BAB II
KAJIAN LITERATUR

2.1 Hotel

Perkembangan industri perhotelan memungkinkan wisatawan untuk dapat menikmati liburan mereka ketika berkunjung ke sebuah negara [21]. Berdasarkan Keputusan Menteri Parpostel no Km 94/HK103/MPPT 1987, hotel adalah salah satu jenis akomodasi yang mempergunakan sebagian atau keseluruhan bagian untuk jasa pelayanan penginapan, penyedia makanan dan minuman serta jasa lainnya bagi masyarakat umum yang dikelola secara komersil. Hotel merupakan salah satu usaha komersial dengan tujuan memberikan pelayanan yang terbaik bagi para tamu ataupun wisatawan yang menginap di hotel [22]. Produk yang ditawarkan pada sebuh hotel terdiri dari [23]:

1. Produk berwujud, termasuk penjualan penginapan, makanan, minuman, kolam renang, dan barang lainnya.
2. Komoditi tidak berwujud, termasuk kehangatan, kemudahan, daya tarik, keamanan, dan sebagainya.
3. Hal-hal yang mudah rusak dan segar, seperti buah-buahan, sayuran, makanan laut, dan daging, tidak tahan lama.
4. Barang tahan lama yang tidak mudah rusak, seperti alkohol, minuman ringan, furnitur untuk tamu, dan lain sebagainya.

Hotel memiliki area-area ruangan tertentu yang dimasukkan dalam kategorisasi dan kriteria hotel. Area-area tersebut diantaranya [23]:

1. Kamar tidur
2. Ruang makan/restoran
3. *Function room*
4. Rekreasi atau olahraga
5. *Drugstore*
6. Lobi
7. Taman
8. Utilitas

Dalam industri perhotelan, ada beberapa klasifikasi hotel untuk membedakan ukuran, jenis, lokasi, fungsi, dan sebagainya. Beberapa cara yang dapat digunakan untuk mengkategorikan sebuah hotel yaitu [23]:

1. Berdasarkan Peringkat Bintang

   Menurut Keputusan Menteri Kebudayaan dan Pariwisata no. Km 3/KW 001/MKP 02, hotel dapat diklasifikasikan berdasarkan jumlah bintang yang diberikan pada sebuah hotel. Jumlah bintang ini bergantung pada kelengkapan fasilitas dan kondisi bangunan, perlengkapan dan pengelolaan, serta mutu pelayanan. Beberapa kategori hotel berdasarkan bintang yaitu:

   a. Hotel Bintang 1

      Memiliki kamar mandi di dalam dengan minimal 15 kamar standar, serta ukuran kamar standar minimal 20 m^2. Pada beberapa hotel bintang satu juga disediakan fasilitas *meeting room*. Hotel ini biasanya dikelola langsung oleh si pemilik [24].

   b. Hotel Bintang 2

      Kamar minimal yang dimiliki yaitu sebanyak 20 kamar standar dengan ukuran luas minimum 22 m^2 dan minimal 1 *suite* dengan ukuran luas minimum 44 m^2, dan memiliki kamar mandi, AC, telepon, wifi, dan juga televisi di dalam kamar.

   c. Hotel Bintang 3

      Memiliki minimal 30 kamar standar, 2 suite, dan memiliki kamar mandi, telepon, televisi, dan AC di dalam kamar. Ukuran minimal untuk sebuah kamar standar yaitu 24 m^2 dan ukuran untuk kamar *suite* memiliki luas minimal 48 m^2. Di dalam hotel juga terdapat fasilitas seperti restoran, bar, fasilitas olahraga, area rekreasi, serta staf pramutamu.

   d. Hotel Bintang 4

      Hotel ini memiliki minimal 59 kamar standar dan 3 *suite* dengan luas minimal yang sama dengan hotel bintang 3. Tersedia kamar mandi dengan air panas dan dingin, telepon, AC, televisi di dalam kamar dan bangunan hotel dilengkapi dengan lobi yang memiliki luas minimal 100 m^2, rekreasi kegiatan, fasilitas olahraga, restoran, tempat istirahat, toilet umum, bar, dan staf pramutamu hotel.

   e. Hotel Bintang 5

      Memiliki minimal 100 kamar standar, minimal 4 *suite* dengan ukuran minimal untuk kamar standar adalah 26 m^2 dan 52 m^2 untuk kamar *suite*. Terdapat juga fasilitas kamar mandi dengan air panas dan dingin, telepon, televisi, AC dan terdapat lobi dengan luas minimal 100 m^2, restoran, bar, layanan kamar 24 jam, fasilitas olahraga, ruang hiburan, tempat peristirahatan, toilet umun, dan layanan pramutamu hotel.

2. Berdasarkan *Plan*

   a. *The American Plan*

Untuk biaya hotel diorganisasikan di mana biaya untuk menginap mencakup akomodasi dan biaya makan. *Plan* ini terdiri dari dua bagian, yaitu: *Full American Plan* (FAP) dan *The Modiefied American Plan* (MAP). Tiga kali makan (sarapan, makan siang, dan makan malam) juga termasuk dalam biaya akomodasi.

b. Paket Bermuda

Pengaturan penginapan di mana biaya akomodasi sudah termasuk sarapan kontinental.

c. Strategi Eropa

Tamu yang menginap cukup mambayar akomodasi sehingga ketika *check-out*, penagihan lebih praktis.

3. Berdasarkan Ukuran Hotel

a. *Small Hotel*

Hotel kecil dengan jumlah kamar kurang dari 150 kamar.

b. *Medium Hotel*

Dibagi menjadi dua bagian:

- *Average Hotel*: jumlah kamar antara 150 sampai dengan 299 kamar.
- *Above Average Hotel*: jumlah kamar antara 300 sampai dengan 600 kamar.

c. *Large Hotel*

Hotel yang memiliki jumlah kamar lebih dari 600.

4. Berdasarkan Lokasi

a. *City Hotel*

Merupakan hotel yang terletak di sebuah kota yang sebagian besar pengunjungnya bepergian untuk urusan bisnis.

b. *Resort Hotel*

Terletak di suatu lokasi wisata yang popular, di mana sebagian besar pengunjung datang untuk bersantai daripada bisnis.

5. Berdasarkan Rata-Rata Lama Menginap

a. *Transient Hotel*

Hotel ini biasanya ditempati oleh wisatawan dengan rata-rata lama menginap adalah satu malam, biasanya disediakan bagi tamu yang membutuhkan akomodasi sementara untuk liburan ataupun pebisnis dalam perjalanan singkat.

b. *Semi-Resident Hotel*

Rata-rata lama menginap di hotel ini adalah lebih dari satu malam, tetapi masih hanya dalam jangka waktu satu minggu hingga satu bulan.

c. *Resident Hotel*

Hotel ini disediakan bagi para pengunjung yang menginap selama satu bulan atau lebih yang ingin menetap di suatu tempat untuk jangka waktu yang lama.

