SVM zur Bestimmung von Handpositionen mittels EMG-Daten

Inhaltsverzeichnis

1. Einführung und Ziele

Dieses Projekt zielt darauf ab, ein Modell zur Klassifizierung von Handpositionen basierend auf Elektromyographie (EMG)-Daten zu entwickeln. Genauer gesagt wird eine Support Vector Machine (SVM) verwendet, um die elektrische Aktivität der Unterarmmuskeln zu analysieren und daraus verschiedene Handpositionen vorherzusagen. Das Endziel ist es, dieses Modell so zu optimieren, dass es auf einem Arduino eingesetzt werden kann, um Echtzeit-EMG-Daten zu klassifizieren und Handgesten zu erkennen.

1.1. Aufgabenstellung

Die Hauptaufgabe dieses Projekts besteht darin, eine zuverlässige und performante Support Vector Machine (SVM) zur Klassifikation von Handpositionen anhand von EMG-Daten zu entwickeln. Die Daten werden durch Elektroden erfasst, die die elektrische Aktivität der Muskeln messen. Anschließend werden die Rohdaten verarbeitet, statistische Merkmale extrahiert und das Modell trainiert.

- Implementierung einer Pipeline zur Vorverarbeitung der EMG-Daten (z.B. Filterung, Normalisierung)
- Training und Validierung eines SVM-Modells basierend auf den extrahierten Merkmalen
- Export des trainierten Modells
- Integration des Modells zur Echtzeit-Klassifikation auf einem Mikrocomputer

1.2. Qualitätsziele

- **Genauigkeit**: Das Modell muss eine hohe Klassifikationsgenauigkeit bei der Bestimmung der Handpositionen erreichen, um im praktischen Einsatz verlässlich zu funktionieren.
- **Leistung**: Das Modell muss effizient und schnell genug sein, um in Echtzeit auf einem ressourcenbeschränkten System zu laufen.
- **Robustheit**: Das System muss auch unter schwierigen Bedingungen wie variierenden EMG-Signalen oder Rauschen stabil arbeiten.
- **Portabilität**: Der erzeugte Code muss so optimiert sein, dass er leicht auf Mikrocomputer eingesetzt werden kann.

1.3. Stakeholder

Rolle	Kontakt	Erwartungshaltung
Projektleiter und Entwickler	Niklas Burczyk	Erfolgreiche Entwicklung eines SVM-Modells zur Handpositionsbestimmung basierend auf EMG-Daten. Umsetzung des Modells auf einem Arduino für Echtzeit-Anwendungen.
Endbenutzer	N/A	Ein funktionierendes System zur Erkennung von Handgesten in Echtzeit, das zuverlässig und leicht zu bedienen ist.

2. Randbedingungen

In diesem Abschnitt werden die architekturellen Randbedingungen beschrieben, die das Design und die Implementierung des Projekts beeinflussen. Diese Randbedingungen resultieren aus technischen, organisatorischen oder regulatorischen Vorgaben. Bestimmte Bedingungen sind zum jetzgen Stand des Projektes noch nicht final.

2.1. Technische Randbedingungen

1. **Einsatz eines Mikrocomputer**: Das trainierte SVM-Modell muss auf einem Mikrocomputer lauffähig sein. Dies stellt besondere Anforderungen an den Speicherbedarf, die Rechenleistung und die Effizienz des Modells. Ein ressourcenbeschränktes System wie der Arduino verfügt nur über begrenzte Rechenkapazitäten und Speicher, weshalb die Größe und Komplexität des Modells entsprechend optimiert werden müssen.
2. **Verwendung von EMG-Daten**: EMG-Daten sind in der Regel verrauscht und variieren stark zwischen einzelnen Personen und Aufnahmen. Daher ist eine robuste Vorverarbeitung der Daten erforderlich, einschließlich Filterung und Normalisierung, um verlässliche Merkmale für das SVM-Modell zu extrahieren.
3. **Echtzeitfähigkeit**: Die Klassifikation der EMG-Daten muss in Echtzeit erfolgen. Dies erfordert, dass sowohl die Vorverarbeitung der Daten als auch die Anwendung des Modells auf einem Mikrocomputer so optimiert werden, dass sie innerhalb der gegebenen Zeitrahmen ablaufen.

