**KLASIFIKASI SERANGAN *DISTRIBUTED DENIAL OF SERVICE* (DDoS) DENGAN ALGORITMA *K-NEAREST NEIGHBOR* (KNN)**

**SKRIPSI**

****

**NI KADEK SUKMA PUTRI RAHAYU**

**1708561038**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**JIMBARAN**

**2021**

# **SURAT PERNYATAAN KEASLIAN KARYA ILMIAH**

# **LEMBAR PENGESAHAN**

# **ABSTRAK**

# **KATA PENGANTAR**

Proposal penelitian dengan judul Klasifikasi Serangan *Distributed Denial Of Service* (DDoS) dengan Algoritma *K-Nearest Neighbor* (KNN)ini disusun dalam rangkaian pelaksanaan Tugas Akhir di Program Studi Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Udayana. Proposal ini disusun dengan harapan dapat menjadi pedoman dan arahan dalam melaksanakan penelitian selanjutnya.

Sehubungan dengan telah selesainya pengerjaan dari proposal ini, maka diucapkan terimakasih dan penghargaan kepada berbagai pihak yang telah membantu untuk pengusulan proposal ini, antara lain:

1. Bapak I Komang Ari Mogi, S.Kom., M.Kom. sebagai Pembimbing I yang telah banyak membantu menyempurnakan proposal ini;
2. Bapak Cokorda Rai Adi Pramartha,ST.MM.PhD sebagai Pembimbing I yang telah banyak membantu menyempurnakan proposal ini;
3. Bapak-bapak dan Ibu-ibu dosen pengajar di Program Studi Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Udayana yang telah meluangkan waktu untuk memberikan saran dan masukan dalam menyempurnakan proposal ini;
4. Kawan-kawan di Program Studi Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Udaayana yang telah memberikan dukungan moral dalam penyelesasian proposal ini.

Disadari pula bahwa sudah tentu proposal ini masih memiliki kelemahan dan kekurangan didalamnya. Memperhatikan hal ini, maka dari itu masukan dan saran-saran penyempurnaan untuk proposal ini sangat diharapkan.

Jimbaran, Maret 2021

Penyusun

# **DAFTAR ISI**

[**SURAT PERNYATAAN KEASLIAN KARYA ILMIAH** 2](#_Toc65849260)

[**LEMBAR PENGESAHAN** 3](#_Toc65849261)

[**ABSTRAK** 4](#_Toc65849262)

[**KATA PENGANTAR** 5](#_Toc65849263)

[**DAFTAR ISI** 6](#_Toc65849264)

[**DAFTAR TABEL** 7](#_Toc65849265)

[**DAFTAR GAMBAR** 8](#_Toc65849266)

[**DAFTAR LAMPIRAN** 9](#_Toc65849267)

[**BAB I** 10](#_Toc65849268)

[**PENDAHULUAN** 10](#_Toc65849269)

[**1.1** **Latar Belakang** 10](#_Toc65849270)

[**1.2** **Rumusan Masalah** 12](#_Toc65849271)

[**1.3** **Batasan Masalah** 12](#_Toc65849272)

[**1.4** **Tujuan Penelitian** 13](#_Toc65849273)

[**1.5** **Manfaat Penelitian** 13](#_Toc65849274)

[**1.6** **Metodelogi Penelitian** 13](#_Toc65849275)

[**1.6.1** **Analisis Sistem** 14](#_Toc65849276)

[**1.6.2** **Desain Penelitian** 14](#_Toc65849277)

[**1.6.3** **Data Penelitian** 15](#_Toc65849278)

[**1.6.4** **Metode Penelitian** 15](#_Toc65849279)

[**1.6.5** **Metodologi Pengembangan Perangkat Lunak** 16](#_Toc65849280)

[**1.6.6** **Evaluasi dan Validasi Hasil** 18](#_Toc65849281)

[**BAB II** 19](#_Toc65849282)

[**TINJAUAN PUSTAKAN** 19](#_Toc65849283)

[**2.1.** **Anomali Jaringan** 19](#_Toc65849284)

[**2.2.** **Serangan DoS** 20](#_Toc65849285)

[**2.3.** **Serangan Distributed Denial of Services** 21](#_Toc65849286)

[**2.3.1** **Konsep Serangan DDoS** 21](#_Toc65849287)

[**2.3.2** **Metode Serangan DDoS** 22](#_Toc65849288)

[**2.4.** **Data Mining** 22](#_Toc65849289)

[**2.5.** **Proses Data Mining** 24](#_Toc65849290)

[**2.6.** **Seleksi Atribut** 25](#_Toc65849291)

[**2.7.** **Klasifikasi** 25](#_Toc65849292)

[**2.8.** ***Information Gain*** 26](#_Toc65849293)

[**2.9.** ***K-Nearest Neighbor*** 27](#_Toc65849294)

[**2.10.** ***K-fold Cross Validation*** 28](#_Toc65849295)

[**BAB III** 30](#_Toc65849296)

[**ANALISIS DAN PERANCANGAN** 30](#_Toc65849297)

[**3.1.** **Variable Penelitian** 30](#_Toc65849298)

[**3.2.** **Analisis Kebutuhan** 36](#_Toc65849299)

[**3.3.** **Analisis Sistem** 36](#_Toc65849300)

[**3.3.1** **Penentuan Input – input Sistem** 37](#_Toc65849301)

[**3.3.2** **Penentuan Proses-proses dalam Operasi Sistem** 38](#_Toc65849302)

[**3.4.** **Perancangan Sistem** 38](#_Toc65849303)

[**3.4.1.** **Flowchart Sistem** 40](#_Toc65849304)

[**3.4.2.** **Flowchart Information Gain** 41](#_Toc65849305)

[**3.4.3.** **Flowchart K-Nearest Neighbor** 43](#_Toc65849306)

[**3.5.** **Skenario Pengujian Sistem** 44](#_Toc65849307)

[**BAB IV** 46](#_Toc65849308)

[**HASIL DAN PEMBAHASAN** 46](#_Toc65849309)

[**4.1.** **Gambaran Umum Sistem** 46](#_Toc65849310)

[**4.2.** **Lingkungan Perancangan dan Implementasi Sistem** 46](#_Toc65849311)

[**4.3.** **Implementasi Program** 47](#_Toc65849312)

[**4.3.1.** **Proses Pemilihan Fitur dengan Information Gain** 47](#_Toc65849313)

[**4.3.2.** **Preprocessing Dataset** 48](#_Toc65849314)

[**4.3.3.** **Proses Perhitungan Jarak Eucledian** 50](#_Toc65849315)

[**4.3.4.** **Proses Perhitungan Klasifikasi dengan Metode *K-Nearest Neighbor*** 51](#_Toc65849316)

[**4.3.5.** **Proses Pengujian dengan *K-Fold Cross Validation*** 52](#_Toc65849317)

[**4.4.** **Pengujian Sistem** 55](#_Toc65849318)

[**4.4.1.** **Seleksi Fitur dengan *Information Gain*** 55](#_Toc65849319)

[**4.4.2.** **Normalisasi Dataset** 58](#_Toc65849320)

[**4.4.3.** **Klasifikasi Serangan DDoS dengan *K-Nearest Neighbor*** 60](#_Toc65849321)

[**4.4.4.** **Pengujian Sistem dengan K-Fold Cross Validation** 60](#_Toc65849322)

[**4.4.5.** **Pengukuran akurasi** 63](#_Toc65849323)

[**4.5.** **Analisis** 64](#_Toc65849324)

[**BAB V** 66](#_Toc65849325)

[**PENUTUP** 66](#_Toc65849326)

[**5.1** **Kesimpulan** 66](#_Toc65849327)

[**5.2** **Saran** 66](#_Toc65849328)

[**DAFTAR PUSTAKA** 68](#_Toc65849329)

[**LAMPIRAN** 71](#_Toc65849330)

# **DAFTAR TABEL**

[**Tabel 3. 1 Jumlah Paket Data** 32](#_Toc65849584)

[**Tabel 3. 2 Fitur Dasar (Basic) Tiap-tiap Koneksi TCP** 33](#_Toc65849585)

[**Tabel 3. 3 Fitur Konten pada Koneksi (Conection Based)** 34](#_Toc65849586)

[**Tabel 3. 4 Fitur Trafik Waktu (Time Based Features)** 35](#_Toc65849587)

[**Tabel 3. 5 Fitur Dataset** 36](#_Toc65849588)

[**Tabel 3. 6 Nama Serangan setiap Tipe Protokol** 40](#_Toc65849589)

[**Tabel 4. 1 Pemisahan Fitur dan Label Dataset** 49](#_Toc65849598)

[**Tabel 4. 2 Pembagian Data Latih dan Data Uji** 49](#_Toc65849599)

[**Tabel 4. 3 Ekstraksi Fitur** 50](#_Toc65849600)

[**Tabel 4. 4 Menghitung Nilai Entropy Tiap Fitur** 50](#_Toc65849601)

[**Tabel 4. 5 Mengurutkan Nilai Entropy Tiap Fitur** 50](#_Toc65849602)

[**Tabel 4. 6 Transformasi Dataset** 50](#_Toc65849603)

[**Tabel 4. 7 Mengubah Nilai Label** 51](#_Toc65849604)

[**Tabel 4. 8 Normalisasi Dataset** 52](#_Toc65849605)

[**Tabel 4. 9 Perhitungan Jarak Eucledian** 52](#_Toc65849606)

[**Tabel 4. 10 Fungsi K-Nearest Neighbors** 53](#_Toc65849607)

[**Tabel 4. 11 Pengujian dengan K-Fold Cross Validation** 54](#_Toc65849608)

[**Tabel 4. 12 Perhitungan Akurasi** 55](#_Toc65849609)

[**Tabel 4. 13 Main Program** 55](#_Toc65849610)

[**Tabel 4. 14 Entropy Setiap Fitur Dataset** 58](#_Toc65849611)

# **DAFTAR GAMBAR**

[**Gambar 1. 1 Tahapan Metode Waterfall** 19](#_Toc65849798)

[**Gambar 2. 1 Kerangka Proses Data Mining** 28](#_Toc65850137)

[**Gambar 2. 2 5-Fold Cross Validation** 31](#_Toc65850138)

[**Gambar 3. 1 Perancangan Sistem** 42](#_Toc65849808)

[**Gambar 3. 2 Flowchart Sistem** 43](#_Toc65849809)

[**Gambar 3. 3 Flowchart Information Gain** 44](#_Toc65849810)

[**Gambar 3. 4 Flowchart K-Nearest Neighbor** 46](#_Toc65849811)

[**Gambar 3. 5 Akurasi k = 1 Pada 21 Fitur** 66](#_Toc65849812)

[**Gambar 3. 6 Akurasi k = 2 Pada 21 Fitur** 66](#_Toc65849813)

[**Gambar 3. 7 Akurasi k = 3 Pada 21 Fitur** 66](#_Toc65849814)

[**Gambar 3. 8 Akurasi k = 4 Pada 21 Fitur** 66](#_Toc65849815)

[**Gambar 3. 9 Akurasi k = 5 Pada 21 Fitur** 67](#_Toc65849816)

[**Gambar 3. 10 Akurasi k = 1 Pada 41 Fitur** 67](#_Toc65849817)

[**Gambar 3. 11 Akurasi k = 2 Pada 41 Fitur** 67](#_Toc65849818)

[**Gambar 3. 12 Akurasi k = 3 Pada 41 Fitur** 67](#_Toc65849819)

[**Gambar 3. 13 Akurasi k = 4 Pada 41 Fitur** 67](#_Toc65849820)

[**Gambar 4. 1 Grafik Information Gain** 59](#_Toc65849821)

[**Gambar 4. 2 Normalisasi Dataset** 62](#_Toc65849822)

[**Gambar 4. 3 Pengujian 21 Fitur dengan k = 1** 64](#_Toc65849823)

[**Gambar 4. 4 Pengujian 41 Fitur dengan k = 1** 65](#_Toc65849824)

# **DAFTAR LAMPIRAN**

# **BAB I**

# **PENDAHULUAN**

## **Latar Belakang**

Pada era milenial ini teknologi berkembang dengan sangat pesat, bahkan kehidupan manusiapun sangat bergantung dengan teknologi yang ada. Teknologi digunakan dalam berbagai bidang mulai dari bidang kesehatan, perbankan, sosial, dan lain-lain. Namun semakin berkembangnya teknologi sekarang ini, maka semakin banyak juga resiko yang akan dihadapi. Salah satu resikonya adalah serangan yang dikirimkan oleh oknum yang tidak bertanggungjawab. Serangan yang dikirimkan terkadang susah dideteksi sehingga membuat tingkat keamanan dari jaringan komputer sangatlah rentan. Banyak jenis serangan yang dapat dikirimkan, tentu saja serangan ini akan membuat komputer menjadi lambat. Jenis serangan yang dikirimkan oleh onum yang tidak bertanggung jawab tersebut tentunya dapat merugikan pihak lain. Untuk meminimalisir resiko dari serangan, maka dibutuhkan sebuah sistem yang dapat digunakan untuk membaca pola yang menyimpang dari jaringan komputer. Sistem tersebut diharapkan dapat mendeteksi pola yang menyimpang dari sebuah jaringan komputer dengan hasil akurasi yang tinggi.

Salah satu kelas serangan yang digunakan oleh penyerang adalah *Distributed Denial of Service* (DDoS). Serangan *Distributed Denial of Service* (DDoS) merupakan serangan yang dikirimkan dari beberapa penyerang terhadap sebuah komputer atau server dalam jumlah yang melebihi kemampuan komputer itu sendiri (Satmoko et al., 2018). DDoS merupakan suatu ancaman yang serius karena tujuan utama dari serangan DDoS yaitu untuk membuat sumber daya jaringan target tidak dapat digunakan (Prakash & Priyadarshini, 2018). Serang DDoS dapat berasal dari software yang dapat di download secara gratis di Internet kemudian software tersebut memberi script kiddie (sebuah program yang digunakan untuk membantu menyerang sebuah server) untuk melakukan serangan DDoS. Serangan DDoS diluncurkan dengan cara menemukan server yang ingin diserang, kemudian dengan bantuan komputer-komputer zombie (botnet) atau komputer yang telah menginstall script kiddie dari komputer utama (penyerang), secara bersamaan akan membantu menyerang server tujuan (korban) dengan memerintahkan botnet untuk meluncurkan serangan (Hoque et al., 2015). Tentu saja hal ini dapat mengganggu lalu lintas pada jaringan sehingga server akan sulit untuk diakses. Untuk memudahkan dalam mendeteksi anomali data dalam jaringan maka diperlukan sebuah sistem yang dapat mengklasifikasikan data anomali dalam aktivitas data normal dalam jaringan, sehingga dapat dilakukan pencegahan terhadap serangan DDoS dan mengatasi permasalahan yang dapat ditimbulkan oleh serangan DDoS.

