Automatic Human Activity Recognition using Smartphone Sensors

Jyotik Parikshya, *CISE Department, University of Florida*

***Abstract*— Humans now have a very complex and active lifestyle for a few days and thus have little time to concentrate on their everyday tasks and exercises. An application that automatically monitors all basic tasks and movements such as running, walking or sleeping is necessary to get a rough estimate of the calorie burnt and the quality of sleep. Embedded smartphones with motion sensor have provided a new activity inference platform. Such devices, which were originally used for improving mobile phone functionality, are now being used in a number of applications. Acknowledging human behavior as one of these. An app was created to serve this function and can take user data from the smartphone and predict the type of movement or activity. Walking, running, climbing stairs, descending stairs, cycling and sleeping are the supporting categories in this project. Therefore the identification of human physical activities from measurements of cell phone sensors is conceived as a question of detection. Many machine learning classification algorithms can solve this classification issue. Via the use of decision tree algorithm, several researchers have shown the most accuracy. Through making necessary changes, I will seek to boost the accuracy and running time of Support Vector Machine, K-nearest Neighbors and Naïve Bayes. The ultimate goal of this project is to create a faster and more accurate algorithm for tracker applications for human activity.**

I. INTRODUCTION

THIS part of the document describes the problem statement and the project specifications.

Recognition of human behavior is the issue of categorizing samples of accelerometer data collected by expert vests or gadgets into well-defined recognised gestures.

Given the large number of observations generated per second, the transient nature of the observation and the absence of a simple way to connect accelerometer data to known actions, it is a challenging problem.

Traditional methods to the challenge incorporate manual craftsmanship of features from graph data focused on fixed-size frames and then training and testing the machine learning models . The challenge is that this engineering capability demands profound field competence.

First step will to convert the Time Series to Discrete Time Data. Signals received are in the form of time series. We convert the data to a more usable format by sampling. Fast Fourier Transform is used to Segregate different frequency signals and generate independent features. These numbers are then Standardised and finally stored in a tabular form.

Second step involves data verification and classifier design. Data is prepared for each design and parameters are decided and some criterion.

Third step involves Quantitative Analysis of results and improvising on implemented models.

Fourth step contains analysis of feature extraction and creating a model and analysing results.

II. DESCRIPTION

A. Data Collection and pre-processing

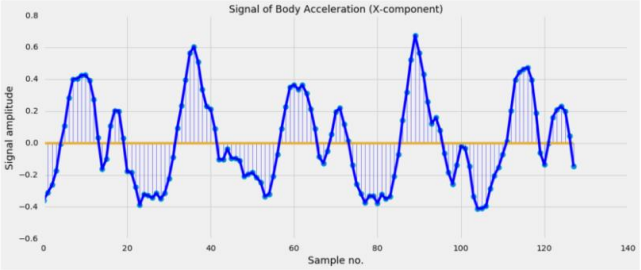
The First Phase of the project is to collect data to train and test the designed Model. This also involves data pre-processing to take care of all the missing and abrupt data.

The tests were performed with a group of 30 participants wit hin a 19-48 year age range.Each person performed six activities(WALKING,WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist.We obtained 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz, using its integ rated accelerometer and gyroscope. The experiments were vide o- recorded to manually mark the data.

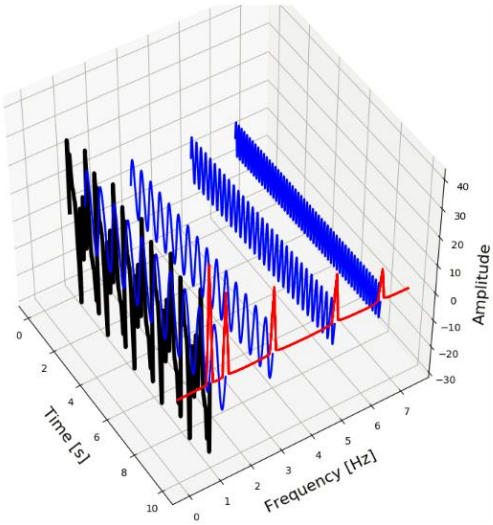
The data collected was randomly partitioned into two groups, where 70 percent of the volunteers were chosen to produce the training data and 30% of the test data.

