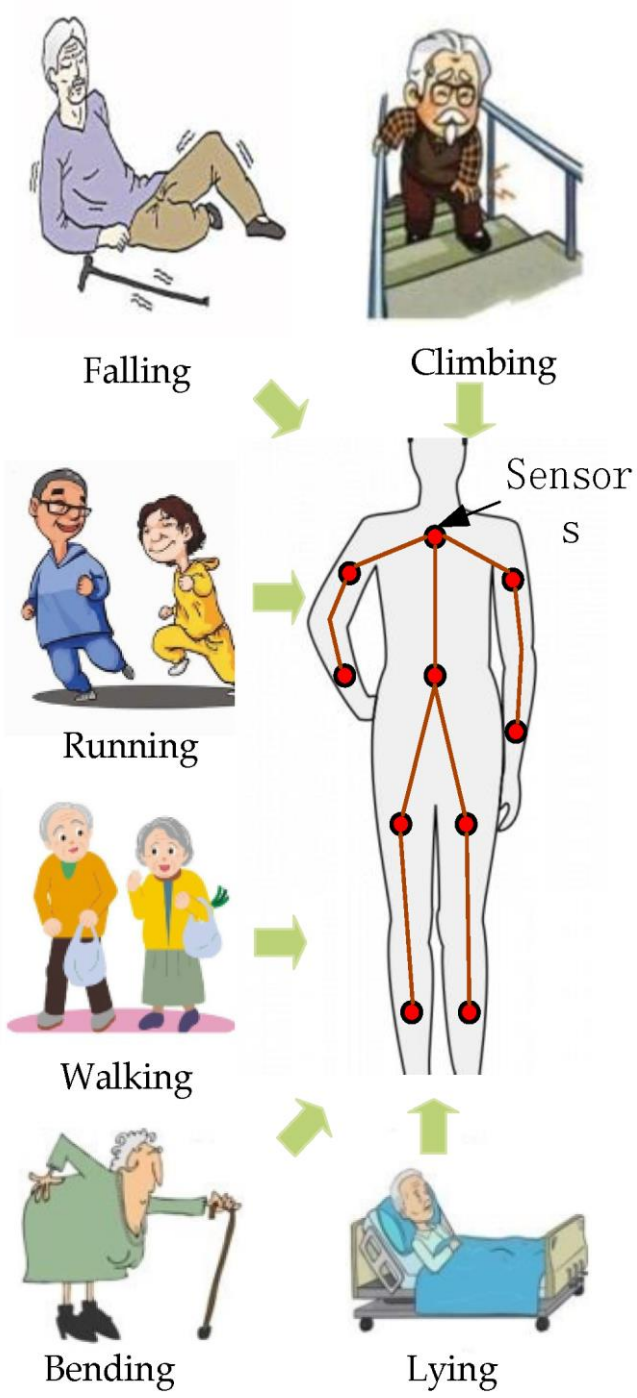
POSTURE RECOGNITION USING BODY LOCATION SENSORS

FINAL PROJECT REPORT



Contents

List of Figures

List of Tables

1. INTRODUCTION

Classification of the body posture based on the location of body sensors was important part of the Project CONFIDENCE - Ubiquitous Care System to Support Independent Living [1]. The main goal of this European project was to ensure the that the elderly people can live independently with the minimal support from the working age population [2]. The primary concern for the independent living of elderly people is to provide the immediate medical support in case of emergency like falling or fainting. In this context, Kaluža et al. [3] developed a multi-agent system for the care of elderly people living at home on their own. The architecture is based on 7 groups of agents communicating among themselves for detection and raising alarms for emergency situation. Out of these 7 agents, Interpretation group of agents is responsible for figuring out if a situation is emergency or not. In this group machine learning agents detect emergency situations based on models induced with machine learning.

These machine learning models uses several algorithms for the detection of emergency situation primarily fall detection. Fall detection techniques can be developed using the data from accelerometers, gyroscopes and location sensors (body sensors or tags). Various papers have been published on fall detection techniques [2, 4, 5, 6, 7, 8].

This project is mainly focussed on the approach describe in Luštrek & Kaluža (2009) [2] and the hyperparameter tuning of machine learning algorithms to increase the accuracy. In the mentioned literature [2], 12 body tags were attached to the shoulders, elbows, wrists, hips, knees and ankles. In the data set available with us [9] only 4 body tags were attached on chest, belt and both ankles. This report is structured as follows: Section 2 gives the detailed overview of the data available, its structure and organization. Section 3 discusses about the information acquired from data, removal of outliers from the data and intercorrelation between the features available in data. Section 4 discusses about the filtering of the noise present in data. Section 5 discusses in detail about the methodology for the formation attributes by Luštrek & Kaluža (2009) [2] and how it is modified for current dataset [9]. Section 6 is focussed on the training of machine algorithms, hyperparameter tuning and metrics used for choosing best algorithm. Section 7 concludes the project report and Section 8 finally outlines the future scope of work.

2. INPUT DATA

The purpose of this study is to classify the user's behaviour into one of the following categories

1. walking
2. falling
3. lying down
4. lying
5. sitting down
6. sitting
7. standing up from lying
8. on all fours
9. sitting on the ground
10. standing up from sitting
11. standing up from sitting on the ground

The data available [9] consists of records of 5 people (A, B, C, D, E) for 5 different time period (A01, A02, A03, A04, A05, B01, B02….). Each instance consists coordinates (x, y, z) of 4 body sensors attached to the chest, belt, left ankle and right ankle. Each instance was recorded by Ubisense (2011) real-time location system. It employs ultra-wideband radio technology to determine the locations of up to four tags worn on the user's chest, waist and both ankles [4]. The instances are sampled at approximately 10Hz frequency. The structure of data is shown in *Figure 1*.

	sequence_name	tag_identificator	time_stamp	date	x_coord	y_coord	z_coord	activity
0	A01	010-000-024-033	633790226051280329	27.05.2009 14:03:25:127	4.062931	1.892434	0.507425	walking
1	A01	020-000-033-111	633790226051820913	27.05.2009 14:03:25:183	4.291954	1.781140	1.344495	walking
2	A01	020-000-032-221	633790226052091205	27.05.2009 14:03:25:210	4.359101	1.826456	0.968821	walking
3	A01	010-000-024-033	633790226052361498	27.05.2009 14:03:25:237	4.087835	1.879999	0.466983	walking
4	A01	010-000-030-096	633790226052631792	27.05.2009 14:03:25:263	4.324462	2.072460	0.488065	walking

Figure 1: Structure of Input Data

The input data consists of total 164860 with the following distribution for each activity

Table 1: Class Distribution

S. No.	Activity	No. of Instances
1.	Walking	32710
2.	Falling down	2973

3.	Lying down	6168
4.	Lying	54480
5.	Sitting down	1706
6.	Sitting	27244
7.	Standing up from lying	18361
8.	On all fours	5210
9.	Sitting on the ground	11779
10.	Standing up from sitting	1381
11.	Standing up from sitting on the ground	2848

The above distribution indicates that the distribution is imbalanced and the input data has to be delt with accordingly during the training of machine learning algorithms.

