**HAR FOR SMARTPHONES**

**Isak Sundström, Carolina Dexwik, David Östling**

**KTH, DM1590**

# 

# **1. Background and motivation**

The group’s main goal with this project was to do something challenging but doable. Something that the group could accomplish using the programming skills achieved during the course labs while also expanding on that knowledge. One of the main goals were to learn and understand multi-classification problems with a dataset. Another merit would be to use this opportunity to prepare the group for any upcoming courses on KTH which served as a major motivation. A perfect fit resulted in collecting the data through sensors and motion activity, solving the multi-classification goal whilst also preparing the group for the next periods’ sensor programming course.

Today, most people use a smartphone daily and carries it around most of the time. Most smartphones on the market have an accelerator sensor and a gyroscope built in. The goal of this project was to find out if it was possible to learn about the users physical activities based on the data gathered from the sensors.

# **2. The dataset used**

The dataset used was carried out with a group of 30 volunteers within the age of 19-48. The test was performed with each person doing six activities; walking, walking upstairs, walking downstairs, sitting, standing and laying down, whilst wearing a Samsung Galaxy S2 smartphone on the waist. Using the phone’s accelerator and gyroscope the dataset captured 3-axial linear acceleration as well as 3-axial angular velocity. The dataset was divided into two groups, 70% of the volunteers generated the training data whereas the remaining 30% created the test data. The dataset in question is called a: HAR (Human Activity Recognition) dataset, it was obtained from UCI and fits perfectly with the group’s planned task.[[1]](#footnote-0)

The time domain signals were captured at a constant rate of 50 Hertz. The time domain signals are denoted with the prefix *t*. To remove noise from the time domain signals, they were processed with a median filter and a third order low pass Butterworth filter of 20 Hertz. The acceleration signal was divided into body and gravity acceleration, denoted tBodyAcc and tGravityAcc, also using a low pass Butterworth filter of corner frequency 0.3 Hertz. The angular velocity and body linear acceleration were derived in time to get Jerk signals, denoted tBodyAccJerk and tGravityAccJerk. To obtain the magnitude of the three dimensional signals, they were calculated with Euclidean norm, denoted tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag and tBodyGyroJerkMag. To get the frequency domain of some of the signals, a fast fourier transform was applied, denoted with the prefix *f*.

# **3. The machine learning methods used**

The first part of the project included analyzing the dataset and to select which features to use in order to make a better prediction in the end. There were over 500 features in the original dataset so to shorten the list down and remove redundant features would most likely give a more precise result. The feature selection methods used in this project were the unsupervised machine learning methods PCA and Kbest.

A few different supervised machine learning methods were used for the prediction of activities in order to find the one that would make the best prediction. These were Naive Bayes, K Nearest Neighbor, and also Linear Discriminant Analysis together with a decision tree. Each classifier was used with the different sets of features selected from the unsupervised learning methods; PCA and Kbest, to be able to compare the results.

## **3.1 Supervised learning - Classifiers**

### **3.1.1 Naive Bayes**

This classifier is based on the Bayes theorem. It predicts probabilities for each class such as the probability that a given record or data point belongs to a particular class. As a result, the class with the highest probability takes the role as the most likely class. The Bayes theorem can be viewed below:

\textrm{P(D \textbar Pos) = }  \frac{\textrm{ P(Pos \textbar D) * P(D)}} {\textrm{P(Pos)}}

The pros of using Naive Bayes is that it is computationally fast and relatively simple to implement. Furthermore, it works very well with a high dimension which matches the groups task. However, there is one major drawback to using naive Bayes, which is that the classifier relies on independence assumption and will perform poorly if that assumption fails to be met.

### **3.1.2 KNN**

KNN or K Nearest Neighbor is a rather simple algorithm that starts out with storing all available cases and classifies the new data/case based on a similarity measure. Primarily it is used to classify a data point dependant on how its neighbors are classified. Moreover, this classifier is heavily based on a value called “k”. The process of figuring out what k should be used is called parameter tuning and is important for better accuracy.

### **3.1.3 LDA**

LDA, short for Linear Discriminant Analysis, is a supervised machine learning method with the main goal of reducing the dimensionality of a dataset while retaining most of the data. The features of the higher dimensions are projected onto a lower dimensional space and thereby removes dependent and redundant features. The algorithm includes; calculating the spreadability between the classes, calculating the spreadability within the classes, and then construct the space with lower dimensionality which minimizes the variance with each class and maximizes the variance between each class.

## **3.2 Unsupervised learning - Feature selection methods**

### **3.2.1 PCA**

PCA or principal component analysis is a form of linear dimensionality reduction that uses singular value decomposition of the given data to project it to a lower-dimensional space. The input data is not scaled but it is centered for each feature before the decomposition is applied.

### **3.2.2 Kbest**

Kbest or SelectKBest is a fairly simple form of feature selection. It picks out x number of features that are the best for predicting the target label. In order to do this it uses a scoring function (for example chi2) which is applied to a pair (X, y) to assign a score to each feature. The features with the lowest score will be almost independent of y while the features with the highest scores will be more related to y and thus will provide better information. SelectKBest returns the x number of features with the highest scores. It is used for decreasing dimensionality in order to avoid overfitting and improve accuracy.

# 

# 

# 

# **4. Results**

## **4.1 Data preparation**

The original dataset was already divided and split into a train and test set, but to have more control during the feature selection process and to be able to divide the dataset using the stratify option, the group decided to concatenate the sets and proceeded to split them later. This resulted in a dataframe with the dimensions (10299, 563).

### **4.1.1 Distribution of Labels**

### 

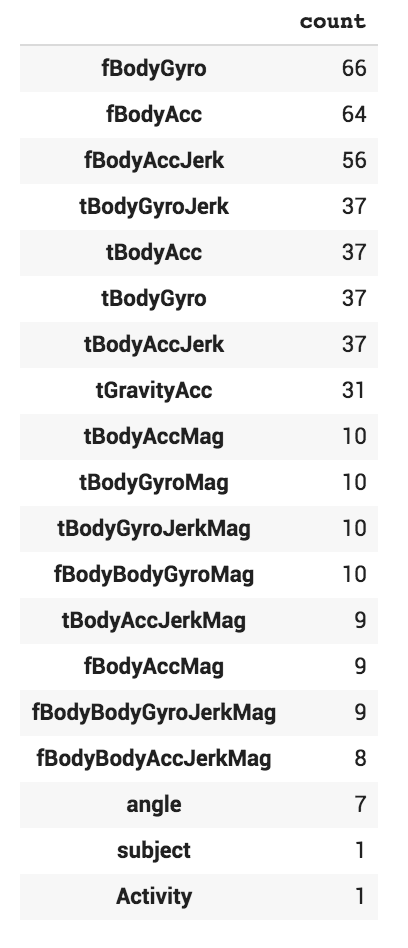
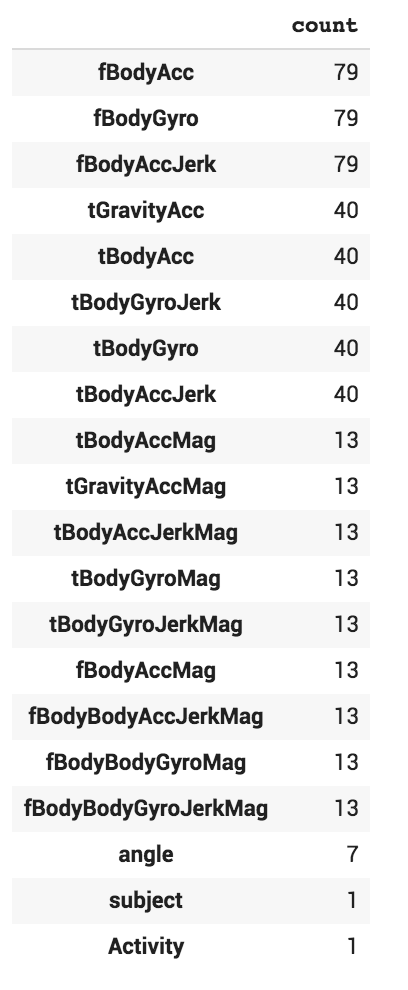
When plotting the distribution of the labels using the full dataframe, the plot showcases a distribution closer to uniform.