2.2. Organisatorische Randbedingungen

1. **Open-Source-Nutzung**: Das Projekt verwendet freie und offene Bibliotheken wie `scikit-learn` und `joblib`. Dies erleichtert die Verfügbarkeit und Wartung des Projekts, kann jedoch durch Lizenzbestimmungen und Updates beeinflusst werden.
2. **Projektstruktur**: Das Projekt muss in einer leicht verständlichen und dokumentierten Weise strukturiert sein, um anderen Entwicklern die Möglichkeit zu geben, es weiterzuentwickeln oder anzupassen. Der Quellcode und die Dokumentation sollten klar gegliedert sein, damit zukünftige Arbeiten oder Wartungen effizient durchgeführt werden können.

2.3. Regulatorische Randbedingungen

Bei der Entwicklung von Anwendungen, die auf medizinische Daten wie EMG-Daten zugreifen,

könnten Datenschutz- und Sicherheitsanforderungen relevant sein. Auch wenn dieses Projekt in erster Linie als Entwicklungs- und Prototyping-Projekt betrachtet wird, sollten in zukünftigen Implementierungen mögliche gesetzliche Anforderungen in Betracht gezogen werden, insbesondere im Hinblick auf den Schutz persönlicher Daten.

3. Kontextabgrenzung

Dieser Abschnitt beschreibt die fachlichen und technischen Schnittstellen des Systems, das in diesem Projekt entwickelt wird. Es geht um die Interaktion des Systems mit externen Komponenten und die Abgrenzung zu anderen Systemen.

3.1. Fachlicher Kontext

Das System ist darauf ausgelegt, Handgesten anhand von EMG-Daten zu erkennen und diese in Echtzeit zu klassifizieren. Der Hauptprozess beginnt mit der Datenerfassung über Elektroden am Unterarm, gefolgt von der Datenverarbeitung und der Klassifikation der Handpositionen (z.B. Faust geschlossen, Faust offen, Präzisionsgriff). Das Ergebnis der Klassifikation wird entweder auf einem externen Display angezeigt oder zur Steuerung eines anderen Geräts verwendet (z.B. einer Prothese oder eines Roboters).

Fachliche Schnittstellen:

Externes System	Schnittstellenname	Beschreibung
EMG-Sensoren	Sensor-Input	Liefert die Roh-EMG-Daten vom Unterarm, die als Eingabe für das System verwendet werden.
Externe Anzeige/Roboter	Klassifikationsergebnis	Gibt die klassifizierte Handposition als Steuerbefehl oder Information an externe Systeme aus (z.B. Arduino-Display, Robotersteuerung).

Erläuterung der externen fachlichen Schnittstellen:

- Die EMG-Sensoren erfassen die elektrische Aktivität der Muskeln und liefern Rohdaten. Diese Daten werden im System verarbeitet und zur Klassifikation verwendet.
- Die Ausgabe des Systems ist die erkannte Handposition, die entweder an eine externe Anzeige ausgegeben oder zur Steuerung eines externen Geräts (wie z.B. eines Roboters) verwendet wird.

3.2. Technischer Kontext

Das System wird auf einem Mikrocomputer (Raspberry Pi) ausgeführt und verwendet eine Reihe von Tools zur Modellentwicklung und Vorverarbeitung. Technisch gesehen umfasst das System mehrere Hauptkomponenten, darunter die Verarbeitung der EMG-Daten, die Extraktion von Merkmalen, das Trainieren des Modells sowie den Export des Modells. Der Arduino dient zur Aufnahme der EMG-Daten. Diese werden über die Serielle Schnittstelle an den Raspberry Pi übermittelt, wo diese weiter verabeitet werden.

