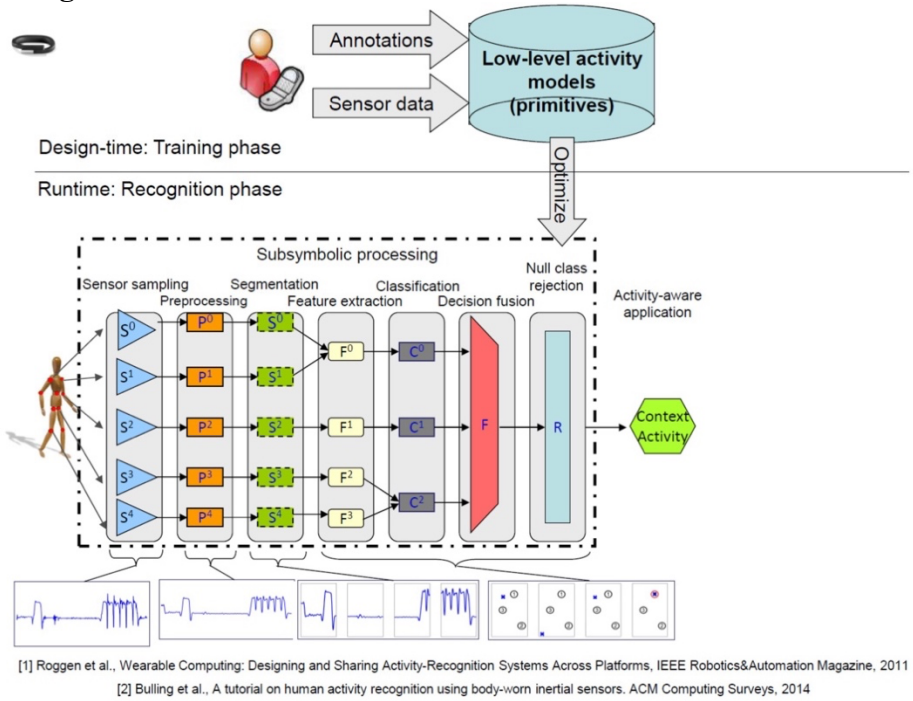
Wearable Technologies (**867H1**)

a) Activity recognition chain



[1] Roggen et al., Wearable Computing: Designing and Sharing Activity-Recognition Systems Across Platforms, IEEE Robotics&Automation Magazine, 2011

[2] Bulling et al., A tutorial on human activity recognition using body-worn inertial sensors. ACM Computing Surveys, 2014

During Design-time, features and the ground truth labels of classes are fed into a classifier model. Using features and model which is pre-trained, we calculate scores for every activity and links the scores to individual class labels.

Activity-recognition chain (ARC) is a set of processing principles generally practiced by most researchers to process the data coming from sensors to infer human gestures or activities. Specific activity recognition system is developed by signal sampling and pre-processing, pattern recognition and machine learning algorithms. The **subsymbolic processing** consists of two stages for achieving the task. First, it maps low-level sensory data (e.g., body-limb acceleration) to expressive action primitives (e.g., grasp), and further maps the sequences of these action primitives (e.g., grasping and cutting) to a higher-level activity (e.g., cooking) (Roggen, et al., 2011)Analysing streams of sensor data helps to infer meaning by comparing them to a known activity and is achieved by signal processing and various machine learning techniques. The outcome of "*subsymbolic processing*" is representative event of primitive action occurrence. ARC terminates at the stage where the context activity of interest comprises of simple gestures (Kallio, et al., 2006). Reasoning, expert knowledge, or statistical approaches can be applied in second stage of mapping to realize the occurrence of action primitives. Large variability in sensor-signal to activity-class mapping is observed, reason being the evolving human behavior as well as sensor deployments. Sub symbolic processing is optimized using sensor selection to maximize comfort as well as recognition performance. The elements of this processing along with their implementation are discussed here.

i. Sensor sampling:

Streams of sensor data or samples '*S*' is acquired using various sensors on different parts of body. Each sensor is sampled at predefined intervals and output becomes a time series data.

Often, sampling rates may differ for certain kinds of sensors, and this is taken care in next step. Sometimes, selecting more than one sensor may improve the performance of the system and multiple sensors are jointly sampled.

ii. Signal Preprocessing (P):

Sensor data is initially pre-processed to remove erratic signals or artifacts usually by various transformation techniques by basic signal pre-processing methods to improve signal-to-noise ratio. Typically, sensor values are normalized or calibrated, along with other processing methods like analysis of discriminant, de-noising, etc. Sensor oriented methods can also be useful, like l_2-norm of accelerometer readings could be calculated when acceleration direction is irrelevant. Sensor values to could also be converted to other units (physical) (Roggen, et al., 2013).

iii. Segmentation (S):

This is a process which involves splitting continuous stream of data into smaller bits or segments, which are then further processed (classified). This addresses the fact that many applications need real-time processing wherein latency is bound. Most common or widely used approach is sliding or a jumping window approach, where a fixed window-size *w* is slide across the data samples. This technique can be used in various periodic movements (e.g., walking, biking, jumping), where signals are being processed in domain of frequency. In event of static movements, feature extraction's robustness is increased in judgment to sample by sample approach and this permits better robustness in consideration to noise. Energy or rest position-based segmentation is another approach, where isolated gestures are performed by user and between gestures, subject returns to rest position. Few more methods exist in context of time series examination, online learning, and hybrid segmentation-classification (Roggen, et al., 2013).

iv. Feature Extraction(F):

A feature is an individual measurable property of a phenomenon being observed and it "summarizes" the signal into a single (or multiple) value having more expressive power.
Features should be selected to enhances characteristics of the signal, thereby making it easier for pattern detection. This makes it simpler to differentiate multiple patterns of interests, for example, this increases class separation between activities. Prime focus while feature selection should be computational complexity and care must be given to select the least.

Various statistical and frequency domain features are used for activity recognition with low computational complexity. Most popular ones being, Mean of acceleration ~ inclination of a sensor, Variance of acceleration ~ how much movement, Standard deviation: square root of variance, Covariance: Joint variability of two variables, Median: most frequent value, Difference between min and max, skew, kurtosis etc.

Features extraction is implemented on identified segments to reduce their dimensionality by projecting a high dimensional feature space onto a lower dimensionality space while preserving "interesting" structures inside data distribution. Principal Component Analysis (PCA) is used to represent the signal in a lower dimensional space, e.g., for visualization and Linear discriminant analysis (LDA) is used to maximize class separation in a lower dimensional space, e.g., for classification.

