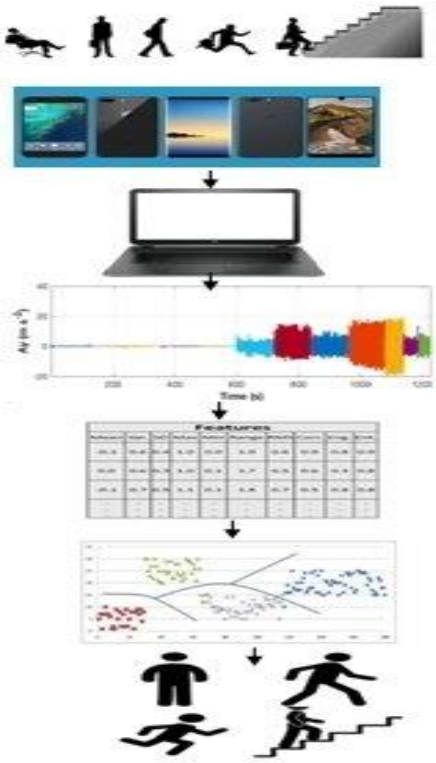
HUMAN ACTIVITY RECOGNITION USING SMARTPHONE SENSORS

ALAA SALIH

As422
https://github.com/Alaa422

CS 634 Data Mining
Final Term Project



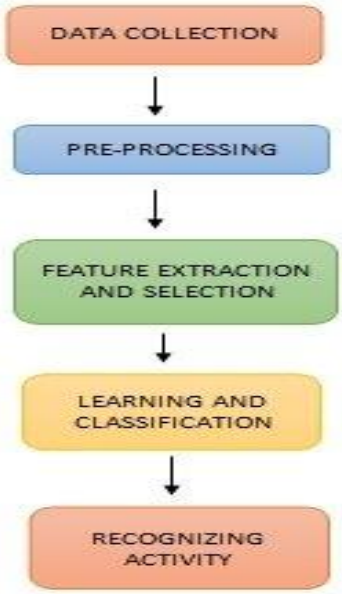


Table of Content:

INTRODUCTION

1.1 Introduction

This project demonstrates how to predict the type of physical activity (e.g., walking, climbing stairs) from tri-axial smartphone accelerometer data using supervised machine learning. Smartphone accelerometers are very precise, and different physical activities give rise to unique patterns of acceleration.

Smartphones have become irreplaceable part of human life. People carry Smartphones throughout the day. This enables smartphone sensors to collect data and hence lets system to detect human activity.

Human Activity Recognition (HAR) aims to identify the actions carried out by a person given a set of observations of him/her and the surrounding environment. Recognition can be accomplished by exploiting the information retrieved from various sources such as environmental or body-worn sensors. Our Aim is to classify the given activities in the form of a dataset into six labels namely sitting, standing, walking, climbing up, climbing down and laying. We present analysis of method for classifying activities, such as walking up stairs or standing, using data from a gyroscope and accelerometer. Analysis is informed by a visualization of the data. We analyse the differences in error rates between different methods.

1.2 Data Exploration

The dataset used for this paper is from the UCI Machine Learning Repository titled "Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set" (SBHAR). The data was collected from experiments with a group of 30 volunteers aged between 19 to 48 years wearing a Samsung Galaxy SII on the waist. Volunteers performed a protocol of activities including six basic postures: three static — standing, sitting, lying and three dynamic — walking, walking downstairs and walking upstairs. The experiment also included transitional postures between static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie and lie-to-stand. The sensor signals were collected from the accelerometer and gyroscope in the smartphones including 3-axial linear acceleration and 3-axial angular velocity at a constant refresh rate of 50Hz.

1.3 Signal Processing Method

According to Figure below, the raw signals of human daily activities such as walking, walking upstairs, walking downstairs, sitting, standing and laying are acquired from the inertial sensors of smartphones. Then, the raw signals are segmented using sliding windows and filtered using median and band pass Butterworth filters to remove the irrelevant information or noise. Next, feature extraction of time and frequency domain are implemented.

