# Klasifikasi Data Penerimaan Calon Mahasiswa Baru Menggunakan Algoritma Random Forest

Kafka Febian[#1], Avinash[#2], Sherly Santiadi[#3], Devion Tanrico[#4],

[#]*Magister Ilmu Komputer, Universitas Kristen Maranatha*
*Jl. Prof. Drg. Surya Sumantri, MPH No.65, Bandung*

[1]mi2279806@student.it.maranatha.edu

[2]mi2279005@student.it.maranatha.edu

[3]mi2279801@student.it.maranatha.edu

[4]mi2279802@student.it.maranatha.edu

*Abstract* — **The administration of new student admissions is always increasing each year. Therefore, Maranatha Christian University is trying to build a system that can help classify these administrative files. It is very important to create a model that can show the tendency of these prospective students to be accepted in the chosen department as material for consideration in the next selection process. In this study, the Random Forest algorithm was applied to address this issue. The evaluation metrics of forecasting models used include Classification Report, Receiver Operation Characteristic Curve, and Confusion Matrix. In addition, an agile methodology was applied in developing two outputs: a website and an application that can be integrated with the prediction model for the classification of new student admissions using the Random Forest algorithm.**

*Keywords* — **RandomForest; Classification; Application**

## I. PENDAHULUAN

### A. *Latar Belakang*

Setiap tahun, universitas di seluruh dunia menerima ribuan aplikasi dari calon mahasiswa baru. Proses penerimaan mahasiswa baru adalah proses yang sangat penting dalam kehidupan universitas karena ini menentukan komposisi mahasiswa dan berdampak pada prestasi akademis universitas. Dikarenakan kesadaran masyarakat akan pentingnya pendidikan di Indonesia, maka para pendaftar pun seringkali bersaing secara ketat agar dapat lolos dalam tahap seleksi calon mahasiswa baru. Oleh karena prosesnya yang rumit dan besarnya peluang terjadi subjektifitas sering kali terjadi dalam tahap seleksi calon penerimaan mahasiswa baru[1].
Dengan demikian, hal ini akan menimbulkan rasa ketidakpuasan terhadap seleksi calon penerimaan mahasiswa baru. Oleh karena itu, topik yang akan diangkat dalam penelitian ini adalah teknik klasifikasi data calon penerimaan mahasiswa baru menggunakan algoritma *Random Forest*. Data penerimaan mahasiswa baru meliputi informasi tentang calon mahasiswa seperti nilai ujian, riwayat pendidikan, dan informasi pribadi. Data ini sangat penting untuk menentukan apakah calon mahasiswa akan diterima atau tidak dalam jurusan yang dipilih.

### B. *Rumusan Masalah*

Berikut merupakan dua hal yang menjadi fokus permasalahan dalam penelitian ini:

1. Bagaimana hasil visualisasi dataset yang digunakan?
2. Bagaimana cara mengimplementasikan algoritma *Random Forest* dalam menyelesaikan permasalahan klasifikasi data penerimaan calon mahasiswa baru?
3. Bagaimana hasil pengukuran kinerja algoritma *Random Forest* terhadap permasalahan klasifikasi data penerimaan calon mahasiswa baru?

### C. *Tujuan*

Adapun tujuan pembangunan sistem penerimaan calon mahasiswa baru dapat dirumuskan sebagai berikut:

1. Mendapatkan hasil visualisasi dataset.
2. Membuat aplikasi *mobile* dan *website* untuk sistem penerimaan calon mahasiswa baru dilengkapi dengan probabilitas mahasiswa tersebut diterima dalam jurusan yang dipilih.
3. Mengimplementasikan algoritma *Random Forest* dalam prediksi penerimaan calon mahasiswa baru.

## II. KAJIAN PUSTAKA

### A. Random Forest

Algoritma *Random Forest* merupakan sebuah algoritma yang cukup populer dalam memprediksi baik secara regresi maupun klasifikasi. *Random Forest* sendiri merupakan gabungan dari beberapa pohon klasifikasi maupun regresi yang menggunakan model sederhana untuk melakukan pemisahan variabel untuk menentukan hasil prediksi. Pohon tersebut biasanya disebut dengan *Decision Tree*. Berbeda dengan *Random Forest* yang bisanya jauh lebih kompleks, *Decision Tree* lebih mudah untuk di implementasi dan diinterpretasikan hasil pemisahan variabelnya. Namun, dibalik kompleksitasnya *Random Forest* menawarkan akurasi yang lebih baik dibandingkan dengan model *single decision tree*. Salah satu keuntungan menggunakan *random forest* adalah *random forest* dapat mengatasi dataset yang memiliki banyak *predictor variable*.

Algoritma *Random Forest* merupakan sebuah algoritma yang cukup populer dalam memprediksi baik secara regresi maupun klasifikasi. *Random Forest* sendiri merupakan gabungan dari beberapa pohon klasifikasi maupun regresi yang menggunakan model sederhana untuk melakukan pemisahan variabel untuk menentukan hasil prediksi. Pohon tersebut biasanya disebut dengan *Decision Tree*. Berbeda dengan *Random Forest* yang bisanya jauh lebih kompleks, *Decision Tree* lebih mudah untuk di implementasi dan diinterpretasikan hasil pemisahan variabelnya. Namun, dibalik kompleksitasnya *Random Forest* menawarkan akurasi yang lebih baik dibandingkan dengan model *single decision tree*. Salah satu keuntungan menggunakan *random forest* adalah *random forest* dapat mengatasi dataset yang memiliki banyak *predictor variable* [2].

Pendekatan yang dilakukan di dalam *Random Forest* menggunakan pendekatan *bagging* (*bootstrap aggregating*). Metode ini digunakan untuk mengurangi variansi di dalam sebuah model dengan cara menggabungkan beberapa model yang sudah dilatih pada *subset* yang berbeda pada data *training*. Hal ini dilakukan untuk mengurangi *overfitting* dan meningkatkan akurasi serta *robustness* di dalam sebuah model [3]. Dengan beberapa kelebihan *random forest* maka algoritma ini dapat diimplementasikan dalam mengklasifikasi calon mahasiswa baru dalam menentukan probabilitas kecocokan terhadap jurusan yang dipilih.

