4/15/19

This is the final report on your implementation. Paste at the beginning as shown below *as is*, except that for the project design+, respond to each comment—within each of their comments. Again, retain the gray parts. Keep in mind the evaluation criteria (at the end) and that they are somewhat altered for this assignment. For voluminous material, reference appendices (at the end). These will be read on an as-needed basis. Excluding appendices and figures, this response should not exceed 6 pages—without prior permission.

Human Activity Recognition using Smartphone Dataset

by Alp Gulden

# PROJECT PROPOSAL WITH YOUR RESPONSES

Please insert your Phase 1 here, including my comments and your response within each comment.

Alp Gulden

# Project Proposal +

Please use this template; retain the gray text. Your phase 1 materials—in black 12-point Times New Roman—should not exceed 5 pages excluding the gray text, references and figures.

## 1.1 SUMMARY DESCRIPTION

One-paragraph overall description of the inputs and outputs for your proposed semester project. Do not go into details because section 1.3 below does this.

Using smart phones to collect human activities (HA) is much easier than before thanks to precise measurement tools embedded new smart phone such as distance sensor, accelerometer, gyroscope, magnetometer, camera and lighting sensors. This data can be used to measure sport and fitness tracking to health care monitoring. In this project data from Kaggle will be used only from accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz will be captured. Random Forest (many Decision Trees), Neural Network, Keras Neural Network, SGD, Support Vector Classification and K-Nearest Neighbor models will be used to classify human activity. Model will be compared based on result and best model will be identified.

Data will be used from Kaggle “Human Activity Recognition with Smartphones” [1] to recognize six human activities. These activities are Standing, Sitting, Laying, Walking, Climbing up and Climbing down. Models will be compared accuracy and performance of using confusion matrix and random simulation. Data randomly divided for training and testing.

## 1.2 I/O EXAMPLES

At least two concrete examples of projected output for designated input. You will not be held to this—it is just explanatory at this point. The examples should be entirely specific, for example given this input,  the output should be *A cat is in the image*.

Human Activity Recognition (HAR) has a lot off application from Health care to Computer Vision and safety. In this project I am focusing on recognizing Standing, Sitting, Laying, Walking, Walking upstairs and Walking downstairs. Although we are considering many columns provided in the data for clacification I have simplify it for an real life example we can categorize if

* Angle(X, Gravity Mean) as -0,805177, Angle(Y, Gravity Mean) as 0,017430, Angle (Z, Gravity Mean) as -0,097521 system should be recognizing this movement as “Sitting”
* Angle(X, Gravity Mean) as -0.760528, Angle(Y, Gravity Mean) as 0,263183, Angle (Z, Gravity Mean) as 0,0302288 system should be recognizing this movement as “Standing”
* Angle(X, Gravity Mean) as -0,948751, Angle(Y, Gravity Mean) as 0,129933, Angle (Z, Gravity Mean) as 0,034815 system should be recognizing this movement as “Walking”

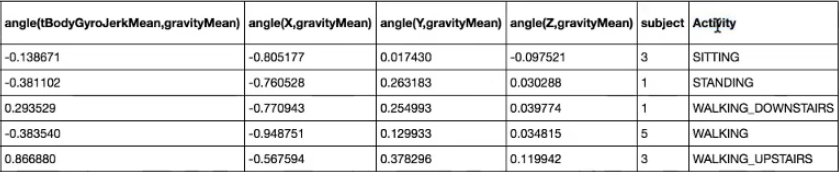


Fig 1. Data Categorization

## 1.3 REQUIREMENTS

High-level requirements statement in 3 roughly equal numbered lists, organized by triage. Separate your requirements into thee approximately even categories using triage (first select the two extreme categories—definite and nice-to-do—and then place the remainder in the middle category):

    1.3.1 Definite (first priority) – 3 to 5 of these

    1.3.2 Not sure yet (second priority) – 3 to 5 of these

    1.3.3 Nice-to-do (can dispense with if time does not allow; third priority) – 3 to 5 of these

Requirements are *declarative* statements of intended functionality such as “the expected number of years to graduate college `shall appear on the console.” (A statement such as “Clean the data” is a procedure step rather than a requirement.)

1.3.1 Definite:

1. Clean up and reduce the dimensions from the dataset given by Kaggle by applying Principal Component Analysis (PCA) technique.
2. Model will be using two main measurement from accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz.
3. Apply Random Forest (many Decision Trees) model
4. Apply Keras Neural Network model.
5. Apply Support Vector Classification model.
6. Apply K-Nearest Neighbor model.
7. Compare model accuracy.
8. Categorize activities accurately for Standing, Sitting, Laying, Walking, Climbing up and Climbing down.
9. Reporting results on a graphic display

1.3.2 Not sure yet:

1. Apply SGD Model
2. Apply TensorFlow Model
3. Apply Grid Search to find the best parameters
4. Apply Confusion matrix to comparison of model accuracy.

1.3.3 Nice-to-do:

1. Use alternative data that collected from smartphones using measurement taking tools and multi-location sensors such as distance sensor, accelerometer, gyroscope, magnetometer.
2. Recognition additional sportive activities to such as Running, riding, exercising (Pull-ups, Push-ups, Squats, Sit-ups and Pull-ups etc.)
3. Come up with smartphone application to collect real time data and classify the activity.

## 1.4 HOW SUCCESS WILL BE ASSESSED

Explain, as specifically as possible (quantification is ideal) how success of the project should be assessed. For example, “the expected number of years to graduate college of 10 BU students, chosen at random, should be correct compared with what actually happened, to within 10%.”

Be able to predict the correct movement will define the success of the application. ∓%90 accuracy range will be considered as very successful model. This will tell us that the neural networks is almost always able to correctly identify the movement type. The success is measured based on the data. The data in Kaggle has already labeled for correct movement. After using the training data, we will be picking the best model to predict the test data and compare the results.

## 1.5 TECHNOLOGY EXPLANATION

Explain what two contrasting machine learning technologies you plan to use--and why you feel they apply. You are permitted to change as your knowledge deepens. It is ideal if both are implemented; second best if one of them is explained for this project but not implemented.

I will be using Decision Tree, Random Forest, Neural Network, Keras Neural Network, SGD, Support Vector Classification and K-Nearest Neighbor, TensorFlow models to classify human activity. Model will be compared based on result and best model will be identified. Confusion matrix will be used for comparison. Coding will be done on python 2.7+, numpy and tesnorflow.

Data is collected from embedded accelerometer, gyroscope and other sensors of Samsung Galaxy S II Smart phone.

## 1.6 BEGINNING DESIGN

Provide the beginning of a design. Phase II will require a complete design but this phase (I) will allow you to show the notation you will use for your design. Try to use at least one figure, showing where inputs go, where outputs come from, and the ML nature of the elements in between. You can use figures from external sources but (a) acknowledge this in the references, and (b) tailor the figures to your particular application.



Fig 2. Generic HAR model [5]

The input data of collected accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity will be used to predict correct human activity in this application. Random Forest (many Decision Trees), Neural Network, Keras Neural Network, SGD, Support Vector Classification and K-Nearest Neighbor models will be used to classify human activity. Model will be compared based on result and best model will be identified. Models will be compared accuracy and performance of using confusion matrix and random simulation. Data randomly divided for training and testing.

## 1.7 DATA SOURCES

Explain whether or not your project requires data. If so, explain were you will obtain it (e.g., Kaggle).

### Kaggle - Human Activity Recognition with Smartphones [2]

The Human Activity Recognition database was built from the recordings of 30 study participants performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors. The objective is to classify activities into one of the six activities performed.

## Description of experiment

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING\_UPSTAIRS, WALKING\_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

## Attribute information

For each record in the dataset the following is provided:

* Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
* Triaxial Angular velocity from the gyroscope.
* A 561-feature vector with time and frequency domain variables.
* Its activity labels.
* An identifier of the subject who carried out the experiment.

## 1.8 SCHEDULE

Rough schedule of design and implementation. (May be adjusted when subsequent assignments are posted)

## 

Fig 3. Project Schedule

## 1.9 REFERENCES FOR PROPOSAL PHASE

Fill in and cite each of the following (e.g., “[2]“) within the text. References can include specific places in the notes and textbook.(Note that this reflects directly on an evaluation criterion.)

[1] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y. Lee, and Tae-Seong Kim,A triaxial accelerometer-based physical -activity recognition via augmented-signal features and a hierarchical recognizer, Information Technology in Biomedicine, IEEE transactions on information technology in biomedicine, Volume 14, NO. 5, September 2010

[2] https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

[3] https://github.com/aiff22/har

[4] https://archive.ics.uci.edu/ml/machine-learning-databases/00240/UCI%20HAR%20Dataset.names

[5] http://122.252.232.85:8080/jspui/bitstream/123456789/17565/1/131260%2C131261.pdf

[6] https://github.com/guillaume-chevalier/Awesome-Deep-Learning-Resources

## 1.7 Evaluation



# PROJECT DESIGN +

Please insert your Phase 2 here, including my comments and your response within each comment.

As usual, please use this Word file template, follow (and retain) the instructions in gray text, and add your work where indicated. The purpose of this template is to assist you in completing a successful project. Keep in mind the evaluation matrix at the end as you do the work and use it to guide what you submit. Use no more than 6 pages of 12-point text excluding figures. You may include as many appendices as you wish for reference. Parts of these may be read as needed.

Your project should involve two technologies, at least one implemented, that approach the same application.

Assignment 2 10/1/19

## 2.1Final Requirements

List your final requirements, numbering them in the form DiX and NiX where:

D/N means “Definite” / “Nice to do” (two categories now, not three)

i = 1, 2, 3, …

X=F when the goal is functional (a conventional requirement) – or – X=L and the goal is a learning goal of yours

For example:

D2F **Wine Suggestion**: The application shall suggest a wine with the dinner. (This is the second definite requirement.)

N4L **TPU Porting**: The application will be ported to Google’s TPU Console to assess the obstacles in doing so, and to measure the difference in execution compared with my GPU-enabled laptop. (This is the fourth nice-to-do requirement.)

There should be 4-8 items in the “Definite” list and at least 4 in the “Nice to do” list. You will reference these numbered requirements in phase 3 when you will be asked to show what the project accomplished.

### Requirements

D1F **Load Data**: Import Pandas Library and load CVS data for test and training.

D2F **Shuffle Data:** To ensure randomness we need to have a random order data.

D3F **Sperate Data Input and Output Labels:** We need to remove the “Activity” column from the data and create sperate dataset.

D4F **Encoding Labels:** Refers to converting the labels into numeric form so as to convert it into the machine-readable form.

D5F **Principal Component Analysis (PCA) technique**: Clean up and reduce the dimensions from the dataset given by Kaggle. Accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity values will only be used.

D6F **Random Forest**: Decision Tree model implementation.

D7F **Keras Neural Network:** Model Implementation.

D8F **SVM Model**: Support Vector Machine Classification model implementation.

D9F **K-Nearest Neighbor:** Model Implementation.

