

PERBANDINGAN PERFORMANSI METODE SUPPORT VECTOR MACHINE DAN XGBOOST PADA DATASET STRES DAN PENGARUHNYA

Laporan ini dibuat untuk memenuhi kelulusan
Matakuliah Program Tugas Akhir



Oleh:

1.18.4.003

Dinda Anik Masruro

**PROGRAM DIPLOMA IV TEKNIK INFORMATIKA
POLITEKNIK POS INDONESIA
BANDUNG
2022**

COMPARISON PERFORMANCE OF SUPPORT VECTOR MACHINE AND XGBOOST METHODS ON STRESS DATASETS AND THEIR EFFECTS

This report is made to meet the graduation requirements

For the thesis program courses



Created By:

1.18.4.003

Dinda Anik Masruro

PROGRAM DIPLOMA IV TEKNIK INFORMATIKA

POLITEKNIK POS INDONESIA

BANDUNG

2022

ABSTRAK

Pengenalan pengaruh bertujuan untuk mendeteksi keadaan afektif seseorang berdasarkan pengamatan, dengan tujuan misalnya meningkatkan interaksi manusia-komputer. Stres jangka panjang diketahui memiliki implikasi parah pada kesejahteraan, yang memerlukan sistem pemantauan stres yang berkelanjutan dan otomatis. Namun, komunitas komputasi afektif tidak memiliki kumpulan data standar yang umum digunakan untuk deteksi stres yang dapat dikenakan yang a) menyediakan data berkualitas tinggi multimodal, dan b) menyertakan beberapa status afektif. Oleh karena itu, peneliti memperkenalkan WESAD, kumpulan data baru yang tersedia untuk umum untuk stres yang dapat dikenakan dan deteksi pengaruh. Kumpulan data multimodal ini menampilkan data fisiologis dan gerakan, direkam dari perangkat yang dikenakan di pergelangan tangan dan dada, dari 15 subjek selama studi laboratorium. Modalitas sensor berikut termasuk: nadi volume darah, elektrokardiogram, aktivitas elektrodermal, elektromiogram, pernapasan, suhu tubuh, dan percepatan tiga sumbu. Selain itu, dataset menjembatani kesenjangan antara studi lab sebelumnya tentang stres dan emosi, dengan memuat tiga keadaan afektif yang berbeda (netral, stres, hiburan). Selain itu, laporan diri subjek, yang diperoleh dengan menggunakan beberapa kuesioner yang telah ditetapkan, terdapat dalam kumpulan data. Selanjutnya, tolok ukur dibuat pada kumpulan data, menggunakan fitur terkenal dan metode *machine learning*. Mengingat masalah klasifikasi tiga kelas (baseline vs stres vs hiburan), peneliti menggunakan dua metode penelitian yaitu *Support Vector Machine* dan *Xgboost*. Dari hasil *Dataset stress dan pengaruhnya* dengan menggunakan algoritma *Support Vector Machine* (SVM) hasil *accuracy* sebesar 98%, *F1 score* sebesar 98%, dan *UAR* sebesar 98%, sedangkan menggunakan *Xgboost* hasil *accuracy* sebesar 98%, *F1 score* sebesar 97%, dan *UAR* sebesar 97%. Maka dari hasil penelitian ini dapat disimpulkan bahwa metode *Support Vector Machine* (SVM) pada *dataset stress dan pengaruhnya* memiliki nilai akurasi yang lebih unggul

Kata kunci: *Machine Learning, Support Vector Machine, Xgboost, Deteksi Stres*

ABSTRACT

Influence recognition aims to detect a person's affective state based on observations, with the aim of, for example, increasing human-computer interaction. Long-term stress is known to have severe implications on well-being, which requires continuous and automated stress monitoring systems. However, the affective computing community lacks a commonly used standardized data set for wearable stress detection that a) provides multimodal high-quality data, and b) includes multiple affective states. Therefore, the researcher introduces WESAD, a new publicly available dataset for wearable stress and affect detection. This multimodal data set features physiological and movement data, recorded from wrist and chest-worn devices, of 15 subjects during a laboratory study. The following sensor modalities include: pulse blood volume, electrocardiogram, electrodermal activity, electromyogram, respiration, body temperature, and three-axis acceleration. In addition, the dataset bridges the gap between previous lab studies of stress and emotion, by containing three different affective states (neutral, stress, entertainment). In addition, subject self-reports, which were obtained using several predefined questionnaires, were included in the data set. Next, benchmarks are created on the data set, using well-known features and machine learning methods. Considering the problem of classifying three classes (baseline vs stress vs entertainment), the researcher uses two research methods, namely Support Vector Machine and Xgboost. From the results of the stress dataset and its effect using the Support Vector Machine (SVM) algorithm, the accuracy is 98%, F1 score is 98%, and UAR is 98%, while using Xgboost the accuracy is 98%, F1 score is 97%, and UAR by 97%. So from the results of this study it can be concluded that the Support Vector Machine (SVM) method on the stress dataset and its effects has a superior accuracy value.

Keywords: Machine Learning, Support Vector Machine, Xgboost, Stress Detection

KATA PENGANTAR

Puji dan syukur saya panjatkan kepada Allah Swt. atas ridanya saya dapat menyelesaikan laporan Tugas Akhir ini. Adapun judul Tugas Akhir saya adalah “Perbandingan Performansi Metode *Support Vector Machine* dan *Xgboost* pada Dataset Stress dan Pengaruhnya”.

Laporan ini diajukan untuk memenuhi syarat kelulusan mata kuliah Tugas Akhir di Diploma IV Politeknik Pos Indonesia. Tidak dapat disangkal bahwa butuh usaha yang keras dalam penyelesaian pengerjaan laporan ini. Namun, karya ini tidak akan selesai tanpa orang-orang tercinta di sekeliling saya yang mendukung dan membantu. Terima kasih saya sampaikan kepada:

1. Kedua orang tua tercinta yang senantiasa memberikan dukungan dan doanya.
2. Dr. Ir. Agus Purnomo, M.T. selaku Direktur Politeknik Pos Indonesia.
3. M.Yusril Helmi Setyawan, S. Kom., M. Kom. selaku Ketua Program Studi DIV Teknik Informatika.
4. Roni Andarsyah, S.T., M.Kom., SFPC selaku Koordinator Tugas Akhir
5. Rolly Maulana Awangga, S.T., M.T.,CAIP.,SFPC selaku Pembimbing I Tugas Akhir.
6. Roni Habibi, S.Kom., M.T., SFPC selaku dosen pembimbing II Tugas Akhir
7. Rolly Maulana Awangga, S.T., M.T.,CAIP.,SFPC selaku dosen wali kelas D4 TI 4 C
8. Serta semua pihak yang tidak dapat penulis sebutkan satu persatu.

Akhir kata penulis berharap semoga laporan ini dapat bermanfaat bagi seluruh mahasiswa khususnya bagi penulis. Penulis sadar bahwa proposal ini masih jauh dari kata sempurna. Oleh karena itu penulis sangat mengharapkan adanya kritik dan saran yang membangun dari para pembaca untuk kesempurnaan proposal ini. Semoga Allah SWT selalu memberikan rahmat dan hidayah-Nya kepada kita semua. Aamiin.

Bandung, 14 Agustus 2022



Dinda Anik Masruro

1.18.4.003

DAFTAR ISI

ABSTRAK	i
<i>ABSTRACT</i>	ii
KATA PENGANTAR.....	iii
DAFTAR ISI.....	iv
DAFTAR GAMBAR	vi
DAFTAR TABEL	vii
DAFTAR SIMBOL.....	viii
BAB I PENDAHULUAN	I-1
1.1 Latar Belakang.....	I-1
1.2 Identifikasi Masalah	I-3
1.3 Tujuan dan Manfaat.....	I-3
1.4 Ruang Lingkup	I-4
1.5 Sistematika Penulisan	I-4
BAB II TINJAUAN PUSTAKA.....	II-6
2.1 State of The Art (SOTA)	II-6
2.2 Landasan Teori	II-7
2.2.1 Stres	II-7
2.2.2 Machine Learning	II-7
2.2.3 Support Vector Machine (SVM)	II-8
2.2.4 Xgboost	II-9
2.2.5 TensorFlow.....	II-9
2.2.6 Python.....	II-9
2.3 Penelitian Terkait.....	II-11
BAB III METODE PENELITIAN.....	III-16
3.1 Diagram Alur Metodologi Penelitian	III-16
3.2 Tahapan-Tahapan Diagram Alur Metodologi Penelitian	III-17
3.2.1 Identifikasi Masalah	III-17
3.2.2 Studi Literatur	III-17
3.2.3 Pengumpulan Data	III-17
3.2.4 Pemodelan Metode.....	III-17
3.2.5 Hasil Evaluasi.....	III-17

3.3	Indikator Capaian Penelitian	III-17
BAB IV HASIL DAN PEMBAHASAN		IV-19
4.1	Pengumpulan Data.....	IV-19
4.2	Hasil Penelitian.....	IV-21
4.2.1	Import Library	IV-21
4.2.2	Pendekatan <i>Classical</i>	IV-22
4.2.3	Pendekatan <i>Fully supervised</i>	IV-24
4.2.4	Pendekatan <i>Self supervised + transfer learning</i>	IV-29
4.2.5	Evaluasi Model.....	IV-39
BAB V PENUTUP.....		V-40
5.1	Kesimpulan.....	V-40
5.2	Saran	V-40
DAFTAR PUSTAKA		V-41
LAMPIRAN		V-45
Lampiran 1. Kambing		V-45
Lampiran 2. Plagiarism		V-46
Lampiran 3. Surat Pernyataan		V-47






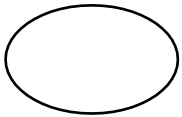
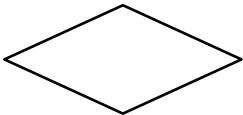
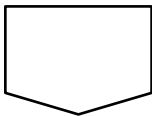
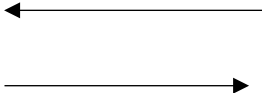
DAFTAR GAMBAR

Gambar 3. 1 Alur Metodologi Penelitian (Reksi & Ernawati, 2020).....	III-16
Gambar 4. 1 Import Library	IV-21
Gambar 4. 2 Dataset Stres	IV-22
Gambar 4. 3 Normalize Data.....	IV-22
Gambar 4. 4 Evaluasi Data Training dan Data Testing Support Vector Machine	IV-23
Gambar 4. 5 Evaluasi Data Training dan Data Testing Xgboost	IV-23
Gambar 4. 6 Epoch Support Vector Machine	IV-26
Gambar 4. 7 Grafik Support Vector Machine	IV-26
Gambar 4. 8 Hasil Evaluasi Support Vector Machine	IV-27
Gambar 4. 9 Epoch Xgboost	IV-28
Gambar 4. 10 Grafik Xgboost	IV-29
Gambar 4. 11 Hasil Epoch Support Vector Machine.....	IV-31
Gambar 4. 12 Grafik Support Vector Machine	IV-31
Gambar 4. 13 Hasil Evaluasi Epoch Akhir	IV-33
Gambar 4. 14 Grafik Akhir Support Vector Machine.....	IV-33
Gambar 4. 15 Hasil Epoch Support Vector Machine.....	IV-35
Gambar 4. 16 Grafik Support Vector Machine	IV-35
Gambar 4. 17 Hasil Evaluasi Epoch Akhir	IV-37
Gambar 4. 18 Grafik Akhir Support Vector Machine.....	IV-37
Gambar 4. 19 Diagram Perbandingan 3 Pendekatan.....	IV-38

DAFTAR TABEL

Tabel 2.3. 1 Penelitian Terkait	II-11
Tabel 3. 1 Indikator Capaian	III-17
Tabel 4. 1 Classical	IV-19
Tabel 4. 2 Fully-supervised	IV-20
Tabel 4. 3 Self-supervised	IV-21
Tabel 4. 4 Hasil Evaluasi Model	IV-39

DAFTAR SIMBOL

No	Simbol	Nama Simbol	Keterangan
1		Input / Output	Sebagai media masukan dan keluaran data
2		Process	Menggambarkan proses transformasi dari data masuk menjadi keluar
3		Predifined Process	Menggambarkan proses yang masih berisi proses lain didalamnya
4		Preparation	Sebagai pemberian nilai awal
5		Start/End	Sebagai awal dan akhir program
6		Connector	Sebagai Penghubung
7		Desicion	Sebagai media untuk melakukan pemilihan
8		Off-Page Connector	Sebagai penghubung beda halamam
9		Data Flow	Simbol yang menggambarkan arus data yang mengalir

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengenalan pengaruh bertujuan untuk mendeteksi keadaan afektif seseorang berdasarkan pengamatan, dengan tujuan misalnya meningkatkan interaksi manusia-komputer. Stres jangka panjang diketahui memiliki implikasi parah pada kesejahteraan, yang memerlukan sistem pemantauan stres yang berkelanjutan dan otomatis. Namun, komunitas komputasi afektif tidak memiliki kumpulan data standar yang umum digunakan untuk deteksi stres yang dapat dikenakan yang a) menyediakan data berkualitas tinggi multimodal, dan b) menyertakan beberapa status afektif. Oleh karena itu, peneliti memperkenalkan WESAD, kumpulan data baru yang tersedia untuk umum tentang stres dan deteksi pengaruhnya.

