KLASIFIKASI DNA TUBERKULOSIS BERDASARKAN K-MER MENGGUNAKAN SUPPORT VECTOR MACHINE (SVM) DAN *VARIABLE NEIGHBORHOOD SEARCH* (VNS)

TESIS

Untuk memenuhi sebagian persyaratan
Memperoleh gelar Magister Ilmu Komputer/Informatika

Disusun Oleh:
Mochammad Anshori
NIM: 176150100111013

PROGRAM STUDI MAGISTER ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KLASIFIKASI DNA TUBERKULOSIS BERDASARKAN K-MER MENGGUNAKAN *SUPPORT VECTOR MACHINE* (SVM) DAN *VARIABLE NEIGHBORHOOD SEARCH* (VNS)

TESIS

Diajukan untuk memenuhi sebagian persyaratan
Memperoleh gelar Magister Komputer

Disusun Oleh:
Mochammad Anshori
NIM: 176150100111013

Tesis ini telah diuji dan dinyatakan lulus pada
8 Agustus 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D.	Ahmad Afif Supianto, Dr.Eng., S.Si., M.Kom
NIP. 19720919 199702 1 001	NIK. 2012018206231001

Mengetahui,
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengerathuan saya, di dalam naskah Tesis ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini serta disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah Tesis ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia Tesis ini digugurkan dan gelar akademik yang telah saya peroleh (Magister) dibatalkan serta diproses sesuati dengan peraturan perundang-undang yang berlaku (UU No. 20 Tahun 2003, Pasal 25 Ayat 2 dan Pasal 70).

Malang, 13 Agustus 2019

Mochammad Anshori
172150100111013

KATA PENGANTAR

Puji dan syukur penulis panjatkan atas kehadirat Allah S.W.T. yang telah melimpahkan segala rahmat dan hidayah-Nya sehingga peulis dapat menyelesaikan tesis ini dengan judul "Klasifikasi DNA Tuberkolosis Berdasarkan K-Mer menggunakan *Support Vector Machine* (SVM) Dan *Variable Neighborhood Search* (VNS)" yang diajukan untuk menempuh ujian akhir Program Studi Magister Ilmu Komputer.

Dalam menyelesaikan penulisan tesis ini tidak terlepas dari peran serta dukungan dari berbagai pihak. Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih sebesar-besarnya kepada:

1. Bapak Wayan Firdaus Mahmudy, S.Si., Ph.D. dan Bapak Ahmad Afif Supianto, Dr.Eng., S.Si., M.Kom selaku dosen pembimbing penulis yang selali memberikan perbaikan serta arahan dengan sangat baik selama proses pengerjaan Tesis ini.
2. Ibu Dr. Eng. Fitri Utaminingrum, S.T., M.T., Bapak Dr. Eng. Fitra A. Bachtiar, S.T., M.Eng., selaku dosen penguji pada saat seminar proposal dan ujian akhir tesis yang selalu memberikan kritik dan saran agar kualitas tesis ini menjadi lebih baik lagi.
3. Orang tua beserta keluarga yang memberikan dukungan kepada pennulis sehingga dapat bersemangat untuk menyelesaikan tesis ini.
4. Teman-teman Magister Ilmu Komputer khususnya angkatan V.

Penulis menyadari bahwa dalam penyusunan laporan tesis ini masih banyak terdapat kekurangan. Oleh karena itu, peulis menyampaikan permohonan maaf sebelumnya, serta diharapkan kritik dan saran yang bersifat membangun dalam penyempurnaan di masa mendatang.

Malang, 13 September 2019

Mochammad Anshori
aanshori.moch@gmail.com

ABSTRAK

Tuberkulosis adalah penyakit yang disebabkan oleh *mycobacterium tuberculosis* dan termasuk kedalam salah satu dari 10 penyebab kematian di dunia. Oleh karena itu diperlukan pendeteksian secara lebih akurat supaya dapat diberikan penanganan yang tepat. Dalam pendeteksiannya, terkadang terjadi kesalahan karena menyerupai dengan penyakit paru-paru lainnya. Penelitian ini menerapkan algoritme *machine learning* dalam melakukan deteksi penyakit Tuberkulosis dengan menggunakan data DNA karena semua organisme memiliki struktur DNA. Metode yang digunakan adalah *support vector machine* (SVM) yang dioptimasi dengan *variable neighborhood search* (VNS). SVM digunakan untuk klasifikasi dan VNS digunakan untuk optimasi dari parameter SVM. SVM dipilih karena bagus dalam generalisasi data. Data DNA sebelum digunakan sebagai masukan kedalam SVM perlu dilakukan *preprocessing* terlebih dahulu dengan menggunakan k-Mer untuk mengambil *substring* DNA kemudian mengkonversinya menjadi data berupa numerik dan dilakukan reduksi dimensi karena fitur data yang banyak. Performa dari SVM tergantung dari pemilihan parameter yang tepat, oleh karena itu dioptimasi dengan VNS dan VNS yang digunakan adalah VNS yang telah dimodifikasi, yaitu *nested* RVNS. k-Mer terbaik pada penelitian ini bernilai k = 5. Hasil akhir setelah dilakukan optimasi adalah akurasi = 0.995708, presisi = 0.995765, *recall* = 0.995708, F *measure* = 0.995557, dan MCC = 0.992659. Akurasi ini lebih baik daripada sebelum dilakukan optimasi, yang bernilai 0.927039. Dengan menggunakan *nested* RVNS, berjalan 2.5 kali lebih cepat daripada VNS dasat dalam mencari parameter SVM yang optimal.

Kata kunci: *mycobacterium tubercolosis*, DNA, k-mer *support vector machine*, *variable neighborhood search*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Tuberkulosis merupakan sebuah penyakit yang diakibatkan oleh bakteri bernama *Mycobacterium Tuberculosis*. Tuberkulosis merupakan penyakit menular yang sudah banyak terjadi dan pada beberapa kasus dapat menyebabkan kematian. Tuberkulosis menyerang paru-paru dengan ciri penderita batuk parah, demam, dan nyeri pada dada. Kematian yang diakibatkan oleh penyakit ini menempati urutan kedua setelah HIV/AIDS (Fogel, 2015).

Di Indonesia sendiri, Tuberkulosis adalah menduduki posisi ke 4 sebagai penyebab kematian (Surya et al., 2017). Sedangkan negara Cina adalah satu dari 30 negara di dunia dengan tingkat terjangkit Tuberkulosis yang tinggi. Sekitar 900.000 kasus baru Tuberkulosis muncul dalam setiap tahunnya (Zhan et al., 2018). Terbukti bahwa pada beberapa negara dengan tingkat penduduk yang tinggi. Tuberkulosis menjadi penyakit yang paling mematikan. Pada beberapa kasus, kesalahan dalam diagnosis penyakit ini sering terjadi. Dalam pencitraan digital yang rumit, seringkali penyakit ini tidak terdeteksi dan menyerupai difus paru-paru lainnya (Zhan et al., 2018). Selain itu, deteksi bakteri secara manual seringkali memakan waktu dan membutuhkan keefisienan dalam menghasilkan diagnosa (Emmanuel, W R Sam; Mithra, 2017). Hal inilah yang membuat penelitian tentang Tuberkulosis masih terus dikembangkan.

Setiap organisme atau makhluk hidup memiliki *deoxyribonucleic acid* (DNA) dan *ribonucleic acid* (RNA), tidak terkecuali bakteri ataupun virus penyebab dari suatu penyakit. DNA adalah bahan genetik dari suatu sel (Yuwono, 2005). DNA dari organisme berbeda-beda yang menjadi identitasnya sendiri termasuk juga *Mycobacterium Tuberculosis* penyebab Tuberkulosis. DNA menjadi sebuah data pembawa informasi biologis dari suatu organisme. Penyusun DNA terdiri dari 4 jenis basa nitrogen dengan kode A, T, C, dan G, yaitu adenin, timin, sitosin, dan guanin (Yuwono, 2005).

Teknologi saat ini semakin berkembang. Dalam identifikasi data DNA dari suatu organisme dapat dilakukan dengan menerapkan salah satu teknologi terkini, yaitu *Machine Learning* (ML) dalam bidang Bioinformatika. Teknik ML berfungsi sebagai metode komputasi potensial untuk mengekstraksi pengetahuan dari data biologis. Eksperimen ML menjadi penting dalam bioinformatika karena dapat membantu dalam membangun model untuk deteksi pola menggunakan data biologis seperti genome, struktur protein, dll (Wassan, Wang and Zheng, 2019). Dahulu disiplin ilmu sebagian besar didapatkan dari hipotesis, kini menjadi ilmu yang digerakkan oleh data (Zucco, 2019). Dengan data yang menjadi masukan ke dalam sebuah ML dapat memberikan hipotesis tersendiri.

Data yang digunakan sebagai masukan dalam ML adalah data DNA. Panjang DNA dari suatu organisme yang berisikan 4 jenis basa nitrogen bisa mencapai ribuan pasang maka dari itu perlu ekstraksi fitur sebelum diolah oleh ML. Metode yang digunakan adalah k-Mer, yaitu string dengan panjang *k* yang dihasilkan dari alfabet. Dalam DNA, alfabet yang digunakan adalah urutan {A,T,C,G} (Ashlock and Datta, 2013). Panjang karakter yang akan diambil berdasarkan nilai dari *k*. Dengan menggunakan k-Mer dapat mengekstraksi fitur secara efektif dan dapat meningkatkan akurasi (Han and Cho, 2018). Oleh karena itu k-Mer akan digunakan pada penelitian ini sebagai metode dalam ekstraksi fitur dari data DNA.

Keluaran dari k-Mer adalah data berupa *substring* DNA dan perlu di ubah kedalam bentuk nilai numerik (Tripathy, Agrawal and Rath, 2016) karena masukan yang digunakan oleh suatu ML berupa data dalam bentuk numerik (Tripathy, Agrawal and Rath, 2016). Ada dua pendekatan yang dapat digunakan, yaitu TF-IDF dan *countvectorizer*. Kedua metode tersebut sama-sama dapat merubah data teks kedalam bentuk numerik. Cara kerja TF-IDF adalah dengan memberikan bobot kata yang paling sering muncul atau yang paling jarang digunakan, sedangkan *countvectorizer* hanya berdasarkan seberapa banyaknya kemunculan kata dalam suatu dokumen (Tripathy, Agrawal and Rath, 2015). Keluaran data dari TF-IDF berupa bobot *substring* DNA, dan keluaran dari *countvectorizer* berupa *sparse* matriks (Tripathy, Agrawal and Rath, 2016). Data keluaran tersebut nantinya akan digunakan sebagai masukan dari *Machine Learning*. Pada penelitian ini akan menggunakan pendekatan-pendekatan tersebut dan mengujinya untuk mengetahui metode yang memiliki kinerja paling baik.

Data keluaran dari k-Mer memiliki variasi yang sangat banyak tergantung dari nilai *k* yang digunakan dan setelah dilakukan ektraksi fitur menggunakan TF-IDF ataupun *countvectorizer* memiliki dimensi data yang besar dan perlu dilakukan reduksi dimensi. Terdapat dua metode yang umum digunakan dalam reduksi dimensi, yaitu *Principal Component Analysis* (PCA) dan *Linear Discriminant Analysis* (LDA). PCA adalah suatu algoritme yang dapat mereduksi dimensi data yang tinggi dan dapat mengurangi beban dari memori (Zhang and Yang, 2016). PCA dapat memproyeksikan data menjadi dimensi yang lebih rendah dari yang sebelumnya berdimensi tinggi (Omurca and Ekinci, 2015; He, He and Wei, 2017). LDA adalah kebalikan dari PCA, yang berbasis *supervised learning* dalam melakukan reduksi dimensinya (Xie, 2015). LDA menghitung arah diskriminan linier antar kelas kemudian merepresentasikan sumbu dan memaksimalkan pemisahan (Ilias et al., 2016). Penelitian sebelumnya ada juga yang membandingkan dua metode tersebut, yaitu Untuk diagnosis kesalahan pada industri modern (Xie, 2015) dan ekstraksi fitur dari data kiprah autisme (Ilias et al., 2016). Selain itu, sebagai perbandingan untuk menentukan metode yang terbaik akan digunakan PCA dan LDA pada penelitian ini.

Penelitian sebelumnya telah berhasil menerapkan algoritme-algoritme ML pada data DNA. Ada beberapa tipe dari ML salah satunya adalah *supervised learning*, yaitu model prediksi yang data masukan dan label kelasnya telah ditentukan sebelumnya (Wassan, Wang and Zheng, 2019). Beberapa diantaranya adalah *Random Forest* (Bobak, Titus and Hill, 2019), *Support Vector Machine* (Bobak, Titus and Hill, 2019; Li et al., 2017; Satpute and Yadav, 2018; Kristensen and Guillaume, 2015; Phan et al., 2017; Iqbal et al., 2014; Wang, Herbster and Mian, 2018), *Deep Neural Network* (Nakano et al., 2018; Fiannaca et al., 2018), *Random Forest* (Mamatjan et al., 2017; Iqbal et al., 2014; Phan et al., 2017; Bobak, Titus and Hill, 2019), ANN (Satpute and Yadav, 2018; Iqbal et al., 2014), *Convolutional Neural Network* (Fiannaca et al., 2018; Nguyen et al., 2016), LSTM (Antonino and Gangi, 2017), dan MLP(Kristensen and Guillaume, 2015). Dari sekian banyak algoritme, pada penelitian ini SVM digunakansebagai metode klasifikasi.

Algoritme ML yang sering digunakan dalam klasifikasi adalah *Support Vector Machine* (SVM). Penelitian terdahulu telah menerapkan SVM untuk data urutan DNA salah satunya adalah untuk identifikasi gen esensial dan gen tidak esensial dari 5 bakteri. Hasil AUC tertinggi mencapai 0.8751 dan rata-rata nilainya adalah 0.8174 (Li et al., 2017). Penelitian lainnya adalah untuk mengklasifikasikan sel eukoriotik dan prokariotik dengan hasil akurasi bernilai 84.6% (Kristensen and Guillaume, 2015), dan dengan SVM menghasilkan tingkat kesalahan yang rendah, yaitu 0.006 untuk klasifikasi genom virus (Wang, Herbster and Mian, 2018). Sebagai metode pembanding, SVM lebih baik daripada metode PLS-DA (*Partial Least Square-Discrimant Analysis*) dengan akurasi SVM adalah 88.75%, sedangkan PLS-DA adalah 85.24% (Bobak, Titus and Hill, 2019). SVM juga dapat menghasilkan nilai *specificity* lebih baik daripada metode pembanding yg digunakan yaitu sebesar 97% dan pembandingnya bernilai 95% (Niehaus et al., 2014). Terbukti bahwa SVM mampu dalam menangani klasifikasi pada data DNA.

Walaupun handal dalam menyelesaikan masalah klasifikasi, kinerja SVM sangat bervariasi tergantung dari pemilihan parameter yang ditetapkan (Vahdani et al., 2017). Perbedaan dalam memberi nilai pada parameter sangat mempengaruhi seberapa baik model *classifier* yang dibangun. Berdasarkan penelitian terdahulu terdapat beberapa parameter SVM yang dapat dioptimasi, yaitu C, dan γ (Huang and Wang, 2017; Liu, He and Cui, 2018). Pada penelitian terdahulu dalam menentukan kesalahan pada *traction converter* didapatkan nilai $C = 32$ dan $\gamma = 0.00035$ dengan akurasi sebesar 97.78% (Jin et al., 2014). Dengan $C = 45.253$ dan $\gamma = 8$ adalah nilai parameter optimal yang didapatkan untuk verifikasi dataset dari gas dan akurasi yang didapatkan sebesar 99.2991% (Liu and Du, 2016). Dalam klasifikasi audio didapatkan nilai $C = 2^{5.6}$ dan $\gamma = 2^{-9.8}$ dengan kernel RBF dan akurasi yang didapatkan sebesar 99.72% (Grama et al., 2017). Tidak ada aturan umum untuk mengatur parameter SVM (Vahdani et al., 2017). Dalam

menentukan parameter SVM dapat dilakukan dengan algoritme pencarian. Salah satu solusi untuk memaksimalkan parameter tersebut adalah dengan menerapkan mode optimasi metaheuristik sebagai alternatif yang lebih efisien (Morales, 2014). Terdapat bermacam-macam algoritma pencarian yang umum digunakan berbasis populasi seperti *Particle Swarm Optimization* (PSO), Artificial Bee Colony(ABC), *Genetic Algorithm* (GA), dan Firefly Algorithm. Ada juga yang berbasis solusi tunggal seperti *Tabu Search*, *Simulated Annealing*, dan *Variable Neighborhood Search* (VNS). VNS adalah salah satu algoritme metaheuristik yang digunakan dalam hal optimasi.

VNS sebagai metaheuristik terus mengeksplorasi ruang pencarian untuk mencapai solusi yang lebih baik. Secara sistematis mengubah struktur lingkungan untuk meningkatkan kemampuan pencarian lokal (Chengli et al., 2018). Karena eksplorasi global yang baik dan pencarian lokal yang intensif, VNS cenderung mendapat solusi yang sangat baik (Meng et al., 2018). VNS memiliki sedikit parameter yang membuatnya efektif (dalam hal menemukan solusi), dan melakukan pencarian yang efisien (dalam menggunakan sumber daya komputasi yang minimum) (Belhaiza, M'Hallah and Brahim, 2017). Berdasarkan penelitian terdahulu, hal-hal tersebut yang menjadi kelebihan dari VNS.

Penelitian terkait dengan dengan VNS telah terbukti dapat memberikan solusi pada beberapa permasalahan seperti *Traveling Salesman Problem* (Hore, Chatterjee and Dewanji, 2018; Meng et al., 2018; Bai, 2015), *VRP with Multiple Time Windows* (Ferreira et al., 2018; Belhaiza, M'Hallah and Brahim, 2017), *CVRP* (Amous et al., 2017), dan penjadwalan (Kocatürk and Özpeynirci, 2014; Thomas and Manni, 2014). VNS yang berbasis pencarian lokal juga dapat digunakan untuk optimasi model klasifikasi, terbukti dari penelitian terdahulu dengan VNS menentukan koefisien untuk membangun *hyperplane* (Caporossi, 2005). VNS mampu menangani masalah optimasi dan dapat diterapkan dalam model klasifikasi.

Berdasarkan penelitian sebelumnya dengan menggunakan VNS dapat meningkatkan akurasi menjadi 86.38% dengan akurasi terkecil adalah 85.94% (Boughaci and Alkhawaldeh, 2018). Sedangkan penelitian lain menunjukkan bahwa dengan hibridisasi ini dapat menurunkan nilai kesalahan dibandingkan dengan tanpa hibridisasi (Vahdani et al., 2017). Karena telah terbukti VNS dapat digunakan untuk optimasi, pada penelitian ini VNS akan digunakan sebagai optimator dalam menentukan parameter dari SVM.

Berdasarkan studi pustaka yang telah dilakukan, k-Mer dapat diterapkan dalam ekstraksi fitur dari untaian DNA. Setelah itu akan diterapkan reduksi fitur dengan PCA dan LDA. Harapannya dapat mereduksi waktu komputasi dan meningkatkan performa klasifikasi. Metode klasifikasi yang digunakan adalah SVM yang telah terbukti memberikan hasil yang baik. Akan tetapi masih dapat

dioptimalkan dengan menentukan nilai parameter yang tepat. Dalam menentukan parameternya, dapat dengan menerapkan algoritme metaheuristik yaitu VNS. VNS digunakan karena memiliki sedikit parameter dan mampu mengesplorasi ruang pencarian. Akan tetapi VNS ada kemungkinan untuk dapat ditingkatkan kemampuannya dengan memodifikasi algoritme tersebut. Modifikasi dilakukan untuk mendapatkan waktu komputasi yang lebih cepat dari VNS dasar akan tetapi tetap menjaga hasil nilai kebugarannya. Harapan dari kombinasi algoritme ini akan memberikan peningkatan hasil dengan waktu komputasi yang lebih baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka dapat diuraikan rumusan masalah sebagai berikut.

1. Bagaimana pengaruh perubahan nilai *k* dari k-Mer dalam ekstraksi fitur dari untaian DNA terhadap akurasi?
2. Bagaimana pengaruh PCA dan LDA terhadap akurasi dan waktu komputasi?
3. Bagaimana pengaruh perubahan parameter *C* serta *gamma* terhadap akurasi klasifikasi dari SVM?
4. Seberapa tinggi akurasi SVM apabila VNS diterapkan sebagai optimasi parameter?
5. Bagaimana hasil dari modifikasi VNS dalam mengoptimalkan parameter SVM?

1.3 Tujuan

Dalam penelitian ini terdapat beberapa tujuan sebagai berikut.

1. Menerapkan k-Mer untuk mendapatkan fitur dari data untaian DNA.
2. Mengetahui pengaruh PCA dan LDA terhadap waktu komputasi dan hasil dari klasifikasi SVM.
3. Menerapkan VNS untuk menentukan parameter yang sesuai pada SVM.
4. Mengetahui pengaruh perubahan parameter dan mendapat parameter terbaik SVM.
5. Mengetahui perbedaan hasil klasifikasi dari SVM dengan optimasi menggunakan VNS dasar dan VNS yang telah dimodifikasi.

1.4 Manfaat

Terdapat beberapa manfaat yang bisa didapatkan dari penelitian ini sebagai berikut.

1. Model algoritme yang dihasilkan pada penelitian ini diharapkan dapat menjadi referensi bagi peneliti lain dalam ruang lingkup bioinformatika.

2. Dapat menjadikan penerapan SVM dengan VNS sebagai metode alternatif dalam deteksi penyakit berdasarkan DNA.
3. Membantu bidang kesehatan dalam hal mendeteksi penyakit Tuberkulosis sehingga dapat mendeteksi secara lebih akurat.

1.5 Batasan Masalah

Terdapat batasan masalah yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Data yang digunakan didapatkan dari *National Center for Biotechnology Information* (NCBI), yaitu situs yang menyediakan data dan informasi tentang genom dengan halaman website https://www.ncbi.nlm.nih.gov/nuccore/.
2. Data yang digunakan adalah data dengan jenis fasta.
3. Data yang digunakan adalah *complete genome* DNA dari *Mycobacterium Tuberculosis*, tidak termasuk data protein.
4. Kernel dari SVM yang akan digunakan adalah *radial basis function* (RBF).
5. Parameter SVM yang akan dioptimasi adalah C, dan γ.

1.6 Sistematika Pembahasan

Adapun sistematika pembahasan guna mempermudah dalam melakukan penulisan tesis. Secara rinci, sistematika pembahasan isi penelitian ini adalah sebagai berikut.

1. BAB I PENDAHULUAN

   Dalam bab ini berisi tentang latar belakang masalah, rumusan masalah, tujuan penulisan, manfaat penelitian, batasan masalah pada penelitian ini, dan juga sistematika pembahasan.

2. BAB II LANDASAN KEPUSTAKAAN

   Dalam bab ini berisi landasan teori yang berhubungan dengan penelitian ini. Pada *preprocessing* ada k-Mer untuk ekstraksi data DNA, TF-IDF dan *countvectorizer* ekstraksi fitur. Ada juga Principal *Component Analysis* dan *Linear Discriminant Analysis* yang digunakan untuk reduksi dimensi data. *Suppport Vector Machine* sebagai algoritme dalam deteksi dan *Variable Neighborhood Search* yang digunakan untuk optimasi. Selain itu juga ada berisi landasan kepustakaan mengenai Tuberkulosis dan hal-hal lain yang masih berhubungan dengan penelitian ini.

3. BAB III METODOLOGI

   Dalam bab ini membahas tahapan penelitian untuk klasifikasi penyakit Tuberkulosis yang akan dilakukan. Termasuk didalamnya tahap *preprocessing*, optimasi SVM menggunakan VNS, dan modifikasi VNS.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tuberkulosis

Tuberkulosis merupakan suatu penyakit yang disebabkan oleh *Mycobacterium Tuberculosis*. Ilmuwan bernama Robert Koch pada tahun 1882 adalah orang yang pertama kali menemukan kuman penyebab Tuberkulosis. Tuberkulosis sudah ada sejak jaman Mesir kuno yang berarti ribuan tahun sebelum masehi. Dahulunya pada tahun 1980-an orang-orang percaya bahwa Tuberkulosis dapat dihilangkan pada akhir abad ke-20, akan tetapi Tuberkulosis muncul menjadi ancaman kesehatan secara global pada abad ini (Asia et al., 2015). Walaupun telah ditemukan antibiotik untuk mengurangi angka kematian akan tetapi kekebalan penyakit ini muncul perlahan-lahan.

World Health Organization (WHO) menyatakan beberapa fakta terkait Tuberkulosis sebagai berikut:

- Tuberkulosis adalah salah satu dari 10 penyebab utama kematian didunia.
- Pada tahun 2017, 10 juta orang sakit dengan mengidap Tuberkulosis dan 1.6juta meninggal (termasuk 0.3 juta diantaranya dengan HIV).
- Pada tahun 2017, diperkirakan 1 juta anak jatuh sakit dengan Tuberkulosis dan 230.000 anak meninggal karenanya.
- Tuberkulosis adalah pembunuh utama orang dengan HIV positif.
- Tuberkulosis yang tahan dengan obat menjadi ancaman kesehatan masyarakat. WHO memperkirakan ada 558.000 kasus baru dengan resistansi terhadap obat yang paling efektif.
- Secara global kejadian Tuberkulosis menurun sekitar 2% per tahun. Ini perlu dipercepat hingga 4-5% pada tahun 2020.
- Mengakhiri epidemi Tuberkulosis pada tahun 2030 adalah salah satu target kesehatan.

Kematian akibat Tuberkulosis pernah menduduki tertinggi ketiga didunia (Li, Huang and Jin, 2018). Hal ini menunjukkan bahwa penyakit ini menjadi ancaman serius bagi kesehatan. Walaupun begitu Tuberkulosis masih dapat untuk dicegah. Orang dewasa memiliki resiko paling tinggi terserang Tuberkulosis, yaitu sekitar 95% kasus dan kematian di negara berkembang. Orang dewasa yang mengkonsumsi tembakau lebih tinggi resiko terserang Tuberkulosis dan 7.9% kematian akibat Tuberkulosis disebabkan oleh merokok.

Dalam penyebarannya, Tuberkulosis menular melalui batuk seseorang yang terjangkit. Ketika orang yang mengidap Tuberkulosis batuk, bersin, meludah sekaligus mengeluarkan kuman penyebab Tuberkulosis ke udara. Dengan begitu apabila ada orang lain yang menghirup udara yang mengandung kuman ini dapat terinfeksi

Tuberkulosis dapat menyebar lintas negara melalui inangnya yang bermigrasi sebagaimana yang terjadi di negara Amerika Serikat (Abubakar et al., 2018). Amerika memiliki sejarah panjang dalam menerima pendatang dari negara lain. Dampaknya adalah penyebaran Tuberkulosis meluas dan Amerika Serikat sekarang mewakili beberapa kasus baru dari Tuberkulosis. Berdasarkan hasil analisis, terdapat 30 orang dari negara asal yang membawa Tuberkulosis ke Amerika Serikat antara tahun 2003 dan 2015.

Tuberkulosis terjadi disetiap bagian dunia, pada tahun 2017 kasus Tuberkulosis terbesar terjadi di wilayah Asia Tenggara dan Pasifik Barat dengan 62% kasus baru. Delapan negara menyumbang dua pertiga kasus Tuberkulosis baru adalah India, Cina, Indonesia, Filipina, Pakistan, Nigeria, Bangladesh, dan Afrika Selatan.

