Predicting Human Activity through Machine Learning Techniques: A Comparative Study of Random Forest, Naïve Bayes, SVM, Decision Tree, and Logistic Regression.

Paul Johnson
Faculty of Engineering Environment and Computing
Coventry University
Coventry, England
johnsonp4@uni.coventry.ac.uk

***Abstract*-This paper evaluates the performance of various machine learning techniques for classifying human activities based on sensor data collected from eight subjects. The methods used include Decision tree, Support Vector algorithm, Logistics Regression, Gaussian Naive Bayes and Random Forest, which were tuned using hyperparameter tuning. The models were evaluated by comparing their classification accuracy on testing data, visualizing the results using confusion matrices, and validating their performance using 10-fold cross-validation. The dataset used in the study consisted of 9120 nos of 5-second samples of activity data collected from 8 subjects performing 19 different activities, each sample has 5625 features. Principal Component Analysis was applied to reduce the dimensionality of the data by extracting the first 600 principal components, which explained over 90% of the variance in the data.**

***Keywords*-** *Classification; machine learning; python; Hyperparameter tuning; Model Selection; Principal Component Analysis*

I. INTRODUCTION

Accurately measuring human movement is a challenging endeavour since the body's various components must work together to carry out an action. To comprehend how each body part moves, each one must also be individually examined. There would be a wide range of practical uses for a machine that could effectively analyse and recognise human behaviour, including the ability for surveillance systems to spot unusual or suspicious behaviour, and for the robotics industry to create devices that can communicate with people more effectively.

Advancements in micro-electro-mechanical systems (MEMS) technology made Inertial sensors smaller, lighter, and less expensive because. These tiny sensors, which may detect angular rate and linear or angular velocity rate,include gyroscopes, accelerometers, and sometimes magnetometers as well. Additionally, the sensors come in single-axis, two-axis, and tri-axis versions with various levels of axis sensitivity. Tri-axial magnetometers can measure the Earth's magnetic field's strength and direction as a vector. Because they are expensive and demand a high level of accuracy, inertial sensors have historically only been used in a few industries. However, new opportunities in fields like human activity recognition have emerged as a result of the accessibility of low-cost, medium-performance sensors. (K. Altun, 2010)

II. LITERATURE REVIEW

The Daily and Sports Activities dataset is a publicly available dataset that can be found on the UCI Machine Learning Repository website. It is created by Professor Bilur Burshan and his team at Bilkent University in Ankara, Turkey. The dataset was collected by conducting an experiment with eight subjects (four females and four males) performing 19 different daily and sports activities for 5 minutes each. The subjects were equipped with five MTx 3-DOF orientation trackers manufactured by Xsens Technologies, which included tri-axial accelerometers, gyroscopes, and magnetometers to acquire 3D acceleration data. (Barshan, 2010)

Data mining, machine learning, and studies relating to the recognition of human activities have all made extensive use of the dataset. Decision tree, random forest, and support vector machine are just a few of the machine learning methods that have been trained and evaluated using it. The dataset has additionally been utilised in studies to create strategies for raising the precision of human activity recognition systems. The association between human activities and physiological variables like heart rate has also been explored in certain research using this dataset.

Overall, the Daily and Sports Activities dataset has been a valuable resource for researchers in the field of human

activity recognition and machine learning, providing a large and diverse dataset for training and evaluating various algorithms. (Yüksek, 2014)

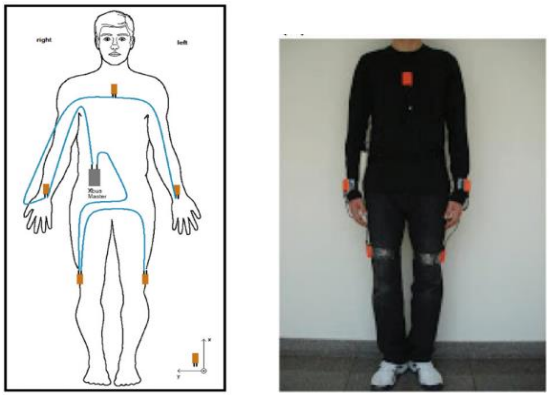

Fig 1 . The 5 sensors fixed in a human.

III. THE DATA SET

Dataset includes data collected from eight subjects (four females and four males) performing 19 different daily and sports activities. The activities are shown in Table I. Each activity was performed for 5 minutes by each subject, resulting in a total of 76 minutes of data per subject.