Thereafter, all the data of extracted features are classified with ensemble classifiers using base learners of SVM and RF. The performance of classification of is measured using performance evaluation metrics such as precision, recall, F-measure, accuracy and ROC.

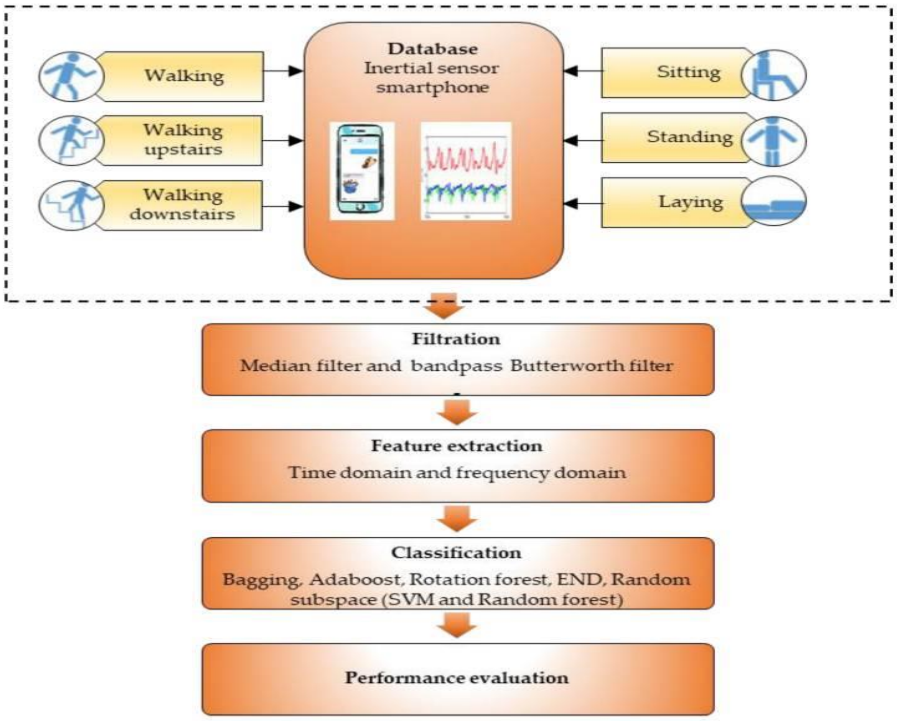


Figure 1

Sensors Processing

2. Project Description:

2.1 Software Specification

- Python 3.8 , Anaconda Jupiter Notebook
- Scikit-Learn: Machine learning software tool
- Libraries : Pandas, NumPy, Metrics and Matplotlib

2.2 Hardware configuration

Laptop processor Intel core i5 , Operating System windows 8.1 pro

2.3 Data source

Human Activity Recognition Using Smart Phone sensors dataset, from

http://archive.ics.uci.edu/ml/datasets

2.4 Framework Structure

The plan will be as follow:

- Import useful packages and libraries.
- Load dataset of smartphone signal from sensors.
- Cleaning and preparing data.
- Features Engineering.
- Data visualization and Analysis.
- Testing and Training.
- Modelling (3 Algorithms).
- Model Evaluation (Accuracy ,scoring and Confusion Matrix)
- Compare and analysis the results of all algorithms.

3. General Architecture:

The objective of Human activity recognition is to detect the actions performed by a person from a given set of the data about him/her and his surrounding environment. A lot of research is being done in the field of Human activity recognition which human behavior is interpreted by deducing features derived from movement, place, physiological signals and information from environment etc... Environmental and sensors which are 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. But these sensors are quite uncomfortable and cannot provide long term solutions.

