

BAB 1

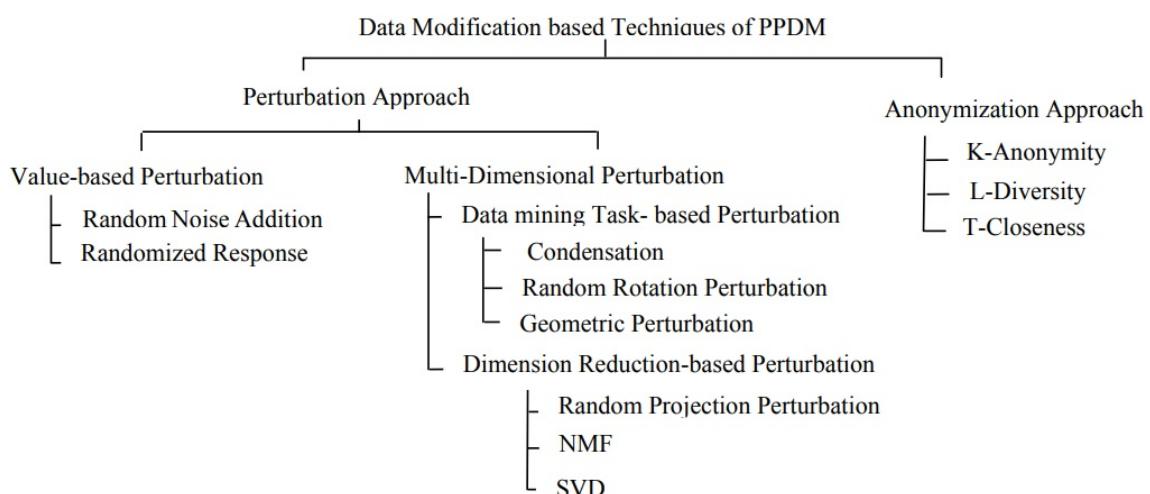
PENDAHULUAN

1.1 Latar Belakang

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

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

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



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

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.

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

Pada penelitian ini, akan dibuat sebuah perangkat lunak yang dapat memproses data yang akan ditambah menambah menjadi data yang telah dimodifikasi dengan metode *Randomization* sehingga privasi pada data tersebut terlindungi, tetapi masih dapat ditambah. Dari berbagai macam teknik dengan metode *Randomization* yang ada, dipilih dua buah teknik yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation* untuk diimplementasikan pada perangkat lunak serta membandingkan hasil dari kedua teknik tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang, rumusan masalah pada penelitian ini adalah sebagai berikut.

1. Bagaimana cara kerja teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* untuk *privacy preserving data mining*?
2. Bagaimana implementasi dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* pada perangkat lunak?
3. Bagaimana perbandingan antara hasil dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*?

1.3 Tujuan

Berdasarkan rumusan masalah, maka tujuan dari penelitian ini adalah sebagai berikut.

1. Mempelajari cara kerja dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* untuk *privacy preserving data mining*
2. Mengimplementasikan teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* pada perangkat lunak
3. Melakukan analisis dan pengujian untuk membandingkan dan mengukur hasil dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*

1.4 Batasan Masalah

Batasan-batasan masalah untuk penelitian ini adalah sebagai berikut.

1. «TODO»

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet

odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

1.5 Metodologi

Metodologi yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Melakukan studi literatur dasar-dasar privasi data
2. Melakukan studi literatur teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* untuk *privacy preserving data mining*
3. Melakukan studi literatur teknik penambangan data yang akan digunakan
4. Melakukan analisis terhadap teknik *Random Rotation Perturbation* dan *Random Projection Perturbation* serta bagaimana penerapannya dengan teknik penambangan data yang akan digunakan
5. Melakukan perancangan perangkat lunak yang mengimplementasikan teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*
6. Membangun perangkat lunak yang mengimplementasikan teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*
7. Menguji perangkat lunak secara fungsional dan eksperimental dengan menggunakan *real data*
8. Menerapkan teknik penambangan data terhadap data yang telah diproses untuk menganalisis hasil dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*
9. Melakukan analisis dan pengujian untuk membandingkan dan mengukur hasil dari teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*
10. Menarik kesimpulan berdasarkan hasil eksperiment yang telah dilakukan

1.6 Sistematika Pembahasan

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

- Bab 1 Pendahuluan
Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
- Bab 2 Dasar Teori
Berisi dasar teori tentang dasar-dasar privasi data, *Random Rotation Perturbation*, *Random Projection Perturbation*, *Random Rotation Perturbation*, dan teknik penambangan data.
- Bab 3 Analisis
Berisi analisis masalah, studi kasus, dan diagram aliran proses.
- Bab 4 Perancangan
Berisi perancangan perangkat lunak yang dibangun meliputi perancangan antarmuka dan diagram kelas yang lengkap.

- Bab 5 Implementasi dan Pengujian
Berisi implementasi antarmuka perangkat lunak, pengujian fungsional, pengujian eksperimental, dan kesimpulan dari pengujian.
- Bab 6 Kesimpulan dan Saran
Berisi kesimpulan dari awal hingga akhir penelitian dan saran untuk pengembangan selanjutnya.

BAB 2

DASAR TEORI

Dalam menjaga privasi data, perlu adanya definisi privasi yang konkret untuk menentukan data seperti apa yang menjadi privasi. Pada penambangan data, perlu ada teknik yang baik untuk menjaga privasi tidak tersebar kepada orang yang tidak berhak. Ada beberapa teknik untuk menjaga privasi pada penambangan data antara lain modifikasi data dengan metode randomisasi yaitu teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*

2.1 Privasi Data

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

2.1.1 Privasi

Dalam mendefinisikan privasi, sulit untuk mendapatkan definisi yang tepat untuk privasi karena setiap individu memiliki kepentingan yang berbeda-beda sehingga privasi pada setiap individu dapat berbeda-beda juga. Beberapa definisi privasi telah dikemukakan dan definisi tersebut bermacam-macam berdasarkan konteks, budaya, dan lingkungan [1]. Menurut Warren dan Brandeis pada papernya, mereka mendefinisikan privasi sebagai “*the right to be alone.*”, hak untuk menyendiri. Lalu pada papernya, Westin mendefinisikan privasi sebagai “*the desire of people to choose freely under what circumstances and to what extent they will expose themselves, their attitude, and their behavior to others*”, keinginan orang untuk memilih secara bebas dalam segala situasi dan dalam hal mengemukakan diri mereka, sikap mereka, dan tingkah laku mereka pada orang lain.

Schoeman mendefinisikan privasi sebagai “*the right to determine what (personal) information is communicated to others*”, hak untuk menentukan informasi pribadi apa saja yang dikomunikasikan kepada yang lain, atau “*the control an individual has over information about himself or herself.*”, kendali seorang individu terhadap informasi tentang dirinya sendiri. Lalu baru-baru ini, Garfinkel menyatakan bahwa “*privacy is about self-possession, autonomy, and integrity.*”, privasi adalah tentang penguasaan diri sendiri, otonomi, dan integritas. Di samping itu, Rosenberg berpendapat bahwa privasi sebenarnya bukan sebuah hak tetapi sebuah rasa: “*If privacy is in the end a matter of individual taste, then seeking a moral foundation for it – beyond its role in making social institutions possible that we happen to prize – will be no more fruitful than seeking a moral foundation for the taste for truffles.*”, intinya setiap orang memiliki perhatian yang berbeda-beda terhadap privasi mereka sendiri sehingga hal tersebut tergantung apa yang dirasakan oleh setiap individu.

Dari definisi-definisi privasi yang telah disebutkan di atas, dapat disimpulkan bahwa privasi dilihat sebagai konsep sosial dan budaya [1]. Konsep privasi pada suatu lingkungan dapat berbeda dari lingkungan lainnya dan hal ini menyebabkan sulitnya menentukan apakah sebuah data termasuk privasi atau bukan. Oleh karena itu, perlu adanya sebuah standar privasi untuk menentukan data mana yang dapat disebut sebuah privasi. Organisasi National Institute of Standards and Technology

dari Amerika Serikat, membuat standar mereka sendiri untuk menentukan informasi seperti apa yang dapat disebut sebagai privasi. Mereka mengemukakan konsep *Personally Identifiable Information* sebagai informasi yang dapat dikatakan personal untuk setiap individu.

2.1.2 *Personally Identifiable Information*

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

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

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

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

2.2 Penambangan Data

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

Penambangan data adalah proses mengekstrak sebuah pola atau sebuah pengetahuan dari kumpulan data yang besar, yang mana dapat direpresentasikan dan diinterpretasikan [3]. Pada penambangan data, teknik *machine learning* dan *pattern recognition* intensif digunakan untuk mendapatkan pola maupun pengetahuan baru dari data. Tujuan utama dari penambangan data adalah untuk membentuk model deskriptif dan prediktif dari suatu data. Model deskriptif berusaha untuk mengubah pola-pola yang ada pada data menjadi deskripsi yang dapat dimengerti oleh orang awam. Sedangkan model prediktif digunakan untuk memprediksi data yang tidak diketahui atau data yang berpotensi muncul di kemudian hari.

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

2.2.1 Classification

Tujuan utama *Classification* (klasifikasi) adalah membuat model yang dalam kasus ini disebut *classifier* yang mana dapat mengidentifikasi nilai kelas dari suatu data [3]. Dalam kata lain, sebuah *classifier* dibuat dari sebuah *training set* dan model ini digunakan untuk mengklasifikasi data tidak diketahui ke dalam salah satu kelas.

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

Teknik *machine learning* yang paling dikenal untuk klasifikasi antara lain *K-nearest Neighbors*, *Decision Tree*, dan *Naive Bayes*. Dalam penelitian ini, hanya teknik *K-nearest Neighbors* yang digunakan untuk pengujian sehingga berikutnya hanya akan dijelaskan teknik *K-nearest Neighbors* saja. Teknik *K-nearest Neighbors* adalah teknik penambangan data klasifikasi yang mencari label terbanyak pada sejumlah tetangga terdekatnya. Teknik ini bergantung pada jarak Euclidean antara titik yang mana adalah data yang akan diprediksi dengan tetangga-tetangganya. Setiap rekord pada data dipetakan ke bidang Euclidean dengan beberapa atribut yang menentukan letaknya pada bidang Euclidean [4].

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

1. Tentukan nilai k yang menentukan seberapa banyak tetangga yang digunakan
2. Lakukan perulangan dengan iterasi sebanyak rekord yang ada selain rekord yang ingin diprediksi labelnya
 - (a) Hitung jarak Euclidean antara rekord iterasi sekarang dengan rekord yang ingin diprediksi labelnya
 - (b) Catat jarak Euclidean dari rekord yang ingin diprediksi dan indeks rekord iterasi sekarang
3. Urutkan jarak Euclidean titik-titik yang sudah dihitung pada perulangan pada langkah sebelumnya secara menaik
4. Pilih rekord teratas (jarak Euclidean yang paling kecil) sebanyak k dari urutan pada langkah sebelumnya
5. Ambil label dari semua rekord yang terpilih pada langkah sebelumnya. Label terbanyak adalah hasil prediksi label pada rekord yang ingin diprediksi

Dalam mengevaluasi model klasifikasi dapat dihitung akurasi model dalam memprediksi suata data. Akurasi pada model klasifikasi berupa persentase jumlah prediksi yang benar dengan jumlah data yang diprediksi [4]. Misalkan apabila model klasifikasi menebak dengan benar 8 buah data dari total data berjumlah 10 data, berarti model tersebut memiliki akurasi sebesar 80%.

2.2.2 Clustering

Clustering adalah proses mengelompokan kumpulan objek ke dalam sebuah kelompok (*cluster*) sedemikian rupa sehingga objek-objek dari suatu *cluster* memiliki lebih banyak kemiripan dari pada objek-objek dari *cluster* lainnya [3].

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

Berikut langkah kerja dari teknik *K-means*. [4]

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

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

Sum of Squared Error adalah jumlah dari kuadrat jarak Euclidean setiap objek data terhadap *centroid* pada masing-masing *clusternya* [4]. Rumus untuk menghitung *Sum of Squared Error* dapat dilihat pada Persamaan 2.1.

$$E = \sum_{i=1}^k \sum_{p \in C_i} dist(\mathbf{p}, \mathbf{c}_i)^2 \quad (2.1)$$

Variabel \mathbf{p} adalah titik pada bidang Euclidean yang merepresentasikan sebuah objek data. Variabel \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 2.2. [4]

$$s(\mathbf{o}) = \frac{b(\mathbf{o}) - a(\mathbf{o})}{\max \{a(\mathbf{o}), b(\mathbf{o})\}} \quad (2.2)$$

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 2.3.

$$a(\mathbf{o}) = \frac{\sum_{\mathbf{o}' \in C_i, \mathbf{o} \neq \mathbf{o}'} dist(\mathbf{o}, \mathbf{o}')}{|C_i| - 1} \quad (2.3)$$

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 2.4.

$$b(\mathbf{o}) = \min_{C_j: 1 \leq j \leq k, j \neq i} \left\{ \frac{\sum_{\mathbf{o}' \in C_j} dist(\mathbf{o}, \mathbf{o}')}{|C_j|} \right\} \quad (2.4)$$

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 [5]. 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 [5]. *Perturbation Approach* adalah pendekatan untuk *privacy preserving data mining* dengan cara mengacaukan data yang ada, tetapi hasil data yang dikacaukan masih tetap dapat ditambang. Sedangkan pada *Anonymization Approach*, data diterapkan de-identifikasi di mana dataset mentah disebarluaskan setelah menghapus inti dari identitas setiap rekord [5].

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 ditambang. Oleh karena itu, properti yang terjaga pun sebaiknya berdasarkan properti yang digunakan pada teknik penambangan data yang digunakan [6]. 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.

2.4 Metode Randomization

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

Berbagai macam teknik dengan metode randomisasi umumnya menerapkan perusakan nilai

```

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

```

Gambar 2.1: Algoritma rekonstruksi

pada data. Salah satu teknik yang pertama kali menggunakan metode randomisasi untuk *privacy preserving data mining* adalah teknik *Random Noise Addition* yang dikemukakan oleh Agrawal dan Srikant pada paper berikut [7]. Teknik *Random Noise Addition* ini dilakukan dengan cara menambahkan nilai random (*noise*) pada data. Nilai random tersebut diambil dari sebuah distribusi. Untuk menambah data yang telah ditambahkan *noise* ini perlu dilakukan rekonsruksi distribusi untuk mendapatkan distribusi yang asli. Oleh karena itu, teknik *Random Noise Addition* ini hanya menjaga distribusi data asli sehingga hanya teknik penambangan data yang bergantung pada distribusi data saja yang dapat digunakan. Penyesuaian pada algoritma penambangan data yang digunakan pun perlu dilakukan agar teknik *Random Noise Addition* ini dapat digunakan dan mendapatkan hasil penambangan data yang hampir sama dengan tanpa menggunakan teknik *Random Noise Addition*.

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

2.4.1 Random Noise Addition

Ide utama dari teknik *Random Noise Addition* [7] adalah mendistorsi nilai pada data dengan cara menambahkan *random noise* yang diambil dari distribusi *Uniform* atau *Gaussian* dan memiliki rata-rata bernilai 0. Tetapi menurut penelitian yang telah dilakukan, distribusi *Gaussian* lebih baik digunakan untuk teknik ini. *Random noise* yang digunakan memiliki nilai yang berbeda untuk setiap nilai pada data.

Dengan teknik *Random Noise Addition*, dari data yang sudah didistorsi bisa didapatkan kembali distribusi data asli dengan merekonstruksi distribusinya tanpa mendapatkan setiap nilai-nilai yang ada pada data asli. Metode rekonsruksi yang digunakan berdasarkan pada aturan *Bayes*. Algoritma rekonsruksi untuk mendapatkan distribusi dari data asli dapat dilihat pada Gambar 2.1.

Algoritma ini berhenti sampai kriteria berhentinya terpenuhi. Kriteria tersebut adalah perbedaan estimasi distribusi iterasi sekarang dengan yang sebelumnya sangat kecil. Algoritma ini akan menghasilkan estimasi distribusi data asli dengan menggunakan data yang telah terdistorsi tanpa menggunakan nilai-nilai pada data asli, sehingga nilai-nilai pada data asli tidak tersebar. Oleh karena teknik *Random Noise Addition* hanya menjaga distribusi pada data maka teknik penambangan data yang dapat digunakan hanya teknik-teknik yang bergantung pada distribusi data saja.

Modifikasi pada algoritma penambangan data yang digunakan pun perlu dilakukan. Contohnya

apabila algoritma pohon keputusan digunakan, maka perlu modifikasi pada algoritma pohon keputusan tersebut. Hal ini menimbulkan masalah pada aplikasi pada dunia nyata karena tidak efisien dan memakan waktu untuk memodifikasi setiap algoritma yang ingin digunakan untuk menyesuaikan dengan teknik *Random Noise Addition*. Masalah mengenai algoritma yang dapat digunakan pun menjadi perhatian karena teknik *Random Noise Addition* hanya dapat digunakan untuk algoritma yang bergantung pada distribusi saja sedangkan teknik randomisasi lain tidak menjaga distribusi pada data. Ada juga penelitian yang mengatakan bahwa teknik *Random Noise Addition* ini memiliki kualitas yang kurang baik dalam menjaga privasi data karena banyaknya celah yang dapat diserang pada teknik ini. Oleh karena masalah-masalah tersebut, akhirnya teknik ini pun tidak akan digunakan untuk diuji kualitas hasilnya. Teknik *Random Projection Perturbation* akan digunakan untuk menggantikan teknik *Random Noise Addition*.

2.4.2 Random Rotation Perturbation

Ide utama dari teknik *Random Rotation Perturbation* adalah jika data direpresentasikan sebagai matrix $X_{n \times d}$, *rotation perturbation* dari dataset X didefinisikan sebagai berikut.

$$G(X) = X_{n \times d} R_{d \times d} \quad (2.5)$$

Dimana $R_{d \times d}$ adalah *random rotation matrix*. *Random rotation matrix* berukuran d dimensi dapat dibuat dengan cara membuat matriks *special orthogonal* acak karena matriks rotasi memiliki sifat *special orthogonal*. Matriks *special orthogonal* adalah matriks yang memiliki sifat *orthogonal* dan determinannya bernilai +1, yang mana matriks *orthogonal* adalah matriks yang menghasilkan matriks identitas apabila dikalikan dengan transposenya sendiri. Matriks rotasi ini dapat dibuat secara efisien mengikuti distribusi Haar [8]. Dari definisi di atas dapat disimpulkan transformasi rotasi tersebut menjaga jarak Euclidean [6].

Teknik ini menjaga beberapa properti pada data antara lain yaitu jarak Euclidean, *inner product*, dan *geometric shape hyper* pada bidang multi-dimensi [5]. Oleh karena itu, beberapa teknik penambangan data tidak berpengaruh (dapat digunakan) terhadap teknik *Random Rotation Perturbation* antara lain yaitu *K-nearest Neighbors*, *Support Vector Machines*, dan *Perceptrons* [6]. Teknik ini dipercaya dapat memberikan hasil penambangan yang maksimal, hasil penambangan data yang telah dirusak persis sama dengan hasil penambangan data aslinya. Sehingga jumlah informasi yang hilang tidak ada, tetapi jumlah privasi yang hilangnya tinggi. Walaupun demikian ada beberapa penelitian yang mengatakan bahwa karena teknik *Random Rotation Perturbation* ini memiliki sifat demikian sehingga teknik ini dikatakan tidak aman dan dapat diserang dengan beberapa teknik untuk mendapatkan data asli yang lengkap.

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

Algoritma

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

1. Dataset yang memiliki atribut sebanyak d dan rekord sebanyak n direpresentasikan dalam bentuk matriks berukuran $n \times d$
2. Buatlah matriks translasi acak yang diambil mengikuti distribusi *uniform* dengan rentang $[0, 100]$ berdimensi $(d + 1) \times (d + 1)$

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

2.4.3 Random Projection Perturbation

Ide utama dari teknik *Random Projection Perturbation* adalah mereduksi dimensi dari representasi matriks data asli dengan syarat dimensi matriks tersebut cukup besar. Dasar dari teknik *Random Projection Perturbation* berdiri pada *Johnson-Lindenstrauss Lemma*. [9]

Lemma 1 (JOHNSON-LINDENSTRAUSS LEMMA). *For any $0 < \epsilon < 1$ and any integer s , let 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 points in \mathbb{R}^m , there is a map $f : \mathbb{R}^m \rightarrow \mathbb{R}^k$ such that, for all $x, y \in S$, $(1 - \epsilon)s||u - v||^2 < ||p(u) - p(v)||^2 < (1 + \epsilon)s||u - v||^2$, where $||\cdot||$ denotes the vector 2-norm.*

Inti dari Lemma ini menunjukkan bahwa titik pada bidang Euclidean d -dimensi dapat diproyeksikan ke bidang Euclidean berdimensi lebih kecil dari d tetapi jarak Euclidean antara dua titik tetap terjaga dengan distorsi yang terkontrol tetapi dengan syarat d harus cukup besar. Oleh karena adanya distorsi yang muncul, properti-properti pada data pun relatif sedikit berubah dan hal ini menyebabkan akurasi pada model yang dibuat dengan data tersebut berkurang dibandingkan data aslinya. [10]

Projection perturbation dari dataset X didefinisikan sebagai berikut.

$$G(X) = X_{n \times d} R_{d \times k} \quad (2.6)$$

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

Jika *random projection matrix* yang digunakan dihasilkan secara acak saja, hasil dari *random projection perturbation* akan terlalu merusak nilai pada data sehingga akurasi pada model yang akan dibuat kemungkinan berkurang drastis. Cara menanggulangi hal tersebut adalah menggunakan matriks *orthogonal* sebagai *random projection matrix*. Tetapi membuat matriks *orthogonal* yang berdimensi tinggi memiliki kompleksitas yang tinggi sehingga memerlukan *cost* yang besar. Pada observasi yang dilakukan Hecht-Neilsen menunjukkan bahwa “*that in a high-dimensional space, vectors with random directions are almost orthogonal*” [11]. Dapat disimpulkan bahwa dalam kasus matriks berdimensi tinggi apabila sebuah matriks dihasilkan secara acak mengikuti suatu distribusi,

matriks tersebut akan kurang lebih hampir *orthogonal*. Oleh karena itu, matriks yang dibuat untuk *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.

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \quad (2.7)$$

Sebuah matriks yang akan diproyeksikan ke dimensi yang lebih kecil akan memiliki distorsi pada jarak Euclidean yang dimiliki oleh titik-titik (setiap elemen dari matriks) pada bidang Euclidean tersebut. Distorsi tersebut ditentukan oleh variabel ϵ , yang mana ϵ menjadi ukuran seberapa baik proyeksi dilakukan. Semakin kecil nilai ϵ maka semakin besar k , yang mana k adalah dimensi minimal matriks yang dihasilkan. Semakin titik-titik pada bidang Euclidean diproyeksikan ke 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 ϵ (eps) yang ditentukan berada pada rentang $(0, 1)$.

$$(1 - \text{eps})\|u - v\|^2 < \|p(u) - p(v)\|^2 < (1 + \text{eps})\|u - v\|^2 \quad (2.8)$$

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

Algoritma

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

1. Dataset yang memiliki atribut sebanyak d dan rekord sebanyak n direpresentasikan dalam bentuk matriks berukuran $n \times d$
2. Tentukan nilai variabel ϵ (epsilon) yang diinginkan dan berada pada rentang $(0, 1)$
3. Hitung nilai minimal variabel k (dimensi minimal) dengan rumus berikut $k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n$
4. Tentukan nilai variabel k yang diinginkan dengan memenuhi persyaratan pada langkah ketiga dan nilai variabel k yang dipilih harus lebih kecil dari d (dimensi dataset aslinya)
5. Buatlah 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 $1/\sqrt{k}$
6. Lakukan proyeksi dengan cara mengkalikan matriks dataset dengan matriks proyeksi yang telah dibuat pada langkah kelima
7. Hasil matriks yang telah diproyeksi sudah dapat langsung digunakan untuk penambangan data

2.5 Bahasa Pemograman Python Beserta *Library*-nya

Bahasa pemograman Python [12] adalah bahasa pemograman yang sudah umum digunakan untuk melakukan penambangan data. Oleh karena itu, komunitas pada bahasa pemograman Python sudah sangat besar dan banyak pihak yang membuat *library* dan *tools* yang mendukung dalam melakukan penambangan data pada bahasa pemograman ini. Tetapi bahasa pemograman Python tidak semata-mata hanya dapat digunakan untuk penambangan data, Python merupakan bahasa pemograman yang multifungsi dan banyak juga digunakan pada bidang lain selain penambangan data. Bahasa pemograman Python dapat juga digunakan untuk membuat antarmuka dengan dukungan *framework* tertentu. Selain itu, Python juga merupakan bahasa pemograman yang

berorientasi objek. Berikut akan dijelaskan beberapa *library*, *framework*, dan *tools* yang berguna untuk membuat antarmuka perangkat lunak, mendukung implementasi metode *Randomization*, dan mendukung proses penambangan data.

- Kivy¹ adalah *framework* yang berfungsi untuk membuat antarmuka perangkat lunak. Kivy dapat membuat antarmuka perangkat lunak pada berbagai *platform* seperti Windows, OS X, Android, iOS, dan Raspberry Pi.
- Numpy dan Scipy [13] adalah *library* yang berfungsi untuk memanipulasi struktur data matriks atau melakukan operasi matematis yang berkaitan struktur data *array* atau matriks. Contohnya yang dipakai pada penelitian ini adalah membuat matriks *orthogonal* secara acak.
- Pandas [14] adalah *library* yang berfungsi untuk menganalisis dan memanipulasi data berbentuk struktur data yang lebih kompleks seperti *dataframe*. Pandas juga menyediakan fungsi untuk membaca dokumen dan mengkonversinya ke struktur data *dataframe* dan sebaliknya.
- Scikit-learn [15] adalah *library* yang berfungsi untuk melakukan proses pembelajaran mesin dan memiliki berbagai macam fungsi pembelajaran mesin yang dapat digunakan untuk penambangan data seperti teknik *k-nearest neighbors* dan *k-means*.
- Matplotlib [16], Plotly², dan Seaborn³ adalah *library* yang berfungsi untuk melakukan berbagai macam visualisasi dengan mudah. Contohnya yang dipakai pada penelitian ini adalah visualisasi dengan *scatter plot* dan *line plot*.
- Spyder⁴ adalah sebuah perangkat lunak *integrated development environment* (IDE) yang berfungsi sebagai editor dan *compiler* bahasa pemrograman Python serta berfungsi untuk menampilkan berbagai macam visualisasi. Perangkat lunak ini sudah umum digunakan untuk penambangan data karena kemudahannya.
- Anaconda⁵ 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.

