Machine learning and Artificial Intelligence

Human activity recognition – Mikołaj Zieliński

Introduction

Human Activity Recognition (HAR) is one of machine learning fields of interest. HAR involves the automatic identification and classification of various human activities using data gathered from sensors. This capability has been used across several domains, such as healthcare, sports, security, and human-computer interaction. By enabling systems to understand and predict human behaviours, HAR enhances the quality of life and provides innovative solutions to complex problems

Motivation

I have been working on a project to recognize human activities using data from smartphone sensors. I'm interested in this because it's important for things like understanding how people move and behave, which can help with health monitoring and detecting health problems.

My goal was to create a computer program that can understand what activities a person is doing by looking at the data from their phone's sensors.

This project matters because we can make use of data which come from sensors we carry with ourselves all the time (smartphones). This might be more and more important taking into consideration the development of IoT. Moreover monitoring our activity can make the tracking of our lifestyle much easier.

I tried few strategies such as: kNN, SVM, Random Forest and Neural Network. Every one of them gave me satisfying results. However the best results were obtained by Neural Network.

Dataset

https://www.kaggle.com/datasets/uciml/human-activity-recognition-with-smartphones

Description of measurements:
„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."*

The data I'm using comes from smartphone sensors (accelerometers and gyroscopes). These sensors capture information about motion and orientation, which are directly related to human activities like walking, standing, or sitting. For instance, accelerometer data records the acceleration measured by the phone along its x, y, and z axes, which can provide insights into the type of activity being performed. Gyroscope data measures the rate of rotation around each axis, which further contributes to recognizing the movements.

Dataset is composed of 563 columns and 2947 rows. Each row contains data from accelerometer and gyroscope. For example, columns which contains data from accelerometer have prefix "tBodyAcc". There are 106 values with this prefix:

mean()-X	mean()-Y	mean()-Z	std()-X	std()-Y	std()-Z	mad()-X	mad()-Y	mad()-Z	max()-X	max()-Y	max()-Z
min()-X	min()-Y	min()-Z	sma()	energy()-X	energy()-Y	energy()-Z	iqr()-X	iqr()-Y	iqr()-Z	entropy()-X	entropy()-Y
entropy()-Z	arCoeff()-X,1	arCoeff()-X,2	arCoeff()-X,3	arCoeff()-X,4	arCoeff()-Y,1	arCoeff()-Y,2	arCoeff()-Y,3	arCoeff()-Y,4	arCoeff()-Z,1	arCoeff()-Z,2	arCoeff()-Z,3
arCoeff()-Z,4	correlation()-X,Y	correlation()-X,Z	correlation()-Y,Z	erk-mean()-X	erk-mean()-Y	erk-mean()-Z	erk-std()-X	erk-std()-Y	erk-std()-Z	erk-mad()-X	erk-mad()-Y
erk-mad()-Z	erk-max()-X	erk-max()-Y	erk-max()-Z	erk-min()-X	erk-min()-Y	erk-min()-Z	erk-sma()	erk-energy()-X	erk-energy()-Y	erk-energy()-Z	erk-iqr()-X
erk-iqr()-Y	erk-iqr()-Z	erk-entropy()-X	erk-entropy()-Y	erk-entropy()-Z	erk-arCoeff()-X,1	erk-arCoeff()-X,2	erk-arCoeff()-X,3	erk-arCoeff()-X,4	erk-arCoeff()-Y,1	erk-arCoeff()-Y,2	erk-arCoeff()-Y,3
erk-arCoeff()-Y,4	erk-arCoeff()-Z,1	erk-arCoeff()-Z,2	erk-arCoeff()-Z,3	erk-arCoeff()-Z,4	erk-correlation()-X,Y	erk-correlation()-X,Z	erk-correlation()-Y,Z	ag-mean()	ag-std()	ag-mad()	ag-max()
ag-min()	ag-sma()	ag-energy()	ag-iqr()	ag-entropy()	ag-arCoeff()1	ag-arCoeff()2	ag-arCoeff()3	ag-arCoeff()4	erkMag-mean()	erkMag-std()	erkMag-mad()
erkMag-max()	erkMag-min()	erkMag-sma()	erkMag-energy()	erkMag-iqr()	erkMag-entropy()	erkMag-arCoeff()1	erkMag-arCoeff()2	erkMag-arCoeff()3	erkMag-arCoeff()4		

