SFU

Automated Fall Detection

CMPT 733 Final Project

April 13, 2019

Eugene Borissov (301378454)

Scott Hind (301180157)

Salil Khandelwal (301376050)

1. Motivation

Falling is a major safety hazard for elderly people. One-third of people over the age of 65 fall at least once every year[1], and 70% of falls result in unconsciousness or confusion, leading to the elderly person not being able to call for help. This often results in a “long lie” on the floor causing dehydration and muscle damage[2]. The long-term damage to the individual is a fear of falling again, loss of independence, no social contacts, lack of movement, and a decrease in their quality of life, which increases the risk of another possible fall[3]. Clearly, falling is a big hazard!

Getting help quickly, after a fall has occurred, is imperative to mitigating the damaging consequences of a fall. Automated fall detection using wearable sensors can improve the response time to falls, mitigate the severity of injuries, and reduce healthcare costs.

2. Background

There are two general types of fall detection models: threshold-based models and machine-learning models. Threshold-based models categorize an event as a “fall” when the sensor readings exceed a certain threshold. For example, when the sternum accelerometer exceeds 10 m/s^2. These models are easy to understand, but they have not performed very well on test data. Machine-learning models are more complex and there are many different types of such models, including tree-based models, logistic regression, and neural networks, and others.

A major drawback of threshold-based models is the need to have a strict threshold; this oversimplifies the problem, and does not allow enough flexibility in detecting falls. Due to this oversimplification, machine-learning models have shown to outperform threshold based models. Figure 1 shows a comparison of five threshold-based and machine-learning models from a 2017 study by Aziz et al., which is one of the most current and relevant work on fall detection at this moment. Key to their findings is the confirmation that machine-learning models consistently outperform threshold-based models, that SVM is the best performing machine-learning model based on sensitivity and specificity, and that near-falls are the primary source of false-positives.

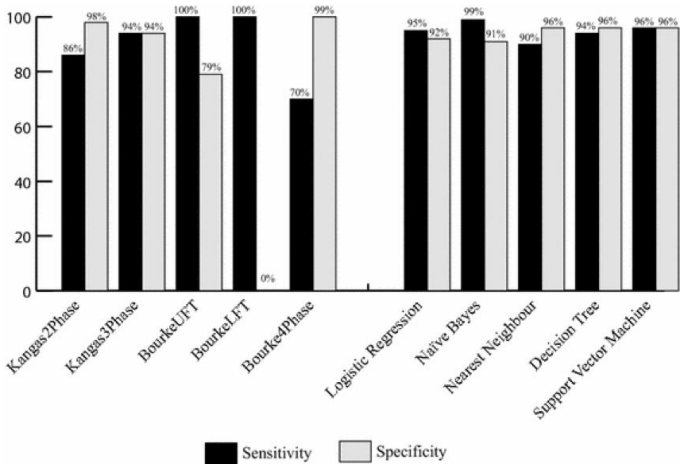

Figure 1: Comparison of sensitivity and specificity between five threshold-based algorithms and five machine-learning algorithms[4]

Various machine learning models that have been tested, including various tree-based models, SVM and logistic regression, however, there is no consensus on what the best machine learning model. We will focus on machine-learning models in our work.

3. Problem Statement

Using data collected from wireless sensors worn by individuals during falls and non-falls, our goal is to:

i) Create a machine learning model that is able to detect that a fall has occurred with high true positive rate (>90%) and have a false positive rate of 0% (as requested in project description);

ii) Determine how well a machine learning model using a single sensor can detect falls;

iii) Explore the different data engineering methodologies to create the best data pipeline to detect falls;

iv) Analyze the precision, recall, accuracy, F1 and confusion plots for the models tested to draw conclusions about the strengths/weakness of the models.

As we start our project, we are well aware that achieving a 0% false positive rate will be particularly difficult because (1) the period before a fall has many similarities to a non-fall, and (2) no models tested previously have been able to achieve this. We are hoping that creative data engineering will allow us to meet this target. If we are not able

to achieve this target, we hope our findings will pave the road for future progress for achieving this goal.

