

# BAB 1

## IMPLEMENTASI DAN PENGUJIAN

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

### 1.1 Implementasi Antarmuka

Antarmuka perangkat lunak diimplementasikan dengan memakai *framework* antarmuka grafis berbasis bahasa pemrograman Python yang bernama Kivy. Implementasi antarmuka disesuaikan dengan rancangan antarmuka perangkat lunak yang telah dibuat pada bab ???. Gambar 1.1 adalah tampilan antarmuka dari implementasi perangkat lunak.

Antarmuka perangkat lunak mempunyai tiga buah bagian yang mempunyai fungsinya masing-masing. Ketiga bagian ini dapat dilihat pada Gambar 1.2 Pertama adalah bagian masukan dan pengaturan, terdapat pada bagian atas yang bernomor satu dan dikelilingi kotak merah. Kedua adalah bagian deskripsi *dataset*, terdapat pada bagian bawah sebelah kiri yang bernomor dua dan dikelilingi kotak biru. Terakhir adalah bagian deskripsi hasil pengacakan, terdapat pada bagian bawah sebelah kanan yang bernomor tiga dan dikelilingi kotak hijau. Ketiga bagian ini akan dijelaskan secara rinci pada subbab-subbab berikutnya.

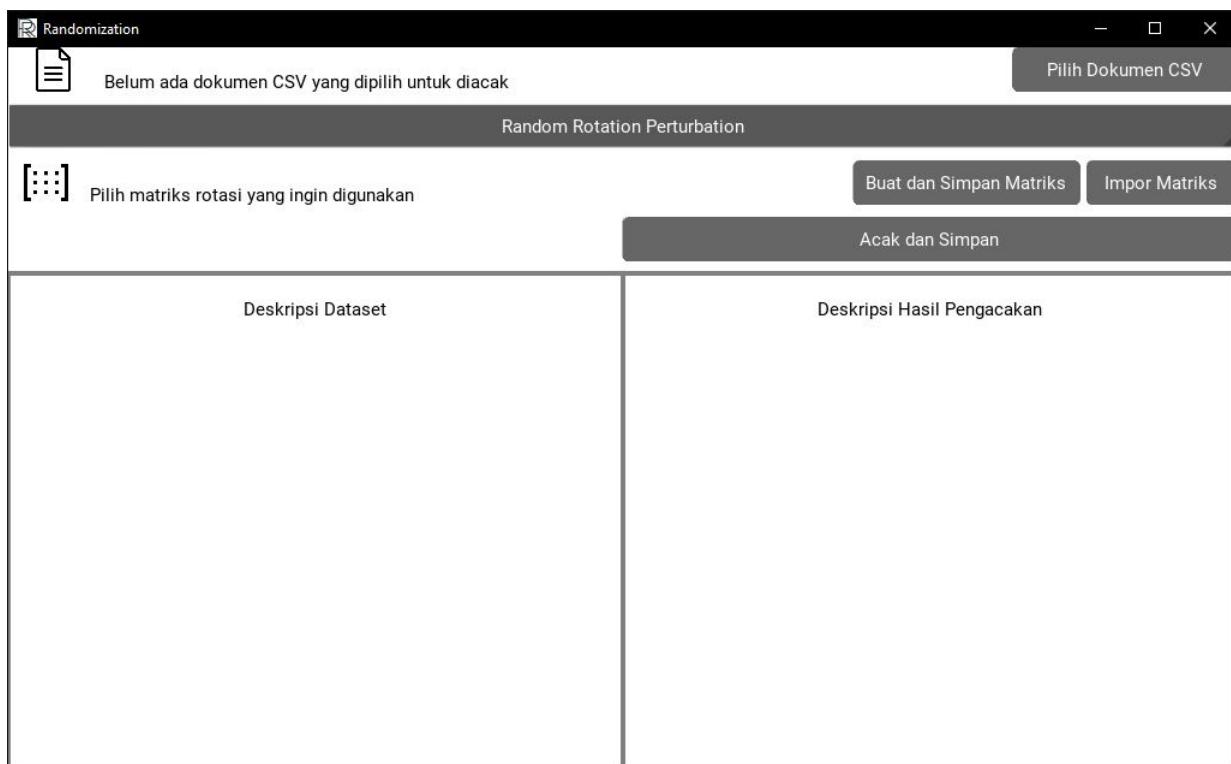
Perangkat lunak *Randomization* mengimplementasikan dua buah teknik *Randomization* yang berbeda yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation*. Oleh karena itu, antarmuka perangkat lunak akan menyesuaikan dengan teknik yang dipilih oleh pengguna. Ketiga bagian antarmuka yang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

#### 1.1.1 Masukan dan Pengaturan

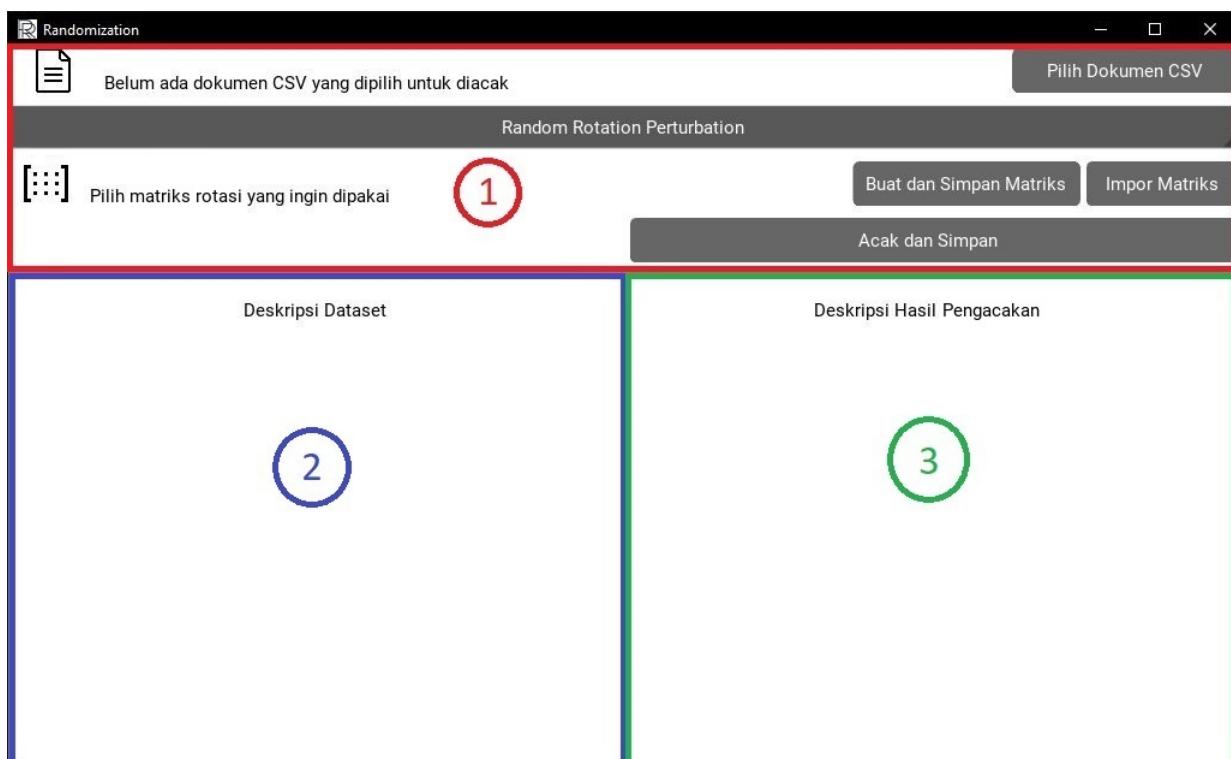
Bagian masukan dan pengaturan menyediakan berbagai interaksi untuk pengguna dapat mengatur masukan yang perlu diberikan kepada perangkat lunak dan menerapkan teknik *Randomization* yang diinginkan. Ada beberapa fungsi inti pada bagian ini yaitu sebagai berikut.

- Masukan *dataset* berupa file *comma-separated values* yang ingin diacak.
- Memilih teknik *Randomization* yang ingin digunakan.
- Membuat baru dan memilih matriks rotasi atau proyeksi yang ingin digunakan.
- Masukan nilai variabel *epsilon* dan nilai variabel *k* untuk teknik *Random Projection Perturbation*.
- Sebuah tombol untuk menerapkan teknik *Randomization* dan menyimpan hasilnya.

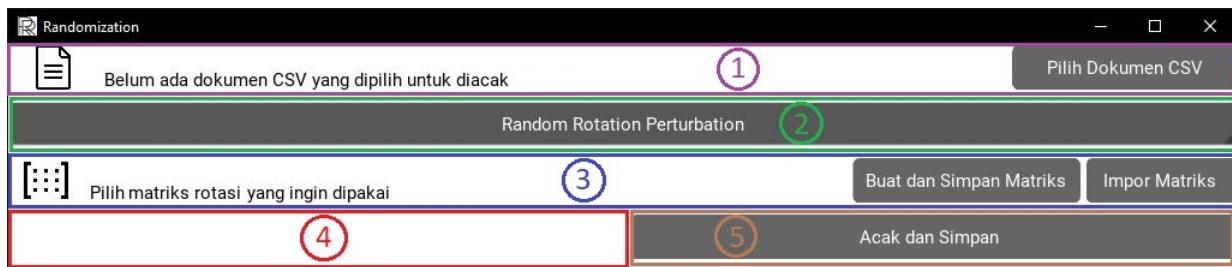
Berikut akan dijelaskan secara rinci dengan gambar setiap fungsi tersebut yang dapat dilihat pada Gambar 1.3 dan cara pemakaianya yang benar secara berturut.



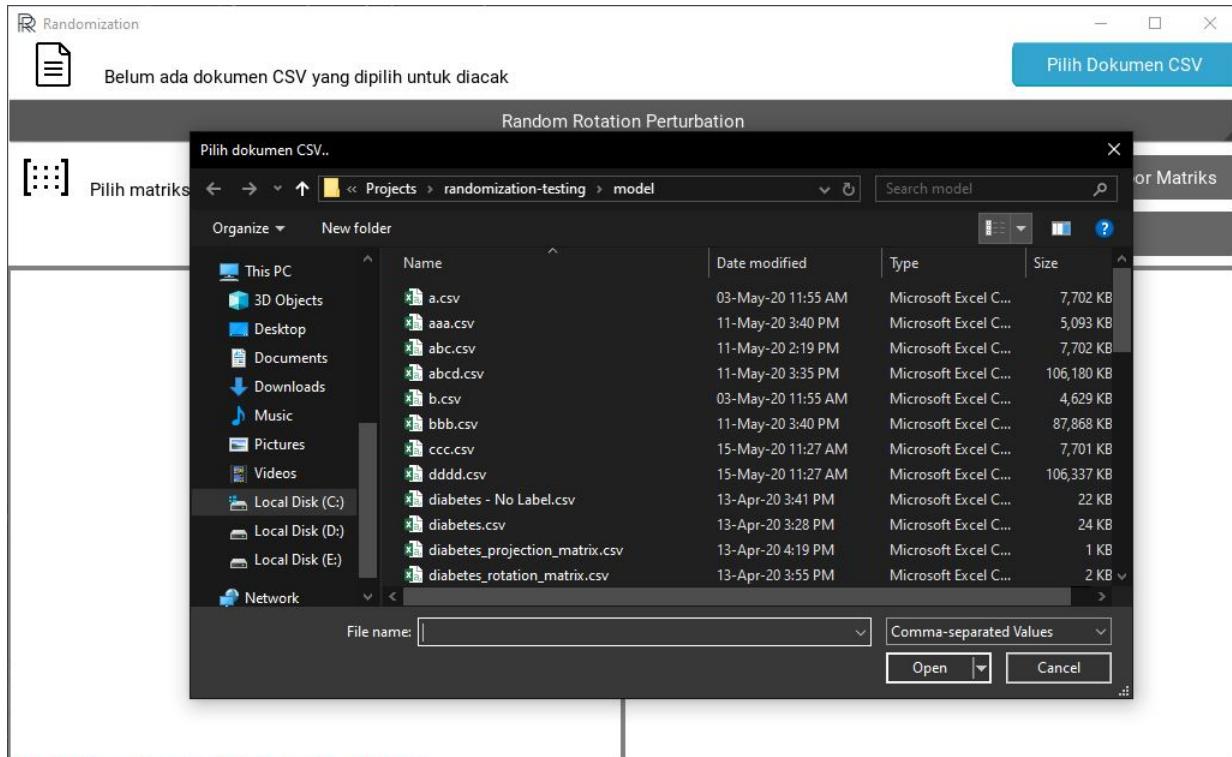
Gambar 1.1: Tampilan perangkat lunak yang pertama ditampilkan saat perangkat lunak baru dibuka



