CARDIFF SCHOOL OF TECHNOLOGIES

4D Digidol | Data | Dylunio | Dyfodol

CIS7031 Programming for Data Analysis

Module Leader: Dr Ambikesh Jayal

Title – Sensor Data Predictive Model Using Neural Network

WRIT1

By

Raghav Gupta

(st20183508)

Data Science (Internship)

Abstract

Human activity acknowledgement joined with deep learning is an invigorating errand yet it is the eventual fate of innovation particularly in the mechanical study of Internet of Things (IoT). AI gives information into as of late grew deep learning draws near and their applications. Deep learning procedures are twice as momentous and capable process with the colossal proportion of the dataset made. The sensor information while performing essential human exercises was gathered on the cell phone with the assistance of MATLAB application which is made by MathWorks. An expert dataset is made from the sensor information recorded while performing fundamental human exercises. Deep learning models are built and the aftereffect of the precision and misfortune is investigated. In profound learning, a model ought to be built in such a way that it restricts the misfortune.

Table of Content

1 Introduction

Human-Activity Recognition (HAR) means to recognize the activities completed by an individual given an informational index of parameters recorded by sensors (Ilisei and Suciu, 2020). HAR intends to distinguish the activities completed by an individual with a bunch of observations of him/herself and the environmental conditions (Reyes-Ortiz et al., 2013). In HAR, different human exercises, for example, sitting, running, walking, standing, cooking, driving, anomalous exercises are perceived. In this project, the sensor data of the three activities is collected. Sensor device used for this project is a smartphone. The use of smartphones with inertial sensors can be a solution for HAR. The information can be gathered from wearable sensors, accelerometer, video casings or images. This type of information is utilized as an assistive technology when a group with different advances like the Internet of Things (IoT) (Jobanputra, Bavishi and Doshi, 2019). In the period of the IoT, a huge measure of sensing gadgets gather and produce exotic sensory information above the long run for a extensive scope of subject and applications. Activity identification build on sensor data incorporates the arising zone of sensor networks with exotic machine learning and data mining methods to demonstrate a extensive scope of human activities. Mechanized gadgets give adequate sensor information and count the ability to empower active work acknowledgement to give an assessment of the energy utilization during regular day to day existence (Mohammadi et al., 2018). Learning requires algorithms and programs that collect data and uncover intriguing or valuable examples. Various business sectors are attempting to adapt a gigantic measure of information implementing machine learning methods. Clients pick machine learning approach to enhance the reason for use. Two stages are utilized in machine learning techniques. Initial stage is training to discover patterns against the collected data. Next stage is evaluating and estimating the obscure information utilizing the learning results (Takami et al., 2016). Deep learning approaches are an all the more remarkable and proficient approach to manage the enormous measure of the information created. Machine learning gives knowledge about enhanced deep learning procedures and their implementations (Zhang et al., 2017).

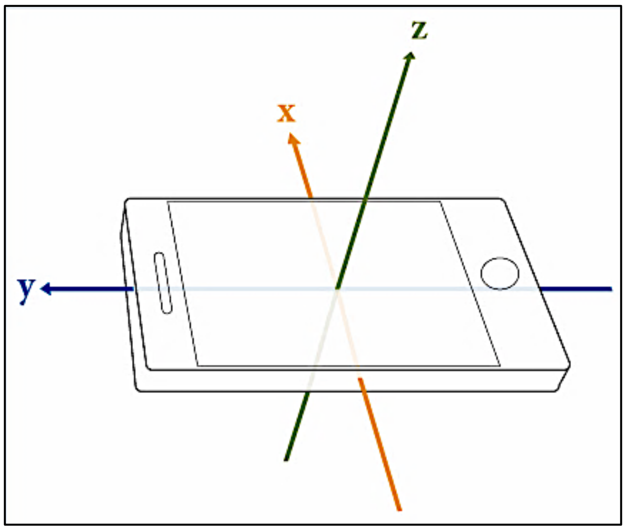
Deep learning is a section of artificial intelligence (AI), which emulates the functionality of a human brain in handling information plus discovering the patterns used in thought processing. Deep learning has networks fit for taking in the unaided form of information known as deep neural network (Hargrave, 2019). Typical neural network(NN) incorporates several straightforward and correlated arrays known as neurons, apiece achieving authentic activations. Input neurons are activated by sensors perceiving the environment and other are activated by weighted alliance from previously active neurons. Deep Learning concerns with precisely allotting attributes across numerous sections (Schmidhuber, 2015). Predictive modelling is additionally utilized in neural networks. It is a process of utilizing realized outcomes to make, measure, and approve a model that can be utilized to figure future results. Two of the most broadly utilized predictive modelling strategies are regression and classification in neural networks (Frankenfield, 2019).

2 Methodology

2.1 Data Collection :

The sensor data while performing basic human activities was collected on the smartphone with the help of MATLAB application which is created by MathWorks. MATLAB consolidates a desktop environment tuned for iterative analysis and configuration measures with a programming language that communicates matrix and array mathematics straightforwardly. MATLAB application enlightens about how various calculations work with the information (uk.mathworks.com, n.d.). It is easy to stream sensor data on MATLAB mobile application as it sends the information continuously with the help of internet to the MathWorks server (uk.mathworks.com, n.d.). While collection sensor data, four attributes were selected, Acceleration studying X, Y and Z values in m/s^2, Magnetic Field studying X, Y and Z values in microtesla, Angular Velocity studying X, Y and Z rotations in radians per second and Orientation studying X, Y and Z values in degrees (see appendix A.1). Figure 1 shows these axes comparative with the smartphone device.

Figure 1: X, Y and Z axis of a smartphone.



Sensor data of three activities were gathered with the smartphone for two minutes each. The three activities are walking with the sensor device, keeping the sensor device stationary and dropping the sensor device. The magnetic field, angular velocity, acceleration and the orientation was recorded for each activity with the help of inbuilt sensors in the MATLAB application.

2.2 Data Preparation and Data Cleaning :

Data preparation is a crucial stage, which includes data preconditioning. The purpose of data preconditioning is to clean, integrate, transform and reduce the original raw data, which is further utilized for analysis. Preparation of dataset into calculative format is essential for predictive modelling. Python language is utilized in data analysis because of its unreservedly accessible tools and libraries (Stančin and Jović, 2019). Major Python libraries used in this project are pandas for data preparation, NumPy, Matplotlib and seaborn for data visualization and TensorFlow, Keras for deep learning and predictive modelling.

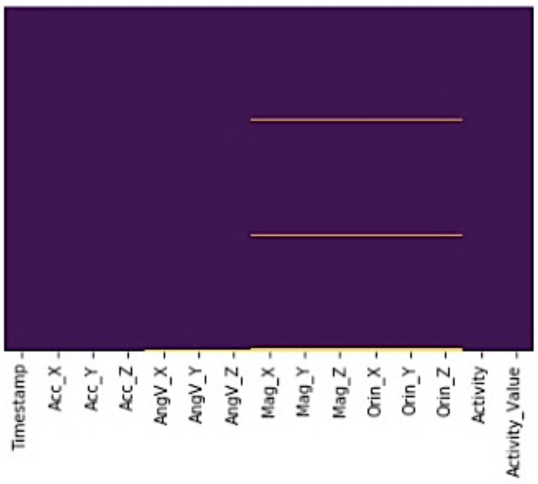
Pandas is the prime and leading Python implementation as it has extensive abilities for input and output formats, such as Excel, CSV, HTML, SQL etc. Additionally, it has strong questioning prospects, statistics computation and some elementary visualizations (Mckinney, 2011). I used pandas for reading the excel files that were created while recording sensor data, to create a master dataset excel file with all three activities combined (see Appendix A.2 – A.5) and to prepare my dataset for further exploration.

TensorFlow is created by Google Brain, it has decent documentation, its functionalities are to a great extent and it is conceivable to make the code very customizable (Usenix Association, 2005). Keras is based on TensorFlow therefore, the coding is exigent and complicated. Keras also has adequate documentation. Keras functions are utilized for generating complicated models with numerous layers (Heller, 2019). TensorFlow and Keras are used in this project for NN and predictive modelling. Matplotlib has plenty of opportunities for personalized plots and graphs. Seaborn library is based on Matplotlib, used for basic visualizations (Stančin and Jović, 2019). Both Matplotlib and seaborn are applied for data exploration plus the visualization of the results.

After generating the master dataset some pre-processing was necessary as it included several null values. The process of data cleaning included detection of the missing values and filling them with actual values. To locate the null values, Heat Map visualization of null values was generated with the help of Matplotlib and seaborn. Figure 2 depicts the heat map of missing values. The Yellow lines in the map reveal the null values in each attribute. It is clearly visible that there are some null values in AngV_X, AngV_Y, AngV_Z, Mag_X, Mag_Y, Mag_Z, Orin_X, Orin_Y and Orin_Z.

Figure 2: Heat map of the master dataset.

```
In [32]: sns.heatmap(df_master_grp.isnull(),yticklabels=False,cbar=False,cmap='viridis')
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcdbe517310>
```



The missing values of the dataset were filled step by step and a heat map of no null values was generated (see Appendix A.6 – A.8). Figure 3 displays the information of the data frame and its attributes. Also, it informs that there are no null values and the dataset is now prepared for further explorations.

Figure 3: Information about the Dataset.