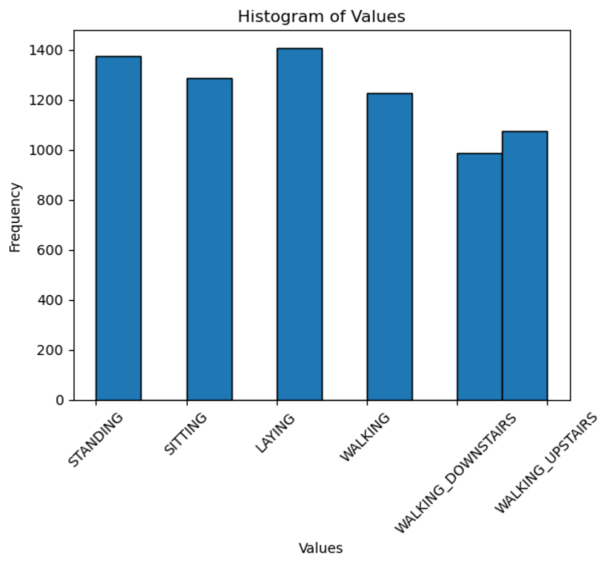
The values from gyroscope looks in similar way. Every value is already normalized to values between -1 and 1.

Every row is categorized to 1 from 6 activites such as:

- Sitting
- Standing
- Lying
- Walking
- Walking upstairs

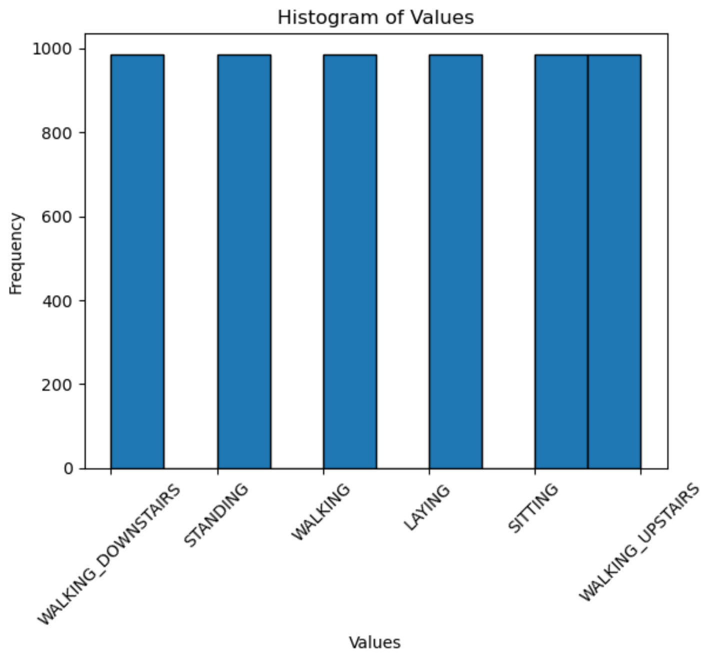
- Walking downstairs

The initial distribution of activities is presented on the histogram

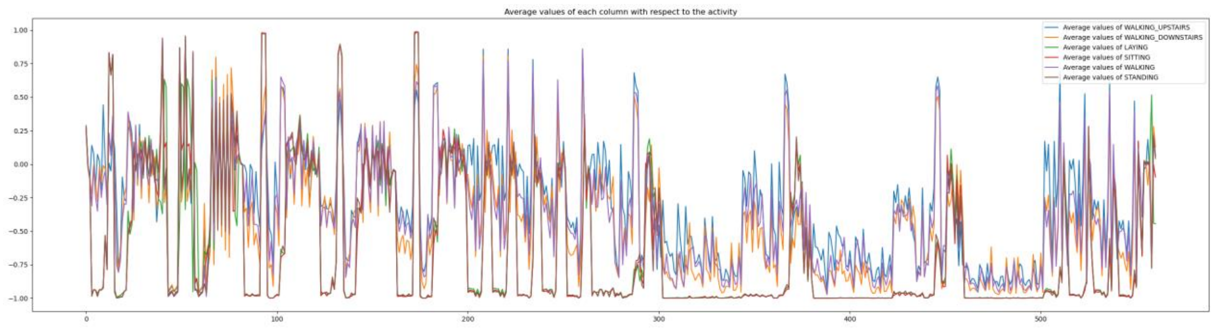


Data preprocessing and feature engineering

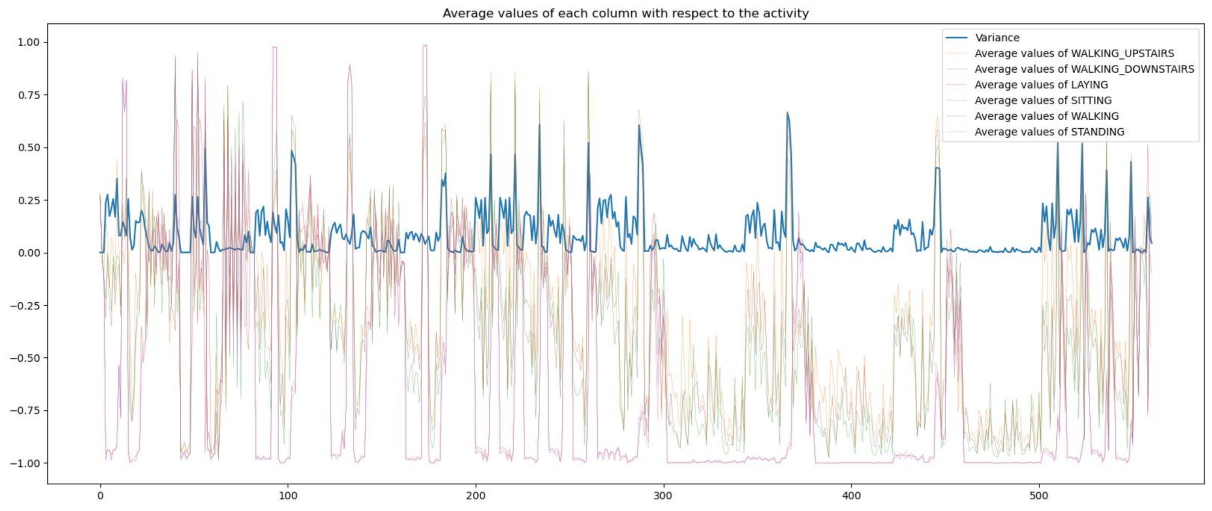
Firstly I performed undersampling in order to balance the dataset. Now every category contains 986 records.



Secondly I wanted to decrease the number of features. In order to find the least meaningful features I calculated the mean value of every feature with respect to each activity.



Having average values of each column with respect to each activity I calculated the variance from those 6 values from each column.



Than I was able to drop 200 features with lowest variance.

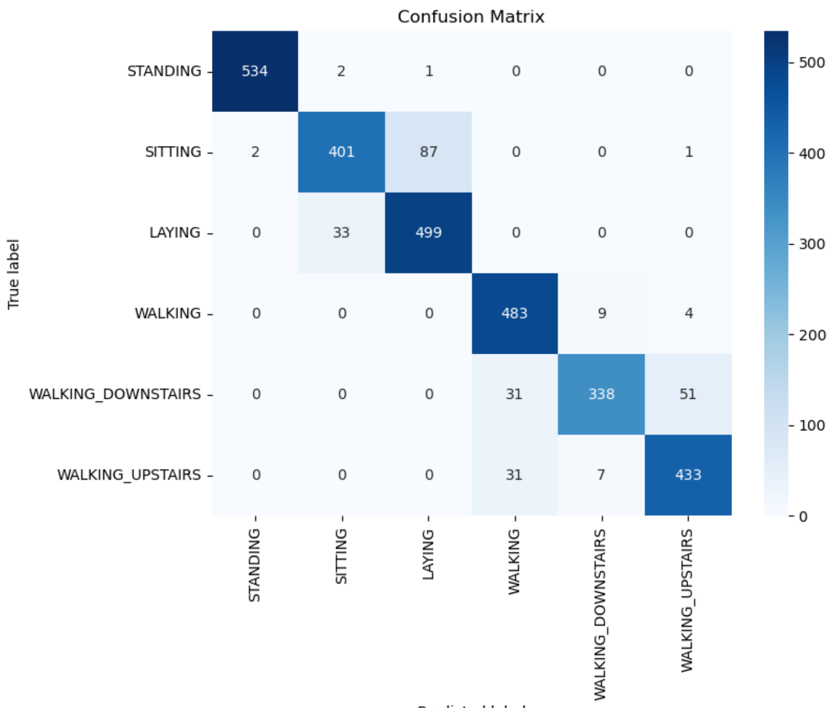
Models

I took inspiration from bin Abdullah, Mohd Fikri Azli, et al. "Classification algorithms in human activity recognition using smartphones." *International Journal of Biomedical and Biological Engineering* 6.8 (2012): 362-369. In this paper the authors mentions approaches such as: kNN, Random Forest, SVM and Neural Network. This were the methods I decided to implement. In order to find most optimal parameters of kNN, SVM and Random Forest I used grid search. To evaluate models I calculated: accuracy, precision, recall and F1 score. In our problem we can focus on accuracy, this was also the metric I optimized for in grid search. Below we can see the best parameters found in grid search, confusion matrix and metrics of every model I have implemented.