### B. REST API

REST API adalah singkatan dari Representational State Transfer Application Programming Interface, yaitu sebuah antarmuka pemrograman aplikasi (API) yang sesuai dengan prinsip-prinsip desain arsitektur *REST*. *REST* adalah gaya arsitektur untuk sistem hipermedia terdistribusi yang pertama kali diperkenalkan oleh ilmuwan komputer Roy Fielding pada tahun 2000 dalam disertasinya. *REST* memiliki enam prinsip atau batasan utama yang harus dipenuhi oleh sebuah API agar dapat disebut sebagai *RESTful*, yaitu: *uniform interface, client-server decoupling, statelessness, cacheability, layered system architecture, dan code on demand* (opsional)**Error! Reference source not found.**.

*REST API* terdiri dari sekumpulan sumber daya yang saling terhubung. Kumpulan sumber daya ini disebut sebagai model sumber daya dari *REST API*. Setiap sumber daya dalam *REST API* diidentifikasi secara unik oleh sebuah resource identifier, yang biasanya berupa uniform resource identifier (*URI*). Sumber daya dapat memiliki berbagai representasi dalam format data yang berbeda, seperti *JSON*, *XML*, *HTML*, atau lainnya. Representasi sumber daya harus menyertakan informasi yang cukup untuk mendeskripsikan bagaimana pesan dapat diproses dan apa saja aksi-aksi tambahan yang dapat dilakukan oleh klien terhadap sumber daya tersebut. Selain itu, representasi sumber daya juga harus mengandung hypermedia, yaitu tautan-tautan yang mengarah ke sumber daya lain yang relevan [4].

*REST API* menggunakan protokol transfer *hypertext* (*HTTP*) sebagai mekanisme komunikasi antara klien dan *server*. Klien dapat mengirimkan permintaan (*request*) ke *server* dengan menggunakan metode *HTTP* yang sesuai dengan operasi yang ingin dilakukan terhadap sumber daya, seperti *GET* untuk mengambil data, *POST* untuk membuat data baru, *PUT* untuk mengubah data yang ada, atau *DELETE* untuk menghapus data. *Server* kemudian akan merespons permintaan tersebut dengan mengirimkan representasi sumber daya yang diminta atau status kode yang menunjukkan hasil dari permintaan tersebut. Permintaan dan respons harus bersifat *stateless*, artinya tidak bergantung pada konteks atau sesi sebelumnya. Jika respons bersifat *cacheable*, maka klien dapat menyimpan data respons untuk digunakan kembali pada permintaan yang sama di kemudian hari. Selain itu, komunikasi antara klien dan server dapat melalui beberapa lapisan sistem yang berbeda **Error! Reference source not found.**.

*REST API* memiliki beberapa kelebihan dibandingkan dengan jenis *API* lainnya, seperti *SOAP* atau *XML-RPC*. Beberapa kelebihan tersebut antara lain adalah: fleksibilitas, karena *REST* API dapat dikembangkan dengan menggunakan hampir semua bahasa pemrograman dan mendukung berbagai format data; kesederhanaan, karena *REST API* hanya membutuhkan *URI* sebagai identitas sumber daya dan *HTTP* sebagai protokol komunikasi, karena *REST API* memanfaatkan mekanisme *caching* untuk meningkatkan performa di sisi klien dan skalabilitas di sisi server, karena *REST API* memisahkan antara antarmuka pengguna (klien) dan penyimpanan data (*server*), sehingga memudahkan migrasi antar-platform.

### C. React Native

*React Native* merupakan Framework Open Source dari javascript yang digunakan untuk membangun Mobile Application secara *cross-platform*. Dengan *React Native*, developer dapat membuat aplikasi yang dapat dijalankan di platform iOS dan Android hanya dengan satu kode sumber. React native ini dibuat berdasarkan React, tetapi tidak mengacu kepada browser, melainkan ke platform mobile.

*React native* sendiri ditulis dengan campuran *JavaScript* dan markup *XML* esque, yang dikenal sebagai *JSX* yang Kemudian React native ini yang menjembatani native rendering *APIs* pada *Objective-C* (untuk *IOS*) dan Java (untuk *Android*) Dengan demikian, aplikasi yang telah dibuat atau dituliskan akan di render menggunakan komponen *UI* aslinya dan bukan sebuah tampilan web sehingga akan terlihat dan terasa seperti aplikasi mobile lainnya.

Di dalam *React Native* ada beberapa komponen utama yang hampir selalu digunakan untuk membangun sebuah aplikasi berbasis *mobile* yaitu:

1. Komponen: *React Native* menyediakan kumpulan komponen yang dapat digunakan untuk membangun tampilan *UI* aplikasi. Komponen ini dapat digunakan untuk membuat tampilan seperti tombol, teks, gambar, input, dan lain-lain,beberapa contoh komponen yaitu *View ,Text, Touchableopacity, Picker, Input* .
2. *Props: Props (Properties)* adalah sebuah argumen yang digunakan untuk mengkonfigurasi komponen. *Props* dapat digunakan untuk menentukan tampilan dan perilaku komponen.
3. *State: State* adalah objek *JavaScript* yang digunakan untuk menyimpan data yang dapat berubah nilainya di dalam komponen. Ketika state berubah, *React Native* akan secara otomatis memperbarui tampilan *UI*.
4. *Style: Style* berfungsi untuk mengatur atau memperindah tampilan komponen. React Native dengan menggunakan *StyleSheet* untuk mengatur style. *StyleSheet* menyediakan banyak properti style seperti warna, ukuran, *margin*, *padding*, dan lain-lain.

*D. MySQL*

*MySQL* adalah sistem manajemen basis data relasional (*RDBMS*) yang bersumber terbuka dan menggunakan bahasa *SQL* (*Structured Query Language*). *SQL* adalah bahasa yang digunakan untuk menambah, mengakses, dan mengelola konten basis data. Bahasa ini dikenal karena keandalan, pemrosesan cepat, fleksibilitas, dan kemudahan penggunaannya.

*MySQL* dikembangkan oleh *Oracle Corporation* dan merupakan salah satu *RDBMS* paling populer di dunia. *MySQL* dapat digunakan untuk berbagai aplikasi, seperti web, cloud, komunikasi, *fintech*, kesehatan, dan lainnya. *MySQL* juga mendukung berbagai fitur canggih, seperti replikasi, klastering, partisi, penyimpanan *JSON*, mesin *InnoDB*, dan lainnya [6].