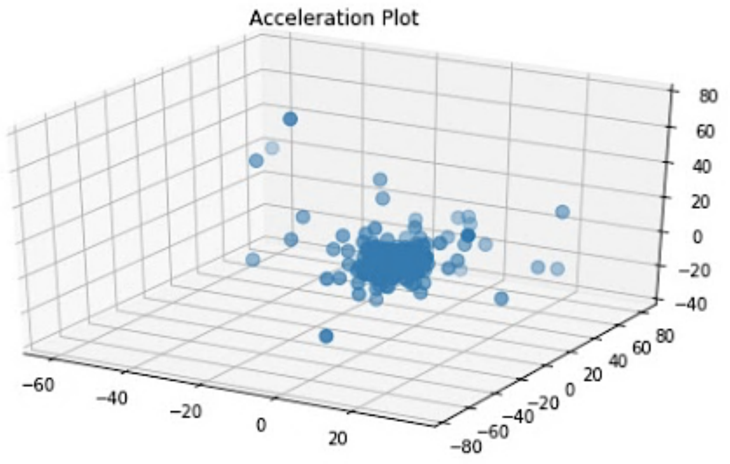
```
In [46]: df_master_grp.info() #no null values, therefore all data cleaned

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 3640 entries, 0 to 3639
         Data columns (total 15 columns):
         Timestamp         3640 non-null datetime64[ns]
         Acc_X             3640 non-null float64
         Acc_Y             3640 non-null float64
         Acc_Z             3640 non-null float64
         AngV_X            3640 non-null float64
         AngV_Y            3640 non-null float64
         AngV_Z            3640 non-null float64
         Mag_X             3640 non-null float64
         Mag_Y             3640 non-null float64
         Mag_Z             3640 non-null float64
         Orin_X            3640 non-null float64
         Orin_Y            3640 non-null float64
         Orin_Z            3640 non-null float64
         Activity          3640 non-null object
         Activity_Value    3640 non-null int64
         dtypes: datetime64[ns](1), float64(12), int64(1), object(1)
         memory usage: 426.7+ KB
```

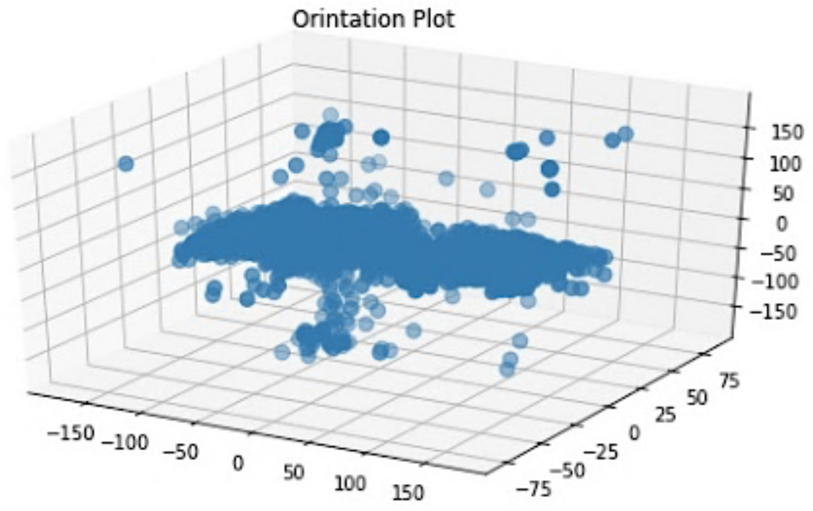
2.3 Data Exploration:

In this project, the data is explored in different types visualizations depicting the values and relations between the X. Y and Z-axis.

1) 3-D Scatter Plot - It is a numerical outline and the fundamental adaptation of three-dimensional arrangement to show the values of information utilizing the cartesian coordinates. Matplotlib's mplot3-D toolkit is utilized to create a 3-D scatter plot. It is generated by utilizing ax.scatter3-D() capacity of the matplotlib library acknowledging an informational collection of values to represent, whereas remaining values of the capacity are equivalent to 2D dissipate plot (Gavande and Yadav, 2020). Figure 4 and 5 are the 3-D scatter plots of acceleration and orientation respectively.

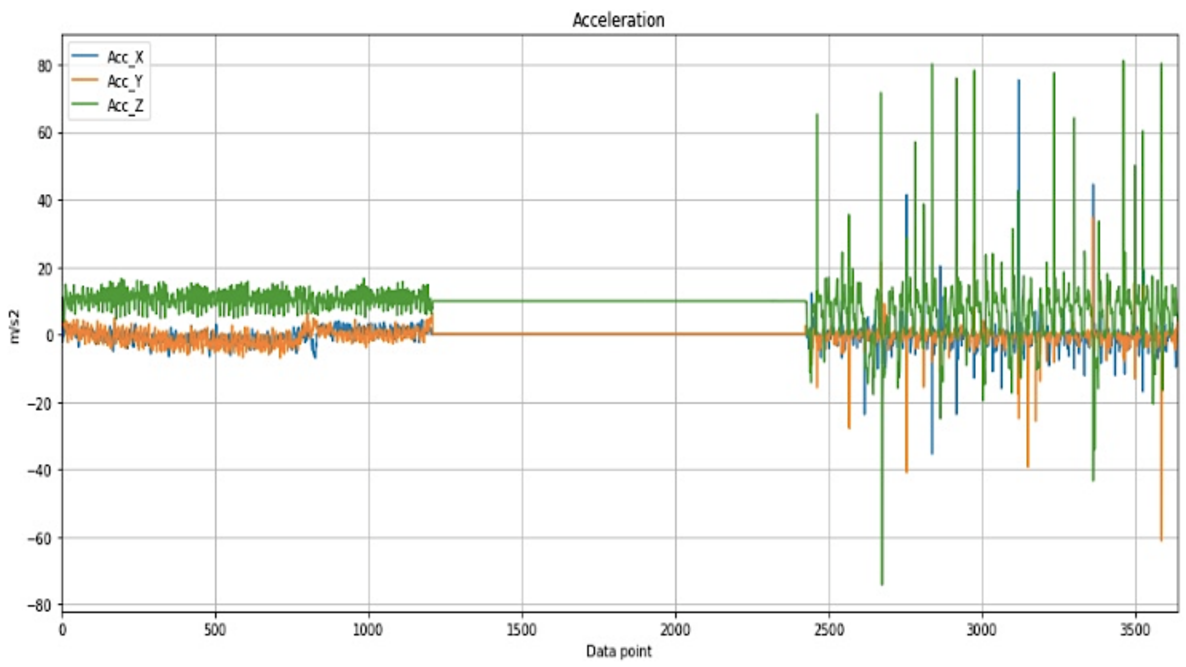
Figure 4 and 5: Acceleration and Orientation 3-D scatter plot respectively.





2) Set Graph or Linear Graph – Set Graph exhibits information as an arrangement of information focus known as 'markers' attached by straight-line sections. Figure 6, 7, 8 and 9 are the set graphs of acceleration, angular velocity, magnetic field and orientation generated from the sensor data of the three activities performed. From figure 6 we can understand that three different colours represent three different relations of acceleration on each axis. In figure 6 the linear representation from data point 0 to 1213 is for the first activity walking, the linear representation from data point 1214 to 2426 is for the second activity stationary and the last representation from data point 2427 to 3639 is for the third activity dropping. Similarly, the other graphs are represented in figure 7, 8 and 9.

Figure 6 and 7: Acceleration and Angular Velocity linear graph respectively.