Stres psikologis dihadapi oleh manusia sebagai respons terhadap tantangan fisik, mental, atau emosional yang diberikan kepada mereka oleh lingkungan. Sinyal fisiologis dapat dengan mudah diukur menggunakan perangkat yang dapat dikenakan dan merupakan indikator stres yang baik, sehingga cocok untuk pemantauan stres berkelanjutan (Marković et al., 2019). Arsitektur pembelajaran mendalam baru-baru ini berkinerja baik untuk tugas-tugas seperti itu, karena kemampuan ekstraksi fitur otomatisnya, berbeda dengan model pembelajaran mesin konvensional yang mengamankan pengetahuan domain untuk membuat fitur heuristik dangkal (Ragav & Gudur, 2020). Ada minat khusus dalam membawa pembelajaran mendalam ke perangkat yang dapat dikenakan untuk memberikan tekanan waktu nyata dan memengaruhi pemantauan, sambil menjaga data pengguna yang sensitif.

Dengan perkembangan zaman yang semakin canggih pendeteksi stress tersebut dapat dideteksi menggunakan *machine learning*. *Machine learning* sendiri memainkan peran penting di berbagai area yang efisien untuk pemrosesan data, salah satunya adalah identifikasi pendeteksi stress (Trivedi et al., 2020). Algoritma yang digunakan pada penelitian ini yaitu *Xgboost* dan *Support Vector Machine* (SVM). Algoritma *Xgboost* merupakan salah satu algoritma prediksi paling populer untuk *gradient boosting machine* (GBM) dan memiliki tingkat *accuracy* yang lebih baik (Jiang et al., 2019), sedangkan Algoritma *Support Vector Machine* (SVM) juga memiliki tingkat *accuracy* yang baik dibandingkan beberapa algoritma lainnya. Oleh karena itu pada penelitian ini, peneliti melakukan deteksi terhadap stress dengan

melakukan perbandingan hasil dari dua metode yaitu algoritma *machine learning* yang digunakan pada penelitian ini adalah *Support Vector Machine* (SVM) dan *Xgboost*.

Penelitian terdahulu juga dilakukan dengan teknologi piranti pendukung keamanan dan pengawasan anak saat ini menjadi ranah penelitian aktif yang banyak dikaji oleh berbagai peneliti di dunia seiring dengan kebutuhan komunitas masyarakat terhadap teknologi ini yang memunculkan berbagai inovasi yang lahir dari kreatifitas para peneliti beberapa diantaranya seperti penelitian dari A. Saranya et al. yang merancang sistem pengawasan anak menggunakan smart tag yang dilekatkan pada seragam anak dan terhubung pada jaringan nirkabel yang berfungsi untuk memberi peringatan kepada orang tua apabila anak yang diawasi keluar dari zona yang telah ditentukan dan kemudian sistem dapat memberi informasi posisi anak secara akurat menggunakan bantuan GPS (Chyan & Kasmara, 2021). Kemudian penelitian lain dari J. Megha et al. yang menghasilkan sistem pengawasan anak sekolah. Produk dari penelitian ini adalah suatu sistem yang mendeteksi keberadaan anak sekolah mulai dari bus sekolah selanjutnya tiba disekolah hingga kembali ke rumah Sistem menggunakan teknologi *NFC* (*Near field Communication*) yang berkomunikasi dengan RFID yang melekat di kartu siswa untuk memantau keberadaan anak tersebut apabila sistem tidak dapat mengkonfirmasi keberadaan anak maka sistem akan mengirimkan laporan kepada pengawas. Penelitian lain oleh S.K. Punjabi menghasilkan sebuah *device* portabel dengan jaringan GSM yang dilengkapi oleh saklar tekan. Perangkat ini bekerja dengan cara diremas ketika anak menghadapi bahaya, sensor tekanan pada perangkat akan mendeteksi adanya tekanan yang diberikan dan kemudian akan mengirimkan SMS kepada nomor kontak yang tersimpan pada profile berupa lokasi subjek yang dibaca melalui modul GPS pada perangkat. Terkait dengan teknologi penyebaran informasi, penelitian dari Utari dan Triana mengembangkan sebuah sistem informasi monitoring siswa yang berfungsi memberi laporan ke orangtua mengenai kinerja akademik anak mereka masing-masing disekolah melalui SMS gateway (Utari & Triana, 2019) dan juga penelitian dari Chyan yang menghasilkan sistem monitoring untuk lansia berbasis IOT mampu mengirimkan informasi ke berbagai media baik SMS, email, dan media sosial lainnya bila sesuatu yang membahayakan menimpa lansia dalam pengawasan (Chyan & Kasmara, 2021).

Selanjutnya penelitian mengenai deteksi stress pada anak sebagian besar menggunakan sinyal audio untuk mendeteksi tangisan yang diasosiasikan sebagai keadaan stress pada anak, kekurangan metode ini apabila digunakan sebagai satu-satunya parameter adalah kesulitan

dalam mengidentifikasi anak yang menangis terutama apabila terdapat banyak anak yang berkumpul dalam sebuah ruangan. Beberapa diantaranya seperti penelitian dari Limantoro, dkk yang merancang aplikasi untuk mendeteksi tangisan bayi dan mencoba untuk menginterpretasikan masalah yang dihadapi bayi berdasarkan tangisan tersebut. Penelitian dengan topik yang sama juga dilakukan oleh Prasetyo, dkk dengan mengklasifikasi masalah bayi berdasarkan tangisannya menurut standar *Dunstan Baby Language* dengan menggunakan metode *K-Nearest Neighbor* (KNN) (Prasetyo, 2020). Beberapa metode deteksi stress memanfaatkan parameter biologis yang diterapkan pada orang dewasa juga dapat diterapkan pada subjek anak untuk meningkatkan akurasi deteksi. Kebutuhan perangkat sensor khusus yang harus ditempelkan di bagian tubuh juga menjadi masalah tersendiri terlebih bila alat tersebut berat atau mengganggu kenyamanan pemakai. Penelitian dari Bakti dan Wardati menggunakan suhu tubuh, denyut jantung dan *Galvanic Skin Response* untuk mendeteksi stress pada manusia dewasa dengan membandingkannya dengan parameter yang sudah ditetapkan akan tetapi dari sisi akurasi masih rendah karena belum ada suatu metode pembelajaran mesin yang diimplementasikan pada algoritma yang digunakan (Bakti & Wardati, 2019; Yuanika et al., 2021).

Dari semua penelitian terdahulu yang dijadikan landasan dari penelitian makan algoritma algoritma *machine learning* digunakan untuk menganalisis hasil deteksi stres dan menganalisis hasilnya adalah *Support Vector Machine* (SVM) dan *Xgboost*. Oleh karena itu penelitian tugas akhir ini berjudul “Perbandingan Performansi Metode *Support Vector Machine* (SVM) dan *Xgboost* Pada Dataset Stres dan Pengaruhnya”.

1.2 Identifikasi Masalah

Adapun identifikasi masalah pada penelitian ini sebagai berikut,

1. Bagaimana cara mendeteksi stress menggunakan algoritma *Support Vector Machine* (SVM) dan *Xgboost*?
2. Bagaimana hasil dari model klasifikasi algoritma *Support Vector Machine* (SVM) dan *Xgboost* untuk mendeteksi stress?

1.3 Tujuan dan Manfaat

Adapun tujuan pada penelitian ini sebagai berikut,

1. Mengetahui bagaimana mendeteksi stress dengan menerapkan algoritma *Support Vector Machine* (SVM) dan *Xgboost*.

2. Menganalisis model klasifikasi algoritma *Support Vector Machine* (SVM) dan *Xgboost* yang menghasilkan akurasi yang optimal dalam mendeteksi stress.

Adapun manfaat pada penelitian ini sebagai berikut,

1. Memberikan informasi jumlah yang mengalami stress.
2. Mengetahui hasil paling baik dari hasil akurasi yang optimal dalam mendeteksi stress.

1.4 Ruang Lingkup

Adapun ruang lingkup pada penelitian ini sebagai berikut,

1. Pada penelitian ini data yang digunakan didapatkan dari dataset pendeteksi stress.
2. Periode waktu 3 bulan.
3. Bahasa pemograman menggunakan *Python*.
4. Software yang digunakan yaitu *Jupyter* atau *Google Colab*.

1.5 Sistematika Penulisan

Berdasarkan latar belakang dan perumusan masalah diatas, maka penyusunan laporan ini dibuat dalam suatu sistematika yang terdiri dalam lima BAB, yaitu:

BAB I PENDAHULUAN

Bab ini berisi penjelasan terkait dengan State of The Art (SOTA) yang menjelaskan mengenai pemaparan teori umum dengan topik yang dibahas secara global dan mengaitkan dengan referensi yang ada. Identifikasi masalah menjelaskan mengenai masalah dalam judul penelitian “Perbandingan Performansi Metode *Support Vector Machine* (SVM) dan *Xgboost* Pada Dataset Pendeteksi Stress” dengan pendekatan *machine learning* dan memberikan solusi atas masalah tersebut. Tujuan menjelaskan tentang solusi dari masalah yang ada. Ruang lingkup menjelaskan mengenai batasan dalam pemodelan dan aplikasi tersebut. Serta sistematika penulisan menjelaskan tentang isi laporan Tugas Akhir.

BAB II LANDASAN TEORI

Bab ini berisi penjelasan mengenai konsep dasar dan pendukung dari sistem yang akan dibangun dengan menggunakan metode tertentu, antara lain State of The Art (SOTA), diagram alur metodologi penelitian, dan penelitian sebelumnya yang berhubungan dengan tema yang diambil.

BAB III METODOLOGI PENELITIAN

Bab ini berisi penjelasan diagram alur metodologi penelitian beserta tahapan-tahapan diagram alur penelitian untuk menyelesaikan penelitian yang sedang dilakukan sehingga bisa mencapai tujuan yang diharapkan.

BAB IV HASIL PENELITIAN

Bab ini akan menjelaskan mengenai tahapan proses pengujian yang dilakukan dan analisis hasil pengujian untuk mendapatkan kesimpulan dari setiap proses pengujian.

BAB V KESIMPULAN

Pada bab ini akan dijelaskan mengenai kesimpulan dari seluruh kegiatan yang telah dilakukan.

BAB II

TINJAUAN PUSTAKA

2.1 State of The Art (SOTA)

Stres didefinisikan sebagai reaksi terhadap situasi lingkungan yang merugikan yang menantang kemampuan adaptif khas seperti yang dirasakan oleh individu. Meskipun stres positif (eustress) membantu individu untuk tetap fokus menghadapi kesulitan, stres negatif (distress) menyebabkan aktivasi sumbu HPA (hipotalamus-hipofisis-adrenokortikal). Aktivasi HPA axis yang berkepanjangan dapat menyebabkan gangguan fisiologis dan psikologis. Stres psikologis juga ditemukan mempengaruhi proses fisiologis dan memiliki efek negatif pada kinerja kerja sehari-hari dan diperkirakan mempengaruhi perekonomian nasional (Rajdeep Kumar et al., 2020). Pemantauan tingkat stres negatif dapat memberikan informasi yang berguna untuk mengidentifikasi stresor dan memberikan kesempatan untuk mengadopsi tindakan pencegahan yang diperlukan dalam mencegah gangguan yang dihasilkan. Dua efek berbeda dari stres negatif didefinisikan sebagai: (i) stres *fisiologis* atau "*objektif*"; dan (ii) stres psikologis atau stres "*subyektif*" yang juga dikenal sebagai stres yang dirasakan. Stres objektif dicerminkan oleh perubahan ukuran fisiologis seperti peningkatan tekanan darah, peningkatan denyut jantung, dan peningkatan kadar kortisol. Stres subjektif adalah persepsi tentang benar atau tidaknya suatu situasi sebagai stres oleh seorang individu. Metode yang paling umum untuk mengukur stres yang dirasakan adalah dengan menggunakan kuesioner stres seperti DASS 21 (*Depresi, Anxiety and Stress Scale - 21 Items*), STAI (*State-Trait Anxiety Inventory*), dan POMS (*Profile of Mood States*). Dua ukuran fisiologis utama untuk stres meliputi: (i) Kortisol (hormon stres) dan (ii) pengukuran sinyal fisiologis seperti GSR (*Galvanic Skin Response*), EKG (*Electrocardiogram*), dan EEG (*Electroencephalogram*). Dalam, penelitian ini telah membahas berbagai ukuran fisiologis stres dan teknologi yang terkait dengan pengukuran metrik stres. Lebih lanjut akan di jelaskan (Yuanika et al., 2021), peneliti telah memberikan gambaran tentang berbagai sensor dan perangkat yang tersedia secara komersial untuk mengukur stres. Oleh karena itu, peneliti akan mengeksplorasi berbagai pendekatan berbasis *machine learning* untuk deteksi stress.

Data yang digunakan pada penelitian ini dataset tentang pendeteksi stress, dalam melakukan proses deteksi stress ini dilakukan dengan menggunakan bahasa pemrograman *python*. Bahasa pemrograman *python* merupakan salah satu bahasa pemrograman tinggi yang dapat melakukan eksekusi sejumlah intruksi multi guna secara langsung dengan *object oriented*

programming serta menggunakan semantik dinamis untuk memberikan tingkat keterbacaan *syntax* (Vallat, 2018). Deteksi pada penelitian ini dilakukan dengan menggunakan algoritma *Xgboost*. *Xgboost* digunakan untuk memprediksi karena *Xgboost* adalah salah satu algoritma prediksi paling populer untuk *gradient boosting machine* (GBM) (Jiang et al., 2019). Keunggulan dari algoritma ini ialah klasifikasi yang dilakukan dengan cepat dan dengan akurasi yang tinggi. Pada penelitian ini selain *Xgboost* juga menggunakan algoritma *Support Vector Machine* (SVM), *Support Vector Machine* (SVM) merupakan teknik pada *machine learning* yang populer digunakan untuk klasifikasi teks serta memiliki performa yang baik pada banyak domain (Ainurrochman et al., 2020). Penggunaan kedua algoritma ini digunakan untuk membandingkan hasil akurasi dari keduanya dengan keunggulan yang berbeda-beda dari keduanya.