3.1 Block diagram Approach :

We have collected data from University of California Machine Learning repository [3]. Data is imported, cleaned and normalized. In order to increase correctness and performance of our system we have reduced dimensions of our original dataset using Principal Component Analysis(PCA) technique. Reduced data is then processed through various supervised Machine Learning algorithms like Random Forest, Support Vector Machine and K-Nearest Neighbour to classify data into six categories namely Sitting, Standing, Laying, Walking, Walking Upstairs and Walking Downstairs. Correctness of the system is determined by generating confusion matrix and by random simulations.

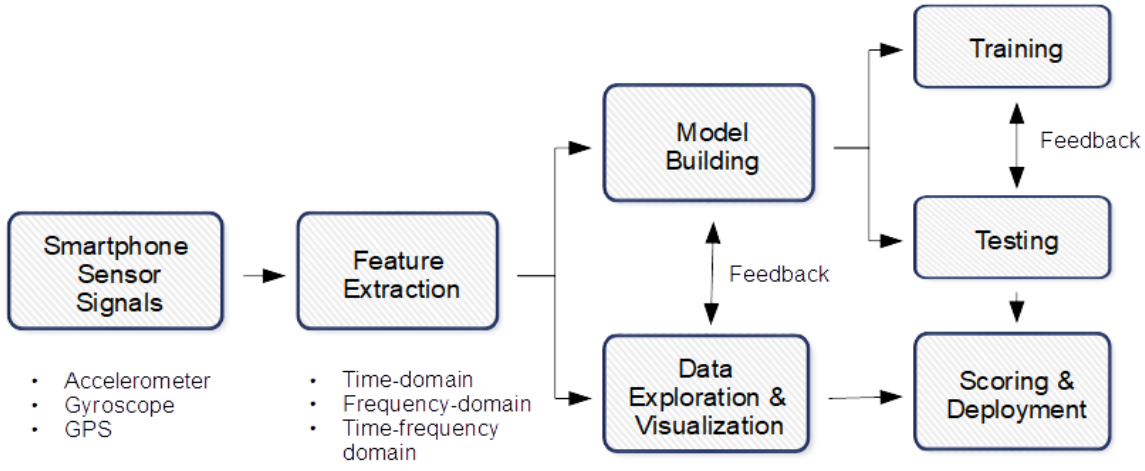


Figure 2

HAR machine learning process flow

- Input data transformation to address different sampling frequency of sensor signals and activity labels
- Feature extraction using frequency-domain analysis
- Data visualizations to visually differentiate signal patterns of each physical activity
- Model building by training classifiers with multiple algorithms and generating test scores from the best classifier

Will show how these steps are performed on Jupyter Notebook of complete Python implementation is accessible on my Github repository.

4. 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, which is why there is no clear distinction on which algorithm is more appropriate than others. The energy and memory consumption and complexity of the problem varies and form a set of conditions which decide which algorithm is more suitable for a particular problem.

4.1 Random forests

Random Forest Classifier builds a forest which is an ensemble of decision trees. It creates a set of decision trees from a randomly selected subset of the training data and aggregates the decision from all the trees to decide the final output. This technique is robust as it prevents the noisy output of some trees affecting the final decision and avoids overfitting. It cancels out the bias by averaging all the predictions. Random forest is differentiated from decision trees as it does not search for the best feature while splitting the node. It instead searches for the most appropriate feature from a subset of features. This provides diversity and randomness to the algorithm. The algorithm can be easily modeled to both, classification and regression problems.

4.2 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.

4.3 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.Frequency domain features provided better results than the time domain features with the use of SVM to classify 6 activities.

4.4 Logistic Regression

The logistic classification model (or logit model) is a binary classification model in which the conditional probability of one of the two possible realizations of the output variable is assumed to be equal to a linear combination of the input variables, transformed by the logistic function.

The achieved result shows that the Logistic Regression is more accurate as compared to other selected classifiers in this study for human activity recognition.

4.5 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.