Gambar 1.2: Bagian-bagian pada antarmuka perangkat lunak



Gambar 1.3: Bagian antarmuka masukan dan pengaturan perangkat lunak

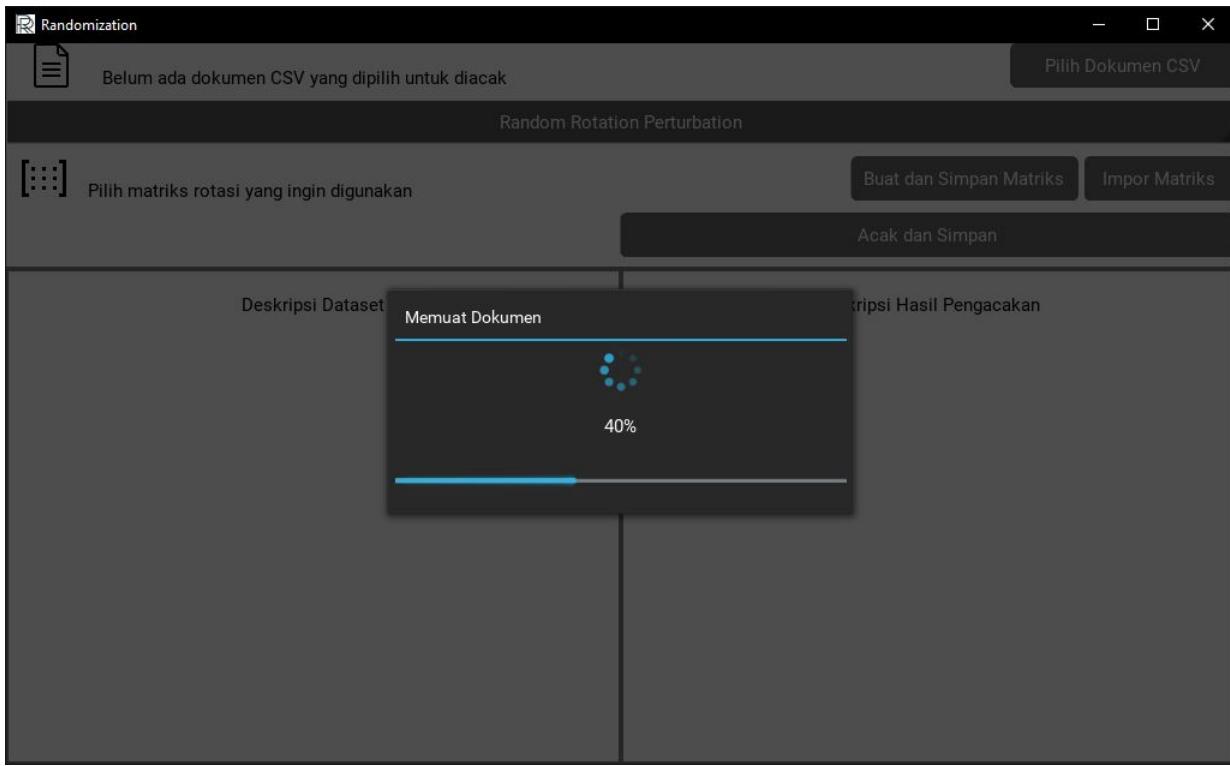


Gambar 1.4: Jendela untuk memilih *dataset* yang berupa dokumen CSV

### 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 1.3 yang terdapat pada bagian yang dikelilingi kotak berwarna merah dan bernomor 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 1.4.

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.



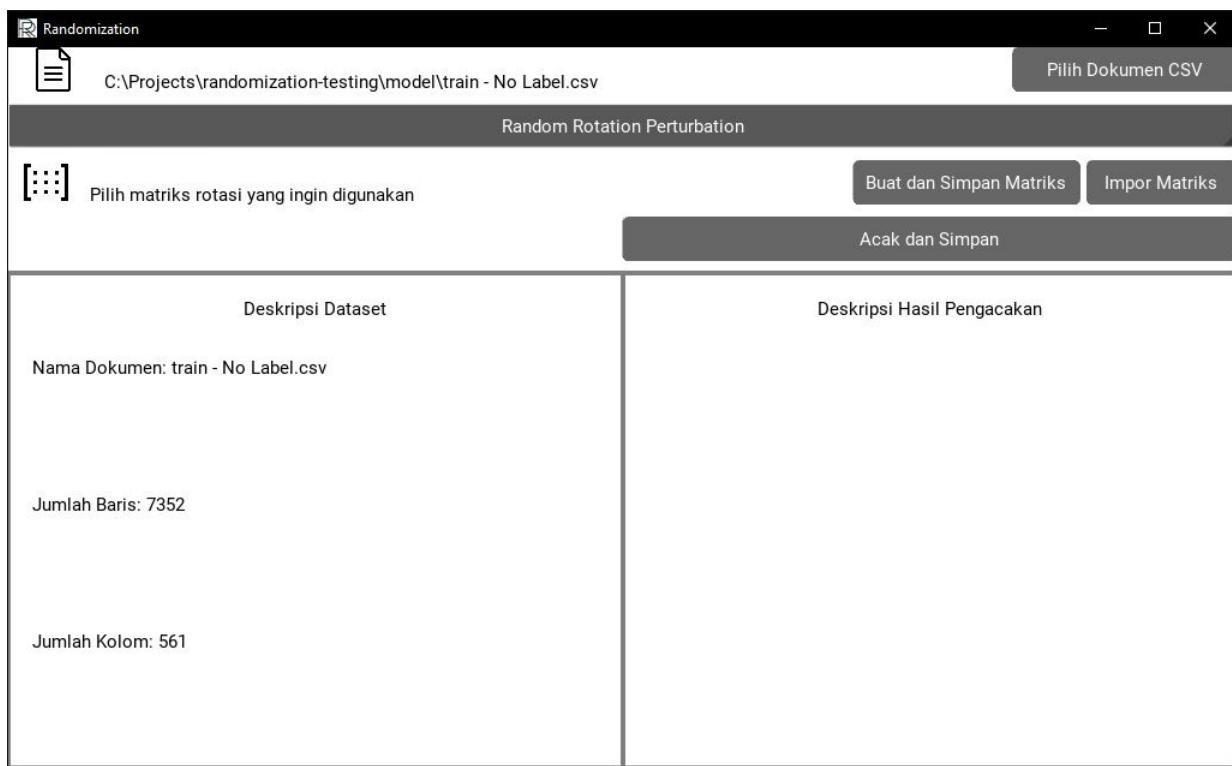
Gambar 1.5: Tampilan *popup* yang ditampilkan saat proses memuat dokumen berlangsung

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 1.5. 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 dijelaskan pada subbab berikutnya. Tampilan antarmuka setelah pengguna memilih dokumen dapat dilihat pada Gambar 1.6

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

### Pemilihan teknik *Randomization*

Setelah pengguna memilih *dataset* yang ingin diacak, pengguna juga harus memilih teknik *Randomization* apa yang ingin diterapkan terhadap *dataset* yang sudah dipilih. Pada awal perangkat lunak dibuka, secara otomatis teknik *Random Rotation Perturbation* yang dipilih. Apabila pengguna ingin mengganti teknik yang ingin diterapkan pada *dataset*, pengguna dapat menekan tombol *dropdown* yang bertuliskan nama teknik *Randomization*. Tombol ini dapat dilihat pada Gambar 1.3 yang



Gambar 1.6: Tampilan antarmuka setelah sebuah dokumen dipilih

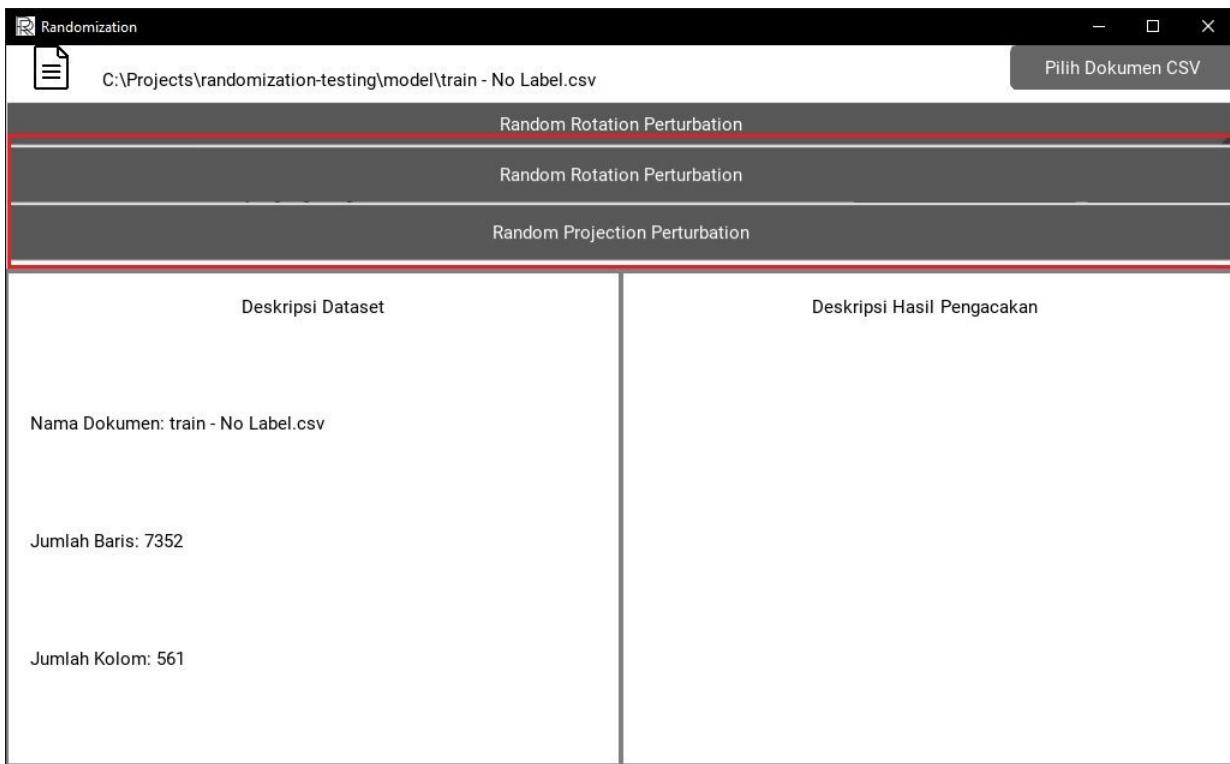
dikelilingi kotak berwarna hijau dan bernomor dua.

Apabila pengguna menekan tombol ini maka perangkat lunak akan menampilkan *dropdown* yang mempunyai dua buah opsi teknik *Randomization* yaitu “Random Rotation Perturbation” dan “Random Projection Perturbation”. Antarmuka tersebut dapat dilihat pada Gambar 1.7 yang dikelilingi oleh kotak merah. Pemilihan teknik ini juga akan memicu beberapa perubahan pada tampilan antarmuka perangkat lunak menyesuaikan dengan teknik yang dipilih. Beberapa perubahan pada perangkat lunak tersebut melengkapi bagian pembuatan dan pemilihan matriks, parameter teknik *Randomization*, dan bagian pengacakan dan simpan yang akan dijelaskan setiap perubahan tersebut pada subbab berikutnya.

### Pembuatan dan Pemilihan Matriks

Setelah pengguna memilih teknik yang ingin diterapkan, pengguna harus memilih matriks yang diinginkan atau membuat baru. Matriks yang dimaksudkan adalah matriks rotasi atau matriks proyeksi sesuai teknik *Randomization* yang dipilih. Apabila teknik *Random Rotation Perturbation* yang dipilih maka perangkat lunak akan mengubah fungsi pembuatan dan pemilihan matriks ini menjadi matriks rotasi. Apabila teknik *Random Projection Perturbation* yang dipilih maka perangkat lunak akan mengubah fungsi pembuatan dan pemilihan matriks ini menjadi matriks proyeksi. Perubahan ini dapat terlihat pada label yang berada di sebelah kanan simbol matriks apabila belum memilih atau membuat matriks maka label tersebut akan menampilkan kalimat “Pilih matriks rotasi yang ingin digunakan” atau “Pilih matriks proyeksi yang ingin digunakan”. Bagian ini dapat dilihat pada Gambar 1.3 yang dikelilingi oleh kotak berwarna hijau dan bernomor dua.

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



Gambar 1.7: Tampilan antarmuka saat pengguna memilih teknik

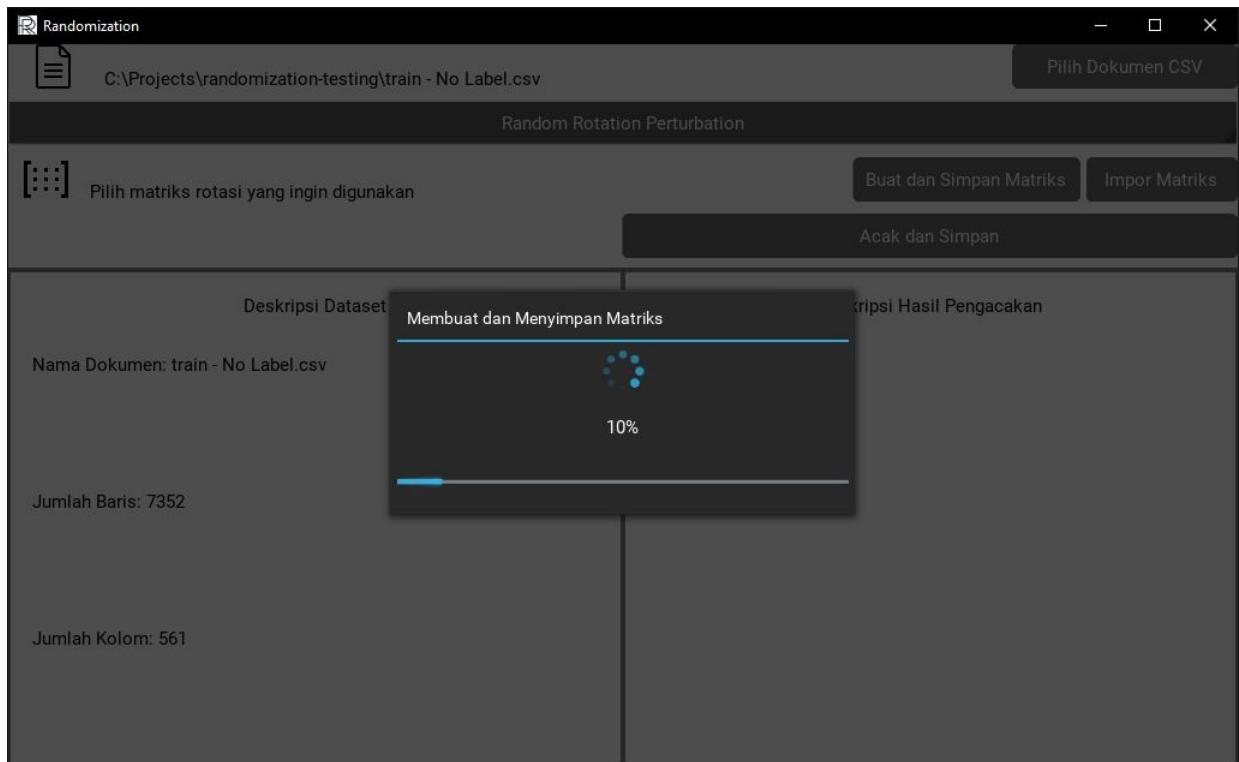
pengguna. Pada saat perangkat lunak sedang memproses matriks tersebut, perangkat lunak akan menampilkan *popup* memuat yang dapat dilihat pada Gambar 1.8. *Popup* ini juga akan tampil saat proses impor matriks dilakukan. Hasil matriks yang dibuat oleh perangkat lunak dapat digunakan kembali untuk lain kali sehingga rotasi atau proyeksi yang diterapkan akan sama dengan yang sebelumnya.