Features could be automatically calculated or arrived manually by expert subject knowledge. By using minimum features that permits Recognition Chain to attain required target performance, we can improve system performance. More features mean additional training data for parameter estimation and also higher computational expenses for classification. Manual selection of such features is cumbersome and methods like wrapper,

filter, or hybrid approaches could be implemented. By choosing the "best" set of features, performance of learning models improve by overcoming the curse of dimensionality, improving generalization capability, speeding up learning process and refining model interpretability.

v. Classification(C):

Feature vector is mapped into a predefined set of output classes (activities and gestures) by a classifier, which is trained during the design time. *Support vector machines (SVM), decision trees, k-nearest neighbor, or Naive Bayes classifiers* are commonly used classifiers. Ranked likelihood of the output class is usually obtained and used for decision fusion, which is the next element. kNN is the most intuitive method, wherein unlabeled examples are classified according to similarity of examples in training dataset. Decision boundary will be mostly represented as sphere encompassing k points centered around the test point. Nearest centroid classifier (NCC) is a simple model, where simple class boundaries problems can be solved. It is most suited when classes are a cluster in feature space. Decision tree is a non-parametric supervised learning method, which is used for both classification as well as regression. This aims to create a model that could predict target values by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation. The deeper the tree, the more complex the decision rules would be, and this makes model more fit or better generalization. Main advantage is that little data preparation is required and techniques like data normalization, dummy variables creation and removal of blank values are not required. Naive Bayes is another supervised ML algorithm, based Bayes' theorem and having an assumption of conditional independence between every pair of features given the value of the class variable. These are extremely fast compared to other sophisticated methods, and it has worked quite well in many real-world situations as they require only a small amount of training data to estimate the necessary parameters.

vi. Decision Fusion:

This element combines information from several sources (multiple sensors or classifiers operating on a sensor) into a decision about an occurred activity. Various studies show that more sensors or classifiers (ensemble classifier) tends to improve the classification performance (Bulling, et al., 2014). The fusion stage pools numerous intermediate classification (mostly weaker) outputs into a single decision and this approach can occur at the beginning (feature level) as well as latter stages (classifier level). Commonly fusion approach in this area is summation, majority voting, Borda count, and Bayesian fusion (Bulling, et al., 2014). This approach could be used if the limits of the existing individual method are reached, and it is hard to develop a better one. Solution being combination of existing well performing methods in hoping for better results.

vii. Null-Class Rejection:

Usually, only a limited amounts of data stream is significant for activity recognition systems. Activities tends to be easily confused with other activities having related forms but are irrelevant, which is known as a NULL class. In certain situations, NULL class can be recognized if equivalent signal characteristics, like signal variance, change noticeably from preferred activities. By thresholding on raw feature values or using confidence scores computed by the classifier, it is possible to recognize the NULL class. System discards certain activities in which classification confidence is small, based on its likelihood. Here, outcome is the detection of action primitive with a likelihood p_i at a time t_i.

b) **System design** for a wrist-worn device for motorcycle riders, which should detect motorbike crash.

The proposed system contains a 3-axis accelerometer and gyroscope. Various studies have proved that a combination of accelerometer and gyroscope would provide the best performance in most fall detection scenarios (Huynh, et al., 2015), (Rakhman, et al., 2014). Accelerometer provides vital information during the impact stage by monitoring the body's inertial changes and body's rotational velocity during event of a motorbike crash is recorded by the gyroscope. Event or outcome of crash produces high acceleration and angular velocity, which makes it possible for the classifier algorithm to detected by CRASH. Although the main aim of the device is to detect crashes, sometimes it could falsely detect some human motions such as sitting down or lying down and may lead to false alarms. The main reason for choosing both sensors is that chances of occurring or measuring both peaks (for accelerometer and gyroscope) are rare considering normal daily routines. The data collected from the gyroscopes are used to rule out most of the non-fall motions and data from accelerometer is then used to make the final judgment or classify it as a CRASH. By utilizing the data from two kinds of motion sensors, the system can detect crash events effectively.

ACTIVITY RECOGNITION CHAIN:

i. Sensor sampling:

Streams of sensor data or samples 'S' is acquired using a 3-axis accelerometer and gyroscope. Each of the sensors is sampled at 50 Hz as studies show any further increase does not improve the performance significantly (Zurbuchen, et al., 2021).

ii. Signal Pre-processing (P):

Sensor data is pre-processed to filter out signal variability or artifacts by a low-pass filtering (LPF), which removes redundant information connected to involuntary human movements (e.g., tremor), which allows for a better classification. Error correction is also performed to eliminate any undesired values captured during measurement or preparation of time series dataset. normalization has proven to be useful for a better generalization of the sample dataset. Neither PCA or not LDA is required as these algorithms are computationally expensive and doesn't yield much improvement in performance.

iii. Segmentation (S):

Sliding window approach is implemented here as it is best suited for scenario involving time series data. The window size chosen for the system is 2 seconds to avoid false alarm as various studies imply interval of 1 to 2s are proven to provide best trade-off between recognition speed and accuracy (Banos, et al., 2014).

This allows us to determine the threshold during the event of crash by calculating magnitude:

$$AT_t = \sqrt{aX_t{}^2 + aY_t{}^2 + aZ_t{}^2}$$

Where, aX, aY and aZ are the linear acceleration along the X, Y and Z axis of accelerometer. Similarly, for the magnetometer :

$$GT_t = \sqrt{gX_t{}^2 + gY_t{}^2 + gZ_t{}^2}$$

iv. Feature Extraction(F):

Statistical features such as mean, variance, and kurtosis is used here considering the simplicity and high performance in plenty of recognition scenarios. Features could be automatically calculated or derived based on expert subject knowledge. Manually strategy (like PCA, LDA) is time consuming, hence we proceed with the automated strategy. Mutual information approach is used for finding the best features for the task at hand, where amount of information that feature as well as class shared is combinedly represented. Features under scrutiny are 1: mean, 2: standard deviation, 3: median, 4: mean crossing count, 5: zero crossing count, 6: kurtosis, 7: skewness, 8: RMS, 9: integral and 10: energy. Each of the above features are calculated and sorted to find the top 3. By choosing the "best" set of features, performance of learning models improve by overcoming the curse of dimensionality, improving generalization capability, speeding up learning process and refining model interpretability.

v. Classification(C):

Feature vector is mapped into a predefined set of output classes (event of crash, or normal riding) by a classifier, which is trained during the design time. Random forest classifier is implemented as this algorithm removes overfitting problem, which is common across Decision Tree algorithm. Most advantages of this classifier is that it automatically creates a list with the most discriminative features. It also has capability to create confidence intervals which indicate the certainty rate of a predicted class for each input data (Zurbuchen, et al., 2021).

vi. Decision Fusion:

This element is not required for the system as we intend to use a single classifier.

vii. Null-Class Rejection:

By thresholding on raw feature values and using confidence scores computed by the classifier, it is possible to recognize the NULL class. System discards certain activities in which classification confidence is small, based on its likelihood. Here, outcome is the detection of action primitive with a likelihood p_i at a time t_i.

Implementation:

The data gathered from sensors located (3DACC+GYRO) on the wrist are processed using a sliding window. A peak detection is performed, and if a peak is found, the data within the sliding window are analyzed to extract several features, which are ultimately classified as CRASH or NOT_CRASH. The Crash Detection block is performed with the ML classifier.

A framework for the system is shown in the Figure 1.

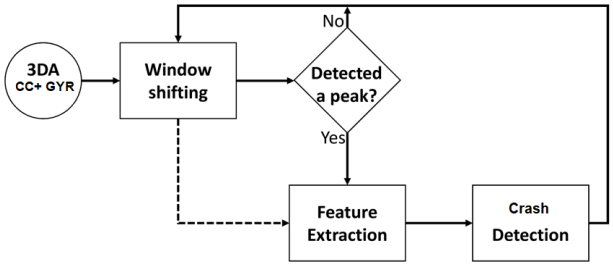


Fig 1: Schematics for CRASH detection system

c) Evaluation of an activity-aware product

Evaluation of an activity aware product is made with consideration of user aspect, performance on task at hand and on also based on cost, using ground truth labelled from reference dataset. User aspects considered might comprise of acceptance for user, comfort while wearing the product etc. Questionnaires and expert interview would shed some light into various subjective elements measured in the aspect of users (Roggen, et al., 2013).