*MySQL* juga memiliki beberapa kelebihan yang membuatnya layak dipertimbangkan untuk dipilih sebagai sistem database, antara lain: *MySQL* mendukung integrasi dengan bahasa pemrograman lain seperti *R, Python, PHP, Java*, dan lainnya. Hal ini memudahkan pengembang untuk membuat aplikasi yang berbasis database dengan berbagai fungsi dan fitur.

*MySQL* memiliki keamanan transaksi yang terjamin dengan fitur seperti enkripsi, autentikasi, dan otorisasi. *MySQL* juga mendukung fitur *ACID* (*Atomicity, Consistency, Isolation, Durability*) yang menjamin integritas data dalam setiap transaksi. *MySQL* memiliki ketersediaan yang handal dengan fitur seperti replikasi dan klastering. Replikasi memungkinkan pengguna untuk membuat salinan database di server lain untuk tujuan backup atau load balancing. Klastering memungkinkan pengguna untuk menggabungkan beberapa server menjadi satu unit logis yang dapat meningkatkan kinerja dan skalabilitas [8].

*E. Metodologi Agile*

*Agile* adalah pendekatan pengembangan perangkat lunak yang fleksibel, adaptif, dan berbasis tim. *Agile* menekankan pada kolaborasi antara pengembang, pengguna, dan pemangku kepentingan untuk memastikan pengembangan produk yang berkualitas [9].

Metodologi *agile* didasarkan pada "*Agile Manifesto*", sebuah dokumen yang ditandatangani pada tahun 2001 oleh sekelompok ahli perangkat lunak. Manifesto menekankan nilai-nilai seperti orang dan interaksi, perangkat lunak fungsional, dan daya tanggap terhadap perubahan [10].

Metodologi *agile* memungkinkan tim pengembangan untuk dengan cepat merespons perubahan permintaan konsumen dengan tetap menjaga kualitas produk dan agar konsumen bisa puas akan produk yang mereka inginkan. Pendekatan ini menggabungkan siklus pengembangan singkat dengan pengujian berkelanjutan dan kolaborasi antara pengembang dan pemangku kepentingan [11].

Metodologi *agile* berpacu pada kolaborasi antar anggota tim, keterlibatan konsumen dan fleksibilitas dalam menghadapi berbagai macam kebutuhan pelanggan, komunikasi yang baik sangat penting dalam proses pengembangan Perangkat lunak agile sehingga mampu menghasilkan produk yang lebih berkualitas [12].

Dari beberapa pernyataan di atas dapat kita simpulkan bahwa metodologi *agile* merupakan sebuah pendekatan pengembang perangkat lunak yang fleksibel, adaptif, dan berbasis tim yang menekankan pada kolaborasi pengembang, pengguna, dan pemangku kepentingan untuk memastikan pengembang produk yang berkualitas. Metodologi ini berfokus pada pertumbuhan inkremental, peningkatan berkelanjutan, dan respon cepat terhadap perubahan permintaan pengguna.

## III. PERANCANGAN

### A. *Sumber Data*

Sumber data untuk penelitian ini adalah dataset dari Penerimaan Mahasiswa Baru (PMB) tahun 2021 pada Universitas Kristen Maranatha. Data yang digunakan meliputi nilai rapor siswa, jurusan yang dipilih saat mendaftar, dan status diterima atau ditolak. Dataset ini diperoleh dari Universitas Kristen Maranatha dengan izin dan persetujuan dari pihak yang berwenang. Data tersebut kemudian diproses *(feature engineering*) guna memastikan data yang digunakan valid. Selanjutnya, data tersebut digunakan untuk mengklasifikasi calon penerimaan mahasiswa baru di Universitas Kristen Maranatha.

### B. *Diagram Perancangan Basis Data*

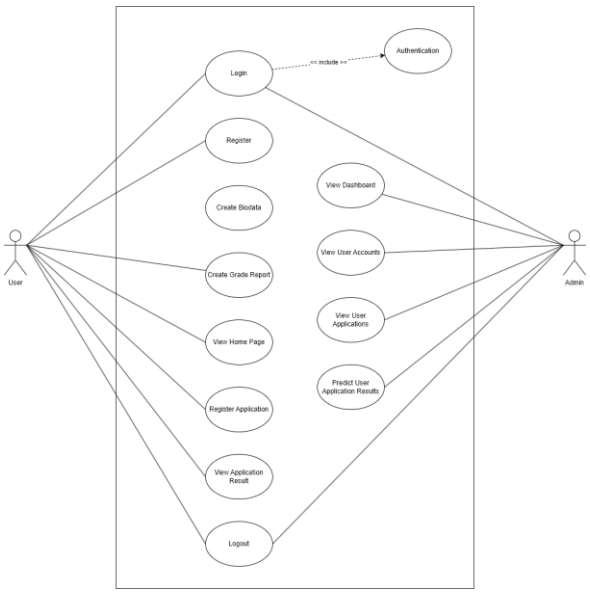
Penggunaan *MySQL Workbench* sebagai basis data dalam perancangan ini sangat penting. *MySQL Workbench* memungkinkan para pengguna untuk dengan mudah membuat tampilan tabel-tabel menjadi sebuah diagram yang sangat mempermudah proses design. Fitur-fitur canggih yang dimiliki oleh *MySQL Workbench* menjadikan proses desain lebih efisien dan produktif, sehingga dapat menghemat waktu dan usaha dalam menghasilkan sebuah basis data yang berkualitas tinggi seperti pada gambar 3.1 di bawah ini.

Gambar 3. 1 Penggunaan My SQL Workbench

Terdapat tujuh tabel yang menjadi bagian dari basis data yang digunakan. Tabel-tabel tersebut adalah *'users', 'user_details', 'applications', 'subjects', 'grades', 'faculties', dan 'study_programs'*. Masing-masing tabel memiliki fungsi dan relasi yang berbeda-beda satu sama lain, dan terhubung secara langsung maupun tidak langsung untuk membangun sebuah sistem yang terintegrasi dengan baik.

