



TUGAS AKHIR - IF184802

**STUDI PERMASALAHAN *K-MOST PROMISING PRODUCTS*
BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI
DENGAN SERIAL WAKTU**

HAFARA FIRDAUSI
NRP 05111540000043

Dosen Pembimbing 1
Bagus Jati Santoso, S.Kom., Ph.D.

Dosen Pembimbing 2
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

Halaman ini sengaja dikosongkan



TUGAS AKHIR - IF184802

**STUDI PERMASALAHAN *K-MOST PROMISING PRODUCTS*
BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI
DENGAN SERIAL WAKTU**

HAFARA FIRDAUSI
NRP 05111540000043

Dosen Pembimbing 1
Bagus Jati Santoso, S.Kom., Ph.D.

Dosen Pembimbing 2
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

Halaman ini sengaja dikosongkan



UNDERGRADUATE THESES - IF184802

**OBTAINING K-MOST PROMISING PRODUCTS BASED ON
TIME INTERVAL ON MULTIDIMENSIONAL TIME SERIES
DATA**

HAFARA FIRDAUSI
NRP 05111540000043

Supervisor 1
Bagus Jati Santoso, S.Kom., Ph.D.

Supervisor 2
Henning Titi Ciptaningtyas, S.Kom., M.Kom.

INFORMATICS DEPARTMENT
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya, 2019

Halaman ini sengaja dikosongkan

**STUDI PERMASALAHAN *K-MOST PROMISING*
PRODUCTS BERBASIS INTERVAL WAKTU PADA DATA
MULTIDIMENSI DENGAN SERIAL WAKTU**

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada

Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

Hafara Firdausi

NRP: 05111540000043

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Bagus Jati Santoso, S.Kom., Ph.D.
NIP: 198611252018031001 (Pembimbing 1)

Henning Titi Ciptaningtyas, S.Kom., M.Kom.
NIP: 198407082010122004 (Pembimbing 2)

**SURABAYA
JUNI 2019**

Halaman ini sengaja dikosongkan

STUDI PERMASALAHAN *K-MOST PROMISING PRODUCTS* BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI DENGAN SERIAL WAKTU

Nama : HAFARA FIRDAUSI
NRP : 05111540000043
Departemen : Informatika FTIK-ITS
Pembimbing I : Bagus Jati Santoso, S.Kom., Ph.D.
Pembimbing II : Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Abstrak

Kemajuan ilmu pengetahuan dan teknologi, terutama di bidang analisis data, telah mempengaruhi cara perusahaan dalam menjalankan bisnis, yaitu dengan mengumpulkan data preferensi pelanggan dari data penjualan produk, kemudian memanfaatkannya untuk mendapatkan informasi yang dapat digunakan untuk membuat keputusan bisnis yang tepat. Saat ini, sudah ada penelitian yang mengembangkan strategi pemilihan produk dengan melakukan pencarian k-produk yang paling banyak diminati oleh pelanggan bernama k-Most Promising Products (k-MPP). Komputasi k-MPP menggunakan dua tipe kueri skyline, yaitu dynamic skyline dan reverse skyline. Sayangnya, komputasi k-MPP tidak mempertimbangkan variabel waktu dalam algoritme perhitungannya dan tidak dapat digunakan untuk memproses kueri berbasis interval waktu.

Tugas Akhir ini bertujuan untuk menjawab permasalahan k-MPP berbasis interval waktu pada data multidimensi dengan serial waktu dengan memodelkan kueri k-MPPTI (k-Most Promising Products in Time Intervals) dan merancang kerangka kerja algoritme yang dapat memproses kueri tersebut. Ada tiga jenis algoritme yang dibuat dan dibandingkan, yaitu k-MPPTI (menggunakan kueri dynamic skyline dan reverse skyline),

k-MPPTI NoRSL (menggunakan kueri dynamic skyline saja), dan k-MPPTI NoRSL-P (menggunakan teknik komputasi paralel). Efektivitas dan efisiensi algoritme diuji menggunakan data asli dan sintetis.

Hasil uji coba menunjukkan bahwa algoritme k-MPPTI NoRSL memiliki performa yang lebih baik daripada algoritme k-MPPTI karena dapat memberikan hasil kueri dengan waktu eksekusi lima kali lebih cepat dan penggunaan memori satu kali lebih hemat dibandingkan dengan algoritme k-MPPTI.

Kata Kunci: *Strategi Pemilihan Produk, Kueri, Dynamic Skyline, Reverse Skyline, Interval Waktu*

OBTAINING K-MOST PROMISING PRODUCTS BASED ON TIME INTERVAL ON MULTIDIMENSIONAL TIME SERIES DATA

Name	:	HAFARA FIRDAUSI
NRP	:	05111540000043
Major	:	Informatics Department Faculty of IT-ITS
Supervisor I	:	Bagus Jati Santoso, S.Kom., Ph.D.
Supervisor II	:	Henning Titi Ciptaningtyas, S.Kom., M.Kom.

Abstract

The advancement of science and technology, especially in the data analytics area, has influenced the way manufacturers do businesses by collecting customer preferences from product sales data, then using it to obtain some informations to make the right business decision. Currently, there is a product selection strategy by searching for k-most preferred product by customers, namely k-Most Promising Products (k-MPP). This computation uses two types of skyline queries, dynamic skyline and reverse skyline. Unfortunately, k-MPP computation doesn't consider the time variable and can't process query based on time intervals.

This study aims to answer the k-MPP query based on time intervals in multidimensional time series data with serial time by modeling k-Most Promising Products in Time Intervals (k-MPPTI) query and designing an algorithmic framework for processing the query. There are three types of algorithm built and compared namely k-MPPTI (using both dynamic skyline and reverse skyline queries), k-MPPTI NoRSL (only using dynamic skyline), and k-MPPTI NorSL-P (using parallel computing techniques). The effectiveness and efficiency of the algorithm was tested using real and synthetic datasets.

Based on the testing results, k-MPPTI NoRSL algorithm has better performance than k-MPPTI algorithm because it provides query results with execution time five times faster and memory usage one-time more efficient than k-MPPTI algorithm.

Keywords: *Product Selection Strategy, Query, Dynamic Skyline, Reverse Skyline, Time Interval*

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah Swt. atas pertolongan dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

STUDI PERMASALAHAN *K-MOST PROMISING PRODUCTS* BERBASIS INTERVAL WAKTU PADA DATA MULTIDIMENSI DENGAN SERIAL WAKTU.

Penelitian Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya. Dengan selesainya Tugas Akhir ini, diharapkan apa yang telah dikerjakan oleh penulis dapat memberikan manfaat bagi perkembangan ilmu pengetahuan, terutama di bidang teknologi informasi, serta bagi diri penulis sendiri selaku peneliti.

Penulis mengucapkan terima kasih sedalam-dalamnya kepada semua pihak yang telah memberikan dukungan, baik secara langsung maupun tidak langsung, selama penulis mengerjakan Tugas Akhir maupun selama menempuh masa studi antara lain:

1. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Departemen Informatika ITS, Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, beserta segenap dosen dan karyawan Informatika yang telah memberikan ilmu dan pengalamannya, serta menyediakan berbagai fasilitas dan pelayanan sehingga penulis dapat menempuh studi di Informatika dengan nyaman.

2. Bapak Bagus Jati Santoso, S.Kom., Ph.D. selaku dosen pembimbing I dan Ibu Henning Titi Ciptaningtyas, S.Kom., M.Kom. selaku dosen pembimbing II yang telah mendampingi penulis sejak penyusunan proposal serta banyak meluangkan waktunya untuk membimbing, memberikan saran dan solusi untuk menyelesaikan Tugas Akhir ini.
3. Ibu, Bapak, kedua adik, Akbar dan Alam, serta segenap keluarga yang senantiasa memberikan perhatian, dukungan, pengetahuan, serta kasih sayang yang menjadi semangat dan motivasi bagi diri penulis untuk menyelesaikan Tugas Akhir.
4. Teman-teman "Sempol Bunda", Ajeng, Salma, Napik, Bela, dan Yola, yang telah menemani dan mewarnai masa-masa perkuliahan penulis sejak jaman mahasiswa baru.
5. Seluruh teman-teman Laboratorium Arsitektur dan Jaringan Komputer (AJK), Mas Syukron, Mas Thoni, Mas Fatih, Nahda, Satria, Awan, Mas Penyok, Fuad, Didin, Hana, Raldo, Aguel, Khawari, Tamtam, Haura, Lia, Sulton, Mail, Yoga, dan Fawwaz, yang telah menemani, mengganggu, dan membantu penulis selama mengerjakan Tugas Akhir di laboratorium.
6. Emak kos terbaik, Mak Ju, atas segala bantuannya selama penulis menempuh studi, dan teman-teman kosan 36, Jakiya, Mutek, Marisa, Mbak Tatak, Alya, Firda, dan Anca.
7. Teman-teman "Penguasa Kosan", Prames, Kikik, Balqis, Tije, Nilam, dan Rini, yang pernah mengajarkan cara bersenang-senang.
8. Teman-teman *Data Engineers*, Hana dan Rio, sebagai teman seperjuangan dan seperbimbingan Tugas Akhir.
9. Seluruh teman-teman TC 2015, Mas Andre, dan pihak-pihak lain yang tidak bisa penulis sebutkan satu persatu namanya, yang secara sengaja maupun tidak sengaja turut berkontribusi dalam penyelesaian studi dan Tugas Akhir.

Penulis mohon maaf apabila masih ada kekurangan pada Tugas Akhir ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan di kemudian hari. Semoga melalui Tugas Akhir ini Penulis dapat memberikan kontribusi dan manfaat yang sebaik-baiknya.

Surabaya, Juni 2019

Hafara Firdausi

Halaman ini sengaja dikosongkan

DAFTAR ISI

SAMPUL	i
LEMBAR PENGESAHAN	vii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR	xiii
DAFTAR ISI	xvii
DAFTAR TABEL	xxi
DAFTAR GAMBAR	xxiii
DAFTAR KODE SUMBER	xxvii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	4
1.4 Tujuan	4
1.5 Manfaat	5
1.6 Metodologi	5
1.7 Sistematika Penulisan	7
BAB II TINJAUAN PUSTAKA	9
2.1 Daftar Notasi	9
2.2 <i>Skyline</i>	10
2.3 Dominansi Dinamis	12
2.4 <i>Dynamic Skyline</i>	14
2.5 <i>Reverse Skyline</i>	15
2.6 Kueri <i>k</i> -Most Promising Products (<i>k</i> -MPP)	16
2.6.1 <i>Uniform Product Adoption</i> (UPA)	17
2.6.2 Strategi Pemilihan Produk	20
2.7 Data	20
2.7.1 Data Multidimensi	21
2.7.2 Data Serial Waktu	21

2.7.3	Data Multidimensi dengan Serial Waktu	22
2.8	Python	23
2.9	Flask	24
2.10	Vis.js	25
BAB III ANALISIS DAN PERANCANGAN SISTEM	27	
3.1	Daftar Notasi	27
3.2	Analisis Sistem	29
3.2.1	Analisis Permasalahan	29
3.2.2	Deskripsi Umum Sistem	29
3.2.3	Fungsi Sistem	30
3.2.4	Analisis Kebutuhan Fungsional	30
3.3	Perancangan Sistem	31
3.3.1	Struktur Data	32
3.3.2	Algoritme Utama	36
3.3.3	Algoritme Tandingan	57
3.3.4	Arsitektur Aplikasi	57
BAB IV IMPLEMENTASI	59	
4.1	Lingkungan Implementasi	59
4.2	Implementasi <i>Data Precomputing</i>	59
4.2.1	Kelas <i>EventQueue</i>	62
4.2.2	Kelas <i>PandoraBox</i>	64
4.2.3	Kelas <i>ReverseSkyline</i>	65
4.2.4	Kelas <i>Dynamic Skyline</i>	67
4.3	Implementasi Algoritme <i>Query Processing</i>	70
4.4	Implementasi Antarmuka Pengguna	72
BAB V UJI COBA DAN EVALUASI	77	
5.1	Lingkungan Uji Coba	77
5.2	Data Uji Coba	77
5.2.1	Data <i>Independent</i> (IND)	78
5.2.2	Data <i>Anti-Correlated</i> (ANT)	78
5.2.3	Data <i>Forest Cover Type</i> (FC)	78
5.3	Skenario Uji Coba	80
5.3.1	Uji Coba Fungsionalitas	80

5.3.2 Uji Coba Performa	80
5.4 Analisis Hasil Uji Coba	82
5.4.1 Uji Coba Fungsionalitas	82
5.4.2 Uji Coba Performa	87
BAB VI KESIMPULAN DAN SARAN	99
6.1 Kesimpulan	99
6.2 Saran	100
DAFTAR PUSTAKA	103
Lampiran A Kode Sumber	105
BIODATA PENULIS	129

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 2.1	Daftar notasi (bagian 1)	9
Tabel 2.2	Contoh <i>dataset</i> (a) produk P dan (b) preferensi pelanggan C	14
Tabel 2.3	Contoh data <i>time series</i>	22
Tabel 2.4	Contoh data multidimensi dengan serial waktu (a) produk P dan (b) preferensi pelanggan C	23
Tabel 3.1	Daftar notasi (bagian 2)	27
Tabel 3.2	Kebutuhan fungsional	30
Tabel 3.3	<i>Key</i> dari <i>DataStorage</i>	33
Tabel 3.4	Atribut dari <i>EventQueue</i>	35
Tabel 3.5	Contoh <i>dataset</i> (a) produk P dan (b) preferensi pelanggan C	38
Tabel 3.6	<i>EventQueue</i>	40
Tabel 3.7	Kontribusi pasar (1)	55
Tabel 3.8	Kontribusi pasar (2)	57
Tabel 5.1	Atribut himpunan data <i>Forest Cover Type</i>	79
Tabel 5.2	Skenario uji coba fungsionalitas	80
Tabel 5.3	Skenario uji coba performa <i>data precomputing</i>	81
Tabel 5.4	Hasil uji coba fungsionalitas	83

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1	Titik <i>skyline</i> dari data produk pada Tabel 2.2	11
Gambar 2.2	(a) Komputasi <i>dynamic skyline</i> dari pelanggan c_4 dan (b) <i>dynamic skyline</i> dari pelanggan c_{10}	15
Gambar 2.3	Komputasi <i>reverse skyline</i> dari produk p_8	17
Gambar 3.1	Struktur data <i>dictionary</i> produk	32
Gambar 3.2	Struktur data <i>dictionary</i> pelanggan	33
Gambar 3.3	Contoh <i>Pandora Box</i> dari <i>dataset 3.5</i>	35
Gambar 3.4	Diagram alur algoritme k-MPPTI	37
Gambar 3.5	Lini masa data produk dan pelanggan	39
Gambar 3.6	<i>Event</i> dalam lini masa data produk dan pelanggan	39
Gambar 3.7	Diagram alur proses <i>product insertion</i>	42
Gambar 3.8	Diagram alur proses <i>product deletion</i>	44
Gambar 3.9	Diagram alur proses <i>customer insertion</i>	45
Gambar 3.10	Diagram alur proses <i>customer deletion</i>	46
Gambar 3.11	Diagram alur komputasi <i>reverse skyline</i>	48
Gambar 3.12	Diagram alur komputasi <i>initial dynamic skyline</i>	50
Gambar 3.13	Diagram alur komputasi <i>dynamic skyline - product insertion</i>	51
Gambar 3.14	Diagram alur komputasi <i>dynamic skyline - product deletion</i>	52
Gambar 3.15	Pemrosesan paralel	54
Gambar 3.16	Ilustrasi interval waktu pencarian	56
Gambar 3.17	Perhitungan kontribusi pasar pada <i>Pandora Box</i> berdasarkan interval waktu pencarian	56
Gambar 3.18	Perancangan arsitektur aplikasi	58
Gambar 4.1	Algoritme <i>precomputing</i> (bagian 1)	60
Gambar 4.2	Algoritme <i>precomputing</i> (bagian 2)	61

Gambar 4.3 Algoritme <i>precomputing</i> (bagian 3)	62
Gambar 4.4 Kelas <i>EventQueue</i>	63
Gambar 4.5 Kelas <i>PandoraBox</i> (bagian 1)	64
Gambar 4.6 Kelas <i>ReverseSkyline</i> (bagian 1)	65
Gambar 4.7 Kelas <i>ReverseSkyline</i> (bagian 2)	66
Gambar 4.8 Kelas <i>ReverseSkyline</i> (bagian 3)	67
Gambar 4.9 Kelas <i>DynamicSkyline</i> (bagian 1)	68
Gambar 4.10 Kelas <i>DynamicSkyline</i> (bagian 2)	69
Gambar 4.11 Fungsi cek dominasi	70
Gambar 4.12 Algoritme <i>query processing</i>	71
Gambar 4.13 Kelas <i>PandoraBox</i> (bagian 2)	71
Gambar 4.14 Implementasi halaman <i>Home</i>	72
Gambar 4.15 Implementasi halaman <i>Precompute</i>	73
Gambar 4.16 Implementasi halaman <i>Search</i>	73
Gambar 4.17 Implementasi halaman <i>Visualization</i> (informasi data)	74
Gambar 4.18 Implementasi halaman <i>Visualization</i> (pratinjau data)	75
Gambar 4.19 Implementasi halaman <i>Visualization</i> (visualisasi data)	75
Gambar 5.1 Hasil uji coba: mengunggah data dan memilih algoritme untuk <i>data precomputing</i>	84
Gambar 5.2 Hasil uji coba: proses <i>data precomputing</i> berhasil	84
Gambar 5.3 Hasil uji coba: melihat informasi data	85
Gambar 5.4 Hasil uji coba: melihat pratinjau data berupa tabel	85
Gambar 5.5 Hasil uji coba: melihat visualisasi data berupa lini masa	86
Gambar 5.6 Hasil uji coba: memasukkan kueri pencarian dan melihat hasil kueri	86
Gambar 5.7 Hasil uji coba: memilih <i>session</i>	87

Gambar 5.8 Grafik pengaruh jumlah data terhadap waktu komputasi algoritme pada data <i>independent</i> (IND)	88
Gambar 5.9 Grafik pengaruh jumlah data terhadap waktu komputasi algoritme pada data <i>anti-correlated</i> (ANT)	89
Gambar 5.10 Grafik pengaruh jumlah data terhadap penggunaan memori algoritme pada data <i>independent</i> (IND)	90
Gambar 5.11 Grafik pengaruh jumlah data terhadap penggunaan memori algoritme pada data <i>anti-correlated</i> (ANT)	91
Gambar 5.12 Grafik pengaruh jumlah dimensi data terhadap waktu komputasi algoritme pada data <i>independent</i> (IND)	92
Gambar 5.13 Grafik pengaruh jumlah dimensi data terhadap waktu komputasi algoritme pada data <i>anti-correlated</i> (ANT)	93
Gambar 5.14 Grafik pengaruh jumlah dimensi data terhadap penggunaan memori algoritme pada data <i>independent</i> (IND)	94
Gambar 5.15 Grafik pengaruh jumlah dimensi data terhadap penggunaan memori algoritme pada data <i>anti-correlated</i> (ANT)	95
Gambar 5.16 Pengujian akurasi hasil kueri pada data <i>independent</i> (IND)	97
Gambar 5.17 Pengujian akurasi hasil kueri pada data <i>anti-correlated</i> (ANT)	97
Gambar 5.18 Pengujian akurasi hasil kueri pada data <i>Forest Cover Type</i> (FC)	98

Halaman ini sengaja dikosongkan

DAFTAR KODE SUMBER

Kode Sumber 1.1	Kode sumber kelas <i>EventQueue</i>	105
Kode Sumber 1.2	Kode sumber kelas <i>PandoraBox</i>	105
Kode Sumber 1.3	Kode sumber kelas <i>ReverseSkyline</i> . . .	106
Kode Sumber 1.4	Kode sumber kelas <i>DynamicSkyline</i> . .	108
Kode Sumber 1.5	Kode sumber algoritme <i>Precompute</i> . .	110
Kode Sumber 1.6	Kode sumber algoritme <i>QueryProcessing</i>	116
Kode Sumber 1.7	Kode sumber <i>Flask Server</i>	117
Kode Sumber 1.8	Kode sumber <i>Flask Routes</i>	117
Kode Sumber 1.9	Kode sumber <i>template</i> halaman web . .	119
Kode Sumber 1.10	Kode sumber halaman web <i>Index</i> . . .	121
Kode Sumber 1.11	Kode sumber halaman web <i>Precompute</i>	122
Kode Sumber 1.12	Kode sumber halaman web <i>Search</i> . .	123
Kode Sumber 1.13	Kode sumber halaman web <i>Visualization</i>	124

Halaman ini sengaja dikosongkan

BAB I

PENDAHULUAN

Pada bab ini dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Pesatnya kemajuan ilmu pengetahuan dan teknologi di bidang analisis data telah mempengaruhi cara perusahaan dalam menjalankan bisnis, yaitu dengan mengumpulkan data-data penjualan, riset pasar, logistik, atau biaya transportasi, kemudian menggunakan untuk membuat keputusan bisnis yang lebih baik. Seorang analis dapat mengumpulkan data preferensi pelanggan terhadap fitur-fitur produk perusahaan tersebut dari data penjualan yang dimiliki. Selain itu, maraknya penggunaan situs web untuk menjual produk secara *online* juga memungkinkan analis mengumpulkan data preferensi pelanggan terhadap fitur produk perusahaan lain.

Dengan memanfaatkan data preferensi pelanggan, sebuah perusahaan dapat mendapatkan informasi yang dapat digunakan untuk membuat keputusan bisnis yang tepat. Misalnya, dengan mendapatkan informasi *k*-produk apa saja yang paling diminati oleh pelanggan beserta fitur-fiturnya, perusahaan dapat menentukan harga produk baru yang akan diluncurkan atau menentukan fitur apa yang hendak diunggulkan dari produk baru yang ingin dibuat.

Saat ini, sudah ada penelitian yang mengembangkan strategi pemilihan produk dengan melakukan pencarian k -produk yang paling banyak diminati oleh pelanggan. Dalam [1], Islam et al. memodelkannya sebagai kueri *k-Most Promising Products* (k -MPP) dan membuat kerangka kerja algoritme untuk memproses kueri tersebut. Komputasi k -MPP menggunakan dua tipe kueri *skyline*, yaitu *dynamic skyline* [2] dan *reverse skyline* [3]. Kueri *dynamic skyline* digunakan untuk mengambil data produk terbaik berdasarkan sudut pandang pelanggan, sedangkan kueri *reverse skyline* digunakan untuk mengambil data pelanggan potensial berdasarkan sudut pandang produk atau perusahaan.

Kelemahan dari strategi pemilihan produk k -MPP adalah tidak adanya pertimbangan variabel waktu dalam algoritme perhitungannya sehingga informasi yang didapatkan kurang valid dengan kondisi yang sebenarnya. Komputasi k -MPP juga tidak dapat memproses kueri berbasis interval waktu. Sebagai contoh, pertanyaan yang mungkin diajukan adalah “*k-produk apa saja yang paling banyak diminati oleh pelanggan pada bulan Februari hingga September?*”. Dalam hal ini, bulan Februari hingga September disebut dengan interval waktu kueri dan data yang berbasis interval waktu disebut dengan data *time series* atau serial waktu.

Sebagai ilustrasi, produk A adalah produk yang paling banyak diminati oleh pelanggan pada bulan Januari hingga Juni, namun posisinya diungguli oleh produk B yang lebih diminati pelanggan pada bulan Juli hingga September. Pada bulan Oktober, produk B tidak diproduksi lagi karena suatu alasan, sehingga produk A kembali diminati pelanggan.

Berdasarkan ilustrasi tersebut, produk yang paling unggul berdasarkan kueri k -MPP adalah produk B karena produk B pernah mengungguli produk A walaupun rentang waktu unggulnya lebih pendek daripada produk A. Hal ini terjadi karena komputasi k -MPP hanya mempertimbangkan skor kontribusi pasar yang dihitung dari

banyaknya jumlah pelanggan yang lebih menyukai produk tersebut daripada produk lainnya tanpa mempertimbangkan faktor durasi waktu.

Sedangkan jika berdasarkan kueri dengan interval waktu Januari hingga Juli maka produk yang paling unggul adalah produk A; jika berdasarkan kueri dengan interval waktu Juli hingga Agustus maka produk yang paling unggul adalah produk B; jika berdasarkan kueri dengan interval waktu Januari hingga Desember maka produk yang paling unggul adalah produk A karena rentang waktu unggulnya lebih lama daripada produk B.

Tugas Akhir ini bertujuan untuk menjawab permasalahan k-MPP berbasis interval waktu pada data multidimensi dengan serial waktu dengan memodelkan kueri k-MPPTI (*k*-Most Promising Products in Time Intervals) dan merancang kerangka kerja algoritme yang dapat memproses kueri tersebut. Ada tiga jenis algoritme pemrosesan yang dibuat dan dibandingkan: (a) k-MPPTI, yaitu algoritme yang mengadaptasi komputasi k-MPP asli (menggunakan dua tipe kueri *skyline*, yaitu *dynamic skyline* dan *reverse skyline*); (b) k-MPPTI NoRSL, yaitu algoritme yang hanya menggunakan kueri *dynamic skyline* saja; (c) k-MPPTI NoRSL-P, yaitu algoritme k-MPPTI NoRSL yang mengimplementasikan teknik komputasi paralel supaya pemrosesan data menjadi lebih cepat. Efektivitas dan efisiensi ketiga algoritme diuji menggunakan data asli dan sintetis.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana desain dan implementasi struktur data dan algoritme untuk menjawab kueri *k*-Most Promising Products (k-MPP) berbasis interval waktu pada data multidimensi

dengan serial waktu?

2. Bagaimana kinerja dari struktur data dan algoritme yang dibangun untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu?
3. Bagaimana strategi yang optimal untuk meningkatkan efisiensi komputasi *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu?

1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan sebagai berikut:

1. Struktur data dan algoritme dalam komputasi *k-Most Promising Products* (k-MPP) berbasis interval waktu hanya dapat menyimpan dan memproses nilai numerik.
2. Implementasi struktur data dan algoritme menggunakan bahasa pemrograman Python.
3. *Dataset* yang digunakan adalah data asli dan sintetis.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Merancang dan mengimplementasikan struktur data dan algoritme untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.
2. Mengevaluasi kinerja dari struktur data dan algoritme yang dibangun untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.

3. Mengimplementasikan strategi yang optimal untuk meningkatkan efisiensi komputasi *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.

1.5 Manfaat

Manfaat yang diharapkan dari penulisan Tugas Akhir ini adalah untuk mengetahui struktur data dan algoritme yang tepat untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktusecara optimal dan efisien. Selain itu, Tugas Akhir ini juga diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi karena algoritme ini dapat digunakan dalam berbagai hal, khususnya bagi perusahaan untuk membuat bisnisnya menjadi lebih baik dan tepat sasaran.

1.6 Metodologi

Metodologi yang digunakan dalam penggerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir

Tahap awal untuk memulai penggerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir yang berisi gagasan untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu. Proposal ini berisi tentang deskripsi pendahuluan dari Tugas Akhir yang akan dibuat, terdiri atas hal yang menjadi latar belakang diajukannya usulan Tugas Akhir, rumusan masalah yang diangkat, batasan masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir.

2. Studi literatur

Pada tahap ini dilakukan pencarian informasi dan literatur mengenai metode yang dapat digunakan dalam merancang dan mengimplementasikan struktur data dan algoritme untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu. Informasi-informasi tersebut bisa didapatkan dari buku, jurnal, maupun internet.

3. Analisis dan perancangan perangkat lunak

Pada tahap ini dilakukan analisis dan perancangan struktur data dan algoritme yang digunakan untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu berdasarkan literatur yang telah dipelajari.

4. Implementasi perangkat lunak

Pada tahap ini dilakukan implementasi atau realiasi dari hasil analisis dan perancangan struktur data dan algoritme yang telah dibuat ke dalam bentuk program.

5. Uji coba dan evaluasi

Pada tahap ini dilakukan uji coba dari struktur data dan algoritme yang telah diimplementasikan. Pengujian akan dilakukan dengan dua cara, yaitu:

(a) Pengujian waktu eksekusi (*runtime*)

Pengujian yang berfokus pada waktu eksekusi dari struktur data dan algoritme yang dibangun untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu.

(b) Pengujian penggunaan memori (*memory usage*)

Pengujian yang berfokus pada konsumsi memori dari struktur data dan algoritme yang dibangun untuk menjawab kueri *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan

serial waktu.

Setelah dilakukan uji coba, maka dilakukan evaluasi terhadap kinerja struktur data dan algoritme yang telah diimplementasikan, dengan harapan dapat diperbaiki ke depannya.

6. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan penyusunan buku Tugas Akhir yang berisi dokumentasi penggerjaan dan laporan hasil penggerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Berikut adalah sistematika penulisan buku Tugas Akhir:

1. BAB I: PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan Tugas Akhir.

2. BAB II: TINJAUAN PUSTAKA

Bab ini berisi dasar teori mengenai permasalahan dan algoritme penyelesaian yang digunakan dalam Tugas Akhir

3. BAB III: ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi analisis dan perancangan struktur data dan algoritme yang digunakan dalam penyelesaian permasalahan.

4. BAB IV: IMPLEMENTASI

Bab ini berisi implementasi berdasarkan analisis dan perancangan struktur data dan algortime yang telah dilakukan pada tahap analisis dan perancangan sistem.

5. BAB V: UJI COBA DAN EVALUASI

Bab ini berisi uji coba dan evaluasi dari hasil implementasi yang telah dilakukan pada tahap implementasi.

6. BAB VI: PENUTUP

Bab ini berisi kesimpulan dan saran yang didapat dari hasil uji coba dan evaluasi yang telah dilakukan.

BAB II

TINJAUAN PUSTAKA

Bab ini menjelaskan dasar teori yang digunakan dalam analisis, perancangan, dan implementasi struktur data dan algoritme untuk menjawab permasalahan *k-Most Promising Products* (*k*-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu yang diangkat dalam Tugas Akhir ini.

2.1 Daftar Notasi

Tabel 2.1 menunjukkan daftar notasi yang digunakan untuk memudahkan beberapa penjelasan pada bab ini berikut dengan deskripsinya.

Tabel 2.1 Daftar notasi (bagian 1)

Notasi	Deskripsi
P	<i>Dataset</i> produk
C	<i>Dataset</i> pelanggan (preferensi pelanggan)
D	$P \cup C$
ob	Sebuah objek data pada D
$ob_1 \prec ob_2$	Objek data ob_1 mendominasi ob_2
$ob_1 \prec_{ob_3} ob_2$	Objek data ob_1 mendominasi ob_2 berdasarkan ob_3
p	Sebuah produk dalam P , $p \in P$
c	Seorang pelanggan dalam C , $c \in C$
d	Jumlah dimensi pada D

Notasi	Deskripsi
i	Dimensi ke-1, ..., d
j	Timestamp ke-1, 2, ..., dst
O	<i>Orthant</i> atau daerah pada komputasi <i>reverse skyline</i>
m	<i>Midpoint</i> antar produk pada komputasi <i>reverse skyline</i>
$DSL(c)$	Hasil <i>dynamic skyline</i> dari pelanggan c
$RSL(p)$	Hasil <i>reverse skyline</i> dari produk p
$Pr(c, p P)$	Probabilitas produk p dibeli oleh pelanggan c
$E(C, p P)$	Kontribusi pasar p
$E(C, P' P)$	Kontribusi pasar subset P' dari P
$k - MPP$	<i>k</i> -Most Promising Products

2.2 Skyline

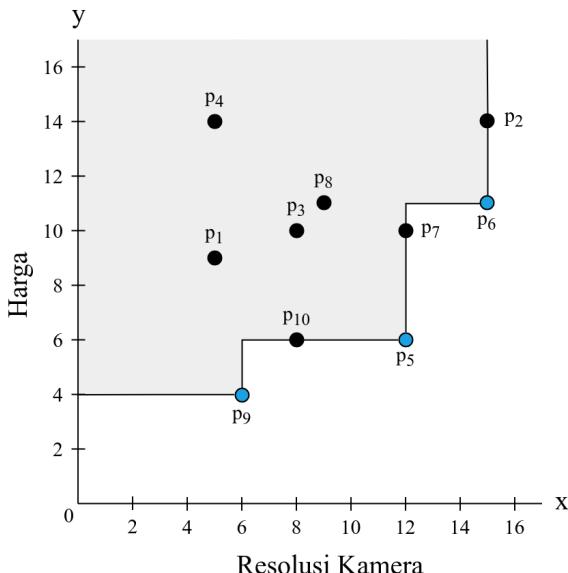
Komputasi *skyline* telah menarik perhatian yang cukup besar dari peneliti sejak diperkenalkan pada komunitas basis data [6], terutama mengenai metode progresif yang dapat mengembalikan hasil kueri dengan cepat tanpa perlu membaca keseluruhan data [2]. Operasi *skyline* digunakan untuk mencari data yang menarik dari suatu himpunan data, yaitu data yang tidak didominasi oleh data lain. *Skyline* didefinisikan sebagai titik-titik yang tidak didominasi oleh titik lain [6].

Diberikan *dataset* produk P yang setiap datanya direpresentasikan sebagai titik d -dimensi. Sebuah titik p_1 dikatakan mendominasi titik lain p_2 , dinotasikan dengan $p_1 \prec p_2$, jika nilai p_1 tidak lebih besar dari p_2 pada semua dimensi dan ada nilai p_1 yang lebih kecil dari p_2 minimal pada satu dimensi. Secara

matematika, relasi $p_1 \prec p_2$ dapat terbentuk jika dan hanya jika memenuhi syarat dominansi pada Persamaan 2.1.

- $$\begin{aligned} \text{(a)} \quad & p_1^i \leq p_2^i, \forall i \in [1, \dots, d] \\ \text{(b)} \quad & p_1^i < p_2^i, \exists i \in [1, \dots, d] \end{aligned} \quad (2.1)$$

Sebagai ilustrasi, seseorang ingin mencari produk *smartphone* terbaik, yaitu *smartphone* yang memiliki harga termurah dan memiliki resolusi kamera terbesar. Menggunakan data produk P pada Tabel 2.2, terdapat data produk *smartphone* yang memiliki atribut resolusi kamera (dim_1) dan harga (dim_2). Setiap datanya direpresentasikan sebagai titik pada bidang dua dimensi, yakni sumbu x adalah resolusi kamera dan sumbu y adalah harga *smartphone*.



Gambar 2.1 Titik *skyline* dari data produk pada Tabel 2.2

Berdasarkan Gambar 2.1, produk *smartphone* yang terbaik adalah p_5 , p_6 , dan p_9 karena tidak ada titik yang lebih baik dari titik-titik tersebut pada semua dimensi. Produk p_5 , p_6 , dan p_9 disebut sebagai titik *skyline* atau *skyline point*. Hubungan antar *skyline point* adalah "saling mendominasi" karena masing-masing data tersebut memiliki minimal satu atribut yang lebih unggul dibanding data lainnya.

Produk p_{10} tidak dapat menjadi *skyline* karena didominasi oleh produk p_5 pada atribut resolusi kamera (sumbu x). Sebagai ilustrasi, jika ada dua atau lebih produk *smartphone* dengan harga yang sama maka orang akan lebih memilih *smartphone* yang memiliki resolusi kamera paling besar. Begitu pula produk p_7 yang didominasi p_5 dan produk p_2 yang didominasi p_6 pada atribut harga (sumbu y). Sebagai ilustrasi, jika ada dua atau lebih produk *smartphone* dengan resolusi kamera yang sama maka orang akan lebih memilih *smartphone* yang memiliki harga paling murah.

Saat ini, komputasi *skyline* telah banyak digunakan sebagai operator pengambil keputusan multikriteria dan perencanaan bisnis [7]. Ada beberapa pengembangan dari komputasi *skyline* seperti *dynamic skyline* dan *reverse skyline*.

2.3 Dominansi Dinamis

Para ahli menyebut *original skyline* sebagai *static skyline* [7] karena sifat dominansinya yang statis. Jika diberikan *dataset* yang sama maka hasil *skyline*-nya selalu sama. Dalam pengembangannya, hasil *skyline* dapat berubah sesuai dengan posisi titik kueri (dominansi dinamis). Jika diberikan *dataset* yang sama, namun titik kueri berbeda, maka hasil *skyline*-nya pun berbeda.

Diberikan *dataset* produk P dan pelanggan (preferensi pelanggan) C yang setiap datanya direpresentasikan sebagai objek data d -dimensi dan hanya dapat menyimpan nilai numerik pada setiap dimensinya. Data produk dan pelanggan pada dimensi ke- i dinotasikan sebagai p^i dan c^i , $i \leq d$. Untuk menggambarkan objek data secara umum digunakan notasi ob .

Suatu objek data ob_1 dikatakan mendominasi objek data ob_2 secara dinamis berdasarkan objek data ob_3 , dinotasikan dengan $ob_1 \prec_{ob_3} ob_2$, jika nilai ob_1 dekat dengan ob_3 pada semua dimensi dan ada nilai ob_1 yang lebih dekat dengan ob_3 dibandingkan nilai ob_2 dengan ob_3 minimal pada satu dimensi. Secara matematika, relasi $ob_1 \prec_{ob_3} ob_2$ terbentuk jika dan hanya jika memenuhi syarat dominansi dinamis pada Persamaan 2.2.

$$\begin{aligned} (a) \quad & |ob_3^i - ob_1^i| \leq |ob_3^i - ob_2^i|, \forall i \in [1, \dots, d] \\ (b) \quad & |ob_3^i - ob_1^i| < |ob_3^i - ob_2^i|, \exists i \in [1, \dots, d] \end{aligned} \quad (2.2)$$

Pada Tabel 2.2, diberikan contoh *dataset* produk dan preferensi pelanggan. Berdasarkan preferensi pelanggan c_1 , produk p_1 dikatakan mendominasi produk p_2 , dinotasikan dengan $p_1 \prec_{c_1} p_2$, karena memenuhi kedua syarat dominansi dinamis yakni (a) $|c_1^1 - p_1^1| = |5 - 5| = 0 \leq |c_1^1 - p_2^1| = |5 - 15| = 10$ dan (b) $|c_1^2 - p_1^2| = |2 - 9| = 7 < |c_1^2 - p_2^2| = |2 - 14| = 12$. Sebaliknya, jika berdasarkan preferensi pelanggan c_6 , maka produk p_2 -lah yang mendominasi p_1 , dinotasikan dengan $p_2 \prec_{c_6} p_1$, karena (a) $|c_6^1 - p_2^1| = |12 - 15| = 3 \leq |c_6^1 - p_1^1| = |12 - 5| = 7$ dan (b) $|c_6^2 - p_2^2| = |14 - 14| = 0 < |c_6^2 - p_1^2| = |14 - 9| = 5$. Dalam hal ini, preferensi pelanggan disebut dengan titik kueri karena dapat mempengaruhi sifat dominansi antar produk.

Mengambil contoh lain, produk p_1 tidak mendominasi p_2 berdasarkan pelanggan c_3 karena ada salah satu syarat dominansi dinamis yang tidak terpenuhi, (a) $|c_3^1 - p_1^1| = |15 - 5| = 10 \not\leq$

Tabel 2.2 Contoh *dataset*
(a) produk P dan (b) preferensi pelanggan C

(a)	(b)				
id	dim1	dim2	id	dim1	dim2
p_1	5	9	c_1	5	2
p_2	15	14	c_2	8	10
p_3	8	10	c_3	15	10
p_4	5	14	c_4	9	7
p_5	12	6	c_5	10	12
p_6	15	11	c_6	12	14
p_7	12	10	c_7	7	13
p_8	9	11	c_8	15	8
p_9	6	4	c_9	5	5
p_{10}	8	6	c_{10}	10	5

$|c_3^1 - p_2^1| = |15 - 15| = 0$ dan (b) $|c_3^2 - p_1^2| = |10 - 9| = 1 < |c_3^2 - p_2^2| = |10 - 14| = 4$. Produk p_1 dan p_2 dikatakan saling mendominasi berdasarkan pelanggan c_2 .

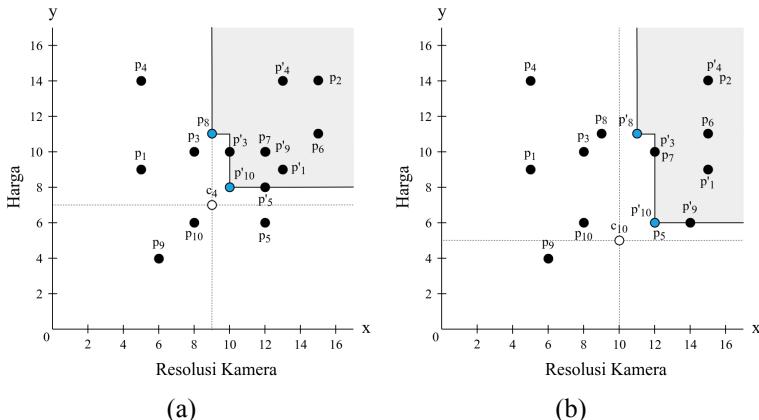
2.4 Dynamic Skyline

Kueri *dynamic skyline* dalam komputasi k -MPP digunakan untuk mencari produk terbaik dari sudut pandang pelanggan [1], sehingga yang menjadi titik kueri adalah pelanggan. *Dynamic skyline* [2] dari seorang pelanggan $c_1 \in C$, dinotasikan dengan $DSL(c_1)$, berisi semua produk $p_1 \in P$ yang tidak didominasi oleh produk lain $p_2 \in P$ berdasarkan preferensi pelanggan c_1 , $p_2 \not\prec_{c_1} p_1$.

Dynamic skyline dapat dihitung menggunakan algoritme komputasi *skyline* tradisional [6], yaitu mentransformasikan semua titik $p \in P$ ke ruang data baru dengan menganggap titik kueri c

sebagai titik asal dan jarak absolut titik p ke c digunakan sebagai fungsi pemetaan seperti yang ditunjukkan pada Gambar 2.2. Fungsi pemetaan f^i didefinisikan sebagai $f^i(p^i) = |c^i - p^i|$.

Menggunakan *dataset* pada Tabel 2.2, *dynamic skyline* dari pelanggan c_4 adalah $DSL(c_4) = \{p_8, p_{10}\}$, karena produk tersebut tidak didominasi oleh produk lain berdasarkan preferensi pelanggan c_4 . Berbeda halnya dengan c_{10} yang memiliki hasil *dynamic skyline* $DSL(c_{10}) = \{p_5, p_8, p_{10}\}$.



Gambar 2.2 (a) Komputasi *dynamic skyline* dari pelanggan c_4 dan (b) *dynamic skyline* dari pelanggan c_{10}

2.5 Reverse Skyline

Dalam komputasi k -MPP, kueri *reverse skyline* digunakan untuk mencari pelanggan potensial dari sudut pandang produsen [1], sehingga yang menjadi titik kueri adalah produk. *Reverse skyline* [3] dari sebuah produk $p_1 \in P$, dinotasikan dengan $RSL(p_1)$, berisi semua pelanggan $c \in C$ yang memiliki p_1 pada hasil *dynamic skyline*-nya.

Ada beberapa tahapan yang harus dilakukan dalam komputasi *reverse skyline* [1]. Pertama, menentukan *orthant* dari produk, dinotasikan dengan O . Setiap produk p memiliki 2^d *orthant* pada data d -dimensi. Kedua, menghitung *midpoint* atau titik tengah antara produk kueri dan produk lainnya, misalnya p_1 (sebagai titik kueri) dan p_2 , dihitung menggunakan rumus berikut:

$$m_2^i = \frac{(p_1^i + p_2^i)}{2} \quad (2.3)$$

Kemudian menentukan *midpoint skyline* (juga dikenal sebagai *mid-skyline* [8]) pada setiap *orthant*.

Langkah ketiga, mengecek apakah pelanggan $c \in C$ didominasi oleh *midpoint skyline* m berdasarkan produk p_1 atau tidak. Pelanggan c dikatakan didominasi oleh *midpoint skyline* m jika dan hanya jika:

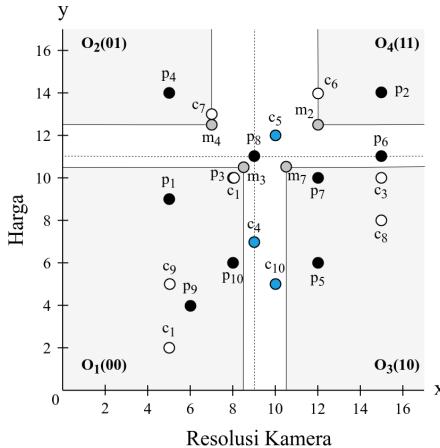
- (a) $|p_1^i - m^i| \leq |p_1^i - c^i|, \forall i \in [1, \dots, d]$
 - (b) $|p_1^i - m^i| < |p_1^i - c^i|, \exists i \in [1, \dots, d]$
- (2.4)

Apabila c tidak didominasi oleh *midpoint skyline* m berdasarkan produk p_1 , maka c menjadi hasil dari *reverse skyline* p_1 , dinotasikan dengan $RSL(p_1)$. Untuk lebih jelasnya, komputasi *reverse skyline* ditunjukkan pada Gambar 2.3.

Sebagai contoh, berdasarkan *dataset* yang diberikan pada Tabel 2.2, *reverse skyline* dari produk p_8 adalah pelanggan c_4 , c_5 , dan c_{10} , dinotasikan dengan $RSL(p_8) = \{c_4, c_5, c_{10}\}$ karena masing-masing pelanggan tersebut memiliki p_8 pada hasil *dynamic skyline*-nya.

2.6 Kueri *k*-Most Promising Products (*k*-MPP)

Kueri *k*-Most Promising Products (*k*-MPP) adalah sebuah strategi pemilihan produk yang dikenalkan oleh Islam dan Liu



Gambar 2.3 Komputasi *reverse skyline* dari produk p_8

dalam Tugas Akhirnya [1].

2.6.1 Uniform Product Adoption (UPA)

Uniform Product Adoption (UPA) mengasumsikan bahwa semua produk $p \in P$ yang muncul pada hasil *dynamic skyline* pelanggan $c \in C$ akan saling berkompetisi satu sama lain untuk menarik pelanggan c , sehingga produk-produk tersebut memiliki probabilitas yang sama untuk dibeli oleh pelanggan c .

Probabilitas produk p dibeli oleh pelanggan c , dinotasikan dengan $Pr(c, p|P)$ dapat dijelaskan oleh persamaan berikut:

$$Pr(c, p|P) = \begin{cases} \frac{1}{|DSL(c)|} & \text{if } p \in DSL(c) \\ 0 & \text{otherwise} \end{cases} \quad (2.5)$$

Berdasarkan Persamaan 2.5, dapat dipastikan bahwa setiap produk yang muncul dalam $DSL(c)$ memiliki kesempatan yang

sama untuk dipilih oleh pelanggan c . Sebaliknya, produk yang tidak muncul dalam $DSL(c)$ tidak memiliki kesempatan sama sekali untuk dipilih oleh c .

Sebagai contoh menggunakan *dataset* pada Tabel 2.2, probabilitas produk p_8 dibeli oleh pelanggan c_4 adalah $Pr(c_4, p_8|P) = \frac{1}{|DSL(c_4)|} = \frac{1}{2}$, sedangkan probabilitas produk p_8 dibeli oleh pelanggan c_{10} adalah $\frac{1}{|DSL(c_{10})|} = \frac{1}{3}$.

2.6.1.1 Market Contribution

Market contribution atau kontribusi pasar sebuah produk $p \in P$ diukur dari total jumlah pelanggan yang mungkin lebih memilih membeli produk p dibandingkan produk lain p' .

Asumsinya, jika seorang pelanggan memiliki dua produk atau lebih dalam hasil *dynamic skyline*-nya, maka ia akan memberikan bobot yang sama pada produk-produk tersebut sebagaimana yang sudah dijelaskan pada Persamaan 2.5. Sehingga, kontribusi pasar sebuah produk dihitung dari hasil akumulasi bobot yang didapatkan dari semua pelanggan $c \in C$.

Kontribusi pasar produk p , dinotasikan dengan $E(C, p|P)$, diperoleh dengan mengakumulasikan probabilitas produk dari setiap pelanggan $c \in C$, sebagai berikut:

$$E(C, p|P) = \sum_{\forall c \in C} Pr(c, p|P) \quad (2.6)$$

Karena probabilitas produk p dipilih oleh pelanggan yang tidak memiliki p pada hasil *dynamic skyline*-nya adalah nol (pada Persamaan 2.5), maka kita hanya perlu mengakumulasikan probabilitas produk dari setiap pelanggan c pada hasil $RSL(p)$. Sehingga, Persamaan 2.6 dapat disederhanakan menjadi:

$$E(C, p|P) = \sum_{\forall c \in RSL(p)} Pr(c, p|P) \quad (2.7)$$

Sebagai contoh menggunakan *dataset* pada Tabel 2.2, kontribusi pasar dari produk p_8 adalah $E(C, p_8|P) = Pr(c_4, p_8|P) + Pr(c_5, p_8|P) + Pr(c_{10}, p_8|P) = \frac{1}{2} + 1 + \frac{1}{3} = \frac{11}{6}$ atau 1.833.

Perhitungan kontribusi pasar juga dapat dilakukan pada sekumpulan produk atau *subset* produk P' , dinotasikan dengan $E(C, P'|P)$, yang dijelaskan pada Persamaan 2.8.

$$E(C, P'|P) = \sum_{\forall p \in P'} E(C, p|P) \quad (2.8)$$

2.6.2 Strategi Pemilihan Produk

Diberikan *dataset* produk P , *dataset* preferensi pelanggan C , dan bilangan bulat positif k yang lebih kecil dari $|P|$. Kueri *k-Most Promising Products* (*k*-MPP), dinotasikan oleh Persamaan 2.9, akan memilih *subset* k produk P' dari P yang memiliki kontribusi pasar lebih besar dibandingkan dengan *subset* k produk P'' dari P yang lain [1].

$$k - MPP(P, C, k) \quad (2.9)$$

Jika merangkum semua penjelasan di atas, langkah-langkah yang harus dilakukan untuk memproses kueri *k*-MPP adalah: (1) menghitung *reverse skyline* dari setiap produk $p \in P$, (2) menghitung *dynamic Skyline* dari setiap pelanggan $c \in RSL(p)$, dan (3) memilih k produk dari P yang memiliki kontribusi pasar terbesar.

2.7 Data

Data merupakan elemen yang esensial dalam sebuah sistem informasi. Data adalah informasi faktual (seperti pengukuran atau statistik) yang digunakan sebagai dasar untuk analisis, diskusi, maupun perhitungan [10].

Meski begitu, data mentah tidaklah berarti dan harus diproses terlebih dahulu supaya menghasilkan informasi yang bermanfaat. Sehingga, dibutuhkanlah sebuah algoritme pemrosesan data yang menerima data sebagai *input*, kemudian memprosesnya menjadi informasi tertentu sesuai dengan kebutuhan pengguna dan mengeluarkannya sebagai *output*.

2.7.1 Data Multidimensi

Model data multidimensi adalah sebuah cara pandang yang melihat data dari berbagai sudut pandang atau dimensi. Model data ini memiliki struktur yang disesuaikan untuk mengoptimalkan analisis berdasarkan data dari *relational database* dan diolah sehingga informasi dapat dikategorikan. Model data multidimensi merupakan variasi dari model relasional yang menggunakan struktur multidimensi untuk menyusun data dan menjelaskan relasi antar data.

Struktur multidimensi merepresentasikan dimensi-dimensi data dalam bentuk kubus. Jika sebuah data multidimensi memiliki lebih dari tiga dimensi, maka disebut dengan *hypercube* [5]. Dalam implementasinya, data multidimensi disajikan dalam bentuk *array* multidimensi yang masing-masing nilai dalam selnya dapat diakses menggunakan sebuah indeks.

Data multidimensi banyak digunakan untuk analisis. Selama beberapa tahun terakhir, konsep data multidimensi telah menjadi hal yang fundamental dalam sistem pengambil keputusan, seperti sistem *data warehouse* [5].

2.7.2 Data Serial Waktu

Data *time series* atau serial waktu adalah nilai-nilai suatu variabel yang berurutan menurut waktu. Data *time series* memiliki nilai dan *timestamp*, sehingga data diurutkan berdasarkan waktu

atau *timestamp*-nya.

Tabel 2.3 Contoh data *time series*

id	timestamp				
	1	2	3	4	5
s_1	8	2	5	10	12
s_2	14	4	10	7	8
s_3	15	6	11	7	3
s_4	3	8	12	9	13
s_5	15	9	10	2	7

Pada Tabel 2.3, diberikan contoh sebuah data *time series* S . Supaya sederhana, kita asumsikan bahwa *timestamp* adalah bilangan bulat positif. Nilai $s_1 \in S$ pada *timestamp* j dinotasikan sebagai $s_1[j]$, sehingga *time series* s_1 jika ditulis secara berurutan menjadi $s_1[1], s_1[2], \dots$, dan seterusnya [4].

2.7.3 Data Multidimensi dengan Serial Waktu

Data multidimensi dengan serial waktu adalah data *multi-attribute* yang memiliki *timestamp* dan berurutan menurut waktu sebagaimana contoh *dataset* produk P dan preferensi pelanggan C yang ditunjukkan pada Tabel 2.4. Pada tabel tersebut, *timestamp* ditulis sebagai interval waktu yang dinotasikan dengan $[t_i : t_e]$, dengan asumsi bahwa nilai masing-masing atribut konstan setiap waktu.

Tabel 2.4 Contoh data multidimensi dengan serial waktu
 (a) produk P dan (b) preferensi pelanggan C

ID	Timestamp		Nilai	
	t_i	t_e	d_1	d_2
p_1	2	10	6	3
p_2	6	13	4	12
p_3	9	15	6	15
p_4	4	9	9	5
p_5	5	15	12	10

ID	Timestamp		Nilai	
	t_i	t_e	d_1	d_2
c_1	1	8	2	8
c_2	4	14	4	10
c_3	10	15	6	11
c_4	3	8	8	12
c_5	5	15	9	10

Jenis data yang diproses pada Tugas Akhir ini adalah data multidimensi dengan serial waktu sebagaimana yang ditunjukkan pada Tabel 2.4; terdiri dari dua data, yaitu data produk P dan pelanggan C yang memiliki atribut lebih dari satu. Sebagai ilustrasi dari data ini, setiap produk $p \in P$ memiliki waktu kapan ia pertama kali diproduksi dan tidak diproduksi lagi, sedangkan setiap pelanggan $c \in C$ memiliki waktu kapan ia lahir dan meninggal dunia.

2.8 Python

Python adalah bahasa pemrograman tingkat tinggi, *interpreted*, dan berorientasi objek yang didukung oleh struktur data *built-in* tingkat tinggi dan semantik yang dinamis [11]. Python dikembangkan oleh Guido van Rossum pada akhir 1980-an dan dikelola oleh *Python Software Foundation*. Saat ini, Python sudah tersedia dalam dua versi, yakni 2.x dan 3.x.

Kelebihan bahasa pemrograman Python adalah pada keterbacaannya karena memiliki sintaksis yang sederhana, sehingga dapat mengurangi biaya pemeliharaan (*maintenance*).

Python mendukung banyak modul dan *package*, serta memiliki banyak *standard library* yang didistribusikan secara gratis. Selain itu, karena Python adalah bahasa *interpreted*, Python tidak memakan biaya untuk kompilasi sehingga proses pengubahan, pengujian, dan debug menjadi lebih cepat.

Melakukan debug pada program Python sangatlah mudah karena tidak akan mengakibatkan *segmentation fault*. Sebagai gantinya, ia akan menimbulkan *Exception* apabila menemukan suatu *error* atau kesalahan. Ketika program tidak menangkap *Exception*, maka Python akan menampilkan *stack trace* yang dapat digunakan untuk menganalisis dan memperbaiki kesalahan yang terjadi [11].

Dalam Tugas Akhir ini, bahasa pemrograman Python digunakan untuk mengimplementasikan struktur data dan algoritme pada sistem perangkat lunak yang akan dibangun. Python yang digunakan adalah versi 3.7.3.

2.9 Flask

Flask adalah kerangka kerja web berbahasa Python yang sederhana, ringan, dan mudah dikembangkan, sehingga Flask kerap disebut dengan *microframework*. Flask dibangun dari dua pustaka utama, yaitu Jinja *template engine* dan Werkzeug WSGI *toolkit*, serta memiliki lisensi BSD. Saat ini, Flask dikembangkan dan dikelola oleh *Pallets team* dan kontributor komunitas [12].

Dalam Tugas Akhir ini, kerangka kerja Flask digunakan untuk mengimplementasikan aplikasi web dan layanan *web server* yang digunakan pada Tugas Akhir ini karena ringan dan lebih mudah digunakan dibandingkan dengan *framework* Python Django. Selain itu, Flask juga memiliki banyak dokumentasi dan tutorial yang dapat diikuti. Flask yang digunakan adalah versi 1.0.2.

