دانشگاه تهران
پردیس فارابی
دانشکده‌ی مهندسی
گروه مهندسی کامپیوتر

توسعه برنامه ی موبایل برای یک سیستم بازشناسی فعالیت ها بر اساس حسگر ژیروسکوپ موبایل

نگارش:

امیرحسین اسدی

استاد راهنما:

دکتر کاظم فولادی

گزارش پروژه برای دریافت درجه‌ی کارشناسی
در رشته‌ی مهندسی کامپیوتر

شهریور ۱۳۹۹


بسم الله الرحمن الرحيم

چکیده

تلفن های همراه ابزارهای فراگیر و متوسطی هستند که دارای قدرت کنترل موثر و توانمندی هستند که می تواند محاسبات کارآمد و قدرتمندی را انجام دهد. یکی از ویژگی هایی که در آن تعبیه شده است سنسورها هستند. تلفن های همراه دارای چندین سنسور می باشد به عنوان مثال ، سنسورهای مجاورت ، سنسورهای دما ، شتاب سنج ها ، ژیروسکوپ ها و موارد دیگر. این سنسورها برای زمینه های مختلف مانند داده کاوی و تجزیه و تحلیل داده ها است. وجود این سنسورها مردم را قادر می سازد برای انجام کارهای مختلف اطلاعات آن را کنترل کنند. یکی از این کارها ، تشخیص حرکت است که به آن فعالیت گفته می شود

در این پروژه یک مجموعه داده توسط ۴ انسان جمع آوری شد که شامل ۴ حرکات مچ دست به چپ ، راست ، پایین و بالا می باشد. ۱۰ ویژگی در این پروژه از داده های خام جمع آوری شده است.

ویژگی ها با استفاده از SVM طبقه بندی شده اند. نتیجه کار نیز دارای بازدهی ۹۶.۵ درصدی بود. در آخر نیز حرکات جدید به صورت انیمیشن در برنامه دسکتاپ نمایش داده شد.

کلمات کلیدی :

- یادگیری ماشین
- استخراج ویژگی
- تحلیل ویژگی
- برنامه موبایل
- برنامه دسکتاپ
- برنامه نویسی
- ژیروسکوپ
- شتاب سنج
- Svm

فهرست

فصل اول

مقدمه

تلفن های همراه فراگیر شده اند و هر روز پیشرفت می کنند. اجزای سازنده تلفن هوشمند حسگرها هستند. سنسورهای GPS ، شتاب سنج ، ژیروسکوپ ، سنسورهای مجاورت ، حسگرهای نور و حسگرهای اثر انگشت تعداد کمی از حسگرهایی هستند که در بسیاری از موبایل های مدرن تعبیه شده اند. وجود این سنسورهای قدرتمند در تلفن های هوشمند، ما را قادر به استفاده از آن ها کرده است. یکی از این کارها شناخت فعالیت با حرکت گوشی موبایل و درک داده هایی است که سنسورها تولید می کنند.

شناخت فعالیت های انسانی یک موضوع تحقیقاتی بسیار پویا و چالش برانگیز است. هدف آن تعیین فعالیتهای یک شخص یا گروهی از افراد بر اساس داده های سنسور می باشد.

به طور کلی ، این فرآیند شامل چندین مرحله است. از جمع آوری اطلاعات مربوط به رفتار انسان از داده های خام گرفته است و تا نتیجه گیری نهایی در مورد فعالیت انجام شده ادامه می یابد. این مراحل به شرح زیر است:

۱. پیش پردازش داده های خام - برای مدیریت ناقص بودن ، از بین بردن افزونگی و انجام تجمع داده ها و نرمال سازی.
۲. تقسیم بندی - شناسایی مهمترین بخشهای داده
۳. استخراج ویژگی - استخراج ویژگی های اصلی (به عنوان مثال اطلاعات زمانی و مکانی) از داده های تقسیم بندی شده.
۴. کاهش ابعاد - کاهش تعداد ویژگی ها برای افزایش کیفیت آنها و کاهش بار محاسباتی مورد نیاز برای طبقه بندی
۵. طبقه بندی ، یادگیری اصلی ماشین
۶. استدلال - تعیین فعالیت داده شده.

اهداف اصلی این سیستم ها مشاهده و تجزیه و تحلیل فعالیت های انسانی و تفسیر موفقیت آمیز وقایع در حال انجام است. با استفاده از داده های بصری و غیر بصری سنسورها ، سیستم ها برای درک رفتار انسان ، داده های متنی را بازیابی و پردازش می کنند. چندین حوزه کاربردی وجود دارد که ما آنها را تقریباً به چهار دسته تقسیم می کنیم: سیستم های فعال و کمک به زندگی برای خانه های هوشمند ، برنامه های نظارت بر مراقبت های بهداشتی ، سیستم های نظارت برای فعالیت های داخلی و خارجی و برنامه های TI .

تلفن های همراه از آنجایی که مقرون به صرفه هستند و قدرت محاسباتی بالایی دارند مورد استفاده قرار گرفته اند. برای طبقه بندی فعالیت هایی که انسان انجام می دهد، از دو سنسور ژیروسکوپ و شتاب سنج که در بیشتر گوشی های هوشمند امروزی وجود دارد استفاده شده است.

یک شـتاب سـنج سـه محور مقدار جابجایی جسـم را در امتداد محور X، Y و Z برمی گرداند. ژیروسـکوپ دسـتگاهی اسـت که مقدار چرخش بدن را در امتداد محور X یعنی حرکت از یک طرف به طرف دیگر ، محور Y یعنی چرخش به عقب و جلو و محور Z یعنی چرخش از حالت عمودی به افقی و بالعکس را بر می گرداند.

یکی از چالش ها، انتخاب از بین سـنسـور های موجود بود که با توجه با پروژه ما، تصـمیم بر آن شـد که دو سـنسـور ژیروسـکوپ و شـتاب سـنج مورد اسـتفاده قرار گیرد. چالش بعدی یافتن ویژگی های مناسـب برای حرکت های موجود در پروژه بود که پس از بررسی به ده ویژگی مطلوب رسیدیم.

روش کار نیز به این صورت بود که ابتدا دیتاسـتی برای پروژه جمع آوری شد. سـپس ویژگی های یافت شـده را بر روی دیتاسـت های جمع شـده اعمال کردیم. حال نوبت به کلاس بندی رسـید که پس از بررسـی و همچنین آزمون و خطا به الگوریتم svm رسـیدیم که با دقت ۹۶.۵ همراه بود. در انتها نیز برنامه دسـکتاپی طراحی شد تا یک حرکت از کاربر گرفته و حرکت مد نظر را به صورت انیمیشن به کاربر نمایش دهد.

در ادامه به مرور کارهای مشـابه، سـاختار سـیسـتم بازشـناسـی فعالیت ها، پیاده سـازی برنامه موبایل، بررسـی نتایج و تحلیل آن ها و همچنین نتیجه گیری و کار های آینده بحث خواهد شد.

فصل دوم

مروری بر کارهای مشابه

طی دهه گذشته ، با کشف های جدید در جهان در زمینه هوش مصنوعی و یادگیری ماشین ، شناخت فعالیت مورد توجه زیادی قرار گرفته است. در طول این ۱۰ سال گذشته ، بهبودهای مداومی در این حوزه ایجاد شده است ، مطالعات اولیه از لکه بینی حرکات با سنسورهای اینرسی فرسوده بدن و راه حل های پیشرفته ارائه فعالیت پیچیده فعالیت انسان با استفاده از تلفن های هوشمند و حسگرهای حرکتی مچ دست می باشد. در جدول زیر مقالاتی که در این زمینه فعالیت داشته اند را نشان می دهیم.

ردیف	دسته بندی	شماره مرجع	محتویات پژوهش
۱	نظارت و امنیت	1	• فعالیت های مشکوک یا خشونت آمیز را از ویدئو تشخیص می دهد. • در صورت تشخیص فعالیت های مشکوک، به فرد میزبان اخطار می دهد.
۲	نظارت و امنیت	2	• ناهنجاری ها را با کمک یادگیری عمیق در فیلم دوربین های مدار بسته تشخیص می دهد. • مجموعه ای از دیتاست های واقغی را از دوربین های مداربسته مختلف جمع آوری و تایید می کند.
۳	سلامت	3	• از سنسورهای پوشیدنی برای تشخیص فعالیت های پیچیده انسان استفاده می کند. • حسگرهایی را درون محیط اتاق قرار می دهد تا فعالیت را دقیق تر تحلیل کند.
۴	سلامت	4	• از مهندسی ویژگی خودکار برای بهبود تشخیص فعالیت در برنامه AAL استفاده می کند. • فعالیت ها را می تواند تنها با تلفن همراه هوشمند یا ساعت هوشمند و با دقت بالا تشخیص دهد.
۵	رانندگی خودمختار	5	• حرکات راننده را با بررسی وضعیت سر و چشم فرد، تشخیص میدهد. • مجموعه ای جدید از ویژگیهای مبتنی بر سر و چشم را ارائه می دهد.
۶	تعامل انسان – ربات	6	• یک سیستم رباتیک نو برای پردازش سیگنال دارای چند سنسور تعریف می کند. • نتایج امیدوارکننده را در تشخیص حرکات انسان نشان می دهد.

۷	تعامل انسان – ربات	7	• مدل و الگوریتمی جدید را پیشنهاد می کند که می تواند به طور خودکار فیلم های ضبط شده توسط یک ربات را پردازش کند، در حالی که با مردم تعامل می کند. • در حین تعامل با انسان، مجموعه ای از احساسات انسان را تشخیص می دهد.

در این بخش ، ما برخی دیگر از روش های تشخیص فعالیت کاربر را تجزیه و تحلیل می کنیم:

- یکی از تلاشهای اولیه برای شناسایی فعالیتهای انسانی با استفاده از سنسورها ، استفاده از حسگرهای فرسوده بدن مانند شتاب سنج ها و ژیروسکوپ ها ،به منظور شناسایی طیف وسیعی از فعالیت های به طور مثال دست دادن ، بلند کردن تلفن ، گذاشتن تلفن، باز کردن در ، نوشیدن ، استفاده از قاشق بود. بدین منظور ، مجموعه ای از حسگرها بر روی بازوی داوطلب ها قرار گرفتند ، که یک سر آن بر روی مچ دست و یک سر دیگر آن در بالای بازو ، نزدیک شانه قرار داشت. داده های جمع آوری شده شامل اطلاعات جهت گیری نسبی ، مانند زاویه نگه داشتن دست و حرکت دست بود. تفسیر نتایج با استفاده از دو معیار انجام شد. مورد اول یادآوری با بهره وری ۰.۸۷ بود و مورد دوم درستی با بهره وری ۰/۶۲ بود.
- یک راه حل دیگر استفاده از شتاب سنج های تلفن همراه برای شناسایی فعالیت های انسانی بود. آن ها ترجیح دادند فقط از سنسور شتاب سنج استفاده کنند زیرا در آن زمان ، این تنها سنسور حرکتی قابل توجه بود که بیشتر دستگاه های تلفن همراه از آن استفاده می کردند. در این فاصله ، چندین حسگر دیگر به تلفن های هوشمند رسیدند. فعالیت های تحت نظر شامل راه رفتن ، دویدن آهسته ، نشستن ، ایستادن ، بالا رفتن از پله ها و پایین آمدن از پله ها بود. این مطالعه به سه بخش اصلی تقسیم شد:
  ۱. جمع آوری داده ها
  ۲. تولید ویژگی ها
  ۳. کارهای تجربی.

  اولین مرحله ، جمع آوری داده ها ، با کمک بیست و نه داوطلب انجام شد که هنگام انجام فعالیت ها یک دستگاه تلفن همراه را در جیب شلوار خود حمل می کردند. پس از آزمایش احتمال وجود استین های ۲۰ ثانیه ای ، سرانجام هر فعالیت در استین های ۱۰ ثانیه ای انجام شد ، و هر ۵۰ میلی ثانیه داده ها را از حسگرها می خواند. تعداد مجموعه های جمع آوری شده برای هر شخص همچنین برای هر فعالیت انجام شده متفاوت است . از آنجا که این راه حل مستلزم استفاده از الگوریتم طبقه بندی است ، در مرحله بعد، هدف اصلی شکل دادن به داده ها بود تا بتوان آنها را به عنوان ورودی به الگوریتم های موجود منتقل کرد.

  در این مرحله شش نوع ویژگی ایجاد شد:
  ۱. میانگین

۲. انحراف معیار
۳. میانگین تفاضل قدر مطلق‌ها
۴. شتاب متوسط نتیجه
۵. زمان بین قله ها
۶. توزیع باند

مرحله آخر بخش عملی آزمایش با ویژگی های استخراج شده و با سه روش طبقه بندی ، یعنی درخت تصمیم ، رگرسیون لجستیک و شبکه های عصبی چند لایه است. نتایج نشان داد که درصد خوبی برای پیاده روی (۹۲٪) ، آهسته دویدن (۹۸٪) ، نشستن (۹۴٪) ، و ایستادن (۹۱٪) تشخیص داده شد. اینها مقادیر متوسط سه روش طبقه بندی هستند. در حالی که بالا رفتن از پله و پایین آمدن به طور متوسط ۴۹٪ و ۳۷٪ بود. اگر رگرسیون لجستیک در نظر گرفته نشود ، به ۶۰٪ و ۵۰٪ افزایش می یابد. این منجر به این شد که در نهایت این دو فعالیت به کلی حذف شوند. در عوض ، یک فعالیت جدید برای بالا رفتن از پله ها با دقت متوسط ۷۷٪ اضافه شد ، که هنوز هم به طور قابل توجهی کمتر از چهار فعالیت دیگر به دست آمده بود ، اما قابل اطمینان تر است.

- یکی از جدیدترین و پیچیده ترین مطالعات ، راه حل مبتنی بر تلفن های هوشمند و حسگرهای حرکتی مچ دست را پیشنهاد می کند.. ایده اصلی این روش این است که نحوه نگهداری تلفن های هوشمند توسط کاربرانشان (به عنوان مثال در جیب شلوار) برای تشخیص فعالیت های انسانی که شامل حرکات دست است ، مناسب نیست. به همین دلیل از سنسورهای اضافی به غیر از سنسورهای دستگاه استفاده می شود. هر دو مجموعه سنسور شامل شتاب سنج ، ژیروسکوپ و شتاب خطی بودند.

