به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش پروژه پایانی

Classify gestures by reading muscle activity

طبقه بندی ژست ها با خواندن فعالیت های ماهیچه ای

[ایمان فکری اسکی-محمد حسین بیاتی]

[۹۹۲۴۶۹۳-۹۹۲۹۰۸۳]

استاد : آقای دکتر مهدی علیاری

بهمن ماه ۱۴۰۲

فهرست مطالب

عنوان شماره صفحه

چکیده :

در این گزارش، داده‌های جمع‌آوری شده از بازوی ربات را با استفاده از روش‌های هوش مصنوعی ارزیابی و طبقه‌بندی می‌کنیم. از شبکه‌های عصبی، درخت تصمیم، و روش‌های هوشمند دیگر برای بهبود دقت و کارایی در طبقه‌بندی داده‌ها استفاده می‌کنیم. نتایج نشان می‌دهند که استفاده از این روش‌ها، امکان بهبود عملکرد و دقت در تحلیل داده‌های بازوی ربات را فراهم می‌کند.

مقدمه :

در دنیای امروز، ربات‌ها به عنوان یکی از فناوری‌های کلیدی هوش مصنوعی پیشرفت یافته‌اند و در مختلف زمینه‌ها، از جمله صنعت، پزشکی، و خدمات خودکار، به کار گرفته می‌شوند. بازوهای رباتیکی از جمله اجزای اساسی این ربات‌ها محسوب می‌شوند که در انجام وظایف مختلفی از جمله جابه‌جایی، جمع‌آوری داده و انجام عملیات پیچیده مشغول به کارند.

در این سیاق، جمع‌آوری و تحلیل داده‌های حاصل از عملکرد بازوی ربات اهمیت زیادی پیدا می‌کند. در این گزارش، تمرکز بر روی طبقه‌بندی داده‌ها با استفاده از روش‌های هوش مصنوعی است. با بهره‌گیری از شبکه‌های عصبی، درخت تصمیم و سایر روش‌های هوشمند، سعی در بهبود دقت و اطمینان در تحلیل داده‌های بازوی ربات داریم.

هدف این گزارش، ارتقاء کارایی در طبقه‌بندی داده‌های بازوی ربات با استفاده از تکنیک‌های پیشرفته هوش مصنوعی است. این پروژه نه‌تنها به بهبود عملکرد بازوی ربات کمک می‌کند بلکه به جامعه تحقیقاتی در زمینه هوش مصنوعی نیز افزوده می‌شود.

بخش۱: معرفی داده یا سیستم

معرفی داده یا سیستم :

این سیستم از چندین جزء تشکیل شده است. ابتدا یک حسگر فعالیت عضلانیEMG(الکترومیوگرافی)به یک اپلیکیشن کاربری اندروید Things Android متصل میشود. اپلیکیشن دادهها را جمع آوری میکند، سپس یک سرور یک مدل Tensorflow را به طور خاص برای این کاربر ایجاد میکند. پس از آن، مدل قابل دانلود است و میتواند در دستگاه اجرا شود تا موتورها یا قطعات دیگر را کنترل کند.

https://github.com/cyber-punk-me

این مجموعه داده میتواند برای نقشه برداری حرکات عضلانی باقیمانده کاربر به اقدامات خاص پروتز مانند باز/بسته کردن دست یا چرخاندن مچ استفاده شود. چهار دسته حرکت از دستبند MYO با کمک اپلیکیشن مورد نظر ضبط شدند. دستبند MYO دارای ۸ حسگر است که بر روی سطح پوست قرار گرفتهاند و هرکدام فعالیت الکتریکی تولید شده توسط عضلات زیر پوست را اندازه گیری میکنند.

هر خط داده شامل ۸ خواندن متوالی از همه ۸ حسگر است، بنابراین ۶۴ ستون داده EMG دارد. آخرین ستون نشان دهنده حرکتی است که در حین ضبط داده انجام شده است (دسته هاای ۰ تا ۳) . بنابراین هر خط ساختار زیر را دارد:

```
[8sensors][8sensors][8sensors][8sensors][8sensors][8sensors][8sensors][8sensors][GESTURE_CLASS]
```

داده ها با نرخ ۲۰۰ هرتز ضبط شده اند، که به این معناست که هر خط زمان ضبط (۱/۲۰۰)*۸ ثانیه یا همان ۴۰ میلی ثانیه است.

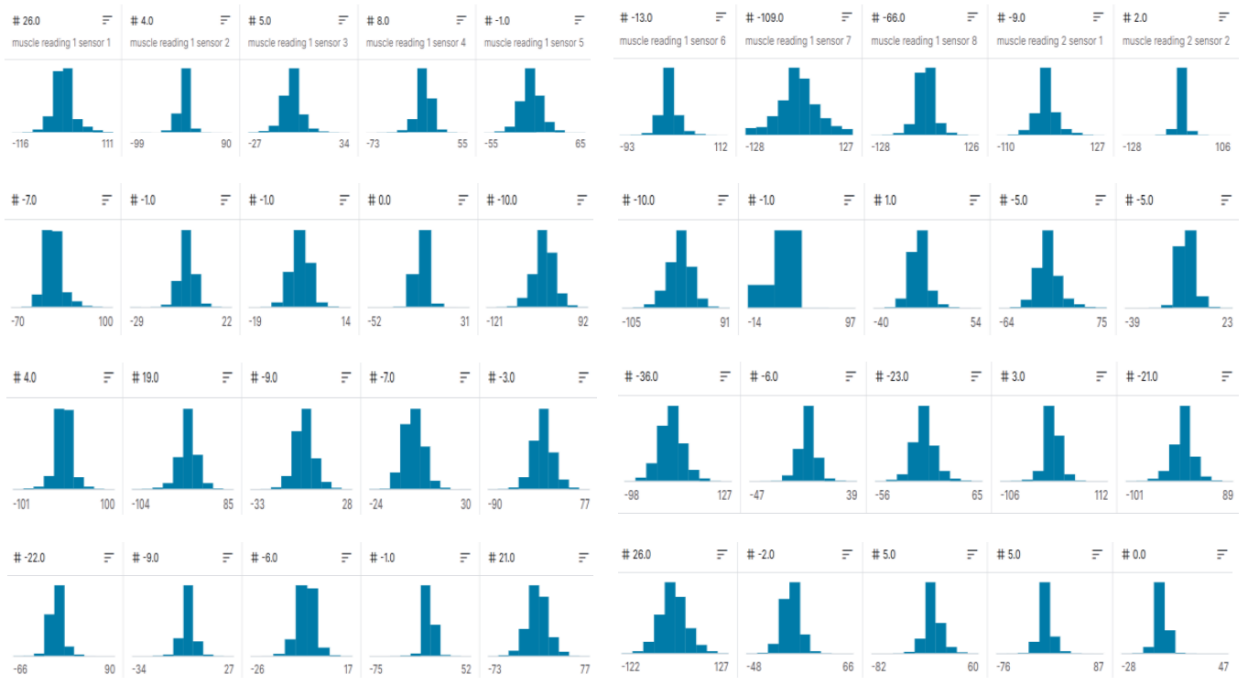
یک طبقه بندی با ورودی ۶۴ عدد، یک کلاس حرکت (۰-۳) را پیشبینی خواهد کرد .کلاس های حرکت به شرح زیر هستند: سنگ - ۰ قیچی - ۱ کاغذ - OK۲ -۳ حرکات سنگ، کاغذ، قیچی مشابه بازی با همین نام هستند، و حرکت OK نشاندهنده این است که انگشت اشاره با انگشت شست دسته می شود و بقیه انگشتان باز است. این حرکات تقریباً به صورت تصادفی انتخاب شده اند.