2.10 Vis.js

Vis.js adalah sebuah pustaka visualisasi berbasis web yang mudah digunakan, dapat menangani jumlah data yang besar secara dinamis, serta memungkinkan manipulasi dan interaksi dengan data. Pustaka ini terdiri dari komponen *DataSet*, *Timeline*, *Network*, *Graph2d*, dan *Graph3d*. Pustaka yang dikembangkan oleh Almende B.V. ini berjalan dengan baik di sebagian besar peramban, seperti Chrome, Firefox, Opera, Safari, IE9 +, dan beberapa peramban seluler [13].

Dalam Tugas Akhir ini, pustaka Vis.js digunakan untuk memvisualisasikan data dalam bentuk lini masa atau *timeline* pada sistem perangkat lunak yang akan dibangun. Vis.js yang digunakan adalah versi 4.21.0.

Halaman ini sengaja dikosongkan

BAB III

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini dijelaskan mengenai analisis dan perancangan sistem perangkat lunak yang akan dibangun, meliputi struktur data, algoritme, dan arsitektur aplikasi.

3.1 Daftar Notasi

Tabel 3.1 menunjukkan daftar notasi yang digunakan dalam bab ini beserta deskripsinya.

Tabel 3.1 Daftar notasi (bagian 2)

Notasi	Deskripsi
P	<i>Dataset</i> produk
C	<i>Dataset</i> pelanggan (preferensi pelanggan)
D	$P \cup C$
E	Himpunan <i>event</i>
p	Sebuah produk dalam P , $p \in P$
c	Seorang pelanggan dalam C , $c \in C$
e	Sebuah <i>event</i> dalam E , $e \in E$
p_{in}	Produk masuk
p_{out}	Produk keluar
c_{in}	Pelanggan masuk
c_{out}	Pelanggan keluar

Notasi	Deskripsi
PA	Himpunan produk yang sedang aktif (di dalam lini masa)
CA	Himpunan pelanggan yang sedang aktif (di dalam lini masa)
d	Jumlah dimensi pada D
i	Dimensi ke-1, ..., d
j	Timestamp ke-1, 2, ..., dst
$diff$	Selisih nilai
O	<i>Orthant</i>
m	<i>Midpoint</i> antar produk
$DSL(c)$	Hasil <i>dynamic skyline</i> dari pelanggan c
$RSL(p)$	Hasil <i>reverse skyline</i> dari produk p
$MSL(o)$	Hasil <i>midpoint skyline</i> dari orthant o
$Pr_t(c, p PA)$	Probabilitas produk p dibeli oleh pelanggan $c \in CA$ pada waktu t
k	Jumlah data
$[t_i : t_e]$	Interval waktu
$E_{[t_i:t_e]}(CA, p PA)$	Kontribusi pasar p dalam interval waktu $[t_i : t_e]$
$k - MPPTI$	<i>k-Most Promising Products in Time Intervals</i>
$PBox$	<i>Pandora Box</i>

3.2 Analisis Sistem

Analisis sistem dijelaskan dalam empat bagian, yakni analisis permasalahan, deskripsi umum sistem, fungsi sistem, dan analisis kebutuhan fungsional.

3.2.1 Analisis Permasalahan

Permasalahan yang ingin diselesaikan pada Tugas Akhir ini adalah bagaimana menjawab kueri *k-Most Promising Products* berbasis interval waktu (*k-MPPTI*). Interval waktu, dinotasikan dengan $[t_i : t_e](t_i \leq t_e)$, digunakan untuk menentukan rentang waktu pencarian.

Permasalahan ini tidak dapat langsung diselesaikan menggunakan metode dan algoritme yang sudah ada [1]. Sehingga, diperlukan pendekatan lain yang akan dijelaskan pada bagian perancangan sistem.

3.2.2 Deskripsi Umum Sistem

Secara umum, sistem yang akan dibangun adalah sebuah sistem berbasis web yang dapat membantu pengguna untuk memilih *k*-produk yang paling menjanjikan. Dikatakan ”menjanjikan” jika produk tersebut memiliki kontribusi pasar yang besar.

Sistem ini memiliki dua proses utama, yaitu (1) *data precomputing* untuk menghitung kontribusi pasar masing-masing produk dan (2) proses utama (selanjutnya akan disebut dengan *query processing*) untuk memproses dan menampilkan hasil kueri pencarian yang dimasukkan oleh pengguna.

Sistem ini dibangun menggunakan arsitektur *client-server*. Aplikasi *client* didesain berbasis web dengan memanfaatkan Flask *microframework*, HTML, CSS, dan JavaScript. Selain itu, Flask juga digunakan sebagai *web server*.

3.2.3 Fungsi Sistem

Sistem yang akan dibangun memiliki beberapa fungsi utama sebagai berikut:

1. Dapat menerima masukan data berupa file dari pengguna
2. Dapat menampilkan informasi dan pratinjau data yang dimasukkan oleh pengguna
3. Dapat menampilkan visualisasi data
4. Dapat melakukan proses *data precomputing* menggunakan algoritme yang dipilih oleh pengguna
5. Dapat menerima masukan kueri pencarian
6. Dapat memproses kueri pencarian
7. Dapat menampilkan hasil kueri

3.2.4 Analisis Kebutuhan Fungsional

Sistem yang dibuat harus mampu memenuhi beberapa fungsi utama yang telah dijelaskan pada sub-bagian sebelumnya. Fungsi-fungsi ini merupakan hasil dari analisis kebutuhan fungsional dari pengguna yang dijelaskan pada Tabel 3.2.

Tabel 3.2 Kebutuhan fungsional

Kode	Deskripsi Kebutuhan
F-001	Mengunggah data
F-002	Melihat informasi dan pratinjau data
F-003	Melihat visualisasi data
F-004	Memilih algoritme yang digunakan untuk <i>data precomputing</i>
F-005	Memasukkan kueri pencarian
F-006	Melihat hasil kueri

Penjelasan rinci dari masing-masing kebutuhan fungsional pada tabel 3.2 dijelaskan sebagai berikut:

1. Mengunggah data

Pengguna dapat mengunggah data produk dan preferensi pelanggan dalam bentuk file berekstensi csv.

2. Melihat informasi dan pratinjau data

Pengguna dapat melihat informasi dan pratinjau dari data yang dimasukkan berupa tabel sebanyak dua puluh baris. Informasi yang ditampilkan antara lain jumlah baris, jumlah kolom, dan nama kolom.

3. Melihat visualisasi data

Pengguna juga dapat melihat visualisasi dari data yang dimasukkan berupa lini masa sederhana.

4. Memilih algoritme yang digunakan untuk *data precomputing*

Pengguna dapat memilih algoritme yang akan digunakan untuk *data precomputing*, yaitu algoritme k-MPPTI dan *Brute Force*.

5. Memasukkan kueri pencarian

Pengguna dapat memasukkan kueri pencarian berupa jumlah produk (k) dan interval waktu.

6. Melihat hasil kueri

Pengguna dapat melihat hasil kueri pencarian berupa k -produk dengan jumlah kontribusi pasar terbesar beserta skor kontribusi pasar-nya.

3.3 Perancangan Sistem

Perancangan sistem akan dibagi menjadi empat bagian, yakni struktur data, algoritme utama, algoritme pembanding menggunakan metode *brute force*, dan arsitektur aplikasi.

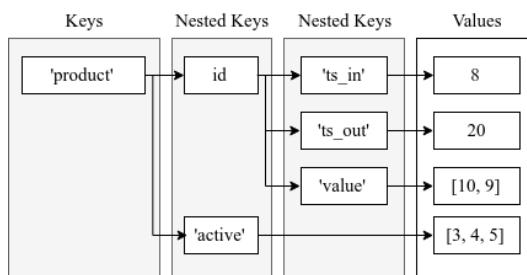
3.3.1 Struktur Data

Struktur data adalah suatu cara untuk menyimpan, menyusun, mengelompokkan, dan merepresentasikan suatu data. Ada tiga struktur data utama yang digunakan dalam komputasi k-MPPTI, yaitu *Data Storage*, *Event Queue*, dan *Pandora Box*.

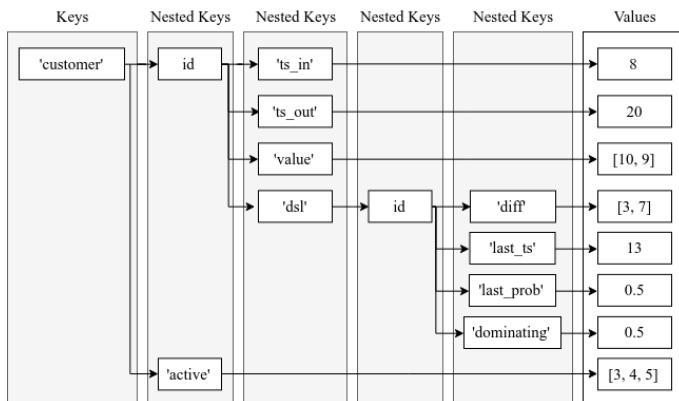
3.3.1.1 Data Storage

Data Storage adalah sebuah struktur data *dictionary* yang digunakan untuk menyimpan data produk dan pelanggan. Struktur data *dictionary* lebih efisien untuk pencarian data karena menggunakan konsep *key-value pairs*, berbeda dengan struktur data *list* atau *array* yang menggunakan indeks untuk mengakses nilai suatu data.

Struktur data *dictionary* yang digunakan berbentuk *nested dictionary* yang terdiri dari dua *key* utama, yaitu '*product*' yang menyimpan data produk dan '*customer*' yang menyimpan data pelanggan. Struktur *nested key* masing-masing data dijelaskan pada Gambar 3.1 dan 3.2 dan deskripsinya dijelaskan pada Tabel 3.3.



Gambar 3.1 Struktur data *dictionary* produk

Gambar 3.2 Struktur data *dictionary* pelangganTabel 3.3 *Key* dari *DataStorage*

Key	Deskripsi
'product'	Menyimpan data produk
'customer'	Menyimpan data pelanggan
'id'	ID data produk atau pelanggan dijadikan sebagai <i>key</i>
'active'	Menyimpan ID data produk atau pelanggan yang sedang aktif dalam bentuk <i>array</i>
'ts_in'	Menyimpan <i>timestamp</i> atau waktu masuk
'ts_out'	Menyimpan <i>timestamp</i> atau waktu keluar
'value'	Menyimpan nilai data produk atau pelanggan pada semua dimensi dalam bentuk <i>array</i>

Key	Deskripsi
'dsl'	Menyimpan hasil <i>dynamic skyline</i> dalam bentuk <i>dictionary</i> dengan <i>id</i> produk sebagai <i>key</i>
'diff'	Menyimpan selisih antara nilai data produk dan pelanggan pada masing-masing dimensi
'last_ts'	Menyimpan <i>timestamp</i> terakhir saat diperbarui ke <i>Pandora Box</i>
'last_prob'	Menyimpan probabilitas terakhir saat diperbarui ke <i>Pandora Box</i>
'dominating'	Menyimpan ID produk lain yang pernah didominasi

3.3.1.2 Event Queue

Event adalah titik-titik tempat terjadinya perubahan di dalam himpunan data, yaitu jika ada data yang masuk atau keluar. Ada empat jenis *event* yang terjadi dalam Tugas Akhir ini, yaitu:

1. Data Produk masuk
2. Data Produk keluar
3. Data Pelanggan masuk
4. Data Pelanggan keluar

Produk dan pelanggan disebut dengan pemilik *event*, sedangkan masuk dan keluar disebut dengan aksi *event*.

Event Queue adalah sebuah struktur data *queue* yang berfungsi untuk menyimpan *event-event* yang terjadi di dalam himpunan data. *Queue* memiliki prinsip FIFO (*First In First Out*), sehingga *event* akan diproses secara berurutan menurut antrian waktu. *Event Queue* menyimpan empat informasi, yaitu *timestamp*,

pemilik *event*, ID pemilik *event*, dan aksi *event*. Untuk lebih jelasnya, atribut *Event Queue* dijelaskan pada Tabel 3.4.

Tabel 3.4 Atribut dari *EventQueue*

Atribut	Deskripsi
<i>timestamp</i>	Waktu terjadinya <i>event</i>
<i>owner</i>	Pemilik <i>event</i> (<i>produk</i> = 0, pelanggan = 1)
<i>ownerId</i>	ID pemilik <i>event</i>
<i>action</i>	Jenis aksi yang dilakukan (<i>masuk</i> = 0, <i>keluar</i> = 1)

3.3.1.3 *Pandora Box*

Pandora Box adalah sebuah struktur data *array* dua dimensi yang terdiri dari sumbu *x* (*time series*) dan sumbu *y* (*produk*). Struktur data ini digunakan untuk menyimpan skor kontribusi pasar produk setiap waktu. Menggunakan contoh *dataset* pada Tabel 3.5, maka model *Pandora Box* yang terbentuk adalah seperti pada Gambar 3.3.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P ₁	0	1	2	1	0.67	0	0	0	0	0	0	0	0	0	0
P ₂	0	0	0	0	0	1.33	1.33	1.33	0.5	1	1	1	1	0	0
P ₃	0	0	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5	1	0.5
P ₄	0	0	0	2	1.67	1.33	1.33	1.33	0.5	0	0	0	0	0	0
P ₅	0	0	0	0	1.67	1.33	1.33	1.33	1	1.5	1.5	1.5	1.5	2	1.5

Gambar 3.3 Contoh *Pandora Box* dari *dataset* 3.5

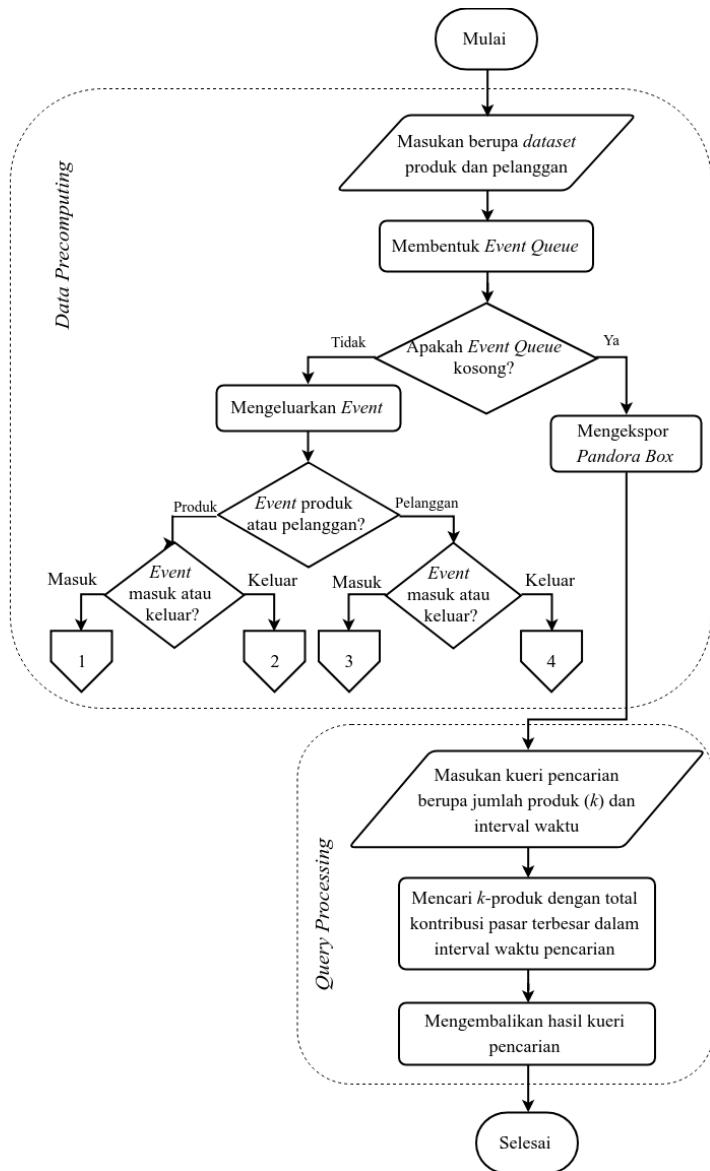
3.3.2 Algoritme Utama

Sebagaimana yang telah dijelaskan sebelumnya bahwa algoritme k-MPPTI terdiri dari dua tahap pemrosesan, yaitu *data precomputing* dan *query processing*. Secara garis besar, alur kerja sistem secara umum disajikan dalam bentuk diagram alur yang dapat dilihat pada Gambar 3.4.

Tahap *data precomputing* bertujuan untuk menghitung skor kontribusi pasar masing-masing produk berdasarkan preferensi pelanggan. Diawali dengan pembentukan *Event Queue* untuk mencatat semua *event* yang terjadi selama pemrosesan data. Kemudian, memproses *event-event* tersebut menggunakan algoritme pemrosesan berdasarkan jenis *event*-nya. Terakhir adalah mengekspor *Pandora Box* untuk digunakan sebagai masukan pada tahap *query processing*.

Tahap kedua adalah *query processing* yang bertujuan untuk memproses kueri pencarian yang dimasukkan oleh pengguna berupa jumlah produk (k) dan interval waktu pencarian. Diawali dengan mencari produk sejumlah k yang memiliki total skor kontribusi pasar terbesar selama interval waktu pencarian, kemudian mengembalikan hasil kueri pencarian berupa k -produk yang paling menjanjikan kepada pengguna.

Untuk memudahkan interaksi antara pengguna dan sistem, dibuatlah aplikasi berbasis web yang memudahkan pengguna memasukkan data produk dan pelanggan, melihat pratinjau dan visualisasi data, memasukkan kueri pencarian, serta melihat hasil kueri pencarian.



Gambar 3.4 Diagram alur algoritme k-MPPTI

3.3.2.1 Data Precomputing

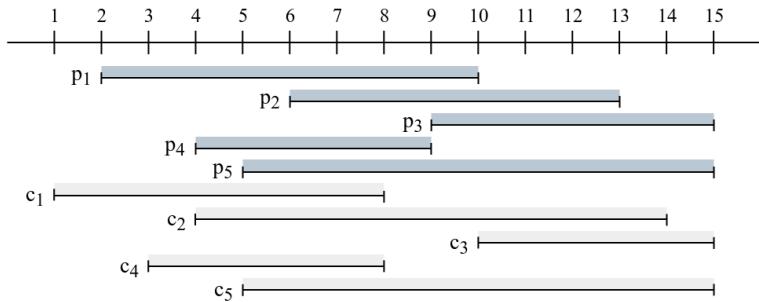
Data precomputing adalah sebuah proses yang dapat menunjang performa algoritme *query processing* supaya dapat bekerja lebih efektif dan efisien. Tidak adanya proses *data precomputing* menyebabkan pengulangan komputasi data setiap kali seseorang memasukkan kueri pencarian. Karena data yang digunakan adalah *historical data*, yaitu data yang dikumpulkan dari kejadian yang telah lalu, maka komputasi data cukup dilakukan satu kali saja di awal (*precomputing*). Berbeda halnya jika data yang digunakan adalah *streaming data* yang nilainya terus berubah dalam periode waktu tertentu.

Tabel 3.5 Contoh *dataset*
 (a) produk P dan (b) preferensi pelanggan C

ID	Timestamp		Nilai	
	t_i	t_e	d_1	d_2
p_1	2	10	6	3
p_2	6	13	4	12
p_3	9	15	6	15
p_4	4	9	9	5
p_5	5	15	12	10

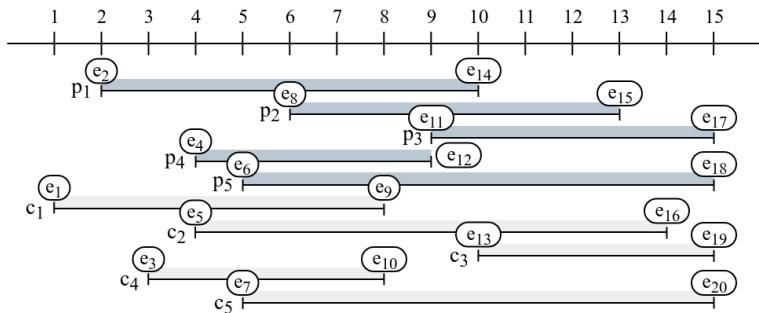
ID	Timestamp		Nilai	
	t_i	t_e	d_1	d_2
c_1	1	8	2	8
c_2	4	14	4	10
c_3	10	15	6	11
c_4	3	8	8	12
c_5	5	15	9	10

Pada Tabel 3.5, diberikan contoh *dataset* produk P dan preferensi pelanggan C yang setiap datanya direpresentasikan sebagai titik d -dimensi dengan serial waktu $[t_i : t_e]$. Pendekatan yang digunakan untuk memproses data multidimensi dengan serial waktu adalah menggunakan lini masa yang diilustrasikan pada Gambar 3.5.



Gambar 3.5 Lini masa data produk dan pelanggan

Lini masa atau alur waktu adalah suatu representasi kronologis urutan peristiwa atau kejadian (*event*). Lini masa dapat dibuat menurut era, abad, tahun, bulan, minggu, atau hari, namun untuk pemodelan ini, waktu direpresentasikan sebagai bilangan bulat positif. Di dalam lini masa, terdapat titik-titik yang mewakili kejadian penting (*event*) yang dinotasikan dengan $e \in E$.

Gambar 3.6 *Event* dalam lini masa data produk dan pelanggan

Ada empat jenis *event* di dalam lini masa, yaitu data produk masuk, produk keluar, pelanggan masuk, dan pelanggan keluar. *Event-event* tersebut dicatat dan dimasukkan ke dalam *Event Queue*, kemudian diproses satu persatu secara berurutan. Contoh *Event Queue* yang terbentuk dari *dataset* pada Tabel 3.5 ditunjukkan pada Tabel 3.6.

Tabel 3.6 *EventQueue*

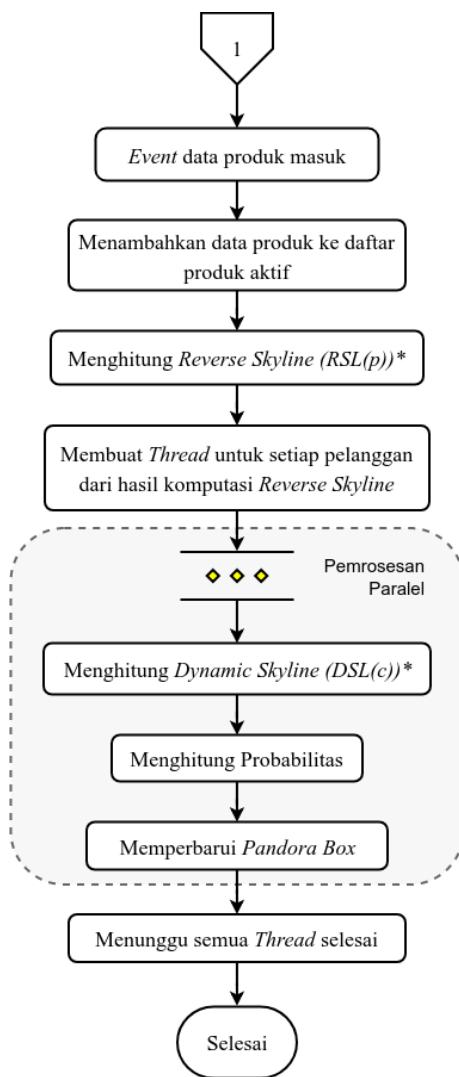
ID Event	Timestamp	ID Data	Aksi
e_1	1	c_1	Masuk
e_2	2	p_1	Masuk
e_3	3	c_4	Masuk
e_4	4	p_4	Masuk
e_5	4	c_2	Masuk
e_6	5	p_5	Masuk
e_7	5	c_5	Masuk
e_8	6	p_2	Masuk
e_9	8	c_1	Keluar
e_{10}	8	c_4	Keluar
e_{11}	9	p_3	Masuk
e_{12}	9	p_4	Keluar
e_{13}	10	c_3	Masuk
e_{14}	10	p_1	Keluar
e_{15}	13	p_2	Keluar
e_{16}	14	c_2	Keluar
e_{17}	15	p_3	Keluar
e_{18}	15	p_5	Keluar
e_{19}	15	c_3	Keluar
e_{20}	15	c_5	Keluar

Ada empat jenis algoritme pemrosesan berdasarkan jenis *event*-nya, yaitu: (1) *Product Insertion*, (2) *Product Deletion*, (3) *Customer Insertion*, dan (4) *Customer Deletion*. Masing-masing proses itu membutuhkan dua jenis komputasi *skyline*, yaitu *dynamic skyline* dan *reverse skyline*, sebagai metode perhitungan probabilitas dan kontribusi pasar [1].

3.3.2.1.1 Proses *Product Insertion*

Proses *Product Insertion* adalah proses yang dijalankan ketika ada data produk yang masuk, dinotasikan dengan p_{in} . Proses ini sangat penting dilakukan karena ada kemungkinan jika produk baru dapat mendominasi produk lama, sehingga hasil *dynamic skyline* seorang pelanggan $c \in C$ dan perhitungan probabilitasnya ikut berubah.

Secara garis besar, algoritme pemrosesan disajikan dalam bentuk diagram alur pada Gambar 3.7. Pemrosesan diawali dengan (1) menambahkan produk p_{in} ke dalam daftar produk aktif PA . Kemudian, (2) menghitung $RSL(p_{in})$. Dari hasil $RSL(p_{in})$ akan didapatkan hasil berupa sejumlah pelanggan yang menganggap p_{in} sebagai hasil *DSL*-nya. Dilanjutkan dengan (3) menghitung $DSL(c)$ untuk masing-masing $c \in RSL(p_{in})$ dan (4) menghitung probabilitas masing-masing produk. Diakhiri dengan (5) memperbarui *Pandora Box*.



Gambar 3.7 Diagram alur proses *product insertion*

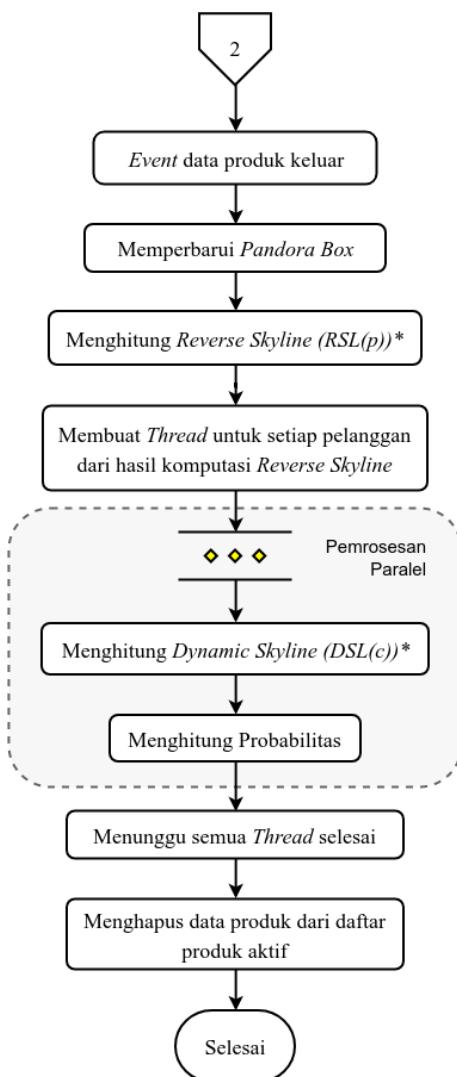
3.3.2.1.2 Proses *Product Deletion*

Proses *Product Deletion* adalah proses yang dijalankan ketika ada data produk yang keluar, dinotasikan dengan p_{out} . Proses ini sangat penting dilakukan karena ada kemungkinan jika sebuah produk yang pernah menjadi hasil *dynamic skyline* seorang pelanggan $c \in C$ keluar, maka produk lain yang pernah didominasi akan menjadi hasil $DSL(c)$ yang baru.

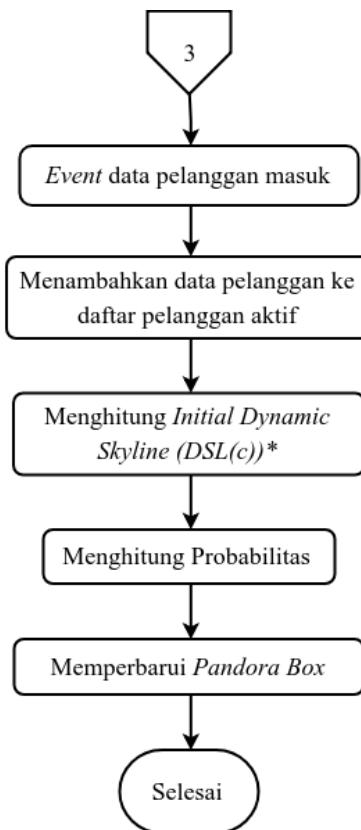
Secara garis besar, algoritme pemrosesan disajikan dalam bentuk diagram alur pada Gambar 3.8. Pemrosesan diawali dengan (1) memperbarui *Pandora Box* untuk mengisi indeks *PBox* sebelumnya yang kosong. Kemudian (2) menghitung $RSL(p_{out})$. Dari hasil $RSL(p_{out})$ akan didapatkan hasil berupa sejumlah pelanggan yang menganggap p_{out} sebagai hasil DSL -nya. Dilanjutkan dengan (3) menghitung $DSL(c)$ untuk masing-masing $c \in RSL(p_{out})$ yang dimaksudkan untuk mencari produk lain yang pernah didominasi. Diakhiri dengan (4) menghitung probabilitas masing-masing produk, serta (5) menghapus produk p_{out} dari daftar produk aktif *PA*.