Various performance metrics can be evaluated, considering scenario of continuous or isolated. In case of *isolated* (pre-segmented datastreams), accuracy is mostly preferred to evaluated performance. It is defined as number of correct recognitions (events) to number of events in total. Another useful metric is confusion matrix, which indicates class distribution of predictions for each activity, that is made by the recognition system. Here, diagonals of matrix contain fraction of activity instances recognized correctly and off-diagonal elements represent fraction of activities confused.

Performance must be evaluated in a continuous or online form in most real-world scenario, where the above metrics fails to capture the sense. In continuous activity recognition, the activities are sometimes (mostly) scattered in periods in which no relevant activity might be occurring (null class). Metrics defined above would be meaningless and for continuous activity recognition we should use metrics suited for the objective. *Precision* (true positives divided by positives), *recall* or sensitivity (true positives divided by true positives and false negatives) and *specificity* (true negatives divided by true negatives and false positives) can be obtained from the confusion matrix. Metrics proposed by (Ward, et al., 2011) includes ***'insertions'***: activities recognized, however in reality it didn't happened; ***'deletions'***: activities happened in reality however not recognized; ***'merges'***: various activities recognized as single; ***'fragmentations'***: single activity instance is recognized but as several instances and lastly, ***'overfill'*** *and* ***'underfill'*** (Roggen, et al., 2013). These metrics satisfactorily explain how duration of a specific activity is recognisable. A. demonstration of these is shown in the below illustrations.

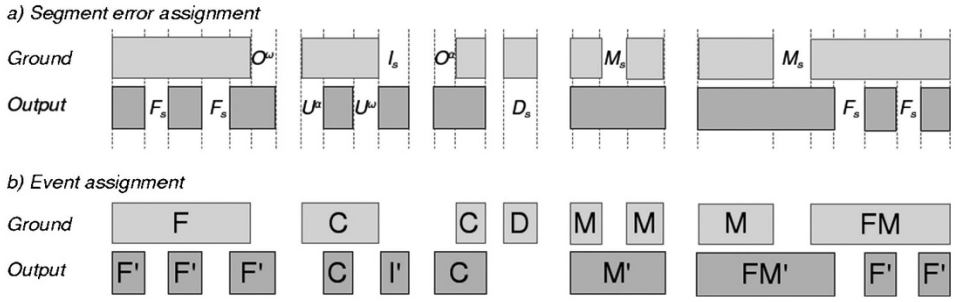


Fig. 2. Typical event anomalies found when comparing a ground truth with a (mock) recognition output: (a) shows the sequence divided into segments, with the FP_s and FN_s segments annotated as described in 3.2; (b) shows the same sequence with all of its ground and output events annotated with the event scores described in 3.4.

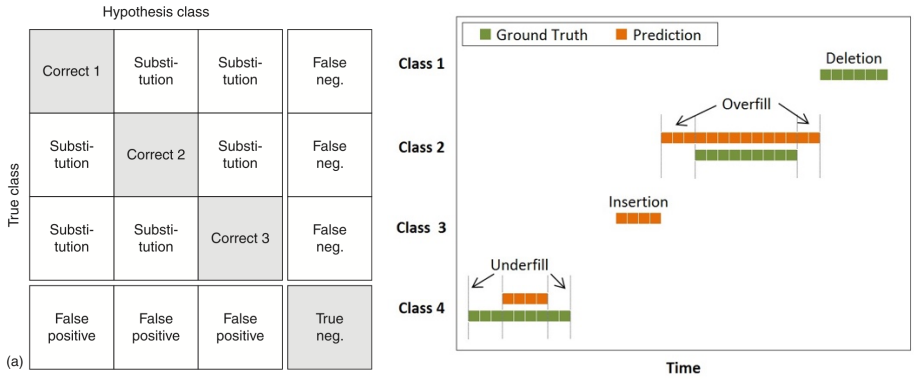


Fig 3: Performance metrics of activity recognition (a) Confusion matrix. (b) Continuous recognition performance (Roggen, et al., 2013).

Recognition latency is a significant criterion, usually applicable in wearable devices. As power consumption critically affects the size of battery, evaluation must involve energy consumed during recognition. It is important to reduce charging cycles specially for long-term monitoring (Van Laerhoven, et al., 2008). Last but not least, it can be evaluated with 'cost' as well as quality' metrics (Roggen, et al., 2013).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Leave-one-out cross validation

This method can adapt to assess generalisation to new situations, thereby making the system robust and adapted to multiple users (user-independent). Leave-one-out cross validation is applied with a specific situation left out to assess how the system generalises to that situation. For example, by *leaving a person out*, we can assess performance of the model on unseen users (making it user-independent) or *leaving a run out* (like day or week) to make the model adapt to behaviour of user over time (time independent). LOOCV is recommended to validate models built on smaller datasets where a standard test/train split may introduce significant bias into the model. If large variations are expected between participants, a LOOCV approach may be the best way to validate the model.

Performance Evaluation for Product 1:

Leave-one-person-out approach is used generalize recognition system to new users. Performance of the pipeline on the training dataset does not reflect the performance of the classifier "in the wild", i.e., when it is applied to new data. Hence, classifier needs to be evaluated on "new" data and several approaches exist for this purpose and among the lot, leave one out cross-validation seems to perform. Leave-one-out is the degenerate case of *K-Fold Cross Validation*, where K is chosen as the total number of examples for a dataset with N examples and then perform N experiments, where for each experiment we use N-1 examples for training and the remaining example for testing.

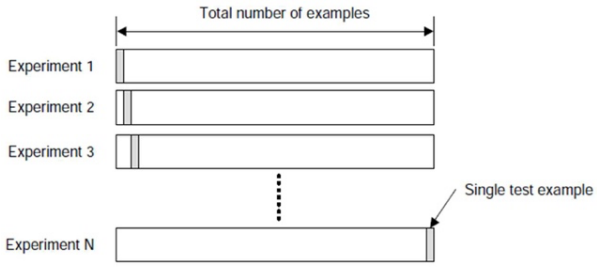


Fig 4: Leave-one-out illustration

Cross-validation technique is used to evaluate in what way recognition system would generalize new situations. Initially, obtained dataset is divided into folds. Here, all folds expect one helps in training classifier and left-out fold is utilized for testing. Process is iterated, until all folds were used once at least for testing. Database usually includes readings of many individuals, various days, several 'runs', leading to duplications of performed activity. While designing a user-independent recognition system, Leave-one-person-out is used to assess generalization to an unseen user. *Precision* and *recall* are calculated for each fold based on the results for the estimated class at each time slice and ***error rate*** is calculated from F-score (1.0 − F -measure). Smaller the error rate, better the performance of classification.

Performance Evaluation for product 2:

In a static system, machine learning techniques learn the mapping between signal patterns and context during the design time itself. System adaptation during calibration phase is performed by reconfiguring the activity recognition chain to achieve the desired recognition goal. The system needs to make sense of the user activity pattern, i.e., adapting to gait of the user to achieve the best performance. This translates into changing mappings between sensor signals and context. Slow changes in the mapping between sensor signals and context translates by a drift of the points corresponding to activity classes in the feature space (Roggen, et al., 2013). By monitoring drift over time, the classifier parameters may be adjusted accordingly, and adaptation can be done after the calibration phase taking advantage of many repetitive occurrences of gait of the user (context) in daily life.

