MUKJIZAT AL-QUR'AN TERHADAP POTENSI PENYEMBUHAN BERDASARKAN PENGAMATAN GELOMBANG DELTA OTAK DENGAN STIMULASI AL-FATIHAH DALAM KONDISI TENANG MENGGUNAKAN ALGORITMA PSD DAN K-NN

SKRIPSI

Diajukan untuk Memenuhi Sebagian Syarat Guna
Memperoleh Gelar Sarjana Strata Satu (S-1)
dalam Ilmu Teknologi Informasi

Diajukan oleh:
THOORIQ NUR ALI
NIM : 1908096004

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI WALISONGO
SEMARANG

2023

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama : Thooriq Nur Ali
NIM : 1908096004
Jurusan : Teknologi Informasi

Menyatakan bahwa skripsi yang berjudul:

MUKJIZAT AL-QUR'AN TERHADAP POTENSI PENYEMBUHAN BERDASARKAN PENGAMATAN GELOMBANG DELTA OTAK DENGAN STIMULASI AL-FATIHAH DALAM KONDISI TENANG MENGGUNAKAN ALGORITMA PSD DAN K-NN

Secara keseluruhan adalah hasil penelitian/karya saya sendiri, kecuali bagian tertentu yang dirujuk sumbernya.

Semarang, 22 September 2023
Pembuat Pertanyaan,

Thooriq Nur Ali
NIM.1908096004

KEMENTRIAN AGAMA REPUBLIK INDONESIA
UNIVERSITAS ISLAM NEGERI WALISONGO
FAKULTAS SAINS DAN TEKNOLOGI
Jl. Prof. Dr. Hamka Ngaliyan Semarang Telp.024-76433366 50185
E-mail: fst.walisongo.ac.id Web:http://fst.walisongo.ac.id

PENGESAHAN

Naskah skripsi berikut ini:

Judul : Mukjizat Al-Qur'an Terhadap Potensi Penyembuhan Berdasarkan Pengamatan Gelombang Delta Otak Dengan Stimulasi Al-Fatihah Dalam Kondisi Tenang Menggunakan Algoritma PSD dan K-NN

Nama : Thooriq Nur Ali

NIM : 1908096004

Jurusan : Teknologi Informasi

Telah diujikan dalam sidang tugas akhir oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam ilmu Teknologi Informasi.

DEWAN PENGUJI

Semarang, 25 September 2023

Penguji I,	Penguji II,
Dr. Masy Ari Ulinuha, M.T	**Mokhamad Iklil Mustofa, M.Kom**
NIP. 198108122011011007	NIP. 198808072019031010
Penguji III,	Penguji IV,
Hery Mustofa, M.Kom	**Azhar Arwani Mahfudh, M.Kom**
NIP. 198703172019031007	NIP. 199107032019031006

KEMENTERIAN AGAMA FAKULTAS SAINS DAN TEKNOLOGI UIN WALISONGO SEMARANG REPUBLIK INDONESIA

Pembimbing I,	Pembimbing II,
Nur Cahyo Hendro Wibowo, S.T., M.Kom	**Mokhamad Iklil Mustofa, M.Kom**
NIP. 197312222006041001	NIP. 198808072019031010

NOTA DINAS PEMBIMBING I

Semarang, 21 September 2023

Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum Wr. Wb.
Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

Judul	:Mukjizat Al-Qur'an terhadap Potensi Penyembuhan Berdasarkan Pengamatan Gelombang Delta Otak dengan Stimulasi Al-Fatihah dalam Kondisi Tenang Menggunakan Algoritma PSD dan K-NN
Nama	: Thooriq Nur Ali
NIM	: 1908096004
Jurusan	: Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diujikan dalam Sidang Munaqosyah.
Wassalamu'alaikum Wr. Wb.

Pembimbing I,

Nur Cahyo H. W., S.T., M.Kom
NIP. 197312222006041001

NOTA DINAS PEMBIMBING II

Semarang, 19 September 2023

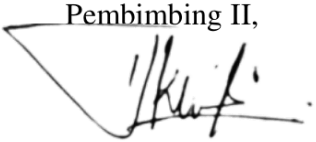
Yth. Ketua Program Studi Teknologi Informasi
Fakultas Sains dan Teknologi
UIN Walisongo Semarang

Assalamu'alaikum Wr. Wb.
Dengan ini diberitahukan bahwa saya telah melakukan bimbingan, arahan dan koreksi naskah skripsi dengan:

Judul	:Mukjizat Al-Qur'an terhadap Potensi Penyembuhan Berdasarkan Pengamatan Gelombang Delta Otak dengan Stimulasi Al-Fatihah dalam Kondisi Tenang Menggunakan Algoritma PSD dan K-NN
Nama	: Thooriq Nur Ali
NIM	: 1908096004
Jurusan	: Teknologi Informasi

Saya memandang bahwa naskah skripsi tersebut sudah dapat diajukan kepada Fakultas Sains dan Teknologi UIN Walisongo untuk diujikan dalam Sidang Munaqosyah.
Wassalamu'alaikum Wr. Wb.

Pembimbing II,



Mokhamad Iklil Mustofa, M.Kom
NIP. 198808072019031010

ABSTRAK

Al-Qur'an merupakan kitab suci dengan kemukjizatan yang dapat dilihat dari berbagai aspek. Salah satu hal yang luar biasa adalah bagaimana al-Qur'an mengisyaratkan ilmu pengetahuan yang berkaitan dengan penyembuhan sebagaimana firman Allah SWT dalam surah al-Isra' ayat 82 dimana secara tersurat menunjukan fungsi al-Qur'an yang dapat digunakan sebagai penawar atau penyembuh. Meski umat Islam percaya dengan sepenuhnya terkait kebenaran dalam al-Qur'an, tidak sedikit orang-orang yang masih meragukannya. Orang-orang yang meragukan kemukjizatan atau keluarbiasaan al-Qur'an beranggapan bahwa kemungkinan keluarbiasaan tersebut merupakan hal yang mustahil dan bertentangan dengan akal pikiran. Padahal yang sebenarnya terjadi adalah keluarbiasaan tersebut hanya susah, tidak, atau belum terjangkau hakikatnya bagaimana keluarbiasaan tersebut dapat diterima oleh akal. Penelitian ini berusaha membuktikan kemukjizatan atau keluarbiasaan al-Qur'an secara ilmiah yang berpotensi sebagai media penyembuhan dengan menganalisa gelombang otak manusia dengan pemberian stimulus surah al-Fatihah ayat 1-7 sebagai salah satu surah dalam al-Qur'an pada 20 partisipan dalam keadaan tenang. Analisa frekuensi gelombang delta dilakukan menggunakan metode PSD menghasilkan data kenaikan gelombang delta atau berpotensi dalam penyembuhan terjadi pada 12 partisipan (60%) sedangkan pada 8 partisipan (40%) tidak terjadi kenaikan gelombang delta. Kemudian dilakukan pembuatan model *machine learning* dengan K-NN dan menghasilkan akurasi sebesar 83%

Kata kunci : Al-Qur'an, Gelombang Otak, PSD, K-NN

KATA PENGANTAR

Alhamdulillahi robbil alamin. Puji syukur atas kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan laporan skripsi dengan judul "Mukjizat Al-Qur'an Terhadap Potensi Penyembuhan Berdasarkan Pengamatan Gelombang Delta Otak Dengan Stimulasi Al-Fatihah Dalam Kondisi Tenang Menggunakan Algoritma PSD dan K-NN". Penulisan laporan skripsi ini ditujukan guna memenuhi salah satu syarat dalam menyelesaikan pendidikan strata satu (S-1) program studi teknologi informasi di Universitas Islam Negeri Walisongo Semarang.

Penulis berharap pula dengan dituliskannya laporan skripsi yang berkaitan dengan pembuktian kemukjizatan al-Qur'an ini dapat memotivasi pembaca maupun para peneliti untuk mengkaji penelitian ilmiah bernafaskan nilai-nilai Islami. Jika di masa lalu Walisongo menyebarkan nilai luhur Islam dengan melakukan akulturasi budaya, maka di zaman modern ini penyebaran nilai luhur Islam dapat dilakukan dengan integrasi ilmu pengetahuan dan teknologi berasaskan kesatuan ilmu pengetahuan (*unity of sciences*).

Pada kesempatan ini penulis menyampaikan rasa terima kasih setulus-tulusnya kepada semua pihak yang telah memberikan bantuan dan dukungan dalam penelitian, penyusunan, hingga penyelesaian penulisan laporan skripsi ini. Penulis menyampaikan rasa terima kasih setulus-tulusnya kepada:

1. Bapak Prof. Dr. Imam Taufiq, M.Ag selaku Rektor Universitas Islam Negeri Walisongo Semarang.
2. Bapak Dr. H. Ismail, M.Ag selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Walisongo Semarang.
3. Bapak Nur Cahyo Hendro Wibowo, S.T., M.Kom selaku ketua program studi Teknologi Informasi Universitas Islam Negeri Walisongo Semarang sekaligus dosen pembimbing I yang telah memberikan bimbingan dalam proses penyusunan laporan skripsi.
4. Bapak Mokhamad Iklil Mustofa, M.Kom selaku dosen pembimbing II yang telah memberikan bimbingan dalam proses penyusunan laporan skripsi.
5. Ibu Heni Sumarti, M,Si selaku dosen mata kuliah instrumentasi kedokteran yang telah memberikan banyak ilmu terkait penelitian terkait kepada penulis.

6. Segenap Bapak dan Ibu dosen di lingkungan Universitas Islam Negeri Walisongo Semarang.
7. Bapak Kasudi, B.Sc dan (Almh.) Ibu Munaroh, S.Pd selaku orang tua penulis yang telah memberikan do'a, rasa cinta, kasih dan sayangnya yang tidak dapat tergantikan.
8. Ke-sembilan kakak yang senantiasa memberikan do'a, dukungan dan motivasi kepada penulis.
9. Teman-teman yang telah memberikan semangat kepada penulis.
10. Serta semua pihak yang tidak dapat penulis sebutkan yang telah terlibat dalam penyusunan laporan skripsi ini.

Penulis merasa sadar bahwa dibalik terselesaikannya laporan skripsi ini, masih terdapat banyak kekurangan dan kelemahan yang ada. Kritik dan saran sangat penulis harapkan dan semoga laporan skripsi ini dapat memberikan manfaat bagi banyak pihak.

Semarang, 22 September 2023

Thooriq Nur Ali

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

A. Latar Belakang Masalah

Al-Qur'an memiliki berbagai aspek yang menjadikan al-Qur'an sebagai suatu mukjizat antara lain dari segi bahasa, isyarat-isyarat ilmu pengetahuan dan teknologi, serta pemberitaan yang gaib (Harsoyo, 2018). Salah satu hal yang luar biasa adalah bagaimana al-Qur'an mengisyaratkan ilmu pengetahuan yang berkaitan dengan penyembuhan. Allah SWT telah menunjukkan keterangan bahwa al-Qur'an dapat dijadikan sebagai penyembuh dari penyakit sebagaimana yang telah di firmankan dalam al-Qur'an surah al-Isra' ayat 82:

وَنُنَزِّلُ مِنَ الْقُرْآنِ مَا هُوَ شِفَاءٌ وَرَحْمَةٌ لِلْمُؤْمِنِينَ ۙ وَلَا يَزِيدُ الظَّالِمِينَ إِلَّا خَسَارًا

Artinya :*"Dan Kami turunkan dari al-Quran suatu yang menjadi penawar (penyembuh) dan rahmat bagi orang-orang yang beriman dan al-Quran itu tidaklah menambah kepada orang-orang yang zalim selain kerugian"*(al-Isra': 82).

Al-Qur’an dipenuhi dengan berbagai petunjuk, undang-undang, dan hukum yang diturunkan sebagai pokok-pokok keterangan dan tidak dapat disangkal kebenarannya (Zubairin, 2020). Meski umat Islam meyakini dengan sepenuhnya kebenaran al-Qur’an, tidak sedikit orang yang meragukan kebenaran dan kemukjizatan al-Qur’an (Harsoyo, 2018). Orang-orang yang meragukan mukjizat atau yang biasa disebut juga sebagai “keluarbiasaan” beranggapan bahwa kemungkinan keluarbiasaan tersebut merupakan suatu hal yang bertentangan dengan akal pikiran sehingga hal tersebut mustahil terjadi. Prof. Muhammad Quraish Shihab berpendapat bahwa sesungguhnya kemungkinan keluarbiasaan itu bukanlah suatu hal yang mustahil dan bertentangan dengan akal pikiran yang sehat, karena yang sebenarnya terjadi dalam keluarbiasaan tersebut adalah bahwa keluarbiasaan hanya susah, tidak, atau belum terjangkau hakikatnya bagaimana peristiwa keluarbiasaan tersebut dapat diterima oleh akal (Shihab, 2013).

Otak menjadi salah satu organ yang dapat menghasilkan aktivitas kelistrikan dan dapat direkam dengan menggunakan elektroensefalografi (EEG).

Perekaman menggunakan elektroensefalografi menghasilkan data gelombang otak yang dapat diolah dan diekstraksi sehingga dapat diketahui aktifitas dan fungsi fisiologis yang terjadi dalam otak manusia. Data perekaman yang diperoleh dari instrumen elektroensefalografi (EEG) merupakan data mentah berbasis waktu (*time based*) sehingga perlu dilakukan pengolahan data lanjutan untuk dapat diekstraksi ke dalam data berbasis frekuensi (*frequency based*).

Algoritma *power spectral density* (PSD) dengan metode *welch* merupakan salah satu algoritma yang dapat digunakan untuk melakukan ekstraksi data ke dalam bentuk frekuensi. Metode *welch* digunakan untuk mengestimasi spektrum daya dari urutan waktu yang ditentukan (Ameera dkk., 2019). Umumnya gelombang otak dibedakan menjadi 5 jenis gelombang berdasarkan frekuensi yaitu *delta*, *theta*, *alpha*, *beta*, dan *gamma* yang masing-masing gelombang mengindikasikan kondisi tertentu yang tengah dialami manusia. Gelombang yang berperan dalam proses penyembuhan adalah gelombang *delta* dengan rentang frekuensi 0,5 – 4,0 Hz (Saminan, 2020). Pancaran gelombang *delta* mengindikasikan tubuh dan pikiran manusia tengah

dalam keadaan beristirahat. Tubuh manusia mengalami proses pemulihan terhadap jaringan yang mengalami kerusakan dan melakukan regenerasi sel (Saminan, 2020). Pada penelitian ini akan dilakukan perekaman gelombang otak yang didapatkan dengan instrumen elektroensefalografi (EEG) dengan stimulus salah satu surah dalam al-Qur'an yaitu surah al-Fatihah dalam kondisi tenang untuk membuktikan kebenaran terkait potensi penyembuhan yang terkandung dalam al-Qur'an. Data hasil perekaman berbasis frekuensi diperoleh setelah ekstraksi gelombang dengan menggunakan algoritma *power spectral density* (PSD). Perbandingan kenaikan dan penurunan aktifitas gelombang *delta* otak terhadap hasil perekaman sebelum dan setelah pemberian stimulus dilakukan untuk mengetahui terjadinya potensi penyembuhan. Pelabelan dilakukan terhadap data dengan dua tingkat berdasarkan pengamatan kenaikan gelombang delta otak yaitu "terjadi kenaikan" dan "tidak terjadi kenaikan" untuk selanjutnya digunakan pada pembuatan model *machine learning* dengan algoritma *k-nearest neighbor* (K-NN).

B. Identifikasi Masalah

Permasalahan yang dapat diidentifikasi berdasarkan latar belakang masalah yang telah dipaparkan adalah sebagai berikut:

1. Kebenaran kemukjizatan al-Qur'an telah diyakini sepenuhnya oleh umat Islam, namun sayangnya masih terdapat orang-orang yang meragukan kebenaran dan kemukjizatan al-Qur'an.
2. Perlu adanya upaya untuk menganalisa potensi penyembuhan untuk membuktikan secara ilmiah atas kebenaran potensi penyembuhan dalam al-Qur'an.
3. Hasil data perekaman elektroensefalografi (EEG) masih berupa data berbasis waktu (*time based*).
4. Perlu dilakukan upaya transformasi data berbasis waktu (*time based*) menjadi data berbasis frekuensi (*frequency based*) yang nantinya digunakan untuk menganalisa potensi penyembuhan serta membuat model klasifikasi.

C. Rumusan Masalah

Rumusan masalah yang diambil dalam penelitian ini adalah bagaimana membuktikan kebenaran mukjizat al-Qur'an terhadap potensi penyembuhan berdasarkan pengamatan gelombang delta otak dengan stimulasi

surah al-Fatihah dalam kondisi tenang menggunakan algoritma *power spectral density (*PSD*)* dan *k-nearest neighbor* (K-NN)?

D. Tujuan Penelitian

Tujuan yang diambil dalam penelitian ini adalah membuktikan kebenaran mukjizat al-Qur'an terhadap potensi penyembuhan berdasarkan pengamatan gelombang otak dengan stimulasi surah al-Fatihah dalam kondisi tenang menggunakan algoritma *power spectral density* (PSD) dan *k-nearest neighbor* (K-NN)

E. Batasan Masalah

Batasan penelitian yang akan dilakukan dalam penelitian ini antara lain:

1. Stimulus atau rangsangan yang digunakan dalam penelitian berupa audio *murottal* surah al-Fatihah ayat 1 – 7.
2. Akuisisi data dilakukan pada sampel mahasiswa partisipan dalam kondisi tenang dengan mata tertutup.
3. Data mentah (*raw data*) berupa data gelombang berbasis waktu (*time based*) didapatkan dengan instrumen elektroensefalografi (EEG).

4. Ekstraksi gelombang dari data mentah (*raw data*) berbasis waktu (*time based*) ke data berbasis frekuensi (*frequency based*) diolah menggunakan algoritma *power spectral density* (PSD).
5. Gelombang otak yang dianalisis diekstraksi menjadi 5 frekuensi gelombang yaitu gelombang *delta*, *theta*, *alpha*, *beta*, dan *gamma*.
6. Pelabelan dilakukan terhadap data dengan 2 tingkat berdasarkan pengamatan kenaikan gelombang delta otak yaitu "terjadi kenaikan" dan "tidak terjadi kenaikan".
7. Model klasifikasi dibuat dengan menggunakan salah satu algoritma klasifikasi dalam *machine learning* yaitu algoritma *k-nearest neighbor* (K-NN).
8. Penilaian akurasi model klasifikasi menggunakan metode *confusion matrix*.

F. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

1. Manfaat Akademis

   Penelitian ini diharapkan dapat bermanfaat sebagai salah satu referensi maupun rujukan, serta kontribusi

ilmu pengetahuan dalam membuktikan sebuah teori maupun dalil dalam al-Qur'an.

2. Manfaat Praktis

   a. Bagi Umat Islam

      Pembuktian terkait kemukjizatan al-Qur'an diharapkan dapat semakin memperkuat kepercayaan atau keimanan umat Islam terhadap agama Islam serta kebenaran al-Qur'an yang merupakan mukjizat terbesar yang difirmankan langsung oleh Allah SWT.

   b. Bagi Masyarakat

      Menambah pengetahuan dan wawasan masyarakat terkait pembuktian salah satu kemukjizatan al-Qur'an secara ilmiah serta memberikan informasi bahwa al-Qur'an dapat bermanfaat untuk penyembuhan.

   c. Bagi Institusi

      Bentuk kontribusi dalam ilmu pengetahuan yang berkaitan dengan konsep kesatuan ilmu pengetahuan (*unity of sciences*) untuk kemanusiaan dan peradaban, serta memberikan tambahan referensi ataupun rujukan untuk

melakukan penelitian lain yang terkait dengan topik penelitian serupa.

d. Bagi Tenaga Kesehatan

Memberikan informasi metode alternatif yang dapat digunakan sebagai salah satu metode penyembuhan yang dibuktikan dengan hasil pengolahan data otak dengan menggunakan instrumen kesehatan elektroensefalografi (EEG) pada seseorang dengan stimulus bacaan al-Qur'an.

e. Bagi Peneliti

Sebagai bentuk pengabdian secara langsung terhadap agama dan ilmu pengetahuan. Manfaat lain yang didapatkan antara lain memberikan informasi, wawasan, pengetahuan, serta pengalaman dalam membuktikan kebenaran dan kemukjizatan al-Qur'an sebagai penyembuh dengan melakukan eksperimen dan pengujian terhadap data gelombang otak.

BAB II
LANDASAN PUSTAKA

A. Kajian Pustaka

1. Mukjizat

Mukjizat secara etimologi diambil dari kata dalam bahasa Arab *a'jaza* (أعجز) yang memiliki arti "melemahkan atau menjadikan tidak mampu", pelaku yang bertindak melemahkan disebut dengan *mu'jiz* (معجز), dan jika kemampuan dalam melemahkan sangat mencolok dan berpengaruh membungkam lawan disebut dengan *mu'jizat* (معجزة) (Shihab, 2013). Sedangkan pengertian mukjizat secara terminologi menurut pakar agama Islam didefinisikan sebagai "Peristiwa luar biasa yang terjadi melalui nabi sebagai bukti kenabiannya, yang ditantangkan kepada orang yang meragukannya untuk melakukan hal serupa namun orang yang meragukan tidak mampu melakukan tantangan itu" (Shihab, 2013).

Mukjizat maupun peristiwa keluarbiasaan menurut sebagian orang merupakan suatu hal yang tidak dapat diterima dengan akal pikiran sehingga orang-orang tersebut menganggap mukjizat maupun peristiwa keluarbiasaan adalah suatu hal yang

mustahil terjadi. Prof. Muhammad Quraish Shihab dalam bukunya yang berjudul Mukjizat Al-Qur'an berpendapat bahwa sesungguhnya kemungkinan keluarbiasaan itu bukanlah suatu hal yang mustahil dan bertentangan dengan akal pikiran yang sehat. Karena yang sebenarnya terjadi dalam keluarbiasaan adalah bahwa keluarbiasaan tersebut hanya susah, tidak, atau belum terjangkau hakikatnya bagaimana peristiwa keluarbiasaan tersebut dapat diterima oleh akal (Shihab, 2013).

Mukjizat dapat dibagi menjadi dua bagian yaitu mukjizat material dan mukjizat rasional (Wardhana, 2016).

a. Mukjizat Material

   Mukjizat material dapat dipahami sebagai mukjizat yang dapat diterima oleh pancaindra namun tidak dapat dipahami secara langsung oleh akal pikiran manusia karena bertentangan dengan hukum alam. Mukjizat material umumnya terjadi pada zaman Nabi.

b. Mujizat Rasional

   Mukjizat rasional dapat diartikan sebagai mukjizat yang dapat dipahami dengan akal

pikiran manusia dan tergantung pada tingkat pemahaman dan ilmu pengetahuan yang dimiliki manusia.

2. Al-Qur’an sebagai Penyembuh

Al-Qur’an memiliki banyak nama lain yang secara tersirat maupun tersurat dapat mengindikasikan fungsi dari al-Qur’an. Nama lain yang mengindikasikan fungsi al-Qur’an secara tersurat dapat dibagi menjadi 4 (Al-Khalidi, 1994). Diantara fungsi al-Qur’an yang tersurat dalam nama-nama lain al-Qur’an antara lain:

a. *Al-Huda* yang berarti petunjuk
b. *Al-Furqan* yang berarti pemisah (benar dan salah)
c. *As-Syifa’* yang berarti penyembuh
d. *Al-Mau’idzoh* yang berarti nasehat

As-Syifa’ merupakan salah satu nama lain al-Qur’an yang berarti penyembuh. Hakikat penyembuhan ini didasarkan atas tiga ayat yang secara tersurat menyebutkan al-Qur’an sebagai penyembuh (Hasballah, 2013). Tiga ayat yang menyebutkan al-Qur’an sebagai penyembuh antara lain:

a. Surah Yunus ayat 57

يَا أَيُّهَا النَّاسُ قَدْ جَاءَتْكُمْ مَوْعِظَةٌ مِنْ رَبِّكُمْ وَشِفَاءٌ لِمَا فِي الصُّدُورِ وَهُدًى وَرَحْمَةٌ لِلْمُؤْمِنِينَ

Artinya :*"Hai manusia, sesungguhnya telah datang kepadamu pelajaran dari Tuhanmu dan penyembuh bagi penyakit-penyakit (yang berada) dalam dada dan petunjuk serta rahmat bagi orang-orang yang beriman"* (Yunus: 57).

b. Surah al-Isra' ayat 82

وَنُنَزِّلُ مِنَ الْقُرْآنِ مَا هُوَ شِفَاءٌ وَرَحْمَةٌ لِلْمُؤْمِنِينَ ۙ وَلَا يَزِيدُ الظَّالِمِينَ إِلَّا خَسَارًا

Artinya :*"Dan Kami turunkan dari al-Quran suatu yang menjadi penawar (penyembuh) dan rahmat bagi orang-orang yang beriman dan al-Quran itu tidaklah menambah kepada orang-orang yang zalim selain kerugian"* (al-Isra': 82).

c. Surah Fussilat ayat 44

...قُلْ هُوَ لِلَّذِينَ آمَنُوا هُدًى وَشِفَاءٌ...

Artinya :*"...al-Quran itu adalah petunjuk dan penawar bagi orang-orang mukmin..."* (Fussilat : 44).

3. Al-Fatihah sebagai Penyembuh

Surah al-Fatihah merupakan surah dengan keutamaan yang luar biasa didalamnya, keutamaan-keutamaan diantaranya adalah semua makna atau intisari al-Qur'an terkandung di dalam surah al-Fatihah sehingga surah al-Fatihah dikenal sebagai *Ummul Qur'an* atau induk al-Qur'an (Sari & Asiva, 2019). Mempelajari kandungan surah al-Fatihah yang disebut memiliki semua makna intisari al-Qur'an berarti juga mempelajari seluruh kandungan al-Qur'an (Arkoun, 1998).

Surah al-Fatihah selain disebut sebagai *Ummul Qur'an,* al-Fatihah juga disebut pula sebagai *asy-Syifa'* yang berarti sebagai penyembuh (Sari & Asiva, 2019). Surah al-Fatihah dipercaya oleh umat Islam sebagai surah yang dapat menyembuhkan berbagai penyakit seperti nama lain yang disandangnya yaitu *asy-Syifa'*. Penyembuhan dalam surah al-Fatihah ini didasarkan atas salah satu hadits Rasulullah SAW yang diriwayatkan oleh ad-Darimi:

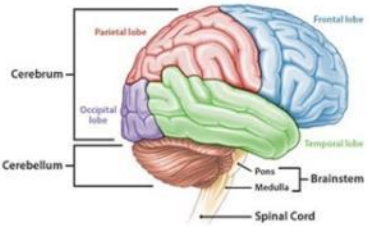
أَخْبَرَنَا قَبِيصَةُ أَخْبَرَنَا سُفْيَانُ عَنْ عَبْدِ الْمَلِكِ بْنِ عُمَيْرٍ قَالَ قَالَ رَسُولُ اللَّهِ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ فِي فَاتِحَةِ الْكِتَابِ شِفَاءٌ مِنْ كُلِّ دَاءٍ

Artinya :"*Telah mengabarkan kepada kami (Qabishah) telah mengabarkan kepada kami (Sufyan) dari (Abdul Malik bin Umair) ia berkata; Rasulullah shallallahu 'alaihi wasallam bersabda: 'Surat Al Fatihah adalah penawar dari segala penyakit'* " (ad-Darimi 3236).

4. Anatomi Otak

Otak manusia merupakan jaringan lunak yang memiliki massa sekitar 0,5 kg dan berisi sekitar 100 miliar sel neuron dengan fungsi yang sangat kompleks sebagai pengatur dan pengendali seluruh aktivitas manusia, mulai dari pengendali aktivitas indera, bahasa, analisa, motorik tubuh, pusat emosi, dan pengambilan keputusan (Moore dkk., 2013).

Otak dapat dibagi menjadi tiga bagian utama yaitu *cerebrum* (otak besar), *cerebellum* (otak kecil), dan *brainstem* (batang otak) (Saminan, 2020; Syarif, 2018).



Gambar 2.1 Anatomi Otak Tampak Lateral
(Syarif, 2018)

a. Cerebrum

Cerebrum memiliki volume lebih dari tiga perempat volume total otak (Rozi, 2019). *Cerebrum* atau otak besar terdiri dari dua bagian *hemisfer* yakni *hemisfer* kiri, dan *hemisfer* kanan. Masing-masing *hemisfer* terdiri dari 4 lobus yaitu lobus frontal (F), lobus parietal (P), lobus oksipital (O), dan lobus temporal (T) (Sherwood, 2014).

Tabel 2.1 Bagian-bagian *Cerebrum*

Bagian	Keterangan
Lobus Frontal (F)	Penalaran, perencanaan, pembuat keputusan, gerakan, kepribadian dan pemecahan masalah. (*primary motor area*)
Lobus Parietal (P)	Orientasi serta penerimaan rangsangan suhu, sentuhan, tekanan, dan getaran. (*primary somatosensory area*)
Lobus Oksipital (O)	Sebagai pusat visual (*primary visual korteks*)
Lobus Temporal (T)	Penerimaan informasi berkaitan dengan pendengaran. (*primary auditory korteks*)

(Rozi, 2019)

b. Cerebellum

Cerebellum berfungsi mengontrol banyak fungsi otomatis otak diantaranya sikap dan posisi tubuh, keseimbangan, hingga koordinasi otot dan gerakan tubuh (Syarif, 2018).

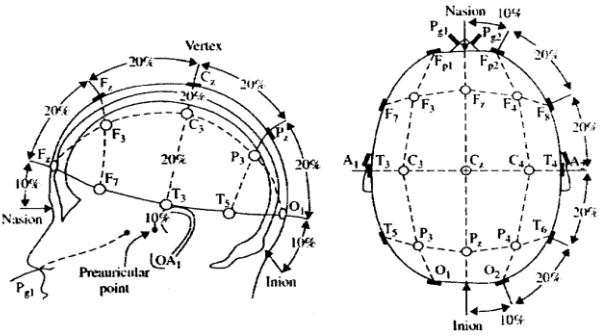
c. Brainstem

Batang otak memiliki fungsi sebagai pengontrol, denyut jantung, kesadaran, tekanan darah, pernafasan, hingga pola makan dan tidur (Syarif, 2018).

5. Elektroensefalografi (EEG)

Elektroensefalografi (EEG) adalah instrumen yang digunakan untuk pemantauan elektrofisiologi dalam merekam aktivitas kelistrikan pada otak (Hindarto & Muntasa, 2018). Perekaman aktivitas kelistikan otak umumnya menggunakan kaidah internasional sistem 10-20 dengan frekuensi *sampling* 128 – 1024 Hz. Kaidah sistem 10-20 ini didasarkan pada rekomendasi komite federasi elektroensefalografi dan neurofisiologi klinik internasional (IFSECN) (Bintoro, 2012). Sistem 10-20 didasarkan dengan anatomi otak manusia. Peletakan elektroda pada kulit kepala (*scalp*) ditandai dengan beberapa huruf yaitu C (*central*), F (*frontal*), O (*occipital*), P (*parietal*), dan T (*temporal*) serta penandaan angka dengan angka

genap untuk sisi kanan dan angka ganjil untuk sisi kiri.



Gambar 2.2 Sistem Internasional 10-20 EEG
(Yulianto dkk., 2013)

Hasil perekaman elektroensefalografi (EEG) berupa hubungan antara tegangan terhadap waktu secara tegak lurus dengan sumbu vertikal (Y) sebagai tegangan dan sumbu (X) sebagai waktu (Deu, 2019). Gelombang otak yang dihasilkan oleh instrumen elektroensefalografi (EEG) merepresentasikan besaran yang biasa dihasilkan dalam melakukan analisa terhadap frekuensi yaitu amplitude dan waktu (Anggara & Rahayu, 2020; Dewi & Djamal, 2015).

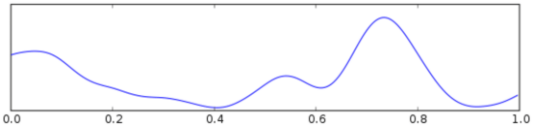
6. Klasifikasi Gelombang Otak

Gelombang otak pada manusia pada umumnya dapat diklasifikasikan berdasarkan rentang frekuensi

yang terbagi kedalam beberapa jenis gelombang, yaitu:

a. Gelombang *Delta* (δ)

Gelombang *delta* adalah gelombang yang terpancar pada otak dalam rentang frekuensi 0,5 – 4 Hz dan amplitude 100 – 200 μV (Saminan, 2020). Gelombang *delta* terpancar ketika manusia berada pada kondisi tertidur pulas tanpa mimpi. Penyembuhan dan regenerasi terstimulus saat dalam gelombang delta (Garg & Keswani, 2017). Tubuh manusia mengalami proses pemulihan melakukan regenerasi sel (Filcek, 2023; Saminan, 2020).

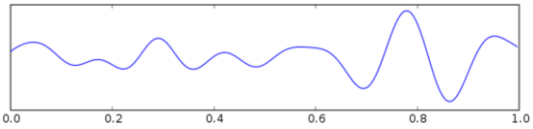


Gambar 2.3 Gelombang *Delta*
(Bermudez dkk., 2020)

b. Gelombang *Theta* (θ)

Gelombang *theta* adalah gelombang yang terpancar pada otak dalam rentang frekuensi 4 – 8 Hz dan amplitude 5 – 10 μV (Saminan, 2020). Gelombang *theta* meningkat ketika seseorang

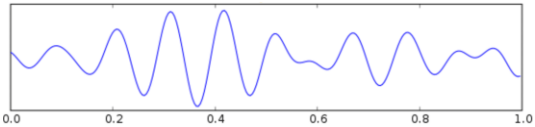
tengah dalam keadaan bersantai atau terbangun dari tidur (Sofiani, 2022).



Gambar 2.4 Gelombang *Theta*
(Bermudez dkk., 2020)

c. Gelombang *Alpha* (α)

Gelombang *alpha* adalah gelombang yang terpancar pada otak dalam rentang frekuensi 8 – 12 Hz dan amplitude 20 – 80 μV (Saminan, 2020). Gelombang *alpha* mengindikasikan manusia tengah dalam keadaan rileks dan tenang. Kondisi nafas dan nadi juga akan lebih stabil pada gelombang alfa (Rozi, 2019).