Di Indonesia sendiri terdapat faktor-faktor penularan Tuberkulosis yaitu sebagai berikut:

a. Faktor lingkungan fisik rumah
b. Faktor perilaku penduduk seperti membuang dahak sembarangan, bersin tanpa menutup mulut, dan kebiasaan merokok.

Organ yang paling sering diserang oleh Tuberkulosis adalah paru-paru, akan tetapi Tuberkulosis juga dapat menyerang organ lain dari penderitanya, seperti otak, liver, lambung, dan payudara. Terdapat kasus dimana Tuberkulosis menginfeksi otak sebagai saraf pusat manusia. Apabila telah menyerang otak dapat menyebabkan komplikasi serius. Hal seperti inilah yang kadang susah untuk didiagnosa. Tuberkulosis liver pernah terjadi tetapi jarang. Kejadian ini terjadi pada anak berusia 4 tahun yang mengidap Tuberkulosis paru juga. Tuberkulosis lambung sangat langka terjadi. Terdapat 4 buah kasus yang terjadi selama 5 tahun (Chaudhary et al., 2018). Tuberkulosis payudara adalah Tuberkulosis yang paling tidak umum dan jarang sekali dilaporkan. Tuberkulosis ini seringkali muncul pada wanita muda dan terbentuk benjolan pada payudaranya tanpa rasa sakit (Sinha, 2018).

2.2 K-MER

Setiap organisme memiliki urutan DNA yang berbeda-beda. Data DNA dari organisme khususnya pada *Mycobacterium Tuberculosis* bisa memiliki panjang data yang mencapai ribuan dan hanya berisikan 4 jenis basa nitrogen, yaitu A, C, T dan G. Pada Tabel 2.1 merupakan contoh dari salah satu jenis dari *Mycobacterium Tuberculosis*. Karena data hanya berisikan deretan dari 4 jenis karakter, oleh karena itu perlu dilakukan *preprocessing* sebelum diolah oleh suatu metode ML untuk ekstrasi fitur dari data DNA, salah satunya adalah dengan k-Mer.

Tabel 2. 1 Contoh DNA Tuberkulosis

Nama	DNA	Panjang
Mycobacterium tuberculosis strain CAS	TTGACCGATGACCCCGGTTCAGGCTTCACCACAGT GTGGAACGCGGTCGTCTCCGAACTTAACGGCGAC C…TGTGGACAGTTCACCTGCCCACAACAACGGTTG TAGCTCGACCCGGAACCAAGACCCGGAACTAACG AGAACCAGGGAGATACGTCG	62780 karakter

K-Mer adalah sebuah cara untuk mengambil substring dengan panjang *k* yang terkandung di dalam sebuah string. Dengan k-Mer dapat mengambil informasi yang terkandung dari DNA. Dalam komputasi genomic, k-Mer merujuk pada semua kemungkin karakter berikutnya dalam string dari urutan DNA. *Words* adalah banyaknya substring yang didapat dari proses k-Mer, seperti yang dirumuskan pada Persamaan (2.1).

$$words = L - k + 1 \quad (2.1)$$

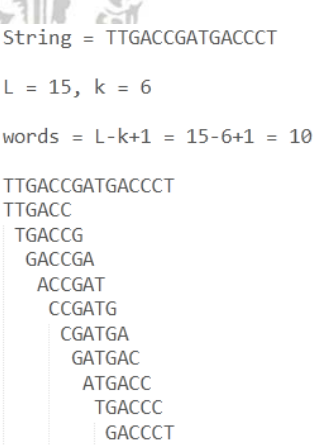
Dengan notasi:

$words$ = banyaknya substring

L = panjang string

k = panjang substring

Sebagai contoh dari penerapan k-Mer dapat dilihat pada Gambar 2.1. Diketahui ada sebuah string "TTGACCGATGACCCT" dengan L = 15 dan k = 6. Dengan Persamaan 2.1 didapatkan banyaknya *words* = 10.



Gambar 2. 1 Contoh penerapan k-Mer

Dalam memecah string, k-Mer bekerja seperti *sliding window* yang bergeser dengan panjang k yang telah ditentukan. Didapatkan substring yaitu $\{(TTGACC), (TGACCG), (GACCGA), \ldots (GACCCT)\}$. Substring tersebut yang menjadi fitur dari urutan DNA dan yang akan digunakan sebagai masukan pada sebuah algoritme *Machine Learning*. Tidak ada aturan khusus dalam menentukan

nilai *k*, dengan pemberian nilai yang tinggi juga dapat menghasilkan akurasi yang tinggi (Dubinkina et al., 2016). Akan tetapi dengan nilai *k* yang rendah tidak menutup kemungkinan memberikan hasil yang rendah. Perbedaan nilai k adalah kunci yang mempengaruhi spesifitas untuk perbandingan (Dubinkina et al., 2016).

2.3 Ekstraksi Fitur

Ada 2 pendekatan yang digunakan dalam ekstraksi fitur dari data yang telah diolah menggunakan k-Mer, yaitu TF-IDF dan Countvectorizer. Metode yang digunakan untuk merubah data yang berbentuk teks menjadi numerik dan untuk menyeragamkan panjang data sebelum masuk ke dalam algoritme *Machine Learning*.

2.3.1 *Term Frequency - Inverse Document Frequency* (TF-IDF)

Term Frequency – Inverse Document Frequency (TF-IDF) adalah pemberian bobot kata dari suatu dokumen (Pan and Tang, 2015) berdasarkan frekuensi kemunculan kata dari dokumen-dokumen. *Term Frequency* (TF) yaitu frekuensi banyaknya kemunculan kata dari tiap dokumen. *Inverse Document Frequency* (IDF) digunakan untuk menghitung bobot kata-kata yang berbeda dari dalam dokumen. Kata yang jarang sekali muncul memiliki nilai IDF yang tinggi. Cara menghitung TF-IDF dapat dilihat pada Persamaan (2.2) dibawah:

$$tfidf(t_j, d_i) = tf(t_j, d_i) \times (\log \frac{N}{df(t_j)}) + 1 \quad (2.2)$$

dimana $tf(t_j, d_i)$ adalah frekuensi kemunculan kata t_j dalam spesifik dokumen, $df(t_j)$ adalah banyaknya dokumen dimana t_j itu muncul dan N adalah nilai keseluruhan banyaknya dokumen. Berdasarkan Persamaan 2.2 diatas, rumus IDF = $(\log \frac{N}{df(t_j)}) + 1$. Penambahan 1 supaya nilai IDF tidak bernilai 0. Apabila tanpa penambahan 1, maka hasil perkalian TF dengan IDF akan menghasilkan nilai 0.

2.3.2 *Countvectorizer*

Countvectorizer merupakan salah satu cara untuk memberi bobot kata dalam suatu dokumen. Cara kerjanya dengan merubah teks menjadi matrik dari perhitungan token yang berasal dari perhitungan kemunculan kata (Tripathy, Agrawal and Rath, 2016). Sebagai contoh, misalkan ada 3 buah dokumen sebagai berikut:

Dokumen 1 = “pemrograman itu menyenangkan”

Dokumen 2 = “pemrograman itu asik”

Dokumen 3 = “pemrograman itu menantang”

maka hasil *countvectorizer* seperti pada Tabel 2.2 dibawah ini. Nilai-nilai pada tiap sel tersebut yang menjadi bobot kata dan bobot tersebut yang akan menjadi masukan kedalam sebuah mesin pembelajaran.

Tabel 2. 2 Contoh *countvectorizer*

	pemrograman	itu	menyenangkan	asik	menantang
Dokumen 1	1	1	1	0	0
Dokumen 2	1	1	0	1	0
Dokumen 3	1	1	0	0	1

Perhitungan banyaknya kemunculan data ini mirip dengan *term frequency* (TF) pada TF-IDF.

2.4 Normalisasi

Data baru telah didapatkan dan data mengandung nilai dengan rentang yang sangat berbeda. Normalisasi diperlukan untuk menormalkan fitur data yang telah di ekstraksi (Nuraisha, 2018). Dengan normalisasi untuk menskalakan data dengan rentang nilai yang sama (Gajera et al., 2016). *Min-Max normalization* adalah salah satu metode yang dapat digunakan dalam menormalisasikan data.

Dengan normalisasi *Min-Max* mentransformasi data dan menghasilkan rentang data dengan nilai tertinggi dan terendah menjadi data yang normal (Nuraisha, 2018). Dengan normalisasi ini juga mempengaruhi efisiensi dan ketepatan klasifikasi, serta meningkatkan akurasi dari Jaringan Syaraf Tiruan dan mempercepat proses pelatihannya (Dinç et al., 2014). Perhitungan dalam normalisasi *Min-Max*, dapat dilihat pada Persamaan (2.3) berikut:

$$x' = (max - min) \times \frac{x - minX}{maxX - minX} + min \quad (2.3)$$

x' = data normalisasi

x = data yang akan dinormalisasi

max = nilai maksimal yang diberikan

min = nilai minimal yang diberikan

maxX = nilai maksimal data per kolom

minX = nilai minimal data per kolom

Apabila data akan dijadikan dengan rentang 0 hingga 1, maka Persamaan (2.2) diatas dapat diubah menjadi Persamaan (2.4) dibawah ini:

$$x' = \frac{x - minX}{maxX - minX} \quad (2.4)$$

Dengan begitu data hasil normalisasi akan berada pada rentang 0 hingga 1. Hasil transformasi normalisasi *Min-Max*, masih tetap menjaga relasi dengan data aslinya (Patro and Kumar, 2015) sehingga informasi yang terkandung tidak hilang.

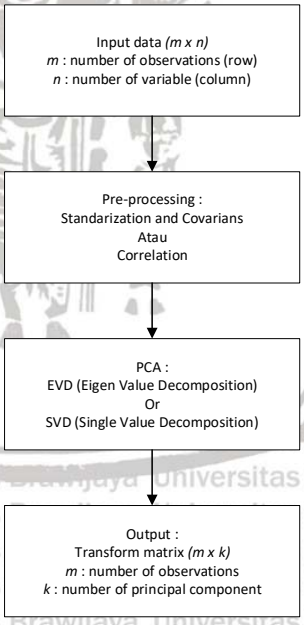
2.5 Reduksi Dimensi

Data hasil dari ekstraksi fitur memiliki dimensi yang cukup besar, oleh karena itu akan diterapkan reduksi dimensi untuk mengurangi dimensi datanya. Reduksi dimensi disini berguna untuk mempercepat proses klasifikasi nantinya. Ada 2 pendekatan yang digunakan sebagai perbandingan, yaitu PCA dan LDA.

2.5.1 *Principal Component Analysis* (PCA)

Principal Component Analysis (PCA) adalah sebuah metode yang digunakan untuk mengurangi dimensi data yang tinggi. Data berdimensi tinggi adalah data dengan jumlah fitur dan variable yang banyak. Dengan PCA dapat mengurangi dimensi data dengan masih menjaga informasi yang terkandung di dalamnya. Kelebihan dari PCA dapat mengurangi waktu komputasi dan akurasi masih tetap stabil.

Proses dari PCA dengan mengkombinasikan secara linier dari informasi asli. Data yang diproses mencakup sebagian besar informasi yang terkandung dari variabel asli (Liu, 2015). Gambar 2.2 adalah langkah-langkah PCA secara umum.



Gambar 2. 2 Langkah-langkah PCA secara umum (Rizanti et al., 2016)

Berdasarkan Gambar 2.2, misalkan data masukan asli dengan ukuran $m \times n$, $A_{m \times n} = [a_1, a_1, \dots, a_n]$. Lakukan *preprocessing* pada matrik A dengan standarisasi nilai dan didapatkan $X_i, i = 1, 2, \dots, n$. PCA melakukan linier transformasi pada matrik X dengan Persamaan 2.5.

$$X = t_1 p_1^T + t_2 p_2^T + \cdots + t_k p_k^T + E \quad (2.5)$$

Dimana $k \leq \min(m, n)$, t_i adalah *score vector*, p_i adalah *loading vector*, E adalah eror matrik, matrik T mengikuti $T'T = I$. Kemudian untuk menghitung kovarian dengan mengikuti Persamaan 2.6.

$$COV(X) = (m-1)^{-1} \sum_{i=1}^{m} (X_i - \bar{X})(X_i - \bar{X})^T \quad (2.6)$$

Selanjutnya dilakukan EVD (*Eigen Value Decomposition*) atau SVD (*Single Value Decomposition*) sebagai proses PCA dari matrik kovarian. *Eigenvectors* V adalah proyeksi dari R dengan menggunakan Persamaan 2.7.

$$RV = \Lambda V$$
$$|R - \lambda I| = 0 \quad (2.7)$$

dimana Λ adalah *eigenvalues* dan didapatkan dengan menyelesaikan Persamaan 2.8 dibawah. Dimana $\lambda(i = 1, 2, \dots, n)$ adalah *eigenvalues,* I adalah matrik identitas.

$$\Lambda = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_k] \quad (2.8)$$

Setelah menentukan matrik V, *principal component* adalah data yang berupa dimensi rendah dan ditentukan dengan Persamaan 2.9 dibawah ini.

$$F = v_{i1}x_1 + v_{i2}x_2 + \cdots + v_{ik}x_k \quad (2.9)$$

Dimana x_i adalah variabel masukan, $v_i(i = 1, 2, \dots, n)$ adalah *eigenvectors* dan F adalah dataset baru. Data keluarannya berupa matrik dengan ukuran $m \; x \; k$ dengan k adalah jumlah *principal component* yang digunakan. Secara umum pemilihan jumlah *principal component* yang baik adalah yang mengandung lebih dari 85% varian dari data asli (Zhongwen and Huanghuang, 2017).

2.5.2 *Linear Discriminant Analysis* (LDA)

Metode lainya untuk reduksi dimensi adalah Linear Discriminant Analysis (LDA). Sedikit berbeda dengan PCA karena LDA mempertimbangkan kelas data. LDA memproyeksikan data menjadi ruang vektor dengan memaksimalkan jarak antar kelas dan meminimalkan jarak dalam kelas data (Xie, 2015).

Ada tiga tahap utama pada LDA dalam mereduksi dimensi data, yaitu menghitung jarak dalam kelas dengan rumus pada Persamaan 2.10 dan Persamaan 2.11, jarak antar kelas data dengan rumus pada Persamaan 2.12, dan membangun ruang dimensi yang lebih rendah dengan rumus pada Persamaan 2.13, dimana memaksimalkan varian antar kelas dan meminimalkan varian dalam kelas.

$$S_i = \sum_{x \in X_i} (x - m_i) \cdot (x - m_i)^T \quad (2.10)$$

dan

$$S_w = \sum_{i=1}^{C} S_i \quad (2.11)$$

dimana m_i adalah rata-rata data dengan kelas yang sama.

$$S_B = \sum_{i=1}^{N} n_i (m_i - m) \cdot (m_i - m)^T \quad (2.12)$$

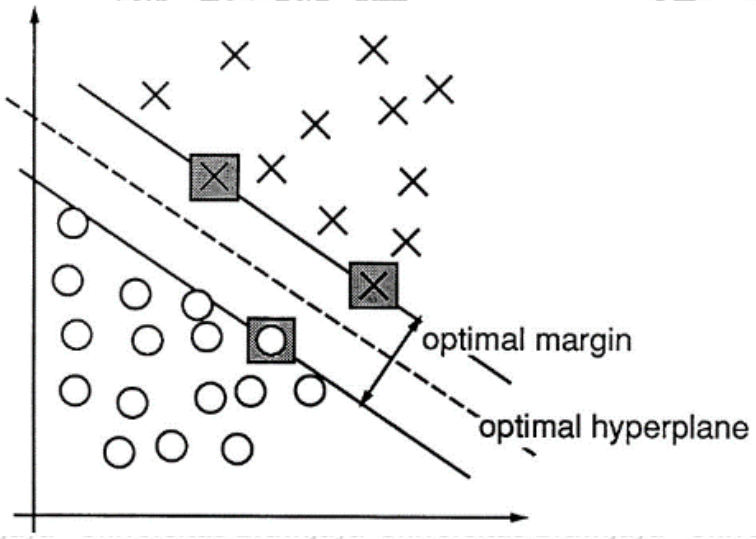
dimana $n_i$adalah banyaknya sampel pada kelas yang sama dan m adalah rata-rata dari keseluruhan sampel.

$$S_B \vee = \wedge S_w \vee \quad (2.13)$$

dimana ∨ adalah *eigenvector* dan ∧ adalah *eigenvalues*. *Eigenvalues* yang bernilai bukan nol menunjukkan kontribusinya dalam mencakup informasi dari data.

2.6 Support Vector Machine (SVM)

Data yang telah dilakukan *preprocessing* sebelumnya telah siap diolah oleh sebuah *Machine Leaning*, algoritme yang akan digunakan adalah SVM. SVM termasuk ke dalam *supervised learning* karena sudah ada kelas yang ditentukan. SVM merupakan salah satu metode dalam *Machine Learning* untuk melakukan tugas klasifikasi ataupun regresi yang diusulkan pertama kali oleh (Boser, Bernhard E. and Guyon, Isabelle M. and Vapnik, 1992). SVM dengan pembelajarannya membuat satu set hyperplane dalam ruang berdimensi tinggi berdasarkan analisis data (Ding et al., 2017). Kinerja SVM dapat dihandalkan dalam klasifikasi linier ataupun non-linier, SVM telah banyak diteliti dan digunakan untuk memecahkan berbagai macam masalah termasuk data yang berdimensi tinggi. Kelebihan lain dari SVM adalah dapat menghindari permasalahan *overfitting* (Demidova, Nikulchev and Sokolova, 2016)



Gambar 2. 3 Contoh permasalahan pada ruang 2 dimensi (Cortes and Vapnik, 1995)

Teknik SVM pada mulanya dikembangkan untuk memisahkan data tanpa adanya eror dengan cara memaksimalkan *margin* antara data latih dan kelasnya (Cortes and Vapnik, 1995). Maksimal margin dengan menentukan *hyperplane*

terlebih dahulu, tersaji pada Gambar 2.3. Setelah *hyperplane* ditentukan, kemudian membuat margin yang optimal.

Dengan memaksimalkan margin, SVM memiliki kemampuan untuk generalisasi data yang tinggi. Tujuan dari SVM adalah untuk menemukan *hyperplane* terbaik yang memisahkan data-data dengan kelasnya yang. Didefinisikan pada Persamaan 2.14.

$$w.x + b = 0 \quad (2.14)$$

x adalah himpunan data latih dengan notasi $x_i \in R^d$, dengan label kelas sebagai $y_i \in -1,1$ untuk i = 1, 2, …, l dimana l adalah jumlah data dan w adalah bobot dan b adalah bias.

Data x_i diklasifikasikan sebagai kelas -1 apabila

$$w.x_i + b \leq -1 \quad (2.15)$$

dan diklasifikasikan sebagai kelas +1 apabila

$$w.x_i + b \geq 1 \quad (2.16)$$

Ketika data tidak terpisah secara linier, *hyperplane* didapatkan dengan variabel *slack* $\{\xi\}_{i=1}^{N}$ akan tetapi ada masalah kuadratik dan dapat diselesaikan dengan Persamaan berikut

$$min\frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \xi_i \quad (2.17)$$

dengan $y_i(wx_i + b) \geq 1 - \xi_i, \xi_i > 0 \; for \; i = 1, \dots, N$

Dimana C adalah parameter pinalti. Semakin besar nilai C semakin besar pula pinalti kesalahan prediksi. Untuk memaksimalkan Persamaan diatas dapat dengan menerapkan *lagrange multipliers* λ dan mengikuti aturan *Karush-Kuhn-Tucker* (KKT)

$$\max L\{\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \lambda_i \lambda_j K(x_i x_j) \quad (2.18)$$

dengan $C \geq \lambda_i \geq 0 \; \forall i = 1, \dots, N$, dan $\sum_{i=1}^{N} \lambda_i \gamma_i = 0$

Dasar dari SVM dalam pembelajarannya adalah untuk menemukan *support vector* yang bergantung pada *dot product* dari data, yaitu $\Phi_i \Phi_j$. *Support* vector adalah data yang posisinya mendekati kelas data yang berbeda. Transformasi Φ tidak diketahui dan susah dipahami maka dari itu perhitungan ini disebut juga dengan fungsi kernel $K(x_i x_j)$. Sebagai contoh kernel linier diformulasikan seperti pada Persamaan dibawah.

$$K(x_i x_j) = \Phi_i(x_i).\Phi_j(x_j) \quad (2.19)$$

Awal mulanya SVM didesain untuk menyelesaikan masalah linier, tetapi karena kebanyakan masalah tidak terpisah secara liner maka faktor *C* dan fungsi kernel ditambahkan.

2.6.1 Kernel SVM

Pada SVM di dalamnya terdapat kernel yang membantu SVM dalam mempelajari data. Dengan kernel, SVM menciptakan *hyperplane* yang paling sesuai dalam menentukan *support* vector. Terdapat 4 kernel yang umumnya SVM gunakan, yaitu sebagai berikut:

a. Linear

$$K(x_i, x_j) = (x_i . x_j) \quad (2.20)$$

Kernel ini paling cocok digunakan untuk kasus data yang kelas data terpisah secara linier. Waktu komputasi yang dibutuhkan cenderung lebih cepat. Tetapi kernel ini lemah dalam menghadapi kelas yang terpisah secara tidak linier

b. Polynomial

$$K(x_i, x_j) = (x_i . x_j + 1)^d \quad (2.21)$$

Kernel ini dapat digunakan untuk data yang terpisah secara tidak linier, menutupi kekurangan dari kernel linier. Tetapi waktu komputasi yang dibutuhkan lebih lama dari kernel linier karena ada perhitungan kuadrat didalamnya.

c. Radial Basis Function (RBF)

$$K(x_i, x_j) = \exp\left(-\frac{1}{2}\left(\frac{\|x_i - x_j\|}{\sigma}\right)^2\right) \quad (2.22)$$

Kernel RBF bagus digunakan untuk data yang terpisah secara linier ataupun non-linier. Kernel ini juga mampu mengatasi data dengan *multi class*. Waktu komputasi yang digunakan relatif lebih cepat dan sering menghasilkan akurasi yang bagus. Kernel ini adalah kernel yang paling sering digunakan.

d. Sigmoid

$$K(x_i, x_j) = tanh(\gamma x_i . x_j + c) \quad (2.23)$$

Kenel sigmoid sering digunakan pada data kelas yang terpisah secara non linier, tetapi kadang terjadi *misclassified* dan waktu yang dikonsumsi lebih besar daripada kernel RBF.

2.7 *Cross Validation*

Dalam melakukan klasifikasi, data harus dibagi terlebih dahulu menjadi dua buah subset, yaitu data latih dan data uji. Pada penelitian ini tidak melakukan pembagian data seperti itu, tetapi menggunakan *cross validation*. *Cross validation* adalah prosedur standar untuk mengukur performa dari suatu *classifier* (Wong and Yang, 2017). Kelebihan dari *cross validation* adalah untuk menghindari *overfitting* dan memaksimalkan generalisasi dari data (Lameiro and Schreier, 2016). Cara kerja dari *cross validation* adalah dengan memecah data berdasarkan nilai *k fold*. Data dipecah menjadi data latih sebanyak (*k*-1) *fold* dan sisanya merupakan data uji sebanyak 1*fold*. Pada Gambar 4.4 merupakan contoh ilustrasi pembagian data dari *cross validation* dengan *k* = 10*fold*. Umumnya nilai yang digunakan adalah *k* dengan 10*fold*.

Data Uji					Data Latih				
fold 1	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 2	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 3	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 4	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 5	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 6	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 7	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 8	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 9	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10
fold 10	fold 2	fold 3	fold 4	fold 5	fold 6	fold 7	fold 8	fold 9	fold 10

Gambar 2. 4 Ilustrasi *Cross Validation*

Data terbagi menjadi 10 karena *k* = 10*fold*. Data latih sebanyak (*k*-1) *fold* = 9*fold* dan sisanya sebagai data uji. Secara bergantian data akan dilatih dan diuji dengan data yang telah cipecah sbanyak *k*. Proses pelatihan dan pengujian berjalan sebanyak 10 iterasi, sesuai dengan nilai *k fold* yang ditentukan. Hasil klasifikasi seperti akurasi akan dicatat dari tiap iterasi kemudian dilakukan rata-rata. Rata-rata akurasi tersebut yang akan menjadi keluaran akhir dari *cross validation*.

2.8 *Simple Moving Average* (SMA)

Simple moving average (SMA) adalah menghitung pergerakan nilai rata-rata dengan menambahkan nilai terakhir yang masuk. SMA adalah sebuah metode yang sering digunakan dalam analisa *time series*. Dengan menggunakan teknik ini, dapat menghindari *noise* dan lebih menghaluskan pergerakan trend (Lauren, 2014). Cara menghitung SMA dapat dilihat pada rumus 2.24 dibawah.

$$SMA = \sum_{i=n-T}^{n-1} \frac{x_i}{T} \quad (2.24)$$

Keterangan:

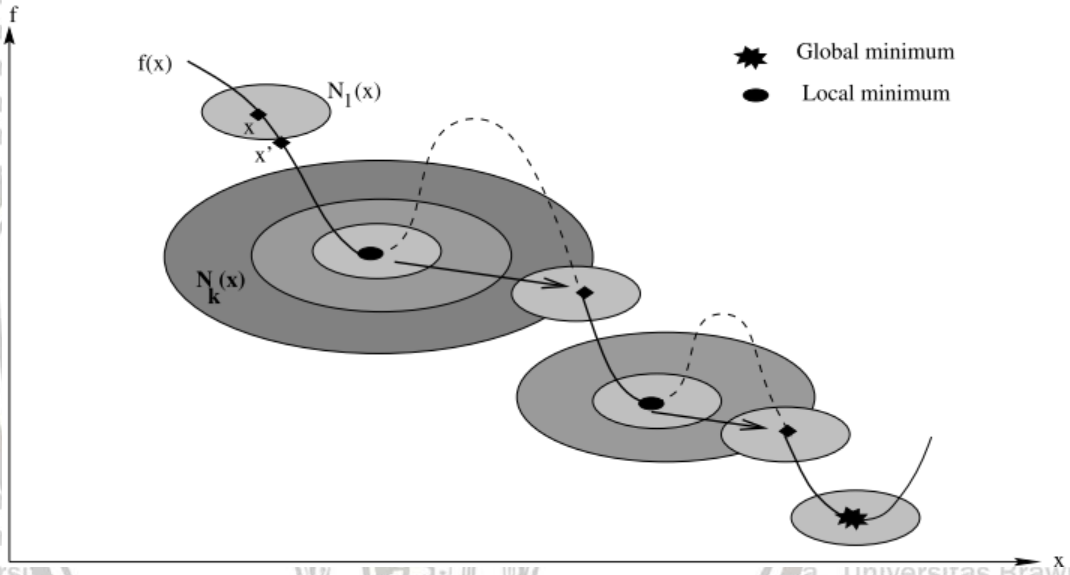
X_i adalah nilai percobaan pada satu periode

T adalah durasi benyaknya percobaan

SMA disini digunakan untuk menentukan berapa banyak percobaan yang harus dilakukan oleh algoritme optimasi nantinya. Setelah didapatkan berapa banyak percobaan, kemudian dilakukan pemilihan parameter algoritma optimasi tersebut untuk mendapatkan nilai parameter yang paling sesuai.

2.8 *Variable Neighborhood Search* (VNS)

Kinerja dari SVM bergantung dari pemilihan parameter yang tepat karena sangat berpengaruh terhadap hasil dari klasifikasinya. Tidak ada aturan khusus dalam menentukan nilai parameter tersebut. Dalam pemilihan parameter ini dapat dilakukan dengan menerapkan salah satu algoritma heuristik yaitu *Variable Neighborhood Search*.