TABLE I
ACTIVITIES PERFORMED

Activity	Description
A1	Sitting
A2	Standing
A3	Lying on back
A4	Lying on right side
A5	Ascending stairs
A6	Descending stairs
A7	Standing in an elevator
A8	Standing in an elevator (moving)
A9	Walking in a parking lot
A10	Walking on a treadmill with a speed of 4 km/h
A11	A10 in a 15-degree inclined position
A12	Running on a treadmill with a speed of 8 km/h
A13	Exercising on a stepper
A14	Exercising on a cross trainer
A15	Cycling on an exercise bike (horizontal position)
A16	Cycling on an exercise bike (vertical position)
A17	Rowing
A18	Jumping
A19	Playing basketball

The dataset is given as segment of 5sec for 5 minutes for an activity giving 60 segments for an activity for a person. Each segment contains 25hz of data meaning 5x25=125 rows for 45 sensors. This gives the no of features in the dataset to be 5625 i.e 5x25x45. All the columns in the dataset are numerical except for the activity column which need to be predicted. The dataset has 60 segments x 8 persons i.e 480 instance per activity giving total 480x19 instances i.e 9120 samples.

This paper looks through different machine learning algorithms Support Vector Classification, Gaussian Naïve Bayes, Decision Tree, Logistic Regression and Random Forest and, then use a voting between best 3 classifiers to predict the Activity.

IV. METHODS

Decision Tree

A decision tree algorithm is a supervised learning method used for classification. A decision tree is a tree-like structure that models a flowchart. It is composed of internal nodes, branches, and leaf nodes. Each internal node represents a feature or attribute and the branches represent decision rules. The leaf nodes represent the outcome. The highest node in the tree is referred to as the root node. The decision tree uses a recursive process to divide the data into smaller subsets by selecting an attribute and creating new internal nodes and branches.The final decision tree is built by repeating this process for all the attributes, and it chooses the attribute that results in the highest information gain or reduction in impurity. There are different methods for selecting features in a decision tree, the common ones used are gini impurity and entropy. Gini impurity calculates the probability of a particular feature randomly selecting a sample from a particular class, and the lower the probability, the higher the information gain. Whereas, the entropy calculates the randomness or disorder in the data, and the higher the disorder, the higher the entropy. The idea is to select the feature that reduces the entropy the most.

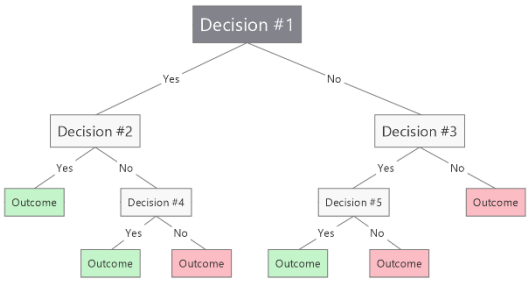

Fig 2. Decision Tree

Entropy = - ∑(p(i) * log2(p(i)))
Gini Impurity = 1 - ∑(p(i)^2)
where p(i) is the probability of a sample belonging to a class ‘i’.

The most common hyperparameters for Decision Tree are
Maximum Depth- Maximum number of levels in a tree
Minimum Samples per leaf- Minimum number of samples required to be at leaf node

Criterion – Gini or Entropy need to be calculated.
Minimum impurity Decrease- minimum impurity required to split a node. This helps to prevent a tree from overfitting by checking whether the impurity reduction is significant.

Support Vector Classification

The Support Vector Classification (SVC) method operates by identifying the optimal hyperplane in a high-dimensional space that maximizes the separation between the various classes. The data points closest to the hyperplane are known as support vectors and have the greatest impact on the position of the hyperplane. The goal of SVC is to find the hyperplane that has the greatest distance to the nearest data points of any class, which is known as the margin. SVCs can be used for both linear and non-linear classification problems and can be extended to handle multiple classes through techniques such as one-vs-all and one-vs-one. The kernel trick is a technique used in SVC to transform the input data into a higher dimensional space where it becomes linearly separable. (M. A. Hearst, 1998)

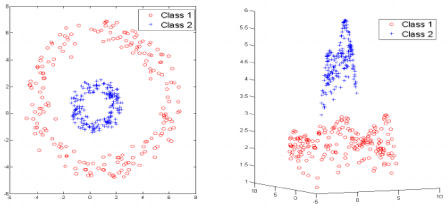

Fig 3. Support Vector Classification nonlinear separation with kernel trick.

The common hyper parameters are:-

C- This regularization parameter balances the goals of minimizing both the training error and testing error. A smaller value of C creates a wider street but more margin violations.

