Assignment 3
Machine Learning for the Quantified Self

Group99: Haochen Wang[2698251], Shuhan Pi[2689783], and Xuan Zhou[2698592]

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam

1 Introduction

This report is about collecting self-quantified data to follow the machine learning for the quantified self cycle. We want to use the collected self-quantified data to train a model and use it to predict the user's activity label. First, we describe how we collected the data and the basic information about the dataset. In chapter 3, we describe how to process the data, including the handling of missing value and noise. In chapter 4, we describe our handling of the data features. We then describe how to partition the training set, the test set, and the test set. In the last two chapters, we present our attempts to train three classification models to predict activity labels and evaluate the findings.

2 Dataset Description

We used SensorLog to create a single user dataset with a duration of approximately 25 minutes, which consisted of 5 activities: walking, climbing stairs, putting the phone on the table, running, and taking the elevator.

After obtaining the raw data, we first performed some basic processing. We chose three sensors: Accelerometer, Gyroscope, and Magnetometer. We used a dataframe to divide the data by sensor and stored them in separate CSV files. Since the data does not contain the exact timestamp of each sensor but only the timestamp since the reboot, we use the phone's boot time plus the timestamp since reboot to get the exact timestamp of each Finally, we created a label CSV file with the start and end of the specific activity time.

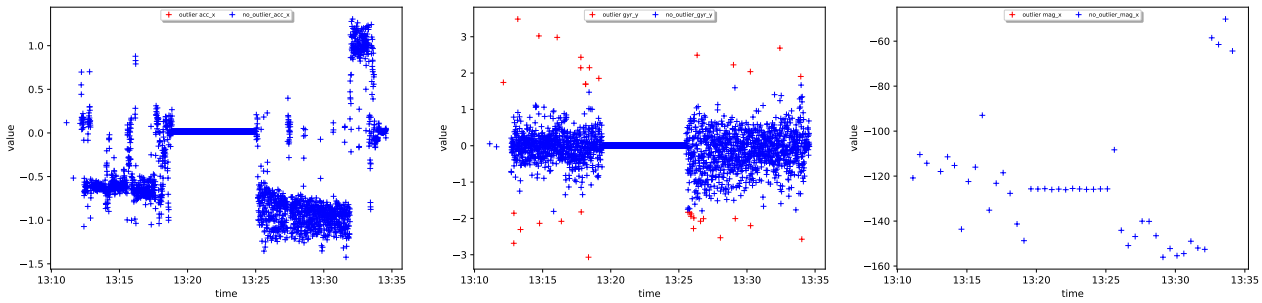
3 Data Pre-processing

The sensor data we have collected still exists an amount of noise, which contaminated our data and hind us from carrying out the machine learning tasks. Therefore, before we perform machine learning for quantified self, we require to preprocess the dataset to improve the accuracy of the results. Data preprocessing consists of detection and removal of outliers, imputation of missing values, and transformation.

3.1 Outliers Detection

Outliers are observations that are at an unusual distance from other values in the data set. When we start working with data from physical sensors, we are likely to encounter highly unlikely outliers that can affect the results of the analysis. So we first have to take some approaches to detect these possible outliers. In this assignment, we first consider Chauvenets Criterion. Because it can be estimated from the normal distribution function by calculating the mean and standard deviation of the observed data, the probability that an extreme value identical to a suspected outlier comes from a normal dataset. Due to the higher setting value of c, the higher probability to identify the true outlier, after several tests, we finally discovered the criteria setting of 5 is the best choice.

In Fig 1, we can see that the Chauvenet criterion does signal outliers for the gyroscope attribute: we found 25 outliers in gyr_y, which seems significant, and the results for the gyr_x and gyr_z attributes are similar to it. For acc_x and mag_x, we do not see very significant outliers, this is an indication that there is indeed a lack of very significant outliers. And by checking that the distribution of the values follows a normal distribution, as well as from visual inspection, it does indicate that the outliers found are appropriate. Based on this result, we decided to remove the outliers, e.g. by replacing the outlier with an unknown value using the Chauvenet criterion.



