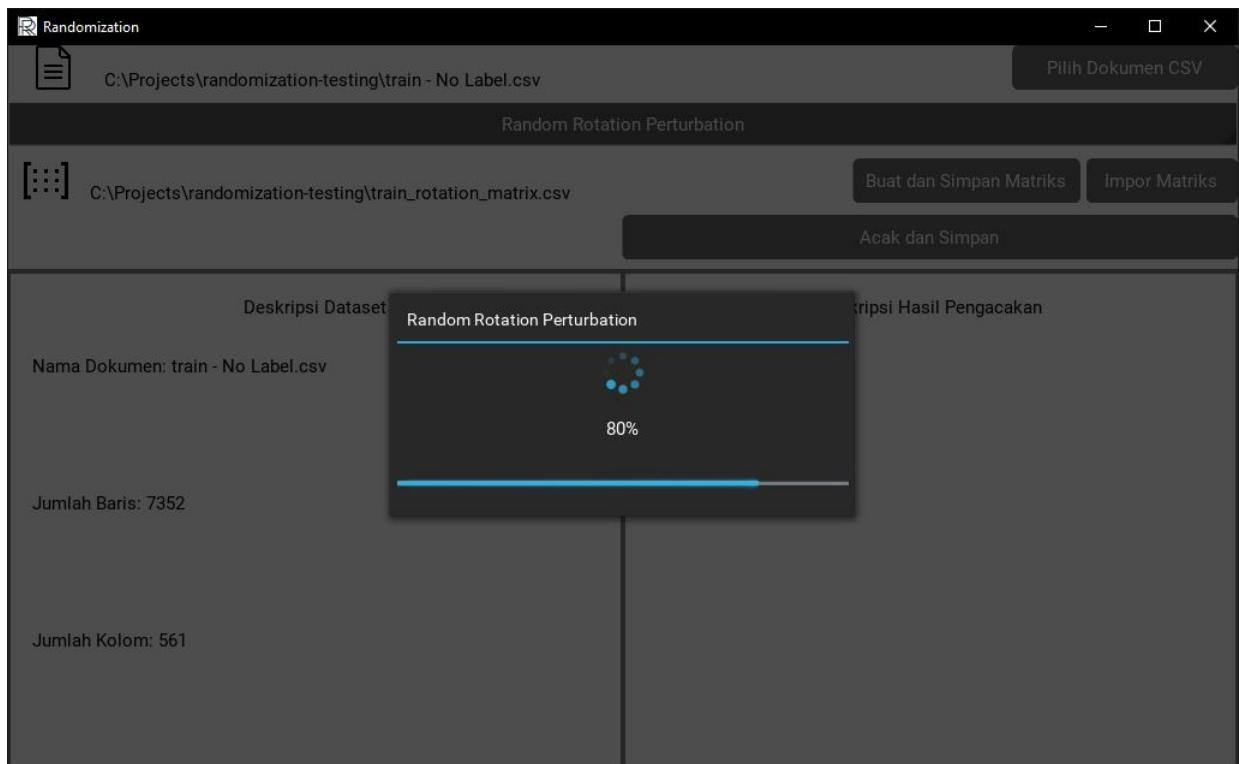
## <sup>1</sup> Pengacakan dan Simpan

<sup>2</sup> Setelah pengguna memberikan masukan yang sesuai dan mengatur pengaturan yang diinginkan maka  
<sup>3</sup> pengguna telah dapat melakukan pengacakan dengan menekan tombol “Acak dan Simpan”. Tombol  
<sup>4</sup> ini akan menerapkan teknik *Randomization* yang dipilih oleh pengguna terhadap *dataset* yang ingin  
<sup>5</sup> diacak menggunakan matriks yang telah dibuat atau dipilih oleh pengguna dan parameter-parameter  
<sup>6</sup> yang pengguna berikan. Tampilan antarmuka saat proses pengacakan dilakukan perangkat lunak  
<sup>7</sup> dapat dilihat pada Gambar 5.13.

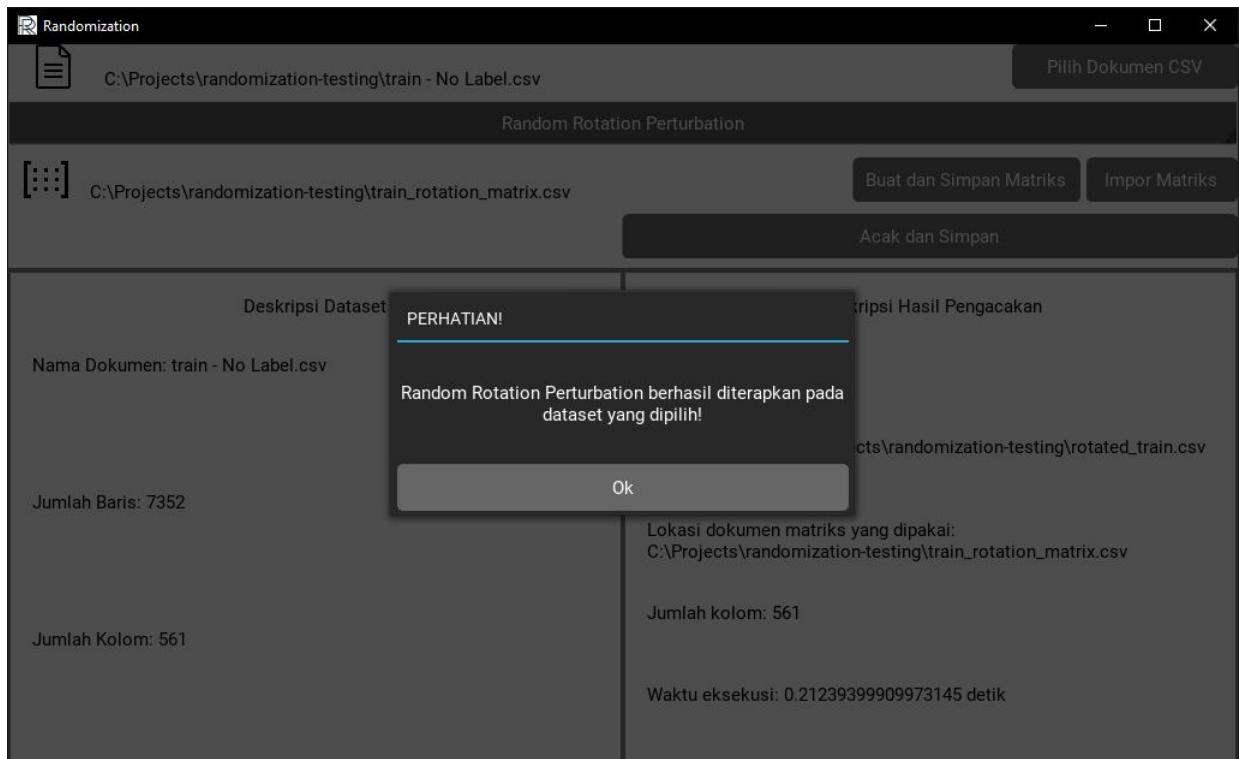
<sup>8</sup> Setelah perangkat lunak berhasil melakukan pengacakan, perangkat lunak akan meminta  
<sup>9</sup> pengguna untuk memilih direktori tempat penyimpanan dan nama dokumen hasil pengacakan.  
<sup>10</sup> Perangkat lunak akan menyimpan hasil pengacakan dalam bentuk dokumen berjenis *comma-*  
<sup>11</sup> *separated values*. Jendela baru untuk memilih direktori penyimpanan akan ditampilkan perangkat  
<sup>12</sup> lunak, apabila pengguna membatalkan atau dengan kata lain menutup jendela tersebut tanpa  
<sup>13</sup> memilih direktori penyimpanan maka perangkat lunak tidak akan melanjutkan proses pengacakan  
<sup>14</sup> dan dianggap gagal. Tampilan antarmuka *popup* yang akan tampil setelah perangkat lunak berhasil  
<sup>15</sup> melakukan proses pengacakan dan menyimpan hasilnya pada direktori yang pengguna pilih dapat  
<sup>16</sup> dilihat pada Gambar 5.14. Perangkat lunak juga akan menampilkan berbagai informasi hasil  
<sup>17</sup> pengacakan pada bagian antarmuka deskripsi hasil pengacakan yang akan dijelaskan pada subbab  
<sup>18</sup> berikutnya.

<sup>19</sup> Ada beberapa persyaratan yang harus dipenuhi oleh pengguna sebelum melakukan pengacakan  
<sup>20</sup> yaitu sebagai berikut.

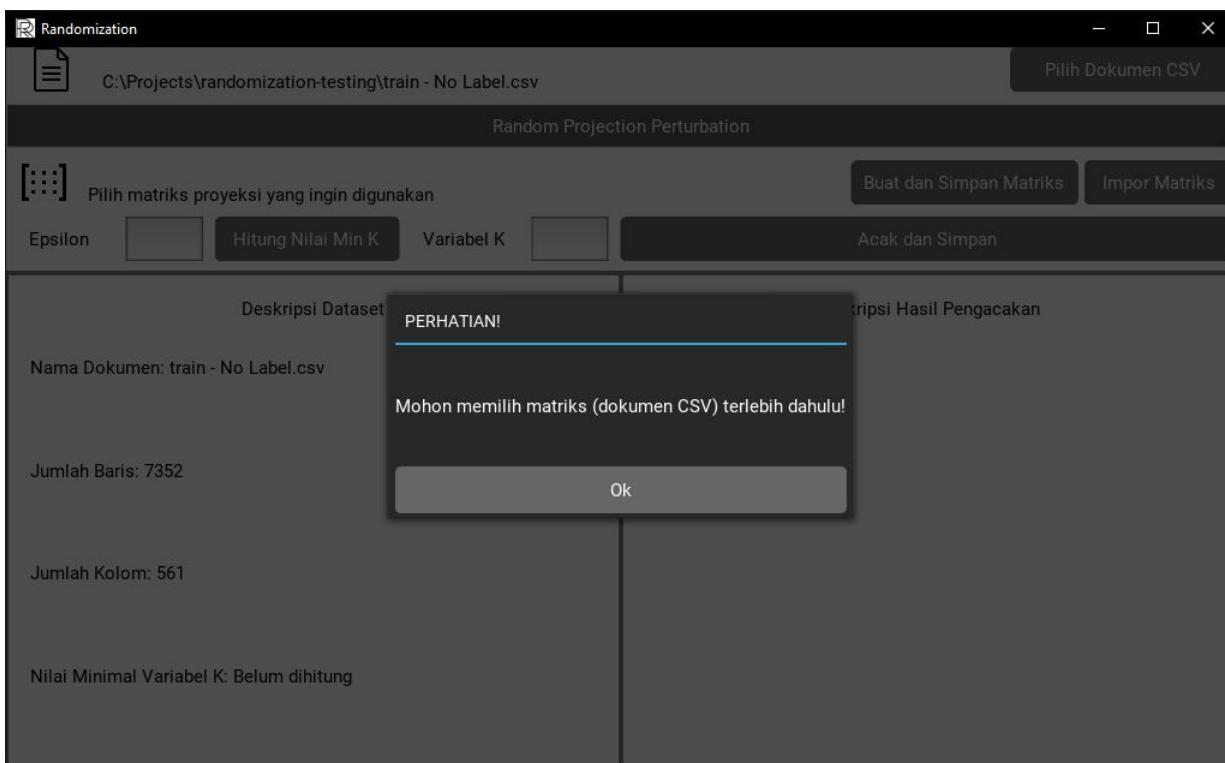
- <sup>21</sup> • Memilih *dataset* yang ingin diacak.
- <sup>22</sup> • Memilih teknik *Randomization* yang diinginkan.
- <sup>23</sup> • Membuat atau memilih matriks rotasi atau proyeksi.
- <sup>24</sup> • Memberikan masukan nilai variabel *epsilon* dan nilai variabel *k*.



Gambar 5.13: Tampilan saat perangkat lunak sedang melakukan proses pengacakan



Gambar 5.14: Tampilan *popup* untuk memberitahukan pengguna bahwa pengacakan berhasil dilakukan



Gambar 5.15: Tampilan *popup* peringatan apabila pengguna belum memilih atau membuat matriks rotasi/proyeksi

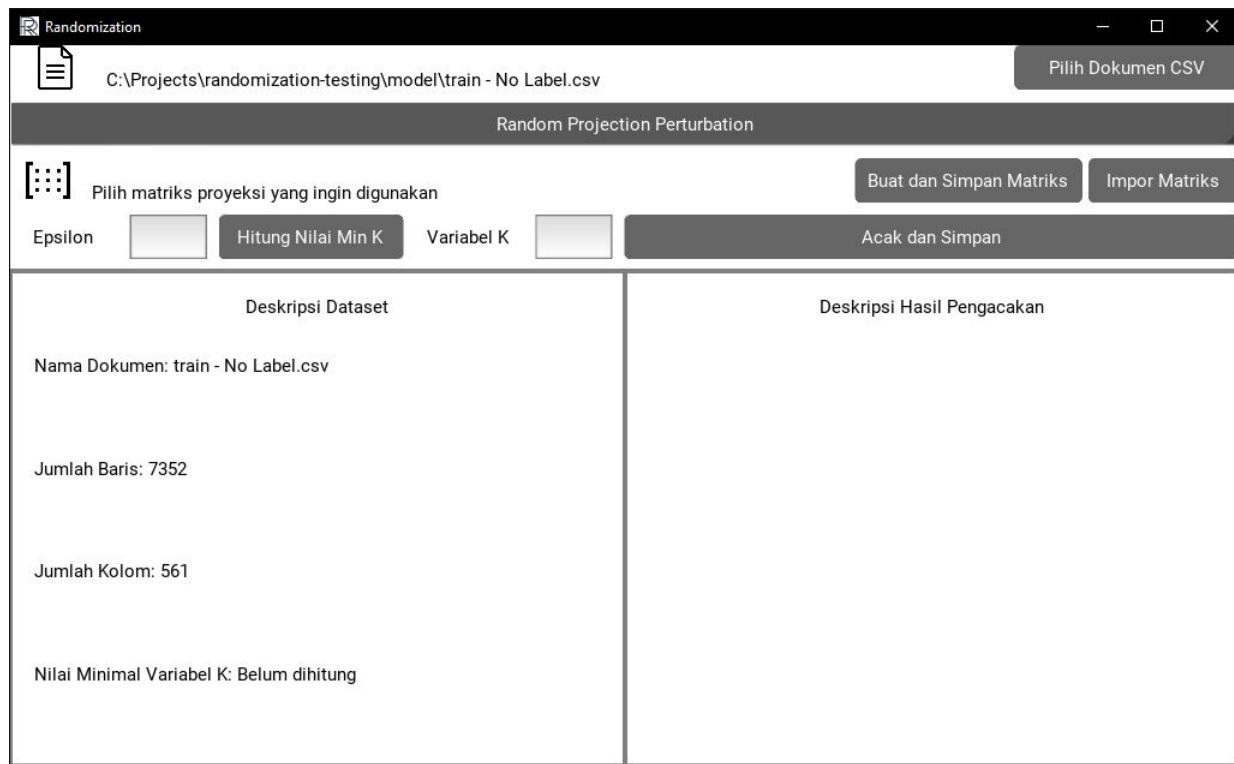
1 Persyaratan terakhir hanya berlaku apabila pengguna memilih teknik *Random Projection Perturbation*. Apabila ada persyaratan yang tidak dipenuhi oleh pengguna maka perangkat lunak  
2 akan menampilkan *popup* untuk memberikan peringatan kepada pengguna dan perangkat lunak  
3 tidak akan melanjutkan proses pengacakan. Perangkat lunak akan menampilkan *popup* peringatan  
4 terhadap pelanggaran masing-masing persyaratan tersebut, salah satu contoh tampilan antarmuka  
5 *popup* tersebut dapat dilihat pada Gambar 5.15.

### 7 5.1.2 Deskripsi *Dataset*

8 Pengguna dapat melihat berbagai informasi dokumen *comma-separated values* yang dipilih sebagai  
9 *dataset* yang ingin diacak pada bagian antarmuka deskripsi *dataset* yaitu sebagai berikut.

- 10 • Nama dokumen
- 11 • Jumlah baris *dataset*
- 12 • Jumlah kolom *dataset*
- 13 • Nilai minimal variabel *k*

14 Seperti yang disinggung pada subbab sebelumnya, pada awalnya nilai minimal variabel *k* belum  
15 diketahui karena belum dihitung. Pengguna harus mengisi variabel *epsilon* dan menekan tombol  
16 ‘‘Hitung Nilai Min K’’ agar perangkat lunak menghitung nilai minimal variabel *k* dan dapat  
17 menampilkannya pada deskripsi *dataset*. Nilai minimal variabel *k* hanya ditampilkan apabila  
18 pengguna memilih teknik *Random Projection Perturbation*.



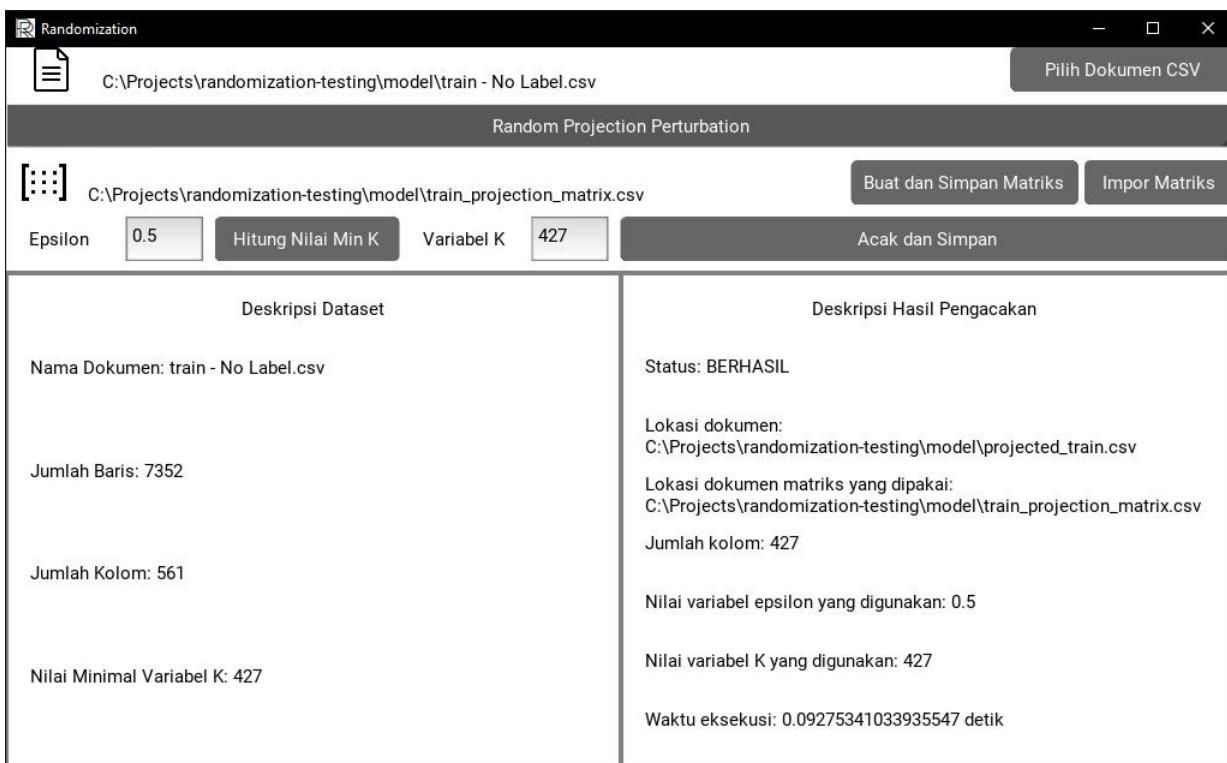
Gambar 5.16: Tampilan antarmuka deskripsi *dataset* setelah pengguna memilih *dataset* yang ingin diacak

- 1 Bagian antarmuka deskripsi *dataset* ini akan selalu secara otomatis diperbaharui setiap pengguna
- 2 memilih *dataset* baru. Tampilan antarmuka bagian deskripsi *dataset* dapat dilihat pada Gambar 5.2
- 3 yang dikelilingi oleh kotak berwarna biru dan bermotor dua. Apabila pengguna telah memilih
- 4 *dataset* yang diinginkan untuk diacak maka perangkat lunak secara otomatis akan memperbaharui
- 5 tampilan antarmuka deskripsi *dataset* yang dapat dilihat pada Gambar 5.16

### 6 5.1.3 Deskripsi Hasil Pengacakan

- 7 Perangkat lunak akan menampilkan isi dari deskripsi hasil pengacakan setelah pengguna menekan
- 8 tombol “Acak dan Simpan” dan perangkat lunak melakukan proses pengacakan. Bagian deskripsi
- 9 hasil pengacakan ini akan menampilkan informasi sebagai berikut.

- 10
  - Status
- 11
  - Lokasi dokumen
- 12
  - Lokasi dokumen matriks yang dipakai
- 13
  - Jumlah kolom
- 14
  - Waktu eksekusi
- 15
  - Nilai variabel *epsilon* yang digunakan
- 16
  - Nilai variabel *k* yang digunakan



Gambar 5.17: Tampilan antarmuka deskripsi hasil pengacakan setelah perangkat berhasil melakukan pengacakan

- <sup>1</sup> Nilai variabel *epsilon* dan nilai variabel *k* hanya ditampilkan apabila pengguna memilih teknik *Randomization Random Projection Perturbation*.
- <sup>3</sup> Bagian antarmuka deskripsi *dataset* ini akan selalu secara otomatis diperbaharui setiap pengguna memilih *dataset* baru. Tampilan antarmuka bagian deskripsi *dataset* dapat dilihat pada Gambar 5.2 yang dikelilingi oleh kotak berwarna biru dan bermotor dua. Apabila pengguna telah memilih *dataset* yang diinginkan untuk diacak maka perangkat lunak secara otomatis akan memperbaharui tampilan antarmuka deskripsi *dataset* yang dapat dilihat pada Gambar 5.17

## <sup>8</sup> 5.2 Pengujian Fungsional

- <sup>9</sup> Pengujian fungsional bertujuan untuk memastikan perangkat lunak *Randomization* dapat mengeksekusi kedua teknik *Randomization* yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation* dengan baik terhadap *dataset* yang memenuhi syarat. Proses pengujian akan dilakukan dengan cara menerapkan kedua teknik tersebut dari awal memasukkan *dataset* sampai menghasilkan *dataset* yang telah diacak dan menguji keluaran perangkat lunak tersebut. Berikut pengujian fungsional pada setiap teknik *Randomization*.

### <sup>15</sup> 5.2.1 Teknik *Random Rotation Perturbation*

- <sup>16</sup> Pengujian teknik *Random Rotation Perturbation* akan menggunakan *dataset* *mall\_customers*<sup>1</sup> yang berisi informasi pribadi pelanggan sebuah mall. *Dataset* ini memiliki 3 buah fitur (bersifat

<sup>1</sup><https://www.kaggle.com/vjchoudhary7/customer-segmentation-tutorial-in-python>

0	1	2	0	1	2	3
-0.27377	0.743342	-0.61032	1	0	0	0
0.853611	-0.1046	-0.5103	0	1	0	0
-0.44317	-0.66068	-0.60589	0	0	1	0
			9	96	88	1

Gambar 5.18: Matriks rotasi dan translasi acak yang dibuat perangkat lunak dan disimpan pada satu dokumen *comma-separated values*

1 numerik) yaitu umur, penghasilan, dan skor pengeluaran. Tiga buah fitur tersebut akan diacak dan  
 2 diharapkan hasilnya akan mengacak *dataset* sehingga nilai tiap fitur tersebut berbeda dari aslinya.  
 3 Selain itu, matriks rotasi harus dapat disimpan dan digunakan kembali untuk lain kali. *Dataset*  
 4 yang telah diacak juga diharapkan masih dapat diterapkan teknik penambangan data dengan hasil  
 5 yang sama dengan *dataset* asli.

