Student ID: s3801950
Student Name: Oyvind Samuelsen

Classifying and predicting activity data with K-nearest neighbors and decision tree

Oyvind Samuelsen
RMIT, School of Science
S3801950@student.rmit.edu.au
June 9. 2020

The solution I reach in this project is that K-nearest neighbors is a superior method to classify activity data, compared to decision tree. To get a good result it is essential to extract features from the frequency data.

Table of Contents

Abstract

The aim of this project is to find which of the classifiers ‘K-nearest neighbor’ or ‘Decision tree’ models activity data with the best accuracy. Data from a dataset of acceleration measurements of 7 activities was used. Features from a time-window of 1 second (52 Hz) were extracted. In the end, K-nearest neighbors was found to give the best accuracy with 84% correct classifications out of a test size of 20% of the available data, compared to decision tree with 67%. Further recommendations include looking into more and better features to extract from this frequency data, and to research using various window lengths.

1 Introduction

When we perform different activities, we have different movement patterns. Devices such as fitness trackers and smart watches can monitor your every movement and give you a report at the end of the day of all the different activities you did, without your interaction. To be able to recognize what activity you are currently doing, the activity tracker can take a sample of movement data and compare it to already known movement patterns for different activities. In other words, it can take your activity data and use an activity pattern model to predict what activity you are preforming. In this report I will build and analyze two such models, namely K-nearest neighbor, and Decision tree.

2 Methodology

The 'Activity Recognition from Single Chest-Mounted Accelerometer' data set was used. It includes activity data from 15 persons and 7 activities. The data was logged with an uncalibrated accelerometer motion sensor while the subjects were preforming the activities. The activities are 'working at computer', 'standing up, walking and going up/down stairs' (renamed 'moving'), 'standing', 'walking', 'going up/down stairs', 'walking and talking with someone', and 'talking while standing'.

2.1 Data loading and cleaning

The CSV data was loaded with pandas, and introductory checks were made. Each row has an id, x-, y-, and z-acceleration value, and activity label. All accelerations have an 'int64' type. The columns do not have a header, so this is set manually. When checking the tail of the data, it became clear that the provided ids were given in a scientific format with not enough precision, so all ids over 100 000 were being rounded off, thereby unusable. This is solved by removing the given id and using the id pandas automatically assigns. There also existed some rows with label '0', which is not documented and is in error. This is solved by removing all rows with label 0. All rows were checked that they do not include any missing values.

2.1.1 Outliers

Accelerometers values are checked for max and min values, to find if there are any outliers/noise. I initially found data points which include y and/or z values which are very low or very high, sometimes at the same time, but not always. Common for all is that it is for the 'Computer' activity. This is unexpected, as one would expect that sitting at the computer would not give a high acceleration in any direction. Examining the data a bit further it is revealed that the abnormal values only happened for certain people, namely person 2, 6, 8, and 10. By plotting the measured accelerations along a time axis (line chart) we see that the discrepancies usually happened in the beginning and/or the end of the measurement. A likely reason could simply be that the person sits down in his or her chair, and this creates a great change in acceleration. This data is an important part of the activity data, and therefore not removed or altered.