Gambar 2. 5 Ilustrasi pencarian pada VNS (Hansen, Mladenović and Moreno Pérez, 2010)

Variable Neighborhood Search (VNS) adalah algoritme metaheuristik yang sering digunakan untuk menyelesaikan masalah kombinatorial dan optimasi. VNS pertama kali diusulkan oleh (Mladenović and Hansen, 1997). Metode dalam melakukan pencarian lokal dengan melakukan perubahan lokal secara berurutan hingga menemukan solusi globalnya. Solusi awal ditentukan secara acak kemudian meningkatkan solusi sampai menemukan lokal optimum. Setelah ditemukan solusi baru dan iterasi terpenuhi, VNS mengeksplorasi lingkungan pencariannya yang semakin jauh dengan melakukan lompatan untuk mendapatkan solusi yang terbaru. Solusi terbaik yang didapatkan akan disimpan kemudian digunakan kembali untuk mencari solusi tetangga terbaru yang menjajikan. Sebagai ilustrasi sistematik pencarian dari VNS dapat dilihat pada Gambar 2.5. Secara umum pseudocode VNS dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Pseudocode VNS

masukan	x, l_{max}, k_{max}
1	$k = 1$
2	WHILE $k < k_{max}$ do
3	$x' \leftarrow$ *Shake* (x, k)
4	$x'' \leftarrow$ *LocalSearch* (x', l_{max})
5	$x, k \leftarrow$ *NeighborhoodChange* (x, x'', k)
6	END WHILE

Variabel k adalah variabel yang bernilai jumlah tetangga yang akan digunakan oleh VNS. Nilai k ditentukan sebagai inisialisasi awal dan bernilai $k = 1, \dots, k_{max}$ yang akan digunakan dalam pencarian untuk menemukan solusi x. Nilai l merupakan jumlah iterasi yang digunakan pada pencarian lokal dengan $l = 1, \dots, l_{max}$.

Alur algoritme mula-mula set $k = 1$, kemudian iterasi berjalan hingga $k = k_{max}$. Solusi dinotasikan dengan x, x' bernilai solusi sementara dan x'' bernilai solusi lokal optimal. Apabila selama pencarian didapatkan solusi baru yang lebih baik dari sebelumnya maka $x = x''$. Apabila selama pencarian tidak menemukan solusi yang lebih bagus maka akan dilakukan *shaking* dengan $k = k + 1$.

2.8.1 *Shaking*

Tabel 2. 4 Pseudocode algoritme *shaking*

masukan	x, k
1	$x' \leftarrow x^w$
2	return x'

Shaking dilakukan untuk menghasilkan solusi secara acak pada satu ruang pencarian. *Shaking* menghasilkan solusi x' secara acak dari tetangga ke k dari x. Sebagai contoh $x' \in \mathcal{N}_k(x)$. Diasumsikan nilai dari $\mathcal{N}_k(x)$ adalah $\{x^1, \dots, x^{|\mathcal{N}_k(x)|}\}$. Tabel 2.4 menunjukkan *pseudocode* dari tahap *shaking*. Solusi dari tahap *shaking* akan digunakan pada tahap pencarian lokal.

2.8.2 *Local Search*

Tabel 2. 5 Pseudocode algoritme *local search*

masukan	x, l_{max}
1	$l = 0, x' \leftarrow x$
2	WHILE $l < l_{max}$ do
3	if $f(x^i) < f(x)$ then
4	$x \leftarrow x^i$
5	end
6	END WHILE
7	$f(x') < f(x)$ then
8	return x'

Local search atau pencarian lokal yaitu proses mencari solusi dalam satu ruang pencarian. Pada VNS pencarian lokal dilakukan sebanyak iterasi yang telah ditentukan. Pada Tabel 2.5 adalah *pseudocode local search* pada VNS secara umum. Solusi yang menjadi masukan pada *local search* adalah hasil dari tahap *shaking*. *Dari local search* ini akan menghasilkan solusi yang baru, yang nantinya akan digunakan pada tahap *change neighborhood*.

2.8.2 *Change Neighborhood*

Tabel 2. 6 Pseudocode algoritme *change neighborhood*

masukan	x, x', k
1	if $f(x') < f(x)$ then
2	$x \leftarrow x'$
3	$k \leftarrow 1$
4	else
5	k $\leftarrow k + 1$

Change Neighborhood dilakukan untuk eksplorasi ruang pencarian dengan membandingkan solusi x dengan x' yang didapatkan dari pencarian di ketetanggaan. Pergantian ini dilakukan apabila saat pencarian lokal ditemukan solusi baru atau selama iterasi pencarian lokal telah terpenuhi tapi tidak mendapatkan solusi baru. *Pseudocode* dari *change neighborhood* dapat dilihat pada Tabel 2.6 diatas. Apabila dari pencarian lokal didapatkan solusi yang baru, maka nilai *k* akan berubah menjadi menjadi 1, hal ini dilakukan untuk memperluas ruang pencarian karena kemungkinan masih adanya solusi yang lebih baik lagi. Apabila tidak ada solusi yang lebih baru, maka nilai *k* akan terus bertambah hingga kondisi berhenti terpenuhi.

2.9 Ukuran Evaluasi

Evaluasi suatu algoritme klasifikasi umumnya hanya berfokus pada nilai akurasi. Untuk mengetahui performa klasifikasi yang lebih spesifik dapat dengan melihat nilai *precision*, *recall*, F *measure* dan Matthews correlation coefficient (MCC). Penggunaan ukuran evaluasi ini berdasarkan penelitian sebelumnya yang meneliti tentang *sequence classification* (Iqbal et al., 2013; Phan et al., 2017; Ma et al., 2012). Metode evaluasi ini digunakan untuk validasi performa algoritme klasifikasi yang digunakan.

2.9.1 Akurasi

Akurasi digunakan untuk mengukur seberapa akurat hasil yang diprediksi dengan kelas aktualnya. Persamaan 2.24 adalah perhitungna dalam menghitung akurasi.

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.24)$$

2.9.2 Presisi

Presisi digunakan untuk mengetahui seberapa dekat informasi yang diberikan oleh *classifier* dengan rumus pada Persamaan 2.25.

$$Presisi = \frac{TP}{TP+FP} \quad (2.25)$$

2.9.3 *Recall*

Evaluasi *recall* (sensitifitas) untuk mengetahui seberapa banyak TP yang diprediksi oleh sistem. Pada Persamaan 2.26 adalah rumus untuk menghitung *recall*.

$$Recall/\ sensitifitas = \frac{TP}{TP+FN} \quad (2.26)$$

2.9.4 F Measure

F *measure* merupakan rata-rata harmonik dari precision dan *recall*. Pada Persamaan 2.27 adalah rumus untuk menghitung F *measure*.

$$F\ Measure = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \quad (2.27)$$

2.9.5 *Matthews Correlation Coefficient* (MCC)

MCC digunakan untuk mengukur kualitas dari klasifikasi, dengan menghitung rata-rata harmonic dari *precision* dan *recall*. Dalam bioinformatika, MCC lebih sering digunakan. MCC lebih informatif karena menggunakan semua kondisi dari *confusion matrix*. Dalam menghitungnya dapat dilihat pada Persamaan 2.28 dibawah.

$$MCC = \frac{TPxTN-FPxFN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (2.28)$$

Dengan ketentuan berdasarkan *confusion matrix* pada Tabel 2.7 dibawah:

Tabel 2. 7 *Confusion Matrix*

		Actual class	
		Positive	*Negative*
Predicted class	*Positive*	*True Positive (TP)*	*False Positive (FP)*
	Negative	*False Negative (FN)*	*True Negative (TN)*

Confusion matrix adalah matrik yang digenerasi oleh *classifier* untuk mengetahui performanya. Matrik ini menunjukkan relasi antara prediksi yang benar dan salah. TP merepresentasikan keseluruhan prediksi yang benar. FP merepresentasikan banyaknya yang diprediksi sebagai positif tetapi pada nilai aktualnya adalah negatif. FN merepresentasikan banyaknya yang diprediksi sebagai negatif tetapi nilai aktualnya positif. TN merepresentasikan banyaknya yang diprediksi sebagai benar-benar negatif.

BAB 3 METODOLOGI

3.1 Metode Penelitian

Metode penelitian dilakukan secara berurutan dan sistematis. Berikut skema langkah-langkah yang akan dilakukan pada penelitian ini terpapar pada Gambar 3.1 dibawah.

Gambar 3. 1 Diagram alur penelitian

Berikut adalah penjelasan tahapan yang dilakukan dalam melakukan penelitian ini:

1. Melakukan studi pustaka terkait objek dan metode-metode yang digunakan.
2. Mengumpulkan data-data sebagai objek penelitian.
3. Melakukan perencangan metode termasuk tahapan-tahapan secara terstruktur dari penelitian dan membuat perancangan dari pengujian metode yang diusulkan.
4. Implementasi metode yang telah diusulkan kedalam program sesuai dengan rancangan sebelumnya.
5. Melakukan percobaan dan pengujian metode serta melakukan perbandingan dari metode. Kemudian dilakukan pembahasan dari tiap-tiap hasil dari uji coba.
6. Mendapatkan dan mengevaluasi hasil dari uji coba. Kemudian membuat kesimpulan yang menjawab rumusan masalah dari uji coba yang dilakukan.

3.2 Studi Pustaka

Studi pustaka diambil dari beberapa sumber, diantaranya yaitu buku, publikasi jurnal ilmiah, dan referensi terkait untuk menunjang penelitian ini. Studi pustaka mencakup tentang Tuberkulosis, DNA, *preprocessing* data seperti ekstraksi fitur dan reduksi dimensi, *Support Vector Machine*, *Variable Neighborhood Search*, serta metode lain yang masih berhubungan dengan pada penelitian ini.

3.3 Pengumpulan Data

Data yang digunakan adalah data yang berhasil dikumpulkan dari situs laman *National Center for Biotechnology Information* (NCBI), yaitu : https://www.ncbi.nlm.nih.gov/nuccore/. Data yang dikumpulkan adalah data DNA dari *Mycobacterium Tuberculosis* serta jenis-jenisnya. Data berisi deretan basa nitrogen penyusun DNA, tidak termasuk data protein. Data yang dikumpulkan adalah data *complete genome* dengan format FASTA.

Tabel 3. 1 *Lineage* dari *Mycobacterium tuberculosis*

No.	Kelas
1	Indo-Oceanic
2	East Asian
3	Central Asia
4	Euro-America
5	West African 1
6	West African 2

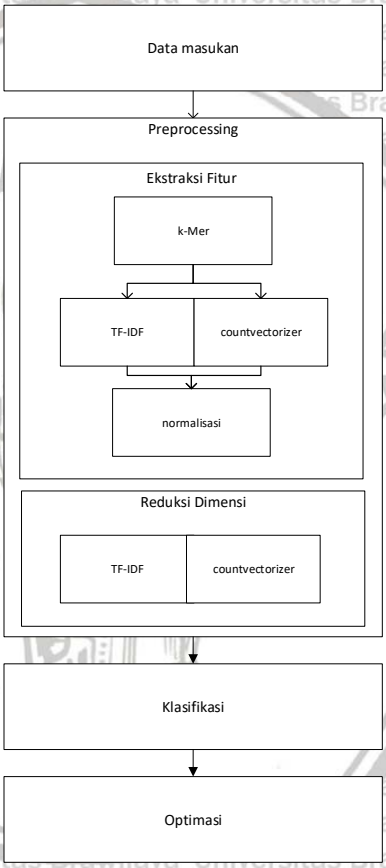
Masing-masing dari data yang dikumpulkan memiliki kelasnya sendiri berdasarkan *lineage* (garis keturunan). Ada 6 *lineage* utama (Yimer et al., 2015) yang digunakan sebagai kelasnya seperti yang dipaparkan pada Tabel 3.1. Pemilihan kelas tersebut karena tiap *lineage* memiliki kemampuan resistensi terhadap obat yang berbeda. Seperti *lineage* dari Beijing yang memiliki resistensi terhadap obat paling tinggi sehingga membutuhkan penanganan yang berbeda. Sedangkan negara India serta negara Afrika bagian timur yang termasuk ke dalam *lineage Central Asia* memiliki resistensi terhadap obat paling rendah (Yimer et al., 2015). *Drug-resistan TB* (DR-TB) menjadi ancaman utama karena pada tahun 2013 sebanyak 3.5% pasien pengidap Tuberkulosis memiliki pertahanan terhadap obat yang diberikan (Évora, L.H.R.A., Seixas, 2015).

Data yang telah dikumpulkan sebanyak 233, dengan detail jumlah dan kelas seperti yang ditunjukkan pada Tabel 3.2. Banyaknya kelas yang digunakan adalah 6 kelas. Data yang telah dikumpulkan memiliki jumlah paling banyak pada kelas *Euro-America* yang berjumlah 130 dan yang paling sedikit pada kelas *West African* 2 yang berjumlah 2. Hal ini menunjukkan bahwa varian virus TB dari kelas *Euro-America* sangat beragam.

Tabel 3. 2 Jumlah data yang dikumpulkan

Kelas	Jumlah
Indo-Oceanic	7
East Asian	74
Central Asia	12
Euro-America	130
West African 1	7
West African 2	3

3.4 Pemodelan Metode



Gambar 3. 2 Pemodelan metode

Secara umum mekanisme dari metode yang akan dikerjakan tertera pada Gambar 3.2. Mula-mula dilakukan pengumpulan data sebagai obyek dari penelitian ini. Data ini akan digunakan sebagai masukan yang kemudian diolah pada tahap berikutnya.

Data yang telah terkumpul sebelumnya akan dilakukan proses *preprocessing*. Awalnya data akan di ekstraksi fitur menggunakan algoritme k-Mer. Dengan menggunakan nilai k yang berbeda, akan didapatkan dataset baru

dengan fitur yang sudah terekstrak dan dataset yang dihasilkan sebanyak berapa nilai k yang digunakan. Data baru berisi substring dari DNA dengan format teks dan panjang masing-masing data yang berbeda. Karena data berupa teks maka perlu dijadikan matriks angka (Tripathy, Agrawal and Rath, 2016). Berdasarkan penelitian sebelumnya, ada dua metode yang dapat digunakan yaitu TF-IDF dan *countvectorizer* (Parmar, 2018; Jimenez-marquez et al., 2019; Hatamian, Serna and Rannenberg, 2019). Penelitian ini akan menggunakan dua metode tersebut sebagai perbandingan untuk mengetahui metode yang mana yang baik digunakan pada kasus ini. Kemudian hasil dari ekstraksi fitur didapatkan data terbaru dan dilakukan normalisasi data supaya data berbentuk normal tidak pada rentang yang sangat berbeda. Dataset setelah dinormalisasi akan dilakukan reduksi fitur dengan menggunakan PCA. Varian data yang digunakan dilihat dari banyaknya data yang dicover. Sebagai perbandingan, akan digunakan LDA karena mampu dalam melakukan reduksi fitur (Ilias et al., 2016). Setelah itu akan didapatkan data baru yang telah dilakukan *preprocessing*.

Data yang telah siap akan digunakan sebagai masukan pada SVM. Klasifikasi SVM dengan menerapkan *cross valdiation* supaya tidak terjadi *overfitting* dan pelatihan serta pengujian data dilakukan secara adil. SVM melakukan klasifikasi dengan parameter *default* dan memberikan hasil berupa evaluasi. Evaluasi tersebut digunakan untuk mengetahui nilai k dan metode *preprocessing* manakah yang paling baik itu yang akan digunakan ke tahap optimasi berikutnya.

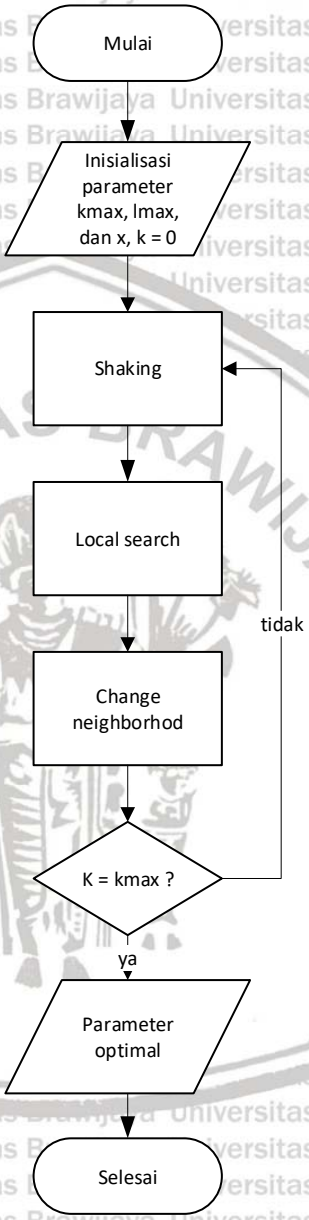
Pada tahap optimasi, inisialisasi parameter SVM dilakukan oleh VNS kemudian di klasifikasikan. Optimasi dilakukan dengan menggunakan VNS dasar, dan VNS yang telah dimodifikasi. Dari hasil optimasi akan didapatkan nilai kebugaran sebagai acuan bahwa solusi tersebut baik. Solusi berupa nilai parameter SVM. Optimasi berlangsung hingga kondisi berhenti dari VNS terpenuhi. Setelah itu dilakukan pembahasan perbandingan VNS dasar dengan VNS yang dimodifikasi.

Keluaran akhir dari seluruh tahap adalah pemilihan nilai k dari k-Mer yang terbaik, metode ekstraksi fitur yang sesuai, metode reduksi dimensi paling bagus, parameter SVM yang paling optimal dan seberapa bagus performa modifikasi VNS yang dilakukan.

3.5 Pemodelan Optimasi

Skema optimasi yang dilakukan pada penelitian ini dapat dilihat pada Gambar 3.4 dibawah. Mula-mula dilakukan inisialisasi parameter dari VNS yaitu *kmax* (banyaknya jumlah ketetanggaan) dan *lmax* (banyaknya iterasi pada pencarian lokal). Serta *x* solusi sebagai inisiasi awal yang berisi parameter default *C* dan *gamma* dari SVM. Nilai variabel *k* = 0, ini akan terus bertambah hingga

mencapai *kmax*. Pada tiap *shaking* dan *local search* dilakukan pencarian solusi kemudian dilakukan klasifikasi hingga didapatkan solusi yang baru berdasarkan nilai akurasinya sebagai acuan. Optimasi akan terus berlangsung hingga kondisi berhenti terpenuhi. Kondisi terpenuhi disini saat nilai *k* mencapai *kmax*. Diakhir optimasi akan didapatkan nilai parameter SVM yang paling optimal.



Gambar 3. 3 Skema optimasi dengan VNS

3.6 Pemodelan Modifikasi VNS

Modifikasi yang dilakukan adalah dengan mengadaptasi variasi dari VNS, yaitu *reduced variable neighborhood search* (RVNS). Skema RVNS adalah dengan menghilangkan *local search* dari VNS sehingga hanya menyisakan proses *shaking* dan *change neighborhood* (Hansen et al., 2017). Kelebihan dari RVNS adalah dapat

mengurangi waktu komputasi dari VNS dasar. Kekurangannya adalah mengorbankan akurasi dari penncarian solusinya.

Tabel 3. 3 Pseudocode reduced variable neighborhood search (RVNS)

masukan	$x,\ k_{max}$
1	$k = 1$
2	WHILE $k < k_{max}$ do
3	$x' \leftarrow$ *Shake* (x, k)
4	$x, k \leftarrow$ *NeighborhoodChange* (x, x', k)
5	END WHILE

Pseudocode RVNS dapat dilihat pada Tabel 3.3. Untuk tetap menjaga akurasi dengan waktu yang lebih cepat, oleh karena itu diusulkan *nested* RVNS. *Nested* RVNS juga merupakan adaptasi dari varian VNS yaitu *nested* VNS tetapi dengan beberapa perubahan didalamnya. Pada Tabel 3.4 adalah pseudocode dari rencana modifikasi VNS yang akan dilakukan pada penelitian ini.

Tabel 3. 4 Pseudocode *nested* VNS

masukan	$x,\ k_{max}$
1	WHILE $k < k_{max}$ do
2	$x' \leftarrow$ *RVNS* (x, k_{max})
3	$x, k \leftarrow$ *NeighborhoodChange* (x, x', k)
4	END WHILE

Pada modifikasi yang diusulkan, secara garis umum strukturnya mirip dengan RVNS akan tetapi perbedaannya adalah tahap *shaking* digantikan dengan RVNS itu sendiri. Jadi RVNS disini menggantikan inisiasi solusi dari *shaking* pencarian solusi dari *local* search dari VNS dasar. Dengan begitu RVNS bisa berjalan cepat tetapi bisa memperluas ruang pencarian solusinya dan memungkinkan untuk ditemukannya solusi yang bagus sehingga tidak mengorbankan akurasinya.

Pada modifikasi ini, hanya diperlukan satu parameter saja, yaitu *kmax*. Nilai *kmax* dari RVNS mengikuti besarnya nilai *kmax* dari *nested RVNS*. Kondisi berhenti pada metode ini ditentukan oleh nilai *kmax*.

3.7 Pengujian

Ada beberapa skenario pengujian, yaitu pengujian k-Mer, pengujian reduksi dimensi, pengujian optimasi SVM dengan VNS dan pengujian modifikasi VNS yang akan dijelaskan lebih detil pada subbab dibawah.

3.7.1 Pengujian k-Mer

Pada pengujian ini dilakukan untuk mendapatkan nilai k berapakah yang terbaik dari k-Mer. Rentang nilai k yang akan diuji cobak bernilai $k \geq 2 \leq 9$. Kemudian dilakukan pembobotan substring dan menyeragamkan panjang data

hasil k-Mer menggunakan TF-IDF dan *countvectorizer*. Kemudian dilakukan normalisasi data menggunakan normalisasi *Min-Max* untuk menormalkan data. Terakhir, untuk mengetahui seberapa baiknya metode yang digunakan akan dilihat hasil evaluasi dari klasifikasinya.

3.7.2 Pengujian Reduksi Dimensi

Pada pengujian ini dilakukan dengan menerapkan PCA terhadap data yang telah diekstraksi fiturnya untuk dilakukan reduksi dimensi. Sebagai pembanding dalam reduksi fitur, akan menggunakan LDA. Mula-mula akan digunakan semua varian data untuk dilihat hasilnya manakah yang terbaik. Hasil yang didapat akan dibandingkan dengan data sebelum dilakukan reduksi dimensi. Sebagai hasil akhir dari pengujian ini akan didapatkan metode yang paling baik.

3.7.3 Pengujian Parameter SVM

Parameter yang diujikan adalah *C* dan *gamma* dari SVM. Sedangkan kernel yang digunakan adalah kernel *radial basis function* (RBF). Parameter *C* merujuk ke nilai pinalty dimana mempengaruhi *trade-off* dan kompleksitas perhitungan, sedangkan *gamma* adalah parameter dari kernel RBF (Jin et al., 2014). Dengan kernel RBF dapat mengatasi data yang kelasnya terpisah secara tidak linier, berbeda dengan kernel linier yang hanya dapat digunakan untuk data yang terpisah secara linier. Bedanya, dengan kernel linier hanya parameter *C* yang digunakan, sedangkan dengan kernel RBF ada parameter *C* dan *gamma* yang digunakan. Selain itu dengan kernel RBF memiliki *hyperparameter* yang lebih sedikit daripada kernel *polynomial*, sehingga membuat kernel RBF memiliki komputasi yang tidak terlalu rumit. Nilai kernel *polynomial* bisa bernilai tidak terbatas apabila nilai *degree* yang besar. Sedangkan kernel *sigmoid* melakukan perkalian hasil dari dua buah vektor yang membuatnya tidak valid(Apostolidis-afentoulis and Box, 2015).

Pengujian parameter SVM dengan merubah nilai parameter *C* dan *gamma* dengan menaik turunkan nilainya. Tujuannya adalah untuuk mengetahui dampak perubahan parameter terhadap hasil klasifikasinya. Selain untuk mendapatkan rentang nilai untuk dilakukan tahap optimasi berikutnya.

3.7.4 Pengujian Optimasi SVM dengan VNS

Pada pengujian ini dilakukan dengan menggunakan VNS sebagai metode optimasi yang diusulkan untuk mendapatkan parameter terbaik. Parameter yang dioptimasi adalah parameter *C* dan *gamma* dari SVM. Skenario pengujiannya adalah dengan menentukan nilai *kmax* dan iterasi yang merupakan parameter dari VNS. Cara pengujiannya adalah dengan melakukan uji *kmax* dengan nilai iterasi yang sudah ditentukan. Kemudian uji iterasi dengan nilai *kmax* saat menemukan solusi yg optimal yang telah didapatkan sebelumnya. Untuk mengetahui seberapa bagus solusi dengan melihat nilai akurasi hasil dari klasifikasi. Pengujian ini

dilakukan sebanyak hasil dari percobaan *Simple Moving Average* (SMA) yang nantinya akan dilakukan.

3.7.5 Pengujian modifikasi VNS

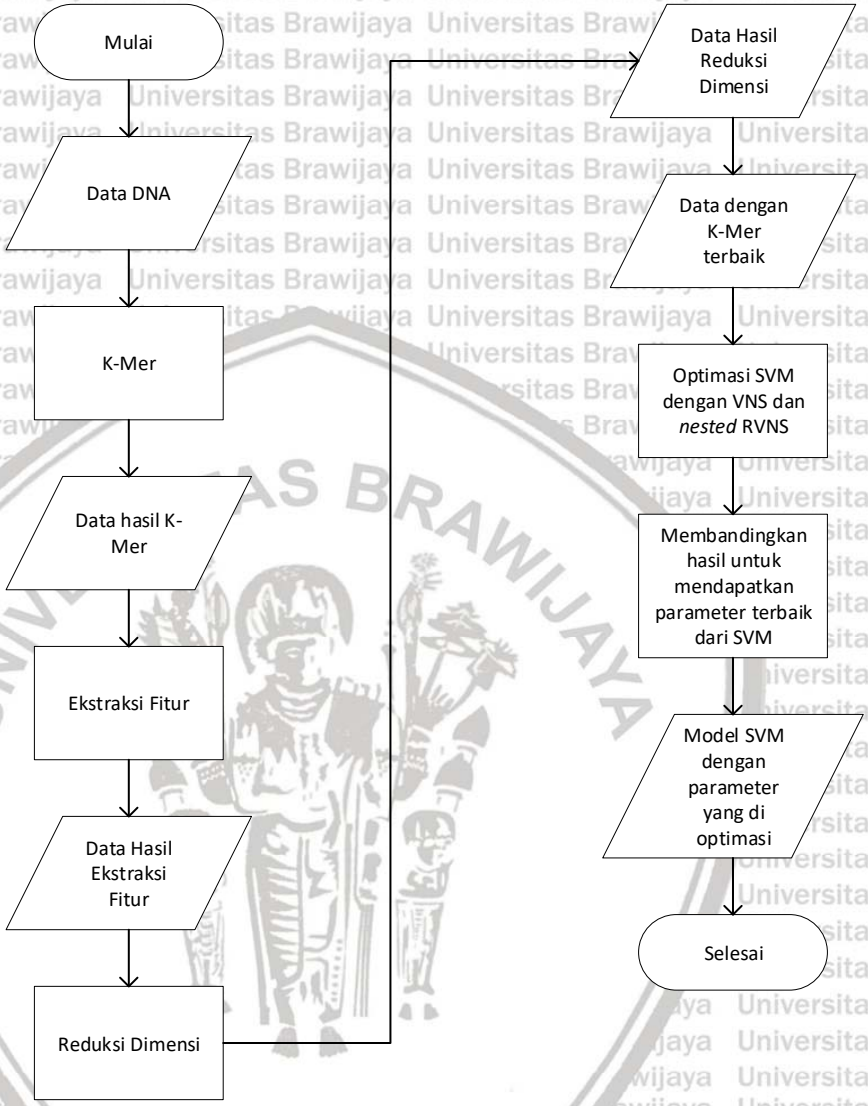
Pada pengujian ini dilakukan dengan mengujikan parameter dari *nested* RVNS, yaitu parameter *kmax*. Hanya terdapat satu buah parameter yang diujikan karena VNS yang dimodifikasi hanya memerlukan satu parameter saja. Sama seperti pengujian VNS dasar, *nested* RVNS akan dijalankan sebanyak hasil dari uji SMA. Kemudian dicatat hasilnya berupa parameter, rata-rata akurasi yang didapatkan serta waktu komputasi yang dibutuhkan untuk mengetahui perbandingan metode yang lebih baik.