Technische Schnittstellen:

Externe Komponente	Schnittstellenname	Beschreibung
EMG-Sensoren	ADC (Analog-Digital-Wandler)	Die analogen EMG-Signale werden vom Arduino über einen ADC in digitale Daten umgewandelt.
SVM-Modell	SVM-Model	Das in Python entwickelte und exportierte SVM-Modell wird als joblib-Datei auf dem Mikrocomputer eingebunden.
Externes Gerät/Anzeige	UART/Serial	Die erkannte Handposition wird über die serielle Schnittstelle des Arduino an ein externes Gerät oder eine Anzeige gesendet.

Mapping fachliche auf technische Schnittstellen:

- Die **EMG-Sensoren** (OLIMEX SHIELD-EKG-EMG) liefern Daten an den **ADC des Arduino** (technische Schnittstelle), der diese Daten digitalisiert.
- Die **Klassifikationsergebnisse** (fachliche Schnittstelle) werden über **UART/Serial** (technische Schnittstelle) an externe Geräte oder Anzeigen gesendet.

4. Lösungsstrategie

Die Lösungsstrategie dieses Projekts basiert auf der Kombination von maschinellem Lernen (Support Vector Machine) und der Verarbeitung von EMG-Daten, um Handpositionen in Echtzeit auf einem Mikrocomputer (Raspberry Pi 4) zu klassifizieren. Im Folgenden werden die wichtigsten Aspekte der Lösungsstrategie beschrieben.

4.1. Datenvorverarbeitung

Der erste Schritt der Lösungsstrategie besteht in der Erfassung und Vorverarbeitung der EMG-Daten, die von Elektroden am Unterarm gesammelt werden. Die Daten werden über einen Analog-Digital-Wandler (ADC) in digitale Signale umgewandelt. Anschließend werden die Rohdaten normalisiert und gefiltert, um Rauschen zu minimieren und nur die relevanten Signale für die Handgesten zu extrahieren.

1. **Bandpassfilterung**: Um den Bereich der Frequenzen, die mit Muskelbewegungen zusammenhängen, zu isolieren, wird ein Bandpassfilter auf die EMG-Daten angewendet. Dies reduziert das Rauschen und verbessert die Datenqualität für die nachfolgende Klassifikation.
2. **Segmentierung**: Die gefilterten Daten werden in Zeitfenster segmentiert, um die relevanten Merkmale für jedes Fenster zu extrahieren.

4.2. Merkmalsextraktion

Nach der Vorverarbeitung werden statistische Merkmale aus den EMG-Daten extrahiert. Diese Merkmale dienen als Grundlage für das Training des Support Vector Machine (SVM)-Modells.

1. **Merkmale**: Zu den wichtigsten Merkmalen, die aus den EMG-Daten extrahiert werden, gehören der Mittelwert, die Varianz und andere statistische Kennzahlen, die für jede Zeitsequenz berechnet werden.
2. **Merkmalsauswahl**: Um die Komplexität des Modells zu reduzieren und die Leistung zu maximieren, werden nur die aussagekräftigsten Merkmale für das Training der SVM verwendet.

4.3. Modellentwicklung und -training

Das SVM-Modell wird auf Basis der extrahierten Merkmale trainiert, um verschiedene Handpositionen (z.B. Faust geschlossen, Faust offen, Präzisionsgriff) zu klassifizieren.

1. **Training**:
   - Das SVM-Modell wird mit einem bestehenden Datensatz trainiert, der verschiedene Handpositionen und die entsprechenden EMG-Daten enthält.
   - Während des Trainingsprozesses wird die Genauigkeit des Modells kontinuierlich validiert und optimiert, um die bestmögliche Leistung zu erzielen.
2. **Modellbewertung**:
   - Die Leistung des Modells wird mit Metriken wie Genauigkeit, Präzision und Recall bewertet, um sicherzustellen, dass das Modell in der Lage ist, die verschiedenen Handgesten zuverlässig zu klassifizieren.

4.4. Modell-Export und Integration auf Mikrocomputer

Ein zentraler Teil der Lösungsstrategie ist die Implementierung des Modells, um die Klassifikation in Echtzeit durchzuführen.

Aktueller Stand: Derzeit wird das trainierte SVM-Modell als `.pkl`-Datei (Pickle-Format) gespeichert.

