جامعة الحسين التقنية
Al Hussein Technical University

Human Activity Recognition Classification Using Python

Hala Waseem Aye

Advisors:

Dr. Ibrahim Abu-Alhaol
Dr. Mohammad Abu-Alhoul
Dr. Rawan Alkurd
Dr. Omar Mesmar
Mr. Ghassan Gammoh

January 28, 2021

Contents

Abstract

Human activity recognition is an emerging field of research. Recognizing human activity has gained a lot of interest due to the rapid development of technology and Smart Phones especially, in the field of motion sensors. It has become critical to extract knowledge from the information the sensors provide. In our work we examined 5 different classification algorithms on the Human Activity Recognition Data set.[3] We obtained very promising results with classification accuracy of human activities recognition of up to 98%.

Keywords

Classification, Principal Component Analysis, Python, Human Activity Recognition

1 Introduction

Smartphones are the most useful tools in our daily lives. With the advanced technologies, they are now more capable of fulfilling consumer needs and demands every day. To make these gadgets even more functional, designers add sensors to the mobile devices to collect a huge amount of information about the user's daily activities.[1] The main significance of this research lies in extracting knowledge from the data acquired by extensive sensors has become a very active area of research. As Smartphones technologies and sensors are growing rapidly; it has become very critical to classify different users' activities for many reasons. Classifying users' activities can be used in health monitoring applications. Such as, patients of diabetes, obesity, or heart diseases are required to follow a specific exercise routine. [4] Classifying the activities accurately will help them monitor the improvement. Also, elderly patients who are suffering from dementia or other mental issues can be monitored to detect abnormal activities in order to prevent undesirable consequences.[9] Our main interest in this research is to build machine learning classification models to classify different human activities and compare results with other researches.

2 Background

Classification of human activities has been the main focus in different studies using different machine learning algorithms. In [1] this study, a data set consists of signals from accelerometer and gyroscope of a smartphone while doing different activities are classified using Decision Trees, Support Vector Machine and K-nearest neighbors (KNN). In [8] they declared k- nearest neighbors (KNN) as the best classifier, but still failed to effectively classify very similar activities. Another research has found that the placement of the device affects the accuracy of measured activity; placing the device on the thigh is the most accurate for classifying activities. In [6] they discussed Human activity recognition with smartphone sensors using deep learning neural networks using greedy-wise tuning of hyper-parameters and, achieved the best performance of approximately 97% with 3 dense layers. In [5] they studied the Correlation Analysis-Based Classification of Human Activity Time Series using both Principal Component Analysis and Canonical Correspondence analysis (CCA), to cluster together consecutive data points that belong to the same event. Achieving the accuracy result of 98.8% for PCA and 76.7 % for CCA.

3 Methodology

We used python to perform the following activities:

(A) Data

We used the Kaggle Human Activity Recognition data set [3], with a small modification from our instructor. The Human Activity Recognition data set was built from the recordings of 30 study participants performing activities of daily living, while carrying a waist-mounted smartphone with embedded inertial sensors. The objective is to classify activities into one of the six activities performed (WALKING, WALKING_UPSTAIRS, LAYING, SITTING, WALKING_DOWNSTAIRS, STANDING). It consisted of 591 feature ,9990 row and, a class feature.

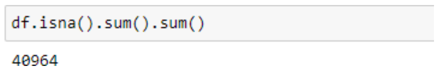
(B) Preprocessing and Exploratory Data Analysis

- **Concatenating Test and Train Data sets**

  We started with excluding the class feature and concatenating the train and test data sets so we can do all preprocessing on the whole data.

- **Check NA values**

  One of the most important things to check at first is, seeing if there is any nulls in the data. After checking the data in figure 1 it turned out that we have 40000 null values located in 30 features. We decided to drop those features because, it will not affect the classifiers as a matter of fact it will produce noise and consumes unnecessary resources.



```
df.isna().sum().sum()
40964
```

Figure 1: Null Values

- **Check Class Balance**

  After excluding the nulls from our data, we calculated the number of instances per class to check the number of instances for every class, as shown in Figure 2. Classes were unbalanced with a slight difference; such difference will not affect the performance of the classifiers.