6 Pada teknik *Random Rotation Perturbation* untuk merotasikan sebuah *dataset* diperlukan  
 7 sebuah matriks rotasi dan translasi acak. Perangkat lunak diharapkan dapat membuat kedua  
 8 matriks tersebut dan menyimpannya ke dalam sebuah dokumen *comma-separated values*. Beberapa  
 9 kolom pada *dataset mall\_customers* harus dihilangkan terlebih dahulu agar hanya fitur (bersifat  
 10 numerik) pada *dataset* tersebut saja yang teracak. *Dataset mall\_customers* mempunyai 3 buah fitur  
 11 yang akan diacak, oleh karena itu matriks rotasi dan translasi acak yang dibuat perangkat lunak  
 12 seharusnya berukuran  $3 \times 3$  kecuali matriks translasi yang akan berukuran  $4 \times 4$  karena matriks  
 13 translasi mempunyai kolom terakhir tambahan yang dibutuhkan untuk melakukan transformasi  
 14 translasi.

15 Pada pengujian yang dilakukan, perangkat lunak berhasil membuat kedua matriks acak tersebut  
 16 dengan ukuran yang benar dan menyimpannya pada satu dokumen *comma-separated values* yang  
 17 isinya dapat dilihat pada Gambar 5.18. Matriks rotasi dan translasi ini akan digunakan untuk  
 18 menerapkan teknik *Random Rotation Perturbation* terhadap *dataset mall\_customers*. Perangkat  
 19 lunak juga terbukti berhasil menggunakan ulang kedua matriks yang telah disimpan tersebut untuk  
 20 menerapkan teknik *Random Rotation Perturbation* terhadap *dataset mall\_customers* dan hasilnya  
 21 sama persis seperti hasil yang pertama kali. Kedua matriks tersebut juga dapat digunakan untuk  
 22 data *mall\_customers* yang lain dengan syarat masih memiliki jumlah fitur yang sama.

23 Berikut akan ditampilkan 20 baris pertama *dataset* asli dan *dataset* yang telah diacak masing-  
 24 masing pada Gambar 5.19 dan Gambar 5.20. Dapat dilihat pada gambar tersebut, *dataset* setelah  
 25 diacak memiliki nilai yang sangat berbeda dengan aslinya. Terlebih lagi jika diperhatikan, nilai  
 26 yang sama pada beberapa baris di *dataset* asli tidak sama dengan *dataset* yang telah diacak pada  
 27 baris yang sama seperti pada kolom *Annual Income* baris 10 sampai 12 memiliki nilai 19 pada  
 28 *dataset* asli tetapi pada *dataset* yang telah diacak ketiga baris tersebut memiliki nilai yang berbeda  
 29 antara satu dengan yang lainnya.

Dalam rangka untuk memastikan perangkat lunak berhasil dengan benar menerapkan teknik *Random Rotation Perturbation* dengan matriks rotasi dan translasi yang sebelumnya sudah dibuat dan disimpan oleh perangkat lunak, perhitungan manual dilakukan terhadap *dataset mall\_customers* dengan menggunakan matriks rotasi dan translasi tersebut yang dapat dilihat pada Gambar 5.18.

CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
11	Male	67	19	14
12	Female	35	19	99
13	Female	58	20	15
14	Female	24	20	77
15	Male	37	20	13
16	Male	22	20	79
17	Female	35	21	35
18	Male	20	21	66
19	Male	52	23	29
20	Female	35	23	98

Gambar 5.19: Dua puluh baris pertama *dataset mall\_customers* asli

CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
1	Male	30.80293576	-74.70371227	-150.6802393
2	Male	11.64235717	-100.965712	-177.34819
3	Female	46.00730281	-52.26243313	-131.8065483
4	Female	13.72109486	-96.94089629	-176.6555807
5	Female	28.78173046	-66.65349299	-159.6305856
6	Female	15.29164496	-97.12815467	-175.9496722
7	Female	43.6079501	-41.3215056	-141.9819702
8	Female	7.89446913	-108.3817053	-187.9762839
9	Male	37.85168599	-17.88714487	-158.3739244
10	Female	16.58136376	-88.74788781	-179.4292919
11	Male	32.15552849	-22.92463185	-166.8696594
12	Female	3.246982854	-102.8696182	-198.8398822
13	Female	35.0299167	-30.37999029	-162.4929572
14	Female	16.86176701	-96.61595268	-179.3071067
15	Male	41.66545616	-44.66880168	-148.4644332
16	Male	16.52297528	-99.4240021	-179.2982406
17	Female	33.31692523	-60.79510937	-161.0836341
18	Male	23.6853071	-92.42640702	-170.7113514
19	Male	33.02903428	-44.40340067	-168.8443689
20	Female	7.10459415	-102.6273334	-200.2751986

Gambar 5.20: Dua puluh baris pertama *dataset mall\_customers* setelah diacak

Tabel 5.1: Contoh perbedaan jarak Euclidean antara *dataset diabetes* asli dan yang telah diacak dengan *Random Rotation Perturbation*

<b>Baris</b>	<b>Baris</b>	<b>Dataset Asli</b>	<b>Dataset Diacak</b>
194	199	77.07788269017254	77.07788261888021
194	200	26.5329983228432	26.53299827877461
195	195	0.0	0.0
195	196	64.13267497929586	64.13267498269495
195	197	13.564659966250536	13.564659906259806

Hasil akhir dari perangkat lunak sama persis dengan hasil dari perhitungan manual, berikut akan ditunjukkan contoh perhitungan manual untuk baris pertama kolom *Age*. Hasil translasi dapat dilihat pada Persamaan 5.1 dengan variabel  $x$  adalah baris pertama kolom *Age*,  $y$  adalah baris pertama kolom *Annual Income*, dan  $z$  adalah baris pertama kolom *Spending Score*.

$$\begin{aligned}
 x &= 19 \times 1 + 15 \times 0 + 39 \times 0 + 1 \times 9 & (5.1) \\
 &= 28 \\
 y &= 19 \times 0 + 15 \times 1 + 39 \times 0 + 1 \times 96 \\
 &= 111 \\
 z &= 19 \times 0 + 15 \times 0 + 39 \times 1 + 1 \times 88 \\
 &= 127
 \end{aligned}$$

Setelah ditranslasi, transformasi rotasi dilakukan dan hasil akhir dapat dilihat pada Persamaan 5.2.

$$\begin{aligned}
 result &= 28 \times -0.27377 + 111 \times 0.853611 + 127 \times -0.44317 & (5.2) \\
 &= -7.66556 + 94.750821 - 56.28259 \\
 &= 30.802671
 \end{aligned}$$

- 1 Dapat dibandingkan hasil akhir perhitungan manual dengan hasil akhir perangkat lunak pada
- 2 Gambar 5.20, baris pertama pada kolom *Age* memiliki nilai yang sama persis sampai 3 angka di
- 3 belakang koma (dikarenakan pembulatan pada matriks rotasi untuk visualisasi di dokumen skripsi
- 4 ini yang dapat dilihat pada Gambar 5.18).
- 5 Pengujian terhadap jarak Euclidean antara seluruh objek data pada *dataset* asli dan yang
- 6 telah diacak juga dilakukan. Pada Tabel 5.1 dapat dilihat sebagian kecil keluaran dari perangkat
- 7 lunak pengujian yang merupakan contoh perbedaan jarak Euclidean antara sebuah baris yang
- 8 merepresentasikan sebuah objek data pada *dataset* dengan objek data lainnya. Seluruh objek data
- 9 pada *dataset* terbukti memiliki jarak Euclidean yang sama pada *dataset* asli dan *dataset* yang
- 10 telah diacak. Pengujian perbandingan jarak Euclidean dilakukan hanya memakai nilai pembulatan
- 11 sampai 2 atau 3 angka di belakang koma saja karena tidak terlalu relevan untuk mengambil seluruh
- 12 angka di belakang koma. Berdasarkan pengujian yang telah dilakukan, dapat disimpulkan bahwa
- 13 perangkat lunak sudah berfungsi dengan baik dan benar dalam menerapkan teknik *Random Rotation*
- 14 *Perturbation*.

### 1 5.2.2 Teknik *Random Projection Perturbation*

2 Pengujian teknik *Random Projection Perturbation* akan menggunakan *dataset mobile\_sensor*<sup>2</sup>  
 3 yang berisi data sensor *smartphone* banyak orang yang sedang melakukan aktivitas tertentu yaitu  
 4 berdiri, duduk, berbaring, berjalan, berjalan menanjak, berjalan menurun. *Dataset* ini memiliki 561  
 5 buah fitur (bersifat numerik) dan sebuah label. Seluruh fitur tersebut akan diacak dan diharapkan  
 6 hasilnya akan mengacak *dataset* sehingga nilai tiap fitur tersebut berbeda dari aslinya. *Dataset*  
 7 yang telah diacak juga diharapkan masih dapat diterapkan teknik penambangan data dengan hasil  
 8 yang mirip dengan *dataset* asli.

Beberapa kolom seperti label pada *dataset mobile\_sensor* yang ingin diacak harus dihilangkan terlebih dahulu agar hanya fitur (bersifat numerik) pada *dataset* tersebut saja yang teracak. Nilai variabel *epsilon* yang dipilih pada pengujian ini adalah sebesar 0.52 dan dengan nilai variabel *epsilon* tersebut nilai minimal variabel *k* adalah sebesar 418.0905. Pada pengujian ini perangkat lunak berhasil menghitung dengan benar nilai minimal variabel *k* yaitu sebesar 418 dengan *dataset mobile\_sensor* dan nilai variabel *epsilon* sebesar 0.52. Dapat dilihat Persamaan 5.3 adalah contoh perhitungan manual nilai minimal variabel *k* dengan nilai variabel *epsilon* sebesar 0.52 dan jumlah baris pada *dataset* sebanyak 10299 baris.

$$\begin{aligned} k &\geq \frac{4 \ln n}{\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}} \\ &\geq \frac{4 \ln 10299}{\frac{0.52^2}{2} - \frac{0.52^3}{3}} \\ &\geq \frac{36.9592}{0.1352 - 0.0468} \\ &\geq 418.0905 \end{aligned} \tag{5.3}$$

9 Pada teknik *Random Projection Perturbation* untuk memproyeksikan sebuah *dataset* diperlukan  
 10 sebuah matriks proyeksi acak. Perangkat lunak diharapkan dapat membuat sebuah matriks  
 11 proyeksi acak dan menyimpannya ke dalam sebuah dokumen *comma-separated values*. *Dataset*  
 12 *mobile\_sensor* memiliki 561 buah fitur yang akan diacak dan sesuai dengan nilai minimal variabel *k*  
 13 yang sudah dihitung sebelumnya yaitu sebesar 418 maka ditentukan nilai variabel *k* yang dipakai  
 14 pada pengujian ini adalah sebesar 427. Oleh karena itu, matriks proyeksi acak yang dibuat perangkat  
 15 lunak seharusnya berukuran  $427 \times 427$ .

16 Pada pengujian yang dilakukan, perangkat lunak berhasil membuat matriks proyeksi acak  
 17 tersebut dengan ukuran yang benar dan menyimpannya pada sebuah dokumen *comma-separated*  
 18 *values* yang 14 kolom pertama pada 20 baris pertamanya dapat dilihat pada Gambar 5.21. Matriks  
 19 proyeksi ini akan digunakan untuk menerapkan teknik *Random Projection Perturbation* terhadap  
 20 *dataset mobile\_sensor*. Perangkat lunak juga terbukti berhasil menggunakan ulang matriks yang  
 21 telah disimpan tersebut untuk menerapkan teknik *Random Projection Perturbation* terhadap *dataset*  
 22 *mobile\_sensor* dan hasilnya sama persis seperti hasil yang pertama kali. Matriks proyeksi tersebut  
 23 juga dapat digunakan untuk data *mobile\_sensor* yang lain dengan syarat masih memiliki jumlah  
 24 fitur yang sama dan dengan penambahan data tersebut, nilai minimal variabel *k* harus tetap lebih  
 25 kurang atau sama dengan nilai variabel *k* yang dipilih. Persyaratan tersebut didasarkan oleh sifat

<sup>2</sup><https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones>

0	1	2	3	4	5	6	7	8	9	10	11	12	13
-0.03276	0.04011	-0.01478	0.01277	0.04694	0.03569	-0.02266	-0.07329	-0.00297	0.01155	0.01492	-0.00755	-0.02141	-0.02183
0.02182	-0.02083	0.02094	-0.03843	-0.03965	0.00785	-0.04395	-0.00083	-0.0051	-0.01889	0.05269	0.06792	0.00704	-0.00355
-0.05636	0.03146	0.00636	-0.03852	-0.08686	-0.02119	0.05541	-0.01802	0.00713	0.08346	0.00546	0.00958	-0.04809	-0.04278
-0.01216	0.0138	-0.04911	0.00955	0.03787	-0.05147	-0.06889	0.03739	-0.03441	0.02876	0.07051	-0.04776	-0.02648	-0.00577
0.04679	-0.0088	0.11821	-0.0284	0.04825	0.03611	0.10739	-0.0315	-0.00811	0.00546	0.10633	0.07007	-0.06928	-0.03557
0.01889	0.00222	-0.03215	0.07379	-0.01914	-0.0313	-0.00257	0.07686	0.06301	-0.01889	-0.07781	-0.03361	-0.04044	0.1252
-0.03261	-0.01111	-0.05009	0.00594	-0.00875	0.03797	0.04032	0.03031	0.01071	0.00685	-0.0172	-0.10396	-0.04537	0.0329
-0.03494	0.08604	0.00615	0.04936	-0.00705	0.00161	0.06081	0.02996	-0.02234	0.05149	-0.0299	0.00458	0.04643	0.03377
0.01214	-0.03205	-0.05187	0.02922	0.02763	0.07964	0.00415	0.03845	0.00205	-0.00182	-0.05704	-0.02609	-0.02879	-0.02153
-0.02034	-0.02208	0.08089	-0.02807	-0.0705	-0.01705	-0.02844	0.00082	0.0085	-0.03901	0.04831	-0.05471	0.10369	-0.11468
-0.02321	0.10714	-0.04824	-0.06769	0.00293	0.04747	0.00726	-0.06324	0.00995	-0.05154	-0.09879	0.0256	-0.00708	-0.02562
0.00229	-0.00249	0.06524	-0.04228	0.00964	0.03129	0.07719	0.03581	0.06612	-0.0307	0.01605	-0.01314	0.03646	0.01542
-0.03059	0.02674	0.0537	-0.04237	-0.0152	-0.00295	0.00916	-0.01843	0.02631	-0.03991	0.02444	0.10179	0.00423	0.02668
-0.06482	0.04987	-0.05252	0.07486	-0.03795	0.03784	0.07387	0.00514	0.02794	-0.02625	0.04185	0.04578	0.01185	0.02188
0.10567	-0.02697	0.08684	-0.07994	-0.06175	0.02735	-0.07561	-0.05315	-0.00534	0.01238	0.0801	0.00761	-0.01669	0.09002
0.06459	-0.01561	0.08549	0.00325	0.01512	-0.01851	0.0423	-0.01951	0.01708	0.05016	-0.02286	-0.01588	0.0199	0.06935
0.06138	-0.07631	0.01451	0.0127	0.04138	-0.02069	0.06849	0.08379	0.00195	-0.00422	0.02668	-0.00234	-0.11837	0.03232
0.03948	0.00755	-0.07557	-0.02994	0.01807	-0.0454	-0.04669	0.02799	0.00699	-0.04009	0.02574	0.02861	0.01554	-0.08876
0.01711	0.01447	-0.03811	0.03356	0.11024	-0.04091	-0.00041	0.01896	-0.01741	-0.05211	0.03584	-0.0251	-0.06278	0.08278
0.01231	0.02438	0.04191	-0.0104	0.13033	-0.00214	-0.01177	0.09572	-0.01282	0.07462	0.01603	-0.05019	-0.05982	-0.03376

Gambar 5.21: Matriks proyeksi acak yang dibuat perangkat lunak dan disimpan pada sebuah dokumen *comma-separated values*

- 1 pada teknik *Random Projection Perturbation* yaitu nilai minimal variabel  $k$  berbanding lurus dengan  
 2 banyaknya objek data pada *dataset* yang ingin diacak.  
 3 Berikut akan ditampilkan 7 fitur terakhir serta label pada 20 baris pertama *dataset* asli dan  
 4 *dataset* yang telah diacak masing-masing pada Gambar 5.22 dan Gambar 5.23. Dapat dilihat pada  
 5 gambar tersebut, *dataset* setelah diacak memiliki nilai yang berbeda dengan aslinya. Terlebih lagi  
 6 setiap fitur pada *dataset* yang telah diacak tidak diketahui arti dari setiap fitur tersebut apa karena  
 7 sudah tereduksi sehingga seluruh fitur pada *dataset* asli tercampur secara acak dan terproyeksikan ke  
 8 dalam 427 fitur yang ada pada *dataset* yang telah diacak. Dalam rangka untuk memastikan perangkat  
 9 lunak berhasil dengan benar menerapkan teknik *Random Projection Perturbation* dengan matriks  
 10 proyeksi acak yang sebelumnya sudah dibuat dan disimpan oleh perangkat lunak, perhitungan  
 11 manual dilakukan terhadap *dataset mobile\_sensor* dengan menggunakan matriks proyeksi tersebut.  
 12 Hasil akhir dari perangkat lunak sama persis dengan hasil dari perhitungan manual. Oleh karena  
 13 itu, dapat disimpulkan bahwa perangkat lunak sudah berfungsi dengan baik dan benar dalam  
 14 menerapkan teknik *Random Projection Perturbation*.
- 15 Dalam rangka memastikan lebih lagi apakah perangkat lunak menerapkan teknik *Random*  
 16 *Projection Perturbation* dengan akurat, perangkat lunak pengujian dibuat untuk menguji jarak  
 17 Euclidean dari setiap titik pada *dataset* terhadap setiap titik lainnya. Pengujian dilakukan dengan  
 18 menggunakan pertidaksamaan rentang jarak Euclidean berikut untuk menguji apakah jarak Eucli-  
 19 dean pada *dataset* yang telah diacak mempunyai distorsi yang sesuai dengan harapan pengguna.  
 20 Hasil dari perangkat lunak (setiap jarak Euclidean antara sebuah objek data dengan objek data  
 21 yang lainnya pada *dataset* yang telah diacak) diharapkan memenuhi Pertidaksamaan 5.4.

$$(1 - \epsilon) \|u - v\|^2 < \|p(u) - p(v)\|^2 < (1 + \epsilon) \|u - v\|^2 \quad (5.4)$$

- 22 Pada pengujian ini, dengan nilai variabel *epsilon* sebesar 0.52 Pertidaksamaan 5.4 terpenuhi  
 23 pada seluruh objek data yang ada pada *dataset*. Pada Tabel 5.2 dapat dilihat sebagian kecil keluaran  
 24 dari perangkat lunak pengujian yang merupakan contoh perbedaan jarak Euclidean antara sebuah

angle(tBodyAcc)	angle(tBodyGyr)	angle(tBodyGyr)	angle(X,gravity)	angle(Y,gravity)	angle(Z,gravity)	Activity
0.030400372	-0.46476139	-0.018445884	-0.84124676	0.17994061	-0.058626924	STANDING
-0.007434566	-0.73262621	0.70351059	-0.8447876	0.18028889	-0.054316717	STANDING
0.17789948	0.10069921	0.80852908	-0.84893347	0.18063731	-0.049117815	STANDING
-0.012892494	0.64001104	-0.48536645	-0.84864938	0.18193476	-0.047663183	STANDING
0.12254196	0.69357829	-0.61597061	-0.84786525	0.18515116	-0.043892254	STANDING
-0.14343901	0.27504075	-0.36822404	-0.84963158	0.18482251	-0.042126383	STANDING
-0.23062193	0.01463669	-0.18951153	-0.85215025	0.18216997	-0.043009987	STANDING
0.59399581	-0.56187067	0.46738333	-0.85101671	0.18377851	-0.041975833	STANDING
0.080936389	-0.23431263	0.11779701	-0.84797148	0.18898248	-0.037363927	STANDING
-0.12773018	-0.48287054	-0.070670135	-0.84829438	0.19031033	-0.034417291	STANDING
0.59579055	-0.47580245	0.11593062	-0.85156175	0.18760932	-0.034681168	STANDING
-0.065980273	0.57886112	-0.65194513	-0.8527234	0.18605036	-0.035852089	STANDING
-0.10122189	0.63908399	0.76548488	-0.85065446	0.18761054	-0.035997955	STANDING
-0.090277545	-0.1324028	0.49881419	-0.84977267	0.1888122	-0.035063399	STANDING
-0.05871932	0.031207971	-0.26879133	-0.73093729	0.28315855	0.036443909	STANDING
-0.029076714	-0.013034217	-0.056927156	-0.76110079	0.26311858	0.02417211	STANDING
-0.048109773	-0.34047349	-0.22915457	-0.75917219	0.26432447	0.027014344	STANDING
0.092367048	-0.82223861	0.36755744	-0.75936327	0.26403279	0.029664019	STANDING
-0.033006915	-0.24057155	0.78819291	-0.76105187	0.26288599	0.029345734	STANDING
0.10256897	0.066134774	-0.41172948	-0.76062023	0.26316935	0.029573033	STANDING

Gambar 5.22: Dua puluh baris terakhir dataset *mobile\_sensor* yang asli

420	421	422	423	424	425	426	Activity
-0.060976974	-0.28720261	0.749987819	-0.314597972	0.862413791	-0.184264921	0.30054771	STANDING
-0.118304471	-0.367147618	0.968242315	-0.635217584	0.657331951	0.213839293	0.471794176	STANDING
-0.192180445	-0.31118681	0.978629636	-0.633964644	0.39068659	0.152568399	0.62044912	STANDING
0.025086451	-0.504801202	0.7858091	-0.907280086	0.869665518	0.3148417	0.468902009	STANDING
-0.102949398	-0.323268879	0.952716391	-0.899688251	0.629212652	0.223750593	0.684361819	STANDING
-0.147008408	-0.447760471	1.123065848	-0.637227607	0.67689574	0.228867562	0.747769951	STANDING
-0.163893961	-0.357135652	0.87956909	-0.938537783	0.615377996	0.334211022	0.427323639	STANDING
-0.247640685	-0.365958546	0.873331126	-0.736313605	0.73465538	0.324703704	0.46087866	STANDING
-0.102040114	-0.272211976	0.990322769	-0.927303583	0.551974442	0.130154901	0.945745752	STANDING
-0.040080611	-0.182285976	0.865031503	-0.913260332	0.679558281	0.227692766	0.44382714	STANDING
-0.048529732	-0.202250452	0.723406522	-0.960376711	0.583341661	0.331778206	0.516310888	STANDING
0.044383724	-0.357253613	0.893175211	-0.769930023	0.684675053	0.269595542	0.705982695	STANDING
0.116948369	-0.345136512	0.930098548	-0.6464027	0.467892769	0.262751838	0.603885473	STANDING
0.248042906	-0.303948701	0.737485374	-0.859802493	0.647402395	0.296905238	0.514994603	STANDING
-0.709711844	0.032196412	1.157554755	-0.647494103	0.548833741	0.301928493	0.474742466	STANDING
-0.137785364	-0.085822981	0.85368708	-0.63551871	0.987405426	0.477568422	0.590664492	STANDING
-0.337459958	-0.243026448	1.095896604	-0.707716697	0.739239517	0.04972716	0.494796784	STANDING
-0.160292217	-0.477462394	0.904414643	-0.587028222	0.662549199	0.143131645	0.361362174	STANDING
0.277181657	-0.236343455	0.672690154	-0.756133708	0.562502489	0.395388677	0.518869726	STANDING
-0.005303415	-0.333661577	0.813756698	-0.928377447	0.7866421	0.402249818	0.458687589	STANDING

Gambar 5.23: Dua puluh baris terakhir dataset *mobile\_sensor* setelah diacak

Tabel 5.2: Contoh perbedaan jarak Euclidean antara *dataset mobile\_sensor* asli dan yang telah diacak dengan *Random Projection Perturbation*

<b>Baris</b>	<b>Baris</b>	<b>Dataset Asli</b>	<b>Dataset Diacak</b>
3	3833	5.640583825006142	5.635041851653843
3	3834	5.493672289123771	5.415793154989641
3	3835	5.639131752043805	5.6762986347602675
3	3836	5.474539329815776	5.369571735368663
3	3837	5.59495589624516	5.518812707950394

- 1 baris yang merepresentasikan sebuah objek data pada *dataset* dengan objek data lainnya. Salah satu  
 2 contoh saat Pertidaksamaan 5.4 tidak terpenuhi adalah apabila nilai variabel *epsilon* ditentukan  
 3 sebesar 0.4. Salah satu jarak Euclidean yang melanggar Pertidaksamaan 5.4 adalah baris 473  
 4 dengan baris 1306 yang memiliki jarak Euclidean sebesar  $8.167734238223167$  pada *dataset* asli dan  
 5  $9.723888530285468$  pada *dataset* yang telah diacak, perhitungan manual pengujinya dapat dilihat  
 6 pada Pertidaksamaan 5.5. Pengujian ini membuktikan bahwa perangkat lunak berhasil menerapkan  
 7 teknik *Random Projection Perturbation* dengan akurat sesuai batas distorsi yang ditentukan oleh  
 8 pengguna.

$$(1 - \text{eps})\|u - v\|^2 < \|p(u) - p(v)\|^2 \not< (1 + \text{eps})\|u - v\|^2 \quad (5.5)$$

$$(1 - 0.4)8.167734238223167^2 < 9.723888530285468^2 \not< (1 + 0.4)8.167734238223167^2$$

$$40.02712955174579 < 94.55400814941728 \not< 93.39663562074017$$

### 9 5.3 Pengujian Eksperimental

- 10 Pengujian eksperimental bertujuan untuk menguji kualitas hasil dari perangkat lunak *Randomization*  
 11 pada kedua teknik *Randomization* dan membandingkan kualitas hasil pengacakan dari kedua teknik  
 12 tersebut pada penambangan data. Pengujian dibagi menjadi beberapa bagian seperti berikut.