D10F **Confusion Matrix:** Compare model accuracy**.**

D11F **Visual Report1**: Graphical Display for each models accuracy

D12F **Visual Report2**: Graphical display for model specific results on categorized activities accurately for Standing, Sitting, Laying, Walking, Climbing up and Climbing down.

N1L **SGD Model:** Model Implementation.

N2L **TensorFlow:** Model Implementation.

N3L **Grid Search**: Apply to find the best parameters

N4L **Alternative measurements for sports:** Use alternative data that collected from smartphones using measurement taking tools and multi-location sensors such as distance sensor, accelerometer, gyroscope, magnetometer. This can be used to recognition additional sportive activities to such as Running, riding, exercising (Pull-ups, Push-ups, Squats, Sit-ups and Pull-ups etc.)

N5L **Application:** Come up with smartphone application to collect real time data and classify the activity.

## 2.2 Design and Theory

Describe the design of your proposed system. Use annotated diagrams. Explain the theory behind your design. This is particularly true of technology not specifically covered in the notes. Explain how the two technologies will interface or compare. The reader should understand how the pieces are going to fit together. Show this at a high level, as well as providing as much relevant detail as you can. Include at least one (meaningful) figure, as in the figure, for example (you may edit it for your purposes if you like).

### Human Activity Recognition

The idea behind capturing Human Activity is became much more common after cell phone usage. Companies like Google, Facebook, Amazon etc. is not only collecting data of where we are and what we do, they also collect what activity we are performing during the day. That will allow them to understand our activity level, sleep level, health level and many more information can capture how we work to how we live. This will allow them to make predictions. I have found a lot of research on the internet is being done in this field of Human activity recognition. It is important however how collect the data. Cell phone worn by the person generates the information which is used to interpret the activity. Good precision can be obtained from sensors which are worn in waist, wrist, chest and thighs. Technology is much advanced and phone sensors are much more accurate. [9]

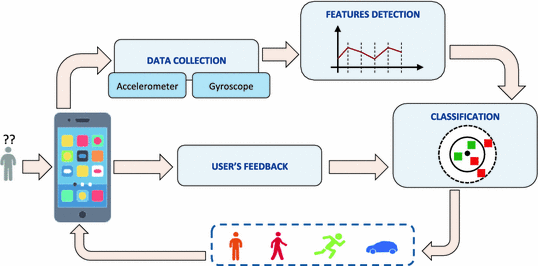


Fig. 4: System overview. [10]

On figure 4. shows smartphones is using capturing the activities performed by the users. Accelerometer and gyroscope are used to collected data. Machine Learning models are used to detect some relevant features which used for classification.

### Main Objective

* Clean up the given dataset so we can have only the necessary information.
* Reduce the dimension by using Principal Component Analysis (PCA).
* Using different Machine Learning models to identify best option.
* Using Confusion Martix pick the best model.
* Calculate error rate, accuracy and performance.



Fig. 5: Generic HAR model [5]

### Design Highlight

Imported, cleaned, normalized and reduced data will be processed various Machine Learning algorithms like Random Forest, Support Vector Machine, Artificial Neural Network and K-Nearest Neighbour to classify data into six categories namely Sitting, Standing, Laying, Walking, Climbing up and Climbing down. Correctness of the system is determined by generating confusion matrix and by random simulations.

### Collecting Data

#### Accelerometer:

An accelerometer is a device which is used for measuring static or dynamic acceleration forces. We can find out the angle by which the device is tilted by determining the static acceleration due to gravity. And we can sense the direction of movement by measuring the dynamic acceleration. Accelerometer may work on the piezoelectric effect or by sensing the changes in capacitance. Other less used methods use piezoresitive effect, hot air bubbles, and light. Piezoelectric accelerometers have microscopic crystals which generates voltage when they are under acceleration. [12]

Accelerometer in smartphone contains seismic mass circuit which is made of silicon. The mass changes its orientation according to the orientation of the device.

#### Gyroscope:

Gyroscopes are small and cheap devices which are used for the measurement of rotational motion that is angular velocity. Degrees per second (°/s) or revolutions per second (RPS) are the units of angular velocity.



Figure 6: Axis of angular rotation for gyroscopes

A triple axis MEMS gyroscope can measure rotation in 3 axes: x, y, and z. gyroscopes with single or dual axis are becoming less popular as the triple axis gyroscopes have become cheaper and small. Very low-current electrical signals are produced, which are later amplified, when the resonating mass in gyroscope changes its position due to rotational motion. MEMS gyroscopes operate in mA or sometimes micro-ampere range. [13]

### Principal Component Analysis (PCA)

PCA is used when the data has many columns. Whit many unnecessary attributes our models might have high error rates. PCA will be using to reduce the dimensions of a large dataset.

Principal component analysis works by converting the variables in the given dataset to a new set of variables, the principal components (PCs). [14]

#### Random Simulation

Random Simulation is used to test accuracy of predictive models and prevent overfitting and underfitting of data. The technique involves randomly dividing the dataset into training and testing set in the ratio 7:3. The whole simulation is repeated 50 times to improve accuracy of model as per the statistical Central limit theorem. Testing dataset provides us with approximation of real time data and provides us with a mechanism to test stability of our model in real life scenario.

### Machine learning algorithms

Different supervised, semi-supervised and unsupervised algorithms can be used to solve the problem of real-time recognition. Different algorithm had proved themselves useful in different applications. To be able to decided which model works better in which condition we need to apply the model and compare the result, performance and accuracy.

#### Random Forests

Random Forest are an ensemble learning method for classification, regression and other tasks. It works by constructing many decision trees at training time and outputs the class that is the mean prediction of decision trees. Random forest prevents over fitting of data. Combination of bagging method and randomization improves the performance of algorithm. The amount of decision trees constructed will be decided during the implementation. [16]



Figure [7] Random Forest Explained [15]

#### K-Nearest Neighbour

KNN is an instance-based classifier. It operates on the principal that classification of unknown instances can be done by relating the unknown instance to known instance on basis of some function. This function is similarity or distance function.

KNN is an instance-based classifier. It operates on the principal that classification of unknown instances can be done by relating the unknown instance to known instance on basis of some function. This function is similarity or distance function.

We have used Euclidean distance function to approximate our learning function. We had determined value of K by plotting graph of error rate vs K value as shown in figure below. [17]



Figure 7: Variation of success rate in K Nearest Neighbour (sample graph)

#### Support Vector Machine (SVM)

Support Vector Machines are based on decision hyperplanes that define decision boundaries. A decision plane separates two set of objects having different class membership. Support Vector Machine aims to maximize decision boundary between hyperplanes. It’s been found that frequency domain features provided better results than the time domain features with the use of SVM to classify 8 activities.

SVM algorithm takes kernel type, cost and gamma as parameters. I will be using chosen Gaussian(radical) kernel as similarity function. [18]

#### Artificial Neural Networks

An artificial neural network mimics the working of neurons in a biological brain. It derives the relationship between the input signals and output signals. The most common method to train data is Back-propagation method. In this method error at the output is determined and then it is propagated back into the network.

Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding. We will be using nnet package to train our dataset, which is used for feed-forward neural networks with a single hidden layer. The nnet package trains the artificial neural network using backpropagation method. In this method error at the output is determined and then it is propagated back into the network. To minimize the error resulting from each neuron, the weights are updated. [19].

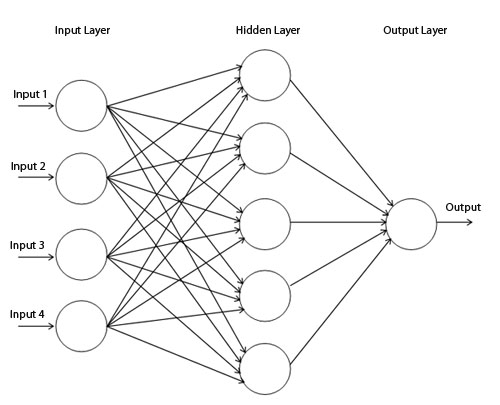
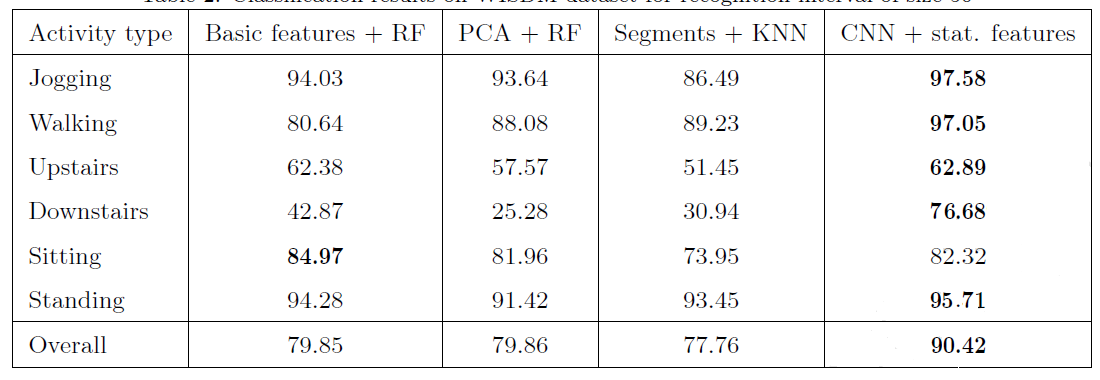


Fig 8. Single hidden layer Neural Network

### Performance Analysis

In this section we have compared performance of different machine learning models based on confusion matrix and time taken to train the model. Time taken to train the model is calculated by taking mean of 30 simulations.





Example of results. In our project similar results will be generated.

### Conclusion

Future Aspects: The model developed can be used to predict Human Activity on real time basis. An application on Android Platform can be used to convey measurements and run the model on those measurements. This application has aspects in monitoring health and performance of athletes etc.

 Innovative Idea: We have found that processing data through PCA (Principal Component Analysis) resulted in optimized model creation which led to improvement in performance.

In this paper we proposed a solution for user-independent human activity recognition problem

that is based on Convolutional Neural Networks augmented with statistical features that embrace

295 global properties of the accelerometer time series. It has the benefits of using short recognition

intervals of size up to 1 second and requiring almost no feature engineering and data preprocessing.

Due to a relatively shallow architecture, the proposed algorithm has a small running time and can

be efficiently executed on mobile devices in real time.

To evaluate the performance of the considered approach we tested it on two popular WISDM

300 and UCI HAR datasets. The obtained results demonstrate that the proposed CNN-based model

significantly outperforms baseline approaches and establishes state-of-the-art results in both cases.

The cross-dataset experiment has further emphasized a platform-independent architecture that can

be applied not only to different users, but to devices with different accelerometer calibrations.

## 2.3 Tools

Describe the tool(s) you will use, or explain why you will build from scratch. Explain your choice. Show that you understand how the tools work.

### Technology and Tools

Phyton Language will be used in PyCharm and JupyterNotebook coding environment.