هر حرکت به مدت ۲۰ ثانیه و ۶ بار ضبط شده است. هر بار ضبط، با حرکت آماده و نگهداشته شده ای شروع شد. ضبط در حالی که حرکت هنوز نگه داشته می شد متوقف شد. در مجموع، هر حرکت به مدت ۱۲۰ثانیه در یک

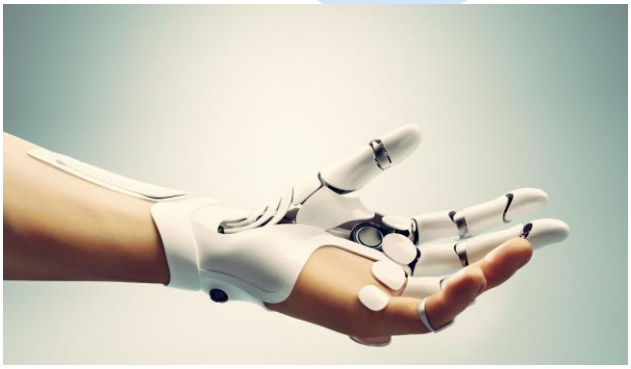
موقعیت ثابت نگه داشته شده است. همه این ضبط ها از همان پایین ساعد راست در یک بازه زمانی کوتاه انجام شده اند. هر ضبط از یک کلاس حرکت مشخص به یک فایل **CSV**. با یک نام متناظر (۰-۳) ادغام شده است .
لینک داده های مورد نظر:

https://www.kaggle.com/datasets/kyr7plus/emg-4/download?datasetVersionNumber=2

نمونه توزیع داده های هر کلاس به صورت زیر می باشد:



بخش۲: معرفی فرآیند و روش



معرفی فرآیند و روش :

هدفی که از انجام این پروژه داریم این است که ، سعی می کنیم تا با استفاده از الگوریتم هایی که تا به حال در درس آموخته ایم مانند MLP و یا

decision tree و یا الگوریتم های دیگر موجود ، به کار طبقه بندی و یا classification بپردازیم. به طور کلی روش هایی که می خواهیم از آن ها استفاده کنیم شبکه عصبی و یا MLP و درخت تصمیم و SVM می باشد که با استفاده از این سه متد می خواهیم از داده های موجودی که در اختیار داریم نوع ژست ماهیچه ای را مشخص کنیم. همان طور که در قسمت های قبل نیز گفته شده است، با استفاده از داده های فایل ها می توانیم با استفاده از ویژگی هایی که در اختیار داریم نوع ژست ماهیچه را از میان چهار حالت سنگ یا کاغذ یا قیچی و یا اوکی ، تشخیص دهیم.

شرح متد های استفاده شده :

شبکه‌های عصبی مکرر (RNN) یک نوع از مدل‌های هوش مصنوعی هستند که برای کار با داده‌های دنباله‌ای و زمانی مناسب هستند. این شبکه‌ها از ساختاری بازگشتی برخوردار هستند که اجازه می‌دهد به اطلاعات گذشته در تحلیل دنباله‌ها توجه شود. معمولاً در وظایفی که وابستگی به ترتیب زمانی داده‌ها وجود دارد (مثل زبان طبیعی، پیش‌بینی سری زمانی و تحلیل حالات زمانی)، از RNN استفاده می‌شود.

تفاوت اصلی RNN با شبکه‌های عصبی معمولی در این است که RNN دارای حلقه (یا بازگشت) است که اطلاعات را از مرحله یک به مرحله دیگر منتقل می‌کند. این حلقه به شبکه امکان می‌دهد اطلاعات گذشته را در حین آموزش و پیش‌بینی متغیرهای آینده مد نظر داشته باشد.

هر نرون در یک لحظه زمانی خاص در یک RNN یک وضعیت (state) دارد که می‌تواند به عنوان حاصل از ورودی‌های گذشته و حالت قبلی آن نرون توصیف شود. این وضعیت به عنوان حافظه کوتاه‌مدت شناخته می‌شود.

هر نرون در یک لحظه زمانی خاص همچنین یک خروجی تولید می‌کند که می‌تواند به عنوان پیش‌بینی یا نتیجه مورد نظر در مسئله مورد استفاده قرار گیرد.

مشکل اصلی RNN به نام مشکل گمشدن گرادیان Vanishing Gradient Problem است که در زمان آموزش ممکن است باعث ضعف عملکرد RNN شود. برای حل این مشکل، معمولاً از نسخه‌های بهبود یافته مانند Long Short-Term Memory (LSTM) یا Gated Recurrent Unit (GRU) استفاده می‌شود که بهترین کنترل بر گرادیان‌ها را دارند و قابلیت حفظ اطلاعات برای مدت زمان طولانی‌تری را دارا هستند.

ماشین بردار پشتیبان یا SVM (Support Vector Machine) یک الگوریتم یادگیری ماشینی است که به عنوان یک روش طبقه‌بندی و یا رگرسیون عمل می‌کند. هدف اصلی آن ایجاد یک صفحه (یا فضایی دیگر به ویژه در ابعاد بالاتر) که بین دسته‌های مختلف داده را جدا کند. این صفحه به عنوان صفحه جداکننده یا مرز تصمیم (Decision Boundary) نیز شناخته می‌شود.

۱ .حداکثر مارژین

SVM سعی دارد حداکثر مارژین (فاصله بین مرز تصمیم و نزدیک‌ترین نقاط داده به هر دو دسته) را بین دسته‌ها بیشینه کند. این مارژین نشان‌دهنده اطمینان الگوریتم از صحت تصمیم‌گیری در مقابل داده‌های تازه و نامعلوم است.

۲ .بردارهای پشتیبان

SVM از بردارهای پشتیبان (Support Vectors) استفاده می‌کند که دقیقاً بر روی مرز تصمیم قرار دارند و به تعیین آن صفحه تصمیم کمک می‌کنند. این بردارها اهمیت بسیاری در ایجاد مرز تصمیم دارند.

۳ .تابع هسته(Kernel Function)

SVM از توابع هسته برای تبدیل داده‌ها به فضایی با بعد بالاتر استفاده می‌کند. این توابع هسته به SVM امکان ایجاد مرزهای تصمیم پیچیده‌تر و مناسب برای داده‌های غیرخطی را می‌دهند. مثال‌هایی از توابع هسته عبارتند از Polynomial Kernel و.Radial Basis Function (RBF) Kernel

روش کار

۱ .آموزش(Training)

در مرحله آموزش، SVMبا استفاده از داده‌های مربوط به دو دسته، یک مدل را آموزش می‌دهد. هدف این مرحله ایجاد یک مرز تصمیم با حداکثر مارژین برای جداسازی دسته‌ها است.

۲ .پیش‌بینی(Prediction)

پس از آموزش، SVMمی‌تواند بر اساس مدل خود، برای داده‌های جدید پیش‌بینی انجام دهد و آن‌ها را در یکی از دسته‌ها قرار دهد.

SVM به دلیل توانایی در مقابله با داده‌های پرت و دارای ابعاد بالا، و همچنین امکان کنترل مارژین و استفاده از توابع هسته، در بسیاری از حوزه‌ها از جمله طبقه‌بندی تصاویر، متن‌ها، و بیوانفورماتیک مورد استفاده قرار می‌گیرد.

Random Forest یک الگوریتم ماشین لرنینگ (Machine Learning) است که برای مسائل طبقه‌بندی و رگرسیون استفاده می‌شود. این الگوریتم به عنوان یک مدل قوی و انعطاف‌پذیر شناخته می‌شود و بر روی تصمیم‌گیری مبتنی بر انبوه (Ensemble Learning) استوار است.

عملکرد

۱ .انتخاب نمونه‌ها

برای ساخت هر درخت تصمیم در جنگل تصادفی، نمونه‌ها به صورت تصادفی از داده‌های آموزشی انتخاب می‌شوند. این انتخاب تصادفی باعث تنوع بیشتر میان درخت‌ها می‌شود.

۲ .ساخت درخت تصمیم

برای هر درخت تصمیم، یک سری از ویژگی‌ها به صورت تصادفی انتخاب می‌شوند. سپس درخت تصمیم بر اساس این ویژگی‌ها و به صورت بازگشتی (Recursively) ساخته می‌شود. درخت تصمیم به گره‌های تصمیم و گره‌های برگ تقسیم می‌شود تا به نمونه‌ها را به دسته‌های مختلف تقسیم کند.

۳ .تصمیم‌گیری اکثریت

در هنگام پیش‌بینی، هر درخت به تصمیمی درباره نمونه جدید می‌رسد. نتایج این درخت‌ها ترکیب شده و تصمیم نهایی بر اساس اکثریت یا میانگین (برای مسائل رگرسیون) این نتایج گرفته می‌شود.

ویژگی‌های اصلیRandom Forest

مقاومت در برابر برازش زیاد(Overfitting)

جنگل تصادفی به دلیل اجتماعی از درخت‌های تصمیم، مقاومت خوبی در برابر برازش زیاد به داده‌های آموزشی دارد.

توانایی در مدیریت ویژگی‌ها

این الگوریتم به صورت تصادفی ویژگی‌ها را انتخاب می‌کند که به ایجاد تنوع کمک می‌کند و از اهمیت وزن‌دهی مناسب به ویژگی‌ها در تصمیم‌گیری استفاده می‌کند.

کارایی بالا

Random Forest به دلیل ترکیب چندین درخت تصمیم، در بسیاری از مسائل ماشین لرنینگ به عنوان یک مدل کارآمد و پرکاربرد شناخته می‌شود.

Random Forest در مسائلی مانند طبقه‌بندی تصویر، تشخیص اشیاء، و پیش‌بینی سری زمانی به خوبی عمل می‌کند.

SVCیا "Support Vector Classification" یک الگوریتم ماشین لرنینگ است که برای مسائل طبقه‌بندی (Classification) استفاده می‌شود و به عنوان یک نسخه از ماشین‌های بردار پشتیبان (SVM) در ادبیات ماشین لرنینگ شناخته می‌شود .