2.2 Aplikasi

Banyak aplikasi yang dikembangkan pada zaman modern ini sangat membantu kehidupan manusia dalam melakukan komunikasi, bekerja, dan juga menambah wawasan [25]. Sebuah program aplikasi adalah perangkat lunak komputer yang menggabungkan fitur-fitur tertentu sedemikian rupa sehingga dapat diakses oleh pengguna atau dapat digunakan untuk aplikasi lain [26]. Berdasarkan *platform*, aplikasi dapat dibagi menjadi 3 yaitu:

1. Aplikasi Desktop

   Aplikasi desktop merupakan suatu aplikasi atau *software* yang berjalan pada desktop (PC dan laptop). Umumnya, apliksai jenis ini beroperasi tanpa terhubung dengan koneksi internet, sehingga bisa dijalankan secara *offline*. Beberapa bahasa pemograman yang dapat digunakan untuk mengembangkan aplikasi jenis ini adalah *C#*, *Java*, dan *Delphi*. Contoh aplikasi desktop diantaranya [25]:

   - Microsoft Word, Porwerpoint, Excel
   - Notepad
   - Adobe Photoshop
   - Paint
   - Corel Draw
   - dan lain-lain

2. Aplikasi Web

   Aplikasi web adalah perangkat lunak yang berjalan pada peramban web. Aplikasi web menjalankan fungsi tertentu di peramban web dan memungkinkan *user* untuk melakukan mengakses aplikasi secara *online*. Sebuah aplikasi web membutuhkan *web server* yang digunakan untuk menangani permintaan (*request*) dari *client*, *application server* untuk mengeksekusi tugas yang diminta, dan *database* untuk menyimpan informasi [27]. Ciri utama yang membuat aplikasi web lebih banyak diminati adalah karena aksesnya yang mudah dan dapat diimplementasikan pada berbagai bidang kehidupan. Aplikasi berbasis web biasanya dikembangkan dengan menggunakan *HTML*, *CSS*, dan *JavaScript* [25]. Cara kerja sebuah aplikasi web adalah sebagai berikut:

   1. *User* membuat permintaan ke server web pada internet melalui sebuah *application interface*.

2. Server web akan mengirimkan permintaan ke server aplikasi web.
3. Server aplikasi web akan mengeksekusi tugas yang diminta dan menghasilkan data yang dibutuhkan.
4. Server aplikasi web akan mengirimkan kembali hasil data ke server web.
5. Server web akan membawa informasi yang diminta oleh *user* dan memunculkannya pada tampilan layar *user*.

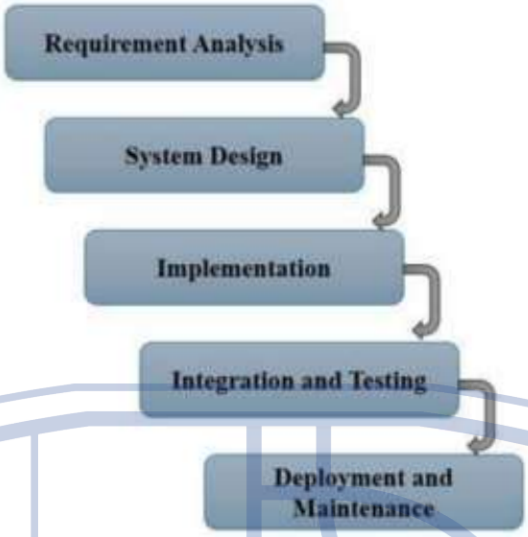
3. Aplikasi *Mobile*

Aplikasi *mobile* adalah sebuah perangkat lunak yang berjalan pada sebuah peralatan *mobile*. Pada tahun 2017, 178,1 milliar aplikasi *mobile* diunduh, dan jumlah ini diprediksi akan teruh meningkat hingga 258,2 milliar pada tahun 2022. Jutaan aplikasi *mobile* tersedia di *app stores* seperti *Google Play* dan *Apple App Store* [28]. Pengembangan aplikasi *mobile* sedang berkembang dengan pesat. Dari bidang ritel, telekomunikasi, *e-commerce*, asuransi, pelayanan kesehatan, pemerintahan, organisasi di seluruh industri harus memenuhi ekspetasi *user* akan cara yang mudah dan *real-time* dalam melakukan transaksi dan mengakses informasi [29]. Perbedaan antara aplikasi *mobile* dengan aplikasi web adalah aplikasi *mobile* perlu diunduh untuk dapat diakses, dan sebagian besar aplikasi *mobile* tidak membutuhkan koneksi internet untuk digunakan, sedangkan aplikasi web awalnya dirancang untuk komputer desktop yang kini diadaptasi untuk format perangkat seluler [30].

2.3 *Waterfall Model*

Model *Waterfall* dikembangkan pada tahun 1970an didasarkan pada pendekatan rekayasa terstruktur yang digunakan dalam industri konstruksi dan manufaktur. Model ini menekankan proses pengembangan perangkat lunak yang sistematis dan dapat diprediksi. Model *Waterfall* dikenal karena kesederhanaan, kejelasan penjadwalan, dan kemudahan dalam perencanaan anggaran, terutama untuk proyek dengan kebutuhan dan hasil yang pasti [31].

Metode *Waterfall* melakukan pendekatan yang bersifat berurutan dan bertahap. Pada metode *Waterfall*, setiap fase harus diselesaikan secara menyeluruh sebelum masuk ke fase berikutnya [32].



Gambar 2.1 Metode *Waterfall* [32]

Adapun tahapan dari metode *Waterfall* adalah sebagai berikut [33]:

1. Analisis Kebutuhan

   Pada tahap awal ini penting untuk menentukan cakupan dan kebutuhan dari proyek. Data dikumpulkan dan diolah untuk mendapatkan informasi tentang kebutuhan pengguna.

2. Perancangan Sistem

   Setelah kebutuhan sistem didefinisikan dengan jelas, tahapan selanjutnya adalah merancang kerangka sistem. Pada tahapan ini dilakukan perancangan desain tampilan dan juga *database*.

3. Implementasi

   Pengkodean atau perancangan sistem informasi menjadi suatu program dilakukan untuk memenuhi kebutuhan analisis yang dilakukan sebelumnya.

4. Integrasi dan *Testing*

   Tahapan ini dilakukan untuk memastikan bahwa sistem informasi dibuat dengan benar dan sesuai dengan ketentuan yang telah ditetapkan.

5. *Deployment* dan *Maintenance*

   Sistem yang dihasilkan diimplementasikan kepada pengguna, dan dilakukan pemeliharaan sesuai kebutuhan.

2.4 *Machine Learning*

Teknik *machine learning* telah menjadi alat revolusioner dalam penelitian ilmu material yang memungkinkan analisis yang cepat dan tepat terhadap sifat material yang rumit, memprediksi *item* baru berdasarkan sifat yang diinginkan dan mengoptimalkan

prosedur fabrikasi. Algoritma ini dapat mengenali sebuah pola, memprediksi sesuatu dari contoh yang diberikan, dan berubah seiring waktu [34]. Dunia digital yang tengah berkembang sangat kaya akan data seperti, data *Internet of Things* (IoT), data keamanan siber, data seluler, data bisnis, data media sosial, data kesehatan, dan lain sebagainya. Untuk menganalisa data-data ini dengan cerdas dan mengembangkan sebuah aplikasi yang cerdas dan otomatis yang sesuai, pengetahuan mengenai *artificial intelligence* (AI), khususnya *machine learning* adalah kuncinya [35]. Dalam proses pengembangan dengan *machine learning*, terdapat *workflow* sebagai berikut [36]:

1. Pengembangan Model (*Model Development*)
    a. Pengumpulan Data dan *Preprocessing* (*Data Collection and Preprocessing*)
    Mengumpulkan data yang relevan dan melakukan pembersihan data untuk memastikan data sejalan dengan kebutuhan model.
    b. Rekayasa Fitur (*Feature Engineering*)
    Merekayasa fitur untuk meningkatkan kemampuan model dalam menangkap pola dan hubungan dalam data.
    c. Pelatihan Model (*Model Training*)
    Melatih model *machine learning* menggunakan data histori, serta menyesuaikan parameter untuk kinerja yang optimal.
2. Evaluasi Model (*Model Evaluation*)
    a. Validasi dan Pengujian (*Validation and Testing*)
    Menilai performa model menggunakan dataset validasi, serta menyempurnakan parameter jika diperlukan.
    b. Metrik dan Evaluasi (*Metrics and Evaluation*)
    Menentukan metrik evaluasi berdasarkan masalah yang dihadapi, dan memastikan keselarasan dengan tujuan bisnis.
3. Pengemasan Model (*Model Packaging*)
    a. Serialisasi Model (*Model Serialization*)
    Melakukan serialisasi model yang telah dilatih ke dalam format yang mudah disimpan dan dipindahkan.
    b. Manajemen Dependensi (*Dependency Management*)
    Mengelola *file* dan dependensi seperti *library* dan *framework*.
4. Kontainerisasi (*Containerization*)
    a. Dokernisasi (*Dockerization*)

Mengemas model beserta dependensinya ke dalam kontainer Docker untuk memastikan konsistensi di berbagai lingkungan.

b. Orkestrasi Kontainer (*Container Orchestration*)

Menggunakan alat orkestrasi kontainer (misalnya Kubernetes) untuk penerapan dan penskalaan yang efisien.