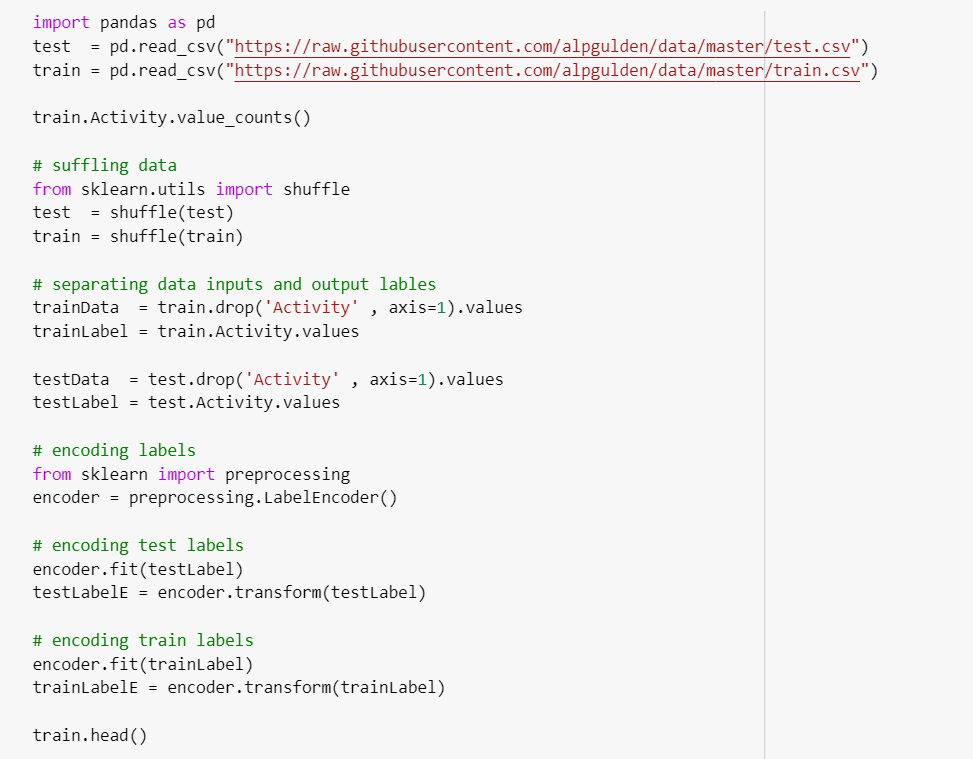
Pandas, sklearn.neural\_network, numpy, pyplot, seaborn, sklearn.decomposition, PCA, Keras, RandomForestClassifier, StandardScaler, KNeighborsClassifier, SGD, scikit-learn, sklearn, scatter\_matrix, DecisionTreeClassifier, accuracy\_score, confusion\_matrix, GaussianNB, QuadraticDiscriminantAnalysis, GridSearchCV, TensorFlow, OpenCV, Plotly

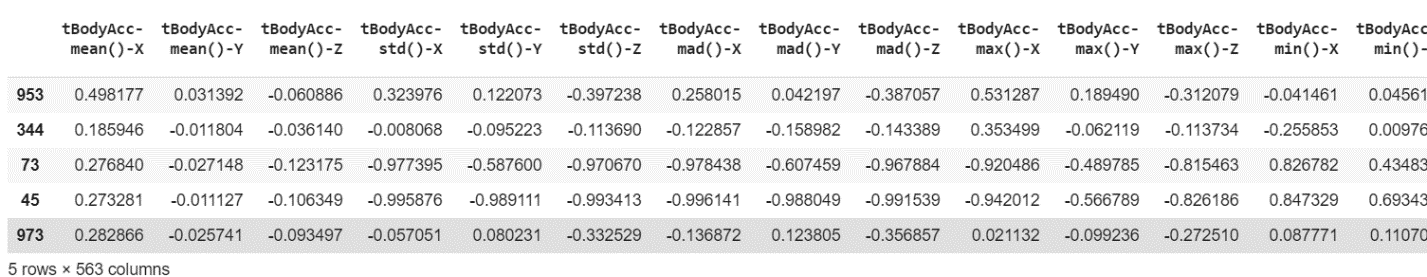
libraries and tools will be used for implementation.

## 2.4 Implementation Fragments

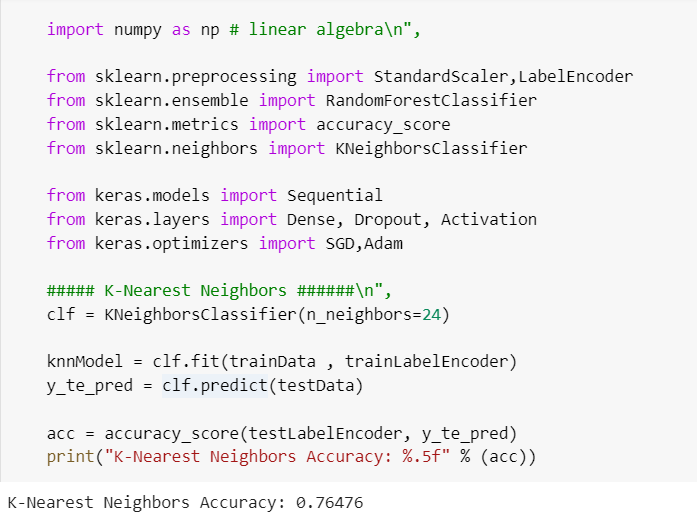
Show enough *parts* of an implementation—or a simplified form of it—to convince the reader (and yourself) that you will have the implementation of the definite requirements completed on time. These can be experimental or exploratory in nature but the purpose is to have something up and running at all times. Your choices can coordinate with section 2.4 below. Cut and paste at most 1 1/2 pages of commented code below. Explain what part of the application they each relate to.

### Data Loading





### KNN-K Nearest Neighbor



## 2.5 Risk Retirement

Identify and prioritize the 5 top risks in carrying out the project. Try as best you can to retire the top three by the time you submit this, by means of experiments, prototypes, or work-arounds. Explain how you did this. Explain how you will retire the remaining risks in advance. (P.S. Consult with Eric or the course TA if and when you need assistance!)

1. **Data Availability:** My original project was working based on HER data. This is highly regulated and protected patient information is not available on the internet. I didn’t want to take the risk to use the data from my workplace.

**How to Resolve:** I have end up changing the project all together while I can and have time. Human Activity Recognition is fun project and available on the internet so I can actually use some previous research as base.

1. **Lack of Time:** I am at the end of my project at work and production deployment phase is pretty hectic.

**How to Resolve:** I knew that would be an issue before taking the class. I am staying late and working hard to keep up with the project. Time management is the key.

1. **Being Original or a Copy:** As there are some examples of HAR system on the internet one of my concern is to copy a project rather and being an original.

**How to Resolve:** As I want to be an ambitious and create something new it was obvious; I could not make it on given time. So, I decided to start with base model available on the internet. I will be incorporating some new features and change the parameters to get better results once I have something to present.

1. **Incorporating multiple data source and features from different HAR projects :** I would like us more measurement and recognize more activities. But this is a risk of not completing project.

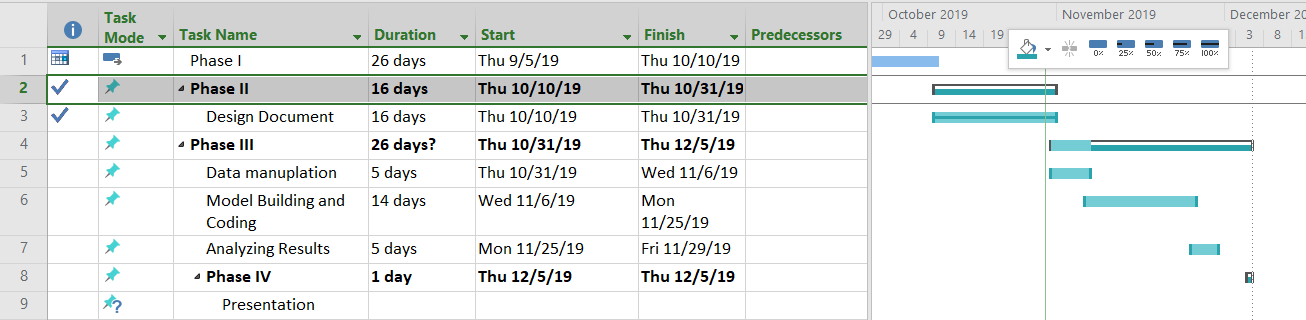
**How to Resolve:** I have put them as nice to have list. So I will pursue if I have time.

1. **Environment and Issues with Coding Tools:** I have struggled running the TensorFlow and Visual Studio on my computer.

**How to Resolve:** I need to research and fix the issues on programming tools and libraries. In worst case scenario I will be running my code on colab.research.google.com

## 2.6 Schedule

Explain in outline the steps you intend to take to carry out the project. Show the completion of the stages. Include a schedule, as detailed as can be reasonably foreseen.



## 2.7 References

## List the references to materials of all kinds that you used in developing this. Each should occur in the body of the text at the appropriate place. Example: Jones [5] recommends running the genetic algorithm for at least 1o generations.

### List of Abbreviations

PCA- Principal Component Analysis

HAR- Human Activity Recognition

NA- Not Applicable

CSV- Comma separated values

SVM-Support Vector Machine

KNN-K Nearest Neighbor

### References

[1] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y. Lee, and Tae-Seong Kim,A triaxial accelerometer-based physical -activity recognition via augmented-signal features and a hierarchical recognizer, Information Technology in Biomedicine, IEEE transactions on information technology in biomedicine, Volume 14, NO. 5, September 2010

[2] https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones

[3] https://github.com/aiff22/har

[4] https://archive.ics.uci.edu/ml/machine-learning-databases/00240/UCI%20HAR%20Dataset.names

[5] https://www.youtube.com/watch?feature=player\_embedded&v=XOEN9W05\_4A

[6] https://github.com/guillaume-chevalier/Awesome-Deep-Learning-Resources

[7] “Human activity recognition using smartphone dataset”, accessed on 1/11/2016, htts://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+ Smartphones

[8] Lester, Jonathan and Choudhury, Tanzeem and Borriello, Gaetano ,”A practical approach to recognizing physical activities”, accessed on 4/10/2016, https://www.cs.cornell.edu/~tanzeem/pubs/JonathanLester\_EDAS-1568973904.pdf

[9]. Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz, “A Public Domain Dataset for Human Activity Recognition Using Smartphones”, European Symposium on Artificial Neural Networks, 24-26 April 2013.

[10] https://link.springer.com/chapter/10.1007/978-3-319-70169-1\_5

[11] Jorge L. Reyes-Ortiz, Alessandro Ghio, Davide Anguita , Xavier Parra, Joan Cabestany, Andreu Catal, “Human Activity and Motion Disorder Recognition: Towards Smarter Interactive Cognitive Environments”, European Symposium on Artificial Neural Networks, 24-26 April 2013.

