MENENTUKAN CLICK-THROUGH RATE IKLAN DARING YANG OPTIMAL DENGAN MENGGUNAKAN ALGORITMA ADAPTIVE EPSILON GREEDY

Makalah

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Matematika
Program Studi Matematika

Disusun oleh:

Paulinus Alan Sanjaya Jamlu

NIM : 193114033

PROGRAM STUDI MATEMATIKA JURUSAN MATEMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

2023

DETERMINING THE OPTIMAL CLICK-THROUGH RATE OF ONLINE ADS USING THE ADAPTIVE EPSILON GREEDY ALGORITHM

Paper

Presented as a partial fulfillment of requirements
to obtain the Degree of Bachelor of Mathematics
Mathematics Study Program

Prensented By:

Paulinus Alan Sanjaya Jamlu

Student Number: 193114033

MATHEMATICS STUDY PROGRAM

FACULTY OF SCIENCE AND TECHNOLOGY

SANATA DHARMA UNIVERSITY YOGYAKARTA

2023

MAKALAH

MENENTUKAN CLICK-THROUGH RATE IKLAN DARING YANG OPTIMAL DENGAN MENGGUNAKAN ALGORITMA ADAPTIVE EPSILON GREEDY

Disusun oleh:

Paulinus Alan Sánjaya Jamlu

NIM: 193114044

Telah disetujui oleh:

Dosen Pembimbing

Y.G. Hartono, S.Si., M.Si., Ph.D.

20 Desember 2023

MAKALAH

MENENTUKAN CLICK-THROUGH RATE IKLAN DARING YANG OPTIMAL DENGAN MENGGUNAKAN ALGORITMA ADAPTIVE EPSILON GREEDY

Dipersiapkan dan ditulis oleh:

Paulinus Alan Sanjaya Jamlu

NIM: 193114044

SUSUNAN DEWAN PENGUJI

JABATAN	NAMA LENGKAP	TANDA TANGAN
Ketua	: Dr. Lusia Krismiyati Budiasih	
Sekertaris	: Cyprianus Kuntoro Adi, S.J., M.A., M.Sc., Ph.D.	
Anggota	: Y.G. Hartono, S.Si., M.Si., Ph.D.	

Yogyakarta, 20 Desember 2023

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan,

Ir. Drs. Haris Sriwindono, M. Kom., Ph.D.

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa makalah yang saya tulis ini tidak memuat karya atau bagian karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka dengan mengikuti ketentuan sebagaimana layaknya karya ilmiah.

Apabila di kemudian hari ditemukan indikasi plagiarisme dalam naskah ini, saya bersedia menanggung segala sanksi sesuai peraturan perundang-undangan yang berlaku.

Yogyakarta, 20 Desember 2023

Penulis,

Paulinus Alan Sanjaya Jamlu

LEMBAR PERYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Yang bertanda tangan di bawah ini, saya mahasiswa Universitas Sanata Dharma:

Nama: Paulinus Alan Sanjaya Jamlu

NIM: 193114033

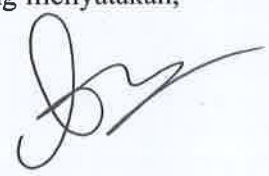
Demi pengembangan ilmu pengetahuan, saya memberikan kepada perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul:

MENENTUKAN CLICK-THROUGH RATE IKLAN DARING YANG OPTIMAL DENGAN MENGGUNAKAN ALGORITMA ADAPTIVE EPSILON GREEDY

beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan hak kepada Perpustakaan Universitas Sanata Dharma baik untuk menyimpan, mengalihkan dalam bentuk media lain, mengolah dalam bentuk pangkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari saya atau memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Dengan demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di Yogyakarta
Pada tanggal: 20 Desember 2023
Yang menyatakan,



Paulinus Alan Sanjaya Jamlu

HALAMAN PERSEMBAHAN

Karya ini saya persembahkan untuk:

Kedua orang tua saya yaitu Wilhelmus Wuati Jamlu dan Ibu Damiana Daiman, dan Sandi, Widi, Intan selaku adik penulis. Serta Bapak Y.G. Hartono, S.Si., M.Si., Ph.D. selaku dosen pembimbing yang bersedia memberikan dukungan dan bimbingan selama penyusunan tugas akhir ini Almamater saya Universitas Sanata Dharma Yogyakarta. Serta untuk saya sendiri yang sudah berusaha.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat, rahmat dan karunia-Nya penulis dapat menyelesaikan tugas akhir yang berjudul "Menentukan Click-Through Rate Iklan Daring yang Optimal dengan Menggunakan Algoritma Adaptive Epsilon Greedy". Penyusunan Tugas akhir ini dibuat dengan tujuan untuk memenuhi syarat memperoleh gelar sarjana Matematika pada Program Studi Matematika, Fakultas Sains dan Teknologi, Universitas Sanata Dharma.

Penulis menyadari dalam penulisan tugas akhir ini, banyak pihak yang telah membantu penulis dan mendapatkan dukungan dari banyak pihak. Oleh karena itu, pada kesempatan ini penulis ingin mengucapkan banyak terima kasih kepada:

1. Y.G. Hartono, S.Si., M.Si., Ph.D., selaku dosen pembimbing yang telah membimbing, memberikan arahan dan saran serta memberikan banyak ilmu pengetahuan kepada penulis. Serta sabar dan sangat bersemangat untuk membantu penulis menyelesaikan tugas akhir ini.
2. Bapak Dr. rer. Nat. Herry P. Suryawan, M.Si., selaku dosen pembimbing akademik dan ketua program studi matematika yang telah memberikan banyak ilmu dan pengalaman kepada penulis selama proses perkuliahan di Program Studi Matematika.
3. Romo Prof. Dr. Frans Susilo, SJ., Prof. Ir. Sudi Mungkasi, S.Si., M.Math.Sc., Ph.D., Ibu Dr. Lusia Krismiyati Budiasih, S.Si., M.Si., Bapak Bapak Ir. Ig. Aris Dwiatmoko, M.Sc., M.Si., Ph.D., Bapak Ricky Aditya, M.Sc., dan Ibu Maria Vianney Any Herawati, S.Si., M.Si., selaku dosen-

dosen Program Studi Matematika yang telah memberikan banyak ilmu dan pengalaman kepada penulis selama proses perkuliahan di Program Studi Matematika.

4. Bapak/Ibu/Karyawan Fakultas Sains dan Teknologi yang telah memberikan banyak bantuan dan informasi kepada penulis selama proses perkuliahan.
5. Papa Wilhelmus Jamlu dan Ibu Damiana Daiman selaku orang tua penulis dan Sandi, Widi dan Intan selaku adik-adik penulis yang menjadi penyemangat penulis.
6. Nene Kanis dan Nene Tina yang dengan tulus membantu penulis selama di Jogja.
7. Teman-teman penulis yang membantu saya dalam menulis dan proses menyelesaikan tugas akhir ini.
8. Semua pihak yang secara langsung maupun tidak langsung telah membantu penulis menyelesaikan tugas akhir ini yang tidak dapat disebutkan satu per satu.

Penulis Menyadari bahwa penyusunan tugas akhir ini masih memiliki kekurangan. Oleh karena itu, penulis membutuhkan masukan dan saran yang bersifat membangun dari semua pihak. Akhir kata penulis berharap tugas akhir ini dapat bermanfaat bagi pembaca dan perkembangan ilmu.

Yogyakarta, 26 Desember 2023

Paulinus Alan Sanjaya Jamlu

ABSTRAK

Dalam dunia periklanan daring, menentukan iklan mana yang paling efektif untuk menarik perhatian pengguna merupakan hal yang sangat penting. Salah satu cara untuk menentukan hal tersebut adalah dengan menggunakan algoritma Multi-Armed Bandit. Dalam penelitian ini, penulis menggunakan dua jenis algoritma Multi-Armed Bandit, yaitu A/B/n testing dan epsilon greedy, untuk menentukan *Click-through Rate* (CTR) iklan daring yang optimal. Hasil penelitian menunjukkan bahwa kedua algoritma tersebut dapat digunakan untuk menentukan CTR yang optimal dan menunjukkan performa hasil yang hampir sama. Untuk menganalisis data dan menentukan iklan daring CTR yang optimal, penulis menggunakan bahasa pemrograman Python. Dengan menggunakan Python, penulis dapat dengan mudah mengimplementasikan algoritma A/B testing dan epsilon greedy serta melakukan analisis data dengan cepat dan akurat.

Kata kunci: Click-Through Rate (CTR), iklan daring, Multi-Armed Bandit, MAB, A/B testing, epsilon greedy.

ABSTRACT

In the world of online advertising, determining which ad is most effective in attracting user attention is very important. One way to determine this is by using the Multi-Armed Bandit algorithm. In this study, the author used two types of Multi-Armed Bandit algorithms, namely A/B/n testing and epsilon greedy, to determine the optimal Click-Through Rate (CTR) for online ads. The results showed that both algorithms can be used to determine the optimal CTR and result similar performance. To analyze data and determine the optimal online ad CTR, the author used the Python programming language. By using Python, the author can easily implement A/B testing and epsilon greedy algorithms and perform data analysis quickly and accurately.

Keywords: Click-Through Rate (CTR), online advertising, Multi-Armed Bandit, MAB, A/B testing, epsilon greedy.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Media iklan konvensional seperti spanduk, koran, majalah, radio, dll. memiliki masalah dalam menjangkau target pasar yang lebih luas karena membutuhkan biaya yang besar dan sulit untuk menjangkau daerah-daerah terpencil. Dengan berkembangnya internet sekarang ini, media iklan seperti blog spot, video, *WhatsApp*, promosi melalui surat elektronik, aplikasi seluler, webinar, dan berbagai media daring lainnya membuat media iklan daring menjadi lebih diminati oleh jasa pemasaran sebuah perusahaan dan terbukti menjadi media pemasaran daring lebih efektif dan stabil (Bhatnagar, 2013). Tentu tidak semua jenis iklan dapat memberikan hasil yang maksimal sehingga diperlukan sebuah algoritma yang membantu perusahaan menentukan jenis atau media iklan mana yang paling optimal untuk digunakan.

Pembelajaran penguatan (*reinforcement learning*) adalah bagian dari pembelajaran mesin (*machine learning*) sebagai algoritma pengambilan keputusan. *Multi-Armed Bandit* (*MAB*) adalah salah satu algoritma yang digunakan pembelajaran penguatan untuk pengambilan keputusan saat kita memiliki banyak pilihan dengan informasi tentang hasil yang akan diterima tidak pasti setelah mengambil tindakan tersebut. Hasil yang diharapkan merupakan hasil optimal dari pemilihan keputusan tersebut. Hasil optimal didapatkan dengan menggunakan konsep utama dari *MAB*, yaitu eksplorasi dan eksploitasi. Eksplorasi adalah menggunakan atau memilih opsi yang tersedia secara acak dan memetakan hasil yang didapatkan. Eksploitasi adalah memilih

opsi yang memberikan hasil terbaik pada langkah eksplorasi. Dalam pemilihan eksplorasi dan eksploitasi, terdapat dilema apakah setelah t kali eksplorasi, lalu dilanjutkan ke tahap eksploitasi dengan hasil terbaik pada tindakan sebelumnya atau tetap mengeksplorasi untuk terus memetakan hasil yang lebih banyak. Dilema ini dapat diatasi dengan algoritma *MAB*. Masalah yang menggunakan bantuan *MAB* sebagai algoritma disebut sebagai masalah bandit (*bandit problem*).

Click-Through Rate (CTR) merupakan rasio yang mengukur seberapa baik kinerja kata kunci atau iklan yang ditampilkan dalam sebuah situs web atau media penyedia iklan daring lainnya. Misalkan sebuah iklan ditampilkan dalam sebuah situs web sebanyak 100 kali dan iklan tersebut diklik sebanyak 20 kali, maka CTR-nya adalah 20%. Dengan menggunakan *MAB*, kita dapat mempertimbangkan penyedia iklan atau jenis iklan mana yang memberikan CTR yang optimal dengan melakukan eksplorasi dan eksploitasi. Hasil optimal adalah hasil dimana dengan dana tertentu, CTR tinggi (mendekati 100%). Pada tugas akhir ini penulis akan mencari CTR paling optimal dari 10 jenis iklan daring dan menentukan jenis iklan mana yang paling baik.

Seorang atau tim yang melakukan pemasaran secara digital dalam masalah bandit disebut pelajar (*learner*) dan *situs web* sebagai media untuk menampilkan iklan disebut lingkungan (*environment*). Dari 10 jenis iklan, seorang pelajar mengambil langkah A pada sebuah waktu t atau langkah A_t dan dari langkah tersebut, lingkungan akan menampilkan hasil atau hadiah R pada waktu t tersebut (R_t). Nilai R akan bernilai 1 jika langkah yang diambil (dalam konteks CTR merupakan jenis iklan yang ditampilkan) memberikan hasil baik

dan 0 untuk hasil yang buruk sehingga $R_t = \{0,1\}$. Pelajar tentu tidak dapat mengetahui dengan pasti hasil langkah yang diambil selanjutnya, sehingga pelajar akan berpatokan pada riwayat (*history*). Tujuan dari pelajar, untuk memilih langkah yang memberikan total kumulatif pencapaian (*reward*) dari semua t langkah yaitu $\sum_{j=1}^{t} R_j$. Untuk memaksimalkan $\sum_{j=1}^{t} R_j$ ada beberapa metode dari algoritma *MAB* yang dapat digunakan seperti metode A/B, ε-*greedy* , *debugging* bandit, batas bawah (*lower bound*), dan lain-lain. Penulis akan fokus pada metode ε-greedy karena metode ini adalah metode yang paling sering digunakan dalam algoritma *MAB* (Mignon & da Rocha, 2017). Nilai ε mempengaruhi jumlah eksplorasi dan eksploitasi. Seorang pelajar akan memberikan nilai epsilon tertentu, sehingga total eksplorasi ditentukan dengan eksplorasi = $\varepsilon \times$ total langkah yang akan diambil. Setelah mengetahui total langkah yang diambil, pelajar akan secara acak memilih opsi yang tersedia hingga batas eksplorasi tercapai. Setelah eksplorasi selesai, pelajar akan melakukan tahap eksploitasi. Dalam langkah eksploitasi, pelajar akan memilih satu pilihan yang memberikan hasil tinggi pada riwayat pemilihan sebelumnya dan selanjutnya secara acak memilih pilihan lain. Hal ini bertujuan jika dalam beberapa langkah eksploitasi pilihan lain memberikan hasil yang lebih baik, pelajar dapat mengubah pilihan eksploitasi ke pilihan baru yang memberikan hasil yang baik. Langkah ini akan diulang hingga total langkah yang akan diambil tercapai. Dalam *MAB* dapat dihitung penyesalan (*regret*) yaitu selisih hadiah maksimal yang mungkin pada langkah tersebut ($R_{\max(j)}$) dan hadiah yang diterima pada langkah tersebut (R_j) atau $L(j) = R_{\max(j)} - R_j$.

1.2 Rumusan Masalah

Bagaimana cara menentukan opsi yang memberikan hasil paling optimal dari berhingga banyak pilihan menggunakan algoritma A/B/n testing dan epsilon greedy.

1.3 Batasan Masalah

Menggunakan data iklan daring dari supermarket di Amerika Serikat yang diakses dari kagle.com.

1.4 Tujuan Penulisan

Menentukan iklan yang memberikan hasil paling optimal dari 10 jenis iklan daring dengan menggunakan algoritma Multi-Armed Bandit.

1.5 Manfaat Penulisan

Untuk menambah wawasan dan pengetahuan penulis tentang algoritma pengambil keputusan terutama algoritma Multi-Armed Bandit.

1.6 Metode Penulisan

Metode penulisan yang digunakan dalam tugas akhir ini adalah metode studi pustaka dengan mempelajari buku-buku dan jurnal-jurnal matematika

yang berkaitan dengan algoritma pengambilan keputusan khususnya algoritma Multi-Armed Bandit, pengambilan data dan pengolahan data dengan menggunakan bahasa pemrograman *Python*.

1.7 Sistematika Penulisan

BAB I PENDAHULUAN

1.1 Latar Belakang

1.2 Rumusan Masalah

1.3 Batasan Masalah

1.4 Tujuan Penulisan

1.5 Manfaat Penulisan

1.6 Metode Penulisan

1.7 Sistematika Penulisan

BAB II PEMBELAJARAN MESIN DAN CLICK-THROUGH RATE

2.1 Click-Throuh Rate

2.2 Pengertian Pembelajaran Mesin

2.3 Jenis-Jenis Pembelajaran Mesin

2.4 Multi-Armed Bandit

BAB III MULTI-ARMED BANDIT

3.1 Bandit Berlengan Banyak

BAB II
CLICK-THROUGH RATE DAN PEMBELAJARAN MESIN

Pada bab ini, akan dibahas secara umum tentang *Click-through Rate* dan pembelajaran mesin. Pada Click-Through Rate, akan dijelaskan tujuan dan cara kerjanya. Pada pembelajaran mesin, akan dijelaskan apa itu pembelajaran mesin dan jenis-jenisnya. Setelah mengenal pembelajaran mesin, akan dibahas salah satu algoritma dalam pembelajaran mesin yaitu Multi Armed Bandit dan beberapa metode yang digunakan dalam algoritma tersebut.

2.1 Click-Through Rate

Click-through Rate (CTR) merupakan rasio yang mengukur seberapa baik kinerja kata kunci atau iklan yang ditampilkan dalam sebuah situs web atau media penyedia iklan daring lainnya. CTR adalah ukuran yang digunakan untuk menganalisis email, situs web, dan iklan daring (Google, Bing, Yahoo, dll). CTR biasanya digunakan untuk mengukur keberhasilan upaya pemasaran. Beberapa contoh CTR adalah:

- Iklan pada media sosial seperti Facebook, Google, dll.
- Tautan *call-to-action* pada email; tautan *call-to-action* adalah tautan yang dirancang agar pembaca email melakukan tindakan tertentu pada email yang diberikan, seperti mengklik tautan untuk mempelajari lebih lanjut tentang produk atau layanan, atau untuk melakukan pembelian.
- *Hyperlink* pada sebuah situs web; *hyperlink* adalah sebuah tautan yang mengarahkan pembaca pada tindakan tertentu, seperti mengarahkan

pembaca ke situs web baru yang memberikan informasi lebih detail atau menjadikan pembaca sebagai pelanggan dari tautan yang diarahkan.

CTR menghitung kinerja tautan atau iklan yang diberikan. Artinya, apakah yang ditampilkan menarik pembaca untuk mengklik elemen yang ditampilkan. Pada umumnya, iklan yang ditampilkan pada sebuah situs daring memerlukan biaya yang disebut *pay-to-click* (PPC) atau *cost per mille* (CPM). PPC adalah model periklanan daring di mana pengiklan membayar kepada penerbit, seperti mesin pencari, pemilik situs web, atau sosial media setiap kali sebuah iklan diklik. Model PPC hanya akan dikenakan biaya ketikan iklan yang ditampilkan diklik oleh pengguna. Model CPM atau biaya per tayangan adalah biaya yang dikenakan kepada pengiklan setiap kali iklan mereka ditampilkan sejumlah tertentu kali, biasanya per seribu tayangan. CTR dapat membantu menganalisis performa iklan dan berapa biaya yang dikeluarkan. Selain mengukur performa iklan, CTR juga digunakan sebagai skor kualitas sebuah iklan. Semakin tinggi skor CTRnya, maka semakin baik kualitas iklan tersebut yang mengarah pada biaya per klik semakin rendah.

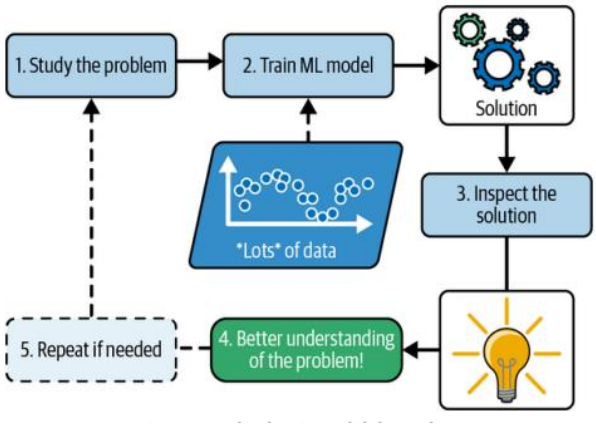
Nilai CTR dapat ditingkatkan performanya dengan menyediakan konten yang sesuai dengan target pasar. Selain itu, posisi iklan pada laman juga mempengaruhi CTR iklan.

2.2 Pengertian Pembelajaran Mesin

Pembelajaran Mesin (*Machine Learning*) adalah sebuah cabang dari *artificial intelligence* (AI) atau sebuah kecerdasan buatan yang berfokus pada memungkinkan mesin untuk melakukan pekerjaan mereka dengan terampil dengan menggunakan perangkat lunak cerdas (*intelligence software*). Menurut Arthur Samuel dalam buku Machine Learning: Algorithms and Applications pembelajaran mesin adalah bidang studi yang memberikan komputer kemampuan untuk belajar tanpa diprogram secara eksplisit. Metode pembelajaran statistika menjadi tulang punggung dari perangkat lunak cerdas yang digunakan untuk mengembangkan kecerdasan mesin (*machine intelligence*). Pembelajaran mesin sebagai sebuah disiplin ilmu membutuhkan data untuk belajar, maka perlu ada hubungan dengan disiplin lain yaitu basis data (*database*). Menurut Profesor Tom Mitchell yang dikutip dari buku Machine Learning: Algorithms and Applications, pembelajaran mesin adalah hasil alami dari pertemuan antara Ilmu Komputer dan Statistik. Secara sederhana, dapat disimpulkan bahwa pembelajaran mesin adalah sebuah ilmu yang memadukan ilmu komputer dan statistik. Ilmu komputer digunakan untuk membangun mesin atau program komputer dan statistik digunakan untuk menyimpulkan data dan asumsi-asumsi. Sehingga, sebuah mesin dapat memprogram dirinya sendiri berdasarkan pengalaman dan beberapa struktur awal untuk memecahkan masalah.