¹<https://kivy.org/>

²<https://plotly.com/>

³<https://seaborn.pydata.org/>

⁴<https://www.spyder-ide.org/>

⁵<https://www.anaconda.com/>

BAB 3

ANALISIS

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

3.1 Analisis Masalah

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

3.1.1 Implementasi Metode *Randomization*

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

Perangkat lunak diharapkan dapat berfungsi untuk menerapkan teknik *Random Rotation Perturbation* terhadap dataset apapun dengan syarat seluruh fiturnya bersifat numerik dan menerapkan teknik *Random Projection Perturbation* terhadap dataset yang memenuhi persyaratan yaitu dimensinya 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 randomisasi akan dibangun beserta antarmukanya dengan bantuan *framework* antarmuka yaitu Kivy.

Teknik *Random Rotation Perturbation*

Teknik *Random Rotation Perturbation* menggunakan matriks orthogonal untuk mengacak dataset tanpa mengubah jarak Euclidean dataset tersebut. Matriks orthogonal relatif sulit dibuat pada dimensi yang tinggi dan kompleksitasnya tidak kecil yaitu $O(n^2)$ [8]. Untuk dataset yang sangat besar, masih dapat dilakukan randomisasi dengan teknik ini walaupun mungkin akan sedikit memakan waktu. Hal ini akan diuji pada bab 5 saat pengujian eksperimental.

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 data yang direpresentasikan sebagai titik 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 [6]. 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.

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. *Error* terburuk yang dapat terjadi pada jarak Euclidean setelah proyeksi diterapkan ditentukan oleh nilai variabel *epsilon* dengan pertidaksamaan sebagai berikut menunjukkan rentang jarak Euclidean data setelah diproyeksi.

$$(1 - \epsilon) \|u - v\|^2 < \|p(u) - p(v)\|^2 < (1 + \epsilon) \|u - v\|^2 \quad (3.1)$$

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

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

Variabel n pada persamaan tersebut adalah jumlah titik pada bidang Euclidean atau dengan kata lain jumlah baris pada dataset yang ingin dirandomisasi. Dengan melihat persamaan tersebut maka bisa disimpulkan bahwa dimensi minimal sebuah dataset diproyeksikan berbanding lurus dengan jumlah baris pada dataset tersebut. Apabila sebuah dataset diproyeksikan ke dimensi paling minimal dan diterapkan teknik penambangan data sehingga menghasilkan sebuah model penambangan data dengan data latihannya adalah dataset tersebut maka model tersebut akan

menggar persamaan di atas jika model tersebut dilatih kembali dengan data yang baru karena artinya bidang Euclidean yang disebutkan tadi akan memiliki titik yang lebih banyak, nilai n yang lebih besar.

Oleh karena itu, berdasarkan analisis di atas teknik *Random Projection Perturbation* tidak relevan untuk model penambangan data yang memiliki sangat banyak data tetapi berdimensi relatif kecil dibandingkan dengan jumlah datanya. Tetapi masalah ini dapat dihindari dengan tidak mereduksi dimensi ke dimensi yang sangat kecil atau paling minimal sehingga tidak terlalu cepat nilai minimal variabel k meningkat sampai menyul dimensi dataset setelah proyeksi. Selain itu, jarak Euclidean setelah diproyeksi juga belum tentu pasti menyentuh batas pertidaksamaan yang di atas sehingga belum tentu hasil randomisasi memiliki distorsi yang sangat besar apabila besar dimensi melebihi nilai minimal variabel k . Hal ini akan diuji pada bab 5 saat melakukan pengujian eksperimental.

Teknik *Random Projection Perturbation* memiliki kompleksitas yang relatif kecil yaitu $O(dkn)$ [11]. Dengan kompleksitas yang cukup rendah maka teknik ini dapat bekerja secara cepat untuk penambangan data dengan dataset yang besar sekalipun. Oleh karena itu, teknik ini berpotensi dapat berkerja dengan baik dalam lingkungan *big data* karena kompleksitas yang relatif kecil dan dapat bekerja pada dataset yang sangat besar. Hal ini disebabkan oleh metode yang digunakan oleh teknik ini adalah melakukan proyeksi dengan cara mengkalikan dataset yang sudah berbentuk matriks dengan matriks proyeksi yang berupa matriks acak saja, tidak ada sifat spesial pada matriks acak tersebut seperti teknik *Random Rotation Perturbation* sehingga pembuatan matriks proyeksinya dapat dilakukan dengan cepat.

3.1.2 Pengujian Dengan Penambangan Data

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

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

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

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

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

1. Membuat 20 (dapat berubah disesuaikan dengan pengujian yang dilakukan) model *k-nearest neighbors* dengan nilai variabel *k* sebesar 1 sampai 20 untuk mengetahui model dengan nilai variabel *k* mana yang memiliki akurasi tertinggi
2. Melatih 20 model tersebut dengan *training set* (60 persen baris pada dataset diambil secara acak) yang sama
3. Melakukan prediksi dengan 20 model tersebut terhadap *training set* yang sama
4. Melakukan prediksi dengan 20 model tersebut terhadap *test set* (40 persen baris pada dataset diambil secara acak) yang sama
5. Menghitung akurasi 20 model tersebut pada hasil prediksi *training set* dan *test set*
6. Membuat grafik hubungan antara nilai variabel *k* dengan akurasi pada *training set* dan *test set*
7. Menampilkan seluruh nilai akurasi pada setiap nilai variabel *k*
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
9. Menampilkan akurasi model, waktu eksekusi untuk melatih model, dan waktu eksekusi dalam melakukan prediksi dengan model tersebut

Proses penambangan data tersebut dilakukan dua kali masing-masing menggunakan dataset asli dan dataset yang telah dirandomisasi 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.

Pengujian Dengan Teknik Penambangan Data *Clustering K-means*

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 dirandomisasi. Pada pengujian yang akan dilakukan, perbandingan antara dataset asli dengan dataset yang telah dirandomisasi 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 dirandomisasi. Apabila jumlah *cluster*-nya sama dan anggota-anggota tiap *cluster* sama atau mirip, maka teknik randomisasi yang digunakan dapat dinyatakan berhasil dan efektif digunakan untuk *privacy preserving data mining* dengan teknik penambangan data *clustering* menggunakan algoritma *k-means*.

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*.

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

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
2. Melatih 9 model tersebut dengan dataset yang sama
3. Menghitung nilai *Sum of Squared Error* dan *Silhouette Score* pada 9 model tersebut
4. Membuat grafik hubungan antara nilai variabel *k* dengan nilai *Sum of Squared Error*
5. Membuat grafik hubungan antara nilai variabel *k* dengan nilai *Silhouette Score*
6. Menampilkan seluruh nilai *Sum of Squared Error* pada setiap nilai variabel *k*
7. Menampilkan seluruh nilai *Silhouette Score* pada setiap nilai variabel *k*
8. Menampilkan nilai *Silhouette Score* tertinggi dan nilai variabel *k*-nya
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. Menampilkan visualisasi *cluster* pada model dengan *Scatter Plot*, *Silhouette Score* model, dan waktu eksekusi untuk melatih model tersebut

Proses penambangan data tersebut dilakukan dua kali masing-masing menggunakan dataset asli dan dataset yang telah dirandomisasi agar dapat dibandingkan visualisasi *cluster* pada model yang terbaik, nilai variabel *k* yang dimiliki oleh model yang terbaik, dan informasi lainnya yaitu *Sum of Squared Error*, *Silhouette Score*, dan waktu pelatihan model. Dalam membandingkan antara model *clustering* yang dilatih dengan dataset asli dan dataset yang telah dirandomisasi, metode *Adjusted Rand Index* digunakan untuk mengukur kemiripan antara kedua model *clustering* tersebut. Dengan menggunakan metode *Adjusted Rand Index* dapat diketahui apakah metode *Randomization* yang digunakan mempunyai kualitas yang baik atau tidak untuk digunakan dalam penambangan data *clustering* dengan teknik *k-means*. Kedua model *clustering* yang dibandingkan dapat dinyatakan sama atau mirip apabila nilai *Adjusted Rand Index* pada kedua model *clustering* tersebut mendekati angka 1.

3.2 Studi Kasus

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

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

3.2.1 Random Rotation Perturbation

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

- Fitur-fitur pada dataset tersebut yang berbentuk tabel akan direpresentasikan sebagai matriks. Labelnya tidak diikutsertakan

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

- Membuat matriks translasi yang diambil mengikuti distribusi *uniform* dengan rentang [0,100] dengan dimensi sesuai dimensi matriks dataset

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 71.35281261 & 93.96479736 & 77.16763568 & 27.88189356 & 1 \end{bmatrix}_{5 \times 5}$$

- Untuk keperluan translasi, matriks dataset ditambahkan sebuah kolom dengan nilai 1 pada setiap barisnya

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

- Dilakukan transformasi translasi dengan matriks translasi yang telah dibuat pada langkah

sebelumnya dengan cara mengkalikan matriks dataset dengan matriks translasi

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 & 1 \\ 4.9 & 3 & 1.4 & 0.2 & 1 \\ 4.7 & 3.2 & 1.3 & 0.2 & 1 \\ 4.6 & 3.1 & 1.5 & 0.2 & 1 \\ 5 & 3.6 & 1.4 & 0.2 & 1 \\ 5.4 & 3.9 & 1.7 & 0.4 & 1 \\ 4.6 & 3.4 & 1.4 & 0.3 & 1 \\ 5 & 3.4 & 1.5 & 0.2 & 1 \\ 4.4 & 2.9 & 1.4 & 0.2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 71.35281261 & 93.96479736 & 77.16763568 & 27.88189356 & 1 \end{bmatrix}$$

5. Berikut adalah hasil translasi pada matriks dataset. Dapat dilihat kolom terakhir adalah kolom yang harus dibuang karena kolom tersebut ada hanya untuk melakukan transformasi translasi yang sudah dilakukan pada langkah sebelumnya

$$\begin{bmatrix} 76.45281261 & 97.46479736 & 78.56763568 & 28.08189356 & 1 \\ 76.25281261 & 96.96479736 & 78.56763568 & 28.08189356 & 1 \\ 76.05281261 & 97.16479736 & 78.46763568 & 28.08189356 & 1 \\ 75.95281261 & 97.06479736 & 78.66763568 & 28.08189356 & 1 \\ 76.35281261 & 97.56479736 & 78.56763568 & 28.08189356 & 1 \\ 76.75281261 & 97.86479736 & 78.86763568 & 28.28189356 & 1 \\ 75.95281261 & 97.36479736 & 78.56763568 & 28.18189356 & 1 \\ 76.35281261 & 97.36479736 & 78.66763568 & 28.08189356 & 1 \\ 75.75281261 & 96.86479736 & 78.56763568 & 28.08189356 & 1 \end{bmatrix}_{9 \times 5}$$

6. Berikutnya matriks rotasi dibuat dengan cara membuat matriks spesial orthogonal yang berdimensi sesuai dimensi matriks dataset. Matriks rotasi berikut dibuat dengan menggunakan *library Scipy*¹ pada bahasa pemrograman Python

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

7. Dilakukan transformasi rotasi dengan matriks rotasi yang telah dibuat pada langkah sebelumnya dengan cara mengkalikan matriks dataset dengan matriks rotasi
8. Berikut adalah hasil rotasi pada matriks dataset. Hasil teknik *Random Rotation Perturbation* pada dataset *iris* ini sudah dapat langsung digunakan untuk dilakukan penambangan data

$$\begin{bmatrix} -70.13483265 & 20.05005561 & 4.11528068 & 130.26146931 \\ -69.8246321 & 19.83726437 & 3.85425381 & 129.97799007 \\ -69.80455936 & 20.11346248 & 3.95103051 & 129.91517319 \\ -69.75103816 & 20.12538172 & 3.71327762 & 129.93678284 \\ -70.13369505 & 20.19121015 & 4.12214059 & 130.25626898 \\ -70.34841122 & 20.14113559 & 4.16552936 & 130.83029591 \\ -69.78963253 & 20.33201179 & 3.93670924 & 130.06284949 \\ -70.06351391 & 20.05586388 & 3.96058467 & 130.23066274 \\ -69.55500808 & 20.11866536 & 3.65305621 & 129.71792106 \end{bmatrix}_{9 \times 4}$$

¹http://scipy.github.io/devdocs/generated/scipy.stats.special_ortho_group.html

3.2.2 Random Projection Perturbation

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*. Jumlah rekordnya adalah 1000 dan jumlah atributnya adalah 500.

Berikut langkah-langkah teknik *Random Projection Perturbation* yang diaplikasikan pada dataset *iris* pada Tabel 3.1.

- Fitur-fitur pada dataset tersebut yang berbentuk tabel akan direpresentasikan sebagai matriks. Labelnya tidak diikutsertakan

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

- Ditentukan nilai ϵ (*epsilon*) yang diinginkan adalah 0.5. Dengan rumus berikut dapat dihitung untuk melihat seberapa besar distorsi yang terjadi pada jarak Euclidean sebuah titik/baris data dengan menunjukkan rentang jarak Euclidean data setelah diproyeksi. Contoh yang akan diberikan berikut menguji rentang jarak Euclidean antara baris ke-6 dan ke-9 setelah diproyeksi

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

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

- Nilai minimal variabel k (dimensi minimal) dihitung dengan rumus berikut

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

- Nilai variabel k dipilih sesuai keinginan, dalam kasus ini dipilih nilai k sebesar 332. Untuk

keperluan kemudahan dalam melakukan studi kasus, nilai k yang digunakan adalah 3

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

$$\begin{bmatrix} 0.11483014 & -0.10167359 & 0.06652355 \\ 0.0638684 & -0.1499892 & 0.10146435 \\ -0.10429573 & 0.03839861 & 0.04955419 \\ -0.0315941 & -0.06905021 & -0.17782438 \end{bmatrix}_{4 \times 3}$$

6. Dilakukan proyeksi dengan cara mengkalikan matriks dataset dengan matriks proyeksi yang telah dibuat pada langkah sebelumnya

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & 3.1 & 1.5 & 0.2 \\ 5 & 3.6 & 1.4 & 0.2 \\ 5.4 & 3.9 & 1.7 & 0.4 \\ 4.6 & 3.4 & 1.4 & 0.3 \\ 5 & 3.4 & 1.5 & 0.2 \\ 4.4 & 2.9 & 1.4 & 0.2 \end{bmatrix} \times \begin{bmatrix} 0.11483014 & -0.10167359 & 0.06652355 \\ 0.0638684 & -0.1499892 & 0.10146435 \\ -0.10429573 & 0.03839861 & 0.04955419 \\ -0.0315941 & -0.06905021 & -0.17782438 \end{bmatrix}$$

7. Berikut adalah hasil matriks yang telah diproyeksi. Hasil dari teknik *Random Projection Perturbation* pada dataset *iris* ini sudah dapat langsung digunakan untuk dilakukan penambangan data

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

3.3 Gambaran Umum Perangkat Lunak

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

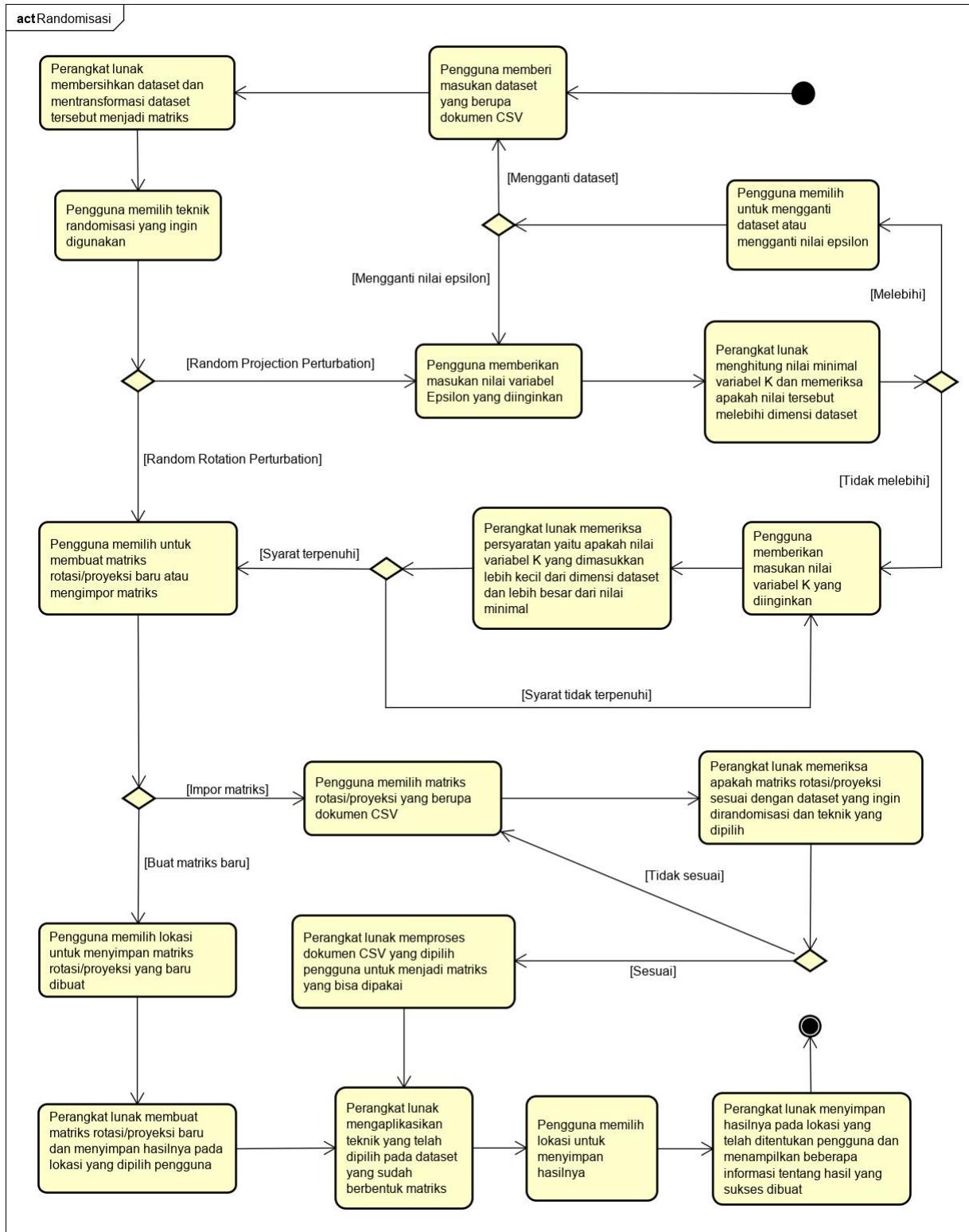
Perangkat lunak yang mengimplementasikan metode *Randomization* akan mempunyai sebuah masukan yaitu dataset berbentuk dokumen berjenis *comma-separated values*. Lalu perangkat lunak akan mengimplementasikan metode *Randomization* terhadap dataset tersebut dan hasil keluarannya adalah dataset yang telah dirandomisasi berbentuk dokumen berjenis *comma-separated values*.

Pada subbab ini akan ditunjukkan gambaran umum perangkat lunak yang akan dijelaskan dengan diagram aktivitas dan diagram kelas.

3.3.1 Diagram Aktivitas

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

1. Pengguna memberikan masukan berupa dataset yang berupa dokumen berjenis *comma-separated values* (CSV). Dokumen ini harus berisi tiap rekord pada barisnya dan tiap fitur pada kolomnya. Selain itu, pada baris pertama dalam dokumen tersebut harus berupa nama setiap fitur pada setiap kolomnya
2. Perangkat lunak akan membersihkan dokumen yang berisi dataset tersebut dan mentransformasi datasetnya menjadi sebuah matriks. Matriks tersebut akan berisi nilai-nilai pada dataset saja tanpa nama kolom
3. Pengguna memilih teknik randomisasi yang ingin digunakan antara *Random Rotation Perturbation* dan *Random Projection Perturbation*. Jika *Random Projection Perturbation* yang dipilih maka akan ada beberapa langkah yang harus dipenuhi yaitu sebagai berikut.
 - (a) Pengguna memberi masukan nilai variabel *epsilon* yang diinginkan. Variabel ini akan menentukan distorsi terburuk yang dapat terjadi pada hasil randomisasi dan menentukan nilai minimal variabel *k*
 - (b) Perangkat lunak menghitung nilai minimal variabel *k* lalu menampilkannya kepada pengguna dan memeriksa sebuah persyaratan yaitu apakah nilai minimal variabel *k* tersebut tidak melebihi dimensi dataset
 - (c) Jika persyaratan tidak terpenuhi maka pengguna harus memilih untuk mengganti datasetnya atau mengganti nilai variabel *epsilon*
 - (d) Jika persyaratan terpenuhi maka berikutnya pengguna memberikan masukan nilai variabel *k* yang diinginkan. Nilai yang diberikan pengguna harus lebih dari nilai minimal variabel *k* yang sudah dihitung oleh perangkat lunak pada langkah sebelumnya
 - (e) Perangkat lunak memeriksa persyaratan teknik yaitu nilai variabel *k* yang diberikan harus lebih kecil dari dimensi dataset dan lebih besar dari nilai minimal variabel *k*.
 - (f) Jika persyaratan tidak terpenuhi maka pengguna harus mengganti nilai variabel *k* kembali.
4. Pengguna memilih untuk membuat matriks rotasi/proyeksi baru sesuai teknik yang dipilih atau pengguna dapat mengimpor matriks yang telah dibuat dan disimpan sebelumnya
5. Berikut ini langkah-langkah tambahan yang harus dilakukan pengguna apabila pengguna memilih untuk membuat matriks rotasi/proyeksi baru
 - (a) Pengguna memilih lokasi direktori pada komputer pengguna sebagai lokasi untuk menyimpan matriks rotasi/proyeksi yang akan dibuat



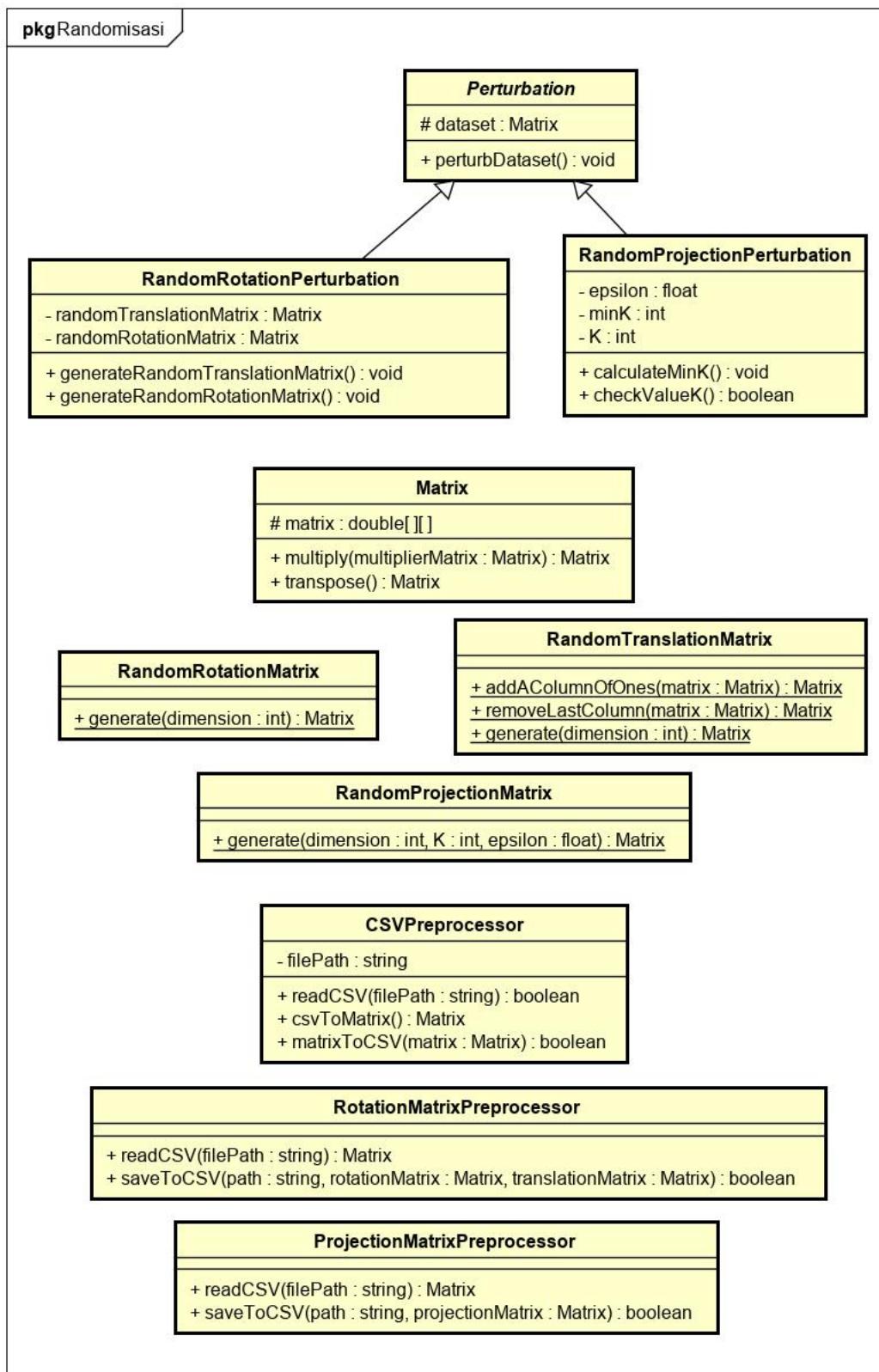
Gambar 3.1: Diagram aktivitas perangkat lunak randomisasi

- (b) Perangkat lunak membuat matriks rotasi/proyeksi baru dan menyimpan hasilnya pada lokasi yang telah dipilih pengguna dalam bentuk dokumen berjenis CSV
6. Berikut ini langkah-langkah tambahan yang harus dilakukan pengguna apabila pengguna memilih untuk mengimpor matriks rotasi/proyeksi yang pernah dibuat sebelumnya
- Pengguna memilih matriks rotasi atau proyeksi sesuai teknik yang dipilih berupa dokumen berjenis CSV pada sebuah direktori komputer pengguna yang dipilih
 - Perangkat lunak memeriksa dokumen CSV yang dipilih berisi matriks rotasi/proyeksi yang sesuai dengan dataset dan teknik randomisasi yang dipilih
 - Jika tidak sesuai maka pengguna harus memilih kembali matriks rotasi/proyeksi dengan benar
 - Jika sudah sesuai dan memenuhi persyaratan maka perangkat lunak akan memproses dokumen CSV yang dipilih pengguna untuk menjadi matriks rotasi/proyeksi yang dapat dipakai untuk teknik randomisasi yang dipilih
7. Perangkat lunak mengaplikasikan teknik yang telah dipilih pada dataset yang sudah berbentuk matriks dengan memakai matriks rotasi/proyeksi yang telah dipilih
8. Pengguna memilih lokasi direktori pada komputer pengguna untuk menyimpan hasil randomisasi dari teknik yang dipilih. Hasilnya adalah sebuah dokumen *comma-separated values* yang berisi dataset yang telah dirandomisasi
9. Perangkat lunak menyimpan hasilnya pada lokasi yang telah ditentukan pengguna dalam bentuk dokumen berjenis *comma-separated values* dan menampilkan beberapa informasi tentang hasil yang sukses dibuat

3.3.2 Diagram Kelas

Perancangan diagram kelas didasarkan pada analisis terhadap algoritma yang ingin diimplementasikan yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation*, serta berdasarkan pada diagram aktivitas yang telah dibuat, dan dengan mempertimbangkan studi literatur, analisis masalah, dan studi kasus yang telah dilakukan pada kedua algoritma randomisasi yang ingin diimplementasikan. Detail dari diagram kelas perangkat lunak randomisasi pada Gambar 3.2 adalah sebagai berikut.