[12] TONI\_K, “Accelerometer Basics”, accessed on 20/9/2016, https://learn.sparkfun.com/tutorials/accelerometer-basics

[13] Toni\_K, “Gyroscope Basics” ,accessed on 20/9/2016, https://learn.sparkfun.com/tutorials/gyroscope

[14] “Practical Guide to Principal Component Analysis (PCA) in R & Python”, accessed on 5/11/2016,

https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/

[15] https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

[16]. Tavish Srivastava, “Introduction to Random forest – Simplified”, accessed on 8/11/2016,

https://www.analyticsvidhya.com/blog/2014/06/introduction-random-forest-simplified/

[17] “k-nearest neighbors algorithm”,accessed on 12/2/2017, https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm

[18] https://en.wikipedia.org/wiki/Support-vector\_machine

[19] Chapman & Hall/CRC Machine Learning & Pattern Recognition Series MACHINE LEARNING

[20] https://www.nicolamanzini.com/single-hidden-layer-neural-network/

## 2.8 Evaluation of Design Phase

Keep in mind that good and excellent responses require more than a minimum.



Keep in mind that good and excellent responses require more than a minimum.

# IMPLEMENTATION

## 3.1 Summary

In a paragraph or two, summarize the outcome of your project functionally and learning-wise but avoid duplication with Section 3.2 below. (Reminder: leave these gray sections in your paper.)

In this project I have managed to implement Random Forest, Neural Network, Support Vector Classification and K-Nearest Neighbor and Tensor flow models using the given data set. I have learned implementing Logistic Regression, Scaling and Principal Component Analysis (PCA) to find the best option to scale, encode, and reduce the database features.

If we look at the models Random Forest has the most accuracy on my research with a score of %92 acuracy and second best was the Keras Neural Network with %90 accuracy. But if we look at the training time K-Nearest Neighbour was the fastest by 2.7 seconds.

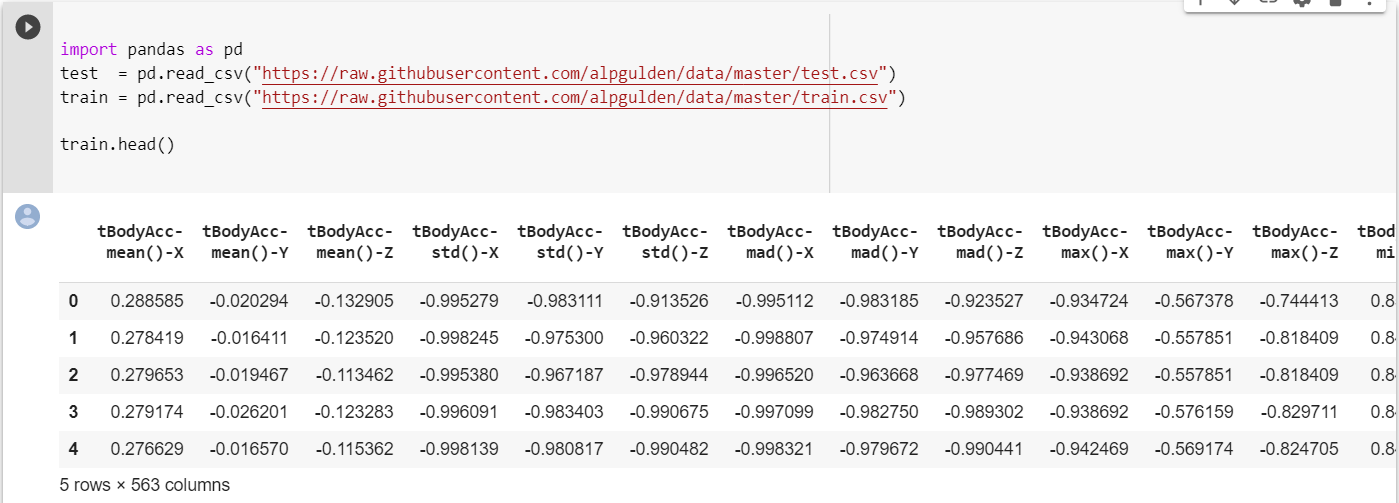
## 3.2 Report on Requirements

Describe the extent to which you accomplished each definite requirement "DiX" (X = F or L) as well as any other fulfilled requirements. For each, include 1-4 sentences and screenshot of relevant input and output. Your effectiveness depends a lot on how much you demonstrate that you learned. Limit: 2 pages of 12-point Times New Roman.

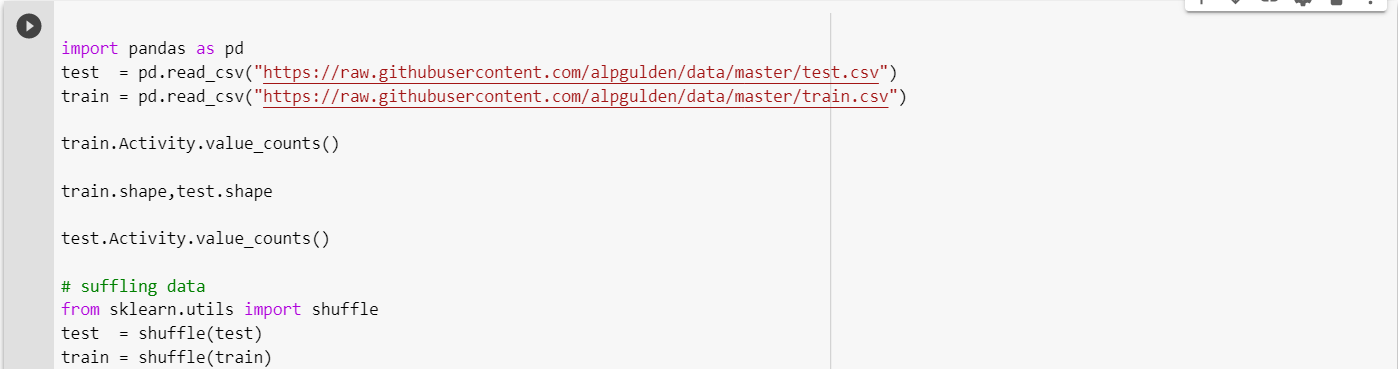
**Data Calcification Requirements**

To understand the data better we need to run some tests and models such as Logistic Regression, PCA or Scala. Only after seeing the result we can decide which data manipulation method to use to get the best performance. Because the data comes with 563 columns and not all that meaningful for us.

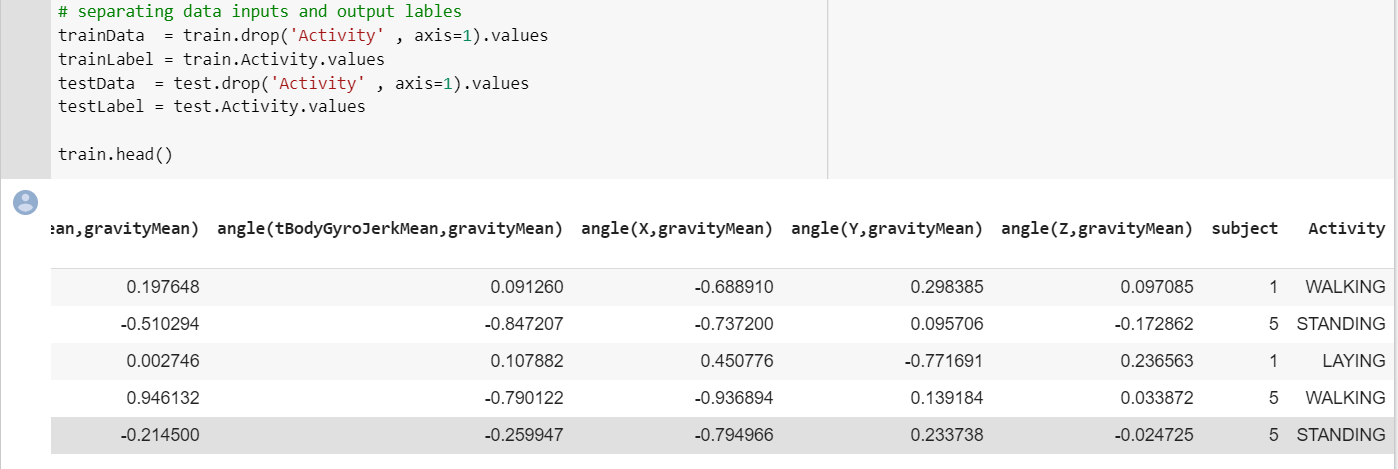
Y1X **Load Data**: Import Pandas Library and load CVS data for test and training.



Y2X **Shuffle Data:** To ensure randomness we need to have a random order data.

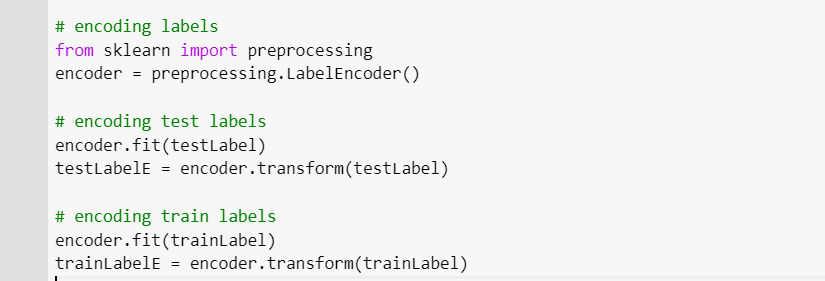


Y3X **Sperate Data Input and Output Labels:** We need to remove the “Activity” column from the data and create sperate dataset.



On testData Activity column dropped for proper training data.

Y4X **Encoding Labels:** Refers to converting the labels into numeric form so as to convert it into the machine-readable form.



We are using sklearn.preprocessing.LabelEncoder [21] functions for labelling. This Encode labels with value between 0 and n\_classes-1. Transform function called to Transform labels to normalized encoding.

Y5X **Identifying witch technique is the best way to manipulate the data for applying models:**

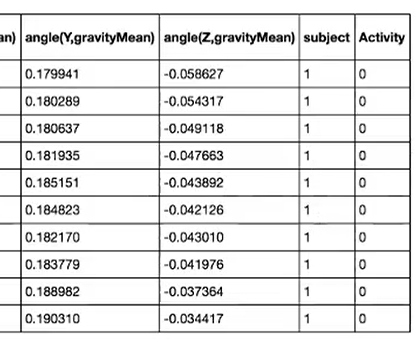
Logistic Regression, Principal Component Analysis (PCA) and Scalar techniques will be test for best performance.

Similar to encoding we are using different technique for the same problem in this section. After finding unique activities we are replacing them with numbers like we did in encoding.