داده های استفاده شده برای سیزده فعالیت از ده شرکت کننده جمع آوری شد ، اما فقط هفت فعالیت توسط همه شرکت کنندگان انجام شد که دقیقاً همان فعالیت هایی است که راه حل پیشنهادی با هدف شناسایی آنها انجام می شود. هر فعالیت به مدت ۳ دقیقه انجام شد ، مقدار کل ۳۰ دقیقه برای هر فعالیت بود و در نتیجه یک مجموعه داده ۳۹۰ دقیقه ای ایجاد شد.. فقط دو ویژگی استخراج شد ، میانگین و انحراف معیار. این ویژگی ها برای پیچیدگی کم و دقت معقول برای فعالیت های مختلف انتخاب شده اند. نتایج از ترکیب سنسورها تشکیل شده است. هر سنسور به تنهایی و سپس در ترکیب با حسگرهای دیگر و همچنین ترکیب دو موقعیت برای دستگاه های تلفن همراه ارزیابی شد. هنگام استفاده از شتاب سنج و ژیروسکوپ در حالت مچ دست ، در تشخیص فعالیت ها چند خطا رخ داده است ، بزرگترین سردرگمی بین راه رفتن و بالا رفتن از پله ، با دقت ۵۳٪ ، بود. نتایج هنگام ترکیب دو موقعیت آزمون بهبود یافت ، دقت کلی به ۹۸ درصد افزایش یافت. اشکال اصلی این راه حل ، وقتی با راه حل پیشنهادی مقایسه می شود ، استفاده از دو دستگاه تلفن همراه بود که در زندگی واقعی غیرممکن است.

- یکی از زمینه هایی که اخیراً بسیار مورد توجه قرار گرفته است فعالیت های ورزشی به ویژه تناسب اندام و دویدن است. نمونه های بی شماری از برنامه ها که از شناسایی فعالیت های انسانی برای کمک به کاربران در پیگیری جلسات آموزشی خود وجود دارند مانند:
  ۱. Samsung health
  ۲. Nike+
  ۳. Endomondo
  ۴. Strava

اگر چه این برنامه ها تعداد بسیار محدودی از فعالیت ها را تشخیص می دهند ، اما نتایج بسیار خوبی دارند ، در تشخیص نوع فعالیت انجام شده بسیار دقیق هستند و به مشتریان خود مزایایی مانند مکث خودکار هنگام تشخیص اینکه کاربر دیگر درحال دویدن نیست و همچنین شخصی سازی برنامه تمرینی.

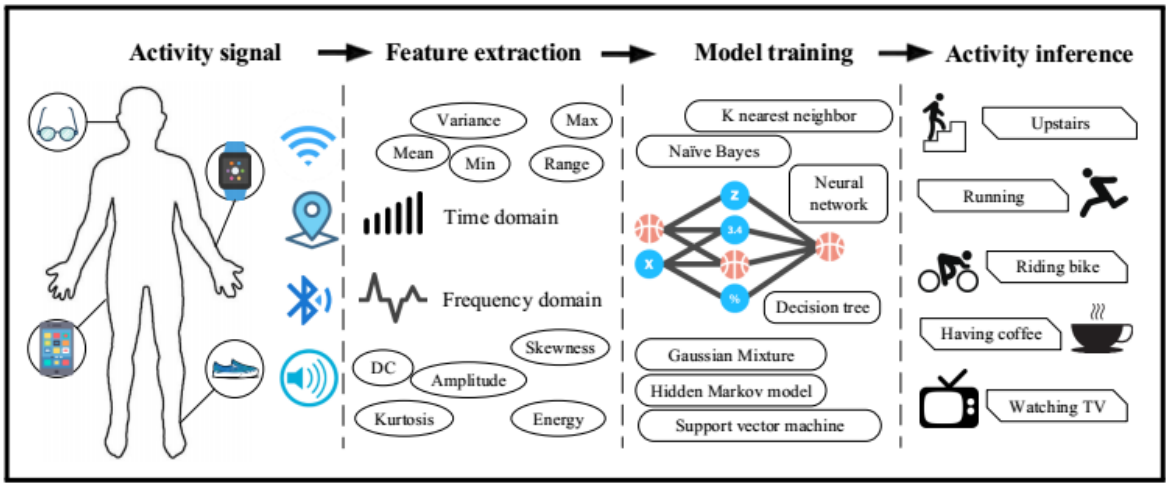
بعلاوه ، در سالهای اخیر ، در ساعت های هوشمند و بندهای تناسب اندام رشد چشمگیری مشاهده شده است. مانند fitbit و apple watch که می توانند تعداد قدم ها ، الگوی خواب ، دوره های بی تحرکی و غیره را ردیابی کنند. این نوع دستگاه ها به عنوان اجزای سیستم های پیچیده تری استفاده می شوند که حسگرهای زیادی را برای انجام فعالیت های عمیق برای سناریوهایی مانند مراقبت های بهداشتی ، پیری سالم ، فناوری اقناعی برای رفتار سالم و غیره را به کار می گیرند.

- لذت مردم از بازی هرگز از بین نمی رود و در زمینه بازی پیشرفت های زیادی حاصل شده است. در چند سال گذشته ، با ایجاد فناوری هایی مانند Kinect، PlayStation Move و Nintendo Wii ، شناسایی فعالیت به بخشی فعال در بازی تبدیل شده است. اگرچه برخی از این فعالیت ها را فقط با استفاده از محاسبات بصری تشخیص می دهند (مانند Kinect) ، سایر موارد به حسگرها متکی هستند. Nintendo Wii دارای یک ریموت است که دارای یک سنسور حرکتی است که شناسایی فعالیت را امکان پذیر می کند. همه اینها به روشهای مختلفی برای انجام بازی ها و حتی ایجاد عادت سالم مورد استفاده قرار می گیرند. با شناخت فعالیتهایی که کاربر در حال انجام آن است ، رایانه می تواند کاربر را درک کرده و بر اساس واکنش های انسانی پاسخی ارائه دهد. از این طریق تعامل انسان و کامپیوتر امکان پذیر است.
- مطالعات اخیر نشان داده است که می توان از فعالیت و رفتار انسان برای نشان دادن وضعیت سلامتی انسان استفاده کرد. تحقیقات انجام شده ، متخصصان پزشکی را به این نتیجه رسانده است که بین میزان فعالیت بدنی و بیماریهای مختلف مربوط به چاقی و متابولیسم ارتباط زیادی وجود دارد. با توجه به مقدار زیادی از داده های قابل جمع آوری در مورد فعالیت فرد ، ایده استفاده از این داده ها برای جمع آوری اطلاعات در مورد وضعیت پزشکی انسان به سرعت رشد کرده است. اگرچه اعتقاد بر این است که این ممکن است راه حل بهتری نسبت به یک قرار ملاقات پزشکی با محدودیت زمانی باشد ، اما به عنوان یک جایگزین در نظر گرفته نمی شود ، بلکه بیشتر شبیه یک ابزار اضافی است.

فصل سوم

ساختار سیستم بازشناسی فعالیت‌ها

در این فصل به نحوه کار و توضیح الگوریتم ها می پردازیم. چرخه تشخیص فعالیت های انسان به صورت زیر است:



قبل از اینکه وارد توضیح استخراج ویژگی و کدهای مربوط به آن شویم ساختار پروژه را بررسی می کنیم.

قسمت یادگیری ماشین پروژه با زبان پایتون پیاده سازی شده است که ساختار آن به صورت زیر است:

- یک فایل پایتون برای پیش پردازیش داده ها (preprocess.py)
- یک فایل برای جمع آوری دیتاست به صورت ورودی مورد انتظار للگوریتم کلاس بندی(concat.py)
- یک فایل مختص کلاس بندی(svm.py)
- یک فایل برای پیشبینی کردن ورودی جدید(main.py)

ابزارها

- Pandas : یک کتابخانه نرم‌افزاری نوشته شده برای زبان برنامه‌نویسی پایتون برای دستکاری و تجزیه و تحلیل داده‌ها است. به‌طور خاص، پانداس ساختارها و عملیات برای دستکاری جداول و سری‌های زمانی را ارایه می‌دهد. پانداس نرم‌افزار آزاد است و تحت مجوز بی‌اس‌دی منتشر شده‌است. نامش برگرفته از عبارت داده‌های پانل در اقتصادسنجی است که برای مشاهدات مجموعه داده‌های دوره‌های زمانی چندگانه است که برای موارد یکسان را شامل می‌شوند.
این کتابخانه به دلیل داشتن ساختارهای داده‌ای مناسب برای تمیز کردن داده‌های خام (داده‌هایی که از منبع به دستِ کاربر می‌رسد) و ابزارهایی برای پر کردن داده‌های از دست رفته، به شدت میان دانشمندان داده محبوب شده‌است.

- امکانات کتابخانه:
  - ایجاد دیتافریم برای دستکاری داده‌ها با شاخص‌گذاری یکپارچه.
  - ابزارهایی برای خواندن و نوشتن داده‌ها بین ساختارهای داده حافظه و فرمت‌های فایل مختلف.
  - همترازی داده‌ها و مدیریت یکپارچه داده‌های از دست رفته.
  - تغییرشکل و چرخش‌های مجموعه داده‌ها.
  - برش توسط برچسب، نمایه فنسی، و خرد کردن داده‌های بزرگ.
  - درج و حذف ستون ساختار داده.
  - گروه‌بندی و اعمال عملیات ترکیبی بر روی مجموعه‌ای از داده‌ها
  - ادغام و اتصال داده
  - از نمایه‌سازی محوری سلسله مراتبی برای کار با داده‌های چند بعدی در ساختار داده‌های چند بعدی استفاده می‌کند.
  - سری‌های زمانی - قابلیت: تولید محدوده زمانی و تبدیل فرکانس، آمار پنجره متحرک، رگرسیون خطی پنجره متحرک، تغییر تاریخ و عقب‌ماندگی.
  - امکان فیلتراسیون یا پالایش داده را فراهم می‌کند.

- Scikit-learn : یک کتابخانه یادگیری ماشین رایگان برای زبان برنامه نویسی پایتون است که شامل الگوریتم های مختلف طبقه بندی ، رگرسیون و خوشه بندی از جمله ماشین های بردار پشتیبانی ، جنگل های تصادفی ، gradient boosting ، k-means و DBSCAN است و برای کار با کتابخانه های عددی و علمی پایتون NumPy و SciPy طراحی شده است.
Scikit-learn تا حد زیادی در پایتون نوشته شده است ، و از numpy به طور گسترده برای عملکردهای جبر خطی و آرایه استفاده می کند. علاوه بر این ، برخی الگوریتم های اصلی در Cython برای بهبود عملکرد نوشته شده اند. ماشین های بردار پشتیبانی توسط یک بسته بندی سایتون در اطراف LIBSVM پیاده سازی می شوند. رگرسیون لجستیک و ماشین های بردار پشتیبانی خطی توسط یک بسته بندی مشابه در اطراف LIBLINEAR پیاده سازی می شوند. در چنین مواردی ، گسترش این روش ها با Python ممکن نیست.
Scikit-learnبه خوبی با بسیاری از کتابخلنه های پایتون دیگر مانند matplotlib و plotly برای رسم نمودار ، numpy برای بردار آرایه ، فریم های داده pandas، scipy و بسیاری دیگر سازگار است.

فایل پیش پردازش داده ها

- پیش پردازش داده ها یک گام جدایی ناپذیر در یادگیری ماشـــین اســـت زیرا کیفیت داده ها و اطلاعات مفیدی که می توان از آنها گرفت به طور مســـتقیم بر تولنایی یادگیری مدل ما تأثیر می گذارد. بنابراین ، بســـیار مهم اســـت که ما داده های خود را قبل از تغذیه آنها در مدل خود پیش پردازش کنیم.
- اســـتخراج ویژگی فرایندی اســـت که در آن با انجام عملیاتی بر روی داده‌ها، ویژگی‌های بارز و تعیین‌کننده آن مشخص می‌شود. هدف استخراج ویژگی این است که داده‌های خام به شکل قابل استفاده‌تری برای پردازش‌های آماری بعدی درآیند.

حال به توضیح کد می پردازیم:

در ابتدا کتابخانه های مورد نیاز را ایمپورت می کنیم.

```
# needed libraries
import os
import pandas as pd
```

کتابخانه os برای کار با فایل های موجود در دایرکتوری پروژه و کتابخانه pandas برای کار با دیتاست ها و استخراج ویژگی هایمان مورد استفاده قرار می گیرد.

در قدم بعد نام پوشه حرکتی که قصد پیش پردازش و استخراج ویژگی آن را داریم از ورودی گرفته و پس از استخراج ویژگی و پردازش در یک فایل csv جدید که نام آن را نیز از از ورودی گرفته ایم ذخیره می کنیم.

```
# input
move = input("Please enter move: ")
name = input("Please enter csv name: ")
```

حال نام فایل های موجود در پوشه گرفته شده را درون یک متغیر می ریزیم.

```
# path of dataset
path = f'dataset/{move}/'
files = os.listdir(path)
```

قبل از بررسی کد استخراج ویژگی به توضیح ویژگی ها می پردازیم:

- میانگین:

میانگین (Mean)، مقدار متوسط اعداد است. نام دیگر آن معدل است. مسلماً همه ما با این واژه در کارنامه‌های تحصیلی خود آشنا هستیم. میانگین شاخصی از نحوه پراکندگی اعضای یک مجموعه را به دست می‌دهد. محاسبه میانگین کار آسانی است. کافی است کل اعداد مورد نظر را با هم جمع

کنید، سپس حاصل را بر تعداد شماره‌های جمع شده تقسیم کنید. به عبارت دیگر، میانگین، از تقسیم مجموع اعداد بر تعداد آنها به دست می‌آید.

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

- میانه

میانه (Median) در آمار و نظریه احتمالات یکی از سنجش‌های گرایش به مرکز است. میانه عددی است که یک جمعیت آماری یا یک توزیع احتمالی را به دو قسمت مساوی تقسیم می‌کند. یکی از مزیت‌های مهم میانه نسبت به میانگین این است که میانه از اعداد بسیار بزرگ و بسیار کوچک مجموعه اندازه‌ها متأثر نمی‌شود. برای پیدا کردن میانه در یک مجموعهN عضوی مجموعه شمارا، ابتدا باید اعداد را از کوچک به بزرگ مرتب کرد.