1. **Modellexport (aktuell)**:
   - Aktuell wird das trainierte SVM-Modell mit der Python-Bibliothek `joblib` als `.pkl`-Datei gespeichert. Diese Datei kann in anderen Python-Projekten wiederverwendet oder weiterverarbeitet werden: `python import joblib joblib.dump(svm_model, 'svm_model.pkl')`

5. Bausteinsicht

5.1. Whitebox Gesamtsystem

Begründung

Das Gesamtsystem ermöglicht die Echtzeitklassifikation von Handpositionen basierend auf EMG-Daten. Die modulare Architektur sorgt für Flexibilität und einfache Wartung.

Enthaltene Bausteine

- **Datenakquisitionsmodul**: Erfasst EMG-Daten über Elektroden.

- **Datenverarbeitungsmodul**: Führt die Vorverarbeitung der EMG-Daten durch, einschließlich Normalisierung und Filterung.
- **Merkmalextraktionsmodul**: Extrahiert relevante Merkmale aus den vorverarbeiteten Daten.
- **SVM-Modell**: Klassifiziert die Handpositionen basierend auf den extrahierten Merkmalen.
- **Raspberry Pi - Integration**: Verwendet das erzeugte SVM-Model und berechnet die Vorhersage in Echtzeit, über einen kurzen Zeitraum von gesammelten EMG-Daten

5.1.1. Datenakquisitionsmodul

Zweck/Verantwortung: Erfassung der EMG-Daten über Elektroden am Unterarm.

5.1.2. Datenverarbeitungsmodul

Zweck/Verantwortung: Vorverarbeitung der EMG-Daten, um Rauschen zu minimieren und relevante Informationen zu extrahieren.

5.1.3. Merkmalextraktionsmodul

Zweck/Verantwortung: Extraktion relevanter statistischer Merkmale aus den vorverarbeiteten EMG-Daten.

5.1.4. SVM-Modell

Zweck/Verantwortung: Klassifikation der Handpositionen basierend auf den extrahierten Merkmalen.

6. Querschnittliche Konzepte

6.1. Datenverarbeitung

Die Datenverarbeitung ist ein zentraler Aspekt dieses Projekts. Sie umfasst die Schritte der Normalisierung, Segmentierung und Filterung der EMG-Daten, um sicherzustellen, dass die verwendeten Daten für die Klassifikation geeignet sind. Ein solider Datenverarbeitungsprozess ist entscheidend für die Genauigkeit und Zuverlässigkeit des Modells.

6.2. Modelltraining und -bewertung

Das Modelltraining ist der Prozess, bei dem das SVM-Modell auf den vorbereiteten EMG-Daten trainiert wird. Dies umfasst das Erlernen von Mustern in den Daten, die mit verschiedenen Handpositionen korrelieren. Die Bewertung des Modells erfolgt anhand von Metriken wie Genauigkeit, Präzision und Recall, um sicherzustellen, dass das Modell gut funktioniert und generalisierbar ist.

6.3. Portabilität und Integrationsfähigkeit

Die Portabilität ist ein bedeutendes Konzept, da das entwickelte Modell auf verschiedenen Hardwareplattformen (wie Raspberry Pi) betrieben werden soll. Die Integrationsfähigkeit gewährleistet, dass das System problemlos in andere Anwendungen oder Umgebungen integriert werden kann, wodurch die Flexibilität und der Nutzen des Projekts erhöht werden.

6.4. Wartbarkeit

Wartbarkeit bezieht sich auf die Leichtigkeit, mit der das System aktualisiert, angepasst oder repariert werden kann. Eine modulare Codebasis, umfassende Dokumentation und automatisierte Tests sind entscheidend, um sicherzustellen, dass das System auch nach der ursprünglichen Implementierung leicht zu warten ist.

7. Architekturentscheidungen

7.1. 1. Wahl des Machine Learning Modells

Entscheidung: Es wurde entschieden, eine Support Vector Machine (SVM) für die Klassifikation der Handpositionen auf Basis von EMG-Daten zu verwenden.