*#identifying the unique activities*unique\_activities = train\_df.Activity.unique()  
replacer = {}  
  
*# assign each activity a number. this is same as encoding. we are using a different technique for the same problem.***for** i, activity **in** enumerate(unique\_activities):  
 replacer[activity] = i  
 train\_df.Activity = train\_df.Activity.replace(replacer)  
 test\_df.Activity = test\_df.Activity.replace(replacer)  
 train\_df.head(10)

Then we drop the subject column which has no importance for the calculation.

train\_df = train\_df.drop(**"subject"**, axis=1)  
test\_df = test\_df.drop(**"subject"**, axis=1)



To load the data we will using a function called LoadAllData

def LoadAllData():

  train\_values = train\_LR.values

  test\_values = test\_LR.values

  np.random.shuffle(train\_values)

  np.random.shuffle(test\_values)

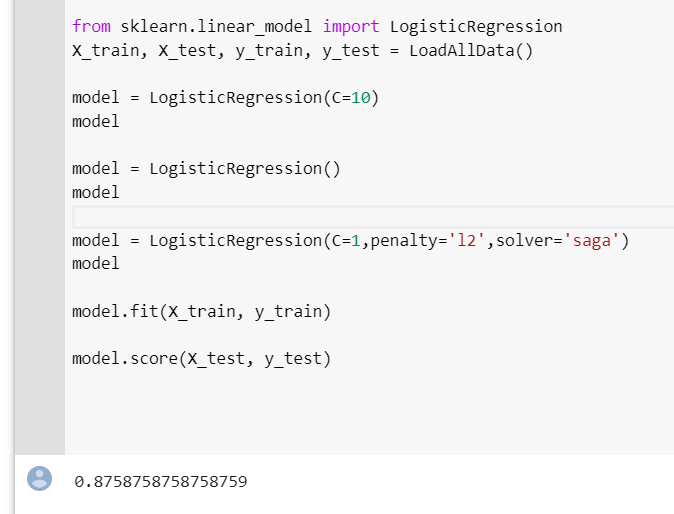
  X\_train = train\_values[:, :-1]

  X\_test = test\_values[:, :-1]

  y\_train = train\_values[:, -1]

  y\_test = test\_values[:, -1]

  return X\_train, X\_test, y\_train, y\_test

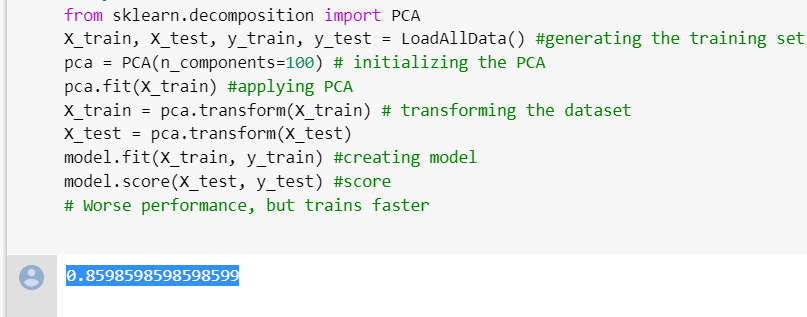


Using the LogisticRegression function from skLearn [22] I have tried different values on penalty, float and solver parameters and find the best accuracy score on C=1, penalty=’l2’ and solver=’saga’ which is %88

To understand if there the variables high correlated between the variables or if they are independent

we will apply Principal Component Analysis (PCA). PCA is one method where we can tell if the predictors linearly or highly corelated to perform predictions so we can reduce the dimension.

We will be using sklearn.decomposition.PCA function [23].

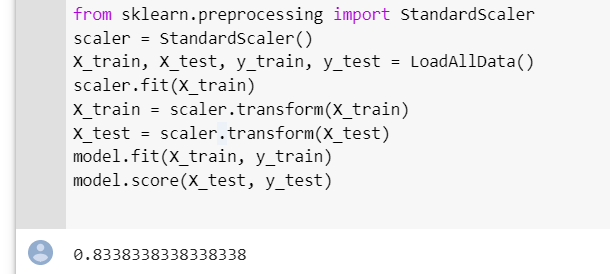
****

if we try to reduce the highly corelated columns to 100. Original data came with 563 columns. However, we can see the score reduce to

**0.8598598598598599**

So it is not better.

Another way of transforming the data would be using the Scaler function [24]. This will standardize features by removing the mean and scaling to unit variance.

****

While by PCA we are combining variables and reducing the number of predictor by Scale we can assign variables to binary numbers from -1 to +1 that might boost the performance of the model. We need to look at the data structure and find out which transformer would give us advantage. In this case PCA and

**Requirement is Applying Models to find best model predicts activity in highest percentage.**

Y6X **Random Forest Classifier**: Decision Tree model implementation.

RandomForestClassifier I applied on scaled dataset with estimator value 100. We got the score of **%92** percent.



Y7X **Keras Neural Network:** Model Implementation

I have used both scaled data and logisticregression based data to compare. While Logistic Regression gets 0.897897889962544 point, scale data performed better with Keras NN model scoring

**0.9059059013714185**.

from keras.models import Sequential

from keras.layers import Dense

from keras.wrappers.scikit\_learn import KerasClassifier

from keras.utils.np\_utils import to\_categorical

X\_train, X\_test, y\_train, y\_test = LoadAllData()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

n\_input = X\_train.shape[1]

# number of features

n\_output = 6

# number of possible labels

n\_samples = X\_train.shape[0]

# number of training samples

n\_hidden\_units = 100

Y\_train = to\_categorical(y\_train)

Y\_test = to\_categorical(y\_test)

print(Y\_train.shape)

print(Y\_test.shape)

def create\_model():

  model = Sequential()

  model.add(Dense(n\_hidden\_units,

  input\_dim=n\_input,

  activation="relu"))

  model.add(Dense(n\_hidden\_units,

  input\_dim=n\_input,

  activation="relu"))

  model.add(Dense(n\_output, activation="softmax"))

  # Compile model

  model.compile(loss="categorical\_crossentropy", optimizer="adam", metrics=['accuracy'])

  return model

estimator = KerasClassifier(build\_fn=create\_model, epochs=20, batch\_size=10, verbose=False)

estimator.fit(X\_train, Y\_train)

print("Score: {}".format(estimator.score(X\_test, Y\_test)))

After many trials changing the n\_hidden\_units, epochs, activation method and optimizer

I have found score goes up to %90 percent.

Y8X **SVM Model**: Support Vector Machine Classification model implementation.

**import** matplotlib.pyplot **as** plt  
%matplotlib inline  
**from** sklearn.svm **import** SVC  
**from** sklearn.model\_selection **import** StratifiedKFold  
**from** sklearn.feature\_selection **import** RFECV #Random Forest using Cross Validation  
**from** sklearn.utils **import** shuffle

Stratified K-Folds cross-validator function [25] provides train/test indices to split data in train/test sets.

Model will randomly will pick up sample in after fold. In Strata it uses groups so for each activity we system will sampling for each activity group. When we do cross validation, we divided data into blocks. It allows us to try the model on each block to prevent overfitting. [26]



Cross Validation [26]

*svc = SVC(kernel=****"linear"****)*

Kernel specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable. If none is given, ‘rbf’ will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n\_samples, n\_samples). [25]

*rfecv = RFECV(estimator=svc, step=1, cv=StratifiedKFold(6),*

*scoring='accuracy')*

RFECV is used for cross validation for 6 folds in this case.

*# train and fit data in the model*

*rfecv.fit(trainData, trainLabel),*

*# Plot number of features VS. cross-validation scores\n",*

*plt.figure(figsize=(32,12))*

*plt.xlabel("Number of features selected")*

*plt.ylabel("Cross validation score (nb of correct classifications)")*

*plt.plot(range(1, len(rfecv.grid\_scores\_) + 1), rfecv.grid\_scores\_)*

*plt.show()*

Number of features in Train : 563

Number of records in Train : 999

Number of features in Test : 563

Number of records in Test : 999

Train Data shape : (999, 561)

Train Label shape : (999,)

Test Data shape : (999, 561)

Test Label shape : (999,)

Label examples:

['LAYING' 'SITTING' 'STANDING' 'WALKING' 'WALKING\_DOWNSTAIRS'

'WALKING\_UPSTAIRS']

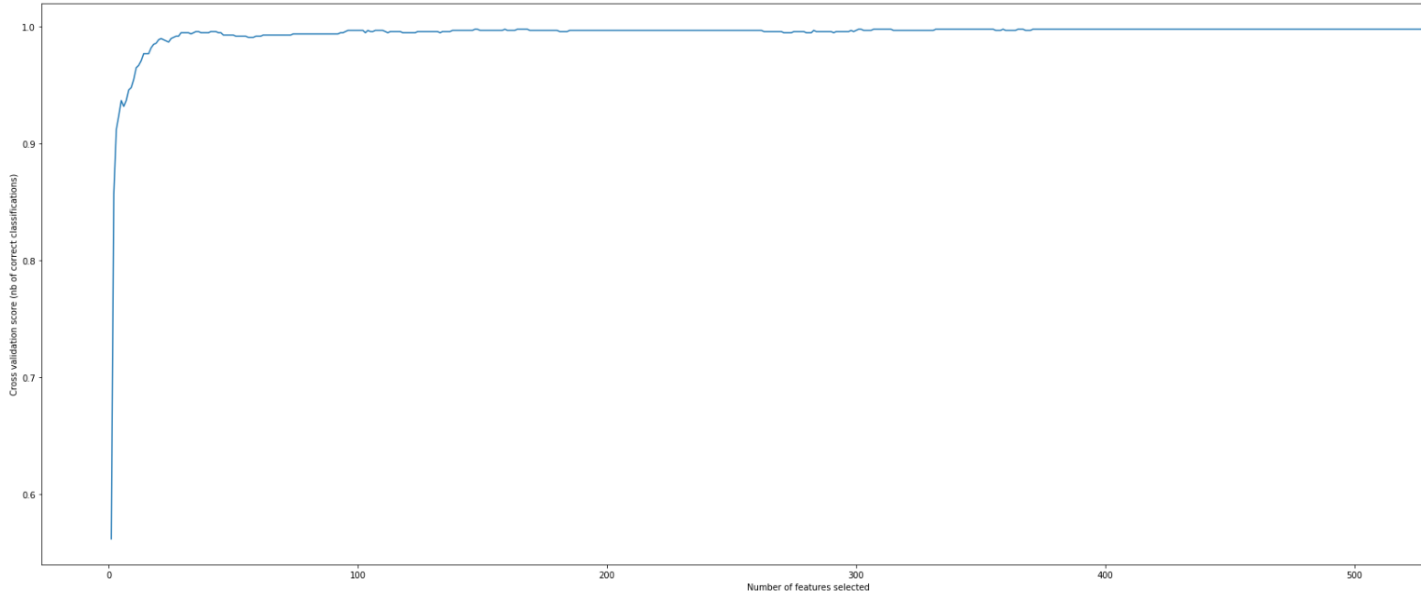
Labels before Shuffle [2 2 2 2 2]

Labels after Shuffle [1 5 5 0 3]

Optimal number of features : **159**

Processing time sec 600.1908029999998

**Accuracy of the SVM model on test data is 0.8748748748748749**



We tried to get the optimal feature count out of 563 and as seen form the plot 159 is the optimal number. Now we need to select best 159 features that we can use.