Deep learning models have the capabilities to learn features of the higher order. Advancement in such models makes it conceivable to learn and improve the performance of the predictive models and find deeper knowledge from human activities.

5. Framework Implementation and results:

5.1 Importing and Cleaning Data

Data was collected from experimentation carried out by group of 30 volunteers. Each person was supposed to perform six activities (Walking, walk up, walk down, Laying, Standing and Sitting). They had used embedded sensors of Smartphone to collect data. Dataset was loaded into Python as .csv file.

```
train.head()
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyroJerkMag-kurtosis()	angle(tBo
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	-0.983185	-0.923527	-0.934724	...	-0.710304	
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	-0.974914	-0.957686	-0.943068	...	-0.861499	
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	-0.963668	-0.977469	-0.938692	...	-0.760104	
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	-0.982750	-0.989302	-0.938692	...	-0.482845	
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	-0.979672	-0.990441	-0.942469	...	-0.699205	

5 rows × 563 columns

Figure 3
Human Activity Recognition Dataset

There is no any possibility of having Outliers. All the values are squeezed between -1 to 1. And Checking for missing NaN/null values

```
train.describe()
#There is no any possibility of having Outliers. All the values are squeezed between -1 to 1.
```

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	tBodyAcc-mad()-Y	tBodyAcc-mad()-Z	tBodyAcc-max()-X	...	fBodyBodyGyr s
count	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	...	73
mean	0.274488	-0.017695	-0.109141	-0.605438	-0.510938	-0.604754	-0.630512	-0.526907	-0.606150	-0.468604	...	
std	0.070261	0.040811	0.056635	0.448734	0.502645	0.418687	0.424073	0.485942	0.414122	0.544547	...	
min	-1.000000	-1.000000	-1.000000	-1.000000	-0.999873	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	...	
25%	0.262975	-0.024863	-0.120993	-0.992754	-0.978129	-0.980233	-0.993591	-0.978162	-0.980251	-0.936219	...	
50%	0.277193	-0.017219	-0.108676	-0.946196	-0.851897	-0.859365	-0.950709	-0.857328	-0.857143	-0.881637	...	
75%	0.288461	-0.010783	-0.097794	-0.242813	-0.034231	-0.262415	-0.292680	-0.066701	-0.265671	-0.017129	...	
max	1.000000	1.000000	1.000000	1.000000	0.916238	1.000000	1.000000	0.967664	1.000000	1.000000	...	

8 rows × 562 columns

Figure 4
Cleaned Dataset

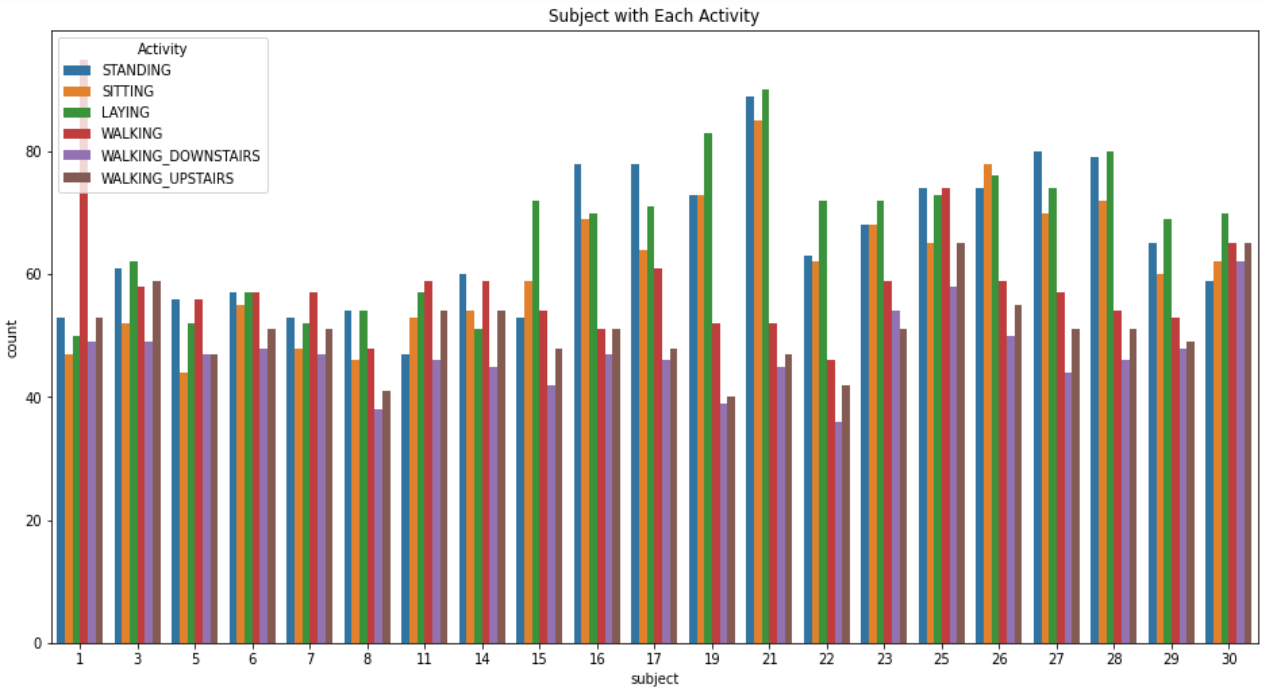


Figure 5
Check for imbalanced dataset

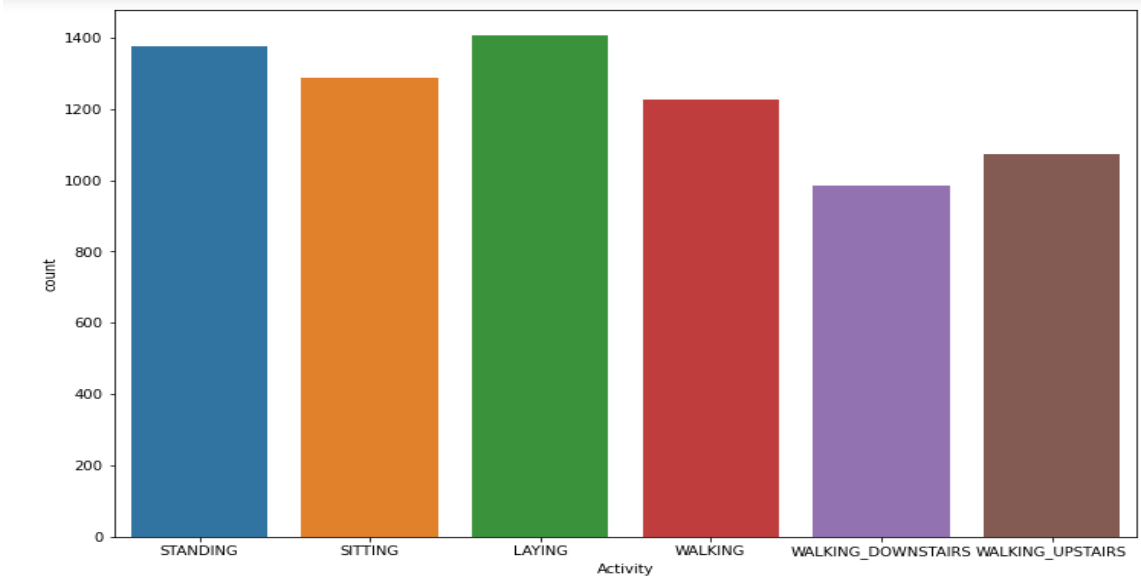


Figure 6
Frequency of Activities in Training Dataset

5.2 Machine Learning Model Evaluation:

A comparative analysis was carried on four classifiers to understand the improvement in the model after the feature selection. The four classification algorithms experimented on are Logistic Regression, Random Forrest Classifier, Support vector Machine, and KNN. The results were compared to the metric of the time taken to build and train the model and the accuracy of the model.

