



دانشگاه خوارزمی

دانشکده علوم ریاضی و کامپیوتر

گروه علوم کامپیوتر

گزارش نهایی پروژه کارشناسی

# مقایسه روش‌های سنتی یادگیری ماشین با یادگیری عمیق در تشخیص سقوط در سالمندان

نگارش:

محمدعلی سفیدی اصفهانی

استاد راهنما:

دکتر محمد سلطانیان

(نیمسال دوم 1399)

(تیر 1400)

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

## تقدیم به پدر و مادر مهربانم

این اولین پروژه‌ای است که جدی روی آن زمان گذاشتم و روی آن کار کردم. این پروژه را به پدر و مادر مهربانم تقدیم می‌کنم و امیدوارم در مراحل بعدی بیشتر از این‌ها خوشحالشان کنم.

## تقدیر

از همه کسانی که برای انجام پروژه من را همراهی کردند به‌ویژه استاد سلطانیان تشکر و قدردانی می‌کنم.

## چکیده گزارش

یکی از نیازهایی که کمبود آن در هر جامعه‌ای حس می‌شود، بحث نگهداری از افراد سالمند است. با توجه سن این افراد و کاهش قوای جسمانی آن‌ها، آسیب‌هایی در کمین افراد سالمند است چندین برابر افراد معمولی است و یکی از اصلی‌ترین خطراتی این افراد را تهدید می‌کند خطر افتادن یا سقوط است.

هدف اصلی این پروژه طراحی و پیاده‌سازی و البته مقایسه مدل‌های یادگیری ماشین است که باتوجه به پارامترهایی که از سنسورها دریافت می‌کند، بتواند سقوط را پیش‌بینی کند. برای پیاده‌سازی مدل‌ها از مجموعه داده SisFall استفاده شده که شامل اطلاعات مربوط به دو سنسور شتاب‌سنج و یک سنسور ژيروسکوپ می‌باشد.

به‌طور کلی در این پروژه مدل‌های شبکه عصبی عمیق، SVM و Logistic Regression با استفاده از شاخص‌های آماری از جنبه‌های مختلف مقایسه شدند. نکته مهمی که برای پیاده‌سازی این مدل‌ها در نظر گرفته شده بحث پنجره‌بندی و متعادل‌سازی داده‌ها است که تأثیر چشم‌گیری در عملکرد مدل‌ها داشته است.

نکته حائز اهمیت دیگری که تأثیر زیادی در بهبود عملکرد و بهینه بودن مدل‌های این پروژه شده، کارکردن با آدرس فایل‌های مجموعه داده‌ها به جای داده‌های درون فایل و همین‌طور کارکردن با numpy array به جای dataframe ها است که سرعت پیش‌پردازش و آماده‌سازی داده‌ها را برای یادگیری مدل را نسبت به روش‌های دیگر تا حد بسیار زیادی بالا برده است.

در نهایت بعد از انجام پروژه به این نتیجه رسیدیم که باتوجه به شاخص‌های آماری شبکه عصبی عمیق، مدل مطمئن‌تری برای تشخیص سقوط می‌باشد. البته که از بین روش‌ها سنتی نیز Logistic Regression نیز عملکرد خوبی مخصوصاً در زمینه پیش‌بینی سقوط دارد.

کلیدواژه: تشخیص سقوط، شبکه عصبی عمیق، Logistic Regression، معیار نمره F1، روش بردار پشتیبان، پیش‌پردازش داده‌ها، مجموعه داده SisFall.

فصل ۱: مقدمه .....	۱
۱-۱- انگیزه پژوهش .....	۲
۱-۲- مروری بر پیشینه و کارهای مشابه .....	۲
۱-۳- هدف و مهم‌ترین دستاوردها .....	۳
۱-۴- خلاصه مطالب بیان‌شده در این گزارش .....	۳
فصل ۲: تشخیص خودکار سقوط .....	۵
۲-۱- روش‌های موجود در زمینه تشخیص سقوط .....	۶
۲-۱-۱- روش‌های سنتی برای تشخیص سقوط .....	۶
۲-۱-۲- استفاده از یادگیری عمیق برای تشخیص سقوط .....	۶
۲-۱-۳- استفاده از الگوریتم‌های هوش مصنوعی برای تشخیص سقوط .....	۷
۲-۲- آشنایی با مجموعه داده Sisfall .....	۷
۲-۲-۱- معرفی کلی .....	۷
۲-۲-۲- آشنایی با محتویات پایگاه‌داده .....	۷
۲-۲-۳- سنسورهای استفاده‌شده در مجموعه داده .....	۱۰
۲-۲-۴- آشنایی با جزئیات بیشتری از مجموعه داده .....	۱۰
۲-۳- جمع‌بندی .....	۱۲
فصل ۳: روش پیشنهادی .....	۱۳
۳-۱- مقدمه .....	۱۴
۳-۲- مجموعه داده .....	۱۴
۳-۳- الگوریتم پیشنهادی .....	۱۵
۳-۴- پلتفرم کدنویسی و کتابخانه‌های مورداستفاده .....	۱۵
۳-۵- پیاده‌سازی روش پیشنهادی .....	۱۶
۳-۵-۱- کتابخانه‌ها و framework های موردنیاز .....	۱۶
۳-۵-۲- دریافت دیتاست اصلی و مکمل .....	۱۷
۳-۵-۳- استخراج آدرس همه فایل‌های داخل دیتاست .....	۱۷
۳-۵-۴- استخراج اطلاعات موردنیاز از هر فایل داخل دیتاست .....	۱۸
۳-۵-۵- استخراج برچسب‌های مربوط به هر فایل .....	۱۹

۳-۵-۶	استخراج ویژگی‌های مدل از هر فایل.....	۲۰
۳-۵-۷	تقسیم‌بندی آدرس‌ها به سه دسته آموزش، ارزیابی و آزمون.....	۲۰
۳-۵-۸	تبدیل آدرس‌ها به ویژگی‌های موردنیازمان.....	۲۱
۳-۵-۹	پنجره‌بندی کردن ویژگی‌ها.....	۲۲
۳-۵-۱۰	تبدیل کردن numpy array به tensor.....	۲۳
۳-۵-۱۱	آماده‌سازی داده‌ها برای مدل‌سازی.....	۲۳
۳-۶	مدل‌سازی و نتایج شبیه‌سازی.....	۲۴
۳-۶-۱	شبکه عصبی به همراه validation.....	۲۵
۳-۶-۲	شبکه عصبی بدون validation.....	۲۷
۳-۶-۳	مدل logistic Regression.....	۲۸
۳-۶-۴	مدل SVM.....	۲۹
۳-۶-۵	نتیجه‌گیری و یک ایده جدید برای بهبود عملکرد.....	۳۰
۳-۶-۶	متعادل کردن مجموعه آموزش.....	۳۱
۳-۶-۷	شبکه عصبی بعد از متعادل شدن مجموعه آموزش.....	۳۱
۳-۶-۸	مدل logistic Regression بعد از متعادل شدن مجموعه آموزش.....	۳۳
۳-۶-۹	مدل SVM بعد از متعادل شدن مجموعه آموزش.....	۳۴
۳-۷	مقایسه و نتیجه‌گیری.....	۳۴
فصل ۴:	جمع‌بندی و پیشنهادها.....	۳۶
۴-۱	نتیجه‌گیری.....	۳۷
۴-۲	پیشنهادهایی برای کارهای آتی.....	۳۸

## فهرست شکل‌ها

شکل ۲-۱. عملکرد سنسورها	۱۱
شکل ۲-۲. فرایند سقوط	۱۲
شکل ۳-۱	۱۶
شکل ۳-۲	۱۷
شکل ۳-۳	۱۷
شکل ۳-۴	۱۸
شکل ۳-۵	۱۸
شکل ۳-۶	۱۹
شکل ۳-۷	۲۰
شکل ۳-۸	۲۰
شکل ۳-۹	۲۱
شکل ۳-۱۰	۲۲
شکل ۳-۱۱	۲۳
شکل ۳-۱۲	۲۳
شکل ۳-۱۳	۲۴
شکل ۳-۱۴	۲۴
شکل ۳-۱۵	۲۵
شکل ۳-۱۶	۲۶
شکل ۳-۱۷	۲۶
شکل ۳-۱۸	۲۶
شکل ۳-۱۹	۲۷
شکل ۳-۲۰	۲۷
شکل ۳-۲۱	۲۸
شکل ۳-۲۲	۲۸
شکل ۳-۲۳	۲۹
شکل ۳-۲۴	۲۹
شکل ۳-۲۵	۳۰
شکل ۳-۲۶	۳۱
شکل ۳-۲۷	۳۲
شکل ۳-۲۸	۳۲
شکل ۳-۲۹	۳۲
شکل ۳-۳۰	۳۳



شکل ۳-۳۱..... ۳۴

شکل ۳-۳۲..... ۳۵

شکل ۳-۳۳..... ۳۵

شکل ۴-۱..... ۳۵

## فهرست جدول‌ها

جدول ۲-۱. فعالیتهای معمول افراد	۸
جدول ۲-۲. افتادن در شرایط مختلف	۸
جدول ۲-۳. ویژگی افراد مشارکت‌کننده	۹
جدول ۳-۱. عملکرد الگوریتم‌ها قبل از متعادل‌سازی	۳۱
جدول ۳-۲. عملکرد الگوریتم‌ها بعد از متعادل‌سازی	۳۴

# فصل ۱: مقدمه

## ۱-۱- انگیزه پژوهش

طی سال‌های گذشته، تحقیقات زیادی در زمینه تشخیص رفتار انسان‌ها و مخصوصاً افرادی که به کمک و نگهداری نیاز دارند شده است. در حوزه سنسورها و همین‌طور اینترنت اشیا نیز یکی از مهم‌ترین بخش فعالیت‌ها به دستیارهای خانگی و البته بهتر کردن وضعیت زندگی افراد برمی‌گردد. در کنار این موضوع گزارش‌های زیادی ناشی از آسیب دیدن افراد سالمند در خانه خودشان رسیده است که با افزایش میانگین سن جمعیت، تعداد این گزارش‌ها بیشتر و بیشتر می‌شوند. علاوه بر این افزایش میانگین سن افراد یک جامعه باعث افزایش هزینه‌های نگهداری و هزینه‌های پزشکی نیز می‌شود.