- 13 1. Properti Data
- 14 (a) *Random Rotation Perturbation* dengan *dataset diabetes*  
 15 (b) *Random Projection Perturbation* dengan *dataset mobile\_sensor*
- 16 2. Penambangan Data Klasifikasi
- 17 (a) *Random Rotation Perturbation* dengan *dataset diabetes*  
 18 (b) *Random Projection Perturbation* dengan *dataset mobile\_sensor*
- 19 3. Penambangan Data *Clustering*
- 20 (a) *Random Rotation Perturbation* dengan *dataset mall\_customers*  
 21 (b) *Random Projection Perturbation* dengan *dataset mobile\_sensor*
- 22 4. Kesimpulan Akhir

1 Pengujian akan dilakukan dengan menggunakan program pengujian yang menerapkan teknik  
2 penambangan data yang telah dibuat pada bahasa pemrograman Python dan didukung oleh perangkat  
3 lunak Spyder untuk menampilkan visualisasi hasil penambangan data.

4 **5.3.1 Properti Data**

5 Pengujian properti data bertujuan untuk membandingkan beberapa properti data setiap kolom  
6 pada *dataset* yaitu sebagai berikut.

- 7 1. Rata-rata
- 8 2. Standar deviasi
- 9 3. Nilai terkecil
- 10 4. Nilai terbesar
- 11 5. Kuartil bawah
- 12 6. Kuartil tengah
- 13 7. Kuartil atas

14 Pengujian ini akan membandingkan properti data pada *dataset* asli dan *dataset* yang telah diacak.  
15 Pengujian akan dibagi menjadi 2 bagian yaitu *dataset* yang diacak dengan teknik *Random Rotation*  
16 *Perturbation* dan *dataset* yang diacak dengan teknik *Random Projection Perturbation*. Berikut  
17 pengujian yang telah dilakukan.

18 ***Random Rotation Perturbation***

19 Pengujian teknik *Random Rotation Perturbation* untuk membandingkan properti data akan dilaku-  
20 kan dengan *dataset diabetes*<sup>3</sup> yang dapat dilihat 20 baris pertamanya pada Gambar 5.24. *Dataset*  
21 ini diacak dengan teknik *Random Rotation Perturbation* dan hasil pengacakan dapat dilihat pada  
22 Gambar 5.25.

23 Properti-properti 4 kolom terakhir pada *dataset diabetes* asli dapat dilihat pada Tabel 5.3.  
24 Sementara untuk *dataset diabetes* yang telah diacak dapat dilihat pada Tabel 5.4. Jika dilihat  
25 pada kedua tabel tersebut, seluruh properti pada *dataset* yang telah diacak mempunyai nilai yang  
26 berbeda kecuali jumlah baris (*count*) dan kolom label (*Outcome*). Hal ini menunjukkan selain nilai  
27 pada setiap data, teknik *Random Rotation Perturbation* juga mengacak bermacam properti data  
28 seperti rata-rata, standar deviasi, nilai terkecil, nilai terbesar, kuartil bawah, kuartil tengah dan  
29 kuartil atas setiap kolom pada *dataset*.

30 ***Random Projection Perturbation***

31 Pengujian teknik *Random Projection Perturbation* untuk membandingkan properti data akan  
32 dilakukan dengan *dataset mobile\_sensor* yang dapat dilihat 20 baris pertamanya pada Gambar 5.22.

---

<sup>3</sup><https://www.kaggle.com/uciml/pima-indians-diabetes-database>

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6		0.627	50
1	85	66	29	0	26.6		0.351	31
8	183	64	0	0	23.3		0.672	32
1	89	66	23	94	28.1		0.167	21
0	137	40	35	168	43.1		2.288	33
5	116	74	0	0	25.6		0.201	30
3	78	50	32	88	31		0.248	26
10	115	0	0	0	35.3		0.134	29
2	197	70	45	543	30.5		0.158	53
8	125	96	0	0	0		0.232	54
4	110	92	0	0	37.6		0.191	30
10	168	74	0	0	38		0.537	34
10	139	80	0	0	27.1		1.441	57
1	189	60	23	846	30.1		0.398	59
5	166	72	19	175	25.8		0.587	51
7	100	0	0	0	30		0.484	32
0	118	84	47	230	45.8		0.551	31
7	107	74	0	0	29.6		0.254	31
1	103	30	38	83	43.3		0.183	33
1	115	70	30	96	34.6		0.529	32

Gambar 5.24: Dua puluh baris pertama dataset diabetes asli

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
-91.37651756	-80.18724466	23.05889315	-18.9693363	110.5088496	-133.4205069		86.0698456	-48.84586837
-75.53749369	-36.05193919	13.61201036	-17.05774302	96.33252668	-90.82560108		76.21980243	-40.35699091
-92.42942058	-94.56928839	1.851857108	3.981465625	114.1469917	-163.8819462		58.28394087	-55.95014122
-126.993899	-19.97252557	41.95044912	12.71359672	146.366212	-80.48590451		42.87623651	-14.4243376
-175.912903	-54.87009029	87.61024212	41.12103859	178.1742884	-93.93255265		23.24115845	15.4702748
-77.49366851	-53.89318148	-6.042582122	-6.564720711	113.553306	-114.0705412		64.92332441	-50.40619642
-120.6678891	-23.70836905	48.43553354	7.584019346	132.5262512	-69.33207772		43.64006616	-4.804724932
-69.83804655	-86.30230284	6.576880008	10.61500227	73.50342656	-98.42959311		36.60831617	-13.21252366
-380.5808875	-22.67446556	243.749478	117.0593059	410.6210675	-85.54982523		-87.46332942	95.24128787
-66.68175924	-53.19432898	4.407546648	-29.43172457	136.7787079	-128.4024803		61.29313373	-67.66505132
-82.77052893	-48.11261586	-14.37485848	-7.899390021	122.7354016	-107.9874489		80.23207042	-54.51218446
-95.0159058	-89.40851965	-6.621076283	0.876107771	119.2961308	-148.640664		71.21720253	-53.07414301
-75.73544771	-79.23855094	1.003430106	-18.05831711	130.9163815	-126.3571694		70.26494314	-54.22574441
-521.8891047	17.70297409	349.3454371	195.9590596	583.3955701	-27.00658802		-199.2092795	183.302024
-177.2238609	-63.14743279	84.04348195	29.40445346	215.1435288	-120.5701915		17.77020272	-6.772259965
-62.0408439	-77.4456268	9.358955118	7.376454038	73.79847691	-87.95326333		34.66589639	-13.57824067
-215.9630713	-15.85940033	104.9031479	39.45825766	228.6949467	-77.25266936		29.06237766	18.15464029
-76.61265022	-51.2057761	-8.491558255	-8.751231878	113.380821	-105.9266292		67.03349362	-46.88195403
-124.2450241	-53.51904364	56.4471758	15.2220361	121.7116444	-79.50822308		47.21844068	0.938260398
-135.5651967	-39.46936443	49.87545502	11.63635561	153.8601703	-96.05165177		51.95779491	-17.56987501

Gambar 5.25: Dua puluh baris pertama dataset diabetes setelah diacak

Tabel 5.3: Properti-properti pada dataset *diabetes* asli

	<b>BMI</b>	<b>DiabetesPedigreeFunction</b>	<b>Age</b>	<b>Outcome</b>
<b>count</b>	768.000000	768.000000	768.000000	768.000000
<b>mean</b>	31.992578	0.471876	33.240885	0.348958
<b>std</b>	7.884160	0.331329	11.760232	0.476951
<b>min</b>	0.000000	0.078000	21.000000	0.000000
<b>25%</b>	27.300000	0.243750	24.000000	0.000000
<b>50%</b>	32.000000	0.372500	29.000000	0.000000
<b>75%</b>	36.600000	0.626250	41.000000	1.000000
<b>max</b>	67.100000	2.420000	81.000000	1.000000

Tabel 5.4: Properti-properti pada dataset *diabetes* yang telah diacak

	<b>BMI</b>	<b>DiabetesPedigreeFunction</b>	<b>Age</b>	<b>Outcome</b>
<b>count</b>	768.000000	768.000000	768.000000	768.000000
<b>mean</b>	-103.101329	50.477502	-23.129061	0.348958
<b>std</b>	24.146554	35.533157	32.695887	0.476951
<b>min</b>	-176.332307	-199.209279	-74.182020	0.000000
<b>25%</b>	-116.872237	35.049918	-46.748389	0.000000
<b>50%</b>	-101.458852	58.320138	-28.463990	0.000000
<b>75%</b>	-86.757613	73.320823	-9.757361	1.000000
<b>max</b>	-22.156712	116.151884	183.302024	1.000000

1 *Dataset* ini diacak dengan teknik *Random Projection Perturbation* dan hasil pengacakan dapat  
 2 dilihat pada Gambar 5.23.

3 Properti-properti 3 buah kolom pada dataset *mobile\_sensor* asli dapat dilihat pada Tabel 5.5.  
 4 Sementara untuk dataset *mobile\_sensor* yang telah diacak dapat dilihat pada Tabel 5.6. Jika  
 5 dilihat pada kedua tabel tersebut, seluruh properti pada dataset yang telah diacak mempunyai  
 6 nilai yang berbeda kecuali jumlah baris (*count*). Hal ini menunjukkan selain nilai pada setiap data,  
 7 teknik *Random Projection Perturbation* juga mengacak bermacam properti dataset seperti rata-rata,  
 8 standar deviasi, nilai terkecil dan terbesar pada sebuah kolom, kuartil bawah, kuartil tengah dan  
 9 kuartil atas setiap kolom pada dataset.

## 10 Kesimpulan

11 Berdasarkan pengujian properti data akan dibuat kesimpulan sekaligus membandingkan antara  
 12 *dataset* asli, *dataset* yang telah diacak menggunakan teknik *Random Rotation Perturbation*, dan

Tabel 5.5: Properti-properti pada dataset *mobile\_sensor* asli

	<b>tBodyAcc-mean()-X</b>	<b>tBodyAcc-mean()-Y</b>	<b>angle(Y,gravityMean)</b>
<b>count</b>	10299.000000	10299.000000	10299.000000
<b>mean</b>	0.274347	-0.017743	0.063255
<b>std</b>	0.067628	0.037128	0.305468
<b>min</b>	-1.000000	-1.000000	-1.000000
<b>25%</b>	0.262625	-0.024902	0.002151
<b>50%</b>	0.277174	-0.017162	0.182028
<b>75%</b>	0.288354	-0.010625	0.250790
<b>max</b>	1.000000	1.000000	1.000000

Tabel 5.6: Properti-properti pada *dataset mobile\_sensor* yang telah diacak

	<b>0</b>	<b>1</b>	<b>425</b>
<b>count</b>	10299.000000	10299.000000	10299.000000
<b>mean</b>	0.136942	-0.289681	0.173610
<b>std</b>	0.528322	0.266849	0.214992
<b>min</b>	-1.433025	-1.206428	-1.437424
<b>25%</b>	-0.340174	-0.482918	0.033424
<b>50%</b>	0.308314	-0.270461	0.179048
<b>75%</b>	0.582374	-0.097266	0.325611
<b>max</b>	1.283867	0.726498	0.978304

Tabel 5.7: Perbandingan Properti Data

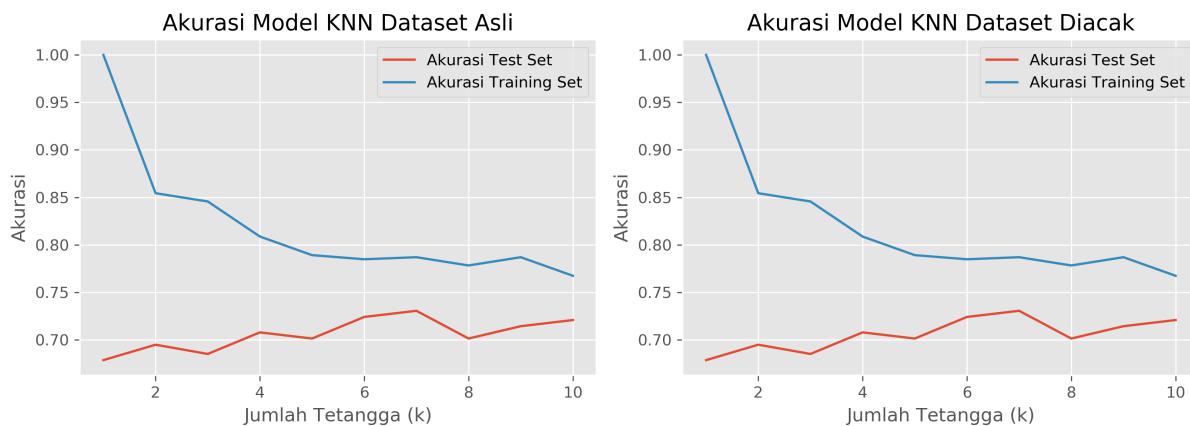
	<b>Rotation</b>	<b>Projection</b>
<b>Rata-rata</b>	Berbeda	Berbeda
<b>Standar Deviasi</b>	Berbeda	Berbeda
<b>Nilai Terkecil</b>	Berbeda	Berbeda
<b>Nilai Terbesar</b>	Berbeda	Berbeda
<b>Kuartil Bawah</b>	Berbeda	Berbeda
<b>Kuartil Tengah</b>	Berbeda	Berbeda
<b>Kuartil Atas</b>	Berbeda	Berbeda

- <sup>1</sup> *dataset* yang telah diacak menggunakan teknik *Random Projection Perturbation*. Pada Tabel 5.7  
<sup>2</sup> dapat dilihat hasil akhir pengujian antara properti data pada *dataset* asli dan *dataset* yang telah  
<sup>3</sup> diacak dengan *Random Rotation Perturbation* dan *Random Projection Perturbation*. Dapat dilihat  
<sup>4</sup> pada tabel tersebut metode *Randomization* mengacak data dengan baik dan tidak menjaga properti-  
<sup>5</sup> properti lain pada data selain jarak Euclidean sehingga properti-properti data tersebut tidak dapat  
<sup>6</sup> digunakan setelah data diacak dengan metode *Randomization*.

### <sup>7</sup> 5.3.2 Penambangan Data Klasifikasi

- <sup>8</sup> Pengujian dengan penambangan data klasifikasi akan berpusat pada pembuatan model dengan  
<sup>9</sup> *dataset* asli dan *dataset* yang telah diacak dan membandingkan kedua model tersebut. Teknik  
<sup>10</sup> penambangan data klasifikasi yang digunakan adalah *k-nearest neighbors*. Pengujian dengan  
<sup>11</sup> penambangan data klasifikasi akan dibagi menjadi 2 bagian yaitu *dataset* yang diacak dengan  
<sup>12</sup> teknik *Random Rotation Perturbation* dan *dataset* yang diacak dengan teknik *Random Projection*  
<sup>13</sup> *Perturbation*. Pengujian akan membandingkan beberapa informasi pada *dataset* asli dan *dataset*  
<sup>14</sup> yang telah diacak. Beberapa informasi tersebut adalah sebagai berikut.

- <sup>15</sup> 1. Akurasi model *k-nearest neighbors*  
<sup>16</sup> 2. Nilai variabel *k* yang modelnya memiliki akurasi tertinggi  
<sup>17</sup> 3. Waktu eksekusi pelatihan model dan prediksi dengan model yang telah dibuat  
<sup>18</sup> Tujuan dari pengujian ini adalah mencari informasi yang masih terjaga (mempunyai hasil yang  
<sup>19</sup> sama atau mirip) antara model yang dilatih dengan *dataset* asli dan *dataset* yang telah diacak.



Gambar 5.26: Grafik akurasi model klasifikasi pada *training set* dan *test set* *dataset diabetes*

### ***1 Random Rotation Perturbation***

Pengujian teknik *Random Rotation Perturbation* untuk penambangan data klasifikasi dengan algoritma *k-nearest neighbors* akan dilakukan dengan *dataset diabetes* yang dapat dilihat 20 baris pertamanya pada Gambar 5.24. *Dataset* ini diacak dengan teknik *Random Rotation Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 5.25. Dengan kedua *dataset* tersebut, dilakukan penambangan data terhadap kedua *dataset* tersebut dan dihasilkan berbagai macam informasi yang dapat dibandingkan. Berikut adalah hasil pengujian dan penjelasannya.

Teknik penambangan data *k-nearest neighbors* diterapkan menggunakan 8 buah fitur yang ada untuk menguji apakah *dataset* asli dan *dataset* yang telah diacak mempunyai akurasi model klasifikasi yang sama persis. Pengujian tersebut didasarkan pada sifat dari teknik *Random Rotation Perturbation* yang menjamin jarak Euclidean setiap titik tidak berubah sama sekali. *Dataset* akan dibagi dua menjadi *train set* dan *test set* yang masing-masing berguna untuk melatih model dan menghitung akurasi model. Akurasi akan dihitung pada setiap jumlah tetangga (nilai variabel *k*) dari 1 sampai 10. Implementasi kode teknik penambangan data ini diterapkan dengan bahasa pemograman Python dan dibantu oleh *library Scikit-learn*. Pada Gambar 5.26 dapat dilihat grafik akurasi model *k-nearest neighbors* dalam memprediksi label *training set* dan *test set* pada *dataset* asli dan *dataset* yang telah diacak. Apabila dibandingkan, kedua grafik tersebut terlihat memiliki nilai akurasi yang sama persis untuk setiap nilai variabel *k*.

Contoh sebagian kecil keluaran perangkat lunak pengujian yaitu akurasi model untuk setiap nilai variabel *k* dari 6 sampai 10 pada *dataset* asli dan *dataset* yang telah diacak dapat dilihat masing-masing pada Listing 5.1 dan Listing 5.2. Akurasi *test set* tertinggi pada model *k-nearest neighbors* dengan *dataset* asli adalah sebesar 0.7305194805194806 dengan nilai variabel *k* sebesar 7. Nilai yang sama juga muncul pada *dataset* yang telah diacak. Hal ini dapat menjadi bukti bahwa teknik *Random Rotation Perturbation* menjaga jarak Euclidean dengan sempurna, tidak ada perubahan nilai pada jarak Euclidean antara seluruh titik. Oleh karena itu model *k-nearest neighbors* yang terbuat memiliki hasil yang sama persis.

Listing 5.1: *Dataset diabetes* Asli

```
Akurasi setiap K pada training
set dataset asli :
6: 0.7847826086956522
7: 0.7869565217391304
8: 0.7782608695652173
9: 0.7869565217391304
10: 0.7673913043478261
```

1

```
Akurasi setiap K pada test
set dataset asli :
6: 0.724025974025974
7: 0.7305194805194806
8: 0.7012987012987013
9: 0.7142857142857143
10: 0.7207792207792207
```

Listing 5.2: *Dataset diabetes* Diacak

```
Akurasi setiap K pada training
set dataset diacak :
6: 0.7847826086956522
7: 0.7869565217391304
8: 0.7782608695652173
9: 0.7869565217391304
10: 0.7673913043478261
```

```
Akurasi setiap K pada test
set dataset diacak :
6: 0.724025974025974
7: 0.7305194805194806
8: 0.7012987012987013
9: 0.7142857142857143
10: 0.7207792207792207
```

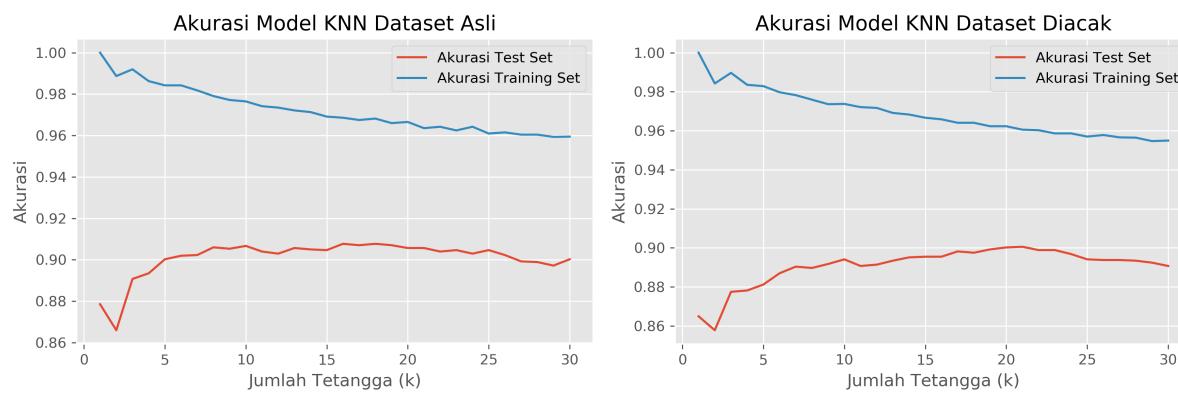
2 Waktu eksekusi yang dibutuhkan untuk melatih model klasifikasi dengan nilai variabel  $k$   
 3 sebesar 7 memakai *dataset* asli adalah sebesar 0.0009961128234863281 detik dan waktu yang  
 4 dibutuhkan untuk memprediksi *test set* adalah sebesar 0.010970354080200195 detik. Sementara  
 5 untuk *dataset* yang telah diacak membutuhkan waktu eksekusi untuk melatih model klasifikasi  
 6 sebesar 0.000997781753540039 detik dan waktu yang dibutuhkan untuk melakukan prediksi adalah  
 7 sebesar 0.011968135833740234 detik. Hal ini menunjukkan tidak ada pengaruh yang signifikan  
 8 terhadap durasi waktu eksekusi untuk melatih model dan memprediksi dengan *dataset* asli dan  
 9 *dataset* yang telah diacak.

#### 10 *Random Projection Perturbation*

11 Pengujian teknik *Random Projection Perturbation* untuk penambangan data klasifikasi dengan  
 12 algoritma *k-nearest neighbors* akan dilakukan dengan *dataset mobile\_sensor* yang dapat dilihat  
 13 20 baris pertamanya pada Gambar 5.22. *Dataset* ini diacak dengan teknik *Random Projection*  
 14 *Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 5.23. Dengan kedua *dataset* tersebut,  
 15 dilakukan penambangan data terhadap kedua *dataset* tersebut dan dihasilkan berbagai macam  
 16 informasi yang dapat dibandingkan. Berikut adalah hasil pengujian dan penjelasannya.

17 Teknik penambangan data *k-nearest neighbors* diterapkan menggunakan 561 fitur yang ada untuk  
 18 menguji apakah *dataset* mempunyai akurasi model klasifikasi yang hampir sama. Pengujian tersebut  
 19 didasarkan pada sifat teknik *Random Projection Perturbation* yang menjamin jarak Euclidean  
 20 setiap titik terjaga dengan besar distorsi yang ditentukan oleh pengguna. *Dataset* akan dibagi dua  
 21 menjadi *train set* dan *test set* yang masing-masing berguna untuk melatih model dan menghitung  
 22 akurasi model. Akurasi akan dihitung dengan jumlah tetangga (nilai variabel  $k$ ) dari 1 sampai 30.  
 23 Implementasi kode teknik penambangan data ini diterapkan dengan bahasa pemrograman Python  
 24 dan dibantu oleh *library* Scikit-learn.