3. DATA CLEANING

Each coordinate has been plotted with time for each sequence (A01, A02….) and it has been observed that apart from the noise in data, there are certain outliers which can be seen as big spikes in the plot (Refer Figure 2). These are the outliers and needs to be removed. These spikes indicate that there is sudden change in position with respect to time and again the position reverts back to its original state, however if this sudden change continues for a period of time then the same can not be considered as outliers. Hence, the outliers are removed where the difference between previous position and next position is large (Refer Figure 3).

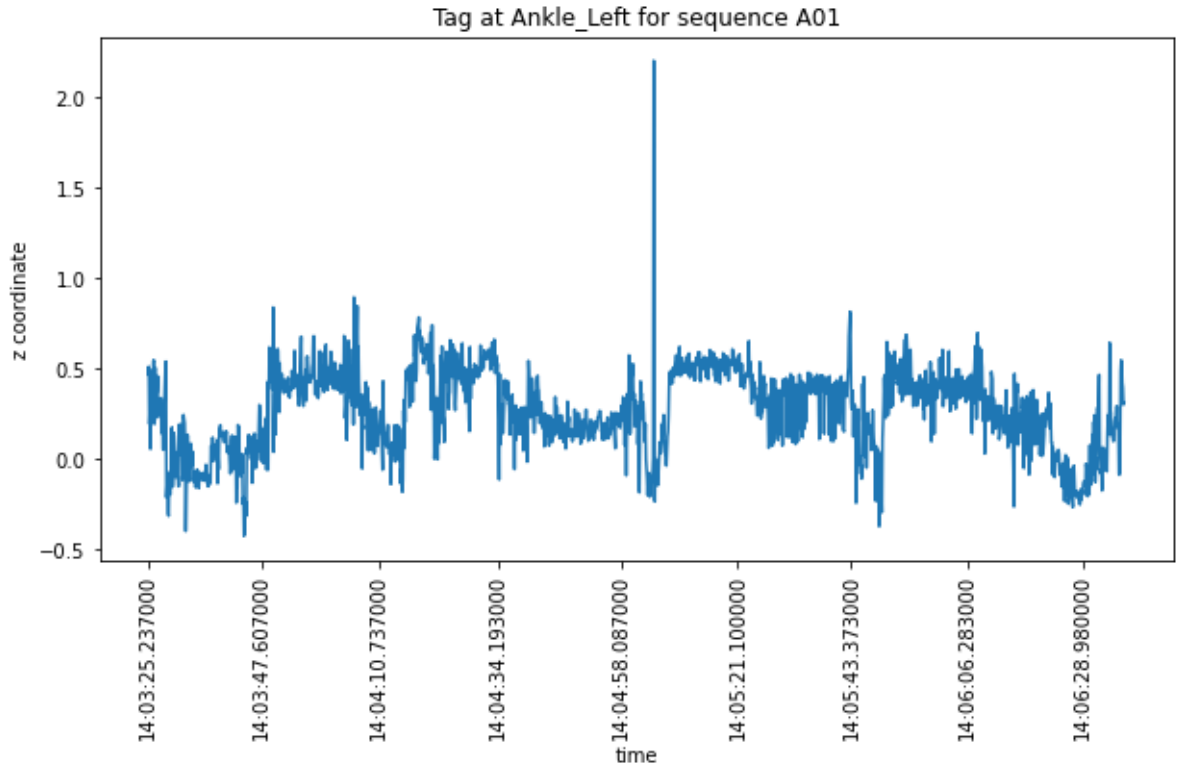


Figure 2: Trend of position coordinate with time

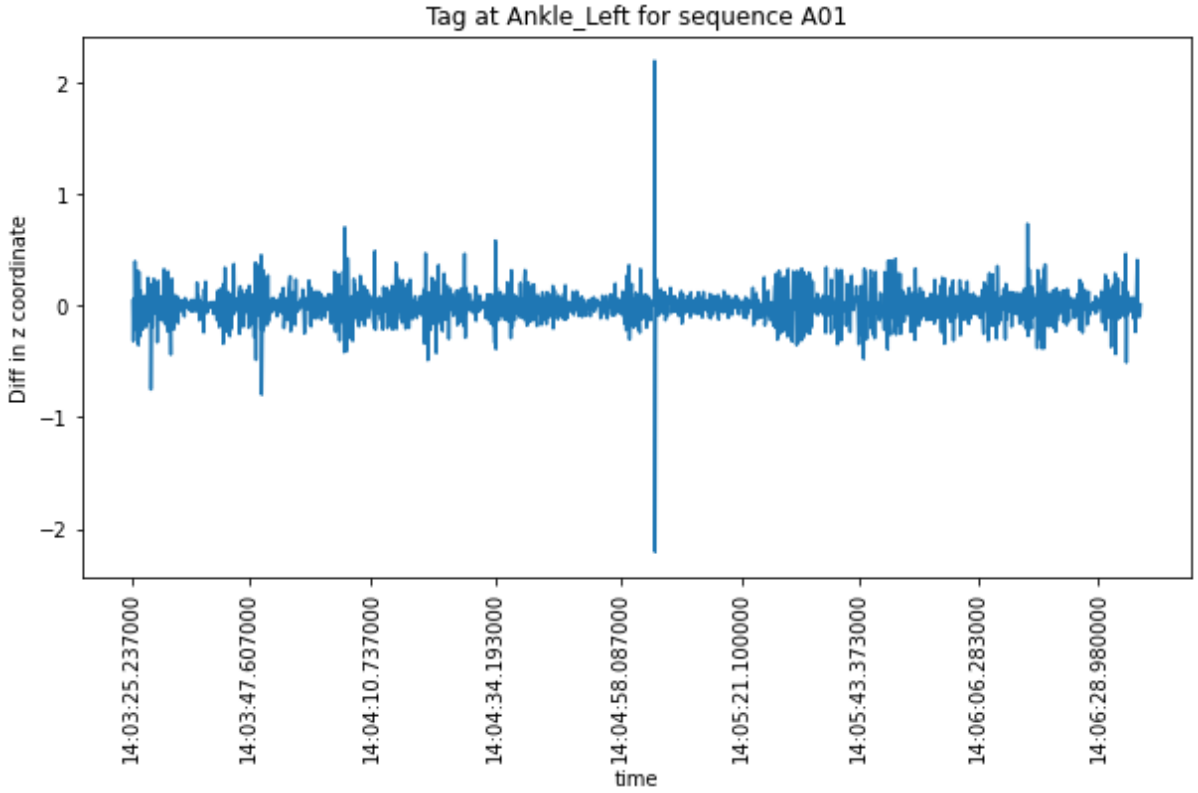


Figure 3: Trend of difference in position coordinate with time

These outliers are removed by observing each trend with time and identifying the outlier value and removing the same from the data. The outliers are removed from each sequence and each position coordinate (x, y, z) such that mostly noise is remain in the dataset (Refer Figure 4). The complete data wrangling can be referred from the jupyter notebook uploaded on the github [10].

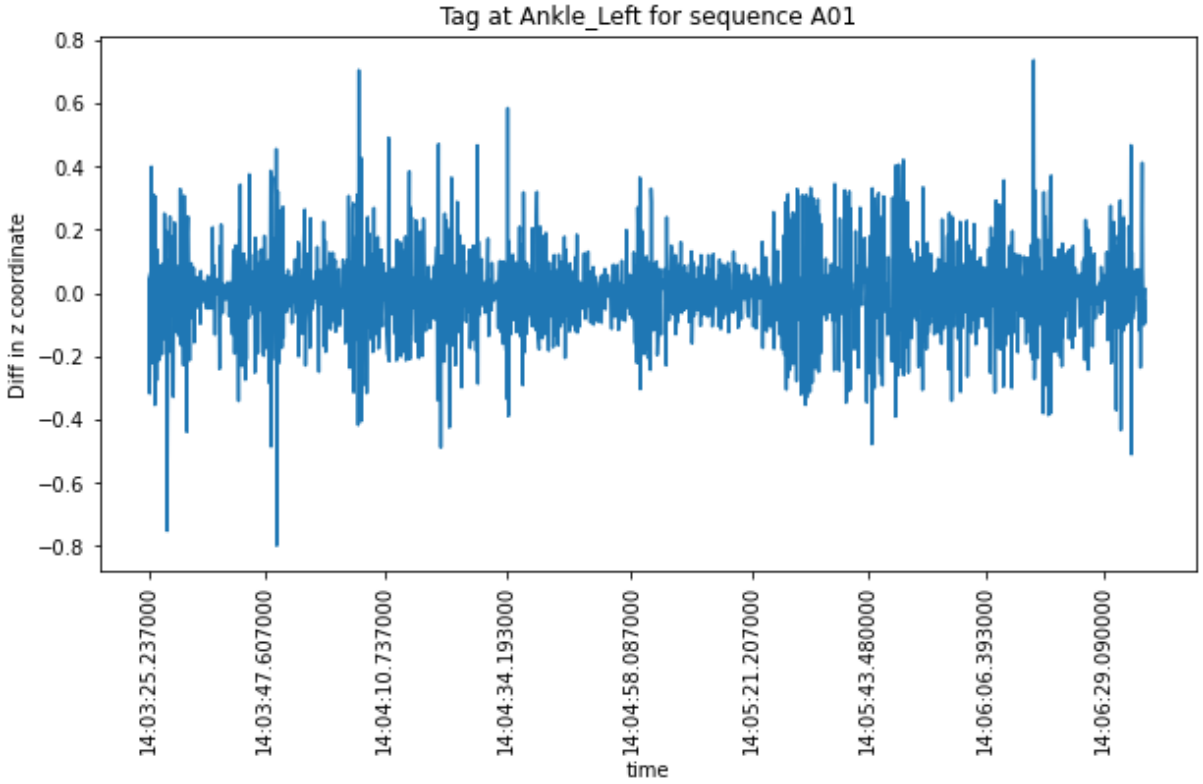
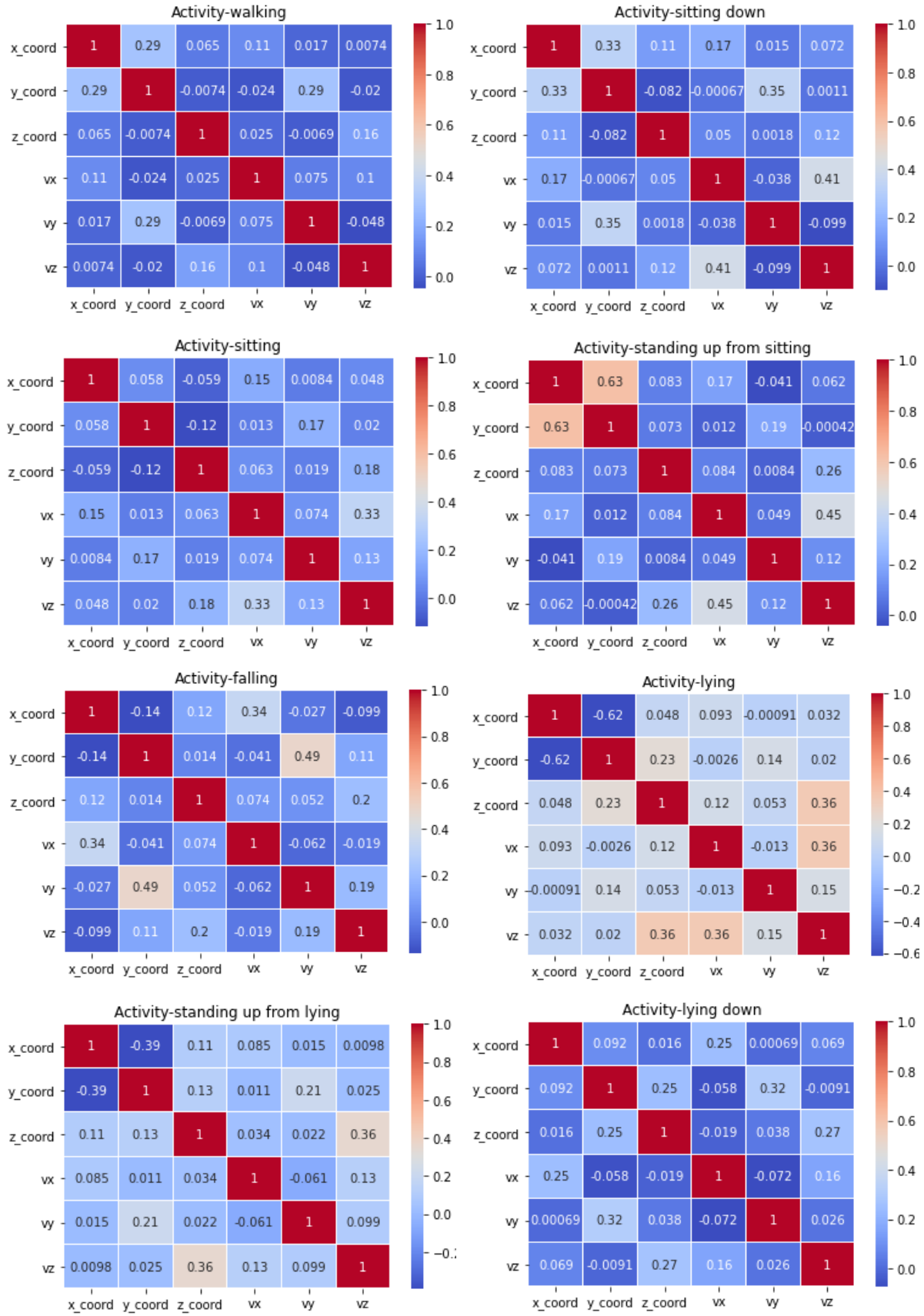