5. Integrasi dengan *Pipeline* Pengembangan (*Integration with Deployment Pipeline*)
   a. *Continuous Integration* (CI) / *Continuous Deployment* (CD)

   Mengintegrasikan proses *deployment machine learning* ke dalam *pipeline* CI/CD untuk pengujian dan penerapan otomatis.

   b. Kontrol Versi (*Control Version*)

   Menjaga kontrol versi untuk model dan kode terkait agar perubahan dapat dilacak dan memudahkan *rollback* bila diperlukan.
6. Pengembangan untuk Produksi (*Deployment to Production*)
   a. *Staging Deployment*

   Men-*deploy* model ke lingkungan *staging* untuk memvalidasi performa dalam situasi yang menyerupai produksi.

   b. Peluncuran Bertahap (*Gradual Rollout*)

   Merilis model secara bertahap ke sebagian pengguna untuk memantau perilakunya dan mengidentifikasi potensial masalah.

   c. *Full Deployment*

   Setelah validasi berhasil, model di-*deploy* ke seluruh lingkungan produksi.
7. Pemantauan dan Pemeliharaan (*Monitoring and Maintenance*)
   a. Pemantauan Performa (*Performance Monitoring*)

   Memantau performa model secara berkelanjutan, termasuk akurasi, waktu respons, dan penggunaan sumber daya.

   b. Penanganan *Error* (*Error Handling*)

   Mengimplementasikan mekanisme penanganan *error* yang kuat untuk mengatasi masalah tak terduga dan mencegah gangguan layanan.

   c. *Feedback Loop*

   Membangun *feedback loop* untuk menangkap umpan balik pengguna dan deteksi perubahan data dan melakukan pelatihan ulang model jika diperlukan.
8. Keamanan dan Kepatuhan (*Security and Compliance*)
   a. Kontrol Akses (*Access Controls*)

Mengimplementasikan kontrol akses untuk membatasi akses model hanya kepada pengguna yang berwenang.

b. Privasi Data (*Data Privacy*)

Memastikan kepatuhan terhadap regulasi privasi data dan melindungi informasi sensitif selama proses *deployment* model.

9. Dokumentasi dan Berbagi Pengetahuan (*Documentation and Knowledge Sharing*)
   a. Dokumentasi (*Documentation*)

   Menjaga dokumentasi yang komprehensif untuk model yang di-*deploy*, termasuk arsitektur model, dependensi, dan prosedur *deployment*.

   b. Transfer Pengetahuan (*Knowledge Transfer*)

   Saling berbagi ilmu antar anggota tim agar manajemen model dapat berjalan dengan baik.
10. Skalabilitas dan Optimasi (*Scalability and Optimization*)
    a. Perencanaan Skalabilitas (*Scalability Planning*)

    Merancang arsitektur *deployment* dengan mempertimbangkan skalabilitas, agar sistem mampu menangani peningkatan beban kerja.

    b. Optimasi (*Optimization*)

    Melakukan evaluasi dan optimasi model secara berkala untuk efisiensi dan pemanfaatan sumber daya yang lebih baik.

Pada *machine learning*, terdapat tiga jenis pembelajaran yang menjadi landasan dari berbagai aplikasi, yaitu [37]:

1. *Supervised Learning*

   Supervised Learning adalah metode *machine learning* di mana algoritma dilatih menggunakan data berlabel. Tujuannya adalah untuk mendapatkan data klasifikasi baru ataupun membuat prediksi terhadap pola yang dipelajari selama pelatihan data [37]. Melalui proses iterasi, model ditingkatkan dan dioptimalisasi, mempelajari dan menghasilkan pola dari data yang diberikan untuk meningkatkan kemampuan prediksi [38]. Pada pendekatan sistem rekomendasi *model-based collaborative filtering*, konsep *machine learning* dengan metode *supervised learning* diterapkan untuk melakukan regresi dalam hal prediksi *rating* [39]. Beberapa algoritma pada *supervised learning* diantaranya [37]:

   1. *Linear Regression*

      Linear regression merupakan algoritma fundamental pada *supervised learning* yang digunakan untuk memodelkan hubungan antara variabel dependen (target) dengan

satu atau lebih variabel independent (fitur). Metode ini umumnya diterapkan untuk menyelesaikan masalah regresi, di mana tujuannya adalah untuk memprediksi output numerik yang bersifat kontinu [37].

2. *Logistic Regression*

   Logistic Regression lebih cocok digunakan untuk variabel dependen yang bersifat kontinu. Regresi jenis ini biasanya digunakan untuk menangani masalah biner, seperti identifikasi spam [37].

3. *Decision Trees*

   Decision Trees secara luas banyak digunakan dalam *supervised learning* untuk klasifikasi dan juga regresi. Algoritma ini bekerja dengan cara membagi data secara rekrusif menjadi subset yang lebih kecil berdasarkan nilai dari fitur masukan. Setiap node internal merepresentasikan sebuah keputusan berdasarkan sebuah fitur spesifik, *branches* sebagai kemungkinan hasil dari keputusan tersebut, dan *leaf nodes* mengilustrasikan hasil prediksi atau klasifikasi [40].

4. *Support Vector Machine* (SVM)

   Algoritma SVM dikembangkan oleh Vladimir Vapnik merupakan salah satu algoritma yang terkenal dengan penerapan pada klasifikasi dan regresi data. Namun, penerapan utamanya adalah pada masalah klasifikasi. Model ini menciptakan sebuah *hyperplane* yang memaksimalkan jarak antara dua kelompok data. *Decision boundary* merupakan *hyperplane* yang memisahkan dua kelompok data menjadi kelompok yang berbeda [37].

5. *Naïve Bayes*

   Naïve Bayes merupakan pendekatan klasifikasi yang didasarkan pada fondasi asumsi independensi bersyarat (*conditional independence*) antar kelas. Asumsi ini menyatakan bahwa pengaruh setiap prediktor dianggap sama dan keberadaan sebuah atribut tidak mempengaruhi atribut lainnya dalam menentukan kemungkinan terjadinya suatu peristiwa tertentu [37].

2. *Unsupervised Learning*

   Metode *unsupervised learning* berfokus dalam mengungkapkan pola, struktur, ataupun hubungan dari data tidak berlabel. Tujuan dari metode ini adalah untuk mendapatkan informasi dari data dengan melakukan *clustering*, *dimensionality reduction*, ataupun *anomaly detection* [37]. Teknik utama yang digunakan pada *unsupervised learning* meliputi *principal component analysis* dan *cluster analysis* [38]. Contoh algoritma pada *unsupervised learning* diantaranya [37]:

1. *K-Means*

   K-Means adalah algoritma *clustering* yang digunakan dalam *unsupervised learning* yang bekerja dengan cara mengelompokkan data ke dalam "k" klaster, di mana "k" ditentukan oleh pengguna. Tujuannya adalah untuk menetapkan setiap titik data ke klaster dengan pusat (*centroid*) terdekat, sehingga meminimalkan jarak antara titik data dan *centroid* masing-masing [41].

