



UNIVERSITAS INDONESIA

**KLASIFIKASI DOMAIN SPESIALISASI DOKTER PADA DATA TEKS FORUM
TANYA JAWAB KESEHATAN**

SKRIPSI

HENDRICO KRISTIawan

1906350912

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JUNI 2023**



UNIVERSITAS INDONESIA

**KLASIFIKASI DOMAIN SPESIALISASI DOKTER PADA DATA TEKS FORUM
TANYA JAWAB KESEHATAN**

SKRIPSI

**Diajukan sebagai salah satu syarat untuk memperoleh gelar
Sarjana Ilmu Komputer**

HENDRICO KRISTIawan

1906350912

**FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JUNI 2023**

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Hendrico Kristiawan

NPM : 1906350912

Tanda Tangan : 

Tanggal : 15 Juni 2023

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan oleh :

Nama : Hendrico Kristiawan
NPM : 1906350912
Program Studi : Sarjana Ilmu Komputer
Judul : Klasifikasi Domain Spesialisasi Dokter pada Data Teks Forum Tanya Jawab Kesehatan

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Ilmu Komputer pada Program Studi Sarjana Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia

DEWAN PENGUJI

Pembimbing	: Alfan Farizki Wicaksono, S.T., M.Sc., Ph.D.	(Nilai telah diberikan melalui SISIDANG pada 24-07-2023, 09:09:16) (Revisi telah disetujui melalui SISIDANG pada 24-07-2023, 09:09:38)
Pembimbing	: Rahmad Mahendra S.Kom., M.Sc.	(Nilai telah diberikan melalui SISIDANG pada 05-07-2023, 17:57:42) (Revisi telah disetujui melalui SISIDANG pada 22-07-2023, 13:17:01)
Penguji	: Dr. Ika Alfina, S.Kom., M.Kom.	(Nilai telah diberikan melalui SISIDANG pada 05-07-2023, 13:34:31) (Revisi telah disetujui melalui SISIDANG pada 17-07-2023, 08:57:52)
Penguji	: Dr. Kurniawati Azizah, M.Phil.	(Nilai telah diberikan melalui SISIDANG pada 10-07-2023, 22:26:40) (Revisi telah disetujui melalui SISIDANG pada 21-07-2023, 22:14:52)

Ditetapkan di : Depok, Jawa Barat

Tanggal : 24 Juli 2023

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala hikmat dan karunia-Nya yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan tugas akhir berjudul "Klasifikasi Domain Spesialisasi Dokter pada Data Teks Forum Tanya Jawab Kesehatan". Tugas akhir ini dikerjakan sebagai syarat kelulusan dan gelar Sarjana Ilmu Komputer Jurusan Ilmu Komputer pada Fakultas Ilmu Komputer Universitas Indonesia. Selama penggeraan tugas akhir, penulis tidak dapat menyelesaikan tugas ini sendiri tanpa bantuan orang lain. Maka penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Kedua orang tua beserta adik penulis yang selalu memberikan doa, kasih sayang, serta bantuan berupa moral dan material kepada penulis.
2. Bapak Alfan Farizki Wicaksono dan juga Bapak Rahmad Mahendra, sebagai dosen pembimbing penulis yang banyak memberikan arahan, masukan, dan bantuan dalam menyelesaikan tugas akhir ini.
3. Bapak Ari Saptawijaya, selaku dosen pembimbing akademik penulis selama kuliah di Failkom UI.
4. Saudara penulis yang memberikan tempat tinggal dan mengurus kehidupan penulis selama penulis kuliah di Universitas Indonesia
5. Hoshimachi Suisei yang menemani penulis dengan lagunya yang indah selama penulis menjalani perkuliahan dan mengerjakan tugas akhir.
6. Zalfa sebagai orang yang spesial bagi penulis yang sudah banyak membantu penulis dalam berbagai hal sejak SMA hingga penulis selesai menyelesaikan tugas akhir.
7. Teman-teman penghuni mebu, yaitu Andre, Dodo, Fadhil, Hadi, Hanif, Haydar, Jojo, Matthew, Steven, William, dan lain-lain, yang telah membantu penulis selama penulis menjalani perkuliahan di Universitas Indonesia.
8. Teman-teman paket hamzah, yaitu Aldan, Ghiyas, Ivan, Rafi, Ryan, dan Sabana yang membantu penulis sejak masa SMA sehingga penulis dapat diterima di Universitas Indonesia.
9. Pihak-pihak lainnya yang sudah membantu penulis selama masa perkuliahan baik secara langsung maupun tidak langsung.

Sebagai penutup, penulis meminta maaf atas kekurangan yang terdapat pada tugas akhir ini. Penulis menyadari bahwa masih banyak kekurangan dalam penelitian yang dilakukan. Oleh karena itu, penulis mengharapkan kritik dan saran yang dapat membangun agar penulis dapat lebih baik lagi kedepannya. Harapannya, penelitian ini dapat bermanfaat bagi perkembangan ilmu pengetahuan di masa depan dan bagi semua pihak yang terlibat.

Depok, 15 Juni 2023



Hendrico Kristiawan

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Hendrico Kristiawan
NPM : 1906350912
Program Studi : Ilmu Komputer
Fakultas : Ilmu Komputer
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Right)** atas karya ilmiah saya yang berjudul:

Klasifikasi Domain Spesialisasi Dokter pada Data Teks Forum Tanya Jawab Kesehatan beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 15 Juni 2023
Yang menyatakan



(Hendrico Kristiawan)

ABSTRAK

Nama : Hendrico Kristiawan
Program Studi : Ilmu Komputer
Judul : Klasifikasi Domain Spesialisasi Dokter pada Data Teks Forum Tanya Jawab Kesehatan
Pembimbing : Alfan Farizki Wicaksono, S.T., M.Sc., Ph.D.
Rahmad Mahendra, S.Kom., M.Sc.

Pertanyaan konsultasi pada sebuah forum daring perlu dijawab oleh dokter spesialis yang tepat agar jawaban yang diberikan akurat dan bermanfaat bagi pengguna yang bertanya. Terkait hal tersebut, penelitian ini membahas tentang pengembangan model yang dapat secara otomatis mengarahkan sebuah pertanyaan konsultasi kesehatan ke dokter dengan spesialisasi yang sesuai. Lebih jauh lagi, model yang dibangun merupakan model klasifikasi *multi-label* karena sebuah pertanyaan dapat terasosiasi dengan lebih dari satu spesialisasi. Penelitian ini dimulai dengan mengevaluasi keefektifan metode pemetaan berbasis aturan dalam memprediksi data yang dianotasi oleh pakar, dan diperoleh hasil yang menunjukkan tingkat keberhasilan yang cukup. Selanjutnya, dikembangkan sebuah model *machine learning* yang melakukan klasifikasi domain spesialis dokter. Pelatihan model dilakukan dengan berbagai metode, termasuk *supervised*, *unsupervised*, serta *semi-supervised learning*. Model terbaik ditemukan melalui metode *domain adaptive pre-training* dengan IndoBERT-large sebagai model acuan dan melibatkan *unsupervised learning*. Selain itu, model *supervised learning* juga digunakan dengan menggunakan model konvensional, dan hasilnya digunakan untuk analisis kontribusi dari fitur-fitur yang digunakan dalam klasifikasi. Terakhir, penelitian ini mengevaluasi kembali anotasi yang dilakukan oleh manusia dengan menggunakan kata kunci sebagai pendekatan untuk mengurangi kesalahan dalam *dataset*. Dengan pendekatan ini, berhasil ditemukan beberapa kesalahan anotasi pada *dataset* yang dianotasi oleh manusia.

Kata kunci:

Klasifikasi Domain Spesialis Dokter, *Supervised-Learning*, *Unsupervised Learning*, *Domain Adaptive Pre-Training*, *Semi-supervised Learning*, GAN-BERT, Analisis Kontribusi Fitur, Data Tidak Berlabel.

ABSTRACT

Name : Hendrico Kristiawan
Study Program : Computer Science
Title : Classification of Doctor Specialization Domain in Health Question and Answer Forum Text Data
Counsellor : Alfan Farizki Wicaksono, S.T., M.Sc., Ph.D.
Rahmad Mahendra, S.Kom., M.Sc.

The consultation questions on an online forum need to be answered by the appropriate specialist doctors to provide accurate and beneficial answers to the users asking the questions. In relation to this, this study discusses the development of a model that can automatically direct a health consultation question to a doctor with the corresponding specialization. Furthermore, the constructed model is a multi-label classification model because a question can be associated with more than one specialization. There are several issues addressed in this work. This research begins by evaluating the effectiveness of rule-based mapping methods in predicting data annotated by experts, and the results show a satisfactory level of success. Furthermore, a multi-label classification model is developed to classify the specialist domains of doctors. The model training is performed using various methods, including supervised learning, unsupervised learning, and semi-supervised learning. The best model is found through domain adaptive pre-training using IndoBERT-large as the reference model and involving unsupervised learning. Additionally, the supervised learning model is also used with a conventional model, and the results are used to analyze the contribution of the features used in the classification. Lastly, this research re-evaluates the annotations made by humans using keyword-based approaches to reduce errors in the dataset. With this approach, several annotation errors were successfully identified in the dataset annotated by humans.

Key words:

Specialized Doctor Domain Classification, Supervised Learning, Unsupervised Learning, Domain Adaptive Pre-Training, Semi-Supervised Learning, GAN-BERT, Analysis of Feature Effects, Unlabeled Data.

DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PENGESAHAN	ii
KATA PENGANTAR	iii
LEMBAR PERSETUJUAN PUBLIKASI ILMIAH	v
ABSTRAK	vi
DAFTAR ISI	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiv
DAFTAR KODE PROGRAM	xviii
DAFTAR SINGKATAN	xix
DAFTAR ISTILAH	xx
DAFTAR LAMPIRAN	xxiii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	4
1.3 Tujuan Penelitian	4
1.4 Ruang Lingkup Penelitian	5
1.5 Tahapan Penelitian	5
1.6 Sistematika Penulisan	6
2 KERANGKA BERPIKIR	8
2.1 Machine Learning	8
2.2 Klasifikasi <i>Multi-Label</i>	9
2.2.1 Binary Relevance (BR)	10
2.2.2 Chain Classifier (CC)	11
2.2.3 Label Powerset (LP)	12
2.3 Algoritma Klasifikasi	14
2.3.1 Logistic Regression	14
2.3.2 Support Vector Machine (SVM)	14
2.3.3 Stochastic Gradient Descent (SGD)	15
2.3.4 Decision Tree	16

2.3.5	Extreme Gradient Boosting (XGBoost)	17
2.3.6	Multinomial Naive Bayes	22
2.4	Pengolahan teks	22
2.5	Pre-Trained Language Model	24
2.5.1	Bidirectional Encoder Representations from Transformers (BERT)	24
2.5.2	Domain Adaptive Pre-training (DAPT)	27
2.5.3	Generative Adversarial Networks BERT (GAN-BERT)	27
2.6	Metrik Evaluasi	29
2.7	Analisis Model <i>Machine Learning</i>	33
2.7.1	Shapley Additive Explanations (SHAP)	33
2.7.2	Almost Stochastic Order (ASO)	35
2.8	Klasifikasi Spesialisasi Dokter pada Teks Kesehatan	35
2.8.1	Domain Spesialisasi Dokter	36
2.8.2	Penelitian sebelumnya	37
3	METODOLOGI	40
3.1	Penelitian Klasifikasi Domain Spesialisasi Dokter	40
3.2	Kajian <i>Dataset</i>	41
3.2.1	<i>Dataset</i> Penelitian	42
3.2.2	Persiapan <i>Dataset</i>	43
3.2.3	Eksplorasi <i>Dataset</i>	45
3.3	Pengembangan Model Prediksi Spesialisasi Dokter	47
3.3.1	Model <i>Machine Learning</i> Konvensional	47
3.3.2	BERT	49
3.4	Pemanfaatan Data Tidak Berlabel	50
3.4.1	Analisis dengan Kata Kunci	51
3.4.2	DAPT	52
3.4.3	GAN-BERT	53
3.5	Evaluasi Model dan <i>Dataset</i>	54
3.5.1	Shapley Additive Explanations	54
3.5.2	Evaluasi pada Anotasi Data	55
4	IMPLEMENTASI	56
4.1	Persiapan Data	56
4.2	Uji Similaritas Data Anotasi Manusia dan Anotasi Mesin	57
4.3	Model <i>Machine Learning</i> Konvensional	59
4.4	Pre-Trained Language Model	61
4.4.1	BERT	62
4.4.2	Domain Adaptive Pre-Training	65
4.4.3	GAN-BERT	67
4.5	Penentuan Kata Kunci	70
4.6	SHAP	72
5	EKSPERIMEN DAN ANALISIS HASIL	74
5.1	Kajian <i>Dataset</i>	74
5.1.1	Eksplorasi Label <i>Dataset</i>	74
5.1.2	Similaritas anotasi manusia dengan anotasi mesin	76

5.2	Pengembangan Model Prediksi Spesialisasi Dokter	79
5.2.1	Metode Konvensional	80
5.2.2	BERT	85
5.3	Pemanfaatan Data Tidak Berlabel	86
5.3.1	Analisis dengan Kata Kunci	87
5.3.2	DAPT	89
5.3.3	GAN-BERT	91
5.4	Evaluasi Model dan <i>Dataset</i>	94
5.4.1	Shapley Additive Explanations (SHAP)	95
5.4.2	Evaluasi pada Anotasi Data	96
6	PENUTUP	99
6.1	Kesimpulan	99
6.2	Saran	100
	DAFTAR REFERENSI	102

DAFTAR GAMBAR

Gambar 1.1.	Contoh pertanyaan konsultasi yang dapat diklasifikasikan ke lebih dari 1 dokter spesialis	2
Gambar 2.1.	Contoh grafik dari fungsi logistik $\sigma(x) = \frac{1}{1+e^{-x}}$	15
Gambar 2.2.	<i>Support vector</i> melakukan klasifikasi dengan memisahkan <i>dataset</i> yang berwarna biru dan merah. Jika sebuah <i>instance</i> baru berada di area merah, maka <i>instance</i> tersebut diklasifikasi sebagai warna merah. Gambar ini didapatkan dari Pedregosa et al. (2011)	16
Gambar 2.3.	Contoh <i>decision tree</i> yang mengklasifikasi sesuai dengan kondisi yang berlaku	18
Gambar 2.4.	Ilustrasi bagaimana hasil prediksi dari model $F(x)$ didapatkan berdasarkan Persamaan 2.14. Gambar diperoleh dari T. Chen dan Guestrin (2016)	19
Gambar 2.5.	Ilustrasi bagaimana XGBoost mencari bobot w_j^* yang membagi data x menjadi <i>instance</i> I_j yang meminimalkan <i>loss</i> . Gambar diperoleh dari T. Chen dan Guestrin (2016)	21
Gambar 2.6.	Contoh kalimat dan hasil tokenisasi teks dengan panjang vektor delapan token	25
Gambar 2.7.	Tok merupakan token dari kalimat masukan dan diubah menjadi E sebagai vektor <i>embedding</i> dari token yang diberikan. Keluaran dari model BERT merupakan vektor T yang digunakan untuk menyelesaikan <i>task</i> baik pada tahap <i>pre-training</i> maupun <i>fine-tuning</i> . Gambar ini didapatkan dari Devlin, Chang, Lee, dan Toutanova (2019)	26
Gambar 2.8.	Ilustrasi hal yang dilakukan DAPT agar <i>pre-trained language model</i> dapat fokus ke domain yang beririsan. Gambar diambil dari penelitian Gururangan et al. (2020)	27
Gambar 2.9.	Arsitektur dari GAN-BERT terdiri dari 3 model besar, yaitu generator (G), diskriminator (D), dan BERT. Masukan <i>noise</i> (z) dimasukkan ke model G dan keluarannya ialah vektor F yang mengikuti distribusi keluaran dari model BERT. Model D menentukan apakah masukan yang diterima berasal dari data asli atau tidak, beserta memprediksi label dari masukan tersebut. Gambar diambil dari penelitian Croce, Castellucci, dan Basili (2020)	30
Gambar 2.10.	Ilustrasi <i>effect size</i> atau kontribusi dari suatu fitur x . Gambar diambil dari penelitian Lundberg dan Lee (2017)	35
Gambar 2.11.	Ilustrasi distribusi yang dicari oleh ASO. Nilai ε_{min} yang dihasilkan merupakan area yang berwarna merah. Gambar diambil dari penelitian Ulmer, Hardmeier, dan Frellsen (2022)	36
Gambar 3.1.	Gambaran garis besar sistem tanya jawab medis	41

Gambar 3.2.	Persebaran data anotasi oleh manusia. Pada Gambar 3.2a, SpKK merupakan kelas "kulit dan kelamin", SpPD sebagai kelas "penyakit dalam", dan SpOG adalah kelas "kandungan".	42
Gambar 3.3.	Persebaran data anotasi oleh mesin. Pada Gambar 3.2a, SpKK merupakan kelas "kulit dan kelamin", SpPD sebagai kelas "penyakit dalam", dan SpOG adalah kelas "kandungan".	43
Gambar 3.4.	Contoh pertanyaan konsultasi yang terdiri dari judul dan isi	53
Gambar 5.1.	Contoh anotasi manusia yang tidak konsisten	75
Gambar 5.2.	Korelasi yang didapatkan dengan metode jarak Jaccard. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0,3 . . .	76
Gambar 5.3.	Korelasi yang didapatkan dengan metode PPMI. Nilai minimum yang dibutuhkan agar terbentuk korelasi harus lebih besar dari 0 . . .	77
Gambar 5.4.	Lima contoh judul yang kurang menjelaskan pertanyaan dan terdapat data pada anotasi manusia dengan label "paru"	79
Gambar 5.5.	Nilai <i>geomean</i> terhadap metrik EMR untuk setiap jumlah fitur SVD yang dipilih	85
Gambar 5.6.	Analisis kemunculan kata kunci untuk beberapa label dengan performa yang rendah	88
Gambar 5.7.	Hasil SHAP dari beberapa sampel kelas yang memiliki pola yang dapat digeneralisasi. Untuk mempersingkat nama fitur, hasil prediksi label sebelumnya disingkat menjadi gelar dari dokter. Yaitu kelas "anak" menjadi SpA, "bedah" menjadi SpB, "gigi" menjadi SKG, "gizi" menjadi SpGK, "jantung" menjadi SpJP, "jiwa" menjadi SpKJ, "kulit dan kelamin" menjadi SpKK, "mata" menjadi SpM, "saraf" menjadi SpN, "kandungan" menjadi SpOG, "tulang" menjadi SpOT, "paru" menjadi SpP, "penyakit dalam" menjadi SpPD, "THT" menjadi SpTHT, dan "urologi" menjadi SpU.	95
Gambar 1.	Data teks yang diduga terjadi kesalahan anotasi pada dataset anotasi oleh dokter	135
Gambar 2.	Korelasi yang didapatkan dengan metode jarak Jaccard. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0,3 . . .	137
Gambar 3.	Korelasi yang didapatkan dengan metode PPMI. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0	139
Gambar 4.	Korelasi yang didapatkan dengan metode PPMI. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0,04	141
Gambar 5.	Hasil SHAP dari beberapa sampel kelas yang memiliki pola yang dapat digeneralisasi. Untuk mempersingkat nama fitur, hasil prediksi label sebelumnya disingkat menjadi gelar dari dokter. Yaitu kelas anak menjadi SpA, bedah menjadi SpB, gigi menjadi SKG, gizi menjadi SpGK, jantung menjadi SpJP, jiwa menjadi SpKJ, kulit dan kelamin menjadi SpKK, mata menjadi SpM, saraf menjadi SpN, kandungan menjadi SpOG, tulang menjadi SpOT, paru menjadi SpP, penyakit dalam menjadi SpPD, THT menjadi SpTHT, dan urologi menjadi SpU.	150

Gambar 6. Data teks yang diduga terjadi kesalahan anotasi pada dataset anotasi oleh dokter 154

DAFTAR TABEL

Tabel 2.1.	Contoh <i>dataset</i> klasifikasi <i>multi-label</i> . Masukan berupa x dan terdapat tiga label y yang harus diprediksi.	10
Tabel 2.2.	Contoh hasil transformasi <i>dataset</i> dengan metode BR berdasarkan data pada Tabel 2.1.	11
Tabel 2.3.	Contoh hasil transformasi <i>dataset</i> dengan metode CC berdasarkan data pada Tabel 2.1.	12
Tabel 2.4.	Contoh <i>dataset</i> hasil transformasi LP. Masukan berupa x dan terdapat sembilan kelas y_{LP} yang menjadi target prediksi.	13
Tabel 2.5.	Contoh data <i>ground truth</i> dan prediksi dari klasifikasi <i>multi-label</i> . . .	32
Tabel 3.1.	Skenario yang dilakukan pada masing-masing tahapan dalam membuat model <i>machine learning</i> konvensional	47
Tabel 3.2.	<i>Hyperparameter</i> yang digunakan saat melakukan <i>fine-tuning</i> model BERT	50
Tabel 3.3.	<i>Hyperparameter</i> yang digunakan saat melakukan DAPT terhadap model BERT terbaik pada tahap <i>supervised learning</i>	52
Tabel 3.4.	Hasil transformasi data teks menjadi <i>dataset</i> yang terdiri dari teks 1, teks 2, dan NSP. NSP bernilai satu ketika teks 2 merupakan kelanjutan dari teks 1 pada data teks awal.	53
Tabel 3.5.	<i>Hyperparameter</i> yang digunakan saat melatih model GAN-BERT . .	54
Tabel 5.1.	Nilai <i>Cohen's kappa</i> kelas "umum" dan rata-rata <i>Cohen's kappa</i> keseluruhan untuk setiap pembagian <i>annotator</i> seperti yang dijelaskan pada Subbab 3.2.1 (halaman 42).	75
Tabel 5.2.	Hasil evaluasi metrik similaritas menggunakan presisi (Prec), <i>recall</i> (Rec), <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), <i>hamming loss</i> (Hamm), dan <i>Cohen's kappa</i> (Kappa). <i>Support</i> (Sup) ditampilkan karena jumlah <i>instance</i> untuk setiap metode percobaan berbeda. Hal tersebut diakibatkan tidak semua pertanyaan menghasilkan label dari penggunaan pemetaan kamus. Untuk hasil yang menggunakan skenario membatasi tiga label diberikan tanda <i>top 3</i>	76
Tabel 5.3.	Nilai presisi (Prec), <i>recall</i> (Rec), & <i>f1-score</i> (F1) dari metode pemetaan kamus dengan jumlah label dibatasi sampai tiga. Data anotasi manusia dijadikan sebagai target label yang perlu diprediksi. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (<i>micro avg</i>).	78
Tabel 5.4.	Hasil evaluasi 469 <i>instance</i> data uji dengan pemetaan berdasarkan judul dan banyak label yang dibatasi. Metrik yang digunakan berupa presisi (Prec), <i>recall</i> (Rec), <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	79

Tabel 5.5.	Hasil evaluasi 10-fold cross validation pada model dengan skenario BR dan menggunakan data anotasi manusia menjadi data latih. Metrik yang digunakan berupa <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	80
Tabel 5.6.	Gugusan yang dihasilkan dari kedua metode dan urutan terbaik dari gugusan tersebut	81
Tabel 5.7.	Hasil evaluasi 469 <i>instance</i> data uji dengan enam model <i>machine learning</i> konvensional terbaik. Metrik yang digunakan berupa <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	82
Tabel 5.8.	Hasil uji t-test untuk enam model terbaik berdasarkan tiga metrik yang digunakan. Tabel dibaca sebagai berikut, jika model pada baris lebih baik dibanding model pada kolom, maka diberikan simbol (+). Sebaliknya jika model pada baris lebih buruk dibanding model pada kolom, maka diberikan simbol (-). Simbol (?) diberikan jika nilai <i>p-value</i> lebih besar dari 0,05 sehingga tidak bisa dibuktikan. Model stem_svd100_svc_cc disingkat sebagai A, model stemstop_svd100_svc_br sebagai B, model stemstop_svd100_svc_cc sebagai C, model stemstop_svd250_svc_br sebagai D, model stemstop_svd250_svc_cc sebagai E, dan model stemstop_svd500_sgd-logreg_cc sebagai F	82
Tabel 5.9.	Hasil uji t-test pada tahap pendekatan multi-label. Tabel dibaca sebagai berikut, jika metode pada baris lebih baik dibanding metode pada kolom, maka diberikan simbol (+). Sebaliknya jika metode pada baris lebih buruk dibanding metode pada kolom, maka diberikan simbol (-). Simbol (?) diberikan jika nilai <i>p-value</i> lebih besar dari 0,05 sehingga tidak bisa dibuktikan.	84
Tabel 5.10.	Hasil uji t-test pada tahap pemilihan model yang digunakan berdasarkan metrik EMR. Cara membaca tabel sama seperti Tabel 5.9.	84
Tabel 5.11.	Hasil uji t-test pada tahap prapemrosesan teks. Hasil t-test terhadap setiap metrik memiliki hasil yang sama seperti. Cara membaca tabel sama seperti Tabel 5.9.	84
Tabel 5.12.	Hasil evaluasi 469 <i>instance</i> data uji dengan lima model <i>supervised learning deep learning</i> yang masing-masing dijalankan sebanyak lima kali. Metrik yang digunakan berupa <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	86
Tabel 5.13.	Hasil ϵ_{min} antar model yang didapatkan dari metode ASO untuk mencari model yang lebih baik. Nilai pada suatu <i>cell</i> merupakan nilai ϵ_{min} dari pernyataan: model pada baris lebih baik dibandingkan model pada kolom ketika nilai $\epsilon_{min} \leq 0,5$. Model IndoNLU-base disingkat sebagai A, model IndoWH sebagai B, model IndoLEM sebagai C, model IndoNLU-large sebagai D, dan model mBERT sebagai E.	86
Tabel 5.14.	Nilai presisi (Prec), <i>recall</i> (Rec), & <i>f1-score</i> (F1) setiap label dari hasil prediksi model IndoNLU-large terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (<i>micro avg</i>).	87

Tabel 5.15. Hasil evaluasi 469 <i>instance</i> data uji dengan 10 model <i>unsupervised learning deep learning</i> yang masing-masing dijalankan sebanyak lima kali. Metrik yang digunakan berupa <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	90
Tabel 5.16. Hasil ϵ_{min} antar model yang didapatkan dari metode ASO untuk mencari model yang lebih baik. Metrik EMR digunakan sebagai pembanding antar model. Nilai pada suatu <i>cell</i> merupakan nilai ϵ_{min} dari pernyataan: model pada baris lebih baik dibandingkan model pada kolom. Model pada baris dikatakan lebih baik dari model pada kolom ketika nilai $\epsilon_{min} \leq 0,5$. Model DAPT pada <i>epoch</i> pertama disingkat sebagai 1, model DAPT pada <i>epoch</i> kedua sebagai 2, dan seterusnya hingga model DAPT pada <i>epoch</i> ke-10 sebagai 10.	90
Tabel 5.17. Nilai presisi (Prec), <i>recall</i> (Rec), & <i>f1-score</i> (F1) setiap label dari hasil prediksi model daptE6 terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (<i>micro avg</i>).	91
Tabel 5.18. Hasil evaluasi 469 <i>instance</i> data uji dengan dua model <i>semi-supervised learning deep learning</i> yang masing-masing dijalankan sebanyak lima kali. Metrik yang digunakan berupa <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	92
Tabel 5.19. Nilai ϵ_{min} yang dihasilkan ASO. Model GAN-BERT-IndoNLU diinisialkan sebagai A dan model GAN-BERT-DAPT sebagai B. . . .	92
Tabel 5.20. Nilai presisi (Prec), <i>recall</i> (Rec), & <i>f1-score</i> (F1) setiap label dari hasil prediksi model GAN-BERT-IndoNLU terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (<i>micro avg</i>).	93
Tabel 5.21. Hasil evaluasi 469 <i>instance</i> data uji dengan enam model <i>deep learning</i> dan dipilih dua model terbaik dari masing-masing pendekatan. Metrik yang digunakan berupa <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	94
Tabel 5.22. Hasil ϵ_{min} antar model yang didapatkan dari metode ASO untuk mencari model yang lebih baik. Nilai pada suatu <i>cell</i> merupakan nilai ϵ_{min} dari pernyataan: model pada baris lebih baik dibandingkan model pada kolom. Model pada baris dikatakan lebih baik dari model pada kolom ketika nilai $\epsilon_{min} \leq 0,5$. Model IndoNLU-large disingkat sebagai A, model daptE6 sebagai B, model daptE9 sebagai C, model GAN-BERT-IndoNLU sebagai D, dan model GAN-BERT-DAPT sebagai E.	94
Tabel 5.23. Hasil evaluasi 469 <i>instance</i> data uji dengan enam model <i>deep learning</i> dan dipilih dua model terbaik dari masing-masing pendekatan. Metrik yang digunakan berupa <i>exact match ratio</i> (EMR), <i>f1-score</i> (F1), dan <i>hamming loss</i> (Hamm).	97
Tabel 5.24. Nilai presisi (Prec), <i>recall</i> (Rec), & <i>f1-score</i> (F1) setiap label dari hasil prediksi model daptE6 terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (<i>micro avg</i>).	97
Tabel 5.25. Nilai ϵ_{min} yang dihasilkan ASO. Model daptE6 yang dilatih sebelum anotasi ulang disebut sebagai A dan model yang dilatih setelah anotasi ulang sebagai B.	98

Tabel 1.	Kamus yang digunakan untuk pemetaan	108
Tabel 2.	Performa setiap skenario yang dijalankan untuk model <i>machine learning</i> konvensional dengan <i>10-fold cross validation</i> menggunakan data anotasi manusia menjadi data latih	125

DAFTAR KODE PROGRAM

Kode 4.1.	Kode membersihkan data teks	56
Kode 4.2.	Kode mengecek similaritas pada data pertanyaan	57
Kode 4.3.	Kode menyaring <i>instance</i> dengan minimal <i>k</i> kata kunci	57
Kode 4.4.	Kode prapemrosesan kalimat agar bisa dipetakan	58
Kode 4.5.	Kode pemetaan dengan jumlah label tidak dibatasi	58
Kode 4.6.	Kode pemetaan dengan jumlah label dibatasi hingga tiga label	58
Kode 4.7.	Kode melakukan pemetaan kamus ke <i>dataset</i>	59
Kode 4.8.	Kode prapemrosesan kalimat untuk <i>machine learning</i> konvensional	59
Kode 4.9.	Kode pipeline untuk ekstraksi dan reduksi fitur	60
Kode 4.10.	Kode membuat model <i>machine learning</i> konvensional	60
Kode 4.11.	Kode penerapan <i>k-fold cross validation</i> untuk permasalahan <i>multi-label</i>	60
Kode 4.12.	Kode untuk menguji t-test secara berpasangan	61
Kode 4.13.	Kode membuat <i>dataloader</i> untuk masukan BERT menggunakan <i>PyTorch</i>	62
Kode 4.14.	Kode kelas dari model klasifikasi dan cara membuat model	63
Kode 4.15.	Kode membuat optimizer dan <i>scheduler</i> beserta menjalankan pelatihan model	63
Kode 4.16.	Kode tahapan melatih model	64
Kode 4.17.	Kode tahapan validasi model	64
Kode 4.18.	Kode implementasi pembuatan <i>dataset</i> NSP	65
Kode 4.19.	Kode implementasi pembuatan <i>dataset</i> MLM	66
Kode 4.20.	Kode pembuatan optimizer dan memulai melatih model DAPT	67
Kode 4.21.	Kode melatih model BERTForPreTraining yang sudah disediakan oleh Hugging Face	67
Kode 4.22.	Kode untuk menggabungkan data berlabel dan tidak berlabel menjadi data latih	67
Kode 4.23.	Kode kelas arsitektur diskriminator dan generator	68
Kode 4.24.	Kode pembuatan arsitektur diskriminator dan generator	69
Kode 4.25.	Kode untuk membuat optimizer dan <i>scheduler</i> pada masing-masing model, serta menjalankan pelatihan model	69
Kode 4.26.	Kode modifikasi <i>loss</i> yang diterapkan	70
Kode 4.27.	Kode kelas untuk menghitung skor dari token	71
Kode 4.28.	Kode implementasi untuk menilai apakah suatu token termasuk kata kunci atau tidak	72
Kode 4.29.	Kode implementasi BR dan CC tanpa <i>library</i>	72
Kode 4.30.	Kode penerapan SHAP <i>explainer</i> untuk kasus <i>multi-label</i>	73
Kode 4.31.	Kode menampilkan grafik hasil analisis SHAP untuk kasus <i>multi-label</i> .	73

DAFTAR SINGKATAN

ASO	: <i>Almost Stochastic Order</i>
BERT	: <i>Bidirectional Encoder Representations from Transformers</i>
BR	: <i>Binary Relevance</i>
CC	: <i>Classifier Chain</i>
DAPT	: <i>Domain Adaptive Pretraining</i>
EMR	: <i>Exact Match Ratio</i>
FKUI	: Fakultas Kedokteran Universitas Indonesia
GAN	: <i>Generative Adversarial Networks</i>
Hamm	: <i>Hamming Loss</i>
IDF	: <i>Inverse Document Frequency</i>
LP	: <i>Label Powerset</i>
LogReg	: <i>Logistic Regression</i>
NB	: <i>Naive Bayes</i>
Prec	: <i>Precision</i>
Rec	: <i>Recall</i>
SHAP	: <i>SHapley Additive exPlanations</i>
SGD	: <i>Stochastic Gradient Descent</i>
Stem	: Melakukan <i>stemming</i>
Stemstop	: Melakukan <i>stemming</i> dan menghapus <i>stopword</i>
SVC	: <i>Support Vector Classifier</i>
SVM	: <i>Support Vector Machine</i>
Sup	: <i>Support</i>
SVD	: <i>Singular Value Decomposition</i>
TF	: <i>Term Frequency</i>
THT	: Telinga Hidung Tenggorokan
XGB/XGBoost	: <i>Extreme Gradient Boosting</i>

DAFTAR ISTILAH

- Attention* : Mekanisme dalam model *neural network* yang memberikan bobot lebih pada bagian penting dari masukan.
- Collinearity* : Hubungan linier yang kuat antara dua atau lebih fitur yang menjadi masukan dalam model *machine learning*.
- Deep Learning* : Subbidang dalam *machine learning* yang menggunakan arsitektur model jaringan saraf untuk mempelajari representasi fitur yang kompleks.
- Embedding* : Representasi data pada dimensi yang lebih rendah dari data berdimensi tinggi seperti kata atau dokumen.
- Epoch* : Satu putaran lengkap dari seluruh dataset latih yang digunakan saat melatih model.
- Fine-Tuning* : Proses menyesuaikan model yang sudah dilatih sebelumnya ke tugas yang lebih spesifik.
- Gradient Descent* : Algoritma optimisasi yang digunakan untuk menemukan nilai optimal dari fungsi objektif dengan *mengiterasi* perubahan parameter model dalam arah negatif *gradiennya*.
- Information Gain* (*Decision Tree*) : Ukuran perubahan informasi pada setiap pemisahan *node* dalam *decision tree* berdasarkan parameter yang ditentukan.
- Jarak Jaccard : Ukuran kemiripan antara dua himpunan berdasarkan persentase elemen yang sama di antara keduanya.
- Kontribusi : Pengaruh suatu fitur terhadap hasil atau performa model.
- Learning Rate* : Parameter yang mengontrol seberapa besar perubahan yang dibuat pada model untuk setiap iterasi dalam proses pelatihan.
- Log Scaling* : Mengubah nilai data ke dalam skala logaritma.
- Loss* : Ukuran numerik yang menggambarkan perbedaan antara hasil prediksi dan nilai sebenarnya yang diinginkan.
- Machine Learning* : Pendekatan komputasional untuk memungkinkan sistem komputer belajar dari data dan meningkatkan kinerjanya seiring berjalannya waktu.

<i>Negative Gradient</i>	: Gradien fungsi objektif yang menunjukkan arah penurunan terhadap nilai parameter model.
<i>N-Gram</i>	: Serangkaian n kata yang berdekatan dalam teks atau dokumen.
<i>Optimisasi</i>	: Proses mencari nilai optimal dalam ruang solusi untuk mencapai tujuan yang ditentukan.
<i>Overfitting</i>	: Keadaan di mana model <i>machine learning</i> terlalu sempurna dalam mempelajari data latih, sehingga menyebabkan performa yang buruk ketika menggunakan data baru.
<i>Positive Pointwise Mutual Information</i>	: Ukuran statistik yang mengukur sejauh mana dua variabel berhubungan satu sama lain dengan fokus pada asosiasi positif.
<i>Pre-training</i>	: Proses pelatihan awal model pada tugas yang serupa sebelum dilakukan penyesuaian lebih lanjut pada tugas yang spesifik.
<i>Prior Distribution</i>	: Distribusi probabilitas yang menggambarkan keyakinan awal tentang parameter atau variabel acak sebelum observasi data.
<i>Regularisasi</i>	: Teknik dalam <i>machine learning</i> yang digunakan untuk menghindari <i>overfitting</i> dan meningkatkan generalisasi model dengan memberikan batasan pada <i>kompleksitas</i> model.
<i>Residual Error</i>	: Selisih antara nilai yang diprediksi oleh model dan nilai sebenarnya dalam data pengujian.
<i>Semi-Supervised Learning</i>	: Tipe pembelajaran mesin di mana model menggunakan gabungan data yang berlabel dan tidak berlabel untuk melatih model.
<i>Single-label</i>	: Jenis masalah di mana setiap sampel hanya memiliki satu label atau kelas yang benar.
<i>Split (Decision Tree)</i>	: Proses membagi <i>node</i> dalam <i>decision tree</i> berdasarkan parameter yang ditentukan untuk menghasilkan percabangan.
<i>Stochastic Order</i>	: Konsep matematika yang digunakan untuk membandingkan tingkat keacakan dari dua variabel acak atau dua proses stokastik.

- Supervised Learning* : Tipe pembelajaran mesin di mana model dilatih menggunakan pasangan data masukan dan keluaran yang sudah diketahui kelas sebenarnya.
- TF-IDF : Skema pembobotan yang digunakan untuk mengevaluasi pentingnya kata dalam sebuah dokumen dalam korpus berdasarkan frekuensi kemunculannya dalam dokumen dan seluruh korpus.
- T-Test : Metode statistik yang digunakan untuk membandingkan dua kelompok sampel dan menguji apakah ada perbedaan signifikan antara rata-rata mereka.
- Transformer* : Arsitektur model deep learning yang menggunakan mekanisme *attention*.
- Unsupervised Learning* : Tipe pembelajaran mesin di mana model belajar dari data yang tidak memiliki label atau dianotasi.
- Weak Learner* : Model *machine learning* sederhana yang memiliki performa yang sedikit lebih baik daripada menebak secara acak, tetapi masih lebih buruk dibandingkan dengan model yang kompleks.

DAFTAR LAMPIRAN

Lampiran 1.	Kata Kunci untuk Pemetaan Label	108
Lampiran 2.	Performa Model <i>Machine Learning</i> Konvensional	124
Lampiran 3.	Kata Kunci yang Merepresentasikan Label	130
Lampiran 4.	Hasil Korelasi Antar Label	136
Lampiran 5.	Hasil SHAP dari Setiap Kelas	141
Lampiran 6.	Kesalahan Anotasi pada Data Latih	150

BAB 1

PENDAHULUAN

Bab 1 mendiskusikan latar belakang dari permasalahan utama yang diangkat pada laporan tugas akhir ini, yaitu permasalahan klasifikasi domain spesialisasi dokter pada sebuah pertanyaan di forum kesehatan secara otomatis. Pertama, Subbab 1.1 membahas motivasi dan manfaat dari tugas klasifikasi domain spesialisasi dokter dari sebuah pertanyaan serta pekerjaan utama yang terkait dengan penelitian ini. Kemudian, Subbab 1.2 menyampaikan empat buah rumusan masalah yang menjadi fokus pada tugas akhir ini dan Subbab 1.3 menampilkan daftar tujuan yang ingin dicapai dari tugas akhir ini. Selanjutnya, Subbab 1.4 menyampaikan batasan penelitian dalam hal *dataset* dan juga jenis spesialisasi dokter. Subbab 1.5 membahas langkah-langkah umum yang digunakan dalam proses penelitian. Terakhir, Subbab 1.6 menutup Bab 1 dengan penjelasan terkait sistematika penulisan yang digunakan pada laporan tugas akhir ini.

1.1 Latar Belakang

Menurut kementerian kesehatan, layanan konsultasi penyakit daring atau yang disebut juga sebagai *telemedicine*, dapat membawa revolusi pelayanan kesehatan di Indonesia¹. Terutama pada pandemi Covid-19 pada tahun 2020 silam, pengguna aplikasi layanan konsultasi daring meningkat hingga 600%¹. Bahkan setelah pandemi berakhir, pengguna layanan konsultasi daring tetap meningkat hingga 44% menurut survei yang dilakukan oleh Katadata Insight Center². Dengan demikian, dapat disimpulkan bahwa layanan konsultasi daring dapat menjadi alternatif yang populer dan efektif dalam memberikan layanan kesehatan di Indonesia.

Beberapa aplikasi yang menyediakan layanan konsultasi daring sudah tersedia di Indonesia, seperti Alodokter³, KlikDokter⁴, dan SehatQ⁵. Salah satu jenis konsultasi

¹<https://www.balaibaturaja.litbang.kemkes.go.id/read-aplikasi-telemedicine-berpotensi-merevolusi-pelayanan-kesehatan-di-indonesia>

²<https://katadata.co.id/desysetyowati/digital/624e9b8b96669/jumlah-pengguna-baru-layanan-telemedicine-capai-44-dalam-6-bulan>

³<https://www.alodokter.com/>

⁴<https://www.klikdokter.com/>

⁵<https://www.sehatq.com/>

Isi : Dok saya mau tanya apakah ada efek samping dari minuman diet seperti wrp pada kehamilan ?
 Label : Gizi & Kandungan

Isi : Saya pernah memiliki gejala penyakit rematik jantung sejak berumur 12 tahun dan pernah dirawat selama 3 minggu . sekarang di usia 28 tahun saya masih memiliki gejala seperti bentol-bentol merah di kaki , dan suka sakit di ulu hati . setiap saya minum kopi badan gemetaran dan jantung berdebar . apakah penyakit itu kambuh lagi ? haruskah saya memeriksakan kembali ke dokter ? terima kasih atas jawaban anda sebelumnya .
 Label : Jantung & Penyakit Dalam

Gambar 1.1: Contoh pertanyaan konsultasi yang dapat diklasifikasikan ke lebih dari 1 dokter spesialis

yang disediakan adalah konsultasi berbasis forum tanya jawab di mana pasien mengajukan sebuah pertanyaan atau konsultasi penyakit, kemudian dokter menjawab pertanyaan tersebut. Dokter yang menjawab merupakan dokter spesialis yang ahli dalam menangani penyakit tersebut sehingga jawaban yang diberikan dapat dipercaya.

Pada konsultasi berbasis forum tanya jawab, tidak semua pasien dapat mengarahkan pertanyaan tentang penyakit yang diderita ke dokter spesialis yang tepat. Oleh karena itu, sangat bermanfaat jika forum mempunyai sistem yang dapat menentukan secara otomatis jenis dokter spesialis yang tepat untuk menjawab pertanyaan pasien tertentu. Hasil dari klasifikasi tersebut juga dapat membantu dokter spesialis untuk menyaring pertanyaan yang sesuai dengan keahliannya.

Nurhayati, Syifa (2019) mengembangkan model yang dapat menentukan label dokter spesialis terhadap sebuah pertanyaan konsultasi medis secara otomatis. Nurhayati, Syifa mengasumsikan bahwa sebuah pertanyaan hanya dipetakan pada satu jenis spesialisasi dokter dan memodelkan permasalahan ini sebagai klasifikasi *single-label*. Namun, beberapa pertanyaan konsultasi mungkin dapat diklasifikasikan ke lebih dari satu label spesialisasi dokter, seperti contoh yang disajikan pada Gambar 1.1. Penelitian kali ini bertujuan untuk menyempurnakan pekerjaan yang dilakukan oleh Nurhayati, Syifa (2019) dengan mengusulkan permasalahan ini sebagai tugas klasifikasi *multi-label*.

Selain pemodelan klasifikasi, ketersediaan *dataset* menjadi isu yang juga dieksplorasi dalam penelitian ini. Hakim, Abid Nurul (2016) mengumpulkan data forum tanya jawab medis dari beberapa sumber dan memberikan label spesialisasi dokter terhadap setiap pertanyaan secara otomatis. Label spesialisasi diberikan dengan memanfaatkan *tag* atau topik yang terdapat pada *website* yang menjadi sumber

pertanyaan. Proses pemberian label secara otomatis tersebut memerlukan aturan pemetaan yang dikembangkan secara manual (Hakim, Abid Nurul, 2016). Sebagai contoh, jika sebuah pertanyaan memiliki *tag* "Abses Gigi", label spesialisasi yang diberikan adalah "Dokter Gigi". Untuk selanjutnya, *dataset* yang dikembangkan oleh Hakim, Abid Nurul (2016) disebut sebagai *dataset* yang "teranotasi oleh mesin dengan pendekatan *rule-based*". Kemudian pada tahun 2022, sejumlah kalimat yang merupakan himpunan bagian dari *dataset* Hakim, Abid Nurul (2016) dianotasi secara manual oleh delapan orang mahasiswa tingkat akhir Fakultas Kedokteran Universitas Indonesia (FKUI)⁶. Dengan adanya dua kelompok *dataset* (yang dianotasi mesin dan manusia), peneliti mengkaji seberapa mirip kedua jenis *dataset* tersebut untuk mencari tahu apakah pemberian label secara manual oleh pakar dapat digantikan oleh mesin. Hal ini dibahas pada penelitian sebab *dataset* latih yang tersedia tidak besar dan ke depannya dibutuhkan sebuah metode untuk memperbesar ukuran *dataset* yang murah dan efektif.

Permasalahan *dataset* juga menjadi salah satu hal yang dibahas pada penelitian ini. Selain menyelidiki kemungkinan pemberian label secara otomatis, penelitian ini juga mencoba menjawab apakah *dataset* tidak berlabel yang berukuran besar dapat dimanfaatkan untuk meningkatkan performa model klasifikasi dengan menerapkan *unsupervised learning* dan *semi-supervised learning*. Metode *unsupervised learning* yang dicoba adalah *domain adaptive pre-training* di mana *language model* dilatih kembali menggunakan *dataset* dengan domain yang sama (Gururangan et al., 2020) sedangkan metode *semi-supervised learning* yang digunakan ialah *generative adversarial networks* BERT (Croce et al., 2020) yang dimodifikasi untuk permasalahan klasifikasi *multi-label*.

Selain isu mengenai *dataset*, kajian dampak dari fitur-fitur yang digunakan pada tugas klasifikasi spesialisasi dokter belum dilakukan secara komprehensif oleh Nurhayati, Syifa (2019) dan Hakim, Abid Nurul (2016). Maka dari itu, penelitian ini menerapkan konsep *Explainable Machine Learning* untuk mengetahui *effect size* dari setiap fitur seperti yang dijelaskan oleh (F. Xu et al., 2019). Salah satu alat yang dapat digunakan untuk mengetahui *effect size* dari masing-masing fitur adalah *SHapley Additive exPlanations* yang disebut juga sebagai SHAP (Lundberg & Lee, 2017). Peneliti menggunakan SHAP untuk mengetahui kontribusi dari masing-masing fitur

⁶Terima kasih kepada Syifa Nurhayati yang menjadi koordinator kegiatan anotasi dari sebagian *dataset* Hakim, Abid Nurul (2016). Pada saat proses anotasi tersebut, Syifa berperan sebagai asisten dosen di mata kuliah Penambangan Data di Fasilkom UI.

yang memengaruhi hasil klasifikasi pada *machine learning*. Dengan begitu, hasil penelitian dapat menjelaskan bagaimana model bekerja secara *white box*.