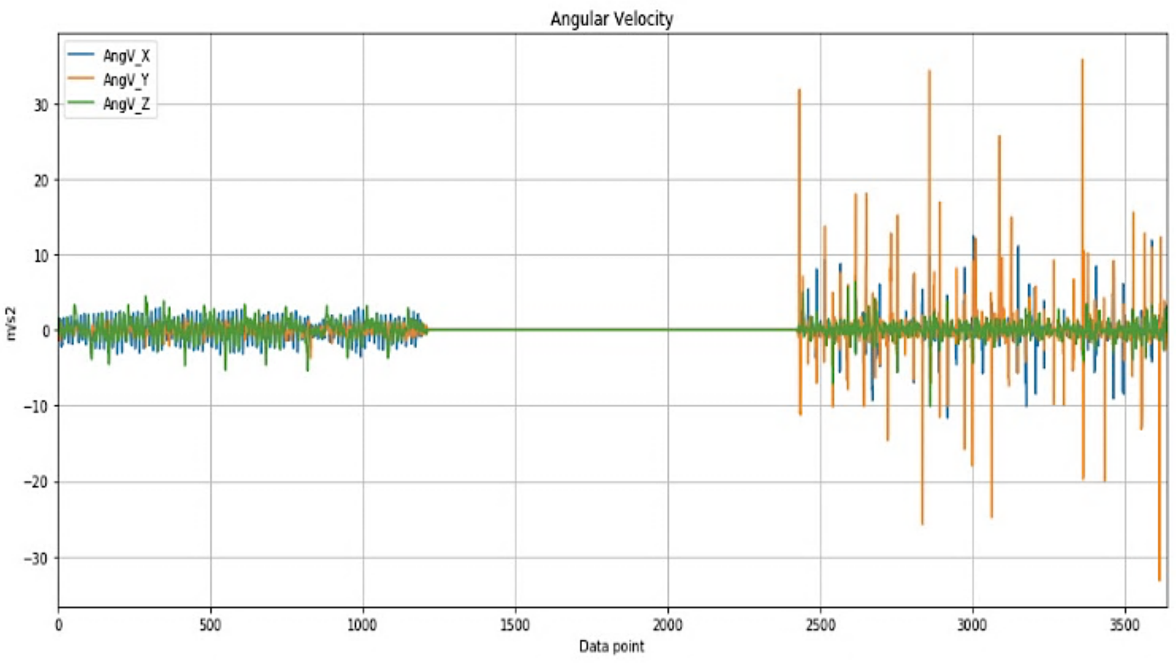
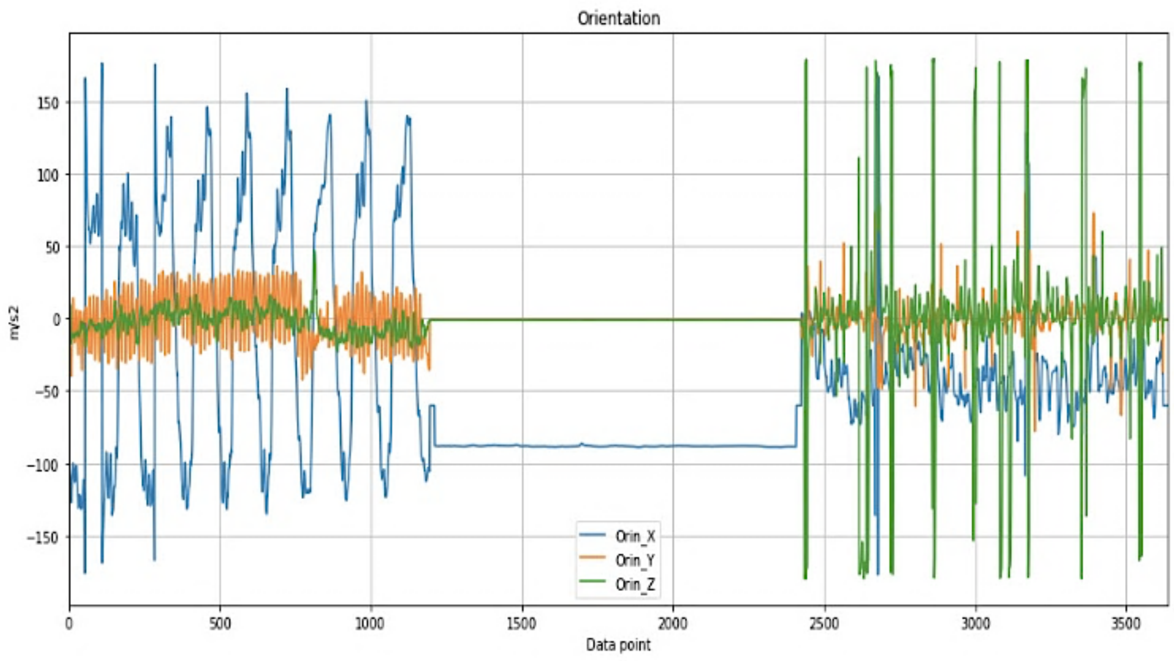
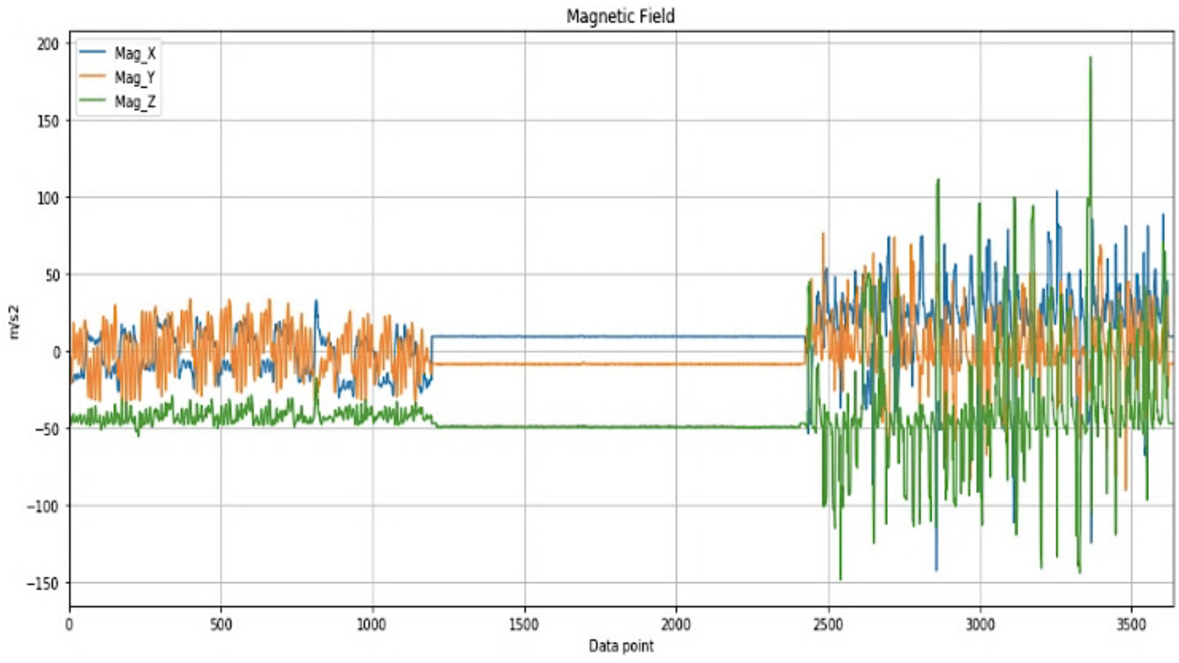


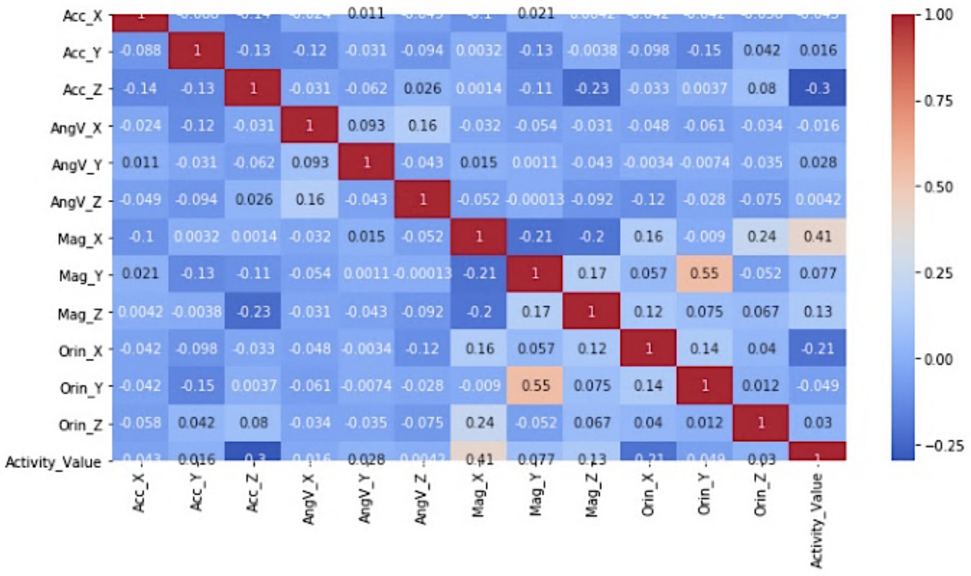
Figure 8 and 9: Orientation and Magnetic Field linear graph respectively.





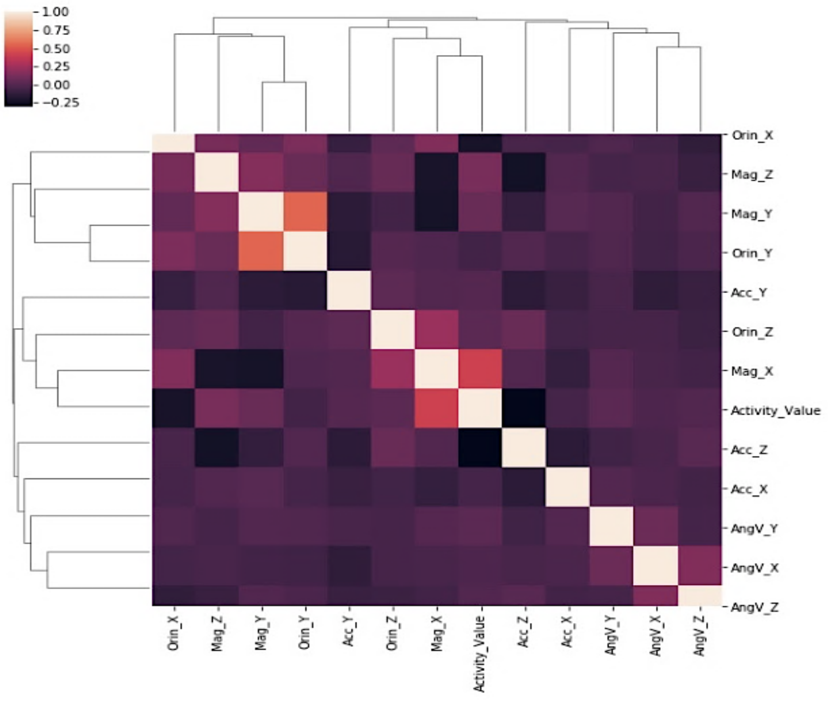
3) Heat Map – It is a 2-D portrayal of information where independent qualities are embraced in a grid, known as shadings. Python seaborn bundle permits the making of commented on heatmaps which can be changed utilizing Matplotlib apparatuses according to the maker's prerequisite (Paradkar, 2016). Figure 10 illustrates the heat map of the dataset with different shades representing different values of each attribute.

Figure 10: Heat map



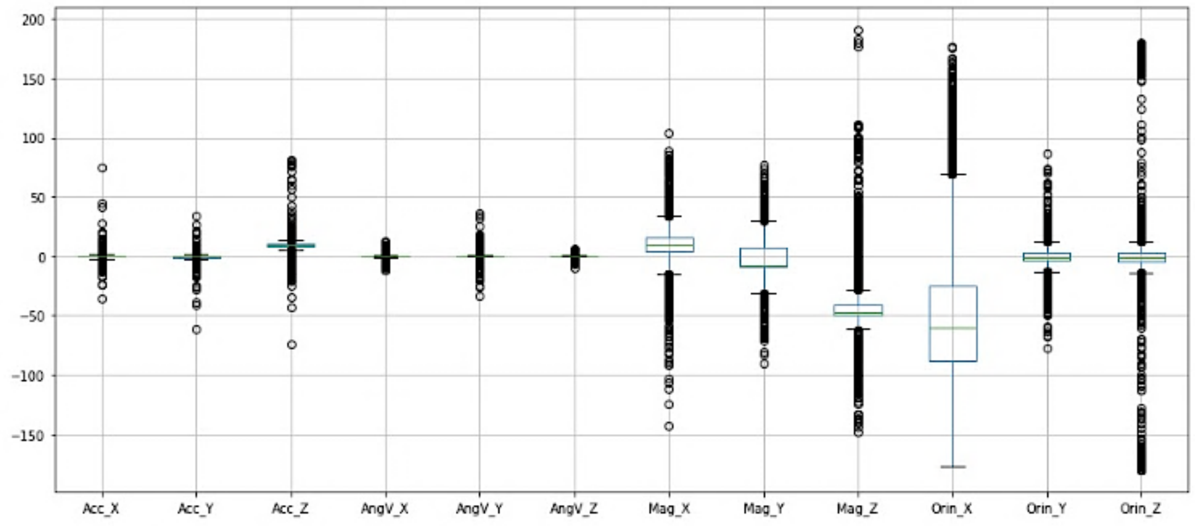
4) Cluster Map – In this map list of colours are utilized to the label for either the rows or columns. This map is useful to evaluate whether samples within a group are clustered together. Figure 11 shows the cluster map of the dataset.