3.3.2.1.3 Proses *Customer Insertion*

Proses *Customer Insertion* adalah proses yang dijalankan ketika ada data pelanggan yang masuk, dinotasikan dengan c_{in} . Secara garis besar, algoritme pemrosesan disajikan dalam bentuk diagram alur pada Gambar 3.9. Pemrosesan diawali dengan (1) menambahkan pelanggan c_{in} ke dalam daftar pelanggan aktif *CA*. Kemudian (2) menghitung *Initial DSL*(c_{in}) untuk mendapatkan hasil *dynamic skyline* awal. Diakhiri dengan (3) menghitung probabilitas dan (4) memperbarui *Pandora Box*.



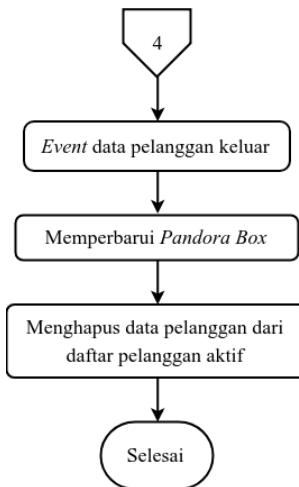
Gambar 3.8 Diagram alur proses *product deletion*



Gambar 3.9 Diagram alur proses *customer insertion*

3.3.2.1.4 Proses *Customer Deletion*

Proses *Customer Deletion* adalah proses yang dijalankan ketika ada data pelanggan yang keluar, dinotasikan dengan c_{out} . Secara garis besar, algoritme pemrosesan disajikan dalam bentuk diagram alur pada Gambar 3.10. Pemrosesan diawali dengan (1) memperbarui *Pandora Box* dengan cara untuk mengisi indeks *PBox* sebelumnya yang kosong, kemudian diakhiri dengan (2) menghapus pelanggan c dari daftar pelanggan aktif *CA*.



Gambar 3.10 Diagram alur proses *customer deletion*

3.3.2.1.5 Komputasi *Reverse Skyline*

Komputasi *reverse skyline* digunakan untuk mencari pelanggan potensial dari sudut pandang produsen [1]. *Reverse skyline* [3] dari sebuah produk $p_1 \in P$, dinotasikan dengan $RSL(p_1)$, berisi semua pelanggan $c \in C$ yang memiliki p_1 pada hasil *dynamic skyline*-nya.

Komputasi *reverse skyline* diawali dengan menentukan *orthant* dari produk, dinotasikan dengan O . Dalam geometri, *orthant* adalah analog dalam ruang data d -dimensi atau biasa dikenal sebagai kuadran dalam bidang dua dimensi. Setiap produk p memiliki 2^d *orthant* pada data d -dimensi.

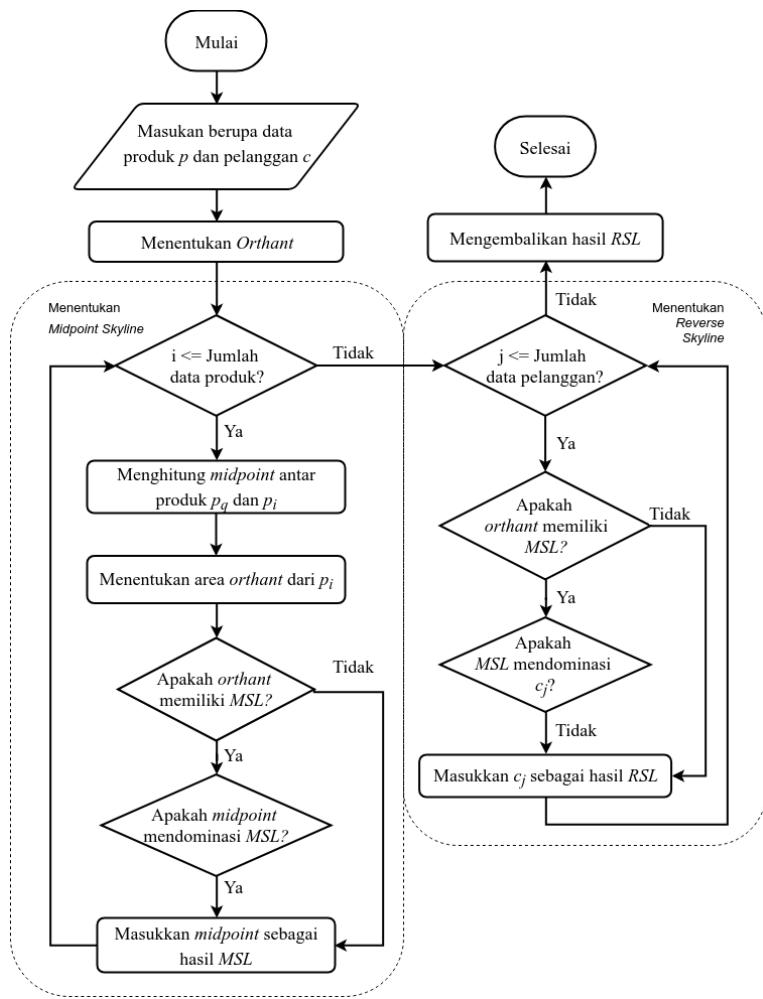
Orthant ditandai menggunakan bilangan biner. Sebagai contoh, terdapat empat *orthant* pada bidang dua dimensi, yaitu O_{00} , O_{01} , O_{10} , dan O_{11} , dan delapan *orthant* pada bidang tiga dimensi, yaitu O_{000} , O_{001} , O_{010} , O_{011} , O_{100} , O_{101} , O_{110} dan O_{111} . Penggunaan bilangan biner bertujuan untuk menandai batas wilayah sebuah *orthant*. Misalnya, *orthant* O_{010} dari produk p_1 memiliki wilayah dengan batas-batas sebagai berikut: sumbu $x[0 : pos_x(p_1)]$, sumbu $y[pos_y(p_1) : max_y]$, dan sumbu $z[0 : pos_z(p_1)]$.

Langkah selanjutnya adalah menghitung *midpoint* atau titik tengah antara produk kueri dan produk lainnya, misalnya p_1 (sebagai titik kueri) dan $p_2 \in P$, menggunakan rumus berikut:

$$m_2^i = \frac{(p_1^i + p_2^i)}{2} \quad (3.1)$$

Kemudian, menentukan *midpoint skyline* atau *mid-skyline* [8] pada setiap *orthant*, dinotasikan dengan $MSL(o)$.

Langkah terakhir adalah mengecek setiap pelanggan $c \in C$ apakah didominasi oleh hasil *mid-skyline* pada masing-masing *orthant* atau tidak (Persamaan 2.4). Jika c didominasi, maka c tidak dapat menjadi hasil *reverse skyline*.

Gambar 3.11 Diagram alur komputasi *reverse skyline*

3.3.2.1.6 Komputasi *Dynamic Skyline*

Komputasi *dynamic skyline* digunakan untuk mencari produk terbaik dari sudut pandang pelanggan [1]. *Dynamic skyline* [2] dari seorang pelanggan $c_1 \in C$, dinotasikan dengan $DSL(c_1)$, berisi semua produk $p_1 \in P$ yang tidak didominasi oleh produk lain $p_2 \in P$ berdasarkan preferensi pelanggan c_1 , $p_2 \not\prec_{c_1} p_1$.

Secara umum, proses komputasi *dynamic skyline* diawali dengan perhitungan selisih absolut dari nilai masing-masing dimensi antara pelanggan dan produk, dinotasikan dengan:

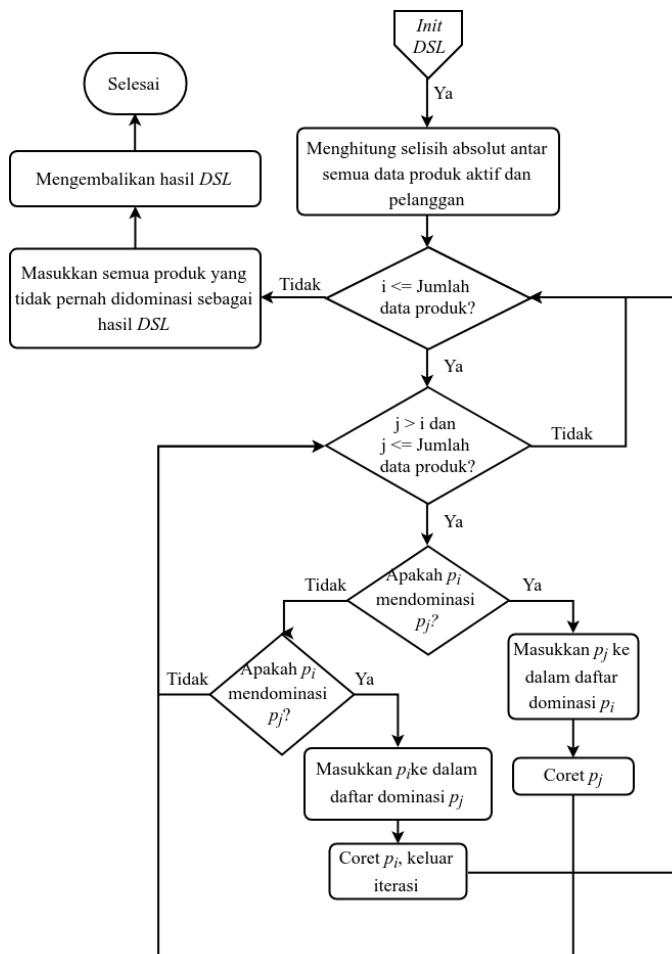
$$diff^i = |c_1^i - p^i| \quad (3.2)$$

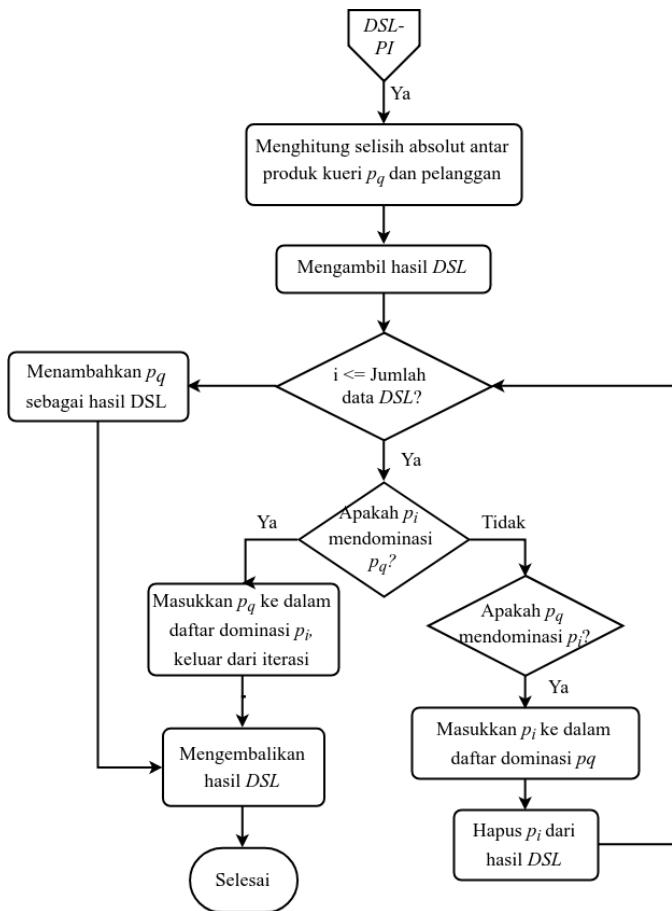
Selanjutnya, mengecek dominansi dinamis antar produk dengan membandingkan selisih absolut-nya. Misalnya, ada dua produk yang akan dibandingkan, dinotasikan dengan p_s sebagai subjek yang dibandingkan dan p_o sebagai objek pembanding. Berdasarkan syarat dominansi dinamis (Persamaan 2.2), p_s dikatakan mendominasi p_o jika dan hanya jika:

- (a) $diff_s^i \leq diff_o^i, \forall i \in [1, \dots, d]$
 - (b) $diff_s^i < diff_o^i, \exists i \in [1, \dots, d]$
- (3.3)

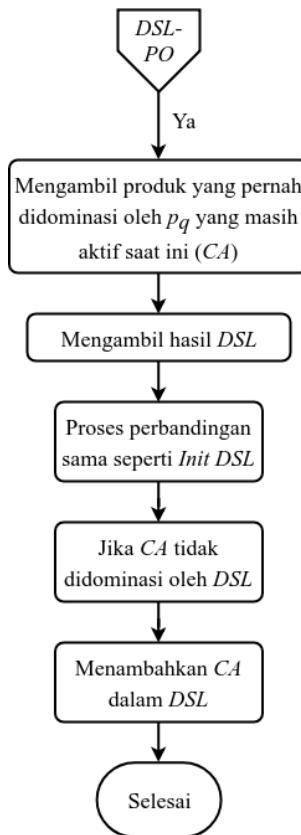
Pengecekan dominansi dinamis ini dilakukan secara iteratif sampai dipastikan suatu p_1 tidak didominasi oleh p_2 lain sama sekali. Jika p_1 pernah didominasi, maka p_1 tidak dapat menjadi hasil *dynamic skyline*.

Komputasi *DSL* dalam k-MPPTI dibagi menjadi 3 jenis, yaitu (1) *Initial DSL*, digunakan ketika ada data pelanggan yang masuk (Gambar 3.12), (2) *DSL – PI*, digunakan ketika ada data produk yang masuk (Gambar 3.13), dan (3) *DSL – PD*, digunakan ketika ada data produk yang keluar (Gambar 3.14).

Gambar 3.12 Diagram alur komputasi *initial dynamic skyline*



Gambar 3.13 Diagram alur komputasi *dynamic skyline - product insertion*



Gambar 3.14 Diagram alur komputasi *dynamic skyline - product deletion*

3.3.2.1.7 Perhitungan Probabilitas

Setelah mendapatkan hasil *dynamic skyline* dan *reverse skyline*, selanjutnya adalah menghitung probabilitas masing-masing produk $p \in P_{active}$ dipilih oleh pelanggan $c \in C_{active}$ yang dinotasikan oleh persamaan berikut:

$$Pr_t(c, p|PA) = \begin{cases} \frac{1}{|DSL(c)|} & \text{if } p \in DSL(c) \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

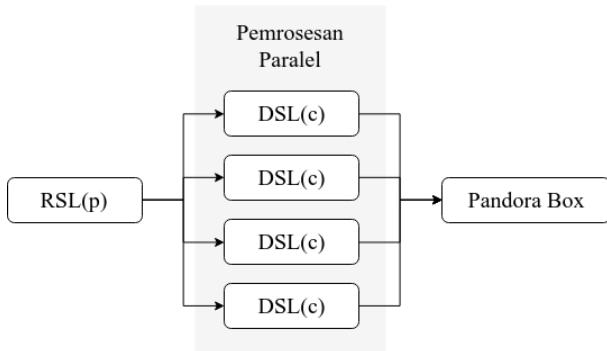
Karena probabilitas produk p dipilih oleh pelanggan yang tidak memiliki p pada hasil *dynamic skyline*-nya adalah nol, maka perhitungan probabilitas dapat disederhanakan menjadi:

$$Pr_t(c, p|PA), \forall c \in RSL(p) \quad (3.5)$$

3.3.2.1.8 Pemrosesan Paralel

Untuk meningkatkan efisiensi waktu komputasi, algoritme k-MPPTI mengimplementasikan konsep pemrosesan paralel, yaitu suatu bentuk komputasi dua atau lebih tugas yang dilakukan secara bersamaan dan beroperasi dengan prinsip bahwa masalah besar seringkali dapat dibagi dan dipecah menjadi masalah yang lebih kecil, kemudian dipecahkan secara bersamaan (paralel) [9].

Pemrosesan paralel dilakukan dengan cara menggunakan satu atau lebih CPU atau prosesor untuk menjalankan program atau *multi-thread*. Karena dilakukan secara bersamaan, maka pemrosesan ini hanya dapat dilakukan jika suatu tugas tidak membutuhkan masukan dari keluaran tugas sebelumnya, misalnya komputasi $DSL(c)$ untuk setiap $c \in RSL(p)$ yang diilustrasikan pada Gambar 3.15.



Gambar 3.15 Pemrosesan paralel

3.3.2.2 *Query Processing*

Query processing adalah algoritme pencarian produk sejumlah k yang paling menjanjikan dalam interval waktu tertentu. Selanjutnya kueri ini disebut dengan k -MPPTI (*k-Most Promising Products in Time Interval*) yang dinotasikan oleh Persamaan 3.6.

$$k - MPPTI(k, [t_i : t_e]) \quad (3.6)$$

Kueri k -MPPTI hanya membutuhkan dua masukan saja, yaitu bilangan bulat positif k yang lebih kecil dari $|P|$ sebagai jumlah produk yang dicari dan interval waktu pencarian yang terdiri dari waktu awal dan waktu akhir $[t_i : t_e]$. Berbeda dengan kueri k -MPP (Persamaan 2.9), kueri k -MPPTI tidak membutuhkan masukan *dataset* produk P dan preferensi pelanggan C lagi karena sudah melalui tahap *data precomputing* yang menghasilkan *Pandora Box*.

Algoritme *query processing* mengadaptasi strategi pemilihan produk k -MPP, yakni memilih *subset* k produk P' dari P yang memiliki kontribusi pasar lebih besar dibandingkan dengan *subset*

k produk P'' dari P yang lain [1] dengan menambahkan interval waktu pencarian. Perhitungan kontribusi pasar dinotasikan dengan persamaan berikut.

$$E_{[t_i:t_e]}(CA, p|PA) = \sum_{t=t_i}^{t_e} \sum_{\forall c \in CA} Pr_t(c, p|PA) \quad (3.7)$$

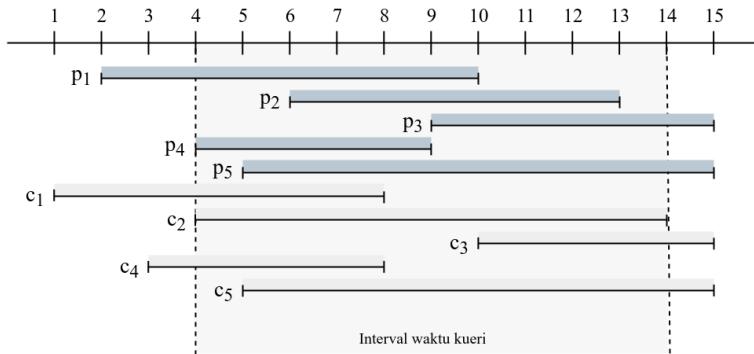
$$E_{[t_i:t_e]}(CA, p|PA) = \sum_{t=t_i}^{t_e} \sum_{\forall c \in RSL(p)} Pr_t(c, p|PA) \quad (3.8)$$

Ada dua langkah pemrosesan yang harus dilakukan, yaitu (1) mengakumulasi skor kontribusi pasar setiap produk $p \in P$ dalam interval waktu pencarian (Persamaan 3.8). Kemudian (2) mengurutkan total skor dari yang terbesar dan mengembalikan produk sejumlah k teratas sebagai hasil dari kueri pencarian.

Misalnya, seorang pengguna ingin mencari 3 produk yang paling menjanjikan dalam interval waktu 4 hingga 14, dinotasikan dengan $k - MPPTI(3, [4 : 14])$. Berdasarkan hasil perhitungan total kontribusi pasar berbasis interval waktu pada Tabel 3.7, produk p_5 , p_2 dan p_4 adalah 3 produk paling menjanjikan dalam interval waktu 4 hingga 14.

Tabel 3.7 Kontribusi pasar (1)

$E_{[4:14]}(CA, p_5 PA)$	14.67
$E_{[4:14]}(CA, p_2 PA)$	8.5
$E_{[4:14]}(CA, p_4 PA)$	8.17
$E_{[4:14]}(CA, p_3 PA)$	3
$E_{[4:14]}(CA, p_1 PA)$	1.67



Gambar 3.16 Ilustrasi interval waktu pencarian

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p ₁	0	1	2	1	0.67	0	0	0	0	0	0	0	0	0	0
p ₂	0	0	0	0	0	1.33	1.33	1.33	0.5	1	1	1	1	0	0
p ₃	0	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5	1	0.5	0.5
p ₄	0	0	0	2	1.67	1.33	1.33	1.33	0.5	0	0	0	0	0	0
p ₅	0	0	0	0	1.67	1.33	1.33	1.33	1	1.5	1.5	1.5	1.5	2	1.5

Interval waktu kueri

Gambar 3.17 Perhitungan kontribusi pasar pada *Pandora Box* berdasarkan interval waktu pencarian

Interval waktu pencarian sangat mempengaruhi hasil kueri. Sebagai bukti, jika interval waktu pencarriannya diubah dari 1 hingga 6, [1 : 6], maka hasil kueri 3 produk teratas yang paling menjanjikan adalah p_4 , p_1 , dan p_5 .

Tabel 3.8 Kontribusi pasar (2)

$E_{[4:14]}(CA, p_4 PA)$	5
$E_{[4:14]}(CA, p_1 PA)$	1.67
$E_{[4:14]}(CA, p_5 PA)$	3
$E_{[4:14]}(CA, p_2 PA)$	1.33
$E_{[4:14]}(CA, p_3 PA)$	0

3.3.3 Algoritme Tandingan

Dalam Tugas Akhir ini juga dibuat sebuah algoritme tandingan untuk membandingkan performanya. Algoritme tandingan kurang lebih mirip dengan dengan algoritme k-MPPTI (Gambar 3.4), namun tidak melalui komputasi *reverse skyline*. Untuk selanjutnya, algoritme ini disebut k-MPPTI NoRSL.

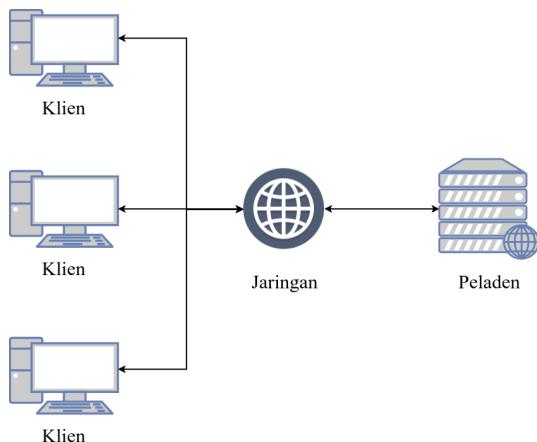
Selain itu, Tugas Akhir ini juga membuat versi algoritme k-MPPTI tanpa menggunakan teknik pemrosesan paralel untuk membandingkan performanya. Untuk selanjutnya, algoritme ini disebut dengan k-MPPTI, sedangkan algoritme utama disebut dengan k-MPPTI Paralel.

3.3.4 Arsitektur Aplikasi

Sistem akan diimplementasikan menggunakan arsitektur *client-server* seperti yang diilustrasikan pada Gambar 3.18. Terdapat dua komponen utama dalam arsitektur ini, yaitu klien (*client*), pihak yang meminta atau menerima layanan, dan peladen (*server*), pihak yang memberikan atau mengirim layanan. Komponen-komponen ini terhubung ke jaringan, baik melalui kabel maupun nirkabel untuk melakukan transmisi data.

Klien mengimplementasikan antarmuka pengguna grafis atau APG (Inggris: *graphical user interface* atau GUI) berbasis web, sedangkan peladen mengimplementasikan *back-end service*

yang berisi algoritme komputasi k-MPPTI yang terdiri atas algoritme *data precomputing* dan *query processing*. Web dibangun menggunakan Flask *microframework* dan bahasa Python, HTML, CSS, serta Javascript, sedangkan *back-end service* diimplementasikan menggunakan bahasa Python. Flask juga sekaligus berperan sebagai peladen web (*web server*).



Gambar 3.18 Perancangan arsitektur aplikasi

Pengguna melakukan masukan data melalui web, kemudian data tersebut dikirimkan ke *back-end service* untuk dilakukan proses *data precomputing*. Setelah hasil *data precomputing* selesai, pengguna melakukan masukan kueri pencarian. Kueri pencarian tersebut akan dikirimkan ke *back-end service* untuk dilakukan proses *query processing*. Hasil yang didapatkan akan dikembalikan ke klien, disertai dengan visualisasi data menggunakan pustaka Vis.js.

BAB IV

IMPLEMENTASI

Pada bab ini dijelaskan mengenai implementasi dari perancangan struktur data dan algoritme untuk menyelesaikan permasalahan *k-Most Promising Products* (k-MPP) berbasis interval waktu sebagaimana yang telah dijelaskan pada Bab III. Penjelasan implementasi terdiri dari penjelasan kelas dan fungsi yang dibuat, disertai dengan *pseudocode* untuk masing-masing fungsi tersebut.

4.1 Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan Tugas Akhir ini meliputi perangkat keras dan perangkat lunak dengan spesifikasi sebagai berikut:

1. Perangkat Keras:
 - Prosesor 2.6 GHz Intel Core i5 (I5-4278U)
 - Memori 8 GB 1600 MHz DDR3
2. Perangkat Lunak:
 - Sistem operasi macOS Mojave Versi 10.14.5
 - *Text editor* Visual Studio Code Versi 1.33.1
 - Bahasa Pemrograman Python 3.7.3
 - Flask *Microframework* 1.0.2

4.2 Implementasi *Data Precomputing*

Data precomputing merupakan pemrosesan tahap pertama dalam algoritme k-MPPTI yang bertujuan untuk mengolah *dataset* produk P dan preferensi pelanggan C dengan jumlah data sebanyak

n dan ukuran dimensi sebesar d , menjadi sebuah *PandoraBox* yang menyimpan skor kontribusi pasar semua produk pada setiap waktu.

Algoritme ini diimplementasikan menggunakan paradigma pemrograman berorientasi objek, sehingga semua data dan fungsi dibungkus dalam kelas-kelas. Ada empat macam kelas yang diimplementasikan, yaitu kelas *EventQueue*, *PandoraBox*, *ReverseSkyline*, dan *DynamicSkyline*.

Algorithm 1 Precomputing

Input : product dataset P , customer dataset C
Output : pandora box PB

- 1: $EQ \leftarrow initEventQueue()$
- 2: $D \leftarrow$ Data indexing P, C
- 3: **for all** $d \in D$ **do**
- 4: **for** $i \leftarrow 0$ to 2 **do**
- 5: **if** i is 0 **then** $enqueue(t_{in}, dt, id, i)$ to EQ
- 6: **else** $enqueue(t_{out}, dt, id, i)$ to EQ
- 7: $sortQueue(EQ)$
- 8: $PA \leftarrow \emptyset$ of active products
- 9: $CA \leftarrow \emptyset$ of active customers
- 10: $MaxTs \leftarrow$ get maximum timestamp
- 11: $MaxVal \leftarrow$ get maximum value
- 12: $PB \leftarrow$ new $initPandoraBox(Len(P), MaxTs)$
- 13: $RSL \leftarrow$ new $initReverseSkyline(P, C, Dim)$
- 14: $DSL \leftarrow$ new $initDynamicSkyline(P, C, Dim)$

Gambar 4.1 Algoritme *precomputing* (bagian 1)

Secara garis besar, langkah yang dilakukan selama *precomputing* adalah (1) *indexing* data dan pembentukan *EventQueue* (baris pertama hingga ke-tujuh pada Algoritme 4.1), (2) inisialisasi dan instansiasi objek (baris ke-delapan hingga terakhir pada Algoritme 4.1), dan (3) pemrosesan *event* (Algoritme 4.3).

