

SKRIPSI

SISTEM PERINGATAN DINI UNTUK MENDETEKSI SPAM PADA EMAIL MENGGUNAKAN DOMAIN NAME SYSTEM- BASED BLACKHOLE LIST (DNSBL) FILTER DAN SUPPORT VECTOR MACHINE (SVM) FILTER

Diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik Informatika



Disusun Oleh :

NAMA : MUHAMMAD HAFIDZ

NIM : A11.2014.08602

Program Studi : Teknik Informatika-S1

**FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO
SEMARANG
Tahun 2018**

PERSETUJUAN SKRIPSI

Nama : Muhammad Hafidz
NIM : A11.2015.09000
Program Studi : Teknik Informatika-S1
Fakultas : Ilmu Komputer
Judul Tugas Akhir : Sistem Peringatan Dini Untuk Mendeteksi Spam Pada Email Menggunakan *Domain Name System-Based Blackhole List(DNSBL) Filter* Dan *Support Vector Machine (SVM) Filter*

Tugas Akhir ini telah diperiksa dan disetujui,

Semarang, 23 Februari 2019

Menyetujui :

Pembimbing

Mengetahui :

Dekan Fakultas Ilmu Komputer

Fahri Firdausillah S.Kom M.CS

Dr. Abdul Syukur

PENGESAHAN DEWAN PENGUJI

Nama : Muhammad Hafidz
NIM : A11.2015.09000
Program Studi : Teknik Informatika-S1
Fakultas : Ilmu Komputer
Judul Tugas Akhir : Sistem Peringatan Dini Untuk Mendeteksi Spam Pada Email
Menggunakan *Domain Name System-Based Blackhole List (DNSBL) Filter Dan Support Vector Machine (SVM) Filter*

Tugas Akhir ini telah diujikan dan di pertahankan didepan Dewan Penguji pada Sidang tugas akhir tanggal 2 Agustus 2019. Menurut pandangan kami, tugas akhir ini memadai dari segi kualitas maupun kuantitas untuk tujuan penganugerahan gelar Sarjana Komputer (S.Kom)

Semarang, 23 Februari 2019

Dewan Penguji :

Adhitya Nugraha, S.Kom, M.CS

Penguji 1

Abas Setiawan, S.Kom, M.CS

Penguji 2

Slamet Sudaryanto N.,ST,M.Kom

Ketua penguji

PERNYATAAN KEASLIAN TUGAS AKHIR

Sebagai mahasiswa Universitas Dian Nuswantoro, yang bertanda tangan dibawah ini,
saya:

Nama : Muhammad Hafidz

NIM : A11.2015.09000

Menyatakan bahwa karya ilmiah saya yang berjudul :

SISTEM PERINGATAN DINI UNTUK MENDETEKSI SPAM PADA EMAIL MENGGUNAKAN DOMAIN NAME SYSTEM-BASED BLACKHOLE LIST (DNSBL) FILTER DAN SUPPORT VECTOR MACHINE (SVM) FILTER-
merupakan karya asli saya (kecuali cuplikan dan ringkasan yang masing-masing telah saya jelaskan sumbernya dan perangkat pendukung seperti web cam dll). Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, yang disertai dengan bukti-bukti yang cukup, maka saya bersedia untuk dibatalkan gelar saya beserta hak dan kewajiban yang melekat pada gelar tersebut. Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Semarang

Pada tanggal : 23 Februari 2019

Yang Menyatakan

(Muhammad Hafidz)

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai mahasiswa Universitas Dian Nuswantoro, yang bertanda tangan di bawah ini, saya :

Nama : Muhammad Hafidz

NIM :A11.2015.09000

demikian mengembangkan Ilmu Pengetahuan, menyetujui untuk memberikan kepada Universitas Dian Nuswantoro Hak Bebas Royalti Non-Eksklusif (*Non exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul :

SISTEM PERINGATAN DINI UNTUK MENDETEKSI SPAM PADA EMAIL MENGUNAKAN DOMAIN NAME SYSTEM-BASED BLACKHOLE LIST (DNSBL) FILTER DAN SUPPORT VECTOR MACHINE (SVM) FILTER

beserta perangkat yang diperlukan(bila ada). Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Dian Nuswantoro berhak untuk menyimpan, mengcopy ulang (memperbanyak), menggunakan, mengelolanya dalam bentuk pangkalan data (database), mendistribusikannya dan menampilkan/mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Dian Nuswantoro, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini. Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Semarang

Pada tanggal : 23 Februari 2019

Yang Menyatakan

(Muhammad Hafidz)

UCAPAN TERIMA KASIH

Dengan memanjatkan puji syukur kehadiran Allah SWT. Tuhan Yang Maha Pengasih dan Maha Penyayang yang telah melimpahkan segala rahmat, hidayah dan inayah-Nya kepada penulis sehingga laporan tugas akhir dengan judul “SISTEM PERINGATAN DINI UNTUK MENDETEKSI SPAM PADA EMAIL MENGGUNAKAN *DOMAIN NAME SYSTEM-BASED BLACKHOLE LIST (DNSBL) FILTER* DAN *SUPPORT VECTOR MACHINE (SVM) FILTER*” dapat penulis selesaikan sesuai dengan rencana karena dukungan dari berbagai pihak yang tidak ternilai besarnya. Oleh karena itu penulis menyampaikan terimakasih kepada:

1. Kedua orang tua, adik-adik dan keluarga yang telah memberikan motivasi penulis dalam pembuatan laporan Tugas Akhir ini.
2. Bapak Prof. Dr. Ir Edi Noersasongko, M.Kom, selaku Rektor Universitas Dian Nuswantoro Semarang.
3. Bapak Dr. Drs. Abdul Syukur, MM, selaku Dekan Fasilkom Universitas Dian Nuswantoro.
4. Bapak Heru Agus Santoso, Ph.D, selaku Ka.Progdi Teknik informatika –S1.
5. Bapak Fahri Firdausillah S.KOM, M.CS selaku pembimbing tugas akhir yang sangat baik, sabar dalam membimbing penulis saat penulis mengalami banyak kesulitan dan sekaligus ketua PSI Udinus atas izin dan bantuannya selama penelitian yang dilakukan penulis.
6. Teman-teman DOSCOMyang telah banyak memberikan doa, semangat, dan bantuan kepada penulis.
7. Kepada semua pihak yang namanya tidak dapat di sebutkan satu persatu.

Semoga Tuhan Yang Maha Esa memberikan balasan yang lebih besar kepada beliau-beliau, dan pada akhirnya berharap bahwa penulisan laporan Tugas Akhir ini dapat bermanfaat dan berguna sebagaimana fungsinya.

Semarang, 23 Februari 2019

(Muhammad Hafidz)

ABSTRAK

Email merupakan media komunikasi bagi pengguna dan penyedia jasa Internet yang efektif. Perusahaan-perusahaan besar mayoritas menggunakan email sebagai media komunikasi dengan para pelanggannya. Namun tidak semua email yang dikirim dapat sampai ke kotak masuk email para pelanggan. Banyak faktor yang mempengaruhi hal tersebut, diantaranya karena konten yang tidak sesuai kaidah penulisan yang baik, alamat email yang tidak valid, domain pengguna yang terdaftar dalam Blacklist dan sebagainya. Berdasarkan hal tersebut, diperlukan adanya perangkat lunak yang digunakan untuk pengecekan email yang akan dikirim untuk meningkatkan kemungkinan email sampai ke pelanggan. Penelitian ini menghasilkan sebuah perangkat lunak Backend API validator yang dikembangkan menggunakan bahasa pemrograman Java dan Kotlin. Perangkat lunak tersebut menggunakan metode Domain Name System-Based Blackhole List (DNSBL) dan Support Vector Machine (SVM) dalam melakukan validasi email sebelum dikirim. Melalui penelitian ini diharapkan dapat mengurangi prosentase email gagal terkirim akibat email terdeteksi sebagai spam.

Kata kunci : Backend API

Kotlin

Filter spam

Email

xi + 103 halaman; 42 gambar; 31 tabel; 6 lampiran

Daftar acuan: 21 (1986 – 2003)

DAFTAR ISI

PERSETUJUAN SKRIPSI	ii
PENGESAHAN DEWAN PENGUJI	iii
PERNYATAAN KEASLIAN TUGAS AKHIR	iv
PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS	v
UCAPAN TERIMA KASIH	vii
ABSTRAK	ix
DAFTAR ISI	x
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	4
1.5 Manfaat Penelitian	4
BAB II LANDASAN TEORI	5
2.1 Tinjauan Studi	5
2.2 Tinjauan Pustaka	10
2.2.1 Surat Elektronik	10
2.2.2 Mail Server	11
2.2.3 Cara Kerja Email	11
2.2.2.1. IMAP	11
2.2.2.2. POP	11
2.2.2.3. SMTP	12
2.2.4 Email Marketing	13
2.2.5 Bounce Message	14
2.2.6 Kotlin	14
2.2.7 Sistem Peringatan Dini	15
2.2.8 Sistem Penanganan Dini pada Spam Email	16
2.2.9 Client-Server Model	18
2.2.10 World Wide Web	24
2.2.11 HTML	25
2.2.12 Hypertext Transfer Protocol	25
2.2.13 Web Service	26

2.2.14	REST API.....	26
2.2.15	DNS dan Tipe DNS.....	27
2.2.16	Jenis-jenis DNS Record	27
2.2.17	DNSBL	28
2.2.18	Jenis jenis DNSBL.....	29
2.2.19	Cara Kerja Pengecekan Domain berbasis DNSBL.....	30
2.2.20	Web Scraping	30
2.2.21	Machine Learning	30
2.2.22	Text Mining.....	31
2.2.23	TF IDF	33
2.2.24	Chi Square.....	33
2.2.25	Support Vector Machine (SVM)	34
2.2.26	Spam Prevention menggunakan Support Vector Machine (SVM)	35
2.3	Kerangka Pemikiran	36
BAB III METODE PENELITIAN		37
3.1	Jenis Dan Sumber Data.....	37
3.2	Teknik Pengumpulan Data.....	37
3.3	Metode Pengembangan Sistem.....	37
3.3.1.	Sprint.....	38
3.3.2.	Sprint Planning	39
3.3.3.	Daily Scrum	40
3.3.4.	Sprint Review	40
3.3.5.	Sprint Retrospective.....	41
3.4	Metode yang Diusulkan	42
3.5	Metode Evaluasi	44
BAB IV RANCANGAN SISTEM DAN IMPLEMENTASI		46
4.1	Pengantar	46
4.2	Analisa Kebutuhan.....	46
4.2.1	Analisis Kebutuhan Data	46
4.2.2	Analisis Kebutuhan Pengguna	46
4.3	Desain Sistem	47
4.2.1	Use Case Diagram	47
4.2.3	Sequence Diagram	53
4.2.3.1	Add DNSBL	53
4.2.3.2	Delete DNSBL.....	54
4.2.3.3	List DNSBL	55

4.2.3.4	Init DNSBL.....	55
4.2.3.5	Check Domain	56
4.2.3.6	Train Dataset	57
4.2.3.7	Predict Content	57
4.2.3.8	Send Email	58
4.4	Desain Antarmuka.....	58
4.5	Algoritam Klasifikasi.....	62
4.5.1	Mengubah file dataset menjadi memory dataset.....	62
4.5.2	Mengubah dataset menjadi fitur data menggunakan DF.....	65
4.5.3	Feature selection menggunakan Chi Square.	69
4.5.4	Membuat problem.....	75
4.5.5	Melakukan pemodelan menggunakan Support Vector Machine.	76
4.5.6	Melakukan prediksi konten email.....	78
BAB V HASIL DAN PENGUJIAN		81
5.1	Source Code	81
5.1.1	Fungsi checkDomain	81
5.1.2	Fungsi scrapDnsbl.....	82
5.1.3	Train Dataset	84
5.2	Antarmuka	85
5.3	Pengujian Sistem.....	88
BAB VI KESIMPULAN DAN SARAN		92
6.1.	Kesimpulan	92
6.2.	Saran	92
DAFTAR PUSTAKA.....		93

DAFTAR TABEL

Tabel 2.1 Tinjauan Pustaka.....	7
Tabel 2.2 Klasifikasi DNSBL yang digunakan CSAIL (Computer Science and Artificial Intelligence Laboratory)	29
Tabel 2.3 Kerangka Pemikiran	36
Tabel 4. 1 Tabel kebutuhan pengguna.....	47
Tabel 4. 2 Tabel DNSBL	50
Tabel 4. 3 Linear SVM	50
Tabel 4. 4 TfidfFeatureWeighter	51
Tabel 4. 5 Tabel Model.....	51
Tabel 4. 6 Tabel SolverType.....	52
Tabel 4. 7 Tabel AbstractModel	52
Tabel 4. 8 Tabel Maile	53
Tabel 4. 9 Tabel hasil pengubahan file dataset menjadi memory dataset.....	62
Tabel 4. 10 Tabel hasil tokenisasi dan stop word removal pada text	62
Tabel 4. 11 Tabel indeks kategori.....	63
Tabel 4. 12 Tabel Indeks Kata	63
Tabel 4. 13 Tabel hasil pengubahan ke fitur data	65
Tabel 4. 14 Tabel hasil perhitungan Chi Square.....	69
Tabel 5. 1 Tabel hasil white box testing.....	88
Tabel 5. 2 Tabel hasil black box testing	89
Tabel 5. 3 Tabel Confusion Matrix.....	90
Tabel 5. 4 Tabel hasil perhitungan Conusion Matrix	91

DAFTAR GAMBAR

Gambar 2. 1 Jalur pengiriman email.....	12
Gambar 2. 2 Alur sistem peringatan dini.....	15
Gambar 2. 3 Metode Deteksi Spam.....	16
Gambar 2. 4 Analisis Spam.....	19
Gambar 2. 5 Client-Server Model.....	24
Gambar 2. 6 Contoh Syntax HTML.....	25
Gambar 2. 7 Proses pencarian pengetahuan untuk text mining.....	32
Gambar 2. 8 Gambaran klasifikasi menggunakan support vector machine.....	34
Gambar 3.1Kerangka Sistem Peringatan Dini Deteksi Spam Pada Email.....	42
Gambar 4.1Use Case Diagram.....	48
Gambar 4.2 Class Diagram.....	49
Gambar 4.3Sequence Diagram : Add DNSBL.....	53
Gambar 4.4Sequence Diagram : Delete DNSBL.....	54
Gambar 4.5Sequence Diagram : List DNSBL.....	55
Gambar 4.6Sequence Diagram : Init DNSBL.....	55
Gambar 4.7Sequence Diagram : Check DNSBL.....	56
Gambar 4.8 Sequence Diagram : Train Dataset.....	57
Gambar 4.9 Sequence Diagram : Predict Content.....	57
Gambar 4.10Sequence Diagram : Send Email.....	58
Gambar 4.11 Desain Halaman Email Sender.....	58
Gambar 4.12Desain Halaman DNSBL.....	59
Gambar 4.13 Desain Halaman Scrap.....	60
Gambar 4.14Desain Halaman Insert DNSBL.....	60
Gambar 4.15DesainHalaman Delete DNSBL.....	61
Gambar 4.16Desain Halaman Check Domain/IP.....	61
Gambar 5. 1 Gambar potongan kode program Check Domain.....	82
Gambar 5. 2 Gambar potongan kode program Scrap DNSBL.....	83
Gambar 5. 3 Gambar potongan kode training svm.....	84
Gambar 5. 4 Gambar halaman email spam checker.....	85
Gambar 5. 5 Halaman Domain Checker.....	86
Gambar 5. 6 14Halaman DNSBL.....	87
Gambar 5. 7 Halaman Spam Checker.....	88

BAB I

PENDAHULUAN

1.1 Latar Belakang

MTarget.co adalah perusahaan *startup* IT dengan model bisnis *SaaS (Software as a Service)* dari Jakarta. Kontribusi mereka dalam aktivitas pemasaran di perusahaan adalah membangun email marketing automation software berbasis cloud, untuk membantu usaha kecil, menengah hingga perusahaan besar dalam mengembangkan bisnis mereka. (MailTarget, 2018)

Permasalahan yang sering terjadi baik pada perusahaan ataupun agensi *email marketing* seperti MTarget.co adalah tidak semua email yang dikirim dapat sampai ke kotak masuk email para pelanggan. Banyak email yang dikirim terdeteksi sebagai spam, dan akhirnya email tersebut mengalami *Bounce*. *Bounce Message* adalah email otomatis yang memberi tahu pengirim pesan sebelumnya bahwa pesan itu belum terkirim (atau masalah pengiriman lainnya terjadi). ^(Wikipedia)

Bounce dengan penyebab internal seperti konten tidak sesuai kaidah penulisan yang baik, alamat email yang tidak valid, domain pengguna yang terdaftar dalam *Blacklist* dan sebagainya dikenal dengan *Hard Bounce*. Sedangkan *Soft Bounce* merupakan *Bounce* yang diakibatkan oleh faktor eksternal seperti penuhnya *Inbox*, server penerima email sedang *down* atau luring, isi pesan yang terlalu besar dan sebagainya.

Semakin besar persentase *Bounce* untuk setiap email dikirim, semakin besar pula dampak yang negatif yang diterima baik bagi perusahaan ataupun pelanggan. Selain menurunkan nilai pemasaran, *IP address* ataupun *DNS address* dari perusahaan atau jasa pengiriman *email marketing* dapat di *banned* oleh server mail dari email pelanggan. Proses *whitelist*/pemurnian email yang diblokir cenderung rumit dan memakan waktu yang relatif lama tergantung dari bagaimana IP atau DNS tersebut terblokir. Hal ini dapat menghambat aktivitas marketing perusahaan yang berakibat menurunnya omset perusahaan.

Menurut Hasan Alkahtani ^(Alkahtani) taksonomi untuk penyaringan spam pada email secara umum dapat dibagi menjadi 2 yaitu *Reputation-Based Filtering* dan *Content-Based Filtering*. *Reputation-Based Filtering* adalah penyaringan spam yang penyebabnya bukan dari konten pada email. Penyaring tersebut membuat penilaian terhadap reputasi dari pengirim, penerima dan perantara dalam proses pengiriman pesan. Sedangkan *Content-Based Filtering* adalah penyaringan spam dengan menilai apakah konten email mengandung kata-kata atau pola berpotensi menyebabkan spam.

Berdasarkan taksonomi tersebut penulis ingin mengimplementasikan beberapa metode untuk melakukan penyaringan spam pada email. Metode tersebut adalah *DNSBL Filtering* untuk implementasi *Reputation-Based Filtering* dan klasifikasi menggunakan metode *Support Vector Machine* untuk implementasi dari *Content-Based Filtering*.

DNSBL merupakan daftar alamat IP yang dicurigai mengirim spam dan digunakan untuk mencegah pesan email yang tidak diinginkan mencapai penerima yang tidak curiga. Satu hal yang penting untuk disebutkan adalah *blacklist* itu sebenarnya tidak memblokir pesan pengirim, tetapi justru sebagai tolak ukur. Penyedia menggunakan informasi ini dari berbagai layanan daftar hitam bersama dengan metrik internal untuk membuat keputusan untuk memblokir pesan atau tidak. (Wikipedia)

Dalam *Machine Learning*, *Support Vector Machine* (SVM) merupakan model pembelajaran *supervised* dengan algoritma pembelajaran asosiasi yang menganalisis data untuk keperluan analisis klasifikasi ataupun regresi. Diberikan serangkaian contoh pelatihan, masing-masing ditandai sebagai milik satu atau yang lain dari dua kategori, algoritma pelatihan SVM membangun model yang memberikan contoh baru untuk satu kategori atau yang lain, menjadikannya sebagai pengelompokan linear biner non-probabilistik (walaupun metode seperti skala Platt ada untuk menggunakan SVM dalam pengaturan klasifikasi probabilistik). (Wikipedia)

Berdasarkan hal tersebut, diperlukan langkah preventif agar email yang dikirim tidak mengalami *Bounce*. Solusi untuk permasalahan tersebut dapat berupa sistem yang memeriksa setiap email yang akan dikirim. Setiap email diperiksa apakah sudah memenuhi kriteria email yang lolos uji spam. Apabila memenuhi kriteria email tersebut akan di kirim ke pengguna dan sebaliknya. Hal tersebut lebih efektif daripada harus mendata setiap email yang mengalami *Bounce* untuk dikoreksi dan dikirim kembali. Selain itu langkah tersebut dapat meminimalisir *bandwidth* yang digunakan beserta beban jaringan. Dari sistem tersebut diharapkan kerugian akibatnya *Bounce* dapat diminimalisir. Oleh karena itu, pada penelitian ini penulis akan melakukan implementasi *DNSBL Filtering* dan *Support Vector Machine(SVM)* pada pembuatan sistem menggunakan Kotlin, dimana dapat disampaikan jika penulis akan membuat tugas akhir yang berjudul “SISTEM PERINGATAN DINI UNTUK MENDETEKSI SPAM PADA EMAIL MENGGUNAKAN DNSBL FILTER DAN SVM”

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dipaparkan diatas diketahui bahwa terdapat beberapa teknik untuk mengklasifikasi spam pada email. Namun, teknik klasifikasi tersebut belum diterapkan ke dalam sistem yang sebenarnya. Oleh karena itu didapatkan rumusan masalah sebagai berikut

1. Bagaimana langkah preventif untuk mendeteksi spam pada email secara dini ?
2. Bagaimana membangun Backend API yang menerapkan Support Vector Machine (SVM) dan DNSBL Filter ?

1.3 Batasan Masalah

Pada penelitian ini, terdapat beberapa batasan terhadap masalah yang akan diselesaikan supaya penelitian dapat dilakukan secara lebih terarah dan lebih dapat dipertanggung jawabkan. Batasan-batasan yang dimaksud antara lain :

1. Menggunakan DNSBL Filter dan Support Vector Machine sebagai metode penyaringan spam.
2. Menggunakan DNSBL dari https://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists.
3. Dataset berupa kumpulan email dalam bahasa inggris.
4. Menggunakan Kotlin sebagai bahasa utama dalam pembuatan sistem.
5. Menggunakan Restful API sebagai metode Web Service dengan hasil berupa JSON.
6. Aplikasi yang dikembangkan dalam penelitian ini berbasis Backend API atau aplikasi sisi server (*server-side*).
7. Fitur utama dari Backend API ini merupakan untuk pengecekan domain dari tiap DNSBL dan pengecekan konten spam.

1.4 Tujuan Penelitian

Dari rumusan masalah yang di jelaskan sebelumnya, tujuan dari kegiatan penelitian ini adalah membangun Sistem Penanganan Dini berbasis Backend API yang menerapkan Support Vector Machine (SVM) dan DNSBL Filter untuk menghindari email terindikasi spam terkirim.

1.5 Manfaat Penelitian

Setelah melaksanakan penelitian, diharapkan penulis dapat memberi manfaat antara lain :

1. Mencegah email bukan spam gagal terkirim ke inbox penerima.
2. Mengurangi potensi email mengalami *Bounce* atau terdeteksi sebagai spam.
3. Meminimalisir bandwidth yang digunakan beserta beban jaringan akibat *Bounce*.
4. Mencegah keputusan *email service provider* untuk melakukan banned IP atau DNS pengirim dan hal-hal merugikan lainnya.

BAB II

LANDASAN TEORI

2.1 Tinjauan Studi

Penelitian dalam bidang penyaringan email spam sudah banyak dikembangkan sehingga menghasilkan beberapa produk yang baru dan bermanfaat bagi pengguna. *State of The Art* atau penelitian sebelumnya, akan membahas mengenai persamaan dan perbedaan yang ada dalam penelitian sebelumnya. Dimana persamaan dan perbedaan akan dilihat dari objek yang diteliti, serta metode penelitiannya. Penyusunan data *State of The Art* akan disusun dalam bentuk data matrik, sehingga dapat memperlihatkan perbandingan antara penelitian sebelumnya dengan penelitian yang akan dilakukan. Berikut merupakan referensi yang digunakan :

1. Penelitian oleh Hasan (Alkahtani). Dalam penelitiannya yaitu *A Taxonomy of Email SPAM Filters*, penelitian tersebut membahas mengenai taksonomi untuk filter email spam.
2. Penelitian oleh Kamini (Bajaj, 2016). Dalam penelitiannya yaitu *A Multi-layer Model to Detect Spam Email at Client Side*, penelitian tersebut membahas mengenai penerapan filter ganda yaitu filter menggunakan *SpamBayes* dan filter terhadap ukuran dan struktur dokumen seperti tanggal dan waktu email, bidang subjek, *hyperlink*, digit angka, jumlah kata, penggunaan karakter khusus pada konten email untuk filter email spam. Filter ganda tersebut dapat meningkatkan performa hingga 60% jika dibandingkan hanya dengan menggunakan *SpamBayes* saja.
3. Penelitian oleh Karthika (D, 2014). Dalam penelitiannya yaitu *Latent Semantic Indexing Based SVM Model for Email Spam Classification*, peneliti melakukan perbandingan beberapa metode *Machine Learning* dalam melakukan filter terhadap email spam. Metode-metode tersebut diantaranya adalah *SVM + TF-IDF*, *SVM + LSI*, *Neural Network*, *PSO* dan *Firefly + Bayes*. Hasil dari perbandingan tersebut menunjukkan bahwa *SVM + LSI* memiliki presisi dan penarikan paling tinggi dan akurasi sedikit dibawah *PSO*.