طبق آمارهای رسمی منتشر شده در کشور آمریکا بیش از 40 درصد افراد بالای 75 سال حداقل یک‌بار در سال این اتفاق برای آنها می‌افتد و با توجه به ضعف‌های جسمی، روحی و البته بالا بودن زمان ریکاوری بدن این افراد، وجود یک سیستم مراقبتی برای این افراد مسن کاملاً حس می‌شود.

## ۱-۲- مروری بر پیشینه و کارهای مشابه

سیستم تشخیص سقوط وظیفه دارد که در مرحله اول با استفاده از سیستم‌های هشداردهنده از سقوط افراد جلوگیری گیرد و قبل از سقوط همراهان و افرادی که نزدیک فرد سالخورده هستند را مطلع کنند. وظیفه دوم این سیستم کاهش صدمات ناشی از سقوط می‌باشد که بسته به نحوه پیاده‌سازی آن‌ها چگونگی حفاظت از افراد متفاوت است. [5]

به‌طور کلی سیستم‌های تشخیص سقوط به سه دسته تقسیم‌بندی می‌شوند: گجت‌های پوشیدنی، سنسورهای محیطی و در نهایت سیستم‌هایی که تحت computer-vision پیاده‌سازی می‌شوند.

گجت‌های پوشیدنی شامل ساعت‌ها و تلفن‌های هوشمند می‌شوند که با استفاده از سنسورهای موجود در این دستگاه‌ها سقوط را تشخیص می‌دهند. سنسورهای محیطی نیز در محیط‌هایی که افراد سالخورده یا افراد تحت مراقبت بیشتر وقتشان را می‌گذرانند نصب می‌شوند و در صورتی که سنسورها پارامترهای غیرمعمول دریافت کنند، سیستم هشداردهنده فعال می‌شود. [4]

امروزه با توجه به این که اکثر افراد از ساعت‌ها و تلفن‌های هوشمند استفاده می‌کنند اکثر فعالیت‌های

انجام شده در این حوزه قرار دارد. البته که از سنسورهای محیطی نیز در بیمارستان‌ها یا آسایشگاه‌ها و حتی برای افرادی که فقط در خانه نگهداری می‌شوند، استفاده می‌شوند. به‌طور کلی این سیستم‌ها وظیفه دارند سقوط را چند ثانیه قبل‌تر از وقوع آن پیش‌بینی کنند و در صورت وقوع آن علاوه بر مطلع کردن افراد نزدیک به صحنه وقوع، بیمارستان و نزدیکان فرد را سریعاً مطلع کند تا صدمات به کمترین حد خود برسد.

### ۳-۱- هدف و مهم‌ترین دستاوردها

هدف اصلی این پروژه تشخیص هرچه بهتر مسئله سقوط توسط الگوریتم‌های مختلف یادگیری ماشین و در نهایت مقایسه این الگوریتم‌ها باهم دیگر است. مهم‌ترین مسائلی که در پیاده‌سازی این پروژه به آن باید توجه شود، چگونگی پنجره‌بندی بهینه داده‌ها و البته نحوه متعادل کردن مجموعه داده‌ها می‌باشد.

### ۴-۱- خلاصه مطالب بیان شده در این گزارش

در این گزارش ابتدا در فصل بعدی به معرفی کلی پروژه Fall Detection و مفاهیم اولیه‌ای که در ابتدای مسیر به آن‌ها نیاز داریم می‌پردازیم. در نهایت در انتهای این فصل با استفاده از نمودارهای مختلف بیشتر با مجموعه داده‌ها آشنا می‌شویم.

در فصل سوم به نحوه‌ی پیاده‌سازی الگوریتم سقوط و البته مقایسه‌های روش‌های سنتی و مدرن یادگیری ماشین و میزان دقت آن‌ها پرداخته می‌شود. ابتدا توابع و method هایی که برای آماده‌سازی داده و پیش‌پردازش داده‌ها استفاده شدند آشنا می‌شویم. همین‌طور وظایف و نحوه پیاده‌سازی هرکدام از این توابع بررسی می‌شود. سپس سراغ نحوه آماده‌سازی داده‌ها و پیاده‌سازی روش‌های یادگیری ماشین می‌پردازیم و نتایج و دقت و عملکرد هریک از این مدل‌ها با شاخص‌های precision و f1-score و recall سنجیده می‌شود.

در این بخش ابتدا دو شبکه عصبی چندلایه متفاوت و در شرایط مختلف پیاده‌سازی می‌شود و عملکرد آن بررسی می‌شود. سپس دو مدل SVM و Logistic Regression به‌عنوان مدل‌های سنتی

یادگیری ماشین پیاده‌سازی می‌شوند و عملکرد آن‌ها بررسی می‌شود. برای بهبود عملکرد این مدل‌ها، در بخش بعدی فصل سوم، متعادل‌سازی مجموعه داده انجام می‌شود تا با بهبود وضعیت مجموعه داده عملکرد مدل‌ها این بهبود پیدا کند. بعد از انجام این کار در آخرین بخش این فصل عملکرد مدل‌ها با استفاده از شاخص‌های مختلف مقایسه می‌شوند و یک نتیجه‌گیری از شواهد موجود ارائه می‌شود. در فصل چهارم نیز خلاصه‌ای فعالیت‌های ارائه شده و نتایج به دست آمده و همین‌طور کارهایی انتظار می‌رود عملکرد این مدل‌ها را در آینده بهتر کند، ارائه می‌شود. در آخرین فصل این گزارش نیز مراجع استفاده شده بیان شده است.

## فصل ۲:

# تشخیص خودکار سقوط

## ۲-۱- روش‌های موجود در زمینه تشخیص سقوط

به‌طور کلی از انواع روش‌های یادگیری ماشین برای پیاده‌سازی این پروژه استفاده می‌شود. همین‌طور در مورد مجموعه داده نیز عده‌ای از پایگاه داده‌های از قبل تهیه شده استفاده می‌کنند و عده‌ای هم شخصاً مجموعه داده جدیدی تهیه و از آن استفاده می‌کنند. به‌طور کلی این پروژه را می‌توان با روش‌های سنتی یادگیری ماشین، الگوریتم‌های مختلف هوش مصنوعی و همین‌طور از روش‌های یادگیری عمیق پیاده‌سازی کرد.

### ۲-۱-۱- روش‌های سنتی برای تشخیص سقوط

باتوجه به این‌که در نهایت ما به دنبال طبقه‌بندی و پیش‌بینی سقوط یا عدم سقوط هستیم، برای پیاده‌سازی این پروژه می‌توان از الگوریتم‌های Logistic Regression ، KNN ، SVM و حتی Random Forest نیز استفاده کرد. من به‌شخصه از Logistic Regression و SVM به‌عنوان روش‌های سنتی یادگیری ماشین استفاده کرده‌ام.

### ۲-۱-۲- استفاده از یادگیری عمیق برای تشخیص سقوط

برای پیاده‌سازی مسئله سقوط می‌توان از انواع مدل‌های شبکه‌های عصبی مثل شبکه‌های عصبی چندلایه ساده و پیچشی استفاده کرد. این شبکه‌های عصبی با استفاده از مجموعه داده‌ای که به‌عنوان ورودی دریافت می‌کنند، آموزش می‌بینند و در صورتی که مجموعه داده با کیفیتی در اختیار آن‌ها قرار بگیرد، عملکرد خوبی خواهند داشت.



### ۳-۱-۲- استفاده از الگوریتم‌های هوش مصنوعی برای تشخیص سقوط

Computer vision یکی از اصلی‌ترین الگوریتم‌های هوش مصنوعی است که در زمینه تشخیص سقوط استفاده می‌شود. در این روش کامپیوتر به کمک روش‌های مختلف هوش مصنوعی دنیای واقعی آشنا می‌شود و می‌تواند به خوبی در مقابل چیزهایی که می‌بیند واکنش داشته باشد. محققان در زمینه تشخیص سقوط از این روش کمترین بهره را برده‌اند.

### ۲-۲- آشنایی با مجموعه داده Sisfall

در این بخش از گزارش قصد دارم بیشتر درباره مجموعه داده مورد استفاده برای پیاده‌سازی این پروژه صحبت کنیم و از جوانب مختلف آن را بررسی کنیم.