Sebagai penutup, penelitian yang dikerjakan menghasilkan dua kontribusi utama. Pertama, penelitian ini melakukan kajian dari permasalahan yang belum dibahas sebelumnya, yaitu permasalahan klasifikasi multi-label domain spesialisasi dokter pada pertanyaan kesehatan berbahasa Indonesia. Kajian yang dilakukan meliputi pengembangan model berbasis *data-driven* yang memanfaatkan data berlabel dan tidak berlabel, serta analisis kontribusi atau *effect size* dari fitur-fitur tekstual. Kedua, penelitian ini mengusulkan metodologi untuk menilai seberapa mungkin sebuah *dataset* dapat dihasilkan secara otomatis dan untuk evaluasi ulang kualitas *dataset* yang sudah diberi label oleh manusia.

1.2 Rumusan Masalah

Dari pembahasan latar belakang yang disampaikan pada Subbab 1.1, pertanyaan menyeluruh yang dibahas pada tugas akhir ini adalah ”sejauh mana domain spesialisasi dokter dari sebuah pertanyaan medis dapat ditentukan secara otomatis?” Pertanyaan besar tersebut kemudian dapat dibagi lagi menjadi empat buah rumusan masalah yang saling terkait:

1. Seberapa besar similaritas antara anotasi manusia (*expert*) dengan anotasi otomatis oleh mesin pada permasalahan klasifikasi domain spesialisasi dokter?
2. Bagaimana performa model *machine learning* untuk masalah klasifikasi *multi-label* domain spesialisasi dokter?
3. Apakah penggunaan data tidak berlabel dapat meningkatkan performa klasifikasi domain spesialisasi dokter?
4. Apa saja faktor-faktor yang memengaruhi klasifikasi dari suatu pertanyaan dan seberapa besar *effect size* dari faktor tersebut?

1.3 Tujuan Penelitian

Dari pembahasan rumusan masalah yang disampaikan pada Subbab 1.2, tujuan yang dicapai dari penelitian ini:

- Mengetahui similaritas hasil anotasi manusia (*expert*) dengan anotasi otomatis oleh mesin pada permasalahan klasifikasi domain spesialisasi dokter;
- Membuat model *machine learning* untuk masalah klasifikasi *multi-label* domain spesialisasi dokter;
- Mengetahui pengaruh penggunaan data tidak berlabel untuk meningkatkan performa klasifikasi domain spesialisasi dokter;
- Mengetahui faktor-faktor yang memengaruhi klasifikasi dari suatu pertanyaan dan *effect size* dari faktor tersebut.

1.4 Ruang Lingkup Penelitian

Penelitian ini merupakan lanjutan dari penelitian Hakim, Abid Nurul (2016) sehingga *dataset* yang digunakan hanya mencakup data penelitian Hakim, Abid Nurul (2016) dan anotasi dokter yang dilakukan oleh Nurhayati (2022). Namun untuk klasifikasi domain spesialis mengikuti anotasi Nurhayati (2022) yang terdiri dari anak, bedah, gigi, gizi, jantung, jiwa, kandungan, kulit dan Kelamin, mata, paru, penyakit dalam, saraf, THT, tulang, umum, dan urologi.

1.5 Tahapan Penelitian

Berikut beberapa tahapan yang dilakukan pada penelitian ini agar mencapai tujuan penelitian yang diinginkan:

- Studi Literatur

Pada tahap ini dilakukan pembelajaran mengenai sistem tanya jawab domain kesehatan, algoritma *machine learning* konvensional, model *deep learning*, metode analisis, dan penelitian sebelumnya.

- Perumusan Masalah

Pada tahap ini dilakukan perumusan masalah untuk menentukan masalah yang ingin diselesaikan dan tujuan dari penelitian.

- Rancangan Penelitian

Pada tahap ini dilakukan perancangan penelitian seperti persiapan data, pengusulan model, dan metode evaluasi yang digunakan.

- Implementasi

Pada tahap ini dilakukan implementasi model yang sudah diusulkan untuk menjawab rumusan masalah pada penelitian.

- Analisis dan Kesimpulan

Pada tahap ini dilakukan analisis dari hasil penelitian yang sudah dilakukan. Hasil dari analisis menjawab perumusan masalah yang sudah ditentukan dan menjadi kesimpulan dari penelitian.

1.6 Sistematika Penulisan

Sistematika penulisan laporan adalah sebagai berikut:

- Bab 1 PENDAHULUAN

Bab ini mencakup latar belakang, rumusan masalah, tujuan penelitian, ruang lingkup penelitian, langkah penelitian, dan sistematika penulisan.

- Bab 2 KERANGKA BERPIKIR

Bab ini menjelaskan literatur yang berkaitan dengan penelitian tentang klasifikasi dan teori yang mendukung penelitian ini.

- Bab 3 METODOLOGI

Bab ini menjelaskan metodologi penelitian, terdiri dari persiapan data, perancangan model, metrik evaluasi, dan metode analisis yang dilakukan dalam penelitian.

- Bab 4 IMPLEMENTASI

Bab ini menjelaskan proses implementasi model *machine learning* yang diusulkan sebelumnya dan eksperimen yang dilakukan pada penelitian ini.

- Bab 5 EKSPERIMEN DAN ANALISIS HASIL

Bab ini menjelaskan hasil dan analisis dari eksperimen yang telah dilakukan.

- Bab 6 PENUTUP

Bab ini berisi kesimpulan dari hasil penelitian serta saran untuk penelitian selanjutnya yang menggunakan atau memperbaiki hasil penelitian ini.

BAB 2

KERANGKA BERPIKIR

Bab 2 menjelaskan literatur yang berkaitan dengan penelitian tentang klasifikasi dan teori yang mendukung penelitian ini. Pertama, Subbab 2.1 menjelaskan konsep-konsep *machine learning* yang penting dan berhubungan dengan penelitian. Kemudian, Subbab 2.2 membahas konsep klasifikasi *multi-label* dan pendekatan yang dapat dilakukan. Selanjutnya, Subbab 2.3 menjelaskan algoritma-algoritma *machine learning* konvensional yang digunakan. Setelah itu, Subbab 2.4 menjelaskan tahapan yang dilakukan untuk mengolah teks. Dilanjutkan dengan Subbab 2.5 membahas *language model* BERT dan arsitektur *deep learning* yang memanfaatkan BERT. Kemudian, Subbab 2.6 membahas metrik yang digunakan pada penelitian. Berikutnya, Subbab 2.7 menjelaskan cara menganalisis model berbasis performa yang didapatkan dan kontribusi fitur dari model. Terakhir, Subbab 2.8 menutup Bab 2 dengan penjelasan mengenai klasifikasi domain spesialisasi dokter beserta penelitian yang sudah pernah dilakukan.

2.1 Machine Learning

Berdasarkan yang sudah dijelaskan pada Subbab 1.2 (halaman 4), penelitian ini menggunakan model *machine learning* untuk menyelesaikan permasalahan klasifikasi domain spesialisasi dokter. Untuk memberikan pemahaman dari topik ini, Subbab 2.1 membahas lebih detail tentang paradigma pemelajaran *machine learning* yang digunakan beserta jenis permasalahan klasifikasi yang dihadapi.

Menurut Bernard (2021), terdapat tiga paradigma pemelajaran yang utama, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Penelitian ini hanya menggunakan paradigma *supervised learning*, *unsupervised learning*, dan gabungan kedua paradigma tersebut, yaitu *semi-supervised learning*.

Paradigma yang pertama adalah *supervised learning*. *Supervised learning* merupakan salah satu paradigma pemelajaran dalam melatih model *machine learning* dengan memanfaatkan label yang terdapat pada *dataset*. Paradigma ini sangat membutuhkan label agar dapat dilakukan, sehingga metode ini sulit diterapkan jika

proses anotasi label pada *dataset* sulit didapatkan dan mahal.

Selain itu, terdapat juga paradigma *unsupervised learning* yang tidak memanfaatkan data berlabel saat melatih model. Menurut Y. Chen, Mancini, Zhu, dan Akata (2022), paradigma ini bertujuan agar model dapat mempelajari semantik dari informasi dan dapat digunakan untuk berbagai macam *task* yang berhubungan. Untuk beberapa model *machine learning* yang dilatih dengan paradigma *unsupervised learning*, dapat dilatih kembali dengan paradigma *supervised learning* untuk menyelesaikan *task* yang spesifik.

Paradigma terakhir yang digunakan adalah *semi-supervised learning*, yaitu paradigma pemelajaran yang menggunakan data berlabel dan data tidak berlabel untuk melatih model (Reddy, Pulabaigari, & B, 2018). Karena itu, hal ini dapat melengkapi kekurangan dari *supervised learning* yang tidak bisa memanfaatkan data tidak berlabel dan secara bersamaan dapat dilatih untuk *task* yang spesifik. Tujuan dari *semi-supervised learning* agar diharapkan model mendapatkan performa yang lebih baik dengan memanfaatkan data tidak berlabel dibandingkan hanya menggunakan data berlabel saja (Ouali, Hudelot, & Tami, 2020).

Penelitian ini menggunakan ketiga paradigma yang sudah dijelaskan untuk menyelesaikan permasalahan klasifikasi domain spesialisasi dokter. Menurut Tsoumacas, Katakis, dan Vlahavas (2006), terdapat dua jenis klasifikasi berdasarkan jumlah label yang diberikan. Ketika label yang diberikan hanya satu saja, maka disebut sebagai klasifikasi *single-label*. Sedangkan klasifikasi disebut *multi-label* ketika jumlah label yang diberikan untuk suatu masukan dapat lebih dari satu. Hal ini sesuai seperti contoh sampel pada Gambar 1.1, di mana terdapat lebih dari satu jenis dokter spesialis. Oleh karenanya, permasalahan domain spesialisasi dokter merupakan permasalahan klasifikasi *multi-label*.

2.2 Klasifikasi *Multi-Label*

Seperti yang sudah dipaparkan pada Subbab 1 (halaman 1), penelitian ini bertujuan untuk mengembangkan model klasifikasi spesialisasi dokter secara *multi-label*. Guna memberikan dasar pengetahuan terhadap permasalahan ini, Subbab 2.2 membahas lebih detail tentang klasifikasi *multi-label* dan metode-metode yang umum digunakan untuk tugas klasifikasi *multi-label*.

Berdasarkan penjelasan Subbab 2.1 (halaman 8), terdapat perbedaan antara klasifikasi

single-label dengan *multi-label*. Klasifikasi *single-label* merupakan proses pemberian sebuah label $y_i \in L$ untuk sebuah masukan x_i dari sekumpulan kelas $L = \{l_1, l_2, \dots, l_j\}$. Sedangkan klasifikasi *multi-label* dapat memiliki label lebih dari satu yang dinotasikan oleh M. L. Zhang, Li, Liu, dan Geng (2018) sebagai:

$$y_i = [y_i^{(1)}, y_i^{(2)}, \dots, y_i^{|L|}],$$

dengan $y_i^{(j)} \in \{0, 1\}$ adalah sebuah indikator apakah sampel x_i terasosiasi dengan label l_j pada L . Hal ini dapat dicontohkan pada Tabel 2.1 sebagai *dataset* klasifikasi *multi-label*.

Tabel 2.1: Contoh *dataset* klasifikasi *multi-label*. Masukan berupa x dan terdapat tiga label y yang harus diprediksi.

x	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$
x_1	1	0	0
x_2	0	1	1
x_3	1	0	1

Meskipun perbedaannya hanya terdapat pada jumlah label yang diberikan, beberapa algoritma *machine learning* yang dapat menyelesaikan permasalahan *single-label* tidak dapat digunakan untuk permasalahan *multi-label*. Karena itu, perlu dilakukan transformasi data yang awalnya merupakan permasalahan klasifikasi *multi-label* menjadi permasalahan klasifikasi *single-label* (Cherman, Monard, & Metz, 2011). Metode transformasi yang bisa diterapkan seperti *binary relevance*, *chain classifier*, dan *label powerset*. Dengan begitu, algoritma *machine learning* yang awalnya khusus untuk permasalahan klasifikasi *single-label* dapat digunakan untuk menyelesaikan permasalahan klasifikasi *multi-label*.

2.2.1 Binary Relevance (BR)

Binary relevance (BR) merupakan metode transformasi permasalahan yang paling sederhana dan mudah diimplementasi. BR dijalankan dengan melakukan klasifikasi dari masing-masing kelas $y_i^{(j)}$ menggunakan masukan x_i yang sama. Oleh karena itu, setiap kelas yang diprediksi memiliki model *machine learning* yang berbeda dan dilatih secara independen. Metode tersebut membuat permasalahan yang awalnya *multi-label* menjadi

single-label karena setiap model hanya memprediksi satu kelas saja. Namun hal tersebut membuat metode BR mengabaikan informasi dari hasil klasifikasi label yang lain (Luaces, Díez, Barranquero, del Coz, & Bahamonde, 2012).

Pada transformasi BR, permasalahan *multi-label* ditransformasi menjadi $|L|$ buah permasalahan klasifikasi biner. Setiap permasalahan *single-label* yang dihasilkan melibatkan *dataset* baru, dinyatakan dengan :

$$D_j = \{(x_1, y_1^{(j)}), (x_2, y_2^{(j)}), \dots, (x_n, y_n^{(j)})\}, \quad (2.1)$$

yang independen dengan *dataset* lainnya seperti pada Tabel 2.2. Masing-masing *dataset* baru digunakan untuk melatih model yang dapat mengklasifikasikan apakah data x_i termasuk ke kelas $y_i^{(j)}$ atau tidak.

Tabel 2.2: Contoh hasil transformasi *dataset* dengan metode BR berdasarkan data pada Tabel 2.1.

(a) Dataset pertama memprediksi $y^{(1)}$		(b) Dataset kedua memprediksi $y^{(2)}$		(c) Dataset ketiga memprediksi $y^{(3)}$	
x	$y^{(1)}$	x	$y^{(2)}$	x	$y^{(3)}$
x_1	1	x_1	0	x_1	0
x_2	0	x_2	1	x_2	1
x_3	1	x_3	0	x_3	1

Dengan menerapkan metode tersebut, M. L. Zhang et al. (2018) mengatakan bahwa terdapat kelebihan dan kekurangan dari metode BR. Salah satu kelebihannya, mudah untuk diterapkan dan dapat mengatasi kasus label yang hilang atau tidak dianotasi pada *dataset*. Sebab setiap label independen menyebabkan label yang hilang tidak memengaruhi hasil klasifikasi label lainnya. Kekurangan yang terdapat dari metode BR seperti tidak bisa menghasilkan performa yang baik ketika *dataset* yang digunakan memiliki jumlah kategori yang tidak seimbang atau timpang. Sifat BR yang independen antar kategori juga membuat BR mengabaikan korelasi antar label yang ada pada data latih (Read, Pfahringer, Holmes, & Frank, 2011).

2.2.2 Chain Classifier (CC)

Sama seperti BR, *chain classifier* (CC) juga merupakan metode transformasi permasalahan dari klasifikasi *multi-label* ke klasifikasi *single-label*. Implementasi juga

tidak berbeda jauh seperti BR, yaitu membuat model sebanyak kelas yang diklasifikasi. Namun perbedaannya terdapat pada masukan yang diterima oleh model. Pada CC, kelas yang sudah diklasifikasi sebelumnya, ikut menjadi masukan pada model yang memprediksi kelas selanjutnya (Read, Pfahringer, Holmes, & Frank, 2021). Oleh sebab itu, notasi persamaan transformasi CC hanya berbeda pada masukan saat melakukan klasifikasi $y_i^{(j)}$. Berbeda dengan Persamaan 2.1, dalam kasus CC, dataset D_j dapat direpresentasikan sebagai:

$$\begin{aligned} D_j = & \{(\{x_1, y_1^{(1)}, y_1^{(2)}, \dots, y_1^{(j-1)}\}, y_1^{(j)}), \\ & (\{x_2, y_2^{(1)}, y_2^{(2)}, \dots, y_2^{(j-1)}\}, y_2^{(j)}), \\ & \vdots \\ & (\{x_n, y_n^{(1)}, y_n^{(2)}, \dots, y_n^{(j-1)}\}, y_n^{(j)})\}. \end{aligned} \quad (2.2)$$

Contoh dari transformasi CC juga dapat dilihat lebih jelas pada Tabel 2.3

Tabel 2.3: Contoh hasil transformasi dataset dengan metode CC berdasarkan data pada Tabel 2.1.

(a) Dataset pertama memprediksi $y^{(1)}$		(b) Dataset kedua memprediksi $y^{(2)}$		(c) Dataset ketiga memprediksi $y^{(3)}$		
x	$y^{(1)}$	x	$y^{(1)}$	$y^{(2)}$	x	$y^{(1)}$
x_1	1	x_1	1	0	x_1	1
x_2	0	x_2	0	1	x_2	0
x_3	1	x_3	1	0	x_3	1

Dengan menjadikan hasil klasifikasi dari model sebelumnya sebagai masukan untuk model berikutnya, CC dapat mempertimbangkan hubungan antar label yang ada. (Read et al., 2011). Cara tersebut membuat CC dapat menutupi kekurangan yang terdapat pada BR dalam hal korelasi antar label. Akibatnya, urutan kategori yang diklasifikasi terlebih dahulu menjadi faktor yang penting untuk meningkatkan performa model yang menggunakan CC.

2.2.3 Label Powerset (LP)

Pada pembahasan sebelumnya, BR dan CC mengubah klasifikasi *multi-label* menjadi klasifikasi biner *single-label*. Maka terdapat metode lain bernama *label powerset* (LP)

yang mengubah klasifikasi *multi-label* menjadi klasifikasi *single-label* dengan kelas lebih dari dua atau yang disebut sebagai *multi-class* (Junior, Faria, Silva, & Cerri, 2017). Transformasi LP dilakukan dengan mengubah semua kemungkinan pasangan kelas yang muncul bersamaan menjadi kelas yang baru. Dengan demikian kelas baru yang dihasilkan setelah transformasi menjadi $2^{|L|}$ kelas. Kelas LP yang baru menjadi $L_{LP} = \{y_{LP}^{(1)}, y_{LP}^{(2)}, \dots, y_{LP}^{(m)}\}$ dengan m adalah jumlah kombinasi pasangan label yang dapat dibentuk dari L dan muncul secara bersamaan. Akibatnya, *dataset* yang sudah ditransformasi dapat dituliskan sebagai Persamaan berikut:

$$D = \{(x_1, y_{LP_1}), (x_2, y_{LP_2}), \dots, (x_n, y_{LP_n})\}. \quad (2.3)$$

Selanjutnya dicontohkan bagaimana penerapan transformasi LP terhadap contoh *dataset* pada Tabel 2.1. Transformasi LP mengubah *dataset* yang awalnya berjumlah tiga kelas menjadi sembilan kelas, namun hanya satu label saja yang dihasilkan. Untuk kombinasi pasangan kelas yang pada mulanya $(y^{(1)}, y^{(2)}, y^{(3)})$, transformasi LP memetakan pasangan kelas seperti:

$$\begin{array}{ll} y_{LP}^{(1)}, = (0, 0, 0) & y_{LP}^{(4)} = (0, 1, 1), \\ y_{LP}^{(2)}, = (0, 0, 1) & \vdots \\ y_{LP}^{(3)}, = (0, 1, 0) & y_{LP}^{(9)} = (1, 1, 1). \end{array}$$

Dengan demikian, *dataset* hasil transformasi dapat dilihat pada Tabel 2.4.

Tabel 2.4: Contoh *dataset* hasil transformasi LP. Masukan berupa x dan terdapat sembilan kelas y_{LP} yang menjadi target prediksi.

x	$y_{LP}^{(1)}$								
x_1	0	0	0	0	1	0	0	0	0
x_2	0	0	0	1	0	0	0	0	0
x_3	0	0	0	0	0	1	0	0	0

2.3 Algoritma Klasifikasi

Berdasarkan yang sudah dijelaskan pada Subbab 1 (halaman 1), penelitian ini membuat model *machine learning* yang dapat melakukan klasifikasi spesialisasi dokter secara otomatis. Dalam upaya memberikan dasar pemahaman yang solid terhadap permasalahan ini, Subbab 2.3 membahas lebih detail tentang algoritma *machine learning* yang digunakan untuk klasifikasi domain spesialis dokter.

2.3.1 Logistic Regression

Logistic Regression merupakan salah satu varian dari *linear model* yang yang dikhusruskan untuk klasifikasi. *Linear model* merupakan sebuah metode untuk memprediksi nilai \hat{y} dengan asumsi bahwa nilai y dapat dihasilkan melalui persamaan *linear* dari fitur x (Pedregosa et al., 2011). Secara matematis *Logistic Regression* dituliskan sebagai berikut:

$$\hat{y} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_px_p. \quad (2.4)$$

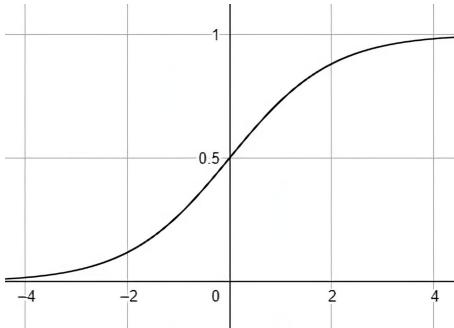
Pada Persamaan 2.4, vektor $w = (w_0, w_1, \dots, w_p)$ adalah bobot model dan vektor $x = (x_1, x_2, \dots, x_p)$ sebagai fitur dari masukan. Namun model tersebut masih berupa regresi, agar dapat berubah menjadi klasifikasi maka perlu memanfaatkan fungsi logistik yang membuat hasil keluaran antara nol hingga satu seperti pada Gambar 2.1. Dengan memanfaatkan fungsi logistik, Persamaan 2.4 menjadi seperti berikut:

$$f(x) = \sigma(w_0 + w_1x_1 + \cdots + w_px_p) = \frac{1}{1 + \exp(-w_0 - w_1x_1 - \cdots - w_px_p)}. \quad (2.5)$$

Pada Persamaan 2.5, $f(x)$ menghasilkan nilai dalam rentang $(0, 1)$ dan merepresentasikan probabilitas seberapa mungkin x berasal dari kelas positif.

2.3.2 Support Vector Machine (SVM)

Support vector machine (SVM) merupakan sebuah algoritma *machine learning* yang dapat digunakan untuk permasalahan klasifikasi dan regresi. SVM bekerja dengan



Gambar 2.1: Contoh grafik dari fungsi logistik $\sigma(x) = \frac{1}{1+e^{-x}}$

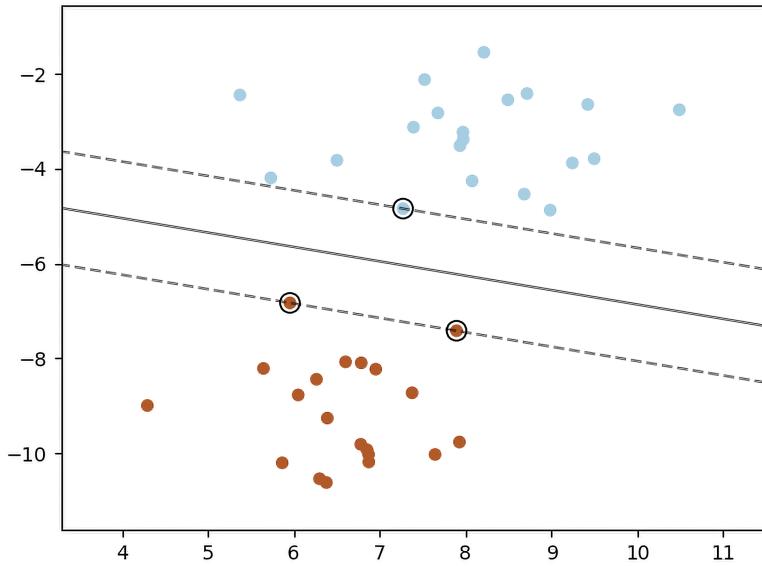
mencari sebuah *support vector* yang dapat memisahkan data menjadi 2 kelas yang berbeda (Pedregosa et al., 2011) seperti pada Gambar 2.2. Pada Gambar 2.2, dapat dilihat bahwa terdapat satu vektor yang memisahkan data berwarna biru dan merah, hal itu adalah *support vector*. Pemanfaatan SVM untuk klasifikasi dinamakan *support vector classification* (SVC) yang digunakan pada permasalahan di penelitian ini. Dalam implementasi SVC oleh Pedregosa et al. (2011), perumusan prediksi suatu data x dirumuskan sebagai:

$$f(x) = \sum_{i \in SV} y_i \alpha_i K(x_i, x) + b. \quad (2.6)$$

Pada Persamaan 2.6, $f(x)$ merupakan hasil prediksi untuk data x dan i adalah indeks dari *support vector* (SV). Oleh karena itu, y_i adalah label dari *support vector* tersebut dan α_i sebagai bobot yang dari *support vector*. Selanjutnya terdapat $K(x_i, x)$ yang merupakan fungsi *kernel* untuk mengukur kesamaan antara data x_i dan x pada ruang vektor yang lebih tinggi dan variabel terakhir b sebagai bias untuk mencegah *overfit*. Dengan menggunakan rumus ini, SVC dapat mengklasifikasikan data baru berdasarkan posisi relatifnya terhadap *support vector* yang telah ditentukan.

2.3.3 Stochastic Gradient Descent (SGD)

Pada algoritma SVM dan *logistic regression*, terdapat proses optimisasi atau *gradient descent* yang dilakukan pada saat proses melatih model untuk mendapatkan parameter yang cocok pada *machine learning*. Terdapat banyak varian dari algoritma *gradient descent* dan salah satunya adalah *stochastic gradient descent* (SGD). Pada algoritma *gradient descent*, optimisasi baru dilakukan setelah menghitung *loss* dari keseluruhan



Gambar 2.2: *Support vector* melakukan klasifikasi dengan memisahkan *dataset* yang berwarna biru dan merah. Jika sebuah *instance* baru berada di area merah, maka *instance* tersebut diklasifikasi sebagai warna merah. Gambar ini didapatkan dari Pedregosa et al. (2011)

data. Sedangkan algoritma SGD, proses optimisasi dilakukan untuk setiap data yang diberikan saat melatih model (Ruder, 2016). Hal ini membuat SGD lebih efisien dibandingkan algoritma *gradient descent* karena tidak membutuhkan banyak komputasi.

2.3.4 Decision Tree

Terdapat juga algoritma *machine learning* yang bekerja seperti pohon yang bercabang. Algoritma tersebut bernama *decision tree* yang membuat pohon yang terdiri dari *node* dan daun. Untuk sebuah *node* m , berisi fitur x_j dan *threshold* t_m dari fitur tersebut. Fitur dan *threshold* tersebut menjadi parameter θ yang perlu dicari nilai optimalnya. Parameter yang dihasilkan membagi (*split*) data Q_m yang merupakan set data yang terdapat pada *node* menjadi dua bagian. Hasil pembagian menghasilkan dua set data lainnya yang disebut daun kiri Q_m^{left} dan daun kanan Q_m^{right} seperti pada Persamaan 2.7 dan Persamaan 2.8 berikut

$$Q_m^{left} = \{(x, y) | x_j \leq t_m\}, \quad (2.7)$$

$$Q_m^{right} = Q_m \setminus Q_m^{left}. \quad (2.8)$$

Hasil *split* tersebut perlu diketahui apakah merupakan *split* terbaik atau tidak. Karena itu, perlu menghitung nilai *impurity* atau *loss H* yang menilai kualitas *split* dari masing-masing daun yang dihasilkan. Nilai *loss H* untuk permasalahan regresi dan klasifikasi berbeda, namun pada penelitian ini menggunakan fungsi *loss Gini* yang dituliskan sebagai:

$$I(y = k) = \begin{cases} 1, & \text{if } y = k \\ 0, & \text{otherwise} \end{cases}, \quad (2.9)$$

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k), \quad (2.10)$$

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk}), \quad (2.11)$$

dengan n_m merupakan banyaknya *instance* pada *subset* Q_m dan p_{mk} sebagai proporsi banyaknya *instance* dengan kelas k pada *subset* Q_m . Gabungan dari seluruh *loss H* yang ada pada daun disebut *information gain G*. Oleh karena itu, *decision tree* mencari parameter θ yang dapat meminimalkan *information gain G*:

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)), \quad (2.12)$$

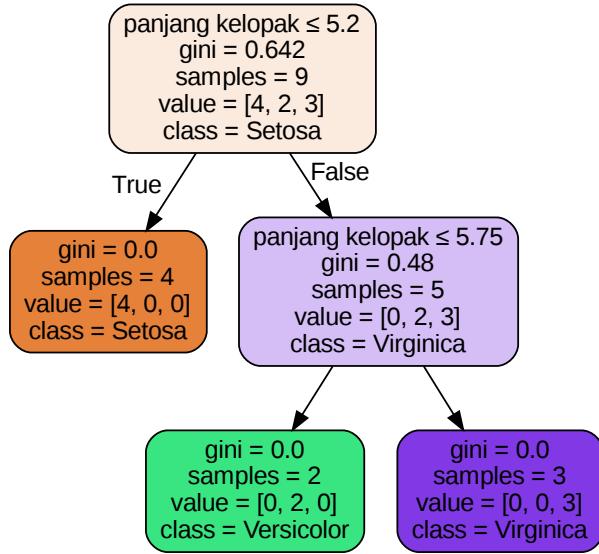
$$\theta^* = \arg \min_{\theta} G(Q_m, \theta), \quad (2.13)$$

dengan n_m^{left} sebagai banyaknya *instance* pada Q_m^{left} dan n_m^{right} sebagai jumlah *instance* pada Q_m^{right} .

Setelah pohon berhasil dibuat, maka tahapan selanjutnya adalah memprediksi data baru. Masukan pertama kali diberikan ke *root node* dan diarahkan sesuai dengan θ pada *node* tersebut ke salah satu cabang. Proses tersebut terus berlanjut hingga masukan sampai ke salah satu daun yang ada. Hasil prediksi dari masukan sesuai dengan label yang ada pada daun tersebut. Contoh *decision tree* dapat dilihat pada Gambar 2.3.

2.3.5 Extreme Gradient Boosting (XGBoost)

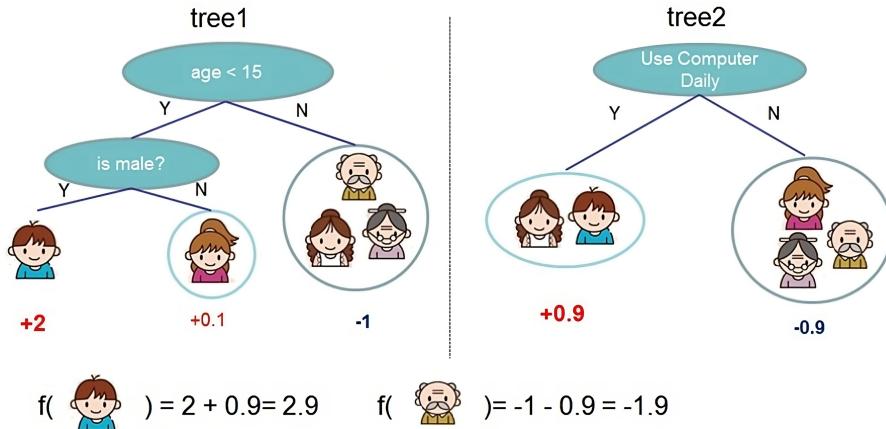
Selanjutnya terdapat *extreme gradient boosting* (XGBoost) yang merupakan salah satu algoritma yang menerapkan konsep *gradient boosting* pada *decision tree*.



Gambar 2.3: Contoh *decision tree* yang mengklasifikasi sesuai dengan kondisi yang berlaku

Menurut Schapire (1999), algoritma *boosting* merupakan algoritma yang menggabungkan beberapa model sederhana (*weak learner*) untuk meningkatkan akurasi dibandingkan satu *weak learner* saja. *Weak learner* yang dibuat setelahnya bertugas untuk memprediksi seberapa besar eror yang dihasilkan oleh *weak learner* sebelumnya. Setelah itu dikembangkan algoritma *gradient boosting* dengan mengubah hal yang diprediksi oleh *weak learner*. Jika *weak learner* memprediksi eror dari *weak learner* sebelumnya, maka *weak leaner* pada *gradient boosting* memprediksi *negative gradient* dari *loss* yang dihasilkan *weak learner* sebelumnya (Natekin & Knoll, 2013). Setelah itu T. Chen dan Guestrin (2016) mengembangkan lagi sebuah algoritma bernama XGBoost yang lebih baik dari *gradient boosting* dari segi kecepatan dalam melatih model serta efisiensi *resource* yang dibutuhkan. *Weak learner* yang digunakan pada XGBoost adalah *decision tree* yang sudah dibahas pada Subbab 2.3.4 (halaman 16).

Berdasarkan penelitian T. Chen dan Guestrin (2016), algoritma XGBoost dapat dijelaskan melalui persamaan-persamaan berikut. Sebelumnya ditentukan *dataset* D yang terdiri dari pasangan masukan x dan label y sebanyak n *instance*. Model XGBoost dengan menggunakan K pohon menerima masukan x_i dan menghasilkan prediksi \hat{y} . Setiap pohon ke- k pada setiap iterasi diberi label f_k , dan F merupakan kumpulan pohon yang terbentuk. Prediksi yang dihasilkan diperoleh dengan menjumlahkan $f_k(x_i)$ dari



Gambar 2.4: Ilustrasi bagaimana hasil prediksi dari model $F(x)$ didapatkan berdasarkan Persamaan 2.14.
Gambar diperoleh dari T. Chen dan Guestrin (2016)

keseluruhan pohon yang dibuat seperti pada Gambar 2.4. Dengan demikian, prediksi \hat{y}_i dapat dituliskan sebagai:

$$F(x) = \hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in F. \quad (2.14)$$

Untuk mengetahui apakah hasil prediksi XGBoost sudah baik atau belum, perlu dirumuskan fungsi *loss* yang dapat menentukan hal tersebut. Fungsi *loss* L pada model XGBoost terdiri dari dua bagian, yaitu *training loss* l dan regularisasi Ω . Dalam bentuk persamaan, fungsi *loss* dapat dituliskan sebagai:

$$L = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k). \quad (2.15)$$

Regularisasi $\Omega(f_k)$ merupakan salah satu improvisasi yang diterapkan pada XGBoost untuk meningkatkan performa terutama saat jumlah data yang besar (Osman, Ahmed, Chow, Huang, & El-Shafie, 2021). Persamaan regularisasi pada XGBoost dapat dituliskan sebagai berikut:

$$\Omega(f_k) = \gamma T + \frac{1}{2} \lambda |w|^2. \quad (2.16)$$

Pada Persamaan 2.16, T mengindikasikan jumlah daun dalam pohon, w merupakan

bobot yang ada pada pohon, γ sebagai konstanta yang mengatur kompleksitas pohon yang dihasilkan, dan λ adalah skala penalti atau nilai regularisasi.

Seperti yang sudah dijelaskan tentang algoritma *gradient boosting*, setiap pohon yang dibangun selanjutnya memprediksi *residual* eror atau perbedaan probabilitas klasifikasi dengan label sebenarnya yang dihasilkan oleh pohon sebelumnya. Dengan demikian, kesalahan yang dihasilkan oleh pohon sebelumnya semakin berkurang. Pada iterasi ke- t , fungsi *loss* pada iterasi tersebut, $L^{(t)}$, dapat dituliskan sebagai:

$$L^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t). \quad (2.17)$$

Perhatikan bahwa $\hat{y}_i^{(t-1)}$ adalah prediksi pada iterasi sebelumnya, dan $f_t(x_i)$ adalah prediksi yang dihasilkan oleh pohon ke- t . Untuk mempercepat perhitungan, dilakukan aproksimasi nilai *loss* menggunakan ekspansi deret Taylor orde 2, dengan mengabaikan bagian *training loss* yang tidak mengandung $f_t(x_i)$. Dalam hal ini, persamaan *loss* pada iterasi ke- t dapat disederhanakan menjadi:

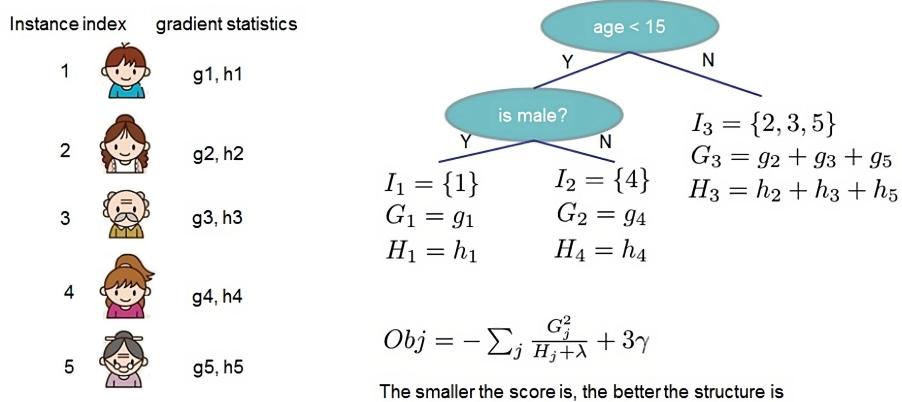
$$L^{(t)} = \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t), \quad (2.18)$$

dengan g_i dan h_i adalah turunan parsial dari fungsi *loss* terhadap $\hat{y}_i^{(t-1)}$.

Selanjutnya, ditentukan *instance* $I_j = \{i | q(x_i) = j\}$ dengan q adalah struktur pohon yang dilalui oleh x_i hingga sampai ke daun j . Ilustrasi *instance* I_j dapat dilihat pada Gambar 2.5. Maka dengan mempertimbangkan *instance* I_j , persamaan *loss* dapat diubah sebagai:

$$L^{(t)} = \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T. \quad (2.19)$$

Berdasarkan Persamaan 2.19, dapat ditemukan w_j^* sebagai bobot optimal dari daun j dan $L(q)$ yang merupakan *loss* dari struktur pohon q .



Gambar 2.5: Ilustrasi bagaimana XGBoost mencari bobot w_j^* yang membagi data x menjadi *instance* I_j yang meminimalkan *loss*. Gambar diperoleh dari T. Chen dan Guestrin (2016)

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \quad (2.20)$$

$$L^t(q) = -\frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \quad (2.21)$$

Dengan menggunakan Persamaan 2.21 yang telah ditemukan, dapat dilakukan pencarian *split* terbaik pada pohon q . *Split* terbaik adalah *split* yang menghasilkan nilai *loss* terendah. Setiap *split* membagi *instance* I menjadi dua bagian I_L dan I_R , sehingga $I = I_L \cup I_R$. Persamaan yang merepresentasikan *loss* yang dihasilkan oleh *split* dapat dituliskan sebagai berikut:

$$L_{\text{split}} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} + \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma. \quad (2.22)$$

Dengan demikian, XGBoost menggunakan fungsi *loss* dan regularisasi yang telah dijelaskan di atas untuk membangun pohon-pohon secara berurutan dan meningkatkan performa model pada setiap iterasinya. Menurut penelitian T. Chen dan Guestrin (2016), ditemukan bahwa perubahan tersebut berguna untuk mencegah *overfitting* dan dapat mempercepat proses komputasi.

2.3.6 Multinomial Naive Bayes

Model terakhir yang digunakan adalah algoritma *naive bayes*. Algoritma ini mengimplementasikan teorema bayes dengan mengasumsikan setiap fitur memiliki pengaruh yang independen dalam memprediksi data. Salah satu variasi dari algoritma *naive bayes* yang sering digunakan untuk melakukan klasifikasi teks adalah *naive bayes multinomial* (S. Xu, Li, & Wang, 2017). Berdasarkan implementasi yang dibuat oleh Pedregosa et al. (2011), hasil prediksi dari algoritma *naive bayes multinomial* sebagai berikut:

$$f(x) = \arg \max_y P(y) \prod_{i=1}^n P(x_i|y), \quad (2.23)$$

$$P(x_i|y) = \frac{N_{yi} + \alpha}{N_y + \alpha n}. \quad (2.24)$$

Dengan suatu data $x = x_1, x_2, \dots, x_i$, mencari $P(y)$ sebagai probabilitas kelas y dan probabilitas dari masing-masing kata yang ada pada data x . Untuk mencari probabilitas dari suatu kata, dibutuhkan N_{yi} sebagai banyak kata x_i pada data latih dan N_y sebagai total kata yang ada pada data latih. Nilai α dilakukan untuk melakukan *smoothing* agar tidak ada hasil perkalian yang menjadi nol (Pedregosa et al., 2011).

2.4 Pengolahan teks

Seperti pembahasan pada Subbab 1 (halaman 1), masukan dari model *machine learning* merupakan sebuah data teks yang berasal dari forum tanya jawab. Maka pada Subbab 2.4 (halaman 1), membangun pemahaman terhadap metode pengolahan teks yang diterapkan pada penelitian ini.

Menurut penelitian Kadhim (2018), terdapat beberapa metode prapemrosesan teks yang dapat digunakan. Yaitu penghapusan *stopword*, mengubah menjadi kata dasar (*stemming*), dan memecah menjadi beberapa token (*tokenisasi*). Tujuan dari tahapan ini supaya menghilangkan informasi yang tidak berguna, sehingga berkurangnya *noise* pada data teks.

Untuk sebuah token yang dibentuk, dapat terdiri dari satu sampai N kata yang berurutan, disebut juga sebagai n -gram kata. Ketika menggunakan 1-gram, maka setiap kata yang ada pada kalimat menjadi sebuah token. Hal yang sama berlaku juga untuk

2-gram. Contohnya untuk sebuah kalimat "hello world halo dunia", didapatkan tiga token, yaitu "hello world", "world halo", dan "halo dunia". Penggunaan n-gram yang lebih besar dapat membantu dalam memperoleh konteks yang lebih kaya dalam teks yang sedang diproses. Selain itu, n-gram juga dapat digunakan sebagai *language model* dan klasifikasi teks.

Setelah teks selesai diproses, dilanjutkan ke tahap *embedding*, yaitu mengubah teks ke dalam format numerik yang terstruktur. Hal ini disebabkan model *machine learning* tidak bisa menerima data yang tidak terstruktur. Salah satu metode *embedding* yang dilakukan adalah *term frequency - inverse document frequency* (TF-IDF). TF-IDF mengubah data teks menjadi vektor berdimensi sebanyak jumlah token unik yang ada pada dokumen. Isi dari vektor tersebut dapat diinterpretasikan sebagai skor dari suatu token dalam dokumen yang dapat dihitung sebagai:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D), \quad (2.25)$$

$$\text{TF}(t, d) = \frac{\text{Jumlah kemunculan token } t \text{ dalam dokumen } d}{\text{Total token dalam dokumen } d}, \quad (2.26)$$

$$\text{IDF}(t, D) = \log \left(\frac{\text{Total dokumen dalam } D}{\text{Total dokumen yang mengandung token } t \text{ dalam } D} \right). \quad (2.27)$$

Semakin besar dokumen yang digunakan, maka lebih beragam juga token unik yang muncul. Dengan begitu, dimensi yang didapatkan dari proses *embedding* TF-IDF juga bertambah tinggi. Dimensi yang terlalu tinggi dapat menimbulkan masalah berupa *collinearity*, model tidak stabil, dan *overfitting* (Verleysen & François, 2005). Salah satu metode yang dapat mengurangi dimensi tanpa menghilangkan informasi adalah *single value decomposition* (SVD). Berdasarkan Klema dan Laub (1980), SVD merupakan faktorisasi matriks A menjadi matriks $U\Sigma V^T$ yang salah satu kegunaannya untuk mereduksi dimensi. Akibatnya, SVD digunakan pada penelitian ini untuk mereduksi dimensi hasil *embedding* sebelum dimasukkan ke dalam model.

Metode *embedding* dengan TF-IDF tidak dapat menangkap konteks dari sebuah kalimat. Hal tersebut membuat nilai dari kata "bisa" sebagai kata kerja memiliki nilai yang sama dengan kata "bisa" sebagai kata benda. Maka digunakan *language model* berbasis *transformers* yang dapat menangkap konteks dari sebuah kalimat. Oleh sebab itu, penelitian ini menggunakan *deep learning* berupa *language model* seperti BERT untuk melakukan klasifikasi domain spesialisasi dokter.

2.5 Pre-Trained Language Model

Berdasarkan yang sudah dijelaskan pada Subbab 2.4 (halaman 22), penelitian ini juga memanfaatkan *language model* yang dapat menangkap konteks dari kalimat yang dimasukkan. Untuk memberikan landasan pemahaman dari topik ini, Subbab 2.5 membahas lebih detail tentang *language model* yang digunakan, beserta pengembangan *language model* dengan paradigma *unsupervised* dan *semi-supervised learning*.

Language model merupakan sebuah model yang memberikan probabilitas dari urutan kata dan dapat menentukan probabilitas untuk kata selanjutnya. *Language model* juga dapat digunakan untuk permasalahan klasifikasi dengan memprediksi sebuah kelas berdasarkan kata yang menjadi fitur. Berdasarkan penjelasan pada Papers With Code¹, terdapat dua jenis metode yang umum digunakan dalam membuat *language model*. Yaitu pendekatan n-gram dan *neural network*. Pada penelitian ini, digunakan *language model* berbasis *neural network* untuk menyelesaikan klasifikasi domain spesialisasi dokter.

2.5.1 Bidirectional Encoder Representations from Transformers (BERT)

Sesuai dengan namanya, *bidirectional encoder representations from transformers* (BERT) merupakan sebuah *pre-trained language model* yang memanfaatkan *multi-layer bidirectional encoder* yang terdapat pada *transformer* (Devlin et al., 2019). *Encoder* merupakan arsitektur yang diusulkan oleh Vaswani et al. (2017) yang menerapkan konsep *multi-head self-attention*. Secara garis besar, *self-attention* merupakan teknik *embedding* data *sequence* yang dapat membuat model untuk fokus pada bagian-bagian penting dari data *sequence* yang diberikan. Teknik ini dapat memberikan representasi yang tepat meskipun jarak antar kedua *sequence* jauh. Hal tersebut membuat *self-attention* lebih baik dibanding teknik *auto-regressive* seperti LSTM yang dapat kehilangan informasi dari *sequence* sebelumnya jika terlalu jauh.

Masukan yang diterima oleh BERT tidak berupa *string* kalimat, namun merupakan vektor token yang berasal dari teks yang sudah ditokenisasi. Sebelumnya, terdapat beberapa token spesial yang dihasilkan pada proses tokenisasi, yaitu [CLS], [SEP], [MASK], dan [PAD]. Token [CLS] selalu berada pada indeks pertama pada vektor token, [SEP] merupakan token yang membatasi ketika terdapat dua kalimat yang berbeda,

¹<https://paperswithcode.com/task/language-modelling>

```

Kalimat : Saya pusing dok. apa obatnya?
Tokenisasi : "[CLS]", "saya", "pusin", "dok.", "apa", "obatnya?", "[SEP]",
            "[PAD]"
Kalimat : ank saya kens copid
Tokenisasi : "[CLS]", "ank", "saya", "ken", "#s", "cop", "#id", "[PAD]"

```

Gambar 2.6: Contoh kalimat dan hasil tokenisasi teks dengan panjang vektor delapan token

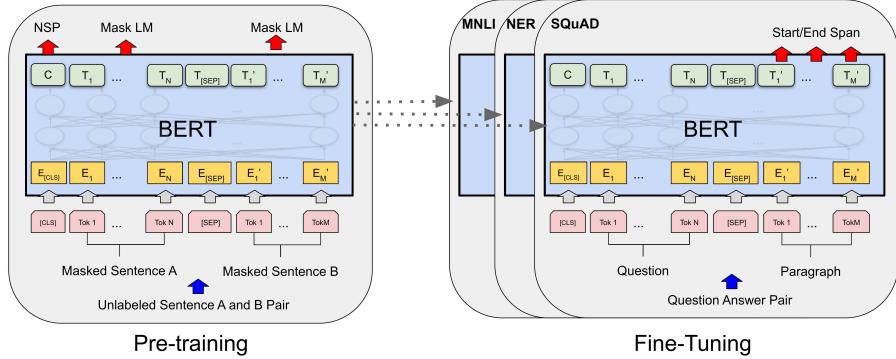
token [MASK] untuk menandakan token yang perlu diprediksi, dan token [PAD] digunakan untuk *padding* ketika panjang vektor kurang dari yang ditentukan oleh model BERT. Selain dari token spesial, proses tokenisasi juga dapat mengatasi untuk kata yang tidak dikenali atau *typo* dengan memecah kata yang tidak dikenali menjadi beberapa kata yang dikenali. Kata yang dipecah ditandai dengan token "##" yang menandakan bahwa token tersebut terhubung dengan token sebelumnya. Pada penerapannya, dapat dilihat Gambar 2.6 yang mencontohkan penerapan tokenisasi menggunakan model dari IndoBERT-base-p1².