4. Keempat oleh Tu Ouyang (Ouyang, 2013). Dalam penelitiannya yaitu *A large-scale empirical analysis of email spam detection through network characteristics in a stand-alone enterprise*, peneliti membangun penyaringan spam sebagai *pipeline* dengan menerapkan penyaringan berbasis DNSBL, penyaringan berbasis fitur paket SYN, penyaringan berbasis karakteristik lalu lintas dan berdasarkan pesan konten. Mereka menemukan bahwa *pipeline* tersebut bekerja sebaik pengaturan operasional. Selain itu setiap lapisan *pipeline* juga dapat bekerja dalam waktu yang lama dan dalam beberapa kasus tertentu dalam beberapa kasus, lapisan selanjutnya dapat mengimbangi kinerja yang buruk di lapisan sebelumnya.
5. Kelima oleh Simranjit Kaur Tuteja (Tuteja, 2016). Dalam penelitiannya yaitu *A Survey on Classification Algorithms for Email Spam Filtering* melakukan ulasan terhadap beberapa metode *Machine Learning* yang dapat digunakan untuk melakukan filter terhadap email spam. Metode tersebut antara lain : *Neural Network (NN)*, *Support Vector Machine (SVM) Classifier*, *Naïve Bayesian (NB) Classifier*, *J48 Classifier*. Percobaan dilakukan berdasarkan ukuran data yang berbeda dan ukuran fitur yang berbeda. Hasil klasifikasi akhir harus '1' jika akhirnya adalah spam, jika tidak, '0'. Klasifikasi *Naive Bayesian* menunjukkan hasil yang baik, tetapi *Neural Network* dan *SVM* tidak menunjukkan hasil yang baik dibandingkan dengan *J48* atau *Classifier Naïve Bayesian*. *Neural Network* dan *SVM* tidak sesuai untuk dataset untuk membuat keputusan biner. Dari percobaan ini, mereka dapat menemukan bahwa *Classifier J48* sederhana dapat memberikan hasil klasifikasi yang lebih baik untuk pemfilteran email spam.
6. Keenam oleh D.S.Silnov (Silnov, 2016). Dalam penelitiannya yaitu *An Analysis of Modern Approaches to the Delivery of Unwanted Emails (Spam)* mengulas cara kerja beberapa tools untuk menangkap spam di tingkat jaringan. Teknologi tersebut termasuk DNSBL (memeriksa untuk melihat apakah alamat IP dimasukkan dalam daftar hitam), memeriksa catatan PTR dari alamat IP dan domain email, SPF dan DKIM baru-baru ini dan beberapa lainnya.

Tabel 2.0-ITinjauan Pustaka

No.	Nama	Tahun	Judul	Metode	Hasil
1.	Hasan Al-kahtani	2019	A Taxonomy of Email SPAM Filters	<ul style="list-style-type: none"> • <i>BlackLists</i> • <i>WhiteLists</i> • <i>Challenge-ResponseSystems(CRS)</i> • <i>OriginDiversityAnalysis</i> • <i>ImplicitTechniques</i> • <i>ExplicitTechniques</i> • <i>RuleBasedFilters</i> • <i>StatisticalFilters</i> • <i>GeneticAlgorithms</i> • <i>ArtificialImmuneSystem</i> • <i>ArtificialNeuralNetworks</i> • <i>ClusteringTechniques</i> • <i>DecisionTreeTechnique</i> • <i>HoneyPots</i> • <i>Zombie-BasedApproach</i> 	Taksonomi mengenai penyarangan spam pada email yang berisi berbagai macam teknik yang telah digunakan atau diusulkan untuk digunakan untuk melawan SPAM, dan filter SPAM mana yang harus digunakan

2.	Kamini (Simi) Bajaj	2017	A Multi-layer Model to Detect Spam Email at Client Side	Model <i>multi-layer</i> yang menggunakan <i>SpamBayes</i> dan penyaringan non-tekstual yang mengeksplorasi teknik pembelajaran mesin alternatif.	Model <i>multi-layer</i> ini meningkatkan akurasi klasifikasi dan menghilangkan email abu-abu menjadi spam dan email ham.
3	Karthika Renuka	2014	Latent Semantic Indexing Based SVM Model for Email Spam Classification	SVM + LSI	Hasil dari perbandingan tersebut menunjukkan bahwa SVM + LSI memiliki presisi dan penarikan paling tinggi dan akurasi sedikit dibawah PSO.
4.	Tu Ouyang	2014	A large-scale empirical analysis of email spam detection through network characteristics in a	Penyaringan spam sebagai <i>pipeline</i> dengan menerapkan penyaringan berbasis DNSBL, penyaringan berbasis fitur paket SYN, penyaringan berbasis karakteristik lalu lintas dan berdasarkan pesan konten.	Mereka menemukan bahwa <i>pipeline</i> tersebut bekerja sebaik pengaturan operasional. Selain itu setiap lapisan <i>pipeline</i> juga dapat bekerja dalam waktu yang lama dan dalam beberapa kasus tertentu dalam beberapa kasus, lapisan selanjutnya dapat

			stand-alone enterprise		mengimbangi kinerja yang buruk di lapisan sebelumnya.
5.	D. S. Silnov	2016	An Analysis of Modern Approaches to the Delivery of Unwanted Emails (Spam)	<ul style="list-style-type: none"> - DNSBL - Memeriksa catatan PTR dari alamat IP dan domain email - SPF - DKIM 	Metode baru untuk menyaring spam pada email di tingkat jaringan
6.	Simranjit Kaur Tuteja	2016	A Survey on Classification Algorithms for Email Spam Filtering	<i>Neural Network (NN), Support Vector Machine (SVM) Classifier, Naïve Bayesian (NB) Classifier, J48 Classifier.</i>	<i>Classifier J48</i> sederhana dapat memberikan hasil klasifikasi yang lebih baik untuk pemfilteran email spam.

2.2 Tinjauan Pustaka

2.2.1 Surat Elektronik

Surat elektronik (akronim: ratel, ratron, surel, atau surat-e) atau pos elektronik (akronim: pos-el.) atau imel (bahasa Inggris: *email*) adalah sarana kirim mengirim surat melalui jalur jaringan komputer (misalnya Internet).

Struktur alamat surel, sebagai contoh:

surelsaya@surabaya.vibriel.net.id

Keterangan:

- surelsaya: nama kotak surat (*mailbox*) atau nama pengguna (*username*) yang ingin dituju dalam *mailserver*
- surabaya.vibriel.net.id: nama *mailserver* tempat pengguna yang dituju, rinciannya:
 - surabaya: *subdomain* (milik pemegang nama *domain*), biasanya merujuk ke suatu komputer dalam lingkungan pemilik *domain*
 - vibriel: nama *domain*, biasanya menunjukkan nama perusahaan/organisasi/perorangan (Vibriel)
 - net: *second level domain*, menunjukkan bahwa *domain* ini termasuk kategori *networking* (net)
 - id: *top level domain*, menunjukkan bahwa *domain* ini terdaftar di otoritas *domain* Indonesia (id) (Wikipedia)

Email merupakan media komunikasi bagi pengguna dan penyedia jasa Internet yang efektif. Menurut The Radiacti Group, Inc, sebuah perusahaan riset market teknologi, pengguna aktif email pada tahun 2015 telah mencapai 2.6 Milyar. (THE RADICATI GROUP, INC., 2015) Jumlah ini lebih banyak daripada Sosial Media seperti Facebook yang berjumlah 1.7 Milyar (Facebook, 2018) dan Twitter yang berjumlah 313 Juta. (Twitter, 2018)

2.2.2 Mail Server

Mail Server adalah sebuah aplikasi komputer dimana aplikasi ini menerima email masuk dari pengguna lokal (orang-orang dalam domain yang sama) serta pengirim jarak jauh dan meneruskan email keluar untuk pengiriman. Komputer yang memasang aplikasi semacam itu juga dapat disebut sebagai *Mail Server*. *Mail Server* dibedakan menjadi 2 yaitu *Mail Server* yang digunakan untuk email keluar disebut sebagai MTA (*Mail Transfer Agent*) dan Mail Server untuk masuk, menggunakan protokol POP3 / IMAP disebut sebagai MDA (*Mail Delivery Agent*).

2.2.3 Cara Kerja Email

Dalam proses pengiriman email setidaknya menggunakan 3 protokol utama yaitu :

2.2.2.1. IMAP

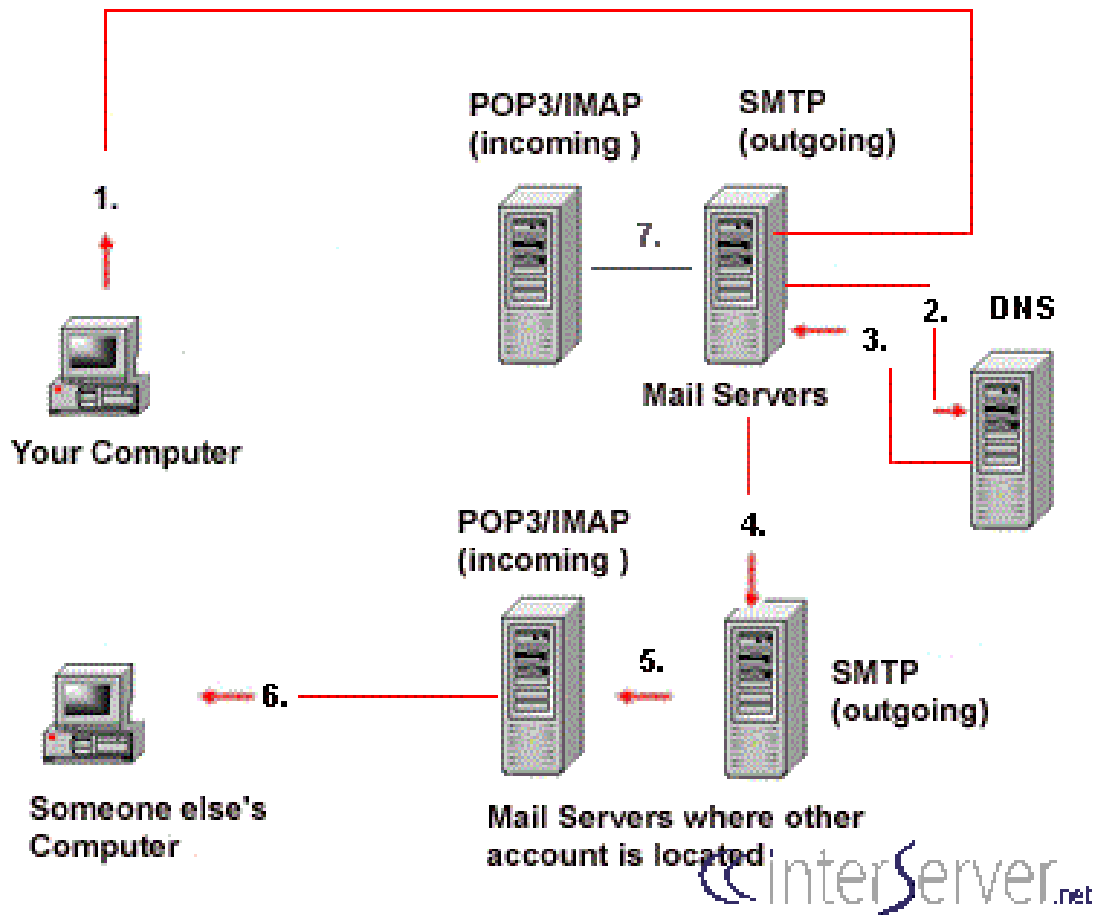
IMAP adalah singkatan dari *Internet Mail Access Protocol*. Protokol ini digunakan saat menerima email. Ketika seseorang menggunakan IMAP, email akan ada di server dan tidak diunduh ke kotak surat pengguna dan dihapus dari server. Ini membantu untuk memiliki lebih sedikit memori yang digunakan di komputer lokal dan memori server meningkat.

2.2.2.2. POP

POP adalah singkatan dari *Post Office Protocol*. Protokol ini juga digunakan untuk email yang masuk. Perbedaan utama dengan kedua protokol adalah bahwa POP mengunduh seluruh email ke komputer lokal dan menghapus data di server setelah diunduh. Ini sangat membantu dalam server dengan memori yang lebih sedikit. Versi POP saat ini adalah POP3.

2.2.2.3. SMTP

SMTP adalah singkatan dari *Simple Mail Transfer Protocol*. Email dikirim menggunakan protokol ini. Diagram di bawah ini menjelaskan jalur yang diambil email dari pengirim ke penerima yang dituju :



Gambar 2. 1 Jalur pengiriman email

Pertama pengirim harus memasukkan alamat email penerima bersama dengan pesan menggunakan aplikasi email. Ini harus dilakukan di komputer lokal. Setelah selesai dan tombol "*Send*" diklik, email akan menuju ke *MTA (Mail Transfer Agent)*. Komunikasi ini dilakukan melalui protokol SMTP.

Langkah selanjutnya adalah pencarian DNS. Sistem mengirim permintaan untuk mencari tahu MTA yang sesuai dari penerima. Ini akan dilakukan dengan bantuan *MX Record*. Di zona DNS, untuk domain alamat penerima, akan ada *MX Record (Mail Exchange Record)*. Ini adalah DNS Record yang menentukan server email suatu domain. Jadi, setelah pencarian DNS, respons diberikan ke server surat yang diminta dengan alamat IP dari server surat penerima. Dengan cara ini server mail 'to' diidentifikasi.

Langkah selanjutnya adalah mentransfer pesan di antara server surat. Protokol SMTP digunakan untuk komunikasi ini. Sekarang pesan sudah berada di *Mail Server penerima (MTA)*.

Sekarang, pesan ini ditransfer ke *Mail Delivery Agent* dan kemudian ditransfer ke komputer lokal penerima. Seperti yang telah kita lihat sebelumnya, dua protokol dapat digunakan di sini. Jika kami menggunakan POP3, maka seluruh email akan diunduh ke komputer lokal dan salinan di server akan dihapus. Jika protokol yang digunakan adalah IMAP, maka pesan email disimpan di server mail itu sendiri, tetapi pengguna dapat dengan mudah memanipulasi email di server mail seperti di komputer lokal. Inilah perbedaannya ketika menggunakan kedua protokol dan ini adalah bagaimana email Anda dikirimkan. Jika beberapa kesalahan terjadi untuk mengirim email, email akan tertunda. Ada antrian surat di setiap server surat. Email-email ini akan menunggu dalam antrian email. Server email akan terus mencoba mengirim ulang email. Setelah pengiriman email gagal secara permanen, server email dapat mengirim pesan email *bouncing* kembali ke alamat email pengirim.

2.2.4 Email Marketing

Email Marketing adalah email yang biasanya dikirim ke sekelompok orang dengan tujuan komersial. *Email marketing* sendiri biasanya memiliki konten berupa iklan, pengajuan kegiatan bisnis, permintaan penjualan atau donasi, dan dimaksudkan untuk membangun kesetiaan, kepercayaan, atau *brand awareness*. *Email marketing* dapat dikirim ke daftar prospek yang dibeli atau database pelanggan saat ini. *Email marketing* biasanya mengacu pada pengiriman pesan email dengan tujuan meningkatkan hubungan pedagang dengan pelanggan saat ini atau

sebelumnya, mendorong loyalitas pelanggan dan mengulang bisnis, memperoleh pelanggan baru atau meyakinkan pelanggan saat ini untuk membeli sesuatu dengan segera, dan berbagi iklan pihak ketiga. (Wikipedia) Dengan *email marketing*, perusahaan dapat mengirim informasi mengenai profil dan produk-produk mereka dengan cepat dan murah. *Email marketing* juga merupakan salah satu media marketing yang memiliki tingkat investasi yang tinggi. Menurut The Direct Marketing Association (DMA) pada tahun 2018 setiap rupiah yang diinvestasikan di email marketing, *Return of Investment*-nya adalah 3228%. (Aldighieri, 2018)

2.2.5 Bounce Message

Bounce Message adalah email otomatis yang memberi tahu pengirim pesan sebelumnya bahwa pesan itu belum terkirim (atau masalah pengiriman lainnya terjadi). Pengirim terkadang menerima *bounce message* dari server emailnya sendiri, melaporkan bahwa ia tidak dapat mengirim pesan, atau meskipun telah menerima pesan tersebut, sekarang pesan itu tidak dapat dikirim, juga menerima tanggung jawab untuk mengirimkan DSN jika pengiriman gagal. Karena berbagai alasan, terutama spam dan email virus, pengguna mungkin menerima pesan pantulan yang salah dikirim sebagai tanggapan terhadap pesan yang sebenarnya tidak pernah mereka kirim. (Wikipedia)

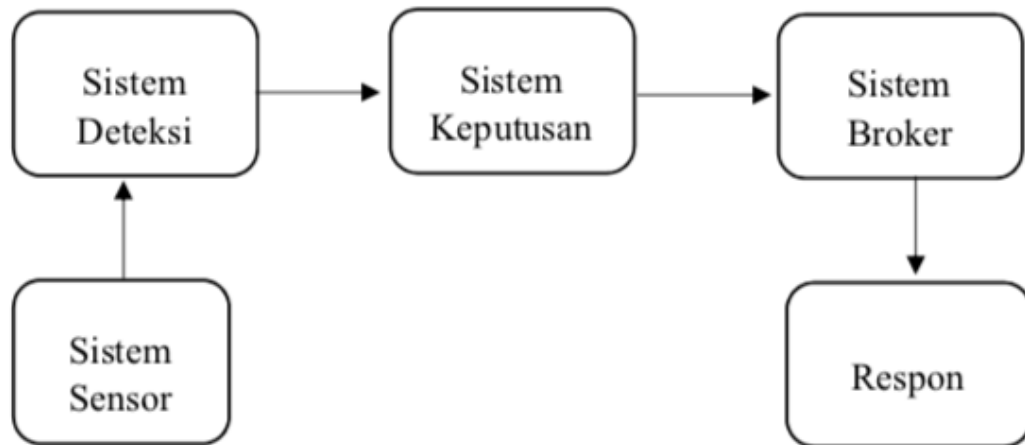
2.2.6 Kotlin

Kotlin adalah sebuah bahasa pemrograman dengan pengetikan statis yang berjalan pada Mesin Virtual Java ataupun menggunakan kompiler LLVM yang dapat pula dikompilasikan kedalam bentuk kode sumber JavaScript. Pengembang utamanya berasal dari tim programer dari JetBrains yang bermarkas di Rusia. Meskipun sintaksisnya tidak kompatibel dengan bahasa Java, Kotlin didesain untuk dapat bekerja sama dengan kode bahasa Java dan bergantung kepada kode bahasa Java dari Kelas Pustaka Java yang ada, seperti berbagai framework Java yang ada. Tim Pengembang memutuskan menamakannya Kotlin dengan mengambil nama dari sebuah pulau di Rusia, sebagaimana Java yang mengambil nama dari pulau Jawa di Indonesia. Setelah Google mengumumkan bahwa Kotlin menjadi bahasa kelas satu

bagi Android, maka bersama Java dan C++, Kotlin menjadi bahasa resmi untuk pengembangan aplikasi-aplikasi Android. (Wikipedia)

2.2.7 Sistem Peringatan Dini

Menurut Waidyanatha sistem peringatan dini merupakan rangkaian sistem komunikasi informasi yang terdiri dari subsistem sensor, deteksi, keputusan, serta broker yang berurutan, bekerja untuk memprediksi gangguan yang tidak diinginkan yang berpotensi dapat mengganggu stabilitas dunia nyata. (Waidyanatha, 2010) Diharapkan dengan adanya prediksi tersebut, dapat dilakukan penanggulangan secara efektif dan *realtime*.



Gambar 2. 2 Alur sistem peringatan dini

Sistem Sensor merupakan bagian dimana sistem menerima informasi masukan. Sumber dan bentuk masukan sangat tergantung dengan domain dari sistem peringatan dini itu sendiri. Sebagai contoh, saat ini banyak pemanfaatan media sosial sebagai sumber masukan untuk penanggulangan bencana alam, militer, krisis ekonomi dan lain sebagainya.

Sistem Deteksi merupakan bagian dimana sistem mengekstraksi data dari kumpulan data yang didapat oleh sensor. Sebagai contoh, sistem peringatan gempa menggunakan akan membaca data dari seismograf dan mendeteksi apakah dari data

seismograf tersebut menunjukkan terjadi gempa serta sistem dapat membedakan gempa sesungguhnya dengan gempa yang disebabkan oleh ledakan.

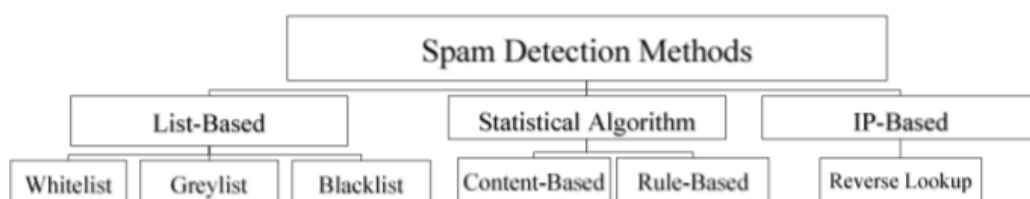
Sistem Keputusan merupakan bagian dimana sistem akan menentukan apakah sistem akan memberikan respon terhadap suatu gangguan atau tidak.

Sistem Broker merupakan sebuah sistem yang memiliki tugas untuk mengirimkan hasil keputusan ke bagian sistem respon. Sistem ini akan merubah data hasil keputusan ke dalam bentuk standard yang dapat di terima oleh sistem respon dan menentukan pesan mana yang akan dikirim.

Sistem respon merupakan keluaran dari sistem peringatan dini yang biasa nya berbentuk pesan peringatan. Dalam pesan peringatan tersebut selain memberitahukan tentang resiko yang akan terjadi, diberitkan juga instruksi berikutnya guna menaggulangi dampak gangguan.

2.2.8 Sistem Penanganan Dini pada Spam Email

Menurut Alireza (Pour, 2012) deteksi spam pada email dapat dilakukan sebelum email dikirim, yaitu dengan memindahkan sistem penyaringan spam ke mail server pengirim. Selain dapat mendeteksi spam dengan waktu yang lebih cepat, juga dapat menghindari penggunaan sumber daya jaringan berlebih. Beberapa metode tersebut dapat diklasifikasikan seperti bagan berikut :



Gambar 2. 3 Metode Deteksi Spam

1. List Based

a. Whitelist

Dalam teknik ini, setiap pengguna menyimpan kontak emailnya dalam daftar yang disebut *Whitelist*. Oleh karena itu, setiap email yang diterima dengan alamat koresponden dari daftar ini diterima, dan semua alamat lain dari daftar ini dianggap tidak pasti.

b. Greylist

Pada langkah pertama, semua email yang diterima ditolak. Karena kebijakan ini, spammer tidak mencoba mengirim ulang email yang ditolak karena memakan waktu lama bagi mereka. Sebagai gantinya, spammer lebih memilih untuk mencari alamat email lain tanpa pemfilteran *Greylist*.

c. Blacklist

Dalam pemfilteran *Blacklist*, alamat IP dan nama domain dari server pengirim disimpan dalam daftar yang disebut *Blacklist* dan email dari alamat IP dan domain tersebut diblokir. Kemudian, berdasarkan kebijakan pihak penerima, email dari alamat IP *Blacklisted* dihapus atau dikirim ke folder spam.

2. Statisfical Algorithm

a. Content-Based

Content-Based Filtering adalah teknik penyaringan yang menggunakan *machine learning*. Untuk mendapatkan hasil yang memuaskan, administrator server mail perlu melatih filter untuk menjalankan fungsinya. Pemfilteran ini mulai berfungsi berdasarkan beberapa kata yang telah ditentukan setelah email diterima seluruhnya. Kata-kata khusus ini dikumpulkan oleh laporan statistik berdasarkan kata-kata dan frasa yang dikumpulkan dari email spam.

b. Rule-Based

Penyaringan berdasarkan aturan mirip dengan yang berbasis konten dengan beberapa perbedaan. Teknik ini bekerja melalui beberapa aturan dan regulasi tertentu. Dengan aturan ini, filter memutuskan untuk meneruskan atau memblokir email yang diterima.