#### ۱-۲-۲- معرفی کلی

مجموعه داده SisFall: A Fall and Movement Dataset در سال 2016 توسط دانشکده مهندسی دانشگاه Universidad de Antiquia تهیه شده و در اختیار همه افراد قرار گرفته است. این پایگاه داده شامل 4510 فایل می‌باشد که هر یک از این فایل‌ها مربوط به یک فعالیت مجزا است. نکته دیگری که در تهیه این مجموعه داده در نظر گرفته شده این است که در این پایگاه داده افراد سالخورده و همین‌طور افراد جوان نیز وجود دارد. در ادامه به جزئیات بیشتری می‌پردازیم.

#### ۲-۲-۲- آشنایی با محتویات پایگاه داده

برای تهیه این دیتاست 38 نفر فعالیت‌های مختلفی را انجام دادند که اطلاعات مربوط به این فعالیت‌ها و همین‌طور اطلاعات فردی این افراد در جدول‌های بعدی آمده است.

جدول ۱-۲. فعالیت‌های معمول افراد

مدت‌زمان (ثانیه)	فعالیت انجام شده	کد فایل
100	آهسته راه رفتن	D01
100	تند راه رفتن	D02
100	نرم دویدن	D03
100	دویدن با سرعت زیاد	D04
25	بالا و پایین رفتن از پله‌ها با سرعت کم	D05
25	بالا و پایین رفتن از پله‌ها با سرعت زیاد	D06
12	آرام نشستن روی صندلی، صبر کردن و آرام بلند شدن	D07
12	سریع نشستن روی صندلی، صبر کردن و سریع بلند شدن	D08
12	آرام نشستن روی صندلی کوتاه، صبر کردن و آرام بلند شدن	D09
12	سریع نشستن روی صندلی کوتاه، صبر کردن و سریع بلند شدن	D10
12	نشستن، تلاش برای بلند شدن و دوباره نش	D11
12	نشستن، آرام درازکشیدن صبر کردن و دوباره نشستن	D12
12	نشستن، سریع درازکشیدن، صبر کردن و دوباره نشستن	D13
12	درازکشیدن روی یک شانه و سپس درازکشیدن روی شانه دیگر	D14
12	ایستادن، آرام نشستن روی زانو و دوباره ایستادن	D15
12	ایستادن، نشستن و دوباره ایستادن	D16
12	ایستادن، سوارشدن توی ماشین و دوباره بیرون آمدن	D17
12	تلوتلو خوردن موقع راه رفتن	D18
12	با تمام قدرت پریدن بدون افتادن	D19

جدول ۲-۲. افتادن در شرایط مختلف

مدت‌زمان (ثانیه)	فعالیت انجام شده	کد فایل
15	افتادن از جلو موقع راه رفتن بر اثر لیز خوردن	F01
15	افتادن از عقب موقع راه رفتن بر اثر لیز خوردن	F02
15	افتادن از کنار موقع راه رفتن بر اثر لیز خوردن	F03
15	افتادن از جلو موقع راه رفتن بر اثر لغزش	F04
15	افتادن از جلو موقع دویدن بر اثر لغزش	F05
15	غش کردن موقع راه رفتن	F06
15	غش کردن موقع راه رفتن و کمک گرفتن از دست‌ها	F07
15	افتادن از جلو موقع تلاش کردن برای بلند شدن	F08
15	افتادن از کنار موقع تلاش کردن برای بلند شدن	F09

F10	افتادن از جلو موقع تلاش کردن برای نشستن	15
F11	افتادن از عقب موقع تلاش کردن برای نشستن	15
F12	افتادن از کنار موقع تلاش کردن برای نشستن	15
F13	افتادن از جلو موقع نشستن بر اثر خوابیدن یا غش کردن	15
F14	افتادن از عقب موقع نشستن بر اثر خوابیدن یا غش کردن	15
F15	افتادن از کنار موقع نشستن بر اثر خوابیدن یا غش کردن	15

جدول ۳-۲. ویژگی افراد مشارکت‌کننده

کد فرد	سن	قد	وزن	جنسیت
SA01	26	165	53	زن
SA02	23	176	58	مرد
SA03	19	156	48	زن
SA04	23	170	72	مرد
SA05	22	172	70	مرد
SA06	21	169	58	مرد
SA07	21	156	63	زن
SA08	21	149	41	زن
SA09	24	165	64	مرد
SA10	21	177	67	مرد
SA11	19	170	80	مرد
SA12	25	153	47	زن
SA13	22	157	55	زن
SA14	27	160	46	زن
SA15	25	160	52	زن
SA16	20	169	61	زن
SA17	23	182	75	مرد
SA18	23	181	73	مرد
SA19	30	170	76	مرد
SA20	30	150	42	زن
SA21	30	183	68	مرد
SA22	19	158	50	زن
SA23	24	156	48	زن
SE01	71	171	102	مرد

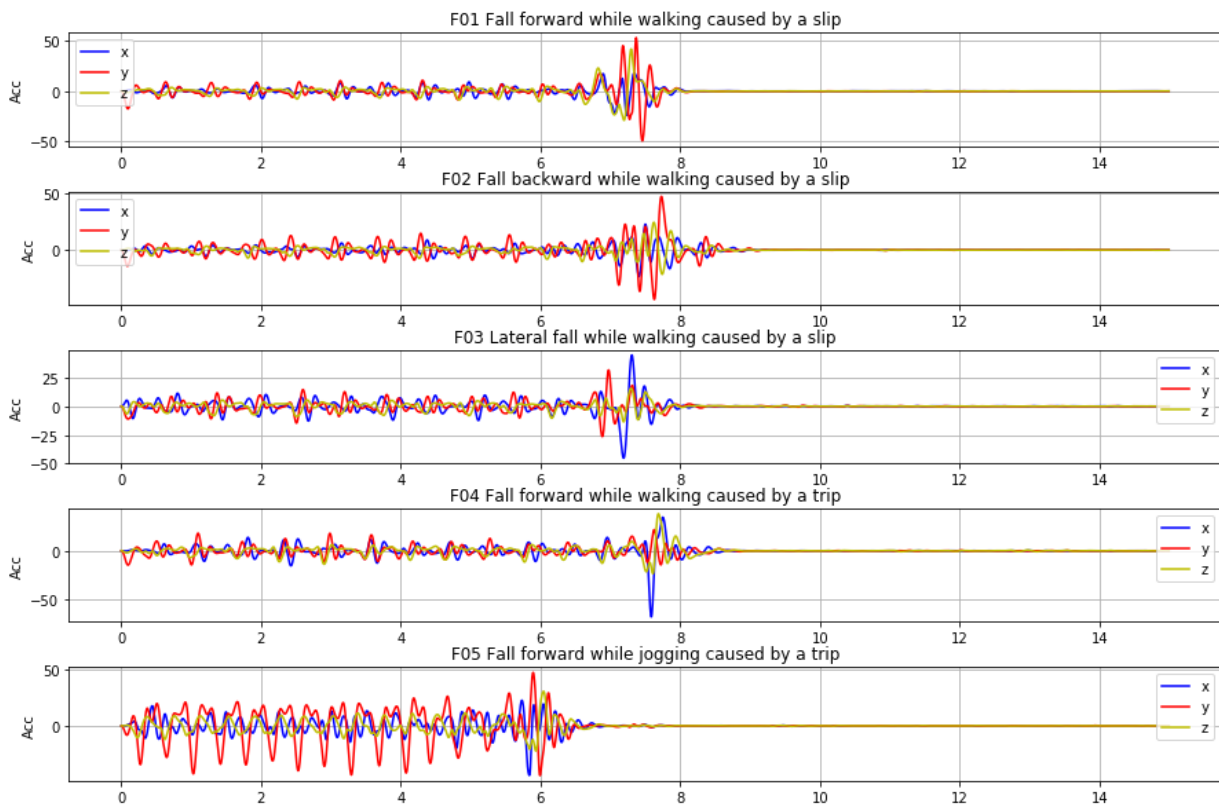
زن	57	150	75	SE02
زن	51	150	62	SE03
زن	59	160	63	SE04
مرد	72	165	63	SE05
مرد	79	163	60	SE06
مرد	76	168	65	SE07
زن	62	163	68	SE08
مرد	65	167	66	SE09
زن	66	156	64	SE10
زن	63	169	66	SE11
مرد	56	164	69	SE12
مرد	73	171	65	SE13
مرد	58	163	67	SE14
زن	50	150	64	SE15

### ۳-۲-۲- سنسورهای استفاده شدن در مجموعه داده

برای تهیه این دیتاست از دو شتاب‌سنج و یکژیروسکوپ استفاده شده که هرکدام از این سنسورها از سه پارامتر در جهت‌های  $x$  و  $y$  و  $z$  تشکیل شده‌اند. به همین خاطر ستون‌های هر یک از فایل‌های پایگاه‌داده از نه ستون تشکیل شده که ستون اول تا سوم مربوط به شتاب‌سنج ADXL345، ستون اول تا سوم مربوط بهژیروسکوپ ITG3200 و ستون اول تا سوم مربوط به شتاب‌سنج MMA8451Q است.