Kernel: The type of kernel function, a mathematical tool, to convert the data into a higher dimensional space. Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid.

Gamma: The parameter for the radial basis function (RBF) kernel, a smaller value of gamma will result in a wider radial basis function and a simpler decision boundary.

Logistic Regression

Logistic regression is the way to predict a binary outcome through regression. We use sigmoid function for the same.

p(x) = 1 / (1 + e^(-w^T x))

Where p(x) is the probability of the target variable being 1 given the input x and w is the vector of weights. The term w^T x is the dot product of the weight vector and the input vector, and e^(-w^T x) is the exponential function of the dot product.

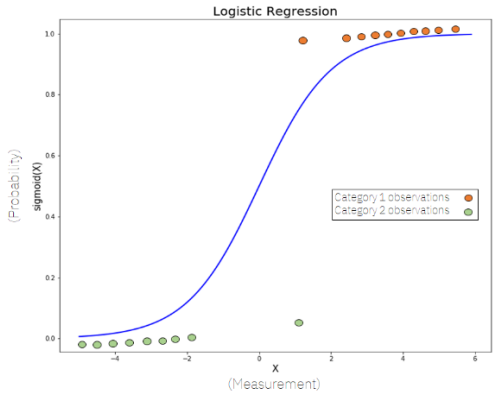

Fig 4. Logistic Regression

The common hyperparameters Regularization in Logistic Regression is regularization term To prevent overfitting and control the complexity of the model, a penalty term is added to the loss function. The regularization term is usually the *L1 (Lasso) or L2 (Ridge)* norm of the coefficients, multiplied by a regularization strength parameter (λ or C).

The equation for L1 (Lasso) regularization in Logistic Regression is:

Loss = -(1/n) * sum(y_i * log(p_i) + (1 - y_i) * log(1 - p_i)) + λ * ||w||_1

The equation for L2 (Ridge) regularization in Logistic Regression is:

Loss = -(1/n) * sum(y_i * log(p_i) + (1 - y_i) * log(1 - p_i)) + λ/2 * ||w||^2_2

where n is the number of samples, y_i are the true labels, p_i are the predicted probabilities, w is the weight vector, ||w||_1 is the L1 norm of the weight vector, and |w||^2_2 is the L2 norm of the weight vector.

The *regularization strength (λ or C)* controls the balance between the error term and the regularization term in the loss function. A larger value of λ (or smaller value of C) results in stronger regularization, while a smaller value of λ (or larger value of C) results in weaker regularization. (Gareth James)

Gaussian Naïve Bayes

Gaussian Naive Bayes is a based on the Bayes Theorem and have a naïve assumption that features are independent of each other. It is called "naive" because it makes a strong independence assumption, which is rarely true in real-world problems. Despite this, it works well in many cases and is often used for text classification and spam filtering. In Gaussian Naive Bayes, the likelihood of the features is assumed to be Gaussian, which means that the algorithm models the data generating process using a

Gaussian distribution. It is a fast and simple algorithm that can be used for both binary and multiclass classification problems.The equation for Gaussian Naive Bayes as per Bayes' theorem, is:

$P(A|B) = P(B|A) * P(A) / P(B)$

In the case of Gaussian Naive Bayes, A represents the class label (e.g. "positive" or "negative"), B represents the feature vector (i.e. the set of input features or variables), and P(A|B) represents the probability of class A given the feature vector B.

The hyperparameter used in the Gaussian Naive Bayes (GNB) algorithm is a smoothing parameter, also known as *Laplace Smoothing or regularization*, to adjust for high variance estimates when the number of samples for a class or feature is small.

$var\_estimate = (N\_i + \alpha) / (N + \alpha * k)$

$pdf = 1 / (sqrt(2 * \pi) * \sigma\_y) * exp(-(x - \mu\_y)^2 / (2 * \sigma^2\_y))$

The smoothed estimate of variance is calculated as $(N\_i + \alpha) / (N + \alpha * k)$, where N_i is the number of samples for a feature in a class, N is the total number of samples in that class, *α is the smoothing parameter*, and k is the number of features. The smoothed variance estimate is used to calculate the Gaussian probability density function (pdf) for each feature given a class, with mean μ_y and variance σ^2_y. The joint probability for each class is then calculated using the pdf, which is used to make the final classification decision.

