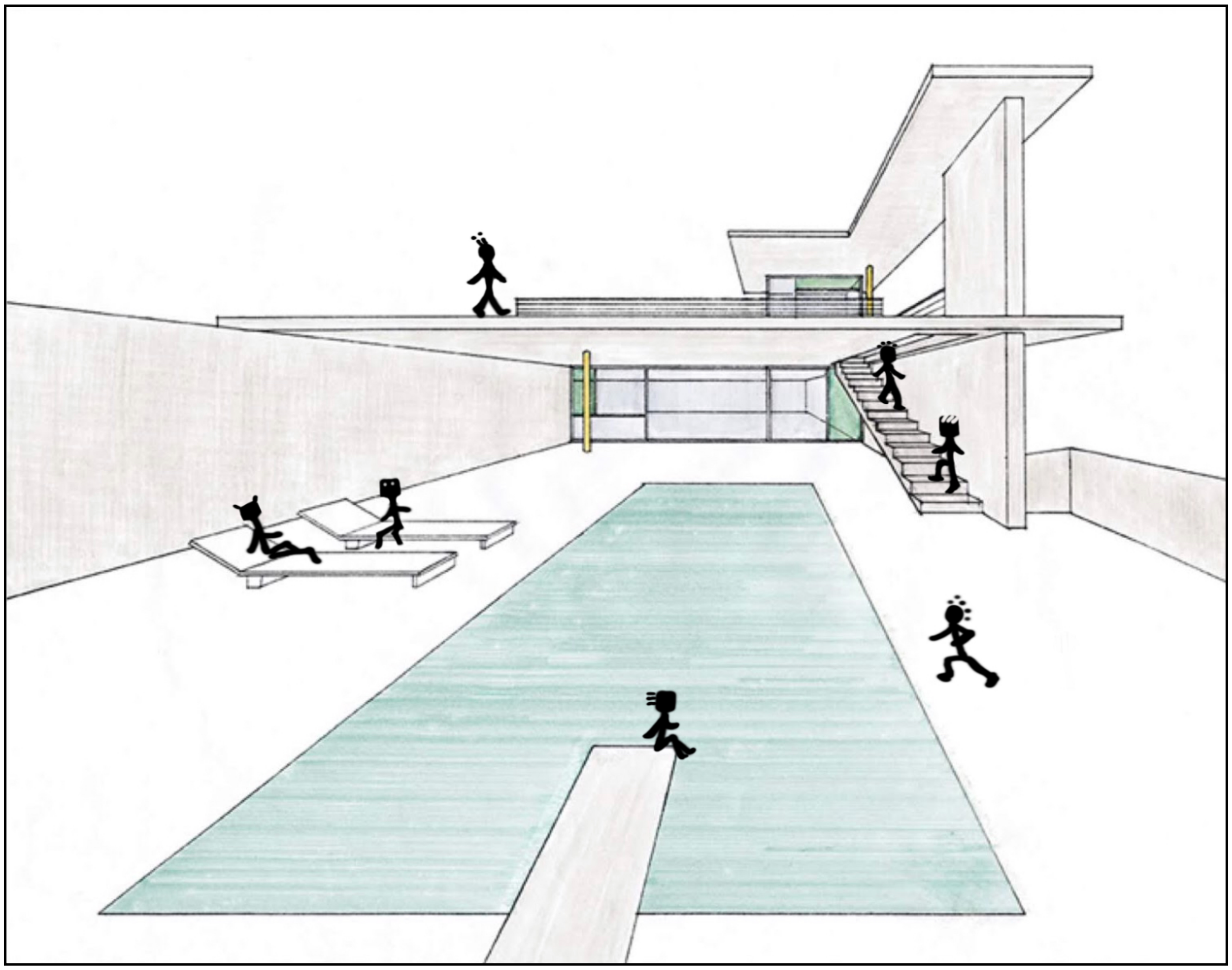
HUMAN ACTIVITY RECOGNITION USING SMARTPHONES SENSOR DATA



A Capstone Project

Prepared for: Udacity Machine Learning Engineer Nanodegree Program

Written by: Shan Sun, London, UK

1 July 2018

I. DEFINITION

1.1 PROJECT OVERVIEW

The availability of affordable wearable equipments and portable computing devices results in behemoth amount of data being collected including motion, location, physiological signals and environmental information. The human activity recognition (HAR) is an active research field to understand how human behaviours are developed by interpreting attributes derived from this data. "*This field is the first component (sensing) of the sequence for achieving Smarter Interactive Cognitive Environments together with data analysis, decision making and taking action, and our subject of research.*"[1] HAR is especially appealing to healthcare applications, such as caring and monitoring for elderly people. For example, physiological signals could be an indicator of a health condition via a change in heart rate and body temperature. We also consider accurately solving HAR problems can provide some indirect assistance to solve the motion sickness problem in VR by classifying human movements accurately and assisting real time re-calibration adjustments in VR visions.

