PROPOSAL TUGAS AKHIR

IMPLEMENTASI SISTEM REKOMENDASI FILM BERBASIS WEBSITE MENGGUNAKAN PENDEKATAN *COLLABORATIVE FILTERING* DENGAN METODE *WEIGHTED HYBRIDIZATION* (SVD-KNN)

RAFI HARLIANTO

2210651167

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH JEMBER

2025

PROPOSAL TUGAS AKHIR

IMPLEMENTASI SISTEM REKOMENDASI FILM BERBASIS WEBSITE MENGGUNAKAN PENDEKATAN *COLLABORATIVE FILTERING* DENGAN METODE *WEIGHTED HYBRIDIZATION* (SVD-KNN)

Diajukan untuk Memenuhi Persyaratan Guna Meraih Gelar Sarjana Komputer

Teknik Informatika Universitas Muhammadiyah Jember

RAFI HARLIANTO

2210651167

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH JEMBER

2025

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Pesatnya perkembangan teknologi informasi dan internet telah mengubah cara manusia dalam mengakses serta mengonsumsi informasi dan hiburan. Di era digital ini, laju pertumbuhan data dan informasi terus meningkat, terutama dalam bentuk konten media seperti film, musik, dan video (Ricci et al., 2011). Kondisi kelimpahan informasi dan media ini menghadirkan tantangan tersendiri, karena dari ribuan hingga jutaan judul film yang tersedia di berbagai platform digital, pengguna sering mengalami kesulitan dalam menemukan film yang sesuai dengan preferensi mereka (Anwar & Uma, 2021; Hssina et al., 2021).

Sistem Rekomendasi (*Recommendation System* - RS) hadir sebagai solusi untuk mengatasi masalah kelimpahan informasi dengan memprediksi preferensi pengguna dan merekomendasikan *item* yang kemungkinan besar disukai oleh pengguna (Anwar & Uma, 2021; Ricci et al., 2011). Dengan kemampuan ini, sistem rekomendasi membantu meningkatkan pengalaman pengguna dalam menentukan berbagai pilihan yang tersedia secara lebih efisien dan relevan. Berbagai pendekatan telah dikembangkan dalam pembangunan RS, mencakup *content-based filtering* (CBF) yang fokus pada atribut item, *collaborative filtering* (CF) yang menganalisis interaksi antar pengguna dan item, serta *hybrid filtering* (HF) yang menggabungkan kekuatan dari beberapa metode untuk meningkatkan efektivitas rekomendasi secara keseluruhan (Isinkaye et al., 2015; Saifudin & Widiyaningtyas, 2024). Masing-masing pendekatan memiliki kelebihan dan kekurangannya sendiri dalam konteks data dan tujuan rekomendasi yang berbeda.

Salah satu pendekatan yang paling populer dalam sistem rekomendasi adalah *Collaborative Filtering* (CF) (Ahmed & Letta, 2023; Anwar & Uma, 2021; Hssina et al., 2021). CF bekerja berdasarkan ide bahwa pengguna yang memiliki selera atau perilaku serupa di masa lalu akan cenderung memiliki selera yang serupa di masa depan. Metode ini menggunakan data interaksi pengguna (seperti *rating* atau riwayat tontonan) untuk menemukan kesamaan antara pengguna (*user-based* CF) atau antara item (*item-based* CF) (Isinkaye et al., 2015; Saifudin & Widiyaningtyas, 2024). Dua algoritma yang sering digunakan dalam pendekatan CF adalah *K-Nearest Neighbors* (KNN) dan *Singular Value Decomposition* (SVD) (Anwar & Uma, 2021; Hssina et al., 2021). Algoritma KNN mengidentifikasi sejumlah K pengguna atau item terdekat (serupa) untuk membuat prediksi

(Hssina et al., 2021). Dalam KNN, kesamaan antar pengguna atau item diukur menggunakan berbagai metrik, salah satunya adalah *cosine similarity*. Metrik ini mengukur cosinus dari sudut antara dua vektor (merepresentasikan pengguna-item dalam matriks *rating*), di mana nilai cosinus yang mendekati 1 menunjukkan kesamaan yang tinggi (Hssina et al., 2021; Isinkaye et al., 2015). *Cosine similarity* sering digunakan karena efektif dalam membandingkan vektor data yang *sparse*. Sementara itu, SVD adalah teknik dekomposisi matriks yang mereduksi dimensi data rating untuk mengungkap faktor laten yang memengaruhi preferensi pengguna dan karakteristik item (Anwar & Uma, 2021; Hssina et al., 2021).

Penelitian sebelumnya telah banyak berkontribusi dalam pengembangan sistem rekomendasi *collaborative filtering*, termasuk eksplorasi algoritma seperti *K-Nearest Neighbors* (KNN) dan *Singular Value Decomposition* (SVD), serta berbagai teknik hibridisasi. Penelitian yang paling relevan telah menguji kombinasi algoritma inti tersebut dan mengevaluasinya. Hssina et al. (2021), menggabungkan algoritma KNN dan SVD dalam sistem rekomendasi hibrida dan mengevaluasinya menggunakan RMSE dan MAE; hasil mereka menunjukkan bahwa pendekatan hibrida memberikan hasil yang lebih baik dibandingkan menggunakan KNN atau SVD secara individual. Sementara itu, Anwar & Uma (2021), melakukan studi komparatif terhadap berbagai pendekatan *collaborative filtering*, termasuk KNN dan SVD, pada dataset MovieLens 100K dan mengevaluasinya dengan RMSE dan MAE; studi tersebut menunjukkan SVD sebagai algoritma individu yang memiliki kinerja lebih baik (RMSE = 0.919, MAE = 0.716) dibandingkan KNN (RMSE = 0.935, MAE = 0.733). Dalam studi terbaru, Yap et al. (2024), mengimplementasikan sistem rekomendasi makanan hibrida yang juga memanfaatkan pendekatan KNN dan SVD dalam skema *weighted hybrid* dan mengevaluasinya menggunakan MAE dan RMSE yang membuktikan bahwa SVD memiliki akurasi prediksi yang lebih tinggi dibandingkan KNN (RMSE ~1.01 vs ~1.15, MAE ~0.79 vs ~0.91), dan kombinasi pembobotan (KNN-SVD) menunjukkan RMSE= 1.05 dan MAE= 0.83, mengindikasikan kinerja yang baik dengan menggabungkan keduanya. Penelitian-penelitian terdahulu ini sangat relevan dan memberikan dasar yang kuat, berdasarkan analisis, penggabungan komprehensif *item-based* KNN (menggunakan *cosine similarity*) dan SVD dalam pendekatan *weighted hybrid* sudah diterapkan. Namun pada sistem rekomendasi film belum ada penggunaan menggunakan dataset MovieLens-1M dan diimplementasikan dalam sistem berbasis web menggunakan Streamlit. Dengan demikian, penelitian ini bertujuan untuk mengisi celah penelitian tersebut dengan mengeksplorasi kinerja kombinasi metode dengan dataset yang berbeda.

Penelitian ini berfokus pada pengembangan sistem rekomendasi film berbasis website yang mengimplementasikan pendekatan *collaborative filtering* menggunakan algoritma KNN dan SVD dengan teknik penggabungan *weighted hybrid* yang digunakan untuk menggabungkan kedua skor dari masing-masing algoritma. Dataset yang akan digunakan sebagai data interaksi pengguna dan film adalah MovieLens-1M, sebuah dataset standar dan bersifat terbuka untuk publik (*open-source*) dalam riset sistem rekomendasi. Untuk mengukur seberapa baik sistem dan algoritma yang dikembangkan, kinerja sistem akan dievaluasi menggunakan metrik akurasi *Root Mean Square Error* (RMSE) dan *Mean Absolute Error* (MAE), yang merupakan metrik evaluasi umum untuk mengukur kesalahan prediksi rating (Anwar & Uma, 2021; Isinkaye et al., 2015). Antarmuka berbasis website akan dikembangkan secara lokal (*localhost*) menggunakan *framework* Streamlit dengan bahasa pemrograman Python untuk memudahkan pengembangan, visualisasi data, dan pengujian sistem.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan sistem rekomendasi film berbasis website menggunakan pendekatan *collaborative filtering* yang menggabungkan algoritma KNN dan SVD dengan teknik *weighted hybrid*?
2. Bagaimana kinerja sistem rekomendasi *collaborative filtering* menggunakan teknik *weighted hybrid* (KNN - SVD) dalam uji evaluasi RMSE dan MAE?

1.3. Tujuan Penelitian

Adapun tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Merancang dan mengimplementasikan sistem rekomendasi film berbasis website menggunakan pendekatan *collaborative filtering* dengan menggabungkan algoritma KNN dan SVD dengan teknik *weighted hybrid*.
2. Mengevaluasi kinerja sistem rekomendasi *collaborative filtering* dengan teknik *weighted hybrid* (SVD - KNN) menggunakan RMSE dan MAE.

1.4. Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini adalah:

1. **Bagi Peneliti:** Mendapatkan pemahaman mendalam mengenai perancangan dan implementasi sistem rekomendasi menggunakan pendekatan *collaborative filtering*, serta meningkatkan keterampilan dalam pengembangan aplikasi berbasis website dan analisis kinerja algoritma.
2. **Bagi Akademisi/Pengembang Lain:** Dapat menjadi referensi dan dasar bagi penelitian atau pengembangan sistem rekomendasi lanjutan, khususnya yang berkaitan dengan pendekatan *collaborative filtering* dan implementasinya dalam platform website.
3. **Bagi Industri:** Memberikan wawasan mengenai potensi penerapan pendekatan KNN - SVD untuk meningkatkan kualitas rekomendasi pada platform mereka, yang dapat berdampak pada peningkatan kepuasan pengguna.