- Kelas *Perturbation* adalah kelas abstrak yang akan di-*extend* oleh kelas yang mengimplementasikan algoritma *Random Rotation Perturbation* dan *Random Projection Perturbation*. Kelas ini mempunyai sebuah atribut yaitu *dataset* yang berfungsi untuk menyimpan dataset yang ingin dirandomisasi. Ada sebuah fungsi pada kelas ini yaitu *perturbDataset* yang berfungsi untuk menerapkan teknik randomisasi pada dataset yang ingin dirandomisasi dengan kata lain atribut *dataset*
- Kelas *RandomRotationPerturbation* adalah kelas turunan dari kelas *Perturbation* yang bertujuan untuk mengimplementasikan algoritma *Random Rotation Perturbation*. Kelas ini memiliki dua tambahan atribut yaitu *randomTranslationMatrix* dan *randomRotationMatrix* yang masing-masing memiliki fungsi untuk menyimpan matriks translasi dan matriks rotasi. Ada dua buah fungsi tambahan juga pada kelas ini yaitu *generateRandomTranslationMatrix* dan *generateRandomRotationMatrix* yang masing-masing memiliki fungsi untuk membuat matriks translasi dan matriks rotasi baru.
- Kelas *RandomProjectionPerturbation* adalah kelas turunan dari kelas *Perturbation* yang bertujuan untuk mengimplementasikan algoritma *Random Projection Perturbation*. Kelas ini memiliki tiga tambahan atribut yaitu *epsilon*, *minK*, dan *K* yang masing-masing memiliki



Gambar 3.2: Diagram kelas perangkat lunak randomisasi

fungsi untuk menyimpan nilai variabel *epsilon*, nilai minimal variabel *k*, dan nilai variabel *k* yang akan dipakai untuk menerapkan algoritma *Random Projection Perturbation*. Ada dua buah fungsi tambahan juga pada kelas ini yaitu *calculateMinK* dan *checkValueK* yang masing-masing memiliki fungsi untuk menghitung nilai minimal variabel *k* dan memeriksa persyaratan pada nilai atribut *K*.

- Kelas *Matrix* adalah kelas untuk merepresentasikan matriks dan menyimpan nilai-nilai pada setiap elemen matriks. Kelas ini memiliki sebuah atribut yaitu *matrix* yang berfungsi untuk menyimpan setiap elemen matriks yang ada. Kelas ini juga memiliki dua buah fungsi yaitu *multiply* dan *transpose* yang masing-masing berfungsi untuk melakukan operasi perkalian dan transpose matriks yang akan digunakan untuk implementasi kedua algoritma teknik randomisasi
- Kelas *RandomTranslationMatrix* adalah kelas yang mempunyai tujuan utama untuk membuat matriks translasi secara acak. Kelas ini adalah kelas statis sehingga tidak bisa diinstansiasi dan hanya memiliki atribut atau fungsi statis saja. Ada tiga buah fungsi statis yaitu *addAColumnOfOnes*, *removeLastColumn*, dan *generate*. Fungsi *addAColumnOfOnes* memiliki fungsi untuk menambahkan sebuah kolom yang pada setiap baris memiliki nilai berupa angka 1 kepada sebuah matriks. Fungsi *removeLastColumn* memiliki fungsi untuk menghapus kolom terakhir pada suatu matriks. Fungsi *generate* adalah fungsi yang bertujuan untuk membuat matriks translasi baru secara acak
- Kelas *RandomRotationMatrix* adalah kelas yang mempunyai tujuan utama untuk membuat matriks rotasi secara acak. Kelas ini adalah kelas statis sehingga tidak bisa diinstansiasi dan hanya memiliki atribut atau fungsi statis saja. Hanya ada sebuah fungsi pada kelas ini yaitu *generate* yang memiliki fungsi untuk membuat matriks rotasi baru secara acak
- Kelas *RandomProjectionMatrix* adalah kelas yang mempunyai tujuan utama untuk membuat matriks proyeksi secara acak. Kelas ini adalah kelas statis sehingga tidak bisa diinstansiasi dan hanya memiliki atribut atau fungsi statis saja. Hanya ada sebuah fungsi pada kelas ini yaitu *generate* yang memiliki fungsi untuk membuat matriks proyeksi baru secara acak
- Kelas *CSVPreprocessor* adalah kelas untuk menangani masukan dataset berupa dokumen *comma-separated values* yang akan direpresentasikan menjadi matriks. Kelas ini berguna untuk mengkonversi dokumen berjenis *comma-separated values* menjadi matriks dan sebaliknya. Hanya ada sebuah atribut pada kelas ini yaitu *filePath* yang berfungsi untuk menyimpan lokasi dokumen yang ingin diproses menjadi matriks. Ada tiga buah fungsi pada kelas ini yaitu *readCSV*, *csvToMatrix*, dan *matrixToCSV* yang masing-masing berfungsi untuk membaca dokumen berjenis *comma-separated values*, mengkonversi dokumen *comma-separated values* menjadi matriks, dan mengkonversi matriks menjadi dokumen *comma-separated values*.
- Kelas *RotationMatrixPreprocessor* adalah kelas untuk menangani masukan matriks rotasi yang digabung dengan matriks translasi berupa sebuah dokumen *comma-separated values* yang akan dikonversi menjadi dua buah matriks (rotasi dan translasi) dan sebaliknya. Ada dua buah fungsi pada kelas ini yaitu *readCSV* dan *saveToCSV*. Fungsi *readCSV* berfungsi untuk membaca dokumen berjenis *comma-separated values* untuk dikonversi menjadi matriks rotasi dan matriks translasi. Fungsi *SaveToCSV* berfungsi untuk mengkonversi matriks rotasi dan matriks translasi menjadi sebuah dokumen berjenis *comma-separated values*.
- Kelas *ProjectionMatrixPreprocessor* adalah kelas untuk menangani masukan matriks proyeksi berupa sebuah dokumen *comma-separated values* yang akan dikonversi menjadi sebuah matriks. Ada dua buah fungsi pada kelas ini yaitu *readCSV* dan *saveToCSV*. Fungsi *readCSV* berfungsi untuk membaca dokumen berjenis *comma-separated values* untuk dikonversi menjadi matriks proyeksi. Fungsi *SaveToCSV* berfungsi untuk mengkonversi matriks proyeksi menjadi sebuah dokumen berjenis *comma-separated values*.

BAB 4

PERANCANGAN PERANGKAT LUNAK

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

4.1 Perancangan Antarmuka

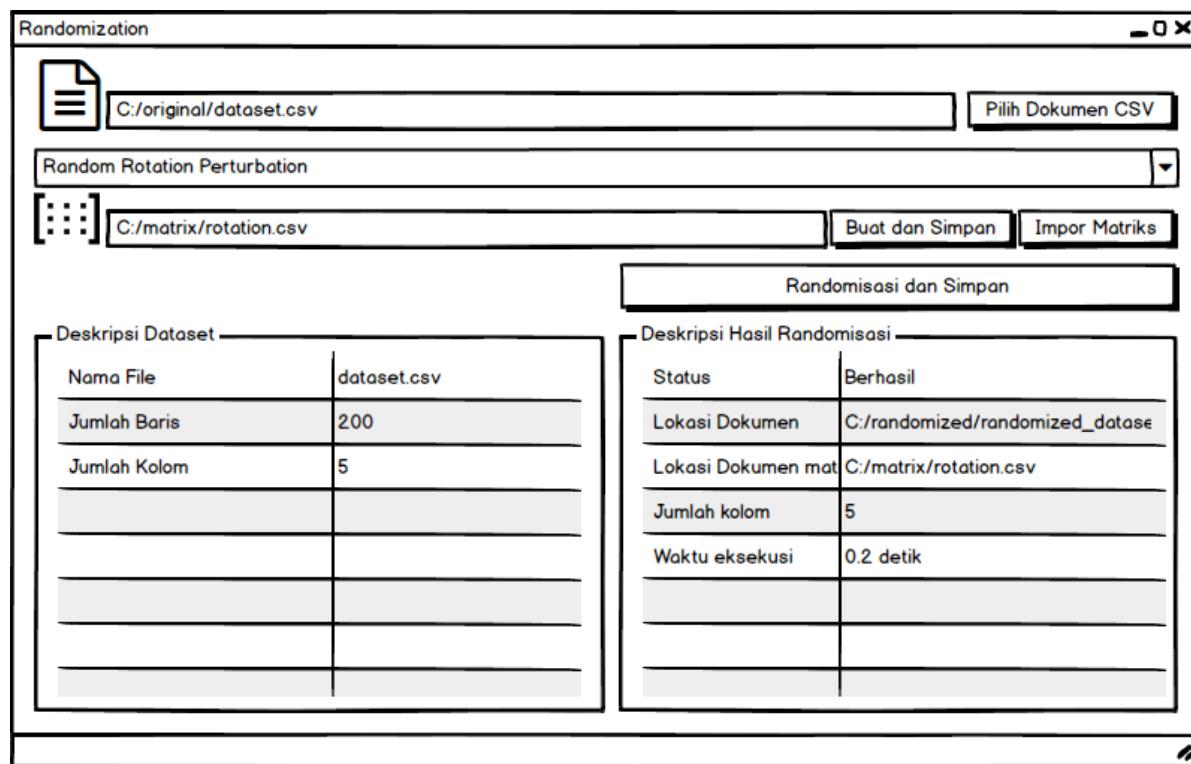
Perancangan antarmuka yang dibuat disesuaikan dengan diagram kelas dan diagram aktivitas yang dibuat sesuai analisis perangkat lunak dilakukan. Rancangan antarmuka dapat dilihat pada Gambar 4.1. Pada bagian atas antarmuka terdapat sebuah kolom untuk memasukkan dokumen berjenis *comma-separated values*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

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

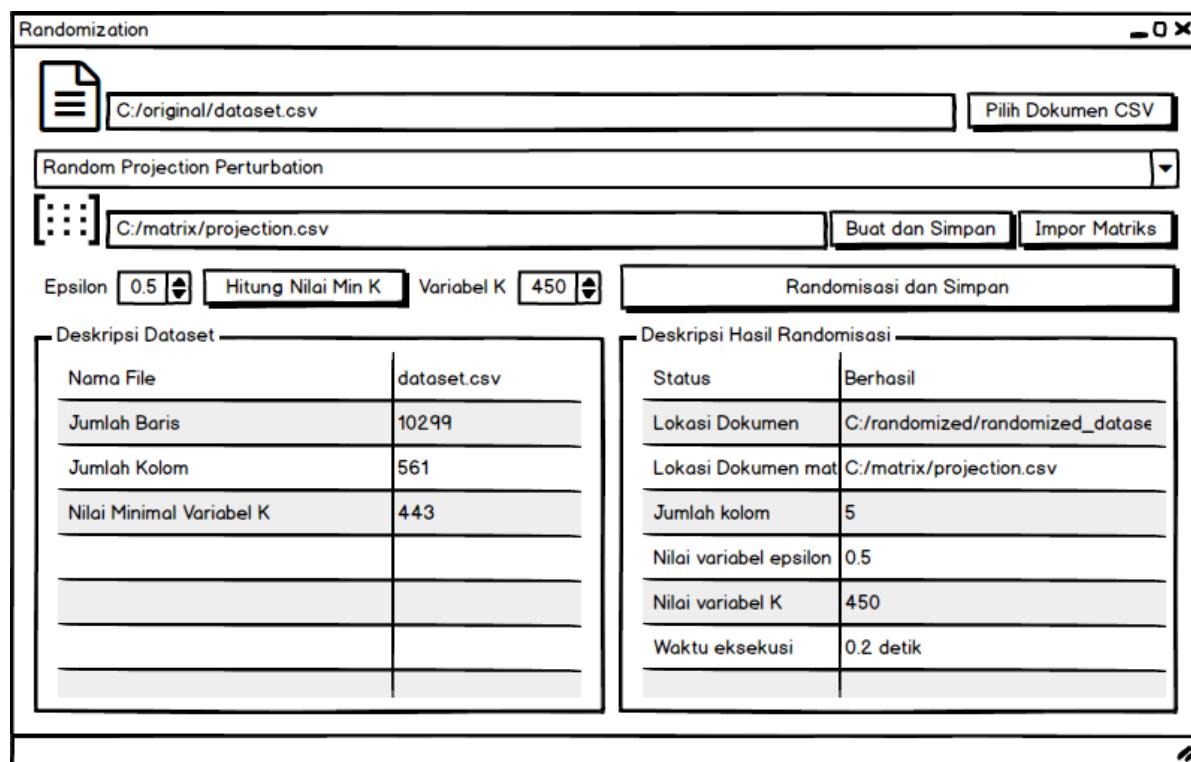
Setelah pengguna melakukan berbagai pengaturan, pengguna dapat melakukan randomisasi dengan menekan tombol "Randomisasi dan Simpan". Apabila randomisasi berhasil dilakukan, perangkat lunak akan menampilkan *popup* yang memberitahukan bahwa randomisasi berhasil dilakukan. *Popup* tersebut dapat dilihat pada Gambar 4.3. Selain itu, perangkat lunak juga akan menampilkan berbagai informasi beserta deskripsi tentang hasil randomisasi yang berhasil dilakukan.

Perangkat lunak akan menampilkan deskripsi hasil dari randomisasi yang dilakukan. Informasi yang ditampilkan oleh perangkat lunak antara lain nama dokumen, ukuran dokumen, jumlah fitur, nilai Epsilon yang dipakai, jumlah dimensi pada hasil randomisasi, dan kolom yang diabaikan. Informasi ini ditampilkan oleh perangkat lunak bertujuan untuk memberitahukan pengguna properti-properti dataset yang telah dirandomisasi dan pengguna dapat memeriksa hasil yang dihasilkan oleh perangkat lunak apakah sesuai dengan keinginan pengguna.

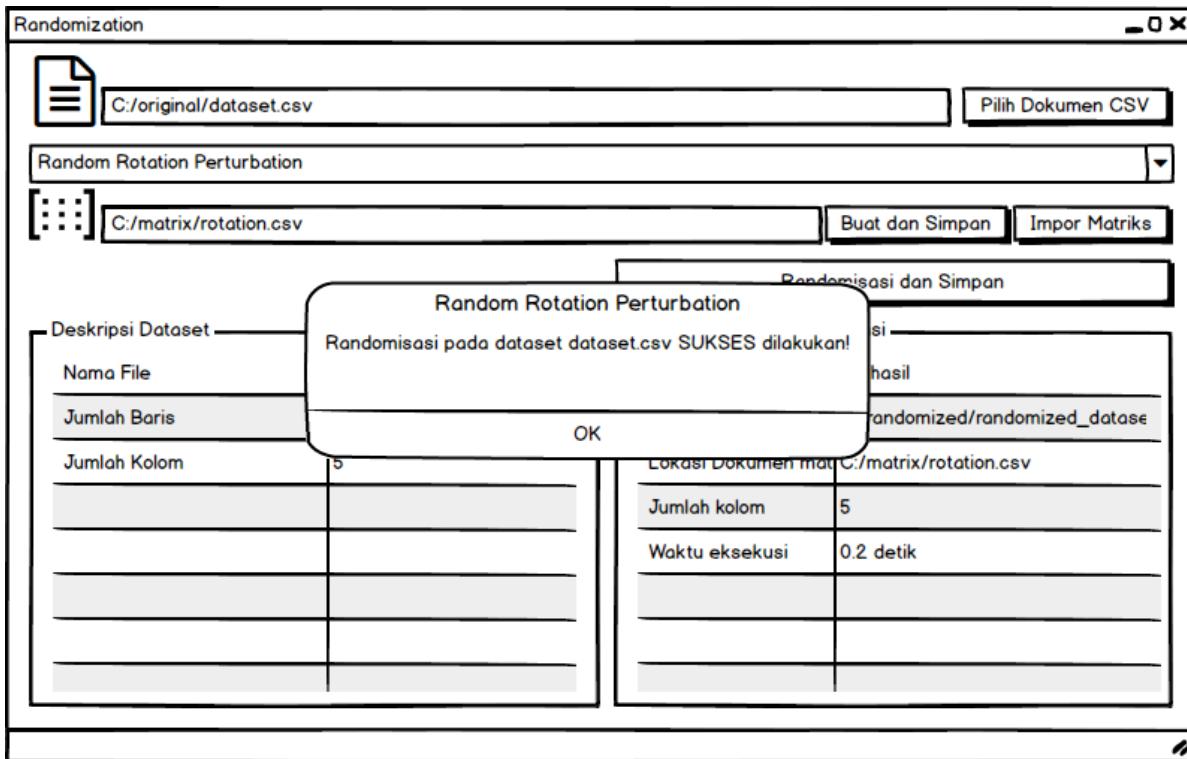
Apabila ada masalah-masalah tertentu dan randomisasi gagal dilakukan, maka perangkat lunak akan memberitahukan bahwa randomisasi telah gagal dilakukan dengan menampilkan *popup* yang



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 randomisasi sukses dilakukan

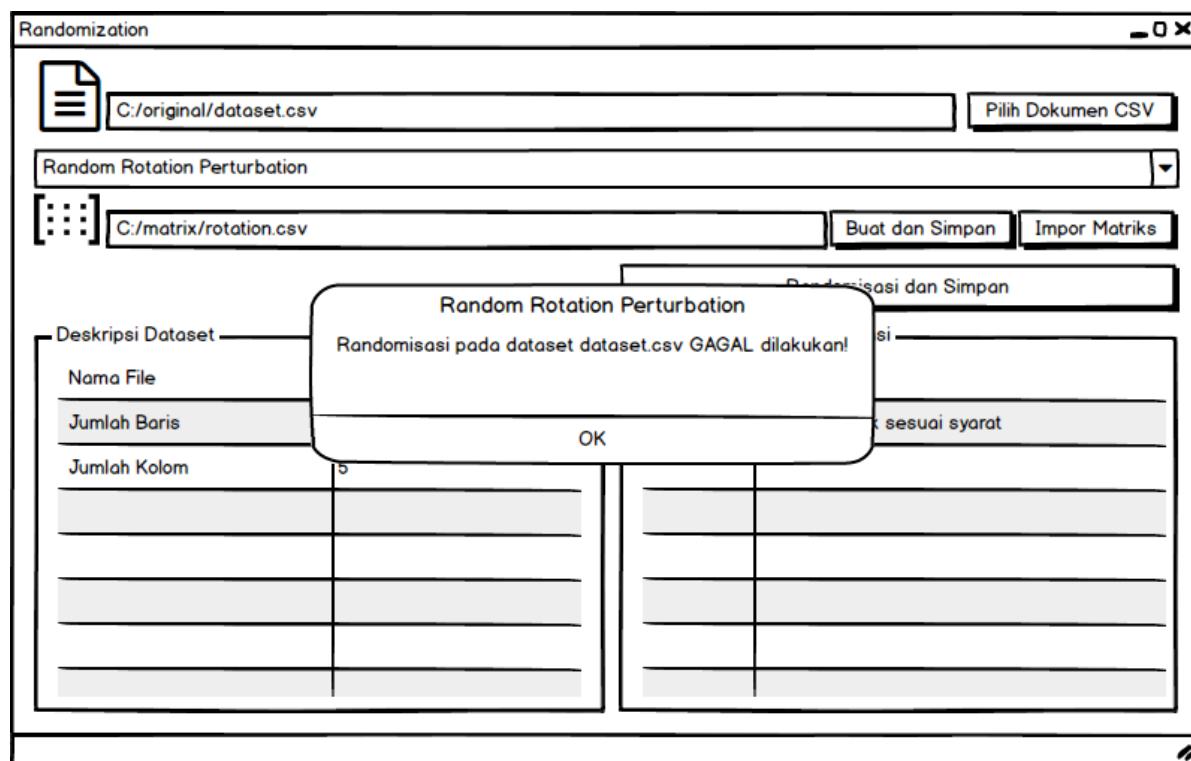
berisi peringatan bahwa randomisasi gagal dilakukan. Hasil randomisasi tidak akan terbuat dan perangkat lunak akan menampilkan informasi bahwa randomisasi gagal dilakukan. *Popup* tersebut dapat dilihat pada Gambar 4.4. Oleh karena adanya persyaratan yang harus dipenuhi oleh pengguna untuk melakukan randomisasi dengan teknik *Random Projection Perturbation*, maka randomisasi 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.

4.2 Perancangan Kelas

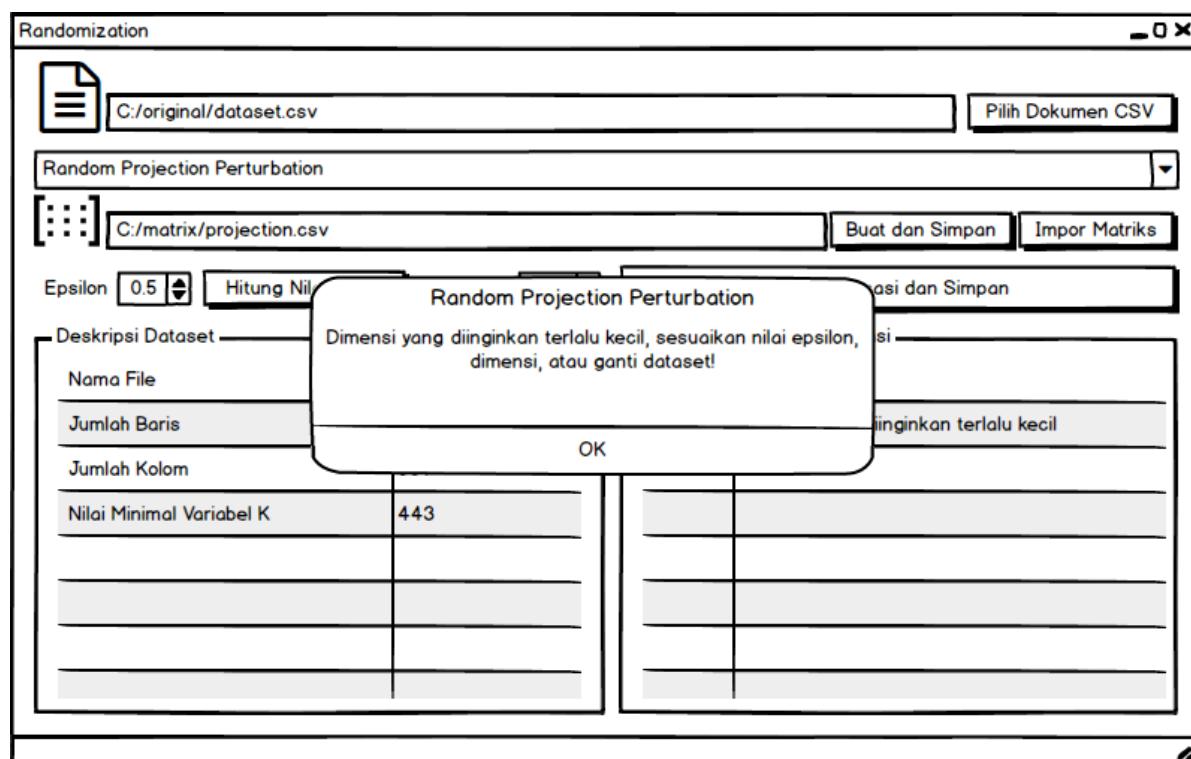
Dalam pengembangan perangkat lunak, perlu adanya perancangan perangkat lunak secara menyeluruh untuk menghindari kesulitan dan kebingungan pada waktu pengembangan. Perangkat lunak yang akan dibuat pada penelitian ini akan berorientasi objek sehingga perlu adanya perancangan kelas-kelas yang akan dibuat. Pada subbab ini akan dijelaskan perancangan kelas perangkat lunak dan apa saja kegunaannya.

4.2.1 Diagram *Package*

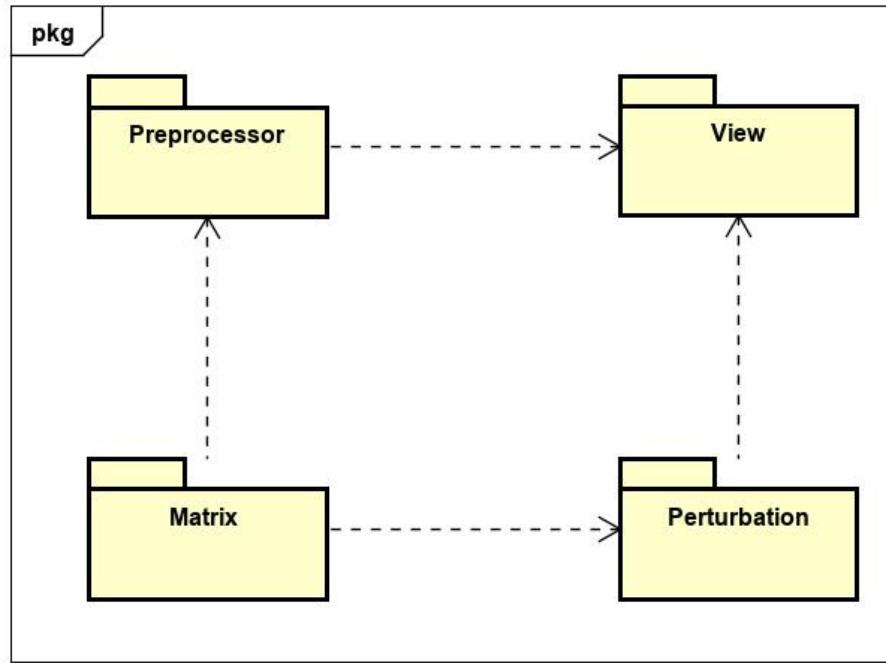
Perangkat lunak akan mempunyai 4 buah *package* yaitu *Perturbation*, *Matrix*, *Preprocessor*, dan *View*. Keempat *package* ini mempunyai fungsinya masing-masing untuk mendukung perangkat lunak berjalan yang akan dijelaskan pada subbab ini. Diagram *Package* perangkat lunak dapat dilihat pada Gambar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



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



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



Gambar 4.6: Diagram *package* perangkat lunak

Package Perturbation

Package Perturbation merupakan *package* yang menangani implementasi algoritma dari teknik randomisasi yang dipakai pada penelitian ini yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation*. Kedua algoritma ini masing-masing akan menjadi sebuah kelas terpisah dan memiliki fungsinya masing-masing. Kedua kelas ini akan berada di dalam *package Perturbation*. Satu kelas lagi akan ada di dalam *package* ini yang berperan sebagai kelas *super* bersifat abstrak untuk kedua kelas lainnya.