Figure 4: Trend of difference in position coordinate with time after removing the outliers

Removing the outliers have reduced the instances in dataset by about 2%. The dataset has then been saved as noisy data set.

One of the most important features in recognition of activity is the velocity of the body part. The sudden change in velocity can be used to identify the falling activity and the direction of velocity can be used to differentiate between different class of activities. The velocity in x, y and z direction can be calculated by differentiating the position coordinate with respect to time (vx = dx/dt, vy = dy/dt….).

The features (x, y, z, $v_{x,}$, v_y, v_z) are then checked for the correlation by plotting the correlation matrix, the same is given in Figure 5. The matrices indicate that there is no correlation between any feature for each activity class. The jupyter notebook can be referred for complete EDA of the dataset [11].


Activity-walking
x_coord 1 0.29 0.065 0.11 0.017 0.0074
y_coord 0.29 1 -0.0074 -0.024 0.29 -0.02
z_coord 0.065 -0.0074 1 0.025 -0.0069 0.16
vx 0.11 -0.024 0.025 1 0.075 0.1
vy 0.017 0.29 -0.0069 0.075 1 -0.048
vz 0.0074 -0.02 0.16 0.1 -0.048 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0
Activity-sitting down
x_coord 1 0.33 0.11 0.17 0.015 0.072
y_coord 0.33 1 -0.082 -0.00067 0.35 0.0011
z_coord 0.11 -0.082 1 0.05 0.0018 0.12
vx 0.17 -0.00067 0.05 1 -0.038 0.41
vy 0.015 0.35 0.0018 -0.038 1 -0.099
vz 0.072 0.0011 0.12 0.41 -0.099 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0
Activity-sitting
x_coord 1 0.058 -0.059 0.15 0.0084 0.048
y_coord 0.058 1 -0.12 0.013 0.17 0.02
z_coord -0.059 -0.12 1 0.063 0.019 0.18
vx 0.15 0.013 0.063 1 0.074 0.33
vy 0.0084 0.17 0.019 0.074 1 0.13
vz 0.048 0.02 0.18 0.33 0.13 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0
Activity-standing up from sitting
x_coord 1 0.63 0.083 0.17 -0.041 0.062
y_coord 0.63 1 0.073 0.012 0.19 -0.00042
z_coord 0.083 0.073 1 0.084 0.0084 0.26
vx 0.17 0.012 0.084 1 0.049 0.45
vy -0.041 0.19 0.0084 0.049 1 0.12
vz 0.062 -0.00042 0.26 0.45 0.12 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0
Activity-falling
x_coord 1 -0.14 0.12 0.34 -0.027 -0.099
y_coord -0.14 1 0.014 -0.041 0.49 0.11
z_coord 0.12 0.014 1 0.074 0.052 0.2
vx 0.34 -0.041 0.074 1 -0.062 -0.019
vy -0.027 0.49 0.052 -0.062 1 0.19
vz -0.099 0.11 0.2 -0.019 0.19 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0
Activity-lying
x_coord 1 -0.62 0.048 0.093 -0.00091 0.032
y_coord -0.62 1 0.23 -0.0026 0.14 0.02
z_coord 0.048 0.23 1 0.12 0.053 0.36
vx 0.093 -0.0026 0.12 1 -0.013 0.36
vy -0.00091 0.14 0.053 -0.013 1 0.15
vz 0.032 0.02 0.36 0.36 0.15 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6
Activity-standing up from lying
x_coord 1 -0.39 0.11 0.085 0.015 0.0098
y_coord -0.39 1 0.13 0.011 0.21 0.025
z_coord 0.11 0.13 1 0.034 0.022 0.36
vx 0.085 0.011 0.034 1 -0.061 0.13
vy 0.015 0.21 0.022 -0.061 1 0.099
vz 0.0098 0.025 0.36 0.13 0.099 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0
Activity-lying down
x_coord 1 0.092 0.016 0.25 0.00069 0.069
y_coord 0.092 1 0.25 -0.058 0.32 -0.0091
z_coord 0.016 0.25 1 -0.019 0.038 0.27
vx 0.25 -0.058 -0.019 1 -0.072 0.16
vy 0.00069 0.32 0.038 -0.072 1 0.026
vz 0.069 -0.0091 0.27 0.16 0.026 1
x_coord y_coord z_coord vx vy vz
1.0 0.8 0.6 0.4 0.2 0.0

