IMPLEMENTASI *RANDOM FOREST UNTUK SISTEM* REKOMENDASI *MULTI-CLASS PARKING STAND* PADA SISTEM INFORMASI *APRON MOVEMENT CONTROL* BANDAR UDARA HALIM PERDANAKUSUMA

PROPOSAL SKRIPSI

SYARIF ADRIAN MANGARJA LUBIS

NIM : 221051013

PROGRAM STUDI INFORMASI

FAKULTAS ILMU KOMPUTER DAN DESAIN

UNIVERSITAS DIRGANTARA MARSEKAL SURYADARMA

JAKARTA

2025

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pergerakan pesawat di kawasan apron merupakan salah satu komponen kritikal dalam sistem operasional bandara yang berkaitan langsung dengan aspek keselamatan dan efisiensi operasional. Unit *Apron Movement Control (AMC)* bertanggung jawab penuh atas seluruh pergerakan di sisi udara (airside) apron, menjadikan peran mereka sangat strategis dalam menjaga kelancaran operasi penerbangan. Kawasan *apron* mencakup berbagai aktivitas esensial seperti parkir, reposisi, dan persiapan keberangkatan pesawat, yang memerlukan koordinasi tinggi antara berbagai unit operasional termasuk *Air Traffic Control (ATC),* layanan ground handling, dan personel teknis lapangan (Yesicatama & Widagdo, 2025).

Alokasi *parking stand* merupakan tugas inti dari unit *AMC*, mencakup sekitar 70% dari seluruh tanggung jawab operasional unit ini. Keputusan alokasi stand yang tidak optimal dapat berdampak langsung terhadap efisiensi *turnaround time* pesawat, jarak tempuh penumpang ke *gate*, serta potensi konflik penggunaan fasilitas *apron.* Berdasarkan observasi langsung selama pelaksanaan program magang di Unit *AMC* Bandar Udara Halim Perdanakusuma, penulis mengidentifikasi bahwa setidaknya 2 kali dalam sebulan, pesawat yang telah mendarat harus menunggu di taxiway selama 1-2 menit karena ketidakpastian dalam penentuan stand optimal. Situasi ini menunjukkan adanya gap dalam proses pengambilan keputusan alokasi stand yang perlu diatasi melalui pendekatan berbasis data.

Bandar Udara Halim Perdanakusuma memiliki karakteristik operasional yang unik karena melayani beragam kategori penerbangan, termasuk kargo, komersial, charter/private, dan bahkan penerbangan militer. Kompleksitas ini menciptakan tantangan tersendiri dalam pengelolaan *apron*, karena setiap kategori penerbangan memiliki kebutuhan spesifik terkait alokasi *stand,* waktu *turnaround,* dan aksesibilitas terminal. Sebagai contoh, beberapa maskapai memiliki preferensi stand tertentu berdasarkan kedekatan dengan *gate* atau terminal mereka, sementara

penerbangan kargo memerlukan *st*and dengan aksesibilitas khusus untuk loading/unloading. Kondisi ini membutuhkan sistem yang mampu mengakomodasi berbagai variabel operasional secara simultan.

Di berbagai bandara, termasuk Bandar Udara Halim Perdanakusuma, pengelolaan pergerakan pesawat dan alokasi parking stand masih dilaksanakan secara manual berdasarkan intuisi dan pengalaman operator. Koordinasi operasional dilakukan menggunakan radio komunikasi *handheld transceiver (HT),* sementara dokumentasi alokasi dan pemantauan dilakukan secara terpisah dalam spreadsheet individual berdasarkan kategori penerbangan. Sistem semacam ini memiliki potensi besar terhadap inkonsistensi data, inefisiensi keputusan, dan keterlambatan respons, terutama saat *traffic* padat (Gunawan & Rahimudin, 2023)

Dalam literatur sistem pendukung keputusan untuk transportasi udara, berbagai metode prediksi dan klasifikasi telah diusulkan untuk mengatasi masalah alokasi sumber daya dinamis seperti *parking stand*. (Suryaman & Wang, 2022) menerapkan algoritma Decision Tree (J48) untuk prediksi slot time penerbangan, menunjukkan bahwa pendekatan klasifikasi berbasis *machine learning* dapat meningkatkan akurasi prediksi dibandingkan metode manual. Namun, konteks penelitian tersebut berfokus pada penjadwalan waktu, bukan alokasi stand secara spasial dengan pertimbangan multi-kelas.

Metode manual berbasis intuisi operator, yang saat ini diterapkan di Bandara Halim Perdanakusuma, memiliki kelebihan dalam fleksibilitas untuk mengakomodasi permintaan khusus seperti alokasi stand prioritas untuk penumpang *VIP* atau evakuasi medis. Namun, metode ini memiliki kelemahan signifikan: lambat dalam pengambilan keputusan (rata-rata 1-2 menit per alokasi), inkonsistensi antar operator, dan ketergantungan pada pengalaman individual tanpa dukungan data historis.

Beberapa algoritma *machine learning* telah dipertimbangkan sebagai alternatif untuk tugas klasifikasi multi-kelas seperti alokasi stand:

Support Vector Machine (SVM) memiliki keunggulan dalam menangani data berdimensi tinggi dan efisien secara memori. Namun, *SVM* tidak cocok untuk masalah multi-kelas dengan banyak kategori target, karena memerlukan strategi

"one-vs-rest" atau *"one-vs-one"* yang menjadi tidak efisien ketika jumlah stand mencapai 20 kategori (A0-A19). Selain itu, *SVM* tidak menghasilkan output probabilitas secara langsung, sehingga sulit untuk mengimplementasikan strategi rekomendasi *Top-K* yang memberikan beberapa opsi kepada operator (Safira et al., 2024).

Neural Networks mampu mempelajari hubungan non-linear yang kompleks dan memiliki fleksibilitas arsitektur tinggi. Namun, pendekatan ini memerlukan data pelatihan dalam jumlah sangat besar jauh melebihi data historis pergerakan apron yang tersedia di Bandara Halim. Neural networks juga bersifat *"black box",* yang menjadi kelemahan kritis dalam operasi bandara di mana operator perlu memahami alasan di balik rekomendasi yang diberikan sistem. Risiko *overfitting* juga tinggi ketika data pelatihan terbatas (Ibrahim, 2023).

Single Decision Tree (J48/C4.5) memiliki kelebihan tinggi dalam hal interpretabilitas dan kecepatan. Namun, algoritma ini memiliki kelemahan fundamental: kecenderungan *overfitting* yang sangat tinggi, *variance* yang besar (perubahan kecil dalam data dapat menghasilkan struktur pohon yang sangat berbeda), dan ketidakstabilan dalam menghadapi *class imbalance*. Dalam konteks alokasi stand di mana beberapa stand digunakan jauh lebih sering daripada yang lain (misalnya, stand komersial lebih sering digunakan dibanding stand kargo), *single decision tree* cenderung menghasilkan prediksi yang bias terhadap kelas mayoritas (Mishra et al., 2025).

Algoritma *Random Forest* dipilih dalam penelitian ini karena kemampuannya mengatasi berbagai kelemahan metode-metode sebelumnya sambil mempertahankan keunggulan yang diperlukan untuk sistem *AMC*. *Random Forest* merupakan algoritma *ensemble learning* yang membangun banyak *decision tree* secara paralel dan melakukan prediksi berdasarkan agregasi hasil dari semua pohon melalui mekanisme *voting*. Pendekatan ini secara dramatis mengurangi *overfitting* melalui *averaging ensemble*, yang sangat penting mengingat keterbatasan data historis yang tersedia (Mishra et al., 2025)

Keunggulan utama *Random Forest* meliputi:

1. Penanganan *Class Imbalance*: Melalui parameter *class_weight='balanced_subsample', Random Forest* dapat menangani ketidakseimbangan distribusi kelas secara otomatis. Hal ini krusial karena *dataset* operasional *AMC* menunjukkan bahwa beberapa *parking stand* (seperti *stand* komersial utama) digunakan jauh lebih sering dibanding *stand* khusus seperti A0 yang hanya diperuntukkan bagi pesawat kecil *(Cessna, Pilatus).*
2. Output Probabilistik: *Random Forest* menghasilkan distribusi probabilitas untuk setiap kelas *stand*, memungkinkan implementasi strategi *Top-K prediction*. Operator dapat menerima 3 rekomendasi stand teratas dengan confidence score masing-masing, memberikan fleksibilitas untuk mempertimbangkan kondisi lapangan secara *real-time* (Petersen et al., 2022)
3. *Feature Importance*: Model dapat mengidentifikasi fitur-fitur yang paling berkontribusi terhadap prediksi (misalnya, tipe pesawat, kategori penerbangan, maskapai), memberikan transparansi operasional yang penting bagi petugas *AMC* dalam memahami dasar rekomendasi sistem.
4. Generalisasi yang Baik: Dengan moderate *dataset size* seperti data historis pergerakan *AMC*, *Random Forest* mampu belajar pola umum tanpa mengalami *overfitting* ekstrem seperti pada *single decision tree* atau *underfitting* seperti pada model linear sederhana (Endut et al., 2022) Kelemahan *Random Forest* terutama terletak pada biaya komputasi yang lebih tinggi dibandingkan *single tree* karena perlu melatih ratusan pohon, serta interpretabilitas yang lebih rendah karena tidak dapat memvisualisasikan satu jalur keputusan yang jelas. Namun, kelemahan ini dapat diterima mengingat keunggulan signifikan dalam akurasi dan stabilitas prediksi, serta fakta bahwa *feature importance* tetap dapat diekstraksi untuk keperluan audit dan transparansi operasional.

Meskipun *Random Forest* menjadi pilihan algoritma yang paling sesuai, implementasinya dalam konteks sistem *AMC* menghadapi beberapa tantangan teknis spesifik:

Pertama, *class imbalance* menjadi masalah utama. *Dataset* operasional *AMC* memiliki 20 kelas parking stand (A0-A19), namun distribusi penggunaannya sangat tidak merata. Stand komersial seperti A1-A5 digunakan hampir setiap hari, sementara *stand* khusus seperti A0 (untuk pesawat kecil) hanya digunakan sesekali ketika ada penerbangan *charter* dengan *Cessna* atau *Pilatus*. Tanpa penanganan khusus, model akan cenderung mengabaikan kelas minoritas dan selalu merekomendasikan *stand* mayoritas.

Kedua, kompleksitas *multi*-kelas dengan 20 kategori stand memerlukan konfigurasi *hyperparameter* yang optimal. Jumlah pohon *(n_estimators),* kedalaman pohon *(max_depth),* dan *parameter sampling* perlu *dituning* secara sistematis untuk mencapai keseimbangan antara akurasi dan efisiensi komputasi. *Grid Search* dengan *cross-validation* diperlukan untuk mengidentifikasi kombinasi parameter terbaik.

Ketiga, integrasi dengan sistem *existing* menimbulkan tantangan arsitektural. Sistem informasi *AMC* dibangun dengan *PHP native* sebagai *backend*, sementara model *Random Forest* dilatih dan dieksekusi menggunakan *Python (scikit-learn).* Diperlukan mekanisme komunikasi yang handal antara kedua platform untuk memungkinkan inferensi model secara real-time tanpa menimbulkan *bottleneck* performa.

Keempat, kebutuhan akan *explainability* dalam konteks operasional bandara. *Operator* AMC perlu memahami mengapa sistem merekomendasikan *stand* tertentu, terutama dalam situasi kritis atau ketika rekomendasi sistem berbeda dari intuisi mereka. *Random Forest* sebagai *ensemble model* memiliki interpretabilitas yang lebih rendah dibanding single *decision tree*.

Untuk mengatasi tantangan *class imbalance*, penelitian ini akan menerapkan *parameter class_weight='balanced_subsample'* dalam konfigurasi *RandomForestClassifier.* Pendekatan ini secara otomatis menyesuaikan bobot sampel untuk setiap *bootstrap sample*, memberikan bobot lebih tinggi pada kelas minoritas seperti stand A0. Dengan demikian, model tidak akan mengabaikan stand yang jarang digunakan namun tetap valid dan penting dalam skenario tertentu.

Untuk optimasi hyperparameter, akan dilakukan *Grid Search dengan 5-fold cross-validation* yang menguji kombinasi sistematis dari *parameter n_estimators* (jumlah pohon), *max_depth* (kedalaman pohon), *min_samples_leaf, min_samples_split, dan class_weight*. *Stratified sampling* akan diterapkan dalam proses *split data* untuk memastikan distribusi kelas yang proporsional antara *training set* dan *test set*, mencegah *bias* evaluasi.

Implementasi strategi *Top-K prediction* akan mengatasi kompleksitas *multi*-kelas dengan memberikan 3 rekomendasi stand teratas beserta *confidence score* masing-masing, bukan hanya satu prediksi tunggal. Pendekatan ini tidak hanya meningkatkan fleksibilitas operasional (operator dapat memilih dari beberapa opsi berdasarkan kondisi lapangan), tetapi juga meningkatkan metrik evaluasi sistem target *Top-3 accuracy minimal* 80% jauh lebih realistis dan praktis dibandingkan *Top-1 accuracy* dalam konteks *decision support system.*

Untuk integrasi *PHP-Python*, akan digunakan fungsi *proc_open()* dengan komunikasi *bidirectional* melalui *JSON*. Data *input* (tipe pesawat, maskapai, kategori penerbangan) akan dikirim dari *PHP* ke *Python* melalui *stdin* dalam format *JSON*, dan model akan mengembalikan hasil prediksi beserta probabilitas melalui stdout. Pendekatan ini lebih robust dibandingkan *exec()* atau *shell_exec()* karena memungkinkan *error handling* yang lebih baik dan menghindari masalah *shell escaping.*

Untuk meningkatkan *explainability*, sistem akan mengekstrak *feature importance* dari *model Random Forest* yang terlatih. Visualisasi *feature importance* akan menunjukkan variabel mana (tipe pesawat, kategori penerbangan, maskapai, zona *stand*) yang paling berpengaruh dalam keputusan alokasi. Informasi ini dapat digunakan operator untuk memvalidasi rekomendasi sistem dan memahami faktor-faktor yang berkontribusi terhadap prediksi.

Kesenjangan yang ingin dijawab oleh penelitian ini adalah belum adanya sistem yang secara holistik menggabungkan pencatatan operasional secara *real-time*, visualisasi grafis kondisi apron, validasi data terintegrasi, dan rekomendasi berbasis *machine learning* dalam satu platform terintegrasi yang dapat diakses oleh berbagai unit operasional. Meskipun telah terdapat penelitian mengenai

pengelolaan apron dan penerapan *machine learning* dalam sektor transportasi, pendekatan yang memadukan prediksi alokasi *parking stand* dengan visualisasi operasional dalam konteks bandara di Indonesia masih sangat terbatas.

Studi serupa mengenai pengembangan sistem informasi *AMC* telah dilakukan di Bandara Internasional Sultan Hasanuddin Makassar (Jumlad & Nurhalisa, 2024) dan sistem informasi parkir pesawat di Bandara Internasional Minangkabau (Ulfa et al., 2023). Namun, sistem-sistem tersebut belum mengintegrasikan kemampuan prediktif berbasis ensemble machine learning dengan visualisasi kondisi apron secara komprehensif serta strategi *Top-K recommendation* yang memberikan fleksibilitas operasional kepada petugas lapangan.

Oleh karena itu, penelitian ini bertujuan untuk merancang dan mengembangkan sistem informasi *Apron Movement Control (AMC)* berbasis web yang terintegrasi, yang menyatukan proses pencatatan operasional, visualisasi status apron secara real-time, dan fitur prediksi alokasi parking stand berbasis algoritma *Random Forest* dengan strategi *Top-K prediction*. Sistem ini diharapkan dapat meningkatkan efisiensi operasional unit *AMC* di Bandar Udara Halim Perdanakusuma dengan menyediakan rekomendasi berbasis data historis yang akurat (target *Top-3 accuracy* minimal 80%), mengurangi waktu pengambilan keputusan alokasi stand, serta meminimalisasi risiko miskomunikasi antarunit operasional melalui platform terpusat dengan *role-based access control.*

Integrasi antara algoritma klasifikasi *ensemble (Random Forest),* visualisasi kondisi operasional secara real-time, manajemen berbasis peran *(role-based access),* dan *Decision Support System* dalam satu platform merupakan pendekatan yang belum banyak ditemukan dalam literatur maupun implementasi sistem sejenis di sektor aviasi Indonesia. Penelitian ini memberikan kontribusi orisinal baik dari perspektif teknis maupun praktikal dalam membangun sistem pendukung keputusan berbasis *Machine Learning* di unit *AMC* yang dapat direplikasi di bandara-bandara lain dengan karakteristik operasional serupa.

1.2 Identifikasi Masalah

1.2.1 Dokumentasi Manual dan Terpisah

Pencatatan operasional dilakukan melalui tiga *file Google Sheets* terpisah: *Apron Monitoring Sheet* (harian), *Monthly Charter Report*, dan *Monthly RON (Remain Overnight) Report*. Fragmentasi dokumentasi ini menyebabkan redundansi data dan inkonsistensi antar dokumen.

Berdasarkan pengamatan penulis, duplikasi nomor penerbangan (*flight number)* pada *Apron Monitoring Sheet* terjadi 1-3 kali per hari operasional. Ketika data dari *monitoring sheet* harian didistribusikan ke laporan bulanan *Charter* dan *RON* di akhir bulan, inkonsistensi ini dapat mengakibatkan kesalahan pencatatan yang berdampak finansial signifikan bagi maskapai. Sebagai contoh, jika terdapat kesalahan mencatat bahwa pesawat suatu maskapai melakukan *RON* selama 3 hari padahal realitanya hanya 1 hari, maskapai tersebut akan dikenakan biaya parkir untuk 3 malam kerugian yang dapat dihindari dengan sistem validasi data yang terintegrasi.

Meskipun sistem *Google Sheets* memungkinkan akses simultan oleh 2 operator, konflik edit jarang terjadi. Namun, ketiadaan mekanisme validasi otomatis dan audit trail menyebabkan kesalahan entri sulit terdeteksi hingga proses rekonsiliasi bulanan dilakukan.

1.2.2 Tidak Ada Visualisasi *Real-Time*

Ketiadaan antarmuka grafis untuk memvisualisasikan status apron secara real-time menghambat operator dalam memahami kondisi lapangan dengan cepat. Berdasarkan pengamatan penulis, seorang operator *AMC* memerlukan waktu rata-rata 30 detik untuk mengidentifikasi stand mana yang sedang terisi dan mana yang tersedia dengan cara membaca spreadsheet baris per baris dan melakukan *cross-check* manual dengan informasi radio.

Dalam sistem yang akan dikembangkan, informasi ini dapat diperoleh secara instan melalui representasi *visual apron* yang menampilkan status setiap *stand* dengan indikator warna (terisi/kosong/dialokasikan). Keterlambatan 30 detik per *query* mungkin tampak kecil, namun dalam kondisi *peak hour* dengan frekuensi

query tinggi, akumulasi waktu ini dapat menghambat responsivitas operasional secara keseluruhan.

Lebih kritis lagi, berdasarkan observasi penulis, kesalahan alokasi *stand* akibat kurangnya visibilitas status *apron* terjadi maksimal 2 kali per hari. Kesalahan ini biasanya berupa alokasi *stand* yang sebenarnya masih terisi atau akan segera digunakan oleh penerbangan lain, yang memaksa operator untuk melakukan realokasi mendadak menambah beban kerja dan potensi konflik jadwal.

1.2.3 Alokasi *Stand* Tanpa Dukungan Sistem

Penetapan parking *stand* saat ini dilakukan sepenuhnya berdasarkan intuisi dan pengalaman operator tanpa dukungan sistem berbasis data historis atau model prediktif. Proses pengambilan keputusan alokasi *stand* memerlukan waktu rata-rata 1-2 menit, dengan durasi yang bervariasi tergantung pada kompleksitas kasus tipe penerbangan, tingkat kesibukan *apron*, dan ada tidaknya kondisi khusus seperti evakuasi medis atau penumpang *VVIP*.

Berdasarkan pengamatan penulis, alokasi *stand* yang tidak optimal terjadi 3-4 kali per hari tergantung pada *volume* lalu lintas. Alokasi tidak optimal dalam konteks ini bukan berarti "salah", melainkan kurang efisien misalnya, menempatkan pesawat kecil di stand besar yang seharusnya dicadangkan untuk pesawat *wide-body*, atau mengalokasikan maskapai ke *stand* yang jauh dari gate mereka padahal ada opsi lebih dekat yang tersedia. Situasi ini mengakibatkan:

1. Pemborosan kapasitas *stand*: *Stand* besar terpakai untuk pesawat kecil
2. Jarak tempuh penumpang lebih jauh: Menambah waktu *turnaround*
3. Ketidakpuasan maskapai: Terutama yang memiliki preferensi *stand* berdasarkan lokasi *gate*

Sistem prediksi berbasis *Random Forest* yang akan dikembangkan bertujuan memberikan rekomendasi *Top-3 stand dengan confidence score*, memungkinkan operator membuat keputusan lebih cepat dan lebih optimal dengan tetap mempertahankan *human judgment* sebagai keputusan final.

1.2.4 Validasi Data Lemah

Tidak adanya sistem validasi data otomatis menyebabkan entri data yang tidak konsisten lolos tanpa terdeteksi hingga tahap rekonsiliasi manual. Berdasarkan observasi penulis, kesalahan *timestamp* terjadi maksimal 2 kali per hari, dengan jenis kesalahan paling umum adalah waktu off-block yang tercatat lebih awal daripada waktu *on-block* akibat kesalahan ketik (*typo*).

Meskipun frekuensi kesalahan ini relatif rendah dan tidak menimbulkan masalah operasional segera, inkonsistensi data dapat mengganggu analisis historis dan mengurangi kredibilitas laporan operasional. Sistem yang akan dikembangkan akan menerapkan validasi logika temporal *(on-block* harus lebih awal dari *off-block)* serta validasi format data untuk mencegah entri yang tidak valid sejak awal.

1.2.5 Koordinasi Antarunit Rentan Miskomunikasi

Komunikasi operasional antara unit AMC dan unit lain, terutama *Air Traffic Control (ATC)*, masih mengandalkan radio HT sebagai sarana utama. Komunikasi radio rentan mengalami *noise*, kesalahan interpretasi, dan keterlambatan transmisi informasi. Berdasarkan pengamatan penulis, miskomunikasi antara *AMC* dan *ATC* terjadi 1-2 kali per hari, meskipun sebagian besar tidak bersifat kritis.