3.8 Evaluasi

Ada beberapa hal yang akan menjadi evaluasi performa SVM, yaitu nilai akurasi, *precision*, *recall*, *F*-measure dan MCC serta waktu komputasi yang digunakan. Nilai tersebut akan digunakan untuk mengevaluasi hasil pada tahap ekstraksi fitur dan preprocessing dalam menentukan metode yang paling tepat. Untuk evaluasi optimasi dengan menggunakan perhitungan akurasi dari klasifikasi dan waktu komputasi. Evaluasi ini untuk digunakan mengetahui performa *classifier* yang digunakan.

BAB 4 PERANCANGAN

4.1 Perancangan Metode

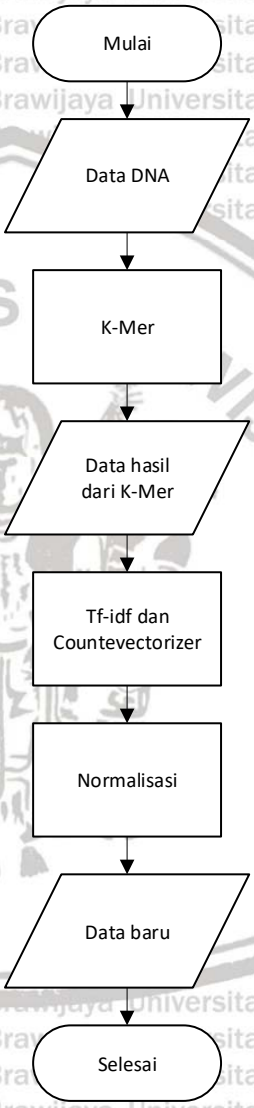


Gambar 4. 1 Gambaran umum diagram alur perancangan metode

Bab ini akan menjelaskan perancangan dan alur proses dari metode yang akan digunakan pada penelitian ini. Secara umum dapat dilihat pada Gambar 4.1 diatas, perancangan terdiri dari masukan berupa data urutan DNA *Mycobacterium Tuberculosis*, kemudian dilakukan ekstraksi data dengan menggunakan K-Mer. Data baru hasil dari K-Mer didapatkan dan langsung dilakukan *preprocessing* ekstraksi fitur menggunakan TF-IDF dan *countvectorizer* untuk merubah data kedalam bentuk numerik. Data hasil ekstraksi fitur memiliki dimensi yang tinggi, maka dari itu dilakukan reduksi dimensi dengan menggunakan PCA dan LDA sebagai pembandingnya. Data-data dari tahap ekstraksi fitur dan reduksi dimensi, semuanya akan diklasifikasi untuk mendapatkan hasil evaluasi yang kemudan akan

dilakukan perbandingan hasilnya. Tujuan dilakukannya ini untuk mengetahui nilai *k* berapakah dari K-Mer yang terbaik yang dapat memberikan evaluasi maksimalnya. Selanjutnya dilakukan optimasi parameter menggunakan VNS dan *nested* RVNS sebagai metode optimasi yang diusulkan juga sebagai pembanding dari VNS dasar. Hasil akhirnya adalah model SVM yang optimal berdasarkan parameter hasil dari optimasi.

4.1.1 *Preprocessing* Data



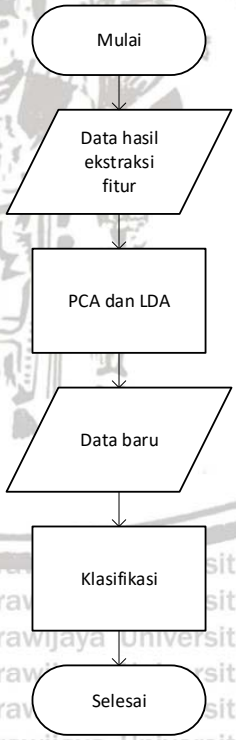
Gambar 4. 2 Diagram alur ekstrarksi fitur data

Dalam mendapatkan fitur data DNA *Mycobacterium Tuberculosis* yang telah dikumpulkan sebelumnya, akan dilakukan ekstraksi fiturnya terlebih dahulu. Sesuai dengan Gambar 4.2, mula-mula data yang masuk akan di proses dengan menggunakan k-Mer untuk mendapatkan substring dari DNA. Data yang berupa substring harus diubah menjadi matriks angka sebelum dapat dilakukan proses klasifikasi. Untuk mengubah substring menjadi data bernilai numerik dengan

menerapkan TF-IDF dan *countvectorizer*. Dengan dua metode tersebut, dapat menyeragamkan panjang data yang sebelumnya memiliki panjang yang tidak seragam.

Data baru hasil ekstraksi fitur telah didapatkan dan data mengandung rentang nilai yang sangat besar. Maka selanjutnya dilakukan normalisasi untuk menormalkan nilai fitur data. Metode normalisasi yang digunakan adalah *Min-Max normalization* dengan rentang 0 hingga 1. Normalisasi ini dilakukan karena data memiliki rentang nilai yang tinggi.

Selanjutnya setelah data dinormalisasi, menerapkan reduksi fitur karena data keluaran dari ekstraksi fitur memiliki dimensi yang besar. Data dengan nilai *k* dari k-Mer yang tinggi, menghasilkan dimensi yang tinggi juga. Metode reduksi dimensi yang digunakan adalah PCA dan LDA, kemudian membandingkan hasilnya untuk mengetahui metode yang memiliki performa terbaik pada penelitian ini. Gambar 4.3 merupakan alur reduksi dimensi yang akan dilakukan. Keluaran dari reduksi dimensi ini adalah data baru dengan data yang berdimensi lebih rendah dari sebelumnya. Untuk mengetahui metode yang terbaik, adalah dengan melihat hasil klasifikasinya, pada penelitian ini klasifikasi dengan menggunakan SVM.



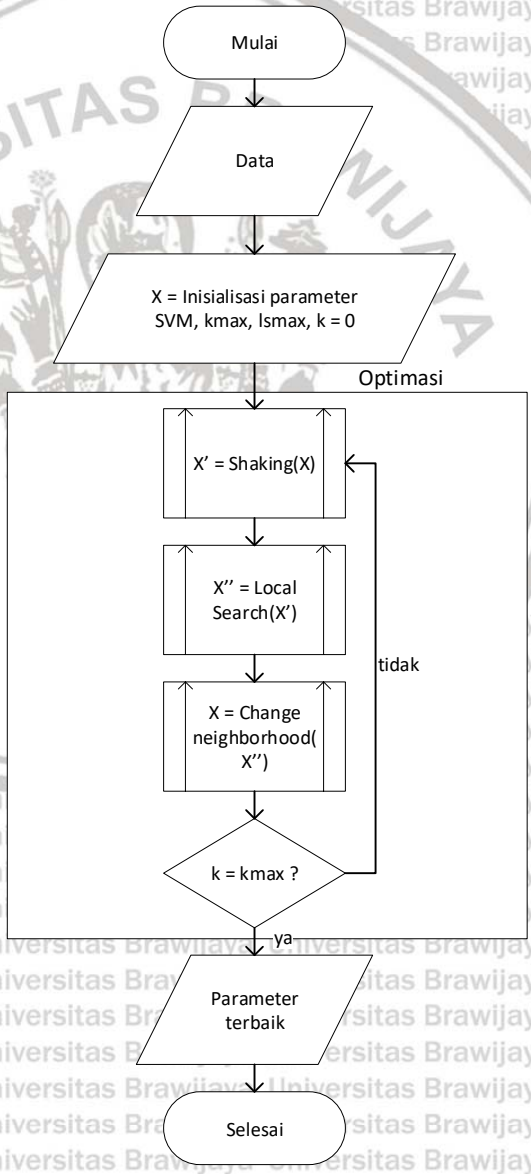
Gambar 4. 3 Diagram alur reduksi dimensi data

Hasil akhir dari tahap *preprocessing* yang didapat adalah data dengan nilai k dari k-Mer, metode ekstraksi fitur, dan metode reduksi dimensi yang terbaik. Data tersebut yang akan digunakan selanjutnya kedalam tahap optimasi dengan VNS untuk mendapatkan parameter SVM paling optimal.

4.1.2 Klasifikasi Data

Tahap klasifikasi adalah klasifikasi data yang telah diperoleh dari tahap sebelumnya. Klasifikasi yang digunakan adalah SVM dengan nilai parameter *C* = 1 dan *gamma* = 1. Sekaligus pada tahap ini dilakukan pengubahan terhadap parameter SVM secara manual dengan menaik dan turunkan nilainya secara bertahap. Tujuannya adalah untuk mengetahui dan menentukan rentang batas nilai yang digunakan pada tahap optimasi berikutnya. Untuk mendapatkan hasil evaluasi klasifikasi yang maksimal, pada penelitian ini menerapkan *cross validation* dengan nilai *k fold* adalah 10. Penerapan *cross validation* disini untuk mencegah *overfitting* dari model klasifikasi yang dibuat.

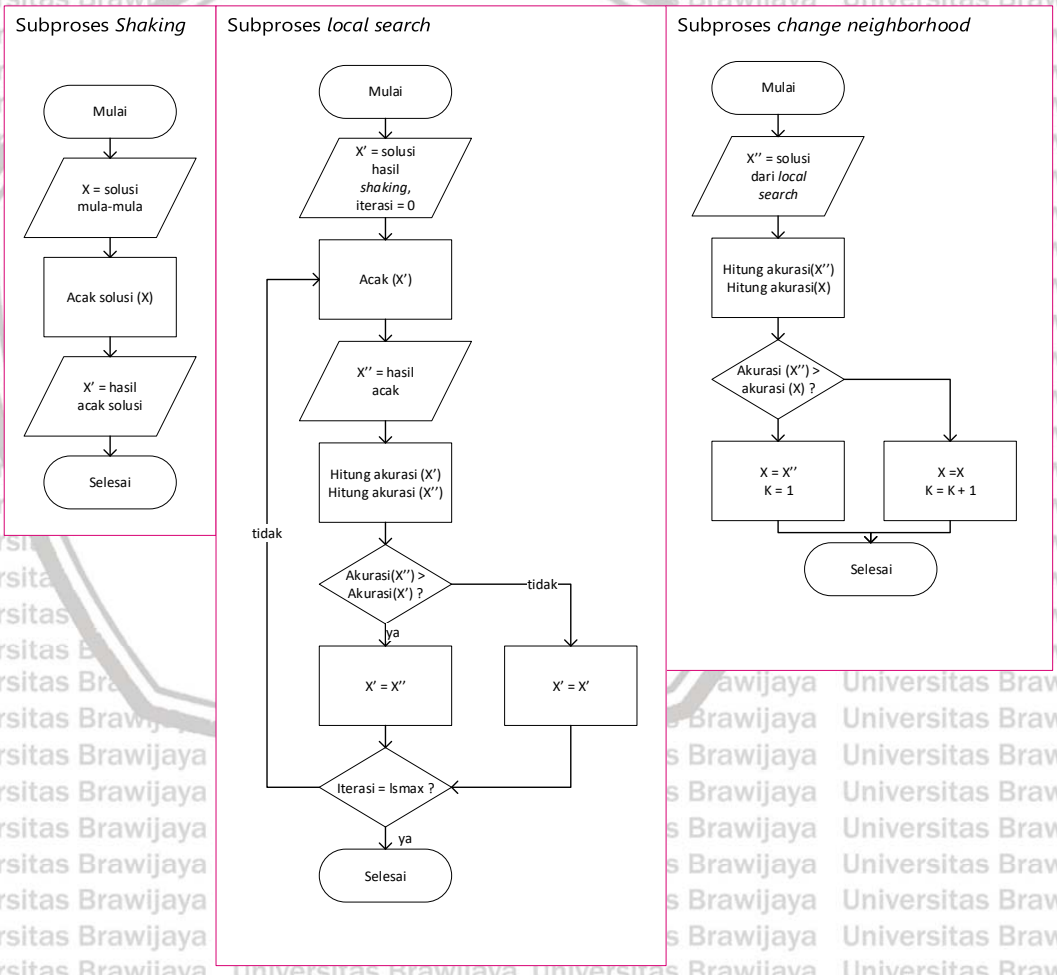
4.1.3 Perancangan Optimasi SVM Dengan VNS



Gambar 4. 4 Diagram alur perancangan optimasi VNS

Perancangan ini merupakan rencana optimasi parameter SVM dengan menerapkan VNS. VNS disini melakukan pencarian parameter terbaik untuk menghasilkan model yang optimal. Sebagai indikator bahwa parameter tersebut baik adalah dengan melihat dari akurasi yang SVM hasilkan. Semakin besar nilai akurasi, semakin bagus model yang didapatkan.

Alur proses VNS berdasarkan Gambar 4.4, masukan berupa data yang akan diklasifikasi dan parameter-parameter VNS. Masukan tersebut adalah inisialisasi solusi yang berupa parameter SVM dan akurasi dengan parameter *default*, *kmax* (jumlah tetangga), jumlah iterasi maksimal *local search* dan k = 0. Terdapat 3 tahap utama pada VNS, yaitu *shaking*, *local search*, dan *change neighborhood*. Subproses dari 3 tahap tersebut dapat dilihat pada Gambar 4.5. VNS akan berlangsung selama kondisi berhenti belum terpenuhi. Kondisi berhenti terpenuhi ketika nilai *k* (ketetanggaan) mencapai *kmax*.



Gambar 4. 5 Diagram alur subproses *shaking*, *local search*, dan *change neighborhood*

Pada tahap optimasi berlangsung, mula-mula akan didapat solusi baru *x'*, yaitu solusi hasil tahap *shaking*. Tahap *shaking* ini mekanismenya dengan

mengacak nilai parameter berdasarkan batas atas rentang nilai yang telah ditentukan. Batas ini ditentukan dari tahap klasifikasi sebelumya. Alur kerja *shaking* dapat dilihat pada Gambar 4.5 dibawah. Solusi yang diacak adalah solusi yang digunakan sebagai masukan, dan solusi hasil acak ini akan menjadi *x'*.

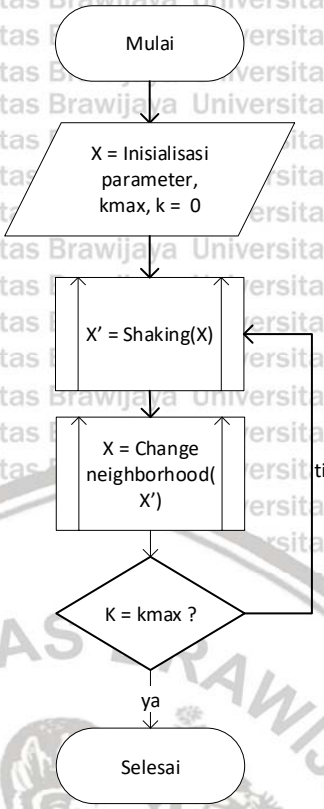
Setelah dilakukan *shaking*, kemudian solusi *x'* yang didapatkan akan digunakan sebagai masukan pada tahap *local search*. Diagram alur dari *local search* dapat dilihat pada Gambar 4.5 dibawah. Dapat diketahui bahwa ada masukan yang diperlukan, yaitu *x'* dan jumlah iterasi. Jumlah iterasi akan digunakan sebagai kondisi berhenti dari *local search*. Mula-mula dilakukan acak solusi lagi lalu didapatkan *x''*, kemudian akan dihitung akurasi dari kedua solusi yang ada. Hitung akurasi dari *x'* dan *x''* dan dibandingkan apakah akurasi *x''* lebih besar dari akurasi *x'*. Apabila lebih besar akurasi dari *x''*, ganti nilai dari *x'* dengan *x''*. apabila tidak, makan solusi *x'* bernilai tetap. Mekanisme ini berjalan hingga iterasi sebagai kondisi berhenti yang telah ditentukan terpenuhi. Keluaran dari tahap *local search* adalah solusi solusi *x''* yang akan digunakan pada tahap *change neighborhood* berikutnya.

Dari proses *local search* sebelumnya telah didapatkan solusi *x''*, solusi ini akan digunakan oleh tahap *change neighborhood*. Sesuai dengan gambar 4.5, masukan dari tahap ini adalah solusi *x''* dan nilai *k* yaitu nilai ketetanggaan. Mula-mula nilai $k = 1$. Pada tahap ini dilakukan perbandingan nilai akurasi dari *x''* dengan akurasi dari solusi *x* mula-mula. Apabila akurasi *x''* lebih besar dari akurasi *x*, maka solusi *x* bernilai *x''* dan nilai $k = 1$. Jika akurasi *x''* tidak lebih dari akurasi *x*, maka solusi *x* bernilai tetap dan nilai $k = k + 1$. Selama VNS berlangsung, nilai *k* akan terus bertambah sampai $k = kmax$ yang merupakan kondisi berhenti dari VNS.

4.1.4 Perancangan *nested* RVNS

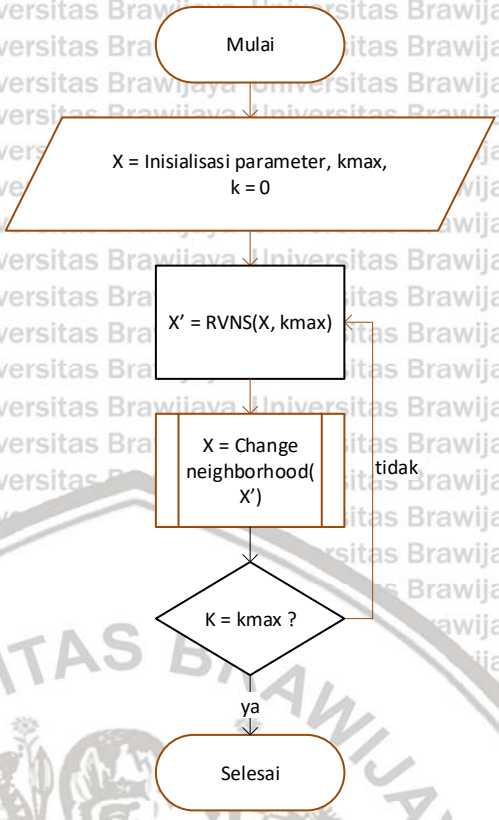
Perancangan *nested* RVNS berdasarkan salah satu variasi dari VNS, yaitu RVNS. Pada RVNS meniadakan tahap *shaking* dan *local search*. Sedangkan pada *nested* RVNS yang diusulkan ini mengganti 2 tahap tersebut dengan RVNS.

Diagram alur RVNS dapat dilihat pada Gambar 4.6 dibawah. Dari gambar tersebut dapat diketahui bahwa tidak ada proses pencarian lokalnya sehingga dapat memangkas waktu komputasinya. Sedangkan parameter yang dibutuhkan hanya *kmax* saja. Tahap *shaking* dan *local search* sama seperti VNS yang digambarkan pada Gambar 4.5. Kekuatan dari VNS adalah pada pencarian lokalnya, sehingga membuat RVNS memiliki kelemahan yaitu hasil optimasinya yang kurang optimal. Dengan usulan *nested* RVNS diharapkan dapat menutupi kekurangan tersebut. Gambar 4.7 adalah diagram alur dari *nested* RVNS.



Gambar 4. 6 Diagram alur RVNS

Secara umum alur tahapan dari *nested* RVNS menyerupai RVNS, yang membedakan pada *shaking* yang digantikan oleh RVNS. RVNS disini mewakili *shaking* dan *local search* dari VNS dasar. Dengan begitu dapat menutupi kekurangan dari RVNS. Parameter yang dimiliki *nested* RVNS adalah *kmax*. Parameter ini digunakan juga oleh RVNS yang berada didalam *nested* RVNS. Jadi menggunakan satu buah parameter untuk keseluruhan alur didalamnya. Nilai *kmax* dari RVNS mengikuti *kmax* dari *nested* RVNS. Pada tahap *change neighborhood* sama seperti VNS dasar yang digambarkan pada Gambar 4.5. Keluaran akhir dari optimasi ini adalah solusi parameter dari SVM yang paling optimal berdasarkan hasil akurasinya.



Gambar 4. 7 Diagram alur *nested* RVNS

4.2 Perancangan Pengujian dan Evaluasi

4.2.1 Uji Coba *Preprocessing*

Nilai *k* pada k-Mer berpengaruh terhadap ekstraksi fitur yang dihasilkan. Semakin besar nilai *k,* semakin banyak jumlah karakter yang didapatkan. Dengan *k* yang besar memberikan jumlah kombinasi karakter yang semakin bervariasi. Dalam pengujian ini, nilai *k* yang akan diuji berada diantara 2 hingga 9. Kemudian dilakukan normalisasi pada data dari masing-masing data per *k* tersebut.

Setelah data DNA menjadi substring-substring, akan dilakukan ekstraksi fitur dengan menggunakan TF-IDF dan *countvectorizer*. Dua metode ini memberikan hasil yang berbeda dan akan diujikan dengan melihat hasil evaluasi dari klasifikasinya menggunakan SVM. Penggunakan ekstraksi fitur ini diberlakukan pada setiap data dengan nilai *k* dari k-Mer yang telah dilakukan sebelumnya.

Panjang data yang mencapai ribuan sehingga perlu dilakukan reduksi dimensi. Tujuan diterapkannya reduksi dimensi supaya dimensi data berkurang dan dapat memangkas waktu komputasi yang lama. Selain itu dengan reduksi dimensi, terkadang dapat meningkatkan akurasi dari klasifikasi terhadap data tersebut. Uji coba reduksi dimensi dengan menggunakan PCA dengan pembanding

yaitu LDA. Tujuan dari uji coba ini untuk mengetahui metode yang paling tepat untuk digunakan dan untuk mengetahui pengaruh dari reduksi dimensi dengan menggunakan PCA ataupun LDA.

Tabel pengujian dapat dilihat pada Tabel 4.1 dibawah. Sebagai ukuran baik tidaknya metode *preprocessing* yang digunakan, dpat dilihat dari evaluasi klasifikasi seperti akurasi, presisi, *recall*, F1 *measure*, dan MCC. Waktu komputasi dicatat untuk mengetahui perbandingan waktu seberapa cepat waktu klasifikasi yang dibutuhkan saat menggunakan PCA ataupun LDA.

Tabel 4. 1 Perancangan pengujian *preprocessing*

Evaluasi	2				...		9			
	TF-IDF		Countvectorizer				TF-IDF		Countvectorizer	
	PCA	LDA	PCA	LDA			PCA	LDA	PCA	LDA
Akurasi										
Presisi										
recall										
F1 *score*										
MCC										
Waktu										

4.2.2 Uji Coba Parameter SVM

Pengujian parameter SVM dengan merubah nilai parameter *C*, dan *gamma* secara teratur. Perubahan parameter dengan menaik dan menurunkan nilainya seacara bertahap. Pengujian ini untuk mendapatkan rentang nilai parameter yang akan digunakan pada optimasi berikutnya. Pada Tabel 4.2 menunjukkan skenario dari pengujian ini. Untuk mendapat parameter yang dioptimal, dengan menerapkan *cross validation* dengan nilai *fold* = 10.

Tabel 4. 2 Perancangan uji parameter SVM

C	*gamma*	Akurasi

4.2.3 Uji Coba SVM dengan VNS Dasar

Uji coba pada optimasi ini dengan uji coba parameter VNS, yaitu *kmax* (jumlah neighborhood) dan jumlah iterasi. Masing-masing nilai yang diujikan adalah *kmax* = 50, dan iterasi maksimal = 50. Uji coba pertama adalah uji coba nilai *kmax* untuk mendapatkan *kmax* terbaiknya. Selama uji coba nilai *kmax*, digunakan iterasi 50. Kemudian uji coba iterasi dengan nilai *kmax* yang sudah didapatkan sebelumnya. Tabel 4.3 adalah perancangan pengujian parameter VNS. Nilai *fitness* adalah nilai rata-rata akurasi dari sekian kali percobaan dengan menggunakan SVM, sedangkan nilai *C*, dan *gamma* didapatkan dari pencarian solusi dengan VNS.

Tabel 4. 3 Perancangan pengujian parameter SVM dengan VNS

Iterasi	Tetangga	*C*	*gamma*	fitness

Nilai evaluasi yang semakin tinggi, semakin bagus model yang didapat. Setiap uji coba akan dijalankan sebanyak sekian kali bedasarkan nilai hasil uji coba *simple moving average* (SMA) untuk mendapatkan hasil yang adil. Selain akurasi, akan dicatat waktu komputasi untuk dilakukan pembahasan dan dibandingkan dengan metode usulan, yaitu *nested* RVNS. Tabel 4.4 adalah Tabel skenario setelah dilakukan sebanyak beberapa kali percobaan.

Tabel 4. 4 Perancangan percobaan 5 kali VNS

Rata-rata Akurasi	Maksimal Akurasi	C	*gamma*	Rata-rata waktu komputasi	Maksimal waktu komputasi	Minimal waktu komputasi

4.2.4 Uji Coba *nested* RVNS

Pengujian *nested* RVNS dengan melakukan pengujian parameter *kmax*. Nilai *kmax* yang diujikan berada pada rentang 1 hingga 50. Pengujian ini untuk mendapatkan *kmax* yang dapat memberikan hasil optimal. Nilai *kmax* RVNS mengikuti banyaknya *kmax* yang telah diatur pada *nested* RVNS. Pengujian ini dilakukan untuk menentukan nilai *kmax* terbaiknya berdassarkan hasil optimasi yang diberikan. Kemudian hasil tersebut akan dibandingkan dengan VNS dasar. Pengujian juga dilakukan sebanyak sekian percobaan berdasarkan hasil SMA. Perancangan pengujian ini dapat dilihat pada Tabel 4.5 dibawah. Hasil yang dicatat adalah akurasi dan waktu komputasi.

Tabel 4. 5 Perancangan pengujian parameter SVM dengan *nested* RVNS

m	Tetangga	*C*	*gamma*	Evaluasi

Setelah itu dilakukan penghitungan rata-rata hasilnya dari sekian percobaan seperti pada Tabel 4.6 dibawah.

Tabel 4. 6 Perancangan pengujian 5 kali multiVNS

Rata-rata Akurasi	Maksimal Akurasi	C	γ	Rata-rata waktu komputasi	Maksimal waktu komputasi	Minimal waktu komputasi

Setelah pengujian *nested* RVNS dijalankan, maka didapatkan nilai parameter yang optimal dengan hasilnya yang paling baik. Kemudian hasil dan nilai parameter tersebut dibandingkan hasilnya dengan VNS dasar yang telah diujikan sebelumnya.

BAB 5 HASIL DAN PEMBAHASAN

Pada bab ini akan menerangkan terkait hasil dan pembahasan dari penelitian ini. Dari hasil dan pembahasan inilah yang akan menjadi dasar dari kesimpulan pada bab selanjutnya.