2.2 Landasan Teori

2.2.1 Stres

Menurut definisi umum stres ditandai dengan perubahan reaksi psikologis dari keadaan tenang ke keadaan yang emosional. Menurut beberapa peneliti stres dibedakan menjadi *eustress* dan *distress*. *Eustress* mengarah ke keadaan emosi yang baik seperti gembira dan bersemangat sementara *distress* mengarah ke emosi negatif, tetapi umumnya istilah stres lebih sering digunakan untuk menggambarkan kondisi emosi yang negatif seperti marah, sedih, gelisah, takut, sakit dan gugup (Gedam & Paul, 2021). Banyak hasil penelitian yang menyimpulkan kaitan yang erat antara keadaan stres dengan penurunan kemampuan mengambil keputusan, penurunan kewaspadaan akan situasi lingkungan serta penurunan kinerja kognitif (Di Martino & Delmastro, 2020).

2.2.2 Machine Learning

Artificial Intelligence (AI) atau Kecerdasan buatan dewasa ini berkembang begitu pesat. Semua hal berkaitan dengan kecerdasan buatan (Ramdhani, 2019) yang terus dikembangkan sesuai dengan perkembangan zaman.

Dalam Kecerdasan buatan memiliki banyak cabang salah satunya *Machine Learning* (ML), Sebenarnya cabang dari kecerdasan buatan itu ada tujuh yaitu *Machine Learning* itu sendiri, *natural language processing*, *expert system*, *vision*, *speech*, *planning* dan *robotics*. Dalam kecerdasan buatan diberikan cabang cabang tersebut sebenarnya hanya untuk mempersempit lingkup dalam pengembangan maupun dalam pembelajaran AI (Brunton et al., 2020) hal ini didasarkan karena kecerdasan buatan memiliki lingkup yang luas.

Dikembangkannya *Machine Learning* didasarkan disiplin ilmu lainnya seperti statistika, matematika dan data mining. Dimana mesin dikembangkan untuk dapat belajar secara mandiri tanpa perlu diarahkan oleh penggunanya dan mesin dapat belajar dengan menganalisa data tanpa perlu memprogram ulang atau memberi perintah ulang. Dasar-dasar *Machine Learning* dan konsep *Machine Learning* pertama ditemukan oleh beberapa ilmuwan matematika yaitu Adrien Marie Legendre, Thomas Bayes dan Andrey Markov pada tahun 1920-an. Dan mulai dari situlah *Machine Learning* mulai dikembangkan. Penerapan *Machine Learning* yang terkenal pada tahun 1996 adalah *Deep Blue* yang dibuat oleh IBM. *Machine Learning* ini dikembangkan untuk bisa belajar serta bermain catur. Pada masa itu *Deep Blue* diuji coba untuk bermain catur dengan juara catur profesional dengan pemenangnya adalah *Machine Learning deep blue*. Manfaat *Machine Learning* pada kehidupan kita sehari-hari dapat kita rasakan bersama (Roscher et al., 2020). Seperti pada kunci layar hp yang menggunakan *face unlock* untuk membuka kunci layar Smartphone. Atau bahkan iklan iklan yang disesuaikan dengan trend saat ini. itu merupakan hasil dari pengolahan data menggunakan *Machine Learning*.

2.2.3 Support Vector Machine (SVM)

Support vector machine (SVM) pertama kali diperkenalkan pada tahun 1992 di Annual Workshop on Computational Learning Theory. *Support vector machine* (SVM) dikembangkan oleh Boser, Guyon dan Vapnik (Ramadhan & Setiawan, 2019). Kelebihan dari algoritma ini yaitu mampu mengidentifikasi *hyperplane* terpisah dengan memaksimalkan *margin* antara dua kelas yang berbeda. Namun algoritma *Support vector machine* (SVM) juga memiliki kekurangan yaitu pada masalah pemilihan fitur yang sesuai (Ratino et al., 2020). *Support Vector Machine* (SVM) dengan menggunakan kernel linier yang memberikan kinerja lebih baik dalam bentuk yang lebih rendah pada himpunan data. Kernel mendefinisikan kesamaan atau ukuran jarak antara data baru dan vektor pendukung (Arafin Mahtab et al., 2018). Konsep klasifikasi dengan *Support Vector Machine* (SVM) adalah mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua kelas data. *Support Vector Machine* (SVM) mampu bekerja pada dataset yang berdimensi tinggi dengan menggunakan kernel trik (Perdana & Fauzi, 2017). Dalam algoritma ini, setiap item data diplot sebagai titik dalam ruang n-dimensi (dimana n adalah jumlah fitur yang dimiliki) dengan nilai setiap fitur menjadi nilai koordinat tertentu. Kemudian, dilakukan klasifikasi dengan menemukan *hyperplane* yang membedakan dua kelas dengan sangat baik ((Al Amrani, Lazaar, El Kadiri., 2018). *Support vector machine* (SVM) telah banyak digunakan karena dapat menyelesaikan

masalah nonlinier dengan mengubahnya menjadi kuadrat pemrograman. Selain itu, solusi *Support vector machine* (SVM) unik dan optimal secara global (Ren et al., 2019). Algoritma *Support vector machine* (SVM) dapat diformulasikan seperti berikut ini :

$$D_{ij} = y_i y_j (K(\vec{x_i} \cdot \vec{x_j}) + \lambda^2)$$

Dimana:

D_{ij} = elemen matriks data ke-ij

y_i = kelas atau label data ke-i

y_j = kelas atau label data ke-j

λ^2 = nilai bias

$K(\vec{x_i} \cdot \vec{x_j})$ = fungsi kernel

2.2.4 Xgboost

Algoritma *XGBoost* digunakan untuk memprediksi karena *XGBoost* adalah salah satu algoritma prediksi paling populer untuk gradient boosting machine (GBM) (Jiang et al., 2019). *Xgboost* telah banyak digunakan di industri karena kinerjanya yang tinggi dalam pemecahan masalah dan persyaratan minimal untuk rekayasa fitur. Kedua, dibandingkan dengan algoritma *deep learning*, *XGBoost* diakui lebih mudah digunakan untuk kumpulan data kecil yang berjalan di CPU (Dong et al., 2020).

2.2.5 TensorFlow

Tensorflow adalah *open source library* untuk *machine learning* yang di *release* oleh Google yang mendukung beberapa bahasa pemrograman (Nugroho et al., 2020). Dalam proses *Transfer Learning*, *Tensorflow* berperan untuk memproses *Inception-v3* Model untuk di training ulang menggunakan data yang baru dan kemudian menghasilkan *classifier* dengan komputasi yang cepat dan akurasi yang baik. *Tensorflow* dapat digunakan pada semua sistem operasi.

2.2.6 Python

Python adalah bahasa scripting tingkat tinggi, ditafsirkan, interaktif, dan berorientasi objek yang kuat yang dibuat oleh Guido Van Rossum pada akhir 1980-an. *Python* adalah bahasa yang sangat cocok untuk programmer tingkat pemula dan mendukung pengembangan berbagai aplikasi mulai dari pemrosesan teks sederhana hingga browser www hingga

pengembangan game (Melo et al., 2019). *Python* saat ini adalah bahasa pemrograman dengan pertumbuhan tercepat di dunia, berkat kemudahan penggunaan, kurva belajar yang cepat, dan berbagai paket berkualitas tinggi untuk data science dan *machine learning* (Vallat, 2018).

2.3 Penelitian Terkait

Tabel 2.3. 1 Penelitian Terkait

No.	Area Penelitian	Tahun	Karakteristik Data	Metode	Model	Hasil Penelitian
1.	Penerapan <i>Deep Learning</i> dalam Deteksi Penipuan (Randhawa et al., 2018)	2021	Data transaksi banksim	<i>Machine Learning</i>	<i>Machine Learning with Ensemble, Deep Learning.</i>	Nilai akurasi transaksi normal dalam <i>dataset</i> adalah 98.789% yang merupakan nilai akurasi dasar untuk membangun model.
2.	Deteksi Penipuan dengan <i>Machine Learning</i> (Prajapati et al., 2021)	2021	Dataset kartu kredit	<i>Machine Learning</i>	<i>Light GBM, Adaboost, Random Forest Classifier</i>	Hasil terbaik dalam hal akurasi yaitu , <i>AdaBoost</i> yang memberikan hasil tertinggi dengan 0,9613. Dalam hal presisi, <i>Light BGM</i> menghasilkan hasil tertinggi dengan 0,986. Dalam hal <i>recall</i> , <i>Adaboost</i> memberikan <i>recall</i> tertinggi dengan 0.889. Dari segi waktu pelaksanaan,
3.	Model Deteksi Penipuan Kartu Kredit yang Efisien (Trivedi et	2020	Dataset kartu kredit	<i>Machine Learning</i>	<i>Random Forest, Naïve Baiyes. Logistic Regression, SVM,</i>	Hasil eksperimen menunjukkan persentase parameter penilaian yang berbeda hanya untuk kumpulan data penipuan kartu kredit untuk teknik

	al., 2020)				<i>kNN</i> , <i>Decision Tree</i> , <i>GBM</i>	<i>machine learning</i> yang berbeda. Hasil menunjukkan bahwa teknik RF menunjukkan persentase akurasi dengan 95,988 persen, meskipun SVM 93,228 persen, LR 92,89 persen, NB 91,2 persen, DC 90,9 persen serta GBM 93,99 persen menunjukkan persentase presisi identifikasi penipuan kartu kredit <i>machine learning</i> ULB.
4.	Klasifikasi pada Kebakaran Hutan dan Lahan (Muslim & Karo, 2020)	2020	Data kebakaran hutan	<i>XGBoost</i> , <i>Feature Importance</i>	<i>Klasifikasi</i>	hasil klasifikasi diperoleh akurasi sebesar 82.51 % dengan running time 10 menit 35 detik. Kami percaya bahwa hasil tersebut tidaklah efektif, mengingat jumlah data yang diolah tidak besar dengan akurasi yang tidak menyentuh angka 85% terlebih lagi memakan waktu yang lama. Selanjutnya akan dipilih beberapa variabel terbaik dengan menggunakan feature importance

5.	Evaluasi Kinerja pada Pendeteksi Penipuan (Mittal & Tyagi, 2019)	2019	Dataset Kartu kredit	<i>Machine Learning, Supervised Learning, Unsupervised Learning.</i>	<i>Random Forest, Neural Networks (NN) and Artificial Neural Networks (ANN), Deep Learning (DL), Support Vector Machine (SVM), Nave Bayes, Logistic Regression, Extended Gradient Boosted Tree (XGBT), Quadratic Discriminant Analysis (QDA), K-Nearest Neighbours</i>	Ada beberapa nilai NaN dalam tabel hasil di mana pengklasifikasi tidak dapat mendeteksi bahkan satu nilai positif atau negatif
6.	Perbandingan Aktifitas System Deteksi Penipuan dengan Menggunakan	2021	Data Kartu Kredit	<i>Machine Learning</i>	<i>Decision Tree, Support vector machine (SVM), Neural Network, Logistic Regression</i>	Hasil yang didapat pada penelitian ini adalah mengetahui metode terbaik sesuai dengan kebutuhan data masing-masing

	Menode Analisis Komprehensif (Veeramani, 2022)					
7.	Deteksi Penipuan Kartu Kredit Menggunakan AdaBoost dan Suara Mayoritas (Randhawa et al., 2018)	2018	Dataset kartu kredit	<i>Machine Learning, AdaBoost</i>	<i>Bayesia, Trees, Neural Network, Linear Regression, Logistic Regression, Support Vector Machine (SVM)</i>	Hasil semua akurasi masih diatas 90% dan tidak ada yang berada di bawah 30%
8.	Bayesian Active Learning for Wearable Stress and Affect Detection (Ragav & Gudur, 2020)	2020	Dataset deteksi stres	<i>Machine Learning</i>	<i>Bayesian Active Learning</i>	Hasil rasio variasi mencapai akurasi 90,38% yang sebanding dengan akurasi pengujian maksimum yang dicapai saat pelatihan pada data sekitar 40% lebih rendah.
9.	Deteksi stress dengan LSTM (Winata et al., 2018)	2018	Dataset deteksi stres	<i>Machine Learning</i>	<i>Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BLSTM),</i>	Model LSTM dua arah dengan perhatian ditemukan sebagai model terbaik dalam hal akurasi (74,1%) dan f-score (74,3%).