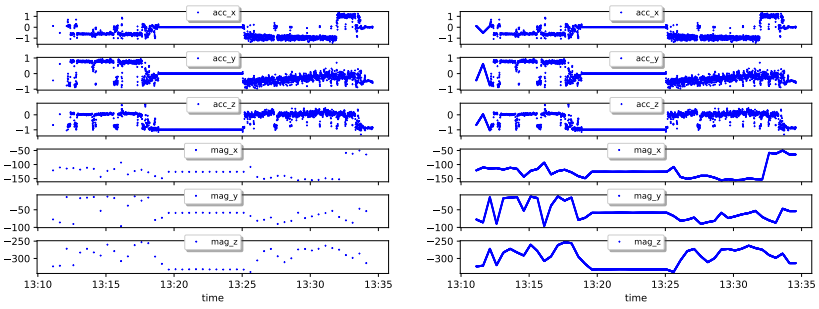
(a) (b) (c)

Fig. 1. Chauvenets Criterion

3.2 Imputation of Missing Values

Due to the fact that the sensors did not record any information at certain time points and our removal of outliers, it was clear that we had a large number of missing values in our dataset, which also affected our final predictions, so we necessarily had to replace these missing values in order to obtain reasonable results in the next tasks. One approach we considered in our assignment is linear interpolation, which works under the assumption that the missing values follow a linear trend, and in these cases, we use the next or previous time point and extrapolate the trend from these time points.

In Fig 2, we can observe the differences between before and after we impute the missing value. We have seen the accelerometer attribute and magnetometer attribute contain the most missing values. Also, we can see that by using interpolation for imputation, the data becomes more complete and it results in more natural values. Hence, this is the most efficient method for the time series in our dataset.



(a) (b)

Fig. 2. Comparison of before and after Imputation

3.3 Transformation

After removing outliers and adding missing values we required transforming our data to filter out subtle noise and identify the portion of our data that explained most of the variance. The approach we chose in this task is principal component analysis since we are able to use it to find the principal components that explain most of the variance in our measurements. We applied principal component analysis to all of our attributes, except for the labels. Since our dataset contains only 5 activities, we decided to select 5 components and include the values of each component for each time point in our dataset. Fig 4 presents the final processed dataset after all the steps we just explained.

4 Feature Engineering

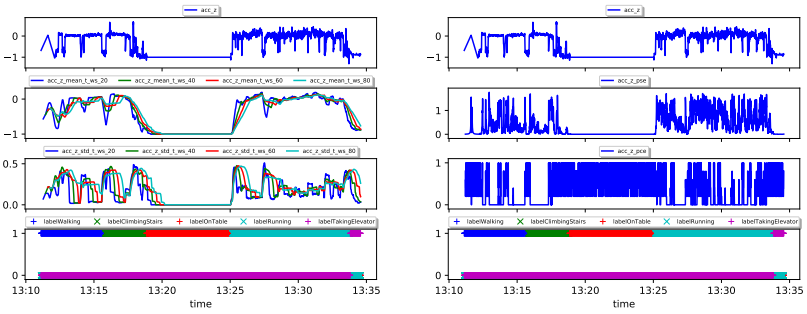
The purpose of feature engineering is to select and find a better feature for predictive models to be more accurate. For our assignment, we will mainly focus feature engineering on time domain, frequency domain, clustering and overlapping.

4.1 Time Domain and Frequency Domain

Because fluctuations in data and changes in measurements are potentially valuable for prediction tasks, to explore our dataset in more detail, we can first focus

on features derived from the time domain. We take the average of the standard deviation and window size to aggregate the data in the time domain. The standard deviation accounts for the variability of the data, while the mean accounts for general observations across past time points, which enables a more limited effect of individual noise measurements in it. We experimented with different window sizes 20 instances (10s), 40 instances (20s), and 60 instances (30s), and because 60 instances (30s) have neither excessive noise nor excessive variation, we finally choose it as the result of our project, and the Fig 3(a) shows the results. From the figure, we can conclude that the amplitude of the time domain is compatible with the dynamic and static state of the activity

Besides the time domain, we also explored frequency domain features in our assignment, and we choose Power spectrum entropy(PSE) and cepstrum as metrics. PSE is used to measure the concentration of the distribution of each frequency component in the power spectrum and cepstrum can easily extract and analyze periodic signals that are difficult to identify on the original spectrogram. From 3(b), we can observe that the amplitude of the PSE criterion is clearly differentiated between stationary and non-stationary states, while the cepstrum criterion can distinguish the amplitude changes in different activities.



