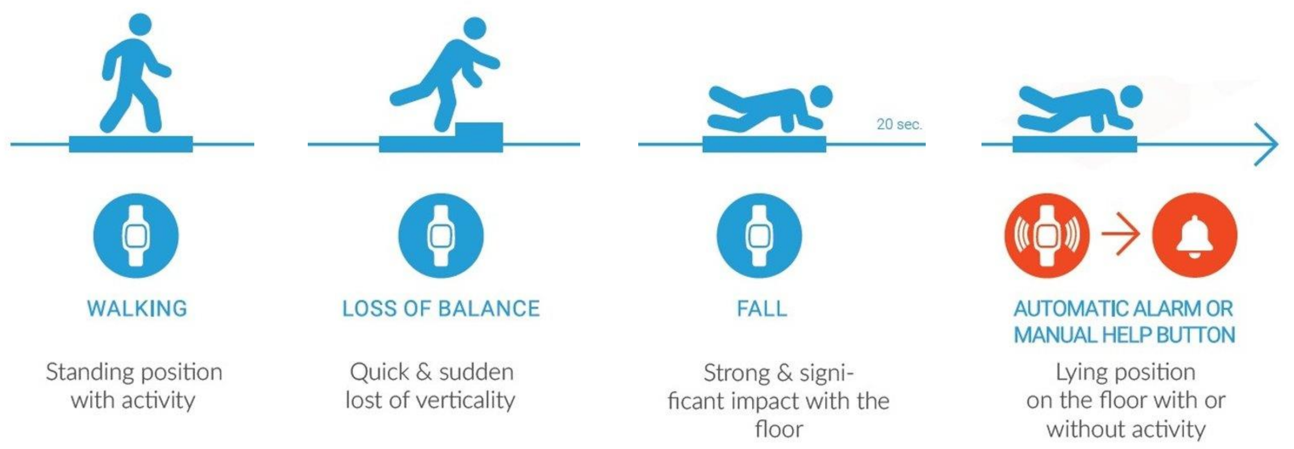
FALL DETECTION USING WEARABLE SENSOR DATA

CMPT 733 – Programming for Big Data 2
Amandeep Singh Kapoor, Inderpreet Singh, Prabhjot Singh
Faculty of Applied Science, Simon Fraser University, Burnaby

INTRODUCTION AND MOTIVATION

- This project is a step towards detecting Fall in real-time through the use of trained Machine Learning Classifiers.
- Accurate "Fall detection" is a very important problem which has a great significance especially in the old age people.



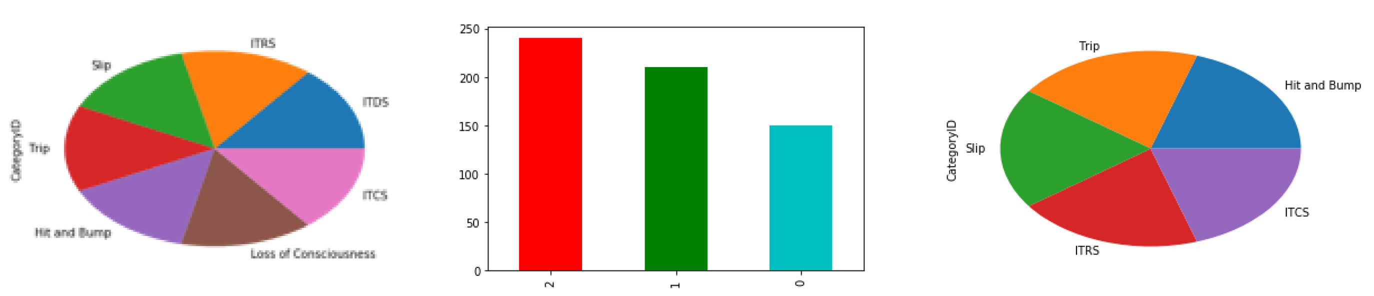
- Over 90 % of hip and wrist fractures and 60 % of traumatic brain injuries in older adults are due to falls.
- Half of elderly people who experienced a long lie (for an hour or more) passed away within 6 months, even if no direct injury occurred from the fall.