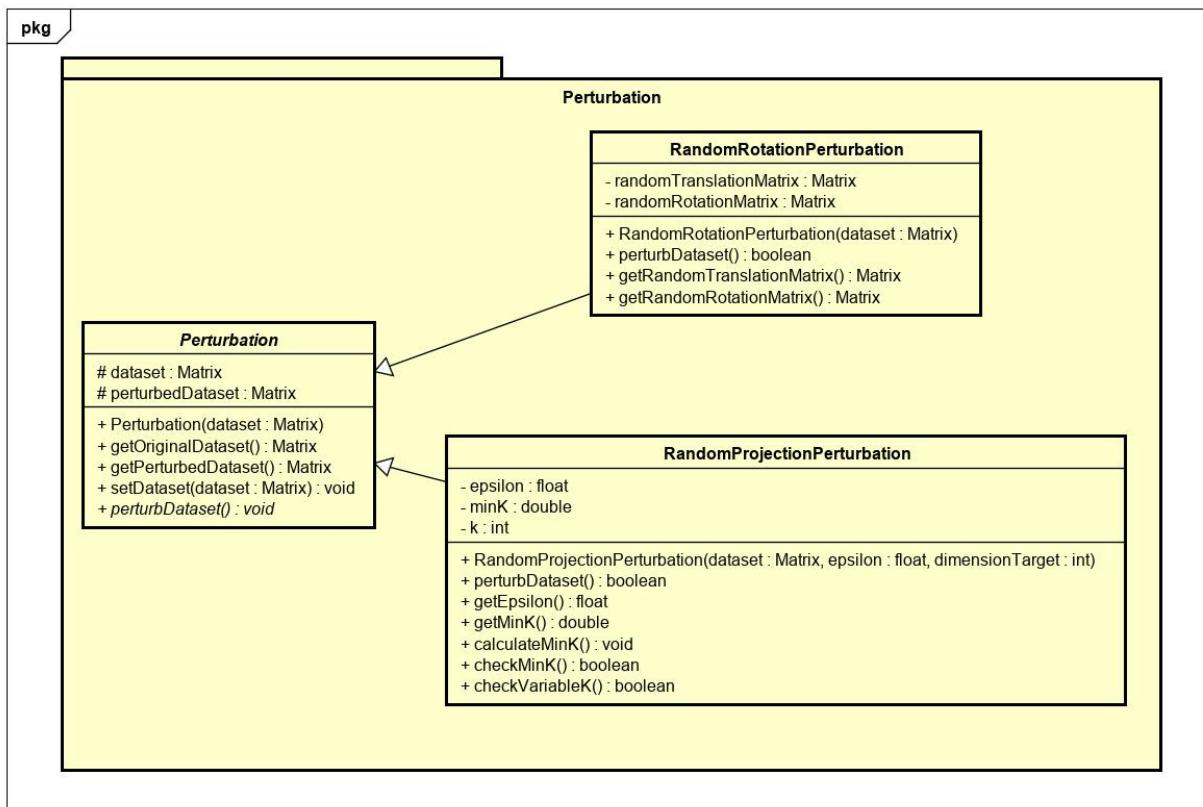
Dalam penerapan algoritma teknik yang dipakai, *package* ini membutuhkan komponen atau fungsi lain untuk membuat algoritma yang diimplementasikan bekerja. Fungsi yang dibutuhkan antara lain adalah membuat matriks dengan sifat tertentu. Oleh karena itu *package Perturbation* membutuhkan *package Matrix*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

Package Matrix

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

Package Preprocessor

Package Preprocessor merupakan *package* yang berfungsi sebagai *preprocessor* untuk masukan dan keluaran perangkat lunak. Tentunya masukan yang diberikan pengguna kepada perangkat lunak tidak bisa langsung diolah begitu saja. Perlu adanya pengolahan terlebih dahulu sebelum masukan



Gambar 4.7: Diagram kelas pada *package Perturbation*

yang diterima dipakai pada perangkat lunak. Oleh karena itu, *package* ini mempunyai fungsi untuk menyelesaikan masalah tersebut yaitu mengolah masukan yang diterima menjadi struktur data yang sesuai untuk digunakan pada perangkat lunak.

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

Package View

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

4.2.2 Diagram Kelas pada *Package Perturbation*

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

Kelas *Perturbation*

Kelas *Perturbation* berperan sebagai kelas abstrak yang akan menjadi kelas *super* dari kedua kelas lainnya dalam package *Perturbation* yaitu kedua kelas yang mengimplementasikan algoritma teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*. Kelas ini mendeklarasikan atribut dan fungsi apa saja yang seharusnya diimplementasikan oleh kelas *RandomRotationPerturbation*, dan *RandomProjectionPerturbation*. Beberapa atribut dan fungsi tersebut masih kosong pada kelas *Perturbation* sehingga berbagai atribut dan fungsi tersebut perlu didefinisikan pada kelas-kelas turunannya. Selanjutnya akan dijelaskan secara rinci tiap atribut dan fungsi pada kelas ini.

Berikut adalah deskripsi setiap atribut pada kelas *Perturbation*.

- *dataset* adalah atribut untuk menampung dataset yang diambil dari masukan perangkat lunak yang ingin dirandomisasi dan sudah berbentuk matriks. Tipe data atribut ini adalah *Matrix*.
- *perturbedDataset* adalah atribut untuk menampung hasil dari dataset yang telah dirandomisasi. Tipe data atribut ini adalah *Matrix*.

Berikut adalah deskripsi setiap fungsi pada kelas *Perturbation*.

- *Perturbation* adalah *constructor* dari kelas *Perturbation*. Tujuan utama fungsi ini adalah mendefinisikan atribut-atribut yang ada. Fungsi ini memiliki sebuah masukan yang dinamakan *dataset* yang bertipe data *Matrix* dan berfungsi untuk mendefinisikan atribut *dataset*.
- *getOriginalDataset* adalah fungsi untuk mendapatkan dataset asli yang belum dirandomisasi atau dengan kata lain mendapatkan atribut *dataset*. Fungsi ini tidak memiliki masukan apapun dan mempunyai tipe data kembalian berupa *Matrix*.
- *getPerturbedDataset* adalah fungsi untuk mendapatkan hasil dari dataset yang telah dirandomisasi atau dengan kata lain mendapatkan atribut *perturbedDataset*. Fungsi ini tidak memiliki parameter apapun. Tipe data kembalian pada fungsi ini berupa *Matrix*.
- *setDataset* adalah fungsi untuk mendefinisikan ulang atribut *dataset* dengan dataset yang baru. Fungsi ini memiliki sebuah masukan yang dinamakan *dataset* yang bertipe data *Matrix* dan berfungsi untuk mendefinisikan atribut *dataset*. Tidak ada kembalian pada fungsi ini.
- *perturbDataset* adalah fungsi untuk melakukan randomisasi pada atribut *dataset* dan menyimpan hasilnya pada atribut *perturbedDataset*. Fungsi ini bersifat abstrak yang berarti fungsi ini belum didefinisikan sehingga fungsi ini harus didefinisikan pada setiap kelas turunannya. Tidak ada kembalian ataupun masukan pada fungsi ini.

Kelas *RandomRotationPerturbation*

Kelas *RandomRotationPerturbation* adalah kelas yang mempunyai tujuan utama melakukan randomisasi dengan teknik *Random Rotation Perturbation* pada dataset yang menjadi masukan perangkat lunak. Kelas ini merupakan kelas turunan dari kelas *Perturbation* sehingga kelas inipun mewarisi semua atribut dan fungsi yang dimiliki oleh kelas *Perturbation*. Oleh karena itu, fungsi *perturbDataset* yang diturunkan dari kelas *Perturbation* harus didefinisikan oleh kelas *RandomRotationPerturbation*. Kelas ini akan mengimplementasikan algoritma dari teknik *Random Rotation Perturbation* untuk melakukan randomisasi pada fungsi *perturbDataset*. Selanjutnya akan dijelaskan secara rinci tiap atribut dan fungsi pada kelas ini.

Berikut adalah deskripsi setiap atribut pada kelas *RandomRotationPerturbation*.

- *randomTranslationMatrix* adalah atribut untuk menampung matriks translasi yang akan dipergunakan untuk implementasi algoritma *Random Rotation Perturbation*. Atribut ini mempunyai tipe data *Matrix*.

- *randomRotationMatrix* adalah atribut untuk menampung matriks rotasi yang akan dipergunakan untuk implementasi algoritma *Random Rotation Perturbation*. Atribut ini mempunyai tipe data *Matrix*.

Berikut adalah deskripsi setiap fungsi pada kelas *RandomRotationPerturbation*.

- *RandomRotationPerturbation* adalah *constructor* untuk mendefinisikan atribut-atribut yang ada dan memanggil *constructor* dari kelas *super* milik kelas *RandomRotationPerturbation* yaitu *Perturbation*. Tujuan utama fungsi ini adalah memanggil *constructor* dari kelas *super* yaitu *Perturbation* sehingga fungsi ini berguna untuk mendefinisikan atribut *dataset*. Selain itu, fungsi ini juga akan mendefinisikan atribut *randomTranslationMatrix* dan *randomRotationMatrix*.
- *perturbDataset* adalah fungsi turunan dari kelas *Perturbation* yang pada kelas *RandomRotationPerturbation* diimplementasikan algoritma teknik *Random Rotation Perturbation*. Fungsi ini akan mengimplementasikan algoritma teknik *Random Rotation Perturbation* kepada atribut *dataset* dan menyimpan hasilnya kepada atribut *perturbedDataset*. Tidak ada masukan pada fungsi ini tetapi memiliki kembalian berupa *boolean* yang menyatakan apakah randomisasi berhasil dilakukan.
- *getRandomTranslationMatrix* adalah fungsi untuk mendapatkan matriks translasi yang digunakan untuk implementasi algoritma teknik *Random Rotation Perturbation* atau dengan kata lain mendapatkan atribut *randomTranslationMatrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa matriks translasi yang bertipe data *Matrix*.
- *getRandomRotationMatrix* adalah fungsi untuk mendapatkan matriks rotasi yang digunakan untuk implementasi algoritma teknik *Random Rotation Perturbation* atau dengan kata lain mendapatkan atribut *randomRotationMatrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa matriks rotasi yang bertipe data *Matrix*.

Kelas *RandomProjectionPerturbation*

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

Berikut adalah deskripsi setiap atribut pada kelas *RandomProjectionPerturbation*.

- *epsilon* adalah atribut yang berguna untuk menentukan seberapa besar batas maksimal distorsi yang dapat terjadi pada hasil randomisasi. Atribut *epsilon* ini akan digunakan untuk menghitung nilai dari atribut *minK* yang akan dijelaskan nanti. Tipe data atribut ini adalah *float*, bilangan riil yang nantinya hanya akan diisi dengan rentang nilai (0, 1). Pengguna akan menentukan seberapa besar batas maksimal distorsi yang dapat terjadi pada hasil randomisasi dengan cara mendefinisikan atribut *epsilon* ini.
- *minK* adalah atribut yang berguna untuk menentukan dimensi terkecil untuk sebuah dataset yang ingin dirandomisasi dapat direduksi dengan distorsi yang terkontrol sesuai atribut *epsilon*. Atribut ini juga menentukan apakah dataset memenuhi persyaratan untuk direduksi yaitu memiliki jumlah kolom yang lebih besar dari nilai *minK*. Tipe data atribut ini adalah *double*, bilangan riil. Atribut ini akan dihitung oleh perangkat lunak sesuai nilai *epsilon* yang telah ditentukan pengguna.

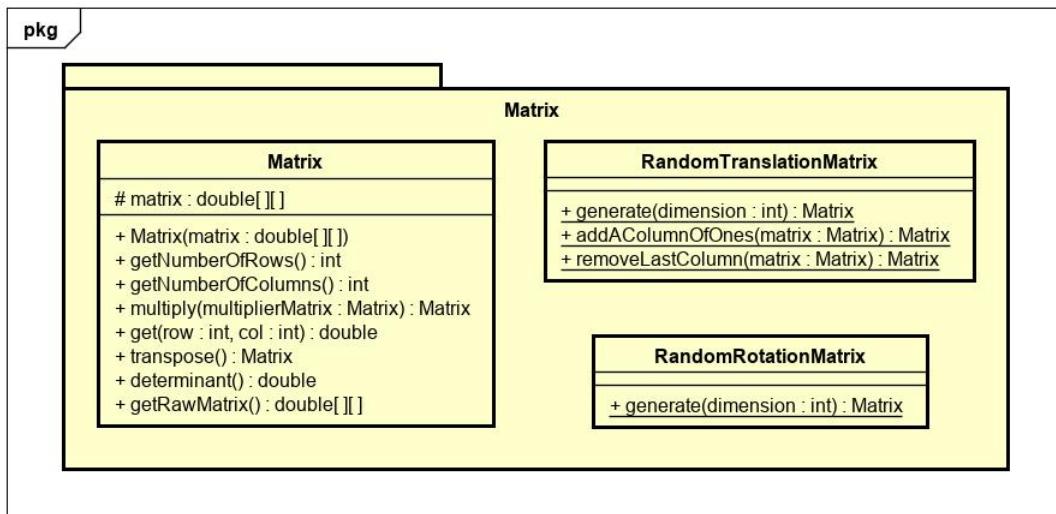
- k adalah atribut untuk menyimpan besar dimensi akhir yang diinginkan pengguna dimiliki oleh hasil dataset yang telah dirandomisasi. Pengguna harus mendefinisikan atribut ini dengan nilai yang lebih besar dari atau sama dengan nilai $minK$, jika tidak maka distorsi yang terjadi pada hasil randomisasi akan lebih besar dari nilai $epsilon$ yang diinginkan. Tipe data atribut ini adalah *integer*, bilangan bulat.

Berikut adalah deskripsi setiap fungsi pada kelas *RandomProjectionPerturbation*.

- *RandomProjectionPerturbation* adalah *constructor* untuk mendefinisikan atribut-atribut yang ada dan memanggil *constructor* dari kelas *super* milik kelas *RandomProjectionPerturbation* yaitu *Perturbation*. Tujuan utama fungsi ini adalah memanggil *constructor* dari kelas *super* yaitu *Perturbation* sehingga fungsi ini berguna untuk mendefinisikan atribut *dataset*. Selain itu, fungsi ini juga akan mendefinisikan atribut *epsilon*, *minK*, dan *k*.
- *perturbDataset* adalah fungsi turunan dari kelas *Perturbation* yang pada kelas *RandomProjectionPerturbation* diimplementasikan algoritma teknik *Random Projection Perturbation*. Fungsi ini akan mengimplementasikan algoritma teknik *Random Projection Perturbation* kepada atribut *dataset* dan menyimpan hasilnya kepada atribut *perturbedDataset*. Tidak ada masukan pada fungsi ini tetapi memiliki kembalian berupa *boolean* yang menyatakan apakah randomisasi berhasil dilakukan.
- *getEpsilon* adalah fungsi untuk mendapatkan nilai batas maksimal distorsi yang dapat terjadi pada hasil randomisasi atau dengan kata lain mendapatkan atribut *epsilon*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa nilai dari atribut *epsilon* yang bertipe data *float*, bilangan riil.
- *getMinK* adalah fungsi untuk mendapatkan besar dimensi minimal hasil dataset yang telah dirandomisasi atau dengan kata lain mendapatkan atribut *minK*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa nilai dari atribut *minK* yang bertipe data *double*, bilangan riil.
- *calculateMinK* adalah fungsi untuk menghitung nilai *minK* sesuai dengan atribut *epsilon* dan jumlah baris pada dataset yang ingin dirandomisasi. Fungsi ini akan menghitung nilai *minK* tersebut dan menyimpan hasilnya pada atribut *minK*. Tidak ada masukan ataupun kembalian pada fungsi ini.
- *checkMinK* adalah fungsi untuk memeriksa apakah dimensi dari *dataset* yang ingin dirandomisasi lebih besar dari nilai *minK*, besar dimensi minimal. Selain itu, fungsi ini memeriksa apakah nilai *k* lebih besar dari atau sama dengan nilai *minK*. Intinya fungsi ini menguji apakah dataset dan keinginan pengguna memenuhi persyaratan untuk mengaplikasikan teknik *Random Projection Perturbation*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa sebuah *boolean* yang menandakan apakah *dataset* dan *k* yang diberikan oleh pengguna memenuhi persyaratan untuk mengaplikasikan teknik *Random Projection Perturbation* dengan *dataset* dan *k* tersebut.
- *checkVariableK* adalah fungsi untuk memeriksa apakah nilai *k* lebih besar dari atau sama dengan nilai *minK*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa sebuah *boolean* yang menandakan apakah nilai variabel *k* yang diberikan oleh pengguna memenuhi persyaratan untuk mengaplikasikan teknik *Random Projection Perturbation* dengan nilai variabel *k* tersebut.

4.2.3 Diagram Kelas pada *Package Matrix*

Package Matrix memiliki empat buah kelas yang bertujuan untuk menangani segala jenis fungsi yang berkaitan dengan matriks maupun pembuatan matriks yang khusus digunakan untuk implementasi



Gambar 4.8: Diagram kelas pada *package Matrix*

algoritma. Keempat kelas tersebut adalah *Matrix*, *RandomTranslationMatrix*, dan *RandomRotationMatrix* yang akan dijelaskan secara detail pada subbab ini. Diagram kelas pada *package Matrix* dapat dilihat pada Gambar 4.8.

Kelas *Matrix*

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

Berikut adalah deskripsi setiap atribut pada kelas *Matrix*.

- *matrix* adalah atribut untuk menyimpan matriks dengan cara menyimpannya memakai *array 2 dimensi* dengan tipe data *double*, bilangan riil.

Berikut adalah deskripsi setiap fungsi pada kelas *Matrix*.

- *Matrix* adalah *constructor* dari kelas *PerturbMatrix*. Tujuan utama fungsi ini adalah mendefinisikan atribut-atribut yang ada. Fungsi ini memiliki sebuah masukan berbentuk array yang dinamakan *matrix* yang bertipe data *double* dan berfungsi untuk mendefinisikan atribut *matrix*.
- *getNumberOfRows* adalah fungsi untuk mendapatkan jumlah baris yang ada pada matriks kelas ini atau dengan kata lain mendapatkan ukuran dari atribut *matrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa jumlah baris yang bertipe data *integer*, bilangan bulat.
- *getNumberOfColumns* adalah fungsi untuk mendapatkan jumlah kolom yang ada pada matriks kelas ini atau dengan kata lain mendapatkan ukuran dari atribut *matrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa jumlah baris yang bertipe data *integer*, bilangan bulat.
- *multiply* adalah fungsi yang berguna untuk mengkalikan matriks yang ada pada atribut *matrix* dengan suatu matriks lain. Tentu saja matriks lain tersebut harus sudah berbentuk objek kelas *Matrix* sehingga mempunyai tipe data yang sama dan dapat dikalikan. Fungsi ini

mempunyai sebuah parameter yaitu *multiplierMatrix* bertipe data objek kelas *Matrix* yang berguna sebagai pengali matriks yang ingin dikalikan. Kembalian pada fungsi ini adalah hasil kali antara kedua matriks dan kembalian ini bertipe data objek kelas *Matrix*.

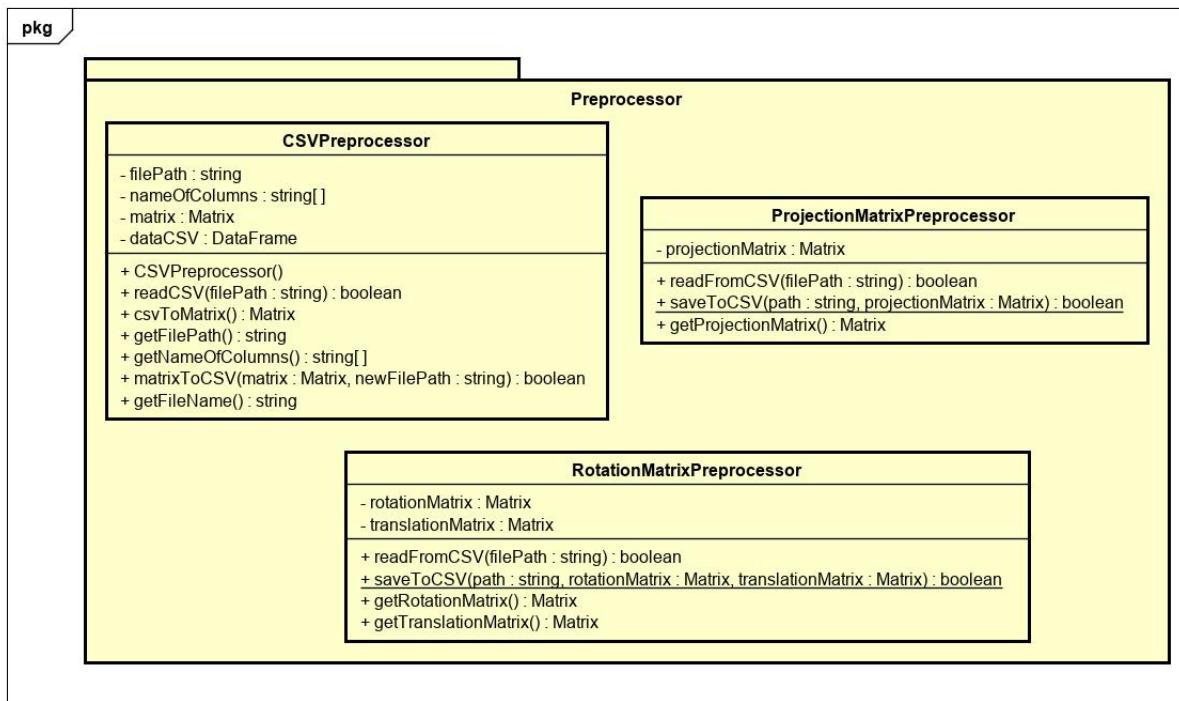
- *get* adalah fungsi untuk mendapatkan sebuah elemen pada matriks atau dengan kata lain sebuah nilai pada atribut *matrix*. Fungsi ini memiliki dua buah masukan yaitu *row* dan *col* yang masing-masing berguna sebagai penunjuk baris dan kolom mana yang ingin didapatkan. Kembalian pada fungsi ini adalah elemen matriks pada baris dan kolom yang diinginkan dan bertipe data *double*, bilangan riil.
- *transpose* adalah fungsi untuk mendapatkan transpose dari matriks kelas ini atau dengan kata lain transpose dari atribut *matrix*. Fungsi ini memiliki kembalian berupa matriks yang merupakan hasil transpose dan bertipe data objek kelas *Matrix*. Tidak ada masukan apapun pada fungsi ini.
- *determinant* adalah fungsi untuk mendapatkan determinan dari matriks kelas ini. Fungsi ini memiliki kembalian berupa determinan matriks yang bertipe data *double*, bilangan riil. Tidak ada masukan apapun pada fungsi ini.
- *getRawMatrix* adalah fungsi untuk mendapatkan matriks pada kelas ini atau dengan kata lain mendapatkan atribut *matrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa *array* yang bertipe data *double*, bilangan riil.

Kelas *RandomTranslationMatrix*

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

Berikut adalah deskripsi setiap fungsi pada kelas *RandomTranslationMatrix*.

- *generate* adalah fungsi statis yang berguna untuk membuat matriks translasi. Fungsi ini memiliki sebuah parameter yaitu *dimension* yang akan menentukan ukuran atau dimensi matriks translasi yang ingin dibuat. Kembalian pada fungsi ini tentu saja sebuah matriks translasi yang bertipe data objek kelas *Matrix*.
- *addAColumnOfOnes* adalah fungsi statis yang berguna untuk menambahkan sebuah kolom pada suatu matriks di posisi terakhir (setelah kolom terakhir). Kolom tersebut akan berisi angka satu pada setiap barisnya. Alasan dibuatnya fungsi ini dikarenakan kebutuhan persyaratan yang harus dipenuhi sebuah matriks apabila ingin diterapkan operasi translasi. Fungsi ini memiliki sebuah parameter yaitu matriks yang diinginkan bernama *matrix* dan bertipe data objek kelas *Matrix*. Kembalian pada fungsi ini adalah matriks yang telah ditambahkan sebuah kolom dan bertipe data objek kelas *Matrix*.
- *removeLastColumn* adalah fungsi statis yang berguna untuk menghapus kolom terakhir pada suatu matriks. Alasan dibuatnya fungsi ini dikarenakan kebutuhan persyaratan yang harus dipenuhi sebuah matriks apabila ingin diterapkan operasi translasi. Setelah matriks ditambahkan dengan menggunakan fungsi *addAColumnOfOnes* dan diterapkan operasi translasi, kolom terakhir pada matriks tersebut perlu dibuang dikarenakan kolom tersebut adalah kolom hasil penambahan yang hanya digunakan untuk operasi translasi dan bukan kolom asli matriks tersebut.



Gambar 4.9: Diagram kelas pada *package Preprocessor*

Kelas *RandomRotationMatrix*

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

Berikut adalah deskripsi setiap fungsi pada kelas *RandomRotationMatrix*.

- *generate* adalah fungsi statis yang berguna untuk membuat matriks rotasi. Pembuatan matriks rotasi dilakukan dengan mengikuti distribusi Haar [8]. Fungsi ini memiliki sebuah parameter yaitu *dimension* yang akan menentukan ukuran atau dimensi matriks rotasi yang ingin dibuat. Kembalian pada fungsi ini tentu saja sebuah matriks rotasi yang bertipe data objek kelas *Matrix*.

4.2.4 Diagram Kelas pada *Package Preprocessor*

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

Kelas *CSVPreprocessor*

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

Berikut adalah deskripsi setiap atribut pada kelas *CSVPreprocessor*.

- *filePath* adalah atribut untuk menyimpan lokasi masukan yang berjenis *comma-separated values* yang ingin diolah pada komputer pengguna. Atribut ini memiliki tipe data *string*.
- *nameOfColumns* adalah atribut untuk menyimpan nama seluruh kolom pada dataset yang telah diolah. Atribut ini berupa array yang bertipe data *string*.
- *matrix* adalah atribut untuk menyimpan objek kelas *Matrix* yang didapatkan dari hasil pengolahan masukan yang berjenis *comma-separated values*. Atribut ini memiliki tipe data objek kelas *Matrix*.
- *dataCSV* adalah atribut untuk menyimpan data dokumen *comma-separated values* yang telah dibaca dalam bentuk *DataFrame*¹. Atribut ini memiliki tipe data objek kelas *DataFrame*.

Berikut adalah deskripsi setiap fungsi pada kelas *CSVPreprocessor*.