Ada beberapa pekerjaan yang dapat dikerjakan oleh manusia dengan sangat mudah atau tanpa memerlukan usaha lebih, namun tidak dapat menjelaskan cara mereka melakukannya. Sebagai contoh, seseorang dapat

mengetahui suara temannya tanpa kesulitan. Namun jika kita ditanya kenapa dia mengetahui bahwa itu adalah suaranya, orang tersebut biasanya kesulitan untuk menjelaskannya dengan tepat. Karena kekurangan informasi atau pengetahuan tentang fenomena ini (dalam hal ini merupakan *speech recognition*), orang-orang kesulitan untuk kesulitan untuk membuat algoritmanya. Algoritma pembelajaran mesin sangat membantu untuk mengisi kekosongan dalam pemahaman ini.



Gambar 1. Alur kerja pembelajaran mesin

Gambar di atas menjelaskan alur kerja pembelajaran mesin. Berikut adalah penjelasan tiap alur kerja tersebut:

2.2.1 *Study the problem* (pelajari masalahnya); manusia berperan besar pada bagian ini untuk menentukan masalah apa yang mesin akan selesaikan. Selain itu, manusia juga berperan untuk melihat penyebab masalah tersebut atau faktor-faktor lain yang berhubungan dengan masalah tersebut.

2.2.2 *Train ML model* (latih model ML); pada langkah ini, mesin akan diberi data dalam jumlah yang besar. Dari data tersebut mesin belajar

struktur, pola, dll. Setelah mempelajari model tersebut, mesin akan memberikan solusi dari masalah yang ditentukan sebelumnya berdasarkan data yang diberikan.

2.2.3 *Inspect the solution* (memeriksa solusi); setelah mesin memberikan solusi dari masalah yang diberikan, manusia berperan untuk melihat apakah solusi yang diberikan memberikan hasil yang baik. Hasil yang baik dapat dilihat dari akurasi atau total keuntungan.

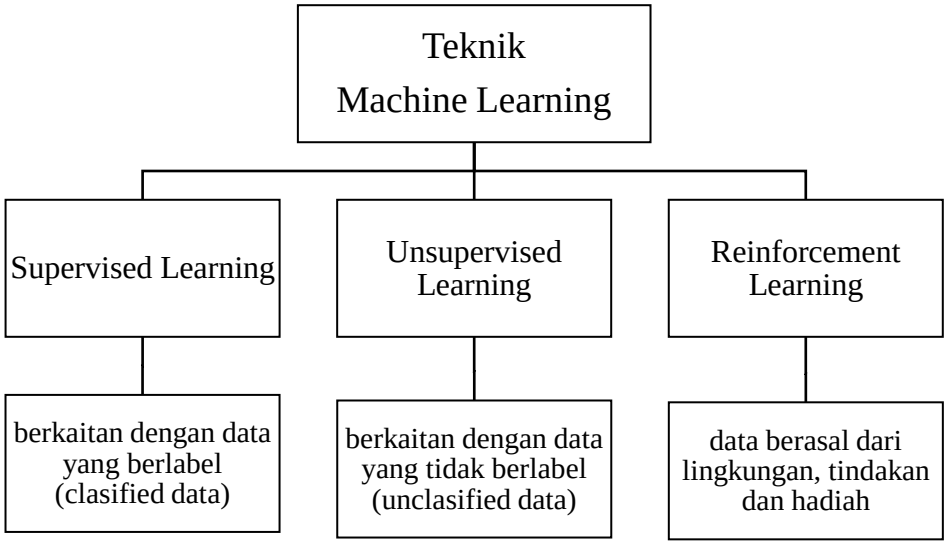
2.2.4 *Better understanding of the problem* (memahami masalah dengan lebih baik); solusi yang diberikan oleh mesin digunakan untuk menyelesaikan atau memahami masalah yang diberikan dengan lebih baik. Dengan memahami masalah dengan lebih baik, kita dapat memberikan solusi yang lebih tepat pada masalah tersebut.

2.2.5 *Repeat if needed* (ulang jika diperlukan); solusi yang diberikan oleh mesin tidak selalu memberikan hasil yang memuaskan. Oleh karena itu, perulangan akan diperlukan untuk memperbaiki siklus mana yang dirasa menghambat atau mengurangi akurasi dari model tersebut. Pada umumnya, hasil yang kurang akurat terjadi karena ketidakakuratan mesin dalam mempelajari model. Hal ini bisa disebabkan oleh program yang dibuat kurang baik atau data yang dilatih oleh mesin tidak “bersih”. Selain itu juga terdapat kesalahan pada manusia, seperti kesalahan interpretasi hasil. Artinya, mesin telah mempelajari data dengan baik dan memberikan hasil yang baik. Namun, orang salah menginterpretasi hasil tersebut sehingga masalah tidak terselesaikan.

Sebagai rangkuman, pembelajaran mesin bagus untuk:

- Masalah-masalah yang solusinya memerlukan *fine-tuning* atau memiliki aturan yang panjang (model pembelajaran mesin biasanya dapat menyederhanakan kode dan memberikan performa yang lebih baik dibandingkan pendekatan tradisional)
- Masalah kompleks yang tidak memiliki solusi yang baik dengan pendekatan tradisional (teknik pembelajaran mesin terbaik mungkin dapat menemukan solusi)
- Lingkungan yang fluktuatif (sistem pembelajaran mesin dapat dengan mudah dilatih ulang dengan data baru, selalu memperbarui pengetahuannya).
- Mendapatkan wawasan tentang masalah kompleks dan data dalam jumlah besar.

2.3 Jenis-Jenis Pembelajaran Mesin



2.3.1 Supervised learning

Supervised Learning (SL/ pembelajaran dengan pengawasan) adalah tentang belajar fungsi matematika yang memetakan himpunan *input* ke *output*/label yang seakurat mungkin. Pada SL, targetnya adalah menyimpulkan suatu fungsi atau pemetaan dari data yang diberi label. Pemberian label ini sangat penting bagi supervised learning karena label tersebut yang akan dipelajari oleh mesin. Beberapa contoh penggunaan SL adalah:

- Model pengenalan objek dari *input* kamera pada mobil *self-driving* sebagai pejalan kaki, rambu STOP, *traffic light* (membedakan warna merah, kuning dan hijau), mobil, dll.
- Model prakiraan yang memprediksi permintaan pelanggan terhadap suatu produk untuk musim liburan tertentu berdasarkan data penjualan sebelumnya.

Tanpa label pada data, sulit bagi mesin untuk memberi fungsi, aturan, pengenalan objek atau faktor yang mempengaruhi. Sehingga, model SL menyimpulkannya dari data yang berlabel. Berikut adalah beberapa poin penting tentang cara kerjanya:

- Saat *training*, model/data diberi label yang sesuai yang disediakan oleh *supervisor* (yang mana bisa merupakan seorang manusia yang ahli di bidang tersebut atau data yang sudah di proses).
- Setelah mesin mempelajari data (*training*), model membuat prediksi tentang *output* apa yang mungkin dari *input* yang diberikan.

- Model menggunakan fungsi aproksimator untuk mewakili data hasil proses yang menghasilkan *output*.

Dua grup yang berada pada ruang lingkup supervised learning adalah:

1) Regresi
2) Klasifikasi

2.3.2 Unsupervised learning

Algoritma Unsupervised Learning (UL) mengidentifikasi pola pada data yang sebelumnya tidak diketahui. Pada unsupervised learning, data tidak memiliki label. Idenya adalah untuk mencari struktur tersembunyi dari data yang dianalisis. Namun, perlu memiliki ide tentang harapan hasilnya, tetapi tidak menyertakan model dengan label.

Beberapa contoh penggunaan UL adalah:

- Mengidentifikasi kesamaan segmen pada gambar yang diberikan oleh kamera pada mobil *self-driving*. Model memungkinkan untuk memisahkan langit, jalan, bangunan dan lain sebagainya berdasarkan tekstur atau jarak pada gambar.
- Mengklaster penjualan bulanan ke dalam beberapa grup berdasarkan volume pembelian. Output yang mungkin adalah volume penjualan kecil, medium dan tinggi.

Jika melihat contoh di atas, terdapat perbedaan antara cara kerja SL dan UL, yaitu dengan cara-cara berikut:

- Model UL tidak mengetahui bagian mana jalan, langit, dan lain-lain dari *input* kamera, karena tidak terdapat label pada *input* tersebut.

UL mengidentifikasi perbedaan pola perbedaan pada data yang di*input* lalu memisahkan berbagai objek ke grup tertentu seperti jalan, langit, rambu, dll.

- Saat proses *inference*, model akan mengelompokkan *input* ke dalam grup tertentu tanpa mengetahui apa yang direpresentasikan grup tersebut.
- Fungsi aproksimator, seperti *neural network*, dipakai pada beberapa algoritma UL, tetapi tidak selalu.

Salah satu grup yang ada pada UL adalah *dimensional reduction* yaitu membagi data dalam kelompok tertentu untuk menyederhanakan data tanpa kehilangan terlalu banyak informasi. Selain itu juga UL juga berguna untuk mendeteksi anomali, seperti mendeteksi penggunaan kartu kredit yang tidak biasa untuk mencegah penipuan, menemukan cacat produksi, atau secara otomatis menghilangkan data pencilan (*outlier*) dari himpunan data sebelum diberikan ke algoritma pembelajaran lain.

2.3.3 Reinforcement learning

Reinforcement Learning (RL) adalah sebuah rangka kerja untuk mempelajari bagaimana cara mengambil keputusan di bawah ketidakpastian untuk memaksimalkan manfaat jangka panjang melalui *trial and error*. Keputusan ini dibuat secara berurutan. Artinya adalah hasil dari keputusan yang dibuat sebelumnya mempengaruhi situasi dan manfaat yang akan didapat nantinya. Ini yang membedakan RL dengan

SL dan UL yang mana tidak melibatkan pengambilan keputusan. RL menggunakan hasil pengamat dari interaksi pada lingkungan untuk mengambil tindakan yang akan memaksimalkan hadiah (*reward*) atau meminimalkan risiko (*risk*). Hasil dari metode ini berupa program pintar (*intelengent programs*) yang disebut agen (*agent*). Untuk menghasilkan program cerdas, RL harus melalui beberapa langkah berikut:

1) *Input* diamati oleh agen
2) Fungsi pengambilan keputusan digunakan untuk membuat agen melakukan tindakan,
3) Setelah tindakan dilakukan, agen menerima hadiah atau risiko dari lingkungan.
4) Informasi pasangan *state-action* disimpan.

Proses ini diulang (bisa diulang hingga berkali-kali sesuai kebutuhan), memungkinkan agen untuk belajar dari pengalamannya dan meningkatkan kinerjanya seiring langkah.

Sebagai contoh, akan ditinjau kembali contoh di atas.

- Pada mobil *self-driving*, model akan mendapat *input* dari kamera tentang posisi, model, sisi, dan sudut dari jalur pada jalan. Model akan belajar cara membelokkan roda mobil dan mengatur kecepatan dari mobil agar mobil melewati mobil di depannya atau melewati tikungan yang ada.
- Dengan mempertimbangkan data penjualan historis suatu produk dan langkah yang dibutuhkan untuk mengirimkan inventaris dari

pemasok ke toko, model dapat memprediksi kapan dan berapa banyak unit yang harus dipesan dari pemasok agar permintaan pelanggan musiman terpenuhi dengan tingkat keberhasilan yang tinggi, sambil meminimalkan biaya persediaan dan pengiriman.

Dari contoh di atas, masalah atau tugas yang dikerjakan oleh RL berbeda dan lebih kompleks daripada 2 model sebelumnya. Berikut akan diuraikan perbedaan RL:

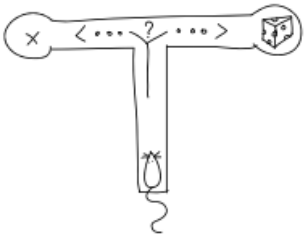
- *Output* dari model RL adalah sebuah keputusan dari sebuah situasi, bukan prediksi atau klastering.
- Model RL tidak diberikan keputusan atau model mana yang ideal dari *supervisor* dalam situasi tertentu. Sebaliknya, model belajar untuk mencari keputusan terbaik dari umpan balik pengalaman atau keputusan yang dibuat di masa lalu. Hal ini menjadikan *history* merupakan bagian yang sangat penting bagi model, karena model akan mengevaluasi *history* untuk mengambil keputusan selanjutnya. Misalnya, model RL akan belajar bahwa, jika terlalu mengebut saat menyalip mobil dapat menyebabkan kecelakaan, dan memesan terlalu banyak persediaan sebelum hari libur akan menyebabkan persediaan berlebih.
- Model RL biasanya menggunakan *output* dari model SL sebagai *input* untuk membuat keputusan. Sebagai contoh, *output* dari sebuah model pengenalan gambar pada mobil *self-driving* bisa digunakan untuk membuat keputusan kemudi. Misalnya model SL memberi

input manusia dan dikenali oleh model RL berada di tengah jalan. Model RL akan membuat keputusan untuk mengurangi laju kendaraan untuk menghindari kecelakaan (dalam hal ini model akan mendapatkan penalti/pengurangan poin jika mengalami kecelakaan).

Jadi, pengambilan keputusan merupakan pembeda RL dari metode ML lain. Yang menjadikannya menarik dan sangat kuat (*powerful*) adalah, kesamaannya tentang bagaimana cara mesin belajar "seperti" manusia untuk membuat keputusan dari pengalaman.

2.4 Multi-Armed Bandit

Masalah bandit (Bandit *problems*) dikenalkan oleh William R. Thompson pada sebuah artikel yang diterbitkan tahun 1933 pada *Biometrika*. Nama Multi-armed bandit sendiri muncul pada 1950-an saat Frederick Mosteller dan Robert Bush memutuskan untuk mempelajari cara binatang belajar dan menjalankan percobaan pada tikus dan kemudian pada manusia. Tikus-tikus tersebut dihadapkan pada dilema memilih untuk pergi ke kiri atau ke kanan tanpa mengetahui di ujung mana mereka akan mendapatkan makanan.



Gambar 2. Ilustrasi dilema

Sumber: “Bandit Algorithms” oleh Tor Lattimore dan Csaba Szepesv´ari.

Untuk mempelajari pengaturan pembelajaran serupa pada manusia, sebuah mesin "two-armed bandit" digunakan, di mana partisipan manusia dapat menarik salah satu tuas dan menerima hadiah dari tuas tersebut. Setiap tuas memberikan hadiah dengan distribusi yang berbeda dan tidak diketahui oleh partisipan. Mesin "two-armed bandit" ini merupakan pengembangan dari mesin "one-armed bandit", sebuah mesin kuno yang dioperasikan dengan tuas (disebut "bandit" karena mesin tersebut mengambil uang dari pemain).

Gambar 3. Mesin bandit

Sumber: https://www.megabagus.id/machine-learning-k-armed-bandit/

MAB menjadi sebuah algoritma yang sangat berguna dalam pengambilan keputusan dengan ketidakpastian akan hasil yang diterima dengan cara menentukan langkah ekplorasi dan eksploitasi. Ini menjadi tantangan yang cukup sulit bagi kebanyakan orang, namun MAB memberikan model yang sederhana untuk masalah tersebut.

Dari percobaan pada manusia dan tikus di atas cukup jelas bahwa MAB merupakan bagian dari RL karena pemain harus membuat keputusan berulang kali agar memberikan hasil yang maksimal. Pemain akan mendapatkan umpan balik (*feedback*) langsung dalam bentuk hadiah berupa uang (keju dalam percobaan tikus). Setelah sekian kali percobaan menarik tuas secara acak, pemain akan belajar menarik tuas yang memberikan hasil yang lebih besar dibandingkan dengan lengan lain berdasarkan pengalaman sebelumnya. Manusia dan tikus yang menjalani percobaan untuk memecahkan masalah bandit ini disebut pelajar atau dalam RL disebut agen.

Misalkan seorang pelajar bermain sebuah mesin bandit dengan dua lengan dan sudah menarik setiap lengan sebanyak 5 kali dan memberikan hadiah berikut. Pada putaran pertama, pelajar menarik lengan kiri dan memberikan hadiah 0, lalu putaran kedua menarik lengan kanan dengan hadiah 10, putaran ketiga menarik lengan kiri dengan hadiah 10, putaran keempat menarik lengan kiri dengan hadiah 0, putaran kelima menarik lengan kanan dengan hadiah 0, putaran keenam menarik lengan kiri dengan hadiah 0, putaran ketujuh hingga sembilan menarik lengan kanan dengan masing-masing memberikan hadiah 0 dan putaran terakhir menarik lengan kiri dengan hadiah 10. Berikut adalah pemetaan hasil penarikan lengan tersebut.

Tabel 1. Pemetaan Hadiah

Putaran	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Kiri	0		10	0		0				10
Kanan		10			0		0	0	0	

Dari **tabel 1**, lengan kiri terlihat memberikan hasil yang lebih baik. Dari 10 putaran, lengan kiri ditarik sebanyak 5 kali dengan total hadiah 20, sehingga rata-rata hadiah yang diterima adalah 4, sementara lengan kanan menerima total hadiah sebanyak 10 setelah menarik lengan sebanyak 5 kali sehingga rata-ratanya adalah 2.

AVG. lengan kiri = $\frac{0+10+0+0+10}{5} = \frac{20}{5} = 4$

AVG. lengan kanan = $\frac{10+0+0+0+0}{5} = \frac{10}{5} = 2$

Keterangan:

AVG = rata-rata.

Misalkan pelajar memiliki 10 kali percobaan tersisa. Strategi mana yang akan digunakan? Apakah akan memilih lengan kiri terus menerus dan mengabaikan lengan kanan atau Anda berasumsi bahwa kinerja buruk pada lengan kanan adalah ketidakberuntungan, lalu mencoba beberapa putaran lagi? Jika ya, berapa putaran lagi yang Anda coba? Ilustrasi ini menjelaskan dilema yang dialami oleh pelajar. Pelajar dihadapkan untuk memilih berapa kali putaran yang harus dilakukan. Dengan hasil pada putaran yang dilakukan, apakah akan dilakukan putaran kembali atau langsung mengambil lengan yang menurut putaran sebelumnya memberikan hasil

yang optimal. Dalam MAB dilema tersebut dapat diselesaikan dengan menggunakan beberapa metode yang akan dijelaskan pada tugas akhir ini, seperti A/B/n testing dan epsilon greedy.

Seperti yang dijelaskan pada pendahuluan dari TA ini, eksplorasi adalah menggunakan atau memilih opsi yang tersedia lalu memetakan hasil yang didapatkan untuk keperluan eksploitasi. Eksploitasi dilakukan berdasarkan perkiraan hasil yang tinggi dari tindakan yang diambil sebelumnya. Eksplorasi umumnya dilakukan lebih dahulu lalu diikuti dengan eksploitasi. Jika pelajar hanya melakukan eksplorasi, pelajar mungkin akan kehilangan lengan yang memberikan hasil yang sangat optimal. Di lain sisi, jika pelajar hanya melakukan eksploitasi, pelajar mungkin akan mengeksploitasi lengan yang tidak optimal, sehingga hanya menggunakan salah satu langkah di atas akan memberikan penyesalan yang besar. Jika lingkungan bersifat stasioner (distribusi probabilitas di setiap lengan tetap seiring langkah), pelajar mungkin bisa mengurangi jumlah eksplorasi seiring langkah. Namun, jika lingkungan bersifat tidak stasioner (distribusi probabilitas di setiap lengan berubah seiring langkah), pelajar harus terus bereksplorasi. MAB memiliki beberapa algoritma untuk menyeimbangkan eksplorasi dan eksploitasi, seperti A/B *testing*, Thompson sampling, $\varepsilon - greedy$, dan beberapa algoritma lainnya. Setiap algoritma memiliki cara untuk menentukan eksplorasi dan eksploitasi yang berbeda dari yang lain. Menentukan jumlah eksplorasi dan cara menentukan eksploitasi sangat menentukan hasil akhir dari masalah bandit ini, sehingga pemilihan algoritma juga berpengaruh besar terhadap hasil akhir. Untuk

masalah bandit tertentu, beberapa metode cenderung lebih baik dari yang lain.

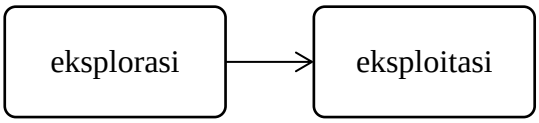
2.4.1 A/B *testing*

Salah satu strategi eksplorasi yang paling sering digunakan adalah A/B *testing*. Metode ini menentukan sesuatu (misal produk daring, iklan, dan lain sebagainya) yang bekerja lebih baik dari dua alternatif yang diberikan dengan melakukan eksplorasi murni pada langkah awal, lalu melakukan eksploitasi murni pada pilihan yang memberikan hasil terbaik pada langkah eksplorasi sebelumnya. Dalam pengujian dua situs web, A/B *testing* digunakan untuk membandingkan 2 buah situs secara langsung untuk menentukan situs mana yang lebih baik.