Dengan adanya tabel *'users'* dan *'user_detail'*, informasi mengenai para pengguna dan detail akun mereka dapat diakses dan dikelola dengan mudah. Sementara itu, tabel *'applications'* menjadi sangat penting dalam proses penerimaan mahasiswa baru, karena informasi mengenai aplikasi dan proses seleksi mahasiswa dapat ditangani secara efisien.

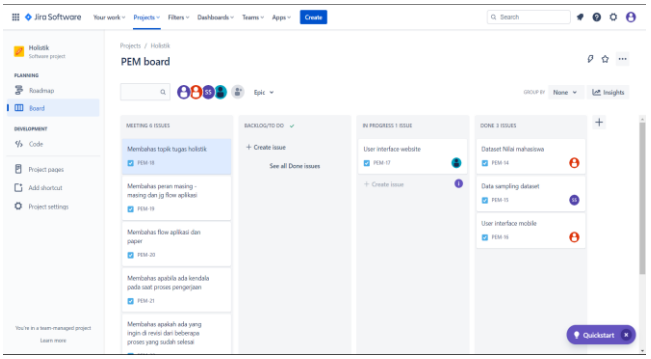
Tabel *'subjects'* dan *'grades'* berkaitan dengan informasi akademik, dimana informasi mengenai mata kuliah dan hasil belajar mahasiswa dapat tersimpan dengan rapi dan mudah diakses. Sementara itu, tabel *'faculties'* dan *'study_programs'* memberikan informasi mengenai fakultas dan program studi yang tersedia, sehingga memudahkan proses pemilihan program studi bagi mahasiswa.



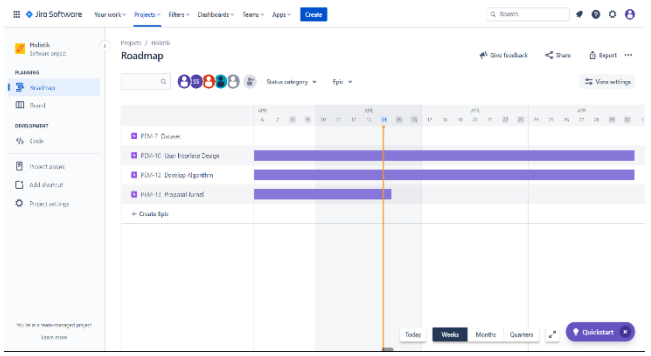

Gambar 3. 2 Use case diagram user dan admin

### C. *Agile Development*

Berikut ini merupakan agile development yang kami kerjakan dalam proyek ini menggunakan Jira Atlassian seperti pada gambar 3.3 dan gambar 3.4




Gambar 3. 3 Penggunaan jira sebagai agile development




Gambar 3. 4 Penggunaan jira sebagai agile development

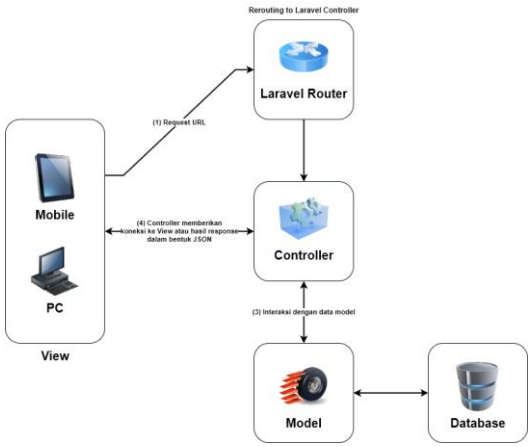
D. *Arsitektur Sistem*

Berikut merupakan arsitektur sistem yang diimplementasikan:

1. Algoritma *Random Forest*
2. Basis data *MySQL*
3. Generator basis data menggunakan *Python*.
4. Laravel sebagai Sistem Manajemen PMB untuk admin

Laravel adalah sebuah framework PHP yang populer dan kuat untuk membangun aplikasi web. Salah satu fitur yang paling berguna dari Laravel adalah kemampuannya untuk membuat dan mengelola RESTful API.

Laravel menyediakan beberapa fitur yang berguna dalam membuat RESTful API, seperti middleware, authentication, dan validation. Middleware dapat digunakan untuk memeriksa permintaan dan memberikan respon yang sesuai. Authentication digunakan untuk mengamankan API dengan autentikasi pengguna. Validation digunakan untuk memvalidasi data input sebelum data disimpan ke dalam database.REST (Representational State Transfer) adalah arsitektur software yang banyak digunakan dalam pembuatan API. RESTful API memungkinkan aplikasi untuk berkomunikasi dengan satu sama lain melalui HTTP protokol. RESTful API pada Laravel dapat digunakan untuk melakukan operasi CRUD (create, read, update, delete) pada data melalui permintaan HTTP seperti GET, POST, PUT, dan DELETE.




Gambar 3. 5 Arsitektur laravel

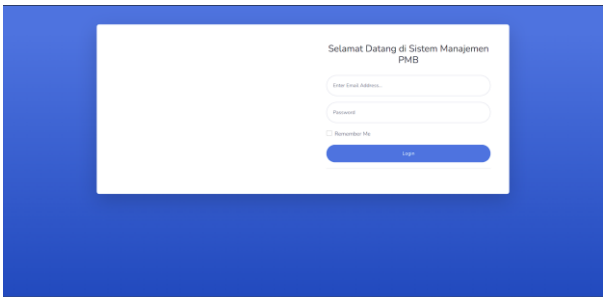
5. Aplikasi *React Native*.

## I. HASIL EKSPERIMEN DAN EVALUASI

### A. *Tampilan Website*

*1. Login*

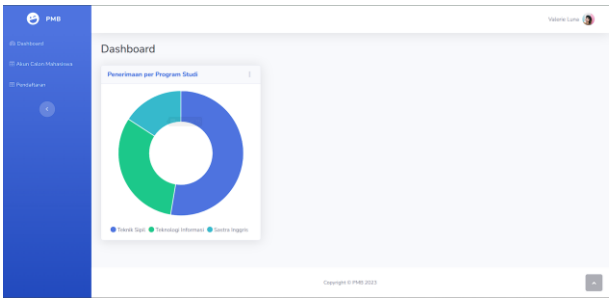
Tampilan login menampilkan halaman awal ketika situs pertama kali diakses seperti pada gambar 3.6. Pada halaman ini, sistem meminta user admin untuk login kedalam sistem.