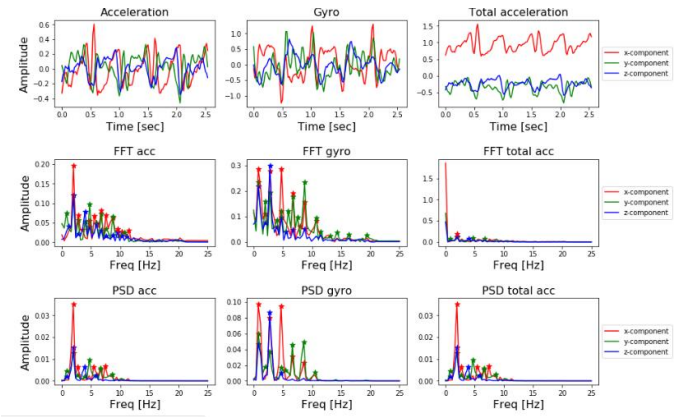
The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in 2.56 sec and 50 percent overlap fixed-width sliding windows (128 readings / window). The sensor acceleration signal, which has gravitational and body motion components, was divided into body acceleration and gravity using a Butterworth low-pass filter. It is believed that the gravitational force has only components of low frequency, so a filter of cut-off frequency of 0.3 Hz was used. A vector of features was obtained from each window by measuring time- and frequency domain variables.

For each record in the dataset:-Accelerometer triaxial acceleration (total acceleration) and estimated acceleration of the body as well as Triaxial Gyroscopic angular velocity is given. A 561-feature vector of time and domain frequency variables is generated operation. An identifier for the participant who performed the experiment. Training data hs around 7352 data points.



The data is taken from UCI Machine Learning Repository which suites best for this project's requirements.



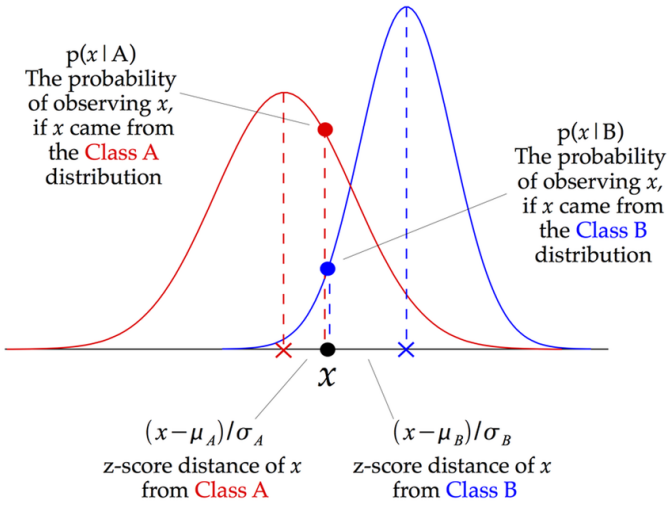


As recommended by the Data-set provider in the repository these steps have to e performed to get best results.

B. Algorithm Designing for NAÏVE BAYES

The Naïve Bayes classifier is a probabilistic classifier based on Bayes' Theorem. Two variants of this has been implemented – Gaussian and Bernoulli Naïve Bayes.

First, Gaussian Naïve Bayes was implemented. It is one of the simplest and basic Model with no arguments. It is based on the Gaussian Probabilistic Model.



GaussianNB of sklearn was used to implement the classifier. All the inputs were reals so gaussian distribution can be assumed. Accuracy turns out to be 77.027 %.

Second, Bernoulli Naïve Bayes was implemented. It is similar to the model of Gaussian but in Bernoulli's Naïve Bayes all the features assume binary values.

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{n} p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)}$$

BernoulliNB of sklearn was used to implement this classifier. Accuracy turned out to be 85.012%

Next, a new type of model was designed that is primarily Bernoulli aided by Gaussian method. Whenever the output of Bernoulli is not equal to Gaussian's output we refer to the probability distribution of Bernoulli Naïve Bayes and if Probability is less than a particular fraction then the gaussian output is chosen over Bernoulli. In this implementation the ideal number came out to be 0.98.

```
for i in range(len(ypredb)):
    if ypredb[i]!=ypredg[i]:
        if yprobb[i][ypredb[i]-1]<0.98:
            ypredb[i]=ypredg[i]
            print yprobb[i]
```

The accuracy increased to 85.6%.

C. *Algorithm Designing for K-Nearest Neighbor*

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The entire training dataset is stored. When a prediction is required, the k-most similar records to a new record from the training dataset are then located. From these neighbours, a summarized prediction is made. The meaning of nearest is very important to define. Two different distances were used - Manhattan and Euclidean to train the model and test the output.