### **4.1.2 Correlation and discarding correlated features**

### 

*Before removing features After removing features*

When visualising the correlation matrix with Pearson Coefficients, the matrix is not of full rank and it turned out to have a lot of features that were collinear. The features that had a PC value > 0.99 were removed which resulted in 114 removed features.



### **4.1.3 Label Encoder**

The label encoder was used to encode the string variables of the labels to numbers.

{2: 'STANDING', 1: 'SITTING', 0: 'LAYING', 3: 'WALKING', 4: 'WALKING\_DOWNSTAIRS', 5: 'WALKING\_UPSTAIRS'}

### **4.1.4 Train test split**

The group decided to use the same percentage 70/30 split for training and test data as the original split was made. The stratify option was used to make sure the distribution over the labels would be uniform. Confusion matrices was used to evaluating the results measuring accuracy.

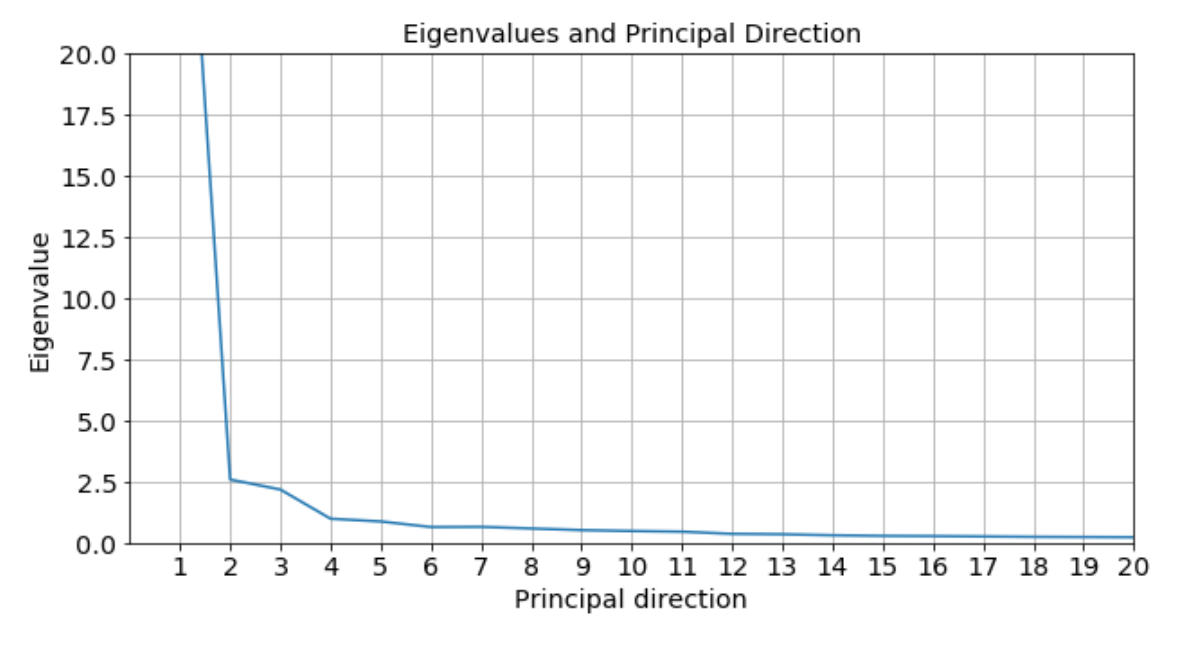
## **4.2 Feature selection Implementation**

In this project, feature selection was crucial because of the dimensions of the dataset. Mostly to reduce overfitting by removing redundant features so the decisions would get less affected by noise in the data. Also to improve accuracy when misleading data was removed.