Penelitian tentang deteksi serangan DDoS pada jaringan sebelumnya sudah pernah kembangkan oleh (Ramadhan et al., 2019) dengan membandingkan dua algoritma yaitu algoritma K-Nearest Neighbor dan Decision Tree, dalam penelitian tersebut mengunakan dataset CICIDS2017. Hasil dari penelitian tersebut didapatkan bahwa algoritma KNN memiliki akurasi sebesar 98,94% dan hal tersebut menunjukan hasil dari algoritma KNN tidak lebih bagus dibandingkan dengan menggunakan algoritma Decision Tree, karena proses perhitungannya algoritma Decision Tree melibatkan nilai entropi dan information gain sehingga lebih detail daripada KNN yang perhitungannya sederhana.

Dalam penelitian yang dilakukan penulis dalam mengklasifikasikan serangan DDoS, peneliti menggunakan algoritma K-Nearest Neighbor (KNN) dengan penambahan seleksi atribut atau yang sering dikenal juga dengan istilah seleksi fitur menggunakan Information Gain untuk mendapatkan hasil yang lebih akurat. Berdasarkan penelitian yang dilakukan oleh (Kurniabudi et al., 2020) yang berjudul Seleksi Fitur dengan Information Gain untuk Meningkatkan Deteksi Serangan DDoS Menggunakan Random Forest dihasilkan bahwa teknik seleksi fitur dengan information gain mampu meningkatkan performa metode klasifikasi pada algoritma Random Forest.

Dalam penelitian yang dilakukan, dataset yang akan digunakan adalah dataset NSL-KDD tentang jenis anomali jaringan. Sebelum masuk ke tahap pengujian dataset, terlebih dahulu dilakukan seleksi fitur terhadap dataset sehingga mendapatkan fitur yang paling relevan dalam hasil data uji yang akan digunakan nantinya. Seleksi fitur ini dilakukan menggunakan metode Information Gain. Setelah mendapatkan fitur-fitur yang akan digunakan dalam pengujian terhadap data uji, maka data uji tersebut diklasifikasikan menggunakan metode *K-Nearest Neighbor* sehingga dapat mengetahui pola serangan DDoS dan dapat mencegah serangan DDoS lebih awal.

## **Rumusan Masalah**

Berdasarkan latar belakang yang disampaikan maka permasalahan dalam pembuatan penelitian ini adalah :

1. Bagaimana cara untuk menseleksi fitur-fitur dari dataset yang digunakan dengan metode *Information Gain*.
2. Bagaimana cara mengklasifikasikan data yang normal dan data serangan *Distributed Denial of Service* (DDoS) menggunakan algoritma *K-Nearest Neighbor* (KNN).
3. Bagaimana akurasi yang dihasilkan oleh program pada pengklasifikasian serangan *Distributed Denial of Service* (DDoS).

## **Batasan Masalah**

Adapun beberapa batasan masalah yang akan digunakan sebagai acuan dalam penelitian ini adalah sebagai berikut :

1. Penelitian ini menggunakan data riset yang bersumber dari dataset NSL-KDD.
2. Fitur-fitur dari dataset dipilih sebanyak 21 fitur, dimana fitur tersebut dipilih menggunakan metode *Information Gain*.
3. Penelitian ini hanya mengklasifikasikan data yang masuk termasuk kedalam data serangan *Distributed Denial of Service* (DDoS) atau data normal.
4. Pengujian program dilakukan dengan menggunakan metode *K-Fold Cross Validation*.

## **Tujuan Penelitian**

Berdasarkan rumusan masalah tersebut maka adapun tujuan dalam penelitian ini adalah :

1. Mengetahui fitur-fitur yang paling relevan dan berpengaruh terhadap jenis data yang masuk.
2. Mendeteksi data yang masuk apakah data tersebut adalah data yang bersifat serangan DDoS atau data normal.
3. Mengetahui akurasi klasifikasi serangan DDoS dengan menggunakan algoritma *K-Nearest Neighbor* dan seleksi fitur *Information Gain.*

## **Manfaat Penelitian**

Beberapa manfaat yang dapat diambil dalam penelitian ini adalah sebagai berikut :

1. Mendapatkan informasi lebih awal terhadap data yang masuk yang berpotensi serangan DDoS.
2. Mendapatkan hasil yang akurat terhadap pendeteksian serangan DDoS.

## **Metodelogi Penelitian**

Metodelogi penelitian ini adalah langkah yang dilakukan oleh peneliti dalam mengumpulkan informasi atau data serta melakukan investigasi pada data yang diperoleh. Pada bagian ini, peneliti menjelaskan penyusunan tugas akhir secara sistematis. Adapun bahasan yang akan dijelaskan dalam penelitian ini yaitu desain penelitian, data penelitian, dan metode yang digunakan dalam penelitian ini.

### **Analisis Sistem**

Pada tahap ini, dilakukan identifikasi perkiraan kebutuhan yang diperlukan untuk penelitian ini berupa materi yang berhubungan dengan penelitian yang dilakukan. Hal yang diperlukan yaitu menyangkut pada penerapan metode Information Gain dalam melakukan seleksi terhadap fitur-fitur yang digunakan dalam dataset dan juga penerapan metode K-Nearest Neighbor dalam melalukan klasifikasi dalam sebuah dataset.

### **Desain Penelitian**

Penelitian ini mengambil judul “Klasifikasi Serangan *Distributed Denial Of Service* (DDoS) dengan Algoritma *K-Nearest Neighbor* (KNN)”, penelitian bertujuan untuk mengetahui jenis data yang masuk ke suatu perangkat apakah data tersebut adalah data yang dikirim sebagai data anomali atau data serangan atau data tersebut adalah data normal atau data yang memang benar diinginkan oleh user. Desain penelitian yang digunakan dalam penelitian ini adalah studi kasus *(Case Studi Research)* dan eksperimental. Studi kasus merupakan penelitian yang memusatkan perhatian pada suatu kasus tertentu dengan menggunakan individu atau kelompok sebagai bahan studi. Penggunaan penelitian studi kasus ini biasanya difokuskan untuk menggali dan mengumpulkan data yang lebih dalam terhadap objek yang diteliti untuk dapat menjawab permasalahan yang sedang terjadi. Sehingga bisa dikatakan penelitian bersifat deskriptif dan eksploratif. Desain penelitian ini dipilih karena penelitian ini difokuskan pada sebuah kasus tertentu, dimana penelitian ini menggunakan dataset NSL-KDD yang didapat dari situs web yaitu kanggle.com. Aplikasi pendeteksi anomali jaringan akibat serangan DDoS *(Distributed Denial of Service)* menggunakan algoritma KNN *(K-Nearest Neighbor)* untuk klasifikasi serangan ini dibangun berdasarkan tingkat kedekatan satu data dengan data lainnya. Sehingga, isi dari klasifikasi yang terbentuk meliputi data-data yang memiliki kedekatan dan kemiripan yang serupa.

Data yang digunakan dalam penelitian ini adalah data sekunder. Data sekunder adalah data yang diperoleh melalui media perantara. Dalam proses pengenalan data ini, tahap yang dilakukan terlebih dahulu adalah memilih fitur-fitur yang paling baik digunakan. Dalam tahap pemilihan fitur ini, metode yang akan digunakan yaitu Information Gain. Setelah mendapatkan fitur-fitur tersebut selanjutkan akan dilakukan proses pelatihan terhadap data, kemudian data yang sudah dilatih digunakan sebagai pembanding terhadap data uji yang dimasukan dalam program.

### **Data Penelitian**

Tahap selanjutnya adalah tahap pengumpulan data yang akan digunakan dalam penelitian ini. Dataset yang digunakan dalam penelitian ini adalah dataset yang berasal dari situs web kanggle.com yaitu NSL-KDD. Data NSL-KDD dipilih karena data ini sudah banyak digunakan oleh para peneliti sebelumnya. Data yang digunakan dalam penelitian ini berjumlah 40000 data dimana data tersebut terdiri dari 2 jenis data yaitu data normal dan serangan yaitu *back, land, ping of death* (POD), dan *smurf.*

### **Metode Penelitian**

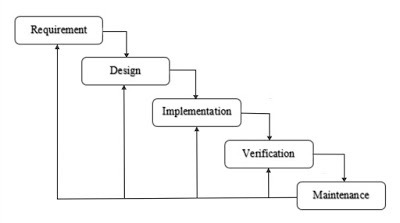
Metode yang digunakan dalam pengklasifikasian serangan DDoS ini adalah metode KNN *(K-Nearest Neighbor*). Sebelum masuk ke tahap pengujian, fitur-fitur yang ada pada data akan diseleksi terlebih dahulu. Tujuan dari seleksi fitur ini adalah untuk mencari fitur yang paling berpengaruh terhadap data sehingga akurasi yang dihasilkan dalam pengklasifikasian ini juga akan maksimal. Metode yang digunakan untuk seleksi fitur ini adalah *Information Gain*. *Information Gain* melakukan seleksi fitur dengan teknik scoring dengann menggunakan maksimal *entropy* pada masing-masing fitur. Jika fitur-fitur sudah diurutkan dari entropy tertinggi ke *entropy* terendah, selanjutkan akan dipilih 21 fitur teratas untuk pengklasifikasin dengan metode *K-Nearest Neighbor*.

Pada tahap klasifikasi dengan *K-Nearest Neighbor*, data dengan fitur-fitur yang telah dileksi akan dilatih terlebih dahulu. Kemudian data yang sudah dilatih tersebut akan digunakan untuk memprediksi data uji yang diinputkan. Dan selanjutnya ditampilkan hasil prediksi yang dilakukan dengan metode *K-Nearest Neighbor tersebut*.

Setelah melakukan prediksi terhadap data uji, selanjutnya dilakukan pengukuran performa terhadap program. Pengujian performa ini berupa pengujian akurasi dari program. Untuk pengujian performa sendiri digunakan metode *K-Fold Cross Validation*. Pada pengujian ini data akan dibagi menjadi 4 bagian dimana setiap bagian tersebut terdapat 4 partisi, 3 partisi digunakan sebagai data latih dan 1 partisi digunakan sebagai data uji. Setelah itu akan muncul seberapa besar akurasi yang dihasilkan oleh program.

### **Metodologi Pengembangan Perangkat Lunak**

Metode air terjun atau yang sering disebut metode waterfall sering dinamakan siklus hidup klasik *(classic life cycle)*, dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan *(planning)*, permodelan *(modeling)*, konstruksi *(construction)*, serta penyerahan sistem ke para pelanggan/pengguna *(deployment)*, yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan (Sasmito, 2017). Tahapan metode waterfall dapat dilihat pada gambar di bawah ini.



**Gambar 1. 1 Tahapan Metode Waterfall**

Sumber : PengetahuanDanTeknologi.com

Dalam pengembangannya metode waterfall memiliki beberapa tahapan yang berurut yaitu : *requirement* (analisis kebutuhan), *design system* (desain sistem), *coding* (pengkodean) & *testing*(pengujian), Penerapan Program, pemeliharaan. Tahapan tahapan dari metode waterfall adalah sebagai berikut (Sasmito, 2017):

1. *Requirement Analisis*

Tahap ini pengembang sistem diperlukan komunikasi yang bertujuan untuk memahami perangkat lunak yang diharapkan oleh pengguna dan batasan perangkat lunak tersebut. Informasi ini biasanya dapat diperoleh melalui wawancara, diskusi atau survei langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

1. *System Design*

Spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam fase ini dan desain sistem disiapkan. Desain Sistem membantu dalam menentukan perangkat keras(hardware) dan sistem persyaratan dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan.

1. *Implementation*

Pada tahap ini, sistem pertama kali dikembangkan di program kecil yang disebut unit, yang terintegrasi dalam tahap selanjutnya. Setiap unit dikembangkan dan diuji untuk fungsionalitas yang disebut sebagai unit testing.

1. *Integration & Testing*

Seluruh unit yang dikembangkan dalam tahap implementasi diintegrasikan ke dalam sistem setelah pengujian yang dilakukan masing-masing unit. Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

1. *Operation & Maintenance*

Tahap akhir dalam model waterfall. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

### **Evaluasi dan Validasi Hasil**

Pada tahap ini program diuji secara keseluruhan sebagai sistem yang lengkap untuk memastikan bahwa persyaratan perangkat lunak telah terpenuhi dan sesuai dengan kebutuhan serta tujuan awal dari penelitian ini. Skenario dari pengujian ini yaitu yang pertama akan dilakukan pemilahan atau penseleksian fitur dengan menggunakan metode *Information Gain.* Penseleksian fitur ini digunakan untuk mengurutkan fitur yang paling berpengaruh dalam dataset yang digunakan. Setelah mengurutkan fitur dari yang terbaik ke yang terendah maka selanjutnya akan dipilih 21 fitur yang akan digunakan untuk pengklasifikasian anomali jaringan dengan menggunakan metode KNN *(K-Nearest Neighbor).* Program akan mengklasifikasikan data yang dibagi menjadi data uji terhadap data latih yang digunakan sebagai patokan dalam pengujian data uji tersebut. Kemudian akurasi dari keberhasilan pendeteksian akan dihitung untuk mengetahui seberapa besar akurasi yang dapat dihasilkan oleh program. Pada penelitian ini akurasi diperoleh dengan membandingkan jumlah data yang berhasil dengan benar diklasifikasikan kedalam 2 buah kelas normal dan kelas serangan DDoS. untuk mendapatkan nilai akurasi dihitung dari jumlah data yang tepat dibagi dengan jumlah data. Perhitungan nilai akurasi sebagai berikut:

# **BAB II**

# **TINJAUAN PUSTAKAN**

## **Anomali Jaringan**

Anomali jaringan adalah suatu keadaan yang terjadi pada sebuah network traffic yang menyebabkan kondisi menjadi tidak normal. Anomali yang terjadi bisa dilihat melalui kenaikan lonjakan pengguna Internet, melalui serangan pada suatu traffic dan lonjakan yang tidak disengaja. Kenaikan lonjakan dapat dilihat pada saat adanya bencana yang terjadi di dunia, kompetisi atau pertandingan dan kejadian yang tidak biasa terjadi setiap hari. Secara tidak sadar, kondisi kenaikan lonjakan ini memberikan dampak negatif bagi beberapa pihak. Kenaikan lonjakan yang terjadi tersebut menimbulkan penurunan performansi dari suatu jaringan. Untuk itu, perlu dilakukan deteksi terhadap anomali yang terjadi (Imam et al., 2019). Dalam penelitian (Imam et al., 2019), disebutkan bahwa anomali jaringan adalah keadaan dimana dalam sebuah *network traffic* (lalu lintas jaringan) tidak normal. Anomali jaringan ini terkadang bisa dirasakan atau dilihat melalui kelonjakan pengguna Internet. Secata tidak sadar, kondisi lonjakan ini dapat berdampak negattif bagi beberapa pihak. Kenaikan lonjakan ini dapat mempengaruhi *bandwidth* sehingga menimbulkan penurunan peformansi dari suatu jaringan.