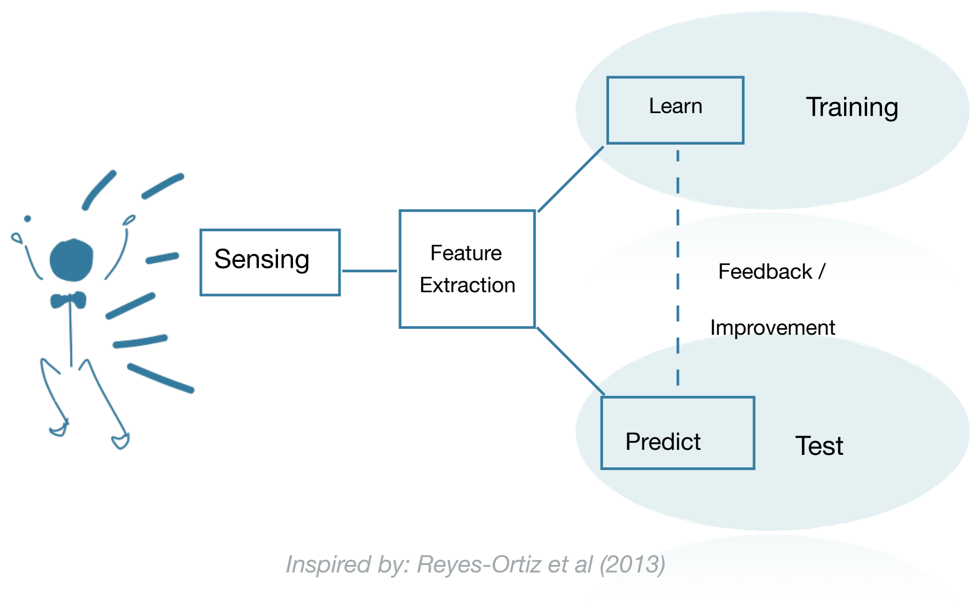
1.2 PROBLEM STATEMENT

As stated above, this paper addresses the HAR problem. The availability of affordable wearable devices, such as smart phones, enables mass data collection from users thanks to the embedded inertial sensors. Sensor signals from accelerometer, heart rate monitors, thermometers and gyroscopes are physiological signals data collected from these devices attached to human bodies. With the availability of data, analyses can be done to better recognise, classify, cluster and predict what human activities are carried out (e.g. stand, sit, lie, walk) for further decision making. Several papers already exploited this area, using supervised and semi-supervised learning methods such as support vector machine, ensemble methods with boosting and, deep learning techniques such as artificial neural network[1][2]. The diagram below summarises the process pipeline for HAR.

As a first step to tackle this problem, this paper explores a readily available dataset from UCI[1]. The dataset was built from the recordings of 30 subjects performing basic activities and postural transitions while carrying a waist-mounted smartphone with embedded inertial sensors. This paper used both a support vector classifier and a convolutional neural network to classify 12 activities included in the dataset before applying the algorithms to a separate testing set to predict the activities. The diagram below illustrates the process pipeline to tackle this problem.

[1] Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set from University of California, Irving. The dataset can be accessed here. http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions#

HAR Process Pipeline



Inspired by: Reyes-Ortiz et al (2013)

1.3 METRICS

With the dataset ready, the HAR problem essentially becomes a classification problem with labelled data under supervised learning. This paper addresses the HAR problem by applying two types of machine learning techniques: supervised learning using several classifiers (e.g. Random Forest, Gradient Boosting, Support Vector Machines, and Gaussian Naive Bayes). A Convoluted Neural Network (CNN) was also used to corroborate the results from the other classifiers. For each of the proposed algorithm, the accuracy score is used as the common metrics to compare model performance.

This paper also looked at the confusion matrix which allows clear algorithm representation whilst identifying error types. Since the HAR problem can be considered as a classification problem in machine learning, a confusion matrix is particularly helpful to evaluate the models used. Specifically, precision represents the number of correctly classified activities out of all the activities being classified; whilst recall represents each type of activity in truth how many the machine correctly classified. In summary, accuracy score coupled with a confusion matrix gives a more robust assessment of the model performances.

II. ANALYSIS

2.1 DATA EXPLORATION

The dataset used for this paper is from the UCI Machine Learning Repository titled "Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set" (SBHAR). The data was collected from experiments with a group of 30 volunteers aged between 19 to 48 years wearing a Samsun Galaxy SII on the waist. Volunteers performed a protocol of activities including six basic postures: three static - standing, sitting, lying and three dynamic - walking, walking downstairs and walking upstairs. The experiment also included transitional postures between static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie and lie-to-stand. The table below summarise the SBHAR dataset structure:

SBHAR DATA SUMMARY

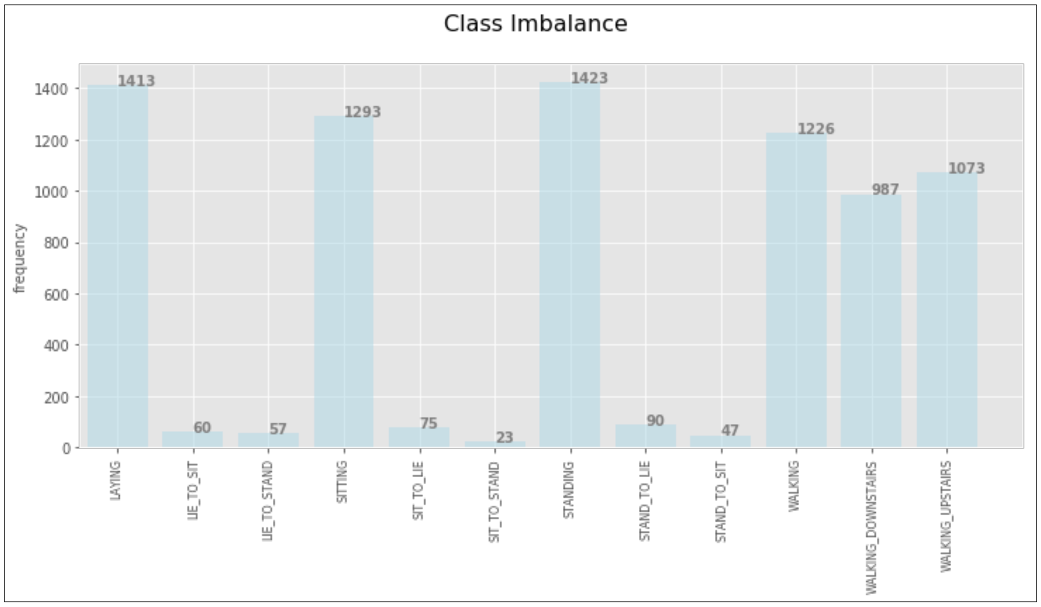
Index	Signal Variables	Functions	Labels	Activity Labels
1	tBodyAcc-XYZ	mean()	**1**	WALKING
2	tGravityAcc-XYZ	std()	**2**	WALKING_UPSTAIRS
3	tBodyAccJerk-XYZ	mad()	**3**	WALKING_DOWNSTAIRS
4	tBodyGyro-XYZ	max()	**4**	SITTING
5	tBodyGyroJerk-XYZ	min()	**5**	STANDING
6	fBodyAcc-XYZ	sma()	**6**	LAYING
7	fBodyAccJerk-XYZ	energy()	**7**	STAND_TO_SIT
8	fBodyGyro-XYZ	iqr()	**8**	SIT_TO_STAND
9	tBodyAccMag	entropy()	**9**	SIT_TO_LIE
10	tGravityAccMag	arCoeff()	**10**	LIE_TO_SIT
11	tBodyAccJerkMag	correlation()	**11**	STAND_TO_LIE
12	tBodyGyroMag	maxInds()	**12**	LIE_TO_STAND
13	tBodyGyroJerkMag	meanFreq()		
14	fBodyAccMag	skewness()		
15	fBodyAccJerkMag	kurtosis()		
16	fBodyGyroMag	bandEnergy()		
17	fBodyGyroJerkMag	angle()		

Features are selected from eight 3-axial raw signal variables and nine single component variables passing through 17 functions resulting in 8*3*17 +(17-8)*17 = 561 features for each example.