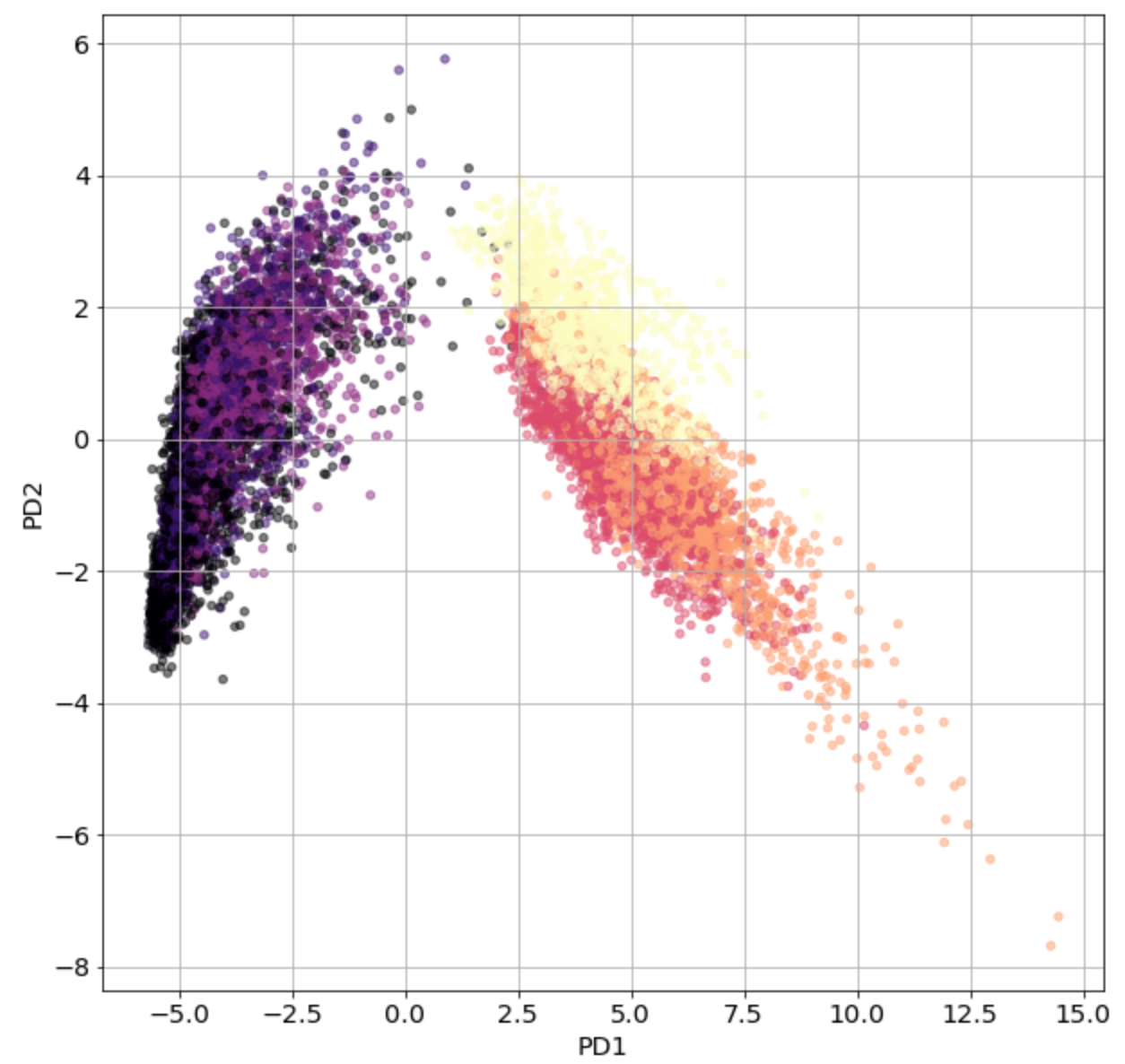
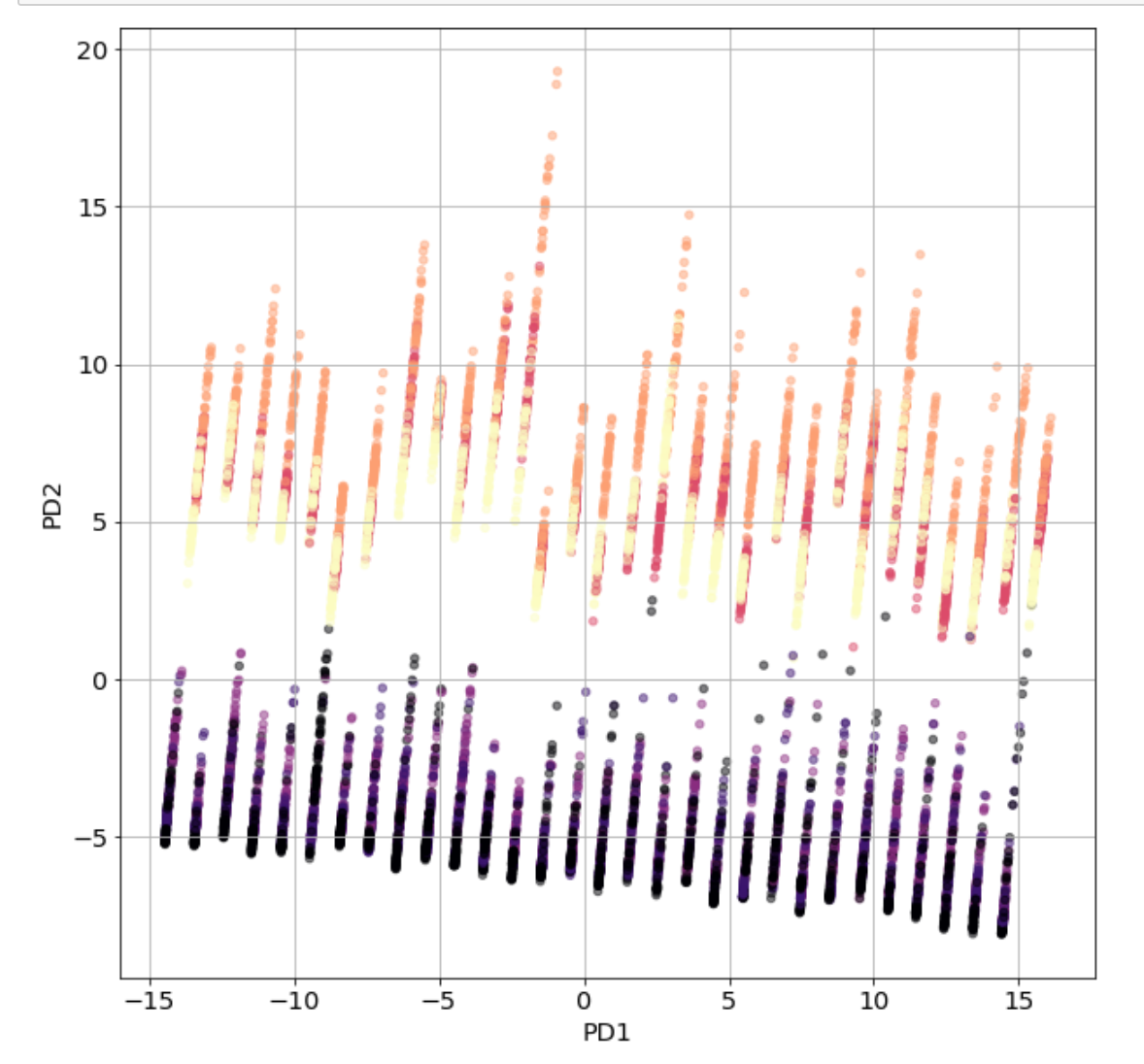
### **4.2.1 PCA**

The principal component analysis was implemented by doing the calculations of the algorithm step by step, rather than using the PCA from the decomposition library in Scikit learn.

The result of performing the eigenvector decomposition of the covariance matrix was partly showcased in the plot of the eigenvalues and principal directions. The group decided to use five features based on the provided graph.