Pengguna dapat melakukan impor matriks dengan cara menekan tombol “Impor Matriks” untuk memilih matriks rotasi atau proyeksi yang diinginkan untuk diterapkan pada *dataset*. Matriks yang dipilih harus sesuai dengan *dataset* yang ingin diacak, misalnya apabila matriks rotasi yang dipilih memiliki dimensi yang berbeda dengan *dataset* maka perangkat lunak akan melarang impor matriks dilakukan karena pengacakan tidak dapat dilakukan. Perangkat lunak akan menampilkan *popup* peringatan untuk pengguna yang dapat dilihat pada Gambar 1.9. Apabila pengguna memilih teknik *Random Projection Perturbation* dan pengguna mengimpor matriks proyeksi maka parameter variabel *k* akan terisi secara otomatis sesuai dengan matriks proyeksi yang diimporkan.

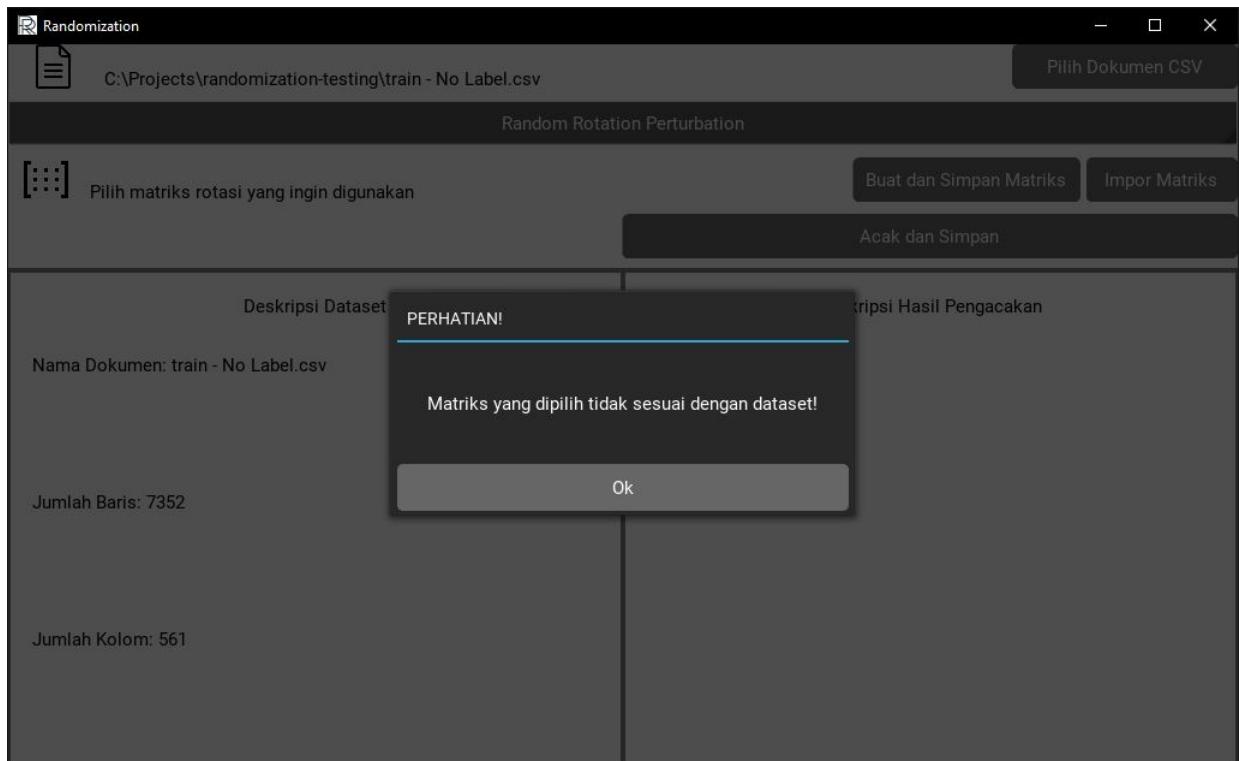
Apabila pengguna belum memilih *dataset* yang ingin diacak, pengguna tidak dapat membuat maupun impor matriks terlebih dahulu. Hal ini dikarenakan perlu ada proses pengecekan terlebih dahulu yang dilakukan perangkat lunak untuk memastikan *dataset* yang ingin diacak sudah sesuai persyaratan dan sesuai dengan matriks yang akan dipilih. Perangkat lunak akan melarang pengguna membuat maupun impor matriks dengan menampilkan sebuah *popup* peringatan yang dapat dilihat pada Gambar 1.10. Pada teknik *Random Projection Perturbation*, pengguna baru bisa membuat matriks proyeksi apabila sudah memenuhi persyaratan yang diminta yaitu mengisi parameter teknik tersebut yang mana adalah variabel *epsilon* dan variabel *k*.

### Parameter teknik *Randomization*

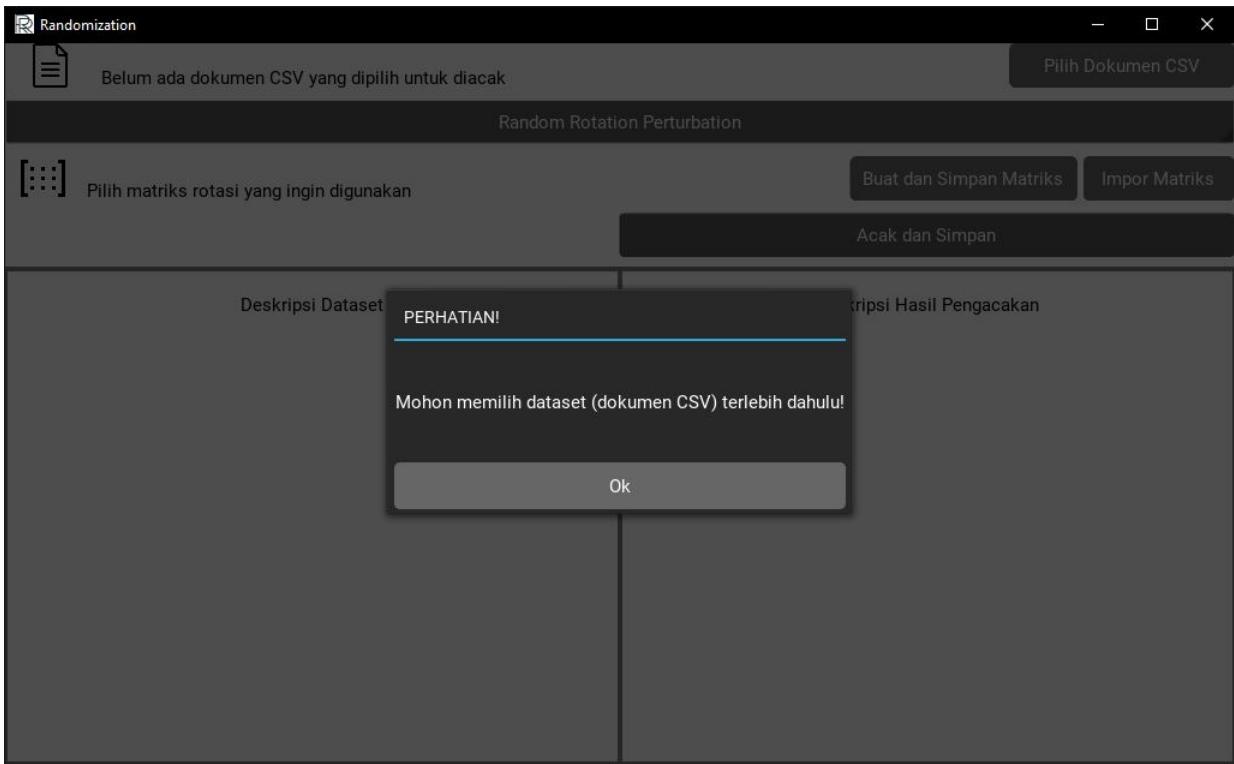
Perangkat lunak hanya meminta kepada pengguna parameter untuk teknik *Random Projection Perturbation* saja apabila pengguna memilih teknik tersebut. Pada teknik *Random Rotation Perturbation* tidak ada parameter yang perlu pengguna berikan. Ada dua buah parameter yang



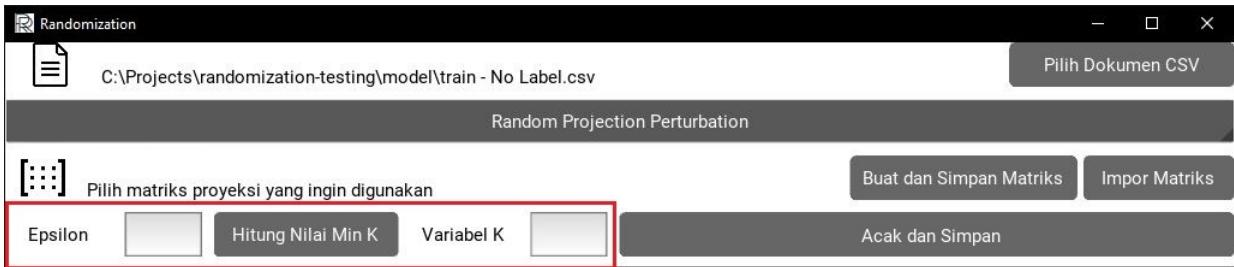
Gambar 1.8: Tampilan antarmuka saat perangkat lunak membuat dan menyimpan matriks



Gambar 1.9: Tampilan *popup* yang ditampilkan apabila matriks yang ingin diimpor tidak sesuai dengan *dataset*



Gambar 1.10: Tampilan *popup* peringatan apabila pengguna belum memilih *dataset* yang diinginkan untuk diacak



Gambar 1.11: Tampilan antarmuka parameter teknik *Random Projection Perturbation*

perlu pengguna berikan yaitu variabel *epsilon* dan variabel *k*. Pengguna dapat memasukkan nilai kedua variabel tersebut dengan menekan kolom variabel tersebut masing-masing. Kedua buah parameter tersebut dapat dilihat antarmukanya pada Gambar 1.11

Seperi yang disinggung pada subbab sebelumnya antarmuka perangkat lunak akan menyesuaikan secara otomatis sesuai teknik yang dipilih pengguna. Pada bagian parameter teknik *Randomization*, perangkat lunak akan menyembunyikan antarmuka parameter *Random Projection Perturbation* apabila pengguna memilih teknik *Random Rotation Perturbation*. Antarmuka tersebut dapat dilihat pada Gambar 1.3 yang dikelilingi oleh kotak berwarna merah dan bernomor empat, dapat dilihat tidak ada parameter apapun yang tampil apabila teknik *Random Rotation Perturbation* yang dipilih.

Selain dua buah parameter, pada bagian ini juga ada sebuah tombol yaitu ‘‘Hitung Nilai Min K’’ yang memiliki fungsi untuk menghitung nilai minimal variabel *k* yang pengguna berikan. Pada teknik *Random Projection Perturbation*, ada beberapa persyaratan yang harus dipenuhi oleh pengguna dan salah satunya adalah variabel *k* yang diberikan harus melebihi sebuah nilai minimal yang dihitung berdasarkan ukuran *dataset* dan nilai variabel *epsilon*. Oleh karena itu, sebelum tombol ini dapat berfungsi, pengguna harus memilih terlebih dahulu *dataset* yang ingin diacak dan memberikan masukan nilai variabel *epsilon* yang sesuai dengan persyaratan variabel *epsilon*.



Gambar 1.12: Tampilan *popup* peringatan tombol “Hitung Nilai Min K”

yaitu nilainya lebih besar dari 0 dan kurang dari 1. Apabila pengguna belum memenuhi kedua persyaratan tersebut, tombol tidak akan berfungsi dan perangkat lunak akan menampilkan *popup* peringatan yang dapat dilihat pada Gambar 1.12. Nilai minimal variabel  $k$  akan ditampilkan pada bagian antarmuka deskripsi *dataset* yang akan dijelaskan pada subbab berikutnya.