Namun, dampak dari miskomunikasi ini tetap signifikan: seperti yang telah diuraikan pada bagian Latar Belakang, penulis mengamati bahwa setidaknya 2 kali dalam sebulan, pesawat yang telah mendarat tidak segera diberikan alokasi *stand* karena ketidakpastian informasi antara *AMC* dan *ATC*, menyebabkan pesawat harus menunggu di *taxiway* selama 1-2 menit sebelum akhirnya dialokasikan. Meskipun durasi penundaan ini relatif singkat, dalam konteks operasi bandara yang padat, akumulasi delay dapat berdampak pada keseluruhan efisiensi sistem dan berpotensi mempengaruhi *on-time performance* maskapai.

1.2.6 Tidak Ada Sistem Terpadu dengan Akses Terbatas

Setiap unit operasional di bandara *AMC, ATC, ground handling* menggunakan sistem pencatatan dan komunikasi yang berbeda-beda. Ketiadaan platform terpusat menyebabkan informasi tidak terdistribusi *secara real-time* antar

unit, mengakibatkan keterlambatan informasi dan potensi miskomunikasi lintas-unit.

Sistem yang akan dikembangkan mengatasi masalah ini dengan menyediakan *role-based access control* yang memungkinkan operator *AirNav (ATC)* untuk melihat status *apron* secara *real-time*, termasuk *stand* mana yang telah dialokasikan untuk pesawat tertentu. Dengan visibilitas bersama terhadap kondisi *apron*, koordinasi antara *AMC* dan *ATC* dapat dilakukan lebih efisien tanpa bergantung sepenuhnya pada komunikasi *radio* yang rentan *error*. Meskipun lag informasi antar unit saat ini hanya dalam hitungan menit, pengurangan *gap* ini tetap berkontribusi terhadap peningkatan efisiensi operasional secara keseluruhan.

Keenam masalah di atas menunjukkan adanya kebutuhan mendesak akan sistem informasi terintegrasi yang mampu menyatukan proses pencatatan, validasi data, visualisasi kondisi operasional secara *real-time*, serta dukungan pengambilan keputusan berbasis data historis melalui model *machine learning*. Sistem yang dirancang dalam penelitian ini diharapkan dapat menjawab seluruh tantangan tersebut dalam satu *platform* terpadu.

1.3 Rumusan Masalah

Berdasarkan latar belakang dan identifikasi masalah yang telah diuraikan pada bagian sebelumnya, rumusan masalah dalam penelitian ini dirumuskan sebagai berikut:

1. Bagaimana merancang arsitektur sistem informasi *Apron Movement Control (AMC)* berbasis web yang dapat mengintegrasikan minimal tiga jenis laporan operasional (*monitoring apron, RON, dan charter*) dengan *response time* maksimal 5 detik, serta menyediakan visualisasi kondisi apron *secara real-time*?
2. Bagaimana mengimplementasikan algoritma *Random Forest* untuk prediksi alokasi *parking stand* dengan target *Top-3 accuracy minimal* 80% berdasarkan empat parameter operasional utama (tipe pesawat, kategori penerbangan, dan preferensi maskapai), serta menghasilkan rekomendasi dalam waktu maksimal 5 detik?

3. Seberapa besar kontribusi sistem terhadap efisiensi operasional *AMC* yang diukur melalui: (a) pengurangan waktu rata-rata pengambilan keputusan alokasi stand, (b) penurunan *tingkat error rate* dalam alokasi stand, dan (c) penurunan frekuensi miskomunikasi antarunit operasional, dibandingkan dengan sistem manual *existing*?

1.4 Batasan Masalah

1. Penelitian ini terbatas pada pengembangan sistem informasi untuk keperluan internal unit *Apron Movement Control (AMC)* di Bandar Udara Halim Perdanakusuma, tanpa mencakup integrasi langsung dengan sistem operasional milik *AirNav* Indonesia maupun unit eksternal lainnya.
2. Fitur prediksi alokasi stand diimplementasikan secara eksklusif menggunakan algoritma *Random Forest* dan dibatasi pada parameter-parameter tertentu yang tersedia dari repositori data historis *AMC*, mencakup jenis penerbangan, tipe pesawat, dan preferensi maskapai penerbangan.
3. Data yang digunakan untuk proses pelatihan dan pengujian model bersumber dari rekaman operasional *AMC* dalam periode waktu tertentu (misalnya satu bulan), dengan asumsi bahwa data tersebut telah tervalidasi dan merepresentasikan pola operasional secara umum.
4. Sistem yang dikembangkan berbasis *platform web* dan hanya mencakup fungsionalitas utama berupa pencatatan pergerakan pesawat, visualisasi status *apron* secara *real-time*, *dashboard* operasional, serta fitur prediksi *stand.* Pengembangan fungsionalitas tambahan seperti sistem notifikasi, integrasi dengan sistem *flight plan*, atau versi aplikasi mobile tidak termasuk dalam cakupan penelitian ini.
5. Evaluasi sistem dibatasi pada aspek fungsionalitas dan kebergunaan (*usability*) melalui metode pengujian *black-box* serta umpan balik terbatas dari pengguna internal. Pengujian performa dalam skala besar atau implementasi langsung dalam lingkungan operasional *real-time* belum dilaksanakan dalam lingkup penelitian ini.

1.5 Metode Penelitian

1.5.1 Metode Pengumpulan Data

Pengumpulan data dalam penelitian ini dilaksanakan melalui tiga pendekatan utama, yaitu observasi lapangan, dokumentasi internal, dan studi literatur. Observasi dilakukan secara langsung oleh penulis selama periode pelaksanaan program magang di Unit *Apron Movement Control (AMC)* Bandar Udara Halim Perdanakusuma, dengan posisi sebagai trainee operator di bawah supervisi penuh petugas *AMC*. Dokumentasi yang digunakan meliputi rekaman operasional berupa spreadsheet pemantauan pergerakan pesawat dan alokasi *stand* yang digunakan secara reguler oleh unit *AMC*. Selanjutnya, studi literatur dilakukan terhadap artikel jurnal ilmiah, publikasi teknis, dan referensi akademik yang relevan guna mendukung perancangan sistem serta pemilihan algoritma prediksi yang tepat.

1.5.2 Metode Pengembangan Sistem

Pengembangan sistem dilaksanakan dengan menerapkan pendekatan metodologi *Agile*, yang memfasilitasi proses iteratif dan adaptif terhadap perubahan kebutuhan operasional di lapangan. Metodologi ini dipilih berdasarkan fleksibilitasnya dalam menangani dinamika operasional serta memungkinkan revisi sistem secara bertahap berdasarkan masukan dari pengguna. Tahapan pengembangan sistem mencakup:

1. **Observasi dan Identifikasi Kebutuhan:** Proses diawali dengan pemahaman komprehensif terhadap alur kerja dan tantangan operasional di unit *AMC* melalui observasi langsung serta pengalaman praktis di lapangan.
2. **Perancangan Sistem :** Tahapan ini meliputi penyusunan desain antarmuka pengguna (user *interface*), perancangan struktur basis data, serta pemetaan alur kerja sistem yang selaras dengan prosedur operasional *AMC*. Fokus perancangan diarahkan pada simplifikasi proses pencatatan dan peningkatan visibilitas status *apron* secara *real-time.*
3. **Implementasi Sistem :** Sistem dikembangkan menggunakan teknologi berbasis web dengan platform *PHP* dan *MySQL*, serta komponen *front-end* untuk menampilkan representasi visual apron secara interaktif. Fungsionalitas

utama mencakup dashboard operasional, formulir pencatatan pergerakan, dan sistem visualisasi *apron*.

4. **Integrasi *Random Forest* :** Data historis pergerakan pesawat akan digunakan sebagai dataset untuk proses pelatihan dan pengujian model pembelajaran mesin (*machine learning*) berbasis algoritma *Random Forest*. *Random Forest* dipilih karena kemampuannya menangani *multi-class classification* dengan 20 kelas parking stand serta menghasilkan prediksi probabilistik untuk strategi *Top-K recommendation*. Model ini akan menghasilkan rekomendasi alokasi parking stand berdasarkan parameter seperti jenis pesawat, kategori penerbangan, dan preferensi maskapai penerbangan, dengan target *Top-3 accuracy* minimal 80%
5. **Pengujian dan Evaluasi Sistem:** Pengujian dilaksanakan menggunakan metode *black-box* untuk memastikan seluruh fungsionalitas sistem beroperasi sesuai dengan spesifikasi yang telah ditetapkan. Selain itu, validasi terbatas dilakukan melalui diskusi dan simulasi bersama operator *AMC* guna memperoleh umpan balik yang relevan untuk perbaikan iteratif sistem.

1.6 Tujuan Penelitian

Penelitian ini bertujuan merancang dan mengembangkan sistem informasi *Apron Movement Control (AMC)* berbasis web yang terintegrasi untuk meningkatkan efisiensi operasional unit *AMC* di Bandar Udara Halim Perdanakusuma. Sistem ini dirancang untuk mensubstitusi proses pencatatan manual tersegmentasi dengan platform terpusat yang menyediakan visualisasi *real-time* kondisi *apron*, serta mengimplementasikan algoritma *Random Forest* untuk menghasilkan prediksi alokasi parking stand berdasarkan parameter operasional (jenis pesawat, kategori penerbangan, dan preferensi maskapai). Implementasi sistem ini diharapkan dapat memfasilitasi pengambilan keputusan yang lebih cepat dan akurat berbasis data historis, khususnya pada kondisi kepadatan tinggi, sekaligus meminimalisasi risiko miskomunikasi antarunit operasional

1.7 Manfaat Penelitian

Penelitian ini diharapkan memberikan kontribusi dari dua perspektif, yaitu manfaat praktis bagi operasional bandara dan manfaat akademis sebagai kontribusi keilmuan dalam bidang Sistem Informasi dan teknologi penerbangan.

1. **Bagi Instansi**

   Sistem yang dikembangkan dapat memfasilitasi operator *AMC* dalam proses pencatatan pergerakan pesawat secara lebih efisien dan akurat, sekaligus menyediakan rekomendasi alokasi stand berbasis data untuk mengakselerasi proses pengambilan keputusan operasional. Melalui fungsionalitas visualisasi *real-time* dan *platform* terpusat, sistem ini dapat mereduksi potensi miskomunikasi serta meningkatkan koordinasi antarunit dalam pengelolaan pergerakan pesawat di kawasan *apron*. Peningkatan efisiensi operasional dan visibilitas komprehensif terhadap kondisi *apron* dapat mendukung pengelolaan bandara yang lebih strategis dan berbasis data.

2. **Bagi Perguruan Tinggi**

   Penelitian ini menyajikan studi kasus implementasi algoritma *machine learning* (*Random Forest*) dalam konteks manajemen pergerakan pesawat dan pengelolaan apron pada lingkungan operasional aktual. Sistem yang dikembangkan dapat dijadikan acuan untuk pengembangan sistem serupa dalam lingkungan aviasi atau sektor transportasi lainnya yang memiliki kompleksitas alokasi sumber daya. Penelitian ini memberikan kontribusi dalam pengembangan keilmuan Sistem Informasi, khususnya pada integrasi antara visualisasi data real-time, pengambilan keputusan berbasis kecerdasan buatan *(AI),* dan sistem operasional berbasis peran *(role-based system).*

3. **Bagi Penulis**

   Penelitian ini memberikan kesempatan bagi penulis untuk mengaplikasikan pengetahuan teoretis yang diperoleh selama perkuliahan ke dalam permasalahan operasional nyata di sektor aviasi. Melalui proses perancangan, implementasi, dan evaluasi sistem, penulis memperoleh pengalaman praktis dalam pengembangan sistem informasi berbasis web, penerapan algoritma pembelajaran mesin, serta pemahaman mendalam

mengenai kompleksitas operasional bandara. Selain itu, penelitian ini memperkaya kompetensi penulis dalam analisis kebutuhan sistem, manajemen proyek pengembangan perangkat lunak, dan kemampuan berpikir kritis dalam mengidentifikasi solusi berbasis teknologi untuk meningkatkan efisiensi operasional.

Selain itu, integrasi antara algoritma klasifikasi *ensemble (Random Forest)*, visualisasi kondisi operasional secara *real-time*, serta manajemen berbasis peran (*role-based access*) dalam satu *platform* terpusat merupakan pendekatan yang belum banyak ditemukan dalam literatur maupun implementasi sistem sejenis di sektor aviasi Indonesia. Hal ini memberikan kontribusi orisinal dalam pengembangan sistem informasi yang tidak hanya bersifat reaktif, tetapi juga prediktif dan kolaboratif.

1.8 Sistematika Penulisan

Sistematika penulisan skripsi ini disusun untuk memberikan gambaran komprehensif mengenai substansi dan alur penelitian yang dilaksanakan. Struktur penyusunan laporan penelitian ini terdiri dari lima bab utama dengan rincian sebagai berikut:

BAB I : Pendahuluan

Bab ini menguraikan latar belakang permasalahan, identifikasi masalah, rumusan masalah, batasan masalah, metodologi yang digunakan, tujuan penelitian, manfaat penelitian, serta sistematika penulisan skripsi secara menyeluruh.

BAB II : Landasan Teori

Landasan Teori Bab ini memuat kerangka teoretis yang melandasi penelitian, mencakup konsep sistem informasi, pengelolaan pergerakan pesawat, algoritma Random Forest, tinjauan pustaka terhadap penelitian terdahulu yang relevan, serta teknologi yang diaplikasikan dalam pengembangan sistem.

BAB III : Metodologi Penelitian

Bab ini menjelaskan metode penelitian yang diterapkan, meliputi teknik pengumpulan data, pendekatan pengembangan sistem menggunakan metodologi Agile, serta tahapan-tahapan dalam proses implementasi dan pengujian sistem. Penjabaran mengenai proses pelatihan dan pengujian model Random Forest juga diuraikan dalam bab ini.

BAB II
LANDASAN TEORI

2.1 Tinjauan Penelitian Terdahulu

2.1.1 Studi Serupa

Beberapa penelitian terdahulu telah mengkaji penerapan sistem informasi, machine learning, serta *decision support system (DSS)* di domain transportasi udara dan logistik. Penelitian-penelitian ini memberikan dasar referensi bagi pengembangan sistem yang lebih terintegrasi seperti yang dikembangkan dalam penelitian ini.

Penelitian oleh Ulfa, Sylvia, dan Saragih (2023) merancang sistem informasi parkir pesawat untuk mendukung layanan *ATC* di Bandara Internasional Minangkabau. Sistem tersebut fokus pada pencatatan dan pengelolaan data parkir secara digital, namun belum mengintegrasikan modul prediksi atau sistem pendukung keputusan.

Selanjutnya, Jumlad dan Nurhalisa, (2024) membahas implementasi sistem informasi AMC di Bandara Internasional Sultan Hasanuddin Makassar. Sistem tersebut menekankan pada manajemen pergerakan pesawat di *apron* secara terpusat, tetapi tidak membahas pemanfaatan algoritma prediktif atau visualisasi *real-time* dalam proses pengambilan keputusan.

Suryaman dan Wang, (2022) melakukan penerapan data mining untuk memprediksi slot time penerbangan menggunakan metode klasifikasi. Fokus penelitian ini adalah efisiensi jadwal penerbangan, bukan manajemen pergerakan di darat, tetapi pendekatannya relevan karena sama-sama menggunakan klasifikasi berbasis machine learning.

Di sisi lain, penelitian oleh (Akbiyik et al., 2024) mengembangkan sistem pendukung keputusan (*DSS*) untuk logistik internal menggunakan pendekatan heuristik. Studi ini menyoroti pentingnya pengambilan keputusan berbasis data di lingkungan operasional, namun konteksnya masih terbatas pada gudang dan distribusi, bukan lingkungan bandara atau *apron control.*

Tabel 2.1 merangkum penelitian-penelitian terdahulu yang relevan dengan penelitian ini beserta perbedaan utama dengan pendekatan yang diusulkan.

Tabel 2. 1 Penelitian Terdahulu

No	Nama Peneliti dan Tahun	Judul	Ringkasan Perbedaan
1	Ulfa, Sylvia, dan Saragih (2023)	*Design of Padang Online Parking Stand Information System to Support Air Traffic Service*	Sistem ini hanya menangani pencatatan dan tampilan informasi parkir tanpa algoritma prediksi atau pengambilan keputusan otomatis. Visualisasinya bersifat statis dan belum mendukung interaktivitas atau integrasi real-time dengan pergerakan aktual pesawat.
2	Jumlad dan Nurhalisa (2024)	Implementasi Sistem Informasi *Apron Movement Control* di Bandara Internasional Sultan Hasanuddin Makassar	Sistem difokuskan pada digitalisasi proses operasional, namun pengambilan keputusan tetap manual oleh petugas. Tidak terdapat integrasi dengan model *machine learning, DSS*, atau modul visualisasi interaktif berbasis data *real-time.*
3	Suryaman dan Wang (2022)	Penerapan *Data Mining* untuk Prediksi *Slot Time* di Bandara Internasional di Indonesia	Penelitian ini menggunakan metode klasifikasi (*Decision Tree*) untuk prediksi waktu slot penerbangan, namun tidak relevan langsung dengan pengelolaan stand atau sistem informasi AMC. Fokus utamanya pada optimasi jadwal, bukan manajemen apron.
4	Akbiyik dan Özen (2024)	*A Decision Support System for Internal Logistics using Intelligent Heuristics*	*DSS* dalam konteks logistik internal (gudang) ini menekankan efisiensi keputusan melalui heuristik cerdas. Namun, domainnya berbeda dan tidak mengakomodasi alokasi sumber daya dalam lingkungan dinamis seperti *apron* bandara. Juga tidak mengintegrasikan algoritma prediksi atau visualisasi status operasional.

2.1.2 Gap Analysis

Berdasarkan studi literatur yang telah diuraikan, dapat disimpulkan bahwa meskipun berbagai penelitian telah membahas sistem informasi *apron*, penerapan algoritma klasifikasi, maupun sistem pendukung keputusan (*DSS*), belum ada pendekatan yang menyatukan ketiganya secara terintegrasi dalam satu sistem operasional yang dapat digunakan langsung oleh petugas *AMC* di lapangan.

Mayoritas studi hanya berfokus pada aspek individual seperti digitalisasi pencatatan, optimasi jadwal, atau pengembangan *DSS* untuk domain non-penerbangan tanpa menghadirkan solusi holistik yang mencakup *pengumpulan data historis, pemrosesan otomatis, prediksi alokasi stand, hingga visualisasi interaktif yang real-time*.

Penelitian ini mencoba menjawab celah yang belum banyak dieksplorasi, yaitu integrasi model machine learning berbasis *Random Forest* ke dalam sistem operasional *AMC* secara *real-time.*

Kontribusi utama penelitian ini dibandingkan studi terdahulu:

1. Prediksi Alokasi *Stand*: Menggunakan algoritma Random Forest untuk menghasilkan prediksi probabilistik (Top-3) terhadap lokasi parkir pesawat.
2. *DSS Modular*: Sistem mendukung pengambilan keputusan berbasis data melalui pipeline klasifikasi.
3. Visualisasi Interaktif: Menyajikan hasil prediksi dalam bentuk *kartu Top-3* dengan *confidence score* dan fitur klik untuk alokasi cepat.

Pendekatan ini tidak hanya meningkatkan akurasi, tetapi juga ketahanan sistem terhadap variasi data, sebagaimana dijelaskan dalam prinsip *ensemble learning* oleh (Mishra et al., 2025).

Tabel 2.2 menunjukkan perbandingan antara pendekatan penelitian terdahulu dan kontribusi dari penelitian ini:

Tabel 2. 2 Perbandingan Penelitian Terdahulu Dengan Penelitian Ini

Dimensi	Penelitian Terdahulu	Penelitian Ini
Prediksi Alokasi *Stand*	Belum diterapkan secara eksplisit; prediksi terbatas pada jadwal slot atau belum ada komponen prediksi	Prediksi Alokasi *Stand*: Menggunakan algoritma *Random Forest ensemble* berbasis fitur historis dan *real-time*, dengan *Target Top-3 accuracy* minimal 80%
DSS Modular	*DSS* digunakan dalam domain logistik atau hanya dalam bentuk sistem informasi statis, tanpa logika inferensi prediktif	*DSS Modular: DSS* diintegrasikan secara langsung dengan pipeline *ML* berbasis Random Forest untuk mendukung pengambilan keputusan *AMC* secara adaptif dan probabilistik *(Top-K recommendation)*
Visualisasi Interaktif	Visualisasi data bersifat statis atau sekadar tabel; tidak mendukung monitoring dinamis kondisi *apron*	Visualisasi *real-time* kondisi apron, status pergerakan, dan alokasi *stand*, dengan *UI* interaktif untuk pengawasan langsung
Konteks *Apron Movement Control*	Belum ada penelitian yang menggabungkan seluruh aspek *AMC* (prediksi, *DSS*, visualisasi) dalam satu sistem	Fokus penuh pada sistem AMC di Bandara Halim dengan pendekatan sistem terintegrasi yang diuji terhadap data operasional riil

2.2 Apron Movement Control (AMC)

2.2.1 Definisi dan Konsep *AMC*

Apron Movement Control (AMC) merupakan unit operasional di lingkungan bandar udara yang bertanggung jawab atas pengaturan dan pengawasan pergerakan pesawat serta kendaraan di area *apron*. Kawasan *apron* sendiri adalah wilayah antara terminal dan landasan pacu yang digunakan untuk parkir, pemuatan dan pembongkaran, pengisian bahan bakar, dan kegiatan lainnya yang melibatkan interaksi langsung dengan pesawat.

Dalam konteks operasional, peran *AMC* seringkali disamakan dengan fungsi "polisi sisi udara", karena seluruh pergerakan baik pesawat, kendaraan pengangkut logistik, *ground handling*, maupun kendaraan pelayanan teknis harus berada dalam

koordinasi dan pengawasan unit ini. *AMC* berbeda *dari Air Traffic Control (ATC)* dalam hal cakupan tugas; *ATC* berwenang atas pergerakan pesawat di udara dan saat berada di *taxiway* atau *runway*, sedangkan *AMC* berfokus pada pengelolaan apron (Yesicatama & Widagdo, 2025).

Struktur organisasi *AMC* berada di bawah Divisi Pelayanan *Airside* dan *Landside Operation*, yang dipimpin oleh *Assistant Manager* dan secara hierarkis melapor langsung ke *EGM* (*Executive General Manager*) Operasional Bandara.

2.2.2 Fungsi dan Tanggung Jawab *AMC*

Apron Movement Control (AMC) merupakan personel bandar udara yang memiliki lisensi dan rating untuk melaksanakan pengawasan terhadap ketertiban dan keselamatan pergerakan lalu lintas di *apron*, menjamin kebersihan wilayah *apron*, serta menentukan parkir pesawat udara. Ruang lingkup kerja *AMC* mencakup berbagai aspek operasional yang krusial bagi kelancaran dan keamanan aktivitas penerbangan di sisi udara.