APPROACH

- This project is based on the paper "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"
- In this we have used the data generated from the tri-axial sensors mounted at 7 different body parts, capturing body kinematics such as Angular Velocity, Magnetic Field and Acceleration
- The major goal of this project revolves around optimizing two requirements which are **Sensitivity** (ability to detect actual fall) and **Specificity** (ability to avoid False Positive which could desensitize the receiver) I.e. how to identify Optimal window size or data points corresponding to each trial
- We have applied advanced machine learning algorithms in order to achieve better sensitivity/specificity. These techniques include Logistic, Decision, Gaussian Naïve Bayes, Ada Boost, Gradient Boost, Random Forest.
- We have also identified ways to simulate real-life scenario in which sensor data is continuously generated, handled the streaming data and then applied a machine learning model to detect fall.

UNDERSTANDING DATA

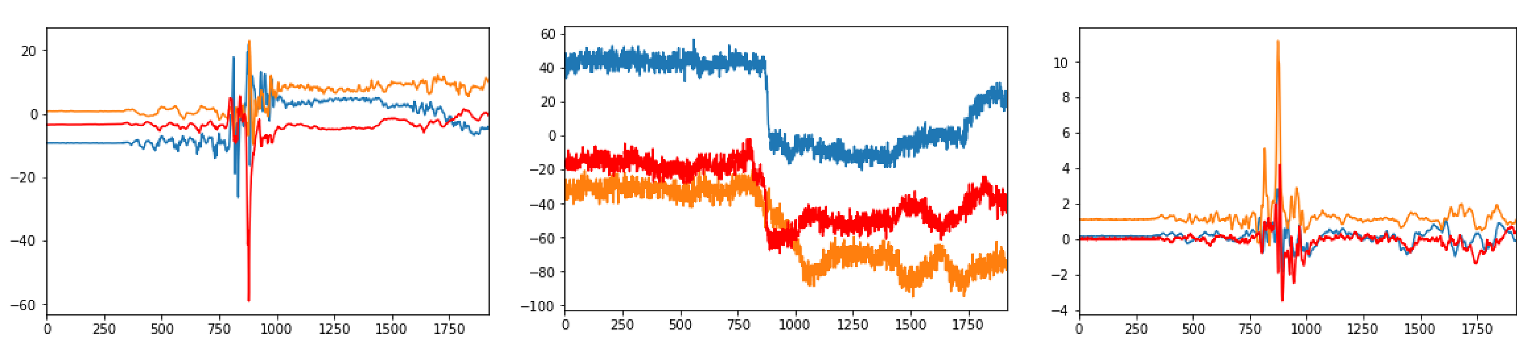
- The current trial data is taken from 10 subjects across 3 categories of Fall, Near Fall and Activity of Daily Living.
- Each of these categories is divided further in to subcategories and there are 3 trials for each of them.



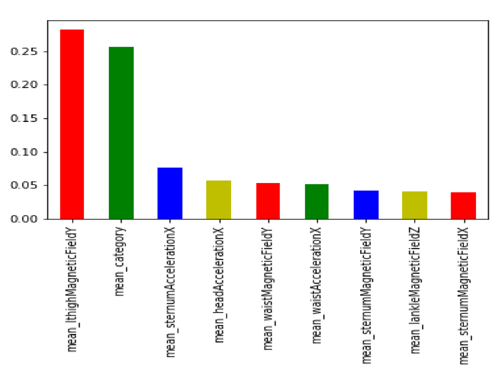
Fall Categories   Class Distribution   Near Fall Categories

FEATURE SELECTION

- The data for each trial is collected for duration of 15 s. This was taken from the tri-axial accelerometer mounted at 5 different body part, recording body kinematics (acceleration, angular velocity, magnetic field.)
- Below is the line graph representing the flow of the collected body kinematics – Acceleration, Magnetic Field and Angular Velocity