5.1 Hasil Uji Coba *Preprocessing*

5.1.1 Hasil Uji Coba Ekstraksi Fitur

Dari hasil uji coba, dimensi data yang dihasilkan dari kedua metode menghasilkan dimensi yang sama dari TF-IDF ataupun *countvectorizer*. Data yang digunakan dalah data yang telah diproses menggunakan k-Mer. Tabel 5.1 menunjukkan besarnya dimensi data dari kedua metode ekstraksi fitur tersebut. Kolom *k* adalah parameter k-Mer yang menunjukkan nilai dari seberapa banyak substring DNA yang akan diambil.

Tabel 5. 1 Dimensi data setelah ekstraksi fitur menggunakan TF-IDF dan *countvectorizer*

k	TF-IDF	*countvectorizer*
2	233, 16	233, 16
3	233, 64	233, 64
4	233, 256	233, 256
5	233, 1024	233, 1024
6	233, 4096	233, 4096
7	233, 16384	233, 16384
8	233, 65521	233, 65521
9	233, 258031	233, 258031

Berdasarkan Tabel 5.1 diatas, semakin besar nilai *k* semakin besar juga dimensi data yang dihasilkan. Semakin besarnya dimensi data disebabkan karena banyaknya varian *substring* DNA yang diambil. Kemudian dengan ekstraksi fitur menggunakan metode TF-IDF ataupun *countvectorizer* sama-sama menghasilkan panjang dimensi data yang sama. Dapat disimpulkan bahwa dengan menggunakan TF-IDF ataupun countvectorizer menghasilkan dimensi data tidak berbeda. Hal ini dikarenakan kedua metode mula-mula mencari kata-kata yang berbeda sebelum dilakukan pemberian bobot.

Dapat diamati juga bahwa banyaknya fitur hasil ekstraksi berupa nilai berpangkat, yaitu 4^k. Hal ini disebabkan karena ada 4 buah karakter dari DNA yang digunakan dalam ekstraksi fitur. Sedangkan *k* adalah nilai k-Mer yang digunakan. banyaknya fitur didapatkan dari kombinasi penyusun DNA, yaitu A, T, C, dan G. pada $k = 8$ dan $k = 9$, jumlah fitur tidak sebanyak 4^8 dan 4^9 karena jumlah fitur 4^k adalah kemungkinan banyaknya fitur yang akan muncul.

Hasil evaluasi berupa akurasi, *precision*, *recall*, F *measure*, dan MCC dapat dilihat pada Tabel 5.2 dan Tabel 5.3. Tabel 5.2 dengan menggunakan TF-IDF dan Tabel 5.3 menggunakan *countvectorizer*. Data yang digunakan telah dinormalisasi menggunakan *Min-Max* dengan rentang 0 hingga 1.

Tabel 5. 2 Hasil uji coba ekstraksi fitur menggunakan TF-IDF

k	2	3	4	**5**	**6**	7	8	9
Accuracy	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
Precision	0.474	0.543	0.617	0.560	**0.699**	0.696	0.528	0.528
Recall	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
F measure	0.430	0.505	0.605	0.592	**0.693**	0.685	0.421	0.421
MCC	0.064	0.209	0.378	0.321	**0.515**	0.541	0.099	0.099
Waktu	31.2 mili detik	78.1 mili detik	178 mili detik	698 mili detik	**3.49 detik**	20.4 detik	1 menit 50 detik	7 menit 19 detik

Pada Tabel 5.2 data yang digunakan adalah data ekstraksi fitur dengaan TF-IDF. Didapatkan hasil bahwa dengan *k* = 6 memberikan evaluasi yang paling optimal. Dengan akurasi = 0.734, presisi = 0.699, *recall* = 0.734, F *measure* = 0.693 dan MCC sebesar 0.515. Waktu komputasi semakin tinggi mengikuti nilai *k* yang digunakan karena semakin besar dimensi datanya.

Tabel 5. 3 Grafik uji coba ekstraksi fitur menggunakan *countvectorizer*

k	2	3	4	5	**6**	7	8	9
Accuracy	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
Precision	0.474	0.543	0.617	0.560	**0.699**	0.696	0.528	0.528
Recall	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
F measure	0.430	0.505	0.605	0.592	**0.693**	0.685	0.421	0.421
MCC	0.064	0.209	0.378	0.321	**0.515**	0.541	0.099	0.099
Waktu	100 mili detik	78 mili detik	255 mili detik	774 mili detik	**3.6 detik**	21 detik	1 menit 50 detik	8 menit 7 detik

Pada Tabel 5.3 adalah hasil ekstraksi fitur menggunakan *countvectorizer*. Hasil evaluasi lebih kecil dibandingkan dengan menggunakan TF-IDF. Dengan akurasi = 0.734, presisi = 0.699, recall = 0.734, F measure = 0.693 dan MCC = 0.515.

Dapat diamati bahwa dengan menggunakan *countvectorizer* sebagai metode ekstraksi datanya, memberikan hasil klasifikasi yang kurang baik dibandingkan dengan menggunakan TF-IDF seperti yang tertera pada Tabel 5.2 dan Tabel 5.3. Nilai *k* terbaik dari masing-masing metode, yaitu TF-IDF dengan *k* = 5, dan *counvectorizer* dengan *k* = 6. Sedangkan dari sisi waktu komputasi, semakin tinggi data nilai *k* yang digunakan, semakin lama waktu klasifikasinya. Dengan nilai

k yang kecil, waktu yang dibutuhkan cenderung lebih cepat dikarenakan dimensi datanya yang sudah telah berkurang.

5.1.2 Hasil Uji Coba Reduksi Dimensi

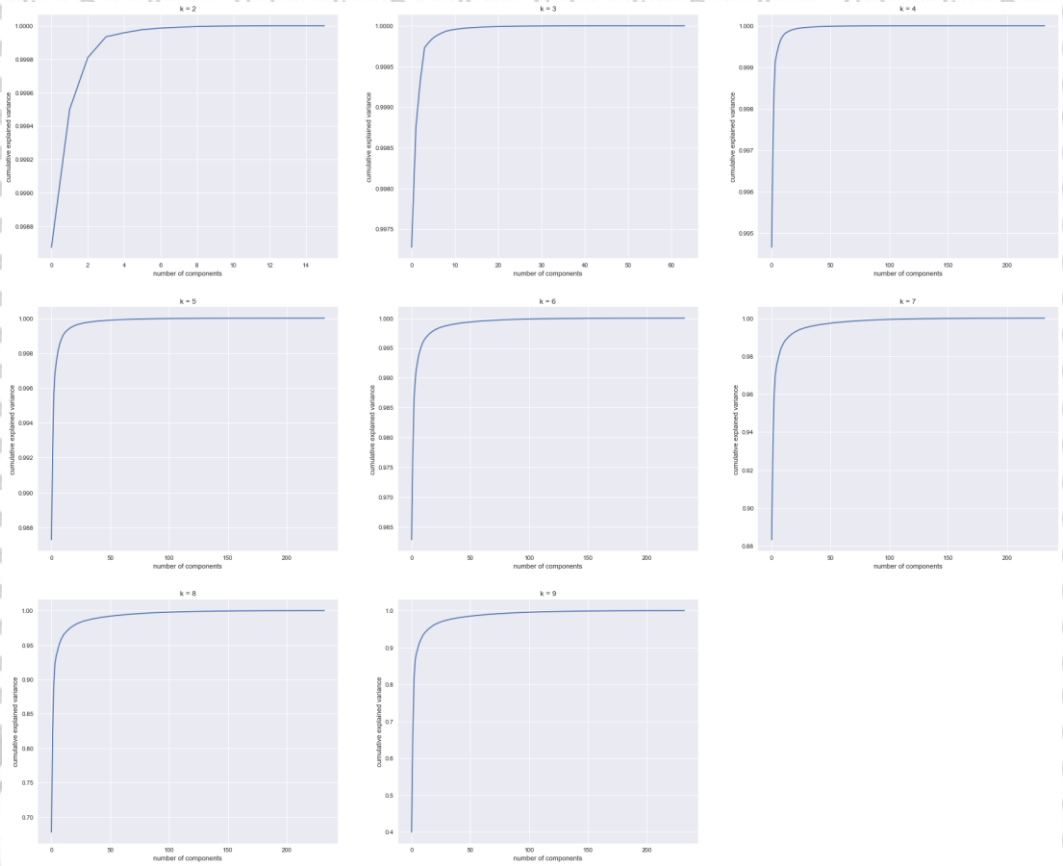
Setelah dilakukan ekstraksi fitur, dilakukan reduksi dimensi data karena data sebelumnya memiliki dimensi yang cukup besar untuk nilai *k* yang semakin tinggi. Uji coba reduksi dimensi ini bertujuan untuk membuktikan bahwa dengan reduksi dimensi dapat mengurangi waktu komputasi dan meningkatkan performa evaluasi klasifikasi. Uji coba dengan menggunakan PCA dan LDA sebagai pembandingnya. Data yang digunakan adalah data hasil dari ekstraksi fitur sebelumnya.

Tabel 5. 4 Hasil uji coba reduksi TF-IDF menggunakan PCA

k	2	3	4	5	**6**	7	8	9
Dimensi	233, 16	233, 64	233, 233	233, 233	**233, 233**	233, 233	233, 233	233, 233
Accuracy	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
Precision	0.474	0.543	0.617	0.560	**0.699**	0.696	0.528	0.528
Recall	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
F measure	0.430	0.505	0.605	0.592	**0.693**	0.685	0.421	0.421
MCC	0.064	0.209	0.378	0.321	**0.515**	0.541	0.099	0.099
Waktu	47.9 ms	65.8 ms	198 ms	242 ms	**327 ms**	435 ms	445 ms	463 ms

Dengan menggunakan PCA dari data hasil TF-IDF, memberikan hasil evaluasi yang sama seperti sebelum dilakukan reduksi dimensi. Nilai *k* terbaik tetap sama, yaitu *k* = 6. Seperti yang tertera pada Tabel 5.4, dengan akurasi = 0.734, presisi = 0.699, *recall* = 0.734, F *measure* = 0.683 dan MCC sebesar 0.515. Perubahan terjadi pada dimensi data dengan *k* > 3. Dimensi data menjadi sama besar dengan banyaknya data, yaitu sebanyak 233. Karena dimensi data berkurang, maka waktu komputasi menjadi jauh lebih cepat daripada sebelum diterapkannya PCA.

Berikutnya akan dicoba untuk mengambil varian data sebanyak 85%, 90%, dan 95% yang kemudian dilakukan perbandingan hasilnya. Varian data dari tiap nilai *k* dari hasil TF-IDF dapat dilihat pada Gambar 5.1 dibawah. Pada gambar tersebut terdapat varian data yang dimiliki dari tiap banyaknya fitur yang dimiliki data. Dapat dilihat bahwa hampir semua data tiap *k* saat variannya bernilai 85% memiliki fitur berjumlah satu. Hal ini terjadi saat *k* bernilai 2 hingga 7. Sedangkan *k* = 8 dan *k* = 9 memiliki jumlah fitur yang lebih dari satu saat banyaknya varian adalah 85%. Dari gambar dapat dilihat grafik kumulatif variannya. Untuk hasil dari 85%, 90%, dan 100% varian data dapat dilihat pada Tabel 5.5.



Gambar 5. 1 Varian data menggunakan PCA dari data hasil TF-IDF

Hasil evaluasi dengan menggunakan 85% varian data dari tiap nilai *k* semakin membaik seiring dengan bertambahnya nilai *k* hingga didapatkan yang terbaik adalah *k* = 9 dengan dimensi data yang didapatkan adalah 233, 4. Akurasi yang diberikan sebesar 0.742, presisi sebesar 0.675, *recall* sebesar 0.742, F *measure* sebesar 0.702 dan MCC sebesar 0.546. nilai ini jauh lebih bagus daripada menggunakan 100% varian data seperti yang tertera pada Tabel 5.4 sebelumnya. Hal ini berarti dengan 100% varian data belum tentu menghasilkan evaluasi yang terbaik. Dengan 85% varian hampir semua data berdasarkan nilai *k*-nya mengalami penurunan dimensi menjadi 1 buah fitur, kecuali pada *k* > 7.

Hasil evaluasi dengan 90% varian data juga dapat dilihat pada Tabel 5.5 dibawah. Untuk *k* yang bernilai antara 2 hingga 6 menghasilkan evaluasi yang sama seperti pada 85% varian data. Disini yang membedakan adalah pada saat *k* > 6 karena terdapat perubahan nilai evaluasi dan dimensi datanya. Tetapi nilai *k* yang terbaik tetap sama yaitu dengan *k* = 9 dan terjadi penurunan hasil evaluasi dibandingakan saat menggunakan 85% varian datanya. Akurasi yang didapat sebesar 0.738, presisi sebesar 0.700, *recall* sebesar 0.738, F *measure* sebesar 0.704, dan MCC sebesar 0.565. Hasil evaluasi dari *k* = 7 dan *k* = 8 juga mengalami kenaikan tetapi bernilai kurang dari saat *k* = 9.

Dengan menggunakan 95% varian data menghasilkan tidak memberikan hasil evaluasi yang lebih baik dari sebelumnya dan ada kecenderungan hasilnya menurun. Disini *k* yang terbaik saat *k* = 8, berubah dari yang sebelumnya *k* = 9, dengan akurasi sebesar 0.734, presisi sebesar 0.691, *recall* sebesar 0.734, F *measure* sebesar 0.699, dan MCC sebesar 0.551. Dengan menggunakan 95% varian juga membuat dimensi data yang bertambah daripada saat variannya 85% dan 90% akan tetapi tidak terlalu memberikan hasil klasifikasi yang lebih baik.

Tabel 5. 5 Hasil evaluasi dari varian data PCA dengan data hasil TF-IDF

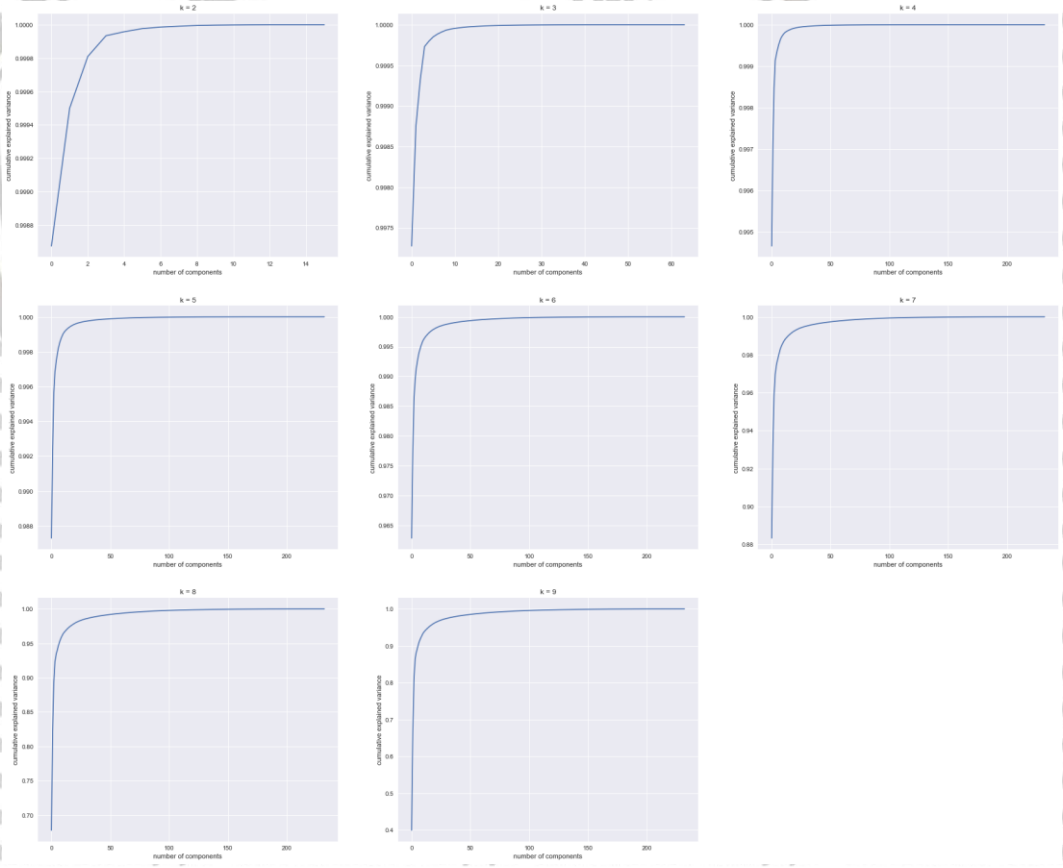
85% Varian Data								
k	2	3	4	5	6	7	8	9
Dimensi	233, 1	233, 1	233, 1	233, 1	233, 1	233, 1	233, 3	**233, 4**
Accuracy	0.562	0.571	0.618	0.639	0.635	0.652	0.738	**0.742**
Precision	0.474	0.471	0.550	0.593	0.571	0.562	0.681	**0.675**
Recall	0.562	0.571	0.618	0.639	0.635	0.652	0.738	**0.742**
F measure	0.430	0.454	0.535	0.568	0.576	0.598	0.699	**0.702**
MCC	0.064	0.107	0.252	0.306	0.294	0.332	0.526	**0.546**
90% Varian Data								
k	2	3	4	5	6	7	8	9
Dimensi	233, 1	233, 1	233, 1	233, 1	233, 1	233, 2	233, 4	**233, 7**
Accuracy	0.562	0.571	0.618	0.639	0.635	0.661	0.738	**0.738**
Precision	0.474	0.471	0.550	0.593	0.571	0.584	0.707	**0.700**
Recall	0.562	0.571	0.618	0.639	0.635	0.661	0.738	**0.738**
F measure	0.430	0.454	0.535	0.568	0.576	0.611	0.698	**0.704**
MCC	0.064	0.107	0.252	0.306	0.294	0.354	0.528	**0.565**
95% Varian Data								
k	2	3	4	5	6	7	8	9
Dimensi	233, 1	233, 1	233, 1	233, 1	233, 1	233, 3	**233, 8**	233, 16
Accuracy	0.562	0.571	0.618	0.639	0.635	0.704	**0.734**	0.721
Precision	0.474	0.471	0.550	0.593	0.571	0.622	**0.691**	0.722
Recall	0.562	0.571	0.618	0.639	0.635	0.704	**0.734**	0.721
F measure	0.430	0.454	0.535	0.568	0.576	0.660	**0.699**	0.691
MCC	0.064	0.107	0.252	0.306	0.294	0.448	**0.551**	0.569

Berdasarkan Tabel 5.5 diatas, hasil evaluasi sangat bervariasi kecuali untuk *k* yang bernilai 1 hingga 6 menghasilkan nilai evaluasi yang sama. Hal ini disebabkan oleh jumlah fitur yang digunakan hanya satu walaupun sudah diambil diambil sekian persen varian dari datanya. Fitur ini berarti mengandung informasi yang banyak dari sebuah data dan fitur lainnya memiliki kandungan informasi yang tidak terlalu banyak. Oleh sebab itu grafik pada Gambar 5.1 meningkat secara signifikan lalu kemudian sangat landai karena dengan satu fitur memiliki pengaruh yang besar. Dapat dilihat juga pada *k* yang bernilai 8 dan 9 dengan mengambil varian datanya sebanyak 85% hingga 95% hanya terjadi penambahan fitur yang sedikit. Hal ini semakin menjelaskan bahwa fitur-fitur lainnya tidak terlalu memiliki pengaruh terhadap data dan memiliki nilai varian yang sangat kecil.

Tabel 5. 6 Hasil uji coba reduksi *countvectorizer* menggunakan PCA

k	2	3	4	5	6	7	8	9
Dimensi	233, 16	233, 64	233, 233	233, 233	**233, 233**	233, 233	233, 233	233, 233
Accuracy	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
Precision	0.474	0.543	0.617	0.560	**0.699**	0.696	0.528	0.528
Recall	0.562	0.601	0.670	0.648	**0.734**	0.717	0.567	0.567
F measure	0.430	0.505	0.605	0.592	**0.693**	0.685	0.421	0.421
MCC	0.064	0.209	0.378	0.321	**0.515**	0.541	0.099	0.099
Waktu	51.9 ms	70.9 ms	181 ms	187 ms	**232 ms**	328 ms	458 ms	546 ms

Hasil uji coba berikutnya adalah dengan data hasil dari *countvectorizer* menggunakan PCA. Nilai evaluasi yang dihasilkan sama seperti sebelum diterapkannya PCA pada data hasil ekstraksi *countvectorizer*. Nilai *k* yang terbaik tetap, yaitu *k* = 6. Seperti yang tertera pada Tabel 5.5, dengan akurasi = 0.734, presisi = 0.699, recall = 0.734, F measure = 0.693 dan MCC = 0.515. Dimensi data berkurang pada *k* > 3, sehingga waktu komputasinya menjadi lebih cepat.



Gambar 5. 2 Varian data menggunakan PCA dari data hasil *countvectorizer*

Sama seperti uji coba PCA sebelumnya, kali ini akan coba reduksi dimensi dengan 85%, 90%, dan 95% varian data hasil dari ekstraksi menggunakan *countvectorizer*. Grafik varian datanya dapat dilihat pada Gambar 5.2 diatas.

Tabel 5. 7 Hasil evaluasi dari varian data PCA dengan data hasil *countvectorizer*

85% Varian Data								
k	2	3	4	5	6	7	8	9
Dimensi	233, 1	233, 1	233, 1	233, 1	233, 1	233, 1	233, 3	**233, 4**
Accuracy	0.562	0.571	0.618	0.639	0.635	0.652	0.738	**0.742**
Precision	0.474	0.471	0.550	0.593	0.571	0.562	0.681	**0.675**
Recall	0.562	0.571	0.618	0.639	0.635	0.652	0.738	**0.742**
F measure	0.430	0.454	0.535	0.568	0.576	0.598	0.699	**0.702**
MCC	0.064	0.107	0.252	0.306	0.294	0.332	0.526	**0.546**
90% Varian Data								
k	2	3	4	5	6	7	8	9
Dimensi	233, 1	233, 1	233, 1	233, 1	233, 1	233, 2	233, 4	**233, 7**
Accuracy	0.562	0.571	0.618	0.639	0.635	0.661	0.738	**0.738**
Precision	0.474	0.471	0.550	0.593	0.571	0.584	0.707	**0.700**
Recall	0.562	0.571	0.618	0.639	0.635	0.661	0.738	**0.738**
F measure	0.430	0.454	0.535	0.568	0.576	0.611	0.698	**0.704**
MCC	0.064	0.107	0.252	0.306	0.294	0.354	0.528	**0.565**
95% Varian Data								
k	2	3	4	5	6	7	8	9
Dimensi	233, 1	233, 1	233, 1	233, 1	233, 1	233, 3	**233, 8**	233, 16
Accuracy	0.562	0.571	0.618	0.639	0.635	0.704	**0.734**	0.721
Precision	0.474	0.471	0.550	0.593	0.571	0.622	**0.691**	0.722
Recall	0.562	0.571	0.618	0.639	0.635	0.704	**0.734**	0.721
F measure	0.430	0.454	0.535	0.568	0.576	0.660	**0.699**	0.691
MCC	0.064	0.107	0.252	0.306	0.294	0.448	**0.551**	0.569

Hasil-hasil pada Tabel 5.7 sangat mirip dengan hasil pada Tabel 5.5, tidak terjadi perbedaan hasil. Hal ini dikarenakan hasil ekstraksi dari TF-IDF ataupun *countvectorizer* menghasilkan data yang sama. Saat diambil 85% dan 90% varian data didapatkan yang terbaik pada *k* = 9, saat 95% varian data yang diambil didapatkan yang terbaik pada *k* = 8. Dari keseluruhan yang terbaik adalah *k* = 9 dengan 80% varian data, dengan akurasi sebesar 0.742, presisi sebesar 0.675, *recall* sebesar 0.742, F *measure* sebesar 0.702 dan MCC sebesar 0.546.

Berdasarkan hasil uji coba diatas, hasil reduksi dimensi menggunakan PCA untuk data hasil ekstraksi dengan TF-IDF dan *countvectorizer* menghasilkan dimensi yang sama besar. Hasil evaluasi yang dihasilkanpun cenderung sama dari setelah ataupun sebelum dilakukan reduksi dimensi. Hal ini menunjukkan bahwa dengan PCA mampu mengurangi dimensi data dan tetap mempertahankan informasi yang dikandung oleh data. Selain itu terbukti juga bahwa dengan PCA dapat mengurangi waktu komputasi karena dimensi data lebih kecil daripada sebelumnya. Dengan PCA juga dapat meningkatkan hasil evaluasi dari metode klasifikasi yang digunakan tergantung dari banyaknya varian data yang digunakan.

Setelah diterapkan PCA, kemudian menerapkan metode pembandingnya, yaitu LDA. LDA berbeda dari PCA karena berbasis *supervised learning* dengan mempertimbangkan kelas data dalam melakukan reduksi dimensi data. Sama seperti PCA, LDA juga diterapkan pada data hasil ekstraksi menggunakan TF-IDF dan *countvectorizer*.

Tabel 5. 8 Hasil uji coba reduksi TF-IDF menggunakan LDA

k	2	3	4	**5**	6	7	8	9
Dimensi	233, 5	233, 5	233, 5	233, 5	233, 5	233, 5	233, 5	233, 5
Accuracy	0.721	0.764	0.918	**0.927**	0.923	0.918	0.914	0.906
Precision	0.665	0.708	0.910	**0.920**	0.915	0.912	0.907	0.900
Recall	0.721	0.764	0.918	**0.927**	0.923	0.918	0.914	0.906
F measure	0.683	0.725	0.909	**0.920**	0.915	0.910	0.905	0.896
MCC	0.518	0.597	0.859	**0.875**	0.867	0.859	0.852	0.837
Waktu	0.0399 detik	0.0408 detik	0.0538 detik	**0.0519 detik**	0.0479 detik	0.0419 detik	0.0449 detik	0.0378 detik

Untuk hasil uji coba menggunakan LDA dari data TF-IDF menghasilkan nilai evaluasi yang beragam untuk tiap data dengan *k* yang berbeda, bahkan jauh lebih baik dari sebelum diterapkanya LDA. Nilai akurasi cenderung meningkat dan menurun pada *k* bernilai 6 hingga 9, dan yang paling optimal berada pada *k* = 5 dengan akurasi sebesar 0.927, presisi sebesar 0.920, recall sebesar 0.927, F *measure* sebesar 0.920, dan MCC sebesar 0.875 seperti yang ditunjukkan pada Tabel 5.8 diatas. Waktu komputasi yang dibutuhkan cenderung sama dan stabil karena dimensi yang dihasilkan tiap *k*-nya adalah 5.

Kemudian hasil uji coba dengan menggunakan LDA dari data hasil *countvectorizer* dapat dilihat pada Tabel 5.9. Pada Tabel tersebut menunjukkan bahwa nilai evaluasi terbaik berada pada *k* = 5 dengan akurasi sebesar 0.927, presisi sebesar 0.920, recall sebesar 0.927, F *measure* sebesar 0.920, dan MCC sebesar 0.875. Waktu komputasi yang dihasilkan cenderung sama karena dimensi data hasil dari LDA adalah 5 untuk tiap nilai *k*-nya.