Tugas utama petugas *AMC* meliputi pengalokasian *parking stand* untuk pesawat yang datang dan berangkat, dengan mempertimbangkan tipe dan kapasitas pesawat yang sesuai dengan fasilitas *apron* yang tersedia. *AMC* juga bertanggung jawab dalam melakukan pencatatan data penerbangan seperti waktu *on-block* dan *off-block*, registrasi pesawat, serta penggunaan fasilitas garbarata ke dalam sistem komputer dan *movement log book* sebagai dokumentasi operasional. Fungsi pengawasan ini menuntut koordinasi yang ketat dengan berbagai unit terkait, termasuk *ATC*, *airline*, *ground handling*, serta tim keamanan bandara (Putri & Suprapti, 2022).

Selain pengelolaan pergerakan pesawat, AMC melaksanakan inspeksi rutin terhadap kebersihan dan kelayakan fasilitas di sisi udara, meliputi *apron*, *parking stand, flood light, dan hydrant pit*. Pengawasan juga dilakukan terhadap kendaraan *Ground Support Equipment (GSE)* dan personel yang beroperasi di apron untuk memastikan kepatuhan terhadap peraturan keselamatan, seperti kecepatan kendaraan, penggunaan *safety vest*, dan kepemilikan izin mengemudi di sisi udara. Komunikasi operasional dilakukan melalui radio *HT* sebagai sarana koordinasi utama antar unit, yang menggambarkan ketergantungan tinggi terhadap sistem

komunikasi verbal dalam menjaga keselamatan operasional penerbangan (Putri & Suprapti, 2022).

2.2.3 Proses Kerja Pengelolaan Pergerakan Pesawat

Alur kerja pengelolaan pergerakan pesawat oleh *AMC* dimulai dari proses *pre-arrival* hingga pesawat meninggalkan *apron*. Setiap kedatangan dicatat secara manual dengan mengisi kolom kedatangan, serta waktu ketika pesawat masuk dan keluar dari *stand*. Penugasan *parking stand* memperhatikan sejumlah faktor seperti preferensi maskapai, tipe pesawat, jadwal keberangkatan selanjutnya, serta prioritas berdasarkan urgensi (misalnya *flight VIP* atau evakuasi medis).

Sebagai ilustrasi, dalam satu kejadian selama magang, dua penerbangan dijadwalkan menggunakan stand yang sama (*Citilink* dan Batik *Air*). Berdasarkan analisis waktu *turnaround*, keputusan diambil untuk mengalokasikan *Citilink* ke stand A3 karena memiliki waktu *turnaround* lebih singkat, sementara Batik *Air* dipindahkan ke stand B7.

2.2.4 Karakteristik Operasional Bandara Halim Perdanakusuma

Bandara Halim Perdanakusuma memiliki profil operasional yang unik karena menangani beragam kategori penerbangan, termasuk reguler, *charter*, *RON (Remain Overnight)*, dan Militer. Struktur *apron* terbagi atas beberapa zona, dengan konfigurasi *stand* yang bervariasi dalam kapasitas pesawat narrow-body dan *wide-body.* Kompleksitas lalu lintas tinggi sering kali menciptakan bottleneck saat *peak hour*, khususnya ketika *RON* dan charter saling tumpang tindih.

Tantangan operasional juga diperparah oleh belum adanya sistem visualisasi kondisi apron secara *real-time*, yang menyulitkan petugas *AMC* dalam merespons cepat perubahan dinamika di lapangan

2.3 Sistem Informasi

2.3.1 Definisi Sistem Informasi

Sistem informasi dalam konteks organisasi didefinisikan sebagai salah satu komponen terpenting yang menjadi fondasi operasional di era digital. (Marsa et al., 2023) menyatakan bahwa *"Sistem informasi adalah komponen penting dalam organisasi modern, yang memungkinkan mereka mengumpulkan, menyimpan,*

memproses, dan menyebarkan informasi untuk berbagai tujuan". Definisi ini menegaskan peran sentral sistem informasi dalam mengelola informasi sehingga dapat digunakan untuk memenuhi beragam kebutuhan dan pencapaian tujuan di dalam organisasi secara efektif.

2.3.2 Komponen Sistem Informasi

Untuk memahami sistem informasi secara menyeluruh, perlu diketahui komponen-komponen utama yang membangunnya. Berdasarkan penjelasan (Marsa et al., 2023), sistem informasi tersusun atas empat komponen inti yang saling berinteraksi, yaitu *"teknologi, data, manusia, dan proses berinteraksi dalam ekosistem sistem informasi yang kompleks"*. Komponen teknologi mencakup perangkat keras dan perangkat lunak yang digunakan; data mencakup semua fakta dan informasi yang diolah; manusia mencakup pengguna, analis, hingga pengambil keputusan yang terlibat; dan proses mencakup prosedur atau langkah-langkah operasi bisnis. Keempat komponen ini bekerja secara terpadu, dimana teknologi digunakan untuk mengolah data, manusia berperan menjalankan dan mengatur sistem, serta proses-proses bisnis ditopang oleh teknologi dan informasi yang dihasilkan sistem tersebut. Interaksi komponen-komponen tersebut membentuk satu kesatuan sistem informasi yang mampu mendukung fungsi organisasi secara kompleks dan terintegrasi.

2.3.3 Manfaat Sistem Informasi

Implementasi sistem informasi di dalam organisasi memberikan berbagai manfaat penting. Secara umum, sistem informasi berperan dalam *"mendukung operasi organisasi, pengambilan keputusan, dan pencapaian tujuan bisnis"*. Artinya, sistem informasi membantu kelancaran operasional dengan memastikan data dan informasi yang dibutuhkan tersedia tepat waktu untuk berbagai proses bisnis harian. Selain itu, sistem informasi mendukung pengambilan keputusan dengan menyediakan informasi yang relevan dan akurat bagi manajer maupun eksekutif, sehingga keputusan dapat diambil secara lebih cepat dan tepat berdasarkan fakta. Lebih lanjut, sistem informasi berkontribusi pada pencapaian tujuan strategis organisasi dengan menyelaraskan informasi dan teknologi terhadap

sasaran bisnis, misalnya melalui pemantauan kinerja dan penyediaan laporan yang membantu perencanaan strategis. Dengan demikian, keberadaan sistem informasi yang efektif memungkinkan organisasi meningkatkan efisiensi operasional, kualitas layanan, dan adaptabilitas dalam menghadapi perubahan, sekaligus memastikan tujuan-tujuan bisnis dapat tercapai secara lebih optimal.

2.4 Decision Support System (DSS)

2.4.1 Konsep Decision Support System (DSS)

Decision Support System (DSS), atau dikenal dalam bahasa Indonesia sebagai Sistem Penunjang Keputusan (SPK), merupakan sistem berbasis komputer yang dirancang untuk membantu proses pengambilan keputusan, terutama dalam situasi kompleks yang melibatkan banyak variabel dan ketidakpastian. Sistem ini mengintegrasikan data historis, pemodelan analitik, serta antarmuka pengguna yang interaktif untuk menghasilkan rekomendasi yang dapat mendukung pengambilan keputusan secara objektif. (Insaurralde & Blasch, 2024)

Dalam studi terbaru oleh (Gui et al., 2024), *DSS* untuk sektor transportasi udara dikembangkan sebagai kerangka tiga modul yang saling terintegrasi: (1) generasi trajektori penurunan berdasarkan strategi seperti *Step-Down Descent (SDO), Continuous Descent Operation (CDO),* dan *path-stretching CDO (ps-CDO);* (2) penjadwalan kedatangan pesawat menggunakan pemrograman integer campuran dan algoritma *Variable Neighborhood Search (VNS);* serta (3) pemilihan trajektori optimal berdasarkan kombinasi biaya keterlambatan dan konsumsi bahan bakar.

Pendekatan ini memanfaatkan data performa pesawat dan kondisi cuaca untuk menghasilkan trajektori yang sesuai, sekaligus memperhitungkan efisiensi bahan bakar dan konflik lalu lintas dalam pengambilan keputusan oleh pengendali lalu lintas udara (*ATC*). Dengan demikian, modul analitik menyediakan alat pemodelan canggih, sementara antarmuka pengguna mendukung skenario pengambilan keputusan real-time di lapangan**.** (Insaurralde & Blasch, 2024) menekankan pentingnya penggunaan model prediktif dalam *DSS* untuk mendukung keputusan waktu nyata *(real-time)* di lingkungan bandara. Mereka juga menyoroti bagaimana

DSS memungkinkan perencanaan skenario "*what-if*" guna mengantisipasi berbagai situasi operasional.

DSS dapat diklasifikasikan menjadi beberapa kategori tergantung pendekatannya:

1. *Data-driven DSS*: Menitikberatkan pada eksplorasi dan analisis data dalam jumlah besar.
2. *Model-driven DSS*: Memanfaatkan pemodelan matematis/statistik sebagai dasar pengambilan keputusan.
3. *Hybrid DSS*: Menggabungkan kekuatan data dan model untuk mendukung analisis yang lebih kompleks dan adaptif.

Dalam konteks penelitian ini, *DSS* berperan sebagai landasan arsitektural sistem *AMC* berbasis *Machine Learning*, karena fungsinya yang mampu menyatukan elemen data historis, algoritma prediksi, dan visualisasi operasional untuk pengambilan keputusan alokasi parking stand secara efisien.

2.4.2 DSS dalam Pengambilan Keputusan Operasional

Dalam konteks pengambilan keputusan operasional, model keputusan yang sering digunakan adalah *Intelligence–Design–Choice (IDC)* yang dikembangkan oleh Herbert Simon. Model ini menjelaskan proses keputusan sebagai serangkaian tahapan: identifikasi masalah (*intelligence*), perancangan alternatif solusi (*design*), dan pemilihan solusi terbaik (*choice*).

Seiring dengan berkembangnya kompleksitas lalu lintas udara, (Gui et al., 2024) menunjukkan bahwa penerapan *DSS* dalam pengelolaan kedatangan pesawat di *Terminal Maneuvering Area (TMA)* telah mampu menggabungkan efisiensi waktu dan bahan bakar melalui penjadwalan trajektori yang fleksibel namun aman. Dengan mempertimbangkan trajektori optimal dan jadwal kedatangan berbasis waktu dan konflik, *DSS* dapat mendukung keputusan operasional yang adaptif terhadap batasan kapasitas dan kepadatan lalu lintas. Selain itu, pendekatan ini memungkinkan *ATC* melakukan negosiasi waktu kedatangan dengan pesawat secara kolaboratif, sekaligus mempertahankan efisiensi operasional.

Integrasi *DSS* berbasis *data mining* juga telah dimanfaatkan untuk prediksi permintaan, optimalisasi parkir, dan penjadwalan pesawat secara otomatis

(Insaurralde & Blasch, 2024). Hal ini menunjukkan bahwa sistem seperti *DSS* dapat menjadi fondasi teknis dari sistem *AMC* berbasis *machine learning* yang diusulkan dalam penelitian ini.

2.5 Knowledge Discovery in Database (KDD)

2.5.1 Definisi dan Tahapan *KDD*

Knowledge Discovery in Database (*KDD*) merupakan suatu proses sistematis untuk mengekstraksi pengetahuan bermakna dari kumpulan data yang besar. *KDD* mencakup seluruh tahapan mulai dari preprocessing hingga evaluasi hasil prediksi. Dalam penelitian ini, tahapan *KDD* digunakan untuk membangun sistem prediksi alokasi *stand* berbasis algoritma *Random Forest*.

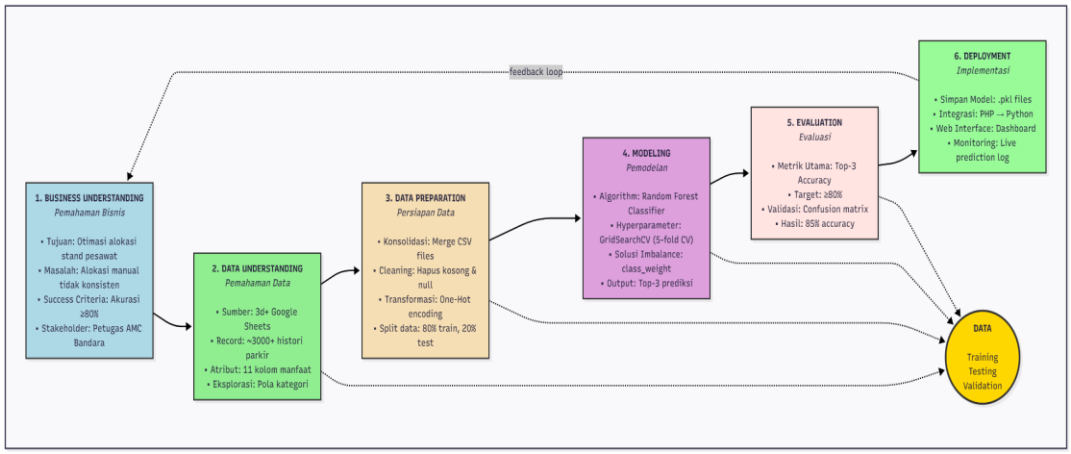
Menurut (Mishra et al., 2025), pendekatan *ensemble* seperti *Random Forest* sangat efektif dalam tahapan data mining karena mampu menangani dataset besar, fitur kompleks, dan *class imbalance*. Seluruh tahapan KDD yang diterapkan meliputi:

1. Seleksi Data: Pemilihan fitur-fitur penting seperti tipe pesawat, maskapai, dan kategori penerbangan.
2. Preprocessing: Normalisasi dan *encoding* atribut, penanganan nilai hilang, serta rekayasa fitur kombinasi.
3. Transformasi: Konversi fitur ke format numerik untuk digunakan dalam pelatihan model.
4. Data Mining: Pelatihan model klasifikasi menggunakan *Random Forest* dengan konfigurasi 100 *estimator* dan parameter *class_weight='balanced_subsample'.*
5. Evaluasi: Pengujian model menggunakan metrik akurasi, *precision, recall*, dan khususnya *Top-K Accuracy (Top-3).*
6. Model *Random Forest* yang akan digunakan dalam penelitian ini ditargetkan mencapai akurasi prediksi *Top-3* minimal 80%, yang menunjukkan kinerja andal dalam lingkungan operasional dengan banyak kelas target.

2.5.2 *CRISP-DM* sebagai *Framework* Penelitian

Framework *CRISP-DM (Cross-Industry Standard Process for Data Mining)* menjadi landasan dalam merancang sistem prediktif ini secara sistematis. Dengan enam tahapan utama, pendekatan *CRISP-DM* membantu memastikan proses terstruktur dari pemahaman masalah hingga implementasi model.

1. *Business Understanding*: Sistem ini ditujukan untuk meningkatkan efisiensi pengalokasian *stand* di *apron* Bandara Halim.
2. *Data Understanding*: Data dikumpulkan dari dokumentasi harian *AMC* dan dikonsolidasikan dalam *dataset* operasional.
3. *Data Preparation*: Meliputi *encoding* atribut seperti tipe pesawat dan maskapai, serta pembersihan data tidak *valid*.
4. *Modeling*: Menggunakan algoritma *Random Forest*, yang memiliki keunggulan dalam menangani data multi-kelas dan *class imbalance*.
5. *Evaluation*: Evaluasi dilakukan dengan metrik *Top-1* dan *Top-3 Accuracy* untuk mengukur efektivitas model dalam konteks operasional.
6. *Deployment*: Model disimpan dalam format .pkl dan diintegrasikan dengan backend *PHP* melalui *script Python* untuk digunakan oleh operator *AMC* secara real-time.



Gambar 2. 1 Diagram Alur *CRISP-DM*

Diagram ini akan mencerminkan tahapan berikut:

1. *Business Understanding*
2. *Data Understanding*
3. *Data Preparation*
4. *Modeling*
5. *Evaluation*
6. *Deployment*

2.6 *Data Mining* dan Konsep Klasifikasi

2.6.1 Pengertian *Data Mining*

Data mining adalah proses untuk menemukan pola-pola yang sebelumnya tidak dikenal dalam kumpulan data yang besar melalui analisis statistik, pembelajaran mesin, dan teknik pemodelan lainnya. Proses ini merupakan inti dari *Knowledge Discovery in Database (KDD)* dan berperan penting dalam sistem informasi yang membutuhkan pengambilan keputusan berbasis data.

Menurut (Safira et al., 2024), data mining memungkinkan organisasi untuk menggali informasi tersembunyi dari data historis dan memanfaatkannya dalam bentuk prediksi atau klasifikasi. Dalam konteks sistem informasi *Apron Movement Control (AMC),* data mining digunakan untuk mempelajari pola historis pergerakan pesawat dan mengklasifikasikan data input ke dalam alokasi *stand* tertentu secara otomatis dan berbasis model.

2.6.2 Konsep Klasifikasi

Klasifikasi adalah teknik *supervised learning* yang memetakan data input ke dalam salah satu dari beberapa kelas yang telah ditentukan. Dalam konteks ini, klasifikasi multi-kelas digunakan untuk memprediksi lokasi parkir pesawat (stand) berdasarkan atribut input.

Model *Random Forest* menghasilkan distribusi probabilitas terhadap semua kelas, yang memungkinkan penerapan *Top-K prediction* yaitu pengambilan K besar kemungkinan kelas sebagai kandidat. Pendekatan ini sangat berguna ketika ketepatan satu-pilihan (*Top-1*) tidak cukup, dan sistem dapat menawarkan opsi *Top-3* terbaik.

(Petersen et al., 2022) menunjukkan bahwa strategi *Top-K* dapat meningkatkan efisiensi sistem klasifikasi dalam aplikasi nyata, khususnya di lingkungan dinamis seperti pengelolaan apron.

2.6.3 Evaluasi Model Klasifikasi

Untuk menilai kinerja model klasifikasi, digunakan berbagai metrik evaluasi berbasis confusion matrix, yang mengklasifikasikan hasil prediksi ke dalam empat kategori:

1. *True Positive* (TP): Prediksi benar untuk kelas positif
2. *True Negative* (TN): Prediksi benar untuk kelas negatif
3. *False Positive* (FP): Prediksi salah, model mengira positif padahal negatif
4. *False Negative* (FN): Prediksi salah, model mengira negatif padahal positif

Berdasarkan confusion matrix, metrik evaluasi utama yang digunakan antara lain:

1. ***Accuracy***

   Mengukur proporsi prediksi yang benar dari seluruh data:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2. ***Precision***

   Mengukur ketepatan prediksi positif:

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. ***Recall (Sensitivity)***

   Mengukur seberapa banyak data positif yang berhasil dikenali:

$$\text{Recall} = \frac{TP}{TP + FN}$$

4. ***F1 Score***

   Rata-rata harmonis antara *precision* dan *recall*:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. ***Top-K Accuracy***

   Mengukur apakah label aktual termasuk dalam prediksi teratas sejumlah *k* besar. Umumnya digunakan untuk klasifikasi multi-kelas.

$$\text{Top-}k\text{ Accuracy} = \frac{\text{Jumlah instance di mana label benar ada dalam Top-}k}{\text{Total instance}}$$

Dalam konteks sistem *AMC*, metrik akurasi menjadi indikator utama karena setiap prediksi *stand* yang salah dapat berdampak terhadap efisiensi pergerakan pesawat. Namun, metrik lain seperti *precision* dan *recall* juga dipertimbangkan untuk menilai potensi kesalahan dalam kondisi operasional dinamis.

2.7 Algoritma Random Forest

2.7.1 Konsep Dasar Random Forest

Random Forest adalah algoritma *ensemble learning* berbasis pohon keputusan yang membangun banyak *decision tree* secara paralel dan melakukan prediksi berdasarkan agregasi hasil dari semua pohon (*voting* untuk klasifikasi). Algoritma ini pertama kali diperkenalkan oleh Leo Breiman pada tahun 2001 sebagai pengembangan dari metode bagging sebelumnya. Pendekatan ensemble ini dirancang untuk memperbaiki kelemahan dari *single decision tree* seperti *overfitting*, *high variance*, dan instabilitas struktur pohon (Ibrahim, 2023; Mishra et al., 2025).

Dalam perkembangannya, *Random Forest* telah menjadi salah satu algoritma yang paling banyak digunakan dalam berbagai aplikasi klasifikasi dan regresi. (Ibrahim, 2023) menjelaskan bahwa setiap *decision tree* dalam *Random Forest* dilatih pada subset acak dari *data training* (dengan pengembalian), dan pada setiap *node*, pemilihan atribut juga dilakukan secara acak dari subset fitur. Kombinasi strategi ini menghasilkan model yang lebih *robust* dan memiliki *generalizability* lebih baik dibandingkan *decision tree* tunggal seperti *J48*.

Penelitian oleh (Mishra et al., 2025) menegaskan bahwa keunggulan utama *Random Forest* dalam konteks penelitian ini meliputi:

1. Reduksi *Variance*: Dengan mengagregasi prediksi dari banyak pohon yang independen, *Random Forest* secara signifikan mengurangi *variance* tanpa

meningkatkan bias. Hal ini membuat model lebih stabil dalam menghadapi variasi data baru.

2. Handling *Class Imbalance*: *Parameter class_weight='balanced_subsample'* memungkinkan model menangani ketidakseimbangan distribusi kelas *parking stand* secara otomatis, yang sangat relevan untuk *dataset* operasional *AMC* di mana beberapa *stand* lebih sering digunakan dibanding yang lain.
3. *Feature Importance*: *Random Forest* dapat mengidentifikasi fitur-fitur yang paling berkontribusi terhadap prediksi, memberikan wawasan tentang faktor-faktor kunci dalam alokasi *stand* seperti tipe pesawat, maskapai, dan kategori penerbangan.
4. *Probabilistic Output*: Model menghasilkan distribusi probabilitas untuk setiap kelas, memungkinkan implementasi strategi *Top-K prediction* yang fleksibel. Dalam konteks sistem *AMC*, output probabilistik ini memungkinkan operator memilih dari beberapa alternatif stand berdasarkan *confidence score* yang dihasilkan model.

Studi komparatif yang dilakukan oleh (Endut et al., 2022) menunjukkan bahwa *Random Forest* unggul dalam tugas *multi-label classification* dibandingkan algoritma tunggal seperti *Support Vector Machine* atau *Logistic Regression,* terutama dalam hal stabilitas prediksi dan kemampuan generalisasi pada data dengan banyak kelas target.

2.7.2 Proses Konstruksi *Random Forest*

Random Forest membangun ensemble dari multiple decision trees melalui proses berikut:

Tahap 1: *Bootstrap Sampling (Bagging)*

Untuk setiap pohon ke-*t* dalam *forest*:

1. Ambil sampel *bootstrap* dari *dataset training* (*sampling* dengan pengembalian)
2. Sekitar 63% data original akan terpilih, sisanya menjadi *Out-of-Bag (OOB) samples*
3. Setiap pohon dilatih pada *bootstrap* sample yang berbeda, menciptakan *diversity*

Secara matematis, jika dataset asli memiliki *N instances*, probabilitas sebuah instance tidak terpilih dalam satu kali sampling adalah:

$$P(\text{tidak terpilih}) = \left(1 - \frac{1}{N}\right)^{N} \rightarrow e^{-1} \approx 0.368 \quad (\text{untuk } N \text{ besar})$$

Ini berarti sekitar 36.8% data menjadi *OOB samples* yang dapat digunakan untuk evaluasi internal tanpa perlu validation set terpisah.