1.5. Batasan Penelitian

Adapun Batasan penelitian ini adalah:

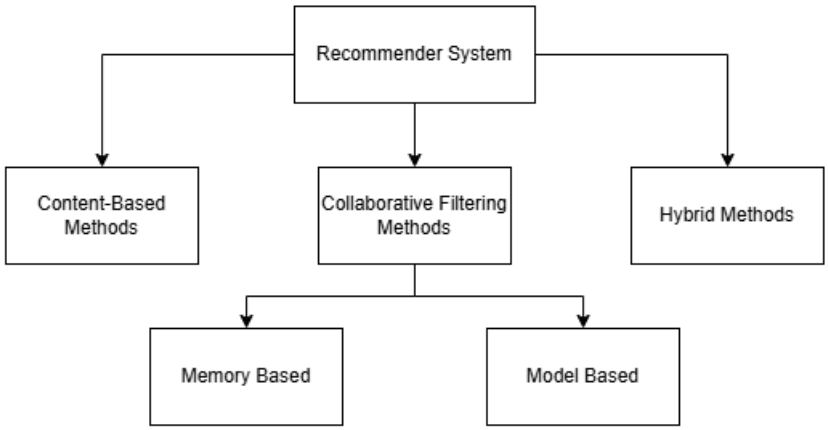
1. Dataset film dan rating yang digunakan adalah data publik siap pakai yang bersifat *open-source* yaitu MovieLens-1M.
2. Implementasi algoritma KNN dan SVD dalam penelitian dilakukan menggunakan modul surprise pada *library* python. Penelitian ini tidak mencakup penjabaran maupun pengembangan dari algoritma KNN atau SVD yang disediakan oleh *library* tersebut.
3. Teknik penggabungan menggunakan *weighted hybrid*, untuk menggabungkan hasil skor prediksi menggunakan bobot yang tertimbang pada kedua algoritma (KNN dan SVD).
4. Evaluasi kinerja sistem rekomendasi akan menggunakan metrik akurasi standar yaitu *Root Mean Square Error* (RMSE) dan *Mean Absolute Error* (MAE).
5. Sistem yang dibangun hanya merupakan aplikasi berbasis website sederhana menggunakan *framework* streamlit untuk demonstrasi implementasi SVD-KNN dengan pembobotan (*weighted hybrid*) dengan dataset movielens 1M, tidak mencakup fitur seperti manajemen pengguna dan *real-time recommendations*.
6. Rekomendasi hanya akan diberikan untuk film, item lain di luar film tidak termasuk dalam cakupan penelitian.

BAB 2
KAJIAN PUSTAKA

2.1 Sistem Rekomendasi

Sistem Rekomendasi (*Recommendation System* - RS) adalah sebuah sistem informasi yang dirancang untuk memprediksi preferensi pengguna terhadap suatu item dan merekomendasikan item yang kemungkinan besar akan disukai oleh pengguna tersebut (Isinkaye et al., 2015). Tujuan utama RS adalah membantu pengguna mengatasi masalah *information overload* dengan menyaring item yang paling relevan dari sekumpulan besar pilihan (Ahmed & Letta, 2023). Sehingga RS telah banyak diimplementasikan di berbagai domain, termasuk *e-commerce*, musik, berita, dan film, untuk meningkatkan pengalaman pengguna dan mendorong interaksi.

Berdasarkan teknik yang digunakan, sistem rekomendasi dapat diklasifikasikan ke dalam beberapa tipe utama. Secara umum, tipe dan klasifikasi sistem rekomendasi dijelaskan pada gambar 2.1.



Gambar 2. 1. Klasifikasi Sistem Rekomendasi

- *Content-Based Filtering* (CBF)

  Merekomendasikan item yang memiliki atribut (fitur) serupa dengan item yang disukai pengguna di masa lalu. CBF menganalisis konten deskriptif dari item tersebut (Isinkaye et al., 2015).

- *Collaborative Filtering* (CF)

  Merekomendasikan item berdasarkan perilaku atau preferensi pengguna lain yang memiliki selera serupa. CF tidak memerlukan analisis konten item secara mendalam (Anwar & Uma, 2021; Hssina et al., 2021).

- *Hybrid Methods*

  Menggabungkan dua atau lebih teknik rekomendasi (misalnya, menggabungkan CBF dan CF, atau menggabungkan dua teknik CF) untuk memanfaatkan kelebihan masing-masing dan mengatasi kelemahannya (Isinkaye et al., 2015).

2.2 *Collaborative Filtering* (CF)

Collaborative filtering (CF) adalah teknik rekomendasi yang paling umum dalam riset dan aplikasi praktis (Ahmed & Letta, 2023; Saifudin & Widiyaningtyas, 2024). CF bekerja dengan menganalisis data interaksi pengguna terhadap item, seperti rating, riwayat pembelian, atau *views*, untuk mengidentifikasi pola dan kesamaan (Hssina et al., 2021). Sehingga dari kesamaan tersebut, pendekatan CF dapat merekomendasikan *item* secara personal kepada pengguna. CF secara umum dibagi menjadi dua pendekatan utama:

- *Memory-Based* (atau *Neighborhood-Based*) CF

  Metode ini beroperasi langsung pada seluruh dataset pengguna-item untuk menghitung kesamaan antara pengguna (*user-based* CF) atau item (*item-based* CF). Prediksi *rating* atau rekomendasi kemudian dihasilkan berdasarkan preferensi dari sejumlah tetangga terdekat yang ditemukan. Algoritma *K-Nearest Neighbors* (KNN) merupakan contoh utama dari pendekatan *memory-based* CF (Delimayanti et al., 2022).

- *Model-Based* CF

  Metode ini terlebih dahulu membangun (melatih) sebuah model berdasarkan dataset pengguna-item. Model ini berusaha menangkap struktur atau pola tersembunyi dalam data, seperti faktor laten yang memengaruhi preferensi pengguna dan karakteristik item. Setelah model dibangun, prediksi rating atau rekomendasi dapat dihasilkan dengan lebih cepat. Singular Value Decomposition (SVD) dan teknik *matrix factorization* lainnya termasuk dalam kategori *model-based* CF (Bhowmick et al., 2021).

Kedua pendekatan ini memiliki kelebihan dan kekurangan, m*emory-based* CF relatif mudah diimplementasikan dan mudah diinterpretasikan, namun kurang skalabel pada dataset besar dan rentan terhadap masalah *sparsity*. *Model-based* CF cenderung lebih skalabel dan efektif dalam menangani *sparsity* setelah model terbentuk, namun modelnya sulit untuk diperbarui secara *real-time* (Anwar & Uma, 2021; Saifudin & Widiyaningtyas, 2024). Dalam penelitian ini, akan dibahas lebih lanjut kedua algoritma yang mewakili kedua pendekatan ini, yaitu SVD (*model-based*) dan KNN (*memory-based*).

2.2.1 *Singular Value Decomposition* (SVD)

Singular Value Decomposition (SVD) merupakan suatu teknik yang digunakan sebagai pendekatan dekomposisi matriks untuk meramalkan elemen yang tidak memiliki peringkat (Saifudin & Widiyaningtyas, 2024). Dengan dekomposisi matriks, matriks utama yang berdimensi lebih besar akan dibagi menjadi matriks berdimensi lebih kecil. SVD efektif dalam mengatasi masalah *sparsity* karena ia menginferensi nilai-nilai yang hilang dari pola faktor laten. Namun, SVD memiliki keterbatasan dalam menangani *cold start* dan kesulitan memasukkan informasi kontekstual tambahan.

SVD dalam sistem rekomendasi merupakan modifikasi dari SVD murni yang bekerja secara langsung pada nilai rating yang diketahui tanpa perlu mengisi nilai yang hilang dan secara eksplisit memfaktorkan matriks rating menjadi dua matriks faktor laten pengguna (p_u) dan item (q_i) berdimensi lebih rendah untuk mengatasi kendala dalam menangani matriks rating yang besar dan jarang (Tiara, 2024). Prediksi rating ($\hat{r}_{u,i}$) dihitung sebagai *dot product* ($p_u \cdot q_i^T$), *p* dan *q* ditemukan melalui proses pembelajaran mesin menggunakan optimasi *Stochastic Gradient Descent* (SGD) untuk meminimalkan fungsi kerugian dan digunakan dengan regularisasi guna mencegah overfitting (Zhang, 2022). SVD yang menggunakan optimasi via SGD ini diperkenalkan pertama kali oleh Simon Funk dalam kompetisi Netflix Prize pada tahun 2006.

Dalam penelitian ini, SVD berfungsi untuk memprediksi *rating* oleh pengguna *u* terhadap item *i*. Perhitungan prediksi *rating* oleh pengguna *u* terhadap item *i* dapat digunakan rumus yang ditentukan dalam persamaan (1.1) (Saifudin & Widiyaningtyas, 2024):

$$\hat{\boldsymbol{r}}_{\boldsymbol{u,i}} = \boldsymbol{\mu} + \boldsymbol{b_u} + \boldsymbol{b_i} + \boldsymbol{p_u q_i^T} \tag{1.1}$$

Keterangan:

$\hat{\boldsymbol{r}}_{\boldsymbol{u,i}}$ = prediksi *rating* pengguna *u* terhadap item *i*

$\boldsymbol{\mu}$ = rata-rata *rating* dari semua item (*global mean*)

$\boldsymbol{b_u}$ = bias pengguna *u*, deviasi rata-rata rating pengguna *u* dari μ

$\boldsymbol{b_i}$ = bias item *i*, deviasi rata-rata *rating* item *i* dari μ

$\boldsymbol{p_u}$ = vektor faktor laten pengguna, yaitu representasi numerik dari preferensi pengguna dalam ruang dimensi *k*

$\boldsymbol{q_i^T}$ = vektor faktor laten item yang ditranspos, yaitu representasi numerik dari preferensi item yang ditranspos dalam ruang dimensi *k*

Rata-rata rating dari semua item (*global mean*) dapat dihitung dengan persamaan (1.2):

$$\boldsymbol{\mu} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum \boldsymbol{r_{u,i}} \tag{1.2}$$

Keterangan:

$\boldsymbol{\mu}$ = rata-rata seluruh *rating* (*global mean*)

$\boldsymbol{r_{u,i}}$ = *rating* asli pengguna *u* terhadap item *i*

$\boldsymbol{N}$ = total *rating*

Pada perhitungan skor prediksi menggunakan SVD digunakan nilai *bias* pengguna *u* dan *bias* item *i*, nilai-nilai ini digunakan untuk mengurangi kesalahan prediksi dari pengguna maupun item (Saifudin & Widiyaningtyas, 2024). Nilai *bias* pengguna *u* dan nilai *bias* item *i* dapat dicari menggunakan persamaan (1.3) dan (1.4):

Bias pengguna *u* (b_u):

$$\boldsymbol{b_u} = \frac{\mathbf{1}}{|\boldsymbol{I_u}|} \sum_{\boldsymbol{u \in I_u}} (\boldsymbol{r_{u,i}} - \boldsymbol{\mu}) \tag{1.3}$$

Bias item *i* (b_i):

$$\boldsymbol{b_i} = \frac{\mathbf{1}}{|\boldsymbol{U_i}|} \sum_{\boldsymbol{u \in U_i}} (\boldsymbol{r_{u,i}} - \boldsymbol{\mu}) \tag{1.4}$$

Keterangan:

$\boldsymbol{I_u}$ = total item *i* yang dinilai pengguna *u*

$\boldsymbol{U_i}$ = total pengguna *u* yang memberi rating pada item *i*

SVD dalam sistem rekomendasi merupakan *model-based collaborative filtering* sehingga penerapan algoritma memerlukan pelatihan parameter menggunakan metode *stochastic gradient descent* (SGD). Proses pelatihan ini bertujuan untuk meminimalkan kesalahan antara rating aktual dan rating prediksi melalui iterasi terhadap setiap pasangan user-item yang memiliki nilai rating (Zhang, 2022). Menurut R. Akbar et al. (2023), pemilihan nilai parameter meliputi jumlah epoch (*n_epochs*), nilai learning rate (γ), nilai regularisasi (λ), dan jumlah faktor laten (*k* / *n_factors*) sangat menentukan akurasi hasil prediksi.