2. *Hierarchical Clustering*

   Algoritma ini membagi data ke dalam hierarki klaster, sering direpresentasikan dalam bentuk struktur seperti pohon (*dendogram*), di mana titik-titik data yang serupa dikelompokkan bersama. Klaster baru terbentuk dengan cara penggabungan dua *dataset*, memasukkan data baru ke dalam klaster yang sudah ada, ataupun dengan menggabungkan dua klaster secara iteratif [37].

3. *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN)

   DBSCAN adalah algoritma *clustering* yang membedakan antara klaster dengan kepadatan tinggi(*high-density*) dengan kepadatan rendah (*low-density*). DBSCAN adalah algoritma berbasis kepadatan (*density-based clustering*), sebuah alat yang kuat untuk menumkan klaster dengan observasi yang serupa pada *dataset* yang besar. Tujuan utama dari algoritma DBSCAN adalah untuk mengidentifikasi klaster yang terdiri dari titik-titik yang padat yang dipisahkan oleh area dengan kepadatan rendah [37].

3. *Reinforcement Learning*

   Reinforcement Learning merupakan salah satu cabang dari *machine learning* yang telah mendapatkan perhatian besar dalam beberapa tahun terakhir [42]. Tujuan dari jenis pembelajaran ini adalah untuk mengajarkan agen agar dapat mengambil tindakan dalam suatu lingkungan yang tidak diketahui untuk memaksimalkan *reward* siring waktu. Mesin belajar melalui proses *trial and error* untuk memetakan situasi tertentu untuk memperoleh *reward* tertinggi [43]. Contoh dari algoritma *reinforcement learning* yaitu [41]:

   1. *Policy-Based Reinforcement Learning*

      Pada algoritma jenis ini, *agent* secara langsung mempelajari *policy network* yang memetakan keadaan (*state*) ke tindakan (*action*). Proses ini dimulai Ketika "*Environment*" memberikan *agent* sebuah "*State*". *Agent* kemudian menggunakan *policy network* yang diberikan untuk menghasilkan "*Action Probabilities*" dan memilih sebuah "*Action*" berdasarkan probabilitas tersebut melalui "*Action Selection*

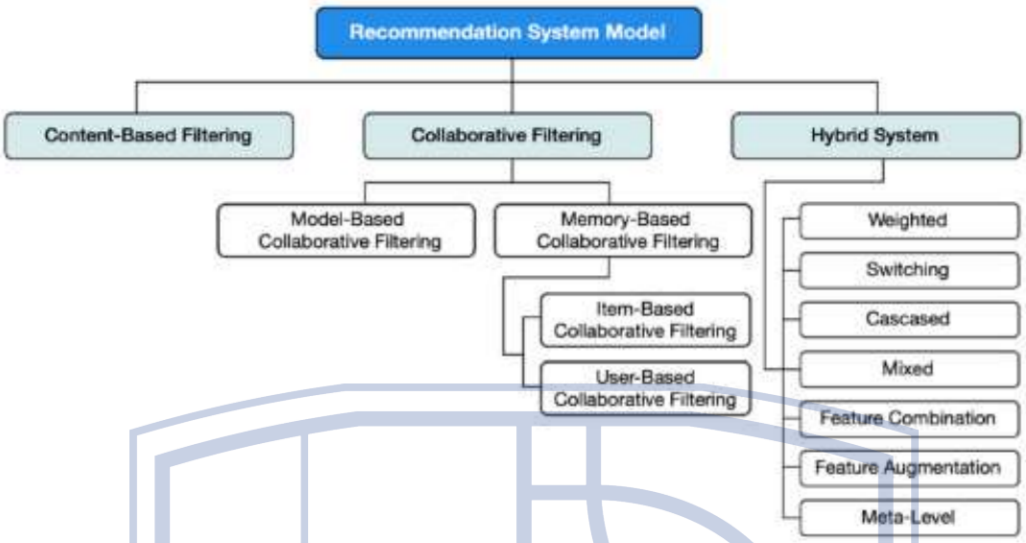
Mechanism". *Environment* kemudian akan merespon dengan memberikan "*Reward Signal*" yang digunakan oleh "*Policy Optimization Algorithm*" untuk memperbarui *policy network* dengan tujuan memaksimalkan *reward* di masa depan [41].

2. *Value-Based Reinforcement Learning*

   Pada *value-based reinforcement learning*, *agent* mempelajari "*Value Function*" yang memperkirakan potensi *reward* jangka panjang dari berbagai keadaan (*state*) dan tindakan (*action*). "*Environment*" akan memberikan "*State*", lalu *agent* akan menggunakan *value function* untuk memilih *action* berdasarkan tindakan yang memaksimalkan pengembalian yang diharapkan di masa depan. Berbeda dengan *policy-based* yang mempelajari *policy network* untuk memetakan *state* ke *action*, *value-based* berfokus pada estimasi nilai dari *state* dan *action* yang memungkinkan pengambilan keputusan yang lebih informatif [41].

2.5 Sistem Rekomendasi

Sistem rekomendasi adalah sebuah sistem bantuan keputusan yang digunakan untuk membantu *user* dalam menentukan dan memilih *item* yang cocok [44]. Sistem rekomendasi memainkan peran penting dalam masalah kelebihan informasi dengan menyaring dan memperoleh informasi serta layanan yang paling relevan dari sejumlah besar data, sehingga didapatkan layanan yang dipersonalisasi. Sistem ini bekerja dengan menyaring informasi dari berbagai sumber dan memprediksi keluaran berdasarkan informasi yang terkait dengan pengguna, *item*, dan interaksi di antara mereka [45]. Pada saat ini, sistem rekomendasi semakin banyak digunakan untuk banyak aplikasi seperti, *web*, buku, *e-learning*, pariwisata, film, musik, *e-commerce*, berita, program televisi, dan sebagainya. Sistem rekomendasi menangani dua hal yaitu, *user* dan *item*, di mana setiap *user* memberikan *rating* (atau nilai preferensi) untuk suatu *item* (atau produk). *Rating* yang diberikan oleh pengguna umumnya dikumpulkan dengan dua metode yaitu, implisit dan eksplisit. Pada metode implisit, *rating* dikumpulkan secara tidak langsung melalui interaksi *user* dengan *item*, sedangkan secara eksplisit, *rating* diberikan langsung oleh *user* dengan memilih nilai pada skala tertentu yang terbatas atau nilai interval yang berlabel. Sebagian besar sistem rekomendasi mengumpulkan *rating* pengguna melalui metode eksplisit maupun implisit [46].



Gambar 2.2 *Recommendation models* [47]

Ada tiga jenis metode yang digunakan dalam sistem rekomendasi yaitu [9]:

1. *Collaborative Filtering*

   Collaborative filtering adalah model filter informasi yang pertama kali muncul pada tahun 1990an dan menjadi batu loncatan untuk penelitian mendatang mengenai sistem rekomendasi. Pendekatan model ini memanfaatkan pengukuran kesamaan antar *user*. Efisiensi dari *collaborative filtering* bergantung pada seberapa akurat algoritma pada model ini dapat menemukan keanggotaan sasaran *user*. Model ini tidak memerlukan informasi mendetail dari sebuah *item* untuk memberikan rekomendasi, dan model ini juga dapat membantu memperluas minat pengguna yang sudah ada dengan *item* yang baru sehingga tidak monoton [46].