Tahap 2: *Random Feature Selection*

Pada setiap *node* internal saat membangun pohon:

1. Pilih *subset* acak dari *m* fitur (dari total *M* fitur)
2. Cari *split* terbaik hanya di antara *m* fitur tersebut menggunakan kriteria Gini *impurity* atau *entropy*
3. Nilai *m* yang umum digunakan: $\sqrt{M}$ untuk klasifikasi

Proses ini disebut random *subspace method* dan menciptakan *decorrelation* antar *pohon* dalam *forest*

Tahap 3: *Tree Growing*

Setiap pohon tumbuh hingga:

1. Mencapai kedalaman maksimum *(max_depth)* jika dibatasi, atau
2. *Node* murni (semua *instances* dalam *node* memiliki kelas sama), atau
3. Jumlah samples di *node* < *min_samples_split*

Tidak ada pruning dilakukan pada *individual trees* (berbeda dengan *C4.5/J48*).

Tahap 4: *Aggregation (Voting)*

Untuk prediksi klasifikasi:

1. Setiap *tree* memberikan satu *vote* untuk kelas yang diprediksinya
2. Kelas *final* ditentukan melalui *majority voting*
3. Probabilitas kelas *c* dihitung sebagai:

$$P(\text{class} = c) = \frac{\text{Jumlah trees yang vote untuk } c}{\text{Total jumlah trees}}$$

Dalam penelitian ini, dengan *100 trees (n_estimators=100)*, jika *45 trees* memprediksi stand A1, maka *P(A1) = 45/100 = 0.45.*

2.7.3 Perbandingan dengan *Decision Tree* Tunggal

Untuk memahami mengapa *Random Forest* dipilih dalam penelitian ini, Tabel 2.3 menunjukkan perbandingan antara *single decision tree* (contoh: algoritma *J48*) dengan pendekatan *ensemble Random Forest*

Tabel 2. 3 Perbandingan *Decision Tree* dengan *Random Forest*

Aspek	*Decision Tree (J48)*	*Random Forest*
Arsitektur	Single tree	Ensemble (ratusan pohon)
Variance	Tinggi (*overfit*)	Rendah (hasil *voting* banyak pohon)
Bias	Rendah	Rendah
Stabilitas	Rentan perubahan kecil	Lebih stabil
Imbalance Handling	Tidak otomatis	*class_weight='balanced_subsample'*
Feature Importance	Perspektif tunggal	Agregat dari banyak pohon

2.7.4 Justifikasi Matematika

Keunggulan *Random Forest* dapat dijelaskan melalui analisis *bias-variance tradeoff*. Jika sebuah *decision tree* tunggal memiliki *variance* sebesar σ^2, maka *ensemble Random Forest* dengan T pohon yang tidak berkorelasi akan memiliki *variance* sebesar**:**

$$\mathrm{Var}_{RF} = \frac{\sigma^2}{T}$$

Ini menunjukkan bahwa dengan jumlah pohon cukup besar, variasi hasil prediksi berkurang secara signifikan, menghasilkan model yang lebih akurat dan stabil. Namun dalam praktik, pohon-pohon dalam *forest* memiliki korelasi ρ (rho) karena dilatih pada data yang berasal dari distribusi sama. Dalam penelitian seminal tentang *Random Forest*, Breiman (2001) menunjukkan bahwa *generalization error* dari *Random Forest* memiliki *upper bound*:

$$PE^* \leq \rho \times \frac{(1 - s^2)}{s^2}$$

dimana:

1. ρ = korelasi rata-rata antar pohon dalam *forest*
2. s = *strength* (kekuatan klasifikasi *individual trees*)

Formula ini mengungkapkan dua prinsip kunci untuk meminimalkan *error*:

1. Mengurangi korelasi antar pohon (melalui random feature selection) → menurunkan *error*
2. Meningkatkan *strength individual trees* → menurunkan error

Penelitian lebih lanjut oleh (Mobley et al., 2021) mengonfirmasi bahwa strategi ensemble berbasis *Random Forest* mampu mencapai keseimbangan optimal antara diversity dan *strength* melalui tiga mekanisme utama:

1. *Bootstrap sampling*: Menciptakan *training sets* yang berbeda untuk setiap pohon, mengurangi korelasi antar model
2. Random feature selection: Memaksa *trees* menggunakan *splits* yang berbeda pada setiap *node*, mengurangi korelasi lebih lanjut
3. *Full tree growth (no pruning)*: Memaksimalkan strength individual trees tanpa mengorbankan kemampuan prediksi

(Ibrahim, 2023)dalam penelitiannya tentang *lead time prediction* menunjukkan bahwa konfigurasi *hyperparameter* seperti jumlah *estimator*, kedalaman pohon, dan *minimum samples per split* sangat berpengaruh terhadap performa akhir model *Random Forest* dalam konteks *decision support system*.

Dalam penelitian ini, dengan *100 estimators, max_depth=None (full growth),* dan *min_samples_split=5*, model *Random Forest* yang akan dibangun diharapkan mencapai keseimbangan *optimal* antara *diversity (low correlation)* dan *strength*, sesuai dengan prinsip-prinsip yang telah terbukti efektif dalam literatur terkini.

2.7.5 Prediksi Multi-Kelas dan Top-K

Random Forest sangat efektif untuk tugas *multi-class classification*, karena setiap pohon memberikan vote untuk salah satu kelas. Dalam penelitian ini, terdapat 20 kelas *parking stand* (A1, A2, A3, ..., B7, dst.) yang harus diprediksi berdasarkan fitur input.

1. Strategi *Top-K Prediction*

Dalam skenario *Top-K Prediction*, model tidak hanya memberikan satu prediksi, tetapi memberikan *K* kelas teratas dengan probabilitas tertinggi sebagai rekomendasi (Petersen et al., 2022). Strategi ini sangat relevan untuk sistem *AMC* karena:

a. Fleksibilitas Operasional: Operator dapat memilih dari 3 alternatif *stand* terbaik berdasarkan kondisi *real-time* (misalnya, stand rekomendasi #1 sedang *maintenance*)
b. *Handling Uncertainty*: Jika probabilitas *top-1* tidak jauh berbeda dengan *top-2*, operator memiliki informasi untuk mempertimbangkan opsi alternatif
c. Evaluasi yang Lebih Realistis: *Top-3 accuracy* lebih mencerminkan kegunaan praktis sistem dibanding *Top-1 accuracy*

2. Implementasi *Top-K* dalam *Random Forest*

Setelah 100 pohon memberikan votes:

a. Hitung probabilitas setiap kelas: *P(class_i) = votes_i / 100*
b. Urutkan kelas berdasarkan probabilitas (*descending*)
c. Ambil 3 kelas teratas sebagai rekomendasi *Top-3*

Contoh Output:

Input*: Aircraft=B738, Airline=Citilink, Category=Domestic,*

Prediksi Top-3:

a. Stand A3 (P=0.45) ← 45 trees vote for A3
b. Stand A2 (P=0.23) ← 23 trees vote for A2
c. Stand B7 (P=0.15) ← 15 trees vote for B7

2.8 *Dataset* dalam *Machine Learning*

2.8.1 Definisi *Dataset*

Dataset dapat didefinisikan sebagai suatu sumber data yang memuat sekumpulan informasi terstruktur yang diimpor untuk tujuan analisis dalam konteks penambangan data (*data mining*). Informasi ini umumnya mencakup beragam karakteristik, atribut, dan variabel yang relevan dengan domain permasalahan yang

diteliti. Sebelum dapat digunakan secara efektif, dataset seringkali memerlukan tahap pra-pemrosesan (*pre-processing*), yang meliputi pembersihan data untuk mengoreksi kesalahan, menangani nilai yang hilang (*missing values*), serta menghilangkan karakter-karakter asing atau tidak relevan, guna mempersiapkan data untuk proses analisis lebih lanjut (Andrade-Arenas et al., 2024).

Kualitas data memegang peranan penting dalam data mining karena data mentah umumnya mengandung *missing values*, karakter yang tidak relevan, duplikasi, dan *noise* yang dapat menurunkan akurasi analisis. Oleh sebab itu, proses *data cleaning* seperti menghapus nilai hilang, menghilangkan duplikasi, dan memilih atribut yang relevan diwajibkan agar data menjadi lebih bersih, reliabel, dan siap digunakan untuk pemodelan (Andrade-Arenas et al., 2024).

Dalam konteks *data mining*, dataset dipahami sebagai kumpulan objek yang direpresentasikan melalui atribut tertentu, di mana setiap objek dapat dinyatakan sebagai *real-valued vectors* berdimensi d sesuai karakteristiknya (Andrade-Arenas et al., 2024). *Dataset* ini kemudian diproses melalui berbagai tahapan seperti seleksi atribut, pembersihan nilai hilang, dan penyaringan contoh untuk memastikan hanya data yang relevan dan berkualitas yang. Setelah dataset dibersihkan dan siap dipakai, tahap berikutnya adalah menyiapkannya untuk proses pemodelan melalui pembagian data ke dalam subset pelatihan dan pengujian *(Train-test Split)*

Train–test split merupakan langkah penting dalam persiapan data, yaitu proses membagi dataset ke dalam dua subset dengan ukuran relatif tertentu untuk keperluan pelatihan dan pengujian model. Pembagian ini memastikan bahwa model dilatih dan dievaluasi pada data yang berbeda sehingga hasil evaluasi bersifat objektif dan mencerminkan kemampuan generalisasi model terhadap data baru.

2.9 Metodologi Penelitian

Penelitian ini menggunakan pendekatan kuantitatif untuk menghasilkan analisis yang terukur dan objektif terhadap sistem rekomendasi parking stand berbasis *Random Forest*. Metode kuantitatif dipilih karena sifat penelitian yang berfokus pada pengukuran kinerja model *machine learning* melalui eksperimen sistematis dengan metrik evaluasi yang dapat dikuantifikasi.

2.9.1 Metode Kuantitatif

Aspek kuantitatif penelitian ini akan berfokus pada pengukuran kinerja model *Random Forest* melalui eksperimen sistematis dengan metrik evaluasi yang terukur. Data historis operasional *parking stand* dari Bandar Udara Halim Perdanakusuma akan digunakan sebagai *dataset* penelitian. Eksperimen akan dilakukan menggunakan metode *Grid Search* untuk optimasi *hyperparameter*, dengan metrik evaluasi utama berupa *Top-3 Accuracy* yang ditargetkan mencapai minimal 80% dan *response time* sistem yang ditargetkan maksimal 5 detik.

Pendekatan kuantitatif ini akan memungkinkan validasi objektif terhadap performa sistem dan membuktikan bahwa algoritma *Random Forest* mampu memberikan rekomendasi *parking stand* dengan akurasi tinggi sesuai target penelitian. Pengukuran akan dilakukan melalui *confusion matrix* untuk menganalisis pola kesalahan prediksi per kelas stand, *classification report* untuk mengevaluasi *precision* dan *recall* setiap *stand*, serta analisis *feature importance* untuk mengidentifikasi variabel yang paling berpengaruh terhadap keputusan alokasi.

Dataset akan dibagi menggunakan teknik *stratified train-test split* dengan rasio 80:20 untuk memastikan distribusi kelas yang proporsional antara *data training* dan *testing*. Proses *Grid Search* akan menggunakan *5-fold cross-validation* untuk menguji kombinasi *hyperparameter* secara sistematis, memastikan bahwa model yang dihasilkan memiliki generalisasi yang baik dan tidak mengalami overfitting pada data training.

Metode kuantitatif dalam penelitian ini sejalan dengan prinsip *practice-based research*, di mana sistem dikembangkan berdasarkan kebutuhan nyata operasional dan dievaluasi melalui metrik kinerja yang terukur (Darono, 2025). Pendekatan ini memastikan bahwa *Decision Support System* yang dihasilkan tidak hanya unggul secara algoritmik, tetapi juga memberikan *actionable insights* yang dapat langsung diterapkan oleh petugas AMC dalam pengambilan keputusan alokasi *parking stand* sehari-hari.

2.10 Metode Pengembangan Sistem

Pengembangan sistem rekomendasi parking stand dalam penelitian ini akan menggunakan metodologi *AGILE* dengan kerangka kerja *Scrum*. Pemilihan metodologi *AGILE* didasarkan pada karakteristik proyek yang membutuhkan fleksibilitas tinggi terhadap perubahan requirement serta kebutuhan akan feedback berkelanjutan dari pengguna akhir (operator *AMC*).(diagram metode *agile*)

2.10.1 Justifikasi Pemilihan *AGILE*

Scrum merupakan proses rekayasa perangkat lunak berulang yang bersifat *lightweight* namun efektif, dengan fokus pada kolaborasi tim dan pengiriman produk secara bertahap sambil meminimalisir pemborosan waktu dan tenaga (Hidayah et al., 2022). Dalam konteks penelitian ini, metodologi *AGILE* akan dipilih karena beberapa alasan utama:

1. Perubahan *Requirement* secara Iteratif

   Selama masa pengembangan sistem, diperkirakan akan terjadi penambahan fitur yang tidak direncanakan di awal proyek, seperti penambahan *dashboard metrics* yang menampilkan *daily movements counter* per kategori pesawat atau fitur *snapshot* harian untuk dokumentasi kondisi apron. Karakteristik *AGILE* yang adaptif terhadap perubahan akan memungkinkan fitur-fitur baru ini diintegrasikan tanpa mengganggu struktur sistem yang telah ada.

2. *Feedback Loop* Berkelanjutan

   Pengembangan sistem akan dilakukan secara iteratif: setiap fitur akan dikembangkan, diujicoba di *localhost*, kemudian ditunjukkan kepada operator *AMC* untuk mendapatkan *feedback*. Sebagai contoh, pada fitur *Apron Map*, *feedback* dari operator dapat menyarankan agar semua *apron* dikompilasi dalam satu *layout map* terpadu, bukan terpisah per *apron*. *Feedback* semacam ini akan segera diterapkan pada iterasi berikutnya, sesuai dengan prinsip *Scrum* yang menekankan *continuous improvement* berdasarkan masukan *stakeholder*.

3. Pengembangan Inkremental

   Sistem akan dikembangkan dalam *sprint-sprint* kecil, di mana setiap sprint akan menghasilkan *increment* berupa fitur yang dapat diuji secara fungsional.

Pendekatan ini terbukti meningkatkan produktivitas tim dan memastikan setiap komponen sistem berfungsi dengan baik sebelum dilanjutkan ke fitur berikutnya

2.10.2 Implementasi Scrum dalam Penelitian

Meskipun penelitian ini merupakan proyek individual (tidak menggunakan *full Scrum team*), prinsip-prinsip Scrum akan tetap diterapkan dengan adaptasi sesuai konteks:

1. *Product Owner*: *Supervisor* magang dan operator *AMC* yang memberikan *requirement*
2. dan *feedback*
3. *Developer*: Peneliti yang bertindak sebagai *full-stack developer*
4. *Sprint Planning*: Akan dilakukan setiap minggu untuk menentukan fitur yang akan
5. dikembangkan
6. *Sprint Review*: Akan dilakukan setiap kali fitur selesai, dengan
7. mendemonstrasikan kepada operator untuk mendapat *feedback*
8. *Sprint Retrospective*: Evaluasi internal untuk memperbaiki proses *development*
9. di *sprint* berikutnya

Pendekatan ini akan memastikan bahwa sistem yang dikembangkan sesuai dengan kebutuhan operasional nyata di lapangan, dengan kualitas yang terjaga melalui iterasi dan testing berkelanjutan.

2.11 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan visual standar yang digunakan untuk menspesifikasikan, memvisualisasikan, mengkonstruksi, dan mendokumentasikan arsitektur sistem perangkat lunak (Zikra et al., 2025). *UML* menyediakan seperangkat diagram yang membantu *developer* dan *stakeholder* memahami struktur dan perilaku sistem yang akan dikembangkan. Dalam penelitian ini, *UML* akan digunakan untuk merancang arsitektur sistem rekomendasi parking stand secara terstruktur sebelum tahap implementasi.

2.11.1 *Use Case Diagram*

Use Case Diagram merupakan diagram yang menggambarkan interaksi antara *actor* (pengguna atau sistem eksternal) dengan *use case* (fungsionalitas atau layanan yang disediakan sistem). Diagram ini memberikan gambaran tingkat tinggi tentang apa yang dapat dilakukan oleh setiap jenis pengguna dalam sistem tanpa menunjukkan detail implementasi teknis. Komponen utama *Use Case Diagram* meliputi:

1. Actor: Representasi pengguna atau sistem eksternal yang berinteraksi dengan sistem. *Actor* digambarkan sebagai *stick figure* dan ditempatkan di luar *boundary sistem*.
2. *Use Case*: Fungsionalitas atau layanan yang disediakan sistem kepada *actor*. *Use case* digambarkan sebagai *oval* dan ditempatkan di dalam *boundary* sistem.
3. *Relationship*: Hubungan antara actor dengan *use case* (*association*), antar *use case (include, extend),* atau antar *actor (generalization).*
4. *System Boundary*: Batasan sistem yang menunjukkan *scope* dari sistem yang dimodelkan, digambarkan sebagai *rectangle* yang mengelilingi semua use *case.*

Use Case Diagram sangat berguna untuk mendokumentasikan requirement fungsional sistem dan memfasilitasi komunikasi antara *developer* dengan *stakeholder* non-teknis. Diagram ini menjawab pertanyaan: "Siapa yang akan menggunakan sistem dan apa yang dapat mereka lakukan?"

2.11.2 *Activity Diagram*

Activity Diagram menggambarkan alur kerja (*workflow*) atau proses bisnis dalam sistem. Diagram ini menunjukkan urutan aktivitas dari satu proses ke proses lainnya, termasuk *decision* points (percabangan logika), *parallel processes* (aktivitas yang berjalan simultan), dan *synchronization points*. *Activity Diagram* mirip dengan *flowchart* tradisional namun lebih ekspresif dalam menangani *concurrency* dan *object flow*. Elemen utama *Activity* Diagram meliputi:

1. *Initial Node*: Titik mulai alur aktivitas, digambarkan sebagai *filled circle*.

2. *Activity*: Langkah atau aksi yang dilakukan dalam proses, digambarkan sebagai *rounded rectangle*.
3. *Decision Node*: Percabangan berdasarkan kondisi *boolean* (*if-else*), digambarkan sebagai *diamond*. Satu input menghasilkan dua atau lebih output berdasarkan *guard condition*.
4. *Merge Node*: Penggabungan beberapa alur menjadi satu alur, digambarkan sebagai *diamond* dengan *multiple inputs* dan single output.
5. *Fork Node:* Pemisahan satu alur menjadi beberapa alur parallel yang berjalan secara bersamaan, digambarkan sebagai *thick horizontal* atau *vertical bar*.
6. *Join Node*: Sinkronisasi beberapa alur *parallel* menjadi satu alur, digambarkan sebagai *thick bar*. Semua alur parallel harus selesai sebelum melanjutkan ke aktivitas berikutnya.
7. *Final Node*: Titik akhir alur aktivitas, digambarkan sebagai *filled circle* dalam *circle*.

Activity Diagram sangat efektif untuk menggambarkan logika kompleks dengan *multiple decision points* dan *parallel processing*, serta untuk mendokumentasikan algoritma atau *business process* sebelum diimplementasikan dalam *code*.

2.11.3 *Sequence Diagram*

Sequence Diagram merupakan diagram interaksi yang menunjukkan bagaimana objek atau komponen sistem berkomunikasi satu sama lain dalam urutan waktu tertentu untuk menyelesaikan suatu fungsi atau use case. Diagram ini fokus pada *message passing* antar objek dan *timing* dari interaksi tersebut. Komponen utama *Sequence Diagram* meliputi:

1. *Participant (Actor/Object):* Entitas yang berpartisipasi dalam interaksi, digambarkan sebagai *box* di bagian atas diagram. *Participant* bisa berupa *user*, *system component*, atau *external system*.
2. *Lifeline*: Garis vertikal putus-putus yang keluar dari participant, merepresentasikan keberadaan objek selama periode waktu tertentu dalam interaksi.

3. *Activation Box*: *Rectangle* tipis pada *lifeline* yang menunjukkan periode waktu ketika objek sedang aktif memproses atau menunggu *response*.
4. *Message*: Komunikasi antar participant, digambarkan sebagai *horizontal arrow* dari *satu lifeline* ke *lifeline* lain. Jenis *message* meliputi:
5. *Synchronous* message *(solid arrow):* Sender menunggu *response* sebelum melanjutkan
6. *Asynchronous message* (*open arrow*): *Sender* tidak menunggu *response*
7. *Return message (dashed arrow): Response* dari *receiver* ke *sender*
8. *Interaction Frame*: *Optional rectangle* yang mengelilingi sekumpulan message untuk menunjukkan *loop*, *alternative scenarios*, atau *optional interactions*.

Sequence Diagram sangat berguna untuk memahami *flow* kontrol dalam sistem, terutama untuk skenario yang melibatkan *multiple components*. Diagram ini menjawab pertanyaan: "Bagaimana urutan komunikasi antar komponen untuk menyelesaikan suatu fungsi?"

2.11.4 *Entity Relationship Diagram (ERD)*

Entity Relationship Diagram (ERD) adalah diagram yang menggambarkan struktur data dan relasi antar entitas dalam database sistem. *ERD* merupakan bagian krusial dari database *design*, membantu memastikan bahwa skema database dapat menyimpan semua informasi yang dibutuhkan dengan efisien, menghindari redundansi data, dan menjaga integritas referensial. Komponen utama *ERD* meliputi:

1. *Entity*: Objek atau konsep yang datanya akan disimpan dalam *database*, digambarkan sebagai *rectangle*. Setiap *entity* akan menjadi tabel dalam database. Contoh: *Customer*, *Product*, *Order*.
2. Attribute: Properti atau karakteristik dari *entity*, digambarkan sebagai *oval* yang terhubung ke *entity*. *Attribute* akan menjadi kolom dalam tabel. Terdapat beberapa jenis *attribute*:
3. *Simple attribute*: *Attribute* yang tidak dapat dipecah lagi (*atomic*)
4. *Composite attribute*: *Attribute* yang terdiri dari beberapa *sub-attribute*
5. *Derived attribute*: *Attribute* yang nilainya dihitung dari *attribute* lain

6. *Multi-valued attribute*: *Attribute* yang dapat memiliki multiple values
7. *Primary Key*: *Attribute* atau kombinasi *attribute* yang *uniquely identify* setiap *record* dalam *entity*, digambarkan dengan underline pada nama *attribute*.
8. *Relationship*: Asosiasi antar *entity*, digambarkan sebagai diamond. *Relationship* menunjukkan bagaimana data dari satu *entity* berhubungan dengan data *entity* lain. Jenis relationship meliputi:
9. *One-to-One (1:1):* Satu *instance entity* A berhubungan dengan maksimal satu *instance entity* B
10. *One-to-Many* (1:N): Satu *instance entity* A dapat berhubungan dengan banyak *instance entity* B
11. *Many-to-Many (M:N):* Banyak *instance entity* A dapat berhubungan dengan banyak *instance entity* B
12. *Foreign Key*: *Attribute* dalam *entity* yang mereferensikan *primary key entity* lain, digunakan untuk mengimplementasikan *relationship* dalam *database* relasional.
13. *Cardinality*: Notasi numerik pada *relationship* yang menunjukkan *minimum* dan *maximum number* of *instances* yang dapat berpartisipasi dalam *relationship* (contoh: 0..1, 1..1, 0..*, 1..*).