A/B *testing* merupakan metode yang di banyak literatur disebut sebagai metode yang mudah/sederhana. Namun, metode ini memiliki beberapa kekurangan yang membuatnya bisa menjadi metode yang "buruk" atau memberikan hasil yang kurang tidak optimal. Beberapa alasan yang menjelaskan kenapa A/B *testing* ini mungkin menjadi metode yang buruk adalah:

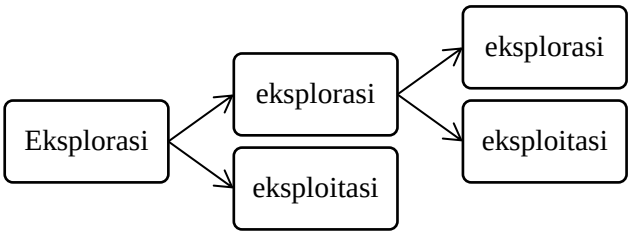
- Dari eksplorasi ke eksploitasi terjadi "lompatan" secara tiba-tiba. Maksudnya adalah ketika eksplorasi selesai dilakukan, metode A/B *testing* akan langsung melakukan eksploitasi tanpa ada transisi yang lebih halus sehingga memungkin untuk tetap melakukan eksplorasi sedikit demi sedikit hingga akhirnya eksploitasi murni dilakukan.
- Saat melakukan tahap eksplorasi murni, pelajar membuang kesempatan dari sebuah lengan yang memberikan tanda-tanda

optimal karena fokus untuk mengumpulkan data yang sebanyak mungkin. Jika sebuah lengan sudah memberikan hasil yang buruk, meskipun eksplorasi belum selesai, pelajar dapat mengeliminasi lengan yang jelas memberikan hadiah rendah.



2.4.2 $\varepsilon - greedy$

Selain A/B *testing*, epsilon greedy juga menjadi salah satu metode yang sering dipakai karena mudah, efektif dan memberikan hasil yang cenderung lebih baik dari A/B *testing*. Pada komputer sains, algoritma greedy merupakan algoritma yang selalu mengambil lengan yang terlihat baik pada saat tersebut meskipun akan mengarah pada konsekuensi buruk dalam jangka panjang. Epsilon greedy merupakan metode yang hampir rakus (*greedy*) karena pada umumnya akan mengeksploitasi pilihan terbaik yang tersedia, tetapi setiap beberapa langkah epsilon greedy akan mengeksplorasi lengan lain. Nilai epsilon akan menentukan banyaknya jumlah eksplorasi dalam algoritma ini. Berbeda dengan A/B test, epsilon greedy menjalankan algoritma dengan melakukan eksplorasi dan ekploitasi secara acak. Jadi, setelah melakukan eksplorasi pada langkah awal, algoritma dapat melakukan eksplorasi atau eksploitasi secara acak sampai jumlah eksplorasi dan eksploitasi terpenuhi.



Keuntungan dan kerugian menggunakan epsilon greedy

- Metode epsilon greedy dan A/B testing tidak efisien dan statis dalam mengalokasikan banyaknya eksplorasi. Pendekatan menggunakan epsilon greedy gagal untuk mengeluarkan atau menghapus lengan yang secara jelas memberikan hasil buruk dan terus mengalokasikan jumlah eksplorasi yang sama ke setiap lengan. Sebagai contoh, pada percobaan, lengan pertama memberikan hasil yang sangat buruk dibanding dengan beberapa lengan lain. Akan sangat efisien jika lengan pertama dihilangkan dan mengalokasikannya untuk mengeksplorasi lengan yang lain untuk mengidentifikasi lengan terbaik.
- Dengan metode epsilon greedy, eksplorasi berlanjut, tidak seperti A/B testing. Ini berarti jika lingkungan tidak stasioner, pendekatan menggunakan epsilon greedy memiliki potensi untuk mengubah dan memodifikasi pilihan lengan yang terbaik. Pada lingkungan yang stasioner, kita dapat berharap bahwa epsilon greedy dan A/B testing memberikan performa yang sama karena secara umum sama kecuali pada saat melakukan eksplorasi.

- Epsilon greedy dapat lebih efisien dengan secara dinamis mengganti nilai epsilon. Sebagai contoh, bisa memulai dengan nilai ε yang tinggi untuk eksplorasi pada awal langkah dan perlahan-lahan menguranginya untuk mengeksploitasi lebih nantinya. Dengan begini, tetap ada eksplorasi yang dilakukan secara kontinu atau berkelanjutan tapi tidak sebanyak pada awal langkah saat kita tidak memiliki pengetahuan tentang lingkungan tersebut.

BAB III
MULTI-ARMED BANDIT

Setelah mengenal pembelajaran mesin dan Multi-Armed Bandit, selanjutnya akan dijelaskan lebih rinci terkait algoritma tersebut. Apa saja bagian penting dari algoritma MAB ini, alur kerja algoritma dan 3 metode dari algoritma Multi-Armed Bandit yang digunakan.

3.1 Bandit Berlengan Banyak

Pada bab sebelumnya telah dijelaskan bahwa setiap tindakan atau langkah yang diambil oleh pelajar akan memberikan hadiah atau penghargaan yang mengikuti distribusi peluang yang berbeda dari setiap lengan dalam suatu lingkungan tertentu. Pada beberapa literatur, hadiah 0 merupakan hukuman atau *punishment*. Masalah pada MAB adalah merancang atau mengatur urutan dan/ rangkaian tindakan (dalam hal ini penarikan lengan) yang akan memaksimalkan total hadiah yang diharapkan (*expected total reward*) selama periode langkah tertentu. Untuk memecahkan masalah ini, alih-alih mempertimbangkan distribusi yang disebabkan oleh tindakan tertentu, pelajar hanya perlu sebuah nilai yang akan mengukur distribusi tersebut. Setiap distribusi memiliki nilai harapan atau rata-rata. Akibatnya, nilai harapan atau nilai rata-rata dapat kita gunakan untuk mengatasi masalah ini. Selain itu, nilai penyesalan juga dapat digunakan untuk mengatasi masalah bandit.

3.1.1 Q Value

Untuk menentukan nilai Q, berikut adalah beberapa definisi yang perlu diperhatikan:

- N adalah jumlah lengan bandit
- Tindakan dilambangkan dengan a_i, dimana $i = 1,2, \dots \dots, N$. Tindakan a_i merujuk pada penarikan atau pemilihan lengan ke-i.
- A_t adalah tindakan yang dipilih pada langkah ke-t. Perlu diperhatikan bahwa A_t adalah variabel acak.
- Tindakan A_t menghasilkan nilai $A_1, A_2, \dots \dots, A_t$.
- Setiap tindakan A_t pada langkah ke-t akan menghasilkan hadiah yang dilambangkan dengan R_t.
- Nilai rata-rata hadiah dari tindakan yang diambil disebut 'nilai tindakan' (*action value*). Secara matematis, ditulis sebagai:

$$Q_i(a) = E(R_t | A_t = a) \quad (1)$$

$$\forall a \, \epsilon \{a_1, a_2, \dots, a_N\}$$

Keterangan:

E = *expected value* (nilai harapan)

$\forall a$= untuk setiap nilai a

Dalam persamaan (1), $Q_i(a)$ merepresentasikan nilai yang diperoleh dari tindakan a_i.

Setiap lengan bandit memiliki nilai Q yang tidak diketahui, sehingga diperlukan algoritma untuk mencari nilai Q tersebut. Dalam menentukan nilai Q, pada setiap langkah t, sebuah tindakan A_t dipilih. Tindakan yang

diambil pada langkah t tersebut dilambangkan dengan A_t. Hadiah yang diperoleh dari setiap tindakan akan dilambangkan dengan R_t. Dengan demikian, seiring bertambahnya langkah t, akan terbentuk rangkaian tindakan $A_1, A_2, A_3, \ldots\ldots,$ dan rangkaian hadiah $R_1, R_2, R_3, \ldots\ldots$ Dari data ini, nilai untuk setiap tindakan yang dilambangkan dengan $Q_{i(a)}$ dapat dicari. Estimasi setiap langkah t dapat dihitung sebagai berikut:

$$Q_{i,t} = \frac{\sum_{j=1}^{t-1} R_{i,j}}{n_{i,j}} \quad (2)$$

Keterangan:

- $R_{i,j}$ merupakan hadiah yang dihasilkan dari tindakan A_i pada langkah j.
- $n_{i,j}$ adalah jumlah penarikan lengan a_i pada langkah j. Jika $n_{i,j}$ adalah 0 (artinya lengan a_i tidak pernah dipilih), maka $Q_{i,t}$ diberi nilai 0 untuk menghindari pembagian dengan 0.

Misalkan maksimal penarikan lengan mesin 4 kali. Maka $t = 4$ dan didapat $A_1 = a_1$, $A_2 = a_1, A_3 = a_2$, dan $A_4 = a_1$, dengan hadiah $R_1 = 1$, $R_2 = 2$, $R_3 = 5$, $R_4 = 4$.

Tabel 2. Tabel pemetaan penarikan lengan

Tarikan	Tindakan (A_t)	Hadiah lengan a_1	Hadiah lengan a_2
1	a_1	1	-
2	a_1	2	-
3	a_2	-	5
4	a_1	4	-

Tabel 2 menunjukkan pemetaan penarikan lengan sebanyak 4 kali ($t = 4$). Pada penarikan pertama (A_1), lengan a_1 terpilih dengan hadiah sebanyak 1. Karena lengan a_1 terpilih, maka lengan a_2 tidak menerima hadiah. Tindakan tersebut akan diulang hingga $t = 4$ dan hasilnya sesuai dengan tabel 2.

$$Q_1(a) = \frac{1+2+4}{3} = \frac{7}{3} \approx 2{,}33$$

$$Q_2(a) = \frac{5}{1} = 5$$

Namun, rumus (2) tidak praktis, karena diharuskan untuk menyimpan hasil dari setiap langkah. Jika t sangat besar, akan sulit untuk memetakan hasilnya. Sebagai alternatif, akan digunakan rumus rekursif untuk rata-ratanya. Rumus tersebut dapat diturunkan sebagai berikut:

$$Q_{t(a)} = \frac{R_1 + R_2 + R_3 + \cdots + R_t}{t}$$

$$Q_{t(a)} = \frac{1}{t}R_t + \frac{1}{t}(R_1 + R_2 + R_3 + \cdots + R_{t-1})$$

$$Q_{t(a)} = \frac{1}{t}R_t + \frac{1}{t}\frac{t-1}{t-1}(R_1 + R_2 + R_3 + \cdots + R_{t-1})$$

$$Q_{t(a)} = \frac{1}{t}R_t + \frac{t-1}{t}Q_{t-1(a)}$$

$$Q_{t(a)} = Q_{t-1} + \frac{1}{t}\left(R_t - Q_{t-1(a)}\right) \qquad (3)$$

Keterangan:

- Q_t adalah nilai rata-rata tindakan a pada langkah ke-t.
- R_{t-1} adalah hadiah yang diperoleh dari tindakan a pada langkah ke$-(t-1)$.

- R_t adalah hadiah yang diperoleh dari tindakan a pada langkah tersebut.
- t adalah jumlah berapa kali tindakan a dipilih.
- R_t adalah hadiah yang diperoleh dari tindakan a pada langkah ke-t.

Dengan rumus ini, nilai rata-rata Q pada setiap langkah dapat diperbarui dengan mempertimbangkan hasil perhitungan sebelumnya. Sebagai contoh, misal pelajar memiliki mesin bandit dengan 2 lengan yaitu lengan A dan B dengan nilai Q awal dari setiap lengan adalah 0. Setelah tarikan pertama pada lengan A dengan hadiah 1, nilai Q untuk lengan A diperbarui menjadi 1 dan jumlah kali lengan A telah ditarik diperbarui menjadi 1. Setelah tarikan kedua pada lengan B dengan hadiah 1, nilai Q untuk lengan B diperbarui menjadi 1 dan jumlah kali lengan B telah ditarik diperbarui menjadi 1. Setelah tarikan ketiga pada lengan A dengan hadiah 0, nilai Q untuk lengan A diperbarui menjadi 0,5 dan jumlah kali lengan A telah ditarik diperbarui menjadi 2. Setelah tarikan keempat pada lengan A dengan hadiah 1, nilai Q untuk lengan A diperbarui menjadi sekitar 0,67 dan jumlah kali lengan A telah ditarik diperbarui menjadi 3. Oleh karena itu, setelah empat tarikan dengan urutan lengan yang ditarik dan hadiah yang diterima yang Anda berikan, nilai-nilai Q untuk lengan A dan B masing-masing sekitar 0,67 dan 1. Dari contoh sederhana di atas, nilainya dapat disubstitusi sehingga didapat:

1) $Q_1(A) = 0 + \frac{1}{1}(1 - 0)$

$= 1$

2) $Q_2(B) = 0 + \frac{1}{1}(1 - 0)$

$= 1$

3) $Q_3(A) = 1 + \frac{1}{2}(0 - 1)$

$= 0{,}5$

4) $Q_4(A) = 0{,}5 + \frac{1}{3}(1 - 0{,}5)$

$= 0{,}67$

sehingga selama 4 langkah, lengan A terpilih sebanyak 3 ($N_4(A) = 3$) kali dan lengan B terpilih sebanyak 1 kali ($N_4(B) = 1$) dengan nilai Q seperti yang tertera di atas.

3.1.2 Penyesalan

Selain menggunakan nilai Q, penyesalan atau *regret* juga dapat digunakan untuk menentukan lengan yang paling optimal. Dalam masalah penarikan lengan, hadiah maksimum yang mungkin diperoleh ketika pelajar menarik lengan terbaik pada setiap langkah. Namun, karena pelajar tidak mengetahui lengan mana yang memberikan hasil terbaik pada langkah n, maka akan sangat mungkin pelajar menarik lengan yang tidak memberikan hasil optimal. Akibatnya, pelajar "menyesal" karena hasil yang tidak optimal. Pada saat penerimaan hadiah, pelajar bisa saja tidak mengetahui bahwa pilihan yang dibuatnya merupakan pilihan yang tidak optimal. Namun, pelajar dapat membandingkan hadiah yang diterima dengan kebijakan optimal yang mungkin diterima. Pada tugas akhir ini, data yang dipakai merupakan data CTR dengan hadiah 0 atau 1, sehingga

kebijakan optimal yang didapat adalah 1, sehingga cukup mudah untuk menghitung penyesalannya. Untuk percobaan lain yang dengan hadiah yang berbeda, kita dapat menentukan kebijakan optimal tersebut dengan melihat percobaan yang sudah dilakukan sebelumnya untuk melihat hadiah yang diterima sebagai patokan untuk melihat kemungkinan kebijakan optimal yang didapat.

Jika nilai Q yang terbaik adalah Q maksimal, penyesalan merupakan kebalikannya. Pelajar akan berusaha agar nilai penyesalan yang didapat seminimal mungkin sehingga hadiah yang diperoleh semaksimal mungkin. Penyesalan dihitung dengan mengambil selisih dari hadiah optimal dengan hasil yang didapat. Berikut adalah rumus untuk menghitung penyesalan:

$$L(j) = R_{\max(j)} - R_j \tag{4}$$

Keterangan :

$L(j)$: Penyesalan pada langkah j

$R_{\max(j)}$: Hadiah maksimal yang mungkin pada langkah j

R_j : Hadiah yang diterima pada langkah j

Sebagai contoh terdapat 3 lengan yang dapat dipilih oleh algoritma. Setiap kali lengan dipilih, algoritma akan menerima hadiah berupa angka 1 atau 0. Setiap lengan memiliki peluang munculnya hadiah 1 sebagai berikut:

Lengan 1: Probabilitas hadiah 1 = 0.8

Lengan 2: Probabilitas hadiah 1 = 0.5

Lengan 3: Probabilitas hadiah 1 = 0.2

Selama periode waktu t, misalnya t = 10, algoritma memilih lengan dan mendapatkan hadiah sebagai berikut:

Langkah 1: Lengan 1, hadiah = 1

Langkah 2: Lengan 2, hadiah = 0

Langkah 3: Lengan 3, hadiah = 1

Langkah 4: Lengan 1, hadiah = 1

Langkah 5: Lengan 3, hadiah = 0

Langkah 6: Lengan 2, hadiah = 1

Langkah 7: Lengan 1, hadiah = 0

Langkah 8: Lengan 1, hadiah = 1

Langkah 9: Lengan 2, hadiah = 0

Langkah 10: Lengan 2, hadiah = 1

Untuk menghitung penyesalan pada waktu t, perlu diketahui lengan optimal (lengan dengan probabilitas hadiah 1 tertinggi), yang dalam kasus ini adalah Lengan 1. Jika pelajar selalu memilih Lengan 1 pada setiap langkah, maka total hadiah yang akan diperoleh adalah: $R(T) = q * (Lengan\ 1) \cdot t = 0.8 \cdot 10 = 8$

Sementara itu, total hadiah yang diperoleh oleh algoritma adalah:

$$\sum X_t = 1 + 0 + 1 + 1 + 0 + 1 + 0 + 1 + 0 + 1 = 7$$

Dengan menggunakan rumus penyesalan

$$L(t) = R_{max(t)} - R_t$$

Penyesalan pada waktu t dapat dihitung

$$L(t) = 8 - 7 = 1$$

Penyesalan pada waktu T adalah 1, yang menunjukkan bahwa algoritma memiliki kerugian 1 dalam memilih lengan yang berbeda dari lengan optimal pada setiap langkah dalam penempatan iklan daring.

Rumus (4) hanya dapat digunakan untuk menghitung penyesalan pada satu langkah tertentu (langkah t). Kekurangannya, pelajar tidak dapat mengetahui apakah setelah melakukan penarikan sebanyak t kali, keputusan yang diambil mengarahkan nilai L menuju nilai minimum (menuju 0). Oleh karena itu, diperlukan untuk menghitung rata-rata penyesalan setiap langkah.

$$Avg.L(j) = \frac{\sum_{j=1}^{t} L_j}{j} \quad (5)$$

Keterangan :

$Avg.L(j)$ = rata-rata penyesalan pada langkah t

L_j = nilai penyesalan pada langkah j

j = banyaknya langkah yang diambil

Dengan rumus (5), pelajar dapat memetakan perubahan nilai L selama j langkah. Pelajar dapat melihat perubahan nilai L sebagai evaluasi keputusan yang diambil. Jika nilai L naik dari langkah sebelumnya, maka keputusan yang diambil merupakan keputusan yang memberikan hasil yang tidak optimal. Sebagai contoh, terdapat mesin dengan 2 lengan dengan 5 kali maksimal penarikan lengan bandit. Pada penarikan pertama, lengan A terpilih dengan hadiah 1 dan penarikan kedua lengan B mendapatkan hadiah 0. 3 penarikan terakhir, pelajar memutuskan untuk

menarik lengan A dengan hadiah berturut-turut adalah 1,0,0. Menggunakan rumus (2), nilai L dapat dihitung:

$$Avg.L(1) = \frac{1-1}{1} = 0$$

$$Avg.L(2) = \frac{(1-1)+(1-0)}{2} = 0.5$$

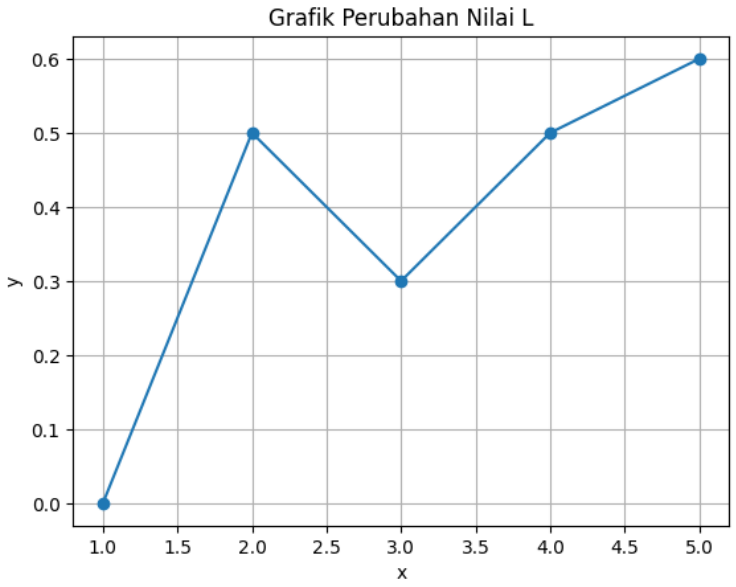
$$Avg.L(3) = \frac{(1-1)+(1-0)+(1-1)}{3} = \frac{1}{3} \approx 0.3$$

$$Avg.L(4) = \frac{(1-1)+(1-0)+(1-1)+(1-0)}{4} = \frac{2}{4} \approx 0.5$$

$$Avg.L(5) = \frac{(1-1)+(1-0)+(1-1)+(1-0)+(1-0)}{5}$$

$$= \frac{3}{5} = 0.6$$

Grafik perubahan nilai L divisualisasikan dengan grafik berikut:



Gambar 4. Grafik perubahan nilai L

Dari grafik di atas, nilai L memiliki tren naik, sehingga mengindikasikan keputusan yang dibuat oleh pelajar tidak memberikan hasil yang optimal yang mengakibatkan pelajar mengalami banyak kerugian (*loss*).

3.2 Algoritma A/B Testing

Pada bagian ini, akan dibahas variabel dan alur kerja algoritma A/B testing. Ada beberapa variabel algoritma A/B testing:

3.2.1 n_test

n_test adalah jumlah eksplorasi yang dilakukan oleh algoritma A/B testing. Jumlah n_test tidak memiliki aturan yang baku sehingga pengguna dapat memilih jumlah n_test secara sembarang. Namun, pada umumnya n_test diambil sekitar 10% dari jumlah maksimal langkah atau penarikan lengan.