SVD untuk sistem rekomendasi menghitung faktor laten pengguna (p_u) dan faktor laten item (q_i) via SGD dengan meng-*update* kedua faktor laten dengan proses berikut (Tiara, 2024):

- Inisialisasi vektor faktor laten pengguna (p_u) dan item (q_i) dengan nilai acak kecil (antara -0.1 hingga 0.1).
- Menghitung *error* prediksi pada persamaan (1.5):

$$\boldsymbol{e_{u,i} = r_{u,i} - \hat{r}_{u,i}} \tag{1.5}$$

- *Update* parameter p_u, q_i dengan persamaan (1.6), (1.7) (Hug, 2020):

K
$$\boldsymbol{p_u \leftarrow p_u + \gamma(e_{u,i} \cdot q_i - \lambda p_u)} \tag{1.6}$$
e
$$\boldsymbol{q_i \leftarrow q_i + \gamma(e_{u,i} \cdot p_u - \lambda q_i)} \tag{1.7}$$

Keterangan:

$\boldsymbol{e_{u,i}}$ = *error* prediksi (selisih rating asli dengan rating prediksi)

$\boldsymbol{\gamma}$ = *learning rate*, (semakin besar, semakin cepat pembaruan parameter)

$\boldsymbol{\lambda}$ = *regularization term* (parameter regularisasi)

- Setelah jumlah iterasi (*n_epochs*) tercapai maka digunakan *final value* dari *update* parameter digunakan untuk memprediksi rating yang kosong.

2.2.2 *K-Nearest Neighbors* (KNN)

Algoritma *K-Nearest Neighbors* (KNN) adalah algoritma yang termasuk dalam kategori *memory-based collaborative filtering*. Dalam CF, KNN dapat diimplementasikan baik sebagai *user-based* maupun *item-based* (Isinkaye et al., 2015). Ide dasarnya adalah menemukan K entitas (pengguna atau item) dalam dataset yang paling mirip dengan entitas target berdasarkan metrik kesamaan tertentu, lalu menggunakan preferensi dari K tetangga terdekat ini untuk membuat prediksi (Hssina et al., 2021). Salah satu metrik kesamaan yang paling umum dan efektif digunakan bersama dengan KNN dalam *collaborative filtering* adalah *cosine similarity* (Airen & Agrawal, 2022).

Cosine Similarity adalah cara untuk mengukur seberapa mirip dua teks dengan melihat sudut antara dua vektor yang mewakili teks tersebut, bukan seberapa jauh jaraknya. Menurut Airen & Agrawal (2022), semakin kecil sudut antara dua vektor, semakin besar nilai cosinusnya (mendekati 1), yang menunjukkan tingkat kesamaan yang lebih tinggi. *Cosine Similarity* sangat berguna dalam konteks data rating yang *sparse* karena fokus pada orientasi vektor, bukan besarannya (Airen & Agrawal, 2022).

Cosine similarity mengukur kesamaan berbasis vektor dengan rentang 0 (kesamaan minimum) hingga 1 (kesamaan maksimum). Kesamaan kosinus antara item

i dan *j* didefinisikan pada persamaan (2.1) (Isinkaye et al., 2015):

$$\mathbf{cosine}(\boldsymbol{i},\boldsymbol{j}) = \frac{\sum_{\boldsymbol{u}\in\boldsymbol{U}} \boldsymbol{r}_{\boldsymbol{u},\boldsymbol{i}} \cdot \boldsymbol{r}_{\boldsymbol{u},\boldsymbol{j}}}{\sqrt{\sum_{\boldsymbol{u}\in\boldsymbol{U}} \boldsymbol{r}_{\boldsymbol{u},\boldsymbol{i}}^{2}} \cdot \sqrt{\sum_{\boldsymbol{u}\in\boldsymbol{U}} \boldsymbol{r}_{\boldsymbol{u},\boldsymbol{j}}^{2}}} \tag{2.1}$$

Keterangan:

$\boldsymbol{cosine}(\boldsymbol{i},\boldsymbol{j})$ = Kemiripan cosinus antara item *i* dengan item *j*

$\boldsymbol{r}_{\boldsymbol{u},\boldsymbol{i}}$ = rating asli pengguna *u* terhadap item *i*

$\boldsymbol{r}_{\boldsymbol{u},\boldsymbol{j}}$ = rating asli pengguna *u* terhadap item *j*

$\boldsymbol{U}$ = himpunan pengguna *u* yang telah menilai item *i* dan *j*

Setelah kesamaan antara item target *i* dan item-item lain dihitung, algoritma *Item-Based* KNN akan mengidentifikasi K item yang paling mirip dengan item *i* yang telah dinilai oleh pengguna *u*. Prediksi rating $\hat{r}_{\{u,i\}}$ untuk item *i* oleh pengguna *u* kemudian dihitung berdasarkan rating yang telah diberikan oleh pengguna *u* pada K item tetangga terdekat tersebut (Isinkaye et al., 2015). Salah satu persamaan prediksi yang umum digunakan dalam *Item-Based* KNN pada persamaan (2.2):

$$\hat{\boldsymbol{r}}_{\boldsymbol{u},\boldsymbol{i}} = \frac{\sum_{\{\boldsymbol{j}\in\boldsymbol{N}(\boldsymbol{i};\boldsymbol{u})\}} \boldsymbol{sim}(\boldsymbol{i},\boldsymbol{j}) \cdot \boldsymbol{r}_{\boldsymbol{u},\boldsymbol{j}}}{\sum_{\{\boldsymbol{j}\in\boldsymbol{N}(\boldsymbol{i};\boldsymbol{u})\}} |\boldsymbol{sim}(\boldsymbol{i},\boldsymbol{j})|} \tag{2.2}$$

Keterangan:

$\hat{\boldsymbol{r}}_{\boldsymbol{u},\boldsymbol{i}}$ = prediksi rating untuk pengguna *u* pada item *i*

$\boldsymbol{N}(\boldsymbol{i};\boldsymbol{u})$ = himpunan K item tetangga terdekat dari item *i* yang telah dinilai oleh pengguna *u*

$\boldsymbol{sim}(\boldsymbol{i},\boldsymbol{j})$ = nilai kesamaan (*similarity*) antara item *i* dan item *j*.

$\boldsymbol{r}_{\boldsymbol{u},\boldsymbol{j}}$ = rating sebenarnya yang diberikan oleh pengguna *u* pada item tetangga *j*

2.3 Weighted Hybrid

Weighted hybrid merupakan salah satu metode penggabungan (*hybridization*) yang menghitung skor prediksi sebagai hasil dari semua pendekatan rekomendasi dengan mempertimbangkannya sebagai variabel dalam kombinasi linier sehingga teknik ini memberikan bobot pada masing-masing pendekatan dan menjumlahkan hasil tertimbang (Suriati et al., 2017).

Menurut Y. A. Akbar et al. (2021), pendekatan *weighted hybrid* memungkinkan sistem rekomendasi untuk memanfaatkan keunggulan dari masing-masing metode yang

digunakan. Desain yang sesuai untuk pendekatan ini adalah hibridisasi paralel, di mana setiap metode dijalankan secara terpisah.

Pada penelitian ini digunakan 2 (dua) pendekatan rekomendasi (*hybrid*) yang digunakan, sehingga *c* (jumlah pendekatan) ditetapkan sama dengan 2 (dua), perhitungan skor prediksi dapat ditulis pada persamaan (3):

$$\hat{\boldsymbol{p}}_{\boldsymbol{u,i}} = \boldsymbol{\alpha} \,.\, \hat{\boldsymbol{r}}^{(\boldsymbol{1})}_{\boldsymbol{u,i}} + \boldsymbol{\beta} \,.\, \hat{\boldsymbol{r}}^{(\boldsymbol{2})}_{\boldsymbol{u,i}} \tag{3}$$

$$\boldsymbol{\alpha} + \boldsymbol{\beta} = \mathbf{1}$$

Keterangan:

$\hat{\boldsymbol{p}}_{\boldsymbol{u,i}}$ = skor prediksi pembobotan untuk user *u* terhadap item *i*

$\boldsymbol{\alpha}$ = bobot (*weight*) untuk metode pertama

$\boldsymbol{\beta}$ = bobot (*weight*) untuk metode kedua

$\hat{\boldsymbol{r}}^{(\boldsymbol{1})}_{\boldsymbol{u,i}}$ = skor prediksi dari metode pertama (dalam penelitian ini SVD)

$\hat{\boldsymbol{r}}^{(\boldsymbol{2})}_{\boldsymbol{u,i}}$ = skor prediksi dari metode kedua (dalam penelitian ini KNN)

Setelah perhitungan pembobotan maka didapat skor prediksi hasil pembobotan yang dapat menentukan rekomendasi *n* teratas (*top-n recommendation*) item kepada pengguna tertentu dari hasil pembobotan.

2.4 *Mean Absolute Error* (MAE)

Mean Absolute Error (MAE) adalah salah satu metrik evaluasi yang umum digunakan dalam sistem rekomendasi, khususnya untuk mengukur akurasi model yang memprediksi rating numerik (Anwar & Uma, 2021; Saifudin & Widiyaningtyas, 2024). Dalam jurnal yang ditulis oleh Y. A. Akbar et al. (2021), MAE digunakan untuk membandingkan nilai prediksi dengan nilai aktual. Semakin rendah nilai MAE, semakin tinggi nilai akurasi rekomendasi yang dihasilkan. Berikut ini adalah rumus MAE pada persamaan (4):

$$\boldsymbol{MAE} = \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\boldsymbol{x_i} - \boldsymbol{y_i}|}{\boldsymbol{n}} \tag{4}$$

Keterangan:

$\boldsymbol{MAE}$ = *Mean Absolute Error*

$\boldsymbol{x_i}$ = peringkat pengguna yang diprediksi pada item i

$\boldsymbol{y_i}$ = peringkat aktual pengguna pada item i

$\boldsymbol{n}$ = jumlah total pasangan peringkat x_i dan y_i

Nilai MAE berkisar antara 0 hingga skala rating maksimum. Nilai MAE yang lebih rendah menunjukkan akurasi prediksi yang lebih tinggi atau, dengan kata lain, kesalahan prediksi rata-rata yang lebih kecil (Isinkaye et al., 2015).

2.5 *Root Mean Square Error* (RMSE)

Root Mean Square Error (RMSE) adalah metrik evaluasi akurasi prediksi rating lainnya yang sangat umum digunakan dalam penelitian sistem rekomendasi (Anwar & Uma, 2021; Saifudin & Widiyaningtyas, 2024). RMSE mengukur akar kuadrat dari rata-rata kuadrat perbedaan antara rating yang diprediksi dan rating sebenarnya. Root mean square error (RMSE) lebih menekankan pada kesalahan absolut yang lebih besar dan semakin rendah RMSE, semakin baik akurasi rekomendasinya (Hssina et al., 2021).