Gambar 3. 6 Halaman login PMB

*2. Dashboard*

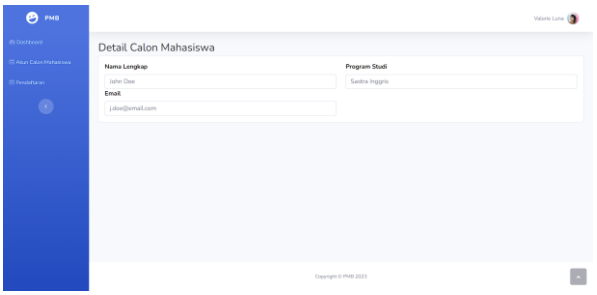
Tampilan dashboard menampilkan data-data statistik seperti pada gambar 3.7. Pada halaman ini, sistem akan menampilkan data statistik yang berguna untuk keperluan pendataan dan visualisasi data




Gambar 3. 7 Halaman dashboard PMB

*3. Detail Calon Mahasiswa*

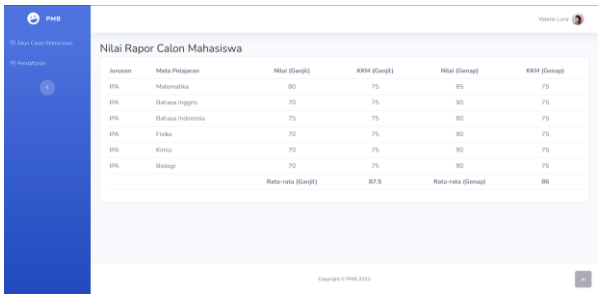
Tampilan detail calon mahasiswa menampilkan data akun dan detail calon mahasiswa seperti pada gambar 3.8 Pada halaman ini, sistem akan menampilkan data calon mahasiswa




Gambar 3. 8 Halaman detail calon mahasiswa

*4. Nilai Rapor Calon Mahasiswa*

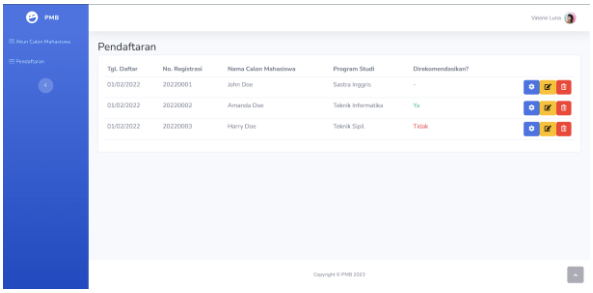
Tampilan nilai rapor calon mahasiswa menampilkan tabel nilai rapor calon mahasiswa seperti pada gambar 3.9. Pada halaman ini, sistem hanya menampilkan data nilainya saja




Gambar 3. 9 Halaman nilai rapor calon mahasiswa

*5. Pendaftaran*

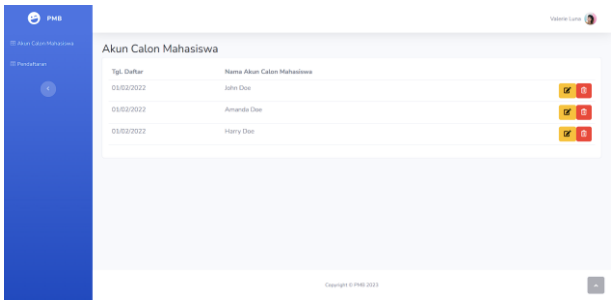
Tampilan pendaftaran menampilkan tabel pendaftaran calon mahasiswa seperti pada gambar 3.10 Pada halaman ini terdapat tombol prediksi saran, hapus dan edit.




Gambar 3. 10 Halaman Pendaftaran

*6. Akun Calon Mahasiswa*

Tampilan akun calon mahasiswa menampilkan tabel akun kredensial calon mahasiswa seperti pada gambar 3.11 Pada halaman ini hanya terdapat tombol hapus dan edit.



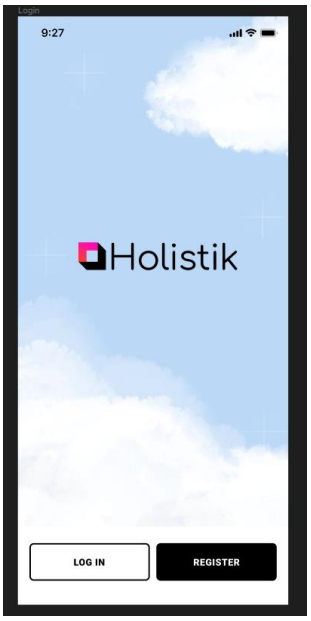

Gambar 3. 11 Halaman akun calon mahasiswa

## B. *Tampilan Aplikasi Android*

Berikut rancangan tampilan aplikasi berbasis *mobile*:

1. *Halaman Intro*

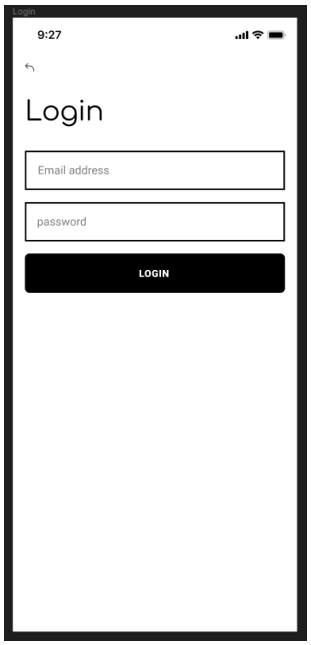
   Berikut rancangan tampilan halaman *intro* aplikasi mobile, pada Gambar 3. 12 terdapat 2 buah tombol yaitu *login* dan *register*, dan juga logo serta nama aplikasi.