Adapun kelas anomali yang dapat ditemukan dalam jaringan :

1. DoS

*Denial of Service* (DoS) adalah kategori serangan, yang victim’s resources sehingga membuatnya tidak dapat menangani legitimate requests, contohnya syn flooding. Relevant features: “source bytes” dan “percentage of packets with errors”. Jenis anomali pada DoS: Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstrom, Processtable, Worm.

1. Probe

Surveillance dan serangan probing lainnya bertujuan untuk mendapatkan informasi tentang korban jarak jauh. Contohnya, port scanning. Relevant features: “duration of connection” dan “source bytes”. Jenis anomali pada PROBE: Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint.

1. R2L

Unauthorized access, dari remote machine, attacker masuk ke remote machine dan mendapatkan local access dari mesin korban. Contohnya, password guessing. Relevant features: Network level features – “duration of connection” dan “service requested” dan host level features - “number of failed login attemp- ts”. Jenis anomali pada R2L: Guess passwd, Ftp write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendemail, named

1. U2R

Unauthorized access untuk local super user (root) privileges adalah jenis serangan, dimana attacker meng- gunakan akun normal untuk masuk ke sistem korban dan mencoba untuk mendapatkan hak istimewa root / administrator dengan mengeksploitasi beberapa vulnerability pada korban, mis. serangan buffer overflow. Relevant features: “number of file creations” dan “number of shell prompts invoked”. Jenis anomali pada U2R: Buffer Overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps.

## **Serangan DoS**

Denial of Service (DoS) adalah salah satu serangan paling umum di jaringan komputer, serangan ini mencegah pengguna dari akses layanan normal karena penyerang mengkonsumsi sumber daya jaringan komputer, memori, prosesor, dan lain-lain yang lebih baik (Cahyaningtyas, Sukarno, 2019). Serangan DoS dilakukan dengan membanjiri target serangan dengan lalu lintas, atau mengirim informasi untuk memicu crash (Gunawan et al., 2018). Kelompok serangan DoS memiliki dua kelas yang berbeda : serangan logika dan serangan sumber daya (Gunawan et al., 2018)

Serangan logika menyerang dengan cara mengeksploitasi kelemahan dari software yang ada, hal ini dilakukan agar perangkat yang jauh mengalami crash atau secara substansial akan menurunkan kinerja dari perangkat tersebut. Sedangkan serangan sumber daya dilakukan dengan membanjiri sumber daya dari komputer korban (misalkan pada CPU dan memori). Serangan sumber daya juga dapat menyerang jaringan korban dengan mengirimkan aliran paket palsu yang dengan jumlah yang sangat banyak atau tidak berhenti dalam waktu beberapa lama.

Serangan DoS dapat diatasi dengan cara memblokir IP perangkat yang membanjiri perangkat korban. Tetapi karena mudah untuk memblokir serang DoS, kini muncullah serangan baru dengan metode yang sama dengan DoS tetapi menggunakan banyak perangkat. Serangan ini disebut dengan istilah Distributed Denial of Service (DDoS)

## **Serangan Distributed Denial of Services**

Serangan *Distributed Denial of Service* (DDoS) merupakan serangan yang mudah dilakukan namun sulit untuk ditanggulangi (Pratama, 2019). Sebelum melakukan serangan DDoS, penyerang akan menyiapkan komputer untuk membantu dalam penyerangan tersebut. Komputer-komputer yang membantu dalam penyerangan tersebut disebut dengan komputer zombie atau botnet, dimana komputer tersebut dikendalikan oleh sebuah server atau komputer utama untuk membantu menyerang korban dan mengakibatkan server menjadi *down* dan mengakibatkan *system error* (Adrian & Isnianto, 2016)

### **Konsep Serangan DDoS**

Konsep dari serangan DDoS adalah membanjiri lalu lintas jaringan dengan data dengan jumlah yang sangat banyak. Konsep serangan DDoS sama halnya seperti serangan DoS akan tetapi serangan DDoS dibantu dengan komputer zombie atau botnet. Adapun konsep serangan DoS adalah sebagai beriktut :

1. *Request Flooding,* konsep ini merupakan teknik yang digunakan dengan membanjiri jaringan menggunakan banyak *request.* Hal ini mengakibatkan pengguna lain yang terdaftar tidak dapat dilayani oleh server tersebut.
2. *Traffic Flooding,* konsep ini merupakan teknik yang digunakan dengan membanjiri lalu lintas jaringan dengan data dengan jumlah yang banyak. Akibatnya, pengguna lain tidah bisa dilayani oleh server.
3. Mengubah sistem konfigurasi atau bahkan merusak komponen dan server, tetapi tidak banyak yang menggunakan cara ini karena sulit untuk dilakukan.

### **Metode Serangan DDoS**

Serangan DDoS adalah teknik penyerangan yang memiliki banyak cara untuk melakukan serangan, seperti menggunakan virus dan botnet.

1. Virus

Penyerang sengaja menciptakan virus untuk menjalankan bot melalui script yang berjalan pada sistem operasi. Penyerang akan menyebarkan virus melalui file yang dibagikan ke berbagai situs yang terhubung dengan internet. Bahkan beberapa virus juga dapat mengambil hak akses dari perangkat yang sudah men*download script* dan dijalankan pada sistem operasi. Jika komputer sudah terinveksi virus DDoS maka virus tersebut akan melakukan serangan DDoS secara aktif ke server atau ke alamat IP tertentu yang sudah ditentukan.

1. Botnet

Dalam melakukan serangan DDoS, serangan DDoS dibantu dengan kumpulan bot yang dijalankan secara bersama-sama. Bot disisipkan pada malware yang kemudian di tanam ke komputer yang terhubung ke jaringan internet. Jumlah komputer yang dijadikan bot bisa mencapai puluhan sampai dengan jutaan, hal ini tergantung dari banyaknya komputer yang telah terinfeksi malware. Semua komputer yang terinfeksi malware tersebut disebut dengan botnet, sedangkan satu komputer yang terinfeksi malware tersebut disebut dengan komputer zombie.

## **Data Mining**

Data mining adalah langkah analisis terhadap proses penemuan pengetahuan di dalam basis data atau *Knowledge Discovery in Database* yang disingkat KDD. *Data mining* didefinisikan sebagai penemuan dari pola-pola baru yang berasal dari kumpulan data yang sangat besar, meliputi metode-metode dari *artificial intelligence, machine learning, statistic,* dan *database system.* Data mining ditujukan untuk mengekstrak (mengambil intisari) pengetahuan dari sekumpulan data sehingga didapatkan struktur yang dapat dimengerti menusia serta meliputi bisis data dan menagemen data, prapemrosesan data, pertimbangan model dan inferensi, ukuran keterkatikan, pertimbangan kompleksitas, pascapemrosesan terhadap struktur yang ditemukan, visualisasi, dan *online updating* (Suyanto, 2017). Data mining atau penambangan data adalah teknik yang relative cepat dan mudah untuk menemukan pengetahuan, pola dan/atau relasi antar data, secara otomatis.

Secara umum, kegunaan dari data mining dapat dibagi menjadi dua yaitu deskriptif dan prediktif. Deskriptif berarti data mining digunakan untuk mencari pola-pola yang dapt fipahami manusia mrnjrlaskan karakteristik data. Sedangkan prediktif berarti data mining digunakan untuk membentuk sebuah model pengetahuan yang akan digunakan untuk melakukan prediksi. Berdasarkan fungsionalitasnya, tugasnya data mining dikelompokkan menjadi beberapa bagian (Mardi, 2017).

1. *Description* (Deskripsi)

Pada bagian ini, biasanya akan dicoba untuk pencarian cara untuk menggambarkan pola serta kecenderungan yang terdapat dalam data. Sebagai contoh, petugas pengumpulan suara mungkin tidak dapat menemukan keterangan atau fakta bahwa siapa yang tidak cukup profesional akan sedikit didukung dalam pemilihan presiden.

1. *Estimation* (Estimasi)

Model ini dibangun dengan menggunakan record lengkap yang menyediakan nilai dari variabel target sebagai nilai prediksi. Selanjutnya, pada peninjauan berikutnya estimasi nilai dari variabel target dibuat berdasarkan nilai variabel prediksi. Contohnya melakukan estimasi IPK dengan melihat nilai indeks prestasi dari mahasiswa.

1. *Prediction* (Prediksi)

Prediksi serupa dengan klasifikasi dan estimasi, hanya saja nilai dari hasil akan ada dimasa mendatang. Contohnya memprediksi tingkat pengangguran pada lima tahun kedepan.

1. *Classification* (Klasifikasi)

Pada bagian klasifikasi biasanya terdiri dari target variabel kategori. Contohnya penggolongan kriteria penjelajahan internet dipisahkan dalam dua kategori yaitu berupa serangan, atau tidak.

1. *Clustering* (Pengklusteran)

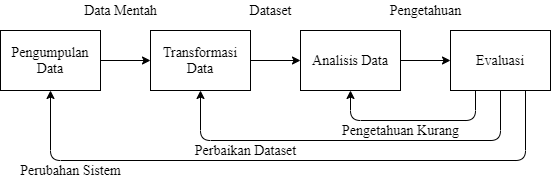
Pengklusteran berbeda dengan klasifikasi karena dalam clustering tidak terdapat variabel target. Pengklsteran merupakan pengelompokan record, pengamatan, atau memperhatikan dan membentuk kelas objek-objek yang memil iki kemiripan. Sebagai contoh, melakukan pengklusteran terhadap ekspresi dari gen, untuk mendapatkan kemiripan perilaku dari gen dalam jumlah besar.

1. *Association* (Asosiasi)

Tugas asosiasi dalam data mining adalah menemukan atribut yang muncul dalam satu waktu. Sebagai contoh asosiasi adalah Meneliti jumlah pelanggan dari perusahaan telekomunikasi seluler yang diharapkan untuk memberikan respons positif terhadap penawaran upgrade layanan yang diberikan.

## **Proses Data Mining**

Proses data mining tersusun atas tiga tahapan, yaitu pengumpulan data *(data collection),* transformasi data *(data transformation),* dan analisis data *(data analysis)* (Ayub, n.d.). Proses tersebut diawali dengan *preprocessing* yang terdiri atas pengumpulan data untuk menghasilkan data mentah (raw data) yang akan digunakan dan dibutuhkan oleh data mining. Kemudian setelah data metah didapatkan tahap selanjutnya adalah transformasi data. Transformasi data ini bertujuan untuk mengubah data mining, misalkan melalui filtrasi atau agregasi. Kemudian hasil dari transformasi data tersebut akan digunakan oleh analisis data untuk membangkitkan pengetahuan dengan menggunakan teknik seperti analisis statistic, *machine learning,* dan visualisasi informasi.



**Gambar 2. 1 Kerangka Proses Data Mining**

Pada Gambar 2.1 ditunjukkan diagram yang menggambarkan aliran informasi damlam proses data mining yang diadaptasi dari (Ayub, n.d.). Proses data mining pada gambar tersebut ditunjukkan sebagai proses yang iterative. Hasil evaluasi pengetahuan yang dihasilkan data mining dapat menimbulkan kebutuhan pengetahuan yang lebih lengkap, perbaikan kumpulan data (dataset) atau perubahan pada sistem.

## **Seleksi Atribut**

Seleksi atribut atau yang sering juga dikenal dengan istilah seleksi fitur adalah salah satu bagian dari *preprocessing* data. Seleksi atribut ini adalah teknik yang digunakan untuk menghilangkan atribut yang kurang berpengaruh terhadap data sehingga atribut tersebut boleh saja tidak digunakan untuk tahap selanjutnya. Jika pada suatu kasus memiliki ratusan atau bahkan ribuan atribut namun tidak semua atribut tersebut penting untuk diperhatikan, artinya atribut atau dimensi data bisa direduksi.

## **Klasifikasi**

Dalam buku Dr. Suyanto, S.T, M.Sc, klasifikasi yaitu bagaimana mempelajari sekumpulan data sehingga dihasilkan aturan yang bisa mengklasifikasikan atau mengenali data-data baru yang belum pernah dipelajari (Suyanto, 2017). Klasifikasi dapat didefinisikan sebagai proses untuk menyatakan suatu objek data sebagai salah satu kategori (kelas) yang telah didefinisikan sebelumnya. Klasifikasi banyak digyunakan dalam bebagai aplikasi, diantaranya adalah deteksi kecurangan *(fraud detection),* pengelolaan pelanggan, diagnosis medis, prediksi penjualan, dan sebagainya.

Model kladifikasi lebih sering dibangun menggunakan teknik pembelajaran dalam bidang *machine learning.* Proses pembelanjaran secara otomatis terhadap suatu himpunan data mampu menghasilkan model klasifikasi (fungsi target) yang memetakan objek data x (input) ke dalah satu kelas y yang telah didefinisikan sebelumnya. Jadi, proses pembelajaran memerlukan masukan *(input)* berupa himpunan data latih *(training set)* yang berlabel (memiliki atribut kelas) dan mengeluarkan output yang berupa sebuah model klasifikasi.

Ada banyak teknik klasifikasi yang diusulkan oleh para ahli, yang dapat dikelompokkan kedalam dua kategori, yaitu teknik klasifikasi global atau memperhitungkan semua data latih dan teknik klasifikasi lokal atau teknik ini hanya memperhitungkan sebagian dari data latih. Masing-masing teknik tersebut mepunyai kelebihan dan kekurangan, serta kekuatan dan kelemahan.

## ***Information Gain***

*Information Gain* merupakan metode seleksi fitur yang paling sederhana, yaitu dengan cara melakukan perangkaian atribut. Information gain mendeteksi fitur-fitur yang paling banyak memiliki informasi berdasarkan kelas tertentu, penentuan atribur ini dilakukan dengan cara menghitung nilai entropy terlebih dahulu (Syafitri Hidayatul AA, Yuita Arum S, 2018). *Entropy* adalah suatu parameter untuk mengukur heterogenitas (keberagaman) dalam suatu himpunan data, semakin heterogen suatu himpunan data, semakin besar pula nilai entropy-nya (Suyanto, 2017). Secara matematis, *entropy* dirumuskan pada persamaan (1)

(2)

Dimana c adalah jumlah nilai yang terdapat pada atribut target (juumlah kelas). Sedangkan menyatakan posisi atau ratio antara jumlah sampel di kelas dengan jumlah semua sampel pada himpunan tersebut.