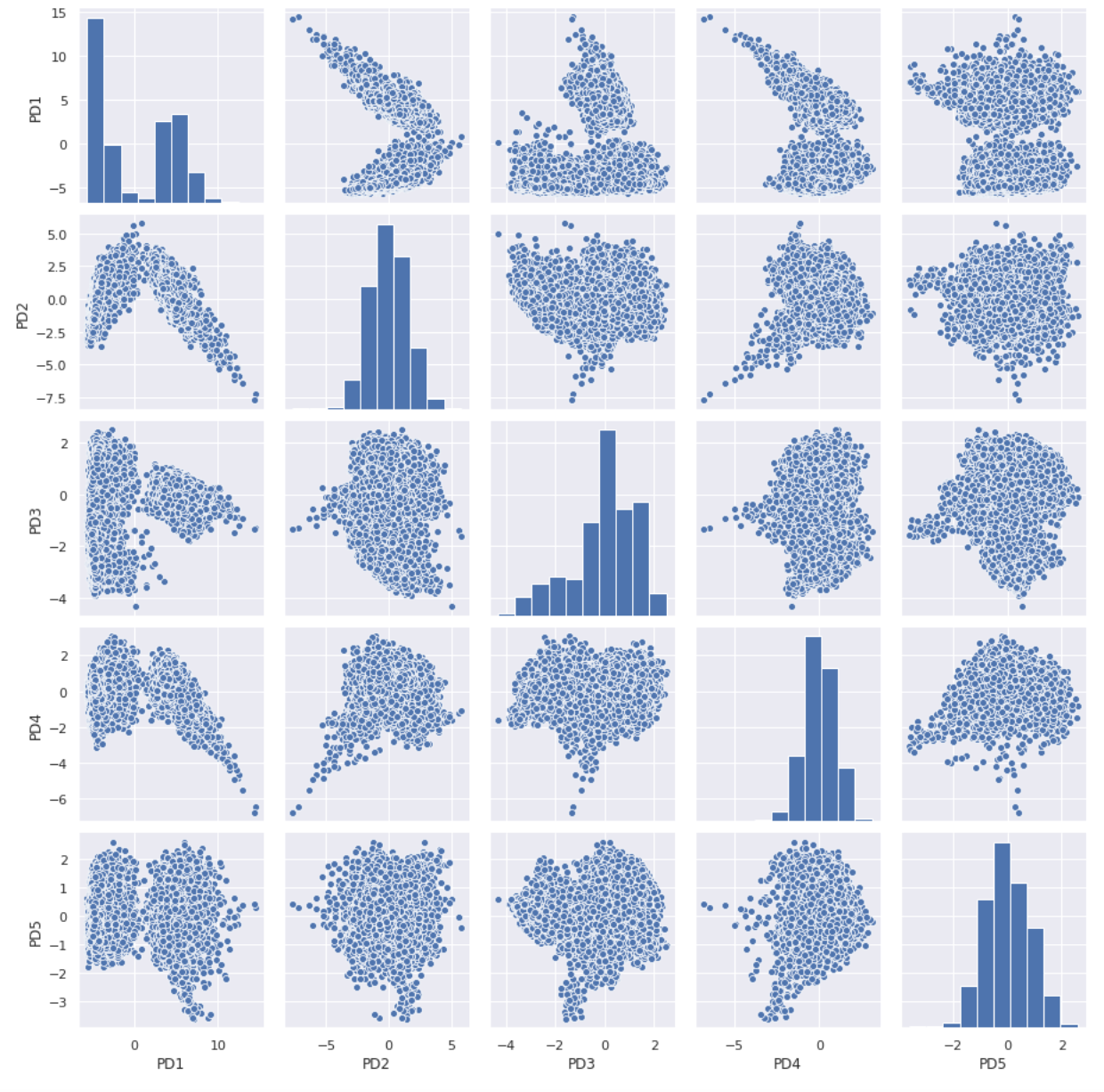


The feature selection is created by projecting the mean-centered observations onto the eigenvectors [Xproj]. Before removing the redundant features that were collinear the plot looked completely different compared to removing them.



*Before removing collinear features Result of projection*

Following plot shows the relationship between the selected features:

**

### **4.2.2 Kbest**

Feature selection with SelectKbest was implemented with the code made in the fourth lab in this course, with a few modifications to accommodate this particular dataset. Instead of using the *chi2* option in *SelectKBest* like in the lab, the *f\_classif* option was chosen because some of the features had negative values which is not compatible with *chi2*. 10 features were selected out of the original 563, significantly lowering the dimensionality. These features were:

['tGravityAcc-mean()-X' 'tGravityAcc-energy()-X'

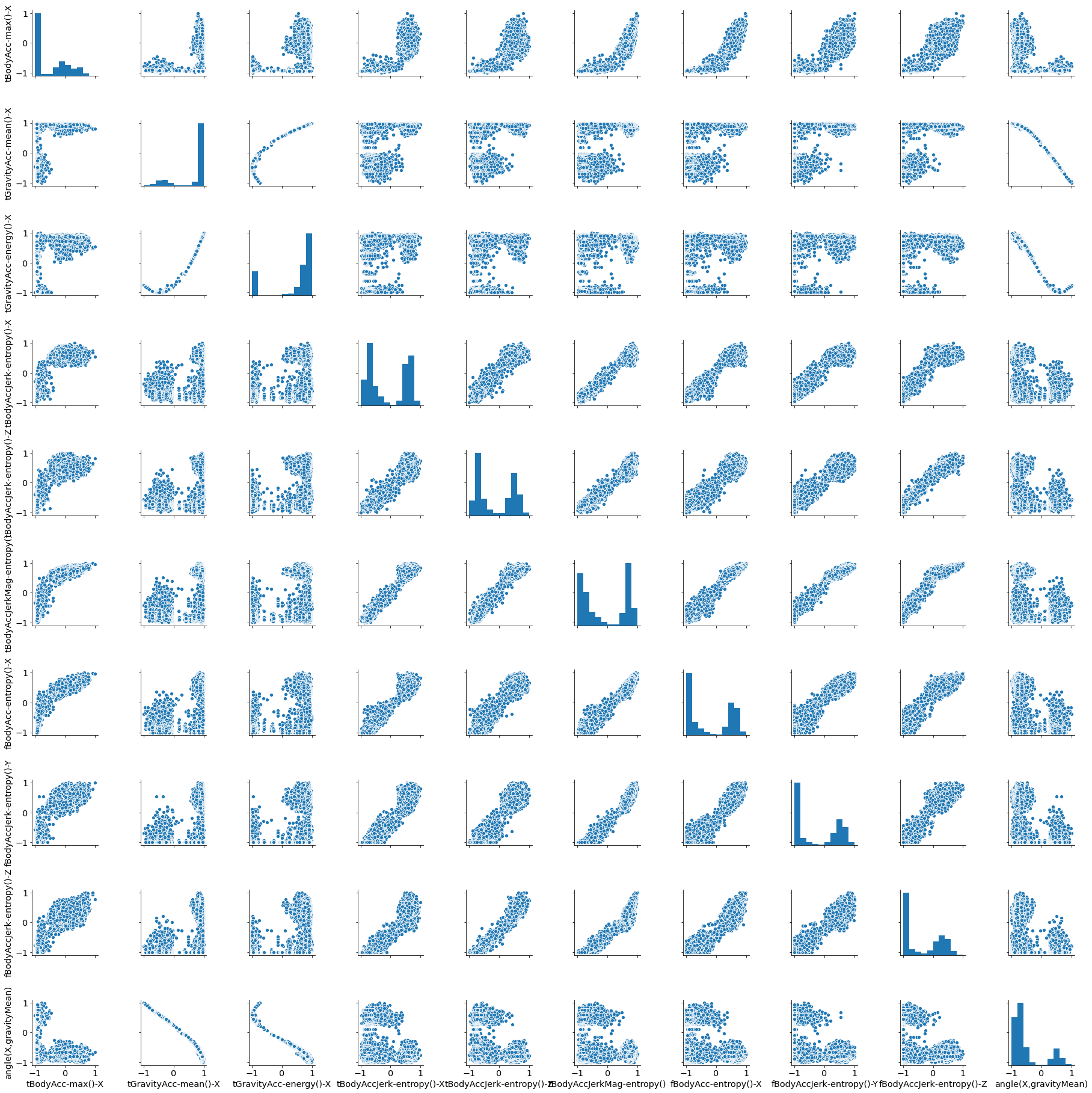
'fBodyAccJerk-entropy()-Y' 'tBodyAccJerkMag-entropy()'

'fBodyAcc-entropy()-X' 'tBodyAccJerk-entropy()-X' 'tBodyAcc-max()-X'

'angle(X,gravityMean)' 'tBodyAccJerk-entropy()-Z'

'fBodyAccJerk-entropy()-Z']

The following plot shows the relationship between the features (zoom in to read the text):