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

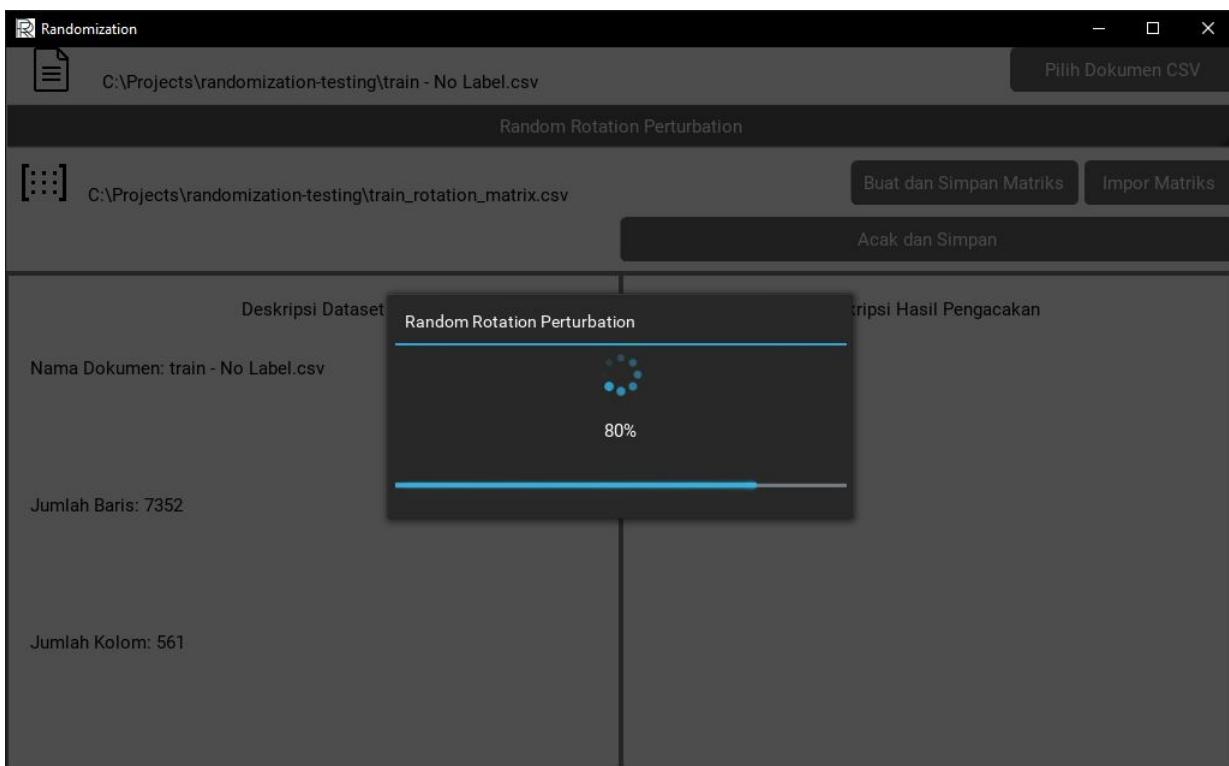
### Pengacakan dan Simpan

Setelah pengguna memberikan masukan yang sesuai dan mengatur pengaturan yang diinginkan maka pengguna telah dapat melakukan pengacakan dengan menekan tombol “Acak dan Simpan”. Tombol ini akan menerapkan teknik *Randomization* yang dipilih oleh pengguna terhadap *dataset* yang ingin diacak menggunakan matriks yang telah dibuat atau dipilih oleh pengguna dan parameter-parameter yang pengguna berikan. Tampilan antarmuka saat proses pengacakan dilakukan perangkat lunak dapat dilihat pada Gambar 1.13.

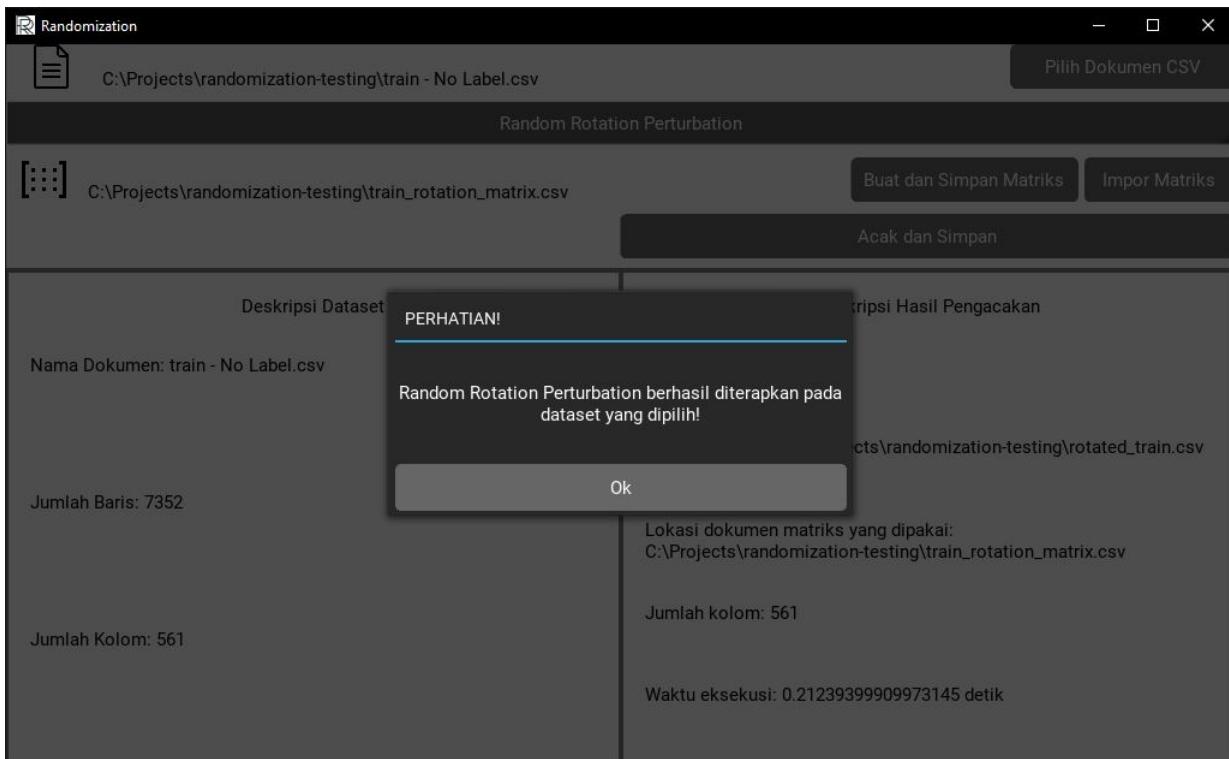
Setelah perangkat lunak berhasil melakukan pengacakan, perangkat lunak akan meminta pengguna untuk memilih direktori tempat penyimpanan dan nama dokumen hasil pengacakan. Perangkat lunak akan menyimpan hasil pengacakan dalam bentuk dokumen berjenis *comma-separated values*. Jendela baru untuk memilih direktori penyimpanan akan ditampilkan perangkat lunak, apabila pengguna membatalkan atau dengan kata lain menutup jendela tersebut tanpa memilih direktori penyimpanan maka perangkat lunak tidak akan melanjutkan proses pengacakan dan dianggap gagal. Tampilan antarmuka *popup* yang akan tampil setelah perangkat lunak berhasil melakukan proses pengacakan dan menyimpan hasilnya pada direktori yang pengguna pilih dapat dilihat pada Gambar 1.14. Perangkat lunak juga akan menampilkan berbagai informasi hasil pengacakan pada bagian antarmuka deskripsi hasil pengacakan yang akan dijelaskan pada subbab berikutnya.

Ada beberapa persyaratan yang harus dipenuhi oleh pengguna sebelum melakukan pengacakan yaitu sebagai berikut.

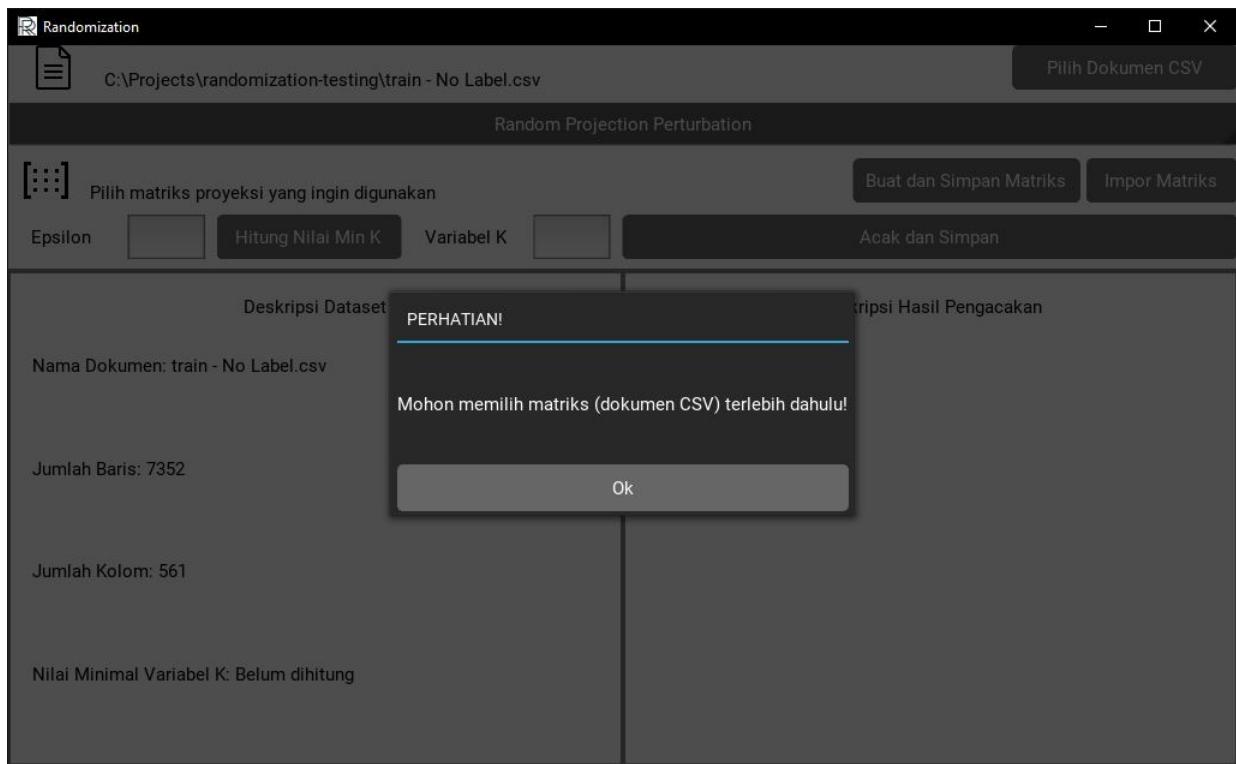
- Memilih *dataset* yang ingin diacak.



Gambar 1.13: Tampilan saat perangkat lunak sedang melakukan proses pengacakan



Gambar 1.14: Tampilan *popup* untuk memberitahukan pengguna bahwa pengacakan berhasil dilakukan



Gambar 1.15: Tampilan *popup* peringatan apabila pengguna belum memilih atau membuat matriks rotasi/proyeksi

- Memilih teknik *Randomization* yang diinginkan.
- Membuat atau memilih matriks rotasi atau proyeksi.
- Memberikan masukan nilai variabel *epsilon* dan nilai variabel *k*.

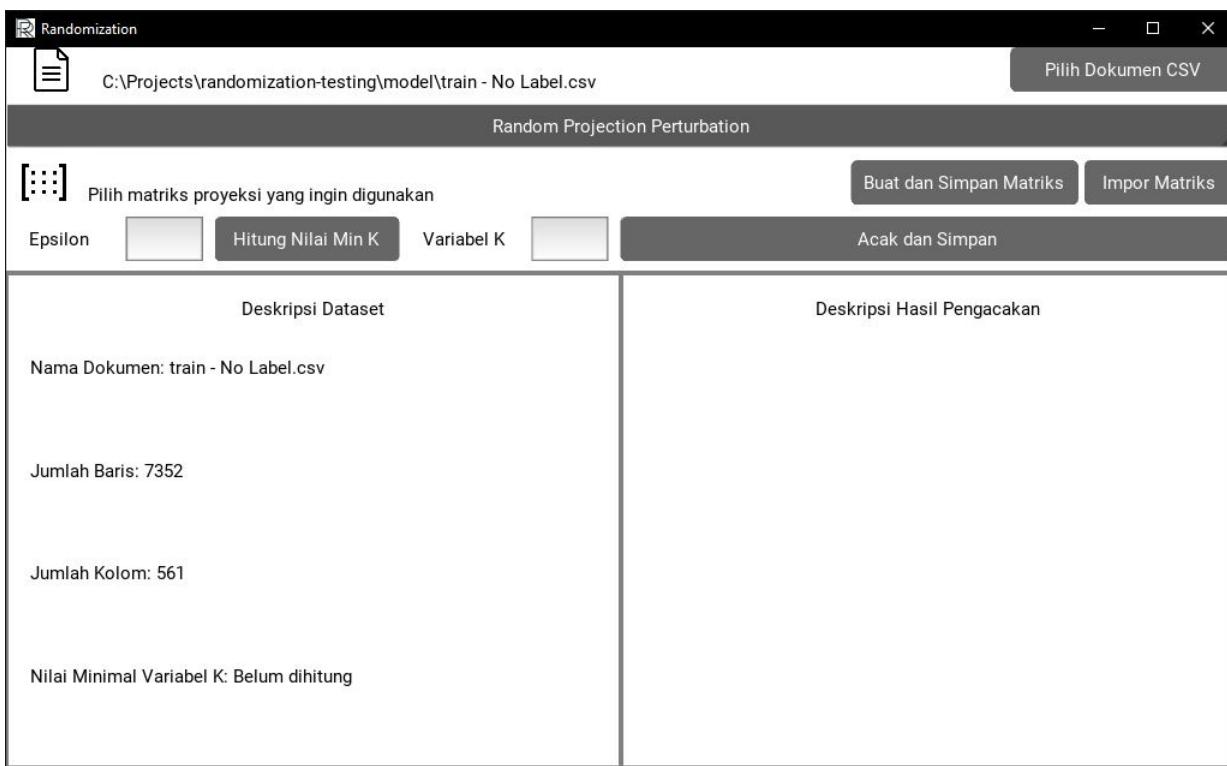
Persyaratan terakhir hanya berlaku apabila pengguna memilih teknik *Random Projection Perturbation*. Apabila ada persyaratan yang tidak dipenuhi oleh pengguna maka perangkat lunak akan menampilkan *popup* untuk memberikan peringatan kepada pengguna dan perangkat lunak tidak akan melanjutkan proses pengacakan. Perangkat lunak akan menampilkan *popup* peringatan terhadap pelanggaran masing-masing persyaratan tersebut, salah satu contoh tampilan antarmuka *popup* tersebut dapat dilihat pada Gambar 1.15.