Secara sistematis, *information gain* dirumuskan pada persamaan (2) (Suyanto, 2017).

(2)

Dimana :

: atribut

: menyatakan suatu nilai yang mungkin untuk atribut A

: himpunan nilai-nilai yang mungkin untuk atribut A

: jumlah sample untuk nilai v

: jumlah seluruh sampel data

: entropy untuk sampel-sampel yang memiliki nilai v

## ***K-Nearest Neighbor***

Algoritma dibagi menjadi dua jenis, yaitu algoritma supervised dan algoritma unsupervised. Algoritma supervised learning memiliki tujuan untuk menemuka pola baru dalam sebuah data dengan mrnghubungkan pola data yang sudah ada dengan data baru, sedangkan pada algoritma unsupervised learning, data belum memiliki pola apapun dan tujuan dari algoritma unsupervised learning adalah uuntuk menemukan pola dalam sebuah data (Liantoni, 2016). Algoritma *K-Nearest Neighbor* termasuk algoritma supervised karena algoritma ini bertujuan untuk mengklasifikasika objek baru berdasarkan atribut dan data training.

Dalam buku yang dikarang oleh Dr. Suyanto, S.T., M.Sc., algoritma *K-Nearest Neighbor* bekerja dengan mencari sejumlah kelompok k objek data atau po;a (dari semuaa pola latih yang ada) yang paling dekat dengan pola masukan, kemudian memilih kelas dengan jumlah pola terbanyak di antara k pola tersebut. Untuk menentukan atau mendefisinikasn jarak antar objek maka digunakan rumus *Eucledian* pada persamaan (3) (Mustakim, 2016)

(3)

Dimana :

: jarak antara titik pada data training dan data testing yang akan diklasifikasikan

: nilai dari data training

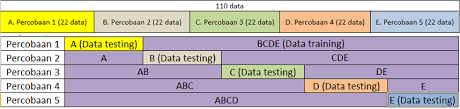
: nilai dari data testing

: nilai atribut

: dimensi data atribut

## ***K-fold Cross Validation***

*K-Fold Cross Validation* merupakan salah satu metode yang digunakan untuk mengetahui rata-rata keberhasilan dari suatu sistem dengan cara melakukan perulangan dengan mengacak atribut masukan sehingga sistem tersebut teruji untuk beberapa atribut input yang acak (Mutiara, 2015). Metode *k-Fold Cross Validation* mempartisi himpunan data *D* secara acak menjadi *k fold* (subhimpunan) yang saling bebas, sehingga masing-masing *fold* berisi 1/*k* bagian data. Untuk himpunan data yang akan dibangun, masing-masing dari himpunan data tersebut akan memiliki (k – 1) *fold* data latih atau data *training* dan 1 *fold* data untuk data uji atau data *testing* (Suyanto, 2017).Contoh pembagian dataset kedalam proses 5-*Fold Cross Validation* dapat dilihat pada Gambar .



**Gambar 2. 2 5-Fold Cross Validation**

Sumber : Pencarian k-Optimal pada Algoritma KNN untuk Prediksi Kelulusan Tepat Waku Mahasiswa Berdasarkan IP Sampai Dengan Semester 4 (Studi Kasus : Mahasiswa Program Studi Ilmu Komputer FMIPA Unlam). 2015.

Cara kerja *k-Fold Cross Validation* adalah sebagai berikut :

1. Total instance dibagi menjadi n bagian
2. *Fold* ke-1 adalah ketika bagian ke-1 menjadi data uji (testing data) dan sisanya menjadi data latih (training data). Selanjutnya, hitung akurasi berdasarkan porsi data tersebut. Perhitungan akurasi tersebut dengan menggunakan persamaan (4) :

(4)

1. Fold ke-2 adalah ketika bagian ke-2 menjadi data uji (testing data) dan sisanya menjadi data latih (training data). Selanjutnya, hitung akurasi berdasarkan porsi data tersebut.
2. Demikian seterusnya hingga mencapai fold ke-K. Hitung rata-rata akurasi dari K buah akurasi. Dan rata-rata akurasi ini menjadi akurasi final.

# **BAB III**

# **ANALISIS DAN PERANCANGAN**

## **Variable Penelitian**

Penelitian yang dilakukan oleh penulis dengan judul “Klasifikasi Serangan *Distributed Denial Of Service* (DDoS) dengan Algoritma *K-Nearest Neighbor* (KNN)”, dimana variable yanag digunakan sebagai tolak ukur keberhasilan dari implementasi sistem klasifikasi data ini adalah pengelompokan atau klasifikasi data berdasarkan tingkat kemiripan tertinggi dari data uji terhadap data latih. Yang perlu diperhatikan dalam sistem ini adalah apakah klasifikasi yang dihasilkan oleh sistem mencangkup data dari paket data dengan tingkat kemiripan tertinggi melalui perhitungan jarak *Eucledian*. Data yang digunakan pada penelitian ini yaitu dataset yang berasal dari NSL-KDD.

Data intrusi yang digunakan untuk penelitian diambil dari dataset NSL-KDD yang mana dataset ini juga sudah banyak digunakan sebagai patokan oleh banyak peneliti. Tabel 3.1 menunjukan jumlah paket data yang digunakan dalam penelitian untuk mengevaluasi rules dan pengujian data guna mendeteksi intrusi. Jumlah fitur dari dataset NSL-KDD yang digunakan dalam penelitian ini adalah sebanyak 41 fitur (dapat dilihat pada Tabel 3.2, Tabel 3.3, dan Tabel 3.4), koneksi diberi label 0 atau 1. Data dikategorikan menjadi dua kelas ketegori utama yaitu DoS *(Deniala of Service)* dan normal (Tabel 3.1). Dalam dataset ini ada 3 group fitur, yaitu fitur dasar *(basic)*, *content based* dan *time based features*.

**Tabel 3. 1 Jumlah Paket Data**

|  |  |  |
| --- | --- | --- |
| **No** | **Nama** | **Jumlah** |
| 1 | Normal | 20000 |
| 2 | Back | 956 |
| 3 | Land | 18 |
| 4 | Neptune | 16179 |
| 5 | POD *(Ping of Death)* | 201 |
| 6 | Smurf | 2646 |

Pada penelitian ini, jumlah dataset yang digunakan adalah sebanyak 40000 data dimana data ini adalah data yang terdiri dari data normal dan data serangan. Serangan yang dideteksi oleh sistem hanya serangan DoS *(Denial of Service).* Nama dari serangan DoS *(Denial of Service)* yang diteksi yaitu *Back, Land, Neptune, Ping of Death (POD), dan Smurf.*

**Tabel 3. 2 Fitur Dasar (Basic) Tiap-tiap Koneksi TCP**

|  |  |  |
| --- | --- | --- |
| **Nama Fitur** | **Keterangan** | **Tipe** |
| duration | lamanya (detik) koneksi | continuous |
| protocol\_type | Tipe protokol (ICMP, TCP, UDP) | discrete |
| service network | service pada tujuan (http, telnet, dll) | discrete |
| src\_bytes | Jumlah rata-rata byte, termasuk informasi header yang diterima oleh destination host | continuous |
| dst\_bytes | Jumlah rata-rata byte termasuk informasi header yang diterima oleh source host | continuous |
| flag status, | normal atau error dari koneksi | discrete |
| Land | 1 jika koneksi bersal dari/ke host yang sama/port; 0 jika tidak | Discrete |
| wrong\_fragment | jumlah ``wrong'' fragments | continuous |
| urgent | jumlah dari paket urgent | continuous |

**Tabel 3. 3 Fitur Konten pada Koneksi (Conection Based)**

|  |  |  |
| --- | --- | --- |
| **Nama fitur** | **Keterangan** | **Tipe** |
| hot | jumlah dari dari indikator ``hot'' | Continuous |
| num\_failed\_logins | jumlah dari percobaan login yang gagal | continuous |
| logged\_in | 1 jika berhasil login; 0 jika tidak | Discrete |
| num\_compromised | jumlah dari kondisi ``compromised'' | Continuous |
| root\_shell | 1 jika root shell dijalankan; 0 jika tidak | Discrete |
| su\_attempted | 1 jika ada percobaan perintah ``su root'' ; 0 jika tidak | Discrete |
| num\_root | jumlah akses ``root'' | Continuous |
| num\_file\_creations | jumlah dari file yang dimunculkan oleh operasi | Continuous |
| num\_shells | jumlah dari shell prompts | Continuous |
| num\_access\_files | jumlah dari operasi pada file kontrol akses | Continuous |
| num\_outbound\_cmds | jumlah dari perintah outbound dalam sesi ftp | Continuous |
| is\_hot\_login | 1 jika login termasuk daftar``hot'' ; 0 jika tidak | Discrete |
| is\_guest\_login | 1 jika login adalah login ``guest''; 0 jika tidak | Discrete |

**Tabel 3. 4 Fitur Trafik Waktu (Time Based Features)**

|  |  |  |
| --- | --- | --- |
| **Nama fitur** | **Keterangan** | **Tipe** |
| count | Fitur yang mengindikasikan jumlah koneksi pada host yang sama | Continuous |
| serror\_rate | % dari koneksi ``SYN'' errors | continuous |
| rerror\_rate | % dari koneksi yang mempunyai ``REJ'' errors | Continuous |
| same\_srv\_rate | % dari konkesi ke service yang sama | Continuous |
| diff\_srv\_rate | % dari koneksi ke services yang berbed | Continuous |
| srv\_count | Jumlah koneksi pada service sama untuk koneksi yang sama dalam 2 detik terakhir | Continuous |
| srv\_serror\_rate | % dari koneksi yang mempunyai ``SYN'' errors | Continuous |
| srv\_rerror\_rate | % dari koneksi yang mempunyai ``REJ'' errors | Continuous |
| srv\_diff\_host\_rate | % dari koneksi ke hosts yang berbeda | Continuous |
| num\_outbound\_cmds | jumlah dari perintah outbound dalam sesi ftp | Continuous |
| is\_hot\_login | 1 jika login termasuk daftar``hot'' ; 0 jika tidak | Discrete |
| is\_guest\_login | 1 jika login adalah login ``guest''; 0 jika tidak | Discrete |

**Tabel 3. 5 Fitur Dataset**

|  |  |
| --- | --- |
| **No** | **Nama** |
| 1 | Duration |
| 2 | protocol\_type |
| 3 | Service |
| 4 | Flag |
| 5 | src\_bytes |
| 6 | dst\_bytes |
| 7 | Land |
| 8 | wrong\_fragment |
| 9 | Urgent |
| 10 | Hot |
| 11 | num\_failed\_logins |
| 12 | logged\_in |
| 13 | num\_compromised |
| 14 | root\_shell |
| 15 | su\_attempted |
| 16 | num\_root |
| 17 | num\_file\_creations |
| 18 | num\_shells |
| 19 | num\_access\_files |
| 20 | num\_outbound\_cmds |
| 21 | is\_host\_login |
| 22 | is\_guest\_login |
| 23 | Count |
| 24 | srv\_count |
| 25 | serror\_rate |
| 26 | srv\_serror\_rate |
| 27 | rerror\_rate |
| 28 | srv\_rerror\_rate |
| 29 | same\_srv\_rate |
| 30 | diff\_srv\_rate |
| 31 | srv\_diff\_host\_rate |
| 32 | dst\_host\_count |
| 33 | dst\_host\_srv\_count |
| 34 | dst\_host\_same\_srv\_rate |
| 35 | dst\_host\_diff\_srv\_rate |
| 36 | dst\_host\_same\_src\_port\_rate |
| 37 | dst\_host\_srv\_diff\_host\_rate |
| 38 | dst\_host\_serror\_rate |
| 39 | dst\_host\_srv\_serror\_rate |
| 40 | dst\_host\_rerror\_rate |
| 41 | dst\_host\_srv\_rerror\_rate |

## **Analisis Kebutuhan**

Pada tahap ini, dilakukan analisis kebutuhan terhadap penelitian yang dilakukan oleh peneliti. Analisis kebutuhan ini meliputi kebutuhan yang diperlukan oleh sistem yang di bangun, kebutuhan sistem tersebut meliputi data yang digunakan pengoperasian sistem dan referensi sebagai sumber pembelajaran yang digunakan sebagai patokan dalam pembuatan sistem. Teknik pengumpulan data yang digunakan dalam penelitian ini menggunakan metode studi literatur. Studi literatur digunakan untuk mengetahui kebutuhan dari penelitian pada sistem klasifikasi yang dibangun.

1. Jenis data yang digunakan pada penelitian ini adalah data sekunder. Data sekunder adalah data yang sudah ada sebelumnya dan sengaja dikumpulkan oleh peneliti sendiri untuk melengkapi kebutuhan data penelitian. Dalam penelitian ini data yang digunakan adalah data riset bersumber dari NSL-KDD.
2. Teknik Studi Literatur, teknik studi literatur dilakukan dengan cara mempelajari referensi baik itu dari jurnal maupun buku yang berkaitan dengan anomali jaringan, metode *Information Gain,* dan metode *K-Nearest Neighbor* (KNN) yang dapat menunjang penelitian ini. Studi literatur dilakukan dengan mengambil literatur – literatur pendukung dari jurnal – jurnal ilmiah, baik itu dari jurnal dalam negeri maupun jurnal luar negeri dan juga dapat dilakukan dengan mempelajari buku – buku terkait. Dalam studi literatur ini, penulis mencari sumber yang berkaitan dengan permasalahan yang perlu menjadi perbaikan dalam penelitian selanjutnya.

## **Analisis Sistem**

Pada tahap ini, dilakukan analisis kebutuhan sistem berupa perangkat yang digunakan untuk menganalisis proses dari sistem tersebut. Perangkat yang digunakan untuk membantu menganalisis kebutuhan sistem adalah *flowchart* (diagram alir). *Flowchart* data digunakan untuk menyusun input, proses, dan juga output dari sistem klasifikasi serangan DDoS yang dikembangkan. Hasil dari tahapan ini adalah spesifikasi kebutuhan sistem dan analisa dari model proses sistem yaitu penentuan input – input dari sistem, penentuan proses – proses dari sistem, dan juga penentuan output dari pemrosesan input sistem.