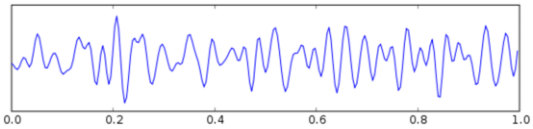


Gambar 2.5 Gelombang *Alpha*
(Bermudez dkk., 2020)

d. Gelombang *Beta* (β)

Gelombang *beta* adalah gelombang yang terpancar pada otak dalam rentang frekuensi 12 – 25 Hz dan amplitude 1 – 5 μV (Saminan, 2020). Gelombang *beta* memiliki keterlibatan dalam

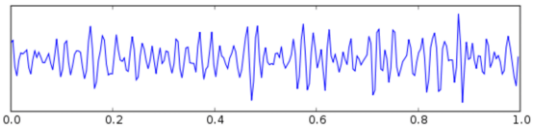
proses berpikir kognitif dan motorik. Gelombang beta terpancar ketika manusia tengah dalam keadaan terkonsentrasi yang dalam, atau sedang dalam fase memecahkan masalah (Deu, 2019).



Gambar 2.6 Gelombang *Beta*
(Bermudez dkk., 2020)

e. Gelombang *Gamma* (γ)

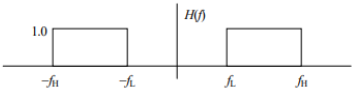
Gelombang *gamma* adalah gelombang yang terpancar pada otak dalam rentang frekuensi 25 - 40 Hz dan amplitude 0,5 – 2 μV (Saminan, 2020). Gelombang *gamma* terbentuk ketika manusia tengah dalam keadaan dengan aktivitas mental yang sangat tinggi seperti panik, ketakutan, tampil di muka umum dan sebagainya (Syarif, 2018).



Gambar 2.7 Gelombang *Gamma*
(Bermudez dkk., 2020)

7. *Bandpass Filter* (BPF) dan *Filter Finite Impulse Response* (FIR)

Bandpass Filter (BPF) atau yang dapat disebut juga dengan filter penerus pita frekuensi adalah filter yang hanya meneruskan sinyal-sinyal yang berkisar antara *cut-off* frekuensi rendah fL (*low cut off frequency*) dan *cut-off* frekuensi tinggi fH (*high cut off frequency*). Secara fisis, fungsi pindah filter ini akan sama dengan satu (1) untuk frekuensi diantara fL dan fH dan akan sama dengan nol (0) untuk frekuensi di luar itu. Secara matematis fungsi pindah ini juga akan sama dengan satu (1) untuk frekuensi diantara $-fL$ dan $-fH$ dan akan sama dengan nol (0) untuk frekuensi di luar frekuensi antara $-fL$ dan $-fH$ (Wati, 2018). Fungsi pindah filter lolos pada *bandpass filter* ini dapat dilihat pada gambar 2.8.



Gambar 2.8 *Bandpass Filter*
(Wati, 2018)

Filter *Finite Impulse Response* (FIR) adalah salah satu metode filter yang umumnya digunakan pada pemrosesan sinyal digital. Filter *Finite Impulse Response* (FIR) merupakan filter yang mempunyai

respon impuls terbatas karena tidak ada *feedback* di dalam filter, jika dimasukkan sebuah impuls dengan sebuah sinyal ‘1’ diikuti dengan banyak sinyal ‘0’, sinyal nol akan keluar ketika sinyal 1 telah melewati semua *delay line* dengan koefisiennya. Filter *Finite Impulse Response* (FIR) memiliki kelebihan dengan sistem yang stabil dan non rekursif, serta *output* yang dihasilkan tidak terpengaruh oleh output sebelumnya. Proses pendesainan filter *Finite Impulse Response* (FIR) melingkupi pengenalan koefisien yang sesuai dengan respon frekuensi. Koefisien menentukan respon dari filter. Penghitungan koefisien filter *Finite Impulse Response* (FIR) didapatkan dengan metode *window* (Dimurtadha dkk., 2019). *Finite Impulse Response* (FIR) dapat dinyatakan dengan persamaan 2.1 (Dimurtadha dkk., 2019).

$$y[k] = \sum_{n=0}^{M-1} h[n]x[k-n] \qquad (2.1)$$

Dimana :

M = panjang filter digital

$h[n]$ = respon impuls filter/koefisien filter

$x[k]$ = sampel sinyal *input*

$x[k-n]$ = sampel sinyal *input* yang ditahan

$y[k]$ = *output* sinyal digital

8. *Independent Component Analysis* (ICA)

Independent Component Analysis (ICA) adalah metode atau teknik perhitungan data statistik dengan tujuan menemukan faktor-faktor tersembunyi pada sekumpulan variabel acak atau variabel multivariat (Riwinoto, 2014). Metode *Independent Component Analysis* (ICA) digunakan untuk memisahkan sinyal-sinyal tercampur yang berasal dari berbagai sumber yang saling bebas statistik satu sama lain, persebaran atau distribusi dari sinyal-sinyal bebas statistik tersebut bersifat *non-gaussian*, dengan kata lain metode *Independent Component Analysis* (ICA) dapat digunakan sebagai metode yang memisahkan sinyal-sinyal luar yang tercampur. *Independent Component Analysis* (ICA) didasarkan dari mekanisme proses penambahan sinyal yang disebut dengan *instantaneous mixture*. Persamaan *instantaneous mixture* dapat dinyatakan dengan persamaan 2.2 (Firstanto dkk., 2013).

$$X(t) = A.S(t) \qquad (2.2)$$

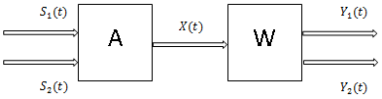
Dimana :

X = nilai sinyal tercampur

A = nilai *mixing matrix*

S = sinyal sumber

Proses *instantaneous mixture* dapat digambarkan dengan ilustrasi pada gambar 2.9.



Gambar 2.9 Proses *Instantaneous Mixture*
(Firstanto dkk., 2013)

Proses untuk mengetahui atau mendapatkan nilai *mixing matrix* (A) adalah dengan pengaturan letak sumber sinyal dengan penerima sinyal. Variabel yang dapat membentuk *mixing matrix* dapat dideskripsikan sebagai berikut:

Sinyal 1 ↔ Penerima Sinyal 1 = A_{11}

Sinyal 1 ↔ Penerima Sinyal 2 = A_{12}

Sinyal 2 ↔ Penerima Sinyal 1 = A_{21}

Sinyal 2 ↔ Penerima Sinyal 2 = A_{22}

Jika ditulis dengan persamaan matriks akan menjadi:

$$\begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{bmatrix} \qquad (2.3)$$

Persamaan proses pencampuran dapat diidentifikasi dari gambar 2.9 yang dapat dinyatakan dengan persamaan 2.4 (Firstanto dkk., 2013).

$$x_1(t) = a_{11}\, s_1(t) + a_{12}\, s_2(t)$$
$$x_2(t) = a_{21}\, s_1(t) + a_{22}\, s_2(t) \qquad (2.4)$$

Nilai W pada gambar 2.9 merupakan nilai *unmixing matrix.* Nilai *unmixing matrix* dapat diperoleh dengan menghitung nilai invers dari *mixing matrix* atau dapat pula dengan menggunakan algoritma yang ada pada metode *Independent Component Analysis* (ICA). Salah satu algoritma dalam *Independent Component Analysis* (ICA) yang dapat digunakan adalah algoritma *FastICA.* Algoritma *FastICA* adalah algoritma sederhana yang dapat digunakan dalam pemisahan sinyal suara (Firstanto dkk., 2013). Metode yang digunakan dalam algoritma *FastICA* menentukan komponen *independent* sehingga akan didapatkan nilai W atau *unmixing matrix* disebut dengan metode *symmetrical*. Nilai W berfungsi memisahkan sinyal campuran yang sudah melalui proses *whitening* menjadi sinyal estimasi (Y). Proses pemisahan sinyal suara dinyatakan dengan persamaan 2.5 (Firstanto dkk., 2013).

$$Y(t) = W.X_n(t) \quad (2.5)$$

Dimana :

Y = sinyal estimasi

W = nilai *unmixing matrix*

X = nilai *mixing matrix*

Tahapan dalam metode *symmetrical* adalah sebagai berikut (Firstanto dkk., 2013):

a. Menentukan variabel m atau jumlah *independent component.*
b. Menentukan nilai awal vektor kompleks w pada persamaan 2.5 dengan y merupakan hasil kali dari w dengan x_n.
c. Menghitung nilai g (ketidak linearitas) dan g' menggunakan persaman 2.6 dan persamaan 2.7.

$$g(y) = \exp\left(\frac{-y^2}{2}\right) \quad (2.6)$$

$$g'(y) = (1-y^2)\exp\left(\frac{-y^2}{2}\right) \quad (2.7)$$

d. Mendapatkan nilai w baru berdasarkan persamaan 2.8.

$$w \leftarrow E\{zg(w^T z)\} - E\{g'(w^T z)\}w \quad (2.8)$$

e. Melakukan iterasi berdasarkan persamaan 2.9 dan 2.10.

$$W \leftarrow \frac{W}{\|W\|} \quad (2.9)$$

$$W \leftarrow \frac{3}{2}W - \frac{1}{2}WW^TW \quad (2.10)$$

f. Nilai W atau *unmixing matrix* berhasil didapatkan.

9. *Power Spectral Density* (PSD) Metode *Welch*

Algoritma *power spectral density* (PSD) dengan metode Welch merupakan salah satu algoritma yang dapat digunakan untuk melakukan ekstraksi data ke dalam bentuk frekuensi. Metode Welch digunakan untuk mengestimasi spektrum daya dari urutan waktu yang ditentukan (Ameera dkk., 2019). *Power Spectral Density* (PSD) dengan metode *Welch* pertama kali diperkenalkan oleh penemunya yang bernama Welch pada tahun 1967. Isyarat masukan dibagi menjadi segmen-segmen pendek serta perhitungan periodogramnya didapatkan berdasarkan perhitungan *Fast Fourier Transform* (FFT), dengan demikian didapatkan perhitungan yang lebih efisien untuk mencari estimasi spektrum daya. Setiap segmen data dilakukan perkalian pada suatu fungsi jendela (*window*) sebelum dilakukan perhitungan periodogramnya. Selanjutnya periodogram yang

telah dimodifikasi melalui proses perkalian dirata-ratakan sehingga menghasilkan spektrum frekuensi yang lebih baik (Antonisfia & Wiryadinata, 2008). Algoritma pada *Power Spectral Density* (PSD) dengan metode *Welch* adalah sebagai berikut (Husain & Aji, 2019):

a. Sinyal masukan (*input*) $x[n]$ dihitung berdasarkan persamaan 2.11.

$$x_l(n) = x(n + (1 + l - 1)M) \quad (2.11)$$

Dimana, $n = 0, \dots, N$ dan $l = 1, \dots, L$. Algoritma PSD dengan metode Welch merekomendasikan nilai $M = \frac{N}{2}$.

b. Tahapan dilanjutkan dengan pencarian *window* pada masing-masing segmen dengan menggunakan persamaan 2.12.

$$A_l(k) = \sum_{n=0}^{N-1} x_l(n) w(n) e^{-j\frac{2\pi}{n}nk} \quad (2.12)$$

Dimana A_l merupakan FFT *windowed* segmen.

c. Setelah *windowed* segmen telah didapatkan, tahapan selanjutnya adalan perhitungan dengan menggunakan persamaan 2.13.

$$\emptyset_l(k) = \frac{1}{NP} |A_l(k)|^2 \quad (2.13)$$

Dimana $\emptyset_l$ merupakan periodogram

d. *Power window P* dihitung berdasarkan persamaan 2.14.

$$P = \sum_{n=0}^{N-1} |w(n)|^2 \quad (2.14)$$

e. Kemudian akan diestimasi menggunakan metode Welch dengan melihat rata-rata dari periodogramnya yang dihitung berdasarkan persamaan 2.15.

$$S(k) = \frac{1}{L} \sum_{l=1}^{L} \emptyset_l(k) \quad (2.15)$$

10. Machine Learning

Machine Learning atau yang dapat disebut juga sebagai pembelajaran mesin adalah salah satu cabang dari ilmu kecerdasan buatan (*artificial intelligence*) yang mempelajari bagaimana komputer dapat meningkatkan kecerdasannya dengan belajar dari data (Wahyono, 2018). *Machine learning* memungkinkan sistem untuk terus belajar dan meningkatkan kemampuan berdasarkan pengalaman (Sarno dkk., 2022).

Machine learning dapat dibagi menjadi empat tipe berdasarkan teknik pembelajarannya yaitu *supervised learning*, *unsupervised learning*, *semi supervised learning*, dan *reinforcement learning* (Wahyono, 2018).

a. *Supervised Learning*

Supervised learning merupakan salah satu paradigma dalam *machine learning* yang mempelajari data berdasarkan data latih yang telah memiliki label (*labeled data*). *Supervised learning* cocok untuk menyelesaikan persoalan yang berhubungan dengan regresi dan klasifikasi (Sarno dkk., 2022). Beberapa algoritma yang termasuk ke dalam *supervised learning* antara lain:

1) *Linear Regression* (LR)
2) *Logistic Regression*
3) *Linear Discriminant Analysis* (LDA)
4) *Quadratic Discriminant Analysis* (QDA)
5) *Decision Tree* (DT)
6) *Random Forest*
7) *Naïve Bayes*
8) *K-Nearest Neighbor* (K-NN)
9) *Support Vector Machine* (SVM)

b. *Unsupervised Learning*

Paradigma *unsupervised learning* mempelajari data tidak berlabel (*unlabeled data*) berdasarkan fitur-fitur dari data tersebut. *Unsupervised*

learning cocok untuk menyelesaikan persoalan yang berhubungan dengan asosiasi dan klasterisasi (Sarno dkk., 2022). Beberapa algoritma yang termasuk ke dalam *unsupervised learning* antara lain:

1) *K-Means*
2) *Gaussian Mixture Clustering*
3) *Mean Shift Clustering*
4) *Agglomerative Hierarchical Clustering*
5) DBSCAN
6) BIRCH
7) OPTICS

c. *Semi Supervised Learning*

Paradigma *semi supervised learning* merupakan kombinasi dari *supervised learning* dan *unsupervised learning* dengan mempelajari data yang berlabel (*labeled data*) dan data tidak berlabel (*unlabeled data*) secara bersamaan untuk dijadikan data latih (*data training*).

d. *Reinforcement Learning*

Reinforcement learning merupakan metode dalam *machine learning* yang belajar menyesuaikan dengan kondisi lingkungannya (*environment*)

(Sarno dkk., 2022). Agen dalam *reinforcement learning* melakukan pembelajaran berdasarkan percobaan *trial and error* dengan mendapatkan hasil atau *reward* terbaik. Beberapa algoritma yang termasuk ke dalam *reinforcement learning* antara lain:

1) *State Action Reward State Action* (SARSA)
2) *Q-Learning*
3) *Deep Q Learning*

11. *K-Nearest Neighbor* (K-NN)

Algoritma *k-nearest neighbor* (K-NN) adalah salah satu algoritma yang termasuk dalam paradigma *supervised learning* yang berarti bahwa algoritma *k-nearest neighbor* (K-NN) bekerja pada *datasets* yang telah memiliki label. Sesuai dengan namanya yaitu "*nearest neighbor*", algoritma K-NN bekerja dengan melakukan kalkulasi untuk menemukan sampel-sampel tetangga terdekat untuk melakukan prediksi kelas terhadap suatu objek (Sarno dkk., 2022). Nilai K pada K-NN merepresentasikan jumlah tetangga terdekat pada data untuk menentukan kelas data yang akan

diprediksi. Tahapan yang dilakukan dalam algoritma *k-nearest neighbor* (K-NN) adalah sebagai berikut (Sarno dkk., 2022):

a. Melakukan inisialisasi *datasets.*
b. Menentukan jumlah nilai K yang merupakan tetangga terdekat.
c. Mengkalkulasi jarak antara *data training* dengan objek data yang tengah diklasifikasi (*data testing*).
d. Memasukkan jarak yang telah dikalkulasi ke dalam indeks data.
e. Mengurutkan jarak yang telah dikalkulasi dari yang terdekat hingga terjauh.
f. Memilih data training yang memiliki jarak paling dekat sejumlah nilai K atau jumlah tetangga yang telah ditentukan sebelumnya.
g. Menentukan label berdasarkan *data training* yang dipilih yang diperoleh dengan melihat kelas terbanyak dalam data.

Algoritma *k-nearest neighbor* (K-NN) menentukan jarak antar data berdasarkan perhitungan *euclidean distance* yang diperoleh berdasarkan teorema *pythagoras* dimana jarak

antara satu titik dengan titik lainnya dihitung secara langsung berdasarkan satu garis lurus dari satu poin data ke data lainnya (Sarno dkk., 2022). Perhitungan jarak dengan menggunakan *euclidean distance* satu dimensi, dua dimensi, dan multi dimensi dihitung berdasarkan persamaan 2.16, persamaan 2.17, dan persamaan 2.18 (Sarno dkk., 2022).

Satu dimensi:

$$d_{euc}(p,q) = \sqrt{(p-q)^2} \quad (2.16)$$

Dua dimensi:

$$d_{euc}(p,q) = \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2} \quad (2.17)$$

Multi dimensi:

$$d_{euc}(p,q) = \sqrt{\sum_{i=1}^{n}(p-q)^2} \quad (2.18)$$

12. Confusion Matrix

Confusion Matrix adalah salah satu metode yang dapat digunakan sebagai pengukur model atau algoritma secara spesifik (Saputro & Sari, 2019). Perhitungan *confusion matrix* dilakukan menghitung empat istilah dari nilai prediksi dan

nilai aktual. Hubungan nilai prediksi dengan nilai aktual dapat dilihat pada tabel 2.2.

Tabel 2.2 *Confusion Matrix*

	Predicted Negative	***Predicted Positive***
Actual Negative	*True Negative (TN)*	*False Positive (FP)*
Actual Positive	*False Negative (FN)*	*True Positive (FP)*

(Saputro & Sari, 2019)

Confusion matrix dapat digunakan untuk menghitung berbagai *performance metrics*. Beberapa *performance metrics* yang umumnya digunakan dalam melakukan validasi model *machine learning* antara lain *accuracy*, *precision*, *recall*, dan *f1-score.*

a. *Accuracy*

Perhitungan nilai *accuracy* dinyatakan dengan persamaan 2.19 (Saputro & Sari, 2019).

$$Accuracy = \frac{TP + TN}{TP+TN+FP+FN} \quad (2.19)$$

b. *Precision*

Perhitungan nilai *precision* dinyatakan dengan persamaan 2.20 (Saputro & Sari, 2019).

$$Precision = \frac{TP}{FP+TP} \quad (2.20)$$

c. *Recall*

Perhitungan nilai *recall* dinyatakan dengan persamaan 2.21 (Saputro & Sari, 2019).

$$Recall = \frac{TP}{FN+TP} \quad (2.21)$$

d. *F1-score*

Perhitungan *f1-score* dinyatakan dengan persamaan 2.22 (Saputro & Sari, 2019).

$$F1 = 2 \times \frac{precision \times recall}{precision+recall} \quad (2.22)$$

B. Kajian Penelitian yang Relevan

Penelitian terkait dengan potensipenyembuhan dalam al-qur'an dengan menganalisa data gelombang otak menggunakan algoritma *power spectral density* (PSD) dan *k-nearest neighbor* (K-NN) ini memiliki relevansi dengan penelitian sebelumnya yang ditunjukkan pada tabel 2.3.

Tabel 2.3 Kajian Penelitian yang Relevan

Peneliti	Judul Penelitian	Relevansi	Hasil Penelitian
(Sukmal dkk., 2019)	*Syifa'* dalam Perspektif Al-Qur'an	Al-Qur'an sebagai media penyembuhan	Menunjukkan bahwa dalam al-Qur'an terdapat penawar (syifa') terhadap segala jenis penyakit. Adapun cara yang digunakan berupa metode Neurofisiologi dan relaksasi transedensi.

(Mardiati dkk., 2018)	Pengaruh Terapi Psikoreligius: Membaca Al-Fatihah Terhadap Skor Halusinasi Pasien Skizofrenia	Al-Fatihah sebagai media penyembuhan	Menunjukkan penurunan nilai median dan terdapat pengaruh baik setelah pemberia terapi Al-Fatihah pada pasien skizofrenia.
(Husain & Aji, 2019)	Klasifikasi Sinyal *Elektroenseph alogram* dengan *Power Spectra Density* Berbasis Metode *Welch* dan *Multi Layer Perceptron Backpropagati on*	Penggunaan algoritma PSD pada data EEG	Penelitian ini berhasil mengklasifikasi kondisi normal dan epilepsi dengan menggunakan PSD, PCA, dan MLP dengan akurasi sebesar 99,68%.
(Aji & Tjandras a, 2017)	Klasifikasi EEG Epilepsi Menggunakan *Singular Spectrum Analysis*, *Power Spectral Density*, dan *Convolution Neural Network*	Penggunaan algoritma PSD pada data EEG	Penelitian ini berhasil mengklasifikasi sinyal EEG dengan target 5 kelas dan 3 kelas menggunakan metode SSA, PSD, dan CNN. Rata-rata akurasi 5 kelas sebesar 92%, dan untuk 3 kelas sebesar 94.4%

(Yean dkk., 2018)	*Analysis of The Distance Metics of KNN Classifier for EEG Signal in Stroke Patients*	Penggunaan algoritma KNN pada data EEG	Penelitian ini berhasil menganalisa hasil akurasi dengan 8 jarak matriks berbeda pada algoritma KNN sebanyak 3 kali dengan rata-rata akurasi pertama adalah 35 - 50%, rata-rata akurasi kedua adalah 40 - 65%, dan rata-rata akurasi ke-tiga pada jarak matriks adalah 30 - 40%.
(Almahdi dkk., 2021)	*EEG Signal Analysis for Epileptic Seizure Detection Using DWT Method with SVM and KNN Classifiers*	Penggunaan algoritma KNN pada data EEG	Penelitian ini berhasil mengklasifikasi sinyal EEG untuk mendeteksi penyakit epilepsi dengan penerapan metode SVM dengan akurasi 72,7% dan KNN dengan akurasi 81,8%.

Penelitian yang dilakukan (Sukmal dkk., 2019) menjelaskan keterkaitan dengan penyembuhan melalui pendekatan al-Qur'an namun masih berupa dalil-dalil yang terdapat dalam ayat al-Qur'an dan

belum dijelaskan secara empiris. Dilanjutkan dengan penelitian oleh (Mardiati dkk., 2018) yang membuktikan adanya pengaruh al-fatihah pada suatu penyakit. Sedangkan dalam empat penelitian selanjtunya menjelaskan metode yang dapat digunakan dalam menganalisis gelombang otak. Dengan beberapa penelitian terkait yang telah dijelaskan, peneliti tertarik untuk melakukan penelitian terkait bagaimana analisa gelombang otak manusia ketika diperdengarkan lantunan ayat suci al-Qur'an dalam kondisi tenang sebagai upaya mengetahui kebenaran al-Qur'an yang memiliki potensi penyembuhan. Al-Fatihah digunakan sebagai stimulus, algoritma PSD untuk ekstraksi fitur. Pelabelan dilakukan terhadap data dengan dua tingkat berdasarkan pengamatan kenaikan gelombang delta otak yaitu "terjadi kenaikan" dan "tidak terjadi kenaikan" untuk selanjutnya digunakan pada pembuatan model *machine learning* dengan algoritma *k-nearest neighbor* (K-NN).

BAB III
METODE PENELITIAN

A. Jenis Penelitian

Pendekatan penelitian yang digunakan dalam ini adalah pendekatan penelitian kuantitatif dengan eksperimen. Pendekatan penelitian kuantitatif merupakan salah satu teknik pendekatan penelitian yang digunakan untuk melakukan penelitian pada pupulasi atau sampel tertentu yang diambil secara acak (*random*) dengan pengumpulan datanya menggunakan instrumen serta menggunakan teknik analisa data statistik (Sugiyono, 2018).

B. Tempat dan Waktu Penelitian

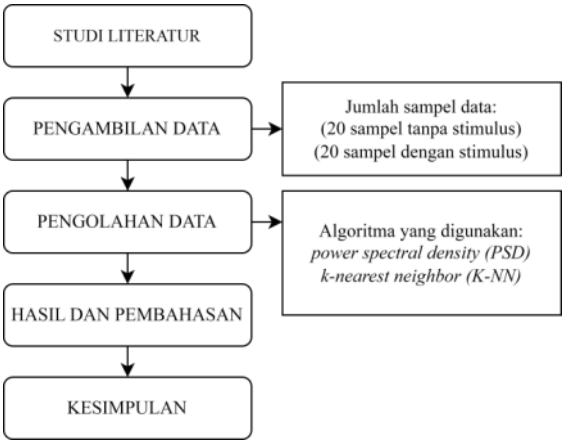
Pengambilan data penelitian bertempat di Laboratorium Terpadu *(Integrated Laboratory)* Kampus III Universitas Islam Negeri Walisongo Semarang dengan spesifik ruang lantai 3 Laboratorium Fisika Modern. Waktu studi literatur, pengambilan data penelitian, pengolahan data penelitian, dan analisis data penelitian dilakukan selama bulan September - November 2022, dan Januari – September 2023.

C. Sampel Penelitian

Pengambilan sampel penelitian dilakukan terhadap 20 partisipan berusia 18 – 19 tahun yang merupakan mahasiswa fisika semester pertama angkatan 2022 Universitas Islam Negeri Walisongo Semarang. Jumlah pengambilan data ini didasarkan pada teori Cohen yang menyatakan dalam metode eksperimental setidaknya memerlukan 15 partisipan (Cohen dkk., 2007). Pengambilan data gelombang otak pada masing-masing sampel penelitian dilakukan sebanyak 2 kali dengan total data sampel keseluruhan berjumlah 40 data yang terdiri atas 20 data gelombang otak tanpa stimulus dan 20 data gelombang otak dengan stimulus pada rentang waktu masing-masing 60 detik.

D. Prosedur Penelitian

Prosedur penelitian yang dilakukan melalui beberapa tahapan yang dimulai dari studi literatur hingga pengambilan kesimpulan yang ditunjukkan pada gambar 3.1.



Gambar 3.1 Prosedur Penelitian

Tahap pertama dalam penelitian yang dilakukan adalah melakukan studi literatur dari berbagai sumber ilmiah terkait topik penelitian. Tahap selanjutnya adalah pengambilan sampel data atau akuisisi data. Pengambilan sampel data gelombang otak dilakukan menggunakan instrumen elektoensefalografi (EEG) dengan menempelkan elektroda pada kulit kepala partisipan menggunakan sistem standar montase EEG 10-20. Tahapan setelah pengambilan sampel data adalah pengolahan data gelombang dengan menggunakan algoritma *power spectral density* (PSD) untuk mentransformasi data berbasis waktu (*time based*) menjadi data berbasis frekuensi (*frequency based*) dan algoritma *k-nearest*

neighbor (K-NN) untuk membuat model *machine learning* terhadap 2 kelas label yaitu "terjadi kenaikan" dan "tidak terjadi kenaikan" pada gelombang delta serta metode *convolutional matix* untuk menghitung nilai akurasi model yang dibuat dengan algoritma *k-nearest neighbor* (K-NN). Tahapan berikutnya adalah perumusan hasil dan pembahasan terhadap bagaimana pengolahan data yang telah dilakukan hingga tahap terakhir adalah penarikan kesimpulan terhadap penelitian yang telah dilakukan.

E. Instrumen dan Teknik Pengumpulan Data

Instrumen dan teknik pengumpulan data yang digunakan dalam penelitian ini antara lain adalah sebagai berikut:

1. Instrumen Pengumpulan Data

   Instrumen elektroensefalografi (EEG) yang digunakan adalah elektroensefalografi (EEG) contec KT88-1016 dengan 16 *channels* elektroda EEG + 2 *channels* elektroda EKG (alternatif). Instrumen pendukung lain dalam proses pengumpulan data antara lain topi kepala

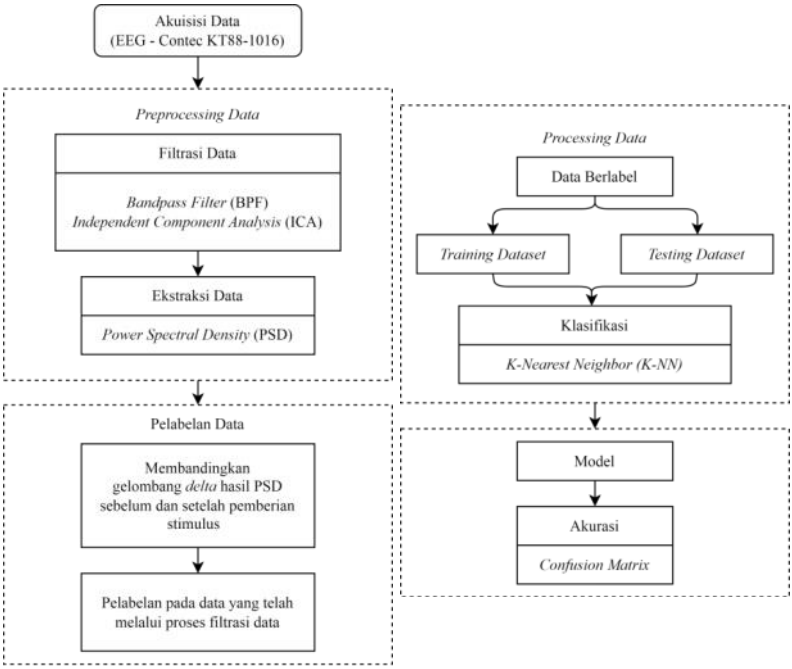
peletakan elektroda, pita pengukur elastis sebagai alat pengukuran peletakan elektroda, dan air bersih sebagai media konduktor untuk memudahkan elektroda menangkap gelombang pada kulit kepala.

2. Teknik Pengumpulan Data

Teknik pengumpulan data pada penelitian ini dilakukan dengan menggunakan kaidah sistem 10-20. Kaidah 10-20 EEG didasarkan pada rekomendasi komite federasi elektroensefalografi dan neurofisiologi klinik internasional (IFSECN) (Bintoro, 2012). Pengambilan sampel data dilakukan sebanyak 2 kali terhadap 20 partisipan dalam kondisi tenang dan mata tertutup, pengambilan pertama dilakukan pada kondisi tanpa pemberian stimulus untuk mendapatkan data gelombang otak normal partisipan, pengambilan data kedua dilakukan dengan pemberian stimulus pendengaran murottal surah al-Fatihah untuk mengetahui perbedaan perubahan gelombang pada otak partisipan. Masing-masing perekaman data dilakukan selama 60 detik.

F. Teknik Pengolahan Data

Pengolahan data pada penelitian ini dilakukan dengan bahasa pemrograman *python 3* dengan menggunakan beberapa paket pustaka (*package library*) antara lain *mne* untuk pengolahan data neurofisiologi otak dan *scikit-learn* untuk pembuatan model klasifikasi menggunakan algoritma K-NN. Pengolahan data dilakukan melalui 3 tahap dasar yaitu *preprocessing data*, *processing data*, dan perhitungan nilai akurasi model *machine learning*. Alur teknik pengolahan data secara detail ditunjukkan pada gambar 3.2.



Gambar 3.2 Teknik Pengolahan Data

1. *Preprocessing Data*

   Tahap *preprocessing data* dilakukan dengan menggunakan beberapa metode dengan tujuan memastikan data memiliki kualitas yang baik dan siap untuk diolah pada tahap *processing data*. *Preprocessing data* dilakukan melalui 3 tahapan yaitu filtrasi data, ekstraksi data, dan pelabelan data.

a. Filtrasi Data

Data akan dibatasi sesuai dengan *band frequency* yang telah ditentukan antara batas bawah gelombang *delta* hingga batas atas gelombang *gamma* yakni antara 0,5 – 40 Hz menggunakan *bandpass filter*. Metode *independent component analysis* (ICA) selanjutnya digunakan pada data untuk mencari yang membentuk komponen sinyal campuran (Firstanto dkk., 2013). Tujuan digunakannya metode *independent component analysis* (ICA) ini adalah untuk membersihkan data gelombang terhadap gangguan noise dan artefak yang masuk pada data gelombang otak.

b. Ekstraksi Data

Data mentah (*raw data*) yang didapatkan dari instrumen elektroensefalografi (EEG) adalah data berbasis waktu (*time based*), untuk dapat memudahkan dalam melakukan analisa data EEG maka perubahan data dari data berbasis waktu (*time based*) ke dalam data berbasis frekuensi (*frequency based*) penting untuk

dilakukan. Algoritma *power spectral density* (PSD) dengan menggunakan metode *welch* merupakan salah satu metode yang dapat digunakan untuk mentransformasi data berbasis waktu ke dalam data berbasis frekuensi. Metode *welch* dapat diterapkan dalam mengestimasi spektrum daya dari urutan waktu yang ditentukan (Ameera dkk., 2019). Pembagian frekuensi gelombang otak dibagi menjadi lima jenis gelombang berdasarkan rentang frekuensi yaitu gelombang *delta* (0,5 – 4 Hz), gelombang *theta* (4 – 8 Hz), gelombang *alpha* (8 – 12 Hz), gelombang *beta* (12 – 25 Hz), dan gelombang *gamma* (25 – 40 Hz).

c. Pelabelan Data

Pelabelan data gelombang dilakukan dengan mengklasifikasikan kondisi kedalam 2 kelas label yaitu “terjadi kenaikan” dan “tidak terjadi kenaikan” pada gelombang delta. Kondisi terjadi kenaikan diidentifikasi ketika terjadi peningkatan persentase gelombang *delta* saat pemberian stimulus, dan kondisi

tidak terjadi kenaikan terjadi ketika persentase gelombang *delta* yang dihasilkan setelah pemberian stimulus tidak mengalami peningkatan dan mengalami penurunan. Pelabelan diletakkan pada data *channel* elektroda yang telah di filtrasi menggunakan *bandpass filter* dan ICA. Gelombang *delta* menjadi indikator dalam potensi penyembuhan dimana pancaran gelombang delta mengindikasikan tubuh manusia mengalami proses pemulihan terhadap jaringan yang mengalami kerusakan dan melakukan regenerasi sel (Saminan, 2020).