2.2 Data exploration

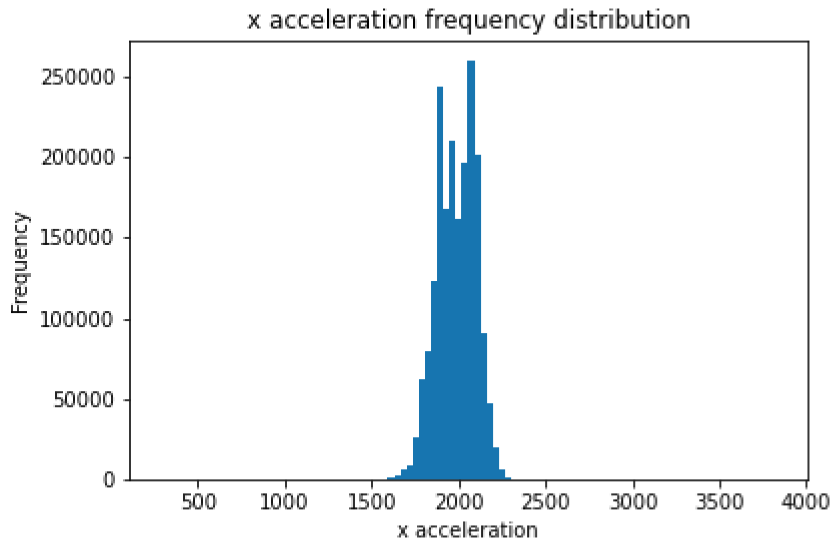
Data has been explored in several different ways; scatter matrix to compare all columns, histograms to look at the distribution of x-, y-, and z-accelerations, boxplots to look at distribution of each acceleration compared to classes, boxplot to look at distribution of x-, y-, and z-accelerations for each class, scatter plot to look at accelerations compared to both each other and class, and a 3d scatter plot to look at all 3 accelerations and 7 classes compared to each other. What is evident is that when the time aspect is not included in the summarization, a lot of information is lost. A good example here is for 'Computer' activity. When comparing activities with box plots, it seems like ‘Computer’ is one of the activities with the most movement, as mentioned in the ‘Outliers’ section above. From intuition this is not the case at all, sitting at a computer should not produce much accelerations in any direction. Looking at the box plots and scatter plots it seems like many of the activities are very similar, with similar mean and standard deviation. However, when plotting with line chart the differences are obvious. All this tell me that using the raw acceleration data, without considering the time factor, would probably not yield good results for the modeling part.

2.2.1 Single column exploration

The different accelerations were explored first by looking at the plot from the scatter matrix, along with calculations of min, max, mean, and std. By generating a frequency distribution with variable number of bins I could see exactly in what range most values were found, and this could also be plotted in a histogram to get a better overview of the distribution. The histograms are shown below. From the histograms and frequency distribution we can observe the most common values, and what values are more uncommon. This could help to spot dominating outliers if it existed. Line charts are also useful when exploring acceleration values, plots of single accelerations are in the notebook, and interpretation of line charts are in the section ‘Relationships between pair of columns’.

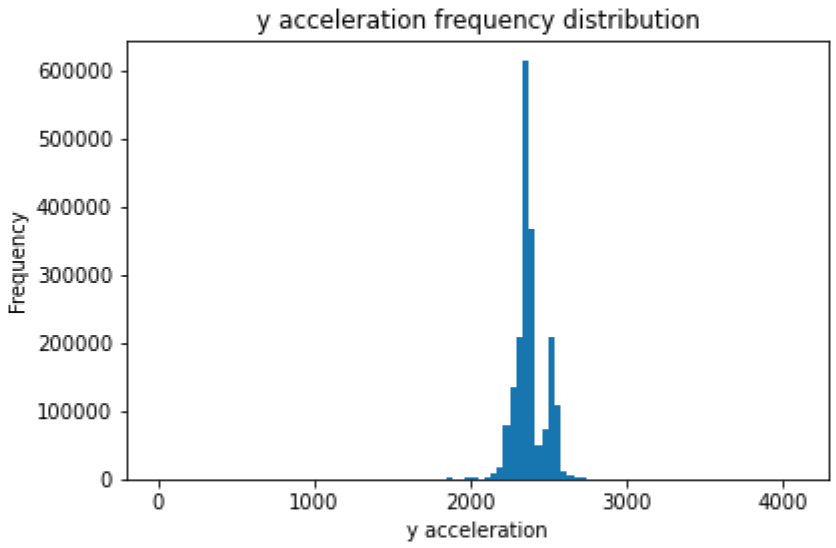
As mention in the source of the dataset, the activity sensors are uncalibrated, and the details of how the sensor measure are not specified. From the data it seems the point of reference (the rest position) is around 2000 if calibrated correctly. This means that if the sensor is calibrated correctly, and is laying perfectly still, all axis accelerations would show 2000. Then, if the sensor is accelerated upwards, y-acceleration would increase (above 2000). The minimum acceleration recorded is 0, and maximum 4095.

