BAB 1

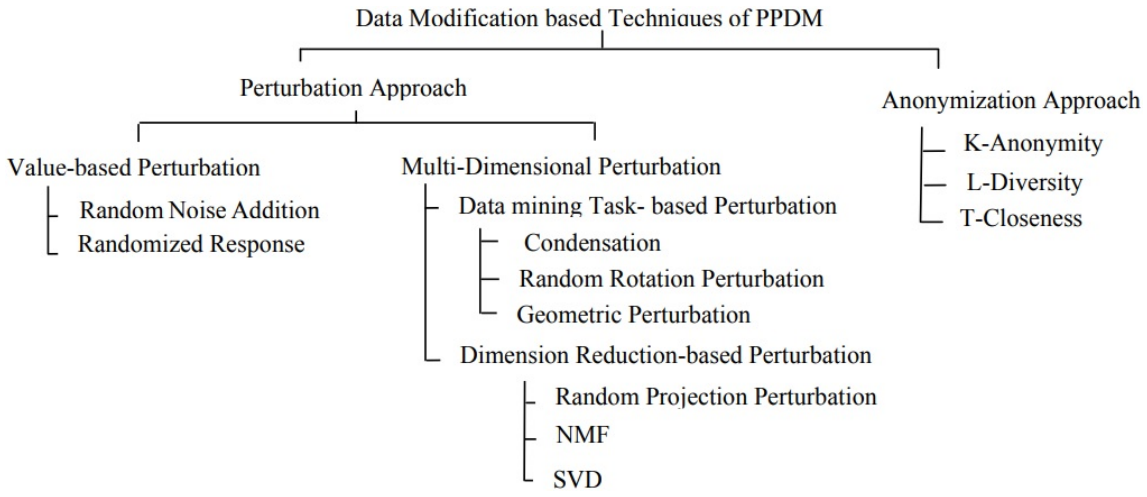
PENDAHULUAN

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 tersebar 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 ditambang. *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 ditambang menjadi data yang telah dimodifikasi dengan metode *Randomization* sehingga privasi pada data tersebut terlindungi, tetapi masih dapat ditambang. 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 konkrit 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

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 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 k yang menentukan seberapa banyak kluster yang diinginkan dan sebuah *treshold* 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 *treshold*

   (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

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 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 menambang data yang telah ditambahkan *noise* ini perlu dilakukan rekonstruksi 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 rekonstruksi yang digunakan berdasarkan pada aturan *Bayes*. Algoritma rekonstruksi 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

(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

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.

$$G(X) = X_{n\times d} R_{d\times d} \tag{2.1}$$

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 tranformasi 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 sehigga 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.

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* $f : \mathbb{R}^m \to \mathbb{R}^k$ *such that, for all* $x, y \in S$*,* $(1 - eps)||u - v||^2 < ||p(u) - p(v)||^2 < (1 + eps)||u - v||^2$*, where* $||.||$ *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, sedemikian rupa sehingga jarak antara dua titik tetap konsisten dengan *error* yang terkontrol tetapi dengan syarat d harus cukup besar. Oleh karena adanya *error* 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. [10]

$$G(X) = X_{n \times d} R_{d \times k} \tag{2.2}$$

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 akan menjadi dimensi matriks. Oleh karena yang ingin dilakukan adalah reduksi dimensi maka 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*". 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.

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \tag{2.3}$$

Sebuah matriks yang akan diproyeksikan ke dimensi yang lebih kecil akan memiliki nilai *error* pada jarak Euclidean yang dimiliki oleh titik-titik (setiap elemen dari matriks) pada bidang Euclidean tersebut. Nilai *error* 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).

$$(1 - eps)||u - v||^2 < ||p(u) - p(v)||^2 < (1 + eps)||u - v||^2 \tag{2.4}$$

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

BAB 3

ANALISIS

Privasi yang perlu dijaga antara lain mengenai identitas seseorang atau hal yang dapat dikaitkan terhadap identitas seseorang. Pada data yang digunakan untuk penambangan data sangat banyak sekali data privasi tersebut sehingga perlu adanya cara untuk menjaga privasi tersebut. Metode randomisasi dapat digunakan untuk menghilangkan privasi pada data tetapi masih dapat dilakukan penambangan data. Metode yang dipilih untuk diimplementasikan adalah teknik *Random Rotation Perturbation* dan *Random Projection Perturbation*. Tetapi ada kekurangan pada kedua teknik tersebut. kekurangan tersebut adalah nilai setiap fitur yang ada pada data harus bersifat numerik dan kedua teknik tersebut hanya menjaga jarak Euclidean sehingga hanya teknik penambangan data yang bergantung pada jarak Euclidean saja yang dapat digunakan.

3.1 *Random Rotation Perturbation*

Seperti yang telah dijelaskan di bab 2, ide dari teknik *Random Rotation Perturbation* adalah merotasi seluruh data yang direpresentasikan sebagai titik pada bidang Euclidean sehingga jarak antara titik-titik yang ada tidak berubah walaupun nilai tiap titik berubah secara drastis. Berikut akan ditunjukkan algoritma dan studi kasus yang telah dilakukan pada teknik *Random Rotation Perturbation*.

3.1.1 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
6. Buatlah *random rotation matrix* 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

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

8. Hasil matriks yang telah dirotasi sudah dapat langsung digunakan untuk penambangan data

3.1.2 Studi Kasus

Untuk lebih memahami bagaimana cara kerja teknik *Random Rotation Perturbation*, studi kasus dilakukan pada dataset *iris*, tetapi untuk memudahkan perhitungan pada studi kasus ini data yang dipakai hanya sebagian kecil saja dari seluruh data pada dataset *iris*. Data 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*

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

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

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 \\ 4.9 & 3 & 1.4 & 0.2 \\ 4.7 & 3.2 & 1.3 & 0.2 \\ 4.6 & 3.1 & 1.5 & 0.2 \\ 5 & 3.6 & 1.4 & 0.2 \\ 5.4 & 3.9 & 1.7 & 0.4 \\ 4.6 & 3.4 & 1.4 & 0.3 \\ 5 & 3.4 & 1.5 & 0.2 \\ 4.4 & 2.9 & 1.4 & 0.2 \end{bmatrix}_{9\times4}$$

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 71.35281261 & 93.96479736 & 77.16763568 & 27.88189356 & 1 \end{bmatrix}_{5\times5}$$

3. Untuk keperluan translasi, matriks dataset ditambahkan sebuah kolom dengan nilai 1 pada

setiap barisnya

$$\begin{bmatrix} 5.1 & 3.5 & 1.4 & 0.2 & 1 \\ 4.9 & 3 & 1.4 & 0.2 & 1 \\ 4.7 & 3.2 & 1.3 & 0.2 & 1 \\ 4.6 & 3.1 & 1.5 & 0.2 & 1 \\ 5 & 3.6 & 1.4 & 0.2 & 1 \\ 5.4 & 3.9 & 1.7 & 0.4 & 1 \\ 4.6 & 3.4 & 1.4 & 0.3 & 1 \\ 5 & 3.4 & 1.5 & 0.2 & 1 \\ 4.4 & 2.9 & 1.4 & 0.2 & 1 \end{bmatrix}_{9\times5}$$

4. 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

$$\begin{bmatrix} 76.45281261 & 97.46479736 & 78.56763568 & 28.08189356 & 1 \\ 76.25281261 & 96.96479736 & 78.56763568 & 28.08189356 & 1 \\ 76.05281261 & 97.16479736 & 78.46763568 & 28.08189356 & 1 \\ 75.95281261 & 97.06479736 & 78.66763568 & 28.08189356 & 1 \\ 76.35281261 & 97.56479736 & 78.56763568 & 28.08189356 & 1 \\ 76.75281261 & 97.86479736 & 78.86763568 & 28.28189356 & 1 \\ 75.95281261 & 97.36479736 & 78.56763568 & 28.18189356 & 1 \\ 76.35281261 & 97.36479736 & 78.66763568 & 28.08189356 & 1 \\ 75.75281261 & 96.86479736 & 78.56763568 & 28.08189356 & 1 \end{bmatrix}_{9\times5}$$

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

$$\begin{bmatrix} -0.45126938 & -0.70425922 & 0.32389616 & 0.44211556 \\ -0.43989334 & 0.70728617 & 0.39249528 & 0.39011226 \\ -0.17797534 & 0.06110969 & -0.83056872 & 0.52416218 \\ 0.75576092 & 0.00555185 & 0.22626167 & 0.61449187 \end{bmatrix}_{4\times4}$$

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

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

3.2 *Random Projection Perturbation*

Ide dari teknik *Random Projection Perturbation* seperti yang telah dijelaskan pada bab 2 adalah mereduksi dimensi dataset sehingga nilai pada setiap kolom akan berubah bahkan kolomnya akan berkurang yang mengakibatkan kolom-kolom yang ada tidak bisa diketahui kolom tersebut adalah kolom apa. Berikut akan ditunjukkan algoritma dan studi kasus yang telah dilakukan pada teknik *Random Projection Perturbation*.