Figure 11: Cluster map



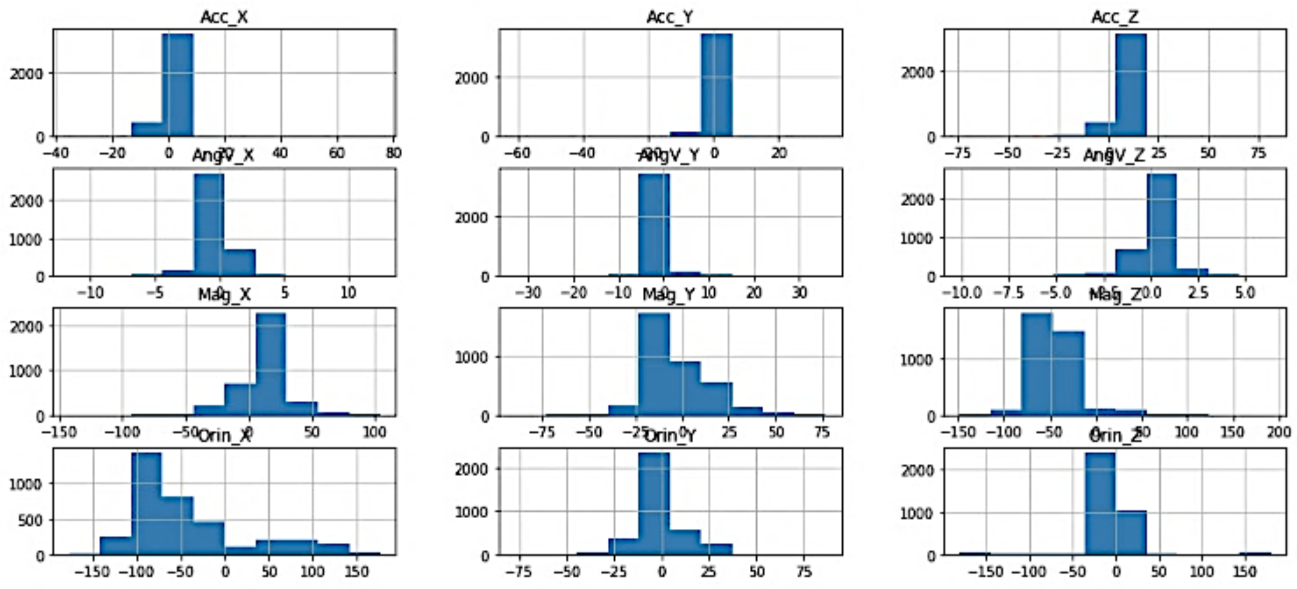
5) Box Plot – This plot graphically represents the organized information by way of divisions. Box expands from first to third division values of the data, separated by a second division line in the middle. Lines inside the box increase initiated at the end of the box to depict the span of the data. Figure 12 displays the box plot graph representing master dataset.

Figure 12: Box Plot of Dataset.



6) Histogram – This graph represents the frequency on both vertical and horizontal axis. It also shows the minimum and maximum data value. Figure 13 displays the histograms of all the attributes of the dataset and their respective maximum and minimum value.

Figure 13: Histogram of each attribute in a dataset.



2.4 Developing A Predictive Model

The main utilization of artificial neural network is demonstrating a framework with an obscure information yield connection. In AI, classification stands for a predictive modelling issue where a class mark is anticipated for a given illustration of information. It is necessary for the classification model to train a dataset with numerous instances of inputs and outputs to grasp the information. Two classification models are created for the project. A model utilizes the trained dataset to compute how to best guide instances of information to explicit class names (Brownlee, 2020).In this project the dataset was split into 6 aspects X_test and Y_test for testing model, X_train and Y_train for training model and X_val and Y_val for validating model (see Appendix A.9). Preparing a model basically implies absorbing great qualities from the bias and weights labelled in class. In deep learning, a model should be constructed in such a manner that it limits the loss. Loss is a quantity demonstrating how terrible the model's anticipation is on the particular epoch. Zero loss represents the prediction is exemplary otherwise, imperfect. The aim of developing a predictive classification model is to roughly decrease the loss on each epoch (Descending into ML: Training and Loss | Machine Learning Crash Course, n.d.). Epoch is an appellation utilized in AI that demonstrates the progress of entire planning dataset the AI calculation has wrapped up. The dataset is gathered into batches and each batch is then processed in a single epoch (Gaillard and Bell, 2020).

Activation functions are numerical conditions, which decides the result of the neural network. These function are connected with every neuron and decides if it ought to be initiated or not, founded on even if every neuron's input is important for prediction. The numeric data points, also known as inputs, are added to neurons of the input layer. Since every neuron has weight therefore, the result of the input value multiplied with weight is the output of neuron, which is moved to the following layer. In deep learning, neural network models utilize non-linear activation functions, which are crucial for learning and modelling complex information (MissingLink.ai, n.d.). The two major activation functions used in constructing the models of the project are -

1. ReLu: Rectified Linear Unit function permits the network to merge rapidly and allows backpropagation.
2. Softmax: This function controls multiple classes and normalizes the output. It is also required for the output layer, which needs to classify inputs into many categories.

The first model is a simple NN model, which consists of single input layer, hidden layer and output layer (see Appendix A.10). The second model is deep NN model, which consists of one input layer, four hidden layers and one output layer (see Appendix A.11).

While making an AI model, it is difficult to make some decisions with respect to how to characterize the model. It is hard and time-consuming to create an ideal model with a large range of possibilities. This is where the process of hyperparameter tuning comes into the picture. In hyperparameter tuning, the machine plays out an investigation and select the parameters for the ideal model. Grid search is one of the fundamental hyperparameter tuning strategies. This technique is used for assembling the model for every conceivable mix of the entirety of the hyperparameters values supplied, and determining the design which delivers the best outcomes (Jordan, 2017).
The hyperparameters selected from this technique in this project are the number of epochs, batch size and the optimizer for both the models (see Appendix A.12 and A.13).

3 Results and Analysis

One of the significant standards utilized in choosing a classification model with respect to sensor data is the precision or accuracy of the data it can give. The classification accuracy is evaluated to see the performance of the model depends on the predicted results (Aronoff, 1982).

3.1 Model 1

According to the results from hyperparameter tuning, model 1 should use 50 epochs with 20 batch size and Adam optimizer (see Appendix A.14) to run ideally. Executing using these hyperparameters, the loss calculated is 0.00895 and the accuracy obtained is 99.58. This result signifies that the classification model 1 is successful in predicting the results with minimum loss. Figure 14 visualizes the results of the model. This figure depicts the loss generated while training and validating the model and the difference between the two losses in each epoch.

Figure 14: Graph representation of Loss in model 1.

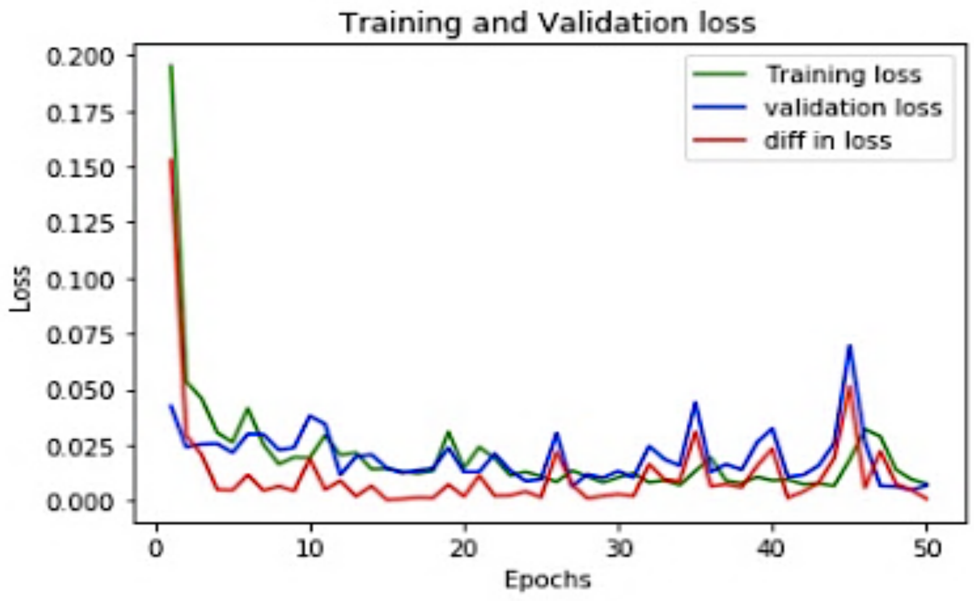
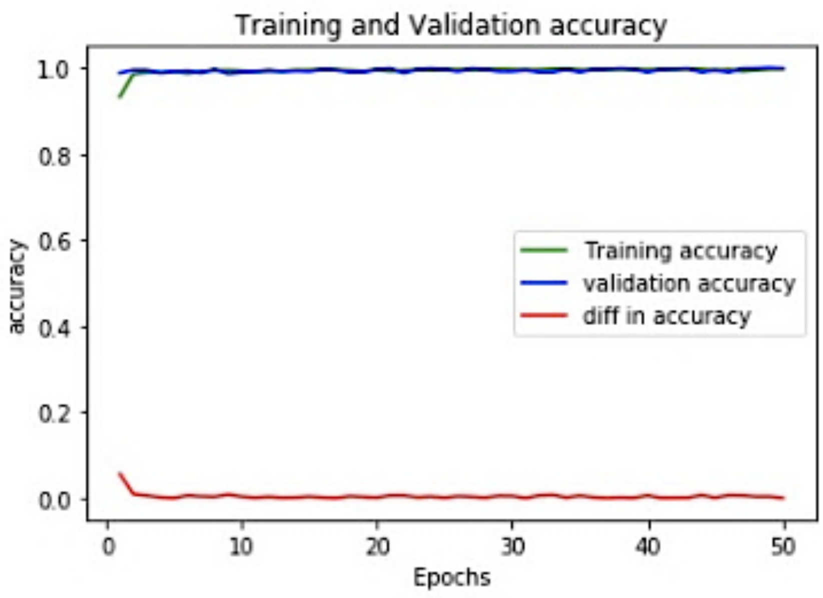


Figure 15 displays the accuracy obtained while training and validating the model and the difference between the accuracies in each epoch.