2.2.1.1 X-acceleration



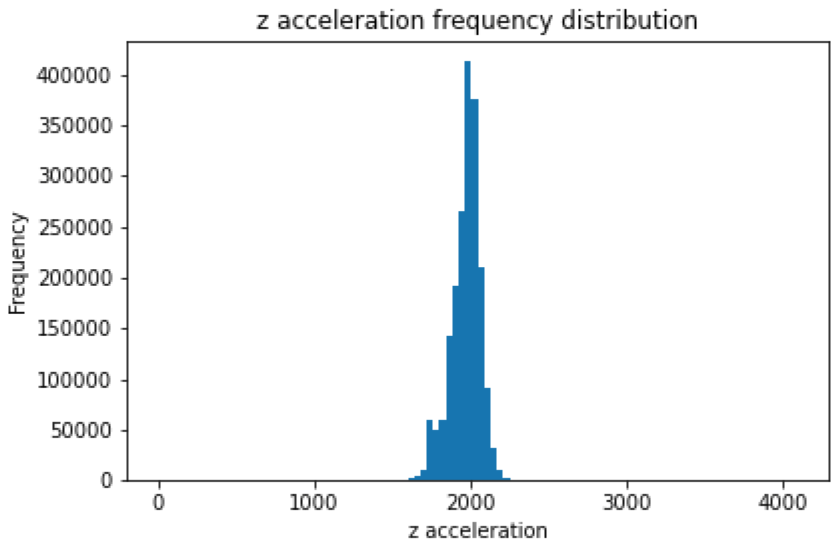
From the figure we can see the distribution of x-accelerations. The mean is around 2000, and very few values are below 1500 or above 2500. When exploring this column, I first printed a sample of the rows, then printed some basic statistics such as max, min, mean, std, and sorted the column and printed the head and tail of the sorted column. To get a better view of the distribution of the x-accelerations, I used this histogram and a box plot.

2.2.1.2 Y-acceleration



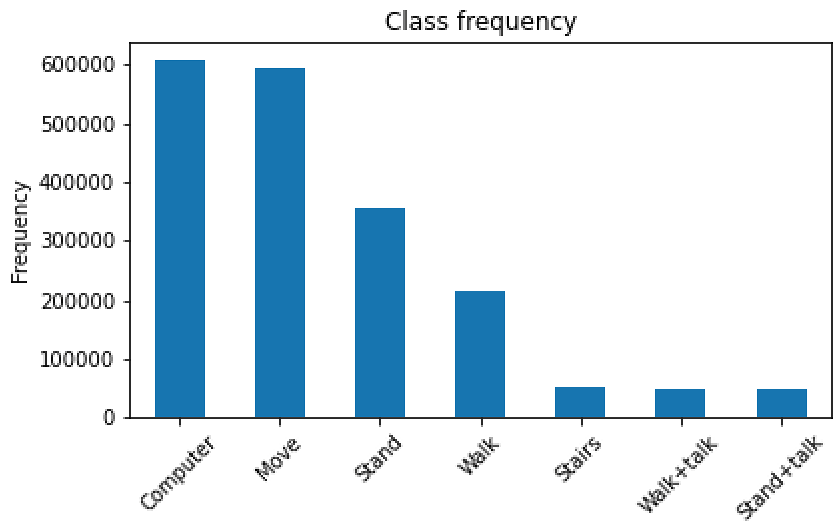
From the figure we can see most of the y-acceleration values are around 2300, with few below 2000 and above 2700. I explored this column the same way as for the x-acceleration, however when exploring this column, I found some unexpected high and low values at the head and tail of the sorted column. By making a chart like the one above, I found what I thought might be outliers or noise, which are mentioned in the 'Outliers' section above.

2.2.1.3 Z-acceleration



From the figure we can see most of the values are around 2000, with few below 1600 and above 2200. Similar outliers as for y-acceleration were found.