3.2.1 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 ϵ (eps) yang diinginkan dan berada pada rentang (0, 1)
3. Hitung nilai k (dimensi minimal) dengan rumus berikut $k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n$
4. Tentukan nilai k yang diinginkan atau tentukan secara acak dengan syarat pada langkah ketiga terpenuhi dan k harus lebih kecil dari d
5. Buatlah matriks proyeksi dengan cara membuat matriks acak yang diambil mengikuti distribusi normal dengan rata-rata bernilai 0 dan standar deviasi bernilai $1/\sqrt{k}$ berdimensi $d \times k$
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

3.2.2 Studi Kasus

Untuk lebih memahami bagaimana cara kerja teknik *Random Projection Perturbation*, studi kasus dilakukan pada dataset *iris*. 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* tersebut tidak memenuhi persyaratan untuk mendapatkan hasil yang baik, tetapi untuk memudahkan perhitungan pada studi kasus ini data yang dipakai adalah dataset *iris* yang memiliki dimensi yang kecil dan hanya sebagian kecil saja data yang dipakai. Data tersebut dapat dilihat pada Tabel 3.1. Dalam menghitung nilai 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.

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

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

2. Ditentukan nilai ϵ (eps) yang diinginkan adalah 0.5

3. Nilai k (dimensi minimal) dihitung dengan rumus berikut

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

4. Nilai k dipilih sesuai keinginan, dalam kasus ini dipilih nilai k sebesar 332

5. Membuat matriks proyeksi dengan cara membuat matriks acak yang diambil mengikuti distribusi normal dengan rata-rata bernilai 0 dan standar deviasi bernilai $1/\sqrt{k}$ berdimensi $d \times k$. 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

$$\begin{bmatrix} 0.65684027 & -1.0035495 & 0.72820632 \\ 0.60194004 & -0.90822018 & 0.66416944 \\ 0.60217727 & -0.92172316 & 0.66620218 \\ 0.56344827 & -0.88887716 & 0.65931422 \\ 0.6517441 & -1.00838106 & 0.7317004 \\ 0.67922914 & -1.09633771 & 0.76805051 \\ 0.58987895 & -0.9446188 & 0.66701567 \\ 0.62854085 & -0.97454336 & 0.71636295 \\ 0.53813813 & -0.84238446 & 0.62076123 \end{bmatrix}_{9\times3}$$

3.3 Perangkat Lunak

Ada beberapa persyaratan yang harus dipenuhi pada implementasi perangkat lunak seperti alur, masukan, keluaran, dan antarmuka 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. Pada subbab ini akan ditunjukkan gambaran singkat perancangan perangkat lunak yang akan dibuat berupa diagram aktivitas dan diagram kelas.

3.3.1 Diagram Aktivitas

Perangkat lunak randomisasi adalah perangkat lunak yang digunakan untuk memodifikasi data dengan metode randomisasi. 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. Detail dari diagram aktivitas tersebut adalah sebagai berikut.

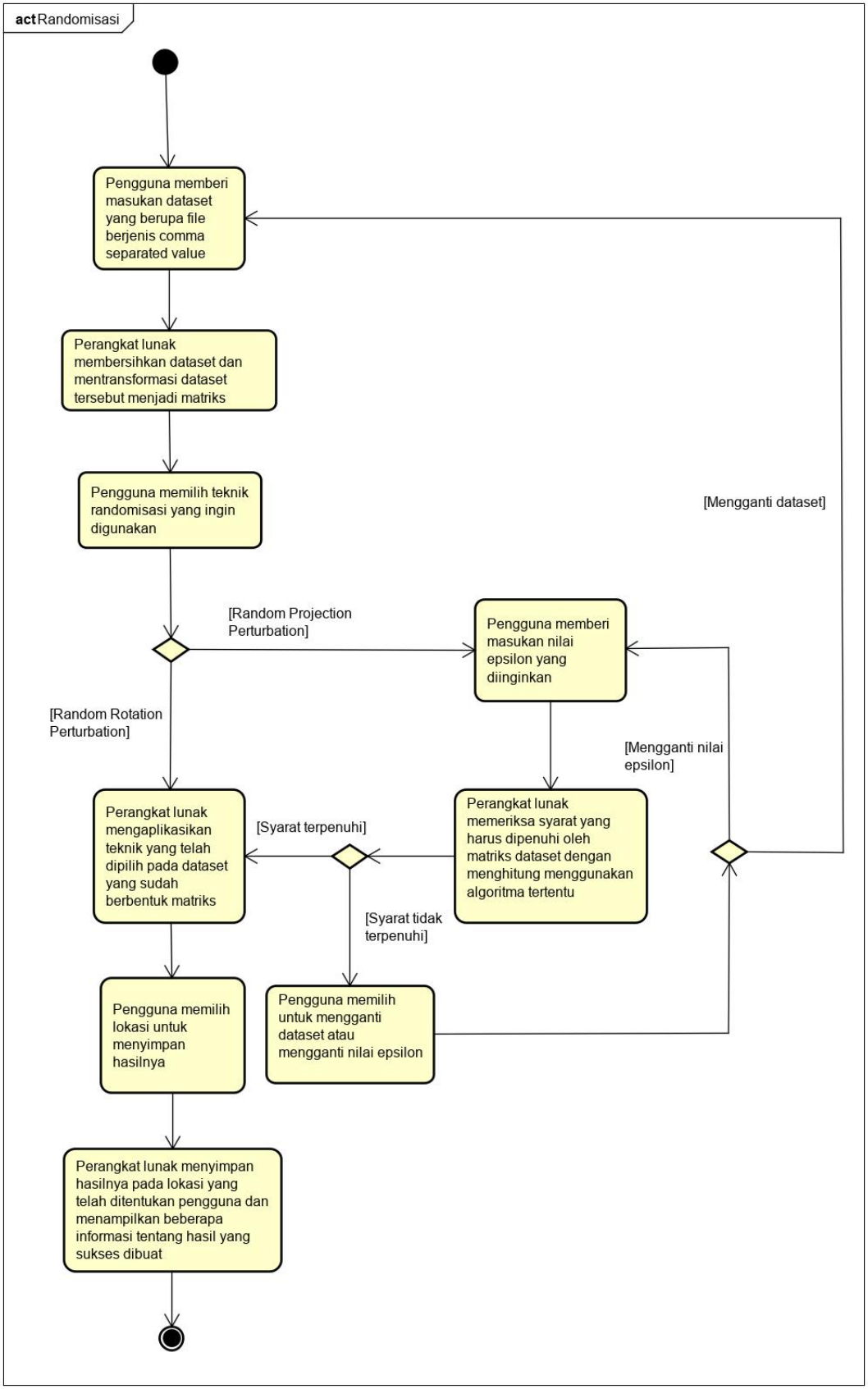
1. Pengguna memberikan masukan berupa dataset yang berupa dokumen berjenis *comma-separated values* (CSV). Dokumen ini harus berisi tiap rekord pada barisnya dan tiap fitur pada kolomnya. Adanya nama kolom diperbolehkan pada baris pertama dalam dokumen tersebut

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

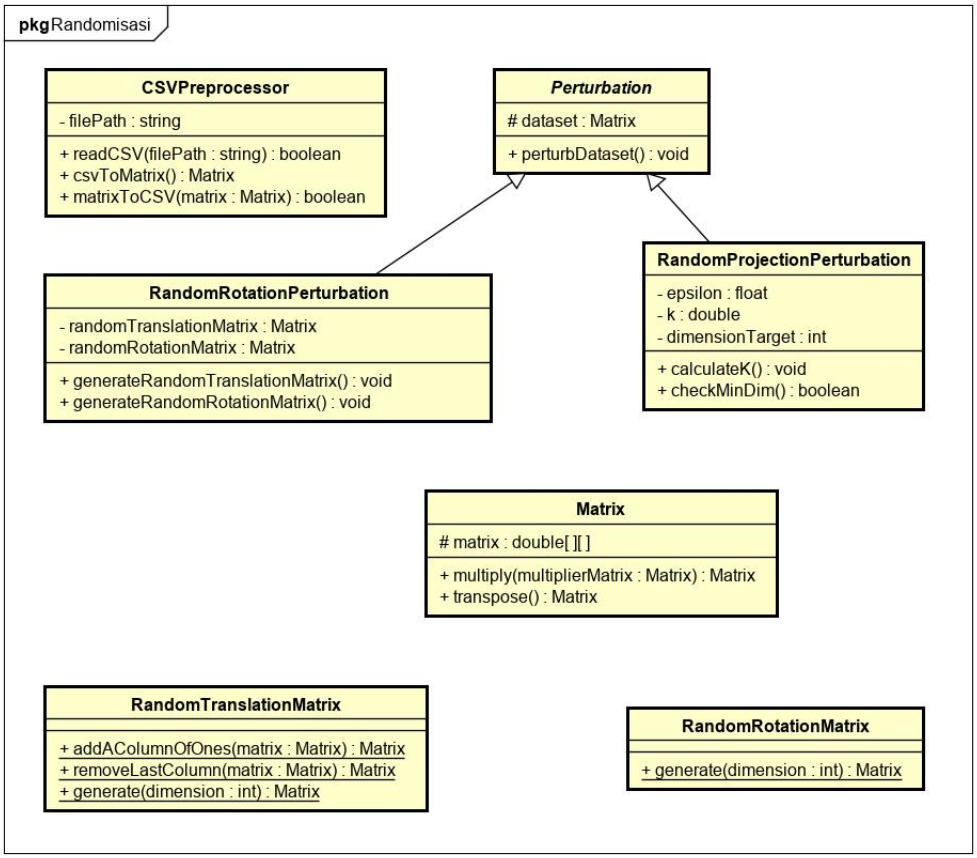
3. Pengguna memilih teknik randomisasi yang ingin digunakan antara *Random Rotation Perturbation* dan *Random Projection Perturbation*. Jika *Random Projection Perturbation* yang dipiilih maka akan ada beberapa langkah yang harus dipenuhi yaitu sebagai berikut.