Figure 15: Graph representation of Accuracy of model 1.



3.2 Model 2

According to the results from hyperparameter tuning, model 2 should use 600 epochs with 100 batch size and Adamax optimizer (see Appendix A.15) to run ideally. Executing using these hyperparameters, the loss calculated is 0.02718 and the accuracy obtained is 99.72. This result signifies that the classification model 2 is also successful in predicting the results with minimum loss. Figure 16 depicts the loss in each epoch similar to figure 14.

Figure 16: Graph representation of Loss in model 2.

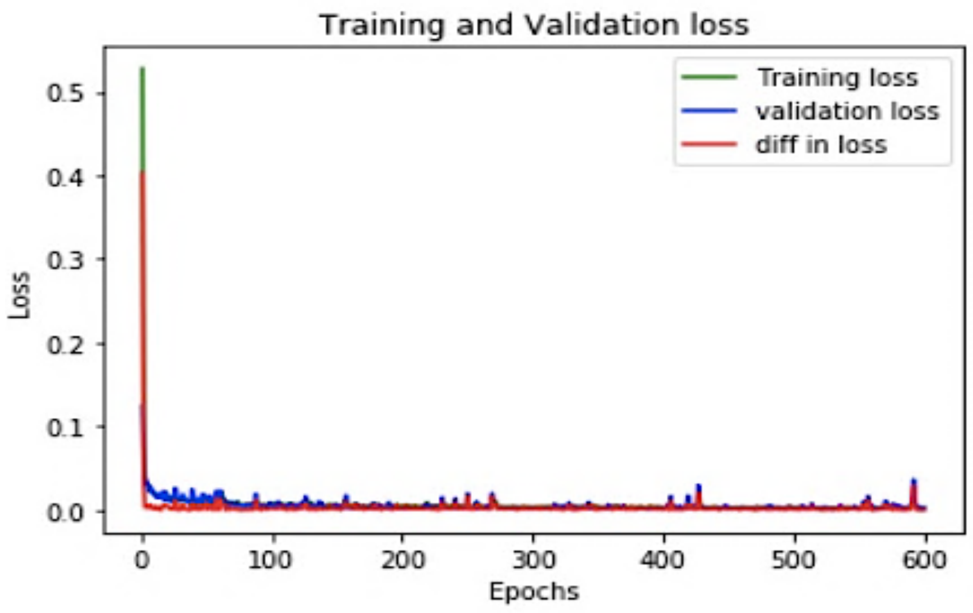
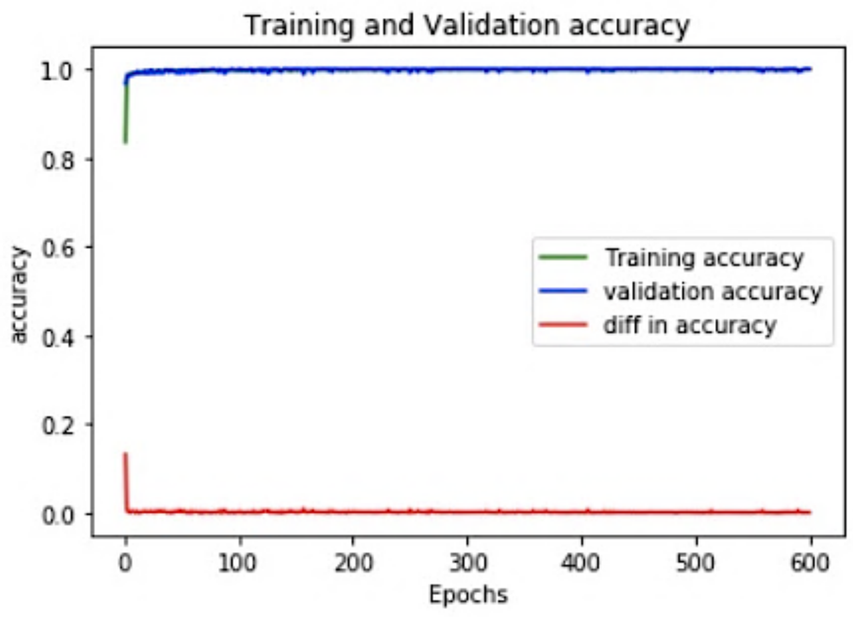


Figure 17 displays the training and validation accuracy in each epoch similar to figure 15.

Figure 17: Graph representation of Accuracy of model 2.



Both models have successfully shown positive results. The difference between the hyperparameter results is due to the number of hidden layers in each model. Since model 2 has more hidden layers, making a deep neural network, the amount of epochs and the batch size is much greater than model 1. The loss in model 2 is slightly more than model 1 whereas, the accuracy obtained in model 2 is greater than model 1. There is a variable difference in training and validation loss in both the models in every epoch, however the difference in training and validation accuracy of both models is nearly zero in each epoch.

4 Critical Review and Analysis of Techniques Used

A deep learning neural network predictive modelling process majorly includes ten components:

1. Collecting and pre-processing data.
2. Data preparation and data cleaning.
3. Exploratory analysis.
4. Selecting predictive methods.
5. Selecting hyperparameters.
6. Creating model.
7. Validating model.
8. Assessing loss and accuracy.
9. Classification predictions and prediction variability.
10. Visualization.

To adopt the most perfect predictive model, all components and their appropriate requirements and elements are necessary to be considered and precisely implemented. For HAR project, more than two models could have been constructed with many hidden layers and different activation functions. Prediction accuracy is the most crucial criteria for selecting the ideal model and for classification anticipations (Li, 2019).

The grid search technique used in hyperparameter tuning can be a shallow approach as it does not always provide the best solution. This technique is divided into two parts in this project – first, deciding the best solution for the number of epochs and the batch size and second, using the best solution provided by the first step to decide the appropriate optimizer. Grid search technique is selecting the best solution from the initial parameters provided. For example, it is selecting the number of epochs only from the range [10, 50, 100, 200, 400, 600] however, there can be an infinite range with huge values. Similarly, for batch size, there are numerous options to select the batch size. Since the solution from the first step is not optimal therefore, selecting the best optimizer is also not trustworthy.

The data collection could have been better as in this project three activities are performed only for two minutes. Many other basic human activities can be formed and for a greater period of time.

5 Conclusion

In this project, various techniques and tools of classification predictive modelling are discussed, which can be further utilized for human activity recognition in a neural network system. A master dataset is created from the sensor data recorded while performing basic human activities. Deep learning models are constructed and the result of the accuracy and loss is analysed. It can be inferred from this project that model with the higher number of hidden layers obtain a higher value of accuracy but also with questionable loss. There are some limitations from which it can be concluded that there are many other ways to predict the recognition of human activity. Human activity recognition combined with deep learning is a stimulating task yet it is the future of technology especially in industrial science of Internet of Things (IoT).

6 Video link

https://www.youtube.com/watch?v=ix8FlNWMcgM

7 References

Aronoff, S. (1982). *Classification Accuracy: A User Approach*. [online] Available at: http://www.asprs.org/wp-content/uploads/pers/1982journal/aug/1982_aug_1299-1307.pdf [Accessed 20 Jan. 2021].

Brownlee, J. (2020). *4 Types of Classification Tasks in Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/types-of-classification-in-machine-learning/#:~:text=In%20machine%20learning%2C%20classification%20refers.

Descending into ML: Training and Loss | Machine Learning Crash Course. (n.d.). *Google Developers*. [online] Available at: https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss#:~:text=Loss%20is%20the%20penalty%20for [Accessed 20 Jan. 2021].

Frankenfield, J. (2019). *Reading Into Predictive Modeling*. [online] Investopedia. Available at: https://www.investopedia.com/terms/p/predictive-modeling.asp [Accessed 20 Jan. 2021].

Gaillard, F. and Bell, D.D.J. (2020). *Epoch (machine learning) | Radiology Reference Article | Radiopaedia.org*. [online] Radiopaedia. Available at: https://radiopaedia.org/articles/epoch-machine-learning?lang=gb#:~:text=An%20epoch%20is%20a%20term [Accessed 20 Jan. 2021].

Gavande, J. and Yadav, K. (2020). *3D Scatter Plotting in Python using Matplotlib*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/3d-scatter-plotting-in-python-using-matplotlib/#:~:text=Generally%203D%20scatter%20plot%20is [Accessed 20 Jan. 2021].

Hargrave, M. (2019). *How Deep Learning Can Help Prevent Financial Fraud*. [online] Investopedia. Available at: https://www.investopedia.com/terms/d/deep-learning.asp#:~:text=Deep%20learning%20is%20an%20AI [Accessed 20 Jan. 2021].

Heller, M. (2019). *What is Keras? The deep neural network API explained*. [online] InfoWorld. Available at: https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html [Accessed 20 Jan. 2021].

Ilisei, D. and Suciu, D.M. (2020). Human-Activity Recognition with Smartphone Sensors. *On the Move to Meaningful Internet Systems: OTM 2019 Workshops*, pp.179–188.

Jobanputra, C., Bavishi, J. and Doshi, N. (2019). Human Activity Recognition: A Survey. *Procedia Computer Science*, [online] 155, pp.698–703. Available at: https://www.sciencedirect.com/science/article/pii/S1877050919310166.

Jordan, J. (2017). *Hyperparameter tuning for machine learning models.* [online] Jeremy Jordan. Available at: https://www.jeremyjordan.me/hyperparameter-tuning/ [Accessed 20 Jan. 2021].

Li, J. (2019). A Critical Review of Spatial Predictive Modeling Process in Environmental Sciences with Reproducible Examples in R. *Applied Sciences*, 9(10), p.2048.

Mckinney, W. (2011). *pandas: a Foundational Python Library for Data Analysis and Statistics*. [online] Available at: https://www.dlr.de/sc/portaldata/15/resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf [Accessed 20 Jan. 2021].

MissingLink.ai. (n.d.). *7 Types of Activation Functions in Neural Networks: How to Choose?* [online] Available at: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/#:~:text=Activation%20functions%20are%20mathematical%20equations [Accessed 20 Jan. 2021].