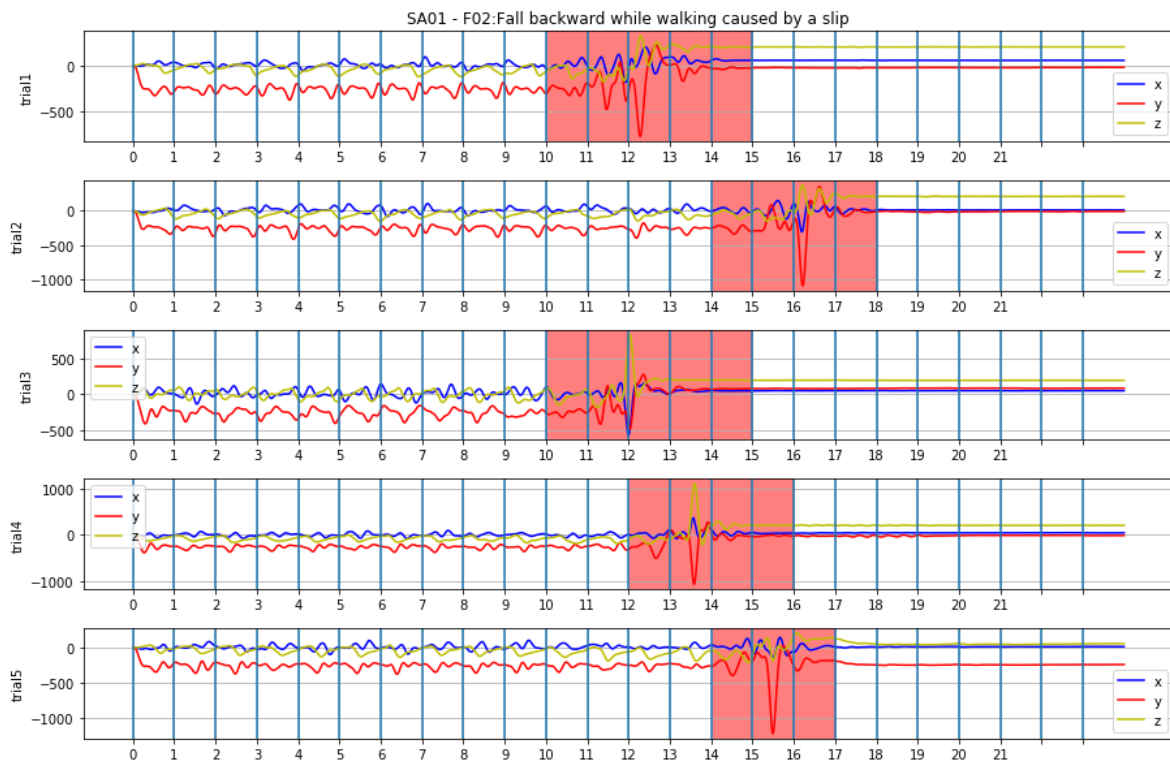
### ۴-۲-۲- آشنایی با جزئیات بیشتری از مجموعه داده

برای آشنایی بیشتر با مجموعه داده‌ها، از دو جهت مجموعه داده‌ها را بررسی می‌کنیم. اول از همه از نظر نحوه کارکرد سنسورها.



شکل ۲-۱. عملکرد سنسورها

همین طور که می بینید نمودار بالا [3] نشان دهنده عملکرد یکی از شتابسنجها هنگام افتادن در انجام فعالیت های مختلف است. این سنسور و البته سایر سنسور در هر لحظه از 15 ثانیه ای که فعالیت های افتادن ادامه دارند، شتاب را در لحظه اندازه گیری می کنند و همین مقادیر برای فعالیت های مختلف است که پایگاه داده را تشکیل می دهد. دومین نموداری که در ادامه بررسی می کنیم فرایند سقوط است.



شکل ۲-۲. فرایند سقوط

نمودار بالا [3] نشان‌دهنده فرایند سقوط فرد SA01 در هنگام انجام فعالیت F02 است. این فعالیت 5 بار توسط این فرد انجام شده و محدوده قرمز رنگ در هر نمودار نشان‌دهنده فرایند سقوط آن فرد به همراه پارامترهای شتاب‌سنج در طول این فعالیت است. همین‌طور که می‌بینید، مقادیر شتاب‌سنج در لحظه سقوط افزایش یا کاهش چشم‌گیری دارند.

### ۲-۳- جمع‌بندی

در این پروژه ما روی مقایسه روش‌های سنتی یادگیری ماشین و روش‌های یادگیری عمیق در مسئله سقوط تمرکز کردیم و دقت و عملکرد این روش‌ها را در تشخیص مسئله سقوط افراد بررسی کردیم.

## فصل ۳: روش پیشنهادی

### ۳-۱- مقدمه

هدف اصلی این پروژه مقایسه روش‌های سنتی یادگیری ماشین با روش‌های یادگیری عمیق است که از Logistic Regression و SVM به عنوان روش‌های سنتی و از شبکه عصبی چندلایه به عنوان روش‌های یادگیری عمیق استفاده شده است.

در ادامه این فصل ابتدا مجموعه داده مورد استفاده در شبیه‌سازی‌ها در بخش 3-2 مورد بررسی قرار می‌گیرد. الگوریتم پیشنهادی پروژه در بخش 3-3 توضیح داده می‌شود. در قسمت 3-4 توضیحی راجع به پلتفرم کدنویسی و کتابخانه‌های مهم مورد استفاده آورده می‌شود. سپس در قسمت 3-5 توابع و بخش‌های مهم کدها آورده شده و به همراه توضیحی راجع به آن‌ها آورده می‌شوند. سپس در بخش 3-6 نتایج شبیه‌سازی ارائه می‌شوند (که شامل جداول، نمودارها، و توضیحاتی راجع به نتایج است). در نهایت در بخش 3-7 نتایج به دست آمده از این سه الگوریتم با یکدیگر مقایسه می‌شوند.

### ۳-۲- مجموعه داده<sup>1</sup>

برای شبیه‌سازی مدل‌های این از مجموعه داده 'SisFall: A Fall and Movement Dataset' استفاده شده که در سال 2016 و توسط دپارتمان مهندسی دانشگاه Udea تهیه شده است. اطلاعات تکمیلی درباره این دیتاست در فصل قبلی و بخش داده شده است. برای دسترسی به این مجموعه داده می‌توانید [روی این لینک](#) کلیک کنید!

علاوه بر این مجموعه داده، از مجموعه داده SisFall Enhanced نیز استفاده شده است که در این مجموعه داده برچسب‌های مربوط فایل در مجموعه داده اولیه آورده شده است. برای دسترسی به این مجموعه داده می‌توانید [روی این لینک](#) کلیک کنید! [1]

<sup>1</sup> Dataset



### ۳-۳- الگوریتم پیشنهادی

باتوجه به این که در این پروژه قصد داریم روش های سنتی یادگیری ماشین را با یادگیری عمیق مقایسه کنیم که SVM و Logistic Regression به عنوان روش های سنتی و از شبکه عصبی چندلایه استفاده شده است.

در این الگوریتم ابتدا آدرس همه فایل های داخل این دیتاست جمع آوری می شود و در داخل یک آرایه ذخیره می شود. سپس این آدرس ها به زیرمجموعه های آموزش، آزمون و ارزیابی (در موارد خاص به دو زیرمجموعه آموزش و ارزیابی) به شکل تصادفی تقسیم بندی می شوند که 70 درصد آدرس های دیتاست سهم آموزش و 30 درصد آزمون و ارزیابی مدل ها هستند.

سپس برای همه آدرس های موجود در هریک از این زیرمجموعه ها سه عملیات مهم انجام می شود. ابتدا داده ها و پارامترهای سنسورها برای هر آدرس خوانده می شود و در مرحله بعدی ما ویژگی هایی که برای آموزش مدل ها مدنظرمان است را از این پارامترها استخراج می کنیم و در آرایه های مجزا ذخیره می شوند.

ویژگی هایی که برای هر آموزش هر کدام از این مدل ها استفاده شده است، ریشه دوم مجموع مجذور مقادیر پارامترهای سه سنسور موجود در این دیتاست به همراه برجسب های موجود در دیتاست مکمل و یا همان Sisfall\_Enhanced است که در بخش بعدی به طور کامل تر توضیح داده می شود. در گام بعدی این ویژگی ها و برجسب های استخراج شده برای هر بخش به طور مجزا پنجره بندی می شوند و در نهایت سه (دو) مجموعه پنجره بندی شده در اختیار ما قرار می گیرد تا استفاده از آن ها مدل ها آموزش ببینند.

در نهایت برای ارزیابی هر کدام از مدل ها از confusion matrix و سه معیار precision ، recall و f1 score استفاده شده است.