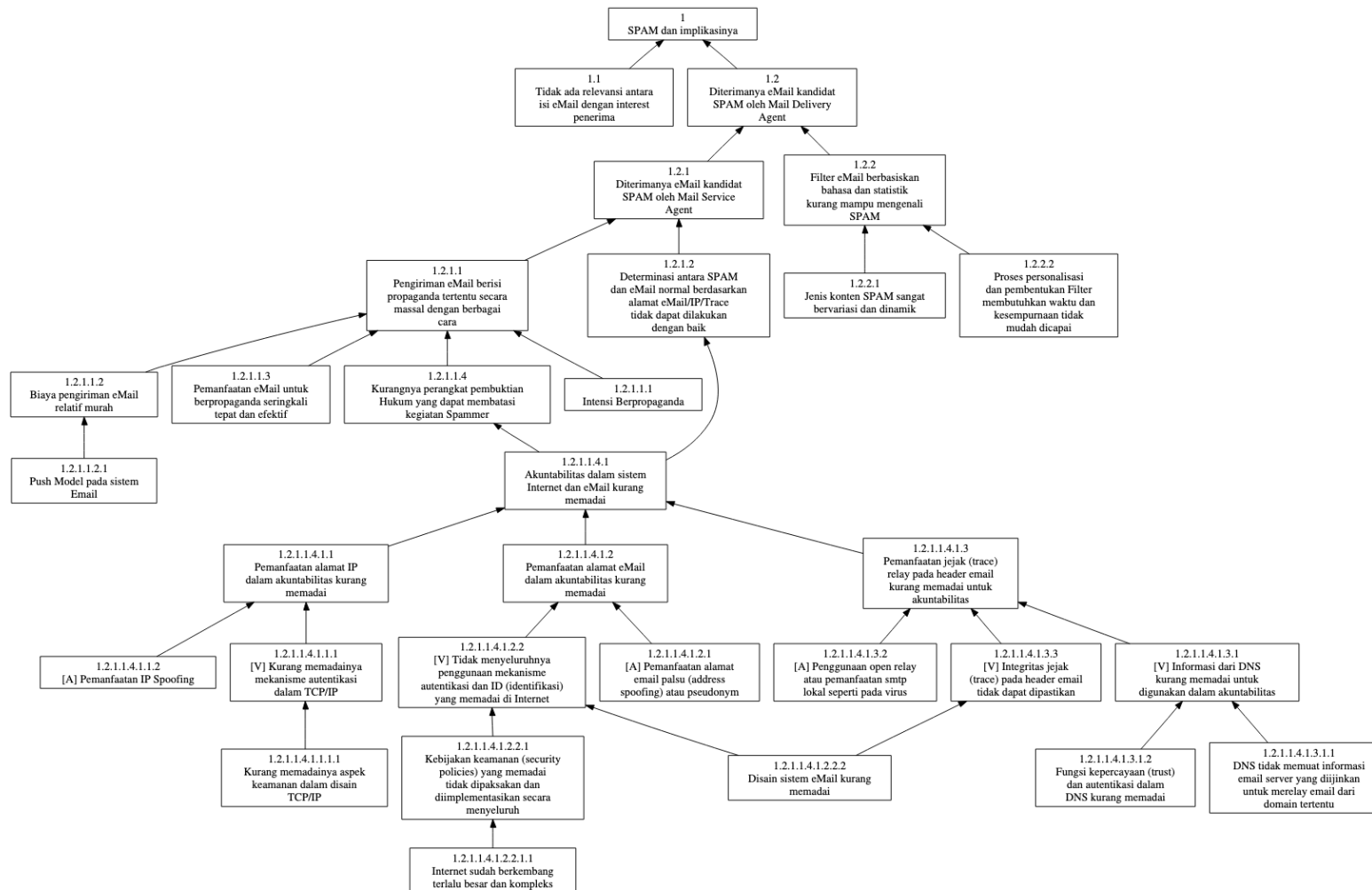
3. IP-Based

a. Reverse Lookup

Di *reverse lookup*, juga dikenal sebagai *reverse DNS (Domain Name System) lookup*, *host* dikaitkan dengan alamat *IP (Internet Protocol)* yang diberikan. Dengan menggunakan metode ini, penerima dapat mengkonfirmasi identitas nama domain pengirim.

2.2.9 Analisis Spam

Analisis ontologi berhasil mengidentifikasi beberapa kekurangan dalam rancangan dan implementasi yang merupakan kelemahan Internet mail untuk mencapai keamanan yang memadai. Kekurangan dan kelemahan tersebut adalah fakta yang digabungkan dengan fakta lainnya, baik dari beberapa penelitian tentang Spam maupun dari pengetahuan analis, sebagai kandidat faktor causal. Fakta-fakta tersebut diurutkan berdasarkan waktu terjadinya dan dihubungkan satu dengan lainnya dengan relasi kausalitas. Pengujian counterfactuals digunakan untuk memeriksa kesahihan penalaran kausalitas. Hasil analisis digambarkan pada gambar berikut.



Gambar 2. 4 Analisis Spam

Hal yang menarik dalam analisis ini adalah ketidakmapanaan akuntabilitas(accountability) sistem Internet mail (1.2.1.1.4.1) yang cukup signifikan sebagai faktor penyebab Spam. Hal ini menyebabkan kurang memadainya proses pemilahan (filtering) antara email biasa dengan spam, yang memanfaatkan alamat IP atau alamat email sebagai patokan. Kurang memadainya akuntabilitas juga menyebabkan kurang kuatnya bukti yang dapat digunakan oleh perangkat hukum untuk membuktikan kegiatan spammer dan membatasinya.

Akuntabilitas merupakan hal yang penting dalam keamanan sistem. Dengan akuntabilitas yang baik, setiap perubahan dalam sistem, waktu, sebab, dan akibatnya, terekam secara rinci, **dapat dipertanggungjawabkan**, dan **tidak dapat diingkari** oleh pelaku. Sehingga hasilnya dapat digunakan untuk keperluan auditing dan dalam mekanisme keamanan sistem lainnya. Akuntabilitas harus memuat beberapa hal seperti: kondisi awal (pre-condition), aktor yang melakukan, aksi, transisi dan perubahan, waktu pencatatan, dan kondisi akhir (post-condition). Beberapa syarat akuntabilitas adalah: autentikasi yang memadai untuk mengidentifikasi aktor, adanya audit-trail, integritas dari audit-trail itu sendiri, dan keadaan yang tidak dapat diingkari (non-repudation). Mekanisme akuntabilitas pada setiap sistem harus dirancang dengan benar, diimplementasikan secara menyeluruh, dan selalu terjaga integritasnya. Sehingga tidak ada contoh (counter-examples) yang dapat menggagalkan penalaran hasil dari proses akuntabilitas.

Kurang memadainya akuntabilitas dalam sistem disebabkan tiga faktor, yaitu kurang memadainya akuntabilitas pada lapisan network (1.2.1.1.4.1.1), kurang memadainya akuntabilitas alamat email (1.2.1.1.4.1.2), dan kurang memadainya jejak (trace) dalam sebagai bagian penting dari proses akuntabilitas (1.2.1.1.4.1.3). Dengan pengujian countefactuals dapat dijelaskan bahwa: akuntabilitas dapat dicapai dengan baik apabila ketiga faktor tersebut dapat diatasi secara menyeluruh tanpa kecuali.

Pada awalnya TCP/IP dilahirkan sebagai hasil riset packet switching network yang dilakukan oleh DARPA dan pada akhirnya dimanfaatkan sebagai media komunikasi antara peneliti pada universitas-universitas di USA. Pemanfaatannya pada publik yang akhirnya membawa Internet berkembang sampai saat ini. TCP/IP tidak didisain untuk mendukung aplikasi yang mengharuskan adanya infrastruktur keamanan sistem yang memadai (1.2.1.1.4.1.1.1), sehingga mekanisme autentikasi pa-

da lapisan network kurang diperhatikan dalam perancangan TCP/IP (1.2.1.1.4.1.1.1). Kelemahan ini dimanfaatkan oleh Spammer untuk mengirimkan dengan menggunakan alamat IP palsu (IP Spoofing - 1.2.1.1.4.1.1.2). Oleh karena itu alamat IP yang tetera dalam atau pemeriksaan paket IP dalam proses pengiriman-penerimaan, tidak dapat dijadikan sumber yang cukup dalam proses akuntabilitas (1.2.1.1.4.1.1). Hal ini juga berdampak bahwa alamat IP kurang kuat untuk dijadikan alat bukti, karena siapa saja dapat mengirimkan dengan alamat IP milik orang lain.

Dalam analisis ontologi sebelumnya, diketahui bahwa rancangan sistem yang digunakan saat ini belum memperhatikan fungsi-fungsi keamanan seperti autentikasi pengirim, autentikasi relai, dan integritas data (1.2.1.1.4.1.1.2). Spammer memanfaatkan kelemahan ini untuk mengirimkan dengan alamat palsu (pseudonym) (1.2.1.1.4.1.2.1). Tentu saja alamat pengirim (**From:**) baik dalam proses berjalannya protokol SMTP maupun yang tercantum dalam header tidak dapat digunakan sebagai sumber yang cukup dalam akuntabilitas (1.2.1.1.4.1.2). Autentikasi pada lapisan aplikasi (application layer) yang saat ini dikembangkan (SMTP/TLS - Simple Mail Transfer Protocol / Transport Layer Security) adalah fungsi tambahan dalam SMTP. Selain itu tengah dikembangkan bakuan S/MIME (Secure-MIME) untuk memelihara integritas dan membuktikan kesahihan identifikasi pengirim, dengan memanfaatkan penyandian (cryptography), fungsi hash, dan tanda tangan digital (digital-signature). Sayangnya kedua mekanisme tersebut dan tidak diimplementasikan secara menyeluruh diseluruh Internet (1.2.1.1.4.1.1.2.1) meskipun keduanya sudah cukup matang dalam penelitian dan implementasinya. Hal ini disebabkan karena perkembangan Internet yang sudah terlalu besar dan kompleks (1.2.1.1.4.1.2.2.1.1). Pembaharuan memerlukan kesediaan seluruh pengguna Internet, penyedia layanan, dan vendor untuk memikul bersama tugas tersebut.

Selain masalah autentikasi dan identifikasi, kelemahan desain sistem juga disebabkan oleh lemahnya integritas dan faktor non-repudiation pada jejak pengiriman (trace) (1.2.1.1.4.1.3.3). Selain itu informasi dari DNS kurang memadai sebagai sumber pelacakan dalam proses akuntabilitas (1.2.1.1.4.1.3.1). Hal ini disebabkan oleh dua faktor; pertama DNS hanya menyediakan data MX (mail-exchanger), yaitu server yang bertanggungjawab sebagai penerima mail dari sebuah domain, sedangkan data server yang berhak melakukan relai (relayserver) tidak ada dalam data

DNS, sehingga proses verifikasi relai server tidak dapat dilakukan (1.2.1.1.4.1.3.1.1); kedua, dalam proses DNS lookup, tidak ada mekanisme autentikasi dan kepercayaan (trust) dari sistem DNS itu sendiri, sehingga data yang diperoleh dari hasil DNS lookup tidak dapat diverifikasi kebenarannya (1.2.1.1.4.1.3.1.2).

Ketidakmampuan sistem untuk memverifikasi integritas jejak pada header dan tidak adanya informasi relai server yang berhak merelai dari domain tertentu merupakan kelemahan sistem yang dimanfaatkan oleh pengirim Spam dengan menggunakan open-relay server, yaitu server mail yang tidak terkonfigurasi dengan benar sehingga dapat melakukan open relaying. Hal ini juga bisa dilakukan dengan menggunakan local smtp yaitu pengiriman tidak melalui server perantara, seperti yang digunakan oleh virus untuk menyebarkan diri lewat (1.2.1.1.4.1.3.2). Ketiga faktor ini menyebabkan pemanfaatan jejak kurang memadai dan tidak dapat dipertanggungjawabkan dalam proses akuntabilitas (1.2.1.1.4.1.3).

Kurang memadainya akuntabilitas dalam sistem menyebabkan kurangnya alat bukti yang bisa digunakan oleh perangkat hukum untuk membuktikan dan membatasi kegiatan pengirim Spam (1.2.1.1.4). Akibatnya pengirim Spam bebas mengirimkan berisi propaganda ke semua orang dengan berbagai cara (1.2.1.1). Selain itu, beberapa faktor lain juga menjadi penyebab kemudahan, kebebasan, dan kontinuitas pengiriman Spam. Faktor kedua adalah biaya pengiriman yang cukup murah (1.2.1.1.2) yang disebabkan oleh push-model dalam sistem (1.2.1.1.2.1). Sebagian besar biaya pada pengiriman dibebankan pada penerima bukan pada pengirimnya. Komponen biaya ini termasuk diantaranya mailbox dan trafik masuk (incoming traffic). Apabila saja biaya pengiriman cukup mahal, misalnya dengan menggunakan pull-model seperti pada [2], maka pengirim spam juga akan membatasi diri dalam kegiatannya. Faktor ketiga adalah Spam seringkali tepat pada sasarannya [5], sehingga pengirim spam termotivasi untuk mengirimkannya lagi (1.2.1.1.3). Faktor lainnya adalah intensi untuk melakukan propaganda (1.2.1.1.1) yang jelas dimiliki oleh pengirim spam.

Dalam sistem, pengiriman dilayani oleh Mail Service Agent (MSA) yang merupakan perisai pertama sistem pencegahan spam. Pada tahap ini MSA dapat memeriksa alamat IP, alamat, maupun jejak yang diterimanya, dan dibandingkan

dengan blacklist atau whitelist yang berisikan data Spam berdasarkan alamat IP pengirim dan alamat penerima. Telah disinggung sebelumnya bahwa determinasi antara dan spam yang menggunakan alamat IP, alamat, dan jejak tidak dapat dilakukan dengan baik (1.2.1.2) akibat tidak memadainya akuntabilitas (1.2.1.1.4.1). Hal ini menyebabkan yang berpotensi menjadi spam diterima oleh MSA (mail accepted) (1.2.1) karena pengirim spam mengelabuinya dengan tindakan-tindakan (1.2.1.1.4.1.1.2-1.2.1.1.4.1.2.1-1.2.1.1.4.1.3.1.2).

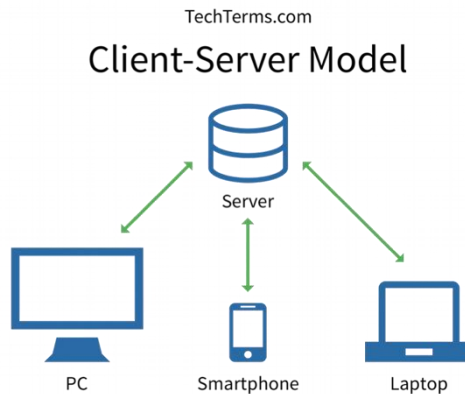
Dalam sistem modern, setelah diterima oleh MSA maka tersebut diambil-alih oleh Mail Delivery Agent (MDA) untuk diperiksa isinya dan ditempatkan pada mailbox. Pemeriksaan ini (content inspection/filtering) mempunyai tujuan yang berbeda-beda, misalnya mendeterminasi berisi pornografi untuk melindungi anak-anak dibawah umur, penggunaan resmi perusahaan untuk kegiatan pribadi, dan tentu saja spam dan virus. Pendeteksi berisi spam saat ini dilakukan dengan menggunakan metoda bayesian (statistik) dari keseringan (frekwensi) kemunculan kata-kata tertentu pada berisi spam dan header yang disimpulkan sebagai karakteristik spam. Metoda di atas membutuhkan proses belajar dan pembaharuan tabel bayesian, yang cukup memakan waktu (1.2.2.2), karena determinasi spam berdasarkan isi sangat bergantung pada interest penerima. Dengan kata lain tidak semua mail berisi propaganda adalah spam untuk penerima tertentu, sehingga penerapan filter kurang dapat diimplementasikan secara system-wide, apalagi jika profil pengguna sistem berbeda-beda. Kelemahan ini disiasati oleh pengirim spam dengan menggunakan karakter-karakter tertentu dan konten yang dinamik (1.2.2.1), sehingga filter kurang mampu mengenali Spam (1.2.2).

Hal ini menyebabkan kandidat spam yang berisi propaganda diterima oleh MDA (1.2) dan ditempatkan pada mailbox penerima. Pada saat penerima membaca tersebut dan isinya tidak relevan dengan interest penerima, maka tersebut dikategorikan sebagai Spam (1) dan berdampak dengan kerugian penerima. (Tarigan, 2004)

2.2.10 Client-Server Model

Merupakan sebuah aplikasi yang terdistribusi dimana memiliki server yang bekerja sebagai penyedia layanan atau *resource* dan client yang menggunakan layanan tersebut (Tech Term). Secara umum relasi yang dimiliki server dengan client

ialah *one-to-many*, jadi sebuah server bisa diakses banyak client dalam waktu yang bersamaan. Berikut gambaran dari *Client- Server Model*. Biasanya client disebut juga frontend dan server disebut backend dimana kedua buah sistem ini saling berhubungan dan menjadi sebuah sistem yang utuh.



Gambar 2. 5 Client-Server Model

- Client-side: merupakan sebuah aplikasi sisi klien yang dijalankan dengan sebuah *device* yang menerima inputan dari pengguna. Aplikasi sisi klien ini juga menyiapkan data atau informasi yang dibutuhkan pengguna, setelah pengguna memasukkan informasi, data akan dikirim ke server atau yang biasanya disebut *request*.
- Server-side: merupakan sebuah aplikasi sisi server yang mana berfungsi sebagai menerima *request* dari aplikasi sisi klien yang langsung memproses *request* tersebut dan mengirimkan tanggapan sesuai dengan permintaan aplikasi sisi klien atau biasa disebut *response*.

2.2.11 World Wide Web

Perkembangan akses internet sangatlah pesat, hal ini menjadi salah satu bukti bahwa teknologi juga ikut berkembang (millions)). Disisi yang sama pertumbuhan sistem informasi juga sangat cepat. Dengan hal ini seluruh sistem informasi lebih gampang untuk diakses. Seluruh sistem informasi yang dapat diakses menggunakan

web browser disebut dengan halaman web(*web page*). Dalam bahasa ilmiah halaman web disebut juga *World Wide Web* atau biasa disingkat dengan WWW (Petra).