RMSE banyak digunakan untuk mengevaluasi kinerja model sistem rekomendasi, yang didefinisikan sebagai rumus (Y. A. Akbar et al., 2021):

$$\boldsymbol{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\boldsymbol{x_i} - \boldsymbol{y_i})^2}{\boldsymbol{n}}} \tag{5}$$

Keterangan:

$\boldsymbol{RMSE}$ = *Root Mean Square Error*

$\boldsymbol{x_i}$ = peringkat pengguna yang diprediksi pada item i

$\boldsymbol{y_i}$ = peringkat aktual pengguna pada item i

$\boldsymbol{n}$ = jumlah total pasangan peringkat x_i dan y_i

Nilai RMSE yang lebih rendah menunjukkan akurasi prediksi yang lebih tinggi. Namun, karena RMSE mengkuadratkan perbedaan *error* sebelum menghitung rata-rata dan akar kuadratnya, RMSE memberikan bobot yang lebih besar pada kesalahan prediksi yang lebih besar (*outliers*) dibandingkan dengan MAE (Isinkaye et al., 2015). RMSE sering dianggap sebagai metrik yang lebih baik untuk menunjukkan seberapa buruk prediksi sistem.

2.6 Penelitian Terdahulu

Beberapa penelitian sebelumnya telah membahas sistem rekomendasi menggunakan *collaborative filtering*, termasuk algoritma KNN dan SVD, serta pendekatan *weighted hybrid*. Tabel 2.1 berikut merangkum beberapa penelitian relevan dan membandingkannya dengan penelitian yang diusulkan:

Tabel 2. 1. Penelitian Terdahulu

Penulis	**Judul**	**Hasil**	**Perbedaan**
(Suriati et al., 2017)	*Weighted hybrid technique for recommender system*	Mengembangkan teknik weighted hybrid untuk sistem rekomendasi. Menguji berbagai metode pembobotan untuk menggabungkan hasil dari algoritma yang berbeda.	Fokus pada teknik *weighted hybrid* secara umum; tidak secara spesifik mengimplementasikan kombinasi KNN dan SVD.
(Do et al., 2020)	*Dynamic Weighted Hybrid Recommender Systems*	Mengusulkan metode weighted hybrid dinamis untuk menggabungkan CF dan CBF. Menguji kinerja hybrid dengan metrik standar.	Menggabungkan CF dan CBF menggunakan bobot dinamis, bukan menggabungkan dua metode CF (KNN dan SVD).
(Hssina et al., 2021)	*Recommendation system using the k-nearest neighbors and singular value decomposition algorithms*	Menerapkan sistem rekomendasi hibrida yang menggabungkan KNN dan SVD untuk mengatasi masalah *cold start*. Uji evaluasi menggunakan RMSE dan MAE.	Melakukan penggabungan KNN+SVD dengan metrik evaluasi namun objek penelitian adalah buku.
(Anwar & Uma, 2021)	*Comparative study of recommender system methodes and movie recommendation using collaborative filtering*	Studi komparatif berbagai pendekatan RS, termasuk KNN, SVD, dan SVD++ pada dataset MovieLens 100K. Mengevaluasi kinerja menggunakan RMSE dan MAE.	Menguji KNN dan SVD secara individual (dan SVD++), bukan dalam pendekatan penggabungan algoritma. Menggunakan MovieLens, tetapi versi 100K.

Penulis	**Judul**	**Hasil**	**Perbedaan**
(Ahmed & Letta, 2023)	*Book Recommendation Using Collaborative Filtering Algorithm*	Menggunakan Collaborative Filtering untuk rekomendasi buku. Mengkomparasi antara SVD dan KNN dengan hasil SVD lebih baik dalam mengatasi *sparsity.*	Membandingkan kedua algoritma KNN dan SVD
(Yap et al., 2024)	*Hybrid-based food recommender system utilizing KNN and SVD methods*	Mengembangkan sistem rekomendasi makanan hibrida menggunakan algoritma KNN dan SVD. Mengevaluasi kinerja hybrid.	Mirip dalam penggabungan KNN+SVD dan metrik evaluasi. Perbedaan terletak pada domain data dan detail implementasi website menggunakan Streamlit.

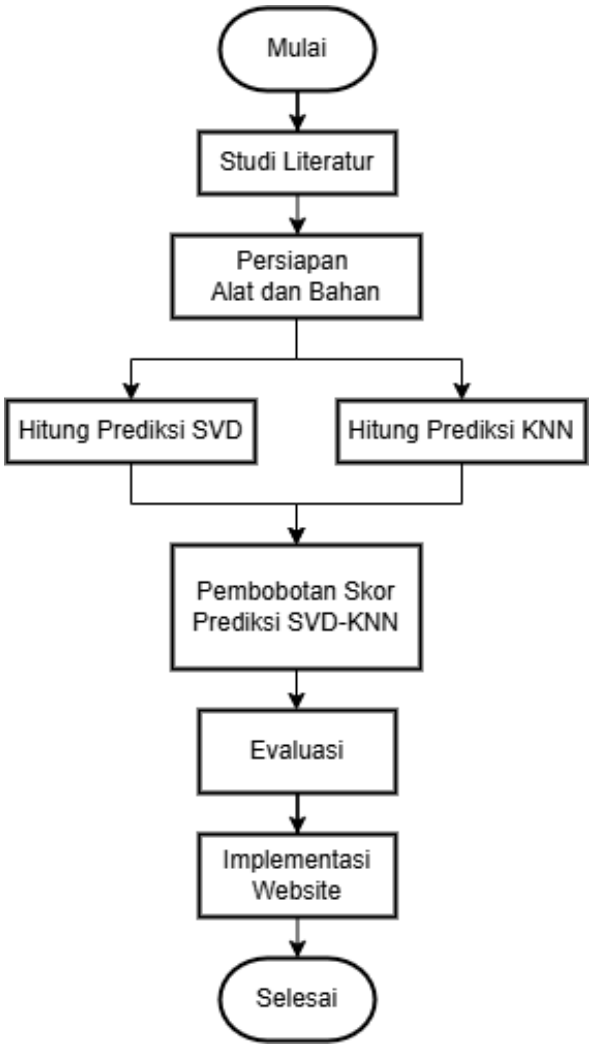
Penelitian ini membangun di atas dasar yang diletakkan oleh studi-studi sebelumnya, khususnya yang mengeksplorasi penggabungan KNN dan SVD dan evaluasi menggunakan RMSE dan MAE pada dataset MovieLens. Kontribusi spesifik penelitian ini adalah implementasi *collaborative filtering* berbasis KNN dan SVD dengan penggabungan *weighted hybrid* pada dataset MovieLens-1M dan pengembangannya dalam platform website menggunakan Streamlit, serta analisis komparatif terhadap kinerja masing-masing komponen dan hasil penggabungan.

BAB 3
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Metode penelitian ini merupakan metode kuantitatif terapan karena berdasarkan jenis data dan implementasi nya dalam pengembangan sistem (Hardani et al., 2020). Berdasarkan pendekatan, jenis penelitian ini termasuk dalam penelitian evaluasi formatif karena dalam penerapannya penelitian ini berfungsi untuk mengevaluasi suatu sistem dengan standar dan program yang telah ditentukan (Abubakar, 2021).

Dalam proses pengembangan sistem rekomendasi film berbasis website pada penelitian ini dilakukan beberapa tahapan dan proses, seperti studi literatur, persiapan alat dan bahan, implementasi algoritma SVD-KNN, pembobotan skor prediksi (SVD-KNN) rekomendasi, dan evaluasi. Gambar 3.1 menunjukkan alur penelitian:



Gambar 3. 1. Alur Penelitian

3.2 Studi Literatur

Pada tahapan ini, studi literatur bertujuan untuk memperoleh pemahaman yang mendalam mengenai konsep dasar, teori, algoritma, dan teknik yang relevan dengan penelitian ini. Studi ini mencakup penelusuran dan analisis terhadap jurnal ilmiah, hasil konferensi, buku, dan sumber terpercaya seperti situs resmi, dokumentasi resmi, dan hal lainnya yang terkait. Hasil studi literatur digunakan sebagai landasan teoritis, panduan dalam perancangan dan implementasi sistem, serta dasar untuk pembanding hasil penelitian ini dengan penelitian sebelumnya (Abubakar, 2021).

3.3 Alat dan Bahan

Pada tahapan ini disiapkan alat dan bahan untuk melakukan penelitian, diantaranya seperti dataset sebagai objek penelitian, perangkat lunak (*software*), dan perangkat keras (*hardware*) untuk menunjang kebutuhan penelitian ini.

3.3.1 *Hardware*

Pada penelitian ini disiapkan perangkat keras sebagai berikut:

- Laptop : Lenovo Thinkpad T470S
- Prosesor (CPU) : Intel(R) Core i7-6600U @ 2.60GHz
- Memori (RAM) : 20GB, *usable at* 19.9 GB
- Penyimpanan : *Solid State Drive* (SSD) NVMe SATA 512 GB
- Internet (Wi-Fi)

3.3.2 *Software*

Perangkat lunak atau *software* yang digunakan untuk penelitian adalah sebagai berikut:

- OS Windows 11 Pro versi 23H2
- Jupyter Notebook
- Python (*library*: surprise, pickle, pandas, numpy, matplotlib, etc)
- Streamlit
- Visual Studio Code (VSCode)
- Microsoft Office (excel, word, etc) 2019

3.3.3 Dataset

Data yang dipakai pada penelitian ini merupakan data sekunder yang didapat dari situs resmi movielens. Data yang digunakan pada penelitian ini, yaitu adalah MovieLens 1M Dataset (https://grouplens.org/datasets/movielens/1m). Dataset ini berisi file *ratings.dat, movies.dat,* dan *users.dat* (Harper & Konstan, 2015).

Dalam penelitian ini, hanya dua file yang digunakan dari dataset MovieLens, yaitu *ratings.dat* dan *movies.dat*. File *ratings.dat* mencakup 1.000.209 data rating yang diberikan oleh 6040 pengguna secara anonim terhadap 3952 film pada tahun 2000, dengan atribut *UserID*, *MovieID*, *Rating*, dan *Timestamp*. Namun, penelitian ini hanya memanfaatkan tiga atribut, yaitu *UserID*, *MovieID,* dan *Rating*. Sementara itu, file *movies.dat* berisi informasi mengenai 3952 judul film dengan atribut *MovieID*, *Title*, dan *Genre.* Dari file ini, hanya atribut *MovieID* dan *Title* yang digunakan, yang berfungsi untuk menampilkan judul film pada hasil akhir rekomendasi berdasarkan skor prediksi. Sehingga dataset terlihat seperti pada tabel 3.1.