(a) (b)

Fig. 3. Time Domain and Frequency Domain

4.2 Clustering and Overlapping

To obtain more insights into our dataset, constructing the clusters is a good approach. Clustering allows members of a cluster to make predictions about the target. Since we only have a single dataset, our goal for this project is to cluster the instances in the dataset and we will focus on the phone magnetometer. By measuring the silhouettes, we determined that the optimal setting for the number of clusters is k = 6. In Fig 5, we see a fairly good consistency of clustering. For example, the yellow cluster appears to contain a large number of labeled points for taking the elevator, and climbing stairs are distributed in large amounts in the blue and green clusters. It follows that clusters and activities of clusters are not randomly distributed.

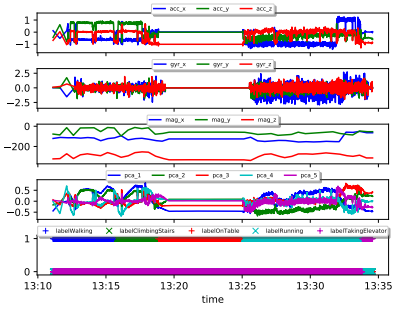


Fig. 4. Dataset after Processing

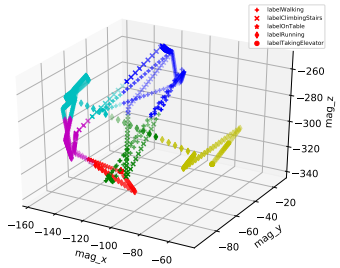


Fig. 5. K-means Clustering

As for overlapping, since after feature engineering, there are 939 instances with 121 features, they are highly correlated, which may cause over-fitting. Therefore, we usually set a maximum overlap for the window and accordingly remove instances that do not satisfy this criterion. The window size in our assignment is set as 0.95, the reason is that the amount of data is limited and we allow some degree of overlap.

5 Data Preparation

5.1 Feature Selection

By now, we have 121 features in our dataset after feature engineering. Useless features would have a severe impact on the performance of our learning algorithms.[4] Therefore, we are going to shrink the number of the features to train. We select the attributes we are going to use to predict the activity by the forward selection.

One simple decision tree with at least 50 sample leaves is applied to find important features. The procedure of feature selection is essentially some machine learning tasks with different features to input. In the forward selection, we use 70% of the dataset for training and the rest 30% for testing. We choose to use the simple decision tree because it is the base learner of one of our models.

The impact of adding the best features iteratively for 100 features is shown in Fig6. We can find that after 4 features the accuracy can reach the highest and keep steady for a while. However, with over 75 features, the accuracy begins to go down and swing with the number increasing. This is a good example to show the severe impact on the performance. Therefore, we choose the first 7 most important features to train in the next section.

5.2 Dataset Splitting

We can now have 939 instances with 7 features and one label. Because of the setting of window size in the feature engineering, 21 instances are containing

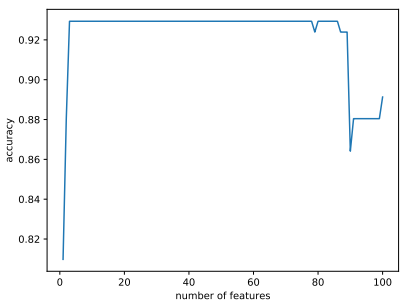


Fig. 6. Features Forward Selection

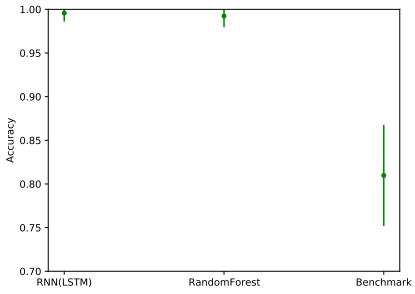


Fig. 7. Accuracy of the models

some null values at the beginning of the dataset. They have to be omitted or deducted. Thus, the total amount of instances in the dataset is 918.

For reasonable in-sample error and out-sample error, the dataset is split into a training set which takes 80% of the data having 734 instances, and a testing set which takes the rest 20% with 184 instances. In this way, it can help us to know the performance and avoid overfitting as well. Here we deploy two approaches of learning algorithms.

For the time isolated method, random forest, we split the dataset into the random sample. It can make sure that the labels are represented equally frequently in both the training set and the testing set. Thanks to the nature of the random forest, the scores of samples out of bags are used to evaluate the performance. With such nature, there is no need to especially implement cross-validation because of the existence of out-sample instances in bootstrapping.