2.2.12 HTML

Sebuah teknologi informasi berbasis situs web tidak bisa terlepas dari teknologi bernama HTML. HTML merupakan teknologi dasar untuk membangun sebuah halaman web(*web page*). HTML digunakan untuk mendefinisikan atau mentranslasikan konten dari halaman web tersebut, seperti link, paragraf, gambar, heading, dan lain sebagainya (Mozilla). Berikut merupakan contoh syntax dari HTML.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title></title>
5    </head>
6    <body>
7
8    </body>
9  </html>
10
```

Gambar 2. 6 Contoh Syntax HTML

2.2.13 Hypertext Transfer Protocol

Merupakan sebuah protokol *application layer* untuk mengirim atau menerima sebuah dokumen seperti HTML dan lain lain. HTTP digunakan untuk menyambungkan antara web browser dan web server. HTTP juga digunakan sebagai penghubung antara *client-server model*, dimana *client* meminta tanggapan(*response*) dengan menggunakan permintaan(*request*) (Mozilla).

2.2.14 Web Service

Merupakan salah satu bentuk *Client-Server* model yang termasuk ke dalam Interoperabilitas dengan melakukan komunikasi melalui *World Wide Web(WWW)* dan *HyperText Transfer Protocol (HTTP)*. *Web Service* menyediakan sebuah layanan yang dapat diakses oleh semua platform dan kerangka kerja (Oracle). *Web service* dapat menerima dan menyimpan informasi dalam format seperti HTTP, XML, SSL, SMTP, SOAP, dan JSON.

2.2.15 REST API

Merupakan sekumpulan fungsi yang mana developer dapat melakukan kegiatan *request* dan *response* (Deering). Ada enam aturan dimana sebuah sistem dikatakan REST API, berikut aturan aturan tersebut (Gustavsson, 2016).

- a. Client-Server : Secara arsitektur REST memisahkan pemrosesan sistem menjadi dua komponen. Server merupakan komponen yang menyediakan layanan dan menanggapi permintaan untuk service tersebut. Client merupakan komponen yang terhubung ke server untuk melakukan permintaan ke server.
- b. Stateless : Server tidak melihat status sesi dari Client. Setiap *Request* yang dikirim melalui Client harus berisi seluruh informasi yang dibutuhkan agar server dapat mengerti apa yang harus dikirim ke Client.
- c. Cacheable : *Response* yang dikirim oleh server harus *cacheable*. Hal ini bertujuan untuk menghindari request yang tidak diperlukan.
- d. Uniform Interface : Dengan perbedaan komponen dari sistem REST untuk melakukan komunikasi dari kedua komponen memerlukan standar yang sama(*Uniform Interface*). Hal ini juga mengurangi efisiensi dalam mengirim informasi, karena informasi yang merupakan bentuk standar sedangkan dari pihak aplikasi client memiliki kebutuhan yang berbeda.
- e. Layered System : Sistem ini berada di layer yang berbeda. Satu layer hanya bisa berinteraksi dengan layer terdekatnya. Tetapi dari komponen komponen sistem tidak perlu mengerti satu sama lain, asalkan keduanya bekerja dengan baik maka komunikasi data juga akan bekerja.

2.2.16 DNS dan Tipe DNS

Sistem Penamaan Domain (bahasa Inggris: (*Domain Name System*; *DNS*) adalah sebuah sistem yang menyimpan informasi tentang nama host ataupun nama domain dalam bentuk basis data tersebar (*distributed database*) di dalam jaringan komputer, misalkan: Internet. DNS menyediakan alamat IP untuk setiap nama host dan mendata setiap server transmisi surat (*mail exchange server*) yang menerima surel (*email*) untuk setiap domain. Menurut browser Google Chrome, DNS adalah layanan jaringan yang menerjemahkan nama situs web menjadi alamat internet.

DNS menyediakan pelayanan yang cukup penting untuk Internet, ketika perangkat keras komputer dan jaringan bekerja dengan alamat IP untuk mengerjakan tugas seperti pengalamatan dan penjaluran (*routing*), manusia pada umumnya lebih memilih untuk menggunakan nama host dan nama domain, contohnya adalah *Uniform Resource Locator*(*URL*) dan alamat surel. Analogi yang umum digunakan untuk menjelaskan fungsinya adalah DNS bisa dianggap seperti buku telepon internet di mana saat pengguna mengetikkan www.indosat.net.id di peramban web maka pengguna akan diarahkan ke alamat IP 124.81.92.144 (IPv4) dan 2001:e00:d:10:3:140::83 (IPv6).

2.2.17 Jenis-jenis DNS Record

Beberapa kelompok penting dari data yang disimpan di dalam DNS adalah sebagai berikut:

- A record atau catatan alamat memetakan sebuah nama host ke alamat IP 32-bit (untuk IPv4).
- AAAA record atau catatan alamat IPv6 memetakan sebuah nama host ke alamat IP 128-bit (untuk IPv6).
- CNAME record atau catatan nama kanonik membuat alias untuk nama domain. Domain yang di-alias-kan memiliki seluruh subdomain dan rekod DNS seperti aslinya.
- MX record atau catatan pertukaran surat memetakan sebuah nama domain ke dalam daftar *mail exchange server* untuk domain tersebut.

- PTR record atau catatan penunjuk memetakan sebuah nama host ke nama kanonik untuk host tersebut. Pembuatan rekod PTR untuk sebuah nama host di dalam domain in-addr.arpa yang mewakili sebuah alamat IP menerapkan pencarian balik DNS (*reverse DNS lookup*) untuk alamat tersebut. Contohnya (saat penulisan / penerjemahan artikel ini), www.icann.net memiliki alamat IP 192.0.34.164, tetapi sebuah rekod PTR memetakan 164.34.0.192.in-addr.arpa ke nama kanoniknya: referrals.icann.org.
- NS record atau catatan server nama memetakan sebuah nama domain ke dalam satu daftar dari server DNS untuk domain tersebut. Perwakilan bergantung kepada rekod NS.
- SOA record atau catatan otoritas awal (*Start of Authority*) mengacu server DNS yang menyediakan otorisasi informasi tentang sebuah domain Internet.
- SRV record adalah catatan lokasi secara umum.
- Catatan TXT mengizinkan administrator untuk memasukkan data acak ke dalam catatan DNS; catatan ini juga digunakan di spesifikasi *Sender Policy Framework*.

Jenis catatan lainnya semata-mata untuk penyediaan informasi (contohnya, catatan LOC memberikan letak lokasi fisik dari sebuah host, atau data ujicoba (misalkan, catatan WKS memberikan sebuah daftar dari server yang memberikan servis yang dikenal (*well-known service*) seperti HTTP atau POP3 untuk sebuah domain (Wikipedia).

2.2.18 DNSBL

Domain Name System-based Blackhole List (DNSBL) atau *Real-time Blackhole List (RBL)* merupakan daftar hitam dari domain yang terdeteksi mengirim email spam. Sebagian besar perangkat lunak mail server dapat dikonfigurasi untuk menolak atau menandai pesan yang berasal dari domain yang sudah tercatat dalam daftar blokir. Istilah “*Blackhole List*” sendiri berasal dari istilah “*blacklist*” dan “*blocklist*”.

Terdapat lusinan DNSBL yang ada, dimana menggunakan array besar yang berisi kriteria dari alamat yang terdaftar dan tidak. DNSBL dapat berisi alamat

komputer zombie atau mesin yang dikhususkan untuk mengirim spam, Penyedia Layanan Internet/*Internet Service Provider (ISP)* yang bersedia menjadi host spammer atau yang mengirim spam ke sistem honeypot (Wikipedia).

2.2.19 Jenis jenis DNSBL

DNSBL sendiri dapat dikelompokkan berdasarkan fokus *blacklist* dan cara pemeliharaan domain yang terblacklist sebagai berikut:

Tabel 2.0-2Klasifikasi DNSBL yang digunakan CSAIL (Computer Science and Artificial Intelligence Laboratory)

Fokus Blacklist	Cara Pemeliharaan	Blacklist
Spammer	Konservatif	sbl.spamhaus.org
Proxy terbuka	Konservatif	opm.blitzed.org
Relay terbuka	Konservatif	rbl.maps.vix.com, list.dsbl.org, multihop.dsbl.org, relays.mail-abuse.org, relays.osirusoft.com, relays.visi.com, relays.orbs.org, relays.ordb.org
Relay terbuka	Agresif	unconfirmed.dsbl.org, dnsbl.sorbs.net
Serangan Virus/Exploitasi	Agresif	xbl.spamhaus.org, cbl.abuseat.org
Netblock ISP/Negara	Agresif	{argentina,att,...}.blackholes.us, dul.maps.vix.com, dul.dnsbl.sorbs.net, dynablock.easynet.nl, blackholes.easynet.nl, dialups.mail-abuse.org
RFC Violators	Mix	{dsn,ipwhois,whois,abuse,postmaster,bogusmx.rfc-ignorant.org
Mix	Mix	sbl-xbl.spamhaus.org, bl.spamcop.net, dnsbl.njabl.org
Commercial	Commercial	hil.habeas.com, sa-hil.habeas.com, que-

		ry.bondedsender.org, sa-other.bondedsender.org, sa-trusted.bondedsender.org
Unknown	Unknown	rbl.dorkslayers.com, rbl.debian.net

2.2.20 Cara Kerja Pengecekan Domain berbasis DNSBL

Berikut merupakan alur proses pengecekan domain berbasis DNSBL :

1. DNS lookup tipe A pada domain yang akan di cek untuk mendapatkan ip. Sebagai contoh DNS lookup pada google.com akan mendapatkan ip 172.217.194.113.
2. Ubah susunan ip dari a.b.c.d menjadi d.c.b.a. Sehingga susunanya menjadi 113.194.217.172.
3. Konkat dengan DNSBL yang ingin di test. Sebagai contoh DNSBL yang akan dipakai adalah sbl.spamhaus.org. Sehingga susunanya menjadi 113.194.217.172.sbl.spamhaus.org.
4. DNS lookup tipe TXT pada ip yang sudah di konkat. Apabila hasilnya sukses maka domain tersebut telah terblokir pada DNSBL yang terkait dan sebaliknya. (Jung, 2004)

2.2.21 Web Scraping

Web scraping, web harvesting atau *web data extraction* adalah *data scraping* untuk mengekstrak data dari website (Boeing, 2016). Perangkat lunak web scrapper akan mengakses *World Wide Web* melalui protokol HTTP atau melalui web browser. *Web scrapping* dapat dilakukan menggunakan *bot* atau *web crawler* (Wikipedia).

2.2.22 Machine Learning

Machine Learning, cabang dari kecerdasan buatan, adalah disiplin ilmu yang mencakup perancangan dan pengembangan algoritme yang memungkinkan komputer untuk mengembangkan perilaku yang didasarkan pada data empiris, seperti dari sensor data basis data. Sistem pembelajar dapat memanfaatkan contoh data untuk menangkap ciri yang diperlukan dari probabilitas yang mendasarinya (yang tidak

diketahui). Data dapat dilihat sebagai contoh yang menggambarkan hubungan antara variabel yang diamati. Fokus besar penelitian pembelajaran mesin adalah bagaimana mengenali secara otomatis pola kompleks dan membuat keputusan cerdas berdasarkan data. Kesukarannya terjadi karena himpunan semua perilaku yang mungkin, dari semua masukan yang dimungkinkan, terlalu besar untuk diliput oleh himpunan contoh pengamatan (data pelatihan). Karena itu pembelajar harus merampatkan (generalisasi) perilaku dari contoh yang ada untuk menghasilkan keluaran yang berguna dalam kasus-kasus baru (Wikipedia).

2.2.23 Text Mining

Untuk dapat mengklasifikasikan suatu teks kita memerlukan sebuah teknik yang dapat mengambil informasi pada data yang berupa teks. Oleh karena itu, kita memerlukan teknik yang disebut *text mining*. *Text mining* merupakan sebuah pengembangan teknik data mining yang mana *text mining* dapat mencari pola tertentu pada kumpulan data tidak terstruktur (text) dalam suatu dokumen. Secara keseluruhan *text mining* melibatkan *data mining*, *machine learning*, *information retrieval*, dan *natural language processing*. Hasil dari sebuah proses *text mining* adalah sebuah pengetahuan berupa pola yang dapat diterapkan untuk menyelesaikan masalah.

Text mining memiliki alur dan proses yang hampir serupa dengan *data mining* pada umumnya. Yang pertama adalah *data preparation*, pada tahapan ini data yang digunakan akan dipilah terlebih dahulu. Selain itu pada tahapan ini dilakukan juga proses *preprocessing* dokumen (kategorisasi teks, ekstraksi informasi, ekstraksi istilah). terdapat beberapa hal yang dilakukan dalam *preprocessing* dokumen sebagai berikut, diantaranya :

1. Tokenisasi

Tokenisasi merupakan proses memecah kalimat menjadi kumpulan token atau kata. proses ini akan menghilangkan beberapa karakter seperti “ ”(spasi), “,”, “.” dan tanda baca lainnya.

2. Stop word removal

Proses ini akan menghilangkan beberapa kata yang tidak berpengaruh seperti “di”, “ke”, “yang”, dan lain sebagainya.

3. Stemming

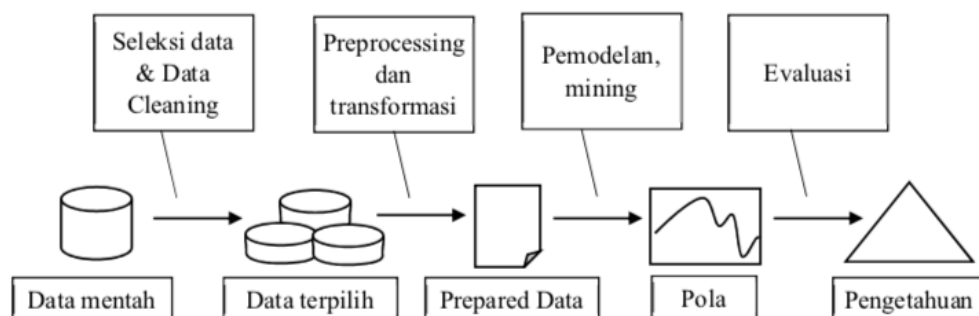
Proses ini akan merubah suatu token ke dalam bentuk dasarnya. Contohnya “menyapu” menjadi “sapu”. Bila tidak ditemukan rule yang mengatur bentuk dasar dari sebuah token maka token tersebut tidak akan berubah.

4. Termweighting

Term weighting atau pembobotan adalah sebuah proses untuk menentukan bobot dari setiap kata. pada proses ini data yang sebelumnya berupa data yang tidak terstruktur sudah berubah menjadi data intermediate yang lebih terstruktur.

Tahap berikutnya adalah pemodelan, pada tahap ini data yang telah di *preprocessing* telah menjadi data *intermediate* yang lebih terstruktur. lalu data *intermediate* tadi dianalisa menggunakan teknik analisa representasi *intemediate* (seperti analisis distribusi, clustering, analisis tren, klasifikasi dan association rules).

Tahap yang terakhir adalah evaluasi, pada tahap ini dilakukan visualisasi dari hasil pemodelan.



Gambar 2. 7 Proses pencarian pengetahuan untuk text mining

2.2.24 TF IDF

Dalam *information retrieval*, tf-idf atau TFIDF, kependekan dari *term-frequency-inverse document frequency*, adalah statistik numerik yang dimaksudkan untuk mencerminkan betapa pentingnya sebuah kata untuk sebuah dokumen dalam kumpulan atau kumpulan. (Rajaraman, Mining of Massive Datasets) Hal ini sering digunakan sebagai faktor pembobotan dalam pencarian pengambilan informasi, penambahan teks dan pemodelan pengguna. Nilai tf-idf meningkat secara proporsional untuk berapa kali sebuah kata muncul dalam dokumen dan diimbangi dengan jumlah dokumen dalam korpus yang mengandung kata, yang membantu untuk menyesuaikan fakta bahwa beberapa kata muncul lebih sering secara umum. TF – IDF adalah salah satu skema pembobotan term yang paling populer saat ini; 83% dari sistem rekomendasi berbasis teks di perpustakaan digital menggunakan tf-idf. (Breitinger, 2016) Variasi skema pembobotan tf-idf sering digunakan oleh mesin pencari sebagai alat utama dalam penilaian dan peringkat relevansi dokumen yang diberikan permintaan pengguna. TF – IDF dapat berhasil digunakan untuk memfilter kata-berhenti di berbagai bidang subjek, termasuk peringkasan dan klasifikasi teks. Salah satu fungsi peringkat paling sederhana dihitung dengan menjumlahkan tf-idf untuk setiap istilah permintaan; banyak fungsi peringkat yang lebih canggih adalah varian dari model sederhana ini.

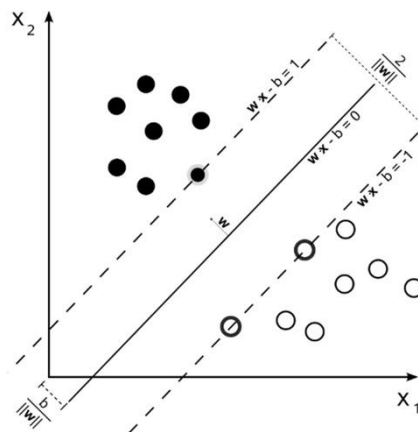
2.2.25 Chi Square

Dalam teori probabilitas dan statistika, distribusi khi-kuadrat (bahasa Inggris: Chi-square distribution) atau distribusi χ^2 dengan k derajat bebas adalah distribusi jumlah kuadrat k peubah acak normal baku yang saling bebas. Distribusi ini seringkali digunakan dalam statistika inferensial, seperti dalam uji hipotesis, atau dalam penyusunan selang kepercayaan. Apabila dibandingkan dengan distribusi khi-kuadrat nonsentral, distribusi ini dapat juga disebut distribusi khi-kuadrat sentral. Salah satu penggunaan distribusi ini adalah uji khi-kuadrat untuk kebersesuaian (goodness of fit) suatu distribusi pengamatan dengan distribusi teoretis, kriteria klasifikasi analisis data yang saling bebas, serta pendugaan selang kepercayaan untuk simpangan baku populasi berdistribusi normal dari simpangan baku sampel.

Sejumlah pengujian statistika juga menggunakan distribusi ini, seperti Uji Friedman. Distribusi khi-kuadrat merupakan kasus khusus distribusi gamma. (Wikipedia)

2.2.26 Support Vector Machine (SVM)

Support Vector Machine merupakan *supervised learning model* yang digunakan untuk melakukan analisa data dalam proses klasifikasi atau *regeresi linear*. Pada proses klasifikasi metode pembelajaran ini akan memetakan data pada suatu bidang dan membedakan kategori yang terpisah secara jelas dengan jarak yang selebar-lebarnya. Kategori yang berbeda tersebut dipisahkan oleh *hyperplane*. Berikut adalah gambaran klasifikasi menggunakan svm.



Gambar 2. 8 Gambaran klasifikasi menggunakan support vector machine

Hyperplane merupakan sebuah garis atau bidang yang memisahkan 2 buah kategori yang berbeda. Sebuah hyperplane dapat kita cari dengan menggunakan rumus sebagai berikut :

$$w \cdot x_i - b = 0 \quad \dots(1)$$

w pada persamaan diatas merupakan vektor yang menentukan bobot. Sementara itu x_i merupakan vektor sampel data dan b merupakan bias. Pada klasifikasi biner, hasil klasifikasi akan menghasilkan true (+1) dan false (-1). Untuk klasifikasi tersebut berlaku rumus sebagai berikut:

$$f(x) = \begin{cases} +1, & \text{jika } w \cdot x_i - b \geq 0 \\ -1, & \text{jika } w \cdot x_i - b \leq 0 \end{cases} \dots(2)$$

Namun, agar hasil klasifikasi menjadi lebih baik kita perlu memaksimalkan nilai margin. Margin sendiri merupakan jarak antara titik-titik positif dan negatif terdekat di sekitar hyperplane. titik-titik positif dan negatif terdekat di sekitar hyperplane ini biasa disebut dengan support vector. Untuk menentukan margin dapat digunakan persamaan berikut :

$$f(x) = \begin{cases} +1, & \text{jika } w \cdot x_i - b \geq 1 \\ -1, & \text{jika } w \cdot x_i - b \leq -1 \end{cases} \dots(3)$$

Pada persamaan di atas batas atas dan batas bawah tidak lagi 0 seperti persamaan. Hal ini dikarenakan $-1 < w \cdot x_i - b < 1$ menyatakan margin. Persamaan dapat disederhanakan menjadi :

$$y_i(w \cdot x_i - b) \geq 1 \dots(4)$$

2.2.27 Spam Prevention menggunakan Support Vector Machine (SVM)

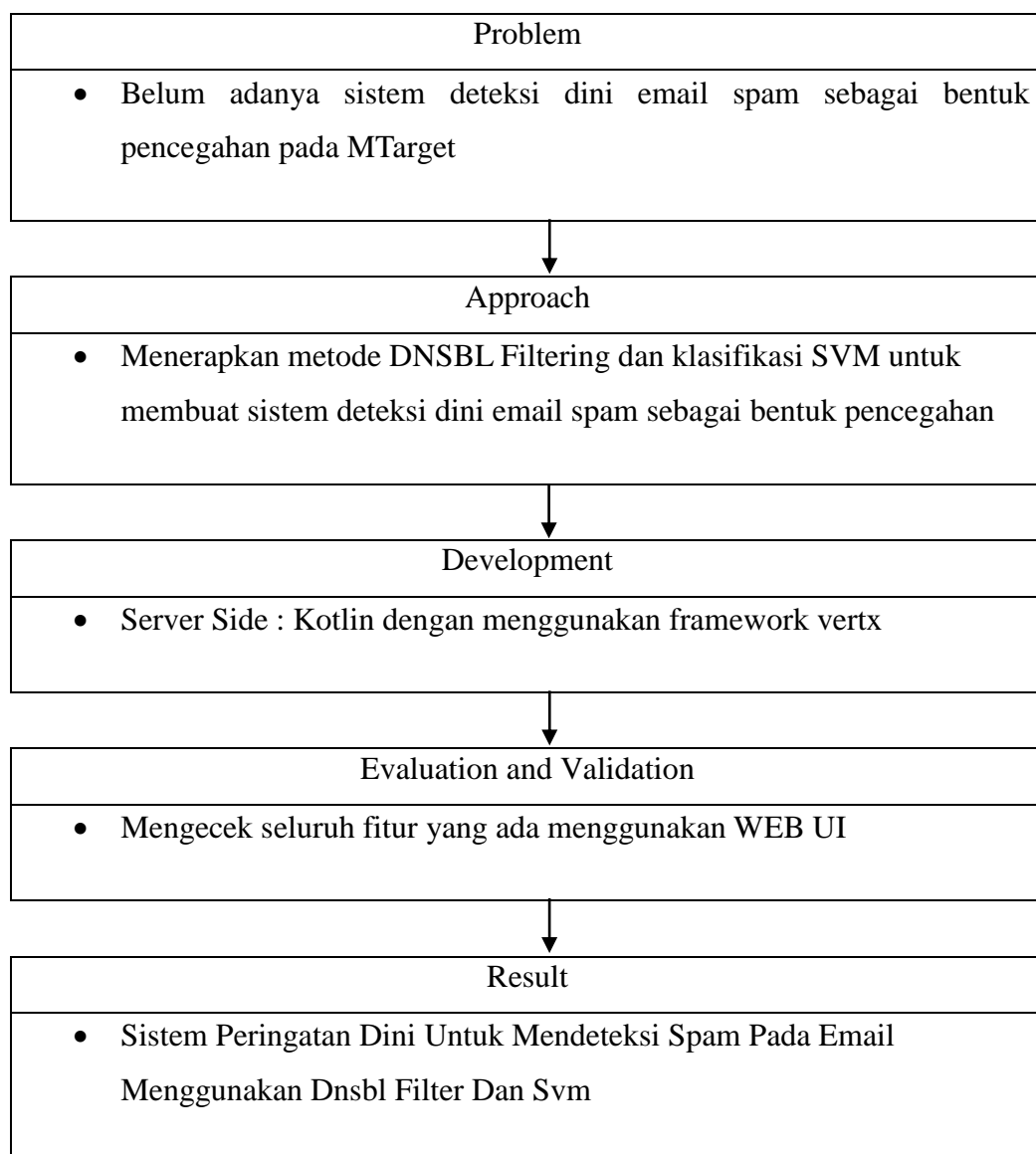