Begründung: SVMs haben sich in der Vergangenheit als effektiv bei der Verarbeitung von hochdimensionalen Daten erwiesen und sind gut geeignet für Klassifikationsprobleme mit klaren Margen zwischen den Klassen. Die Fähigkeit der SVM, nichtlineare Entscheidungsgrenzen zu modellieren, ist vorteilhaft, da die Beziehung zwischen EMG-Daten und Handpositionen komplex sein kann.

7.2. 2. Datenverarbeitung und -vorverarbeitung

Entscheidung: Die EMG-Daten werden vor der Klassifikation normalisiert, segmentiert und bandpassgefiltert.

Begründung: Normalisierung ist erforderlich, um die Daten auf einen gemeinsamen Maßstab zu bringen und die Auswirkungen von Rauschen zu minimieren. Segmentierung ermöglicht eine detailliertere Analyse der EMG-Signale und fördert die Extraktion relevanter Merkmale. Bandpassfilterung hilft, nur die Frequenzen zu extrahieren, die für die Handpositionsklassifikation relevant sind, und reduziert z.B. ein 50 Hz Netzbrummen.

7.3. 3. Verwendung von Python für die Modellentwicklung

Entscheidung: Python wurde als Hauptprogrammiersprache für die Entwicklung des SVM-Modells und die Datenverarbeitung gewählt.

Begründung: Python bietet eine Vielzahl von leistungsstarken Bibliotheken (z.B. `scikit-learn`,

`numpy`, `pandas`), die die Entwicklung von Machine Learning-Modellen und die Datenmanipulation erheblich erleichtern. Die große Community und die umfangreiche Dokumentation machen es einfacher, Unterstützung und Ressourcen zu finden.

7.4. 4. Export des Modells für die Verwendung auf dem Raspberry Pi

Entscheidung: Das trainierte SVM-Modell wird als `.pkl`-Datei gespeichert.

Begründung: Die Verwendung von `joblib` zur Speicherung des Modells ermöglicht eine einfache Wiederverwendbarkeit und Integration in andere Python-Projekte.

7.5. 5. Modularer Codeansatz

Entscheidung: Der Code wurde modular strukturiert, um die Wartbarkeit und Erweiterbarkeit zu fördern.

Begründung: Ein modularer Ansatz erleichtert die Anpassung und Erweiterung des Systems, z.B. wenn neue Handpositionen hinzugefügt oder Änderungen an den Algorithmen vorgenommen werden müssen. Eine klare Trennung von Funktionen (z.B. Datenverarbeitung, Modelltraining, Klassifikation) erhöht die Lesbarkeit und Testbarkeit des Codes.

7.6. 6. Einsatz von Docker Compose für Containerisierung auf Linux-Systemen

Entscheidung: Das gesamte System wird als Docker Compose Stack bereitgestellt und kann in einem Container-basierten Umfeld auf Linux-Systemen gestartet werden.

Begründung: Die Containerisierung ermöglicht eine einfache und konsistente Bereitstellung des Systems unabhängig von der zugrundeliegenden Linux-Distribution oder Umgebung. Durch Docker Compose können alle benötigten Komponenten (z.B. Datenverarbeitung, Modellservice, eventuell weitere Services) als verbundene Container orchestriert werden, was den Betrieb vereinfacht und Skalierbarkeit sowie Wartbarkeit verbessert. Dies unterstützt insbesondere die einfache Integration in bestehende Linux-basierte Infrastruktur und erleichtert automatisierte Deployments und Updates.

8. Qualitätsanforderungen

8.1. Qualitätsbaum

Der Qualitätsbaum zeigt die hierarchische Struktur der Qualitätsanforderungen. Die obersten Ziele können in spezifischere Qualitätsmerkmale unterteilt werden, um sicherzustellen, dass alle Aspekte abgedeckt sind.

- **Funktionale Qualität**

- Korrektheit: Das System muss die Handpositionen zuverlässig klassifizieren.
- Vollständigkeit: Das System sollte alle vorgesehenen Handpositionen erfassen können.
- **Nicht-funktionale Qualität**
- Zuverlässigkeit: Das System sollte eine hohe Verfügbarkeit haben und in 85 % der Fälle korrekt klassifizieren.
- Leistung: Die Klassifikation sollte in Echtzeit erfolgen (z.B. innerhalb von 100 ms).
- **Technische Qualität**
- Wartbarkeit: Der Code sollte modular und gut dokumentiert sein, um zukünftige Anpassungen zu erleichtern.
- Portabilität: Das System sollte auf verschiedenen Hardwareplattformen (z.B. Arduino) problemlos betrieben werden können.