2. *Content-based Filtering*

   Pada *content-based filtering, item* yang direkomendasikan kepada *user* adalah *item* yang memiliki kualitas yang sebanding yang sudah ada dalam koleksi *user*. Rekomendasi yang dihasilkan dari model ini diperoleh dari informasi pada karakteristik suatu barang [48]. Keterbatasan yang ada pada model ini adalah kebutuhan representasi yang kaya akan suatu barang, yang mana tidak cocok untuk *item* pariwisata yang memiliki karakteristik yang luas dan bervariasi [49].

3. *Hybrid System*

   Sistem rekomendasi *hybrid* adalah sebuah teknik yang dikembangkan dengan tujuan untuk mengatasi kendala yang dihadapi ketika menggunakan *content-based filtering* dan *collaborative filtering* [48]. Hasil dari rekomendasi ini dapat diperoleh dengan teknik

yang terpisah, atau menggunakan *content-based filtering* di dalam *collaborative filtering*, begitu juga sebaliknya [46].

2.6 *Collaborative Filtering*

Collaborative filtering adalah model yang membangun sebuah *database* preferensi *user* menggunakan data evaluasi *user* untuk memprediksi *item* yang sesuai dengan selera *user*, lalu menggunakannya untuk rekomendasi [47]. Metode *collaborative filtering* akan melakukan proses penyaringan data berdasarkan *profile* tingkah laku karakteristik pengguna sistem. Dengan demikian, sistem akan dapat memberikan informasi baru kepada pengguna lainnya, dikarenakan sistem memberikan informasi berdasarkan pola satu kelompok pengguna yang *match* (mirip) [50]. Model rekomendasi ini dapat diklasifikasi menjadi dua yaitu [47]:

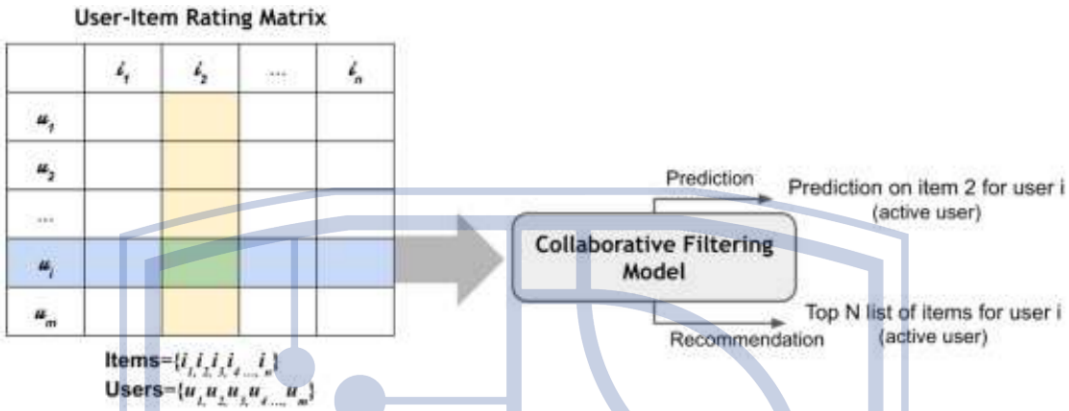
1. *Memory-Based Collaborative Filtering*

   Metode ini memanfaatkan kesamaan antar pengguna atau *item* untuk mengambil informasi bagi pengguna target dan membuat rekomendasi berdasarkan hasil yang diperoleh [51]. Pendekatan model ini merekomendasikan *item* baru dengan mempertimbangkan preferensi lingkungannya menggunakan matriks utilitas secara langsung untuk prediksi [46]. *Memory-based collaborative* terbagi lagi menjadi dua tipe yaitu: *user-based collaborative filtering* dan *item-based collaborative filtering*. Membandingkan kesamaan antar *user* dilakukan melalui *user-based collaborative filtering* dengan memeriksa *rating* yang diberikan pengguna untuk hal yang sama. Melalui riwayat *rating* yang diberikan *user*, *item-based collaborative filtering* memprediksi *item* yang mungkin disukai *user* dengan memanfaatkan kesamaan antar objek. Di sisi lain, metode ini tentunya memiliki kelemahan, seperti *gray sheep*, *cold start*, dan *sparsity* (banyaknya data yang kosong) [48].

2. *Model-based Collaborative Filtering*

   Model-based Collaborative Filtering membangun sebuah model yang dapat memprediksi *rating* atau preferensi dari *user* yang menjadi sasaran untuk sebuah *item* dan memberikan rekomendasi berdasarkan estimasi *rating* [51]. Teknik *clustering*, *single-variable decomposition* (SVD), dan *principal component analysis* (PCA) adalah beberapa teknik yang digunakan dalam pengembangan metode ini [48]. Teknik *model-based* mampu untuk mengatasi masalah umum pada sistem rekomendasi seperti *sparsity* dan skalabilitas dengan menerapkan teknik pengurangan dimensi dan pembelajaran model. Metode ini juga tidak memerlukan penambahan profil *user* dari seorang *user* baru

pada matriks utilitas sebelum membuat prediksi. Rekomendasi dapat dibuat bahkan untuk user yang tidak ada di dalam model. Sistem *model-based* menggunakan berbagai algoritma *machine learning* untuk mengembangkan sebuah model dalam memprediksi *rating user* untuk *item* yang belum diberi *rating* [46].



Gambar 2.3 *Recommendation Principle of Collaborative Filtering Model* [47]

Dikarenakan sistem rekomendasi *collaborative filtering* menggunakan interaksi data antara *user* dengan *item* untuk mengenali pola dan kesamaan antar *user* dan *item*, maka model CF dapat mengalami masalah *cold start* yang mana sistem kesulitan untuk memberikan rekomendasi untuk *user* atau *item* baru dengan data yang terbatas [52]. Dengan dimikian, data *user* atau *item* harus dikenali terlebih dahulu oleh model agar bisa memberikan rekomendasi kepada pengguna, jika tidak maka model akan kesulitan dalam memberikan rekomendasi yang sesuai dengan keinginan pengguna [53]. Seiring dengan meningkatnya kebutuhan akan rekomendasi yang lebih personal, metode *collaborative filtering* juga terus berkembang. Salah satu pengembangannya adalah multi-kriteria *collaborative filtering*, yaitu pendekatan rekomendasi di mana sistem mempertimbangkan berbagai aspek penilaian terhadap suatu *item*, contohnya pertimbangan kebersihan, lokasi, kamar, dan pelayanan dalam konteks rekomendasi hotel [14].

2.7 *Singular Value Decomposition*

Singular Value Decomposition (SVD) adalah salah satu model *matrix factorization* yang paling banyak digunakan dalam aplikasi *data science*. Pada metode SVD, terdapat suatu matriks $A \in R^{m \times n}$ yang dibagi menjadi tiga komponen seperti pada persamaan berikut [54]:

$$A_{mxn} = U_{mxm}\, \Sigma_{mxn}\, V^{T}_{nxn} \quad \text{.....(1)}$$

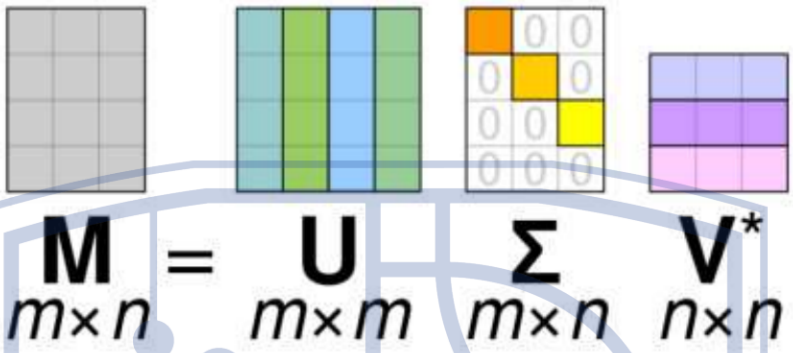
Di mana:

A = Matriks berukuran m x n

U = Matriks ortogonal m x m $[u_1, \ldots, u_r]$

Σ = Matriks diagonal m x n $diag(\sigma_1, \ldots, \sigma_r)$

V^T = Matriks ortogonal n x n $[v_1, \ldots, v_r]$



Gambar 2.4 Persamaan pada SVD [55]

Dalam pengukuran matriks yang besar, untuk sebagian besar kasus, algoritma *truncated* SVD digunakan [54]. Pada metode ini, pilih nilai k di mana k < rank(A), sehingga persamaan yang diperoleh menjadi sebagai berikut:

$$A_{mxn} = U_{mxk}\, \Sigma_{kxk}\, V^T_{kxn} \quad \ldots\ldots(2)$$

Di mana:

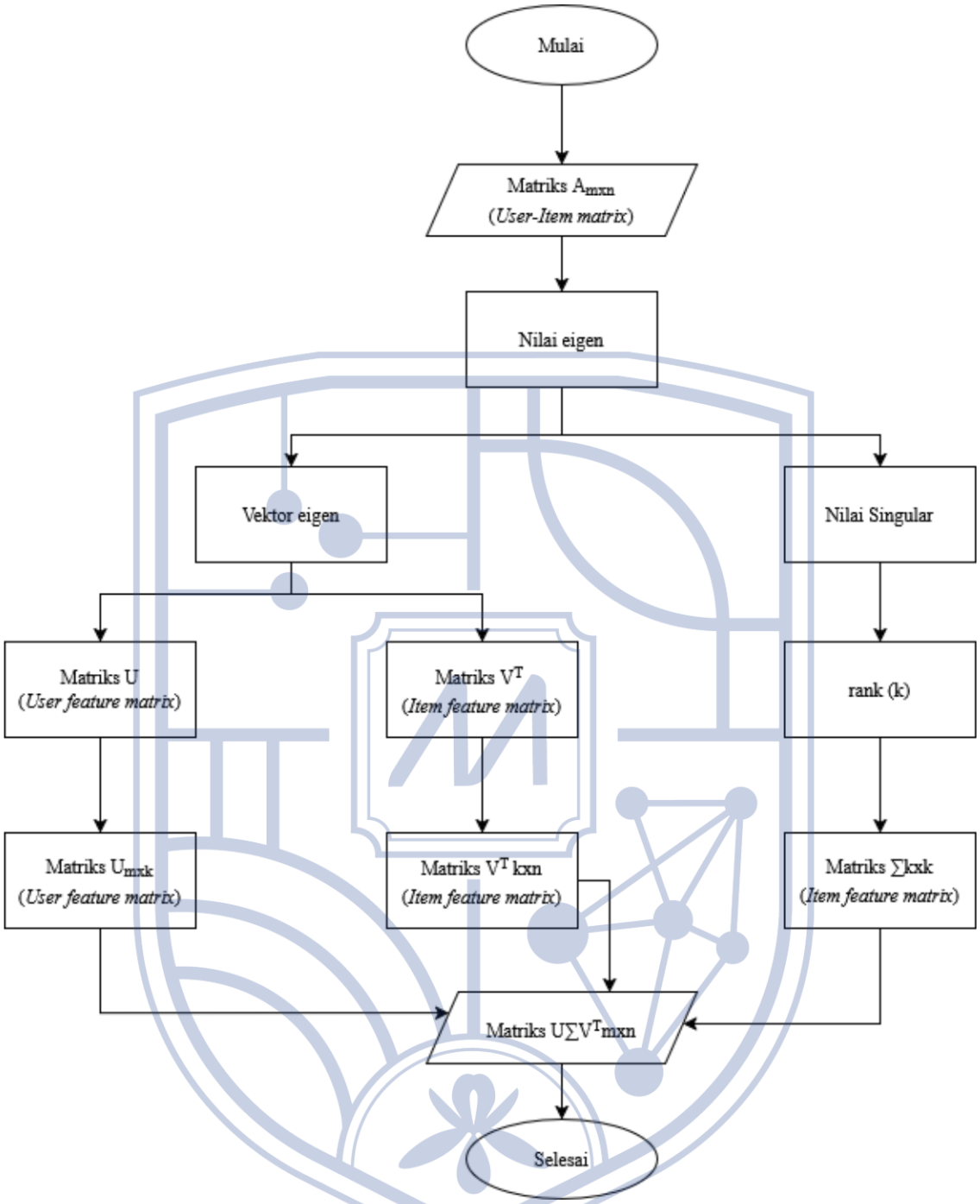
A = Matriks berukuran m x n

U = Matriks ortogonal m x k $[u_1, \ldots, u_k]$

Σ = Matriks diagonal k x k $diag(\sigma_1, \ldots, \sigma_k)$

V^T = Matriks ortogonal k x n $[v_1, \ldots, v_k]$

Berdasarkan persamaan (2), A adalah sebuah matriks m x n, di mana dalam contoh kasus ini m adalah pengguna dan n adalah hotel. Dalam SVD, nilai-nilai singular dari matriks A adalah $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k > \sigma_{k+1} = \ldots = \sigma_n = 0$. Misalkan $[v_1, \ldots, v_k]$ adalah himpunan vektor singular kanan dari A, di mana v_i berkorespondensi dengan σ_i, dan $[u_1, \ldots, u_k]$ merupakan anggota dari himpunan vektor singular kiri dari A. U adalah matriks ortogonal m x k dengan anggota vektor singular kiri. V adalah matriks ortogonal k x n dengan anggota vektor singular kanan, dan Σ mempresentasikan matriks diagonal k x k. Dikarenakan nilai rank(A) = k harus kurang dari atau sama dengan nilai m dan n, semua nilai k dari nilai singular yang bukan nol untuk A akan dijadikan sebagai diagonal utama dari Σ [56].



Gambar 2.5 Flowchart Algoritma SVD [56]

Langkah-langkah dalam perhitungan SVD untuk mendekomposisi A_{mxn} menjadi $U, \Sigma,$ V^{T} adalah sebagai berikut:

1. Untuk menghitung vektor pada singular kiri, hitung nilai-nilai eigen dari AA^T

$$|AA^T - \lambda I| = 0 \quad \ldots\ldots(3)$$

Di mana:

A = Matriks m x n

A^T = *Transpose* dari matriks A

λI = Nilai eigen

2. Menentukan vektor-vektor eigen $u1$, …, uk yang berkoresponden dengan nilai-nilai eigen dari AA^T

$$(AA^T - \lambda I)x = 0 \quad \text{.....(4)}$$

Di mana:

A = Matriks m x n

A^T = *Transpose* dari matriks A

λI = Nilai eigen

x = Vektor eigen

3. Normalisasi vektor eigen untuk $u1$, …, uk dengan persamaan berikut.

$$u_i = \frac{x_i}{\sqrt{\Sigma x_i^2}} \quad \text{.....(5)}$$

Di mana:

u_i = Kolom pada matriks U ke-i (*left singular value*)

x_i = Vektor eigen ke-i

4. Matriks U

Diperoleh matriks U yang terdiri dari vektor eigen yang telah dilakukan normalisasi (vektor eigen unit), susunannya pada matriks U adalah sebagai berikut.

$$U = [u_1 \quad u_2 \quad .... \quad u_x] \quad \text{.....(6)}$$

Matriks U juga dapat diperoleh dengan persamaan seperti di bawah ini.

$$u_i = \frac{1}{\sigma_i} A.v_i \quad \text{.....(7)}$$

Di mana:

U = Matriks m x n untuk vektor singular kiri

u_i = Kolom pada matriks U ke-i (*left singular value*)

v_i = Kolom pada matriks V ke-i (*right singular vector*)