2. *Processing Data*

Setelah data berlabel (*labeled data*) berhasil didapatkan maka tahap selanjutnya dalam *processing data* adalah membagi data ke dalam data latih (*data training*) dan data uji (*data testing*). Persentase data latih yang digunakan adalah sebesar 70% dari keseluruhan data dan data uji yang digunakan adalah sebesar 30% dari keseluruhan data. Selanjutnya dilakukan pembuatan model *machine learning*

menggunakan algoritma *k-nearest neighbor* (K-NN) berdasarkan data yang telah berlabel.

3. Perhitungan Akurasi

Tahap terakhir adalah perhitungan akurasi yang dilakukan untuk menguji performa model klasifikasi dengan beberapa parameter dalam metode *confusion matrix* berupa *accuracy*, *precision, recall*, dan *f1-score.*

BAB IV
HASIL DAN PEMBAHASAN

A. Pengumpulan Data Penelitian

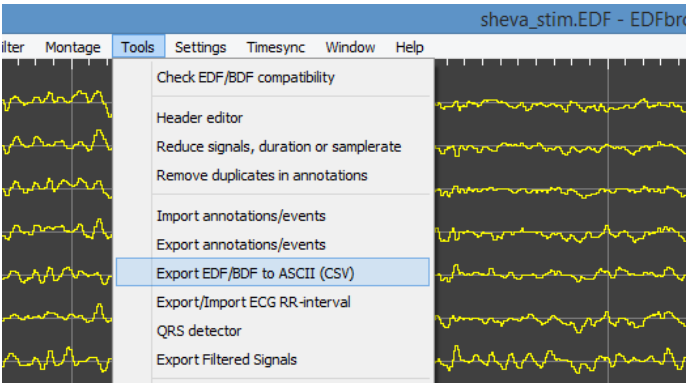
Proses pengumpulan atau akuisisi data penelitian dilakukan pada tiap mahasiswa partisipan yang diawali dengan persiapan instrumen berupa topi kepala peletakan elektroda EEG, elektroda EEG, instrumen EEG contec KT88-1016, dan pita pengukur elastis. Proses penggukuran dilakukan dengan menggunakan pita pengukur elastis pada kulit kepala partisipan. Posisi peletakan elektroda pada kulit kepala berpedoman pada kaidah 10-20 EEG yang didasarkan atas rekomendasi komite federasi elektroensefalogafi dan neurofisiologi klinik internasional (IFSECN) yang diperlihatkan pada gambar 2.2. Proses pengambilan data dilakukan setelah seluruh komponen elektroda telah terpasang dengan baik pada kulit kepala partisipan. Pengambilan data dilakukan dalam kondisi partisipan tengah duduk tenang dengan mata tertutup. Kondisi tenang ini dipilih dengan tujuan meminimalisir adanya artefak maupun *noise* yang muncul pada data EEG akibat adanya gangguan

maupun gerakan yang tidak diperlukan dan akan mengganggu proses analisis data nantinya. Pengambilan data pada tiap partisipan dilakukan sebanyak dua kali dengan rentang waktu selama 60 detik serta jeda waktu istirahat sebelum pemberian stimulus selama 60 detik. Pengambilan data pertama dilakukan dengan tanpa pemberian stimulus untuk menghasilkan data gelombang otak normal (batas dasar) partisipan yang dilanjutkan dengan jeda istirahat. Pengambilan data kedua dilakukan dengan pemberian rangsangan stimulus *murottal* surah al-Fatihah ayat 1-7 yang mewakili salah satu surah dalam al-Qur'an. Pemberian rangsangan stimulus ditujukan untuk mengetahui apakah ada perbedaan frekuensi gelombang otak pada kondisi normal (batas dasar) dan pada kondisi setelah pemberian stimulus. Data yang diperoleh pada proses pengambilan atau akuisisi data pada seluruh partisipan berjumlah 40 data sampel yang berupa 20 data gelombang tanpa stimulus dan 20 data gelombang dengan stimulus dengan rentang waktu masing-masing data selama 60 detik. Perekaman data dengan instrumen EEG menghasilkan data

berbasis waktu (*time based*) yang berupa hubungan antara tegangan terhadap waktu secara tegak lurus dengan sumbu vertikal (Y) sebagai tegangan dan sumbu (X) sebagai waktu (Deu, 2019). Perekaman dengan instrumen EEG menghasilkan data gelombang otak berekstensi EDF (*european data formating*) sebagai ekstensi standar data EEG.

B. Hasil Penelitian

Perekaman data gelombang otak dengan instrumen EEG contec KT88-1016 menghasilkan data EEG dengan format ekstensi EDF (*european data formating*) yang perlu dirubah ke dalam format ekstensi CSV (*comma separated value*) untuk memudahkan dan menyesuaikan kebutuhan dalam proses analisa data. Perubahan format data EDF ke CSV dilakuan dengan menggunakan perangkat lunak “EDF Browser” seperti yang ditampilkan dalam gambar 4.1.



Gambar 4.1 Konversi Data EDF ke CSV

Tahapan pengolahan data dilakukan setelah seluruh data EDF telah dirubah ke dalam ekstensi data CSV. Tahapan pengolahan data dalam penelitian ini telah dijabarkan pada gambar 3.2. Hasil pengolahan data diimplementasikan dengan menggunakan bahasa pemrograman *python* 3 dengan menggunakan beberapa paket pustaka (*package library*) untuk proses pengolahan data. Pengolahan data dilakukan melalui beberapa tahapan yaitu tahap *preprocessing data*, *processing data*, hingga tahap perhitungan nilai akurasi model.

1. *Preprocessing Data*

   Proses analisa data pada penelitian ini menggunakan perangkat lunak IDE (*integrated development environtment*) berbasis web untuk

bahasa pemrograman *python* milik *google* yaitu *google colaboratory*. Pengolahan data menggunakan python pada *google colaboratory* akan lebih mudah apabila data yang akan kita analisa telah kita simpan pada *google drive*, yaitu penyimpanan berbasis *cloud* yang terintegrasi dengan *google colaboratory*. Sebelum melakukan proses analisa data, maka data yang akan dianalisa terlebih dahulu disimpan dalam penyimpanan *google drive*. Setelah data berhasil disimpan, maka perlu mengimpor penyimpanan *google drive* pada *google colaboratory* dengan menggunakan kode perintah yang ditampilkan pada gambar 4.2.

```
1. from google.colab import drive
2. drive.mount('/content/drive')
```

Gambar 4.2 *Import Google Drive*

Gambar 4.3 mendefinisikan penggunaan *library* pandas yang digunakan untuk membaca file CSV ke dalam bentuk *dataframe* pandas yang ditujukan pada baris kode ke 3 kemudian menampilkan 5 data awal pada data EEG yang ditujukan dengan baris kode ke 4.

```
1. import pandas as pd
2. name_file = "/content/drive/MyDrive/nama_file.csv"
3. df = pd.read_csv(name_file)
4. df.head(5)
```

Gambar 4.3 read CSV file ke pandas

Tampilan 5 data awal pada data EEG ditujukan pada gambar 4.4.

```
   Time          1          2          3          4          5          6  \
0  0.00  22.372168  25.275044  21.671473  26.876631  11.861753  25.375143
1  0.01  12.162051  18.067903   6.456397  16.666514  -5.955901   9.058976
2  0.02   7.857786  12.362249   0.750744  10.860761 -12.862745  -0.050050
3  0.03   7.357290   6.356298  -0.850843   3.653620 -14.364233  -3.053025
4  0.04   4.554513   1.651637  -1.751736   3.353323 -11.461357   1.851835

          7          8          9         10         11         12  \
0   2.752728   7.857786  -8.057984 -21.070878  11.561456  22.071870
1 -12.362249   1.051041 -13.763638 -23.973754   0.750744  13.563439
2 -16.266117   0.950942  -7.457389 -17.367208  -2.752728   9.959869
3 -15.265126   6.756695   2.752728  -5.055009  -3.953918   9.459373
4  -9.859770  10.360266  10.059968  11.861753  -6.956893   5.855802

          13         14         15         16         17         18
0  11.861753  18.668498   8.458381   0.750744   9.659571   9.058976
1   1.251240   9.659571   2.152132  -6.656596   9.759670   8.758679
2  -0.550546   4.254215  -1.851835  -4.554513   9.159075   8.258183
3  -3.353323   7.557488  -5.755703   8.758679   9.058976   8.158083
4  -7.257191  14.164034  -6.556497  24.173953   9.359274   8.258183
```

Gambar 4.4 5 Data awal Dataframe

Penghapusan data yang tidak diperlukan yaitu kolom "Time" dapat menggunakan kode perintah yang ditunjukkan pada gambar 4.5.

```
1. df.drop("Time", axis=1, inplace=True)
```

Gambar 4.5 Menghapus kolom yang tidak diperlukan

Pendefinisian informasi data EEG perlu dilakukan agar data EEG dapat dianalisa menggunakan *library* MNE. Instalasi paket

library MNE dan pendefinisian informasi data EEG ditunjukkan pada gambar 4.6 dan gambar 4.7.

```
1.  !pip install mne
```

Gambar 4.6 Instalasi *Library* MNE

```
1.  sfreq = 100
2.  ch_types = [“eeg”]*df.shape[1]
3.  ch_names =
        ["Fp1","Fp2","F3","F4","C3","C4","P3","P4","O1","O2”,
        "F7","F8","T3", "T4","T5","T6","A1","A2"]
4.  montage =
        mne.channels.make_standard_montage("standard_10
        20")
5.  info = mne.create_info(ch_names=ch_names, sfreq=sfreq,
        ch_types=ch_types)
6.  samples = df.T*1e-6
7.  raw_data = mne.io.RawArray(samples, info)
8.  raw_data.set_montage(montage=montage)
```

Gambar 4.7 Pendefinisian Informasi Data EEG

Gambar 4.7 mendefinisikan beberapa informasi dasar pada data EEG yang akan dianalisa. Informasi data EEG pada penelitian ini berupa *sample* frekuensi pada baris kode ke 1, tipe *channel* yang berupa EEG pada baris ke 2, nama *channel* yang digunakan dalam instumen EEG pada baris ke 3, jenis montase pada baris ke 4, pembuatan variabel informasi dan sampel pada baris 5 dan 6, input raw data dengan menggunakan *library* mne dengan informasi data yang telah disiapkan pada baris

ke 7, dan mengatur jenis montase pada baris ke 8 dengan variabel *montage* yang telah dibuat.

a. Filtrasi Data

Filtrasi data dilakukan dengan tujuan membersihkan data dari *noise* dan artefak pada data EEG. Pada penelitian ini data EEG di filter dengan dua tahap filtrasi, yaitu pembatasan rentang frekuensi pada band frakuensi tertentu dengan *bandpass filter*, dan pembersihan *noise* dan artefak pada data EEG dengan menggunakan *independent component analysis* (ICA). Pada laporan tugas akhir ini hanya akan dijelaskan tahapan dalam mengimplementasikan filtrasi data pada salah satu data partisipan setelah stimulus. Pengimplementasian ini juga dilakukan pada 20 data partisipan sebelum dan 20 data setelah pemberian stimulus pada partisipan lainnya.

Filter bandpass diimplementasikan pada data EEG dengan tujuan membatasi rentang frekuensi yang hanya diperlukan pada tahap analisis. Rentang frekuensi yang diperlukan

pada tahap analisis berkisar antara 0,5 Hz hingga 40 Hz. Kode perintah filtrasi dengan *bandpass filter* ditunjukkan pada gambar 4.8.

```
1.  raw_data=raw_data.copy().filter(0.5, 40.,
        fir_design='firwin')
```

Gambar 4.8 *Banpass Filter Python*

Tampilan proses bandpass filter ditunjukkan pada gambar 4.9.

```
Filtering raw data in 1 contiguous segment
Setting up band-pass filter from 0.5 - 40 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 0.50
- Lower transition bandwidth: 0.50 Hz (-6 dB cutoff frequency: 0.25 Hz)
- Upper passband edge: 40.00 Hz
- Upper transition bandwidth: 10.00 Hz (-6 dB cutoff frequency: 45.00 Hz)
- Filter length: 661 samples (6.610 s)

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    0.0s finished
```

Gambar 4.9 Tampilan Proses *Banpass Filter*

Pembersihan data EEG terhadap noise dan artefak dilakukan dengan menerapkan metode *independent component analysis* (ICA) dengan menggunakan algoritma *fastICA.* Pada penelitian ini metode ICA digunakan untuk memecah komponen menjadi 15 komponen independen untuk

melihat komponen matriks tercampur pada data EEG. Kode petintah implementasi metode ICA dengan *python* ditunjukkan pada gambar 4.10.

```
1. from mne.preprocessing import ICA
2. ica_eeg = ICA(n_components=15, random_state=0)
3. ica_eeg.fit(raw_data)
```

Gambar 4.10 ICA *Python*

Tampilan proses *fitting* ICA ditunjukkan pada gambar 4.11.

```
Fitting ICA to data using 18 channels (please be patient, this may take a while)
Selecting by number: 15 components
Fitting ICA took 1.3s.

<ICA | raw data decomposition, method: fastica (fit in 55 iterations on 6000
samples), 15 ICA components (18 PCA components available), channel types: eeg,
no sources marked for exclusion>
```

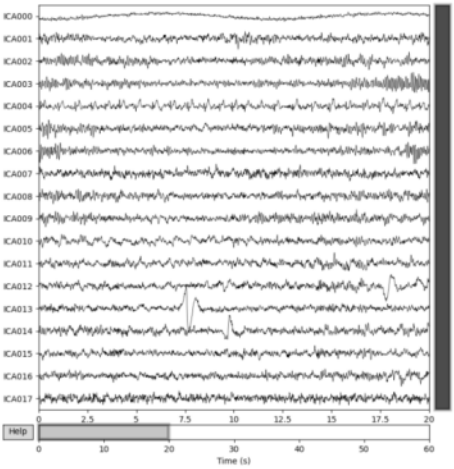
Gambar 4.11 *Fitting* ICA *Python*

Metode *independent component analysis* (ICA) merupakan metode yang memecah komponen menjadi komponen independen sejumlah n komponen, kode perintah pada gambar 4.12 adalah kode untuk dapat melihat seluruh komponen independen ICA pada data EEG dengan grafik pada baris kode ke 2.

```
1. import matplotlib.pyplot as plt
2. ica_eeg.plot_source(raw_data)
```

Gambar 4.12 *Plotting* ICA

Tampilan visualisasi *plotting* grafik ICA ditunjukkan pada gambar 4.13.



Gambar 4.13 Visualisasi Grafik ICA

Fungsi visualisasi *ploting* ICA adalah melihat adanya artefak pada data. Setelah letak artefak diketahui pada komponen ICA, pengecualian dilakukan dengan menggunakan fungsi "exclude" dengan tujuan mengecualikan komponen artefak pada data asal. Kode perintah untuk mengimplementasikan ICA setelah dilakukan pengecualian (*exclude*) ditunjukkan pada gambar 4.14 dimana pada baris kode ke 2 dilakukan pengecualian (*exclude*) pada komponen artefak ICA, dan

implementasi ICA pada data dilakukan pada baris kode ke 4.

```
1.  raw_data.load_data()
2.  ica_eeg.exclude = [13]
3.  reconst_data = raw_data.copy()
4.  ica_eeg.apply(reconst_data)
```

Gambar 4.14 Implementasi ICA

Tampilan implementasi ICA ditunjukkan pada gambar 4.15.

```
Applying ICA to Raw instance
    Transforming to ICA space (15 components)
    Zeroing out 1 ICA component
    Projecting back using 18 PCA components
```

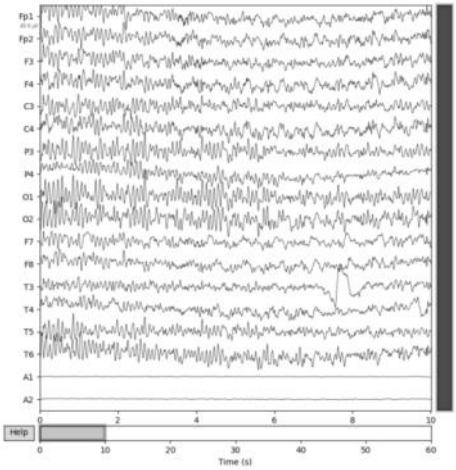
Gambar 4.15 Implementasi proses ICA

Perbandingan data sebelum dan sesudah proses filtrasi dapat dilihat dengan menampilkan visualisasi data dengan kode perintah yang ditunjukkan pada gambar 4.16.

```
    #data sebelum ICA
1.  raw_data.plot(duration=10, show=False)
    #data sesudah ICA
2.  reconst_data.plot(duration=10, show=False)
3.  plt.show()
```

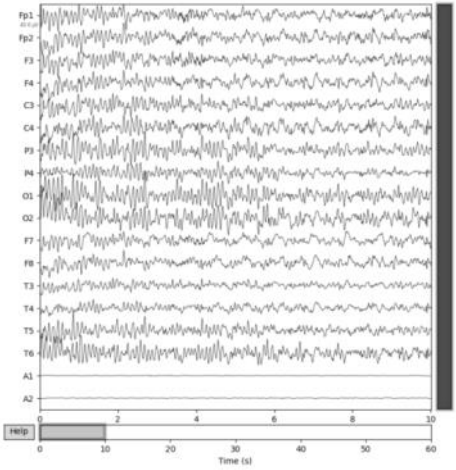
Gambar 4.16 Ploting sebelum dan sesudah ICA

Visualisasi data sebelum filtrasi dengan ICA ditunjukkan pada gambar 4.17.



Gambar 4.17 Visualisasi Data Asli sebelum ICA

Visualisasi data setelah filtrasi dengan ICA ditunjukkan pada gambar 4.18.



Gambar 4.18 Visualisasi Data Asli Sesudah ICA

Simpan data setelah filtrasi dengan menggunakan *library* EDFlib untuk dapat

menyimpan data dalam ekstensi EDF. Instalasi *library* EDFlib ditunjukkan pada gambar 4.19.

```
1.  !pip install EDFlib-Python
```

Gambar 4.19 Instalasi *library* EDFlib

Kode perintah untuk menyimpan data dengan ekstensi EDF ditunjukkan pada gambar 4.20.

```
1.  name_file_export =
        "/content/drive/MyDrive/nama_file_stim.edf"
2.  export = mne.export.export_raw(name_file_export,
        reconst_data, overwrite=True)
3.  export
```

Gambar 4.20 Menyimpan Data EDF

b. Ekstraksi Data

Perubahan data EEG berbasis waktu (*time based*) ke dalam data berbasis frekuensi (*frequency based*) dilakukan pada tahap ekstraksi data pada penelitian ini menggunakan algoritma *power spectral density* (PSD). Pada laporan tugas akhir ini hanya akan dijelaskan tahapan dalam melakukan transformasi data pada salah satu partisipan pada data EEG kemudian membandingkan data PSD sebelum stimulus

dan data setelah stimulus. Transformasi data dengan menggunakan *power spectral density* (PSD) pada data EEG ini juga dilakukan pada 20 data partisipan sebelum dan 20 data setelah pemberian stimulus pada partisipan lainnya.

Impor data EDF dengan menggunakan library MNE pada data yang telah diolah sebelumnya pada tahap filtrasi dengan menggunkan kode perintah yang ditunjukkan pada gambar 4.21.

```
1. import mne
2. xfil_data_nonstim=mne.io.read_raw_edf(
        '/content/drive/MyDrive/nama_file_nonstim.edf',
        preload=True)
3. xfil_data_stim=mne.io.read_raw_edf(
        '/content/drive/MyDrive/nama_file_stim.edf',
        preload=True)
```

Gambar 4.21 Mengimpor Data EDF

Rentang pembagian frekuensi yang digunakan dalam penelitian ini antara lain dibagi menjadi 5 jenis frekuensi gelombang antara lain gelombang *delta* (0,5 – 4 Hz), gelombang *theta* (4 – 8 Hz), gelombang *alpha* (8 – 12 Hz), gelombang *beta* (12 – 25 Hz), dan gelombang *gamma* (25 – 40 Hz). Pendefinisian variabel dengan rentang

gelombang yang telah ditentukan ditunjukkan pada gambar 4.22 dengan pendefinisian rentang waktu minimum dan maksimum pada baris kode ke 1, dan pendefinisian variabel rentang frekuensi secara berurutan dari gelombang *delta* hingga *gamma* pada kode ke 2 sampai 6.

```
1. tmin, tmax = 0, 60
2. lower_delta, higher_delta = 0.5, 4.0
3. lower_theta, higher_theta = 4.0, 8.0
4. lower_alpha, higher_alpha = 8.0, 12.0
5. lower_beta, higher_beta = 12.0, 25.0
6. lower_gamma, higher_gamma = 25.0, 40.0
```

Gambar 4.22 Pendefinisian Rentang Frekuensi EEG

Kode perintah untuk mengatur ukuran plot visualisasi data ditunjukkan pada gambar 4.23.

```
1. import matplotlib.pyplot as plt
2. plt.style.use('seaborn-bright')
3. plt.rcParams['figure.dpi']=300
4. plt.rcParams['figure.figsize']=(8, 6)
```

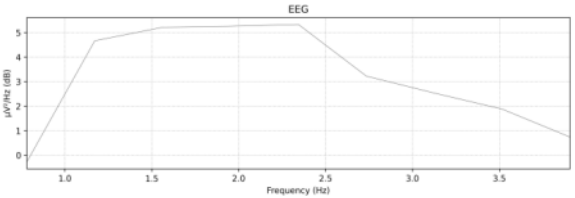
Gambar 4.23 Pengaturan Visualisasi *Ploting*

Kode perintah pada gambar 4.24 merupakan kode perintah untuk menampilkan *ploting* gelombang *delta* pada kondisi tanpa stimulus.

```
1.  xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_delta, fmax=higher_delta,
        show=False, average=True, color='black',
        area_mode=None)
```

Gambar 4.24 Kode Delta Non-Stimulus

Tampilan *ploting* gelombang *delta* pada kondisi tanpa stimulus ditunjukkan dengan gambar 4.25.



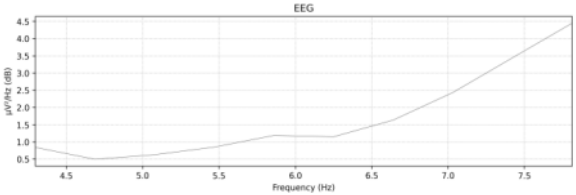
Gambar 4.25 Plot Delta Non-Stimulus

Kode perintah pada gambar 4.26 merupakan kode perintah untuk menampilkan *ploting* gelombang *theta* pada kondisi tanpa stimulus.

```
1.  xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_theta, fmax=higher_theta,
        show=False, average=True, color='black',
        area_mode=None)
```

Gambar 4.26 Kode *Theta* Non-Stimulus

Tampilan *ploting* gelombang *theta* pada kondisi tanpa stimulus ditunjukkan dengan gambar 4.27.



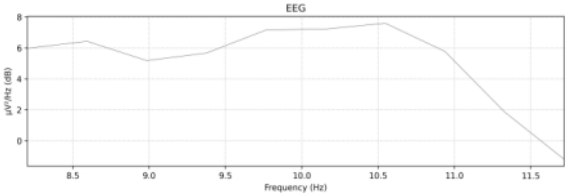
Gambar 4.27 *Plot Theta* Non-Stimulus

Kode perintah pada gambar 4.28 merupakan kode perintah untuk menampilkan *ploting* gelombang *alpha* pada kondisi tanpa stimulus.

```
1.  xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_alpha, fmax=higher_alpha,
        show=False, average=True, color='black',
        area_mode=None)
```

Gambar 4.28 Kode *Alpha* Non-Stimulus

Tampilan *ploting* gelombang *alpha* pada kondisi tanpa stimulus ditunjukkan dengan gambar 4.29.



Gambar 4.29 Plot *Alpha* Non-Stimulus

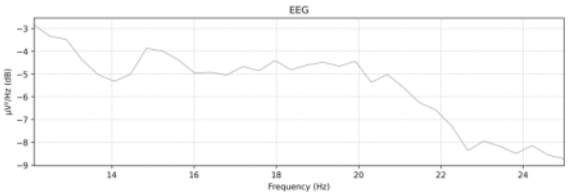
Kode perintah pada gambar 4.30 merupakan kode perintah untuk

menampilkan *ploting* gelombang *beta* pada kondisi tanpa stimulus.

```
1.  xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_beta, fmax=higher_beta,
        show=False, average=True, color='green',
        area_mode=None)
```

Gambar 4.30 Kode *Beta* Non-Stimulus

Tampilan *ploting* gelombang *beta* pada kondisi tanpa stimulus ditunjukkan dengan gambar 4.31.



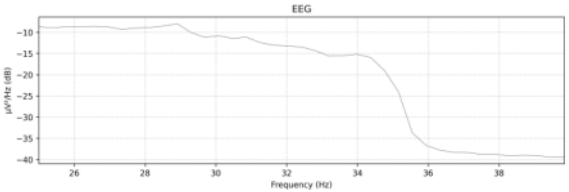
Gambar 4.31 Plot *Beta* Non-Stimulus

Kode perintah pada gambar 4.32 merupakan kode perintah untuk menampilkan *ploting* gelombang *gamma* pada kondisi tanpa stimulus.

```
1.  xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_gamma, fmax=higher_gamma,
        show=False, average=True, color='green',
        area_mode=None)
```

Gambar 4.32 Kode *Gamma* Non-Stimulus

Tampilan *ploting* gelombang *gamma* pada kondisi tanpa stimulus ditunjukkan dengan gambar 4.33.



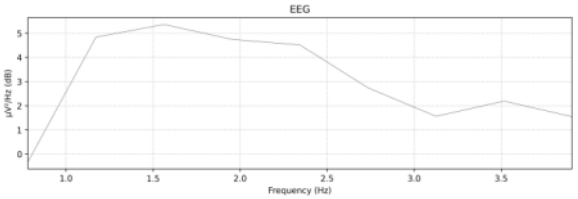
Gambar 4.33 Plot *Gamma* Non-Stimulus

Kode perintah pada gambar 4.34 merupakan kode perintah untuk menampilkan *ploting* gelombang *delta* pada kondisi dengan stimulus.

```
1.  xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_delta, fmax=higher_delta,
        show=False, average=True, color='black',
        area_mode=None)
```

Gambar 4.34 Kode *Delta* Stimulus

Tampilan *ploting* gelombang *delta* pada kondisi dengan stimulus ditunjukkan dengan gambar 4.35.



Gambar 4.35 Plot *Delta* Stimulus

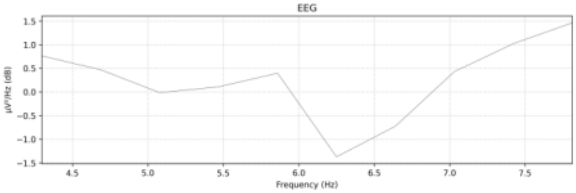
Kode perintah pada gambar 4.36 merupakan kode perintah untuk

menampilkan *ploting* gelombang *theta* pada kondisi dengan stimulus.

```
1.  xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_theta, fmax=higher_theta,
        show=False, average=True, color='black',
        area_mode=None)
```

Gambar 4.36 Kode *Theta* Stimulus

Tampilan *ploting* gelombang *theta* pada kondisi dengan stimulus ditunjukkan dengan gambar 4.37.



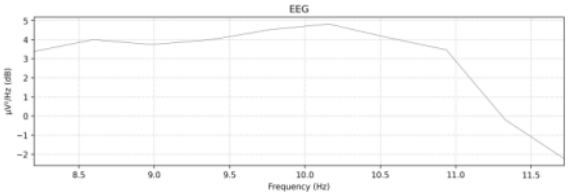
Gambar 4.37 Plot *Theta* Stimulus

Kode perintah pada gambar 4.38 merupakan kode perintah untuk menampilkan *ploting* gelombang *alpha* pada kondisi dengan stimulus.

```
1.  xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_alpha, fmax=higher_alpha,
        show=False, average=True, color='black',
        area_mode=None)
```

Gambar 4.38 Kode *Alpha* Stimulus

Tampilan *ploting* gelombang *alpha* pada kondisi dengan stimulus ditunjukkan dengan gambar 4.39.



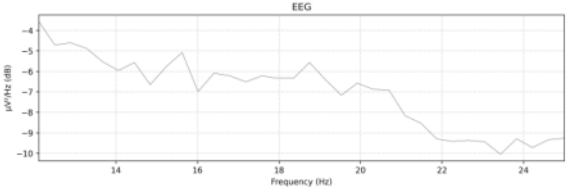
Gambar 4.39 Plot *Alpha* Stimulus

Kode perintah pada gambar 4.40 merupakan kode perintah untuk menampilkan *ploting* gelombang *beta* pada kondisi dengan stimulus.

```
1.  xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_beta, fmax=higher_beta,
        show=False, average=True, color='green',
        area_mode=None)
```

Gambar 4.40 Kode *Beta* Stimulus

Tampilan *ploting* gelombang *beta* pada kondisi dengan stimulus ditunjukkan dengan gambar 4.41.



Gambar 4.41 Plot *Beta* Stimulus

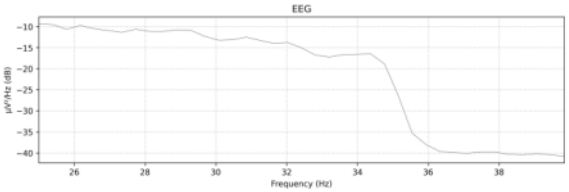
Kode perintah pada gambar 4.42 merupakan kode perintah untuk

menampilkan *ploting* gelombang *gamma* pada kondisi dengan stimulus.

```
1.  xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax,
        fmin=lower_gamma, fmax=higher_gamma,
        show=False, average=True, color='green',
        area_mode=None)
```

Gambar 4.42 Kode *Gamma* Stimulus

Tampilan *ploting* gelombang *gamma* pada kondisi dengan stimulus ditunjukkan dengan gambar 4.43.



Gambar 4.43 Plot *Gamma* Stimulus

Pengimplementasian *power spectral density* (PSD) dapat memanfaatkan fungsi psd_welch() yang tersedia dalam paket *library* MNE. Kode perintah pada gambar 4.44 merupakan kode perintah untuk mengimplemensatikan *metode power spectral density* (PSD).

```
1.  import numpy as np
2.  from mne.time_frequency import psd_welch
3.  def eeg_power_band(raw):
4.      FREQ_BANDS = {'delta': [0.5, 4.0],
                      'theta': [4.0, 8.0],
                      'alpha': [8.0, 12.0],
                      'beta': [12.0, 25.0],
                      'gamma': [25.0, 40.0]}
5.      psds, freqs = psd_welch(raw, picks='eeg',
                      fmin=0.5, fmax=40)

        # Normalisasi psd
6.      psds /= np.sum(psds, axis=-1, keepdims=True)

7.      X = []
8.      for fmin, fmax in FREQ_BANDS.values():
9.          psds_band = psds[:, (freqs >= fmin) & (freqs
                < fmax)].mean(axis=-1)
10.         X.append(psds_band.reshape(len(psds), -1))
11.     return np.concatenate(X, axis=1)
    #menghitung psd
12. power_psd_nonstim =
        eeg_power_band(xfil_data_nonstim)
13. power_psd_stim =
        eeg_power_band(xfil_data_stim)
```

Gambar 4.44 Pembuatan Fungsi eeg_power_band

Gambar 4.44 mendefinisikan fungsi untuk melakukan perhitungan metode *power spectral density* (PSD) dengan menggunakan fungsi dalam paket *library* MNE yaitu psd_welch(), pembuatan fungsi eeg_power_band() pada baris kode ke 3 dimulai dengan pendefinisian *dictionary* band frekuensi spesifik pada baris kode ke 4. Pemanggilan fungsi psd_welch() dilakukan pada baris kode ke 5, dan normalisasi data

ditunjukkan pada baris kode ke 6. Pada baris ke 7-11 perulangan (*looping*) dilakukan untuk setiap band frekuensi yang telah didefinisikan. Hasil perhitungan disimpan dalam variabel power_psd_nonstim dan power_psd_stim pada baris kode ke 12 dan 13.