KNeighborsClassifier of sklearn was used. The parameters used here are p which determines the type of metric space Manhattan or Euclidean and k which is the number of nearest neighbours. As value of k can vary and with it the accuracy of the model. The experiment was conducted to iterate for all value of k and get a plot for both Manhattan and Euclidean.

```
scores = []
for i in range(1, 50):
    knn = KNeighborsClassifier(n_neighbors = i,p=1, n_jobs = -1)
    knn.fit(xtrain, ytrain)
    ypred = knn.predict(xtest)
    scores.append(accuracy_score(ytest,ypred))
```

p=1 stands for Manhattan while the default value 2 stands for Euclidean.

Accuracy of 92.29 % was possible at k=18 in Manhattan space.

D. *Algorithm Designing for Support Vector Machine*

SVMs are supervised learning models with associated learning algorithms that analyze data used for classification. SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. Two versions of SVM will be implemented – linear and RBF. While Linear SVM is one of the simplest ones, RBF Kernel is built for high accuracy and works well with high number of features.

The Parameters ‘C’ and ‘gamma’ has to be carefully selected for maximum accuracy. While ‘C’ is common for both Linear and RBF, ‘gamma’ is specific to RBF. We will try many combinations of both gamma and C for Linear and RBF.

```
classifier=svm.SVC()

parameters=[{'kernel': ['rbf'], 'gamma': [0.001, 0.0001], 'C': [1, 10, 100, 1000]},
            {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]

model=GridSearchCV(classifier,parameters,n_jobs=-1,cv=4,verbose=4)
model.fit(xtrain.as_matrix(),ytrain.as_matrix().ravel().T)

from sklearn.metrics import accuracy_score
ypred=model.predict(xtest)
accuracy=accuracy_score(ytest,ypred)

print 'Best Parameters: '+ str(model.best_params_)
print 'Accuracy Score: '+ str(accuracy*100) + ' %'
```

With help of GridSearchCV all permutations of both linear and RBF with All combinations of C and gamma were tested. The accuracy turned out to be 96.53% at C=1000 and gamma= 0.001 in RBF Kernel.

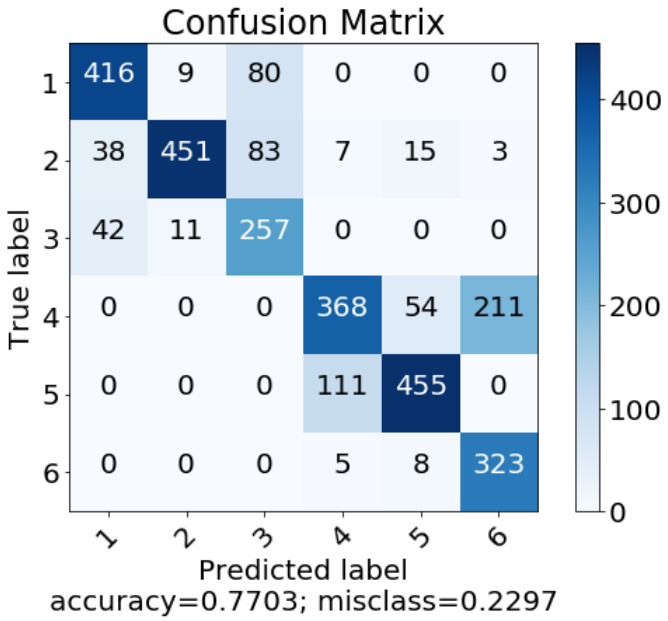
E. *Feature Extraction and Modelling*

After completing the modelling and implementation part we try to reduce the number of features to check if any of the features are dependent on each other. For this purpose we use Independent Component Analysis (ICA) to reduce to i components I varying from 2 to 100. After reducing the training data we train the Naïve Bayes and SVM Model and look for potential reductions but the results show that it was not the case.