عملکرد الگوریتمSVC

۱ .حفظ مارژین

هدف اصلی SVC این است که یک مارژین بیشینه بین دسته‌های مختلف داده ایجاد کند. مارژین به عنوان فاصله بین مرز تصمیم و نقاط نزدیک‌تر به هر دو دسته مشخص می‌شود. این فاصله بیشینه حاکی از اطمینان مدل در پیش‌بینی داده‌های تازه و نامعلوم است.

۲ .دسته‌بندی با استفاده از بردارهای پشتیبان

SVC از بردارهای پشتیبان (Support Vectors) استفاده می‌کند که نقاطی هستند که به مارژین و مرز تصمیم بسیار نزدیک هستند. این بردارها تاثیر زیادی در تعیین مرز تصمیم دارند و مشخص می‌کنند که داده‌ها به کدام دسته تعلق دارند.

۳ .توابع هسته(Kernel Functions)

برای مسائل طبقه‌بندی غیرخطی، SVCاز توابع هسته استفاده می‌کند تا داده‌ها را به فضایی با بعد بالاتر تبدیل کند و مرز تصمیم پیچیده‌تری ایجاد کند. توابع هسته معمولاً به داده‌ها امکان می‌دهند در فضای غیرخطی مسئله طبقه‌بندی شوند.

۴ .کنترل پارامترها

SVC دارای پارامترهایی مانند C و gamma است که تاثیر زیادی در عملکرد الگوریتم دارند. پارامتر C نشان‌دهنده نرخ خطا در تطبیق مرز تصمیم و مارژین است، و پارامتر gamma تاثیر توابع هسته را کنترل می‌کند.

ویژگی‌های اصلیSVC

مناسب برای دسته‌بندی داده‌های خطی و غیرخطی:

SVC می‌تواند در مسائل طبقه‌بندی خطی و غیرخطی مؤثر باشد، امکانی که بسیاری از مسائل واقعی دارند.

حفظ مارژین بیشینه:

SVC با تلاش برای ایجاد یک مارژین بیشینه بین دسته‌ها، به دقت بالا و عملکرد قوی در مسائل طبقه‌بندی می‌رسد.

SVC در بسیاری از حوزه‌های کاربردی از جمله طبقه‌بندی تصویر، تشخیص چهره، و طبقه‌بندی متن با موفقیت استفاده می‌شود.

درخت تصمیم (Decision Tree) یک الگوریتم ماشین لرنینگ است که برای مسائل طبقه‌بندی و رگرسیون استفاده می‌شود. این الگوریتم با ساخت یک ساختار درختی از تصمیم‌ها، مسائل را به شکل سلسله مراتبی و سلسله‌مراتبی حل می‌کند. هر گره درخت یک تصمیم است که به یک ویژگی از داده اشاره دارد و هر شاخه از گره نشان‌دهنده یک گزینه یا تصمیم ممکن در مورد آن ویژگی است.

عملکردDecision Tree

۱ .انتخاب ویژگی

در هر گام ایجاد گره، الگوریتم انتخاب می‌کند که کدام ویژگی از داده برای تصمیم بعدی مورد استفاده قرار گیرد. این انتخاب بر اساس معیارهایی مانند اندازه مارژین (برای درخت‌های طبقه‌بندی) یا میزان کاهش واریانس (برای درخت‌های رگرسیون) صورت می‌گیرد.

۲ .تقسیم داده‌ها

بر اساس ویژگی انتخاب شده، داده‌ها به دو یا چند زیرمجموعه تقسیم می‌شوند. این تقسیم بر اساس مقادیر مختلف ویژگی انجام می‌شود.

۳ .تکرار مراحل ۱ و ۲

این فرآیند تکرار می‌شود برای هر زیرمجموعه حاصله، تا زمانی که یک شرط توقف (مانند عمق مشخص شده یا تعداد حداکثر گره‌ها) برقرار شود.

۴ .پیش‌بینی

در نهایت، هر برگ درخت با یک کلاس (در درخت‌های طبقه‌بندی) یا یک مقدار عددی (در درخت‌های رگرسیون) مرتبط می‌شود. هنگامی که یک داده جدید وارد می‌شود، با طی کردن مسیر درخت تصمیم، به یک برگ می‌رسیم و پیش‌بینی نهایی انجام می‌شود.

ویژگی‌های اصلی Decision Tree

قابل فهم و تفسیر

درخت تصمیم به دلیل ساختار سلسله مراتبی و انتخاب ویژگی‌های قابل فهم، به راحتی توسط افراد غیرتخصصی نیز قابل فهم است.

مناسب برای داده‌های با ویژگی‌های مختلف

Decision Tree می‌تواند با داده‌های با ویژگی‌های مختلف و همچنین داده‌های دسته‌ای و عددی کار کند.

مقاومت متوسط به برازش زیاد

الگوریتم‌های درخت تصمیم معمولاً مقاومت متوسطی در برابر برازش زیاد (Overfitting) دارند. این مقاومت می‌تواند با استفاده از روش‌هایی مانند تقریب میزان عمق یا استفاده از Random Forest افزایش یابد.

Decision Tree در زمینه‌های مختلفی از جمله طبقه‌بندی، رگرسیون، و حل مسائل تصمیم‌گیری به کار می‌رود.

متد هوشمند MLP به متدی اشاره دارد که از شبکه‌های عصبی چند لایه (MLP) برای حل مسائل هوش مصنوعی استفاده می‌کند .MLP به عنوان یک نوع از شبکه‌های عصبی عمیق شناخته می‌شود، که شامل حداقل سه لایه است: لایه ورودی، لایه مخفی(ها) و لایه خروجی. در MLP هر نورون یک وزن دارد و هر لینک (ارتباط بین نورون‌ها) با یک وزن مشخص می‌شود.

متد هوشمند MLP عمدتاً از الگوریتم‌های یادگیری ماشین برای بهینه‌سازی وزن‌ها به منظور دستیابی به یک مدل دقیق و توانمند استفاده می‌کند. این الگوریتم‌ها به صورت تکراری داده‌های آموزشی را به مدل معرفی کرده و با مقایسه خروجی مدل با مقدار مورد انتظار، وزن‌ها را به گونه‌ای تنظیم می‌کنند که خطا کمینه شود.

یکی از الگوریتم‌های یادگیری ماشین متد هوشمندMLP ، الگوریتم پس‌انتشار خطا (Backpropagation) است. در این الگوریتم، خطا بین خروجی مدل و مقدار مورد انتظار محاسبه می‌شود و سپس این خطا به عقب انتشار داده می‌شود تا وزن‌های هر لایه به‌روزرسانی شوند.

مزایای متد هوشمند MLP شامل اینکه این مدل‌ها می‌توانند الگوهای پیچیده و غیرخطی را نیز یاد بگیرند و در حل مسائل گوناگون مورد استفاده قرار گیرند. با این حال، نکته مهمی که باید در نظر گرفت این است که این مدل‌ها ممکن است در مواجهه با داده‌های کم یا در مواقعی که داده‌ها ناپایدار یا ناکافی هستند، به مشکل بخورند.

پیاده سازی روش های بیان شده :

چهار فایل به عنوان دیتاست داریم که هر کدام مربوط به یک کلاس خاصی می باشد. در ادامه باید چهار فایل مربوطه را در محیط مورد نظر وارد کنیم.

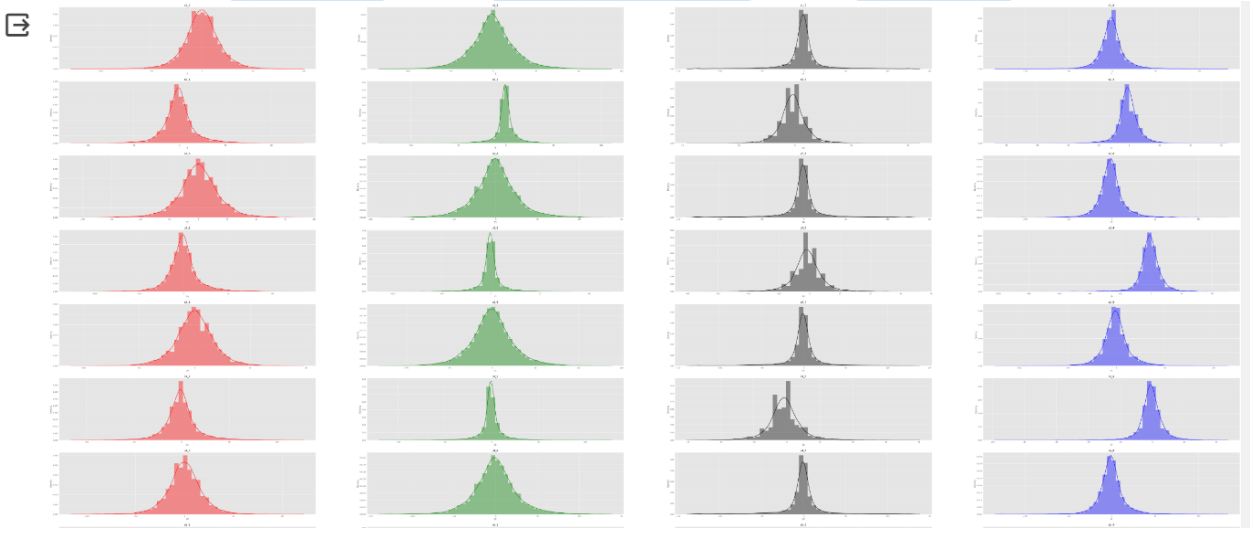
```
#original emg data
df0_emg = pd.read_csv("/content/0.csv", header=None )
df1_emg = pd.read_csv("/content/1.csv", header=None )
df2_emg = pd.read_csv("/content/2.csv", header=None )
df3_emg = pd.read_csv("/content/3.csv", header=None )
df_emg = pd.concat([df0_emg,df1_emg,df2_emg,df3_emg], axis = 0)
#concatenate the dataframe
df0_emg , df1_emg , df2_emg , df3_emg , df_emg
```