The sensor signals were collected from the accelerometer and gyroscope in the smartphones including 3-axial linear acceleration and 3-axial angular velocity at a constant refresh rate of 50Hz. Manually labelled data was also added to the video footage of this experiment [1]. After a series of preprocessing, the final dataset has 561-feature vector per example derived from 17 action patterns and 17 functions over 12 activities labels.

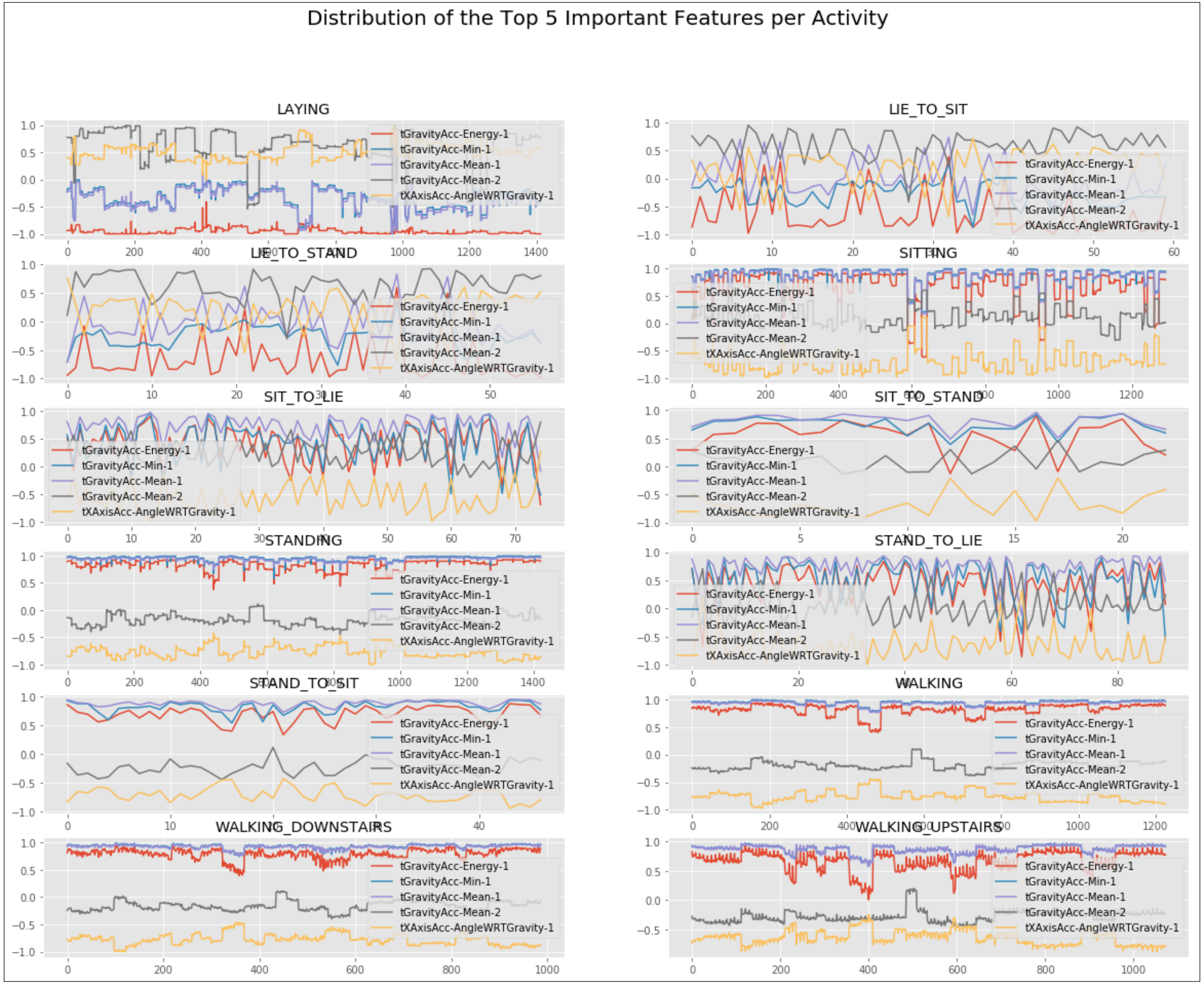
2.2 EXPLORATORY VISUALISATION

There are 10929 data points in the full dataset. To help the learner model to learn, the SBHAR dataset was randomly split into training and testing data by the ratio of 70:30. The resultant training set has 7767 data points whilst the testing set has 3162. To understand the dataset better, we looked at the number of activities recorded per each of the 12 activity groups. The dataset has an imbalance issue with transiting activities being under represented in comparison to static and dynamic activities. The figure below provides a clear picture with the number of observations for each type of activity displayed on top of the bars:



It is clear from the diagram static and dynamic activities have much more observation points (typically over a thousand) than transitional activities (much less under a hundred). The dataset therefore suffers from class imbalance. Class imbalance could present difficulties in accurately classifying and predicting data. Datasets with a disproportionate ratio of observations in each class may jeopardise the legitimacy of using accuracy as the metric for performance measurement. It is because a high accuracy score could be attributed to sampling from the over represented class but very limited data from the under represented class was captured. To address this problem, several methods could be considered: oversampling under represented data class; under sampling over represented data class; using different algorithms or metrics that lessen the reliance of accuracy score only or padding in data to augment under represented data class.

Another observation on the dataset is the numerous amount of features, 561 in total, as discussed in the section above. High dimensionality of data is often coined as the 'curse of dimensionality' in machine learning as it could potentially complicate the learning process a learner algorithm can learn. As an exploratory exercise, we looked the feature importance used an extra tree classifier to estimate the rankings of the 561 features. Based on the importance score, we plotted the important features for each activity and its distribution within the dataset below.



The most important five features selected by the extra tree classifier are:

FEATURE SELECTION (TOP 5)

Ranking	Features	Attribute Scores
1	tGravityAcc-Energy-1	0.031462
2	tGravityAcc-Min-1	0.029561
3	tGravityAcc-Mean-1	0.027234
4	tGravityAcc-Mean-2	0.022124
5	tXAxisAcc-AngleWRTGravity-1	0.021163

To further understand the magnitude of the dimensionality, a principal component analysis (PCA) is used. Deep learning is also considered a possible solution to high dimensionality[2], a convolutional neural network is therefore also applied to address this problem. The section below details algorithms and techniques used to tackle both imbalance and high dimensionality problems observed in the dataset.