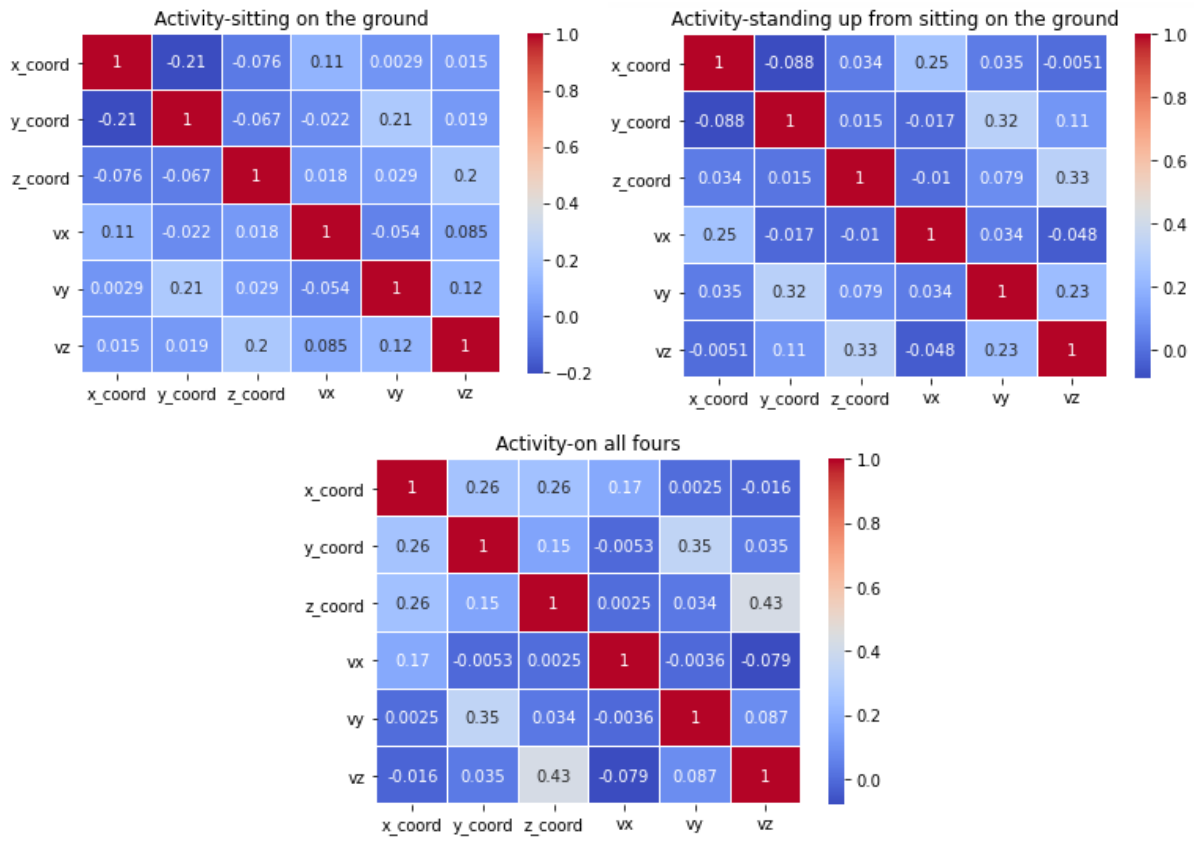


Figure 5: Correlation Matrix for each class of activity

4. FILTERING OF NOISY DATA

As discussed in the previous section the data of location sensors are noisy and hence filtering is needed to smoothen the noisy data. The specified accuracy for the sensors is 15 cm [4]. The noisy data is smoothened with Kalman filter [2] which smoothens sharp changes in both location and velocity. The complete algorithm of Kalman filter for thus project is given in the jupyter notebook [12].

5. MACHINE LEARNING ATTRIBUTES

As discussed in Section 2, the locations are sampled in approximately 10 Hz frequency. All the instances are resampled into one second intervals and the activity class with most frequently occurring in that one second window is taken as classification activity. The coordinates and velocity are averaged over this one second interval. This gave us 5448 instances which are cleaned for conflicting activity class defined for different tags. After cleaning, finally 5230 instances have been derived for which class distribution is as follows:

Table 2: Class Distribution after resampling over one second interval

S. No.	Activity	No. of instances
1	lying	1741
2	walking	1058
3	sitting	863
4	standing up from lying	591
5	sitting on the ground	384
6	lying down	180
7	on all fours	165
8	standing up from sitting on the ground	88
9	falling	82
10	sitting down	40
11	standing up from sitting	38

Luštrek & Kaluža (2009) [2] designed three sets of attributes for describing the user's behaviour which is Reference attributes, Body attributes and Angle attributes. Reference attributes are expressed in the reference coordinate system, which is fixed with respect to the user's environment. Body attributes are expressed in a coordinate system affixed to the user's body. Angle attributes are the angles between adjacent body parts [2].

5.1. Reference Attributes

These attributes are based on the reference coordinate system i.e. the global coordinate system describing the location of the user but Luštrek & Kaluža (2009) [2] used only z coordinate since activity can take place at any location. Hence, following attributes have been used in Luštrek & Kaluža (2009) [2] and the same attributes have been used in present report but instead of 12 tags it is 4 tags.

(i) z coordinate of tag at time t (z^t_{iR}) – 4 nos.

(ii) absolute velocity of the tag (v^t_{iR}) – 4 nos.

(iii) velocity of tag in z direction (v^t_{ziR}) – 4 nos.

(iv) absolute distance between the tags i and j (d^t_{ijR}) – 6 nos.

(v) distance between tag I and j in z direction (d^t_{zijR}) – 6 nos.

Hence, there are total 24 reference attributes for machine learning.