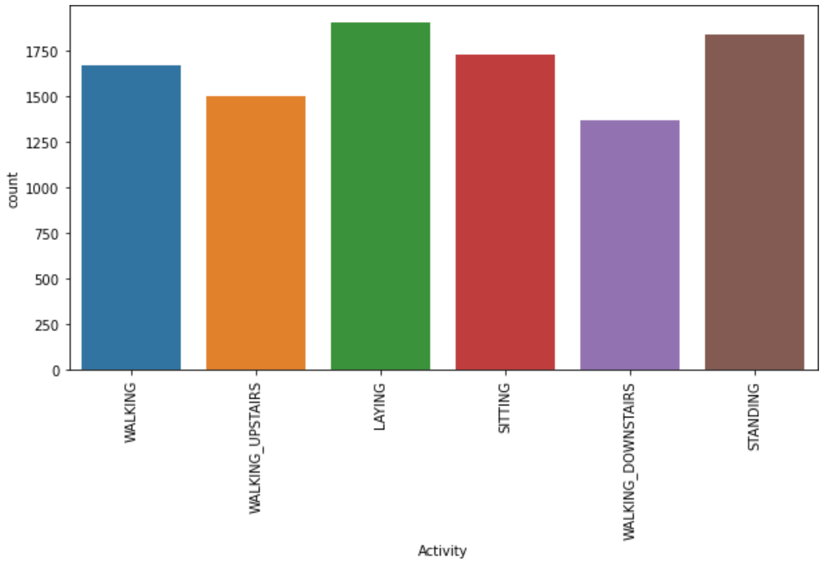


Figure 2: Unique Activity Count

(C) Feature selection and Principal component analysis (PCA)

Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning.[7] High dimensionality means that the data set has a large number of features. The primary problem associated with high-dimensionality in the machine learning field is model over-fitting, which reduces the ability to generalize beyond the examples in the training set [2]. To reduce dimensionality therefor, reduce computational power and, avoid over-fitting. A vital part of using PCA in practice is the ability to estimate how many components are needed to describe the data. We used the cumulative explained variance ratio as our measurement of the number of components.

This curve in Figure 3 quantifies how much of the total, 591-dimensional variance is contained within the first N components. For example, we see that with the digits the first 100 components contain approximately 97% of the variance, while you need around 200 components to describe close to 100% of the variance. So, we choose 200 components for our PCA.

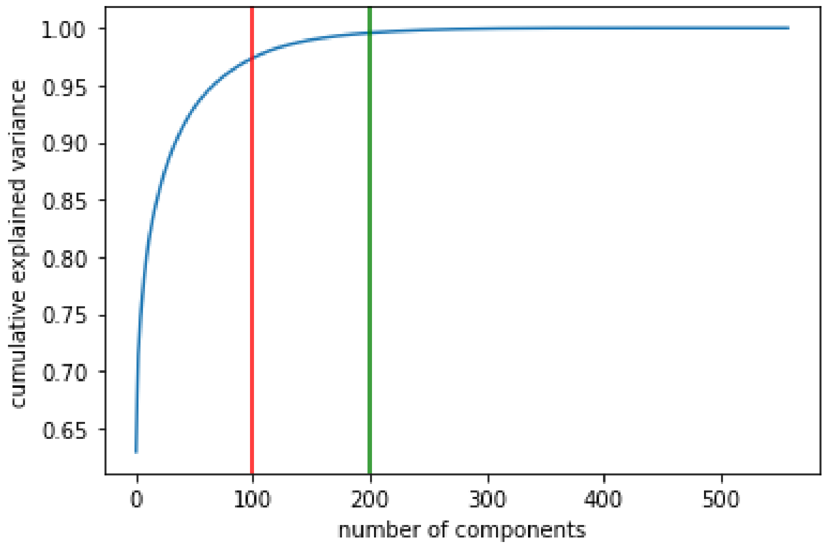


Figure 3: Principal Component Analysis

(D) Classifiers

In this research we discussed the following classifiers which you will find the screen shots of in the appendix:

(a) Random Forest Classifier (Figure 6)

(b) Light Gradient Boosting Machine Classifier (Figure 5)

(c)K-Nearest Neighbors Classifier (Figure 7)

(d)Logistic Regression (Figure 8)

(e)Support Vector Classifier (Figure 9)

4 Results and Discussion

Some previous researchers have found that K-Nearest Neighbor and Light Gradient Boosting Ma- chine Classifier gave the highest accuracy. Here in our work, we found that they scored approximately the same with a slight difference 0.49% The following results were the accuracy of our models. Unlike the previous researches, the best accuracy score was the Logistic Regression Classifier with the accuracy of 98.08%.