ERD dapat digambarkan dalam berbagai notasi (*Chen notation, Crow's Foot notation, UML notation*), Pada Penelitian ini, *Crow's Foot Notation* dipilih dikarenakan memungkinkan untuk banyaknya *Entity* dan *Attribute* dan tetap dapat divisualisasikan dengan *compact*. namun semua notasi memiliki tujuan yang sama: menspesifikasikan struktur data dan *constraint* relasional sebelum implementasi *database* fisik.

Penggunaan *UML* dalam pengembangan sistem informasi modern terbukti meningkatkan kualitas desain sistem, mempermudah proses implementasi, dan memfasilitasi komunikasi antar *stakeholder* (Zikra et al., 2025). Dengan dokumentasi *UML* yang lengkap, sistem dapat dikembangkan secara terstruktur dan memudahkan *maintenance* di masa mendatang.

2.11.5 *Flowchart*

Flowchart adalah representasi grafis dari suatu proses atau algoritma yang menggambarkan urutan langkah-langkah secara logis dari awal hingga akhir. *Flowchart* digunakan untuk memodelkan logika program, prosedur sistem, maupun proses bisnis, dan menjadi alat bantu yang efektif dalam tahap analisis maupun desain sistem informasi (Dennis et al., 2015).

Flowchart mempermudah komunikasi antar pengembang dan *stakeholder* dengan menyajikan proses secara visual dan mudah dipahami. Diagram ini juga bermanfaat untuk mengidentifikasi potensi kesalahan logika atau inefisiensi dalam proses yang sedang dianalisis.

Komponen utama dalam *flowchart* meliputi:

1. *Terminator*: Menandakan titik awal atau akhir dari proses. Digambarkan dengan bentuk oval.
2. *Process*: Menunjukkan suatu aktivitas atau langkah dalam proses. Digambarkan sebagai persegi panjang.
3. *Decision*: Menandakan titik percabangan berdasarkan kondisi logis (ya/tidak). Digambarkan sebagai belah ketupat.
4. *Input/Output*: Menunjukkan aktivitas *input* (masukan) atau *output* (keluaran) *data*. Digambarkan sebagai jajar genjang.
5. *Connector*: Digunakan untuk menghubungkan dua bagian *flowchart* yang berjauhan. Digambarkan sebagai lingkaran kecil.
6. *Arrow* (Panah): Menunjukkan arah aliran proses dari satu langkah ke langkah berikutnya.

Flowchart biasanya dibagi menjadi dua jenis:

1. *System Flowchart*, yang menggambarkan alur data dan proses dalam sistem informasi secara menyeluruh.
2. *Program Flowchart*, yang mendeskripsikan logika internal dari suatu program atau modul secara lebih detail.

2.12 *Software* dan *Tools* yang Digunakan

Pengembangan sistem rekomendasi *parking stand* dalam penelitian ini akan menggunakan berbagai *software* dan *tools modern* untuk memastikan efisiensi *development*, *kualitas code*, dan performa sistem yang optimal. Pemilihan teknologi didasarkan pada kebutuhan integrasi antara aplikasi *web* berbasis *PHP* dan *model machine learning* berbasis *Python*, dengan mempertimbangkan kemudahan *deployment* dan *maintainability* sistem.

2.12.1 *Backend Development*

1. *PHP* (*Hypertext Preprocessor*) 8.3.25 Menurut (Lincopinis et al., 2023), *PHP* adalah bahasa pemrograman *server-side* yang fundamental untuk membangun logika bisnis sistem, termasuk *REST API* untuk komunikasi dengan model *machine learning*. *PHP* dipilih karena kemampuannya dalam menangani *web application development* yang cepat dan mudah diintegrasikan dengan database seperti MySQL. Versi 8.3.25 dipilih karena menyediakan fitur-fitur modern seperti *improved type system*, *performance enhancements*, dan *better error handling* yang mendukung pengembangan aplikasi *enterprise-grade*.
2. *PHP Native* dengan Pola *MVC Kustom*

   Backend sistem akan dibangun menggunakan arsitektur *Model-View-Controller (MVC)* yang dikembangkan secara *native*. Pendekatan ini dipilih untuk menghindari *framework* eksternal seperti *Laravel* atau *CodeIgniter*, yang juga merupakan pilihan umum dalam ekosistem *PHP* untuk membangun aplikasi web yang skalabel (Lincopinis et al., 2023). Pendekatan *custom MVC* ini memberikan kontrol penuh terhadap struktur aplikasi dan menghindari *overhead* dari *framework* yang tidak digunakan, menghasilkan aplikasi yang lebih ringan dan cepat. Struktur *MVC* kustom akan mencakup: *routing system* untuk *URL handling*, *middleware* untuk *authentication* dan *authorization*, *repository pattern* untuk *data access layer*, dan *service layer* untuk *business logic*.

3. *MySQL (MariaDB 10.4.32)*

   Relational Database Management System (RDBMS) yang akan digunakan untuk menyimpan data operasional sistem, termasuk data pesawat, parking stand, *flight schedule*, dan log prediksi. MySQL dipilih karena stabilitas, performa, dan kompatibilitas yang baik dengan PHP (Lincopinis et al., 2023). MariaDB 10.4.32 merupakan *drop-in replacement* untuk MySQL yang dibundel dengan XAMPP, yang juga menyertakan Apache dan PHP dalam paketnya (Chandra & Setyaningsih, n.d.), menyediakan fitur-fitur tambahan seperti *improved query optimizer* dan *better handling* untuk *concurrent connections* yang penting dalam lingkungan operasional bandara.

4. *PDO (PHP Data Objects)*

   Extension PHP yang akan digunakan untuk koneksi *database* dengan *prepared statements* untuk mencegah *SQL injection*. *PDO* menyediakan *abstraction layer* yang memungkinkan *switching* antar *database engines* dengan minimal *code changes*, serta mendukung *parameterized queries* untuk keamanan maksimal.

2.12.2 *Frontend Development*

1. *HTML5*

   HTML5 akan digunakan untuk membangun struktur halaman dengan *semantic elements*. (Bhatt et al., 2024) menjelaskan bahwa *HTML* menyediakan fondasi dan struktur dari setiap halaman web, termasuk penempatan teks, gambar, dan elemen lainnya

2. *CSS3* (*Cascading Style Sheets*)

   CSS3 dimanfaatkan untuk *styling* dengan *modern features* seperti *flexbox* dan *grid layout*. Ini sejalan dengan fungsi *CSS* untuk mendefinisikan tampilan *visual* dan *layout* halaman web, termasuk warna, *font*, dan latar belakang (Bhatt et al., 2024).

3. *JavaScript*

   JavaScript digunakan untuk interaktivitas dan *dynamic content updates*. Penggunaan *JavaScript* sangat krusial untuk meningkatkan interaktivitas

pengguna dan menyediakan fungsionalitas dinamis pada halaman web (Bhatt et al., 2024).

2.12.3 *Machine Learning Development*

1. *Python 3.13.5*

   Bahasa pemrograman yang akan digunakan untuk mengembangkan model *machine learning Random Forest*. *Python* (Lawson & Chris Mayfield, 2024) dipilih karena ekosistem *library machine learning* yang lengkap dan *mature* , serta popularitasnya yang tinggi di berbagai bidang

2. *scikit-learn 1.7.2*

   Library machine learning untuk *Python* yang menyediakan implementasi algoritma *Random Forest* melalui *class RandomForestClassifier*. *scikit-learn* dipilih karena kemudahan penggunaan, dokumentasi lengkap, dan performa yang terbukti baik untuk klasifikasi *multi-class* seperti dalam penelitian ini. *Library* ini juga menyediakan *tools* untuk *Grid Search hyperparameter tuning* dan *cross-validation* yang akan digunakan dalam proses optimasi model.

3. *pandas 2.3.3 & NumPy 2.3.3*

   Library untuk manipulasi dan analisis data. *pandas* akan digunakan untuk *data preprocessing*, *cleaning*, dan *feature engineering*, sementara *NumPy* akan digunakan untuk komputasi *array* dan operasi matematis dalam *training* model (Lawson & Chris Mayfield, 2024). *pandas* menyediakan struktur *DataFrame* yang *powerful* untuk menangani *dataset* tabular dengan operasi seperti *filtering*, *grouping*, dan *merging* yang *efisien*.

4. *Pickle & Joblib 1.5.2*

   Module untuk serialisasi model machine learning. Model *Random Forest* yang telah dilatih akan disimpan dalam format .pkl menggunakan *pickle* atau *joblib*, memungkinkan *model* di-*load* kembali untuk inferensi tanpa perlu *re-training*. *Joblib* dipilih untuk model besar karena efisiensi dalam *handling numpy arrays* dan kompresi file yang lebih baik dibanding *pickle* standar.

5. *Matplotlib 3.10.7 & Seaborn 0.13.2*

   Library visualisasi untuk membuat *confusion matrix*, *feature importance plots*, dan grafik evaluasi model lainnya. *Seaborn* menyediakan *interface high-level* untuk membuat visualisasi statistik yang *aesthetic* dan informatif, *built on top of matplotlib*.

2.12.4 *Development Tools*

1. *XAMPP 8.2.12*

   Package bundle yang menyediakan *Apache web server*, *MySQL database* (MariaDB), dan *PHP interpreter* dalam satu instalasi (Chandra & Setyaningsih, 2025). *XAMPP* akan digunakan sebagai *local development environment* selama tahap *development* dan *testing* sistem. Studi oleh (Chandra & Setyaningsih, 2025) mengidentifikasi *XAMPP* sebagai salah satu solusi *server* lokal yang populer karena kompatibilitas *cross-platform*-nya.

2. *Visual Studio Code*

   Sebuah *Integrated Development Environment (IDE) lightweight* yang akan digunakan untuk menulis *code* (Tan et al., 2023). *VS Code* dipilih karena *lightweight*, *extensible* dengan berbagai *plugin* atau extensions, dan mendukung multiple languages *(PHP, Python, JavaScript)* dalam satu *editor*. Fitur-fitur utamanya seperti *syntax highlighting*, *IntelliSense code completion*, dan *integrated debugging*.

3. *Blackbox Testing*

   Metode pengujian ini akan digunakan untuk memvalidasi fungsionalitas sistem dari perspektif pengguna. Menurut (Maspupah, 2024), pengujian *black box* berfokus pada penilaian fungsionalitas sistem berdasarkan kebutuhan tanpa menganalisis struktur kode internal. Metode ini efektif untuk mengevaluasi perilaku perangkat lunak dan mengidentifikasi kesalahan pada fungsi, antarmuka, dan akses data eksternal (Maspupah, 2024).

BAB III
METODE YANG DIUSULKAN

3.1 Desain Penelitian

Penelitian ini merupakan jenis penelitian eksperimen terapan yang menggabungkan perancangan sistem informasi operasional dengan penerapan metode machine learning untuk mendukung pengambilan keputusan. Fokus utama dari sistem adalah sebagai *Decision Support System (DSS)* untuk mempermudah pengelolaan pergerakan pesawat dan alokasi stand di apron bandara.

Pendekatan metodologis yang digunakan dalam penelitian ini adalah kerangka *CRISP-DM* (*Cross Industry Standard Process for Data Mining*) sebagai dasar alur pengembangan data mining dan model klasifikasi. Di samping itu, siklus pembangunan sistem *frontend* dan *backend* akan mengikuti pola *Agile development* untuk memungkinkan iterasi dan evaluasi bertahap selama implementasi prototipe.

3.2 Pengumpulan Data

Data yang akan digunakan dalam penelitian ini diperoleh dari hasil **observasi langsung di lapangan** selama proses magang di unit *Apron Movement Control (AMC)* Bandara Halim Perdanakusuma. Data tersebut merupakan catatan operasional harian pergerakan pesawat yang dicatat secara manual oleh petugas AMC.

Setiap hari, data dicatat dalam format *spreadsheet* dan dikonsolidasikan selama periode tiga bulan. Proses penggabungan dilakukan secara manual oleh peneliti untuk menyatukan file harian menjadi satu *file .csv* yang dapat diproses secara *digital*. Validasi awal terhadap data dilakukan oleh *supervisor AMC* sebagai penanggung jawab operasional di lapangan.

3.3 Metodologi *CRISP-DM*

3.3.1 Business Understanding

Identifikasi Masalah Operasional Berdasarkan observasi langsung selama masa magang di *Unit Apron Movement Control (AMC)* Bandar Udara Halim Perdanakusuma, teridentifikasi bahwa proses alokasi parking stand saat ini berjalan

secara manual dan bergantung pada *judgment* subjektif operator tanpa dukungan sistem berbasis data. Proses pengambilan keputusan memerlukan waktu rata-rata 1-2 menit per alokasi, dengan tingkat konsistensi yang bervariasi antar operator karena ketiadaan panduan prediktif. Secara spesifik, tujuan bisnis yang ingin dicapai meliputi:

1. Menyediakan rekomendasi alokasi *parking stand* berbasis data historis yang dapat membantu operator membuat keputusan lebih cepat dan konsisten
2. Mengurangi waktu pengambilan keputusan dari rata-rata 1-2 menit menjadi maksimal 5 detik (termasuk waktu inferensi model dan *rendering UI*)
3. Meningkatkan konsistensi keputusan alokasi antar operator dengan memberikan baseline recommendations yang objektif berdasarkan pola historis
4. Menyediakan platform terintegrasi yang menggantikan sistem *spreadsheet* tersegmentasi dengan *database* terpusat dan *visualisasi real-time*
5. Memfasilitasi monitoring dan evaluasi performa operasional melalui *dashboard analytics* dan *automated reporting*

Keberhasilan proyek akan diukur berdasarkan kriteria kuantitatif dan kualitatif sebagai berikut:

1. Kriteria:
    a. *Top-3 Accuracy Model:* Model *machine learning* harus mencapai akurasi minimal 80% dalam memprediksi 3 *parking stand* teratas yang paling sesuai untuk setiap kombinasi input (*aircraft type, airline, category*)
    b. *Response Time* Sistem: Waktu total dari user input hingga *display* rekomendasi tidak boleh melebihi 5 detik, mencakup *processing time PHP backend*, *inferensi Python ML model*, dan *rendering frontend*
    c. *Prediction Acceptance Rate*: Minimal 70% dari rekomendasi *ML* diterima oleh operator (*stand* yang dipilih berada dalam Top-3 *recommendations*)

2. Batasan:

   Beberapa batasan dan asumsi yang perlu dicatat dalam proyek ini:

   a. Model akan fokus pada 20 *parking stand* operasional di *Main Apron*, tidak mencakup 63 *stand* lainnya yang digunakan untuk *RON* dan *repositioning*
   b. *Dataset* historis terbatas pada periode 3 Bulan *Operational* (Mei-Juli 2025)
   c. Sistem tidak mengintegrasikan faktor eksternal seperti kondisi cuaca atau *maintenance schedule stand*

3. Asumsi:

   a. Pola historis alokasi *stand* yang tercatat dalam *dataset* mencerminkan *best practices* operasional dan dapat dijadikan *baseline* untuk prediksi
   b. Operator akan tetap memiliki *final* authority untuk menerima atau menolak rekomendasi ML (*human-in-the-loop approach*)
   c. Distribusi tipe pesawat, maskapai, dan kategori penerbangan di masa depan tidak akan berbeda drastis dari periode *training* data dengan pemahaman bisnis yang jelas ini, tahap selanjutnya dalam *CRISP-DM* adalah Data *Understanding* untuk mengeksplorasi karakteristik *dataset* operasional *AMC* secara mendalam.

4. Penerjemahan ke *Data Mining Goals*

   Tujuan bisnis di atas diterjemahkan ke dalam tujuan data mining sebagai berikut:

   a. Membangun Model Klasifikasi *Multi-Class*

      Mengembangkan model *Random Forest* untuk klasifikasi *multi-class* dengan 20 kelas target (*parking stands A0-A3, B1-B13, WR01-WR03*) yang dapat memprediksi *parking stand* optimal berdasarkan 3 fitur input utama: *aircraft_type*, *operator_airline*, dan *category.*

   b. Optimasi *Hyperparameter* untuk Mencapai *Target Accuracy*

      Melakukan hyperparameter tuning menggunakan Grid Search dengan 5-*fold cross-validation untuk menemukan kombinasi parameter Random*

Forest (n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, class_weight) yang menghasilkan *Top-3 Accuracy ≥ 80%.*

c. Evaluasi Performa Model dengan *Multiple Metrics*
Mengukur performa model tidak hanya berdasarkan *accuracy*, tetapi juga *precision*, *recall*, *F1-score* per kelas, *confusion matrix* untuk mengidentifikasi misklasifikasi *patterns*, dan *feature importance* untuk memahami kontribusi setiap variabel.

d. *Deployment* Model ke Sistem *Production*
Mengintegrasikan model yang telah dilatih ke dalam *web application* menggunakan *PHP-Python interprocess communication*, dengan model *persistence* menggunakan *pickle* format dan *response time optimization* untuk memenuhi target ≤5 detik.

3.3.2 *Data Understanding*

Dataset operasional yang digunakan dalam penelitian ini bersumber dari catatan historis *Apron Movement Control (AMC)* Bandar Udara Halim Perdanakusuma yang mencakup periode Mei-Juli 2025. Data dikumpulkan dari *Google Sheets* harian yang diisi secara manual oleh operator *AMC* setiap kali terjadi pergerakan pesawat (*landing dan takeoff*). Sumber dan Karakteristik Data *Dataset* mencakup total 4069 entri pergerakan pesawat dengan 4 atribut operasional yang relevan untuk prediksi alokasi *parking stand*. Data mencerminkan kondisi nyata operasional bandara dengan berbagai tipe pesawat, maskapai penerbangan, dan kategori penerbangan yang beroperasi di Halim Perdanakusuma.

Bandar Udara Halim Perdanakusuma memiliki total 83 *parking stand* yang tercatat dalam *database* sistem, mencakup *Main Apron* (A0-A3, B1-B13), *South Apron* (SA01-SA30), *New South Apron* (NSA01-NSA15), *Remote West* (RW01-RW03), *Remote East* (RE01-RE07), dan Remote West (RW01-RW11). Namun, penelitian ini membatasi cakupan prediksi model hanya pada 20 parking stand operasional di Main Apron (A0-A3, B1-B13, WR01-WR03) yang digunakan untuk aktivitas *landing* dan *takeoff* pesawat. *Stand-stand* di *apron* lainnya (total 63 stand) digunakan untuk keperluan *Remain Overnight (RON)* dan *repositioning* pesawat, sehingga tidak termasuk dalam sistem rekomendasi alokasi *stand* berbasis *machine*

learning. Dengan demikian, model *Random Forest* yang dikembangkan merupakan *multi-class classification* dengan 20 kelas target stand, bukan 83 kelas.

Dataset awal direkonsiliasi secara manual, disatukan ke dalam satu file .csv berjudul *DATASET AMC.csv* yang kemudian digunakan sebagai dasar dalam pengolahan dan pelatihan model klasifikasi.

Dataset menunjukkan distribusi penggunaan yang tidak seimbang antar *parking stand*, dengan beberapa *stand* seperti *A1, A2, dan B1* memiliki frekuensi penggunaan lebih tinggi dibanding *stand* lainnya. Hal ini mencerminkan preferensi operasional dan karakteristik stand yang berbeda (ukuran, aksesibilitas, kedekatan dengan terminal).

1. *COMMERCIAL* (penerbangan terjadwal maskapai komersial) : 60.97%
2. *CHARTER* (penerbangan *Charter* terjadwal dan tidak terjadwal) : 28.09%
3. *CARGO* (Penerbangan membawa barang ekspedisi) : 10.94%

Setelah seluruh *file* disatukan, dilakukan pembersihan data dan transformasi awal. *Dataset* akhir disimpan dalam bentuk *parking_history_clean.csv*, yang terdiri atas [4069] entri dan [4] kolom, dan digunakan sebagai input utama dalam proses *data mining*. *Dataset* mentah memiliki 11 kolom atribut utama, yang masing-masing merepresentasikan karakteristik peristiwa parkir pesawat:

1. *REGISTRATION* – Registrasi unik pesawat
2. *TYPE* – Tipe pesawat (misal: B737, ATR)
3. *ON BLOCK* – Waktu pesawat mulai parkir
4. *OFF BLOCK* – Waktu pesawat meninggalkan stand
5. *PARKING STAND* – Lokasi parkir aktual
6. *FROM* – Bandara asal
7. *TO* – Bandara tujuan
8. *ARR* – Kode kedatangan
9. *DEP* – Kode keberangkatan
10. *OPERATOR / AIRLINES* – Nama maskapai/operator
11. *CATEGORY* – Kategori pergerakan (komersial, *charter*, *VIP*, dll.)