### 1.1.2 Deskripsi Dataset

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

- Nama dokumen
- Jumlah baris *dataset*
- Jumlah kolom *dataset*
- Nilai minimal variabel *k*

Seperti yang disinggung pada subbab sebelumnya, pada awalnya nilai minimal variabel *k* belum diketahui karena belum dihitung. Pengguna harus mengisi variabel *epsilon* dan menekan tombol "Hitung Nilai Min K" agar perangkat lunak menghitung nilai minimal variabel *k* dan dapat



Gambar 1.16: Tampilan antarmuka deskripsi *dataset* setelah pengguna memilih *dataset* yang ingin diacak

menampilkannya pada deskripsi *dataset*. Nilai minimal variabel *k* hanya ditampilkan apabila pengguna memilih teknik *Random Projection Perturbation*.

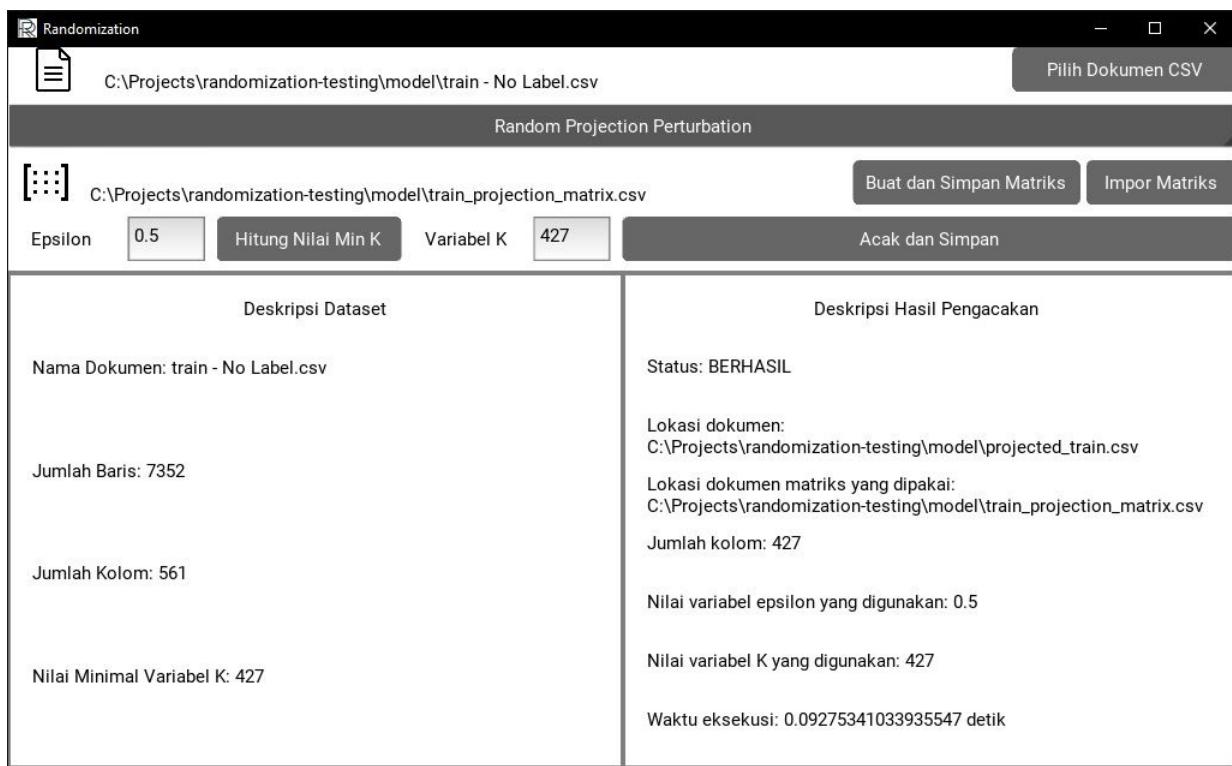
Bagian antarmuka deskripsi *dataset* ini akan selalu secara otomatis diperbaharui setiap pengguna memilih *dataset* baru. Tampilan antarmuka bagian deskripsi *dataset* dapat dilihat pada Gambar 1.2 yang dikelilingi oleh kotak berwarna biru dan bermotor dua. Apabila pengguna telah memilih *dataset* yang diinginkan untuk diacak maka perangkat lunak secara otomatis akan memperbaharui tampilan antarmuka deskripsi *dataset* yang dapat dilihat pada Gambar 1.16

### 1.1.3 Deskripsi Hasil Pengacakan

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

- Status
- Lokasi dokumen
- Lokasi dokumen matriks yang dipakai
- Jumlah kolom
- Waktu eksekusi
- Nilai variabel *epsilon* yang digunakan
- Nilai variabel *k* yang digunakan

Nilai variabel *epsilon* dan nilai variabel *k* hanya ditampilkan apabila pengguna memilih teknik *Randomization Random Projection Perturbation*.



Gambar 1.17: Tampilan antarmuka deskripsi hasil pengacakan setelah perangkat berhasil melakukan pengacakan

Bagian antarmuka deskripsi *dataset* ini akan selalu secara otomatis diperbaharui setiap pengguna memilih *dataset* baru. Tampilan antarmuka bagian deskripsi *dataset* dapat dilihat pada Gambar 1.2 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 1.17

## 1.2 Pengujian Fungsional

Pengujian fungsional bertujuan untuk memastikan perangkat lunak *Randomization* dapat menerapkan 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*.

### 1.2.1 Teknik *Random Rotation Perturbation*

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

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

0	1	2	0	1	2	3
-0.52066	-0.84872	0.09268	1	0	0	0
-0.76273	0.41361	-0.49717	0	1	0	0
0.38362	-0.32955	-0.86269	0	0	1	0
			51.3473	78.832	24.2875	1

Gambar 1.18: Matriks rotasi dan translasi acak yang dibuat perangkat lunak dan disimpan pada satu dokumen *comma-separated values*

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

Pada pengujian yang dilakukan, perangkat lunak berhasil membuat kedua matriks acak tersebut dengan ukuran yang benar dan menyimpannya pada satu dokumen *comma-separated values* yang isinya dapat dilihat pada Gambar 1.18. Matriks rotasi dan translasi ini akan digunakan untuk menerapkan teknik *Random Rotation Perturbation* terhadap *dataset mall\_customers*. Perangkat lunak juga terbukti berhasil menggunakan ulang kedua matriks yang telah disimpan tersebut untuk menerapkan teknik *Random Rotation Perturbation* terhadap *dataset mall\_customers* dan hasilnya sama persis seperti hasil yang pertama kali. Kedua matriks tersebut juga dapat digunakan untuk data *mall\_customers* yang lain dengan syarat masih memiliki jumlah fitur yang sama.

Berikut akan ditampilkan 20 baris pertama *dataset* asli dan *dataset* yang telah diacak masing-masing pada Gambar 1.19 dan Gambar 1.20. Dapat dilihat pada gambar tersebut, *dataset* setelah diacak memiliki nilai yang sangat berbeda dengan aslinya. Terlebih lagi jika diperhatikan, nilai yang sama pada beberapa baris di *dataset* asli tidak sama dengan *dataset* yang telah diacak pada baris yang sama seperti pada kolom *Annual Income* baris 10 sampai 12 memiliki nilai 19 pada *dataset* asli tetapi pada *dataset* yang telah diacak ketiga baris tersebut memiliki nilai yang berbeda 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 1.18. 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 1.1 dengan variabel *x* adalah baris pertama kolom *Age*, *y* adalah baris pertama kolom *Annual Income*, dan *z* adalah baris pertama kolom *Spending Score*.

$$\begin{aligned}
 x &= 19 \times 1 + 15 \times 0 + 39 \times 0 + 1 \times 51.3473 & (1.1) \\
 &= 70.3473 \\
 y &= 19 \times 0 + 15 \times 1 + 39 \times 0 + 1 \times 78.832 \\
 &= 93.832 \\
 z &= 19 \times 0 + 15 \times 0 + 39 \times 1 + 1 \times 24.2875 \\
 &= 63.2875
 \end{aligned}$$

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 1.19: Dua puluh baris pertama dataset *mall\_customers* asli

CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
1	Male	-83.9167684	-41.75111372	-94.72755928
2	Male	-68.84610412	-57.289563	-130.7751899
3	Female	-97.8595706	-31.31113682	-66.66325546
4	Female	-72.18462466	-57.25519421	-127.6362252
5	Female	-91.30652421	-51.4380519	-95.47236638
6	Female	-72.81030801	-55.66331508	-127.3633861
7	Female	-107.1949262	-43.2146812	-66.26731791
8	Female	-67.18855698	-62.03028277	-143.2962966
9	Male	-124.2076581	-66.42524758	-61.48855616
10	Female	-80.0355158	-60.30764141	-124.1654797
11	Male	-121.5498336	-72.59642898	-70.70009724
12	Female	-72.281113	-73.44902584	-146.994699
13	Female	-117.2429966	-64.87390135	-72.89411688
14	Female	-75.75618517	-56.44945965	-129.5322071
15	Male	-107.0763662	-46.39172652	-73.11511512
16	Male	-73.94762704	-55.41111943	-131.4429575
17	Female	-98.35815929	-51.53073181	-92.77684172
18	Male	-78.65607432	-49.01594705	-120.9105174
19	Male	-111.0365478	-63.15442469	-87.01939187
20	Female	-75.71563183	-71.4650253	-148.1206751

Gambar 1.20: Dua puluh baris pertama dataset *mall\_customers* setelah diacak

Tabel 1.1: 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

Setelah ditranslasi, transformasi rotasi dilakukan dan hasil akhir dapat dilihat pada Persamaan 1.2.

$$\begin{aligned}
 result &= 70.3473 \times -0.52066 + 93.832 \times -0.76273 + 63.2875 \times 0.38362 \\
 &= -36.627077284 - 71.56848136 + 24.27835075 \\
 &= -83.917207894
 \end{aligned} \tag{1.2}$$

Dapat dibandingkan hasil akhir perhitungan manual dengan hasil akhir perangkat lunak pada Gambar 1.20, baris pertama pada kolom *Age* memiliki nilai yang sama persis sampai 2 angka di belakang koma (dikarenakan pembulatan pada matriks rotasi untuk visualisasi di dokumen skripsi ini yang dapat dilihat pada Gambar 1.18).

Pengujian terhadap jarak Euclidean antara seluruh objek data pada *dataset* asli dan yang telah diacak juga dilakukan. Pada Tabel 1.1 dapat dilihat sebagian kecil keluaran dari perangkat lunak pengujian yang merupakan contoh perbedaan jarak Euclidean antara sebuah baris yang merepresentasikan sebuah objek data pada *dataset* dengan objek data lainnya. Seluruh objek data pada *dataset* terbukti memiliki jarak Euclidean yang sama pada *dataset* asli dan *dataset* yang telah diacak. Pengujian perbandingan jarak Euclidean dilakukan hanya memakai nilai pembulatan sampai 2 atau 3 angka di belakang koma saja karena tidak terlalu relevan untuk mengambil seluruh angka di belakang koma. Berdasarkan pengujian yang telah dilakukan, dapat disimpulkan bahwa perangkat lunak sudah berfungsi dengan baik dan benar dalam menerapkan teknik *Random Rotation Perturbation*.

### 1.2.2 Teknik *Random Projection Perturbation*

Pengujian teknik *Random Projection Perturbation* akan menggunakan *dataset mobile\_sensor*<sup>2</sup> yang berisi data sensor *smartphone* banyak orang yang sedang melakukan aktivitas tertentu yaitu berdiri, duduk, berbaring, berjalan, berjalan menanjak, berjalan menurun. *Dataset* ini memiliki 561 buah fitur (bersifat numerik) dan sebuah label. Seluruh fitur tersebut akan diacak dan diharapkan hasilnya akan mengacak *dataset* sehingga nilai tiap fitur tersebut berbeda dari aslinya. *Dataset* yang telah diacak juga diharapkan masih dapat diterapkan teknik penambangan data dengan hasil 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 1.3 adalah contoh perhitungan manual rumus nilai minimal variabel *k* dengan nilai variabel *epsilon* sebesar 0.52 dan

<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 1.21: Matriks proyeksi acak yang dibuat perangkat lunak dan disimpan pada sebuah dokumen *comma-separated values*

Jumlah baris pada *dataset* sebanyak 10299 baris.

$$\begin{aligned}
 k &= \frac{4 \ln n}{\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}} \\
 &= \frac{4 \ln 10299}{\frac{0.52^2}{2} - \frac{0.52^3}{3}} \\
 &= \frac{36.9592}{0.1352 - 0.0468} \\
 &= 418.0905
 \end{aligned} \tag{1.3}$$

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

Pada pengujian yang dilakukan, perangkat lunak berhasil membuat matriks proyeksi acak tersebut dengan ukuran yang benar dan menyimpannya pada sebuah dokumen *comma-separated values* yang 14 kolom pertama pada 20 baris pertamanya dapat dilihat pada Gambar 1.21. Matriks proyeksi ini akan digunakan untuk menerapkan teknik *Random Projection Perturbation* terhadap *dataset mobile\_sensor*. Perangkat lunak juga terbukti berhasil menggunakan ulang matriks yang telah disimpan tersebut untuk menerapkan teknik *Random Projection Perturbation* terhadap *dataset mobile\_sensor* dan hasilnya sama persis seperti hasil yang pertama kali. Matriks proyeksi tersebut juga dapat digunakan untuk data *mobile\_sensor* yang lain dengan syarat masih memiliki jumlah fitur yang sama dan dengan penambahan data tersebut, nilai minimal variabel  $k$  harus tetap lebih kurang atau sama dengan nilai variabel  $k$  yang dipilih. Persyaratan tersebut didasarkan oleh sifat pada teknik *Random Projection Perturbation* yaitu nilai minimal variabel  $k$  berbanding lurus dengan banyaknya objek data pada *dataset* yang ingin diacak.