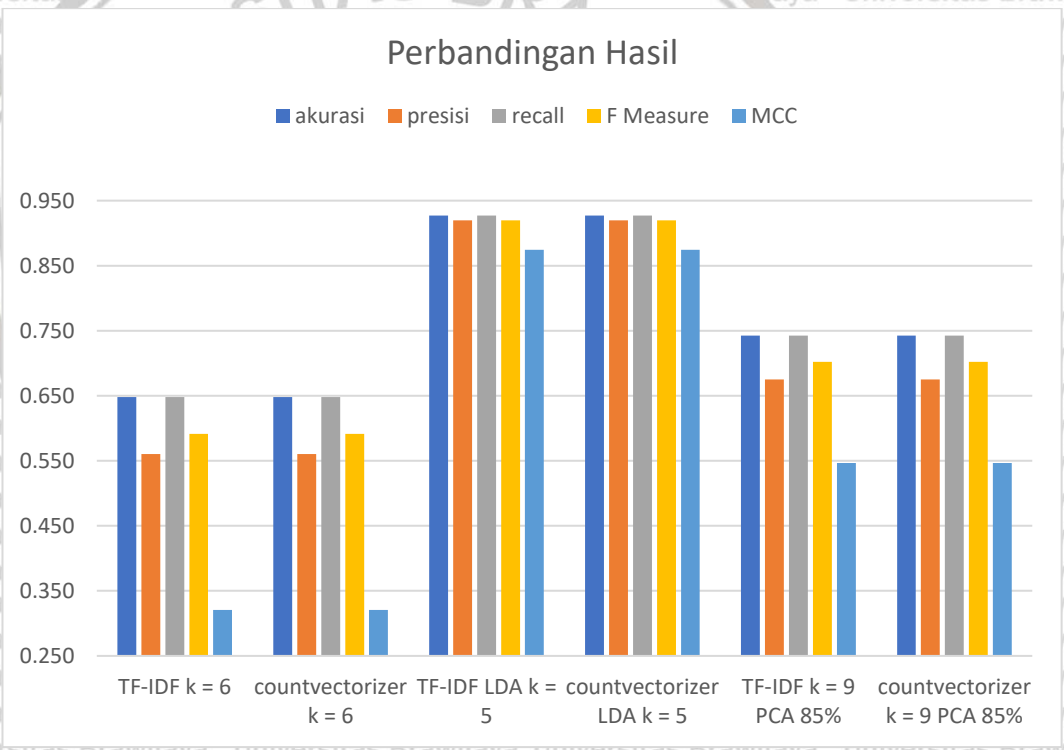
Tabel 5. 9 Hasil uji coba reduksi *countvectorizer* menggunakan LDA

k	2	3	4	**5**	6	7	8	9
Dimensi	233, 5	233, 5	233, 5	**233, 5**	233, 5	233, 5	233, 5	233, 5
Accuracy	0.721	0.764	0.918	**0.927**	0.923	0.918	0.914	0.906
Precision	0.665	0.708	0.910	**0.920**	0.915	0.912	0.907	0.900
Recall	0.721	0.764	0.918	**0.927**	0.923	0.918	0.914	0.906
F measure	0.683	0.725	0.909	**0.920**	0.915	0.910	0.905	0.896
MCC	0.518	0.597	0.859	**0.875**	0.867	0.859	0.852	0.837
Waktu	0.0549 detik	0.0479 detik	0.0608 detik	**0.0509 detik**	0.0509 detik	0.0478 detik	0.0379 detik	0.0389 detik

Apabila diamati, hasil reduksi dimensi dengan LDA untuk data hasil TF-IDF dan *countvectorizer* sama-sama memberikan nilai *k* yang terbaiknya, yaitu *k* = 5. Sedangkan dimensi data yang dihasilkan juga sama, yaitu dengan jumlah fitur

sebanyak 5 dan waktu komputasi juga cenderung stabil. Jumlah fitur sebanyak 5 karena pada LDA untuk data *multi-class* akan menghasilkan fitur sebanyak C-1, dimana C adalah banyaknya kelas data (Xie, 2015; Wang and Chen, 2017). Pada kasus ini, TF-IDF ataupun *countvectorizer* dengan LDA memiliki performa yang sama bagusnya.

Dilihat dimensi data yang dihasilkan menggunakan LDA, dimensi data cenderung sama dengan dimensi 233, 5. Dimensi ini jauh lebih kecil daripada dengan diterapkannya PCA dan secara otomatis waktu komputasi yang dibutuhkan jauh lebih cepat dengan menggunakan LDA. Sedangkan apabila diamati dari sisi evaluasi, dengan LDA memberikan evaluasi yang lebih bagus dan untuk setiap nilai *k* yang berbeda. Hal ini dikarenakan LDA berbasis *supervised learning* yang membuat data cenderung mendekat ke kelasnya yang sama dan memaksimalkan batas antar kelas yang berbeda (Ilias et al., 2016). Sedangkan PCA berkebalikan dengan LDA, PCA berbasis *unsupervised learning* yang mereduksi data tanpa mempertimbangkan kelas data tersebut.



Gambar 5. 3 Hasil uji coba *preprocessing* data

Hasil terbaik dari pengujian *preprocessing* dapat dilihat pada Gambar 5.3 diatas. Berdasarkan gambar tersebut, data hasil PCA dan LDA untuk masing-masing hasil ekstraksi menggunakan TF-IDF maupun *countvectorizer* tidak memberikan perbedaan hasil. Hal ini dikarenakan hasil ekstraksi TF-IDF dan *countvectorizer* menghasilkan data yang sama. Data yang sama ini disebabkan saat pembobotan TF-IDF dikalikan dengan nilai satu yang dihasilkan dari perhitungan

IDF dengan rumus $\log\left(\frac{N}{df(t_j)}\right)+1$. Nilai satu dihasilkan karena hampir semua dokumen memiliki substring yang sama seperti pada dokumen-dokumen lainnya. Pada TF-IDF ada dua komponen perhitungan, yaitu TF dan IDF. Saat nilai IDF bernilai sama dengan 1, maka yang tersisa adalah nilai TF itu sendiri. Nilai TF didapatkan dari frekuensi banyaknya data yang muncul pada suatu dokumen. TF itu sendiri bernilai sama seperti *countvectorizer* dengan menghitung banyaknya kata yang muncul. Oleh sebab itu, dapat disimpulkan bahwa pada penelitian ini dengan menggunakan TF-IDF ataupun *countvectorizer* tidak memberikan perbedaan hasil. Untuk selanjutnya dapat dipilih dengan menggunakan data TF-IDF ataupun *countvectorizer* pada k = 5 dengan reduksinya yaitu LDA.

5.2 Hasil Uji Coba Parameter SVM

Uji coba parameter untuk menentukan batas rentang nilai yang akan dicari dengan optimasi menggunakan VNS. Pada klasifikasi ini menggunakan data yang telah dilakukan *preprocessing* sebelumnya, yaitu dengan *k* = 5, ekstraksi fitur menggunakan TF-IDF dan reduksi dimensi menggunakan LDA.

Tabel 5. 10 Hasil uji coba parameter SVM

C	*gamma*	Akurasi	Presisi	Recall	F *measure*	MCC
1	1	0.939914	0.931080	0.939914	0.932982	0.896586
2	1	0.944206	0.934768	0.944206	0.937263	0.903865
3	1	0.944206	0.934768	0.944206	0.937263	0.903865
0.9	1	0.927039	0.919659	0.927039	0.919797	0.874534
1	2	0.918455	0.911852	0.918455	0.910232	0.859406
1	0.9	0.944206	0.934768	0.944206	0.937263	0.903865

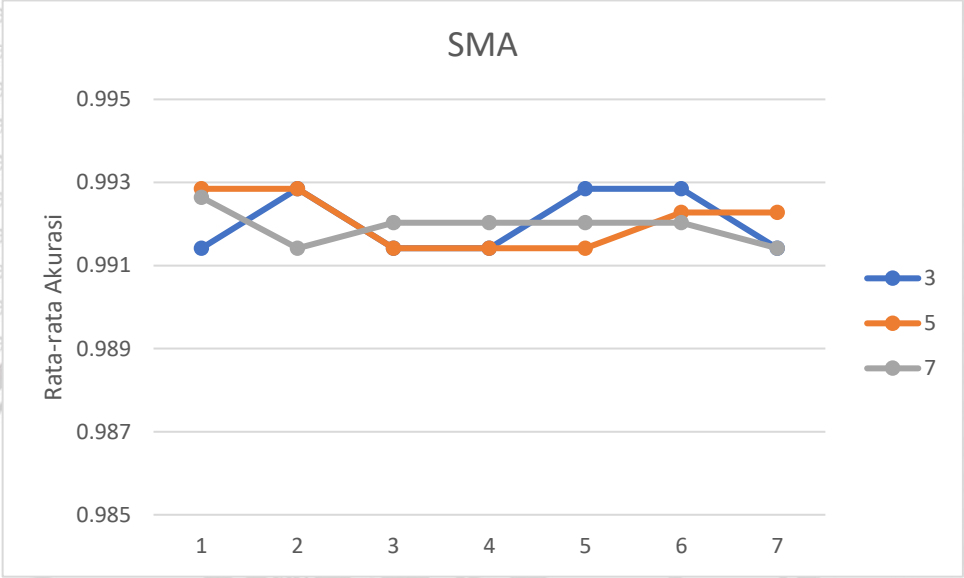
Dapat dilihat pada Tabel 5.8 diatas bahwa dengan nilai *C* yang dinaikkan menjadi *C* = 2 dan nilai *gamma* = 1, hasil evaluasi akurasi, presisi, *recall*, *F measure*, dan MCC berubah menjadi lebih baik daripada *C* = 1 dan *gamma* = 1. Kemudian saat nilai *C* dinaikkan lagi tidak terjadi perubahan pada nilai evaluasinya. Ini berarti untuk rentang nilai *C* yang akan di optimasi adalah 0 hingga 2. Berikutnya uji coba untuk menentukan batasan rentang nilai *gamma*. Saat nilai *gamma* dinaikkan menjadi 2 dengan *C* = 1, hasil evaluasi menurun. Selanjutnya coba menurunkan nilai *gamma* menjadi 0.9 dan hasil evaluasinya mengalami peningkatan.

Hasil akhir untuk rentang nilai parameter yang digunakan adalah *C* dengan nilai 0 hingga 2, dan *gamma* dengan nilai 0 hingga 1. Nilai batasan inilah yang digunakan pada tahap optimasi berikutnya untuk membatasi ruang pencarian parameter SVM dengan VNS.

5.3 Hasil Uji Coba Simple Moving Average

Pengujian ini dilakukan untuk menentukan berapa kali banyak percobaan yang akan dilakukan oleh metode optimasi yang akan digunakan. Untuk

menentukannya dengan menerapkan SMA. Pengujian kali ini menggunakan VNS dasar dengan parameter *kmax* = 50 dan iterasi = 50 untuk optimasi SVM. Percobaan dilakukan sebanyak 7 kali untuk mengetahui garis trend. Kemudian dilakukan rata-rata hasil akurasi masing-masing dari tiap percobaan sebagai indikasi keterbaikan hasil optimasi. Tujuan dilakukannya percobaan SMA adalah untuk mengetahui saat percobaan berapakah yang dapat memberikan rata-rata hasil yang stabil. Hasil uji coba dapat dilihat pada gambar 5.4 dibawah.



Gambar 5. 4 Hasil uji coba SMA

Dari gambar 5.4 diatas, banyaknya periode percobaan yang dilakukan yaitu 3, 5, dan 7. Berdasarkan hasil yang ditunjukkan, dengan percobaan yang sedikit memberikan grafik yang sedikit fluktuatif. Dari percobaan tersebut, dengan menggunakan 7 periode percobaan memberikan garis trend yang mulai cenderung stabil. Dapat diasumsikan bahwa dengan 7 atau lebih dari 7 kali percobaan, memberikan rata-rata hasil yang lebih stabil. Sehingga pada penelitian ini akan menggunakan 10 kali percobaan pada tahap optimasi dan kemudian diambil nilai rata-rata hasilnya.

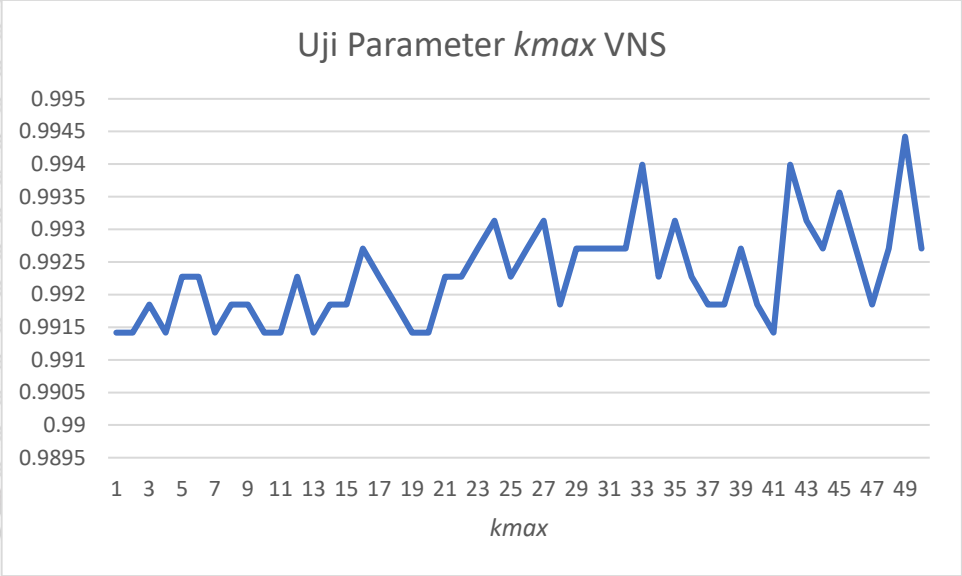
5.4 Hasil Uji Coba SVM dengan VNS Dasar

VNS adalah salah satu algoritme pencarian yang digunakan dalam hal optimasi. Pada penelitian ini menerapkan VNS sebagai algoritme untuk optimasi parameter dari SVM. Ada dua parameter yang akan dioptimasi, yaitu parameter *C* dan parameter *gamma*. Untuk data yang digunakan, dipilih data hasil ekstraksi menggunakan data hasil TF-IDF dan LDA dengan *k* = 5.

5.4.1 Hasil Uji Coba Parameter *kmax*

Parameter yang diujikan pada bagian ini adalah parameter *neighborhood* (*kmax*) dan iterasi dari VNS. Untuk mengetahui seberapa baik solusi dengan

menggunakan fungsi *fitness*. Pada penelitian ini nilai *fitness* yang digunakan adalah nilai akurasi dari klasifikasi. Parameter yang pertama kali diuji adalah parameter *neighborhood* dengan *kmax* bernilai 1 hingga 50 dan iterasi = 50. Gambar 5.5 adalah hasil rata-rata pengujian parameter *kmax* dari 10 kali percobaan.



Gambar 5. 5 Grafik hasil uji coba *kmax* dari VNS

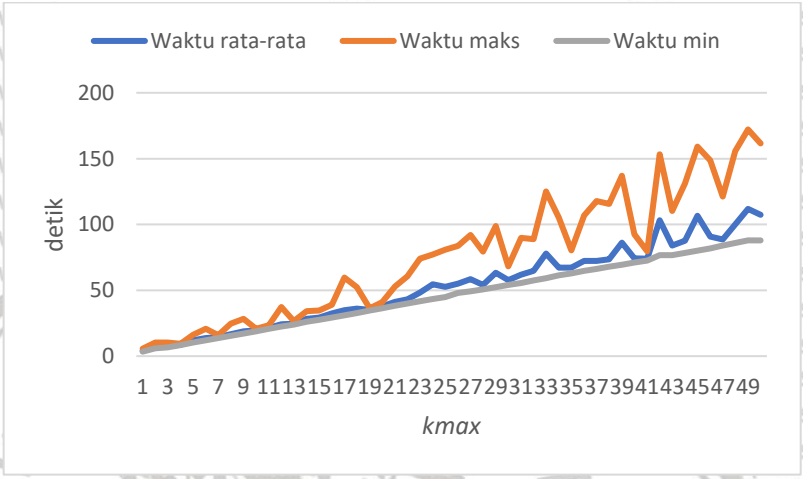
Dari grafik pada Gambar 5.5 dapat diketahui bahwa dengan nilai *kmax* yang rendah belum tentu dapat menemukan solusi terbaiknya. Grafik uji parameter *kmax* terlihat fluktuatif karena terdapat parameter random yang membuatnya tidak selalu bisa mendapatkan solusi yang lebih baik. Dapat diamati dengan seiring meningkatnya nilai *kmax*, VNS dapat menemukan solusi yang semakin bagus. Seperti pada saat *kmax* bernilai 1 hingga 33 grafiknya cenderung meningkat walau terkadang tidak menemukan solusi yang bagus pada beberapa titik. Antara *kmax* 33 dan 42 terjadi penurunan hasil dikarenakan VNS adalah algoritma yang bersifat *stochastic*, yaitu selalu dapat menghasilkan perbedaan solusi walaupun diterapkan pada permasalahan yang sama. Pada pengujian ini dengan *kmax* = 49 telah mendapatkan solusi yang paling optimal. Solusi tersebut bernilai *C* = 1.22363 dan *gamma* = 0.00029 dengan akurasi sebesar 0.99570. Hasil secara rinci tertera pada Tabel 5.11.

Tabel 5. 11 Hasil uji coba *kmax* dari VNS

kmax	Rata-rata Akurasi	Maksimal Akurasi	*C*	*gamma*	Rata-rata waktu komputasi
49	**0.9944**	**0.995708**	**1.2236**	**0.0002**	**111.8334**

Berdasarkan Tabel 5.11, rata-rata waktu komputasi yang dibutuhkan lebih dari 1 menit untuk menjalankan VNS. Grafik waktu komputasi dapat dilihat pada Gambar 5.6 dibawah. Semakin besar nilai *kmax* yang digunakan, cenderung semakin besar juga waktu komputasi yang dibutuhkan. Akan tetapi waktu

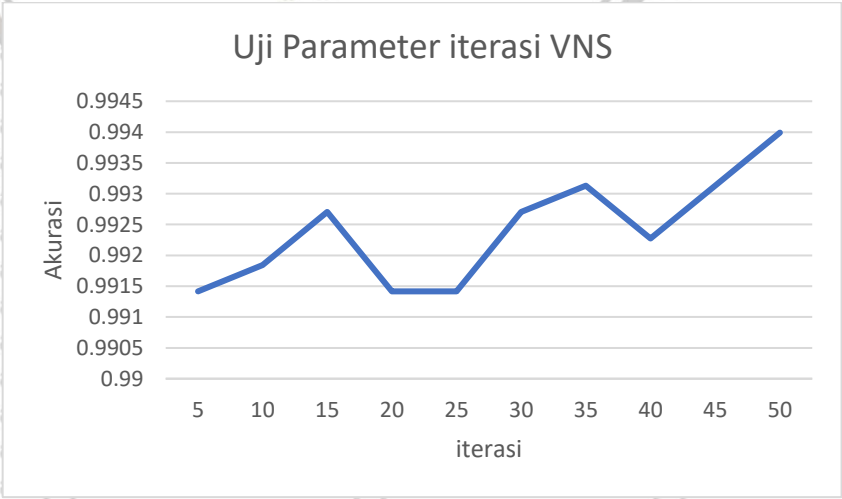
komputasi tidak selalu tinggi untuk kenaikan nilai *kmax*. Hal ini disebabkan adanya fungsi *change neighborhood* yang menyebabkan waktu semakin lama apabila ditemukan solusi yang baru dari proses pencarian lokalnya. Apabila tidak ditemukannya solusi yang lebih baik, maka VNS akan berjalan lebih cepat. Dapat diamati juga saat waktu komputasi yang tinggi, cenderung selalu dapat menghasilkan nilai akurasi yang lebih baik daripada saat waktu komputasi yang rendah.



Gambar 5. 6 Grafik waktu komputasi pengujian *kmax*

5.4.2 Hasil Uji Coba Parameter Iterasi

Nilai *kmax* telah didapatkan, kemudian dilakukan pengujian iterasi dari *local search* VNS. Parameter yang digunakan yaitu *kmax* = 44 karena sebelumnya dengan nilai tersebut VNS sudah dapat menemukan solusi terbaiknya. Iterasi yang diujikan adalah 5 hingga 50 dengan interval 5. Sama seperti sebelumnya, pengujian dilakukan sebanyak 10 kali kemudian didapatkan rata-rata hasilnya dan ditunjukkan pada Gambar 5.7.



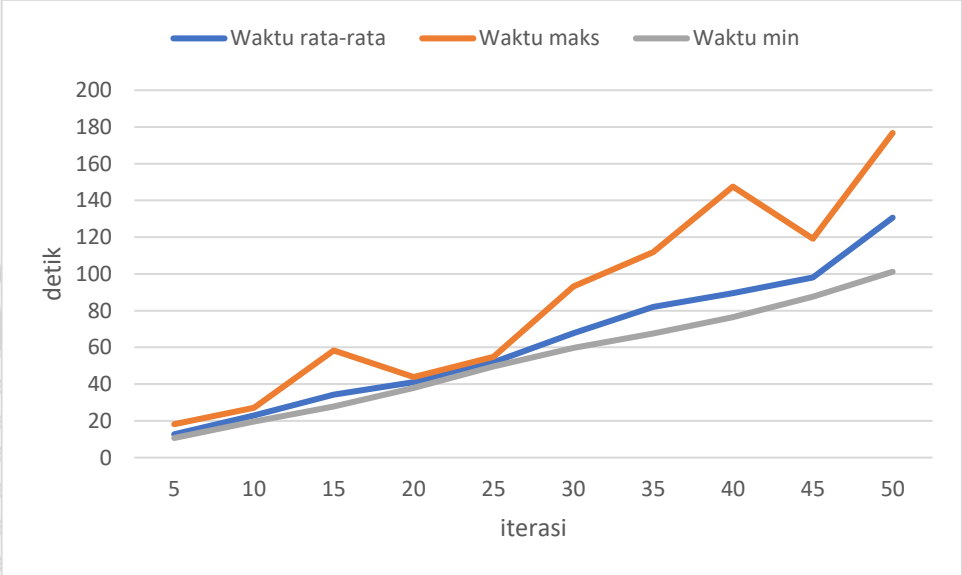
Gambar 5. 7 Grafik uji coba iterasi dari VNS

Dari Gambar 5.7 diatas, dapat dilihat bahwa dengan jumlah iterasi yang kecil menghasilkan rata-rata akurasi yang rendah. Seiring bertambahnya nilai parameter iterasi, rata-rata akurasi yang dihasilkan semakin bagus tapi hal ini tidak selalu berlaku. Ini berarti dengan menaikkan iterasi tidak ada jaminan akan bisa mendapatkan solusi yang lebih baik. Seperti pada saat iterasi = 20, hasil solusi yang diberikan tidak terlalu bagus dibandingkan dengan iterasi = 15 sebelumnya. Pada pengujian ini didapatkan satu kali solusi yang terbaik, yaitu pada saat iterasi bernilai 50. Hasil secara lebih rinci dapat dilihat pada Tabel 5.12 dibawah.

Tabel 5. 12 Hasil uji coba iterasi dari VNS

iterasi	Rata-rata Akurasi	Maksimal Akurasi	*C*	*gamma*	Rata-rata waktu komputasi
50	**0.9939**	**0.995708**	**1.2135**	**0.0001**	**0.000282**

Berdasarkan Tabel diatas, rata-rata akurasi dan akurasi maksimal yang diberikan sama-sama memiliki nilai 0.99399 dan 0.995708. Dapat ditentukan iterasi terbaik disini dengan nilai 50 dengan hasil paling optimalnya dengan nilai *C* = 1.2135 dan *gamma* = 0.0001. Waktu komputasi yang diperlukan pada iterasi = 50 sekitar lebih dari 1.5 menit.



Gambar 5. 8 Grafik perbandingan waktu uji iterasi VNS

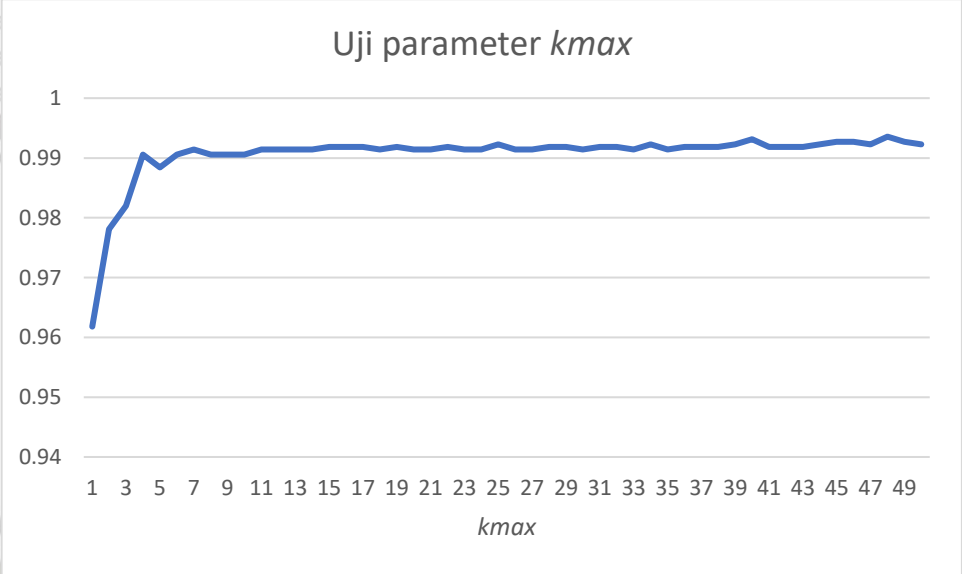
Gambar 5.8 diatas merupakan grafik waktu komputasi saat melakukan pengujian iterasi. Terlihat grafik tersebut terus meninggi dan sedikit fluktuatif yang disebabkan oleh sifat dari *change neighborhood* dari VNS. Grafik yang meningkat karena pencarian yang lebih lama dan luas sehingga membutuhkan waktu yang lebih. Hal ini juga menunjukkan bahwa dengan menambah iterasi dari pencarian lokal belum tentu menghasilkan solusi yang lebih baik berdasarkan perbandingan

hasil dari Tabel 5.11 dan 5.12. Tetapi masih dapat menemukan solusi dengan hasil akurasi maksimal yang sama besarnya.

5.5 Hasil Uji Coba SVM dengan *nested* RVNS

5.5.1 Hasil Uji Coba Parameter *kmax*

Uji coba parameter *kmax* dengan nilai 50. Pengujian ini dilakukan sebanyak 10 kali dan dicatat nilai rata-rata hasilnya. Parameter *kmax* ini menentukan sebanyak berapakali metode ini akan dijalankan dan juga menjadi parameter kondisi berhenti dari *nested* RVNS. Tidak ada parameter lain yang diujikan pada pengujian ini karena tidak ada proses pencarian lokal sehingga tidak diperlukan parameter iterasi dari pencarian lokal. Hasil pengujian dapat dilihat pada Gambar 5.9 dibawah.