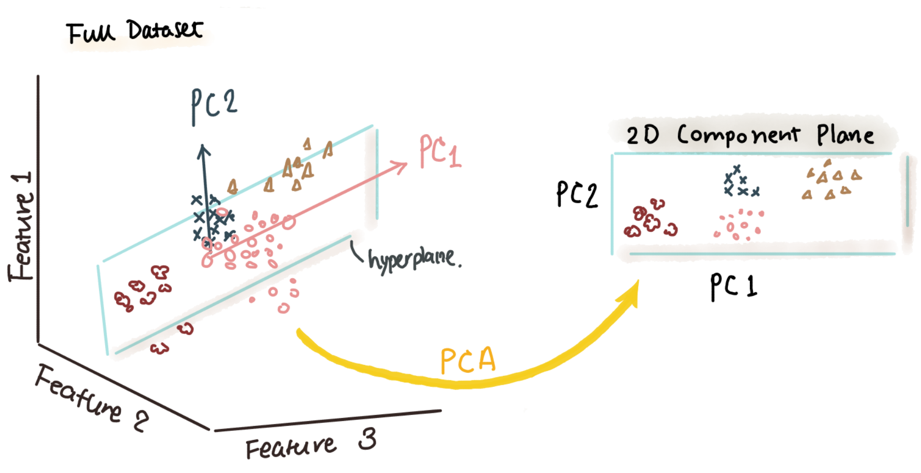
2.3 ALGORITHMS AND TECHNIQUES

2.3.1 Principal Components Analysis

As discussed in the section above, the SBHAR data appears to have class imbalance and high dimensionality problems. To address the high dimensionality problem, the PCA was applied to attempt to reduce dimensionality and select features. PCA works to simplify the complexity in high dimensional data whilst retaining trends and patterns within a dataset. It is similar to clustering by finding patterns without reference to prior knowledge about whether the samples within a dataset is coming from different classes. The main idea behind PCA is to geometrically projecting high dimension data onto lower dimensions (i.e. principal components or PCs) aiming to find the best conclusion of data patterns with only limited PCs instead of every feature in the dataset. The first PC is chosen to minimise the total distance between the data and its projection onto the PC, i.e. it minimises the perpendicular distance between data and the PC, instead of minimising the distance between independent and dependent variables, as the case in a linear regression. Under the hood, PCA maximises the variance of the projected points to capture any pattern displayed within the dataset. The diagram below illustrates how PCA works:

[2] See this paper that explains further on how deep learning algorithms could potentially deal with a high dimensional dataset. http://www.iro.umontreal.ca/~lisa/pointeurs/bengio+lecun_chapter2007.pdf

Illustration of a Simple 2D PCA



For a supervised learning classification problem such as the one presented in the SBHAR dataset, two types of algorithms are considered following the PCA: a Support Vector Classifier (SVC) and an artificial neural network. Below provides a short explanation on each of these algorithms.

2.3.2 Support Vector Classifier (SVC)

SVC essentially is a type of supervised learning models for classification and regression analysis. It works to map each data point within a dataset to separate categories so there is a clear gap between these categories as wide as possible. Data is predicted to belong to a specific category based on which side of the gap it falls into. Specifically SVC constructs a (or several) hyperplane(s) to be used for classification of data. Intuitively the hyperplane that creates the largest distance to the nearest training data point of any class is the best classification. This is also known as 'margin maximisation'. The diagram to the right indicates the margin maximisation in a linear data space scenario. In this case, the hyperplane for separating data is just a line. In higher dimension, there maybe several hyperplanes.

Illustration of a Simple Linear SVC

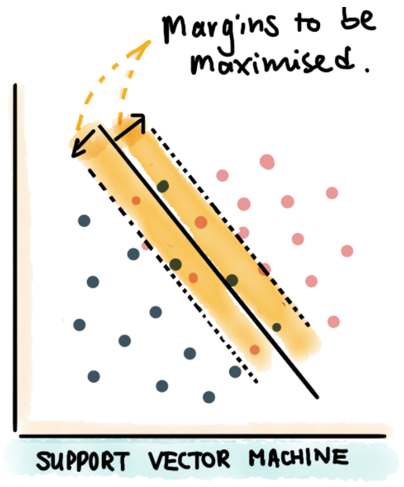
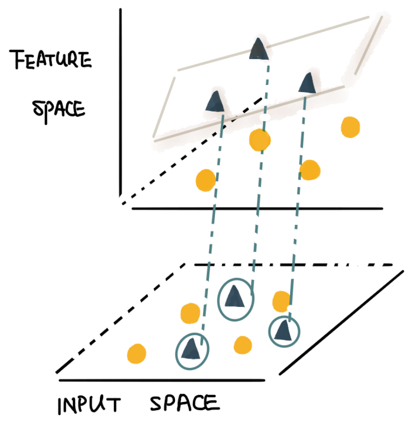


Illustration of Kernel Trick



For non linear classification, there is a kernel trick to be deployed in SVC. Intuitively, the kernel trick projects hard to separate data into different dimensions where they could potentially be divided by a hyperplane. To manipulate data matrices, the kernel trick calculates the dot products of features by projecting data points to the non linear space directly and thus finding a complex non linear boundary separate classes. The diagram to the left illustrates the intuition behind the kernel trick.

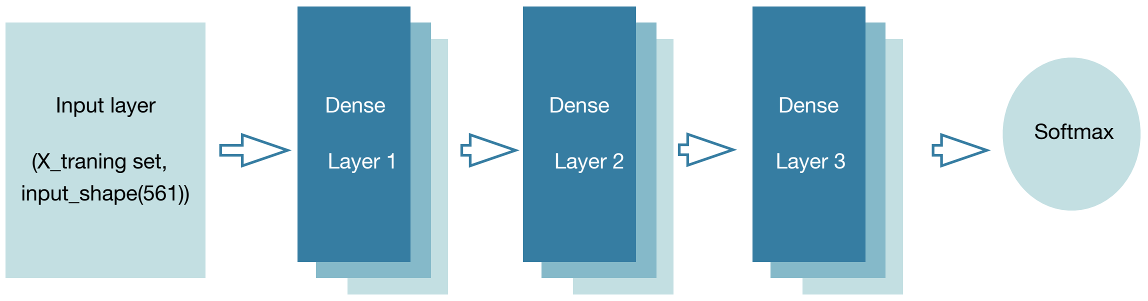
Mathematically, the kernel trick is used to maximise the following function[3]:

$$\begin{aligned}\text{maximize} \quad & \mathcal{L}_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i^T \mathbf{x}_j} \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \ \forall i\end{aligned}$$

Where Xi and Xj are any pair of input, a represents the weights for the training samples y. The The expression labelled in a red box is very costly to compute but the kernel trick allows a kernel function **K**(Xi, Xj) to replace the costly function, making the computation much faster. The kernel K can be linear, polynomial, or rbf etc.