Sebelum menggunakan model BERT, terdapat 2 tahap pelatihan yang harus dilakukan, yaitu tahap *pre-training* dan *fine-tuning*. Pada tahap *pre-training*, BERT dilatih untuk menyelesaikan 2 *task* secara bersamaan. Kedua *task* itu adalah *masked language model* (MLM) dan *next sentence prediction* (NSP). Dalam menyelesaikan *task* tersebut, BERT dilatih dengan metode *unsupervised learning* karena hanya menggunakan korpus teks yang tidak dianotasi. Setelah itu, pada tahap *fine-tuning* dilatih secara *supervised learning* untuk *task* yang lebih spesifik seperti klasifikasi teks.

Task MLM dijalankan dengan memberikan mask kepada beberapa token masukan. Selanjutnya BERT melakukan klasifikasi untuk menentukan token sebenarnya dari token yang diberikan *mask*. Sedangkan pada tahap NSP, BERT menerima dua kalimat A dan B sebagai masukan yang dipisahkan dengan token [SEP]. Setelah itu BERT menentukan apakah kalimat B merupakan kelanjutan dari kalimat A atau tidak. Hasil klasifikasi dari NSP didapatkan dari vektor pertama dari keluaran atau vektor keluaran yang berkorespondensi dengan masukan token [CLS].

Selanjutnya pada tahap *fine-tuning*, keluaran dari BERT dihubungkan dengan *feed-forward neural network* agar dapat melakukan *task* lain. Beberapa *task* yang sudah dilakukan pada penelitian Devlin et al. (2019) dengan *fine-tuning* BERT berupa klasifikasi teks, tanya jawab, dan *task* NLP lainnya. Saat dilakukan *fine-tuning*, model yang sudah dilatih pada tahap *pre-training*, dilatih kembali dengan *dataset* yang baru

²<https://huggingface.co/indobenchmark/indobert-base-p1>



Gambar 2.7: Tok merupakan token dari kalimat masukan dan diubah menjadi E sebagai vektor embedding dari token yang diberikan. Keluaran dari model BERT merupakan vektor T yang digunakan untuk menyelesaikan *task* baik pada tahap *pre-trainig* maupun *fine-tuning*. Gambar ini didapatkan dari Devlin et al. (2019)

dan lebih spesifik dengan *task* yang diinginkan. *fine-tuning* dilakukan dengan tujuan agar bobot pada BERT dapat menyesuaikan dengan *task* yang menjadi isu utama. Ilustrasi penggunaan BERT dapat dilihat pada Gambar 2.7

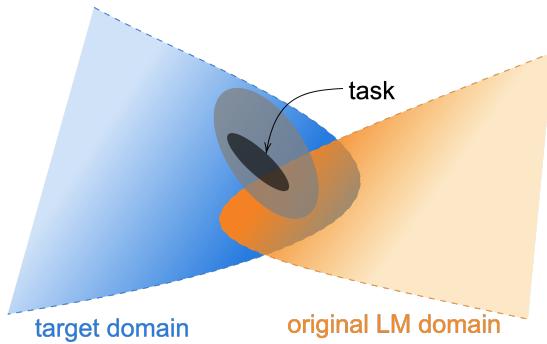
Penelitian ini menggunakan lima model *pre-trained* BERT dengan empat model dilatih dengan bahasa Indonesia dan satu lagi dilatih dengan berbagai bahasa. Kelima model BERT itu adalah IndoNLU-base², IndoNLU-large³, IndoLEM⁴, mBERT⁵, dan IndoNLU-base yang sudah dilatih kembali oleh Halim, Steven Wiryadinata (2022), disebut sebagai IndoWH⁶. Model IndoNLU dan IndoLEM dilatih menggunakan *dataset* teks berbahasa Indonesia yang berbeda. Berdasarkan penelitian Wilie et al. (2020), model IndoNLU dilatih dengan empat miliar kata. Sedangkan model IndoLEM hanya dilatih dengan 220 juta kata (Koto, Rahimi, Lau, & Baldwin, 2020). Untuk model mBERT yang diusulkan oleh Pires, Schlinger, dan Garrette (2019), dilatih menggunakan 104 bahasa yang berbeda. Hal ini membuat mBERT dapat digunakan untuk berbagai bahasa yang berbeda. Model terakhir yang digunakan adalah IndoWH. Model ini dilatih oleh Halim, Steven Wiryadinata (2022) menggunakan *dataset* forum tanya jawab kesehatan berbahasa Indonesia dengan *task* MLM. Berdasarkan yang sudah dibahas, terdapat empat model BERT *base* yang terdiri dari 12 *encoder layers* dan sebuah model BERT *large* yang dirakit dengan 24 *encoder layers*.

²<https://huggingface.co/indobenchmark/indobert-large-p1>

⁴<https://huggingface.co/indolem/indobert-base-uncased>

⁵<https://huggingface.co/bert-base-multilingual-cased>

⁶<https://huggingface.co/stevenwh/indobert-base-p2-finetuned-mer-80k>



Gambar 2.8: Ilustrasi hal yang dilakukan DAPT agar *pre-trained language model* dapat fokus ke domain yang beririsan. Gambar diambil dari penelitian Gururangan et al. (2020)

2.5.2 Domain Adaptive Pre-training (DAPT)

Untuk melatih BERT dari awal membutuhkan *resource* komputasi dan data yang sangat besar. Maka dapat menggunakan pendekatan *domain adaptive pre-training* (DAPT) yang dilakukan secara *unsupervised learning*. Berdasarkan penelitian Gururangan et al. (2020), DAPT dilakukan dengan melatih kembali *language model* menggunakan *dataset* korpus yang memiliki domain atau topik yang sama dengan *dataset* yang diprediksi pada *task* yang lebih spesifik. Proses DAPT dilakukan sebelum memasuki tahap *fine-tuning* pada *language model* yang sudah dilatih sebelumnya. Berdasarkan Gambar 2.8, DAPT dilakukan agar *language model* dapat lebih fokus pada domain yang beririsan antara domain model *pre-trained* sebelumnya dengan domain target. Penelitian yang dilakukan oleh Gururangan et al. (2020) menghasilkan kesimpulan bahwa DAPT dapat meningkatkan akurasi pada model RoBERTa.

2.5.3 Generative Adversarial Networks BERT (GAN-BERT)

Penelitian ini juga menggunakan paradigma pemelajaran secara *semi-supervised learning*. Metode yang digunakan adalah *generative adversarial networks* BERT (GAN-BERT) yang merupakan penerapan *language model* BERT ke dalam *framework* GAN. *Framework* GAN pertama kali diusulkan oleh Goodfellow et al. (2020) yang memanfaatkan dua model yang terdiri dari model generatif G dan model diskriminan D dengan tujuan yang bertolak belakang. Sesuai dengan namanya, model G bertujuan untuk membuat data berdasarkan *noise* z dengan distribusi $p_z(z)$ menjadi sebuah data $G(z)$ yang menyerupai distribusi *dataset* asli $p_{data}(x)$. Sedangkan model D menerima masukan dengan dua sumber data yang berbeda. Yaitu *instance* x yang berasal dari

dataset asli dan *instance* palsu $G(z)$ yang dihasilkan oleh model G . Dalam hal ini model D perlu menentukan apakah masukan yang diberikan berasal dari data asli atau palsu. Hal tersebut ditulis $D(x)$ sebagai besar peluang x berasal dari data asli. Karena itu, *loss* untuk model G dan D bergantung terhadap seberapa besar performa yang didapatkan oleh model D . Hal tersebut dapat dituliskan sebagai persamaan *minimax* $V(G,D)$ yang dirumuskan sebagai berikut:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] , \quad (2.28)$$

dengan $\mathbb{E}_{x \sim p_{data}(x)} f(x)$ dan $\mathbb{E}_{z \sim p_z(z)} f(z)$ merupakan rata-rata dari fungsi f berdasarkan distribusi nilai $p_{data}(x)$ dan $p_z(z)$. Maka dari itu, fungsi objektif $V(G,D)$ ingin memperbesar nilai yang dihasilkan ketika masukan berasal dari data asli, dan sebaliknya memperkecil nilai keluaran ketikan masukan didapatkan dari data palsu.

Framework GAN tersebut pada awalnya hanya dapat melakukan metode *unsupervised learning* di mana tidak menggunakan data berlabel. Maka Salimans et al. (2016) mengusulkan cara agar GAN dapat menerapkan metode *semi-supervised learning* sehingga dapat melakukan klasifikasi terhadap data yang berlabel. Usulan pertama dengan menambahkan keluaran model D menjadi vektor dengan panjang $K+1$:

$$D(x) = \{l_1, l_2, \dots, l_K, l_{K+1}\} , \quad (2.29)$$

dengan K adalah jumlah kelas yang diprediksi dan label ke- $K+1$ (l_{K+1}) untuk menentukan apakah masukan berasal dari data asli atau palsu.

Hal tersebut membuat model D dapat bekerja sebagai diskriminasi dan juga melakukan klasifikasi. Namun karena pendekatan tersebut merupakan *semi-supervised*, maka diusulkan untuk memisahkan perhitungan *loss supervised* dengan *unsupervised*. *loss supervised* merupakan *loss* dari klasifikasi sedangkan *loss unsupervised* merupakan *loss* dari GAN seperti sebelumnya.

$$L = L_{\text{supervised}} + L_{\text{unsupervised}}, \quad (2.30)$$

$$L_{\text{supervised}} = -\mathbb{E}_{x,y \sim p_{\text{data}}(x,y)} \log p_{\text{model}}(y|x, y < K+1), \quad (2.31)$$

$$\begin{aligned} L_{\text{unsupervised}} = & -\{\mathbb{E}_{x \sim p_{\text{data}}(x)} \log [1 - p_{\text{model}}(y = K+1|x)] \\ & + \mathbb{E}_{x \sim G} \log [p_{\text{model}}(y = K+1|x)]\}, \end{aligned} \quad (2.32)$$

dengan $p_{\text{model}}(y|x, y < K+1)$ merupakan probabilitas masukan vektor x sebagai label y_1 hingga y_K . Sedangkan $p_{\text{model}}(y = K+1|x)$ sebagai peluang vektor x sebagai data asli atau data palsu yang dihasilkan oleh model G .

Arsitektur GAN-BERT yang diusulkan oleh Croce et al. (2020) menggabungkan *framework* GAN dengan model BERT sehingga dapat menerapkan metode *semi-supervised* GAN untuk permasalahan yang melibatkan data teks. Berdasarkan Gambar 2.9, BERT hanya menerima masukan dari data asli saja. Karena itu, model G pada GAN-BERT diharapkan dapat menghasilkan keluaran $G(z)$ yang menyerupai distribusi dari keluaran BERT. Karenanya, diperkenalkan *loss feature matching* untuk menilai kemiripan *intermediate layer* $f(x)$ dari model G dan BERT. Oleh karena itu, persamaan *loss* keseluruhan pada arsitektur GAN-BERT, yang dinyatakan dengan L_G , dapat dirumuskan sebagai berikut:

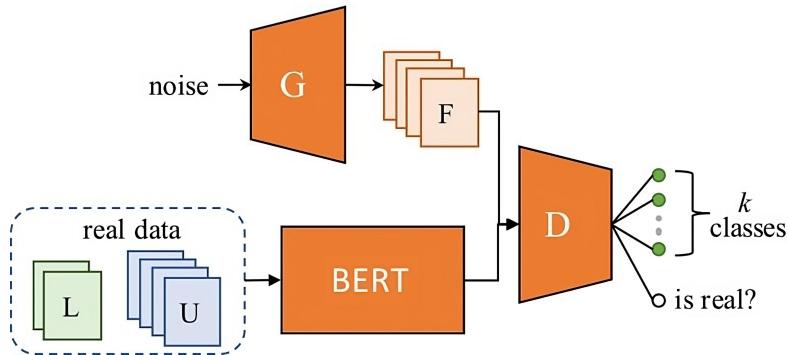
$$L_G = L_{G_{\text{feature matching}}} + L_{G_{\text{unsupervised}}}, \quad (2.33)$$

$$L_{G_{\text{feature matching}}} = \|\mathbb{E}_{x \sim p_{\text{data}}} f(x) - \mathbb{E}_{x \sim p_G} f(x)\|_2^2, \quad (2.34)$$

$$L_{G_{\text{unsupervised}}} = -\mathbb{E}_{x \sim G} \log [p_{\text{model}}(y = K+1|x)]. \quad (2.35)$$

2.6 Metrik Evaluasi

Setelah model dilatih, maka dibutuhkan metrik untuk mengetahui model yang paling baik untuk menyelesaikan *task* yang ingin dicapai pada penelitian. Namun karena permasalahan yang dihadapi adalah klasifikasi *multi-label*, maka metrik evaluasi dibagi menjadi dua. Yaitu metrik evaluasi *single-label* dari setiap kelas lalu dirata-ratakan dan metrik evaluasi untuk klasifikasi *multi-label*.



Gambar 2.9: Arsitektur dari GAN-BERT terdiri dari 3 model besar, yaitu generator (G), diskriminasi (D), dan BERT. Masukan $noise$ (z) dimasukkan ke model G dan keluarannya ialah vektor F yang mengikuti distribusi keluaran dari model BERT. Model D menentukan apakah masukan yang diterima berasal dari data asli atau tidak, beserta memprediksi label dari masukan tersebut. Gambar diambil dari penelitian Croce et al. (2020)

Untuk menguji seberapa besar kemiripan antara data anotasi mesin dan anotasi manusia, digunakan metrik *Cohen's kappa*. Metrik *Cohen's kappa* dapat diinterpretasikan sebagai tingkat persetujuan antar dua *annotator* dalam proses anotasi *dataset*. Berdasarkan McHugh (2012), persamaan kappa κ dituliskan sebagai:

$$\kappa = \frac{P_o - P_e}{1 - P_e}, \quad (2.36)$$

dengan P_o sebagai persentase persetujuan yang diobservasi dan P_e merupakan persentase persetujuan secara kebetulan. Persamaan *Cohen's kappa* juga digunakan sebagai metrik klasifikasi dengan dua *annotator* A dan B di mana salah satu *annotator* menjadi prediksi dan *annotator* lainnya menjadi *ground truth*. Maka perumusan *Cohen's kappa* menjadi:

$$\kappa = \frac{2 \times (TP \times TN - FN \times FP)}{(TP + FN) \times (FP + TN) \times (TP + FN) \times (FN + TN)}, \quad (2.37)$$

dengan TP (*true positive*) ketika prediksi dan *ground truth* bernilai benar dan TN (*true negative*) saat prediksi dan *ground truth* bernilai salah. Pada saat *prediksi* bernilai benar namun *ground truth* bernilai salah, disebut sebagai FP (*false positive*) dan FN (*false negative*) sewaktu prediksi bernilai salah namun *ground truth* bernilai benar. Nilai κ yang dihasilkan bernilai -1 hingga 1 dan diinterpretasikan menurut McHugh (2012).

Metrik *single-label* lainnya adalah presisi (*Prec*), *recall* (*Rec*), dan *f1-score* (*F1*). Presisi menilai seberapa akurat model dalam memprediksi label positif dan *recall*

mengukur seberapa baik model dalam menemukan semua *instance* yang sebenarnya positif. Terakhir *F1-score* merupakan keseimbangan antara presisi dan *recall*. Ketiga metrik ini dirata-ratakan menggunakan rerata mikro (*micro*), sebab rerata mikro memberikan bobot yang sama untuk masing-masing data (D. Zhang, Wang, & Zhao, 2015). Akibatnya, rerata mikro cocok untuk *dataset* yang memiliki jumlah kelas yang tidak seimbang. Penerapan rerata mikro untuk ketiga metrik yang digunakan sebagai:

$$Prec_{Micro} = \frac{\sum_{j \in L} TP_j}{\sum_{j \in L} TP_j + \sum_{j \in L} FP_j}, \quad (2.38)$$

$$Rec_{Micro} = \frac{\sum_{j \in L} TP_j}{\sum_{j \in L} TP_j + \sum_{j \in L} FN_j}, \quad (2.39)$$

$$F1_{Micro} = \frac{2 \times Prec_{Micro} \times Rec_{Micro}}{Prec_{Micro} + Rec_{Micro}}, \quad (2.40)$$

dengan L adalah kelas yang terdapat pada *dataset* dan TP_j merupakan seluruh TP yang ada pada kelas j .

Terdapat juga metrik *multi-label* yang menghitung performa dari keseluruhan kelas tanpa melakukan rata-rata. Metrik evaluasi *multi-label* yang digunakan pada penelitian ini berupa *exact match ratio* (EMR) dan *hamming loss*. EMR menghitung berapa banyak prediksi yang benar-benar sama untuk keseluruhan label yang ada dan memiliki fungsi yang sama dengan akurasi, sehingga disebut sebagai akurasi pada penelitian ini. Sedangkan *hamming loss* menghitung besaran pinalti model yang dihasilkan dari prediksi yang salah. Perhitungan metrik *multi-label* dirumuskan sebagai berikut:

$$EMR = \frac{1}{n} \sum_{i=1}^n I(y_i = \hat{y}_i), \quad (2.41)$$

$$Hamming Loss = \frac{1}{nL} \sum_{i=1}^n \sum_{j=1}^L \left(I(y_i^{(j)} \neq \hat{y}_i^{(j)}) \right), \quad (2.42)$$

dengan n merupakan jumlah keseluruhan data dan fungsi I merupakan fungsi indikator yang sudah dijelaskan pada Persamaan 2.9.

Tabel 2.5: Contoh data *ground truth* dan prediksi dari klasifikasi *multi-label*

No	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$	No	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$
1	0	1	0	1	0	1	1
2	0	1	1	2	0	1	1
3	1	0	1	3	0	1	0
4	0	0	1	4	0	0	0

(a) *Ground truth*

(b) Prediksi

Selanjutnya dicontohkan bagaimana cara menghitung nilai masing-masing metrik dari klasifikasi *multi-label*. Diberikan *ground truth* pada Tabel 2.5a dan contoh hasil prediksi pada Tabel 2.5b. Berikut contoh bagaimana cara menghitung *Cohen's kappa* label $y^{(2)}$ dari contoh prediksi:

$$\begin{aligned}\kappa_2 &= \frac{2 \times (2 \times 1 - 0 \times 1)}{(2+0) \times (1+1) \times (2+0) \times (0+1)} \\ &= 0,5\end{aligned}$$

Nilai presisi, *recall*, dan *f1-score* dengan rerata mikro dapat dihitung sebagai berikut:

$$\begin{aligned}Prec_{micro} &= \frac{0+2+1}{(0+2+1)+(0+1+1)} & Rec_{micro} &= \frac{0+2+1}{(0+2+1)+(1+0+2)} \\ &= 0,6 & &= 0,5 \\ F1_{micro} &= \frac{2 \times 0,6 \times 0,5}{0,6 + 0,5} \\ &= 0,545\end{aligned}$$

Terakhir, hasil dari EMR dan *hamming loss* dari contoh dituliskan sebagai:

$$\begin{aligned}EMR &= \frac{1}{4} \times (0+1+0+0) & Hamming Loss &= \frac{1}{4 \times 3} (1+1+3) \\ &= 0,25 & &= 0,417\end{aligned}$$

Maka terdapat enam metrik berbeda yang digunakan. Namun metrik *Cohen's kappa* hanya digunakan ketika mencari similaritas antar *dataset*. Metrik presisi dan *recall*

dimanfaatkan ketika ingin menganalisis lebih jauh untuk setiap kelas yang ada. Sisanya *f1-score*, EMR, dan *hamming loss* lebih diutamakan sebagai pembanding antar model. Dengan menggunakan metrik tersebut, dapat lebih banyak menganalisis performa model baik secara rata-rata setiap label maupun keseluruhan *multi-label*.

2.7 Analisis Model *Machine Learning*

Seperti pembahasan pada Subbab 1 (halaman 1), penelitian juga menganalisis hasil yang didapatkan dari model yang sudah dikembangkan. Maka pada Subbab 2.7, membangun pemahaman terhadap cara menganalisis model *machine learning* menggunakan *shapley additive explanations* dan *almost stochastic order*.

Penelitian ini mengkaji hasil eksperimen berdasarkan dua unsur utama. Pertama tentang bagaimana cara kerja model dan seberapa baik performa yang dihasilkan. Analisis cara model bekerja dilihat menggunakan kontribusi atau *effect size* dari fitur yang diberikan kepada model. Sedangkan performa dilihat berdasarkan metrik yang sudah dibahas pada Subbab 2.6 (halaman 29). Namun hanya melihat performa saja tidak cukup. Perlu dilakukan uji secara statistik untuk memastikan bahwa model yang digunakan memang benar lebih baik dibanding model lainnya.

2.7.1 Shapley Additive Explanations (SHAP)

SHapley Additive exPlanations (SHAP) yang diusulkan oleh Lundberg dan Lee (2017), merupakan sebuah metode untuk menginterpretasi hasil keluaran dari sebuah model. Hasil interpretasi berupa besaran *effect size* atau kontribusi yang dihasilkan dari masing-masing fitur terhadap hasil keluaran. Menurut Molnar (2022), SHAP dapat menemukan kontribusi masing-masing fitur secara global atau keseluruhan *dataset*. Dengan demikian dapat menemukan fitur-fitur yang paling berpengaruh dalam melakukan prediksi. Namun untuk mendapatkan kontribusi fitur secara global, SHAP perlu menghitung kontribusi dari setiap permutasi pasangan fitur yang ada. Oleh sebab itu, SHAP membutuhkan waktu yang lama ketika data yang digunakan sangat banyak.

SHAP memiliki 3 sifat penting yang membuat metode ini lebih baik dengan metode interpretasi lainnya. Sifat yang pertama adalah *local accuracy* yang dapat dijelaskan sebagai berikut. Untuk sebuah model $f(x)$ dan sebuah data (x_1, x_2, \dots, x_p) dengan M Sebagai banyak fitur untuk sebuah data. Terdapat interpretasi model $g(x')$ dengan

$x' \in \{0,1\}^M$ adalah bentuk simplifikasi dari x . Proses simplifikasi dilakukan dengan memetakan fungsi $x = f_x(x')$. Dengan demikian persamaan dari sifat *local accuracy* dituliskan sebagai:

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^M \phi_i x'_i, \quad (2.43)$$

dengan ϕ_0 merupakan nilai prediksi ketika semua fitur bernilai salah atau tidak ada fitur yang digunakan. Sedangkan ϕ_i merupakan kontribusi dari fitur x'_i terhadap keluaran $f(x)$. Sifat kedua adalah *missingness*. Jika suatu fitur tidak diobservasi, maka fitur tersebut tidak memiliki kontribusi. Hal ini dinyatakan dengan:

$$x_i = 0 \longrightarrow \phi_i = 0. \quad (2.44)$$

Sifat yang terakhir adalah *consistency*. Menurut Molnar (2022), Jika nilai suatu fitur bertambah atau tetap, maka nilai kontribusi fitur juga harus bertambah atau tetap. *Consistency* juga dapat dinotasikan sebagai berikut. Ditentukan $f_x(z') = f(h_x(z'))$ dan $z' \setminus i$ sebagai notasi dari $z' = 0$. Untuk dua model yang berbeda f dan f' dan jika

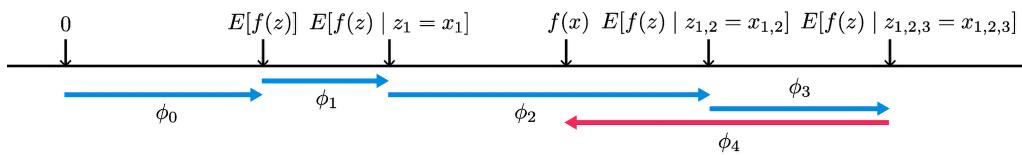
$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i), \quad (2.45)$$

untuk semua masukan $z' \in \{0,1\}^M$, maka $\phi_i(f', x) \geq \phi_i(f, x)$

Dalam hal ini kontribusi dari suatu fitur ϕ_i harus memenuhi ketiga sifat tersebut. Maka Lundberg dan Lee (2017) mengusulkan persamaan ϕ_i sebagai berikut:

$$\phi_j(f, x) = \sum_{z' \in x'} \frac{|z'|! (M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)]. \quad (2.46)$$

Berdasarkan persamaan tersebut, z' adalah *subset* dari x' dengan isi dari z' tidak boleh 0 semua. Pengaruh dari ϕ_i atau kontribusi fitur dapat dilihat pada Gambar 2.10



Gambar 2.10: Ilustrasi *effect size* atau kontribusi dari suatu fitur x . Gambar diambil dari penelitian Lundberg dan Lee (2017)

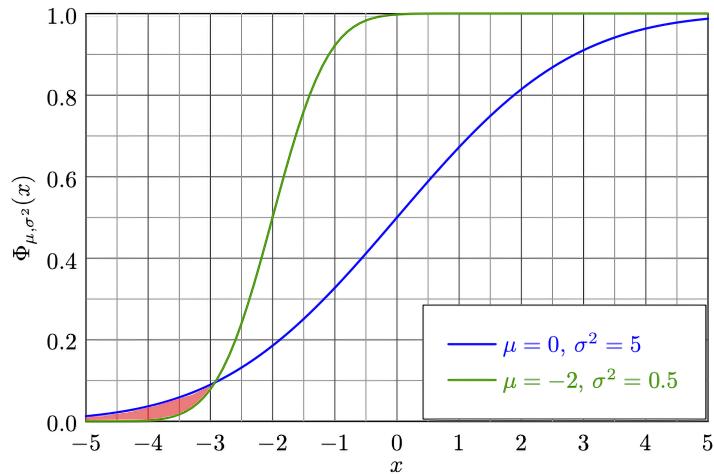
2.7.2 Almost Stochastic Order (ASO)

Untuk membandingkan *machine learning* konvensional, dapat diuji dengan t-test untuk mengetahui seberapa baik performa suatu model dan seberapa besar hasil tersebut dapat dipercaya Hamarashid (2021). Namun untuk *deep learning*, tidak dapat dilakukan uji hipotesis dengan menggunakan t-test. Menurut Dror, Shlomov, dan Reichart (2019), hal yang menyebabkan *deep learning* tidak dapat diuji dengan t-test karena terdapat banyak *hyperparameter* yang dapat berpengaruh dan pilihan acak seperti inisialisasi bobot dan *layer dropout* yang juga memengaruhi hasil akhir. Akibatnya, Ulmer et al. (2022) mengimplementasikan metode uji hipotesis bernama *almost stochastic order* (ASO) berdasarkan penelitian dari Dror et al. (2019) dan Del Barrio, Cuesta-Albertos, dan Matrán (2018).

ASO membandingkan distribusi dari kedua performa model untuk mengetahui model yang terbaik. Distribusi performa model didapatkan dengan menjalankan model beberapa kali sehingga dapat diketahui rata-rata dan varian dari performa model. ASO menghitung ϵ_{min} yang merupakan pelanggaran pada *stochastic order*. Berdasarkan Gambar 2.11, area yang berwarna merah merupakan nilai dari ϵ_{min} antar dua distribusi yang dibandingkan. Semakin kecil nilai ϵ_{min} , maka semakin tinggi tingkat kepercayaan suatu model lebih baik dibanding model lainnya. Menurut Ulmer et al. (2022), uji hipotesis terbukti benar jika $\epsilon_{min} < \tau$ dengan nilai τ bernilai kurang atau sama dengan 0,5.

2.8 Klasifikasi Spesialisasi Dokter pada Teks Kesehatan

Seperti yang sudah dijelaskan pada Subbab 1 (halaman 1), penelitian ini merupakan kelanjutan dari penelitian yang dilakukan oleh Hakim, Abid Nurul (2016) dan Nurhayati, Syifa (2019) yang membahas klasifikasi domain spesialisasi dokter pada data teks forum tanya jawab. Guna memberikan landasan pemahaman dari tema ini, Subbab 2.8 menjelaskan tentang domain spesialisasi dokter, beserta penelitian yang sudah pernah dilakukan.



Gambar 2.11: Ilustrasi distribusi yang dicari oleh ASO. Nilai ϵ_{min} yang dihasilkan merupakan area yang berwarna merah. Gambar diambil dari penelitian Ulmer et al. (2022)

2.8.1 Domain Spesialisasi Dokter

Menurut FKUI⁷, dokter spesialis adalah seorang ahli yang memiliki pengetahuan, keterampilan, dan prosedur yang mendalam dalam bidang spesifik yang merupakan bagian dari cabang ilmu kedokteran tertentu atau area minat khusus. Untuk mendapatkan pengakuan resmi, dokter spesialis harus diakui oleh kolegium yang menangani cabang ilmu kedokteran terkait dan disahkan oleh Konsil Kedokteran Indonesia. Keberadaan spesialisasi ini diperlukan karena penyakit memiliki tingkat kesulitan yang beragam dan tidak semua orang memiliki kemampuan untuk mengobatinya.

Mengacu kepada Konsil Kedokteran Indonesia⁸, kemampuan seorang dokter dalam menangani penyakit dikelompokkan menjadi empat tingkat. Tingkat pertama merupakan tingkat kemampuan yang paling mendasar dan mudah dilakukan, sedangkan tingkat keempat adalah tingkatan yang paling kompleks dan hanya dapat ditangani oleh dokter yang telah menyelesaikan pendidikan kedokteran atau menjadi dokter spesialis. Dengan demikian, spesialisasi menjadi syarat mutlak untuk mengatasi penyakit pada tingkat kemampuan empat.

Sesuai surat Pusbanglin⁹ pada 2021, terdapat 47 jurusan atau domain spesialis yang dapat ditempuh untuk menjadi dokter spesialis. Setiap spesialis memiliki fokus yang mendalam pada bidang tertentu, seperti spesialis anak, bedah, penyakit dalam, dan lain-lain. Oleh sebab itu, penyakit yang harus ditangani oleh suatu spesialis tertentu, belum

⁷<https://fk.ui.ac.id/program-subspesialis.html>

⁸https://www.kki.go.id/assets/data/arsip/SKDI_Perkonsil_11_maret_13.pdf

⁹<https://infodrg.com/wp-content/uploads/2021/12/wp-1640967253810.pdf>

tentu dapat ditangani oleh spesialis lain.

Dalam pekerjaan ini, domain spesialisasi dokter merupakan target label yang diprediksi oleh sebuah model jika diberikan sebuah pertanyaan konsultasi medis pada forum kesehatan. Label spesialisasi dokter yang diprediksi dimanfaatkan untuk mengarahkan pertanyaan pengguna forum ke dokter dengan spesialisasi yang sesuai. Dengan begitu, dokter spesialis dapat langsung menerima pertanyaan yang sesuai dengan keahliannya dan menghasilkan jawaban yang diharapkan lebih akurat serta berkualitas.

2.8.2 Penelitian sebelumnya

Permasalahan klasifikasi domain spesialisasi dokter sudah dilakukan oleh beberapa peneliti sebelumnya. Penelitian yang dilakukan oleh Kim, Kim, Kim, Song, dan Joo (2023) melakukan hal yang sama seperti pada penelitian ini, yaitu melakukan klasifikasi domain spesialisasi berdasarkan data teks pertanyaan konsultasi. Namun Kim et al. (2023) menggunakan data teks berbahasa korea, sehingga menerapkan KoRean BERT (KR-BERT) untuk menyelesaikan permasalahan klasifikasi. Terdapat juga klasifikasi spesialisasi dokter dalam bahasa Inggris juga dilakukan oleh Weng, Wagholar, McCray, Szolovits, dan Chueh (2017) dan Almuhana dan Abbas (2022). Meskipun begitu, masukan yang diberikan merupakan rekam medis yang ditulis oleh dokter. Almuhana dan Abbas (2022) menggunakan satu dimensi *convolutional neural network* (CNN) dengan *embedding* kata menggunakan *one-hot encoding*. Sedangkan Weng et al. (2017) mengaplikasikan beberapa model, seperti CNN, SVM, *naive bayes*, dan *adaboost* dengan teks *embedding* berupa *bag of words* dan ditambahkan dengan kata kunci medis. Berdasarkan penelitian yang dibahas, semua masalah yang diselesaikan adalah klasifikasi *single-label* dan belum ada yang menyelesaikan permasalahan klasifikasi *multi-label*.

Selanjutnya dijelaskan juga penelitian dengan topik yang sama, namun dengan *dataset* berbahasa Indonesia. Permasalahan klasifikasi domain spesialisasi dokter dari data teks forum tanya jawab dalam bahasa Indonesia sudah dikembangkan oleh Hakim, Abid Nurul (2016). Pada penelitian oleh Hakim, Abid Nurul (2016), permasalahan klasifikasi spesialisasi dokter ditujukan sebagai komponen dalam sistem tanya jawab medis. Tujuan dari penelitian ini untuk menghindari kesalahan jawaban yang dapat membahayakan kesehatan pengguna atau penanya. Untuk mencapai tujuan tersebut, Hakim, Abid Nurul (2016) mengembangkan *dataset* yang berasal dari *website*

forum tanya jawab berbahasa Indonesia. Proses pengembangan *dataset* dilakukan dengan melakukan pemetaan kamus terhadap *tag* atau topik pertanyaan yang ada pada *website* ke salah satu label dari 13 label spesialis yang tersedia, yaitu kandungan, penyakit dalam, anak, kulit dan kelamin, gizi, THT, gigi, mata, bedah, jiwa, tulang, dan jantung. *Dataset* yang dihasilkan dengan metode ini, disebut sebagai *dataset* anotasi mesin dengan pendekatan *rule-based*. Dengan *dataset* yang sudah dikembangkan, dibuat model klasifikasi menggunakan SVM dan *naive bayes*.

Setelahnya, Nurhayati, Syifa (2019) melanjutkan pengembangan *dataset* spesialisasi yang sudah dilakukan sebelumnya. Pengembangan yang dilakukan oleh Hakim, Abid Nurul sangat bergantung terhadap *tag* yang terdapat pada *website*, sedangkan tidak semua *website* menyediakan *tag*. Dengan demikian terdapat dua isu utama yang dilakukan Nurhayati, Syifa dalam penelitiannya. Pertama, melakukan anotasi ulang menggunakan pemetaan kamus yang dibuat oleh Hakim, Abid Nurul (2016) dengan menghasilkan dua versi *dataset*. Untuk versi pertama, jumlah label pada *instance* tidak dibatasi hanya sampai satu label saja. Sebab, ketika suatu *instance* konsultasi memiliki banyak *tag*, maka konsultasi perlu ditangani oleh beberapa dokter spesialis yang berbeda. Sedangkan versi kedua, label pada *instance* hanya dibatasi hingga satu label saja. *Dataset* versi kedua digunakan untuk menyelesaikan isu berikutnya, yaitu melatih model *machine learning naive bayes* yang nantinya digunakan untuk memprediksi pertanyaan konsultasi yang tidak memiliki *tag*. Sebab model *naive bayes* dilatih dengan *dataset* versi kedua, maka model yang dilatih masih merupakan klasifikasi *single-label*.

Selanjutnya pada tahun 2022, Nurhayati, Syifa memilih 2.525 pertanyaan dari *dataset* konsultasi yang masih belum diberi label untuk dianotasi. Data yang sudah dianotasi ini digunakan sebagai tugas klasifikasi pada mata kuliah Penambangan Data ketika Nurhayati, Syifa sedang menjabat sebagai asisten dosen mata kuliah Penambangan Data. Proses anotasi melibatkan 8 orang mahasiswa FKUI 2015 untuk menganotasi pertanyaan ke domain spesialisasi dokter. Dipilih mahasiswa FKUI yang berada pada tingkat akhir agar memiliki ilmu yang hampir menyerupai dengan dokter spesialis sesungguhnya. Data tersebut dibagi menjadi lima bagian, yaitu empat bagian sebagai data latih sebanyak 2.025, dan satu bagian menjadi data uji berjumlah 500 *instances*. Setiap bagian pertanyaan yang berasal dari data latih dianotasi oleh dua mahasiswa FKUI, sehingga total ada delapan mahasiswa FKUI yang terlibat dalam proses anotasi. Sedangkan proses anotasi untuk data uji dipilih dua orang dari delapan

orang mahasiswa yang menganotasi data latih. Dalam proses anotasi oleh dokter spesialis, terdapat 16 label yang dapat diberikan, yaitu kandungan, penyakit dalam, anak, kulit dan kelamin, gizi, THT, gigi, mata, bedah, jiwa, tulang, jantung, urologi, saraf, paru, dan umum. Atas saran dari salah satu *annotator*, ditetapkan sebuah pertanyaan dapat memiliki maksimal tiga label dokter spesialisasi.

Maka penelitian yang dilakukan pada tugas akhir ini mengembangkan model klasifikasi berdasarkan *dataset* anotasi manusia (*expert*) yang sudah dikembangkan oleh Nurhayati, Syifa. Oleh karena itu, model yang dikembangkan merupakan model klasifikasi *multi-label* yang belum pernah dilakukan oleh Hakim, Abid Nurul dan Nurhayati, Syifa. Penelitian ini juga mencari *effect size* atau kontribusi fitur terhadap klasifikasi suatu label. Hal ini juga belum pernah dilakukan oleh penelitian sebelumnya.

Metode klasifikasi yang dilakukan oleh peneliti sebelumnya hanya menggunakan model *machine learning* konvensional berupa *naive bayes* dan *SVM* dengan pendekatan *supervised learning*. Dengan demikian, penelitian ini juga menggunakan model klasifikasi konvensional lainnya seperti *logistic regression*, *XGBoost*, dan lain-lain. Peneliti juga memanfaatkan arsitektur *deep learning* yang memiliki performa yang bagus seperti BERT (Devlin et al., 2019), sebab belum pernah dicoba oleh penelitian Nurhayati, Syifa (2019). Selain itu juga memanfaatkan *dataset* yang masih belum diberi label untuk pendekatan *unsupervised* dan *semi-supervised learning*.

BAB 3

METODOLOGI

Bab 3 menjelaskan metodologi yang digunakan pada penelitian. Pertama, Subbab 3.1 menjelaskan hal yang dilakukan pada penelitian ini secara keseluruhan. Kemudian, Subbab 3.2 membahas *dataset* yang digunakan beserta persiapan yang dilakukan. Selanjutnya Subbab 3.3 menerangkan cara pembuatan model *machine learning* secara *supervised learning* beserta skenario yang diuji dalam membuat model. Setelah itu Subbab 3.4 membahas metodologi dalam mengembangkan model *machine learning* yang menggunakan data tidak berlabel. Terakhir, Subbab 3.5 menutup Bab 3 dengan penjelasan mengenai bagaimana model dan *dataset* dievaluasi.

3.1 Penelitian Klasifikasi Domain Spesialisasi Dokter

Penelitian yang dikaji merupakan bagian dari sistem tanya jawab medis berbahasa Indonesia seperti pada Gambar 3.1, yaitu sistem klasifikasi tipe spesialisasi atau domain spesialisasi dokter. Dalam sistem tersebut, *request* atau pertanyaan konsultasi pasien diprediksi oleh sistem yang dibuat pada penelitian ini. Hasil prediksi digunakan untuk melakukan pencarian pertanyaan serupa dengan domain spesialisasi dokter yang sama. Hal ini membuat proses pencarian pertanyaan serupa lebih baik dan menghasilkan *query* yang lebih relevan.

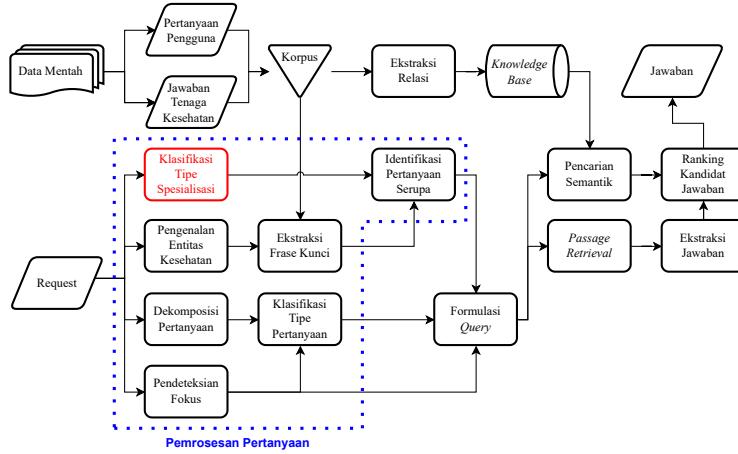
Pada penelitian Hakim, Abid Nurul (2016), dikumpulkan data forum tanya jawab dari beberapa *website*, yaitu Alodokter¹, DetikHealth², Dokter Sehat³, KlikDokter³, dan Tanyadok⁴. Hakim, Abid Nurul (2016) juga membuat kamus pemetaan untuk menganotasi *dataset* yang sudah didapatkan agar memiliki label. Selain itu, terdapat juga *dataset* yang dikembangkan oleh Nurhayati, Syifa yang dianotasi oleh ahli ke-16 domain spesialisasi dokter. Perbedaan anotasi *dataset* yang dilakukan oleh Hakim, Abid Nurul dan Nurhayati, Syifa dibandingkan agar diketahui seberapa baik kamus pemetaan yang sudah diciptakan.

¹www.alodokter.com/komunitas/diskusi/penyakit

²health.detik.com/konsultasi

³fitur sudah tidak ada pada *website*

⁴*website* sudah tidak beroperasi



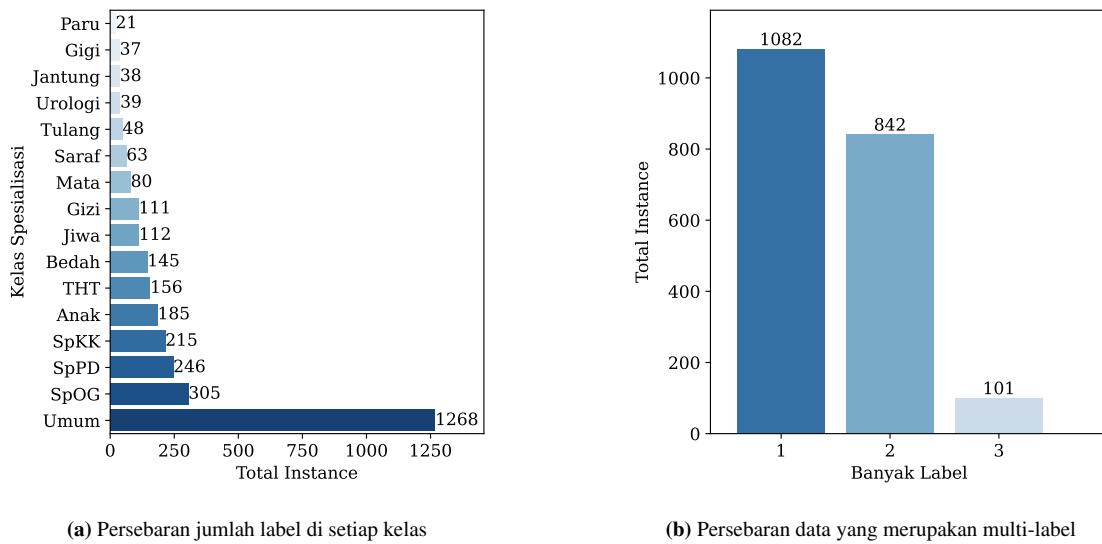
Gambar 3.1: Gambaran garis besar sistem tanya jawab medis

Dalam mengembangkan sebuah model *machine learning* yang dapat melakukan klasifikasi domain spesialisasi dokter, digunakan *dataset* yang sudah dianotasi oleh Nurhayati, Syifa. Model yang dikembangkan merupakan model dengan pendekatan konvensional dan *pre-trained language model* sebagai *deep learning*. Setiap model yang dibuat perlu dianalisis secara statistik agar didapatkan model dengan performa yang terbukti lebih baik dibandingkan model yang lain. Selain secara statistik, model juga dianalisis menggunakan SHAP untuk mendapatkan pemahaman bagaimana pengaruh fitur dalam menghasilkan prediksi.

Selain *dataset* yang sudah dianotasi oleh dokter, masih terdapat banyak data yang masih belum diberi anotasi yang dikumpulkan oleh Hakim, Abid Nurul (2016). Data yang belum diberi label dimanfaatkan dalam mengembangkan *pre-trained language model* dengan harapan dapat meningkatkan performa model. *Language model* perlu dilatih dengan paradigma pembelajaran *unsupervised* dan *semi-supervised learning* yang menggunakan data tidak berlabel.

3.2 Kajian Dataset

Seperti yang sudah dijelaskan pada Subbab 2.8.2 (halaman 37), penelitian ini menggunakan *dataset* yang sudah dikembangkan oleh Hakim, Abid Nurul (2016) dan Nurhayati, Syifa. Subbab 3.2 menjelaskan lebih dalam mengenai pembagian *dataset* yang digunakan, proses persiapan data, beserta eksplorasi yang dilakukan.



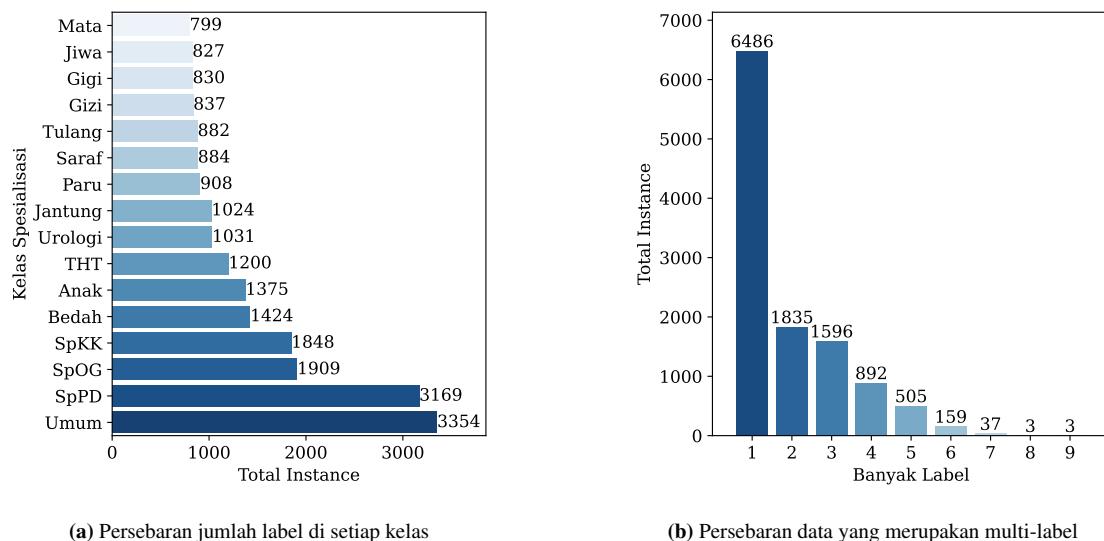
Gambar 3.2: Persebaran data anotasi oleh manusia. Pada Gambar 3.2a, SpKK merupakan kelas "kulit dan kelamin", SpPD sebagai kelas "penyakit dalam", dan SpOG adalah kelas "kandungan".