```
|: # Logistic regression model:
from sklearn.linear_model import LogisticRegression
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import matplotlib.cm as cm
```

```
|: # K-FOLD= 10
kfold = model_selection.KFold(n_splits=10, random_state=42)
kfold
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

```
|: logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
#probs_y=logmodel.predict_proba(X_test)
accuracy_scores1 = accuracy_score(y_test, predictions)*100
print('Logistic Regression accuracy: {}%'.format(accuracy_scores1))
```

```
Logistic Regression accuracy: 98.87005649717514%
```

Figure 7
Logistic regression Accuracy

```
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
```

	precision	recall	f1-score	support
LAYING	1.00	1.00	1.00	170
SITTING	0.97	0.97	0.97	152
STANDING	0.97	0.97	0.97	137
WALKING	0.99	1.00	1.00	154
WALKING_DOWNSTAIRS	1.00	0.98	0.99	123
WALKING_UPSTAIRS	0.99	1.00	1.00	149
accuracy			0.99	885
macro avg	0.99	0.99	0.99	885
weighted avg	0.99	0.99	0.99	885

Figure 8
Classification Report

```
# Support Vector Classifier
from sklearn.svm import SVC
clf2 = SVC().fit(X_train, y_train)
prediction = clf2.predict(X_test)
accuracy_scores2 = accuracy_score(y_test, prediction)*100
print('Support Vector Classifier accuracy: {}%'.format(accuracy_scores2))
```

```
Support Vector Classifier accuracy: 97.6271186440678%
```

```
from sklearn.metrics import confusion_matrix
y_pred=clf2.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[170,   0,   0,   0,   0,   0],
       [  0, 148,   4,   0,   0,   0],
       [  0,  11, 126,   0,   0,   0],
       [  0,   0,   0, 153,   0,   1],
       [  0,   0,   0,   1, 118,   4],
       [  0,   0,   0,   0,   0, 149]], dtype=int64)
```

Figure 9
SVM Accuracy

```
#Random Forest model
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores3 = accuracy_score(y_test, prediction)*100
print('Random Forest Classifier accuracy: {}%'.format(accuracy_scores3))
```

```
Random Forest Classifier accuracy: 97.17514124293785%
```

Figure 10
Random Forest Accuracy

```
# K Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier().fit(X_train, y_train)
prediction = knn.predict(X_test)
accuracy_scores4 = accuracy_score(y_test, prediction)*100
print('K Nearest Neighbors Classifier accuracy: {}%'.format(accuracy_scores4))
```

```
K Nearest Neighbors Classifier accuracy: 96.045197740113%
```

Figure 11
KNN Model Accuracy

After applied different machine learning algorithms; found that Logistic Regression performed the best in classifying different activities.

Model	Score
logistic regression	98.870056
LinearSVM	97.627119
RandomForest	97.175141
KNN	96.045198

Table 1
Comparison between Models

6. Conclusion:

In this Project, we have presented the general architecture utilized to build human activity recognition systems and emphasized the design issues such as selection of sensors, obtrusiveness, flexibility, etc. which are independently evaluated based on the kind of system which is being developed.

The achieved result shows that the Logistic Regression is more accurate as compared to other selected classifiers in this study for human activity recognition. So Better feature selection methods and improvement in tuning the parameters can assist further to improve accuracy.
In future adding further layers to the network or increasing the complexity would further boost the recognition accuracy of the deep learning algorithm.

