**Journal of Applied Informatics and Computing (JAIC)**
Vol.8, No.1, July 2024, pp. 194~202
e-ISSN: 2548-6861


# Comparative Analysis of the Performance of Decision Tree and Random Forest Algorithms in SQL Injection Attack Detection

**Alfatarizky Budi Aulianoor [1]*, Muhammad Koprawi [2]***
* Teknik Komputer, Universitas Amikom Yogyakarta
alfatarizkyba@students.amikom.ac.id [1] , koprawi@amikom.ac.id [2]

**Article Info**

***Article history:***

Received 2024-07-05
Revised 2024-07-09
Accepted 2024-07-11

***Keyword:***

*Machine Learning,*
*Decision Tree,*
*Random Forest,*
*SQL Injection,*
*Database.*

**ABSTRACT**

This study compares the performance of two machine learning algorithms the Decision Tree and Random Forest. SQL Injection attacks continue to threaten web applications because they exploit vulnerabilities by injecting malicious code into SQL statements executed on database servers. Therefore, machine learning algorithms are used to identify SQL Injection attacks. The dataset used is 33761 in the form of random query data input in a CSV tabular containing sentence and label columns. The research software used is Google Colaboratory and Microsoft Edge. The series of research conducted by Collect Data is data collection, Preprocessing handling missing values, deleting rows that contain duplicates, and the same query having different labels. Train and Test is used to build models and prepare test data, Build and Compile involves building Decision Tree and Random Forest models. The final step is to evaluate both algorithm models to determine which performs better. After conducting a series of research processes, the results of the Random Forest algorithm are slightly better than the Decision Tree algorithm, with an accuracy of 99.81%, precision of 99.79%, recall of 99.65%, and an average F1-score of 99.72%.



## I. PENDAHULUAN

Database adalah kumpulan suatu data yang disimpan secara sistematis di komputer, baik secara lokal atau di cloud menggunakan teknologi seperti database relasional atau database NoSQL yang dapat digunakan untuk menghasilkan informasi yang akurat[1]. Database relasional menyimpan data ke dalam bentuk tabel yang telah dinormalisasikan dan terstruktur, database tidak dapat menangani data yang tidak dinormalisasi dan ukuran besar sebagai contoh perusahaan seperti Google, Facebook, dan Amazon memilih database NoSQL sebagai opsi penyimpanan database mereka[2]. Menurut Jin Zhang, dkk. (2023), membahas perencanaan database emosional dalam pembelajaran lingkungan belajar menyelidiki desain dan ekspresi untuk mengatasi masalah metode anotasi data yang berisi tiga bentuk data berupa gambar ekspresi, urutan ekspresi, dan klip video, namun pada penelitian sebelumnya membentuk basis data yang berjumlah 8.000 gambar ekspresi yang berisi 36 peserta didik[3]. Sedangkan pada penelitian ini penulis menggunakan dataset berupa data tabular CSV yang berisi 2 kolom, yaitu *sentence* dan label[4]. File *Comma-Separated Value* (CSV) adalah format yang populer untuk data tabular karena memiliki kesederhanaan dan kemudahan bagi pengguna[5].

Database relasional seperti MySQL [6] menyimpan data yang terstruktur merupakan salah satu aspek yang membedakan database NoSQL dari database relasional adalah bahwa tabel dan bahasa SQL tidak selalu digunakan[7]. MySQL dan SQL merupakan istilah yang berhubungan dengan database, walaupun keduanya sering digunakan namun keduanya memiliki fungsi yang berbeda. Dengan jumlah pengguna *internet* yang meningkat setiap tahun, telah terjadi perkembangan layanan yang diakses melalui *internet* seperti *e-commerce*, dukungan bisnis, dan repositori data besar[8].

Serangan SQL *Injection* dianggap serangan paling berbahaya dari kategori injection karena membahayakan layanan keamanan utama seperti kerahasiaan otentikasi,

integritas, dan menjalankan perintah jarak jauh bisa disebut juga *remote command execution*[9]. Otentikasi memungkinkan penyerang untuk mengakses aplikasi database dengan menggunakan *username* palsu dan *password*. Integritas data yang dirusak membantu penyerang dalam mengubah maupun menghapus beberapa informasi data yang berada dalam database[10]. Sistem operasi host dapat dipengaruhi oleh eksekusi perintah jarak jauh. Konfigurasi dan pemrograman database yang tepat diperlukan untuk mencegah serangan SQL *Injection*. Selama serangan SQL *Injection*, penyerang dapat mengakses informasi database dan melakukan tindakan di database *back-end* untuk tujuan seperti mengekstrak, mengubah, dan menghapus data[11].

Menurut penelitian Binh An Pham dan Vinitha Subburaj (2020), menemukan bahwa ada beberapa metode dapat digunakan untuk menyelesaikan suatu masalah SQL *Injection*. Studi tersebut menggunakan algoritma *machine learning* untuk menemukan kode berbahaya dalam data menggunakan kode SQL dan *input* pengguna[12]. Meskipun demikian, sebagian besar dari penelitian tidak dapat menemukan semua jenis SQL *Injection*. Dalam mengatasi masalah tergantung pada jenis serangan, dengan menggunakan berbagai pendekatan termasuk *machine learning, hybrid* statis, dan dinamis serta alat berbasis web. Pendekatan ini digunakan untuk analisis kerentanan, pemindaian, mengurangi risiko, deteksi, pencegahan dan menghindari serangan berbasis aplikasi web. Oleh karena itu, pendekatan tersebut digunakan karena mekanisme keamanan yang ada pada server database[13]. Banyak industri menggunakan algoritma machine learning, seperti deteksi objek, pengenalan suatu pola, dan penelitian. Konsep dasar dari algoritma ini adalah membuat algoritma untuk menerima *input* dan memprediksi *output* menggunakan analisis statistik[14].

Penelitian ini memanfaatkan algoritma Decision Tree dan Random Forest[15]. Dalam penelitian yang dilakukan oleh Aldiyansah, dkk. (2024), algoritma Decision Tree dan Random Forest digunakan untuk menemukan penerima Bantuan Pangan Non Tunai (BNPT) di Desa Slangit pada bulan Juli dan Agustus 2023. Hasilnya menunjukkan bahwa algoritma ini memiliki tingkat akurasi 97.86%. Kedua algoritma tersebut memenuhi syarat untuk Presisi, *recall*, dan F1-*score*, menurut evaluasi model[16]. Namun, penelitian sebelumnya fokus pada algoritma Decision Tree dan Random Forest serta nilai Presisi, *recall*, dan F1-*score* mereka. Sedangkan dalam penelitian ini menambahkan model akurasi sebagai perbandingan tambahan.