### **Penentuan Input – input Sistem**

Input – input yang digunakan dalam sistem yaitu data yang dimasukan ke sistem kemudian data tersebut akan diproses oleh sistem dan selanjutnya menghasilkan output yang sudah ditentukan sebelumnya. Data yang dimasukkan kedalam sistem adalah data berupa kumpulan data riset yang akan dilakukan tahap klasifikasi pada data tersebut. Dataset yang digunakan dalam penelitian ini berperan sebagai kumpulan data – data yang belum diolah dan akan dijadikan sebagai objek penelitian. Dataset yang digunakan adalah dataset yang berasal dari NSL-KDD dengan format file \*.csv.

Fitur dari input data pada sistem ini akan dipilih sesuai dengan nilai *Entropy* terbesarnya sebelum masuk ke tahap pembagian kumpulan data. Metode yang digunakan dalam pemilihan atau seleksi fitur ini adalah *Information Gain.* Pembagian kumpulan data ini dibagi menjadi dua, yaitu data pelatihan dan data pengujian. Dataset yang dibagi menjadi data pelatihan ini digunakan oleh metode *K-Nearest Neighbor* (KNN) untuk membuat kelas – kelas. Sedangkan data pengujian digunakan untuk mengevaluasi keakuratan model dari sistem ini. Kemudian sistem membagi kumpulan data secara acak kedalam kereta dan dataset dengan menggunakan metode *K-Fold Cross Validation.*

Dalam dataset NSL-KDD training dari tiga protokol yang memiliki 23 tipe serangan ini dibagi menjadi dua kelas, yaitu normal dan serangan. Setiap penyimpangan dari prilaku normal dikatakan sebagai serangan.

**Tabel 3. 6 Nama Serangan setiap Tipe Protokol**

|  |  |
| --- | --- |
| **Protocol tipe** | **Nama Serangan** |
| UDP | normal, teardrop, satan, nmap, rootkit |
| TCP | normal, neptune, guess\_passwd, land, portsweep, buffer\_overflow, phf, warezmaster, ipsweep, multihop, warezclient, perl, back, ftp\_write, loadmodule, satan, spy, imap, rootkit |
| ICMP | normal, portsweep, ipsweep, smurf, satan, pod, nmap |

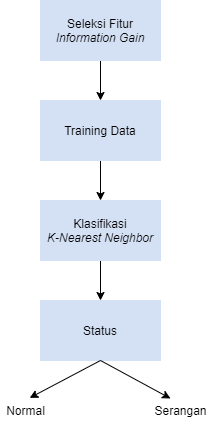
### **Penentuan Proses-proses dalam Operasi Sistem**

Secara garis besar, proses – proses yang dilakukan pada sistem klasifikasi serangan *Distributed Denial of Service* (DDoS) ini adalah sebagai berikut :

1. Proses input dataset ke dalam sistem.
2. Proses ekstrasi fitur dari dataset menggunakan metode *Information Gain* untuk menghilangkan fitur – fitur yang tidak dibutuhkan.
3. Proses normalisasi dataset dan pengujian terhadap data uji pada rentang 0 – 1.
4. Proses menghitung kesamaan data uji dengan dataset untuk menentukan kelas data uji dengan menggunakan perhitungan jara *Eucledian.*
5. Proses klasifikasi dengan metode *K-Nearest Neighbor* (KNN).
6. Proses penghitungan akurasi sistem dengan metode *K-Fold Cross Validation.*

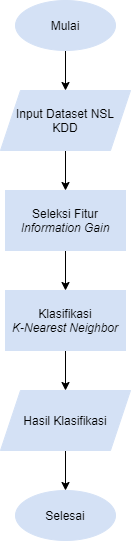
## **Perancangan Sistem**

Pada Gambar 3.1 dijelaskan desain sistem untuk pendeteksian anomali jaringan akibat serangan *Distributed Denial of Service* (DDoS) menggunakan metode *K-Nearest Neighbor (KNN).* Dalam sistem ini terdapat dua proses utama. Pertama, sistem melakukan seleksi fitur terhadap dataset. Tujuan dari seleksi fitur ini adalah untuk mendapatkan fitur yang paling relevan terhadap data. Metode yang digunakan dalam tahap seleksi fitur ini adalah metode *Information Gain*. Setelah mendapatkan fitur – fitur yang digunakan dalam sistem, selanjutnya dilakukan pengelompokkan dataset dimana data ini pada tahap ini data dikelompokkan ke dalam 2 tipe yaitu normal dan serangan. Tipe serangan disini adalah serangan DDoS yaitu *back, land, neptune, ping of death* (POD)*,* dan *smurf.* Klasifikasi data tersebut dilakukan dengan menggunakan metode *K-Nearest Neighbor* (KNN). Sistem ini menggunakan dataset NSL-KDD, dimana dataset ini juga sudah banyak digunakan dalam penelitian sistem deteksi anomali jaringan.



**Gambar 3. 1 Perancangan Sistem**

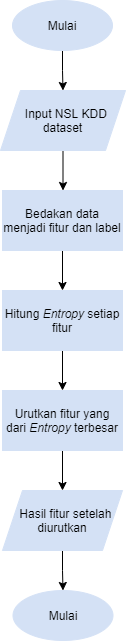
## **Flowchart Sistem**



**Gambar 3. 2 Flowchart Sistem**

Sistem menerima input dataset yaitu dataset NSL-KDD. Kemudian fitur – fitur dari data tersebut diseleksi dengan menggunakan metode *Informatin Gain*. Selanjutnya dilakukan normalisasi terhadap atribut dataset yang memiliki tipe data kategorial. Setelah dilakukan normalisasi data, selanjutnya data tersebut dihitung dengan menggunakan metode *K-Nearest Neighbor* (KNN). Kemudian hasil dari perhitungan KNN tersebut akan dideteksi apakah paket dari data uji atau normal atau serangan.

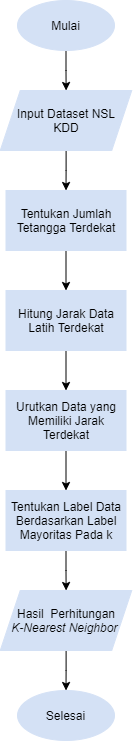
## **Flowchart Information Gain**



**Gambar 3. 3 Flowchart Information Gain**

Pada proses seleksi fitur dengan menggunakan metode *Information Gain,* tahap pertama yang dilakukan adalah memasukkan dataset NSL-KDD. Kemudian data dibedakan sesuai kebutuhan yaitu menjadi fitur dan label. Selanjutnya sistem menghitung nilai *Entropy* dari masing – masing fitur dan kemudian mengurutkan fitur yang memliki nilai *Entropy* paling besar sampai paling kecil. Dan tahap terakhir pada seleksi fitur ini adalah sistem menampulkan diagram nilai *Entropy* dari semua fitur kemudian dapat dipilih fitur mana saja yang digunakan untuk proses selanjutnya.

## **Flowchart K-Nearest Neighbor**



**Gambar 3. 4 Flowchart K-Nearest Neighbor**

Tahap pertama yang dilakukan pada proses ini adalah melakukan input data dari dataset NSL-KDD. Setelah dilakukan input data, selanjutnya tentukan jumlah tetangga terdekat. Penentuan jumlah tetangga terdekat ini dapat ditentukan oleh user, hasil yang dari penentuan jumlah tetangga ini mungkin akan berbeda pada setiap inputannya. Kemudian hitung jarak data latih dengan menggunakan rumus *Eucledian.* Selanjutnya urutkan data latih yang memiliki jarak paling kecil terhadap data uji. Setelah mendapatkan jarak data latih terdekat dengan k yang sudah ditentunkan sebelumnya maka beri label pada data uji apakah data uji tersebut termasuk kedalam data normal atau serangan. Tampilkan hasil yang diperoleh dari pengujian data tersebut.

## **Skenario Pengujian Sistem**

Menurut (Rouf, 2012), pengujian perangkat lunak sangat diperlukan dalam sebuah sitem, dengan melakukan pengujian perangkat lunak ini dapat diketahui kesalahan atau error yang terjadi dalam sistem itu sendiri. Pada tahap pengujian sistem pada penelitian ini, peneliti menggunakan scenario pengujian sebagai berikut.

Skenario dari pengujian ini yaitu yang pertama akan dilakukan pemilahan atau penseleksian fitur dengan menggunakan metode *Information Gain.* Penseleksian fitur ini digunakan untuk mengurutkan fitur yang paling berpengaruh dalam dataset yang digunakan. Setelah mengurutkan fitur dari yang terbaik ke yang terendah maka selanjutnya akan dipilih 9 fitur yang akan digunakan untuk pengklasifikasian anomali jaringan dengan menggunakan metode KNN *(K-Nearest Neighbor).* Program akan mengklasifikasikan data yang dibagi menjadi data uji terhadap data latih yang digunakan sebagai patokan dalam pengujian data uji tersebut. Kemudian akurasi dari keberhasilan pendeteksian akan dihitung untuk mengetahui seberapa besar akurasi yang dapat dihasilkan oleh program. Pada penelitian ini akurasi diperoleh dengan membandingkan jumlah data yang berhasil dengan benar diklasifikasikan kedalam 2 buah kelas normal dan kelas serangan DDoS. untuk mendapatkan nilai akurasi dihitung dari jumlah data yang tepat dibagi dengan jumlah data. Perhitungan nilai akurasi sebagai berikut :

# **BAB IV**

# **HASIL DAN PEMBAHASAN**

## **Gambaran Umum Sistem**

Pada bab ini membahas gambaran umum dari sistem klasifikasi serangan *Distributed Denial of Service* (DDoS). Dalam penelitian ini, penulis menggunakan data yang berasal dari NSL-KDD dengan jumlah data sebanyak 40000 data dimana data tersebut terdiri dari data normal dan data serangan *Denial of Service* (DoS) yaitu *back, land, neptune, ping of death* (POD), dan *smurf.* Penelitian ini juga menggunakan 41 atribut yang ada pada dataset NSL-KDD.

Pada tahapan ini dibutuhkan masukkan data berupa file dataset yang memiliki format \*.csv. Sebelum masuk ke proses klasifikasi sebelumnya fitur – fitur dari dataset diseleksi terlebih dahulu dan ditentukan jumlah fitur yang digunakan untuk tahap selanjutnya. Penulis menggunakan label 0 dan 1 untuk membedakan kelas pada sistem, dimana 0 artinya data tersebut adalah data yang masuk kedalam kelas normal dan 1 artinya data tersebut adalah data yang masuk pada kelas serangan.

## **Lingkungan Perancangan dan Implementasi Sistem**

Perancangan sistem dilakukan pada sistem operasi Windows 10*.* Sistem dirancang di sebuah laptop yang memiliki prosessor Intel® Core™ i7-7700HQ CPU @ 2.80GHz (8 CPUs), ~2.8GHz dan RAM sebesar 8 GB.

Sistem implementasi dengan menggunakan bahasa pemrograman Pyhton. Dalam perancangan dan implementasi sistem ini, digunakan beberapa perangkat lunak untuk dapat memenuhi kebutuhan implementasi sistem. berikut adalah perangkat lunak yang digunakan.

1. Anaconda Navigator
2. Jupyter Notebook
3. Microsoft Excel
4. Google Chrome

## **Implementasi Program**

Implementasi program pada sub bab ini membahas mengenai implementasi perancangan yang sebelumnya telah dibuat kedalam bahasa pemrograman. Adapun pembahasan dalam kode program ini adalah sebagai berikut.

### **Proses Pemilihan Fitur dengan Information Gain**

Pada tahap seleksi fitur ini metode yang digunakan dalam melakukan seleksi fitur adalah *Information Gain.* Seleksi fitur ini bertujuan untuk memilih fitur yang paling relevan terhadap proses klasifikasi yang selanjutnya dilakukan.

Adapun tahapan dari pengklasifikasian dataset NSL-KDD adalah sebagai berikut.

1. Tahap pertama yang dilakukan adalah memisahkan kolom fitur dengan kolom label. Pemisahan ini bertujan untuk menandai label yang akan digunakan pada tahap pengklasifikasian

**Tabel 4. 1 Pemisahan Fitur dan Label Dataset**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | fitur = data.drop('label', axis = 1) |
| 2 | target = data['label'] |
| 3 | fitur.shape, target.shape |

1. Tahap berikutnya adalah membagi data menjadi dua yaitu data latih dan data uji.

**Tabel 4. 2 Pembagian Data Latih dan Data Uji**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | X\_train, X\_test, y\_train, y\_test = train\_test\_split(fitur, target, test\_size = 0.2, random\_state = 0) |

1. Selanjutnya lakukan ekstrasi fitur dengan threshold = 0.0 (ditentukan oleh user).

**Tabel 4. 3 Ekstraksi Fitur**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | constant\_filter = VarianceThreshold(threshold=0.0) |
| 2 | constant\_filter.fit(X\_train) |
| 3 | X\_train\_filter = constant\_filter.transform(X\_train) |
| 4 | X\_test\_filter = constant\_filter.transform(X\_test) |

1. Hitung nilai *Entropy* pada setiap fitur. Disini perhitungan nilai *Entropy* menggunakan library *Random Forest Classifier*.

**Tabel 4. 4 Menghitung Nilai Entropy Tiap Fitur**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | mi = mutual\_info\_classif(X\_train\_unique, y\_train) |

1. Tahap selanjutnya adalah urutkan nila *Entropy* dari fitur mulai dari nilai paling besar hingga nilai paling kecil.

**Tabel 4. 5 Mengurutkan Nilai Entropy Tiap Fitur**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | mi.sort\_values(ascending=False, inplace = True) |

### **Preprocessing Dataset**

Dalam proses preprosesing ini tahapan yang dilakukan adalah tranformasi dan normalisasi dataset yang yang digunakan dalam penelitian yaitu NSL-KDD. Tahap pertama yaitu dilakukan tranformasi file dataset yang masih berupa format \*.cvs ke dalam bentuk array untuk memudahkan proses preprosesing.

**Tabel 4. 6 Transformasi Dataset**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | with open('21fitur.csv') as csvfile : |
| 2 | readCSV = csv.reader(csvfile) |
| 3 | df1=pd.read\_csv("21fitur.csv") |

Tahap selanjutnya adalah mengubah label pada dataset untuk mengenali apakah data tersebut adalah data normal atau serangan. Untuk label yang digunakan pada program ini adalah 0 dan 1, dimana 0 berarti normal dan 1 berarti serangan.