4. Data Science Pipeline

Our data science pipeline consists of all steps taken to transform the raw data provided to us into our machine learning models. The pipeline consists of 5 steps, as shown in Figure 2 below.

4.1. Raw Data

The dataset for this project is the Inertial Measurement Unit (IMU) Fall Detection dataset, provided by SFU RADAR. Per SFU RADAR, this dataset was devised to benchmark fall detection and prediction algorithms based on measurements from body-worn sensors[5].

To collect this data, 10 subjects were outfitted with body-worn sensors mounted on 7 locations on the body, including the left and right ankle, left and right thigh, waist, sternum, and head (figure). Each sensor collects data for acceleration, angular velocity, and magnetic fields, recording at 128 Hertz. This means each sensor records a measurement every 1/128 = 0.007 seconds, resulting in 128 measurements per second. Measurements are recorded in the X, Y, and Z dimensions. In total, there are 63 measurements made each cycle, as there are 7 sensors recording 3 types of measurements in 3 different dimensions.

Each subject underwent 60 trials of roughly 15 seconds each where they simulated a type of physical movement. These movements were split into 3 categories: Activity of Daily Livings (ADLs), Falls, and Near-falls (Table 1).

Table 1: Data Distribution

Activity	Number of Trials	Percent of Trials
Activity of Daily Living	240	40%
Near-fall	150	25%
Fall	210	35%
Total	**600**	**100%**

The data collected from these 600 trials serves as the basis for our machine learning algorithms that aim to detect whether or not a fall has occurred. Data was provided as a directory of Excel spreadsheets, with each trial existing in its own spreadsheet. For further information on the dataset, please refer to the IMU Repository[5].

4.2. Data Processing

The initial processing step involved moving the data from Excel spreadsheets into a more data science friendly format, CSV. Additionally, since individual trials were in their own spreadsheet, these trials were consolidated into a single file.

Overall, the data provided was tidy so no cleaning was necessary. The only tweaks made to the raw data, other than converting to CSV format, was renaming the columns to a more standard format. The original data had column names formatted in the style “l.ankle Angular Velocity Z (rad/s)”. These headings were converted to a more brief format of “l_ank_ang_vel_X”.

During this processing phase, we also had to assign labels to the trials. The labels were implicitly provided to us in the directory structure, so we used this information to add a “label” column to the data. For our labeling structure, we chose to group ADLs and Near Falls together into a single “Non-fall” category, which made this is a binary classification rather than a ternary classification. We chose to make this decision because our objective is to detect whether someone has fallen or not, which is a binary question. It should be noted that each data point had a label, meaning each trial consists of hundreds of data points each with the same label.

4.3. Exploratory Data Analysis

We conducted preliminary EDA to help us understand the structure of the data. Part of this was creating a dashboard that visualized the data for each of the three trial types. The dashboard allows the user to select a sensor, measurement type, and dimension, then plots that data for each of the 3 trial types in the same graph, allowing for quick comparison of the trial types.

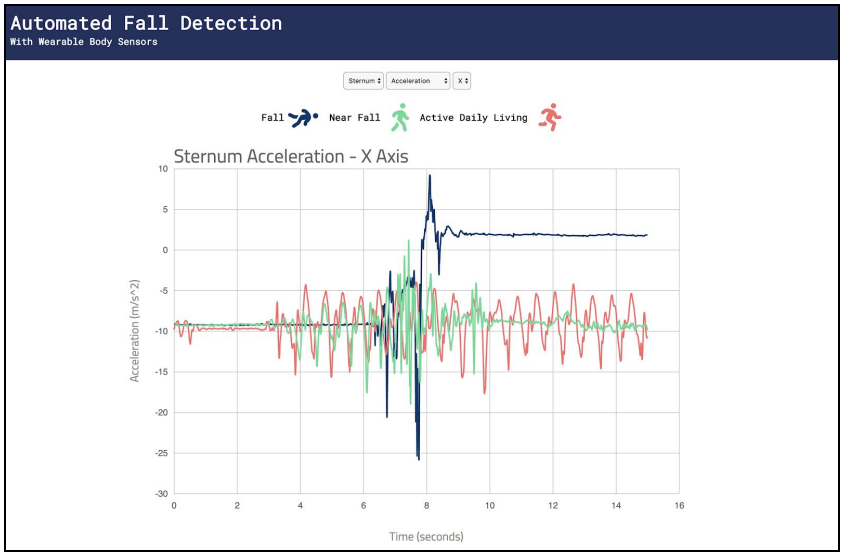

Figure 2. Our dashboard plotting data from different trials and sensors.