3.2.2 n_prod

n_prod adalah banyaknya eksploitasi dari algoritma A/B testing.

Nilai n_test, n_prod, akan diberikan nilai untuk menentukan jumlah eksplorasi dan eksploitasi. Beberapa nilai lain juga akan diberi nilai awal 0. Setelah setiap variabel diberi nilai, selanjutnya akan dilakukan pemilihan lengan. Pada pemilihan lengan, algoritma A/B testing secara berurutan menjalankan eksplorasi lalu eksploitasi. Pada bagian eksplorasi, algoritma secara acak memilih lengan seperti pada eksplorasi algoritma MAB pada umumnya. Saat proses eksplorasi, nilai Q_n akan dihitung untuk setiap

lengan yang ada. Selain menghitung Q_n, pelajar juga akan menghitung total kumulatif hadiah yang diterima selama proses eksplorasi.

Setelah eksplorasi dilakukan, algoritma akan menjalankan eksploitasi murni. Artinya, hasil optimal pada langkah eksplorasi sebelumnya akan digunakan untuk mengeksploitasi satu lengan tertentu tanpa memperhatikan lengan lain. Lengan yang dieksploitasi adalah lengan dengan nilai Q_n tertinggi. Setelah didapat lengan dengan nilai Q_n tertinggi, pelajar akan menarik lengan tersebut hingga batas eksploitasi tercapai. Selama eksploitasi berlangsung, pelajar juga perlu untuk menghitung total hadiah yang diterima dan dijumlahkan dengan hadiah yang diterima saat proses eksplorasi untuk mengetahui total hadiah yang diterima. Sebagai contoh, perhatikan tabel himpunan data CTR berikut.

Tabel 3. Contoh data CTR

iklan	Banyak langkah									
	1	2	3	4	5	6	7	8	9	10
Ad 1	1	0	0	0	0	1	0	1	0	0
Ad 2	0	0	0	1	0	1	0	1	0	0
Ad 3	0	0	0	0	0	0	0	0	0	1

Misal, jumlah eksplorasinya adalah 4 kali (n_test = 4) sehingga eksplorasinya sebanyak 6 kali (n_prod = 6). Seorang pelajar akan melakukan langkah eksplorasi dan menghasilkan riwayat berikut

Tabel 4. Hasil penarikan

iklan	Langkah			
	1	2	3	4
Ad 1	1			
Ad 2			0	1
Ad 3		0		

Dengan menggunakan rumus (3) kita dapat mencari nilai Q dari setiap iklan. Setelah menghitung nilai tersebut, didapat hasil berikut

$$Q\_Ad1 = 0 + \frac{1}{1}(1 - 0) = 1$$

$$Q\_Ad2 = 0 + \frac{1}{1}(0 - 0) = 0$$

$$Q\_Ad2 = 0 + \frac{1}{2}(1 - 0) = 0.5$$

$$Q\_Ad3 = 0 + \frac{1}{1}(0 - 0) = 0$$

Dalam tabel, berikut adalah hasil nilai Q selama 4 langkah

Tabel 5. Nilai Q

iklan	Langkah				Hasil
	1	2	3	4	
Ad 1	1	-	-	-	1
Ad 2	-	-	0	0.5	0.5
Ad 3	-	0	-	-	0

Tanda (-) menyatakan bahwa pada langkah tersebut, iklan tersebut tidak terpilih. Pada kolom hasil, didapat bahwa Ad1 memiliki nilai Q terbesar. Sehingga pelajar akan memilih Ad1 selama 6 kali selama eksploitasi. Setelah dijumlahkan, total CTR yang didapat oleh pelajar adalah 3. Untuk menghitung penyesalan, dapat menggunakan cara yang serupa

dengan beberapa modifikasi sesuai dengan aturan penggunaan rumus penyesalan.

3.3 Algoritma Epsilon Greedy

Epsilon menjadi bagian penting dari metode ini, karena nilai epsilon akan mempengaruhi jumlah eksplorasi dan eksploitasi yang akan dijalankan. Epsilon adalah nilai bilangan desimal yang berada pada selang 0 hingga 1. Nilai epsilon biasanya diambil cukup kecil seperti 0.1, 0.05, 0.025, dan lainnya. Jika mengambil nilai epsilon 0.1, maka eksplorasi akan dilakukan sebanyak 10% dari total tarikan lengan (banyak langkah).

$$eksplorasi = \varepsilon \times banyak\ langkah$$

$$ekploitasi = banyak\ langkah - eksplorasi$$

Algoritma epsilon greedy tidak mengharuskan untuk menjalankan eksploitasi terlebih dahulu. Akibatnya, saat eksploitasi dijalankan lebih dahulu, pelajar dapat memilih secara acak lengan yang tersedia karena memiliki nilai Q yang sama yaitu 0. Setelahnya, nilai Q akan diambil berdasarkan nilai tertinggi. Eksplorasi dapat dijalankan di antara eksploitasi. Pelajar dapat secara acak memilih eksploitasi atau eksplorasi pada saat n dengan mengambil sebuah angka pada rentang tertentu. Setiap rentang diberi label eksplorasi dan eksploitasi, sehingga jika sebuah angka terpilih pada rentang tersebut maka tahap tersebut akan dijalankan. Sebagai contoh, ambil epsilon 0.1, maka pada rentang 0 hingga 1 diambil sembarang angka (bisa menggunakan fungsi pada perangkat lunak Excel atau bahasa pemrograman seperti Python). Jika angka

yang terambil ada pada rentang [0, 0.1], maka tahap eksplorasi akan dijalankan pada langkah tersebut dan jika angka yang terambil ada pada rentang (0.1,1] maka tahap eksploitasi akan dijalankan. Perlu diperhatikan bahwa peluang terambilnya angka pada selang tersebut haruslah sama (distribusi seragam) sehingga banyaknya eksploitasi dan eksplorasi sesuai dengan yang diharapkan. Sebagai contoh, kita gunakan kembali **tabel 5** dengan jumlah eksplorasi dan eksploitasi yang sama. Sehingga berikut adalah pemetaannya.

Tabel 6. Contoh data CTR

iklan	Banyak langkah									
	1	2	3	4	5	6	7	8	9	10
Ad 1	1	0		0						
Ad 2			0			1	0	1	0	0
Ad 3					0					

Perhatikan **tabel 6**. Warna abu pada kolom tabel menandakan bahwa pada langkah tersebut algoritma melakukan eksplorasi. Secara umum, untuk menentukan secara acak kapan eksplorasi dan eksploitasi adalah dengan menggunakan bantuan perangkat lunak. Asumsikan pada saat tersebut pelajar memilih untuk melakukan eksplorasi sesuai dengan **tabel 5** di atas. Dengan menggunakan rumus (3), kita dapat melakukan cara yang sama dengan contoh algoritma A/B/n test sebelumnya dan mendapat hasil nilai Q sebagai berikut:

Tabel 7. Hasil nilai Q

iklan	langkah									
	1	2	3	4	5	6	7	8	9	10
Q_Ad 1	1	0.5	-	0.16	-	-	-	-	-	-
Q_Ad 2	-	-	0	-	-	0.5	-	-	-	-
Q_Ad 3	-	-	-	-	0	-	-	-	-	-

Pada langkah pertama, pelajar memilih untuk melakukan eksplorasi dan mengambil Ad1 sehingga mendapat nilai Q sebesar 1. Pada langkah ke-2, pelajar menentukan untuk melakukan eksploitasi dan mendapat nilai Q 0.5. pelajar mengeksploitasi iklan dengan nilai Q tertinggi. Karena hanya Ad1 memiliki nilai tertinggi, maka eksploitasi dilakukan pada Ad1. Eksplorasi dan eksploitasi ini diulang hingga langkah ke-10. Karena pada langkah ke-6, total eksplorasi sudah terpenuhi, maka algoritma akan memberi rekomendasi untuk mengeksploitasi Ad2. Setelah semua langkah terpenuhi, total CTR yang didapat pada algoritma ini adalah 3.

Pada kasus di mana semua nilai Q masih 0, ada beberapa cara yang dapat digunakan untuk menentukan lengan yang akan dieksploitasi. Cara pertama adalah dengan memilih lengan yang paling jarang ditarik. Hal ini karena lengan tersebut masih memiliki peluang mendapat hasil optimal yang lebih baik dibanding lengan lain. Cara kedua adalah dengan memilih sembarang lengan. Ini merupakan “jalan pintas” untuk menentukan lengan yang akan dieksploitasi. Terjadi penurunan nilai Q karena pada langkah tersebut iklan memberikan hasil 0. Terdapat perbedaan **tabel 5** dan **tabel 7** yaitu pada **tabel 5** tidak terdapat kolom hasil. Ini dikarenakan pada algoritma epsilon greedy, setelah semua langkah telah diambil itu

menunjukkan bahwa algoritma sudah selesai. Maka nilai Q tidak diperlukan lagi.

3.4 Adaptive Epsilon Greedy

Adaptive epsilon greedy merupakan metode yang berdasar pada metode epsilon greedy dengan memungkinkan nilai epsilon untuk berubah dengan terkontrol selama eksekusi. Perubahan nilai epsilon dicapai dengan melakukan tindakan adaptif yang dipicu selama proses tersebut. Nilai baru untuk epsilon dihitung berdasarkan hadiah yang diterima dari lingkungan.

Pada metode ini terdapat 2 parameter yaitu l dan f. Parameter l digunakan untuk mengatur seberapa banyak ekpsplorasi dilakukan sebelum menggunakan mode adaptif yang mengubah nilai epsilon. Parameter f digunakan mengatur ulang hadiah yang diterima untuk mendapatkan nilai yang memadai dari fungsi untuk menghasilkan nilai baru untuk epsilon.

Berikut adalah algoritmanya

Algoritma Adaptive epsilon greedy

1. max_prev ← 0
2. k ← 0
3. if (random number from a normal distribution) ≤ ε then
4. $max_{prev} \leftarrow Q_t(A_t^*)$
5. $k \leftarrow k + 1$
6. If $k = l$ then
7. $\Delta \leftarrow (max_{curr} - max_{prev}) * f$
8. $if\ \Delta > 0\ then$
9. $\varepsilon \leftarrow sigmoid(\Delta)$
10. else
11. if Δ < 0 then
12. ε ← 0.5
13. end if
14. end if
15. $max_{prev} \leftarrow max_{curr}$

16. k ← 0
17. end if
18. randomly selects an action
19. else
20. selects A_t^*
21. end if

Algoritma di atas adalah implementasi dari metode *adaptive* ε-*greedy*. Algoritma ini digunakan untuk memilih sebuah tindakan (*action*) A saat sistem berada dalam suatu keadaan (*state*) t. Algoritma ini memutuskan apakah akan melakukan mode eksploitasi atau mode eksplorasi. Jika memilih mode eksploitasi, algoritma akan memilih tindakan dengan reward rata-rata tertinggi (A_t^*). Jika memilih mode eksplorasi, algoritma akan memilih tindakan secara acak. Ketika algoritma berada dalam mode eksplorasi, algoritma dapat melakukan tindakan adaptif, tergantung pada konfigurasinya, yang akan mengubah nilai ε.

Variabel max_{prev} dan k adalah variabel yang digunakan untuk menyimpan status algoritma. Variabel k digunakan untuk menghitung berapa kali algoritma melakukan mode eksplorasi setelah terakhir kali mengubah nilai ε. Ketika variabel k mencapai batas yang ditentukan dalam parameter l, algoritma akan memutuskan apakah akan mengubah nilai ε. Untuk melakukannya, algoritma menghitung selisih (Δ) antara hadiah rata-rata tertinggi (max_{curr}) dan hadiah rata-rata tertinggi sebelumnya (max_{prev}), yang diperoleh pada saat variabel k mencapai batas l terakhir kali. Selisih ini kemudian dikalikan dengan nilai parameter f untuk mengatur nilainya. Jika nilai Δ lebih besar dari nol, maka akan dihitung nilai baru untuk ε. Jika nilai Δ lebih kecil dari nol, itu berarti setelah pengaturan terakhir nilai ε, algoritma lebih sering memilih tindakan yang tidak optimal,

dan oleh karena itu, algoritma harus terus mencari tindakan yang optimal dengan mengatur ε menjadi 0,5. Jika nilai Δ sama dengan nol, nilai ε tidak berubah. Setelah mengatur nilai ε yang baru, variabel k diriset menjadi 0 dan variabel max_{prev} menerima nilai dari variabel max_{curr}.

Nilai baru untuk ε dihitung menggunakan fungsi sigmoid:

$$sigmoid(x) = \frac{1}{1+e^{-2x}} - 0.5 \quad (6)$$

Berdasarkan fungsi ini, nilai ε dapat bervariasi antara 0,0 dan 0,5. Secara empiris, ditemukan bahwa tidak ada keuntungan signifikan jika nilai ε lebih besar dari 0,5. Kesimpulannya, nilai ε akan berubah dengan keadaan berikut:

$$\varepsilon = \begin{cases} \varepsilon = sigmoid & ,jika\ \Delta > 0 \\ \varepsilon = 0.5 & ,jika\ \Delta < 0 \\ \varepsilon = \varepsilon\ sebelumnya & ,jika\ \Delta = 0 \end{cases}$$

BAB IV
APLIKASI EPSILON GREEDY DAN A/B/n TEST DALAM MENENTUKAN CLICK-THROUGHT RATE IKLAN DARING YANG OPTIMAL

Pada bab ini, algoritma yang sudah dipelajari atau dijelaskan sebelumnya akan diimplementasikan dengan cara memecahkan masalah di dunia nyata yaitu mencari jenis iklan yang paling baik berdasarkan nilai CTR dari berbagai jenis pilihan iklan yang tersedia. Secara umum, algoritma ini dapat digunakan untuk memecahkan masalah lain yang memerlukan pengambilan keputusan seperti mengambil pilihan yang terbaik dari berbagai pilihan yang ada. Algoritma dijalankan dengan pemrograman komputer menggunakan bahasa pemrograman Python.

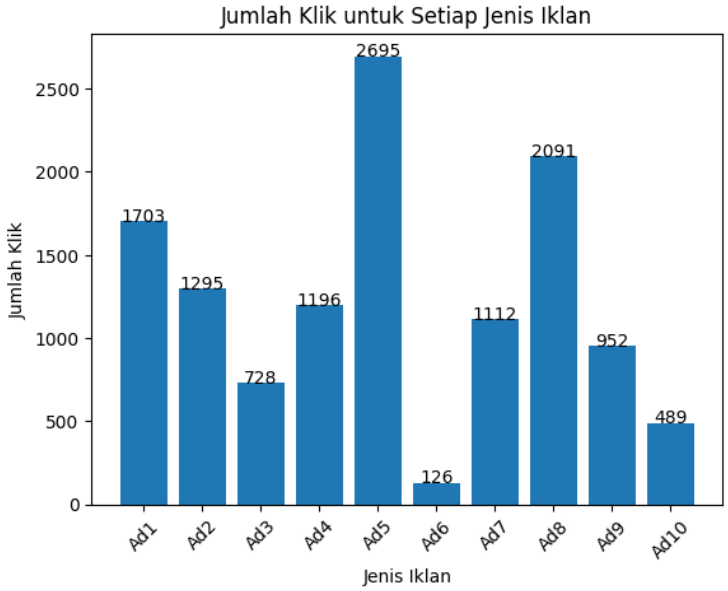
4.1 Himpunan Data CTR Iklan Daring

Tabel 8. Cuplikan himpunan data CTR iklan daring

	Ad 1	Ad 2	Ad 3	Ad 4	Ad 5	Ad 6	Ad 7	Ad 8	Ad 9	Ad 10
1	1	0	0	0	1	0	0	0	1	0
2	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
9996	0	0	1	0	0	0	0	1	0	0
9997	0	0	0	0	0	0	0	0	0	0
9998	0	0	0	0	0	0	0	0	0	0
9999	1	0	0	0	0	0	0	1	0	0
10000	0	1	0	0	0	0	0	0	0	0

Himpunan data iklan daring yang digunakan adalah 10 jenis iklan dari supermarket yang ada di Amerika Serikat. Himpunan data ini merupakan hasil CTR yang terdiri dari 10,000 baris data dengan nilai 0 dan 1. Nilai 1 berarti sebuah iklan yang ditampilkan pada sebuah situs web diklik oleh pengguna, sedangkan nilai 0 berarti sebuah iklan yang ditampilkan pada sebuah situs web tidak diklik oleh pengguna yang berarti setiap iklan ditampilkan pada sebuah situs web sebanyak 10,00 kali lalu dipetakan hasilnya. Jika digabungkan, supermarket menampilkan sebanyak satu juta kali iklan sehingga supermarket menghabiskan sangat banyak dana untuk iklan yang belum tentu menarik perhatian pelanggan. Menggunakan algoritma MAB, pihak supermarket hanya perlu menampilkan iklan sebanyak 10,000 kali untuk mencari jenis iklan mana yang menarik bagi pelanggan. Menarik diukur dari apakah pelanggan mengeklik iklan yang ditampilkan pada situs web yang dikunjungi. Lalu, situs web hanya perlu memilih satu jenis iklan saja yang memberikan hasil yang lebih baik dan menghiraukan iklan lain yang tidak menguntungkan perusahaan.

Berikut adalah plot seberapa banyak sebuah iklan diklik selama 10000 kali ditampilkan.



Gambar 5. Total klik setiap iklan

Dari gambar 5 di atas, terlihat bahwa Ad5 merupakan iklan dengan jumlah CTR paling banyak diklik oleh pengunjung pengunjung website daripada iklan lain, dengan nilai CTR pada Ad5 yaitu 2695. Sehingga, hasil simulasi dari algoritma yang sudah dipelajari, jika berjalan dengan optimal, akan mengeksploitasi Ad5 lebih banyak dibanding jenis iklan lain yang menjadikan Ad5 sebagai jenis iklan terbaik.

4.2 Simulasi Program

Algoritma ini dijalankan dengan menggunakan Python. Setiap algoritma dijalankan dengan perulangan (*looping*) sebanyak 500 kali. Artinya, setiap langkah t akan diulangi sebanyak 500 kali dan diambil rata-ratanya. *Loop* ini digunakan untuk melihat konsistensi dari algoritma A/B test dan epsilon greedy karena tahap eksplorasi dapat memberikan total hadiah yang berbeda karena merupakan pengambilan sampel acak.

Setiap algoritma dicoba dengan berbagai kemungkinan nilai dari parameter, lalu ditentukan nilai parameter yang memberikan hasil terbaik atau hasilnya dapat merepresentasikan nilai parameter lain. Setelah semua algoritma dijalankan, algoritma yang memberikan rata-rata total CTR paling tinggi merupakan algoritma dengan performa terbaik dan jenis iklan yang terbaik ditentukan oleh total berapa kali iklan tersebut dipilih. Semakin banyak lengan tersebut dipilih mengindikasikan bahwa iklan tersebut sering dieksploitasi oleh algoritma karena nilai Q dari iklan tersebut yang paling tinggi. Dari gambar 5, nilai CTR dari jenis iklan terbaik adalah 2695. Sehingga, kita dapat menyatakan bahwa algoritma kita berjalan dengan baik jika iklan yang terpilih adalah Ad5 dan nilai rata-rata CTR yang didapat mencapai 2695.

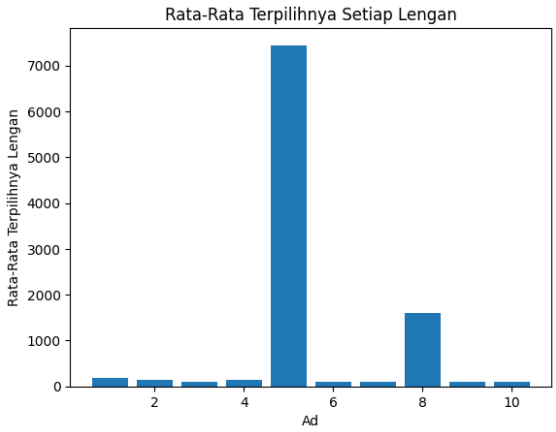
4.2.1 A/B/n test

Pada algoritma A/B test, eksplorasi dan eksploitasi dibagi dengan persentase 10% dan 90%, sehingga: $n_{test} = 1000$ dan $n_{prod} = 9000$. Pemilihan eksplorasi sebanyak 10% dari total langkah didapat dengan berbagai percobaan untuk mencari nilai eksplorasi paling optimal. Penulis mencoba beberapa nilai eksplorasi yang dijalankan pada algoritma ini dan mendapatkan bahwa eksplorasi sebanyak 10% memberikan hasil yang paling optimal.