سپس می توانیم داده های را با یکدیگر در داخل یک فایل ترکیب کنیم :

	0	1	2	3	4	5	6	7	8	9	...	55	56	57	58	59	60	61	62	63	64
0	26.0	4.0	5.0	8.0	-1.0	-13.0	-109.0	-66.0	-9.0	2.0	...	-28.0	61.0	4.0	8.0	5.0	4.0	-7.0	-59.0	16.0	0
1	-47.0	-6.0	-5.0	-7.0	13.0	-1.0	35.0	-10.0	10.0	-4.0	...	-25.0	47.0	6.0	6.0	5.0	13.0	21.0	111.0	15.0	0
2	-19.0	-8.0	-8.0	-8.0	-21.0	-6.0	-79.0	12.0	0.0	5.0	...	-83.0	7.0	7.0	1.0	-8.0	7.0	21.0	114.0	48.0	0
3	2.0	3.0	0.0	2.0	0.0	22.0	106.0	-14.0	-16.0	-2.0	...	-38.0	-11.0	4.0	7.0	11.0	33.0	39.0	119.0	43.0	0
4	6.0	0.0	0.0	-2.0	-14.0	10.0	-51.0	5.0	7.0	0.0	...	38.0	-35.0	-8.0	2.0	6.0	-13.0	-24.0	-112.0	-69.0	0

5 rows × 65 columns

در ادامه می خواهیم توزیع داده ها را مشاهده کنیم :



حال به سراغ preprocessing می رویم :

این کد برای آماده‌سازی داده‌ها برای استفاده در مدل‌های یادگیری ماشین استفاده می‌شود. این شامل تبدیل ماتریس ورودی به بردار ستونی، و استانداردسازی این داده‌ها با میانگین صفر و انحراف معیار یک می‌شود.

```
X_train = X_train.reshape(X_train.shape[0]*X_train.shape[1], 1) #m x n
matrix --> column vector
X_test = X_test.reshape(X_test.shape[0]*X_test.shape[1], 1)
#x.shape[0] = number of rows in x
#x.shape[1] = number of columns in x

sc = StandardScaler() #select the scaler
X_train = sc.fit_transform(X_train) #fit and transform x_train
X_test = sc.transform(X_test) #transform x_test
X_train.shape , X_test.shape , y_train.shape , y_test.shape
```

داده ها را پس از balance کردن suffle می کنیم.

روند پیش‌پردازش داده در این کد به صورت زیر است:

۱ .تغییر ابعاد داده‌ها به بردار ستونی

`X_train` و `X_test` که ابتدا ماتریس‌های دو بعدی (m × n) بودند، با استفاده از `reshape` به بردارهای ستونی تبدیل می‌شوند. این کار برای سازگاری با الگوریتم‌های یادگیری ماشین است.

۲ .استفاده از استانداردسازی

از `StandardScaler` از کتابخانه `scikit-learn` برای استانداردسازی استفاده می‌شود. این استانداردسازی به این صورت انجام می‌شود که میانگین داده‌ها به صفر و انحراف معیار به یک تغییر می‌کنند.

مدل استانداردسازی بر اساس داده‌های آموزش (`X_train`) ساخته می‌شود و همان مقادیر برای داده‌های آموزش و تست استفاده می‌شوند.

به این ترتیب، داده‌ها پس از این پیش‌پردازش، به شکلی مناسب برای ورود به مدل یادگیری ماشین تبدیل می‌شوند. این فرآیند از تغییر ابعاد تا استانداردسازی، اغلب بهبود عملکرد مدل‌های یادگیری ماشین کمک می‌کند و می‌تواند مشکلات مرتبط با مقیاس و توزیع داده‌ها را حل کند.

حال می خواهیم متد های بیان شده را پیاده سازی کنیم :

روش شبکه های عصبی بازگشتی :

این کد یک مدل شبکه عصبی با چند لایه LSTM برای دسته‌بندی با ۴ کلاس ایجاد می‌کند. از Dropout برای جلوگیری از بیش‌برازش استفاده شده و مدل با الگوریتم Adam و تابع هزینه categorical_crossentropy کامپایل شده است.

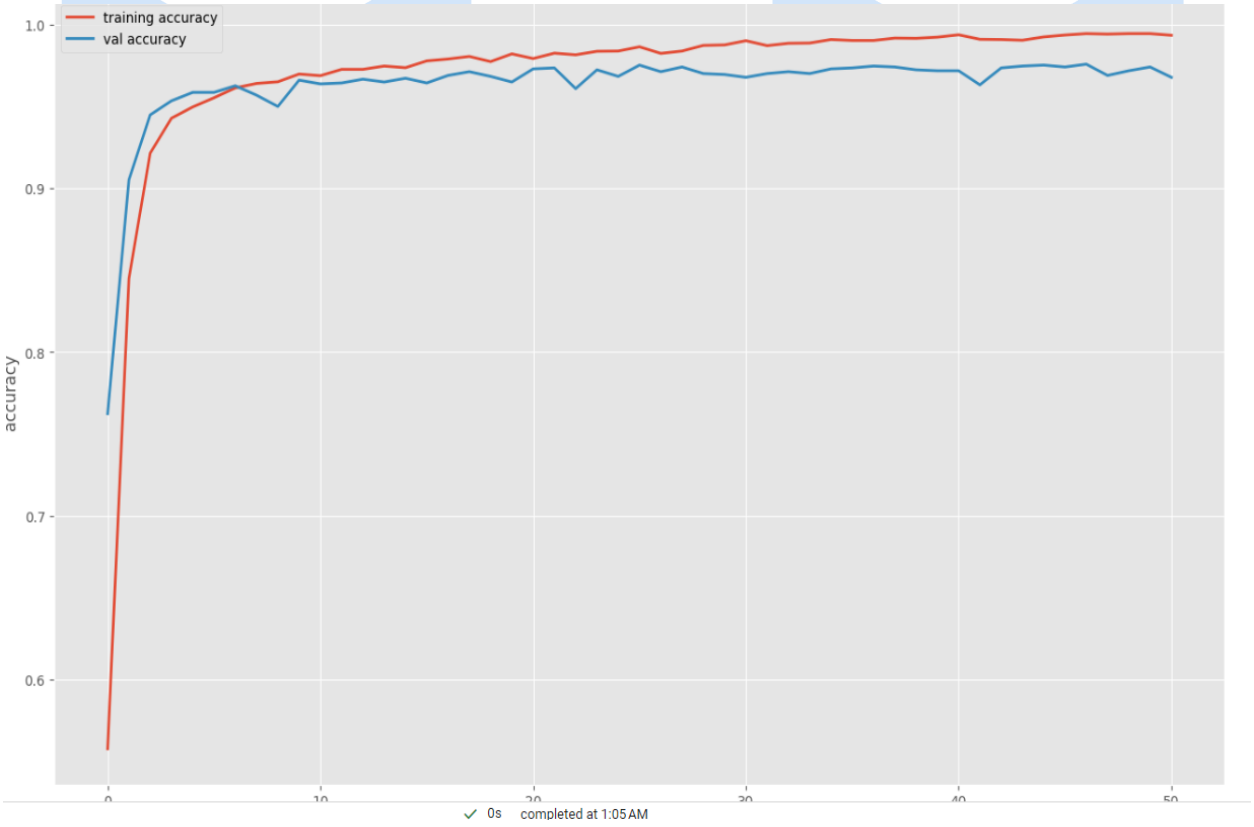
```
model = Sequential()

model.add(LSTM(units=50, return_sequences=True,
input_shape=(X_train.shape[1], 8)))
model.add(Dropout(0.2)) #dropout rate = 20%

model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50, return_sequences = True))
model.add(Dropout(0.2))

model.add(LSTM(units = 50))
model.add(Dropout(0.2))

model.add(Dense(units = 64))
model.add(Dense(units = 128))

model.add(Dense(units = 4, activation="softmax")) #4 as the output classes
model.compile(optimizer = "adam" , loss = "categorical_crossentropy",
metrics=["accuracy"]) #***
```