### ۳-۴- پلتفرم کدنویسی و کتابخانه های مورد استفاده

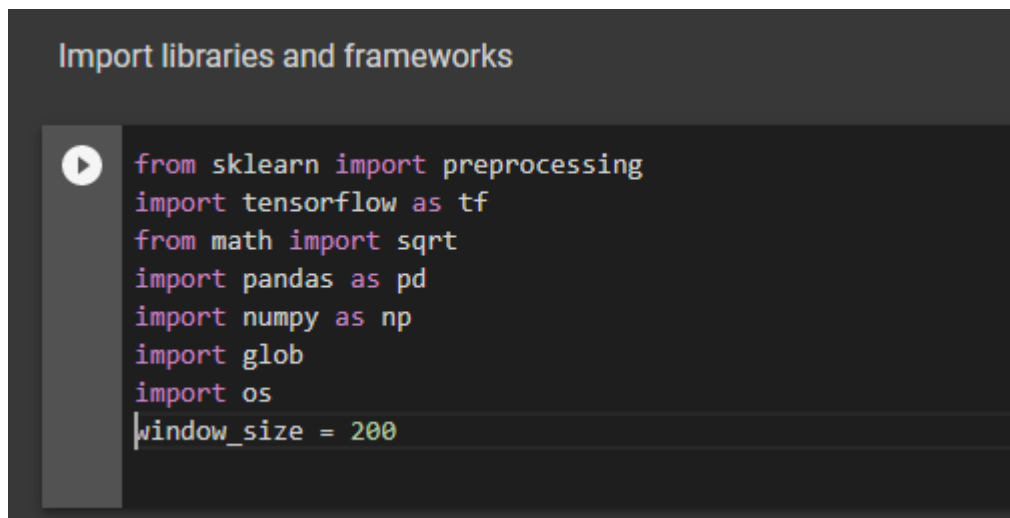
برای کدنویسی این پروژه از زبان python3.7 به عنوان زبان مرجع و برای پیاده سازی کدها از پلتفرم

google colab استفاده شده که 12.69 GB Ram و 107 GB Disk به عنوان سخت افزار در اختیار کاربر قرار می دهد. همین طور Tensorflow و Sklearn کتابخانه های مهمی هستند که برای پیاده سازی کدها استفاده شده است.

### ۳-۵- پیاده سازی روش پیشنهادی

پیاده سازی این پروژه از بخش های مختلفی تشکیل شده که در ادامه این بخش به توضیح آن می پردازیم.

#### ۳-۵-۱- کتابخانه ها و framework های مورد نیاز



```

Import libraries and frameworks

from sklearn import preprocessing
import tensorflow as tf
from math import sqrt
import pandas as pd
import numpy as np
import glob
import os
window_size = 200
  
```

شکل ۳-۱.

برای پیاده سازی این پروژه از کتابخانه های pandas برای کار کردن با dataframe ها و فایل های csv. ، از numpy برای کار کردن با آرایه ها و همین طور انجام عملیات ریاضی، از tensorflow برای پیاده سازی شبکه عصبی و از بخش preprocessing کتابخانه sklearn برای انجام نرمال سازی استفاده شده است. همین طور در این بخش طول (سایز) هر پنجره در متغیر window\_size ذخیره

می‌شود.

## ۲-۵-۳- دریافت دیتاست اصلی و مکمل

```
Get the sisfall and sisfall_enhanced dataset

!wget http://sistemic.udea.edu.co/wp-content/uploads/2016/03/SisFall_dataset.zip
!gdown --id 1gvOuxPc8dNgTnxuvPcVuCKif0f98-TV0
!unzip SisFall_dataset.zip
!unzip SisFall_enhanced.zip
```

شکل ۲-۳.

در این بخش ابتدا با استفاده از دستورات `!wget` و `!gdown` دیتاست های `Sisfall` و `Sisfall_enhanced` رو دریافت می‌کنیم و با استفاده از دستور `!unzip` این دیتاست ها را از حالت فشرده خارج می‌کنیم!

## ۳-۵-۳- استخراج آدرس همه فایل‌های داخل دیتاست

```
Get all addresses

[4] def get_file_name(path):
    allfiles = []
    allFolders = glob.glob(path + "*")
    for files in allFolders:
        allfiles.append(glob.glob(files+"/*.txt"))
    if 'desktop.ini' in allfiles:
        allfiles.remove('desktop.ini')
    return np.hstack(allfiles)
```

شکل ۳-۳.

در این بخش تابع `get_file_name` یک آدرس به‌عنوان ورودی می‌گیرد که ما همان آدرس دیتاست

Sisfall را به آن پاس می‌دهیم. این تابع ابتدا آدرس تمام فولدرهای داخل دیتاست را در متغیر allFolders ذخیره می‌کند و سپس با حرکت کردن روی همه فولدرها، آدرس تمام فایل‌ها که خواسته ما هم همین است به دست می‌آورد!

```
get_file_name("SisFall_dataset/")

array(['SisFall_dataset/SA18/D16_SA18_R02.txt',
       'SisFall_dataset/SA18/D07_SA18_R02.txt',
       'SisFall_dataset/SA18/D19_SA18_R05.txt', ...,
       'SisFall_dataset/SE03/D11_SE03_R05.txt',
       'SisFall_dataset/SE03/D17_SE03_R03.txt',
       'SisFall_dataset/SE03/D17_SE03_R04.txt'], dtype='<U37')
```

شکل ۳-۴.

### ۳-۵-۴- استخراج اطلاعات موردنیاز از هر فایل داخل دیتاست

```
read dataset from path

[ ] def read_data(data_path):
    data = pd.read_csv(data_path, header=None)
    data.columns = ['ADXL345_x', 'ADXL345_y', 'ADXL345_z', 'ITG3200_x', 'ITG3200_y', 'ITG3200_z', 'MMA8451Q_x',
                    'MMA8451Q_y', 'MMA8451Q_z']
    data['MMA8451Q_z'] = data['MMA8451Q_z'].map(lambda x: str(x)[-1])
    for name in data.columns:
        data[name] = data[name].astype(float)
    return data
```

شکل ۳-۵.

در این بخش تابع read\_data یک آدرس به‌عنوان ورودی دریافت می‌کند و اطلاعات داخل آن فایل را استخراج می‌کنیم. این تابع به این شکل عمل می‌کند ابتدا هر یک ستون‌ها را نام‌گذاری می‌کند. نحوه نام‌گذاری هم به این شکل است که سه ستون اول مربوط به سه پارامتر شتاب‌سنج ADXL345، سه ستون دوم مربوط به سه پارامترژیروسکوپ ITG3200 و سه ستون آخر مربوط به سه پارامتر شتاب‌سنج MMA8451Q است.

درنهایت بعد از نام‌گذاری ستون‌ها، ابتدا؛ موجود در ستون آخر حذف می‌شود و بعد از همه پارامترها به فرمت float ذخیره می‌شوند و به‌عنوان خروجی این پارامترها استخراج می‌شوند.

### ۵-۳- استخراج برچسب‌های مربوط به هر فایل!

get the label

```
[ ] def get_label(data_path):
    label = data_path[21]
    if label == 'D':
        return int(0)
    elif label == 'F':
        label_path = data_path.replace('dataset', 'enhanced')
        labels = pd.read_csv(label_path, header=None)
        labels[labels == 2] = 1
        return labels
```

شکل ۳-۶.

همان‌طور که در بخش مقدمه توضیح دادیم، فایل‌های fall و not-fall با توجه به نحوه نام‌گذاری‌شان شناسایی می‌شوند. به شکلی که اگر ابتدای اسم فایل کاراکتر D بود آن فایل not-fall و اگر ابتدای اسم فایل کاراکتر F بود آن فایل مربوط به fall می‌شد.

این تابع ابتدا با توجه به آدرس فایل، نوع فایل را شناسایی می‌کند و در صورتی که آن فایل not-fall بود یک بردار تماماً صفر قرار می‌دهد. در صورتی که آن فایل fall بود ابتدا در آدرسش به جای عبارت dataset، عبارت enhanced را قرار می‌دهد تا به معادل همین فایل در دیتاست Sisfall\_enhanced ارجاع داده شود و برچسب‌های جدید به شکل یک بردار به عنوان خروجی داده می‌شود.

نکته‌ای که در این بخش اهمیت دارد این است که در دیتاست Sisfall\_enhanced برای لحظاتی که فرد در شرف افتادن است از عدد 2 استفاده شده است. با توجه به اینکه این لحظات جزئی از افتادن به حساب می‌آیند، این تابع این لحظات را معادل افتادن در نظر می‌گیرد و آن‌ها را با عدد 1 جایگزین می‌کند! البته در نسخه پیشرفته این پروژه می‌توان این لحظات را به عنوان یک برچسب مجزا در نظر گرفت و مدل را در حالت سه کلاسه آموزش داد!

### ۳-۵-۶- استخراج ویژگی‌های مدل از هر فایل

get out the features

```
def add_features(dataset, data_path):
    new_dataset = pd.DataFrame()
    new_dataset['acc_1'] = dataset.apply(
        lambda row: sqrt((row.ADXL345_x ** 2 + row.ADXL345_y ** 2 + row.ADXL345_z ** 2)), axis=1)
    new_dataset['acc_2'] = dataset.apply(
        lambda row: sqrt((row.MMA8451Q_x ** 2 + row.MMA8451Q_y ** 2 + row.MMA8451Q_z ** 2)), axis=1)
    new_dataset['geo'] = dataset.apply(
        lambda row: sqrt((row.ITG3200_x ** 2 + row.ITG3200_y ** 2 + row.ITG3200_z ** 2)), axis=1)
    new_dataset['label'] = get_label(data_path)
    return np.round(new_dataset.to_numpy(), 2)
```

شکل ۳-۷.