Algorithm 2 Precomputing

```

15: while  $EQ$  not empty() do
16:    $e \leftarrow dequeue(EQ)$ 
17:   if  $e$  role is product then
18:     if  $e$  action is insertion then
19:        $PA \leftarrow append\ p \in e$ 
20:        $RSL(p) \leftarrow computeRSL(p)$ 
21:       for all  $c \in RSL(p)$  parallel do
22:          $DSL(c) \leftarrow computeDSL(c, ts, act, p)$ 
23:       for all  $c \in CA$  parallel do
24:          $updatePBox(DSL(c))$ 
25:     else if  $e$  act is deletion then
26:       for all  $c \in CA$  parallel do
27:          $DSL(c) \leftarrow updatePBox(DSL(c))$ 
28:        $RSL(p) \leftarrow computeRSL(p)$ 
29:       for all  $c \in RSL(p)$  parallel do
30:          $DSL(c) \leftarrow computeDSL(c, ts, act, p)$ 
31:        $PA \leftarrow remove\ p \in e$ 
  
```

Gambar 4.2 Algoritme *precomputing* (bagian 2)

Algorithm 3 Precomputing

```

32:   else if  $e$  role is customer then
33:     if  $e$  act is insertion then
34:        $CA \leftarrow \text{append } c \in e$ 
35:        $DSL(c) \leftarrow \text{compute}DSL(c, ts)$ 
36:        $\text{update}PBox(DSL(c))$ 
37:     else if  $e$  act is deletion then
38:        $DSL(c) \leftarrow \text{compute}DSL(c, ts)$ 
39:        $\text{update}PBox(DSL(c))$ 
40:        $CA \leftarrow \text{remove } c \in e$ 

```

Gambar 4.3 Algoritme *precomputing* (bagian 3)**4.2.1 Kelas *EventQueue***

Kelas *EventQueue* mendefinisikan bentuk dan perilaku dari objek *Event Queue* yang berfungsi untuk menyimpan *event-event* yang terjadi di dalam himpunan data. *EventQueue* tidak dibuat menggunakan struktur data *queue* karena data yang digunakan adalah *historical* dan diurutkan berdasarkan *timestamp* dari data tersebut, bukan berdasarkan urutan masuknya data. Sehingga, *EventQueue* diimplementasikan menggunakan *array* yang bekerja seperti *queue*, yakni FIFO (*First In First Out*).

Ada enam fungsi yang diimplementasikan dalam kelas ini sebagaimana yang ditunjukkan oleh Algoritme 4.4, namun hanya ada tiga fungsi utama, yaitu *enqueue* untuk memasukkan data ke dalam *array* pada indeks pertama, *dequeue* untuk mengeluarkan data dari *array* pada indeks terakhir; dan *sortQueue* untuk mengurutkan data di dalam array.

Algorithm 4 EventQueue Class

Input : timestamp t , role o (product/customer), data ID oid , action a (insertion/deletion)

Output : event queue E

- 1: **procedure** $initEventQueue$
- 2: $E \leftarrow \emptyset$
- 3: **procedure** $enqueue(t, o, oid, a)$
- 4: $e \leftarrow [t, o, oid, a]$
- 5: $E \leftarrow \text{append } e$
- 6: **procedure** $dequeue$
- 7: $e \leftarrow \text{pop an element from } E$
- 8: **return** e
- 9: **procedure** $sortQueue$
- 10: $E \leftarrow \text{sort elements in descending order based on sorting priority (timestamp, insertion act, deletion act, product, customer, data ID)}$
- 11: **procedure** $getTotalQueue$
- 12: **return** number of elements in E
- 13: **procedure** $getMaxTimestamp$
- 14: **return** timestamp in the first index of E (descending order)
- 15: **procedure** $getMinTimestamp$
- 16: **return** timestamp in last first index of E (descending order)
- 17: **procedure** $empty$
- 18: **if** E is empty **then return** True
- 19: **else return** False

Gambar 4.4 Kelas *EventQueue*

4.2.2 Kelas *PandoraBox*

Kelas *PandoraBox* mendefinisikan bentuk dan perilaku dari objek *Pandora Box* yang berfungsi untuk menyimpan skor kontribusi pasar masing-masing produk $p \in P$ pada setiap waktu dalam interval hidupnya $t \in [t_i : t_e]$. *PandoraBox* diimplementasikan menggunakan struktur data *array* dua dimensi, yaitu ID produk dan *timestamp*.

Algorithm 5 PandoraBox Class

Input : $DSL(c)$, timestamp now ts , probability score pr , last timestamp lts , last probability score lpr

Output : filled pandora box $PBox$

- 1: **procedure** $initPandoraBox$
- 2: $PBox \leftarrow \emptyset$
- 3: **procedure** $updatePBox(DSL(c))$
- 4: **for all** $p \in DSL(c)$ **do**
- 5: **if** $ts > lts$ **then**
- 6: $updateScore(p, ts, pr, lts, lpr)$
- 7: **else**
- 8: $PBox(p, ts) \leftarrow PBox(p, ts) + pr$
- 9: **procedure** $updateScore(p, ts, lts, pr, lpr)$
- 10: **for** $i \leftarrow lts + 1$ to ts **do**
- 11: $PBox(p, i) \leftarrow PBox(p, i) + lpr$
- 12: $PBox(p, i) \leftarrow PBox(p, i) + pr$

Gambar 4.5 Kelas *PandoraBox* (bagian 1)

Ada dua fungsi utama yang digunakan dalam proses *data precomputing*, yaitu *updatePBox* untuk memperbarui skor kontribusi pasar pelanggan c dan *updateScore* untuk memperbarui "kotak" sebelumnya menggunakan nilai probabilitas terakhir. Kelas *Pandora Box* ditunjukkan oleh Algoritme 4.5.

4.2.3 Kelas *ReverseSkyline*

Kelas *ReverseSkyline* mendefinisikan bentuk dan perilaku dari objek *Reverse Skyline* yang berfungsi untuk menangani komputasi *reverse skyline*, meliputi (1) pembentukan *orthant* pada fungsi *defineOrthant* dan *getOrthantId*, (2) menghitung *midpoint* antara produk kueri dan produk lainnya pada fungsi *calcMidpoint*, (3) menentukan *midpoint skyline* masing-masing *orthant* pada fungsi *findMidSkyline*, dan (4) menentukan *reverse skyline* pada fungsi *findReverseSkyline*. Kelas *ReverseSkyline* ditunjukkan pada Algoritme 4.6, 4.7, 4.8.

Algorithm 6 ReverseSkyline Class

Input : product as query point p_q , active products PA , active customers CA
Output : $RSL(p)$

- 1: **procedure** *initReverseSkyline*(P, C, d)
- 2: $PA \leftarrow$ get product active from P
- 3: $CA \leftarrow$ get customer active from C
- 4: $O \leftarrow defineOrthant(Dim)$
- 5: **procedure** *computeRSL*(p_q)
- 6: $findMidSkyline(p_q, PA)$
- 7: $RSL(p) \leftarrow findReverseSkyline(CA)$

Gambar 4.6 Kelas *ReverseSkyline* (bagian 1)

Algorithm 7 ReverseSkyline Class

```

8: procedure findMidSkyline( $p_q, PA$ )
9:   for all  $p \in PA$  do
10:    if  $p \neq p_q$  then
11:       $m \leftarrow calcMidpoint(p_q, p)$ 
12:       $id \leftarrow getOrthantId(p)$ 
13:      if  $O_{id}$  is empty then  $O_{id} \leftarrow m$ 
14:      else
15:        for each  $ms \in MSL(o)$  do
16:          if  $m \prec ms$  then
17:             $MSL(o) \leftarrow$  delete  $ms$ 
18:          else if  $ms \prec m$  then
19:            exit the loop
20:          if  $m \not\prec ms$  then
21:             $MSL(o) \leftarrow$  insert  $m$ 
22: procedure findReverseSkyline( $CA$ )
23:    $RSL(p) \leftarrow \emptyset$ 
24:   for  $c \in CA$  do
25:      $id \leftarrow getOrthantId(c)$ 
26:     if  $O_{id}$  is empty then
27:        $RSL(p) \leftarrow$  insert  $c$ 
28:     else
29:        $dom \leftarrow 0$ 
30:       for all  $m \in O_{id}$  do
31:         if  $m \prec c$  then
32:            $dom \leftarrow dom + 1$ 
33:         if  $dom$  is 0 then
34:            $RSL(p) \leftarrow$  insert  $c$ 
35:   return  $RSL(p)$ 

```

Gambar 4.7 Kelas *ReverseSkyline* (bagian 2)

Algorithm 8 ReverseSkyline Class

```

36: procedure calcMidpoint( $p_q, p$ )
37:    $m \leftarrow \emptyset$ 
38:   for each  $i \in d$  do  $m \leftarrow \frac{(p_q^i + p^i)}{2}$ 
39:   return  $m$ 
40: procedure getOrthantId( $D$ )
41:    $o \leftarrow \emptyset$ 
42:   for each  $i \in d$  do
43:     if  $D^i \leq p_q^i$  then  $o \leftarrow \text{append } 0$ 
44:     else  $o \leftarrow \text{append } 1$ 
45:   return  $o$ 

```

Gambar 4.8 Kelas *ReverseSkyline* (bagian 3)**4.2.4 Kelas *Dynamic Skyline***

Kelas *DynamicSkyline* mendefinisikan bentuk dan perilaku dari objek *Dynamic Skyline* yang berfungsi untuk menangani komputasi *dynamic skyline* yang memiliki 3 jenis pemrosesan *event*, yaitu (1) *Initial DSL*, digunakan ketika ada data pelanggan yang masuk (*customer insertion*), (2) *DSL – PI*, digunakan ketika ada data produk yang masuk (*product insertion*), dan (3) *DSL – PD*, digunakan ketika ada data produk yang keluar (*product deletion*). Kelas *DynamicSkyline* ditunjukkan pada Algoritme 4.9 dan 4.10.

Algorithm 9 DynamicSkyline Class

Input : customer c , product data P
Output : $DSL(p)$

- 1: **procedure** $initDynamicSkyline(P, C, d)$
- 2: $PA \leftarrow$ get product active from P
- 3: **procedure** $computeDSL(c, ts, act, p)$
- 4: **if** act is customer insertion **then**
- 5: $initDSL(c, ts)$
- 6: **else if** act is product insertion **then**
- 7: $productIn(c, ts, p)$
- 8: **else if** act is product deletion **then**
- 9: $productOut(c, ts, p)$
- 10: **procedure** $initDSL(c)$
- 11: $CAND \leftarrow PA$
- 12: sort $CAND$
- 13: **for** $i \leftarrow 0$ to length of $CAND$ **do**
- 14: **for** $j \leftarrow i + 1$ to length of $CAND$ **do**
- 15: **if** $p_i \prec p_j$ **then**
- 16: add p_j to the child list of p_i
- 17: $CAND \leftarrow$ remove p_j
- 18: **else if** $p_j \prec p_i$ **then**
- 19: add p_i to the child list of p_j
- 20: $CAND \leftarrow$ remove p_i
- 21: break the loop
- 22: **return** $CAND$ as $DSL(c)$

Gambar 4.9 Kelas *DynamicSkyline* (bagian 1)

Algorithm 10 DynamicSkyline Class

```
23: procedure productIn(p)
24:    $CAND \leftarrow p, DSL(c)$ 
25:   sort  $CAND$ 
26:    $index \leftarrow$  get index of  $p$  in  $CAND$ 
27:   for  $i \leftarrow 0$  to length of  $CAND$  do
28:     if  $i < index$  then
29:       if  $p_i \prec p_{index}$  then
30:         add  $p_{index}$  to the child list of  $p_i$ 
31:          $CAND \leftarrow$  remove  $p_{index}$ 
32:         break the loop
33:     else if  $i > index$  then
34:       if  $p_{index} \prec p_i$  then
35:         add  $p_i$  to the child list of  $p_{index}$ 
36:          $CAND \leftarrow$  remove  $p_i$ 
37:     return  $CAND$  as  $DSL(c)$ 
38: procedure productOut(p)
39:    $AC \leftarrow$  find active childs of  $p$ 
40:    $DSL(c) \leftarrow productIn(AC)$ 
41:   return  $DSL(c)$ 
```

Gambar 4.10 Kelas *DynamicSkyline* (bagian 2)

Algorithm 11 Check Domination

Input : value of subject (val_s), value of target (val_t), value of query point (val_q), dimension of data (d)
Output : is subject dominating target based on query point?
 $(s \prec_q t)$

```

1: procedure isDominating( $val_s$ ,  $val_t$ ,  $val_q$ )
2:    $dg \leftarrow 0$                                 // counter of dominating
3:    $dd \leftarrow 0$                                 // counter of dominated
4:   for each  $i \in d$  do
5:      $diff_s^i \leftarrow |val_s^i - val_q^i|$ 
6:      $diff_t^i \leftarrow |val_t^i - val_q^i|$ 
7:     if  $diff_a^i = diff_b^i$  then
8:       continue the loop
9:     else if  $diff_s^i < diff_t^i$  then
10:       $dg \leftarrow dg + 1$ 
11:    else if  $diff_s^i > diff_t^i$  then
12:       $dd \leftarrow dd + 1$ 
13:    if  $dd = 0$  and  $dg \geq 1$  then
14:      return True
15:    else
16:      return False

```

Gambar 4.11 Fungsi cek dominasi

4.3 Implementasi Algoritme *Query Processing*

Query processing merupakan pemrosesan tahap kedua dalam algoritme k-MPPTI yang bertujuan untuk memproses kueri yang dimasukkan oleh pengguna dengan memanfaatkan *Pandora Box* dari hasil *precomputing*. Proses diawali dengan mengakumulasi skor kontribusi pasar semua produk dalam interval waktu kueri, kemudian hasil akumulasi diurutkan dari yang terbesar. Langkah

terakhir adalah mengembalikan hasil kueri berupa produk sejumlah k beserta skor kontribusi pasarnya.

Algorithm 12 Solution

Input : Pandora Box $PBox$, number of products k , time interval (time init t_i , time end t_e)
Output : k products

```

1: procedure solution( $PBox$ ,  $k$ ,  $t_i$ ,  $t_e$ )
2:    $Res \leftarrow \emptyset$ 
3:   for all  $p \in P$  do
4:      $MC(p) \leftarrow getScore(p, t_i, t_e)$ 
5:   sort  $MC$  in ascending order
6:   for  $i \leftarrow 0$  to  $k$  do
7:      $Res \leftarrow$  append  $MC(i)$ 
return  $Res$ 
```

Gambar 4.12 Algoritme *query processing*

Algorithm 13 PandoraBox Class

Input : product p , time interval (time init t_i , time end t_e)
Output : total market contribution score MC

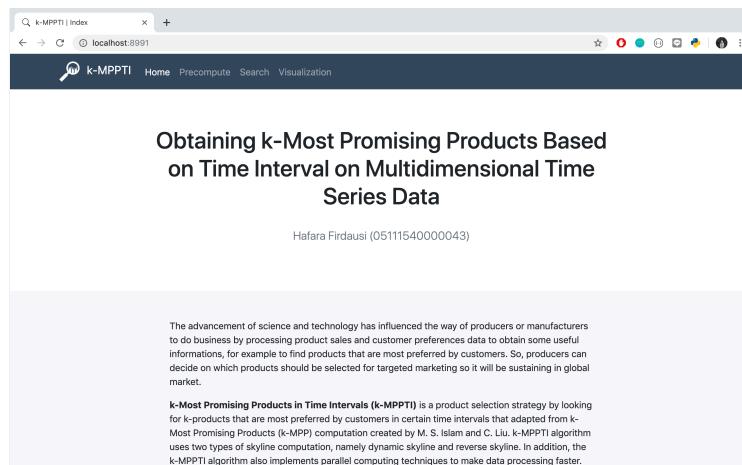
```

13: procedure getScore( $p[id]$ ,  $t_i$ ,  $t_e$ )
14:    $MC \leftarrow 0$ 
15:   for  $i \leftarrow t_i$  to  $t_e + 1$  do
16:      $MC \leftarrow MC + PBox(p, i)$ 
return  $MC$ 
```

Gambar 4.13 Kelas *PandoraBox* (bagian 2)

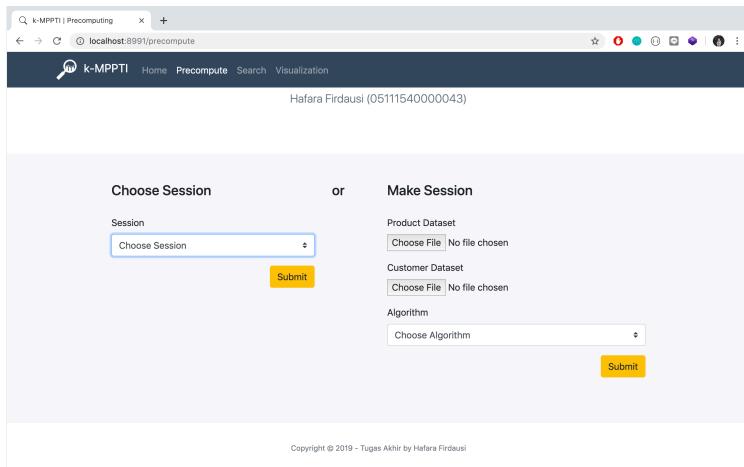
4.4 Implementasi Antarmuka Pengguna

Antarmuka pengguna diimplementasikan menggunakan bahasa pemrograman Python, HTML, CSS, dan JavaScript, serta Flask sebagai kerangka kerja *back-end*, Bootstrap sebagai kerangka kerja *front-end*, dan pustaka Vis.js untuk menampilkan visualisasi data.

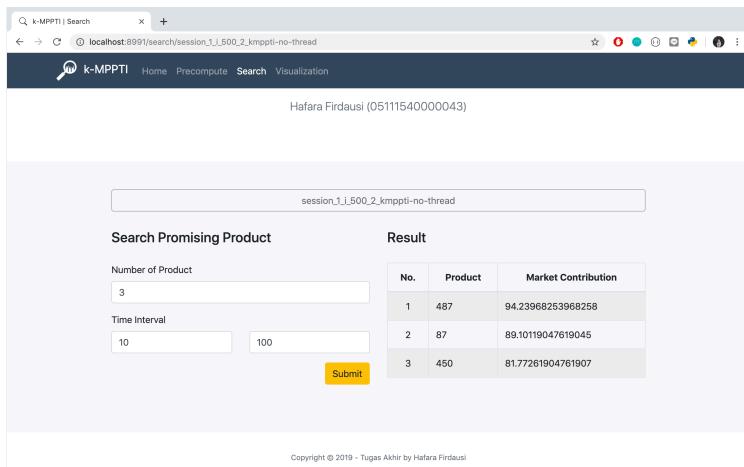


Gambar 4.14 Implementasi halaman *Home*

Antarmuka pengguna memiliki empat menu utama yaitu, "*Home*" sebagai halaman utama yang berisi pengenalan algoritme k-MPPTI, "*Precompute*" untuk melakukan *data precomputing*, "*Search*" untuk melakukan pencarian *k*-produk yang paling diminati pelanggan dalam interval waktu tertentu, dan "*Visualization*" untuk melihat visualisasi data.



Gambar 4.15 Implementasi halaman *Precompute*



Gambar 4.16 Implementasi halaman *Search*

Pada menu “*Precompute*” (Gambar 4.15), pengguna diberikan opsi untuk memasukkan data baru atau menggunakan *session* yang sudah ada. Jika pengguna memasukkan data baru, artinya pengguna membuat *session* baru dan data akan di-*precompute* terlebih dahulu. Jika pengguna memilih *session* yang sudah ada, data tidak perlu di-*precompute* ulang.

Setelah melalui proses *precomputing*, pengguna dapat memasukkan kueri pencarian *k*-produk yang paling menjanjikan dalam interval waktu tertentu. Halaman web akan mengembalikan data berupa hasil kueri pencarian, yakni berupa ID produk dan skor kontribusi pasarnya sebagaimana yang ditunjukkan pada Gambar 4.16.

The screenshot shows a web browser window titled "k-MPPTI | Visualization". The address bar displays "localhost:8991/visualization/session_1_i_500_2_kmppti-no-thread". The page header includes a search icon, a magnifying glass icon, and navigation links for "Home", "Precompute", "Search", and "Visualization". Below the header, a search bar contains the text "session_1_i_500_2_kmppti-no-thread". The main content area is titled "Data Information" and contains a table with the following data:

Data Type	Independent (generate synthetic data)	Number of Dimension	2
Number of Product	500	Maximum Value	291
Number of Customer	500	Minimum Value	1
Attribute of Product	id, label, ts_in, ts_out, dim_0, dim_1,	Maximum Timestamp	339
Attribute of Customer	id, label, ts_in, ts_out, dim_0, dim_1,	Minimum Timestamp	1

Below the table, there are two sections: "Product Data" and "Customer Data", each with a link to "Showfirst 1 to 20 data".

Gambar 4.17 Implementasi halaman *Visualization* (informasi data)

Pengguna juga dapat melihat visualisasi data pada menu “*Visualization*” berupa pratinjau data dalam bentuk tabel dan lini masa. Selain itu, pengguna juga dapat melihat informasi detail dari data. Berikut adalah beberapa cuplikan tampilan visualisasi data

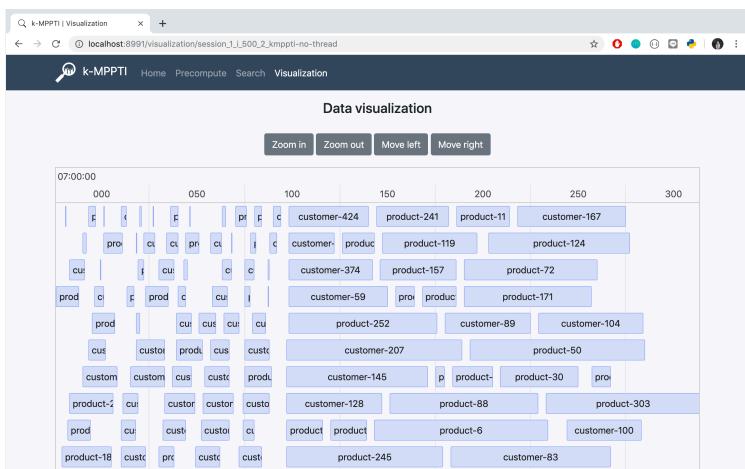
pada Gambar 4.17, 4.18, dan 4.19.

This screenshot shows the 'Visualization' page of the k-MPPTI system. At the top, there's a navigation bar with links for Home, Precompute, Search, and Visualization. Below the navigation bar, two tables are displayed side-by-side under the headings 'Product Data' and 'Customer Data'. Both tables show data from index 1 to 20. The 'Product Data' table has columns for id, label, ts_in, ts_out, dim_0, and dim_1. The 'Customer Data' table has columns for id, label, ts_in, ts_out, dim_0, and dim_1.

ID	Label	ts_in	ts_out	dim_0	dim_1
1	product-1	133	163	118	155
2	product-2	60	92	167	56
3	product-3	260	289	228	154
4	product-4	159	241	236	179
5	product-5	20	32	120	160
6	product-6	168	259	9	189
7	product-7	207	218	85	147
8	product-8	97	138	122	172
9	product-9	28	57	73	115
10	product-10	128	156	141	121

ID	Label	ts_in	ts_out	dim_0	dim_1
1	customer-1	164	198	97	46
2	customer-2	56	98	45	149
3	customer-3	67	86	74	69
4	customer-4	210	303	76	167
5	customer-5	258	261	10	33
6	customer-6	197	238	55	217
7	customer-7	171	192	42	201
8	customer-8	64	72	36	212
9	customer-9	166	194	274	205
10	customer-10	99	187	81	160

Gambar 4.18 Implementasi halaman *Visualization* (pratinjau data)



Gambar 4.19 Implementasi halaman *Visualization* (visualisasi data)

Halaman ini sengaja dikosongkan

BAB V

UJI COBA DAN EVALUASI

Bab ini membahas skenario uji coba aplikasi yang telah dirancang dan diimplementasikan, serta analisis hasil pengujian. Ada dua jenis pengujian, yakni uji coba fungsionalitas dan uji coba performa. Uji coba dilakukan untuk menganalisis kinerja aplikasi dengan lingkungan uji coba yang telah ditentukan.

5.1 Lingkungan Uji Coba

Lingkungan yang digunakan untuk pengujian meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut.

1. Perangkat Keras:
 - Prosesor 2.6 GHz Intel Core i5 (I5-4278U)
 - Memori 8 GB 1600 MHz DDR3
2. Perangkat Lunak:
 - Sistem operasi macOS Mojave Versi 10.14.5
 - Bahasa Pemrograman Python 3.7.3

5.2 Data Uji Coba

Terdapat tiga jenis data yang akan digunakan dalam pengujian Tugas Akhir ini, yaitu data *independent* (IND), *anti-correlated* (ANT), dan *Forest Cover Type* (FC).

5.2.1 Data *Independent* (IND)

Data *independent* (IND) adalah himpunan data yang memiliki persebaran nilai atribut yang acak dan tidak saling terpengaruh satu sama lain. Data ini adalah data sintetis yang dibuat untuk menguji performa algoritma jika berhadapan dengan data yang nilai atributnya tidak memiliki keterkaitan satu sama lain. Rentang nilai yang digunakan untuk setiap atribut adalah 1 – 300.

5.2.2 Data *Anti-Correlated* (ANT)

Data *anti-correlated* (ANT) adalah himpunan data yang memiliki persebaran nilai atribut yang saling bertolak belakang antara satu atribut dengan atribut lainnya, artinya sebuah data memiliki nilai yang sangat baik pada salah satu atributnya, namun sangat buruk pada atribut lainnya. Data ini adalah data sintetis yang dibuat untuk menguji performa algoritma jika berhadapan dengan data yang memiliki banyak titik skyline. Rentang nilai yang digunakan untuk setiap atribut adalah 1 – 300.

5.2.3 Data *Forest Cover Type* (FC)

Data *Forest Cover Type* (FC) adalah himpunan data asli yang berisi hasil pengamatan pohon dari empat area Hutan Nasional Roosevelt di Colorado. Semua pengamatan adalah variabel kartografi dari 30 meter x 30 meter bagian hutan. Dataset ini mencakup informasi tentang jenis pohon, jangkauan bayangan, jarak ke *landmark* terdekat (jalan dan sebagainya), jenis tanah, dan topografi lokal. *Dataset* ini adalah bagian dari *UCI Machine Learning Repository* yang dikumpulkan oleh Blackard, Dean, dan Anderson (1998) di Colorado State University [14].

Himpunan data *Forest Cover Type* (FC) terdiri dari 581.012 data dan memiliki 55 dimensi atribut. Tidak ditemukan nilai yang hilang atau anomali pada himpunan data ini. Seluruh atribut bertipe data numerik dengan rincian dapat dilihat pada Tabel 5.1.

Tabel 5.1 Atribut himpunan data *Forest Cover Type*

Nama Kolom	Jumlah Kolom
Elevation	1
Aspect	1
Slope	1
Horizontal_Distance_To_Hydrology	1
Vertical_Distance_To_Hydrology	1
Horizontal_Distance_To_Roadways	1
Hillshade_9am	1
Hillshade_Noon	1
Hillshade_3pm	1
Horizontal_Distance_To_Fire_Points	1
Wilderness_Area	4
Soil_Type	40
Cover_Type	1

Data *Forest Cover Type* (FC) adalah himpunan data yang diambil dari sumber sebenarnya. Penggunaan data ini bertujuan untuk menguji performa algoritma pada data dengan persebaran dan rentang nilai atribut sebenarnya yang bersifat saling berkaitan.

5.3 Skenario Uji Coba

Uji coba ini dilakukan untuk menguji apakah program yang telah diimplementasikan dapat berjalan dengan sebagaimana mestinya. Uji coba akan dibagi menjadi dua jenis, yaitu uji coba fungsionalitas dan uji coba performa.

5.3.1 Uji Coba Fungsionalitas

Skenario uji coba fungsionalitas berfokus pada pengujian apakah aplikasi dapat digunakan sesuai dengan kebutuhan fungsional pada Tabel 3.2. Skenario pengujian fungsionalitas aplikasi ditunjukkan pada Tabel 5.2.

Tabel 5.2 Skenario uji coba fungsionalitas

No.	Deskripsi Kebutuhan
1	Pengguna dapat mengunggah data
2	Pengguna dapat melihat informasi dan pratinjau data
3	Pengguna dapat melihat visualisasi data
4	Pengguna dapat memilih algoritme yang digunakan untuk <i>data precomputing</i>
5	Pengguna dapat memasukkan kueri pencarian
6	Pengguna dapat melihat hasil kueri

5.3.2 Uji Coba Performa

Pengujian performa bertujuan untuk membandingkan kinerja masing-masing algoritme dalam melakukan *data precomputing*. Dalam sebuah percobaan atau pengujian, biasanya dikenal adanya tiga jenis variabel, yaitu variabel kontrol, manipulasi, dan respon.

Variabel kontrol adalah variabel yang dikendalikan, biasanya dibuat konstan sehingga pengaruh variabel manipulasi terhadap variabel respon tidak dipengaruhi oleh faktor luar yang tidak diteliti. Variabel manipulasi adalah variabel yang dapat memicu suatu perubahan bagi variabel respon. Variabel respon adalah variabel yang berubah akibat dari variabel manipulasi.