4.2.1 Total hadiah

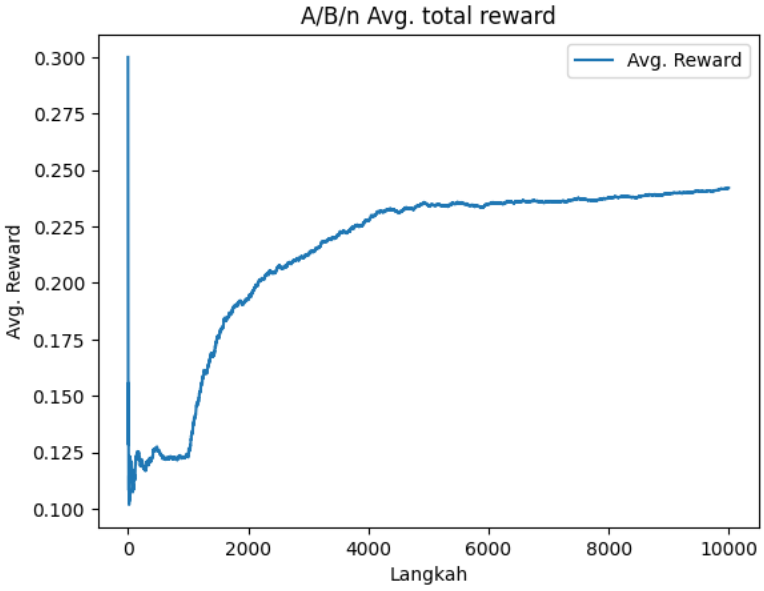
Hasil simulasi program dengan algoritma A/B/n test menunjukkan Ad5 adalah jenis iklan yang paling optimal dan dengan *Average* Total *CTR* (ATC) adalah 2429.21. ATC didapat melalui mencari rata-rata dari 500 hasil CTR dari algoritma ini yang dijalankan

pada program Python. Berikut adalah plot rata-rata jumlah pemilihan iklan selama algoritma dijalankan.



Gambar 6. Grafik A/B/n rata-rata terpilihnya lengan

Dari gambar di atas, terlihat bahwa Ad5, merupakan jenis iklan yang paling banyak dipilih atau dieksploitasi oleh algoritma. Jika dilihat, Ad8 juga lebih tinggi dari beberapa iklan lain. Hal ini bisa disebabkan karena, pada langkah eksplorasi, algoritma mendapatkan bahwa Ad8 dan AD5 memberikan hasil yang paling baik. Namun, karena nilai AD5 merupakan yang paling tinggi, maka pada langkah eksploitasi, algoritma A/B/n test memilih Ad5 sebagai iklan yang paling optimal.

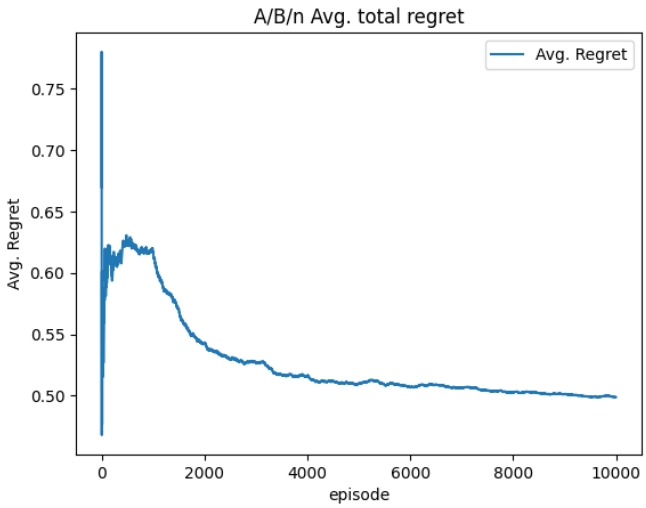


Gambar 7. Grafik A/B/n rata-rata total reward

Pada tahap eksplorasi murni dengan 1000 langkah, rata-rata hadiah dari algoritma ini tidak menunjukan peningkatan (lihat **Gambar 7)**. Hal ini dikarenakan algoritma masih mengambil iklan secara acak sehingga rata-rata hadiah yang diterima memberikan hasil nilai rata-rata yang cenderung kecil di 0.125. Pada langkah pertama, grafik menunjukkan perubahan drastis dari 0 hingga 0.3. Ini disebabkan pada langkah pertama tersebut nilai rata-rata hadiah yang diterima itu antara 0 atau 1. Karena algoritmanya dijalankan sebanyak 500 kali, kita mencari rata-rata dari nilai tersebut sehingga didapat nilainya antara 0 hingga 0.3 yang terlihat berbeda dari langkah setelahnya.

Setelah eksplorasi murni, eksploitasi dilakukan dengan hasil iklan yang optimal adalah Ad5, sehingga algoritma mengeksploitasi Ad5. Pada t yang lebih besar dari 1000, nilai rata-rata hadiah memiliki tren naik yang mengindikasikan bahwa algoritma mendapat jenis iklan yang optimal dan performa dari algoritma berjalan dengan baik.

4.2.2 Penyesalan



Gambar 8. Grafik total rata-rata regret

Pada grafik Avg. *Regret*, nilai penyesalan lebih tinggi pada 1000 langkah awal karena algoritma masih dalam tahap eksplorasi untuk mencari iklan dengan performa terbaik. Setelah tahap eksplorasi selesai, nilai penyesalan perlahan menurun mendekati 0.5 yang menunjukkan bahwa algoritma telah menemukan jenis iklan yang optimal dibandingkan dengan iklan-iklan lainnya.

4.2.2 Epsilon greedy

Algoritma ini akan dijalankan untuk 4 epsilon yang diambil yaitu 0.1, 0.075, 0.05, 0.025. Alasan pemilihan 4 nilai epsilon ini karena nilai epsilon yang lebih dari 0,1 cenderung memberikan hasil yang serupa dengan epsilon 0.075, 0.05 dan 0.025. Sehingga penulis menggunakan 4 epsilon tersebut karena merepresentasikan hasil dari epsilon yang lebih besar dari 0.1.

Untuk menentukan eksplorasi dan eksploitasi, penulis menggunakan bahasa pemrograman Python dengan *syntax* sebagai berikut:

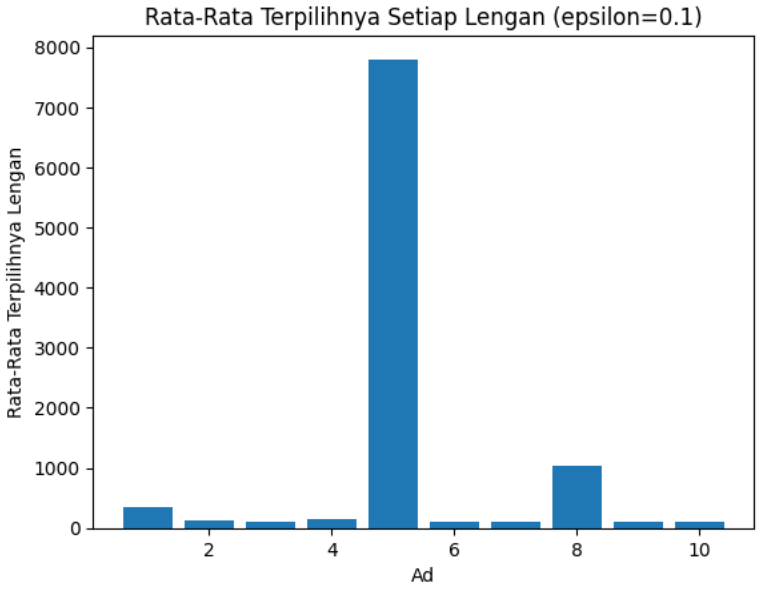
```
def choose_ad():
    if np.random.uniform() < epsilon:
        return random.randrange(num_ads)
    else:
        max_val = max(q_values)
        return q_values.index(max_val)
```

Program tersebut akan memberikan sebuah angka secara acak dari 0 hingga 1 dengan kemungkinan terambilnya setiap angka sama. Jika angka yang terambil lebih kecil dari epsilon, maka algoritma akan melakukan langkah eksplorasi. Sebaliknya, jika angka yang terambil lebih besar dari

epsilon maka akan dilakukan eksploitasi. Setelah program dieksekusi, berikut adalah plot hasil rata-rata dari setiap langkah untuk keempat epsilon tersebut.

a. Total hadiah

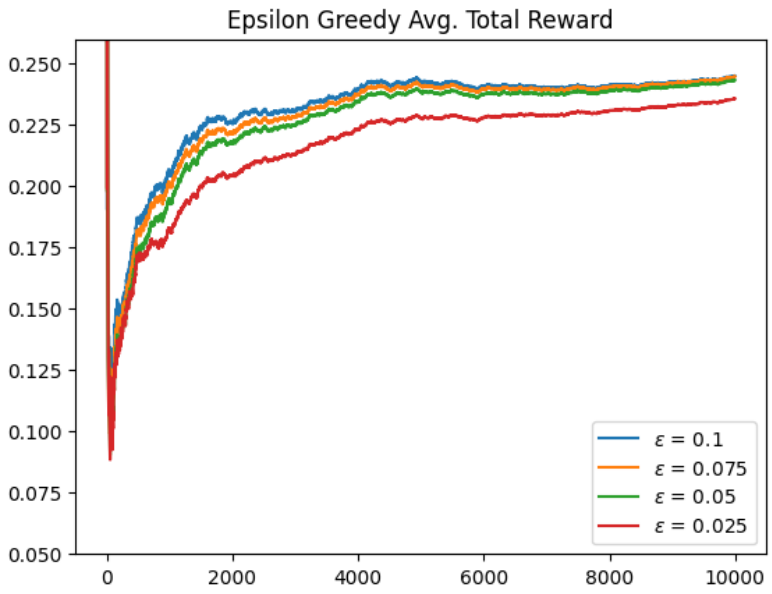
Hasil simulasi dari algoritma ini menunjukkan bahwa Ad5 merupakan jenis iklan yang memberikan hasil paling baik, sama seperti algoritma A/B/n testing. Berikut adalah plot rata-rata jumlah pemilihan iklan selama algoritma dijalankan dari epsilon 0.1. Epsilon lain juga memberikan hasil yang serupa. Jika ingin melihat plot dari epsilon lain, dapat dilihat pada bagian lampiran.



Gambar 9. Grafik total rata-rata pemilihan lengan epsilon 0.1

Dari gambar di atas terlihat bahwa algoritma melakukan eksploitasi yang jauh lebih banyak ke Ad5, sehingga dapat disimpulkan bahwa Ad5 merupakan jenis iklan yang paling baik. Selain itu juga Ad8 juga sama seperti algoritma sebelumnya memberikan hasil yang cukup baik

dikarenakan performa Ad8 ini juga merupakan iklan dengan performa terbaik ke-2 setelah Ad5 jika dilihat pada plot himpunan data awal. Ada kemungkinan pada langkah tertentu, algoritma menjalankan eksploitasi pada Ad8, lalu berubah ke Ad5 karena memberikan nilai Q tertinggi atau penyesalan paling rendah



Gambar 10. Grafik rata-rata hadiah

Pada grafik di atas, nilai Avg. Reward berkisar antara 0 dan 1. Nilai 0 menunjukkan bahwa algoritma tidak berhasil memilih jenis iklan yang menghasilkan hasil optimal pada langkah tersebut, sedangkan nilai 1 menunjukkan bahwa algoritma berhasil memilih jenis iklan yang menghasilkan hasil optimal pada langkah tersebut. Pada awalnya (sekitar titik 0), nilai Avg. Reward mencapai 0.025. Hal ini terjadi karena dalam langkah awal, algoritma melakukan eksplorasi di mana ia secara acak memilih 10 jenis iklan, dan salah satu iklan yang dipilih memiliki nilai 1. Oleh karena itu, nilai Avg. Reward awalnya adalah

0.025. Namun, setelah beberapa percobaan, algoritma memilih jenis iklan lain yang menghasilkan nilai 0, sehingga nilai Avg. Reward mendekati 0. Bagian awal ini sama dengan langkah pertama yang terjadi pada algoritma A/B/n test. Algoritma belajar dari riwayat ini dan mengambil tindakan untuk memilih iklan yang memiliki potensi lebih baik dari iklan lainnya. Seiring berjalannya waktu, sekitar langkah ke-10000, Avg. Reward mulai meningkat, menandakan bahwa algoritma berhasil memilih iklan yang lebih baik.

Dari grafik tersebut tidak terlihat jenis iklan mana yang memberikan hasil yang paling optimal karena yang divisualisasikan pada grafik di atas adalah besaran Avg. *Reward* dari setiap langkah. Jenis iklan optimal dapat dilihat dari iklan dengan nilai Q paling tinggi. Dari setiap epsilon, jenis iklan dengan performa terbaik adalah ad5 seperti yang ditampilkan pada program. Namun, setiap epsilon memberikan rata-rata total hadiah yang berbeda karena setiap pemilihan epsilon berpengaruh pada kinerja algoritma.

Tabel 9. Nilai ATR

no	Algoritma	ATR
1	Epsilon 0.1	2447.63
2	Epsilon 0.075	2434.762
3	Epsilon 0.05	2432.48
4	Epsilon 0.025	2363.888

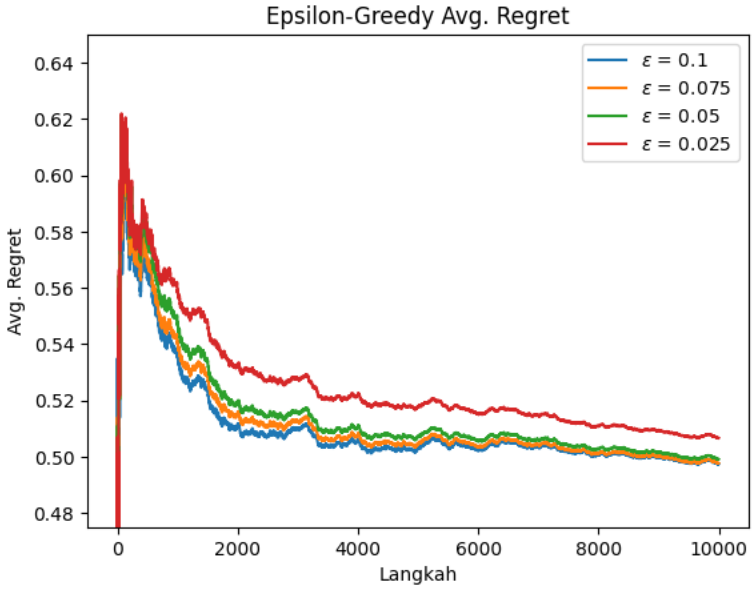
Pada **tabel 9** terlihat bahwa rata-rata nilai CTR yang didapat berkisar pada 2400, dengan nilai paling tinggi pada epsilon 0.1 dan total rata-rata total CTRnya adalah 2447.63.

b. Penyesalan

Tabel 10. Nilai Penyesalan

no	Algoritma	Regret
1	Epsilon 0.1	0.496881
2	Epsilon 0.075	0.497312
3	Epsilon 0.05	0.502262
4	Epsilon 0.025	0.503778

Pada **tabel 10** terlihat bahwa epsilon 0.1 memberikan hasil ATR tertinggi dan penyesalan terkecil.



Gambar 11. Grafik rata-rata regret epsilon greedy

Secara visual, grafik Avg. Regret ini merupakan kebalikan dari Avg. Reward. Hal itu disebabkan karena setiap *loss* yang diterima oleh pelajar, akan mengurangi rata-rata hadiah yang diterima oleh pelajar.

4.2.3 Adaptive epsilon greedy

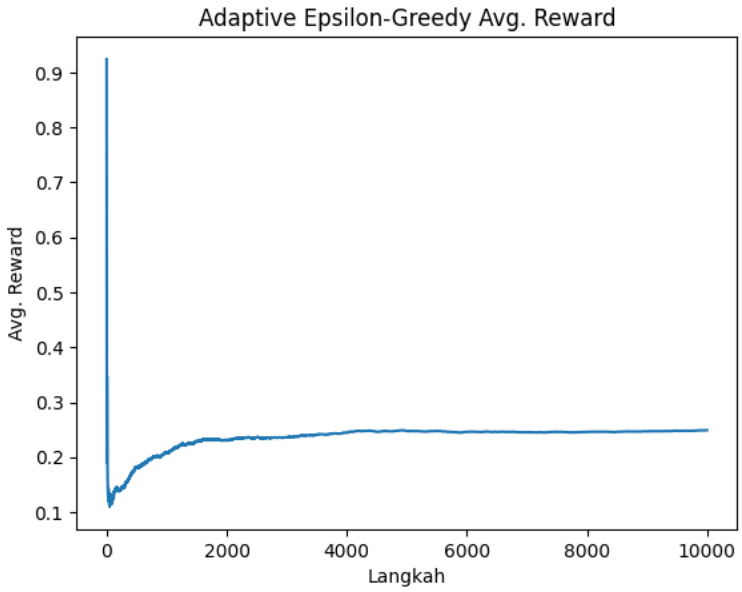
Pada algoritma ini parameter l diberi nilai 10 dan f diberi nilai 7. Pengambilan nilai parameter ini didapat dengan mecoba berbagai kombinasi nilai *l* dan *f*, sehingga didapat nilai paramater di atas merupakan nilai yang cukup optimal. Setelah menentukan nilai parameter, nilai epsilon awalnya adalah 0.1. Nilai epsilon awal tidak memiliki pengaruh yang besar pada algoritma ini karena nilai epsilon akan berubah seiring langkah.

$$l = 10$$

$$f = 7$$

$$\varepsilon = 0.1$$

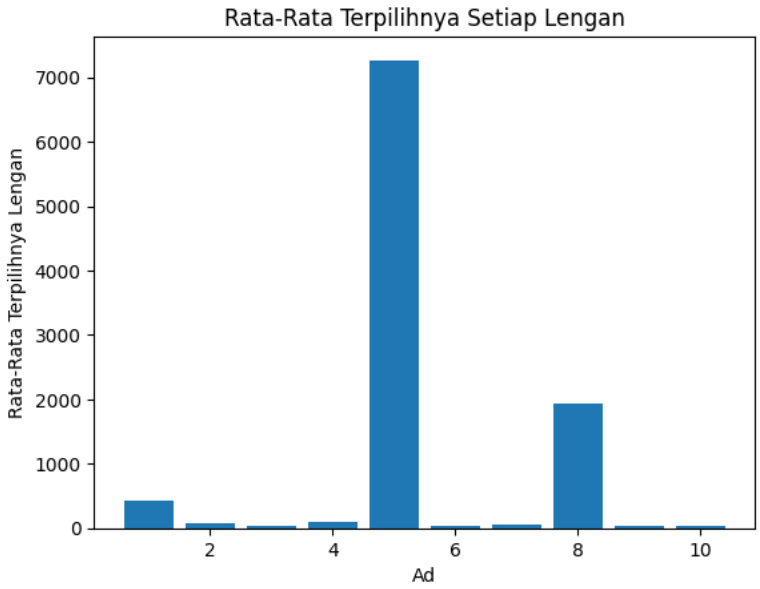
a. Total hadiah



Gambar 12. Rata-rata hadiah adaptive epsilon greedy

Sama seperti kedua algoritma sebelumnya, pada bagian awal langkah, nilai Avg. Reward tidak stabil. Setelah sekian langkah, algoritma mulai stabil dan Avg. Rewardnya mulai naik terus hingga

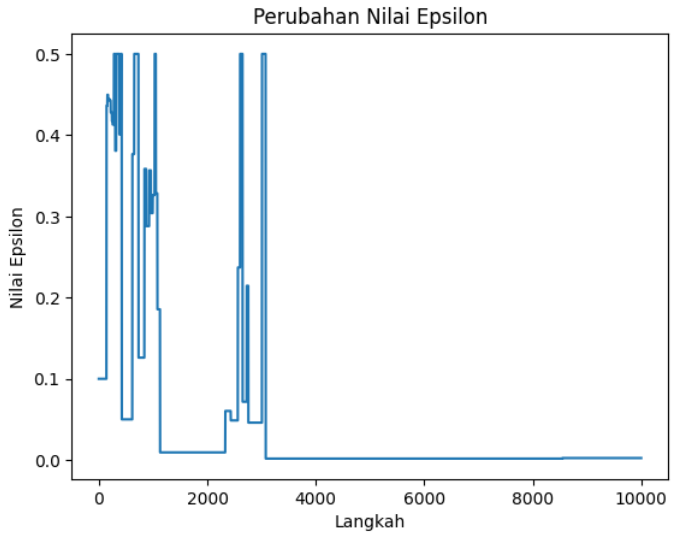
akhir langkah. Setelah perubahan drastis pada awal langkah, algoritma adaptive epsilon greedy ini cenderung lebih konsisten naik hingga langkah terakhir dan nilainya berkisar di atas 0.2. Total CTR yang didapat dari algoritma setelah di-*looping* sebanyak 500 kali adalah 2468.152.



Gambar 13. Rata-rata total pemilihan tiap iklan

Dari gambar di atas, terlihat bahwa iklan jenis ke 5 (ad5) terpilih jauh lebih banyak daripada jenis iklan yang lain. Hal ini menandakan bahwa jenis iklan tersebut adalah yang paling optimal dibanding jenis iklan yang lain karena sering dieksploitasi.

b. Perubahan nilai Epsilon



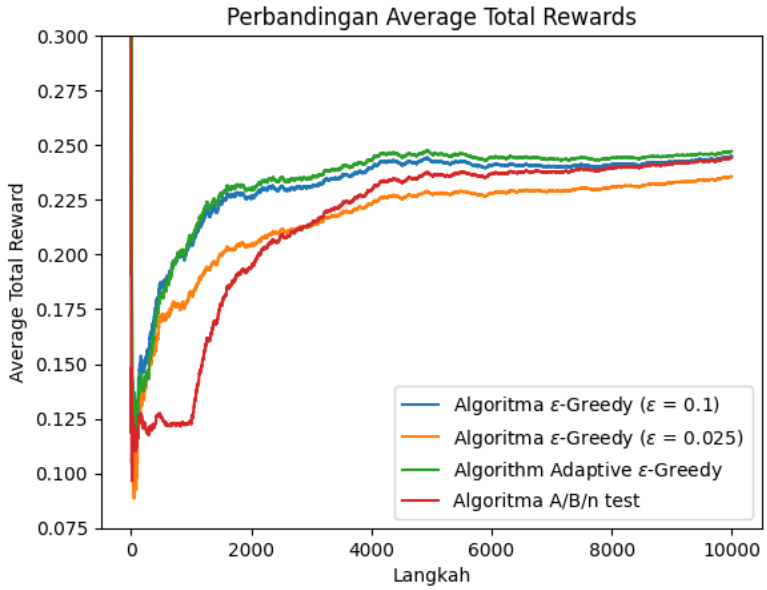
Gambar 14. Grafik perubahan nilai epsilon

Dari grafik di atas, terlihat bahwa nilai epsilon sering berubah. Hal ini menandakan bahwa algoritma beradaptasi dengan lingkungan dengan mengubah nilai epsilonnya. Nilai epsilon tersebut ada pada selang 0 hingga 0.5. Diketahui bahwa nilai epsilon berpengaruh pada jumlah eksploitasi dan eksplorasi pada algoritma epsilon greedy. Pada saat Q_{value} tinggi, algoritma akan beradaptasi sehingga menurunkan nilai epsilon agar algoritma terus mengeksploitasi jenis iklan yang memberikan hasil optimal. Sebaliknya, jika Q_{value} rendah, algoritma akan menaikkan nilai epsilon dengan maksimal 0.5 agar algoritma melakukan eksplorasi lengan yang lain yang mungkin memberikan hasil yang lebih baik.