Acceleration   Magnetic Field   Angular Velocity



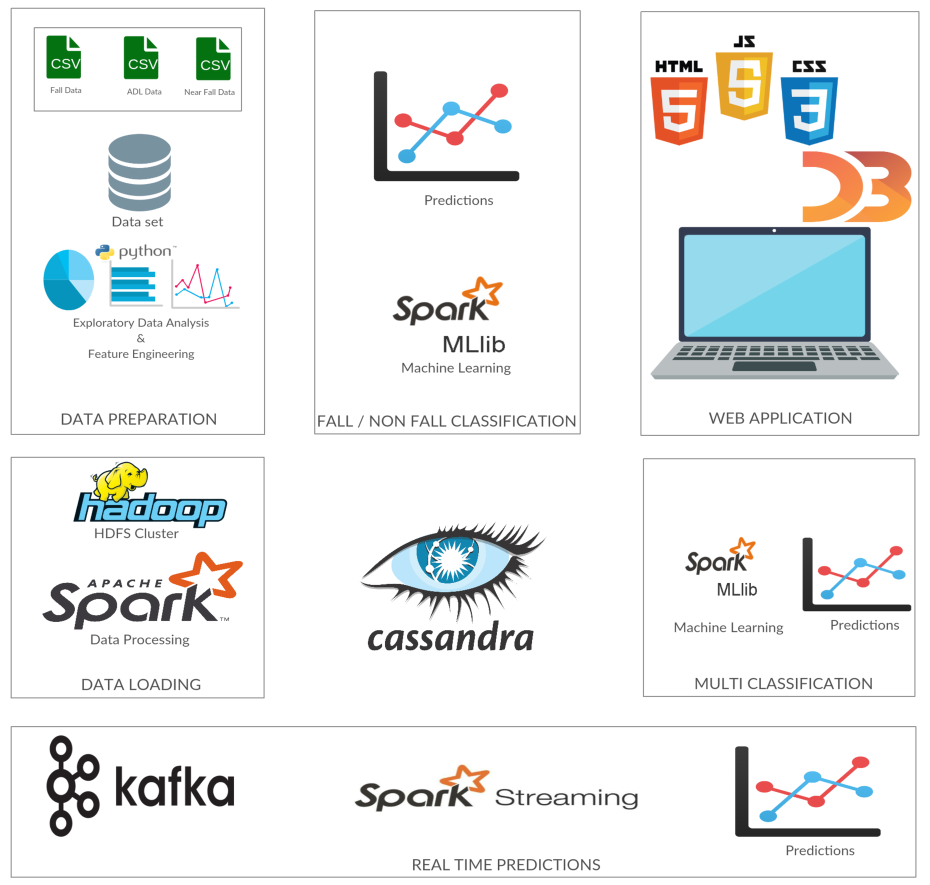
Top 10 Features

- For training our model, we have only considered those data points (collected over a certain time period) which are consistent with the type of the trial i.e. Fall, Near-Fall and ADL.
- Feature transformation for the identified data points was done using various statistical techniques like Mean, Standard Deviation and Covariance
- Feature selection was done using the correlation between dependent and independent features. (as shown in "Top 10 Features")