## **14.3 Predictions and Accuracy**

### **4.3.1 Naive Bayes**

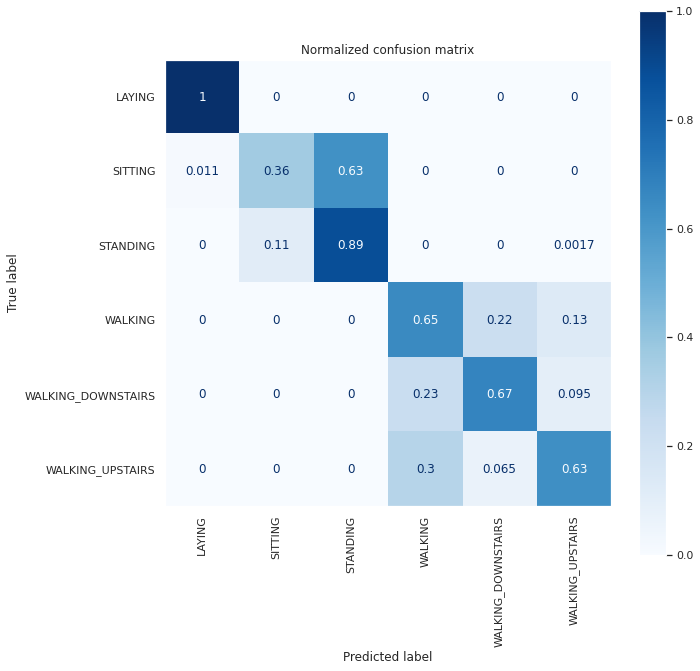
### **All Features**

The accuracy score received when using all features with the Naive Bayes classifier yielded an overall good result compared to the other feature selection methods. However it appears to have struggled when it found the *laying* accuracy. In correlation to this it got a far better *sitting* score compared to the other two methods. Perhaps a combination of the three feature selections using majority vote could have provided a better accuracy score with Naive Bayes.

### **PCA**

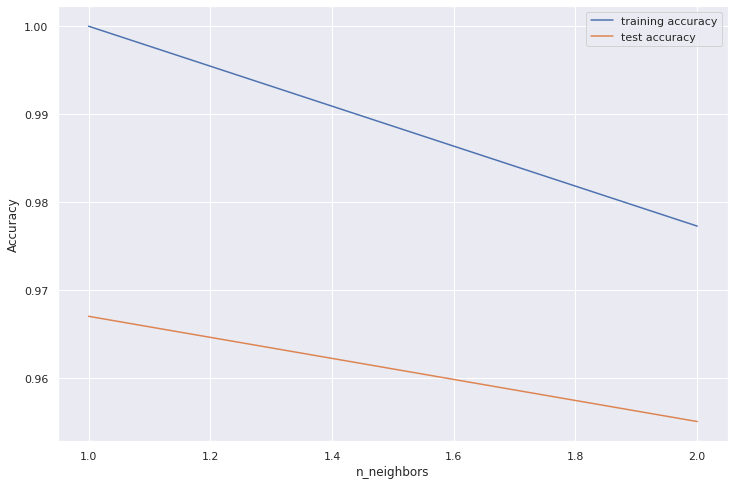
Using PCA we get an overall moderate accuracy score. It struggles a lot with determining the accuracy of *sitting* (though not as bad as with k best) as well as *walking downstairs* (worse score than k best). There is absolutely room for improvement here, luckily there are other classifiers and feature selections to resort to. The PCA method achieved the *second* best overall accuracy score while using Naive Bayes as classifier.

### **K Best**

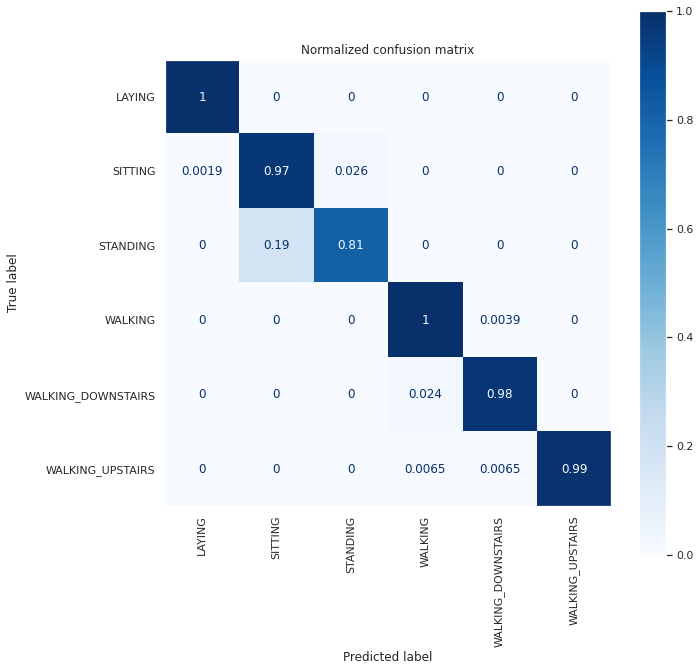
The overall accuracy for the k-best method is by far the worst out of all performed tests. However the main reason to this was the “*sitting accuracy*” of 36%. This could possibly be the result of having trouble to determine the difference between sitting and standing/laying. This is because sitting is a sort of middle ground of laying and standing. When a person sits the sensors could determine it as the person laying down/standing up instead, depending on the angle of the sitting (when the phone is stored in a pocket). Therefore an upgrade of the sensors used could probably benefit the accuracy of this method. However even if this would be improved upon this method would probably still struggle in many ways. Fortunately, there are other methods to achieve better accuracy, even with the current sensors.

### **4.3.2 KNN**

All Features

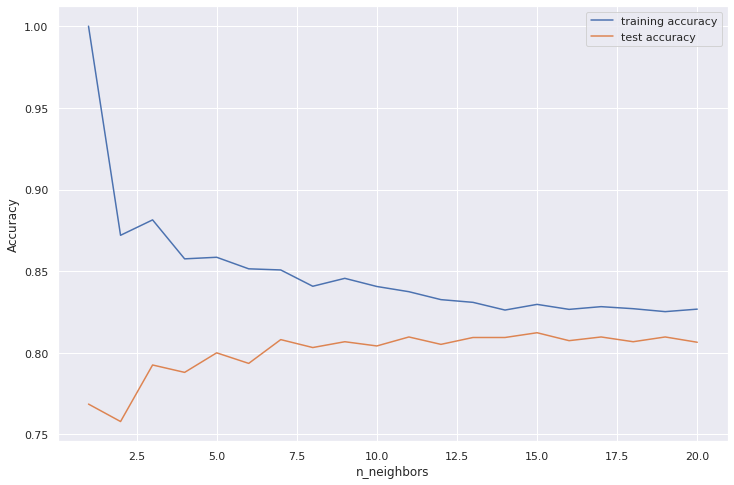


When using all of the features, k=1 seems to give the best accuracy for the test set.