تابع `add_feature` یک جدول داده اولیه و خام به همراه آدرسش به عنوان ورودی دریافت می‌کند. برای هر از سطرهای آن جدول، ریشه دوم مجموع مجذور مقادیر پارامترهای سه سنسور را اندازه‌گیری می‌کند و در سه ستون مجزا، در یک `dataframe` جدید ذخیره می‌کند. در کنار این سه ویژگی، به کمک آدرسی که به تابع داده شده و همین‌طور تابع `get_label` برچسب‌های مربوط به آن جدول استخراج می‌شود و به عنوان ستون چهارم در `dataframe` ایجاد شده ذخیره می‌شود. در نهایت همه مقادیر ویژگی‌ها تا دو رقم اعشار گرد می‌شوند و به شکل `numpy_array` به عنوان خروجی قرار گرفته می‌شود!

### ۳-۵-۷- تقسیم‌بندی آدرس‌ها به سه دسته آموزش، ارزیابی و آزمون

split to train/ validate/ test

```
[15] def split_address(dataset_address):
    np.random.shuffle(dataset_address)
    train, validate, test = np.split(dataset_address, [int(len(dataset_address)*0.7), int(len(dataset_address)*0.8)])
    return train, validate, test
```

شکل ۳-۸.

در این بخش همه آدرس‌های موجود در دیتاست `Sisfall` به عنوان ورودی گرفته می‌شود و سپس

این آدرس‌ها Shuffle می‌شوند. در نهایت 70 درصد این آدرس‌ها برای train ، 20 درصد برای test و 10 درصد آدرس‌ها برای validate کردن مدل‌ها در نظر گرفته می‌شود و به‌عنوان خروجی برگردانده می‌شود.

نکته مهمی که باید به آن توجه شود این است که در شرایطی که شبکه عصبی ما همه epoch های در نظر گرفته شده برایش را طی می‌کند، دیگر نیازی به آدرس‌های validate نداریم و 30 درصد آدرس‌ها برای test کردن مدل‌ها استفاده می‌شود.

### ۸-۵-۳- تبدیل آدرس‌ها به ویژگی‌های موردنیازمان

extract features from addresses

```
def datasets_to_ndarray(datasets_address_array):
    result = np.empty((0, 4), int)
    for address in datasets_address_array:
        result = np.concatenate(
            (result, add_features(read_data(address), address)), axis=0)
    return result
```

شکل ۹-۳.

در این بخش ابتدا یک numpy array با چهار ستون (سه ستون برای ویژگی‌ها و یک ستون برای برچسب‌ها) ایجاد می‌شود. سپس ما روی آرایه‌ای شامل همه آدرس‌هاست حرکت می‌کنیم. سپس اتفاقی که رخ می‌دهد این است که ویژگی‌های موجود در هر کدام از این فایل‌ها استخراج می‌شود و به‌صورت ستونی به آرایه خالی که ابتدا ایجاد کردیم اضافه می‌شود. وقتی که روی همه آدرس‌های که به‌عنوان ورودی دریافت شده، حرکت کنیم، آرایه result که خروجی این تابع است، شامل همه ویژگی‌های موردنیاز مدل‌ها برای آموزش دیدن است.

## ۹-۵-۳- پنجره‌بندی کردن ویژگی‌ها

## windowing of the dataset

```

def windowing(dataset, window_size):
    window = window_size * (dataset.shape[1]-1)
    cut = dataset.shape[0] % window_size
    feature = dataset[:-cut, 0:-1]
    label = dataset[:-cut, -1]
    feature = feature.ravel().reshape(feature.size//window, window)
    label = label.reshape(label.size//window_size, window_size)
    label = label.sum(axis=1)
    label[label > 0] = 1
    return feature, label

```

شکل ۱۰-۳.

برای پنجره‌بندی کردن ویژگی‌ها، ابتدا نیاز داریم که مشخص کنیم یک پنجره چه ابعادی دارد. در اینجا چون سائز هر پنجره 200 در نظر گرفته شده و ما سه ویژگی داریم، هر پنجره شامل 600 ویژگی است که به همراه یک ستون مربوط به برچسب آن، هر پنجره شامل 601 داده می‌شود. در مرحله بعدی ما باید داده‌های اضافه را از انتهای آرایه ورودی حذف کنیم. دلیل این کار هم این است که باتوجه به سائز هر پنجره و تعداد ویژگی‌ها، ممکن است برای ایجاد آخرین پنجره داده کافی وجود نداشته باشد و به همین دلیل برای جلوگیری از این اتفاق باید داده‌های اضافه رو حذف کنیم.

در مرحله بعدی ما باید ویژگی‌ها و برچسب‌هایی را که داریم دوباره تقسیم‌بندی کنیم و به پنجره تبدیل کنیم. برای این کار ابتدا همه ویژگی‌ها با دستور ravel به یک آرایه 1 بعدی تبدیل می‌شوند و سپس با دستور reshape دوباره پنجره‌بندی می‌شوند.

این اتفاق همانند ویژگی‌ها روی برچسب‌ها هم اتفاق می‌افتد. با این تفاوت که اگر حتی در یک بخش کوچک از هر پنجره افتادن رخ بدهد، همه آن پنجره به عنوان پنجره افتادن در نظر گرفته می‌شود.



### ۱۰-۵-۳- تبدیل کردن numpy array به tensor

save as tensor

```
def dataset_to_tensor(validate,test,train>window_size):
    validate_feature , validate_label = windowing(datasets_to_ndarray(validate),window_size)
    np.savez('Sisfall_data_validation', inputs=validate_feature, targets=validate_label)
    test_feature , test_label = windowing(datasets_to_ndarray(test),window_size)
    np.savez('Sisfall_data_test', inputs=test_feature, targets=test_label)
    train_feature , train_label = windowing(datasets_to_ndarray(train),window_size)
    np.savez('Sisfall_data_train', inputs=train_feature, targets=train_label)
```

شکل ۱۱-۳.

باتوجه به این که در این پروژه ما شبکه عصبی را با استفاده از کتابخانه tensorflow پیاده سازی می کنیم. داده هایی که به عنوان ورودی به شبکه عصبی داده می شوند باید شکل tensor داشته باشند که این تابع برای ما همین کار را انجام می دهد و سه بخش validate و train و test را به tensor تبدیل می کند و در نهایت سه فایل در حافظه ایجاد می شود.

### ۱۱-۵-۳- آماده سازی داده ها برای مدل سازی

```
[19] train, validate, test = split_address(get_file_name("SisFall_dataset/"))
dataset_to_tensor(validate,test,train>window_size)
```

شکل ۱۲-۳.

برای آماده سازی داده ها ما ابتدا، آدرس ها را تقسیم بندی می کنیم و در قدم بعدی ویژگی های درون هر فایل را استخراج و بعد از آن، این ویژگی ها را پنجره بندی و در نهایت به فرمت tensor درمی آوریم.

## load the tensor and normalization

```

▶ npz = np.load("Sisfall_data_train.npz")
  train_inputs = preprocessing.scale(npz["inputs"].astype(np.float))
  train_targets = npz["targets"].astype(np.int)

  npz = np.load("Sisfall_data_validation.npz")
  validation_inputs = preprocessing.scale(npz["inputs"].astype(np.float))
  validation_targets = npz["targets"].astype(np.int)

  npz = np.load("Sisfall_data_test.npz")
  test_inputs = preprocessing.scale(npz["inputs"].astype(np.float))
  test_targets = npz["targets"].astype(np.int)

```

شکل ۳-۱۳.

در قدم بعدی باید ویژگی‌ها و برچسب هر کدام از فایل‌های npz. که به شکل tensor هستند را روی یک متغیر ذخیره کنیم تا بتوانیم در ادامه راه از آن‌ها استفاده کنیم. در این بین مقادیر ویژگی‌ها را به کمک کتابخانه sklearn نرمال‌سازی می‌کنیم تا دقت و البته سرعت مدل‌ها افزایش پیدا کنند. الان همه چیز آماده مدل‌سازی است! البته قبل از مدل‌سازی نگاه کردن به ابعاد هر یک از قسمت‌های آموزش، ارزیابی و آزمون خالی از لطف نیست!

```

▶ print(validation_inputs.shape)
  print(test_inputs.shape)
  print(train_inputs.shape)

☞ (7863, 600)
   (15492, 600)
   (55938, 600)

```

شکل ۳-۱۴.

## ۳-۶- مدل‌سازی و نتایج شبیه‌سازی

برای پیاده‌سازی این پروژه، همان‌طور که در ابتدای این گزارش ذکر شد، از شبکه عصبی به عنوان

روش‌های یادگیری ماشین مدرن و از logistic regression و SVM به‌عنوان روش‌های سنتی استفاده شده است.

### ۱-۶-۳- شبکه عصبی به همراه validation

```

input_size = 3
output_size = 1
hidden_layer_size = 50

model = tf.keras.Sequential([
    tf.keras.layers.Dense(hidden_layer_size, activation="relu"),
    tf.keras.layers.Dense(hidden_layer_size, activation="relu"),
    tf.keras.layers.Dense(hidden_layer_size, activation="relu"),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

batch_size = 100
max_epochs = 50

```

شکل ۳-۱۵.