Accuracy with given parameters is **0.87** for SVM.

Y9X **K-Nearest Neighbor:** Model Implementation.

***from*** *sklearn.neighbors* ***import*** *KNeighborsClassifier*

class **sklearn.neighbors.KNeighborsClassifier**(n\_neighbors=5, weights='uniform', algorithm='auto', leaf\_size=30, p=2, metric='minkowski', metric\_params=None, n\_jobs=None, \*\*kwargs)

*##### K-Nearest Neighbors ######*clf = KNeighborsClassifier(n\_neighbors=24)  
  
knnModel = clf.fit(trainData , trainLabelEncoder)  
y\_te\_pred = clf.predict(testData)  
  
acc = accuracy\_score(testLabelEncoder, y\_te\_pred)  
**print**(**"K-Nearest Neighbors Accuracy: %.5f"** % (acc))

By using the ‘predict’ function this model tries to optimize by looking at nearest values of classifies.

#applying k-nearest neighbours

from sklearn.neighbors import KNeighborsClassifier as knn

import numpy as np

knnScoreDistance=np.zeros(51)

knnScoreUniform=np.zeros(51)

for num in range(5,51):

   knnclf = knn(n\_neighbors=num, weights='distance')

   knnModel = knnclf.fit(trainData , trainLabelsE)

   knnScoreDistance[num]=knnModel.score(testData  , testLabelsE )

   print("Testing  set score for KNN\_Distance(k=%d): %f" %(num,knnScoreDistance[num]))

for num in range(5,51):

   knnclf = knn(n\_neighbors=num, weights='uniform')

   knnModel = knnclf.fit(trainData , trainLabelsE)

   knnScoreUniform[num]=knnModel.score(testData  , testLabelsE )

   print("Testing  set score for KNN\_Uniform(k=%d): %f" %(num,knnScoreUniform[num]))

import matplotlib.pyplot as plt

x=np.array(range(5,51))

plt.plot(x,knnScoreDistance[5:])

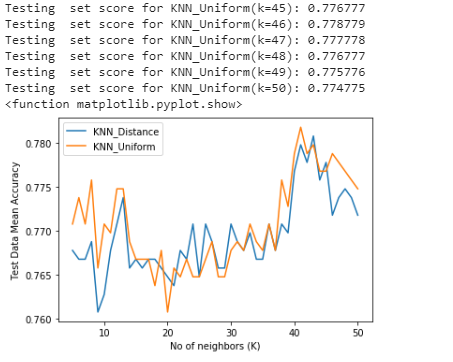
plt.plot(x,knnScoreUniform[5:])

plt.xlabel("No of neighbors (K)")

plt.ylabel("Test Data Mean Accuracy")

plt.legend(['KNN\_Distance','KNN\_Uniform'])

plt.show



After running the K-NN for both distance and uniform the scores came up to **0.77%.**

Y9X **Neural Network:** Model Implementation.

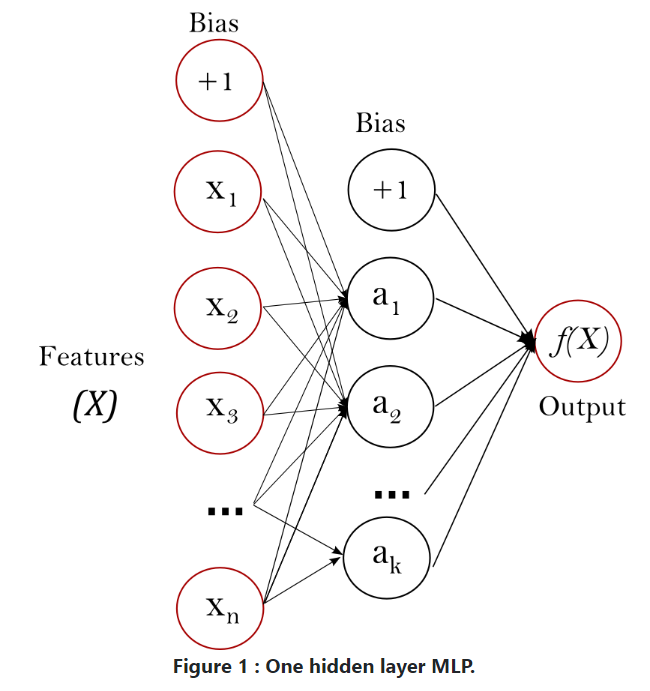
Applying NN with multi-layer perceptron classifier (MLPClassifier) [29]

MLP is different from Logical Regression. It is a supervised learning algorithm that learns a function  by training on a dataset, where is the number of dimensions for input and is the number of dimensions for output. Given a set of features 

and a target , it can learn a non-linear function approximator for either classification or regression. [29]

***import*** *sklearn.neural\_network* ***as*** *nn  
mlpADAM = nn.MLPClassifier(hidden\_layer\_sizes=(10,)  
 , max\_iter=1000 , alpha=1e-4  
 , solver=****'adam'*** *, verbose=10  
 , tol=1e-19 , random\_state=1  
 , learning\_rate\_init=.001)*

*nnModelADAM = mlpADAM.fit(trainData , trainLabelE)*



After 1000 iteration with 10 hidden layers, default activation= ‘relu’ and solver= ‘adam’ we can see the loss drops to 0.00009863 and score is **0.857858**

Iteration 998, loss = 0.00009905

Iteration 999, loss = 0.00009911

Iteration 1000, loss = 0.00009863

Training set score: 0.857858

/usr/local/lib/python3.6/dist-packages/sklearn/neural\_network/multilayer\_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and the optimization hasn't converged yet.

% self.max\_iter, ConvergenceWarning)

MLPClassifier(activation='relu', alpha=0.0001, batch\_size='auto', beta\_1=0.9,

beta\_2=0.999, early\_stopping=False, epsilon=1e-08,

hidden\_layer\_sizes=(10,), learning\_rate='constant',

learning\_rate\_init=0.001, max\_iter=1000, momentum=0.9,

n\_iter\_no\_change=10, nesterovs\_momentum=True, power\_t=0.5,

random\_state=1, shuffle=True, solver='adam', tol=1e-19,

validation\_fraction=0.1, verbose=10, warm\_start=False)

for 1000 hidden layers and 100 iteration we got

Training set score: **0.877878**

With 1000 hidden layers and 1000 iteration the score ends up

Training set score: **0.879880**

Finally with solver= ‘sgd’ and 90 hidden layer with 1000 iteration

Training set score: **0.880881**

Y10X **Confusion Matrix:** Compare model accuracy**.**

We are using DecisionTreeClassifier and

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score,confusion\_matrix

sklearn.metrics.confusion\_matrix(y\_true, y\_pred, labels=None, sample\_weight=None, normalize=None) function to Compute confusion matrix to evaluate the accuracy of a classification.

decsnTreeClf= DecisionTreeClassifier(criterion='entropy')

tree=decsnTreeClf.fit(trainData,trainLabelsE)

testPred=tree.predict(testData)

acc= accuracy\_score(testLabelsE,testPred)

cfs = confusion\_matrix(testLabelsE, testPred)

print("Accuracy: %f" %acc)

plot function used to get the plot below

def plot\_confusion\_matrix(cm, classes,

                         normalize=False,

                         title='Confusion matrix',

                         cmap=plt.cm.Blues):

#This function prints and plots the confusion matrix.

#Normalization can be applied by setting `normalize=True`.

   plt.imshow(cm, interpolation='nearest', cmap=cmap)

   plt.title(title)

   plt.colorbar()

   tick\_marks = np.arange(len(classes))

   plt.xticks(tick\_marks, classes, rotation=45)

   plt.yticks(tick\_marks, classes)

   if normalize:

        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

        print("Normalized confusion matrix")

   else:

     print('Confusion matrix, without normalization')

   print(cm)

   thresh = cm.max()/2

   for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

       plt.text(j, i, cm[i, j],

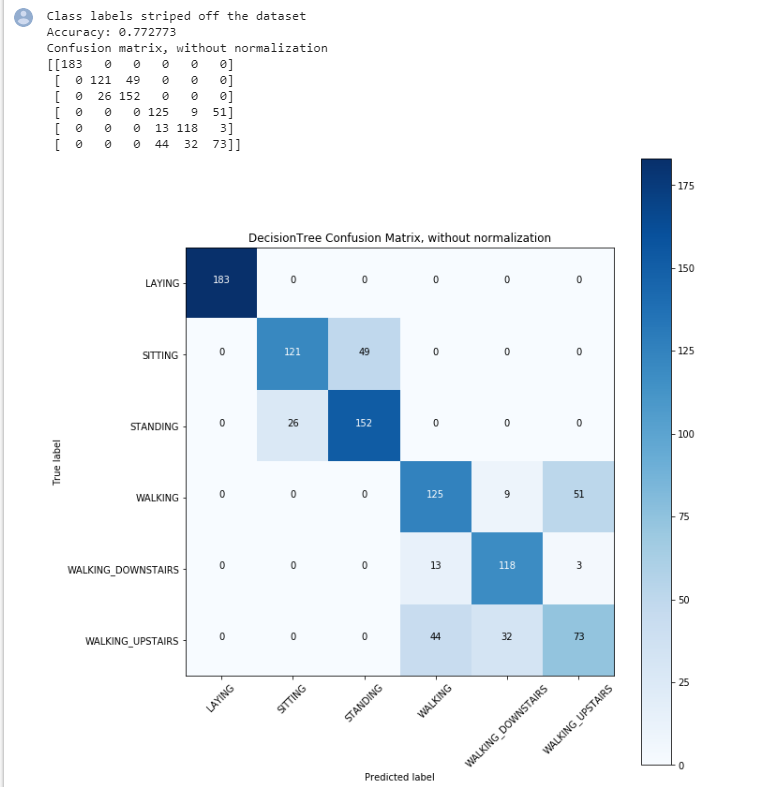
              horizontalalignment="center",

              color="white" if cm[i, j] > thresh else "black")

   plt.tight\_layout()

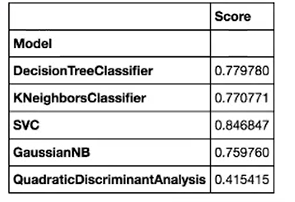
   plt.ylabel('True label')

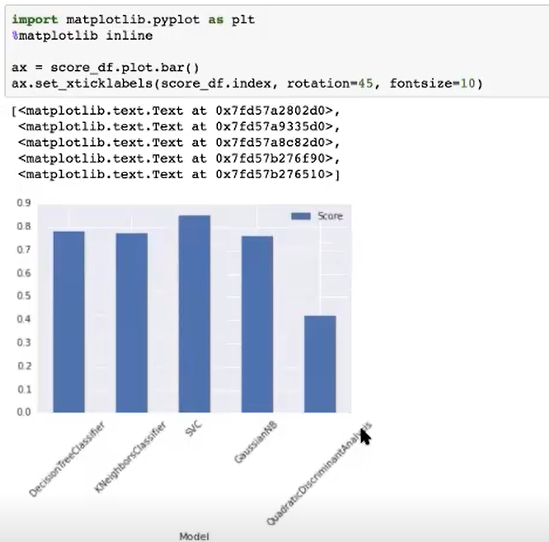
   plt.xlabel('Predicted label')



D11F **Visual Report1**: Graphical Display for each models accuracy

When compare DecisionTreeClassifier, KNeighborsClassifier, SVC(), GaussianNB() and QuadraticDiscriminantAnalysin on same plot





D12F **Visual Report2**: Graphical display for model specific results on categorized activities accurately for Standing, Sitting, Laying, Walking, Climbing up and Climbing down.