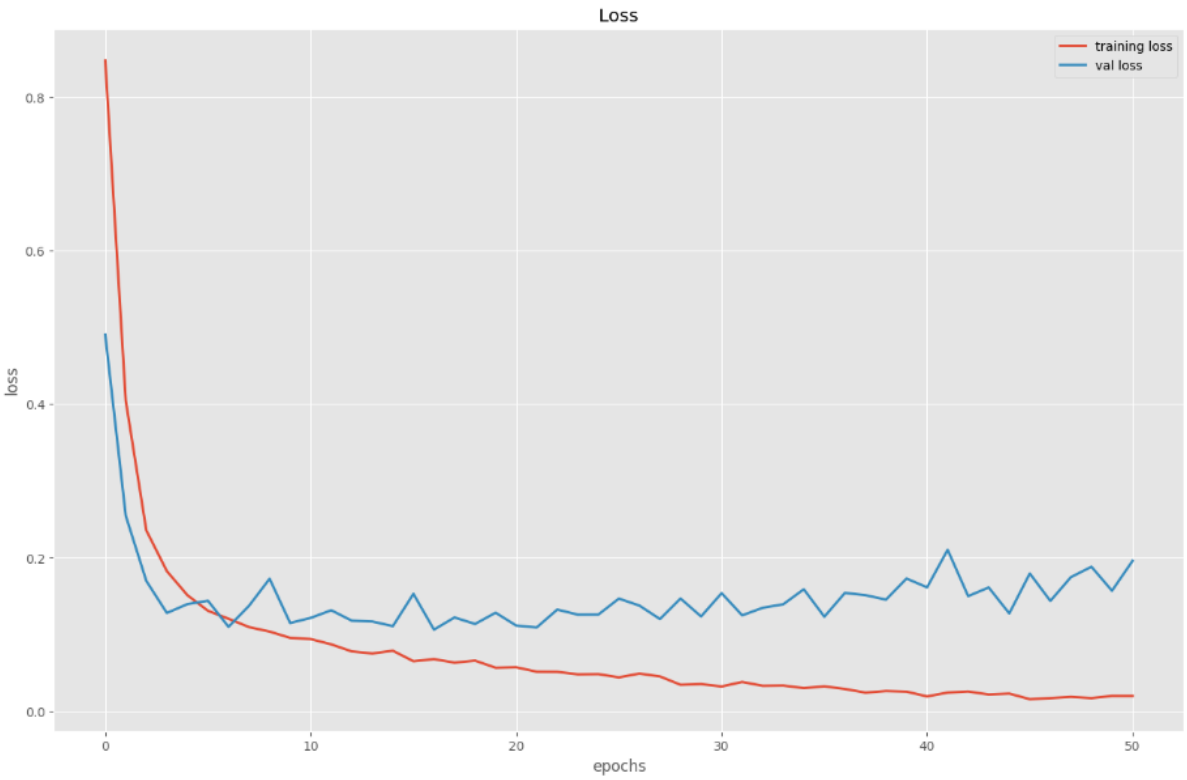
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 8, 50)	11800
dropout (Dropout)	(None, 8, 50)	0
lstm_1 (LSTM)	(None, 8, 50)	20200
dropout_1 (Dropout)	(None, 8, 50)	0
lstm_2 (LSTM)	(None, 8, 50)	20200
dropout_2 (Dropout)	(None, 8, 50)	0
lstm_3 (LSTM)	(None, 50)	20200
dropout_3 (Dropout)	(None, 50)	0
dense_4 (Dense)	(None, 64)	3264
dense_5 (Dense)	(None, 128)	8320
dense_6 (Dense)	(None, 4)	516

این کد، مدل را بر روی داده‌های آموزش و برچسب‌های متناظرش آموزش می‌دهد. تعداد ۲۵۰ دوره آموزش، دسته‌های ۳۲ تایی، با نمایش لاگ و استفاده از ۲۰٪ از داده‌های آموزش به عنوان داده اعتبارسنجی.

```
Epoch 45/250
218/218 - 3s - loss: 0.0225 - accuracy: 0.9925 - val_loss: 0.1271 - val_accuracy: 0.9753 - 3s/epoch - 12ms/step
Epoch 46/250
218/218 - 3s - loss: 0.0153 - accuracy: 0.9937 - val_loss: 0.1792 - val_accuracy: 0.9742 - 3s/epoch - 12ms/step
Epoch 47/250
218/218 - 3s - loss: 0.0166 - accuracy: 0.9945 - val_loss: 0.1434 - val_accuracy: 0.9759 - 3s/epoch - 12ms/step
Epoch 48/250
218/218 - 3s - loss: 0.0183 - accuracy: 0.9943 - val_loss: 0.1744 - val_accuracy: 0.9690 - 3s/epoch - 13ms/step
Epoch 49/250
218/218 - 3s - loss: 0.0167 - accuracy: 0.9945 - val_loss: 0.1879 - val_accuracy: 0.9719 - 3s/epoch - 12ms/step
Epoch 50/250
218/218 - 3s - loss: 0.0195 - accuracy: 0.9945 - val_loss: 0.1566 - val_accuracy: 0.9742 - 3s/epoch - 12ms/step
Epoch 51/250
218/218 - 3s - loss: 0.0195 - accuracy: 0.9935 - val_loss: 0.1956 - val_accuracy: 0.9679 - 3s/epoch - 12ms/step
```

این کد یک نمودار برای دقت مدل در حین آموزش و اعتبارسنجی ایجاد می‌کند و نمودارهای دو خط را نمایش می‌دهد. یک خط برای دقت آموزش و دیگری برای دقت اعتبارسنجی.



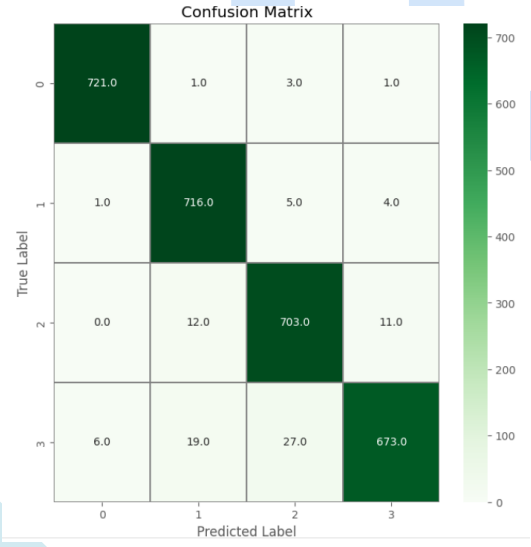


۱ .مدل روی داده‌های تست (`X_test`) پیش‌بینی می‌کند و نتایج را به شکل احتمالات متناظر با هر کلاس (one-hot vectors)در `y_pred_categorical` ذخیره می‌کند.

۲ .این احتمالات را به کلاس‌های پیش‌بینی شده تبدیل کرده و در `y_pred` ذخیره می‌کند.

۳ .از ماتریس اشتباهات (confusion matrix) برای ارزیابی عملکرد مدل استفاده می‌کند.

۴. یک نمودار حرارتی از ماتریس اشتباهات را به همراه دقت پیش‌بینی‌ها رسم می‌کند.



این کد از کتابخانه scikit-learn استفاده می‌کند و گزارش دقت دسته‌بندی را بر اساس ماتریس اشتباهات به صورت خلاصه چاپ می‌کند. این گزارش شامل معیارهایی مانند دقت، بازیابی و اف۱ می‌باشد و برای ارزیابی کلیت عملکرد مدل بر روی داده‌های تست استفاده می‌شود.

نتایج نهایی خروجی :

	precision	recall	f1-score	support
0	0.99	0.99	0.99	726
1	0.96	0.99	0.97	726
2	0.95	0.97	0.96	726
3	0.98	0.93	0.95	725
accuracy			0.97	2903
macro avg	0.97	0.97	0.97	2903
weighted avg	0.97	0.97	0.97	2903

با تغییر پارامتر های مختلف شبکه ی مورد نظر و چندین بار ران کردن به بالا ترین دقت مورد نیاز رسیده ایم. تعداد ایپاک ها هم بعد از یک تعدادی به بعد دارای دقت ثابتی می شوند و نیازی به بالابردن تعداد ایپاک ها نیست و با افزایش آن دقت افزایش یا کاهش نمی یابد.