25 Pada Gambar 5.27 dapat dilihat grafik akurasi model *k-nearest neighbors* dalam memprediksi



Gambar 5.27: Grafik akurasi model klasifikasi pada *training set* dan *test set* dataset *mobile\_sensor*

- 1 label *training set* dan *test set* pada *dataset* asli dan *dataset* yang telah diacak. Apabila dibandingkan,
- 2 kedua grafik tersebut memiliki nilai akurasi yang mirip. Walaupun begitu, teknik *Random Projection*
- 3 *Perturbation* tidak menjamin jarak Euclidean terjaga dengan sempurna maka ada sedikit perbedaan
- 4 yang lumayan terlihat khususnya pada akurasi *test set* dan jumlah tetangga (nilai variabel *k*) pada
- 5 model yang memiliki akurasi tertinggi. Tetapi perbedaan nilai akurasi tertingginya masih mirip
- 6 dengan *dataset* asli.
- 7 Contoh sebagian kecil keluaran perangkat lunak pengujian yaitu akurasi model untuk setiap
- 8 nilai variabel *k* dari 19 sampai 23 pada *dataset* asli dan *dataset* yang telah diacak dapat dilihat
- 9 masing-masing pada Listing 5.3 dan Listing 5.4. Dapat dilihat pada kedua listing tersebut akurasi
- 10 *test set* pada kedua *dataset* berbeda. Akurasi *test set* tertinggi pada *dataset* asli adalah sebesar
- 11 0.9077027485578555 dengan nilai variabel *k* sebesar 16. Sementara pada *dataset* yang telah diacak,
- 12 akurasi *test set* tertingginya adalah sebesar 0.9005768578215134 dengan nilai variabel *k* sebesar
- 13 21. Jika dihitung perbedaan akurasinya adalah sebesar 0.0071258907363421 yang mana relatif
- 14 kecil tetapi ada perbedaan nilai variabel *k* pada model yang memiliki akurasi tertinggi. Dengan
- 15 perbedaan akurasi yang relatif kecil maka dapat disimpulkan bahwa teknik *Random Projection*
- 16 *Perturbation* menjaga jarak Euclidean dengan baik dan distorsinya terkontrol sesuai yang pengguna
- 17 inginkan. Oleh karena itu, model *k-nearest neighbors* yang terbuat memiliki hasil yang sangat mirip.

Tabel 5.8: Perbandingan Model *k-nearest neighbors* antara model yang dilatih dengan *dataset* asli dan yang telah diacak

	<i>Rotation</i>	<i>Projection</i>
<b>Akurasi Model</b>	Sama	Sangat Mirip
<b><i>k</i> terbaik</b>	Sama	Dapat Berbeda
<b>Waktu Eksekusi</b>	Sama	Lebih Cepat

Listing 5.3: *Dataset mobile\_sensor* Asli

```
Akurasi setiap K pada training
set dataset asli:
19: 0.9659956474428727
20: 0.9665397170837867
21: 0.9635473340587595
22: 0.9642274211099021
23: 0.9624591947769314
```

1

```
Akurasi setiap K pada test
set dataset asli:
19: 0.9070240922972514
20: 0.9056667797760435
21: 0.9056667797760435
22: 0.9039701391245334
23: 0.9046487953851374
```

2

Waktu eksekusi yang dibutuhkan untuk melatih model klasifikasi dengan nilai variabel *k* sebesar 16 memakai *dataset* asli adalah sebesar 0.2872335910797119 detik dan waktu yang dibutuhkan untuk memprediksi *test set* adalah sebesar 17.754546642303467 detik. Sementara untuk *dataset* yang telah diacak membutuhkan waktu eksekusi untuk melatih model klasifikasi dengan nilai variabel *k* sebesar 21 adalah sebesar 0.5630350112915039 detik dan waktu yang dibutuhkan untuk melakukan prediksi adalah sebesar 12.86760687828064 detik. Pada *dataset* yang telah diacak waktu prediksi lebih cepat dikarenakan fitur-fitur yang ada lebih sedikit daripada *dataset* asli. Hal ini menunjukkan ada pengaruh yang signifikan terhadap durasi waktu eksekusi untuk melatih model dan memprediksi dengan *dataset* asli dan *dataset* yang telah diacak.

## 11 Kesimpulan

Berdasarkan pengujian dengan teknik penambangan data klasifikasi menggunakan teknik *k-nearest neighbors* pada *dataset* asli, *dataset* yang telah diacak menggunakan teknik *Random Rotation Perturbation*, dan *dataset* yang telah diacak menggunakan teknik *Random Projection Perturbation* dibuat kesimpulan sekaligus membandingkan antara kedua teknik *Randomization*. Pada Tabel 5.8 dapat dilihat hasil akhir pengujian antara model yang dilatih dengan *dataset* asli dan *dataset* yang telah diacak dengan *Random Rotation Perturbation* dan *Random Projection Perturbation*. Akurasi model *k-nearest neighbors* terjaga dengan baik pada kedua teknik *Randomization* yang dikarenakan oleh jarak Euclidean tetap terjaga setelah *dataset* diacak. Tetapi untuk teknik

Listing 5.4: *Dataset mobile\_sensor* Diacak

```
Akurasi setiap K pada training
set dataset diacak:
19: 0.9623231773667029
20: 0.9623231773667029
21: 0.9605549510337323
22: 0.9602829162132753
23: 0.9586507072905331
```

```
Akurasi setiap K pada test
set dataset diacak:
19: 0.8992195453003053
20: 0.9002375296912114
21: 0.9005768578215134
22: 0.8988802171700034
23: 0.8988802171700034
```

1 *Random Projection Perturbation* ada persyaratan yang harus dipenuhi agar jarak Euclidean dapat  
2 terjaga dengan baik dan tidak bisa sama persis dengan aslinya seperti teknik *Random Rotation*  
3 *Perturbation*. Nilai variabel  $k$  terbaik (memiliki akurasi tertinggi) dalam pembuatan model *k-nearest*  
4 *neighbors* sama persis pada teknik *Random Rotation Perturbation*. Tetapi pada teknik *Random*  
5 *Projection Perturbation* dapat berubah. Hal ini menyebabkan perlunya perhitungan kembali untuk  
6 menentukan nilai variabel  $k$  yang baik untuk dipakai. Waktu eksekusi pembuatan model dan  
7 prediksi hanya memiliki perbedaan pada model yang menggunakan *dataset* yang telah diacak  
8 dengan teknik *Random Projection Perturbation*. Waktu eksekusinya lebih cepat dikarenakan oleh  
9 data pada *dataset* lebih sedikit karena dimensinya direduksi.

10 **5.3.3 Penambangan Data *Clustering***

11 Pengujian dengan penambangan data *clustering* akan berpusat pada pembuatan model *clustering*  
12 dengan *dataset* asli dan *dataset* yang telah diacak dan membandingkan kedua model tersebut. Teknik  
13 penambangan data *clustering* yang digunakan adalah *k-means*. Pengujian akan membandingkan  
14 beberapa informasi pada *dataset* asli dan *dataset* yang telah diacak. Beberapa informasi tersebut  
15 adalah sebagai berikut.

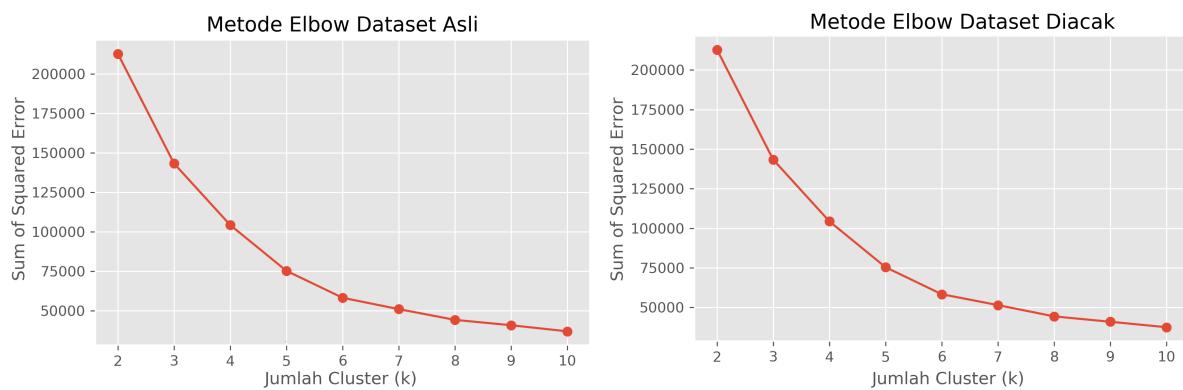
- 16 1. *Sum of Squared Error*  
17 2. *Silhouette Score*  
18 3. Nilai variabel  $k$  (jumlah *cluster*) yang modelnya memiliki kualitas *clustering* terbaik  
19 4. Visualisasi *cluster*  
20 5. Waktu eksekusi pelatihan model

21 Tujuan dari pengujian ini adalah mencari informasi yang masih terjaga (mempunyai hasil yang sama  
22 atau mirip) antara model yang dilatih dengan *dataset* asli dan *dataset* yang telah diacak. Selain itu  
23 metode *Adjusted Rand Index* juga digunakan untuk menghitung kemiripan antara kedua model.

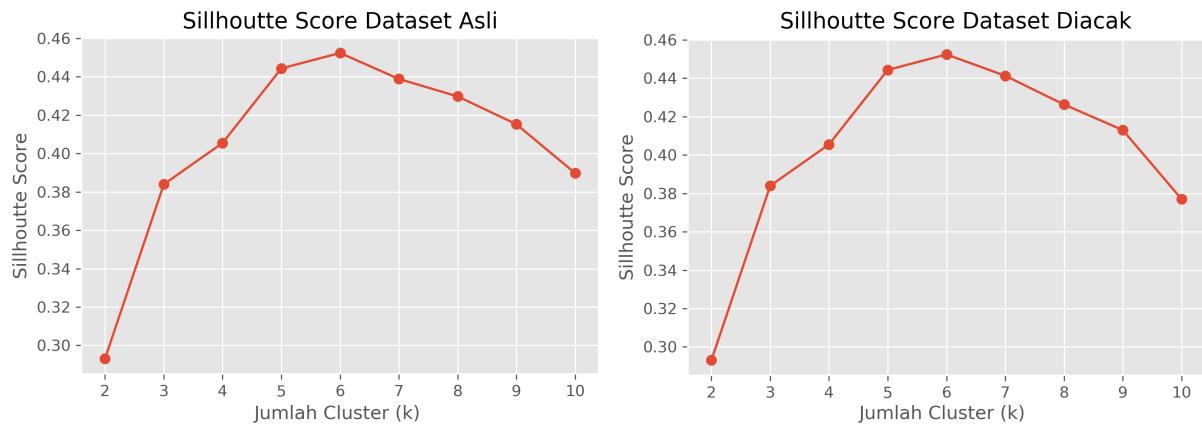
24 ***Random Rotation Perturbation***

25 Pengujian teknik *Random Rotation Perturbation* untuk penambangan data *clustering* dengan  
26 menggunakan algoritma *k-means* akan dilakukan dengan *dataset* *mall\_customers* yang dapat dilihat  
27 20 baris pertamanya pada Gambar 5.19. *Dataset* ini diacak dengan teknik *Random Rotation*  
28 *Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 5.20. Dengan kedua *dataset* tersebut,  
29 dilakukan penambangan data terhadap kedua *dataset* tersebut dan dihasilkan berbagai macam  
30 informasi yang dapat dibandingkan. Berikut adalah hasil pengujian dan penjelasannya.

31 Dalam menentukan jumlah *cluster* atau nilai variabel  $k$  yang terbaik untuk membuat model  
32 *clustering* dengan algoritma *k-means* perlu ada metode untuk menentukan nilai tersebut. Metode  
33 Elbow menjadi salah satu metode yang digunakan untuk menentukan nilai variabel  $k$  yang terbaik  
34 untuk dipakai. Pada Gambar 5.28 terdapat grafik *Sum of Squared Error* untuk menggunakan  
35 metode Elbow pada *dataset* asli dan *dataset* yang telah diacak. Terlihat nilai variabel  $k$  sebesar  
36 5, 6, dan 7 adalah kandidat terbaik untuk menjadi nilai variabel  $k$  yang dipakai pada *dataset* asli  
37 maupun *dataset* yang telah diacak. Pada kedua *dataset* tersebut grafik *Sum of Squared Error*-nya



Gambar 5.28: Grafik *Sum of Squared Error* model *clustering* pada dataset *mall\_customers*



Gambar 5.29: Grafik *Silhouette Score* model *clustering* pada dataset *mall\_customers*

- 1 terlihat sama persis dikarenakan teknik *Random Rotation Perturbation* menjaga jarak Euclidean
- 2 dengan sempurna. Dalam menentukan nilai variabel  $k$  yang terbaik antara ketiga nilai tersebut,
- 3 *Silhouette Score* dapat digunakan untuk metode alternatif untuk menentukan nilai variabel  $k$  yang
- 4 terbaik.

Pada Gambar 5.29 terdapat grafik *Silhouette Score* dari dataset *mall\_customers* asli dan yang telah diacak. Dapat terlihat kedua grafik *Silhouette Score* terlihat sama persis dan dapat dilihat nilai-nilai pada setiap  $k$  tersebut di Listing 5.5 dan Listing 5.6 ada sedikit perbedaan yang tidak terlalu signifikan. Hal ini mungkin dikarenakan oleh nilai pada setiap data yang berbeda dan mempengaruhi sedikit algoritma pada *Silhouette Score*. Dapat terlihat *Silhouette Score* pada nilai variabel  $k$  sebesar 6 adalah nilai paling besar yaitu 0.4523443947724053 pada dataset asli dan 0.4523443947780976 pada dataset yang telah diacak. Hal ini menunjukkan teknik *Random Rotation Perturbation* tidak mempengaruhi secara signifikan nilai *Silhouette Score* pada setiap  $k$ .

Listing 5.5: Dataset *mall\_customers* Asli

Silhouette Score setiap K pada dataset asli:

- 1 2: 0.293166070535953
- 3: 0.3839349967742105
- 4: 0.40546302077733304
- 5: 0.44504314844253573
- 6: 0.4523443947724053
- 7: 0.43978902692261157
- 8: 0.42790288922594905
- 9: 0.4137641526186506
- 10: 0.3750147687842441

Listing 5.6: Dataset *mall\_customers* Diacak

Silhouette Score setiap K pada dataset diacak:

- 2: 0.29316607053507854
- 3: 0.383934996807901
- 4: 0.40546302082487856
- 5: 0.44428597567883826
- 6: 0.4523443947780976
- 7: 0.44128075766857394
- 8: 0.42815090435529995
- 9: 0.3861502477348431
- 10: 0.3897532214988177

2 Teknik penambangan data *k-means* diterapkan menggunakan 3 buah fitur yang ada untuk  
 3 menguji apakah *dataset* asli dan *dataset* yang telah diacak menghasilkan kluster dan bentuk yang  
 4 sama dengan sudut yang berbeda saat divisualisasikan. Pengujian tersebut didasarkan pada sifat  
 5 teknik *Random Rotation Perturbation* yang menjamin jarak Euclidean antara setiap titik tidak  
 6 berubah sama sekali tetapi merotasi seluruh titik yang ada pada bidang Euclidean. Implementasi  
 7 kode teknik penambangan data ini diterapkan dengan bahasa pemrograman Python dan dibantu  
 8 oleh *library* Scikit-learn.

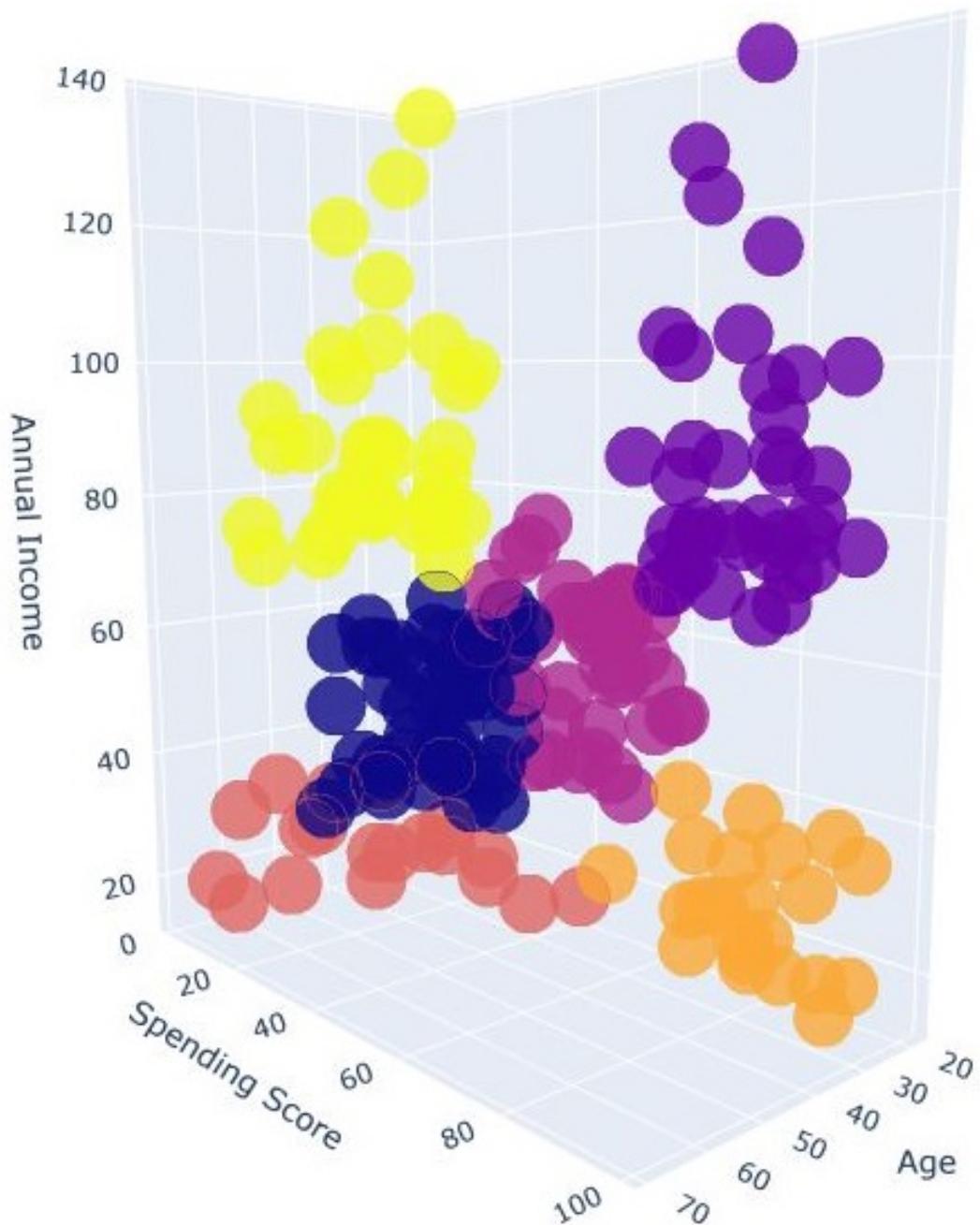
9 Visualisasi model *clustering* dengan nilai variabel *k* sebesar 6 pada *dataset mall\_customers* asli  
 10 dan yang telah diacak dapat dilihat pada Gambar 5.30 dan Gambar 5.31. Dapat dilihat pada kedua  
 11 visualisasi tersebut mempunyai jumlah *cluster* yang sama dan terlihat dari lokasi titik-titik yang  
 12 ada jika dibandingkan terlihat seperti dirotasi searah jarum jam dan bentuknya masih terlihat sama.  
 13 Apabila dihitung kemiripan *cluster* tersebut dengan metode *Adjusted Rand Index* maka kedua hasil  
 14 *clustering* tersebut mempunyai nilai 1.0 yang berarti titik-titik yang ada pada setiap *cluster* pada  
 15 kedua model persis adanya.

16 Waktu eksekusi yang dibutuhkan untuk melatih model *clustering* dengan algoritma *k-means*  
 17 adalah sebesar 0.02995157241821289 detik pada *dataset* yang asli dan sebesar 0.032910823822021484  
 18 detik pada *dataset* yang telah diacak, hal ini menunjukkan bahwa teknik *Random Rotation Pertur-*  
 19 *bation* tidak mempengaruhi secara signifikan waktu eksekusi untuk melatih model *k-means*.

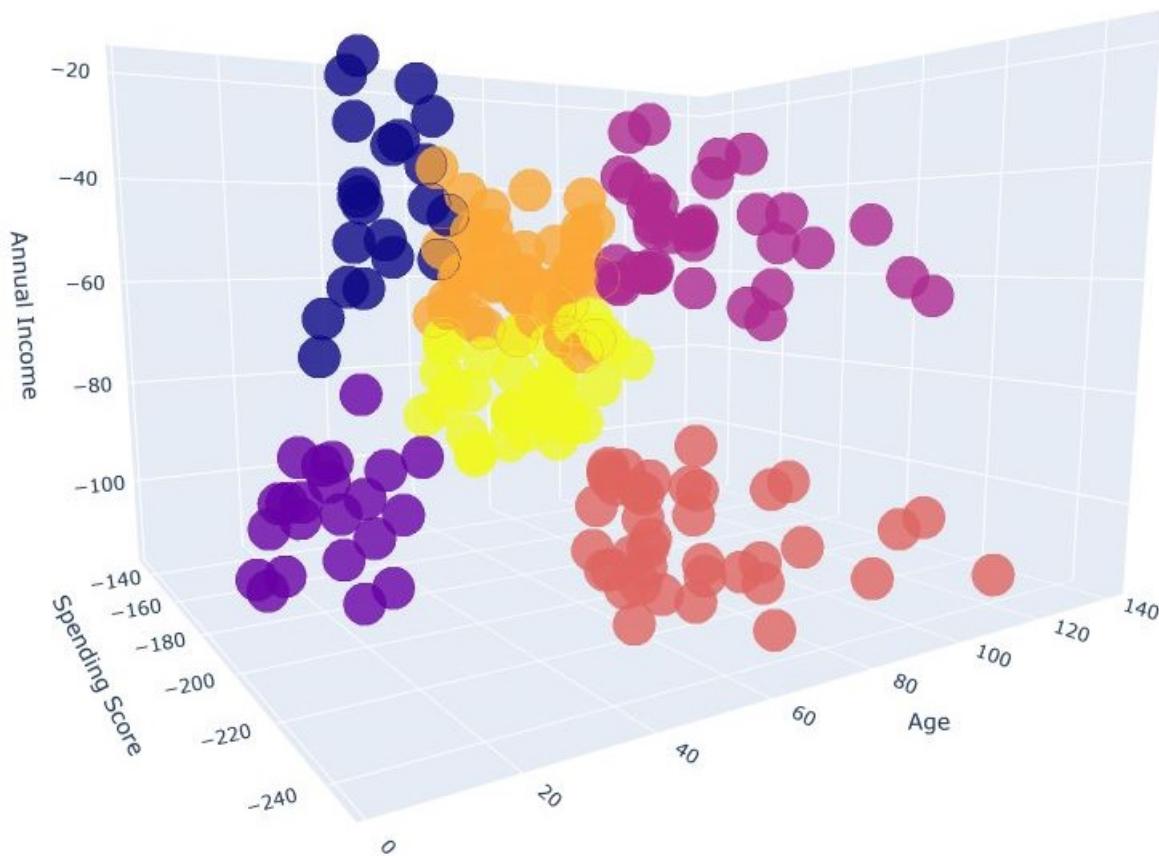
## 20 **Random Projection Perturbation**

21 Pengujian teknik *Random Projection Perturbation* untuk penambangan data *clustering* dengan  
 22 algoritma *k-means* akan dilakukan dengan *dataset mobile\_sensor* yang dapat dilihat 20 baris  
 23 pertamanya pada Gambar 5.22. *Dataset* ini diacak dengan teknik *Random Projection Perturbation*  
 24 dan hasil pengacakan dapat dilihat pada Gambar 5.23. Dengan kedua *dataset* tersebut, dilakukan  
 25 penambangan data terhadap kedua *dataset* tersebut dan dihasilkan berbagai macam informasi yang  
 26 dapat dibandingkan. Berikut adalah hasil pengujian dan penjelasannya.