Mohammadi, M., Al-Fuqaha, A., Sorour, S. and Guizani, M. (2018). Deep Learning for IoT Big Data and Streaming Analytics: A Survey. *IEEE Communications Surveys & Tutorials*, 20(4), pp.2923–2960.

Paradkar, M. (2016). *Using Seaborn Python Package for Creating Heatmap*. [online] QuantInsti. Available at: https://blog.quantinsti.com/creating-heatmap-using-python-seaborn/#:~:text=A%20heatmap%20is%20a%20two [Accessed 20 Jan. 2021].

Reyes-Ortiz, J.-L., Oneto, L., Samà, A., Parra, X. and Anguita, D. (2013). Transition-Aware Human Activity Recognition Using Smartphones. *Neurocomputing*, [online] 171, pp.754–767. Available at: https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2013-84.pdf.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, pp.85–117.

Stančin, I. and Jović, A. (2019). *An overview and comparison of free Python libraries for data mining and big data analysis*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/8757088?casa_token=ESV1wj50ZtcAAAAA:_E05u0-rrqqgeP0Ue9KA6M3eEahi8L7g6J1Vx1rPYn2yViF-ysTdedjywYk73KN8PcsIkl4p [Accessed 20 Jan. 2021].

Takami, G., Tokuoka, M., Goto, H. and Nozaka, Y. (2016). *Machine Learning Applied to Sensor Data Analysis Machine Learning Applied to Sensor Data Analysis*. [online] Available at: https://web-material3.yokogawa.com/rd-te-r05901-006.pdf [Accessed 20 Jan. 2021].

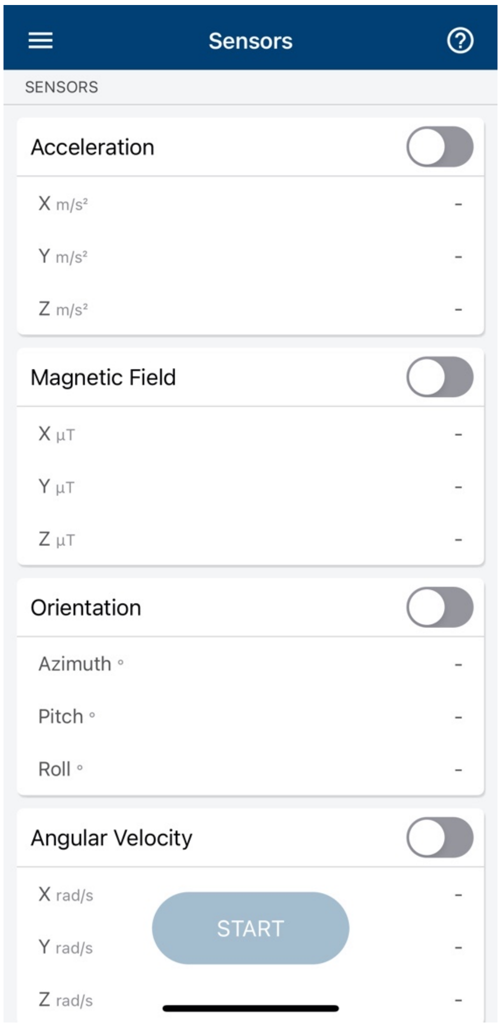
uk.mathworks.com. (n.d.). *MATLAB - MathWorks*. [online] Available at: https://uk.mathworks.com/products/matlab.html [Accessed 20 Jan. 2021].

Usenix Association (2005). Papers presented at the 2005 workshop on Wireless traffic measurements and modeling. [online] Available at: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi [Accessed 20 Jan. 2021].

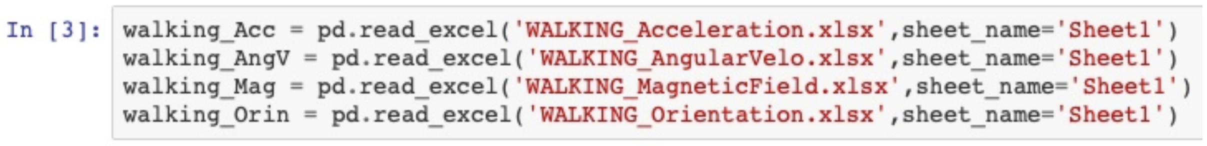
Zhang, L., Tan, J., Han, D. and Zhu, H. (2017). From machine learning to deep learning: progress in machine intelligence for rational drug discovery. *Drug Discovery Today*, 22(11), pp.1680–1685.

8 Appendix

A.1 MATLAB mobile application sensors.

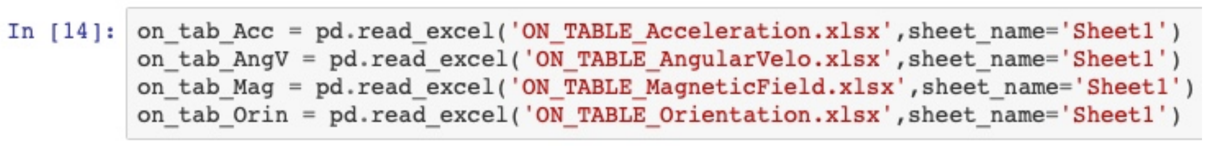


A.2 Reading excel files of first activity.



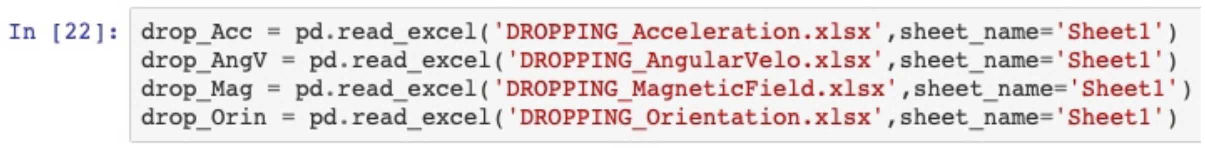
```
In [3]: walking_Acc = pd.read_excel('WALKING_Acceleration.xlsx',sheet_name='Sheet1')
        walking_AngV = pd.read_excel('WALKING_AngularVelo.xlsx',sheet_name='Sheet1')
        walking_Mag = pd.read_excel('WALKING_MagneticField.xlsx',sheet_name='Sheet1')
        walking_Orin = pd.read_excel('WALKING_Orientation.xlsx',sheet_name='Sheet1')
```

A.3 Reading excel files of second activity.



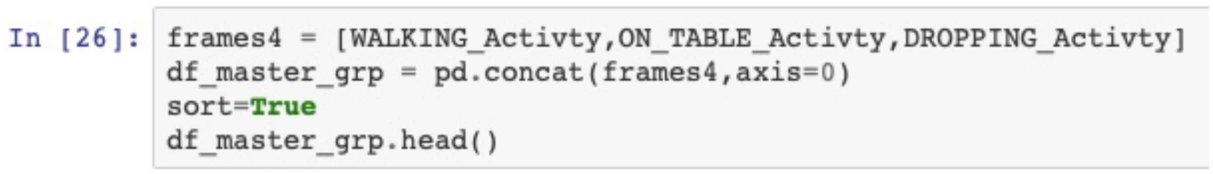
```
In [14]: on_tab_Acc = pd.read_excel('ON_TABLE_Acceleration.xlsx',sheet_name='Sheet1')
         on_tab_AngV = pd.read_excel('ON_TABLE_AngularVelo.xlsx',sheet_name='Sheet1')
         on_tab_Mag = pd.read_excel('ON_TABLE_MagneticField.xlsx',sheet_name='Sheet1')
         on_tab_Orin = pd.read_excel('ON_TABLE_Orientation.xlsx',sheet_name='Sheet1')
```

A.4 Reading excel files of third activity.



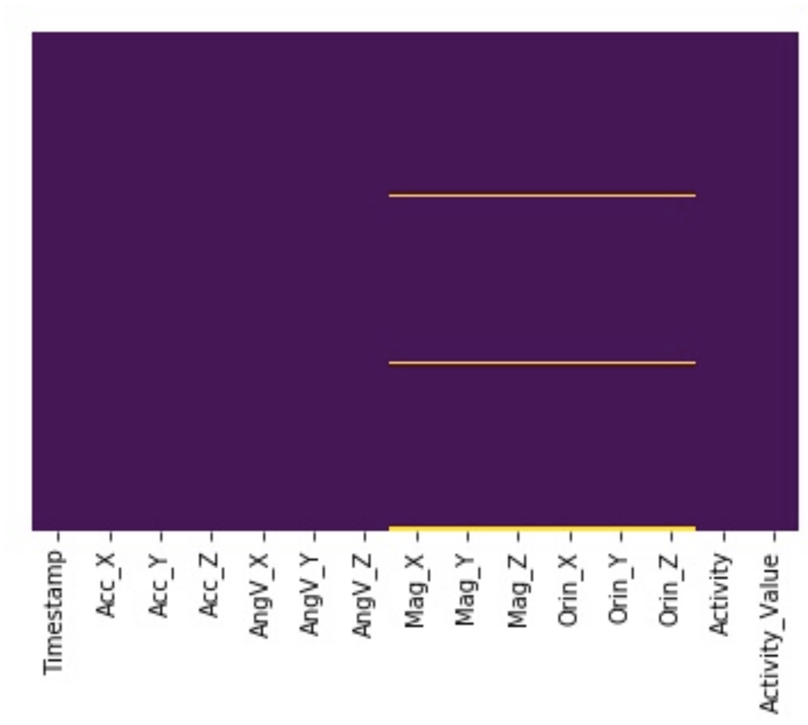
```
In [22]: drop_Acc = pd.read_excel('DROPPING_Acceleration.xlsx',sheet_name='Sheet1')
         drop_AngV = pd.read_excel('DROPPING_AngularVelo.xlsx',sheet_name='Sheet1')
         drop_Mag = pd.read_excel('DROPPING_MagneticField.xlsx',sheet_name='Sheet1')
         drop_Orin = pd.read_excel('DROPPING_Orientation.xlsx',sheet_name='Sheet1')
```

A.5 Combining all three activities to master dataset.