3.2.1 Dataset Penelitian

Terdapat tiga bagian data yang sudah dikembangkan oleh Nurhayati, Syifa, yaitu *dataset* yang dianotasi oleh ahli (*dataset* manusia), *dataset* yang diberi label oleh pemetaan kamus (*dataset* mesin), dan *dataset* yang belum memiliki label. Berdasarkan Gambar 3.2, dapat dilihat persebaran label dalam *dataset* yang dianotasi oleh manusia. Terlihat bahwa terdapat ketidakseimbangan kelas pada Gambar 3.2a, di mana jumlah sampel dalam kelas spesialisasi umum sangat berbeda jauh dengan jumlah sampel dalam kelas lainnya. Selain itu, *instance* yang dianotasi oleh dokter umumnya memiliki satu atau dua label spesialisasi, seperti yang ditunjukkan pada Gambar 3.2b. Didapatkan juga bahwa hanya ada beberapa *instance* yang memiliki label sebanyak tiga.

Berbeda dengan *dataset* anotasi manusia, *dataset* yang dianotasi mesin hanya berasal dari tiga *website* saja, yaitu Alodokter, DetikHealth, dan KlikDokter. Hal tersebut dikarenakan hanya ketiga *website* itu saja yang sudah mengelompokkan pertanyaan ke *tag* atau kata kunci tertentu. Seperti yang sudah dijelaskan pada Subbab 2.8.2 (halaman: 37), terdapat perbedaan jumlah label antar kedua *dataset*. Oleh sebab itu, pada penelitian ini mengembangkan kamus pemetaan kata kunci yang sudah dibuat sebelumnya agar dapat sesuai dengan 16 label yang baru. Kata kunci yang dikembangkan dapat dilihat pada Lampiran 1 (halaman 108).

Berdasarkan data anotasi mesin yang dikembangkan oleh Nurhayati, Syifa dengan memanfaatkan kamus Hakim, Abid Nurul (2016), didapatkan 11.516 pertanyaan yang



Gambar 3.3: Persebaran data anotasi oleh mesin. Pada Gambar 3.2a, SpKK merupakan kelas "kulit dan kelamin", SpPD sebagai kelas "penyakit dalam", dan SpOG adalah kelas "kandungan".

dianotasi dengan cara pemetaan kamus. Persebaran jumlah data berdasarkan kelas dan banyaknya label untuk suatu *instance* dapat dilihat di Gambar 3.3. Jika dibandingkan dengan persebaran data anotasi manusia, terlihat bahwa ketidakseimbangan kelas pada data anotasi mesin di Gambar 3.3a tidak seburuk data anotasi manusia. Hal tersebut dikarenakan data anotasi mesin tidak memiliki batasan jumlah label maksimal untuk suatu *instance*. Contohnya seperti pada Gambar 3.3b, maksimal jumlah label untuk suatu *instance* mencapai sembilan label. Informasi ini berguna untuk melihat *prior distribution* dari label yang mungkin saja berguna untuk pengusulan model atau penerapan metode penyeimbangan data seperti *oversampling* dan *undersampling*.

Dataset yang terakhir adalah *dataset* yang belum diberi label atau tidak memiliki label. Jumlah data yang tidak berlabel mencapai 69.181 *instance* pertanyaan konsultasi. Jumlah data ini cukup besar jika dibandingkan dengan *dataset* lainnya. Oleh karena itu, data ini dapat digunakan untuk meningkatkan kinerja model dengan pendekatan paradigma pembelajaran *semi-supervised learning* atau *unsupervised learning*, terutama jika data anotasi manusia masih belum mencukupi.

3.2.2 Persiapan Dataset

Langkah pertama yang dilakukan adalah melihat apakah terdapat irisan antara *dataset*. Hal ini perlu dipastikan agar tidak ada *instance* yang berada di dua bagian data yang berbeda. Setelah tidak ada *instance* yang beririsan, langkah selanjutnya adalah

melakukan pembersihan data. Pembersihan dilakukan dengan menghapus kode HTML, URL, dan alamat email pada judul dan isi pertanyaan. Berikutnya, data yang tidak relevan dengan domain penelitian, seperti teks iklan dan teks jawaban konsultasi, perlu dihapus. Terakhir, kalimat-kalimat yang hampir identik atau sama persis dengan kalimat lainnya juga dihapus. Untuk mendeteksi kalimat yang diduga duplikat, digunakan *library difflib*⁵ dengan batas kemiripan sebesar 0,7.

Ketika data selesai dibersihkan, *dataset* sudah dapat digunakan untuk melatih model. Saat melatih model, data dibagi menjadi dua bagian: data latih dan data validasi. Rasio pembagian data adalah 8 banding 2, di mana 80% merupakan data latih dan 20% merupakan data validasi. Pembagian data dilakukan dengan menggunakan metode stratifikasi agar persebaran kelas antara data latih dan data validasi tidak memiliki perbedaan yang signifikan. Dalam penelitian ini, aplikasi metode tersebut memanfaatkan *library scikit-multilearn* yang dikembangkan oleh Szymánski dan Kajdanowicz (2019) untuk melakukan stratifikasi pada data *machine learning*.

Dalam pengembangan model *unsupervised* maupun *semi-supervised learning*, diperlukan data tambahan yang tidak memiliki label. Pada penelitian ini, data teks yang ada pada *dataset* tidak berlabel dan *dataset* anotasi mesin dimanfaatkan sebagai data tambahan untuk metode tersebut. Data anotasi mesin digabungkan dengan data tidak berlabel karena distribusi jumlah label pada data anotasi mesin berbeda dengan *dataset* yang dilabeli oleh manusia, sehingga tidak cocok untuk digunakan dalam *supervised learning*. Namun, karena *dataset* tidak berlabel sangat besar dan sumber daya terbatas, dilakukan penyaringan data tambahan untuk mempercepat proses pelatihan model. Proses penyaringan bertujuan agar hanya pertanyaan yang mengandung kata kunci medis yang digunakan dalam pendekatan *unsupervised* dan *semi-supervised learning*. Dalam penelitian ini, kategori topik pada *website Alodokter*⁶ digunakan sebagai kata kunci medis. Untuk mengurangi jumlah data hasil penyaringan yang terlalu besar, ditetapkan bahwa minimal terdapat tiga kata kunci medis agar pertanyaan dapat dimasukkan ke dalam data tambahan. Dengan begitu, total terdapat 5.957 pertanyaan yang tidak berlabel yang digunakan untuk *unsupervised* dan *semi-supervised learning*.

⁵<https://docs.python.org/3/library/difflib.html>

⁶<https://www.alodokter.com/komunitas/topik>

3.2.3 Eksplorasi Dataset

Berdasarkan pembahasan pada Subbab 3.2.1 (halaman 42), terdapat perbedaan sumber data antara anotasi manusia dan anotasi mesin. Oleh karena itu, beberapa data anotasi manusia tidak memiliki label anotasi mesin. Selain itu, perbedaan himpunan kelas antara anotasi manusia dan anotasi mesin juga menyebabkan adanya kelas yang tidak mungkin memiliki kesamaan. Dengan demikian, dalam penelitian ini dilakukan anotasi ulang dengan menggunakan aturan pemetaan yang telah diperbarui, seperti yang tercantum dalam Lampiran 1 (halaman 108). Dalam pengembangan kamus, peneliti terus menambahkan kata kunci sehingga semua *instance* memiliki label ketika judul dan isi dari *instance* dipetakan.

Terdapat empat skenario anotasi ulang berdasarkan bagian yang dianotasi. Skenario pertama dengan menganotasi *tag* yang terdapat pada data. Skenario ini hanya menggunakan data anotasi manusia yang berasal dari *website* Alodokter, DetikHealth, dan KlikDokter sebagai *website* yang sudah menyediakan *tag*. Selanjutnya, skenario kedua melakukan anotasi berdasarkan kata yang ada pada judul sedangkan skenario ketiga melakukan anotasi berdasarkan kata yang ada pada isi. Dengan skenario tersebut, semua data anotasi manusia memiliki label anotasi mesin karena semua *instance* memiliki judul dan isi. Skenario keempat menggabungkan kalimat judul dan isi, setelah itu baru dilakukan anotasi.

Selain dari aspek bagian yang dianotasi, terdapat juga skenario untuk membatasi label anotasi. Berdasarkan penelitian yang dilakukan oleh Hakim, Abid Nurul (2016), setiap *tag* yang ditemukan menghasilkan label spesialisasi. Hal ini menyebabkan beberapa *instance* memiliki hingga 9 label, seperti yang terlihat pada Gambar 3.3b. Oleh karena itu, skenario berikutnya mengusulkan proses *filtering* untuk membatasi jumlah label menjadi maksimal tiga. Proses yang diusulkan adalah sebagai berikut: ketika terdapat suatu *tag*, *tag* tersebut memberikan skor kepada label yang terkait. Setiap label yang mendapatkan skor dihitung rata-ratanya dan nilai standar deviasinya. Suatu label dikatakan relevan dengan *instance* tersebut jika selisih skor label dengan skor label tertinggi tidak melebihi nilai standar deviasi. Setiap label yang memenuhi syarat tersebut dianggap sebagai klasifikasi untuk *instance* tersebut. Jika terdapat lebih dari tiga label yang memenuhi syarat, hanya tiga label dengan skor tertinggi yang digunakan. Selain itu, dilakukan juga anotasi ke *dataset* data uji yang dianggap sebagai prediksi dengan metode pemetaan *tag*. Dengan begitu, dapat diketahui performa metode pemetaan *tag*

untuk klasifikasi.

Selain itu, perlu dilakukan pengecekan terhadap korelasi antar label. Hal ini penting karena permasalahan klasifikasi *machine learning* bisa saja memiliki label yang saling dependensi. yang digunakan bergantung pada label yang diprediksi sebelumnya. Jika terdapat korelasi antara beberapa label, itu berarti terdapat dependensi antar label yang perlu dipertimbangkan dalam proses prediksi. Beberapa metode yang digunakan untuk mencari korelasi antar label adalah jarak Jaccard (*Jaccard distance*), *positive pointwise mutual information* (PPMI), dan *Cohen's kappa* yang sama seperti yang dijelaskan dalam Subbab 2.6. Jarak Jaccard dihitung dengan persamaan Persamaan 3.1 berikut:

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}, \quad (3.1)$$

dengan A dan B merupakan dua himpunan label yang ingin dicari korelasinya. Kemudian, nilai PPMI dihitung dengan Persamaan 3.2 berikut:

$$PPMI(A; B) = \max \left(\log \left(\frac{P(A, B)}{P(A) \cdot P(B)} \right), 0 \right), \quad (3.2)$$

dengan

$$P(A) = \frac{|A|}{|N|}, \quad (3.3)$$

$$P(A, B) = \frac{|A \cap B|}{|N|}. \quad (3.4)$$

Perhatikan bahwa N adalah *dataset*; $|A|$ merupakan ukuran data yang diberi label A ; dan $|N|$ adalah ukuran data yang ada pada *dataset*.

Kesamaan jarak Jaccard, PPMI, dan *Cohen's Kappa* dipilih karena hanya mempertimbangkan korelasi antar label yang muncul bersama dan tidak memperhitungkan korelasi dari label yang tidak muncul bersama. Oleh karena itu, metode ini diharapkan bisa mendapatkan pola dari label yang muncul bersamaan. Dari korelasi antar kelas yang didapatkan, ditentukan nilai ambang batas (*threshold*) yang menentukan apakah pasangan label memiliki dependensi atau tidak. Nilai ambang batas ini ditentukan secara heuristik setelah melihat nilai korelasi antar label.

3.3 Pengembangan Model Prediksi Spesialisasi Dokter

Peneliti melakukan pengembangan model *machine learning* dengan pendekatan konvensional dan *deep learning* berbasis *language model*. Subbab 3.3 menjelaskan lebih dalam mengenai skenario-skenario pemodelan yang diuji dalam penelitian.

3.3.1 Model *Machine Learning* Konvensional

Pendekatan yang diuji pertama dilakukan dengan pendekatan *machine learning* konvensional. Hanya paradigma *supervised learning* yang digunakan saat melatih *machine learning* konvensional. Namun sebelum membuat model, terdapat beberapa tahapan prapemrosesan data yang dilakukan sebelum digunakan untuk melatih model. Keseluruhan alur proses yang dilakukan dalam membuat model meliputi pengolahan teks, reduksi fitur, pembuatan model, dan pemilihan pendekatan *multi-label*. Setiap tahapan memiliki berbagai pilihan yang dieksplorasi untuk mencari model terbaik. Rincian mengenai setiap tahapan dan pilihan yang diuji dapat dilihat pada Tabel 3.1.

Namun sebelum melakukan pengolahan teks, perlu dilakukan transformasi pada data. Hal ini dikarenakan setiap pertanyaan terdiri dari judul dan isi, sedangkan *machine learning* konvensional hanya menerima satu vektor masukan. Oleh karena itu, judul dan isi perlu digabungkan menjadi satu kalimat masukan sebelum dilakukan eksperimen. Pada penelitian *machine learning* konvensional ini, bagian judul dan isi digabungkan dengan menggunakan spasi sebagai pemisah.

Tabel 3.1: Skenario yang dilakukan pada masing-masing tahapan dalam membuat model *machine learning* konvensional

Tahapan	Pilihan yang Diuji
Pengolahan teks	Tanpa perubahan, <i>stemming</i> , penghapusan <i>stopword</i> dan <i>stemming</i>
Reduksi fitur	Tanpa reduksi, SVD 100 fitur, SVD 250 fitur, SVD 500 fitur
Pendekatan <i>multi-label</i>	<i>Binary relevance</i> dan <i>classifier chain</i>
Algoritma <i>machine learning</i>	<i>Decision tree</i> , <i>multinomial naive bayes</i> , <i>logistic regression</i> dengan dan tanpa SGD, SVC dengan dan tanpa SGD, dan XGBoost

Selanjutnya, pada tahapan pengolahan teks, proses penghapusan *stopword* dan *stemming* menggunakan *library* PySastrawi⁷. Namun, untuk penghapusan *stopword*, beberapa kata dimasukkan ke dalam *whitelist* karena berhubungan dengan konteks penyakit. Kata-kata tersebut adalah "mata", "ingat", dan "orang". Selain itu, beberapa kata juga ditambahkan sebagai *stopword*, yaitu "dok", "doc", "dokter", "terima", "kasih", dan "terimakasih".

Sebelum dimasukkan ke dalam model, kalimat perlu diubah menjadi vektor menggunakan metode *word embedding*. Pada penelitian ini, digunakan metode TF-IDF dari scikit-learn yang diimplementasikan oleh Pedregosa et al. (2011). Penggunaan TF-IDF memungkinkan setiap kata dalam data latih atau korpus menjadi fitur untuk model. Jika terdapat banyak kata dalam korpus, maka dimensi TF-IDF juga semakin banyak. Oleh karena itu, dilakukan reduksi fitur menggunakan *single value decomposition* (SVD) yang juga diimplementasikan oleh scikit-learn (Pedregosa et al., 2011). Reduksi fitur menggunakan SVD memungkinkan jumlah fitur atau dimensi TF-IDF berkurang, sehingga waktu yang dibutuhkan untuk melatih model menjadi lebih cepat, namun tetap mempertahankan informasi dari kalimat. Pada penelitian ini, dilakukan reduksi fitur sebanyak 100, 250, dan 500.

Setelah mendapatkan vektor masukan, tahapan selanjutnya adalah melatih model *machine learning* konvensional berdasarkan prapemrosesan yang telah dilakukan. Terdapat beberapa model konvensional yang digunakan dalam penelitian ini, yaitu *multinomial naive bayes*, *decision tree*, *logistic regression*, *support vector machine* (SVC), dan *extreme gradient boosting* (XGBoost). Hampir semua model tersebut menggunakan *library* scikit-learn (Pedregosa et al., 2011), kecuali XGBoost yang menggunakan *library* khusus XGBoost yang dikembangkan oleh T. Chen dan Guestrin (2016). Semua model dilatih menggunakan *hyperparameter default* yang ditentukan oleh *library* yang digunakan. Oleh karena itu, tidak dilakukan proses *hyperparameter tuning* pada eksperimen.

Berdasarkan penjelasan sebelumnya, semua model *machine learning* konvensional yang digunakan dalam penelitian ini tidak dapat menangani langsung masalah klasifikasi *multi-label*. Oleh karena itu, perlu dilakukan transformasi *dataset* yang awalnya merupakan permasalahan klasifikasi *multi-label* menjadi klasifikasi *single-label*. Metode yang digunakan dalam penelitian ini adalah *binary relevance* (BR) dan *chain classifier*

⁷<https://github.com/har07/PySastrawi>

(CC). Metode *label powerset* tidak digunakan karena menghasilkan terlalu banyak kelas, yaitu 2^{16} . Dengan menggunakan pendekatan BR dan CC, setiap kelas akan memiliki model yang berbeda dan digabungkan agar dapat menyelesaikan klasifikasi *multi-label*.

Berdasarkan desain yang telah dijelaskan, dibuat model untuk setiap kombinasi proses yang dapat dilakukan. Setiap skenario model dievaluasi menggunakan metode *10-fold cross validation* pada data latih. Hal ini dilakukan untuk memastikan bahwa performa model yang dihasilkan merupakan performa yang *robust* terhadap perubahan dan tidak mengalami *overfitting*.

Metrik yang digunakan dalam menilai performa model *machine learning* konvensional berupa EMR, rerata mikro *f1-score*, dan *hamming loss*. Dikarenakan terdapat banyak skenario yang perlu diuji, maka uji t-test dilakukan hanya untuk tiga model terbaik berdasarkan masing-masing metrik yang digunakan. Uji t-test digunakan untuk mendapatkan model dengan performa terbaik secara statistik. Model terbaik tersebut dilatih ulang sebanyak 10 kali agar didapatkan uji t-test yang lebih akurat.

3.3.2 BERT

Language model BERT yang digunakan sudah dilatih secara *pre-training*, maka tahap eksperimen ini hanya perlu melakukan *fine-tuning* yang merupakan metode *supervised learning*. Model yang diuji menggunakan IndoNLU-base⁸, IndoNLU-large⁹, IndoLEM¹⁰, mBERT¹¹, dan IndoWH¹². *Hyperparameter* yang digunakan pada tahap *fine-tuning* memiliki konfigurasi yang sama untuk semua model BERT yang digunakan. Konfigurasi tersebut mencakup penggunaan metode optimisasi AdamW, yang telah terbukti lebih baik dalam generalisasi model dibandingkan dengan metode optimisasi Adam (Loshchilov & Hutter, 2019). Eksperimen ini juga menggunakan regularisasi dan pemanasan *learning rate*, mengikuti penelitian yang dilakukan oleh Devlin et al. (2019). Informasi lebih rinci mengenai konfigurasi tersebut dapat dilihat pada Tabel 3.2. Setiap model dilatih selama 100 *epoch*, dan model yang memiliki nilai EMR terbaik pada tahap validasi digunakan.

⁸<https://huggingface.co/indobenchmark/indobert-base-p1>

⁹<https://huggingface.co/indobenchmark/indobert-large-p1>

¹⁰<https://huggingface.co/indolem/indobert-base-uncased>

¹¹<https://huggingface.co/bert-base-multilingual-cased>

¹²<https://huggingface.co/stevenwh/indobert-base-p2-finetuned-mer-80k>

Pre-trained language model BERT yang digunakan juga perlu ditambahkan sebuah *layer feed forward neural network* (FFNN) diakhir arsitektur agar dapat menyelesaikan permasalahan klasifikasi. *Layer FFNN* yang ditambahkan menggunakan fungsi *sigmoid* sebagai fungsi aktivasi beserta *binary cross entropy* sebagai fungsi untuk menghitung nilai *loss*. Implementasi ini diperlukan agar model BERT dapat menyelesaikan permasalahan klasifikasi *multi-label*. Dengan begitu, sebuah kelas diprediksi benar jika hasil keluaran dari model bernilai lebih besar atau sama dengan 0,5.

Metrik yang digunakan juga sama seperti model konvensional, yaitu EMR, rerata mikro *f1-score*, dan *hamming loss*. Namun untuk menemukan model terbaik, tidak bisa menggunakan uji t-test. Uji statistik dilakukan menggunakan metode ASO dengan tingkat kepercayaan $\alpha = 0,05$ dan mengembalikan nilai ϵ_{min} . Suatu model dikatakan lebih baik ketika nilai ϵ_{min} lebih kecil atau sama dengan 0,5. Setelah didapatkan model terbaik, dilakukan analisis lebih mendalam dengan menghitung presisi dan *recall* dari masing-masing kelas.

Tabel 3.2: Hyperparameter yang digunakan saat melakukan *fine-tuning* model BERT

Hyperparameter	Nilai
Optimisasi	AdamW
<i>Learning rate</i>	10^{-5}
β_1, β_2	(0.9, 0.999)
Fungsi aktivasi	LeakyRelu
<i>Dropout</i>	0.01
Regularisasi	0.02
Rasio Pemanasan	0.1
<i>Epoch</i>	100

3.4 Pemanfaatan Data Tidak Berlabel

Peneliti juga memanfaatkan data yang tidak berlabel dalam meningkatkan performa model *machine learning* yang sudah dikembangkan pada Subbab 3.3 (halaman 37). Oleh sebab itu, Subbab 3.4 memaparkan lebih detail mengenai motivasi mengapa menggunakan data tidak berlabel beserta pendekatan paradigma *unsupervised* dan *semi-supervised learning* yang diterapkan.

3.4.1 Analisis dengan Kata Kunci

Sebelum menggunakan data tidak berlabel, perlu melihat apakah data anotasi manusia memang kurang banyak untuk dilakukan pendekatan *supervised learning*. Maka peneliti mengusulkan metode dengan menggunakan kata kunci yang merepresentasikan setiap kelas dan melihat apakah kata kunci berada pada data latih dan data uji. Kata kunci yang digunakan dalam analisis terdiri dari token unigram, bigram, dan trigram. Penggunaan token hingga trigram diperlukan karena terdapat beberapa kata medis yang terdiri dari beberapa kata, seperti "sakit jantung" dan "sakit tulang punggung".

Untuk mendapatkan kata kunci, pertama-tama dicari skor untuk masing-masing token pada data latih. Skor tersebut mencerminkan seberapa baik suatu token dalam memprediksi label tertentu. Skor diperoleh dengan menghitung berapa kali token tersebut muncul dalam label tertentu pada data latih. Namun, karena terdapat ketidakseimbangan jumlah sampel pada setiap kelas, skor perlu dinormalisasi. Normalisasi dilakukan dalam dua tahap. Pertama, dengan mempertimbangkan jumlah token yang muncul pada setiap kelas, dan kedua, dengan mempertimbangkan seberapa banyak token tersebut muncul dalam keseluruhan *dataset*.

Setelah mendapatkan skor, langkah selanjutnya adalah melihat apakah suatu token cenderung muncul pada salah satu label atau tidak. Jika skor suatu token merata untuk setiap label, maka token tersebut tidak dapat menjadi kata kunci untuk label tertentu. Untuk menentukan kecenderungan munculnya token pada suatu label, diusulkan sebuah metode yang menggunakan *log scaling* terhadap skor. Dengan *log scaling*, skor berubah menjadi bilangan bulat negatif. Kemudian, dicari nilai tertinggi dari skor yang telah di-*log scaling*. Label yang memiliki skor *log scaling* tertinggi mendapatkan token tersebut sebagai kata kunci. Namun, karena ini merupakan masalah klasifikasi *multi-label*, suatu kata kunci hanya dapat mewakili hingga tiga kelas untuk sebuah sampel, sesuai dengan jumlah maksimum label pada anotasi manusia. Jika terdapat lebih dari tiga label yang memiliki nilai *log scaling* tertinggi, maka token tersebut tidak dianggap sebagai kata kunci. Langkah terakhir adalah melakukan tinjauan manual terhadap kata kunci yang telah didapatkan, untuk memastikan apakah kata kunci tersebut mewakili label dengan baik atau terlalu umum untuk menjadi kata kunci.

Kata kunci yang telah didapatkan untuk setiap label digunakan untuk melakukan analisis terhadap kesalahan klasifikasi. Berdasarkan kata kunci tersebut, dicari seberapa sering kata kunci tersebut muncul pada data latih dan pada sampel yang salah

diklasifikasikan dalam data uji. Jika terdapat banyak sampel misklasifikasi yang tidak mengandung kata kunci yang sesuai, hal ini mengindikasikan bahwa data latih yang digunakan kurang mencakup variasi yang cukup untuk memprediksi sampel tersebut.

3.4.2 DAPT

Berdasarkan model BERT terbaik yang sudah didapatkan, model tersebut digunakan kembali untuk melakukan *domain adaptive pre-training* (DAPT). Proses DAPT merupakan metode *unsupervised learning* yang hanya menggunakan data tidak berlabel untuk melatih kembali *language model* BERT yang telah dipilih sebelumnya. Dalam penerapan DAPT, *dataset* yang digunakan untuk melatih model harus memiliki domain yang sama dengan permasalahan yang ingin diselesaikan, yaitu domain konsultasi medis atau *dataset* tidak berlabel. Oleh karena itu, dalam proses pelatihan model BERT, model dilatih dengan menggunakan tugas *next sentence prediction* (NSP) dan *masked language model* (MLM) menggunakan teks konsultasi pada data latih dan data yang tidak berlabel. Model dilatih selama 10 *epoch*, dan bobot model disimpan untuk dibandingkan dengan tahap *fine-tuning*. Tahapan *fine-tuning* berikutnya dilatih untuk menyelesaikan tugas klasifikasi domain spesialis dokter dan performa setiap *epoch* dibandingkan untuk menentukan *epoch* yang memberikan hasil terbaik. Model dengan hasil terbaik digunakan dalam tahap *semi-supervised learning* berikutnya.

Dalam pelatihan DAPT, konfigurasi yang digunakan tidak jauh berbeda dengan konfigurasi yang digunakan dalam *supervised learning*. Informasi lebih rinci mengenai konfigurasi model untuk melatih DAPT dapat dilihat pada Tabel 3.3.

Tabel 3.3: Hyperparameter yang digunakan saat melakukan DAPT terhadap model BERT terbaik pada tahap *supervised learning*

Hyperparameter	Nilai
Optimisasi	AdamW
<i>Learning rate</i>	10^{-5}
β_1, β_2	(0.9, 0.999)
Fungsi aktivasi	LeakyRelu
Regularisasi	0.02
<i>Epoch</i>	10

Selain itu, diperlukan pengubahan data teks yang tidak berlabel menjadi *dataset* yang

Judul	:	berat badan turun
Isi	:	mohon bantu dok . cara tambah berat ?
Judul	:	batuk pilek . tolong dok !
Isi	:	resep obat batuk !?

Gambar 3.4: Contoh pertanyaan konsultasi yang terdiri dari judul dan isi

dapat digunakan. Hal ini dilakukan karena permasalahan NSP membutuhkan dua kalimat yang saling berhubungan, sehingga sebuah paragraf dibagi menjadi beberapa kalimat. Jika dua kalimat berasal dari satu *instance* dan tahapan yang sama serta merupakan kelanjutan dari kalimat sebelumnya, maka *instance* baru tersebut diberi label NSP bernilai satu (benar). Ilustrasi pemecahan data teks menjadi *dataset* NSP dapat dilihat pada Gambar 3.4 dan Tabel 3.4.

Tabel 3.4: Hasil transformasi data teks menjadi *dataset* yang terdiri dari teks 1, teks 2, dan NSP. NSP bernilai satu ketika teks 2 merupakan kelanjutan dari teks 1 pada data teks awal.

Teks 1	Teks 2	NSP
berat badan turun .	mohon bantu dok .	0
mohon bantu dok .	cara tambah berat ?	1
cara tambah berat ?	batuk pilek .	0
batuk pilek .	tolong dok !	1
tolong dok !	resep obat batuk !?	0

Selain permasalahan NSP, permasalahan MLM juga perlu dilatih secara bersamaan. Proses pembuatan *dataset* MLM mengikuti implementasi dari Devlin et al. (2019), di mana setiap token memiliki peluang 15% untuk di-*masking*. Ada tiga prosedur *masking* yang diterapkan secara acak. Pertama, dengan peluang 80%, suatu token diubah menjadi token khusus [MASK]. Kedua, dengan peluang 10%, suatu token diubah menjadi token acak. Ketiga, dengan peluang 10%, suatu token tidak berubah menjadi token lain. Meskipun demikian, semua token yang telah ditandai untuk proses *masking* diprediksi oleh BERT. Jika hasil prediksi berbeda dengan teks asli, maka model diberikan *loss* (Devlin et al., 2019).

3.4.3 GAN-BERT

Model selanjutnya dilatih dengan paradigma *semi-supervised learning*, yaitu dengan mengimplementasikan GAN-BERT. Model GAN-BERT pada tahap *semi-supervised learning* memanfaatkan data anotasi manusia sebagai data berlabel dan data tambahan yang telah disaring sebagai data tidak berlabel. Namun, arsitektur GAN-BERT yang

diimplementasikan oleh Croce et al. (2020) tidak dapat digunakan untuk klasifikasi multi-label. Oleh karena itu, dalam penelitian ini dilakukan modifikasi terhadap implementasi tersebut.

Tabel 3.5: *Hyperparameter* yang digunakan saat melatih model GAN-BERT

Hyperparameter	Nilai
<i>Hidden layer</i> generator & diskriminat	1
Optimisasi generator & diskriminat	AdamW
<i>Learning rate</i> generator & diskriminat	10^{-5}
β_1, β_2	(0.9, 0.999)
Fungsi aktivasi	LeakyRelu
Ukuran <i>noise</i>	100
Regularisasi	0.02
Penyeimbangan data	Benar
<i>Dropout</i>	0.01
Rasio Pemanasan	0.1
<i>Epoch</i>	100

Salah satu modifikasi yang dilakukan adalah mengubah fungsi *loss* dari *supervised* diskriminat. Fungsi *loss* sebelumnya, yaitu *log loss*, diubah menjadi *binary cross entropy* sebagai *loss* dari klasifikasi *multi-label*. Selain itu, terdapat juga modifikasi pada diskriminat, di mana fungsi aktivasi softmax diganti menjadi sigmoid. Rincian *hyperparameter* yang digunakan dalam arsitektur GAN-BERT seperti pada Tabel 3.5.

3.5 Evaluasi Model dan Dataset

Evaluasi yang dilakukan meliputi analisis bagaimana model bekerja beserta evaluasi data berlabel yang digunakan untuk melatih model. Subbab 3.3 menyajikan penjelasan lebih dalam tentang ketentuan model yang dievaluasi dengan SHAP dan cara mengevaluasi data.

3.5.1 Shapley Additive Explanations

Alat SHAP digunakan untuk mengevaluasi bagaimana model bekerja dengan melihat *effect size* dari setiap fitur. Akibat keterbatasan sumber daya, hanya model konvensional

terbaik dari metode *machine learning* yang dianalisis menggunakan metode SHAP. Agar model dapat diinterpretasikan dengan SHAP, model yang dievaluasi hanya model yang tidak menggunakan SVD pada tahap reduksi fitur. SHAP tidak dapat mengevaluasi model untuk permasalahan *multi-label*. Maka dicari *effect size* dari setiap label yang terdapat pada *dataset*.

3.5.2 Evaluasi pada Anotasi Data

Selain digunakan untuk analisis kesalahan model BERT, kata kunci juga digunakan untuk mengidentifikasi kesalahan pada anotasi data latih. Analisis dilakukan dengan mencari data yang merupakan *false negative* (FN) dan *false positive* (FP) pada *dataset*. Suatu sampel dikatakan sebagai FN jika anotasi yang diberikan adalah negatif, tetapi memiliki banyak kata kunci yang seharusnya menandakan sampel tersebut bisa saja positif. Sebaliknya, suatu sampel disebut sebagai FP ketika anotasi yang diberikan adalah positif, namun tidak mengandung kata kunci yang seharusnya terkait dengan label tersebut.

Setelah itu, peneliti memeriksa sampel-sampel yang diduga sebagai FN dan FP untuk mengevaluasi apakah terdapat kesalahan dalam anotasi. Jika terjadi kesalahan, sampel tersebut ditandai dan dilakukan peninjauan ulang terhadap data yang salah dianotasi. Namun, karena peneliti bukan ahli di bidang medis, peninjauan ulang hanya dilakukan untuk kelas gigi, gizi, jiwa, kandungan, kulit dan kelamin, mata, dan tulang. Peneliti juga menggunakan sumber daya *internet* untuk mencari informasi mengenai label yang sebenarnya dari pertanyaan konsultasi tersebut.

BAB 4

IMPLEMENTASI

Bab 4 menjelaskan cara penulis implementasi program yang digunakan pada penelitian. Pertama, Subbab 4.1 menjelaskan tentang cara membersihkan, menghapus duplikat, dan menyaring teks data. Kemudian, Subbab 4.2 membahas implementasi cara mengklasifikasi *instance* berdasarkan kata kunci. Selanjutnya Subbab 4.3 menerangkan cara melakukan prapemrosesan dan pembuatan model konvensional yang dibuat. Dilanjutkan dengan Subbab 4.4 yang membahas kode untuk membuat model *machine learning* dan cara melatih model. Setelah itu, Subbab 4.5 menjelaskan cara mencari kata kunci yang merepresentasikan suatu label. Terakhir, Subbab 4.6 menutup Bab 4 dengan implementasi kode penggunaan SHAP agar dapat digunakan untuk masalah *multi-label*.

4.1 Persiapan Data

Sebelum menggunakan data, data perlu dibersihkan untuk mengurangi *noise* yang dapat memengaruhi performa model. Beberapa langkah yang dilakukan dalam proses pembersihan data antara lain menghilangkan kode HTML, URL, dan email, menghapus data yang tidak relevan, serta mengeliminasi data duplikat.

Pada Kode 4.1, diimplementasikan fungsi untuk menghapus kode HTML dan teks lainnya dengan menggantinya menjadi *whitespace* menggunakan *regex*. Selain itu, terdapat beberapa data yang tidak relevan karena bukan merupakan pertanyaan, tetapi termasuk dalam *dataset*. Data yang tidak relevan memiliki pola tertentu, seperti *link* Twitter dan teks "team doktersehat" yang menandakan jawaban dari sebuah pertanyaan. Oleh karena itu, dapat dibuat *regex* untuk mengidentifikasi *instance* yang mengandung pola tersebut dan kemudian menghapusnya.

```
1 def cleaning_text(df):
2     # Menghapus substring yang tidak dibutuhkan
3     replace_word_in_dataframe(df, r"[regex kode html]", " ")
4     replace_word_in_dataframe(df, r"[regex link url]", " ")
5     replace_word_in_dataframe(df, r"[regex alamat email]", " ")
6     replace_word_in_dataframe(df, r" {2,}", " ")
7     # Menghapus instance yang tidak berhubungan
```

```
8     drop_data_contains(df, r"[katakunci dari data yang tidak berhubungan]")
```

Kode 4.1: Kode membersihkan data teks

Selanjutnya, dilakukan pencarian *instance* yang diduga duplikat berdasarkan isi pertanyaan. Digunakan *library* difflib yang disediakan oleh Python dengan batas similaritas sebesar 0,7 untuk menandai *instance* yang diduga duplikat. Setiap *instance* yang ditandai dicek kembali secara manual untuk memastikan apakah benar-benar duplikat atau mungkin hanya kalimat yang terlalu pendek sehingga mirip dengan *instance* lain. Implementasi langkah ini dapat dilihat pada Kode 4.2.

```
1 import difflib
2
3 def check_duplicate(df):
4     df['SIM'] = [(len(difflib.get_close_matches(x, df['ISI'],
5             cutoff=0.7)) > 1) * 1 for x in df['ISI']]
6
7     return df[df['SIM'] == 1]
```

Kode 4.2: Kode mengecek similaritas pada data pertanyaan

Persiapan data yang terakhir adalah menyaring data tidak berlabel agar tidak terlalu banyak. Untuk mengatasi jumlah data yang terlalu banyak, diimplementasikan penyaringan data seperti pada potongan kode Kode 4.3. Dengan menggunakan daftar topik atau kata kunci medis pada halaman AloDokter¹, dihitung jumlah kata kunci medis untuk setiap data yang tidak memiliki label. Selanjutnya, menyimpan *instance* yang memiliki jumlah kata kunci lebih besar dari batas yang ditentukan.

```
1 def filter_by_keyword(df, keyword, k):
2     ranking = []
3     for (index, row) in df.iterrows():
4         ranking.append((index, count_keyword_in_teks(keyword, row.TEKS)))
5     new_index = [x[0] for x in ranking if x[1] > k]
6
7     return df.loc[new_index]
```

Kode 4.3: Kode menyaring *instance* dengan minimal *k* kata kunci

4.2 Uji Similaritas Data Anotasi Manusia dan Anotasi Mesin

Sebelum melakukan uji similaritas, dilakukan anotasi ulang berbasis pemetaan kamus dengan menggunakan kamus yang sudah dimodifikasi. Oleh karena itu, data teks perlu melalui proses prapemrosesan seperti penghapusan *stopword*, *stemming*, dan

¹www.alodokter.com/komunitas/topik

penggabungan teks judul dan isi. Namun, tidak semua *stopword* dihapus dari data teks karena masih diperlukan untuk kamus pemetaan. Implementasi proses pengolahan teks dan penghapusan *stopword* dapat dilihat pada Kode 4.4.

```

1 def cleaning_text(texts):
2     texts = lower_text(texts)
3     texts = remove_punctuation(texts)
4     texts = strip_text(texts)
5     custom_stopwords = ["di", "pada", "ingin", "dan", "yang", "sangat", "luar",
6                          "biasa", "bagi", "bagian", "saat", "hanya", "asa",
7                          "terasa", "saat", "dengan", "bikin", "dari", "dalam",
8                          "sering", "saya", "untuk"]
9     texts = remove_stopword(texts, custom_stopwords)
10    # Mengabaikan kata menyusui agar tidak bergabung dengan kata susu
11    ignore_stemming = ['menyusui']
12    texts = stemming(texts, ignore_stemming)
13    return texts

```

Kode 4.4: Kode prapemrosesan kalimat agar bisa dipetakan

Selanjutnya, dilakukan pemetaan kamus sebagai metode untuk mengklasifikasikan domain spesialisasi dokter. Jika label tidak dibatasi, digunakan Kode 4.5. Namun, jika label dibatasi hingga tiga, digunakan Kode 4.6 untuk pemetaan. Fungsi `is_in` digunakan untuk memeriksa apakah kata kunci terdapat dalam data teks atau tidak.

```

1 def mapping(s, mapping_table):
2     res = set()
3     for keyword, labels in mapping_table.items():
4         if is_in(keyword.lower(), s.lower()):
5             res.update(labels)
6     return list(res)

```

Kode 4.5: Kode pemetaan dengan jumlah label tidak dibatasi

```

1 import statistics
2 def mapping_top3(s, mapping_table):
3     res = {}
4     for keyword, labels in mapping_table.items():
5         if is_in(keyword.lower(), s.lower()):
6             for label in labels:
7                 if label not in res:
8                     res[label] = 0
9                     res[label] += 1
10    return filter(res)[:3]
11
12 def filter(res):
13     if len(res) == 0:
14         return []
15     elif len(res) == 1:

```

```

16     return res.keys()
17 max_value = max(res.values())
18 std = statistics.stdev(res.values())
19 top = [(key, value) for key, value in res.items() if value >= (max_value - std)]
20 return sorted(top, key=lambda x: x[1], reverse=True)

```

Kode 4.6: Kode pemetaan dengan jumlah label dibatasi hingga tiga label

Setelah itu, dilakukan pemetaan dengan menggunakan metode seperti yang ditunjukkan pada Kode 4.7 untuk setiap bagian. Hal ini dilakukan untuk membandingkan performa dari setiap bagian yang digunakan dalam pemetaan. Implementasi pemetaan kamus tersebut memungkinkan untuk melakukan uji similaritas dan mengevaluasi performa dari model yang dikembangkan.

```

1 df["CLASS_JUDUL"] = [mapping(s) for s in df["JUDUL"]]
2 df["CLASS_ISI"] = [mapping(s) for s in df["ISI"]]
3 df["CLASS_JUDUL_ISI"] = [mapping((j + " SEP " + i))
4                           for j, i in df[["JUDUL", "ISI"]].values]

```

Kode 4.7: Kode melakukan pemetaan kamus ke *dataset*

4.3 Model *Machine Learning* Konvensional

Pada setiap tahapan yang dilakukan, terdapat beberapa pilihan yang berbeda. Pada tahap pertama, yaitu prapemrosesan teks, terdapat dua proses yang dapat dijalankan, yaitu *stemming* dan penghapusan *stopword*. Implementasi pada Kode 4.8 hanya menampilkan implementasi untuk skenario *stemming* dan penghapusan *stopword*. Jika ingin melakukan skenario yang berbeda, cukup menghapus baris kode yang tidak dibutuhkan.

```

1 def cleaning_text(texts):
2     texts = lower_text(texts)
3     texts = remove_punctuation(texts)
4     texts = strip_text(texts)
5     texts = remove_stopword(texts)
6     texts = stemming(texts)
7     return texts

```

Kode 4.8: Kode prapemrosesan kalimat untuk *machine learning* konvensional

Selanjutnya pada tahap reduksi fitur, digunakan *pipeline* dan SVD dari scikit-learn. Dengan menggunakan *pipeline*, proses TF-IDF dan SVD dapat digabungkan menjadi satu objek dan dijalankan secara berurutan. Selain itu, hasil dari SVD dinormalisasi menggunakan MinMaxScaler agar mempercepat pembelajaran model. Implementasi proses tersebut dapat dilihat pada Kode 4.9.

```

1 from sklearn.pipeline import make_pipeline
2 from sklearn.preprocessing import MinMaxScaler, FunctionTransformer
3 from sklearn.feature_extraction.text import TfidfVectorizer
4 from sklearn.decomposition import TruncatedSVD
5
6 pipeline_svd = make_pipeline(
7     TfidfVectorizer(min_df=2, sublinear_tf = True, ngram_range=(1,3)),
8     # n_components diubah sesuai dengan target fitur yang diinginkan
9     TruncatedSVD(n_components=100, n_iter=10),
10    MinMaxScaler()
11 )
12 pipeline_raw = make_pipeline(
13     TfidfVectorizer(min_df=2, sublinear_tf = True, ngram_range=(1,3)),
14     FunctionTransformer(lambda x: x.toarray(), accept_sparse=True),
15     MinMaxScaler()
16 )

```

Kode 4.9: Kode pipeline untuk ekstraksi dan reduksi fitur

Setelah semua data siap digunakan, langkah selanjutnya adalah membuat model yang diinginkan. Metode *chain classifier* dan *binary relevance* yang disediakan oleh scikit-multilearn dapat diintegrasikan dengan model *machine learning* dari scikit-learn dan XGBoost. Yang perlu diubah hanyalah kelas model yang dipanggil, seperti yang ditunjukkan pada Kode 4.10.

```

1 from skmultilearn.problem_transform import ClassifierChain, BinaryRelevance
2 from xgboost import XGBClassifier
3 from sklearn.svm import SVC
4
5 clf_br = BinaryRelevance(classifier=SVC(), require_dense=[False, True])
6 clf_cc = ClassifierChain(classifier=XGBClassifier(), require_dense=[False, True])

```

Kode 4.10: Kode membuat model *machine learning* konvensional

Saat melatih model, diterapkan teknik *10-fold cross validation* agar performa yang dihasilkan mewakili performa sebenarnya. Namun karena permasalahan ini bersifat *multi-label*, perlu dibuat fungsi khusus untuk menerapkan teknik tersebut. Pada implementasi pada Kode 4.11, digunakan *IterativeStratification* yang disediakan oleh scikit-multilearn untuk melakukan *stratified k-fold* pada data *multi-label*. Dengan menggunakan metode ini, distribusi jumlah label antara data latih dan data validasi tidak terlalu berbeda. *Pipeline* yang digunakan adalah tahap reduksi fitur dan normalisasi yang telah dijelaskan sebelumnya. Terakhir, model diprediksi dan hasil rata-rata evaluasi dikembalikan untuk dibandingkan.

```

1 from skmultilearn.model_selection import IterativeStratification
2 from tqdm import tqdm

```

```

3 import numpy as np
4
5 def cross_val(model, X, y, k, pipeline):
6     score = []
7     cv = IterativeStratification(n_splits=k, order=1)
8     with tqdm(total=k, desc="Fold") as pbar:
9         for i, (train_idx, val_idx) in enumerate(cv.split(X_train, y_train)):
10            X_train = np.array([X[i] for i in train_idx])
11            y_train = np.array([y[i] for i in train_idx])
12            X_val = np.array([X[i] for i in val_idx])
13            y_val = np.array([y[i] for i in val_idx])
14            X_train = pipeline.fit_transform(X_train)
15            X_val = pipeline.transform(X_val)
16            model.fit(X_train, y_train)
17            y_pred = model.predict(X_val)
18            score.append(evaluate(y_val, y_pred))
19    return mean(score)

```

Kode 4.11: Kode penerapan *k-fold cross validation* untuk permasalahan *multi-label*

Saat membandingkan model terbaik atau urutan model terbaik, digunakan uji t-test terhadap hasil dari 10 kali percobaan. Setiap hasil percobaan dimasukkan ke dalam *dictionary* dengan nama model sebagai kunci, dan hasil performa dari 10 kali percobaan sebagai nilai. Kemudian, dilakukan uji t-test untuk setiap pasangan model, seperti yang ditunjukkan pada Kode 4.12. Dengan demikian, dapat dibuat tabel hasil berdasarkan *dictionary* yang dihasilkan.

```

1 from itertools import permutations
2 from scipy.stats import ttest_rel
3
4 def t_test(dict_values):
5     comparisons = {}
6     for (key1, key2) in permutations(dict_values.keys(), 2):
7         comparison = ttest_rel(dict_values[key1], dict_values[key2], equal_var=False)
8         comparisons[(key1, key2)] = comparison
9     return comparisons

```

Kode 4.12: Kode untuk menguji t-test secara berpasangan

4.4 Pre-Trained Language Model

Terdapat beberapa metode yang memanfaatkan *language model* dalam penelitian ini. Setiap metode memiliki perbedaan dalam penerapannya, sehingga perlu dibahas secara terpisah. Pada penelitian ini, menggunakan *library* PyTorch untuk membuat dan melatih *deep learning* berbasis *language model*.