4.3 Perbandingan

Untuk memastikan bahwa algoritma epsilon greedy yang digunakan memberikan hasil optimal, epsilon greedy akan dibandingkan dengan algoritma MAB yang lain seperti A/B testing.

4.3.1 Nilai Avg. *Reward*



Gambar 15. Perbandingan Avg. total rewards

Pada grafik rata-rata hadiah yang didapat dari ketiga algoritma di atas, nilai Avg. *reward* dari algoritma adaptive epsilon greedy lebih tinggi dibanding 2 algoritma lainnya. Pada awal langkah, nilai Avg. *Reward* ketiga algoritma sama-sama mulai dari 1 lalu mendekati nol dan perlahan naik menuju nilai optimal dari kedua algoritma tersebut. Namun, adaptive epsilon greedy lebih cepat naik dibanding kedua grafik 2 algoritma lainnya. epsilon greedy dengan epsilon 0.1 dan adaptive epsilon greedy memiliki grafik dengan kenaikan yang hampir sama yang menunjukkan bahwa

performa kedua algoritma tersebut khususnya saat epsilon greedy menggunakan epsilon 0.1, memberikan hasil yang sama optimalnya. Algoritma dengan hasil yang paling buruk merupakan algoritma A/B/n testing dengan grafik yang naiknya paling lama dibanding 2 algoritma lainnya.

4.3.2 Rata-rata total hadiah

Berdasarkan rata-rata total *hadiah* yang didapat dari kedua algoritma tersebut, akan dibandingkan hasil dari performa algoritma epsilon greedy dengan A/B testing dengan melihat nilai total *Average reward* yang didapat dari setiap algoritma.

Tabel 11. Tabel Nilai ATR

no	Algoritma	CTR
1	A/B/n testing	2429.21
2	Epsilon 0.1	2447.63
3	Epsilon 0.075	2434.76
4	Epsilon 0.05	2432.48
5	Epsilon 0.025	2363.88
6	Adaptive Epsilon Greedy	2468.15

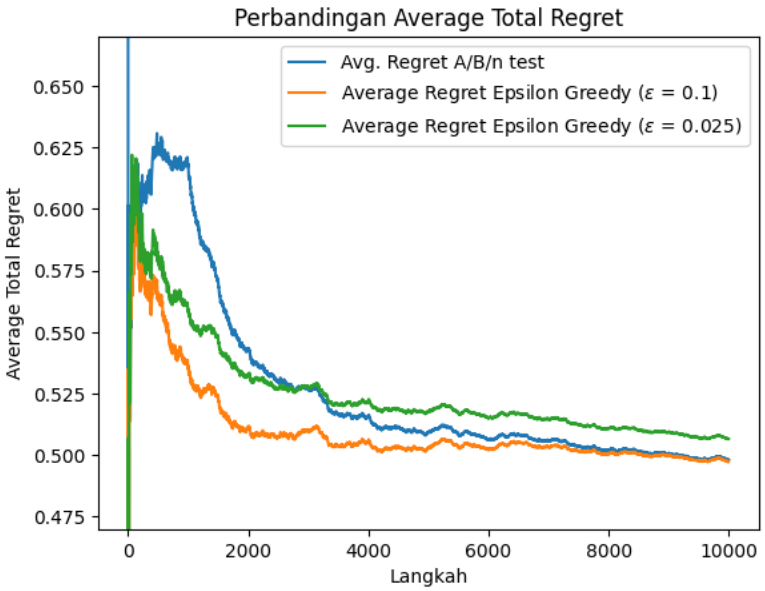
Dari hasil tersebut, nilai ATR dari ketiga algoritma memberikan hasil yang serupa yang berkisar pada 2400-an. Lebih tepatnya, algoritma adaptive epsilon greedy memberikan hasil paling tinggi dengan total CTR 2468.15 lalu yang kedua adalah epsilon greedy dengan epsilon 0.1 dengan total CTR 2447.63. Sebagai tambahan, berikut adalah nilai CTR maksimal dan minimal dari setiap algoritma

Tabel 12. Tabel Nilai min-max CTR

no	Algoritma	Min CTR	Max CTR
1	A/B/n testing	1177	2587
2	Epsilon 0.1	1719	2591
3	Epsilon 0.075	1675	2605
4	Epsilon 0.05	1654	2637
5	Epsilon 0.025	1622	2669
6	Adaptive Epsilon Greedy	1176	2686

Pada **tabel 12**, setiap algoritma memiliki selang nilai CTR yang cukup tinggi. Hal ini dikarenakan pada saat eksplorasi, algoritma memilih sembarang iklan yang menyebabkan perbedaan nilai CTR yang didapat. Jika diperhatikan pada algoritma adaptive epsilon greedy yang merupakan algoritma yang paling optimal berdasarkan rata-rata CTR, jarak antara minimal dan maksimal CTRnya cukup jauh. Hal ini dikarenakan perubahan epsilon yang terjadi sangat bergantung pada Q_{value} yang didapat. Algoritma bisa saja mendapatkan nilai Q_{value} kecil dan mengakibatkan epsilon membesar. Hal ini memaksa algoritma untuk melakukan eksplorasi yang lebih banyak dan tidak mengeksplorasi iklan yang optimal, dalam hal ini merupakan ad5.

4.3.3 *Penyesalan*



Gambar 16. Grafik rata-rata total regret

Pada bagian awal grafik Avg. Total Regret, algoritma epsilon greedy menunjukkan hasil yang lebih baik karena tidak melewati titik 0.7. Hal ini bisa disebabkan karena saat langkah awal, algoritma mendapatkan iklan yang memberikan hasil yang optimal sehingga algoritma epsilon greedy mengeksploitasi iklan tersebut. A/B testing memberikan hasil yang kurang baik di awal langkah karena algoritma masih menjalankan eksplorasi murni, sehingga saat memilih iklan secara acak dan mendapatkan iklan dengan performa yang buruk. Setelah sekian langkah, grafik penyesalan dari setiap algoritma terlihat mengalami penurunan. Epsilon greedy dengan epsilon 0.1 terlihat memiliki kurva dengan Average Total Regret paling kecil dibanding yang lain. Pada langkah ke 6000, setiap algoritma memiliki nilai penyesalan yang hampir sama. Hal ini terjadi karena pada saat tersebut algoritma

mendapatkan iklan yang memberikan hasil optimal dan perhitungan penyesalan juga semakin besar (“lihat rumus penyesalan”) yang membuat perubahan nilai penyesalan tidak seekstrem pada awal langkah.

BAB V
KESIMPULAN

5.1 Kesimpulan

Algoritma Multi-Armed Bandit (MAB) merupakan algoritma yang relatif mudah namun sangat *powerful* dalam mencari pilihan paling optimal dari beberapa pilihan yang tersedia. MAB juga merupakan algoritma yang cukup sering digunakan dalam berbagai bidang untuk membantu membuat keputusan yang tepat seperti bidang *marketing* (pemasaran). Beberapa metode yang dapat dipakai dalam algoritma ini adalah A/B testing dan epsilon greedy. Kedua metode ini merupakan metode yang paling sering digunakan dalam bidang pemasaran karena relatif mudah dan memberikan hasil yang cukup baik. Untuk lingkungan yang stasioner, kedua algoritma ini memberikan hasil yang hampir sama sehingga dilihat dari penyesalan maupun hadiah yang diterima. Namun, dalam lingkungan tidak stasioner, epsilon greedy lebih baik daripada A/B testing karena algoritma ini tetap melakukan eksplorasi dan memperbarui Q_{value} sehingga saat terjadi perubahan, epsilon greedy dapat menyesuaikan lengan baru yang memiliki nilai yang terbaik hingga maksimal penarikan atau pemilihan tercapai.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, disarankan untuk menggunakan algoritma epsilon greedy dalam menentukan *Click-through Rate* iklan daring yang optimal, terutama dalam lingkungan yang tidak stasioner. Hal ini dikarenakan algoritma epsilon greedy dapat menyesuaikan diri dengan perubahan yang terjadi dan memberikan hasil yang lebih baik dibandingkan dengan algoritma A/B testing. Namun, perlu diingat bahwa dalam menggunakan algoritma epsilon greedy, penting untuk menentukan nilai epsilon yang tepat agar algoritma dapat bekerja dengan optimal. Oleh karena itu, penulis menyarankan untuk menggunakan algoritma adaptive epsilon greedy sebagai alternatif untuk meminimalisir kesalahan dalam pemilihan nilai epsilon, karena adaptive epsilon greedy selalu memperbarui nilai epsilonnya.

DAFTAR PUSTAKA

Bhatnagar, Harshinta. (2013). Online Vs Traditional Advertisement Media- A Comparative Analysis.*Pacific Business Reviev International*. 6(4), 54-58.

Bilgin, Enes. (2020). *Mastering Reinforcement Learning with Python*. Birmingham. Packt Publishing.

Dinata, Rozzi Kesuma dan Novia Hasdyna. (2020). *Machine Learning*. Aceh: Unimal Press.

Gallo, Amy. *A Refresher on A/B Testing*. Harvard Business Review. https://hbr.org/2017/06/a-refresher-on-ab-testing (diakses pada tatnggal 18 Juni 2023 jam 20:00)

Kaibel, Chris and Torsten Biemann. (2021*).* Rethinking the Gold Standard With Multi-armed Bandits: Machine Learning Allocation Algorithms for Experiments. *Organizational Research Methods* .24(1), 78-103.

Lattimore, Tor and Szepesv´ari, Csaba. (2020). *Bandit Algorithms*. Cambridge: Cambridge University Press.

Mignon, Alexandre dos Santos and Ricardo Luis de Azevedo da Rocha. (2017). An Adaptive Implementation of ε-Greedy in Reinforcement Learning. *International Workshop on Adaptive Technology*. 109C.1146-1151.

Mohammed, Mohssen, Muhammad Badruddin Khan, and Eihab Bashier Mohammed Bashier. (2016). *Machine Learning: Algorithms and Applications*. Boca Raton: CRC Press.

Ponteves, Hadelin de. (2019). *AI Crash Course*. Birmingham: Packt Publishing.

Rafftery, Anna N., Joseph Jay Wiliams and Huiji Ying. (2019). Statistical Consequences of using Multi-armed Bandits to Conduct Adaptive Educational Experiments. *Journal of Educational Data Mining, Volume 11, No 1, 47-79.*

Robert, Steve. *Bandit Algorithms: Multi-Armed Bandit: Part 3*. Towards Data Science. https://towardsdatascience.com/bandit-algorithms-34fd7890cb18 (diakses tanggal 20 Maret 2023 jam 23:00 WIB)

White, John Myles. (2012). *Bandit Algorithms for Website Optimization*. Sebastopol: O'Reilly Media.

LAMPIRAN

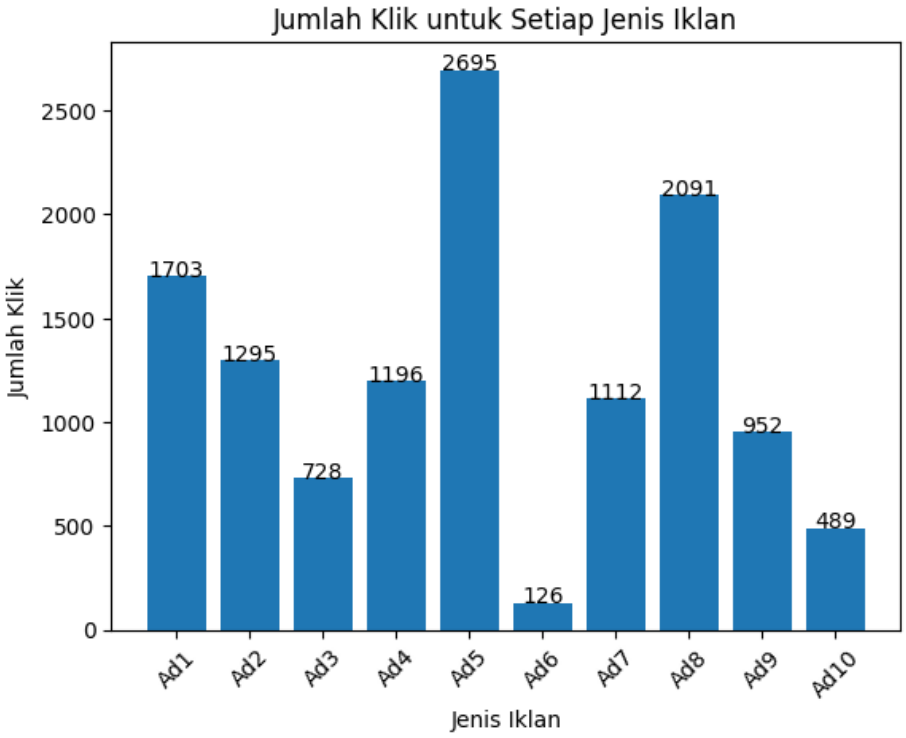
Lampiran 1. Himpunan data CTR iklan daring

Himpunan data yang digunakan pada tugas akhir ini dapat diakses melalui tautan berikut.

https://www.kaggle.com/datasets/akram24/ads-ctr-optimisation?resource=download

Lampiran 2. Plot Himpunan Data

```
import pandas as pd
import matplotlib.pyplot as plt

# Membaca dataset
df = pd.read_csv("D:/Tugas
Akhir/dataset/Ads_CTR_Optimisation.csv")

# Menghitung jumlah klik untuk setiap jenis iklan
click_counts = df.sum(axis=0)

# Menyiapkan data untuk plot
jenis_iklan = [f'Ad{i}' for i in range(1, 11)]  # Asumsikan
iklan dinamai Iklan-1 hingga Iklan-10
jumlah_klik = click_counts.values

# Membuat bar plot
plt.bar(jenis_iklan, jumlah_klik)
plt.xlabel('Jenis Iklan')
plt.ylabel('Jumlah Klik')
plt.title('Jumlah Klik untuk Setiap Jenis Iklan')
plt.xticks(rotation=45)  # Untuk memudahkan membaca label jenis
iklan

# Menambahkan nilai sebenarnya jumlah klik pada barplot
for i in range(len(jenis_iklan)):
    plt.text(i, jumlah_klik[i], jumlah_klik[i], ha = 'center')

plt.show()
```



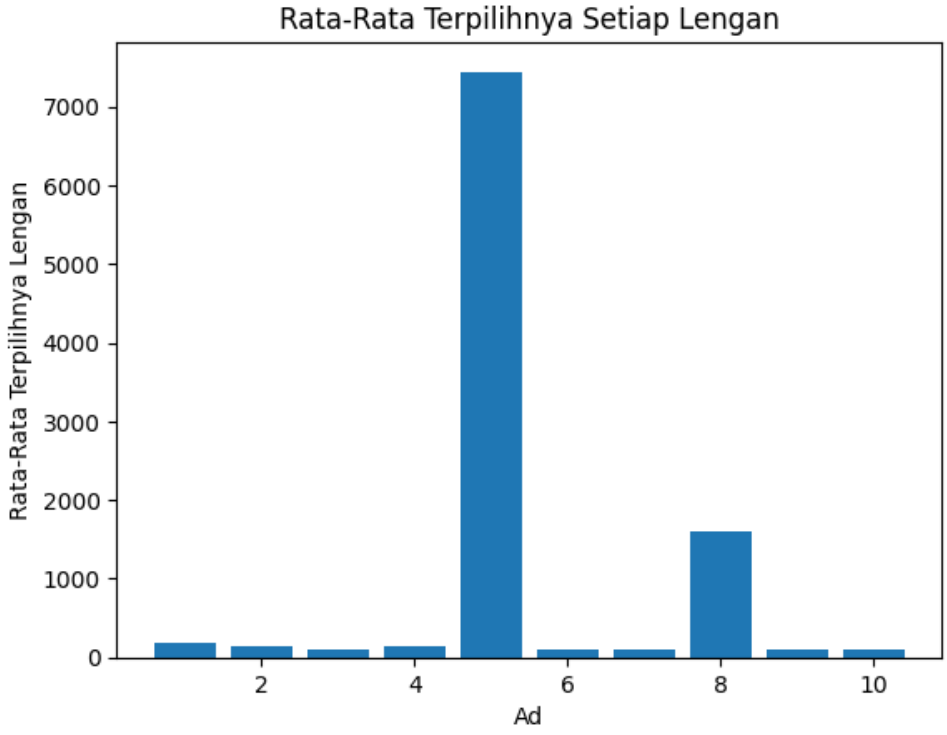
Lampiran 3. Program A/B/n Testing

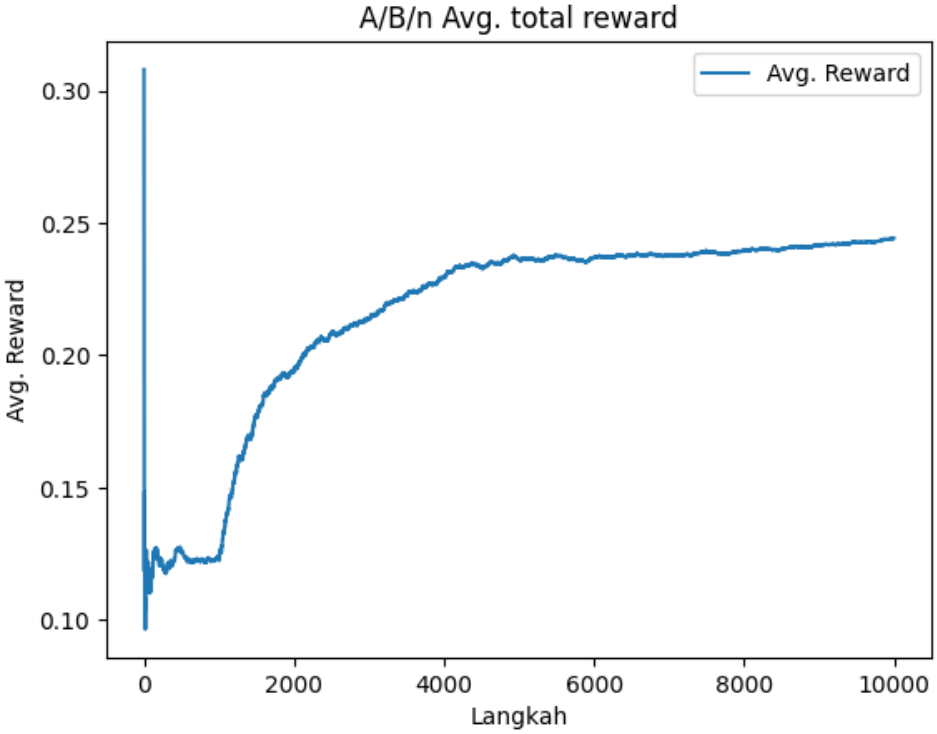
```
 1. import numpy as np
 2. import pandas as pd
 3. import matplotlib.pyplot as plt
 4. from scipy import stats
 5.
 6. df = pd.read_csv("D:/Tugas
Akhir/dataset/Ads_CTR_Optimisation.csv")
 7. ads = df.values.tolist()
 8. n_ads = len(ads[0])
 9.
10. n_test = 1000
11. n_prod = 9000
12. loop = 500
13. avg_total_rewards = []  # menyimpan average total rewards
setiap run
14. avg_rewards_all = []
15. avg_regred_AB_all = []
16.
17.
18. ad_choices = np.zeros((loop, n_ads))
19.
20. for run in range(loop):
21.     Q = np.zeros(n_ads)  # Q, action values
22.     N = np.zeros(n_ads)  # N, total impressions
23.     total_reward = 0
24.     avg_rewards = []  # menyimpan sata-rata reward
25.     avg_regred_AB = []
```

