LAPORAN HASIL
PROYEK BERBASIS TEXT MINING UNTUK KLASIFIKASI TOPIK ARTIKEL ILMIAH BERBASIS ABSTRAK DAN JUDUL

Disusun oleh:

Kelompok 2

1. Shevrilla Vilnafa B. S. (22/492511/PA/21118)
2. Priskilla N. P. Br Silalahi (22/493324/PA/21176)
3. Intan Dwi Febryanti (22/494760/PA/21285)
4. Dhanada Santika Putri (22/497239/PA/21407)
5. Yessica Thipandona (22/497660/PA/21441)
6. Febriana Nur Syifa Rizqi (22/499532/PA/21541)

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

2025

Pendahuluan

Dalam era digital saat ini, volume data teks yang tersedia secara daring meningkat secara eksponensial, baik melalui media sosial, artikel berita, maupun publikasi ilmiah. Hal ini mendorong kebutuhan untuk mengembangkan metode pengolahan data teks secara efisien dan otomatis. Salah satu pendekatan yang digunakan adalah *text mining*, yaitu proses ekstraksi informasi bernilai dari data berbasis teks. Melalui *text mining*, berbagai *insight* penting seperti sentimen pengguna, topik dominan, atau pola diskusi dapat ditemukan dari kumpulan teks yang sangat besar, atau yang dikenal dengan istilah *big text data*. Proyek ini menerapkan *pipeline text mining* secara *end-to-end*, mulai dari proses *ingest data* hingga visualisasi dan *deployment* hasil. Tugas ini mencakup seluruh proses kerja yang umumnya digunakan dalam industri maupun riset, seperti pembersihan data (*preprocessing*), ekstraksi fitur, pemodelan topik atau klasifikasi, evaluasi performa, serta penyajian hasil dalam bentuk dashboard atau API. Dengan mengintegrasikan metode *text mining* dan *platform* penyimpanan *big data*, proyek ini diharapkan dapat memberikan pemahaman praktis tentang bagaimana menangani data teks dalam jumlah besar secara sistematis, sekaligus melatih keterampilan dalam kolaborasi, dokumentasi, serta penyajian data secara interaktif.

Dataset

Dataset yang digunakan dalam penelitian ini bersumber dari *platform* Kaggle, dengan judul "*Topic Modeling for Research Articles*". Dataset ini dirancang untuk mendukung eksperimen dan penerapan teknik pemodelan topik (*topic modeling*) pada teks akademik atau ilmiah. Dataset ini terdiri dari dua *file* utama yaitu ‘train.csv’ dan ‘test.csv’. Kedua *file* tersebut masing-masing berisi satu kolom utama, yaitu *abstract*, yang berisi ringkasan dari artikel-artikel penelitian ilmiah dalam bahasa Inggris. *File* ‘train.csv’ yang terdiri atas 20972 baris digunakan sebagai data latih untuk membangun atau melatih model, sedangkan ‘test.csv’ terdiri atas 8989 baris yang digunakan untuk keperluan evaluasi atau pengujian performa model. Dalam penelitian ini, kedua *file* CSV tersebut dimigrasikan ke dalam sistem manajemen basis data non-relasional MongoDB Atlas agar dapat diproses dan dikelola secara efisien. Dataset disimpan dalam *database* bernama ‘kaggle_datasets’, dengan dua koleksi yaitu ‘train_data’ dan ‘test_data’.