4.4.1 BERT

Sebelum membuat model BERT, langkah pertama adalah melakukan tokenisasi yang mengubah teks menjadi vektor yang berisi id token. Panjang vektor sesuai dengan konfigurasi pada Tabel 3.2, yaitu 256 token. Jika hasil tokenisasi kurang dari 256, maka ditambahkan token PAD hingga mencapai batas maksimal. Jika hasil tokenisasi lebih dari 256, maka dipotong sehingga hanya mengandung 256 token. Selain *input id*, BERT juga membutuhkan *attention mask* yang memiliki panjang yang sama dengan *input id*. *Attention mask* berisi nilai satu atau nol, di mana satu menandakan bahwa *input id* bukan token PAD dan nol sebaliknya. Setelah *input id*, *attention mask*, dan label sudah disiapkan, selanjutnya diubah menjadi *tensor* dan dimasukkan ke dalam *dataloader*, seperti yang ditunjukkan pada Kode 4.13. Dengan menggunakan *dataloader*, tidak perlu melakukan tokenisasi untuk setiap epoch yang dijalankan.

```

1 import torch
2 from torch.utils.data import TensorDataset, DataLoader, RandomSampler
3
4
5 def create_data_loader(x, y):
6     input_ids = []
7     input_att_mask = []
8     input_label = []
9     for instance, label in zip(x, y):
10         encoded_sent = tokenizer.encode(instance,
11                                         add_special_tokens=True, max_length=MAX_SEQ_LENGTH,
12                                         padding='max_length', truncation=True)
13         input_ids.append(encoded_sent)
14         input_att_mask.append([int(token_id > 0) for token_id in encoded_sent])
15         input_label.append(label)
16     input_ids = torch.tensor(input_ids)
17     input_att_mask = torch.tensor(input_att_mask)
18     input_label = torch.tensor(input_label)
19     dataset = TensorDataset(input_ids, input_att_mask, input_label)
20     return DataLoader(dataset, sampler=RandomSampler(dataset), batch_size=BATCH_SIZE)
```

Kode 4.13: Kode membuat *dataloader* untuk masukan BERT menggunakan PyTorch

Setelah *dataset* disiapkan, langkah selanjutnya adalah membuat model BERT dan model untuk melakukan klasifikasi. Model BERT dibuat dengan memanfaatkan AutoModel yang disediakan oleh Hugging Face², sedangkan model klasifikasi hanya terdiri dari satu lapisan *dense* yang menerima keluaran dari BERT dan menghasilkan vektor klasifikasi sepanjang jumlah kelas. Kelas untuk klasifikasi dan pembuatan model

²https://huggingface.co/transformers/v3.0.2/model_doc/auto.html

ditunjukkan pada Kode 4.14.

```

1 from transformers import AutoConfig, AutoModel
2 import torch.nn as nn
3
4 class Classifier(nn.Module):
5     def __init__(self, input_size=512, output_size=3, dropout_rate=0.1):
6         super(Classifier, self).__init__()
7         layers = []
8         layers.append(nn.Linear(input_size, output_size))
9         layers.append(nn.LeakyReLU(0.2, inplace=True))
10        layers.append(nn.Dropout(dropout_rate))
11        layers.append(nn.Sigmoid())
12        self.layers = nn.Sequential(*layers)
13    def forward(self, input):
14        return self.layers(input)
15
16 config = AutoConfig.from_pretrained(MODEL_NAME)
17 transformer = AutoModel.from_pretrained(MODEL_NAME)
18 classifier = Classifier(input_size=config.hidden_size, output_size=len(categories))

```

Kode 4.14: Kode kelas dari model klasifikasi dan cara membuat model

Selanjutnya, pada Kode 4.15, perlu dibuat optimizer yang mengoptimalkan bobot saat melatih model, yaitu menggunakan optimizer AdamW. Penelitian ini juga mengikuti implementasi dari Devlin et al. (2019) yang menggunakan *warmup scheduler* pada *learning rate* dengan rasio tertentu. Setelah itu, dilakukan pelatihan model dengan menjalankan fungsi yang telah dibuat.

```

1 from transformers import get_constant_schedule_with_warmup
2 import torch
3
4 transformer_vars = [p for p in transformer.parameters()]
5 classifier_vars = [p for p in classifier.parameters()]
6 all_vars = transformer_vars + classifier_vars
7 optimizer = torch.optim.AdamW(all_vars, lr=LEARNING_RATE, weight_decay=REGULARIZATION)
8 num_train_instances = len(train_instances)
9 num_train_steps = int(num_train_instances / BATCH_SIZE * EPOCH)
10 num_warmup_steps = int(num_train_steps * WARMUP_PROPORTION)
11 scheduler = get_constant_schedule_with_warmup(optimizer,
12                                              num_warmup_steps=num_warmup_steps)
13 info = train_model(transformer, classifier, optimizer, scheduler, num_epochs=EPOCH)

```

Kode 4.15: Kode membuat optimizer dan *scheduler* beserta menjalankan pelatihan model

Saat melatih model, terdapat dua tahapan yang dilakukan, yaitu pelatihan (*training*) dan validasi (*validation*). Pada tahap pelatihan, model menggunakan *binary cross entropy* dan melakukan *backpropagation* seperti yang ditunjukkan pada potongan kode Kode 4.16. Setelah selesai dilatih, dilanjutkan dengan tahap validasi untuk mengevaluasi

sejauh mana model telah belajar dan seberapa baik performa pada data validasi yang belum pernah dilihat sebelumnya. Pada tahap validasi, dilakukan prediksi hasil keluaran dari model klasifikasi yang telah dibuat. Karena ini merupakan permasalahan *multi-label*, maka vektor klasifikasi diberikan batas ambang 0,5 agar dapat diklasifikasikan ke label-label yang relevan. Jika semua nilai elemen vektor kurang dari 0,5, maka label yang dipilih adalah elemen terbesar pada vektor klasifikasi.

```

1 import torch
2
3 (input_ids, att_mask, label) = train_dataloader.get_data()
4 bert_outputs = transformer(input_ids, attention_mask=att_mask)
5 cls_hidden_states = bert_outputs.pooler_output
6 probs = classifier(cls_hidden_states)
7
8 # Categorical Cross-Entropy
9 loss = -torch.mean(torch.sum(label * torch.log(probs) + (1 - label)
10                      * torch.log(1 - probs), dim=1))
11
12 optimizer.zero_grad()
13 loss.backward()
14 optimizer.step()
15 scheduler.step()
```

Kode 4.16: Kode tahapan melatih model

```

1 (input_ids, att_mask, label) = train_dataloader.get_data()
2 with torch.no_grad():
3     bert_outputs = transformer(input_ids, attention_mask=att_mask)
4     cls_hidden_states = bert_outputs.pooler_output
5     probs = classifier(cls_hidden_states)
6
7     # Categorical Cross-Entropy
8     loss = -torch.mean(torch.sum(batch_label * torch.log(probs) + (1
9                             - batch_label) * torch.log(1 - probs), dim=1))
10 pred_l = []
11 for prob in probs:
12     pred = [(1 if x >= 0.5 else 0) for x in prob]
13     if np.sum(pred) == 0:
14         _, idx = torch.max(prob, dim=0)
15         pred[idx] = 1
16     pred_l.append(pred)
17 pred_labels += pred_l
18 true_labels += label
19
20 loss.backward()
21 optimizer.step()
22 scheduler.step()
```

Kode 4.17: Kode tahapan validasi model

4.4.2 Domain Adaptive Pre-Training

Namun, untuk menerapkan metode DAPT, data yang sudah disaring perlu diubah menjadi *dataset* yang dapat digunakan untuk melatih model BERT. Untuk itu, dibuat potongan kode Kode 4.18 untuk mengimplementasikan pembuatan *dataset* seperti yang telah dijelaskan pada Tabel 3.4. Dalam proses ini, kata kunci SEP digunakan untuk menandakan apakah sebuah kalimat merupakan kelanjutan dari kalimat sebelumnya atau tidak. Dengan demikian, *dataset* NSP dapat dibentuk.

```

1 import re
2
3 def generate_dataset(df_values):
4     buffer = []
5     for title, content in df_values:
6         if len(title) != 0 and title[-1] not in ".!?" :
7             title += '.'
8         if len(content) != 0 and content[-1] not in ".!?" :
9             content += '.'
10        title_sentences = re.split('(?<=[.!?]) +',title)
11        content_sentences = re.split('(?<=[.!?]) +',content)
12        combined_sentences = title_sentences + ["SEP"] + content_sentences + ["SEP"]
13        # Membuat buffer
14        for sentence in combined_sentences:
15            # Jika kalimat terlalu pendek, maka akan digabung dengan kalimat sebelumnya
16            if (len(buffer) != 0 and len(sentence) < 5 and
17                sentence != 'SEP' and buffer[-1] != 'SEP') :
18                buffer[-1] += sentence
19            else:
20                buffer.append(sentence)
21        # Generator dataset
22        while len(buffer) > 2:
23            text_a = buffer.pop(0)
24            isNSP = 1
25            if (buffer[0] == 'SEP'):
26                buffer.pop(0)
27                isNSP = 0
28            text_b = buffer[0]
29            yield ({'Text A':text_a, 'Text B':text_b, 'nsp':isNSP})

```

Kode 4.18: Kode implementasi pembuatan *dataset* NSP

Setelah generator untuk NSP selesai dibuat, langkah selanjutnya adalah membuat *dataset* MLM. Berdasarkan yang telah dibahas pada Subbab 3.4.3, terdapat tiga prosedur *masking* yang dilakukan. Token yang telah di-”*masked*” menjadi masukan (*input*) model BERT, sedangkan token yang asli menjadi label (*target*) untuk prediksi BERT. Hal ini dilakukan di dalam fungsi *tokenize*, seperti yang terlihat pada potongan kode

Kode 4.19, yang dipanggil saat melatih model.

```

1 import random
2 import torch
3
4 def tokenize(a, b):
5     inputs = tokenizer(a, b, add_special_tokens = True, \
6                         max_length = MAX_SEQ_LENGTH, \
7                         padding = "max_length", truncation = True)
8     tokens = inputs['input_ids']
9     labels = []
10    # Berikan mask secara random ke setiap token
11    masked_tokens = []
12    for token in tokens:
13        if token > 4 and random.random() < 0.15:
14            labels.append(token)
15            cum_rand = random.random()
16            # Prosedur pertama
17            if (cum_rand < 0.8):
18                masked_tokens.append(4)
19            # Prosedur kedua
20            elif (cum_rand < 0.9):
21                masked_tokens.append(random.randint(5, 30520))
22            # Prosedur ketiga
23            elif (cum_rand < 1):
24                masked_tokens.append(token)
25        else:
26            labels.append(-100)
27            masked_tokens.append(token)
28    masked_tokens = torch.tensor([masked_tokens])
29    labels = torch.tensor([labels])
30    inputs = {
31        'input_ids' : masked_tokens,
32        'token_type_ids' : torch.tensor([inputs['token_type_ids']]),
33        'attention_mask' : torch.tensor([inputs['attention_mask']])
34    }
35    return {'inputs' : inputs, 'labels' : labels}

```

Kode 4.19: Kode implementasi pembuatan *dataset* MLM

Setelah semua *dataset* siap digunakan, langkah selanjutnya adalah membuat model BERTForPreTraining³ yang telah disediakan oleh Hugging Face. Proses pelatihan model pada potongan kode Kode 4.20 mirip dengan potongan kode Kode 4.15, yang membedakan hanya pada tahap pembuatan model. Namun, ada sedikit perbedaan dalam fungsi pelatihan BERT pada potongan kode Kode 4.16. Karena menggunakan *dataset* untuk MLM dan NSP, fungsi tokenisasi yang telah dibuat sebelumnya dipanggil saat melatih model, seperti yang ditunjukkan pada potongan kode Kode 4.21. Setelah model

³https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertForPreTraining

BERT selesai dilatih, tahap selanjutnya adalah *fine-tuning*, sebagaimana telah dibahas di Subbab 4.4.1 (halaman 62).

```

1 from transformers import get_constant_schedule_with_warmup, BertForPreTraining
2 import torch
3
4 model = BertForPreTraining.from_pretrained(MODEL_NAME)
5 optimizer = torch.optim.AdamW(model.parameters(),
6                               lr=LEARNING_RATE, weight_decay=REGULARIZATION)
7 num_train_instances = len(train_instances)
8 num_train_steps = int(num_train_instances / BATCH_SIZE * EPOCH)
9 num_warmup_steps = int(num_train_steps * WARMUP_PROPORTION)
10 scheduler = get_constant_schedule_with_warmup(optimizer,
11                                               num_warmup_steps=num_warmup_steps)
12 train_model(model, optimizer, num_epochs=EPOCH)

```

Kode 4.20: Kode pembuatan optimizer dan memulai melatih model DAPT

```

1 import torch
2
3 text_A, text_B, isNSP = train_dataloader.get_data()
4 tokens = tokenize(text_A, text_B)
5 X = tokens["inputs"]
6 label = tokens["labels"]
7 nsp = torch.tensor(isNSP)
8 outputs = model(**X, labels=label, next_sentence_label=nsp)
9 # Total loss yang sudah dihitung oleh BERTForPreTraining dari HuggingFace
10 loss = outputs.loss
11 optimizer.zero_grad()
12 loss.backward()
13 optimizer.step()

```

Kode 4.21: Kode melatih model BERTForPreTraining yang sudah disediakan oleh Hugging Face

4.4.3 GAN-BERT

Metode GAN-BERT membutuhkan data yang berlabel dan tidak berlabel, sehingga kedua *dataset* ini perlu digabungkan agar dapat digunakan. Namun sebelum digabungkan, perlu dibuat suatu penanda (*mask*) untuk membedakan antara data yang berlabel dan tidak berlabel saat proses penggabungan dilakukan. Dalam implementasi yang terlihat pada potongan kode Kode 4.22, *mask* bernilai satu diberikan pada data yang berlabel, sedangkan *mask* bernilai nol diberikan pada data yang tidak berlabel. Selain itu, data yang tidak berlabel diberikan *pseudolabel* dengan nilai -1 agar dapat digabungkan dalam *dataloader* yang sama.

```
1 import numpy as np
```

```

2
3 train_label_masks = np.ones(len(train_examples), dtype=bool)
4 if unlabeled_examples:
5     train_examples = list(train_examples) + list(unlabeled_examples)
6     tmp_masks = np.zeros(len(unlabeled_examples), dtype=bool)
7     tmp_labels = [[-1] * len(categories)] * len(unlabeled_examples)
8     train_label_masks = np.concatenate([train_label_masks, tmp_masks])
9     label_examples = list(label_examples) + list(tmp_labels)
10    train_dataloader = generate_data_loader(train_examples, label_examples,
11                                              train_label_masks)

```

Kode 4.22: Kode untuk menggabungkan data berlabel dan tidak berlabel menjadi data latih

Pada *framework GAN*, terdapat dua arsitektur yang perlu dibuat, yaitu generator (G) dan diskriminator (D). Berdasarkan potongan kode Kode 4.23, G dan D hanya terdiri dari *layer feed forward neural network* (FFNN) yang mengikuti implementasi yang telah dijelaskan oleh Croce et al. (2020). Namun, untuk menangani permasalahan *multi-label*, fungsi aktivasi yang awalnya menggunakan *softmax* diubah menjadi *sigmoid*. Selanjutnya, jumlah *layer* FFNN yang ingin digunakan pada model G dan D harus ditentukan terlebih dahulu sebelum pembuatan model, seperti yang terlihat pada potongan kode Kode 4.24.

```

1 import torch.nn as nn
2
3 class Generator(nn.Module):
4     def __init__(self, noise_size=100, output_size=512, hidden_sizes=[512],
5                  dropout_rate=0.1):
6         super(Generator, self).__init__()
7         layers = []
8         hidden_sizes = [noise_size] + hidden_sizes
9         for i in range(len(hidden_sizes)-1):
10            layers.extend([nn.Linear(hidden_sizes[i], hidden_sizes[i+1]),
11                           nn.LeakyReLU(0.2, inplace=True), nn.Dropout(dropout_rate)])
11         layers.append(nn.Linear(hidden_sizes[-1], output_size))
12         self.layers = nn.Sequential(*layers)
13
14     def forward(self, noise):
15         output_rep = self.layers(noise)
16         return output_rep
17
18 class Discriminator(nn.Module):
19     def __init__(self, input_size=512, hidden_sizes=[512], num_labels=2,
20                  dropout_rate=0.1):
21         super(Discriminator, self).__init__()
22         self.input_dropout = nn.Dropout(p=dropout_rate)
23         layers = []
24         hidden_sizes = [input_size] + hidden_sizes
25         for i in range(len(hidden_sizes)-1):

```

```

25         layers.extend([nn.Linear(hidden_sizes[i], hidden_sizes[i+1]),
26                           nn.LeakyReLU(0.2, inplace=True), nn.Dropout(dropout_rate)])
27     self.layers = nn.Sequential(*layers)
28     self.logit = nn.Linear(hidden_sizes[-1], num_labels+1)
29     self.sigmoid = nn.Sigmoid()
30
31     def forward(self, input_rep):
32         input_rep = self.input_dropout(input_rep)
33         last_rep = self.layers(input_rep)
34         logits = self.logit(last_rep)
35         probs = self.sigmoid(logits)
36         return last_rep, probs

```

Kode 4.23: Kode kelas arsitektur diskriminatator dan generator

```

1 from transformers import AutoConfig, AutoModel
2
3 config = AutoConfig.from_pretrained(MODEL_NAME)
4 hidden_size = int(config.hidden_size)
5 # Menentukan banyak hidden layer dari model G dan D
6 hidden_levels_g = [hidden_size for i in range(0, NUM_HIDDEN_LAYERS_G)]
7 hidden_levels_d = [hidden_size for i in range(0, NUM_HIDDEN_LAYERS_D)]
8 transformer = AutoModel.from_pretrained(MODEL_NAME)
9 generator = Generator(noise_size=NOISE_SIZE, output_size=hidden_size,
10                         hidden_sizes=hidden_levels_g, dropout_rate=OUT_DROPOUT_RATE)
11 discriminator = Discriminator(input_size=hidden_size, hidden_sizes=hidden_levels_d,
12                                 num_labels=len(categories), dropout_rate=OUT_DROPOUT_RATE)

```

Kode 4.24: Kode pembuatan arsitektur diskriminatator dan generator

Pada akhirnya, terdapat tiga model yang dilatih, yaitu model diskriminatator, generator, dan BERT. Berdasarkan implementasi yang dijelaskan oleh Croce et al. (2020), terdapat dua optimisasi yang digunakan, yaitu optimisasi untuk generator dan optimisasi untuk diskriminatator yang digabungkan dengan BERT. Hal ini diimplementasikan dalam potongan kode Kode 4.25.

```

1 from transformers import get_constant_schedule_with_warmup
2 import torch
3
4 transformer_vars = [i for i in transformer.parameters()]
5 d_vars = transformer_vars + [v for v in discriminator.parameters()]
6 g_vars = [v for v in generator.parameters()]
7 # Optimizer
8 dis_optimizer = torch.optim.AdamW(d_vars, lr=LEARNING_RATE_DISCRIMINATOR)
9 gen_optimizer = torch.optim.AdamW(g_vars, lr=LEARNING_RATE_GENERATOR)
10 # Scheduler
11 num_train_instances = len(train_dataloader)
12 num_train_steps = int(num_train_instances / BATCH_SIZE * EPOCH)
13 num_warmup_steps = int(num_train_steps * WARMUP_PROPORTION)

```

```

14 dis_scheduler = get_constant_schedule_with_warmup(dis_optimizer,
15                                     num_warmup_steps = num_warmup_steps)
16 gen_scheduler = get_constant_schedule_with_warmup(gen_optimizer,
17                                     num_warmup_steps = num_warmup_steps)
18 train_model(transformer, generator, discriminator, gen_optimizer,
19             dis_optimizer, gen_scheduler, dis_scheduler, num_epochs=EPOCH)

```

Kode 4.25: Kode untuk membuat optimizer dan *scheduler* pada masing-masing model, serta menjalankan pelatihan model

Untuk implementasi yang dijelaskan dalam penelitian ini, hanya menjelaskan modifikasi pada fungsi *loss* agar model GAN-BERT dapat menangani permasalahan *multi-label*. Untuk implementasi yang lebih lengkap, dapat melihat implementasi GAN-BERT⁴ yang telah dibuat oleh Croce et al. (2020).

Hal yang berbeda dalam memprediksi *multi-label* hanyalah *supervised loss* yang dihasilkan oleh model D . Berdasarkan *logit* dari data asli, dihitung nilai *loss* menggunakan persamaan *binary cross entropy*. Namun, karena data latih terdiri dari data yang berlabel dan tidak berlabel, perlu dilakukan penyaringan agar hanya data yang berlabel yang diikutsertakan dalam perhitungan *loss*. Dengan demikian, *supervised loss* dari model diskriminasi yang diimplementasikan dalam potongan kode Kode 4.26 sudah mempertimbangkan kasus *multi-label* ketika jumlah label lebih dari satu.

```

1 import torch
2
3 logits = D_real_probs[:, 0:-1]
4 per_example_loss = -torch.sum(label * torch.log(logits) + (1 - label) *
5                               torch.log(1 - logits), dim=-1)
6 per_example_loss = torch.masked_select(per_example_loss, mask)
7 labeled_example_count = per_example_loss.type(torch.float32).numel()
8 if labeled_example_count == 0:
9     D_L_Supervised = torch.tensor([0])
10 else:
11     D_L_Supervised = torch.div(torch.sum(per_example_loss), labeled_example_count)

```

Kode 4.26: Kode modifikasi *loss* yang diterapkan

4.5 Penentuan Kata Kunci

Untuk menentukan kata kunci, langkah pertama yang dilakukan adalah menghitung skor untuk setiap token yang ada. Token yang digunakan terdiri dari unigram, bigram, dan trigram. Untuk melakukan ini, dibuat sebuah kelas yang dapat menghitung kemunculan

⁴<https://github.com/crux82/ganbert-pytorch>

token dan mengeluarkan skor dari setiap token. Potongan kode Kode 4.27 menunjukkan implementasi dari kelas ini. Setiap token yang dimasukkan dihitung secara terpisah sesuai dengan jenis ngram token tersebut. Fungsi `get_ngram` digunakan untuk mengembalikan kamus yang sesuai dengan token yang ingin ditambahkan atau dicari. Keluaran dari fungsi `get_token` adalah skor token yang digunakan untuk menentukan kata kunci.

```

1 import numpy as np
2
3 class TokenCounter:
4     def __init__(self, category):
5         self.unigram = {}
6         self.bigram = {}
7         self.trigram = {}
8         self.total_token = np.array([0] * len(category))
9         self.token_count = {}
10        self.category = category
11
12    def add(self, token, labels):
13        ngram = self.get_ngram(token)
14        if token not in ngram:
15            ngram[token] = np.array([0] * len(self.category))
16            self.token_count[token] = 0
17        ngram[token] = np.sum([ngram[token], labels], axis=0)
18        self.token_count[token] += 1
19        self.total_token = np.sum([self.total_token, labels], axis=0)
20
21    def get_score(self, token):
22        ngram = self.get_ngram(token)
23        token_count = ngram[token]
24        norm = np.divide(token_count, self.total_token)
25        norm = np.divide(norm, self.token_count[token])
26        return norm

```

Kode 4.27: Kode kelas untuk menghitung skor dari token

Selanjutnya, perlu ada metode untuk memeriksa apakah skor yang dihasilkan memiliki kecenderungan terhadap beberapa label atau tidak. Berdasarkan pembahasan pada Subbab 3.4.1 (halaman 51), diimplementasikan fungsi `condition` seperti pada potongan kode Kode 4.28. Fungsi ini berisi kriteria yang harus dipenuhi agar suatu token dapat dikategorikan sebagai kata kunci untuk suatu label. Jika skor token memenuhi persyaratan tersebut, maka token tersebut dianggap sebagai kata kunci untuk label dengan yang berada pada indeks `idx`. Jika tidak, token tersebut tidak dianggap sebagai kata kunci. Setelah mendapatkan semua kata kunci menggunakan metode ini, perlu dilakukan pengecekan manual untuk menghilangkan kata kunci yang kurang cocok. Setelah semua proses selesai, kata kunci dapat digunakan untuk melakukan analisis

kesalahan dan memeriksa anotasi.

```

1 def condition(score, idx, thres):
2     score = np.log10(score).round()
3     if score[idx] == score.max() and np.count_nonzero(score == score.max()) <= thres:
4         return True
5     return False

```

Kode 4.28: Kode implementasi untuk menilai apakah suatu token termasuk kata kunci atau tidak

4.6 SHAP

Selanjutnya, metode terakhir yang digunakan adalah SHAP untuk memahami bagaimana model bekerja. Seperti yang dijelaskan dalam Subbab 3.5.1 (halaman 54), hanya model konvensional tanpa SVD yang dapat dijelaskan menggunakan SHAP. Namun, SHAP tidak secara otomatis menghasilkan analisis untuk data *multi-label*. Oleh karena itu, perlu dibuat implementasi SHAP khusus untuk data *multi-label* dengan menganalisis setiap label secara terpisah. Untuk melakukan ini, model BR dan CC perlu dibuat secara manual tanpa menggunakan *library* tertentu, seperti yang ditunjukkan dalam potongan kode Kode 4.29. Khusus untuk pendekatan CC, masukan *X* yang telah dimodifikasi juga perlu disimpan bersama dengan model, karena digunakan untuk analisis SHAP selanjutnya.

```

1 import numpy as np
2
3 def create_binary_model(clf, category, X, y, **kwargs):
4     models = {}
5     for i in range(len(category)):
6         clfi = clf(**kwargs)
7         clfi.fit(X, y[:, i])
8         models[category[i]] = clfi
9     return models
10
11 def create_chain_model(clf, category, X, y, **kwargs):
12     res = {}
13     new_X = X.copy()
14     for (i, col) in enumerate(category):
15         resi = {}
16         resi['X'] = new_X
17         clfi = clf(**kwargs)
18         clfi.fit(new_X, y[:, i])
19         resi['model'] = clfi
20         y_pred = clfi.predict(new_X)
21         resi[col] = resi
22         new_X = np.column_stack((new_X, y_pred))

```

```
23     return res
```

Kode 4.29: Kode implementasi BR dan CC tanpa *library*

Untuk menampilkan kontribusi fitur dengan SHAP, perlu menggunakan model dan data latih untuk menghitung kontribusi dari setiap fitur, seperti yang ditunjukkan dalam potongan kode Kode 4.30. Kemudian, potongan kode Kode 4.31 menggunakan *explainer* yang telah diperoleh untuk membuat plot kontribusi dari masing-masing fitur.

```
1 import shap
2
3 def create_explainer(clfs, category, X):
4     explainers = {}
5     for i in category:
6         explaineri = shap.Explainer(clfs[i], X)
7         explainers[i] = explaineri
8     return explainers
9
10 def create_chain_explainer(chains):
11     explainers = {}
12     for (col, chain) in chains.items():
13         explaineri = shap.Explainer(chain['model'], chain['X'])
14         explainers[col] = explaineri
15     return explainers
```

Kode 4.30: Kode penerapan SHAP *explainer* untuk kasus *multi-label*

```
1 import shap
2 import matplotlib.pyplot as plt
3
4 def summary_plots(explainers, category, X1, tfidf):
5     for i in range(len(category)):
6         cat = category[i]
7         plt.title(cat)
8         shap.summary_plot(explainers[cat].shap_values(X1), X1,
9                           feature_names=tfidf.get_feature_names_out(),
10                          plot_size=None, show=False)
11     plt.show()
12
13 def summary_chain_plots(explainers, chain_test, tfidf_chain):
14     for cat in explainers.keys():
15         plt.title(cat)
16         shap.summary_plot(explainers[cat].shap_values(chain_test[cat]),
17                           chain_test[cat],
18                           feature_names=tfidf_chain[cat],
19                           plot_size=None, show=False)
20     plt.show()
```

Kode 4.31: Kode menampilkan grafik hasil analisis SHAP untuk kasus *multi-label*

BAB 5

EKSPERIMENT DAN ANALISIS HASIL

Bab 5 menjelaskan hasil dari eksperimen yang sudah dilakukan. Pertama, Subbab 5.1 menjelaskan hasil yang didapatkan dari eksplorasi *dataset*. Selanjutnya Subbab 5.2 menerangkan hasil performa model *machine learning* konvensional maupun *deep learning* yang dilatih dengan cara *supervised learning* atau hanya menggunakan data berlabel. Setelah itu, Subbab 5.3 menjelaskan hasil model DAPT dan GAN-BERT yang memanfaatkan data tidak berlabel sebagai data latih. Terakhir, Subbab 5.4 menutup Bab 5 dengan penjelasan tentang hasil evaluasi model dengan SHAP beserta evaluasi anotasi yang sudah diberikan oleh dokter.

5.1 Kajian Dataset

Bagian ini membahas *dataset* yang sudah dijelaskan pada Subbab 3.2 (halaman 41). Hal yang dibahas berupa hasil eksplorasi seperti korelasi antar label dan mencari similaritas anotasi manusia dengan anotasi mesin.

5.1.1 Eksplorasi Label Dataset

Berdasarkan eksplorasi yang dilakukan, ditemukan bahwa terdapat inkonsistensi dalam anotasi untuk label "umum" antar pembagian *annotator* yang berbeda. Beberapa contoh inkonsistensi dapat dilihat pada Gambar 5.1. Terlihat bahwa meskipun konteks pertanyaan sama, label anotasi yang diberikan berbeda pada kelas "umum". Hal ini terlihat dari nilai *Cohen's kappa* untuk kelas "umum" pada setiap pembagian pasangan *annotator* yang tercantum pada Tabel 5.1. Terlihat bahwa nilai *Cohen's kappa* untuk kelas "umum" bahkan bisa mencapai nilai negatif, sementara nilai rata-rata kappa secara keseluruhan jauh lebih baik daripada *Cohen's kappa* untuk kelas "umum". Oleh karena itu, dalam penelitian ini diambil keputusan untuk menghapus kelas "umum" dan mengabaikan pertanyaan konsultasi yang hanya dianotasi dengan kelas "umum". Akibatnya, jumlah data anotasi manusia yang tersisa hanya 1.559 pertanyaan yang digunakan sebagai data latih.

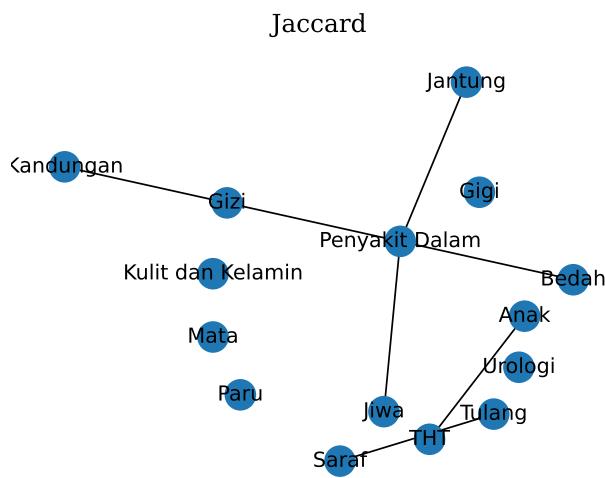
Id : KD-1753
 Isi : dok , ijin bertanya . tb 170 berat badan 55 . kenapa saya susah sekali menambah berat badan saya . berat badan saya tergolong stabil (tidak naik dan tidak turun) , apa solusinya agar dapat menambah berat badan saya ? dan dapat membuat badan saya lebih berisi . terima kasih
 Label : Gizi, Umum
 Id : KD-10114
 Isi : dok , kok saya susah naikin berat badan yah ? pola makan sama olahraga sudah saya lakukan . tetapi berat badan saya belum naik .
 Label : Gizi
 Id : TD-12671
 Isi : badang saya sering mengeluarkan keringat dengan volume yang cukup banyak pada saat sedang makan , atau secara tiba-tiba setelah ada kegiatan tertentu . apakah itu termasuk wajar atau mungkin ada kelainan ? mohon penjelasannya . terima kasih
 Label : Umum
 Id : TD-8846
 Isi : dok penyebab keringat berlebih ? cara penanggulangannya
 Label : Kulit dan Kelamin, Umum
 Id : AD-6068
 Isi : depresi itu penyebabnya apa ya dok ? obat untuk depresi apa ya dok ?
 Label : Umum
 Id : AD-5074
 Isi : halo dok saya depresi Apa obatnya dok ?
 Label : Jiwa

Gambar 5.1: Contoh anotasi manusia yang tidak konsisten

Tabel 5.1: Nilai *Cohen's kappa* kelas "umum" dan rata-rata *Cohen's kappa* keseluruhan untuk setiap pembagian *annotator* seperti yang dijelaskan pada Subbab 3.2.1 (halaman 42).

Bagian Anotasi	Kappa "Umum"	Rata-Rata Kappa
Bagian pertama	-0,1014	0,3145
Bagian kedua	0,0387	0,3153
Bagian ketiga	-0,0164	0,3927
Bagian keempat	-0,1008	0,2684

Berdasarkan eksplorasi mengenai hubungan antar label, dicari korelasi menggunakan beberapa metode yang telah disebutkan. Untuk menghemat ruang, hanya grafik yang dihasilkan oleh metode jarak Jaccard dan PPMI yang ditampilkan. Grafik yang dihasilkan oleh *Cohen's kappa* sama dengan grafik jarak Jaccard, sehingga tidak perlu ditampilkan. Untuk lebih jelasnya, grafik tersebut dapat dilihat pada Lampiran 4 (halaman: 136). Berdasarkan Gambar 5.2, terdapat dua gugusan yang terbentuk dengan menggunakan metode jarak Jaccard. Sedangkan metode PPMI menghasilkan tiga gugusan, seperti yang ditunjukkan pada Gambar 5.3. Gugusan yang ditemukan menandakan bahwa terdapat dependensi antar label yang ada.



Gambar 5.2: Korelasi yang didapatkan dengan metode jarak Jaccard. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0,3

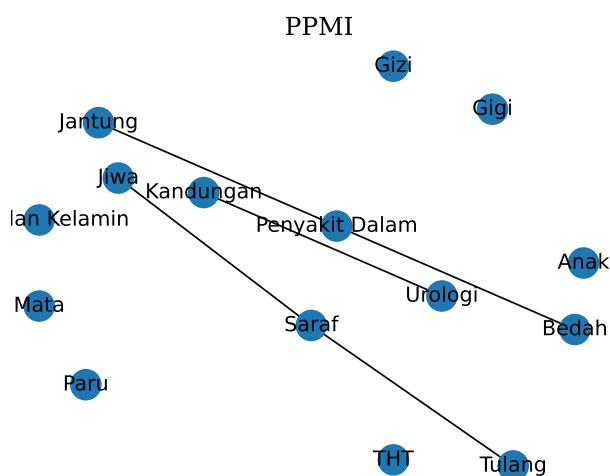
5.1.2 Similaritas anotasi manusia dengan anotasi mesin

Oleh karena itu, dalam penelitian ini, hasil similaritas yang dibahas tidak mempertimbangkan kelas "umum". Pembahasan hasil penelitian membandingkan skenario anotasi mesin yang menggunakan semua label yang ada, dengan skenario yang membatasi jumlah anotasi hingga tiga label terbanyak.

Tabel 5.2: Hasil evaluasi metrik similaritas menggunakan presisi (Prec), recall (Rec), exact match ratio (EMR), f1-score (F1), hamming loss (Hamm), dan Cohen's kappa (Kappa). Support (Sup) ditampilkan karena jumlah *instance* untuk setiap metode percobaan berbeda. Hal tersebut diakibatkan tidak semua pertanyaan menghasilkan label dari penggunaan pemetaan kamus. Untuk hasil yang menggunakan skenario membatasi tiga label diberikan tanda *top 3*.

Bagian	Prec	Rec	F1	EMR	Hamm	Kappa	Sup
Topik	0,6294	0,7421	0,6811	0,5502	0,0534	0,5892	1263
Topik (<i>top 3</i>)	0,7284	0,7044	0,7162	0,6245	0,0429	0,6074	1263
Judul	0,5930	0,7711	0,6704	0,4869	0,0580	0,6252	1415
Judul (<i>top 3</i>)	0,6392	0,7434	0,6873	0,5491	0,0518	0,6384	1415
Isi	0,4180	0,8564	0,5618	0,2795	0,1030	0,4945	1506
Isi (<i>top 3</i>)	0,5877	0,7220	0,6479	0,4807	0,0605	0,5768	1506
Judul + Isi	0,4186	0,8762	0,5665	0,2765	0,1033	0,5047	1559
Judul + Isi (<i>top 3</i>)	0,6015	0,7368	0,6623	0,4926	0,0579	0,5934	1559

Berdasarkan Tabel 5.2, dapat dilihat bahwa metode pemetaan berdasarkan topik



Gambar 5.3: Korelasi yang didapatkan dengan metode PPMI. Nilai minimum yang dibutuhkan agar terbentuk korelasi harus lebih besar dari 0

menghasilkan metrik EMR, *f1-score*, dan *hamming loss* yang lebih baik dibandingkan dengan metode lainnya. Oleh karena itu, dapat disimpulkan bahwa topik yang diberikan pada *website* memiliki kualitas yang baik dan dapat menggunakan kamus pemetaan untuk melakukan klasifikasi secara cepat dan efisien. Namun, metode pemetaan berdasarkan topik hanya terbatas pada tiga *website* tertentu dan tidak dapat digunakan untuk data lain yang tidak memiliki topik yang serupa.

Dengan begitu, skenario terbaik tanpa mempertimbangkan topik adalah melakukan pemetaan berdasarkan judul dan membatasinya hingga tiga label. Pemetaan ini menghasilkan EMR, *hamming loss*, dan *f1-score* yang tertinggi selanjutnya, bahkan mendapatkan nilai *Cohen's kappa* tertinggi dibandingkan dengan metode lainnya. Nilai *Cohen's kappa* sebesar 0,6384 dapat diinterpretasikan bahwa terdapat tingkat persetujuan yang sedang antara data anotasi manusia dan anotasi mesin, dan sekitar 35 – 63% data dapat diandalkan menurut McHugh (2012). Hal ini diduga karena judul pertanyaan pada umumnya singkat dan mencerminkan inti dari pertanyaan, sehingga pemetaan berdasarkan judul memiliki akurasi yang lebih baik daripada pemetaan berdasarkan isi pertanyaan yang cenderung lebih panjang.

Jika dibandingkan antara skenario yang dibatasi dan tidak dibatasi, terdapat beberapa hal yang dapat disimpulkan. Dapat dilihat bahwa skenario yang dibatasi memiliki performa yang lebih baik daripada skenario yang tidak dibatasi, kecuali untuk metrik

recall. Skenario yang tidak dibatasi memungkinkan suatu *instance* memiliki banyak label, sehingga menghasilkan *recall* yang tinggi, namun metrik lainnya menurun karena adanya misklasifikasi.

Maka dapat disimpulkan bahwa metode pemetaan kamus yang menggunakan judul *instance* dan membatasi jumlah label merupakan skenario terbaik. Hasil ini menunjukkan adanya tingkat similaritas yang sedang antara *dataset* anotasi oleh manusia dengan anotasi mesin. Selanjutnya, dilakukan analisis perkelas yang dapat dilihat pada Tabel 5.3. Berdasarkan Tabel 5.3, ditemukan bahwa nilai *f1-score* terendah terdapat pada label "paru". Jika dilihat lebih lanjut pada *dataset* dengan label "paru", terdapat beberapa *instance* dengan judul yang kurang deskriptif terhadap suatu penyakit, seperti yang ditunjukkan pada Gambar 5.4. Hal ini menyebabkan pemetaan menggunakan judul menjadi kurang efektif untuk label "paru".

Tabel 5.3: Nilai presisi (Prec), *recall* (Rec), & *f1-score* (F1) dari metode pemetaan kamus dengan jumlah label dibatasi sampai tiga. Data anotasi manusia dijadikan sebagai target label yang perlu diprediksi. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (*micro avg*).

Label	Prec	Rec	F1	Sup
Anak	0,7432	0,6111	0,6707	180
Bedah	0,4724	0,6861	0,5595	137
Gigi	0,9167	0,9429	0,9296	35
Gizi	0,4300	0,8558	0,5723	104
Jantung	0,4375	0,7778	0,5600	36
Jiwa	0,9500	0,6264	0,7550	91
Kandungan	0,6863	0,8185	0,7466	270
Kulit dan Kelamin	0,6920	0,8693	0,7706	199
Mata	0,9359	0,9733	0,9542	75
Paru	0,3636	0,5000	0,4211	16
Penyakit Dalam	0,5336	0,6368	0,5806	212
Saraf	0,4694	0,4600	0,4646	50
THT	0,8978	0,8483	0,8723	145
Tulang	0,7027	0,6341	0,6667	41
Urologi	0,5357	0,4412	0,4839	34
<i>Micro Avg</i>	0,6392	0,7434	0,6873	1625

Peneliti juga mencoba menggunakan metode pemetaan kamus berdasarkan judul dengan membatasi jumlah label untuk mengklasifikasikan data uji. Hasil klasifikasi

Judul : 10 kebiasaan buruk yang dapat merusak otak
 Label : Paru
 Judul : apakah ini efek dari antibiotik ?
 Label : Paru, Penyakit Dalam
 Judul : sore dokter anak saya usia 4.5 tahun dan dalam 2 bulan terakhir sudah 3 x mimisan . tepatnya tanggal 14 agst15 bangun tidur mengalami mimisan dan dalam jeda waktu +- 20
 Label : Paru, Anak
 Judul : efek samping obat
 Label : Paru
 Judul : jantung berdebar
 Label : Paru

Gambar 5.4: Lima contoh judul yang kurang menjelaskan pertanyaan dan terdapat data pada anotasi manusia dengan label "paru"

secara detail dapat dilihat pada Tabel 5.4. Berdasarkan Tabel 5.4, metode ini menghasilkan *f1-score* sebesar 0,7882, yang lebih baik daripada similaritas antar *dataset*. Terlihat juga bahwa nilai presisi dan *recall* tidak memiliki perbedaan yang signifikan, yaitu sekitar 0,78. Hal ini menunjukkan bahwa metode pemetaan kamus tidak menghasilkan banyak eror *false positive* dan *false negative*. Selanjutnya, EMR sebesar 0,6887 menunjukkan bahwa sekitar 68% data dapat diprediksi dengan benar. Hal ini juga didukung oleh hasil *hamming loss* yang sangat baik, mendekati 0, yang menunjukkan sedikit adanya misklasifikasi. Oleh karena itu, dapat ditarik kesimpulan bahwa metode pemetaan kamus berdasarkan judul dengan membatasi hingga tiga label terbanyak dapat digunakan untuk klasifikasi dan menghasilkan performa yang cukup baik.

Tabel 5.4: Hasil evaluasi 469 *instance* data uji dengan pemetaan berdasarkan judul dan banyak label yang dibatasi. Metrik yang digunakan berupa presisi (Prec), *recall* (Rec), *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Bagian	Prec	Rec	F1	EMR	Hamm
Judul (top 3)	0,7890	0,7874	0,7882	0,6887	0,0286

5.2 Pengembangan Model Prediksi Spesialisasi Dokter

Berdasarkan hasil yang didapatkan dari Subbab 5.1.2 (halaman 76), didapatkan hasil anotasi mesin tidak begitu mirip untuk digabungkan dengan data anotasi oleh manusia. Maka untuk melatih model *machine learning*, hanya menggunakan data anotasi manusia saja. Subbab 5.2 menjelaskan performa dari model *machine learning* yang sudah dikembangkan untuk permasalahan klasifikasi berdasarkan ketentuan pada Subbab 3.3 (halaman 47). Hal yang dijelaskan berupa performa dari *machine learning* konvensional beserta *language model* BERT.

5.2.1 Metode Konvensional

Dalam mengimplementasikan model konvensional, dilakukan total 168 skenario. Separuh dari skenario tersebut melibatkan pendekatan CC. Sebelumnya, perlu dicari model terbaik dengan pendekatan BR, dan kemudian dilanjutkan dengan menentukan urutan label yang tepat untuk melatih model menggunakan pendekatan CC.

Selama proses pelatihan model, ditemukan bahwa model *naive bayes* tidak memberikan performa yang baik. Oleh karena itu, tahap reduksi fitur tidak diterapkan pada model *naive bayes*. Sebagai hasilnya, terdapat total 150 skenario, dengan 75 skenario menggunakan pendekatan BR. Karena jumlah skenario yang banyak, pada Tabel 5.5 hanya ditampilkan tiga model terbaik yang diurutkan berdasarkan EMR. Setiap model diberi nama berdasarkan skenario yang digunakan dan ditulis singkat, yaitu {prapemrosesan teks}-{reduksi fitur}-{*machine learning* konvensional}-{pendekatan *multi-label*}. Oleh karena itu, model "stemstop_svd100_svc_br" digunakan untuk menguji kombinasi urutan label yang telah ditemukan.

Tabel 5.5: Hasil evaluasi 10-fold cross validation pada model dengan skenario BR dan menggunakan data anotasi manusia menjadi data latih. Metrik yang digunakan berupa *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Skenario Model	EMR	F1	Hamm
stemstop_svd100_svc_br	0,6470	0,7688	0,0304
stemstop_svd100_xgb_br	0,6308	0,7504	0,0332
stemstop_svd250_xgb_br	0,6201	0,7448	0,0334

Langkah selanjutnya yang dilakukan adalah mencari urutan label terbaik dari korelasi antar label yang sudah ditemukan pada Subbab 5.1.1 (halaman 74). Dalam mencoba urutan label terbaik, digunakan model terbaik sebelumnya, yaitu "stemstop_svd100_svc_br". Namun, setelah dilakukan uji t-test, tidak ditemukan dukungan statistik karena hasilnya tidak stabil. Oleh karena itu, dicoba menggunakan model terbaik kedua, yaitu "stemstop_svd100_xgb_br". Uji coba dilakukan dengan melatih model hanya menggunakan kelas yang ingin diuji. Model dilatih menggunakan 5-fold cross validation dan diulang hingga 10 kali. Selanjutnya, dilakukan uji t-test untuk mendapatkan urutan label terbaik, seperti yang ditunjukkan pada Tabel 5.6.

Tabel 5.6: Gugusan yang dihasilkan dari kedua metode dan urutan terbaik dari gugusan tersebut

Metode	Gugusan	Hasil Terbaik
Jarak Jaccard	[Kandungan, Gizi, Jantung, Penyakit Dalam, Bedah, Jiwa]	[Kandungan, Gizi, Jantung, Bedah, Jiwa, Penyakit Dalam]
	[Anak, Tulang, Saraf, THT]	[THT, Anak, Tulang, Saraf]
PPMI	[Jantung, Penyakit Dalam, Bedah]	[Jantung, Penyakit Dalam, Bedah]
	[Jiwa, Saraf, Tulang]	[Saraf, Jiwa, Tulang]
	[Kandungan, Urologi]	[Urologi, Kandungan]

Berikutnya, setiap gugusan terbaik dari masing-masing metode digabungkan dan ditambahkan dengan label lain yang tidak memiliki korelasi atau independen. Label independen ditempatkan di awal tanpa memperhatikan posisi, dan dilanjutkan dengan gugusan terbaik. Kemudian, dilakukan uji untuk setiap kombinasi posisi gugusan terbaik untuk mendapatkan urutan terbaik untuk metode jarak Jaccard dan PPMI secara terpisah. Terakhir, dibandingkan urutan terbaik antara metode korelasi jarak Jaccard dan PPMI. Setelah dilakukan uji tersebut, diperoleh hasil urutan terbaik berasal dari metode jarak Jaccard dengan urutan label: kulit dan kelamin, mata, paru, gigi, urologi, kandungan, gizi, jantung, bedah, jiwa, penyakit dalam, THT, anak, tulang, saraf. Urutan ini digunakan untuk melatih model *machine learning* konvensional dengan pendekatan CC.

Performa yang dihasilkan dengan metode CC digabungkan dengan hasil pendekatan BR sebelumnya. Karena hasil yang terlalu banyak, performa dari setiap skenario dapat dilihat secara keseluruhan pada Lampiran 2 (halaman 125). Model terbaik yang diuji dengan t-test didapatkan dari tiga model terbaik berdasarkan EMR, tiga model terbaik berdasarkan *f1-score*, dan tiga model terbaik berdasarkan *hamming loss*. Model-model ini dilatih ulang dan diuji sebanyak 10 kali dengan mencatat nilai EMR, *f1-score*, dan *hamming loss* terhadap data uji pada setiap percobaan. Rata-rata performa dari model-model tersebut dihasilkan dan ditampilkan pada Tabel 5.7. Selain itu, hasil metrik tersebut digunakan untuk melakukan uji t-test. Dari hasil yang didapatkan, dapat diketahui bahwa model "stemstop_svd100_svc_cc" merupakan model terbaik dari segi EMR. Hasil tersebut dibuktikan juga dengan hasil t-test yang tercantum pada Tabel 5.8a. Sedangkan model "stemstop_svd100_svc_br" merupakan model terbaik untuk metrik *f1-score* dan *hamming loss*, sesuai dengan hasil uji t-test yang terdapat pada Tabel 5.8b dan Tabel 5.8c.