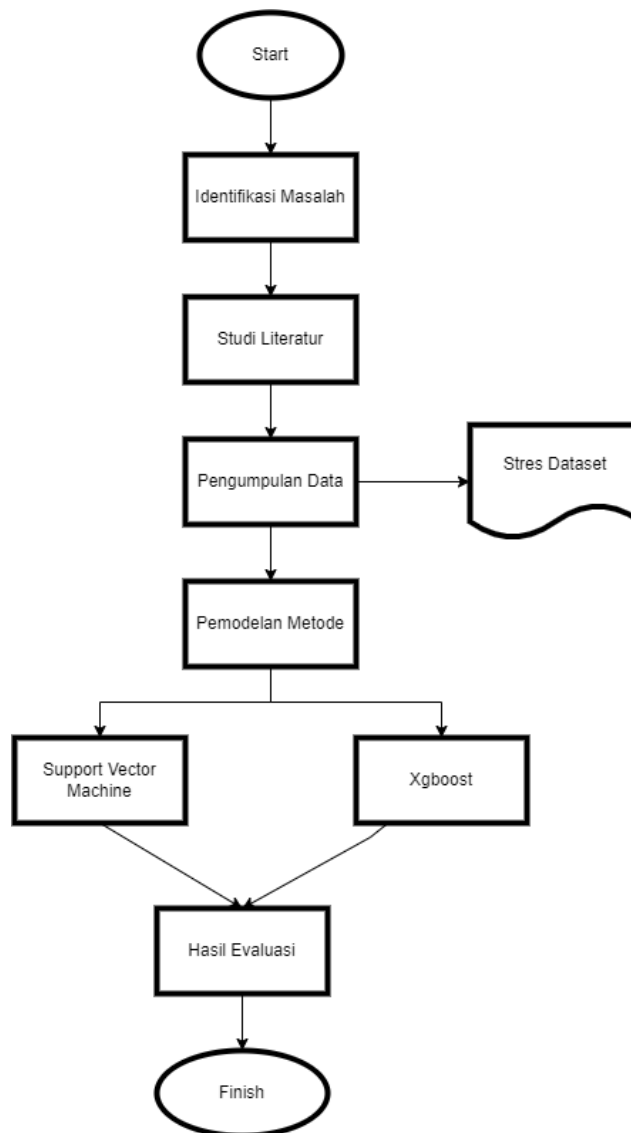
					<i>Support Vector Machine (SVM)</i>	
10.	Deteksi Stres Pengemudi Berbasis EKG (Amin et al., 2022)	2022	Dataset stress pengemudi	<i>Deep Transfer</i>	<i>Logika Fuzzy</i>	Hasil menunjukkan bahwa Model 5 berdasarkan Xception mengungguli model berbasis GoogLeNet, DarkNet-53, ResNet-101, InceptionResNetV2, DenseNet-201, dan InceptionV3 masing-masing sebesar 11,32%, 11,32%, 9,45%, 7,54%, 5,66%, dan 1,88% dan mencapai akurasi validasi keseluruhan 98,11%.

BAB III

METODE PENELITIAN

3.1 Diagram Alur Metodologi Penelitian

Adapun alur metodologi penelitian yang menjelaskan alur, metode, atau cara bagaimana penelitian dilakukan yang dimulai dari menentukan topik penelitian dari fenomena tertentu kemudian berkembang untuk mengidentifikasi dan merumuskan permasalahan, referensi terkait untuk menentukan model, hingga proses visualisasi dan seterusnya



Gambar 3. 1 Alur Metodologi Penelitian (Reksi & Ernawati, 2020)

3.2 Tahapan-Tahapan Diagram Alur Metodologi Penelitian

Adapun tahapan-tahapan dalam penelitian adalah diagram yang menjelaskan tiap tahap-tahap dalam menyelesaikan penelitian yang diuraikan dan dijelaskan lebih detail seperti berikut:

3.2.1 Identifikasi Masalah

Merumuskan permasalahan yang ada di dalam penelitian sehingga peneliti mendapatkan gambaran dari ruang lingkup penelitian yang akan dilakukan.

3.2.2 Studi Literatur

Studi literatur dengan membaca jurnal, buku dan referensi lainnya untuk mengenal deteksi stress, konsep, library keras, bahasa pemrograman python dan teori lainnya untuk mendukung jalannya penelitian.

3.2.3 Pengumpulan Data

Pengumpulan data merupakan data yang digunakan yaitu dataset tentang stress dan pengaruhnya yang bersumber dari uci dataset dengan alamat url sebagai berikut:

<https://archive.ics.uci.edu/ml/datasets/WESAD+%28Wearable+Stress+and+Affect+Detection%29>

3.2.4 Pemodelan Metode

Pemodelan metode adalah tahapan untuk memproses data dengan menggunakan perbandingan 2 metode yaitu *support vector machine* dan *xgboost* untuk mengolah data dan membandingkan hasilnya.

3.2.5 Hasil Evaluasi

Pada proses evaluasi hasil ini peneliti menjelaskan hasil penelitian yang sudah dilakukan.

3.3 Indikator Capaian Penelitian

Berdasarkan diagram alur metodologi penelitian diatas, terdapat indikator capaian sebagai berikut.

Tabel 3. 1 Indikator Capaian

No.	Tahapan		Indikator capaian
1.	Identifikasi Masalah	→	1. Rancangan Researce question.
2.	Studi Literatur	→	2. Deskripsi teori, temuan dan bahan penelitian terkait
3.	Pengumpulan Data	→	3. Pengumpulan data dilakukan

			dengan mengambil sampel data pada uci dataset tentang stres yang dijadikan bahan penelitian
4.	Pemodelan Metode	→	4. Pemodelan metode merupakan tahap mempersiapkan data untuk selanjutnya dilakukan analisis. Beberapa tahap yang dilakukan pada pemodelan yaitu mempersiapkan dua metode yaitu <i>support vector machine</i> dan <i>xgboost</i> .
6.	Evaluasi Hasil	→	5. Hasil penelitian.
7.	Diseminasi Hasil	→	6. Laporan tugas akhir dan draft jurnal.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Pada penelitian ini penulis mengambil sampel data dari dataset stress dan pengaruhnya yang di ambil dari uci dataset dengan alamat url sebagai berikut:

<https://archive.ics.uci.edu/ml/datasets/WESAD+%28Wearable+Stress+and+Affect+Detection%29>. Dimana isi dataset tersebut berisi tentang kumpulan data multimodal baru yang tersedia untuk umum. Data telah direkam menggunakan dua perangkat yang berbeda (satu berbasis dada dan satu berbasis pergelangan tangan), masing-masing termasuk modalitas fisiologis resolusi tinggi (BVP, EKG, EDA, EMG, RESP, dan TEMP) dan gerak (ACC). Data yang didapat dalam bentuk pkl yang kemudian diringkas dengan meminta 15 partisipan untuk melakukan proses pengecekan pengaruh berdasarkan data elektrokardiografi (EKG) menggunakan tiga pendekatan:

1. *Classical*: memanfaatkan fitur berbasis pengetahuan dan training model *Support Vector Machine* (SVM) dan *Xgboost* untuk masalah klasifikasi tiga kelas.
2. *Fully-supervised deep learning based*: memanfaatkan sinyal mentah untuk training *Convolutional Neural Network* (CNN), langsung pada data target.
3. *Self-supervised deep learning based*: memanfaatkan model yang telah dilatih sebelumnya pada dataset yang diubah menggunakan *transfer learning*.

Berikut merupakan data mentah yang digunakan sampelnya pada penelitian ini:

1. *Classical*

Tabel 4. 1 Classical

	0	1	2
0	0.832	0.599	0.55
1	0.795	0.588	0.543
2	0.759	0.55	0.509
3	0.576	0.323	0.438
4	0.623	0.388	0.461
5	0.698	0.493	0.517
6	0.45	0.323	0.316
7	0.752	0.541	0.499
8	0.801	0.582	0.534
9	0.778	0.576	0.569
10	0.688	0.529	0.504

11	0.703	0.502	0.551
12	0.717	0.53	0.531
13	0.603	0.435	0.595
14	0.69	0.513	0.528

Dimana merupakan hasil lab yang menggambarkan:

0 = base/normal

1 = fun

2 = stress

2. *Fully-supervised*

Tabel 4. 2 Fully-supervised

	0	1	2
0	0.991	0.988	0.992
1	0.969	0.959	0.964
2	0.973	0.966	0.959
3	0.979	0.973	0.98
4	0.986	0.982	0.983
5	0.993	0.991	0.989
6	0.987	0.983	0.991
7	0.978	0.971	0.98
8	0.973	0.963	0.977
9	0.983	0.978	0.988
10	0.968	0.961	0.948
11	0.997	0.997	0.997
12	0.986	0.982	0.988
13	0.979	0.972	0.975
14	0.979	0.972	0.985

Dimana merupakan hasil lab yang menggambarkan:

0 = base/normal

1 = fun

2 = stress

3. Self-supervised

Tabel 4. 3 Self-supervised

	0	1	2
0	0.994	0.993	0.992
1	0.995	0.993	0.995
2	0.988	0.984	0.99
3	0.976	0.978	0.985
4	0.984	0.979	0.976
5	0.996	0.995	0.994
6	0.994	0.992	0.991
7	0.984	0.989	0.996
8	0.992	0.989	0.993
9	0.94	0.955	0.995
10	0.95	0.973	0.986
11	0.988	0.995	0.992
12	0.961	0.959	0.963
13	0.983	0.995	0.992
14	0.995	0.993	0.995

Dimana merupakan hasil lab yang menggambarkan:

0 = base/normal

1 = fun

2 = stress

Dari ketiga sampel data diatas merupakan sampel data mentah dengan menggunakan 3 pendekatan yaitu *Classical*, *Fully-supervised*, dan *Self-supervised*.

4.2 Hasil Penelitian

4.2.1 Import Library

```
In [1]: import os
import gc

import pandas as pd
import numpy as np
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers

from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, recall_score, f1_score, confusion_matrix
from sklearn.model_selection import LeaveOneGroupOut, LeavePGroupsOut

from matplotlib import pyplot as plt
```

Gambar 4. 1 Import Library

Pada gambar diatas dilakukan import library yang akan digunakan untuk preprocessing data dan untuk memanggil algoritma *Support Vector Machine* (SVM) dan *Xgboost*.

4.2.2 Pendekatan *Classical*

Pertama mulai dengan menghitung baseline untuk data keseluruhan. Kemudian baca fitur berbasis expert dan mencetak beberapa baris dan menampilkan barus dari data.

```
In [2]: ecg_expert = pd.read_csv('data/ECG_expert_features.csv', index_col=0)
ecg_expert.head()
```

Out[2]:

	pid	lab	bpm	ibi	sdnn	sdsd	rmssd	pnn20	pnn50	hr_mad	sd1	sd2	s	sd1/sd2
S10_000	S10	Base	100.351586	597.897874	20.918173	3.974686	6.208745	0.000000	0.0	15.62500	4.371335	29.151830	400.340724	0.149951
S10_001	S10	Base	101.107492	593.427835	19.516506	4.361267	6.585369	0.000000	0.0	11.71875	4.647473	27.295107	398.521312	0.170268
S10_002	S10	Base	101.610857	590.488080	18.094553	5.043338	7.379010	0.010526	0.0	11.71875	5.211182	25.070505	410.439547	0.207861
S10_003	S10	Base	102.648846	584.517045	18.080210	4.874242	6.949337	0.010309	0.0	11.71875	4.907236	25.190906	388.356480	0.194802
S10_004	S10	Base	103.052318	582.228535	17.769968	4.959666	7.270157	0.010309	0.0	11.71875	5.140778	24.741109	399.574576	0.207783

breathingrate	lf	hf	lf/hf
0.069023	4317.830834	886.033184	4.873216
0.121707	1633.377329	646.859113	2.525090
0.069896	4289.566142	420.117395	10.210399
0.069183	11034.009898	871.615284	12.659266
0.069453	14116.895674	1392.987428	10.134259

Gambar 4. 2 Dataset Stres

Kemudian proses membaginya menjadi train dan test, lalu menormalkan data.

```
# Define train and test sets
train_participants = ['S2', 'S3', 'S4', 'S5', 'S6', 'S7', 'S8', 'S9', 'S10', 'S11', 'S13', 'S14']
test_participants = ['S15', 'S16', 'S17']

# Subsample indices for each set
train_idx = ecg_expert[ecg_expert.pid.isin(train_participants)].index
test_idx = ecg_expert[ecg_expert.pid.isin(test_participants)].index

# Split data
X_train = ecg_expert.loc[train_idx].iloc[:,2:].to_numpy() # iloc[:,2:] drops first two columns (pid and lab)
y_train = ecg_expert.loc[train_idx].lab.to_numpy()
y_train = class_to_number(y_train)

X_test = ecg_expert.loc[test_idx].iloc[:,2:].to_numpy()
y_test = ecg_expert.loc[test_idx].lab.to_numpy()
y_test = class_to_number(y_test)

# Normalize data
scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train)
X_test_norm = scaler.transform(X_test)
```

Gambar 4. 3 Normalize Data

Sekarang data telah siap untuk di lakukan train classifier dan mengecek hasil prediksi dengan menggunakan pendekatan *classical*

- *Support Vector Machine*

```
# Define classifier (SVM for classification) with default parameters
clf = SVC()

# Fit it to train data
clf.fit(X_train_norm, y_train)

# Make predictions
pred_classical = clf.predict(X_test_norm)

print('Accuracy\t{}'.format(np.round(accuracy_score(pred_classical, y_test),3)))
print('F1 score\t{}'.format(np.round(f1_score(pred_classical, y_test, average='macro'),3)))
print('UAR\t\t{}'.format(np.round(recall_score(pred_classical, y_test, average='macro'),3)))
```

Accuracy	0.739
F1 score	0.552
UAR	0.525

Gambar 4. 4 Evaluasi Data Training dan Data Testing Support Vector Machine

Dengan diperoleh evaluasi model yaitu:

Accuracy = 73%

F1 Score = 55%

UAR = 52%

- *Xgboost*

```
# Define classifier (SVM for classification) with default parameters
clf = XGBClassifier()

# Fit it to train data
clf.fit(X_train_norm, y_train)

# Make predictions
pred_classical = clf.predict(X_test_norm)

print('Accuracy\t{}'.format(np.round(accuracy_score(pred_classical, y_test),3)))
print('F1 score\t{}'.format(np.round(f1_score(pred_classical, y_test, average='macro'),3)))
print('UAR\t\t{}'.format(np.round(recall_score(pred_classical, y_test, average='macro'),3)))
```

Accuracy	0.598
F1 score	0.517
UAR	0.52

Gambar 4. 5 Evaluasi Data Training dan Data Testing Xgboost

Dengan diperoleh evaluasi model yaitu:

Accuracy = 59%

F1 Score = 51%

UAR = 52%

4.2.3 Pendekatan *Fully supervised*

Pada langkah ini peneliti akan menggunakan CNN untuk mengenali pengaruh stress, dengan arsitektur yang diusulkan atau digunakan adalah tensorflow dan keras untuk membuat, train, dan test model yang digunakan. Dimana dimulai dari membaca data yang digunakan terlebih dahulu.