Random Forest

This algorithm uses an ensemble learning method averaging of prediction from multiple decision trees to make final prediction. By this error from some trees will cancel out when final prediction is made. (Grąbczewski, 2014)

Some hyperparameters for the Random Forest is n_estimators , criterion.

n_estimators: The number of trees in the forest.

Criterion: Gini or Entropy same as that of decision tree.

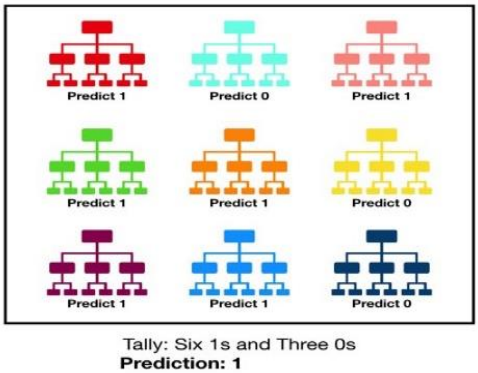

Fig 5. Random Forest

V. Experimental Setup

The original data set was stored in a folder called "Data" within the UCI Machine Learning Repository, which contained 19 subfolders, each representing a different class of activity. These subfolders each had 8 additional subfolders, which held 60 text files with 125 lines of comma-separated values, corresponding to the 45 feature columns. The data was then consolidated into a single file with 9120 lines, where each line represented the 5625 features (45 attribute columns x 125 samples) of a single instance, along with its class. The data was well-balanced, with an equal number of instances for each activity, and no missing data. To make the data more suitable for machine learning techniques, it was standardized (Hackeling, 2014) and then dimensionality reduction was applied using Principal Component Analysis (PCA) (S. Wold, 1987). This resulted in a reduction of the number of features from 5625 to 600 as 600 principal components explained 90% cumulative variance of the original data. These, can be observed from the Fig 6. are where the first 600 principal components were plotted to show the cumulative explained variance of the data. The data is prepared from raw data by using pyhton package OS and the python code file for the same is 'datapreparation.ipynb' and the resulting dataframe is saved as csv file named 'activitydata.csv'. Then, with the prepared data a new python file named 'Model_Building.ipynb' is made for dimensionality reduction and model building. The entire python execution is done through Google Colab and the model building PCA and analysis is done through python package Scikit-Learn(sklearn).

VI. Experiment Results

The data has first 5625 features ,Principal Component Analysis is applied to reduce the dimensions of the data and plotted the variance proportion and cumulative variance proportion. It can be observed that with 600 principal components 90% variance in data is captured. The data is then transformed to 600 features . The dataset is divided in to training 75% and testing 25%.

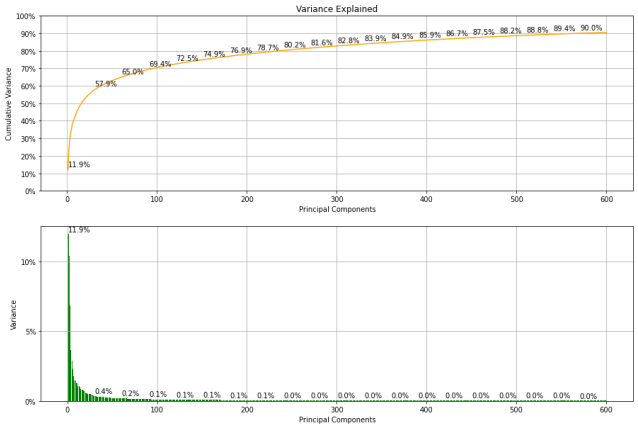

Fig 6. 600 Principal Components

There are 5 classification techniques used. Decision tree, Logistic Regression, Gaussian Naïve Bayes as well as Random Forest to classify 19 activities. Then, the training data is applied to the classifiers with Hyperparameter tuning and a model is built with best hyperparameter for each classifier and then the score with respect to testing data is found.

The comparison of different classifier is done through different methods. Firstly, testing data score is compared in a table(Table II). Secondly, K-fold cross validation is applied in 10 splits to the training data for different classifiers and their scores are plotted in a boxplot(Fig 7). Thirdly, heatmap of confusion matrix is plotted for testing data for each classifier. Finally, heatmap for the f1 score for each class with respect to classifiers is plotted. The F1 score is a measure of a model's accuracy that takes into account both precision and recall. It ranges from 0 to 1, where 1 is the best score, which means a perfect balance of precision and recall, and 0 is the worst case, which means a model that doesn't predict any positive example.

F1 = 2 * (precision * recall) / (precision + recall)

The f1 score of each algorithm is plotted as heat map in Fig 8.