After analyzing the data we can see how data has been categorized based on regression models

fig = plt.figure(figsize=(32,24))

ax1 = fig.add\_subplot(221)

ax1 = sb.stripplot(x='Activity', y=sub15.iloc[:,0], data=sub15, jitter=True)

ax2 = fig.add\_subplot(222)

ax2 = sb.stripplot(x='Activity', y=sub15.iloc[:,1], data=sub15, jitter=True)

plt.show()

fig = plt.figure(figsize=(32,24))

ax1 = fig.add\_subplot(221)

ax1 = sb.stripplot(x='Activity', y=sub15.iloc[:,2], data=sub15, jitter=True)

ax2 = fig.add\_subplot(222)

ax2 = sb.stripplot(x='Activity', y=sub15.iloc[:,3], data=sub15, jitter=True)

plt.show()

fig = plt.figure(figsize=(32,24))

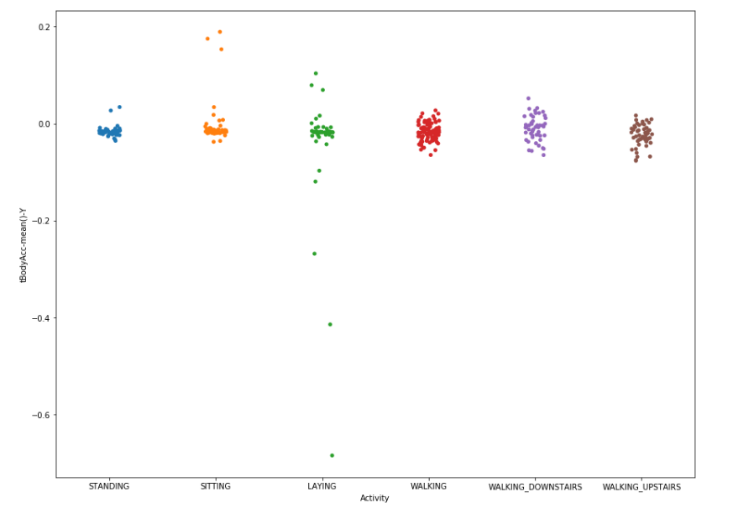
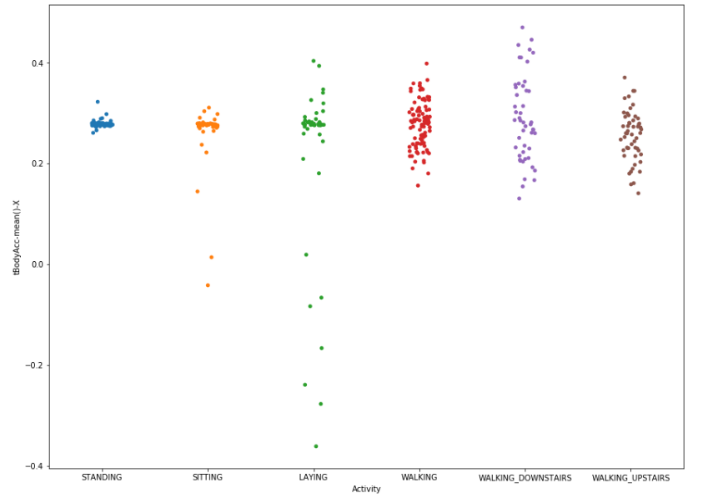
ax1 = fig.add\_subplot(221)

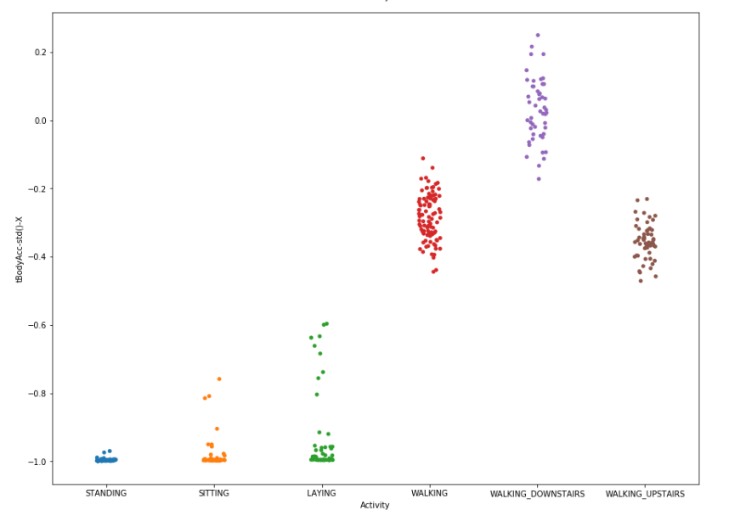
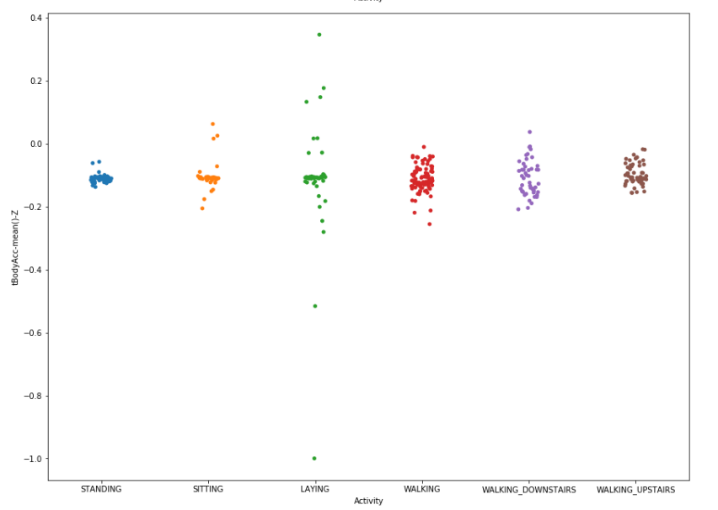
ax1 = sb.stripplot(x='Activity', y=sub15.iloc[:,4], data=sub15, jitter=True)

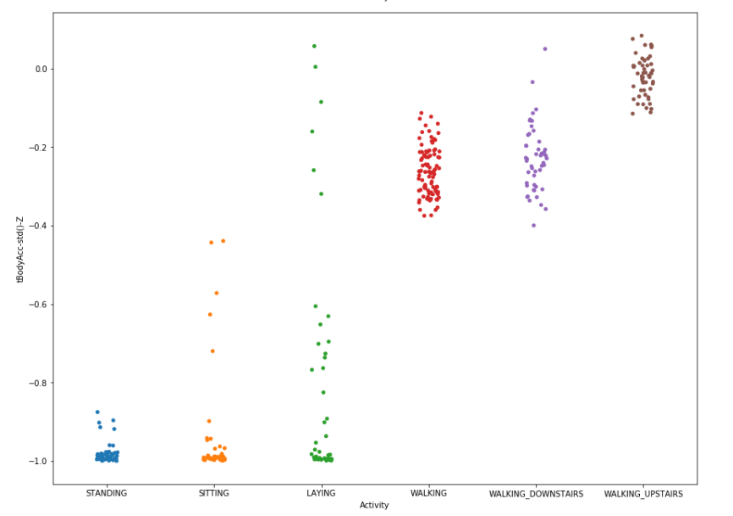
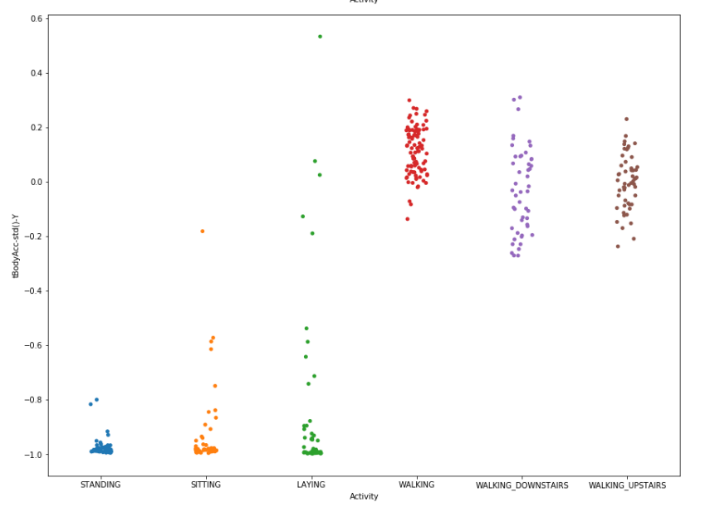
ax2 = fig.add\_subplot(222)

ax2 = sb.stripplot(x='Activity', y=sub15.iloc[:,5], data=sub15, jitter=True)

plt.show()







**Nice to have Requirements**

N1L **SGD Model:** Model Implementation.

Y2X **TensorFlow:** Model Implementation.

import tensorflow as tf # using TensorFlow library

from sklearn.utils import shuffle

from sklearn.preprocessing import LabelBinarizer #using LabelBinarizer

feature that highly corelated will be

Creating functions for weight, bias and add\_layer and Train the layers with Relu.

*def weight\_variable(shape):*

*return tf.Variable(tf.truncated\_normal(shape, stddev=0.1))*

*def bias\_variable(shape):*

*return tf.Variable(tf.truncated\_normal(shape, stddev=0.1))*

*def add\_layer(inputs, input\_size, output\_size, activation=None):*

*W = weight\_variable([input\_size, output\_size])*

*b = bias\_variable([output\_size])*

*wxb = tf.matmul(inputs, W) + b*

*if activation:*

*return activation(wxb)*

*else:*

*return wxb*

We are generating out tensorflow model with 4 layers here. Each layer will be assigned weight and bias values.

*X = tf.placeholder(tf.float32, [None, 562])*

*layer1 = add\_layer(X, 562, 1000, tf.nn.relu)*

*layer2 = add\_layer(layer1, 1000, 300, tf.nn.relu)*

*layer3 = add\_layer(layer2, 300, 50, tf.nn.relu)*

*output = add\_layer(layer3, 50, 6)*

*y\_ = tf.placeholder(tf.float32, [None, 6])*

Next we define loss and optimizer.