Berikut akan ditampilkan 7 fitur terakhir serta label pada 20 baris pertama *dataset* asli dan *dataset* yang telah diacak masing-masing pada Gambar 1.22 dan Gambar 1.23. Dapat dilihat pada gambar tersebut, *dataset* setelah diacak memiliki nilai yang berbeda dengan aslinya. Terlebih lagi

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

setiap fitur pada *dataset* yang telah diacak tidak diketahui arti dari setiap fitur tersebut apa karena sudah tereduksi sehingga seluruh fitur pada *dataset* asli tercampur secara acak dan terproyeksikan ke dalam 427 fitur yang ada pada *dataset* yang telah diacak. Dalam rangka untuk memastikan perangkat lunak berhasil dengan benar menerapkan teknik *Random Projection Perturbation* dengan matriks proyeksi acak yang sebelumnya sudah dibuat dan disimpan oleh perangkat lunak, perhitungan manual dilakukan terhadap *dataset mobile\_sensor* dengan menggunakan matriks proyeksi tersebut. Hasil akhir dari perangkat lunak sama persis dengan hasil dari perhitungan manual. Oleh karena itu, dapat disimpulkan bahwa perangkat lunak sudah berfungsi dengan baik dan benar dalam menerapkan teknik *Random Projection Perturbation*.

Dalam rangka memastikan lebih lagi apakah perangkat lunak menerapkan teknik *Random Projection Perturbation* dengan akurat, perangkat lunak pengujian dibuat untuk menguji jarak Euclidean dari setiap titik pada *dataset* terhadap setiap titik lainnya. Pengujian dilakukan dengan menggunakan pertidaksamaan rentang jarak Euclidean berikut untuk menguji apakah jarak Euclidean pada *dataset* yang telah diacak mempunyai distorsi yang sesuai dengan harapan pengguna. Hasil dari perangkat lunak (setiap jarak Euclidean antara sebuah objek data dengan objek data yang lainnya pada *dataset* yang telah diacak) diharapkan memenuhi Pertidaksamaan 1.4.

$$(1 - \epsilon)|u - v|^2 < |p(u) - p(v)|^2 < (1 + \epsilon)|u - v|^2 \quad (1.4)$$

Pada pengujian ini, dengan nilai variabel *epsilon* sebesar 0.52 Pertidaksamaan 1.4 terpenuhi pada seluruh objek data yang ada pada *dataset*. Pada Tabel 1.2 dapat dilihat sebagian kecil keluaran dari perangkat lunak pengujian yang merupakan contoh perbedaan jarak Euclidean antara sebuah baris yang merepresentasikan sebuah objek data pada *dataset* dengan objek data lainnya. Salah satu contoh saat Pertidaksamaan 1.4 tidak terpenuhi adalah apabila nilai variabel *epsilon* ditentukan sebesar 0.4. Salah satu jarak Euclidean yang melanggar Pertidaksamaan 1.4 adalah baris 473 dengan baris 1306 yang memiliki jarak Euclidean sebesar 8.167734238223167 pada *dataset* asli dan 9.723888530285468 pada *dataset* yang telah diacak, perhitungan manual pengujianya dapat dilihat

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 1.23: Dua puluh baris terakhir *dataset mobile\_sensor* setelah diacakTabel 1.2: Contoh perbedaan jarak Euclidean antara *dataset mobile\_sensor* asli dan yang telah diacak dengan *Random Projection Perturbation*

Baris	Baris	Dataset Asli	Dataset Diacak
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

pada Pertidaksamaan 1.5. Pengujian ini membuktikan bahwa perangkat lunak berhasil menerapkan teknik *Random Projection Perturbation* dengan akurat sesuai batas distorsi yang ditentukan oleh pengguna.

$$(1 - \text{eps})\|u - v\|^2 < \|p(u) - p(v)\|^2 \not< (1 + \text{eps})\|u - v\|^2 \quad (1.5)$$

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

$$40.02712955174579 < 94.55400814941728 \not< 93.39663562074017$$

### 1.3 Pengujian Eksperimental

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

1. Properti Data
  - (a) *Random Rotation Perturbation* dengan *dataset diabetes*
  - (b) *Random Projection Perturbation* dengan *dataset mobile\_sensor*

2. Penambangan Data Klasifikasi
  - (a) *Random Rotation Perturbation* dengan dataset *diabetes*
  - (b) *Random Projection Perturbation* dengan dataset *mobile\_sensor*
3. Penambangan Data *Clustering*
  - (a) *Random Rotation Perturbation* dengan dataset *mall\_customers*
  - (b) *Random Projection Perturbation* dengan dataset *mobile\_sensor*
4. Kesimpulan Akhir

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

### 1.3.1 Properti Data

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

1. Rata-rata
2. Standar deviasi
3. Nilai terkecil
4. Nilai terbesar
5. Kuartil bawah
6. Kuartil tengah
7. Kuartil atas

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

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

Pengujian teknik *Random Rotation Perturbation* untuk membandingkan properti data akan dilakukan dengan *dataset diabetes*<sup>3</sup> yang dapat dilihat 20 baris pertamanya pada Gambar 1.24. *Dataset* ini diacak dengan teknik *Random Rotation Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 1.25.

Properti-properti 4 kolom terakhir pada *dataset diabetes* asli dapat dilihat pada Tabel 1.3. Sementara untuk *dataset diabetes* yang telah diacak dapat dilihat pada Tabel 1.4. Jika dilihat pada kedua tabel tersebut, seluruh properti pada *dataset* yang telah diacak mempunyai nilai yang berbeda kecuali jumlah baris (*count*) dan kolom label (*Outcome*). Hal ini menunjukkan selain nilai pada setiap data, teknik *Random Rotation Perturbation* juga mengacak bermacam properti data seperti rata-rata, standar deviasi, nilai terkecil, nilai terbesar, kuartil bawah, kuartil tengah dan kuartil atas setiap kolom pada *dataset*.

---

<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 1.24: 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 1.25: Dua puluh baris pertama dataset diabetes setelah diacak

Tabel 1.3: 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 1.4: 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

### Random Projection Perturbation

Pengujian teknik *Random Projection Perturbation* untuk membandingkan properti data akan dilakukan dengan *dataset mobile\_sensor* yang dapat dilihat 20 baris pertamanya pada Gambar 1.22. *Dataset* ini diacak dengan teknik *Random Projection Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 1.23.

Properti-properti 3 buah kolom pada *dataset mobile\_sensor* asli dapat dilihat pada Tabel 1.5. Sementara untuk *dataset mobile\_sensor* yang telah diacak dapat dilihat pada Tabel 1.6. Jika dilihat pada kedua tabel tersebut, seluruh properti pada *dataset* yang telah diacak mempunyai nilai yang berbeda kecuali jumlah baris (*count*). Hal ini menunjukkan selain nilai pada setiap data, teknik *Random Projection Perturbation* juga mengacak bermacam properti *dataset* seperti rata-rata, standar deviasi, nilai terkecil dan terbesar pada sebuah kolom, kuartil bawah, kuartil tengah dan kuartil atas setiap kolom pada *dataset*.

Tabel 1.5: 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 1.6: 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 1.7: 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

## Kesimpulan

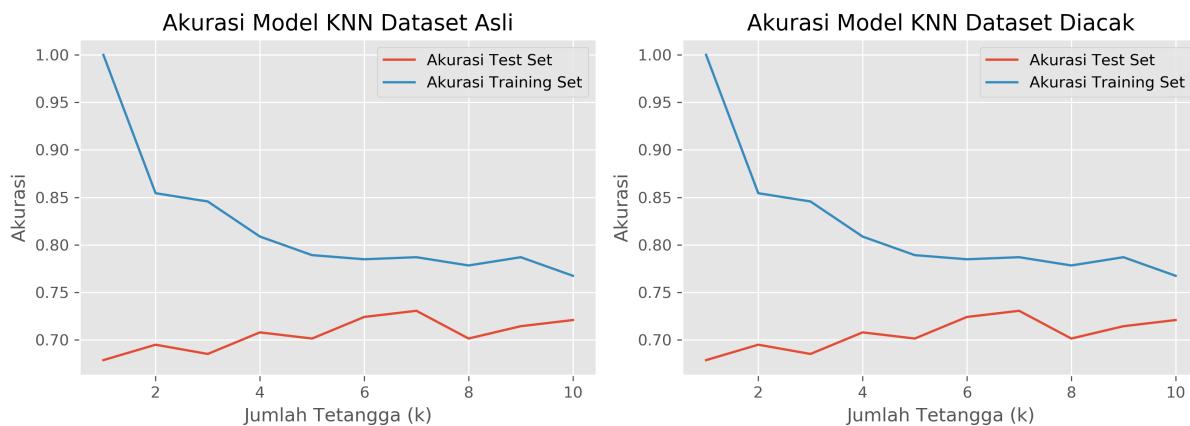
Berdasarkan pengujian properti data akan dibuat kesimpulan sekaligus membandingkan antara *dataset* asli, *dataset* yang telah diacak menggunakan teknik *Random Rotation Perturbation*, dan *dataset* yang telah diacak menggunakan teknik *Random Projection Perturbation*. Pada Tabel 1.7 dapat dilihat hasil akhir pengujian antara properti data pada *dataset* asli dan *dataset* yang telah diacak dengan *Random Rotation Perturbation* dan *Random Projection Perturbation*. Dapat dilihat pada tabel tersebut metode *Randomization* mengacak data dengan baik dan tidak menjaga properti-properti lain pada data selain jarak Euclidean sehingga properti-properti data tersebut tidak dapat digunakan setelah data diacak dengan metode *Randomization*.

### 1.3.2 Penambangan Data Klasifikasi

Pengujian dengan penambangan data klasifikasi akan berpusat pada pembuatan model dengan *dataset* asli dan *dataset* yang telah diacak dan membandingkan kedua model tersebut. Teknik penambangan data klasifikasi yang digunakan adalah *k-nearest neighbors*. Pengujian dengan penambangan data klasifikasi akan dibagi menjadi 2 bagian yaitu *dataset* yang diacak dengan teknik *Random Rotation Perturbation* dan *dataset* yang diacak dengan teknik *Random Projection Perturbation*. Pengujian akan membandingkan beberapa informasi pada *dataset* asli dan *dataset* yang telah diacak. Beberapa informasi tersebut adalah sebagai berikut.

1. Akurasi model *k-nearest neighbors*
2. Nilai variabel *k* yang modelnya memiliki akurasi tertinggi
3. Waktu eksekusi pelatihan model dan prediksi dengan model yang telah dibuat

Tujuan dari pengujian ini adalah mencari informasi yang masih terjaga (mempunyai hasil yang sama atau mirip) antara model yang dilatih dengan *dataset* asli dan *dataset* yang telah diacak.



Gambar 1.26: Grafik akurasi model klasifikasi pada *training set* dan *test set* *dataset diabetes*

### ***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 1.24. *Dataset* ini diacak dengan teknik *Random Rotation Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 1.25. 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 1.26 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 1.1 dan Listing 1.2. 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 1.1: *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
```

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

Listing 1.2: *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
```

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

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

Pengujian teknik *Random Projection Perturbation* untuk penambangan data klasifikasi dengan algoritma *k-nearest neighbors* akan dilakukan dengan *dataset mobile\_sensor* yang dapat dilihat 20 baris pertamanya pada Gambar 1.22. *Dataset* ini diacak dengan teknik *Random Projection Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 1.23. 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 561 fitur yang ada untuk menguji apakah *dataset* mempunyai akurasi model klasifikasi yang hampir sama. Pengujian tersebut didasarkan pada sifat teknik *Random Projection Perturbation* yang menjamin jarak Euclidean setiap titik terjaga dengan besar distorsi yang ditentukan oleh pengguna. *Dataset* akan dibagi dua menjadi *train set* dan *test set* yang masing-masing berguna untuk melatih model dan menghitung akurasi model. Akurasi akan dihitung dengan jumlah tetangga (nilai variabel  $k$ ) dari 1 sampai 30. Implementasi kode teknik penambangan data ini diterapkan dengan bahasa pemrograman Python dan dibantu oleh *library* Scikit-learn.

Pada Gambar 1.27 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 memiliki nilai akurasi yang mirip. Walaupun begitu, teknik *Random Projection Perturbation* tidak menjamin jarak Euclidean terjaga dengan sempurna maka ada sedikit perbedaan yang lumayan terlihat khususnya pada akurasi *test set* dan jumlah tetangga (nilai variabel  $k$ ) pada model yang memiliki akurasi tertinggi. Tetapi perbedaan nilai akurasi tertingginya masih mirip dengan *dataset* asli.

Contoh sebagian kecil keluaran perangkat lunak pengujian yaitu akurasi model untuk setiap nilai variabel  $k$  dari 19 sampai 23 pada *dataset* asli dan *dataset* yang telah diacak dapat dilihat masing-masing pada Listing 1.3 dan Listing 1.4. Dapat dilihat pada kedua listing tersebut akurasi



Gambar 1.27: Grafik akurasi model klasifikasi pada *training set* dan *test set* dataset *mobile\_sensor*

*test set* pada kedua *dataset* berbeda. Akurasi *test set* tertinggi pada *dataset* asli adalah sebesar 0.9077027485578555 dengan nilai variabel  $k$  sebesar 16. Sementara pada *dataset* yang telah diacak, akurasi *test set* tertingginya adalah sebesar 0.9005768578215134 dengan nilai variabel  $k$  sebesar 21. Jika dihitung perbedaan akurasinya adalah sebesar 0.0071258907363421 yang mana relatif kecil tetapi ada perbedaan nilai variabel  $k$  pada model yang memiliki akurasi tertinggi. Dengan perbedaan akurasi yang relatif kecil maka dapat disimpulkan bahwa teknik *Random Projection Perturbation* menjaga jarak Euclidean dengan baik dan distorsinya terkontrol sesuai yang pengguna inginkan. Oleh karena itu, model *k-nearest neighbors* yang terbuat memiliki hasil yang sangat mirip.

Listing 1.3: 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
```

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

Listing 1.4: 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
```

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.