SVM Classifier🎨

```
from tslearn.svm import TimeSeriesSVC

classifier = TimeSeriesSVC(kernel='linear',random_state = 93)
classifier.fit(X_train_svm,y_train_svm)
y_pred_svm_1 = classifier.predict(X_test_svm)

classifier = TimeSeriesSVC(kernel='rbf',random_state = 93)
classifier.fit(X_train_svm,y_train_svm)
y_pred_svm_2 = classifier.predict(X_test_svm)

classifier = TimeSeriesSVC(kernel='poly',random_state = 93)
classifier.fit(X_train_svm,y_train_svm)
y_pred_svm_3 = classifier.predict(X_test_svm)

classifier = TimeSeriesSVC(kernel='sigmoid',random_state = 93)
classifier.fit(X_train_svm,y_train_svm)
y_pred_svm_4 = classifier.predict(X_test_svm)
```

این کد از کتابخانه tslearnبرای استفاده از مدل‌های دسته‌بندی مبتنی بر سری زمانی با استفاده از SVM استفاده می‌کند. برای چهار نوع مختلف از هسته‌های SVMخطی، RBF، چند جمله‌ای و سیگموئید)، مدل SVM را تعریف و بر روی داده‌های آموزش آموزش داده شده و سپس از مدل برای پیش‌بینی دسته‌ها بر روی داده‌های تست استفاده می‌شود. نتایج پیش‌بینی هر مدل در متغیرهای y_pred_svm_1 تا `y_pred_svm_4` ذخیره می‌شوند.

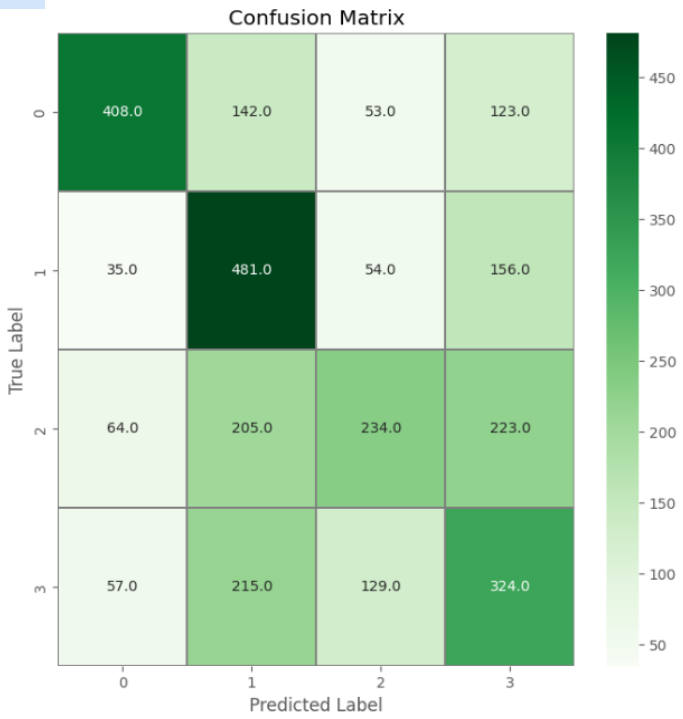
از توابع کرنل متفاوت استفاده کرده ایم که نتایج به صورت زیر خواهد بود :

```
from sklearn import metrics
#model accuracy
print("SVM classifier accuracy:",metrics.accuracy_score(y_test_svm,
y_pred_svm_1))
print("SVM classifier accuracy:",metrics.accuracy_score(y_test_svm,
y_pred_svm_2))
print("SVM classifier accuracy:",metrics.accuracy_score(y_test_svm,
y_pred_svm_3))
print("SVM classifier accuracy:",metrics.accuracy_score(y_test_svm,
y_pred_svm_4))
```

```
SVM classifier accuracy: 0.34516017912504304
SVM classifier accuracy: 0.2497416465725112
SVM classifier accuracy: 0.4984498794350672
SVM classifier accuracy: 0.15707888391319325
```

مشخص است که با کرنل poly بهتر کار کرده است.

این کد یک ماتریس اشتباهات confusion matrix برای مدل SVM با هسته چند جمله‌ای ایجاد می‌کند و سپس این ماتریس اشتباهات را با استفاده از کتابخانه seaborn به همراه یک نمودار حرارتی نمایش می‌دهد. این نمودار حرارتی با استفاده از رنگها، تعداد نمونه‌هایی که به درستی یا اشتباه دسته‌بندی شده‌اند را نشان می‌دهد.



دقت نتایج نهایی به صورت زیر خواهد بود :

```
Classification report - SVM
              precision    recall  f1-score   support

           0       0.72      0.56      0.63       726
           1       0.46      0.66      0.54       726
           2       0.50      0.32      0.39       726
           3       0.39      0.45      0.42       725

    accuracy                           0.50      2903
   macro avg       0.52      0.50      0.50      2903
weighted avg       0.52      0.50      0.50      2903
```

که میبینیم بهترین حالت ممکن در میان تمامی حالت ها ، دارای دقت بسیار پایینی به اندازه ۵۰ درصد می باشد.

Random Forest Classifier

این کد برای استخراج ویژگی از داده‌های سری زمانی استفاده می‌شود. برای هر داده سری زمانی در مجموعه آموزش و تست، میانگین مقادیر در هر کانال در ۸ گام زمانی محاسبه می‌شود. این میانگین‌ها سپس به عنوان ویژگی‌های جدید برای هر داده ذخیره می‌شوند. این فرآیند برای هر نمونه از داده‌های آموزش و تست انجام می‌شود. در نهایت، دو مجموعه جدید از ویژگی‌ها به نام‌های X_train_rf و X_test_rf ساخته می‌شوند که ابعاد مناسبی برای استفاده در مدل دسته‌بندی دارند.

پارامتر ها در بهترین حالت با چندین بار ران گرفتن و در حالت بهینه به صورت زیر خواهد بود :

```python
#import the classifier
from sklearn.ensemble import RandomForestClassifier

#fit of the classifier to the training data
rf_model = RandomForestClassifier(n_estimators=100,random_state=93)
rf_model.fit(X_train_rf, y_train_rf)

#predictions
y_pred_rf = rf_model.predict(X_test_rf)
```

```python
# Model Accuracy
print("RandomForestClassifier accuracy:",metrics.accuracy_score(y_test_rf,
y_pred_rf))
```

```
RandomForestClassifier accuracy: 0.6930761281433
```

این کد یک ماتریس اشتباهات confusion matrix برای مدل دسته‌بندی با استفاده از ویژگی‌های میانگین (استخراج شده از سری زمانی) بر روی داده‌های تست ایجاد می‌کند. سپس این ماتریس اشتباهات را با استفاده از کتابخانه seaborn به همراه یک نمودار حرارتی نمایش می‌دهد. این نمودار حرارتی با استفاده از رنگها، تعداد نمونه‌هایی که به درستی یا اشتباه دسته‌بندی شده‌اند را نمایش می‌دهد.


```
print('Classification report - Random Forest')
print(classification_report(y_test_rf, y_pred_rf))
```

نتایج نهایی خروجی :

```
Classification report - Random Forest
              precision    recall  f1-score   support

           0       0.79      0.74      0.76       726
           1       0.66      0.75      0.70       726
           2       0.73      0.72      0.72       726
           3       0.60      0.56      0.58       725

    accuracy                           0.69      2903
   macro avg       0.69      0.69      0.69      2903
weighted avg       0.69      0.69      0.69      2903
```

SVC Classifier

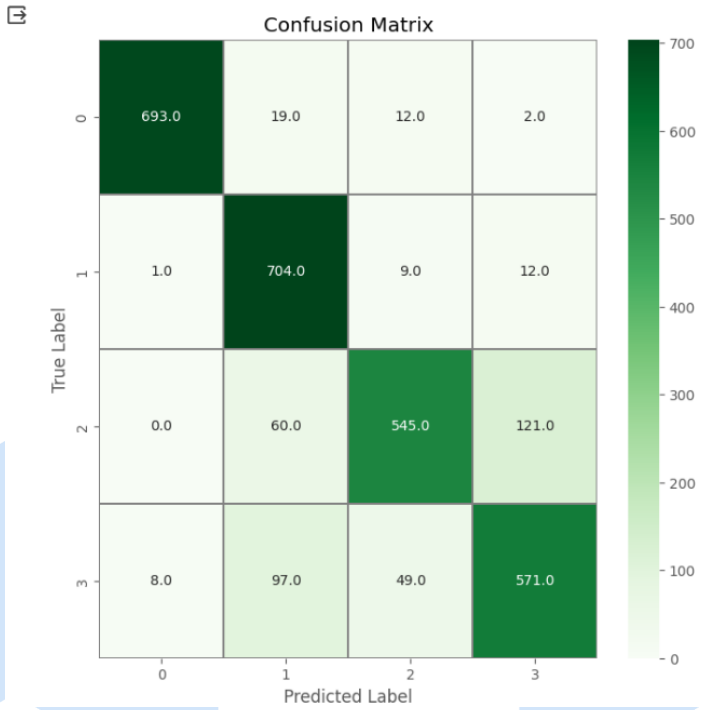
```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score,
f1_score
model = SVC(kernel = 'rbf',decision_function_shape='ovo')
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(classification_report(y_test, predictions))
```

این کد یک مدل دسته‌بندی ماشین بردار پشتیبان SVM با هسته RBF ایجاد می‌کند و آن را بر روی داده‌های آموزش می‌آموزد. سپس از این مدل برای پیش‌بینی دسته‌ها بر روی داده‌های تست استفاده می‌شود. نتایج پیش‌بینی شده سپس با استفاده از توابع ارزیابی مانند classification_report، accuracy_score و f1_score چاپ می‌شوند.