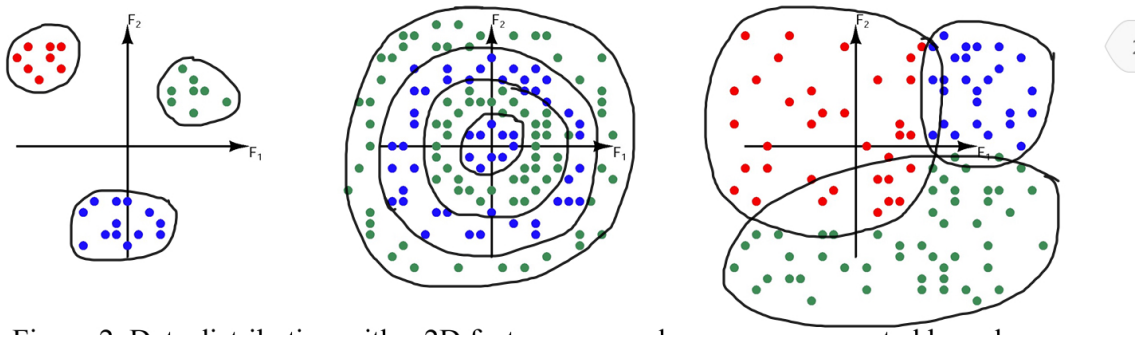
Activity Evaluation:

Leave-one-run-out method is applied to evaluate user-specific performance. Activity executions could adapt gradually over a period of time and leave-one-week-out is useful in evaluation of robustness over a period of time (Roggen, et al., 2013). Evaluation across multiple days, ensures robustness against natural daily fluctuations in gait of user. Device could be in calibration phase for a week and by combining all the data from user while wearing during the calibration phase, classifier keeps on updating on daily basis. Assuming the dataset has a separate class for persons with walking disability (e.g., Parkinson), we separate the disability class so that further studies can be performed. After filtering out the disabled class, we perform the split of test/ train on the disabled class. This is followed by filtering out one week's data (calibration phase) from all the subjects inside training set and Leave- One-Subject-Out Cross Validation (LOSOCV) is performed on rest of dataset. The calibration algorithm is deployed over each subject to self-adapt to gait of specific user during the calibration phase. This algorithm will tune the classifier as per daily patterns or trends detected (e.g., Freezing of gait).

By the end of calibration phase (1 week), the model will be fully capable of predicting the specific gait of the user as it would have completely adapted to the particular user.

d) Classifier selection

i. kNN

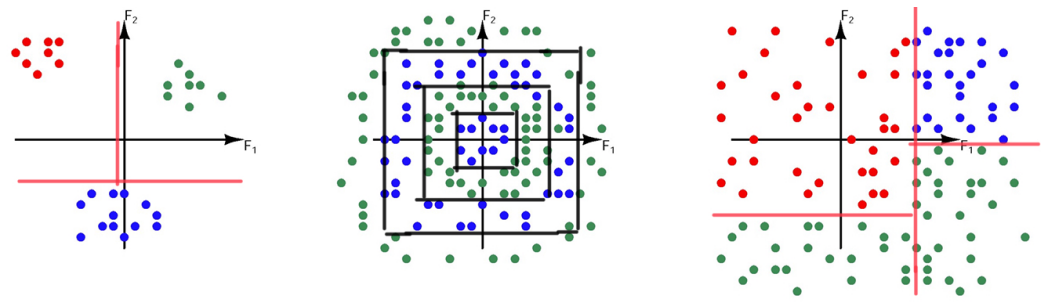


kNN classify classes to represented in a sphere encompassing k points centered around the test point. Nature of decision boundaries for kNN is shown above. This algorithm works really well if the datapoints are distinct or evenly clustered. In the first case, this works well in creating 3 distinct decision boundaries for each class and prediction will be accurate. Classification in second scenario is really poor as there is a lot of overlapping between classes thereby making the predictions poor. In case 3, classifier performs satisfactorily in creating boundaries for each class (even though a small overlap).

Advantages: This algorithm is simple to implement and is very intuitive to understand. This is nearly optimal in case of large sample limit (N→∞) and use local information that can constantly evolves with new data.

Disadvantages: Large storage requirements, computationally intensive recall and highly susceptible to the curse of dimensionality. Not suitable for overlapping classes.

ii. Decision Tree:

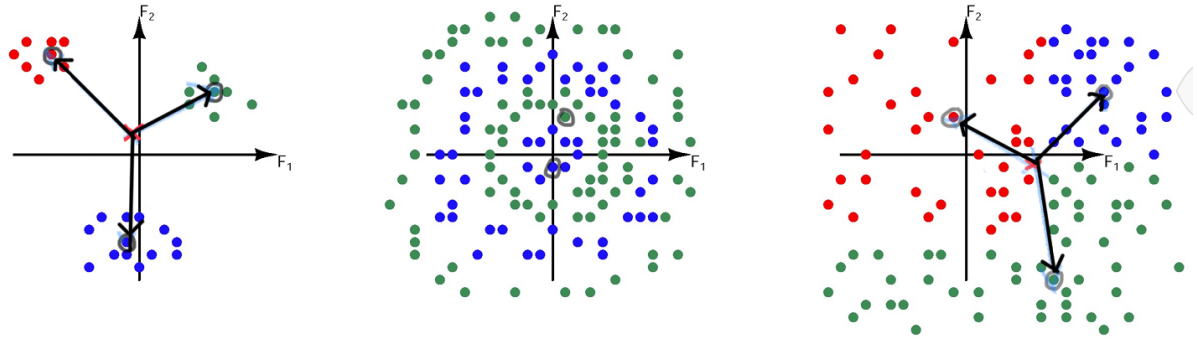


Nature of Decision tree classifier is shown above, which can be used for both classification as well as regression. This model classifies and predicts target values by learning simple decision rules inferred from the data features or class. The deeper the tree, the more complex the decision rules would be, and this makes model more fit for better generalization. In all the cases, classifier performs really well in discriminating the classes (even though a small overlap occurs in second scenario).

Advantages: This algorithm is useful for continuous as well as discrete values, i.e., it is suitable for both categorical and numerical data. It doesn't require scaling or normalizing.

Disadvantages: It often involves higher time to train the model, a small change in data can result in an overall change in the decision tree structure and there is a high probability of overfitting data.

iii. NCC

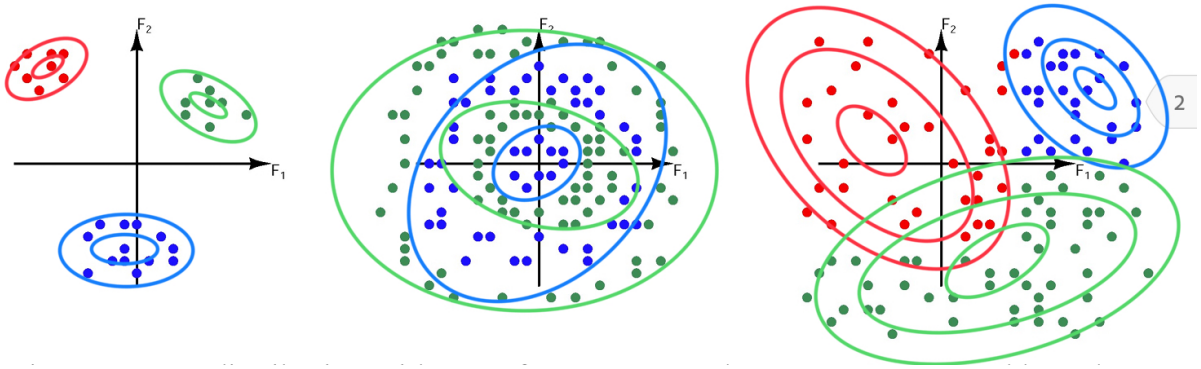


NCC is a linear estimator, simplest classification methods, which requires no parameters. It basically classifies to the nearest class center available. Decision boundaries are intersections of hyperspheres resulting in hyperplane decision boundaries and is useful for simple decision boundaries. This algorithm works well if the datapoints are distinctly clustered. In the first case, this works well in creating 3 class centres and prediction will be accurate. Classification as there no clear class centre. In case 3, classifier performs satisfactorily in creating class centres (even though a small overlap).