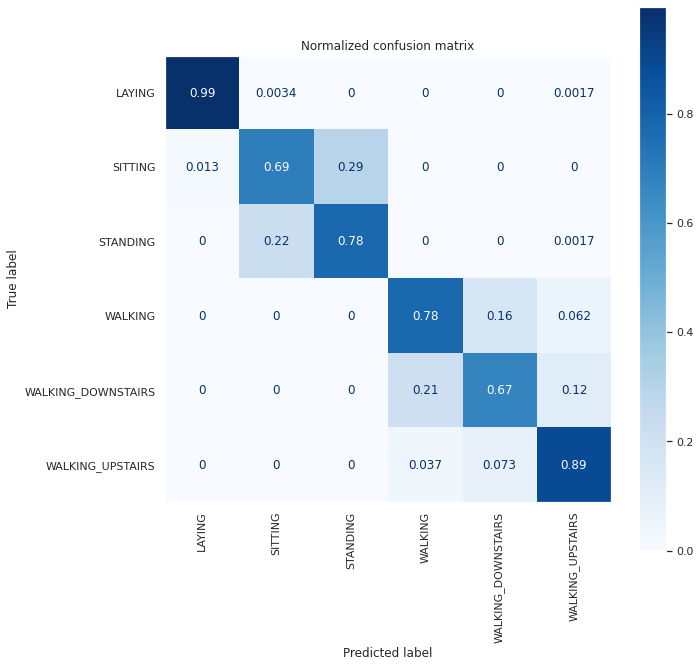


According to the confusion matrix, the accuracy of the test is very good when using all of the features. However, since k=1 was chosen and because of the high dimensionality, this could be the cause of overfitting. Nonetheless, because of the very high dimensionality, these results are probably not accurate. The graph showing test and train accuracy indicates that the test accuracy constantly declines when increasing the value of k, which makes it difficult to find a good value of k. Using feature selection and decreasing the dimensionality would give a much clearer result.

**PCA**

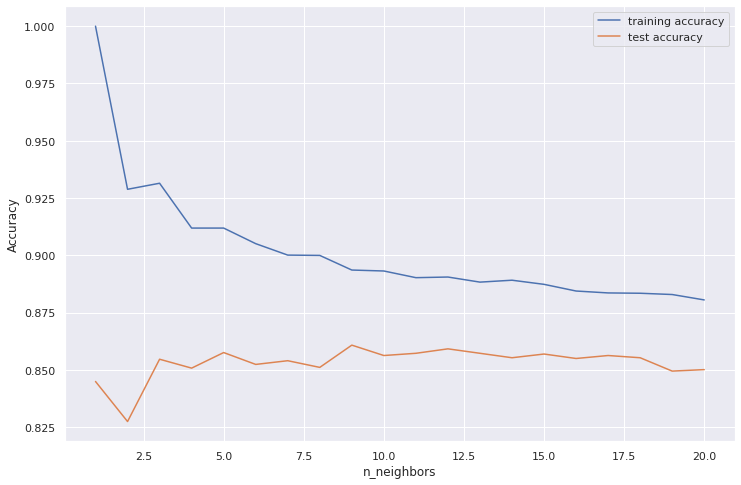


With the PCA features, maximum accuracy was achieved using *k=15*.

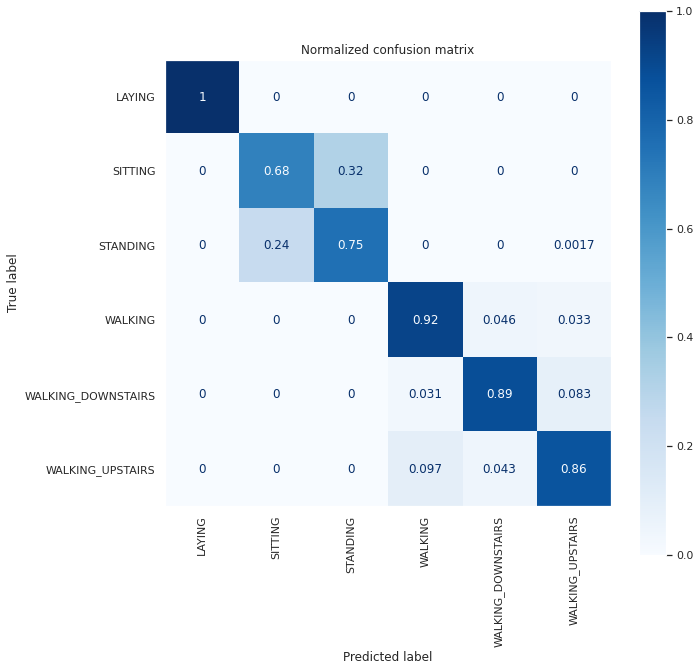


The accuracy when using the PCA features seemed to be somewhat worse than when using all features, although since the test with all features had a much too high dimensionality, this was probably a more accurate result.

**KBest**



With the Kbest features, maximum accuracy is achieved with k=9.



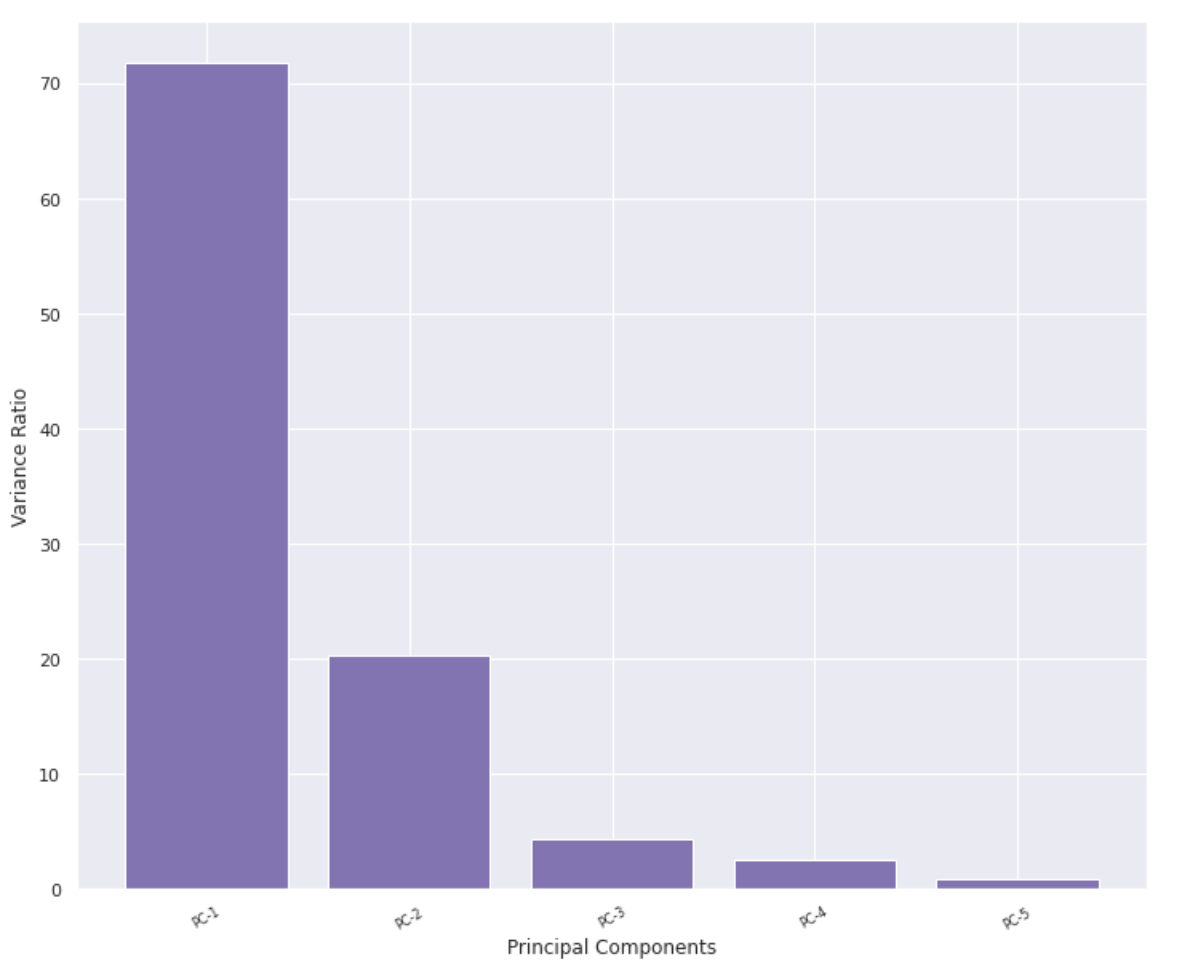
The Kbest features seem to give a slightly better result than the PCA features. Both seem to be having slight difficulties with differentiating sitting from standing, possibly because these activities are very similar to each other. Both involve being very still. There is also some lesser confusion between the three walking activities, which is also understandable considering that they are similar. Using the PCA features seems to give slightly better results with *sitting* and *standing* while the Kbest features are better at predicting the other four activities.