References

1- http://archive.ics.uci.edu/ml/datasets
2- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6308488/
3- https://medium.com/@xiaoshansun
4- D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, "Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine," in *Ambient Assisted Living and Home Care*, Lecture Notes in Computer Science, vol 7657. Springer, Berlin, Heidelberg, 2012
5- Data Mining and Machine Learning Lectures Notes

Appendix:

```
# Load the libraries

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

#Load Dataset
train = pd.read_csv("data/train.csv")
test = pd.read_csv("data/test.csv")
train['Activity'].value_counts()
#Handling outliers:
train.describe()
#Checking for missing  NaN/null values
print("Total Null values in Train: {}\n".format(train.isnull().values.sum()))
print("Total Null values in Test: {} \n".format(test.isnull().values.sum()))
#Check for imbalanced dataset
plt.figure(figsize = (16,8))
plt.title("Subject with Each Activity")
sns.countplot(hue = 'Activity', x='subject',data = train);
plt.show()
# There is no any huge amount of gap between them.
plt.figure(figsize = (12,8))
sns.countplot(x = 'Activity', data = train);
# Correcting feature names by remove ()columns = train.columns
columns = columns.str.replace('[()]','')
columns = columns.str.replace('[-]','')
columns = columns.str.replace('[,]','')
train.columns = columns
test.columns = columns
```

```
train.columns

# Splitting training and testing
X_train = train.drop(["subject","Activity"], axis = 1)
y_train = train.Activity

X = test.drop(["subject","Activity"], axis = 1)
y = test.Activity

print('Training data size:', X.shape)
print('Test data size:', X.shape)
model_score = pd.DataFrame(columns = ("Model","Score"))

Models and Cross Validations
# Logistic regression model:
from sklearn.linear_model import LogisticRegression
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import matplotlib.cm as cm

# K-FOLD= 10
kfold = model_selection.KFold(n_splits=10, random_state=42)
kfold
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=42)
logmodel = LogisticRegression()
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
#probs_y=logmodel.predict_proba(X_test)
accuracy_scores1 = accuracy_score(y_test, predictions)*100
print('Logistic Regression accuracy: {}%'.format(accuracy_scores1))

Logistic Regression accuracy: 98.87005649717514%
from sklearn.metrics import confusion_matrix
y_pred=logmodel.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
model_score = model_score.append(pd.DataFrame({'Model':["logistic regression"],'Score':[accuracy_scores1]}))
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))
# Support Vector Classifier
from sklearn.svm import SVC
```

```
clf2 = SVC().fit(X_train, y_train)
prediction = clf2.predict(X_test)
accuracy_scores2 = accuracy_score(y_test, prediction)*100
print('Support Vector Classifier accuracy: {}%'.format(accuracy_scores2))
Support Vector Classifier accuracy: 97.6271186440678%
from sklearn.metrics import confusion_matrix
y_pred=clf2.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
model_score = model_score.append(pd.DataFrame({'Model':["LinearSVM"],'Score':[accuracy_scores2]}))
#Random Forest model
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores3 = accuracy_score(y_test, prediction)*100
print('Random Forest Classifier accuracy: {}%'.format(accuracy_scores3))
Random Forest Classifier accuracy: 97.17514124293785%
model_score = model_score.append(pd.DataFrame({'Model':["RandomForest"],'Score':[accuracy_scores3]}))
# K Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier().fit(X_train, y_train)
prediction = knn.predict(X_test)
accuracy_scores4 = accuracy_score(y_test, prediction)*100
print('K Nearest Neighbors Classifier accuracy: {}%'.format(accuracy_scores4))
K Nearest Neighbors Classifier accuracy: 96.045197740113%
from sklearn.metrics import confusion_matrix
y_pred=knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
model_score = model_score.append(pd.DataFrame({'Model':["KNN"],'Score':[accuracy_scores4]}))
```

Also measured based on the comparison of overall accuracy rate between different classifiers

```
# Compare between models
model_score.head()
```

	Model	Score
0	logistic regression	98.870056
0	LinearSVM	97.627119
0	RandomForest	97.175141
0	KNN	96.045198