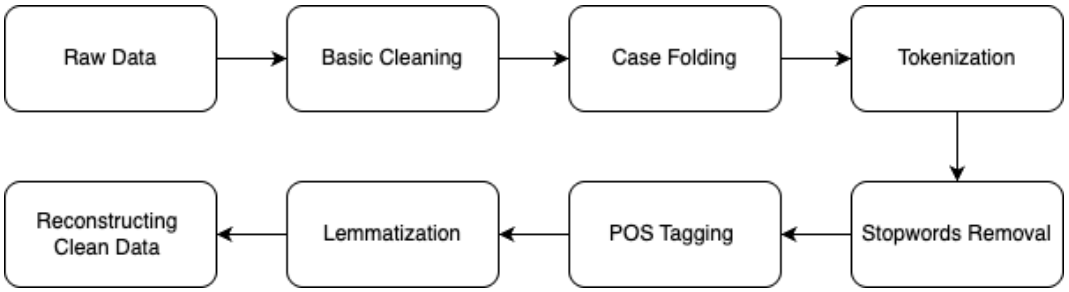
Metode

Penyimpanan Data ke MongoDB

Tahap awal dilakukan dengan membaca *file* ‘train.csv’ dan ‘test.csv’ menggunakan pustaka Python pandas, kemudian mengubah data menjadi format *dictionary* agar kompatibel dengan MongoDB. Setiap baris pada *file* CSV dikonversi menjadi satu dokumen dalam koleksi MongoDB. Proses ini mencakup membuat koneksi ke MongoDB Atlas menggunakan URI, kemudian memasukkan data ke dalam *database* ‘kaggle_datasets’, serta melakukan validasi keberhasilan koneksi dan input data dengan pengecekan ping dan pengambilan sampel. Tujuan dari pemindahan ini adalah agar data dapat diproses langsung di MongoDB serta disimpan secara efisien untuk keperluan analisis lanjutan.

Preprocessing

Setelah data tersedia di MongoDB, dilakukan proses *preprocessing* teks. *Preprocessing* adalah tahap penting dalam *text mining* yang bertujuan untuk mengubah teks mentah menjadi format yang lebih bersih dan terstruktur agar dapat digunakan oleh algoritma NLP. Dalam proyek ini, *preprocessing* dilakukan dalam beberapa langkah sistematis sebagai berikut:



Gambar 1. Tahapan Preprocessing Data

1. **Basic cleaning**

Tahapan ini bertujuan untuk mengurangi *noise* yang tidak penting dalam analisis semantik dengan menghapus elemen-elemen teks yang tidak relevan. Elemen teks tersebut diantaranya adalah sebagai berikut:

- Teks yang berasal dari artikel daring atau scraping web seringkali mengandung elemen HTML (<p>, <a>, dll). Tag ini dihapus menggunakan regex.
- Tautan website (misalnya https://...) tidak mengandung informasi semantik yang relevan untuk analisis.
- Mention atau kata-kata seperti @username dihapus karena umumnya hanya menyebut akun tertentu dan bukan bagian dari konten analisis.
- Membersihkan spasi ganda, tab, dan newline (\n) untuk merapikan teks.
- Penghapusan simbol, angka, dan karakter non-huruf lainnya.

2. **Case Folding**

Tahapan *case folding* ini bertujuan untuk mengubah semua huruf menjadi huruf kecil (*lowercasing*), agar menyamakan representasi kata yang seharusnya identik secara semantik.

3. **Tokenization**

Tahap ini merupakan proses memecah kalimat menjadi satuan kata atau token. Digunakan *library* nltk.word_tokenize, yang memperhatikan aturan linguistik dalam membagi kata.

4. **Stopwords removal**

Stopwords adalah kata-kata umum dalam bahasa yang tidak memiliki makna penting dalam konteks analisis, seperti *the*, *is*, *and*, *was*, dan lainnya. *Stopwords* dihapus untuk meningkatkan efisiensi pemrosesan dan fokus pada kata bermakna tinggi.

5. **POS Tagging dan Lemmatization**

Part-of-Speech Tagging merupakan tahap labeling setiap kata berdasarkan jenis katanya, seperti N (*noun*), V (*verb*), J (*adjective*), dan R (*adverb*). *POS Tagging* penting karena akan mempengaruhi hasil lematisasi. Kemudian setelah setiap kata melalui *POS*

Tagging, tahap selanjutnya adalah *Lemmatization* yaitu tahap mengubah kata menjadi bentuk dasarnya (*lemma*) dengan memperhatikan jenis katanya. Ini berbeda dengan *stemming* yang hanya memotong akhir kata tanpa peduli maknanya. *Lemmatization* mempertahankan struktur bahasa alami, sehingga hasil lebih akurat secara semantik dibandingkan *stemming*.