8.2. Qualitätsszenarien

1. **Szenario 1: Echtzeit-Klassifikation der Handpositionen**
   - **Beschreibung**: Das System erfasst EMG-Daten in Echtzeit, klassifiziert die Handposition und gibt die Ergebnisse über die serielle Schnittstelle aus.
   - **Qualitätsmerkmale**:
   - Leistung: <100 ms Reaktionszeit für die Klassifikation.
   - Zuverlässigkeit: 95 % Genauigkeit bei der Klassifikation der Handpositionen.
   - **Akzeptanzkriterien**: Das System muss in 90 % der Fälle innerhalb von 100 ms eine Klassifikation liefern.
2. **Szenario 2: Robustheit bei variierenden EMG-Daten**
   - **Beschreibung**: Das System soll auch unter verschiedenen Bedingungen (z.B. unterschiedlicher Muskelspannung, Rauschen) stabil klassifizieren.
   - **Qualitätsmerkmale**:
   - Robustheit: Das Modell soll nicht von der Qualität der EMG-Daten abhängen.
   - Fehlerresistenz: Das System sollte in der Lage sein, auch bei Rauschen oder unvollständigen Daten eine Klassifikation vorzunehmen.
   - **Akzeptanzkriterien**: Die Klassifikation sollte auch bei 20 % Rauschen in den EMG-Daten eine Genauigkeit von mindestens 85 % erreichen.
3. **Szenario 3: Benutzerfreundlichkeit der Schnittstelle**
   - **Beschreibung**: Die Benutzeroberfläche des Arduino-Programms sollte einfach zu bedienen sein und klare Rückmeldungen geben.
   - **Qualitätsmerkmale**:
   - Benutzbarkeit: Die Benutzeroberfläche sollte in weniger als 5 Minuten erlernt werden können.
   - Zugänglichkeit: Informationen über den aktuellen Status der Klassifikation sollten jederzeit verfügbar sein.

- **Akzeptanzkriterien**: Mindestens 80 % der Benutzer sollten die Schnittstelle als intuitiv und benutzerfreundlich bewerten.

Die Qualitätsszenarien sind als Szenarien anzusehen, die in späteren Projekten umgesetzt werden könnten.

9. Risiken und technische Schulden

9.1. Risiken

1. **Datenqualität**: Die Genauigkeit und Zuverlässigkeit des SVM-Modells hängen stark von der Qualität der EMG-Daten ab. Rauschen und Artefakte in den Daten können zu fehlerhaften Klassifikationen führen.
2. **Modellüberanpassung**: Wenn das SVM-Modell zu stark auf die Trainingsdaten angepasst ist, kann dies zu einer schlechten Generalisierbarkeit auf neue, unbekannte Daten führen. Dies ist besonders kritisch, wenn das System in realen Anwendungsszenarien eingesetzt wird.
3. **Echtzeitanforderungen**: Die Klassifikation der Handpositionen muss in Echtzeit erfolgen, was eine effiziente Verarbeitung erfordert. Verzögerungen können die Benutzererfahrung erheblich beeinträchtigen und die Anwendung unbrauchbar machen.
4. **Hardwareabhängigkeit**: Das System ist auf spezifische Hardwarekomponenten (z.B. Arduino, Elektroden) angewiesen. Probleme mit der Hardware oder Inkompatibilitäten können die Leistung des Systems beeinträchtigen.