Dataframe pandas digunakan untuk menampilkan data PSD dalam bentuk tabel. Kode perintah pada gambar 4.45 merupakan kode perintah untuk menampilkan data rerata daya band PSD pada data dalam keadaan tanpa stimulus.

```
1.  import pandas as pd
2.  np_arr_nonstim = np.array(power_psd_nonstim)
3.  df_psd_nonstim = pd.DataFrame(np_arr_nonstim,
                index=xfil_data_nonstim.info['ch_names'],
                columns=['delta',
                    'theta',
                    'alpha',
                    'beta',
                    'gamma'])
4.  df_psd_nonstim.head(18)
```

Gambar 4.45 *Average Power* PSD Non-Stimulus

Tampilan *dataframe* rerata daya band PSD pada data dalam keadaan tanpa stimulus ditunjukkan dengan gambar 4.46.

	delta	theta	alpha	beta	gamma
Fp1	0.036126	0.016107	0.040823	0.002479	0.000609
Fp2	0.031840	0.015225	0.044618	0.002553	0.000789
F3	0.028455	0.018324	0.044057	0.002747	0.000755
F4	0.027876	0.017581	0.044124	0.003058	0.000798
C3	0.021497	0.017745	0.050358	0.003175	0.000531
C4	0.032952	0.017551	0.040091	0.003257	0.000500
P3	0.016160	0.018530	0.056783	0.002646	0.000362
P4	0.021037	0.018179	0.053010	0.002603	0.000330
O1	0.010451	0.014194	0.066153	0.002720	0.000326
O2	0.015534	0.013471	0.061679	0.002934	0.000304
F7	0.041259	0.016450	0.035916	0.002529	0.000553
F8	0.039362	0.016173	0.037350	0.002646	0.000595
T3	0.022198	0.016735	0.051023	0.003036	0.000576
T4	0.032815	0.017455	0.038928	0.003432	0.000707
T5	0.015741	0.017176	0.054246	0.003532	0.000707
T6	0.018280	0.014971	0.057284	0.002959	0.000392
A1	0.034388	0.019281	0.010947	0.008926	0.002402
A2	0.028400	0.020196	0.013471	0.009230	0.002644

Gambar 4.46 *Dataframe Average Power* PSD Non-Stimulus

Kode perintah pada gambar 4.47 merupakan kode perintah untuk menampilkan data rerata daya band PSD pada data dalam keadaan dengan stimulus.

```
1.  np_arr_stim = np.array(power_psd_stim)
2.  df_psd_stim = pd.DataFrame(np_arr_stim,
                    index=xfil_data_nonstim.info['ch_names'],
                    columns=['delta',
                        'theta',
                        'alpha',
                        'beta',
                        'gamma'])
3.  df_psd_stim.head(18)
```

Gambar 4.47 *Average Power* PSD Stimulus

Tampilan *dataframe* rerata daya band PSD pada data dalam keadaan dengan stimulus ditunjukkan dengan gambar 4.48.

	delta	theta	alpha	beta	gamma
Fp1	0.045930	0.014924	0.031778	0.002743	0.000746
Fp2	0.045431	0.015366	0.033026	0.002328	0.000779
F3	0.040947	0.017379	0.031591	0.003348	0.000802
F4	0.036346	0.019224	0.033721	0.003420	0.000784
C3	0.031122	0.020302	0.038313	0.003438	0.000520
C4	0.037013	0.018975	0.034235	0.003456	0.000531
P3	0.025270	0.019247	0.047236	0.002928	0.000285
P4	0.027775	0.015327	0.045926	0.003403	0.000646
O1	0.016094	0.014392	0.060848	0.002748	0.000309
O2	0.023166	0.012700	0.056221	0.002734	0.000309
F7	0.053725	0.016045	0.025815	0.002375	0.000500
F8	0.049529	0.015538	0.028749	0.002687	0.000582
T3	0.037186	0.018489	0.034243	0.003501	0.000576
T4	0.041571	0.018330	0.029385	0.003655	0.000720
T5	0.024931	0.017199	0.046089	0.003635	0.000585
T6	0.028515	0.013444	0.049606	0.002862	0.000472
A1	0.034118	0.016072	0.014755	0.008596	0.002589
A2	0.033188	0.020683	0.012195	0.008240	0.002580

Gambar 4.48 *Dataframe Average Power* PSD Stimulus

Perhitungan nilai persentase gelombang pada kondisi tanpa stimulus dan dengan stimulus dilakukan dengan menghitung rata-rata keseluruhan pada *datframe average power* PSD yang telah dibuat sebelumnya. Kode perintah dalam perhitungan nilai persentase gelombang pada kondisi tanpa stimulus dan dengan stimulus ditunjukkan pada gambar 4.49 dan 4.51.

```
#menghitung nilai persentase gelombang pada kondisi non-stimulus
df_nstim = df_psd_nonstim
df_nstim['total'] = df_nstim.sum(axis=1)
df_nstim['%delta'] = df_nstim['delta'] / df_nstim['total']
df_nstim['%theta'] = df_nstim['theta'] / df_nstim['total']
df_nstim['%alpha']= df_nstim['alpha'] / df_nstim['total']
df_nstim['%beta'] = df_nstim['beta'] / df_nstim['total']
df_nstim['%gamma']=df_nstim['gamma']/df_nstim['total']
pr_nstim_delta = np.array(df_nstim['%delta'].mean())
pr_nstim_theta = np.array(df_nstim['%theta'].mean())
pr_nstim_alpha = np.array(df_nstim['%alpha'].mean())
pr_nstim_beta = np.array(df_nstim['%beta'].mean())
pr_nstim_gamma = np.array(df_nstim['%gamma'].mean())
print("Persentase Gelombang pada Keadaan Non Stimulus\n")
array_nonstim = (
        ['delta','{:.0%}'.format(pr_nonstim_delta)],
        ['theta', '{:.0%}'.format(pr_nonstim_theta)],
        ['alpha', '{:.0%}'.format(pr_nonstim_alpha)],
        ['beta', '{:.0%}'.format(pr_nonstim_beta)],
        ['gamma', '{:.0%}'.format(pr_nonstim_gamma)])
df_persentase_nstim = pd.DataFrame(data=array_nstim,
        columns=['gelombang','persentase'])
df_persentase_nstim.head()
```

Gambar 4.49 Kode Nilai Persentase Gelombang Pada Kondisi Non-Stimulus

Tampilan nilai persentase gelombang pada kondisi tanpa stimulus ditunjukkan dengan gambar 4.50.

```
Persentase Gelombang pada Keadaan Non Stimulus
```

	gelombang	persentase
0	delta	29%
1	theta	19%
2	alpha	48%
3	beta	4%
4	gamma	1%

Gambar 4.50 Hasil Nilai Persentase Gelombang Pada Kondisi Non-Stimulus

```
#menghitung nilai persentase gelombang pada kondisi
stimulus
df_stim = df_psd_stim
df_stim['total'] = df_stim.sum(axis=1)
df_stim['%delta'] = df_stim['delta'] / df_stim['total']
df_stim['%theta'] = df_stim['theta'] / df_stim['total']
df_stim['%alpha'] = df_stim['alpha'] / df_stim['total']
df_stim['%beta'] = df_stim['beta'] / df_stim['total']
df_stim['%gamma'] = df_stim['gamma'] / df_stim['total']
pr_stim_delta = np.array(df_stim['%delta'].mean())
pr_stim_theta = np.array(df_stim['%theta'].mean())
pr_stim_alpha = np.array(df_stim['%alpha'].mean())
pr_stim_beta = np.array(df_stim['%beta'].mean())
pr_stim_gamma = np.array(df_stim['%gamma'].mean())
print("Persentase Gelombang pada Keadaan dengan
Stimulus\n")
array_stim = (['delta', '{:.0%}'.format(pr_stim_delta)],
    ['theta', '{:.0%}'.format(pr_stim_theta)],
    ['alpha', '{:.0%}'.format(pr_stim_alpha)],
    ['beta', '{:.0%}'.format(pr_stim_beta)],
    ['gamma', '{:.0%}'.format(pr_stim_gamma)])
df_persentase_stim = pd.DataFrame(data=array_stim,
  columns=['gelombang', 'persentase'])
df_persentase_stim.head()
```

Gambar 4.51 Kode Nilai Persentase Gelombang Pada Kondisi Stimulus

Tampilan nilai persentase gelombang pada kondisi dengan stimulus ditunjukkan dengan gambar 4.52.

```
Persentase Gelombang pada Keadaan dengan Stimulus
```

	gelombang	persentase
0	delta	38%
1	theta	18%
2	alpha	39%
3	beta	4%
4	gamma	1%

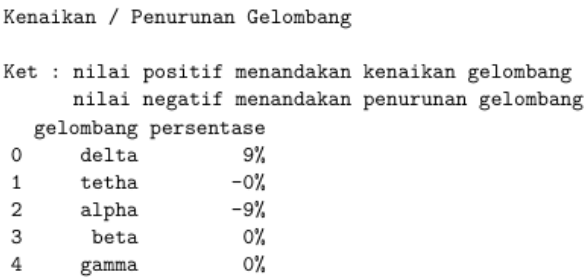
Gambar 4.52 Hasil Nilai Persentase Gelombang Pada Kondisi Stimulus

Kenaikan dan penurunan gelombang pada tiap jenis gelombang didapatkan dengan menghitung selisih masing-masing gelombang pada data tanpa stimulus dan data setelah stimulus. Kode perintah untuk menghitung selisih persentase tiap gelombang pada data tanpa stimulus dan data setelah stimulus ditunjukkan pada gambar 4.53.

```
    #menghitung nilai persentase gelombang (kenaikan dan
    penurunan)
1.  akhir_delta =  pr_stim_delta - pr_nonstim_delta
2.  akhir_theta =  pr_stim_theta - pr_nonstim_theta
3.  akhir_alpha =  pr_stim_alpha - pr_nonstim_alpha
4.  akhir_beta =  pr_stim_beta - pr_nonstim_beta
5.  akhir_gamma = pr_stim_gamma - pr_nonstim_gamma
    #menampilkan hasil persentase
6.  print("Kenaikan / Penurunan Gelombang \n")
7.  print("Ket : nilai positif menandakan kenaikan
    gelombang \n")
8.  print("     nilai negatif menandakan penurunan
    gelombang")
9.  array_akhir = (['delta', '{:.0%}'.format(akhir_delta)],
                   ['tetha', '{:.0%}'.format(akhir_theta)],
                   ['alpha', '{:.0%}'.format(akhir_alpha)],
                   ['beta', '{:.0%}'.format(akhir_beta)],
                   ['gamma', '{:.0%}'.format(akhir_gamma)])
10. df_persentase_akhir = pd.DataFrame(data=array_akhir,
                columns=['gelombang', 'persentase'])
11. df_persentase_akhir.head()
```

Gambar 4.53 Persentase Kenaikan dan Penurunan pada Tiap Gelombang

Tampilan hasil kenaikan dan penurunan gelombang pada tiap jenis gelombang ditunjukkan dengan gambar 4.54.



```
Kenaikan / Penurunan Gelombang

Ket : nilai positif menandakan kenaikan gelombang
      nilai negatif menandakan penurunan gelombang
  gelombang persentase
0     delta         9%
1     tetha        -0%
2     alpha        -9%
3      beta         0%
4     gamma         0%
```

Gambar 4.54 Persentase Kenaikan dan Penurunan pada Tiap Gelombang

c. Pelabelan Data

Pelabelan data dilakukan setelah seluruh data ditransformasikan ke dalam bentuk data berbasis frekuensi (*frequency based*) dengan menggunakan algoritma *power spectral density* (PSD). Perubahan data ke dalam bentuk frekuensi akan memudahkan dalam proses analisa dengan mengamati perubahan kenaikan gelombang pada tiap kondisi, baik pada kondisi tanpa pemberian stimulus maupun kondisi pasca pemberian stimulus pada tiap partisipan. Data keseluhan persentase gelombang pada

partisipan sebelum dan setelah pemberian stimulus ditunjukkan pada lampiran 7.

Pelabelan data pada penelitian ini diklasifikasikan menjadi dua label berbeda berdasarkan perubahan persentase gelombang delta otak sebelum dan setelah pemberian stimulus murottal surah al-Fatihah. Analisa pada perubahan gelombang delta dipilih karena penyembuhan dan regenerasi terstimulus saat dalam gelombang delta (Garg & Keswani, 2017). Tubuh manusia mengalami proses pemulihan melakukan regenerasi sel (Filcek, 2023; Saminan, 2020). Munculnya gelombang delta dapat menjadi potensi terjadinya proses penyembuhan dalam tubuh manusia.

Proses pelabelan dilakukan dengan menggunakan bahasa pemrograman *python* pada data mentah yang telah melalui proses filtrasi. Kode perintah dalam melakukan labelisasi dilakukan melalui beberapa tahap. Tahap pertama adalah membaca data csv ke dalam bentuk *dataframe* pandas seperti yang

ditunjukkan pada gambar 4.3 sebelumnya. Kemudian penghapusan kolom yang tidak diperlukan yaitu kolom “Time” dilakukan dengan menggunakan kode perintah pada gambar 4.5 yang telah ditunjukkan sebelumnya. Data mentah yang akan digunakan merupakan data seluruh channel dari tiap partisipan dengan jumlah kolom sebanyak 18 dan jumlah baris sebanyak 6000 data, untuk dapat memberikan label pada tiap data partisipan maka transformasi data diperlukan dengan merubah data menjadi data satu baris dengan menggunakan kode perintah yang ditunjukkan pada gambar 4.55.

```
1.  df_stack = df.stack()
2.  df_stack = df_stack.reset_index(drop=True)
3.  df_row = df_stack.to_frame(name='channel').reset_index()
4.  df_transpose = df_row.transpose()
5.  name_file_transpose = 'nama_file_transpose.csv'
6.  df_transpose.to_csv(name_file_transpose, index=False)
7.  df_col = pd.read_csv(name_file_transpose)
8.  df_col.drop(index=0, inplace=True)
```

Gambar 4.55 Transformasi Data Satu Baris

Pelabelan dibuat dengan menggunakan bilangan biner 0 yang berarti tidak terjadi kenaikan pada gelombang delta atau terjadi

penurunan pada gelombang delta, sedangkang 1 berarti terjadi kenaikan pada gelombang delta. Kode perintah dalam melakukan pelabelan pada *dataframe* dan menampilkan 5 data awal pada *dataframe* ditunjukkan pada gambar 4.56.

```
#note: 0=delta_down, 1=delta_up
df_col["label"] = 1
df_col.head()
```

Gambar 4.56 *Data Labelling*

Tampilan 5 data awal pada *dataframe* yang telah berlabel ditunjukkan pada gambar 4.57.

```
          0         1         2         3         4         5         6  \
1 -0.001665 -0.008432  0.176971 -0.005725  0.001041  0.003748  0.003748

          7         8         9  …    107991    107992    107993    107994  \
1  0.048407  0.007808  0.001041  …  0.001041 -0.062564  0.006454 -0.020612

     107995    107996    107997    107998    107999  label
1  0.002394  0.083593  0.015928 -0.000312 -0.001665      1

[1 rows x 108001 columns]
```

Gambar 4.57 Tampilan 5 Data Awal *Dataframe* Berlabel

Simpan data berlabel dengan menggunakan kode perintah yang ditunjukkan pada gambar 4.58.

```
df_col.to_csv("nama_file_berlabel.csv", index=False)
```

Gambar 4.58 Menyimpan Data Berlabel

Pemberian label ini kemudian dilakukan secara berurutan pada 20 data pada partisipan pasca pemberian stimulus dengan label 0 berarti tidak terjadi kenaikan delta dan label 1 ketika terjadi kenaikan gelombang delta. Setelah pemberian label telah dilakukan pada seluruh data gelombang partisipan pasca pemberian stimulus, penyatuan data dilakukan sebelum melangkah ke tahap pembuatan model *machine learning* pada tahap *processing data*. Kode perintah untuk menyatukan data CSV pada satu folder ditunjukkan pada gambar 4.59.

```
1.  import os
2.  import glob
3.  import pandas as pd
4.  os.chdir("/folder_file_data_berlabel")
5.  extension = 'csv'
6.  all_files = [i for i in glob.glob('*.{}'.format(extension))]
7.  comb_files = pd.concat([pd.read_csv(f) for f in all_files])
8.  comb_files.to_csv('nama_file_data_berlabel_lengkap.csv',
        index=False)
```

Gambar 4.59 Menyatukan Data Berlabel

2. Processing Data

Proses setelah didapatkan data berlabel adalah pembuatan model *machine learning*. Data berlabel yang telah disatukan diimpor

kembali ke dalam bentuk *dataframe* dengan menggunakan kode perintah yang ditunjukkan pada gambar 4.60.

```
1.  import pandas as pd
2.  df_eeg =
        pd.read_csv('/content/drive/MyDrive/
        eeg_data_stim_alfatihah/
        nama_file_data_berlabel_lengkap.csv')
3.  df_eeg.head()
```

Gambar 4.60 Impor *Dataframe* Data Berlabel yang Telah Disatukan

Tampilan 5 data awal pada *dataframe* berlabel yang telah disatukan ditunjukkan pada gambar 4.61.

```
          0         1         2         3         4         5         6  \
0  0.000319  0.000319  0.000319  0.000319  0.000319  0.000319  0.000319
1  0.001285  0.001285  0.001285  0.001285  0.001285  0.001285  0.001285
2  0.001803  0.001803 -0.002239 -0.000218  0.003824 -0.002239 -0.000218
3  0.004527  0.004527  0.004527  0.004527  0.004527  0.004527  0.004527
4  0.021220  0.004357  0.004357  0.004357  0.004357  0.004357  0.004357

          7         8         9  …     107991    107992     107993  \
0  0.000319  0.000319  0.000319  …  -3.185368 -3.810312   0.556671
1  0.001285  0.001285  0.001285  … -14.106295 -3.829575   1.838489
2 -0.000218 -0.000218  0.003824  …   9.587521 -1.057214 -11.988934
3  0.004527  0.004527  0.004527  … -12.162396  5.730739 -11.524449
4  0.012789  0.000141  0.038084  …   0.038084  0.004357   0.004357

     107994     107995    107996     107997    107998    107999  label
0 -6.441552 -13.401670  8.301778  -4.513373 -0.140674 -0.186402      0
1 -4.319049  -6.799945 -1.008953  -9.703264 -0.615586  0.121977      0
2 -0.228594 -11.006716 -2.463847   5.565683  0.282725  0.369629      0
3  7.358779  -9.911719  9.354277 -12.846275  0.708821 -0.220030      1
4  0.004357   0.004357  0.004357   0.004357  0.000141  0.000141      0

[5 rows x 108001 columns]
```

Gambar 4.61 5 Data Awal *Dataframe* Berlabel yang Telah Disatukan

Informasi pada data dapat diketahui dengan menggunakan kode perintah yang ditunjukkan pada gambar 4.62.

```
1.  df_eeg.info()
```

Gambar 4.62 Kode Informasi *Dataframe*

Tampilan informasi *dataframe* ditunjukkan pada gambar 4.63.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Columns: 108001 entries, 0 to label
dtypes: float64(108000), int64(1)
memory usage: 16.5 MB
```

Gambar 4.63 Informasi *Dataframe*

Setelah memastikan tidak ada data kosong pada *dataframe* proses selanjutnya adalah memisahkan data atribut yang digunakan sebagai variabel fitur dan data label sebagai variabel target. Kode perintah pembuatan variabel fitur (X) dan variabel target (y) ditunjukkan pada gambar 4.64.

```
1.  X = df_eeg.iloc[:,0:108000]
2.  y = df_eeg.iloc[:,108000]
```

Gambar 4.64 Variabel Fitur dan Variabel Target

a. Memisahkan (*Spliting*) *Data Training* dan *Data Testing*

Pemisahan data latih (*data training*) dan data uji (*data testing*) pada python dapat memanfaatkan fungsi "*train test split*" pada *library scikit learn*. Pada penelitian ini data

latih dan data uji di pisahkan dengan persentase sebanyak 70:30 yang berarti 70% data digunakan sebagai data latih dan 30% data digunakan sebagai data uji. Kode perintah untuk melakukan *spliting* data latih dan data uji ditunjukkan pada gambar 4.65.

```
1. from sklearn.model_selection import train_test_split
2. X_train, X_test, y_train, y_test = train_test_split(X, y,
       test_size=0.3, random_state=12)
```

Gambar 4.65 *Spliting Data Traning* dan *Data Testing*

b. Membuat Model *Machine Learning* dengan Algoritma *K-Nearest Neighbor* (K-NN)

Pada penelitian ini algoritma yang dipilih untuk pembuatan model *machine learning* adalah algoritma *k nearest neighbor* (K-NN). Seperti namanya, algoritma K-NN melakukan prediksi label kelas berdasarkan dengan menghitung kedekatan tetangga "*nearest neighbor*". Jumlah nilai K atau jumlah tetangga terdekat dapat menentukan hasil akhir nilai akurasi model *machine learning* dengan K-NN. Dengan demikian penentuan nilai K paling optimal sangat

penting untuk dilakukan pada algoritma K-NN. Penentuan nilai K paling optimal dapat dilakukan dengan melakukan perhitungan secara berulang (*looping*) dengan menguji nilai K yang berbeda pada algoritma K-NN. Kode perintah untuk melakukan perhitungan algoritma K-NN secara berulang ditunjukkan pada gambar 4.66 dimana pada penelitian ini uji coba perhitungan nilai K dilakukan dengan nilai K=1 hingga nilai K=10 kemudian dilakukan perhitungan untuk melihat nilai *error* pada tiap nilai K untuk mengetahui nilai K paling optimal dengan nilai error terendah.

```
1.  import numpy  as np
2.  from sklearn.neighbors import KNeighborsClassifier
3.  error_rating = []
4.  for k in range(1, 11):
        knn = KNeighborsClassifier(n_neighbors=k,
                metric='euclidean')
        knn.fit(X_train, y_train)
        y_pred = knn.predict(X_test)
        error_rating.append(np.mean(y_pred != y_test))
```

Gambar 4.66 *Looping* Algoritma K-NN

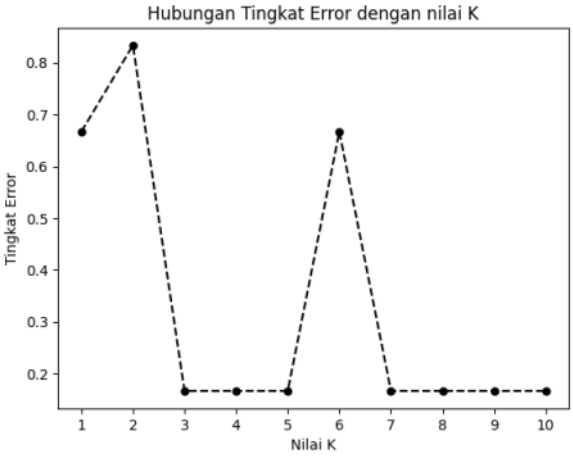
Visualisasi data dilakukan untuk mengetahui nilai K dengan jumlah tingkat nilai *error* terendah dengan memanfaatkan *library* matplotlib pada python. Kode

perintah untuk melakukan visualisasi hubungan tingkat *error* dengan nilai K ditunjukkan pada gambar 4.67.

```
1. import matplotlib.pyplot as plt
2. plt.plot(range(1,11), error_rating, marker="o",
        markerfacecolor="black", linestyle="dashed",
        color="black", markersize=5)
3. plt.title('Hubungan Tingkat Error dengan nilai K')
4. plt.xlabel('Nilai K')
5. plt.ylabel('Tingkat Error')
6. plt.xticks(range(1,11))
7. plt.show()
```

Gambar 4.67 Visualisasi Tingkat *Error* dengan Nilai K

Hasil visualisasi hubungan tingkat *error* dengan nilai K ditunjukkan pada gambar 4.68.



Gambar 4.68 Grafik Hubungan Tingkat *Error* dengan Nilai K

Grafik hubungan tingkat *error* dengan nilai K menunjukkan bahwa terdapat dua nilai K dengan tingkat *error* terendah yaitu nilai K=3,4,5 dan nilai K=7,8,9,10 dengan nilai *error* yang sama. Pada penelitian ini dipilih nilai K=10 sebagai nilai K paling optimal untuk pembuatan model *machine learning* dengan K-NN. Kode perintah pembuatan model *machine learning* algoritma K-NN dengan jumlah tetangga K=5 ditunjukkan pada gambar 4.69.

```
1. knn_optimal = KNeighborsClassifier(n_neighbors=10,
       metric='euclidean')
2. knn_optimal.fit(X_train, y_train)
3. y_pred_opt = knn_optimal.predict(X_test)
```

Gambar 4.69 Model *Machine Learning* K-NN dengan Nilai K Optimal

3. Perhitungan Nilai Akurasi Model

Nilai akurasi model pada penelitian ini dihitung menggunakan metode *convusion matrix*. Metode *convusion matrix* pada python dapat dihitung dengan memanfaatkan *library* pada *scikit learn*. Kode perintah untuk menghitung nilai akurasi model ditunjukkan pada gambar 4.70, sedangkan kode perintah

untuk menampilkan visualisasi *confusion matrix* ditunjukkan pada gambar 4.72.

```
1. from sklearn.metrics import accuracy_score
2. from sklearn.metrics import classification_report
3. acc = (accuracy_score(y_test, y_pred_opt)*100)
4. print("Nilai akurasi model klasifikasi k-nn : %.0f" % acc +"%")
5. print(classification_report(y_test, y_pred_opt))
```

Gambar 4.70 Kode Laporan Akurasi

Hasil laporan akurasi model ditunjukkan pada gambar 4.71 yang menunjukkan nilai akurasi model dengan algoritma K-NN sebesar 80%.

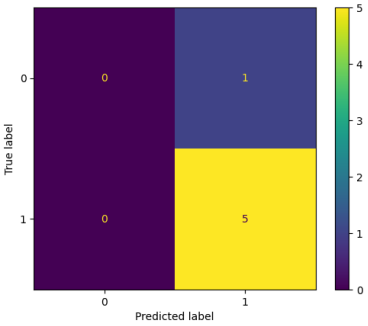
```
Nilai akurasi model klasifikasi k-nn : 83%
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.83      1.00      0.91         5

    accuracy                           0.83         6
   macro avg       0.42      0.50      0.45         6
weighted avg       0.69      0.83      0.76         6
```

Gambar 4.71 Laporan Akurasi Model