Gambar 3. 12 Halaman Intro

2. *Halaman Login*

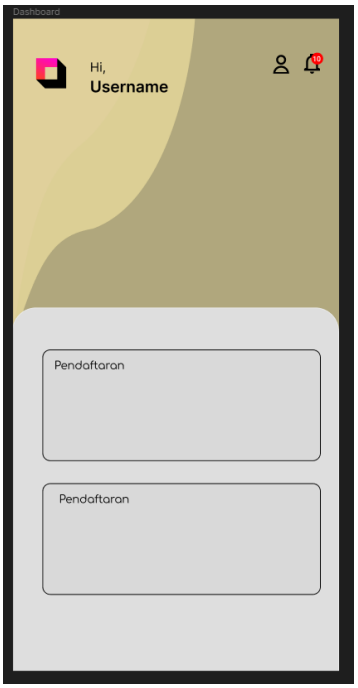
   Berikut rancangan tampilan halaman *login* aplikasi *mobile*, pada gambar 3.13 terdapat 2 buah *text field*/input. Pengguna diminta memasukan alamat *email* dan *password* yang sudah didaftarkan. Jika berhasil maka pengguna akan diarahkan ke halaman *dashboard* atau utama, jika tidak maka pengguna akan menerima *alert* atau *pop-up window* yang bersifat sebagai validasi.




Gambar 3. 13 Halaman login

3. *Halaman Dashboard*

   Berikut rancangan tampilan halaman *dashboard*. Di gambar 1.3 terlihat ada *username* dan sebuah *row* yang berisi *icon profile* dan notifikasi yang jika ditekan akan mengarahkan pengguna ke halaman *profile* dan halaman berisi daftar notifikasi, dan berisi daftar pendaftaran yang terbuka pada tahun ajaran yang berbeda.



Gambar 3. 14 Halaman dashboard

Berikut tampilan jika pengguna sudah mendaftar, maka akan muncul tambahan menu yang berisi nilai yang dimasukan, jurusan yang diambil dan status pengguna diterima atau ditolak.

Gambar 3. 15 Halaman ketika pengguna sudah mendaftar

## C. *Eksperimen*

### 1. *Pengumpulan Dataset*

Dataset yang digunakan adalah dataset nilai rapor calon mahasiswa angkatan 2022 dan pilihan program studinya serta status diterima atau tidaknya.

Dataset itu sendiri terdiri dari banyak kolom nilai dan KKM per mata pelajaran dan semesternya, serta pilihan program studi dan hasil keputusannya. Berikut ini contoh daripada tabelnya:

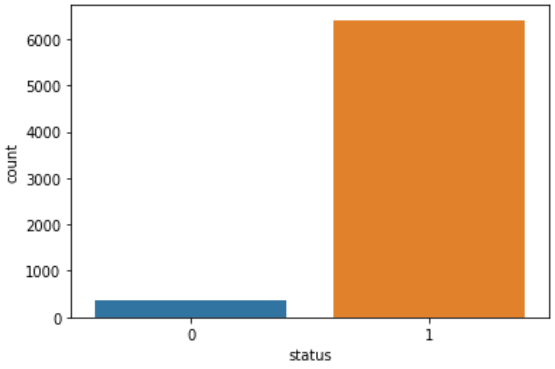
| Matematika Nilai Ganjil | Matematika KKM Ganjil | …. | Program Studi | Status |
|---|---|---|---|---|
| 80 | 75 | … | MI | 1 |
| 76 | 75 | … | IF | 0 |
| 75 | 75 | … | SI | 1 |

Tabel 1. 1 Dataset Calon Penerimaan Mahasiswa Baru

### 2. *Pembersihan Dataset*

Dataset yang mentah pastinya memiliki data yang error, artinya sistem tidak dapat menghitung, memproses, memanipulasi data tersebut dan tergantung jenis errornya seperti apa. Maka dari itu dilakukanlah pembersihan dataset. Dataset dibersihkan dengan cara menghapus nilai-nilai rapor dan KKM mata pelajaran yang bernilai 'NULL'. Nilai ini artinya data tersebut memang bisa dibilang kosong dan berpotensi bermasalah pada proses perhitungan dan analisa.

### 3. *Visualisasi Dataset*



Gambar 3. 16 Visualisasi Dataset

Dataset yang sudah berhasil dibersihkan selanjutnya akan divisualisasikan menggunakan *library Seaborn*. Dataset yang digunakan memiliki 6779 baris data serta 53 kolom dengan komposisi:

- 370 baris yang memiliki label bernilai 0 (jumlah mahasiswa tidak diterima di jurusan tersebut).
- 6409 baris data yang memiliki label bernilai 1 (jumlah mahasiswa diterima di jurusan tersebut).

$$\frac{370}{6679} \cdot 100\% = 5.46\% \text{ ditolak dari jurusan yang dipilih}$$

$$\frac{6409}{6679} \cdot 100\% = 94.54\% \text{ diterima dari jurusan yang dipilih}$$

Dari persentase tersebut, dapat dilihat bahwa dataset yang dimiliki tidak seimbang dan hal ini akan sangat berpengaruh pada hasil prediksi model.

4. *Mengimplementasikan One Hot Encoding*

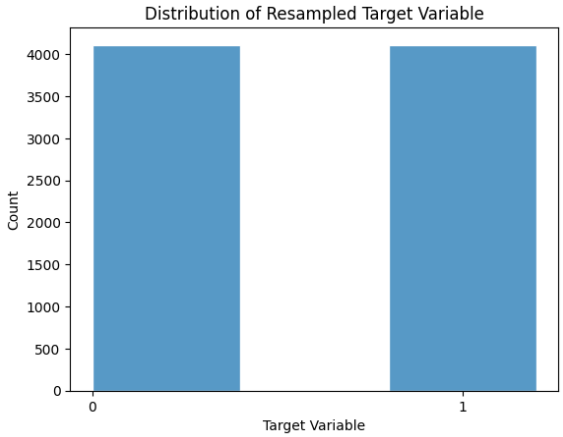
```
 31  kkm_bahasa_inggris_even      6779 non-null   int64
 32  grade_bahasa_indonesia_odd   6779 non-null   int64
 33  kkm_bahasa_indonesia_odd     6779 non-null   int64
 34  grade_bahasa_indonesia_even  6779 non-null   int64
 35  kkm_bahasa_indonesia_even    6779 non-null   int64
 36  grade_bahasa_mandarin_odd    6779 non-null   int64
 37  kkm_bahasa_mandarin_odd      6779 non-null   int64
 38  grade_bahasa_mandarin_even   6779 non-null   int64
 39  kkm_bahasa_mandarin_even     6779 non-null   int64
 40  grade_bahasa_jepang_odd      6779 non-null   int64
 41  kkm_bahasa_jepang_odd        6779 non-null   int64
 42  grade_bahasa_jepang_even     6779 non-null   int64
 43  kkm_bahasa_jepang_even       6779 non-null   int64
 44  grade_bahasa_korea_odd       6779 non-null   int64
 45  kkm_bahasa_korea_odd         6779 non-null   int64
 46  grade_bahasa_korea_even      6779 non-null   int64
 47  kkm_bahasa_korea_even        6779 non-null   int64
 48  grade_bahasa_jerman_odd      6779 non-null   int64
 49  kkm_bahasa_jerman_odd        6779 non-null   int64
 50  grade_bahasa_jerman_even     6779 non-null   int64
 51  kkm_bahasa_jerman_even       6779 non-null   int64
 52  name                         6779 non-null   object
 53  status                       6779 non-null   int64
dtypes: int64(53), object(1)
memory usage: 2.8+ MB
```