   (a) Pengguna memberi masukan nilai epsilon yang diinginkan

   (b) Perangkat lunak memeriksa syarat yang harus dipenuhi oleh matriks dataset dengan menghitung menggunakan algoritma tertentu. Pengecekan ini adalah pengecekan jumlah kolom pada matriks dataset apakah cukup untuk matriks tersebut direduksi dimensinya.



Gambar 3.1: Diagram aktivitas perangkat lunak randomisasi



Gambar 3.2: Diagram kelas perangkat lunak randomisasi

Jika syarat terpenuhi maka langkah selanjutnya adalah perangkat lunak mengaplikasikan teknik yang dipilih

(c) Jika syarat tidak terpenuhi maka pengguna harus memilih untuk mengganti datasetnya atau mengganti nilai epsilon

4. Perangkat lunak mengaplikasikan teknik yang telah dipilih pada dataset yang sudah berbentuk matriks

5. Pengguna memilih lokasi pada komputer pengguna untuk menyimpan hasil dari teknik yang dipilih. Hasilnya adalah sebuah dokumen *comma-separated values* yang berisi dataset yang sudah diaplikasikan teknik randomisasi

6. Perangkat lunak menyimpan hasilnya pada lokasi yang telah ditentukan pengguna 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 *RandomRotationPerturbation* adalah kelas yang mengimplementasikan algoritma *Random Rotation Perturbation*. Kelas ini merupakan kelas turunan dari kelas *Perturbation*
- Kelas *RandomProjectionPerturbation* adalah kelas yang mengimplementasikan algoritma *Random Projection Perturbation*. Kelas ini merupakan kelas turunan dari kelas *Perturbation*
- Kelas *Matrix* adalah kelas untuk merepresentasikan matriks dan menyimpan nilai-nilai pada setiap elemen matriks. Kelas ini juga memiliki fungsi perkalian dan transpose yang akan digunakan untuk implementasi algoritma *Random Rotation Perturbation* dan *Random Projection Perturbation*
- 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.
- 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.
- 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.

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. Perangkat lunak randomisasi akan memiliki 3 halaman antarmuka yang diimplementasikan dengan desain antarmuka tabs. Rancangan antarmuka halaman utama dapat dilihat pada Gambar 4.1, halaman ini akan berisi pesan selamat datang berupa deskripsi singkat perangkat lunak beserta cara kerjanya dan petunjuk singkat cara penggunaan perangkat lunak. 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.

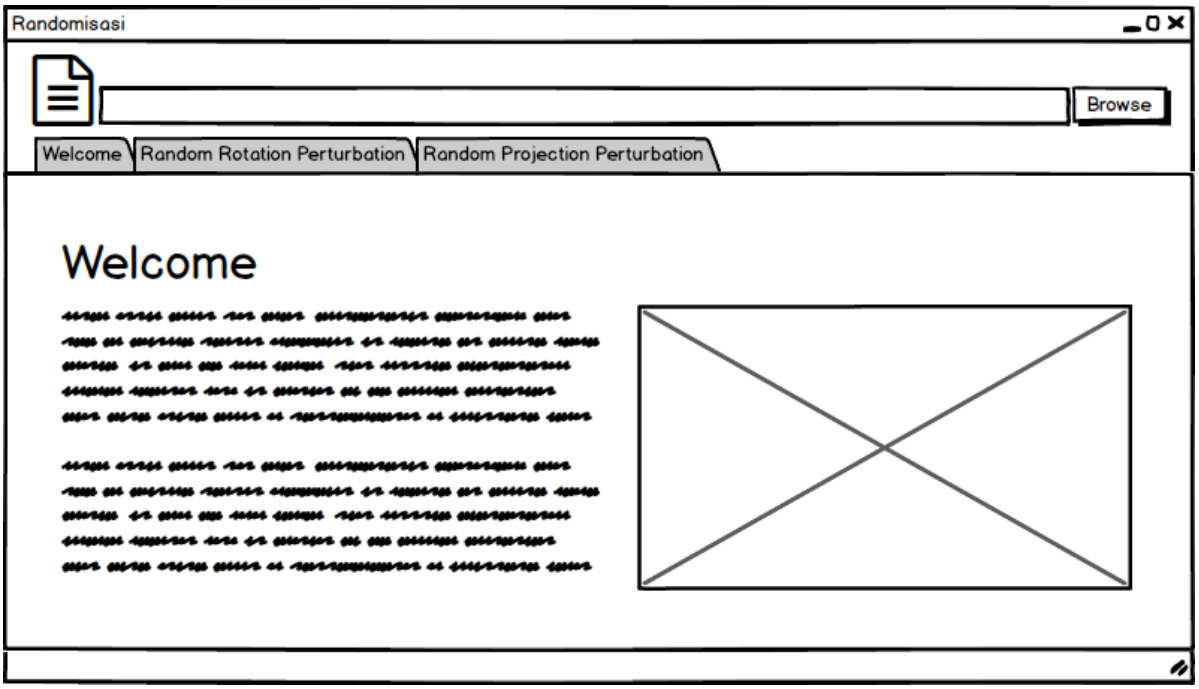
4.1.1 Halaman *Random Rotation Perturbation*

Halaman ini memiliki fungsi untuk melakukan teknik *Random Rotation Perturbation*. Setelah user memasukkan dokumen yang diinginkan pada kolom yang berada pada bagian atas antarmuka, pengguna baru bisa mengakses halaman ini. Pada halaman ini terdapat berbagai informasi dataset yang dimasukkan dan hasil dari randomisasi jika berhasil dilakukan, serta pengaturan apa saja fitur yang akan dirandomisasi pada dataset yang pengguna masukkan. Rancangan dari halaman ini dapat dilihat pada Gambar 4.2.

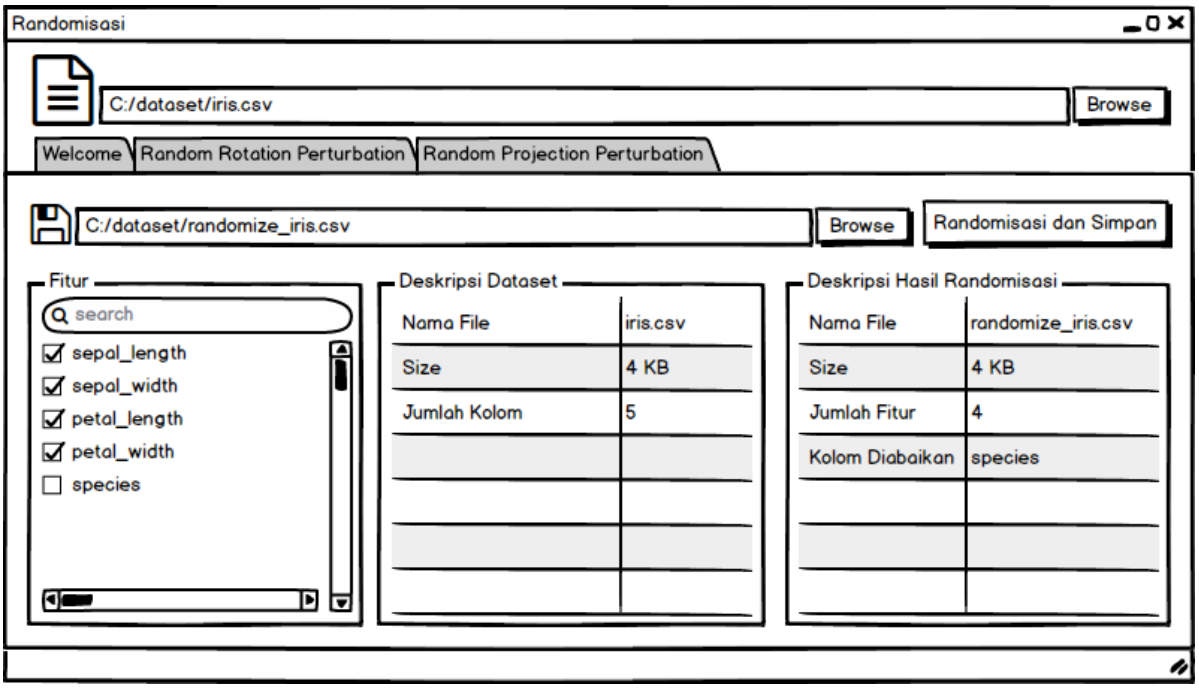
Sebelum melakukan randomisasi, pengguna perlu memeriksa fitur mana saja yang ingin dirandomisasi karena bisa saja pada dataset tersebut ada kolom yang berupa label. Perangkat lunak akan menyimpan hasil randomisasi pada dokumen dataset pengguna yang telah diubah nilainya dan perangkat lunak akan menyimpan dokumen tersebut di suatu lokasi yang pengguna harus tentukan. Pengguna bisa menentukan lokasi dokumen hasil randomisasi pada kolom yang terdapat di sebelah kanan ikon simpan dan sebelah kiri tombol "Randomisasi dan Simpan".