اگر تعداد اعداد مجموعه مرتب شده، فرد باشد عدد وسط میانه خواهد بود. اگر تعداد اعداد مجموعه مرتب شده، زوج باشد، میانه برابر میانگین دو عدد میانی خواهد بود.

- انحراف معیار

انحراف معیار (standard deviation) (نماد σ) یکی از شاخص های پراکندگی است که نشان می‌دهد به‌طور میانگین داده‌ها چه مقدار از مقدار متوسط فاصله دارند. اگر انحراف معیار مجموعه‌ای از داده‌ها نزدیک به صفر باشد، نشانه آن است که داده‌ها نزدیک به میانگین هستند و پراکندگی اندکی دارند؛ در حالی که انحراف معیار بزرگ بیانگر پراکندگی قابل توجه داده‌ها می‌باشد. انحراف معیار برابر ریشه دوم واریانس است. خوبی آن نسبت به واریانس، این است که هم بعد با داده‌ها می‌باشد. انحراف معیار برای تعیین ضریب اطمینان در تحلیل‌های آماری نیز به کار می‌رود. در مطالعات علمی، معمولاً داده‌های با اختلاف بیشتر از دو انحراف معیار از مقدار میانگین به عنوان داده‌های پرت در نظر گرفته و از تحلیل، خارج می‌شوند. انحراف معیار برای یک مجموعه متناهی، برابر است با جذر میانگین مربعات اختلاف داده‌ها با میانگینشان. مقدار انحراف معیار به دست آمده در صورتی درست است که از همه جمعیت موجود استفاده شود. اگر نمونه‌های تصادفی از داده‌ها انتخاب شده و انحراف معیار برای آن نمونه‌ها به دست آید، باید یک واحد از مقدار مخرج در گام پیش از نهایی کم شود.

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

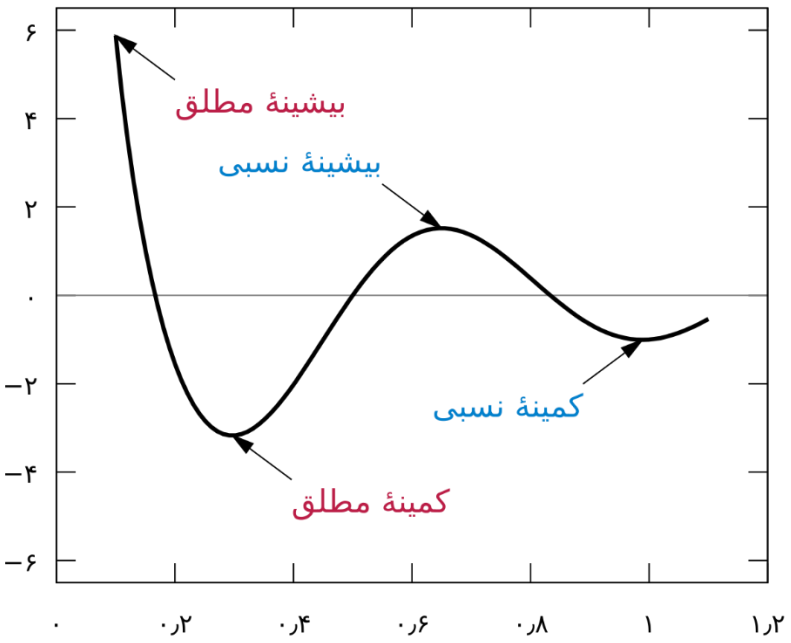
- واریانس

در نظریه احتمالات و آمار واریانس (Variance) نوعی سنجش پراکندگی است. مقدار واریانس با میانگین‌گیری از مربع فاصله مقدار محتمل یا مشاهده شده با مقدار مورد انتظار محاسبه می‌شود. در مقایسه با میانگین می‌توان گفت که میانگین مکان توزیع را نشان می‌دهد، در حالی که واریانس مقیاسی است که نشان می‌دهد که داده‌ها حول میانگین چگونه پخش شده‌اند. واریانس کمتر بدین معنا است که انتظار می‌رود که اگر نمونه‌ای از توزیع مزبور انتخاب شود مقدار آن به میانگین نزدیک باشد. یکای واریانس مربع یکای کمیت اولیه می‌باشد. ریشه دوم واریانس که انحراف معیار نامیده می‌شود دارای واحدی یکسان با متغیر اولیه است.

$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$$

- بیشینه و کمینه

بیشینه (ماکسیمم) و کمینه (مینیمم) تابع در یک بازه، به بزرگترین مقدار و کوچکترین مقدار تابع در آن بازه گفته می‌شود. در اصطلاح به ماکسیمم و همین‌طور مینیمم، نقاط اکسترمم تابع گفته می‌شود. منظور ما در این بخش بیشنیه و کمینه مطلق است.



- چارک اول

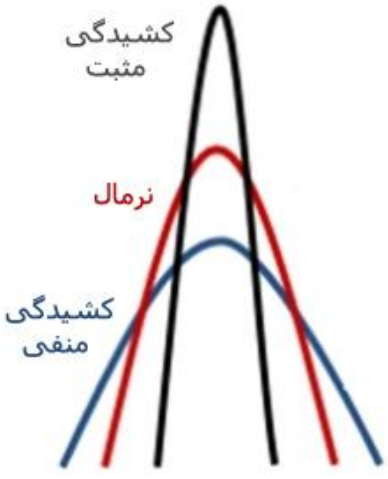
مشـــاهده ای از مجموعه داده های مورد بررســـی اســـت که یک چهارم داده ها (یعنی ۲۵ درصـــد مشاهدات) از آن کوچکتر و سه چهارم داده ها (یعنی ۷۵ درصد مشاهدات) از آن بزرگتر می باشد.
روش بدست آوردن چارک اول به این صورت است که ابتدا میانه داده ها را بدست آورده سپس برای نیمه اول داده ها (از کوچکترین عدد تا میانه) مجددا یکبار دیگر میانه را محاســـبه می نماییم. این عدد که میانه نیمه اول داده ها است همان چارک اول می باشد.

- چارک سوم

مشـــاهده ای از مجموعه داده های مورد بررســـی اســـت که ســـه چهارم داده ها (یعنی ۷۵ درصـــد مشاهدات) از آن کوچکتر و یک چهارم داده ها (یعنی ۲۵ درصد مشاهدات) از آن بزرگتر می باشد.
روش بدست آوردن چارک سوم به این صورت است که ابتدا میانه داده ها را بدست آورده سپس برای نیمه دوم داده ها (از میانه تا بزرگترین عدد) مجددا یکبار دیگر میانه را محاســـبه می نماییم. این عدد که میانه نیمه دوم داده ها است همان چارک سوم می باشد.

- کشیدگی

کشـیدگی (kurtosis) توزیع داده ها به عبارت سـاده به همان ارتفاع توزیع داده ها اشـاره می کند. براسـاس یک تعریف علمی کشـیدگی برابر با گشـتاور چهارم نرمال شـده‌اسـت. کشـیدگی معیاری از تیزی منحنی در نقطه ماکزیمم است. در آمار کشـیدگی توصـیف کننده میزان قله‌ای بودن و مسـطح بودن یک توزیع احتمالی اسـت. هرچقدر شـکل تابع چگالی احتمال (قله ای تر و دارای دم پهن تر یا دنباله پهن ترباشد میزان شاخص کشیدگی برای آن بیشتر است.

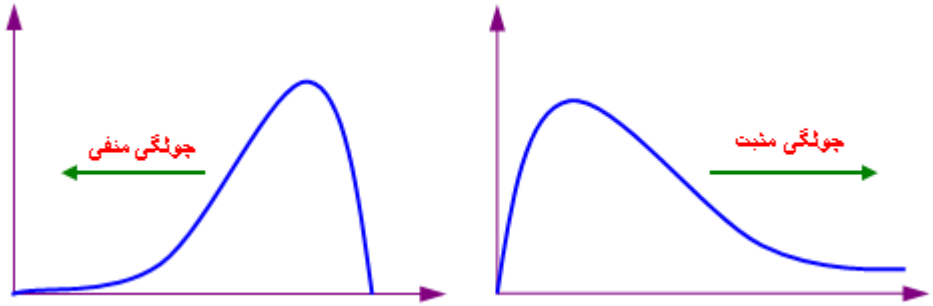


$$\text{Kurtosis} = \frac{\sum_{i=1}^{N} \frac{\left(X_i - \bar{X}\right)}{N}}{s^4}$$

where,
$\bar{X}$ is the mean,
s is the standard deviation
and N is the sample size

- چولگی

در تئوری احتمال و آمار، چولگی (skewness) بیانگر میزان عدم تقارن توزیع احتمال داده‌ها حول میانگینشان است. مقدار چولگی می‌تواند منفی یا مثبت باشد. ممکن است تصور شود میزان تمایل منحنی توزیع احتمال یک سری داده، چولگی است ولی این معیار بیانگر عدم تقارن در دم‌های این منحنی است. در حالتی که داده‌ها دارای توزیع متقارن باشند میزان کشیدگی دم‌های سمت راست و چپ یکی است.



- دامنه میان چارکی

دامنه بین چارکی (interquartile range) یک شاخص برای پراکندگی است که اختلاف صدک ۲۵ و ۷۵ را محاسبه می کند.

دامنه بین چارکی برخلاف دامنه تغیرات کمتر توسط مقادیر انتهایی تحت تاثیر قرار می گیرد.

- مقدار مؤثر

در ریاضیات، جذر متوسط مربع که با نام مقدار RMS و مقدار مؤثر نیز شناخته می‌شود، معیاری آماری از اندازه کمیت متغیر است.

$$f_{\mathrm{rms}} = \lim_{T\to\infty} \sqrt{\frac{1}{2T}\int_{-T}^{T} [f(t)]^2\,dt}.$$

ذکر این نکته ضروری است که دو ویژگی اخر تاثیر کمی داشتند به این دلیل آن ها را حذف کردیم.

حال که با ویژگی ها آشنا شدیم نوبت به بررسی کد این قسمت است. ابتدا فایل csv را به کمک pandas باز می کنیم. سپس دیتا ژیروسکوپ و شتاب سنج را جدا می کنیم. حال دیتا ژیروسکوپ و شتاب سنج را نرمال سازی می کنیم. بعد استخراج ویژگی به کمک pandas برای هر سنسور به صورت جداگانه صورت می گیرد. در آخر نیز ویژگی ها را در یک لیست قرار داده و آن لیست را به یک لیست نهایی اضافه می کنیم. سپس ستون ها را نیز به یک لیست دیگر اضافه می کنیم. حال مجموعه ستون ها و لیستی از ویژگی های هر دیتاست را داریم که با کمک pandas این دو لیست را به دیتافریم تبدیل کرده و در نهایت به صورت فایل csv ذخیره می کنیم.