**Tabel 4. 7 Mengubah Nilai Label**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | newar=[]; |
| 2 | label =0 |
| 3 | label\_array = [] |
| 4 | for row in readCSV: |
| 5 | if len(row)>21: |
| 6 | if row[21]=='normal': |
| 7 | label = 0 |
| 8 | if row[21]=='back': |
| 9 | label = 1 |
| 10 | if row[21]=='land': |
| 11 | label = 1 |
| 12 | if row[21]=='neptune': |
| 13 | label = 1 |
| 14 | if row[21]=='pod': |
| 15 | label = 1 |
| 16 | if row[21]=='smurf': |
| 17 | label = 1 |
| 18 | data\_ = [float(row[0]), float(row[1]), float(row[2]), float(row[3]),float(row[4]),float(row[5]),float(row[6]),float(row[7]), float(row[8]), float(row[9]), float(row[10]), float(row[11]), float(row[12]), float(row[13]), float(row[14]), float(row[15]), float(row[16]), float(row[17]), float(row[18]), float(row[19]), float(row[20]), label] |
| 19 | newar.append(data\_) |
| 20 | label\_array.append(label) |

Tahap selanjutnya yang dilakukan adalah menyimpan data kedalam bentuk frame dan juga melakukan normalisasi terhadap dataset yang digunakan untuk mempermudah proses perhitungan dengan metode *K-Nearest Neighbors.*

**Tabel 4. 8 Normalisasi Dataset**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | data = newar |
| 2 | df = pd.DataFrame(data) |
| 3 | min\_max\_scaler = preprocessing.MinMaxScaler() |
| 4 | np\_scaled = min\_max\_scaler.fit\_transform(df) |
| 5 | df\_normalized = pd.DataFrame(np\_scaled) |

### **Proses Perhitungan Jarak Eucledian**

Proses perhitungan jarak *Eucledian* dilakukan untuk mengetahui tingkat kemiripan dari data uji terhadap data latih. Jika jarak yang dihasilkan dari perhitungan data uji ini semakin kecil maka tingkat kemiripan terhadap data latih semakin besar. Sebaliknya, jika jarak yang dihasilkan oleh data uji terhadap data latih semakin besar maka tingkap kemiripan data uji terhadap data latih itu semakin kecil. Pada perhitungan jarak *Eucledian* ini, semua jarak dari data latih terhadap data uji akan dihitung. Perhitungan ini dilakukan dengan mencari selisih dari data – data yang sama pada setiap *packet* data.

**Tabel 4. 9 Perhitungan Jarak Eucledian**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | def euclideanDistance(instance1, instance2, length): |
| 2 | distance = 0 |
| 3 | for x in range(length): |
| 4 | distance += pow((instance1[x] - instance2[x]), 2) |
| 5 | return math.sqrt(distance) |

### **Proses Perhitungan Klasifikasi dengan Metode *K-Nearest Neighbor***

Pada tahap ini, dilakukan proses klasifikasi pada dataset NSL-KDD dengan menggunakan metode *K-Nearest Neighbor* (KNN). Fitur yang digunakan pada tahap ini berjumlah 21 fitur dimana fitur tersebut dipilih berdasarkan urutan nilai *Entropy* pada fitur – fitur yang ada pada dataset NSL-KDD.

Perhitungan klasifikasi data uji terhadap data latih pada tahap ini dilakukan dengan mencari jumlah k yang ditentukan pada proses utama. Pengelompokkan yang dilakukan pada tahap ini ada 2 yaitu data dideteksi sebagai data normal atau data serangan. Pada program data normal diberi label 0 dan data serangan diberi label 1.

**Tabel 4. 10 Fungsi K-Nearest Neighbors**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | def getNeighbors(trainingSet, testInstance, k): |
| 2 | distances = [] |
| 3 | length = len(testInstance)-1 |
| 4 | for x in range(len(trainingSet)): |
| 5 | dist = euclideanDistance(testInstance, trainingSet[x], length) |
| 6 | distances.append((trainingSet[x], dist)) |
| 7 | distances.sort(key=operator.itemgetter(1)) |
| 8 | neighbors = [] |
| 9 | for x in range(k): |
| 10 | neighbors.append(distances[x][0]) |
| 11 | return neighbors |
| 12 |  |
| 13 | def getResponse(neighbors): |
| 14 | classVotes = {} |
| 15 | for x in range(len(neighbors)): |
| 16 | response = neighbors[x][-1] |
| 17 | if response in classVotes: |
| 18 | classVotes[response] += 1 |
| 19 | else: |
| 20 | classVotes[response] = 1 |
| 21 | sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True) |
| 22 | return sortedVotes[0][0] |

### **Proses Pengujian dengan *K-Fold Cross Validation***

Pada tahap ini, dilakukan pengujian terhadap dataset yang digunakan. Dataset dibagi menjadi sejumlah bagian untuk memastikan seluruh dataset yang digunakan mendapatkan kesempatan menjadi data uji. Pengujian ini menggunakan metode *K-Fold Cross Validation.*

**Tabel 4. 11 Pengujian dengan K-Fold Cross Validation**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | kf = KFold(n\_splits=4) |
| 2 | for train, test in kf.split(df\_normalized): |
| 3 | accuracy = 0 |
| 4 | train\_data = np.array(df\_normalized)[train] |
| 5 | test\_data = np.array(df\_normalized)[test] |

1. **Proses Perhitungan Akurasi**

Pada tahap proses pengujian metode yang digunakan adalah *K-Fold Cross Validation.* Metode *K-Fold Cross Validation* bekerja dengan cara membagi dataset kedalam sejumlah bagian kemudian pada setiap bagian tersebut tedapat partisi dimana partisi tersebut bagi menjadi data uji dan data latih. Pada penelitian ini, peneliti membagi dataset menjadi 4 bagian dengan jumlah data yang sama. Dimana masing – masing bagian tersebut terdapat 4 buah partisi, kemudian 3 partisi digunakan sebagai data latih dan 1 partisi digunakan sebagai data uji sehingga semua data dapat berperan sebagai data uji dan data latih.

Pada setiap bagian akan dihitung akurasi yang dihasilkan, kemudian akurasi dari semua bagian tersebut dijumlahkan dan selanjutnya dibagi 4 untuk mendapatkan rata – rata akurasi yang dihasilkan oleh program.

**Tabel 4. 12 Perhitungan Akurasi**

|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | def getAccuracy(test\_data, predictions): |
| 2 | correct = 0 |
| 3 | for x in range(len(test\_data)): |
| 4 | if test\_data[x][-1] == predictions[x]: |
| 5 | correct += 1 |
| 6 | return (correct/float(len(test\_data))) \* 100.0 |

1. **Main Program**

Pada main program ditampilkan sintaks yang digunakan dalam pengujian sistem yaitu dengan menggunakan metode *K-Fold Cross Validation.* Output yang ditampilkan oleh main program adalah hasil dari klasifikasi yang dilakukan dengan menggunakan metode *K-Nearest Neighbors.* Dan juga akurasi pada setiap bagian dari pengujian yang dilakukan ditampilkan pada output ini. Dan yang terakhir output dari program ini menampilkan rata – rata akurasi yang dihasilkan oleh program.

**Tabel 4. 13 Main Program**

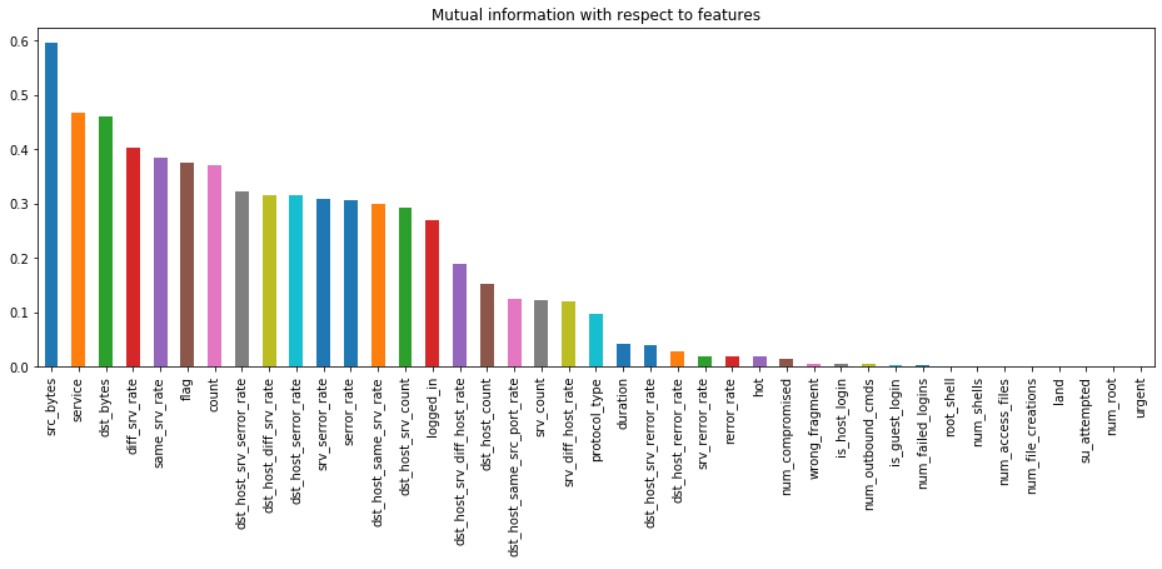
|  |  |
| --- | --- |
| **Baris** | **Kode** |
| 1 | kf = KFold(n\_splits=4) |
| 2 | k = 5 |
| 3 | predictions=[] |
| 4 | sum =0 |
| 5 | sama = '' |
| 6 | for train, test in kf.split(df\_normalized): |
| 7 | accuracy = 0 |
| 8 | train\_data = np.array(df\_normalized)[train] |
| 9 | test\_data = np.array(df\_normalized)[test] |
| 10 | a = 1 |
| 11 | total\_sama = 0 |
| 12 | for x in range(len(test\_data)): |
| 13 | neighbors = getNeighbors(train\_data, test\_data[x], k) |
| 14 | result = getResponse(neighbors) |
| 15 | predictions.append(result) |
| 16 | if( float(result)==float(test\_data[x][-1])): |
| 17 | sama = "sama" |
| 18 | total\_sama=total\_sama+1 |
| 19 | else: |
| 20 | sama="beda" |
| 21 | print((str(a) +'> predicted=' + repr(result) + ', actual=' + repr(test\_data[x][-1])) + ' => ' + sama) |
| 22 | a=a+1 |
| 23 | accuracy = float(float(total\_sama)/float(len(test\_data)))\*100 |
| 24 | sum = sum + accuracy |
| 25 | print (str(total\_sama)+' / '+str(len(test\_data))) |
| 26 | print ("accuracy") |
| 27 | print (accuracy) |
| 28 |  |
| 29 | average = sum/4 |
| 30 | print ("average") |
| 31 | print (average) |
| 32 | print ("======") |

## **Pengujian Sistem**

Pada tahap ini program diuji secara keseluruhan sebagai sistem yang lengkap untuk memastikan bahwa persyaratan perangkat lunak telah terpenuhi dan sesuai dengan kebutuhan serta tujuan awal dari penelitian ini. Skenario dari pengujian ini yaitu yang pertama akan dilakukan pemilahan atau penseleksian fitur dengan menggunakan metode Information Gain. Penseleksian fitur ini digunakan untuk mengurutkan fitur yang paling berpengaruh dalam dataset yang digunakan. Setelah mengurutkan fitur dari yang terbaik ke yang terendah maka selanjutnya akan dipilih 9 fitur yang akan digunakan untuk pengklasifikasian anomali jaringan dengan menggunakan metode KNN (K-Nearest Neighbor). Program akan mengklasifikasikan data yang dibagi menjadi data uji terhadap data latih yang digunakan sebagai patokan dalam pengujian data uji tersebut. Kemudian akurasi dari keberhasilan pendeteksian akan dihitung untuk mengetahui seberapa besar akurasi yang dapat dihasilkan oleh program. Pada penelitian ini akurasi diperoleh dengan membandingkan jumlah data yang berhasil dengan benar diklasifikasikan kedalam 2 buah kelas normal dan kelas serangan DDoS.

### **Seleksi Fitur dengan *Information Gain***

Tahap pertama yang dilakukan pada pengujian sistem ini adalah mencari fitur yang paling relevan terhadap dataset yang digunakan yaitu dataset NSL-KDD. Jumlah fitur yang terdapat dalam dataset NSL-KDD ini adalah sebanyak 41 fitur. Berikut adalah diagram dan tabel dari *Entropy* setiap fitur pada dataset NSL-KDD.



**Gambar 4. 1 Grafik Information Gain**

**Tabel 4. 14 Entropy Setiap Fitur Dataset**

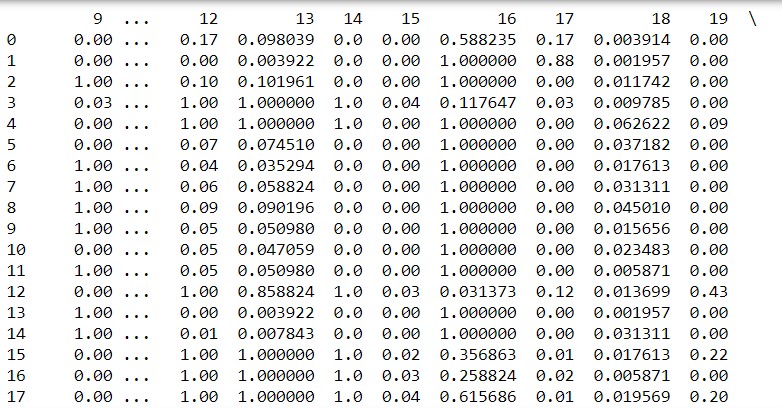
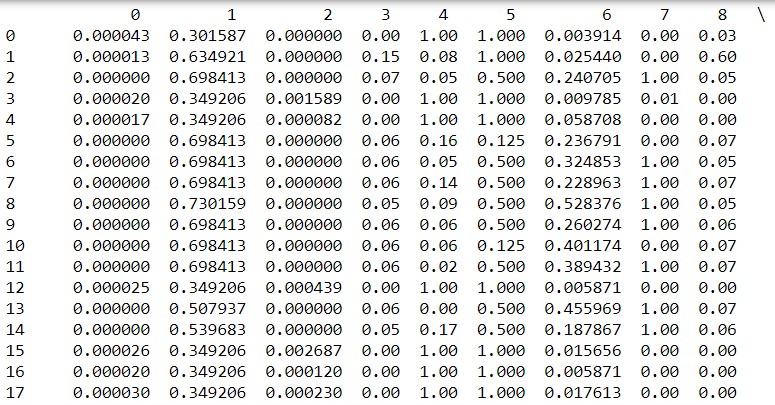
|  |  |
| --- | --- |
| **Nama Fitur** | ***Entropy*** |
| src\_bytes | 0.594335 |
| service | 0.467795 |
| dst\_bytes | 0.460298 |
| diff\_srv\_rate | 0.403451 |
| same\_srv\_rate | 0.383803 |
| flag | 0.375944 |
| count | 0.370305 |
| dst\_host\_srv\_serror\_rate | 0.322255 |
| dst\_host\_diff\_srv\_rate | 0.314277 |
| dst\_host\_serror\_rate | 0.314159 |
| srv\_serror\_rate | 0.308831 |
| serror\_rate | 0.306131 |
| dst\_host\_same\_srv\_rate | 0.300153 |
| dst\_host\_srv\_count | 0.293089 |
| logged\_in | 0.269388 |
| dst\_host\_srv\_diff\_host\_rate | 0.188294 |
| dst\_host\_count | 0.152122 |
| dst\_host\_same\_src\_port\_rate | 0.124786 |
| srv\_count | 0.122932 |
| srv\_diff\_host\_rate | 0.120552 |
| protocol\_type | 0.098174 |
| duration | 0.04175 |
| dst\_host\_srv\_rerror\_rate | 0.0388 |
| dst\_host\_rerror\_rate | 0.028178 |
| srv\_rerror\_rate | 0.019175 |
| rerror\_rate | 0.018893 |
| hot | 0.01811 |
| num\_compromised | 0.01505 |
| wrong\_fragment | 0.005984 |
| is\_host\_login | 0.004521 |
| num\_outbound\_cmds | 0.004423 |
| is\_guest\_login | 0.002776 |
| num\_failed\_logins | 0.002715 |
| root\_shell | 0.001782 |
| num\_shells | 0.00166 |
| num\_access\_files | 0.001625 |
| num\_file\_creations | 0.001348 |
| land | 0 |
| su\_attempted | 0 |
| num\_root | 0 |
| urgent | 0 |