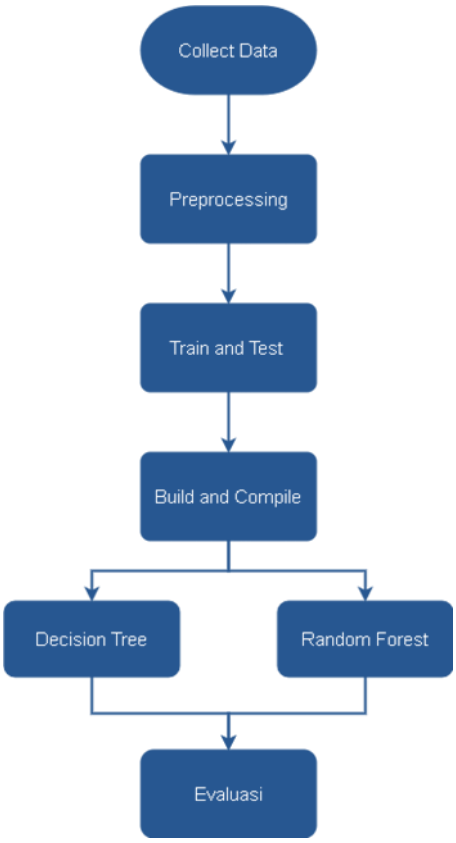
Untuk klasifikasi data dengan kombinasi dapat menggunakan Random Forest dengan bantuan Decision Tree sebagai pengenalan *preprocessing* dataset dalam pembuatan model klasifikasi biner[17]. Decision Tree, atau disebut sebagai "pohon keputusan", adalah algoritma machine learning yang dimanfaatkan untuk tujuan klasifikasi[18]. Decision Tree merupakan salah satu bagian dari algoritma supervised learning yang digunakan untuk klasifikasi data numerik dan klasifikasi data berbentuk kelas. Decision tree memiliki struktur sederhana memungkinkan pohon untuk memproses sejumlah data besar dengan cepat, namun menjadi lebih kompleks dan tujuan menjadi lebih sulit untuk dicapai[19].

Algoritma Decision Tree bagian dari algoritma pembelajaran yang paling diminati untuk penyelesaian klasifikasi serta memiliki kategori dan variabel sebagai target[20]. Studi ini menjelaskan penggunaan 311 kumpulan dataset yang tersedia secara gratis di sumber *online* melakukan perbandingan kumpulan data dengan hasil biner, kumpulan data dengan banyak prediksi, kumpulan data dengan hasil yang tidak seimbang, dan beberapa jenis Random Forest yang standar. Menurut penelitian ini, metode pemilihan variabel terbaik untuk kumpulan data yang berisi hasil biner adalah metode Jiang dan metode yang diimplementasikan dalam paket VSURF R. Untuk kumpulan data dengan banyak prediksi, metode dari paket R varSEIRF dan Boruta diterapkan karena memiliki efisiensi dalam pemecahan masalah[21].

Random Forest merupakan metode ansambel paling kuat dan baik dengan kinerja yang bagus dalam kasus data yang berdimensi sebanding atau lebih besar[22]. Random Forest berada di bawah pengawasan *machine learning*, seperti Decision Tree. Untuk kategorisasi, Random Forest menggunakan suara mayoritas dari masing-masing kelompok untuk membuat banyak Decision Tree dari sejumlah sampel data[23]. Setiap Random Forest dapat dibuat dengan sampel data secara acak dari sampel yang disiapkan. Selain itu, hanya sebagian atribut yang dipilih secara acak untuk memecah data pada setiap titik Decision Tree. Dengan adanya variasi tersebut, Random Forest dapat mengurangi *overfitting* serta meningkatkan konsep dengan lebih mudah diprediksi dan diklasifikasikan[24]. Dengan memanfaatkan banyak pohon keputusan dan menggabungkan hasilnya bertujuan untuk mengetahui bagaimana performa kedua algoritma *machine learning*, yaitu algoritma Decision Tree dan Random Forest bekerja dengan baik saat digunakan untuk membandingkan dataset dari *Kaggle*. Melalui perbandingan kedua algoritma ini, penelitian dapat menunjukkan bagaimana dataset yang berupa *query* acak file berjenis *Comma-Separated Value* (CSV) dengan algoritma Random Forest dapat meningkatkan akurasi dan mengurangi kesalahan jika dibandingkan dengan penggunaan algoritma Decision Tree dalam konteks deteksi SQL *Injection*.

## II. METODE

Penelitian ini menggunakan algoritma machine learning seperti algoritma Decision Tree dan Random Forest. Untuk dataset melalui Kaggle yang berupa tabular CSV berisi kolom sentence dan label. Software penelitian yang digunakan adalah *Google Collaboratory* dan *Microsoft Edge*. Berikut merupakan flowchart rangkaian proses penelitian yang akan dilakukan pada Gambar 1.




Gambar 1. Alur Penelitian

### A. *Collect Data*

Proses penelitian dimulai dari tahap collect data yang merupakan pengumpulan data dalam machine learning, seperti proses mengidentifikasi, mendapatkan data, dan mempersiapkan data yang relevan untuk kebutuhan penelitian. Pengambilan data penelitian dari website kaggle.com dengan kata kunci "SQL Injection Dataset". Dataset berupasqlv2.csv digunakan karena relevan dengan penelitian, dataset berisi *query* acak berdasarkan dari log SQL yang di-*upload* oleh *publisher*. Metode observasi digunakan saat pengumpulan data dengan menentukan jenis data yang akan diolah pada penelitian berupa data *Comma-Separated Values* (CSV) yang berisi beberapa kolom mencakup kalimat berbentuk *query* SQL dan kolom label yang terdapat pada dataset ini apakah terindikasi sebagai serangan SQL *Injection* atau tidak.