- *CSVPreprocessor* adalah *constructor* kelas ini yang berfungsi untuk membuat sebuah objek kelas *CSVPreprocessor* dan menginisialisasi semua atribut pada kelas ini. Fungsi ini tidak memiliki masukan apapun.
- *readCSV* adalah fungsi untuk membaca dokumen berjenis *comma-separated values* yang ingin diolah oleh kelas ini. Tujuan utama dari fungsi ini adalah menyimpan lokasi dokumen pada atribut *filePath* dan menyimpan data dokumen *comma-separated values* yang telah dibaca dalam bentuk *DataFrame* pada atribut *dataCSV*. Fungsi ini memiliki sebuah masukan bernama *filePath* yang berguna untuk menentukan lokasi dokumen yang ingin diolah. Kembalian pada fungsi ini adalah sebuah *boolean* yang menyatakan apakah fungsi ini berhasil membaca dokumen yang menjadi masukan fungsi *readCSV*.
- *csvToMatrix* adalah fungsi untuk mengolah dokumen *comma-separated values* yang telah dibaca menjadi sebuah objek kelas *Matrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa objek kelas *Matrix* yang didapatkan dari hasil pengolahan dokumen *comma-separated values*.
- *getFilePath* adalah fungsi untuk mendapatkan lokasi dokumen *comma-separated values* yang telah dibaca atau dengan kata lain mendapatkan atribut *filePath*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa lokasi dokumen yang bertipe data *string*.
- *getNameOfColumns* adalah fungsi untuk mendapatkan nama semua kolom yang ada pada dokumen *comma-separated values* yang telah dibaca atau dengan kata lain mendapatkan atribut *nameOfColumns*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa *array* yang bertipe data *string*.
- *matrixToCSV* adalah fungsi untuk mengolah sebuah objek kelas *Matrix* menjadi sebuah dokumen *comma-separated values* atau dengan kata lain mengkonversi sebuah matriks menjadi dokumen berjenis *comma-separated values*. Fungsi ini bisa dikatakan kebalikan dari fungsi *csvToMatrix*. Ada dua buah masukan pada fungsi ini yaitu sebuah matriks dan lokasi penyimpanan dokumen yang baru. Kembalian pada fungsi ini adalah sebuah *boolean* yang menyatakan apakah konversi berhasil dilakukan.
- *getFileName* adalah fungsi untuk mendapatkan nama dokumen *comma-separated values* yang telah dibaca. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa nama dokumen yang bertipe data *string*.

¹<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

Kelas *ProjectionMatrixPreprocessor*

Kelas *ProjectionMatrixPreprocessor* berguna untuk mengolah masukan perangkat lunak berupa matriks proyeksi yang berjenis *comma-separated values* sebelum diterapkan teknik randomisasi agar masukan tersebut sesuai dengan persyaratan teknik randomisasi. Tujuan utama dari kelas ini adalah mengubah masukan matriks proyeksi berjenis *comma-separated values* menjadi sebuah objek kelas *Matrix* dan sebaliknya. Kelas ini memiliki sebuah atribut dan tiga buah fungsi yang selanjutnya akan dijelaskan secara rinci.

Berikut adalah deskripsi setiap atribut pada kelas *ProjectionMatrixPreprocessor*.

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

Berikut adalah deskripsi setiap fungsi pada kelas *ProjectionMatrixPreprocessor*.

- *readFromCSV* adalah fungsi untuk membaca dokumen berjenis *comma-separated values* menjadi matriks proyeksi yang dapat dipakai perangkat lunak. Fungsi ini memiliki sebuah masukan bernama *filePath* yang berguna untuk menentukan lokasi dokumen yang ingin diolah. Kembalian pada fungsi ini adalah sebuah *boolean* yang menyatakan apakah pembacaan dokumen berhasil dilakukan.
- *saveToCSV* adalah fungsi statis untuk mengolah sebuah matriks proyeksi berbentuk objek kelas *Matrix* menjadi sebuah dokumen *comma-separated values* dan menyimpannya pada sebuah direktori pengguna. Fungsi ini bisa dikatakan kebalikan dari fungsi *readFromCSV*. Ada dua buah masukan pada fungsi ini yaitu sebuah matriks proyeksi dan lokasi penyimpanan dokumen yang baru. Kembalian pada fungsi ini adalah sebuah *boolean* yang menyatakan apakah penyimpanan berhasil dilakukan.
- *getProjectionMatrix* adalah fungsi untuk mendapatkan matriks proyeksi berbentuk objek kelas *Matrix* yang didapatkan dari fungsi *readFromCSV* atau dengan kata lain mendapatkan variabel *projectionMatrix*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa matriks proyeksi yang bertipe data *Matrix*.

Kelas *RotationMatrixPreprocessor*

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

Berikut adalah deskripsi setiap atribut pada kelas *RotationMatrixPreprocessor*.

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

Berikut adalah deskripsi setiap fungsi pada kelas *RotationMatrixPreprocessor*.

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

4.3 Masukan Perangkat Lunak

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

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

```
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
4.9,3.1,1.5,0.1,setosa
```

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

BAB 5

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.

5.1 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 4. Gambar 5.1 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 5.2 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 randomisasi, terdapat pada bagian bawah sebelah kanan yang bernomor tiga dan dikelilingi kotak hijau. Ketiga bagian ini akan dijelaskan secara rinci pada subbab-subbab berikutnya.

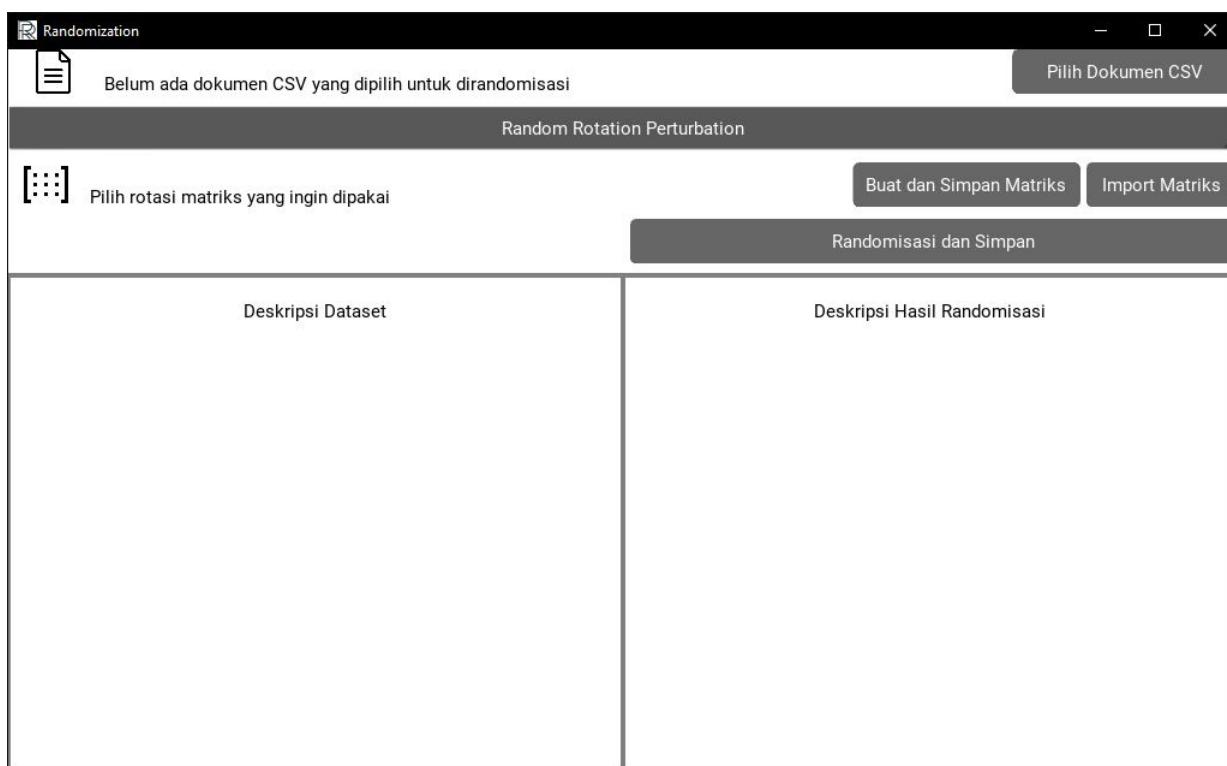
Perangkat lunak randomisasi ini mengimplementasikan dua buah teknik randomisasi yang berbeda yaitu *Random Rotation Perturbation* dan *Random Projection Perturbation*. Oleh karena itu, antarmuka perangkat lunak akan menyesuaikan dengan teknik yang dipilih oleh pengguna. Ketiga bagian antarmuka yang telah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

5.1.1 Masukan dan Pengaturan

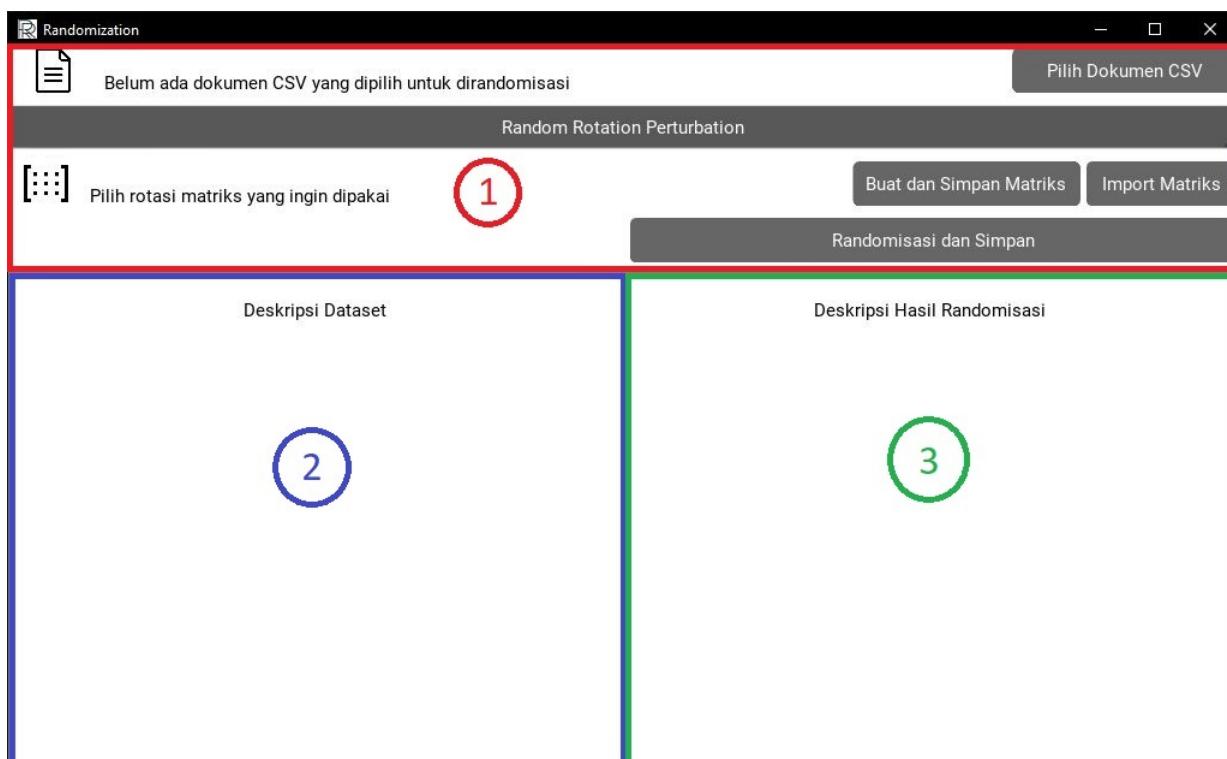
Bagian masukan dan pengaturan menyediakan berbagai interaksi untuk pengguna dapat mengatur masukan yang perlu diberikan kepada perangkat lunak dan menerapkan teknik randomisasi yang diinginkan. Ada beberapa fungsi inti pada bagian ini yaitu sebagai berikut.

- Masukan dataset berupa file *comma-separated values* yang ingin dirandomisasi
- Memilih teknik randomisasi 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 randomisasi dan menyimpan hasilnya

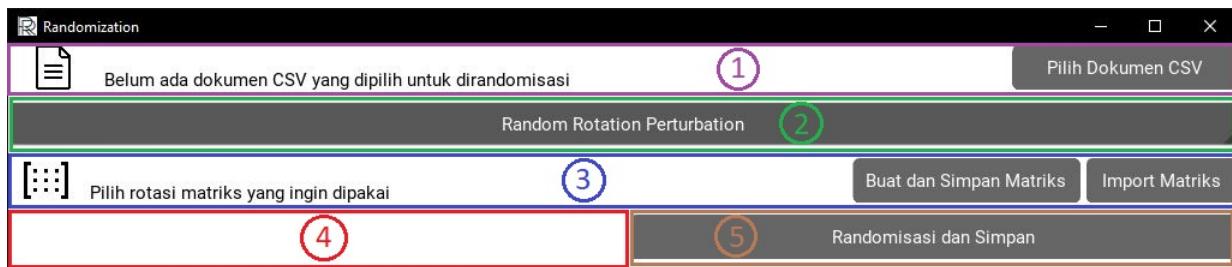
Berikut akan dijelaskan secara rinci dengan gambar setiap fungsi tersebut yang dapat dilihat pada Gambar 5.3 dan cara pemakaianya yang benar secara berturut.



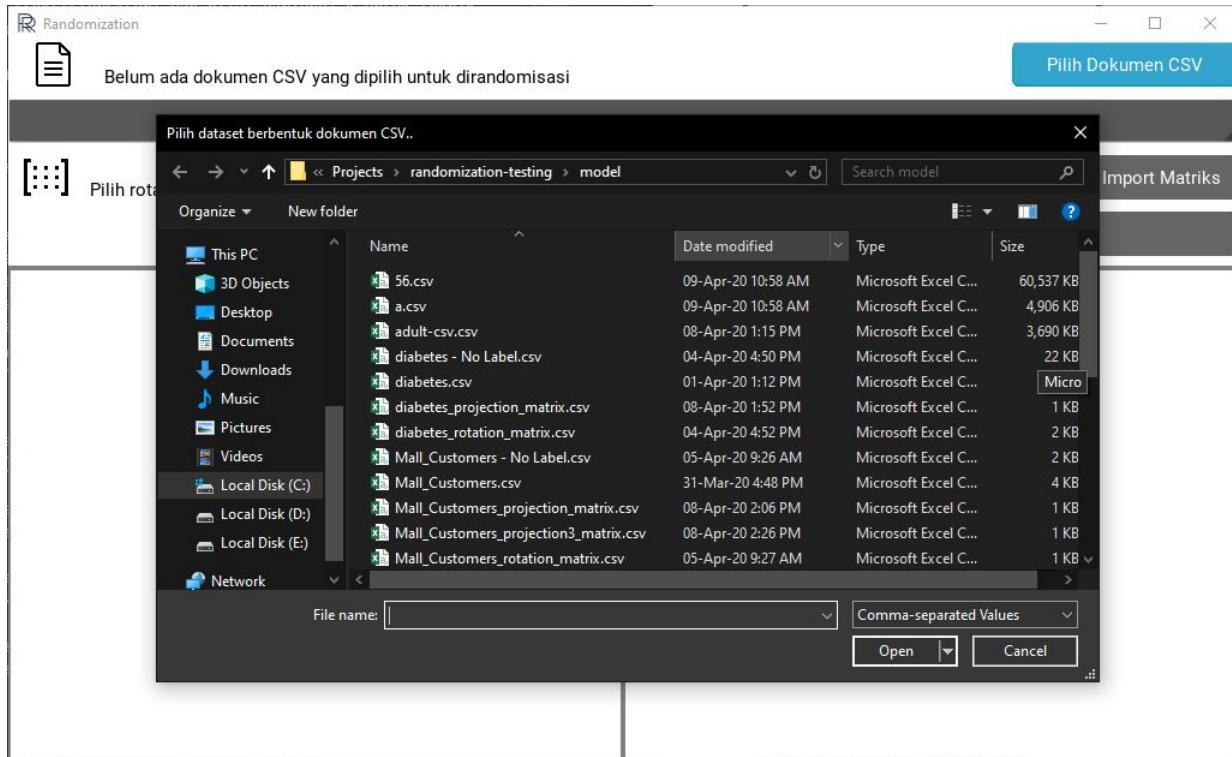
Gambar 5.1: Tampilan perangkat lunak yang pertama ditampilkan saat perangkat lunak baru dibuka



Gambar 5.2: Bagian-bagian pada antarmuka perangkat lunak



Gambar 5.3: Bagian antarmuka masukan dan pengaturan perangkat lunak

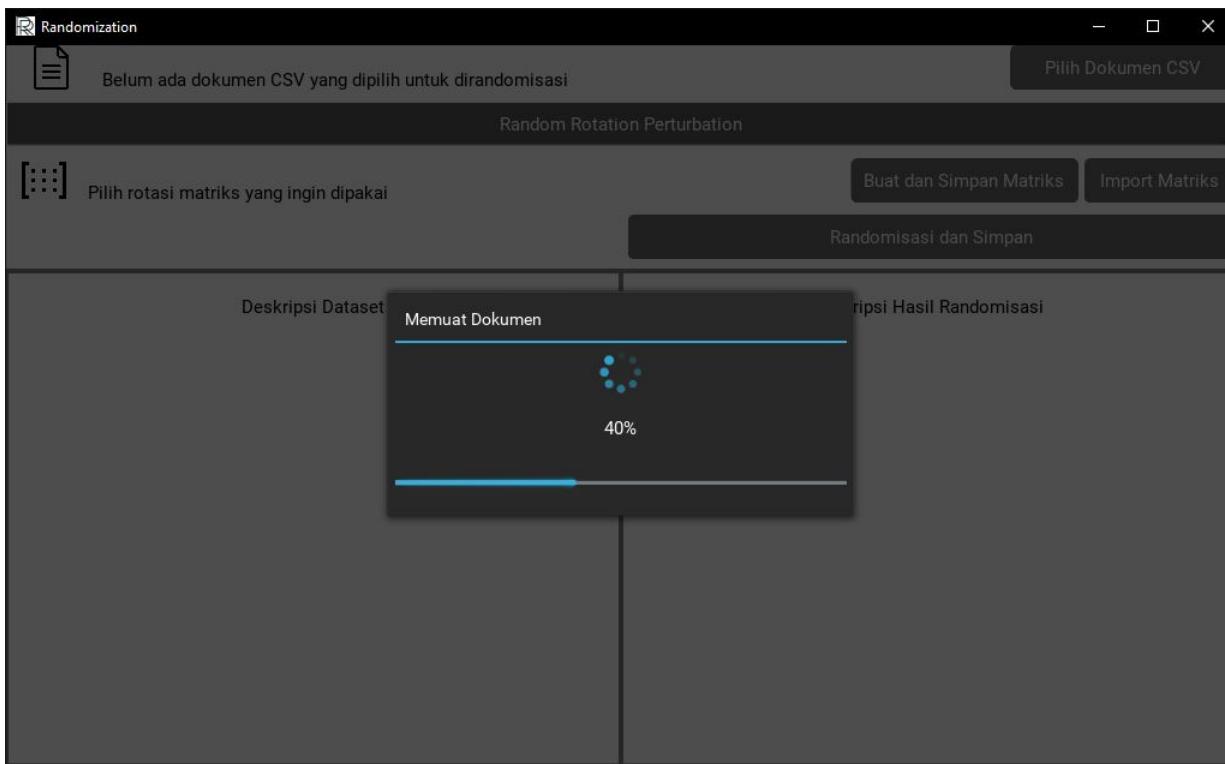


Gambar 5.4: Jendela untuk memilih dataset yang berupa dokumen CSV

Masukan Dataset

Pertama pengguna perlu memberikan masukan dataset yang ingin dirandomisasi 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 bernomor satu. Pengguna dapat menekan tombol “Pilih Dokumen CSV” yang terletak pada ujung sebelah kanan. Tombol ini bertujuan untuk memilih dokumen yang ingin dirandomisasi 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 dirandomisasi” yang menunjukkan bahwa belum ada dokumen yang dipilih oleh pengguna. Jika pengguna memilih ulang dokumen, maka secara otomatis juga perangkat lunak akan memperbaharui lokasi dokumen sesuai dokumen yang dipilih pengguna.



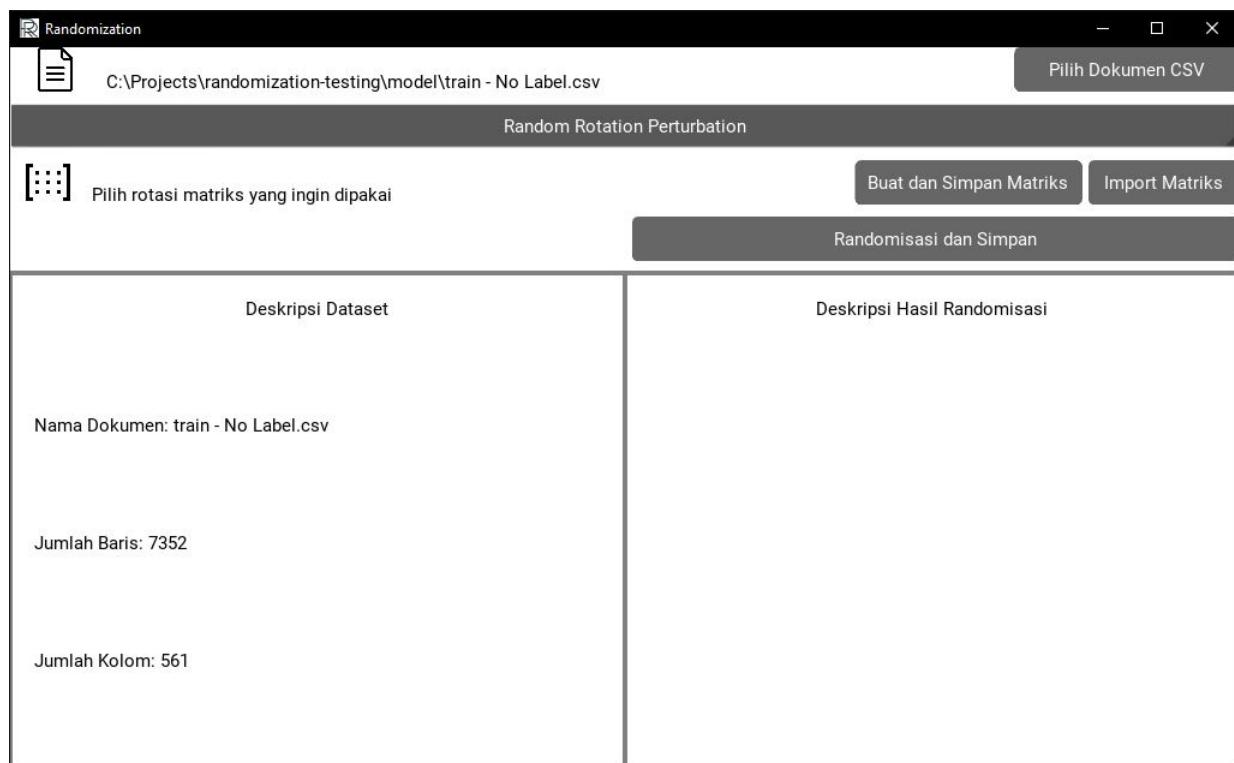
Gambar 5.5: Tampilan *popup* yang ditampilkan saat proses berlangsung

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

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 Randomisasi

Setelah pengguna memilih dataset yang ingin dirandomisasi, pengguna juga harus memilih teknik randomisasi 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 randomisasi. Tombol ini dapat dilihat pada Gambar 5.3



Gambar 5.6: Tampilan antarmuka setelah sebuah dokumen dipilih

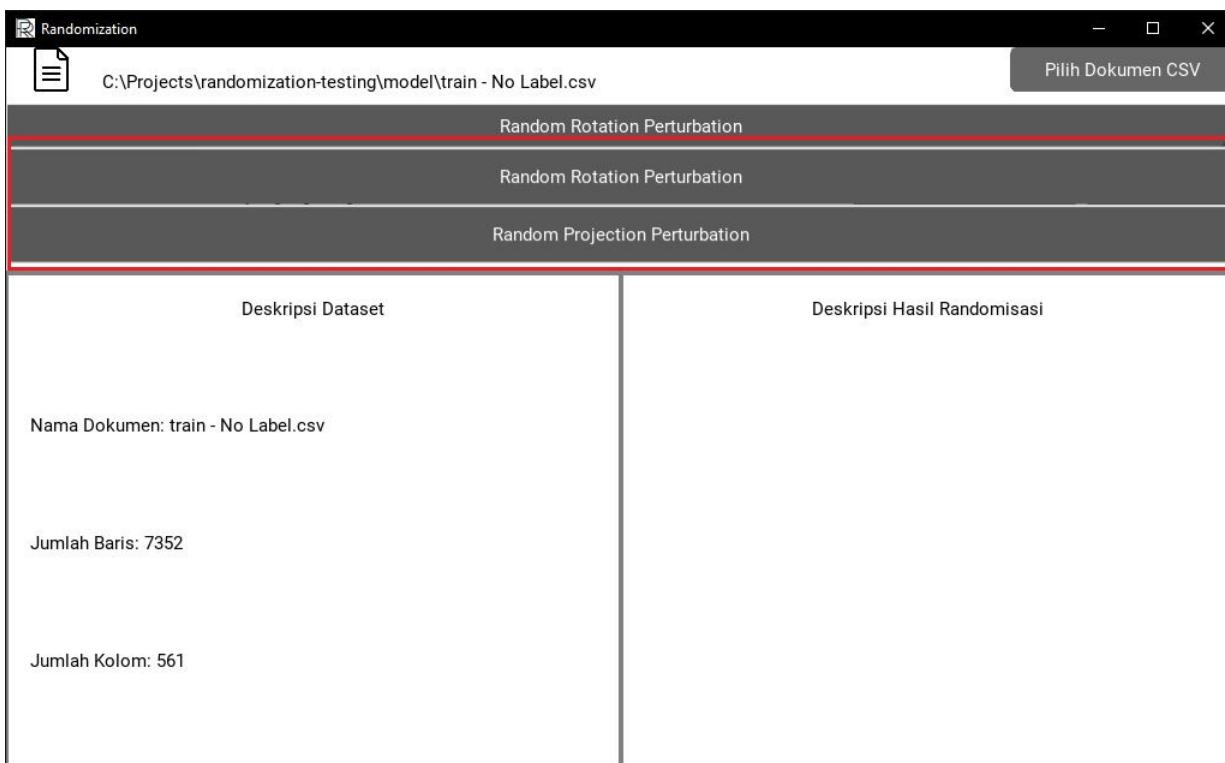
yang dikelilingi kotak berwarna hijau dan bernomor dua.