*loss = tf.reduce\_mean(tf.nn.sigmoid\_cross\_entropy\_with\_logits(labels=output, logits=y\_))*

*optimizer = tf.train.GradientDescentOptimizer(0.001)*

*train\_step = optimizer.minimize(loss)*

After running 10000 times the Tensor Flow Neural Network Model the error find is **0.18318318**

*for i in range(10000):*

*batch = np.random.choice(train\_features.shape[0], 100)*

*\_, cost = sess.run([train\_step, loss], feed\_dict = {X:train\_features[batch], y\_:train\_labels[batch]})*

*print(sess.run(score, feed\_dict={X: test\_features, y\_: test\_labels}))*

Y3X **GridSearch**: Apply to find the best parameters

GridSearch is use to find out the best parameters for the giving model. We will be using SVC model in GridSearch and we will identify parameters and values for the function. GridSearch should give use the best options for the best accuracy value.

import matplotlib.pyplot as plt

%matplotlib inline

ax = score\_df.plot.bar()

ax.set\_xticklabels(score\_df.index, rotation=45, fontsize=10)

from sklearn.model\_selection import GridSearchCV

parameters = {

  'kernel': ['linear', 'rbf', 'poly','sigmoid'],

   'C': [100, 50, 20, 1, 0.1]

}

selector = GridSearchCV(SVC(), parameters, scoring='accuracy')

selector.fit(X, y)

Best parameter set found:

{'C': 100, 'kernel': 'sigmoid'}

Detailed grid scores:

0.927 (+/-0.098) for {'C': 100, 'kernel': 'linear'}

0.925 (+/-0.095) for {'C': 100, 'kernel': 'rbf'}

0.915 (+/-0.105) for {'C': 100, 'kernel': 'poly'}

0.931 (+/-0.090) for {'C': 100, 'kernel': 'sigmoid'}

0.927 (+/-0.098) for {'C': 50, 'kernel': 'linear'}

0.926 (+/-0.092) for {'C': 50, 'kernel': 'rbf'}

0.905 (+/-0.097) for {'C': 50, 'kernel': 'poly'}

0.929 (+/-0.092) for {'C': 50, 'kernel': 'sigmoid'}

0.927 (+/-0.098) for {'C': 20, 'kernel': 'linear'}

0.927 (+/-0.089) for {'C': 20, 'kernel': 'rbf'}

0.900 (+/-0.090) for {'C': 20, 'kernel': 'poly'}

0.918 (+/-0.102) for {'C': 20, 'kernel': 'sigmoid'}

0.927 (+/-0.098) for {'C': 1, 'kernel': 'linear'}

0.859 (+/-0.097) for {'C': 1, 'kernel': 'rbf'}

0.604 (+/-0.043) for {'C': 1, 'kernel': 'poly'}

0.762 (+/-0.042) for {'C': 1, 'kernel': 'sigmoid'}

0.927 (+/-0.095) for {'C': 0.1, 'kernel': 'linear'}

0.398 (+/-0.028) for {'C': 0.1, 'kernel': 'rbf'}

0.388 (+/-0.001) for {'C': 0.1, 'kernel': 'poly'}

0.388 (+/-0.001) for {'C': 0.1, 'kernel': 'sigmoid'}

If we apply GridSearch for DecisionTreeClassifier we can find the best function parameters.

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

param\_grid = { 'criterion':['gini','entropy'],'max\_depth': np.arange(3, 15)}

# decision tree model

dtree\_model=DecisionTreeClassifier()

#use gridsearch to test all values

dtree\_gscv = GridSearchCV(dtree\_model, param\_grid, cv=6)

dtree\_gscv.fit(X, y)

GridSearchCV(cv=6, error\_score='raise-deprecating',

estimator=DecisionTreeClassifier(class\_weight=None,

criterion='gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,min\_samples\_leaf=1,   
min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

presort=False, random\_state=None,

splitter='best'), iid='warn', n\_jobs=None,

param\_grid={'criterion': ['gini', 'entropy'],

'max\_depth': array([ 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])}, pre\_dispatch='2\*n\_jobs', refit=True, return\_train\_score=False,

scoring=None, verbose=0)

N4L **Alternative measurements for sports:** Use alternative data that collected from smartphones using measurement taking tools and multi-location sensors such as distance sensor, accelerometer, gyroscope, magnetometer. This can be used to recognition additional sportive activities to such as Running, riding, exercising (Pull-ups, Push-ups, Squats, Sit-ups and Pull-ups etc.)

N5L **Application:** Come up with smartphone application to collect real time data and classify the activity.

## 3.3 Report on Design

Describe the design that you finally used. Indicate how, where, and why it differed from your planned design. Describe its advantages and its shortcomings. Include a description of how the technologies you explored (not the tools—those are described below) leveraged each other. Include at least one diagram. Limit: 2 pages of 12-point Times New Roman.

## 3.4 Tools

Describe the machine learning tool(s) that you used. Show samples. Describe their advantages and their shortcomings. (List other tools separately.) Limit: 1 page of 12-point Times New Roman.

I have used PyCharm and google collaborator to run the Python code. (<https://colab.research.google.com>) on Kaggle source data.

## 3.5 Contrast between approaches

You were to include two approaches to your problem, and implement at least one. Contrast the two approaches as they specifically relate to your project.

<please replace this with your response>

## 3.6 What did *not* work well

Explain the most important aspects of your project that fell short of your plans or desires.

I could not apply models with the data I am generation from the mobile app. I didn’t have time to complete all of my nice to have requirements.

## 3.7 What *did* work well

In paragraph form, explain the most important aspects of your project that met or exceeded your plans or desires.

I have managed to apply all the models that I wanted to research. I have learned great details on each. Understand which one to use in what cases. Also I realize how important to work with data before applying the models and understand the data in hand thoroughly.

## 3.8 Presentation Materials

Prepare a presentation for the last class based on the above. Include a short demonstration. This should last no more than 15 minutes. Include in this report copies of the slides, two per page.



## 3.9 Sample Source

Supply up to 1 page of key excerpts from your source code—or what comes closest to “source code.” Limit: 2 pages of 12-point Times New Roman. Include an explanation of where the excerpts fit in your implementation.

**import** pandas **as** pd  
test = pd.read\_csv(**"https://raw.githubusercontent.com/alpgulden/data/master/train.csv"**)  
train = pd.read\_csv(**"https://raw.githubusercontent.com/alpgulden/data/master/test.csv"**)  
  
train.head()  
  
train.Activity.value\_counts()  
  
train.shape,test.shape  
  
test.Activity.value\_counts()  
  
*# suffling data***from** sklearn.utils **import** shuffle  
test = shuffle(test)  
train = shuffle(train)  
  
*# separating data inputs and output lables*trainData = train.drop(**'Activity'** , axis=1).values  
trainLabel = train.Activity.values  
testData = test.drop(**'Activity'** , axis=1).values  
testLabel = test.Activity.values  
  
*# encoding labels***from** sklearn **import** preprocessing  
encoder = preprocessing.LabelEncoder()  
  
*# encoding test labels*encoder.fit(testLabel)  
testLabelE = encoder.transform(testLabel)  
  
*# encoding train labels*encoder.fit(trainLabel)  
trainLabelE = encoder.transform(trainLabel)  
  
  
*# classification models:  
  
# applying supervised neural network using multi-layer preceptron***import** sklearn.neural\_network **as** nn  
mlpSGD = nn.MLPClassifier(hidden\_layer\_sizes=(90,)  
 , max\_iter=1000 , alpha=1e-4  
 , solver=**'sgd'** , verbose=10  
 , tol=1e-19 , random\_state=1  
 , learning\_rate\_init=.001)  
  
mlpADAM = nn.MLPClassifier(hidden\_layer\_sizes=(90,)  
 , max\_iter=1000 , alpha=1e-4  
 , solver=**'adam'** , verbose=10  
 , tol=1e-19 , random\_state=1  
 , learning\_rate\_init=.001)  
  
mlpLBFGS = nn.MLPClassifier(hidden\_layer\_sizes=(90,)  
 , max\_iter=1000 , alpha=1e-4  
 , solver=**'lbfgs'** , verbose=10  
 , tol=1e-19 , random\_state=1  
 , learning\_rate\_init=.001)  
  
nnModelSGD = mlpSGD.fit(trainData , trainLabelE)  
  
nnModelSGD = mlpLBFGS.fit(trainData , trainLabelE)  
nnModelSGD  
  
nnModelADAM = mlpADAM.fit(trainData , trainLabelE)

## 3.10 Source

Attach source code (or what comes closest to it) and input where possible. You may refer the reader to github for source if you prefer.

<https://github.com/alpgulden/METCS767MachineLearning>

<https://github.com/alpgulden/data>

## References

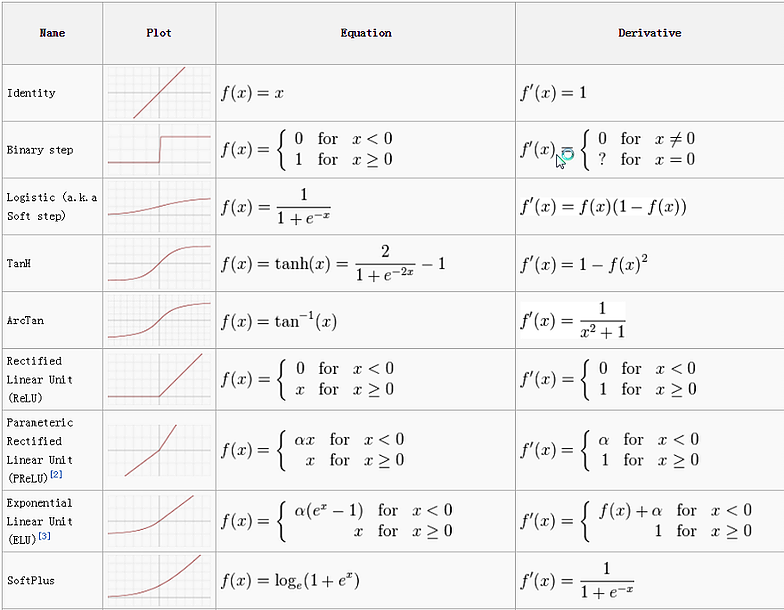
Show that you used a wide variety of resources by listing them below and clearly indicating in the body above where you used them.

[21] <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html> [22] <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>  
[23] <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>  
[24] <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>  
[25] <https://patrickhoo.wixsite.com/diveindatascience/single-post/2019/06/13/Activation-functions-and-when-to-use-them>  
[26] <https://stats.stackexchange.com/questions/1826/cross-validation-in-plain-english>  
[27] <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html>  
[28] <https://towardsdatascience.com/from-scikit-learn-to-tensorflow-part-2-66c56985d6c7>  
[29] <https://scikit-learn.org/stable/modules/neural_networks_supervised.html>  
[30] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>

# Appendices

If necessary, supply one or more appendices with material that you want to make available. Appendices will be read on an as-needed basis only. They should be referred to in the body of the paper.

## Appendix 1: Activation Functions



Activation Functions [25]

# 

# Evaluation