- *Support Vector Machine*

Pertama kita akan membaca data dan menormalkan data:

```
def read_data(participants):
    # This function reads the data according to participants list taken as an argument

    X = []

    for filename in train_participants:
        temp = pd.read_csv('data/{}.csv'.format(filename), index_col=0)
        X.append(temp)

    X = pd.concat(X, axis=0)
    y = X.lab.to_numpy()
    y = class_to_number(y)
    X = X.iloc[:,2:].to_numpy()

    return X, y

# Define train and test sets
train_participants = ['S2', 'S3', 'S4', 'S5', 'S6', 'S7', 'S8', 'S9', 'S10']
devel_participants = ['S11', 'S13', 'S14']
test_participants = ['S15', 'S16', 'S17']

# Read files for train and test participants
X_train, y_train = read_data(train_participants)
X_devel, y_devel = read_data(devel_participants)
X_test, y_test = read_data(test_participants)

# Normalize data
scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train)
X_devel_norm = scaler.transform(X_devel)
X_test_norm = scaler.transform(X_test)

# Convert labels to one-hot
y_train_one_hot = one_hot_encoding(y_train, 3)
y_devel_one_hot = one_hot_encoding(y_devel, 3)
y_test_one_hot = one_hot_encoding(y_test, 3)
```

Setelah data sudah dinormalkan dan data siap, kemudian dapat dilanjutkan dengan pemodelan:

```
# Create a model
inputs, outputs = create_graph(input_shape=2560, num_classes=3)
model_fully_supervised = keras.Model(inputs, outputs)

# Define an optimizer and callbacks (to stop the training when the development loss does not decrease)
opt = keras.optimizers.Adam(lr=0.001)
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Compile and fit the model
model_fully_supervised.compile(optimizer=opt, loss='categorical_crossentropy')
history_fs = model_fully_supervised.fit(X_train_norm, y_train_one_hot, validation_data=(X_devel_norm, y_devel_one_hot))
```

```
Epoch 1/100
64/64 - 93s - loss: 1.2698 - val_loss: 0.9882 - 93s/epoch - 1s/step
Epoch 2/100
64/64 - 85s - loss: 0.9946 - val_loss: 0.9106 - 85s/epoch - 1s/step
Epoch 3/100
64/64 - 85s - loss: 0.9163 - val_loss: 0.8683 - 85s/epoch - 1s/step
Epoch 4/100
64/64 - 85s - loss: 0.8740 - val_loss: 0.8229 - 85s/epoch - 1s/step
Epoch 5/100
64/64 - 85s - loss: 0.7578 - val_loss: 0.8773 - 85s/epoch - 1s/step
Epoch 6/100
64/64 - 86s - loss: 0.6528 - val_loss: 0.7859 - 86s/epoch - 1s/step
Epoch 7/100
64/64 - 86s - loss: 0.5908 - val_loss: 0.5717 - 86s/epoch - 1s/step
Epoch 8/100
64/64 - 86s - loss: 0.6030 - val_loss: 0.6381 - 86s/epoch - 1s/step
Epoch 9/100
64/64 - 85s - loss: 0.5949 - val_loss: 0.4686 - 85s/epoch - 1s/step
Epoch 10/100
```

...

```
64/64 - 88s - loss: 0.1897 - val_loss: 0.1637 - 88s/epoch - 1s/step
Epoch 35/100
64/64 - 89s - loss: 0.1848 - val_loss: 0.2235 - 89s/epoch - 1s/step
Epoch 36/100
64/64 - 89s - loss: 0.2090 - val_loss: 0.1577 - 89s/epoch - 1s/step
Epoch 37/100
64/64 - 87s - loss: 0.2532 - val_loss: 0.2167 - 87s/epoch - 1s/step
Epoch 38/100
64/64 - 85s - loss: 0.2266 - val_loss: 0.1666 - 85s/epoch - 1s/step
Epoch 39/100
64/64 - 85s - loss: 0.1725 - val_loss: 0.1570 - 85s/epoch - 1s/step
Epoch 40/100
64/64 - 85s - loss: 0.2056 - val_loss: 0.1797 - 85s/epoch - 1s/step
Epoch 41/100
64/64 - 85s - loss: 0.1748 - val_loss: 0.1092 - 85s/epoch - 1s/step
Epoch 42/100
64/64 - 85s - loss: 0.1990 - val_loss: 0.1432 - 85s/epoch - 1s/step
```

...

```

Epoch 74/100
64/64 - 87s - loss: 0.2032 - val_loss: 0.1523 - 87s/epoch - 1s/step
Epoch 75/100
64/64 - 85s - loss: 0.1075 - val_loss: 0.0597 - 85s/epoch - 1s/step
Epoch 76/100
64/64 - 86s - loss: 0.0883 - val_loss: 0.0765 - 86s/epoch - 1s/step
Epoch 77/100
64/64 - 85s - loss: 0.1058 - val_loss: 0.1862 - 85s/epoch - 1s/step
Epoch 78/100
64/64 - 85s - loss: 0.4104 - val_loss: 0.2086 - 85s/epoch - 1s/step
Epoch 79/100
64/64 - 86s - loss: 0.1657 - val_loss: 0.1086 - 86s/epoch - 1s/step
Epoch 80/100
64/64 - 85s - loss: 0.1265 - val_loss: 0.1018 - 85s/epoch - 1s/step
Epoch 81/100
64/64 - 87s - loss: 0.1404 - val_loss: 0.0711 - 87s/epoch - 1s/step
Epoch 82/100
64/64 - 86s - loss: 0.0920 - val_loss: 0.0595 - 86s/epoch - 1s/step

```

Gambar 4. 6 Epoch Suport Vectore Machine

Bisa dilihat bahwa data memiliki 100 *epoch*. Penjelasan untuk setiap nilai di tiap *epoch* adalah sebagai berikut:

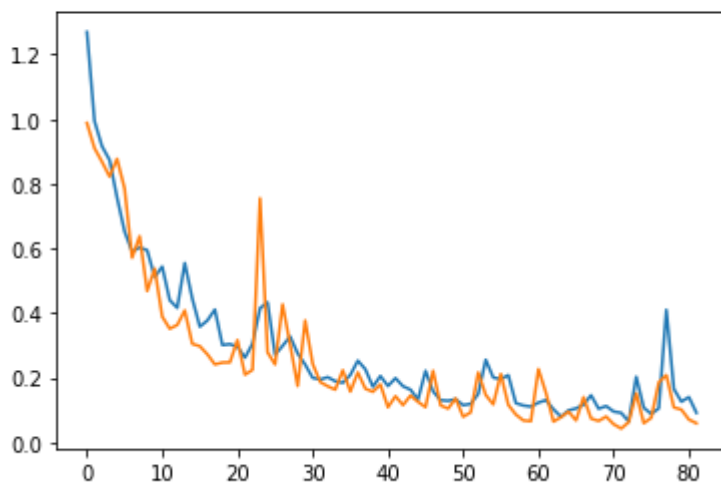
1. *Loss* menyatakan nilai *loss function* di *training set*.
2. *Val_loss* adalah nilai *loss function* di *test set*.

plot grafik train dan los validasi.

```

plt.plot(history_fs.history['loss'])
plt.plot(history_fs.history['val_loss'])
plt.show()

```



Gambar 4. 7 Grafik Support vector Machine

Selanjutnya membuat prediksi untuk kumpulan teks dan menghitung nilai matriks:

```
print('Accuracy\t{}'.format(np.round(accuracy_score(np.argmax(pred_fully_supervised, axis=1), y_test),3)))
print('F1 score\t{}'.format(np.round(f1_score(np.argmax(pred_fully_supervised, axis=1), y_test, average='macro'),3)))
print('UAR\t{}'.format(np.round(recall_score(np.argmax(pred_fully_supervised, axis=1), y_test, average='macro'),3)))
```

```
Accuracy      0.984
F1 score      0.98
UAR           0.985
```

Gambar 4. 8 Hasil Evaluasi Support Vector Machine

Dengan diperoleh evaluasi model yaitu:

Accuracy = 98%

F1 Score = 98%

UAR = 98%

- *Xgboost*

Sama seperti pemodelan sebelumnya setelah data sudah dinormalkan dan data siap, kemudian dapat dilanjutkan dengan pemodelan:

```
# Create a model
inputs, outputs = create_graph(input_shape=2560, num_classes=3)
model_fully_supervised = keras.Model(inputs, outputs)

# Define an optimizer and callbacks (to stop the training when the development loss does not decrease)
opt = keras.optimizers.Adam(lr=0.001)
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Compile and fit the model
model_fully_supervised.compile(optimizer=opt, loss='categorical_crossentropy')
history_fs = model_fully_supervised.fit(X_train_norm, y_train_one_hot, validation_data=(X_devel_norm, y_devel_one_hot))
```

```
Epoch 1/100
64/64 - 117s - loss: 1.2309 - val_loss: 0.9745 - 117s/epoch - 2s/step
Epoch 2/100
64/64 - 105s - loss: 0.9927 - val_loss: 0.9409 - 105s/epoch - 2s/step
Epoch 3/100
64/64 - 105s - loss: 0.9672 - val_loss: 0.8702 - 105s/epoch - 2s/step
Epoch 4/100
64/64 - 106s - loss: 0.8144 - val_loss: 0.8052 - 106s/epoch - 2s/step
Epoch 5/100
64/64 - 116s - loss: 0.7426 - val_loss: 0.6786 - 116s/epoch - 2s/step
Epoch 6/100
64/64 - 110s - loss: 0.6624 - val_loss: 0.6980 - 110s/epoch - 2s/step
Epoch 7/100
64/64 - 112s - loss: 0.5685 - val_loss: 0.7694 - 112s/epoch - 2s/step
Epoch 8/100
64/64 - 108s - loss: 0.5591 - val_loss: 0.7299 - 108s/epoch - 2s/step
Epoch 9/100
64/64 - 111s - loss: 0.6698 - val_loss: 0.7671 - 111s/epoch - 2s/step
```

...

```

Epoch 45/100
64/64 - 95s - loss: 0.3167 - val_loss: 0.2167 - 95s/epoch - 1s/step
Epoch 46/100
64/64 - 93s - loss: 0.1823 - val_loss: 0.1360 - 93s/epoch - 1s/step
Epoch 47/100
64/64 - 96s - loss: 0.2429 - val_loss: 0.1398 - 96s/epoch - 1s/step
Epoch 48/100
64/64 - 97s - loss: 0.1591 - val_loss: 0.0905 - 97s/epoch - 2s/step
Epoch 49/100
64/64 - 91s - loss: 0.1732 - val_loss: 0.0915 - 91s/epoch - 1s/step
Epoch 50/100
64/64 - 96s - loss: 0.1627 - val_loss: 0.2137 - 96s/epoch - 1s/step
Epoch 51/100
64/64 - 98s - loss: 0.1622 - val_loss: 0.1301 - 98s/epoch - 2s/step
Epoch 52/100
64/64 - 91s - loss: 0.1425 - val_loss: 0.1027 - 91s/epoch - 1s/step
Epoch 53/100
64/64 - 92s - loss: 0.1080 - val_loss: 0.0619 - 92s/epoch - 1s/step
...
64/64 - 89s - loss: 0.2475 - val_loss: 0.1102 - 89s/epoch - 1s/step
Epoch 56/100
64/64 - 87s - loss: 0.1512 - val_loss: 0.0975 - 87s/epoch - 1s/step
Epoch 57/100
64/64 - 88s - loss: 0.1335 - val_loss: 0.1722 - 88s/epoch - 1s/step
Epoch 58/100
64/64 - 87s - loss: 0.1342 - val_loss: 0.0788 - 87s/epoch - 1s/step
Epoch 59/100
64/64 - 87s - loss: 0.1412 - val_loss: 0.0951 - 87s/epoch - 1s/step
Epoch 60/100
64/64 - 86s - loss: 0.1897 - val_loss: 0.1404 - 86s/epoch - 1s/step
Epoch 61/100
64/64 - 86s - loss: 0.1079 - val_loss: 0.0885 - 86s/epoch - 1s/step
Epoch 62/100
64/64 - 88s - loss: 0.1099 - val_loss: 0.1105 - 88s/epoch - 1s/step
Epoch 63/100
64/64 - 89s - loss: 0.3059 - val_loss: 0.1686 - 89s/epoch - 1s/step