The confusion matrix for each classifier is also plotted for testing data. Confusion matrix represents the no of prediction of classes against actual classes.(Fig 9 to Fig 13)

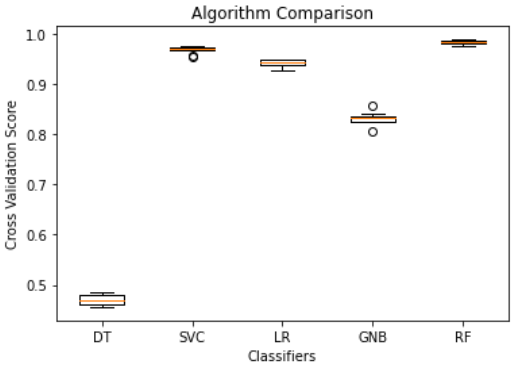

Fig 7. Cross Validation Score Boxplot

TABLE II
TESTING SCORES

Classifier	Testing Score	Training Time (Seconds)
Decision Tree	45.48	14192.83
Support Vector	96.58	686.72
Logistic Regression	94.12	643.36
Gaussian-Naïve-Bayes	82.98	9.25
Random Forest	98.68	7315.46

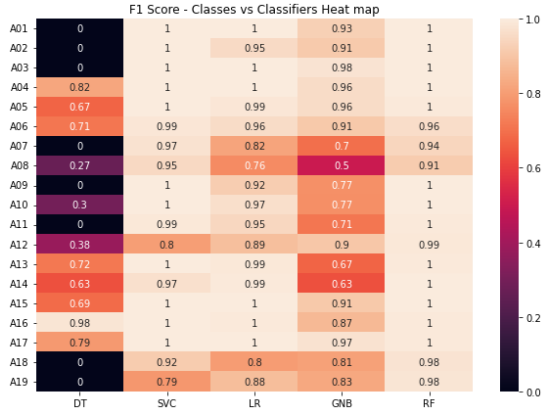

Fig 8. F1 score Heat map for each classes to each classifier

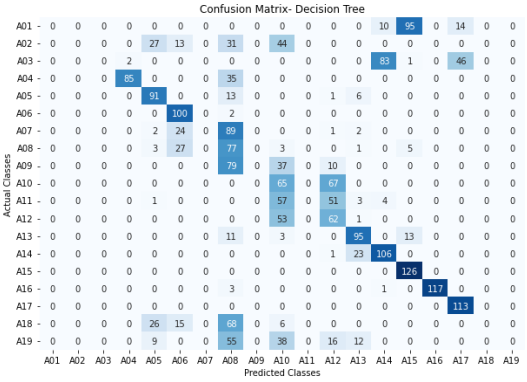

Fig 9. Confusion Matrix -Decision Tree Classifier

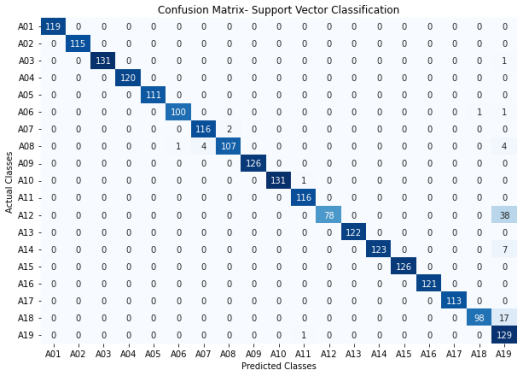

Fig 10. Confusion Matrix - Support Vector Classifier

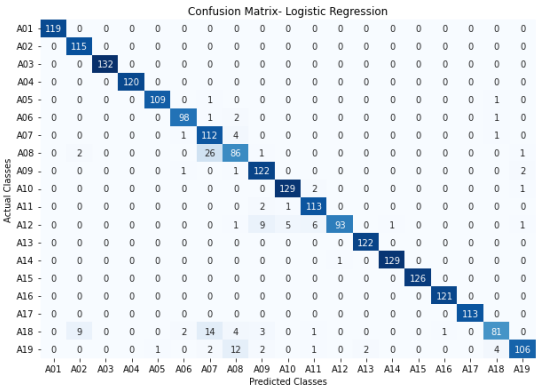

Fig 11. Confusion Matrix - Logistic Regression

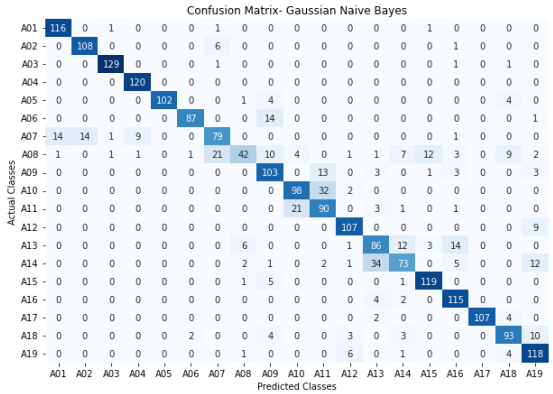

Fig 12. Confusion Matrix for Gaussian-Naïve-Bayes

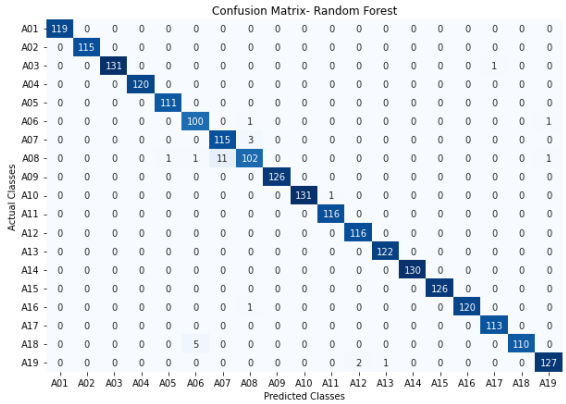

Fig 13. Confusion Matrix Heat Map Random Forest

From the analysis Random Forest was the best performer among the algorithms tested, with a score of 98.68 on the test data. Support Vector and Logistic Regression also performed well, with scores of 96.58 and 94.12 respectively. Gaussian Naive Bayes had a score of 82.98, while Decision Tree was the worst, misclassifying many samples and with a score of 45.48. Also, the Random Forest model showed high accuracy in identifying classes correctly, according to the confusion matrix and f1 score heat map. Conversely, the Decision Tree was the worst performer. The cost of computation, which is correlated to the time taken for training each algorithm varied, with Gaussian Naive Bayes being the fastest to train at 9.25 sec and Random Forest and Decision Tree taking the longest.

VII. Discussion and Conclusion

The paper aims to study and compare five machine learning algorithms (Decision Tree, Random Forest, SVM, Gaussian NB, and Logistic Regression) by tuning their hyper-parameters and using PCA for dimensionality reduction (due to the large number of features, 5625). The evaluation of the model performance is conducted using various metrics including testing score, computational time, confusion matrix, cross-validation boxplot, and F1 score heat map. The objective is to identify the best model for activity prediction.