2.3.2 Convolutional Neural Network (CNN)

As discussed previously, given the high dimensionality nature of the dataset, an CNN model was also applied. The CNN essentially has three types of layers, an input layer taking in data, some hidden layers determined by activities of the input units and weights on the connections between input and hidden layers, and an output layer churning out result of the classification. The CNN was designed to have several dense layers and dropout layers before fully connecting to the activation function set as softmax. The diagram below illustrates the CNN structure.



[3] Equation was taken from http://www.chioka.in/, The equation is the dual version solution of a minimisation problem modelled by a Lagrange primal problem statement.

2.4 BENCHMARK

As mentioned above, for each of the proposed algorithm, the accuracy score is used as the common metrics to compare model performance. We also looked at the confusion matrix which allows clear algorithm representation whilst identifying error types. A confusion matrix is particularly helpful to evaluate the models used. The accuracy score represents the number of correctly predicted class over the total number of data points. Specifically, it is the percentage of correctly predicted activity type over the total sample size. Confusion matrix, on the hand, provides a clearer picture on what types errors are at present.

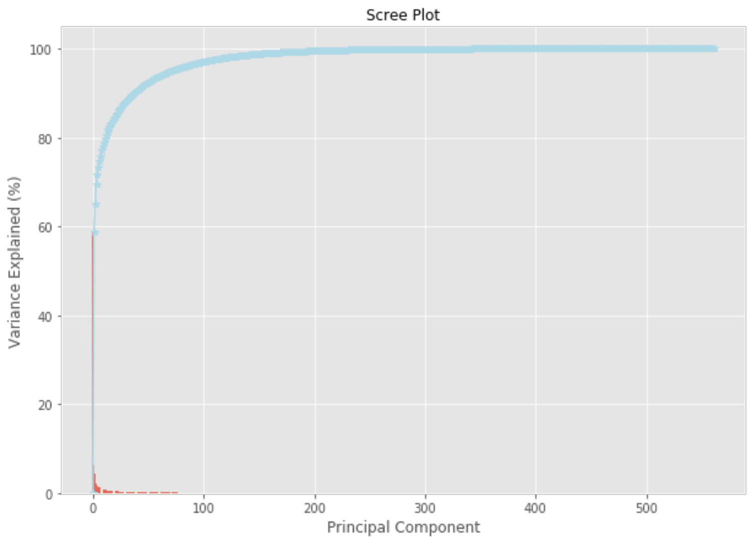
III. METHODOLOGY

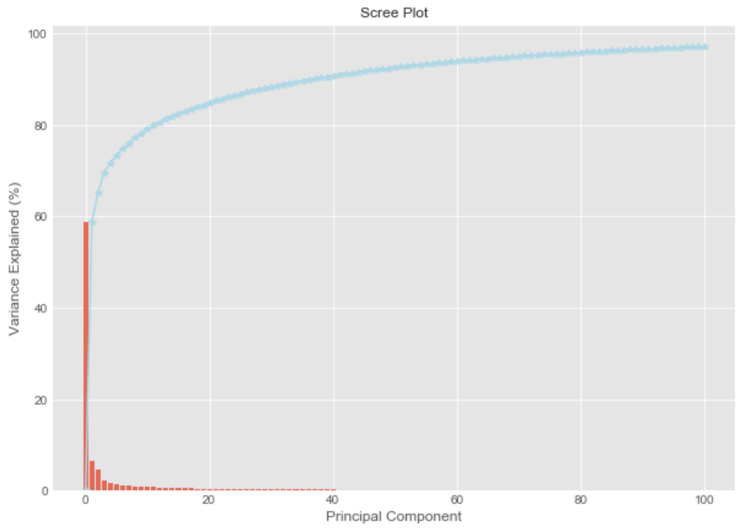
3.1 DATA PREPROCESSING

The dataset used in this paper had already been pre-processed. In particular, the SBHAR data generated around 5-hours of experimental data [3], and was also pre-processed with noise filters sampled by fixed-width sliding windows (2.56 sec with 50% overlap, i.e.128 readings/window). Other data transformations were also applied including the calculation of Jerk signals from time, body linear acceleration and angular velocity information; magnitude using Euclidean norm and, the frequency domain signals using Fast Fourier Transform (FFT). The preprocessing carried out appears reasonable and justified. A survey of similar studies within the HAR research field also indicates the preprocessing procedures carried were acceptable. Detailed of the literature review was included in Project Proposal.

Next, based on the pre-treated dataset, we looked at how to perform dimension reduction from the 561 features. One common technique to visualise the relationship between the number of principal components and the percentage of variance explained by the principal components is to use a scree plot. This paper first drew the scree plot to look at the entire dataset. It appears up to 90% of the variance could be explained by the first 100 principal components. A second scree plot was then drawn to zoom in on the first 100 principal components to have a closer look at the relationship. The diagram overleaf presents the two scree plots.

A closer look at the scree plot shows that the first three principal components explains almost 70% of variances within the dataset whilst the first 100 principal components explains up to 97%. It means potentially the 561 dimensions could be reduced using feature selection and limit to a smaller subset than the entire dataset.





To further determine how best to perform dimension reduction and feature selection, we obtained the optimal number of principal components by calculating the number of principal components that achieved the highest accuracy score at 103. Applying SVC on the first 103 principal components was then used as the benchmark model for further refinement of the model prediction. The result of using the 103 principal components is listed below.

Benchmark Model	**PCA 103**
Accuracy on training data	0.983
Accuracy on testing data	0.613

Given the accuracy on testing data is much lower than that on the training data, the algorithm appeared to suffer from overfitting. One way to deal with overfitting is to increase the size of dataset used for training. Applying deep neural network could also help reduce overfitting given the dataset used would be augmented to full. Alternatively some form of regularisation to penalise overfitting may also be helpful to alleviate the problem.