Karthika mengusulkan teknik klasifikasi spam email menggunakan *Latent Semantic Indexing Based SVM Model*. Pada awalnya, dataset input diberikan pada langkah pra-pemrosesan yang menghilangkan *stop word* dan tanda baca sehingga kata kunci yang lebih relevan diperoleh. Kata kunci yang diekstraksi kemudian diberikan kepada ekstraksi fitur di mana, *Term Frequency (TF)* dan *Invers Document Frequency (IDF)* memproses kata kunci yang diambil dari langkah-langkah pra-pemrosesan. Setelah pembentukan matriks fitur, dimensi yang sesuai untuk klasifikasi yang lebih baik ditemukan menggunakan model LSI yang memetakan ruang fitur ke ruang LSI menggunakan analisis berbasis korelasi. Akhirnya, ruang LSI diberikan ke algoritma SVM yang melatih berdasarkan pola yang diberikan dalam ruang pelatihan LSI. Dalam fase pengujian, email masukan yang diwakili dalam ruang LSI diklasifikasikan sebagai Spam atau Ham berdasarkan *hyperplane*

optimal yang dihasilkan dalam pelatihan SVM. Dengan menggunakan langkah-langkah ini, klasifikasi spam dan email ham telah dilakukan secara efektif.

2.3 Kerangka Pemikiran

Berdasarkan teori yang telah dipaparkan sebelumnya, penelitian akan dilakukan dengan kerangka pemikiran sebagai berikut :

Tabel 2.0-3Kerangka Pemikiran



BAB III

METODE PENELITIAN

Metode penelitian yang digunakan dalam penulisan Tugas Akhir ini, sebagai berikut :

3.1 Jenis Dan Sumber Data

Penulis telah mengumpulkan beberapa jenis data sebagai acuan penelitian. Data tersebut termasuk dalam data sekunder, yaitu data yang dijadikan landasan teori dan penunjang yang diperoleh peneliti dari sumber yang sudah ada. Data sekunder didapatkan dari studi literatur dan dokumen penelitian terkait sebelumnya. Studi pustaka dilakukan untuk mencari 2 jenis data yaitu dataset email spam & ham dan daftar DNSBL.

3.2 Teknik Pengumpulan Data

Teknik pengumpulan data yang digunakan dalam penelitian ini adalah Studi Pustaka. Dataset email spam & ham diambil dari Enron-Spam Dataset, sedangkan daftar DNSBL diambil dari hasil *web scrapping* dari website https://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists.

3.3 Metode Pengembangan Sistem

Pembangunan sebuah sistem yang baik tidaklah terlepas dari sebuah metode pengembangan sistem yang sesuai dengan kebutuhan sistem tersebut. Dalam menyelesaikan penelitian tentang Sistem Peringatan Dini Untuk Mendeteksi Spam Pada Email, penulis menggunakan metode Scrum.

Scrum adalah *agile framework* untuk mengelola pekerjaan pengetahuan, terutama dalam pengembangan perangkat lunak. (Wykowski, 2018) Scrum dirancang untuk tim yang terdiri dari tiga hingga sembilan anggota, yang memecah pekerjaan mereka menjadi tindakan yang dapat diselesaikan dalam iterasi timebox, disebut sprint, tidak lebih dari satu bulan dan paling sering dua minggu, kemudian melacak

kemajuan dan rencana ulang dalam 15 menit pertemuan, disebut scrum harian. (Schwaber, Agile Project Management with Scrum)

Scrum adalah kerangka kerja yang ringan, berulang, dan bertahap untuk mengelola pengembangan produk. (What is Scrum? An Agile Framework for Completing Complex Projects - Scrum Alliance.) (Verheyen) Scrum mendefinisikan "strategi pengembangan produk yang fleksibel dan holistik di mana tim bekerja sebagai unit untuk mencapai tujuan bersama", (Takeuchi, 2010) untuk pengembangan produk, dan memungkinkan tim untuk mandiri serta komunikasi tatap muka setiap hari di antara semua anggota tim dan disiplin ilmu yang terlibat.

Prinsip utama Scrum adalah pengakuan bahwa pelanggan akan berubah pikiran tentang apa yang mereka inginkan atau butuhkan (sering disebut volatilitas persyaratan (Henry, 1993)) dan bahwa akan ada tantangan yang tidak dapat diprediksi — di mana pendekatan prediktif atau terencana tidak cocok. Dengan demikian, Scrum mengadopsi pendekatan empiris berbasis bukti — menerima bahwa masalahnya tidak dapat sepenuhnya dipahami atau didefinisikan di muka, dan sebaliknya berfokus pada bagaimana memaksimalkan kemampuan tim untuk menyampaikan dengan cepat, untuk menanggapi kebutuhan yang muncul, dan untuk beradaptasi dengan berkembang teknologi dan perubahan kondisi pasar. Ada beberapa langkah dalam mengembangkan sistem dengan menggunakan metode ini, berikut adalah macamnya:

3.3.1. Sprint

Sprint (atau iterasi) adalah unit dasar pengembangan di Scrum. Sprint adalah upaya *timeboxed* yang dibatasi pada durasi tertentu. Durasi ditetapkan di muka untuk setiap sprint dan biasanya antara satu minggu dan satu bulan, dengan dua minggu yang paling umum. (Schwaber, Agile Project Management with Scrum, 2004)

Setiap sprint dimulai dengan acara perencanaan sprint yang bertujuan untuk mendefinisikan tumpukan sprint, mengidentifikasi pekerjaan untuk sprint, dan membuat perkiraan perkiraan untuk tujuan sprint. Setiap sprint berakhir dengan ulasan sprint dan sprint retrospektif, (Sutherland, 2004) yang meninjau kemajuan untuk di-

tunjukkan kepada para pemangku kepentingan dan mengidentifikasi pelajaran dan perbaikan untuk sprint berikutnya.

Scrum menekankan produk yang bekerja pada akhir sprint yang benar-benar dilakukan. Dalam hal perangkat lunak, ini kemungkinan termasuk bahwa perangkat lunak telah sepenuhnya terintegrasi, diuji dan didokumentasikan, dan berpotensi dapat dirilis. (Deemer, 2012)

3.3.2. Sprint Planning

Pada awal sprint, tim scrum mengadakan acara perencanaan sprint (Arif) untuk:

- Saling berdiskusi dan sepakat tentang ruang lingkup pekerjaan yang dimaksudkan untuk dilakukan selama sprint itu
- Pilih item jaminan produk yang dapat diselesaikan dalam satu sprint
- Mempersiapkan sprint backlog yang mencakup pekerjaan yang diperlukan untuk menyelesaikan item backlog produk yang dipilih
- Durasi yang disarankan adalah empat jam untuk sprint dua minggu (pro-rata untuk durasi sprint lainnya)
 - Selama babak pertama, seluruh tim scrum (tim pengembangan, master scrum, dan pemilik produk) memilih item jaminan produk yang mereka yakini dapat diselesaikan dalam sprint tersebut.
 - Selama babak kedua, tim pengembangan mengidentifikasi pekerjaan terperinci (tugas) yang diperlukan untuk menyelesaikan item jaminan produk tersebut; menghasilkan backlog sprint yang dikonfirmasi
 - Karena pekerjaan yang terperinci diuraikan, beberapa item jaminan simpanan produk dapat dipecah atau dimasukkan ke dalam jaminan simpanan produk jika tim tidak lagi percaya bahwa mereka dapat menyelesaikan pekerjaan yang diperlukan dalam satu sprint.
- Setelah tim pengembang telah menyiapkan sprint backlog mereka, mereka memperkirakan (biasanya dengan memilih) tugas yang akan diberikan dalam sprint.

3.3.3. Daily Scrum

Setiap hari selama sprint, tim memegang scrum harian (atau stand-up) dengan pedoman khusus:

- Semua anggota tim pengembangan siap. Scrum harian:
 - Dimulai tepat waktu bahkan jika beberapa anggota tim pengembangan hilang
 - Harus terjadi pada waktu dan tempat yang sama setiap hari
 - Terbatas (timebox) hingga lima belas menit
- Siapa pun boleh, meskipun hanya anggota tim pengembangan yang harus berkontribusi.
- Selama scrum harian, setiap anggota tim biasanya menjawab tiga pertanyaan:
 - Apa yang saya selesaikan kemarin yang berkontribusi pada tim yang memenuhi tujuan sprint kami?
 - Apa yang saya rencanakan untuk selesaikan hari ini untuk berkontribusi pada tim yang memenuhi tujuan sprint kami?
 - Apakah saya melihat ada halangan yang dapat mencegah saya atau tim untuk mencapai tujuan sprint kami?

Setiap hambatan (misalnya, batu sandungan, risiko, masalah, ketergantungan tertunda, asumsi terbukti tidak berdasar) diidentifikasi dalam scrum harian harus ditangkap oleh master scrum dan ditampilkan pada papan scrum tim atau pada papan risiko bersama, dengan orang yang setuju ditunjuk untuk bekerja menuju resolusi (di luar scrum harian). Tidak ada diskusi terperinci yang harus dilakukan selama scrum harian.

3.3.4. Sprint Review

Di akhir sprint, tim mengadakan dua acara: sprint review dan sprint retrospective.

Pada ulasan sprint, tim:

- Meninjau pekerjaan yang selesai dan pekerjaan yang direncanakan yang tidak selesai
- Menyajikan pekerjaan yang telah diselesaikan kepada para pemangku kepentingan (mis. demo)
- Bekerja sama dengan para pemangku kepentingan tentang apa yang harus dikerjakan selanjutnya

Pedoman untuk ulasan sprint:

- Pekerjaan yang tidak lengkap tidak dapat ditunjukkan.
- Durasi yang disarankan adalah dua jam untuk sprint dua minggu (sebanding dengan sprint-durasi lainnya).

3.3.5. Sprint Retrospective

Di sprint retrospective, tim:

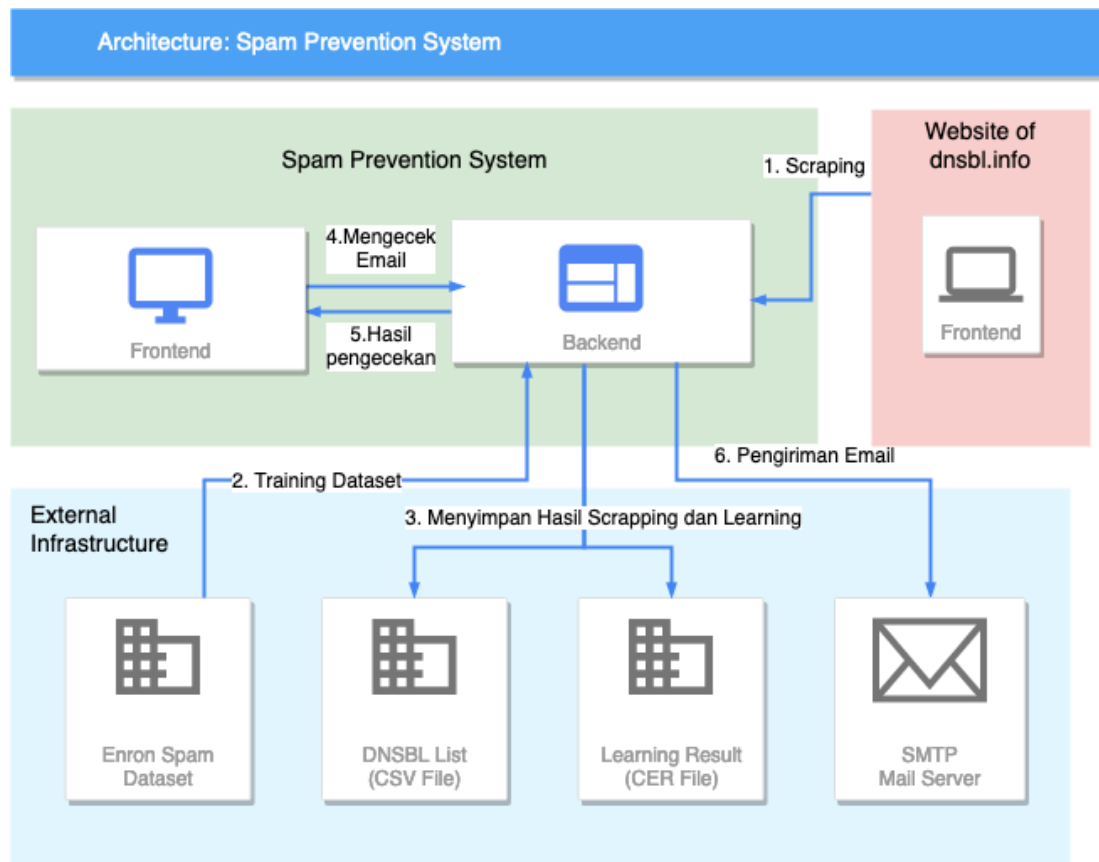
- Merefleksikan pada sprint terakhir
- Mengidentifikasi dan menyetujui tindakan peningkatan proses yang berkelanjutan

Pedoman untuk retrospektif sprint:

- Tiga pertanyaan utama diajukan dalam sprint retrospective: Apa yang berjalan dengan baik selama sprint? Apa yang tidak beres? Apa yang bisa ditingkatkan untuk produktivitas yang lebih baik di sprint berikutnya?
- Durasi yang disarankan adalah satu setengah jam untuk sprint dua minggu (sebanding dengan durasi sprint lainnya)
- Acara ini difasilitasi oleh master scrum

3.4 Metode yang Diusulkan

Rancangan sistem peringatan dini untuk mendeteksi spam pada email yang lama kurang lebih seperti berikut:



Gambar 3.1 Kerangka Sistem Peringatan Dini Deteksi Spam Pada Email

Alur dari Sistem Peringatan Dini Untuk Mendeteksi Spam Pada Email sebagaimana dilihat pada gambar 3.1 diatas dijelaskan sebagai berikut:

a. Proses scrapping DNSBL

Proses ini dilakukan untuk mengambil data DNSBL dengan melakukan scrapping pada laman

https://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists, dijelaskan sebagai berikut :

1. Backend melakukan HTTP Request ke laman https://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists mendapatkan laman tersebut dalam format HTML.
2. Backend melakukan ekstraksi HTML tersebut menghasilkan daftar DNSBL.
3. Daftar DNSBL kemudian disimpan kedalam file berekstensi csv.

b. Proses trainingdataset email spam dan ham

Proses ini dilakukan untuk membuat pemodelan menggunakan metode Support Vector Machine, dijelaskan sebagai berikut :

1. Backend membaca setiap file dari folder dataset email, ham & spam.
2. Backend mengubah dataset menjadi fitur data menggunakan DF.
3. Backend melakukan feature selection menggunakan Chi Square.
4. Backend melakukan pembobotan kata menggunakan TF IDF.
5. Backend melakukan pemodelan menggunakan Support Vector Machine.
6. Hasil training / model kemudian disimpan dalam file berekstensi crt.

c. Proses pengecekan email

Proses ini dilakukan untuk menentukan apakah email termasuk spam atau tidak, dijelaskan sebagai berikut :

1. User mengirim email dari aplikasi Frontend Web.
2. Backend membaca domain dari alamat email pengirim.
3. Backend melakukan pengecekan menggunakan DNSBL Filter, apabila terdeteksi sebagai spam maka akan ditampilkan pesan ke aplikasi Frontend Web, jika tidak lanjut ke tahap berikutnya.
4. Backend membaca konten email.
5. Backend melakukan text mining dengan melakukan stemming, tokenisasi dan stop word removal pada konten email.
6. Backend melakukan prediksi pada hasil text mining tersebut.

7. Backend melakukan prediksi menggunakan SVM Filter apabila terdeteksi sebagai spam maka akan ditampilkan pesan ke aplikasi Frontend Web, jika tidak lanjut ke tahap berikutnya

d. Proses pengambilan keputusan terhadap hasil pengecekan email

Setelah proses pengecekan email selesai, backend akan mengeluarkan dokumen yang berisi hasil pengecekan. Dari hasil tersebut user dapat menentukan apakah email tersebut layak untuk di kirim atau direvisi.

3.5 Metode Evaluasi

Evaluasi dilakukan untuk mengetahui apakah aplikasi yang telah dibangun dapat berjalan dengan baik dan memenuhi spesifikasi yang telah ditentukan. Dalam penelitian ini penulis melakukan evaluasi dengan menggunakan metode *white-box testing*. Untuk melakukan pengujian *white-box* penulis menulis kode unit testing. White-box testing dilakukan untuk memastikan setiap komponen berjalan sesuai tujuan awal. Pengujian ini dilakukan pada bagian fungsi-fungsi utama. Pengujian dilakukan dengan memasukan inputan ke setiap *endpoint* dan dibandingkan dengan keluaran yang dibutuhkan. Beberapa komponen yang akan diuji adalah fungsi pengecekan domain, fungsi scrapping, penambahan, pengurangan & penambian dnsbl, training dataset, pengecekan konten spam dan mengirim email. White-box testing dilakukan menggunakan JUnit, *unit testing framework* untuk bahasa pemrograman yang menggunakan *Java Virtual Machine (JVM)* seperti Java dan Kotlin. Setelah semua sudah sesuai maka Backend API yang sudah dibangun siap untuk diluncurkan.

Selain menggunakan *white-box testing* penulis juga akan menggunakan *black-box testing* pada penelitian kali ini. *Black-Box Testing* merupakan bentuk pengecekan sistem berdasarkan spesifikasi kebutuhan dari sistem itu sendiri dan tanpa melakukan pengecekan kode. *Black-Box Testing* murni melakukan pengecekan berdasarkan dari tampilan pengguna (Nidhra, 2012). Blackbox digunakan untuk memastikan integrasi antar komponen dapat berjalan dengan baik.

BAB IV

RANCANGAN SISTEM DAN IMPLEMENTASI

4.1 Pengantar

Pada pembahasan kali ini penulis akan menjelaskan tentang rancangan sistem dan memaparkan hasil implementasi dari metode yang diusulkan untuk menyelesaikan masalah yang telah dibahas. Tujuan dari pembahasan ini ialah untuk membuktikan DNSBL dan Support Vector Machine dapat digunakan untuk menyaring spam pada email sebagai bentuk pencegahan.

4.2 Analisa Kebutuhan

4.2.1 Analisis Kebutuhan Data

Pada penelitian ini diperlukan beberapa data yang berupa daftar DNSBL. Untuk mendapatkan data tersebut peneliti melakukan pengumpulan data secara manual. Selain itu peneliti menyiapkan *scraper* yang dapat mengumpulkan DNSBL dari laman https://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists secara otomatis. Dari proses scrapping tersebut didapatkan DNSBL sebanyak 57 buah.

Selain itu peneliti juga memerlukan data berupa dataset email spam yang telah diberi label spam & ham. Untuk mendapatkan data tersebut peneliti menggunakan public dataset yang ada di internet yaitu Enron Spam Dataset. Dataset tersebut berisi 2553 buah email spam dan 1434 buah email ham.

4.2.2 Analisis Kebutuhan Pengguna

Pada kebutuhan pengguna, kebutuhan dibedakan menjadi 2 jenis yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Berikut adalah gambaran mengenai kebutuhan dari pengguna.

Tabel 4. 1 Tabel kebutuhan pengguna

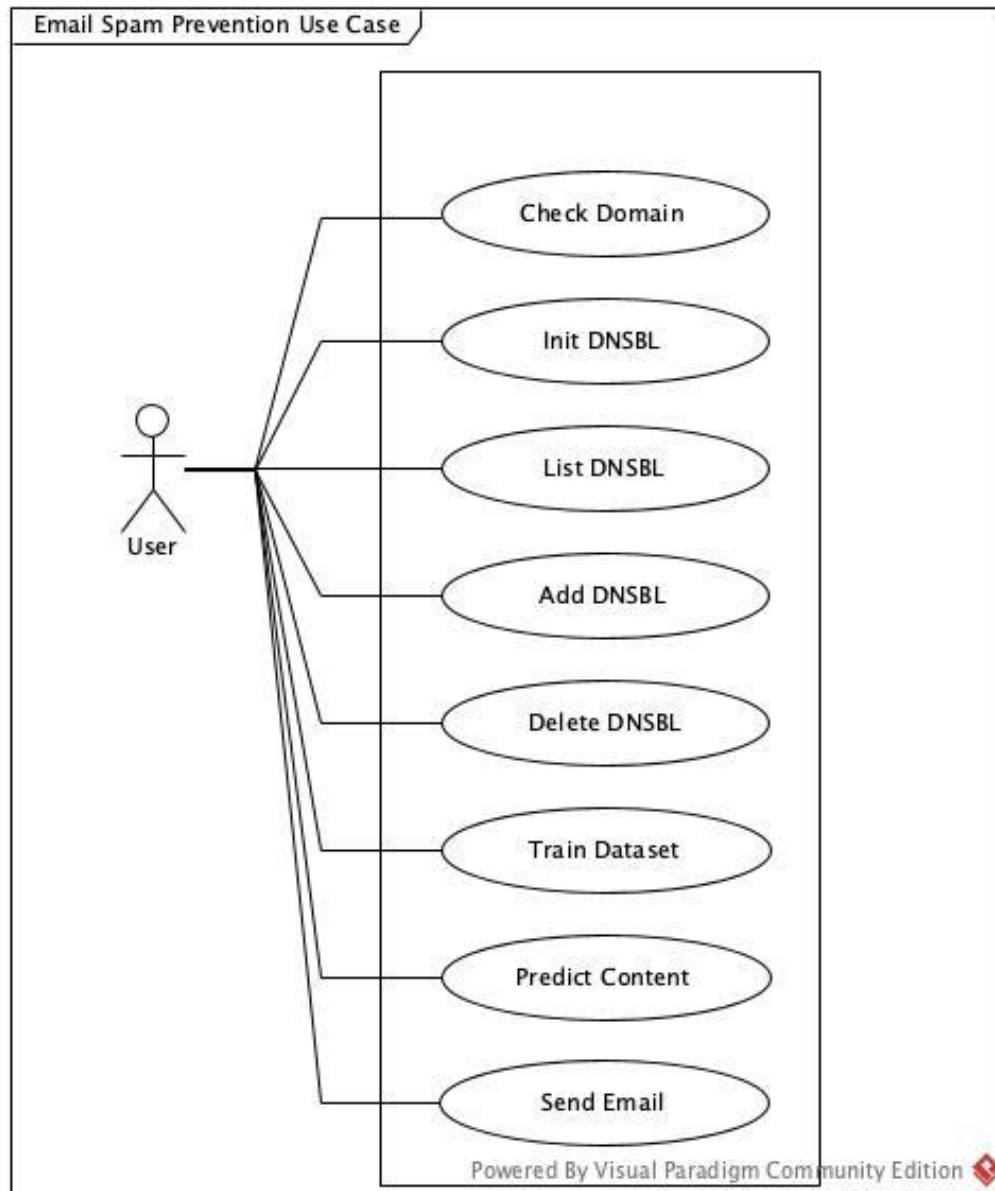
Tipe Kebutuhan	Kebutuhan
Fungsional	<ol style="list-style-type: none">1. Pengguna dapat menambahkan dataset2. Pengguna dapat menjalankan proses pembelajaran3. Pengguna dapat memilih model yang akan digunakan4. Pengguna dapat melakukan proses evaluasi terhadap model yang dihasilkan5. Sistem dapat menampilkan peringatan jika email terindikasi mengandung spam
Non-fungsional	Ergonomy – sistem memiliki antarmuka yang mudah digunakan

4.3 Desain Sistem

Pada bagian desain, proses desain digambarkan dengan menggunakan UML (Unified Modelling Language) seperti Use Case Diagram, Class Diagram dan Sequence Diagram.

4.2.1 Use Case Diagram

Berikut merupakan use case diagram dari sistem peringatan dini untuk mendeteksi spam pada email



Gambar 4.1 Use Case Diagram

Gambar diatas merupakan rancangan *Use Case* dari *Email Spam Prevention System*, menunjukan bahwa terdapat 1 aktor yang terlibat, yaitu User. User mampu melakukan pengecekan domain (use case Check Domain), menginisialisasi daftar DNSBL dengan menggunakan metode webscrapping (use case Init DNSBL), melihat daftar DNSBL (use case List DNSBL), menambah DNSBL (use case Add DNSBL), menghapus DNSBL (use case Delete DNSBL), melakukan training pada dataset (use case Train Dataset), memprediksi apakah konten email

4.2.2 Class Diagram



49

1. Class Diagram DNSBL

Class Diagram ini berfungsi sebagai representasi meta data untuk DNSBL. Pada Class Diagram tersebut menunjukkan bahwa terdapat 1 variabel yaitu :

Tabel 4. 2 Tabel DNSBL

No	Nama Field	Tipe Data	Keterangan
1.	Name	string	url DNSBL

Selain variable tersebut terdapat juga beberapa fungsi yaitu :

- Fungsi untuk mengecek domain apakah tidak tercantum pada DNSBL dengan memasukkan nama domain dan dengan atau tidak mencantumkan list dnsbl
- Fungsi untuk menginisialisasi data DNSBL dengan melakukan scrapping pada laman https://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists.
- Fungsi untuk menambah, menghapus dan melihat data

2. Class Diagram LinearSVM

Class Diagram “SVMLinear” berfungsi sebagai representasi meta data untuk metode Support Vector Machine yang menggunakan kernel linear. Pada Class Diagram tersebut menunjukkan bahwa terdapat beberapa variabel yaitu :

Tabel 4. 3 Linear SVM

No	Nama Field	Tipe Data	Keterangan
1.	datasetCount	Integer	Jumlah dataset yang digunakan
2.	categoryCount	Integer	Jumlah kategori
3.	featureCount	Integer	Jumlah fitur
4.	Weight	IFeatureWeighter	Interface untuk menghitung berat fitur
5.	svmModel	Model	Model SVM secara general

3. Class Diagram IFeatureWeighter