Dalam pengujian ini, variabel kontrol yang digunakan adalah jenis data dan jenis algoritme, variabel manipulasinya adalah jumlah data dan jumlah dimensi data, sedangkan variabel responnya berupa waktu komputasi dan penggunaan memori. Skenario pengujian performa ditunjukkan pada Tabel 5.3.

Tabel 5.3 Skenario uji coba performa *data precomputing*

No.	Jenis Algoritme	Jenis Data	Jumlah Data (n)	Jumlah Dimensi (d)
1	k-MPPTI	IND	100, 500, 1000, 5000, 10000	3
	k-MPPTI Paralel			
	k-MPPTI NoRSL			
2	k-MPPTI	ANT	100, 500, 1000, 5000, 10000	3
	k-MPPTI Paralel			
	k-MPPTI NoRSL			
3	k-MPPTI	FC	100, 500, 1000, 5000, 10000	3
	k-MPPTI Paralel			
	k-MPPTI NoRSL			
4	k-MPPTI	IND	1000	2, 3, 5, 7, 10
	k-MPPTI Paralel			
	k-MPPTI NoRSL			

No.	Jenis Algoritme	Jenis Data	Jumlah Data (n)	Jumlah Dimensi (d)
5	k-MPPTI	ANT	1000	2, 3, 5, 7, 10
	k-MPPTI Paralel			
	k-MPPTI NoRSL			
6	k-MPPTI	FC	1000	2, 3, 5, 7, 10
	k-MPPTI Paralel			
	k-MPPTI NoRSL			

5.4 Analisis Hasil Uji Coba

Setelah pengujian dilakukan, selanjutnya adalah menganalisis hasil uji coba. Analisis hasil uji coba dibagi menjadi dua bagian, yaitu hasil uji coba fungsionalitas dan hasil uji coba performa.

5.4.1 Uji Coba Fungsionalitas

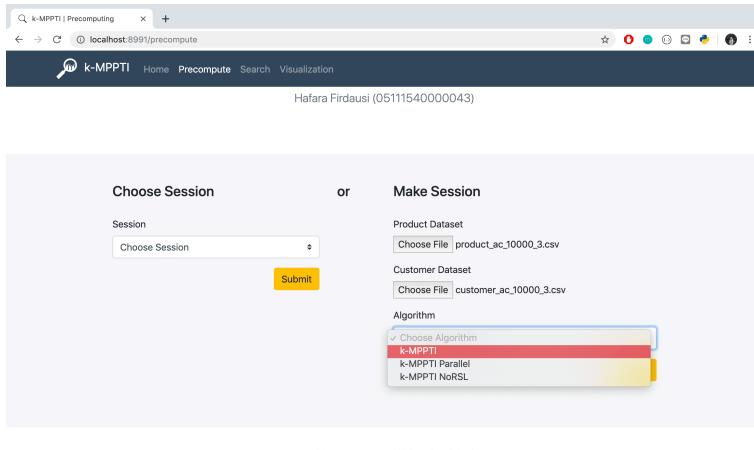
Pengujian fungsionalitas dilakukan sesuai dengan skenario daftar kebutuhan fungsional pada Tabel 5.2 yang hasilnya dapat dilihat pada Tabel 5.4.

Berdasarkan hasil uji coba, aplikasi dapat memenuhi semua kebutuhan fungsional, yakni mengunggah data (Gambar 5.1), melihat informasi (Gambar 5.3) dan pratinjau data (Gambar 5.4), melihat visualisasi data dalam bentuk lini masa (Gambar 5.5), memilih algoritme untuk *data precomputing* (Gambar 5.1), memasukkan kueri pencarian (Gambar 5.6), dan melihat hasil kueri (Gambar 5.6).

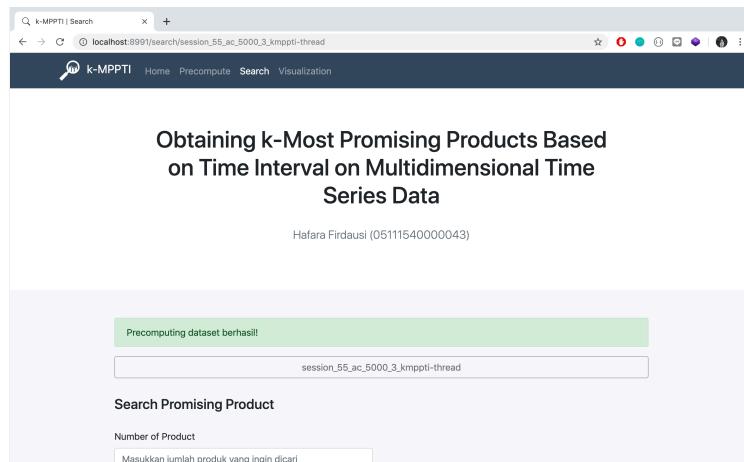
Tabel 5.4 Hasil uji coba fungsionalitas

No.	Deskripsi Kebutuhan	Status
1	Pengguna dapat mengunggah data	Berhasil
2	Pengguna dapat melihat informasi dan pratinjau data	Berhasil
3	Pengguna dapat melihat visualisasi data	Berhasil
4	Pengguna dapat memilih algoritme yang digunakan untuk <i>data precomputing</i>	Berhasil
5	Pengguna dapat memasukkan kueri pencarian	Berhasil
6	Pengguna dapat melihat hasil kueri	Berhasil

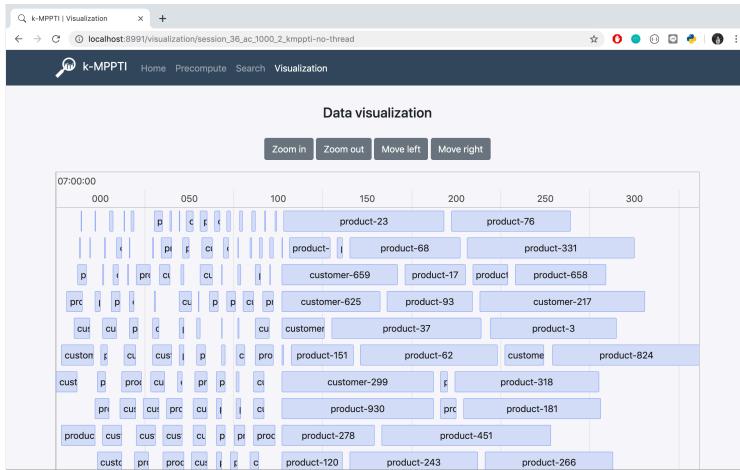
Sebagai tambahan, aplikasi juga memiliki fitur *session* (Gambar 5.7) supaya pengguna dapat melakukan dua kali proses *data precomputing* menggunakan data yang berbeda tanpa saling menumpuk satu sama lain. Pengguna hanya perlu memilih *session* yang diinginkan, kemudian melakukan kueri pencarian atau melihat visualisasi data.



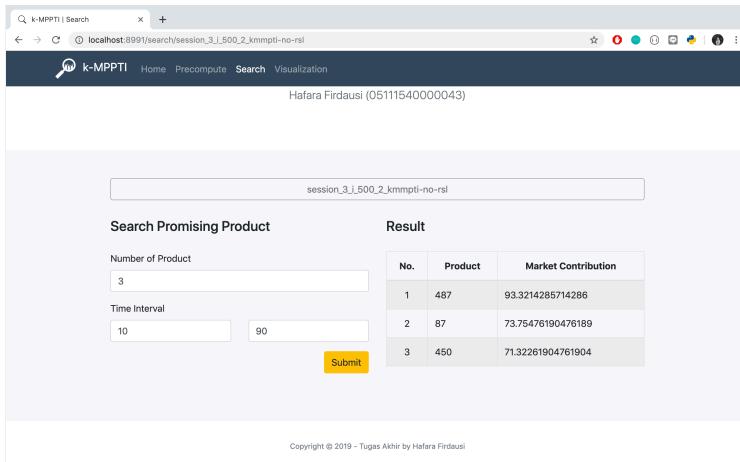
Gambar 5.1 Hasil uji coba: mengunggah data dan memilih algoritme untuk *data precomputing*



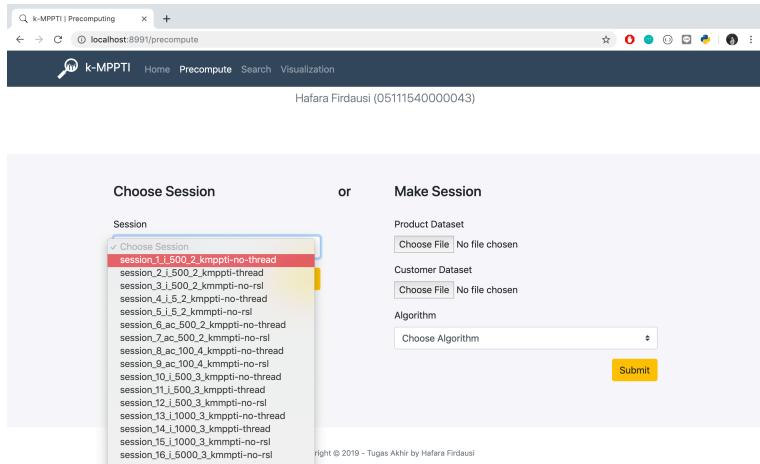
Gambar 5.2 Hasil uji coba: proses *data precomputing* berhasil



Gambar 5.5 Hasil uji coba: melihat visualisasi data berupa lini masa



Gambar 5.6 Hasil uji coba: memasukkan kueri pencarian dan melihat hasil kueri



Gambar 5.7 Hasil uji coba: memilih *session*

5.4.2 Uji Coba Performa

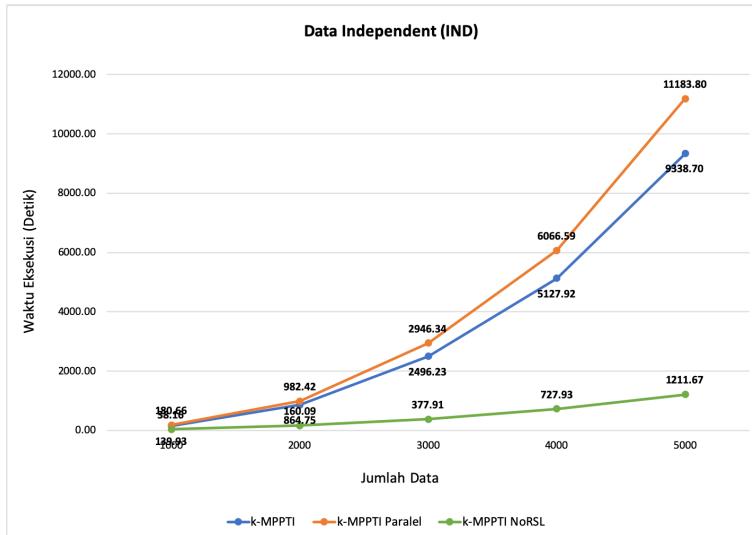
Pengujian performa dilakukan sesuai dengan skenario daftar kebutuhan fungsional pada Tabel 5.3 yang hasilnya dianalisis sebagai berikut.

5.4.2.1 Pengaruh Jumlah Data Terhadap Performa Algoritme

Berdasarkan hasil pengujian menggunakan skenario ke-1, 2, dan 3 pada Tabel 5.3, dapat diketahui pengaruh perubahan jumlah data terhadap performa masing-masing algoritme. Pengujian ini dilakukan pada ketiga data untuk melihat apakah ada perbedaan yang signifikan antar ketiganya.

Hasil pengujian pada Grafik 5.8 dan 5.9 menunjukkan bahwa perubahan jumlah data memiliki pengaruh yang sangat signifikan terhadap waktu komputasi. Hal ini terjadi karena semakin banyak data, maka *event* yang harus diproses menjadi empat kali lebih

banyak (ada dua data, yakni data produk dan pelanggan). Sebagai contoh, ada 2000 data produk dan pelanggan yang dimasukkan, maka ada 8000 *event* yang harus diproses. Semakin banyak *event* yang diproses, maka semakin lama pula komputasinya.

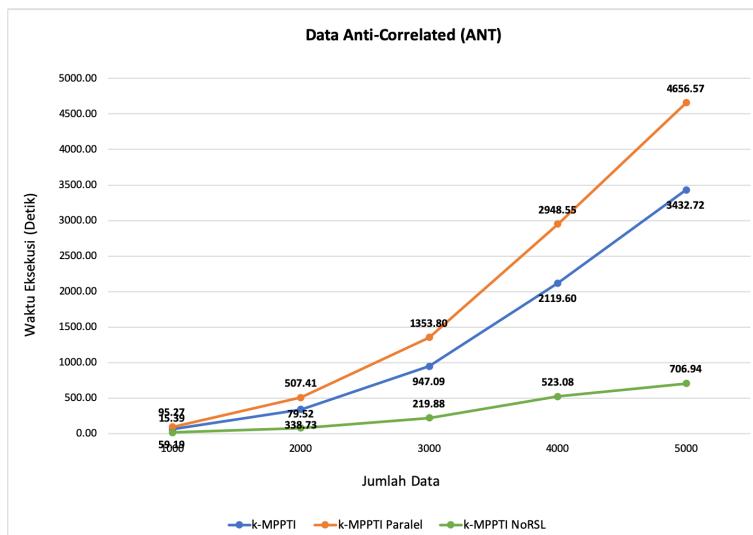


Gambar 5.8 Grafik pengaruh jumlah data terhadap waktu komputasi algoritme pada data *independent* (IND)

Jika melihat perbandingan antar data, waktu komputasi pada data *anti-correlated* (ANT) relatif lebih cepat dibandingkan komputasi data *independent* (IND) pada semua algoritme. Namun, data IND memiliki perubahan waktu komputasi yang relatif lebih besar dibandingkan data ANT untuk setiap perubahan jumlah data. Jika diperhatikan lebih dalam lagi, semakin besar jumlah dimensi, maka selisih waktu komputasi antara data IND dan ANT juga semakin besar.

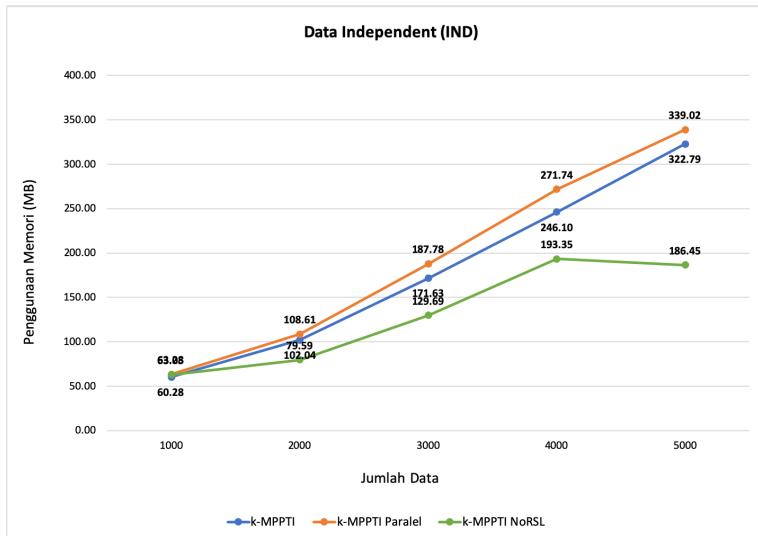
Di sisi lain, jika melihat perbandingan performa algoritme,

algoritme k-MPPTI NoRSL memiliki waktu komputasi paling cepat pada semua jenis data dan algoritme k-MPPTI Paralel memiliki waktu komputasi paling lama pada semua jenis data, walaupun tidak terlalu berbeda jauh dengan waktu komputasi k-MPPTI biasa. Padahal hasil kueri ketiganya mirip dan tidak berbeda jauh (dijelaskan pada sub bagian 5.4.2.3).



Gambar 5.9 Grafik pengaruh jumlah data terhadap waktu komputasi algoritme pada data *anti-correlated* (ANT)

Algoritme k-MPPTI paralel memiliki waktu komputasi paling lama dibandingkan kedua algoritme yang lain karena teknik komputasi paralel diimplementasikan hanya menggunakan satu *resource* menggunakan *thread*, sehingga lebih memberatkan CPU karena harus membagi *resource*-nya. Lambatnya waktu eksekusi berbanding lurus dengan jumlah data karena *thread* dibentuk sejumlah data pelanggan, sehingga semakin banyak data, maka semakin banyak pula *thread* yang terbentuk.



Gambar 5.10 Grafik pengaruh jumlah data terhadap penggunaan memori algoritme pada data *independent* (IND)



Gambar 5.11 Grafik pengaruh jumlah data terhadap penggunaan memori algoritme pada data *anti-correlated* (ANT)

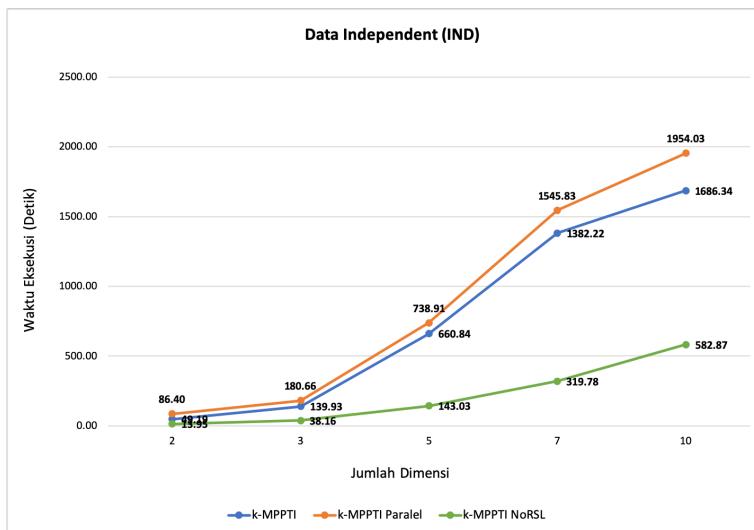
Hasil uji coba pengaruh perubahan jumlah data terhadap penggunaan memori algoritme ditunjukkan oleh Grafik 5.10 dan 5.11. Dalam grafik tersebut, perubahan memori terjadi secara signifikan pada kedua data IND dan ANT. Perubahan memori cenderung berbanding lurus dengan perubahan jumlah data, namun menunjukkan ketidakstabilan pada kedua data. Sama seperti sebelumnya, penggunaan memori pada data IND cenderung lebih banyak dibandingkan data ANT.

5.4.2.2 Pengaruh Jumlah Dimensi Data Terhadap Performa Algoritme

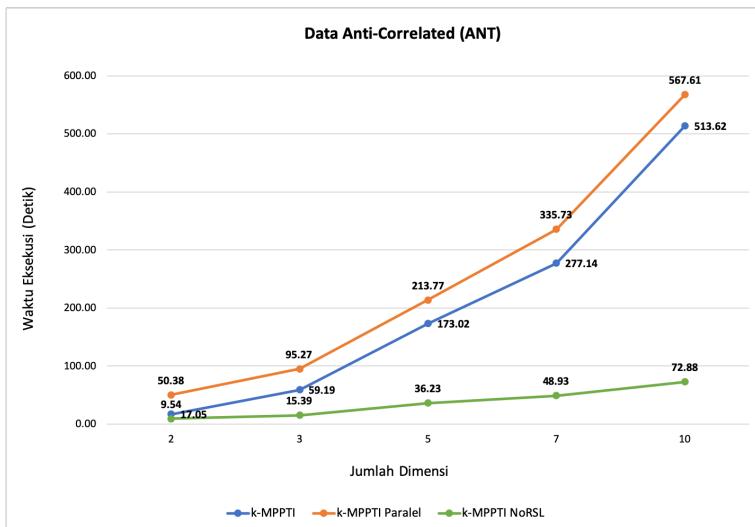
Berdasarkan hasil pengujian menggunakan skenario ke-4, 5, dan 6 pada Tabel 5.3, dapat diketahui pengaruh perubahan jumlah dimensi data terhadap performa masing-masing algoritme.

Pengujian ini dilakukan pada ketiga data untuk melihat apakah ada perbedaan yang signifikan antar ketiganya.

Hasil pengujian pada Grafik 5.12 dan 5.13 menunjukkan bahwa perubahan jumlah dimensi data memiliki pengaruh yang sangat signifikan terhadap waktu komputasi. Hal ini dikarenakan jumlah dimensi sangat berpengaruh terhadap penentuan dominansi antar data, sehingga nilai setiap dimensi harus dibandingkan satu per-satu. Selain itu, iterasi setiap dimensi dilakukan dua kali. Pertama untuk menghitung selisih (selanjutnya disimpan dalam struktur data supaya dapat digunakan kembali tanpa komputasi ulang). Kedua untuk mengecek dominansi antar produk.

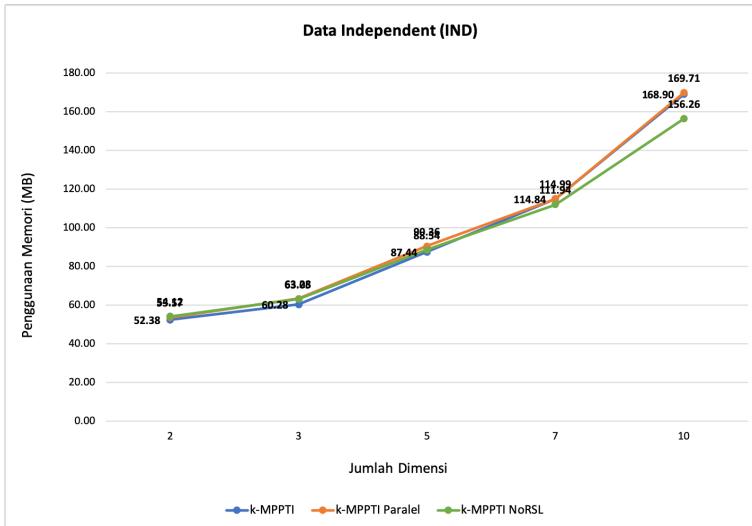


Gambar 5.12 Grafik pengaruh jumlah dimensi data terhadap waktu komputasi algoritme pada data *independent* (IND)



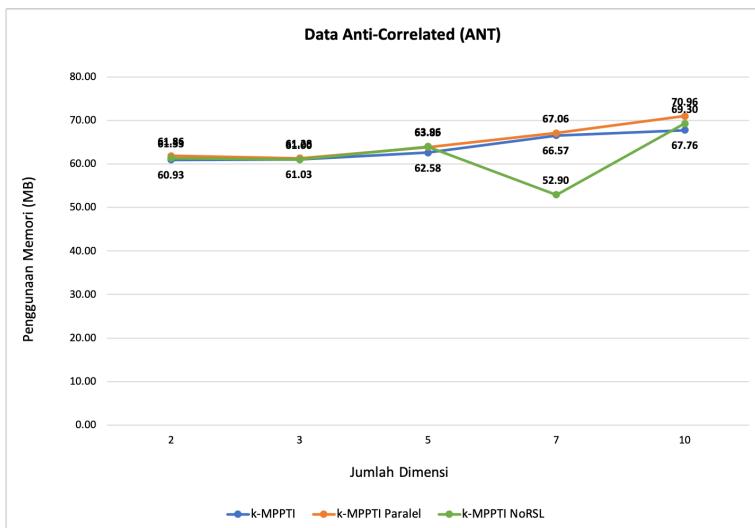
Gambar 5.13 Grafik pengaruh jumlah dimensi data terhadap waktu komputasi algoritme pada data *anti-correlated* (ANT)

Jika melihat perbandingan antar data, waktu komputasi pada data *anti-correlated* (ANT) relatif lebih cepat dibandingkan komputasi data *independent* (IND) pada semua algoritme. Namun, data IND memiliki perubahan waktu komputasi yang relatif lebih besar dibandingkan data ANT untuk setiap perubahan jumlah dimensi. Hal ini disebabkan karena nilai atribut pada data IND tidak memiliki keterkaitan satu sama lain, sehingga membutuhkan pemrosesan lebih lama. Jika diperhatikan lebih dalam lagi, semakin besar jumlah dimensi, maka selisih waktu komputasi antara data IND dan ANT juga semakin besar.



Gambar 5.14 Grafik pengaruh jumlah dimensi data terhadap penggunaan memori algoritme pada data *independent* (IND)

Di lain sisi, jika melihat perbandingan performa algoritme, algoritme k-MPPTI NoRSL memiliki waktu komputasi paling cepat pada semua jenis data dan algoritme k-MPPTI Paralel memiliki waktu komputasi paling lama pada semua jenis data, walaupun tidak terlalu berbeda jauh dengan waktu komputasi k-MPPTI biasa. Padahal hasil kueri ketiganya mirip dan tidak berbeda jauh (dijelaskan pada sub bagian 5.4.2.3).



Gambar 5.15 Grafik pengaruh jumlah dimensi data terhadap penggunaan memori algoritme pada data *anti-correlated* (ANT)

Hasil uji coba pengaruh perubahan jumlah dimensi terhadap penggunaan memori algoritme ditunjukkan oleh Grafik 5.14 dan 5.15. Dalam grafik tersebut, perubahan memori terjadi secara signifikan pada data IND, namun tidak pada data ANT. Pada data ANT, ada satu hasil yang menunjukkan ketidakstabilan, yakni pada algoritme k-MPPTI NoRSL ketika memproses data dengan tujuh dimensi. Sama seperti sebelumnya, penggunaan memori pada data IND cenderung lebih banyak dibandingkan data ANT.

5.4.2.3 Akurasi Hasil Kueri

Salah satu kekurangan dalam penelitian ini adalah tidak adanya uji coba akurasi terhadap hasil kueri masing-masing algoritme karena tidak ada acuan yang dapat memastikan algoritme yang diimplementasikan benar atau salah. Sehingga, hal yang dapat dilakukan adalah dengan membandingkan hasil kueri antar algoritme. Perbandingan hasil kueri pada masing-masing data ditunjukkan pada Gambar 5.16, 5.18, dan 5.17.

Hasil kueri kedua algoritme pada data *independent* (IND) dengan jumlah 1000 data dengan 5 dimensi 100% sama, namun memiliki hasil perhitungan kontribusi pasar yang berbeda. Hasil kueri kedua algoritme pada data *anti-correlated* (ANT) dengan jumlah 4000 data dengan 3 dimensi memiliki anggota yang sama, namun peringkat dan hasil perhitungan kontribusi pasarnya yang berbeda. Sedangkan hasil kueri kedua algoritme pada data *forest cover type* dengan jumlah 1000 data dengan 3 dimensi memiliki perbedaan yang signifikan karena memiliki satu anggota yang berbeda dan hasil peringkat satu pada komputasi k-MPPTI tidak menjadi hasil pada komputasi k-MPPTI NoRSL.

Search Promising Product

No.	Product	Mark
1	986	134.1€
2	487	132.8€
3	571	132.2€
4	597	125.9€
5	222	123.5€

Number of Product: 5
Time Interval: 10 - 200

Result

Copyright © 2019 - Tugas Akhir by Hafara Firdausi

Gambar 5.16 Pengujian akurasi hasil kueri pada data *independent* (IND)

Product

No.	Product	Mark
1	816	200.0
2	1964	160.7€
3	356	159.4€
4	2807	148.6€
5	212	147.5
6	2315	146.6€
7	2488	145.0
8	275	138.1€
9	2371	137.0
10	3343	136.6€

Number of Product: 10
Time Interval: 10 - 100

Result

Copyright © 2019 - Tugas Akhir by Hafara Firdausi

Gambar 5.17 Pengujian akurasi hasil kueri pada data *anti-correlated* (ANT)

The image displays two side-by-side screenshots of a web application interface titled "k-MPPTI". Both screens show a search form and a resulting table.

Left Screen (Search Session 101):

Search Promising Product

No.	Product	Mark
1	75	4859.
2	275	4162.
3	101	3446.
4	88	2939.
5	858	2541.

Number of Product: 5
Time Interval: 10 1000
Submit

Copyright © 2019 - Tugas Akhir by Hafara Firdausi

Right Screen (Search Session 102):

Search Promising Product

No.	Product	Mark
1	57	4379.
2	75	3850.
3	275	3265.
4	101	2987.
5	88	2603.

Number of Product: 5
Time Interval: 1000
Submit

Copyright © 2019 - Tugas Akhir by Hafara Firdausi

Gambar 5.18 Pengujian akurasi hasil kueri pada data *Forest Cover Type* (FC)

BAB VI

KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan dan saran dari hasil uji coba yang telah dilakukan.