### B. *Preprocessing*

Pada *preprocessing* merupakan langkah untuk mempersiapkan data uji coba menjadi data yang sudah siap digunakan untuk analisis dan pemodelan. Sebelumnya melakukan pengecekan dataset sebelum dilakukan pembersihan. Setelah itu, pembersihan data dilakukan dari atribut yang tidak diperlukan untuk penelitian. Tujuan utama adalah memastikan bahwa kebersihan dan kualitas data sebelum dilakukan analisis atau proses pemodelan, langkah-langkah dalam proses pembersihan data mencakup deteksi serta penanganan nilai yang hilang, penghapusan baris yang berisi duplikat, dan query yang sama memiliki label yang berbeda. Lalu melakukan pengecekan kembali dataset yang telah dilakukan pembersihan dataset. Selanjutnya analisa Word Clouds dan fitur-fitur yang digunakan dalam ekstraksi.

### C. *Train and Test*

Setelah data selesai diproses, langkah berikutnya meliputi *splitting* data menjadi dua bagian menjadi data *training dan* data pengujian. Data *training* digunakan untuk melatih model dimana proses mengenali pola dan hubungan dalam data. Data ini penting karena memungkinkan model untuk mengembangkan kemampuan prediksi berdasarkan kinerja model data yang telah dilakukan pada *preprocessing*. Dengan menguji model data dapat mengukur akurasi dan penyamarataan model dalam memprediksi data baru. Proses pembagian ini memastikan bahwa model tidak hanya bekerja dengan baik pada data yang sudah dikenal, tetapi juga mampu memberikan prediksi yang akurat pada data baru pada proses selanjutnya.

### D. *Build and Compile (Decision Tree and Random Forest)*

Tahap berikutnya adalah *Build* dan *Compile* dalam pengolahan data menggunakan algoritma Random Forest dan Decision Tree merupakan tahap penting dalam pengembangan model. Tahap *build* melibatkan pembangunan model Random Forest dan Decision Tree berdasarkan data yang telah diproses sebelumnya. Decision tree dirancang dengan membagi data berdasarkan keputusan berdasarkan fitur-fitur yang ada, sementara Random Forest menggabungkan beberapa Decision Tree yang telah dirancang secara terpisah tujuannya adalah untuk memastikan bahwa model siap digunakan untuk membuat prediksi yang akurat berdasarkan data yang baru dikumpulkan. Lalu, setelah menganalisa dan memprediksi masuk ke dalam proses Confusion Matrix dilihat pada Tabel I berikut ini.

TABEL I
ISTILAH CONFUSION MATRIX

| **Kelas Prediksi** | **Kelas sebenarnya** | |
|---|---|---|
| | **Prediksi Positif** | **Prediksi Negatif** |
| Negatif | TP | FN |
| Positif | FP | TN |

*True Positive* (TP) dan *True Negative* (TN) terjadi saat prediksi sesuai dengan kondisi yang sebenarnya, sedangkan *False Positive* (FP) dan *False Negative* (FN) terjadi ketika prediksi tidak sesuai dengan kondisi sebenarnya. *True Positive* (TP) merupakan data positif yang teridentifikasi secara benar. *True Negative* (TN) merupakan jumlah data negatif yang diklasifikasikan secara benar. *False Positive* (FP) merupakan data negatif yang terklasifikasi sebagai data positif, sedangkan *False Negative* (FN) merupakan data positif yang teridentifikasi sebagai data negatif. Dari nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) dapat dihitung untuk

menentukan akurasi, presisi, *recall*, dan F1-*score*. Sehingga Tabel II di bawah dapat digunakan sebagai pedoman perhitungan pada penelitian.

TABEL II
PENJELASAN INDIKASI DAN RUMUS

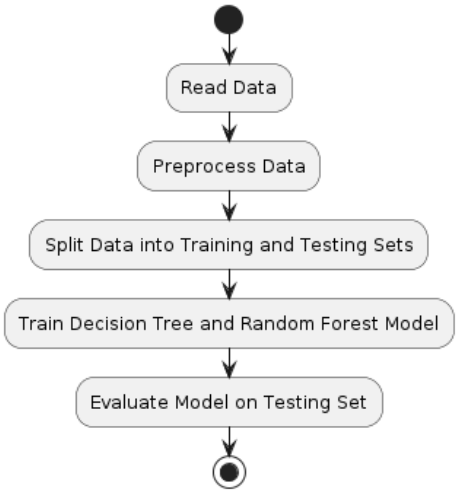
| Indikasi | Penjelasan | Rumus |
|---|---|---|
| Akurasi | Akurasi merupakan total prediksi benar, baik bernilai positif maupun negatif. | Akurasi = $\frac{TP + TN}{TP + TN + FP + FN}$ |
| Presisi | Presisi merupakan prediksi positif yang benar-benar positif | Presisi = $\frac{TP}{TP + FP}$ |
| *Recall* | *Recall* merupakan *instance* positif yang benar terprediksi positif | *Recall* = $\frac{TP}{TP + FN}$ |
| F1-*score* | F1-*score* merupakan gabungan nilai rata-rata dari Presisi dan *Recall* | F1-*score* = 2 x $\frac{Presisi\ x\ Recall}{Presisi + Recall}$ |

### E. Evaluasi

Selanjutnya membandingkan model yang telah dirancang, yaitu model Decision Tree dan Random Forest. Langkah terakhir mengevaluasi kinerja kedua model algoritma Decision Tree dan Random Forest, kemudian diuji dan dibandingkan menggunakan data pengujian. Evaluasi tersebut disajikan dalam bentuk gambar. Tujuan evaluasi ini adalah untuk menentukkan model mana yang memiliki performa terbaik dalam hal akurasi, presisi, *recall*, dan F1-*score*.

## III. HASIL DAN PEMBAHASAN

Algoritma Decision Tree merupakan salah satu algoritma *machine learning* yang sederhana namun efektif. Algoritma ini bekerja dengan membagi dataset menjadi subset yang lebih kecil, meskipun efektif Decision Tree memiliki kecenderungan untuk overfitting terhadap data pelatihan terutama jika tidak melakukan pembersihan data-data dengan baik. Sebaliknya Random Forest suatu kumpulan dari beberapa Decision Tree yang menjalankan kinerja secara bersama-sama dalam meningkatkan akurasi prediksi dan mengurangi risiko *overfitting*. Dengan memanfaatkan banyak pohon keputusan, Random Forest mampu memberikan hasil yang lebih stabil dan lebih baik dalam akurasi. Melalui perbandingan kedua algoritma ini penelitian dapat menunjukkan bagaimana Random Forest dapat meningkatkan akurasi dan mengurangi kesalahan jika dibandingkan dengan penggunaan Decision Tree dalam konteks deteksi SQL *Injection*. Oleh karena itu berikut merupakan rancangan *machine learning* pada penelitian ini, dengan tahapan-tahapan sebagai berikut.