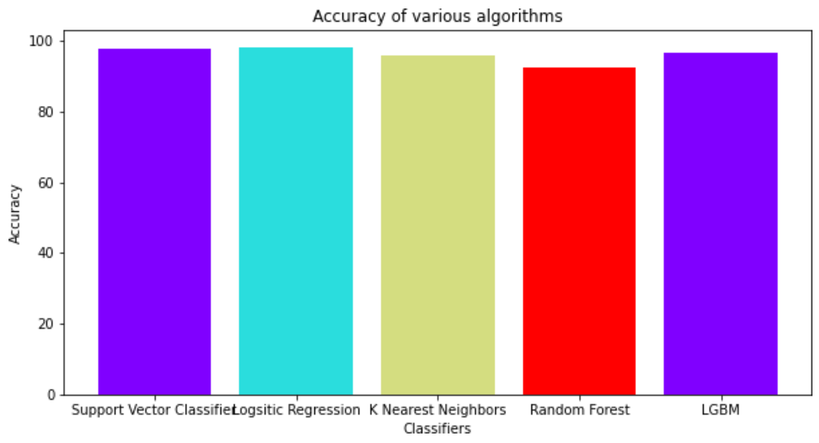


Figure 4: Classifiers Comparison

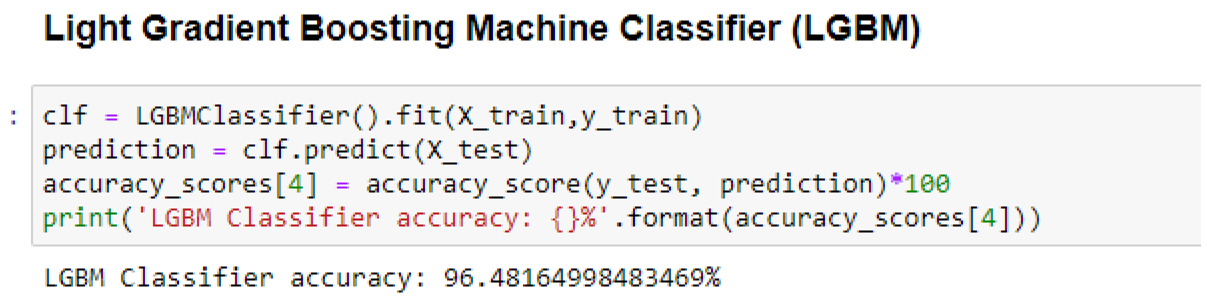
Table 1: Classifier Accuracy.

Classifier	Accuracy
Support Vector Classifier	97.69
Logistic Regression Classifier	98.08
K Nearest Neighbors Classifier	95.99
Random Forest Classifier	92.35
LGBM Classifier	96.48

5 Conclusions

This research was aimed to extract knowledge from human activities recognition data set, to identify different users' activity from the information gathered by the different sensors. Dissimilar to previous results conducted from previous research we deduced different results. K-Nearest Neighbors and Light Gradient Boosting Machine Classifier (LGBM) did not score the best accuracy results. In fact, Logistic Regression Classifier got the highest accuracy of 98.08%. The closest accuracy score to Logistic Regression was Support Vector Classifier 97.69%. K-Nearest Neighbors and Light Gradient Boosting Machine Classifier (LGBM) scored approximately the same with a slight difference the score of 96%.