5.2. Body Attributes

The body attributes are expressed in coordinate system with reference to the user's body. Luštrek & Kaluža (2009) [2] uses the centre of left and right hip as the origin and the vector from this origin to the centre of left and right shoulder as z axis of the body. Since we have tags at belt and chest, we used the coordinate of belt as origin (***o***) and vector from belt to chest as the body z axis.

For y axis Luštrek & Kaluža (2009) [2] uses the vector from body origin to the left hip, however we do not have the knowledge of the body direction, hence in this study we used an assumed y axis which is perpendicular to body z axis. So, the following equations are referred for body coordinate system

$\boldsymbol{o} = \frac{\boldsymbol{b}}{|\boldsymbol{b}|}$ where ***b*** is vector of belt tag

$\boldsymbol{k} = \frac{\boldsymbol{c}-\boldsymbol{o}}{|\boldsymbol{c}-\boldsymbol{o}|}$ where ***c*** is vector of chest tag

j is chosen such that ***j.k*** = 0

i = ***j*** x ***k***

Then the coordinates of tag are transformed in body coordinate system by defining a proper transformation matrix. The vector $\boldsymbol{p}_{\mathbf{R}}$ = (x_R, y_R, z_R, 1) corresponds to the point (x_R, y_R, z_R) in the reference coordinate system. The vector $\boldsymbol{p}_{\boldsymbol{B}}$ = (x_B, y_B, z_B, 1) corresponds to the point (x_B, y_B, z_B) in a body coordinate system. $\mathbf{T}_{\mathbf{R}\rightarrow\mathbf{B}}$ is the transformation matrix from the reference to the body coordinate system. Notation $\boldsymbol{i}_{(B)R}$ refers to the basis vector ***i*** belonging to the body coordinate system, expressed in the reference coordinate system.

$\boldsymbol{p}_{\mathbf{B}} = \mathbf{T}_{\mathbf{R}\rightarrow\mathbf{B}}\, \boldsymbol{p}_{\mathbf{R}}^{\mathbf{T}}$

$$\boldsymbol{T}_{R\rightarrow B} = \begin{bmatrix} x_{i(B)R} & y_{i(B)R} & z_{i(B)R} & -\boldsymbol{o}_{(B)R}\boldsymbol{i}_{(B)R} \\ x_{j(B)R} & y_{j(B)R} & z_{j(B)R} & -\boldsymbol{o}_{(B)R}\boldsymbol{j}_{(B)R} \\ x_{k(B)R} & y_{k(B)R} & z_{k(B)R} & -\boldsymbol{o}_{(B)R}\boldsymbol{k}_{(B)R} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The following attributes are used in body coordinate system

(i) coordinates of the tag I at time t (x^t_{iB}, y^t_{iB}, z^t_{iB}) – 7 nos. (only for ankle left and ankle right and z coordinate of chest, since belt will always be origin and x, y of chest will also be 0)

(ii) absolute velocity of the tag (v^t_{iB}) – 4 nos.

(iii) angles of movement of the tag with respect to the z axis and xz plane (ϕ^t_{iB}, θ^t_{iB}) – 8 nos.

(iv) z coordinate of the origin of the body (z^t_{oR}) – 1 no.

(v) direction of x axis of the body coordinate system with respect to z axis and xz plane (Φ^t_{oR}, Θ^t_{oR}) – 2 nos.

(vi) absolute velocity of the origin of the coordinate system (v^t_{oR}) – This attribute will be same as (ii) and hence can be omitted in this project

(vii) angles of movement of the origin of the body coordinate system with respect to z axis and xz plane (ϕ^t_{oR}, θ^t_{oR}) – 2 nos.

In Luštrek & Kaluža (2009) [2] another body coordinate system with reference z was used in which reference z axis was used as body z axis and x axis was determined such that it is perpendicular to body z axis and vector of body origin an left hip. Since in this case we are using arbitrary x or y axis it is as good as using reference coordinate and hence it is not considered in this study.

Moreover, the attributes with first snapshot coordinate system was also used in Luštrek & Kaluža (2009) [2] but later proved to be less significant and therefore is not included in this study also. Therefore, there are 25 body attributes that are fed into machine learning algorithms.

5.3. Angle Attributes

Luštrek & Kaluža (2009) [2] uses angle attributes as the angles between body parts that rotates like shoulder angles and hip angles w.r.t upper torso, angle between upper and lower torso and elbow and knee angles. However, in this project used only the angle between lower and upper torso (q^t_T) as angle attribute which is angle between upper torso & left ankle and angle between upper torso & right ankle. So only 2 attributes of angle are defined for machine learning.

The complete feature engineering jupyer notebook can be referred from [13].

6. MACHINE LEARNING EXPERIMENTS

After generation of all the attributes required for building the machine learning models, the data is then split into training dataset which is 70% of the existing dataset used for training the machine learning model. Rest 30% of the data set is used as test dataset required to determine the accuracy.

Prior to train the machine learning algorithms, the most important process after generating all the attributes is to process the training dataset used for modelling. The processing in this project includes the normalization of the attributes to better train the algorithms and to consider the imbalance of the class distribution.

The normalization of the attributes is done using the standardization where each attribute values are converted to corresponding z score value by subtracting the mean and normalizing by standard deviation.

As observed from Table 2 the classes of activity are not evenly distributed and to overcome this class imbalance issue, resampling has been done by synthesizing new instances from the existing records. This process is called as Synthetic Minority Oversampling Technique (SMOTE). SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbours [16]. Finally, five machine algorithms are tried to train the classifiers for classifying the behaviour into eleven activities. The algorithms are implemented with open source programming language Python which offers various libraries for machine learning. All the machine learning algorithms are tuned on their hyperparameters using GridSearch process and the best parameters are chosen based on the 10-fold cross validation accuracy. The machine algorithms trained and the hyperparameters chosen for grid search are given in Table 3. The algorithms are trained on all three attributes (reference, body and angle) and every possible combination of these. The algorithm with best parameters is chosen to calculate the accuracy on test data set.