DATA PIPELINE

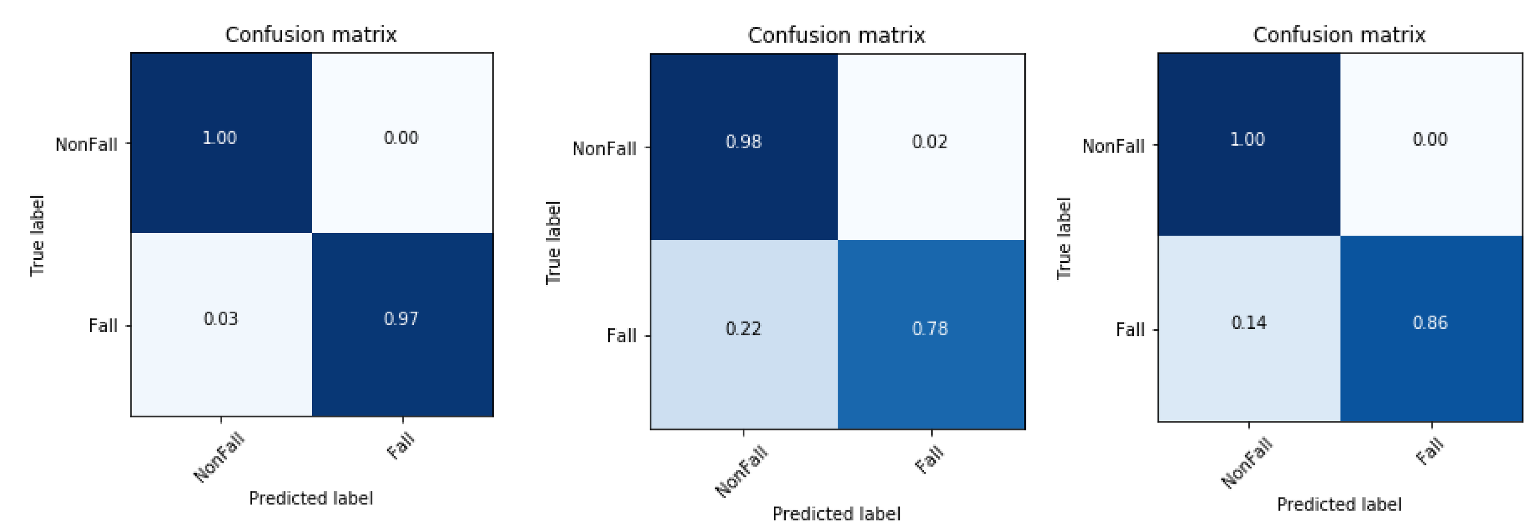
- **Step 1**
  - **Data Processing**
  - Performed data processing and transformation using pandas and numpy. Used statistical techniques to transform selected data points
- **Step 2**
  - **Feature Selection**
  - Used Sci-kit functionalities for identifying the correlation between Independent and Dependent feature.Selected the highly correlated features for model training
- **Step 3**
  - **Machine Learning**
  - Used Sklearn and Spark ML libraries. Trained models are based on various algorithms such as Gradient Boosting, Random Forest, Naïve Baise
- **Step 4**
  - **Prediction and Accuracy Measurement-**
  - Model evaluation was based upon two parameters Sensitivity and Specificity
- **Step 5**
  - **Real-Time Prediction-**
  - Kafka was used as a message broker for processing streaming data. Integrated with Spark Structured streaming for performing real-time prediction
- **Step 6**
  - **Web Application-**
  - Created Web dashboard using d3.js, canvas.js for visualizing the dynamic graphs for displaying the behavior of the simulated data

- This project data flow is broadly divided in to 2 categories: **Offline Model Training** using Batch Processing and **Real-Time prediction** using trained Model.
- In the offline model training, regular batch processing concept is utilized. That is loading the batches/generated data in to Cassandra/HDFS and utilizing Spark ML for training the classifier based on different Machine Learning Models like Random Forest, Gradient Boost
- For real-time prediction, we have used Kafka message broker for storing the stream of data getting generated in real-time
- Spark Structured Streaming is used to consume the data in real-time from the message broker and load the already trained classifiers for making prediction in real-time.
- Once prediction is made, spark streaming would act as a producer and store the data in to a different topic in Kafka where it can be used for web application
- We have also considered a model for **Active Learning,** where the real-time data sitting in Kafka would be loaded in to Cassandra for Batch Processing and Offline training.