III. EVALUATION

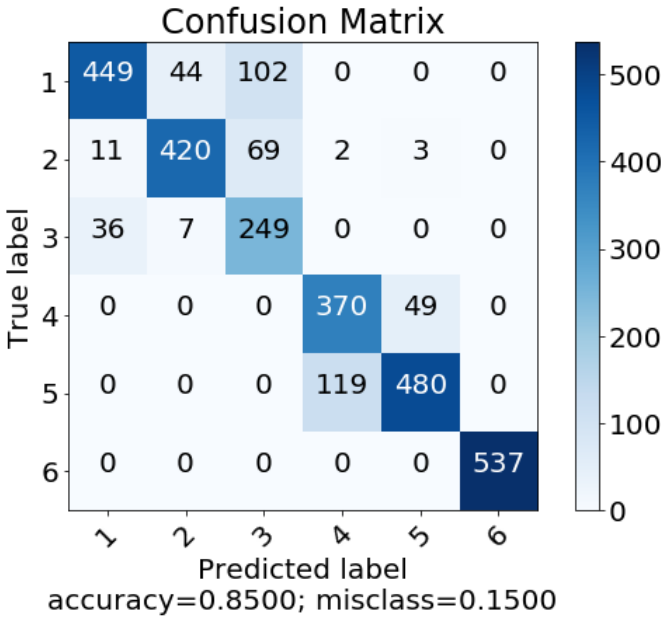
This section of the document will show all the results, conclusions and some modifications.

Starting with **Gaussian Naïve Bayes**, the confusion matrix looks like:

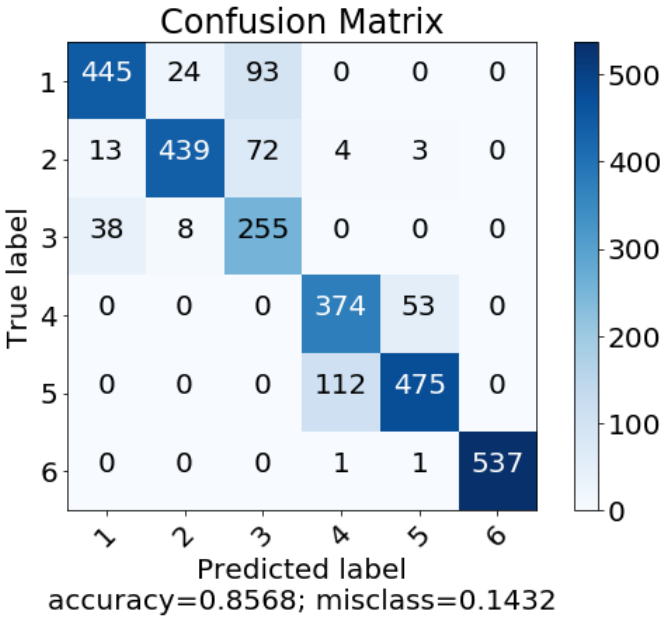


As seen the accuracy is not up to the mark and the confusion matrix gives a clear picture.

Next Looking at **Bernoulli Naïve Bayes,** the confusion matrix is little better than Gaussian as the accuracy goes up to 85.001 :

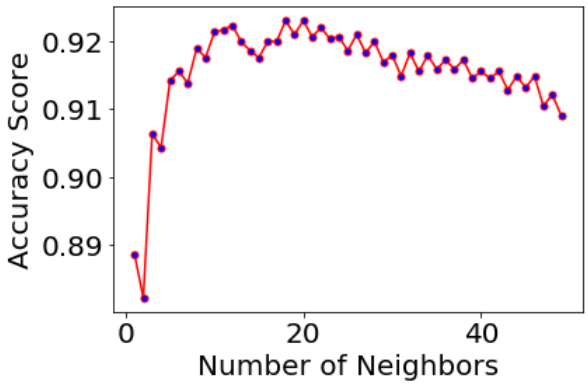


Here, a **combination of both variants** were implemented with the following confusion matrix:



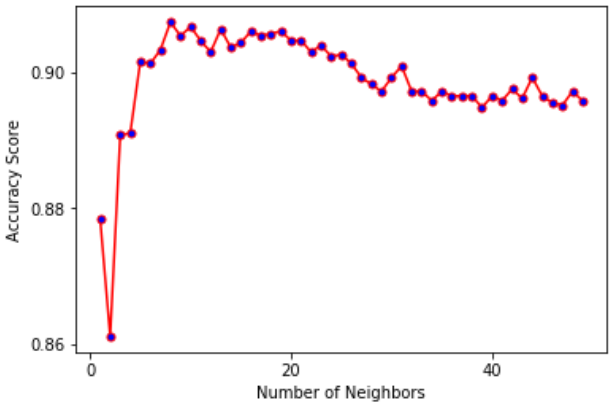
Next for **implementation of KNN** the following plot was obtained:

Manhattan:

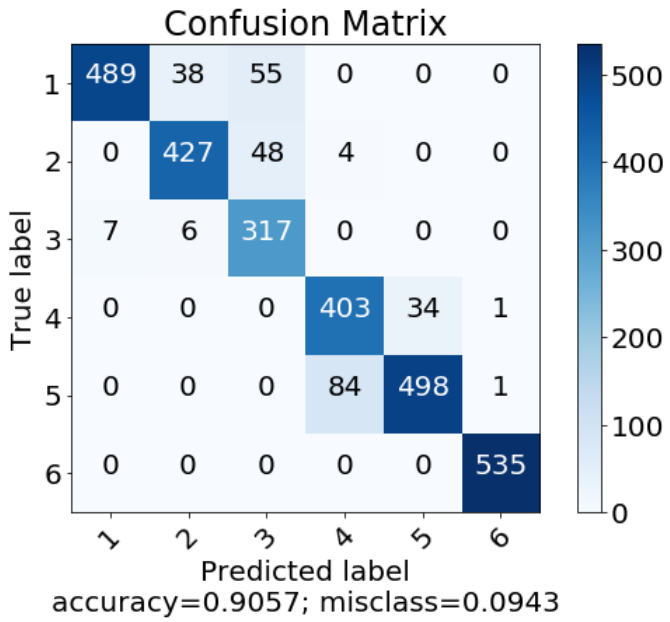



```
Optimal No. Of Neighbors:  18
Accuracy Score: 92.29725144214456 %
```

Euclidean:

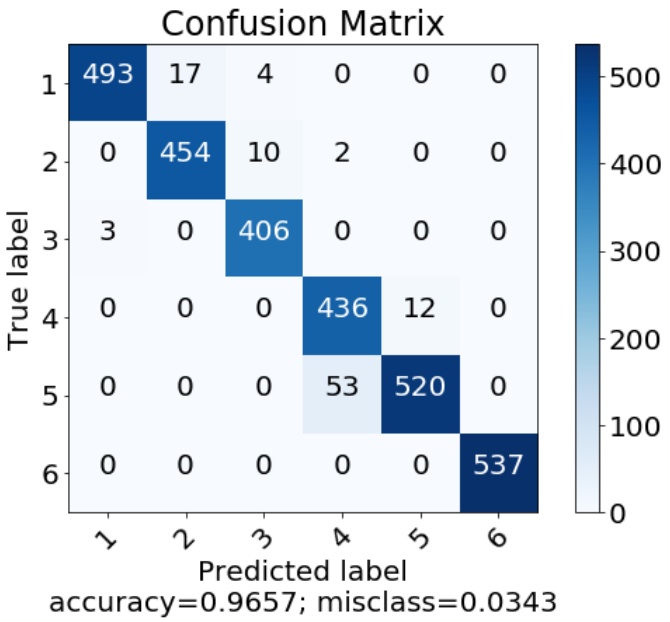



```
Optimal No. Of Neighbors:  8
Accuracy Score: 90.73634204275535 %
```



Evidently, Manhattan works better than Euclidean. But optimal value for both are at different k's.

Next implementing **SVM** we get the confusion matrix :



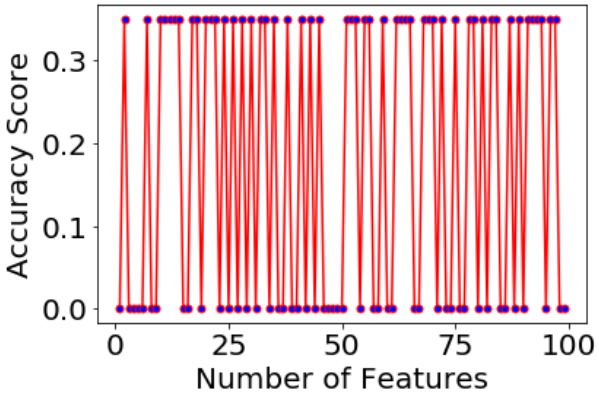
Clearly SVM is the winner here. 96.57% accuracy is very high for such a big data set.

For experimenting with reducing the number of features and testing it's independence the below logic was implemented:

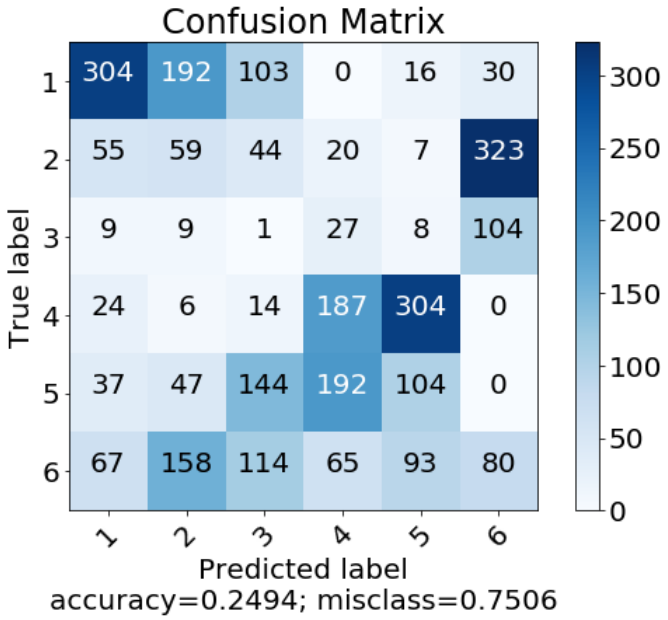
```
from sklearn.decomposition import FastICA
scores=[]
for i in range(1,100):
 ica = FastICA(n_components=i)
 xtrain = ica.fit_transform(xtrain)
 xtest = ica.fit_transform(xtest)
```

After which both SVM and Naïve were tested:

SVM results were too bad (below 25% at best) and Naïve Bayes gave this plot:




```
Optimal No. Of Neighbors:  2
Accuracy Score: 34.98473023413641 %
```



Confusion Matrix shows predictions are no where near the actual results.

Although one thing to be noticed is that the percentage increase in misclass is for (5,4) from 53 to 192 and 53 was the highest misclass in the previous Confusion Matrix. This might be a indicator that there are still more features to be extracted which help classify these 53.

Hence all features are necessary to classify the points.

IV. RELATED WORK

There has been a lot of work in this area both published and unpublished.

Starting from data processing, sampling of this data is a huge task, although they have extracted 561 features people are still working on finding some stray signals to reach that 98-99 accuracy mark. As mentioned earlier in the report there is a sign of new features. Although on the other side of earth people have pointed out that 31 features are enough for fitting. They have sampled it very differently and have been able to achieve accuracy of about 95.2% which much less computation. It remains work under progress. Many new data sets are created daily and modern fit bits and mobile applications are implementing different models to attain fast and accurate result. But somehow the balance between speed and accuracy is always a puzzle to solve.

Some researchers have used KNN, Decision Tree, ANN, Random Forest to attain high accuracy and have succeeded to obtain more than 97 on this data set but these models struggle with bigger data sets.

The biggest challenge still lies in signal processing and running FFTs in proper frames. All this process happens every time there is a signal so it is not a one time process. Hence it needs to be fast as well as accurate.

V. Summary and Conclusions

To Summarize I believe the best way is to tabulate the discussed results:

Model Type	Percentage Accuracy
Gaussian Naïve Bayesian	77.03
Bernoulli Naïve Bayesian	85.00
Bernoulli aided Gaussian	85.65
KNN-Manhattan	92.29
KNN-Euclidean	90.73
SVM	96.57

The Best way to predict Human Activity based on this dataset is SVM model with RBF kernel which gives an accuracy of 96.57%.

While doing this project I tried many things, much of which was not producing the desired result but is a very good learning experience. For instance, I was trying to build a combination of several classifiers by predicting what the majority of classifiers predict, tie-breaker is decided by checking their individual Accuracy.

For this particular model, it doesn't work as there is a set of points in class 4 that gets predicted as class 5 in all classifiers and this shortcoming is not due to any classifier but the nature of data. There might be some way to normalize or standardize the data more for particular uses or find some new features in the form of residual or stray signals. It is a very interesting area to be working on.

In conclusion, various classifiers can be used for Human activity prediction. Among the ones I implemented SVM turned out to be the best with accuracy of 96.57. Also, the lower bound for number of independent features extracted from signals comfortably sits at 361.

VI. References

[1] A. Anjum and M. U. Ilyas, "Activity recognition using smartphone sensors," 2013 IEEE 10th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, 2013, pp. 914-919.

[2] Mandong, Al-Montazer & Munir, Usama. (2018). Smartphone Based Activity Recognition using K-Nearest Neighbor Algorithm. B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.

[3] https://archive.ics.uci.edu/ml/datasets/Smartphone-Based%2BRecognition%2Bof%2BHuman%2BActivities%2Band%2BPostural%2BTransitions

[4] J. Kwapisz, G. Weiss, and S. Moore, "Activity recognition using cell phone accelerometers, " ACM SIGKDD Explorations Newsletter, vol. 12, no. 2, pp. 74-82, 2011.

[5] https://machinelearningmastery.com/how-to-model-human-activity-from-smartphone-data/