This tool revealed two key findings. First, Fall were relatively indistinguishable from Non-falls (Near-falls and ADLS), other than a short time period where the measurements peak, presumedly when the fall is occurring. Second, Falls and Near-falls behave very similarly, except the magnitude of the measurements for Falls is slightly more pronounced around these peaks.

This dashboard was created using Flask, D3, and HTML, and the other EDA was performed using Matplotlib, StatsModels, and Plotly.

4.4. Machine Learning

Once the raw data was processed and understood, we were able to begin the process of creating machine learning models. This process included feature selection and feature engineering. We then experimented with several different classification methods.

Our machine learning utilizes Spark ML, scikit-learn, and Matplotlib.

4.5 Results

After formulating our different modeling approaches, we compiled the results for the models and compared them. This was an iterative process, as we experimented with different methods of feature engineering and modeling until we reached our target results.

5. Methodology

We took three different approaches to develop machine learning models to detect falls using our processed data set. Before doing this, we first did some feature engineering. We computed a resultant vector for each sensor, which aggregates the measurements from all three X, Y, and Z dimensions into a single measurement. The resultant vector is computed with the formula *Resultant* = $\sqrt{X^2 + Y^2 + Z^2}$.

For our machine learning models, we tested 6 different classification models.

- Logistic Regression
- Support Vector Machines
- Naive Bayes
- Random Forests
- Gradient Boosted Trees
- Decision Trees

We took three different approaches and applied each classification method for all of them.

5.1. Approach A: Basic Model

For this approach, we used only the raw data with the resultant vector as inputs into our models.

5.2 Approach B: Windowed Data

For this method, we aggregated the data into “windows”. For each window of a predetermined length, we computed the maximum values, minimum values, and variance for each measurement within the window. We experimented with a window size of 0.1s, 0.5s, 1s, 1.5s, and 2s.

5.3 Approach C: Fall Peak Extraction

This approach was based on the EDA mentioned in the previous section. As discussed, we discovered that within the 15s Fall trials, the actual act of falling occurs within roughly a 1-2s window where peak measurements occur. This means that the data outside of this peak window was labeled as a fall, but closely resembled the data of a

Non-fall trial. We hypothesized that these incidentally mislabelled data points would negatively impact the model results.

To solve this, we extracted the data within +/- 1 second of the peak of the Fall and discarded the data from Fall trials outside this peak window. To determine the peak, we found the maximum acceleration of the Sternum sensor in the X dimension. This specific measurement was chosen after examining results from our basic models. By using the Feature Importance property of Gradient Boosted Trees, it was shown that this measurement from the Sternum sensor was the most important feature for detecting falls. Our models were then trained on this dataset. Figure 3 illustrates the data extracted from this approach.

Additionally, we aimed to discover which single sensor is best at detecting falls. In a real-life situation, it is not practical to outfit humans with seven sensors to monitor their movement. A practical application of these models would likely involve only a single sensor. To account for this scenario, we took our best model of the approaches above and applied it to individual sensors to see which sensors are best at detecting falls.