Apabila pengguna menekan tombol ini maka perangkat lunak akan menampilkan *dropdown* yang mempunyai dua buah opsi teknik randomisasi yaitu “Random Rotation Perturbation” dan “Random Projection Perturbation”. Antarmuka tersebut dapat dilihat pada Gambar 5.7 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 randomisasi, dan bagian randomisasi 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 randomisasi 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 5.3 yang dikelilingi oleh kotak berwarna hijau dan bernomor dua.

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



Gambar 5.7: Tampilan antarmuka saat pengguna memilih teknik

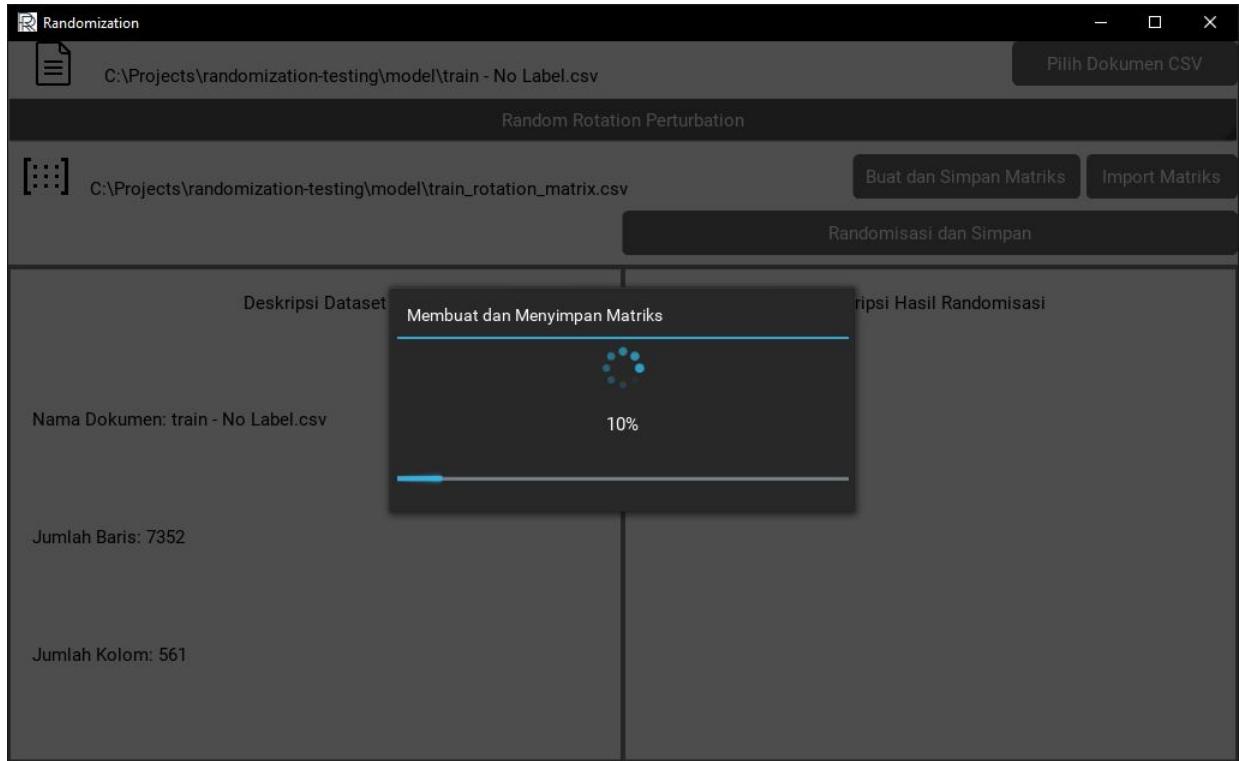
popup memuat yang dapat dilihat pada Gambar 5.8. *Popup* ini juga akan tampil saat proses impor matriks dilakukan. Hasil matriks yang dibuat oleh perangkat lunak dapat digunakan kembali untuk lain kali sehingga rotasi atau proyeksi yang diterapkan akan sama dengan yang sebelumnya.

Pengguna dapat melakukan impor matriks dengan cara menekan tombol “Import Matriks” untuk memilih matriks rotasi atau proyeksi yang diinginkan untuk diterapkan pada dataset. Matriks yang dipilih harus sesuai dengan dataset yang ingin dirandomisasi, misalnya apabila matriks rotasi yang dipilih memiliki dimensi yang berbeda dengan dataset maka perangkat lunak akan melarang impor matriks dilakukan karena randomisasi tidak dapat dilakukan. Perangkat lunak akan menampilkan *popup* peringatan untuk pengguna yang dapat dilihat pada Gambar 5.9. 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 diimpor.

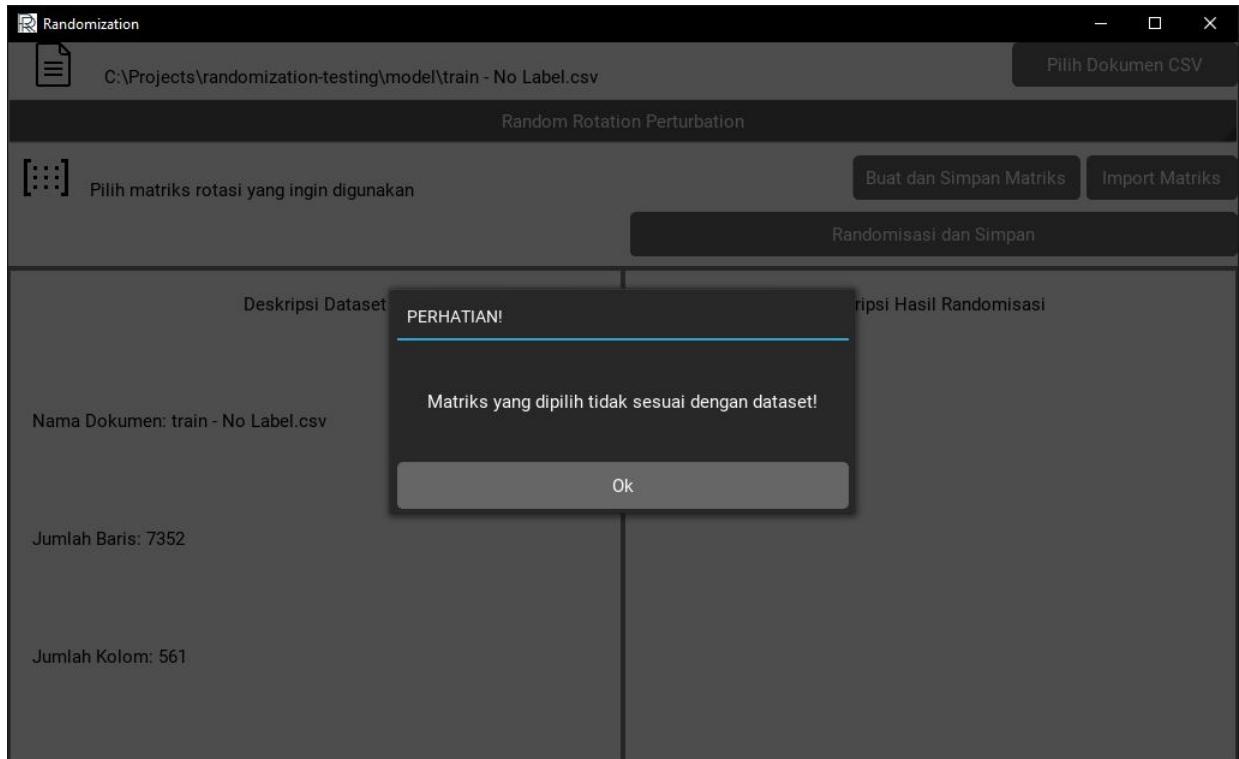
Apabila pengguna belum memilih dataset yang ingin dirandomisasi, pengguna tidak dapat membuat maupun impor matriks terlebih dahulu. Hal ini dikarenakan perlu ada proses pengecekan terlebih dahulu yang dilakukan perangkat lunak untuk memastikan dataset yang ingin dirandomisasi sudah sesuai persyaratan 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 5.10. 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 Randomisasi

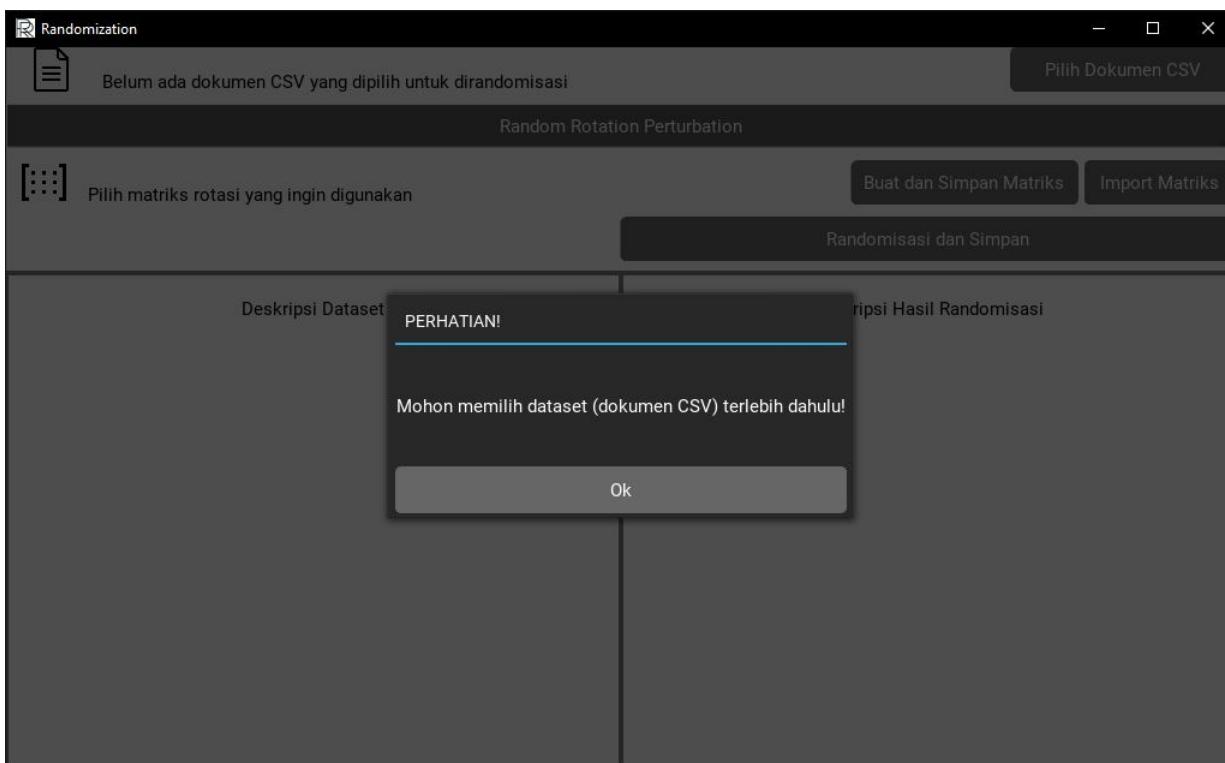
Perangkat lunak hanya meminta kepada pengguna parameter untuk teknik *Random Projection Perturbation* saja apabila pengguna memilih teknik tersebut. Pada teknik *Random Rotation Perturbation* tidak ada parameter yang perlu pengguna berikan. Ada dua buah parameter yang perlu pengguna berikan yaitu variabel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



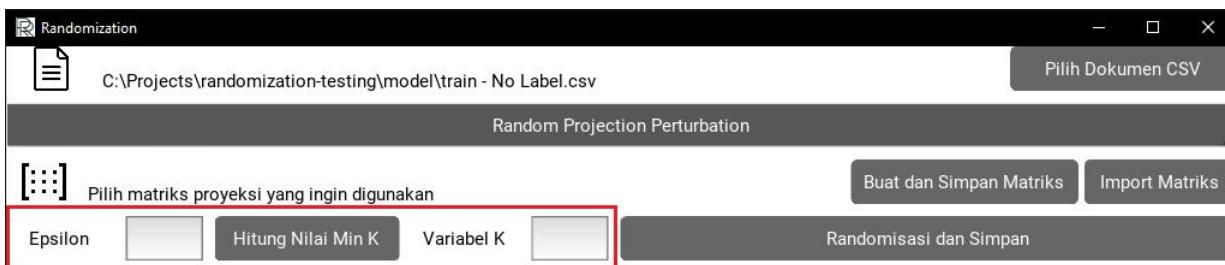
Gambar 5.8: Tampilan antarmuka saat perangkat lunak membuat dan menyimpan matriks



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



Gambar 5.10: Tampilan *popup* yang ditampilkan apabila pengguna belum memilih dataset yang ingin dirandomisasi

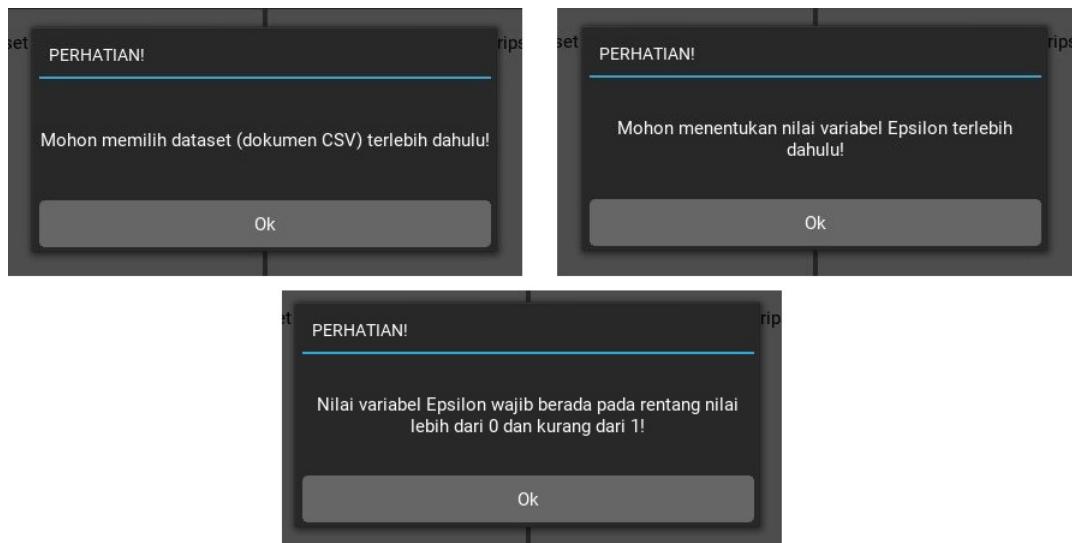


Gambar 5.11: Tampilan antarmuka parameter teknik randomisasi *Random Projection Perturbation*

parameter tersebut dapat dilihat antarmukanya pada Gambar 5.11

Seperti yang disinggung pada subbab sebelumnya antarmuka perangkat lunak akan menyesuaikan secara otomatis sesuai teknik yang dipilih pengguna. Pada bagian parameter teknik randomisasi, perangkat lunak akan menyembunyikan antarmuka parameter *Random Projection Perturbation* apabila pengguna memilih teknik *Random Rotation Perturbation*. Antarmuka tersebut dapat dilihat pada Gambar 5.3 yang dikelilingi oleh kotak berwarna merah dan bermotif 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 ϵ . Oleh karena itu, sebelum tombol ini dapat berfungsi, pengguna harus memilih terlebih dahulu dataset yang ingin dirandomisasi dan memberikan masukan nilai variabel ϵ yang sesuai dengan persyaratan variabel ϵ 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



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

menampilkan *popup* peringatan yang dapat dilihat pada Gambar 5.12. 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 k melebihi dimensi dari dataset yang ingin dirandomisasi karena salah satu persyaratan dari teknik *Random Projection Perturbation* adalah nilai variabel k harus lebih kecil daripada jumlah dimensi pada dataset yang ingin dirandomisasi. 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.

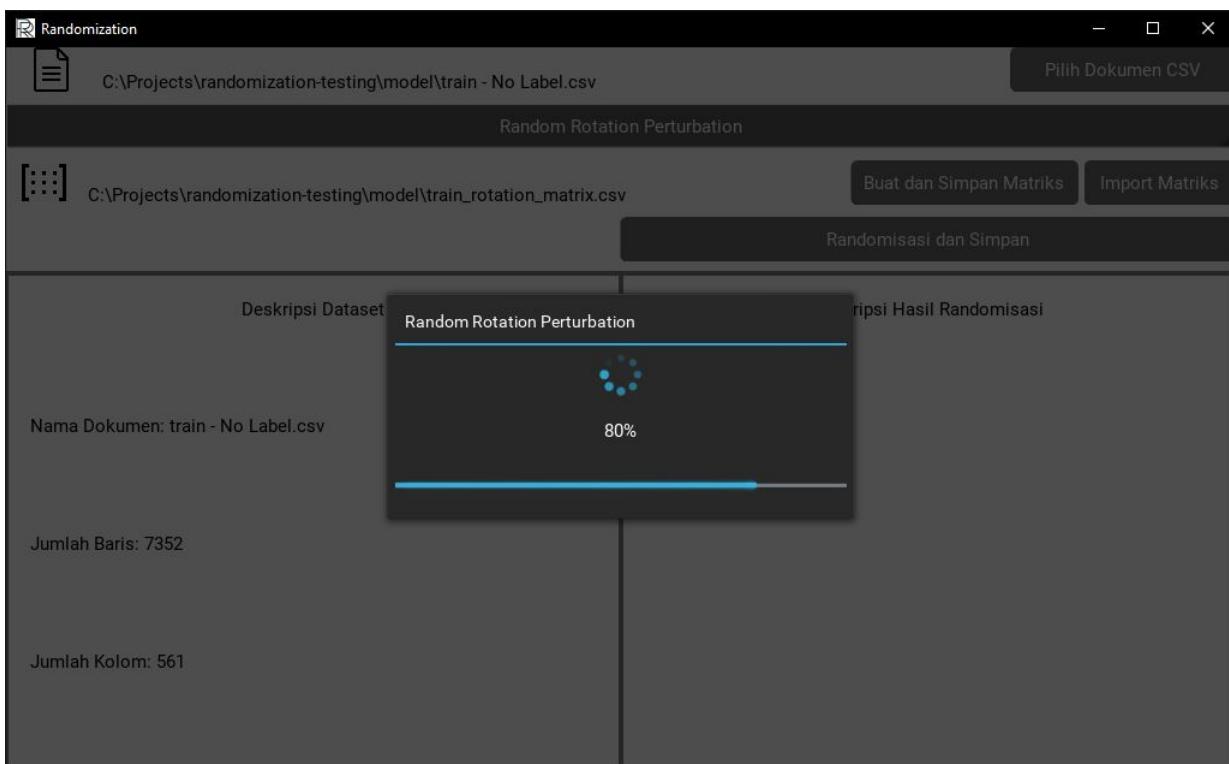
Randomisasi dan Simpan

Setelah pengguna memberikan masukan yang sesuai dan mengatur pengaturan yang diinginkan maka pengguna telah dapat melakukan randomisasi dengan menekan tombol “Randomisasi dan Simpan”. Tombol ini akan menerapkan teknik randomisasi yang dipilih oleh pengguna terhadap dataset yang ingin dirandomisasi menggunakan matriks yang telah dibuat atau dipilih oleh pengguna dan parameter-parameter yang pengguna berikan. Tampilan antarmuka saat proses randomisasi dilakukan perangkat lunak dapat dilihat pada Gambar 5.13.

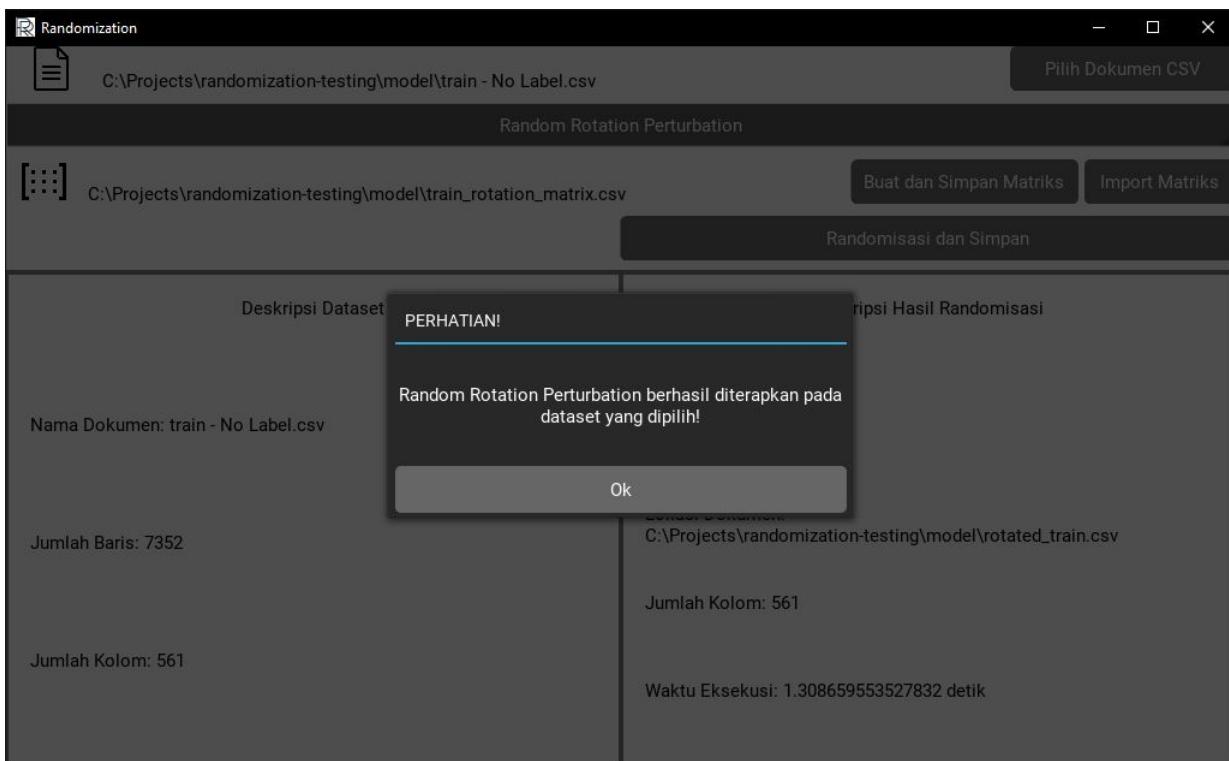
Setelah perangkat lunak berhasil melakukan randomisasi, perangkat lunak akan meminta pengguna untuk memilih direktori tempat penyimpanan dan nama dokumen hasil randomisasi. Perangkat lunak akan menyimpan hasil randomisasi dalam bentuk dokumen berjenis *comma-separated values*. Jendela baru untuk memilih 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 randomisasi dan dianggap gagal. Tampilan antarmuka *popup* yang akan tampil setelah perangkat lunak berhasil melakukan proses randomisasi dan menyimpan hasilnya pada direktori yang pengguna pilih dapat dilihat pada Gambar 5.14. Perangkat lunak juga akan menampilkan berbagai informasi hasil randomisasi pada bagian antarmuka deskripsi hasil randomisasi yang akan dijelaskan pada subbab berikutnya.

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

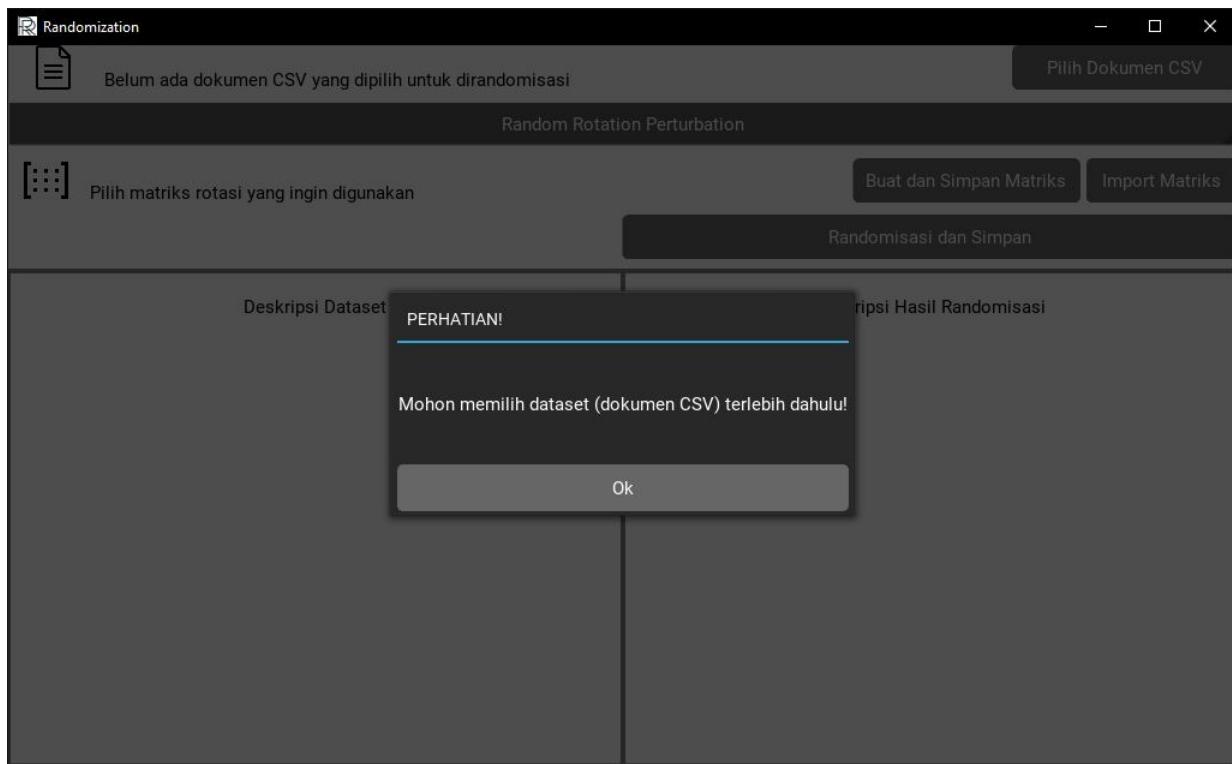
- Memilih dataset yang ingin dirandomisasi
- Memilih teknik randomisasi yang diinginkan



Gambar 5.13: Tampilan saat perangkat lunak sedang melakukan proses randomisasi



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



Gambar 5.15: Tampilan *popup* peringatan apabila pengguna belum memilih dataset yang diinginkan untuk dirandomisasi

- Membuat atau memilih matriks rotasi atau proyeksi
- Memberikan masukan nilai variabel *epsilon* dan nilai variabel *k*

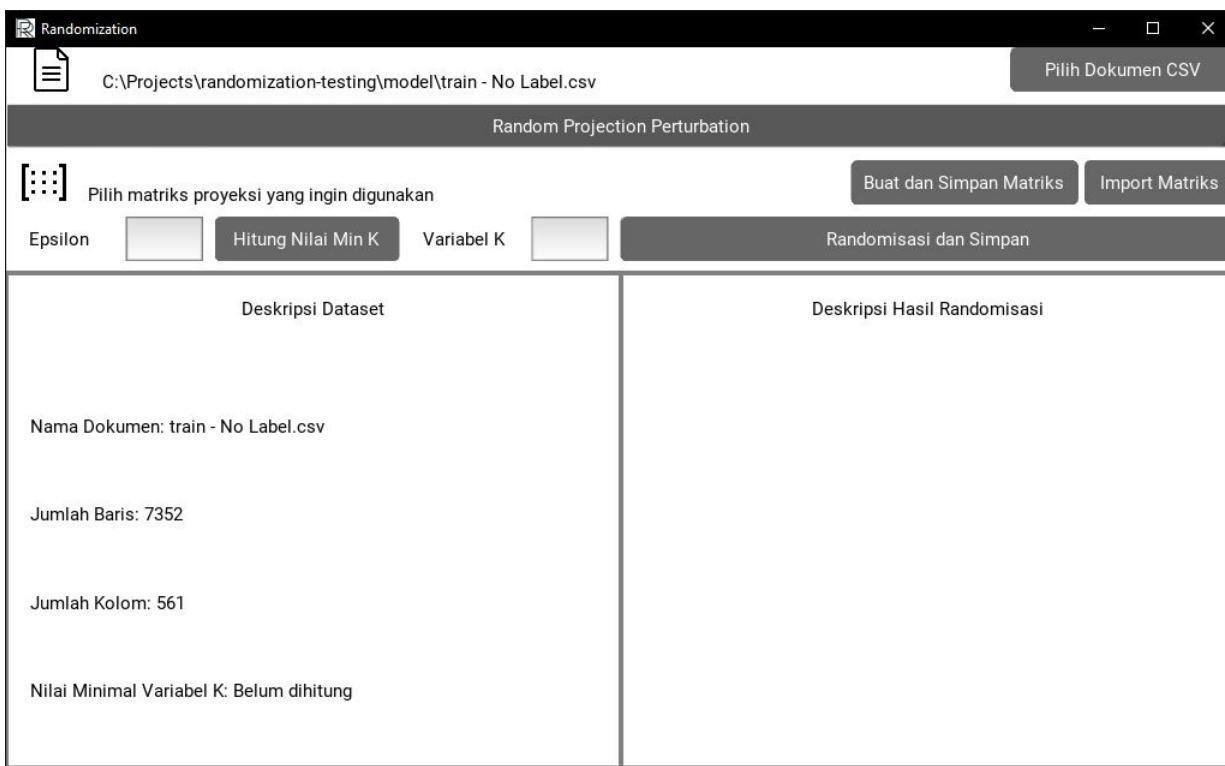
Persyaratan terakhir hanya berlaku apabila pengguna memilih teknik *Random Projection Perturbation*. Apabila ada persyaratan yang tidak dipenuhi oleh pengguna maka perangkat lunak akan menampilkan *popup* untuk memberikan peringatan kepada pengguna dan perangkat lunak tidak akan melanjutkan proses randomisasi.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

5.1.2 Deskripsi Dataset

Pengguna dapat melihat berbagai informasi dokumen *comma-separated values* yang dipilih sebagai dataset yang ingin dirandomisasi 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 menampilkannya pada deskripsi dataset. Nilai minimal variabel *k* hanya ditampilkan apabila pengguna memilih teknik *Random Projection Perturbation*.