5. Menghitung nilai singular

$$\sigma_i = \sqrt{\lambda I} \quad \text{.....(8)}$$

Di mana:

σ = Nilai singular

λI = Nilai eigen

6. Matriks Σ

   Setelah menghitung nilai-nilai singular, didapatkan elemen-elemen diagonal matriks Σ adalah nilai-nilai singular dari matriks A dengan susunan besar ke kecil.

   $$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_k) \quad (9)$$

   Di mana:

   Σ = Nilai-nilai singular

   σ = Nilai singular

7. Untuk menghitung vektor pada singular kanan, hitung nilai-nilai eigen dari $A^T A$.

   $$|A^T A - \lambda I| = 0 \quad (10)$$

   Setelah dilakukan perhitungan nilai singular, maka selanjutnya adalah menentukan vektor eigen $v_1, v_2, \dots, v_n$ dengan cara berikut.

   $$v_i = \frac{1}{\sigma_i} A^T . u_i \quad (11)$$

   Di mana:

   A = Matriks m x n

   A^T = *Transpose* dari matriks A

   λI = Nilai eigen

   v_i = Kolom pada matriks V ke-i (*right singular vector*)

   σ_i = Nilai singular ke-i

   u_i = Kolom pada matriks U ke-i (*left singular value*)

   Setelah matriks V didapatkan, nilai pada matriks di-*transpose*, dan dihasilkan matriks V^T.

8. Matriks U_{mxk}

   Matriks U yang telah didapatkan dipotong menjadi matriks U dengan kolom sebanyak *rank* (k), sehingga didapatkan matriks U_{mxk}.

9. Matriks V^T_{kxn}

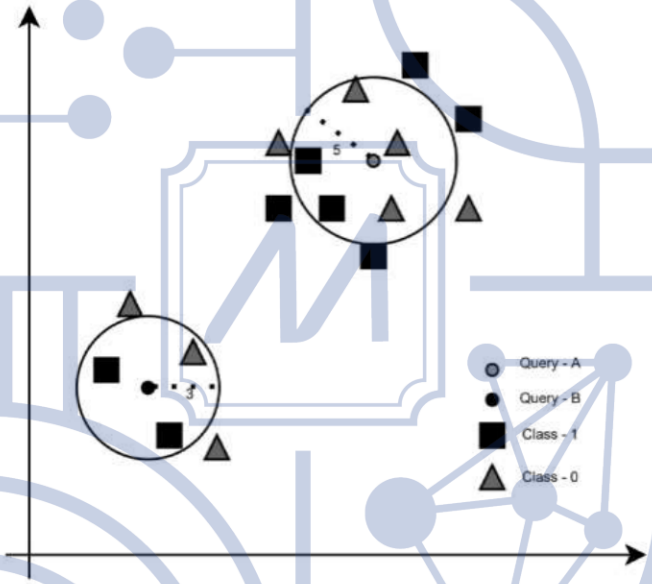
   Matriks V^T yang telah diperoleh dipotong menjadi matriks V^T dengan baris sebanyak *rank* (k), dan diperoleh matriks V^T_{kxn}.

10. Matriks $U\Sigma V^T$

    Setelah mendapatkan nilai pada setiap matriks U, matriks Σ, dan matriks V^T, ketiga matriks ini selanjutnya dikalikan untuk menghasilkan matriks A.

2.8 *K-Nearest Neighbor*

Algoritma *K-Nearest Neighbor* (KNN) adalah salah satu metode pengenalan pola yang terkenal dan merupakan salah satu algoritma klasifikasi teks. KNN adalah salah satu algoritma *machine learning* berbasis *supervised learning*, dan telah berkembang menjadi algoritma penting yang digunakan pada *data mining*, sistem rekomendasi, dan *Internet of Things* (IoT) [57]. Metode yang paling penting dalam *supervised learning* adalah klasifikasi, yang mana digunakan untuk melakukan prediksi. Berdasarkan data yang ada, algoritma memberikan *output* yang sepenuhnya bergantung pada kualitas *data training.* Ketika data baru diberikan, algoritma ini akan menghasilkan keluaran hanya berdasarkan pengalaman dan data sebelumnya [58].



Gambar 2.6 Ilustrasi Visual dari Algoritma KNN [59]

Seperti yang dapat dilihat pada Gambar 2.6, jika nilai k = 3 untuk *Query* B, maka algoritma KNN akan mencari 3 tetangga terdekat dari *Query* B, dan ditemukan dari 3 tetangga terdekat, 2 merupakan anggota dari *Class* 1, dan 1 merupakan anggota dari *Class* 0. Sama halnya untuk k = 5 pada *Query* A, di mana dapat dilihat bahwa lebih banyak tetangga terdekat yang dikarakterisasikan sebagai *Class* 0, maka *Query* A diklasifikasikan sebagai *Class* 0 [59]. Langkah-langkah dalam algoritma KNN adalah sebagai berikut [58]:

1. Tentukan nilai *k* sebagai jumlah dari anggota terdekat.
2. Menghitung jarak *Euclidean* dari objek dengan data sampel yang ada.

$$d_{(x,y)} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad \text{.....(12)}$$

Di mana:

$d_{(x,y)}$ = Jarak

x_i = Data pada sampel

y_i = Data yang akan diuji

n = Dimensi data

3. Setelah didapatkan nilai jarak antara data sampel dan uji, jarak diurutkan dari yang paling kecil (dekat) sampai yang paling besar (jauh).
4. Kumpulkan kategori dari data sampel yang ada.
5. Berdasarkan jumlah anggota terdekat, dalam hal ini *k*, tentukan kategori *nearest neighbor* yang paling banyak, maka hasil prediksi berupa klasifikasi dari jumlah nilai keanggotaan terdekat yang paling banyak.

2.9 Normalisasi Data

Normalisasi adalah suatu proses mentransformasikan nilai dari suatu objek ke dalam interval tertentu. Perbedaan nilai objek dapat menyebabkan pembobotan yang implisit dalam klasifikasi [60]. Normalisasi data dilakukan pada fase *preprocessing*, di mana merupakan proses yang sangat penting untuk meningkatkan performa *machine learning*. Penggunaan normalisasi data sangat penting dilakukan apabila menggunakan metode *data mining* yang melibatkan pengukuran jarak seperti KNN [61]. Ada beberapa metode normalisasi data diantaranya *Mean Centering* [62], *MinMax Normalization*, dan *Z-Score Normalization* [63].

2.9.1 *Mean Centering*

Mean Centering adalah metode normalisasi yang digunakan untuk menyeimbangkan data terhadap nilai rata-rata (*mean*) dari distribusi setiap sampel. Penggunaan *Mean Centering* bertujuan untuk mencegah *outlier* pada data, y akan mewakili nilai dari data, dan $\bar{y}$ mewakili rata-rata dari keseluruhan data pada sampel [62]. Normaliasi data dengan *Mean Centering* akan digunakan pada perhitungan SVD.

$$\tilde{y}_{ij} = y_{ij} - \bar{y}_i \quad (13)$$

Di mana:

$\tilde{y}_{ij}$ = Nilai hasil normalisasi

y_{ij} = Nilai yang akan dinormalisasi

$\bar{y}_i$ = Nilai rata-rata pada baris ke-i

2.9.2 *MinMax Normalization*

Pada penelitian yang dilakukan oleh [60], algoritma yang dianjurkan apabila menggunakan algoritma KNN untuk klasifikasi adalah *Standard*, *Robust*, dan *MinMax normalization*. *MinMax normalization* cenderung lebih sesuai untuk *dataset* dengan jumlah *instance* yang banyak, fitur yang banyak, dan terdistribusi tidak normal [61]. Oleh karena itu, pada penelitian ini, algoritma yang digunakan untuk normalisasi data pada metode KNN adalah *MinMax normalization*. Metode ini adalah metode yang merubah rentang nilai data menjadi 0 hingga 1. Persamaan dari *MinMax normalization* adalah sebagai berikut [64].