عکس تکه کد توضیح داده شده در صفحات بعدی می باشد.

```
# feature extracting
for file in files:

    # read csv file
    df = pd.read_csv(path + file)

    # select sensors
    gyroscope_df = df.loc[
        df['ACC_or_GYR'] == 'GYR'
        ].reset_index(drop=True)
    accumulator_df = df.loc[
        df['ACC_or_GYR'] == 'ACC'
        ].reset_index(drop=True)

    # normalized gyro
    gyro_max = gyroscope_df[['x', 'y', 'z']].max()
    df_max = gyro_max.max()
    gyroscope_df = gyroscope_df[['x', 'y', 'z']] / df_max

    # normalized acc
    acc_max = accumulator_df[['x', 'y', 'z']].max()
    df_max = acc_max.max()
    accumulator_df = accumulator_df[['x', 'y', 'z']] / df_max

    # 25 gyro axis
    gyro_x_25 = gyroscope_df['x'].quantile(0.25)
    gyro_y_25 = gyroscope_df['y'].quantile(0.25)
    gyro_z_25 = gyroscope_df['z'].quantile(0.25)

    # 25 acc axis
    acc_x_25 = accumulator_df['x'].quantile(0.25)
    acc_y_25 = accumulator_df['y'].quantile(0.25)
    acc_z_25 = accumulator_df['z'].quantile(0.25)

    # 75 gyro axis
    gyro_x_75 = gyroscope_df['x'].quantile(0.75)
    gyro_y_75 = gyroscope_df['y'].quantile(0.75)
    gyro_z_75 = gyroscope_df['z'].quantile(0.75)

    # 75 acc axis
    acc_x_75 = accumulator_df['x'].quantile(0.75)
    acc_y_75 = accumulator_df['y'].quantile(0.75)
    acc_z_75 = accumulator_df['z'].quantile(0.75)
```

```
    # mean gyro axis
    gyro_x_mean = gyroscope_df['x'].mean()
    gyro_y_mean = gyroscope_df['y'].mean()
    gyro_z_mean = gyroscope_df['z'].mean()

    # mean acc axis
    acc_x_mean = accumulator_df['x'].mean()
    acc_y_mean = accumulator_df['y'].mean()
    acc_z_mean = accumulator_df['z'].mean()

    # median gyro axis
    gyro_x_median = gyroscope_df['x'].median()
    gyro_y_median = gyroscope_df['y'].median()
    gyro_z_median = gyroscope_df['z'].median()

    # median acc axis
    acc_x_median = accumulator_df['x'].median()
    acc_y_median = accumulator_df['y'].median()
    acc_z_median = accumulator_df['z'].median()

    # std gyro axis
    gyro_x_std = gyroscope_df['x'].std()
    gyro_y_std = gyroscope_df['y'].std()
    gyro_z_std = gyroscope_df['z'].std()

    # std acc axis
    acc_x_std = accumulator_df['x'].std()
    acc_y_std = accumulator_df['y'].std()
    acc_z_std = accumulator_df['z'].std()

    # var gyro axis
    gyro_x_var = gyroscope_df['x'].var()
    gyro_y_var = gyroscope_df['y'].var()
    gyro_z_var = gyroscope_df['z'].var()

    # var acc axis
    acc_x_var = accumulator_df['x'].var()
    acc_y_var = accumulator_df['y'].var()
    acc_z_var = accumulator_df['z'].var()

    # min gyro axis
    gyro_x_min = gyroscope_df['x'].min()
    gyro_y_min = gyroscope_df['y'].min()
    gyro_z_min = gyroscope_df['z'].min()
```

```
    # min acc axis
    acc_x_min = accumulator_df['x'].min()
    acc_y_min = accumulator_df['y'].min()
    acc_z_min = accumulator_df['z'].min()

    # max gyro axis
    gyro_x_max = gyroscope_df['x'].max()
    gyro_y_max = gyroscope_df['y'].max()
    gyro_z_max = gyroscope_df['z'].max()

    # max acc axis
    acc_x_max = accumulator_df['x'].max()
    acc_y_max = accumulator_df['y'].max()
    acc_z_max = accumulator_df['z'].max()

    # kurtosis gyro axis
    gyro_x_kurtosis = gyroscope_df['x'].kurtosis()
    gyro_y_kurtosis = gyroscope_df['y'].kurtosis()
    gyro_z_kurtosis = gyroscope_df['z'].kurtosis()

    # kurtosis acc axis
    acc_x_kurtosis = accumulator_df['x'].kurtosis()
    acc_y_kurtosis = accumulator_df['y'].kurtosis()
    acc_z_kurtosis = accumulator_df['z'].kurtosis()

    # skew gyro axis
    gyro_x_skew = gyroscope_df['x'].skew()
    gyro_y_skew = gyroscope_df['y'].skew()
    gyro_z_skew = gyroscope_df['z'].skew()

    # skew acc axis
    acc_x_skew = accumulator_df['x'].skew()
    acc_y_skew = accumulator_df['y'].skew()
    acc_z_skew = accumulator_df['z'].skew()

    item = [
        move,
        gyro_x_mean,
        gyro_y_mean,
        gyro_z_mean,
        acc_x_mean,
        acc_y_mean,
        acc_z_mean,
        gyro_x_median,
        gyro_y_median,
```

```
        gyro_z_median,
        acc_x_median,
        acc_y_median,
        acc_z_median,
        gyro_x_std,
        gyro_y_std,
        gyro_z_std,
        acc_x_std,
        acc_y_std,
        acc_z_std,
        gyro_x_var,
        gyro_y_var,
        gyro_z_var,
        acc_x_var,
        acc_y_var,
        acc_z_var,
        gyro_x_min,
        gyro_y_min,
        gyro_z_min,
        acc_x_min,
        acc_y_min,
        acc_z_min,
        gyro_x_max,
        gyro_y_max,
        gyro_z_max,
        acc_x_max,
        acc_y_max,
        acc_z_max,
        gyro_x_25,
        gyro_y_25,
        gyro_z_25,
        acc_x_25,
        acc_y_25,
        acc_z_25,
        gyro_x_75,
        gyro_y_75,
        gyro_z_75,
        acc_x_75,
        acc_y_75,
        acc_z_75,
        gyro_x_kurtosis,
        gyro_y_kurtosis,
        gyro_z_kurtosis,
        acc_x_kurtosis,
        acc_y_kurtosis,
```

```
        acc_z_kurtosis,
        gyro_x_skew,
        gyro_y_skew,
        gyro_z_skew,
        acc_x_skew,
        acc_y_skew,
        acc_z_skew,
    ]
    items.append(item)

columns = [
    'type',
    'mean gyro x',
    'mean gyro y',
    'mean gyro z',
    'mean acc x',
    'mean acc y',
    'mean acc z',
    'median gyro x',
    'median gyro y',
    'median gyro z',
    'median acc x',
    'median acc y',
    'median acc z',
    'std gyro x',
    'std gyro y',
    'std gyro z',
    'std acc x',
    'std acc y',
    'std acc z',
    'var gyro x',
    'var gyro y',
    'var gyro z',
    'var acc x',
    'var acc y',
    'var acc z',
    'min gyro x',
    'min gyro y',
    'min gyro z',
    'min acc x',
    'min acc y',
    'min acc z',
    'max gyro x',
    'max gyro y',
    'max gyro z',
```

```
    'max acc x',
    'max acc y',
    'max acc z',
    '25 gyro x',
    '25 gyro y',
    '25 gyro z',
    '25 acc x',
    '25 acc y',
    '25 acc z',
    '75 gyro x',
    '75 gyro y',
    '75 gyro z',
    '75 acc x',
    '75 acc y',
    '75 acc z',
    'gyro x kurtosis',
    'gyro y kurtosis',
    'gyro z kurtosis',
    'acc x kurtosis',
    'acc y kurtosis',
    'acc z kurtosis',
    'gyro x skew',
    'gyro y skew',
    'gyro z skew',
    'acc x skew',
    'acc y skew',
    'acc z skew',
]

new_df = pd.DataFrame(data=items, columns=columns)
new_df.to_csv(f'{name}.csv', index=False)
```

فصل چهارم

پیاده‌سازی برنامه موبایل

زبان برنامه نویسی برای پیاده سازی اپلیکیشن موبایل c++ می باشد که از فریمورک qt استفاده می کنیم.

Qt یا کیوت یک چارچوب نرم افزاری چند پلتفرمی یا به عبارتی کراس پلتفرم است و از آن برای توسعه نرم افزار های کاربردی که می توان آن ها را بر روی پلتفرم های مختلف سخت افزاری و نرم افزاری و بدون تغییر یا با تغییرات خیلی کم در کد اصلی، اجرا کرد استفاده می شود و در عین حال یک محیط توسعه نرم افزار از نوع Native به شمار می رود و از توانایی ها و سرعت این حالت برنامه نویسی برخوردار می باشد.

Qt در حال حاضر به صورت کد منبع باز توسط شرکت Qt و تیم پروژه Qt شامل توسعه دهندگان مختلف و شرکت های مختلف که برای پیشبرد Qt تلاش می کنند، در حال توسعه یافتن می باشد. Qt هم به صورت تجاری و هم به صورت کد منبع باز و تحت پروانه های GPL 2.0، GPL 3.0 و LGPL 3.0 در دسترس می باشد.

Qt برای توسعه برنامه های کاربردی چند پلتفرمی و رابط های کاربری گرافیکی (GUI ها) مورد استفاده قرار می گیرد با این وجود برنامه های بدون رابط کاربری گرافیکی مانند ابزار های Command Line و یا کنسول سرور ها را نیز می توان به کمک آن توسعه داد.

مثالی از یک برنامه بدون رابط کاربری گرافیکی که با Qt توسعه داده شده است فریم ورک تحت وب Cutelyst می باشد. برنامه های دارای رابط کاربری گرافیکی که با Qt توسعه داده شده اند می توانند رابط کاربری مشابه حالت Native داشته باشند که در این صورت Qt در دسته ابزارهای ویجت قرار می گیرد.

Qt از c++ استاندارد به همراه افزونه هایی شامل سیگنال ها و شکاف ها استفاده می کند (سیگنال و اسلات یه ساختار زبان به کار رفته در QT می باشد که برای ارتباط بین اشیا معرفی شده است) که این مدیریت رویداد ها را آسان تر می کند و این خود به توسعه رابط کاربری گرافیکی و برنامه های کاربردی سرور که اطلاعات رویداد مخصوص به خود را دارند و باید آن ها را به نحو مناسب پردازش کنند کمک می کند.

Qt از کامپایلر های متعددی پشتیبانی می کند که از بین آن ها می توان به کامپایلرGCC و نیز ویژوال استودیو اشاره کرد. Qt همچنین فریم ورک Qt Quick را نیز ارائه می کند که شامل یک زبان اسکریپت نویسی و به عبارتی یک زبان مدل سازی به نام QML است که امکان استفاده از جاوا اسکریپت برای بخش منطقی را فراهم می کند. با Qt Quick، توسعه سریع برنامه های کاربردی برای گوشی های تلفن همراه امکان پذیر شد البته می توان منطق را با استفاده از کد Native نیز نوشت تا بهترین عملکرد ممکن به دست آید.

Qt می تولند با کمک قید های زبانی به چند زبان برنامه نویسی دیگر نیز مورد استفاده قرار گیرد. این چارچوب بر روی پلتفرم های اصلی دسکتاپ و بر خی از پلتفرم های موبایل اجرا می شود و پشتیبانی جامعی

از بین المللی سازی دارد. ویژگی های بدون نیاز به رابط کاربری گرافیکی شامل دسترسی به بانک های اطلاعاتی SQL، تجزیه XML، تجزیه JSON، مدیریت ریسه ها و پشتیبانی شبکه می باشد.

چندین رابط کاربری گرافیکی و محیط های دسکتاپ از Qt به عنوان ابزار ویجت استفاده می کنند.

- AsteroidOS: سیستم عاملی منبع باز برای ساعت های هوشمند
- Avionics: سیستم سرگرمی داخل پرواز شرکت پاناسونیک
- DDE (محیط دسکتاپ Deepin) برای DDE لینوکس
- Hawaii: یک محیط دسکتاپ بر پایه Wayland و Qt Quick
- KDE Plasma: یک محیط دسکتاپ برای شکل های مختلف مانند کامپیوتر ها، تبلت ها و گوشی های هوشمند
- LiriOS: یک محیط کار بر مبنای Qt/QML
- Lumina: یک محیط دسکتاپ که برای PC-BSD طراحی شده است
- LXQt(محیط دسکتاپ سبک X11): جانشین LXDE که بر مبنای Qt طراحی شده است.
- OPIE: یک رابط کاربری گرافیکی برای Sharp Zaurus
- Sailfish OS: یک سیستم عامل تلفن همراه که توسط شرکت Jolla توسعه داده شده است.
- Sky Q: سیستم سرگرمی خانگی برای شرکت Sky
- Tesla Model S: مورد استفاده در رابط کاربری ماشین های تسلا
- Ubuntu Touch: یک رابط کاربری تلفن همراه که توسط شرکت Canonical توسعه داده شده است.
- webOS: یک سیستم عامل چند کاره برای دستگاه های هوشمند نظیر تلویزیون ها و ساعت های هوشمند
- Yunit: شاخه جامعه کاربری Unity8-Shell

نرم افزار های مهمی که از Qt یا QML استفاده می کنند:

- Adobe Photoshop Elements
- Radeon Software Crimson
- Autodesk Maya
- Bitcoin Core
- CryEngine
- Dragonframe: نرم افزار انیمیشن های استاپ موشن
- FreeMat: محیطی کد منبع باز برای محاسبات عددی
- Google Earth
- Orange: بسته ویژه داده کاوی
- QGIS: سیستم اطلاعات جغرافیایی

- Scribus: نرم افزار انتشاری برای دسکتاپ
- Sibelius: نرم افزار ساخت قطعات موسیقی
- Skype
- Spotify نسخه لینوکس
- Stellarium یک نرم افزار آسمان نما
- Subsurface: برنامه ای برای ثبت و برنامه ریزی غواصی که برای اولین بار توسط Linus Torvalds توسعه داده شد.
- Telegram Desktop: نرم افزاری پیام رسان که برای ویندوز، مک و لینوکس در دسترس است.
- VirtualBox: بسته نرم افزاری مجازی سازی سیستم عامل ها

مفهوم کامل رابط کاربری گرافیکی

در زمان عرضه، Qt از موتور رنگ و کنترل های مخصوص خود استفاده می کرد که هنگامی که ویجت های مورد نظر کشیده می شد ظاهر آن ها را در پلتفرم های مختلفی که قرار بود بر روی آن ها اجرا شود شبیه سازی می کرد. این قابلیت عملیات پورت بین پلتفرم های مختلف را آسان تر می کرد زیرا تنها تعداد کمی دسته در Qt حقیقتا به پلتفرم هدف وابسته بودند. با این وجود این قضیه گاها باعث ظهور اختلافات کوچکی می شد که علت آن بی نقض نبودن عملیات شبیه سازی بود. نسخه های جدید Qt، در پلتفرم هایی که دارای مجموعه ابزار ویجت به صورت نیتیو هستند از API های نیتیو پلتفرم های مختلف استفاده می کنند تا معیار های مختلف را جستجو کنند، بیشترین کنترل ممکن را داشته باشند و در نهایت دچار چنین مشکلاتی نشوند. در برخی از پلتفرم ها(مانند MeeGo و Qt (KDE به عنوان API نیتیو به شمار می رود. برخی جعبه ابزار های گرافیکی قابل حمل تصمیم های طراحی متفاوتی گرفته اند برای مثال wxWidgets از جعبه ابزار های پلتفرم هدف برای اجرای خود استفاده می کند.

سیگنال ها و اسلات ها در کیوت Qt

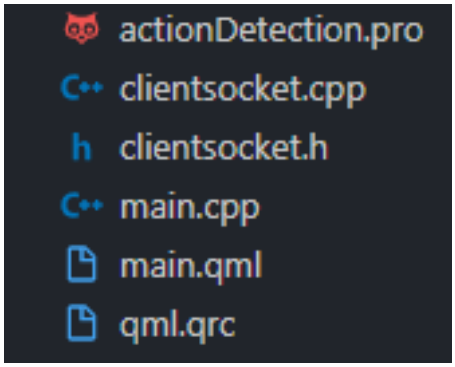
یک واحد زبانی مورد استفاده در Qt که بین اشیا مختلف ارتباط برقرار می کند. این کار اجرای observer pattern را راحت می کند و در عین حال از ایجاد کد boilerplate جلوگیری می کند. مفهوم کلی به این صورت است که ویجت های رابط کاربری گرافیکی می توانند سیگنال هایی را ارسال کنند که حاوی اطلاعات مربوط به رویداد هاست که این اطلاعات توسط کنترل های دیگر و به کمک توابعی ویژه که به آن ها شکاف ها گفته می شود دریافت می شوند .

کامپایلر Meta Object

کامپایلر Meta Object که به اختصار moc خوانده می شود ابزاری است که با استفاده منابع یک نرم افزار Qt اجرا می شود.این ابزار برخی از ماکرو های کد ++c را به عنوان یادداشت ترجمه می کند و از آن برای تولید کد افزوده ++c به همراه اطلاعات meta در مورد کلاس های به کار رفته در برنامه، استفاده می کند. این اطلاعات meta توسط Qt مورد استفاده قرار می گیرند تا تا ویژگی های برنامه نویسی را که به

صـــورت native در c++ موجود نیســـت در اختیار کاربر بگذارند. این ویژگی ها عبارتند از: ســـیگنال ها و شکاف ها،درون گرایی و فراخوانی توابع به صورت غیر هم زمان.