Tabel 3. 1. Data *ratings.dat*

No	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
…	…	…	…	…
1000206	6040	562	5	956704746
1000207	6040	1096	4	956715648
1000208	6040	1097	4	956715569

3.4 Implementasi Algoritma

Pada tahapan ini dilakukan implementasi algoritma SVD dan KNN pada dataset MovieLens-1M. Untuk keperluan pelatihan dan pengujian (evaluasi), data dibagi menggunakan pendekatan *hold-out validation* dengan rasio 80% data pelatihan dan 20% data pengujian. Pendekatan ini mengacu pada praktik umum yang telah digunakan luas

dalam evaluasi sistem rekomendasi berbasis collaborative filtering, sebagaimana disebutkan oleh Bauer et al. (2024) bahwa selain memberikan efisiensi dan kemudahan implementasi dalam evaluasi, rasio ini juga sesuai untuk konteks sistem rekomendasi film berbasis preferensi data eksplisit tanpa mempertimbangkan urutan waktu (data statis) (Bauer et al., 2024).

Namun, untuk contoh perhitungan pada tahap ini dataset akan dibagi 70% contoh data latih pada tabel 3.2 dan 30% contoh data uji pada tabel 3.11 untuk penyerdehanaan perhitungan dan kemudahan penjelasan alur implementasi. Data uji memiliki *rating* kosong dengan asumsi pengguna belum melakukan *rating* dan data latih berisi nilai *rating* pengguna yang sebenarnya. Contoh implementasi perhitungan masing-masing algoritma akan memakai data contoh pada tabel 3.2 sebagai berikut:

Tabel 3. 2. Contoh Data Uji *Rating* User-Item

User/Item	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	5	3	4	4	-	2	-	1	5	-
u_2	3	2	4	-	3	5	4	1	-	-
u_3	4	5	-	3	2	1	2	4	-	-
u_4	-	-	3	5	4	-	1	2	5	4
u_5	2	1	2	3	4	3	5	-	-	-

Keterangan:

- $u_1 - u_5$ = merepresentasikan pengguna
- $i_1 - i_{10}$ = merepresentasikan item (film)
- Kolom berisi tanda (-) diasumsikan pengguna belum melakukan *rating* pada film
- Nilai dalam tabel adalah rentang *rating* (0-5) yang diberikan pengguna (*u*) pada film (*i*)

3.4.1 SVD

Pada tahap ini dilakukan implementasi algoritma SVD untuk sistem rekomendasi yang dapat memprediksi *rating user-item* dalam matriks yang jarang (*sparse*). Untuk dapat menhitung skor prediksi dengan SVD dengan persamaan (1.1) maka diperlukan nilai-nilai lain yang harus dicari sebagai berikut:

Hitung μ (*global mean)* dengan persamaan (1.2):

$$\boldsymbol{\mu} = \frac{24 + 22 + 21 + 24 + 20}{35} = \frac{111}{35} \approx 3.171$$

Hitung *bias* pengguna *u* (b_u) menggunakan persamaan (1.3):

Contoh perhitungan *bias* pengguna pada u_1:

$$\boldsymbol{b_{u_1}} = \frac{1.83 - 0.17 + 0.83 + 0.83 - 1.17 - 2.17 + 1.83}{7} = \frac{1.81}{7} \approx 0.257$$

dan seterusnya untuk *bias* pengguna $u_2 - u_5$:

$\boldsymbol{b_{u_2}} = -0.029$ $\boldsymbol{b_{u_4}} = 0.257$

$\boldsymbol{b_{u_3}} = -0.171$ $\boldsymbol{b_{u_5}} = -0.258$

Hitung *bias* item *i* (b_i) menggunakan persamaan (1.4):

Contoh perhitungan *bias* item *i* pada i_1:

$$\boldsymbol{b_{i_1}} = \frac{1.83 - 0.17 + 0.83 - 1.17}{4} \approx 0.329$$

dan seterusnya untuk *bias* item $i_2 - i_{10}$:

$\boldsymbol{b_{i_2}} = -0.171$ $\boldsymbol{b_{i_7}} = 0.079$

$\boldsymbol{b_{i_3}} = -0.171$ $\boldsymbol{b_{i_8}} = -0.671$

$\boldsymbol{b_{i_4}} = 0.079$ $\boldsymbol{b_{i_9}} = 1.829$

$\boldsymbol{b_{i_5}} = -0.171$ $\boldsymbol{b_{i_{10}}} = 0.829$

$\boldsymbol{b_{i_6}} = -0.171$

Perhitungan vektor faktor laten (p_u dan q_i^T):

Pada perhitungan vektor laten perlu ditetapkan *learning rate* dan *regularization term* sesuai dengan standar *default* surprise (Hug, 2020).

Pada contoh perhitungan ini ditetapkan parameter sebagai berikut:

- Learning rate ; $\boldsymbol{\gamma} = 0.005$
- *Regularization term* *;* $\boldsymbol{\lambda} = 0.02$
- Jumlah dimensi faktor laten ; $\boldsymbol{k} = 2$
- Jumlah *update* iterasi ; **epochs** = 1
- Inisialisasi awal faktor laten pengguna; $\boldsymbol{p_u} = [0.1, 0.2]$
- Inisialisasi awal laten item ; $\boldsymbol{q_i} = [0.1, 0.2]$

Perhitungan vektor faktor laten (p_u):

Contoh perhitungan faktor laten (p_u) pengguna u_1 dengan interaksi terhadap item i_1:

Hitung *rating* prediksi (u_1,i_1) dengan persamaan (1.1):

$$\hat{\boldsymbol{r}}_{\boldsymbol{u_1,i_1}} = 3.171 + 0.258 + 0.329 + (0.01 \cdot 0.01) + (0.02 \cdot 0.02)$$

$\hat{r}_{u_1,i_1} = 3.808$

Hitung *error* prediksi dengan persamaan (1.5):

$\hat{e}_{u_1,i_1} = 5 - 3.808 = 1.193$

Update faktor laten pengguna p_{u_1} (k=2) dengan persamaan (1.6):

Dimensi 1: $p_{u_1}^{(1)} = 0.1 + 0.005 \cdot (1.192 \cdot 0.1 - 0.02 \cdot 0.1) = 0.1006$

Dimensi 2: $p_{u_1}^{(2)} = 0.2 + 0.005 \cdot (1.192 \cdot 0.2 - 0.02 \cdot 0.2) = 0.2012$

Berikut *update* faktor laten u_1 pengguna untuk setiap item ($i_1 - i_{10}$) pada tabel 3.3:

Tabel 3. 3. *Update* Faktor Laten Pengguna (p_{u_1})

Item	**Rating**	q_i	$\hat{r}_{u,i}$	$\hat{e}_{u,i}$	$p_{u_1}^{(1)}$	$p_{u_1}^{(2)}$
i_1	5	[0.1, 0.2]	3.807	1.193	0.1006	0.2012
i_2	3	[0.1, 0.2]	3.057	-0.057	0.1005	0.2011
i_3	4	[0.1, 0.2]	3.557	0.443	0.1008	0.2015
i_4	4	[0.1, 0.2]	4.058	-0.058	0.1007	0.2014
i_6	2	[0.1, 0.2]	3.058	-1.058	0.1002	0.2004
i_8	1	[0.1, 0.2]	2.307	-1.307	0.0995	0.1990
i_9	5	[0.1, 0.2]	5.307	-0.307	**0.0994**	**0.1987**

Maka didapat nilai *final update* p_{u_1} adalah **0.0994** dan **0.1987**.

Dan seterusnya untuk perhitungan *final update* faktor laten pengguna $p_{u_1} - p_{u_5}$ dapat dilihat pada tabel 3.4:

Tabel 3. 4. *Final Update* Seluruh Faktor Laten Pengguna (p_u)

User	$p_u^{(1)}$	$p_u^{(2)}$
u_1	0.0994	0.1987
u_2	0.1006	0.2012
u_3	0.0993	0.1987
u_4	0.1008	0.2017
u_5	0.0988	0.1977

Perhitungan vektor faktor laten (q_i):

Contoh perhitungan faktor laten pengguna q_{i_1} dengan interaksi terhadap item u_1:

Hitung *rating* prediksi pengguna u_1 dengan item i_1 dengan persamaan (1.1):

$\hat{\boldsymbol{r}}_{\boldsymbol{u_1,i_1}} = 3.171 + 0.258 + 0.329 + (0.01 \cdot 0.01) + (0.02 \cdot 0.02)$

$\hat{\boldsymbol{r}}_{\boldsymbol{u_1,i_1}} = 3.808$

Hitung *error* prediksi dengan persamaan (1.5):

$\hat{\boldsymbol{e}}_{\boldsymbol{u_1,i_1}} = 5 - 3.808 = 1.193$

Update faktor laten item q_{i_1} (k=2) dengan persamaan (1.7):

Dimensi 1: $\boldsymbol{q}_{\boldsymbol{i_1}}^{\boldsymbol{(1)}} = 0.1 + 0.005 \cdot (1.192 \cdot 0.1 - 0.02 \cdot 0.1) = 0.1006$

Dimensi 2: $\boldsymbol{q}_{\boldsymbol{i_1}}^{\boldsymbol{(2)}} = 0.2 + 0.005 \cdot (1.192 \cdot 0.2 - 0.02 \cdot 0.2) = 0.2012$

Berikut *update* faktor laten item (q_{i_1}) untuk setiap pengguna ($u_1 - u_5$) yang telah memberi rating pada item (i_1) pada tabel 3.5:

Tabel 3. 5. Update Faktor Laten Item (q_{i_1})

User	**Rating**	$\boldsymbol{p_u}$	$\hat{\boldsymbol{r}}_{\boldsymbol{u,i}}$	$\hat{\boldsymbol{e}}_{\boldsymbol{u,i}}$	$\boldsymbol{q}_{\boldsymbol{i_1}}^{\boldsymbol{(1)}}$	$\boldsymbol{q}_{\boldsymbol{i_1}}^{\boldsymbol{(2)}}$
$\boldsymbol{u_1}$	5	[0.1, 0.2]	3.807	1.193	0.1006	0.2012
$\boldsymbol{u_2}$	3	[0.1, 0.2]	3.521	-0.521	0.1003	0.2006
$\boldsymbol{u_3}$	4	[0.1, 0.2]	3.379	0.829	0.1007	0.2014
$\boldsymbol{u_5}$	2	[0.1, 0.2]	3.236	-1.171	**0.1001**	**0.2002**

Maka didapat nilai *final update* q_{i_1} adalah **0.1001** dan **0.2002**.