$$x' = \frac{x_i - \min(x)}{\max(x) - \min(x)} \quad \text{.....(14)}$$

Di mana:

x’ = Nilai hasil normalisasi

x_i = Nilai yang akan dinormalisasi

min(x) = Nilai terkecil yang ada pada atribut

max(x) = Nilai terbesar yang ada pada atribut

2.10 *Dataset*

Dataset merupakan komponen yang sangat penting dalam pengembangan kecerdasan buatan (AI). Dataset harus mencakup berbagai skenario, variasi, dan kompleksitas agar model yang dihasilkan mampu melakukan generalisasi yang baik. Ada banyak sumber dataset yang tersedia secara daring yang mendukung perkembangan penelitian AI [65]. Beberapa platform populer di antaranya adalah:

1. *Kaggle*

Sebuah platform komunitas bagi para ilmuwan data. *Kaggle* menawarkan ribuan dataset terbuka dari berbagai bidang, mulai dari pengenalan gambar, prediksi data tabular, hingga pemrosesan teks.

2. *Hugging Face Datasets*

Sumber dataset yang banyak digunakan dalam penelitian pemodelan bahasa alami (NLP) dan machine learning, dengan fokus pada teks, gambar, dan data multi-modal.

3. *UCI Machine Learning Repository*

Salah satu repositori dataset tertua dan paling sering digunakan oleh komunitas machine learning, menawarkan dataset dalam berbagai domain seperti kesehatan, bisnis, dan lingkungan.

4. *Open Data Portal*

Banyak pemerintah dan organisasi internasional menyediakan portal data terbuka, yang mencakup dataset di berbagai sektor seperti pendidikan, kesehatan, ekonomi, dan transportasi.

Di samping dari platform-platform di atas, *dataset* juga dapat diperoleh dengan melakukan *web scraping*. API dari sebuah website dapat digunakan untuk mendapatkan informasi dengan mudah dari sebuah *website*, namun tidak semua akses ke API dapat dilakukan karena dapat dibatasi oleh pemilik situs. *Web scraping* dinilai lebih cepat dan lebih efektif dalam mengumpulkan informasi dari ribuan, bahkan lebih dari jutaan halaman *web*. Teknik ini sangat penting dalam berbagai aplikasi, terutama dalam bidang *business intelligence* [66].

Setelah dilakukan *preprocessing* data, dataset diambil untuk melakukan *modelling*, yang mana dataset ini terbagi menjadi dua yaitu *training* data dan *testing* data. *Training* data merupakan dataset yang digunakan untuk membangun model, sementara *testing* data digunakan untuk menghitung *performance* dari model yang terbentuk dengan membandingkan label data sebenarnya dan label data hasil klasifikasi model [67].

2.11 Evaluasi Model

Evaluasi model pada tahapan *machine learning* berfungsi untuk menilai kinerja dari model dengan dataset percobaan untuk mengukur seberapa baik model dapat melakukan generalisasi terhadap data baru. Langkah ini sangat penting untuk memastikan bahwa model tidak mengalami *overfitting* terhadap data pelatihan dan layak digunakan dalam penerapan langsung [37]. *Mean Absolute Error* (MAE) adalah metode umum yang digunakan untuk menghitung rata-rata perbedaan antara nilai yang diprediksi dan nilai yang diketahui. Dalam mengevaluasi sebuah model klasifikasi, metrik yang umum digunakan adalah *accuracy, precision*, dan *recall* [68].

2.11.1 *Mean Absolute Error*

Mean Absolute Error (MAE) adalah sebuah metode pengukuran yang banyak digunakan untuk evaluasi model [69]. Nilai MAE menunjukkan nilai pengukuran kesalahan rata-rata absolut dalam prediksi statistik atau *machine learning*. Metrik pengukuran MAE menghitung rata-rata dari selisih absolut antara nilai yang diprediksi dan nilai sebenarnya. Rumus perhitungan MAE adalah sebagai berikut [70].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| \quad (15)$$

Di mana:

f_i = Nilai hasil peramalan

y_i = Nilai sebenarnya

n = Banyaknya data

Semakin kecil nilai MAE menunjukkan perbedaan antara nilai prediksi dan nilai actual minimal. Hasil 0 pada MAE menunjukkan nilai terbaik (tidak ada kesalahan), dan +∞ menandakan nilai yang buruk untuk sebuah model [71].

2.11.2 *Confusion Matrix*

Confusion Matrix merupakan cara untuk memvisualisasi kerja yang diharapkan dari model prediktif dalam data *mining*. Setiap data dari setiap kelas dalam tabel *confusion matrix* menunjukkan jumlah prediksi yang dibuat untuk mengklasifikasikan kelas sebagai benar atau salah [72]. Dalam *confusion matrix*, ada empat variabel dalam proses klasifikasi, yaitu *Fasle Positive* (FP), *True Positive* (TP), *True Negative* (TN), dan *False Negative* (FN). TP menunjukkan data positif yang diklasifikasikan secara benar, TN menunjukkan data negatif yang diklasifikasikan benar, FP adalah data negatif tetapi diklasifikasikan sebagai data positif, dan FN adalah data data positif tetapi diklasifikasi sebagai data negatif [73].

Tabel 2.1 *Confusion Matrix Table Plotting*

		Predicted	
		Negative	**Positive**
Actual	**Negative**	TN	FP
	Positive	FN	TP

Perhitungan pada *confusion matrix* menghasilkan keluaran seperti *accuracy*, *precission*, dan *recall* [74]. *Accuracy* merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data [75].

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} * 100\% \quad (16)$$

Precision adalah rasio dari data yang diprediksi positif dibandingkan dengan semua prediksi yang memiliki nilai positif [73].

$$Precision = \frac{TP}{TP+FP} * 100\% \quad (17)$$

Recall merupakan rasio prediksi benar positif dengan keseluruhan data yang benar positif [75].

$$Recall = \frac{TP}{TP+FN} * 100\% \quad (18)$$

2.12 *Blackbox Testing*

Pengujian adalah satu set aktivitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi sebuah sistem yang telah dibangun. Pengujian pada sistem dengan *blackbox testing* bertujuan untuk mengetahui kelemahan dari sistem agar sistem yang dihasilkan sesuai dengan sistem yang telah dirancang, sehingga meminimalisir kekurangan dan kesalahan pada aplikasi sebelum digunakan oleh user [76].

Metode *blackbox* merupakan metode pengujian yang memungkinkan pengujian fungsional setiap fitur aplikasi dalam pengujian sebuah perangkat lunak. Metode ini memungkinkan penguji untuk menguji fungsionalitas sebuah sistem melalui berbagai input tertentu untuk mengetahui apakah program yang dibuat telah memenuhi kebutuhan spesifikasi yang telah dirancang [77].

Pada *blackbox testing*, pengujian berfokus pada verifikasi fungsionalitas sistem tanpa memperhatikan struktur kode dan arsitekturnya. Dalam prosesnya, penguji menggunakan berbagai skenario untuk menguji beragam input pengguna guna menemukan kesalahan atau perilaku yang tidak diinginkan [78]. Tahapan dalam melakukan *blackbox testing* meliputi [79]:

1. Membuat *test case* untuk menguji berbagai fungsi dalam aplikasi.
2. Mengembangkan *test case* untuk mengevaluasi seberapa jauh alur kerja suatu fungsi sesuai dengan kebutuhan dan keinginan pengguna.
3. Mengidentifikasi *bug* atau kesalahan berdasarkan antarmuka aplikasi.