6 Appendix

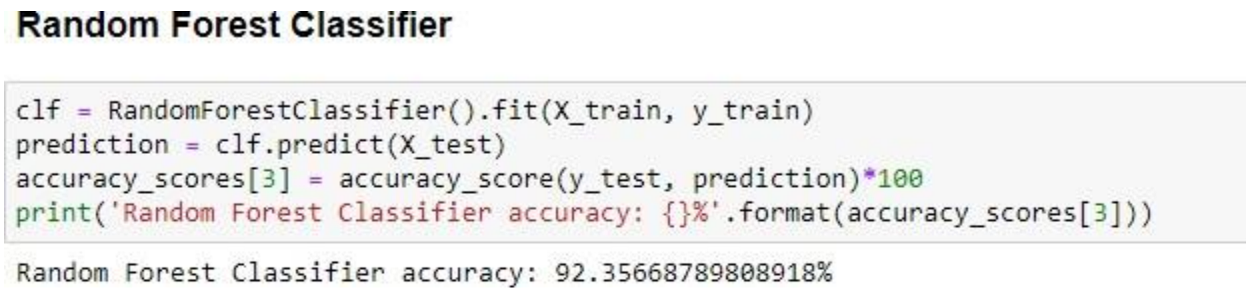


Light Gradient Boosting Machine Classifier (LGBM)

```
clf = LGBMClassifier().fit(X_train,y_train)
prediction = clf.predict(X_test)
accuracy_scores[4] = accuracy_score(y_test, prediction)*100
print('LGBM Classifier accuracy: {}%'.format(accuracy_scores[4]))
```

```
LGBM Classifier accuracy: 96.48164998483469%
```

Figure 5: LGBM Classifier

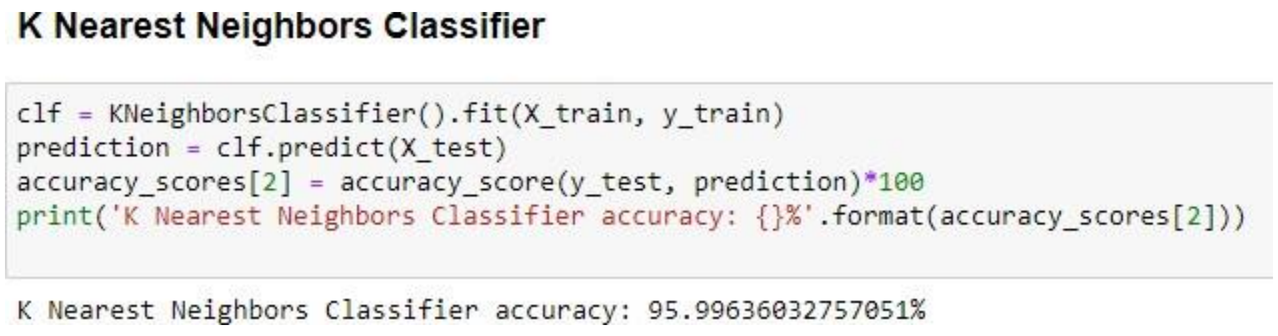


Random Forest Classifier

```
clf = RandomForestClassifier().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[3] = accuracy_score(y_test, prediction)*100
print('Random Forest Classifier accuracy: {}%'.format(accuracy_scores[3]))
```

```
Random Forest Classifier accuracy: 92.35668789808918%
```