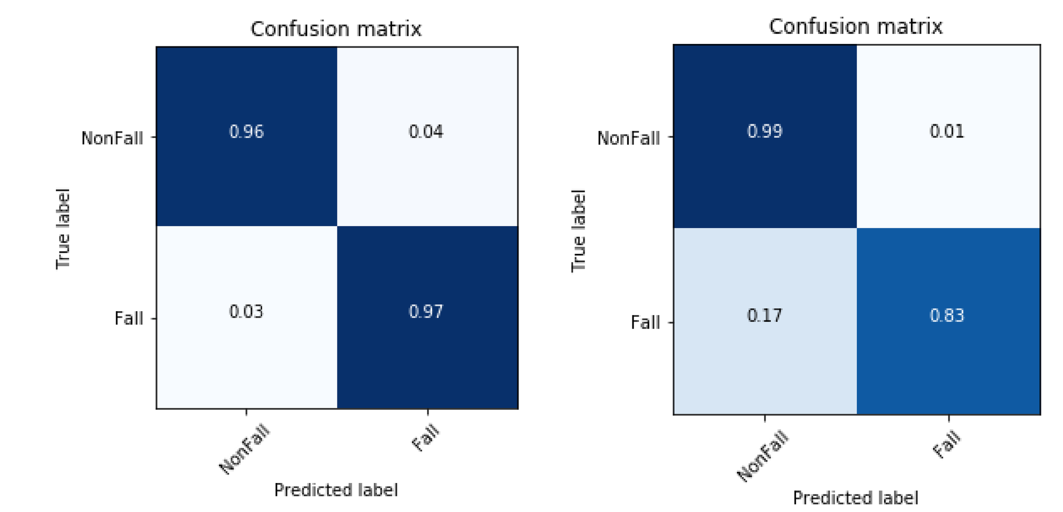


RESULTS

- We have used the values of specificity and sensitivity for evaluating the accuracy of our trained classifiers.
- Specificity is True Negative / (True Negative + False Positive)
- Sensitivity is True Positive/ (True Positive + False Negative)
- Below are the confusion matrices showing the rate of True Negative, True Positive, False Positive and False Negative from different machine learning models. (**2.5 sec window**)



Logistic Regression   Decision Tree   GBT Regressor

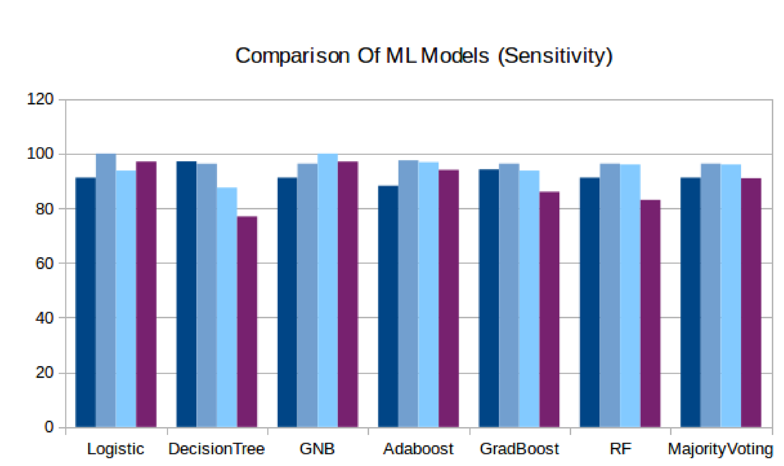


Naïve Bayes   Random Forest

Sensitivity Analysis corresponding to different window sizes							
Window Size(in sec)	Logistic	DecisionTree	GNB	Adaboost	GradBoost	RForest	MajorityVoting
1	91.176	97.05	91.176	88.23	94.17	91.17	91.176
1.5	98.4	96.26	96.29	97.5	96.29	96.29	96.29
2	93.75	87.5	97	96.8	93.75	96	96
2.5	97	77	97	94	86	83	91

Specificity Analysis corresponding to different window sizes							
Window Size(in sec)	Logistic	DecisionTree	GNB	Adaboost	GradBoost	RForest	MajorityVoting
1	97.27	97.27	95.45	96.36	95.45	96.36	96.36
1.5	94.87	98.29	88.88	99.145	95.72	96.58	95.72
2	98.2	97.3	92.85	99.1	99.1	98.214	98.2
2.5	100	98	96	99	100	99	99

Sensitivity and Specificity Analysis for different Window Size (in secs)



Sensitivity across different models

- We have trained our model on various window sizes (in sec).
- It was done to identify the best possible data points that exhibits the behavior of a particular event. (Fall, Near Fall, ADL)
- Based upon our evaluation on different models and window sizes. The best identified model is **Majority Voting** and Optimal Window is **1.5 sec**.
- The best sensitivity is 96.29 and specificity is 95.72 for Majority Voting.

REFERENCES

- Aziz, Omar, Magnus Musngi, Edward J. Park, Greg Mori, and Stephen N. Robinovitch. "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, no. 1 (2017): 45-55.
- Aziz, Omar, Colin M. Russell, Edward J. Park, and Stephen N. Robinovitch. "The effect of window size and lead time on pre-impact fall detection accuracy using support vector machine analysis of waist mounted inertial sensor data." In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pp. 30-33. IEEE, 2014.
- Hsu, Yu-Wei, Kuang-Hsuan Chen, Jing-Jung Yang, and Fu-Shan Jaw. "Smartphone-based fall detection algorithm using feature extraction." In *Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), International Congress on*, pp. 1535-1540. IEEE, 2016.