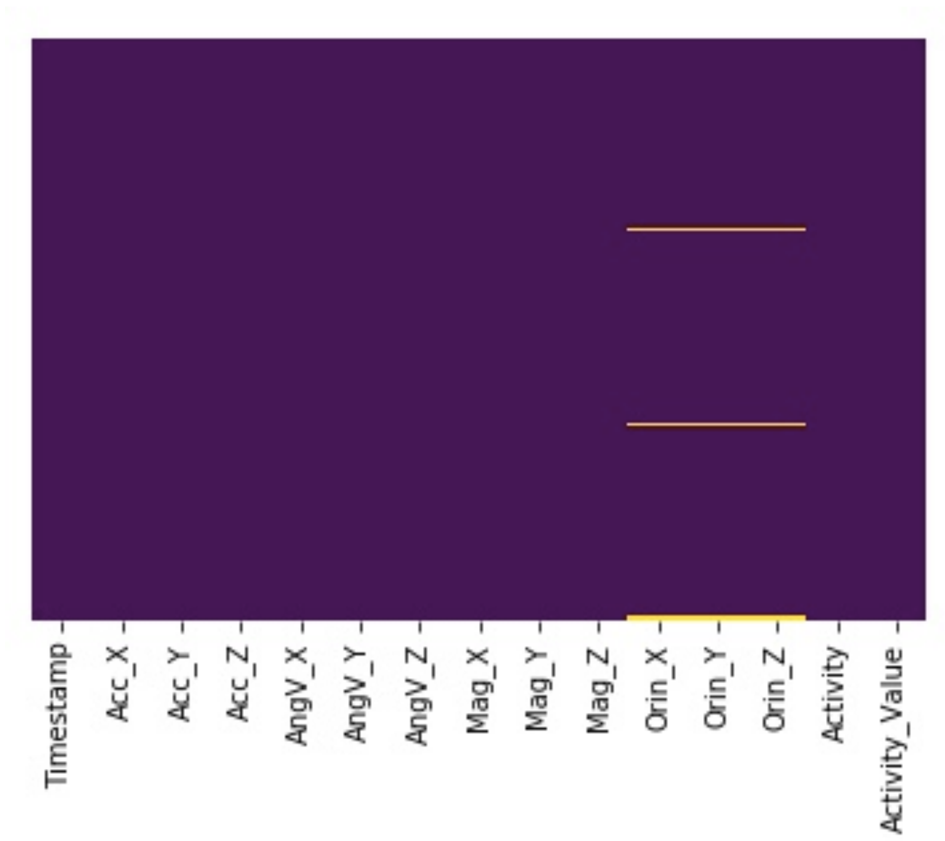


```
In [26]: frames4 = [WALKING_Activty,ON_TABLE_Activty,DROPPING_Activty]
         df_master_grp = pd.concat(frames4,axis=0)
         sort=True
         df_master_grp.head()
```

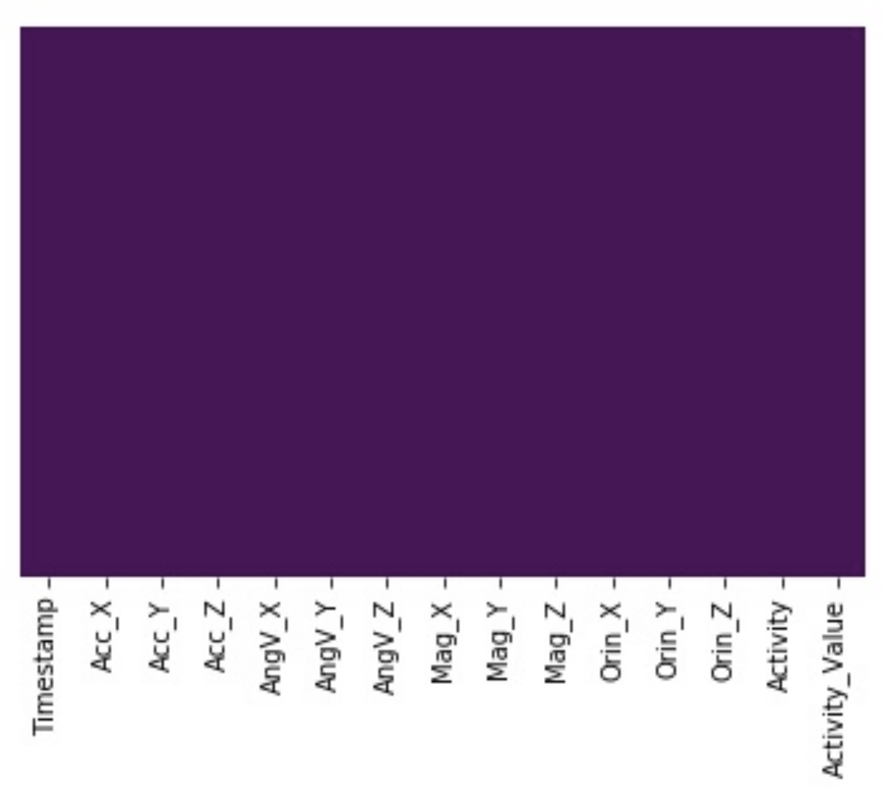
A.6 Heat map with no null values in AngV_X, AngV_Y and AngV_Z.



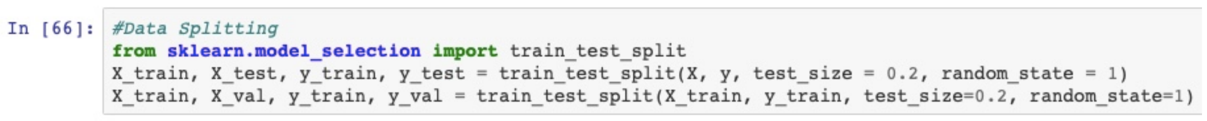
A.7 Heat map with no null values in Mag_X, Mag_Y and Mag_Z.



A.8 Heat map with no null values in Orin_X, Orin_Y and Orin_Z.

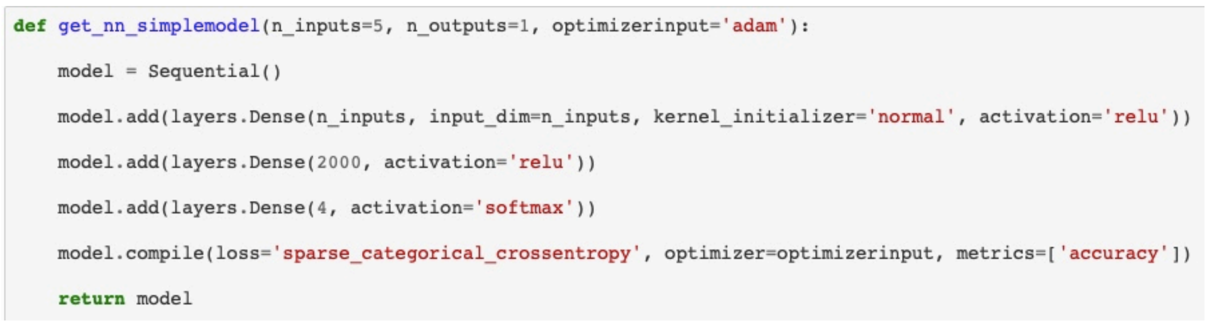


A.9 Splitting the dataset.



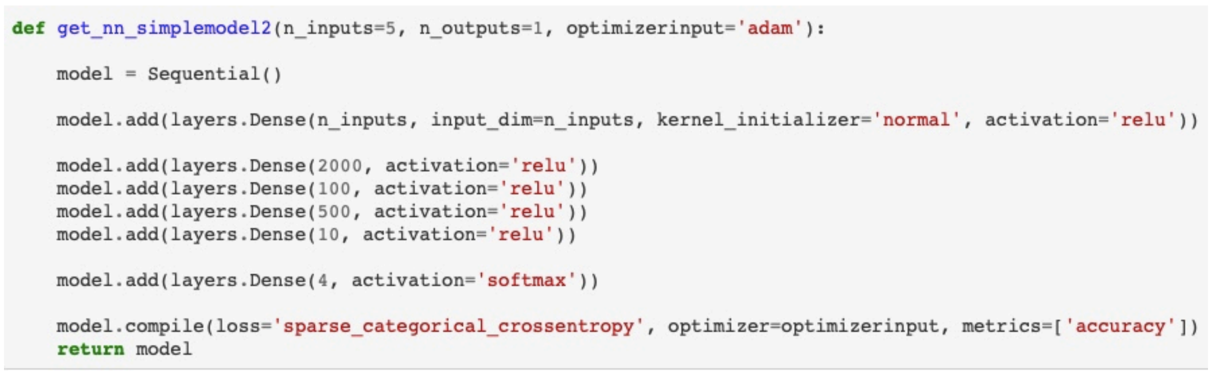
```
In [66]: #Data Splitting
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 1)
         X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=1)
```

A.10 Model 1



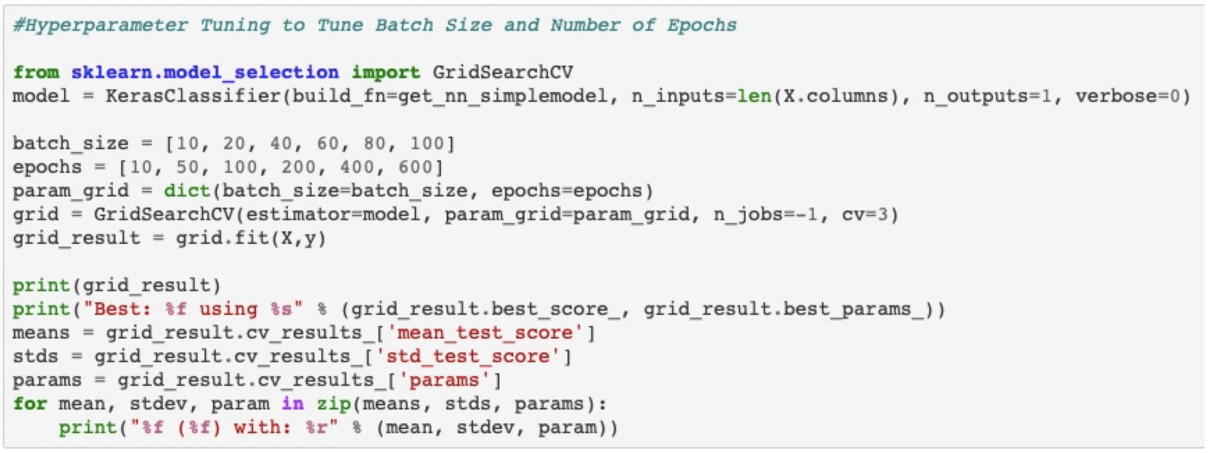
```
def get_nn_simplemodel(n_inputs=5, n_outputs=1, optimizerinput='adam'):

    model = Sequential()

    model.add(layers.Dense(n_inputs, input_dim=n_inputs, kernel_initializer='normal', activation='relu'))

    model.add(layers.Dense(2000, activation='relu'))

    model.add(layers.Dense(4, activation='softmax'))

    model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizerinput, metrics=['accuracy'])

    return model
```