6. **Reconstructing Clean Text**

   Setelah seluruh proses di atas, token hasil akhir digabungkan kembali menjadi string teks yang siap dianalisis. Hasil dari preprocessing disimpan kembali ke MongoDB dalam kolom baru bernama 'processed_abstract', dan juga diekspor ke *file* CSV (train_processed_mongodb.csv dan test_processed_mongodb.csv) untuk dokumentasi dan analisis lanjutan. Masing-masing *file* hasil ekspor menyimpan tiga kolom utama yaitu ID MongoDB (mongodb_id), teks asli (ABSTRACT), dan hasil teks yang sudah diproses (processed_abstract).

Ekstraksi Fitur

Ekstraksi fitur bertujuan mengubah teks menjadi representasi numerik agar dapat diproses oleh algoritma machine learning. Metode representasi ini harus mampu menangkap struktur statistik maupun makna semantik dari teks.

1. **Bag of Words (BoW)**

   BoW merepresentasikan dokumen sebagai vektor frekuensi kata tanpa memperhatikan urutan atau konteks. Meskipun sederhana dan mudah diterapkan, metode ini menghasilkan dimensi tinggi dan tidak menangkap makna semantik.
2. **TF-IDF**

   TF-IDF memberikan bobot pada kata berdasarkan frekuensinya dalam dokumen dan seberapa umum kata tersebut di korpus. Representasi ini menyoroti kata-kata penting dan lebih informatif dibanding BoW, meski tetap mengabaikan konteks.
3. **Word2Vec**

   Word2Vec menghasilkan vektor kata berdimensi tetap berdasarkan konteks sekitar kata. Dengan pendekatan CBOW dan dimensi 300, vektor dari semua kata dalam dokumen dirata-ratakan untuk merepresentasikan keseluruhan dokumen.
4. **Sentence Embedding (BERT)**

   BERT menghasilkan representasi kalimat berdimensi 384 dengan mempertimbangkan konteks dan struktur bahasa. Model pretrained MiniLM digunakan untuk menghasilkan embedding yang efektif dalam memahami makna kalimat.

Reduksi Dimensi dan Visualisasi

Karena representasi seperti TF-IDF dan BERT memiliki dimensi tinggi, reduksi dimensi dilakukan untuk efisiensi komputasi dan keperluan visualisasi tanpa kehilangan informasi penting.

1. **Truncated SVD**

   Truncated SVD mereduksi dimensi TF-IDF tanpa mengubah struktur sparse-nya, sehingga cocok untuk data teks dan mendukung visualisasi yang lebih efisien.

2. **PCA (Principal Component Analysis)**
   PCA mereduksi dimensi secara linier dengan menjaga variansi terbesar. Meski sederhana dan cepat, PCA kurang efektif untuk menangkap struktur non-linier dalam teks.
3. **t-SNE (t-distributed Stochastic Neighbor Embedding)**
   t-SNE memproyeksikan data berdimensi tinggi ke dua dimensi dengan menjaga hubungan lokal antar titik. Hasil visualisasinya menarik, tetapi sensitif terhadap parameter dan memakan waktu.
4. **UMAP (Uniform Manifold Approximation and Projection)**
   UMAP menjaga struktur lokal dan global saat memetakan data ke dua dimensi. Dibanding t-SNE, UMAP lebih cepat, stabil, dan memberikan visualisasi distribusi dokumen yang lebih informatif.

Modelling

1. **LDA Topic Modelling**

Latent Dirichlet Allocation (LDA) adalah metode pemodelan topik yang digunakan untuk mengidentifikasi struktur topik tersembunyi dalam kumpulan dokumen. LDA mengasumsikan bahwa setiap dokumen terdiri dari campuran beberapa topik, dan setiap topik direpresentasikan oleh distribusi kata-kata tertentu. Dengan ini, LDA membantu mengelompokkan dokumen berdasarkan tema atau topik yang mendasarinya, sehingga memudahkan analisis dan pemahaman isi teks dalam jumlah besar secara otomatis.