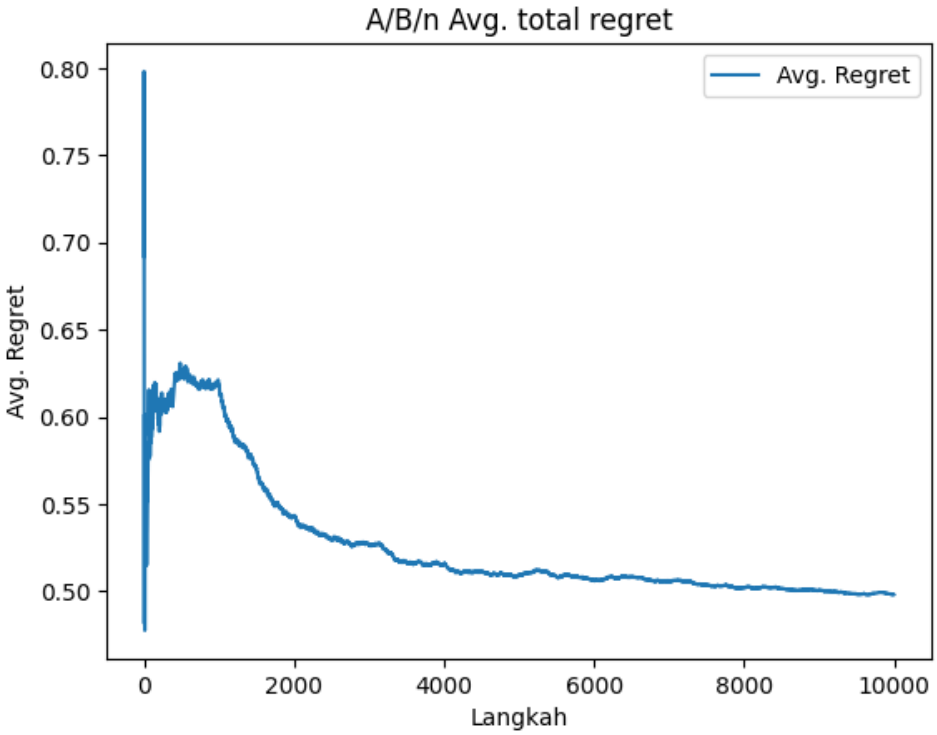
```
26.     total_regret = 0
27.
28.     # A/B/n test
29.     for i in range(n_test):
30.         ad_chosen = np.random.randint(n_ads)
31.         R = ads[i][ad_chosen]
32.         N[ad_chosen] += 1
33.         Q[ad_chosen] += (1 / N[ad_chosen]) * (R -
            Q[ad_chosen])
34.         total_reward += R
35.         avg_rewards.append(total_reward / (i+1))
36.
37.         optimal_reward = max(ads[i])
38.         total_regret += optimal_reward - R
39.         avg_regred_AB.append(total_regret / (i+1))
40.
41.         # Increment the count for the chosen ad
42.         ad_choices[run, ad_chosen] += 1
43.
44.     avg_reward_so_far = total_reward / n_test
45.
46.     best_ad_index = np.argmax(Q)  # mencari lengan terbaik
47.
48.
49.     ad_chosen = best_ad_index
50.
51.     for i in range(n_prod):
52.         if i+n_test < len(ads):
53.             R = ads[i+n_test][ad_chosen]
54.             total_reward += R
55.             avg_rewards.append(total_reward / (n_test + i +
1))
56.
57.             optimal_reward = max(ads[i+n_test])
58.             total_regret += optimal_reward - R
59.             avg_regred_AB.append(total_regret / (n_test + i
+ 1))
60.
61.             # Increment the count for the chosen ad
62.             ad_choices[run, ad_chosen] += 1
63.
64.     avg_total_rewards.append(total_reward)
65.     avg_rewards_all.append(avg_rewards)
66.
67.     avg_regred_AB_all.append(avg_regred_AB)
68.
69. print("Rata-rata total reward selama {} kali running:
    {:.2f}".format(loop,
            np.mean(avg_total_rewards)))
70. print('Min CTR: ', min(avg_total_rewards))
71. print('Max CTR: ', max(avg_total_rewards))
```

```
72.
73.
74. avg_ad_choices = ad_choices.mean(axis=0)
75. plt.bar(range(1, n_ads+1), avg_ad_choices)
76. plt.xlabel('Ad')
77. plt.ylabel('Rata-Rata Terpilihnya Lengan')
78. plt.title('Rata-Rata Terpilihnya Setiap Lengan')
79. plt.show()
80.
81. # Plot rata-rata rewards dan regrets
82. plt.plot(np.mean(avg_rewards_all, axis=0), label="Avg.
     Reward")
83. plt.title("A/B/n Avg. total reward")
84. plt.xlabel('Langkah')
85. plt.ylabel('Avg. Reward')
86. plt.legend()
87. plt.show()
88.
89. plt.plot(np.mean(avg_regred_AB_all, axis=0), label="Avg.
     Regret")
90. plt.title("A/B/n Avg. total regret")
91. plt.xlabel('Langkah')
92. plt.ylabel('Avg. Regret')
93. plt.legend()
94. plt.show()
```

```
Hasil
Rata-rata total reward selama 500 kali running: 2442.81
Min CTR:  1171
Max CTR:  2582
```




A/B/n Avg. total reward
Avg. Reward
0.30
0.25
0.20
0.15
0.10
Avg. Reward
0
2000
4000
6000
8000
10000
Langkah


A/B/n Avg. total regret
Avg. Regret
0.80
0.75
0.70
0.65
0.60
0.55
0.50
Avg. Regret
0
2000
4000
6000
8000
10000
Langkah

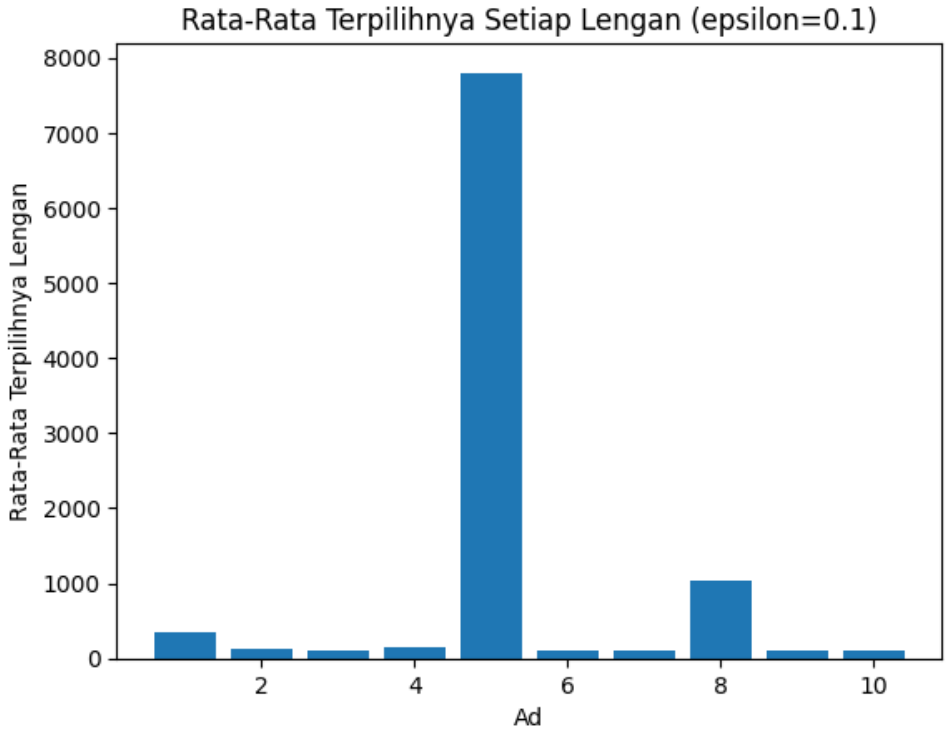
Lampiran 4. Epsilon greedy

```
 1. import numpy as np
 2. import pandas as pd
 3. import timeit
 4. import matplotlib.pyplot as plt
 5. import random
 6. from scipy import stats
 7.
 8. def choose_ad():
 9.     if np.random.uniform() < epsilon:
10.         return random.randrange(num_ads)
11.     else:
12.         max_val = np.max(q_values)
13.         return np.argmax(q_values)
14.
15. awal = timeit.default_timer()
16.
17. df = pd.read_csv("D:/Tugas
Akhir/dataset/Ads_CTR_Optimisation.csv")
18. ads = df.values
19. num_ads = ads.shape[1]
20. num_Langkahs = ads.shape[0]
21.
22. epsilons = [0.1, 0.075, 0.05, 0.025]
23. avg_total_rewards = []
24. avg_regrets = []
25. avg_reward_records = []
26.
27. for epsilon in epsilons:
28.     total_rewards = np.zeros(500)
29.     regrets = np.zeros((500, num_Langkahs))
30.     reward_records = []
31.     ad_choices = np.zeros((500, num_ads))
32.
33.     for run in range(500):
34.         q_values = np.zeros(num_ads)
35.         n_values = np.zeros(num_ads)
36.         total_reward = 0
37.         regret = 0
38.
39.         reward_record = [0] * num_Langkahs
40.         for i in range(num_Langkahs):
41.             ad_chosen = choose_ad()
42.             ad_choices[run, ad_chosen] += 1
43.
44.             reward = ads[i, ad_chosen]
45.             total_reward += reward
46.
47.             n_values[ad_chosen] += 1
48.             q_values[ad_chosen] += (reward -
```

```
                    q_values[ad_chosen]) /
                    n_values[ad_chosen]

                optimal_reward = ads[i].max()
                regret += optimal_reward - reward
                regrets[run, i] = regret / (i + 1)

                reward_record[i] = total_reward / (i + 1)

            total_rewards[run] = total_reward
            reward_records.append(reward_record)

        avg_total_reward = total_rewards.mean()
        avg_regret = regrets.mean(axis=0)

        avg_total_rewards.append(avg_total_reward)
        avg_regrets.append(avg_regret)

        avg_reward_record = [sum(x) / len(x) for x in
            zip(*reward_records)]
        avg_reward_records.append(avg_reward_record)

        print("Epsilon:", epsilon)
        print("Average Total Reward:", avg_total_reward)
        print("Average Regret:", avg_regret)
        print ('Min CTR: ', min(total_rewards))
        print ('Max CTR: ', max(total_rewards))
        print(f"Optimal Ad for epsilon {epsilon}:
            {np.argmax(q_values)+1}")


        avg_ad_choices = ad_choices.mean(axis=0)
        plt.bar(range(1, num_ads+1), avg_ad_choices)
        plt.xlabel('Ad')
        plt.ylabel('Rata-Rata Terpilihnya Lengan')
        plt.title(f'Rata-Rata Terpilihnya Setiap Lengan
            (epsilon={epsilon})')
        plt.show()

        print('-------------------------------------------')

# Plot rata-rata reward
for i in range(len(epsilons)):
        plt.plot(avg_reward_records[i], label=f'$\epsilon$ =
            {epsilons[i]}')
plt.title("Epsilon Greedy Avg. Total Reward")
plt.legend()
plt.ylim(0.05, 0.26)  # Set batas sumbu y
plt.show()

# Plot rata-rata regret
```

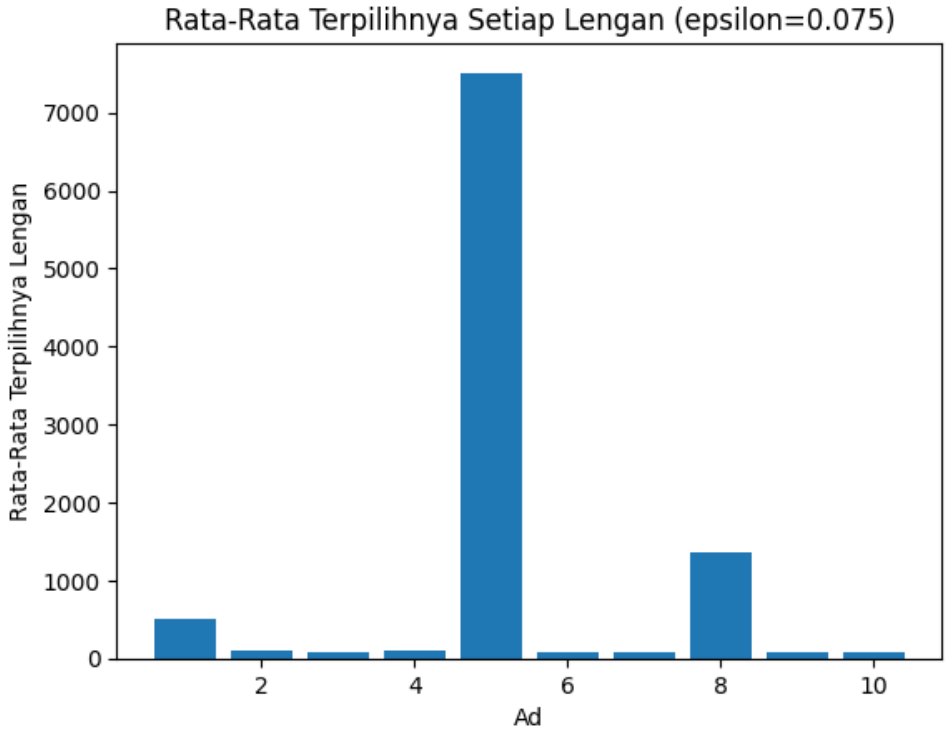
```
for i in range(len(epsilons)):
    plt.plot(avg_regrets[i], label=f'$\epsilon$ =
        {epsilons[i]}')
plt.title("Epsilon-Greedy Avg. Regret")
plt.xlabel('Langkah')
plt.ylabel('Avg. Regret')
plt.legend()
plt.ylim(0.475, 0.65)  # Set batas sumbu y
plt.show()

akhir = timeit.default_timer()
print("Running time =", (akhir - awal) / 60, "menit")
```

```
Epsilon: 0.1

Average Total Reward: 2450.044

Average Regret: [0.094      0.535      0.35666667 ... 0.49730046
0.49734613 0.4973956 ]

Min CTR:  1859.0

Max CTR:  2592.0

Optimal Ad for epsilon 0.1: 5
```

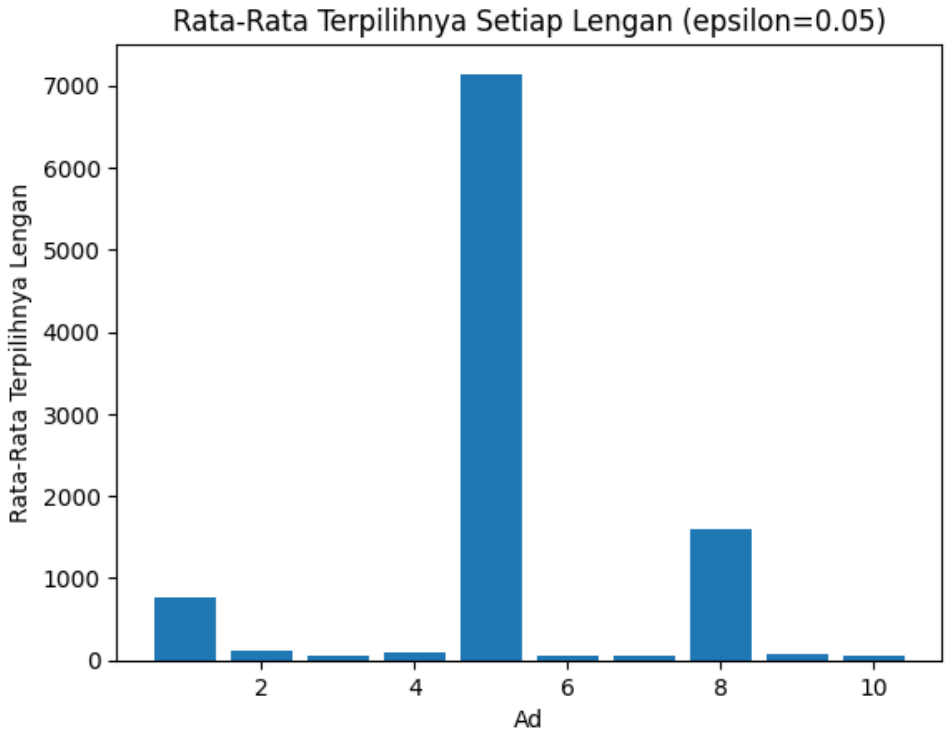



```
------------------------------------------------

Epsilon: 0.075
```

```
Average Total Reward: 2447.28

Average Regret: [0.038      0.514      0.34266667 ... 0.49757892
0.49762276 0.497672  ]

Min CTR:  1800.0

Max CTR:  2614.0

Optimal Ad for epsilon 0.075: 5
```




```
------------------------------------------------

Epsilon: 0.05

Average Total Reward: 2433.206

Average Regret: [0.036      0.511      0.34066667 ... 0.4989912
0.4990297  0.4990794 ]

Min CTR:  1543.0

Max CTR:  2632.0

Optimal Ad for epsilon 0.05: 5
```



Epsilon: 0.025

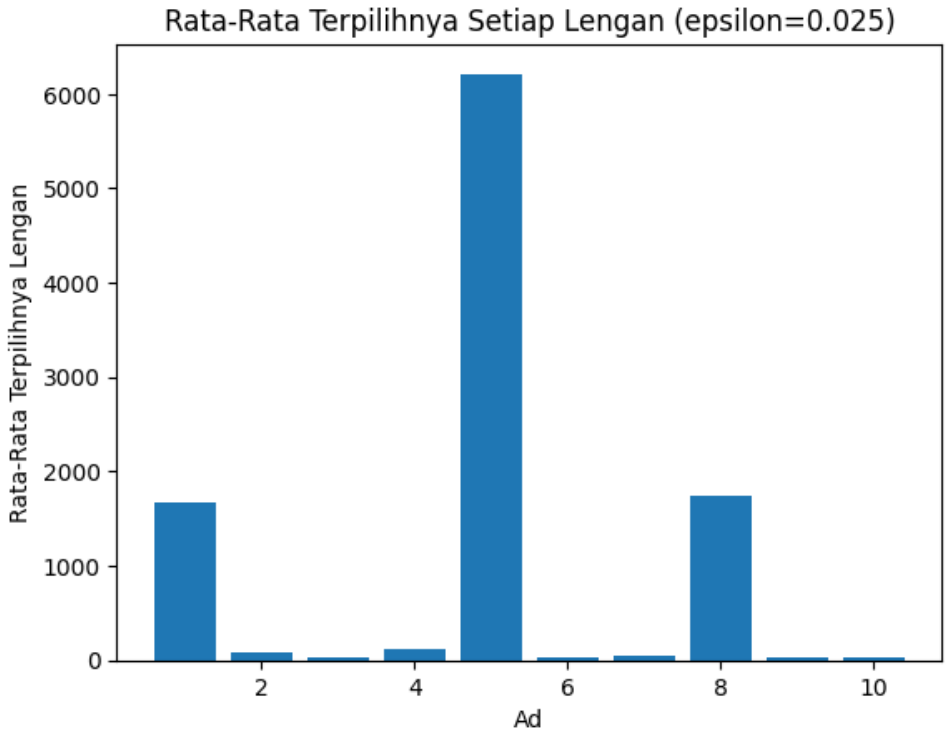
Average Total Reward: 2357.76

Average Regret: [0.018 0.507 0.338 ... 0.50654631 0.50657486 0.506624]

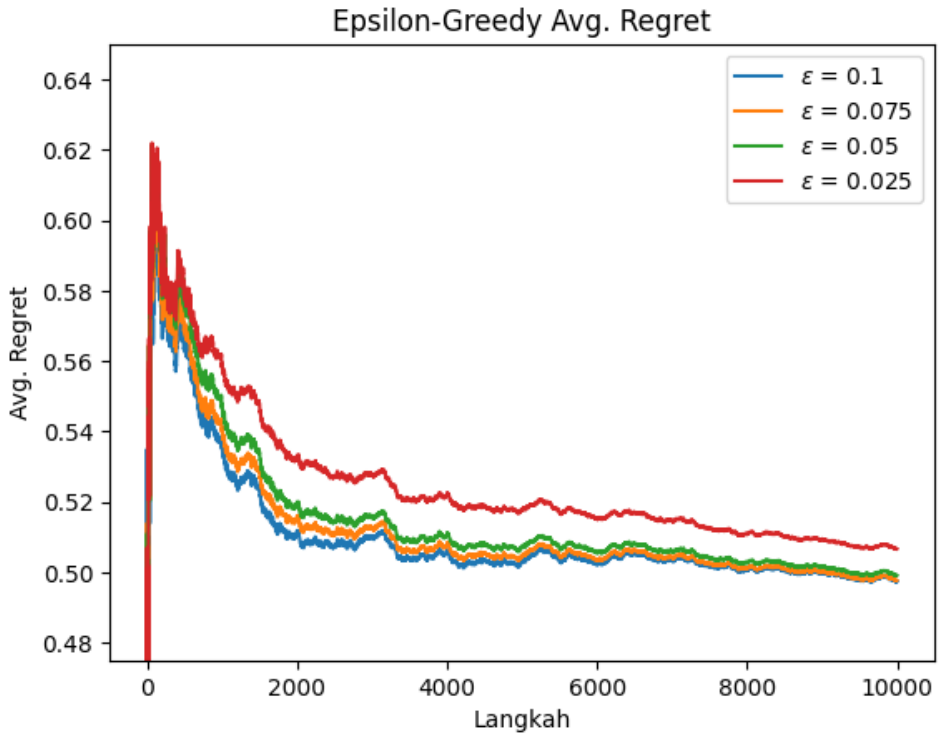
Min CTR: 1175.0

Max CTR: 2666.0

Optimal Ad for epsilon 0.025: 5


Rata-Rata Terpilihnya Setiap Lengan (epsilon=0.025)
Rata-Rata Terpilihnya Lengan
Ad
0
1000
2000
3000
4000
5000
6000
2
4
6
8
10


Epsilon Greedy Avg. Total Reward
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0
2000
4000
6000
8000
10000
ε = 0.1
ε = 0.075
ε = 0.05
ε = 0.025




```
Running time = 12.764325841666668 menit
```