ساختار پوشه برنامه موبایل به صورت زیر است:



ابتدا به توضیح actionDetection.pro می پردازیم.

```
QT += quick sensors network
CONFIG += c++11

# The following define makes your compiler emit warnings if you use
# any Qt feature that has been marked deprecated (the exact warnings
# depend on your compiler). Refer to the documentation for the
# deprecated API to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
        main.cpp \
    clientsocket.cpp

RESOURCES += qml.qrc

# Additional import path used to resolve QML modules in Qt Creator's code model
QML_IMPORT_PATH =

# Additional import path used to resolve QML modules just for Qt Quick Designer
QML_DESIGNER_IMPORT_PATH =

# Default rules for deployment.
qnx: target.path = /tmp/${TARGET}/bin
else: unix:!android: target.path = /opt/${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

HEADERS += \
    clientsocket.h
```

در ابتدا ماژول های مورد نیاز را تعیین می کنیم.

- Quick : ماژول نرم افزار های کاربردی دارای رابط کاربری گرافیکی که با استفاده از ۲QML نوشته شده اند
- Sensors : دسته هایی برای دسترسی به سنسور های سخت افزاری مختلف موبایل ها. در Qt ۴.۰ بخشی از ماژول Qt Mobile به شمار می رفت. بر روی اندروید، بلک بری، ios، WinRT، Mer و لینوکس پشتیبانی می شود.

- Network :لایه انتزاعی شبکه به همراه پشتیبانی از TCP, UDP, HTTP, SSL و از نسخه ۵.۳ به بعد با پشتیبانی از SPDY

قدم بعدی تعیین ورژن کامپایلر c++ می باشد.

حال فایل های cpp و qml را تعیین می کنیم.

در آخر نیز قوانینی برای دیپلوی کردن و همچنین فایل های header را تعیین می کنیم.

فایل بعدی main.py می باشد.

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include "clientsocket.h"

int main(int argc, char *argv[])
{
    QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);

    QGuiApplication app(argc, argv);

//    ClientSocket mSocket;
//    mSocket.test();

    //-- QSetting configuration --//
    app.setOrganizationName("univesity of tehran");
    app.setOrganizationDomain("ut.ac.ir");
    app.setApplicationName("Activity detection");

    //-- socket class --//
    qmlRegisterType<ClientSocket>("com.socket", 1, 0, "ClientSocket");

    QQmlApplicationEngine engine;
    engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
    if (engine.rootObjects().isEmpty())
        return -1;

    return app.exec();
}
```

این فایل اصلی برنامه می باشد که اجرا می شود. در این فایل ابتدا core ای برای کار با گوشی های مختلف تعریف می کنیم . سپس اپلیکیشن گرافیکی به نام app می سازیم. قدم بعدی تعیین نام سازمان و نام برنامه می باشد. سپس کلاس clientsocket را به فایل qml متصل می کنیم. در آخر نیز یک برنامه qml ساخته و فایل qml خود را به آن می دهیم تا آن را اجرا کند.

فایل بعدی clientsocket.h می باشد که header ای برای clientsocket.cpp است که در آن یک کلاس clientsocket ساخته شده است.

```
#ifndef CLIENTSOCKET_H
#define CLIENTSOCKET_H

#include <QObject>
#include <QDebug>
#include <QTcpSocket>
#include <QAbstractSocket>

class ClientSocket : public QObject
{
    Q_OBJECT
public:
    explicit ClientSocket(QObject *parent = nullptr);

signals:
    void msg(QString msg);

public slots:
    void connected();
    void dissconnected();
    void byteWritten(qint16 bytes);
    void readyRead();

    bool connectToIp(QString IP);
    void disConnect();
    void sendData(QString Data);

private:
    QTcpSocket *socket;
    bool isSocketConnected = false;
};

#endif // CLIENTSOCKET_H
```

در این کلاس:

- یک تابع سازنده
- یک سیگنال به نام msg
- هفت اسلات:
  - Connect
  - Dissconnect
  - Bytewritten
  - readyRead
  - connectToIp

- disConnect
- sendData

- یک اشاره گر خصوصی سوکت
- یک فلگ خصوصی برای بررسی وصل بودن به سوکت

فایل بعد clientsocket.cpp می باشد که به شرح زیر است.

```
#include "clientsocket.h"

ClientSocket::ClientSocket(QObject *parent) : QObject(parent)
{

}


void ClientSocket::connected()
{
    qDebug() << "connected";
    emit msg("connected");
}

void ClientSocket::dissconnected()
{
    qDebug() << "dissconnected";
    emit msg("dissconnected");
}

void ClientSocket::byteWritten(qint16 bytes)
{

    qDebug() << "we wrote " << bytes;
}

void ClientSocket::readyRead()
{

    qDebug() << "reading... ";
    QString str = socket->readAll();
    qDebug() << str;
}
```

تابع اول ،تابع سازنده است که قرار نیست کار انجام دهد و صرفا تعریف شده است.

تابع connect، برای برقراری ارتباط با سوکت می باشد که سیگنالی را برای qml می فرستد تا سمت کاربر از اتصال به سوکت آگاه باشد.

تابع dissconnect برای قطع ارتباط از سوکت می باشد که سیگنالی را برای qml می فرستد تا سمت کاربر از قطع شدن ارتباط آگاه باشد.

تابع bytewriiten برای نوشتن مورد استفاده قرار میگیرد ولی چون ما نیازی به آن نداریم صرفا تعریف شده است.

تابع readyRead برای خواندن اطلاعات از سوکت می باشد.

```
bool ClientSocket::connectToIp(QString IP)
{
    socket = new QTcpSocket(this);

    connect(socket, SIGNAL(connected()), this, SLOT(connected()));
    connect(socket, SIGNAL(dissconnected()), this, SLOT(dissconnected()));
    connect(socket, SIGNAL(readyRead()), this, SLOT(readyRead()));
    connect(socket, SIGNAL(byteWritten(qint16)), this, SLOT(byteWritten(qint16)));

    qDebug() << "connecting...";
    emit msg("connecting to " + IP + "...");

    socket->connectToHost(IP, 1234);

    if(!socket->waitForConnected(1000)){

        qDebug() << "Error:" << socket->errorString();
        emit msg("Error:" + socket->errorString());
        isSocketConnected = false;
        return false;
    }

    isSocketConnected = true;
    return true;
}

void ClientSocket::disConnect()
{
    socket->deleteLater();
    emit msg("disConnect");
}

void ClientSocket::sendData(QString Data)
{
    if(isSocketConnected){

        socket->write(Data.toUtf8());
    }
}
```

تابع connectToIp برای اتصال به سوکت مورد استفاده قرار می گیرد که در آن یک شی سوکت ساخته شده است و سپس با دستور connect چهار حالت ممکن را بررسی کرده ایم. سپس به ip لوکال سرور با پورت ۱۲۳۴ درخواست اتصال می دهیم. اگر ارتباط بعد از یک ثانیه برقرار نشد فلگ وضعیت اتصال را false می کنیم ولی اگر ارتباط برقرار شد فلگ را true کرده و سپس مقدار true که به معنای اتصال به سوکت است را بر می گردانیم.

تابع disConnect برای موقعی است که کاربر می خواهد ارتباط با سوکت را توسط خودش قطع کند.

تابع sendData نیز برای ارسال دیتا به سمت سرور می باشد که در ابتدا چک می کند که آیا اتصال به سوکت برقرار است یا خیر.

فایل بعدی main.qml می باشد که رابط کاربری اپلیکیشن در آن پیاده سازی شده است.

```
import QtQuick 2.9
import QtQuick.Controls 2.4
import QtQuick.Layouts 1.3
import QtSensors 5.9
import QtQuick.Window 2.2
import Qt.labs.settings 1.0
import com.socket 1.0

Window {
    visible: true
    width: 640
    height: 480
    title: qsTr("Hello World")

    property int cntr_gyros: 0
    property int cntr_acc: 0

    property var date
    property var startCaptureTime

    ClientSocket{
        id: socket

        //-- QString msg --//
        onMsg: {
            txt_error.text += msg + "\n"
        }
    }

    Settings{
            id: setting

            property alias ipAddress: txt_ip.text

        }
```

در ابتدا پکیج های مورد نظر را ایمپورت می کنیم. ســـپس پنجره ای برای قرار دادن آیتم هایمان ایجاد می کنیم که عرض و طول آن را مشخص کرده ایم و همچنین عنوانی برای این پنجره انتخاب کرده ایم. در ادامه ۴ ویژگی تعریف می کنیم:

- تعداد سنسور ژیروسکوپ
- تعداد سنسور شتاب سنج
- تاریخ
- زمان شروع جمع آوری اطلاعات

حال قسمتی را ایجاد می کنیم که پیغام های مرتبط به اتصال به سوکت رو در برنامه نمایش دهد.

در قسمت بعدی ip address وارد شده را در تنظیمات ذخیره می کنیم تا در دفعات بعد در برنامه موجود باشد.

```
//-- tools --//
ColumnLayout{
    anchors.fill: parent

    //-- connect ip --//
    Item {
        Layout.fillWidth: true
        Layout.preferredHeight: btn_connect.height * 1.5

        RowLayout{
            anchors.fill: parent

            Item{Layout.fillWidth: true}

            //-- connect button --//
            Button{
                id: btn_connect

                property bool isConnect: false

                text: isConnect ? "disconnect" : "connect"

                onClicked:{
                    if(!isConnect){

                        isConnect = socket.connectToIp(txt_ip.text)
                        cntr_acc= 0;
                        cntr_gyros= 0;
                    }
                    else{
                        accel.active = false
                        gyroscope.active = false
                        orient.active = false

                        socket.disConnect()
                        isConnect = false
                    }

                }
            }

            TextField{
                id: txt_ip

                text: "192.168.1.104"
                selectByMouse: true
            }

            Item{Layout.fillWidth: true}
        }
    }
```

در این قسمت باکس ورودی ip قرار دارد که ip نمایش داده شده در اپلیکیشن دسکتاپ را در آنجا وارد می کنیم. حال دکمه ای داریم که دو حالت ممکن است به وجود آورد.

اگر به دسکتاپ متصل نباشیم متن دکمه برابر connect است و در صورت کلیک کردن روی آن، به دسکتاپ متصل می شود و تعداد سنسور ژیروسکوپ و شتاب سنج را نیز برابر صفر قرار می دهیم.

اگر که به دسکتاپ متصل بود متن دکمه برابر disconnect است و در صورت کلیک کردن روی آن، اتصال به دسکتاپ و سوکت قطع می شود و تمامی سنسور ها را خاموش می کنیم.

```
//-- data rate --//
Item {
    Layout.fillWidth: true
    Layout.preferredHeight: btn_connect.height * 1.5

    RowLayout{
        anchors.fill: parent

        Item{Layout.fillWidth: true}

        Label{
            text: "data rate:"
        }

        //-- dataRate --//
        TextField{
            id: txt_dataRate

            text: "100"
            selectByMouse: true
        }

        Item{Layout.fillWidth: true}
    }
}
```

در این قسمت نرخ گرفتن داده را تعیین می کنیم.

```
//-- active Buttons --//
Item {
    Layout.fillWidth: true
    Layout.preferredHeight: btn_connect.height * 1.5

    RowLayout{
        anchors.fill: parent

        Item{Layout.fillWidth: true}

        Button{
            id: btn_active

            text: accel.active ? "deactivate Accelerometer/gyroscope" : "activate Accelerometer/gyroscope"
            onClicked: {
                var d = new Date()
                startCaptureTime = d.getTime();
                console.log("start date = " + startCaptureTime)
                gyroscope.active = !gyroscope.active
                accel.active = !accel.active
            }
        }

        Rectangle{
            Layout.preferredHeight: btn_active.height * 0.5
            Layout.preferredWidth: height
            radius: width * 0.5
            border.color: "black"
            border.width: 1
            smooth: true

            color:  accel.active ? "Green" : "Red"
        }

        Item{Layout.fillWidth: true}

    }

}
```

در این قسمت دکمه ای برای روشن و خاموش کردن سنسور ها قرار داده ایم که پس از برقراری ارتباط با سرور می توانیم از آن استفاده کنیم.

```
        //-- sensor data presenter --//
        Item {
            Layout.fillWidth: true
            Layout.fillHeight: true

            Label{
                id: lbl_data
                anchors.centerIn: parent
                text: ""
            }

            Label{
                id: lbl_gyr
                anchors.centerIn: parent
                anchors.verticalCenterOffset: 100
                text: ""
            }

            Label{
                id: lbl_orri
                visible: false
                anchors.centerIn: parent
                anchors.verticalCenterOffset: 200
                text: ""
            }

            TextArea {
                id: txt_error
                anchors.centerIn: parent
                anchors.verticalCenterOffset: 200
                anchors.fill: parent
                text: qsTr("")
            }
        }
    }
```

در این قسمت مقادیر لحظه ای سنسور ها در برنامه موبایل نمایش داده می شود.