3.2 IMPLEMENTATION

A number of steps were carried out to perform the classification task using different models described above. These are summarised below for ease of reference:

1) Several classifiers (gradient boosting, support vector, random forest, k-neighbors and GaussianNB) were applied to the training set; Accuracy scores were obtained;

2) SVC was selected for further fine turning;

3) Grid search was carried out over several hyper parameters to obtain the optimal parameter combination before applying SVC again using the optimal parameters to get the best accuracy score on the training and testing set;

4) A simple CNN structure was applied to the training set;

5) Manually fine tuned several parameters for the CNN (number of episodes, batch size, dropout rates, density etc) to achieve better accuracy scores.

Below describes the details under each step. Firstly, a number of classifiers that potentially would be suitable to the present classification problem was also applied to compare the accuracy score achievable without any fine turning of the hyper parameters. In particular, the following classifiers are used, together with the accuracy scores each achieved individually:

SELECTION OF CLASSIFIERS

Classifier	Accuracy Score
Gradient Boosting (GBC)	0.9197
Support Vector (SVC)	0.9184
Random Forest	0.8889
K-Neighbors	0.8855
GaussianNB	0.7473

Gradient boosting classifier (GBC) and the support vector classifier (SVC) appeared to achieve the highest accuracy scores without any fine tuning. However given the dataset has imbalanced class problem, only SVC was explored further. This is because SVC has a built in feature to penalise mistakes on the minority class proportionally to how under represented the class within the dataset is. In addition, SVC also allows Platt scaling, wherein first the SVC is trained, then a separate cross-validated logistic regression is trained to map the SVC outputs into probabilities of each class within the dataset enabling the calculation of ROC-AUC score[4] to address class imbalance.

The coding difficulties I encountered are largely related to the fine tuning process whenever arbitrary judgment is required. For example, when performing a grid search for fine tuning hyper parameters, what kind of range for which parameters to parse to the model is a harsh trial on the engineer's understanding of the algorithm, the problem itself and the domain context, in addition to the assessment of the trade off between time and performance. Another example is on designing a suitable neural network structure, the permutation is endless and requires an engineer's industry experience as well as creativity.

3.3 REFINEMENT

Further to discussions above, SVC with class balancing was selected for implementation. The implementation was carried out by fine tuning hyper parameters of the SVC model using grid search. Specifically, we looked at the following three hyper parameters: kernel, penalty C and gamma. Below we briefly discussed the ranges used to select the best set of hyper parameters in turn:

- Kernel is defaulted to rbf in the Scikit-Learn package, linear kernel was added in providing a linear extrapolation on decision boundary. However, given the dataset has high dimensionality, this kernel was not expected to be selected as the optimal hyper parameter.
- Penalty parameter C punishes a misclassified data and acts as a regularisation tool for high bias and low variance data, if set small, and vice versa, if set large. It affects the SVC model by specifying the extent we could tolerate misclassification of each class in the dataset. Parameter C in the range of 1, 10, 100 and 1000 were hence given to grid search.
- Gamma is the hyper parameter that affects the class boundary of non linear hyperplanes. Lower gamma provides a more smooth separation between class boundaries whilst higher gamma may lead to the overfitting problem by creating 'island' boundaries for each data point. Given we already observed potential overfitting problem using 103 principal components in SVC, gamma was set to the lower end at 0.001 and 0.0001.

The final set of parameters selected by grid search was {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'} and was applied to the SVC model over the entire training dataset to further overcome the class imbalance and overfitting issue. The SVC model was improved as a result, with accuracy scores jumped up to 0.995 for the training set and 0.954 for the testing set. It indicates the fine tuning worked to alleviate the overfitting problem presented in the benchmark model.

[4] Nonetheless accuracy score with confusion matrix using SVC with class balancing and CNN seemed to address this problem sufficiently, ROC-AUC score was thus not calculated in the end.

For the CNN model, the model structure, loss function and optimizers were all fine tuned several times to achieve better model performance. The original design of the CNN model had one input layer, two dense layers with one time drop out in between. Adding an additional dense layer with further drop out appeared to have helped with the accuracy score. In addition, instead of using the categorical cross entropy as the loss function, binary cross entropy appeared to improve the model performance. It may be due to the sparsity of the prediction matrices. Finally, a number of optimizers were explored including Adam, Adamax, Adadelta, Adagrad and Rmsprop. Adam appeared to generate the best accuracy score. The final CNN has one input layer, three dense layers with three times drop out before fully connected to the activation function softmax. The model minimises the loss function binary cross entropy with the optimizer Adam, achieving an accuracy score on the training dataset at 0.997 and 0.991 on the testing dataset.