Gambar 2. Perancangan Machine Learning

### A. Read Data

Pada penelitian ini, dataset diperoleh melalui website Kaggle dengan mencari kata kunci "SQL Injection Dataset" dan nantinya akan muncul beberapa dataset yang ditampilkan. Dalam penelitian ini mengambil dataset dengan nama "sqlv2.csv" berisi sebanyak 33761 inputan data query acak berupa tabular CSV kolom sentence dan label. Dataset yang sudah diunduh kemudian di-*upload* pada folder *Google Drive* yang telah dipersiapkan untuk melalui proses *mounting* pada sebelum masuk ke dalam perancangan model pada platform *Google Collaboratory*. Untuk langkah awal read dataset yang telah di *mount* pada *Google Collaboratory*, kemudian lanjut pada *preprocess* data.

### B. Preprocess Data

Pada tahap *preprocess* data analisis distribusi titik data dilakukan untuk mengevaluasi keakuratan *query* terhadap label, visualisasi wordclouds, dan feature extraction. Sebelumnya *preprocessing*, dataset menunjukkan bahwa label 1 memiliki sebanyak 11456 data dengan persentase 33.93%, sementara pada label 0 memiliki dataset sebanyak 22305 data dengan persentase 66.07% ini memberikan gambaran awal tentang proporsi data antara kedua label tersebut. Pada Gambar 3. Melakukan pengecekan baris duplikat.

```
data.dropna(inplace=True)
data.isnull().sum()
```

```
Sentence    0
Label       0
dtype: int64
```

```
#checking if there any duplicate rows present in dataset
data.duplicated(subset = ['Sentence','Label']).sum()
```

```
30
```

Gambar 3. Pengecekan Duplikat

Pengecekan duplikat terdapat 30 baris yang merupakan duplikat artinya telah dihulang, sehingga dapat menghapus baris duplikat tersebut dilihat pada Gambar 4.

```
#remove the duplicate rows
data.drop_duplicates(subset = ['Sentence','Label'],inplace = True)

#checking the shape of data after removing duplicates
data.shape

(33727, 2)

#rechecking
data.duplicated().sum()

0
```

Gambar 4. Penghapusan Baris Duplikat

Melakukan pemeriksaan apa ada nilai (kosong) yang hilang pada dataset tersaji pada Gambar 5.

```
#checking if there are any null value present in dataset
data.isnull().sum()

Sentence    0
Label       0
dtype: int64
```

Gambar 5. Pengecekan Nilai NULL

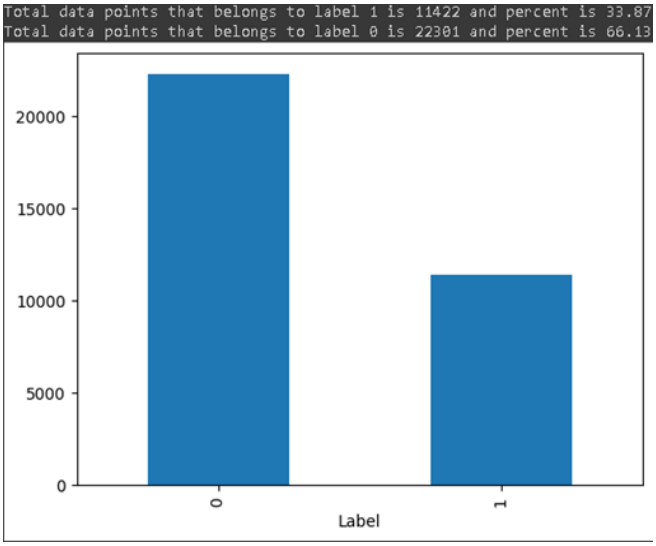
Dengan demikian, dari kode dan hasil diatas dapat disimpulkan bahwa tidak ada nilai “null” dalam kolom “Sentence” dan “Label” pada dataset yang sedang diperiksa.

```
data.duplicated(subset = ['Sentence'],keep = False).sum()

4

# drop the duplicates
data.drop_duplicates(subset = ['Sentence'],keep= False,inplace = True)

data.shape

(33723, 2)
```

Gambar 6. Pengecekan Query Kalimat Dengan Label

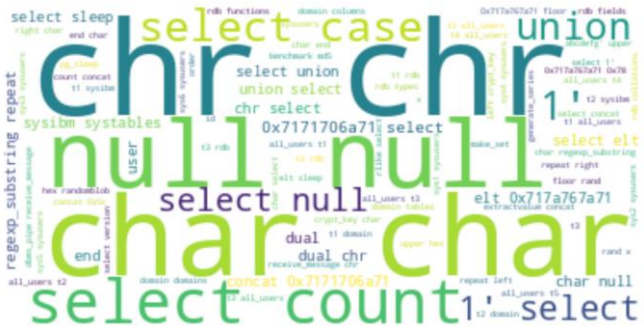
Kode pada Gambar 6. digunakan untuk menghitung jumlah baris duplikat dalam kolom *sentence* pada DataFrame data. Fungsi data.duplicated() menghasilkan sebuah boolean di mana setiap nilai adalah True, jika baris tersebut adalah duplikat dari baris lain berdasarkan kolom “Sentence”. Perintah keep=false artinya duplikat akan ditandai sebagai True. Kemudian, fungsi .sum menghitung jumlah True dalam boolean tersebut dengan menunjukkan jumlah total baris duplikat berdasarkan kolom “Sentence”. Hasilnya ada 4 baris yang dianggap duplikat berdasarkan nilai-nilai yang ada dalam kolom “Sentence”. Setelah penghapusan baris duplikat pada kolom “Sentence” sekarang hanya memiliki 2 baris.

Setelah dilakukan penghapusan baris yang berisi duplikat, penanganan nilai yang hilang ,dan query yang sama memiliki label yang berbeda. Lalu melakukan pengecekan dataset, hasilnya pada label 1 mengalami sedikit penurunan menjadi sebesar 11422 data dengan persentase 33.87% dan label 0 sebanyak 22301 data dengan persentase 66.13%. Perubahan ini dapat dilihat pada Gambar 7. Yang menampilkan titik data setelah preprocessing. Meskipun perubahan persentase dan jumlah data tidak signifikan, langkah ini penting untuk memastikan bahwa dataset bebas dari duplikasi yang bisa mempengaruhi hasil analisis dan keakuratan model Decision Tree dan Random Forest.