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

Bagian antarmuka deskripsi dataset ini akan selalu secara otomatis diperbarui setiap pengguna memilih dataset baru. Tampilan antarmuka bagian deskripsi dataset dapat dilihat pada Gambar 5.2 yang dikelilingi oleh kotak berwarna biru dan bernomor dua. Apabila pengguna telah memilih dataset yang diinginkan untuk dirandomisasi maka perangkat lunak secara otomatis akan memperbarui tampilan antarmuka deskripsi dataset yang dapat dilihat pada Gambar 5.16

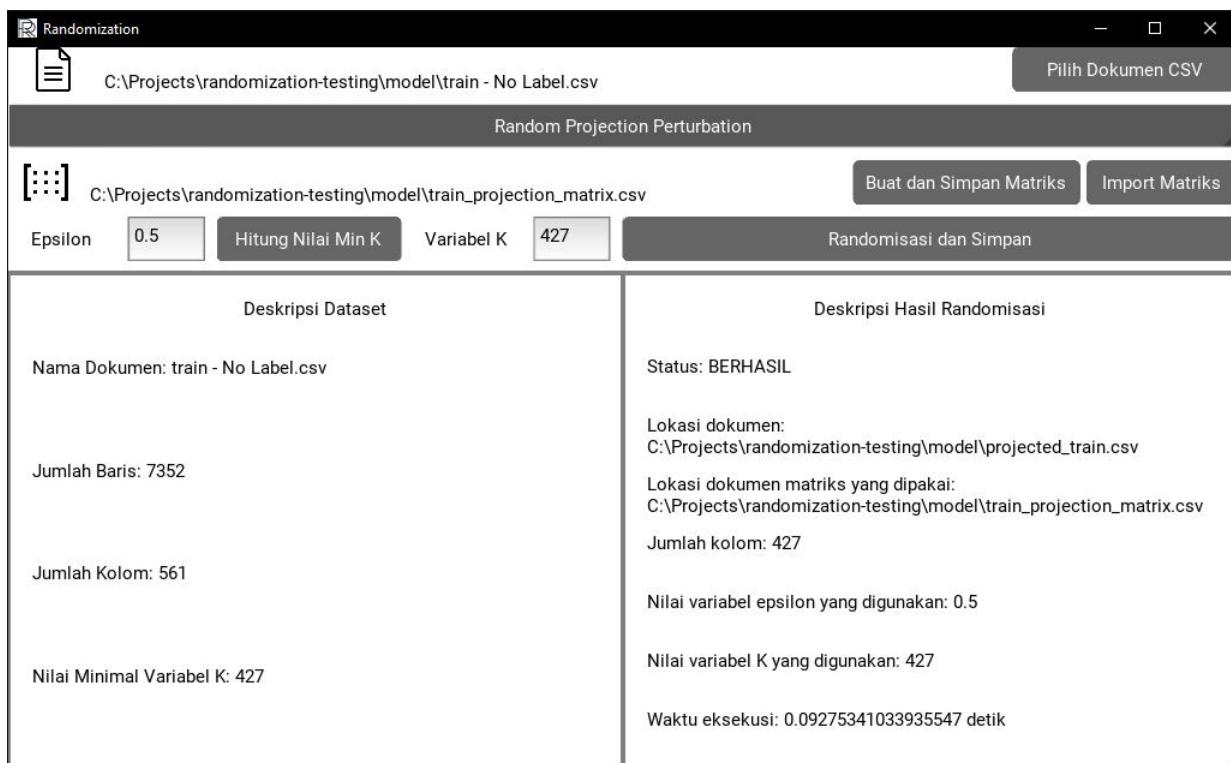
5.1.3 Deskripsi Hasil Randomisasi

Perangkat lunak akan menampilkan isi dari deskripsi hasil randomisasi setelah pengguna menekan tombol “Randomisasi dan Simpan” dan perangkat lunak melakukan proses randomisasi. Bagian deskripsi hasil randomisasi 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 randomisasi *Random Projection Perturbation*.

Bagian antarmuka deskripsi dataset ini akan selalu secara otomatis diperbarui setiap pengguna memilih dataset baru. Tampilan antarmuka bagian deskripsi dataset dapat dilihat pada Gambar 5.2



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

yang dikelilingi oleh kotak berwarna biru dan bernomor dua. Apabila pengguna telah memilih dataset yang diinginkan untuk dirandomisasi maka perangkat lunak secara otomatis akan memperbarui tampilan antarmuka deskripsi dataset yang dapat dilihat pada Gambar 5.17

5.2 Pengujian Fungsional

Pengujian fungsional bertujuan untuk memastikan perangkat lunak randomisasi dapat menerapkan kedua teknik randomisasi 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 dirandomisasi. Berikut pengujian pada setiap teknik randomisasi dengan menerapkan teknik penambangan data. Berikut pengujian pada setiap teknik randomisasi dengan menerapkan teknik penambangan data.

5.2.1 Teknik *Random Rotation Perturbation*

Pengujian teknik *Random Rotation Perturbation* akan menggunakan dataset *mall_customers* yang berisi informasi pribadi pelanggan sebuah mall. Dataset ini memiliki 4 buah fitur yaitu jenis kelamin, umur, penghasilan, dan skor pengeluaran. Empat buah fitur tersebut akan dirandomisasi 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 dirandomisasi juga diharapkan masih dapat diterapkan teknik penambangan data dengan hasil yang kualitasnya sama dengan dataset asli, pengujian untuk hal ini akan dilakukan pada bagian berikutnya yaitu pada pengujian eksperimental.

Pada teknik *Random Rotation Perturbation* untuk merotasikan sebuah dataset diperlukan sebuah matriks rotasi dan translasi acak. Perangkat lunak diharapkan dapat membuat kedua

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*

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 terandomisasi. Dataset *mall_customers* mempunyai 3 buah fitur yang akan dirandomisasi, oleh karena itu matriks rotasi dan translasi acak yang dibuat perangkat lunak seharusnya berukuran 3×3 kecuali matriks translasi yang akan berukuran 4×4 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 5.18. Matriks rotasi dan translasi ini akan digunakan untuk menerapkan teknik *Random Rotation Perturbation* terhadap dataset *mall_customers*. Perangkat lunak juga 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 dirandomisasi masing-masing pada Gambar 5.19 dan Gambar 5.20. Dapat dilihat pada gambar tersebut, dataset setelah dirandomisasi 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 dirandomisasi pada baris yang sama seperti pada kolom *insulin* baris 10 sampai 12 memiliki nilai 19 pada dataset asli tetapi pada dataset yang telah dirandomisasi 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. 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 Rotation Perturbation*.

Dalam rangka memastikan lebih lagi apakah perangkat lunak menerapkan teknik *Random Rotation Perturbation* dengan akurat, perhitungan manual pada baris pertama dataset dapat dilihat sebagai berikut «TODO»

5.2.2 Teknik *Random Projection Perturbation*

Pengujian teknik *Random Projection Perturbation* akan menggunakan dataset *mobile_sensor* 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 dan sebuah label. Seluruh fitur tersebut akan dirandomisasi dan diharapkan hasilnya akan mengacak dataset sehingga nilai tiap fitur tersebut berbeda dari aslinya. Dataset yang telah dirandomisasi juga diharapkan masih dapat diterapkan teknik penambangan data dengan hasil yang kualitasnya mirip dengan dataset asli, pengujian untuk hal ini akan dilakukan pada bagian

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 dirandomisasi

berikutnya yaitu pada pengujian eksperimental.

Label pada dataset *mobile_sensor* yang ingin dirandomisasi harus dihilangkan terlebih dahulu agar hanya fitur (bersifat numerik) pada dataset tersebut saja yang terandomisasi. 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 5.1 adalah contoh perhitungan manual rumus nilai minimal variabel *k* dengan nilai variabel *epsilon* sebesar 0.52 dan jumlah baris pada dataset sebanyak 10299 baris.

$$\begin{aligned}
 k &= \frac{4 \ln n}{\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}} \\
 &= \frac{4 \ln 10299}{\frac{0.52^2}{2} - \frac{0.52^3}{3}} \\
 &= \frac{36.9592}{0.1352 - 0.0468} \\
 &= 418.0905
 \end{aligned} \tag{5.1}$$

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 dirandomisasi dan sesuai dengan nilai minimal variabel *k* yang sudah dihitung sebelumnya yaitu sebesar 418 maka ditentukan nilai *k* yang dipakai pada pengujian ini adalah sebesar 427. Oleh karena itu, matriks proyeksi acak yang dibuat perangkat lunak seharusnya berukuran 427×427 .

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 10 kolom pertama pada 20 baris pertamanya dapat dilihat pada Gambar 5.21. Matriks proyeksi ini akan digunakan untuk menerapkan teknik *Random Projection Perturbation* terhadap dataset *mobile_sensor*. Perangkat lunak juga 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 *k* yang dipilih. Persyaratan tersebut didasarkan oleh sifat pada teknik *Random Projection Perturbation* yaitu nilai minimal variabel *k* berbanding lurus dengan banyaknya data.

Berikut akan ditampilkan 7 fitur terakhir serta label pada 20 baris pertama dataset asli dan dataset yang telah dirandomisasi masing-masing pada Gambar 5.22 dan Gambar 5.23. Dapat dilihat pada gambar tersebut, dataset setelah dirandomisasi memiliki nilai yang berbeda dengan aslinya. Terlebih lagi setiap fitur pada dataset yang telah dirandomisasi 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 dirandomisasi. 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

	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*

angle(tBodyAcc)	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 dirandomisasi

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 dirandomisasi mempunyai distorsi yang lebih besar dari harapan pengguna. Hasil dari perangkat lunak diharapkan (dataset yang telah dirandomisasi) memenuhi Pertidaksamaan 5.2.

$$(1 - \text{eps})\|u - v\|^2 < \|p(u) - p(v)\|^2 < (1 + \text{eps})\|u - v\|^2 \quad (5.2)$$

Pada pengujian ini, dengan nilai variabel *epsilon* sebesar 0.52 Pertidaksamaan 5.2 terpenuhi pada seluruh data yang ada pada dataset. Ada salah satu kasus saat Pertidaksamaan 5.2 tidak terpenuhi yaitu apabila nilai variabel *epsilon* ditentukan sebesar 0.4. Salah satu jarak Euclidean yang melanggar Pertidaksamaan 5.2 adalah baris 473 dengan baris 1306 yang memiliki jarak Euclidean sebesar 8.167734238223167 pada dataset asli dan 9.723888530285468 pada dataset yang telah dirandomisasi. Pengujian ini membuktikan bahwa perangkat lunak berhasil menerapkan teknik *Random Projection Perturbation* dengan akurat sesuai batas distorsi yang ditentukan pengguna.

5.3 Pengujian Eksperimental

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

1. Penambangan Data Klasifikasi
 - (a) *Random Rotation Perturbation* dengan dataset *diabetes*
 - (b) *Random Projection Perturbation* dengan dataset *mobile_sensor*
2. Penambangan Data *Clustering*
 - (a) *Random Rotation Perturbation* dengan dataset *mall_customers*

(b) *Random Projection Perturbation* dengan dataset *mobile_sensor*

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.

5.3.1 Penambangan Data Klasifikasi

Pengujian dengan penambangan data klasifikasi akan berpusat pada pembuatan model dengan dataset asli dan dataset yang telah dirandomisasi 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 dirandomisasi dengan teknik *Random Rotation Perturbation* dan dataset yang dirandomisasi dengan teknik *Random Projection Perturbation*. Pengujian akan membandingkan beberapa informasi pada dataset asli dan dataset yang telah dirandomisasi. Beberapa informasi tersebut dijabarkan sebagai berikut.

1. Properti-properti pada dataset yaitu rata-rata, standar deviasi, nilai terkecil dan terbesar pada sebuah kolom, kuartil bawah, kuartil tengah dan kuartil atas.
2. Akurasi model *k-nearest neighbors*
3. Nilai variabel *k* yang modelnya memiliki akurasi tertinggi
4. 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 dirandomisasi.

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 dirandomisasi dengan teknik *Random Rotation Perturbation* dan hasil randomisasi 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.

Properti-properti 4 kolom terakhir pada dataset *diabetes* asli dapat dilihat pada Tabel 5.1. Sementara untuk dataset *diabetes* yang telah dirandomisasi dapat dilihat pada Tabel 5.2. Jika dilihat pada kedua tabel tersebut, seluruh properti pada dataset yang telah dirandomisasi 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 dan terbesar pada sebuah kolom, kuartil bawah, kuartil tengah dan kuartil atas.

Teknik penambangan data *k-nearest neighbors* diterapkan menggunakan 8 buah fitur yang ada untuk menguji apakah dataset asli dan dataset yang telah dirandomisasi 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 *k*) dari 1 sampai 20. Implementasi kode teknik penambangan data ini diterapkan dengan bahasa pemrograman 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 dirandomisasi. Apabila dibandingkan, kedua grafik tersebut terlihat memiliki nilai akurasi yang sama persis untuk setiap nilai *k*.

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

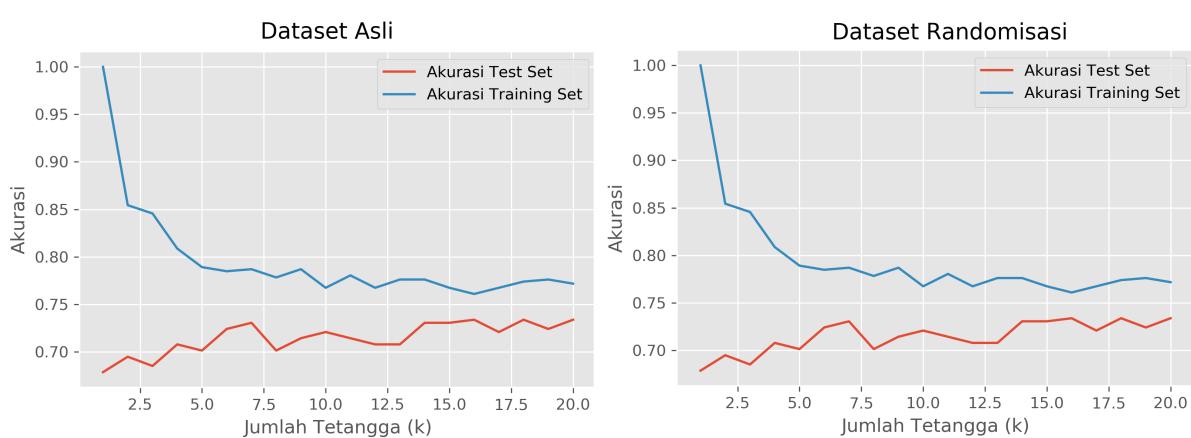
Gambar 5.25: Dua puluh baris pertama dataset *diabetes* setelah dirandomisasi

Tabel 5.1: Properti-properti pada dataset *diabetes* asli

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

Tabel 5.2: Properti-properti pada dataset *diabetes* yang telah dirandomisasi

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

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

Akurasi model untuk setiap nilai k dari 14 sampai 18 pada dataset asli dan dataset yang telah dirandomisasi dapat dilihat masing-masing pada Listing 5.1 dan Listing 5.2. Akurasi tertinggi pada model k -nearest neighbors dengan dataset asli adalah sebesar 0.7337662337662337 dengan nilai k sebesar 16. Nilai yang sama juga muncul pada dataset yang telah dirandomisasi. Hal ini dapat menjadi bukti bahwa teknik *Random Rotation Perturbation* menjaga jarak Euclidean dengan sempurna, tidak ada perubahan sama sekali pada jarak Euclidean antara seluruh titik. Oleh karena itu model k -nearest neighbors yang terbuat memiliki hasil yang sama persis.

Listing 5.1: Akurasi Dataset Asli

```
Akurasi setiap K pada
training set dataset asli:
14: 0.7760869565217391
15: 0.7673913043478261
16: 0.7608695652173914
17: 0.7673913043478261
18: 0.7739130434782608

Akurasi setiap K pada
test set dataset asli:
14: 0.7305194805194806
15: 0.7305194805194806
16: 0.7337662337662337
17: 0.7207792207792207
18: 0.7337662337662337
```

Listing 5.2: Akurasi Dataset Randomisasi

```
Akurasi setiap K pada training
set dataset randomisasi:
14: 0.7760869565217391
15: 0.7673913043478261
16: 0.7608695652173914
17: 0.7673913043478261
18: 0.7739130434782608

Akurasi setiap K pada test
set dataset randomisasi:
14: 0.7305194805194806
15: 0.7305194805194806
16: 0.7337662337662337
17: 0.7207792207792207
18: 0.7337662337662337
```

Waktu eksekusi yang dibutuhkan untuk melatih model klasifikasi dengan nilai k sebesar 16 memakai dataset asli adalah sebesar 0.0009965896606445312 detik dan waktu yang dibutuhkan untuk memprediksi *test set* sebesar 0.015623807907104492 detik. Sementara untuk dataset yang telah dirandomisasi membutuhkan waktu eksekusi untuk melatih model klasifikasi sebesar 0.0009937286376953125 detik dan waktu yang dibutuhkan untuk melakukan prediksi adalah sebesar 0.01565837860107422 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 dirandomisasi.

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 5.22. Dataset ini dirandomisasi dengan teknik *Random Projection Perturbation* dan hasil randomisasi dapat dilihat pada Gambar 5.23. 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.

Properti-properti 4 buah kolom pada dataset *mobile_sensor* asli dapat dilihat pada Tabel 5.4. Sementara untuk dataset *mobile_sensor* yang telah dirandomisasi dapat dilihat pada Tabel 5.4. Jika dilihat pada kedua tabel tersebut, seluruh properti pada dataset yang telah dirandomisasi 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.

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

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

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	angle(Y,gravityMean)	angle(Z,gravityMean)
count	10299.000000	10299.000000	10299.000000	10299.000000
mean	0.274347	-0.017743	0.063255	-0.054284
std	0.067628	0.037128	0.305468	0.268898
min	-1.000000	-1.000000	-1.000000	-1.000000
25%	0.262625	-0.024902	0.002151	-0.131880
50%	0.277174	-0.017162	0.182028	-0.003882
75%	0.288354	-0.010625	0.250790	0.102970
max	1.000000	1.000000	1.000000	1.000000

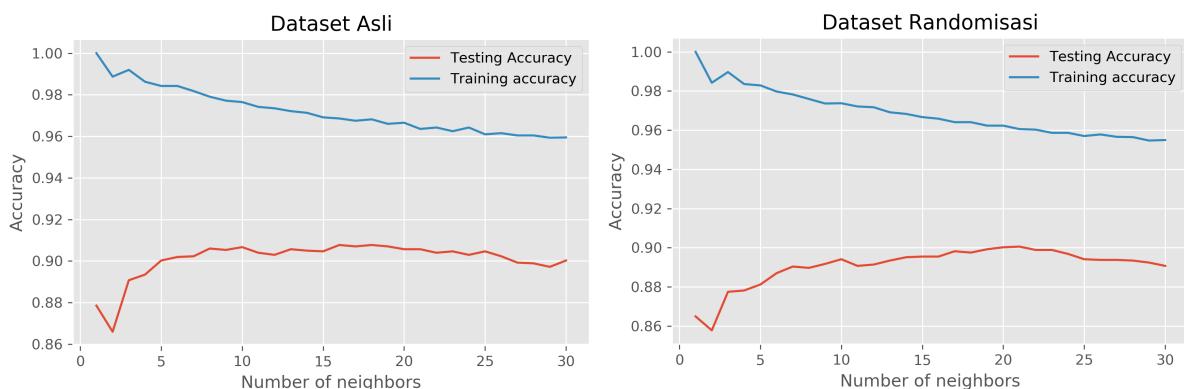
Tabel 5.4: Properti-properti pada dataset *mobile_sensor* yang telah dirandomisasi

	0	1	425	426
count	10299.000000	10299.000000	10299.000000	10299.000000
mean	0.136942	-0.289681	0.173610	0.353280
std	0.528322	0.266849	0.214992	0.475443
min	-1.433025	-1.206428	-1.437424	-0.976616
25%	-0.340174	-0.482918	0.033424	-0.049233
50%	0.308314	-0.270461	0.179048	0.322172
75%	0.582374	-0.097266	0.325611	0.688616
max	1.283867	0.726498	0.978304	1.786145

titik terjaga dengan besar distorsi yang ditentukan 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 k) dari 1 sampai 30. Implementasi kode teknik penambangan data ini diterapkan dengan bahasa pemrograman Python dan dibantu oleh *library* Scikit-learn.

Pada Gambar 5.27 dapat dilihat grafik akurasi model *k-nearest neighbors* dalam memprediksi label *training set* dan *test set* pada dataset asli dan dataset yang telah dirandomisasi. 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 nilai k yang memiliki akurasi tertinggi. Tetapi perbedaan nilai akurasinya masih mirip dengan dataset asli.

Akurasi model untuk setiap nilai k dari 19 sampai 23 pada dataset asli dan dataset yang telah dirandomisasi dapat dilihat masing-masing pada Listing 5.3 dan Listing 5.4. Dapat dilihat pada kedua listing tersebut akurasi *test set* pada kedua dataset berbeda. Akurasi *test set* tertinggi pada

Gambar 5.27: Grafik akurasi model klasifikasi pada *training set* dan *test set* dataset *mobile_sensor*

dataset asli adalah sebesar 0.9077027485578555 dengan nilai k sebesar 16. Sementara pada dataset yang telah dirandomisasi, akurasi *test set* tertingginya adalah sebesar 0.9005768578215134 dengan nilai k sebesar 21. Jika dihitung perbedaan akurasinya adalah sebesar 0.0071258907363421 yang mana relatif kecil tetapi ada perbedaan pada nilai k yang memiliki akurasi tertinggi. Dengan perbedaan akurasi yang relatif kecil dan dataset yang telah dirandomisasi teracak dengan baik 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 5.3: Akurasi Dataset 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 5.4: Akurasi Dataset Randomisasi

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

Akurasi setiap K pada test
set dataset randomisasi :
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 k sebesar 16 memakai dataset asli adalah sebesar 0.2872335910797119 detik dan waktu yang dibutuhkan untuk memprediksi *test set* sebesar 17.75454642303467 detik. Sementara untuk dataset yang telah dirandomisasi membutuhkan waktu eksekusi untuk melatih model klasifikasi dengan nilai k sebesar 21 adalah 0.5630350112915039 detik dan waktu yang dibutuhkan untuk melakukan prediksi adalah sebesar 12.86760687828064 detik. Pada dataset yang telah dirandomisasi waktu prediksi lebih cepat dikarenakan fitur-fitur yang ada pun 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 dirandomisasi.

Kesimpulan

Berdasarkan pengujian dengan teknik penambangan data klasifikasi menggunakan teknik *k-nearest neighbors* pada dataset asli, dataset yang telah dirandomisasi menggunakan teknik *Random Rotation Perturbation*, dan dataset yang telah dirandomisasi menggunakan teknik *Random Projection Perturbation* dibuat kesimpulan sekaligus membandingkan antara kedua teknik randomisasi. Pada Tabel 5.5 dapat dilihat hasil akhir pengujian antara model yang dilatih dengan dataset asli dan dataset yang telah dirandomisasi dengan *Random Rotation Perturbation* dan *Random Projection Perturbation*.

Metode *Randomization* tidak menjaga properti-properti pada data selain jarak Euclidean. Akurasi model *k-nearest neighbors* terjaga dengan baik pada kedua teknik randomisasi yang dikarenakan oleh jarak Euclidean tetap terjaga setelah dataset dirandomisasi. 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*

Tabel 5.5: Perbandingan Model *k-nearest neighbors* dan Properti Data

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

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 dirandomisasi dengan teknik *Random Projection Perturbation*. Waktu eksekusinya lebih cepat dikarenakan oleh data pada dataset lebih sedikit karena dimensinya direduksi.