Figure 6: Random Forest Classifier

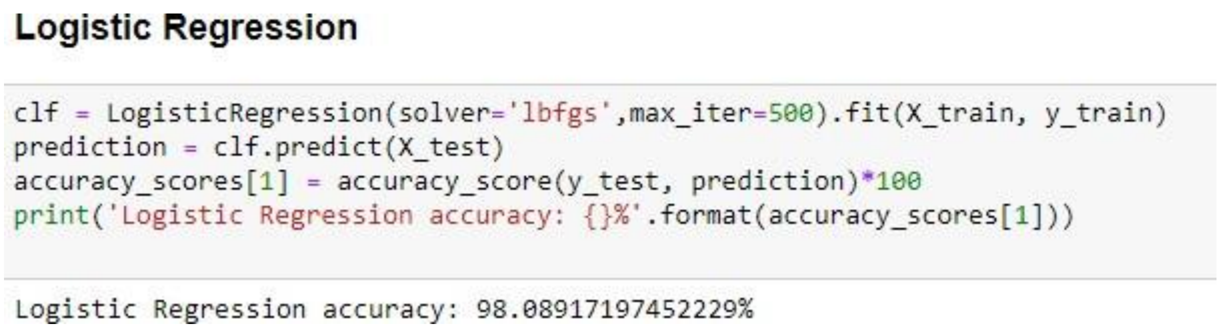


K Nearest Neighbors Classifier

```
clf = KNeighborsClassifier().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[2] = accuracy_score(y_test, prediction)*100
print('K Nearest Neighbors Classifier accuracy: {}%'.format(accuracy_scores[2]))
```

```
K Nearest Neighbors Classifier accuracy: 95.99636032757051%
```

Figure 7: K-Nearest Neighbor Classifier

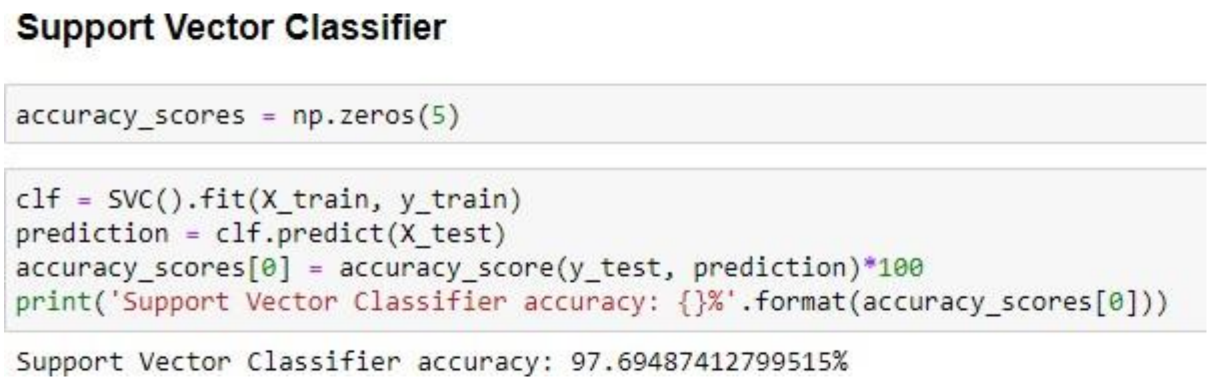


Logistic Regression

```
clf = LogisticRegression(solver='lbfgs',max_iter=500).fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[1] = accuracy_score(y_test, prediction)*100
print('Logistic Regression accuracy: {}%'.format(accuracy_scores[1]))
```

```
Logistic Regression accuracy: 98.08917197452229%
```

Figure 8: Linear Regression



Support Vector Classifier

```
accuracy_scores = np.zeros(5)
```

```
clf = SVC().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[0] = accuracy_score(y_test, prediction)*100
print('Support Vector Classifier accuracy: {}%'.format(accuracy_scores[0]))
```

```
Support Vector Classifier accuracy: 97.69487412799515%
```

Figure 9: Support Vector Classifier

References

[1] Erhan Bulbul, Aydin Cetin, and Ibrahim Alper Dogru. Human activity recognition using smartphones. In *2018 2nd international symposium on multidisciplinary studies and innovative technologies (ismsit)*, pages 1–6. IEEE, 2018.

[2] Apprentice Journal. Principle component analysis. https://cutt.ly/Yj7qKPQ.

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

[4] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials*, 15(3):1192–1209, 2012.

[5] Akshay Malhotra, Ioannis D Schizas, and Vangelis Metsis. Correlation analysis-based classification of human activity time series. *IEEE Sensors Journal*, 18(19):8085–8095, 2018.

[6] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with smartphone sensors using deep learning neural networks. *Expert systems with applications*, 59:235–244, 2016.

[7] Harri Valpola. From neural pca to deep unsupervised learning. In *Advances in independent component analysis and learning machines*, pages 143–171. Elsevier, 2015.

[8] Wanmin Wu, Sanjoy Dasgupta, Ernesto E Ramirez, Carlyn Peterson, and Gregory J Norman. Classification accuracies of physical activities using smartphone motion sensors. *Journal of medical Internet research*, 14(5):e130, 2012.

[9] Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1082–1090, 2008.