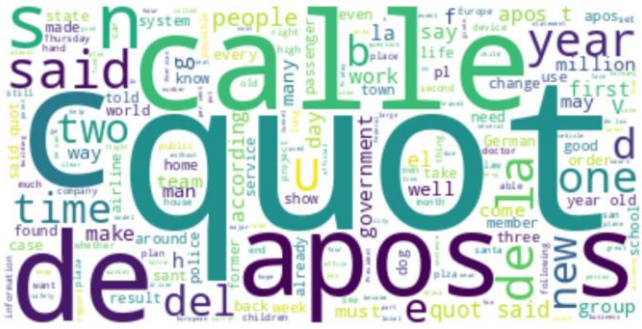

Gambar 7. Setelah Preprocessing

Dalam penelitian ini, eksperimen dilakukan dengan menggunakan visualisasi Analisis Word Clouds untuk menganalisis data teks. Word Clouds menampilkan semua kata yang terdapat dalam teks, sehingga kata-kata dengan frekuensi tertinggi akan ditampilkan dengan ukuran font yang lebih besar dalam visualisasi Word Clouds. Kalimat yang terindikasi serangan SQL *Injection* dalam Gambar 8. yaitu null, chr, char, union, select count adalah kata-kata yang paling sering muncul dalam *query* SQL *Injection*.




Gambar 8. Kata terindikasi serangan

Analisis Word Clouds kalimat non-serangan SQL *Injection* dalam Gambar 9. dapat melihat bahwa query tersebut memiliki komposisi karakter khusus, tanda baca, dan lain-lainnya. Ini semua adalah fitur yang membedakan antara Kalimat SQL normal dan Kalimat yang terindikasi SQL *Injection*.




Gambar 9. Kata tidak terindikasi serangan

Pada proses *query* dataset sebelumnya bahwa *query* terdiri dari komposisi karakter khusus, tanda baca, dan fitur yang lainnya merupakan antara SQL *Sentence* normal dan SQL *Injected Sentence*. Oleh karena itu, tidak melakukan penghapusan tanda baca, tag html, dan elemen lainnya selama proses *preprocessing* teks. Karena menghapus beberapa karakter khusus yang sangat penting untuk rekayasa fitur dan untuk membedakan *query* SQL dan SQL *Injection*. Oleh karena itu, teks hanya akan diubah menjadi huruf kecil selama preprocessing untuk analisis. Dalam proses pengujian model Decision Tree dan Random Forest, tujuan dari *feature extraction* adalah untuk mengambil fitur yang bermanfaat. Dapa melihat daftar fitur beserta penjelasan yang ditunjukkan dalam Tabel III.

TABEL III
FITUR

| Fitur | Penjelasan |
|---|---|
| Sentence_len | Panjang setiap kalimat |
| num_word_sentence | Jumlah total kata dalam sebuah kalimat |
| no_single_qts | Jumlah kutipan tunggal dalam sebuha kalimat |
| no_double_qts | Jumlah kutipan ganda dalam sebuah kalimat |
| no_punctn | Jumlah tanda baca dalam sebuah kalimat |
| no_single_cmnt | Jumlah komentar satu baris dalam sebuah kalimat |
| no_double_cmnt | Jumlah komentar baris ganda dalam sebuah kalimat |
| no_white_space | Jumlah spasi dalam sebuah kalimat |
| no_percent | Jumlah simbol persentase |
| no_log_optr | Jumlah total operator aritmatika |
| no_arith_optr | Jumlah total aritmatika |
| no_null_vall | Jumlah total nilai null dalam sebuah kalimat |
| no_hexdec_val | Jumlah total nilai desimal heksa |
| no_alphabet | Jumlah total huruf dalam sebuah kalimat |
| no_digits | Jumlah total digit |
| len_of_chr_char_null | Panjang kunci chr + chr + null fitur ini didasarkan pada wordcloud diatas |
| genuine_keywoards | Kata kunci ini juga dihasilkan menggunakan wordclouds untuk teks kelas label 0 kata kunci ini mencakup pilih, atas, pesan, ambil, gabung, rata-rata, hitung, jumlah, dan baris |

Selain itu untuk pengembangan performa model algoritma untuk deteksi serangan SQL *Injection*, dapat dilakukan dengan salah satu pendekatan dengan *hyperparameter tuning* menggunakan *Bag of Words* dengan *Count vectorizer* sebagai parameter. Proses penting untuk mengoptimalkan kinerja model adalah pengaturan hyperparameter. Dalam proses ini disajikan pada Tabel IV hyperparameter digunakan untuk mengontrol berbagai aspek dari model Decision Tree maupun Random Forest.

TABEL IV
HYPERPARAMETER

| Hyperparameter | Penjelasan |
|---|---|
| n_estimators | Menentukan jumlah pohon keputusan. |
| max_depth | Menentukan kedalaman maksimal dari setiap pohon |
| min_samples_split | Menentukan jumlah minimum sampel yang diperlukan untuk membagi node internal. |
| min_samples_leaf | Menentukan jumlah minimum sampel yang diperlukan untuk berada di node daun. |
| max_features | Menentukan jumlah fitur untuk membagi node. |
| bootstrap | Menggunakan bootsrap samples saat membangun pohon. |
| oob_score | Menggunakan sampel out-of-bag untuk memperkirakan akurasi. |
| n_jobs | Menggunakan semua prosesor yang tersedia |
| random_state | Seed untuk pengacakan. |
| verbose | Mengontrol output logging. |
| class_weight | Untuk menyeimbangkan bobot kelas. |

Fitur-fitur tersebut disesuaikan dengan dataset yang digunakan dalam pendekatan *Bag of Words* untuk mengubah representasi teks menjadi vektor dengan menghitung frekuensi setiap kata yang muncul dalam teks. Parameter *Count Vectorizer* mengubah koleksi teks menjadi matriks yang menunjukkan jumlah kata-kata berbeda muncul dalam setiap teks ini berperan penting dalam meningkatkan performa model.