Dalam penelitian ini, proses pemodelan topik dimulai dengan mengonversi dokumen menjadi representasi Bag-of-Words (BoW). Setiap dokumen diubah menjadi daftar pasangan kata dan frekuensinya berdasarkan kamus yang dibuat dari seluruh korpus. Selanjutnya, model Latent Dirichlet Allocation (LDA) dilatih secara iteratif dengan variasi jumlah topik mulai dari 5 hingga 10, untuk mengevaluasi pengaruh jumlah topik terhadap kualitas model. Setiap model yang dihasilkan kemudian dievaluasi menggunakan metrik coherence score, yang mengukur kekompakan dan konsistensi semantik kata-kata dalam masing-masing topik. Berdasarkan nilai coherence tertinggi yang diperoleh, jumlah topik optimal ditentukan, yaitu sebanyak 8 topik. Model LDA akhir kemudian dilatih ulang dengan parameter num_topics=8 untuk menghasilkan pemodelan topik yang terbaik dan paling representatif dari data.

2. **Klasifikasi Topik dengan Random Forest**

Penelitian ini bertujuan untuk mengklasifikasikan topik paper ilmiah berdasarkan abstraknya. Dataset dibagi menjadi data latih dan data uji dengan rasio 80:20 menggunakan stratified sampling agar proporsi kelas tetap seimbang. Fitur teks diubah menjadi vektor numerik menggunakan metode Term Frequency-Inverse Document Frequency (TF-IDF). Karena terdapat ketidakseimbangan kelas dalam data, teknik Synthetic Minority Over-sampling Technique (SMOTE) diterapkan pada data latih untuk menghasilkan sampel sintetis dari kelas minoritas, sehingga distribusi kelas menjadi lebih seimbang. Model klasifikasi yang digunakan adalah Random Forest Classifier, sebuah

algoritma ensembel berbasis pohon keputusan yang efektif untuk data berdimensi tinggi dan kompleksitas non-linear. Model dilatih pada data latih hasil SMOTE dan diuji pada data uji.

Evaluasi

1. **Topic Modelling**

Pada topic modelling, digunakan dua metrik evaluasi, yaitu coherence score dan perplexity. Coherence score digunakan untuk mengukur sejauh mana kata-kata dalam suatu topik memiliki keterkaitan semantik, di mana nilai yang tinggi menunjukkan topik yang lebih bermakna. Sementara itu, perplexity mengukur seberapa baik model probabilistik seperti LDA dalam memprediksi data yang belum pernah dilihat, dengan nilai yang lebih rendah menunjukkan pemahaman model yang lebih baik terhadap distribusi kata.

2. **Klasifikasi**

Untuk evaluasi model klasifikasi, digunakan metrik-metrik standar seperti accuracy, recall, precision, F1-score, dan confusion matrix. Accuracy memberikan gambaran umum terhadap performa model, meskipun bisa menyesatkan jika terjadi ketidakseimbangan kelas. Recall menekankan pada kemampuan model dalam mengenali seluruh data positif, sementara precision menilai ketepatan prediksi positif yang dilakukan. F1-score menggabungkan kedua metrik tersebut dalam satu nilai yang seimbang, terutama penting saat distribusi kelas tidak merata. Terakhir, confusion matrix digunakan untuk memberikan representasi visual terhadap jumlah prediksi benar dan salah di tiap kelas, membantu dalam analisis kesalahan klasifikasi secara lebih rinci.

3. Batch Scoring di Spark

Dalam implementasi sistem inferensi skala besar, digunakan pendekatan batch scoring dengan bantuan Apache Spark. Spark dipilih karena kemampuannya dalam menangani data besar secara terdistribusi dan paralel, sehingga sangat cocok untuk memproses kumpulan data mingguan yang terus bertambah. Data awal dimuat ke dalam Spark DataFrame, lalu dikonversi ke format Pandas DataFrame untuk dilakukan proses transformasi teks menggunakan model TF-IDF yang telah dilatih sebelumnya.

Setelah vektorisasi selesai, data dimasukkan ke model klasifikasi berbasis Random Forest untuk melakukan prediksi. Model dan vektorisasi telah diserialisasi menggunakan joblib agar dapat dimuat ulang dengan cepat tanpa perlu pelatihan ulang. Hasil prediksi kemudian disimpan ke dalam file CSV atau format Parquet, tergantung kebutuhan downstream dan efisiensi I/O. Contoh hasil prediksi untuk minggu ke-23 menunjukkan bahwa mayoritas data diklasifikasikan ke dalam kelas 0 (48.47%), dengan proporsi cukup besar juga pada kelas 1 (25.80%) dan kelas 2 (21.67%). Dominasi pada tiga kelas ini menjadi acuan baseline distribusi untuk monitoring model drift.