Dan seterusnya untuk perhitungan *final update* faktor laten item $q_{i_1} - q_{i_{10}}$ dapat dilihat pada tabel 3.6:

Tabel 3. 6. *Final Update* Seluruh Faktor Laten Item (q_i)

Item	$\boldsymbol{q}_{\boldsymbol{i}}^{\boldsymbol{(1)}}$	$\boldsymbol{q}_{\boldsymbol{i}}^{\boldsymbol{(2)}}$
$\boldsymbol{i_1}$	0.1001	0.2002
$\boldsymbol{i_2}$	0.0998	0.1987
$\boldsymbol{i_3}$	0.0999	0.1997
$\boldsymbol{i_4}$	0.1007	0.2013
$\boldsymbol{i_5}$	0.1003	0.2001
$\boldsymbol{i_6}$	0.0994	0.1987
$\boldsymbol{i_7}$	0.0997	0.2006
$\boldsymbol{i_8}$	0.0986	0.1972
$\boldsymbol{i_9}$	0.1007	0.2015
$\boldsymbol{i_{10}}$	0.1004	0.2008

Hitung skor prediksi

Setelah mendapatkan nilai semua variabel, *rating* kosong pada data tabel 3.2 dapat dihitung menggunakan persamaan (1.1) sebagai berikut:

Contoh perhitungan skor prediksi *rating* pengguna (u_1) pada item (i_5):

$$\hat{r}_{u_1,i_5} = 3.171 + 0.257 + 0.079 + (0.0994 * 0.1003) + (0.1987 * 0.2001)$$

$$\hat{r}_{u_1,i_5} \approx 3.56$$

dan seterusnya untuk semua *rating* kosong pada tabel 3.2, sehingga didapat skor prediksi SVD pada tabel 3.7:

Tabel 3. 7. Hasil Skor Prediksi *Rating* SVD

User/Item	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	5	3	4	4	**3.56**	2	**3.31**	1	5	**4.31**
u_2	3	2	4	**3.77**	3	5	4	1	**5.02**	**4.02**
u_3	4	5	**3.13**	3	2	1	2	4	**4.88**	**3.88**
u_4	**3.81**	**3.06**	3	5	4	**3.06**	1	2	5	4
u_5	2	1	2	3	4	3	5	**1.73**	**4.74**	**3.74**

Keterangan:

Nilai dalam kolom berwarna biru muda merupakan skor prediksi SVD.

Dalam contoh perhitungan ini hanya dilakukan update parameter faktor laten pengguna dan faktor laten item. Namun, pada implementasi sebenarnya dalam algoritma SVD, proses optimasi dengan SGD juga mencakup pembaruan parameter bias pengguna (b_u) dan bias item (b_i) secara bersamaan dengan faktor laten. Selama pelatihan, model SVD secara iteratif meminimalkan fungsi objektif (*loss function*) yang terdiri dari selisih kuadrat antara rating aktual dan prediksi ditambah dengan term regularisasi untuk mencegah overfitting (Zhang, 2022). Optimasi ini terus berlangsung hingga *loss function* mencapai nilai minimum lokal atau model telah menyelesaikan jumlah epoch yang telah ditentukan. Hasil dari proses ini adalah model yang mampu menghasilkan prediksi rating yang mendekati rating aktual, dengan parameter yang telah dilatih mencerminkan preferensi pengguna dan karakteristik item secara laten.

3.4.2 KNN

Pada tahapan implementasi algoritma KNN diperlukan perhitungan *similarity* terlebih dahulu antar item (*item-based*). Pada penelitian ini digunakan perhitungan *cosine similarity* yang dihitung menggunakan persamaan (2.1).

Hitung *cosine similarity* antar item:

Contoh perhitungan similarity pada item (i_1) dengan item (i_2):

$$\boldsymbol{cosine}(\boldsymbol{i_1}, \boldsymbol{i_2}) = \frac{(5 * 3) + (3 * 2) + (4 * 5) + (2 * 1)}{\sqrt{5^2 + 3^2 + 4^2 + 5^2} \cdot \sqrt{3^2 + 2^2 + 5^2 + 1^2}} = 0.937$$

dan seterusnya hitung *similarity* antar item sehingga didapat nilai-nilai pada tabel 3.9:

Tabel 3. 8. *Cosine Similarity* Antar Item

Item	$\boldsymbol{i_1}$	$\boldsymbol{i_2}$	$\boldsymbol{i_3}$	$\boldsymbol{i_4}$	$\boldsymbol{i_5}$	$\boldsymbol{i_6}$	$\boldsymbol{i_7}$	$\boldsymbol{i_8}$	$\boldsymbol{i_9}$	$\boldsymbol{i_{10}}$
$\boldsymbol{i_1}$	1.000									
$\boldsymbol{i_2}$	0.937	1.000								
$\boldsymbol{i_3}$	0.973	0.980	1.000							
$\boldsymbol{i_4}$	0.971	0.870	0.972	1.000						
$\boldsymbol{i_5}$	0.862	0.678	0.928	0.966	1.000					
$\boldsymbol{i_6}$	0.763	0.615	0.919	0.917	0.910	1.000				
$\boldsymbol{i_7}$	0.830	0.626	0.831	0.724	0.879	0.932	1.000			
$\boldsymbol{i_8}$	0.800	0.956	0.893	0.802	0.770	0.473	0.667	1.000		
$\boldsymbol{i_9}$	1.000	1.000	0.990	0.994	1.000	1.000	1.000	0.949	1.000	
$\boldsymbol{i_{10}}$	0.000	0.000	1.000	1.000	1.000	0.000	1.000	1.000	1.000	1.000

Keterangan:

Nilai 1.0 sangat mirip

Nilai 0.0 tidak ada kemiripan

Kolom abu-abu dibiarkan kosong agar tidak menampilkan nilai berulang

Setelah didapatkan nilai *similarity* antar item, skor prediksi dihitung dengan memperhitungkan nilai kesamaan antar item teteangga (*k*) sehingga prediksi *rating* dapat dihitung menggunakan persamaan (2.2). Contoh perhitungan skor prediksi *rating* pengguna (u_1) terhadap item (i_5):

$$\hat{r}_{u_1,i_5} = \frac{(5 * 0.862) + (3 * 0.678) + (4 * 0.928) + (4 * 0.966) + (2 * 0.91) + (1 * 0.77) + (5 * 1)}{|0.862| + |0.678| + |0.928| + |0.966| + |0.91| + |0.77| + |1|}$$

$$\hat{r}_{u_1,i_5} = 3.518 \approx 3.52$$

dan seterusnya menghitung semua *rating* kosong pada tabel 3.2, sehingga didapat skor prediksi KNN pada tabel 3.9:

Tabel 3. 9. Hasil Skor Prediksi *Rating* KNN

User/Item	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	5	3	4	4	**3.52**	2	**3.53**	1	5	**3.50**
u_2	3	2	4	**3.17**	3	5	4	1	**3.16**	**3.00**
u_3	4	5	**3.04**	3	2	1	2	4	**2.99**	**2.75**
u_4	**3.43**	**3.43**	3	5	4	**3.47**	1	2	5	4
u_5	2	1	2	3	4	3	5	**2.72**	**2.86**	**3.50**

Nilai *k* pada perhitungan manual ini adalah *k*=7, karena pada data contoh setiap pengguna me-*rating* 7 film, sehingga dihitung semua *similarity* antar item dalam perhitungan skor prediksi. Namun pada implementasi data sebenarnya akan digunakan *k*=40 karena menurut Airen & Agrawal (2022) pada artikel yang berjudul "*Movie Recommender System Using K-Nearest Neighbors Variants*", hasil pengujian seperti RMSE dan MAE menjadi stabil ketika jumlah tetangga (*neighbors*) mencapai 40.

3.5 Pembobotan Skor Prediksi (SVD-KNN)

Setelah dihasilkan skor prediksi *rating* pengguna *u* terhadap item *i* menggunakan *model-based* SVD dan *memory-based* KNN, dilakukan pembobotan menggunakan teknik *weighted hybrid*. Pada perhitungan contoh ini digunakan bobot seimbang yaitu 0.5 untuk kedua skor prediksi (*weighted average*), namun pada penelitian ini dalam data *real* akan diuji besarnya pembobotan atau *alpha* (α) dan *beta* (β) di rentang [0.1 – 0.9] untuk mencari nilai optimal untuk pengujian RMSE dan MAE.

Pembobotan skor prediksi untuk dapat dihitung menggunakan persamaan (3). Contoh perhitungan pembobotan skor prediksi *rating* pengguna u_1 terhadap item i_5:

$$\left.\begin{matrix} \boldsymbol{\alpha} = 0.5 \\ \boldsymbol{\beta} = 0.5 \end{matrix}\right\} \; dimana, \alpha + \beta = 1$$

$\widehat{\boldsymbol{p}}_{\boldsymbol{u_1},\boldsymbol{i_5}} = 0.5 * 3.56 + 0.5 * 3.52$

$\widehat{\boldsymbol{p}}_{\boldsymbol{u_1},\boldsymbol{i_5}} = 3.54$

dan seterusnya untuk pembobotan skor prediksi *rating* yang telah dihasilkan oleh SVD dan KNN. Seluruh nilai pembobotan SVD-KNN dapat dilihat pada tabel 3.10:

Tabel 3. 10. Hasil Pembobotan Skor Prediksi *Rating* SVD-KNN

User/Item	$\boldsymbol{i_1}$	$\boldsymbol{i_2}$	$\boldsymbol{i_3}$	$\boldsymbol{i_4}$	$\boldsymbol{i_5}$	$\boldsymbol{i_6}$	$\boldsymbol{i_7}$	$\boldsymbol{i_8}$	$\boldsymbol{i_9}$	$\boldsymbol{i_{10}}$
$\boldsymbol{u_1}$	5	3	4	4	**3.54**	2	**3.42**	1	5	**3.90**
$\boldsymbol{u_2}$	3	2	4	**3.47**	3	5	4	1	**4.09**	**3.51**
$\boldsymbol{u_3}$	4	5	**3.08**	3	2	1	2	4	**3.94**	**3.31**
$\boldsymbol{u_4}$	**3.62**	**3.24**	3	5	4	**3.26**	1	2	5	4
$\boldsymbol{u_5}$	2	1	2	3	4	3	5	**2.23**	**3.80**	**3.62**

Maka top-n rekomendasi item untuk pengguna dapat disarankan berdasarkan dengan besarnya skor prediksi *rating* yang telah dibobotkan. Keseluruhan rekomendasi dengan 2 prediksi tertinggi (*top-2*) dapat dilihat pada tabel 3.11:

Tabel 3. 11. Hasil Top-2 Rekomendasi Film

Pengguna	**Top-2 Rekomendasi Film**			
	Rekomendasi 1	**Skor 1**	**Rekomendasi 2**	**Skor 2**
u_1	i_{10}	3.90	i_5	3.54
u_2	i_9	4.09	i_{10}	3.51
u_3	i_9	3.94	i_{10}	3.31
u_4	i_1	3.62	i_6	3.26
u_5	i_9	3.80	i_{10}	3.62

3.6 Evaluasi

Setelah melakukan perhitungan skor prediksi dari masing-masing pendekatan dan menggabungkan kedua skor prediksi (SVD-KNN) menggunakan pembobotan, tahap selanjutnya adalah mengevaluasi keakuratan model. Teknik evaluasi yang secara umum telah digunakan untuk sistem rekomendasi film adalah RMSE dan MAE.