```
1. from sklearn.metrics import confusion_matrix,
       ConfusionMatrixDisplay
2. cm = confusion_matrix(y_test, y_pred_opt,
       labels=knn.classes_)
3. disp = ConfusionMatrixDisplay(confusion_matrix=cm,
       display_labels=knn.classes_)
4. disp.plot()
```

Gambar 4.72 Kode Visualisasi *Convusion Matrix*

Hasil visualisasi *convusion matrix* ditunjukkan pada gambar 4.73.



Gambar 4.73 Visualisasi *Convusion Matrix*

Hasil perhitungan *convusion matrix* dilakukan dengan melakukan perhitungan terhadap beberapa parameter antara lain:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{5}{6} = 0{,}83 \approx 83\%$$

$$Precission = \frac{TP}{FP+TP} = \frac{5}{6} = 0{,}83 \approx 83\%$$

$$Recall = \frac{TP}{FN+TP} = \frac{5}{5} = 1 \approx 100\%$$

$$F1Score = 2 \times \frac{Recall \times Precission}{Recall+Precission} = 2 \times \frac{1 \times 0{,}83}{1+0{,}83} = 0{,}91 \approx 91\%$$

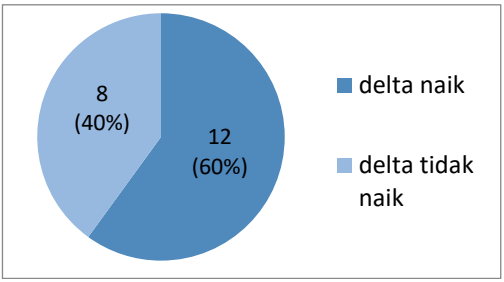
C. Pembahasan

Pembuktian terkait dengan mukjizat al-Qur'an terhadap potensi penyembuhan berdasarkan pengamatan gelombang delta otak dengan menggunakan stimulasi surah al-Fatihah menggunakan algoritma PSD dan K-NN merupakan tujuan utama penelitian ini. Pembuktian kemukjizatan al-Qur'an secara ilmiah pada

penelitian ini menggunakan data gelombang otak yang diakusisi dengan menggunakan salah satu instrumen medis yang umumnya digunakan dalam pemantauan elektrofisiologi dalam merekam aktifitas kelistrikan pada otak manusia yaitu elektroensefalografi (EEG). Perekaman menggunakan elektroensefalografi (EEG) menghasilkan hubungan secara tegak lurus antara waktu pada sumbu horizontal (X) dengan tegangan pada sumbu vertikal (Y) (Deu, 2019). Pengamatan dan analisa terhadap data hasil elektroensefalografi (EEG) berbasis waktu (*time based*) ini masih sangat susah untuk dilakukan. Perubahan maupun transformasi data EEG berbasis waktu (*time based*) ke dalam data berbasis frekuensi (*frequency based*) merupakan upaya yang dapat dilakukan dalam memudahkan proses analisa terhadap data EEG. Salah satu metode maupun algoritma untuk menghitung frekuensi data gelombang yang dapat digunakan adalah algoritma *power spectral density* (PSD).

Algoritma *power spectral density* (PSD) digunakan sebagai metode dalam melakukan

transformasi data mentah berbasis waktu (*time based*) ke dalam data berbasis frekuensi. Pada penelitian ini algoritma *power spectral density* (PSD) digunakan untuk menghitung frekuensi gelombang otak ke dalam 5 jenis frekuensi gelombang yaitu gelombang *delta* (0,5 – 4 Hz), gelombang *theta* (4 – 8 Hz), gelombang *alpha* (8 – 12 Hz), gelombang *beta* (12 – 25 Hz), dan gelombang *gamma* (25 – 40 Hz). Perhitungan dilakukan terhadap 20 data gelombang partisipan tanpa pemberian stimulus dan 20 data gelombang partisipan dengan pemberian stimulus murottal surah al-Fatihah ayat 1-7. Hasil perhitungan persentase tiap gelombang pada seluruh sampel partisipan ditunjukkan pada lampiran 7. Setelah seluruh hasil perhitungan didapatkan, perbandingan dilakukan terhadap kenaikan gelombang *delta* pada 20 partisipan sebelum dan setelah pemberian stimulus. Hasil perbandingan pada 20 partisipan menunjukkan terjadi kenaikan gelombang *delta* pada 12 partisipan (60%) dan tidak terjadi kenaikan gelombang *delta* pada 8 partisipan (40%) setelah pemberian stimulus surah al-Fatihah.



Gambar 4.74 *Pie Chart* Jumlah Partisipan Dengan Kenaikan Gelombang Delta

Gelombang *delta* merupakan gelombang yang dapat menjadi indikator dimana terjadi proses penyembuhan dalam tubuh manusia. Penyembuhan dan regenerasi terstimulus saat dalam gelombang delta (Garg & Keswani, 2017). Tubuh manusia mengalami proses pemulihan dengan meregenerasi sel (Filcek, 2023; Saminan, 2020).

Terjadinya kenaikan gelombang *delta* setelah pemberian stimulus surah al-Fatihah yang merupakan salah satu surah dalam al-Qur'an, membuktikan adanya potensi penyembuhan yang terjadi pada partisipan saat pemberian stimulus surah al-Fatihah. Potensi penyembuhan ini membuktikan pula kebenaran dan kemukjizatan yang tersurat dalam al-Qur'an firman Allah SWT surah al-Isra' ayat 82:

وَنُنَزِّلُ مِنَ الْقُرْآنِ مَا هُوَ شِفَاءٌ وَرَحْمَةٌ لِلْمُؤْمِنِينَ ۙ وَلَا يَزِيدُ الظَّالِمِينَ إِلَّا خَسَارًا

Artinya : *"Dan Kami turunkan dari al-Quran suatu yang menjadi penawar (penyembuh) dan rahmat bagi orang-orang yang beriman dan al-Quran itu tidaklah menambah kepada orang-orang yang zalim selain kerugian"*(al-Isra': 82).

Potensi penyembuhan yang termaktub dalam al-Qur'an didasarkan atas adanya kata *Syifa'* yang dapat dimaknai pula sebagai penyembuh. Pemaknaan *as-Syifa*' dapat dimaknai bahwa isi al-Qur'an secara maknawi, surat-surat, ayat-ayat memiliki potensi penyembuh (Latif, 2014). Al-Qur'an sebagai *as-Syifa'* atau penyembuh tidak secara spesifik menjelaskan jenis penyakit yang dapat disembuhkan dengan menggunakan al-Qur'an. Setidaknya terdapat dua kecenderungan ulama dalam memahami makna *as-Syifa',* kecenderungan pertama berpendapat bahwa al-Qur'an hanya berdampak pada penyembuhan pada penyakit hati, ruhaniah, atau kejiwaan, sedangkan kecenderungan kedua berpendapat dengan memperluas efek penyembuhannya tidak hanya pada ruhani namun mencakup pula pada penyakit fisik atau jasmani (Hasballah, 2013). Efek positif al-Qur'an sebagai

terapi terhadap suatu penyakit fisik didukung pula dengan penelitian yang dilakukan oleh Mardiati dkk (2018) dimana surah al-Fatihah dapat memberikan efek penurunan nilai median dan terdapat pengaruh yang baik setelah pemberian terapi al-Fatihah pada pasien skizofernia dengan halusinasi. Al-Qur'an bukan merupakan buku sains maupun buku kedokteran, namun secara langsung al-Qur'an menjelaskan bahwa al-Qur'an dapat menjadi "penyembuh" yang oleh kaum Muslim diartikan bahwa isi kandungannya dapat membawa manusia pada kesehatan spiritual, psikologis, dan fisik. Terdapat empat cara untuk mendapat efek penyembuhan dengan menggunakan ayat-ayat al-Qur'an yaitu dengan membaca al-Qur'an, menyimak dan mendengarkan al-Qur'an, dengan *taddabur* atau menghayati makna isi dan kandungan al-Qur'an, serta yang terakhir adalah dengan mengamalkan ajaran al-Qur'an (Hasballah, 2013).

Al-Qur'an dapat digunakan pula sebagai terapi dalam mencegah maupun menyembuhkan penyakit seperti yang disampaikan dalam penelitian oleh Sukmal, Syamsuwir, dan Satriadi (2019) yang

membagi terapi menjadi dua metode terapi antara lain:

a. Terapi Neurofisiologi Al-Qur'an

   Terapi neurofisiologi al-Qur'an adalah terapi dengan memperdengarkan ayat-ayat al-Qur'an sebagai salah satu media dalam melakukan pengobatan. Pengaruh neurofisiologi al-Qur'an dapat dirasakan meskipun tanpa mengetahui makna ayat maupun surat yang dibacakan, namun efeknya akan lebih maksimal apabila disertai dengan pemahaman maknanya (Ali, 2015).

b. Relaksasi Transedensi

   Metode relaksasi transedensi al-Qur'an merupakan metode relaksasi dengan membaca al-Qur'an. Manfaat metode ini dapat mengurangi tingkat stress dan mendatangkan ketenangan jiwa (Sukmal dkk., 2019).

Pembuktian kemukjizatan al-Qur'an terhadap potensi penyembuhan telah dibahas dalam penelitian ini dengan melakukan perbandingan hasil fluktuasi persentase gelombang *delta* pada data gelombang otak sebelum dan setelah pemberian stimulus surah al-Fatihah. Persentase gelombang ini

didapatkan dengan melakukan ekstraksi data berbasis waktu (*time based*) menjadi data berbasis frekuensi (*frequency based*) dengan menggunakan algoritma *power spectral density* (PSD) metode *welch*. Keberhasilan metode PSD dalam mengekstrasi data gelombang otak sesuai pula dengan penelitian yang telah dilakukan oleh Husain dan Aji (2019) dimana data gelombang otak diekstraksi dengan menggunakan PSD berbasis metode *welch* kemudian dilakukan perhitungan akurasi model MLP pada data gelombang otak dan menghasilkan akurasi sebesar 99.68%.

Upaya pembuatan model *machine learning* hingga penghitungan akurasi model algoritma *machine learning* juga dilakukan dalam penelitian ini dengan menggunakan model algoritma *machine learning k-nearest neighbor* (K-NN). Penggunaan algoritma K-NN pernah dilakukan sebelumnya pada data gelombang otak dan menuai keberhasilan dengan tingkat akurasi model yang cukup baik pula yang telah dibandingkan dengan algoritma lain, penelitian ini dilakukan oleh Almahdi dkk (2021) yang melakukan analisa terhadap sinyal EEG untuk

deteksi penyakit epilepsi dan menghasilkan akurasi dengan model SVM sebesar 72,7% dan KNN sebesar 81.8%.

Pembuatan model *machine learning* dengan algoritma *k-nearest neighbor* (KNN) pada penelitian ini dilakukan terhadap data dengan dua tingkat label berdasarkan pengamatan kenaikan gelombang delta otak yaitu "terjadi kenaikan" dan "tidak terjadi kenaikan" setelah pemberian stimulus surah al-Fatihah.

Pembuatan model *machine learning* dengan algoritma *k-nearest neighbor* (K-NN) pada penelitian ini dilakukan dengan membagi data latih (*data training*) sebanyak 70% data dan data uji (*data testing*) sebanyak 30% data dari total keseluruhan data.

Hasil perhitungan nilai akurasi model K-NN dengan nilai K=10 mengunakan *convusion matix* pada penelitian ini menunjukan nilai *accuracy* = 83%, *precision* sebesar 83%, *recall* sebesar 100%, dan *f1-score* sebesar 91%. Upaya pembandingan dengan algoritma lainnya juga dilakukan pada penelitian ini seperti yang ditunjukkan pada lampiran

8. Hasil perbandingan dengan algoritma model *machine learning* lainnya menunjukkan bahwa model *machine learning* dengan algoritma *k-nearest neighbor* (K-NN) menempati posisi dengan nilai akurasi tertinggi sehingga penggunaan algoritma K-NN pada studi kasus pada penelitian ini sudah cukup tepat dengan menghasilkan nilai akurasi terbaik.

BAB V
PENUTUP

A. Simpulan

Berdasarkan analisa perbandingan kenaikan gelombang *delta* otak untuk membuktikan potensi penyembuhan dalam al-Qur'an menggunakan algoritma PSD dan K-NN, maka dapat disimpulkan bahwa:

1. Potensi penyembuhan dalam al-Qur'an dapat dibuktikan dengan melakukan perbandingan hasil transformasi gelombang berbasis waktu ke dalam gelombang berbasis frekuensi dengan algoritma PSD terhadap kenaikan gelombang *delta* sebelum dan setelah pemberian stimulus surah al-Fatihah saat kondisi tenang. Hasil perbandingan menyatakan dari total 20 partisipan, 12 partisipan (60%) mengalami kenaikan gelombang *delta* dan 8 partisipan (40%) tidak mengalami kenaikan gelombang delta.
2. Perhitungan model *machine learning* dengan algoritma K-NN menghasilkan nilai *accuracy* =

83%, *precision* sebesar 83%, *recall* sebesar 100%, dan *f1-score* sebesar 91%.

B. Saran

Berdasarkan hasil penelitian mengenai pembuktian mukjizat al-Qur'an ini maka saran yang dapat disampaikan untuk kemajuan penelitian selanjutnya adalah sebagai berikut:

1. Tidak semua sampel mengalami kenaikan geombang, oleh karena itu penelitian lebih lanjut mengenai penyebab terjadinya kenaikan dan penurunan gelombang terhadap stimulus surah al-Qur'an masih harus mendapatkan pengkajian yang lebih mendalam.
2. Penambahan dan peningkatan kualitas data perlu untuk dilakukan untuk mendapatkan nilai akurasi yang lebih baik lagi.

DAFTAR PUSTAKA

Aji, N. B., & Tjandrasa, H. (2017). Klasifikasi EEG Epilepsi Menggunakan Singular Spectrum Analysis, Power Spectral Density dan Convolutional Neural Network. *Jurnal Ilmiah Teknologi Informasi*, 185–194. https://doi.org/dx.doi.org/10.12962/j24068535.v15i2.a662

Ali, S. (2015). Pengobatan Alternatif dalam Perspektif Hukum Islam. *Al-'Adalah*, *12*(2), 867–890. https://doi.org/doi.org/10.24042/adalah.v12i2.218

Al-Khalidi, S. A. F. (1994). *Mafatih Li al-Ta'amul ma'a al-Qur'an*. Maktabah al-Manar.

Almahdi, A. J., Yaseen, A. J., & Dakhil, A. F. (2021). EEG Signal Analysis for Epileptic Seizure Detection Using DWT Method with SVM and KNN Classifiers. *Iraqi Journal of Science*, *2*, 54–62. https://doi.org/10.24996/ijs.2021.SI.2.6

Ameera, A., Saidatul, A., & Ibrahim, Z. (2019). Analysis of EEG Spectrum Bands Using Power Spectral Density for Pleasure and Displeasure State. *IOP Conference Series: Materials Science and Engineering*, *557*(1). https://doi.org/10.1088/1757-899X/557/1/012030

Anggara, R., & Rahayu, Y. (2020). Sistem Electroencephalogram (EEG) Untuk Analisis Sinyal Gelombang Otak Pada Pasien Depresi. *Jom FTEKNIK*, *7*(1), 1–6.

Antonisfia, Y., & Wiryadinata, R. (2008). Ekstraksi Ciri pada Isyarat Suara Jantung Menggunakan Power Spectral Density Berbasis Metode Welch. *Media Informatika*, *6*(1), 71–84.

Arkoun, M. (1998). *Kajian Kontemporer al-Qur'an*. Penerbit Pustaka.

Bermudez, D., Steyrl, D., & Muller-Putz, G. R. (2020). Implementation of Machine Learning Algorithm to

Exploit Information from Multimodal fMRI/EEG Fused Image Data. *ResearchGate*. http://dx.doi.org/10.13140/RG.2.2.28622.82240

Bintoro, A. C. (2012). Pemeriksaan EEG untuk Diagnosis dan Monitoring pada Kelainan Neurologi. *Medica Hospitalia*, *1*(1), 64–70.

Cohen, L., Manion, L., & Morrison, K. (2007). *Research Method in Education*. Routledge.

Deu, A. (2019). *Aktivitas Gelombang Alfa Otak saat Diperdengarkan Surah Al-Fatihah dengan Mengetahui Maknanya*. UIN Syarif Hidayatullah.

Dewi, D. A. P., & Djamal, E. C. (2015). Klasifikasi Sinyal Elektroensephalogram terhadap Kewaspadaan Menggunakan Support Vector Machine. *Prosiding SNIJA*, 242–246.

Dimurtadha, Melinda, Elizar, & Meutia, E. D. (2019). *Analisis Filter Finite Impulse Response (FIR) pada Sinyal Electroensephalogram (EEG)*. Seminar Nasional dan Expo Teknik Elektro.

Filcek, M. (2023). Vinci Power Nap® Synchronization Technology—Neuroarchitecture for better sleep, adding Energy for Life, Reducing Stress and Jet Lag, Improving Wellbeing and Body-Mind Regeneration. Helpful for Leaders, Doctors, Soldiers, Children, Drivers, Pilots, Astronauts, People Traveling On Earth And Beyond. *Journal of Biotechnology and Biomedicine*, *6*(2), 105–128.

Firstanto, M. A., Wirawan, & Widjiati, E. (2013). Kombinasi Metode Independent Component Analysis (ICA) dengan Beamforming untuk Pemisahan Sinyal Akustik Bawah Air. *JURNAL TEKNIK POMITS*, *2*(2), 300–305. https://doi.org/doaj.org/toc/2337-3539

Garg, S., & Keswani, N. (2017). Brainwaves Signaling System during Accident. *International Research Journal of Engineering and Technology (IRJET)*, *4*(1), 663–668.

Harsoyo, A. P. (2018). Mukjizat Numerikal Al-Qur'an. *OSF Preprints*. https://doi.org/10.31219/osf.io/2aqcf

Hasballah, Z. (2013). Al-Qur'an sebagai Syifa' (Penyembuhan). *Ibnu Nafis*, *2*(1), 45–53.

Hindarto, & Muntasa, A. (2018). Ekstraksi Sinyal EEG Menggunakan Koefisien Subband Transformasi Wavelet Diskrit. *Jurnal Saintek*, *15*(2), 61–65.

Husain, N. P., & Aji, N. B. (2019). Klasifikasi Sinyal EEG Dengan Power Spectra Density Berbasis Metode Welch Dan MLP Backpropagation. *Jurnal ELTIKOM*, *3*(1), 17–25. https://doi.org/10.31961/eltikom.v3i1.99

Latif, U. (2014). Al-Qur'an Sebagai Sumber Rahmat dan Obat Penawar (Syifa') bagi Manusia. *Jurnal Al-Bayan*, *21*(30), 77–88.

Mardiati, S., Elita, V., & Sabrian, F. (2018). Pengaruh Terapi Psikoreligius: Membaca Al-Fatihah Terhadap Skor Halusinasi Pasien Skizofrenia. *Jurnal Ners Indonesia*, *8*(2), 110–123.

Moore, K., Dalley, A., Agur, A., & Moore, M. (2013). *Anatomi Berorientasi Klinis* (5 ed.). Erlangga.

Riwinoto. (2014). Penemuan Sinyal Asli dengan Metode Independent Component Analysis pada Sinyal Tercampur Tunggal. *Jurnal Integrasi*, *6*(1), 77–83.

Rozi, M. F. (2019). *Aktivitas Gelombang Alfa pada Otak Manusia Saat Mendengar Murrotal Surah Al Quran Tentang Hari Kiamat*. UIN Syarif Hidayatullah.

Saminan, N. F. (2020). Frekuensi Gelombang Otak dalam Menangkap Ilmu Imajinasi dan Realita (Berdasarkan Ontologi). *Jurnal Filsafat Indonesia*, *3*(2), 40–47.

Saputro, I. W., & Sari, B. W. (2019). Uji Performa Algoritma Naïve Bayes untuk Prediksi Masa Studi Mahasiswa. *Citec Journal*, *6*(1), 1–11.

Sari, D. R., & Asiva, Z. (2019). Pengaruh Murottal al-Qur'an Surah al-Fatihah untuk Menurunkan Tingkat Insomnia pada Mahasiswa. *Jurnal Psikologi Islam*, *6*(2), 23–36.

Sarno, R., Sabilla, S. I., Malikhah, Purbawa, D. P., & Ardani, M. S. H. (2022). *Machine Learning dan Deep Learning—Konsep dan Pemrograman Python* (I). Andi.

Sherwood, L. (2014). *Fisiologi Manusia dari Sel ke Sistem* (8 ed.). EGC.

Shihab, M. Q. (2013). *Mukjizat Al-Qur'an: Ditinjau dari Aspek Kebahasaan, Isyarat Ilmiah, dan Pemberitaan Gaib*. Mizan.

Sofiani, R. N. (2022). Klasifikasi Jenis Emosi Berdasarkan Gelombang Otak Menggunakan Dimensi Higuchi dengan K-Nearest Neighbor. *Jurnal Ilmiah Matematika*, *10*(1), 150–160.

Sugiyono. (2018). *Metode Penelitian Kuantitatif, Kualitatif, dan R&D*. Alfabeta.

Sukmal, M., Syamsuwir, & Satriadi, I. (2019). Syifa' dalam Perspektif Alquran. *Istinarah : Riset Keagamaan, Sosial, dan Budaya*, *1*(2), 75–87.

Syarif, A. N. R. (2018). *Perbandingan Aktivitas Gelombang Alfa Elektroensefalografi (EEG) Otak Sebelum dan Setelah Perlakuan saat Diperdengarkan Murottal Al Quran Surah Al Insyiqaq pada Mahasiswa Kedokteran UIN Syarif Hidayatullah Jakarta*. UIN Syarif Hidayatullah.

Wahyono, T. (2018). *Fundamental of Python for Machine Learning* (1 ed.). Penerbit Gava Media.

Wardhana, W. A. (2016). *Hadiah Nobel dan Sains Modern dalam Al-Qur'an*. Pustaka Pelajar.

Wati, E. K. (2018). *Teori Pengolahan Sinyal Digital*. LP_UNAS.

Yean, C. W., Khairunizam, W., Omar, M. I., Murugappan, M., Zheng, B. S., Bakar, S. A., Razlan, Z. M., & Ibrahim, Z. (2018). Analysis of The Distance Metics of KNN Classifier for EEG Signal in Stroke Patients. *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)*, 1–4. https://doi.org/doi: 10.1109/ICASSDA.2018.8477601

Yulianto, E., Susanto, A., Widodo, T. S., & Wibowo, S. (2013). Spektrum Frekuensi Sinyal EEG Terhadap Pergerakan Motorik dan Imajinasi Pergerakan Motorik. *Forum Teknik*, *35*(1), 21–32.

Zubairin, A. (2020). Upaya Pembuktian Otentisitas Al-Qur'an Melalui Pendekatan Sastra (Tafsir Adabiy). *Jurnal Asy-Syukriyyah*, *21*(1), 34–48.

LAMPIRAN

Lampiran 1. Lembar Pengesahan Proposal

LEMBAR PENGESAHAN PROPOSAL SKRIPSI

Judul Skripsi	:Mukjizat Al-Qur'an Terhadap Potensi Penyembuhan Berdasarkan Pengamatan Gelombang Delta Otak Dengan Stimulasi Al-Fatihah Dalam Kondisi Tenang Menggunakan Algoritma PSD dan K-NN
Nama	: Thooriq Nur Ali
NIM	: 1908096004
Program Studi	: Teknologi Informasi

Telah diujikan dalam sidang komprehensif oleh Dewan Penguji Fakultas Sains dan Teknologi UIN Walisongo Semarang dan dapat diterima sebagai salah satu syarat memperoleh gelar sarjana dalam program studi Teknologi Informasi.

Semarang, 19 September 2023

DEWAN PENGUJI

Penguji I,

Dr. Masy Ari Ulinuha, M.T
NIP. 198108122011011007

Penguji II,

Mokhamad Iklil Mustofa, M.Kom.
NIP. 198808072019031010

Penguji III,

Hery Mustofa, M.Kom.
NIP. 197703172019031007

Penguji IV,

Adzhal Arwani Mahfudh, M.Kom.
NIP. 199107032019031006

Lampiran 2. Lembar Bimbingan Tugas Akhir

LEMBAR BIMBINGAN TUGAS AKHIR

Nama : Thooriq Nur Ali

NIM : 1908096004

Judul : Mukjizat Al-Qur'an Terhadap Potensi Penyembuhan Berdasarkan Pengamatan Geombang Otak dengan Stimulasi Al-Fatihah dalam Kondisi Tenang Menggunakan Algoritma PSD dan K-NN

Dosen Pembimbing I : Nur Cahyo Hendro Wibowo, S.T., M.Kom

No	Tanggal	Rincian Kegiatan Bimbingan	Tanda Tangan Pembimibing
1	05-07-2023	Bimbingan BAB I, BAB II, dan BAB III	
2	11-07-2023	Bimbingan BAB I, BAB II, dan BAB III	
3	21-08-2023	Bimbingan BAB IV, dan BAB V	

LEMBAR BIMBINGAN TUGAS AKHIR

Nama : Thooriq Nur Ali

NIM : 1908096004

Judul : Mukjizat Al-Qur'an Terhadap Potensi Penyembuhan Berdasarkan Pengamatan Geombang Otak dengan Stimulasi Al-Fatihah dalam Kondisi Tenang Menggunakan Algoritma PSD dan K-NN

Dosen Pembimbing II : Mokhamad Iklil Mustofa, M.Kom

No	Tanggal	Rincian Kegiatan Bimbingan	Tanda Tangan Pembimbing
1	05-07-2023	Bimbingan BAB 1, BAB II, dan BAB III	
2	10-07-2023	Bimbingan BAB 1, BAB II, dan BAB III	
3	19-08-2023	Bimbingan BAB 1V, dan BAB V	

Lampiran 3. Kode Program Proses ICA untuk Satu Partisipan

```
[ ]: #menginstall paket pustaka (library package) mne
     !pip install mne
```

```
[ ]: #mengimpor data dari penyimpanan google drive
     from google.colab import drive
     drive.mount('/content/drive')
```

```
[ ]: #nama file
     name_file = "/content/drive/MyDrive/eeg_data_stim_alfatihah/raw_data_csv/
      ↪sheva_stim_data.csv"
     name_file_export = "/content/drive/MyDrive/eeg_data_stim_alfatihah/
      ↪reconst_data_edf/sheva_stim_data_reconst.edf"
```

```
[ ]: #mengimpor paket library
     import pandas as pd
     import mne
     import matplotlib.pyplot as plt
     from mne.preprocessing import ICA

     #konversi csv ke dataframe pandas
     df = pd.read_csv(name_file)
     df.head()
```

```
[ ]: #menghapus kolom (time)
     df.drop("Time", axis=1, inplace=True)
```

```
[ ]: #definisi informasi EEG (raw data)
     sfreq = 100
     ch_types = ["eeg"]*df.shape[1]
     ch_names = ["Fp1","Fp2","F3","F4","C3","C4","P3","P4","O1","O2","F7","F8","T3",
                 "T4","T5","T6","A1","A2"]
     montage = mne.channels.make_standard_montage("standard_1020")
     info = mne.create_info(ch_names=ch_names, sfreq=sfreq, ch_types=ch_types)
     samples = df.T*1e-6
     raw_data = mne.io.RawArray(samples, info)
     raw_data.set_montage(montage=montage)
```

[ ]:
```
#Melakukan filter bandpass
raw_data=raw_data.copy().filter(0.5, 40., fir_design='firwin')
```

[ ]:
```
#filtrasi dengan menggunakan ICA
ica_eeg = ICA(n_components=15, random_state=0) #setup ICA
ica_eeg.fit(raw_data) #jalankan ICA
```

[ ]:
```
#visualisasi
ica_eeg.plot_sources(raw_data) #plot time series
```

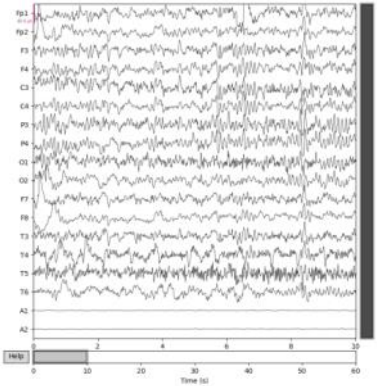
[ ]:
```
raw_data.load_data()
ica_eeg.exclude = [13]
reconst_data = raw_data.copy()
ica_eeg.apply(reconst_data)

#raw data
raw_data.plot(duration=10, show=False)