Advantages: This algorithm is very simple and easily implemented and is also quick (less memory requirement). In case of online model updating, adding and remove classes would not impact this classifier as (centroid) would adapt class centre.

Disadvantages: Suitable for simple class boundaries and only when classes cluster in the feature space.

iv. Naïve Bayes



Naïve Bayes is a probabilistic model, and the decision boundary tends to be in shape of ellipses as shown in the above figure. This algorithm works very fast and can easily predict the class

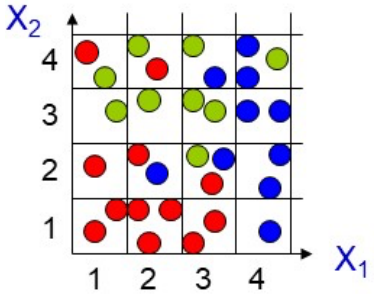
of a test dataset and can be used to solve multi-class prediction problems as shown. In all the cases, the classifier performs really well, only exception being the last one (where few outliers are noticeable).
Analysis of computational complexity can be performed using the table below,

N: Datapoints d: Dimension k: neighbours c: class

Sl No.	Classifier	Training time complexity	Training space complexity	Prediction time complexity	Prediction space complexity
1	kNN	$O(1)$	$O(Nd)$	$O(k*Nd)$	$O(1)$
2	Decision Tree	$O(N\log N*d)$	$O(N\log N*d)$	$O(N\log N*d)$	$O(N\log N*d)$
3	NCC	$O(1)$	$O(Nd)$	$O(k*Nd)$	$O(1)$
4	Naive Bayes	$O(Ndc)$	$O(dc)$	$O(dc)$	$O(dc)$

*Table: d.1, Computational Complexity of various classifiers (*considering asymptotic Big O)*

e) Feature selection



Mutual Information (MI) is the amount of information that two variables share. It is expressed as the amount of information provided by variable *X*, that could reduce amount of uncertainty in a variable *C*. In cases where random variables are statistically independent, MI value becomes zero. Generally, in classification problems, *C* is the class and *X* is the feature vector. MI between feature vector *X* and class label *C*, I(C;X) measures amount by which uncertainty in the class C is decreased by knowledge of the feature vector X:

$$I(X;C) = I(C;X) = \sum_{c,x} p(x,c)\log\frac{p(x,c)}{p(x)p(c)}$$

Assuming a feature vector X= [X_1 X_2 ... X_N] and each feature X_i is to be evaluated individually with the mutual information criteria that produce ranking of feature J(Xi):

$$J(X_i) = MI(X_i;C)$$

MI for the provided class is calculated as below:

$$I(C;X_1) = \sum_{c=1}^{3}\sum_{x=1}^{4} p(C=c,X_1=x)\log\frac{p(C=c,X_1=x)}{p(C=c)p(X_1=x)}$$

FeatureX_1 : *Joint pdf, **p** (C, X1):*

P(C,X1)	X1=1	X1=2	X1=3	X1=4	P(C)

C=1	4/31	5/31	3/31	0/31	12/31
C=2	2/31	2/31	4/31	1/31	9/31
C=3	0/31	1/31	2/31	7/31	10/31
P(X1)	6/31	8/31	9/31	8/31	

I(C;X1)	P(C,X1)	P(X1)	P(C)	P(C,X1)* log [P(C,X1) / P(X1)* P(C)]
(C=1,X1=1)	4/31	6/31	12/31	+0.1012
(C=1,X1=2)	5/31	8/31	12/31	+0.1115
(C=1,X1=3)	3/31	9/31	12/31	-0.0209
(C=1,X1=4)	0/31	8/31	12/31	0.0000
(C=2,X1=1)	2/31	6/31	9/31	+0.0129
(C=2,X1=2)	2/31	8/31	9/31	-0.0139
(C=2,X1=3)	4/31	9/31	9/31	+0.0793
(C=2,X1=4)	1/31	8/31	9/31	-0.0392
(C=2,X1=1)	0/31	6/31	10/31	0.0000
(C=2,X1=2)	1/31	8/31	10/31	-0.0441
(C=2,X1=3)	2/31	9/31	10/31	-0.0347
(C=2,X1=4)	7/31	8/31	10/31	+0.3251
				$I(C;X_1)$= 0.4771

Feature X_2: *Joint pdf p(C,X2):*

P(C,X2)	X2=1	X2=2	X2=3	X2=4	P(C)
C=1	7/31	3/31	0/31	2/31	12/31
C=2	0/31	1/31	4/31	4/31	9/31
C=3	1/31	4/31	2/31	3/31	10/31
P(X2)	8/31	8/31	6/31	9/31	

I(C;X2)	P(C,X2)	P(X2)	P(C)	P(C,X2)* log [P(C,X2) / P(X1=2)* P(C)]
C=1,X2=1	7/31	8/31	12/31	+0.2657
C=1,X2=2	3/31	8/31	12/31	-0.0044
C=1,X2=3	0/31	6/31	12/31	0.0000
C=1,X2=4	2/31	9/31	12/31	-0.0517
C=2,X2=1	0/31	8/31	9/31	0.0000
C=2,X2=2	1/31	8/31	9/31	-0.0392
C=2,X2=3	4/31	6/31	9/31	+0.1547
C=2,X2=4	4/31	9/31	9/31	+0.0793
C=3,X2=1	1/31	8/31	10/31	-0.0441
C=3,X2=2	4/31	8/31	10/31	+0.0816
C=3,X2=3	2/31	6/31	10/31	+0.0031
C=3,X2=4	3/31	9/31	10/31	+0.0046
				$I(C;X_2)$= 0.4495

The mutual information between the classes and feature X_1 is $I(C;X_1)$=**0. 4771**, which is higher than the mutual information between the classes and feature X_2 which is $I(C;X_2)$=**0. 4495**.
$J(X_1)$ = **0. 4771** and $J(X_2)$ = **0. 4495**
As $J(X_1)>J(X_2)$, we select feature X_1 as the most important feature.

f) Activity recognition chain implementation :

i.) Sensors:

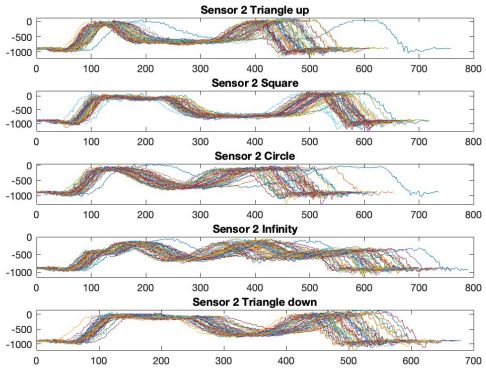


Fig a. Gestures (classes) from sensor 2

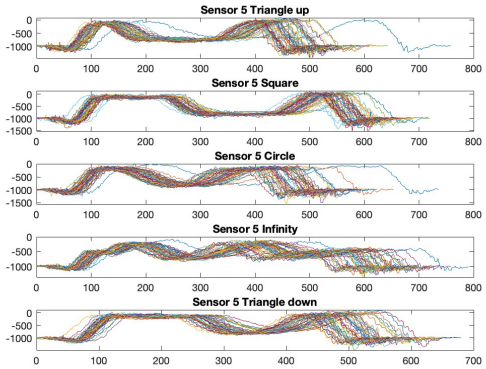


Fig b. Gestures (classes) from sensor 5

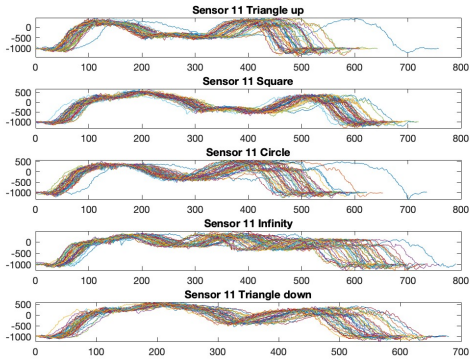


Fig c: Gestures (classes) from sensor 11

Gestures from various sensors were plotted and analyzed to recognize the best sensor which can clearly differentiate each activities. Each line in the plot represents one repetition (or instance) of the gesture. From figures above, it is clear that there exisit varation among each instanes (out of total 50), where a specific activity (class) was performed. After careful observation, three channels (2,5 and 11) were shortlisted as shown above, which appears to give visually distinct patterns. Amongst three, further we tune down to a single channel which would be optimal for recognition of activity.
In Fig 1, signals from sensor 2 (as well as sensor 5 ~ similar) is most recognizable and each activity (gestures) can be identified. While performing Triangle up, recognizable patterns occur between time intervals of 80 to 200 (gradually peaks and then plummets), 200 to 325 (remain steady) and then 325 to around 450 (peaks and plummets) to resting position. In case of square, 80 to 250 (gradually increases and remains steady), 250 to 300 (plummets), 300 to 425 (remain steady) and finally 425 to 550 (peaks and plummets) to resting position. Triangle down is quite opposite to scenario of triangle up.

Finding the best sensor makes a huge impact on the performance of the system as the patterns needs to be clearly identifiable by the classifier to predict activity being performed. By choosing best sensor, the error rates or false predictions of the system can be reduced

drastically. Since the pattern was distinct and clearly identifiable, it was sensible to proceed with a single sensor as combing multiple sensors would make the system more complex and further improvement in performance might not be noticeable.

Sensor	NCC	kNN	Decision Tree	Naïve Bayes
Sensor 2	**0.878788**	**0.878788**	**0.969697**	**0.969697**
sensor 5	0.840909	0.863636	0.946970	0.962121
sensor 11	0.878788	0.818182	0.863636	0.871212
sensor 20	0.878788	0.787879	0.840909	0.893939

Table F.1: shows accuracy for various sensors (considering features mean, standard deviation and skewness).

ii.) Pre-processing:

Two preprocessing techniques are performed and compared in search for best activity recognition system.

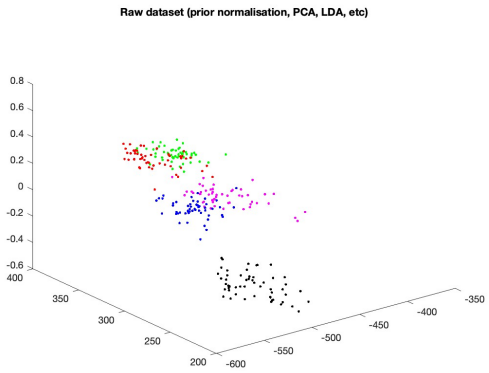


Fig a.

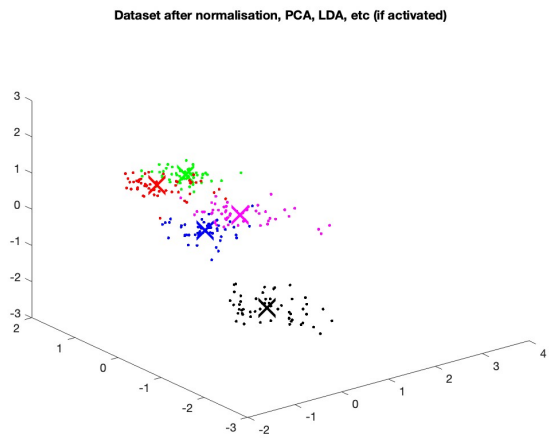


Fig. b

Normalization

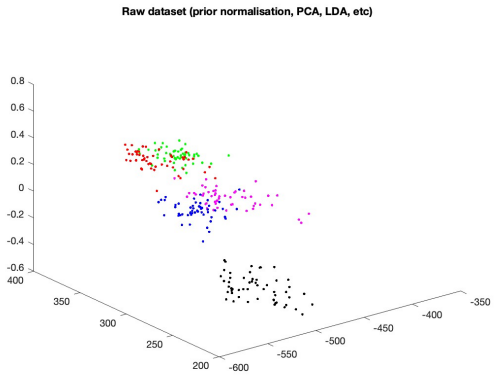
Preprocessing step of *normalization* is carried out first to understand the impact on performance. As seen from the above Fig b, the transformation of data is clear and more recognizable. The axis has become more uniform compared to Fig a. Normalization achieves data consistency within the database and a better execution as well.

Sensor	NCC	kNN	Decision Tree	Naïve Bayes
Sensor 2	**0.916667**	**0.969697**	**0.969697**	**0.969697**
sensor 5	0.916667	0.946970	0.946970	0.962121
sensor 11	0.863636	0.871212	0.863636	0.871212
sensor 20	0.886364	0.871212	0.840909	0.893939

Table F.2: shows accuracy for various sensors after normalization.

After performing normalization, the performance improves for NCC and kNN classifier and there is no effect in case of Decision tree and Naïve bayes classifier. Now, we study the effect of dimensionality reduction by performing PCA.

Dimensionality reduction with PCA



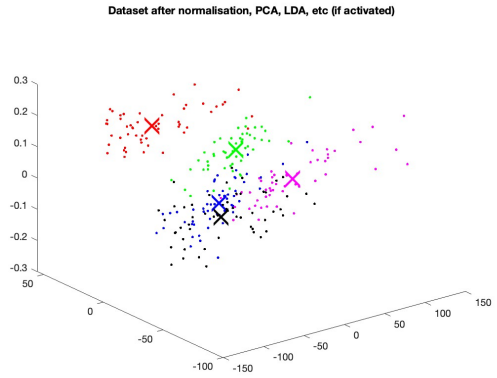


Fig a Fig b

Dimensionality reduction with PCA is carried out and as seen from the above Fig b, the transformation of data makes it more unrecognizable as compared to the model without performing any pre-processing. Thus, the outcome of PCA is not desirable as it would hinder the performance of activity recognition. We perform classifiers on this dataset to understand the impacts on accuracy more clearly.