نمایش نتایج :

```
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       726
           1       0.80      0.97      0.88       726
           2       0.89      0.75      0.81       726
           3       0.81      0.79      0.80       725

    accuracy                           0.87      2903
   macro avg       0.87      0.87      0.86      2903
weighted avg       0.87      0.87      0.86      2903
```

این کد یک ماتریس اشتباهات confusion matrix را برای مدل دسته‌بندی با استفاده از ماشین بردار پشتیبان SVMبا هسته RBF بر روی داده‌های تست ایجاد می‌کند. سپس این ماتریس اشتباهات را با استفاده از کتابخانه seabornبه همراه یک نمودار حرارتی نمایش می‌دهد. این نمودار حرارتی با استفاده از رنگها، تعداد نمونه‌هایی که به درستی یا اشتباه دسته‌بندی شده‌اند را نمایش می‌دهد.



حال می خواهیم روش های مختلفی که تا به اینجا پیاده سازی کرده ایم را با یکدیگر مقایسه کنیم :

```
print('Classification report - Recurrent NN')
print(classification_report(y_test, y_pred))
print('-----------------------------------------------------------')
print('Classification report - SVM')
print(classification_report(y_test_svm, y_pred_svm_3))
print('-----------------------------------------------------------')
print('Classification report - Random Forest')
print(classification_report(y_test_rf, y_pred_rf))
print('-----------------------------------------------------------')
print('Classification report - SVC')
print(classification_report(y_test, predictions))
```

```
Classification report - Recurrent NN
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       726
           1       0.96      0.99      0.97       726
           2       0.95      0.97      0.96       726
           3       0.98      0.93      0.95       725

    accuracy                           0.97      2903
   macro avg       0.97      0.97      0.97      2903
weighted avg       0.97      0.97      0.97      2903
```

```
-----------------------------------------------------------
Classification report - SVM
              precision    recall  f1-score   support

           0       0.72      0.56      0.63       726
           1       0.46      0.66      0.54       726
           2       0.50      0.32      0.39       726
           3       0.39      0.45      0.42       725

    accuracy                           0.50      2903
   macro avg       0.52      0.50      0.50      2903
weighted avg       0.52      0.50      0.50      2903
```

```
------------------------------------------------------------
Classification report - Random Forest
              precision    recall  f1-score   support

           0       0.79      0.74      0.76       726
           1       0.66      0.75      0.70       726
           2       0.73      0.72      0.72       726
           3       0.60      0.56      0.58       725

    accuracy                           0.69      2903
   macro avg       0.69      0.69      0.69      2903
weighted avg       0.69      0.69      0.69      2903
```

```
Classification report - SVC
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       726
           1       0.80      0.97      0.88       726
           2       0.89      0.75      0.81       726
           3       0.81      0.79      0.80       725

    accuracy                           0.87      2903
   macro avg       0.87      0.87      0.86      2903
weighted avg       0.87      0.87      0.86      2903
```

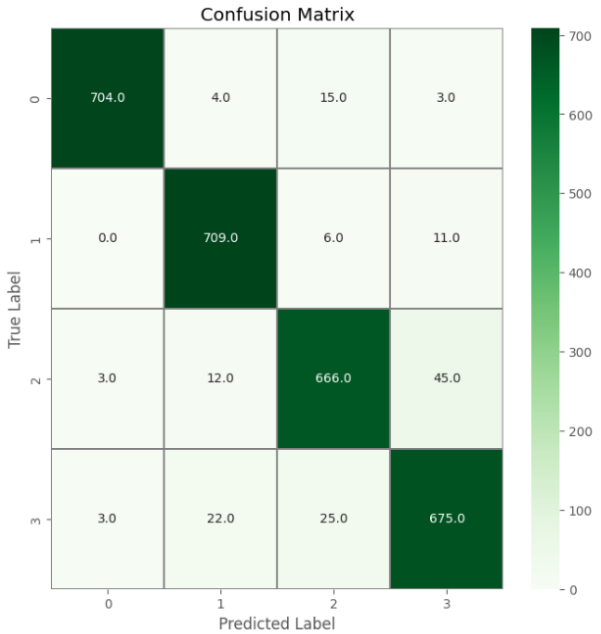
MLP

پارامتر های آن به صورت زیر می باشد : (پس از بهینه سازی پارامتر ها)

```
model = MLPClassifier(hidden_layer_sizes=(80,), activation='relu',
solver='adam',
                      alpha=0.1,max_iter=100, shuffle=True,
random_state=93)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

```
0.9486737857388908
```

این کد پیش‌بینی دسته‌ها بر روی داده‌های تست با استفاده از مدل SVM انجام می‌دهد. سپس ماتریس اشتباهات confusion matrixبرای ارزیابی دقت پیش‌بینی‌ها ایجاد می‌شود و با استفاده از کتابخانه seaborn به همراه یک نمودار حرارتی نمایش داده می‌شود. این نمودار حرارتی با استفاده از رنگها، تعداد نمونه‌هایی که به درستی یا اشتباه دسته‌بندی شده‌اند را نشان می‌دهد.



نتایج خروجی :

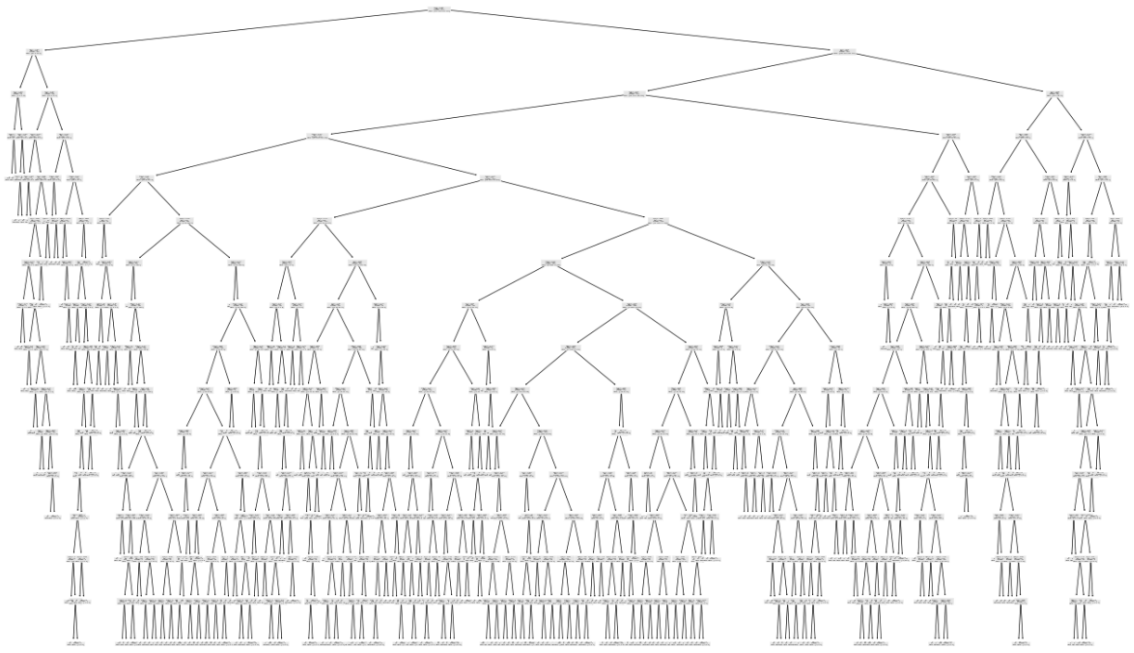
	precision	recall	f1-score	support
0	0.99	0.97	0.98	726
1	0.95	0.98	0.96	726
2	0.94	0.92	0.93	726
3	0.92	0.93	0.93	725
accuracy			0.95	2903
macro avg	0.95	0.95	0.95	2903
weighted avg	0.95	0.95	0.95	2903

Decision Tree (sklearn)

```
clf = tree.DecisionTreeClassifier(max_depth=18, random_state=42,
ccp_alpha=0.001)
clf.fit(X_train, y_train)
```

پارامتر های درخت تصمیم پس از چندین بار ران کردن به صورت بالا نتیجه گیری می شوند که دارای دقت نسبتا خوبی می باشد :

```
DecisionTreeClassifier
DecisionTreeClassifier(ccp_alpha=0.001, max_depth=18, random_state=42)
```