Dari atribut mentah tersebut, dilakukan rekayasa fitur (*feature engineering*) untuk mendapatkan variabel input yang lebih representatif terhadap pola klasifikasi:

1. *aircraft_type*: normalisasi dari kolom *TYPE*
2. *operator_airline*: normalisasi dari kolom *OPERATOR / AIRLINES*
3. *category*: hasil normalisasi *CATEGORY*
4. *airline_category*: hasil kombinasi *operator_airline* dan *category*
5. *aircraft_airline*: kombinasi *TYPE* dan *operator_airline*
6. *aircraft_category*: kombinasi *TYPE* dan *category*

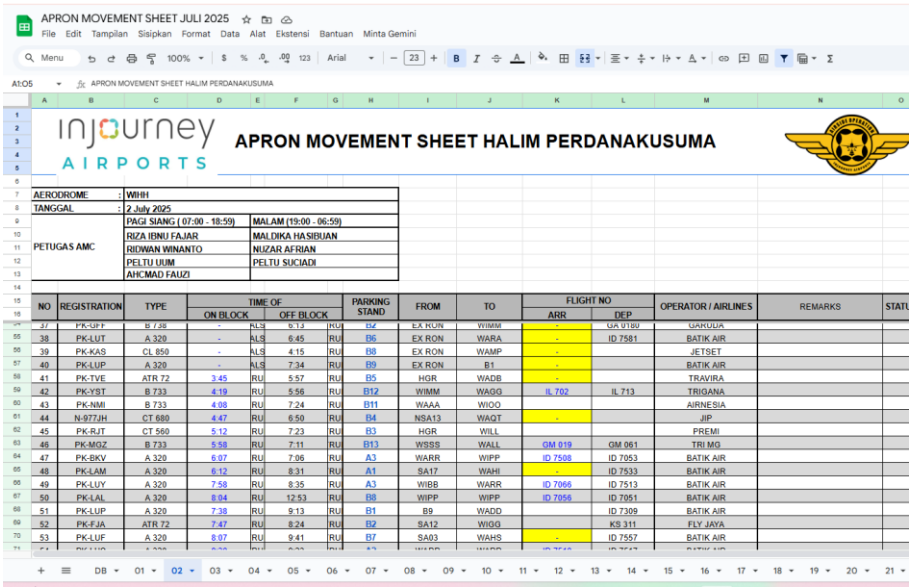
Target (label) dalam klasifikasi ini adalah kolom *PARKING STAND*, yang merepresentasikan *stand* aktual tempat pesawat ditempatkan oleh *AMC*.

3.3.3 Data Preparation

Sebelum data mentah digunakan dalam proses pelatihan model, data tersebut harus melalui serangkaian tahapan pra-pemrosesan untuk memastikan kualitas, konsistensi, dan kompatibilitas dengan algoritma *machine learning*. Proses ini dibagi menjadi dua tahap utama: rekonsiliasi data mentah dan transformasi fitur.

1. **Proses Rekonsiliasi dan Pembersihan Data**

   Dataset awal yang digunakan berasal dari catatan pergerakan harian yang disimpan dalam 24 *file Google Sheets* terpisah (mencakup periode Mei-Juli 2025).

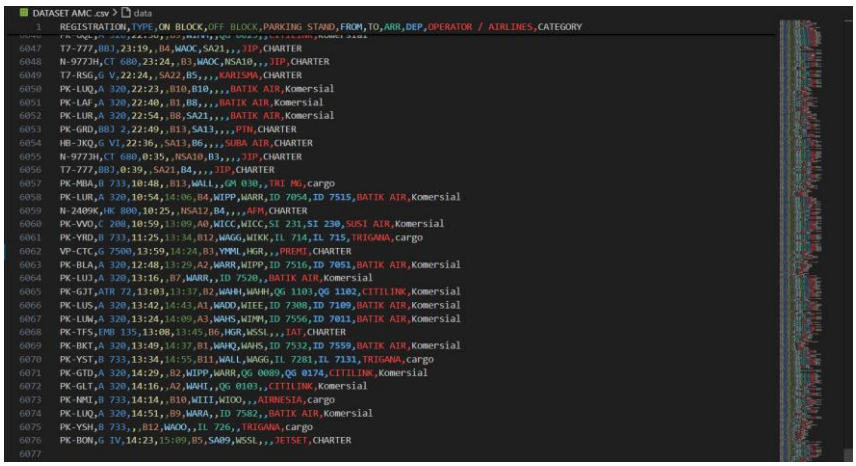


Gambar 3. 1 *Apron Movement Sheet*

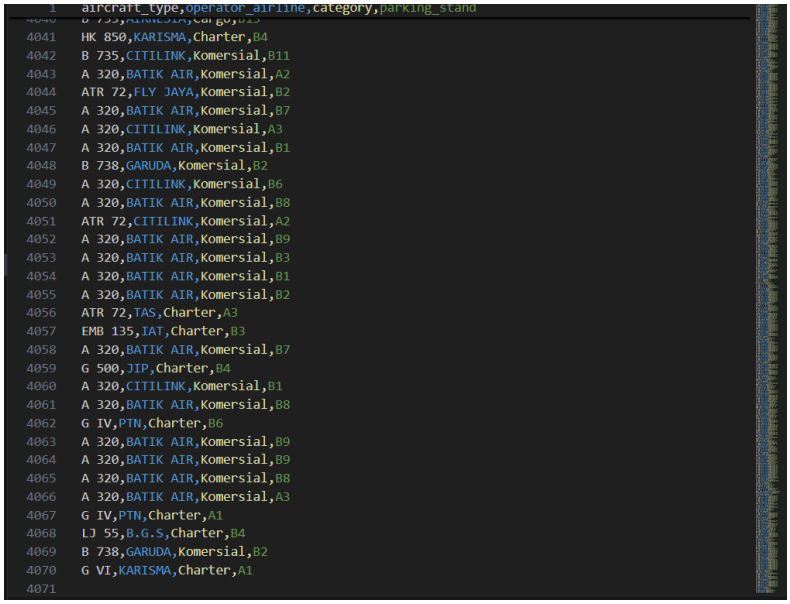
Proses rekonsiliasi dan migrasi data ini melibatkan beberapa langkah:

a. **Ekspor Data:** Setiap file *Google Sheets* harian diekspor ke format *file* individual.
b. **Konsolidasi:** Seluruh *file* tersebut digabungkan (direkonsiliasi) secara manual ke dalam satu *file master* utama, yaitu *parking_history.csv*.
c. **Pembersihan Awal:** Dilakukan pembersihan data dan normalisasi. Ini termasuk menghapus duplikasi entri, mengubah format kolom mentah seperti *TYPE*, *OPERATOR / AIRLINES*, dan *CATEGORY* ke format yang konsisten (misalnya, *uppercase*), dan memvalidasi tipe data, terutama atribut waktu (*ON BLOCK* dan *OFF BLOCK*).

Setelah pembersihan awal dan penanganan *missing values* (dijelaskan di bawah), dataset akhir digunakan sebagai input utama dalam proses *data mining*.



Gambar 3. 2 *Csv Dataset*



Gambar 3. 3 *Csv Clean Dataset*

2. Transformasi Fitur

Untuk mempersiapkan data agar kompatibel dengan *algoritma Random Forest*, dilakukan transformasi fitur sebagai berikut:

a. One-Hot Encoding* untuk *Aircraft_Type

Tipe pesawat *(TYPE* yang dinormalisasi menjadi *aircraft_type*) merupakan variabel kategorik nominal. *One-Hot Encoding* akan mentransformasi setiap *aircraft type* unik (*B 737, A 320, ATR 72*, dll.) menjadi *fitur* biner terpisah. Contoh: jika terdapat 15 *unique aircraft types*, maka akan dibuat 15 kolom biner baru (*aircraft_B737, aircraft_A320*, ...) dengan nilai 1 untuk *type* yang sesuai dan 0 untuk lainnya.

b. *Label Encoding* untuk *Operator_Airline*

Maskapai penerbangan *(OPERATOR / AIRLINES* yang dinormalisasi menjadi *operator_airline*) ditransformasi menjadi label numerik menggunakan *LabelEncoder* dari scikit-learn. Setiap maskapai diberikan integer unik (*BATIK AIR=0, GARUDA=1, CITILINK=2, dst).*

c. *Categorical Encoding* untuk *Category*

Kategori penerbangan *(COMMERCIAL, CHARTER, CARGO, STATE, VVIP)* ditransformasi menjadi nilai numerik. Tergantung pada jumlah kategori unik, *ordinal encoding* dapat digunakan jika jumlahnya ≤5, atau *one-hot encoding* jika >5 untuk menghindari asumsi urutan implisit.

d. *Encoding* Target Label (*parking_stand*)

Fitur target, yaitu *parking_stand*, juga di-*encode* menggunakan *LabelEncoder. Mapping* (pemetaan) dari *encoder* ini disimpan secara terpisah sebagai *file* .pkl (contoh: *encoders_redo.pkl*) untuk memastikan konsistensi *encoding* dan dekoding antara fase pelatihan dan fase prediksi/inferensi saat *deployment*.

e. Handling Missing Values

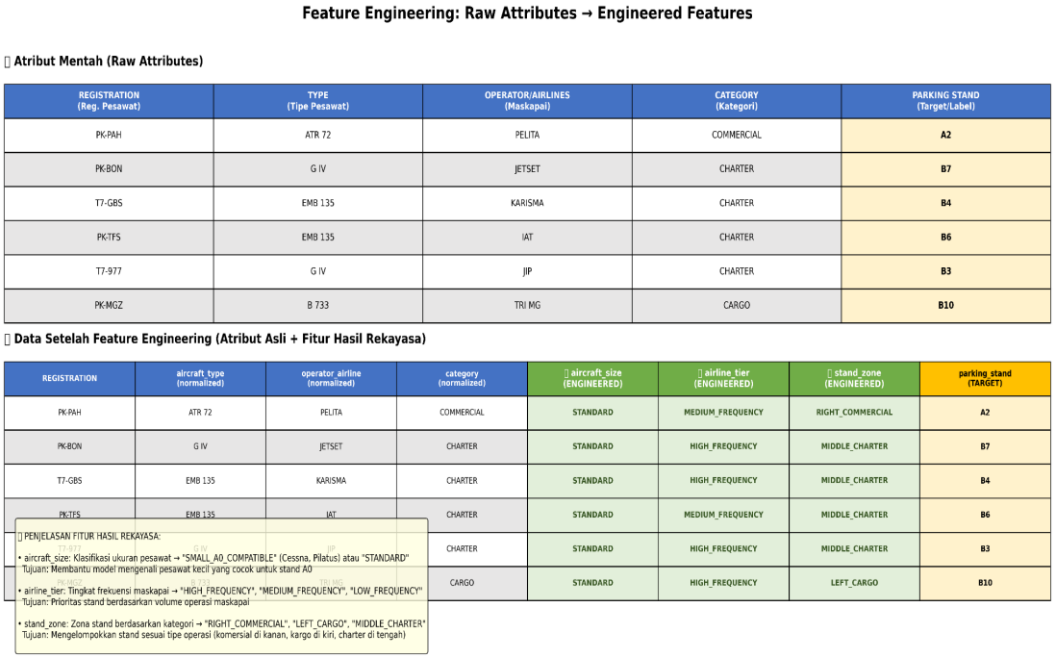
Missing values pada kolom-kolom kritikal yang esensial untuk pelatihan model, seperti *REGISTRATION, TYPE (aircraft_type), CATEGORY, OPERATOR (operator_airline),* dan *parking_stand,* akan dihapus (baris data di-*drop*).

Missing values pada kolom non-kritikal atau yang tidak dipakai model (seperti *remarks*, *FROM*, *TO*) akan diisi dengan *placeholder string* "*UNKNOWN" atau "N/A"* agar tetap konsisten.

f. Data Validation

Validasi akhir dilakukan untuk memastikan:

1) Kode *parking_stand valid* (hanya 20 *operational stands* yang terdaftar, nilai di luar rentang resmi akan disaring).
2) *Aircraft types* konsisten (tidak ada tipo seperti "*B737*" vs "*B 737*").
3) *Timestamps* dalam format yang benar.
4) Tidak ada entri duplikat untuk *REGISTRATION* + *timestamp* yang sama.



Feature Engineering: Raw Attributes → Engineered Features

 Atribut Mentah (Raw Attributes)

REGISTRATION (Reg. Pesawat)	TYPE (Tipe Pesawat)	OPERATOR/AIRLINES (Maskapai)	CATEGORY (Kategori)	PARKING STAND (Target/Label)
PK-PAH	ATR 72	PELITA	COMMERCIAL	A2
PK-BON	G IV	JETSET	CHARTER	B7
T7-GBS	EMB 135	KARISMA	CHARTER	B4
PK-TFS	EMB 135	IAT	CHARTER	B6
T7-977	G IV	JIP	CHARTER	B3
PK-MGZ	B 733	TRI MG	CARGO	B10

 Data Setelah Feature Engineering (Atribut Asli + Fitur Hasil Rekayasa)

REGISTRATION	aircraft_type (normalized)	operator_airline (normalized)	category (normalized)	aircraft_size (ENGINEERED)	airline_tier (ENGINEERED)	stand_zone (ENGINEERED)	parking_stand (TARGET)
PK-PAH	ATR 72	PELITA	COMMERCIAL	STANDARD	MEDIUM_FREQUENCY	RIGHT_COMMERCIAL	A2
PK-BON	G IV	JETSET	CHARTER	STANDARD	HIGH_FREQUENCY	MIDDLE_CHARTER	B7
T7-GBS	EMB 135	KARISMA	CHARTER	STANDARD	HIGH_FREQUENCY	MIDDLE_CHARTER	B4
PK-TFS	EMB 135	IAT	CHARTER	STANDARD	MEDIUM_FREQUENCY	MIDDLE_CHARTER	B6
T7-977	G IV	JIP	CHARTER	STANDARD	HIGH_FREQUENCY	MIDDLE_CHARTER	B3
PK-MGZ	B 733	TRI MG	CARGO	STANDARD	HIGH_FREQUENCY	LEFT_CARGO	B10

 PENJELASAN FITUR HASIL REKAYASA:

• aircraft_size: Klasifikasi ukuran pesawat → "SMALL_A0_COMPATIBLE" (Cessna, Pilatus) atau "STANDARD"
Tujuan: Membantu model mengenali pesawat kecil yang cocok untuk stand A0

• airline_tier: Tingkat frekuensi maskapai → "HIGH_FREQUENCY", "MEDIUM_FREQUENCY", "LOW_FREQUENCY"
Tujuan: Prioritas stand berdasarkan volume operasi maskapai

• stand_zone: Zona stand berdasarkan kategori → "RIGHT_COMMERCIAL", "LEFT_CARGO", "MIDDLE_CHARTER"
Tujuan: Mengelompokkan stand sesuai tipe operasi (komersial di kanan, kargo di kiri, charter di tengah)

Gambar 3. 4 Contoh *Subset Data* Kolom Hasil *Feature Engineering*

3.3.4 Format *File* dan Integrasi Sistem

Dataset disimpan dalam format *CSV* (*Comma-Separated Values*) untuk memudahkan pengolahan menggunakan bahasa pemrograman *Python*. Spesifikasi teknis dari *file dataset bersih* yang digunakan untuk pelatihan adalah sebagai berikut:

Spesifikasi *File CSV*:

a. ***Filename:*** *parking_history_clean.csv*
b. ***Delimiter:*** *comma (,)*
c. ***Encoding:*** *UTF-8*
d. ***Header row*:** *Yes* (baris pertama berisi nama kolom)
e. ***Row count:*** 4069 *records*
f. **Quote character:** *double quote* (") untuk *string* yang mengandung *comma*

Struktur *Header*:

Registration,Aircraft_Type,On_Block_Time,Parking_Stand,From,To,Flight_No_Arr,Operator_Airline,Category,Off_Block_Time,Movement_Date

Contoh Data *Row* :

PKGFL,B737,14:30,A1,WIII,WID,ID71,BATIKAIR,COMMERCIAL,16:45,2024-03-15

Alur Integrasi Data dalam Sistem

Data mengalir melalui *pipeline* berikut dari sumber hingga digunakan oleh model *machine learning*:

1. **Data Collection Phase:**
   a. Proses dimulai dengan ekspor data dari *file Google Sheets* harian per bulan.
   b. Dilakukan penggabungan dan penyatuan data secara manual ke dalam satu *master file* (*DATASET AMC.csv*).
   c. Struktur awal dari tabel data mentah ini dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Struktur Tabel Dataset

Kolom	Deskripsi
REGISTRATION	Registrasi unik pesawat
TYPE	Tipe pesawat
OPERATOR / AIRLINES	Maskapai atau operator
ON BLOCK	Waktu mulai parkir
OFF BLOCK	Waktu selesai parkir
FROM	Bandara asal
TO	Bandara tujuan
CATEGORY	Jenis penerbangan
PARKING STAND	Stand aktual tempat parkir pesawat

Dataset ini menjadi fondasi dalam proses pelatihan model klasifikasi menggunakan algoritma *Random Forest*, yang kemudian diintegrasikan ke dalam sistem informasi *AMC* sebagai modul pendukung keputusan prediktif.

2. ***Data Cleaning Phase:***
    a. Penghapusan duplikasi dan pembersihan data kosong (sesuai kriteria di 3.3.3).
    b. Transformasi tipe data dan konversi atribut waktu.
    c. Alur proses: *DATASET AMC.csv → Python preprocessing script* (*clean_data.py*) → *parking_history_clean.csv*
3. ***Training Phase:***
    a. *parking_history_clean.csv → Python training script (train_model.py) → Feature engineering → Model training → parking_stand_model_rf_redo.pkl + encoders_redo.pkl*
4. ***Deployment Phase:***
    a. *File model (.pkl)* disimpan di direktori ml/ dalam aplikasi web.
5. ***Inference Phase (Real-time Prediction):***
    a. *User input via form web PHP → JSON payload → Python inference script (predict.py) → Load model + encoders → Prediksi → JSON response → Backend PHP → Tampilan Frontend.*

Integrasi dengan *Database MySQL*:

Dataset ini menjadi fondasi dalam proses pelatihan model klasifikasi menggunakan algoritma *Random Forest*. Setelah model dilatih, data historis *(parking_history_clean.csv)* juga akan diimpor ke *database MySQL* untuk:

a. Menyediakan data referensi untuk validasi.
b. Menyimpan *log* prediksi untuk evaluasi model.
c. Mendukung *dashboard reporting* dan analitik.

Proses impor dilakukan menggunakan perintah *MySQL LOAD DATA INFILE*

Dengan struktur *file* dan integrasi data yang jelas ini, sistem dapat dengan mudah di-*maintain* dan diperbarui ketika data baru tersedia atau ketika model perlu dilatih ulang dengan *dataset* yang lebih baru.

3.4 Analisa Sistem Berjalan

Sebelum merancang sistem yang diusulkan, perlu dilakukan analisis mendalam terhadap sistem yang saat ini berjalan *di Unit Apron Movement Control (AMC)* Bandar Udara Halim Perdanakusuma. Analisis ini bertujuan untuk memahami proses bisnis *existing*, mengidentifikasi *stakeholder* yang terlibat, serta memetakan permasalahan yang menjadi dasar pengembangan sistem baru.

3.4.1 Proses Bisnis Sistem Manual

Sistem operasional *AMC* saat ini berjalan secara manual dengan mengandalkan koordinasi verbal melalui radio komunikasi *HT (Handie-Talkie)* dan dokumentasi *spreadsheet* terpisah. Proses alokasi parking stand dimulai ketika pesawat akan melakukan *landing*, dan operator *AMC* menerima informasi melalui dua jalur: pertama, notifikasi langsung dari *Air Traffic Control (ATC) via radio HT;* atau kedua, *monitoring* proaktif oleh operator *AMC* melalui *platform FlightRadar* untuk memantau pesawat yang akan masuk ke wilayah bandara.

Setelah menerima informasi kedatangan pesawat (*flight number, aircraft type, airline, estimated time of arrival),* operator *AMC* melakukan langkah-langkah berikut:

1. Pengecekan *Stand Availability*: Operator membuka *Google Sheets* untuk melihat *stand* mana yang sedang terisi atau kosong. Proses ini memerlukan *scroll manual* melalui baris-baris data dan *cross-check* dengan informasi radio untuk memastikan data ter-update.
2. Analisis Kelayakan *Stand*: Operator mengevaluasi *stand* yang tersedia berdasarkan beberapa kriteria: (a) tipe dan ukuran pesawat untuk memastikan kompatibilitas fisik dengan stand, (b) kategori penerbangan (*commercial, charter*, atau *cargo*) yang berpengaruh pada zonasi *stand*, (c) preferensi maskapai jika ada request khusus untuk *stand* tertentu, dan (d) pola historis frekuensi penggunaan stand berdasarkan pengalaman operasional sebelumnya.
3. Pengambilan Keputusan: Berdasarkan analisis tersebut, operator menentukan stand yang paling sesuai. Keputusan ini bersifat subjektif dan bergantung pada *judgment* operator, tanpa dukungan sistem berbasis data atau model

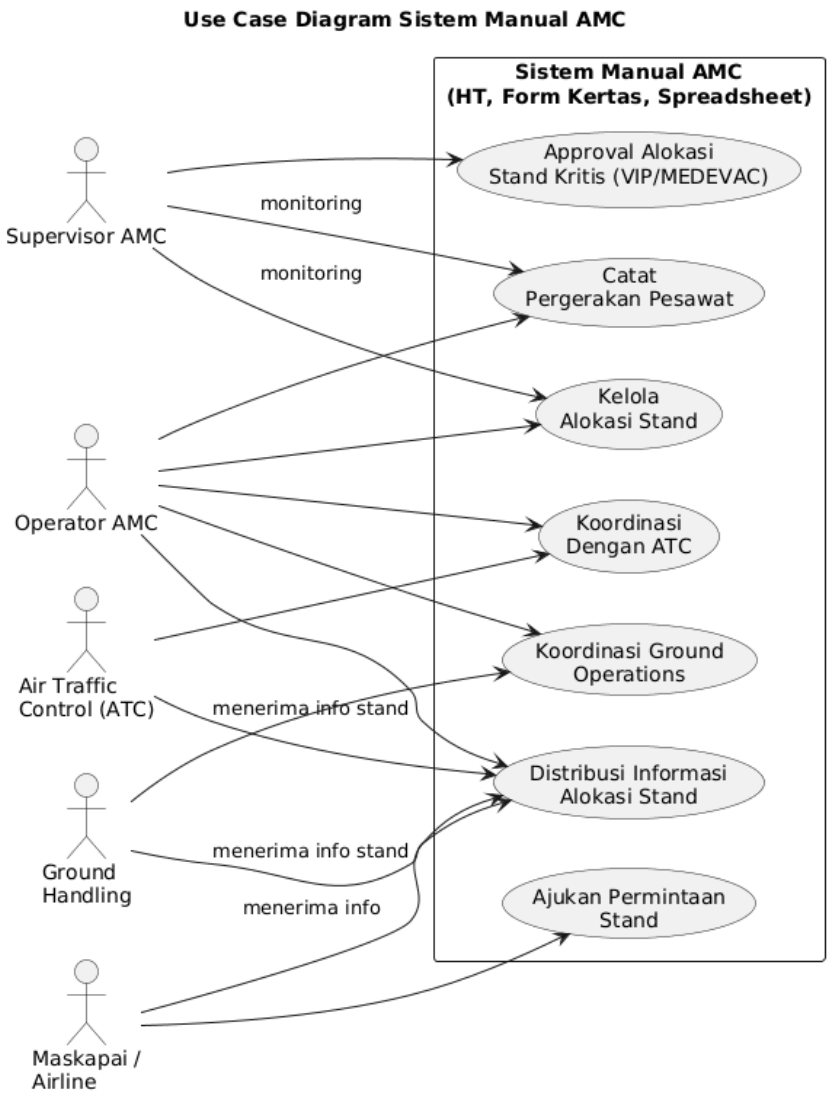
prediktif. Proses pengambilan keputusan rata-rata memerlukan waktu 1-2 menit, dengan durasi yang bervariasi tergantung kompleksitas situasi (*peak hour, special requests*, atau kondisi khusus seperti evakuasi medis atau *VVIP*).