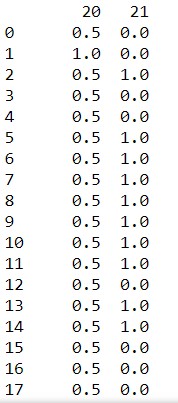
Pada Table 4.14 dapat dilihat besar *Entropy* yang dihasilkan oleh setiap fitur dataset. Dari hasil tersebut digunakan 21 fitur untuk proses klasifikasi dengan menggunakan metode *K-Nearest Neighbor.* Adapun fitur yang digunakan dalam klasifikasi serangan *Distributed Denial of Service (DDoS)* ini adalah :

1. src\_bytes
2. service
3. dst\_bytes
4. diff\_srv\_rate
5. same\_srv\_rate
6. flag
7. count
8. dst\_host\_srv\_serror\_rate
9. dst\_host\_diff\_srv\_rate
10. dst\_host\_serror\_rate
11. srv\_serror\_rate
12. serror\_rate
13. dst\_host\_same\_srv\_rate
14. dst\_host\_srv\_count
15. logged\_in
16. dst\_host\_srv\_diff\_host\_rate
17. dst\_host\_count
18. dst\_host\_same\_src\_port\_rate
19. srv\_count
20. srv\_diff\_host\_rate
21. protocol\_type

### **Normalisasi Dataset**

Pada tahap ini dilakukan normalisasi terhadap dataset untuk memudahkan metode dalam pengolahan data selanjutnya. Hasil dari normalisasi terhadap dataset adalah sebagai berikut :





**Gambar 4. 2 Normalisasi Dataset**

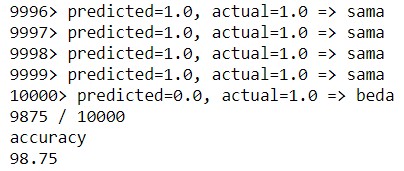
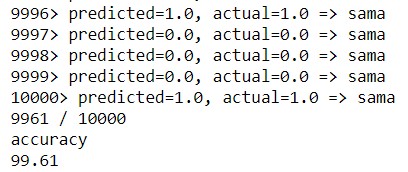
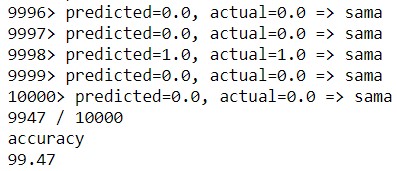
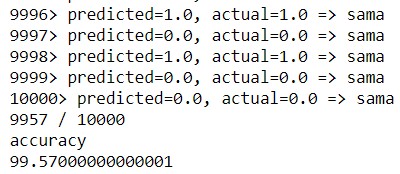
### **Klasifikasi Serangan DDoS dengan *K-Nearest Neighbor***

Pada tahap klasifikasi serangan *Distributed Denial of Service* (DDoS) mengunakan metode *K-Nearest Neighbor* ini, jumlah k yang digunakan oleh peneliti yaitu k = 1, k = 2, k = 3, k = 4, dan k = 5. Sebelum dilakukan klasifikasi terhadap dataset, program terlebih dahulu menghitung jarang *Eucledian* dari masing – masing *packet* data. Jumlah dataset yang digunakan adalah 40000 dataset yang terdiri dari 20000 data normal dan 20000 data serangan. Dimana data serangan yang diprediksi oleh sistem adalah data serang *Denial of Service* (DoS) yaitu *back, land, neptune, ping of death* (POD), dan *smurf.*

### **Pengujian Sistem dengan K-Fold Cross Validation**

Pengujian sistem menggunakan metode *K-Fold Cross Validation,* dimana dataset dibagi menjadi 4 bagian dan masing – masing bagian memiliki 4 partisi yang terdiri dari 3 partisi data latih dan 1 partisi data uji. Pengujian dengan *K-Fold Cross Validation* ini dilakukan untuk melihat seberapa akurat dataset yang digunakan.

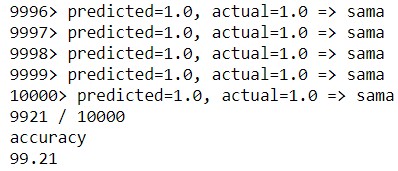
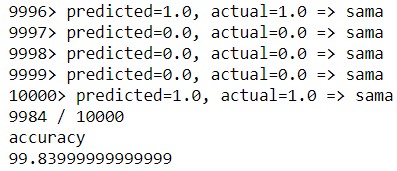
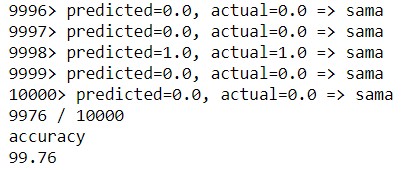
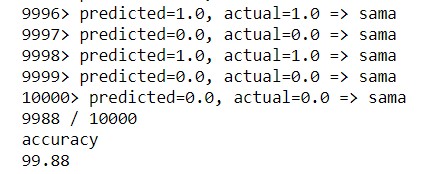
Berikut adalah hasil pengujian dataset dengan k = 1 menggunakan metode *K-Fold Cross Validation* terhadap dataset NSL-KDD dengan jumlah fitur yang digunakan adalah sebanyak 21 fitur dan 41 fitur.



**Gambar 4. 3 Pengujian 21 Fitur dengan k = 1**

Dapat dilihat pada Gambar 4.3, pengujian dilakukan dengan membagi dataset menjadi 4 bagian. Kemudian dataset tersebut dibagi kedalam 4 partisi dimana partisi tersebut dibagi menjadi 3 partisi untuk data latih dan 1 data uji. Pada setiap partisi data uji yang digunakan berjumlah 10000 data, k yang digunakan dalam pengklasifikasian pada Gambar 4.3 adalah k = 3 dan jumlah fitur yang digunakan adalah sebanyak 21 fitur. Dapat dilihat pulan masing – masing dari partisi tersbut memeliki akurasi sebagai berikut :

1. Partisi pertama menghasilkan akurasi sebesar 99,57%.
2. Partisi pertama menghasilkan akurasi sebesar 99,47%.
3. Partisi pertama menghasilkan akurasi sebesar 99,61%.
4. Partisi pertama menghasilkan akurasi sebesar 98,75%.



**Gambar 4. 4 Pengujian 41 Fitur dengan k = 1**

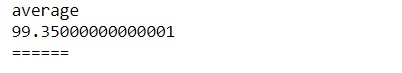
Dapat dilihat pada Gambar 4.3, pengujian dilakukan dengan membagi dataset menjadi 4 bagian. Kemudian dataset tersebut dibagi kedalam 4 partisi dimana partisi tersebut dibagi menjadi 3 partisi untuk data latih dan 1 data uji. Pada setiap partisi data uji yang digunakan berjumlah 10000 data, k yang digunakan dalam pengklasifikasian pada Gambar 4.3 adalah k = 3 dan jumlah fitur yang digunakan adalah sebanyak 21 fitur. Dapat dilihat pulan masing – masing dari partisi tersbut memeliki akurasi sebagai berikut :

1. Partisi pertama menghasilkan akurasi sebesar 99,88%.
2. Partisi pertama menghasilkan akurasi sebesar 99,76%.
3. Partisi pertama menghasilkan akurasi sebesar 99,83999999999%.
4. Partisi pertama menghasilkan akurasi sebesar 99,21%.

### **Pengukuran akurasi**

Akurasi yang dihasilkan dari program dengan k = 1, k = 2, k = 3, k = 4, dan k = 5 adalah sebagi berikut :

* 1. Akurasi k = 1 dengan 21 fitur



**Gambar 4. 5 Akurasi k = 1 Pada 21 Fitur**

* 1. Akurasi k = 2 dengan 21 fitur



**Gambar 4. 6 Akurasi k = 2 Pada 21 Fitur**

* 1. Akurasi k = 3dengan 21 fitur



**Gambar 4. 7 Akurasi k = 3 Pada 21 Fitur**

* 1. Akurasi k = 4 dengan 21 fitur



**Gambar 4. 8 Akurasi k = 4 Pada 21 Fitur**

* 1. Akurasi k = 5 dengan 21 fitur



**Gambar 4. 9 Akurasi k = 5 Pada 21 Fitur**

* 1. Akurasi k = 1 dengan 41 fitur



**Gambar 4. 10 Akurasi k = 1 Pada 41 Fitur**

* 1. Akurasi k = 2 dengan 41 fitur



**Gambar 4. 11 Akurasi k = 2 Pada 41 Fitur**

* 1. Akurasi k = 3 dengan 41 fitur



**Gambar 4. 12 Akurasi k = 3 Pada 41 Fitur**

* 1. Akurasi k = 4 dengan 41 fitur



**Gambar 4. 13 Akurasi k = 4 Pada 41 Fitur**

* 1. Akurasi k = 5 dengan 41 fitur



Gambar 4. 14 Akurasi k = 5 Pada 41 Fitur

## **Analisis**

Pada bagian analisis program, dilakukan perbandingan terhadap jumlah fitur yang digunakan yaitu perbandingan antar 21 fitur dengan 41 fitur. Nilai k yang digunakan pada masing – masing fitur tersebut berjumlah 5 yaitu k = 1, k = 2, k = 3, k = 4, dan k = 5.

Pada Gambar adalah grafik analisis perbandingan terhadap metode *K-Nearest Neighbor* dengan menggunakan 21 fitur dan 41 fitur.

**Gambar 4. 15 Hasil Klasifikasi**

Pengujian dilakukan dengan menggunakan 2 jumlah fitur yang berbeda yaitu dengan 41 fitur dan 21 fitur. Jumlah fitur 21 di dapatkan dari seleksi fitur dengan mengurutkan nilai *entropy* dari semua fitur yang ada pada dataset. Jumlah k yang digunakan pada saat pengujian adalah k = 1, k = 2, k = 3, k = 4, dan k = 5. Pada Gambar 4.15 dapat dilihat bahwa hasil pengujian tertinggi dihasilkan oleh nilai k = 1 pada kedua data. Pada data yang menggunakan 41 fitur hasil rata – rata akurasi tertinggi yaitu 99.67249999999999% sedangkan rata – rata akurasi tertinggi pada data yang menggunakan 21 fitur adalah 99.35000000000001%. Dapat dilihat pula bahwa akurasi yang dihasilkan cenderung menurun jika nilai k semakin besar.

Dari hasil pengujian tersebut dapat disimpulkan bahwa nilai k akan mempengaruhi hasil akurasi dari sistem. Nilai k optimal dari kedua data tersebut adalah k = 1. Jika k semakin besar maka akurasi yang dihasilkan semakin menurun. Selain itu, data dengan 41 fitur menghasilkan akurasi yang lebih tinggi dari pada data yang dengan 21 fitur. Namun adapun kelemahan dalam menjalankan program ini adalah lamanya waktu yang diperlukan untuk menjalankan program. Dari hasil pengamatan data dengan fitur berjumlah 21 lebih cepat mendapatkan hasil akurasi dibandingkan dengan data yang memiliki 41 fitur. Selisih waktu untuk menjalankan kedua program tersebut kurang lebih 4 jam 30 menit.

# **BAB V**

# **PENUTUP**

## **Kesimpulan**

Adapun kesimpulan yang diperoleh dari implementasi algoritma *K-Nearest Neighbor* (KNN) untuk klasifikasi serangan *Distributed Denial of Service* (DDoS) dengan menggunakan dataset yang berasal dari NSL-KDD.

1. Setiap fitur memiliki nilai *Entropy* yang berbeda – beda sehingga bisa diketahui fitur – fitur apa saja yang memiliki nilai *Entropy* yang lebih tinggi. Seleksi fitur pada sistem menggunakan metode *Information Gain.*
2. Akurasi yang dihasilkan oleh sistem dengan menggunakan metode *K-Nearest Neighbor* (KNN) dengan 41 fitur rata – rata lebih tinggi dibandingkan dengan akurasi yang dihasilkan oleh metode *K-Nearest Neighbor* dengan seleksi fitur menggunakan metode *Information Gain*.
3. Waktu yang diperlukan untuk menjalankan program dengan menggunakan metode *K-Nearest Neighbor* dengan seleksi fitur *Information Gain* relative lebih cepat dibandingkan tidak menggunakan seleksi fitur. Selisih waktu yang diperlukan untuk menjalankan program dengan kedua fitur tersebut -+ 4 jam 30 menit.

## **Saran**

Adapun saran yang dapat diberikan untuk pengembangan sistem lebih lanjut adalah sebagai berikut :

1. Dalam sistem klasifikasi ini fitur yang digunakan hanya diseleksi menggunakan metode *Information Gain* sehingga perlu percobaan seleksi fitur dengan menggunakan metode lain juga.
2. Dapat menggunakan metode atau algoritma lain selain *K-Nearest Neighbor* (KNN) atau kombinasikan beberapa algoritma untuk menghasilkan klasifikasi data yang lebih baik.
3. Dapat menggunakan metode atau algoritma lain selain *K-Nearest Neighbor* untuk membangdingkan waktu yang perlukan dalam menjalankan program.