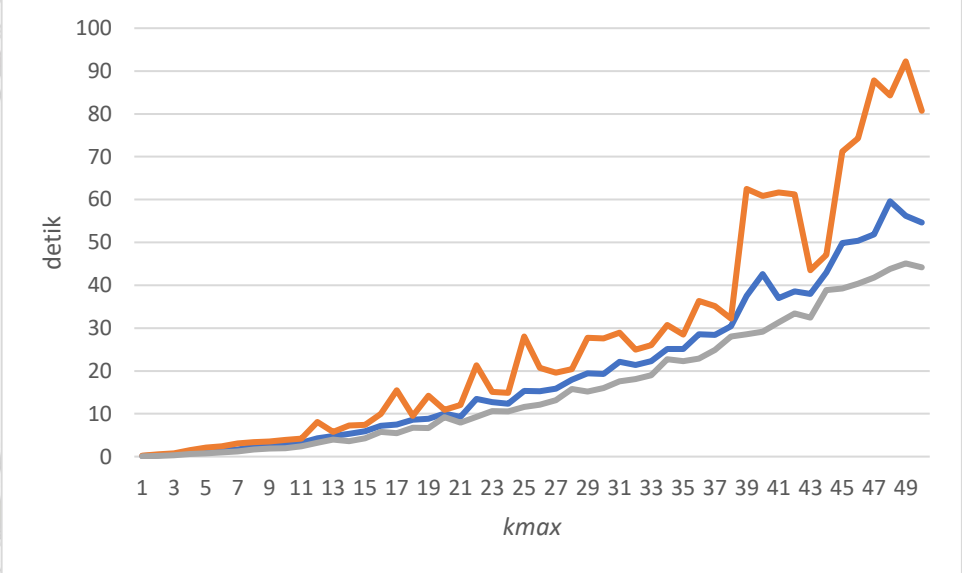
Gambar 5. 9 Grafik uji coba *kmax* dari *nested* RVNS

Berdasarkan grafik pada gambar diatas, dengan *kmax* yang kecil, *nested* RVNS belum bisa menemukan solusi yang baik. Seiring bertambahnya nilai *kmax*, *nested* RVNS menemukan solusi yang cenderung lebih baik. Terlihat pada saat *kmax* bernilai 1 hingga 7, grafik cenderung meningkat dan menemukan solusi terbaru kecuali saat *kmax* bernilai 5. Hal ini disebabkan oleh sifat random dari parameter *kmax* pada RVNS yang belum tentu bisa meraih solusi yang lebih baik dari sebelumnya. Saat *kmax* bernilai 11 hingga 50, grafik cenderung stabil yang berarti hasil akurasi dari solusi yang dihasilkan tidak jauh berbeda. Pada beberapa titik *nested* RVNS menghasilkan rata-rata akurasi yang lebih baik dari beberapa percobaan sebelumnya, yaitu saat *kmax* bernilai 25, 34, 40 dan mendapatkan solusi maksimalnya pada saat *kmax* bernilai 48 dan rincian hasilnya disajikan pada Tabel 5.13 dibawah.

Tabel 5. 13 Hasil uji coba parameter *kmax nested* RVNS

kmax	Rata-rata Akurasi	Maksimal Akurasi	*C*	*gamma*	Rata-rata waktu komputasi
48	**0.9932**	**0.995708**	**1.3803**	**0.0001**	**59.58249**

Berdasarkan Tabel 5.13 diatas dapat dilihat bahwa rata-rata akurasi bernilai 0.99325 dan maksimal akurasinya mencapai 0.995708. Disini dipilih dengan *kmax* = 48 sebagai solusi terbaik dengan rata-rata waktu komputasi yang rendah, yaitu 28.42596 detik sudah bisa mengeluarkan solusi optimalnya. Dengan nilai *C* = 1.38034 dan *gamma* = 0.00019. Untuk maksimal waktu komputasi bisa mencapai 2 kali lebih besar dari waktu minimal karena disebabkan *processor* yang digunakan untuk melakukan komputasi sedang menjalankan proses lainnya. Sedangkan waktu komputasi minimal tidak terlalu jauh berbeda dari rata-rata waktu komputasinya.



Gambar 5. 10 Grafik waktu komputasi *nested* RVNS

Pada gambar 5.10 tertera grafik waktu komputasi keseluruhan pengujian dari tiap nilai *kmax*. Grafik cenderung naik dan membutuhkan lebih banyak waktu komputasi. Saat *kmax* bernilai 46 dan 48 terjadi gejolak waktu komputasi karena didalam *nested* RVNS terdapat banyak perulangan dari nilai *k*.

Dari hasil keseluruhan pengujian *nested* RVNS, cenderung memberikan hasil yang stabil dan bertahap menemukan solusi yang lebih bagus. Hal ini dikarenakan pada *nested* RVNS apabila ditemukan solusi yang baru maka akan merubah nilai *k* dari *kmax* menjadi nilai pada saat inisialisasi sehingga eksplorasi pencarian solusi menjadi lebih luas.

5.6 Hasil Perbandingan Uji Coba VNS dan *nested* RVNS

Sebagai perbandingan, akan dilakukan pengujian dengan membandingkan hasil VNS dasar dengan *nested* RVNS. Pengujian dilakukan dengan menggunakan nilai parameter berdasarkan hasil uji coba parameter. Tujuan pengujian ini untuk membuktikan metode optimasi manakah yang lebih unggul untuk diterapkan pada penelitian ini. Pengujian dilakukan sebanyak 10 kali yang kemudian dilakukan rata-rata pada hasilnya. Parameter VNS yang akan digunakan yaitu *kmax* = 49 dan iterasi = 50. Sedangkan parameter *nested* RVNS adalah dengan *kmax* = 48. Hasil perbandingan dapat dilihat pada Tabel 5.14 dibawah.

Tabel 5. 14 Tabel parameter perbandingan

Metode	Rata-rata Akurasi	Maksimal Akurasi	*C*	*gamma*	Rata-rata waktu komputasi
VNS Dasar	0.99270	0.995708	0.88346	0.00038	120.9853
***Nested* RVNS**	**0.99313**	**0.995708**	**0.59890**	**0.00044**	**47.01916**

Berdasarkan Tabel diatas, hasilnya adalah dengan metode yang diusulkan dapat memberikan hasil rata-rata akurasi yang lebih baik dan menghabiskan waktu komputasi relatif lebih singkat dibandingkan dengan VNS dasar. Dengan *nested* RVNS dapat mendapatkan akurasi maksimal sebesar 0.995708 dengan *C* = 0.59890 dan *gamma* = 0.59890. waktu yang dibutuhkanpun relatif lebih singkat dengan rata-rata 47.01916 detik. Waktu ini 2.5 kali lebih cepat detik daripada VNS dasar.

Nested RVNS dapat menemukan solusi yang lebih baik disebabkan didalamnya terdapat banyak sekali eksplorasi ruang pencarian yang dinotasikan dengan parameter *kmax*. Setiap menemukan solusi, ruang pencarian akan diperluas lagi hinggan nilai *k* bernilai sama dengan nilai *kmax* yang merupakan kondisi berhenti dari *nested* RVNS. Sedangkan waktu komputasi bisa menjadi lebih cepat karena tidak adanya proses pencarian lokal didalamnya. Pada VNS dasar perluasan ruang pencarian dipengaruhi oleh solusi hasil pencarian lokal. Apabila solusi hasil pencarian lokal lebih bagus daripada solusi yang ada, baru kemudian dilakukan penambahan nilai yang mempengaruhi luasnya ruang pencarian.

Hasil prediksi berdasarkan parameter dari *nested* RVNS disajikan pada Tabel 5.15 dibawah. Dapat dilihat bahwa hampir semua prediksi sesuai dengan kelas aktualnya. Tetapi hanya ada satu data dengan kelas prediksi yang tidak tepat dengan kelas aktualnya. Hal ini yang mengakibatkan akurasi tidak sepenuhnya 100%. Kelas data dinotasikan dengan angka yang berarti kelas 1 = Indo-Oceanic, 2 = East Asian, 3 = Central Asia, 4 = Euro-America, 5 = West African 1, dan 6 = West African 2.

Tabel 5. 15 *Confusion matrix*

Prediksi	1	2	3	4	5	6
Aktual						
1	6	1	0	0	0	0
2	0	74	0	0	0	0
3	0	0	12	0	0	0
4	0	0	0	130	0	0
5	0	0	0	0	7	0
6	0	0	0	0	0	3

Berdasarkan Tabel 5.15 diatas, didapatkan nilai TP (*true positive*) = 232, FP (*false positive*) = 1, TN (*true negative*) = 0, dan FN (*false negative*) = 0. Sehingga diperoleh akurasi = 0.995708, presisi = 0.995765, *recall* = 0. 995708, F *measure* = 0. 995708, dan MCC = 0.992659. Hasil perbandingan lebih detil dari *confusion matrix* diatas, disajikan pada Tabel 5.16 dibawah. Keseluruhan data memiliki kelas prediksi yang sama kecuali pada data ke-enam dengan nama file 'NZ_CP009427.1.fasta' yang seharusnya memiliki kelas 1 (Indo-Oceanic) tetapi diprediksi kedalam kelas 2 (East Asian).

Tabel 5. 16 Perbandingan kelas aktual dan prediksi

No.	Nama file	Kelas Aktual	Kelas Prediksi
1	'AP018033.1.fasta',	1	1
2	'AP018034.1.fasta',	1	1
3	'AP018035.1.fasta',	1	1
4	'AP018036.1.fasta',	1	1
5	'CP010895.1.fasta',	1	1
6	'NZ_CP009427.1.fasta',	1	2
7	'NZ_CP010968.1.fasta',	1	1
8	'AP010918.1.fasta',	2	2
9	'AP014573.1.fasta',	2	2
10	'AP017901.1.fasta',	2	2
11	'CP007809.1.fasta',	2	2
12	'CP008959.1.fasta',	2	2
13	'CP008960.1.fasta',	2	2
14	'CP008961.1.fasta',	2	2
15	'CP008962.1.fasta',	2	2
16	'CP008963.1.fasta',	2	2
17	'CP008964.1.fasta',	2	2
18	'CP008965.1.fasta',	2	2
19	'CP008966.1.fasta',	2	2
20	'CP008967.1.fasta',	2	2
21	'CP008968.1.fasta',	2	2
22	'CP008969.1.fasta',	2	2
23	'CP008970.1.fasta',	2	2
24	'CP008971.1.fasta',	2	2
25	'CP008972.1.fasta',	2	2

26	'CP008973.1.fasta',	2	2
27	'CP008974.1.fasta',	2	2
28	'CP008975.1.fasta',	2	2
29	'CP008976.1.fasta',	2	2
30	'CP008977.1.fasta',	2	2
31	'CP008978.1.fasta',	2	2
32	'CP008979.1.fasta',	2	2
33	'CP008980.1.fasta',	2	2
34	'CP008981.1.fasta',	2	2
35	'CP008982.1.fasta',	2	2
36	'CP008983.1.fasta',	2	2
37	'CP009100.1.fasta',	2	2
38	'CP009101.1.fasta',	2	2
39	'CP009426.1.fasta',	2	2
40	'CP009427.1.fasta',	2	2
41	'CP010329.1.fasta',	2	2
42	'CP010330.1.fasta',	2	2
43	'CP010335.1.fasta',	2	2
44	'CP010336.1.fasta',	2	2
45	'CP010337.1.fasta',	2	2
46	'CP010338.1.fasta',	2	2
47	'CP010339.1.fasta',	2	2
48	'CP010340.1.fasta',	2	2
49	'CP010873.1.fasta',	2	2
50	'CP011510.1.fasta',	2	2
51	'CP013475.1.fasta',	2	2
52	'CP017593.1.fasta',	2	2
53	'CP017594.1.fasta',	2	2
54	'CP017595.1.fasta',	2	2
55	'CP017596.1.fasta',	2	2
56	'CP017597.1.fasta',	2	2
57	'CP017598.1.fasta',	2	2
58	'CP020381.2.fasta',	2	2
59	'CP022014.1.fasta',	2	2
60	'CP022704.2.fasta',	2	2
61	'JLBJ01000001.1.fasta',	2	2
62	'JLBS01000001.1.fasta',	2	2
63	'JLBS01000002.1.fasta',	2	2
64	'JLBS01000003.1.fasta',	2	2
65	'KK339435.1.fasta',	2	2
66	'KK339476.1.fasta',	2	2
67	'LHCK01000001.1.fasta',	2	2
68	'LHCK01000004.1.fasta',	2	2
69	'NC_017522.1.fasta',	2	2
70	'NC_017523.1.fasta',	2	2
71	'NC_021054.1.fasta',	2	2
72	'NZ_CP002871.1.fasta',	2	2
73	'NZ_CP009426.1.fasta',	2	2

74	'NZ_CP018304.1.fasta',	2	2
75	'NZ_CP018305.1.fasta',	2	2
76	'NZ_CP019610.1.fasta',	2	2
77	'NZ_CP019611.1.fasta',	2	2
78	'NZ_CP019612.1.fasta',	2	2
79	'NZ_CP019613.1.fasta',	2	2
80	'NZ_CP022577.1.fasta',	2	2
81	'NZ_CP022578.1.fasta',	2	2
82	'CP023169.1.fasta',	3	3
83	'CP023170.1.fasta',	3	3
84	'CP026742.1.fasta',	3	3
85	'CP028428.1.fasta',	3	3
86	'CP029065.1.fasta',	3	3
87	'CP029326.1.fasta',	3	3
88	'NC_017026.1.fasta',	3	3
89	'NC_017524.1.fasta',	3	3
90	'NC_017528.1.fasta',	3	3
91	'NC_021192.1.fasta',	3	3
92	'NC_021193.1.fasta',	3	3
93	'NC_021194.1.fasta',	3	3
94	'AL123456.3.fasta',	4	4
95	'AM412059.2.fasta',	4	4
96	'CM000788.2.fasta',	4	4
97	'CM000789.2.fasta',	4	4
98	'CP001658.1.fasta',	4	4
99	'CP001662.1.fasta',	4	4
100	'CP001664.1.fasta',	4	4
101	'CP001976.1.fasta',	4	4
102	'CP002095.1.fasta',	4	4
103	'CP007027.1.fasta',	4	4
104	'CP009243.1.fasta',	4	4
105	'CP012506.2.fasta',	4	4
106	'CP016794.1.fasta',	4	4
107	'CP016888.1.fasta',	4	4
108	'CP018778.1.fasta',	4	4
109	'CP025593.1.fasta',	4	4
110	'CP025594.1.fasta',	4	4
111	'CP025595.1.fasta',	4	4
112	'CP025596.1.fasta',	4	4
113	'CP025598.1.fasta',	4	4
114	'CP025599.1.fasta',	4	4
115	'CP025600.1.fasta',	4	4
116	'CP025601.1.fasta',	4	4
117	'CP025602.1.fasta',	4	4
118	'CP025603.1.fasta',	4	4
119	'CP025604.1.fasta',	4	4
120	'CP025605.1.fasta',	4	4
121	'CP025606.1.fasta',	4	4

122	'CP025607.1.fasta',	4	4
123	'CP025608.1.fasta',	4	4
124	'CP030093.1.fasta',	4	4
125	'KK339365.1.fasta',	4	4
126	'KK341218.1.fasta',	4	4
127	'KK341219.1.fasta',	4	4
128	'KK341220.1.fasta',	4	4
129	'LR027516.1.fasta',	4	4
130	'LWDQ01000001.1.fasta',	4	4
131	'LWDR01000001.1.fasta',	4	4
132	'NC_002755.1.fasta',	4	4
133	'NC_009525.1.fasta',	4	4
134	'NC_016768.1.fasta',	4	4
135	'NC_016934.1.fasta',	4	4
136	'NC_018143.2.fasta',	4	4
137	'NC_020089.1.fasta',	4	4
138	'NC_020559.1.fasta',	4	4
139	'NZ_CM000787.2.fasta',	4	4
140	'NZ_CM000788.2.fasta',	4	4
141	'NZ_CM000789.2.fasta',	4	4
142	'NZ_CM001225.1.fasta',	4	4
143	'NZ_CM001226.1.fasta',	4	4
144	'NZ_CM001227.1.fasta',	4	4
145	'NZ_CP016972.1.fasta',	4	4
146	'NZ_CP017920.1.fasta',	4	4
147	'NZ_CP018300.1.fasta',	4	4
148	'NZ_CP018301.1.fasta',	4	4
149	'NZ_CP018302.1.fasta',	4	4
150	'NZ_CP018303.1.fasta',	4	4
151	'NZ_CP023573.1.fasta',	4	4
152	'NZ_CP023574.1.fasta',	4	4
153	'NZ_CP023575.1.fasta',	4	4
154	'NZ_CP023576.1.fasta',	4	4
155	'NZ_CP023577.1.fasta',	4	4
156	'NZ_CP023578.1.fasta',	4	4
157	'NZ_CP023579.1.fasta',	4	4
158	'NZ_CP023580.1.fasta',	4	4
159	'NZ_CP023581.1.fasta',	4	4
160	'NZ_CP023582.1.fasta',	4	4
161	'NZ_CP023583.1.fasta',	4	4
162	'NZ_CP023584.1.fasta',	4	4
163	'NZ_CP023585.1.fasta',	4	4
164	'NZ_CP023586.1.fasta',	4	4
165	'NZ_CP023587.1.fasta',	4	4
166	'NZ_CP023588.1.fasta',	4	4
167	'NZ_CP023589.1.fasta',	4	4
168	'NZ_CP023590.1.fasta',	4	4
169	'NZ_CP023591.1.fasta',	4	4

170	'NZ_CP023592.1.fasta',	4	4
171	'NZ_CP023593.1.fasta',	4	4
172	'NZ_CP023594.1.fasta',	4	4
173	'NZ_CP023595.1.fasta',	4	4
174	'NZ_CP023596.1.fasta',	4	4
175	'NZ_CP023597.1.fasta',	4	4
176	'NZ_CP023598.1.fasta',	4	4
177	'NZ_CP023599.1.fasta',	4	4
178	'NZ_CP023600.1.fasta',	4	4
179	'NZ_CP023601.1.fasta',	4	4
180	'NZ_CP023602.1.fasta',	4	4
181	'NZ_CP023603.1.fasta',	4	4
182	'NZ_CP023604.1.fasta',	4	4
183	'NZ_CP023605.1.fasta',	4	4
184	'NZ_CP023606.1.fasta',	4	4
185	'NZ_CP023607.1.fasta',	4	4
186	'NZ_CP023608.1.fasta',	4	4
187	'NZ_CP023609.1.fasta',	4	4
188	'NZ_CP023610.1.fasta',	4	4
189	'NZ_CP023611.1.fasta',	4	4
190	'NZ_CP023612.1.fasta',	4	4
191	'NZ_CP023613.1.fasta',	4	4
192	'NZ_CP023614.1.fasta',	4	4
193	'NZ_CP023615.1.fasta',	4	4
194	'NZ_CP023616.1.fasta',	4	4
195	'NZ_CP023617.1.fasta',	4	4
196	'NZ_CP023618.1.fasta',	4	4
197	'NZ_CP023619.1.fasta',	4	4
198	'NZ_CP023620.1.fasta',	4	4
199	'NZ_CP023621.1.fasta',	4	4
200	'NZ_CP023622.1.fasta',	4	4
201	'NZ_CP023623.1.fasta',	4	4
202	'NZ_CP023624.1.fasta',	4	4
203	'NZ_CP023625.1.fasta',	4	4
204	'NZ_CP023626.1.fasta',	4	4
205	'NZ_CP023627.1.fasta',	4	4
206	'NZ_CP023628.1.fasta',	4	4
207	'NZ_CP023629.1.fasta',	4	4
208	'NZ_CP023630.1.fasta',	4	4
209	'NZ_CP023631.1.fasta',	4	4
210	'NZ_CP023632.1.fasta',	4	4
211	'NZ_CP023633.1.fasta',	4	4
212	'NZ_CP023634.1.fasta',	4	4
213	'NZ_CP023635.1.fasta',	4	4
214	'NZ_CP023636.1.fasta',	4	4
215	'NZ_CP023637.1.fasta',	4	4
216	'NZ_CP023638.1.fasta',	4	4
217	'NZ_CP023639.1.fasta',	4	4

218	'NZ_CP023640.1.fasta',	4	4
219	'NZ_CP030093.1.fasta',	4	4
220	'NZ_KK339370.1.fasta',	4	4
221	'NZ_KK339487.1.fasta',	4	4
222	'NZ_MNBY01000001.1.fasta',	4	4
223	'PDLG01000001.1.fasta',	4	4
224	'DF126614.1.fasta',	5	5
225	'NC_015758.1.fasta',	5	5
226	'NC_015848.1.fasta',	5	5
227	'NC_019950.1.fasta',	5	5
228	'NC_019951.1.fasta',	5	5
229	'NC_019952.1.fasta',	5	5
230	'NC_019965.1.fasta',	5	5
231	'NC_002945.4.fasta',	6	6
232	'NZ_GG663503.1.fasta',	6	6
233	'NZ_KK338483.1.fasta'	6	6

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil dan pembahasan, kesimpulan dari penelitian ini adalah sebagai berikut:

1. Nilai *k* dari k-Mer memiliki pengaruh terhadap akurasi. Jelas terlihat perbedaan ketika nilai *k* yang digunakan berbeda-beda dalam klasifikasi. Untuk metode yang digunakan ekstrasi dari substring DNA yang berupa string menjadi nilai numerik dapat digunakan TF-IDF ataupun *countvectorizer* karena memberikan hasil yang tidak berbeda.
2. PCA memiliki pengaruh terhadap waktu komputasi dari klasifikasi karena dimensi data yang berkurang. Banyaknya varian data dari PCA juga mempengaruhi hasil akurasi dan banyaknya data yang digunakan. Terbukti juga dengan mengambil sebagian varian dari data dapat menaikkan akurasi. LDA sebagai pembanding dari PCA telah dapat bekerja lebih baik. LDA selain dapat mengurangi dimensi dari data juga dapat meningkatkan performa evaluasi dari klasifikasinya karena LDA bersifat *supervised learning*.
3. Perubahan parameter *C* dan *gamma* dari SVM sangat mempengaruhi performa dari SVM. Dengan menambah atau mengurangi nilai dari parameter *C* dan *gamma* menghasilkan perubahan akurasi SVM. Perubahan parameter tidak selalu memberikan akurasi yang baik dan susah untuk menentukan parameter yang tepatnya.
4. Setelah dilakukan optimasi dengan menggunakan VNS, akurasi tertinggi SVM mencapai 0.995708 dan akurasi sebelum dilakukannya optimasi sebesar 0.927039. Hal ini menunjukkan perbaikan akurasi dengan peningkatan sebanyak 0.068669.
5. *Nested* RVNS adalah modifikasi VNS yang dilakukan pada penelitian ini. Hasil dari *nested* RVNS ini memiliki performa yang lebih bagus dari VNS dasar. Dengan *nested* RVNS, akurasi yang dicapai adalah 0.995708 sama besar nilainya dengan VNS dasar. Keunggulan dari *nested* RVNS adalah dari sisi waktu komputasi, dengan *nested* RVNS dapat mendapatkan solusi optimalnya dengan waktu 2.5 kali lebih cepat dari VNS dasar.

6.2 Saran

Adapun saran untuk pengembangan penelitian ini kedepannya sebagai berikut.

1. Membuat perbaikan pada metode k-Mer supaya menjadi lebih cepat dalam ekstraksi fitur saat menggunakan nilai *k* yang tinggi.

2. Perlu dilakukan improvisasi metode VNS supaya dapat meningkatkan performanya menjadi lebih efektif untuk digunakan.

DAFTAR PUSTAKA

Abubakar, I., Cohen, T., Jackson, C., Rangaka, M., Menzies, N.A., Hill, A.N., Cohen, T. and Salomon, J.A., 2018. The impact of migration on tuberculosis in the United States. *The International Journal of Tuberculosis and Lung Disease*, [online] 22(July), pp.1392–1403. Available at: <http://dx.doi.org/10.5588/ijtld.17.0185>.

Amous, M., Toumi, S., Jarboui, B. and Eddaly, M., 2017. A variable neighborhood search algorithm for the capacitated vehicle routing problem. *Electronic Notes in Discrete Mathematics*, [online] 58, pp.231–238. Available at: <http://dx.doi.org/10.1016/j.endm.2017.03.030>.

Antonino, M. and Gangi, D., 2017. Deep Learning Architectures for DNA Sequence Classification. [online] 10147, pp.162–171. Available at: <http://link.springer.com/10.1007/978-3-319-52962-2>.

Apostolidis-afentoulis, V. and Box, P.O., 2015. SVM CLASSIFICATION WITH LINEAR AND RBF KERNELS. [online] (1), pp.1–7. Available at: <http://doi.org/10.13140/RG.2.1.3351.4083>.

Ashlock, W. and Datta, S., 2013. Evolved features for DNA sequence classification and their fitness landscapes. *IEEE Transactions on Evolutionary Computation*, [online] 17(2), pp.185–197. Available at: <https://doi.org/10.1109/TEVC.2012.2207120>.

Asia, S., Paci, W., Congress, I., Evolution, T. and Meeting, T.B.E., 2015. Tuberculosis in evolution. [online] (April), pp.3–5. Available at: <http://dx.doi.org/10.1016/j.tube.2015.04.007>.

Bai, Q., 2015. An Improved Hybrid Algorithm for Traveling Salesman Problem. [online] (Bmei), pp.806–809. Available at: <https://doi.org/10.1109/BMEI.2015.7401613>.

Belhaiza, S., M'Hallah, R. and Brahim, G. Ben, 2017. A new Hybrid Genetic Variable Neighborhood search heuristic for the Vehicle Routing Problem with Multiple Time Windows. *2017 IEEE Congress on Evolutionary Computation (CEC)*, [online] pp.1319–1326. Available at: <http://ieeexplore.ieee.org/document/7969457/>.

Bobak, C.A., Titus, A.J. and Hill, J.E., 2019. Comparison of common machine learning models for classification of tuberculosis using transcriptional biomarkers from integrated datasets ☆. *Applied Soft Computing Journal*, [online] 74, pp.264–273. Available at: <https://doi.org/10.1016/j.asoc.2018.10.005>.

Boser, Bernhard E. and Guyon, Isabelle M. and Vapnik, V.N., 1992. Training Algorithm Margin for Optimal Classifiers. *COLT '92 Proceedings of the fifth annual workshop on Computational learning theory*, [online] pp.144–152. Available at: <https://doi.org/10.1145/130385.130401>.

Boughaci, D. and Alkhawaldeh, A.A., 2018. Three local search-based methods for feature selection in credit scoring. *Vietnam Journal of Computer Science*, [online] 5(2), pp.107–121. Available at: <https://doi.org/10.1007/s40595-018-0107-y>.

Caporossi, G., 2005. Arbitrary-norm Separation by Variable Neighborhood Search. *IMA Journal of Management Mathematics*. [online] Available at: <https://doi.org/10.1093/imaman/dpm014>.

Chaudhary, P., Khan, A.Q., Lal, R. and Bhadana, U., 2018. Gastric tuberculosis. *Indian*

Journal of Tuberculosis. [online] Available at: <http://www.sciencedirect.com/science/article/pii/S0019570718303263>.

Chengli, F.A.N., Qiang, F.U., Guangzheng, L. and Qinghua, X., 2018. Hybrid artificial bee colony algorithm with variable neighborhood search and memory mechanism. *Journal of Systems Engineering and Electronics*, [online] 29(2), pp.405–414. Available at: <https://doi.org/10.21629/JSEE.2018.02.20>.

Cortes, C. and Vapnik, V., 1995. Support-Vector Networks. *Machine Learning*, [online] 20, pp.273–297. Available at: <https://doi.org/10.1023/A:1022627411411>.

Demidova, L., Nikulchev, E. and Sokolova, Y., 2016. Big Data Classification Using the SVM Classifiers with the Modified Particle Swarm Optimization and the SVM Ensembles. *International Journal of Advanced Computer Science and Applications*, [online] 7(5), pp.294–312. Available at: <http://dx.doi.org/10.14569/IJACSA.2016.070541>.

Dinç, İ., Sigdel, M., Dinç, S., Sigdel, M.S., Pusey, M.L. and Aygün, R.S., 2014. Evaluation of Normalization and PCA on the Performance of Classifiers for Protein Crystallization Images. *IEEE SOUTHEASTCON 2014*, [online] pp.0–5. Available at: <https://doi.org/10.1109/SECON.2014.6950744>.