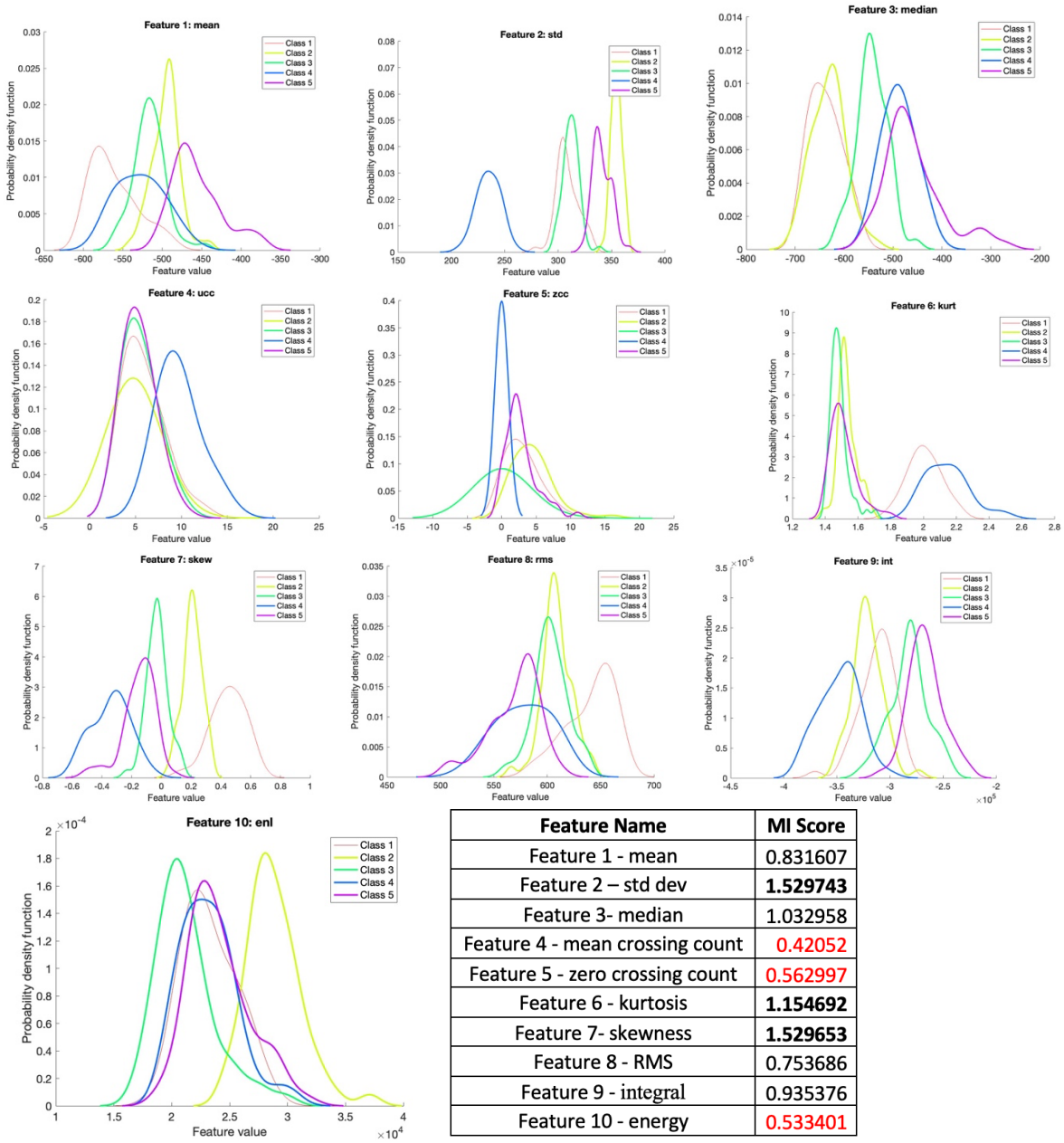
Sensor	NCC	kNN	Decision Tree	Naïve Bayes
Sensor 2	0.878788	0.878788	0.909091	0.939394
sensor 5	0.840909	0.863636	0.863636	0.909091
sensor 11	0.878788	0.818182	0.848485	0.878788
sensor 20	0.878788	0.787879	0.878788	0.909091

Table F.3: shows accuracy for various sensors after PCA.

Table above shows the performance of the system after performing PCA. It is evident that the performance of Decision Tree and Naïve Bayes classifiers are taking a hit, whereas there is no impact on NCC and kNN classifiers. Hence, neither PCA or not LDA is required as these algorithms are computationally expensive and doesn't yield much improvement in performance.

iii. Features:

There are manual and automated strategies to identify suitable features. Manually strategy is time consuming, hence we proceed with the automated strategy. Mutual information approach is used for finding the best features for the task at hand, where amount of information that feature as well as class shared is combinedly represented. Features under scrutiny are 1: mean, 2: standard deviation, 3:median, 4:mean crossing count, 5:zero crossing count, 6:kurtosis, 7:skewness, 8:RMS, 9:integral and 10:energy. Each of the above features are calculated and sorted to find the top 3. Various plots given below demonstrate how the best features are representative of activity classes being predicted. Below tables show the values obtained from MI method for feature selection and best scores are highlighted.



Feature Name	MI Score
Feature 1 - mean	0.831607
Feature 2 – std dev	**1.529743**
Feature 3- median	1.032958
Feature 4 - mean crossing count	0.42052
Feature 5 - zero crossing count	0.562997
Feature 6 - kurtosis	**1.154692**
Feature 7- skewness	**1.529653**
Feature 8 - RMS	0.753686
Feature 9 - integral	0.935376
Feature 10 - energy	0.533401

Table F.4: shows mutual information gain for various features.

From Table above, features ***2, 7 and 6 (***std-dev, kurtosis, and skewness) are having the top score; 5, 10 and 4 turned out to be the worst scores obtained for Mutual Information. Various plots given below demonstrate how the best features are representative of activity classes being predicted. Now, we combine the best three features and pass it to the classifiers for evaluating the performance. The combination of feature ***2,6 and 7*** has resulted in astonishing performance of 98.5% for the Naïve Bayes classifier and 97% for decision tree as shown in Table x below.

Classifier	Accuracy
NCC:	0.765152
kNN:	0.80303
DT:	0.969697
NB:	**0.984848**

Table F.5: shows accuracy of model after choosing top 3 MI features

iv. Classifier:

Most used classifiers are Naïve Bayes, Decision Tree, NCC and kNN for the activity recognition challenge and performance of each classifier with and without normalization is shown in below Table F.6. We have considered best features extracted from the previous step, i.e., *std-dev, kurtosis, and skewness* for developing this model. The decision boundary of all the classifiers is also show below and it is evident that Naïve Bayes classifier has performed the best in recognizing activities with 99% accuracy. All the five classes are clearly visible and is distinct, making it the best activity recognition system.