```
    Accelerometer {
        id: accel

        property string buffer: "0_0_0"

        dataRate: 100 //parseInt(txt_dataRate.text)

        active: false
        alwaysOn: true

        onReadingChanged: {

            lbl_data.text = accel.reading.x + "\n" + accel.reading.y + "\n" + accel.reading.z
            var now = new Date()
            if(accel.active) socket.sendData(
                                "#" + "ACC" + "_" +
                                (now.getTime() - startCaptureTime) + "_" +
                                accel.reading.x + "_" +
                                accel.reading.y + "_" +
                                accel.reading.z
                                )

        }

    }

    Gyroscope {
        id: gyroscope
        dataRate: 100//parseInt(txt_dataRate.text)

        active: false
        alwaysOn: true

        onReadingChanged: {

            lbl_gyr.text = gyroscope.reading.x + "\n" + gyroscope.reading.y + "\n" + gyroscope.reading.z
            var now = new Date()
            if(gyroscope.active) socket.sendData(
                                    "#" + "GYR" + "_" +
                                    (now.getTime() - startCaptureTime) + "_" +
                                    gyroscope.reading.x + "_" +
                                    gyroscope.reading.y + "_" +
                                    gyroscope.reading.z
                                    )

        }

    }

}
```

در این قسمت سنسور های ژیروسکوپ و شتاب سنج را تعریف می کنیم و دو تا ویژگی active و alwaysOn را برای آن در نظر می گیریم. در قسمت onReadingChanged اگر سنسور فعال بود، اطلاعات سنسور را همراه با زمانشان به صورت لحظه ای به سمت سرور منتقل می کنیم.

پیاده سازی قسمت اتصال به پایتون برنامه تشخیص حرکت دسکتاپ

چارچوب کلی این برنامه توسط آقای براری پیاده سازی شده است و تنها بخش اتصال به فایل پایتون توسط اینجانب صورت گرفته است.

در این قسمت از کلاس Qprocess استفاده شده است. کلاس QProcess برای شروع برنامه های خارجی و برقراری ارتباط با آنها استفاده می شود.

```
#ifndef CONNECT2PY_H
#define CONNECT2PY_H

#include <QObject>
#include <QDebug>
#include <QProcess>


class Connect2Py : public QObject
{
    Q_OBJECT
public:
    Connect2Py(QObject *parent = nullptr);
    ~Connect2Py();

public slots:
    void readFromPython();
    void setActionFileUrl(QString address);
    void setPythonMainScript(QString script);

private slots:
    void printProcessStatus();
    void printProcessError();
    void printProcessOutput();

private:
    QProcess *process;
    QString fileUrl;
    QString pythonScript;

    const char *string2char(QString str);

signals:
    void detectAction(QString action);
};

#endif // CONNECT2PY_H
```

ابتدا به header آن اشاره می کنیم که درون آن یک کلاس برای اتصال به پایتون نوشته شده است که دارای:

۱. سازنده
۲. مخرب
۳. سه اسلات عمومی
۴. سه اسلات خصوصی
۵. سه متغیر خصوصی
۶. یک تابع خصوصی
۷. یک سیگنال

فایل بعدی connect2py.cpp می باشد که به شرح زیر است:

```cpp
#include <QStandardPaths>
#include <QDir>
#include "connect2py.h"

Connect2Py::Connect2Py(QObject *parent)
{
    pythonScript = "C:/Users/Amirhossein/PycharmProjects/bachelor-project/venv/Scripts/python.exe C:/Users/Amirhossein/PycharmProjects/bachelor-project/main.py";
}

Connect2Py::~Connect2Py()
{

    process->closeWriteChannel();
    process->close();
    process->deleteLater();
}

void Connect2Py::readFromPython()
{

    qDebug() << "start";
    process = new QProcess(this);
    process->setProcessChannelMode(QProcess::SeparateChannels);
    process->start(pythonScript);

    connect(process, SIGNAL(stateChanged(QProcess::ProcessState)), SLOT(printProcessStatus()));
    connect(process, SIGNAL(error(QProcess::ProcessError)), SLOT(printProcessError()));
    connect(process, SIGNAL(readyRead()), SLOT(printProcessOutput()));

}
```

در این قسمت فایل های مورد نیاز را اضافه می کنم سپس به تعریف تابع های کلاسمان می کنیم.

در سازنده محل پایتون و محل اسکریپت مورد نظر را در متغیری ذخیره می کنیم.

در مخرب چنلی که توسط Qprocess ایجاد شده است را پاک می کنیم.

در تابع readFromPython یک Qprocess ایجاد کرده و چنل های جداگانه را ست می کنیم و سپس اسکریپت مورد نظر را اجرا می کنیم. در آخر نیز حالات مختلف را بررسی می کنیم.

```
void Connect2Py::printProcessStatus()
{
    qDebug() << process->state();

}

void Connect2Py::printProcessError()
{
    qDebug() << process->errorString();
}

void Connect2Py::printProcessOutput()
{
    QString str = process->readAll();

    qDebug() << "str:" << str;

    if(str.indexOf("#f-->") > -1){
//        qDebug() << "send file addres " <<  string2char(fileUrl + "\n");
        process->write(string2char(fileUrl + "\n"));
    }
    else if(str.indexOf("up") > -1){
        emit detectAction("UP");
    }
    else if(str.indexOf("down") > -1){
        emit detectAction("DOWN");
    }
    else if(str.indexOf("right") > -1){
        emit detectAction("RIGHT");
    }
    else if(str.indexOf("left") > -1){
        emit detectAction("LEFT");
    }

}
```

در تابع printProcessStatus وضعیت فرایند ایجاد شده را چاپ می کنیم.

در تابع printProcessError خطای فرایند ایجاد شده را چاپ می کنیم.

در تابع printProcessOutput خروجی را چاپ می کنیم که در آن اسکریپت با علامت #f→ درخواست ورودی می کند که ما حرکت همان لحظه مان که در فایل csv ذخیره شده است را به آن می دهیم و سپس اسکریپت هر مقداری را برگرداند آن مقدار را به کمک سیگنال به سمت qml می فرستیم تا حرکتی متناسب با آن صورت پذیرد.

```
void Connect2Py::setActionFileUrl(QString address)
{
    fileUrl = address;
}

void Connect2Py::setPythonMainScript(QString script)
{
    pythonScript = script;
}

/**
 * @brief Connect2Py::string2char
 *  convert QString to const char*
 * @param str
 * @return
 */
const char *Connect2Py::string2char(QString str)
{

    QByteArray ba = str.toLocal8Bit();
    const char *c_str2 = ba.data();
    return c_str2;
}
```

در تابع setActionFileUrl آدرس فایل csv در متغیری ذخیره می شود.

در تابع setPythonMainScript آدرس اسکریپت در متغیری ذخیره می شود.

و در تابع آخر نیز Qstring را به کاراکتر تبدیل می کنیم.

پیاده سازی فایل پیشبینی کننده حرکت انسان

برای اجرا و تست نهایی به این صورت عمل شده است که ابتدا کد پایتونی در main.py نوشته شده است که هنگام اجرای آن از شما فایل csv می خواهد تا آن را پیشبینی کرده و نتیجه را در ترمینال چاپ کند.

اینکه چطور پیشبینی می کند به صورت زیر است:

در این فایل ابتدا محلی که مدل ذخیره شده است را به آن می دهیم تا مدل را بارگذاری کند.

```
filename = 'C:/Users/Amirhossein/PycharmProjects/bachelor-project/finalized_model.sav'
loaded_model = pickle.load(open(filename, 'rb'))
```

سپس فایل csv داده شده را توسط pandas به دیتافریم تبدیل می کنیم و دو سنسور ژیروسکوپ و شتاب سنج آن را از همه جدا می کنیم.

```
path = input("#f-->")
df = pd.read_csv(path)
gyroscope_df = df.loc[df['ACC_or_GYR'] == 'GYR'].reset_index(drop=True)
accumulator_df = df.loc[df['ACC_or_GYR'] == 'ACC'].reset_index(drop=True)
```

سپس دیتافریم ژیروسکوپ و شتاب سنج را نرمال می کنیم.

```
# normalized gyro
gyro_max = gyroscope_df[['x', 'y', 'z']].max()
df_max = gyro_max.max()
gyroscope_df = gyroscope_df[['x', 'y', 'z']] / df_max

# normalized acc
acc_max = accumulator_df[['x', 'y', 'z']].max()
df_max = acc_max.max()
accumulator_df = accumulator_df[['x', 'y', 'z']] / df_max
```

حال ویژگی های گفته شده را بر روی این ورودی استخراج می کنیم.

```
gyro_x_mean = gyroscope_df['x'].mean()
gyro_y_mean = gyroscope_df['y'].mean()
gyro_z_mean = gyroscope_df['z'].mean()
acc_x_mean = accumulator_df['x'].mean()
acc_y_mean = accumulator_df['y'].mean()
acc_z_mean = accumulator_df['z'].mean()
gyro_x_median = gyroscope_df['x'].median()
gyro_y_median = gyroscope_df['y'].median()
gyro_z_median = gyroscope_df['z'].median()
acc_x_median = accumulator_df['x'].median()
acc_y_median = accumulator_df['y'].median()
acc_z_median = accumulator_df['z'].median()
gyro_x_std = gyroscope_df['x'].std()
gyro_y_std = gyroscope_df['y'].std()
gyro_z_std = gyroscope_df['z'].std()
acc_x_std = accumulator_df['x'].std()
acc_y_std = accumulator_df['y'].std()
acc_z_std = accumulator_df['z'].std()
gyro_x_var = gyroscope_df['x'].var()
gyro_y_var = gyroscope_df['y'].var()
gyro_z_var = gyroscope_df['z'].var()
acc_x_var = accumulator_df['x'].var()
acc_y_var = accumulator_df['y'].var()
acc_z_var = accumulator_df['z'].var()
gyro_x_min = gyroscope_df['x'].min()
gyro_y_min = gyroscope_df['y'].min()
gyro_z_min = gyroscope_df['z'].min()
acc_x_min = accumulator_df['x'].min()
acc_y_min = accumulator_df['y'].min()
acc_z_min = accumulator_df['z'].min()
gyro_x_max = gyroscope_df['x'].max()
gyro_y_max = gyroscope_df['y'].max()
gyro_z_max = gyroscope_df['z'].max()
acc_x_max = accumulator_df['x'].max()
acc_y_max = accumulator_df['y'].max()
acc_z_max = accumulator_df['z'].max()
gyro_x_25 = gyroscope_df['x'].quantile(0.25)
gyro_y_25 = gyroscope_df['y'].quantile(0.25)
gyro_z_25 = gyroscope_df['z'].quantile(0.25)
acc_x_25 = accumulator_df['x'].quantile(0.25)
acc_y_25 = accumulator_df['y'].quantile(0.25)
acc_z_25 = accumulator_df['z'].quantile(0.25)
gyro_x_75 = gyroscope_df['x'].quantile(0.75)
gyro_y_75 = gyroscope_df['y'].quantile(0.75)
gyro_z_75 = gyroscope_df['z'].quantile(0.75)
acc_x_75 = accumulator_df['x'].quantile(0.75)
acc_y_75 = accumulator_df['y'].quantile(0.75)
acc_z_75 = accumulator_df['z'].quantile(0.75)
```

```
gyro_x_kurtosis = gyroscope_df['x'].kurtosis()
gyro_y_kurtosis = gyroscope_df['y'].kurtosis()
gyro_z_kurtosis = gyroscope_df['z'].kurtosis()
acc_x_kurtosis = accumulator_df['x'].kurtosis()
acc_y_kurtosis = accumulator_df['y'].kurtosis()
acc_z_kurtosis = accumulator_df['z'].kurtosis()
gyro_x_skew = gyroscope_df['x'].skew()
gyro_y_skew = gyroscope_df['y'].skew()
gyro_z_skew = gyroscope_df['z'].skew()
acc_x_skew = accumulator_df['x'].skew()
acc_y_skew = accumulator_df['y'].skew()
acc_z_skew = accumulator_df['z'].skew()
```

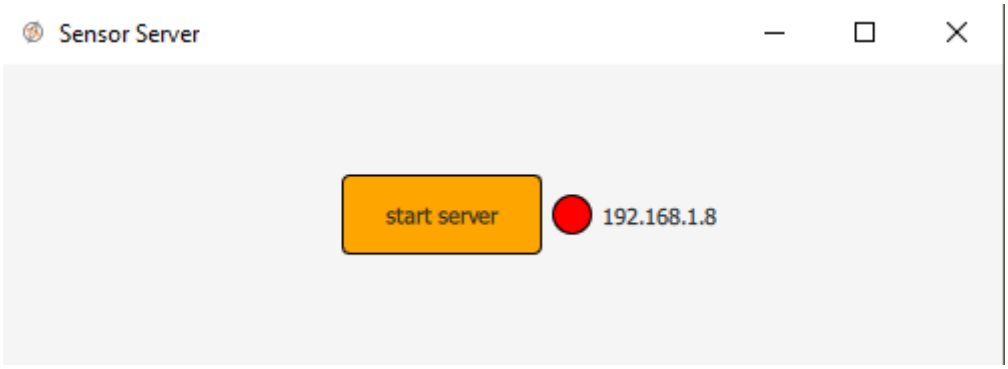
حال که ویژگی ها استخراج شده اند یک دیتافریم جدید با این ویژگی ها ساخته و آن را به پیشبینی کننده مدل svm می دهیم تا آن را پیشبینی کند و در خروجی چاپ کند.