6.1 Kesimpulan

Dari proses perancangan hingga uji coba, dapat diambil beberapa kesimpulan sebagai berikut:

1. Desain dan implementasi struktur data dan algoritme untuk menjawab kueri *k-Most Promising Products* (*k*-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu adalah menggunakan struktur data *array*, *queue*, dan *dictionary* untuk pengindeksan data, penyimpanan *events*, dan penyimpanan skor kontribusi pasar. Selain itu, algoritme menggunakan dua jenis komputasi *skyline*, yaitu *dynamic skyline* dan *reverse skyline*
2. Biaya komputasi pada algoritme *k*-MPPTI yang tidak menggunakan komputasi *reverse skyline* jauh lebih baik dibandingkan algoritme *k*-MPPTI yang menggunakan komputasi *reverse skyline* dari sisi waktu komputasi dan penggunaan memori. Komputasi algoritme *k*-MPPTI NoRSL lima kali lebih cepat dibandingkan *k*-MPPTI biasa. Sedangkan dari sisi penggunaan memori, algoritme *k*-MPPTI NoRSL hanya satu kali lebih hemat dibandingkan algoritme *k*-MPPTI biasa. Hal ini menandakan bahwa komputasi *reverse skyline* kurang cocok untuk diimplementasikan dalam penyelesaian masalah ini.

3. Strategi yang optimal untuk meningkatkan efisiensi komputasi *k-Most Promising Products* (k-MPP) berbasis interval waktu pada data multidimensi dengan serial waktu adalah dengan menggunakan teknik komputasi paralel yang tidak diimplementasikan menggunakan satu *resource* saja karena hanya akan memberatkan kinerja CPU. Jika hanya memiliki satu *resource*, lebih efektif jika menggunakan algoritme k-MPPTI tanpa menggunakan teknik komputasi paralel.

6.2 Saran

Berikut beberapa saran terkait pengembangan lebih lanjut:

1. Sebaiknya dilakukan pengujian menggunakan jumlah data yang lebih banyak dan dilakukan beberapa kali untuk hasil analisis yang lebih akurat supaya tidak terjadi kesalahan analisis.
2. Menemukan metode yang tepat untuk menguji akurasi hasil kueri antar algoritme supaya dapat diketahui mana algoritme yang bekerja lebih tepat.
3. Menemukan struktur data yang tepat sebagai pengembangan dari algoritme ini supaya penggunaan memori menjadi lebih sedikit. Selain itu, kompleksitas algoritma juga harus lebih diperhatikan lagi.
4. Sebagai pengembangan, algoritme k-MPPTI dapat dirancang sedemikian rupa supaya dapat memproses data *streaming*.

5. Komputasi *reverse skyline* membuat penghitungan kontribusi pasar menjadi lebih efisien pada komputasi k-MPP [1], namun, kurang cocok jika digunakan untuk menyelesaikan permasalahan yang diangkat pada Tugas Akhir ini karena hanya akan memberatkan komputasi. Diperlukan penelitian dan percobaan lebih jauh untuk mencari metode perhitungan kontribusi pasar yang lebih efisien.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

- [1] M. S. Islam and C. Liu, "Know Your Customer: Computing K-Most Promising Products," *The VLDB Journal*, pp. 545–570, 2016.
- [2] D. Papadias, Y. Tao, G. Fu and B. Seeger, "Progressive Skyline Computation in Database Systems," *ACM Transactions on Database Systems*, Vol. 30, No. 1, pp. 41–82, 2005.
- [3] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," *VLDB Endowment*, pp. 291-302, 2007.
- [4] B. Jiang and J. Pei, "Online Interval Skyline Queries on Time Series," *IEEE International Conference on Data Engineering*, pp. 1036-1047, 2009.
- [5] M. Golfarelli and S. Rizzi, "Introduction to Data Warehousing," in *Data Warehouse Design: Modern Principles and Methodologies*, New York: McGraw-Hill, 2009, pp. 1-42.
- [6] S. Borzsonyi, D. Kossmann and K. Stocker, "The Skyline Operator," In: *ICDE*, pp. 421-430, 2001.
- [7] L. Zou, L. Chen, M. T. Özsü and D. Zhao, "Dynamic Skyline Queries in Large Graphs," *DASFAA '10 Proceedings of the 15th International Conference on Database Systems for Advanced Applications - Volume Part II*, pp. 62-78, 2010.
- [8] X. Wu, Y. Tao, R. C.-W. Wong, L. Ding and J. X. Yu, "Finding the Influence Set through Skylines," *EDBT*, pp. 1030-1041, 2009.
- [9] G.S. Almasi and A. Gottlieb, *Highly Parallel Computing*. Redwood City, CA: Benjamin-Cummings Publishers, 1989.

- [10] Merriam-webster.com. "Definition of DATA". [Online]. Available: <https://www.merriam-webster.com/dictionary/data>. [Accessed 13 Mei 2019]
- [11] Python.org. "What is Python? Executive Summary". [Online]. Available: <https://www.python.org/doc/essays/blurb/>. [Accessed 18 Mei 2019].
- [12] Flask.pocoo.org. "Welcome to Flask". [Online]. Available: <http://flask.pocoo.org/docs/1.0/>. [Accessed 18 Mei 2019]
- [13] Visjs.org. "Vis.js". [Online]. Available: <https://visjs.org/>. [Accessed 9 Juni 2019]
- [14] J. A. Blackard, D. J. Jean and C. W. Anderson, "UCI Machine Learning Repositories," 1 Agustus 1998. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/covtype>. [Accessed 9 Juni 2018].

LAMPIRAN A

KODE SUMBER

```
1  """
2  Event Queue Handling
3  """
4
5  from operator import itemgetter
6
7  class EventQueue:
8      def __init__(self):
9          self.events = []
10
11     def is_empty(self):
12         return self.events == []
13
14     def enqueue(self, timestamp, owner, owner_id, action):
15         self.event = [timestamp, owner, owner_id, action]
16         self.events.append(self.event)
17
18     def dequeue(self):
19         event = self.events.pop()
20         return event
21
22     def sort_queue(self):
23         self.events = sorted(self.events, key=itemgetter(0,3,1,2), reverse=True)
24
25     def get_total_queue(self):
26         return len(self.events)
27
28     def get_max_timestamp(self):
29         return self.events[0][0]
30
31     def get_min_timestamp(self):
32         return self.events[len(self.events)-1][0]
```

Kode Sumber 1.1 Kode sumber kelas *EventQueue*

```
1  """
2  Pandora Box Handling
3  """
4
5  import csv
6  from app import app
7
8  class PandoraBox:
9      def __init__(self, total_prod=None, max_ts=None, prod_id=None):
10          try:
11              self.box = [[0 for col in range(0, max_ts)]
12                         for row in range(0, total_prod)]
13          except:
14              self.box = []
```

```

15
16     def update(self, dsl):
17         for p in dsl:
18             if 'last_ts' in dsl[p] and 'last_prob' in dsl[p]:
19                 if dsl[p]['ts'] > dsl[p]['last_ts']:
20                     self.update_score(p, dsl[p]['ts'], dsl[p]['last_ts'],
21                                       dsl[p]['prob'], dsl[p]['last_prob'])
22             else:
23                 self.box[p][dsl[p]['ts']] += dsl[p]['prob']
24
25     def update_score(self, p, ts, last_ts, prob, last_prob):
26         for i in range(last_ts + 1, ts):
27             self.box[p][i] += last_prob
28         self.box[p][ts] += prob
29
30     def insert_score(self, prod_score):
31         self.box.append(prod_score)
32
33     def get_score(self, prod_id, ts_start, ts_end):
34         total_score = 0
35         try:
36             for i in range(ts_start, ts_end + 1):
37                 total_score += self.box[prod_id][i]
38         except:
39             print('out_of_list')
40         finally:
41             return total_score
42
43     def export_csv(self, pandora_path):
44         with open(pandora_path + '/pandora_box.csv', 'w') as csvFile:
45             writer = csv.writer(csvFile)
46             writer.writerows(self.box)
47         csvFile.close()

```

Kode Sumber 1.2 Kode sumber kelas *PandoraBox*

```

1 """
2 Reverse Skyline Computation Handling
3 """
4
5 class ReverseSkyline:
6     def __init__(self, product, customer, dim):
7         self.product = product['data']
8         self.customer = customer['data']
9         self.product_active = product['active']
10        self.customer_active = customer['active']
11        self.dim = dim
12        self.define_orthant()
13
14    def compute(self, pid):
15        self.my_id = pid
16        self.my_value = self.product[pid]['value']
17        self.find_midpoint_skyline()
18        return self.find_reverse_skyline()
19
20    def define_orthant(self):
21        self.orthant = {}
22        for i in range(2**self.dim):
23            id = format(i, '#0{}b'.format(self.dim + 2))[2:]
24            self.orthant[id] = {}
25

```

```

26     def get_orthant_area(self, product_val):
27         res = []
28         for i in range(self.dim):
29             if product_val[i] <= self.my_value[i]:
30                 res.append('0')
31             else:
32                 res.append('1')
33         res = ''.join(res)
34         return res
35
36     def find_midpoint_skyline(self):
37         for pid in self.product_active:
38             if pid != self.my_id:
39                 midpoint = self.calc_midpoint(self.my_value, self.product[pid]['value'])
40                 area = self.get_orthant_area(self.product[pid]['value'])
41                 if not self.orthant[area]:
42                     self.orthant[area][pid] = midpoint
43                 else:
44                     self.orthant[area] = self.update_midskyline(self.orthant[area],
45                                                       midpoint, pid)
45
46
47     def find_reverse_skyline(self):
48         rsl = []
49         for cid in self.customer_active:
50             area = self.get_orthant_area(self.customer[cid]['value'])
51             if not self.orthant[area]:
52                 rsl.append(cid)
53             else:
54                 dom = 0
55                 for i in self.orthant[area]:
56                     if self.is_dominating(self.orthant[area][i],
57                                           self.customer[cid]['value']):
58                         dom += 1
59                 if dom == 0:
60                     rsl.append(cid)
61         return rsl
62
63     def update_midskyline(self, msl, cand, cand_id):
64         res = {}
65         for i in msl:
66             if self.is_dominating(cand, msl[i]):
67                 res[cand_id] = cand
68             else:
69                 res[i] = msl[i]
70                 if not self.is_dominating(msl[i], cand):
71                     res[cand_id] = cand
72         return res
73
74     def is_dominating(self, subject, target):
75         dominating = 0
76         dominated = 0
77         for i in range(0, self.dim):
78             subj_diff = abs(self.my_value[i] - subject[i])
79             target_diff = abs(self.my_value[i] - target[i])
80             if subj_diff == target_diff:
81                 continue
82             elif subj_diff < target_diff:
83                 dominating += 1
84             elif subj_diff > target_diff:
85                 dominated += 1
86         if dominated == 0 and dominating >= 1:
87             return True
88         else:
89             return False

```

```

90
91     def calc_midpoint(self, val1, val2):
92         res = []
93         for i in range(self.dim):
94             res.append((val1[i] + val2[i])/2)
95
96         return res

```

Kode Sumber 1.3 Kode sumber kelas *ReverseSkyline*

```

1 """
2 Dynamic Skyline Computation Handling
3 """
4
5 class DynamicSkyline:
6     def __init__(self, product, customer, dim):
7         self.product = product['data']
8         self.customer = customer['data']
9         self.product_active = product['active']
10        self.dim = dim
11
12    def product_out(self, cust_id, ts, prod_id):
13        cust_value = self.customer[cust_id]['value']
14        active_child = None
15        try:
16            if 'dominating' in self.customer[cust_id]['dsl'][prod_id]:
17                active_child = self.find_active_child(
18                    self.customer[cust_id]['dsl'][prod_id])
19            del self.customer[cust_id]['dsl'][prod_id]
20            if active_child:
21                self.customer[cust_id]['dsl'] = self.update_dsl(active_child,
22                                                               cust_id, cust_value)
23            self.update(cust_id, ts)
24        except KeyError:
25            pass
26
27    def product_in(self, cust_id, ts, prod_id):
28        cust_value = self.customer[cust_id]['value']
29        self.customer[cust_id]['dsl'] = self.update_dsl(prod_id, cust_id,
30                                                       cust_value)
31        self.update(cust_id, ts)
32
33    def customer_in(self, cust_id, ts):
34        cust_value = self.customer[cust_id]['value']
35        prod_id = self.product_active
36        self.customer[cust_id]['dsl'] = self.init_dsl(prod_id, cust_value)
37        self.update(cust_id, ts)
38
39    def init_dsl(self, cand, cust_value):
40        dsl = {}
41        for prod_id in cand:
42            dsl[prod_id] = self.calc_diff(cust_value, self.product[prod_id]['value'])
43        dsl_id = sorted(dsl, key=lambda x: dsl[x]['diff'])
44        for i in range(len(dsl_id)):
45            if dsl_id[i] is not None:
46                for j in range(i + 1, len(dsl_id)):
47                    if dsl_id[j] is not None:
48                        if self.is_dominating(dsl[dsl_id[i]], dsl[dsl_id[j]]):
49                            dsl, dsl_id[j] = self.update_cand(dsl, dsl_id[i], dsl_id[j])
50                        elif self.is_dominated(dsl[dsl_id[j]], dsl[dsl_id[i]]):
51                            dsl, dsl_id[i] = self.update_cand(dsl, dsl_id[j], dsl_id[i])
52                            break

```

```

53     return dsl
54
55 def update_dsl(self, cand, cust_id, cust_value):
56     dsl = {}
57     for prod_id in cand:
58         dsl[prod_id] = self.calc_diff(cust_value, self.product[prod_id]['value'])
59     if 'dsl' in self.customer[cust_id]:
60         dsl.update(self.customer[cust_id]['dsl'])
61     dsl_id = sorted(dsl, key=lambda x: dsl[x]['diff'])
62     for j in range(len(cand)):
63         if cand[j] in dsl_id:
64             index = dsl_id.index(cand[j])
65             for i in range(len(dsl_id)):
66                 if dsl_id[index] is not None and dsl_id[i] is not None:
67                     if i < index:
68                         if self.is_dominating(dsl[dsl_id[i]], dsl[dsl_id[index]]):
69                             dsl, dsl_id[index] = self.update_cand(dsl, dsl_id[i],
70                                                               dsl_id[index])
71                         break
72                     elif i > index:
73                         if self.is_dominating(dsl[dsl_id[index]], dsl[dsl_id[i]]):
74                             dsl, dsl_id[i] = self.update_cand(dsl, dsl_id[index], dsl_id[i])
75     return dsl
76
77 def update_cand(self, dsl, psubj_id, pobj_id):
78     del dsl[pobj_id]
79     if 'dominating' in dsl[psubj_id]:
80         if pobj_id not in dsl[psubj_id]['dominating']:
81             dsl[psubj_id]['dominating'].append(pobj_id)
82     else:
83         dsl[psubj_id]['dominating'] = [pobj_id]
84     pobj_id = None
85     return dsl, pobj_id
86
87 def is_dominating(self, psubj, pobj):
88     dominating = 0
89     dominated = 0
90     for i in range(0, self.dim):
91         if psubj['diff'][i] < pobj['diff'][i]:
92             dominating += 1
93         elif psubj['diff'][i] > pobj['diff'][i]:
94             dominated += 1
95         else:
96             continue
97     if dominated == 0 and dominating >= 1:
98         return True
99     else:
100        return False
101
102 def calc_probability(self, dsl_result):
103     try:
104         return 1.0/len(dsl_result.keys())
105     except:
106         return 0
107
108 def calc_diff(self, val1, val2):
109     res = {}
110     res['diff'] = []
111     for i in range(0, self.dim):
112         res['diff'].append(abs(val1[i] - val2[i]))
113     return res
114
115 def find_active_child(self, dsl_result):
116     res = []

```

```

17     for dom in dsl_result['dominating']:
18         if dom in self.product_active:
19             res.append(dom)
20     return res
21
22 def update(self, cust_id, ts):
23     for prod_id in self.customer[cust_id]['dsl']:
24         try:
25             self.customer[cust_id]['dsl'][prod_id]['ts'] = ts
26             self.customer[cust_id]['dsl'][prod_id]['prob'] = self.calc_probability(
27                                         self.customer[cust_id]['dsl'])
28         except KeyError:
29             pass
30
31 def update_history(self, cust_id):
32     for prod_id in self.customer[cust_id]['dsl']:
33         self.customer[cust_id]['dsl'][prod_id]['last_ts'] =
34             self.customer[cust_id]['dsl'][prod_id]['ts']
35         self.customer[cust_id]['dsl'][prod_id]['last_prob'] =
36             self.customer[cust_id]['dsl'][prod_id]['prob']

```

Kode Sumber 1.4 Kode sumber kelas *DynamicSkyline*

```

1 import csv, sys, threading, os, time, cProfile, json
2 import numpy as np
3 from app import app
4 from app.src.kmppti.pandora_box import PandoraBox
5 from app.src.kmppti.event_queue import EventQueue
6 from app.src.kmppti.reverse_skyline import ReverseSkyline
7 from app.src.kmppti.dynamic_skyline import DynamicSkyline
8 from app.src.kmppti.logger import Logger
9
10 """
11 Data Handling
12 """
13 def input_csv(dataset, event_queue):
14     data = {}
15     for i in range(len(dataset)):
16         if i == 0:
17             data['product'] = {}
18             data['product']['data'], data['product']['attr'] =
19                 data_indexing(i, dataset[i], event_queue)
20         else:
21             data['customer'] = {}
22             data['customer']['data'], data['customer']['attr'] =
23                 data_indexing(i, dataset[i], event_queue)
24     return data
25
26 def data_indexing(data_id, data_file, event_queue):
27     data = {}
28     with open(data_file, 'r') as csv_file:
29         csv_reader = csv.DictReader(csv_file, delimiter=',')
30         for row in csv_reader:
31             attr = csv_reader.fieldnames
32             col_cnt = len(attr)
33             id = int(row[attr[0]])
34             data[id] = {}
35             data[id]['label'] = row[attr[1]]
36             data[id]['ts_in'] = int(row[attr[2]])
37             data[id]['ts_out'] = int(row[attr[3]])
38             data[id]['value'] = []

```

```

39         for i in range(4, col_cnt):
40             data[id]['value'].append(int(row[attr[i]]))
41         for i in range(0, 2):
42             event_queue.enqueue(data[id]['ts_in'])
43             if i == 0 else data[id]['ts_out'], data_id, id, i)
44     return data, attr
45
46 def get_maxmin_value(product, customer):
47     all_val = []
48     for pid in product.values():
49         all_val.append(pid['value'])
50     for cid in customer.values():
51         all_val.append(cid['value'])
52     return np.max(all_val), np.min(all_val)
53
54 def get_metadata(event_queue, dataset, data):
55     metadata = [
56         event_queue.get_max_timestamp(),
57         event_queue.get_min_timestamp(),
58         get_maxmin_value(data['product']['data'], data['customer']['data']),
59         len(data['product']['attr']) - 4,
60         len(data['product']['data']),
61         len(data['customer']['data']),
62         data['product']['attr'],
63         data['customer']['attr'],
64         dataset[0].split('/')[-len(dataset[0].split('/'))-1],
65         dataset[1].split('/')[-len(dataset[1].split('/'))-1],
66     ]
67     return metadata
68
69 """
70 Event Handling
71 """
72 def update_pbox(cust_id, ts, dsl, pandora_box, data):
73     dsl.update(cust_id, ts)
74     pandora_box.update(data['customer']['data'][cust_id]['dsl'])
75     dsl.update_history(cust_id)
76
77 def compute_dsl(cust_id, ts, act, data_id, dsl):
78     if act == 0:
79         dsl.product_in(cust_id, ts, [data_id])
80     else:
81         dsl.product_out(cust_id, ts, data_id)
82         dsl.update_history(cust_id)
83
84 def recheck(cust_id, ts, dsl, data):
85     cand = set(data['product']['active']) -
86             set(data['customer']['data'][cust_id]['dsl'].keys())
87     if cand:
88         cands = [c for c in cand]
89         dsl.product_in(cust_id, ts, cands)
90
91 """
92 Session Handling
93 """
94 def make_session(data_info, algorithm, metadata):
95     session_file = app.config['SESSION_DIR']
96     if not os.path.exists(session_file):
97         os.mkdir(session_file)
98     try:
99         with open(session_file + '/session.json') as json_file:
100             data = json.load(json_file)
101             num = data['num']
102     except:

```

```

03     data = {}
04     num = 0
05     session_name = 'session_' + str(num+1) + '_' + data_info[1] + ' ' +
06                     data_info[2] + '_' + data_info[3].split('.')[0] + 
07                     '_' + algorithm
08     my_data = {}
09     my_data[session_name] = {
10         'pbox_path': session_file + '/' + session_name + '/pandora_box.csv',
11         'max_ts': metadata[0],
12         'min_ts': metadata[1],
13         'max_val': int(metadata[2][0]),
14         'min_val': int(metadata[2][1]),
15         'dim': metadata[3],
16         'total_prod': metadata[4],
17         'total_cust': metadata[5],
18         'attr_prod': metadata[6],
19         'attr_cust': metadata[7],
20         'data_type': data_info[1],
21         'product_data': metadata[8],
22         'customer_data': metadata[9]
23     }
24     if 'session' in data:
25         data['session'].update(my_data)
26     else:
27         data['session'] = my_data
28     data['num'] = num + 1
29     with open(session_file + '/session.json', 'w') as json_file:
30         json.dump(data, json_file)
31     return session_name, session_file
32
33 """
34 Precomputing
35 """
36 def kmppti_precompute(dataset):
37     algorithm = 'kmppti-no-thread'
38     logger = Logger(algorithm, 'precomputing')
39     logger.set_time(0)
40     logger.set_data_info(dataset[0].split('_'))
41     print ('Precompute started')
42
43     event_queue = EventQueue()
44     data = {}
45     data = input_csv(dataset, event_queue)
46     event_queue.sort_queue()
47     metadata = get_metadata(event_queue, dataset, data)
48
49     data['product']['active'] = []
50     data['customer']['active'] = []
51
52     pandora_box = PandoraBox(metadata[4] + 1, metadata[0] + 1,
53                             data['product']['data'])
54     rsl = ReverseSkyline(data['product'], data['customer'], metadata[3])
55     dsl = DynamicSkyline(data['product'], data['customer'], metadata[3])
56
57     while not event_queue.is_empty():
58         event = event_queue.dequeue()
59         if event[1] == 0:
60             if event[3] == 0:
61                 data['product']['active'].append(event[2])
62                 rsl_result = rsl.compute(event[2])
63                 for cust_id in rsl_result:
64                     compute_dsl(cust_id, event[0], event[3], event[2], dsl)
65                 for cust_id in data['customer']['active']:
66                     if 'dsl' in data['customer']['data'][cust_id].keys():

```

```

167         update_pbox(cust_id, event[0], dsl, pandora_box, data)
168
169     elif event[3] == 1:
170         for cust_id in data['customer']['active']:
171             if 'dsl' in data['customer']['data'][cust_id].keys():
172                 update_pbox(cust_id, event[0], dsl, pandora_box, data)
173             rsl_result = rsl.compute(event[2])
174             for cust_id in rsl_result:
175                 compute_dsl(cust_id, event[0], event[3], event[2], dsl)
176             data['product']['active'].remove(event[2])
177             for cust_id in rsl_result:
178                 recheck(cust_id, event[0], dsl, data)
179
180     elif event[1] == 1:
181         if event[3] == 0:
182             data['customer']['active'].append(event[2])
183             dsl.customer_in(event[2], event[0])
184             if 'dsl' in data['customer']['data'][event[2]].keys():
185                 pandora_box.update(data['customer']['data'][event[2]]['dsl'])
186                 dsl.update_history(event[2])
187
188     elif event[3] == 1:
189         dsl.update(event[2], event[0])
190         if 'dsl' in data['customer']['data'][event[2]].keys():
191             pandora_box.update(data['customer']['data'][event[2]]['dsl'])
192             dsl.update_history(event[2])
193             data['customer']['active'].remove(event[2])
194
195     session_name, session_file = make_session(dataset[0].split('_'),
196                                              algorithm, metadata)
197     pandora_path = session_file + '/' + session_name
198     os.mkdir(pandora_path)
199     pandora_box.export_csv(pandora_path)
200
201     logger.set_time(1)
202     logger.set_runtime()
203     logger.set_mem_usage()
204     logger.export_log()
205
206     return session_name
207
208 def kmppti_precompute_using_thread(dataset):
209     algorithm = 'kmppti-thread'
210     logger = Logger(algorithm, 'precomputing')
211     logger.set_time(0)
212     logger.set_data_info(dataset[0].split('_'))
213     print ('PrecomputeStarted')
214
215     event_queue = EventQueue()
216     data = {}
217     data = input_csv(dataset, event_queue)
218     event_queue.sort_queue()
219     metadata = get_metadata(event_queue, dataset, data)
220
221     data['product']['active'] = []
222     data['customer']['active'] = []
223
224     pandora_box = PandoraBox(metadata[4] + 1, metadata[0] + 1,
225                               data['product']['data'])
226     rsl = ReverseSkyline(data['product'], data['customer'], metadata[3])
227     dsl = DynamicSkyline(data['product'], data['customer'], metadata[3])
228
229     while not event_queue.is_empty():
230         event = event_queue.dequeue()

```

```

131     if event[1] == 0:
132         if event[3] == 0:
133             data['product']['active'].append(event[2])
134             rsl_result = rsl.compute(event[2])
135             threads = []
136             for cust_id in rsl_result:
137                 t = threading.Thread(target=compute_dsl, args=(cust_id, event[0],
138                                         event[3], event[2], dsl))
139                 threads.append(t)
140             for t in threads:
141                 t.start()
142             for t in threads:
143                 t.join()
144             threads = []
145             for cust_id in data['customer']['active']:
146                 if 'dsl' in data['customer']['data'][cust_id].keys():
147                     t = threading.Thread(target=update_pbox, args=(cust_id, event[0],
148                                         dsl, pandora_box, data))
149                     threads.append(t)
150             for t in threads:
151                 t.start()
152             for t in threads:
153                 t.join()
154
155 elif event[3] == 1:
156     threads = []
157     for cust_id in data['customer']['active']:
158         if 'dsl' in data['customer']['data'][cust_id].keys():
159             t = threading.Thread(target=update_pbox, args=(cust_id, event[0],
160                                         dsl, pandora_box, data))
161             threads.append(t)
162     for t in threads:
163         t.start()
164     for t in threads:
165         t.join()
166     rsl_result = rsl.compute(event[2])
167     threads = []
168     for cust_id in rsl_result:
169         t = threading.Thread(target=compute_dsl, args=(cust_id, event[0],
170                                         event[3], event[2], dsl))
171         threads.append(t)
172     for t in threads:
173         t.start()
174     for t in threads:
175         t.join()
176     data['product']['active'].remove(event[2])
177     threads = []
178     for cust_id in rsl_result:
179         t = threading.Thread(target=recheck, args=(cust_id, event[0],
180                                         dsl, data))
181         threads.append(t)
182     for t in threads:
183         t.start()
184     for t in threads:
185         t.join()
186
187 elif event[1] == 1:
188     if event[3] == 0:
189         data['customer']['active'].append(event[2])
190         dsl.customer_in(event[2], event[0])
191         if 'dsl' in data['customer']['data'][event[2]].keys():
192             pandora_box.update(data['customer']['data'][event[2]]['dsl'])
193             dsl.update_history(event[2])
194

```

```

$95     elif event[3] == 1:
$96         dsl.update(event[2], event[0])
$97         if 'dsl' in data['customer']['data'][event[2]].keys():
$98             pandora_box.update(data['customer']['data'][event[2]]['dsl'])
$99             dsl.update_history(event[2])
$100            data['customer']['active'].remove(event[2])
$101
$102    session_name, session_file = make_session(dataset[0].split('_'),
$103                                                 algorithm, metadata)
$104    pandora_path = session_file + '/' + session_name
$105    os.mkdir(pandora_path)
$106    pandora_box.export_csv(pandora_path)
$107
$108    logger.set_time(1)
$109    logger.set_runtime()
$110    logger.set_mem_usage()
$111    logger.export_log()
$112
$113    return session_name
$114
$115 def kmptti_precompute_without_rsl(dataset):
$116     algorithm = 'kmptti-no-rsl'
$117     logger = Logger(algorithm, 'precomputing')
$118     logger.set_time(0)
$119     logger.set_data_info(dataset[0].split('_'))
$120     print ('Precompute started')
$121
$122     event_queue = EventQueue()
$123     data = {}
$124     data = input_csv(dataset, event_queue)
$125     event_queue.sort_queue()
$126     metadata = get_metadata(event_queue, dataset, data)
$127
$128     data['product']['active'] = []
$129     data['customer']['active'] = []
$130
$131     pandora_box = PandoraBox(metadata[4] + 1, metadata[0] + 1,
$132                               data['product']['data'])
$133     dsl = DynamicSkyline(data['product'], data['customer'], metadata[3])
$134
$135     while not event_queue.is_empty():
$136         event = event_queue.dequeue()
$137         if event[1] == 0:
$138             if event[3] == 0:
$139                 data['product']['active'].append(event[2])
$140                 for cust_id in data['customer']['active']:
$141                     compute_dsl(cust_id, event[0], event[3], event[2], dsl)
$142                 for cust_id in data['customer']['active']:
$143                     if 'dsl' in data['customer']['data'][cust_id].keys():
$144                         update_pbox(cust_id, event[0], dsl, pandora_box, data)
$145
$146             elif event[3] == 1:
$147                 for cust_id in data['customer']['active']:
$148                     if 'dsl' in data['customer']['data'][cust_id].keys():
$149                         update_pbox(cust_id, event[0], dsl, pandora_box, data)
$150                     for cust_id in data['customer']['active']:
$151                         compute_dsl(cust_id, event[0], event[3], event[2], dsl)
$152                     data['product']['active'].remove(event[2])
$153
$154             elif event[1] == 1:
$155                 if event[3] == 0:
$156                     data['customer']['active'].append(event[2])
$157                     dsl.customer_in(event[2], event[0])
$158                     if 'dsl' in data['customer']['data'][event[2]].keys():