### 

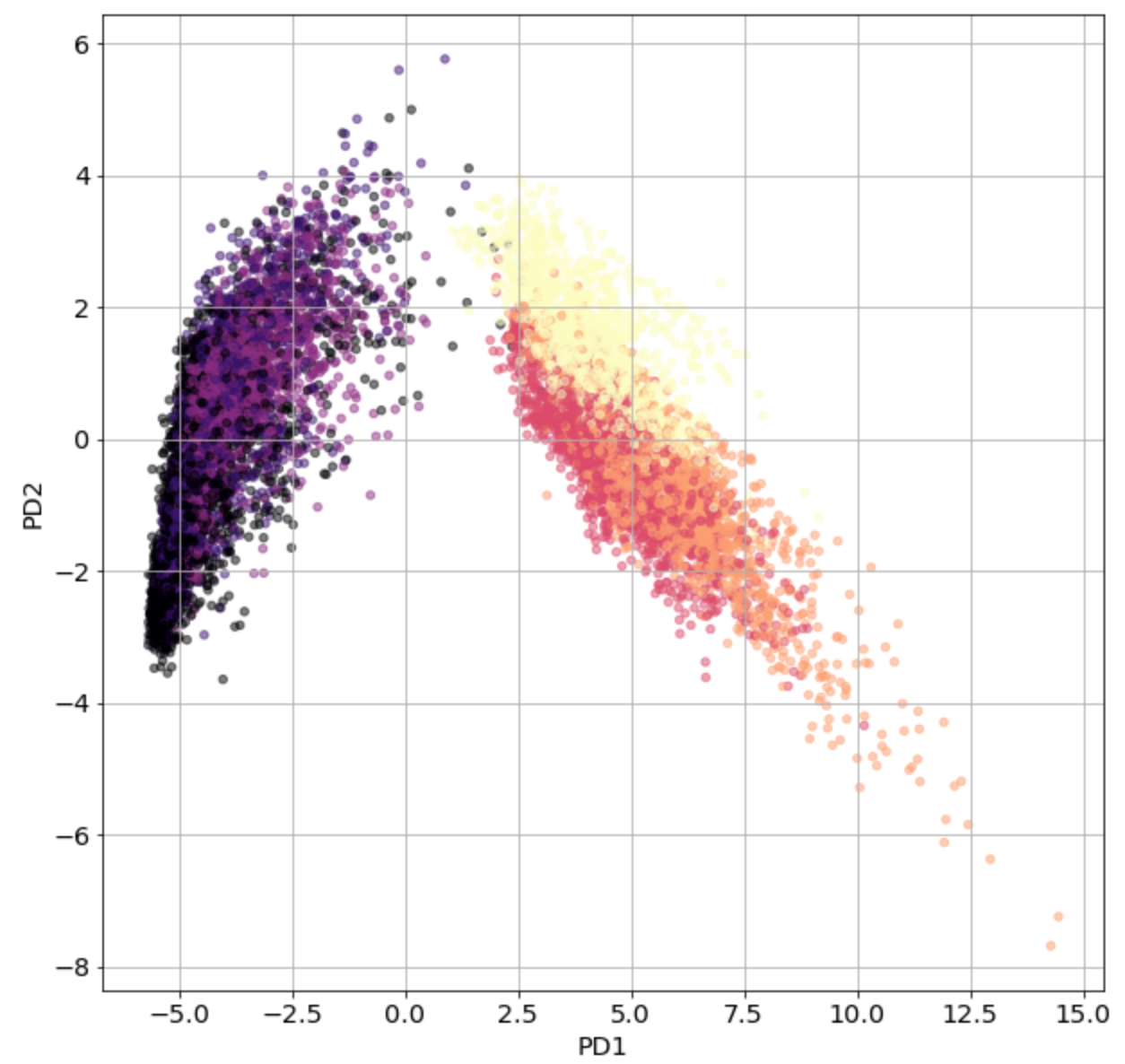
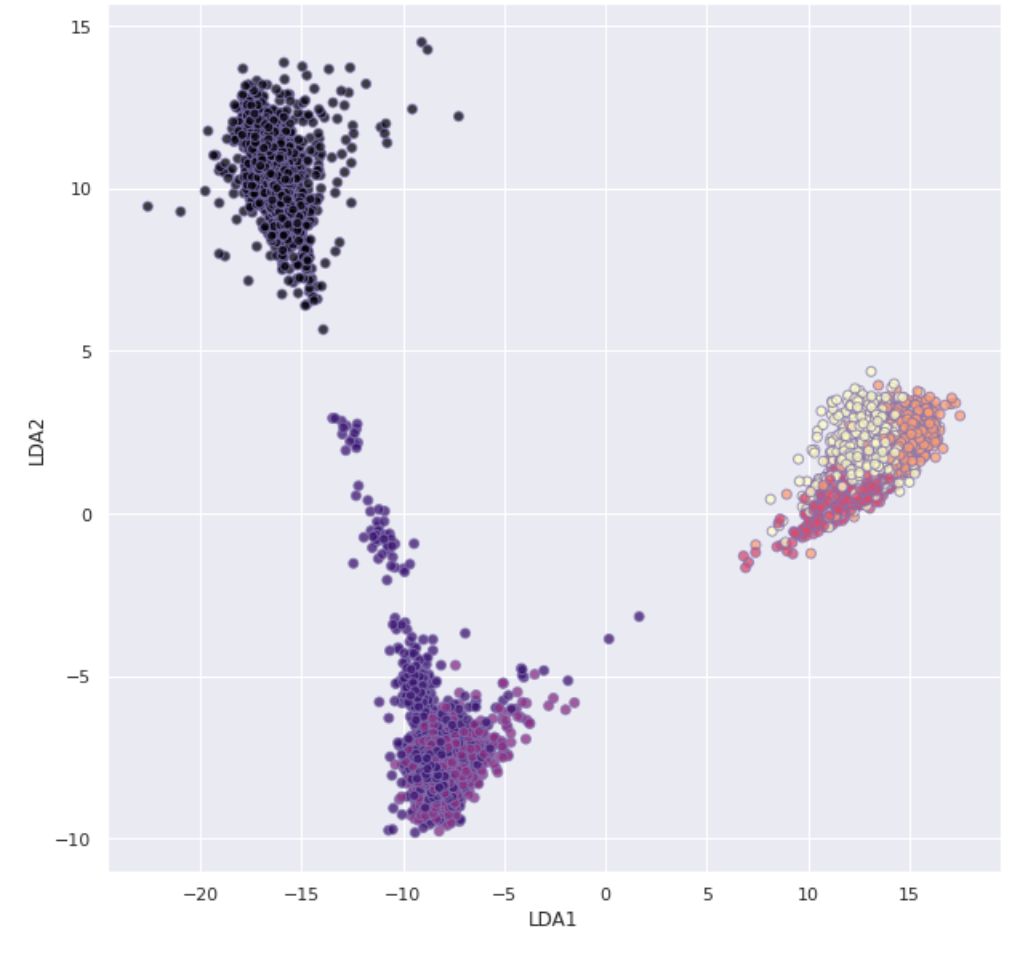
### **4.3.3 LDA**

The linear discriminant analysis was implemented with Sk-learns built in discriminant\_analysis library with 5 chosen features. According to the documentation the built-in LDA seems to apply a naive bayes classifier after the dimensionality reduction.