Features: 2+7+6		
Classifier	**Accuracy with Normalization**	**Accuracy Before Norm**
NCC:	0.977273	0.765152
kNN:	0.977273	0.856061
DT:	0.977273	0.977273
NB:	**0.984848**	**0.984848**

Table F.6: shows accuracy of various machine learning classifiers

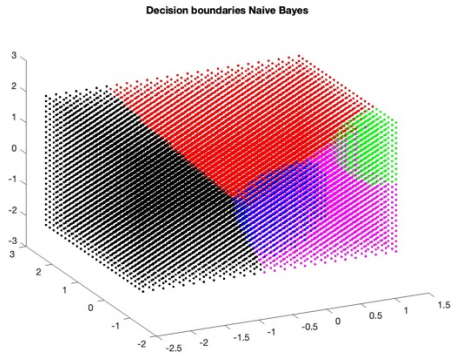


Fig. 1

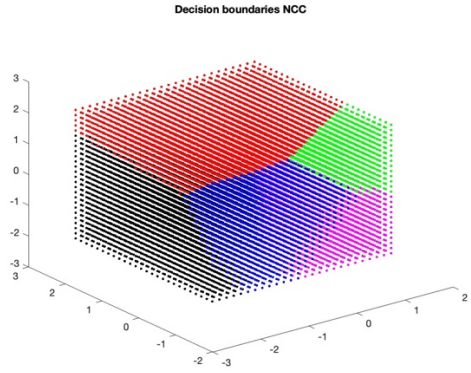


Fig. 2

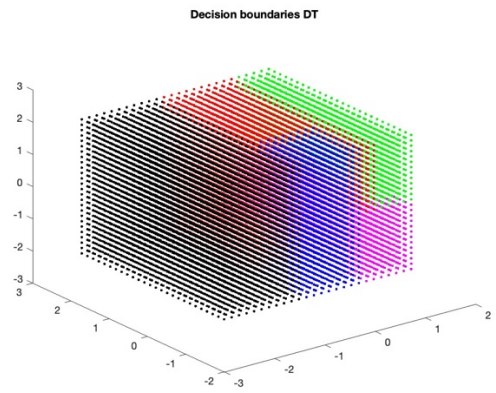


Fig. 3

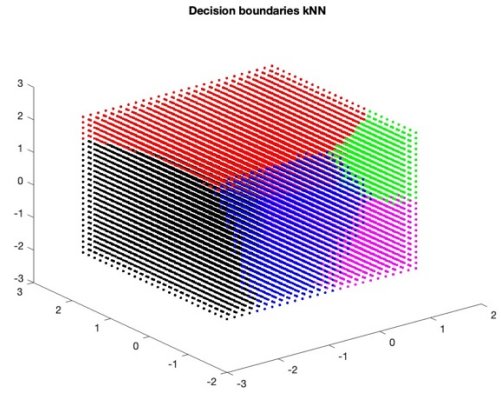


Fig. 4

This task of activity recognition is coming under the sweet spot of Naïve Bayes classifier. This algorithm works very fast and can easily predict the class of a test dataset, and here we had 5 classes to predict. Another advantage is that NB performs better than other models with less

training data and if the assumption of independence of features holds good, which is the exact scenario here.

Decision tree can be considered as the second best with 98% accuracy. As seen from table above, it doesn't require scaling or normalizing the data. Continuous and discrete values, i.e., it is suitable for both categorical and numerical data (which is the case here).

NCC is suited only when classes cluster in the feature space and can detect only simple class. This is the reason why, after normalization, the performance of this classifier has improved substantially (from 77% to 97%). The biggest advantage of kNN classifier is the use of local information, which can constantly evolve with new data (highly adaptive). This classifier performed good with 86% prior to normalization and after norm, accuracy improved to 98%.

v. Computational complexity:

Computational complexity during training stage for the Naive Bayes classifier used is $O(Nd)$, as it needs to compute the frequency of every feature value di for each class and in testing stage, Naive Bayes is $O\ (cd)$ since we must retrieve d feature values for each of the c classes.

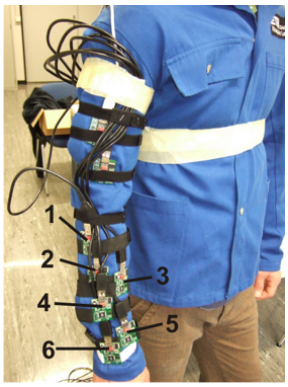


Fig v: Acceleration sensors (USB) placed on the right lower arm (Förster09)

Based on the experimental data and analysis, the "BEST" or recommended activity recognition chain is discussed in this section. Sensors are in the order: 3, 4, 18, 19, 20, 27, 29 and 31as per description. From above experiments, we found sensor 2 (which is sensor 4 as shown in above figure) as the best performing sensor among the lot and could be solely used for the gesture (activity) recognition task. The trends or patterns are very distinct for this sensor and that is the primary reason to choose this sensor alone and not look for any other combinations. Also, from a customer-oriented product point-of-view, there is no sense to include multiple sensor location, if a single sensor can provide best accuracy (in our case, accuracy was close to 99%). Now, coming to pre-processing, normalization has proven to be useful for a better generalization of the sample dataset. Neither PCA or not LDA is required as these algorithms are computationally expensive and doesn't yield much improvement in performance. Main part of the challenge is feature selection and considering high number of features (10), it is advisable to go for an automatic feature selection method like Mutual Information score to figure out the best features (at-least top 3) among the lot. Features found most accurate are std-dev, kurtosis, and skewness and using these, we develop the best performance model by train the classifier with these features. Naïve Bayes is one of the most used classifiers for categorical input variables, and the algorithm performs exceptionally well in comparison to numerical variables. Also, this can be used to solve multi-class prediction problems, like the activity recognition.

In summary, Sensor = sensor 2; Pre-processing = Normalization; Feature selection = combination of std-dev, kurtosis, and skewness; Classifier = Naïve Bayes.

Bibliography

Bächlin, M. et al., 2010. A wearable system to assist walking of Parkinson s disease patients. *Methods of information in medicine,* 49(01), pp. 88--95.

Banos, O. et al., 2014. Window size impact in human activity recognition. *Sensors,* 14(4), pp. 6474--6499.

Bulling, A., Blanke, U. & Schiele, B., 2014. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR),* 46(3), pp. 1--33.

Franke, T., Pieringer, C. & Lukowicz, P., 2011. *How should a wearable rowing trainer look like? A user study.* s.l., s.n., pp. 15--18.

Gemperle, F. et al., 1998. *Design for wearability.* s.l., s.n., pp. 116--122.

Huynh, Q. T. et al., 2015. Optimization of an accelerometer and gyroscope-based fall detection algorithm. *Journal of Sensors,* Volume 2015.

Kallio, S., Kela, J., Korpipää, P. & Mäntyjärvi, J., 2006. User independent gesture interaction for small handheld devices. *International Journal of Pattern Recognition and Artificial Intelligence,* 20(04), pp. 505--524.

Rakhman, A. Z., Nugroho, L. E. & others, 2014. *Fall detection system using accelerometer and gyroscope based on smartphone.* s.l., s.n., pp. 99--104.

Roggen, D., Magnenat, S., Waibel, M. & Tröster, G., 2011. Designing and sharing activity recognition systems across platforms: methods from wearable computing. *IEEE Robotics and Automation Magazine,* Volume 12, pp. 83--95.

Roggen, D., Tröster, G. & Bulling, A., 2013. Signal processing technologies for activity-aware smart textiles. *Multidisciplinary know-how for smart-textiles developers*, pp. 329--365.

Van Laerhoven, K., Kilian, D. & Schiele, B., 2008. *Using rhythm awareness in long-term activity recognition.* s.l., s.n., pp. 63--66.

Villalonga, C. et al., 2009. *Bringing quality of context into wearable human activity recognition systems.* s.l., s.n., pp. 164--173.

Ward, J. A., Lukowicz, P. & Gellersen, H. W., 2011. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology (TIST),* 2(1), pp. 1--23.

Zurbuchen, N., Wilde, A. & Bruegger, P., 2021. A machine learning multi-class approach for fall detection systems based on wearable sensors with a study on sampling rates selection. *Sensors,* 21(3), p. 938.