```
clf.predict(X_test)
clf.score(X_test, y_test)
clf.predict_proba(X_test)
```

```
array([[0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 1.        , 0.        ],
       [0.        , 0.        , 0.        , 1.        ],
       ...,
       [0.03      , 0.04      , 0.        , 0.93      ],
       [0.        , 0.89772727, 0.03977273, 0.0625    ],
       [0.        , 0.        , 0.        , 1.        ]])
```

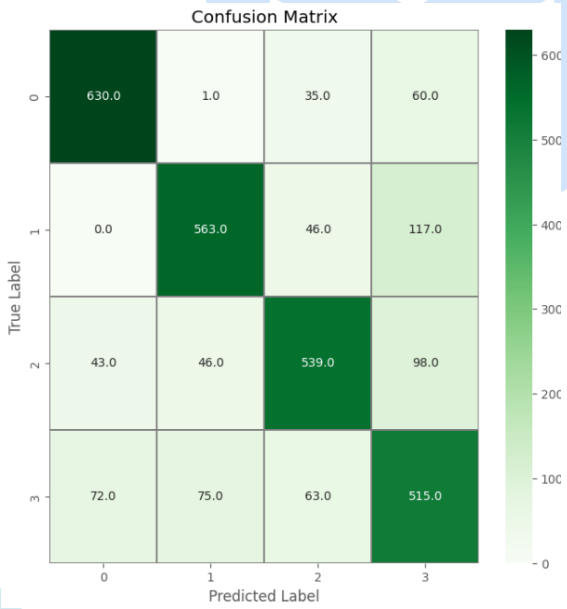
```
y_pred = clf.predict(X_test)
```

این کد، مدل یادگیری ماشینی که با نام clf تعریف شده است، را بر روی داده‌های تست X_test اجرا می‌کند و نتایج پیش‌بینی را در متغیر y_pred ذخیره می‌کند. این خط کد به وسیله مدل آموزش دیده شده، برای پیش‌بینی خروجی مربوط به ویژگی‌های تست از داده‌های X_test استفاده می‌کند.

این کد یک ماتریس اشتباهات (confusion matrix) را برای مدل دیکشنری درخت تصمیم ایجاد می‌کند و سپس این ماتریس اشتباهات را به همراه یک نمودار حرارتی نمایش می‌دهد. در این نمودار حرارتی، رنگها نشان‌دهنده تعداد نمونه‌هایی هستند که به درستی یا اشتباه به هر یک از دسته‌ها دسته‌بندی شده‌اند.

سپس با استفاده از تابع classification_report از کتابخانه scikit-learn، یک گزارش دقیق از معیارهای دقت، بازیابی و اف۱ برای ارزیابی عملکرد مدل روی داده‌های تست چاپ می‌شود.

نتایج ماتریس در هم ریختگی و دقت خروجی :




```
Classification report - Decision tree
              precision    recall  f1-score   support

           0       0.85      0.87      0.86       726
           1       0.82      0.78      0.80       726
           2       0.79      0.74      0.77       726
           3       0.65      0.71      0.68       725

    accuracy                           0.77      2903
   macro avg       0.78      0.77      0.77      2903
weighted avg       0.78      0.77      0.77      2903
```

```
from sklearn.neighbors import (NeighborhoodComponentsAnalysis,
KNeighborsClassifier)
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline

nca = NeighborhoodComponentsAnalysis(random_state=42)
knn = KNeighborsClassifier(n_neighbors=4)
nca_pipe = Pipeline([('nca', nca), ('knn', knn)])
nca_pipe.fit(X_train, y_train)
print(nca_pipe.score(X_test, y_test))
```

این کد یک مدل دسته‌بندی K-Nearest Neighbors (KNN) با استفاده از تحلیل مؤلفه‌های محلی Neighborhood Components Analysis - NCA ایجاد کرده و دقت این مدل را روی داده‌های تست اندازه‌گیری و چاپ می‌کند.

دقت :

```
0.826730967964175
```

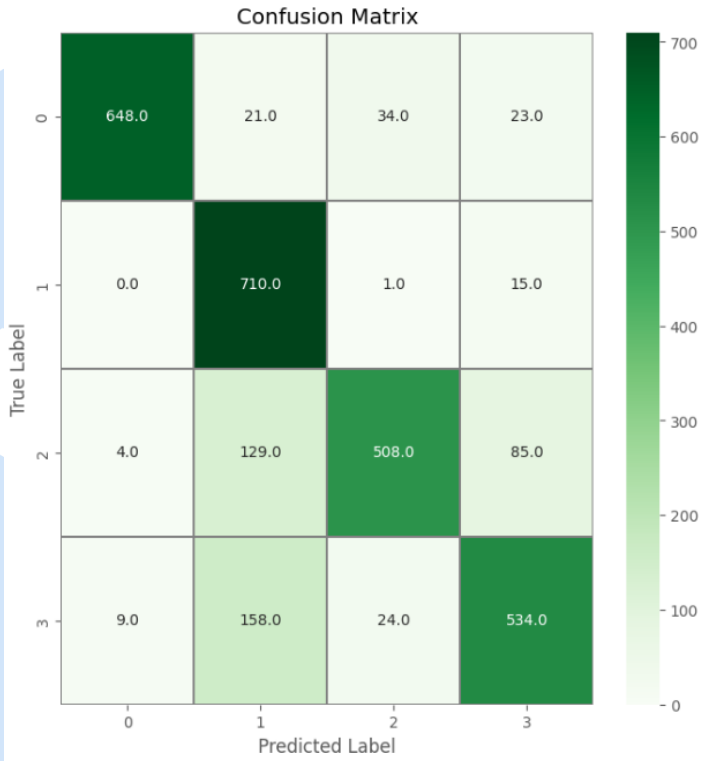
```
y_pred = nca_pipe.predict(X_test)
confusion_mtx = confusion_matrix(y_test, y_pred)

f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True,
linewidths=0.01,cmap="Greens",linecolor="gray", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

print('Classification report - Decision tree')
print(classification_report(y_test, y_pred))
```

این کد یک ماتریس اشتباهات (confusion matrix) را برای مدل دسته‌بندی ارائه شده (احتمالاً درخت تصمیم) بر روی داده‌های تست ایجاد کرده و سپس این ماتریس اشتباهات را به همراه یک نمودار حرارتی نمایش

می‌دهد. این نمودار حرارتی با استفاده از رنگها، تعداد نمونه‌هایی که به درستی یا اشتباه دسته‌بندی شده‌اند را نشان می‌دهد.

سپس با استفاده از تابع classification_report از کتابخانه scikit-learn، یک گزارش دقیق از معیارهای دقت، بازیابی و اف۱ برای ارزیابی عملکرد مدل روی داده‌های تست چاپ می‌شود.

نتایج ماتریس در هم ریختگی و دقت خروجی :



Classification report - Decision tree

	precision	recall	f1-score	support
0	0.98	0.89	0.93	726
1	0.70	0.98	0.81	726
2	0.90	0.70	0.79	726
3	0.81	0.74	0.77	725
accuracy			0.83	2903
macro avg	0.85	0.83	0.83	2903
weighted avg	0.85	0.83	0.83	2903

توضیحات بیشتر روش NN :

به طور کلی، شبکه‌های عصبی (NN) یک روش هوش مصنوعی هستند که از ساختاری از یک یا چند لایه از واحدهای نورونی (عصب‌ها) و اتصالات بین آن‌ها بهره می‌برند.

در شبکه‌های عصبی، هر نورون یک واحد پردازشگر است که ورودی‌های خود را با وزن‌های مخصوص آن‌ها ضرب می‌کند، این محصولات را جمع می‌کند، و سپس این جمع‌ها را به تابع فعال‌سازی (activation function) منتقل می‌کند. این فرایند برای هر نورون تکرار می‌شود تا به لایه‌های عمیق‌تر شبکه برسیم.

شبکه‌های عصبی انواع مختلفی دارند که شامل شبکه‌های عصبی پرسپترون(Perceptrons) ، شبکه‌های عصبی چندلایه(Multi-layer Neural Networks) ، شبکه‌های عصبی بازگشتی (Recurrent Neural Networks)، و شبکه‌های عصبی کانولوشنی (Convolutional Neural Networks) می‌شوند.

در طی سال‌های اخیر، شبکه‌های عصبی عمیق Deep Neural Networks یا DNNs به دلیل توانمندی‌های بالایشان در تسک‌های یادگیری ماشین و هوش مصنوعی، بسیار محبوب شده‌اند. این شبکه‌ها از بسیاری از لایه‌های نورونی تشکیل شده‌اند و به کمک الگوریتم‌های یادگیری عمیق Deep Learning آموزش داده می‌شوند.

مراجع

https://github.com/cyber-punk-me

https://www.kaggle.com/datasets/kyr7plus/emg-4/download?datasetVersionNumber=2

K. N. Toosi University of Technology

Faculty of Electrical Engineering

[Final Project_AI]

Classify gestures by reading muscle activity

By:

[Iman Fekri

Mohammad Hossein Bayati]

Student number:

[9929083

9924693]

Professor:

[Dr.Aliyari]

Winter 2024