4. Monitoring Sederhana – Deteksi Model Drift

Monitoring model drift dilakukan dengan membandingkan distribusi prediksi antar batch, yaitu antara batch baseline (prediksi awal saat model pertama kali di-deploy) dan batch current (prediksi terbaru). Berdasarkan hasil visualisasi distribusi prediksi, terlihat bahwa proporsi prediksi antar label pada batch baseline dan batch current relatif stabil. Label 0 (Computer Science) masih mendominasi prediksi di kedua batch dengan

proporsi sekitar 48-49%, diikuti oleh label 1 (Physics) dan label 2 (Mathematics) dengan proporsi yang juga cukup konsisten. Selisih proporsi antar batch untuk setiap label berada di bawah 5%, sehingga tidak terdapat indikasi model drift yang signifikan. Hal ini menunjukkan bahwa model masih memiliki kinerja yang stabil terhadap data terbaru, dan belum memerlukan proses retraining. Monitoring seperti ini perlu dilakukan secara berkala untuk memastikan keandalan model seiring perubahan karakteristik data yang masuk.

5. Optimasi Performa

Agar proses inferensi berjalan efisien dan dapat diskalakan, beberapa strategi optimasi diterapkan. Pertama, caching diterapkan untuk menyimpan model, vectorizer, dan hasil prediksi ke dalam file cache. Sebelum dilakukan proses prediksi atau pelatihan ulang, sistem akan terlebih dahulu mengecek apakah file tersebut sudah tersedia. Jika ya, maka akan langsung dimuat dari disk, menghemat waktu dan sumber daya komputasi. Kedua, data prediksi batch dipartisi berdasarkan minggu menggunakan partitionBy('week') saat disimpan dalam format Parquet. Dengan ini, Spark dapat mengakses dan memproses subset data mingguan secara efisien tanpa perlu membaca seluruh dataset. Ketiga, Spark memanfaatkan prinsip lazy evaluation, di mana transformasi data tidak langsung dieksekusi sampai aksi akhir dijalankan. Hal ini memastikan bahwa hanya transformasi yang benar-benar dibutuhkan yang akan dijalankan, mengurangi beban komputasi dan mempercepat eksekusi secara keseluruhan.

Hasil

Berdasarkan visualisasi distribusi topik hasil pemodelan LDA, ditemukan bahwa topik 3 menjadi yang paling dominan dengan sekitar 5.5 ribu dokumen (sekitar 36% dari total), diikuti oleh Topik 7 dengan 4.5 ribu dokumen (sekitar 30%), dan Topik 5 dengan 3 ribu dokumen (sekitar 20%). Sementara itu, Topik 4 hanya muncul pada sekitar 1 ribu dokumen (kurang dari 7%), menjadikannya topik dengan representasi terendah. Distribusi ini menguatkan dugaan bahwa topik-topik tertentu memang menjadi fokus pembahasan utama dalam kumpulan artikel yang dianalisis, ada kecenderungan tema tertentu yang lebih banyak dibahas dalam literatur, sedangkan topik lain hanya muncul pada sebagian kecil dokumen. Distribusi ini memberikan gambaran awal mengenai konsentrasi isi atau tema utama dari kumpulan artikel yang diteliti.

Evaluasi performa model klasifikasi seperti pada Gambar x menunjukkan bahwa akurasi keseluruhan mencapai 78%, yang menandakan model cukup baik dalam mengenali mayoritas topik berdasarkan abstrak paper. Dari sisi F1-score, performa terbaik diperoleh pada kelas Mathematics (0.81), Physics (0.86), dan Computer Science (0.80). Hal ini menunjukkan bahwa model mampu membedakan dengan baik abstrak-abstrak yang tergolong dalam topik-topik tersebut, yang juga merupakan kelas dengan jumlah data terbesar.