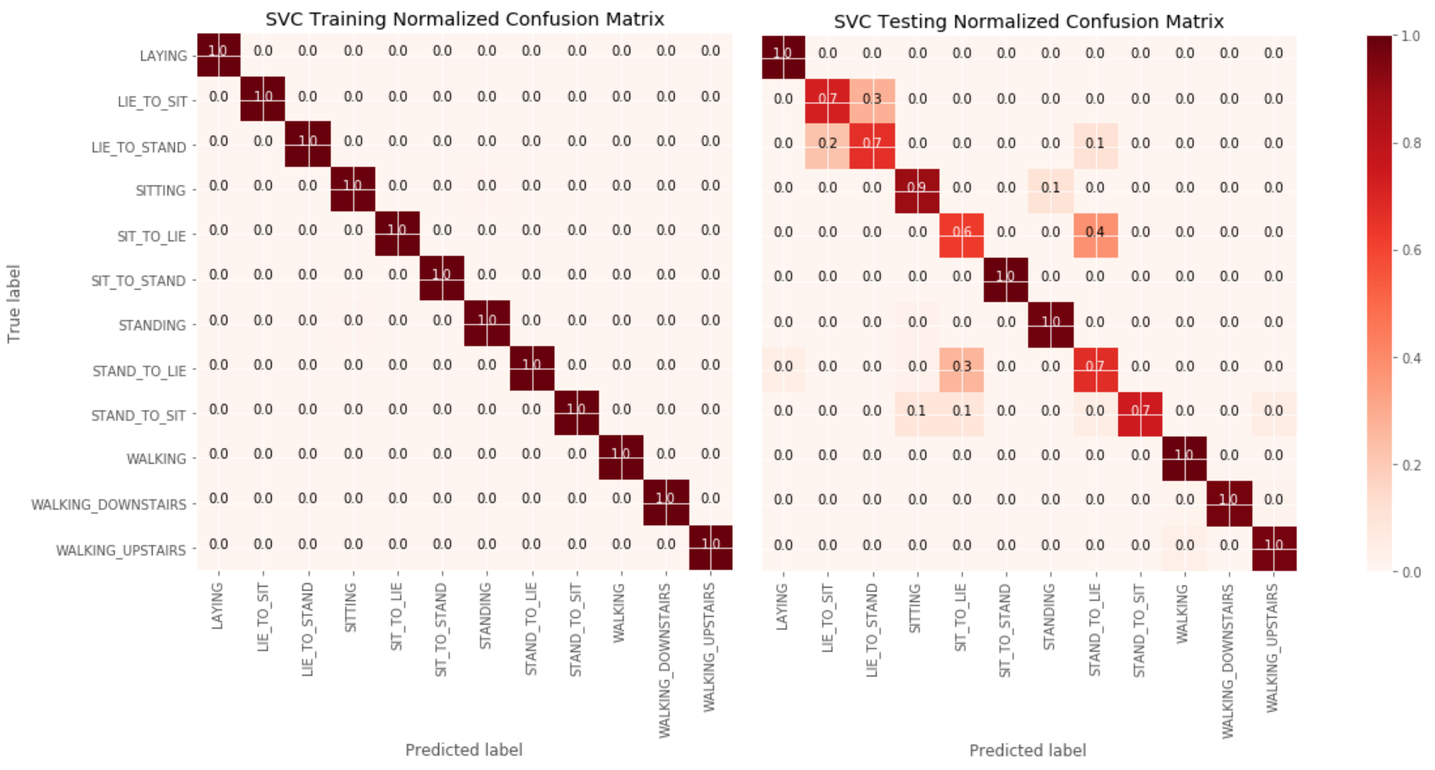
IV. RESULTS

4.1 MODEL EVALUATION AND VALIDATION

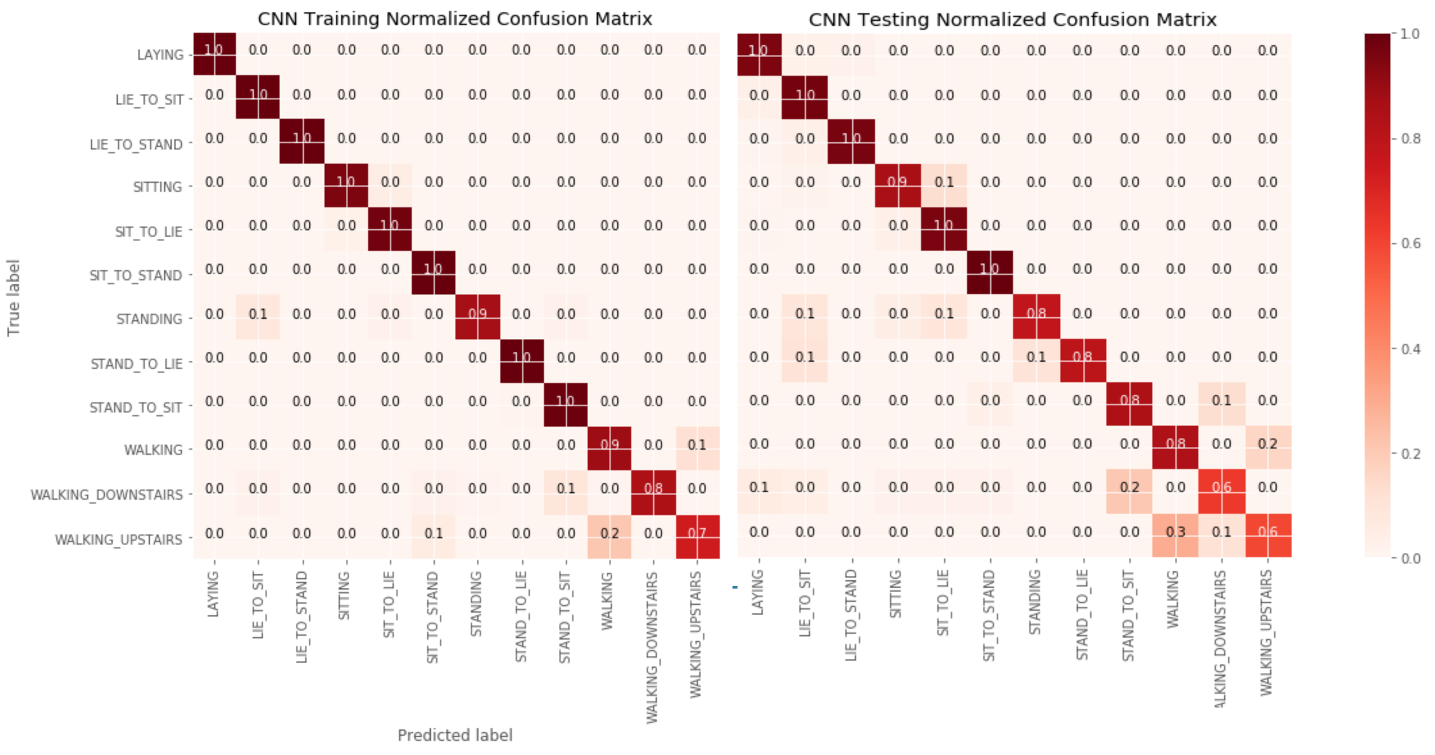
We discussed earlier that due to class imbalance presented in the data, a more robust methodology is to examine accuracy together with a confusion matrix to understand where the errors were coming from. This paper specifically looked at two types of models to predict HAR, a fine-tuned SVC and a fined tuned CNN. Below are the confusion matrices for each model on the training and testing datasets respectively. Note the confusion matrices are normalised for ease of comparison.

First are the comparison of the confusion matrices on the SVC model performance of the training and testing data.

The SVC confusion matrix on the testing data shows most errors were around misclassifying transitioning activities such as lie to sit and lie to stand, as well as stand to lie and sit to lie. These are understandable as the motion and movement between the transiting states can be too subtle to detect.



Next two confusion matrices present the CNN model performance on training and testing data respectively. Given the CNN model performed better than SVC model on both data sets respectively, fewer errors were expected. Interestingly, CNN made mistakes on primarily classifying walking and walking upstairs, as well as walking downstairs and stand to sit. Reasons behind these errors are slightly less clear intuitively. It maybe a common disadvantage of neural network models for being in a 'black box' situation providing limited insight into the reason behind the result of the model.