### *C. Split Data into Training and Testing Sets*

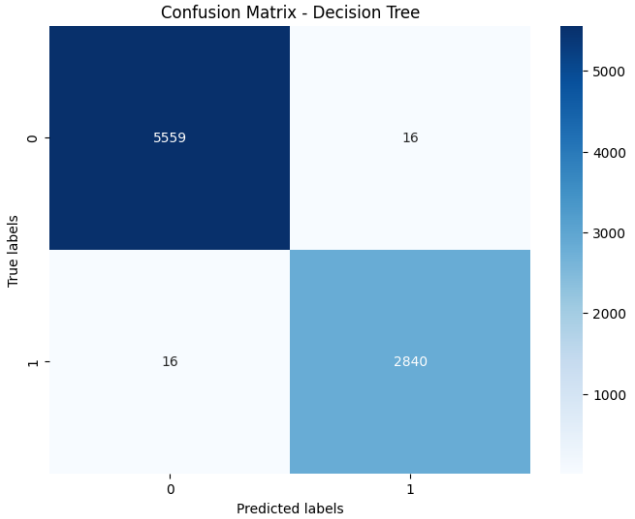
Pada proses *training* dan *testing* digunakan untuk membagi data menjadi dua kelompok terpisah, yaitu data pelatihan dan data pengujian yang ada pada Gambar 10. Fungsi *library* scikit-learn train_test_split membagi data dengan rasio 75:25 menjadi dua subset: subset pelatihan dan subset pengujian. Variabel x, y mengandung data fitur (x) dan label atau target (y) yang ingin dibagi. Data fitur (x) adalah data input yang digunakan untuk memprediksi label atau target (y). Parameter train_size = 0.75 menentukan bahwa 75% dari data akan digunakan untuk pelatihan. Parameter test_size = 0.25 menunjukkan bahwa 25% dari data akan digunakan untuk ujian. Biasanya, jika train_size sudah ditentukan, test_size akan secara otomatis menjadi sisa data (dalam hal ini, 1 - 0.75 = 0.25).

```
# Split the data into training and testing sets (75% in to training and 25% for testing)
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.75, test_size = 0.25)
```

Gambar 10. Train and Test

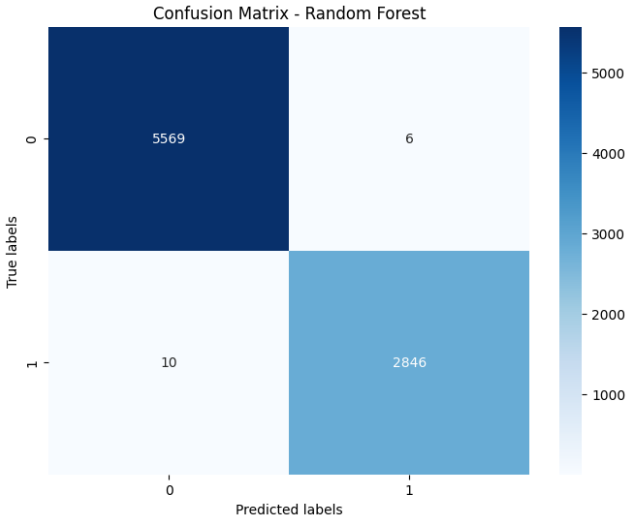
*D. Train Decision Tree and Random Forest Model*

Langkah selanjutnya melakukan import library digunakan untuk membuat model Decision Tree dan Random Forest. Lalu data *Train* dan *Test* untuk pelatihan serta pengujian dimuat dari file yang telah tersimpan dalam format sebelumnya. Label data pelatihan y_train dan pengujian y_test dimuat dari file pickle. Masuk ke dalam pembuatan model Decision Tree, yaitu "DecisionTreeClassifier" dengan melatih model dengan data pelatihan, melakukan prediksi pada dataset yang telah siap, menghitung akurasi prediksi dengan membandingkan hasil prediksi dengan label, dan yang terakhir menghasilkan akurasi model Decision Tree. Gambar 11 menunjukkan visualisasi confusion matrix yang digunakan. Confusion Matrix adalah representasi tabular yang memberikan gambaran umum tentang kinerja model klasifikasi dengan membandingkan label yang diantisipasi dengan benar. Ini menunjukkan berapa banyak prediksi model adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).




Gambar 11. Confusion Matrix Decision Tree

Setelah membuat model Decision Tree, selanjutnya membuat model Random Forest, yaitu "RandomForestClassifier". Kemudian melatih model dengan dataset yang telah tersedia, setelah dataset siap dilakukan pengujian dataset. Lalu menghitung akurasi prediksi dengan membandingkan hasil prediksi dengan label, langkah terakhir menentukan akurasi dari model Random Forest. Hasilnya ditunjukkan pada Gambar 12.




Gambar 12. Confusion Matrix Random Forest

Selanjutnya, Confusion Matrix Gambar 7 dan Gambar 8 merupakan evaluasi dari model Decision Tree dan Random Forest yang digunakan untuk klasifikasi biner. Matrix ini terdiri dari empat kuadran yang menunjukkan hasil prediksi dibandingkan dengan label sebenarnya. Pada sumbu horizontal terdapat label yang diprediksi oleh model, sementara sumbu vertikal menunjukkan label sebenarnya. Matrix ini menunjukkan bagaimana model algoritma Decision Tree dan Random Forest berjalan. Pada Gambar 7. matrix Decision Tree memperlihatkan bahwa dari 5575 sampel yang sebenarnya negatif (0), model berhasil mengidentifikasi 5559 sampel dengan benar sebagai negatif (*True Negative*) dan kurang tepat dalam mengidentifikasi 16 sampel sebagai positif (*False Positive*). Sebaliknya, dari 2856 sampel yang sebenarnya positif (1), model mengidentifikasi 2840 sampel dengan benar sebagai positif (*True Positive*) dan kurang tepat dalam mengidentifikasi 16 sampel sebagai negatif (*False Negative*). Perhitungan dilakukan untuk menentukan nilai akurasi, presisi, *recall*, dan F1-*score*.

Akurasi merupakan total prediksi benar, baik bernilai positif maupun negatif. Untuk melihat cara perhitungan nilai Akurasi, bisa merujuk ke bagian bawah.