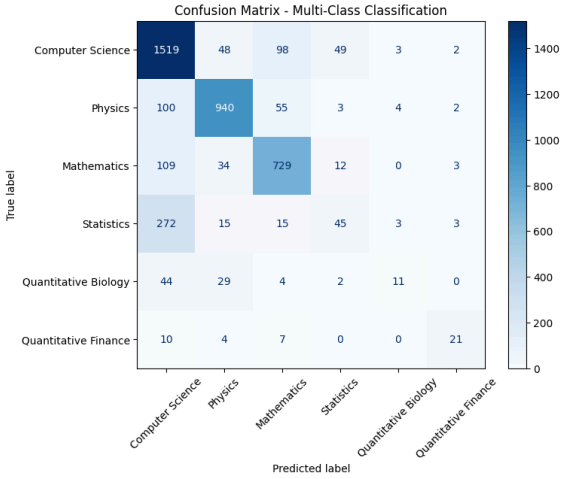
Classification Report:

	precision	recall	f1-score	support
Computer Science	0.74	0.88	0.80	1719
Physics	0.88	0.85	0.87	1104
Mathematics	0.81	0.82	0.81	887
Statistics	0.41	0.14	0.21	353
Quantitative Biology	0.60	0.17	0.26	90
Quantitative Finance	0.70	0.50	0.58	42
accuracy			0.78	4195
macro avg	0.69	0.56	0.59	4195
weighted avg	0.76	0.78	0.76	4195

Gambar 2. Classification report model Random Forest

Namun, performa model menurun drastis pada kelas dengan jumlah data yang lebih kecil seperti Statistics, Quantitative Biology, dan Quantitative Finance. F1-score untuk Statistics hanya 0.21 dan Quantitative Biology 0.26, menunjukkan bahwa model kesulitan dalam mengklasifikasikan paper-paper dengan topik tersebut. Hal ini juga tercermin pada nilai macro average F1-score sebesar 0.59, yang menandakan adanya ketidakseimbangan performa antar kelas.

Confusion matrix dari model ini ditunjukkan dalam Gambar x. Dari confusion matrix tersebut, terlihat bahwa banyak paper dengan label Statistics justru diklasifikasikan sebagai Computer Science, serta adanya kesalahan serupa pada Quantitative Biology yang juga cenderung diklasifikasikan ke kelas mayoritas. Hal ini menunjukkan bahwa meskipun SMOTE telah diterapkan, model masih belum cukup sensitif terhadap pola unik dari kelas-kelas minor.