Class Diagram “IFeatureWeighter” berfungsi untuk mengukur beban tiap fitur. Dalam kasus ini, IFeatureWeighter akan mengimplementasikan class diagram TfidfFeatureWeighter.

4. Class Diagram TfidfFeatureWeighter

Class Diagram “TfidfFeatureWeighter” berfungsi untuk mengukur beban tiap fitur menggunakan metode TF-IDF. Pada Class Diagram tersebut menunjukkan bahwa terdapat beberapa variabel yaitu :

Tabel 4. 4 TfidfFeatureWeighter

No	Nama Field	Tipe Data	Keterangan
1.	numDocs	Int	Jumlah dokumen
2.	Df	int[]	Frekuensi dokumen

5. Class Diagram Model

Class Diagram “Model” berfungsi sebagai representasi meta data untuk Model Machine Learning secara umum. Pada Class Diagram tersebut menunjukkan bahwa terdapat beberapa variabel yaitu :

Tabel 4. 5 Tabel Model

No	Nama Field	Tipe Data	Keterangan
1.	Bias	Float	Bias
2.	Label	int[]	Label untuk tiap class
3.	nr_class	Int	Jumlah class
4.	nr_feature	Int	Jumlah fitur
5.	solverType	SolverType	Tipe Solver
6.	W	double[]	Berat fitur dalam array

6. Class Diagram SolverType

Class Diagram “SolverType” berfungsi sebagai representasi dari tipe tipe solver dan kapasitasnya sebagai LogisticRegression Solver dan SupportVectorRegression. Pada Class Diagram tersebut menunjukkan bahwa terdapat beberapa variabel yaitu :

Tabel 4. 6 Tabel SolverType

No	Nama Field	Tipe Data	Keterangan
1.	Id	Int	Id solver type
2.	logisticRegressionSolver	boolean	Kapasitas sebagai LogisticRegression Solver
3.	supportVectorRegression	boolean	Kapasitas sebagai SupportVectorRegression

7. Class Diagram Abstract Model

Class Diagram “AbstractModel” berfungsi sebagai representasi dari model untuk semua model klasifikasi teks. Pada Class Diagram tersebut menunjukkan bahwa terdapat beberapa variabel yaitu :

Tabel 4. 7 Tabel AbstractModel

No	Nama Field	Tipe Data	Keterangan
1.	Catalog	String[]	Tabel kategori
2.	Tokenizer	ITokenizer	Interface untuk Segmentasi kata
3.	wordIdTrie	BinTrie<int>	Word-to-map

8. Class Diagram ITokenizer

Class Diagram “ITokenizer” berfungsi untuk melakukan tokenisasi. Metode tokenisasi yang dilakukan sendiri yaitu Bigram Tokenizer, Blank Tokenizer dan HanLPTokenizer.

9. Class Diagram Mail

Tabel 4. 8 Tabel Maile

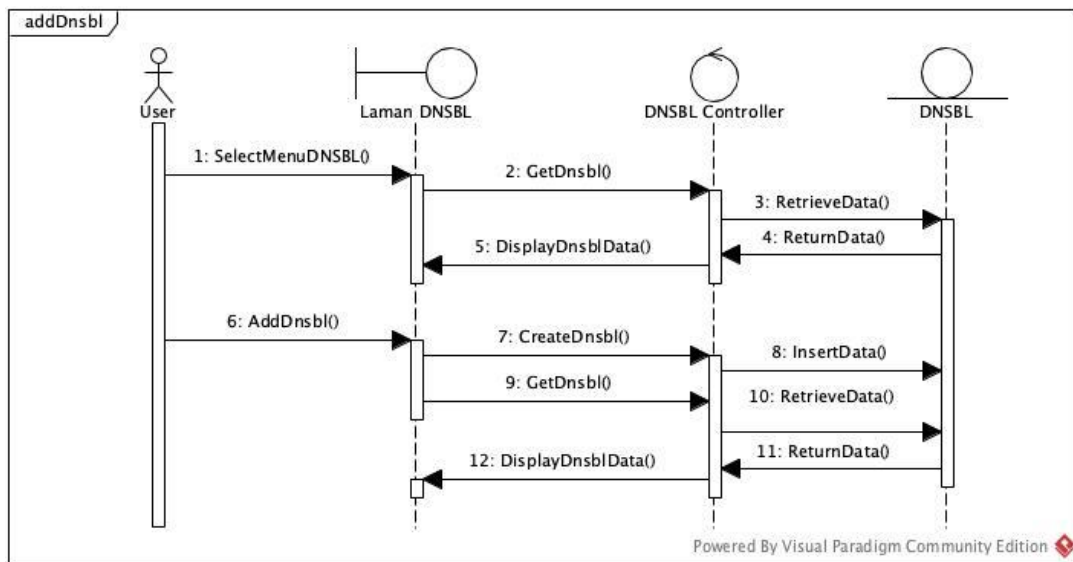
No	Nama Field	Tipe Data	Keterangan
1.	Host	String	Host mail server
2.	Username	String	Alamat email pengirim
3.	password	String	Password pengirim
4.	recipient	String[]	Daftart alamat email penerima
5.	subject	String	Subjek email
6.	body	String	Body email

Selain variable tersebut terdapat juga beberapa fungsi yaitu :

- Fungsi untuk mengirim email menggunakan Gmail smtp server.

4.2.3 Sequence Diagram

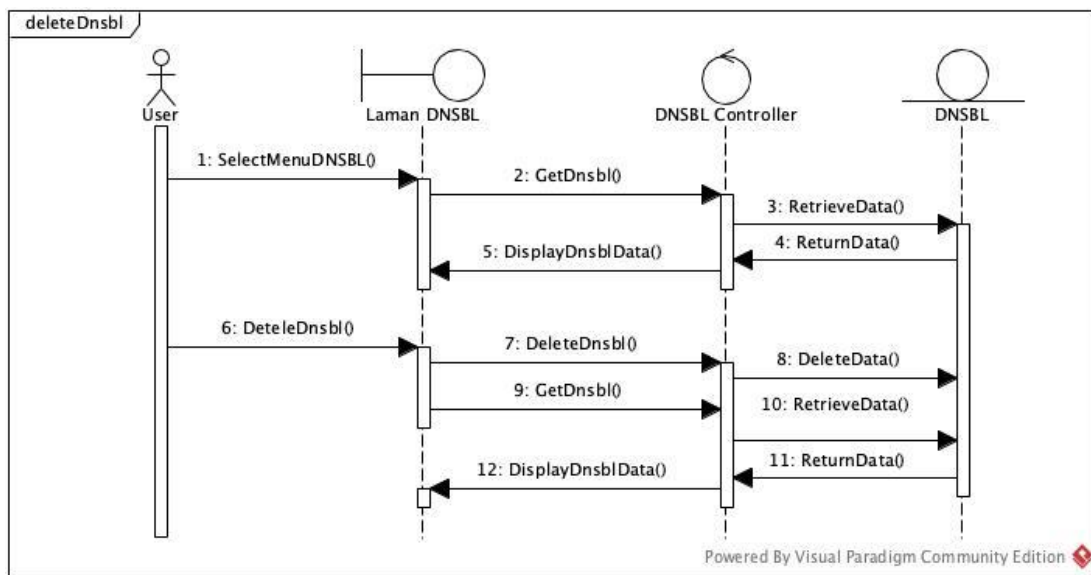
4.2.3.1 Add DNSBL



Gambar 4.3Sequence Diagram : Add DNSBL

Pada gambar diatas Sequence Diagram menunjukkan proses menambahkan data DNSBL dari web hingga data ditulis ke file. Pertama setelah user mengunjungi halaman DNSBL maka web akan mengambil data DNSBL dari database melalui controller dan menampilkanya. Kemudian user menambahkan DNSBL melalui menu dari web, dan diteruskan controller untuk ditulis ke file. Kemudian web akan memuat ulang tampilan dengan data yang terbaru.

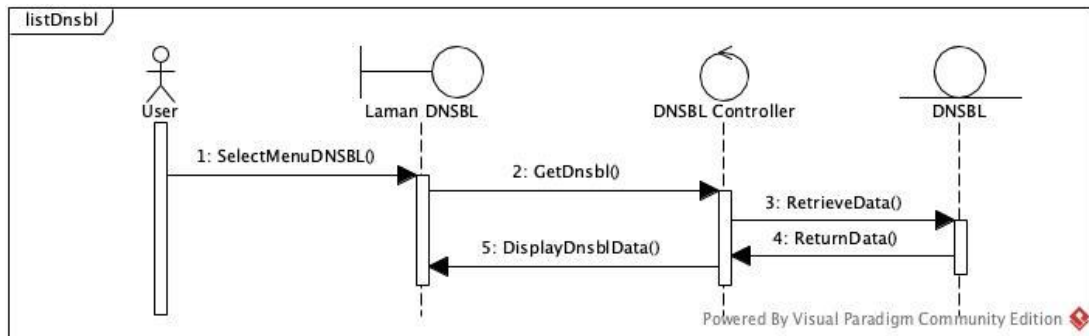
4.2.3.2 Delete DNSBL



Gambar 4.4Sequence Diagram : Delete DNSBL

Pada gambar diatas Sequence Diagram menunjukkan proses penghapusan data DNSBL dari web hingga dihapus dari file. Pertama setelah user mengunjungi halaman DNSBL maka web akan mengambil data DNSBL dari database melalui controller dan menampilkanya. Kemudian user menghapus DNSBL melalui menu dari web, dan diteruskan controller untuk dihapus dari file. Kemudian web akan memuat ulang tampilan dengan data yang terbaru.

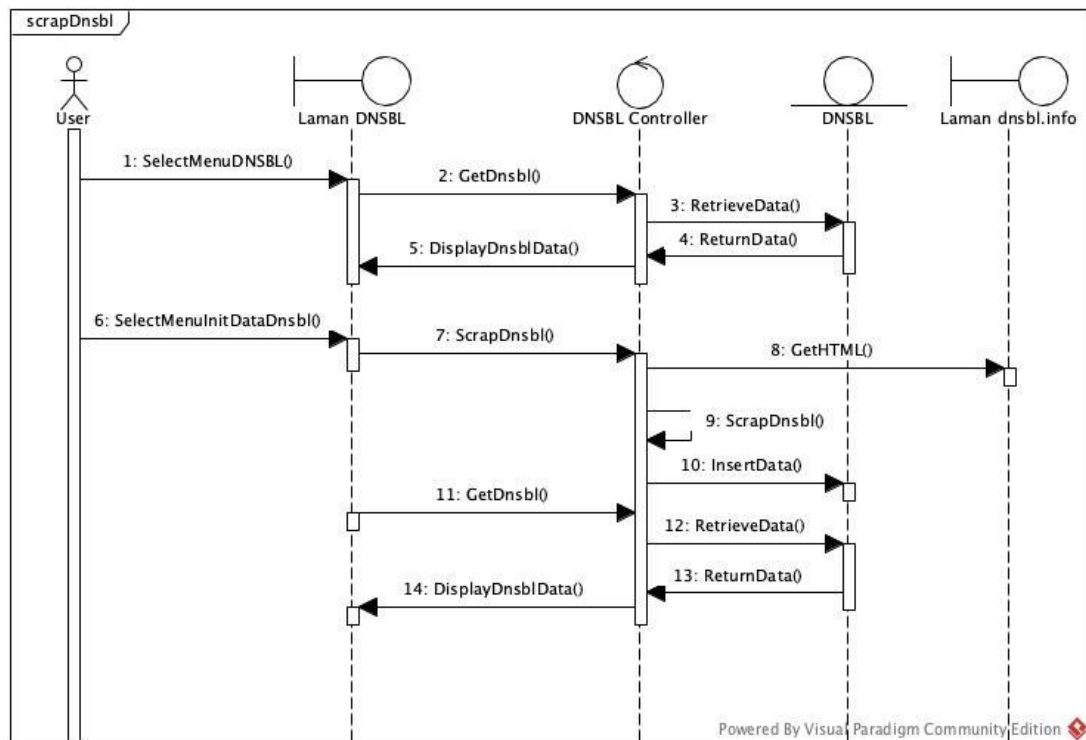
4.2.3.3 List DNSBL



Gambar 4.5 Sequence Diagram : List DNSBL

Pada gambar diatas Sequence Diagram menunjukkan proses pengambilan data DNSBL dari file hingga ditampilkan ke web. Setelah user mengunjungi halaman DNSBL maka web akan mengambil data DNSBL dari database melalui controller dan menampilkanya.

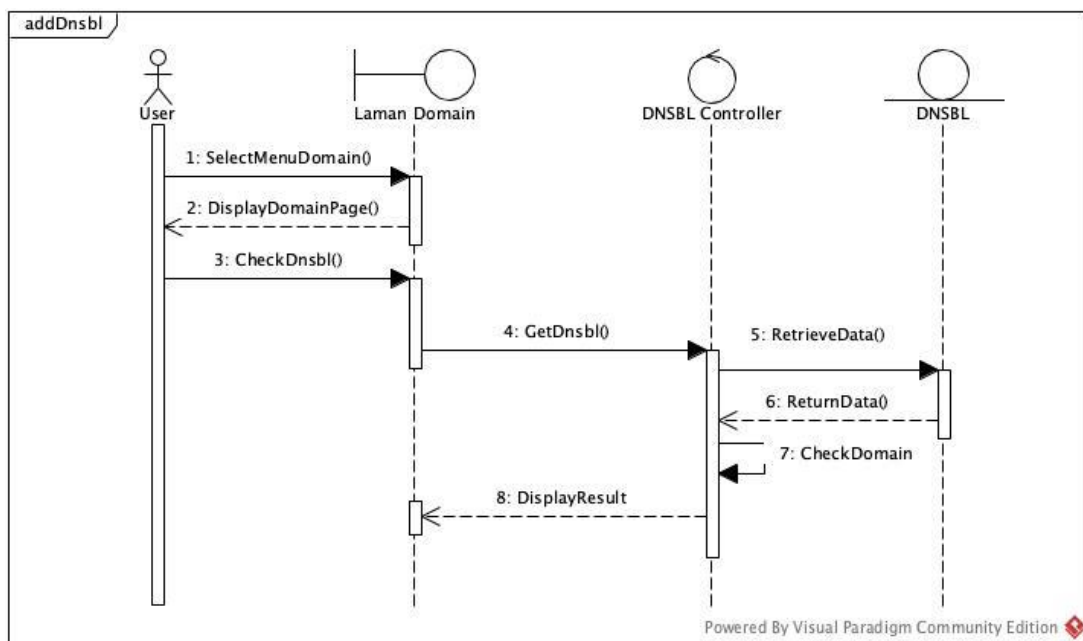
4.2.3.4 Init DNSBL



Gambar 4.6 Sequence Diagram : Init DNSBL

Pada gambar diatas Sequence Diagram menunjukkan proses inialisasi data data DNSBL dari file hingga ditampilkan ke web. Setelah user mengunjungi halaman DNSBL maka web akan mengambil data DNSBL dari database melalui controller dan menampilkanya. Kemudian user dapat menginisialisasi data DNSBL melalui menu dari web, yang kemudian perintahnya akan diteruskan ke controller. Controller melakukan http request ke https://en.wikipedia.org/wiki/Comparison_of_DNS_blacklists untuk mendapatkan laman. Kemudian laman tersebut di lakukan proses scrapping untuk mendapatkan data DNSBL. Data DNSBL itu kemudian di simpan kedalam file. Kemudian web akan memuat ulang tampilan dengan data yang terbaru.

4.2.3.5 Check Domain

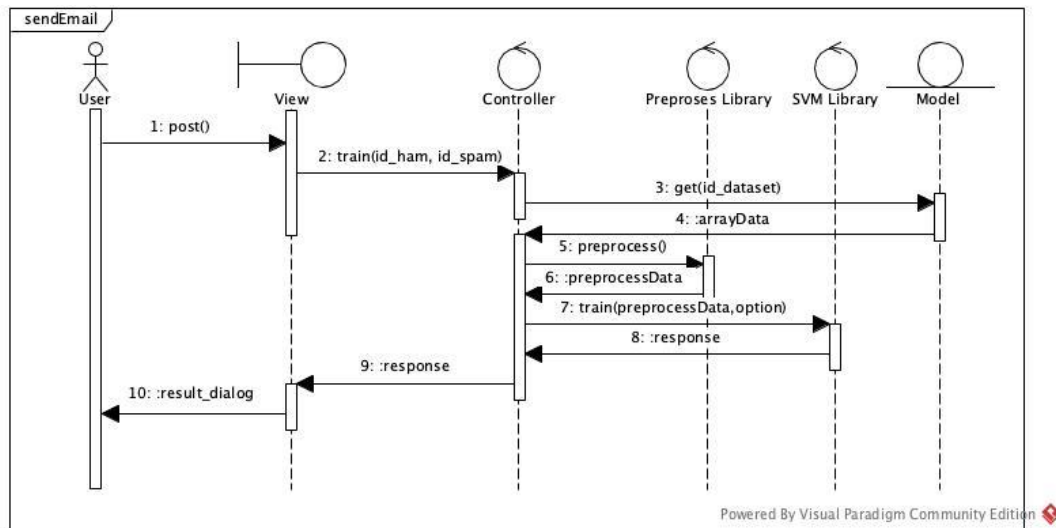


Gambar 4.7 Sequence Diagram : Check DNSBL

Pada gambar diatas Sequence Diagram menunjukkan proses pengecekan domain berbasis DNSBL melalui web. Ketika user mengunjungi halaman utama maka akan ditampilkan laman Domain Checker. Kemudian user dapat memasukkan alamat domain ke dalam menu, yang kemudian perintahnya akan diteruskan ke controller. Controller mengambil data DNSBL dari file dimana di tiap domain tersebut dilakukan pengecekan domain berbasis DNSBL sesuai domain yang di inputkan. Untuk

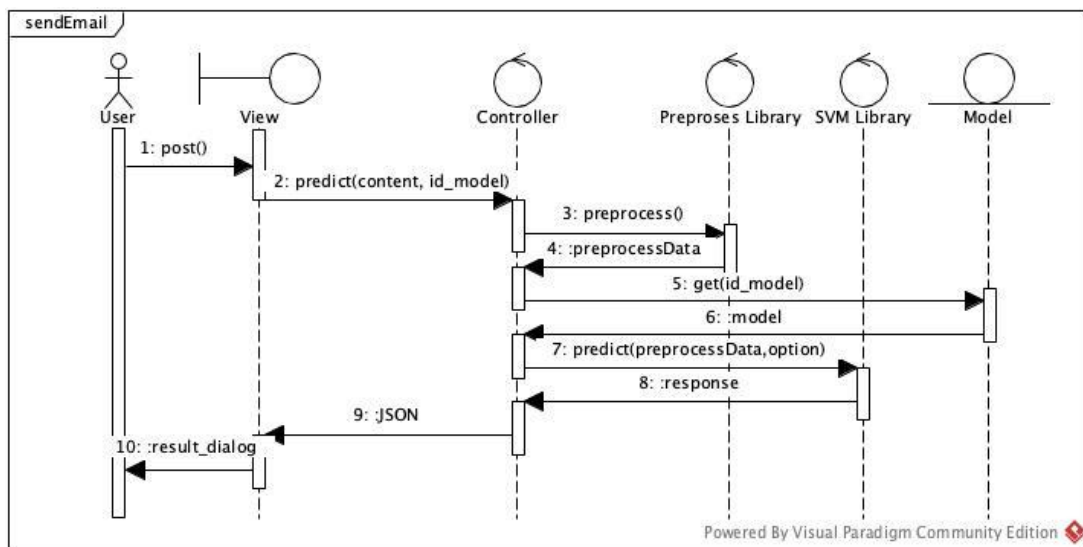
tiap DNSBL yang memblokir domain tersebut maka akan ditampilkan untuk ditampilkan ke web apabila proses pengecekan untuk semua DNSBL selesai.

4.2.3.6 Train Dataset



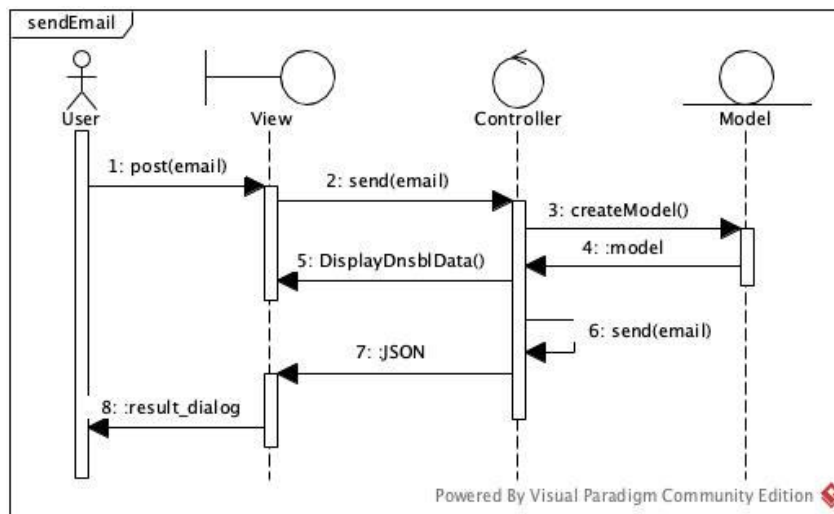
Gambar 4.8 Sequence Diagram : Train Dataset

4.2.3.7 Predict Content



Gambar 4.9 Sequence Diagram : Predict Content

4.2.3.8 Send Email



Gambar 4.10 Sequence Diagram : Send Email

4.4 Desain Antarmuka

Untuk memberikan gambaran serta konsep tentang antarmuka yang akan didapat oleh pengguna, Berikut adalah desain antarmuka yang terdiri dari mockup yang akan dibuat:

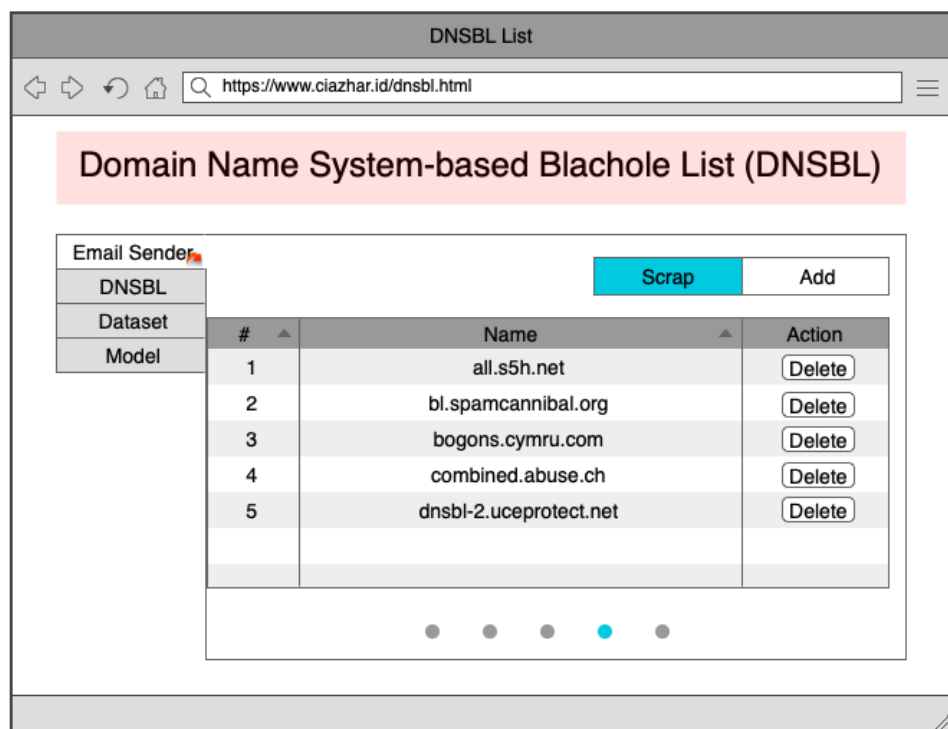
1. Halaman Email Checker

The mockup shows a web browser window titled "DNSBL List" with the URL "https://www.ciazhar.id/dnsbl.html". The main content area has a pink header "E-Mail Checker & Sender". Below this, there is a sidebar with a tabbed interface containing "Email Sender", "DNSBL", "Dataset", and "Model". The "Email Sender" tab is active, showing a form with fields for "To" (containing "to"), "Subject" (containing "subject"), and "Body" (a large text area). A "Send" button is located at the bottom right of the form.

Gambar 4.11 Desain Halaman Email Sender

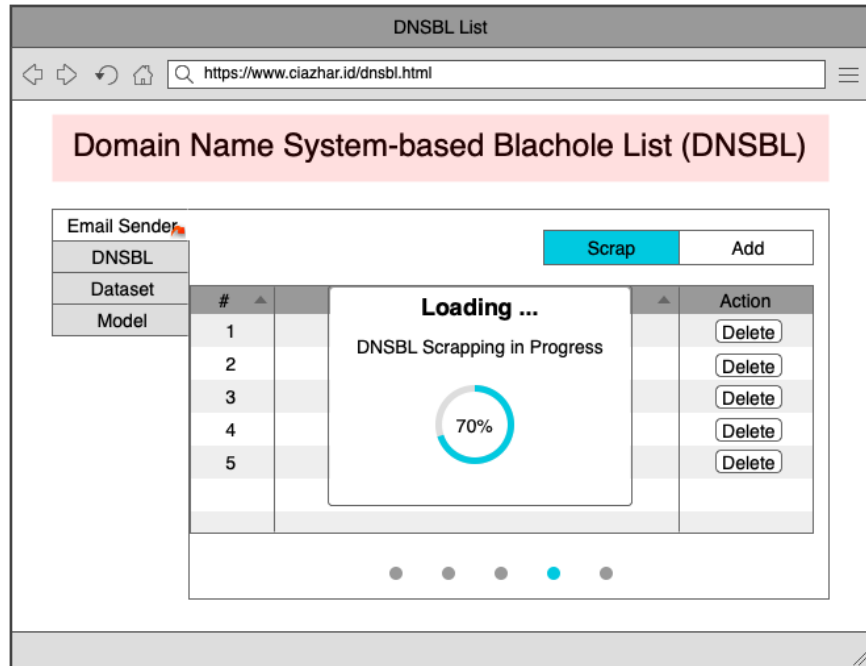
Halaman ini digunakan untuk mengecek apakah email dapat dikategorikan sebagai spam atau ham. Pada halaman tersebut terdapat form input untuk alamat email tujuan, subjek email dan konten email. Ketika tombol send di tekan maka akan melakukan validasi konten terlebih dahulu. Apabila konten teridentifikasi sebagai spam maka pengiriman email akan dibatalkan. Apabila tidak maka pengiriman email akan diteruskan.

2. Halaman Daftar DNSBL



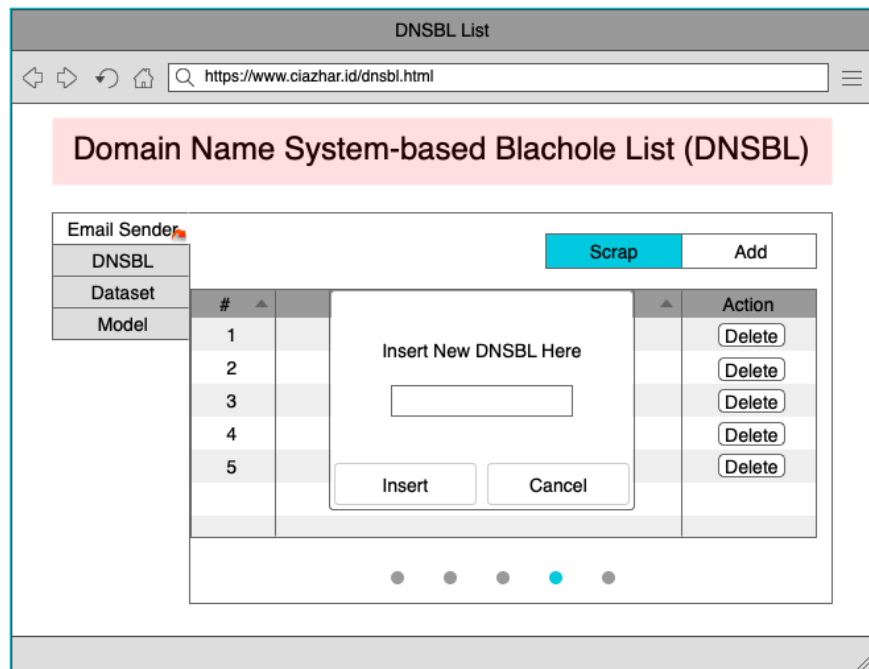
Gambar 4.12Desain Halaman DNSBL

Halaman ini berisi daftar DNSBL yang akan digunakan untuk DNSBL Filtering. Terdapat juga Menu untuk scrap DNSBL, menambah DNSBL dan menghapus DNSBL. Ketika tombol scrap ditekan maka akan dilakukan proses scrapping seperti gambar dibawah :



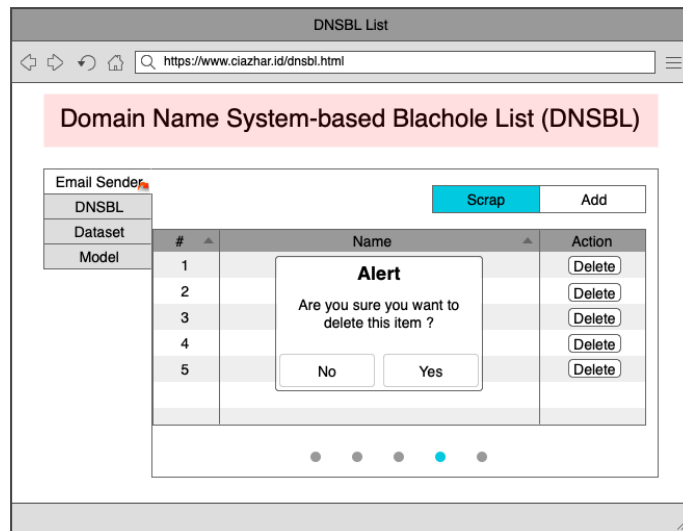
Gambar 4.13 Desain Halaman Scrap

Setelah Proses scrapping selesai daftar DNSBL akan terisi sesuai dengan item item yang ada pada web yang discrapnya. Menambah DNSBL juga dapat dilakukan secara manual menggunakan tombol Add seperti gambar berikut :



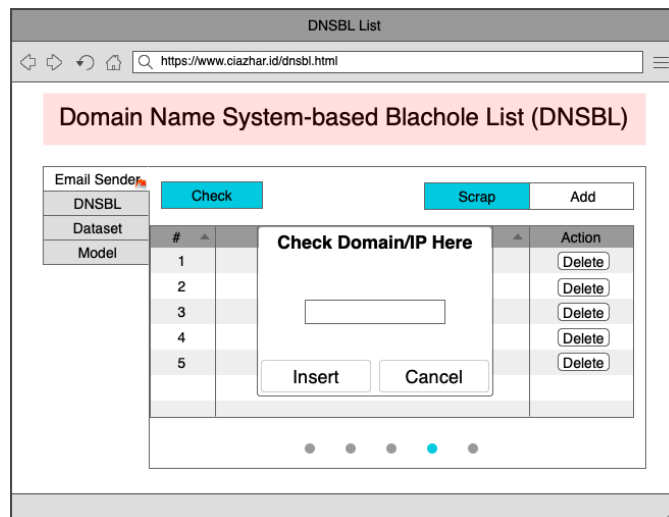
Gambar 4.14 Desain Halaman Insert DNSBL

Untuk menghapus masing masing baris DNSBL yang ada pada daftar, dapat dilakukan dengan menekan tombol Delete pada bagian *action* tiap baris. Sehingga akan muncul pop up seperti berikut :



Gambar 4.15DesainHalaman Delete DNSBL

Ketika tombol Check ditekan maka akan muncul dialog seperti gambar dibawah :



Gambar 4.16Desain Halaman Check Domain/IP

Setelah memasukkan IP/domain,backend akan memprosesnya dan menentukan IP/domain tersebut berbahaya atau tidak.

4.5 Algoritam Klasifikasi

4.5.1 Mengubah file dataset menjadi memory dataset.

1. Menyimpan file dataset ke memory

Dalam folder dataset terdapat dua folder yaitu spam dan ham. Folder spam berisi dataset email spam sedangkan folder ham berisi dataset email ham. Setiap file dataset email disimpan kedalam array yang berisi kategori email dan konten email. Karena hanya terdapat 2 folder, maka hanya terdapat 2 *category* juga. Hasilnya sebagai berikut :

Tabel 4. 9 Tabel hasil pengubahan file dataset menjadi memory dataset

Indeks Dokumen	Category	Text
1	ham	tw deal analysis - basis differential michelle , the changes for tw deal analysis to use latest basis differential based on receipt point area is in production . thanks , mei - ling
2	ham	transportation contract # 25374 michelle , please ammend oneok buston processings transportation contract # 25374 to include the month of january , 2001 . thank you , andrew pacheco
3	spam	the original your woman needs an 8 inch man . be that man for her . learn how here . turn off notifications here . df international exports ltd st . lina # 8777 belize city , belize
4	spam	you ' ll need this hello , if you want that roock harrrd johnson check out the first and original one on the market . . . don ' t be fooled by imitations and copy - cats . later , romero

2. Melakukan tokenisasi dan stop word removal pada text

Kalimat panjang pada field text dilakukan tokenisasi dan stop word removal menggunakan library HanLP. Hasil proses tokenisasi menjadi sebagai berikut :

Tabel 4. 10 Tabel hasil tokenisasi dan stop word removal pada text

Indeks Dokumen	Category	Text
1	ham	tw, deal, analysis, -, basis, differential, michelle, changes, tw, deal, analysis, use, latest, basis, differential, based, receipt, point, area, production, thanks, mei, -, ling.
2	ham	transportation, contract, #, michelle, ammend, oneok, buston, processings, transportation, contract, #, include, month, january, thank, andrew, pacheco.

3	spam	original, woman, needs, inch, man, man, learn, turn, notifications, df, international, exports, ltd, st, lina, #, belize, city, belize.
4	spam	', ll,hello, want, roock, harrrd, john, son, check, out, original, market, don,', t, fooled, imitations, copy, -, cats, later, romero.

3. Melakukan pengindeksan pada kategori dataset dan kata yang terdapat pada dataset.

Berikut merupakan table indeks kategori

Tabel 4. 11 Tabel indeks kategori

Index	Nama Kategori
0	spam
1	ham

Berikut merupakan table indeks kata

Tabel 4. 12 Tabel Indeks Kata

Index	Nama Kata
0	'
1	ll
2	Hello
3	Want
4	roock
5	Harrrd
6	John
7	Son
8	Check
9	Original
10	Market
11	Don

12	T
13	Fooled
14	Imitations
15	Copy
16	-
17	Cats
18	Later
19	Romero
20	Woman
21	Needs
22	Inch
23	Man
24	Learn
25	Turn
26	Notifications
27	Df
28	international
29	Exports
30	St
31	Lina
32	#
33	Belize
34	City
35	Transportation
36	Contract
37	Michelle
38	Amend
39	Oneok
40	buston
41	Processings

42	Include
43	Month
44	January
45	Thank
46	Andrew
47	Pacheco
48	Tw
49	Deal
50	Analysis
51	Basis
52	Differential
53	Changes
54	Use
55	Latest
56	Based
57	Receipt
58	Point
59	Area
60	Production
61	Thanks
62	Mei
63	ling

4.5.2 Mengubah dataset menjadi fitur data menggunakan DF.

Kemudian dilakukan penghitungan jumlah kata yang digunakan untuk setiap kategorinya. Berikut hasilnya :

Tabel 4. 13 Tabel hasil pengubahan ke fitur data

Word Index	Categoery Index	
	0	1
0	2	0

1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	2	0
10	1	0
11	1	0
12	1	0
13	1	0
14	1	0
15	1	0
16	1	2
17	1	0
18	1	0
19	1	0
20	1	0
21	1	0
22	1	0
23	2	0
24	1	0
25	1	0
26	1	0
27	1	0
28	1	0
29	1	0
30	1	0

31	1	0
32	1	2
33	1	0
34	1	0
35	0	2
36	0	2
37	0	2
38	0	1
39	0	1
40	0	1
41	0	1
42	0	1
43	0	1
44	0	1
45	0	1
46	0	1
47	0	1
48	0	2
49	0	2
50	0	2
51	0	2
52	0	2
53	0	1
54	0	1
55	0	1
56	0	1
57	0	1
58	0	1
59	0	1
60	0	1

61	0	1
62	0	1
63	0	1

4.5.3 Feature selection menggunakan Chi Square.

Kemudian dilakukan seleksi fitur menggunakan metode Chi Square dengan critical value sebesar 10.83. Berikut hasilnya :

Tabel 4. 14 Tabel hasil perhitungan Chi Square

Feature Index	Word Index	Category Index	Word Count	N1dot (\sum Word Count)	N0dot (\sum Dataset - N1dot)	N11 (Word Count)	N01 (Category Count [Category Index] - N11)	N00 (N0dot - N01)	N10 (N1dot - N11)	Score
1	0	0	2	2	2	2	0	2	0	10,66666667
2		1	0			0	2	0	2	10,66666667
3	1	0	1	1	3	1	1	2	0	3,2
4		1	0			0	2	1	1	2,66666667
5	2	0	1	1	3	1	1	2	0	3,2
6		1	0			0	2	1	1	2,66666667
7	3	0	1	1	3	1	1	2	0	3,2
8		1	0			0	2	1	1	2,66666667
9	4	0	1	1	3	1	1	2	0	3,2
10		1	0			0	2	1	1	2,66666667
11	5	0	1	1	3	1	1	2	0	3,2
12		1	0			0	2	1	1	2,66666667
13	6	0	1	1	3	1	1	2	0	3,2
14		1	0			0	2	1	1	2,66666667
15	7	0	1	1	3	1	1	2	0	3,2
16		1	0			0	2	1	1	2,66666667

17	8	0	1	1	3	1	1	2	0	3,2
18		1	0			0	2	1	1	2,666666667
19	9	0	1	1	3	1	1	2	0	3,2
20		1	0			0	2	1	1	2,666666667
21	10	0	1	1	3	1	1	2	0	3,2
22		1	0			0	2	1	1	2,666666667
23	11	0	1	1	3	1	1	2	0	3,2
24		1	0			0	2	1	1	2,666666667
25	12	0	1	1	3	1	1	2	0	3,2
26		1	0			0	2	1	1	2,666666667
27	13	0	1	1	3	1	1	2	0	3,2
28		1	0			0	2	1	1	2,666666667
29	14	0	1	1	3	1	1	2	0	3,2
30		1	0			0	2	1	1	2,666666667
31	15	0	1	1	3	1	1	2	0	3,2
32		1	0			0	2	1	1	2,666666667
33	16	0	1	1	3	1	1	2	0	3,2
34		1	0			0	2	1	1	2,666666667
35	17	0	1	1	3	1	1	2	0	3,2
36		1	0			0	2	1	1	2,666666667
37	18	0	1	1	3	1	1	2	0	3,2
38		1	0			0	2	1	1	2,666666667
39	19	0	1	1	3	1	1	2	0	3,2
40		1	0			0	2	1	1	2,666666667
41	20	0	1	1	3	1	1	2	0	3,2
42		1	0			0	2	1	1	2,666666667

43	21	0	1	1	3	1	1	2	0	3,2
44		1	0			0	2	1	1	2,666666667
45	22	0	1	1	3	1	1	2	0	3,2
46		1	0			0	2	1	1	2,666666667
47	23	0	1	1	3	1	1	2	0	3,2
48		1	0			0	2	1	1	2,666666667
49	24	0	1	1	3	1	1	2	0	3,2
50		1	0			0	2	1	1	2,666666667
51	25	0	1	1	3	1	1	2	0	3,2
52		1	0			0	2	1	1	2,666666667
53	26	0	1	1	3	1	1	2	0	3,2
54		1	0			0	2	1	1	2,666666667
55	27	0	1	1	3	1	1	2	0	3,2
56		1	0			0	2	1	1	2,666666667
57	28	0	1	1	3	1	1	2	0	3,2
58		1	0			0	2	1	1	2,666666667
59	29	0	1	1	3	1	1	2	0	3,2
60		1	0			0	2	1	1	2,666666667
61	30	0	1	1	3	1	1	2	0	3,2
62		1	0			0	2	1	1	2,666666667
63	31	0	1	1	3	1	1	2	0	3,2
64		1	0			0	2	1	1	2,666666667
65	32	0	1	1	3	1	1	2	0	3,2
66		1	0			0	2	1	1	2,666666667
67	33	0	1	1	3	1	1	2	0	3,2
68		1	0			0	2	1	1	2,666666667

69	34	0	1	1	3	1	1	2	0	3,2
70		1	0			0	2	1	1	2,666666667
71	35	0	0	1	3	0	2	1	1	2,666666667
72		1	1			1	1	2	0	3,2
73	36	0	0	1	3	0	2	1	1	2,666666667
74		1	1			1	1	2	0	3,2
75	37	0	0	1	3	0	2	1	1	2,666666667
76		1	1			1	1	2	0	3,2
77	38	0	0	1	3	0	2	1	1	2,666666667
78		1	1			1	1	2	0	3,2
79	39	0	0	1	3	0	2	1	1	2,666666667
80		1	1			1	1	2	0	3,2
81	40	0	0	1	3	0	2	1	1	2,666666667
82		1	1			1	1	2	0	3,2
83	41	0	0	1	3	0	2	1	1	2,666666667
84		1	1			1	1	2	0	3,2
85	42	0	0	1	3	0	2	1	1	2,666666667
86		1	1			1	1	2	0	3,2
87	43	0	0	1	3	0	2	1	1	2,666666667
88		1	1			1	1	2	0	3,2
89	44	0	0	1	3	0	2	1	1	2,666666667
90		1	1			1	1	2	0	3,2
91	45	0	0	1	3	0	2	1	1	2,666666667
92		1	1			1	1	2	0	3,2
93	46	0	0	1	3	0	2	1	1	2,666666667
94		1	1			1	1	2	0	3,2

95	47	0	0	1	3	0	2	1	1	2,666666667
96		1	1			1	1	2	0	3,2
97	48	0	0	1	3	0	2	1	1	2,666666667
98		1	1			1	1	2	0	3,2
99	49	0	0	1	3	0	2	1	1	2,666666667
100		1	1			1	1	2	0	3,2
101	50	0	0	1	3	0	2	1	1	2,666666667
102		1	1			1	1	2	0	3,2
103	51	0	0	1	3	0	2	1	1	2,666666667
104		1	1			1	1	2	0	3,2
105	52	0	0	1	3	0	2	1	1	2,666666667
106		1	1			1	1	2	0	3,2
107	53	0	0	1	3	0	2	1	1	2,666666667
108		1	1			1	1	2	0	3,2
109	54	0	0	1	3	0	2	1	1	2,666666667
110		1	1			1	1	2	0	3,2
111	55	0	0	1	3	0	2	1	1	2,666666667
112		1	1			1	1	2	0	3,2
113	56	0	0	1	3	0	2	1	1	2,666666667
114		1	1			1	1	2	0	3,2
115	57	0	0	1	3	0	2	1	1	2,666666667
116		1	1			1	1	2	0	3,2
117	58	0	0	1	3	0	2	1	1	2,666666667
118		1	1			1	1	2	0	3,2
119	59	0	0	1	3	0	2	1	1	2,666666667
120		1	1			1	1	2	0	3,2

121	60	0	0	1	3	0	2	1	1	2,666666667
122		1	1			1	1	2	0	3,2
123	61	0	0	1	3	0	2	1	1	2,666666667
124		1	1			1	1	2	0	3,2
125	62	0	0	1	3	0	2	1	1	2,666666667
126		1	1			1	1	2	0	3,2
127	63	0	0	1	3	0	2	1	1	2,666666667
128		1	1			1	1	-1	0	2

Untuk mencari score dapat menggunakan rumus sebagai berikut :

$$\frac{\sum_{dataset} x ((N11 \times N00) - (N10 \times N01))^2}{((N11+N01) \times (N11+N10) \times (N10 \times N00) \times (N01 \times N00))}$$

Keterangan :

- N11 = Jumlah dokumen yang memiliki fitur ini dan termasuk dalam kategori
- N01 = Jumlah dokumen yang tidak mengandung fitur ini tetapi termasuk dalam kategori
- N00 = Jumlah dokumen yang tidak mengandung fitur atau milik kategori
- N10 = Jumlah dokumen yang memiliki fitur ini tetapi tidak dalam kategori

Dikarenakan setiap kata memiliki score dibawah critical value maka seluruh fitur menjadi terseleksi. Namun untuk memberikan contoh untuk langkah selanjutnya, hasil seleksi fitur akan diabaikan.

Dari tabel diatas kemudian dicari df dan wordIdTrie (Peta fitur baru) dengan menggunakan kode sebagai berikut :