2.2.1.4 Activity classes



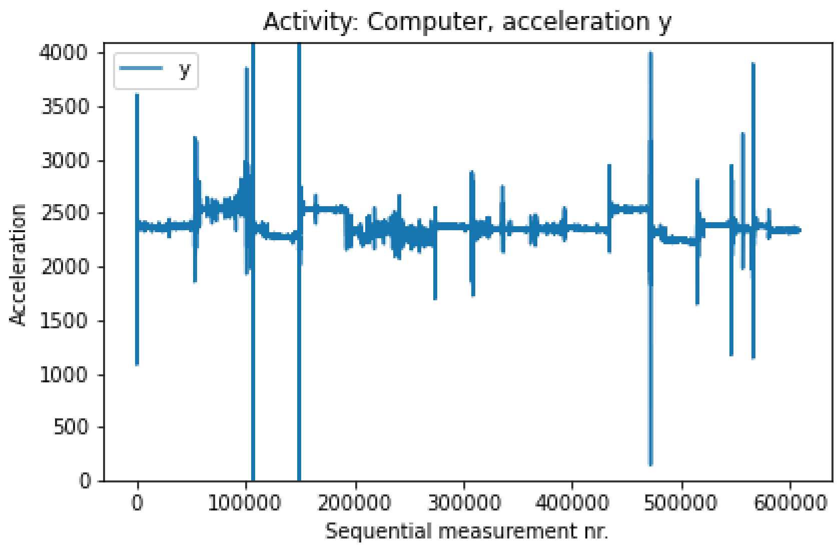
The bar chart shows how many measurements there are of each activity. We can see that there are a lot more measurements of 'Computer' than for example 'Standing + talking', telling us that the data is unbalanced. When exploring this column, I first looked at the value counts for the different activities, then plotted it in the chart above to get a better comparison.

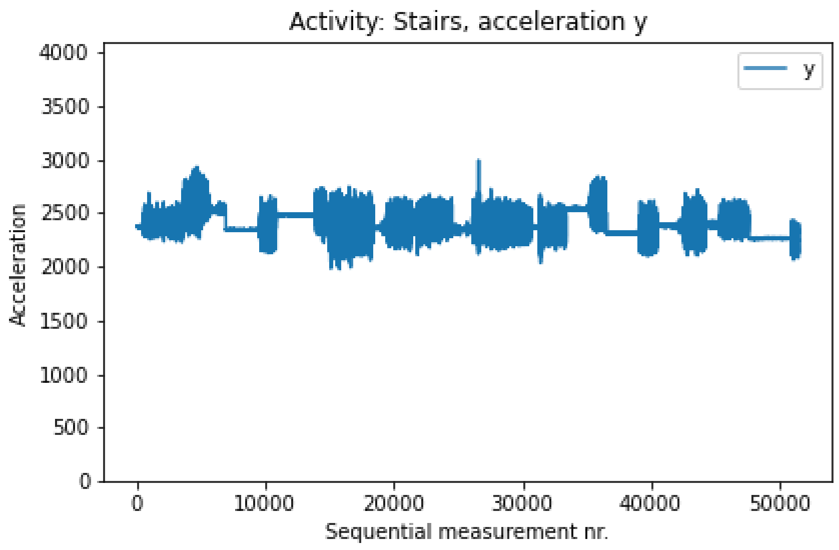
2.2.2 Relationships between pairs of columns

There are 4 columns to compare, x, y, z, and label. Each label (class of activity) has their own corresponding measurements of x-, y-, and z-acceleration.

2.2.2.1 Y-acceleration vs. activities

My hypothesis for y-axis acceleration is that I expect to see different measurement patterns for different activities. More specifically I expect the activities where the participant walks in stairs will have the biggest variability, since it is the activity where you move the most up and down, and the activities where the person is standing or sitting still will have the least y-acceleration variability.





From the first graph we can see the outliers mentioned above. Beside the outliers, the y-acceleration for 'Computer' is most of the time not changing very much. For 'Stairs' this is opposite, it is consistently changing, and one can clearly see when the person walked in the stairs. From the graph we then know that the variability over a short period of time will on average be higher for 'Stairs', however since it is cyclic data (it moves up and down around the rest point) the mean might be the same for both activities. This is important information we can use for feature extraction. The rest of the graphs can be found in the notebook. Comparing all, we can conclude that the hypothesis is valid.