kNN

```
Best parameters found:
```

- `'knn__metric': 'manhattan'`
- `'knn__n_neighbors': 6`
- `'knn__weights': 'distance'`

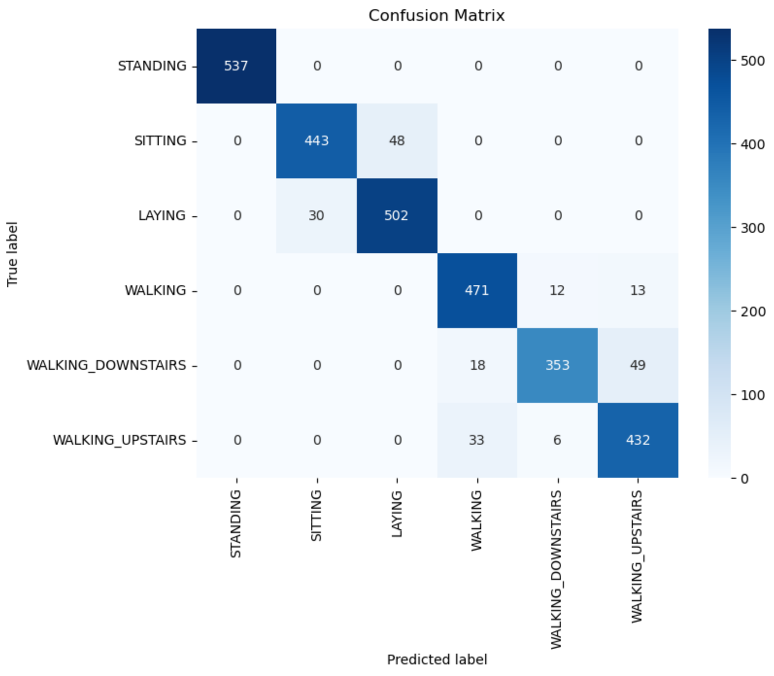


```
Accuracy: 0.9121140142517815
Recall: 0.9078261275756004
Precision: 0.9154333284714156
F1 score: 0.9092977603507179
```

Random forest

Best parameters found:

- 'criterion': 'gini'
- 'max_depth': 20
- 'max_features': 'log2'
- 'n_estimators': 300

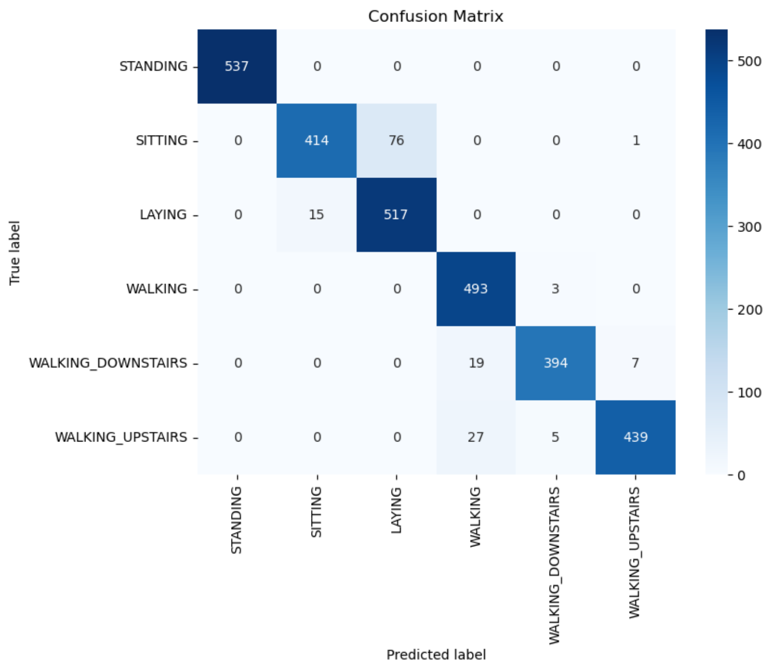


```
Accuracy: 0.9290804207668816
Recall: 0.925519960886835
Precision: 0.9295962638443301
F1 score: 0.9267033497560614
```