27 Sebelum melakukan teknik *clustering* dengan algoritma *k-means*, *dataset* yang memiliki fitur yang  
 28 sangat banyak tersebut dimensinya harus direduksi terlebih dahulu agar model dapat divisualisasikan  
 29 dengan mudah. *Dataset* yang akan di-*cluster* akan direduksi dimensinya sampai hanya memiliki 2



Gambar 5.30: Visualisasi *cluster* pada *dataset mall\_customers* yang asli

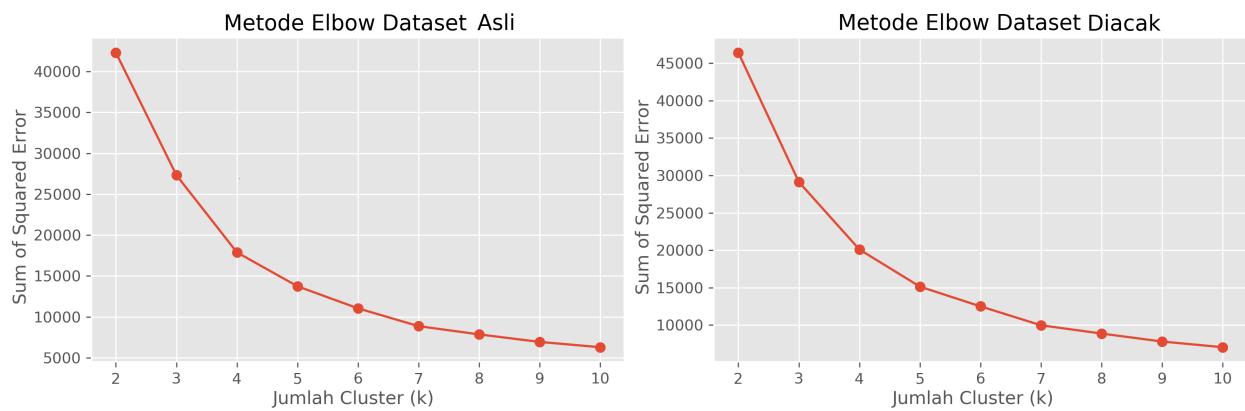


Gambar 5.31: Visualisasi *cluster* pada *dataset mall\_customers* yang telah diacak

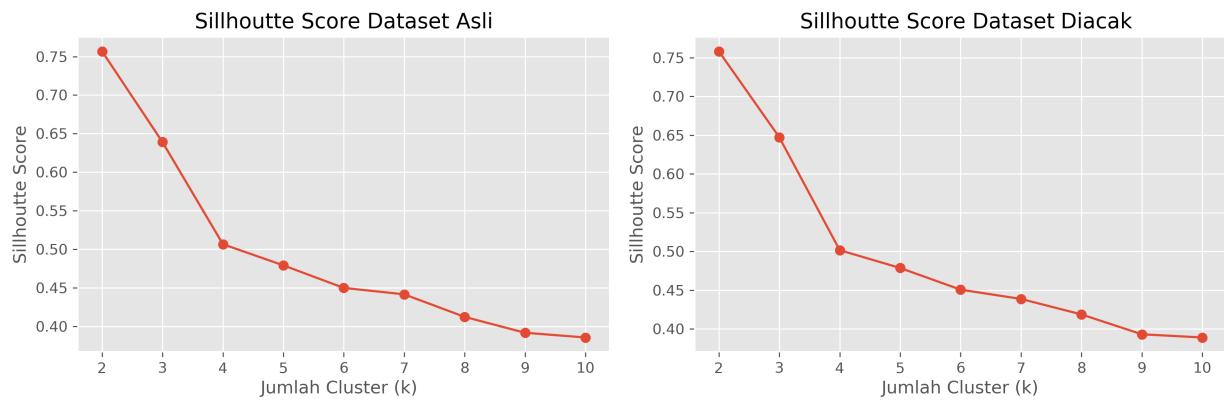
1 dimensi. Reduksi dimensi dilakukan dengan menerapkan teknik *Principal Component Analysis* yang  
 2 sudah umum digunakan pada penambangan data dan hasilnya relatif baik. Dalam menguji teknik  
 3 *Random Projection Perturbation*, *dataset* yang tidak akan diacak langsung direduksi dimensinya  
 4 dengan teknik *Principal Component Analysis*. *Dataset* yang akan diacak akan terlebih dahulu  
 5 diacak menggunakan teknik *Random Projection Perturbation* baru direduksi dimensinya menjadi 2  
 6 dimensi dengan teknik *Principal Component Analysis*.

7 Pada Gambar 5.32 terdapat grafik *Sum of Squared Error* untuk menggunakan metode Elbow  
 8 pada *dataset* asli dan *dataset* yang telah diacak. Pada grafik ini tidak terlalu terlihat nilai yang  
 9 terbaik untuk menjadi nilai variabel  $k$  yang dipakai pada *dataset* asli maupun *dataset* yang telah  
 10 diacak. Walaupun seperti itu, pada kedua *dataset* tersebut grafik *Sum of Squared Error*-nya terlihat  
 11 sangat mirip dikarenakan teknik *Random Projection Perturbation* menjaga jarak Euclidean dengan  
 12 baik dan terkontrol. Dalam menentukan nilai variabel  $k$  yang terbaik untuk dipakai membuat  
 13 model, *Silhouette Score* dapat digunakan untuk metode alternatif untuk menentukan nilai variabel  $k$   
 14 yang terbaik.

15 Pada Gambar 5.33 terdapat grafik *Silhouette Score* dari *dataset mobile\_sensor* yang asli dan yang  
 16 telah diacak. Dapat terlihat kedua grafik *Silhouette Score* terlihat sangat mirip dan dapat dilihat  
 17 nilai-nilai pada setiap  $k$  tersebut di Listing 5.7 dan Listing 5.8 ada sedikit perbedaan yang tidak  
 18 terlalu signifikan. Hal ini dikarenakan oleh jarak Euclidean pada *dataset* asli dan yang telah diacak  
 19 sedikit berbeda sehingga mempengaruhi sedikit pada hasil algoritma *Silhouette Score*. Dapat dilihat  
 20 *Silhouette Score* pada nilai variabel  $k$  sebesar 2 adalah nilai paling besar yaitu 0.7568010158989575



Gambar 5.32: Grafik *Sum of Squared Error* model *clustering* pada dataset *mobile\_sensor*



Gambar 5.33: Grafik *Silhoutte Score* model *clustering* pada dataset *mobile\_sensor*

- 1 pada *dataset* asli dan 0.7581796759180272 pada *dataset* yang telah diacak. Hal ini menunjukkan teknik *Random Projection Perturbation* tidak mempengaruhi secara signifikan nilai *Silhoutte Score* pada setiap *k*.

Listing 5.7: *Dataset mobile\_sensor* Asli

```
Silhoutte Score setiap K
pada dataset asli :
2: 0.7568010158989575
3: 0.6390101777235029
4: 0.506497255331499
5: 0.4790910470659918
6: 0.4497981866661921
7: 0.4414685328088866
8: 0.4122363491089296
9: 0.39158160384319435
10: 0.3854996654126945
```

Listing 5.8: *Dataset mobile\_sensor* Diacak

```
Silhoutte Score setiap K
pada dataset diacak :
2: 0.7581796759180272
3: 0.6472419899391577
4: 0.5015650888399312
5: 0.47862510668209096
6: 0.45064475921629954
7: 0.43865532313764366
8: 0.4186927518384631
9: 0.3930205669353965
10: 0.3889412224520463
```

- 5 Teknik penambangan data *k-means* diterapkan menggunakan dua buah fitur hasil dari teknik *Principal Component Analysis* untuk menguji apakah *dataset* asli dan *dataset* yang telah diaacak menghasilkan *cluster* yang hampir sama. Pengujian tersebut didasarkan pada sifat teknik *Random Projection Perturbation* yang menjamin jarak Euclidean terjaga dengan besar distorsi

yang ditentukan oleh pengguna. Model *k-means* akan dibuat dengan nilai variabel *k* yang terbaik dihitung menggunakan metode Elbow dan nilai variabel *k* yang memiliki *Silhouette Score* tertinggi. Implementasi kode teknik penambangan data ini diterapkan dengan bahasa pemrograman Python dan dibantu oleh *library* Scikit-learn.

Visualisasi model *clustering* dengan nilai variabel *k* sebesar 2 pada *dataset mobile\_sensor* asli dan yang telah diacak dapat dilihat pada Gambar 5.34. Dapat dilihat pada kedua visualisasi tersebut mempunyai jumlah *cluster* yang sama yaitu 2 *cluster* dan terlihat dari lokasi titik-titik yang ada jika dibandingkan ada sedikit perbedaan tetapi tidak terlalu signifikan. Hasil *clustering* menyatakan bahwa ada dua buah *cluster* yaitu kelompok orang-orang yang diam dan kelompok orang-orang yang bergerak, hasil *clustering* ini tidak secara spesifik menentukan aktivitas setiap orang karena informasi tersebut sudah hilang oleh teknik yang digunakan sebelumnya, *Principal Component Analysis*.

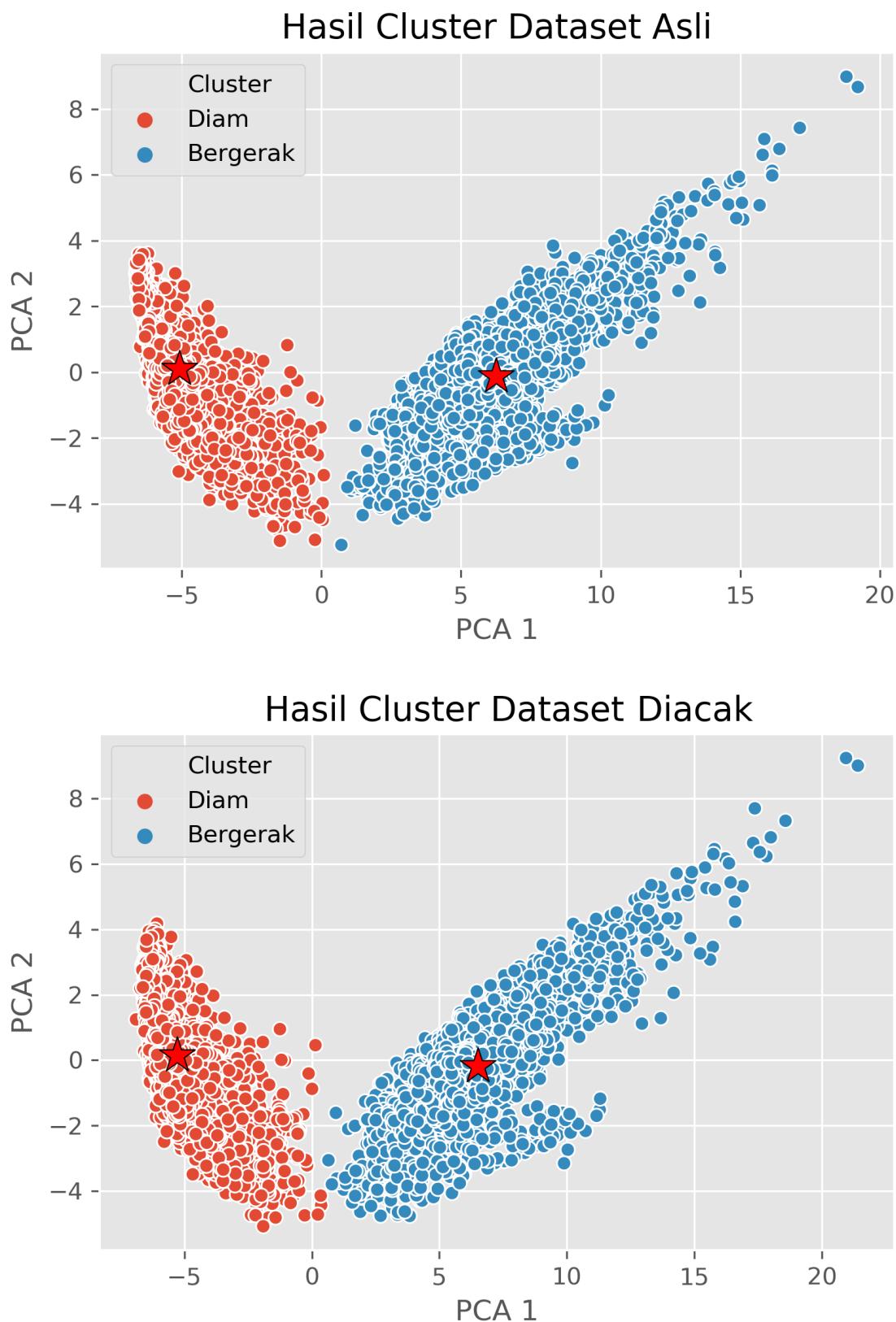
Apabila dihitung kemiripan *cluster* tersebut dengan metode *Adjusted Rand Index* maka kedua hasil *clustering* tersebut mempunyai nilai 0.9994558701020273 yang berarti titik-titik yang ada pada setiap *cluster* pada kedua model sangat mirip sekali. Hal ini dikarenakan jarak Euclidean kedua buah *dataset* tidak rusak secara signifikan dan distorsinya terkendali sesuai yang pengguna inginkan.

Waktu eksekusi yang dibutuhkan untuk melatih model *k-means* dengan nilai variabel *k* sebesar 2 adalah sebesar 0.031239032745361328 detik pada *dataset* asli dan sebesar 0.031217575073242188 detik pada *dataset* yang telah diacak. Jika dibandingkan, kedua *dataset* memiliki waktu eksekusi yang hampir sama, hal ini dikarenakan kedua *dataset* tersebut diterapkan terlebih dahulu teknik *Principal Component Analysis* sehingga kedua *dataset* memiliki ukuran yang sama. Perbedaan pada kedua *dataset* dapat terlihat pada waktu eksekusi teknik *Principal Component Analysis* pada kedua *dataset* yang mana masing-masing sebesar 0.1405951976776123 detik dan 0.1093449592590332 detik. Oleh karena itu, teknik *Random Projection Perturbation* mempengaruhi waktu eksekusi penambangan data *clustering* karena besar dimensi *dataset* berkurang sehingga waktu eksekusinya juga berkurang.

## 28 Kesimpulan

Berdasarkan pengujian dengan teknik penambangan data *clustering* menggunakan teknik *k-means* pada *dataset* asli, *dataset* yang telah diacak menggunakan teknik *Random Rotation Perturbation*, dan *dataset* yang telah diacak menggunakan teknik *Random Projection Perturbation* dibuat kesimpulan sekaligus membandingkan antara kedua teknik *Randomization*. Pada Tabel 5.9 dapat dilihat hasil akhir pengujian antara model yang dilatih dengan *dataset* asli dan *dataset* yang telah diacak dengan *Random Rotation Perturbation* dan *Random Projection Perturbation*.

Model *clustering* yang dilatih oleh *dataset* yang diacak sangat mirip dengan *dataset* asli. *Sum of Squared Error* dan *Silhouette Score* pada kedua teknik *Randomization* memiliki hasil yang sangat mirip dengan aslinya sehingga metode Elbow dan pemilihan fitur dengan *Silhouette Score* dapat dilakukan setelah *dataset* diacak. Jumlah *cluster* (nilai variabel *k*) pada model *clustering* terbaik yang dilatih dengan kedua teknik *Randomization* sama dengan model *clustering* terbaik yang dilatih dengan *dataset* asli. Visualisasi *cluster* pada kedua teknik sedikit berbeda karena teknik *Random Rotation Perturbation* merotasi seluruh data sehingga visualisasinya akan terotasi dan



Gambar 5.34: Visualisasi *cluster* pada *dataset mobile\_sensor*

Tabel 5.9: Perbandingan model *k-means* antara model yang dilatih dengan *dataset* asli dan yang telah diacak

	<i>Rotation</i>	<i>Projection</i>
<i>Sum of Squared Error</i>	Sangat Mirip	Sangat Mirip
<i>Silhouette Score</i>	Sangat Mirip	Sangat Mirip
<i>Jumlah cluster</i>	Sama	Sama
<i>Visualisasi cluster</i>	Sedikit Berbeda	Sedikit Berbeda
<i>Nilai Adjusted Rand Index</i>	1.0	Sangat Mendekati 1.0
<i>Waktu Eksekusi</i>	Sama	Lebih Cepat

1 teknik *Random Projection Perturbation* tidak menjaga jarak Euclidean secara sempurna. *Adjusted*  
 2 *Rand Index* pada kedua teknik memiliki nilai yang baik yaitu masing-masing bernilai 1.0 dan  
 3 mendekati 1.0. Hal ini menunjukkan bahwa model *clustering* antara yang dilatih dengan *dataset*  
 4 yang telah diacak dan *dataset* aslinya sama persis atau sangat mirip. Waktu eksekusi pembuatan  
 5 model hanya memiliki perbedaan pada model yang menggunakan *dataset* yang telah diacak dengan  
 6 teknik *Random Projection Perturbation*. Waktu eksekusinya lebih cepat dikarenakan oleh *dataset*  
 7 setelah diacak memiliki fitur yang lebih sedikit karena dimensinya direduksi.

#### 8 5.3.4 Kesimpulan Akhir

9 Berdasarkan hasil pengujian eksperimental yang telah dilakukan dapat disimpulkan bahwa teknik  
 10 *Random Rotation Perturbation* dan *Random Projection Perturbation* dapat digunakan untuk meng-  
 11 hilangkan privasi pada data dengan mengacak data tersebut tetapi masih dapat digunakan untuk  
 12 penambangan data klasifikasi dan *clustering* masing-masing dengan teknik *k-nearest neighbors* dan  
 13 *k-means* dengan hasil yang sama dengan *dataset* asli untuk teknik *Random Rotation Perturbation*  
 14 dan hasil yang sangat mirip dengan *dataset* asli untuk teknik *Random Projection Perturbation*.  
 15 Hal tersebut dikarenakan teknik *Random Rotation Perturbation* menjaga jarak Euclidean secara  
 16 sempurna, sementara teknik *Random Projection Perturbation* tidak menjaga jarak Euclidean secara  
 17 sempurna karena ada distorsi yang terkontrol pada jarak Euclidean antara setiap titik yang me-  
 18 representasikan sebuah objek data pada *dataset* yang diacak. Waktu eksekusi dalam pembuatan  
 19 model klasifikasi maupun *clustering* lebih cepat pada *dataset* yang diacak dengan teknik *Random*  
 20 *Projection Perturbation* karena ukuran *dataset* berkurang akibat reduksi dimensi yang dilakukan  
 21 teknik tersebut.

22 Ada persyaratan yang harus dipenuhi untuk teknik *Random Projection Perturbation* bekerja  
 23 dengan baik yaitu data yang dipakai harus cukup besar. Teknik *Random Projection Perturbation*  
 24 juga memiliki waktu eksekusi untuk melakukan pengacakan dan pembuatan model yang lebih  
 25 cepat daripada teknik *Random Rotation Perturbation*. Oleh karena itu, teknik *Random Projection*  
 26 *Perturbation* lebih cocok dipakai untuk *dataset* yang sangat besar. Sedangkan teknik *Random*  
 27 *Rotation Perturbation* masih dapat dipakai untuk *dataset* yang besar juga tetapi tidak mendapatkan  
 28 keuntungan waktu eksekusi yang lebih cepat seperti teknik *Random Projection Perturbation*. Teknik  
 29 *Random Rotation Perturbation* lebih cocok digunakan apabila penambangan data klasifikasi atau  
 30 *clustering* yang akan dilakukan diharapkan memiliki hasil yang tidak memiliki perbedaan sama  
 31 sekali dengan penambangan data klasifikasi atau *clustering* yang menggunakan *dataset* asli.

32 Kualitas hasil dari teknik *Random Projection Perturbation* akan menurun dalam 2 kondisi yaitu

- 1 pertama apabila data direduksi ke dimensi yang lebih kecil lagi (semakin besar dimensinya, semakin
- 2 kecil distorsi pada jarak Euclidean) dan kedua adalah data yang diacak bertambah banyak. Kondisi
- 3 kedua dikarenakan oleh nilai variabel  $k$  (dimensi minimal) berbanding lurus dengan banyaknya data.
- 4 Oleh karena itu semakin sering model dilatih dengan data baru, ada kemungkinan kualitas model
- 5 tersebut akan menurun apabila nilai variabel  $k$  (dimensi minimal) sudah melebihi besar dimensi
- 6 *dataset* yang telah diacak.



1

## BAB 6

2

### KESIMPULAN DAN SARAN

3 Bab ini berisi kesimpulan dari awal hingga akhir penelitian beserta saran untuk penelitian selanjut-  
4 nya.

5 **6.1 Kesimpulan**

6 Kesimpulan yang dapat ditarik dari penelitian ini adalah sebagai berikut.

- 7 • Teknik *Random Rotation Perturbation* mengacak data dengan cara merotasi setiap titik yang  
8 merepresentasikan sebuah objek data pada bidang Euclidean sehingga nilai setiap objek data  
9 berubah tetapi jarak Euclidean antara setiap titik dengan titik lainnya tidak berubah. Oleh  
10 karena itu, data yang telah diacak tersebut masih dapat digunakan untuk teknik penambangan  
11 data yang hanya memanfaatkan jarak Euclidean.
- 12 • Teknik *Random Projection Perturbation* mengacak data dengan cara memproyeksikan data  
13 yang berdimensi cukup besar ke dimensi yang lebih kecil. Teknik ini memanfaatkan *Johnson-*  
14 *Lindenstrauss Lemma* yang menyatakan bahwa titik-titik pada bidang Euclidean  $d$ -dimensi  
15 dapat diproyeksikan ke bidang Euclidean yang berdimensi lebih kecil dari  $d$  tetapi jarak  
16 Euclidean antara setiap titik tetap terjaga dengan distorsi yang terkontrol tetapi dengan  
17 syarat  $d$  harus cukup besar. Oleh karena itu, data yang telah diacak tersebut masih dapat  
18 digunakan untuk teknik penambangan data yang hanya memanfaatkan jarak Euclidean.
- 19 • Perangkat lunak diimplementasikan dengan bahasa pemrograman Python. Antarmuka per-  
20 angkat lunak dibuat dengan bantuan *framework* antarmuka bernama Kivy. Fungsi-fungsi  
21 pada perangkat lunak yang berkaitan dengan masukan, keluaran, dan implementasi algoritma  
22 dibantu oleh berbagai *library* seperti Numpy, Scipy, Pandas, dan Scikit-learn. Perangkat lunak  
23 hanya dapat menerima masukan dataset berupa dokumen *comma-separated values* yang setiap  
24 fiturnya bersifat numerik saja.
- 25 • Teknik *Random Rotation Perturbation* diimplementasikan dengan cara mengolah dataset yang  
26 ingin diacak menjadi matriks, lalu melakukan transformasi translasi pada matriks dataset  
27 tersebut dengan cara mengkalikannya dengan matriks translasi acak yang dibuat mengikuti  
28 distribusi *uniform* dengan rentang nilai  $[0, 100]$ . Kemudian transformasi rotasi dilakukan pada  
29 matriks dataset yang telah ditranslasi dengan cara mengkalikannya dengan matriks rotasi  
30 acak yang dibuat mengikuti distribusi Haar dengan bantuan *library* Scipy.