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 dengan teknik *Random Rotation Perturbation* pada dataset yang dimasukkan 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.

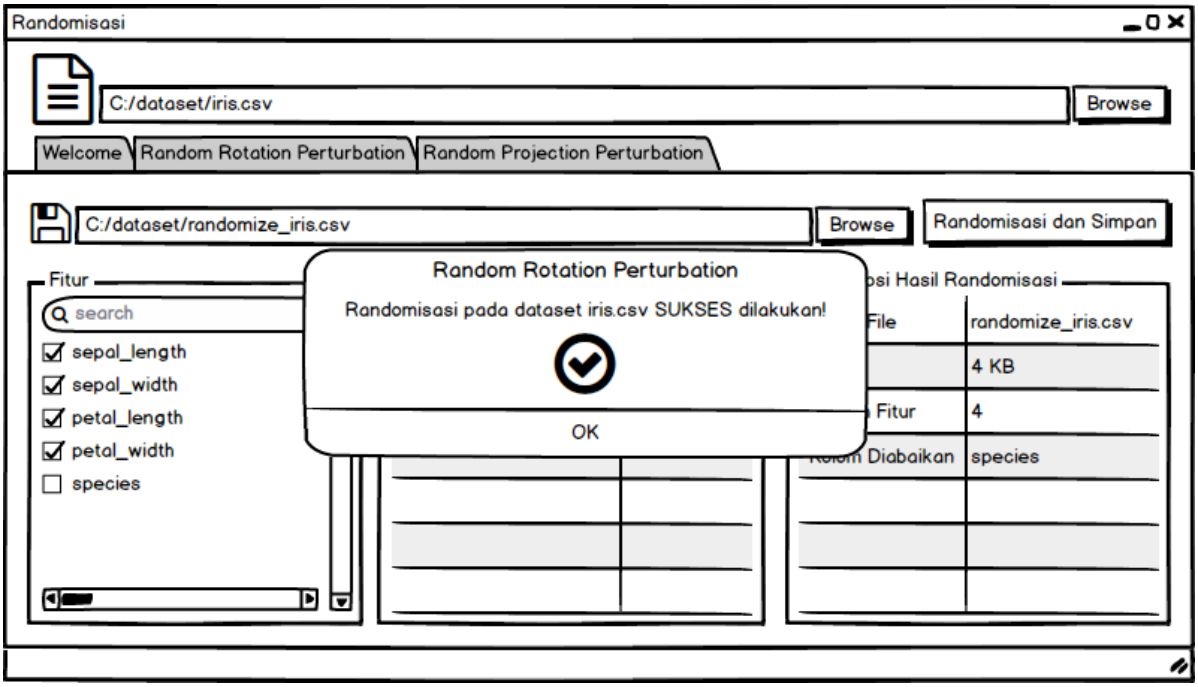
Perangkat lunak akan menampilkan deskripsi hasil dari randomisasi yang dilakukan. Informasi



Gambar 4.1: Halaman utama yang berisi petunjuk cara penggunaan perangkat lunak



Gambar 4.2: Halaman untuk melakukan teknik *Random Rotation Perturbation*



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

yang ditampilkan oleh perangkat lunak antara lain nama dokumen, ukuran dokumen, jumlah fitur, 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.

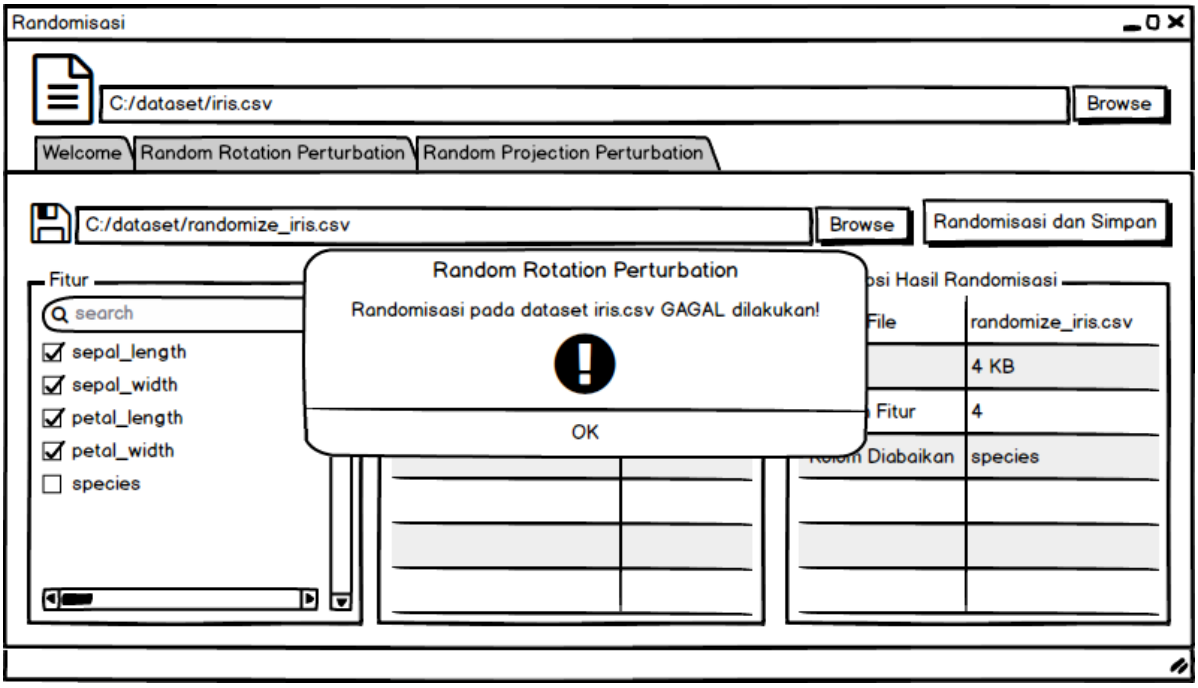
Apabila ada masalah-masalah tertentu dan randomisasi gagal dilakukan, maka perangkat lunak akan memberitahukan bahwa randomisasi telah gagal dilakukan dengan menampilkan *popup* yang berisi peringatan bahwa randomisasi gagal dilakukan. Hasil randomisasi tidak akan terbuat dan perangkat lunak tidak akan menampilkan informasi apapun tentang hasil randomisasi. *Popup* tersebut dapat dilihat pada Gambar 4.4.