Tabel 1.8: 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

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

### 1.3.3 Penambangan Data *Clustering*

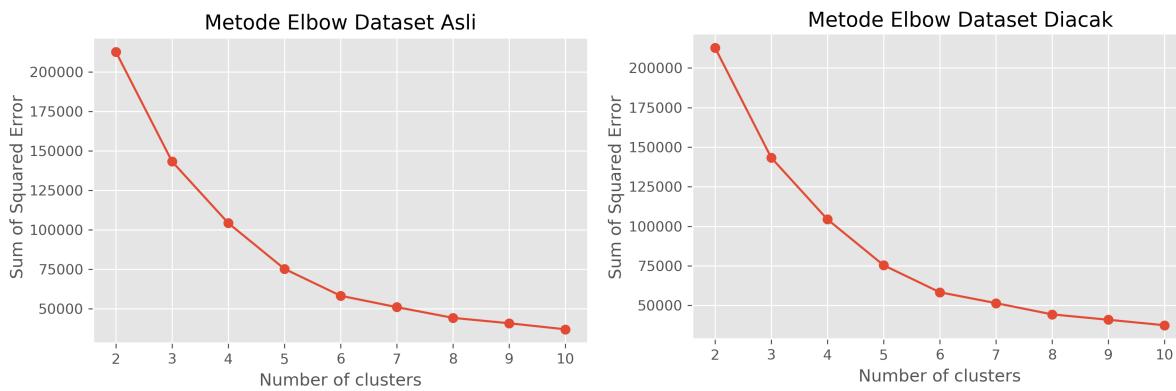
Pengujian dengan penambangan data *clustering* akan berpusat pada pembuatan model *clustering* dengan *dataset* asli dan *dataset* yang telah diacak dan membandingkan kedua model tersebut. Teknik penambangan data *clustering* yang digunakan adalah *k-means*. Pengujian akan membandingkan beberapa informasi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

1. *Sum of Squared Error*
2. *Silhouette Score*
3. Nilai variabel *k* (jumlah *cluster*) yang modelnya memiliki kualitas *clustering* terbaik
4. Visualisasi *cluster*
5. Waktu eksekusi pelatihan model dan prediksi dengan model yang telah dibuat

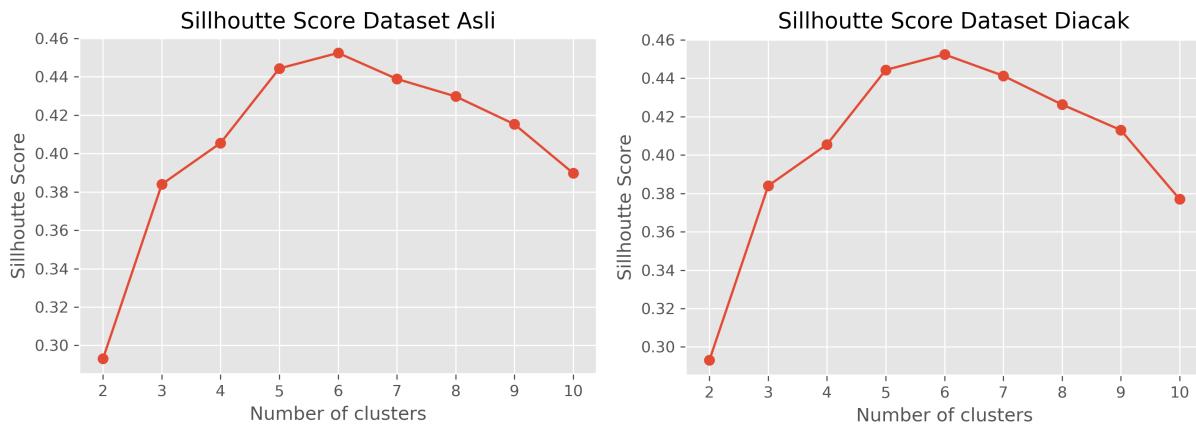
Tujuan dari pengujian ini adalah mencari informasi yang masih terjaga (mempunyai hasil yang sama atau mirip) antara model yang dilatih dengan *dataset* asli dan *dataset* yang telah diacak. Selain itu metode *Adjusted Rand Index* juga digunakan untuk menghitung kemiripan antara kedua model.

#### *Random Rotation Perturbation*

Pengujian teknik *Random Rotation Perturbation* untuk penambangan data *clustering* dengan menggunakan algoritma *k-means* akan dilakukan dengan *dataset* *mall\_customers* yang dapat dilihat 20 baris pertamanya pada Gambar 1.19. *Dataset* ini diacak dengan teknik *Random Rotation*



Gambar 1.28: Grafik *Sum of Squared Error* model *clustering* pada dataset *mall\_customers*



Gambar 1.29: Grafik *Silhouette Score* model *clustering* pada dataset *mall\_customers*

*Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 1.20. 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.

Dalam menentukan jumlah *cluster* atau nilai variabel *k* yang terbaik untuk membuat model *clustering* dengan algoritma *k-means* perlu ada metode untuk menentukan nilai tersebut. Metode Elbow menjadi salah satu metode yang digunakan untuk menentukan nilai variabel *k* yang terbaik untuk dipakai. Pada Gambar 1.28 terdapat grafik *Sum of Squared Error* untuk menggunakan metode Elbow pada *dataset* asli dan *dataset* yang telah diacak. Terlihat nilai variabel *k* sebesar 5, 6, dan 7 adalah kandidat terbaik untuk menjadi nilai variabel *k* yang dipakai pada *dataset* asli maupun *dataset* yang telah diacak. Pada kedua *dataset* tersebut grafik *Sum of Squared Error*-nya terlihat sama persis dikarenakan teknik *Random Rotation Perturbation* menjaga jarak Euclidean dengan sempurna. Dalam menentukan nilai variabel *k* yang terbaik antara ketiga nilai tersebut, *Silhouette Score* dapat digunakan untuk metode alternatif untuk menentukan nilai variabel *k* yang terbaik.

Pada Gambar 1.29 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 1.5 dan Listing 1.6 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 1.5: Dataset *mall\_customers* Asli

```
Silhouette Score setiap K
pada dataset asli :
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 1.6: 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
```

Teknik penambangan data *k-means* diterapkan menggunakan 3 buah fitur yang ada untuk menguji apakah *dataset* asli dan *dataset* yang telah diacak menghasilkan kluster dan bentuk yang sama dengan sudut yang berbeda saat divisualisasikan. Pengujian tersebut didasarkan pada sifat teknik *Random Rotation Perturbation* yang menjamin jarak Euclidean antara setiap titik tidak berubah sama sekali tetapi merotasi seluruh titik yang ada pada bidang Euclidean. 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 6 pada *dataset mall\_customers* asli dan yang telah diacak dapat dilihat pada Gambar 1.30 dan Gambar 1.31. Dapat dilihat pada kedua visualisasi tersebut mempunyai jumlah *cluster* yang sama dan terlihat dari lokasi titik-titik yang ada jika dibandingkan terlihat seperti dirotasi searah jarum jam dan bentuknya masih terlihat sama. Apabila dihitung kemiripan *cluster* tersebut dengan metode *Adjusted Rand Index* maka kedua hasil *clustering* tersebut mempunyai nilai 1.0 yang berarti titik-titik yang ada pada setiap *cluster* pada kedua model persis adanya.

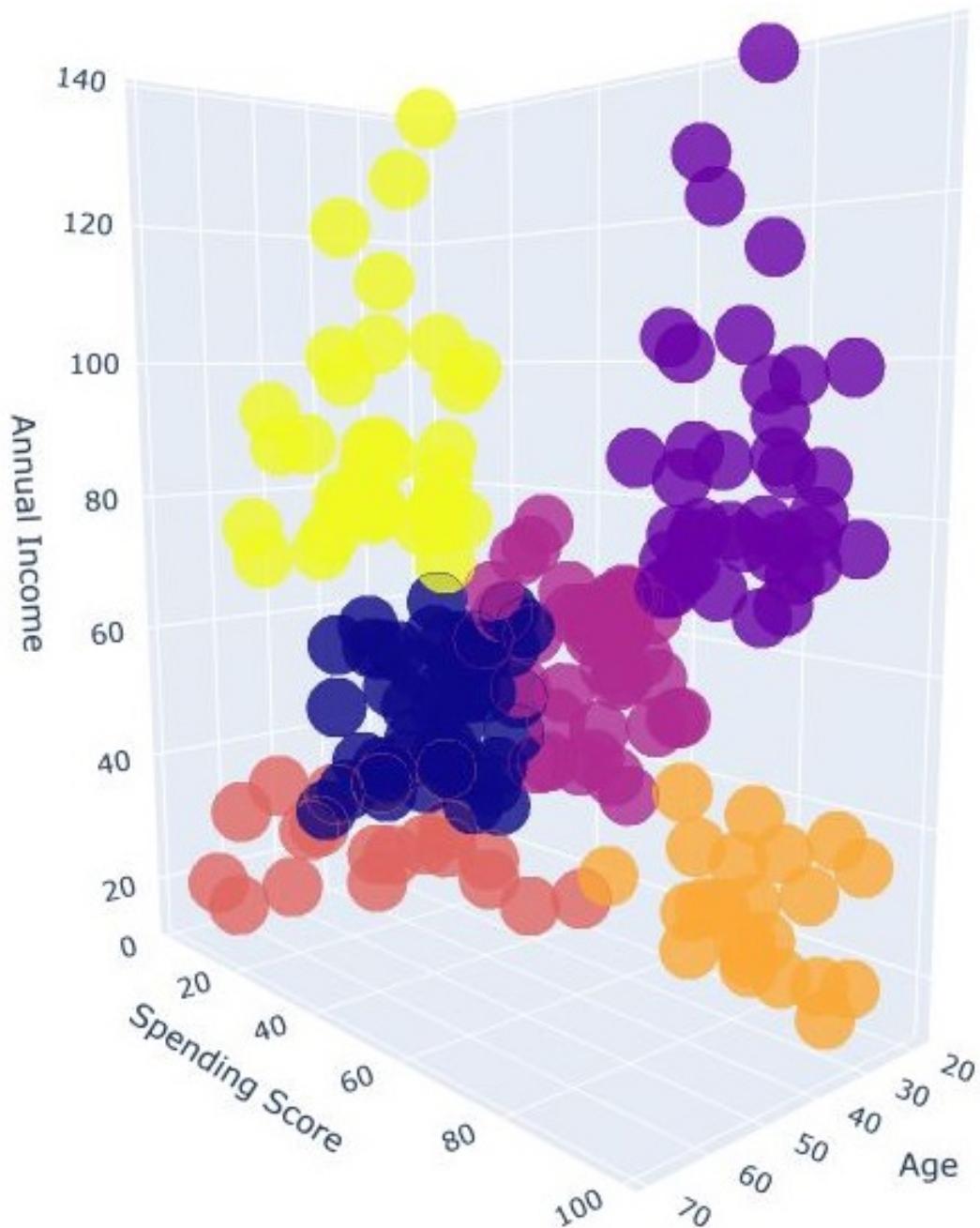
Waktu eksekusi yang dibutuhkan untuk melatih model *clustering* dengan algoritma *k-means* adalah sebesar 0.02995157241821289 detik pada *dataset* yang asli dan sebesar 0.032910823822021484 detik pada *dataset* yang telah diacak, hal ini menunjukkan bahwa teknik *Random Rotation Perturbation* tidak mempengaruhi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

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

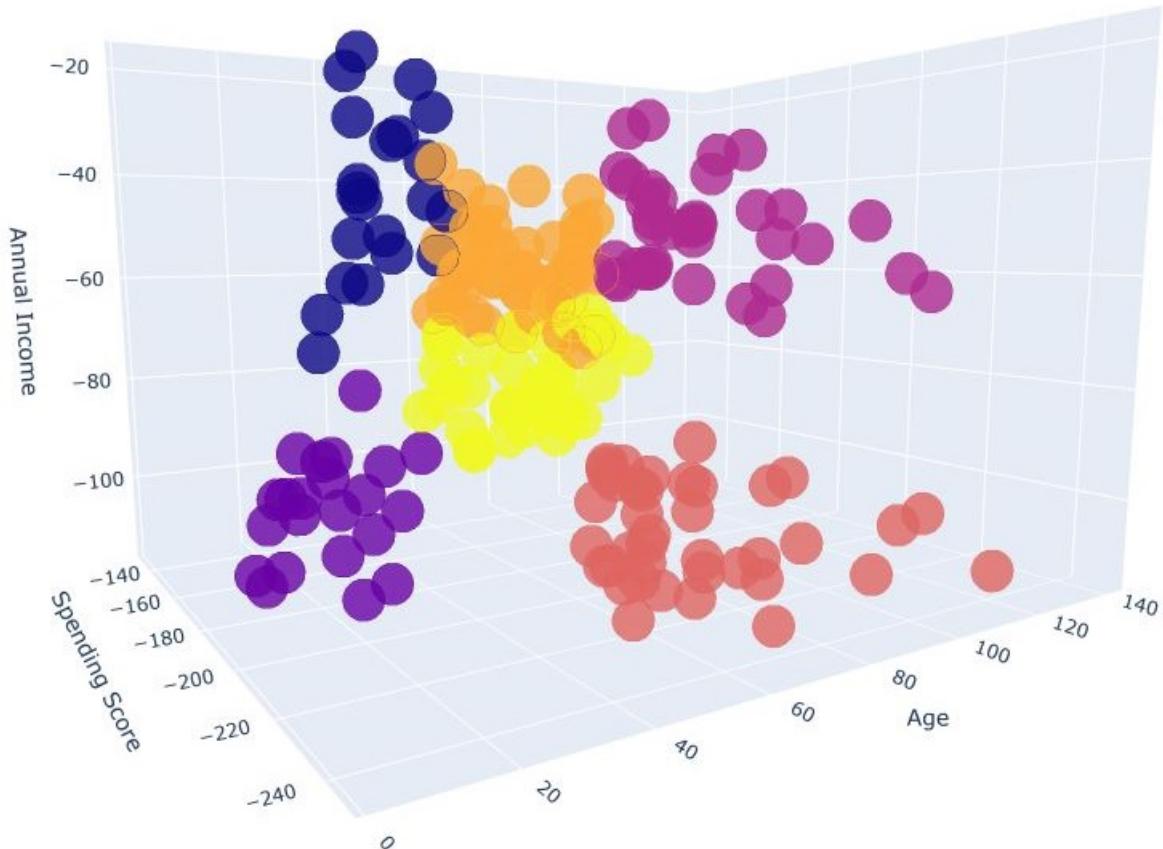
Pengujian teknik *Random Projection Perturbation* untuk penambangan data *clustering* dengan algoritma *k-means* akan dilakukan dengan *dataset mobile\_sensor* yang dapat dilihat 20 baris pertamanya pada Gambar 1.22. *Dataset* ini diacak dengan teknik *Random Projection Perturbation* dan hasil pengacakan dapat dilihat pada Gambar 1.23. 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.