- 1     ● Teknik *Random Projection Perturbation* diimplementasikan dengan cara mengolah dataset yang  
2       ingin diacak menjadi matriks, lalu melakukan pemeriksaan persyaratan terlebih dahulu dan  
3       menentukan target dimensi yang akan menjadi dimensi dataset hasil pengacakan. Kemudian  
4       apabila persyaratan terpenuhi maka proyeksi akan dilakukan dengan cara mengkalikan matriks  
5       dataset tersebut dengan matriks proyeksi acak yang dibuat mengikuti distribusi normal.
- 6     ● Teknik *Random Rotation Perturbation* terbukti menjaga jarak Euclidean dengan sempur-  
7       na tanpa ada distorsi dan teknik *Random Projection Perturbation* terbukti menjaga jarak  
8       Euclidean dengan distorsi yang sesuai dengan keinginan pengguna.
- 9     ● Berdasarkan pengujian eksperimental yang telah dilakukan, metode *Randomization* mengacak  
10      data dan berbagai properti yaitu rata-rata, standar deviasi, nilai terkecil, nilai terbesar, kuartil  
11      bawah, kuartil tengah, dan kuartil atas setiap kolom tanpa merusak jarak Euclidean pada  
12      data tersebut.
- 13     ● *Dataset* yang diacak dengan teknik *Random Rotation Perturbation* terbukti masih dapat  
14      digunakan untuk penambangan data klasifikasi menggunakan teknik *k-nearest neighbors*  
15      dengan akurasi model klasifikasi yang sama persis. Sedangkan untuk teknik *Random Projection*  
16      *Perturbation* memiliki akurasi model klasifikasi yang sangat mirip dengan akurasi model yang  
17      dilatih dengan *dataset* asli dan jumlah tetangga (nilai variabel *k*) pada model yang memiliki  
18      akurasi tertinggi juga berbeda dengan model yang dilatih dengan *dataset* asli.
- 19     ● *Dataset* yang diacak dengan teknik *Random Rotation Perturbation* terbukti masih dapat  
20      digunakan untuk penambangan data *clustering* menggunakan teknik *k-means* dengan hasil  
21      *cluster* yang sama persis dengan model yang dilatih dengan *dataset* asli. Sedangkan untuk  
22      teknik *Random Projection Perturbation* memiliki hasil *cluster* yang sangat mirip dengan model  
23      yang dilatih menggunakan *dataset* asli.
- 24     ● Waktu eksekusi dalam proses penambangan data klasifikasi dan *clustering* menggunakan  
25      *dataset* yang diacak dengan teknik *Random Rotation Perturbation* tidak memiliki perbedaan  
26      yang signifikan dibandingkan penambangan data dengan *dataset* asli. Sedangkan untuk waktu  
27      eksekusi dalam proses penambangan data klasifikasi dan *clustering* menggunakan *dataset* yang  
28      diacak dengan teknik *Random Projection Perturbation* lebih cepat dibandingkan penambangan  
29      data dengan *dataset* asli karena jumlah fitur yang ada pada *dataset* lebih sedikit.

## 30     6.2 Saran

31     Saran untuk penelitian selanjutnya adalah sebagai berikut.

- 32     ● Pada penelitian ini tidak menguji apakah data yang telah diacak berpotensi dapat dikembalikan  
33       ke aslinya. Untuk penelitian selanjutnya, metode *Randomization* dapat diuji untuk mengetahui  
34       apakah ada potensi data yang telah diacak dengan metode *Randomization* dapat dikembalikan  
35       ke aslinya.
- 36     ● Pada penelitian ini metode *Randomization* diuji dengan menggunakan penambangan data  
37       biasa, bukan dalam lingkungan *big data*. Untuk penelitian selanjutnya, metode *Randomization*

1 dapat diuji untuk diimplementasikan dalam lingkungan *big data* dan mengukur kecepatan  
2 dan kualitas teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*.

- 3 • Untuk penelitian selanjutnya dapat dilakukan pengujian lebih lanjut lagi terhadap teknik  
4 *Random Projection Perturbation* menggunakan data yang lain untuk mengetahui nilai variabel  
5 *epsilon* yang tepat sehingga kualitas model penambangan data yang dilatih menggunakan  
6 *dataset* yang telah diacak masih dapat ditoleransi perbedaannya dari yang asli dan bagaimana  
7 menanggulangi masalah nilai variabel *k* (dimensi minimal) pada teknik *Random Projection*  
8 *Perturbation* semakin menaik seiring model penambangan data dilatih dengan data yang baru.
- 9 • Untuk penelitian selanjutnya dapat dilakukan eksperimen mengenai perbedaan teknik *Principal*  
10 *Component Analysis* dengan teknik *Random Projection Perturbation* dalam mereduksi dimensi  
11 data. Eksperimen dapat bertujuan untuk menganalisa kualitas hasil dari kedua teknik tersebut  
12 dan apakah ada potensi sebuah data yang telah direduksi dengan kedua teknik tersebut dapat  
13 dikembalikan ke aslinya.



## DAFTAR REFERENSI

- [1] Keyvanpour, M. dan Moradi, S. S. (2011) Classification and evaluation the privacy preserving data mining techniques by using a data modification-based framework. *International Journal on Computer Science and Engineering*, **3**, 862–870.
- [2] NIST Special Publication 800-122 (2010) *Guide to Protecting the Confidentiality of Personally Identifiable Information (PII)*. National Institute of Standards and Technology, U.S. Department of Commerce, Erika McCallister, Tim Grance, Karen Scarfone. Gaithersburg, Maryland.
- [3] Oliveira, S. R. M. dan Zaïane, O. R. (2004) Towards standardization in privacy-preserving data mining. *ACM SIGKDD 3rd Workshop on Data Mining Standards*, **3**, 862–870.
- [4] MENDES, R. dan VILELA, J. P. (2017) Privacy-preserving data mining: Methods, metrics, and applications. *IEEE Access*, **5**, 10562–10582.
- [5] Han, J., Kamber, M., dan Pei, J. (2012) *Data Mining: Concepts and Techniques*, 3rd edition. Morgan Kaufmann, Waltham.
- [6] Torra, V. (2017) *Data Privacy: Foundations, New Developments and the Big Data Challenge*, 1st edition. Springer, Warsaw.
- [7] Chen, K. dan Liu, L. (2005) A random rotation perturbation approach to privacy preserving data classification. Technical Report GIT-CC-05-12. Georgia Institute of Technology, Georgia.
- [8] Agrawal, R. dan Srikant, R. (2000) Privacy preserving data mining. In *Proceedings of the ACM SIGMOD*, **3**, 439–450.
- [9] STEWART, G. W. (1980) The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, **17**, 403–409.
- [10] Johnson, W. B. dan Lindenstrauss, J. (1984) Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, **26**, 189–206.
- [11] Kun Liu, J. R., Hillol Kargupta (2006) Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, **18**, 92–106.
- [12] Ella Bingham, H. M. (2001) Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the ACM SIGKDD*, **7**, 245–250.
- [13] Rossum, G. V. (1995) Python tutorial. Technical Report CS-R9526. Centrum voor Wiskunde en Informatica (CWI), Amsterdam.
- [14] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E. W., Vand erPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa,

- F., van Mulbregt, P., dan Contributors, S. . . (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, **17**, 261–272.
- [15] Wes McKinney (2010) Data Structures for Statistical Computing in Python. Bagian dari Stéfan van der Walt dan Jarrod Millman (ed.), *Proceedings of the 9th Python in Science Conference*, pp. 56 – 61.
- [16] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., dan Duchesnay, E. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.
- [17] Hunter, J. D. (2007) Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, **9**, 90–95.

# LAMPIRAN A

## KODE PERANGKAT LUNAK *RANDOMIZATION*

### A.1 *Package Perturbation*

Dalam *package Perturbation* terdapat 3 buah kelas. Kode program untuk masing-masing kelas dalam *package* ini adalah sebagai berikut.

#### A.1.1 Kelas *Perturbation*

Kode program untuk kelas *Perturbation* adalah sebagai berikut.

Listing A.1: perturbation.py

```
1 from abc import ABC, abstractmethod
2
3
4 class Perturbation(ABC):
5     def __init__(self, dataset):
6         self._dataset = dataset
7         self._perturbedDataset = None
8
9     def getOriginalDataset(self):
10        return self._dataset
11
12    def getPerturbedDataset(self):
13        return self._perturbedDataset
14
15    def setDataset(self, dataset):
16        self._dataset = dataset
17
18    @abstractmethod
19    def perturbDataset(self):
20        pass
```

#### A.1.2 Kelas *RandomRotationPerturbation*

Kode program untuk kelas *RandomRotationPerturbation* adalah sebagai berikut.

Listing A.2: random\_rotation\_perturbation.py

```
1 from .perturbation import Perturbation
2 from matrix.random_translation_matrix import RandomTranslationMatrix
3
4
5 class RandomRotationPerturbation(Perturbation):
6     def __init__(self, dataset, randomTranslationMatrix, randomRotationMatrix):
7         super().__init__(dataset)
8
9         self._randomTranslationMatrix = randomTranslationMatrix
10        self._randomRotationMatrix = randomRotationMatrix
11
12    def perturbDataset(self):
13        dataset_with_ones = RandomTranslationMatrix.addAColumnOfOnes(self._dataset)
14        translated_dataset_with_ones = dataset_with_ones.multiply(self._randomTranslationMatrix)
15        translated_dataset = RandomTranslationMatrix.removeLastColumn(translated_dataset_with_ones)
16
17        self._perturbedDataset = translated_dataset.multiply(self._randomRotationMatrix)
18        return True
19
20    def getRandomTranslationMatrix(self):
21        return self._randomTranslationMatrix
22
23    def getRandomRotationMatrix(self):
24        return self._randomRotationMatrix
```

### A.1.3 Kelas *RandomProjectionPerturbation*

Kode program untuk kelas *RandomProjectionPerturbation* adalah sebagai berikut.

Listing A.3: random\_projection\_perturbation.py

```

1 from .perturbation import Perturbation
2 from sklearn.random_projection import johnson_lindenstrauss_min_dim, GaussianRandomProjection
3 from matrix.matrix import Matrix
4
5
6 class RandomProjectionPerturbation(Perturbation):
7     def __init__(self, dataset, epsilon, k=0, randomProjectionMatrix=False):
8         super().__init__(dataset)
9         self._epsilon = epsilon
10        self._k = k
11
12        self._MinK = None
13        self.calculateMinK()
14
15        self._transformer = self.GaussianRandomProjection(n_components=k, eps=epsilon,
16                                                        randomProjectionMatrix=randomProjectionMatrix)
17
18    def perturbDataset(self):
19        self._perturbedDataset = Matrix(self._transformer.fit_transform(self._dataset.getRawMatrix()))
20
21    def getEpsilon(self):
22        return self._transformer.eps
23
24    def getMinK(self):
25        return self._MinK
26
27    def calculateMinK(self):
28        self._MinK = johnson_lindenstrauss_min_dim(
29            self._dataset.getNumberofRows(), self._epsilon)
30
31    def checkMinK(self):
32        return self._MinK < self._dataset.getNumberofColumns()
33
34    def checkVariableK(self):
35        return self._MinK <= self._k < self._dataset.getNumberofColumns()
36
37    class GaussianRandomProjection(GaussianRandomProjection):
38        def __init__(self, n_components, eps, randomProjectionMatrix):
39            super().__init__(n_components=n_components, eps=eps)
40            self._randomProjectionMatrix = randomProjectionMatrix
41
42        def _make_random_matrix(self, n_components, n_features):
43            return self._randomProjectionMatrix.getRawMatrix()

```

## A.2 Package *Matrix*

Dalam *package Matrix* terdapat 4 buah kelas. Kode program untuk masing-masing kelas dalam *package* ini adalah sebagai berikut.

### A.2.1 Kelas *Matrix*

Kode program untuk kelas *Matrix* adalah sebagai berikut.

Listing A.4: matrix.py

```

1 from numpy.linalg import det
2
3
4 class Matrix:
5     def __init__(self, matrix):
6         self._matrix = matrix
7
8     def getNumberofRows(self):
9         return self._matrix.shape[0]
10
11    def getNumberofColumns(self):
12        return self._matrix.shape[1]
13
14    def multiply(self, multiplierMatrix):
15        return Matrix(self._matrix @ multiplierMatrix.getRawMatrix())
16
17    def get(self, row, col):
18        return self._matrix[row][col]
19
20    def getRawMatrix(self):
21        return self._matrix
22
23    def determinant(self):
24        return det(self._matrix)
25
26    def transpose(self):
27        return Matrix(self._matrix.T)

```

### A.2.2 Kelas *RandomRotationMatrix*

Kode program untuk kelas *RandomRotationMatrix* adalah sebagai berikut.

Listing A.5: random\_rotation\_matrix.py

```

1 from .matrix import Matrix
2 from scipy.stats import special_ortho_group
3
4
5 class RandomRotationMatrix:
6     @staticmethod
7     def generate(dimension):
8         return Matrix(special_ortho_group.rvs(dim=dimension))

```

### A.2.3 Kelas *RandomTranslationMatrix*

Kode program untuk kelas *RandomTranslationMatrix* adalah sebagai berikut.

Listing A.6: random\_translation\_matrix.py

```

1 from .matrix import Matrix
2 from numpy import identity, append, ones
3 from random import randint
4
5
6 class RandomTranslationMatrix:
7     @staticmethod
8     def generate(dimension):
9         matrix = identity(dimension)
10        matrix[-1][-1] = [randint(0, 100) for x in matrix[-1][-1]]
11
12    return Matrix(matrix)
13
14 @staticmethod
15 def addAColumnOfOnes(matrix):
16    matrixTransposed = matrix.transpose()
17    return Matrix(append(matrixTransposed.getRawMatrix(), ones((1, matrixTransposed.getNumberOfColumns())), axis=0)).transpose()
18
19
20 @staticmethod
21 def removeLastColumn(matrix):
22    return Matrix(matrix.transpose().getRawMatrix()[:-1].T)

```

### A.2.4 Kelas *RandomProjectionMatrix*

Kode program untuk kelas *RandomProjectionMatrix* adalah sebagai berikut.

Listing A.7: random\_projection\_matrix.py

```

1 from sklearn.random_projection import GaussianRandomProjection
2
3 from matrix.matrix import Matrix
4
5
6 class RandomProjectionMatrix:
7     @staticmethod
8     def generate(originalDimension, k, epsilon):
9         transformer = GaussianRandomProjection(n_components=k, eps=epsilon)
10        return Matrix(transformer._make_random_matrix(k, originalDimension))

```

## A.3 Package Preprocessor

Dalam *package Preprocessor* terdapat 3 buah kelas. Kode program untuk masing-masing kelas dalam *package* ini adalah sebagai berikut.

### A.3.1 Kelas *CSVPreprocessor*

Kode program untuk kelas *CSVPreprocessor* adalah sebagai berikut.

Listing A.8: csv\_preprocessor.py

```

1 import ntpath
2
3 from numpy import array
4 from pandas import DataFrame, read_csv
5 from matrix.matrix import Matrix
6

```

```

7 def path_leaf(path):
8     head, tail = ntpath.split(path)
9     return tail or ntpath.basename(head)
10
11
12 class CSVPreprocessor:
13     def __init__(self):
14         self._filePath = None
15         self._dataCSV = None
16
17     def csvToMatrix(self):
18         self._matrix = Matrix(array(self._dataCSV.values))
19         self._nameOfColumns = self._dataCSV.columns
20
21     def getFilePath(self):
22         return self._filePath
23
24     def getNameOfColumns(self):
25         return self._nameOfColumns
26
27     def getFileName(self):
28         return path_leaf(self._filePath)
29
30     def matrixToCSV(self, matrix, newFilePath):
31         try:
32             if matrix.getNumberOfColumns() != self._nameOfColumns.size:
33                 df = DataFrame(matrix.getRawMatrix())
34             else:
35                 df = DataFrame(matrix.getRawMatrix(), columns=self._nameOfColumns)
36
37             df.to_csv(newFilePath, index=False)
38             return True
39         except:
40             return False
41
42
43
44
45

```

### A.3.2 Kelas *ProjectionMatrixPreprocessor*

Kode program untuk kelas *ProjectionMatrixPreprocessor* adalah sebagai berikut.

Listing A.9: *projection\_matrix\_preprocessor.py*

```

1 from pandas import DataFrame
2
3 from preprocessor.csv_preprocessor import CSVPreprocessor
4
5
6 class ProjectionMatrixPreprocessor:
7     def __init__(self):
8         self._projectionMatrix = None
9
10    def readFromCSV(self, filePath, dimension):
11        csvPreprocessor = CSVPreprocessor()
12        csvPreprocessor.readCSV(filePath)
13        matrix = csvPreprocessor.csvToMatrix()
14        if matrix.getNumberOfColumns() != dimension or matrix.getNumberOfRows() >= dimension:
15            return False
16        else:
17            self._projectionMatrix = matrix
18            return True
19
20    def getProjectionMatrix(self):
21        return self._projectionMatrix
22
23    @staticmethod
24    def saveToCSV(path, projectionMatrix):
25        try:
26            df = DataFrame(projectionMatrix.getRawMatrix())
27            df.to_csv(path, index=False)
28            return True
29        except:
30            return False

```

### A.3.3 Kelas *RotationMatrixPreprocessor*

Kode program untuk kelas *ProjectionMatrixPreprocessor* adalah sebagai berikut.

Listing A.10: *rotation\_matrix\_preprocessor.py*

```

1 from pandas import DataFrame, concat
2
3 from matrix.matrix import Matrix
4 from preprocessor.csv_preprocessor import CSVPreprocessor
5
6
7 class RotationMatrixPreprocessor:

```

```

8  def __init__(self):
9      self._randomTranslationMatrix = None
10     self._randomRotationMatrix = None
11
12    def readFromCSV(self, filePath, dimension):
13        csvPreprocessor = CSVPreprocessor()
14        csvPreprocessor.readCSV(filePath)
15        matrix = csvPreprocessor.csvToMatrix()
16
17        if matrix.getNumberOfRows() != dimension + 1 or matrix.getNumberOfColumns() != dimension * 2 + 1:
18            return False
19        else:
20            self._randomRotationMatrix = Matrix(matrix.getRawMatrix()[:dimension, :dimension])
21            self._randomTranslationMatrix = Matrix(matrix.getRawMatrix(:, dimension:))
22            return True
23
24    def getRandomRotationMatrix(self):
25        return self._randomRotationMatrix
26
27    def getRandomTranslationMatrix(self):
28        return self._randomTranslationMatrix
29
30    @staticmethod
31    def saveToCSV(path, randomRotationMatrix, randomTranslationMatrix):
32        try:
33            df1 = DataFrame(randomRotationMatrix.getRawMatrix())
34            df2 = DataFrame(randomTranslationMatrix.getRawMatrix())
35            df = concat([df1, df2], axis=1)
36            df.to_csv(path, index=False)
37            return True
38        except:
39            return False

```

## A.4 Package View

Dalam *package View* terdapat beberapa buah kelas yang berfungsi untuk menangani antarmuka perangkat lunak yang dibuat dengan *framework* Kivy. Kode program untuk masing-masing dokumen dalam *package* ini adalah sebagai berikut.

### A.4.1 Dokumen *main\_menu.py*

Kode program untuk dokumen *main\_menu.py* adalah sebagai berikut.

Listing A.11: *main\_menu.py*

```

1 import pathlib
2 from threading import Thread
3 from time import time
4
5 from kivy.properties import ObjectProperty
6 from kivy.uix.behaviors import FocusBehavior
7 from kivy.uix.gridlayout import GridLayout
8 from kivy.uix.label import Label
9 from kivy.uix.popup import Popup
10 from plyer import filechooser
11
12 from matrix.random_projection_matrix import RandomProjectionMatrix
13 from matrix.random_rotation_matrix import RandomRotationMatrix
14 from matrix.random_translation_matrix import RandomTranslationMatrix
15 from perturbation.random_projection_perturbation import RandomProjectionPerturbation
16 from perturbation.random_rotation_perturbation import RandomRotationPerturbation
17 from preprocessor.csv_preprocessor import CSVPreprocessor
18 from preprocessor.projection_matrix_preprocessor import ProjectionMatrixPreprocessor
19 from preprocessor.rotation_matrix_preprocessor import RotationMatrixPreprocessor
20
21
22 def hide_widget(wid, do_hide=True):
23     if hasattr(wid, 'saved_attrs'):
24         if not do_hide:
25             wid.height, wid.size_hint_y, wid.opacity, wid.disabled = wid.saved_attrs
26             del wid.saved_attrs
27     elif do_hide:
28         wid.saved_attrs = wid.height, wid.size_hint_y, wid.opacity, wid.disabled
29         wid.height, wid.size_hint_y, wid.opacity, wid.disabled = 0, None, 0, True
30
31
32 class MainMenu(GridLayout):
33     load_file_path = ObjectProperty(None)
34
35     dimension_label = ObjectProperty(None)
36     dimension_value = ObjectProperty(None)
37     epsilon_label = ObjectProperty(None)
38     epsilon_value = ObjectProperty(None)
39     calculate_k_button = ObjectProperty(None)
40
41     technique_spinner = ObjectProperty(None)
42
43     dataset_description_layout = ObjectProperty(None)

```

```

44 randomization_result_description_layout = ObjectProperty(None)
45
46 load_matrix_path = ObjectProperty(None)
47
48 def __init__(self, **kwargs):
49     super(GridLayout, self).__init__(**kwargs)
50     self.on_technique_spinner_select("Random_Rotation_Perturbation")
51
52 def calculate_k_button_action(self):
53     if self.load_file_path_empty() or self.epsilon_not_valid():
54         return
55
56     randomizer = RandomProjectionPerturbation(self.dataset, float(self.epsilon_value.text))
57
58     self.calculate_and_check_k(randomizer)
59
60 def calculate_and_check_k(self, randomizer):
61     self.k_label.change_value(str(randomizer.getMinK()))
62
63     if not randomizer.checkMinK():
64         WarningPopup().open("Nilai_minimal_variabel_K_lebih_besar_dari_dimensi_dataset!\nMohon_mengganti_nilai_variabel_Epsilon!")
65         return False
66
67     return True
68
69 def browse_load_button_action(self):
70     self.path = self.browse_load_file()
71     if not self.path:
72         return
73
74     self.loading_popup = LoadingPopup()
75     self.loading_popup.title = "Memuat_Dokumen"
76     self.loading_popup.open()
77     self.loading_popup.progress(40)
78
79     load_file_thread = Thread(target=self.browse_load)
80     load_file_thread.start()
81
82 def browse_load(self):
83     self.csv_preprocessor = CSVPreprocessor()
84     try:
85         self.csv_preprocessor.readCSV(self.path)
86     except:
87         WarningPopup().open("Dokumen_CSV_yang_dimasukkan_tidak_sesuai_dengan_persyaratan!")
88         self.loading_popup.dismiss()
89         return
90
91     self.dataset = self.csv_preprocessor.csvToMatrix()
92
93     self.randomization_result_description_layout.clear_widgets()
94     self.load_file_path.text = self.path
95     if self.technique_spinner.text == "Random_Rotation_Perturbation":
96         self.load_matrix_path.text = "Pilih_matriks_rotasi_yang_ingin_digunakan"
97     else:
98         self.load_matrix_path.text = "Pilih_matriks_proyeksi_yang_ingin_digunakan"
99     self.randomTranslationMatrix = None
100    self.randomRotationMatrix = None
101    self.randomProjectionMatrix = None
102
103    self.dataset_description_layout.clear_widgets()
104
105    self.loading_popup.progress(60)
106    self.dataset_description_layout.add_widget(
107        DescriptionLabel("Nama_Dokumen", self.csv_preprocessor.getFileName()))
108    self.dataset_description_layout.add_widget(
109        DescriptionLabel("Jumlah_Baris", str(self.dataset.getNumberofRows())))
110    self.loading_popup.progress(80)
111    self.dataset_description_layout.add_widget(
112        DescriptionLabel("Jumlah_Kolom", str(self.dataset.getNumberofColumns())))
113    self.k_label = DescriptionLabel("Nilai_Minimal_Variabel_K", "Belum_dihitung")
114    self.dataset_description_layout.add_widget(self.k_label)
115
116    if self.technique_spinner.text == "Random_Rotation_Perturbation":
117        hide_widget(self.k_label)
118    self.loading_popup.progress(100)
119    self.loading_popup.dismiss()
120
121 def browse_load_file(self):
122     path = filechooser.open_file(title="Pilih_dokumen_CSV..",
123                                 filters=[("Comma-separated_Values", "*.csv")])
124     if len(path) != 0:
125         return path[0]
126     else:
127         return False
128
129 def browse_save(self):
130     path = filechooser.save_file(title="Simpan_hasil_berbentuk_dokumen_CSV..",
131                                 filters=[("Comma-separated_Values", "*.csv")])
132
133     if len(path) != 0:
134         if pathlib.Path(path[0]).suffix == ".csv":
135             return path[0]
136         else:
137             return path[0] + ".csv"
138     else:
139         return False
140
141 def on_technique_spinner_select(self, text):
142     self.randomization_result_description_layout.clear_widgets()