4. Koordinasi dengan *Stakeholder*: Setelah keputusan dibuat, operator mengkomunikasikan alokasi *stand* kepada *ATC* via *radio HT* atau telepon *landline* dengan frekuensi ratusan kali per hari. Operator juga menginformasikan kepada *ground handling (towing, marshalling)* milik maskapai terkait melalui *radio HT*. Jika terdapat request khusus dari maskapai untuk stand tertentu, operator akan mengevaluasi kelayakan request tersebut; namun *final authority* untuk *approval* tetap berada di tangan unit *AMC*.
5. Dokumentasi Manual: Operator mencatat data pergerakan pesawat ke *Google Sheets* harian (*Apron Monitoring Sheet*) dengan mengisi kolom: *Registration, Type, On-Block time, Parking Stand, From, To, Flight No Arr/Dep, Operator/Airlines, dan Category.* Dokumentasi ini dilakukan secara manual dan *real-time* saat pesawat *on-block*.
6. *Update Off-Block*: Ketika pesawat siap berangkat dan meninggalkan *stand (off-block),* operator kembali *update spreadsheet* dengan mengisi kolom *Off-Block time*, menandakan *stand* kembali tersedia untuk alokasi berikutnya.
7. Rekonsiliasi Bulanan: Di akhir bulan, data dari *Apron Monitoring Sheet* harian (sekitar 30 *sheet* per bulan) dikonsolidasikan secara manual ke dalam dua laporan terpisah: *Monthly Charter Report* yang merangkum semua penerbangan *charter* pada bulan tersebut, dan *Monthly RON* (*Remain Overnight*) *Report* yang mencatat pesawat-pesawat yang melakukan parkir semalam. Proses rekonsiliasi ini rentan terhadap *human error*, terutama duplikasi data atau inkonsistensi timestamp.

Keseluruhan proses ini menunjukkan ketergantungan tinggi terhadap kemampuan manual operator, sistem komunikasi verbal yang rentan *noise*, dan dokumentasi tersegmentasi yang menyulitkan monitoring *real-time* serta analisis historis.

3.4.2 *Use Case* Diagram Sistem Manual

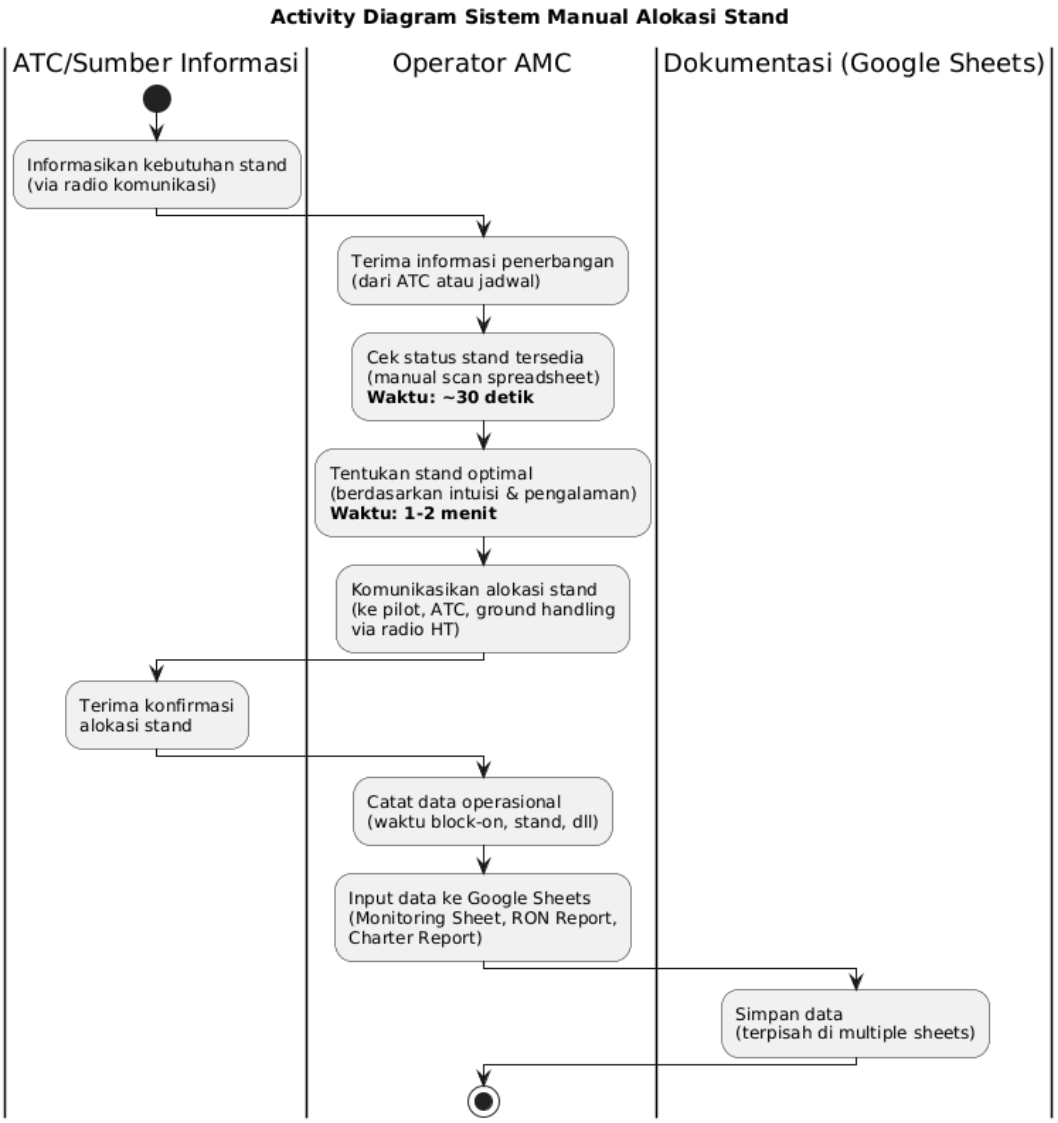
Use Case Diagram sistem manual menggambarkan interaksi antara para unit operational bandara dengan proses operasional *existing* sebelum implementasi sistem berbasis *web*.



Gambar 3. 5 *Use Case Diagram Sistem Manual AMC*

3.4.3 *Activity Diagram* Alokasi Stand Manual

Activity Diagram sistem manual menggambarkan alur kerja lengkap dari proses alokasi parking *stand* secara manual, mencakup *decision points* dan *parallel activities* yang terjadi selama operasional.



Gambar 3. 6 *Activity Diagram* Sistem Manual AMC

3.4.4 Permasalahan Sistem Berjalan

Sebagaimana telah diuraikan pada BAB I *Section* 1.2 (Identifikasi Masalah), sistem manual yang saat ini berjalan menghadapi enam permasalahan utama yang menjadi dasar pengembangan sistem baru:

1. Dokumentasi Manual dan Terpisah
2. Tidak Ada Visualisasi *Real-Time*
3. Alokasi *Stand* Tanpa Dukungan Sistem
2. Validasi Data Lemah
3. Koordinasi Antarunit Rentan Miskomunikasi
4. Tidak Ada Sistem Terpadu dengan Akses Terbatas

Keenam masalah ini menunjukkan adanya kebutuhan mendesak akan sistem informasi terintegrasi yang mampu menyatukan proses pencatatan, validasi data, visualisasi kondisi operasional secara *real-time*, serta dukungan pengambilan keputusan berbasis data historis melalui model *machine learning.*

3.5 Perancangan Sistem yang Diusulkan

Berdasarkan analisis permasalahan sistem berjalan, penelitian ini akan merancang sistem informasi *Apron Movement Control (AMC)* berbasis web yang terintegrasi, menggabungkan proses pencatatan operasional, visualisasi *real-time,* dan fitur prediksi alokasi *parking stand*.

Sistem akan menggunakan pendekatan klasifikasi *supervised learning* dengan algoritma *Random Forest* yang diimplementasikan menggunakan *RandomForestClassifier* dari pustaka *scikit-learn* di *Python*. *Random Forest* dipilih karena kemampuannya dalam:

1. Menangani *Multi-Class Classification*: Dengan 20 kelas *parking stand, Random Forest* mampu memberikan prediksi yang stabil dan akurat melalui mekanisme *ensemble voting*.
2. Mengurangi *Overfitting*: Berbeda dengan *single decision tree* yang rentan *overfitting*, *Random Forest* menggunakan *bagging* dan *random feature selection* untuk meningkatkan *generalization*.

3. *Handling Class Imbalance*: *Parameter class_weight='balanced_subsample'* memungkinkan model menangani distribusi kelas yang tidak seimbang (beberapa *stand* lebih sering digunakan).
4. *Probabilistic Output*: Model menghasilkan *confidence score* untuk setiap kelas, memungkinkan implementasi strategi *Top-K prediction* yang memberikan operator pilihan *alternatif stand.*

Model *Random Forest* yang akan dibangun direncanakan menggunakan konfigurasi dasar dengan *100 trees (n_estimators=100)* dan parameter lainnya akan ditentukan melalui proses *hyperparameter tuning* yang dijelaskan pada bagian 3.5.

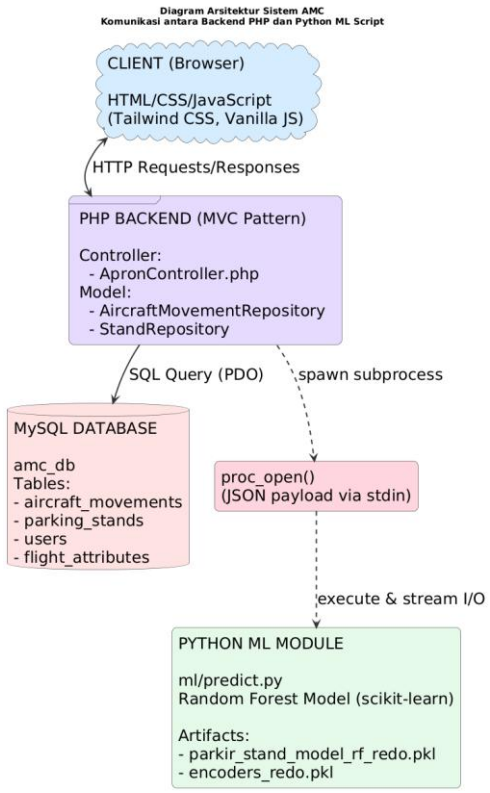
3.5.1 Arsitektur Sistem Informasi Web *AMC*

Sistem yang akan dikembangkan mengadopsi arsitektur *three-tier* yang memisahkan *presentation layer (frontend), application logic layer* (*backend*), dan data *layer* (*database* dan *ML model*).

1. *Presentation Layer (Frontend):*
   a. *User Interface* berbasis *HTML5, CSS3 (Tailwind CSS*), dan *Vanilla JavaScript*
   b. Komponen utama: *Apron Map* (visualisasi *real-time* status *stand*), *Dashboard* (*metrics* operasional), *Form Input Movement, Prediction Result Cards (Top-3 recommendations)*
   c. *Communication via AJAX* (*Fetch API*) ke *backend REST API*
   d. *Real-time updates via polling mechanism* (5 detik untuk status *apron*, 60 detik untuk *ML metrics*)
2. *Application Logic Layer* (*Backend*):
   a. *PHP 8.3.25 native dengan custom MVC pattern*
   b. *Controller: ApronController (movement CRUD, ML prediction request), SnapshotController, UserController*
   c. *Repository Pattern: Database abstraction* untuk *CRUD operations*
   d. *Service Layer: Business logic* untuk *validasi data, stand availability checking*
   e. *Middleware: AuthMiddleware* untuk *session-based authentication* dan *role-based access control*

f. *PHP-Python IPC: proc_open()* untuk komunikasi dengan *ML module via JSON payload*

3. *Data Layer:*
   a. *MySQL (MariaDB 10.4.32): 13 tables* dengan focus pada *aircraft_movements, parking_stands, ml_prediction_log*
   b. *Python ML Module: Random Forest model (parking_stand_model_rf_redo.pkl, 2.6 MB), encoders (encoders_redo.pkl), inference script (ml/predict.py)*
   c. *Model training artifacts: feature importance plots, confusion matrix (20×20), classification reports*
4. *Integration Flow:*
   a. *User request prediction via frontend*
   b. *Frontend send AJAX POST to /api/apron/recommend*
   c. *Backend PHP receive request, validate input, construct JSON payload*
   d. *PHP call Python script via proc_open() with JSON via stdin*
   e. *Python load model from .pkl file, run inference, return Top-3 predictions with probabilities via stdout*
   f. *PHP parse JSON response, log to ml_prediction_log table, return to frontend*
   g. *Frontend display Top-3 recommendation cards with click-to-allocate functionality*



Gambar 3. 7 Diagram Arsitektur Program

5. *Role Management & Access Control*

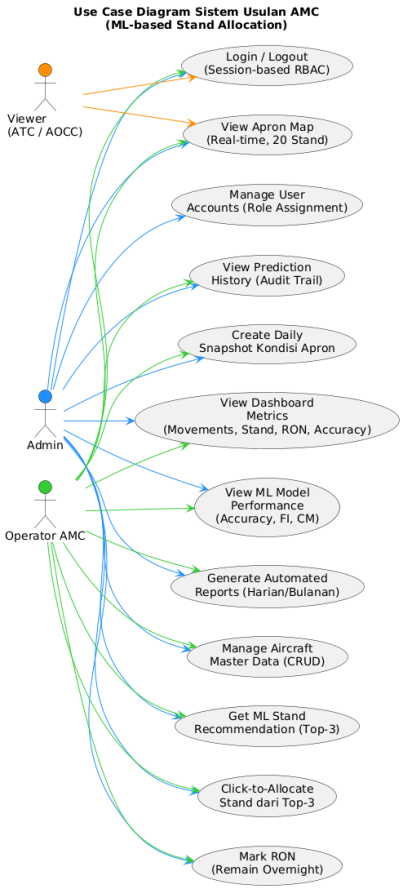
Sistem mendukung tiga peran pengguna dengan hak akses berbeda:

Tabel 3. 2 Hak Akses

Role	***Akses***
Admin	Kelola akun pengguna (*CRUD*), reset sandi, akses penuh ke *logbook*, *snapshot*, dan semua modul.
***Operator* (AMC)**	*Input & edit data, panggil prediksi ML, tandai RON, buat snapshot, akses dashboard.*
***Viewer* (*ATC*/*AOCC*)**	Hanya baca: status *apron*, data historis, tampilan visual; tidak bisa mengedit atau menyimpan.

3.5.2 *Use Case* Diagram Sistem Usulan

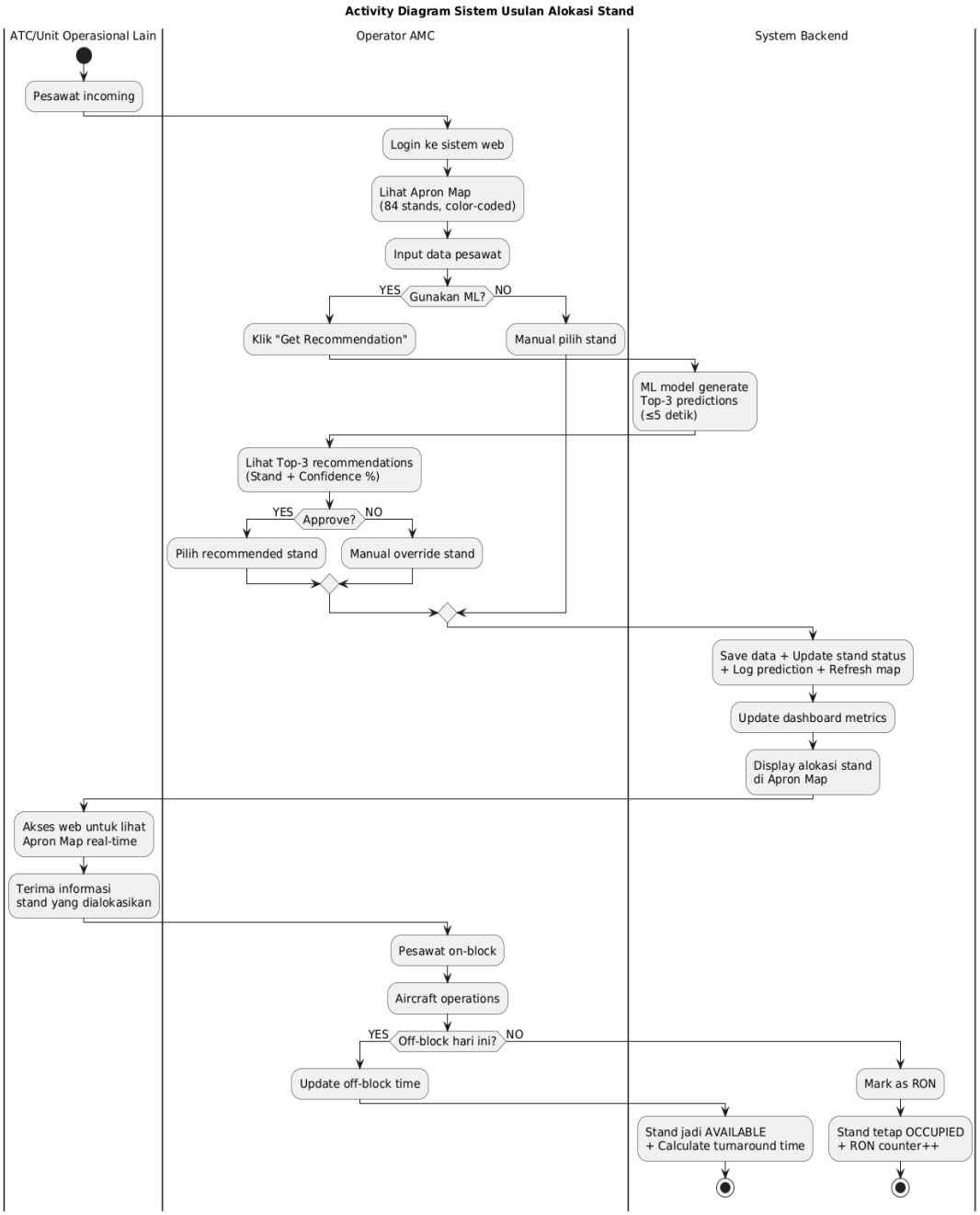
Use Case Diagram sistem usulan menggambarkan interaksi yang akan tersedia dalam sistem baru, dengan peningkatan signifikan dibanding sistem manual.



Gambar 3. 8 *Use Case* Diagram Sistem Usulan

3.5.3 *Activity Diagram* Sistem Usulan

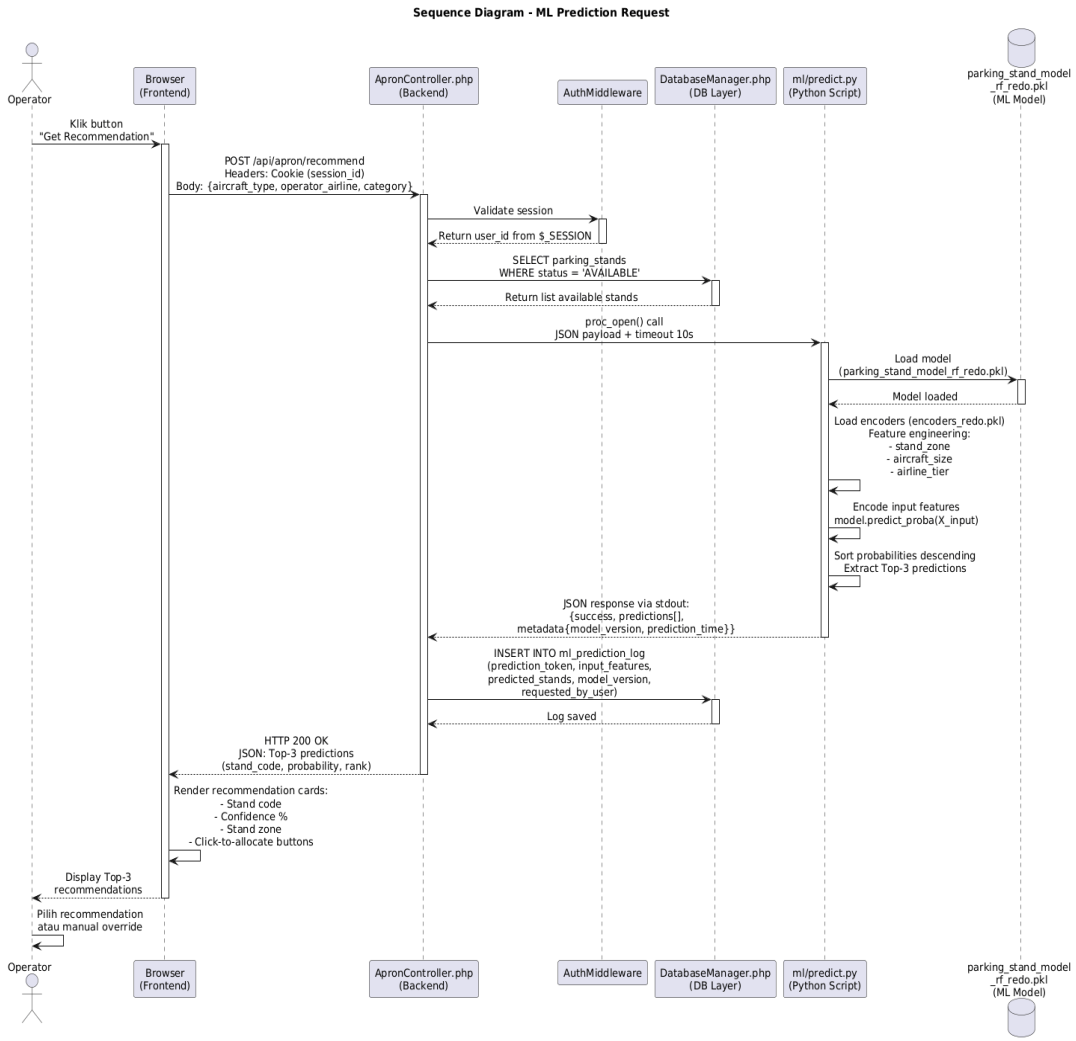
Activity Diagram sistem usulan menunjukkan alur proses alokasi *stand* dengan dukungan *machine learning*, menggambarkan perbedaan signifikan dengan proses manual.