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{2840+5559}{2840+5559+16+16} = \frac{8399}{8399} = 0.997$$

atau 99.70%

Presisi merupakan prediksi positif yang benar-benar positif, untuk melihat cara perhitungan nilai Presisi, bisa merujuk ke bagian bawah.

$$\text{Presisi} = \frac{TP}{TP+FP} = 0.9944 \text{ atau } 99.44\%$$

*Recall* merupakan *instance* positif yang benar terprediksi positif, untuk melihat cara perhitungan nilai *recall*, bisa merujuk ke bagian bawah.

*Recall* = $\frac{TP}{TP+FN}$ = $\frac{2840}{2840+16}$ = 0.9944 atau 99.44%

F1-*score* merupakan nilai rata-rata dari Presisi dan *Recall*, untuk melihat cara perhitungan nilai F1-*score* , bisa merujuk ke bagian bawah.

F1-*score* = 2 x $\frac{Presisi\ x\ Recall}{Presisi+Recall}$ = 2 x $\frac{0.9944\ x\ 0.9944}{0.9944+0.9944}$ = 0.9944 atau 99.44%

Pada Gambar 8. matrix Random Forest ini memperlihatkan bahwa dari 5575 sampel yang sebenarnya negatif (0), model berhasil mengidentifikasi 5569 sampel dengan benar sebagai negatif (*True Negative*) dan kurang tepat dalam mengidentifikasi 6 sampel sebagai positif (*False Positive*). Sebaliknya, dari 2856 sampel yang sebenarnya positif (1), model mengidentifikasi 2846 sampel dengan benar sebagai positif (*True Positive*) dan kurang tepat dalam mengidentifikasi 10 sampel sebagai negatif (*False Negative*). Perhitungan menggunakan rumus untuk menentukan akurasi, presisi, *recall*, dan F1-*score*.

Akurasi merupakan total prediksi benar, baik bernilai positif maupun negatif. Untuk melihat cara perhitungan nilai Akurasi, bisa merujuk ke bagian bawah.

Akurasi = $\frac{TP+TN}{TP+TN+FP+FN}$ = $\frac{2840+5559}{2840+5559+16+16}$ = $\frac{8399}{8399}$ = 0.997 atau 99.70%

Presisi merupakan prediksi positif yang benar-benar positif, untuk melihat cara perhitungan nilai Presisi, bisa merujuk ke bagian bawah.

Presisi = $\frac{TP}{TP+FP}$ = $\frac{2840}{2840+16}$ = 0.9944 atau 99.44%

*Recall* merupakan *instance* positif yang benar terprediksi positif, untuk melihat cara perhitungan nilai *recall*, bisa merujuk ke bagian bawah.

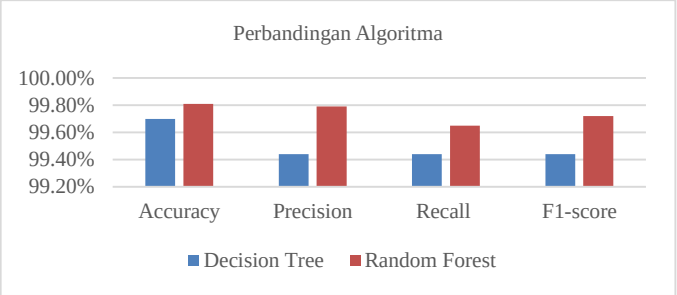
*Recall* = $\frac{TP}{TP+FN}$ = $\frac{2840}{2840+16}$ = 0.9944 atau 99.44%

F1-*score* merupakan nilai rata-rata dari Presisi dan *Recall*, untuk melihat cara perhitungan nilai F1-*score* , bisa merujuk ke bagian bawah.

F1-*score* = 2 x $\frac{Presisi\ x\ Recall}{Presisi+Recall}$ = 2 x $\frac{0.9944\ x\ 0.9944}{0.9944+0.9944}$ = 0.9944 atau 99.44%

### *E. Evaluate Model on Testing Set*

Evaluasi performa berdasarkan perhitungan yang dilakukan menggunakan algoritma Decision Tree dan Random Forest pada bagian *Train*, penulis menyiapkan rumus kemudian menghitung rata-rata nilai. Hasil perhitungan tersebut selanjutnya disajikan dalam bentuk gambar perbandingan, seperti yang ditampilkan Gambar 13.




Gambar 13. Perbandingan Algoritma

## IV. KESIMPULAN

Studi ini menggunakan algoritma machine learning, yaitu algoritma Decision Tree dan Random Forest dengan rasio perbandingan 75:25 lalu rangkaian tahapan penelitian mencakup *Collect Data*, *Preprocessing*, *Train and Test*, *Build and Compile*, dan yang terakhir Evaluasi. Setelah melalui tahapan-tahapan penelitian tersebut, hasilnya menunjukkan bahwa algoritma Decision Tree memperoleh accuracy 99.70%, precision 99.44%, *recall* 99.44%, dan F1-*score* 99.44%. Sementara itu, algoritma Random Forest menunjukkan bahwa perolehan accuracy 99.81%, precision 99.79%, *recall* 99.65, dan F1-*score* rata-rata 99.72%. Hasilnya menunjukkan bahwa algoritma Random Forest sedikit lebih baik dalam mengklasifikasikan kedua kelas dengan kesalahan yang sangat kecil. Terdapat perbedaan yang cukup signifikan antara nilai FP dan FN antara kedua algoritma. Nilai FP dan FN algoritma Decision Tree masing-masing sebesar 16, sedangkan algoritma Random Forest menghasilkan nilai prediksi FP sebesar 10 dan nilai FN sebesar 6.

Meskipun nilai FP dan FN yang sama adalah indikasi yang baik, namun penting untuk melakukan evaluasi lebih lanjut dengan berbagai metrik dan melakukan analisis tambahan untuk memastikan performa model sesuai dengan tujuan dan kebutuhan aplikasi. Penerapan praktis dari perbandingan performa algoritma Random Forest sesuai untuk digunakan dalam sistem deteksi serangan yang membutuhkan akurasi tinggi dan dapat menangani berbagai jenis data dengan baik. Meskipun algoritma Decision Tree kurang akurat dibandingkan Random Forest, algoritma ini dapat digunakan dalam situasi kecepatan pelatihan dan dapat berguna dalam lingkungan industri yang memerlukan penjelasan secara transparan tentang bagaimana keputusan dibuat, seperti saat melakukan audit keamanan.