```
new_df = pd.DataFrame([item], columns=column)
predicted = loaded_model.predict(new_df)
print(predicted)
```

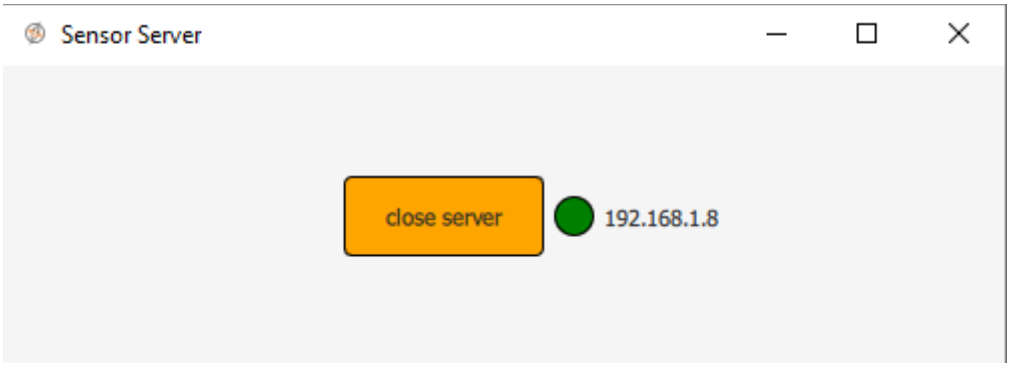
فصل پنجم

جمع آوری مجموعه داده

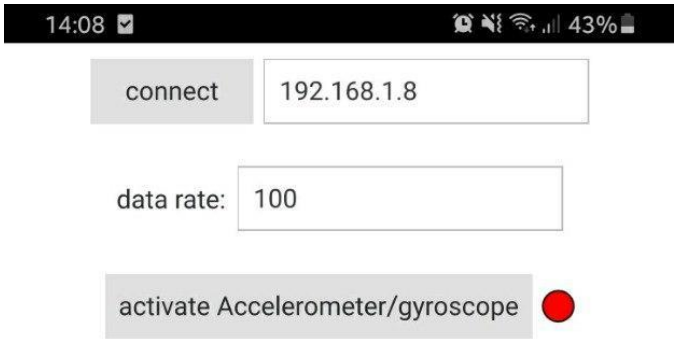
در ابتدا در مورد تهیه مجموعه داده توضیح داده می شود که چگونه جمع آوری شده است.



برنامه بالا برای جمع آوری دیتا توسط آقای براری پیاده سازی شده است که با کلیک بر روی start server سرور فعال می شود. برنامه بعد از فعال شدن سرور به صورت زیر در می آید.

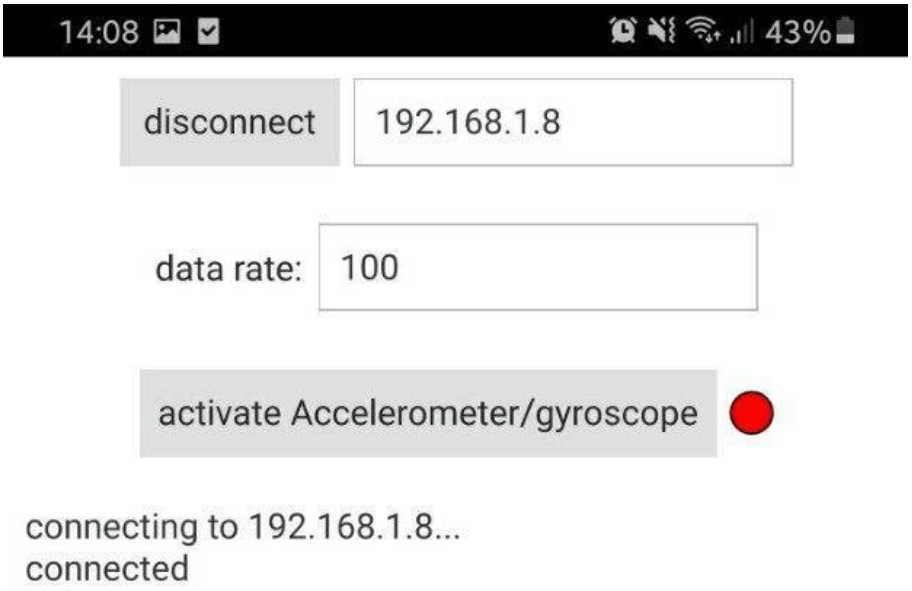


حال به سراغ برنامه موبایل می رویم که توسط اینجانب پیاده سازی شده است.

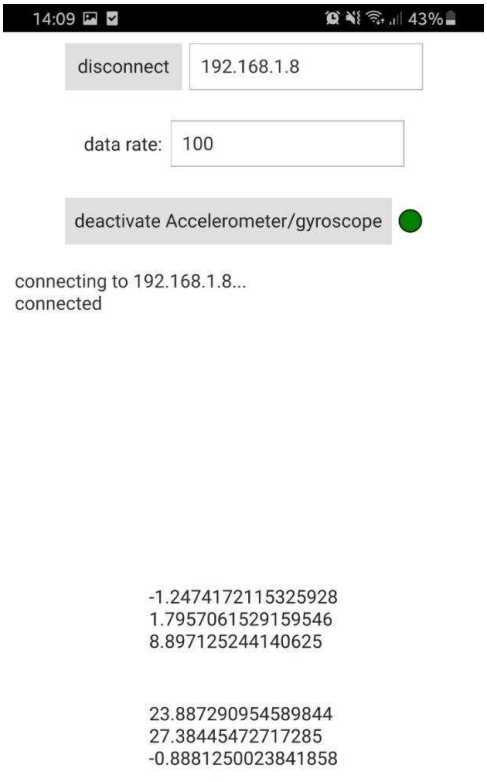


نکته مهم این است که موبایل و کامپیوتر باید به یک شبکه متصل شده باشند. حال ip نمایش داده شده در برنامه کامپیوتر را در برنامه موبایل وارد می کنیم سپس بر روی دکمه اتصال می زنیم تا به سرور متصل

شویم. اگر به سرور متصل شد متن اتصال موفقیت آمیز نمایش داده می شود که در زیر این موضوع را نشان داده ایم.



در قدم بعدی سنسور ها رو فعال می کنیم که همانطور که می بینید سه محور دو سنسور در برنامه نشان داده شده است و در لحظه تغییر می کنند.



به محض روشن شدن سنسور ها، عمل ذخیره سازی شروع می شود. حال حرکت مورد نظر را انجام می دهیم و بر روی دکمه disconnect می زنیم تا حرکت ما در یک فایل csv در محل نصب برنامه دسکتاپ ذخیره شود.

پس به طور کلی روند ذخیره یک حرکت به این صورت است که برنامه دسکتاپ را اجرا می کنیم و سرور را در آن فعال می کنیم. سپس با برنامه موبایل به آن سرور متصل می شویم. بعد از فعال کردن سنسور ها، حرکت مد نظر را انجام می دهیم و بر روی دکمه disconnect کلیک می کنیم تا آن حرکت در یک فایل csv ذخیره شود.

حال قسمتی از یک فایل csv را در زیر می توانید مشاهده کنید که شامل یک حرکت است.

1	ACC_or_G	milisecon	x	y	z
2	ACC	60	0.240019	0.654814	10.15022
3	GYR	62	7.080079	1.586914	-0.91553
4	ACC	70	0.212485	0.661398	10.42615
5	GYR	71	10.62012	0.305176	-1.95313
6	ACC	80	0.249596	0.642245	10.56142
7	GYR	81	13.73291	-1.15967	-2.13623
8	ACC	89	0.246603	0.616507	10.25137
9	GYR	90	17.94434	-2.13623	-2.13623
10	ACC	100	0.203507	0.662595	10.10353
11	GYR	102	21.78955	-0.30518	-2.68555
12	ACC	109	0.129885	0.778116	10.03949
13	GYR	110	22.46094	1.525879	-2.13623
14	ACC	123	0.096965	0.81343	9.966463
15	GYR	124	23.31543	3.845215	-1.15967
16	ACC	129	0.093972	0.920571	9.971251
17	GYR	139	23.55957	5.67627	-0.79346
18	ACC	140	0.110732	0.961272	10.21187
19	GYR	141	24.04785	7.873536	-0.18311
20	ACC	156	0.140061	0.837971	10.33038

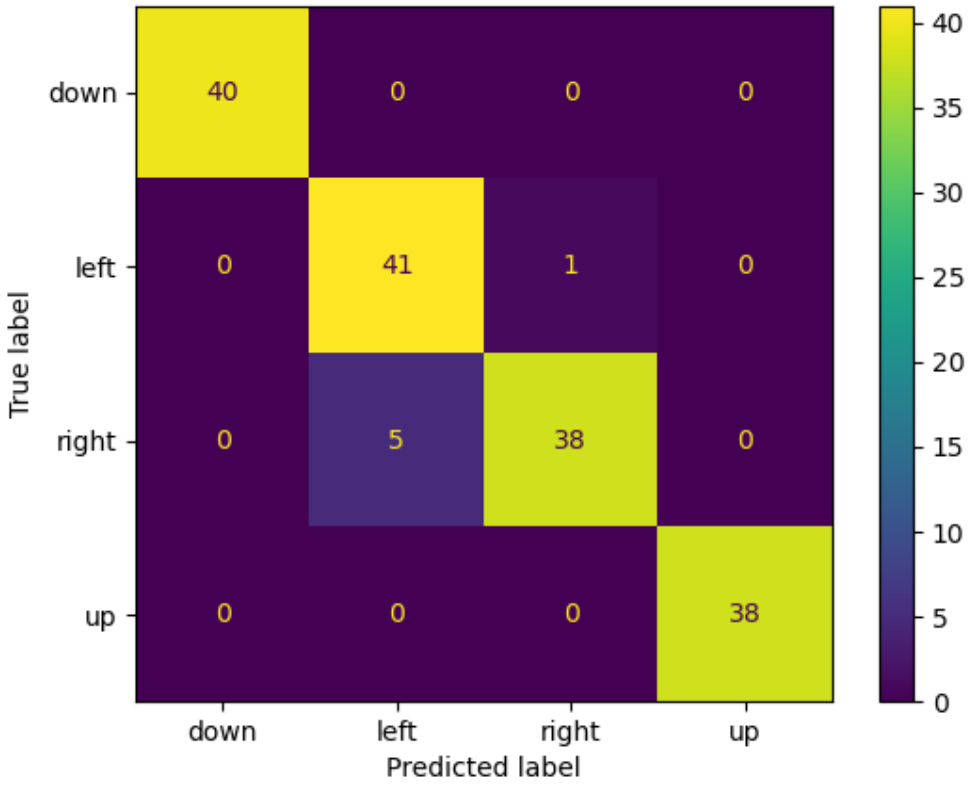
ستون اول بیانگر این است که این مقادیر برای کدام سنسور می باشد. ستون دوم زمان بر حسب میلی ثانیه می باشد. ستون دوم تا چهارم مقادیر سه محور سنسور می باشد.

نتایج و تحلیل آن ها

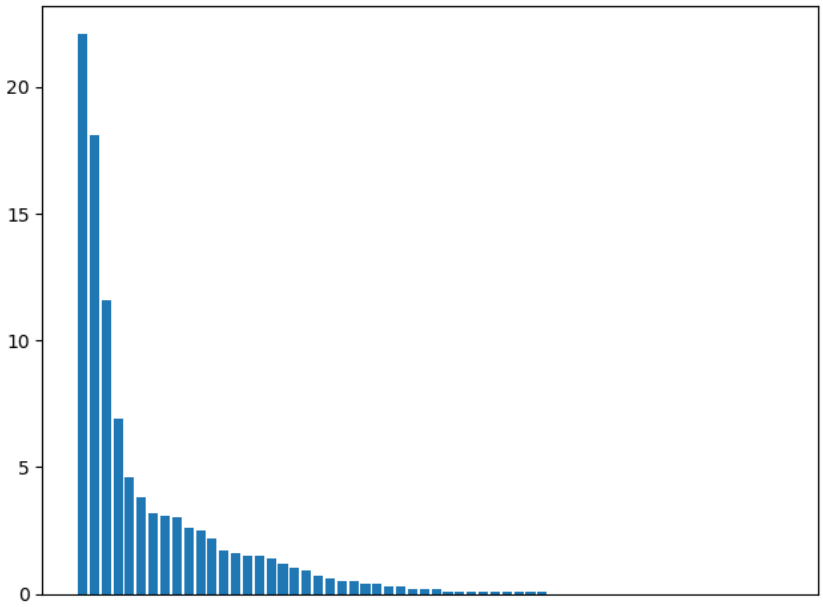
در ابتدا به کمک scikit-learn مجموعه داده را به دو قسمت ترین و تست جدا می کنیم که ۷۰ درصد سهم ترین و ۳۰ درصد سهم تست می باشد.

پس از اینکه مجموعه داده را ترین کردیم به پیشبینی کننده آن مجموعه تست را می دهیم. سپس درصد صحت این پیشبینی را به کمک یک تابع کمکی حساب می کنیم که برابر ۹۶.۵ درصد شد و همچنین سرعت پاسخ گویی نیز مناسب بود.

در زیر نیز می توانید ماتریس درهم ریختگی را ببینید که خطاهای کم آن در تشخیص حرکت چپ و راست می باشد.



همچنین می توانید تاثیر هر ویژگی بر روی مدلمان را در زیر ببینید.



همچنین ترین کردن مجموعه داده بدون دو ویژگی کشیدگی و چولگی نیز انجام شد که درصد صحت به ۹۵.۱ درصد رسید.

فصل ششم

خلاصه:

حال می خواهیم خلاصه ای از پروژه را بازگو کنیم. این پروژه از سه قسمت اصلی یادگیری ماشین، اپلیکیشن موبایل و اپلیکیشن دسکتاپ تشکیل شده است.

روند کلی کار به این صورت بود که ابتدا دو اپلیکیشن موبایل و دسکتاپ برای جمع آوری مجموعه داده نوشته شد. قدم بعدی تهیه مجموعه داده برای چهار حرکت مچ دست به بالا، پایین، چپ و راست بود.

بعد از جمع آوری مجموعه داده نوبت به استخراج ویژگی شد. بعد از بررسی مقالات مختلف به ۱۲ ویژگی رسیدیم که ۱۰ ویژگی برای مجموعه داده ما کفایت کرد. این ویژگی ها شامل میانگین، انحراف معیار، واریانس، بیشینه، کمینه، چولگی، کشیدگی، چارک اول، چارک سوم، دامنه بین چارکی، جذر متوسط مربع بود که دو مورد آخر تاثیر چندانی در دقت پروژه نداشت. ویژگی ها نیز برای هر ۳ محور x، y، z هر دو سنسور محاسبه شد. بعد از محاسبه ویژگی ها، تمامی حرکت ها را با ویژگی هایشان و برچسب خود حرکت درون یک فایل dataset.csv ذخیره کردیم که هر سطر در این فایل بیانگر یک حرکت می باشد.

قدم بعدی کلاس بندی می باشد که با بررسی مقالات مختلف به الگوریتم svm رسیدیم. روند کلاس بندی به این صورت بود که ابتدا فایل dataset.csv را می خوانیم و به کمک ابزار scikit-learn مجموعه داده هایمان را با الگوریتم svm ترین می کنیم. بعد از ترین کردن مجموعه داده به درصد درستی ۹۶.۵ رسیدیم که درصد قابل قبولی است. همچنین مدل خود را نیز در یک فایل ذخیره کردیم تا برای پیشبینی حرکت جدید از آن استفاده کنیم.

در نهایت نیز یک اپلیکیشن دسکتاپ برای تشخیص حرکت پیاده سازی شد که ابتدا حرکت مورد نظر را به کمک برنامه موبایل انجام می دهیم. پس از اتمام حرکت، برنامه کامپیوتر حرکت را به صورت فایل csv ذخیره می کند و آن را به مدلمان می دهد. مدل پس از تشخیص حرکت، حرکت انجام شده را به برنامه کامپیوتر می گوید و برنامه کامپیوتر انیمیشن مربوط به آن حرکت را اجرا می کند.

بحث:

نقاط قوت پروژه

- درصد درستی ۹۶.۵٪ که قابل قبول می باشد و حرکات را به خوبی تشخیص می دهد
- پیاده سازی برنامه ها بر پایه c++ که یکی از سریع ترین و مقیاس پذیرترین زبان ها می باشد
- کراس پلتفورم بودن برنامه موبایل

نقاط ضعف پروژه

- حرکات محدودی مورد قبول است.
- عدم تشخیص دنباله ای از حرکات
- ظاهر ساده برنامه ها

نتیجه‌گیری:

تلفن های همراه دستگاه های فشرده و در عین حال قدرتمندی هستند که توانایی انجام کارهای بی شماری را دارند. در این پروژه از حسگرهای ژیروسکوپ و شتاب سنج برای شناسایی فعالیت های انجام شده توسط یک شخص استفاده شده است. برای بحث در مورد نتایج طبقه بندی، ترکیب سنسورها بهترین نتیجه را دارد. مطلوب ترین الگوریتم های یادگیری ماشین برای شناسایی فعالیت RF ، svm ، knn هستند. نتیجه این تحقیق پتانسیل کاربردی را دارد که می تواند برای تشخیص فعالیت روزمره انسان مورد استفاده قرار گیرد.

کارهای آینده:

کار های آینده ای که قرار است صورت گیرد به شرح زیر است:

- افزایش تعداد حرکت قابل پیشبینی
- توانایی تشخیص و جداسازی دنباله ای از حرکات
- پیاده سازی اپلیکیشن های کاربردی
- بهینه سازی بیشتر ویژگی ها

مراجع