```
val wordIdArray = dataSet.lexicon.wordIdArray
val idMap = IntArray(wordIdArray.size)
Arrays.fill(idMap, -1)
featureData.wordIdTrie = BinTrie()
featureData.df = IntArray(selectedFeatures.size)
var p = -1
for (feature in selectedFeatures.keys) {
    ++p
    featureData.wordIdTrie.put(wordIdArray[feature], p)
    featureData.df[p] = MathUtili-
ty.sum(*featureData.featureCategoryJointCount[feature])
    idMap[feature] = p
}
```

4.5.4 Membuat problem

Problem merupakan model merepresentasikan data yang akan dihitung, berisi jumlah data training (l), jumlah fitur(n), array yang berisi nilai target(x) dan node fitur yang tersebar(y). Menggunakan kode sebagai berikut :

```
private fun createLiblinearProblem(dataSet: IDataset, baseFeatureData:
BaseFeatureData, weighter: IFeatureWeighter): Problem {
    val problem = Problem()
    val n = dataSet.size()
    problem.l = n
```

```

        problem.n = baseFeatureData.featureCategoryJointCount.size
        problem.x = arrayOfNulls<Array<FeatureNode>>(n)
        problem.y = DoubleArray(n)
        val iterator = dataSet.iterator()
        for (i in 0 until n) {
            val document = iterator.next()
            problem.x[i] = buildDocumentVector(document, weighter)
            problem.y[i] = document.category.toDouble()
        }
        return problem
    }

    private fun buildDocumentVector(document: Document, weighter: IFeature-
    Weighter): Array<FeatureNode> {
        val termCount = document.tfMap.size
        val x = arrayOfNulls<FeatureNode>(termCount)
        val tfMapIterator = document.tfMap.entries.iterator()
        for (j in 0 until termCount) {
            val tfEntry = tfMapIterator.next()
            val feature = tfEntry.key
            val frequency = tfEntry.value[0]
            x[j] = FeatureNode(feature + 1, weighter.weight(feature, frequency))
        }

        var normalizer = 0.0
        for (j in 0 until termCount) {
            val weight = x[j]!!.getValue()
            normalizer += weight * weight
        }
        normalizer = Math.sqrt(normalizer)
        for (j in 0 until termCount) {
            val weight = x[j]!!.getValue()
            x[j]!!.setValue(weight / normalizer)
        }

        return x
    }

```

4.5.5 Melakukan pemodelan menggunakan Support Vector Machine.

Dari model Problem yang sudah didefinisikan dilakukan training menggunakan L1R_LR (L1-regularized logistic regression) nilai C = 500 dan kriteria berhenti = 0.01.

```

private fun solveLibLinearProblem(problem: Problem): Model {
    val lparam = Parameter(SolverType.L1R_LR, 500.0, 0.01)
    return Linear.train(problem, lparam)
}

public static Model train(Problem prob, Parameter param) {
    if (prob == null) throw new IllegalArgumentException("problem must not be
    null");
    if (param == null) throw new IllegalArgumentException("parameter must not be
    null");

    if (prob.n == 0) throw new IllegalArgumentException("problem has zero fea-
    tures");
    if (prob.l == 0) throw new IllegalArgumentException("problem has zero in-
    stances");
}

```

```

for (Feature[] nodes : prob.x) {
    int indexBefore = 0;
    for (Feature n : nodes) {
        if (n.getIndex() <= indexBefore) {
            throw new IllegalArgumentException("feature nodes must be sorted by index in
ascending order");
        }
        indexBefore = n.getIndex();
    }
}

int l = prob.l;
int n = prob.n;
int w_size = prob.n;
Model model = new Model();

if (prob.bias >= 0)
    model.nr_feature = n - 1;
else
    model.nr_feature = n;

    model.solverType = param.solverType;
    model.bias = prob.bias;

if (param.solverType.isSupportVectorRegression()) {
    model.w = new double[w_size];
    model.nr_class = 2;
    model.label = null;

checkProblemSize(n, model.nr_class);

train_one(prob, param, model.w, 0, 0);
    } else {
int[] perm = new int[l];

GroupClassesReturn rv = groupClasses(prob, perm);
int nr_class = rv.nr_class;
int[] label = rv.label;
int[] start = rv.start;
int[] count = rv.count;

checkProblemSize(n, nr_class);

        model.nr_class = nr_class;
        model.label = new int[nr_class];
for (int i = 0; i < nr_class; i++)
    model.label[i] = label[i];

double[] weighted_C = new double[nr_class];
for (int i = 0; i < nr_class; i++)
    weighted_C[i] = param.C;
for (int i = 0; i < param.getNumWeights(); i++) {
    int j;
    for (j = 0; j < nr_class; j++)
        if (param.weightLabel[i] == label[j]) break;

    if (j == nr_class) throw new IllegalArgumentException("class label " +
param.weightLabel[i] + " specified in weight is not found");
    weighted_C[j] *= param.weight[i];
}

Feature[][] x = new Feature[l][];
for (int i = 0; i < l; i++)
    x[i] = prob.x[perm[i]];

```

```

        Problem sub_prob = new Problem();
        sub_prob.l = 1;
        sub_prob.n = n;
        sub_prob.x = new Feature[sub_prob.l][];
        sub_prob.y = new double[sub_prob.l];

    for (int k = 0; k < sub_prob.l; k++)
        sub_prob.x[k] = x[k];

    if (param.solverType == SolverType.MCSVM_CS) {
        model.w = new double[n * nr_class];
        for (int i = 0; i < nr_class; i++) {
            for (int j = start[i]; j < start[i] + count[i]; j++) {
                sub_prob.y[j] = i;
            }
        }

        SolverMCSVM_CS solver = new SolverMCSVM_CS(sub_prob, nr_class,
            weighted_C, param.eps);
        solver.solve(model.w);
    } else {
        if (nr_class == 2) {
            model.w = new double[w_size];

            int e0 = start[0] + count[0];
            int k = 0;
            for (; k < e0; k++)
                sub_prob.y[k] = +1;
            for (; k < sub_prob.l; k++)
                sub_prob.y[k] = -1;

            train_one(sub_prob, param, model.w, weighted_C[0], weighted_C[1]);
        } else {
            model.w = new double[w_size * nr_class];
            double[] w = new double[w_size];
            for (int i = 0; i < nr_class; i++) {
                int si = start[i];
                int ei = si + count[i];

                int k = 0;
                for (; k < si; k++)
                    sub_prob.y[k] = -1;
                for (; k < ei; k++)
                    sub_prob.y[k] = +1;
                for (; k < sub_prob.l; k++)
                    sub_prob.y[k] = -1;

                train_one(sub_prob, param, w, weighted_C[i], param.C);
            }
            for (int j = 0; j < n; j++)
                model.w[j * nr_class + i] = w[j];
        }
    }
}
return model;
}

```

4.5.6 Melakukan prediksi konten email

Setelah melakukan training dataset dapat dilakukan prediksi konten menggunakan kode berikut :

```

fun predict(classifier: IClassifier, text: String) : String {
    return classifier.classify(text)
}

@Override
public String classify(String text) throws IllegalArgumentException, IllegalStateExcp
tion
{
    Map<String, Double> scoreMap = predict(text);

    return CollectionUtility.max(scoreMap);
}

@Throws(IllegalArgumentException::class, IllegalStateException::class)
override fun predict(text: String?): Map<String, Double> {
    if (model == null) {
        throw IllegalStateException("Model yang tidak terlatih! Tidak dapat men-
jalankan perkiraan!")
    }
    if (text == null) {
        throw IllegalArgumentException("Parameter teks == null")
    }

    //Segmentasi kata, buat dokumen
    val document = Document(model!!.wordIdTrie, model!!.tokenizer.segment(text))

    return predict(document)
}

@Override
public Map<String, Double> predict(Document document)
{
    AbstractModel model = getModel();
    if (model == null)
    {
        throw new IllegalStateException("Model yang tidak terlatih! Tidak dapat men-
jalankan perkiraan!");
    }
    if (document == null)
    {
        throw new IllegalArgumentException("Parameter teks == null ");
    }

    double[] probs = categorize(document);
    Map<String, Double> scoreMap = new TreeMap<String, Double>();
    for (int i = 0; i < probs.length; i++)
    {
        scoreMap.put(model.catalog[i], probs[i]);
    }
    return scoreMap;
}

@Throws(IllegalArgumentException::class, IllegalStateException::class)
override fun categorize(document: Document): DoubleArray {
    val x = buildDocumentVector(document, model!!.featureWeighter)
    val probs = DoubleArray(model!!.svmModel.nrClass)
    Linear.predictProbability(model!!.svmModel, x, probs)
    return probs
}

public static double predictProbability(Model model, Feature[] x, double[]
prob_estimates) throws IllegalArgumentException {
    if (!model.isProbabilityModel()) {
        StringBuilder sb = new StringBuilder("probability output is only

```

```

supported for logistic regression");
    sb.append(". This is currently only supported by the following solv-
ers: ");
    int i = 0;
    for (SolverType solverType : SolverType.values()) {
        if (solverType.isLogisticRegressionSolver()) {
            if (i++ > 0) {
                sb.append(", ");
            }
            sb.append(solverType.name());
        }
    }
    throw new IllegalArgumentException(sb.toString());
}
int nr_class = model.nr_class;
int nr_w;
if (nr_class == 2)
    nr_w = 1;
else
    nr_w = nr_class;

double label = predictValues(model, x, prob_estimates);
for (int i = 0; i < nr_w; i++)
    prob_estimates[i] = 1 / (1 + Math.exp(-prob_estimates[i]));

if (nr_class == 2) // for binary classification
    prob_estimates[1] = 1. - prob_estimates[0];
else {
    double sum = 0;
    for (int i = 0; i < nr_class; i++)
        sum += prob_estimates[i];

    for (int i = 0; i < nr_class; i++)
        prob_estimates[i] = prob_estimates[i] / sum;
}

return label;
}

```

Kemudian dicari kategori spam atau ham yang memiliki score maksimal.

```

public static String max(Map<String, Double> scoreMap)
{
    double max = Double.NEGATIVE_INFINITY;
    String best = null;
    for (Map.Entry<String, Double> entry : scoreMap.entrySet())
    {
        Double score = entry.getValue();
        if (score > max)
        {
            max = score;
            best = entry.getKey();
        }
    }

    return best;
}

```

BAB V

HASIL DAN PENGUJIAN

5.1 Source Code

5.1.1 Fungsi checkDomain

```
fun checkDomain(domain: String, dnsbl: String): Observable<Boolean> {
    return Observable
        .from(Lookup(domain, Type.A).run()).subscribeOn(Schedulers.newThread())
        .flatMap {

            Observable.just(it).subscribeOn(Schedulers.newThread()).map {

                var result = false
                it as ARecord
                val ip = reverseIp(it.address.hostAddress)
                val lookup2 = Lookup("$ip.$dnsbl", Type.TXT)
                val resolver = SimpleResolver()
                    lookup2.setResolver(resolver)
                    lookup2.setCache(null)
                    lookup2.run()

                if (lookup2.result == Lookup.SUCCESSFUL) {
                    result = true
                }
                else if (lookup2.result == Lookup.HOST_NOT_FOUND) {
                    result = false
                }

                return@map result
            }
        }.toList().map {
            return@map !it.joinToString().contains("false")
        }
}
```

```
private fun reverseIp(content: String) : String {
    return content.split(".").reversed().joinToString(".")
}
```

Gambar 5. 1 Gambar potongan kode program Check Domain

Gambar diatas merupakan potongan kode program *Check Domain*. Fungsi tersebut memiliki 2 parameter yaitu domain dan dnsbl. Untuk setiap domain yang di proses akan dilakukan pengecekan apakah terdaftar dalam dnsbl. Proses pengecekan dilakukan secara asynronus untuk menghemat waktu. Domain yang terindikasi bermasalah maka program akan mengembalikan nilai true dan sebaliknya.

5.1.2 Fungsi scrapDnsbl

```
override fun scrapDnsbl(fileName : String) : String {
    var dnsblList : List<Dnsbl> = readFromCsv(fileName)
    val dnsbls : MutableList<Dnsbl> = mutableListOf()
    val doc =
        Jsoup.connect("https://en.m.wikipedia.org/wiki/Comparison_of_DNS_blacklists").get()
    val table = doc.select("table").get(2)
    val rows = table.select("tr")

    for (i in 1 until rows.size) {
        val row = rows.get(i)
        val cols = row.select("td")
        val dnsbl = html2text(cols.get(0).html()).split(" ")[0]
        if (
            !dnsbl.contains("ahbl.org") &&
            !dnsbl.contains("orbitrbl.com") &&
            !dnsbl.contains("Paid") &&
            !dnsbl.contains("proxybl.org") &&
            !dnsbl.contains("spamcannibal.org") &&
            !dnsbl.contains("drand.net") &&
            !dnsbl.contains("surgate.net") &&
            !dnsbl.contains("quorum.to")
        ) {
            dnsbls.add(Dnsbl(
                name = dnsbl
            ))
        }
    }
    dnsblList += dnsbls
    dnsblList = dnsblList.distinctBy { it.name }
    return writeToCsv(fileName, dnsblList)
}
```



```

fun readFromCsv(fileName: String) : MutableList<Dnsbl>{
    var fileReader: BufferedReader? = null
        val dnsbls = mutableListOf<Dnsbl>()

    try {
        var line: String?