2.2.2.2 X-acceleration vs. activities

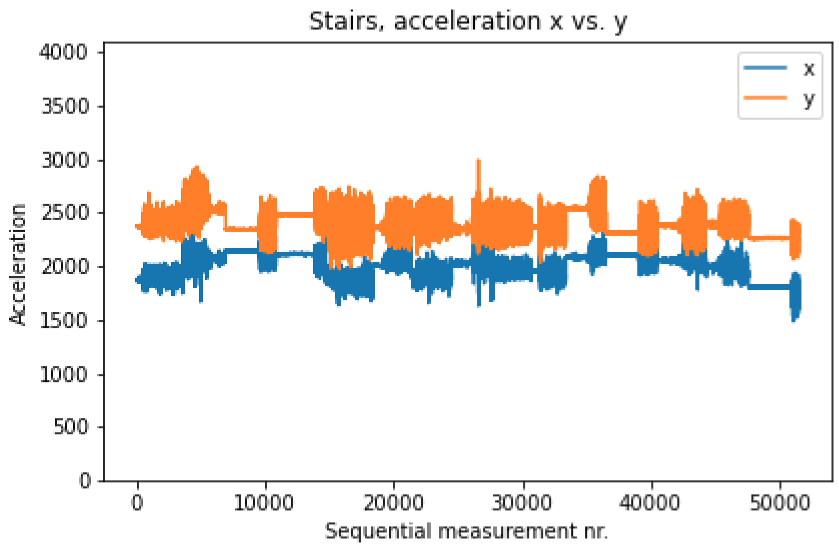
Like above, my hypothesis is that activities where the participants move will have the most variance. From the graphs in the Jupyter notebook we can see that this is valid as well. 'Walking' had the biggest consistent variability in x-acceleration, and in 'Walking + talking' we can see that they move, but less frequently.

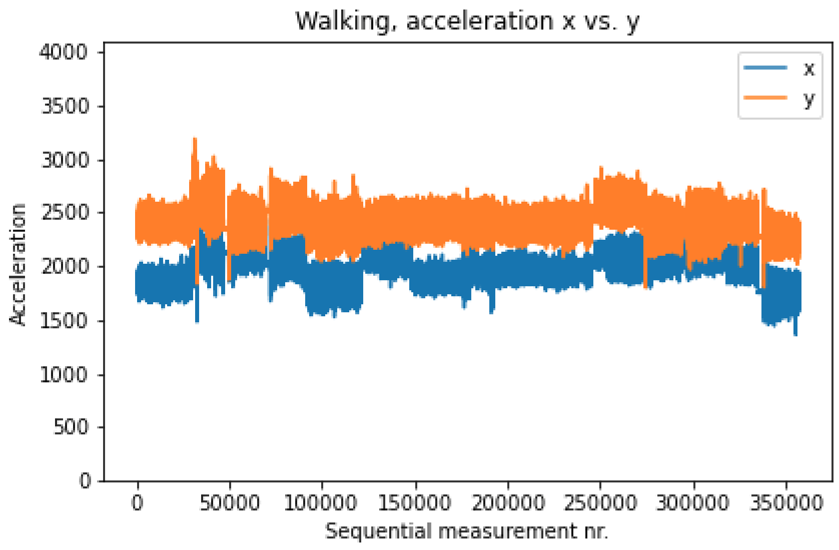
2.2.2.3 Z-axis acceleration vs. activities

Like above, my hypothesis is that activities where the participants move will have the most variance. From the graphs in Jupyter notebook we reach the same conclusion as above.

2.2.2.4 X- vs y-acceleration

My hypothesis is that we can see a correlation for the moving activities, e.g. when walking up a stair both the x- and y-acceleration would be sometimes higher and lower, however y will be higher since we in general accelerate more up and down than forward and backwards.