Table 3: Algorithms trained and their hyperparameters

S. No.	Algorithm	Hyperparameters Grid
1.	Decision Tree	criterion: [gini, entropy] max_depth: [2, 4, 6, …..40, None]
2.	k-nearest neighbor	n_neighbors: [1, 2, 3, 4….20]
3.	Support Vector Machine	C = [0.001, 0.01, 0.1, 1, 10, 100] gamma = [.0001, .001, 0.01, 1, 'scale', 'auto']
4.	AdaBoost	--

5.	XGBoost	learning_rate = [.01, 0.1, 1] max_depth = [5, 6, 7]

All the above algorithms are trained on noisy as well as clean dataset. The accuracies for training dataset with best hyperparameters are given in Table 4 and the accuracies on test dataset with the best machine learning model is given in Table 5. The accuracy of the best attribute for each algorithm is highlighted in green and the accuracy of the best algorithm for each attribute set is in bold type.

Table 4: 10-fold cross validation accuracy on training dataset

Noisy Data								
S. No.	Algorithm	Reference Attributes	Body Attributes	Angle Attributes	Reference + Body Attributes	Reference + Angle Attributes	Body + Angle Attributes	All
1	Decision Tree	90.11	87.24	43.67	90.81	89.84	87.76	90.34
2	K-Nearest Neighbor	97.78	96.16	**48.68**	97.65	97.98	96.48	97.68
3	Support Vector Machine	**98.29**	**98.51**	25.15	**98.96**	**98.46**	**98.64**	**99.02**
4	AdaBoost	98.19	98.34	48.00	98.78	98.33	98.42	98.92
5	XGBoost	97.63	97.53	43.55	98.21	97.44	97.75	98.33
Clean Data								
1	Decision Tree	93.05	91.47	46.84	93.83	93.72	91.69	94.02
2	K-Nearest Neighbor	98.70	98.30	**52.65**	98.86	98.73	98.30	98.81
3	Support Vector Machine	98.73	**98.87**	29.68	**99.09**	98.74	**98.95**	**99.24**
4	AdaBoost	**98.96**	98.73	50.88	99.00	**98.77**	98.82	99.13
5	XGBoost	98.25	98.10	47.09	98.59	98.29	98.20	98.59

Table 5: Accuracy on test dataset

S. No.	Algorithm	Reference Attributes	Body Attributes	Angle Attributes	Reference + Body Attributes	Reference + Angle Attributes	Body + Angle Attributes	All
Noisy Data								
1	Decision Tree	72.15	65.26	15.07	73.68	69.31	65.87	71.53
2	K-Nearest Neighbor	76.28	62.81	**15.15**	73.37	76.28	62.50	74.06
3	Support Vector Machine	80.64	81.25	14.23	85.76	81.63	82.01	85.46
4	AdaBoost	**85.92**	**83.85**	14.46	**87.83**	**86.30**	**84.54**	**87.98**
5	XGBoost	82.78	82.17	14.92	86.38	83.39	82.47	85.69
Clean Data								
1	Decision Tree	77.54	71.64	**17.54**	78.77	77.54	70.26	79.08
2	K-Nearest Neighbor	86.28	83.75	16.93	88.35	85.59	85.05	87.89
3	Support Vector Machine	89.42	88.50	10.26	90.72	89.57	88.88	90.65
4	AdaBoost	**89.50**	**89.96**	17.47	**90.88**	**90.26**	**89.88**	**91.34**
5	XGBoost	88.42	87.04	16.16	90.11	88.04	87.12	89.96

Falling is more important as fall activity detection is the main goal of this project. Hence, precision, recall and F1-score of fall detection have also been reported in respectively.

Table 6: Precision for detecting falling class

S. No.	Algorithm	Reference Attributes	Body Attributes	Angle Attributes	Reference + Body Attributes	Reference + Angle Attributes	Body + Angle Attributes	All
Noisy Data								
1	Decision Tree	0.14	0.28	0.02	0.26	0.08	0.27	0.23
2	K-Nearest Neighbor	0.28	0.22	0.02	0.28	0.20	0.25	0.26
3	Support Vector Machine	0.30	0.57	0.01	0.68	0.32	0.50	0.71
4	AdaBoost	**0.53**	**0.80**	0.02	**0.81**	**0.50**	**0.88**	**0.90**
5	XGBoost	0.40	0.66	0.02	0.52	0.44	0.45	0.56
Clean Data								
1	Decision Tree	0.17	0.22	0.03	0.30	0.23	0.17	0.33
2	K-Nearest Neighbor	0.35	0.33	0.04	0.47	0.33	0.50	0.36
3	Support Vector Machine	0.52	0.42	0.01	0.53	0.53	0.46	0.53
4	AdaBoost	**0.55**	**0.60**	0.02	**0.62**	**0.71**	**0.63**	**0.75**
5	XGBoost	0.54	0.63	0.02	0.53	0.30	0.58	0.42

Table 7: Recall for detecting falling class

S. No.	Algorithm	Reference Attributes	Body Attributes	Angle Attributes	Reference + Body Attributes	Reference + Angle Attributes	Body + Angle Attributes	All
Noisy Data								
1	Decision Tree	0.30	0.35	0.10	**0.55**	0.15	**0.45**	0.35
2	K-Nearest Neighbor	0.40	0.35	0.10	0.40	0.35	0.40	0.40
3	Support Vector Machine	**0.50**	0.20	0.10	**0.55**	**0.45**	0.25	**0.50**
4	AdaBoost	0.40	**0.40**	0.10	0.45	0.35	0.40	**0.50**

5	XGBoost	0.40	0.30	0.10	0.45	0.40	0.25	0.45
Clean Data								
1	Decision Tree	0.26	**0.42**	0.15	0.31	0.36	0.26	**0.47**
2	K-Nearest Neighbor	0.36	0.36	**0.21**	**0.47**	**0.42**	**0.36**	0.36
3	Support Vector Machine	**0.47**	0.31	0.10	0.36	**0.42**	0.31	0.36
4	AdaBoost	0.26	0.31	0.15	0.26	0.26	**0.36**	0.31
5	XGBoost	0.31	0.36	0.10	0.42	0.21	**0.36**	0.31