```

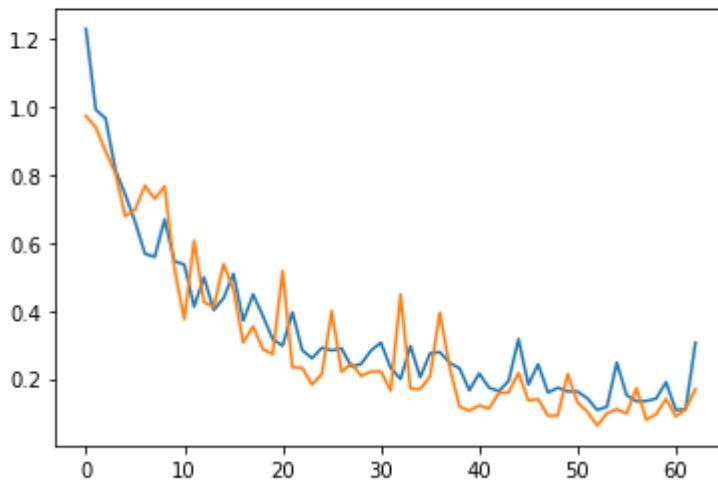
Gambar 4. 9 Epoch Xgboost

Bisa dilihat bahwa data memiliki 100 *epoch*. Penjelasan untuk setiap nilai di tiap *epoch* adalah sebagai berikut:

1. *Loss* menyatakan nilai *loss function* di *training set*.
2. *Val_loss* adalah nilai *loss function* di *test set*.

plot grafik train dan los validasi.

```
plt.plot(history_fs.history['loss'])
plt.plot(history_fs.history['val_loss'])
plt.show()
```



Gambar 4. 10 Grafik Xgboost

Selanjutnya membuat prediksi untuk kumpulan teks dan menghitung nilai matriks:

```
print('Accuracy\t\t').format(np.round(accuracy_score(np.argmax(pred_fully_supervised, axis=1), y_test),3)))
print('F1 score\t\t').format(np.round(f1_score(np.argmax(pred_fully_supervised, axis=1), y_test, average='macro'),3)))
print('UAR\t\t\t').format(np.round(recall_score(np.argmax(pred_fully_supervised, axis=1), y_test, average='macro'),3)))
```

```
Accuracy      0.98
F1 score      0.975
UAR           0.973
```

Dengan diperoleh evaluasi model yaitu:

Accuracy = 98%

F1 Score = 97%

UAR = 97%

4.2.4 Pendekatan *Self supervised + transfer learning*

Selanjutnya, menggunakan pendekatan self-supervised untuk mempelajari representasi dari data dan menerapkan transfer learning dengan menggunakan data yang sudah di normalisasikan pada pendekatan sebelumnya.

- *Support Vector Machine*

```
# Create a model
inputs, outputs = create_graph(input_shape=2560, num_classes=4)
model_pretext = keras.Model(inputs, outputs)

# Define an optimizer and callbacks (to stop the training when the development loss does not decrease)
opt = keras.optimizers.Adam(lr=0.001)
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Compile and fit the model
model_pretext.compile(optimizer=opt, loss='categorical_crossentropy')
history = model_pretext.fit(X_train_transformed, y_train_transformed_one_hot, validation_data=(X_devel_transformed,
                                                                                             y_devel_transformed_one_hot),
                           callbacks=[callback], epochs=100, batch_size=32, verbose=1)
```

```
Epoch 1/100
253/253 [=====] - 346s 1s/step - loss: 1.0083 - val_loss: 0.8631
Epoch 2/100
253/253 [=====] - 352s 1s/step - loss: 0.4881 - val_loss: 0.4849
Epoch 3/100
253/253 [=====] - 395s 2s/step - loss: 0.3431 - val_loss: 0.2957
Epoch 4/100
253/253 [=====] - 491s 2s/step - loss: 0.3174 - val_loss: 0.3501
Epoch 5/100
253/253 [=====] - 808s 3s/step - loss: 0.2978 - val_loss: 0.9242
Epoch 6/100
253/253 [=====] - 813s 3s/step - loss: 0.2733 - val_loss: 0.3236
Epoch 7/100
253/253 [=====] - 833s 3s/step - loss: 0.2435 - val_loss: 0.4686
Epoch 8/100
253/253 [=====] - 894s 4s/step - loss: 0.2276 - val_loss: 0.6687
Epoch 9/100
253/253 [=====] - 926s 4s/step - loss: 0.1846 - val_loss: 0.3279
Epoch 10/100
253/253 [=====] - 888s 4s/step - loss: 0.1883 - val_loss: 0.3650
Epoch 11/100
253/253 [=====] - 822s 3s/step - loss: 0.1697 - val_loss: 0.3371
Epoch 12/100
253/253 [=====] - 718s 3s/step - loss: 0.1535 - val_loss: 0.3480
```

....


```

Epoch 20/100
253/253 [=====] - 665s 3s/step - loss: 0.0960 - val_loss: 0.3045
Epoch 21/100
253/253 [=====] - 663s 3s/step - loss: 0.0835 - val_loss: 0.2446
Epoch 22/100
253/253 [=====] - 663s 3s/step - loss: 0.0852 - val_loss: 0.0365
Epoch 23/100
253/253 [=====] - 665s 3s/step - loss: 0.0795 - val_loss: 0.0305
Epoch 24/100
253/253 [=====] - 670s 3s/step - loss: 0.1032 - val_loss: 0.0605
Epoch 25/100
253/253 [=====] - 674s 3s/step - loss: 0.0910 - val_loss: 0.1102
Epoch 26/100
253/253 [=====] - 663s 3s/step - loss: 0.0840 - val_loss: 0.0654
Epoch 27/100
253/253 [=====] - 662s 3s/step - loss: 0.0806 - val_loss: 0.6335
Epoch 28/100
253/253 [=====] - 545s 2s/step - loss: 0.0720 - val_loss: 0.1576
Epoch 29/100
253/253 [=====] - 341s 1s/step - loss: 0.0859 - val_loss: 0.1776
Epoch 30/100
253/253 [=====] - 345s 1s/step - loss: 0.0984 - val_loss: 0.0607
Epoch 31/100
253/253 [=====] - 342s 1s/step - loss: 0.0784 - val_loss: 0.0861
Epoch 32/100
253/253 [=====] - 343s 1s/step - loss: 0.0650 - val_loss: 0.3134
Epoch 33/100
253/253 [=====] - 349s 1s/step - loss: 0.0679 - val_loss: 0.1643

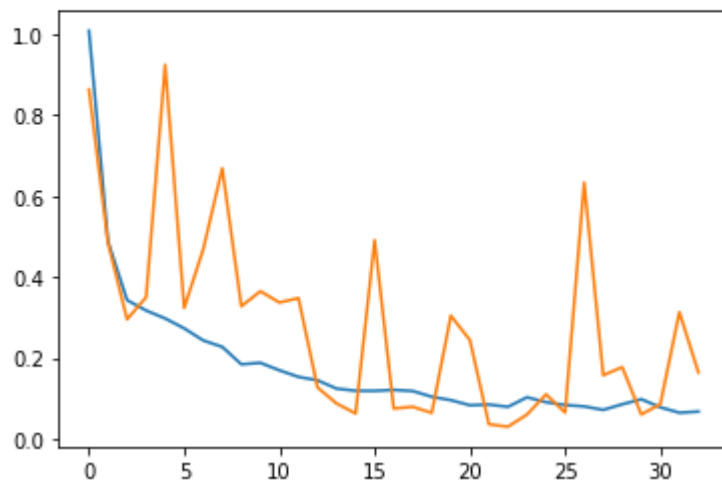
```

Gambar 4. 11 Hasil Epoch Support Vector Machine

```

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.show()

```



Gambar 4. 12 Grafik Support Vector Machine

Kemudian untuk lebih spesifik lagi pada pendekatan ini dapat digunakan downstream model.

```
# Create a model
model_downstream = keras.Model(inputs, outputs_downstream)
for i in range(len(model_downstream.layers)-5):
    model_downstream.layers[i].trainable = False

# Define an optimizer and callbacks (to stop the training when the development loss does not decrease)
opt = keras.optimizers.Adam(lr=0.001)
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Compile and fit the model
model_downstream.compile(optimizer=opt, loss='categorical_crossentropy')
history = model_downstream.fit(X_train_norm, y_train_one_hot, validation_data=(X_devel_norm, y_devel_one_hot))
```

```
Epoch 1/100
64/64 - 64s - loss: 1.2291 - val_loss: 0.9826 - 64s/epoch - 1s/step
Epoch 2/100
64/64 - 48s - loss: 1.0377 - val_loss: 0.9571 - 48s/epoch - 747ms/step
Epoch 3/100
64/64 - 48s - loss: 0.9842 - val_loss: 0.9514 - 48s/epoch - 752ms/step
Epoch 4/100
64/64 - 48s - loss: 0.9703 - val_loss: 0.9815 - 48s/epoch - 757ms/step
Epoch 5/100
64/64 - 48s - loss: 0.9567 - val_loss: 0.9100 - 48s/epoch - 752ms/step
Epoch 6/100
64/64 - 48s - loss: 0.9414 - val_loss: 0.8927 - 48s/epoch - 751ms/step
Epoch 7/100
64/64 - 48s - loss: 0.9301 - val_loss: 0.8784 - 48s/epoch - 750ms/step
Epoch 8/100
64/64 - 48s - loss: 0.9066 - val_loss: 0.8715 - 48s/epoch - 753ms/step
Epoch 9/100
64/64 - 48s - loss: 0.9234 - val_loss: 0.8739 - 48s/epoch - 753ms/step
```

...

```
Epoch 47/100
64/64 - 48s - loss: 0.6980 - val_loss: 0.6118 - 48s/epoch - 749ms/step
Epoch 48/100
64/64 - 48s - loss: 0.6814 - val_loss: 0.5926 - 48s/epoch - 749ms/step
Epoch 49/100
64/64 - 48s - loss: 0.6854 - val_loss: 0.6338 - 48s/epoch - 746ms/step
Epoch 50/100
64/64 - 47s - loss: 0.6963 - val_loss: 0.6330 - 47s/epoch - 740ms/step
Epoch 51/100
64/64 - 49s - loss: 0.6900 - val_loss: 0.6014 - 49s/epoch - 758ms/step
Epoch 52/100
64/64 - 48s - loss: 0.6897 - val_loss: 0.5947 - 48s/epoch - 753ms/step
Epoch 53/100
64/64 - 49s - loss: 0.6959 - val_loss: 0.6336 - 49s/epoch - 764ms/step
Epoch 54/100
64/64 - 49s - loss: 0.6842 - val_loss: 0.6822 - 49s/epoch - 764ms/step
Epoch 55/100
64/64 - 48s - loss: 0.6946 - val_loss: 0.5822 - 48s/epoch - 757ms/step
```

...

```

Epoch 92/100
64/64 - 61s - loss: 0.5911 - val_loss: 0.5236 - 61s/epoch - 951ms/step
Epoch 93/100
64/64 - 60s - loss: 0.5870 - val_loss: 0.5773 - 60s/epoch - 943ms/step
Epoch 94/100
64/64 - 61s - loss: 0.5938 - val_loss: 0.5120 - 61s/epoch - 956ms/step
Epoch 95/100
64/64 - 61s - loss: 0.6131 - val_loss: 0.5367 - 61s/epoch - 955ms/step
Epoch 96/100
64/64 - 61s - loss: 0.5823 - val_loss: 0.4866 - 61s/epoch - 951ms/step
Epoch 97/100
64/64 - 61s - loss: 0.5586 - val_loss: 0.5208 - 61s/epoch - 952ms/step
Epoch 98/100
64/64 - 61s - loss: 0.5851 - val_loss: 0.4657 - 61s/epoch - 954ms/step

Epoch 99/100
64/64 - 63s - loss: 0.6095 - val_loss: 0.4627 - 63s/epoch - 977ms/step
Epoch 100/100
64/64 - 68s - loss: 0.5602 - val_loss: 0.4833 - 68s/epoch - 1s/step

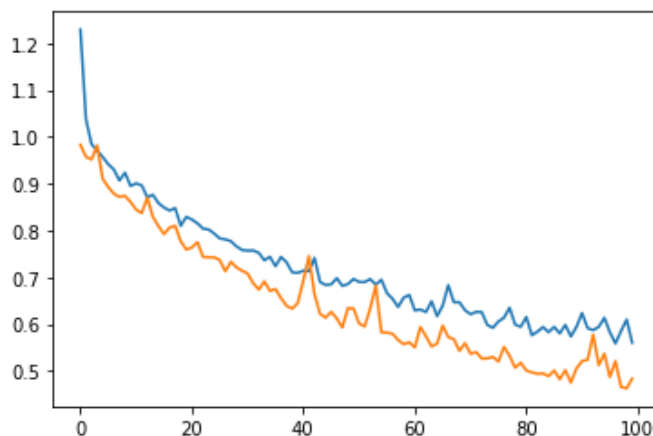
```

Gambar 4. 13 Hasil Evaluasi Epoch Akhir

```

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.show()