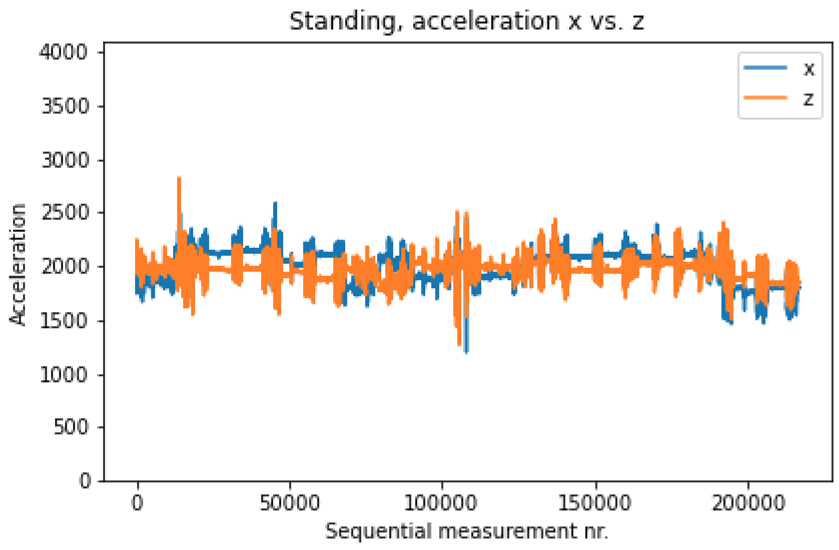




From the graphs we can see that y-acceleration is indeed higher in 'Stairs' and 'Walking' compared to x-acceleration, and this pattern is true for the rest of the graphs with moving activities in the Jupyter notebook. The hypothesis is valid. However, in activities where the participants do not move much the y- and x-acceleration is similar. We also see the effect of the uncalibrated sensor, we would expect the accelerations to rest at an equal level, however they do not.

2.2.2.5 X- vs. z-acceleration

My hypothesis is that since we normally do not move too much in z-direction, I expect x-acceleration to generally variate more than z-acceleration.



From the graphs we can see that z-acceleration generally follows the other acceleration, with small differences. One activity is shown above. From the graph it is not clear if it changes more of less than x-acceleration, but from the computed values we know that x-acceleration standard deviation (for all activities) is 111 and z-std is 94, so the hypothesis is true. The lower z-std could suggest that z-std is a less discrimination feature in the modeling stage.

2.2.2.6 Y- vs. z-acceleration

Same as above, I expect y-acceleration to variate more than z-acceleration. From the graph in the notebook and the computed values we reach the same conclusion as above, the hypothesis is valid. Y-std is 110.

2.2.3 Feature extraction

As per my findings in the data exploration stage, the raw acceleration data is not suited as features for modeling. We know the activity data is sequentially recorded at a frequency of 52Hz. A simple way to summarize the time data is to move a 'sliding window' over the data points, summarizing the data inside the window. A set of 520 data points (for the same activity) describes 10 seconds of movement. By having a window size of 52 data points, we can transform the 520 data points into 10 new data points, extracting the features we want. The features I extracted are mean, standard deviation, and the difference between max and min measurement. In the end we get 3 * 3 = 9 new features.

2.2.4 Selecting features

We now have 9 features; the next step is to see which features differentiates the different activities and will therefore be used in the modeling. There is no need to include a feature in the modeling if it does not improve the model score. To do this I use simple hill climbing algorithm, which will select features one-by-one and see if it improves the model score. By also using different parameters for the models I can see how it scores with the optimal features selected.

2.2.5 Classification models

I have used K-nearest neighbors (KNN) and decision tree (DT) classification models. These models operate differently, KNN looks at the neighboring datapoints to classify, while DT operates on a binary tree where it splits classes based on the most differentiating features. Since they work differently, I used hill climbing on both to select the features which best suited for each model.

2.2.5.1 K-nearest neighbors

By using hill climbing the selected features were: x-mean, y-mean, z-mean, x-std, y-std, and z-std. In other words, all max-min differences were removed. The steps to select features was to iteratively run hill climbing with iteratively higher number of neighbors, starting at k=1. The hyperparameters 'weights and 'p' were also changed to see what gave the best score. For each k, hill climbing would print the best score it could get, and which features it used to get this score. Then I could see where the score 'flattened' out, which happened to be when k=3. To be sure of the final features to select, I then ran hill climbing several times with k=3 and looked at which features it selected each time. In the end, the best result was with 3 neighbors, along with hyperparameters 'weights=distance' and 'p=2'. Having 'distance' as weight parameter means the model will give distance score by the inverse of the distance, closer points getting higher score. p=2 means it is using Euclidean distance. Compared to having p=1, a difference in one dimension will play a bigger role for the end distance score. The parameters are further verified by using k-fold cross validation. Several iterations of hill climbing and k-folds cross validation was used to confirm that these settings gave the best result.

2.2.5.2 Decision tree

For decision tree the selected features: x-mean, y-mean, z-mean, x-std, x-diff, and y-diff. Y-std, z-std, and z-diff were removed. The steps to select features was similar to K-nearest neighbor, only now I iterated over the hyperparameters 'depth' and changed 'min_samples_split' to see what gave the best result. A depth of 5 seemed to be a good balance, more could risk the model being overfitted. By setting the hyperparameter 'min_samples_split' to 40 we can further reduce possibility of overfitting without impacting the test result. By having a depth of 5, the maximum amount of leaf nodes will be 2^5 = 32. By setting minimum samples split, we control that all inner nodes will have at least this amount of class instances. Several iterations of hill climbing and k-folds cross validation was used to confirm that these settings gave the best result.