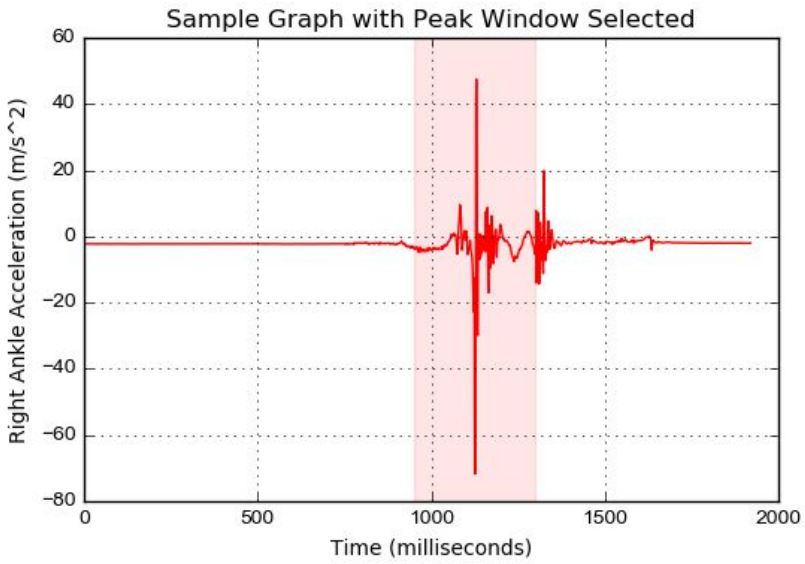

Figure 3. An example of a Fall trial with the peak window indicated by the red background.

6. Evaluation

Approach C produced by far the best results compared to the basic models and window models. This confirms our hypothesis that the start and end of the Fall trials, where no falling was actually occurring, was causing the models to incorrectly classify some Non-falls as Falls. Figure 4 shows the results from the Peak Fall Extraction models.

As a reminder, the target results are a 0% false positive rate and an acceptable true positive rate above 90%. Per the confusion matrices shown below, Logistic Regression was the best model and met both target percentages. The tree-based methods (Decision Trees, Gradient Boosted Trees, and Random Forests) had a near 0% false positive rate, yet did not meet the acceptable true positive rate. Support Vector Machines performed slightly better but still did not meet the true positive rate. Naive Bayes exceeded the true positive rate but had the worst false positive rate of 6%.