Tabel 5.7: Hasil evaluasi 469 *instance* data uji dengan enam model *machine learning* konvensional terbaik. Metrik yang digunakan berupa *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Skenario Model	EMR	F1	Hamm
stemstop_svd100_svc_cc	0,8631 ± 0,0038	0,8847 ± 0,0029	0,0154 ± 0,0004
stemstop_svd250_svc_cc	0,8516 ± 0,0044	0,8849 ± 0,0037	0,0152 ± 0,0005
stem_svd100_svc_cc	0,8375 ± 0,0046	0,8694 ± 0,0050	0,0171 ± 0,0007
stemstop_svd100_svc_br	0,8213 ± 0,0039	0,8895 ± 0,0022	0,0140 ± 0,0003
stemstop_svd250_svc_br	0,8009 ± 0,0029	0,8856 ± 0,0020	0,0142 ± 0,0002
stemstop_svd500_sgd-logreg_cc	0,7599 ± 0,0815	0,8465 ± 0,0504	0,0211 ± 0,0082

Tabel 5.8: Hasil uji t-test untuk enam model terbaik berdasarkan tiga metrik yang digunakan. Tabel dibaca sebagai berikut, jika model pada baris lebih baik dibanding model pada kolom, maka diberikan simbol (+). Sebaliknya jika model pada baris lebih buruk dibanding model pada kolom, maka diberikan simbol (-). Simbol (?) diberikan jika nilai *p-value* lebih besar dari 0,05 sehingga tidak bisa dibuktikan. Model stem_svd100_svc_cc disingkat sebagai A, model stemstop_svd100_svc_br sebagai B, model stemstop_svd100_svc_cc sebagai C, model stemstop_svd250_svc_br sebagai D, model stemstop_svd250_svc_cc sebagai E, dan model stemstop_svd500_sgd-logreg_cc sebagai F

(a) Hasil uji t-test berdasarkan nilai EMR

	A	B	C	D	E	F
A	-	(+)	(-)	(+)	(-)	(+)
B	(-)	-	(-)	(+)	(-)	(?)
C	(+)	(+)	-	(+)	(+)	(+)
D	(-)	(-)	(-)	-	(-)	(?)
E	(+)	(+)	(-)	(+)	-	(+)
F	(-)	(?)	(-)	(?)	(-)	-

(b) Hasil uji t-test berdasarkan nilai *f1-score*

	A	B	C	D	E	F
A	-	(-)	(-)	(-)	(-)	(?)
B	(+)	-	(+)	(+)	(+)	(+)
C	(+)	(-)	-	(?)	(?)	(+)
D	(+)	(-)	(?)	-	(?)	(+)
E	(+)	(-)	(?)	(?)	-	(?)
F	(?)	(-)	(-)	(-)	(?)	-

(c) Hasil uji t-test berdasarkan nilai *hamming loss*

	A	B	C	D	E	F
A	-	(-)	(-)	(-)	(-)	(?)
B	(+)	-	(+)	(+)	(+)	(+)
C	(+)	(-)	-	(-)	(?)	(?)
D	(+)	(-)	(+)	-	(+)	(+)
E	(+)	(-)	(?)	(-)	-	(?)
F	(?)	(-)	(?)	(-)	(?)	-

Selain mendapatkan model dengan hasil terbaik, penelitian ini juga membandingkan performa dari masing-masing tahapan penelitian yang sudah dibahas pada Tabel 3.1.

Metrik yang digunakan untuk evaluasi adalah EMR, rerata mikro *f1-score*, dan *hamming loss* sebagai metrik yang digunakan untuk menilai model dalam klasifikasi *multi-label*. Tahapan-tahapan yang dibandingkan dengan t-test antara lain pengolahan teks, algoritma *machine learning*, dan pendekatan *multi-label*. Untuk membandingkan tahapan reduksi fitur, menggunakan metode *geomean* karena melibatkan jumlah fitur yang dipilih yang bersifat numerik.

Berdasarkan hasil yang tercantum dalam Tabel 5.10b, dapat disimpulkan bahwa metode *classifier chain* terbukti mampu meningkatkan akurasi dengan lebih baik daripada metode *binary relevance*. Namun hal tersebut tidak berlaku untuk metrik rerata mikro *f1-score* dan *hamming loss*, sebab nilai *p-value* lebih besar dari 0,05. Selain itu, pada tahap pengolahan teks, penggunaan metode penghapusan *stopword* dan *stemming* juga berhasil meningkatkan performa secara keseluruhan dibandingkan dengan metode lainnya yang terdapat dalam Tabel 5.11.

Selanjutnya, dilakukan juga uji t-test untuk setiap model yang digunakan dalam penelitian ini. Berdasarkan hasil uji t-test yang tercantum pada Tabel 5.9a, dapat disimpulkan bahwa model *logistic regression* dengan optimisasi SGD (*sgd-logreg*) merupakan model terbaik jika metrik evaluasi yang dijadikan acuan adalah EMR. Namun, ketika dilakukan uji t-test berdasarkan metrik *f1-score*, tidak dapat disimpulkan bahwa model *sgd-logreg* secara signifikan lebih baik daripada model XGBoost, seperti yang terlihat pada Tabel 5.9b.

Meskipun begitu, dalam eksperimen ini, model SVC menghasilkan performa terbaik. Namun ketika melihat hasil t-test pada Tabel 5.9, terlihat bahwa banyak *significance level* dari uji t-test terhadap model SVC yang lebih besar dari 0,05, atau bisa dikatakan bahwa tidak ada dukungan statistik yang signifikan. Hal ini menunjukkan bahwa performa model SVC sangat dipengaruhi oleh tahapan prapemrosesan yang dilakukan, sehingga hasil yang diperoleh menjadi sangat bervariasi.

Selanjutnya, berdasarkan hasil reduksi yang ditunjukkan dalam Gambar 5.5, dapat dilihat bahwa skenario yang menggunakan SVD memiliki akurasi yang lebih baik daripada skenario tanpa SVD. Namun, perlu diperhatikan bahwa performa terbaik tercapai pada SVD dengan 100 fitur, meskipun tidak terdapat perbedaan yang signifikan dengan penggunaan SVD sebanyak 250 fitur.

Tabel 5.9: Hasil uji t-test pada tahap pendekatan multi-label. Tabel dibaca sebagai berikut, jika metode pada baris lebih baik dibanding metode pada kolom, maka diberikan simbol (+). Sebaliknya jika metode pada baris lebih buruk dibanding metode pada kolom, maka diberikan simbol (-). Simbol (?) diberikan jika nilai $p\text{-value}$ lebih besar dari 0,05 sehingga tidak bisa dibuktikan.

(a) Hasil uji t-test berdasarkan EMR

	logreg	sgd-log	svc	sgd-svc	tree	xgb	nb
logreg	-	(-)	(?)	(-)	(+)	(-)	(?)
sgd-log	(+)	-	(+)	(+)	(+)	(+)	(+)
svc	(?)	(-)	-	(?)	(?)	(?)	(?)
sgd-svc	(+)	(-)	(?)	-	(+)	(?)	(?)
tree	(-)	(-)	(?)	(-)	-	(-)	(?)
xgb	(+)	(-)	(?)	(?)	(+)	-	(?)
nb	(?)	(-)	(?)	(?)	(?)	(?)	-

(b) Hasil uji t-test berdasarkan rerata mikro $f1\text{-score}$

	logreg	sgd-log	svc	sgd-svc	tree	xgb	nb
logreg	-	(-)	(?)	(-)	(?)	(-)	(?)
sgd-log	(+)	-	(+)	(+)	(+)	(?)	(+)
svc	(?)	(-)	-	(-)	(?)	(-)	(?)
sgd-svc	(+)	(-)	(+)	-	(+)	(-)	(?)
tree	(?)	(-)	(?)	(-)	-	(-)	(?)
xgb	(+)	(?)	(+)	(+)	(+)	-	(?)
nb	(?)	(-)	(?)	(?)	(?)	(?)	-

Tabel 5.10: Hasil uji t-test pada tahap pemilihan model yang digunakan berdasarkan metrik EMR. Cara membaca tabel sama seperti Tabel 5.9.

(a) Hasil uji t-test berdasarkan EMR

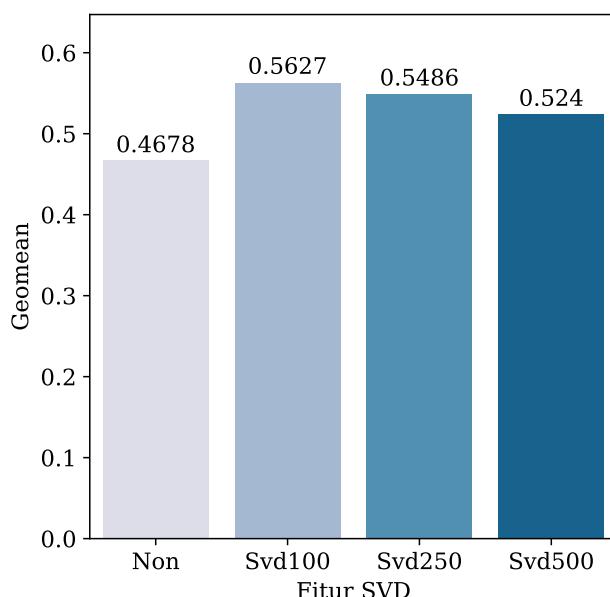
	BR	CC
BB	-	(-)
CC	(+)	-

(b) Hasil uji t-test berdasarkan rerata mikro $f1\text{-score}$ dan $hamming\ loss$ yang menghasilkan hasil yang sama

	BR	CC
BB	-	(?)
CC	(?)	-

Tabel 5.11: Hasil uji t-test pada tahap prapemrosesan teks. Hasil t-test terhadap setiap metrik memiliki hasil yang sama seperti. Cara membaca tabel sama seperti Tabel 5.9.

	Raw	Stem	Stem-stop
Raw	-	(?)	(-)
Stem	(?)	-	(-)
Stem-stop	(+)	(+)	-



Gambar 5.5: Nilai *geomean* terhadap metrik EMR untuk setiap jumlah fitur SVD yang dipilih

5.2.2 BERT

Selain menggunakan model *machine learning* konvensional, penelitian ini juga menerapkan pendekatan *deep learning* dengan *language model* BERT. Sesuai dengan pembahasan yang telah dilakukan pada Subbab 3.3.2 (halaman 49), terdapat lima model BERT di mana empat model berukuran normal (*base*) dan satu model berukuran besar (*large*). Setiap model melalui proses *fine-tuning* sebanyak lima kali, dan rata-rata serta standar deviasi dari hasil eksperimen tersebut dihitung dan dimasukkan pada Tabel 5.12. Dari Tabel 5.12, dapat dilihat bahwa model BERT-large menunjukkan performa yang lebih baik daripada model-model lainnya. Selanjutnya, hasil tersebut diuji dengan ASO. Namun, ketika menggunakan metrik EMR pada Tabel 5.13b, tidak ditemukan model terbaik dengan uji ASO. Oleh karena itu, dicoba penggunaan metrik *f1-score* dalam uji ASO, dan hasilnya tercantum pada Tabel 5.13b. Terlihat bahwa nilai ϵ_{min} dari perbandingan model IndoNLU-large dengan model lainnya kurang dari 0,5. Hasil uji ASO dengan metrik *hamming loss* juga menghasilkan nilai yang tidak jauh berbeda dengan uji ASO dengan *f1-score*. Maka disimpulkan bahwa model IndoNLU-large lebih baik dibandingkan model lain yang dicoba.

Tabel 5.12: Hasil evaluasi 469 *instance* data uji dengan lima model *supervised learning deep learning* yang masing-masing dijalankan sebanyak lima kali. Metrik yang digunakan berupa *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Nama Model	EMR	F1	Hamm
IndoNLU-base	0,8687 ± 0,0064	0,9018 ± 0,0051	0,0136 ± 0,0007
IndoWH	0,8628 ± 0,0122	0,9027 ± 0,0037	0,0136 ± 0,0006
IndoLEM	0,8601 ± 0,0102	0,8932 ± 0,0056	0,0148 ± 0,0008
IndoNLU-large	0,8682 ± 0,0117	0,9095 ± 0,0055	0,0126 ± 0,0008
mBERT	0,8200 ± 0,0226	0,8481 ± 0,0202	0,0209 ± 0,0028

Tabel 5.13: Hasil ϵ_{min} antar model yang didapatkan dari metode ASO untuk mencari model yang lebih baik. Nilai pada suatu *cell* merupakan nilai ϵ_{min} dari pernyataan: model pada baris lebih baik dibandingkan model pada kolom. Model pada baris dikatakan lebih baik dari model pada kolom ketika nilai $\epsilon_{min} \leq 0,5$. Model IndoNLU-base disingkat sebagai A, model IndoWH sebagai B, model IndoLEM sebagai C, model IndoNLU-large sebagai D, dan model mBERT sebagai E.

(a) Hasil uji ASO berdasarkan EMR

	A	B	C	D	E
A	1,00	0,622	0,280	1,00	0,00
B	1,00	1,00	0,920	1,00	0,00
C	1,00	1,00	1,00	1,00	0,001
D	0,998	0,791	0,508	1,00	0,00
E	1,00	1,00	1,00	1,00	1,00

(b) Hasil uji ASO berdasarkan *f1-score*

	A	B	C	D	E
A	1,00	1,00	0,087	1,00	0,00
B	1,00	1,00	0,073	1,00	0,00
C	1,00	1,00	1,00	0,996	0,00
D	0,030	0,012	0,013	1,00	0,00
E	1,00	1,00	1,00	1,00	1,00

5.3 Pemanfaatan Data Tidak Berlabel

Setelah mendapatkan model BERT terbaik pada Subbab 5.1.2 (halaman 76), model dikembangkan kembali dengan memanfaatkan data tidak berlabel untuk meningkatkan performa sebelumnya. Subbab 5.3 menyebutkan motivasi yang menjadi alasan digunakannya data tidak berlabel. Selain itu, dijelaskan juga hasil dari model DAPT dan

GAN-BERT yang memanfaatkan data tidak berlabel untuk meningkatkan performa *language model* BERT.

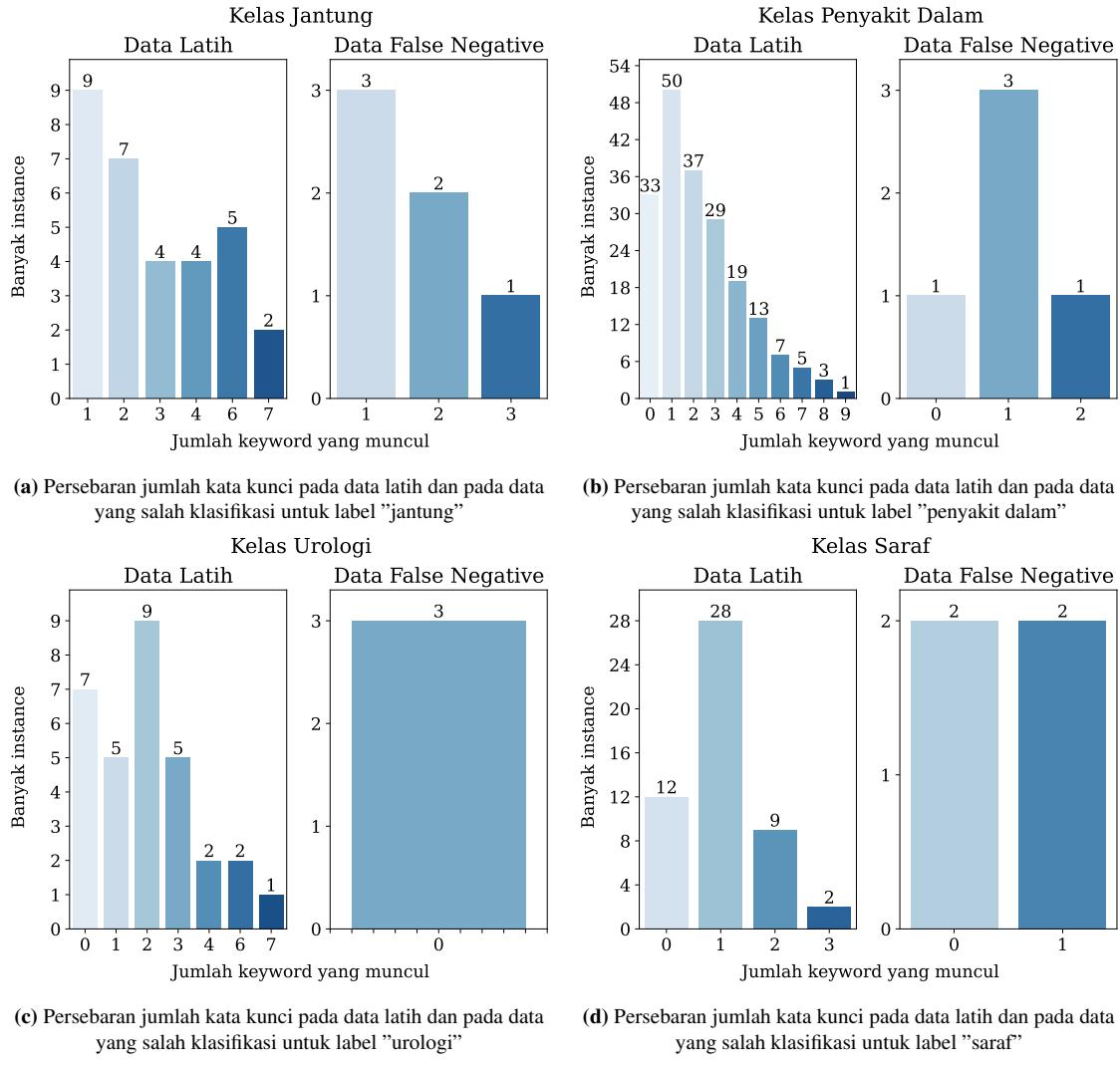
5.3.1 Analisis dengan Kata Kunci

Berdasarkan hasil yang sudah diperoleh dari penggunaan metode *deep learning* pada Subbab 5.2.2 (halaman 85), dilakukan analisis menggunakan kata kunci yang telah dibuat sebagaimana yang tercantum pada Lampiran 3 (halaman 131). Sebelumnya, model yang dianalisis adalah IndoNLU-large sebagai model BERT terbaik. Namun, karena 15 kelas terlalu banyak, sehingga analisis hanya dilakukan terhadap kelas-kelas yang memiliki performa rendah. Rincian performa untuk setiap label dapat dilihat pada Tabel 5.14.

Tabel 5.14: Nilai presisi (Prec), *recall* (Rec), & *f1-score* (F1) setiap label dari hasil prediksi model IndoNLU-large terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (*micro avg*).

Label	Prec	Rec	F1	Sup
Anak	0,9762	0,9535	0,9647	43
Bedah	0,8182	0,9310	0,8710	29
Gigi	1,0000	1,0000	1,0000	42
Gizi	0,9767	0,9767	0,9767	43
Jantung	0,8000	0,4000	0,5333	10
Jiwa	0,9286	0,9070	0,9176	43
Kandungan	0,9333	0,9767	0,9545	43
Kulit dan Kelamin	0,9773	0,9348	0,9556	46
Mata	1,0000	1,0000	1,0000	26
Paru	1,0000	1,0000	1,0000	2
Penyakit Dalam	0,7115	0,8810	0,7872	42
Saraf	0,9091	0,7143	0,8000	14
THT	0,8269	1,0000	0,9053	43
Tulang	0,9211	1,0000	0,9589	35
Urologi	0,8462	0,7857	0,8148	14
Micro Avg	0,9061	0,9347	0,9202	475

Terlihat bahwa performa terutama *recall* untuk kelas "jantung", "penyakit dalam", "urologi", dan "saraf" masih termasuk rendah dibandingkan kelas yang lain. Maka dilakukan pengecekan dengan kata kunci untuk melihat berapa banyak *instance* yang tidak mengandung kata kunci pada data latih dan pada *instance* yang misklasifikasi.



Gambar 5.6: Analisis kemunculan kata kunci untuk beberapa label dengan performa yang rendah

Berdasarkan Gambar 5.6, setiap kelas yang memiliki *false negative* terdapat *instance* yang tidak memiliki kata kunci. Meskipun begitu, masih terdapat juga *instance* data latih yang tidak mengandung kata kunci. Hal ini diduga bahwa metode pencarian kata kunci masih belum bisa menangkap semua kata kunci yang ada. Contohnya pada kelas "urologi", masih terdapat beberapa *instance* data latih dan yang tidak memiliki kata kunci yang dominan untuk beberapa *instance*. Namun seperti Gambar 5.6c, masih terdapat tiga *instance* pada *false negative* yang tidak memiliki kata kunci. Hal ini disebabkan dengan kurangnya data latih yang menyebabkan gagalnya suatu kata menjadi kata kunci, atau terdapat kata kunci yang tidak muncul sama sekali pada data latih. Dengan begitu, dapat diduga bahwa data tidak diklasifikasi sebagai kelas "urologi", sebab tidak ada kata kunci yang terdapat di *instance* tersebut. Landasan ini yang

membuat peneliti memutuskan mencoba memanfaatkan data tidak berlabel untuk membuat model dengan pendekatan *unsupervised* dan *semi-supervised*.

5.3.2 DAPT

Pemanfaatan data tidak berlabel yang pertama diterapkan adalah DAPT dengan paradigma *unsupervised learning*. DAPT dilakukan dengan melatih kembali model BERT dengan data tidak berlabel. Model BERT yang digunakan adalah model terbaik, yaitu model IndoNLU-large yang dilatih kembali dengan data tidak berlabel yang sudah disaring dan memiliki kata-kata medis. Model BERT tersebut dilatih selama 10 *epoch* dan bobot model setiap *epoch* disimpan untuk dilakukan *fine-tuning*. Setelah DAPT selesai dijalankan, maka setiap model dilakukan *fine-tuning* sebanyak lima kali agar dapat dilakukan uji ASO untuk mendapatkan *epoch* model DAPT terbaik.

Hasil dari setiap model yang dilatih sebanyak lima kali dapat dilihat pada Tabel 5.15. Model dinamai berdasarkan berapa *epoch* latih dilakukan. Jika model baru dilatih selama satu *epoch*, maka disebut daptE1 dan daptE10 menandakan model sudah dilatih sebanyak 10 *epoch*. Dapat dilihat bahwa model DAPT yang dilatih pada *epoch* keenam (daptE6) memiliki rata-rata performa EMR, *f1-score*, dan *hamming loss* yang tertinggi. Ketika diuji dengan ASO untuk metrik EMR, didapatkan hasil seperti Tabel 5.16. Menunjukkan bahwa model daptE6 memiliki EMR yang lebih baik dibandingkan model lainnya. Namun untuk model daptE7, daptE8, dan daptE9 memiliki tingkat kepercayaan yang kurang karena nilai τ masih lebih dari 0,5. Hal itu juga berlaku untuk uji ASO terhadap *f1-score* dan *hamming loss* namun tidak ditampilkan karena tidak berbeda jauh.

Setelah didapatkan bahwa model daptE6 sebagai model terbaik dengan metode *unsupervised learning*, maka dilihat performa dari setiap labelnya untuk dibandingkan dengan performa *supervised learning*. Dapat diamati bahwa terdapat perbaikan hasil pada performa Tabel 5.17. Terutama nilai *recall* jantung yang sebelumnya rendah menjadi lebih baik dari pada performa IndoNLU-large pada Tabel 5.14. Meskipun begitu, nilai *recall* dari kelas "penyakit dalam" tidak berubah, bahkan performa saraf dan urologi menurun. Meskipun begitu, secara keseluruhan, metrik presisi, *recall*, dan *f1-score* juga lebih tinggi dibanding metode *supervised learning*.

Tabel 5.15: Hasil evaluasi 469 *instance* data uji dengan 10 model *unsupervised learning deep learning* yang masing-masing dijalankan sebanyak lima kali. Metrik yang digunakan berupa *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Nama Model	EMR	F1	Hamm
daptE1	$0,8652 \pm 0,0199$	$0,9066 \pm 0,0101$	$0,0130 \pm 0,0016$
daptE2	$0,8166 \pm 0,1252$	$0,8443 \pm 0,1376$	$0,0212 \pm 0,0183$
daptE3	$0,8665 \pm 0,0117$	$0,9074 \pm 0,0056$	$0,0129 \pm 0,0009$
daptE4	$0,7522 \pm 0,2635$	$0,7832 \pm 0,2730$	$0,0294 \pm 0,0365$
daptE5	$0,6258 \pm 0,0102$	$0,7205 \pm 0,0045$	$0,0427 \pm 0,0008$
daptE6	$0,8806 \pm 0,0180$	$0,9162 \pm 0,0075$	$0,0116 \pm 0,0011$
daptE7	$0,8708 \pm 0,0228$	$0,9109 \pm 0,0104$	$0,0124 \pm 0,0016$
daptE8	$0,8627 \pm 0,0079$	$0,9012 \pm 0,0104$	$0,0137 \pm 0,0013$
daptE9	$0,8733 \pm 0,0155$	$0,9160 \pm 0,0073$	$0,0117 \pm 0,0011$
daptE10	$0,8674 \pm 0,0146$	$0,9036 \pm 0,0147$	$0,0134 \pm 0,0020$

Tabel 5.16: Hasil ϵ_{min} antar model yang didapatkan dari metode ASO untuk mencari model yang lebih baik. Metrik EMR digunakan sebagai pembanding antar model. Nilai pada suatu *cell* merupakan nilai ϵ_{min} dari pernyataan: model pada baris lebih baik dibandingkan model pada kolom. Model pada baris dikatakan lebih baik dari model pada kolom ketika nilai $\epsilon_{min} \leq 0,5$. Model DAPT pada *epoch* pertama disingkat sebagai 1, model DAPT pada *epoch* kedua sebagai 2, dan seterusnya hingga model DAPT pada *epoch* ke-10 sebagai 10.

	1	2	3	4	5	6	7	8	9	10
1	1,00	0,690	1,00	0,752	0,00	1,00	1,00	0,877	1,00	1,00
2	1,00	1,00	1,00	0,727	0,045	1,00	1,00	1,00	1,00	1,00
3	1,00	0,750	1,00	0,761	0,00	1,00	1,00	0,763	1,00	1,00
4	1,00	1,00	1,00	1,00	0,793	1,00	1,00	1,00	1,00	1,00
5	1,00	1,00	1,00	1,00	1,00	0,997	0,996	0,995	0,995	0,996
6	0,426	0,473	0,380	0,578	0,005	1,00	0,589	0,241	0,649	0,452
7	0,666	0,629	0,690	0,707	0,005	1,00	1,00	0,647	1,00	0,721
8	1,00	0,774	1,00	0,778	0,006	1,00	1,00	1,00	1,00	1,00
9	0,757	0,665	0,536	0,714	0,006	1,00	1,00	0,380	1,00	0,693
10	0,888	0,717	0,961	0,747	0,005	1,00	1,00	0,684	1,00	1,00

Setelah didapatkan bahwa model daptE6 sebagai model terbaik dengan metode *unsupervised learning*, maka dilihat performa dari setiap labelnya untuk dibandingkan dengan performa *supervised learning*. Dapat diamati bahwa terdapat perbaikan hasil

pada performa Tabel 5.17. Terutama nilai *recall* jantung yang sebelumnya rendah menjadi lebih baik dari pada performa IndoNLU-large pada Tabel 5.14. Meskipun begitu, nilai *recall* dari kelas "penyakit dalam" tidak berubah, bahkan performa saraf dan urologi menurun. Meskipun begitu, secara keseluruhan, metrik presisi, *recall*, dan *f1-score* juga lebih tinggi dibanding metode *supervised learning*.

Tabel 5.17: Nilai presisi (Prec), *recall* (Rec), & *f1-score* (F1) setiap label dari hasil prediksi model daptE6 terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (*micro avg*).

Label	Prec	Rec	F1	Sup
Anak	0,9762	0,9535	0,9647	43
Bedah	0,8438	0,9310	0,8852	29
Gigi	1,0000	1,0000	1,0000	42
Gizi	0,8776	1,0000	0,9348	43
Jantung	0,8333	1,0000	0,9091	10
Jiwa	0,9535	0,9535	0,9535	43
Kandungan	0,8936	0,9767	0,9333	43
Kulit dan Kelamin	0,9348	0,9348	0,9348	46
Mata	1,0000	1,0000	1,0000	26
Paru	1,0000	1,0000	1,0000	2
Penyakit Dalam	0,7708	0,8810	0,8222	42
Saraf	0,8182	0,6429	0,7200	14
THT	0,8936	0,9767	0,9333	43
Tulang	0,9211	1,0000	0,9589	35
Urologi	0,8333	0,7143	0,7692	14
<i>Micro Avg</i>	0,9054	0,9474	0,9259	475

5.3.3 GAN-BERT

Berdasarkan eksperimen dengan paradigma *supervised learning* (BERT) dan *unsupervised learning* (DAPT) yang dilakukan, didapatkan hasil yang cukup bagus. Maka digunakan pendekatan *semi-supervised learning* dengan metode GAN-BERT untuk melihat seberapa baik performa yang dihasilkan. Terdapat dua model BERT yang digunakan untuk metode GAN-BERT, yaitu IndoNLU-large dan daptE6 yang merupakan model BERT terbaik dari masing-masing paradigma pemelajaran *machine learning*. Maka ada dua model GAN-BERT yang dilatih masing-masing sebanyak lima

kali agar dapat dibandingkan dengan ASO.

Hasil eksperimen dari GAN-BERT tertera pada Tabel 5.18 yang berisi rata-rata dan standar deviasi dari lima kali percobaan. Dapat dikatakan bahwa model GAN-BERT dengan IndoNLU-large menghasilkan performa lebih baik dibandingkan dengan model GAN-BERT menggunakan daptE6. Namun ketika dilakukan uji ASO, didapatkan hasil seperti Tabel 5.19 yang mengatakan bahwa tidak dapat menentukan model yang lebih baik di antara kedua model yang digunakan.

Tabel 5.18: Hasil evaluasi 469 *instance* data uji dengan dua model *semi-supervised learning deep learning* yang masing-masing dijalankan sebanyak lima kali. Metrik yang digunakan berupa *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Nama Model	EMR	F1	Hamm
GAN-BERT-IndoNLU	0,8631 ± 0,014	0,9087 ± 0,0057	0,0127 ± 0,0009
GAN-BERT-DAPT	0,8554 ± 0,017	0,9090 ± 0,0082	0,0128 ± 0,0012

Tabel 5.19: Nilai ϵ_{min} yang dihasilkan ASO. Model GAN-BERT-IndoNLU diinisialkan sebagai A dan model GAN-BERT-DAPT sebagai B.

Metrik	A > B	B > A
EMR	0,564	1,00
F1	1,00	1,00
Hamm	0,564	1,00

Berdasarkan hasil tersebut, maka dipilih model GAN-BERT-IndoNLU sebagai hasil terbaik sementara untuk dianalisis setiap kelasnya. Berdasarkan hasil GAN-BERT-IndoNLU pada Tabel 5.20, terlihat bahwa nilai *recall* dari kelas "jantung", "penyakit dalam", dan "urologi", terlihat terdapat kenaikan yang cukup besar dibandingkan hasil IndoNLU-large pada Tabel 5.14. Sedangkan nilai *recall* untuk kelas "saraf" tetap sama tidak berubah. Maka dapat dikatakan, bahwa metode *semi-supervised learning* dapat mengklasifikasi *instance* yang kekurangan kata kunci dengan memanfaatkan data tidak berlabel. Meskipun begitu, terlihat adanya penurunan presisi mikro yang cukup banyak. Pada hasil IndoNLU-large, didapatkan hasil presisi mikro sebesar 0,9061, sedangkan presisi mikro GAN-BERT-IndoNLU menurun hingga 0,8677.

Tabel 5.20: Nilai presisi (Prec), *recall* (Rec), & *f1-score* (F1) setiap label dari hasil prediksi model GAN-BERT-IndoNLU terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (*micro avg*).

Label	Prec	Rec	F1	Sup
Anak	0,9286	0,9070	0,9176	43
Bedah	0,8000	0,9655	0,8750	29
Gigi	1,0000	1,0000	1,0000	42
Gizi	0,9149	1,0000	0,9556	43
Jantung	1,0000	0,7000	0,8235	10
Jiwa	1,0000	0,8372	0,9114	43
Kandungan	0,8571	0,9767	0,9130	43
Kulit dan Kelamin	0,9545	0,9130	0,9333	46
Mata	1,0000	0,9231	0,9600	26
Paru	1,0000	1,0000	1,0000	2
Penyakit Dalam	0,6119	0,9762	0,7523	42
Saraf	0,6250	0,7143	0,6667	14
THT	0,9149	1,0000	0,9556	43
Tulang	0,8333	1,0000	0,9091	35
Urologi	0,8571	0,8571	0,8571	14
<i>Micro Avg</i>	0,8677	0,9389	0,9019	475

Berdasarkan keseluruhan eksperimen yang dilakukan dengan *pre-trained language model*, masing-masing pendekatan memiliki beberapa model yang tidak bisa ditentukan model terbaiknya. Maka dipilih paling banyak dua model dari setiap pendekatan *deep learning* untuk dibandingkan. Untuk model BERT, hanya model IndoNLU-large saja yang terbaik. Sedangkan model daptE6 dan daptE9 sebagai pendekatan DAPT. Model GAN-BERT-IndoNLU dan GAN-BERT-DAPT untuk pendekatan GAN-BERT, tidak bisa ditentukan model dengan metrik terbaik. Maka hasil dari kelima skenario model dapat dirangkum sebagai Tabel 5.21. Berdasarkan rata-rata, dapat dikatakan bahwa model daptE6 lebih baik dibandingkan model yang lain. Namun berdasarkan hasil ASO Tabel 5.22, tidak bisa ditemukan model dengan performa terbaik. Namun terlihat berdasarkan metrik *f1-score* dan *hamming loss*, didapatkan model dengan pendekatan *unsupervised learning*, yaitu daptE6 dan daptE9 hampir mendominasi model lainnya.

Tabel 5.21: Hasil evaluasi 469 *instance* data uji dengan enam model *deep learning* dan dipilih dua model terbaik dari masing-masing pendekatan. Metrik yang digunakan berupa *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Nama Model	EMR	F1	Hamm
IndoNLU-large	0.8682 ± 0.0117	0.9095 ± 0.0055	0.0126 ± 0.0008
daptE6	0.8806 ± 0.0180	0.9162 ± 0.0075	0.0116 ± 0.0011
daptE9	0.8733 ± 0.0155	0.9160 ± 0.0073	0.0117 ± 0.0011
GAN-BERT-IndoNLU	0.8631 ± 0.0140	0.9087 ± 0.0057	0.0127 ± 0.0009
GAN-BERT-DAPT	0.8554 ± 0.0170	0.9090 ± 0.0082	0.0128 ± 0.0012

Tabel 5.22: Hasil ϵ_{min} antar model yang didapatkan dari metode ASO untuk mencari model yang lebih baik. Nilai pada suatu *cell* merupakan nilai ϵ_{min} dari pernyataan: model pada baris lebih baik dibandingkan model pada kolom. Model pada baris dikatakan lebih baik dari model pada kolom ketika nilai $\epsilon_{min} \leq 0,5$. Model IndoNLU-large disingkat sebagai A, model daptE6 sebagai B, model daptE9 sebagai C, model GAN-BERT-IndoNLU sebagai D, dan model GAN-BERT-DAPT sebagai E.

(a) Hasil ASO berdasarkan metrik EMR

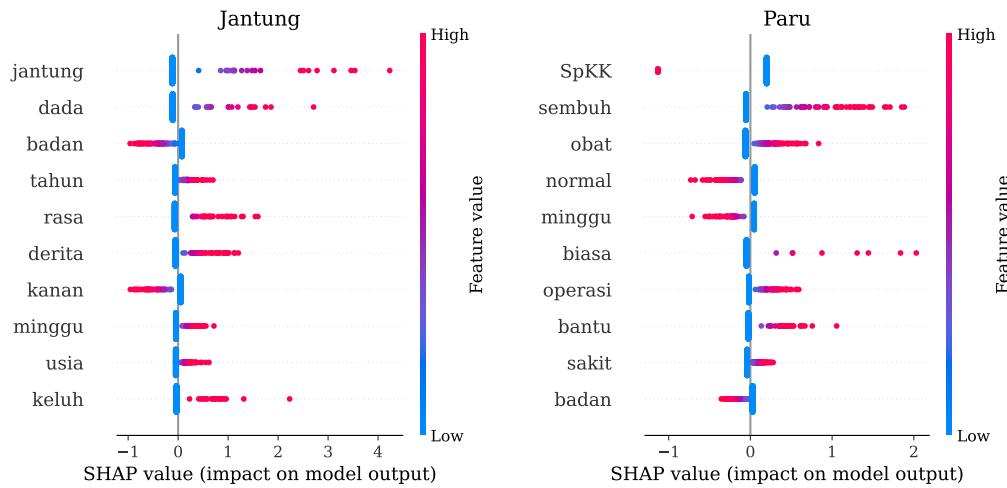
	A	B	C	D	E
A	1,00	1,00	1,00	0,810	0,299
B	0,452	1,00	0,654	0,331	0,191
C	0,696	1,00	1,00	0,477	0,236
D	1,00	1,00	1,00	1,00	0,574
E	1,00	1,00	1,00	1,00	1,00

(b) Hasil ASO berdasarkan metrik *f1-score*. Hasil ASO dengan metrik *hamming loss* tidak berbeda jauh dengan hasil berikut sehingga tidak ditampilkan.

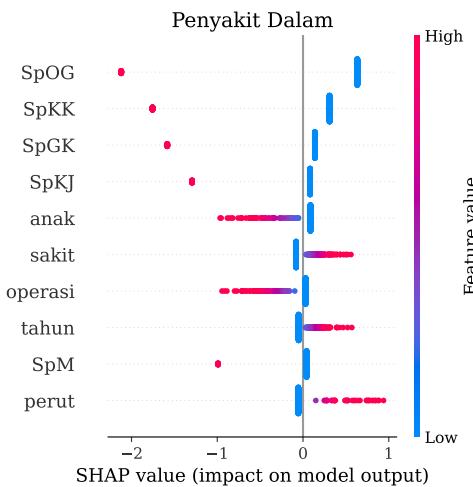
	A	B	C	D	E
A	1,00	1,00	1,00	0,945	0,983
B	0,267	1,00	1,00	0,330	0,414
C	0,303	1,00	1,00	0,241	0,305
D	1,00	1,00	1,00	1,00	1,00
E	1,00	1,00	1,00	1,00	1,00

5.4 Evaluasi Model dan Dataset

Terakhir, peneliti mengevaluasi hasil yang sudah didapatkan dari eksperimen yang dilakukan. Subbab 5.4 menjabarkan hasil yang ditemukan dengan memanfaatkan SHAP beserta mengevaluasi *dataset* yang sudah dianotasi oleh dokter.



(a) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas "Jantung" (b) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas "paru"



(c) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas "penyakit dalam"

Gambar 5.7: Hasil SHAP dari beberapa sampel kelas yang memiliki pola yang dapat digeneralisasi. Untuk mempersingkat nama fitur, hasil prediksi label sebelumnya disingkat menjadi gelar dari dokter. Yaitu kelas "anak" menjadi SpA, "bedah" menjadi SpB, "gigi" menjadi SKG, "gizi" menjadi SpGK, "jantung" menjadi SpJP, "jiwa" menjadi SpKJ, "kulit dan kelamin" menjadi SpKK, "mata" menjadi SpM, "saraf" menjadi SpN, "kandungan" menjadi SpOG, "tulang" menjadi SpOT, "paru" menjadi SpP, "penyakit dalam" menjadi SpPD, "THT" menjadi SpTHT, dan "urologi" menjadi SpU.

5.4.1 Shapley Additive Explanations (SHAP)

Setelah semua model sudah dilatih, maka langkah terakhir adalah melakukan analisis agar mengerti bagaimana model bekerja. Sesuai dengan ketentuan pada Subbab 3.5.1 (halaman 54), hanya model konvensional dan tanpa SVD saja yang dilakukan analisis. Maka berdasarkan Lampiran 2 (halaman 125), model dengan EMR terbaik adalah stemstop_non_sgd-svc_cc yang dianalisis menggunakan SHAP. Karena terdapat 15 kelas,

maka hanya beberapa kelas saja yang dianggap unik untuk dianalisis. Grafik untuk keseluruhan kelas dapat dilihat pada Lampiran 5 (halaman 142).

Berdasarkan Gambar 5.7, terdapat tiga pola yang ditemukan dari 15 kelas. Pola yang pertama seperti Gambar 5.7a, yakni kata kunci yang berkaitan memang memiliki pengaruh yang kuat dalam klasifikasi, seperti jantung, dan dada. Selanjutnya ada juga pola seperti Gambar 5.7b, yaitu tidak ada kata kunci medis yang berkontribusi untuk klasifikasi. Bahkan kata paru sendiri tidak masuk ke dalam kontribusi ke dalam 10 fitur teratas pada kelas "paru". Terakhir ada pola seperti Gambar 5.7c, di mana sebagian besar fitur yang berkontribusi adalah hasil prediksi dari kelas sebelumnya.

5.4.2 Evaluasi pada Anotasi Data

Pada tahapan terakhir, peneliti memeriksa *dataset* yang digunakan untuk mencari tahu apakah terdapat kesalahan pada proses anotasi atau tidak. Analisis dilakukan dengan mencari *false positive* dan *false negative* seperti yang sudah dijelaskan pada Subbab 3.5.2 (halaman 55). Karena keterbatasan waktu, maka hanya sebagian sampel saja yang diperiksa dengan metode ini. Untuk data yang diduga *false positive*, terdapat 38 data dan di antaranya 34 berasal dari anotasi manusia dan 4 sisanya didapatkan dari data uji. Data *false positive* dari anotasi manusia, 32 diberi dianotasi label umum, sama seperti data uji di mana dua di antaranya dianotasi label umum. Maka dapat dikatakan bahwa *instance* yang merupakan *false positive*, hanya dianotasi label umum meskipun dapat diberikan juga label lain.

Sedangkan untuk *false negative*, hanya terdapat lima data di mana semua kesalahan anotasi berasal dari anotasi manusia. Data yang termasuk *false negative* dapat dikatakan terjadi karena kesalahan dari *annotator*. Teks yang salah dapat dilihat pada Lampiran 6 (halaman 151), dan dapat terlihat kebanyakan *instance* yang salah dianotasi berasal dari kelas "kulit dan kelamin".

Setelah *dataset* sudah diperbaiki, maka melatih kembali model terbaik yang didapatkan, yaitu daptE6 untuk dilatih dengan *dataset* yang sudah diperbaiki. Berdasarkan hasil dari model daptE6 dengan data yang sudah dianotasi ulang pada Tabel 5.23, dapat dilihat adanya penurunan rata-rata performa dibanding rata-rata sebelum anotasi ulang pada Tabel 5.15. Meskipun begitu, hasil analisis dengan ASO seperti yang tertulis pada Tabel 5.25 juga belum dapat dipastikan model terbaik sebab nilai ϵ_{min} tidak ada yang mencapai 0,5.

Tabel 5.23: Hasil evaluasi 469 *instance* data uji dengan enam model *deep learning* dan dipilih dua model terbaik dari masing-masing pendekatan. Metrik yang digunakan berupa *exact match ratio* (EMR), *f1-score* (F1), dan *hamming loss* (Hamm).

Nama Model	EMR	F1	Hamm
Sebelum reanotasi	0,8806 ± 0,018	0,9162 ± 0,0075	0,0116 ± 0,0011
Setelah reanotasi	0,8790 ± 0,016	0,9152 ± 0,0074	0,0118 ± 0,0011

Ketika dilakukan analisis performa untuk setiap kelas pada Tabel 5.24, terjadi penurunan dibandingkan Tabel 5.17. Performa dari kelas "kulit dan kelamin" juga terlihat menurun meskipun sudah ditambahkan *instance* dan mengeluarkan beberapa *instance* yang salah anotasi pada data latih. Maka diduga proses anotasi ulang yang dilakukan masih salah sehingga performa tidak membaik.

Tabel 5.24: Nilai presisi (Prec), *recall* (Rec), & *f1-score* (F1) setiap label dari hasil prediksi model daptE6 terhadap data uji. Nilai untuk setiap label dirata-ratakan dengan rerata mikro (*micro avg*).

Label	Prec	Rec	F1	Sup
Anak	0,9762	0,9535	0,9647	43
Bedah	0,8235	0,9655	0,8889	29
Gigi	1,0000	1,0000	1,0000	42
Gizi	0,9348	1,0000	0,9663	43
Jantung	0,6429	0,9000	0,7500	10
Jiwa	0,9767	0,9767	0,9767	43
Kandungan	0,8936	0,9767	0,9333	43
Kulit dan Kelamin	0,8936	0,9130	0,9032	46
Mata	1,0000	0,9643	0,9818	28
Paru	1,0000	1,0000	1,0000	2
Penyakit Dalam	0,8864	0,9286	0,9070	42
Saraf	0,6000	0,8571	0,7059	14
THT	0,7925	0,9767	0,8750	43
Tulang	0,9487	1,0000	0,9737	37
Urologi	0,9000	0,6429	0,7500	14
<i>Micro Avg</i>	0,8961	0,9541	0,9242	479

Tabel 5.25: Nilai ϵ_{min} yang dihasilkan ASO. Model daptE6 yang dilatih sebelum anotasi ulang disebut sebagai A dan model yang dilatih setelah anotasi ulang sebagai B.

Metrik	A > B	B > A
EMR	0,943	1,00
F1	0,902	1,00
Hamm	0,843	1,00

BAB 6

PENUTUP

Bab 6, Penulis memaparkan kesimpulan dari penelitian dan saran untuk penelitian berikutnya. Kesimpulan yang ditemukan berasal dari hasil eksperimen yang sudah dilakukan dan dibahas pada Bab 5. Sedangkan saran yang diberikan berupa hal-hal yang masih dapat dikembangkan lagi dari penelitian ini, beserta hal-hal yang dapat dilakukan pada penelitian berikutnya.

6.1 Kesimpulan

Similaritas antara anotasi manusia (*expert*) dengan anotasi otomatis oleh mesin. Hasil penelitian menyimpulkan bahwa metode pemetaan kamus atau anotasi oleh mesin memiliki similaritas yang sedang berdasarkan nilai *Cohen's kappa* dan dapat memiliki nilai *recall* yang cukup besar. Dengan melakukan anotasi terhadap judul, didapatkan nilai *Cohen's kappa* lebih besar dari 0,6 yang diartikan sebagai similaritas sedang antara data anotasi manusia dan anotasi mesin. Sedangkan melakukan anotasi terhadap judul dan isi berhasil mendapatkan nilai *recall* yang cukup tinggi hingga 87%. Selain permasalahan similaritas, peneliti juga menyimpulkan bahwa terdapat masalah pada *dataset*, seperti permasalahan pada kelas "umum" dan kesalahan pada anotasi.

Performa model *machine learning*. Berdasarkan percobaan yang dilakukan, didapatkan bahwa model *machine learning* baik secara konvensional maupun *deep learning* mendapatkan performa yang tinggi. Untuk model konvensional, didapatkan model stemstop_svd100_svc baik dengan pendekatan BR atau CC merupakan skenario dengan hasil terbaik dari segi EMR, rerata mikro *f1-score*, dan *hamming loss*. Selanjutnya, untuk model *deep learning* dengan arsitektur BERT berhasil mendapatkan hasil yang lebih baik dibandingkan model konvensional dengan menggunakan model *pre-trained* IndoNLU-large. Didapatkan nilai EMR sekitar 0,8682 dengan f1-score mencapai 0,9095.

Performa penggunaan data tidak berlabel. Terdapat dua metode pemanfaatan data tidak berlabel, yaitu DAPT dan GAN-BERT. Ditemukan bahwa metode DAPT berhasil

mendapatkan performa yang lebih baik dibandingkan performa BERT saja. Model DAPT terbaik mendapatkan EMR sebesar 0,8806 dan *f1-score* hingga 0,9162. Sedangkan metode GAN-BERT gagal mendapatkan performa yang lebih tinggi dibandingkan model BERT. Peneliti menduga bahwa modifikasi yang dilakukan masih belum baik agar model GAN-BERT dapat menyelesaikan klasifikasi *multi-label*.

Faktor-faktor yang memengaruhi klasifikasi. Penelitian sudah menggunakan SHAP untuk mencari faktor-faktor beserta *effect size* dari fitur yang digunakan oleh model. Didapatkan tiga pola umum dari kontribusi yang dihasilkan oleh fitur. Pola yang ditemukan terdiri dari:

1. Kontribusi kata kunci medis tinggi

Kontribusi token "jantung" tinggi dalam memprediksi kelas "Jantung" pada Gambar 5.7a

2. Kontribusi kata kunci medis rendah

Kontribusi token "paru" dan "asma" rendah terhadap kelas "paru", sehingga tidak termasuk ke 10 kontribusi terbesar pada Gambar 5.7b

3. Kontribusi prediksi sebelumnya tinggi

Kontribusi label prediksi SpOG, SpKK, SpGK, dan SpKJ lebih besar dibanding fitur yang lain dalam prediksi kelas "penyakit dalam" pada Gambar 5.7c.