Ding, J., Wang, J., Kang, X. and Hu, X.H., 2017. Building an SVM Classifier for Automated Selection of Big Data. *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, [online] pp.15–22. Available at: <https://doi.org/10.1109/BigDataCongress.2017.12>.

Dubinkina, V.B., Ischenko, D.S., Ulyantsev, V.I., Tyakht, A. V and Alexeev, D.G., 2016. Assessment of k-mer spectrum applicability for metagenomic dissimilarity analysis. *BMC Bioinformatics*, [online] pp.1–11. Available at: <http://dx.doi.org/10.1186/s12859-015-0875-7>.

Emmanuel, W R Sam; Mithra, K.S., 2017. An Efficient Approach to Sputum Image Segmentation using Improved Fuzzy Local Tuberculosis Diagnosis. *2017 International Conference on Inventive Computing and Informatics (ICICI)*, [online] (Icici), pp.126–130. Available at: <https://doi.org/10.1109/ICICI.2017.8365321>.

Évora, L.H.R.A., Seixas, J.M., 2015. Artificial Neural Network Models for Diagnosis Support of Drug and Multidrug Resistant Tuberculosis. *Latin America Congress on Computational Intelligence (LA-CCI)*, [online] pp.1–5. Available at: <https://doi.org/10.1109/LA-CCI.2015.7435954>.

Ferreira, H.S., Bogue, E.T., Noronha, T.F., Belhaiza, S. and Prins, C., 2018. Variable Neighborhood Search for Vehicle Routing Problem with Multiple Time Windows. *Electronic Notes in Discrete Mathematics*, [online] 66, pp.207–214. Available at: <https://doi.org/10.1016/j.endm.2018.03.027>.

Fiannaca, A., La Paglia, L., La Rosa, M., Lo Bosco, G., Renda, G., Rizzo, R., Gaglio, S. and Urso, A., 2018. Deep learning models for bacteria taxonomic classification of metagenomic data. *BMC Bioinformatics*, [online] 19(Suppl 7). Available at: <https://doi.org/10.1186/s12859-018-2182-6>.

Fogel, N., 2015. Tuberculosis : A disease without boundaries. *Tuberculosis*, [online] 95(5), pp.527–531. Available at: <http://dx.doi.org/10.1016/j.tube.2015.05.017>.

Gajera, V., Gupta, R., Jana, P.K. and Member, I.S., 2016. An Effective Multi-Objective Task Scheduling Algorithm using Min-Max Normalization in Cloud Computing. *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, [online] pp.812–816. Available at: <https://doi.org/10.1109/ICATCCT.2016.7912111>.

Grama, L., Tuns, L., Rusu, C., Mel, A.L. and Cepstral, F., 2017. On the Optimization of SVM Kernel Parameters for Improving Audio Classification Accuracy. *2017 14th International Conference on Engineering of Modern Electric Systems (EMES)*, [online] pp.224–227. Available at: <https://doi.org/10.1109/EMES.2017.7980420>.

Han, G. and Cho, D., 2018. Genomics Genome classification improvements based on k-mer intervals in sequences. *Genomics*, [online] (October), pp.0–1. Available at: <https://doi.org/10.1016/j.ygeno.2018.11.001>.

Hansen, P., Mladenović, N. and Moreno Pérez, J.A., 2010. Variable neighbourhood search: Methods and applications. *Annals of Operations Research*, [online] 175(1), pp.367–407. Available at: <https://doi.org/10.1007/s10479-009-0657-6>.

Hansen, P., Mladenović, N., Todosijević, R. and Hanafi, S., 2017. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, [online] 5(3), pp.423–454. Available at: <https://doi.org/10.1007/s13675-016-0075-x>.

Hatamian, M., Serna, J. and Rannenberg, K., 2019. Revealing the unrevealed : Mining smartphone users privacy perception on app markets. *Computers & Security*, [online] 83(675730), pp.332–353. Available at: <https://doi.org/10.1016/j.cose.2019.02.010>.

He, Z., He, Y. and Wei, Y., 2017. Big data oriented root cause identification approach based on PCA and SVM for product infant failure. *Proceedings of 2016 Prognostics and System Health Management Conference, PHM-Chengdu 2016*, pp.1–5.

Hore, S., Chatterjee, A. and Dewanji, A., 2018. Improving variable neighborhood search to solve the traveling salesman problem. *Applied Soft Computing Journal*, [online] 68, pp.83–91. Available at: <https://doi.org/10.1016/j.asoc.2018.03.048>.

Huang, H. and Wang, Z., 2017. Efficient Parameter Selection for SVM : The Case of Business Intelligence Categorization. *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, [online] pp.158–160. Available at: <https://doi.org/10.1109/ISI.2017.8004897>.

Ilias, S., Tahir, N., Jailani, R. and Alam, S., 2016. Feature Extraction of Autism Gait Data Using Principal Component Analysis and Linear Discriminant Analysis. *2016 IEEE Industrial Electronics and Applications Conference (IEACon)*, [online] pp.275–279. Available at: <https://doi.org/10.1109/IEACON.2016.8067391>.

Iqbal, M.J., Faye, I., Said, A.M. and Samir, B.B., 2013. A distance-based feature-encoding technique for protein sequence classification in bioinformatics. *Proceeding - IEEE CYBERNETICSCOM 2013: IEEE International Conference on Computational Intelligence and Cybernetics*, [online] pp.1–5. Available at: <https://doi.org/10.1109/CyberneticsCom.2013.6865770>.

Iqbal, M.J., Faye, I., Samir, B.B. and Md Said, A., 2014. Efficient feature selection and classification of protein sequence data in bioinformatics. *The Scientific World Journal*, [online] 2014. Available at: <http://dx.doi.org/10.1155/2014/173869>.

Jimenez-marquez, J.L., Gonzalez-carrasco, I., Lopez-cuadrado, J.L. and Ruiz-mezcua, B., 2019. Towards a big data framework for analyzing social media content. *International Journal of Information Management*, [online] 44(September 2018), pp.1–12. Available at: <https://doi.org/10.1016/j.ijinfomgt.2018.09.003>.

Jin, Z., Chaorong, W., Chengguang, H. and Feng, W., 2014. Parameter optimization algorithm of SVM for fault classification in traction converter. *The 26th Chinese Control and Decision Conference (2014 CCDC)*, [online] pp.3786–3791. Available at: <https://doi.org/10.1109/CCDC.2014.6852839>.

Kocatürk, F. and Özpeynirci, Ö., 2014. Variable neighborhood search for the pharmacy duty scheduling problem. *Computers and Operations Research*, [online] 51, pp.218–226. Available at: <http://dx.doi.org/10.1016/j.cor.2014.06.001>.

Kristensen, T. and Guillaume, F., 2015. Different regimes for classification of DNA sequences. *Proceedings of the 2015 7th IEEE International Conference on Cybernetics and Intelligent Systems, CIS 2015 and Robotics, Automation and Mechatronics, RAM 2015*, [online] 7, pp.114–119. Available at: <https://doi.org/10.1109/ICCIS.2015.7274558>.

Lameiro, C. and Schreier, P.J., 2016. Cross-validation techniques for determining the number of correlated components between two data sets when the number of samples is very small. *2016 50th Asilomar Conference on Signals, Systems and Computers*, pp.601–605.

Lauren, S., 2014. Stock Trend Prediction Using Simple Moving Average Supported by News Classification. *2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*, [online] (1), pp.135–139. Available at: <https://doi.org/10.1109/ICAICTA.2014.7005929>.

Li, L., Huang, H. and Jin, X., 2018. AE-CNN Classification of Pulmonary Tuberculosis Based on CT images. *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, [online] pp.39–42. Available at: <https://doi.org/10.1109/ITME.2018.00020>.

Li, Y., Lv, Y., Li, X., Xiao, W. and Li, C., 2017. Sequence comparison and essential gene identification with new inter-nucleotide distance sequences. *Journal of Theoretical Biology*, [online] 418(October 2016), pp.84–93. Available at: <http://dx.doi.org/10.1016/j.jtbi.2017.01.031>.

Liu, C., 2015. Soft Sensing Technology Based on PCA and SVM for Coal Powder Amount in Medium Speed Mill. *2015 Chinese Automation Congress (CAC)*, [online] pp.2039–2042. Available at: <https://doi.org/10.1109/CAC.2015.7382839>.

Liu, Y. and Du, J., 2016. Parameter Optimization of the SVM for Big Data. *Proceedings - 2015 8th International Symposium on Computational Intelligence and Design, ISCID 2015*, 2(1), pp.341–344.

Liu, Y., He, Y. and Cui, W., 2018. An improved SVM classifier based on multi-verse optimizer for fault diagnosis of autopilot. *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, [online] (Iaeac), pp.941–944. Available at: <https://doi.org/10.1109/IAEAC.2018.8577808>.

Ma, X., Guo, J., Liu, H. De, Xie, J.M. and Sun, X., 2012. Sequence-based prediction of DNA-binding residues in proteins with conservation and correlation information. *IEEE/ACM*

Transactions on Computational Biology and Bioinformatics, [online] 9(6), pp.1766–1775. Available at: <https://doi.org/10.1109/TCBB.2012.106>.

Mamatjan, Y., Agnihotri, S., Goldenberg, A., Tonge, P., Mansouri, S. and Zadeh, G., 2017. Molecular Signatures for Tumor Classification An Analysis of The Cancer Genome Atlas Data. *The Journal Of Molecular Diagnostics*, [online] 19(6). Available at: <https://doi.org/10.1016/j.jmoldx.2017.07.008>.

Meng, X., Li, J., Dai, X. and Dou, J., 2018. Variable Neighborhood Search for a Colored Traveling Salesman Problem. *IEEE Transactions on Intelligent Transportation Systems*, [online] 19(4), pp.1018–1026. Available at: <https://doi.org/10.1109/TITS.2017.2706720>.

Mladenović, N. and Hansen, P., 1997. Variable neighborhood search. *Computers & Operations Research*, [online] 24(11), pp.1097–1100. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0305054897000312>.

Morales, D.R., 2014. A nested heuristic for parameter tuning in Support Vector Machines. [online] Available at: <https://doi.org/10.1016/j.cor.2013.10.002>.

Nakano, F.K., Mastelini, S.M., Barbon, S. and Cerri, R., 2018. Improving Hierarchical Classification of Transposable Elements using Deep Neural Networks. *Proceedings of the International Joint Conference on Neural Networks*, [online] 2018-July, pp.1–8. Available at: <https://doi.org/10.1109/IJCNN.2018.8489461>.

Nguyen, N.G., Tran, V.A., Ngo, D.L., Phan, D., Lumbanraja, F.R., Faisal, M.R., Abapihi, B., Kubo, M. and Satou, K., 2016. DNA Sequence Classification by Convolutional Neural Network. *Journal of Biomedical Science and Engineering*, [online] 09(05), pp.280–286. Available at: <http://www.scirp.org/journal/doi.aspx?DOI=10.4236/jbise.2016.95021>.

Niehaus, K.E., Walker, T.M., Crook, D.W., Peto, T.E.A. and Clifton, D.A., 2014. Machine learning for the prediction of antibacterial susceptibility in Mycobacterium tuberculosis. [online] pp.618–621. Available at: <https://doi.org/10.1109/BHI.2014.6864440>.

Nuraisha, S., 2018. Evaluation of Normalization in Fake Fingerprint Detection with Heterogeneous Sensor. *2018 International Seminar on Application for Technology of Information and Communication*, [online] pp.83–86. Available at: <https://doi.org/10.1109/ISEMANTIC.2018.8549742>.

Omurca, S.I. and Ekinci, E., 2015. An alternative evaluation of post traumatic stress disorder with machine learning methods. *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, [online] (MI), pp.1–7. Available at: <http://ieeexplore.ieee.org/document/7276754/>.

Pan, L. and Tang, H., 2015. An Identification Method of News Scientific Intelligence Based on TF-IDF. *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, [online] pp.501–504. Available at: <https://doi.org/10.1109/DCABES.2015.131>.

Parmar, P.S., 2018. Multiclass Text Classification and Analytics for Improving Customer Support Response through different Classifiers. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, [online] pp.538–542. Available at: <https://doi.org/10.1109/ICACCI.2018.8554881>.

Patro, S.G.K. and Kumar, K., 2015. Normalization : A Preprocessing Stage. *International Advanced Research Journal in Science, Engineering and Technology*, [online] 2(3), pp.2393–2395. Available at: <https://doi.org/10.17148/IARJSET.2015.2305>.

Phan, D., Nguyen, N.G., Lumbanraja, F.R. and Faisal, M.R., 2017. Combined Use of k-Mer Numerical Features and Position-Specific Categorical Features in Fixed-Length DNA Sequence Classification. *Journal of Biomedical Science and Engineering*, [online] 10(8), pp.390–401. Available at: <https://doi.org/10.4236/jbise.2017.108030>.

Rizanti, N.A., Setyaningrum, A.H., Si, M. and Iris, C., 2016. Colon Detection Using Principal Component Analysis (PCA) and Support Vector Machine (SVM). *2016 4th International Conference on Cyber and IT Service Management*, [online] pp.1–7. Available at: <https://doi.org/10.1109/CITSM.2016.7577526>.

Satpute, B. and Yadav, R., 2018. Machine Intelligence Techniques for Protein Classification. *2018 3rd International Conference for Convergence in Technology (I2CT)*, [online] pp.1–4. Available at: <https://doi.org/10.1109/I2CT.2018.8529495>.

Sinha, R., 2018. Breast tuberculosis. *Indian Journal of Tuberculosis*, [online] pp.6–11. Available at: <https://doi.org/10.1016/j.ijtb.2018.07.003>.

Surya, A., Setyaningsih, B., Nasution, H.S., Parwati, C.G., Yuzwar, Y.E., Osberg, M., Hanson, C.L., Hymoff, A., Mingkwan, P., Makayova, J., Gebhard, A. and Waworuntu, W., 2017. Quality Tuberculosis Care in Indonesia : Using Patient Pathway Analysis to Optimize Public – Private Collaboration. *The Journal of Infectious Diseases*, [online] 216. Available at: <https://doi.org/10.1093/infdis/jix379>.

Thomas, B.W. and Manni, E., 2014. Scheduled penalty Variable Neighborhood Search. *Computers and Operations Research*, [online] 52, pp.170–180. Available at: <http://dx.doi.org/10.1016/j.cor.2013.12.004>.

Tripathy, A., Agrawal, A. and Rath, S.K., 2015. Classification of Sentimental Reviews Using Machine Learning Techniques. *International Conference on Recent Trends in Computing 2015*, [online] 57, pp.821–829. Available at: <http://dx.doi.org/10.1016/j.procs.2015.07.523>.

Tripathy, A., Agrawal, A. and Rath, S.K., 2016. Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems With Applications*, [online] 57, pp.117–126. Available at: <http://dx.doi.org/10.1016/j.eswa.2016.03.028>.

Vahdani, B., Mousavi, S.M., Tavakkoli-Moghaddam, R. and Hashemi, H., 2017. A New Enhanced Support Vector Model Based On General Variable Neighborhood Search Algorithm For Supplier Performance Evaluation: A Case Study. *International Journal of Computational Intelligence Systems*, [online] 10(1), pp.293–311. Available at: <https://doi.org/10.2991/ijcis.2017.10.1.20>.

Wang, T., Herbster, M. and Mian, I.S., 2018. Virus genome sequence classification using features based on nucleotides, words and compression. [online] pp.1–36. Available at: <http://arxiv.org/abs/1809.03950>.

Wang, Y. and Chen, Y., 2017. A New Feature Extraction Algorithm Based on Fisher Linear Discriminant Analysis. *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, (1), pp.414–417.

Wassan, J.T., Wang, H. and Zheng, H., 2019. Machine Learning in Bioinformatics. *Encyclopedia of Bioinformatics and Computational Biology*, [online] pp.300–308. Available at: <https://linkinghub.elsevier.com/retrieve/pii/B9780128096338203312>.

Wong, T. and Yang, N., 2017. Dependency Analysis of Accuracy Estimates in k-fold Cross Validation. 4347(c), pp.1–12.

Xie, Y., 2015. A Fault Diagnosis Approach Using SVM with Data Dimension Reduction by PCA and LDA Method. *2015 Chinese Automation Congress (CAC)*, [online] pp.869–874. Available at: <https://doi.org/10.1109/CAC.2015.7382620>.

Yimer, S.A., Norheim, G., Namouchi, A., Zegeye, E.D., Kinander, W. and Tønjum, T., 2015. Mycobacterium tuberculosis Lineage 7 Strains Are Associated with Prolonged Patient Delay in Seeking Treatment for Pulmonary Tuberculosis in Amhara Region , Ethiopia. *Journal of Clinical Microbiology*, [online] 53(4), pp.1301–1309. Available at: <https://doi.org/10.1128/JCM.03566-14>.

Zhan, Y., Li, B., Huo, Y., Lin, A. and Wu, H., 2018. A case of multiple organ tuberculosis. *Radiology of Infectious Diseases*, [online] pp.0–4. Available at: <https://doi.org/10.1016/j.jrid.2018.02.003>.

Zhang, T. and Yang, B., 2016. Big Data Dimension Reduction Using PCA. *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, [online] pp.152–157. Available at: <http://ieeexplore.ieee.org/document/7796166/>.

Zhongwen, Z. and Huanghuang, G., 2017. Visualization Study of High-Dimensional Data Classification Based on PCA-SVM. *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, [online] pp.346–349. Available at: <http://ieeexplore.ieee.org/document/8005497/>.

Zucco, C., 2019. Data Mining in Bioinformatics. *Encyclopedia of Bioinformatics and Computational Biology*, [online] pp.328–335. Available at: <https://www.sciencedirect.com/science/article/pii/B9780128096338203816>.

LAMPIRAN

Berikut adalah tabel alamat tautan dimana dataset didapatkan beserta kelasnya :

Tabel 6. 1 Tabel link dataset

Nama	Alamat Tautan	Kelas
AP018033.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AP018032.1	Indo Oceanic
AP018034.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AP018034.1	Indo Oceanic
AP018035.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AP018035.1	Indo Oceanic
AP018036.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AP018036.1	Indo Oceanic
CP010895.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010895.1	Indo Oceanic
NZ_CP009427.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP009427.1	Indo Oceanic
NZ_CP010968.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP010968.1	Indo Oceanic
AP010918.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AP010918.1	East Asia
AP014573.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AP014573.1	East Asia
AP017901.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AP017901.1	East Asia
CP007809.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP007809.1	East Asia
CP008959.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008959.1	East Asia
CP008960.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008960.1	East Asia
CP008961.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008961.1	East Asia
CP008962.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008962.1	East Asia
CP008963.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008963.1	East Asia
CP008964.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008964.1	East Asia
CP008965.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008965.1	East Asia
CP008966.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008966.1	East Asia
CP008967.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008967.1	East Asia
CP008968.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008968.1	East Asia

CP008969.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008969.1	East Asia
CP008970.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008970.1	East Asia
CP008971.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008971.1	East Asia
CP008972.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008972.1	East Asia
CP008973.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008973.1	East Asia
CP008974.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008974.1	East Asia
CP008975.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008975.1	East Asia
CP008976.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008976.1	East Asia
CP008977.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008977.1	East Asia
CP008978.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008978.1	East Asia
CP008979.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008979.1	East Asia
CP008980.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008980.1	East Asia
CP008981.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008981.1	East Asia
CP008982.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008982.1	East Asia
CP008983.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP008983.1	East Asia
CP009100.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP009100.1	East Asia
CP009101.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP009101.1	East Asia
CP009426.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP009426.1	East Asia
CP009427.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP009427.1	East Asia
CP010329.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010329.1	East Asia
CP010330.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010330.1	East Asia
CP010335.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010335.1	East Asia
CP010336.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010336.1	East Asia
CP010337.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010337.1	East Asia
CP010338.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010338.1	East Asia

CP010339.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010339.1	East Asia
CP010340.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010340.1	East Asia
CP010873.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP010873.1	East Asia
CP011510.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP011510.1	East Asia
CP013475.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP013475.1	East Asia
CP017593.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP017593.1	East Asia
CP017594.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP017594.1	East Asia
CP017595.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP017595.1	East Asia
CP017596.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP017596.1	East Asia
CP017597.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP017597.1	East Asia
CP017598.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP017598.1	East Asia
CP020381.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP020381.2	East Asia
CP022014.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP022014.1	East Asia
CP022704.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP022704.2	East Asia
JLBJ01000001.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/JLBJ01000001.1	East Asia
JLBS01000001.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/JLBS01000001.1	East Asia
JLBS01000002.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/JLBS01000002.1	East Asia
JLBS01000003.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/JLBS01000003.1	East Asia
KK339435.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/KK339435.1	East Asia
KK339476.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/KK339476.1	East Asia
LHCK01000001.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/LHCK01000001.1	East Asia
LHCK01000004.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/LHCK01000004.1	East Asia
NC_017522.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_017522.1	East Asia
NC_017523.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_017523.1	East Asia
NC_021054.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_021054.1	East Asia

NZ_CP002871.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP002871.1	East Asia
NZ_CP009426.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP009426.1	East Asia
NZ_CP018304.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP018304.1	East Asia
NZ_CP018305.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP018305.1	East Asia
NZ_CP019610.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP019610.1	East Asia
NZ_CP019611.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP019611.1	East Asia
NZ_CP019612.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP019612.1	East Asia
NZ_CP019613.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP019613.1	East Asia
NZ_CP022577.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP022577.1	East Asia
NZ_CP022578.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP022578.1	East Asia
CP023169.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP023169.1	Central Asia
CP023170.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP023170.1	Central Asia
CP026742.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP026742.1	Central Asia
CP028428.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP028428.1	Central Asia
CP029065.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP029065.1	Central Asia
CP029326.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP029326.1	Central Asia
NC_017026.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_017026.1	Central Asia
NC_017524.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_017524.1	Central Asia
NC_017528.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_017528.1	Central Asia
NC_021192.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_021192.1	Central Asia
NC_021193.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_021193.1	Central Asia
NC_021194.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_021194.1	Central Asia
AL123456.3.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AL123456.3	Euro America
AM412059.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/AM412059.2	Euro America
CM000788.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CM000788.2	Euro America

CM000789.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CM000789.2	Europe America
CP001658.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP001658.1	Europe America
CP001662.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP001662.1	Europe America
CP001664.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP001664.1	Europe America
CP001976.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP001976.1	Europe America
CP002095.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP002095.1	Europe America
CP007027.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP007027.1	Europe America
CP009243.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP009243.1	Europe America
CP012506.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP012506.2	Europe America
CP016794.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP016794.1	Europe America
CP016888.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP016888.1	Europe America
CP018778.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP018778.1	Europe America
CP025593.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025593.1	Europe America
CP025594.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025594.1	Europe America
CP025595.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025595.1	Europe America
CP025596.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025596.1	Europe America
CP025598.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025598.1	Europe America
CP025599.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025599.1	Europe America
CP025600.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025600.1	Europe America
CP025601.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025601.1	Europe America
CP025602.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025602.1	Europe America
CP025603.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025603.1	Europe America
CP025604.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025604.1	Europe America
CP025605.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025605.1	Europe America
CP025606.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025606.1	Europe America

CP025607.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025607.1	Euro America
CP025608.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP025608.1	Euro America
CP030093.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/CP030093.1	Euro America
KK339365.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/KK339365.1	Euro America
KK341218.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/KK341218.1	Euro America
KK341219.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/KK341219.1	Euro America
KK341220.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/KK341220.1	Euro America
LR027516.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/LR027516.1	Euro America
LWDQ01000001.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/LWDQ01000001.1	Euro America
LWDR01000001.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/LWDR01000001.1	Euro America
NC_002755.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_002755.1	Euro America
NC_009525.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_009525.1	Euro America
NC_016768.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_016768.1	Euro America
NC_016934.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_016934.1	Euro America
NC_018143.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_018143.2	Euro America
NC_020089.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_020089.1	Euro America
NC_020559.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_020559.1	Euro America
NZ_CM000787.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CM000787.2	Euro America
NZ_CM000788.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CM000788.2	Euro America
NZ_CM000789.2.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CM000789.2	Euro America
NZ_CM001225.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CM001225.1	Euro America
NZ_CM001226.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CM001226.1	Euro America
NZ_CM001227.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CM001227.1	Euro America
NZ_CP016972.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP016972.1	Euro America
NZ_CP017920.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP017920.1	Euro America

		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP018300.1.fasta	18300.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP018301.1.fasta	18301.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP018302.1.fasta	18302.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP018303.1.fasta	18303.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023573.1.fasta	23573.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023574.1.fasta	23574.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023575.1.fasta	23575.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023576.1.fasta	23576.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023577.1.fasta	23577.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023578.1.fasta	23578.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023579.1.fasta	23579.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023580.1.fasta	23580.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023581.1.fasta	23581.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023582.1.fasta	23582.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023583.1.fasta	23583.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023584.1.fasta	23584.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023585.1.fasta	23585.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023586.1.fasta	23586.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023587.1.fasta	23587.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023588.1.fasta	23588.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023589.1.fasta	23589.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023590.1.fasta	23590.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023591.1.fasta	23591.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023592.1.fasta	23592.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023593.1.fasta	23593.1		America

		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023594.1.fasta	23594.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023595.1.fasta	23595.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023596.1.fasta	23596.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023597.1.fasta	23597.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023598.1.fasta	23598.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023599.1.fasta	23599.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023600.1.fasta	23600.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023601.1.fasta	23601.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023602.1.fasta	23602.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023603.1.fasta	23603.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023604.1.fasta	23604.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023605.1.fasta	23605.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023606.1.fasta	23606.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023607.1.fasta	23607.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023608.1.fasta	23608.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023609.1.fasta	23609.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023610.1.fasta	23610.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023611.1.fasta	23611.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023612.1.fasta	23612.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023613.1.fasta	23613.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023614.1.fasta	23614.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023615.1.fasta	23615.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023616.1.fasta	23616.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023617.1.fasta	23617.1		America
		https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro
NZ_CP023618.1.fasta	23618.1		America

	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023619.1.fasta	23619.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023620.1.fasta	23620.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023621.1.fasta	23621.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023622.1.fasta	23622.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023623.1.fasta	23623.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023624.1.fasta	23624.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023625.1.fasta	23625.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023626.1.fasta	23626.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023627.1.fasta	23627.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023628.1.fasta	23628.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023629.1.fasta	23629.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023630.1.fasta	23630.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023631.1.fasta	23631.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023632.1.fasta	23632.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023633.1.fasta	23633.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023634.1.fasta	23634.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023635.1.fasta	23635.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023636.1.fasta	23636.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023637.1.fasta	23637.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023638.1.fasta	23638.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023639.1.fasta	23639.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP023640.1.fasta	23640.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_CP0	Euro	
NZ_CP030093.1.fasta	30093.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_KK3	Euro	
NZ_KK339370.1.fasta	39370.1	America	
	https://www.ncbi.nlm.nih.gov/nuccore/NZ_KK3	Euro	
NZ_KK339487.1.fasta	39487.1	America	

NZ_MNBY01000001.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_MNBY01000001.1	Euro America
PDLG01000001.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/PDLG01000001.1	Euro America
DF126614.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/DF126614.1	West African1
NC_015758.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_015758.1	West African1
NC_015848.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_015848.1	West African1
NC_019950.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_019950.1	West African1
NC_019951.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_019951.1	West African1
NC_019952.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_019952.1	West African1
NC_019965.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_019965.1	West African1
NC_002945.4.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NC_002945.4	West African2
NZ_GG663503.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_GG663503.1	West African2
NZ_KK338483.1.fasta	https://www.ncbi.nlm.nih.gov/nuccore/NZ_KK338483.1	West African2