3 Results

K-nearest neighbor

Class	Precision	Recall	F1-score	Support
Computer	0.94	0.94	0.94	4633
Move	0.58	0.41	0.48	400
Stand	0.69	0.62	0.65	1689
Walk	0.82	0.9	0.85	2678
Stairs	0.53	0.48	0.5	391
Walk+talk	0.55	0.41	0.47	381
Stand+talk	0.86	0.89	0.88	4632

Accuracy: 84%

Decision tree

Class	Precision	Recall	F1-score	Support
Computer	0.73	0.8	0.76	4633
Move	0.53	0.05	0.1	400
Stand	0.42	0.13	0.2	1689
Walk	0.75	0.87	0.81	2678
Stairs	0.56	0.07	0.13	391
Walk+talk	0	0	0	381
Stand+talk	0.6	0.77	0.67	4632

Accuracy: 67%

The results from K-nearest neighbor and decision tree classifiers can be seen in the tables above. K-nearest neighbor had an accuracy of 84%, while the decision tree had 67%. A graph of the decision tree can be found in the Jupyter notebook.

4 Discussion

K-nearest neighbor had the best accuracy for classifying activity acceleration data. The reason for this is likely that all the data were continuous, and decision tree models does not work well with this type of data, it loses a lot of information when splitting (COSC2670 Lecture week 7). The performance of the decision tree could be improved by increasing the depth, however, it would then be more susceptible to overfitting.

K-nearest has a superior f1 score for all activities. This is the harmonic mean for precision and recall, so there is no situation where one would favor decision tree over K-nearest neighbor.

One big challenge for both models is that the data is unbalanced, for example there are way more instances of 'Computer' than 'Walk + talk', in fact, 'Walk + talk' has only 7% of the instances 'Computer' has. For K-nearest neighbor classifier this result in way more instances of possible 'Computer' neighbors. We can see how this affects the precision, K-nearest neighbor classifies 'Computer' correctly 94% of the time, while for 'Walk + talk' only 55%. On the other side, decision tree does not manage to classify 'Walk + talk' at all with given hyperparameters. The reason for this is that with the current depth, the model does not manage to split such that one leaf node is in surplus of 'Walk+talk' classes.

From the decision tree we can see that the most significant variable is ‘y diff’, in other words the biggest difference in y-acceleration during a 1 second time window. From the data exploration stage we know this value can be seen on the line chart when y moves a lot up and down, typical for activities where you move, e.g. ‘Walking’ or ‘Stairs’.

In this project I decided to extract features from a 1 second time window. The value was arbitrary, and other intervals might work better. The same goes for features extracted, there exists probably more and better ways to extract features from frequency data, such as from this acceleration sensor data. The reason I decided to generate new features instead of just using the raw x-, y-, and z-acceleration values was because I could see in the data exploration stage that a lot of the raw values were very similar, especially evident in the scatter plot and histograms. After generating classification models based on this raw data, K-nearest neighbor had OK results, but decision tree failed completely to classify 2 classes (these models are in the bottom of the Jupyter Notebook). However, even though the basic summarization of the raw data were similar, when including a time aspect it was evident that the patterns in the data were different for each activity.

5 Conclusion

In this report I have looked at how to classify activity acceleration data. The conclusion for the task of choosing the best model to classify activity data is to use K-nearest neighbor. This model is best suited for the data and provides the best result. The decision tree is giving an unsatisfactory result, probably because of the data only having continuous variables.

To further improve the test result, one could balance the data, using equal numbers of datapoints of each activity to make the model. Another way to improve the predicting power is to find new features to extract, and find a better time interval to extract from.

6 References

Casale, P. Pujol, O. & Radeva, P. 2012, '*Personalization and user verification in wearable systems using biometric walking patterns*', Personal and Ubiquitous Computing, 16(5), 563-580, viewed 8 Jun 2020, <https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer>

COSC2670 Lecture week 7.