Table 8: F1-score for detecting falling class

Noisy Data								
S. No.	Algorithm	Reference Attributes	Body Attributes	Angle Attributes	Reference + Body Attributes	Reference + Angle Attributes	Body + Angle Attributes	All
1	Decision Tree	0.19	0.31	0.03	0.35	0.10	0.34	0.28
2	K-Nearest Neighbor	0.33	0.27	0.03	0.33	0.25	0.31	0.32
3	Support Vector Machine	0.37	0.29	0.02	**0.61**	0.38	0.33	0.58
4	AdaBoost	**0.45**	**0.53**	0.03	0.58	0.41	**0.55**	**0.64**
5	XGBoost	0.40	0.41	0.03	0.48	**0.42**	0.32	0.50
Clean Data								
1	Decision Tree	0.20	0.29	0.05	0.30	0.28	0.21	0.39
2	K-Nearest Neighbor	0.35	0.35	0.06	**0.47**	0.37	0.42	0.36
3	Support Vector Machine	**0.50**	0.36	0.02	0.43	**0.47**	0.37	0.43
4	AdaBoost	0.35	0.41	0.04	0.37	0.38	**0.46**	**0.44**
5	XGBoost	0.39	**0.46**	0.03	**0.47**	0.25	0.45	0.36

Since prediction of falling class is of prime concern, instead of accuracy precision, recall and F1-score have been studied to choose the best algorithm and best attribute combination. From the above tables, it seems that best algorithm is AdaBoost considering all set of attributes. One of the most important observation to mention here is that although clean data depicts higher accuracy fall class is best predicted by noisy data. This is because of the reason that while smoothening, the sudden change in

velocities have been lowered by the Kalman filter and the sudden change in velocity is the most important feature to predict the falling class.

7. CONCLUSION

This project aims at predicting the activity of the person based on the data of location sensors attached on 4 body parts i.e. chest, belt and both ankles. Fall detection is the main goal of this project so that elderly people can live independently and any emergency situation can be directly communicated to the nursing care and family members.

The activities are classified into 11 classes and the attributes are the coordinates of the body parts in the reference coordinate system, body coordinate system and the angle between upper and lower torso. The machine learning algorithms are trained on these single attributes and combinations of these three attributes. The machine learning algorithms are also hyperparameter tuned with 10-fold cross validation.

The best training accuracy of 99.02% on noisy data and 99.24% on clean data has been achieved by Support vector machine considering all attributes. The best accuracy on test data of 87.98% on noisy and 91.34% on clean data has been depicted by AdaBoost considering all attributes.

Since falling is of primary concern with this study, precision recall and F1-score have been given more importance while choosing the best algorithm and best attributes combination. In this context best precision of 0.90, recall of 0.50 and F1-score of 0.64 has been given by AdaBoost on noisy data. The clean data exhibits comparatively less F1-score of 0.50 for falling class.

8. SCOPE OF FUTURE WORK

There could be three possible directions for future work. The first one is to tune wide range of hyperparameters and the second direction is to use more robust methods like artificial neural networks. The third direction is using ensemble method for stack generalization in which probabilities are calculated for each algorithm and then stacked with certain weightages. Then a threshold probability should be chosen so as to maximize the F1-score.

9. REFERENCES

1. Confidence project 2011. http://www.confidence-eu.org/
2. Luštrek, Mitja & Kaluža, Boštjan. (2009). Fall Detection and Activity Recognition with Machine Learning. Informatica 33 (2009) 205–212.
3. Kaluza, Bostjan & Mirchevska, Violeta & Dovgan, Erik & Lustrek, Mitja & Gams, Matjaz. (2010). An Agent-Based Approach to Care in Independent Living. 177-186. 10.1007/978-3-642-16917-5_18.
4. Luštrek, Mitja & Gjoreski, Hristijan & Kozina, Simon & Cvetkovic, Božidara & Mirchevska, Violeta & Gams, Matjaž. (2011). Detecting Falls with Location Sensors and Accelerometers. Proceedings of the Twenty-Third Innovative Applications of Artificial Intelligence Conference.
5. Mirčevska, Violeta & Luštrek, Mitja & Gams, Matjaž (2009). Combining Machine Learning and Expert Knowledge for Classifying Human Posture. In Proceedings of ERK 2009, 183–186.
6. Gjoreski, Hristijan & Gams, Matjaz. (2011). Activity/Posture Recognition using Wearable Sensors Placed on Different Body Locations. 10.2316/P.2011.716-067.
7. Mirchevska, Violeta & Lustrek, Mitja & Velez, Igone & González Vega, Narciso & Gams, Matjaz. (2009). Classifying Posture Based on Location of Radio Tags.. 85-92. 10.3233/978-1-60750-480-1-85.
8. Lustrek, Mitja & Kaluza, Bostjan & Dovgan, Erik & Pogorelc, Bogdan & Gams, Matjaz. (2009). Behavior Analysis Based on Coordinates of Body Tags. Lecture Notes in Computer Science. 14-23. 10.1007/978-3-642-05408-2_2.
9. Luštrek, M., Kaluža, B., Piltaver, R., Krivec, J., and Vidulin, V. 2010. Localization Data for Person Activity Data Set. http://archive.ics.uci.edu/ml/datasets/Localization+Data+for+Person+Activity.
10. Jupyter Notebook – Data Wrangling for Capstone Two. https://github.com/pgupta88/Springboard/blob/master/Capstone_Two/Data%20Wrangling/Capstone_Two.ipynb
11. Jupyter Notebook – Exploratory Data Analysis for Capstone Two. https://github.com/pgupta88/Springboard/blob/master/Capstone_Two/EDA/Capstone_Two_EDA.ipynb

12. Jupyter Notebook – Kalman Filter for Capstone Two. https://github.com/pgupta88/Springboard/blob/master/Capstone_Two/Filtering/Capstone_Two_Filtering.ipynb
13. Jupyter Notebook – Feature Engineering for Capstone Two. https://github.com/pgupta88/Springboard/blob/master/Capstone_Two/Feature%20Enginnering/Capstone_two_feature_engineering.ipynb
14. Jupyter Notebook – Modelling on noisy data for Capstone Two. https://github.com/pgupta88/Springboard/blob/master/Capstone_Two/Modelling/Capstone_two_Modelling.ipynb
15. Jupyter Notebook – Modelling on smooth data for Capstone Two. https://github.com/pgupta88/Springboard/blob/master/Capstone_Two/Modelling/Capstone_two_Modelling_Filtered_data.ipynb
16. Chawla, Nitesh V., Bowyer, Kevin W., Hall, Lawrence O. & Keglemeyer, W. Phillip. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16 (2002) 321–357.