Gambar 3. 9 *Activity Diagram* Sistem Usulan

3.5.4 *Sequence Diagram* Sistem Usulan

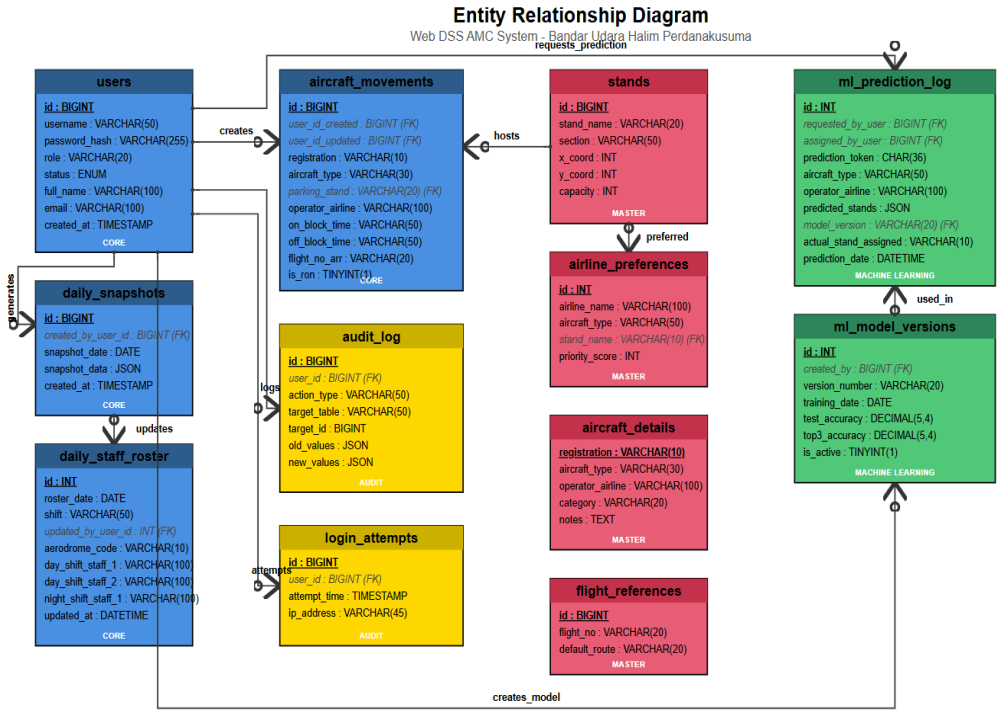
Sequence Diagram menggambarkan interaksi detail antar komponen sistem saat operator melakukan *request ML prediction*. *Participants* yang terlibat meliputi: *Operator* (*User*), *Browser* (*Frontend*), *ApronController.php* (*Backend PHP*), *DatabaseManager.php* (*DB Layer*), *ml/predict.py* (*Python ML Script*), dan *parking_stand_model_rf_redo.pkl (ML Model File).*



Gambar 3. 10 *Sequence Diagram ML Prediction Request*

3.5.5 *Entity Relationship* Diagram Sistem Usulan

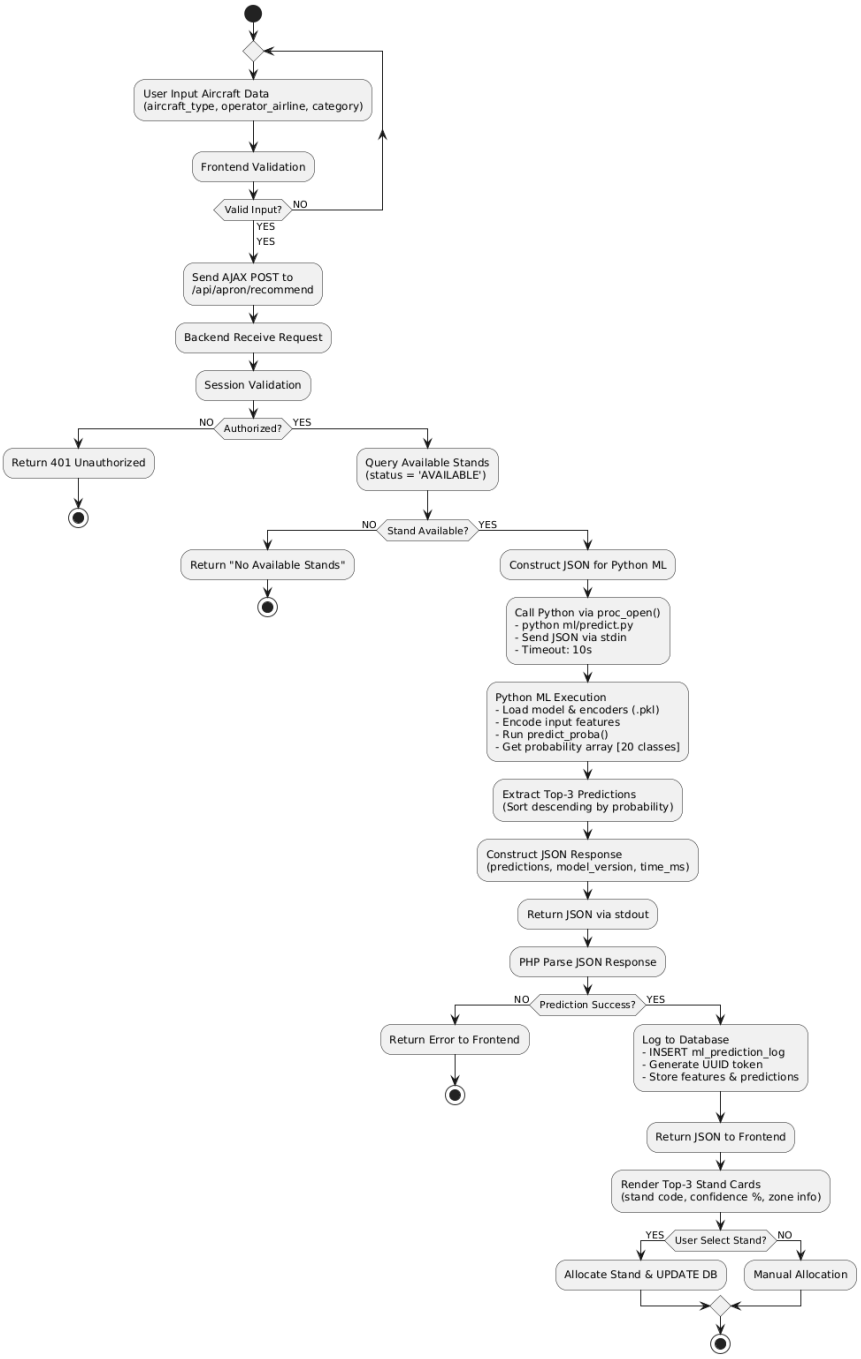
ERD sistem usulan menggambarkan struktur *database* lengkap dengan 13 tabel yang saling berelasi.



Gambar 3. 11 *Entity Relationship Diagram* Sistem

Catatan Penting: *Database* dirancang untuk mendukung 83 parking stand fisik yang ada di bandara, namun sistem *machine learning* akan fokus pada 20 parking stand operasional di *Main Apron* (*A0-A3, B1-B13, WR01-WR03*) yang digunakan untuk *landing* dan *takeoff*. Stand-stand lainnya (63 stands) tetap tercatat di database untuk keperluan *RON tracking* dan *repositioning*, namun tidak termasuk dalam *scope* prediksi model *Random Forest*.

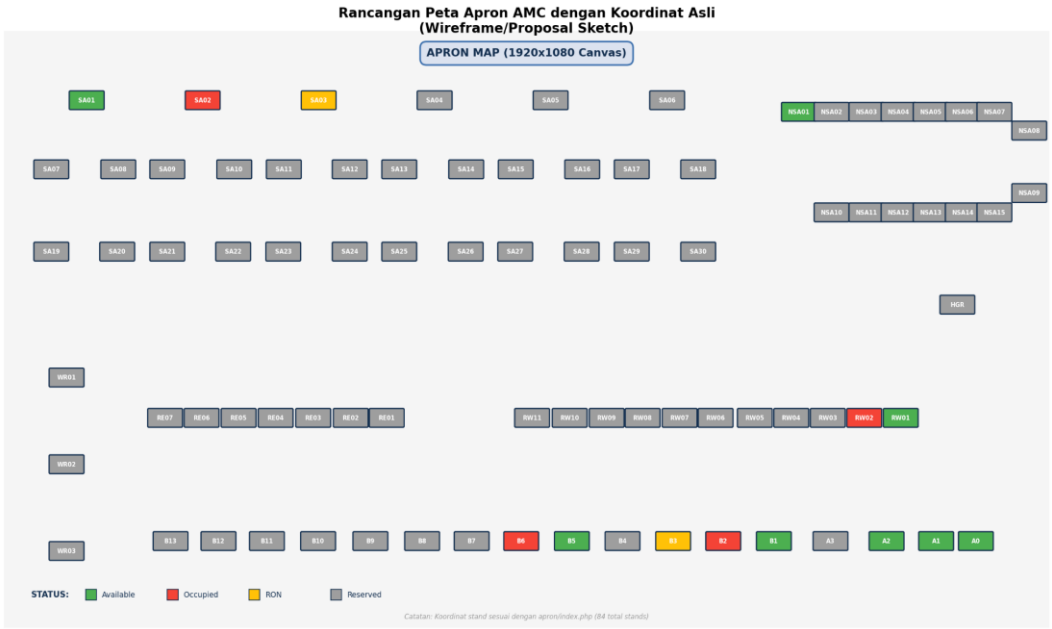
3.5.6 *Flowchart* Alur Prediksi



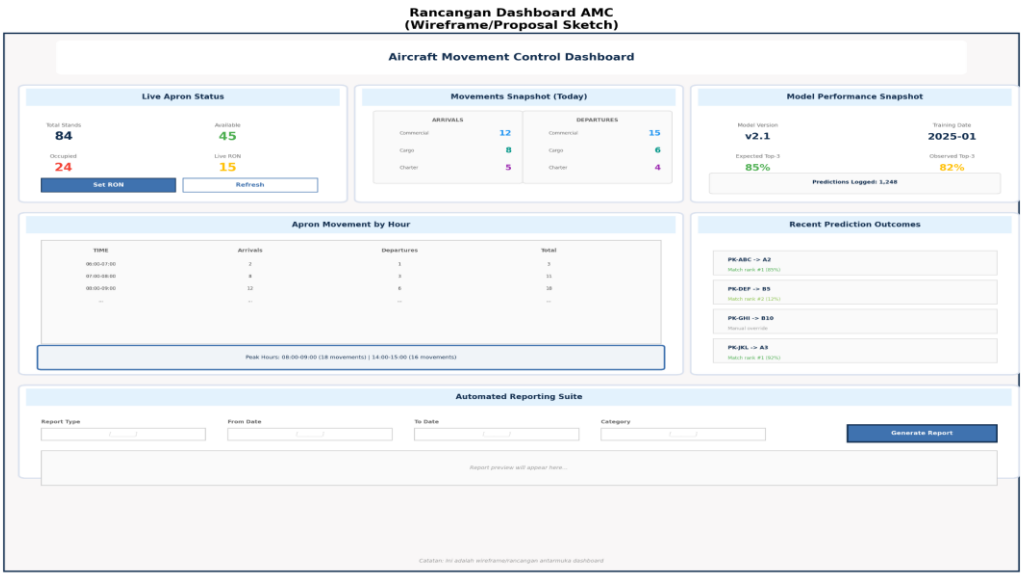
Gambar 3. 12 *Flowchart Prediction*

3.5.7 Wireframe Design Interface

Wireframe menggambarkan rancangan visual antarmuka pengguna sistem yang mengutamakan kemudahan penggunaan (*usability*) dan efisiensi operasional. Berikut adalah *wireframe* utama dari sistem:



Gambar 3. 13 Rancangan *Index* / *Apron Map*



Gambar 3. 14 Rancangan *Dashboard*

3.5.8 Fitur Utama Sistem

Sistem ini dirancang untuk mendukung seluruh alur kerja operasional petugas *AMC*, dari pencatatan pergerakan hingga eksekusi rekomendasi berdasarkan machine learning. Beberapa fitur utama yang tersedia meliputi:

a. *Input* & *Edit* Peristiwa Parkir Pesawat

Operator *AMC* dapat memasukkan data seperti registrasi pesawat, waktu *on/off-block*, maskapai, serta stand yang digunakan.

b. Prediksi Alokasi *Stand* (*Random Forest*)

Dengan satu klik, sistem memanggil modul *machine learning* untuk merekomendasikan stand optimal berdasarkan atribut input.

c. Visualisasi Interaktif Status *Apron*

Dashboard menyajikan peta *apron* secara *real-time* dengan warna, label, dan indikator pergerakan pesawat yang mudah dibaca.

d. *Snapshot* & Rekapitulasi Operasional Harian

Operator dapat menyimpan snapshot kondisi apron, menandai RON *(Remain Overnight),* dan mengekspor log pergerakan harian.

e. Manajemen Akun dan Hak Akses

Sistem mendukung *multi-role access* untuk mengatur batasan fitur berdasarkan tipe pengguna.

3.6 Eksperimen dan Pengujian

Pelatihan model klasifikasi dilakukan dengan skrip *Python ml/train_model.py. Dataset* dibagi menggunakan teknik *hold-out* menjadi 80% untuk pelatihan dan 20% untuk pengujian akhir.

Proses *hyperparameter* tuning dilakukan *menggunakan GridSearchCV dengan 5-fold cross-validation untuk mengidentifikasi k*ombinasi parameter optimal. Parameter grid yang diuji:

1. *'n_estimators': [100, 200],*
2. *'max_depth': [None, 20, 30],*
3. *'min_samples_leaf': [2, 5, 10],*
4. *'min_samples_split': [5, 10],*

5. *'class_weight': ['balanced', 'balanced_subsample']*

Proses *Grid* Search menguji **72 kombinasi parameter** (2 × 3 × 3 × 2 × 2) dengan total **360 *model fits*** (72 kombinasi × 5 *folds*).

3.7 Evaluasi dan Validasi Hasil

Evaluasi model akan dilakukan dengan mengukur performa pada *set* uji menggunakan metrik berikut:

1. ***Accuracy***

Mengukur proporsi prediksi yang benar dari seluruh data:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

2. ***Precision***

Mengukur ketepatan prediksi positif:

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. ***Recall* (*Sensitivity*)**

Mengukur seberapa banyak data positif yang berhasil dikenali:

$$\text{Recall} = \frac{TP}{TP + FN}$$

4. ***F1 Score***

Rata-rata harmonis antara *precision* dan *recall*:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. ***Top-3 Accuracy***

Digunakan untuk mengukur apakah label aktual masuk dalam tiga besar hasil rekomendasi model berguna dalam skenario sistem *DSS* yang menawarkan beberapa opsi ke petugas.

$$\text{Top-}k\text{ Accuracy} = \frac{\text{Jumlah instance di mana label benar ada dalam Top-}k}{\text{Total instance}}$$

DAFTAR PUSTAKA

Akbiyik, D., Özen, S., Ulusoy, T., Dagsuyu, E., Akpinar, E., Ucak, Y. B., Uyduran, S., Demir, D., & Fescioglu-Unver, N. (2024). A decision support system for internal logistics operations management. *Journal of Building Design and Environment*. https://doi.org/10.70401/jbde.2024.0001

Andrade-Arenas, L., Rubio-Paucar, I., & Yactayo-Arias, C. (2024). Data mining for predictive analysis in gynecology: a focus on cervical health. *International Journal of Electrical and Computer Engineering*, *14*(3), 2822–2833. https://doi.org/10.11591/ijece.v14i3.pp2822-2833

Ashari Tri Gunawan, W., & -Sekolah Tinggi Teknologi Kedirgantaraan Yogyakarta, R. (2023). Operational Analysis of the Apron Movement Control (AMC) Unit in Handling Aircraft at Abdulrachman Saleh Airport Apron Malang. *JETISH: Journal of Education Technology Information Social Sciences and Health E-ISSN*.

Bhatt, S., Shrivastava, V., Pandey, A., & Scholar, B. T. (2024). A Review of Current Front-End Development Technologies. In *International Journal of Research Publication and Reviews Journal homepage: www.ijrpr.com* (Vol. 5). www.ijrpr.com

Chandra, A. Y., & Setyaningsih, W. (2025). BULLETIN OF COMPUTER SCIENCE RESEARCH Benchmarking Local Development Environments: Analyzing the Performance of XAMPP, MAMP, and Laragon. *Media Online)*, *5*(3), 193–206. https://doi.org/10.47065/bulletincsr.v5i3.493

Darono, A. (2025). Dealing with Last-Mile Analytics. *Scientax*, *6*(2), 107–123. https://doi.org/10.52869/st.v6i2.592

Dennis, A., Wixom, B., & Tegarden, D. (2015). *SYSTEMS ANALYSIS & DESIGN An Object-Oriented Approach with UML D E N N I S W I X O M T E G A R D E N*. http://store.visible.com/Wiley.aspx

Endut, N., Hamzah, W. M. A. F. W., Ismail, I., Yusof, M. K., Baker, Y. A., & Yusoff, H. (2022). A Systematic Literature Review on Multi-Label Classification based on Machine Learning Algorithms. *TEM Journal*, *11*(2), 658–666. https://doi.org/10.18421/TEM112-20

Gui, D., Le, M., Huang, Z., & D'Ariano, A. (2024). A Decision Support Framework for Aircraft Arrival Scheduling and Trajectory Optimization in Terminal Maneuvering Areas. *Aerospace*, *11*(5). https://doi.org/10.3390/aerospace11050405

Hidayah, N. W., Sasmita, R. R., Mayangsari, M. K., Kusuma, O. G. W., Rante, H., & Fariza, A. (2022). Invitin Project: Scrum Framework Implementation

in a Software Development Project Management. *INTEK: Jurnal Penelitian*, *9*(1), 58–65. https://doi.org/10.31963/intek.v9i1.3332

Ibrahim, M. (2023). Evolution of Random Forest from Decision Tree and Bagging: A Bias-Variance Perspective. *Dhaka University Journal of Applied Science and Engineering*, *7*(1), 66–71. https://doi.org/10.3329/dujase.v7i1.62888

Insaurralde, C. C., & Blasch, E. (2024). *Ontological Airspace-Situation Awareness for Decision System Support*. https://doi.org/10.3390/aerospace

Jumlad, W., & Nurhalisa. (2024). *IMPLEMENTASI SISTEM INFORMASI APRON MOVEMENT CONTROL DI BANDAR UDARA INTERNASIONAL SULTAN HASANUDDIN MAKASSAR*. *6*(1). https://doi.org/10.56521/attendant-dirgantara.v6i1.1134

Lawson, A., & Chris Mayfield, W. (2024). *Introduction to Python Programming UDAYAN DAS, SAINT MARY'S COLLEGE OF CALIFORNIA*.

Lincopinis, D., Paul Apiag, C. W., Bryan Cadiz, E. S., & Lincopinis, D. R. (2023). *A Review on PHP Programming Language*. https://orcid.org/0000-0001-9503-8965,

Marsa, A., Syelly, R., Amelia, R., Sopandi, A., Suharsono, Amalia, R., Irsa, R., widiyawati, Maulana, N., Irzavika, N., Pradana, M., Gusti, K., Budiman, A., Niqotaini, Z., & Sayekti, I. (2023). *Konsep Sistem Informasi*. https://www.researchgate.net/publication/382304762

Maspupah, A. (2024). LITERATURE REVIEW: ADVANTAGES AND DISADVANTAGES OF BLACK BOX AND WHITE BOX TESTING METHODS. *Jurnal Techno Nusa Mandiri*, *21*(2), 151–162. https://doi.org/10.33480/techno.v21i2.5776

Mishra, D., Tripathi, S. M., Chaurasia, A., & Chaurasia, P. K. (2025). A Review on Ensemble Learning Methods: Machine Learning Approach. *International Journal of Research Publication and Reviews*, *6*(2), 3795–3803. https://doi.org/10.55248/gengpi.6.0225.0971

Mobley, W., Sebastian, A., Blessing, R., Highfield, W. E., Stearns, L., & Brody, S. D. (2021). Quantification of continuous flood hazard using random forest classification and flood insurance claims at large spatial scales: A pilot study in southeast Texas. *Natural Hazards and Earth System Sciences*, *21*(2), 807–822. https://doi.org/10.5194/nhess-21-807-2021

Petersen, F., Kuehne, H., Borgelt, C., & Deussen, O. (2022). *Differentiable Top-k Classification Learning*.

Putri, V., & Suprapti, S. (2022). JURNAL AKUNTANSI, EKONOMI DAN MANAJEMEN BISNIS ANALISIS KINERJA PETUGAS APRON MOVEMENT CONTROL (AMC) DALAM MENINGKATKAN

KESELAMATAN PENERBANGAN DI BANDARA UDARA INTERNASIONAL ADI SOEMARMO SOLO. *Juli*, *2*(2), 190–197.

Safira, M. S., Rahaningsih, N., & Dana, R. D. (2024). PENERAPAN DATA MINING UNTUK KLASIFIKASI PENJUALAN OBAT MENGGUNAKAN METODE K-NEAREST NEIGHBOR. In *Jurnal Mahasiswa Teknik Informatika* (Vol. 8, Issue 1).

Suryaman, R. W., & Wang, G. (2022). *Penerapan Data Mining Untuk Prediksi Slot Time di Bandara Internasional di Indonesia : Algoritma J48*.

Tan, J., Chen, Y., & Jiao Shuyin. (2023). Visual Studio Code in Introductory Computer Science Course: An Experience Report. *CEUR Workshop Proceedings*, *2657*, 1–9. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Ulfa, S., Sylvia, T., & Azhari Saragih, N. (2023). Perancangan Sistem Informasi Padang Online Parking Stand (POSTAND) Dalam Pelayanan Lalu Lintas Penerbangan. *Airman: Jurnal Teknik Dan Keselamatan Transportasi*, *6*(2), 78–96. https://doi.org/10.46509/ajtk.v6i2.339

Yesicatama, S., & Widagdo, J. (2025). Analisis Optimalisasi Parking Stand oleh Unit Apron Movement Control (AMC) pada Saat Peak Hour di Bandar Udara Internasional I Gusti Ngurah Rai Bali. *Jurnal Manajemen Dirgantara*, *18*(1), 43–54. https://doi.org/10.56521/manajemen-dirgantara.v18i1.1385

Zikra, A. A., Darnilasari, A., Yanuary, R., & Sari, K. (2025). Design of Web-Based Project Management System with Multi-Level Role-Based Access Control. *Journal of Computer Science and Informatics Engineering*, *4*(3), 201–215. https://doi.org/10.55537/cosie.v4i3.1179