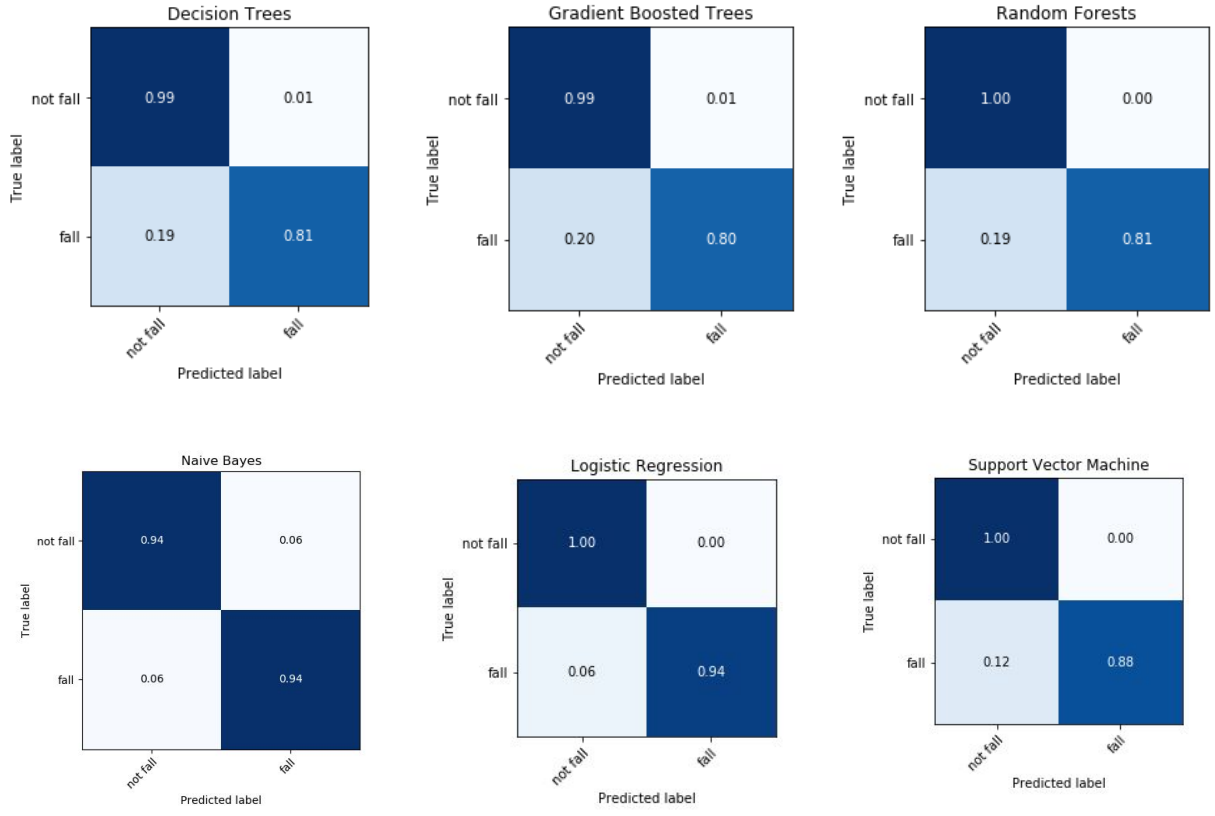

Figure 4. Confusion matrices showing the accuracy of our machine learning models.

All models performed well as shown in Figure 5, especially in detecting when someone has not fallen. However, Logistic Regression was the best performing model when considering our objectives.

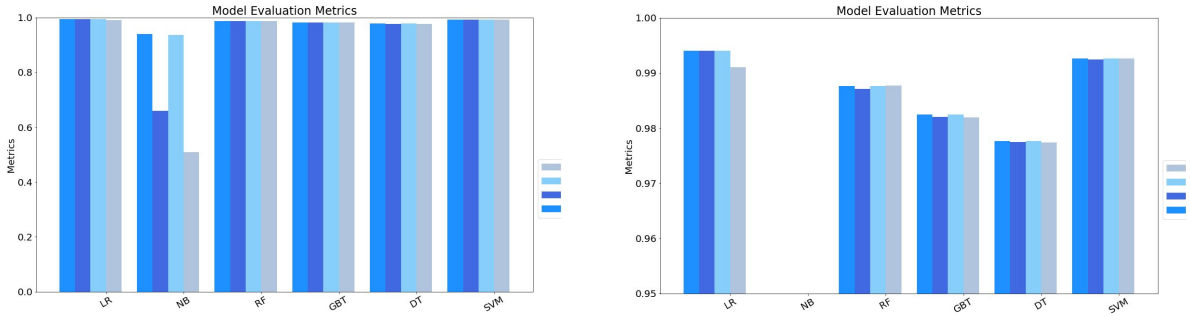

Figure 5. Precision, Recall, F1, and Accuracy for each model. Full results (left) and magnified results (right) to show differences at the peaks.

With Logistic Regression as our best model, we applied the model to data from single sensors to see how well they can detect falls. The Sternum sensor had the lowest false positive rate of 1% but did not perform well in detecting true positives (Figure 6). The right ankle sensor had the highest true positive rate of 72% to go with a 2% false positive rate. The false positive rates are encouraging for single sensors, however, they proved to be insufficient in detecting falls when compared to the models using all seven sensors.

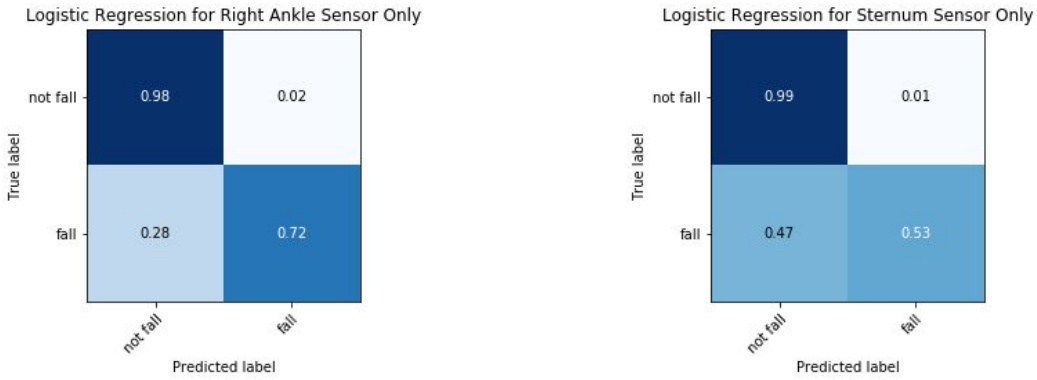

Figure 6. Confusion matrices showing the accuracy of Logistic Regression applied to individual sensors, the right ankle (left) and sternum (right).