4.1.2 Halaman *Random Projection Perturbation*

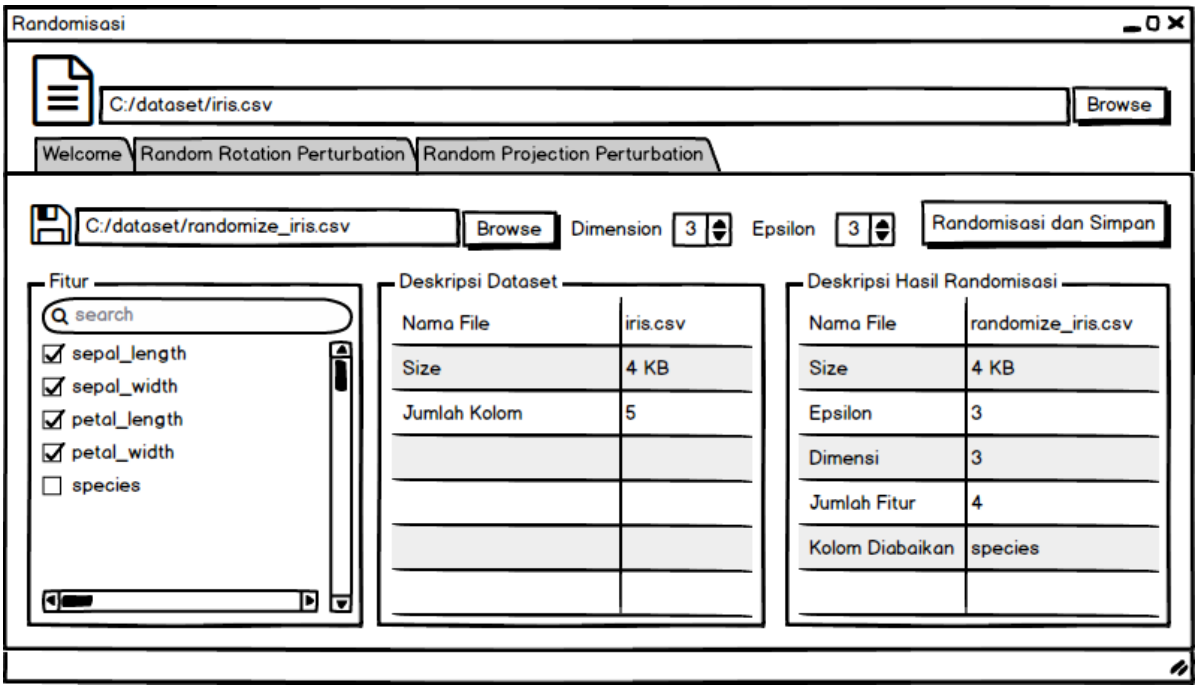
Halaman ini memiliki fungsi untuk melakukan teknik *Random Projection Perturbation*. Rancangan dari halaman ini dapat dilihat pada Gambar 4.5. Sebelum pengguna melakukan randomisasi, pengguna harus mengatur beberapa pengaturan yang ada. Sama seperti halnya pada halaman *Random Rotation Perturbation*, pengguna perlu menentukan lokasi penyimpanan hasil randomisasi yang akan dilakukan dan kolom apa saja pada dataset yang menjadi fitur. Pada bagian pemilihan fitur pada dataset, perangkat lunak mempunyai fitur kolom pencarian untuk mencari fitur pada dataset secara mudah dengan mengetikkan nama fitur yang diinginkan.

Selain itu, pada halaman ini yang khusus untuk teknik *Random Projection Perturbation* pengguna harus menentukan dimensi yang diinginkan dan nilai epsilon. Setelah pengaturan-pengaturan tersebut diatur, pengguna dapat melakukan randomisasi menggunakan teknik *Random Projection Perturbation* dengan menekan tombol "Randomisasi dan Simpan", perangkat lunak akan melakukan randomisasi dan menyimpan hasil randomisasi di lokasi yang telah pengguna tentukan.

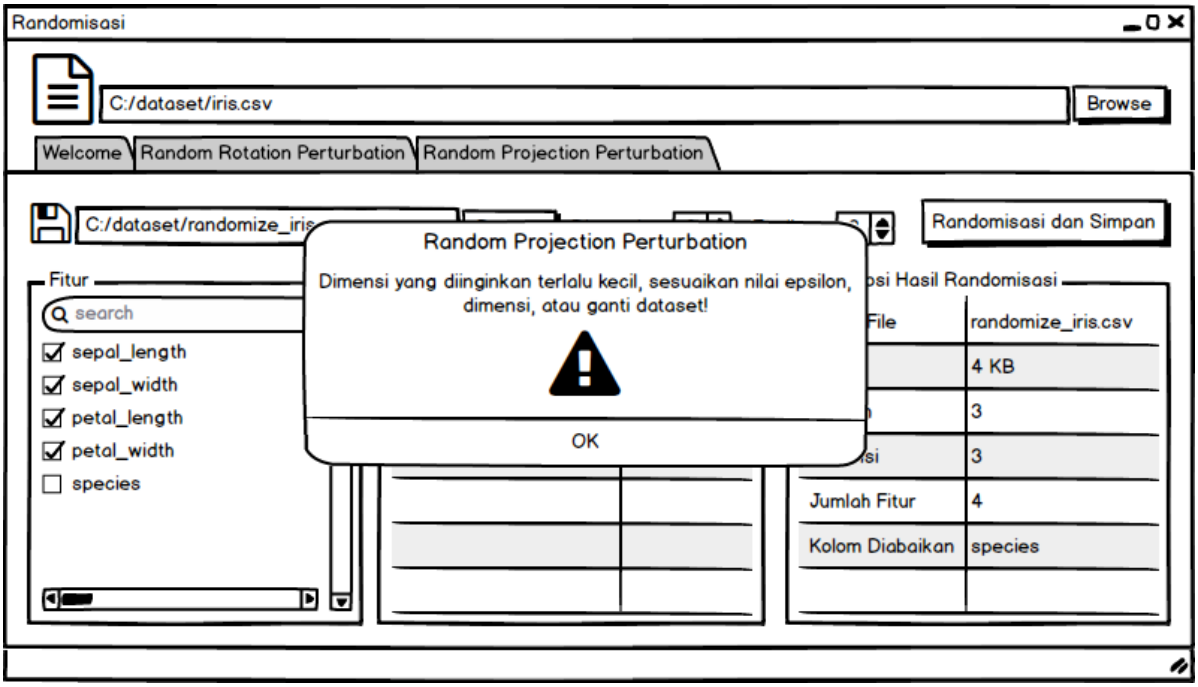
Pada teknik *Random Projection Perturbation*, ada persyaratan yang harus dipenuhi mengenai dimensi akhir yang diinginkan dan nilai epsilon. Kedua nilai tersebut perlu sesuai dengan dataset yang ada sehingga pengguna tidak bisa sembarangan menentukan dimensi dan nilai epsilon. Perangkat lunak akan membuat aturan mengenai minimal dimensi yang bisa digunakan sehingga pengguna tidak bisa memasukan dimensi yang lebih kecil dari minimal yang telah ditentukan. Minimal dimensi ini bergantung pada banyaknya baris pada dataset dan nilai epsilon seperti yang



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



Gambar 4.5: Halaman untuk melakukan teknik *Random Projection Perturbation*



Gambar 4.6: *Popup* yang akan ditampilkan apabila syarat pada dataset tidak terpenuhi

telah dijelaskan pada bagian analisis bab 3.

Apabila randomisasi berhasil dilakukan, maka perangkat lunak akan menampilkan beberapa deskripsi tentang hasil randomisasi yang berhasil dilakukan seperti nama dokumen hasil randomisasi, ukuran dokumen, nilai epsilon yang dipakai, jumlah dimensi pada hasil randomisasi, dan kolom-kolom apa saja yang diabaikan. Perangkat lunak juga akan menyimpan hasil randomisasi yang telah dilakukan dalam bentuk dokumen berjenis *comma-separated values* di lokasi yang telah ditentukan sebelumnya oleh pengguna.

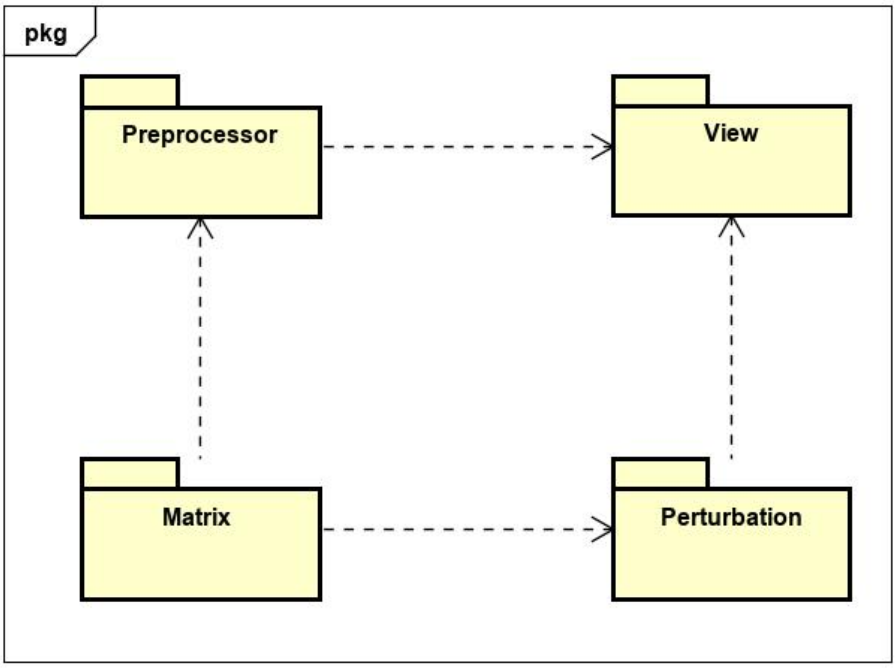
Oleh karena adanya persyaratan yang harus dipenuhi oleh pengguna untuk melakukan randomisasi dengan teknik *Random Projection Perturbation*, maka randomisasi bisa saja gagal dilakukan karena persyaratan yang ada tidak dipenuhi oleh pengguna. Persyaratan yang disebutkan adalah persyaratan jumlah dimensi 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 4.6.