            fileReader = BufferedReader(FileReader(fileName))
        fileReader.readLine()
        line = fileReader.readLine()
        while (line != null) {
            val tokens = line.split(",")
            if (tokens.isNotEmpty()) {
                val dnsbl = Dnsbl(tokens[DNSBL_NAME])
                    dnsbls.add(dnsbl)
                }
                line = fileReader.readLine()
            }
        } catch (e: Exception) {
            println("Reading CSV Error!")
            e.printStackTrace()
        } finally {
            try {
                fileReader!!.close()
            } catch (e: IOException) {
                println("Closing fileReader Error!")
                e.printStackTrace()
            }
        }
    }
    return dnsbls
}

```

Gambar 5. 2 Gambar potongan kode program Scrap DNSBL

Gambar diatas merupakan potongan kode program *Scrap DNSBL*. Fungsi tersebut memiliki 1 parameter yaitu *fileName* yaitu nama file untuk menyimpan data hasil scrapping. Proses scrapping dilakukan pada alamat web https://en.m.wikipedia.org/wiki/Comparison_of_DNS_blacklists. Program akan mencari table yang berisi daftar dnsbl, mengambilalamat dnsbl dari *td* terse-

but.Menghapus tag html dan memfilter beberapa dnbsl yang sudah tidak aktif. Hasil dari scrapping menghasilkan list dnsbl yang kemudian ditulis kedalam file.

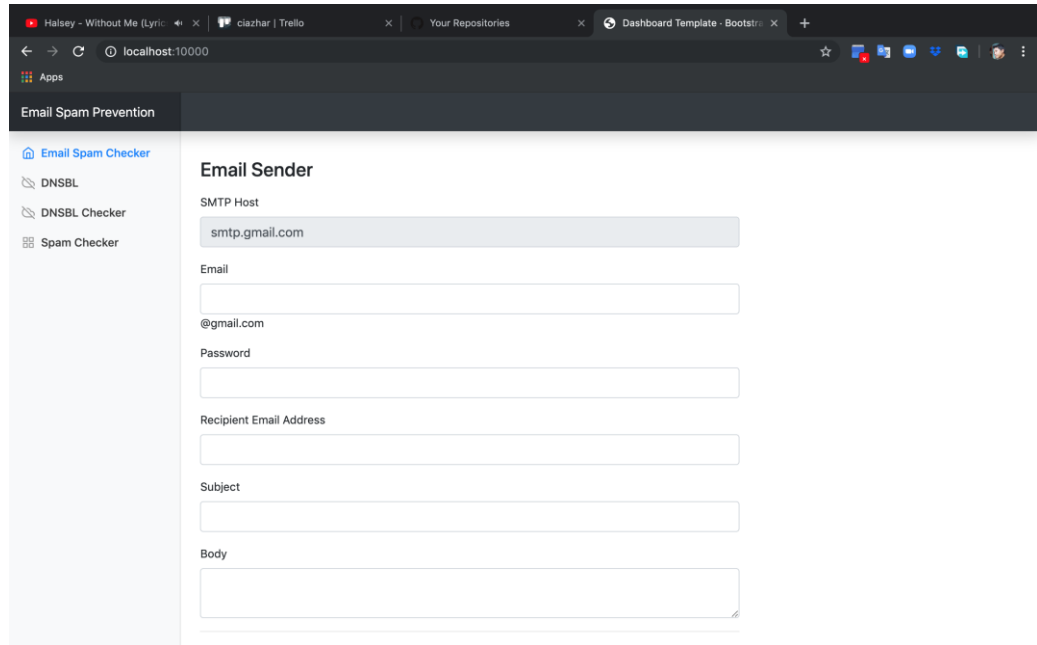
5.1.3 Train Dataset

```
public void train(IDataset dataSet){
    if (dataSet.size() == 0) throw new IllegalArgumentException("Kumpulan
    data pelatihan kosong dan tidak dapat melanjutkan pelatihan");
    // Fitur seleksi
    DfFeatureData featureData = selectFeatures(dataSet);
    // Logika perhitungan berat konstruksi
    IFeatureWeighter weighter = new TfIdfFeatureWeighter(dataSet.size(),
    featureData.df);
    // Membangun masalah SVM
    Problem problem = createLiblinearProblem(dataSet, featureData, weight-
    er);
    // Memori bebas
    BinTrie<Integer> wordIdTrie = featureData.wordIdTrie;
        featureData = null;
        ITokenizer tokenizer = dataSet.getTokenizer();
        String[] catalog = dataSet.getCatalog().toArray();
        dataSet = null;
        System.gc();
    // Memecahkan masalah SVM
    Model svmModel = solveLibLinearProblem(problem);
    // Tinggalkan data yang berguna untuk tinggal
    model = new LinearSVMModel();
    model.tokenizer = tokenizer;
    model.wordIdTrie = wordIdTrie;
    model.catalog = catalog;
    model.svmModel = svmModel;
    model.featureWeighter = weighter;
}
```

Gambar 5. 3 Gambar potongan kode training svm

5.2 Antarmuka

- Halaman Email Spam Checker



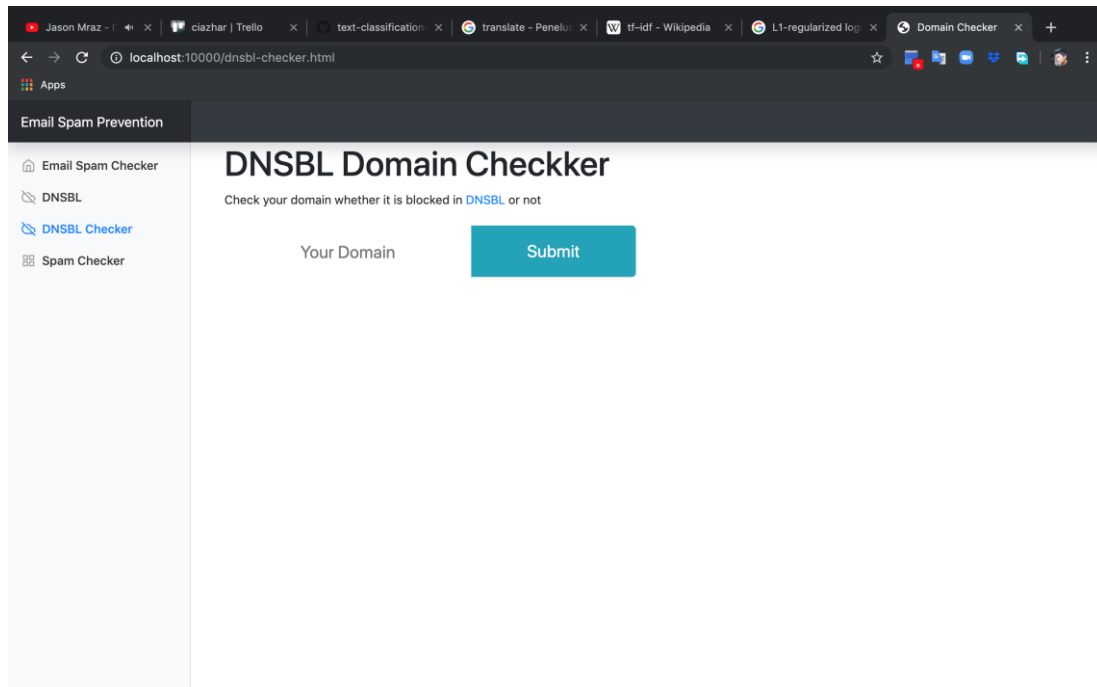
The screenshot shows a web browser window with the address bar displaying 'localhost:10000'. The browser has several tabs open, including 'Halsey - Without Me (Lyric)', 'clazhar | Trello', 'Your Repositories', and 'Dashboard Template - Bootstrap'. The application interface is titled 'Email Spam Prevention' in the top navigation bar. On the left, there is a sidebar menu with the following items: 'Email Spam Checker' (highlighted with a blue icon), 'DNSBL', 'DNSBL Checker', and 'Spam Checker'. The main content area is titled 'Email Sender' and contains the following form fields:

- SMTP Host:** A text input field containing 'smtp.gmail.com'.
- Email:** A text input field containing '@gmail.com'.
- Password:** A text input field.
- Recipient Email Address:** A text input field.
- Subject:** A text input field.
- Body:** A large text area for the email body.

Gambar 5. 4 Gambar halaman email spam checker

Gambar diatas merupakan Halaman Email Spam Checker yang dapat digunakan untuk mengecek apakah email termasuk spam atau tidak sebelum dikirim.

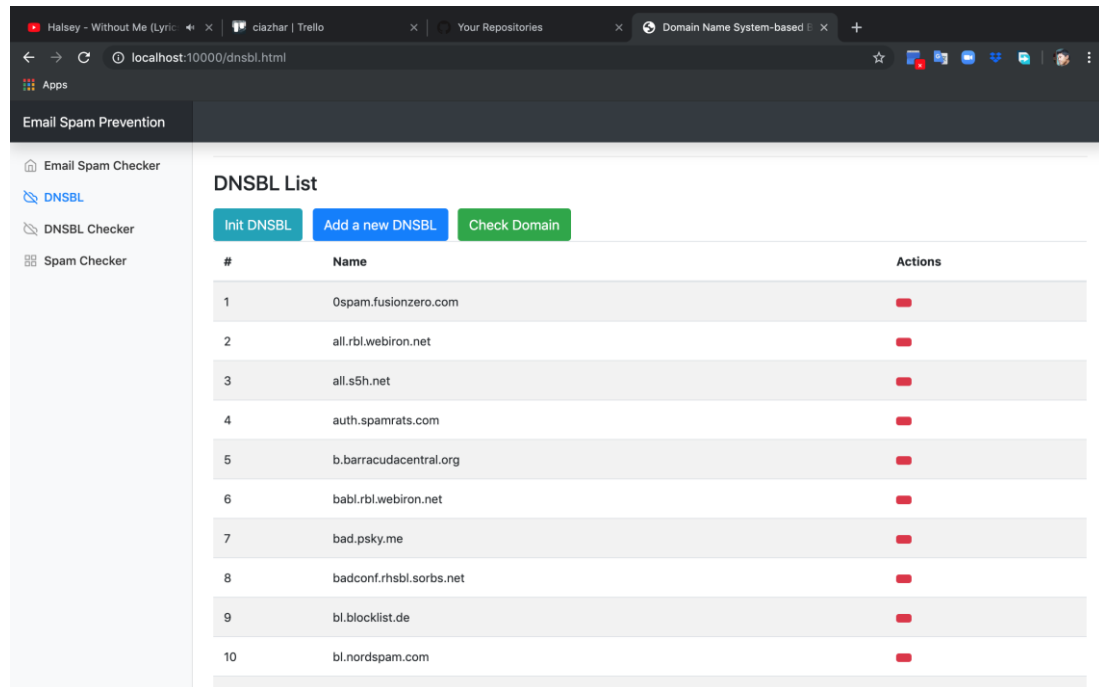
- Halaman DNSBL Checker



Gambar 5. 5 Halaman Domain Checker

Gambar diatas merupakan Halaman Domain Chcker yang berisi menu untuk pengecekan domain berbasis DNSBL melalui web. Dengan memasukkan nama domain pada kolom Your Domain dan menekan tombol submit, program akan memproses dan menampilkan hasilnya di bagian bawah menu. Hasil pengecekan domain dapat berupa daftar dnsbl yang memblokir domain tersebut untuk domain bermasalah atau pesan bahwa domain tesebut aman.

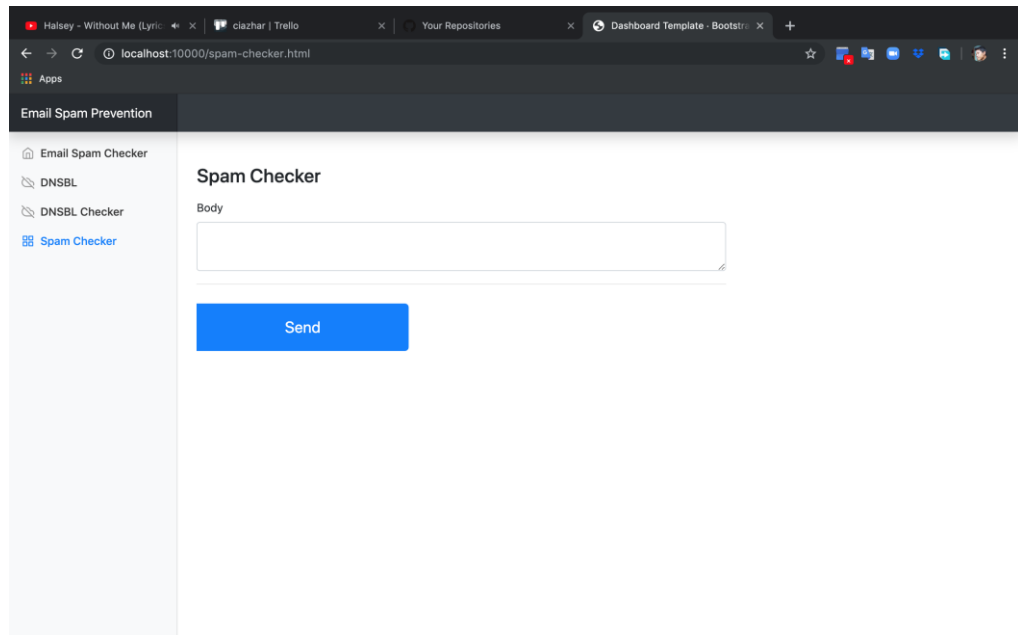
- Halaman DNSBL



Gambar 5. 6 14Halaman DNSBL

Gambar diatas merupakan Halaman DNSBL yang berisi daftar DNSBL yang akan digunakan untuk mengecek apakah domain yang akan di test. DNSBL dapat diperoleh dari proses scrapping dengan menekan tombol *Init DNSBL* atau dengan menginputkan manual nama DNSBL. Selain itu dimungkinkan juga menghapus DNSBL yang sudah ada.

- Halaman Spam Checker



Gambar 5. 7 Halaman Spam Checker

Gambar diatas merupakan Halaman Spam Checker yang digunakan untuk mengecek kalimat.

5.3 Pengujian Sistem

Pengujian dilakukan untuk mengetahui apakah aplikasi yang telah dibangun dapat berjalan dengan baik dan memenuhi spesifikasi yang telah ditentukan.

5.3.1 White Box Testing

Untuk melakukan pengujian *white-box* penulis menulis kode unit testing. Pengujian ini dilakukan pada bagian fungsi yang paling penting, yaitu fungsi check-Domain dan scrapDnsbl. Sehingga dapat dihasilkan table hasil testing sebagai berikut.

Tabel 5. 1 Tabel hasil white box testing

No	Modul	Hasil yang diharapkan	Hasil Pengujian
1.	ScrapDnsbl	Mendapatkan daftar DNSBL dari alamat https://en.m.wikipedia.org/wiki/Comparison_	Sesuai

		of_DNS_blacklistsdan menyimpannya pada file dnsbl.csv	
2.	CheckSecure-DomainSuccess	Membuktikan bahwa domain dinus.ac.id tidak bermasalah.	Sesuai
3.	CheckVulnerable-DomainSuccess	Membuktikan bahwa domain spgdt.id bermasalah	Sesuai

5.3.2 Black Box Testing

Black-Box Testing Domain Checker dilakukan melalui Web. Di bawah ini merupakan tabel hasil pengujian aplikasi menggunakan *Black-Box Testing*.

Tabel 5. 2 Tabel hasil black box testing

No	Modul	Hasil yang diharapkan	Hasil Pengujian
1.	Laman Email Spam Checker	Mengecek email apakah termasuk spam atau tidak, dan apabila tidak maka akan mengirimkan email tersebut.	Sesuai
2.	Laman DNSBL Checker	Mengecek apakah domain bermasalah atau tidak dan menampilkan hasilnya.	Sesuai
3.	Laman DNSBL	Menampilkan halaman DNSBL yang berisi daftar DNSBL.	Sesuai
4.	Init DNSBL	Pada laman DNSBL terdapat tombol Init DNSBL, ketika ditekan akan menambahkan daftar DNSBL dari alamat https://en.m.wikipedia.org/wiki/Comparison_of_DNS_blacklists dan memasukkannya ke daftar DNSBL	Sesuai
5.	Tambah DNSBL	Pada laman DNSBL terdapat tombol “Add a new DNSBL” dan data yang di inputkan akan masuk kedalam daftar DNSBL	Sesuai
6.	Delete	Pada laman DNSBL, di setiap baris DNSBL terdapat	Sesuai

	DNSBL	tombol merah, apabila ditekan maka data akan dihapus dari daftar DNSBL.	
7.	Spam Checker	Mengategorikan teks apakah termasuk sebuah konten ham atau spam	Sesuai

5.3.3 Confusion Matrix

Untuk menguji tingkat akurasi, penelitian ini menggunakan metode Confusion Matrix. Confusion Matrix adalah metode yang digunakan untuk melakukan perhitungan akurasi pada konsep data mining. Confusion Matrix untuk penelitian ini memiliki tabel sebagai berikut

Tabel 5. 3 Tabel Confusion Matrix

		Label Prediksi	
		Ham	Spam
Label Sebenarnya	Ham	A	B
	Spam	C	D

Dengan rumus untuk menghitung akuraisnya adalah :

$$\text{Accuracy} = \frac{A+D}{A+B+C+D}$$

Keterangan:

A = data ham yang diprediksi sebagai ham

B = data ham yang diprediksi sebagai spam

C = data spam yang diprediksi sebagai ham

D = data spam yang diprediksi sebagai spam

Dari data training dipilih data secara acak untuk dijadikan data sample yang akan di uji dalam penelitian ini. Dalam penelitian ini dipilih 1000 data acak dari data training. Dari sample data tersebut didapatkan nilai sebagai berikut :

Tabel 5. 4 Tabel hasil perhitungan Conusion Matrix

Chance	A	B	C	D	Accuracy
1	500	0	0	500	100
2	500	0	3	497	99,7
3	500	0	0	500	100
4	500	0	0	500	100
5	500	0	0	500	100
6	500	0	0	500	100
7	500	0	2	498	99,8
8	500	0	0	500	100
9	500	0	0	500	100
10	500	0	1	499	99,9
Average Accuracy					99,94

Dari table diatas dapat disimpulkan bahwa akurasi nya adalah 99,94% .

BAB VI

KESIMPULAN DAN SARAN

6.1. Kesimpulan

Setelah melakukan penerapan DNSBL dan Support Vector Machine (SVM) pada sistem peringatan dini untuk mendeteksi spam pada email, kesimpulan dari penelitian ini adalah :

- SVM, TF-IDF, Chi Square dan DNSBL filter dapat diimplementasikan pada sistem peringatan dini pendeteksi spam pada email. Sistem peringatan dini ini direalisasikan ke dalam bentuk sebuah pengaya web browser.
- Sistem yang telah dibuat menggunakan SVM dan DNSBL sebagai teknik klasifikasi telah diimplementasikan secara benar dan sesuai dengan hasil yang diharapkan.

6.2. Saran

Penulis sadari bahwa apa yang penulis buat saat ini belumlah sempurna, oleh karena itu penulis memberikan saran kepada pengembang sistem peringatan dini untuk mendeteksi spam pada email tahap selanjutnya. Berikut merupakan saran yang diberikan penulis:

1. Untuk menambah fitur atau metode lain yang dapat meningkatkan efektivitas filter spam.
2. Mengurangi waktu pemfilteran dari masing masing fitur.
3. Melakukan optimasi sehingga sistem peringatan dini tersebut menjadi lebih sempurna.

DAFTAR PUSTAKA

- Aldighieri, R. (2018). *Marketer email tracker*. DMA.
- Alkahtani, H. (t.thn.). A Taxonomy of Email SPAM Filters.
- Allen, D. (2010). Windows To Linux. Dalam *Network+ Guide To Networks* (hal. 192). Prentice Hall.
- Android. (t.thn.). *Kotlin*. Dipetik Juli 21, 2018, dari <https://developer.android.com/kotlin/index.html>
- Arif, G. (t.thn.). *Agile Scrum for Web Development*. Dipetik September 25, 2015, dari <https://www.neonrain.com/agile-scrum-web-development>
- Bajaj, K. (2016). A Multi-layer Model to Detect Spam Email at Client Side.
- Boeing, G. (2016). New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings. *Journal of Planning Education and Research* .
- Breitinger, C. (2016). Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries* .
- D, K. R. (2014). Latent Semantic Indexing Based SVM Model for Email Spam Classification. *Journal of Scientific & Industrial Research* , 73, 437-442.
- Deemer, P. (2012). *The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum (Version 2.0)*. InfoQ.
- Deering, S. (t.thn.). *Do you know what a REST API is?* . Dipetik Maret 11, 2019, dari <https://www.sitepoint.com/developers-rest-api/>
- Facebook. (2018). *Facebook Q4 2018 Results*.
- Gustavsson, E. S. (2016). Efficient data communication between a webclient and a cloud environment,.
- Heiss, J. J. (2019, Juli 1). *The Advent of Kotlin: A Conversation with JetBrains' Andrey Breslav*. Diambil kembali dari Oracle: <https://www.oracle.com/technetwork/articles/java/breslav-1932170.html>
- Heiss, J. (2013, April). *The Advent of Kotlin: A Conversation with JetBrains' Andrey Breslav*. (Oracle Technology Network) Dipetik Februari 2, 2014, dari <http://www.oracle.com/technetwork/articles/java/breslav-1932170.html>
- Henry, J. (1993). Quantitative assessment of the software maintenance process and requirements volatility. In *Proc. of the ACM Conference on Computer Science* , 346–351.
- Jung, J. (2004). An Empirical Study of Spam Traffic and the Use of DNS Black Lists.
- MailTarget. (2018). *MailTarget Company Profile*.
- millions), N. o. (t.thn.). (Statista) Dipetik Maret 11, 2019, dari <https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/>
- Mobius. (2015, Januari 8). *Андрей Бреслав — Kotlin для Android: коротко и ясно*. Dipetik Mei 28, 2017
- Mozilla. (t.thn.). *HTML*. Dipetik Maret 11, 2019, dari https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML
- Mozilla. (t.thn.). *HTTP*. Dipetik Maret 9, 2019, dari <https://developer.mozilla.org/en-US/docs/Web/HTTP>

Nidhra, S. (2012). BLACK BOX AND WHITE BOX TESTING TECHNIQUES – ALITERATURE REVIEW. *International Journal of Embedded Systems and Applications (IJESA)* , 2 (2), 29-50.

Oracle. (t.thn.). *What Are Web Services?* Dipetik Maret 11, 2019, dari <https://docs.oracle.com/javaee/6/tutorial/doc/gijvh.html>

Ouyang, T. (2013). A large-scale empirical analysis of email spam detection through network characteristics in a stand-alone enterprise. *Computer Networks* .

Petra. (t.thn.). *Apa itu World Wide Web ?* Dipetik Maret 11, 2019, dari http://faculty.petra.ac.id/dwikris/docs/desgrafisweb/www/4-apaitu_www.html

Pour, A. N. (2012). MINIMIZING THE TIME OF SPAM MAIL DETECTION BY RELOCATING FILTERING SYSTEM TO THE SENDER MAIL SERVER. *International Journal of Network Security & Its Applications (IJNSA)* , 4 (2), 53-62.

Rajaraman, A. *Mining of Massive Datasets*. 2011.

Rajaraman, A. (2011). *Mining of Massive Datasets*.

Schwaber, K. *Agile Project Management with Scrum*. Microsoft Press.

Schwaber, K. (2004). *Agile Project Management with Scrum*. Microsoft Press.

Silnov, D. S. (2016). An Analysis of Modern Approaches to the Delivery of Unwanted Emails (Spam). *Indian Journal of Science and Technology* , 9.

Sutherland, J. (2004). *Agile Development: Lessons learned from the first Scrum*.

Takeuchi, H. (2010). The New New Product Development Game. *Harvard Business Review* .

Tech Term. (t.thn.). *Client–server model*. Dipetik Maret 10, 2019, dari https://techterms.com/definition/client-server_model

THE RADICATI GROUP, INC. (2015). *Email Statistics Report, 2015-2019*.

Tuteja, S. K. (2016). A Survey on Classification Algorithms for Email Spam Filtering. *International Journal of Engineering Science and Computing* , 6 (5), 5937-5940.

Twitter. (2018). *Selected Company Financials and Metrics*.

Verheyen, G. (t.thn.). *Scrum: Framework, not methodology*. Dipetik Februari 24, 2016, dari <https://guntherverheyen.com/2013/03/21/scrum-framework-not-methodology/>

Waidyanatha, N. (2010). Towards a typology of integrated functional early warning systems. *Int. J. Crit. Infrastructures* , 6 (1), 31-51.

What is Scrum? An Agile Framework for Completing Complex Projects - Scrum Alliance. (t.thn.). Dipetik Februari 24, 2016, dari <https://www.scrumalliance.org/learn-about-scrum>

Wikipedia. (t.thn.). *Bounce Message*. Dipetik Juni 3, 2018, dari https://en.wikipedia.org/wiki/Bounce_message

Wikipedia. (t.thn.). *Distribusi khi-kuadrat* . Diambil kembali dari Wikipedia: https://id.wikipedia.org/wiki/Distribusi_khi-kuadrat

Wikipedia. (t.thn.). *DNSBL*. Dipetik Juli 21, 2018, dari <https://en.wikipedia.org/wiki/DNSBL>

Wikipedia. (t.thn.). *DNSBL*. Dipetik Juli 21, 2018, dari <https://en.wikipedia.org/wiki/DNSBL>

Wikipedia. (t.thn.). *Email Marketing*. Dipetik 05 30, 2018, dari https://en.wikipedia.org/wiki/Email_marketing

Wikipedia. (t.thn.). *Java*. Dipetik Maret 21, 2019, dari <https://id.wikipedia.org/wiki/Java>

Wikipedia. (t.thn.). *Kotlin (bahasa pemrograman)* . Diambil kembali dari Wikipedia:
[https://id.wikipedia.org/wiki/Kotlin_\(bahasa_pemrograman\)](https://id.wikipedia.org/wiki/Kotlin_(bahasa_pemrograman))
Wikipedia. (t.thn.). *Pemelajaran Mesin*. Dipetik Maret 11, 2019, dari
https://id.wikipedia.org/wiki/Pemelajaran_mesin
Wikipedia. (t.thn.). *Sistem Penamaan Domain*. Dipetik Juli 11, 2018, dari
https://id.wikipedia.org/wiki/Sistem_Penamaan_Domain
Wikipedia. (t.thn.). *Support Vector Machine*. Dipetik Februari 26, 2019, dari
https://en.wikipedia.org/wiki/Support-vector_machine
Wikipedia. (t.thn.). *Surat Elektronik*. Dipetik Mei 20, 2018, dari
https://id.wikipedia.org/wiki/Surat_elektronik
Wikipedia. (t.thn.). *Web Scrapping*. Dipetik Oktober 21, 2018, dari
https://en.wikipedia.org/wiki/Web_scraping
Wykowski, T. (2018). Lessons learned: Using Scrum in non-technical teams. XP
2018 Conference.