9.2. Technische Schulden

1. **Dokumentation**: Unzureichende Dokumentation des Codes und der Architektur kann die Wartung des Systems erschweren. Eine klare und umfassende Dokumentation sollte fortlaufend aktualisiert werden, um technische Schulden zu vermeiden.
2. **Modularität des Codes**: Falls der Code nicht modular und gut strukturiert ist, kann dies die Erweiterung und Anpassung des Systems in der Zukunft erschweren. Technische Schulden in der Codebasis können die Implementierung neuer Funktionen oder Änderungen komplizierter machen.
3. **Abhängigkeiten von Bibliotheken**: Das System ist möglicherweise auf spezifische externe Bibliotheken angewiesen (z.B. `joblib`, `sklearn-porter`). Änderungen oder das Eintreten von Inkompatibilitäten in diesen Bibliotheken können Probleme verursachen. Regelmäßige Updates und Überprüfungen sollten eingeplant werden.
4. **Technologische Überalterung**: Die verwendeten Technologien und Tools könnten in Zukunft veraltet sein. Ein fortlaufendes Monitoring der Technologien und eine rechtzeitige Migration auf aktuellere Standards sind notwendig, um technische Schulden zu vermeiden.

10. Glossar

Begriff	Definition
EMG (Elektromyographie)	*Ein Verfahren zur Messung der elektrischen Aktivität der Muskeln, das zur Analyse und Klassifikation von Muskelaktivitäten verwendet wird.*
SVM (Support Vector Machine)	*Ein überwachtes Machine-Learning-Modell, das für Klassifikations- und Regressionsaufgaben verwendet wird und sich auf die Maximierung der Trennung zwischen verschiedenen Klassen konzentriert.*
Kernel	*Eine Funktion, die es SVMs ermöglicht, in höherdimensionalen Räumen zu arbeiten, ohne dass die Daten tatsächlich dorthin transformiert werden müssen. Dadurch kann eine nichtlineare Trennung der Daten ermöglicht werden.*
Normalisierung	*Der Prozess der Anpassung der Werte in einem Datensatz auf einen einheitlichen Maßstab, um Verzerrungen zu vermeiden und die Analyse zu erleichtern.*
Segmentierung	*Die Aufteilung eines kontinuierlichen Datensatzes (z. B. EMG-Daten) in kleinere, analysierbare Teile oder Abschnitte, die jeweils bestimmte Merkmale enthalten.*
Bandpassfilter	*Ein elektronischer Filter, der nur Frequenzen innerhalb eines bestimmten Bereichs (Band) durchlässt und Frequenzen außerhalb dieses Bereichs dämpft.*
Modellbewertung	*Der Prozess der Analyse der Leistung eines Machine-Learning-Modells anhand von verschiedenen Metriken, um seine Genauigkeit, Präzision und Zuverlässigkeit zu bestimmen.*
Portabilität	*Die Fähigkeit eines Softwaremodells, auf verschiedenen Plattformen oder Systemen zu laufen, ohne dass signifikante Änderungen am Code erforderlich sind.*
Wartbarkeit	*Die Leichtigkeit, mit der ein Software-System aktualisiert, angepasst oder repariert werden kann, häufig gefördert durch sauberen, gut strukturierten und dokumentierten Code.*
Feature (Merkmal)	*Eine einzelne messbare Eigenschaft oder Charakteristik eines Datensatzes, die zur Klassifikation oder Regression verwendet wird, z. B. Mittelwert, Varianz oder Frequenzinhalt der EMG-Daten.*
Hyperparameter	*Parameter, die vor dem Training eines Machine-Learning-Modells festgelegt werden und die Struktur und das Verhalten des Modells beeinflussen, z. B. die Wahl des Kernels in einer SVM oder die Anzahl der Bäume in einem Random Forest.*
Training	*Der Prozess, bei dem ein Machine-Learning-Modell auf einem Datensatz trainiert wird, um Muster zu lernen und Vorhersagen zu treffen. Dies geschieht durch Anpassung der Modellparameter auf Basis der Eingabedaten und deren zugehörigen Labels.*

Begriff	Definition
Validierung	*Der Prozess der Beurteilung eines Modells während oder nach dem Training, um seine Leistung auf einem unabhängigen Datensatz zu testen und Überanpassung zu vermeiden.*
Überanpassung (Overfitting)	*Ein Phänomen, bei dem ein Modell zu komplex ist und die Trainingsdaten zu genau lernt, was zu schlechter Leistung bei neuen, unsichtbaren Daten führt.*