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 4.7



Gambar 4.7: 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.

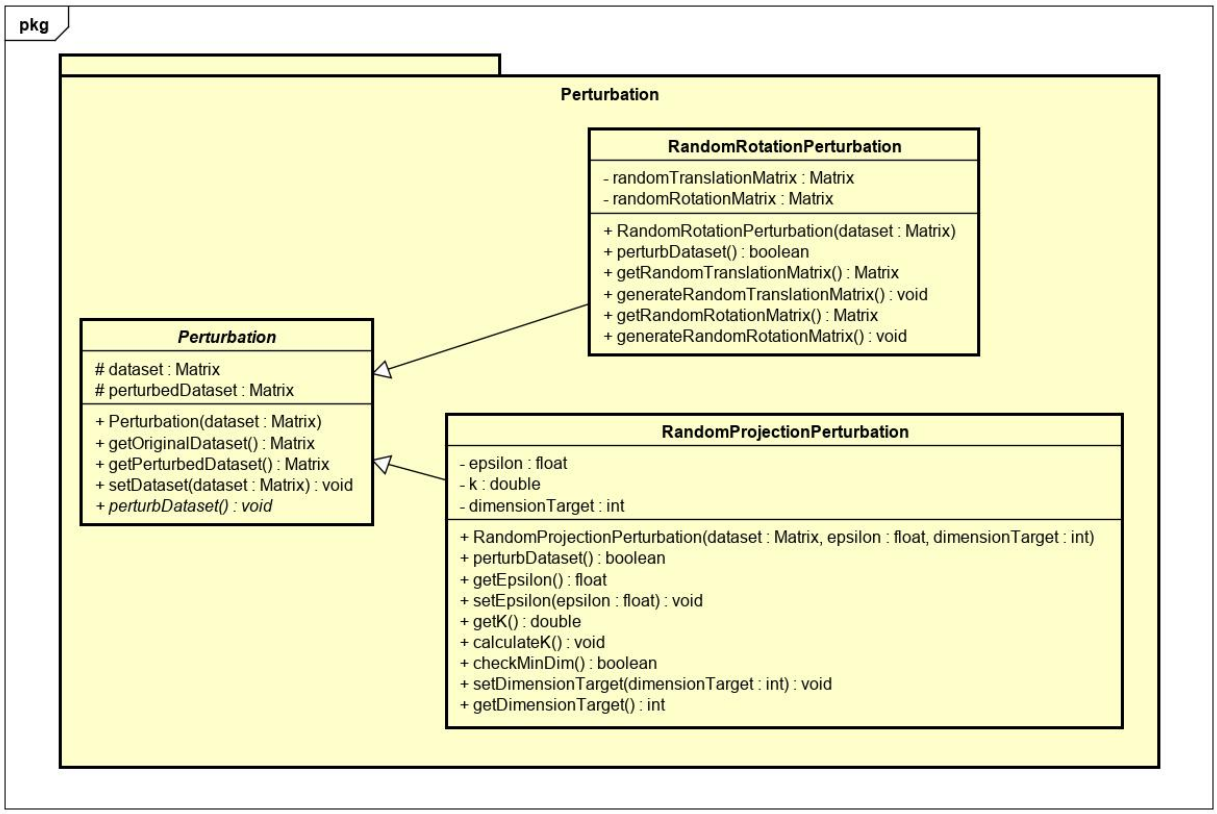
Dalam penerapan algoritma teknik yang dipakai, *package* ini membutuhkan komponen atau fungsi lain untuk membuat algoritma yang diimplementasikan bekerja. Fungsi yang dibutuhkan antara lain adalah membuat matriks dengan sifat tertentu. Oleh karena itu *package Perturbation* membutuhkan *package Matrix* 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 perangkat lunak. Tentunya masukan yang diberikan pengguna kepada perangkat lunak tidak bisa langsung diolah begitu saja. Perlu adanya pengolahan terlebih dahulu sebelum masukan yang



Gambar 4.8: Diagram kelas pada *package Perturbation*

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 syarat 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 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 pemograman Python yang bernama Kivy[1]. *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*

[1]https://kivy.org/#home

yang akan dijelaskan secara detail pada subbab ini. Diagram kelas pada *package Perturbation* dapat dilihat pada Gambar 4.8.

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*. Fungsi *generateRandomTranslationMatrix* akan mendefinisikan atribut ini dengan membuat matriks translasi acak, yang akan dijelaskan kemudian. Atribut ini mempunyai tipe data *Matrix*.

- *randomRotationMatrix* adalah atribut untuk menampung matriks rotasi yang akan dipergunakan untuk implementasi algoritma *Random Rotation Perturbation*. Fungsi *generateRandomRotationMatrix* akan mendefinisikan atribut ini dengan membuat matriks translasi acak, yang akan dijelaskan kemudian. 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*.

- *generateRandomTranslationMatrix* adalah fungsi untuk membuat matriks translasi secara acak yang akan digunakan untuk implementasi algoritma teknik *Random Rotation Perturbation*. Fungsi ini akan membuat matriks translasi acak tersebut dan menyimpan hasilnya pada atribut *randomTranslationMatrix*. Tidak ada masukan ataupun kembalian pada fungsi ini.

- *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*.

- *generateRandomRotationMatrix* adalah fungsi untuk membuat matriks rotasi secara acak yang akan digunakan untuk implementasi algoritma teknik *Random Rotation Perturbation*. Fungsi ini akan membuat matriks rotasi acak tersebut dan menyimpan hasilnya pada atribut *randomRotationMatrix*. Tidak ada masukan ataupun kembalian pada fungsi ini.

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 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 *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 *k* 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.

- *k* 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 syarat untuk direduksi yaitu memiliki jumlah kolom yang lebih besar dari nilai *k*. Tipe data atribut ini adalah *double*, bilangan riil. Atribut ini akan dihitung oleh perangkat lunak sesuai nilai *epsilon* yang telah ditentukan pengguna.

- *dimensionTarget* 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 *k*, 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*.

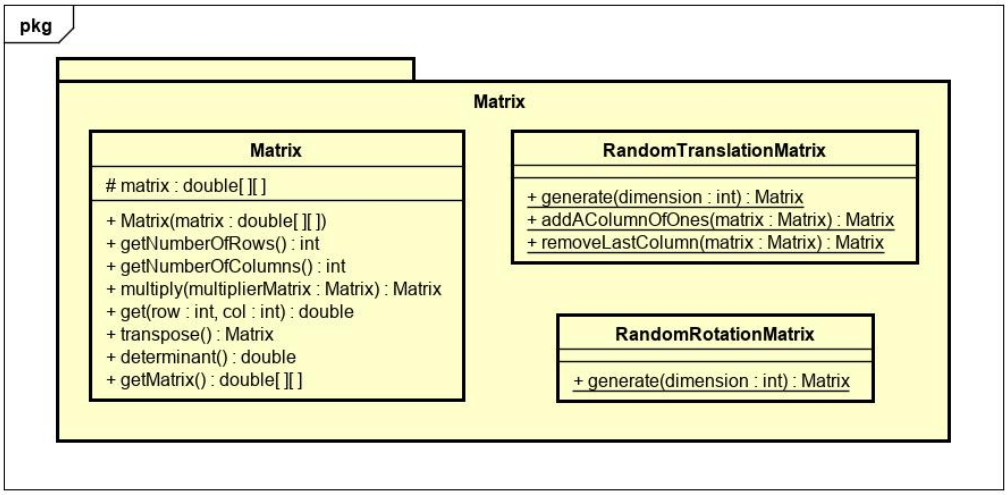
- *RandomProjectionPerturbation* adalah *constructor* untuk mendefinisikan atribut-atribut yang ada dan memanggil *constructor* dari kelas *super* milik kelas *RandomProjectionPerturbation* yaitu *Perturbation*. Tujuan utama fungsi ini adalah memanggil *constructor* dari kelas *super* yaitu *Perturbation*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

- *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.

- *setEpsilon* adalah fungsi untuk mengubah nilai dari atribut *epsilon* dengan nilai baru yang menjadi masukan fungsi ini. Fungsi *setEpsilon* memiliki sebuah masukan *epsilon* yang bertipe data *float*, bilangan riil dan berfungsi untuk mendefinisikan atribut *epsilon*. *Parameter* ini akan menjadi nilai baru dari atribut *epsilon*. Tidak ada kembalian pada fungsi ini.