4.2 JUSTIFICATION

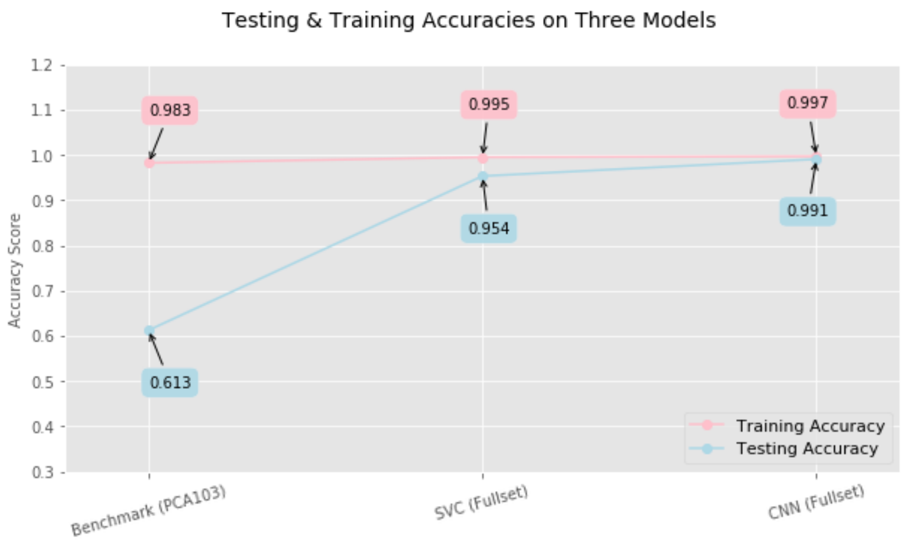
Both the fine tuned SVC model and the CNN applied over the entire training dataset outperformed the benchmark model limiting the training dataset to the first 103 principal components. There is little surprise given the models were performed over larger dataset with optimised hyper parameters to achieve high accuracy scores. In addition, since both SVC and CNN models achieved over 99% accuracy on the testing datasets, it is considered as the appropriate solutions to address the HAR problem adequately based on the dataset provided.

However, another important aspect looking into the machine learning result is to discuss its robustness. Particular, the model used should be able to generalize well and be less sensitive to outliers and small changes in data. One way to assess the model robustness is to use cross validation on training and testing data. Using 10-fold cross validation, the SVC model achieved an accuracy score of 0.945 on the training set and 0.938 on the testing set. The model performance dropped a little but was still better than the benchmark model and the models without any fine tuning. We therefore consider the models used here are reliable and adequately addressed the HAR problem.

V. CONCLUSION

5.1 RESULT VISUALISATION

After various steps of fine tuning, both SVC and CNN achieved good accuracy scores on both the full training set and the testing set. The diagram below illustrates the results with comparison of the two models with the benchmark model.



5.2 REFLECTION

Same as all my previous projects in the machine learning engineer course from Udacity, I enjoyed this project very much. I appreciate how much work was put in by the researchers to sanitise and preprocess the dataset, enabling me to focus purely on applying machine learning techniques and focusing on improving the model performance. However, I am fully aware the fact that such well treated dataset is rare in practice. As a machine learning engineer, I am prepared to spend a large chunk of my time to find, tackle, tidy, and explore real life data. Another difficulty which often is at present in an applied machine learning scenario is to define a problem accurately. It directly affects the type of algorithms a machine learning engineer may deploy, be it a supervised or unsupervised learning model, or a deep learning neural network. Again, this project eliminated this difficulty by providing labeled data and defining the problem as a discrete classification problem, enabling me to focus on optimising model performance.

5.3 IMPROVEMENT

Given my reflection stated above, one aspect of improvement would be to further fine tune the grid search function to improve the model performance. However, given both models almost achieved 100 percent accuracy on the testing data, another improvement could be instead noting the time each model spent on performing the classification and prediction. The time cost of an algorithm can be very important in assessing whether the model would be able to scale the application to a wider use in reality. Applying both models using the raw dataset could also be explored to see whether the preprocessing procedures have significant impact on the model performance. In addition, given the errors were predominantly present in transitioning activities, partitioning the static and dynamic activities from the transitioning activities may also be helpful to eliminate the class imbalance problem. If no significant improvement were found after the partition, it might mean further sensor devices or data points maybe required to delineate different transitioning activities from each other.

REFERENCES

[1] Jorge Luis Reyes-Ortiz, Alessandro Ghio, Xavier Parra-Llanas, Davide Anguita, Joan Cabestany, Andreu Català. Human Activity and Motion Disorder Recognition: Towards Smarter Interactive Cognitive Environments. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

[2] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. A Public Domain Dataset for Human Activity Recognition Using Smartphones. 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013. Bruges, Belgium 24-26 April 2013.

[3] Jorge-Luis Reyes-Ortiz, Luca Oneto, Alessandro Ghio, Albert SamÃ¡, Davide Anguita and Xavier Parra. Human Activity Recognition on Smartphones With Awareness of Basic Activities and Postural Transitions. Artificial Neural Networks and Machine Learning â€“ ICANN 2014. Lecture Notes in Computer Science. Springer. 2014.

[4] Andrea Mannini and Angelo Maria Sabatini. Machine learning methods for classifying human physical activity from on-body accelerometers. Sensors, 10(2):1154–1175, 2010

[5] Najafi B., Aminian K., Paraschiv-Ionescu A., Loew F., Büla C., Robert P. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. IEEE Trans. Biomed. Eng. 2003;50:711–723.

[6] Lee S.W., Mase K. Activity and location recognition using wearable sensors. IEEE Perv. Comput. 2002;1:24–32.

[7] Allen F.R., Ambikairajah E., Lovell N.H., Celler B.G. Classification of a known sequence of motions and postures from accelerometry data using adapted Gaussian mixture models. Physiol. Meas. 2006;27:935–951.

[8] Mantyjarvi J., Himberg J., Seppanen T. Recognizing human motion with multiple acceleration sensors. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics; Tucson, AZ, USA. October 7–10, 2001; pp. 747–752.

[9] Reyes-Ortiz, Jorge Luis et al. “Transition-Aware Human Activity Recognition Using Smartphones.” Neurocomputing 171 (2016): 754-767.