SVM

`Best parameters found:`

- `'C': 100`
- `‘degree': 1`
- `'gamma': 'auto'`
- `'kernel': 'rbf'`

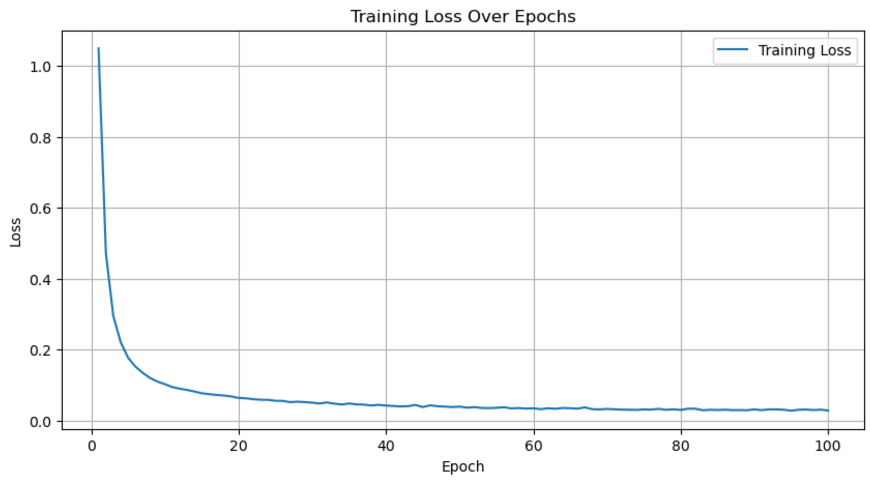


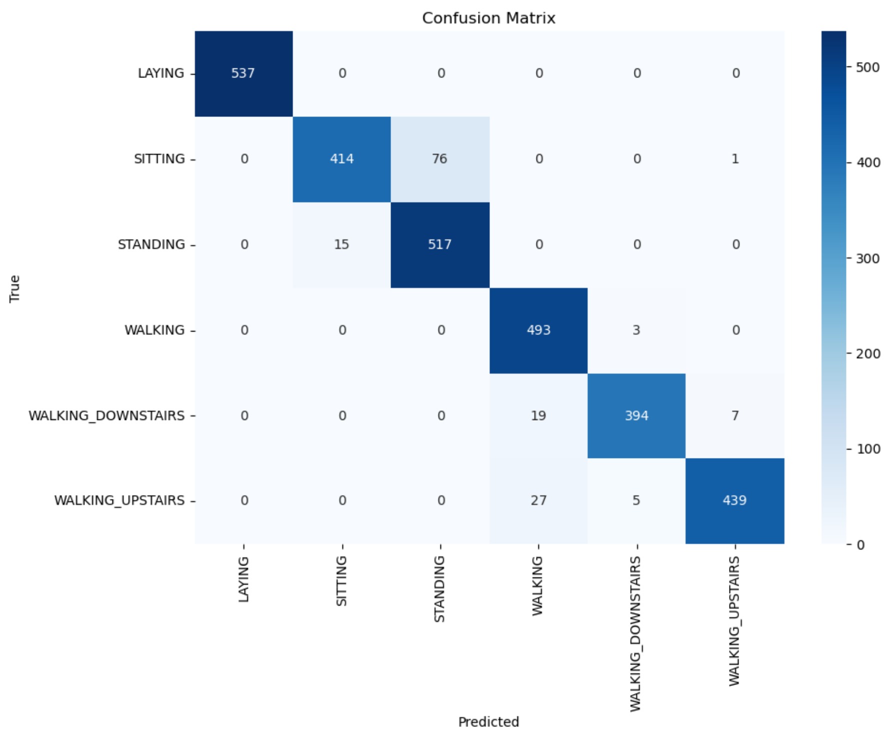
```
Accuracy: 0.9480827960637936
Recall: 0.9465146666115071
Precision: 0.9522887098163405
F1 score: 0.947805433463158
```

Neural Network

In contrast to other models Neural Network trained on unprocessed data.

```
      model = nn.Sequential(
    nn.Linear(input_dim, 64),
    nn.Linear(64, 128),
    nn.Linear(128, 32),
    nn.Linear(32, output_dim)
)
```





```
Accuracy: 0.9514760773668137
Recall: 0.951221182020706
Precision: 0.9539386045423924
F1 score:  0.9525779552852685
```

Results and Conclusion

Definitely we can see that best results are given by Neural Network (NN). However every model I implemented achieved over 90% accuracy. NN was definitely the most computational heavy, however the results I obtained was produced by a model, which trained on the unprocessed data. Where the rest of the models: kNN, SVM, Random Forest required preprocessing to achieve such a results.
I believe that the architecture of NN could be much better, however I believe that 95% accuracy is satisfying result.