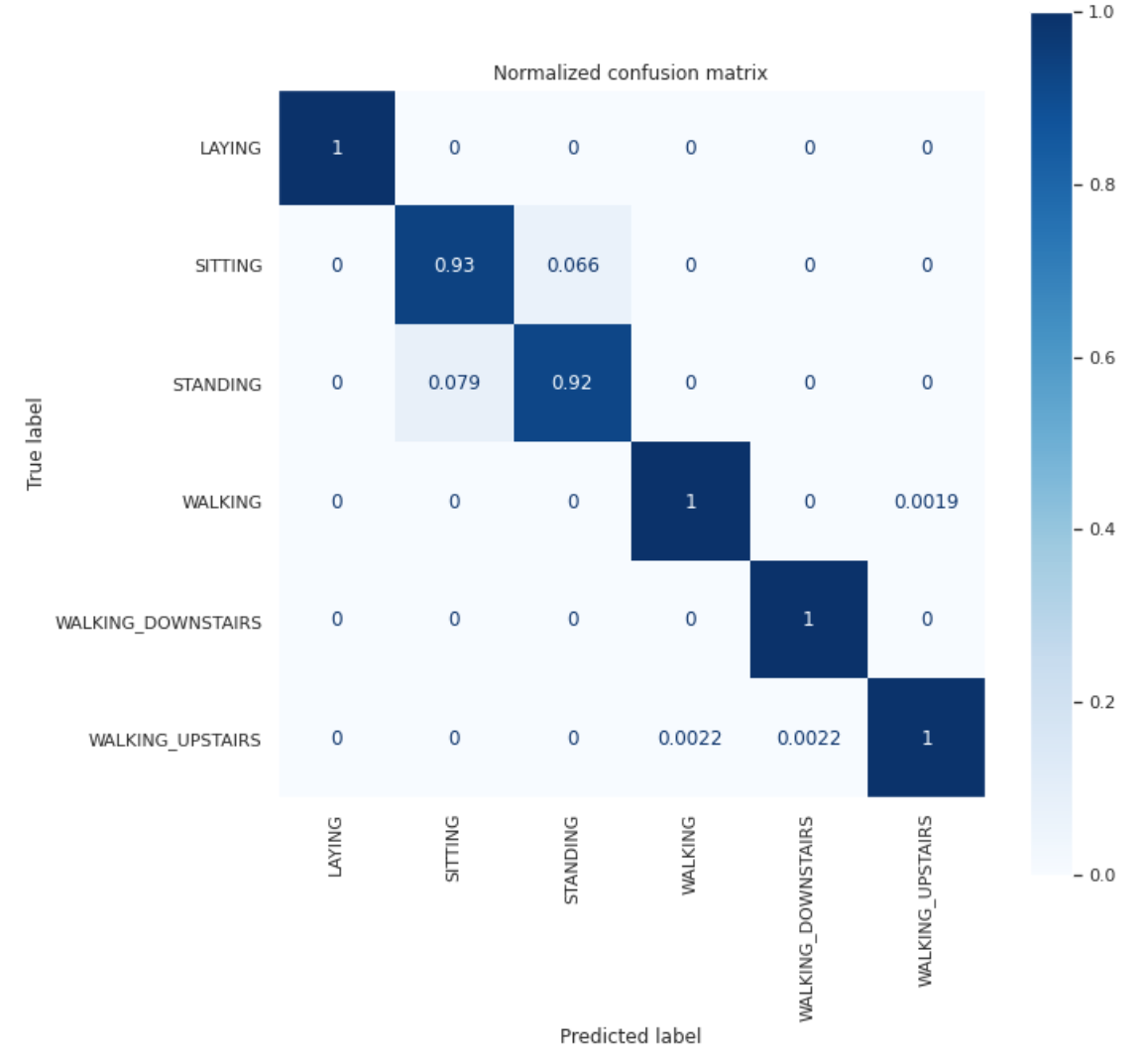
The following graph shows the variance ratio in the different principal components that are created during the linear discriminant analysis. When comparing LDA and PCA as dimensionality reduction methods: PCA intends to find the principal components that maximizes spreadability in the dataset and ignores class labels. LDA focuses on maximizing the separation between the classes by finding the linear discriminants that represents those axis that maximizes the variance.



The next graph is showcasing a plot of the data of the two first two LDA components, coloring each activity with a different color. It looks very different compared to the plot of the first two PCA components.



The result of using LDA performed the best for accuracy of the test data with really good results, with almost full accuracy on all activities.



# **5. Reflections and perspectives**

According to the results, especially from the LDA, one can very accurately predict which of the tested activities is being performed by simply observing the outputs from the accelerometer and gyroscope in a person’s smartphone. One possibly alarming consequence of this is that big tech companies already store a lot of our personal data, and according to these results they could theoretically also know what a person is doing at any given time (out of the activities tested).

To connect to the group’s goal of using a dataset that is connected to the upcoming sensor programming course, if wanting to implement machine learning in a project using an accelerator sensor and a gyroscope, the LDA algorithm would be the one the group would try to implement first. This could be used for example to build an application to detect the activity of the user and remind the user to move around after sitting too long. These sensors are also used in activity watches such as fitbit.

Overall the group members were pleased by the result of the project while some aspects absolutely could be improved upon. To begin with the project could have benefitted from doing more research for each algorithm used, sometimes it was hard to know what was correct or not. If more research had been done a lot of time could have been saved while also yielding improved accuracy results regarding the classifiers. To add upon the time issue, the group could have benefitted further from planning the coding better rather than resorting to the “*learn as you go*” method.

# **6. Project distribution**

The overall structure of the project was set up together by the group whereas the classifiers, kbest and PCA were split amongst the members. The PCA was written by Carolina while the K-best feature selection was written by Isak, both codes were inspired by lecture notes. Furthermore the classifiers were written by David and Carolina. These classifiers were inspired by lecture notes (primarily *KNN* and *Naive Bayes*) combined with web information (primarily used for the LDA[[2]](#footnote-1) but also to build upon the code inspired by the lectures). Carolina also did most of the data preparation including removing the correlated features and took responsibility for making the plots look neat and clear.

Regarding the report, the group worked together by writing about what each member wrote in the code. However, the group had to extend their knowledge by evenly splitting the remaining sections of the report.

1. <https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones> [↑](#footnote-ref-0)
2. <https://www.apsl.net/blog/2017/07/18/using-linear-discriminant-analysis-lda-data-explore-step-step/> [↑](#footnote-ref-1)