برای این بخش یک شبکه عصبی چهار لایه پیاده‌سازی شده است که یک پنجره 600 تایی را به‌عنوان ورودی دریافت می‌کند و در نهایت عدد صفر یا یک به‌عنوان و به معنای fall یا not-fall بودن را برمی‌گرداند.

هر لایه داخلی این شبکه عصبی از 50 node تشکیل شده و تابع فعال‌ساز هر لایه داخلی تابع relu می‌باشد و در لایه آخر به دلیل این که خروجی صفر یا یک است از تابع فعال‌ساز sigmoid استفاده شده است. علاوه بر این در این شبکه عصبی از تابع adam به‌عنوان optimizer استفاده شده و loss به‌صورت binary\_crossentropy محاسبه می‌شود! [2]

```
early_stopping = tf.keras.callbacks.EarlyStopping(patience=4)

model.fit(train_inputs,
          train_targets,
          batch_size=batch_size,
          epochs=max_epochs,
          callbacks=[early_stopping],
          shuffle = True,
          validation_data=(validation_inputs,validation_targets),
          verbose = 1
        )
```

شکل ۳-۱۶.

بعد از اجرای شبکه عصبی بالا، confusion matrix و پارامترهای آماری زیر به دست آمده است.

```
485/485 [=====] - 1s 2ms/step - loss: 0.0712 - accuracy: 0.9791
----- Confusion Matrix Created -----
array([[14796,    0],
       [ 623,   73]], dtype=int32)
```

شکل ۳-۱۷.

همان طور که مشاهده می کنید، شبکه عصبی به دلیل این که اکثر برچسب های موجود در این دیتاست صفر و یا همان not-fall می باشد، مدلی که آموزش دیده است هم به طور کلی بخش عظیمی از داده های که برای پیش بینی به آن داده می شود را not-fall در نظر می گیرد.

	Precision	Recall	F1score
0	1.0	0.104885	0.20977

شکل ۳-۱۸.

یکی از راه هایی که برای حل این مشکل پیشنهاد می شود این است که مدل را بدون مجموعه validation اجرا کنیم.

## ۲-۶-۳- شبکه عصبی بدون validation

در این بخش اثر مجموعه validation را حذف می‌کنیم و به مدل اجازه می‌دهیم که تمام مراحل آموزش را طی کند.

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

batch_size = 100
max_epochs = 50

model.fit(train_inputs,
          train_targets,
          batch_size=batch_size,
          epochs=max_epochs,
          shuffle = True,
          verbose = 1
        )
```

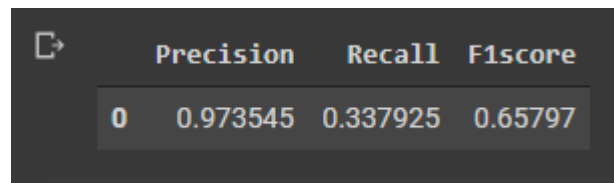
شکل ۱۹-۳.

علاوه بر این کار باید در نحوه پیاده‌سازی توابع آماده‌سازی داده‌ها تجدیدنظر کنیم و داده‌های بخش validation را به داده‌های test الحاق کنیم. بعد از اجرای شبکه عصبی جدید نتایج آماری جدید به دست می‌آیند.

```
736/736 [=====] - 2s 3ms/step - loss: 0.1287 - accuracy: 0.9781
----- Confusion Matrix Created -----
array([[22426, 10],
       [ 721, 368]], dtype=int32)
```

شکل ۲۰-۳.

همان‌طور که می‌بینید مشکلی که پیش آمده بود تا حد زیادی رفع شده و شبکه عصبی دیگر مشکل قبلی را ندارد. نکته مهم و قابل ذکر دیگر این است که با حذف داده‌های validation، شبکه عصبی overfit نشده است.



	Precision	Recall	F1score
0	0.973545	0.337925	0.65797

شکل ۳-۲۱.

علاوه بر این، داده‌های آماری precision، recall و f1 score رشد قابل توجهی کردند و مدل ساخته شده، عملکرد خیلی خوبی دارد.

### ۳-۶-۳- مدل Logistic Regression

```
x = train_inputs
y = train_targets
x_test = test_inputs
y_test = test_targets

[29] reg = LogisticRegression()
      reg.fit(x,y)
```

شکل ۳-۲۲.

برای ایجاد این مدل ابتدا ویژگی‌ها و برچسب‌های مربوط به آموزش و آزمون رو به صورت جداگانه مشخص می‌کنیم و در نهایت مدل ایجاد شده را با پارامترهای از قبل تعیین شده fit می‌کنیم. بعد از اجرا و آموزش این مدل نتایج آماری زیر به دست می‌آید.

```

➤ score on test: 0.9577045696068013
  score on train: 0.9592605210780183

```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	22436
1	0.66	0.18	0.28	1089
accuracy			0.96	23525
macro avg	0.81	0.59	0.63	23525
weighted avg	0.95	0.96	0.95	23525

```

[[22335  101]
 [  894  195]]

```

شکل ۳-۲۳.

همان‌طور که مشاهده می‌کنید، مدل ساخته شده روی داده‌های not-fall عملکرد و دقت بسیار بالایی دارد ولی روی داده‌های fall دقت خیلی بالایی ندارد و شاخص‌های آماری recall و f1-score خیلی قابل توجه نیستند.

البته به این نکته‌ام در confusion matrix باید توجه کنید که تعداد fall‌هایی که از دست داده کمی بالاست!

#### ۳-۶-۴- مدل SVM

```

▶ svm=LinearSVC(C=0.0001)
  svm.fit(x, y)

```

```

➤ LinearSVC(C=0.0001, class_weight=None, dual=True, fit_intercept=True,
  intercept_scaling=1, loss='squared_hinge', max_iter=1000,
  multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
  verbose=0)

```

شکل ۳-۲۴.

برای بخش SVM مدل را با پارامتر  $c=0.0001$  ایجاد شده و آن را روی ویژگی‌ها و برچسب‌ها fit کردیم. بعد از اجرا، آموزش و ارزیابی این مدل نتایج آماری زیر به دست می‌آید.

```

➡ score on test: 0.9546014877789586
   score on train: 0.9544549839516577

```

	precision	recall	f1-score	support
0	0.95	1.00	0.98	22436
1	0.86	0.02	0.04	1089
accuracy			0.95	23525
macro avg	0.91	0.51	0.51	23525
weighted avg	0.95	0.95	0.93	23525

```

[[ 22432    4]
 [  1064   25]]

```

شکل ۲۵-۳.

همان طور که می بینید مدل SVM ایجاد شده عملکرد بسیار ضعیفی روی داده های fall دارد و اصلاً کارایی لازم را ندارد. عملاً کاری که این مدل انجام می دهد این است که اکثر برچسب ها را not-fall پیش بینی می کند.

### ۵-۶-۳- نتیجه گیری و یک ایده جدید برای بهبود عملکرد

همان طور که دیدید تا این جای کار از بین سه مدل آموزش داده شده، شبکه عصبی عملکرد و پیش بینی بهتری روی مجموعه test دارد و از لحاظ پارامترهای آماری عملکرد آن تا حدی قابل قبول است. بعد از شبکه عصبی نیز Logistic Regression و در نهایت SVM قرار می گیرد. با توجه به نتایج آماری و confusion matrix هایی که بررسی کردیم، به این موضوع پی می بریم که دلیل اصلی ضعف این مدل ها، کم بودن تعداد برچسب های fall نسبت به تعداد برچسب های not-fall است. برای رفع این مشکل باید دیتاست را balance و متعادل کنیم.



روش یادگیری	presicion	recall	F1-score
شبکه عصبی چندلایه	0.97	0.33	0.65
Logistic Regression	0.66	0.18	0.28
SVM	0.86	0.02	0.04

جدول ۳-۱. عملکرد الگوریتم‌ها قبل از متعادل‌سازی

### ۳-۶-۶- متعادل کردن مجموعه آموزش

برای متعادل‌سازی مجموعه آموزش باید هر کدام از برچسب‌های fall را چندین بار تکرار کنیم. برای این کار ابتدا باید تشخیص دهیم که کدام سطر از دیتاست fall است و بعد از آن، باید آن سطر را تکثیر کنیم.

```
[6] new_train_dataset = np.concatenate((train_inputs, train_targets.reshape((train_targets.shape[0], 1))), axis=1)
df = pd.DataFrame(new_train_dataset, columns = None)
df.columns = [*df.columns[:-1], 'label']
reps = [305 if val == 1 else 1 for val in df.label]
df = df.loc[np.repeat(df.index.values, reps)].reset_index(drop=True)
new_train = df.to_numpy()
print(new_train_dataset.shape)
new_train.shape

(55769, 601)
(834313, 601)
```

شکل ۳-۲۶.

برای پیاده این بخش، من عدد تکثیر را برای شبکه عصبی 305 و برای مدل‌های Logistic و SVM و Regression عدد 128 را در نظر گرفتیم.

### ۳-۶-۷- شبکه عصبی بعد از متعادل شدن مجموعه آموزش

بعد از متعادل کردن مجموعه آموزش، داده‌های و برچسب‌های جدید را به عنوان ورودی به شبکه

عصبی می‌دهیم.

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

batch_size = 100
max_epochs = 50

model.fit(new_train[:, :-1],
          new_train[:, -1],
          batch_size=batch_size,
          epochs=max_epochs,
          shuffle = True,
          verbose = 1
        )
```

شکل ۳-۲۷.