[1] P. K. Roy, H. Om, Suspicious and violent activity detection of humans using hog features and svm classifier in surveillance videos, in: Advances in Soft Computing and Machine Learning in Image Processing, Springer, 2018, pp. 277–294

[2] A. Thyagarajmurthy, M. Ninad, B. Rakesh, S. Niranjan, B. Manvi, Anomaly detection in surveillance video using pose estimation, in: Emerging Research in Electronics, Computer Science and Technology, Springer, 2019, pp. 753–766.

[3] M. Ehatisham-Ul-Haq, A. Javed, M. A. Azam, H. M. Malik, A. Irtaza, I. H. Lee, M. T. Mahmood, Robust human activity recognition using multimodal feature-level fusion, IEEE Access 7 (2019) 60736–60751.

[4] E. Zdravevski, P. Lameski, V. Trajkovik, A. Kulakov, I. Chorbev, R. Goleva, N. Pombo, N. Garcia, Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering, Ieee Access 5 (2017) 5262–5280.

[5] T. Billah, S. M. Rahman, M. O. Ahmad, M. Swamy, Recognizing distractions for assistive driving by tracking body parts, IEEE Transactions on Circuits and Systems for Video Technology 29 (4) (2018) 1048–1062.

[6] L. Mart´ınez-Villase˜nor, H. Ponce, A concise review on sensor signal acquisition and transformation applied to human activity recognition and human–robot interaction, International Journal of Distributed Sensor Networks 15 (6) (2019) 1550147719853987.

[7] R. Mojarad, F. Attal, A. Chibani, S. R. Fiorini, Y. Amirat, Hybrid approach for human activity recognition by ubiquitous robots, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 5660– 5665.

[8] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz1,“A Public Domain Dataset for Human Activity Recognition Using Smartphone’s,” ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013.

[9] Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore, “Activity Recognition using Cell Phone Accelerometers,” ACM SIGKDD Explorations Newsletter Volume 12 Issue 2, Pages 74-82, December 2010.

[10] B. Jagadeesh, C. M. Patil, Video based human activity detection, recognition and classification of actions using svm, Transactions on Machine Learning and Artificial Intelligence 6 (6) (2019) .

[11]. Han J., Shao L., Xu D., Shotton J. Enhanced computer vision with microsoft kinect sensor: A review. IEEE Trans. Cybern. 2013;43:1318–1334

[12] .Junker H., Amft O., Lukowicz P., Tröster G. Gesture spotting with body-worn inertial sensors to detect user activities. Pattern Recogn. 2008;41:2010–2024. doi: 10.1016/j.patcog.2007.11.016.

[13] .Shoaib M., Bosch S., Incel O.D., Scholten H., Havinga P.J.M. Complex human activity recognition using smartphone and wrist-worn motion sensors. Sensors. 2016;16:426. doi: 10.3390/s16040426.

[14] .Kwapisz J.R., Weiss G.M., Moore S.A. Activity recognition using cell phone accelerometers. SIGKDD Explor. Newsl. 2011;12:74–82. doi: 10.1145/1964897.1964918.

[15]. Su X., Tong H., Ji P. Activity recognition with smartphone sensors. Tsinghua Sci. Technol. 2014;19:235–249. doi: 10.1109/TST.2014.6838194.

[16] .De D., Bharti P., Das S.K., Chellappan S. Multimodal wearable sensing for fine-grained activity recognition in healthcare. IEEE Internet Comput. 2015;19:26–35. doi: 10.1109/MIC.2015.72.

[17] .Paul S.S., Tiedemann A., Hassett L.M., Ramsay E., Kirkham C., Chagpar S., Sherrington C. Validity of the Fitbit activity tracker for measuring steps in community-dwelling older adults. BMJ Open Sport Exerc. Med. 2015;1 doi: 10.1136/bmjsem-2015-000013.

[18] .Fritz T., Huang E.M., Murphy G.C., Zimmermann T. Persuasive technology in the real world: A study of long-term use of activity sensing devices for fitness; Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI ’14); Toronto, ON, Canada. 26 April–1 May 2014; New York, NY, USA: ACM; 2014. pp. 487–496.

[19] .Boas Y.A.G.V. Overview of virtual reality technologies; Proceedings of the Interactive Multimedia Conference; Bogotá, Colombia. 28–30 August 2013

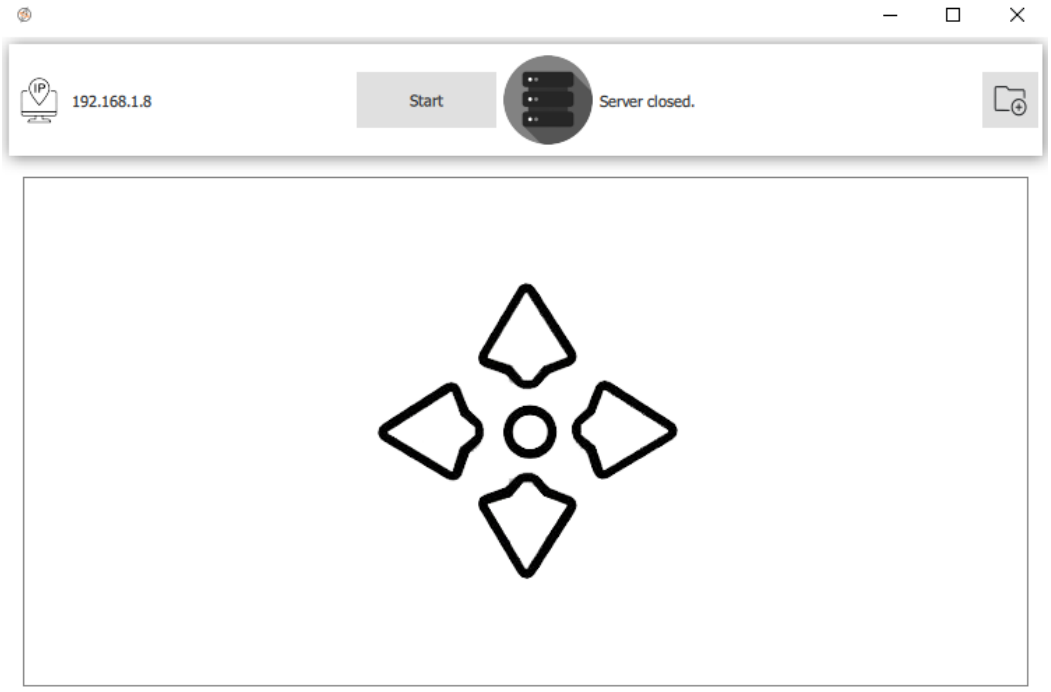
[20] .Deutsch J.E., Brettler A., Smith C., Welsh J., John R., Guarrera-Bowlby P., Kafri M. Nintendo Wii sports and Wii fit game analysis, validation, and application to stroke rehabilitation. Top. Stroke Rehabil. 2011;18:701–719. doi: 10.1310/tsr1806-701.

[21] Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore, “Activity Recognition using Cell Phone Accelerometers,” ACM SIGKDD Explorations Newsletter Volume 12 Issue 2, Pages 74-82, December 2010.

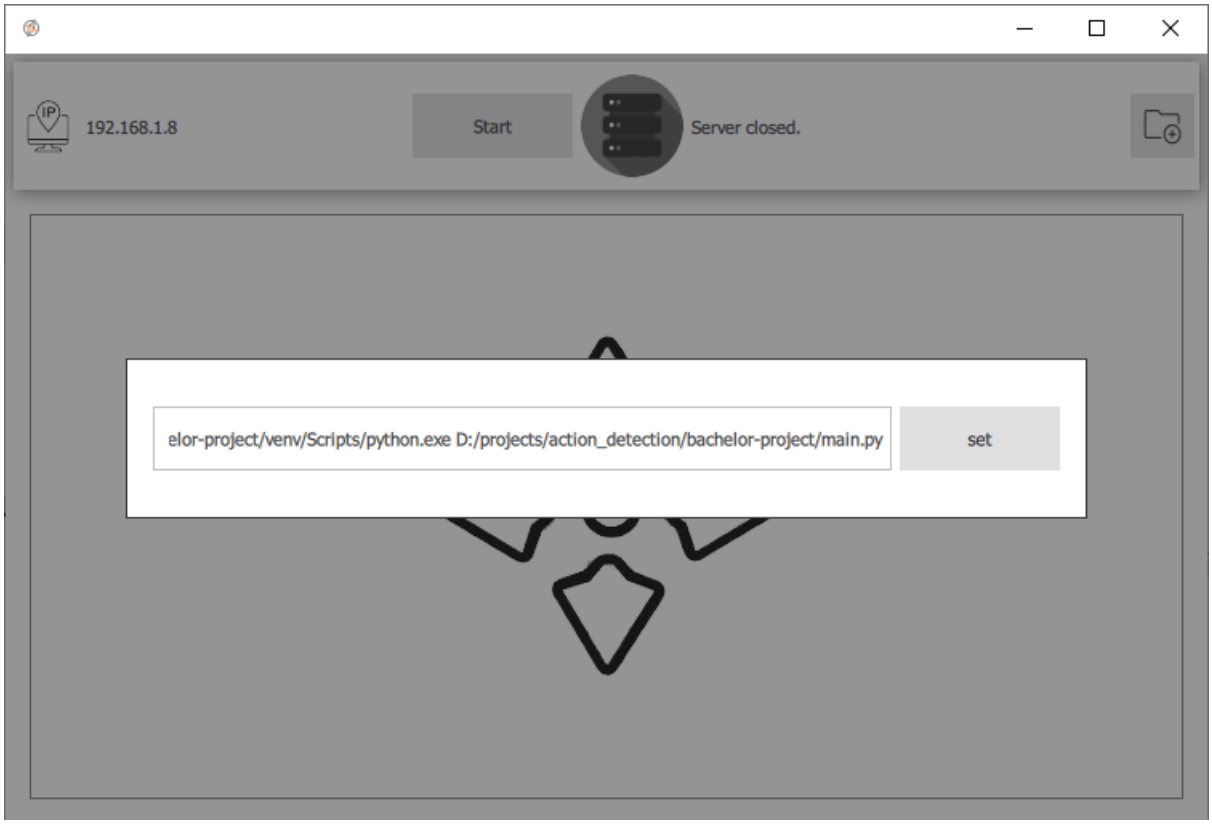
[22] Weiss, G. M., and Hirsh, H. Learning to predict rare events in event sequences, In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA, 359-363, 1998

پیوست

ابتدا برنامه تشخیص حرکت را اجرا می کنیم.

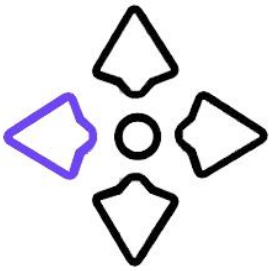


بعد از اجرا بر روی پوشه سمت راست برنامه کلیک کرده و آدرس فایل پایتون محیط مجازی و آدرس فایل main.py را به آن می دهیم.



سپس سرور را روشن می کنیم که امکان اتصال برنامه موبایل به آن وجود داشته باشد.

در برنامه موبایل ip نمایش داده شده در سمت را در آن وارد کرده و سپس دکمه اتصال را می زنیم تا برنامه گوشی به برنامه کامپیوتر متصل شود. پس از اتصال، دکمه فعال سازی سنسور ها را زده، حرکت مد نظر رو خود را انجام می دهید و پس از اتمام حرکت بر روی دکمه قطع اتصال می زنیم. هنگامی که اتصال قطع می شود برنامه دسکتاپ تمامی ورودی که تا آن لحظه گرفته است را درون یک فایل csv ذخیره می کند و سپس فایل main.py را اجرا کرده و به ورودی آن فایل csv را می دهد. فایل main.py نیز پس از پردازش، خروجی مد نظر را چاپ کرده و برنامه دسکتاپ متناسب با خروجی چاپ شده، انیمیشنی را به سمت جهت چاپ شده، نمایش می دهد.



نحوه انجام حرکات نیز در پوشه video گذاشته شده است.

University of Tehran
College of Farabi
Faculty of Engineering
Department of Computer Engineering

Mobile Application Development for an Activity Recognition System Based on Mobile Gyroscope Sensor

By:

Amirhossein Asadi

Under Supervision of:

Dr. Kazim Fouladi

A Project Report as a Requirement for the Degree of Bachelor of Science in Computer Engineering

September 2020