#rekonstruksi data setelah mengilangkan artefak
reconst_data.plot(duration=10, show=False)
plt.show()
```

[ ]:
```
#menginstal paket pustaka (library package)
!pip install EDFlib-Python
```

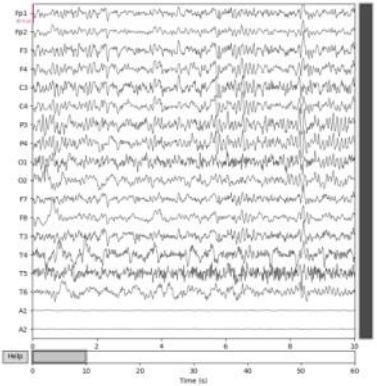
[ ]:
```
#ekspor
export = mne.export.export_raw(name_file_export, reconst_data, overwrite=True)
export
```

Lampiran 4. Visualisasi Pembersihan Data dengan ICA pada Tiap Partisipan

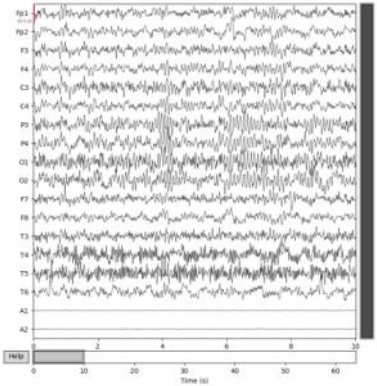
Data Partisipan 1 (non-stim) Sebelum ICA



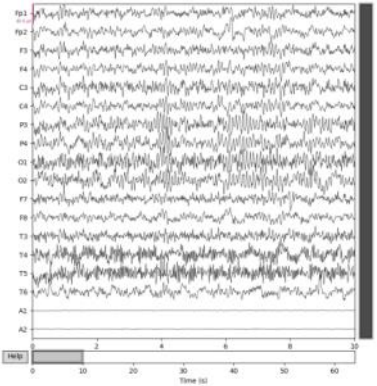
Data Partisipan 1 (non-stim) Setelah ICA



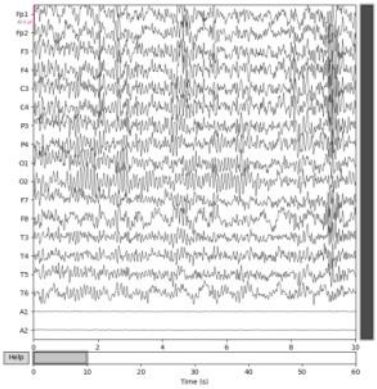
Data Partisipan 1 (stimulus) Sebelum ICA



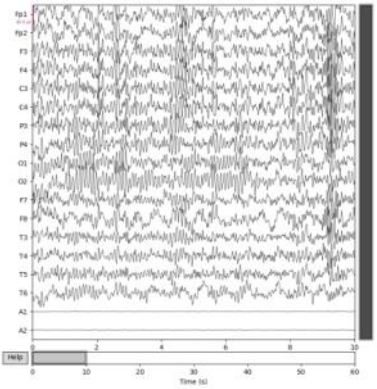
Data Partisipan 1 (stimulus) Setelah ICA



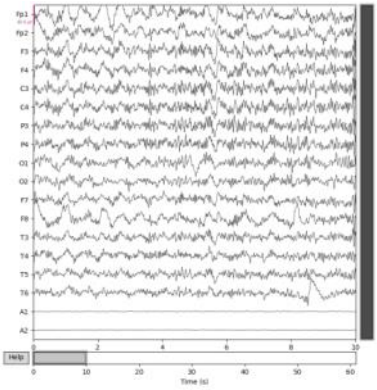
Data Partisipan 2 (non-stim)
Sebelum ICA



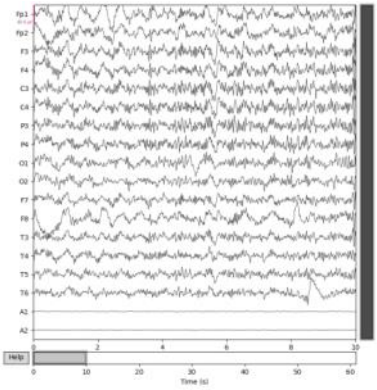
Data Partisipan 2 (non-stim)
Setelah ICA



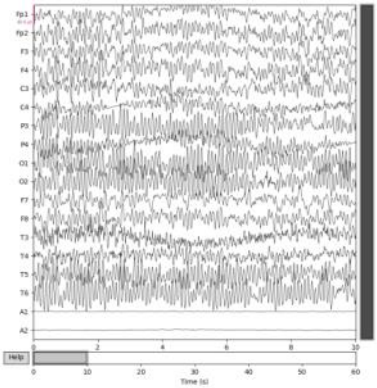
Data Partisipan 2 (stimulus)
Sebelum ICA



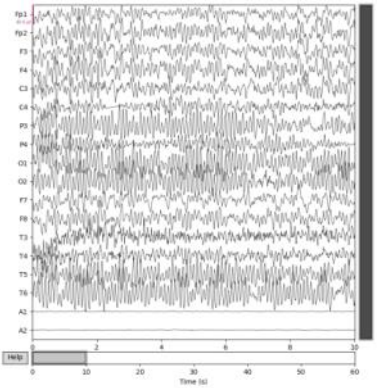
Data Partisipan 2 (stimulus)
Setelah ICA



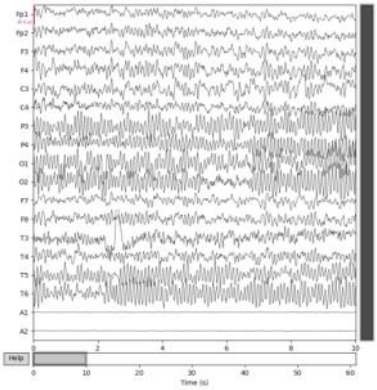
Data Partisipan 3 (non-stim)
Sebelum ICA



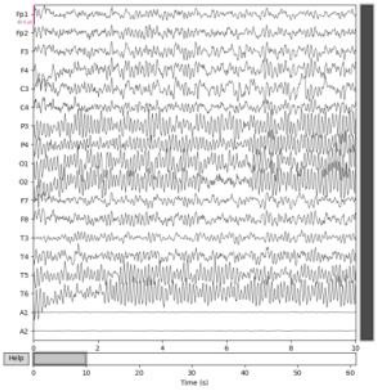
Data Partisipan 3 (non-stim)
Setelah ICA



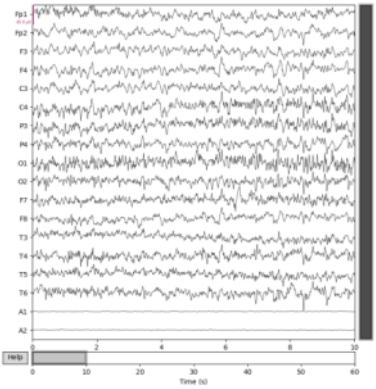
Data Partisipan 3 (stimulus)
Sebelum ICA



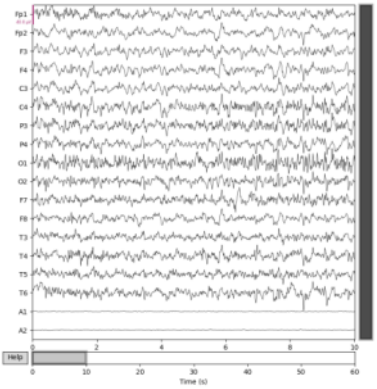
Data Partisipan 3 (stimulus)
Setelah ICA



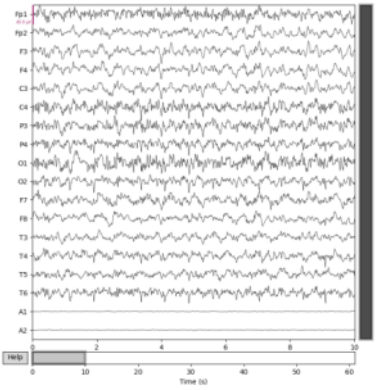
Data Partisipan 4 (non-stim)
Sebelum ICA



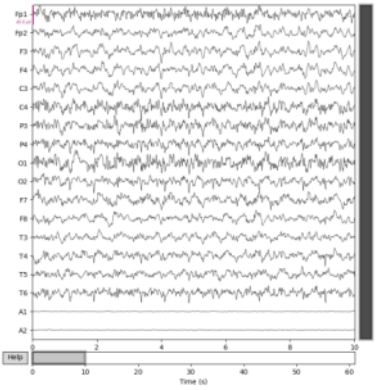
Data Partisipan 4 (non-stim)
Setelah ICA



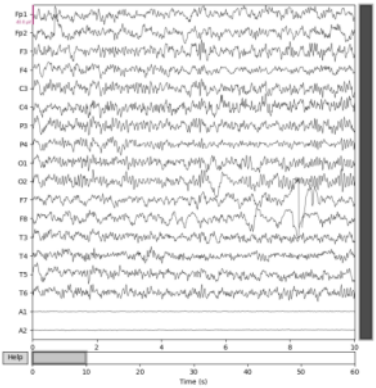
Data Partisipan 4 (stimulus)
Sebelum ICA



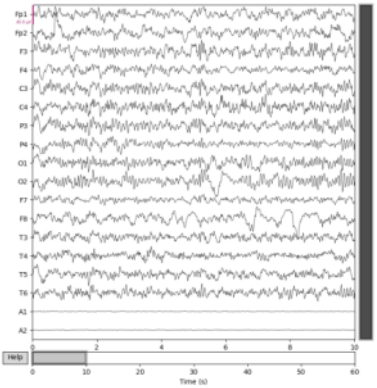
Data Partisipan 4 (stimulus)
Setelah ICA



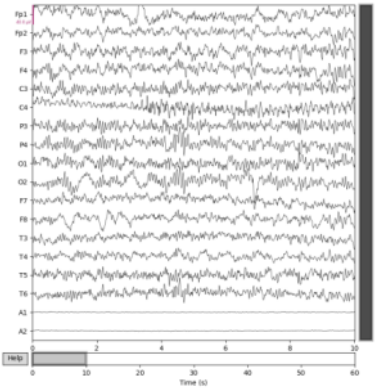
Data Partisipan 5 (non-stim)
Sebelum ICA



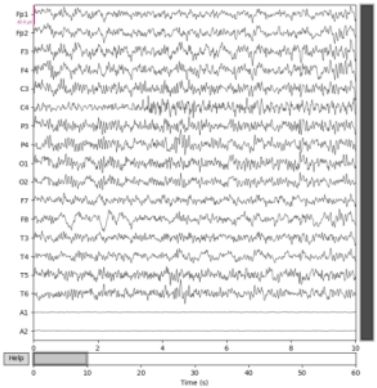
Data Partisipan 5 (non-stim)
Setelah ICA



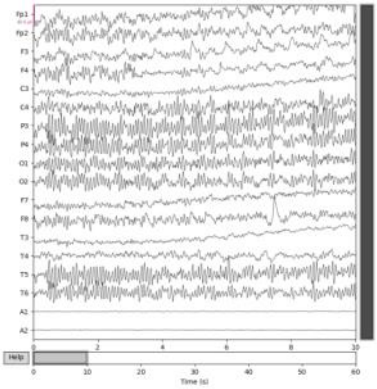
Data Partisipan 5 (stimulus)
Sebelum ICA



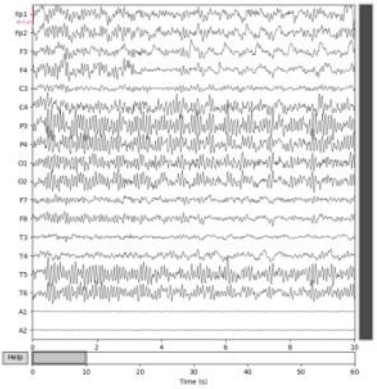
Data Partisipan 5 (stimulus)
Setelah ICA



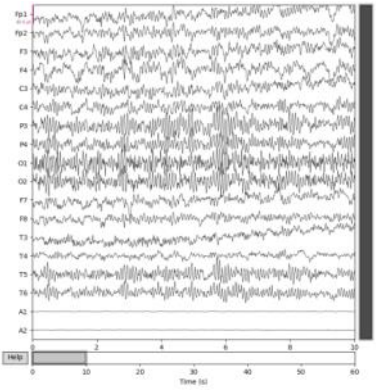
Data Partisipan 6 (non-stim)
Sebelum ICA



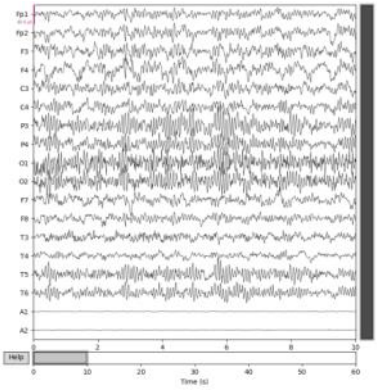
Data Partisipan 6 (non-stim)
Setelah ICA



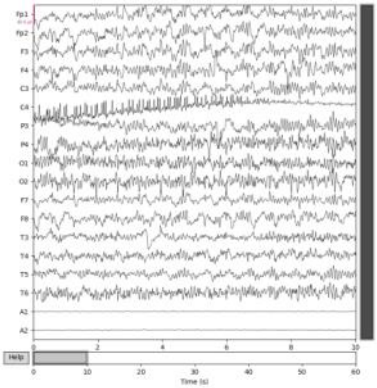
Data Partisipan 6 (stimulus)
Sebelum ICA



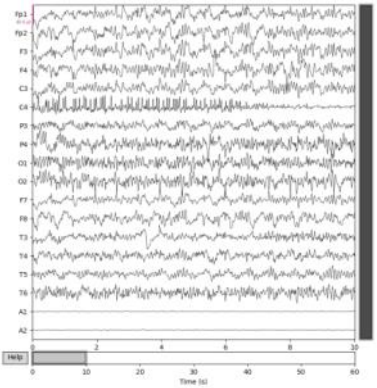
Data Partisipan 6 (stimulus)
Setelah ICA



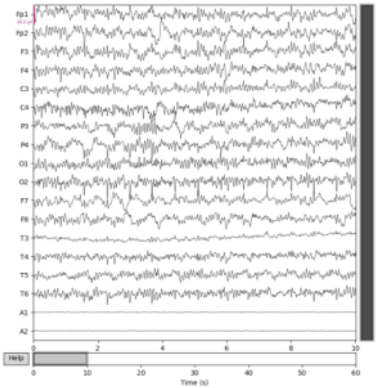
Data Partisipan 7 (non-stim)
Sebelum ICA



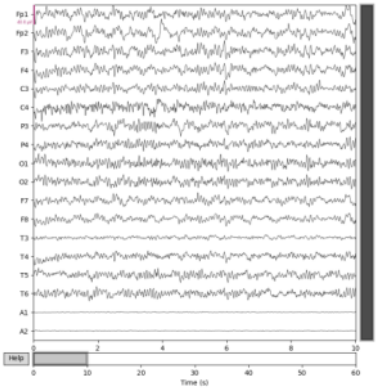
Data Partisipan 7 (non-stim)
Setelah ICA



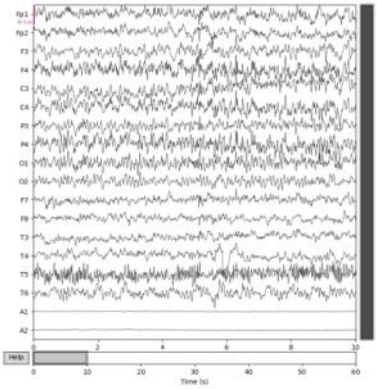
Data Partisipan 7 (stimulus)
Sebelum ICA



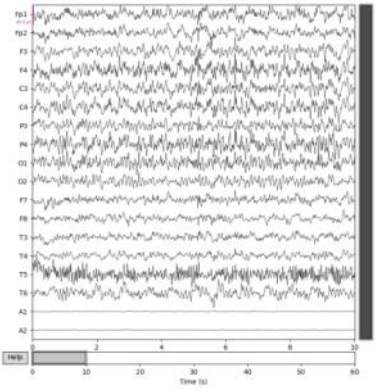
Data Partisipan 7 (stimulus)
Setelah ICA



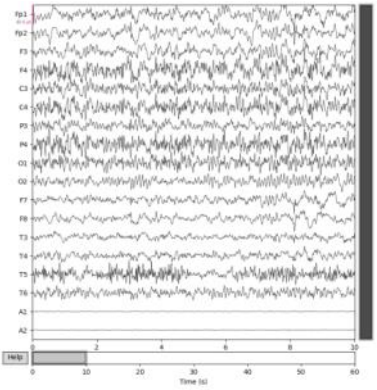
Data Partisipan 8 (non-stim)
Sebelum ICA



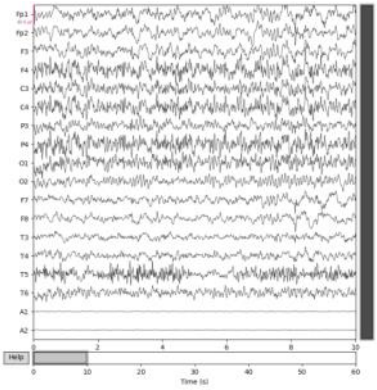
Data Partisipan 8 (non-stim)
Setelah ICA



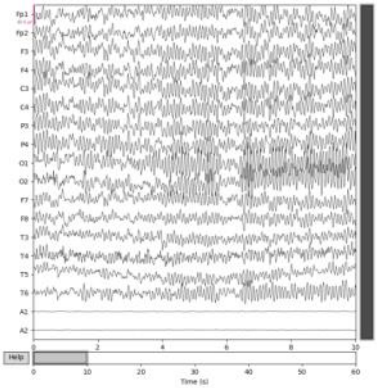
Data Partisipan 8 (stimulus)
Sebelum ICA



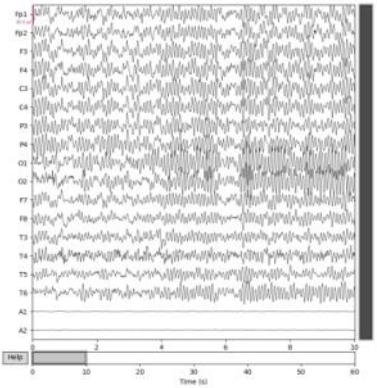
Data Partisipan 8 (stimulus)
Setelah ICA



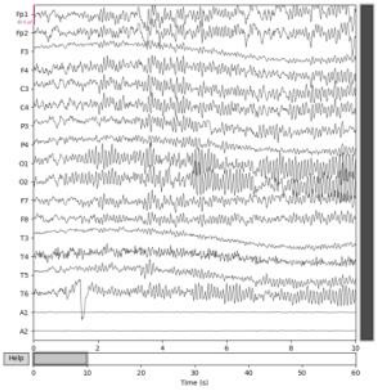
Data Partisipan 9 (non-stim)
Sebelum ICA



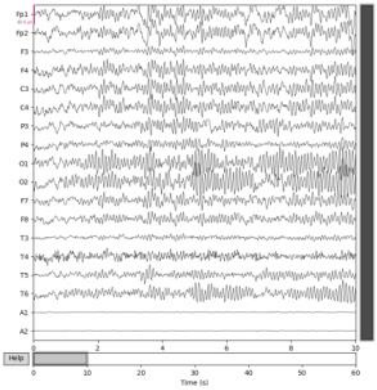
Data Partisipan 9 (non-stim)
Setelah ICA



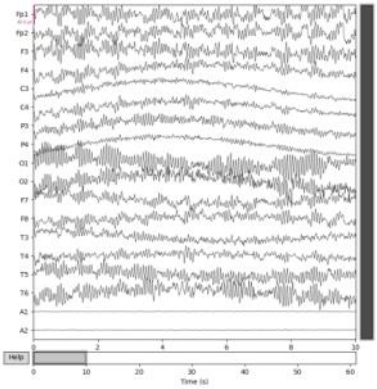
Data Partisipan 9 (stimulus)
Sebelum ICA



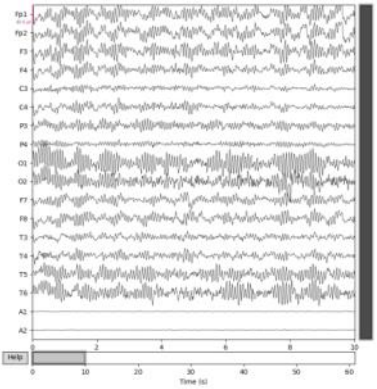
Data Partisipan 9 (stimulus)
Setelah ICA



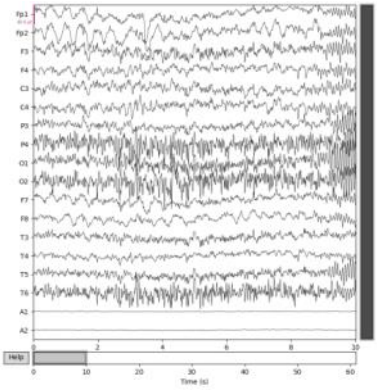
Data Partisipan 10 (non-stim)
Sebelum ICA



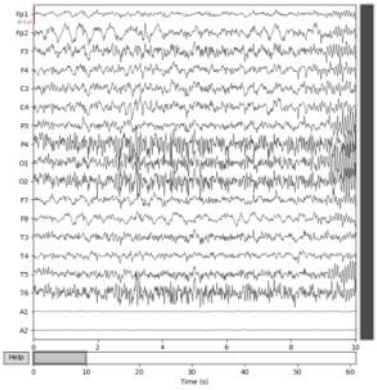
Data Partisipan 10 (non-stim)
Setelah ICA



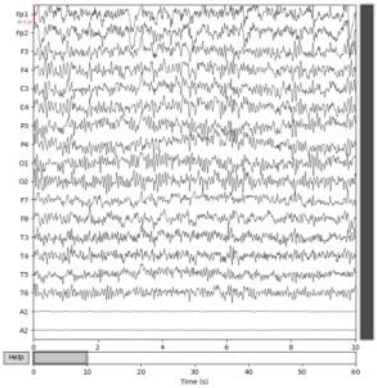
Data Partisipan 10 (stimulus)
Sebelum ICA



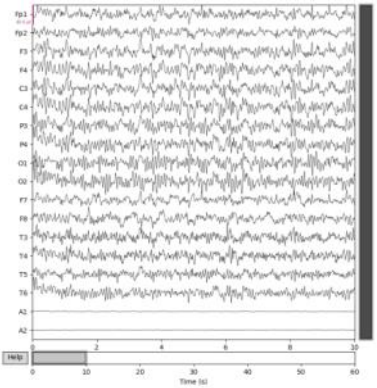
Data Partisipan 10 (stimulus)
Setelah ICA



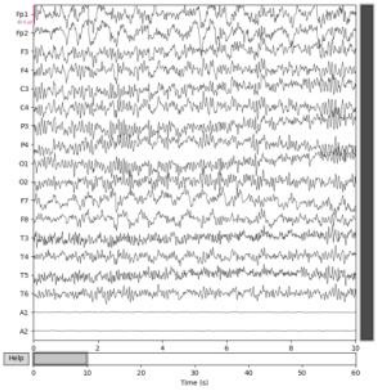
Data Partisipan 11 (non-stim)
Sebelum ICA



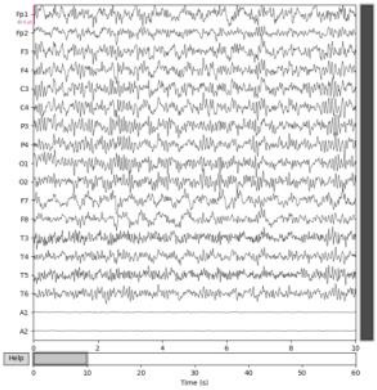
Data Partisipan 11 (non-stim)
Setelah ICA



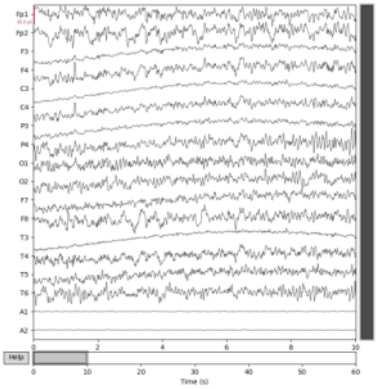
Data Partisipan 11 (stimulus)
Sebelum ICA



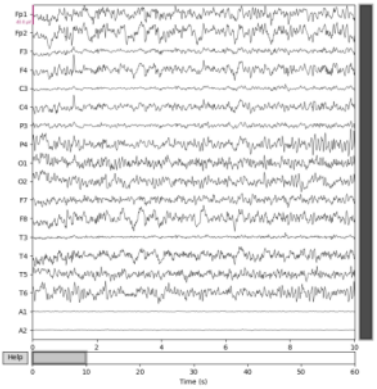
Data Partisipan 11 (stimulus)
Setelah ICA



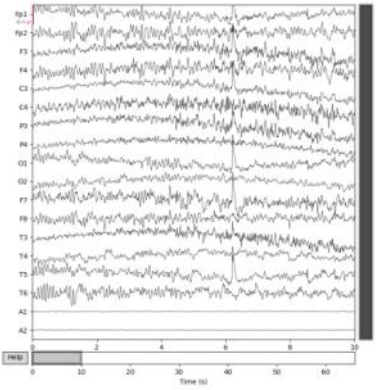
Data Partisipan 12 (non-stim)
Sebelum ICA



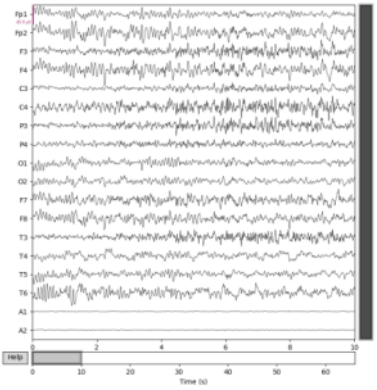
Data Partisipan 12 (non-stim)
Setelah ICA



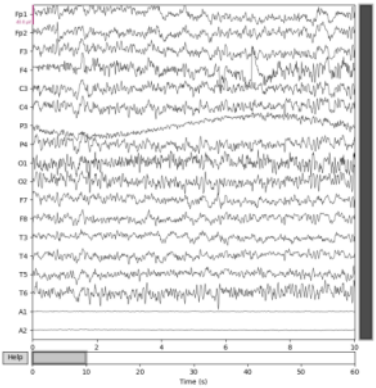
Data Partisipan 12 (stimulus)
Sebelum ICA



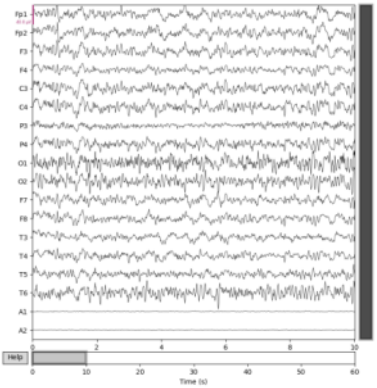
Data Partisipan 12 (stimulus)
Setelah ICA



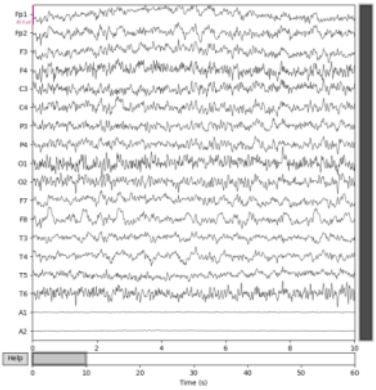
Data Partisipan 13 (non-stim)
Sebelum ICA



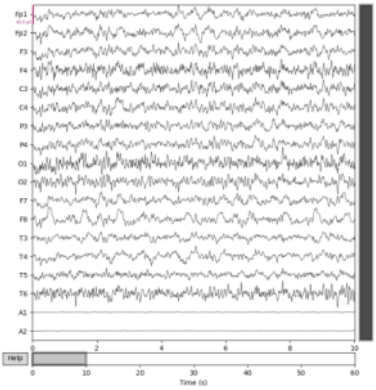
Data Partisipan 13 (non-stim)
Setelah ICA



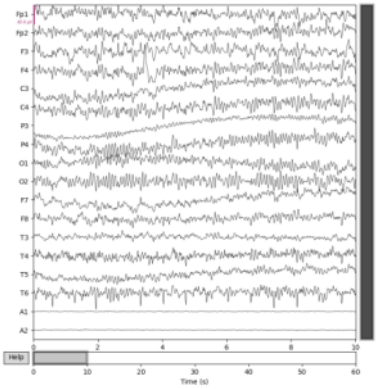
Data Partisipan 13 (stimulus)
Sebelum ICA



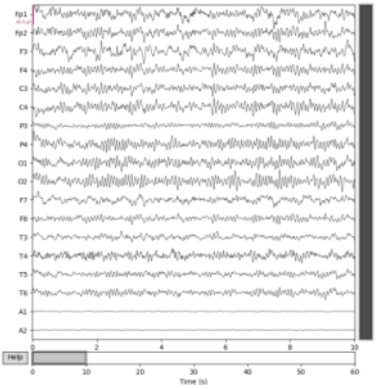
Data Partisipan 13 (stimulus)
Setelah ICA



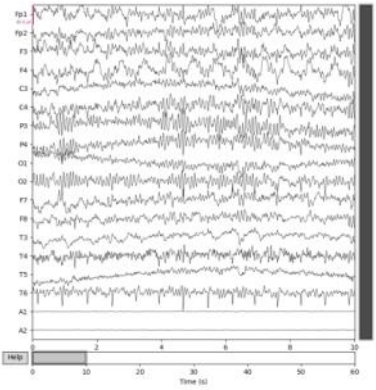
Data Partisipan 14 (non-stim)
Sebelum ICA



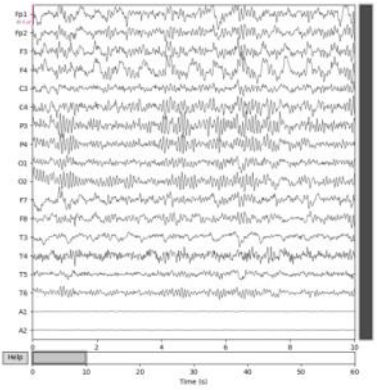
Data Partisipan 14 (non-stim)
Setelah ICA



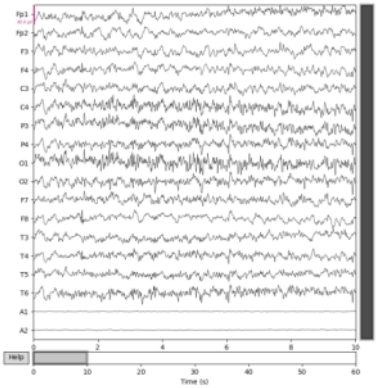
Data Partisipan 14 (stimulus)
Sebelum ICA



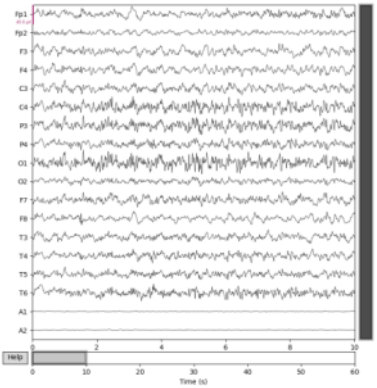
Data Partisipan 14 (stimulus)
Setelah ICA



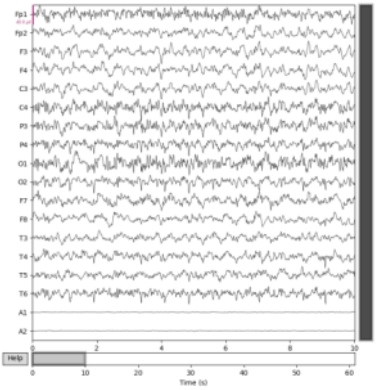
Data Partisipan 15 (non-stim)
Sebelum ICA



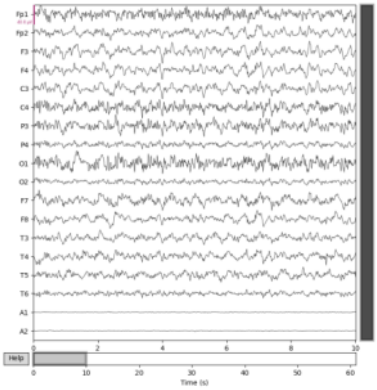
Data Partisipan 15 (non-stim)
Setelah ICA



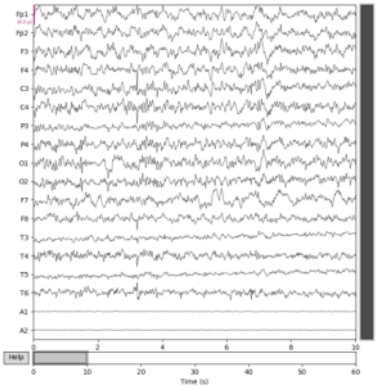
Data Partisipan 15 (stimulus)
Sebelum ICA



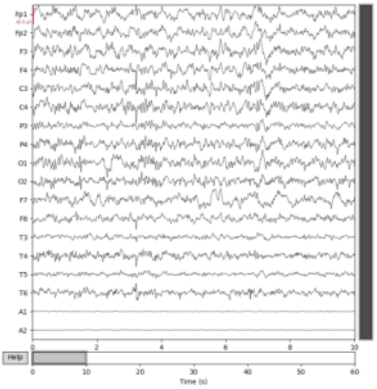
Data Partisipan 15 (stimulus)
Setelah ICA



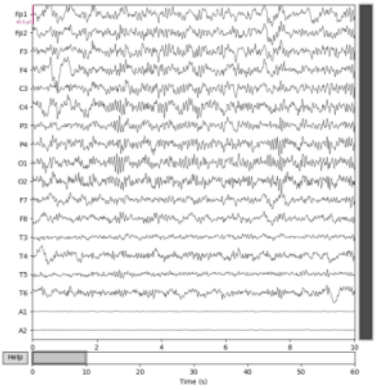
Data Partisipan 16 (non-stim)
Sebelum ICA



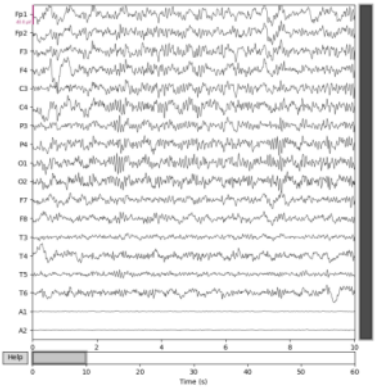
Data Partisipan 16 (non-stim)
Setelah ICA



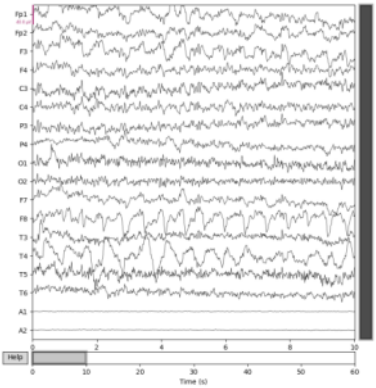
Data Partisipan 16 (stimulus)
Sebelum ICA



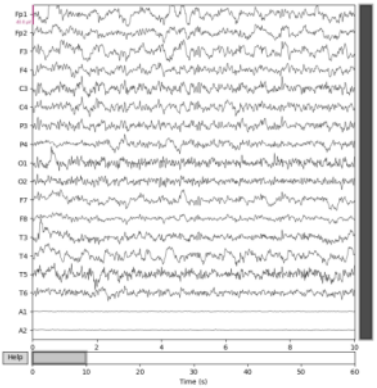
Data Partisipan 16 (stimulus)
Setelah ICA



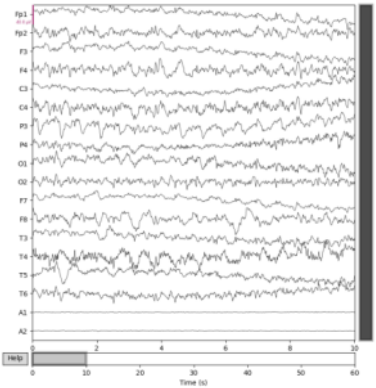
Data Partisipan 17 (non-stim)
Sebelum ICA



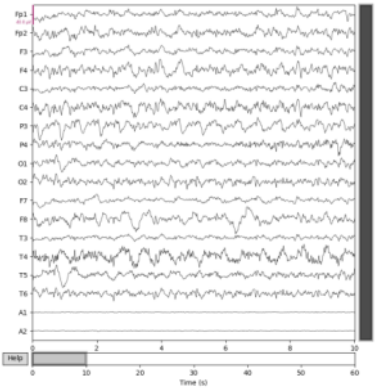
Data Partisipan 17 (non-stim)
Setelah ICA



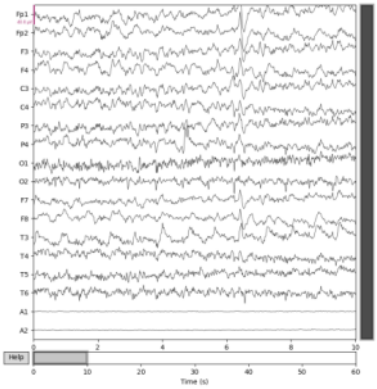
Data Partisipan 17 (stimulus)
Sebelum ICA



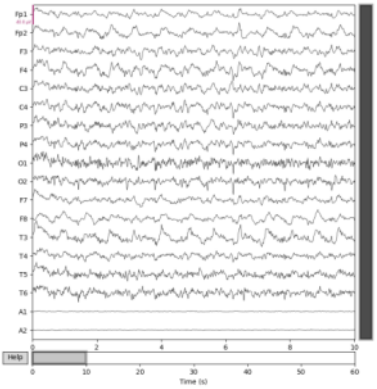
Data Partisipan 17 (stimulus)
Setelah ICA



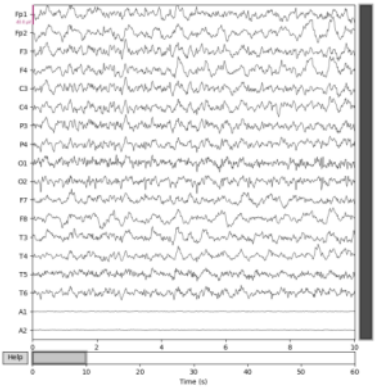
Data Partisipan 18 (non-stim)
Sebelum ICA



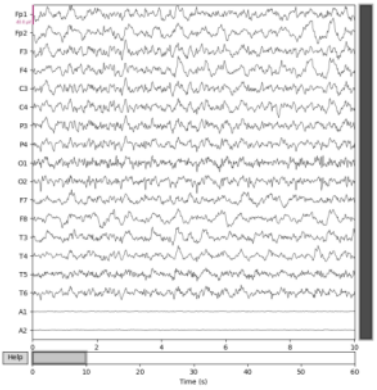
Data Partisipan 18 (non-stim)
Setelah ICA



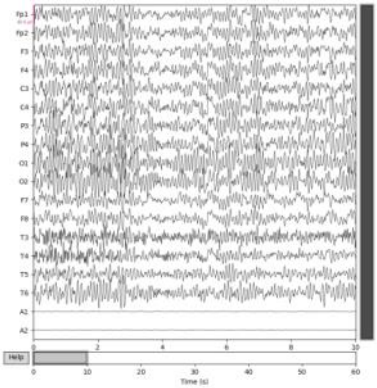
Data Partisipan 18 (stimulus)
Sebelum ICA



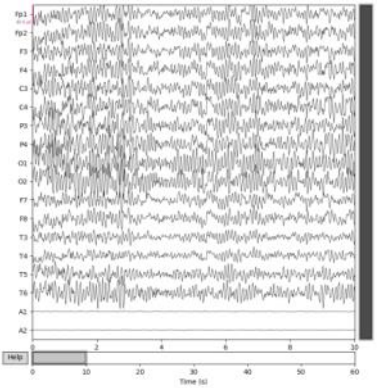
Data Partisipan 18 (stimulus)
Setelah ICA



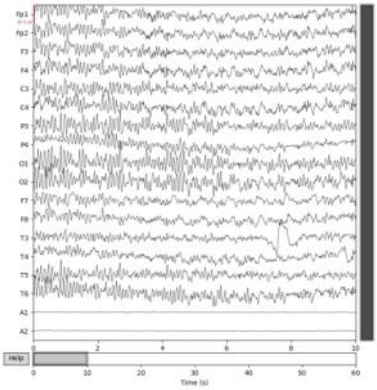
Data Partisipan 19 (non-stim)
Sebelum ICA



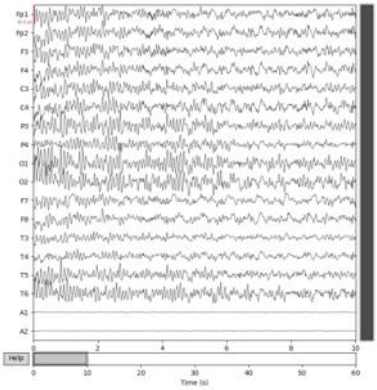
Data Partisipan 19 (non-stim)
Setelah ICA



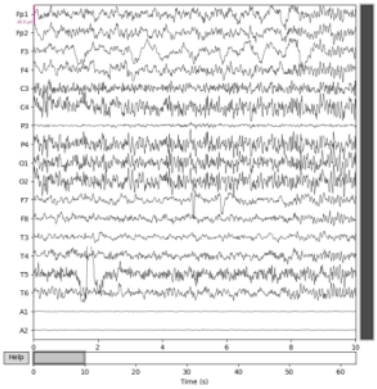
Data Partisipan 19 (stimulus)
Sebelum ICA



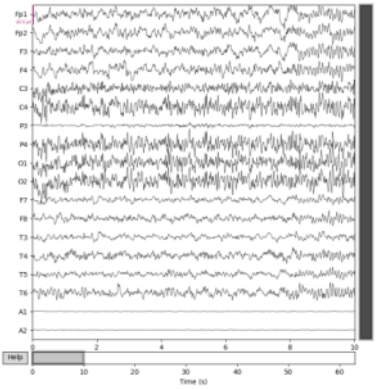
Data Partisipan 19 (stimulus)
Setelah ICA



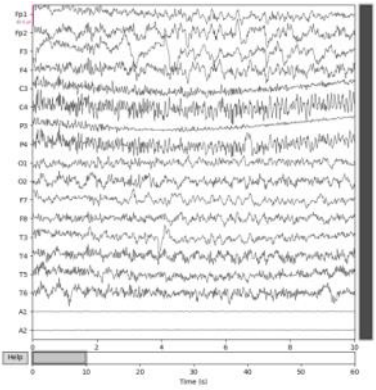
Data Partisipan 20 (non-stim)
Sebelum ICA



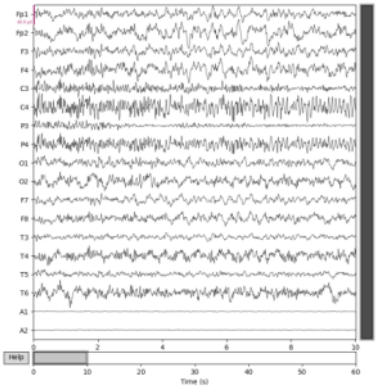
Data Partisipan 20 (non-stim)
Setelah ICA



Data Partisipan 20 (stimulus)
Sebelum ICA



Data Partisipan 20 (stimulus)
Setelah ICA



Lampiran 5. Kode Program Proses PSD untuk Satu Partisipan

```
[ ]: #menginstall library
     !pip install mne==1.2.0
```

```
[ ]: #mengimpor library
     import mne
     import numpy as np
     import pandas as pd
     from mne.time_frequency import psd_welch

     #mengimpor library untuk plotting
     import matplotlib.pyplot as plt
     #tampilan plot
     plt.style.use('seaborn-bright')
     #mengatur DPI
     plt.rcParams['figure.dpi']=300
     #mengatur ukuran figura
     plt.rcParams['figure.figsize']=(8, 6)
```

```
[ ]: #mengimpor data edf
     from google.colab import drive
     drive.mount('/content/drive')
```

```
[ ]: #mengimpor data edf
     xfil_data_nonstim=mne.io.read_raw_edf('/content/drive/MyDrive/
       ↪eeg_data_stim_alfatihah/reconst_data_edf/sheva_nonstim_data_reconst.edf',␣
       ↪preload=True)
     xfil_data_stim=mne.io.read_raw_edf('/content/drive/MyDrive/
       ↪eeg_data_stim_alfatihah/reconst_data_edf/sheva_stim_data_reconst.edf',␣
       ↪preload=True)
```

```
[ ]: #mendefinisikan variabel
     tmin, tmax = 0, 60
     lower_delta, higher_delta = 0.5, 4.0
     lower_theta, higher_theta = 4.0, 8.0
     lower_alpha, higher_alpha = 8.0, 12.0
     lower_beta, higher_beta = 12.0, 25.0
     lower_gamma, higher_gamma = 25.0, 40.0
```

[ ]:
```
#menampilkan plot gelombang delta pada kondisi non stimulus
print("Gelombang Delta pada Kondisi Non-Stimulus")
xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_delta,␣
 ↪fmax=higher_delta, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang theta pada kondisi non stimulus
print("Gelombang Theta pada Kondisi Non-Stimulus")
xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_theta,␣
 ↪fmax=higher_theta, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang alpha pada kondisi non stimulus
print("Gelombang Alpha pada Kondisi Non-Stimulus")
xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_alpha,␣
 ↪fmax=higher_alpha, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang beta pada kondisi non stimulus
print("Gelombang Beta pada Kondisi Non-Stimulus")
xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_beta,␣
 ↪fmax=higher_beta, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang gamma pada kondisi non stimulus
print("Gelombang Gamma pada Kondisi Non-Stimulus")
xfil_data_nonstim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_gamma,␣
 ↪fmax=higher_gamma, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang delta pada kondisi stimulus
print("Gelombang Delta pada Kondisi Stimulus")
xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_delta,␣
 ↪fmax=higher_delta, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang theta pada kondisi stimulus
print("Gelombang Theta pada Kondisi Stimulus")
xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_theta,␣
 ↪fmax=higher_theta, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang alpha pada kondisi stimulus
print("Gelombang Alpha pada Kondisi Stimulus")
xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_alpha,␣
 ↪fmax=higher_alpha, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang beta pada kondisi stimulus
print("Gelombang Beta pada Kondisi Stimulus")
xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_beta,␣
 ↪fmax=higher_beta, show=False, average=True, color='green', area_mode=None)
```

[ ]:
```
#menampilkan plot gelombang gamma pada kondisi stimulus
print("Gelombang Gamma pada Kondisi Stimulus")
```

```
xfil_data_stim.plot_psd(tmin=tmin, tmax=tmax, fmin=lower_gamma,␣
 ↪fmax=higher_gamma, show=False, average=True, color='green', area_mode=None)
```

```
[ ]: # Mendefinisikan fungsi
def eeg_power_band(raw):

    # band frekuensi spesifik
    FREQ_BANDS = {'delta': [0.5, 4.0],
                  'theta': [4.0, 8.0],
                  'alpha': [8.0, 12.0],
                  'beta': [12.0, 25.0],
                  'gamma': [25.0, 40.0]}

    psds, freqs = psd_welch(raw, picks='eeg', fmin=0.5, fmax=40)

    # Normalisasi psd
    psds /= np.sum(psds, axis=-1, keepdims=True)

    X = []
    for fmin, fmax in FREQ_BANDS.values():
        psds_band = psds[:, (freqs >= fmin) & (freqs < fmax)].mean(axis=-1)
        X.append(psds_band.reshape(len(psds), -1))

    return np.concatenate(X, axis=1)

#menghitung psd
power_psd_nonstim = eeg_power_band(xfil_data_nonstim)
power_psd_stim = eeg_power_band(xfil_data_stim)
```

```
[ ]: #Menampilkan Hasil PSD pada Kondisi Non-Stimulus
#Average Band Power
np_arr_nonstim = np.array(power_psd_nonstim)
df_psd_nonstim = pd.DataFrame(np_arr_nonstim,
              index=xfil_data_nonstim.info['ch_names'],
              columns=['delta',
                       'theta',
                       'alpha',
                       'beta',
                       'gamma'])
df_psd_nonstim.head(18)
```

```
[ ]: #Menampilkan Hasil PSD pada Kondisi Stimulus
#Average Band Power
np_arr_stim = np.array(power_psd_stim)
df_psd_stim = pd.DataFrame(np_arr_stim,
              index=xfil_data_stim.info['ch_names'],
              columns=['delta',
                       'theta',
                       'alpha',
                       'beta',
                       'gamma'])
df_psd_stim.head(18)
```

[ ]:
```
#menghitung nilai persentase gelombang pada kondisi non-stimulus
#Relative Power
df_nonstim = df_psd_nonstim
df_nonstim['total'] = df_nonstim.sum(axis=1)
df_nonstim['%delta'] = df_nonstim['delta'] / df_nonstim['total']
df_nonstim['%theta'] = df_nonstim['theta'] / df_nonstim['total']
df_nonstim['%alpha'] = df_nonstim['alpha'] / df_nonstim['total']
df_nonstim['%beta'] = df_nonstim['beta'] / df_nonstim['total']
df_nonstim['%gamma'] = df_nonstim['gamma'] / df_nonstim['total']
df_nonstim
```

[ ]:
```
#menghitung nilai persentase gelombang pada kondisi stimulus
#Relative Power
df_stim = df_psd_stim
df_stim['total'] = df_stim.sum(axis=1)
df_stim['%delta'] = df_stim['delta'] / df_stim['total']
df_stim['%theta'] = df_stim['theta'] / df_stim['total']
df_stim['%alpha'] = df_stim['alpha'] / df_stim['total']
df_stim['%beta'] = df_stim['beta'] / df_stim['total']
df_stim['%gamma'] = df_stim['gamma'] / df_stim['total']
df_stim
```

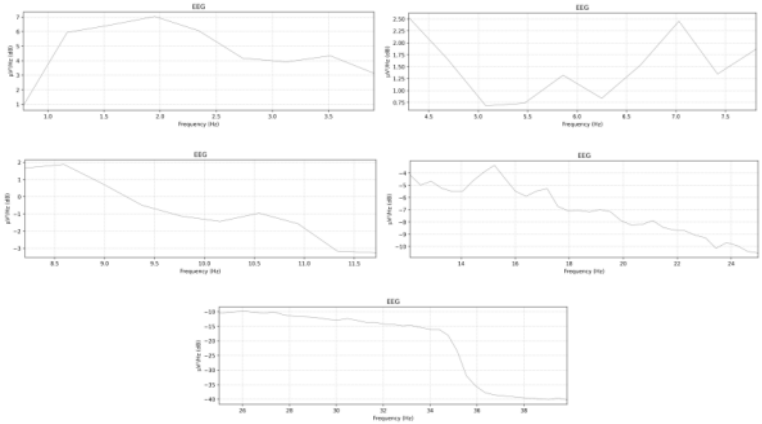
[ ]:
```
#lanjutan... (menghitung nilai persentase gelombang pada kondisi non-stimulus)
pr_nonstim_delta = np.array(df_nonstim['%delta'].mean())
pr_nonstim_theta = np.array(df_nonstim['%theta'].mean())
pr_nonstim_alpha = np.array(df_nonstim['%alpha'].mean())
pr_nonstim_beta = np.array(df_nonstim['%beta'].mean())
pr_nonstim_gamma = np.array(df_nonstim['%gamma'].mean())