Untuk keperluan pengujian skor prediksi *rating* menggunakan RMSE dan MAE maka disiapkan data uji. Berikut pada tabel 3.11 adalah contoh data uji dengan *rating* asli yang telah di-*rating* oleh pengguna:

Tabel 3. 12. Contoh Data Uji *Rating* Asli User-Item

User/Item	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
u_1	5	3	4	4	**4**	2	**3**	1	5	**4**
u_2	3	2	4	**4**	3	5	4	1	**4**	**4**
u_3	4	5	**3**	3	2	1	2	4	**4**	**3**
u_4	**4**	**3**	3	5	4	**3**	1	2	5	4
u_5	2	1	2	3	4	3	5	**2**	**4**	**4**

Hitung terlebih dahulu *error* prediksi (selisih rating asli dengan rating prediksi), nilai mutlak (*absolute*) *error*, dan kuadrat *error* untuk mempermudah perhitungan uji evaluasi RMSE dan MAE, pada tabel 3.12.

Tabel 3. 13. Nilai *Error*, *Absolute Error*, dan Kuadrat *Error*

SVD			**KNN**			***Weighted* (SVD-KNN)**		
e	$\|e\|$	e^2	e	$\|e\|$	e^2	e	$\|e\|$	e^2
0.443	0.443	0.196	0.482	0.482	0.232	0.463	0.463	0.214
-0.307	0.307	0.094	-0.526	0.526	0.277	-0.416	0.416	0.173
-0.307	0.307	0.094	0.500	0.500	0.250	0.096	0.096	0.009
0.228	0.228	0.052	0.831	0.831	0.690	0.529	0.529	0.280
-1.022	1.022	1.045	0.843	0.843	0.710	-0.090	0.090	0.008
-0.022	0.022	0.000	1.000	1.000	1.000	0.489	0.489	0.239
-0.128	0.128	0.016	-0.035	0.035	0.001	-0.082	0.082	0.007
-0.879	0.879	0.772	1.007	1.007	1.015	0.064	0.064	0.004
-0.878	0.878	0.772	0.250	0.250	0.063	-0.314	0.314	0.099
0.192	0.192	0.037	0.569	0.569	0.324	0.381	0.381	0.145
-0.057	0.057	0.003	-0.432	0.432	0.187	-0.245	0.245	0.060
-0.057	0.057	0.003	-0.467	0.467	0.218	-0.262	0.262	0.069
0.266	0.266	0.071	-0.720	0.720	0.518	-0.227	0.227	0.052
-0.735	0.735	0.541	1.142	1.142	1.304	0.203	0.203	0.041
Total	**5.787**	**3.767**		**9.303**	**7.038**		**4.244**	**1.546**

Setelah didapat jumlah *absolute error* dan kuadrat *error* maka dapat dihitung seberapa besar nilai MAE dan RMSE menggunakan persamaan (4) dan (5), sehingga hasil evaluasi seperti pada tabel 3.13:

Tabel 3. 14. Hasil Evaluasi RMSE dan MAE

Pendekatan	**RMSE**	**MAE**
SVD	0.386	0.501
KNN	0.620	0.685
***Weighted* (SVD-KNN)**	**0.283**	**0.321**

Hasil evaluasi menunjukkan bahwa teknik pembobotan pada algoritma SVD dan KNN memberikan performa paling optimal dengan nilai RMSE sebesar 0.283 dan MAE sebesar 0.321. Nilai ini lebih rendah dibandingkan dengan pendekatan SVD yang menghasilkan RMSE sebesar 0.386 dan MAE sebesar 0.501, serta pendekatan KNN yang memiliki RMSE sebesar 0.620 dan MAE sebesar 0.685. Hal ini mengindikasikan bahwa metode penggabungan ini mampu mengombinasikan kelebihan SVD dalam menangkap pola laten *user–item* dengan kekurangan KNN dalam dataset yang *sparse*, sehingga menghasilkan prediksi yang lebih akurat (Gong et al., 2009). Berdasarkan hasil tersebut, pendekatan ini dipilih sebagai strategi utama dalam sistem rekomendasi berbasis website yang dibangun.

3.7 Implementasi Website

Pada tahap ini dilakukan implementasi kedua algoritma terhadap sistem rekomendasi film berbasis website yang akan dibangun menggunakan *framework* streamlit berbasis bahasa pemrograman python. Implementasi algoritma (SVD dan KNN) akan dijadikan model terlebih dahulu menggunakan *library* pickle dari python kemudian akan di-*import* bersamaan dengan dataset dan *library* lainnya.

Sistem rekomendasi film berbasis website yang akan dibangun merupakan demonstrasi implementasi algoritma SVD dan KNN yang dibobotkan (*weighted*) pada suatu sistem yang sudah berjalan, sehingga hasil rekomendasi film merupakan prediksi tertinggi berdasarkan dengan userID pengguna yang ada pada dataset pada rentang ID 1 (satu) hingga 6040 (enam ribu empat puluh). Hal ini dikarenakan kelemahan dalam pendekatan *collaborative filtering* yang terbatas pada permasalahan *cold start*, yaitu dalam penanganan pengguna baru maupun item baru yang masuk ke dalam sistem sehingga tidak dapat menampilkan hasil rekomendasi karena pengguna baru belum me-*rating* suatu item dan item baru belum di-*rating* oleh seorang pengguna.

3.7.1 Lingkungan Pengembangan

Website dikembangkan menggunakan bahasa pemrograman Python dengan memanfaatkan framework Streamlit yang memungkinkan pembuatan aplikasi web berbasis data secara cepat dan interaktif. Sistem dijalankan secara lokal (l*ocalhost*) tanpa menggunakan basis data (*database*), sehingga seluruh data diproses secara langsung melalui file lokal.

Beberapa *tools* dan *library* yang digunakan antara lain:

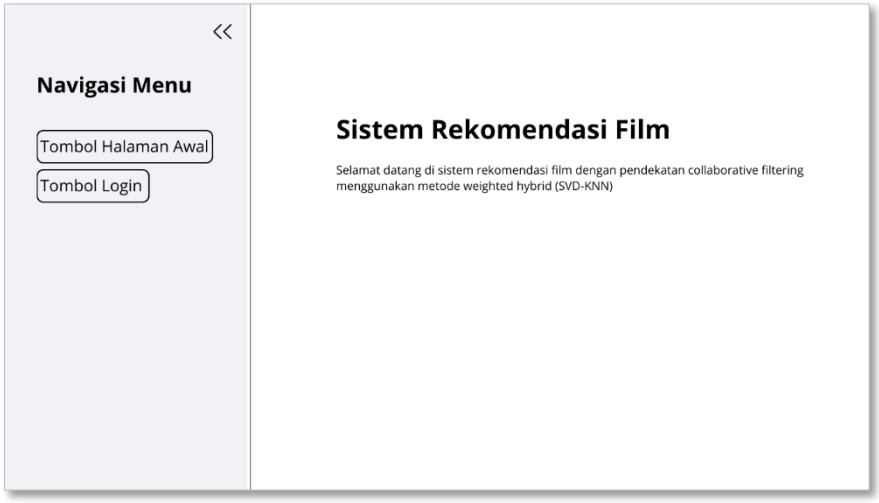
- Bahasa : Python
- *Framework* : Streamlit
- *Library* : surprise, pandas, numpy, scikit-learn, pickle
- Dataset : MovieLens 1M (ratings.dat, movies.dat, imageURL.csv)
- Editor : Visual Studio Code
- Akses : Jaringan lokal (*localhost*) pada *port* 8501

Dari dokumentasi Streamlit (2025), streamlit merupakan kerangka kerja *open-source* berbasis Python yang dirancang untuk memudahkan data scientist dan engineer AI/ML dalam membangun aplikasi data interaktif hanya dengan beberapa baris kode. *Framework* ini memungkinkan pengembangan dan penyebaran aplikasi data yang kuat secara cepat dan efisien (Eedi & Kolla, 2022). Berbeda dengan *framework* website tradisional yang mungkin memerlukan pemahaman mendalam tentang *frontend* dan *backend*, streamlit dirancang untuk memungkinkan data scientist atau peneliti membangun antarmuka pengguna hanya dengan kode Python. Dengan menggunakan Streamlit, pengembangan sistem rekomendasi film dapat diintegrasikan dengan mudah karena kelengkapan *library* yang digunakan untuk pengembangan. Pada penelitian ini akan digunakan *library* pandas, numpy, surprise, dan matplotlib.

3.7.2 Struktur Website

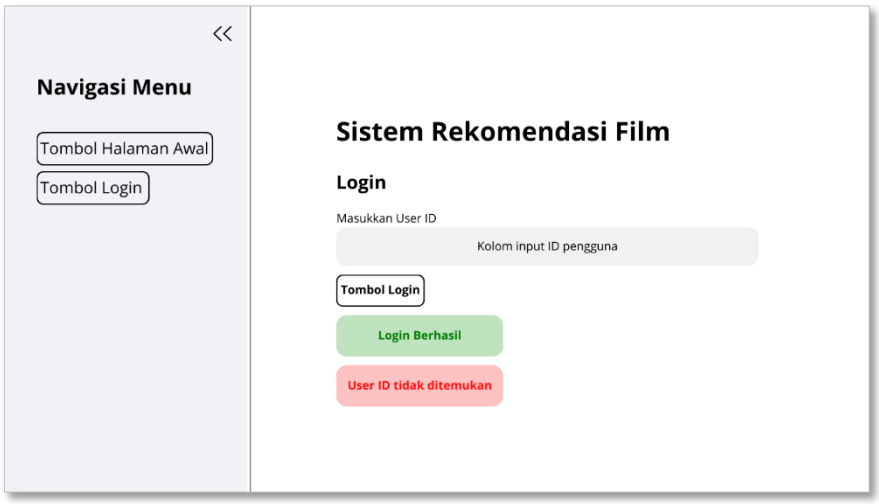
Struktur website akan dibuat dalam 3 (tiga) halaman utama, yaitu halaman awal (*landing page*), halaman *login*, dan halaman rekomendasi film. Gambar 3.2, 3.3, dan 3.4 merupakan desain antarmuka (*User Interface* - UI) dari sistem rekomendasi film berbasis website yang akan dibangun.