Kesimpulan lain yang didapatkan dari hasil SHAP, yaitu unigram lebih berpengaruh dalam prediksi dibandingkan bigram dan trigram.

6.2 Saran

Berdasarkan hasil penelitian ini, berikut ini adalah saran untuk pengembangan penelitian berikutnya:

1. Pada penelitian ini, anotasi dari data latih yang digunakan masih belum konsisten. Maka untuk penelitian selanjutnya dapat melakukan anotasi ulang, terutama dalam menentukan kelas umum yang cukup ambigu.
2. Dalam pembuatan kamus pemetaan, peneliti menggunakan bantuan *internet* dalam mencari dokter spesialis yang tepat untuk suatu kata kunci. Penelitian berikutnya

dapat mengembangkan kamus pemetaan yang melibatkan dokter agar hasil prediksi dengan metode kamus pemetaan menjadi lebih baik.

3. Ketika melatih model *machine learning* konvensional, peneliti tidak menggunakan *hyperparameter tuning* untuk mencari parameter terbaik untuk suatu model. Untuk kedepannya, peneliti dapat fokus dalam melakukan proses *hyperparameter tuning* dengan prapemrosesan teks berupa *stemming* dan menghapus *stopword* beserta SVD 100 yang sudah terbukti lebih baik dibanding yang lain.
4. Model *unsupervised learning* yang dibuat hanya menggunakan pendekatan DAPT. Penelitian selanjutnya dapat juga menerapkan *task adaptive pre-training* (TAPT) secara bersamaan untuk meningkatkan performa lebih lanjut.
5. Modifikasi GAN-BERT yang peneliti terapkan untuk permasalahan klasifikasi *multi-label* juga belum mendapatkan hasil yang baik. Untuk kedepannya dapat mencoba modifikasi lain agar model GAN-BERT dapat digunakan untuk klasifikasi *multi-label*.

DAFTAR REFERENSI

- Almuhana, H. A.-J., & Abbas, H. H. (2022). Classification of medical specialty for text medical report based on natural language processing and deep learning. *International journal of health sciences*. doi: 10.53730/ijhs.v6ns7.12476
- Bernard, E. (2021). *Introduction to machine learning*. Wolfram Media, Incorporated. Diakses dari <https://books.google.co.id/books?id=x6K7zgEACAAJ>
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016*. doi: 10.1145/2939672.2939785
- Chen, Y., Mancini, M., Zhu, X., & Akata, Z. (2022). Semi-supervised and unsupervised deep visual learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. doi: 10.1109/tpami.2022.3201576
- Cherman, E. A., Monard, M. C., & Metz, J. (2011, 04). Multi-label problem transformation methods: a case study. *CLEI Electronic Journal*, 14, 4 - 4. Diakses dari http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S0717-50002011000100005&nrm=iso doi: 10.19153/cleiej.14.1.4
- Croce, D., Castellucci, G., & Basili, R. (2020). Gan-bert: Generative adversarial learning for robust text classification with a bunch of labeled examples. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. doi: 10.18653/v1/2020.acl-main.191
- Del Barrio, E., Cuesta-Albertos, J. A., & Matrán, C. (2018). An optimal transportation approach for assessing almost stochastic order. In (Vol. 142). doi: 10.1007/978-3-319-73848-2_3
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1.
- Dror, R., Shlomov, S., & Reichart, R. (2019). Deep dominance - how to properly compare deep neural models. In A. Korhonen, D. R. Traum, & L. Màrquez

- (Eds.), *Proceedings of the 57th conference of the association for computational linguistics, ACL 2019, florence, italy, july 28- august 2, 2019, volume 1: Long papers* (pp. 2773–2785). Association for Computational Linguistics. Diakses dari <https://doi.org/10.18653/v1/p19-1266> doi: 10.18653/v1/p19-1266
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63. doi: 10.1145/3422622
- Gururangan, S., Marasovic, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., & Smith, N. A. (2020). Don't stop pretraining: Adapt language models to domains and tasks. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. doi: 10.18653/v1/2020.acl-main.740
- Hakim, Abid Nurul. (2016). Pemrosesan pertanyaan pada sistem tanya jawab bidang kesehatan dengan pendekatan pembelajaran mesin (Skripsi). *Universitas Indonesia*.
- Halim, Steven Wiryadinata. (2022). Pengenalan Entitas Kesehatan pada Indonesian Consumer Health Documents Menggunakan Continued Pre-training dan Self training (Skripsi). *Universitas Indonesia*.
- Hamarashid, H. K. (2021). Utilizing statistical tests for comparing machine learning algorithms. *Kurdistan Journal of Applied Research*. doi: 10.24017/science.2021.1.8
- Junior, J. D. C., Faria, E. R., Silva, J. A., & Cerri, R. (2017). Label powerset for multi-label data streams classification with concept drift. *Proceedings [do] 5nd Symposium on knowledge Discovery, mining and learning (KDMile)*, 5.
- Kadhim, A. I. (2018). An evaluation of preprocessing techniques for text classification. *International Journal of Computer Science and Information Security*, 16.
- Kim, Y., Kim, J. H., Kim, Y. M., Song, S., & Joo, H. J. (2023). Predicting medical specialty from text based on a domain-specific pre-trained bert. *International Journal of Medical Informatics*, 170. doi: 10.1016/j.ijmedinf.2022.104956
- Klema, V. C., & Laub, A. J. (1980). The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25. doi: 10.1109/TAC.1980.1102314
- Koto, F., Rahimi, A., Lau, J. H., & Baldwin, T. (2020, December). IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian

- NLP. , 757–770. Diakses dari <https://aclanthology.org/2020.coling-main.66> doi: 10.18653/v1/2020.coling-main.66
- Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*.
- Luaces, O., Díez, J., Barranquero, J., del Coz, J. J., & Bahamonde, A. (2012). Binary relevance efficacy for multilabel classification. *Progress in Artificial Intelligence, 1*. doi: 10.1007/s13748-012-0030-x
- Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems, 2017-December*.
- McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochemia Medica, 22*. doi: 10.11613/bm.2012.031
- Molnar, C. (2022). *Interpretable machine learning* (2nd ed.). Diakses dari <https://christophm.github.io/interpretable-ml-book>
- Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics, 7*. doi: 10.3389/fnbot.2013.00021
- Nurhayati, Syifa. (2019). Pencarian Pertanyaan Serupa Pada Forum Konsultasi Kesehatan Online Dengan Pendekatan Perolehan Informasi (Skripsi). *Universitas Indonesia*.
- Osman, A. I. A., Ahmed, A. N., Chow, M. F., Huang, Y. F., & El-Shafie, A. (2021). Extreme gradient boosting (xgboost) model to predict the groundwater levels in selangor malaysia. *Ain Shams Engineering Journal, 12*. doi: 10.1016/j.asej.2020.11.011
- Ouali, Y., Hudelot, C., & Tami, M. (2020). An overview of deep semi-supervised learning. *ArXiv, abs/2006.05278*.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Édouard Duchesnay (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*.
- Pires, T., Schlinger, E., & Garrette, D. (2019, July). How multilingual is multilingual BERT? , 4996–5001. Diakses dari <https://aclanthology.org/P19-1493> doi: 10.18653/v1/P19-1493
- Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning, 85*. doi: 10.1007/s10994-011-5256-5
- Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2021). Classifier chains: A review and perspectives. *Journal of Artificial Intelligence Research, 70*. doi: 10.1613/

- Reddy, Y., Pulabaigari, V., & B, E. (2018, 6). Semi-supervised learning: a brief review. *International Journal of Engineering Technology*, 7, 81. doi: 10.14419/ijet.v7i1.8 .9977
- Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv*, *abs/1609.04747*.
- Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. *Advances in Neural Information Processing Systems*.
- Schapire, R. E. (1999). A brief introduction to boosting. *IJCAI International Joint Conference on Artificial Intelligence*, 2.
- Szymański, P., & Kajdanowicz, T. (2019). Scikit-multilearn: A scikit-based python environment for performing multi-label classification. *Journal of Machine Learning Research*, 20.
- Tsoumakas, G., Katakis, I., & Vlahavas, I. (2006). A review of multi-label classification methods. *Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD 2006)*.
- Ulmer, D., Hardmeier, C., & Frellsen, J. (2022). deep-significance-easy and meaningful statistical significance testing in the age of neural networks. *arXiv preprint arXiv:2204.06815*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December.
- Verleysen, M., & François, D. (2005). The curse of dimensionality in data mining and time series prediction. *Lecture Notes in Computer Science*, 3512. doi: 10.1007/11494669_93
- Weng, W. H., Wagholarikar, K. B., McCray, A. T., Szolovits, P., & Chueh, H. C. (2017). Medical subdomain classification of clinical notes using a machine learning-based natural language processing approach. *BMC Medical Informatics and Decision Making*, 17. doi: 10.1186/s12911-017-0556-8
- Wilie, B., Vincentio, K., Winata, G. I., Cahyawijaya, S., Li, X., Lim, Z. Y., ... Purwarianti, A. (2020, December). IndoNLU: Benchmark and resources for evaluating Indonesian natural language understanding. , 843–857. Diakses dari

<https://aclanthology.org/2020.aacl-main.85>

- Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., & Zhu, J. (2019). Explainable ai: A brief survey on history, research areas, approaches and challenges. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11839 LNAI. doi: 10.1007/978-3-030-32236-6_51
- Xu, S., Li, Y., & Wang, Z. (2017). Bayesian multinomial naïve bayes classifier to text classification. *Lecture Notes in Electrical Engineering*, 448. doi: 10.1007/978-981-10-5041-1_57
- Zhang, D., Wang, J., & Zhao, X. (2015). Estimating the uncertainty of average f1 scores. *ICTIR 2015 - Proceedings of the 2015 ACM SIGIR International Conference on the Theory of Information Retrieval*. doi: 10.1145/2808194.2809488
- Zhang, M. L., Li, Y. K., Liu, X. Y., & Geng, X. (2018). Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12. doi: 10.1007/s11704-017-7031-7

LAMPIRAN

LAMPIRAN 1: KATA KUNCI UNTUK PEMETAAN LABEL

Tabel 1: Kamus yang digunakan untuk pemetaan

Kata Kunci	Domain Spesialisasi
Abses	Bedah
Acylovir	Kulit dan Kelamin
Aids	Penyakit Dalam
Alergi	Kulit dan Kelamin
Alergi Anak	Anak
Alergi Dingin	Kulit dan Kelamin
Alkohol	Penyakit Dalam
Alzheimer	Saraf, Jiwa
Amandel	THT
Amlodipine	Penyakit Dalam
Anak	Anak
Anak Remaja	Anak
Anemia	Penyakit Dalam
Anemia Defisiensi Besi	Penyakit Dalam
Angin Duduk	Jantung
Anosmia	THT
Ansietas	Jiwa
Anxiety	Jiwa
Anyang Anyang	Urologi, Penyakit Dalam
Artritis	Penyakit Dalam
Asam Lambung	Penyakit Dalam
Asam Mefenamat	Gigi, Kandungan
Asam Urat	Penyakit Dalam
Asi	Kandungan
Asma	Anak, Paru

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Autisme	Anak
Awat Muka	Kulit dan Kelamin
Ayang	Urologi, Penyakit Dalam
Bab	Penyakit Dalam
Badan Kurus	Gizi
Bak Sakit	Penyakit Dalam, Bedah, Urologi
Batu Empedu	Bedah
Batu Ginjal	Bedah
Batuk	THT
Bau Mulut	Penyakit Dalam, Gigi
Bayi	Anak
Bedah	Bedah
Bekas Luka	Kulit dan Kelamin
Belakang Mental	Jiwa
Bells Palsy	Saraf
Benjol	Bedah
Benjolan	Bedah
Berat Badan	Gizi
Berat Badan Ideal	Gizi
Berhubungan Suami	Kandungan
Bersalin	Kandungan
Bersin	THT
Biang Keringat	Kulit dan Kelamin
Bibir	Kulit dan Kelamin
Bidan Kandung	Kandungan
Bidur	Kulit dan Kelamin
Biduran	Kulit dan Kelamin
Bindeng	THT
Bintit	Mata
Bintitan	Mata

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Bipolar	Jiwa
Bisul	Kulit dan Kelamin
Bph	Urologi, Bedah
Bronchitis	Paru
Bronkitis	Paru
Buah Zakar	Urologi, Kulit dan Kelamin
Buang Air Besar	Penyakit Dalam
Buang Air Kecil	Urologi
Buduk	Kulit dan Kelamin
Buka Mulut	Gigi
Bumbu Masak	Gizi
Buncit	Gizi
Bunions	Bedah, Tulang
Bunuh Diri	Jiwa
Buta Warna	Mata
Cacar	Kulit dan Kelamin
Cacar Air	Kulit dan Kelamin
Cair Hidung	THT
Campak	Anak
Cantik	Kulit dan Kelamin
Cedera	Bedah
Cefadroxil	THT, Urologi, Kandungan, Kulit dan Kelamin, Tulang, Paru, Anak
Cemas	Jiwa
Chlamydia	Kulit dan Kelamin
Cholelithiasis	Bedah
Ciprofloxacin	Paru, THT, Urologi, Kulit dan Kelamin, Penyakit Dalam, Tulang
Clindamycin	Kulit dan Kelamin, Kandungan, Paru, Tulang, Penyakit Dalam, Bedah

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Combivent	Paru
Dada Sakit	Paru, Jantung
Darah Rendah	Jantung
Darah Tinggi	Jantung
Datang Bulan	Kandungan
Dehidrasi	Penyakit Dalam
Demam	Penyakit Dalam
Demam Berdarah	Penyakit Dalam
Demam Darah	Penyakit Dalam
Demam Pada Anak	Anak
Depresi	Jiwa
Dermatitis	Kulit dan Kelamin
Dexamethasone	Penyakit Dalam, Tulang, Kulit dan Kelamin
Diabetes	Penyakit Dalam
Diabetes Tipe 1	Penyakit Dalam
Diabetes Tipe 2	Penyakit Dalam
Diare	Penyakit Dalam
Diet	Gizi
Disentri	Penyakit Dalam
Disorder	Jiwa
Edar Darah	Jantung
Eksim	Kulit dan Kelamin
Endometriosis	Kandungan
Endoscopy	Bedah, Penyakit Dalam
Endoskopi	Bedah, Penyakit Dalam
Enzim Hati	Penyakit Dalam
Fat Loss dan Fitness	Gizi
Feses	Penyakit Dalam
Firusa	Bedah
Fisura	Bedah

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Flek	Kulit dan Kelamin, Kandungan
Flu	THT
Flu Singapura	THT
Fobia	Jiwa
Gagal Ginjal	Penyakit Dalam
Gagal Jantung	Jantung
Ganggu Jiwa	Jiwa
Gastritis	Penyakit Dalam
Gastroenteritis	Penyakit Dalam
Gatal	Kulit dan Kelamin
Gemuk	Gizi
Genetika	Kandungan
Gigi	Gigi
Gigi Mulut	Gigi
Gizi	Gizi
Gizi Anak	Anak
Glaukoma	Mata
Gondok	Penyakit Dalam
Gonore	Kulit dan Kelamin
Gugur	Kandungan
Gusi Darah	Gigi
Haid	Kandungan
Halusinasi	Jiwa
Hamil	Kandungan
Hamil Anggur	Kandungan
Hamil Ektopik	Kandungan
Hantu	Jiwa
Hbv	Penyakit Dalam
Hematuria	Urologi, Penyakit Dalam, Kandungan
Hepatitis	Penyakit Dalam

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Hepatitis A	Penyakit Dalam
Hepatitis B	Penyakit Dalam
Hepatitis C	Penyakit Dalam
Hernia	Bedah
Herpes Genital	Kulit dan Kelamin
Herpes Zoster	Kulit dan Kelamin
Hidrosefalus	Anak
Hidung Sumbat	THT
Hidung Tersumbat	THT
Hilang Bulu	Kulit dan Kelamin
Hipertensi	Jantung
Hipertiroidisme	Penyakit Dalam
Hipoglikemi	Penyakit Dalam
Hipoglikemia	Penyakit Dalam
Hipokondria	Jiwa
Hipotensi	Jantung
Hipotiroidisme	Penyakit Dalam
Hitam Mulut	Gigi
Hiv	Penyakit Dalam
Hormon	Kandungan
Hpv	Kandungan, Kulit dan Kelamin
Hubung Intim	Kandungan
Hubungan Intim	Kandungan
Hulu Hati	Bedah, Penyakit Dalam
Ibuprofen	Gigi, Tulang, Kandungan, Bedah, Saraf, Anak
Impetigo	Kulit dan Kelamin
Impotensi	Urologi
Imunisasi	Anak
Imunitas	Gizi
Infeksi	Penyakit Dalam

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Infeksi Gigi	Gigi
Infeksi Ginjal	Penyakit Dalam
Infeksi Jamur	Kulit dan Kelamin
Infeksi Kulit	Kulit dan Kelamin
Infeksi Mata	Mata
Infeksi Rahim	Kandungan
Infeksi Salur Kemih	Urologi
Infeksi Saluran Kemih	Urologi
Infeksi Telinga	THT
Infeksi Telinga Tengah	THT
Infertilitas	Kandungan
Influenza	THT
Inseminasi	Kandungan
Insomnia	Saraf, Jiwa
Intim	Kandungan
Intim Laki	Kulit dan Kelamin
Intim Laki Laki	Kulit dan Kelamin, Urologi
Intim Laki-Laki	Kulit dan Kelamin
Ispa	Paru, Anak
Itp	Penyakit Dalam
Janin	Kandungan
Jantung	Jantung
Jerawat	Kulit dan Kelamin
Kacamata	Mata
Kadaluarsa	Gizi
Kaki Dingin	Jiwa
Kalori	Gizi
Kandung Kemih	Urologi
Kandungan	Kandungan
Kangker Otak	Bedah, Saraf

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Kanker	Bedah, Penyakit Dalam
Kanker Darah	Penyakit Dalam, Bedah
Kanker Kulit	Kulit dan Kelamin
Kanker Otak	Bedah, Saraf
Kanker Paru	Paru, Bedah
Kanker Paru Paru	Paru, Bedah
Kanker Paru-Paru	Paru, Bedah
Kanker Payudara	Bedah, Penyakit Dalam
Kanker Prostat	Urologi, Bedah
Kanker Rahim	Kandungan, Bedah
Kanker Serviks	Bedah, Kandungan
Kanker Tulang	Tulang, Bedah
Kanker Usus Besar	Bedah, Penyakit Dalam
Katarak	Mata
Kb	Kandungan
Kebidanan Kandungan	Kandungan
Kecantikan	Kulit dan Kelamin
Kecil Paha	Gizi
Keguguran	Kandungan
Kehamilan	Kandungan
Kehamilan Ektopik	Kandungan
Kejang	Saraf
Kelahiran	Kandungan
Kelamin	Urologi, Kulit dan Kelamin
Kelamin Lelaki	Kulit dan Kelamin, Urologi
Kelenjar Getah Bening	THT, Bedah, Penyakit Dalam
Keloid	Bedah, Kulit dan Kelamin
Kencing Sedikit	Urologi
Kentut	Penyakit Dalam
Keputihan	Kandungan

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Keringat	Kulit dan Kelamin
Kesehatan Jiwa	Jiwa
Kesehatan Mental	Jiwa
Keseleo	Bedah
Kesemutan	Saraf
Ketoconazole	Kulit dan Kelamin, Penyakit Dalam
Ketombe	Kulit dan Kelamin
Khayal	Jiwa
Kista	Kandungan
Kista Ovarium	Kandungan
Kolesterol	Jantung, Penyakit Dalam
Kolesterol Tinggi	Jantung, Penyakit Dalam
Komedo	Kulit dan Kelamin
Konstipasi	Penyakit Dalam
Kontrasepsi	Kandungan
Kram	Tulang
Krim Malam	Kulit dan Kelamin
Krim Pagi	Kulit dan Kelamin
Kudis	Kulit dan Kelamin
Kulit	Kulit dan Kelamin
Kulit Bersisik	Kulit dan Kelamin
Kulit Dan Kelamin	Kulit dan Kelamin
Kulit Sisik	Kulit dan Kelamin
Kurap	Kulit dan Kelamin
Kutil	Kulit dan Kelamin
Kutil Kelamin	Kulit dan Kelamin
Kutu Rambut	Kulit dan Kelamin
Lahir	Kandungan
Laktasi	Kandungan
Lansoprazole	Penyakit Dalam

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Lensa Kontak	Mata
Lhran	Kandungan
Linu	Tulang, Penyakit Dalam
Lipoma	Bedah
Liver	Penyakit Dalam
Lulur	Kulit dan Kelamin
Lupus	Penyakit Dalam
Maag	Penyakit Dalam
Makan	Gizi
Malaria	Penyakit Dalam
Mancung Hidung	Bedah
Masalah Jiwa	Jiwa
Masalah Kulit	Kulit dan Kelamin
Mata	Mata
Mata Merah	Mata
Melahirkan	Kandungan
Meningitis	Penyakit Dalam, Saraf
Menopause	Kandungan
Menstruasi	Kandungan
Menyusui	Kandungan
Merencanakan Kehamilan	Kandungan
Merokok	Paru
Metabolisme	Gizi
Micin Bodoh	Gizi
Miconazole	Kulit dan Kelamin
Mie Instan	Gizi
Migrain	Saraf
Mimisan	THT
Miom	Kandungan
Mual	Penyakit Dalam

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Muda Marah	Jiwa
Muntah	Penyakit Dalam
Neurogenik	Saraf
Neuropati	Saraf
Ngilu Pusar	Penyakit Dalam
Nutrisi	Gizi
Nyeri Dada	Jantung
Nyeri Haid	Kandungan
Nyeri Sendi	Penyakit Dalam, Tulang
Nystatin	Kulit dan Kelamin, THT, Gigi
Obesitas	Gizi
Omeprazole	Penyakit Dalam
Onkologi	Bedah, Penyakit Dalam
Operasi	Bedah
Osteoarthritis	Tulang
Osteoporosis	Tulang
Otot Lengan	Tulang
Ovulasi	Kandungan
Panu	Kulit dan Kelamin
Paracetamol	Anak, Kandungan, Tulang, Gigi
Patah Tulang	Tulang
Payudara	Bedah
Pemutihan Kulit	Kulit dan Kelamin
Peningkatan Berat Badan	Gizi
Penyakit Dalam	Penyakit Dalam
Penyakit Ginjal	Penyakit Dalam
Penyakit Jantung	Jantung
Penyakit Kuning	Penyakit Dalam
Penyakit Mata	Mata
Penyakit Menular Seksual	Kulit dan Kelamin

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Penyakit Radang Pinggul	Tulang
Perawatan Muka	Kulit dan Kelamin
Peredaran Darah	Jantung
Perih Bak	Penyakit Dalam, Bedah, Urologi
Perih Kencing	Penyakit Dalam, Bedah, Urologi
Perut Kembung	Penyakit Dalam
Pilek	THT
Pingsan	Penyakit Dalam, Jantung, Saraf
Pita Suara	THT
Plasenta Previa	Kandungan
Pneumonia	Anak, Paru
Polip	THT
Ppok	Paru
Prostat	Urologi
Psikiater	Jiwa
Psikiatri	Jiwa
Psikologi Seks Dan Kawin	Jiwa
Psikologi Seks Dan Perkawinan	Jiwa
Psoriasis	Kulit dan Kelamin
Punggung Sakit	Tulang, Saraf
Pusing	Saraf
Putih	Kandungan
Putih Kulit	Kulit dan Kelamin
Rabies	Penyakit Dalam, Saraf
Rabun Jauh	Mata
Radang Gusi	Gigi
Radang Otak	Saraf
Radang Usus	Penyakit Dalam
Rahang	Gigi

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Rahim	Kandungan
Rambut	Kulit dan Kelamin
Rambut Rontok	Kulit dan Kelamin
Rematik Artritis	Tulang
Rencana Hamil	Kandungan
Rheumatoid Arthritis	Tulang
Rhinitis	THT
Rokok	Paru
Rongga Mulut	Gigi
Rproduksi	Kandungan
Ruam Kulit	Kulit dan Kelamin
Rubella	Penyakit Dalam
Sakit Badan	Tulang, Saraf
Sakit Bak	Penyakit Dalam, Bedah, Urologi
Sakit Dada	Paru, Jantung
Sakit Dalam	Penyakit Dalam
Sakit Gigi	Gigi
Sakit Ginjal	Penyakit Dalam
Sakit Jantung	Jantung
Sakit Kaki	Tulang
Sakit Kepala	Saraf
Sakit Kepala Tegang	Saraf
Sakit Kuning	Penyakit Dalam
Sakit Lambung	Penyakit Dalam
Sakit Leher	Saraf
Sakit Lutut	Tulang
Sakit Maag	Penyakit Dalam
Sakit Malu	Kulit dan Kelamin, Urologi
Sakit Mata	Mata
Sakit Perut	Penyakit Dalam

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Sakit Pingang	Penyakit Dalam
Sakit Pinggang	Penyakit Dalam
Sakit Radang Pinggul	Tulang
Sakit Tel	Gigi, THT
Sakit Telapak Kaki	Tulang
Sakit Telinga	THT
Sakit Tenggorokan	THT
Sakit Tular Seksual	Kulit dan Kelamin
Salbutamol	Paru
Salin	Kandungan
Salur Kemih	Urologi
Sariawan	Gigi
Sedih	Jiwa
Sehat Jiwa	Jiwa
Sehat Mental	Jiwa
Seks	Kandungan, Kulit dan Kelamin
Sel Telur	Kandungan
Selalu Kencing	Urologi
Semut	Saraf
Serang Jantung	Jantung
Serang Panik	Jiwa
Serangan Jantung	Jantung
Serangan Panik	Jiwa
Sering Lupa	Saraf
Sesak Dada	Paru
Sesak Nafas	Paru
Sesak Napas	Paru
Sgot	Penyakit Dalam
Sgpt	Penyakit Dalam
Sifilis	Kulit dan Kelamin

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Sinus	THT
Sinusitis	THT
Sirosis	Penyakit Dalam
Skizofrenia	Jiwa
Skoliosis	Tulang
Sperma	Kandungan, Urologi
Stres	Jiwa
Stroke	Saraf, Penyakit Dalam
Suara Hilang	THT
Sukralfat	Penyakit Dalam
Sulit Ingat	Saraf
Sulit Nafas	THT
Sunat	Kulit dan Kelamin
Suntik Silikon	Kulit dan Kelamin
Suplemen	Gizi
Susu	Gizi, Kandungan
Tahi Lalat	Kulit dan Kelamin
Tekan Darah	Jantung, Penyakit Dalam
Tekanan Darah	Jantung, Penyakit Dalam
Telat Datang Bulan	Kandungan
Telinga	THT
Telinga Berdenging	THT
Telinga Denging	THT
Tenggorok	THT
Tenggorokan	THT
Tetanus	Penyakit Dalam
Thalassemia	Anak
Tht	THT
Tifus	Penyakit Dalam
Tinggi Badan	Gizi

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
Tingkat Berat Badan	Gizi
Tonsilitis	THT
Trigliserida	Penyakit Dalam
Trikomoniasis	Kulit dan Kelamin
Tuberkulosis	Penyakit Dalam
Tubuh Kurus	Gizi
Tukak Lambung	Penyakit Dalam
Tulang	Tulang
Tumor	Bedah
Tumor Otak	Saraf, Bedah
Tumor Payudara	Bedah
Ulu Hati	Bedah, Penyakit Dalam
Urethral	Bedah, Penyakit Dalam
Urin	Urologi
Usus Buntu	Bedah
Vagina	Kulit dan Kelamin, Kandungan
Vaginitis	Kulit dan Kelamin
Vaginosis	Kulit dan Kelamin
Vaksin	Anak
Vaksinasi	Anak
Varikokel	Urologi, Bedah
Varises	Jantung
Vertigo	THT, Saraf
Vitamin A	Mata
Vitamin C	Kulit dan Kelamin
Vitamin E	Kulit dan Kelamin
Vitiligo	Kulit dan Kelamin
Wajah	Penyakit Dalam
Wasir	Bedah
biasa onani	Jiwa

Dilanjutkan pada halaman berikutnya

Tabel 1 – lanjutan dari halaman sebelumnya

Kata Kunci	Domain Spesialisasi
bintik bintik	Kulit dan Kelamin
cairanx pnuh v	Kandungan
celaka motor	Penyakit Dalam, Tulang
dahak ada darah	THT, Penyakit Dalam
erythromycin	Kulit dan Kelamin, Penyakit Dalam, Urologi
filler hidung	Kulit dan Kelamin
ganggu lihat	Mata
insulin	Penyakit Dalam
jari kaki kaku	Penyakit Dalam
lepas iud	Kandungan
ligamen	Tulang
ludah campur darah	THT, Penyakit Dalam
manusia paling dosa	Jiwa
mr p	Kulit dan Kelamin
nyeri	Penyakit Dalam
promag	Penyakit Dalam
shabu	Jiwa
tanam benang	Kulit dan Kelamin
tendon	Tulang
thorax	Paru

LAMPIRAN 2: PERFORMA MODEL *MACHINE LEARNING* KONVENTIONAL

Tabel 2: Performa setiap skenario yang dijalankan untuk model *machine learning* konvensional dengan *10-fold cross validation* menggunakan data anotasi manusia menjadi data latih

Penamaan Model	EMR	F1	Hamm
raw_svd100_logreg_br	0,4469	0,5979	0,0450
raw_svd100_sgd-logreg_br	0,5692	0,7028	0,0391
raw_svd100_svc_br	0,6052	0,7317	0,0338
raw_svd100_sgd-svc_br	0,5521	0,7046	0,0420
raw_svd100_xgb_br	0,5578	0,6974	0,0385
raw_svd100_tree_br	0,4397	0,6087	0,0508
raw_svd250_logreg_br	0,4579	0,6090	0,0442
raw_svd250_sgd-logreg_br	0,5847	0,7214	0,0367
raw_svd250_svc_br	0,5603	0,6984	0,0366
raw_svd250_sgd-svc_br	0,5214	0,6764	0,0428
raw_svd250_xgb_br	0,5496	0,6877	0,0384
raw_svd250_tree_br	0,4485	0,5973	0,0502
raw_svd500_logreg_br	0,4558	0,6079	0,0443
raw_svd500_sgd-logreg_br	0,5841	0,7229	0,0360
raw_svd500_svc_br	0,4334	0,5800	0,0462
raw_svd500_sgd-svc_br	0,4804	0,6588	0,0468
raw_svd500_xgb_br	0,5270	0,6661	0,0404
raw_svd500_tree_br	0,4492	0,5954	0,0498
raw_non_logreg_br	0,4388	0,5934	0,0460
raw_non_sgd-logreg_br	0,5051	0,6538	0,0438
raw_non_svc_br	0,1425	0,2321	0,0671
raw_non_sgd-svc_br	0,5286	0,6775	0,0427
raw_non_xgb_br	0,5676	0,7213	0,0378

Dilanjutkan pada halaman berikutnya

Tabel 2 – lanjutan dari halaman sebelumnya

Penamaan Model	EMR	F1	Hamm
raw_non_nb_br	0,4881	0,6343	0,0441
raw_non_tree_br	0,4699	0,6731	0,0509
stem_svd100_logreg_br	0,4546	0,6077	0,0443
stem_svd100_sgd-logreg_br	0,5827	0,7138	0,0378
stem_svd100_svc_br	0,6068	0,7367	0,0334
stem_svd100_sgd-svc_br	0,5453	0,6920	0,0421
stem_svd100_xgb_br	0,5654	0,6975	0,0380
stem_svd100_tree_br	0,4355	0,5993	0,0535
stem_svd250_logreg_br	0,4658	0,6194	0,0433
stem_svd250_sgd-logreg_br	0,5624	0,7179	0,0382
stem_svd250_svc_br	0,5620	0,6975	0,0368
stem_svd250_sgd-svc_br	0,5043	0,6647	0,0442
stem_svd250_xgb_br	0,5365	0,6766	0,0396
stem_svd250_tree_br	0,4486	0,6046	0,0502
stem_svd500_logreg_br	0,4598	0,6074	0,0445
stem_svd500_sgd-logreg_br	0,5961	0,7297	0,0350
stem_svd500_svc_br	0,4462	0,5875	0,0457
stem_svd500_sgd-svc_br	0,5052	0,6549	0,0447
stem_svd500_xgb_br	0,5245	0,6619	0,0407
stem_svd500_tree_br	0,4182	0,5845	0,0512
stem_non_logreg_br	0,4490	0,6009	0,0454
stem_non_sgd-logreg_br	0,5067	0,6531	0,0440
stem_non_svc_br	0,1589	0,2572	0,0659
stem_non_sgd-svc_br	0,5401	0,6781	0,0427
stem_non_xgb_br	0,5537	0,7131	0,0388
stem_non_nb_br	0,4689	0,6175	0,0454
stem_non_tree_br	0,4908	0,6844	0,0493
stemstop_svd100_logreg_br	0,5528	0,6919	0,0375
stemstop_svd100_sgd-logreg_br	0,6195	0,7525	0,0349
stemstop_svd100_svc_br	0,6470	0,7688	0,0304

Dilanjutkan pada halaman berikutnya

Tabel 2 – lanjutan dari halaman sebelumnya

Penamaan Model	EMR	F1	Hamm
stemstop_svd100_sgd-svc_br	0,5885	0,7421	0,0384
stemstop_svd100_xgb_br	0,6308	0,7504	0,0332
stemstop_svd100_tree_br	0,5011	0,6565	0,0489
stemstop_svd250_logreg_br	0,5612	0,7055	0,0362
stemstop_svd250_sgd-logreg_br	0,6074	0,7461	0,0367
stemstop_svd250_svc_br	0,6115	0,7391	0,0330
stemstop_svd250_sgd-svc_br	0,5182	0,6918	0,0454
stemstop_svd250_xgb_br	0,6201	0,7448	0,0334
stemstop_svd250_tree_br	0,5147	0,6573	0,0464
stemstop_svd500_logreg_br	0,5508	0,6938	0,0374
stemstop_svd500_sgd-logreg_br	0,6115	0,7450	0,0346
stemstop_svd500_svc_br	0,5388	0,6725	0,0388
stemstop_svd500_sgd-svc_br	0,4972	0,6863	0,0489
stemstop_svd500_xgb_br	0,6046	0,7298	0,0349
stemstop_svd500_tree_br	0,5128	0,6660	0,0444
stemstop_non_logreg_br	0,5381	0,6820	0,0389
stemstop_non_sgd-logreg_br	0,5887	0,7202	0,0364
stemstop_non_svc_br	0,3740	0,5151	0,0507
stemstop_non_sgd-svc_br	0,6128	0,7378	0,0362
stemstop_non_xgb_br	0,5818	0,7348	0,0374
stemstop_non_nb_br	0,5948	0,7294	0,0370
stemstop_non_tree_br	0,4939	0,6898	0,0484
raw_svd100_logreg_cc	0,5301	0,6706	0,0398
raw_svd100_sgd-logreg_cc	0,6049	0,7189	0,0389
raw_svd100_svc_cc	0,6541	0,7402	0,0354
raw_svd100_sgd-svc_cc	0,6088	0,7005	0,0436
raw_svd100_xgb_cc	0,5774	0,7063	0,0373
raw_svd100_tree_cc	0,4395	0,6011	0,0519
raw_svd250_logreg_cc	0,5315	0,6767	0,0391
raw_svd250_sgd-logreg_cc	0,5988	0,7157	0,0396

Dilanjutkan pada halaman berikutnya

Tabel 2 – lanjutan dari halaman sebelumnya

Penamaan Model	EMR	F1	Hamm
raw_svd250_svc_cc	0,6182	0,6773	0,0451
raw_svd250_sgd-svc_cc	0,5646	0,6659	0,0462
raw_svd250_xgb_cc	0,5468	0,6831	0,0390
raw_svd250_tree_cc	0,4347	0,5960	0,0513
raw_svd500_logreg_cc	0,5286	0,6692	0,0396
raw_svd500_sgd-logreg_cc	0,6359	0,7483	0,0341
raw_svd500_svc_cc	0,5103	0,5657	0,0596
raw_svd500_sgd-svc_cc	0,5212	0,6454	0,0486
raw_svd500_xgb_cc	0,5264	0,6662	0,0403
raw_svd500_tree_cc	0,4229	0,5883	0,0507
raw_non_logreg_cc	0,4660	0,6190	0,0440
raw_non_sgd-logreg_cc	0,5399	0,6711	0,0430
raw_non_svc_cc	0,1615	0,2595	0,0658
raw_non_sgd-svc_cc	0,5515	0,6825	0,0434
raw_non_xgb_cc	0,5698	0,7241	0,0377
raw_non_nb_cc	0,4812	0,6295	0,0445
raw_non_tree_cc	0,5008	0,6876	0,0489
stem_svd100_logreg_cc	0,5229	0,6627	0,0409
stem_svd100_sgd-logreg_cc	0,6063	0,7087	0,0410
stem_svd100_svc_cc	0,6573	0,7444	0,0351
stem_svd100_sgd-svc_cc	0,6136	0,7000	0,0439
stem_svd100_xgb_cc	0,5734	0,7072	0,0374
stem_svd100_tree_cc	0,4368	0,5928	0,0537
stem_svd250_logreg_cc	0,5357	0,6766	0,0392
stem_svd250_sgd-logreg_cc	0,6180	0,7330	0,0375
stem_svd250_svc_cc	0,6207	0,6886	0,0435
stem_svd250_sgd-svc_cc	0,5796	0,6947	0,0422
stem_svd250_xgb_cc	0,5349	0,6773	0,0397
stem_svd250_tree_cc	0,4502	0,5988	0,0507
stem_svd500_logreg_cc	0,5181	0,6637	0,0404

Dilanjutkan pada halaman berikutnya

Tabel 2 – lanjutan dari halaman sebelumnya

Penamaan Model	EMR	F1	Hamm
stem_svd500_sgd-logreg_cc	0,6364	0,7501	0,0340
stem_svd500_svc_cc	0,5355	0,5910	0,0557
stem_svd500_sgd-svc_cc	0,5347	0,6747	0,0470
stem_svd500_xgb_cc	0,5346	0,6687	0,0404
stem_svd500_tree_cc	0,4320	0,5923	0,0507
stem_non_logreg_cc	0,4807	0,6282	0,0434
stem_non_sgd-logreg_cc	0,5572	0,6899	0,0407
stem_non_svc_cc	0,1759	0,2813	0,0647
stem_non_sgd-svc_cc	0,5603	0,6891	0,0426
stem_non_xgb_cc	0,5822	0,7230	0,0379
stem_non_nb_cc	0,4732	0,6208	0,0450
stem_non_tree_cc	0,5093	0,6872	0,0482
stemstop_svd100_logreg_cc	0,6082	0,7291	0,0348
stemstop_svd100_sgd-logreg_cc	0,6478	0,7527	0,0358
stemstop_svd100_svc_cc	0,6817	0,7744	0,0313
stemstop_svd100_sgd-svc_cc	0,6156	0,7285	0,0398
stemstop_svd100_xgb_cc	0,6382	0,7522	0,0333
stemstop_svd100_tree_cc	0,5054	0,6565	0,0483
stemstop_svd250_logreg_cc	0,6106	0,7346	0,0340
stemstop_svd250_sgd-logreg_cc	0,6435	0,7557	0,0356
stemstop_svd250_svc_cc	0,6576	0,7428	0,0350
stemstop_svd250_sgd-svc_cc	0,5590	0,7031	0,0457
stemstop_svd250_xgb_cc	0,6202	0,7433	0,0337
stemstop_svd250_tree_cc	0,5108	0,6617	0,0465
stemstop_svd500_logreg_cc	0,5903	0,7233	0,0351
stemstop_svd500_sgd-logreg_cc	0,6423	0,7622	0,0342
stemstop_svd500_svc_cc	0,5941	0,7066	0,0369
stemstop_svd500_sgd-svc_cc	0,5294	0,6839	0,0476
stemstop_svd500_xgb_cc	0,6041	0,7310	0,0347
stemstop_svd500_tree_cc	0,5067	0,6531	0,0462

Dilanjutkan pada halaman berikutnya

Tabel 2 – lanjutan dari halaman sebelumnya

Penamaan Model	EMR	F1	Hamm
stemstop_non_logreg_cc	0,5723	0,7057	0,0369
stemstop_non_sgd-logreg_cc	0,6379	0,7448	0,0347
stemstop_non_svc_cc	0,4099	0,5461	0,0486
stemstop_non_sgd-svc_cc	0,6387	0,7359	0,0377
stemstop_non_xgb_cc	0,6040	0,7413	0,0368
stemstop_non_nb_cc	0,6001	0,7257	0,0373
stemstop_non_tree_cc	0,5257	0,6931	0,0471

**LAMPIRAN 3: KATA KUNCI YANG MEREPRESENTASIKAN
LABEL**

Anak	: anak, alergi, demam, bayi, asi, susu, imunisasi, anak usia, anak anak, anak umur, bayi minum, minum asi, minum susu, susu formula, formula anak, ganti susu, bayi usia, alergi susu, anak alergi, susu sapi, asi susu, susu, tahun anak, anak makan, susu anak, sapi anak, imunisasi anak, anak susu, usia anak, anak laki-laki, anak minum, formula asi, badan anak, susu soya, anak umur tahun, minum asi minum, asi minum susu, minum susu formula, susu formula anak, susu formula bayi, pilih susu formula, susu formula coba, bayi alergi susu, anak alergi susu, alergi susu sapi, asi susu formula, susu sapi anak, anak susu formula, bayi minum susu, anak minum susu, susu formula asi, berat badan anak, konsumsi susu formula, anak konsumsi susu, minum susu soya
Bedah	: perut, usus, benjol, anus, wasir, bab, bengkak, operasi, luka, empedu, usus buntu, gejala usus, obat wasir, buntu operasi, bab darah, bekas luka, bekas operasi, perut kanan sakit, usus buntu siang, wasir obat wasir, pasca operasi usus
Gigi	: mulut, sariawan, saraf, lidah, gigi, geraham, gusi, gigi gigi, gigi lubang, ngilu ganggu, gigi tambal, saraf gigi, alami sariawan, tambal gigi, gigi bersih, karang gigi, gigi ngilu, alami ngilu, gigi darah, gusi bengkak, obat gigi, pasta gigi, cabut gigi, sikat gigi, derita sariawan, sariawan minggu, sariawan sembah, gigi kuning, mutih gigi, gigi warna, gigi putih, putih gigi, kuning gigi, gusi darah, darah sariawan, gigi paiah, warna gigi, bau mulut, gigi warna kuning, warna kuning gigi

Dilanjukkan pada halaman berikutnya

Gambar 1 – lanjutan dari halaman sebelumnya

Gizi	: kurus, makan, diet, gemuk, tinggi, makan derita, minum susu, program hamil, badan tinggi, porsi makan, badan makan, badan kurus, milik berat, badan ideal, kurus berat badan, pantang makan derita, makan derita asam, milik berat badan, susah berat badan, keluh berat badan, berat badan milik, berat badan selamat, badan tinggi badan, berat badan capai, badan capai kg
Jantung	: dada, nafas, sesak, panik, jantung, ekg, detak, berdebarberdebar, obatobat, bisoprolol, panic, hipertiroid, aritmia, dada tengah, dada belah, akibat jantung, dosis mghari, tinggi keluh, rekam jantung, nyaman dada, jantung akibat, turun tekan, tinggi keluh sakit
Jiwa	: panik, depresi, sedih, cemas, takut, anxiety, disorder, jiwa, psikiater, panic, alami depresi, anxiety disorder, sulit tidur, depresi berat, obat depresi, ganggu jiwa, social anxiety, panic disorder, panic attack, efek samping pusing, suka banting barang, orang mudah kaget, disorder fobia sosial
Kandungan:	perut, hamil, haid, flek, kandung, usg, kista, rahim, operasi, caesar, menstruasi, mens, lahir, kb, janin, tanda hamil, indung telur, operasi caesar, haid atur, usia kandung, program hamil, meni tahun, hamil minggu, siklus haid, usia hamil, flek hamil, badan janin, berat janin, pasca operasi caesar, makan mie instan, usia hamil minggu, berat badan janin
Kulit dan Kelamin	: keringat, benjol, jerawat, gatal, bintikbintik, kulit, bintik, kelamin, kutul, tahi lalat, tangan kaki, kulit kelupas, wajah jerawat, telapak tangan, hilang tahi, muncul bintikbintik, pakai produk, kutul kelamin, bekas jerawat, telapak tangan kaki, hilang tahi lalat, tahi lalat hilang, telapak tangan telapak

Dilanjutkan pada halaman berikutnya

Gambar 1 – lanjutan dari halaman sebelumnya

Mata	: mata, silinder, kacamata, belek, softlens, mata minus, kiri minus, mata anak, minus silinder, silinder minus, pakai kacamata, minus mata, air belek, mata normal, tetes mata, merah mata, obat mata, mata silinder, kanan silinder, sembuhan mata, mata merah, silinder sembuhan mata, silinder mata, mata ubah, ganggu mata, merah belek, buta warna, air mata, mata belek, belek selamat, belek mata, minus silindris, silinder pakai, tangga mata, periksa mata, pakai softlens, softlens minus, mata minus silinder, silinder minus silinder, minus silinder minus, obat mata silinder, mata silinder adik, mata silinder sembuhan mata, silinder mata silinder, mata merah belek, mata ubah warna, mata anak belek, mata minus silindris, minus silinder pakai, bangun tidur mata, merah bangun tidur, pakai softlens minus, mata merah air, merah air belek, mata silinder mata, belek bangun tidur
Paru	: rokok, asap, batuk, dahak, bronkitis, ingus, asma, tbc, fdc, tb, fibrosis, asma kambuh, hidung sumbat, obat tbc
Penyakit	: darah, perut, usus, bab, punggung, dada, lambung, mual, hati, sesak, jantung, ginjal, maag, diabetes, stroke, usus buntu, gejala sakit, sakit ulu, ulu hati, hati sakit, nyeri ulu, sakit ulu hati, ulu hati sakit, nyeri ulu hati
Dalam	
Saraf	: otak, pinggang, kepala, leher, syaraf, stroke, bentur kepala, cedera kepala, akibat bentur, sakit tulang punggung tenggorok, demam, pusing, mimisan, batuk, hidung, flu, ingus, radang, telinga, dengung, dengar, tht, pilek, sumbat, gendang, telinga belah, dengung telinga, hidung belah, telinga kiri, dengar telinga, telinga kanan, mimisan mimisan, hidung sumbat, darah hidung, gendang telinga, telinga air, kotor telinga, telinga dengung, telinga sakit, tetes telinga, sakit telinga, telinga belah kanan, dengung telinga belah, hidung belah kanan, gendang telinga pecah, telinga belah kiri, obat tetes telinga, belah kiri dengung, telinga dengung telinga, dengung telinga dengung
THT	