```

```

142     self.randomTranslationMatrix = None
143     self.randomRotationMatrix = None
144     self.randomProjectionMatrix = None
145     if text == "Random_Rotation_Perturbation":
146         self.load_matrix_path.text = "Pilih_matriks_rotasi_yang_ingin_digunakan"
147         hide_widget(self.dimension_label)
148         hide_widget(self.dimension_value)
149         hide_widget(self.epsilon_label)
150         hide_widget(self.epsilon_value)
151         try:
152             hide_widget(self.k_label)
153         except AttributeError:
154             pass
155         hide_widget(self.calculate_k_button)
156     else:
157         self.load_matrix_path.text = "Pilih_matriks_proyeksi_yang_ingin_digunakan"
158         hide_widget(self.dimension_label, False)
159         hide_widget(self.dimension_value, False)
160         hide_widget(self.epsilon_label, False)
161         hide_widget(self.epsilon_value, False)
162         try:
163             hide_widget(self.k_label, False)
164         except AttributeError:
165             pass
166         hide_widget(self.calculate_k_button, False)
167
168     def load_file_path_empty(self):
169         if self.load_file_path.text == "Belum_ada_dokumen_CSV_yang_dipilih_untuk_diacak":
170             WarningPopup().open("Mohon_memilih_dataset_(dokumen_CSV)_terlebih_dahulu!")
171             return True
172         else:
173             return False
174
175     def load_matrix_path_empty(self):
176         if self.load_matrix_path.text == "Pilih_matriks_rotasi_yang_ingin_digunakan" or self.load_matrix_path.text == "Pilih_matriks_proyeksi_yang_ingin_digunakan":
177             WarningPopup().open("Mohon_memilih_matriks_(dokumen_CSV)_terlebih_dahulu!")
178             return True
179         else:
180             return False
181
182     def epsilon_not_valid(self):
183         if self.epsilon_value.text == "":
184             WarningPopup().open("Mohon_menentukan_nilai_variabel_Epsilon_terlebih_dahulu!")
185             return True
186
187         try:
188             float(self.epsilon_value.text)
189         except ValueError:
190             WarningPopup().open(
191                 "Anda_tidak_memasukan_nilai(bilangan_desimal)_dengan_benar_untuk_variabel_Epsilon!\nMohon_periksa_"
192                 "kembali_dengan_teliti!")
193             return True
194
195         if not 1 > float(self.epsilon_value.text) > 0:
196             WarningPopup().open(
197                 "Nilai_variabel_Epsilon_wajib_berada_pada_rentang_nilai_lebih_dari_0_dan_kurang_dari_1!")
198             return True
199
200         return False
201
202     def dimension_target_not_valid(self):
203         if self.dimension_value.text == "":
204             WarningPopup().open("Mohon_menentukan_nilai_variabel_K_terlebih_dahulu!")
205             return True
206
207         try:
208             int(self.dimension_value.text)
209         except ValueError:
210             WarningPopup().open(
211                 "Anda_tidak_memasukan_nilai(bilangan_bulat)_dengan_benar_untuk_variabel_K!\nMohon_periksa_kembali_"
212                 "dengan_teliti!")
213             return True
214
215         return False
216
217     def create_save_matrix_button_action(self):
218         if self.load_file_path_empty():
219             return
220
221         if self.technique_spinner.text == "Random_Projection_Perturbation":
222             if self.epsilon_not_valid() or self.dimension_target_not_valid():
223                 return
224
225             randomizer = RandomProjectionPerturbation(self.dataset, float(self.epsilon_value.text),
226                                                 int(self.dimension_value.text))
227
228             if not self.calculate_and_check_k(randomizer):
229                 return
230
231             if not randomizer.checkVariableK():
232                 WarningPopup().open(
233                     "Nilai_dari_variabel_K_wajib_lebih_besar_dari_atau_sama_dengan_nilai_minimal_variabel_K_"
234                     "dan_lebih_kecil_dari_dimensi_dataset_yang_ingin_diacak!\nMohon_mengganti_nilai_variabel_K!")
235             return
236
237         self.matrix_path = self.browse_save()
238         if not self.matrix_path:
239             return

```

```

240
241     self.loading_popup = LoadingPopup()
242     self.loading_popup.title = "Membuat_dan_Menyimpan_Matriks"
243     self.loading_popup.open()
244
245     self.loading_popup.progress(10)
246
247     randomize_thread = Thread(target=self.create_save_matrix)
248     randomize_thread.start()
249
250     def create_save_matrix(self):
251         if self.technique_spinner.text == "Random_Rotation_Perturbation":
252             start_time = time()
253             self.randomRotationMatrix = RandomRotationMatrix.generate(self.dataset.getNumberOfColumns())
254             self.randomTranslationMatrix = RandomTranslationMatrix.generate(self.dataset.getNumberOfColumns() + 1)
255             self.create_matrix_time = time() - start_time
256
257             self.loading_popup.progress(50)
258
259             if RotationMatrixPreprocessor.saveToCSV(self.matrix_path, self.randomRotationMatrix, self.randomTranslationMatrix):
260                 self.loading_popup.dismiss()
261                 WarningPopup().open(
262                     "Tolong_menutup_dokumen_yang_dipilih_yaitu_" + self.matrix_path)
263                 return
264
265             else:
266                 start_time = time()
267                 self.randomProjectionMatrix = RandomProjectionMatrix.generate(self.dataset.getNumberOfColumns(),
268                                     int(self.dimension_value.text),
269                                     float(self.epsilon_value.text))
270
271                 self.create_matrix_time = time() - start_time
272
273                 self.loading_popup.progress(50)
274
275                 if ProjectionMatrixPreprocessor.saveToCSV(self.matrix_path, self.randomProjectionMatrix):
276                     self.loading_popup.dismiss()
277                     WarningPopup().open(
278                         "Tolong_menutup_dokumen_yang_dipilih_yaitu_" + self.matrix_path)
279                     return
280
281             self.load_matrix_path.text = self.matrix_path
282
283             self.loading_popup.progress(100)
284             self.loading_popup.dismiss()
285
286     def import_matrix_button_action(self):
287         self.create_matrix_time = 0
288         if self.load_file_path_empty():
289             return
290
291         self.matrix_path = self.browse_load_file()
292         if not self.matrix_path:
293             return
294
295         self.loading_popup = LoadingPopup()
296         self.loading_popup.title = "Memuat_Matriks"
297         self.loading_popup.open()
298
299         self.loading_popup.progress(10)
300
301         randomize_thread = Thread(target=self.import_matrix)
302         randomize_thread.start()
303
304     def import_matrix(self):
305         if self.technique_spinner.text == "Random_Rotation_Perturbation":
306             try:
307                 rotationMatrixPreprocessor = RotationMatrixPreprocessor()
308                 if not rotationMatrixPreprocessor.readFromCSV(self.matrix_path, self.dataset.getNumberOfColumns()):
309                     self.loading_popup.dismiss()
310                     WarningPopup().open(
311                         "Matriks_yang_dipilih_tidak_sesuai_dengan_dataset!")
312
313                 self.loading_popup.progress(50)
314
315                 self.randomTranslationMatrix = rotationMatrixPreprocessor.getRandomTranslationMatrix()
316                 self.randomRotationMatrix = rotationMatrixPreprocessor.getRandomRotationMatrix()
317
318             except:
319                 self.loading_popup.dismiss()
320                 WarningPopup().open("Dokumen_CSV_yang_dimasukkan_tidak_sesuai_dengan_persyaratan!")
321
322             else:
323                 projectionMatrixPreprocessor = ProjectionMatrixPreprocessor()
324                 if not projectionMatrixPreprocessor.readFromCSV(self.matrix_path, self.dataset.getNumberOfColumns()):
325                     self.loading_popup.dismiss()
326                     WarningPopup().open(
327                         "Matriks_yang_dipilih_tidak_sesuai_dengan_dataset!")
328
329             self.loading_popup.progress(50)
330
331             self.randomProjectionMatrix = projectionMatrixPreprocessor.getProjectionMatrix()
332             self.dimension_value.text = str(self.randomProjectionMatrix.getNumberOfRows())
333
334             self.load_matrix_path.text = self.matrix_path
335
336             self.loading_popup.progress(100)
337             self.loading_popup.dismiss()
338
339     def randomize_button_action(self):

```

```

339     self.randomization_result_description_layout.clear_widgets()
340
341     if self.load_file_path_empty() or self.load_matrix_path_empty():
342         return
343
344     if self.technique_spinner.text == "Random_Projection_Perturbation":
345         if self.epsilon_not_valid() or self.dimension_target_not_valid():
346             return
347         elif self.randomProjectionMatrix.getNumberOfRows() != int(self.dimension_value.text):
348             WarningPopup().open("Dimensi_matriks_tidak_sama_dengan_dimensi_target_yang_diinginkan!")
349             return
350
351     self.loading_popup = LoadingPopup()
352     self.loading_popup.title = self.technique_spinner.text
353     self.loading_popup.open()
354
355     self.loading_popup.progress(10)
356
357     randomize_thread = Thread(target=self.randomize)
358     randomize_thread.start()
359
360 def randomize(self):
361     try:
362         self.loading_popup.progress(30)
363
364         if self.technique_spinner.text == "Random_Rotation_Perturbation":
365             start_time = time()
366             randomizer = RandomRotationPerturbation(self.dataset, self.randomTranslationMatrix,
367                                                       self.randomRotationMatrix)
368             initialize_time = time() - start_time
369             self.loading_popup.progress(50)
370             try:
371                 start_time = time()
372                 randomizer.perturbDataset()
373                 perturb_time = time() - start_time
374             except TypeException:
375                 self.loading_popup.dismiss()
376                 WarningPopup().open(
377                     "Ada_nilai.yang_tidak_bersifat_numerik_pada_dataset!_Semua_nilai_harus_berupa_numerik!")
378                 self.randomization_result_description_layout.add_widget(DescriptionLabel("Status", "GAGAL"))
379                 self.randomization_result_description_layout.add_widget(DescriptionLabel("Alasan",
380
381                                         "Ada.nilai.yang.tidak."
382                                         "bersifat.numerik.pada."
383                                         "dataset!.Semua.nilai.harus."
384                                         "berupa.numerik!"))
385
386         else:
387             start_time = time()
388             randomizer = RandomProjectionPerturbation(self.dataset, float(self.epsilon_value.text),
389                                                       int(self.dimension_value.text), self.randomProjectionMatrix)
390             initialize_time = time() - start_time
391
392             if not self.calculate_and_check_K(randomizer):
393                 self.loading_popup.dismiss()
394                 return
395
396             if not randomizer.checkVariableK():
397                 self.loading_popup.dismiss()
398                 WarningPopup().open(
399                     "Nilai_dari_variabel_K_wajib_lebih_besar_dari_atau_sama_dengan_nilai_minimal_variabel_K."
400                     "dan_lebih_kecil_dari_dimensi_dataset_yang_ingin_diacak!\nMohon_mengganti_nilai_variabel_K!")
401                 self.randomization_result_description_layout.add_widget(DescriptionLabel("Status", "GAGAL"))
402                 self.randomization_result_description_layout.add_widget(DescriptionLabel("Alasan",
403
404                                         "Nilai_dari_variabel_K."
405                                         "wajib_berada_pada_rentang."
406                                         "nilai_lebih_besar_dari_atau."
407                                         "sama_dengan_K."
408                                         "dan_lebih_kecil_dari_atau."
409                                         "sama_dengan_dimensi_dataset."
410                                         "yang_ingin."
411                                         "diacak!\nMohon."
412                                         "mengganti."
413                                         "variabel_K!"))
414
415             return
416             self.loading_popup.progress(50)
417             try:
418                 start_time = time()
419                 randomizer.perturbDataset()
420                 perturb_time = time() - start_time
421             except TypeException:
422                 self.loading_popup.dismiss()
423                 WarningPopup().open(
424                     "Ada_nilai.yang_tidak_bersifat_numerik_pada_dataset!_Semua_nilai_harus_berupa_numerik!")
425                 self.randomization_result_description_layout.add_widget(DescriptionLabel("Status", "GAGAL"))
426                 self.randomization_result_description_layout.add_widget(DescriptionLabel("Alasan", "Ada.nilai.yang."
427
428                                         "tidak_bersifat."
429                                         "numerik_pada."
430                                         "dataset!.Semua."
431                                         "nilai_harus."
432                                         "berupa_numerik!"))
433
434             self.loading_popup.progress(80)
435
436             save_path = self.browse_save()
437
438             if not save_path:
439                 self.loading_popup.dismiss()
440                 WarningPopup().open("Penyimpanan_dokumen_dibatalkan!")

```

```

438         self.randomization_result_description_layout.add_widget(DescriptionLabel("Status", "GAGAL"))
439         self.randomization_result_description_layout.add_widget(
440             DescriptionLabel("Alasan", "Penyimpanan_dokumen_dibatalkan!"))
441         return
442
443     if self.csv_preprocessor.matrixToCSV(randomizer.getPerturbedDataset(), save_path):
444         self.loading_popup.dismiss()
445         WarningPopup().open()
446         "Tolong_menutup_dokumen_yang_dipilih_yaitu_" + save_path)
447         self.randomization_result_description_layout.add_widget(DescriptionLabel("Status", "GAGAL"))
448         self.randomization_result_description_layout.add_widget(DescriptionLabel("Alasan",
449                                         "Sudah_ada_dokumen_yang_"
450                                         "mempunyai_nama_yang_sama_yaitu_" +
451                                         save_path +
452                                         "_dan_dibuka_oleh_program_lain!"
453                                         "\nMohon_ditutup_terlebih_dahulu!"))
454
455     return
456
457     self.randomization_result_description_layout.add_widget(DescriptionLabel("Status", "BERHASIL"))
458     self.randomization_result_description_layout.add_widget(
459         DescriptionLabel("Lokasi_dokumen", save_path))
460     self.randomization_result_description_layout.add_widget(
461         DescriptionLabel("Lokasi_dokumen_matrix_yang_dipakai", self.matrix_path))
462     self.randomization_result_description_layout.add_widget(
463         DescriptionLabel("Jumlah_kolom", str(randomizer.getPerturbedDataset().getNumberOfColumns())))
464     if self.technique_spinner.text == "Random_Projection_Perturbation":
465         self.randomization_result_description_layout.add_widget(
466             DescriptionLabel("Nilai_variabel_epsilon_yang_digunakan", str(randomizer.getEpsilon())))
467         self.randomization_result_description_layout.add_widget(
468             DescriptionLabel("Nilai_variabel_K_yang_digunakan", self.dimension_value.text))
469     self.randomization_result_description_layout.add_widget(
470         DescriptionLabel("Waktu_eksekusi", str((initialize_time + perturb_time + self.create_matrix_time) % 60) + "detik"))
471
472     self.loading_popup.progress(100)
473     self.loading_popup.dismiss()
474     WarningPopup().open(self.technique_spinner.text + "berhasil_diterapkan_pada_dataset_yang_dipilih!")
475 except Exception as e:
476     self.loading_popup.dismiss()
477     WarningPopup().open(str(e))
478     self.randomization_result_description_layout.add_widget(DescriptionLabel("Status", "GAGAL"))
479     self.randomization_result_description_layout.add_widget(DescriptionLabel("Alasan", str(e)))
480
481 class WarningPopup(Popup):
482     text_label = ObjectProperty(None)
483     close_button = ObjectProperty(None)
484
485     def open(self, text, *largs, **kwargs):
486         self.text_label.text = text
487         # self.close_button.focus = True
488         super().open()
489
490
491 class LoadingPopup(Popup):
492     progress_bar = ObjectProperty(None)
493     progress_label = ObjectProperty(None)
494
495     def progress(self, value):
496         self.progress_bar.value = value
497         self.progress_label.text = str(value) + "%"
498
499
500 class DescriptionLabel(Label):
501     def __init__(self, title, value):
502         self.title = title
503         super().__init__(text=title + ":" + value)
504
505     def change_value(self, value):
506         self.text = self.title + ":" + value
507

```

#### A.4.2 Dokumen *randomization\_app.py*

Kode program untuk dokumen *randomization\_app.py* adalah sebagai berikut.

Listing A.12: *randomization\_app.py*

```

1 from kivy.config import Config
2 Config.set('graphics', 'width', '1024')
3 Config.set('graphics', 'height', '600')
4 Config.set('kivy', 'window_icon', 'view/assets/r.ico')
5 Config.set('input', 'mouse', 'mouse,multitouch_on_demand')
6 Config.set('kivy', 'exit_on_escape', '0')
7
8 from kivy.uix.textinput import TextInput
9 from kivy.uix.spinner import Spinner, SpinnerOption
10 from kivy.core.window import Window
11 # Window.clearcolor = (214/255.0, 217/255.0, 223/255.0, 1)
12 Window.clearcolor = (1, 1, 1, 1)
13 from kivy.properties import BooleanProperty, ObjectProperty
14 from kivy.app import App
15 from kivy.uix.button import Button
16 from kivy.uix.behaviors import FocusBehavior
17

```

```

18
19 from .main_menu import MainMenu
20 class HoverBehavior(object):
21     """Hover behavior.
22     :Events:
23         'on_enter'
24             Fired when mouse enter the bbox of the widget.
25         'on_leave'
26             Fired when the mouse exit the widget
27
28
29     hovered = BooleanProperty(False)
30     border_point= ObjectProperty(None)
31     '''Contains the last relevant point received by the Hoverable. This can
32     be used in 'on_enter' or 'on_leave' in order to know where was dispatched the event.
33     '''
34
35     def __init__(self, **kwargs):
36         self.register_event_type('on_enter')
37         self.register_event_type('on_leave')
38         Window.bind(mouse_pos=self.on_mouse_pos)
39         super(HoverBehavior, self).__init__(**kwargs)
40
41     def on_mouse_pos(self, *args):
42         if not self.get_root_window():
43             return # do proceed if I'm not displayed <=> If have no parent
44         pos = args[1]
45         #Next line to_widget allow to compensate for relative layout
46         inside = self.collide_point(*self.to_widget(*pos))
47         if self.hovered == inside:
48             #We have already done what was needed
49             return
50         self.border_point = pos
51         self.hovered = inside
52         if inside:
53             self.dispatch('on_enter')
54         else:
55             self.dispatch('on_leave')
56
57     def on_enter(self):
58         pass
59
60     def on_leave(self):
61         pass
62
63 from kivy.factory import Factory
64 Factory.register('HoverBehavior', HoverBehavior)
65
66 class HoverButton(Button, HoverBehavior):
67     def on_enter(self, *args):
68         Window.set_system_cursor("hand")
69         self.background_color = 1,1,1,1
70
71     def on_leave(self, *args):
72         Window.set_system_cursor("arrow")
73         self.background_color = 0,0,0,0
74
75 class PopupCloseButton(FocusBehavior, HoverButton):
76     def keyboard_on_key_down(self, window, keycode, text, modifiers):
77         super().keyboard_on_key_down(window, keycode, text, modifiers)
78         if keycode[1] == 'enter' or keycode[1] == 'numpadenter': # deal with cycle
79             self.trigger_action()
80
81         return True
82     return False
83
84 class HoverSpinner(Spinner, HoverBehavior):
85     def on_enter(self, *args):
86         Window.set_system_cursor("hand")
87
88     def on_leave(self, *args):
89         Window.set_system_cursor("arrow")
90
91 class HoverTextInput(TextInput, HoverBehavior):
92     def on_enter(self, *args):
93         Window.set_system_cursor("ibeam")
94
95     def on_leave(self, *args):
96         Window.set_system_cursor("arrow")
97
98 class RandomizationApp(App):
99     def build(self):
100        return MainMenu()

```

### A.4.3 Dokumen *randomization.kv*

Kode program untuk dokumen *randomization.kv* adalah sebagai berikut.

Listing A.13: randomization.kv

```

1 #:kivy 2.0.0
2 #:import Window kivy.core.window.Window
3
4 <MainMenu>
5     cols: 1

```

```

6   size: root.width, root.height
7   spacing: 10
8
9   load_file_path: load_file_path
10
11  dimension_label: dimension_label
12  epsilon_label: epsilon_label
13  dimension_value: dimension_value
14  epsilon_value: epsilon_value
15  calculate_k_button: calculate_k_button
16
17  technique_spinner: technique_spinner
18
19  dataset_description_layout: dataset_description_layout
20  randomization_result_description_layout: randomization_result_description_layout
21
22  load_matrix_path: load_matrix_path
23
24
25  GridLayout:
26      cols: 3
27      size_hint: (root.width, 0.09)
28      spacing: 5
29      Image:
30          source: 'view/assets/browse.jpg'
31          size_hint: (0.1, root.height)
32      Label:
33          id: load_file_path
34          text: "Belum ada dokumen CSV yang dipilih untuk diacak"
35          size: self.texture_size
36          text_size: self.size
37          halign: 'left'
38      HoverButton:
39          text: "Pilih Dokumen CSV"
40          size_hint: (0.25, root.height)
41          on_release: root.browse_load_button_action()
42
43
44  GridLayout:
45      cols: 1
46      size_hint: (root.width, 0.09)
47      HoverSpinner:
48          color: 1,1,1,1
49          id: technique_spinner
50          size_hint: (root.width, root.height)
51          text: 'Random Rotation Perturbation'
52          values: 'Random Rotation Perturbation', 'Random Projection Perturbation'
53          on_text: root.on_technique_spinner_select(self.text)
54
55
56  GridLayout:
57      cols: 4
58      size_hint: (root.width, 0.09)
59      spacing: 5
60      Image:
61          source: 'view/assets/matrix.png'
62          size_hint: (0.1, root.height)
63      Label:
64          id: load_matrix_path
65          text: "Pilih matriks rotasi yang ingin digunakan"
66          size: self.texture_size
67          text_size: self.size
68          halign: 'left'
69      HoverButton:
70          text: "Buat dan Simpan Matriks"
71          size_hint: (0.3, root.height)
72          on_release: root.create_save_matrix_button_action()
73      HoverButton:
74          text: "Impor Matriks"
75          size_hint: (0.2, root.height)
76          on_release: root.import_matrix_button_action()
77
78
79  GridLayout:
80      cols: 6
81      size_hint: (root.width, 0.09)
82      spacing: 10
83      Label:
84          id: epsilon_label
85          size_hint: (0.07, root.height)
86          text: "Epsilon"
87      HoverTextInput:
88          id: epsilon_value
89          size_hint: (0.05, root.height)
90          input_filter: 'float'
91          write_tab: False
92          multiline: False
93          on_text_validate: root.calculate_k_button_action()
94      HoverButton:
95          id: calculate_k_button
96          text: "Hitung Nilai Min K"
97          size_hint: (0.12, root.height)
98          on_release: root.calculate_k_button_action()
99      Label:
100         id: dimension_label
101         size_hint: (0.07, root.height)
102         text: "Variabel K"
103     HoverTextInput:
104         id: dimension_value

```

```
105     size_hint: (0.05, root.height)
106     input_filter: 'int'
107     write_tab: False
108     multiline: False
109 HoverButton:
110     text: "Acak dan Simpan"
111     size_hint: (0.4, root.height)
112     on_release: root.randomize_button_action()
113
114
115
116 BoxLayout:
117     orientation: "vertical"
118     BoxLayout:
119         orientation: "horizontal"
120         BoxLayout:
121             orientation: "vertical"
122             canvas.before:
123                 Color:
124                     rgba: .5, .5, .5, 1
125                 Line:
126                     width: 2
127                     rectangle: self.x, self.y, self.width, self.height
128             Label:
129                 text: "Deskripsi Dataset"
130                 size_hint: (1, 0.2)
131                 text_size: self.size
132                 halign: "center"
133                 valign: "top"
134                 padding: (20, 20)
135
136             BoxLayout:
137                 id: dataset_description_layout
138                 orientation: "vertical"
139
140
141             BoxLayout:
142                 orientation: "vertical"
143                 canvas.before:
144                     Color:
145                         rgba: .5, .5, .5, 1
146                     Line:
147                         width: 2
148                         rectangle: self.x, self.y, self.width, self.height
149             Label:
150                 text: "Deskripsi Hasil Pengacakkan"
151                 size_hint: (1, 0.2)
152                 text_size: self.size
153                 halign: "center"
154                 valign: "top"
155                 padding: (20, 20)
156
157             BoxLayout:
158                 id: randomization_result_description_layout
159                 orientation: "vertical"
160
161
162
163 <WarningPopup>
164     title: 'PERHATIAN!'
165     size_hint: (None, None)
166     size: (400, 200)
167     auto_dismiss: False
168
169     text_label: text_label
170     close_button: close_button
171
172     GridLayout:
173         cols: 1
174         Label:
175             markup: True
176             color: 1,1,1,1
177             id: text_label
178             text_size: self.width, None
179             size_hint: 1, None
180             halign: "center"
181         PopupCloseButton:
182             id: close_button
183             size_hint: (root.width, 0.5)
184             text: "Ok"
185             on_release: root.dismiss(); Window.set_system_cursor("arrow")
186
187
188 <LoadingPopup>
189     title: 'Random Projection Perturbation'
190     size_hint: (None, None)
191     size: (400, 200)
192     auto_dismiss: False
193
194     progress_bar: progress_bar
195     progress_label: progress_label
196     GridLayout:
197         cols: 1
198         Image:
199             source: 'view/assets/loading.gif'
200             anim_delay: 0.025
201             mipmap: True
202             allow_stretch: True
203         Label:
```

```
204         id: progress_label
205         color: 1,1,1,1
206         text: "0%"
207     ProgressBar:
208         id: progress_bar
209         max: 100
210
211 <DescriptionLabel>
212     size_hint: (1, 0.5)
213     text_size: self.size
214     halign: "left"
215     valign: "top"
216     padding: (20, 0)
217
218 <Label>:
219     markup: True
220     color: 0,0,0,1
221
222 <HoverButton>:
223     color: 1,1,1,1
224     background_color: 0,0,0,0 # the last zero is the critical one, make invisible
225     canvas.before:
226         Color:
227             rgba: (.4,.4,.4,1) if self.state=='normal' else (0,.7,.7,1) # visual feedback of press
228         RoundedRectangle:
229             pos: self.pos
230             size: self.size
231             radius: [5,]
232
233 <SpinnerOption>
234     color: 1,1,1,1
```

## LAMPIRAN B

### KODE PERANGKAT LUNAK PENGUJIAN

#### B.1 Pengujian Penambangan Data Klasifikasi

Kode program untuk pengujian penambangan data klasifikasi dibagi menjadi 2 bagian yaitu sebagai berikut.