A.11 Model 2



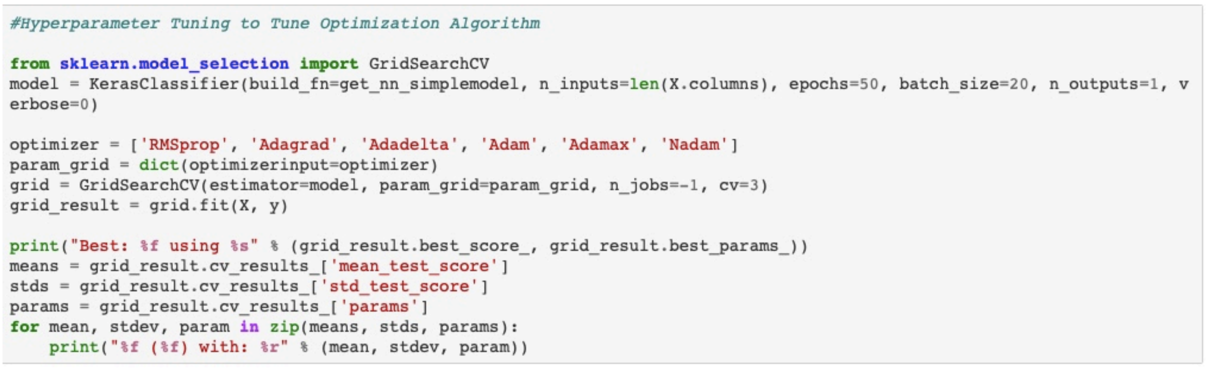
```
def get_nn_simplemodel2(n_inputs=5, n_outputs=1, optimizerinput='adam'):

    model = Sequential()

    model.add(layers.Dense(n_inputs, input_dim=n_inputs, kernel_initializer='normal', activation='relu'))

    model.add(layers.Dense(2000, activation='relu'))
    model.add(layers.Dense(100, activation='relu'))
    model.add(layers.Dense(500, activation='relu'))
    model.add(layers.Dense(10, activation='relu'))

    model.add(layers.Dense(4, activation='softmax'))

    model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizerinput, metrics=['accuracy'])
    return model
```

A.12 Hyperparameter tuning for number of epochs and batch size.



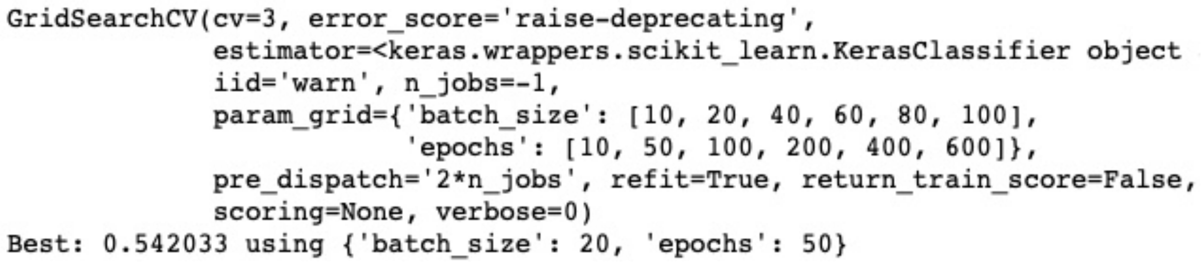
```
#Hyperparameter Tuning to Tune Batch Size and Number of Epochs

from sklearn.model_selection import GridSearchCV
model = KerasClassifier(build_fn=get_nn_simplemodel, n_inputs=len(X.columns), n_outputs=1, verbose=0)

batch_size = [10, 20, 40, 60, 80, 100]
epochs = [10, 50, 100, 200, 400, 600]
param_grid = dict(batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X,y)

print(grid_result)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

A.13 Hyperparameter tuning for optimizer.

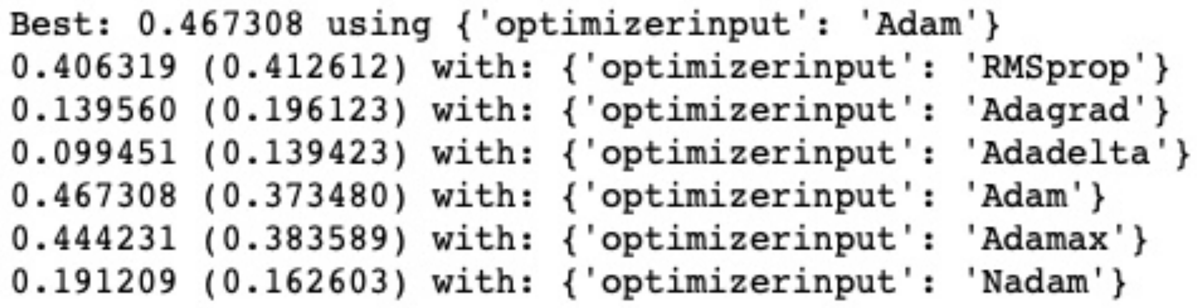


```
#Hyperparameter Tuning to Tune Optimization Algorithm

from sklearn.model_selection import GridSearchCV
model = KerasClassifier(build_fn=get_nn_simplemodel, n_inputs=len(X.columns), epochs=50, batch_size=20, n_outputs=1, v
erbose=0)

optimizer = ['RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
param_grid = dict(optimizerinput=optimizer)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X, y)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

A.14 Result of hyperparameter tuning for model 1.

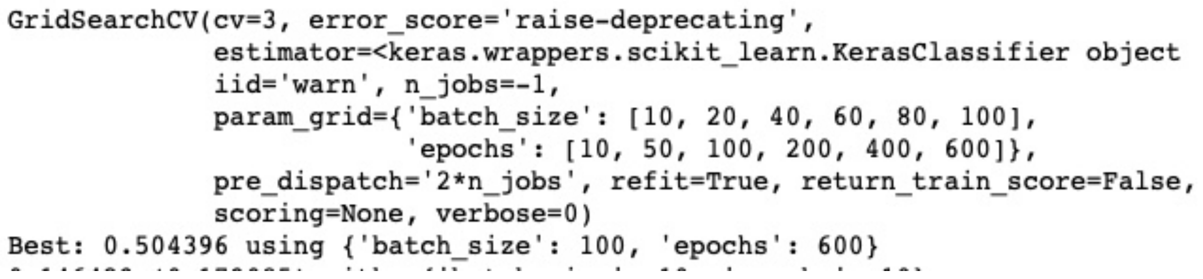


```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=<keras.wrappers.scikit_learn.KerasClassifier object
             iid='warn', n_jobs=-1,
             param_grid={'batch_size': [10, 20, 40, 60, 80, 100],
                         'epochs': [10, 50, 100, 200, 400, 600]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
Best: 0.542033 using {'batch_size': 20, 'epochs': 50}
```

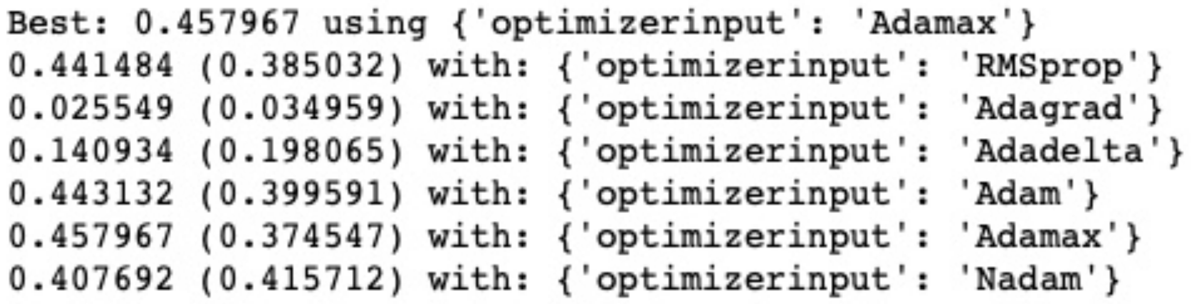


```
Best: 0.467308 using {'optimizerinput': 'Adam'}
0.406319 (0.412612) with: {'optimizerinput': 'RMSprop'}
0.139560 (0.196123) with: {'optimizerinput': 'Adagrad'}
0.099451 (0.139423) with: {'optimizerinput': 'Adadelta'}
0.467308 (0.373480) with: {'optimizerinput': 'Adam'}
0.444231 (0.383589) with: {'optimizerinput': 'Adamax'}
0.191209 (0.162603) with: {'optimizerinput': 'Nadam'}
```

A.15 Result of hyperparameter tuning for model 2.



```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=<keras.wrappers.scikit_learn.KerasClassifier object
             iid='warn', n_jobs=-1,
             param_grid={'batch_size': [10, 20, 40, 60, 80, 100],
                         'epochs': [10, 50, 100, 200, 400, 600]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
Best: 0.504396 using {'batch_size': 100, 'epochs': 600}
```



```
Best: 0.457967 using {'optimizerinput': 'Adamax'}
0.441484 (0.385032) with: {'optimizerinput': 'RMSprop'}
0.025549 (0.034959) with: {'optimizerinput': 'Adagrad'}
0.140934 (0.198065) with: {'optimizerinput': 'Adadelta'}
0.443132 (0.399591) with: {'optimizerinput': 'Adam'}
0.457967 (0.374547) with: {'optimizerinput': 'Adamax'}
0.407692 (0.415712) with: {'optimizerinput': 'Nadam'}
```