Gambar 3. 17 Implementasi One hot encoding

Pada salah satu kolom fitur yaitu ‘name’ memiliki tipe data *object*. Hal ini disebabkan di dalam kolom ‘name’ berisikan data-data jurusan yang dipilih oleh calon mahasiswa seperti: Psikologi, Kedokteran, Teknik Informatika, dan lain-lain. Kebanyakan algoritma *machine learning* akan bekerja lebih baik apabila data-data yang diolah berupa data numerik, oleh karena itu fitur ‘name’ ini akan diubah ke dalam data numerik menggunakan pendekatan *One hot encoding*. Pendekatan ini dipilih dikarenakan pada fitur ‘name’ tidak memiliki *ranking* oleh karena itu pendekatan *one hot encoding* akan lebih cocok dibandingkan dengan menggunakan pendekatan *ordinal encoder*.

5. *Oversampling Dataset*

Sebelum melakukan *oversampling* data tersebut telah dipisah ke menggunakan *train_test_split* dengan persentase *test_size* sebesar 20%. Pemisahan data ini penting dilakukan sebelum menerapkan *oversampling* agar tidak ada kebocoran data yaitu ketika model bisa melihat data uji. Untuk mengatasi ketidakseimbangan dataset maka dilakukan pengambilan *sample* dengan metode *SMOTE* (*Synthetic Minority Over-sampling Technique*).




Gambar 3. 18 Oversampling dataset

Setelah dilakukan *oversampling* data berlabel 0 berjumlah sama dengan data berlabel 1 yaitu dengan jumlah 4111.

6. *Model Random Forest dengan Random Search*

```
RandomForestClassifier
RandomForestClassifier(max_depth=20, n_estimators=200)
```

Gambar 3. 19 Model random forest dengan random search

Setelah selesai melakukan *oversampling*, maka dibangun sebuah model *random forest* pada awal eksperimen *hyperparameter* yang digunakan adalah *default hyperparameter*. Namun, pada penelitian ini, telah dilakukan *hyperparameter tuning* menggunakan *Randomized Search*. Hasil *hyperparameter* terbaik adalah dengan kedalaman *tree* 20 dan *n_estimators* 200.

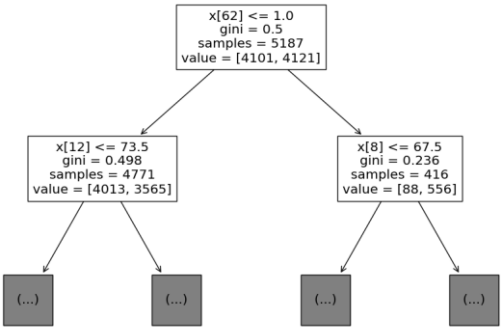
7. *Classification Report*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.09 | 0.08 | 0.09 | 88 |
| 1 | 0.94 | 0.95 | 0.94 | 1268 |
| accuracy | | | 0.89 | 1356 |
| macro avg | 0.51 | 0.51 | 0.51 | 1356 |
| weighted avg | 0.88 | 0.89 | 0.89 | 1356 |

Gambar 3. 20 classification report

Hasil klasifikasi terhadap data uji didapatkan tingkat akurasi sebesar 89% serta pada data latih sebesar 95%. Namun, hal ini belum dapat menjadi metrik uji yang baik mengingat dataset yang dimiliki sebelumnya adalah dataset yang tidak seimbang.

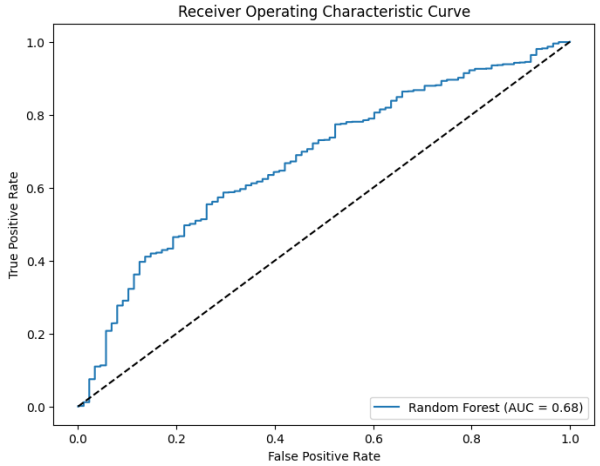
8. *Visualisasi Tree*




Gambar 3. 21 Visualisasi tree

Berikut merupakan hasil plot dari pohon keputusan yang sudah dibuat. Untuk data uji pertama, dengan kedalaman 1 dapat dilihat bahwa fitur ke-62 yang pertama kali memisahkan data tersebut diikuti dengan fitur ke-12, serta ke-8, dan seterusnya. Hasil peluang dari data uji tersebut yaitu 0.82 diterima dan 0.18 ditolak dari jurusan yang dipilih.