```

```

$59      pandora_box.update(data['customer']['data'][event[2]]['dsl'])
$60      dsl.update_history(event[2])
$61
$62      elif event[3] == 1:
$63          dsl.update(event[2], event[0])
$64          if 'dsl' in data['customer']['data'][event[2]].keys():
$65              pandora_box.update(data['customer']['data'][event[2]]['dsl'])
$66              dsl.update_history(event[2])
$67              data['customer']['active'].remove(event[2])
$68
$69      session_name, session_file = make_session(dataset[0].split('_'),
$70                                         algorithm, metadata)
$71      pandora_path = session_file + '/' + session_name
$72      os.mkdir(pandora_path)
$73      pandora_box.export_csv(pandora_path)
$74
$75      logger.set_time(1)
$76      logger.set_runtime()
$77      logger.set_mem_usage()
$78      logger.export_log()
$79
$80      return session_name

```

Kode Sumber 1.5 Kode sumber algoritme *Precompute*

```

1  import csv
2  from app.src.kmppti.pandora_box import PandoraBox
3  from app.src.kmppti.logger import Logger
4
5  def solution(file, k_product, time_start, time_end):
6      logger = Logger('kmppts', 'query')
7      logger.set_time(0)
8
9      pandora_box = PandoraBox()
10     market_contr = {}
11     output = []
12
13     k_product = int(k_product)
14     time_start = int(time_start)
15     time_end = int(time_end)
16
17     with open(file, 'r') as csv_file:
18         csv_reader = csv.reader(csv_file, delimiter=',')
19         total_prod = 0
20         for row in csv_reader:
21             prod_score = []
22             for col in row:
23                 prod_score.append(float(col))
24             pandora_box.insert_score(prod_score)
25             total_prod += 1
26
27         for i in range(1, total_prod):
28             market_contr[i] = pandora_box.get_score(i, time_start, time_end)
29         print('MarketContribution')
30         for key in market_contr:
31             print('{' + u':u}'.format(key, market_contr[key]))
32
33         sorted_prod = sorted(market_contr, key=lambda x: (market_contr[x]),
34                               reverse=True)
35
36         for i in range(0, k_product):

```

```

37     product = {}
38     product['id'] = sorted_prod[i]
39     product['market_contr']= market_contr[sorted_prod[i]]
40     output.append(product)
41
42     logger.set_time(1)
43     logger.set_runtime()
44     logger.set_mem_usage()
45     logger.set_query_info(k_product, time_start, time_end)
46     logger.export_log()
47
48     return output

```

Kode Sumber 1.6 Kode sumber algoritme *QueryProcessing*

```

1  # application imports
2
3  from flask_script import Manager, Server
4  from app import app
5
6  manager = Manager(app)
7
8  manager.add_command("run", Server(
9      use_debugger = True,
10     use_reloader = True,
11     host = '0.0.0.0',
12     port = '8991')
13 )
14
15 if __name__ == '__main__':
16     manager.run()

```

Kode Sumber 1.7 Kode sumber Flask Server

```

1  import os, json, csv
2  from collections import OrderedDict
3  from flask import render_template, request, redirect, url_for, flash, jsonify
4  from werkzeug.utils import secure_filename
5  from app import app
6  from app.src import solution as ss
7  from app.src import precompute as pr
8
9  ALLOWED_EXTENSIONS = set(['csv'])
10
11 def allowed_file(filename):
12     return '.' in filename and filename.rsplit('.', 1)[1].lower()
13     in ALLOWED_EXTENSIONS
14
15 def read_session():
16     session = []
17     try:
18         data = json.load(open(app.config['SESSION_FILE']))
19         for key in data['session']:
20             session.append(key)
21     except:
22         session = ['NoSession']
23     finally:
24         return session

```

```

25
26     def get_metadata(session_name):
27         data = json.load(open(app.config['SESSION_FILE']))
28         return data['session'][session_name]
29
30     def get_data(session_name):
31         data = json.load(open(app.config['SESSION_FILE']))
32         product_path = app.config['UPLOAD_DIR'] + '/' +
33             data['session'][session_name]['product_data']
34         customer_path = app.config['UPLOAD_DIR'] + '/' +
35             data['session'][session_name]['customer_data']
36         p_data = read_csv(product_path)
37         c_data = read_csv(customer_path)
38         return p_data, c_data
39
40     def read_csv(path):
41         data = []
42         with open(path) as csv_file:
43             reader = csv.DictReader(csv_file)
44             for row in reader:
45                 data.append(dict(OrderedDict(row)))
46         return data
47
48 @app.route('/')
49 @app.route('/index', methods=['GET'])
50 def index():
51     return render_template('index.html', title = 'Index')
52
53 @app.route('/precompute', methods=['GET', 'POST'])
54 def precompute():
55     if request.method == 'GET':
56         return render_template('precompute.html', title = 'Precomputing',
57                               session = read_session())
58     else:
59         if request.form.get('session'):
60             session_name = request.form.get('session')
61             return redirect ( url_for('search', session_name = session_name) )
62     datasets, filenames = [], []
63     datasets.append(request.files.get('p_dataset'))
64     datasets.append(request.files.get('c_dataset'))
65     algorithm = request.form.get('algorithm')
66     for dataset in datasets:
67         if dataset and allowed_file(dataset.filename):
68             filename = secure_filename(dataset.filename)
69             if not os.path.exists(app.config['UPLOAD_DIR']):
70                 os.mkdir(app.config['UPLOAD_DIR'])
71             dataset.save(os.path.join(app.config['UPLOAD_DIR'], filename))
72             filenames.append(app.config['UPLOAD_DIR'] + '/' + filename)
73
74     else:
75         flash('not allowed')
76     return redirect(request.url)
77     if algorithm == 'kmppti':
78         session_name = pr.kmppti_precompute(filenames)
79     elif algorithm == 'kmppti-t':
80         session_name = pr.kmppti_precompute_using_thread(filenames)
81     elif algorithm == 'kmppti-no-rsl':
82         session_name = pr.kmppti_no_rsl(filenames)
83         flash('success')
84     return redirect ( url_for('search', session_name = session_name) )
85
86 @app.route('/search', methods=['GET', 'POST'])
87 def search_session():
88     if request.method == 'GET':

```

```

89     return render_template('search_session.html', title = 'Search',
90                           session = read_session())
91 else:
92     session_name = request.form.get('session')
93     return redirect ( url_for('search', session_name = session_name) )
94
95 @app.route('/search/<session_name>', methods=['GET', 'POST'])
96 def search(session_name):
97     if request.method == 'GET':
98         return render_template('search.html', title = 'Search',
99                               session_name = session_name)
100 else:
101     k_product = request.form.get('num-of-product')
102     time_start = request.form.get('time-start')
103     time_end = request.form.get('time-end')
104     pandora_path = app.config['SESSION_DIR'] + '/' + session_name +
105                   '/pandora_box.csv'
106
107     result = ss.solution(pandora_path, k_product, time_start, time_end)
108     result_json = json.dumps(result)
109     loaded_r = json.loads(result_json)
110     return render_template('search.html', title = 'Search', result = loaded_r,
111                           param = [k_product, time_start, time_end], session_name = session_name)
112
113 @app.route('/visualization', methods=['GET', 'POST'])
114 def visualization():
115     if request.method == 'GET':
116         return render_template('visualization.html', title = 'Visualization',
117                               session = read_session())
118     else:
119         session_name = request.form.get('session')
120         return redirect ( url_for('vis', session_name = session_name) )
121
122 @app.route('/visualization/<session_name>', methods=['GET'])
123 def vis(session_name):
124     metadata = get_metadata(session_name)
125     p_data, c_data = get_data(session_name)
126     return render_template('vis.html', title = 'Visualization',
127                           session_name = session_name, metadata = metadata,
128                           product = p_data, customer = c_data)
129
130 @app.route('/get_vis_data/<session_name>', methods=['GET'])
131 def get_vis_data(session_name):
132     p_data, c_data = get_data(session_name)
133     data = p_data + c_data
134     new_data = []
135     counter = 0
136     for d in data:
137         dt = {}
138         dt['id'] = counter
139         dt['content'] = d['label']
140         dt['start'] = d['ts_in']
141         dt['end'] = d['ts_out']
142         new_data.append(dt)
143         counter += 1
144     return jsonify(new_data)

```

Kode Sumber 1.8 Kode sumber Flask Routes

```

1  <!DOCTYPE html>
2  <html lang="en">

```

```

3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>k-MPTI | {{ title }}</title>
8     <link rel='shortcut icon' type='image/x-icon' href="{{url_for('static',
9     filename='img/logo.ico')}}" />
10    <!-- load css -->
11    <link href="{{url_for('static',filename='vendor/bootstrap-4.3.1/css/
12     bootstrap.min.css')}}" rel="stylesheet">
13    <link href="{{url_for('static',filename='vendor/vis-4.21.0/vis.css')}}" rel="stylesheet" type="text/css" />
14    <!-- load script -->
15    <script src="{{url_for('static',filename='vendor/vis-4.21.0/vis.js')}}" />
16    </script>
17  </head>
18  {% if session %}
19  <body onload="get_session('{{session}})">
20  {% else %}
21  </body>
22  {% endif %}
23  <header>
24    <nav class="navbar navbar-expand-md navbar-dark fixed-top"
25      style="background-color:#32485c;">
26      <div class="container">
27        <a class="navbar-brand" href="{{request.script_root}}/index">
28          
30        <button type="button" class="navbar-toggler" data-toggle="collapse"
31          data-target="#navbarCollapse">
32          <span class="navbar-toggler-icon"></span>
33        </button>
34        {% set active_page = active_page|default('index') -%}
35        <div class="collapse navbar-collapse" id="navbarCollapse">
36          <div class="navbar-nav">
37            <a href="{{request.script_root}}/index" class="nav-item nav-link
38              {% if active_page=='index' %}>Home</a>
39            <a href="{{request.script_root}}/precompute"
40              class="nav-item nav-link{% if active_page=='precompute'
41                %}>Precompute</a>
42            <a href="{{request.script_root}}/search" class="nav-item nav-link
43              {% if active_page=='search' %}>Search</a>
44            <a href="{{request.script_root}}/visualization" class="nav-item
45              {% if active_page=='visualization' %}>Visualization</a>
46            </div>
47          </div>
48        </div>
49      </div>
50    </nav>
51  </header>
52  <main role="main">
53    <section class="jumbotron text-center" style="margin-top:60px;
54      margin-bottom:0;
55      background-color:#fff;">
56      <div class="row justify-content-center">
57        <div class="col-8">
58          <h1 style="margin-bottom:30px">Obtaining k-Most Promising Products
59          Based on Time Interval on Multidimensional Time Series Data</h1>
60          <p class="lead text-muted">Hafara Firdausi (0511154000043)</p>
61        </div>
62      </div>
63    </section>

```

```

67      <div class="py-5 bg-light">
68          <div class="container" style="margin-bottom:30px">
69              {# block content #{% endblock %}
70          </div>
71      </main>
72      <footer class="text-muted" style="margin-bottom:30px; margin-top:30px;
73         background-color:#fff;">
74          <div class="container text-center">
75              <span class="text-secondary mb-1 small">Copyright ©, 2019 -
76                  Tugas Akhir by
77                  <a href="https://github.com/mocatfrio/tugas-akhir"
78                      class="text-secondary"
79                      target="_blank">Hafara Firdausi</a>
80              </span>
81          </div>
82      </footer>
83      <!-- load script -->
84      <script type="text/javascript" charset="utf8" src="{{url_for('static',
85         filename='vendor/jquery/jquery-3.4.1.slim.min.js')}}"></script>
86      <script type="text/javascript" charset="utf8" src="{{url_for('static',
87         filename='vendor/jquery/jquery-3.4.1.min.js')}}"></script>
88      <script type="text/javascript" charset="utf8" src="{{url_for('static',
89         filename='vendor/bootstrap-4.3.1/js/bootstrap.bundle.min.js')}}"></script>
90      <!-- custom script -->
91      <script type="text/javascript">
92          function get_session(session_name) {
93              let dropdown = $('#session');
94              dropdown.empty();
95              dropdown.append('<option selected disabled>Choose Session</option>');
96              dropdown.prop('selectedIndex', 0);
97              $.each(session_name, function(index, value) {
98                  dropdown.append('<option></option>').attr('value', value).text(value));
99              });
100          }
101      </script>
102      {# block script #{% endblock %}
103  </body>
104 </html>

```

Kode Sumber 1.9 Kode sumber *template* halaman web

```

1  {% extends 'main.html' %}
2  {% set active_page = "index" %}
3
4  {% block content %}
5  <div class="row justify-content-center">
6      <div class="col-8 text-center">
7          <p class="text-left">The advancement of science and technology, especially
8             in the data analytics area, has influenced the way manufacturers do businesses
9             by collecting sales and customer preferences data, then using it to obtain
10            some informations to make the right business decision.</p>
11          <p class="text-left">Currently, there is a product selection strategy by
12             searching for k-most preferred product by customers, namely k-Most Promising
13             Products (k-MPP). This computation uses two types of skyline queries, dynamic
14             skyline and reverse skyline. Unfortunately, k-MPP computing has some
15             shortcomings. First, it's not considering the time variable, so the query
16             results are invalid to be used as a consideration in decision making based on
17             time. Second, k-MPP computing cannot process query based on time intervals.</p>
18          <p class="text-left">This study and research aims to answer the k-MPP query
19             based on time intervals in multidimensional time series data with serial time

```

```

20 by modeling k-Most Promising Products in Time Intervals (k-MPPTI ) query and
21 designing an algorithmic framework for processing the query. The algorithm is
22 implemented using parallel computing techniques to make data processing faster.
23 The effectiveness and efficiency of the algorithm was tested using real and
24 synthetic datasets.</p>
25     <a class="btn btn-outline-secondary"
26 href="{% request.script_root %}/precompute"
27     role="button" style="margin-top:20px">Let's Try!</a>
28 </div>
29 </div>
30
31 {# endblock %}

```

Kode Sumber 1.10 Kode sumber halaman web *Index*

```

1  {# extends 'main.html' %}
2  {# set active_page = "precompute" %}
3
4  {# block content %}
5  <div class="row justify-content-center">
6      <div class="col-10">
7          {# with messages = get_flashed_messages() %}
8          {# if messages[0] == 'not allowed' %}
9              <div class="alert alert-danger" role="alert">
10                  Filetype not allowed, please use CSV file
11              </div>
12          {# endif %}
13          {# endwith %}
14      </div>
15  </div>
16  <div class="row justify-content-center">
17      <div class="col-4">
18          <h4 style="margin-bottom:30px">Choose Session</h4>
19          <form action="{% url_for('precompute') %}" method="POST"
20          enctype="multipart/form-data">
21              <div class="form-group">
22                  <label for="session">Session</label>
23                  <select class="custom-select form-control" id="session" name="session"
24                  required>
25                  </select>
26              </div>
27              <button type="submit" class="btn btn-warning float-right">Submit</button>
28          </form>
29      </div>
30      <div class="col-1">
31          <h4>or</h4>
32      </div>
33      <div class="col-5">
34          <h4 style="margin-bottom:30px">Make Session</h4>
35          <form action="{% url_for('precompute') %}" method="POST"
36          enctype="multipart/form-data">
37              <div class="form-group">
38                  <label for="dataset">Product Dataset</label>
39                  <input type="file" class="form-control-file" id="dataset"
40                  name="p_dataset" required>
41              </div>
42              <div class="form-group">
43                  <label for="dataset">Customer Dataset</label>
44                  <input type="file" class="form-control-file" id="dataset"
45                  name="c_dataset" required>
46              </div>

```

```

47 <div class="form-group">
48     <label for="algorithm">Algorithm</label>
49     <select class="custom-select form-control" id="algorithm"
50         name="algorithm" required>
51         <option selected disabled>Choose Algorithm</option>
52         <option value="kmppti">k-MPPTI </option>
53         <option value="kmppti-t">k-MPPTI Parallel</option>
54         <option value="kmppti-no-rsl">k-MPPTI NoRSL</option>
55     </select>
56 </div>
57 <button type="submit" class="btn btn-warning float-right">Submit</button>
58 </form>
59 </div>
60 </div>
61 {# endblock #}
62

```

Kode Sumber 1.11 Kode sumber halaman web *Precompute*

```

1  {# extends 'main.html' #}
2  {# set active_page = "search" #}
3
4  {# block content #}
5  <div class="row justify-content-center">
6      <div class="col-10">
7          {# with messages = get_flashed_messages() #}
8          {# if messages[0] == 'success' #}
9              <div class="alert alert-success" role="alert">
10                  Precomputing dataset berhasil!
11              </div>
12          {# elif messages[0] == 'error' #}
13              <div class="alert alert-danger" role="alert">
14                  Silahkan lakukan precomputing terlebih dahulu!
15              </div>
16          {# endif #}
17          {# endwith #}
18          <nav class="nav nav-pills nav-fill" style="margin-bottom: 30px">
19              <a class="nav-item btn btn-outline-secondary disabled" role="button">
20                  {{ session_name }}</a>
21          </nav>
22      </div>
23  </div>
24  <div class="row justify-content-center">
25      <div class="col-5">
26          <h4 style="margin-bottom: 30px">Search Promising Product</h4>
27          <form action="{{ url_for('search'), session_name=session_name }}" method="POST">
28              <div class="form-group">
29                  <label for="num-of-product">Number of Product</label>
30                  <input type="text" class="form-control" id="num-of-product"
31                      placeholder="Masukkan jumlah produk yang ingin dicari"
32                      name="num-of-product" value="{{ param[0] if param else '' }}"
33                      required>
34              </div>
35              <div class="form-group">
36                  <label for="interval">Time Interval</label>
37                  <div class="row justify-content-center">
38                      <div class="col-6">
39                          <input type="text" class="form-control" id="time-start"
40                              placeholder="Choose Start Time" name="time-start"
41                              value="{{ param[1] if param else '' }}" required>
42

```

```

43          </div>
44          <div class="col-6">
45              <input type="text" class="form-control" id="time-end"
46                  placeholder="Choose End Time" name="time-end"
47                  value="{{uparam[2] if uparam else ''}}" required>
48          </div>
49      </div>
50      <button type="submit" class="btn btn-warning float-right">
51          Submit</button>
52      </form>
53  </div>
54  <div class="col-5">
55      {# if result %}
56      <h4 style="margin-bottom:30px">Result</h4>
57      <div class="table-responsive">
58          <table class="table-bordered-table-striped-table-hover"
59              id="table-police">
60              <thead>
61                  <tr>
62                      <th class="text-center-align-middle">No.</th>
63                      <th class="text-center-align-middle">Product</th>
64                      <th class="text-center-align-middle">
65                          Market Contribution</th>
66                  </tr>
67              </thead>
68              <tbody>
69                  {# for res in result %}
70                  <tr>
71                      <td class="text-center">{{ loop.index }}</td>
72                      <td>{{ res['id'] }}</td>
73                      <td>{{ res['market_contr'] }}</td>
74                  </tr>
75                  {# endfor %}
76              </tbody>
77          </table>
78      </div>
79      {# endif %}
80  </div>
81  </div>
82 {# endblock %}
83 
```

Kode Sumber 1.12 Kode sumber halaman web Search

```

1  {% extends 'main.html' %}
2  {% set active_page = "visualization" %}
3
4  {% block content %}
5  <div class="row justify-content-center">
6      <div class="col-10 text-center">
7          <nav class="nav-pills nav-fill" style="margin-bottom:30px">
8              <a class="nav-item btn-outline-secondary disabled" role="button">
9                  {{ session_name }}</a>
10             </nav>
11             <h4 style="margin-bottom:30px">Data Information</h4>
12         </div>
13     </div>
14     <div class="row justify-content-center">
15         <div class="col-5">
16             <table class="table">
17                 <tbody>
```

```

18      <th scope="row">Data Type</th>
19      {% if metadata['data_type'] == 'i' %}
20      <td>Independent (generate synthetic data)</td>
21      {% elif metadata['data_type'] == 'ac' %}
22      <td>Anti-Correlated (generate synthetic data)</td>
23      {% elif metadata['data_type'] == 'fc' %}
24      <td>Forest Cover Type Dataset (original data)</td>
25      {% else %}
26      <td>Unknown</td>
27      {% endif %}
28  </tr>
29  <tr>
30      <th scope="row">Number of Product</th>
31      <td>{{ metadata['total_prod'] }}</td>
32  </tr>
33  <tr>
34      <th scope="row">Number of Customer</th>
35      <td>{{ metadata['total_cust'] }}</td>
36  </tr>
37  <tr>
38      <th scope="row">Attribute of Product</th>
39      <td>
40          {% for a in metadata['attr_prod'] %}
41              {{ a }},
42          {% endfor %}
43      </td>
44  </tr>
45  <tr>
46      <th scope="row">Attribute of Customer</th>
47      <td>
48          {% for a in metadata['attr_cust'] %}
49              {{ a }},
50          {% endfor %}
51      </td>
52  </tr>
53  </tbody>
54 </table>
55 </div>
56 <div class="col-5">
57     <table class="table">
58         <tbody>
59             <tr>
60                 <th scope="row">Number of Dimension</th>
61                 <td>{{ metadata['dim'] }}</td>
62             </tr>
63             <tr>
64                 <th scope="row">Maximum Value</th>
65                 <td>{{ metadata['max_val'] }}</td>
66             </tr>
67             <tr>
68                 <th scope="row">Minimum Value</th>
69                 <td>{{ metadata['min_val'] }}</td>
70             </tr>
71             <tr>
72                 <th scope="row">Maximum Timestamp</th>
73                 <td>{{ metadata['max_ts'] }}</td>
74             </tr>
75             <tr>
76                 <th scope="row">Minimum Timestamp</th>
77                 <td>{{ metadata['min_ts'] }}</td>
78             </tr>
79         </tbody>
80     </table>
81 
```

```

82      </div>
83  </div>
84  <div class="row justify-content-center" style="margin-top:20px">
85    {%
86      if product %}
87      <div class="col-5">
88        <h4 style="margin-bottom:30px">Product Data</h4>
89        <p>Showing 1 to 20 data</p>
90        <div class="table-responsive">
91          <table class="table table-bordered table-striped table-hover"
92            id="table-product">
93            <thead>
94              <tr>
95                {%
96                  for a in metadata['attr_prod'] %}
97                  <th class="text-center align-middle">{{ a }}</th>
98                {%
99                  endfor %}
100            </tr>
101        </thead>
102        <tbody>
103          {%
104            for p in product %}
105            {%
106              if loop.index <= 20 %}
107              <tr>
108                {%
109                  for a in metadata['attr_prod'] %}
110                  <td>{{ p[a] }}</td>
111                {%
112                  endfor %}
113              </tr>
114            {%
115              endif %}
116            {%
117              endfor %}
118        </tbody>
119      </table>
120    </div>
121  {%
122    endif %}
123  {%
124    if customer %}
125  <div class="col-5">
126    <h4 style="margin-bottom:30px">Customer Data</h4>
127    <p>Showing 1 to 20 data</p>
128    <div class="table-responsive">
129      <table class="table table-bordered table-striped table-hover"
130        id="table-customer">
131        <thead>
132          <tr>
133            {%
134              for a in metadata['attr_cust'] %}
135              <th class="text-center align-middle">{{ a }}</th>
136            {%
137              endfor %}
138          </tr>
139        </thead>
140        <tbody>
141          {%
142            for p in customer %}
143            {%
144              if loop.index <= 20 %}
145              <tr>
146                {%
147                  for a in metadata['attr_cust'] %}
148                  <td>{{ p[a] }}</td>
149                {%
150                  endfor %}
151              </tr>
152            {%
153              endif %}
154            {%
155              endfor %}
156        </tbody>
157      </table>
158    </div>
159  {%
160    endif %}
161  <div class="row justify-content-center" style="margin-top:20px">
162    <div class="col-12 text-center">
```

```
|46   <h4 style="margin-bottom: 30px">Data visualization</h4>
|47   <div id="visualization">
|48     <div class="menu" style="margin-bottom: 20px">
|49       <input type="button" class="btn btn-secondary" id="zoomIn" value="Zoom in">
|50       <input type="button" class="btn btn-secondary" id="zoomOut" value="Zoom out">
|51       <input type="button" class="btn btn-secondary" id="moveLeft" value="Move left">
|52       <input type="button" class="btn btn-secondary" id="moveRight" value="Move right">
|53     </div>
|54   </div>
|55 <% endblock %>
|56
|57
|58
|59
|60
|61
|62
|63 {# block script #}
|64 <script type="text/javascript">
|65 $(function() {
|66   // DOM element where the Timeline will be attached
|67   var container = document.getElementById('visualization');
|68   // Create a DataSet (allows two way data-binding)
|69   $.getJSON('{{ url_for('get_vis_data', session_name = session_name) }}',
|70     function(data) {
|71       var length = data.length
|72       var new_data = []
|73       $.each(data, function(i, item) {
|74         new_data.push({id: item.id, content: item.content,
|75           start: parseInt(item.start), end: parseInt(item.end)})
|76       });
|77       console.log(new_data)
|78       var items = new vis.DataSet(new_data)
|79       var options = {
|80         max: parseInt('{{ metadata['max_ts'] }}'),
|81         min: parseInt('{{ metadata['min_ts'] }}'),
|82         horizontalScroll: true,
|83         zoomKey: 'ctrlKey',
|84         margin: {
|85           item: 10, // minimal margin between items
|86           axis: 5 // minimal margin between items and the axis
|87         },
|88         orientation: 'top'
|89       };
|90
|91       var timeline = new vis.Timeline(container, items, options);
|92
|93       function move (percentage) {
|94         var range = timeline.getWindow();
|95         var interval = range.end - range.start;
|96
|97         timeline.setWindow({
|98           start: range.start.valueOf() - interval * percentage,
|99           end: range.end.valueOf() - interval * percentage
|100         });
|101       }
|102       // attach events to the navigation buttons
|103       document.getElementById('zoomIn').onclick = function () {
|104         timeline.zoomIn( 0.2 );
|105       }
|106       document.getElementById('zoomOut').onclick = function () {
|107         timeline.zoomOut( 0.2 );
|108       }
|109       document.getElementById('moveLeft').onclick = function () {
|110         move( 1 );
|111       }
|112       document.getElementById('moveRight').onclick = function () {
```

```
210      move(-1); });
211  });
212 });
213
214
215
216 </script>
217 {%- endblock %}
```

Kode Sumber 1.13 Kode sumber halaman web *Visualization*

BIODATA PENULIS



Hafara Firdausi, lahir di Surabaya pada tanggal 19 September 1998. Penulis merupakan anak pertama dari 3 bersaudara. Penulis telah menempuh pendidikan formal di SD Negeri Gelam 1 Candi (2004-2010), SMP Negeri 1 Sidoarjo (2010-2013), dan SMA Negeri 1 Sidoarjo (2013-2015). Kemudian, penulis melanjutkan studi kuliah program sarjana di Departemen Informatika, Institut Teknologi Sepuluh Nopember Surabaya.

Selama menempuh pendidikan di Informatika ITS, penulis pernah menjadi asisten dosen dan praktikum untuk mata kuliah Jaringan Komputer (2017-2018) dan Sistem Operasi (2019), serta menjadi Administrator Laboratorium Arsitektur dan Jaringan Komputer (AJK). Selain itu, penulis juga aktif dalam kegiatan organisasi dan kepanitiaan, antara lain menjadi Badan Pengurus Harian I 3D (Desain, Dekorasi, dan Dokumentasi) Schematics ITS 2017, staff dan staff ahli Departemen Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS, dan Panitia Gemastik ke-11 Bidang Keamanan Jaringan dan Sistem Informasi (KJSI). Penulis dapat dihubungi melalui surel di hafarafirdausi@gmail.com.