# **DAFTAR PUSTAKA**

Adrian, R., & Isnianto, H. N. (2016). {Analisa} {Pengaruh} {Variasi} {Serangan} {Ddos} {Pada} {Performa} {Router}. *{SEMINAR} {NASIONAL} {TEKNOLOGI} {TERAPAN} ({SNTT} 2016), {Sekolah} {Vokasi} {Universitas} {Gadjah} {Mada}*, *6*(November), 1257–1259. https://www.researchgate.net/profile/Ronald\_Adrian/publication/311219100\_ANALISA\_PENGARUH\_VARIASI\_SERANGAN\_DDOS\_PADA\_PERFORMA\_ROUTER/links/583f7f0608ae2d217557e6cf.pdf

Ayub, M. (n.d.). *Proses Data Mining dalam Sistem Pembelajaran Berbantuan Komputer*. 21–30.

Cahyaningtyas, Sukarno, N. (2019). *Deteksi Serangan Denial of Service (DoS) menggunakan Algoritma Probabilistic Neural Network (PNN)*. *6*(2), 8808–8818.

Gunawan, G. B., Sukarno, P., & Putrada, A. G. (2018). Pendeteksian Serangan Denial of Service ( DoS ) pada Perangkat Smartlock Berbasis Wifi Menggunakan SNORT IDS. *Issn*, *5*(3), 7875–7884.

Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2015). Botnet in DDoS Attacks: Trends and Challenges. *IEEE Communications Surveys and Tutorials*, *17*(4), 2242–2270. https://doi.org/10.1109/COMST.2015.2457491

Imam, R. M., Sukarno, P., & Nugroho, M. A. (2019). *Deteksi Anomali Jaringan Menggunakan Hybrid Algorithm*. *6*(2), 8766–8787.

Kurniabudi, K., Harris, A., & Rahim, A. (2020). Seleksi Fitur Dengan Information Gain Untuk Meningkatkan Deteksi Serangan DDoS menggunakan Random Forest. *Techno.Com*, *19*(1), 56–66. https://doi.org/10.33633/tc.v19i1.2860

Liantoni, F. (2016). Klasifikasi Daun Dengan Perbaikan Fitur Citra Menggunakan Metode K-Nearest Neighbor. *Jurnal ULTIMATICS*, *7*(2), 98–104. https://doi.org/10.31937/ti.v7i2.356

Mardi, Y. (2017). Data Mining : Klasifikasi Menggunakan Algoritma C4.5. *Jurnal Edik Informatika*, *2*(2), 213–219.

Mustakim, G. O. (2016). Algoritma K-Nearest Neighbor Classification. *Jurnal Sains, Teknologi Dan Industri*, *13*(2), 195–202. http://ejournal.uin-suska.ac.id/index.php/sitekin

Mutiara, I. dan A. (2015). Penerapan K-Optimal Pada Algoritma Knn Untuk Prediksi Kelulusan Tepat Waktu Mahasiswa Program Studi Ilmu Komputer Fmipa Unlam Berdasarkan Ip Sampai Dengan Semester 4. *Klik - Kumpulan Jurnal Ilmu Komputer*, *2*(2), 159–173.

Prakash, A., & Priyadarshini, R. (2018). An Intelligent Software defined Network Controller for preventing Distributed Denial of Service Attack. *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, *Icicct*, 585–589. https://doi.org/10.1109/ICICCT.2018.8473340

Pratama, D. (2019). *Serangan Ddos Pada Software-Defined Network*. https://doi.org/10.31227/osf.io/a86cr

Ramadhan, I., Sukarno, P., & Nugroho, M. A. (2019). *Analisis Perbandingan Algoritma K-Nearest Neighbor dan Decision Tree Dalam Mendeteksi Distributed Denial of Service*. *6*(2), 8548–8558.

Rouf, A. (2012). *Pengujian Perangkat Lunak Dengan Menggunakan Metode White Box dan Back Box*. *vol 8 no1*, 1–7. http://www.ejournal.himsya.ac.id/index.php/HIMSYATECH/article/view/28/27

Sasmito, G. W. (2017). Penerapan Metode Waterfall Pada Desain Sistem Informasi Geografis Industri Kabupaten Tegal. *Jurnal Informatika:Jurnal Pengembangan IT (JPIT)*, *2*(1), 6–12.

Satmoko, D. B., Sukarno, P., & Jadied, E. M. (2018). *Peningkatan Akurasi Pendeteksian Serangan DDoS Menggunakan Multiclassifier Ensemble Learning dan Chi-Square Pendahuluan Studi Terkait*. *5*(3), 7977–7985.

Suyanto. (2017). *Data Mining untuk Klasifikasi dan Klasterisasi Data*. Informatika Bandung.

Syafitri Hidayatul AA, Yuita Arum S, A. A. (2018). Seleksi Fitur Information Gain untuk Klasifikasi Penyakit Jantung Menggunakan Kombinasi Metode K-Nearest Neighbor dan Naïve Bayes. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(9), 2546–2554.

# **LAMPIRAN**

1. *Information Gain*

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score  from sklearn.feature\_selection import VarianceThreshold, mutual\_info\_classif, mutual\_info\_regression  from sklearn.feature\_selection import SelectKBest, SelectPercentile  import sys  import sklearn  import io  import random  import time  from sklearn.preprocessing import LabelEncoder,OneHotEncoder  from sklearn import preprocessing  data = pd.read\_csv('data\_ig.csv')  #data.head(5)  # konversi label, 0 untuk normal dan 1 untuk serangan  label\_data = data['label']  # ubah kolom label  new\_label\_data = label\_data.replace({'normal' : 0, 'back' : 1, 'land' : 1, 'pod' : 1, 'neptune' : 1, 'smurf' : 1})  # tambahkan kolom label baru kedalam dataet  data['label'] = new\_label\_data  #print(data)  # tandai fitur dan label dataset  fitur = data.drop('label', axis = 1)  target = data['label']  fitur.shape, target.shape  # bagi dataset menjadi data training dan data testing, 80% untuk training dan 20% untuk testing  X\_train, X\_test, y\_train, y\_test = train\_test\_split(fitur, target, test\_size = 0.2, random\_state = 0)  # tentukan threshold untuk memunculkan target yang ingin ditampilkan  constant\_filter = VarianceThreshold(threshold=0)  constant\_filter.fit(X\_train)  X\_train\_filter = constant\_filter.transform(X\_train)  X\_test\_filter = constant\_filter.transform(X\_test)  X\_train\_T = X\_train\_filter.T  X\_test\_T = X\_test\_filter.T  X\_train\_T = pd.DataFrame(X\_train\_T)  X\_test\_T = pd.DataFrame(X\_test\_T)  X\_train\_T.duplicated().sum()  duplicated\_features = X\_train\_T.duplicated()  mi = mutual\_info\_classif(X\_train, y\_train)  len(mi)  mi = pd.Series(mi)  mi.index = X\_train.columns  mi.sort\_values(ascending=False, inplace = True)  mi  plt.title('Mutual information with respect to features')  mi.plot.bar(figsize = (16,5))  plt.show() |

1. *K-Nearest Neighbor* tanpa seleksi fitur

|  |
| --- |
| import csv, random  import pandas as pd  import numpy as np  import math  import operator  from sklearn import preprocessing  import nltk  from sklearn.model\_selection import train\_test\_split,KFold  def euclideanDistance(instance1, instance2, length):  distance = 0  for x in range(length):  distance += pow((instance1[x] - instance2[x]), 2)  return math.sqrt(distance)  def getNeighbors(trainingSet, testInstance, k):  distances = []  length = len(testInstance)-1  for x in range(len(trainingSet)):  dist = euclideanDistance(testInstance, trainingSet[x], length)  distances.append((trainingSet[x], dist))  distances.sort(key=operator.itemgetter(1))  neighbors = []  for x in range(k):  neighbors.append(distances[x][0])  return neighbors  def getResponse(neighbors):  classVotes = {}  for x in range(len(neighbors)):  response = neighbors[x][-1]  if response in classVotes:  classVotes[response] += 1  else:  classVotes[response] = 1  sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True)  return sortedVotes[0][0]  def getAccuracy(test\_data, predictions):  correct = 0  for x in range(len(test\_data)):  if test\_data[x][-1] == predictions[x]:  correct += 1  return (correct/float(len(test\_data))) \* 100.0  with open('data40.csv') as csvfile :  readCSV = csv.reader(csvfile)  df1=pd.read\_csv("data40.csv")  # print(df1)  newar=[];  a = 0;  label =0  label\_array = []  for row in readCSV:  if len(row)>41:  if row[41]=='normal':  label = 0  if row[41]=='back':  label = 1  if row[41]=='land':  label = 1  if row[41]=='neptune':  label = 1  if row[41]=='pod':  label = 1  if row[41]=='smurf':  label = 1  data\_ = [float(row[0]), float(row[1]), float(row[2]), float(row[3]),float(row[4]),float(row[5]),float(row[6]),float(row[7]), float(row[8]), float(row[9]), float(row[10]), float(row[11]), float(row[12]), float(row[13]), float(row[14]), float(row[15]), float(row[16]), float(row[17]), float(row[18]), float(row[19]), float(row[20]), float(row[21]), float(row[22]), float(row[23]), float(row[24]), float(row[25]), float(row[26]), float(row[27]), float(row[28]), float(row[29]), float(row[30]), float(row[31]), float(row[32]), float(row[33]), float(row[34]), float(row[35]), float(row[36]), float(row[37]), float(row[38]), float(row[39]), float(row[40]), label]  newar.append(data\_)  label\_array.append(label)  data = newar  df = pd.DataFrame(data)  min\_max\_scaler = preprocessing.MinMaxScaler()  np\_scaled = min\_max\_scaler.fit\_transform(df)  df\_normalized = pd.DataFrame(np\_scaled)  #print (df\_normalized)  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(df\_normalized, label\_array, test\_size=0.2)  kf = KFold(n\_splits=4)  k = 5  predictions=[]  sum =0  sama = ''  for train, test in kf.split(df\_normalized):  accuracy = 0  train\_data = np.array(df\_normalized)[train]  #print("=======")  #print (train\_data)  #print("======")  test\_data = np.array(df\_normalized)[test]  #print("data uji")  #print(test\_data)  a = 1  total\_sama = 0  for x in range(len(test\_data)):  neighbors = getNeighbors(train\_data, test\_data[x], k)  result = getResponse(neighbors)  predictions.append(result)  if( float(result)==float(test\_data[x][-1])):  sama = "sama"  total\_sama=total\_sama+1  else:  sama="beda"  print((str(a) +'> predicted=' + repr(result) + ', actual=' + repr(test\_data[x][-1])) + ' => ' + sama)  a=a+1  accuracy = float(float(total\_sama)/float(len(test\_data)))\*100    sum = sum + accuracy  print (str(total\_sama)+' / '+str(len(test\_data)))  print ("accuracy")  print (accuracy)  average = sum/4  print ("average")  print (average)  print ("======") |

1. *K-Nearest Neighbor* dengan seleksi fitur *Information Gain*

|  |
| --- |
| import csv, random  import pandas as pd  import numpy as np  import math  import operator  from sklearn import preprocessing  import nltk  from sklearn.model\_selection import train\_test\_split,KFold  def euclideanDistance(instance1, instance2, length):  distance = 0  for x in range(length):  distance += pow((instance1[x] - instance2[x]), 2)  return math.sqrt(distance)  def getNeighbors(trainingSet, testInstance, k):  distances = []  length = len(testInstance)-1  for x in range(len(trainingSet)):  dist = euclideanDistance(testInstance, trainingSet[x], length)  distances.append((trainingSet[x], dist))  distances.sort(key=operator.itemgetter(1))  neighbors = []  for x in range(k):  neighbors.append(distances[x][0])  return neighbors  def getResponse(neighbors):  classVotes = {}  for x in range(len(neighbors)):  response = neighbors[x][-1]  if response in classVotes:  classVotes[response] += 1  else:  classVotes[response] = 1  sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), reverse=True)  return sortedVotes[0][0]  def getAccuracy(test\_data, predictions):  correct = 0  for x in range(len(test\_data)):  if test\_data[x][-1] == predictions[x]:  correct += 1  return (correct/float(len(test\_data))) \* 100.0  with open('21fitur.csv') as csvfile :  readCSV = csv.reader(csvfile)  df1=pd.read\_csv("21fitur.csv")  # print(df1)  newar=[];  a = 0;  label =0  label\_array = []  for row in readCSV:  if len(row)>21:  if row[21]=='normal':  label = 0  if row[21]=='back':  label = 1  if row[21]=='land':  label = 1  if row[21]=='neptune':  label = 1  if row[21]=='pod':  label = 1  if row[21]=='smurf':  label = 1  data\_ = [float(row[0]), float(row[1]), float(row[2]), float(row[3]),float(row[4]),float(row[5]),float(row[6]),float(row[7]), float(row[8]), float(row[9]), float(row[10]), float(row[11]), float(row[12]), float(row[13]), float(row[14]), float(row[15]), float(row[16]), float(row[17]), float(row[18]), float(row[19]), float(row[20]), label]  newar.append(data\_)  label\_array.append(label)  data = newar  df = pd.DataFrame(data)  # print(df)  min\_max\_scaler = preprocessing.MinMaxScaler()  np\_scaled = min\_max\_scaler.fit\_transform(df)  df\_normalized = pd.DataFrame(np\_scaled)  #print (df\_normalized)  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(df\_normalized, label\_array, test\_size=0.2)  kf = KFold(n\_splits=4)  k = 5  predictions=[]  sum =0  sama = ''  for train, test in kf.split(df\_normalized):  accuracy = 0  train\_data = np.array(df\_normalized)[train]  #print("=======")  #print (train\_data)  #print("======")  test\_data = np.array(df\_normalized)[test]  #print("data uji")  #print(test\_data)  a = 1  total\_sama = 0  for x in range(len(test\_data)):  neighbors = getNeighbors(train\_data, test\_data[x], k)  result = getResponse(neighbors)  predictions.append(result)  if( float(result)==float(test\_data[x][-1])):  sama = "sama"  total\_sama=total\_sama+1  else:  sama="beda"  print((str(a) +'> predicted=' + repr(result) + ', actual=' + repr(test\_data[x][-1])) + ' => ' + sama)  a=a+1  accuracy = float(float(total\_sama)/float(len(test\_data)))\*100  sum = sum + accuracy  print (str(total\_sama)+' / '+str(len(test\_data)))  print ("accuracy")  print (accuracy)  average = sum/4  print ("average")  print (average)  print ("======") |