Sebelum melakukan teknik *clustering* dengan algoritma *k-means*, *dataset* yang memiliki fitur yang sangat banyak tersebut dimensinya harus direduksi terlebih dahulu agar model dapat divisualisasikan dengan mudah. *Dataset* yang akan di-*cluster* akan direduksi dimensinya sampai hanya memiliki 2 dimensi. Reduksi dimensi dilakukan dengan menerapkan teknik *Principal Component Analysis* yang sudah umum digunakan pada penambangan data dan hasilnya relatif baik. Dalam menguji teknik *Random Projection Perturbation*, *dataset* yang tidak akan diacak langsung direduksi dimensinya dengan teknik *Principal Component Analysis*. *Dataset* yang akan diacak akan terlebih dahulu diacak menggunakan teknik *Random Projection Perturbation* baru direduksi dimensinya menjadi 2 dimensi dengan teknik *Principal Component Analysis*.

Pada Gambar 1.32 terdapat grafik *Sum of Squared Error* untuk menggunakan metode Elbow pada *dataset* asli dan *dataset* yang telah diacak. Pada grafik ini tidak terlalu terlihat nilai yang terbaik untuk menjadi nilai variabel *k* yang dipakai pada *dataset* asli maupun *dataset* yang telah



Gambar 1.30: Visualisasi *cluster* pada dataset *mall\_customers* yang asli

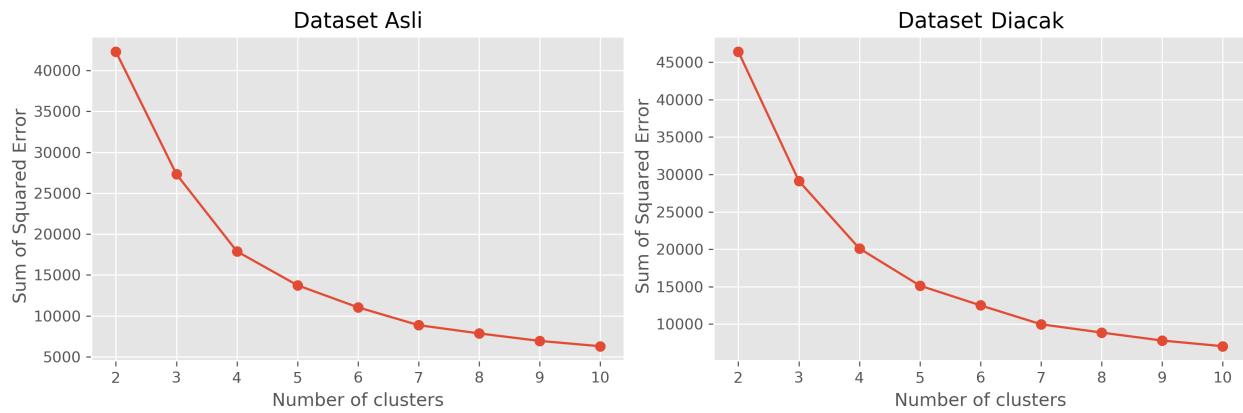


Gambar 1.31: Visualisasi *cluster* pada dataset *mall\_customers* yang telah diacak

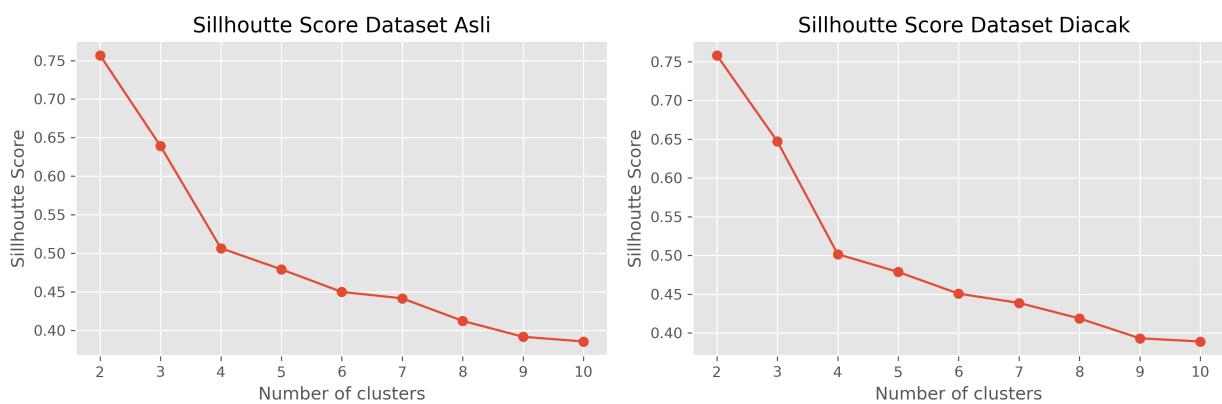
diacak. Walaupun seperti itu, pada kedua *dataset* tersebut grafik *Sum of Squared Error*-nya terlihat sangat mirip dikarenakan teknik *Random Projection Perturbation* menjaga jarak Euclidean dengan baik dan terkontrol. Dalam menentukan nilai variabel  $k$  yang terbaik untuk dipakai membuat model, *Silhouette Score* dapat digunakan untuk metode alternatif untuk menentukan nilai variabel  $k$  yang terbaik.

Pada Gambar 1.33 terdapat grafik *Silhouette Score* dari *dataset* *mobile\_sensor* yang asli dan yang telah diacak. Dapat terlihat kedua grafik *Silhouette Score* terlihat sangat mirip dan dapat dilihat nilai-nilai pada setiap  $k$  tersebut di Listing 1.7 dan Listing 1.8 ada sedikit perbedaan yang tidak terlalu signifikan. Hal ini dikarenakan oleh jarak Euclidean pada *dataset* asli dan yang telah diacak sedikit berbeda sehingga mempengaruhi sedikit pada hasil algoritma *Silhouette Score*. Dapat dilihat *Silhouette Score* pada nilai variabel  $k$  sebesar 2 adalah nilai paling besar yaitu 0.7568010158989575 pada *dataset* asli dan 0.7581796759180272 pada *dataset* yang telah diacak. Hal ini menunjukkan teknik *Random Projection Perturbation* tidak mempengaruhi secara signifikan nilai *Silhouette Score* pada setiap  $k$ .

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 diacak 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.



Gambar 1.32: Grafik *Sum of Squared Error* model *clustering* pada dataset *mobile\_sensor*



Gambar 1.33: Grafik *Silhouette Score* model *clustering* pada dataset *mobile\_sensor*

Listing 1.7: Dataset mobile\_sensor Asli

```
Silhouette 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 1.8: Dataset mobile\_sensor Diacak

```
Silhouette 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
```

Visualisasi model *clustering* dengan nilai variabel  $k$  sebesar 2 pada *dataset mobile\_sensor* asli dan yang telah diacak dapat dilihat pada Gambar 1.34. 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*.

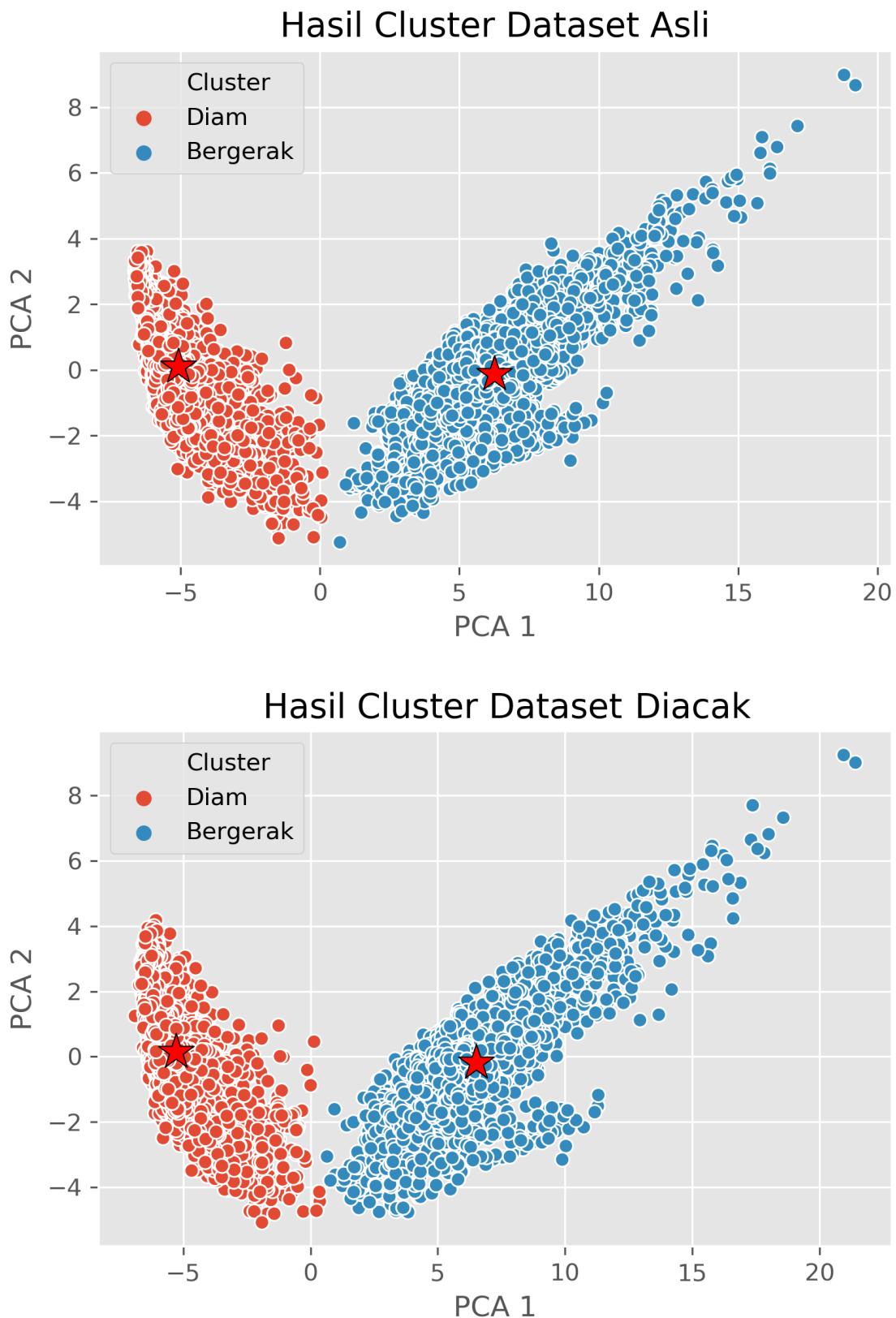
Apabila dihitung kemiripan *cluster* tersebut dengan metode *Adjusted Rand Index* maka kedua hasil *clustering* tersebut mempunyai nilai 0.9994558701020273 yang berarti titik-titik yang ada pada setiap *cluster* pada kedua model sangat mirip sekali. Hal ini dikarenakan jarak Euclidean kedua buah *dataset* 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.

## 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 1.9 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 1.34: Visualisasi *cluster* pada *dataset mobile\_sensor*

Tabel 1.9: 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

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

#### 1.3.4 Kesimpulan Akhir

Berdasarkan hasil pengujian eksperimental yang telah dilakukan dapat disimpulkan bahwa teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* dapat digunakan untuk menghilangkan privasi pada data dengan mengacak data tersebut tetapi masih dapat digunakan untuk penambangan data klasifikasi dan *clustering* masing-masing dengan teknik *k-nearest neighbors* dan *k-means* dengan hasil yang sama dengan *dataset* asli untuk teknik *Random Rotation Perturbation* dan hasil yang sangat mirip dengan *dataset* asli untuk teknik *Random Projection Perturbation*. Hal tersebut dikarenakan teknik *Random Rotation Perturbation* menjaga jarak Euclidean secara sempurna, sementara teknik *Random Projection Perturbation* tidak menjaga jarak Euclidean secara sempurna karena ada distorsi yang terkontrol pada jarak Euclidean antara setiap titik yang merepresentasikan sebuah objek data pada *dataset* yang diacak. Waktu eksekusi dalam pembuatan model klasifikasi maupun *clustering* lebih cepat pada *dataset* yang diacak dengan teknik *Random Projection Perturbation* karena ukuran *dataset* berkurang akibat reduksi dimensi yang dilakukan teknik tersebut.

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

Kualitas hasil dari teknik *Random Projection Perturbation* akan menurun dalam 2 kondisi yaitu pertama apabila data direduksi ke dimensi yang lebih kecil lagi (semakin besar dimensinya, semakin kecil distorsi pada jarak Euclidean) dan kedua adalah data yang diacak bertambah banyak. Kondisi kedua dikarenakan oleh nilai variabel *k* (dimensi minimal) berbanding lurus dengan banyaknya data. Oleh karena itu semakin sering model dilatih dengan data baru, ada kemungkinan kualitas model tersebut akan menurun apabila nilai variabel *k* (dimensi minimal) sudah melebihi besar dimensi *dataset* yang telah diacak.



## **DAFTAR REFERENSI**