- *getK* adalah fungsi untuk mendapatkan besar dimensi minimal hasil dataset yang telah dirandomisasi atau dengan kata lain mendapatkan atribut *k*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa nilai dari atribut *k* yang bertipe data *double*, bilangan riil.



Gambar 4.9: Diagram kelas pada *package Matrix*

- *calculateK* adalah fungsi untuk menghitung nilai k sesuai dengan atribut *epsilon* dan jumlah baris pada dataset yang ingin dirandomisasi. Fungsi ini akan menghitung nilai k tersebut dan menyimpan hasilnya pada atribut k. Tidak ada masukan ataupun kembalian pada fungsi ini.
- *checkMinDim* adalah fungsi untuk memeriksa apakah dimensi dari *dataset* yang ingin dirandomisasi lebih besar dari nilai k, besar dimensi minimal. Selain itu, fungsi ini memeriksa apakah nilai *dimensionTarget* lebih besar dari atau sama dengan nilai k. Intinya fungsi ini menguji apakah dataset dan keinginan pengguna memenuhi syarat untuk mengaplikasikan teknik *Random Projection Perturbation*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa sebuah *boolean* yang menandakan apakah *dataset* dan *dimensionTarget* yang diberikan oleh pengguna memenuhi syarat untuk mengaplikasikan teknik *Random Projection Perturbation* dengan *dataset* dan *dimensionTarget* tersebut.
- *setDimensionTarget* adalah fungsi untuk mengubah nilai dari atribut *dimensionTarget* dengan nilai baru yang menjadi masukan fungsi ini. Fungsi *setDimensionTarget* memiliki sebuah masukan *dimensionTarget* yang bertipe data *integer*, bilangan bulat dan berfungsi untuk mendefinisikan atribut *dimensionTarget*. *Parameter* ini akan menjadi nilai baru dari atribut *dimensionTarget*. Tidak ada kembalian pada fungsi ini.
- *getDimensionTarget* adalah fungsi untuk mendapatkan besar dimensi akhir yang diinginkan pengguna dimiliki oleh hasil dataset yang telah dirandomisasi atau dengan kata lain mendapatkan atribut *dimensionTarget*. Fungsi ini tidak memiliki masukan apapun tetapi mempunyai kembalian berupa nilai dari atribut *dimensionTarget* yang bertipe data *integer*, bilangan bulat.

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 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 4.9.

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 *PerturbMatrixation*. 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.
- *getMatrix* 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.

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*.

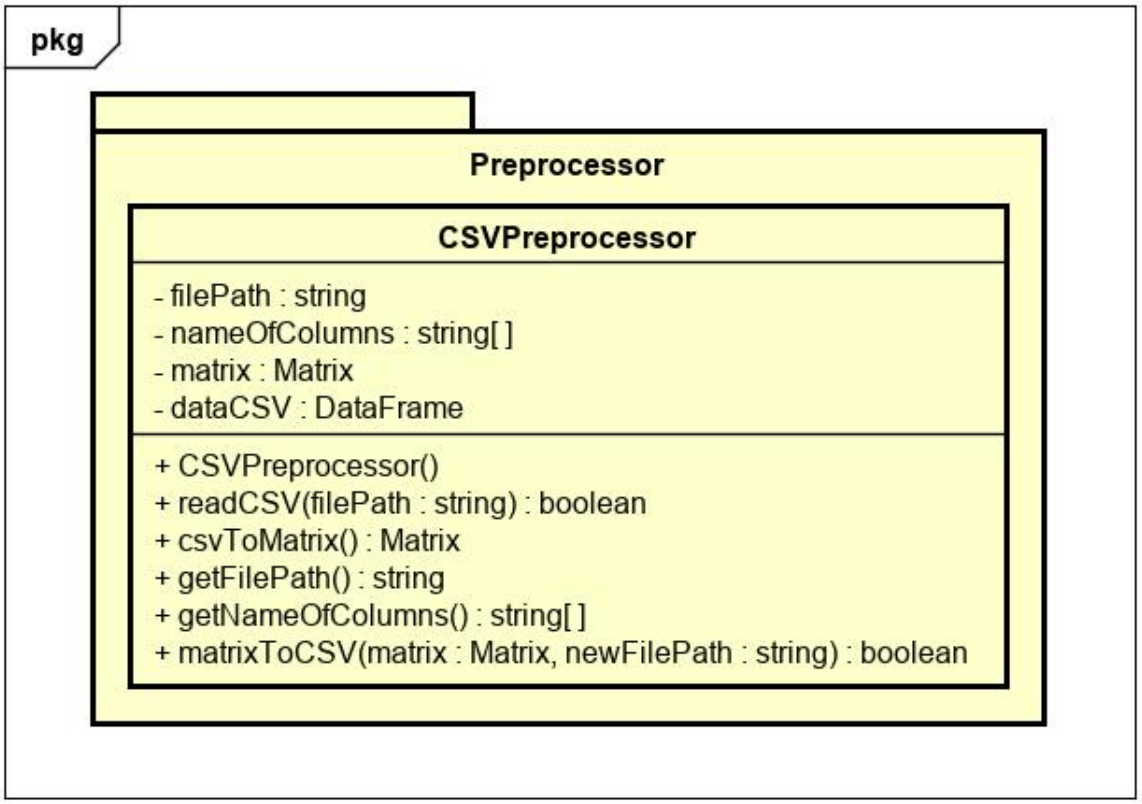
- *generate* adalah fungsi statis yang berguna untuk membuat matriks rotasi. Pembuatan matriks rotasi dilakukan dengan mengikuti distribusi Haar [8]. Fungsi ini memiliki sebuah parameter yaitu *dimension* yang akan menentukan ukuran atau dimensi matriks rotasi yang ingin dibuat. Kembalian pada fungsi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

4.2.4 Diagram Kelas pada *Package Preprocessor*

Package Preprocessor memiliki sebuah kelas yang bertujuan untuk mengolah masukan perangkat lunak agar dapat diterapkan teknik randomisasi. Kelas tersebut bernama *CSVPreprocessor* yang akan dijelaskan secara detail pada subbab ini. Diagram kelas pada *package Preprocessor* dapat dilihat pada Gambar 4.10.

Kelas *CSVPreprocessor*

Kelas CSVPreprocessor berguna untuk mengolah masukan perangkat lunak yang berjenis *comma-separated values* sebelum diterapkan teknik randomisasi agar masukan tersebut sesuai dengan



Gambar 4.10: Diagram kelas pada *package Preprocessor*

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*[2]. 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

[2]https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

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.

4.3 Masukan Perangkat Lunak

Perangkat lunak akan mempunyai sebuah masukan yang berupa dataset. Dataset yang ingin dirandomisasi memiliki syarat 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 syarat 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 juga akan diterapkan pada perangkat lunak secara fungsional dan eksperimental. Hasil dari pegujian akan dijelaskan secara rinci dan sistemasi serta akan dibuat kesimpulan untuk pengujian yang telah dilakukan.

5.1 Implementasi Antarmuka

Antarmuka perangkat lunak diimplementasikan dengan memakai *framework* antarmuka grafis berbasis bahasa pemograman Python yang bernama Kivy[1]. Implementasi antarmuka disesuaikan dengan rancangan antarmuka perangkat lunak yang telah dibuat pada bab 4. Berikut ini adalah tampilan antarmuka dari implementasi perangkat lunak.

Antarmuka perangkat lunak mempunyai tiga buah bagian yang mempunyai fungsinya masing-masing. 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.