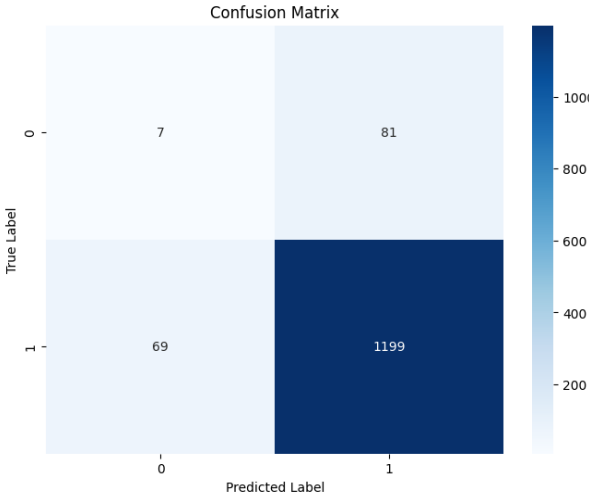
9. *Visualisasi Receiver Operation Characteristic Curve*




Gambar 3. 22 *Visualisasi receiver operation characteristic curve*

Hasil visualisasi *Receiver Operation Characteristic Curve* dapat terlihat bahwa masih banyak *room of improvement* yang bisa dilakukan untuk menjadikan model *Random Forest* ini menjadi lebih baik lagi.

10. *Visualisasi Confusion Matrix*




Gambar 3. 23 Visualisasi confusion matrix

Di dalam *Confusion Matrix* dapat dilihat walaupun akurasi yang dihasilkan sebesar 89% namun model masih banyak salah prediksi terhadap data-data yang berlabel 0.

## D. *Alokasi Kerja*

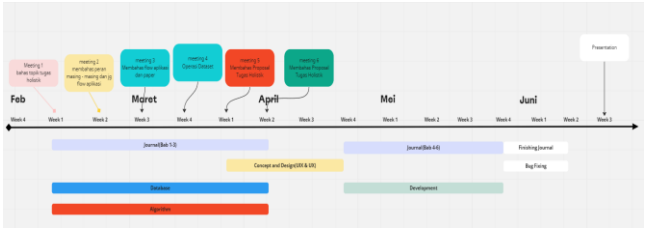
Berikut ini merupakan alokasi kerja yang digunakan tim saat melakukan proses pengerjaan proyek terletak pada tabel 1.2.

| Nama Anggota | *Role* | Tugas |
|---|---|---|
| Kafka | *Web Developer* | - Membuat *website*<br>- Membuat API<br>-Membuat *database* |
| Avinash | *Mobile Developer* | Membuat aplikasi *mobile* |
| Sherly | *Machine Learning Engineer* | Membuat algoritma |
| Devion | *Srum Master* | - Membuat *weekly meeting*<br>- *Product planning* |

Tabel 1. 2 Alokasi kerja

## E. *Jadwal dan Rencana Kerja*

Berikut adalah jadwal dan rencana kerja dari tahap awal yaitu konsep dan topik awal tugas holistik hingga presentasi akhir. Setiap minggu terdapat *weekly meeting* yang akan membahas *progress* dari tugas holistik, baik dari pembersihan data, pengerjaan jurnal, pengembangan aplikasi, pembuatan *database*, serta penerapan *Algoritma Random Forest* dan persiapan untuk presentasi akhir. Setiap minggu dilakukan evaluasi terhadap hasil pekerjaan yang telah dilakukan pada minggu sebelumnya, serta memberikan *feedback* untuk perbaikan kedepannya. Tujuannya adalah untuk memastikan bahwa setiap tahapan dari tugas holistik dapat diselesaikan dengan baik dan tepat waktu, serta menghasilkan hasil yang maksimal dan memuaskan. Dengan adanya jadwal dan rencana kerja yang teratur kemudian diikuti dengan *weekly meeting* serta evaluasi, diharapkan tim dapat bekerja secara efektif dan efisien dalam menyelesaikan tugas holistik.




Gambar 3. 24 Jadwal dan rencana kerja

## IV. KESIMPULAN

Dari hasil penelitian yang dilakukan, masih terdapat *room of improvement* yang perlu dilakukan oleh Model *Random Forest* terkait permasalahan data tidak seimbang. Walaupun sudah diterapkan pendekatan SMOTE namun masih belum sepenuhnya berhasil menangani permasalah data tidak seimbang. Terdapat beberapa metode lainnya yang layak dicoba untuk meningkatkan hasil prediksi yaitu:

1. Mengumpulkan data hingga dataset yang dimiliki lebih seimbang.
2. Mengimplementasikan *undersampling* dengan *folding.*
3. Menggunakan model klasifikasi lain seperti *Gradient Boosting, SVM, neural network.*

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh dosen, staf, dan rekan-rekan di Universitas Kristen Maranatha yang telah memberikan dukungan penuh dalam penulisan artikel ini.

## DAFTAR PUSTAKA

[1] G. V. Parmonang, "Sistem Pendukung Keputusan Penerimaan Mahasiswa Baru," Galang Tanjung, no. 2504, pp. 1–9, 2019.

[2] J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, "A comparison of random forest variable selection methods for classification prediction modeling," *Expert Syst. Appl.*, vol. 134, pp. 93–101, 2019, doi: 10.1016/j.eswa.2019.05.028.

[3] R. R. Waliyansyah and N. D. Saputro, "Forecasting New Student Candidates Using the Random Forest Method," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 11, no. 1, p. 44, 2020, doi: 10.24843/lkjiti.2020.v11.i01.p05.

[4] IBM. "What is a REST API?" IBM. https://www.ibm.com/topics/rest-apis. Diakses pada 10 April 2023.

[5] What is REST - REST API Tutorial. https://restfulapi.net/ Accessed 4/10/2023.

[6] Red Hat. "What is a REST API?" Red Hat. https://www.redhat.com/en/topics/api/what-is-a-rest-api. Diakses pada 10 April 2023.

[7] MySQL. https://www.mysql.com/ Accessed 4/10/2023.

[8] What Is MySQL? | Oracle. https://www.oracle.com/mysql/what-is-mysql/ Accessed 4/10/2023.

[9] Syed, A.A. (2010). Agile Software Development: A Review of the Literature. IEEE Conference on Agile Management, pp. 357-362.

[10] Healy, J. (2016). The Agile Manifesto: A Study of Software Development Methodologies. International Journal of Business Information Systems, 22(1), pp. 66-83.

[11] Rao, S. (2017). The Agile Methodology: An Overview and Comparison with Waterfall. Journal of Applied Computer Science & Mathematics, 21(5), pp. 33-40.

[12] Ahmed, R., Al-Qutaish, R., & Kamal, M. A. (2019). Agile software development: a comprehensive review of literature. International Journal of Project Management, 37(2), 340-359.