Dilanjutkan pada halaman berikutnya

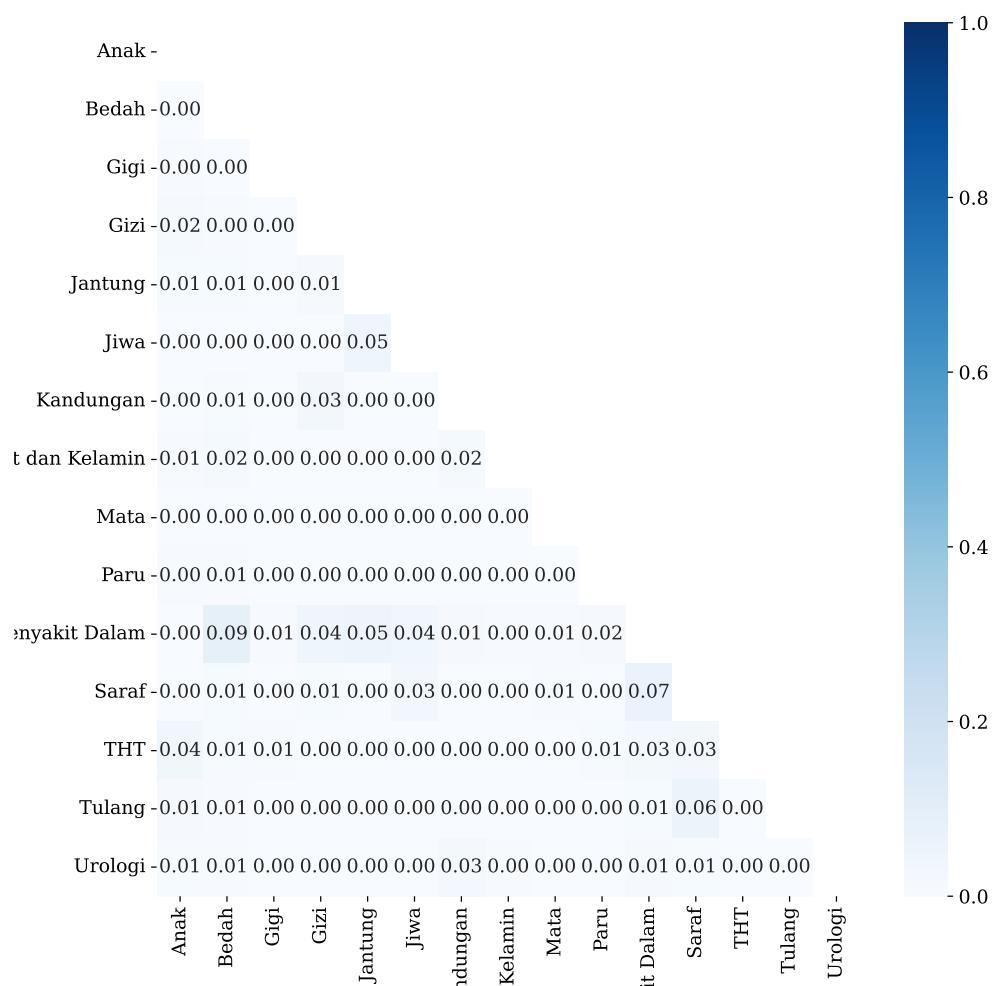
Gambar 1 – lanjutan dari halaman sebelumnya

Tulang : tulang sendi, ligamen, patah tulang, tulang kering, tulang patah, lutut kanan, kering kanan, bentur separator, separator busway, motor jatuh, kanan bentur, tulang dada, tulang tulang, cedera lutut, lutut kiri, tulang sendi, tulang kering kanan, bentur separator busway, kering kanan bentur

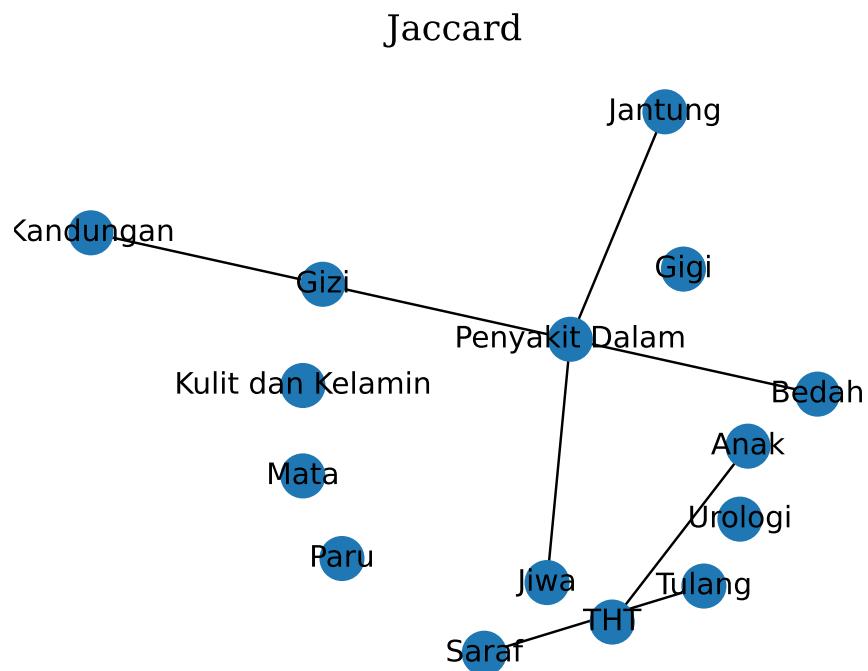
Urologi : tumor, ginjal, bak, kemih, urin, kencing, bedah, batu, urologi, testis, prostat, kateter, besar, selang ginjal, usg ginjal, bedah urologi, pasang selang, retensi urin, ginjal operasi, operasi batu ginjal, batu ginjal operasi

Gambar 1: Data teks yang diduga terjadi kesalahan anotasi pada dataset anotasi oleh dokter

LAMPIRAN 4: HASIL KORELASI ANTAR LABEL

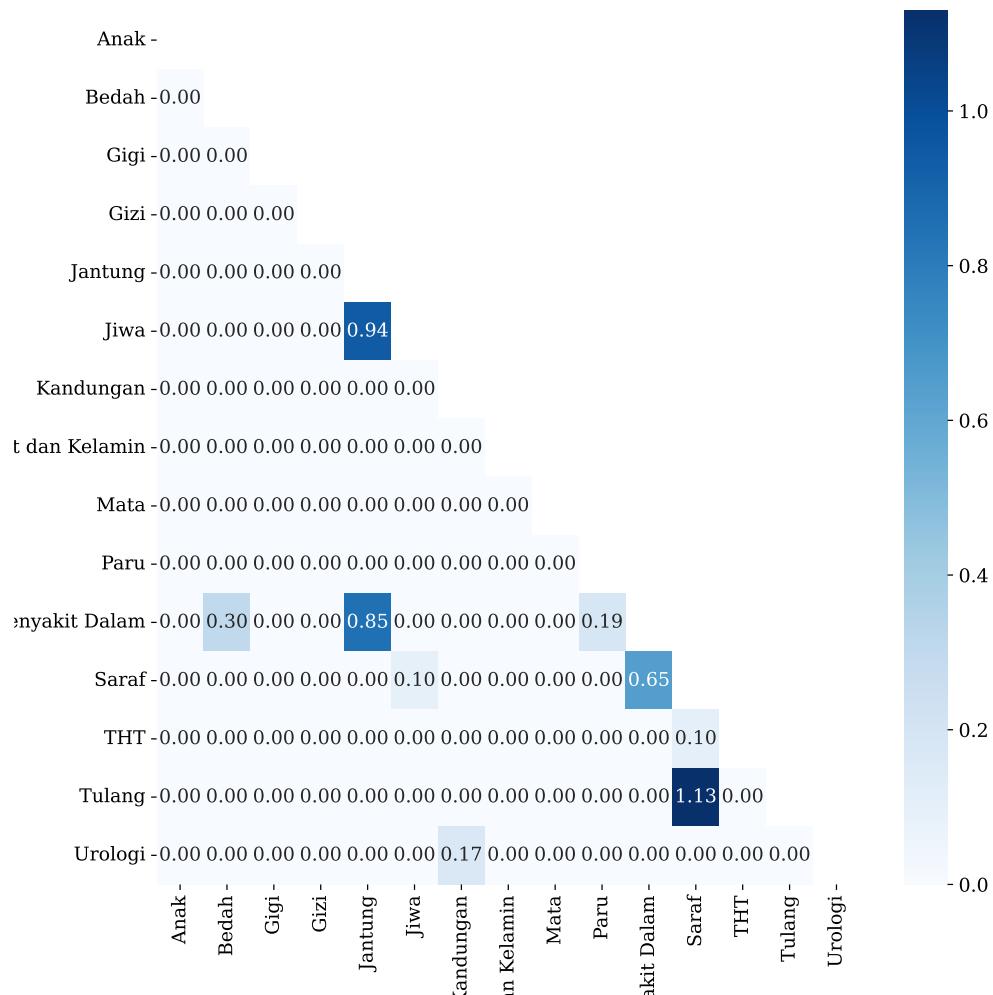


(a) Matriks segitiga bawah yang memperlihatkan nilai korelasi dari pasangan label

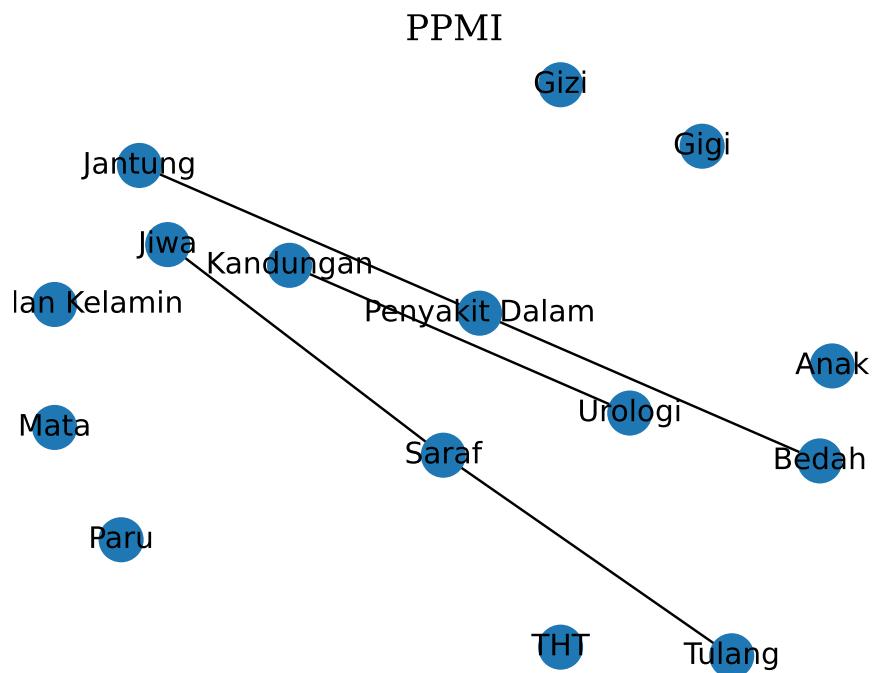


(b) Gugusan yang terbentuk dengan metode jarak Jaccard

Gambar 2: Korelasi yang didapatkan dengan metode jarak Jaccard. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0,3

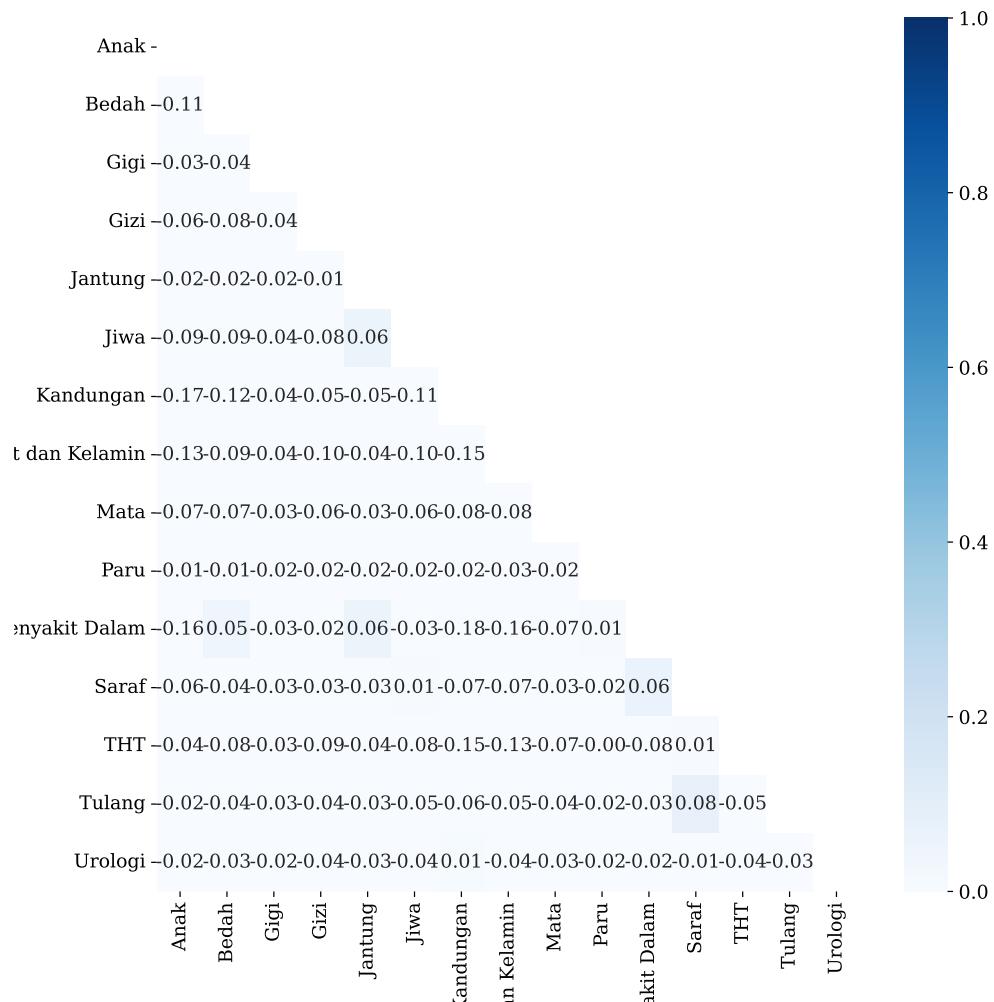


(a) Matriks segitiga bawah yang memperlihatkan nilai korelasi dari pasangan label

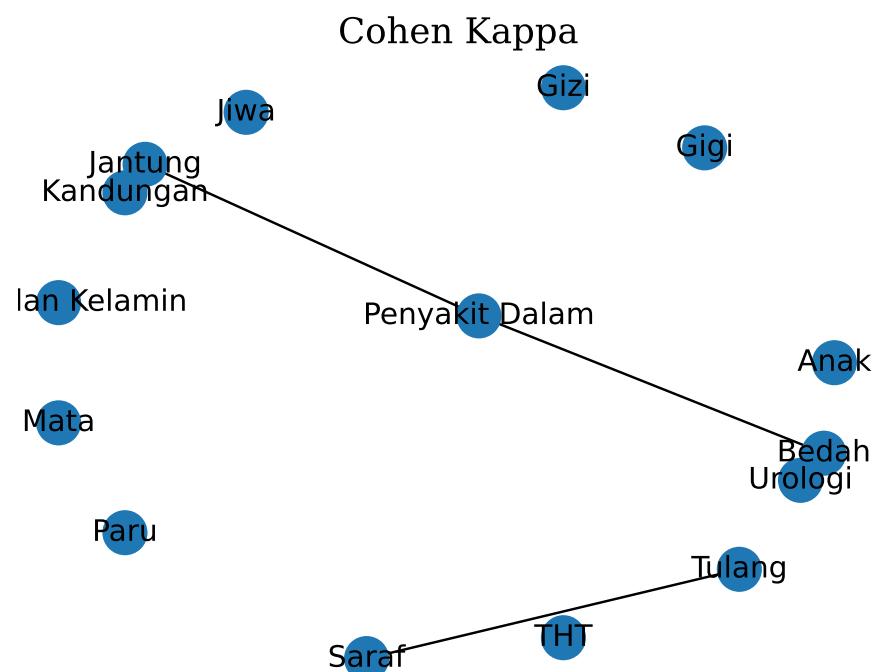


(b) Gugusan yang terbentuk dengan metode PPMI

Gambar 3: Korelasi yang didapatkan dengan metode PPMI. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0



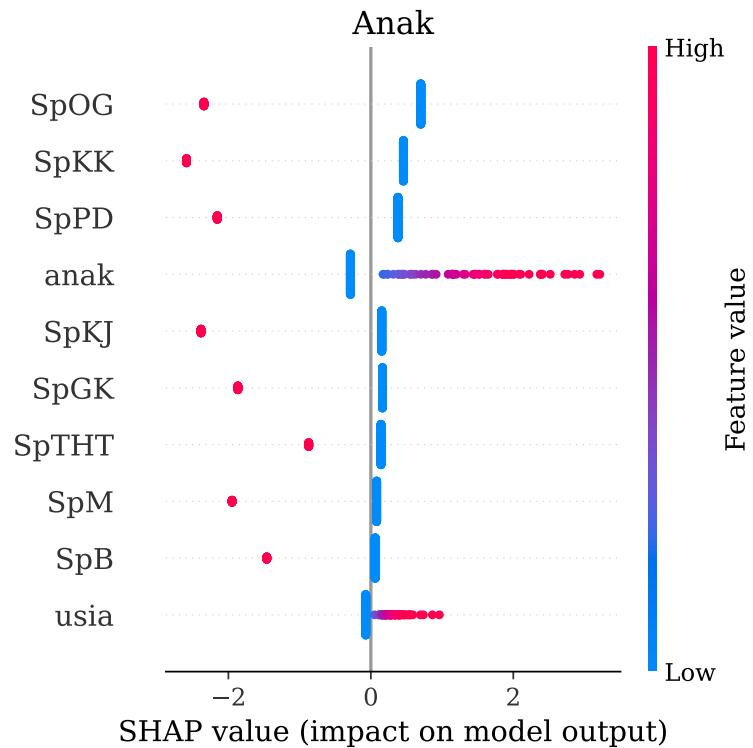
(a) Matriks segitiga bawah yang memperlihatkan nilai korelasi dari pasangan label



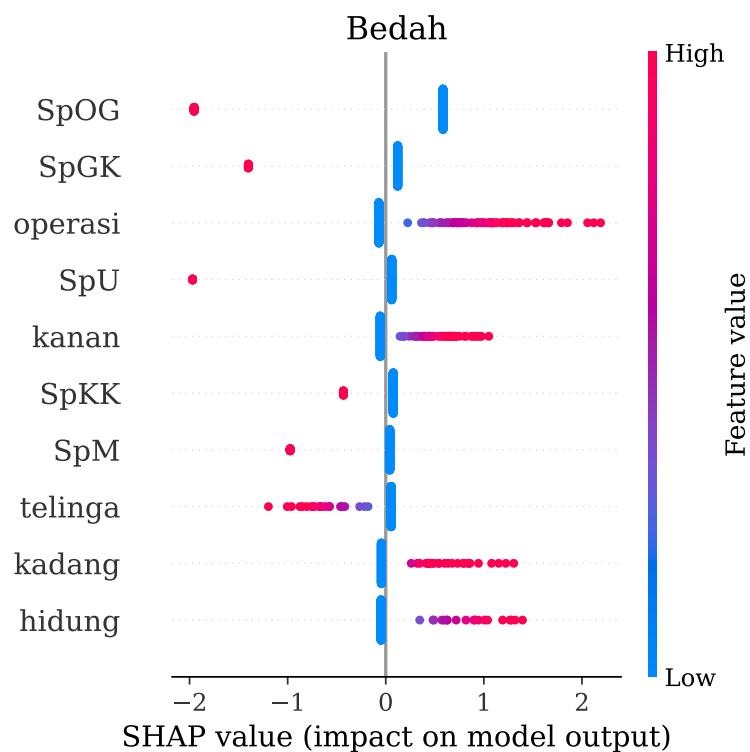
(b) Gugusan yang terbentuk dengan metode Cohen's kappa

Gambar 4: Korelasi yang didapatkan dengan metode PPMI. Nilai minimum yang dibutuhkan agar terbentuk korelasi sebesar 0,04

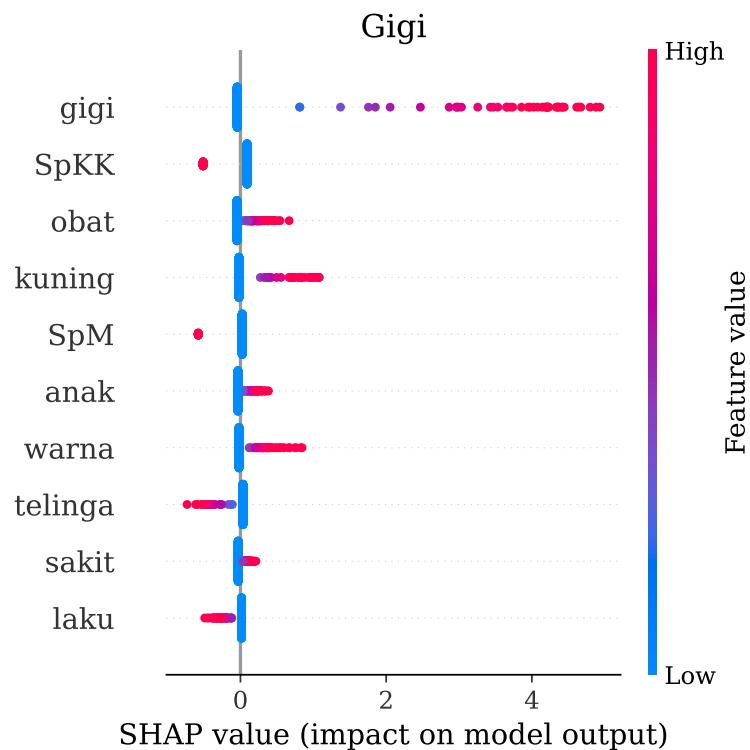
LAMPIRAN 5: HASIL SHAP DARI SETIAP KELAS



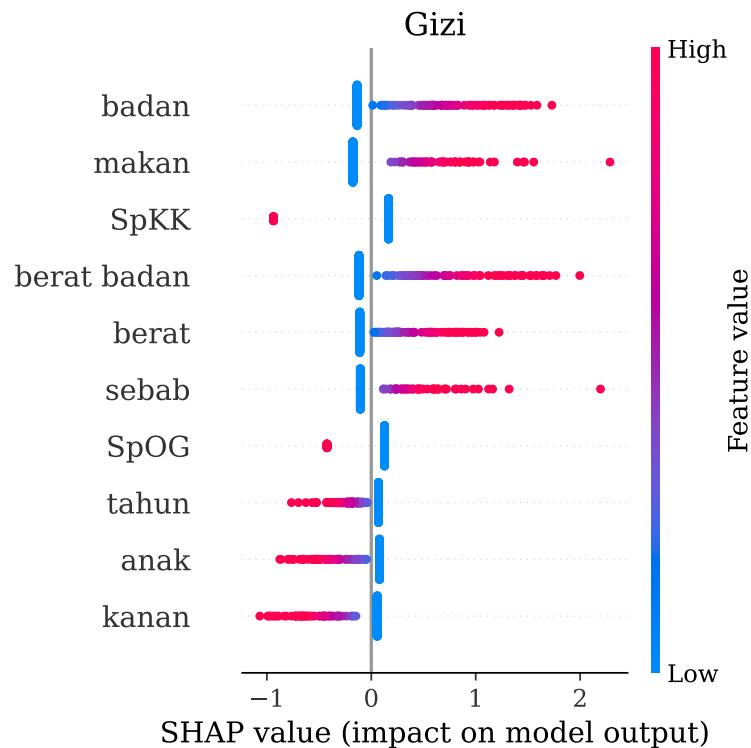
(a) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas anak



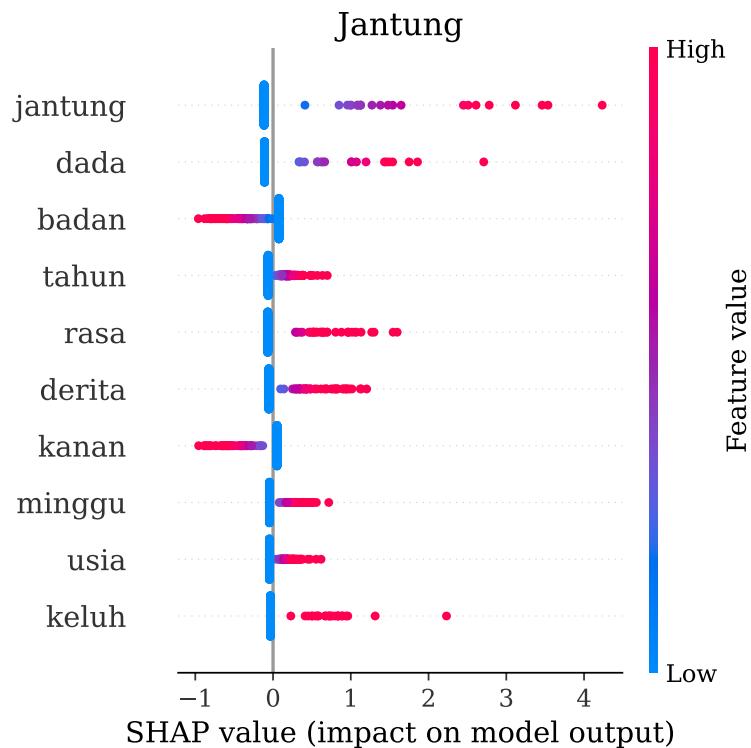
(b) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas bedah



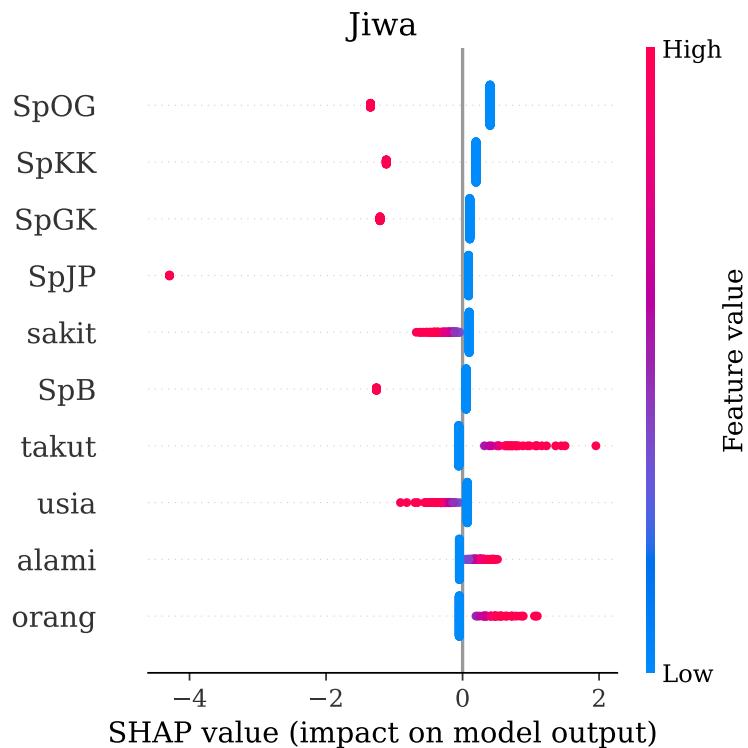
(c) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas gigi



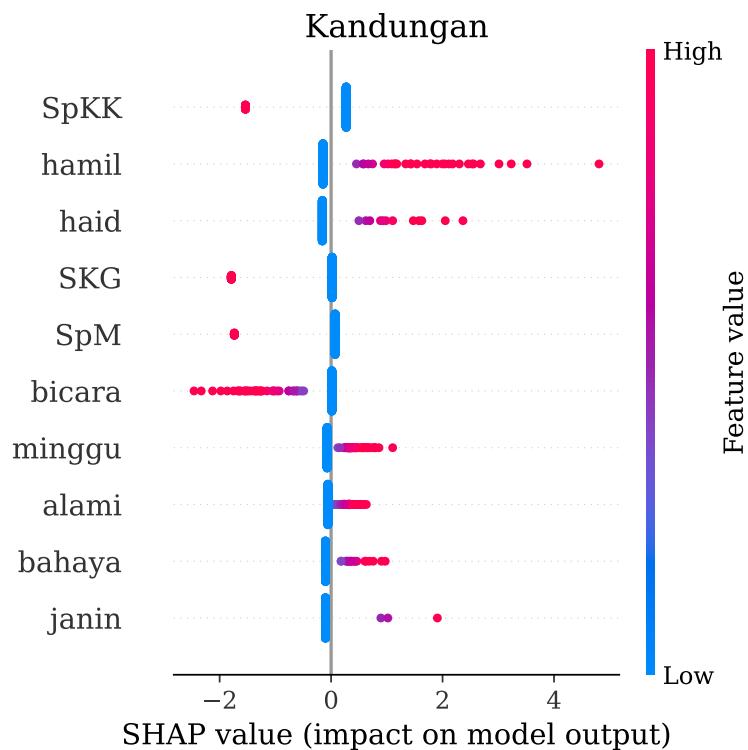
(d) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas gizi



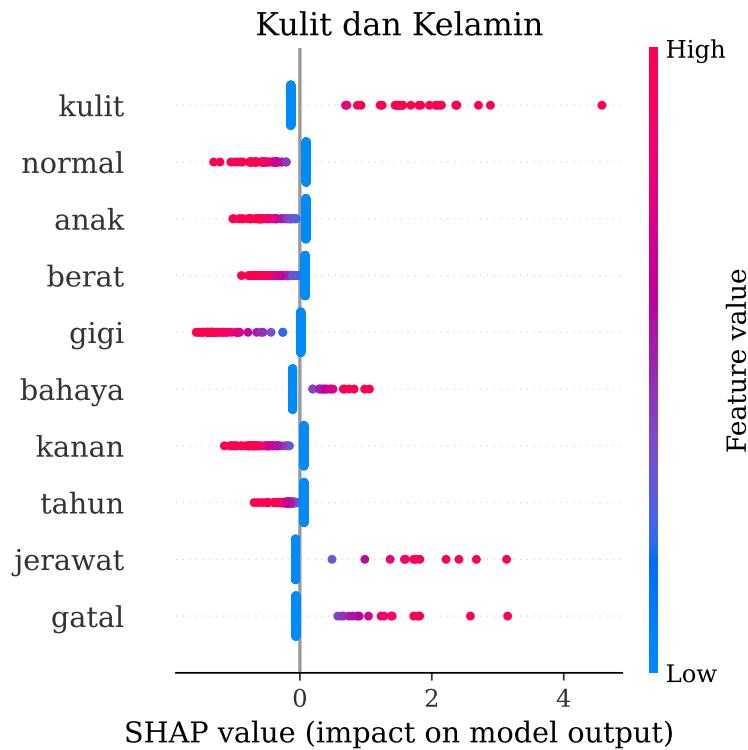
(e) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas jantung



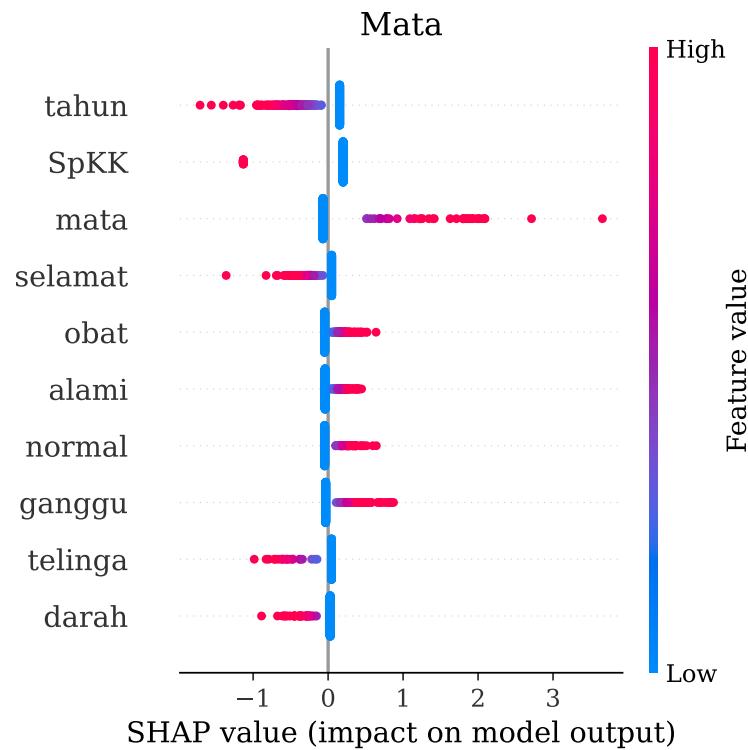
(f) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas jiwa



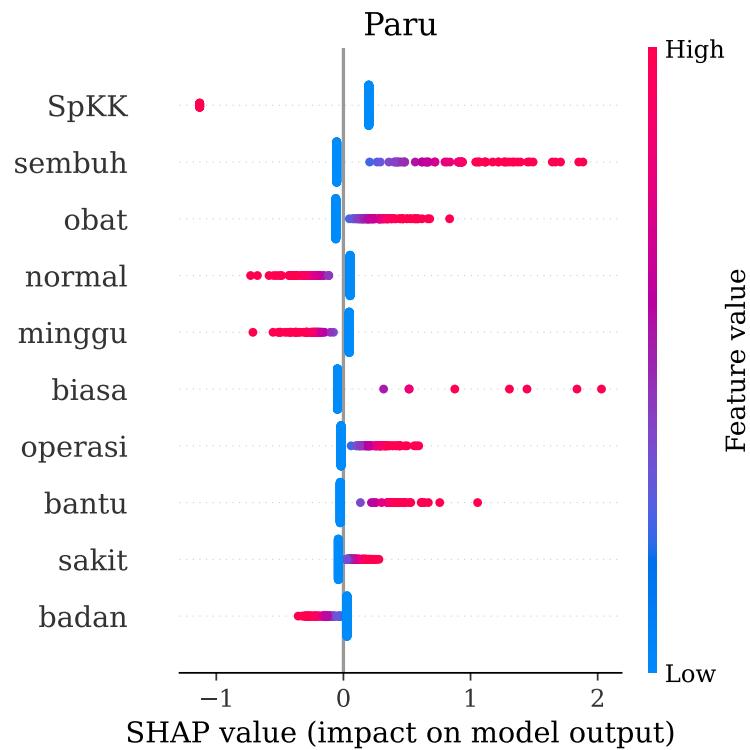
(g) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas kandungan



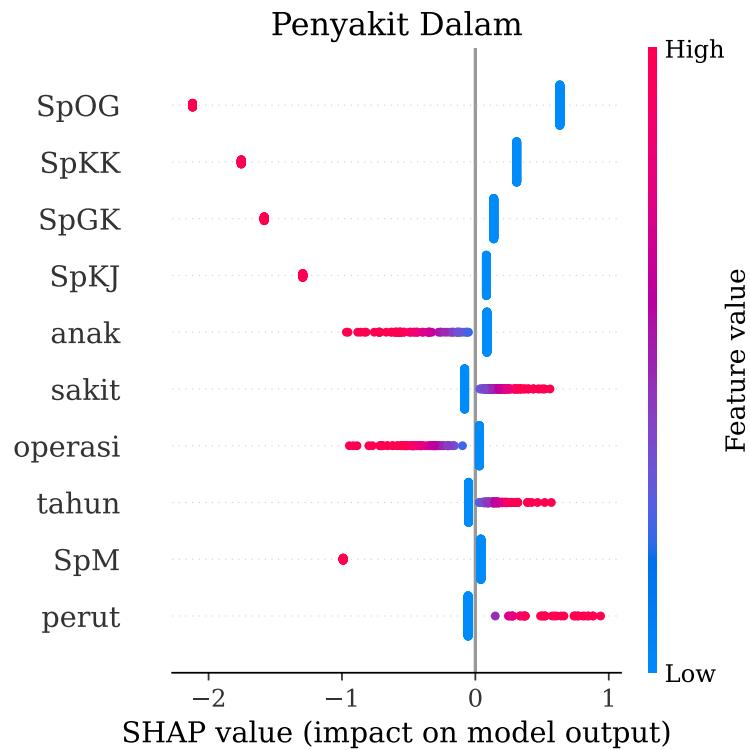
(h) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas kulit dan kelamin



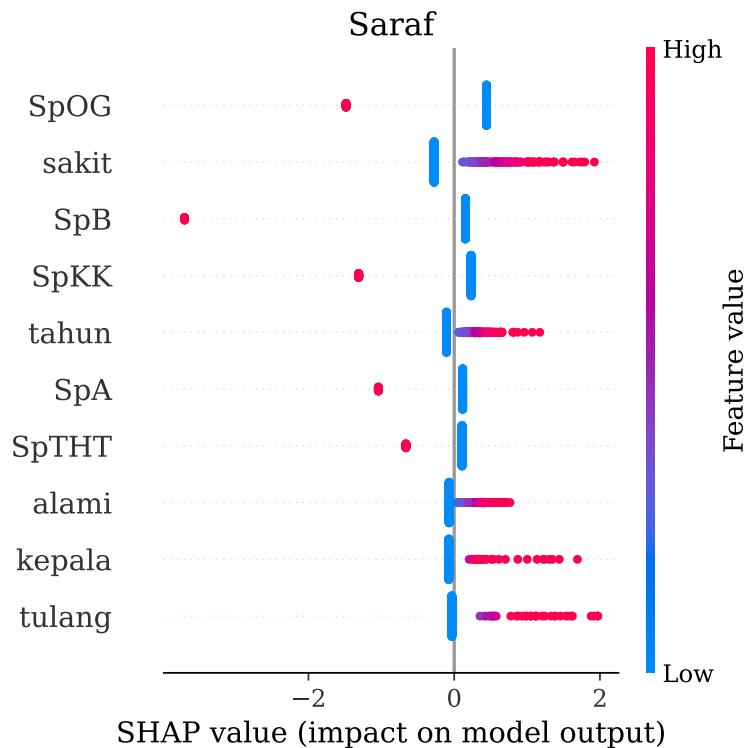
(i) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas mata



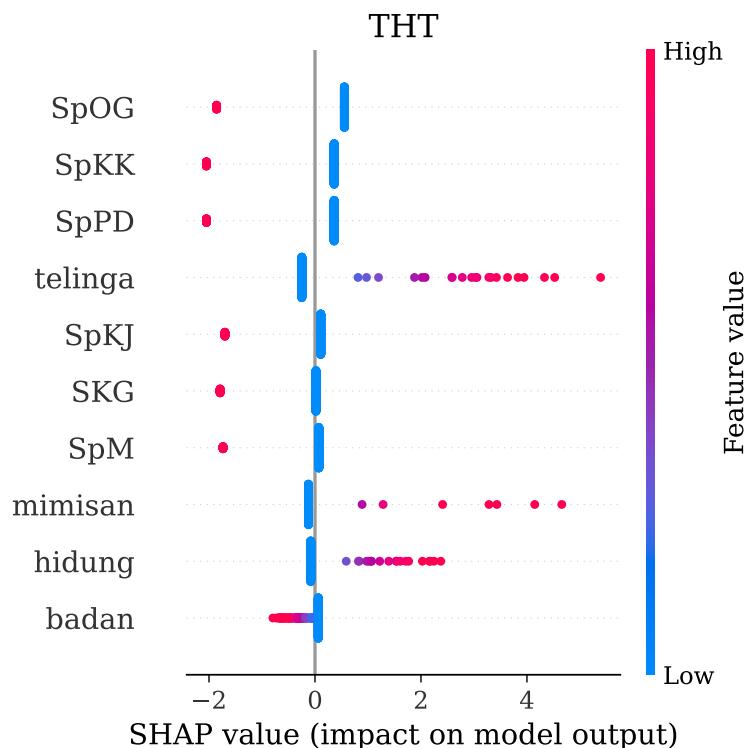
(j) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas paru



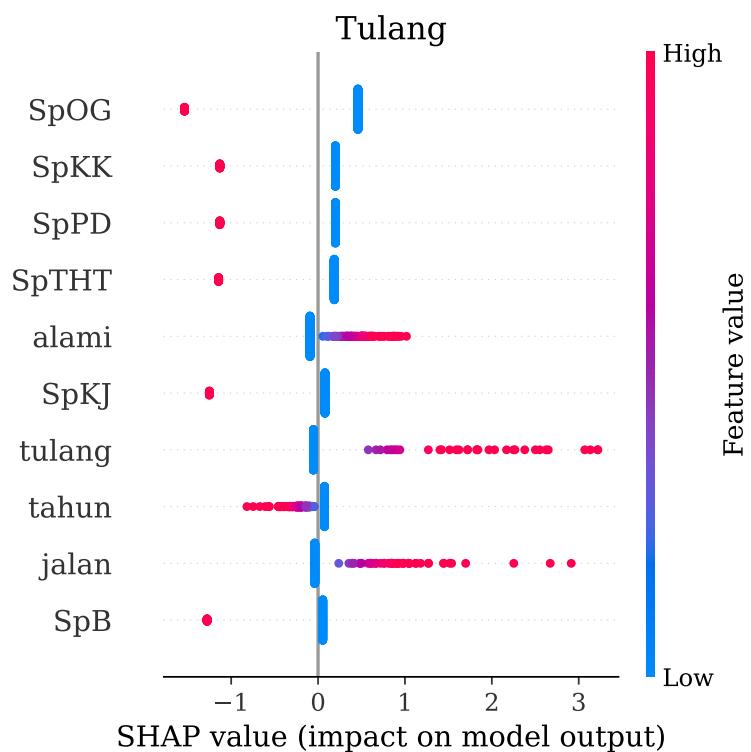
(k) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas penyakit dalam



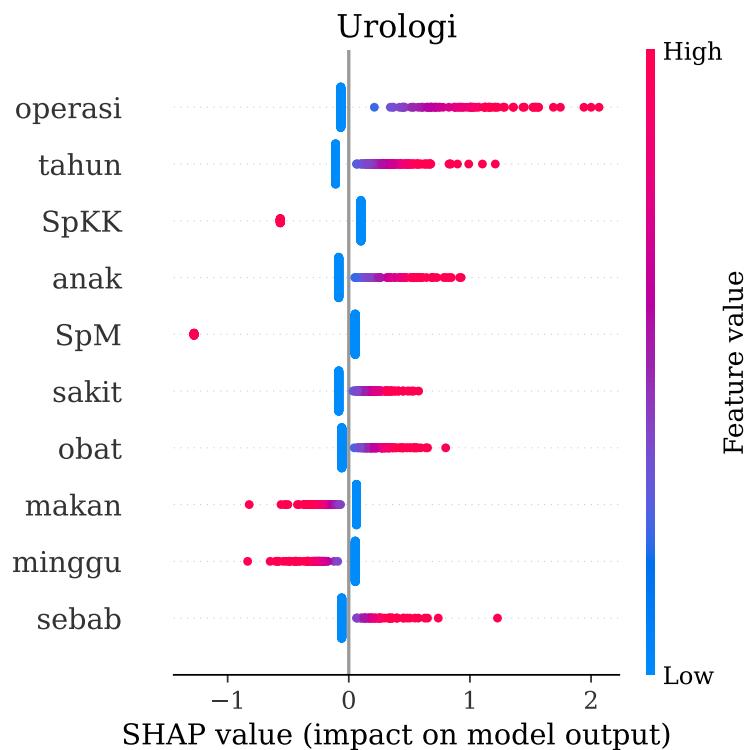
(l) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas saraf



(m) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas THT



(n) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas tulang



(o) Kontribusi tiap fitur yang memengaruhi klasifikasi kelas urologi

Gambar 5: Hasil SHAP dari beberapa sampel kelas yang memiliki pola yang dapat digeneralisasi. Untuk mempersingkat nama fitur, hasil prediksi label sebelumnya disingkat menjadi gelar dari dokter. Yaitu kelas anak menjadi SpA, bedah menjadi SpB, gigi menjadi SKG, gizi menjadi SpGK, jantung menjadi SpJP, jiwa menjadi SpKJ, kulit dan kelamin menjadi SpKK, mata menjadi SpM, saraf menjadi SpN, kandungan menjadi SpOG, tulang menjadi SpOT, paru menjadi SpP, penyakit dalam menjadi SpPD, THT menjadi SpTHT, dan urologi menjadi SpU.

LAMPIRAN 6: KESALAHAN ANOTASI PADA DATA LATIH

Id	:	AD-15810	
Isi	:	assalamualaikum.wr.wb selamat siang dok umur 22 tahun , saya mau bertanya dok , sudah lebih dari 1 minggu ini saya mengalami : -deman (panas , pusing) disertai batuk -saat batuk perut terasa sakit -kadang-kadang keluar darah bercampur dengan lendir kondisi kesehatan saya , 6 bulan yang lalu saya sakit demam berdarah . dan 6 bulan setelahnya saya alhamdulillah merasa sehat . tetapi selama 1 minggu ini saya mengalami demam seperti diatas . jadi pertanyaan saya , apa ada hubungannya demam dan batuk yang saya alami sekarang ada kaitannya dengan demam saya yang saya rasakan 6 bulan sebelumnya dok ? atau ini hal yang baru , karena saya seorang perokok ? langkah awal yang seharusnya saya lakukan untuk sembuh kembali apa dok ? terima kasih sebelumnya	
Salah	:	Gigi	
Benar	:	Paru	
Id	:	AD-10008	
Isi	:	pagii dok setelah kencing nanah saya sudah agak baikan tetapi kenapa perut saya jadi kembung dan seperti tidak enak.terus bab saya juga tidak lancar setelah saya ke dokter perut agak baikan dan tetapi tetap susah bab .	
Salah	:	Kulit dan Kelamin	
Benar	:	Penyakit Dalam	
Id	:	AD-20382	

Dilanjutkan pada halaman berikutnya

Gambar 6– lanjutan dari halaman sebelumnya

Isi : pagi dok dodi umur 28 tahun sekitar sebulan yang lalu saya hub sek dari 7 hari setelah itu kencing saya sakit dan keluar nanah.sama seperti teman sama karena satu cewek setelah minum obat tiamicin rasa sakit dan nanah hilang tetapi setelah itu saya rasakan perut terasa kembung.dan tidak enak saya ke dokter setelah diberi obat perut sama kencing saya agak lumayan hanya tidak bisa bab sekali bab kaya lendir dok.teman saya yang kena sudah sembuh saya selalu takut dok . di cemas jadi saya merasa lemas

Salah : Kulit dan Kelamin

Benar : Penyakit Dalam

Id : KD-18813

Isi : dok , saya mau tanya ini . apakah anak sewaktu bayi telah melakukan imunisasi lengkap , dan sewaktu sekolah (sd) ada imunisasi lagi seperti dt atau tadt , harus ikut imunisasi lagi atau tidak ya dok . terima kasih sebelumnya

Salah : Kulit dan Kelamin, Umum

Benar : Anak

Id : KD-37887

Dilanjutkan pada halaman berikutnya

Gambar 6– lanjutan dari halaman sebelumnya

Isi : saya seorang wanita 34 tahun , saya sudah menikah selama 13 tahun tetapi belum ja dikaruniai anak.saya pernah mengalami infaksi tetapi saat itu saya tidak tahu saya terkena infeksi apa namanay , waktu itu pacar saya memaksa saya untuk mlakukan hubungan badan , tiga hari kemudian perut saya sakit dan setiap saya kencing rasanya sakit , saya tidak beranikedokter karena waktu itu saya belum menikah saya malu.kata pacar saya-saya tertular penyakit kelamin.dan akhirnya saya menikah yang bukan pacar saya tadi , sampai meikah rasa nyeri diperut bawah dan nyeri yang sangat pada waktu haid masih ada.kemudian saya periksa kedoketer kandungan , saya diberi obat anti biotik , sakit waktu kecing hilang tetapi perut bawah kanan saya sampai sekarang masih terasa bila diraba dan ditekan.saya sudah usg kedokter kandungan katanya keadaan kandungan saya baik-baik saja , cuma saat ini saya juga belum hamil dan perut kanan bawah saya masih terasa nyeri , yang saya tanyakan 1 . ke mana saya harus periksa untuk mengetahui sakit saya ini karena semua dokter yang saya kunjungi bilang saya baik-baik saja ? 2 .apakah saya masih bisa mempunyai kesempatan untuk hamil karena saya dan suami sangat mendambakan anak ? 3 .saya pernah terlambat haid selama 3 minggu tetapi setelah saya tes , dengan tes pack hasilnya negative , kenapa kok begitu demikian pertanyaan saya , mohon penjelasan dokter.terimaksah atas perhatiannya . 3 .

Salah : Kulit dan Kelamin, Kandungan

Benar : Kandungan

Gambar 6: Data teks yang diduga terjadi kesalahan anotasi pada dataset anotasi oleh dokter