بعد از اجرای شبکه عصبی جدید نتایج آماری جدید به دست می‌آیند که نسبت به نتایج قبلی تغییرات داشته‌اند.

```
736/736 [=====] - 2s 3ms/step - loss: 0.3652 - accuracy: 0.9750
----- Confusion Matrix Created -----
array([[22357, 79],
       [ 442, 647]], dtype=int32)
```

شکل ۳-۲۸.

همین‌طور که می‌بینید با متعادل‌سازی داده‌های آموزش، مدل ما عملکرد بهتری روی داده‌های fall داشته و پیش‌بینی‌های انجام شده دقیق‌تر هستند. همین‌طور شاخص‌های آماری نیز افزایش پیدا کردند.

	Precision	Recall	F1score
0	0.89	0.59	0.71

شکل ۳-۲۹.

باتوجه به confusion matrix و البته شاخص‌های آماری به دست آمده، شبکه عصبی آموزش داده شده، عملکرد قابل قبولی روی این مجموعه دارد. البته که با افزایش تکثیر داده‌های fall و متعادل تر شده داده‌ها، پیش‌بینی می‌شود که عملکرد شبکه عصبی باز هم بهبود پیدا کند.

### ۸-۶-۳- مدل logistic Regression بعد از متعادل شدن مجموعه آموزش

```

score on test: 0.47519659936238046
score on train: 0.9004739958426943

```

	precision	recall	f1-score	support
0	0.99	0.45	0.62	22436
1	0.08	0.94	0.14	1089
accuracy			0.48	23525
macro avg	0.54	0.70	0.38	23525
weighted avg	0.95	0.48	0.60	23525

```

[[10155 12281]
 [   65  1024]]

```

شکل ۳-۳۰.

با متعادل تر شدن مجموعه داده‌های آموزش، عملکرد و دقت مدل روی داده‌های not-fall افت پیدا کرد، اما نکته مثبت و مزیتی که این مدل نسبت به مدل قبلی دارد این است تعداد fall هایی که از دست می‌رود، بسیار کاهش پیدا کرده است و دقت بسیار افزایش یافته ولی از طرفی دیگر تعداد not-fall هایی که به درستی پیش‌بینی شده عملاً نصف شده است.

### ۹-۶-۳- مدل SVM بعد از متعادل شدن مجموعه آموزش

```

score on test: 0.16017003188097767
score on train: 0.868050160623176
      precision    recall  f1-score   support

     0       0.99      0.12      0.21      22436
     1       0.05      0.98      0.10       1089

 accuracy          0.16      23525
 macro avg          0.52      0.55      0.16      23525
 weighted avg          0.95      0.16      0.21      23525

[[ 2697 19739]
 [   18  1071]]

```

شکل ۳-۳۱.

همین‌طور که می‌بینید، شرایط مدل SVM نیز مشابه Logistic Regression است، با این تفاوت که عملکرد و دقت مدل SVM پایین‌تر آمده است و عملاً اکثر داده‌هایی که به‌عنوان ورودی به این مدل داده شده را fall پیش‌بینی کرده است.

### ۷-۳- مقایسه و نتیجه‌گیری

باتوجه به نتایج و شاخص‌های آماری که در بخش ۳-۶ بررسی کردیم، از بین این سه مدل، شبکه عصبی عملکرد دقت بهتری نسبت به دو مدل دیگر روی مجموعه test دارد.

روش یادگیری	presicion	recall	F1-score
شبکه عصبی چندلایه	0.89	0.59	0.71
Logistic Regression	0.08	0.94	0.14
SVM	0.10	0.98	0.05

جدول ۳-۲. عملکرد الگوریتم‌ها بعد از متعادل‌سازی

ولی مسئله‌ای که وجود دارد این است که این مدل تعداد fall هایی که شبکه عصبی نمی‌تواند پیش‌بینی کند کمی زیاد است که با بهینه‌تر شدن شبکه عصبی و استفاده از روش‌های بهتری برای آماده‌سازی داده‌ها، دقت مدل از این هم بالاتر می‌رود.

```

----- Confusion Matrix Created -----

[[23624   56]
 [  414  657]]

Precision Recall F1score
0          0.92   0.61   0.73

```

شکل ۳-۳۲.

البته که از بین مدل‌های سنتی Logistic Regression از این جهت که fall را به درستی تشخیص می‌دهد، مدل خیلی خوبی به شمار می‌رود. (این مدل بیش از 95 درصد داده‌هایی که به عنوان ورودی گرفته است را fall پیش‌بینی کرده است و پایین بودن معیارهای precision و f1-score این موضوع را به خوبی نشان می‌دهد). فراموش نکنیم که هدف اصلی این پروژه تشخیص درست fall از سمت مدل ایجاد شده است ولی از طرف دیگر این موضوع که مدل اکثر داده‌هایی که به عنوان ورودی به آن داده می‌شود را fall پیش‌بینی می‌کند، می‌تواند برای فردی که از سنسورها استفاده می‌کند، آزاردهنده باشد.

```

score on test: 0.47519659936238046
score on train: 0.9004739958426943

precision recall f1-score support

0          0.99   0.45   0.62   22436
1          0.08   0.94   0.14   1089

accuracy          0.48   23525
macro avg         0.54   0.70   0.38   23525
weighted avg      0.95   0.48   0.60   23525

[[10155 12281]
 [    65  1024]]

```

شکل ۳-۳۳.

## فصل ۴:

# جمع‌بندی و پیشنهاد

## ۴-۱- نتیجه گیری

در فصل سوم به طور کامل با نحوه آماده سازی داده ها و عملکرد مدل آشنا شدیم. نکته مهمی که باید به آن توجه شود، اهمیت پنجره بندی کردن داده ها است که اگر این کار انجام نشود مدل روند یادگیری به خوبی تشخیص نمی دهد و یادگیری به درستی انجام نمی شود.

یکی از مزیت های اصلی روش پیشنهادی بنده نسبت به روش های دیگر، سرعت بالای آماده سازی داده ها نسبت به روش های دیگر است. به طور کلی آماده سازی داده ها جهت آموزش دیدن مدل ها چیزی حدود 28 دقیقه زمان می برد که در مقایسه با روش های دیگر که اکثراً بالای 90 دقیقه زمان می برند، عملکرد خیلی مناسبی دارد.

دلیل اصلی برتری این روش، استفاده حداکثری از آدرس فایل ها (به جای خود فایل ها) و همین طور کارکردن روی numpy array ها (به جای کارکردن روی dataframe) است که سرعت الگوریتم را تا حد زیادی بالا می برد.

در پایان به عنوان نتیجه گیری، مدل شبکه عصبی چندلایه نسبت به مدل های سنتی مثل SVM و Logistic Regression عملکرد قابل قبول و بهتری روی داده های test است و انتظار می رود با بهینه تر شدن الگوریتم و در نظر گرفتن مواردی که در بخش بعد آورده شده، عملکرد مدل بهبود چشمگیری را تجربه کند.

	Precision	Recall	F1score
Algorithm			
Neural Network	0.980000	0.330000	0.490000
Logistic Regression	0.530000	0.170000	0.260000
SVM	0.380000	0.010000	0.030000
Neural Network after Balancing	0.880000	0.600000	0.710000
Logistic Regression after Balancing	0.080000	0.930000	0.150000
SVM after Balancing	0.060000	0.970000	0.110000

شکل ۴-۱.

## ۴-۲- پیشنهادهایی برای کارهای آتی

به طور کلی دو ایده و پیشنهاد مطرح می شود که در این مقاله فرصت بررسی آن را پیدا نکردم. با توجه به این که از این سنسورها در محیط های واقعی استفاده می شود، یکی ایده هایی که مطرح است، استفاده از میانگین هندسی هنگام استخراج ویژگی ها است. ایده دومی که پیش بینی می شود عملکرد مدل ها را تا حد زیادی بهبود می دهد، تغییر شیوه پنجره بندی به گونه ای است که قسمتی از پنجره های ایجاد شده با هم هم پوشانی داشته باشند.



## مراجع :

- [1] [https://bitbucket.org/unipv\\_cvmlab/sisfalltemporallyannotated/src/master/](https://bitbucket.org/unipv_cvmlab/sisfalltemporallyannotated/src/master/)
- [2] [https://github.com/naomikgrant/SisFall\\_DL](https://github.com/naomikgrant/SisFall_DL)
- [3] <https://github.com/WJMatthew/SisFallAnalysis>
- [4] <https://www.semanticscholar.org/paper/A-Smartphone-Application-for-a-Portable-Fall-System-Boehner/1253cbac7aa08671abd870576b39f2fff11008bd>
- [5] <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0140929>



**Kharazmi University**  
**Faculty of Mathematical Sciences and Computer**  
**Department of Computer Sciences**

Final Report of bsc Project

**(Fall detection algorithm)**

By:

**Mohammadali Sefidi Esfahani**

Supervisor:

**Dr.Soltanian**

**(3992)**

(June 2021)