There can be further studies and investigation done to the data for different areas like medical field, health analysis, fall detection etc.. (Barshan, 2010) But we are limiting the learning to building an effective model to predict the activity.

The paper concludes that, in terms of accuracy, Random Forest is the most favorable model. However, if the computation cost for training the model is an issue, then Support Vector Classifier would be a more suitable option as it strikes a better balance between accuracy and computational cost.

VIII. Appendix

The modelbuilding('Model_Building.ipynb') python file, datapreparation('datapreparation.ipynb') python file and the prepared data ('activitydata.csv') as well as the model, scaling and pca are in the google drive link: https://drive.google.com/drive/folders/1lQpN2t01g1kzOMORTO-6Hw_rpkwmqWpp?usp=share_link

IX. References

Barshan, K. A. (2010). Human activity recognition using inertial/magnetic sensor units. *First International Workshop on Human Behavior Understanding (in conjunction with the 20th Int. Conf. on Pattern Recognition)* (pp. 38-51). Istanbul: Springer: Berlin, Heidelberg.

Gareth James, D. W. (n.d.). "An Introduction to Statistical Learning".

Grąbczewski, K. (2014). Meta-Learning in Decision Tree Induction. In K. Grąbczewski, *Studies in Computational Intelligence, Volume 498.* New York: Springer.

Hackeling, G. (2014). *Mastering Machine Learning with Scikit-Learn.* Birmingham: Packt Publishing.

K. Altun, B. B. (2010). `Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*.

M. A. Hearst, S. T. (1998). Support Vector Machines. *IEEE Intelligent Systems and their Applications*, 18-28.

Raschka, S. (2015). *Python Machine Learning.* Packt Publishing.

S. Wold, K. E. (1987). Principal Component Analysis. *Chemometrics and Intelligent Laboratory Systems*, 37-52.

Yüksek, B. B. (2014). Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *The Computer Journal,*.