```



Gambar 4. 14 Grafik Akhir Support Vector Machine

```

print('Accuracy\t{}'.format(np.round(accuracy_score(np.argmax(pred_downstream, axis=1), y_test),3)))
print('F1 score\t{}'.format(np.round(f1_score(np.argmax(pred_downstream, axis=1), y_test, average='macro'),3)))
print('UAR\t\t{}'.format(np.round(recall_score(np.argmax(pred_downstream, axis=1), y_test, average='macro'),3)))

```

Accuracy	0.81
F1 score	0.69
UAR	0.825

Dengan diperoleh evaluasi model yaitu:

Accuracy = 81%

F1 Score = 69%

UAR = 82%

- *Xgboost*

```
# Create a model
inputs, outputs = create_graph(input_shape=2560, num_classes=4)
model_pretext = keras.Model(inputs, outputs)

# Define an optimizer and callbacks (to stop the training when the development loss does not decrease)
opt = keras.optimizers.Adam(lr=0.001)
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Compile and fit the model
model_pretext.compile(optimizer=opt, loss='categorical_crossentropy')
history = model_pretext.fit(X_train_transformed, y_train_transformed_one_hot, validation_data=(X_devel_transformed,
                                                                                             y_devel_transformed_one_hot),
                           callbacks=[callback], epochs=100, batch_size=32, verbose=1)
```

```
Epoch 1/100
253/253 [=====] - 395s 1s/step - loss: 1.0614 - val_loss: 0.8570
Epoch 2/100
253/253 [=====] - 351s 1s/step - loss: 0.4564 - val_loss: 0.3582
Epoch 3/100
253/253 [=====] - 350s 1s/step - loss: 0.3638 - val_loss: 0.3814
Epoch 4/100
253/253 [=====] - 350s 1s/step - loss: 0.3236 - val_loss: 0.3120
Epoch 5/100
253/253 [=====] - 355s 1s/step - loss: 0.3047 - val_loss: 0.5033
Epoch 6/100
253/253 [=====] - 350s 1s/step - loss: 0.2947 - val_loss: 0.2517
Epoch 7/100
253/253 [=====] - 353s 1s/step - loss: 0.2799 - val_loss: 0.5369
Epoch 8/100
253/253 [=====] - 354s 1s/step - loss: 0.2574 - val_loss: 0.2748
Epoch 9/100
253/253 [=====] - 350s 1s/step - loss: 0.2409 - val_loss: 0.3694
Epoch 10/100
253/253 [=====] - 349s 1s/step - loss: 0.2070 - val_loss: 0.1827
Epoch 11/100
253/253 [=====] - 347s 1s/step - loss: 0.1822 - val_loss: 0.3309
Epoch 12/100
253/253 [=====] - 349s 1s/step - loss: 0.1705 - val_loss: 0.1329
Epoch 13/100
253/253 [=====] - 349s 1s/step - loss: 0.1480 - val_loss: 0.1957
Epoch 14/100
```

....

```

Epoch 27/100
253/253 [=====] - 465s 2s/step - loss: 0.0799 - val_loss: 0.0385
Epoch 28/100
253/253 [=====] - 466s 2s/step - loss: 0.0773 - val_loss: 0.1110
Epoch 29/100
253/253 [=====] - 449s 2s/step - loss: 0.0863 - val_loss: 0.0565
Epoch 30/100
253/253 [=====] - 481s 2s/step - loss: 0.0680 - val_loss: 0.0631
Epoch 31/100
253/253 [=====] - 504s 2s/step - loss: 0.0705 - val_loss: 0.0392
Epoch 32/100
253/253 [=====] - 515s 2s/step - loss: 0.1106 - val_loss: 0.0815
Epoch 33/100
253/253 [=====] - 490s 2s/step - loss: 0.0819 - val_loss: 0.0429
Epoch 34/100
253/253 [=====] - 453s 2s/step - loss: 0.0761 - val_loss: 0.0706
Epoch 35/100
253/253 [=====] - 419s 2s/step - loss: 0.0745 - val_loss: 0.0556
Epoch 36/100
253/253 [=====] - 423s 2s/step - loss: 0.0819 - val_loss: 0.1039
Epoch 37/100
253/253 [=====] - 418s 2s/step - loss: 0.0595 - val_loss: 0.0582

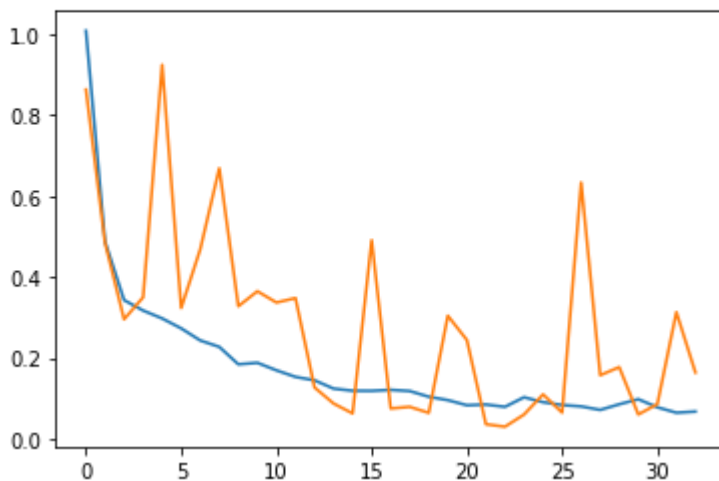
```

Gambar 4. 15 Hasil Epoch Support Vector Machine

```

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.show()

```



Gambar 4. 16 Grafik Support Vector Machine

Kemudian untuk lebih spesifik lagi pada pendekatan ini dapat digunakan downstream model.

```
# Create a model
model_downstream = keras.Model(inputs, outputs_downstream)
for i in range(len(model_downstream.layers)-5):
    model_downstream.layers[i].trainable = False

# Define an optimizer and callbacks (to stop the training when the development loss does not decrease)
opt = keras.optimizers.Adam(lr=0.001)
callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Compile and fit the model
model_downstream.compile(optimizer=opt, loss='categorical_crossentropy')
history = model_downstream.fit(X_train_norm, y_train_one_hot, validation_data=(X_devel_norm, y_devel_one_hot))
```

```
Epoch 1/100
64/64 - 47s - loss: 1.2500 - val_loss: 1.0713 - 47s/epoch - 737ms/step
Epoch 2/100
64/64 - 31s - loss: 1.0463 - val_loss: 0.9591 - 31s/epoch - 490ms/step
Epoch 3/100
64/64 - 32s - loss: 0.9989 - val_loss: 0.9338 - 32s/epoch - 499ms/step
Epoch 4/100
64/64 - 33s - loss: 0.9675 - val_loss: 0.9273 - 33s/epoch - 509ms/step
Epoch 5/100
64/64 - 33s - loss: 0.9405 - val_loss: 0.8827 - 33s/epoch - 511ms/step
Epoch 6/100
64/64 - 32s - loss: 0.9382 - val_loss: 0.8710 - 32s/epoch - 506ms/step
Epoch 7/100
64/64 - 32s - loss: 0.9085 - val_loss: 0.8627 - 32s/epoch - 503ms/step
Epoch 8/100
64/64 - 32s - loss: 0.8910 - val_loss: 0.8401 - 32s/epoch - 506ms/step
Epoch 9/100
64/64 - 32s - loss: 0.8970 - val_loss: 0.8406 - 32s/epoch - 502ms/step
Epoch 10/100
...

Epoch 43/100
64/64 - 33s - loss: 0.6356 - val_loss: 0.6045 - 33s/epoch - 518ms/step
Epoch 44/100
64/64 - 35s - loss: 0.6780 - val_loss: 0.5543 - 35s/epoch - 540ms/step
Epoch 45/100
64/64 - 33s - loss: 0.6549 - val_loss: 0.5945 - 33s/epoch - 517ms/step
Epoch 46/100
64/64 - 35s - loss: 0.6572 - val_loss: 0.5579 - 35s/epoch - 545ms/step
Epoch 47/100
64/64 - 33s - loss: 0.6256 - val_loss: 0.5370 - 33s/epoch - 518ms/step
Epoch 48/100
64/64 - 33s - loss: 0.6186 - val_loss: 0.5887 - 33s/epoch - 514ms/step
Epoch 49/100
64/64 - 33s - loss: 0.6340 - val_loss: 0.5571 - 33s/epoch - 514ms/step
Epoch 50/100
64/64 - 34s - loss: 0.6049 - val_loss: 0.5261 - 34s/epoch - 524ms/step
Epoch 51/100
...
```

```

Epoch 92/100
64/64 - 31s - loss: 0.5698 - val_loss: 0.4697 - 31s/epoch - 491ms/step
Epoch 93/100
64/64 - 31s - loss: 0.5360 - val_loss: 0.4333 - 31s/epoch - 491ms/step
Epoch 94/100
64/64 - 32s - loss: 0.5374 - val_loss: 0.4386 - 32s/epoch - 493ms/step
Epoch 95/100
64/64 - 31s - loss: 0.5267 - val_loss: 0.4238 - 31s/epoch - 487ms/step
Epoch 96/100
64/64 - 32s - loss: 0.5398 - val_loss: 0.4635 - 32s/epoch - 493ms/step
Epoch 97/100
64/64 - 31s - loss: 0.5755 - val_loss: 0.4499 - 31s/epoch - 488ms/step
Epoch 98/100
64/64 - 31s - loss: 0.5377 - val_loss: 0.4334 - 31s/epoch - 489ms/step

Epoch 99/100
64/64 - 31s - loss: 0.5391 - val_loss: 0.4452 - 31s/epoch - 488ms/step
Epoch 100/100
64/64 - 32s - loss: 0.5704 - val_loss: 0.4615 - 32s/epoch - 499ms/step

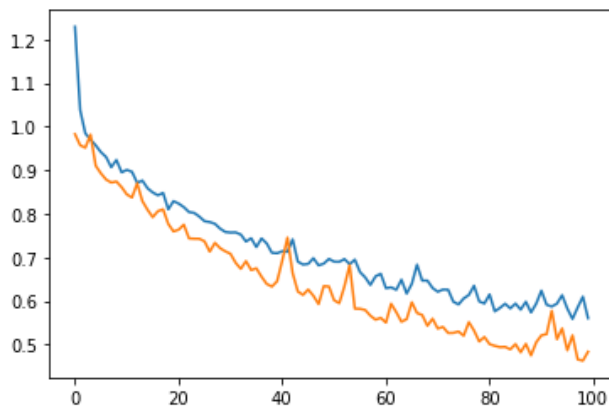
```

Gambar 4. 17 Hasil Evaluasi Epoch Akhir

```

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.show()

```



Gambar 4. 18 Grafik Akhir Support Vector Machine

```

print('Accuracy\t\t{}'.format(np.round(accuracy_score(np.argmax(pred_downstream, axis=1), y_test),3)))
print('F1 score\t\t{}'.format(np.round(f1_score(np.argmax(pred_downstream, axis=1), y_test, average='macro'),3)))
print('UAR\t\t\t{}'.format(np.round(recall_score(np.argmax(pred_downstream, axis=1), y_test, average='macro'),3)))

```

```

Accuracy      0.806
F1 score      0.704
UAR           0.823

```

Dengan diperoleh evaluasi model yaitu:

Accuracy = 80%

F1 Score = 70%

UAR = 82%

Dan untuk hasil akhirnya maka dapat dibandingkan 2 pendekatan deep learning yaitu *Self supervised* dan *Fully supervised*:

- Support Vector Machine

```
print('Self supervised UAR\t{}'.format(np.round(recall_score(np.argmax(pred_downstream, axis=1), y_test, average='macro'),3)))
print('Fully supervised UAR\t{}'.format(np.round(recall_score(np.argmax(pred_fully_supervised, axis=1), y_test, average='macro'))))
```

Self supervised UAR	0.825
Fully supervised UAR	0.985

Dengan diperoleh evaluasi model yaitu:

Self Supervised UAR = 82%

Fully Supervised UAR = 98%

- Xgboost

```
print('Self supervised UAR\t{}'.format(np.round(recall_score(np.argmax(pred_downstream, axis=1), y_test, average='macro'),3)))
print('Fully supervised UAR\t{}'.format(np.round(recall_score(np.argmax(pred_fully_supervised, axis=1), y_test, average='macro'))))
```

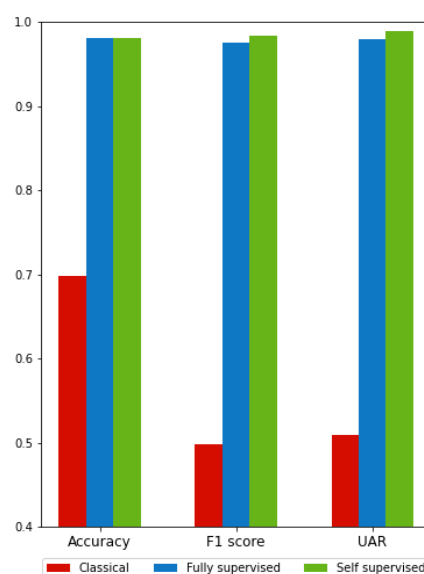
Self supervised UAR	0.823
Fully supervised UAR	0.973

Dengan diperoleh evaluasi model yaitu:

Self Supervised UAR = 82%

Fully Supervised UAR = 97%

Untuk lebih jelasnya maka dapat dilihat pada diagram berikut untuk melihat pendekatan terbaik:



Gambar 4. 19 Diagram Perbandingan 3 Pendekatan

Dari tiga pendekatan yang dilakukan maka dapat dilihat dari diagram diatas bahwa pendekatan self supervised lebih unggul dari 2 pendekatan yang lainnya. Untuk

4.2.5 Evaluasi Model

Tabel 4. 4 Hasil Evaluasi Model

Metode	Accuracy	F1 Score	UAR
Support Vector Machine	98%	98%	98%
Xgboost	98%	97%	97%

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil penelitian yang dilakukan dengan menggunakan algoritma *support vector machine* dan *xgboost*. menggunakan dua metode penelitian yaitu *Support Vector Machine* dan *Xgboost*. Maka dapat dilihat dari hasil *Dataset stress dan pengaruhnya* dengan menggunakan algoritma *Support Vector Machine* (SVM) hasil *accuracy* sebesar 98%, *F1 score* sebesar 98%, dan *UAR* sebesar 98%, sedangkan menggunakan *Xgboost* hasil *accuracy* sebesar 98%, *F1 score* sebesar 97%, dan *UAR* sebesar 97%. Maka dari hasil penelitian ini dapat disimpulkan bahwa metode *Support Vector Machine* (SVM) pada *dataset stress dan pengaruhnya* memiliki nilai akurasi yang lebih unggul

5.2 Saran

Saran daei penelitian ini yaitu dapat dikembangkan lagi dengan menggunakan metode lain dan juga dapat dikembangkan lebih lanjut dengan membuat alat yang dapat mengecek langsung stress dengan menggunakan sinyal EKG.