#menampilkan hasil persentase
print("Persentase Gelombang pada Keadaan Non Stimulus\n")
array_nonstim = (['delta', '{:.0%}'.format(pr_nonstim_delta)],
                 ['theta', '{:.0%}'.format(pr_nonstim_theta)],
                 ['alpha', '{:.0%}'.format(pr_nonstim_alpha)],
                 ['beta', '{:.0%}'.format(pr_nonstim_beta)],
                 ['gamma', '{:.0%}'.format(pr_nonstim_gamma)])
df_persentase_nonstim = pd.DataFrame(data=array_nonstim,
             columns=['gelombang',
                      'persentase'])
df_persentase_nonstim.head()
```

```
[ ]: #lanjutan... (menghitung nilai persentase gelombang pada kondisi stimulus)
pr_stim_delta = np.array(df_stim['%delta'].mean())
pr_stim_theta = np.array(df_stim['%theta'].mean())
pr_stim_alpha = np.array(df_stim['%alpha'].mean())
pr_stim_beta = np.array(df_stim['%beta'].mean())
pr_stim_gamma = np.array(df_stim['%gamma'].mean())

#menampilkan hasil persentase
print("Persentase Gelombang pada Keadaan dengan Stimulus\n")
array_stim = (['delta', '{:.0%}'.format(pr_stim_delta)],
              ['theta', '{:.0%}'.format(pr_stim_theta)],
              ['alpha', '{:.0%}'.format(pr_stim_alpha)],
              ['beta', '{:.0%}'.format(pr_stim_beta)],
              ['gamma', '{:.0%}'.format(pr_stim_gamma)])
df_persentase_stim = pd.DataFrame(data=array_stim,
             columns=['gelombang',
                      'persentase'])
df_persentase_stim.head()
```

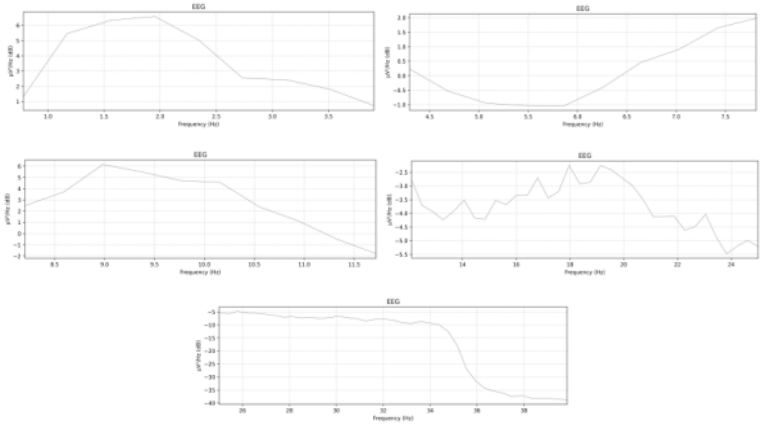
```
[ ]: #menghitung nilai persentase gelombang (kenaikan dan penurunan)
akhir_delta =  pr_stim_delta - pr_nonstim_delta
akhir_theta =  pr_stim_theta - pr_nonstim_theta
akhir_alpha =  pr_stim_alpha - pr_nonstim_alpha
akhir_beta =  pr_stim_beta - pr_nonstim_beta
akhir_gamma = pr_stim_gamma - pr_nonstim_gamma

#menampilkan hasil persentase
print("Kenaikan / Penurunan Gelombang \n")
print("Ket : nilai positif menandakan kenaikan gelombang \n")
print("      nilai negatif menandakan penurunan gelombang")
array_akhir = (['delta', '{:.0%}'.format(akhir_delta)],
               ['tetha', '{:.0%}'.format(akhir_theta)],
               ['alpha', '{:.0%}'.format(akhir_alpha)],
               ['beta', '{:.0%}'.format(akhir_beta)],
               ['gamma', '{:.0%}'.format(akhir_gamma)])
df_persentase_akhir = pd.DataFrame(data=array_akhir,
             columns=['gelombang',
                      'persentase'])
df_persentase_akhir.head()
```

Lampiran 6. Visualisasi Grafik PSD Pada Tiap Partisipan

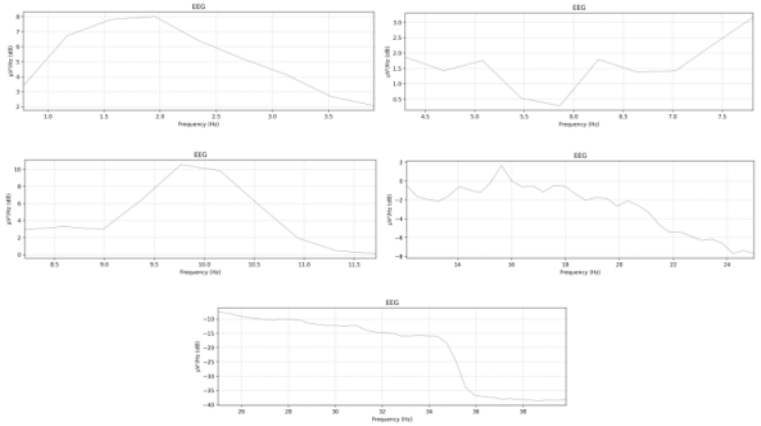
Grafik PSD Partisipan 1 *delta, tetha, alpha, beta, gamma* (non-stimulus)



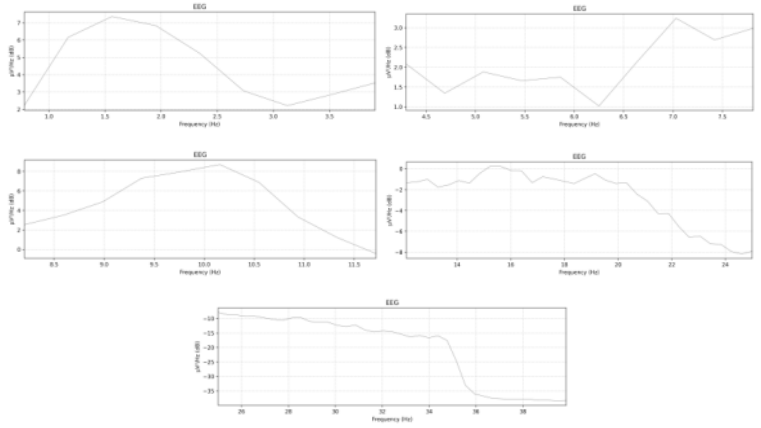
Grafik PSD Partisipan 1 *delta, tetha, alpha, beta, gamma* (stimulus)



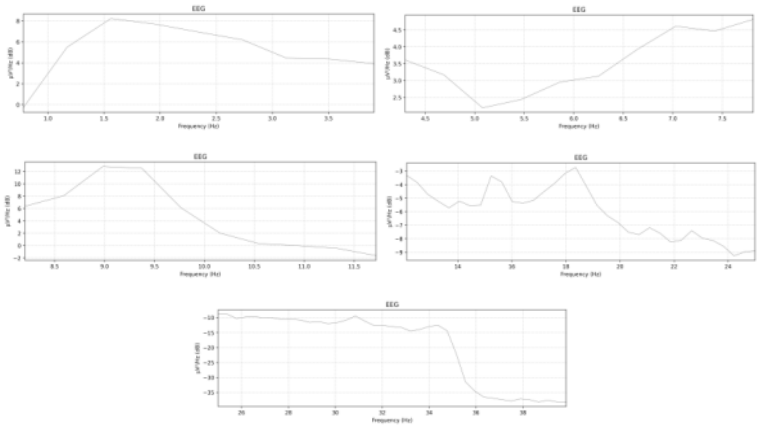
Grafik PSD Partisipan 2 *delta, tetha, alpha, beta, gamma* (non-stimulus)



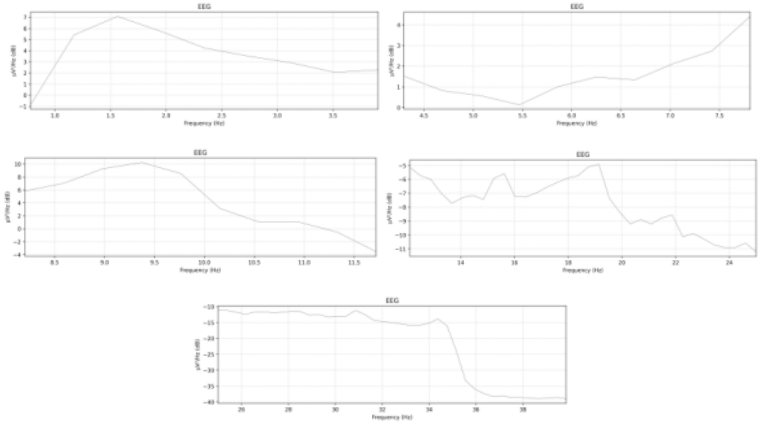
Grafik PSD Partisipan 2 *delta, tetha, alpha, beta, gamma* (stimulus)



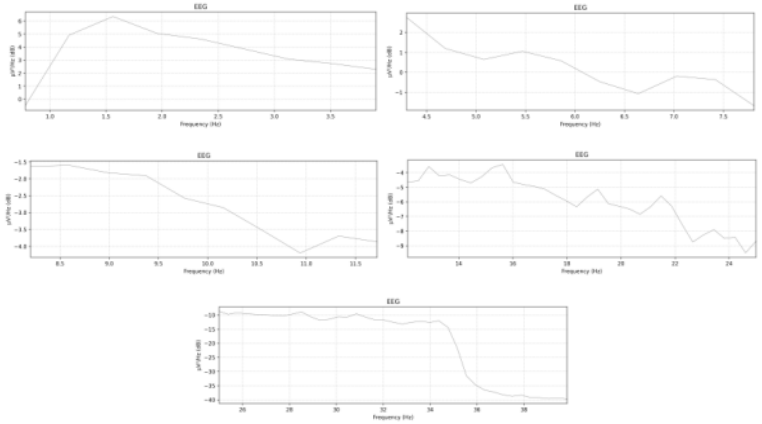
Grafik PSD Partisipan 3 *delta, tetha, alpha, beta, gamma* (non-stimulus)



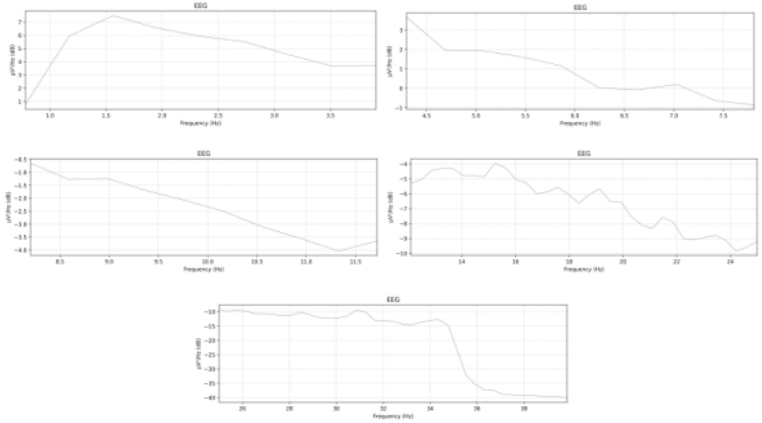
Grafik PSD Partisipan 3 *delta, tetha, alpha, beta, gamma* (stimulus)



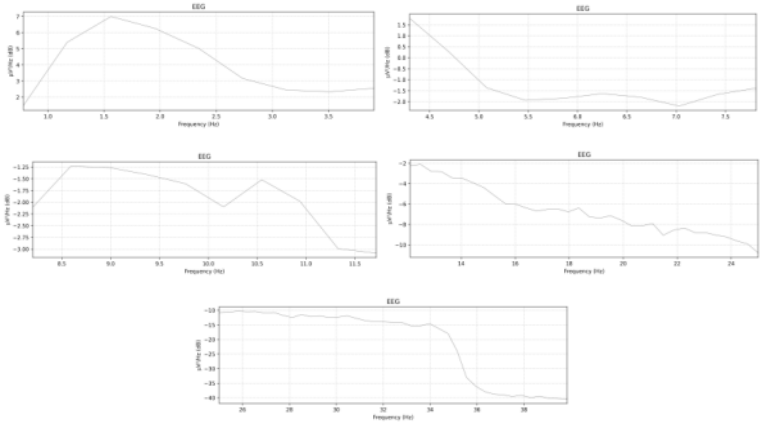
Grafik PSD Partisipan 4 *delta, tetha, alpha, beta, gamma* (non-stimulus)



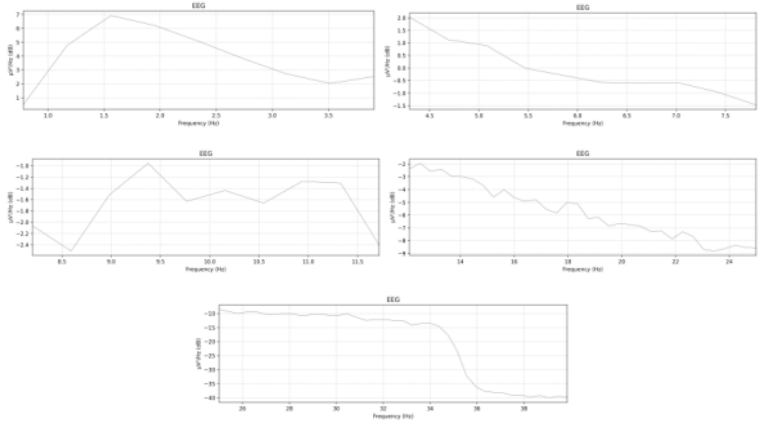
Grafik PSD Partisipan 4 *delta, tetha, alpha, beta, gamma* (stimulus)



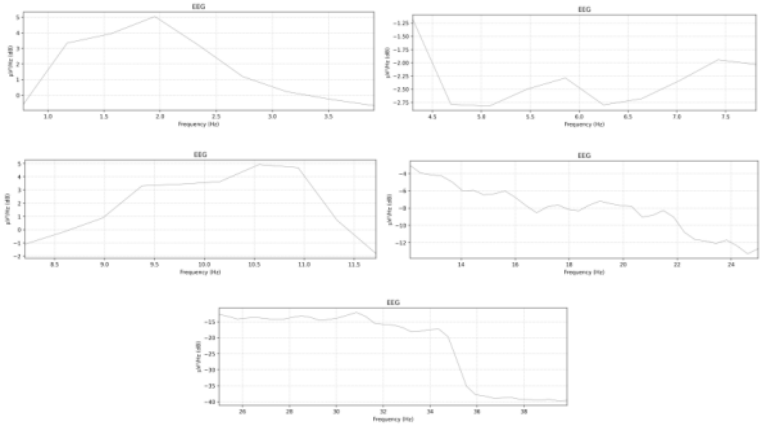
Grafik PSD Partisipan 5 *delta, tetha, alpha, beta, gamma* (non-stimulus)



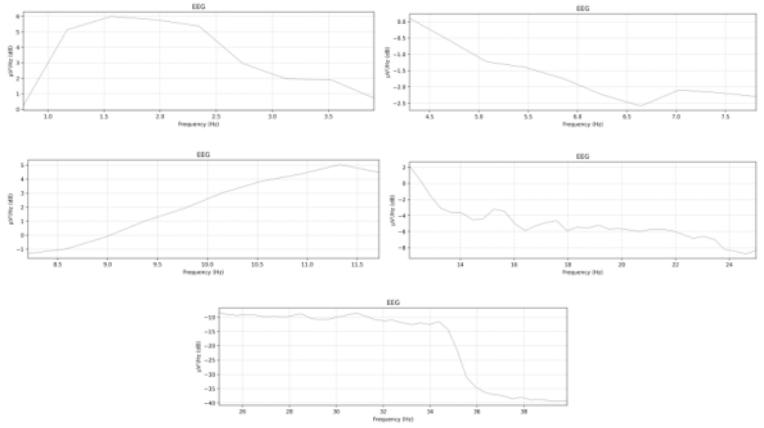
Grafik PSD Partisipan 5 *delta, tetha, alpha, beta, gamma* (stimulus)



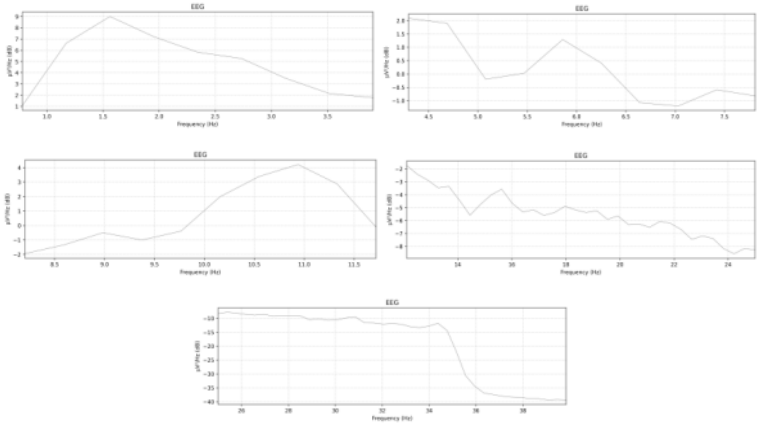
Grafik PSD Partisipan 6 *delta, tetha, alpha, beta, gamma* (non-stimulus)



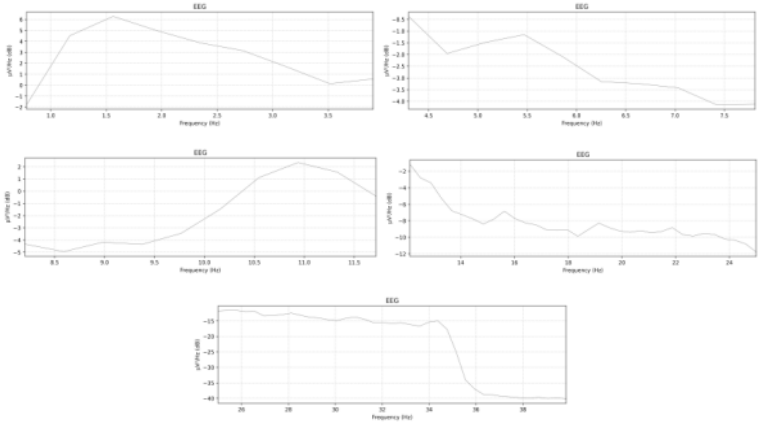
Grafik PSD Partisipan 6 *delta, tetha, alpha, beta, gamma* (stimulus)



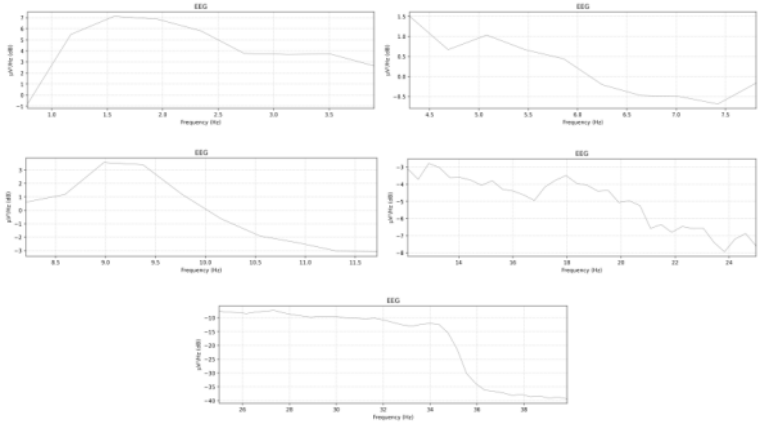
Grafik PSD Partisipan 7 *delta, tetha, alpha, beta, gamma* (non-stimulus)



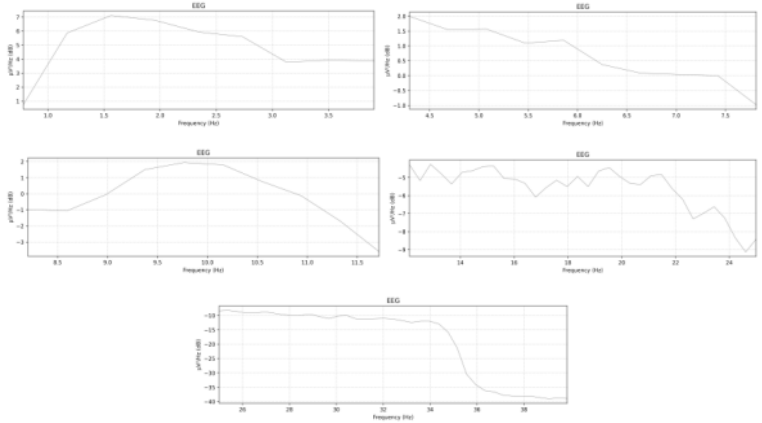
Grafik PSD Partisipan 7 *delta, tetha, alpha, beta, gamma* (stimulus)



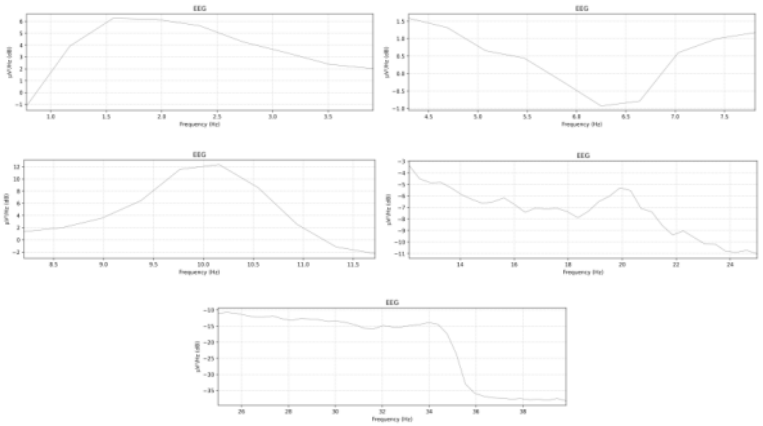
Grafik PSD Partisipan 8 *delta, tetha, alpha, beta, gamma* (non-stimulus)



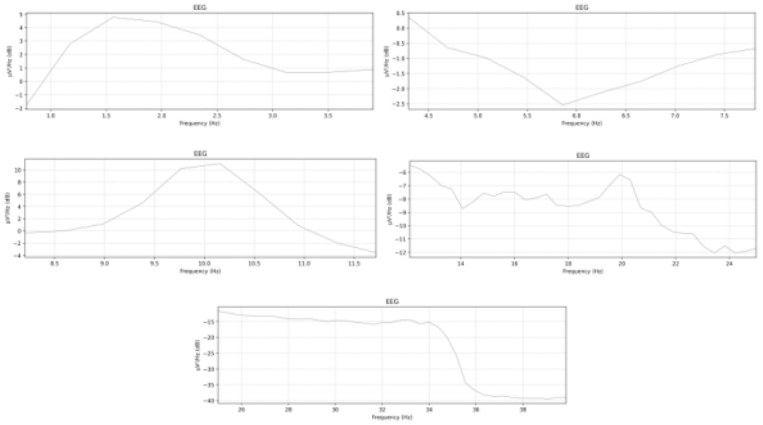
Grafik PSD Partisipan 8 *delta, tetha, alpha, beta, gamma* (stimulus)



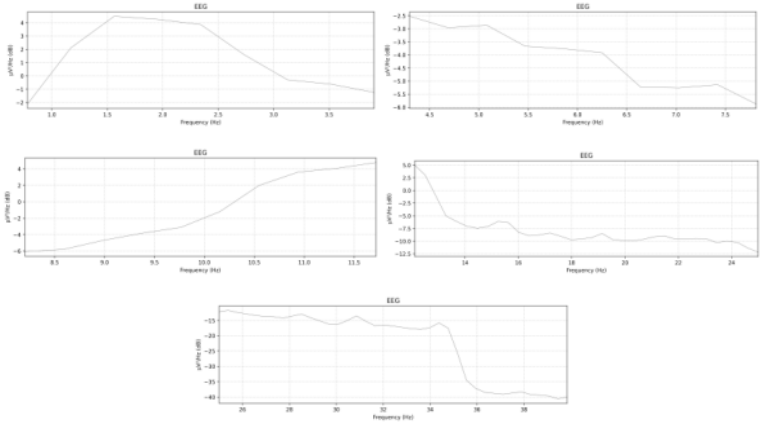
Grafik PSD Partisipan 9 *delta, tetha, alpha, beta, gamma* (non-stimulus)



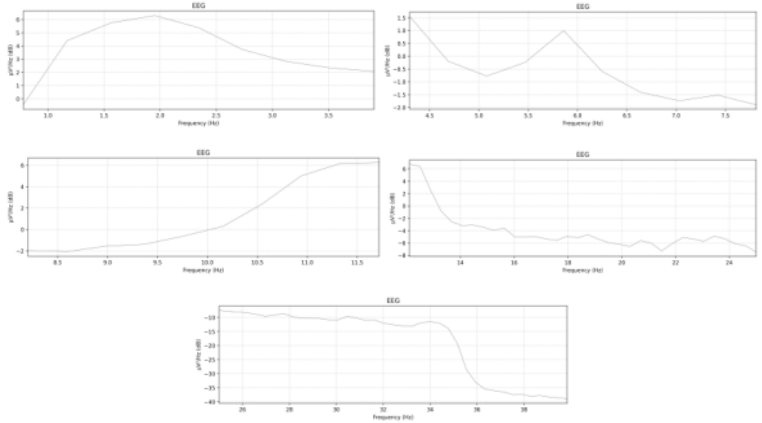
Grafik PSD Partisipan 9 *delta, tetha, alpha, beta, gamma* (stimulus)



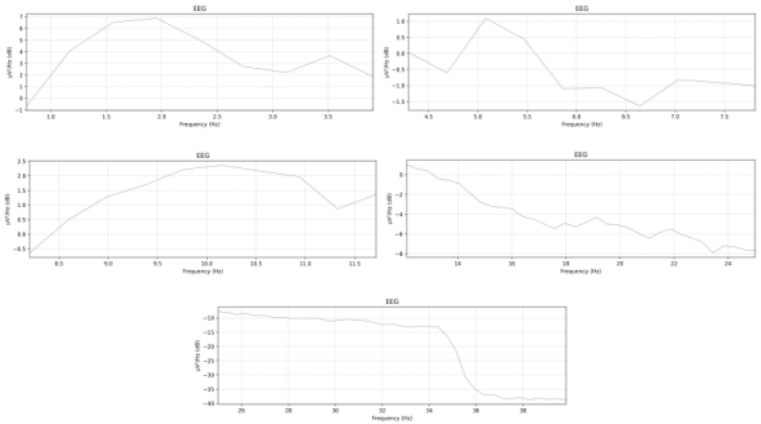
Grafik PSD Partisipan 10 *delta, tetha, alpha, beta, gamma* (non-stimulus)



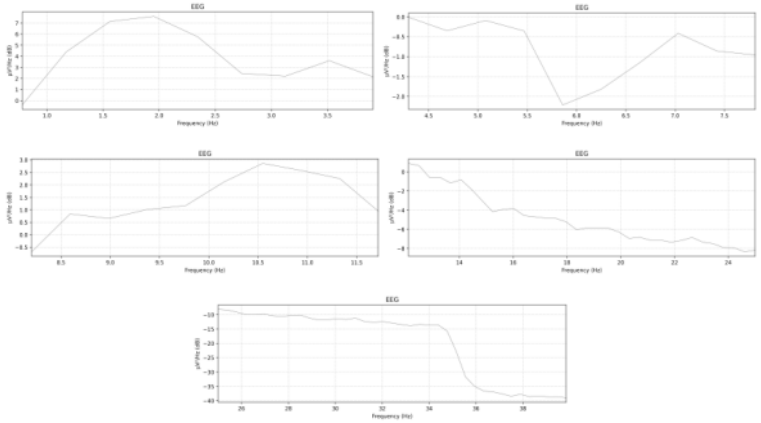
Grafik PSD Partisipan 10 *delta, tetha, alpha, beta, gamma* (stimulus)



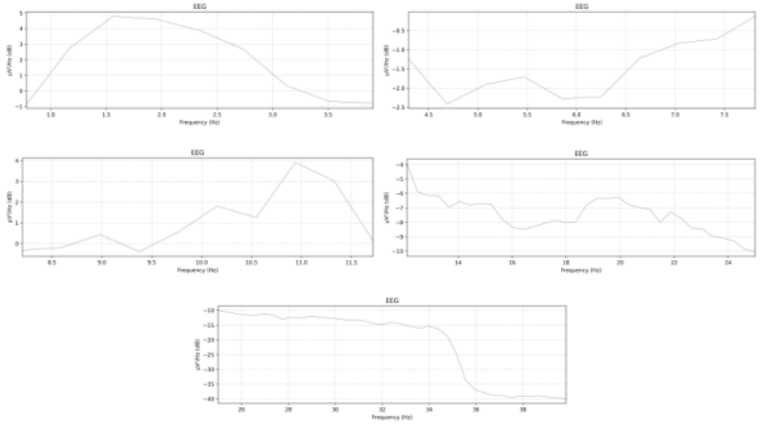
Grafik PSD Partisipan 11 *delta, tetha, alpha, beta, gamma* (non-stimulus)



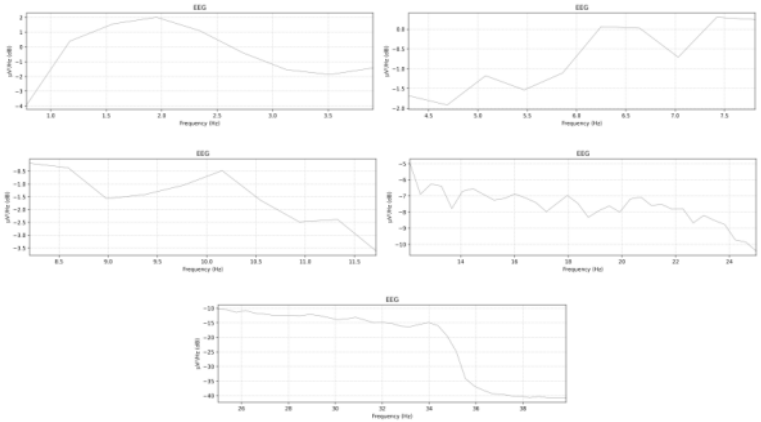
Grafik PSD Partisipan 11 *delta, tetha, alpha, beta, gamma* (stimulus)



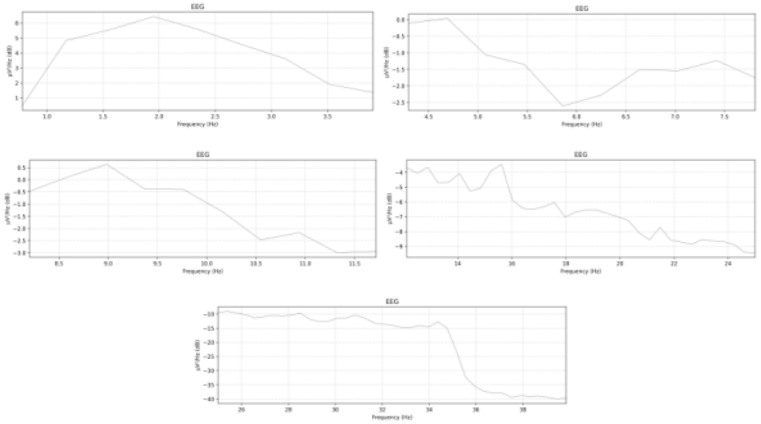
Grafik PSD Partisipan 12 *delta, tetha, alpha, beta, gamma* (non-stimulus)



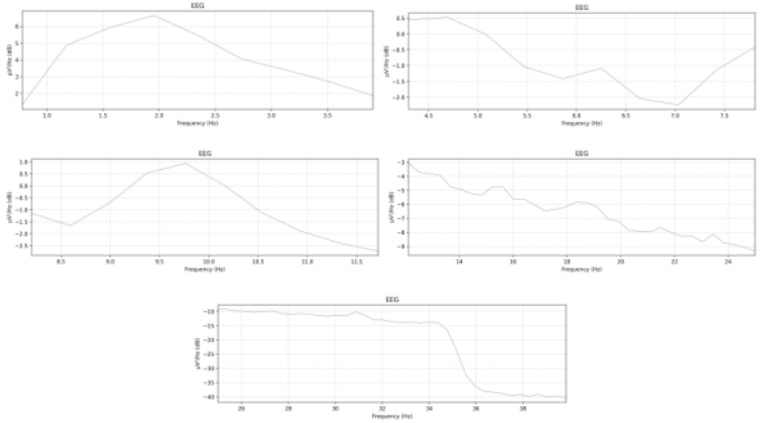
Grafik PSD Partisipan 12 *delta, tetha, alpha, beta, gamma* (stimulus)



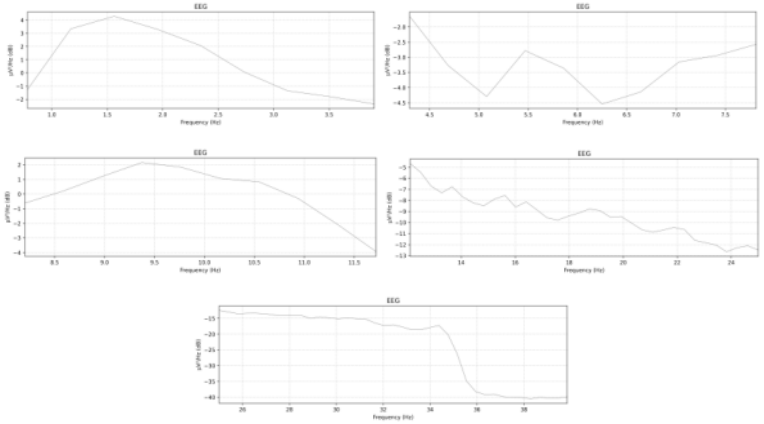
Grafik PSD Partisipan 13 *delta, tetha, alpha, beta, gamma* (non-stimulus)



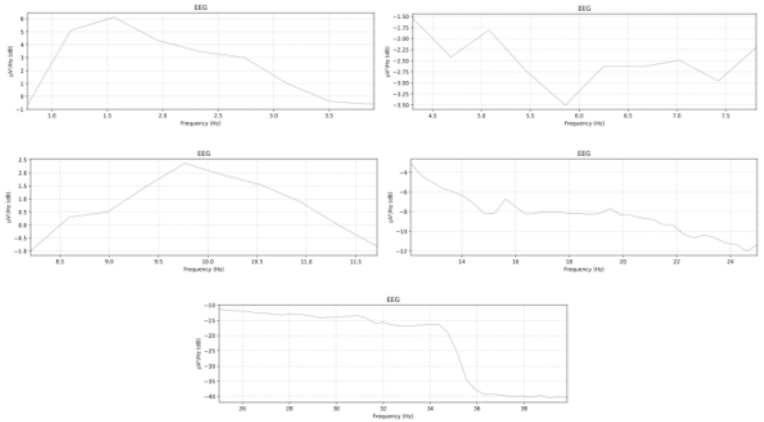
Grafik PSD Partisipan 13 *delta, tetha, alpha, beta, gamma* (stimulus)



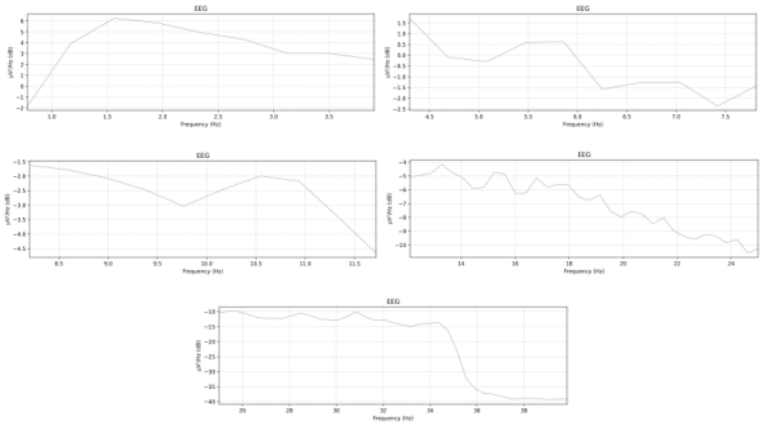
Grafik PSD Partisipan 14 *delta, tetha, alpha, beta, gamma* (non-stimulus)



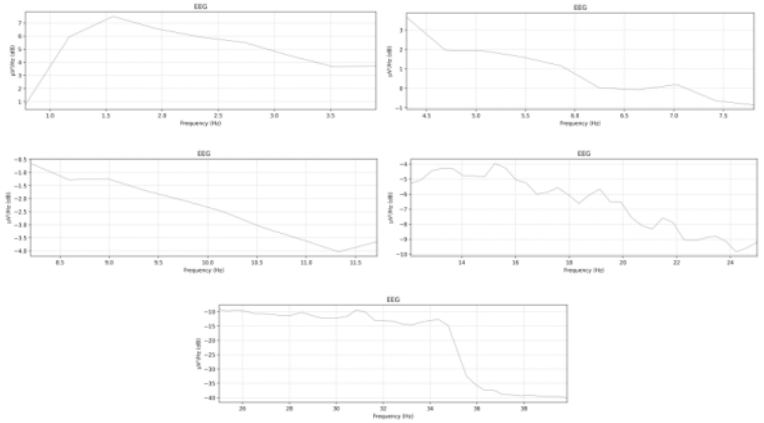
Grafik PSD Partisipan 14 *delta, tetha, alpha, beta, gamma* (stimulus)



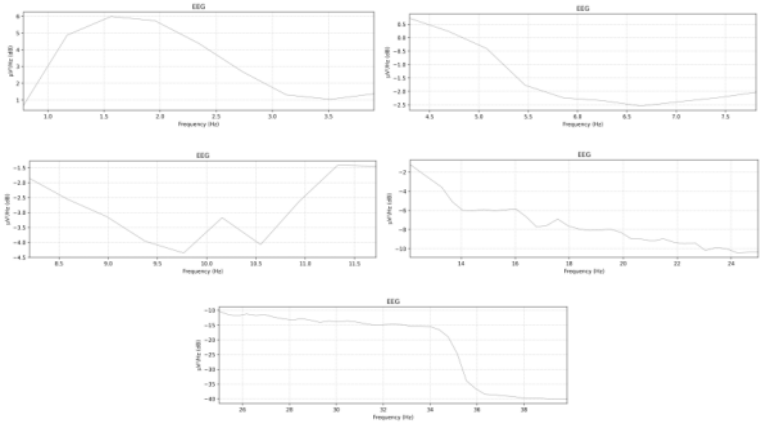
Grafik PSD Partisipan 15 *delta, tetha, alpha, beta, gamma* (non-stimulus)



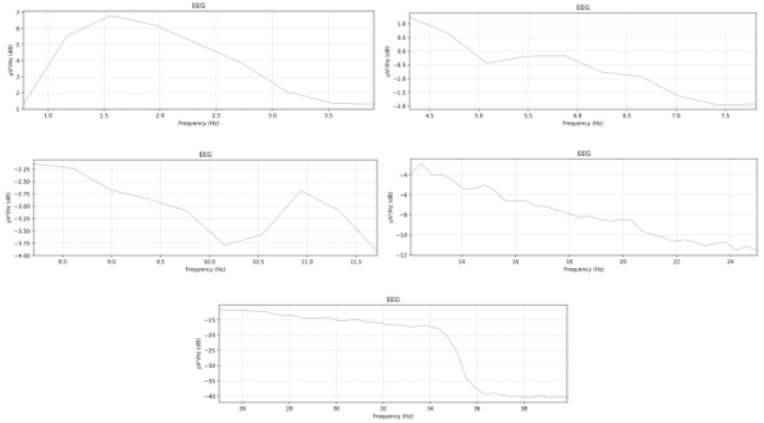
Grafik PSD Partisipan 15 *delta, tetha, alpha, beta, gamma* (stimulus)



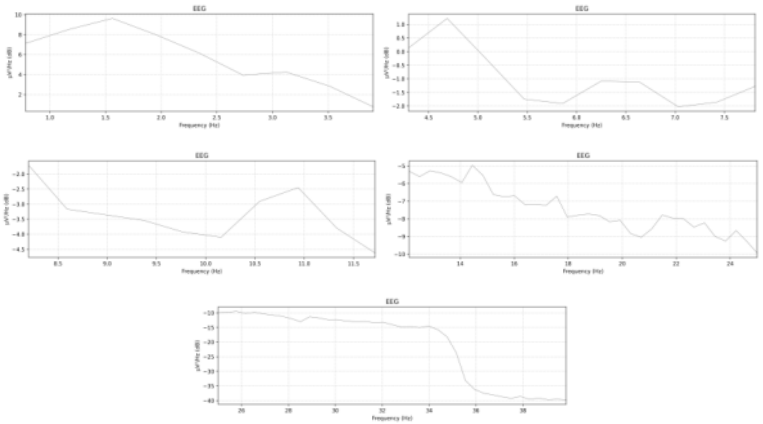
Grafik PSD Partisipan 16 *delta, tetha, alpha, beta, gamma* (non-stimulus)



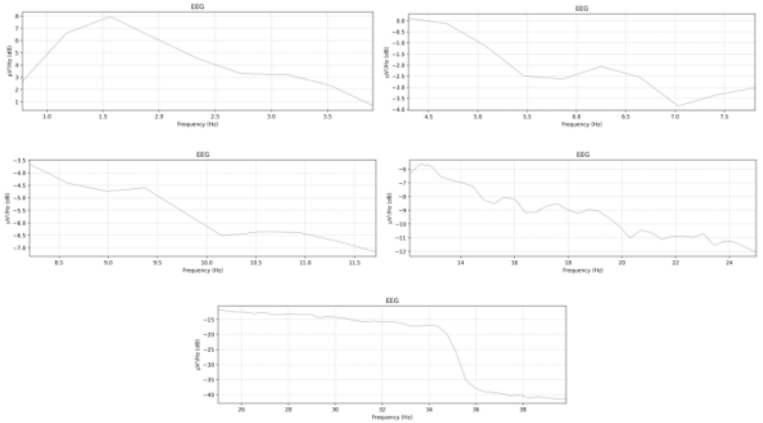
Grafik PSD Partisipan 16 *delta, tetha, alpha, beta, gamma* (stimulus)



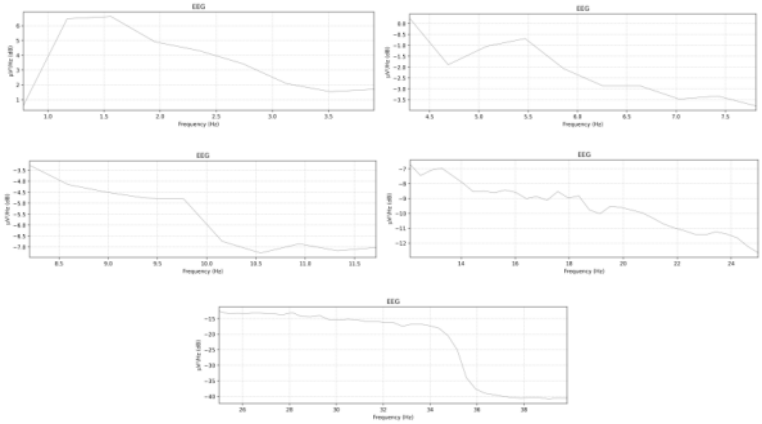
Grafik PSD Partisipan 17 *delta, tetha, alpha, beta, gamma* (non-stimulus)



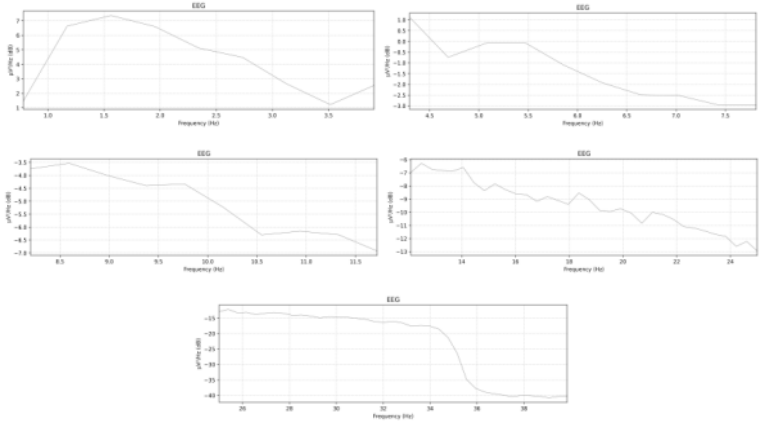
Grafik PSD Partisipan 17 *delta, tetha, alpha, beta, gamma* (stimulus)



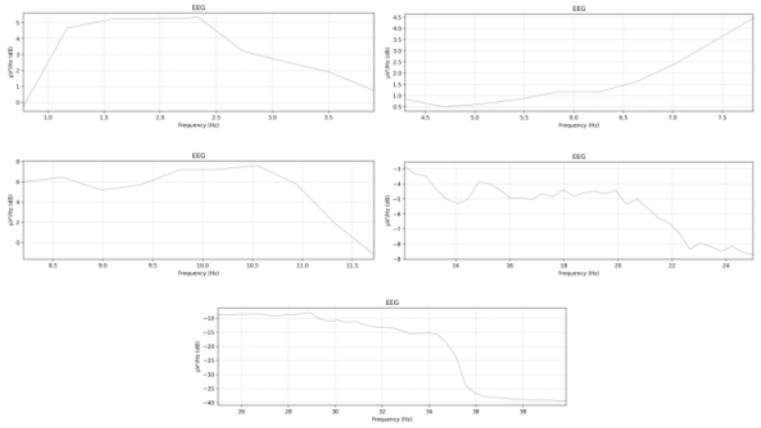
Grafik PSD Partisipan 18 *delta, tetha, alpha, beta, gamma* (non-stimulus)



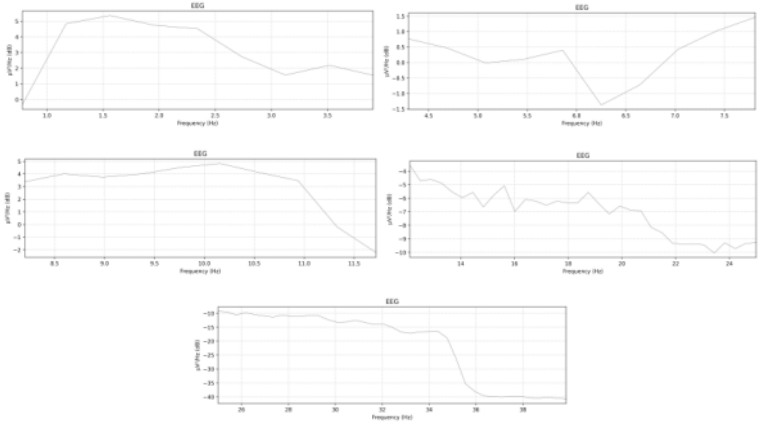
Grafik PSD Partisipan 18 *delta, tetha, alpha, beta, gamma* (stimulus)



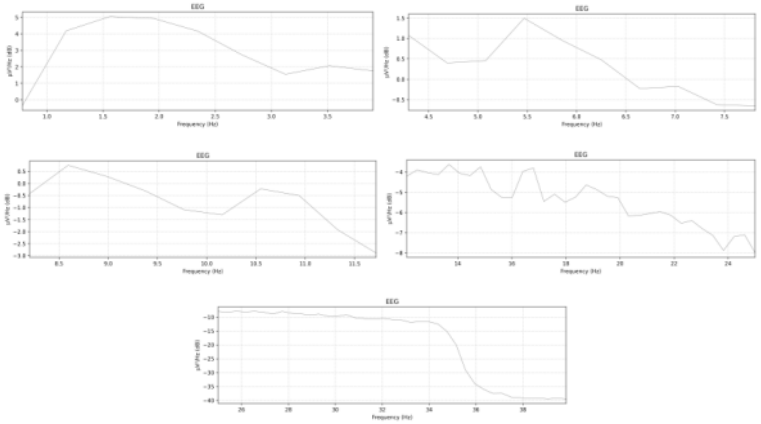
Grafik PSD Partisipan 19 *delta, tetha, alpha, beta, gamma* (non-stimulus)



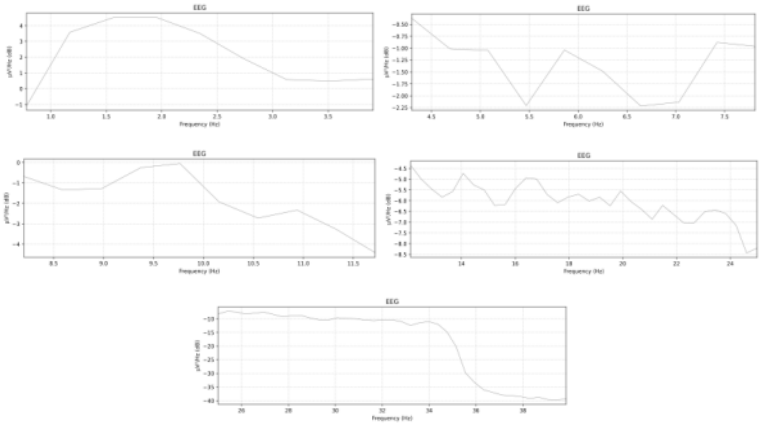
Grafik PSD Partisipan 19 *delta, tetha, alpha, beta, gamma* (stimulus)



Grafik PSD Partisipan 20 *delta, tetha, alpha, beta, gamma* (non-stimulus)



Grafik PSD Partisipan 20 *delta, tetha, alpha, beta, gamma* (stimulus)



Lampiran 7. Tabel Gelombang Otak Berlabel

id	δ (%)	θ (%)	α (%)	β (%)	γ (%)	ket.	$Gel.\delta$ naik
part_1_nstim	54	24	16	5	1	nonstim	-
part_1_stim	40	16	35	7	2	stimulus	tidak
part_2_nstim	37	15	41	6	1	nonstim	-
part_2_stim	34	18	40	7	1	stimulus	tidak
part_3_nstim	30	18	48	3	1	nonstim	-
part_3_stim	29	17	50	3	1	stimulus	tidak
part_4_nstim	55	24	12	7	2	nonstim	-
part_4_stim	59	23	11	5	1	stimulus	ya
part_5_nstim	60	18	15	6	1	nonstim	-
part_5_stim	56	22	14	7	1	stimulus	tidak
part_6_nstim	39	13	43	4	1	nonstim	-
part_6_stim	44	12	36	7	1	stimulus	ya
part_7_nstim	54	16	22	7	2	nonstim	-
part_7_stim	56	15	23	6	1	stimulus	ya
part_8_nstim	53	18	21	7	2	nonstim	-
part_8_stim	55	19	19	6	1	stimulus	ya
part_9_nstim	27	12	57	3	1	nonstim	-
part_9_stim	26	12	58	3	1	stimulus	tidak
part_10_nstim	44	11	34	10	1	nonstim	-
part_10_stim	42	15	30	12	1	stimulus	tidak
part_11_nstim	47	17	26	9	1	nonstim	-
part_11_stim	50	15	26	8	1	stimulus	ya
part_12_nstim	42	17	35	5	1	nonstim	-
part_12_stim	35	31	25	7	2	stimulus	tidak
part_13_nstim	58	16	18	6	1	nonstim	-
part_13_stim	57	17	18	6	1	stimulus	tidak
part_14_nstim	43	15	37	5	1	nonstim	-
part_14_stim	49	13	32	5	1	stimulus	ya
part_15_nstim	58	21	14	6	1	nonstim	-
part_15_stim	59	23	11	5	1	stimulus	ya
part_16_nstim	59	19	15	7	1	nonstim	-
part_16_stim	61	21	12	5	1	stimulus	ya
part_17_nstim	72	14	9	4	1	nonstim	-
part_17_stim	73	16	7	3	1	stimulus	ya
part_18_nstim	69	18	9	4	1	nonstim	-
part_18_stim	70	17	8	4	1	stimulus	ya

part_19_nstim	29	19	48	4	1	nonstim	-
part_19_stim	38	18	39	4	1	stimulus	ya
part_20_nstim	47	23	21	8	2	nonstim	-
part_20_stim	50	20	20	8	2	stimulus	ya

Lampiran 8. Kode Perbandingan Algoritma Machine Learning Pada Data Gelombang Otak