##### B.1.1 *Random Rotation Perturbation*

Kode program untuk pengujian penambangan data klasifikasi terhadap teknik *Random Rotation Perturbation* dengan *dataset diabetes* adalah sebagai berikut.

Listing B.1: knn\_diabetes.py

```
1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4 import numpy as np
5
6 from sklearn.model_selection import train_test_split
7 from sklearn.neighbors import KNeighborsClassifier
8 from sklearn.metrics import confusion_matrix
9
10 import matplotlib.pyplot as plt
11 plt.style.use('ggplot')
12
13 import seaborn as sns
14
15 from time import time
16
17 # =====
18
19 df = pd.read_csv('rotated_diabetes.csv', delimiter = ',')
20
21 X = df.drop('Outcome', axis = 1).values
22 y = df['Outcome'].values
23
24 # =====
25
26 pd.options.display.max_columns = 10
27 print(df.describe())
28 print()
29
30 # -----
31
32 plt.title("Jumlah_Kasus_Diabetes")
33 sns.countplot(x = "Outcome", data = df)
34 plt.savefig('distribusi_label_diabetes.png', dpi=300)
35 plt.show()
36
37 # -----
38
39 g = sns.pairplot(df, hue = "Outcome", vars = df.columns[:-1])
40 g.fig.suptitle("Pairplot_Seluruh_Fitur_Dataset_Diacak", y=1.05, fontsize=75)
41 plt.savefig('distribusi_kolom_diabetes_diacak.png', dpi=300)
42 plt.show()
43
44 # -----
45
46 plt.title("Heatmap_Korelasi_Antar_Variabel_Dataset_Diacak")
47 ax = sns.heatmap(df.corr(), annot = True, cmap = 'RdYlGn')
48 ax.set_xticklabels(ax.get_xticklabels(), rotation = 23, ha="right")
49 ax.set_yticklabels(ax.get_yticklabels(), rotation = 60, ha="right")
50 plt.tight_layout()
51 plt.savefig('heatmap_diabetes_diacak.png', dpi=300)
52 plt.show()
53
54 # =====
55
56 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42, stratify = y)
57
58 # -----
```

```

59
60 neighbors = np.arange(1, 11)
61 train_accuracy = np.empty(len(neighbors))
62 test_accuracy = np.empty(len(neighbors))
63
64 for i, k in enumerate(neighbors):
65     knn = KNeighborsClassifier(n_neighbors = k)
66     knn.fit(X_train, y_train)
67
68     #Compute accuracy on the training set
69     train_accuracy[i] = knn.score(X_train, y_train)
70     #Compute accuracy on the test set
71     test_accuracy[i] = knn.score(X_test, y_test)
72
73 # -----
74
75 plt.title('Akurasi_Model_KNN_Dataset_Diacak')
76 plt.plot(neighbors, test_accuracy, label = 'Akurasi_Test_Set')
77 plt.plot(neighbors, train_accuracy, label = 'Akurasi_Training_Set')
78 plt.legend()
79 plt.xlabel('Jumlah_Tetangga_(k)')
80 plt.ylabel('Akurasi')
81 plt.savefig('plot_akurasi_diabetes_diacak.png', dpi=300)
82 plt.show()
83
84 # -----
85
86 print("Akurasi_setiap_K_pada_training_set_dataset_diacak:")
87 for i, accuracy in enumerate(train_accuracy):
88     print(str(i + 1) + ":" + str(accuracy))
89 print()
90
91 # -----
92
93 print("Akurasi_setiap_K_pada_test_set_dataset_diacak:")
94 for i, accuracy in enumerate(test_accuracy):
95     print(str(i + 1) + ":" + str(accuracy))
96 print()
97
98 # -----
99
100 highest_test_accuracy = test_accuracy.max()
101 k_highest_test_accuracy = np.argmax(test_accuracy) + 1
102
103 print("K_terbaik_adalah_" + str(k_highest_test_accuracy) + "_dengan_akurasi_test_set_sebesar_" + str(highest_test_accuracy))
104 print()
105
106 # =====
107
108 start_time = time()
109
110 knn = KNeighborsClassifier(n_neighbors = k_highest_test_accuracy)
111 knn.fit(X_train, y_train)
112
113 print("..._Waktu_yang_dibutuhkan_untuk_melatih_model_adalah_%s_detik_---" % (time() - start_time))
114 print()
115
116 print("Akurasi_pada_model_KNN_yang_digunakan:" + str(knn.score(X_test, y_test)))
117 print()
118
119 # -----
120
121 start_time = time()
122
123 y_pred = knn.predict(X_test)
124
125 print("..._Waktu_yang_dibutuhkan_untuk_melakukan_prediksi_adalah_%s_detik_---" % (time() - start_time))
126 print()
127
128 plt.title('Confusion_Matrix_pada_Test_Set_Dataset_Diacak')
129 sns.heatmap(confusion_matrix(y_pred, y_test), annot = True, annot_kws={"size": 16}, fmt='g')
130 plt.savefig('confusion_diabetes_diacak.png', dpi=300)
131 plt.show()
132
133 # =====

```

### B.1.2 Random Projection Perturbation

Kode program untuk pengujian penambangan data klasifikasi terhadap teknik *Random Rotation Perturbation* dengan *dataset mobile\_sensor* adalah sebagai berikut.

Listing B.2: knn\_mobile\_sensor.py

```

1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4 import numpy as np
5
6 from sklearn.neighbors import KNeighborsClassifier
7 from sklearn.preprocessing import LabelEncoder
8 from sklearn.metrics import confusion_matrix
9
10 import matplotlib.pyplot as plt
11 plt.style.use('ggplot')

```

```

12
13 import seaborn as sns
14
15 from time import time
16
17 # =====
18 df_train = pd.read_csv('projected_train.csv', delimiter = ',')
19 X_train = df_train.drop(['subject', 'Activity'], axis = 1).values
20 label_np = df_train['Activity'].values
21
22 # -----
23
24 label_np = label_np.ravel()
25 le = LabelEncoder()
26
27 y_train = le.fit_transform(label_np)
28
29 # =====
30
31 df_test = pd.read_csv('projected_test.csv', delimiter = ',')
32 X_test = df_test.drop(['subject', 'Activity'], axis = 1).values
33 label_np_test = df_test['Activity'].values
34
35 # -----
36 label_np_test = label_np_test.ravel()
37 y_test = le.fit_transform(label_np_test)
38
39 # =====
40 df_train_test = df_train.append(df_test)
41
42 # -----
43
44 pd.options.display.max_columns = 5
45 print(df_train_test.drop(['subject'], axis = 1).describe())
46 print()
47
48 # -----
49 plt.title("Distribusi_Label_(Aktivitas)")
50 ax = sns.countplot(x = "Activity", data = df_train_test)
51 ax.set_xticklabels(ax.get_xticklabels(), rotation = 20, ha="right")
52 plt.tight_layout()
53 plt.savefig('distribusi_label_mobile_sensor_diacak.png', dpi=300)
54 plt.show()
55
56 # -----
57 neighbors = np.arange(1,31)
58 train_accuracy = np.empty(len(neighbors))
59 test_accuracy = np.empty(len(neighbors))
60
61 for i,k in enumerate(neighbors):
62     knn = KNeighborsClassifier(n_neighbors = k)
63     knn.fit(X_train, y_train)
64
65     #Compute accuracy on the training set
66     train_accuracy[i] = knn.score(X_train, y_train)
67     #Compute accuracy on the test set
68     test_accuracy[i] = knn.score(X_test, y_test)
69
70 # -----
71
72 plt.title('Akurasi_Model_KNN_Dataset_Aslি')
73 plt.plot(neighbors, test_accuracy, label = 'Akurasi_Test_Set')
74 plt.plot(neighbors, train_accuracy, label = 'Akurasi_Training_Set')
75 plt.legend()
76 plt.xlabel('Jumlah_Tetangga_(K)')
77 plt.ylabel('Akurasi')
78 plt.tight_layout()
79 plt.savefig('akurasi_mobile_sensor_diacak.png', dpi=300)
80 plt.show()
81
82 # -----
83
84 print("Akurasi_setiap_K_pada_training_set_dataset_diacak:")
85 for i, accuracy in enumerate(train_accuracy):
86     print(str(i + 1) + ":" + str(accuracy))
87 print()
88
89 # -----
90
91 print("Akurasi_setiap_K_pada_test_set_dataset_diacak:")
92 for i, accuracy in enumerate(test_accuracy):
93     print(str(i + 1) + ":" + str(accuracy))
94 print()
95
96 # -----
97
98 print("Akurasi_setiap_K_pada_test_set_dataset_diacak:")
99 for i, accuracy in enumerate(test_accuracy):
100    print(str(i + 1) + ":" + str(accuracy))
101 print()
102
103 # -----
104
105 highest_test_accuracy = test_accuracy.max()
106 k_highest_test_accuracy = np.argmax(test_accuracy) + 1
107
108 print("K_terbaik_adalah_" + str(k_highest_test_accuracy) + "_dengan_akurasi_test_set_sebesar_" + str(highest_test_accuracy))
109 print()
110

```

```

111 # =====
112 start_time = time()
113
114 knn = KNeighborsClassifier(n_neighbors = k_highest_test_accuracy)
115 knn.fit(X_train, y_train)
116
117 print("----Waktu_yang_dibutuhkan_untuk_melatih_model_adalah_%s_detik----" % (time() - start_time))
118 print()
119
120 print("Akurasi_pada_model_KNN_yang_digunakan:" + str(knn.score(X_test, y_test)))
121 print()
122
123 #
124 # -----
125 start_time = time()
126
127 y_pred = knn.predict(X_test)
128
129 print("----Waktu_yang_dibutuhkan_untuk_melakukan_prediksi_adalah_%s_detik----" % (time() - start_time))
130 print()
131
132 plt.title('Dataset_Diacak')
133 sns.heatmap(confusion_matrix(y_pred, y_test), annot = True, annot_kws={"size": 16}, fmt='g')
134 plt.tight_layout()
135 plt.savefig('confusion_matrix_mobile_sensor_diacak.png', dpi=300)
136 plt.show()
137
138 #
139 # =====

```

## B.2 Pengujian Penambangan Data *Clustering*

Kode program untuk pengujian penambangan data *clustering* dibagi menjadi 2 bagian yaitu sebagai berikut.

### B.2.1 *Random Rotation Perturbation*

Kode program untuk pengujian penambangan data *clustering* terhadap teknik *Random Rotation Perturbation* dengan *dataset mall\_customers* adalah sebagai berikut.

Listing B.3: kmeans\_mall.py

```

1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4 import numpy as np
5
6 from sklearn.cluster import KMeans
7 from sklearn.metrics import silhouette_score
8
9 import matplotlib.pyplot as plt
10 plt.style.use('ggplot')
11
12 import plotly.graph_objs as go
13 from plotly.offline import plot
14
15 import seaborn as sns
16
17 from time import time
18
19 # =====
20
21 df = pd.read_csv('rotated_Mall_Customers.csv', delimiter = ',')
22 X = df[['Age', 'Annual_Income_(k$)', 'Spending_Score_(1-100)']].values
23
24 #
25
26 pd.options.display.max_columns = 10
27 print(df.describe())
28 print()
29
30 #
31
32 sns.pairplot(df, vars = df.columns[2:])
33 plt.tight_layout()
34 plt.savefig('pairplot_mall_customers_diacak.png', dpi=300)
35 plt.show()
36
37 #
38
39 plt.title("Heatmap_Korelasi_Antar_Variabel_Dataset_Diacak")
40 ax = sns.heatmap(df.corr(), annot = True, cmap = 'RdYlGn')
41 ax.set_xticklabels(ax.get_xticklabels(), rotation = 40, ha="right")
42 ax.set_yticklabels(ax.get_yticklabels(), rotation = 68, ha="right")
43 plt.tight_layout()
44 plt.savefig('heatmap_mall_customers_diacak.png', dpi=300)
45 plt.show()
46

```

```

47 # =====
48 kmax = 10
49
50 inertia = []
51 sil = []
52
53 for k in range(2, kmax + 1):
54     kmeans = KMeans(n_clusters = k).fit(X)
55     labels = kmeans.labels_
56
57     inertia.append(kmeans.inertia_)
58     sil.append(silhouette_score(X, labels, metric = 'euclidean'))
59
60 # -----
61
62 plt.title("Metode_Elbow_Dataset_Diacak")
63 plt.plot(range(2, kmax + 1), inertia, marker='o')
64 plt.xlabel('Jumlah_Cluster_(k)')
65 plt.ylabel('Sum_of_Squared_Error')
66 plt.tight_layout()
67 plt.savefig('elbow_mall_customers_diacak.png', dpi=300)
68 plt.show()
69
70 #
71 #
72 plt.title("Sillhoutte_Score_Dataset_Diacak")
73 plt.plot(range(2, kmax + 1), sil, marker='o')
74 plt.xlabel('Jumlah_Cluster_(k)')
75 plt.ylabel('Sillhoutte_Score')
76 plt.tight_layout()
77 plt.savefig('siluet_mall_customers_diacak.png', dpi=300)
78 plt.show()
79
80 #
81 #
82 print("Sillhoutte_Score_setiap_K:")
83 for i, score in enumerate(sil):
84     print(str(i + 2) + ":" + str(score))
85 print()
86
87 #
88 highest_sil = max(sil)
89 k_highest_sil = np.argmax(sil) + 2
90
91 print("K_terbaik_adalah_" + str(k_highest_sil) + "_dengan_Sillhoutte_Score_sebesar_" + str(highest_sil))
92 print()
93
94 # =====
95
96 start_time = time()
97
98 kmeans = KMeans(n_clusters = k_highest_sil).fit(X)
99
100 print("---_Waktu_yang_dibutuhkan_untuk_melatih_model_adalah_%s_detik---" % (time() - start_time))
101 print()
102
103 labels = kmeans.labels_
104 centroids = kmeans.cluster_centers_
105
106 #
107 df['label'] = labels
108
109 trace1 = go.Scatter3d(
110     x = df['Age'],
111     y = df['Spending_Score_(1-100)'],
112     z = df['Annual_Income_(k$)'],
113
114     mode = 'markers',
115
116     marker = dict(
117         color = df['label'],
118         size = 15,
119         line = dict(
120             color = df['label'],
121             width = 12
122         ),
123         opacity = 0.8
124     )
125 )
126
127 )
128
129 data = [trace1]
130
131 layout = go.Layout(
132     title= 'Clusters',
133     scene = dict(
134         xaxis = dict(title = 'Age'),
135         yaxis = dict(title = 'Spending_Score'),
136         zaxis = dict(title = 'Annual_Income')
137     )
138 )
139
140 fig = go.Figure(data=data, layout=layout)
141 plot(fig)
142
143 # =====

```

### B.2.2 Random Projection Perturbation

Kode program untuk pengujian penambangan data *clustering* terhadap teknik *Random Projection Perturbation* dengan *dataset mobile\_sensor* adalah sebagai berikut.

Listing B.4: kmeans\_mobile\_sensor.py

```

1 # -*- coding: utf-8 -*-
2
3 import pandas as pd
4 import numpy as np
5
6 from sklearn.cluster import KMeans
7 from sklearn.metrics import silhouette_score
8
9 import matplotlib.pyplot as plt
10 plt.style.use('ggplot')
11
12 import seaborn as sns
13
14 from time import time
15
16 # =====
17
18 df = pd.read_csv('projected_train.csv', delimiter = ',')
19 X = df.drop(['subject', 'Activity'], axis = 1).values
20
21 # -----
22 start_time = time()
23
24 from sklearn.decomposition import PCA
25 pca = PCA(n_components = 2)
26 X = pca.fit_transform(X)
27
28 print("--- Waktu yang dibutuhkan untuk PCA adalah %s detik ---" % (time() - start_time))
29 print()
30
31 # =====
32
33 print(df.describe())
34 print()
35
36 # -----
37
38 plt.title("Activities_Count")
39 ax = sns.countplot(x = "Activity", data = df)
40 ax.set_xticklabels(ax.get_xticklabels(), rotation = 20, ha="right")
41 plt.show()
42
43 #
44
45 df_pca = pd.DataFrame(X, columns = ['PCA_1', 'PCA_2']).join(df[["Activity"]])
46
47 plt.title("Dataset_Diacak")
48 sns.scatterplot(x = 'PCA_1', y = 'PCA_2', hue = "Activity", data = df_pca)
49 plt.legend(bbox_to_anchor=(1, 1))
50 plt.tight_layout()
51 plt.savefig('scatter_mobile_sensor_diacak.png', dpi=300)
52 plt.show()
53
54 # =====
55
56 kmax = 10
57
58 inertia = []
59 sil = []
60
61 for k in range(2, kmax + 1):
62     kmeans = KMeans(n_clusters = k).fit(X)
63     labels = kmeans.labels_
64
65     inertia.append(kmeans.inertia_)
66     sil.append(silhouette_score(X, labels, metric = 'euclidean'))
67
68 #
69
70 plt.title("Metode_Elbow_Dataset_Diacak")
71 plt.plot(range(2, kmax + 1), inertia, marker='o')
72 plt.xlabel('Jumlah_Cluster_(k)')
73 plt.ylabel('Distortion')
74 plt.tight_layout()
75 plt.savefig('elbow_mobile_sensor_diacak.png', dpi=300)
76 plt.show()
77
78 #
79
80 plt.title("Sillhoutte_Score_Dataset_Diacak")
81 plt.plot(range(2, kmax + 1), sil, marker='o')
82 plt.xlabel('Jumlah_Cluster_(k)')
83 plt.ylabel('Sillhoutte_Score')
84 plt.tight_layout()
85 plt.savefig('siluet_mobile_sensor_diacak.png', dpi=300)
86 plt.show()
87
88 #
89

```

```

91 print("Sillhouette_Score_setiap_K:")
92 for i, score in enumerate(sil):
93     print(str(i + 2) + ":" + str(score))
94 print()
95 #
96 # -----
97
98 highest_sil = max(sil)
99 k_highest_sil = np.argmax(sil) + 2
100 print("K_terbaik_adalah_" + str(k_highest_sil) + "_dengan_Sillhouette_Score_sebesar_" + str(highest_sil))
101 print()
102
103 # =====
104 start_time = time()
105
106 kmeans = KMeans(n_clusters = k_highest_sil).fit(X)
107
108 print("Waktu_yang_dibutuhkan_untuk_melatih_model_adalah_%s_detik---" % (time() - start_time))
109 print()
110
111 labels = kmeans.labels_
112 centroids = kmeans.cluster_centers_
113
114 # -----
115
116 df_pca['Cluster'] = labels
117 df_pca['Cluster'] = df_pca['Cluster'].map({0: 'Bergerak', 1: 'Diam'})
118
119 plt.title("Hasil_Cluster_Dataset_Diacak")
120 sns.scatterplot(x = "PCA_1", y = "PCA_2", hue = "Cluster", data = df_pca)
121 plt.scatter(
122     centroids[:, 0],
123     centroids[:, 1],
124     s=250,
125     marker='*',
126     c='red',
127     edgecolor='black'
128 )
129 plt.tight_layout()
130 plt.savefig('cluster_mobile_sensor_diacak.png', dpi=300)
131 plt.show()
132
133 # =====
134
135 # =====

```

## B.3 Pengujian Lainnya

Kode program untuk pengujian lainnya seperti pengujian jarak Euclidean dan *Adjusted Rand Index* adalah sebagai berikut.

Listing B.5: pengujian.py

```

1 # -*- coding: utf-8 -*-
2
3 # Dua baris dibawah adalah kode untuk menghitung ADJUSTED RAND INDEX
4 # variabel labels_asli dan labels_proyeksi perlu diambil dari hasil
5 # clustering yang sudah dilakukan. Tips: dapat menggunakan fitur copy/paste
6 # variabel pada perangkat lunak Spyder di tab Variable explorer
7
8 # from sklearn.metrics import adjusted_rand_score
9 # adjusted_rand_score(labels_asli, labels_proyeksi)
10
11
12 # Berikut adalah kode untuk menguji hasil Random Projection Perturbation
13 # apakah berada pada rentang jarak Euclidean yang benar
14 import pandas as pd
15
16 from scipy.spatial import distance
17 import sys
18
19 df_asli = pd.read_csv('train.csv', delimiter = ',')
20 df_projected = pd.read_csv('projected_train.csv', delimiter = ',')
21 asli = df_asli.drop(['subject', 'Activity'], axis = 1).values
22 projected = df_projected.drop(['subject', 'Activity'], axis = 1).values
23
24 eps = 0.52
25 for i in range(0, asli.shape[0]):
26     for j in range(i, asli.shape[0]):
27         d_asli = distance.euclidean(asli[i], asli[j])
28         d_projected = distance.euclidean(projected[i], projected[j])
29         if (1-eps) * d_asli**2 > d_projected**2 > (1+eps) * d_asli**2:
30             print("-----ERROR:" + str(i+1) + ":" + str(j+1) + ":" +
31                   str(d_asli) + " " + str(d_projected) + "-----")
32             sys.exit()
33         else:
34             print(str(i+1) + ":" + str(j+1) + ":" + str(d_asli) + " " + str(d_projected))
35
36 print("ok")
37
38
39 # Berikut adalah kode untuk menguji hasil Random Rotation Perturbation

```

```
41 # apakah mempunyai jarak Euclidean yang sama persis dengan aslinya
42 df_asli = pd.read_csv('Mall_Customers.csv', delimiter = ',')
43 asli = df_asli[['Age' , 'Annual_Income_(k$)' , 'Spending_Score_(1-100)']].values
44 df_rotated = pd.read_csv('rotated_Mall_Customers.csv', delimiter = ',')
45 rotated = df_rotated[['Age' , 'Annual_Income_(k$)' , 'Spending_Score_(1-100)']].values
46
47 for i in range(0, asli.shape[0]):
48     for j in range(i, asli.shape[0]):
49         d_asli = distance.euclidean(asli[i], asli[j])
50         d_rotated = distance.euclidean(rotated[i], rotated[j])
51         if round(d_asli, 2) != round(d_rotated, 2) and round(d_asli, 3) != round(d_rotated, 3):
52             print("-----ERROR: " + str(i+1) + ":" + str(j+1) + ":" + 
53                   str(d_asli) + "," + str(d_rotated) + "-----")
54             sys.exit()
55         else:
56             print(str(i+1) + ":" + str(j+1) + ":" + str(d_asli) + "," + str(d_rotated))
57
58 print("ok")
```