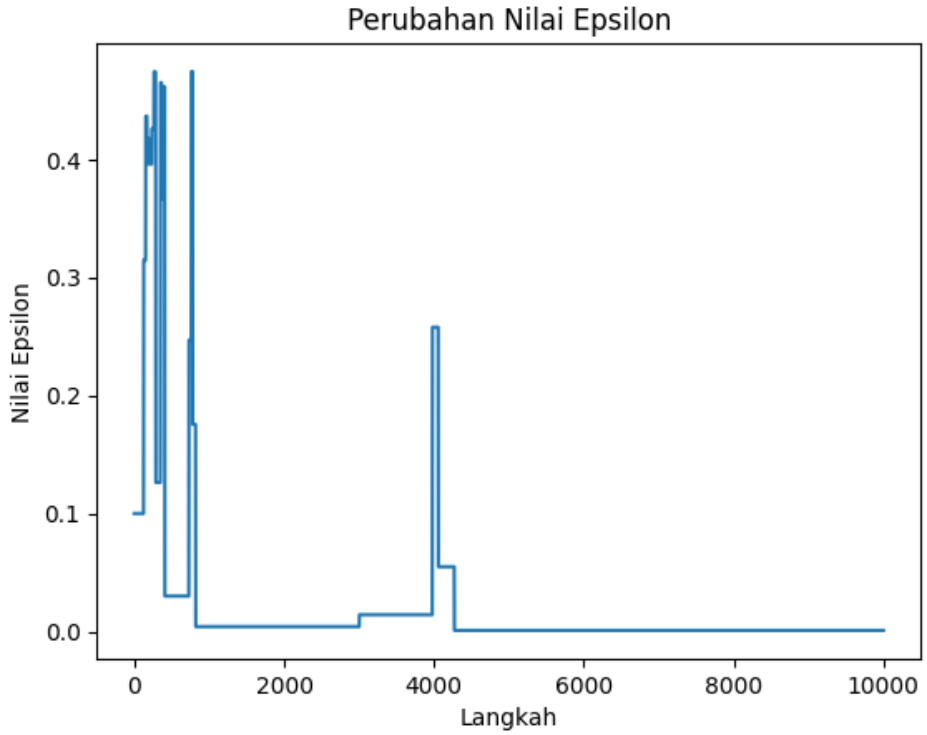
Lampiran 5. Adaptive epsilon greedy (1 kali)

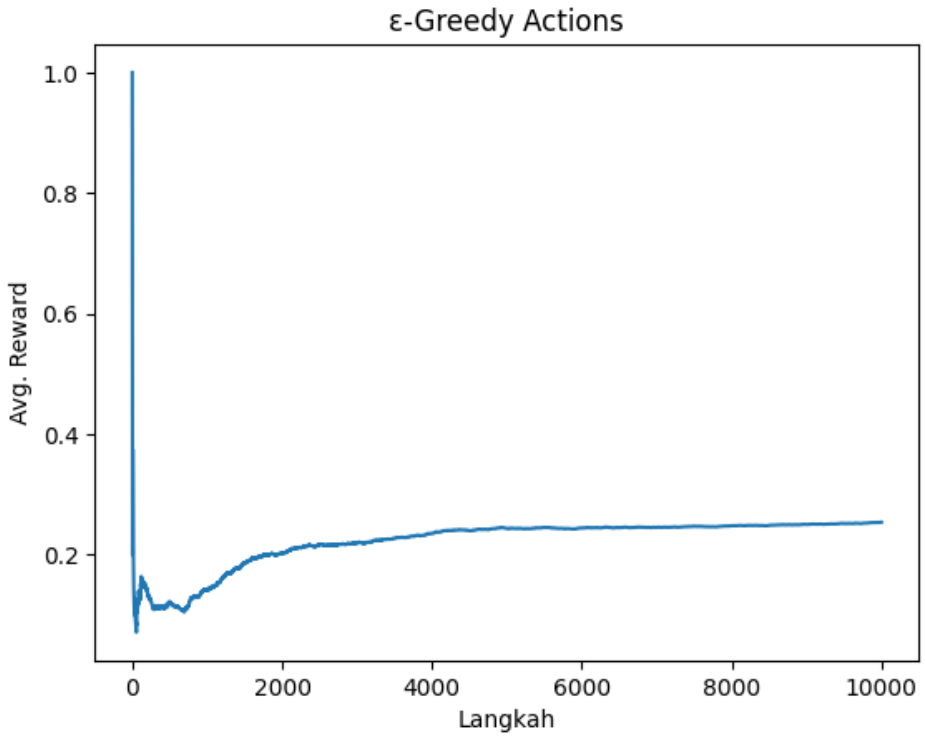
```
 1. import numpy as np
 2. import pandas as pd
 3. import matplotlib.pyplot as plt
 4. import math
 5.
 6. # Read the dataset from the CSV file
 7. df = pd.read_csv("D:/Tugas
Akhir/dataset/Ads_CTR_Optimisation.csv")
 8.
 9. # Number of ads
10. num_ads = 10
11.
12. # Parameter l dan f
13. l = 10
14. f = 7
15.
16. # Initialize Q-values for each ad
17. Q_values = np.zeros(num_ads)
18.
19. # Initialize epsilon and a list to store epsilon values
20. epsilon = 0.1
21. epsilon_values = [epsilon]
22. epsilon_history = [0.1]
```

```
23.
24. # To store the results of ad selection at each time step
25. selected_ads = []
26.
27. # To count how many times each ad is selected
28. ad_counts = np.zeros(num_ads)
29.
30. # Initialize variables for calculating epsilon
31. max_prev = Q_values.copy()
32. k = 0
33.
34. # Total click-through rate (CTR)
35. total_CTR = 0
36.
37. # Number of impressions (rounds)
38. num_rounds = len(df)
39.
40. # Lists to store cumulative rewards for each epsilon value
41. avg_rewards = []
42.
43. for t in range(num_rounds):
44.     if np.random.rand() <= epsilon:
45.         # Exploration mode: Choose an ad randomly
46.         ad = np.random.choice(num_ads)
47.         k += 1
48.         if k == 1:
49.             max_curr = np.max(Q_values)
50.             delta = (max_curr - max_prev[ad]) * f
51.             if delta > 0:
52.                 epsilon = (1 / (1 + math.exp(-2 * delta)))
- 0.5
53.             elif delta < 0:
54.                 epsilon = 0.5
55.             max_prev[ad] = max_curr
56.             k = 0
57.
58.             # Append the new epsilon value to the list
59.             epsilon_values.append(epsilon)
60.     else:
61.         # Exploitation mode: Choose the ad with the highest
Q-value
62.         ad = np.argmax(Q_values)
63.     epsilon_history.append(epsilon)
64.     selected_ads.append(ad)
65.     reward = df.values[t, ad]
66.
67.     # Update Q-value for the selected ad and add reward to
total_CTR
68.     Q_values[ad] += (reward - Q_values[ad]) /
(ad_counts[ad] + 1)
69.     ad_counts[ad] += 1
```

```
70.
71.     total_CTR += df.values[t, ad]
72.
73.     # Calculate average reward so far and add it to the
list
74.     avg_reward_so_far = total_CTR / (t + 1)
75.     avg_rewards.append(avg_reward_so_far)
76.
77. # Print the total Click-Through Rate (CTR)
78. print(f"Total CTR: {total_CTR}")
79.
80.
81. # Plot epsilon values over time (similar to the previous
program)
82. plt.plot(range(len(epsilon_history)), epsilon_history)
83. plt.xlabel("Langkah")
84. plt.ylabel("Nilai Epsilon")
85. plt.title("Perubahan Nilai Epsilon")
86. plt.show()
87.
```

```
Hasil
Total CTR: 2540
Total CTR: 2642
```





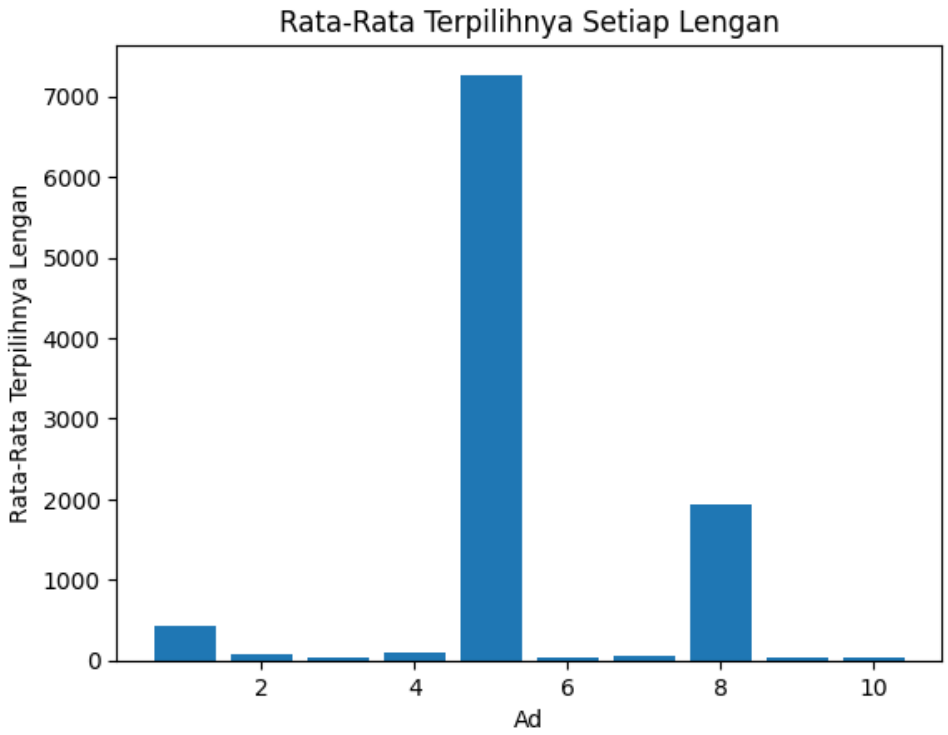
Lampiran 6. Adaptive epsilon greedy (500 kali)

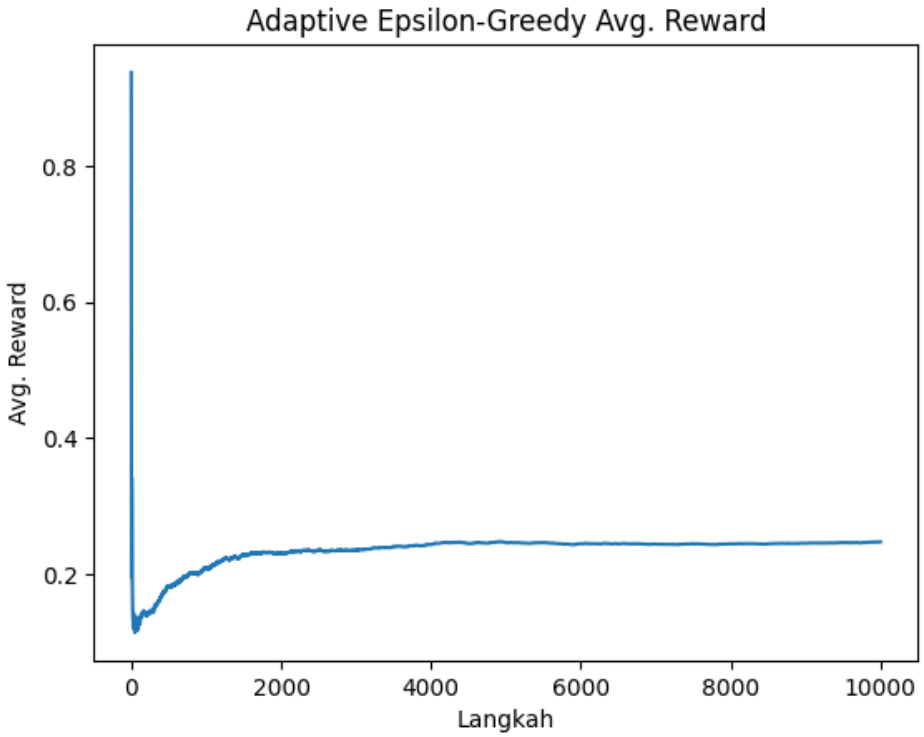
```
 1. import numpy as np
 2. import pandas as pd
 3. import matplotlib.pyplot as plt
 4. import math
 5. from scipy import stats
 6.
 7. def choose_ad():
 8.     if np.random.uniform() < epsilon:
 9.         return random.randrange(num_ads)
10.     else:
11.         max_val = np.max(q_values)
12.         return np.argmax(q_values)
13.
14. df = pd.read_csv("D:/Tugas
Akhir/dataset/Ads_CTR_Optimisation.csv")
15. ads = df.values
16. num_ads = ads.shape[1]
17. num_Langkahs = ads.shape[0]
18.
19. l = 10
20. f = 7
21.
22. total_CTRs = []
23. all_avg_rewards = []
24.
25. # Initialize an array to store the number of times each ad
is chosen
```

```
ad_choices = np.zeros((500, num_ads))

for run in range(500):
    Q_values = np.zeros(num_ads)
    epsilon = 0.1
    selected_ads = []
    ad_counts = np.zeros(num_ads)
    max_prev = Q_values.copy()
    k = 0

    total_CTR = 0
    num_rounds = len(df)
    avg_rewards = []

    for t in range(num_rounds):
        if np.random.rand() <= epsilon:
            ad = np.random.choice(num_ads)
            k += 1
            if k == 1:
                max_curr = np.max(Q_values)
                delta = (max_curr - max_prev[ad]) * f
                if delta > 0:
                    epsilon = (1 / (1 + math.exp(-2 *
                        delta))) - 0.5
                elif delta < 0:
                    epsilon = 0.5
                max_prev[ad] = max_curr
                k = 0

        else:
            ad = np.argmax(Q_values)

        # Increment the count for the chosen ad
        ad_choices[run, ad] += 1

        selected_ads.append(ad)
        reward = df.values[t, ad]

        Q_values[ad] += (reward - Q_values[ad]) /
         (ad_counts[ad] + 1)
        ad_counts[ad] += 1

        total_CTR += df.values[t, ad]

        avg_reward_so_far = total_CTR / (t + 1)
        avg_rewards.append(avg_reward_so_far)

    total_CTRs.append(total_CTR)
    all_avg_rewards.append(avg_rewards)

avg_total_CTR = np.mean(total_CTRs)
```

```
avg_avg_rewards = np.mean(all_avg_rewards, axis=0)

print(f"Average Total CTR: {avg_total_CTR}")
print('Min CTR: ', min(total_CTRs))
print('Max CTR: ', max(total_CTRs))


avg_ad_choices = ad_choices.mean(axis=0)
plt.bar(range(1, num_ads+1), avg_ad_choices)
plt.xlabel('Ad')
plt.ylabel('Rata-Rata Terpilihnya Lengan')
plt.title('Rata-Rata Terpilihnya Setiap Lengan')
plt.show()

plt.plot(range(num_rounds), avg_avg_rewards)
plt.xlabel("Langkah")
plt.ylabel("Avg. Reward")
plt.title("Adaptive Epsilon-Greedy Avg. Reward")
plt.show()

```

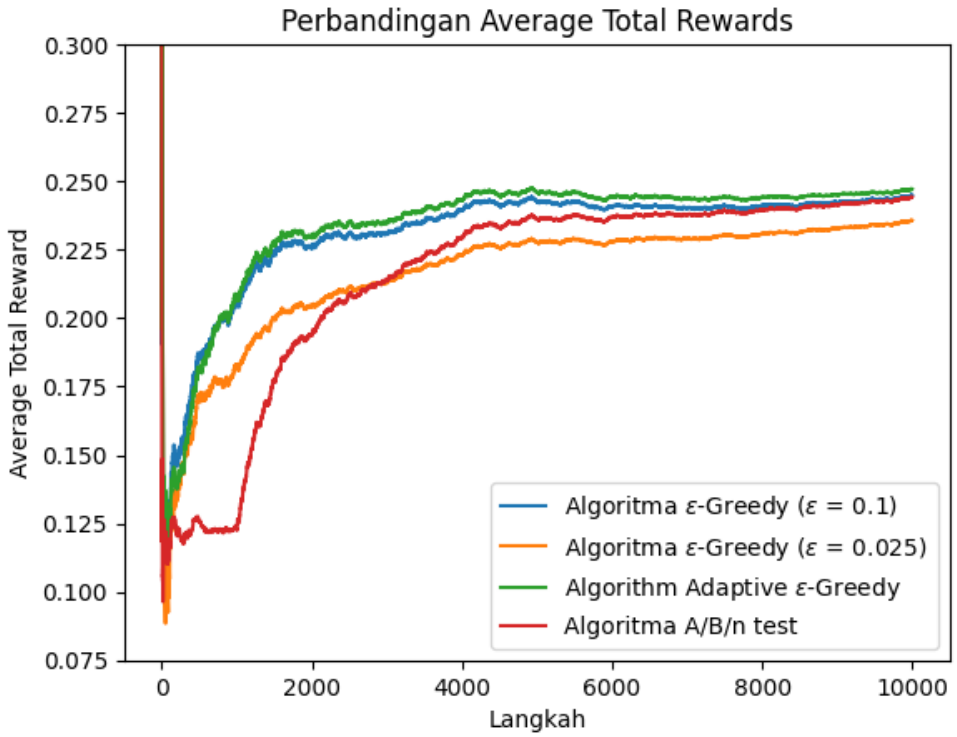
```
Hasil
Average Total CTR: 2468.152
Min CTR =  1176
Max CTR =  2686
```





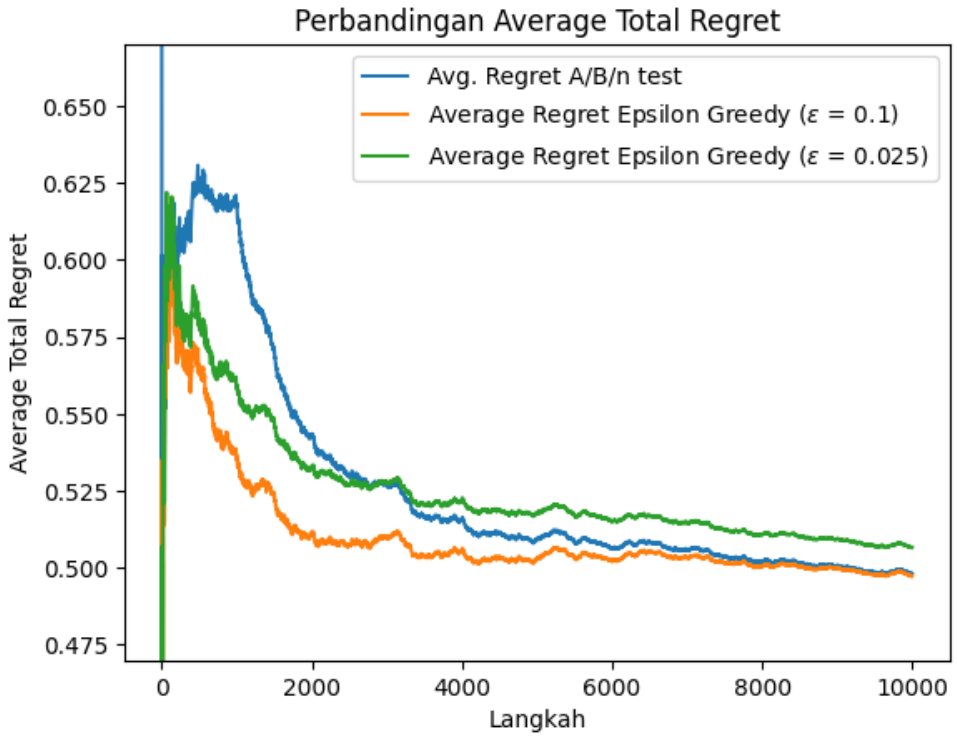
Lampiran 7. Perbandingan nilai Avg. total reward

```
 1. # plt.figure(figsize=(12, 6))
 2. plt.plot(avg_reward_records[0], label= "Algoritma
$\epsilon$-Greedy ($\epsilon$ = 0.1)")
 3. plt.plot(avg_reward_records[3], label= "Algoritma
$\epsilon$-Greedy ($\epsilon$ = 0.025)")
 4. plt.plot(avg_avg_rewards, label='Algorithm Adaptive
$\epsilon$-Greedy')
 5. plt.plot(np.mean(avg_rewards_all, axis=0), label="Algoritma
A/B/n test")
 6. plt.xlabel('Langkah')
 7. plt.ylabel('Average Total Reward')
 8. plt.title('Perbandingan Average Total Rewards')
 9. plt.legend()
10. plt.ylim(0.075, 0.3)
11. plt.show()
```



Lampiran 8. Perbandingan nilai Avg. regret

```
# plt.figure(figsize=(12, 6))
#plt.plot(avg_regrets_mean, label="Avg. Regret")
plt.plot(np.mean(avg_regred_AB_all, axis=0), label="Avg.
Regret A/B/n test")
plt.plot(avg_regrets[0], label= 'Average Regret Epsilon Greedy
($\epsilon$ = 0.1)')
plt.plot(avg_regrets[3], label= 'Average Regret Epsilon Greedy
($\epsilon$ = 0.025)')
plt.xlabel('Langkah')
plt.ylabel('Average Total Regret')
plt.title('Perbandingan Average Total Regret')
plt.legend()
plt.ylim(0.47, 0.67)
plt.show()
```


Perbandingan Average Total Regret
Avg. Regret A/B/n test
Average Regret Epsilon Greedy (ε = 0.1)
Average Regret Epsilon Greedy (ε = 0.025)
Average Total Regret
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0.475
0
2000
4000
6000
8000
10000
Langkah