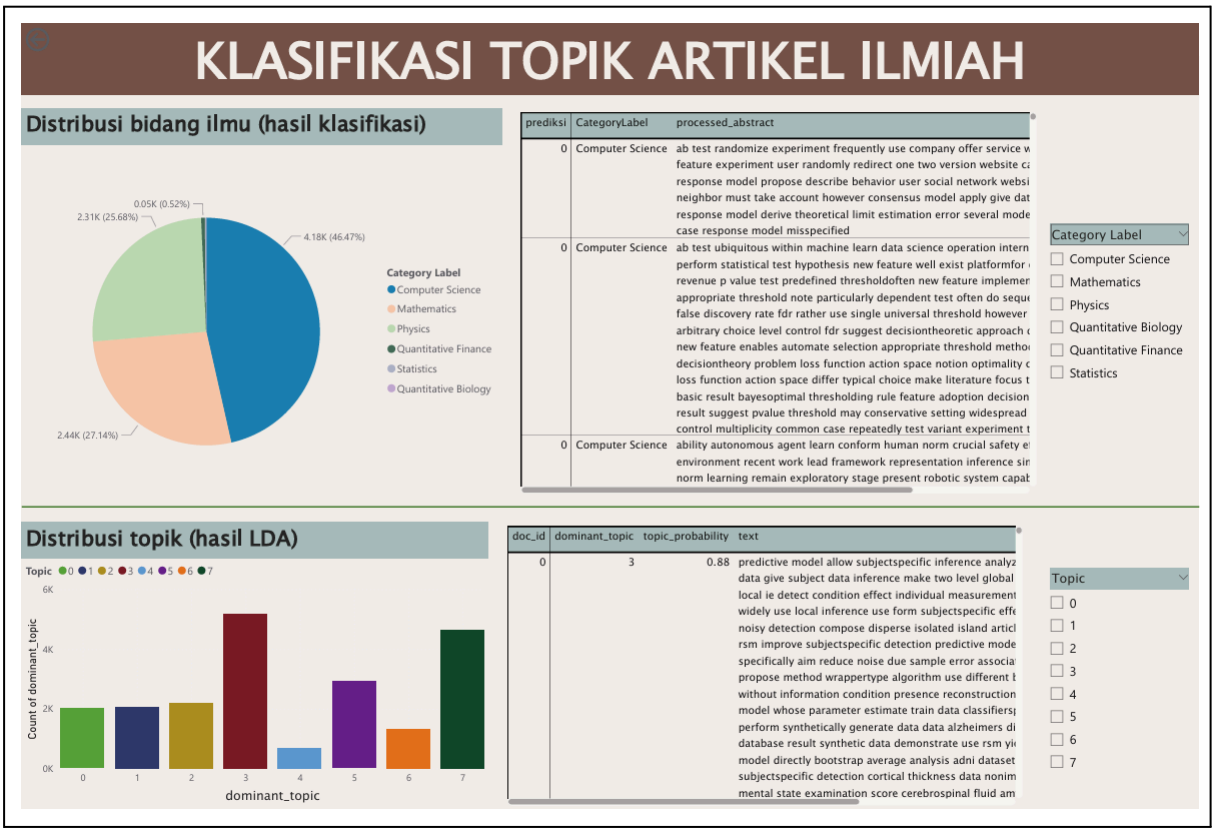


Gambar 3. Confusion matrix model Random Forest

Secara keseluruhan, model Random Forest menunjukkan performa yang kuat pada topik dominan namun masih perlu ditingkatkan kemampuannya dalam membedakan kelas-kelas dengan data yang lebih sedikit atau lebih mirip secara konten. Pendekatan tambahan seperti peningkatan fitur, rebalancing lebih lanjut, atau eksplorasi model lain bisa menjadi strategi ke depan.

Untuk hasil klasifikasi bidang ilmu, ditemui bahwa sebagian besar artikel termasuk dalam kategori Computer Science, yaitu sebanyak ± 4800 dokumen atau sekitar 46,47% dari total data. Disusul oleh kategori Mathematics sebanyak ± 2440 dokumen

(27,14%), dan Physics dengan sekitar 2300 dokumen (25,68%). Sementara itu, tiga kategori lainnya (Quantitative Finance, Statistics, dan Quantitative Biology) memiliki jumlah dokumen yang sangat kecil, dengan persentase masing-masing di bawah 1%. Visualisasi dalam bentuk diagram lingkaran memperjelas dominasi kategori Computer Science. Selain itu, disediakan tabel yang menampilkan hasil klasifikasi per dokumen beserta abstrak yang telah diproses (processed_abstract), yang memungkinkan eksplorasi lebih lanjut terhadap isi teks setiap dokumen.



Gambar 4. Dashboard Visualisasi Hasil

Kesimpulan

Proyek ini berfokus pada penerapan *text mining pipeline* secara menyeluruh terhadap kumpulan abstrak artikel ilmiah dari Kaggle, dengan tujuan melakukan *topic modeling* dan klasifikasi topik secara otomatis. Metode yang digunakan meliputi penyimpanan data di MongoDB, *preprocessing* teks, ekstraksi fitur (TF-IDF, Word2Vec, BERT), reduksi dimensi, pemodelan topik menggunakan LDA, serta klasifikasi menggunakan algoritma Random Forest yang dilatih dengan teknik SMOTE untuk mengatasi ketidakseimbangan kelas.

Hasil yang diperoleh menunjukkan bahwa model LDA dengan 8 topik menghasilkan *coherence score* tertinggi, menandakan kualitas pemodelan topik yang baik. Sementara itu, model klasifikasi Random Forest mencapai performa memuaskan berdasarkan metrik *accuracy, precision, recall*, dan *F1-score*, dengan dukungan proses inferensi skala besar menggunakan batch *scoring* di Apache Spark.

Ke depan, sistem ini dapat ditingkatkan dengan mengintegrasikan *real-time streaming inference*, memperluas korpus ke berbagai domain teks lainnya, serta menerapkan *fine-tuning* model *embedding* seperti BERT agar hasil klasifikasi dan pemodelan topik semakin kontekstual dan relevan. Selain itu, penerapan *dashboard* interaktif dan API publik juga disarankan untuk mempermudah pemanfaatan sistem oleh pengguna non-teknis.