5.3.2 Penambangan Data *Clustering*

Pengujian dengan penambangan data *clustering* akan berpusat pada pembuatan model *clustering* dengan dataset asli dan dataset yang telah dirandomisasi 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 dirandomisasi. Beberapa informasi tersebut dijabarkan sebagai berikut.

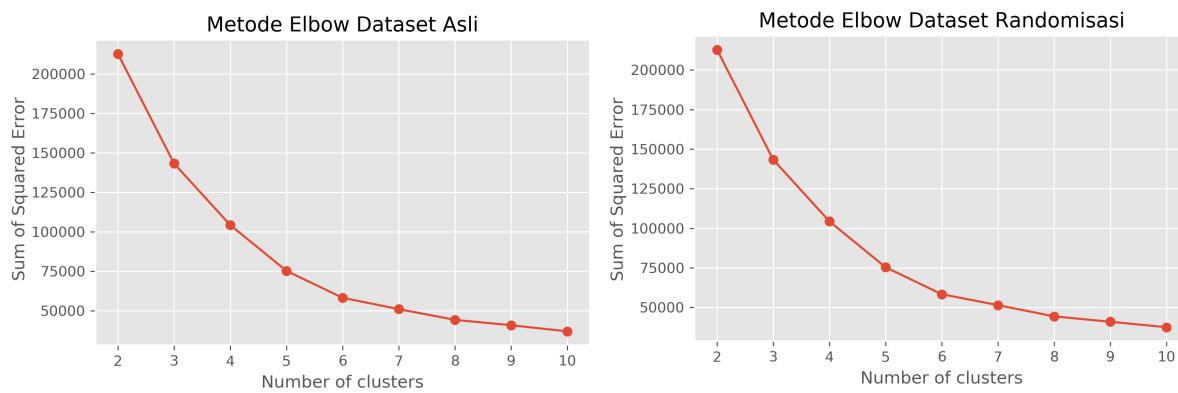
1. *Sum of Squared Error*
2. *Silhouette Score*
3. Nilai variabel *k* (jumlah *cluster*) yang modelnya memiliki kualitas *clustering* terbaik
4. Visualisasi *cluster*
5. Waktu eksekusi pelatihan model dan prediksi dengan model yang telah dibuat

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 dirandomisasi. Selain itu metode *Adjusted Rand Index* juga digunakan untuk menghitung kemiripan antara kedua model.

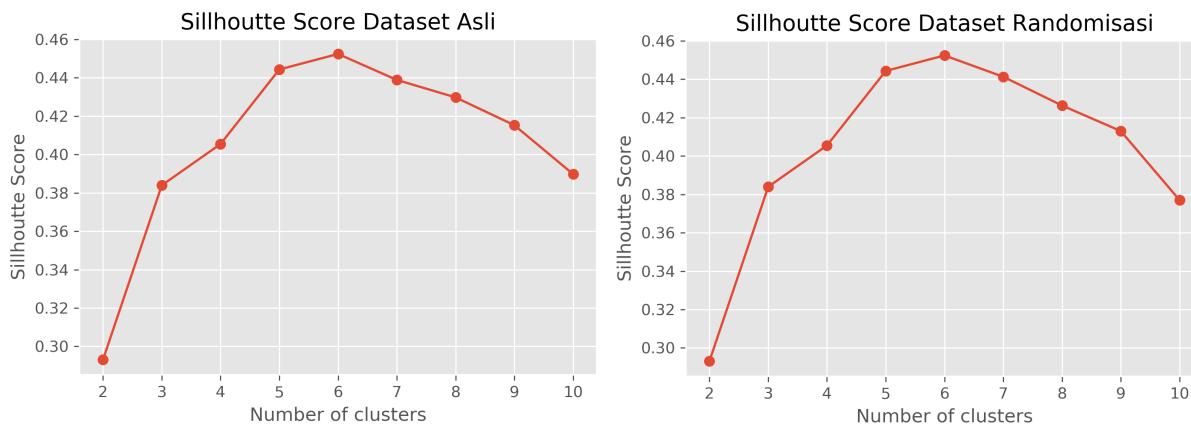
Random Rotation Perturbation

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 5.19. Dataset ini dirandomisasi dengan teknik *Random Rotation Perturbation* dan hasil randomisasi dapat dilihat pada Gambar 5.20. 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 *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 *k* yang terbaik untuk dipakai. Pada Gambar 5.28 terdapat grafik *Sum of Squared Error* untuk menggunakan metode Elbow pada dataset asli dan dataset yang telah dirandomisasi. Terlihat nilai *k* sebesar 5, 6, dan 7 adalah kandidat yang terbaik untuk menjadi nilai *k* yang dipakai pada dataset asli maupun dataset randomisasi. 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 *k* yang terbaik antara ketiga nilai tersebut, *Silhouette Score* dapat digunakan untuk metode alternatif untuk menentukan nilai *k* yang terbaik.



Gambar 5.28: Grafik *Sum of Squared Error* untuk menggunakan metode Elbow untuk mencari nilai k terbaik



Gambar 5.29: Grafik *Silhouette Score* model *clustering* pada dataset *mall_customers*

Pada Gambar 5.29 terdapat grafik *Silhouette Score* dari dataset *mall_customers* asli dan yang telah dirandomisasi. 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 k 6 adalah nilai paling besar yaitu 0.4523443947724053 pada dataset asli dan 0.4523443947780976 pada dataset yang telah dirandomisasi. Hal ini menunjukkan teknik *Random Rotation Perturbation* tidak mempengaruhi secara signifikan nilai *Silhouette Score* pada setiap k .

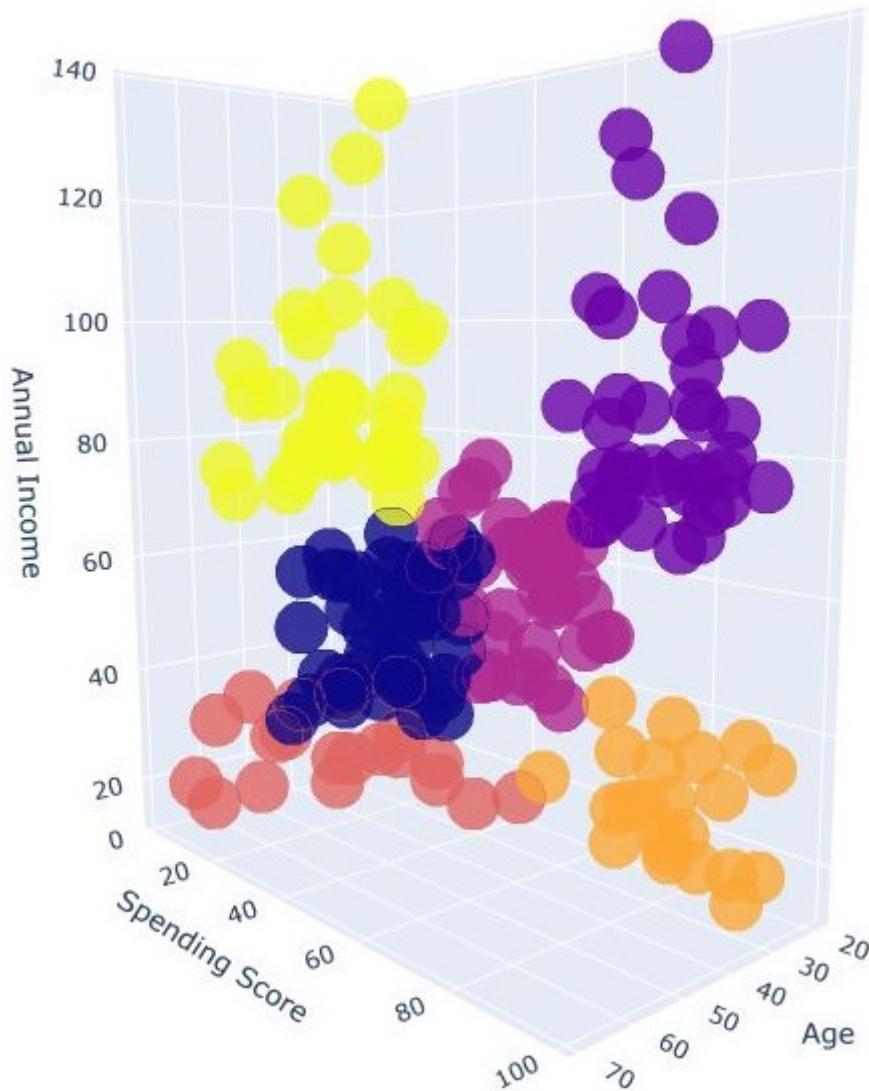
Listing 5.5: Silhouette Score Asli

```
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 5.6: Silhouette Score Randomisasi

```
Silhouette Score setiap K
pada dataset randomisasi:
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 4 buah fitur yang ada untuk

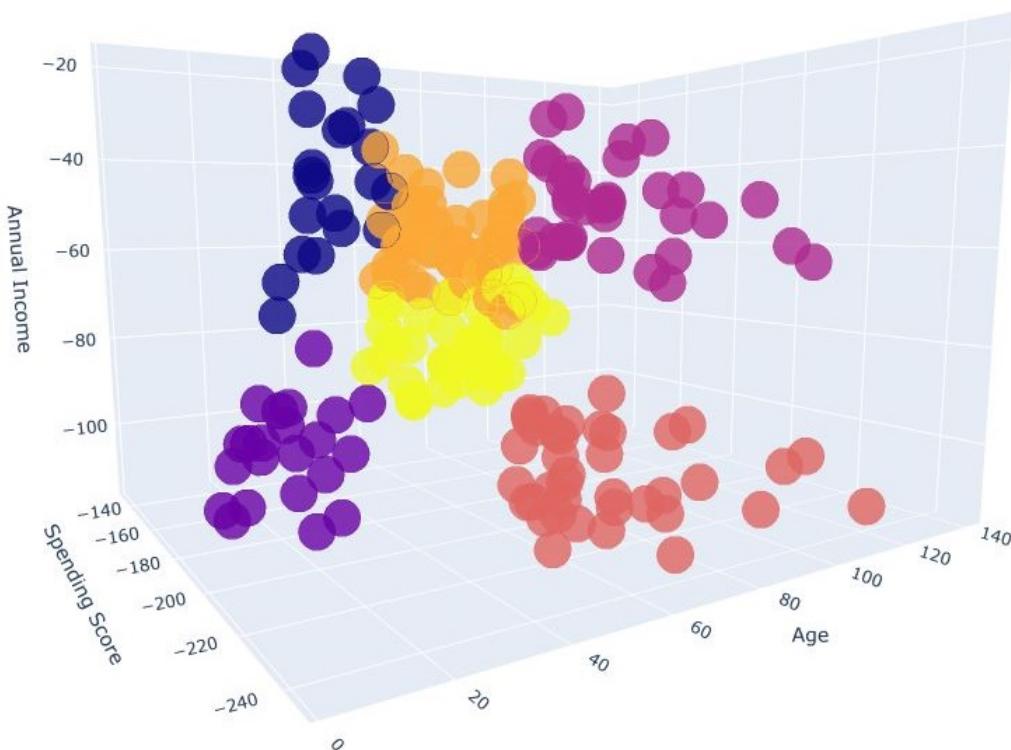


Gambar 5.30: Visualisasi *cluster* pada dataset yang asli

menguji apakah dataset asli dan dataset yang telah dirandomisasi 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 setiap titik tidak berubah sama sekali tetapi merotasi seluruh titik yang ada pada bidang Euclidean. Visualisasi *cluster* pada dataset asli dan dataset yang telah dirandomisasi masing-masing dapat dilihat pada Gambar 5.30 dan Gambar 5.31. Implementasi kode teknik penambangan data ini diterapkan dengan bahasa pemrograman Python dan dibantu oleh *library* Scikit-learn. Visualisasi model *clustering* dengan nilai k sebesar 6 pada dataset *mall_customers* asli dan yang telah dirandomisasi dapat dilihat pada Gambar 5.30 dan Gambar 5.31.

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 0.02995157241821289 detik pada dataset yang asli dan 0.032910823822021484 detik pada dataset yang telah dirandomisasi, hal ini menunjukkan bahwa teknik *Random Rotation Perturbation* tidak mempengaruhi secara signifikan waktu eksekusi untuk melatih model *k-means*.



Gambar 5.31: Visualisasi *cluster* pada dataset yang telah dirotasi

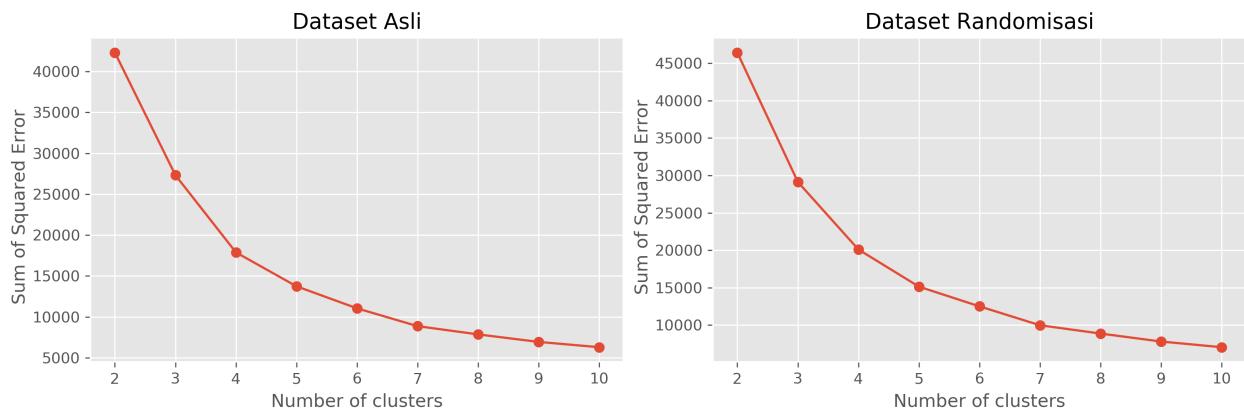
Random Projection Perturbation

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 5.22. Dataset ini dirandomisasi dengan teknik *Random Projection Perturbation* dan hasil randomisasi dapat dilihat pada Gambar 5.23. 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.

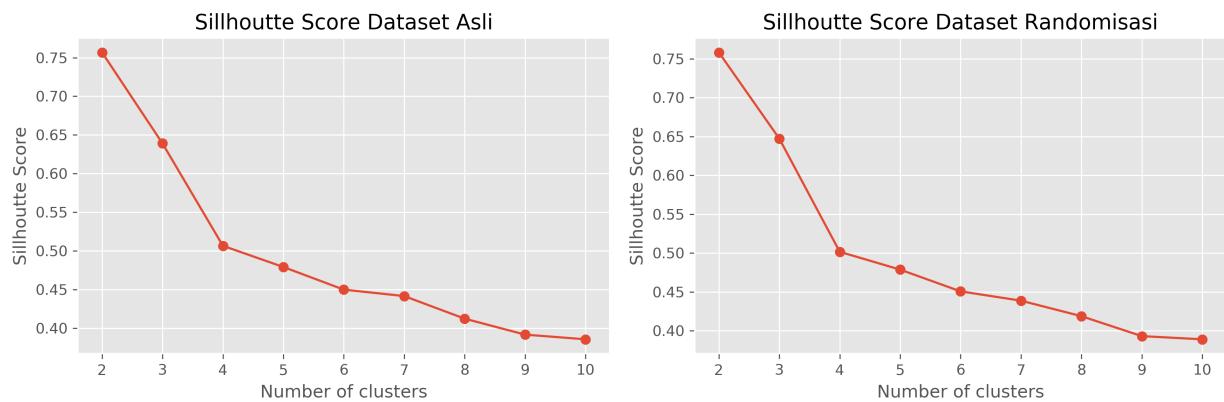
Sebelum melakukan teknik *clustering* dengan algoritma *k-means*, dataset yang memiliki fitur yang sangat banyak tersebut harus direduksi terlebih dahulu agar model dapat divisualisasikan dengan mudah. Dataset yang akan di-*cluster* akan direduksi dimensinya sampai hanya memiliki 2 dimensi. Reduksi dimensi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

Pada Gambar 5.32 terdapat grafik *Sum of Squared Error* untuk menggunakan metode Elbow pada dataset asli dan dataset yang telah dirandomisasi. Pada grafik ini tidak terlalu terlihat nilai yang terbaik untuk menjadi nilai k yang dipakai pada dataset asli maupun dataset randomisasi. 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 k yang terbaik untuk dipakai membuat model, *Silhouette Score* dapat digunakan untuk metode alternatif untuk menentukan nilai k yang terbaik.

Pada Gambar 5.33 terdapat grafik *Silhouette Score* dari dataset *mobile_sensor* yang asli dan yang telah dirandomisasi. Dapat terlihat kedua grafik *Silhouette Score* terlihat sangat mirip dan dapat dilihat nilai-nilai pada setiap k tersebut di Listing 5.7 dan Listing 5.8 ada sedikit perbedaan yang tidak terlalu signifikan. Hal ini dikarenakan oleh jarak Euclidean pada dataset asli dan



Gambar 5.32: Grafik *Sum of Squared Error* untuk menggunakan metode Elbow untuk mencari nilai k terbaik



Gambar 5.33: Grafik *Silhouette Score* model *clustering* pada dataset *mobile_sensor*

yang telah dirandomisasi sedikit berbeda sehingga mempengaruhi sedikit pada hasil algoritma *Silhouette Score*. Dapat terlihat *Silhouette Score* pada nilai k sebesar 2 adalah nilai paling besar yaitu 0.7568010158989575 pada dataset asli dan 0.7581796759180272 pada dataset yang telah dirandomisasi. Hal ini menunjukkan teknik *Random Projection Perturbation* tidak mempengaruhi secara signifikan nilai *Silhouette Score* pada setiap k .

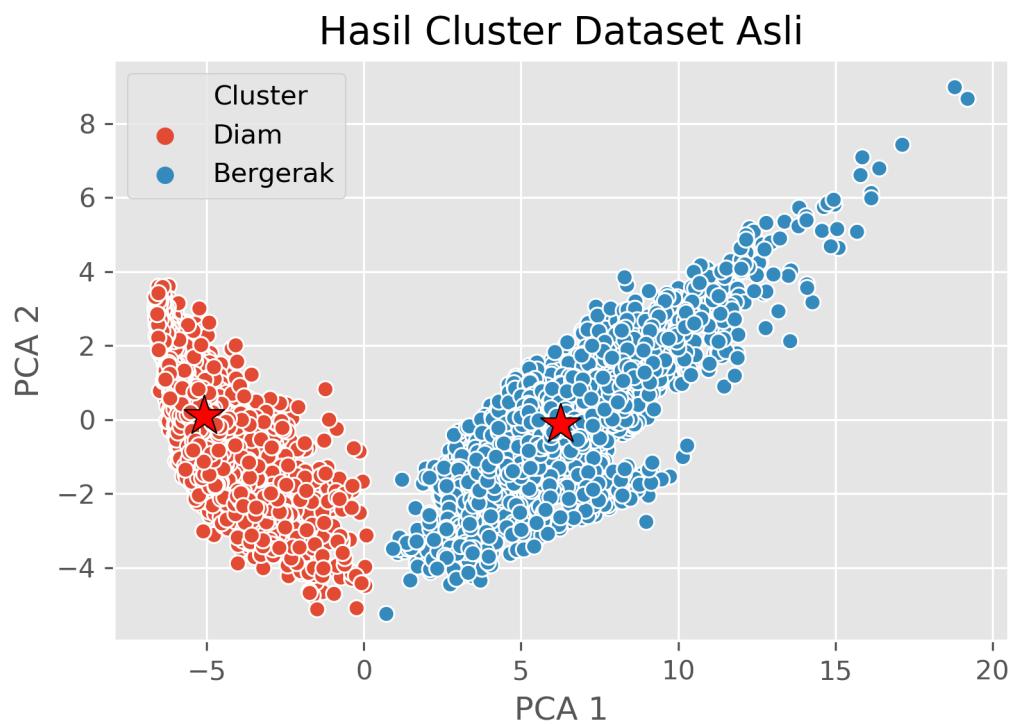
Listing 5.7: Silhouette Score Asli

```
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 5.8: Silhouette Score Randomisasi

```
Silhouette Score setiap K
pada dataset randomisasi :
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
```

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



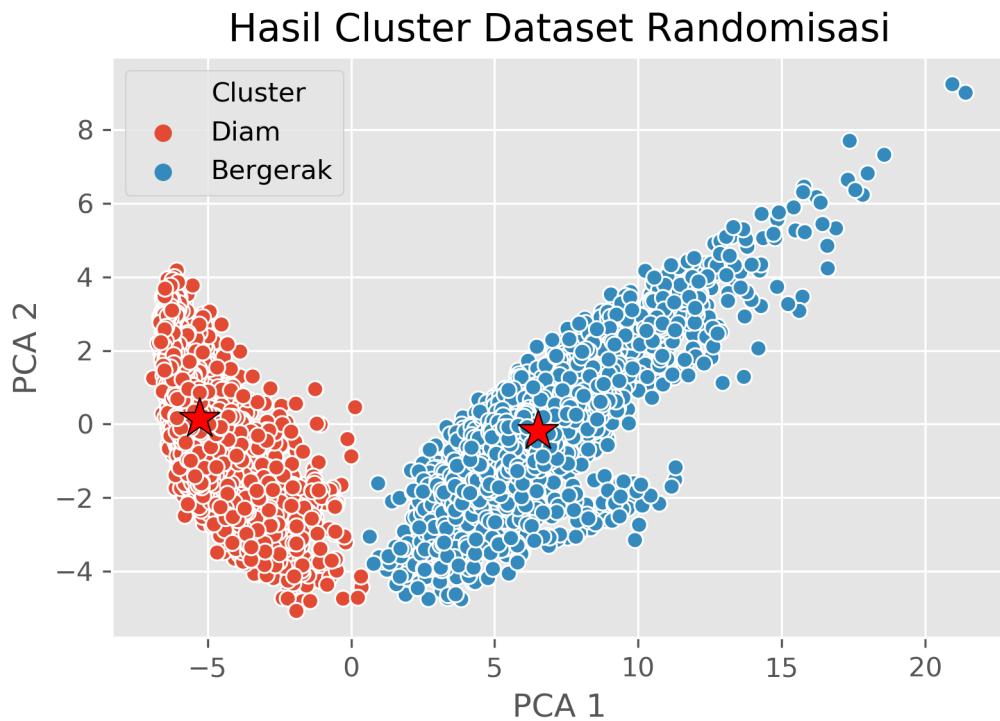
Gambar 5.34: Visualisasi *cluster* pada dataset yang asli

distorsi yang ditentukan pengguna. Model *k-means* akan dibuat dengan nilai k yang terbaik dihitung menggunakan metode Elbow dan nilai 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 k sebesar 2 pada dataset *mobile_sensor* asli dan yang telah dirandomisasi dapat dilihat pada Gambar 5.34 dan Gambar 5.35. 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 k sebesar 2 adalah sebesar 0.031239032745361328 detik pada dataset asli dan 0.031217575073242188 detik pada dataset yang telah dirandomisasi. 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 eksekusinya karena ukuran dataset berkurang sehingga waktu eksekusinya juga berkurang.



Gambar 5.35: Visualisasi *cluster* pada dataset yang telah diproyeksi

Tabel 5.6: Perbandingan Model *k-means*

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

Kesimpulan

Berdasarkan pengujian dengan teknik penambangan data *clustering* menggunakan teknik *k-means* pada dataset asli, dataset yang telah dirandomisasi menggunakan teknik *Random Rotation Perturbation*, dan dataset yang telah dirandomisasi menggunakan teknik *Random Projection Perturbation* dibuat kesimpulan sekaligus membandingkan antara kedua teknik randomisasi. Pada Tabel 5.6 dapat dilihat hasil akhir pengujian antara model yang dilatih dengan dataset asli dan dataset yang telah dirandomisasi dengan *Random Rotation Perturbation* dan *Random Projection Perturbation*.

Model *clustering* yang dilatih oleh dataset yang dirandomisasi sangat mirip dengan dataset asli. *Sum of Squared Error* dan *Silhouette Score* pada kedua teknik randomisasi memiliki hasil yang sangat mirip dengan aslinya sehingga metode Elbow dan pemilihan fitur dengan *Silhouette Score* dapat dilakukan setelah dataset dirandomisasi. Jumlah *cluster* (nilai variabel *k*) sama pada kedua teknik dengan sebelum dirandomisasi. Visualisasi *cluster* pada kedua teknik sedikit berbeda karena teknik *Random Rotation Perturbation* merotasi seluruh data sehingga visualisasinya akan terotasi dan teknik *Random Projection Perturbation* tidak menjaga jarak Euclidean secara sempurna. *Adjusted Rand Index* pada kedua teknik memiliki nilai yang baik yaitu bernilai 1 atau mendekati 1. Hal ini menunjukkan bahwa model *clustering* antara yang dilatih dengan dataset yang telah dirandomisasi dan dataset aslinya sama persis atau sangat mirip. Waktu eksekusi pembuatan model dan prediksi

hanya memiliki perbedaan pada model yang menggunakan dataset yang telah dirandomisasi dengan teknik *Random Projection Perturbation*. Waktu eksekusinya lebih cepat dikarenakan oleh data pada dataset lebih sedikit karena dimensinya direduksi.

5.3.3 Kesimpulan Akhir

Berdasarkan hasil pengujian eksperimental 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 secara sempurna karena ada distorsi yang terkontrol pada jarak Euclidean setiap titik pada dataset yang dirandomisasi.

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 melakukan randomisasi 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 dilakukan diharapkan memiliki hasil yang tidak memiliki perbedaan sama sekali dengan penambangan data klasifikasi atau *clustering* yang menggunakan dataset asli.

DAFTAR REFERENSI

- [1] 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.
- [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] MENDES, R. dan VILELA, J. P. (2017) Privacy-preserving data mining: Methods, metrics, and applications. *IEEE Access*, **5**, 10562–10582.
- [4] Han, J., Kamber, M., dan Pei, J. (2012) *Data Mining: Concepts and Techniques*, 3rd edition. Morgan Kaufmann, Waltham.
- [5] 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.
- [6] 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.
- [7] Agrawal, R. dan Srikant, R. (2000) Privacy preserving data mining. In *Proceedings of the ACM SIGMOD*, **3**, 439–450.
- [8] 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.
- [9] Johnson, W. B. dan Lindenstrauss, J. (1984) Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, **26**, 189–206.
- [10] 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.
- [11] 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.
- [12] Rossum, G. V. (1995) Python tutorial. Technical Report CS-R9526. Centrum voor Wiskunde en Informatica (CWI), Amsterdam.
- [13] 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, I., 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.

- [14] 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.
- [15] 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.
- [16] Hunter, J. D. (2007) Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, **9**, 90–95.