DAFTAR PUSTAKA

- (Al Amrani, Lazaar, El Kadiri., 2018). (2018). Random Forest and Support Vector Machine based Hybrid Approach to SA --RF.pdf. In *The First International Conference on Intelligent Computing in Data Sciences* (pp. 511–520).
- Ainurrochman, Adi, D. P., & Gumelar, A. B. (2020). Deteksi Emosi Wicara pada Media On-Demand menggunakan SVM dan LSTM. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 4(5), 799–804. <https://doi.org/10.29207/resti.v4i5.2073>
- Amin, M., Ullah, K., Asif, M., Waheed, A., Haq, S. U., Zareei, M., & Biswal, R. R. (2022). ECG-Based Driver's Stress Detection Using Deep Transfer Learning and Fuzzy Logic Approaches. *IEEE Access*, 10, 29788–29809. <https://doi.org/10.1109/ACCESS.2022.3158658>
- Arafin Mahtab, S., Islam, N., & Mahfuzur Rahaman, M. (2018). Sentiment Analysis on Bangladesh Cricket with Support Vector Machine. *2018 International Conference on Bangla Speech and Language Processing, ICBSLP 2018*, 1–4. <https://doi.org/10.1109/ICBSLP.2018.8554585>
- Bakti, W. T., & Wardati, N. K. (2019). Alat Deteksi Tingkat Stres Manusia Berbasis Android Berdasarkan Suhu Tubuh, Heart Rate dan Galvanic Skin Response (GSR). *Jurnal Teknik Elektro Dan Komputasi (ELKOM)*, 1(2), 93–98. <https://doi.org/10.32528/elkom.v1i2.3089>
- Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). Machine Learning for Fluid Mechanics. *Annual Review of Fluid Mechanics*, 52, 477–508. <https://doi.org/10.1146/annurev-fluid-010719-060214>
- Chyan, P., & Kasmara, Y. (2021). Sistem Monitoring dan Deteksi Stres Pada Anak Berbasis Wearable Device. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 5(5), 943–949. <https://doi.org/10.29207/resti.v5i5.3503>
- Di Martino, F., & Delmastro, F. (2020). High-Resolution Physiological Stress Prediction Models based on Ensemble Learning and Recurrent Neural Networks. *Proceedings - IEEE Symposium on Computers and Communications, 2020-July*, 2–7. <https://doi.org/10.1109/ISCC50000.2020.9219716>
- Dong, W., Huang, Y., Lehane, B., & Ma, G. (2020). XGBoost algorithm-based prediction of concrete electrical resistivity for structural health monitoring. *Automation in Construction*, 114(January), 103155. <https://doi.org/10.1016/j.autcon.2020.103155>
- Gedam, S., & Paul, S. (2021). A Review on Mental Stress Detection Using Wearable Sensors

- and Machine Learning Techniques. *IEEE Access*, 9, 84045–84066. <https://doi.org/10.1109/ACCESS.2021.3085502>
- Jiang, Y., Tong, G., Yin, H., & Xiong, N. (2019). A Pedestrian Detection Method Based on Genetic Algorithm for Optimize XGBoost Training Parameters. *IEEE Access*, 7, 118310–118321. <https://doi.org/10.1109/ACCESS.2019.2936454>
- Kalid, S. N., Ng, K. H., Tong, G. K., & Khor, K. C. (2020). A Multiple Classifiers System for Anomaly Detection in Credit Card Data with Unbalanced and Overlapped Classes. *IEEE Access*, 8, 28210–28221. <https://doi.org/10.1109/ACCESS.2020.2972009>
- Marković, D., Vujičić, D., Stojić, D., Jovanović, Ž., Pesović, U., & Randić, S. (2019). Monitoring System Based on IoT Sensor Data with Complex Event Processing and Artificial Neural Networks for Patients Stress Detection. *2019 18th International Symposium INFOTEH-JAHORINA, INFOTEH 2019 - Proceedings, March, 20–22*. <https://doi.org/10.1109/INFOTEH.2019.8717748>
- Melo, L. S., Sampaio, R. F., Leão, R. P. S., Barroso, G. C., & Bezerra, J. R. (2019). Python-based multi-agent platform for application on power grids. *International Transactions on Electrical Energy Systems*, 29(6), 1–14. <https://doi.org/10.1002/2050-7038.12012>
- Mittal, S., & Tyagi, S. (2019). Performance evaluation of machine learning algorithms for credit card fraud detection. *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019*, 320–324. <https://doi.org/10.1109/CONFLUENCE.2019.8776925>
- Muslim, I., & Karo, K. (2020). Implementasi Metode XGBoost dan Feature Importance untuk Klasifikasi pada Kebakaran Hutan dan Lahan. *Journal of Software Engineering, Information and Communication Technology*, 1(1), 10–16.
- Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (Cnn) Pada Ekspresi Manusia. *Algor*, 2(1), 12–21.
- Perdana, R. S., & Fauzi, M. A. (2017). *Analisis Sentimen Tingkat Kepuasan Pengguna Penyedia Layanan Telekomunikasi Seluler Indonesia Pada Twitter dengan Metode Support Vector Machine dan Lexicon Based Features Optimasi Sisa Bahan Baku Pada Industri Mebel Menggunakan Algoritma Genetika View pro. October*.
- Prajapati, D., Tripathi, A., Mehta, J., Jhaveri, K., & Kelkar, V. (2021). Credit Card Fraud Detection Using Machine Learning. *2021 7th IEEE International Conference on Advances in Computing, Communication and Control, ICAC3 2021*, 8592(1), 16–19.

- <https://doi.org/10.1109/ICAC353642.2021.9697227>
- Prasetyo, R. H. (2020). *Sistem Identifikasi Arti Tangisan Bayi Menggunakan Metode Mfcc, Dwt Dan Knn Pada Raspberry Pi*. 7(2), 6–28.
- Ragav, A., & Gudur, G. K. (2020). *Bayesian Active Learning for Wearable Stress and Affect Detection. 1*, 1–6. <http://arxiv.org/abs/2012.02702>
- Ramadhan, D., & Setiawan, E. B. (2019). Analisis Sentimen Program Acara di SCTV pada Twitter Menggunakan Metode Naive Bayes dan Support Vector Machine. ... *Telkomuniversity.Ac.Id*, 6(2), 9736–9743.
- Ramdhani, A. (2019). Studi Algoritma Linear Support Vector Machine pada Deteksi Ujaran Kebencian Berbahasa Indonesia. *Prosiding Seminar Nasional Teknoka*, 3(2502), 42. <https://doi.org/10.22236/teknoka.v3i0.2899>
- Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). Credit Card Fraud Detection Using AdaBoost and Majority Voting. *IEEE Access*, 6, 14277–14284. <https://doi.org/10.1109/ACCESS.2018.2806420>
- Ratino, Hafidz, N., Anggraeni, S., & Gata, W. (2020). Sentimen Analisis Informasi Covid-19 menggunakan Support Vector Machine dan Naïve Bayes. *Jurnal JUPITER*, 12(2), 1–11.
- Reksi, E., & Ernawati, I. (2020). Penggunaan Convolutional Neural Network Dalam Identifikasi Bahan Kulit Sapi dan Babi dengan Tensorflow. *Seinasi-Kesi*, 3(1), 140–146.
- Ren, R., Wu, D. D., & Wu, D. D. (2019). Forecasting stock market movement direction using sentiment analysis and support vector machine. *IEEE Systems Journal*, 13(1), 760–770. <https://doi.org/10.1109/JSYST.2018.2794462>
- Roscher, R., Bohn, B., Duarte, M. F., & Garcke, J. (2020). Explainable Machine Learning for Scientific Insights and Discoveries. *IEEE Access*, 8, 42200–42216. <https://doi.org/10.1109/ACCESS.2020.2976199>
- Trivedi, N. K., Simaiya, S., Lilhore, U. K., & Sharma, S. K. (2020). An efficient credit card fraud detection model based on machine learning methods. *International Journal of Advanced Science and Technology*, 29(5), 3414–3424.
- Utari, H., & Triana, Y. S. (2019). Sistem Informasi Monitoring Siswa Menggunakan SMS Gateway. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 3(3), 328–335. <https://doi.org/10.29207/resti.v3i3.916>
- Vallat, R. (2018). Pingouin: statistics in Python. *Journal of Open Source Software*, 3(31), 1026. <https://doi.org/10.21105/joss.01026>
- Veeramani, V. (2022). *Comparison of Frequent Fraud Activity Detection System by using the*

Comprehensive Analysis Method. October 2021.

- Winata, G. I., Kampman, O. P., & Fung, P. (2018). Attention-Based LSTM for Psychological Stress Detection from Spoken Language Using Distant Supervision. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2018-April*, 6204–6208. <https://doi.org/10.1109/ICASSP.2018.8461990>
- Yuanika, M., Ayu, N., & Nurul, D. (2021). Analisis perbandingan algoritma Naïve Bayes , k-Nearest Neighbor dan Neural Network untuk permasalahan class-imbalanced data pada kasus credit card fraud dataset Comparative analysis of the Naïve Bayes , k-Nearest Neighbor and Neural Network algorithms fo. *11*(2), 69–73.
- Zamachsari, F., & Puspitasari, N. (2021). Penerapan Deep Learning dalam Deteksi Penipuan Transaksi Keuangan Secara Elektronik. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *5*(2), 203–212. <https://doi.org/10.29207/resti.v5i2.2952>

LAMPIRAN

Lampiran 1. Kambing

Lampiran 2. Plagiarism

Cek Plagiarisme - Cek Plagiat Fr...

check-plagiarism.com/id/

100%
Konten unik

0%
Konten yang dijiplak

✓ COMPLETED
100%

Kalimat hasil bijak URL yang Cocok

Buat Laporan Plagiarisme

unik	Peningkatan penggunaan kartu kredit terjadi secara signifikan seiring dengan perkem....
unik	Namun peningkatan penggunaan tersebut tak terlepas dari kenaikan terjadinya penipuan....
unik	Hal ini tentunya sangat merugikan perusahaan pengelola keuangan setiap tahun.
unik	Dengan perkembangan teknologi yang ada maka kasus penipuan transaksi pada kartu kre....
unik	Pada penelitian ini penulis telah melakukan analisa pada.
unik	penggunaan kartu kredit untuk meminimalisir adanya penipuan pada kartu kredit.
unik	Selain itu juga pada penelitian ini digunakan beberapa metode yang digunakan untuk....
unik	yang baik, sehingga dapat diketahui metode mana yang paling baik digunakan untuk men....
unik	kredit.
unik	Dengan menggunakan empat metodologi penelitian yaitu Decision tree, Random forest,
unik	Maka dari hasil penelitian ini dapat disimpulkan bahwa metode Xgboost memiliki nil....
unik	kredit.

Kata Kunci Kata Kepadatan

1- word 2- words 3- words

Feedback

Type here to search

6:35 AM 7/7/2022

Lampiran 3. Surat Pernyataan**SURAT PERNYATAAN
TIDAK MELAKUKAN PLAGIARISME**

Yang bertanda tangan dibawah ini:

Nama : Dinda Anik Masruro

NPM : 1184003

Program Studi : D4 Teknik Informatika

Judul : Perbandingan Performansi Metode *Support Vector Machine* dan *Xgboost* pada Dataset Stress dan Pengaruhnya

Menyatakan bahwa:

1. Program Tugas Akhir saya ini adalah asli dan belum pernah diajukan untuk memenuhi kelulusan matakuliah Internship pada Program Studi D4 Teknik Informatika baik di Politeknik Pos Indonesia maupun di Perguruan Tinggi lainnya.
2. Program Tugas Akhir ini adalah murni gagasan, rumusan, dan penelitian saya sendiri tanpa bantuan pihak lain, kecuali arahan pembimbing.
3. Dalam Program Tugas Akhir ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan-penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya ini, serta sanksi lainnya sesuai dengan norma yang berlaku diperguruan tinggi lain.

Bandung, 13 Agustus 2022
Yang Membuat Pernyataan,



Dinda Anik Masruro
NPM:1184003