- Halaman Awal (*Landing Page)*



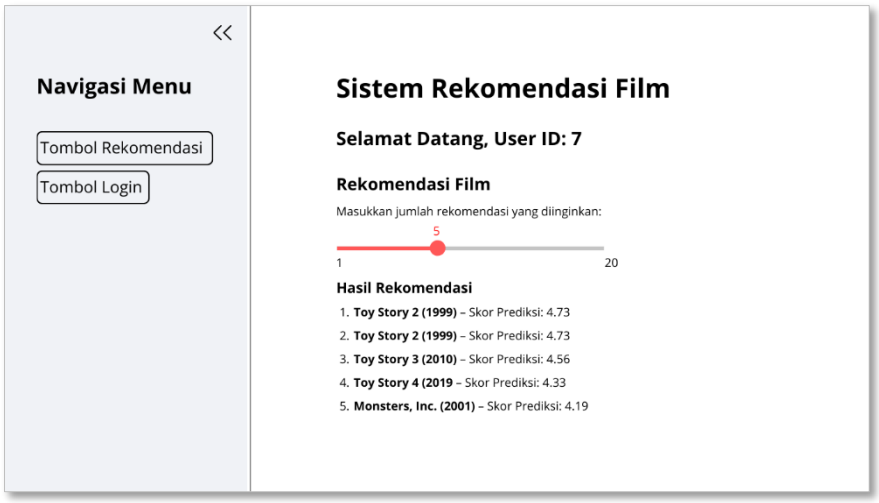
Gambar 3. 2. Desain UI Halaman Awal (*Landing Page*)

- Halaman Login



Gambar 3. 3. Desain UI Halaman Login

- Halaman Rekomendasi Film



Gambar 3. 4. Desain UI Halaman Rekomendasi Film

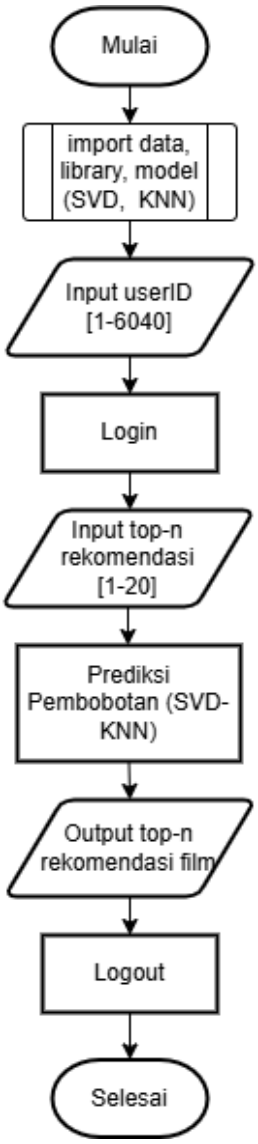
3.7.3 Fitur Website

Fitur-fitur pada sistem rekomendasi film berbasis website yang akan diimplementasikan adalah sebagai berikut:

- Login berdasarkan User ID pada dataset Movielens-1M (1-6040).
- *Input* berupa *slider* top-N rekomendasi yang diinginkan.
- *Output* hasil rekomendasi berdasarkan model SVD dan KNN yang dibobotkan.
- Logout untuk mengakhiri sesi penggunaan dan kembali ke halaman awal.

3.7.4 Alur Kerja Sistem

Gambar 3.5 merupakan diagram alur kerja sistem rekomendasi film berbasis website yang mengimplementasi pendekatan *collaborative filtering* menggunakan pembobotan kedua algoritma (SVD-KNN) mulai dari login hingga logout oleh pengguna.



Gambar 3. 5. Diagram Alur Sistem

3.7.5 Pengujian Sistem

Pengujian terhadap sistem berbasis website akan dilakukan menggunakan *black-box testing*, di mana pengujian difokuskan pada keluaran sistem berdasarkan input yang diberikan, tanpa memperhatikan struktur internal dari kode program. *Black-box testing* adalah metode pengujian perangkat lunak tanpa pengetahuan tentang struktur internal atau implementasi sistem yang diuji (Lee et al., 2014). Pengujian akan dilakukan dengan *functional testing* yang melakukan *positive testing* dan *negative testing* berdasarkan seluruh fitur website.

Menurut Salih & Saefullah (2024) pada artikelnya yang membahas *black-box testing* pada suatu website, *positive testing* merupakan pengujian dilakukan dengan memberikan input yang sesuai (valid) untuk memastikan bahwa setiap fitur bekerja sesuai dengan yang diharapkan. Sebaliknya, pada *negative testing*, sistem diuji dengan input yang tidak sesuai (invalid) dan memastikan muncul peringatan atau kegagalan sistem untuk mengeksekusi *input* pengguna.

3.8 Jadwal Penelitian

Jadwal penelitian ini dilakukan mulai dari Bulan April hingga Bulan Juli mulai dari studi literatur, pembuatan sistem, pengujian sistem, dan penyusunan laporan akhir.

Tabel 3. 15. Jadwal Penelitian

No	Uraian	April				Mei				Juni				Juli			
		Minggu Ke															
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi Literatur	■	■	■													
2	Alat dan Bahan				■												
3	Perancangan Sistem				■	■	■										
4	Implementasi SVD							■	■								
5	Implementasi KNN							■	■								
6	Implementasi *Weighted Hybrid (*SVD-KNN)							■	■								
7	Implementasi Website									■	■	■					
8	Uji Evaluasi											■					
9	Pengujian Sistem												■	■			
10	Penyusunan Laporan/Tugas Akhir														■	■	■

DAFTAR PUSTAKA

Abubakar, R. (2021). *Pengantar Metodologi Penelitian*. SUKA-Press UIN Sunan Kalijaga.

Ahmed, E., & Letta, A. (2023). Book Recommendation Using Collaborative Filtering Algorithm. *Applied Computational Intelligence and Soft Computing*, *2023*. https://doi.org/10.1155/2023/1514801

Airen, S., & Agrawal, J. (2022). Movie Recommender System Using K-Nearest Neighbors Variants. *National Academy Science Letters*, *45*(1), 75–82. https://doi.org/10.1007/s40009-021-01051-0

Akbar, R., Richasdy, D., & Dharayani, R. (2023). *Sistem Rekomendasi Buku Dengan Collaborative Filtering Menggunakan Metode Singular Value Decomposition (SVD)*.

Akbar, Y. A., Baizal, Z. K. A., & Wibowo, A. T. (2021). Tourism Recommender System using Weighted Parallel Hybrid Method with Singular Value Decomposition. *Indonesian Journal on Computing*, *6*. https://doi.org/10.34818/indojc.2021.6.2.579

Anwar, T., & Uma, V. (2021). Comparative study of recommender system approaches and movie recommendation using collaborative filtering. *International Journal of System Assurance Engineering and Management*, *12*(3), 426–436. https://doi.org/10.1007/s13198-021-01087-x

Bauer, C., Said, A., & Zangerle, E. (2024). *Evaluation Perspectives of Recommender Systems: Driving Research and Education* (Vol. 14). Association for Computing Machinery. https://doi.org/10.4230/DagRep.14.5.58

Bhowmick, H., Chatterjee, A., & Sen, J. (2021). *Comprehensive Movie Recommendation System*. https://doi.org/10.48550/arXiv.2112.12463

Delimayanti, M. K., Laya, M., Warsuta, B., Faydhurrahman, M. B., Khairuddin, M. A., Ghoyati, H., Mardiyono, A., & Naryanto, R. F. (2022). Web-Based Movie Recommendation System using Content-Based Filtering and KNN Algorithm. *Proceedings - 2022 9th International Conference on Information Technology, Computer and Electrical Engineering, ICITACEE 2022*, 314–318. https://doi.org/10.1109/ICITACEE55701.2022.9923974

Do, H.-Q., Le, T.-H., & Yoon, B. (2020). *Dynamic Weighted Hybrid Recommender Systems*. IEEE.

Eedi, H., & Kolla, M. (2022). A Customized Recommendation System using Streamlit. *Journal for New Zealand Herpetology*, *11*(3), 2022.

Gong, S. J., Wu Ye, H., & Tan, H. S. (2009). Combining memory-based and model-based collaborative filtering in recommender system. *Proceedings of the 2009 Pacific-Asia Conference on Circuits, Communications and System, PACCS 2009*, 690–693. https://doi.org/10.1109/PACCS.2009.66

Hardani, Auliya, N. H., Andriani Helmina, Fardani, R. A., Ustiawaty, J., Utami, E. F., Sukmana, D. J., & Istiqomah, R. R. (2020). *Buku Metode Penelitian Kualitatif & Kuantitatif*. https://www.researchgate.net/publication/340021548

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, *5*(4). https://doi.org/10.1145/2827872

Hssina, B., Grota, A., & Erritali, M. (2021). Recommendation system using the k-nearest neighbors and singular value decomposition algorithms. In *International Journal of Electrical and Computer Engineering* (Vol. 11, Issue 6, pp. 5541–5548). Institute of Advanced Engineering and Science. https://doi.org/10.11591/ijece.v11i6.pp5541-5548

Hug, N. (2020). Surprise: A Python library for recommender systems. *Journal of Open Source Software*, *5*(52), 2174. https://doi.org/10.21105/joss.02174

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. In *Egyptian Informatics Journal* (Vol. 16, Issue 3, pp. 261–273). Elsevier B.V. https://doi.org/10.1016/j.eij.2015.06.005

Lee, N., Jung, J. J., Selamat, A., & Hwang, D. (2014). Black-box testing of practical movie recommendation systems: A comparative study. *Computer Science and Information Systems*, *11*(1), 241–249. https://doi.org/10.2298/CSIS130226006L

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook* (pp. 1–35). Springer US. https://doi.org/10.1007/978-0-387-85820-3_1

Saifudin, I., & Widiyaningtyas, T. (2024). Systematic Literature Review on Recommender System: Approach, Problem, Evaluation Techniques, Datasets. *IEEE Access*, *12*, 19827–19847. https://doi.org/10.1109/ACCESS.2024.3359274

Salih, Y., & Saefullah, R. (2024). Black Box Testing on Website-Based Guestbook Registration Applications. *International Journal of Mathematics, Statistics, and Computing*, *2*(2), 44–49.

Streamlit. (2025). *Streamlit Documentation*. https://docs.streamlit.io/

Suriati, S., Dwiastuti, M., & Tulus, T. (2017). Weighted hybrid technique for recommender system. *Journal of Physics: Conference Series*, *930*(1).

https://doi.org/10.1088/1742-6596/930/1/012050

Tiara, V. (2024). *Behind Netflix's Recommendations: The Influence of Singular Value Decomposition (SVD)*. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Makalah/Makalah-IF2123-Algeo-2024%20(42).pdf

Yap, Z. T., Haw, S. C., & Binti Ruslan, N. E. (2024). Hybrid-based food recommender system utilizing KNN and SVD approaches. *Cogent Engineering*, *11*(1). https://doi.org/10.1080/23311916.2024.2436125

Zhang, Y. (2022). *An Introduction to Matrix factorization and Factorization Machines in Recommendation System, and Beyond*. http://arxiv.org/abs/2203.11026

LAMPIRAN-LAMPIRAN