```
[ ]: #import google drive
     from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

```
[ ]: #mengimpor library yang diperlukan
     import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
```

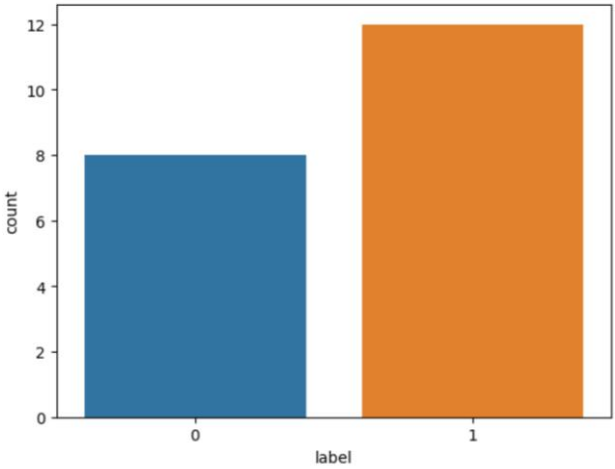
```
[ ]: #mengimpor dataset sebagai dataframe
     df_eeg = pd.read_csv('/content/drive/MyDrive/eeg_data_stim_alfatihah/
       ↪combined_data_eeg_labeled_one_row.csv')
     df_eeg.head()
```

```
[ ]:          0         1         2         3         4         5         6  \
     0  0.000319  0.000319  0.000319  0.000319  0.000319  0.000319  0.000319
     1  0.001285  0.001285  0.001285  0.001285  0.001285  0.001285  0.001285
     2  0.001803  0.001803 -0.002239 -0.000218  0.003824 -0.002239 -0.000218
     3  0.004527  0.004527  0.004527  0.004527  0.004527  0.004527  0.004527
     4  0.021220  0.004357  0.004357  0.004357  0.004357  0.004357  0.004357

               7         8         9  …     107991    107992     107993  \
     0  0.000319  0.000319  0.000319  …  -3.185368 -3.810312   0.556671
     1  0.001285  0.001285  0.001285  … -14.106295 -3.829575   1.838489
     2 -0.000218 -0.000218  0.003824  …   9.587521 -1.057214 -11.988934
     3  0.004527  0.004527  0.004527  … -12.162396  5.730739 -11.524449
     4  0.012789  0.000141  0.038084  …   0.038084  0.004357   0.004357

          107994     107995    107996     107997    107998    107999  label
     0 -6.441552 -13.401670  8.301778  -4.513373 -0.140674 -0.186402      0
     1 -4.319049  -6.799945 -1.008953  -9.703264 -0.615586  0.121977      0
     2 -0.228594 -11.006716 -2.463847   5.565683  0.282725  0.369629      0
     3  7.358779  -9.911719  9.354277 -12.846275  0.708821 -0.220030      1
     4  0.004357   0.004357  0.004357   0.004357  0.000141  0.000141      0

     [5 rows x 108001 columns]
```

```
[ ]: #melihat informasi data
     df_eeg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Columns: 108001 entries, 0 to label
dtypes: float64(108000), int64(1)
memory usage: 16.5 MB
```

```
[ ]: #exploratory data analysis (EDA)
     #melihat isi grafik label

     plt.figure()
     sns.countplot(x='label', data=df_eeg)
     plt.show()
```



```
[ ]: #menentukan variabel penjelas (X) (feature)
     X = df_eeg.drop(['label'], axis=1)
```

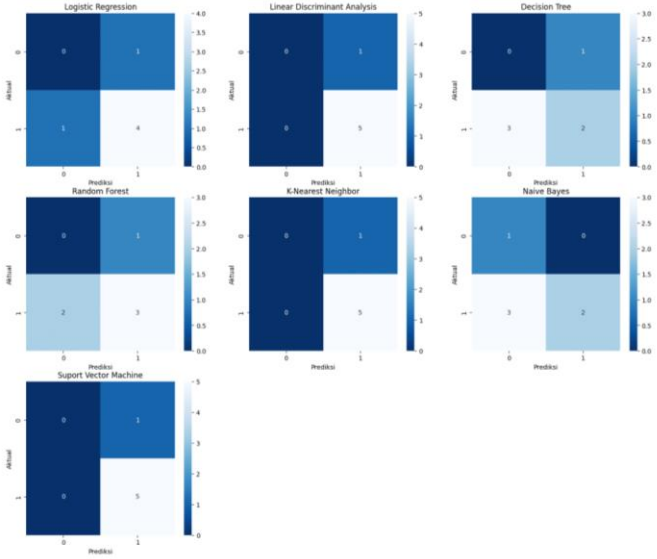
```
[ ]: #menentukan variabel respon (y) (target)
     y = df_eeg['label']
```

```
[ ]: #membuat model machine learning
     #impor library untuk splitting dan testing
     from sklearn.model_selection import train_test_split, cross_val_score
     #memisahkan (splitting) dataset menjadi data training dan data testing
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
      ↪random_state=12)
```

```
[ ]: #membuat pipeline model untuk machine learning
     from sklearn.linear_model import LogisticRegression
     from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.naive_bayes import GaussianNB
     from sklearn.svm import SVC

     model_pipeline = []
     model_pipeline.append(LogisticRegression())
     model_pipeline.append(LinearDiscriminantAnalysis())
     model_pipeline.append(DecisionTreeClassifier())
     model_pipeline.append(RandomForestClassifier())
     model_pipeline.append(KNeighborsClassifier())
     model_pipeline.append(GaussianNB())
     model_pipeline.append(SVC())
```

```
[ ]: #evaluasi model machine learning
     from sklearn import metrics
     from sklearn.metrics import classification_report
     from sklearn.metrics import confusion_matrix

     model_list = ['Logistic Regression','Linear Discriminant Analysis',
                   'Decision Tree', 'Random Forest', 'K-Nearest Neighbor',
                   'Naive Bayes' , 'Suport Vector Machine']

     acc_list = []
     auc_list = []
     cm_list = []

     for model in model_pipeline:
       model.fit(X_train, y_train)
       y_pred = model.predict(X_test)
       acc_list.append(metrics.accuracy_score(y_test, y_pred))
       fpr, tpr, _thresholds = metrics.roc_curve(y_test, y_pred, pos_label=1)
       auc_list.append(round(metrics.auc(fpr, tpr),2))

       cm_list.append(confusion_matrix(y_test, y_pred))
```

```
[ ]: #tampilan plot confusion matrix
fig = plt.figure(figsize = (18,15))
for i in range(len(cm_list)):
  cm = cm_list[i]
  model = model_list[i]
  sub = fig.add_subplot(3, 3, i+1).set_title(model)
  cm_plot = sns.heatmap(cm, annot=True, cmap='Blues_r')
  cm_plot.set_xlabel('Prediksi')
  cm_plot.set_ylabel('Aktual')
```




```
[ ]: #melihat akurasi dan auc
hasil_df = pd.DataFrame({'Model':model_list, 'Akurasi':acc_list, 'AUC':
  ↪auc_list})
hasil_df
```

[ ]:

	Model	Akurasi	AUC
0	Logistic Regression	0.666667	0.4
1	Linear Discriminant Analysis	0.833333	0.5
2	Decision Tree	0.333333	0.2
3	Random Forest	0.500000	0.3
4	K-Nearest Neighbor	0.833333	0.5
5	Naive Bayes	0.500000	0.7
6	Suport Vector Machine	0.833333	0.5

Lampiran 9. Kode Algoritma K-NN dengan K Optimal

```
[ ]: #import google drive
     from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

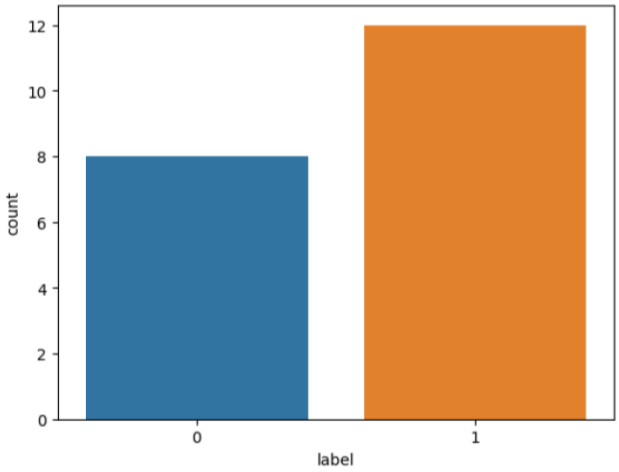
```
[ ]: #import library
     import pandas as pd
     #read csv data
     df_eeg = pd.read_csv('/content/drive/MyDrive/eeg_data_stim_alfatihah/
       ↪combined_data_eeg_labeled_one_row.csv')
     df_eeg.head()
```

```
[ ]:          0         1         2         3         4         5         6  \
     0  0.000319  0.000319  0.000319  0.000319  0.000319  0.000319  0.000319
     1  0.001285  0.001285  0.001285  0.001285  0.001285  0.001285  0.001285
     2  0.001803  0.001803 -0.002239 -0.000218  0.003824 -0.002239 -0.000218
     3  0.004527  0.004527  0.004527  0.004527  0.004527  0.004527  0.004527
     4  0.021220  0.004357  0.004357  0.004357  0.004357  0.004357  0.004357

               7         8         9  …     107991    107992     107993  \
     0  0.000319  0.000319  0.000319  …  -3.185368 -3.810312   0.556671
     1  0.001285  0.001285  0.001285  … -14.106295 -3.829575   1.838489
     2 -0.000218 -0.000218  0.003824  …   9.587521 -1.057214 -11.988934
     3  0.004527  0.004527  0.004527  … -12.162396  5.730739 -11.524449
     4  0.012789  0.000141  0.038084  …   0.038084  0.004357   0.004357

          107994     107995    107996     107997    107998    107999  label
     0 -6.441552 -13.401670  8.301778  -4.513373 -0.140674 -0.186402      0
     1 -4.319049  -6.799945 -1.008953  -9.703264 -0.615586  0.121977      0
     2 -0.228594 -11.006716 -2.463847   5.565683  0.282725  0.369629      0
     3  7.358779  -9.911719  9.354277 -12.846275  0.708821 -0.220030      1
     4  0.004357   0.004357  0.004357   0.004357  0.000141  0.000141      0

     [5 rows x 108001 columns]
```

```
[ ]: #look for data information
     df_eeg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Columns: 108001 entries, 0 to label
dtypes: float64(108000), int64(1)
memory usage: 16.5 MB
```

```
[ ]: #exploratory data analysis (EDA)
     import seaborn as sns
     import matplotlib.pyplot as plt
     plt.figure()
     sns.countplot(x='label', data=df_eeg)
     plt.show()
```

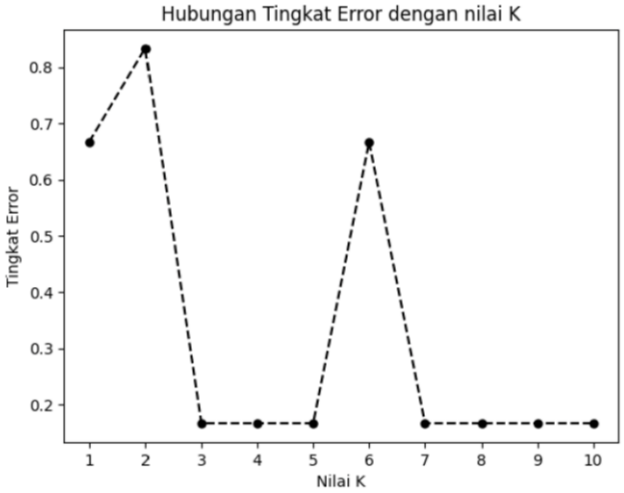



```
[ ]: #featur variable (X) and target variable(y)
     X = df_eeg.iloc[:,0:108000]
     y = df_eeg.iloc[:,108000]
```

```
[ ]: #splitting data
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
      ↪random_state=12)
```

```
[ ]: #train data
     import numpy  as np
     from sklearn.neighbors import KNeighborsClassifier
     error_rating = []
     for k in range(1, 11):
       knn = KNeighborsClassifier(n_neighbors=k, metric='euclidean')
       knn.fit(X_train, y_train)
       y_pred = knn.predict(X_test) #predict
       error_rating.append(np.mean(y_pred != y_test))
```

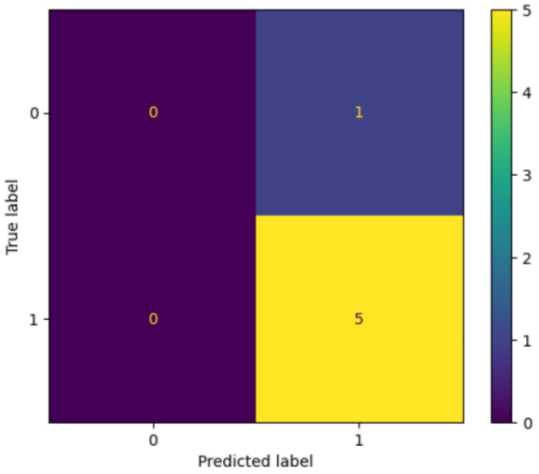
```
[ ]: #plotting
plt.plot(range(1,11), error_rating, marker="o", markerfacecolor="black",
         linestyle="dashed", color="black", markersize=5)
plt.title('Hubungan Tingkat Error dengan nilai K')
plt.xlabel('Nilai K')
plt.ylabel('Tingkat Error')
plt.xticks(range(1,11))
plt.show()
```




```
[ ]: #optimal k-value
knn_optimal = KNeighborsClassifier(n_neighbors=10, metric='euclidean')
knn_optimal.fit(X_train, y_train)
y_pred_opt = knn_optimal.predict(X_test)
```

```
[ ]: #convusion matrix plot
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, y_pred_opt, labels=knn.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=knn.classes_)
disp.plot()
```

```
[ ]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
     0x794741a59180>
```




```
[ ]: #value acc
from sklearn.metrics import accuracy_score
acc = (accuracy_score(y_test, y_pred_opt)*100)
print("Nilai akurasi model klasifikasi k-nn : %.0f" % acc +"%")

#report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_opt))
```

```
Nilai akurasi model klasifikasi k-nn : 83%
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.83      1.00      0.91         5

    accuracy                           0.83         6
   macro avg       0.42      0.50      0.45         6
weighted avg       0.69      0.83      0.76         6
```

RIWAYAT HIDUP

A. Identitas Diri

1. Nama Lengkap : Thooriq Nur Ali
2. Tempat, Tanggal Lahir : Kab. Semarang, 25 April 2001
3. Alamat Rumah : Ling. Sikunir RT 07 RW 07 Bergas Lor, Kec. Bergas Kab. Semarang
4. HP : 08999333005
5. E-mail : thooriq.na@gmail.com

B. Riwayat Pendidikan

1. Sekolah Dasar (SD) Islam Istiqomah Ungaran
2. Sekolah Menengah Pertama (SMP) Negeri 1 Bergas
3. Sekolah Mengengah Atas (SMA) Negeri 1 Bergas

Semarang, 22 September 2023

Thooriq Nur Ali
NIM.1908096004