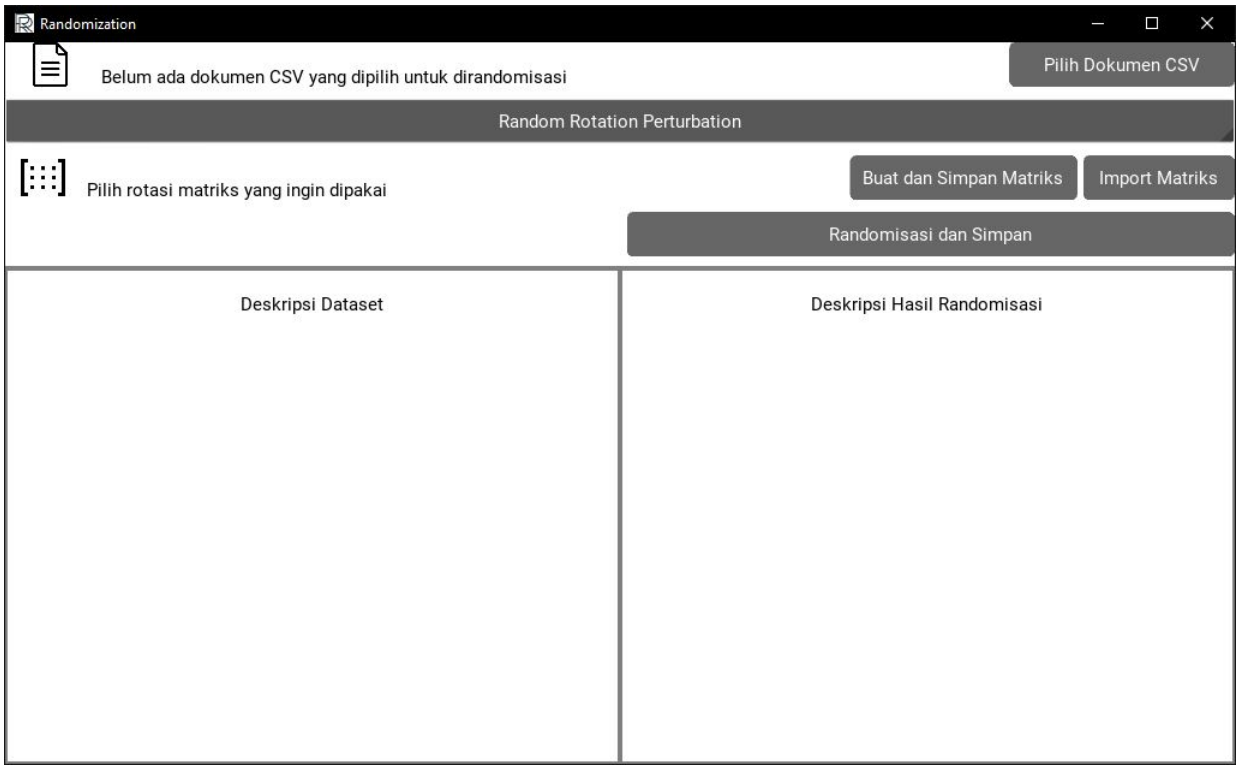
5.1.1 Masukan dan Pengaturan

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 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*, dan sebuah tombol untuk menerapkan teknik randomisasi dan menyimpan hasilnya. Berikut akan dijelaskan secara rinci dengan gambar setiap fungsi tersebut dan cara pemakaiannya yang benar secara berturut.

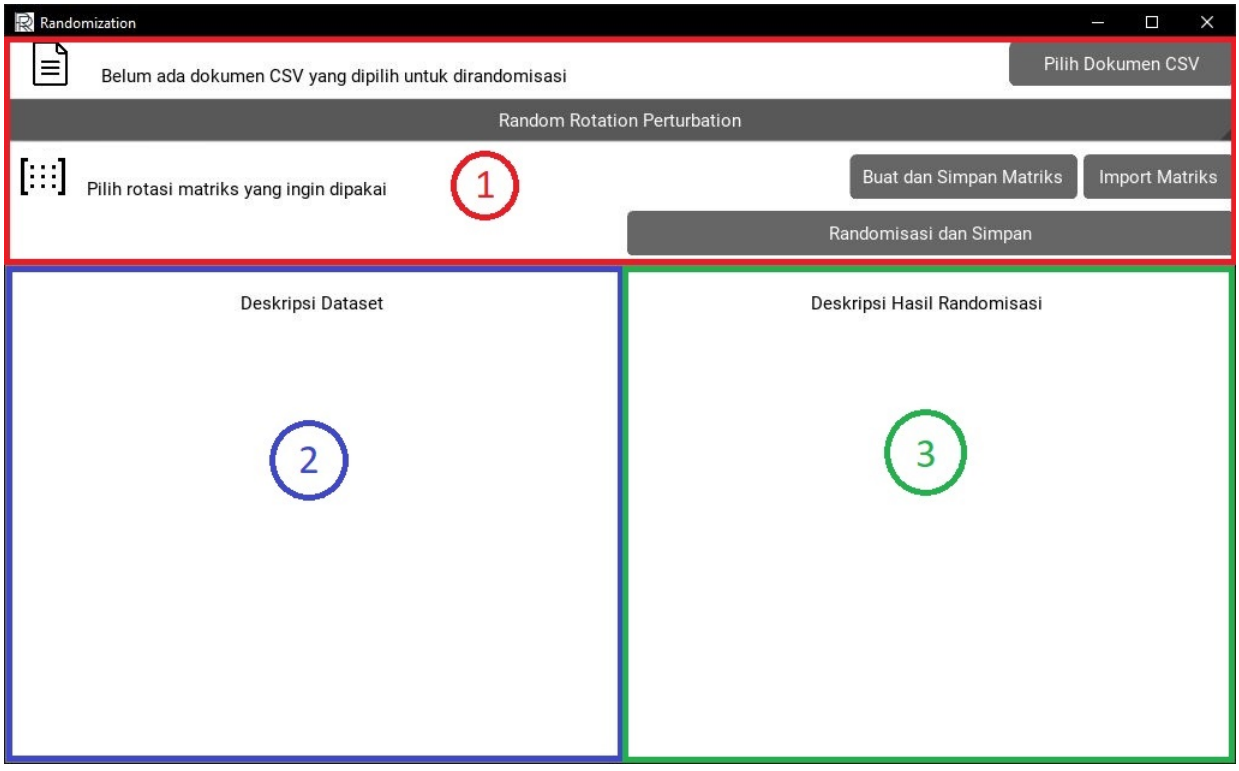
Berikut akan dijelaskan setiap fungsi pada bagian masukan dan pengaturan sekaligus menjelakan cara pemakaian bagian antarmuka perangkat lunak ini. Angka pada gambar akan sesuai dengan nomor urutan berikut.

1. abc

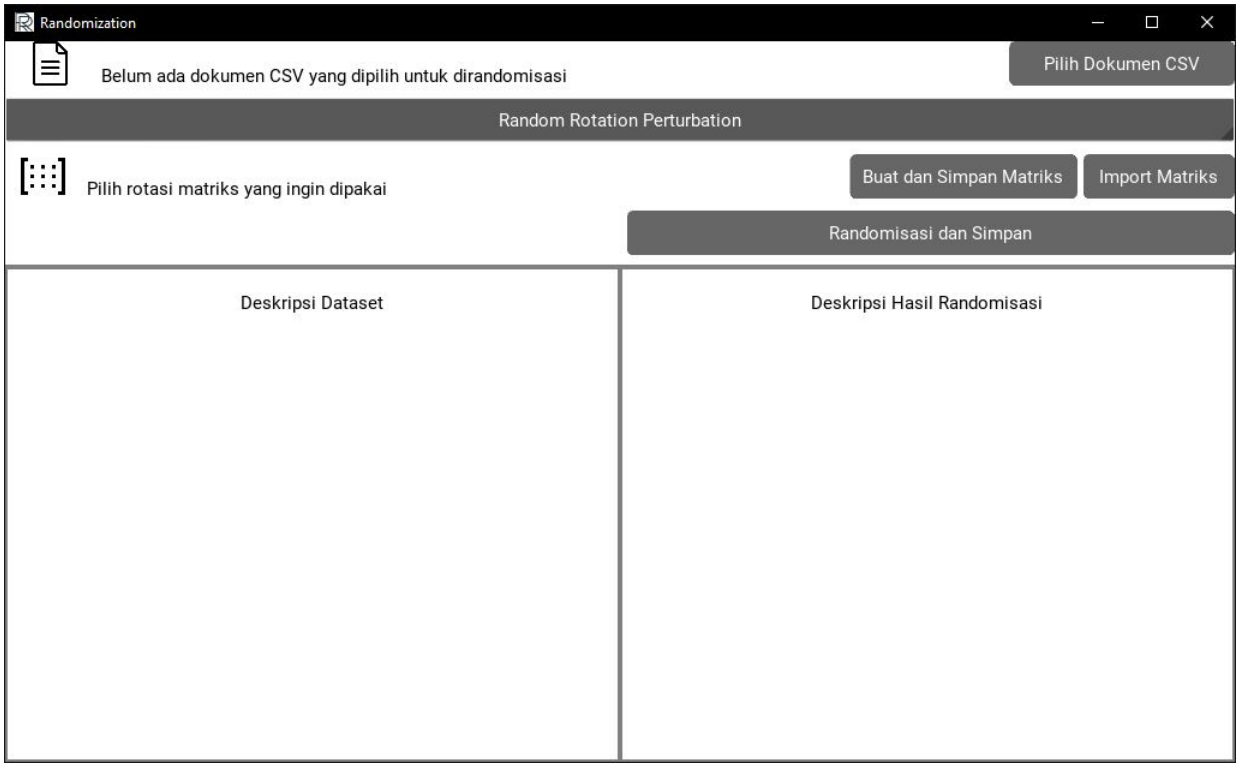
[1] https://kivy.org/#home



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-bagian pada antarmuka perangkat lunak

5.1.2 Deskripsi Dataset

5.1.3 Deskripsi Hasil Randomisasi

5.2 Pengujian Perangkat Lunak

5.2.1 Pengujian Fungsional

5.2.2 Pengujian Eksperimental

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.