7. Data Product

The models we created can be used to detect falls in real time. Our logistic regression model can be deployed to handle real-time sensor data - this model can detect falls with a 0% false positive rate and 94% true positive rate. This model can also be deployed to handle a single sensor, although this would result in lower accuracy. We believe that, if deployed on sensors in the real world, this model can help detect falls, and shorten the response time for first-responders to people who have fallen and cannot call for help.

To allow a non-technical audience explore and understand the the data used in our project, we created a dashboard that utilizes a user-friendly graphic user interface to visualize sensor data. The dashboard allows the user to select a sensor, reading, and direction of the reading, for each type of trial, and displays a graph the select data. and Our data product can help in improving the response time to falls, which can help in mitigating the severity of injuries, especially in elderly people.

8. Lessons Learned

The lessons learned during this project are:

- The project was an excellent learning experience for us. We enhanced our knowledge of the data science pipeline by creating one from scratch.
- We learned about creating a data product, taking into account real-world complications.
- In this project, we learned about exploratory data analysis, feature engineering, machine learning classifiers, data visualization, and also data manipulation using windowing and concentrating the data around the peak as mentioned earlier in the report.
- We handled class imbalance by analyzing multiple metrics (precision, recall, F1, and accuracy) to assess our models.
- We determined that sternum sensors' acceleration provides the most variation for detecting a fall.
- We learned about visualization using D3 and JavaScript as well.
- We observed why machine learning methods are better than conventional threshold-based approaches.
- Given more time, we would explore how deep neural networks and recurrent neural networks perform when applied to this problem. We would also like to stream the sensor data in order to produce real-time prediction using our models.

9. Summary

Our project achieved the goals that we set out to solve in our problem statement. We utilized data engineering to create a data pipeline for the logistic regression classification model that was able to detect a fall with a 0% false positive rate and 94% true positive rate. We also determined that a single sensor does not achieve good enough results using the models that we tested and that more complex machine learning models may be required to achieve acceptable results using a single sensor.

In conclusion, our data product can detect falls very well, and if deployed, it will reduce the response time to elderly people who have fallen and are unable to call for help, mitigating the severity of their injuries.

10. References

(1) Hu, X., & Qu, X. (2016). Pre-impact fall detection. Biomedical engineering online, 15(1), 61. doi:10.1186/s12938-016-0194-x

(2) Aziz O, Klenk J, Schwickert L, Chiari L, Becker C, Park EJ, Mori G, Robinovitch SN. Validation of accuracy of SVM-based fall detection system using real-world fall and non-fall datasets. PLoS One. 2017 Jul 5;12(7):e0180318. doi: 10.1371/journal.pone.0180318. PubMed PMID: 28678808; PubMed Central PMCID: PMC5498034.

(3) Vallabh, P., & Malekian, R. (2017). Fall detection monitoring systems: a comprehensive review. Journal of Ambient Intelligence and Humanized Computing, 9(6), 1809–1833. doi:10.1007/s12652-017-0592-3

(4) Aziz, Omar & Musngi, Magnus & J. Park, Edward & Mori, Greg & Robinovitch, Stephen. (2016). A comparison of accuracy of fall detection algorithms (threshold-based vs. machine learning) using waist-mounted tri-axial accelerometer signals from a comprehensive set of falls and non-fall trials. Medical & Biological Engineering & Computing. 55. 10.1007/s11517-016-1504-y.

(5) Robinovitch, S. & Aziz, O. Inertial Measurement Unit Fall Detection Dataset. https://researchdata.sfu.ca/islandora/object/islandora%3A9085