**UCAPAN TERIMA KASIH**

Terima kasih penulis ucapkan kepada Program Studi Teknik Komputer Universitas Amikom Yogyakarta, Dosen Pembimbing yang terlibat, serta dukungan dari orang tua, dan teman-teman sekalian yang membantu dalam menyelesaikan penelitian ini.

## DAFTAR PUSTAKA

[1] O. Alotaibi and E. Pardede, "Transformation of schema from relational database (RDB) to NoSQL databases," *Data (Basel)*, vol. 4, no. 4, Dec. 2019, doi: 10.3390/data4040148.

[2] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Győrödi, G. A. Gabor, and G. D. Pecherle, "Performance analysis of nosql and relational databases with couchdb and mysql for application's data storage," *Applied Sciences (Switzerland)*, vol. 10, no. 23, pp. 1–21, Dec. 2020, doi: 10.3390/app10238524.

[3] J. Zhang, S. Hu, Z. Shi, and S. Han, "A Learning Sentiment Database for Machine Learning," in *Journal of Physics: Conference Series*, Institute of Physics, 2023. doi: 10.1088/1742-6596/2504/1/012030.

[4] F. Abdelhedi, R. Jemmali, and G. Zurfluh, "Relational Databases Ingestion into a NoSQL Data Warehouse."

[5] G. J. J. van den Burg, A. Nazábal, and C. Sutton, "Wrangling messy CSV files by detecting row and type patterns," *Data Min Knowl Discov*, vol. 33, no. 6, pp. 1799–1820, Nov. 2019, doi: 10.1007/s10618-019-00646-y.

[6] K. Sidharta and T. Wibowo, "Studi Efisiensi Sumber Daya Terhadap Efektivitas Penggunaan Database : Studi Kasus Sql Server Dan Mysql." [Online]. Available: http://journal.uib.ac.id/index.php/cbssit

[7] H. Jurnal *et al.*, "Jurnal Informatika Dan Teknologi Komputer Analisa Perbandingan Kinerja Response Time Query Mysql Dan Mongodb," *Juli*, vol. 2, no. 2, pp. 158–166, 2022.

[8] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review," *Journal of Cybersecurity and Privacy*, vol. 2, no. 4, pp. 764–777, Dec. 2022, doi: 10.3390/jcp2040039.

[9] O. Cheikhrouhou, H. Hamam, A. Mahfoudhi, and I. Jemal, "SQL Injection Attack Detection and Prevention Techniques Using Machine Learning," 2020. [Online]. Available: http://www.ripublication.com

[10] Institute of Electrical and Electronics Engineers, *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA).*

[11] Y. Abdulmalik, "An Improved SQL Injection Attack Detection Model Using Machine Learning Techniques," *International Journal of Innovative Computing*, vol. 11, no. 1, pp. 53–57, Apr. 2021, doi: 10.11113/ijic.v11n1.300.

[12] B. A. Pham and V. H. Subburaj, "An Experimental setup for Detecting SQLi Attacks using Machine Learning Algorithms," 2020.

[13] F. G. Deriba, A. O. Salau, S. H. Mohammed, T. M. Kassa, and W. B. Demilie, "Development of a Compressive Framework Using Machine Learning Approaches for SQL Injection Attacks," *Przeglad Elektrotechniczny*, vol. 98, no. 7, pp. 181–187, 2022, doi: 10.15199/48.2022.07.30.

[14] Abhishek, A. Dhankar, and N. Gupta, "A systematic review of techniques, tools and applications of machine learning," in *Proceedings of the 3rd International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, ICICV 2021*, Institute of Electrical and Electronics Engineers Inc., Feb. 2021, pp. 764–768. doi: 10.1109/ICICV50876.2021.9388637.

[15] M. Bagas, A. Darmawan, F. Dewanta, and S. Astuti, "Analisis Perbandingan Algoritma Decision Tree, Random Forest, dan Naïve Bayes untuk Prediksi Banjir di Desa Dayeuhkolot Comparative Analysis of Decision Tree, Random Forest, and Naïve Bayes Algorithm for Flood Prediction at Dayeuhkolot Village," *TELKA*, vol. 9, no. 1, pp. 52–61, 2023.

[16] A. Irma Purnamasari and I. Ali, "Perbandingan Tingkat Akurasi Algoritma Decision Tree Dan Random Forest Dalam Mengklasifikasi Penerima Bantuan Sosial Bpnt Di Desa Slangit," 2024.

[17] K. K. Dutta, S. A. Sunny, A. Victor, A. G. Nathu, M. Ayman Habib, and D. Parashar, "Kannada alphabets recognition using decision tree and random forest models," in *Proceedings of the 3rd International Conference on Intelligent Sustainable Systems, ICISS 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 534–541. doi: 10.1109/ICISS49785.2020.9315972.

[18] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.

[19] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020, doi: 10.1109/ACCESS.2020.2973219.

[20] M. Bansal, A. Goyal, and A. Choudhary, "A comparative analysis of K-Nearest Neighbor, Genetic, Support Vector Machine, Decision Tree, and Long Short Term Memory algorithms in machine learning," *Decision Analytics Journal*, vol. 3, p. 100071, Jun. 2022, doi: 10.1016/j.dajour.2022.100071.

[21] J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, "A comparison of random forest variable selection methods for classification prediction modeling," *Expert Systems with Applications*, vol. 134. Elsevier Ltd, pp. 93–101, Nov. 15, 2019. doi: 10.1016/j.eswa.2019.05.028.

[22] K. Shah, H. Patel, D. Sanghvi, and M. Shah, "A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification," *Augmented Human Research*, vol. 5, no. 1, Dec. 2020, doi: 10.1007/s41133-020-00032-0.

[23] S. Tufail, H. Riggs, M. Tariq, and A. I. Sarwat, "Advancements and Challenges in Machine Learning: A Comprehensive Review of Models, Libraries, Applications, and Algorithms," *Electronics (Switzerland)*, vol. 12, no. 8. MDPI, Apr. 01, 2023. doi: 10.3390/electronics12081789.

[24] S. Sza Amulya Larasati, E. Nuraida Kusuma Dewi, B. Hanif Farhansyah, F. Abdurrachman Bachtiar, F. Pradana, and U. Brawijaya, "Penerapan Decision Tree Dan Random Forest Dalam Deteksi Tingkat Stres Manusia Berdasarkan Kondisi Tidur," vol. 10, no. 7, pp. 1503–1510, 2023, doi: 10.25126/jtiik.2023107993.