For the time-related method, recurrent neural network, we split the dataset with the time order. We take the first 80% of the data directly as the training set. And 33.3% of it is used as the validating set. The last 20% of the data is the testing set.

6 Models

As we mentioned before, our machine learning task is to do classification on the activities. We build three models for the task. They are random forest which is without time notion, recurrent neural network(LSTM) which is with time notion, and a simple decision tree which serves as a benchmark. We are going to detailedly discuss this in this section.

6.1 Random Forest

Random forest(RF) uses a decision tree as a base learner and builds a bagging aggregation.[1] Several base decision trees can be integrated or aggregated into a final learning model.[2] It is the reason we exploit the simple decision tree to select important features. It can effectively boost performance. That is also why we use a decision tree as a benchmark method later.

To be specific, there are 7 features in the initial. As illustrated before, the processed data contains the values representing principal components, time domain, frequency domain, and clustering information. We try to find the best parameter to regularize these values. We decide to use 0.001 which is the best regularization parameter among 0.0001, 0.001, 0.01, 0.1.

Besides, the number of estimators (trees) is 50. The maximum depth is 10. The number of minimum samples leaves is 2. The criterion of information gain is *gini* method. All the above parameters are concluded from the grid search, which can pick up the combination of parameters leading to relative best performance. The accuracy of out-of-bag instance is used to show the performance when training, the same function as validation. The accuracy of the testing set is used to evaluate the random forest model performance in Section 7.

6.2 Recurrent Neural Network

Quantified-self data, in this project the sensory data from our daily activities with labels, must be related to time sequence. The activities together with sensory data could be influenced by the previous states. In order to capture the features of such sequence, we propose a learning model with time notion, recurrent neural network (RNN). We use one of its famous varieties, LSTM (long short-term memory).

Considering the past sequences of our data are not short, LSTM might perform better at storing and processing such longer sequences.[3] From the data preparation in Section 5, we use these 7 features for RNN to train. The data splitting is according to Section 5. Since from the same dataset and classification task, the regularization is the same as what we have done in the random forest model.

For the architecture of RNN, We deploy two layers of LSTM and one fully connected output layer in our model. The two LSMT layers are 256 cells and 64 cells respectively with an activation function of $tanh$. And a dropout layer of 0.3 is followed each LSTM to reduce the potential of overfitting. The fully connected (dense) layer is of 5 neurons, which represent five classes of the activities. The activation function of output is $Softmax$. The metric to optimize is accuracy, which is common for classification. When training, the validation with the validating set, stated in Section 5.2, can indicate the performance and avoid overfitting. Additionally, we do not use early stopping because the number of the epoch is set to 20, not big.

6.3 Benchmark

To prove the effectiveness of our models and compare them, we implement a benchmark model of one simple decision tree. The reason we use the simple decision tree as the benchmark model is that we used it in feature selection and the base learner of random forest, which fits our goal of evaluation. The performance of the benchmark model is a baseline in evaluating the above two models.

The benchmark is straightforward both in feature selection and parameter setting. The features we use are the original data from the accelerometer, gyroscope, and magnetometer with activity labels. The maximum depth is 3. The minimum sample leaves is 1. The information gain is *gini* method, which keeps consistent with the random forest model.

7 Results and Evaluation

The results for the models are shown in Fig9 and Fig7. From the table, the classification task is solved by high-performance machine learning methods. Fig8 is the confusion matrix for the prediction of the random forest model. With a total of 184 testing instance, there is only one incorrect prediction that it misclassifies taking elevator into running.

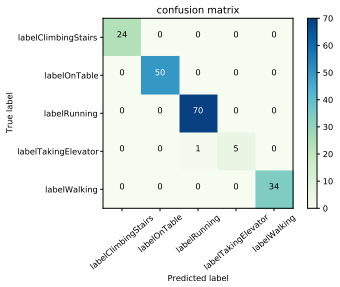


Fig. 8. Prediction with RF

Model	Mean Accuracy
Random forest	99.239%
RNN(LSTM)	99.565%
Benchmark	80.978%

Fig. 9. Accuracy of the models

From Fig7, we can clearly observe that our recurrent neural network model and random forest model are valid models to realize the classification task. Compared with the benchmark, we can say that these two models are much more accurate than the benchmark. From our observations, we conclude that the RNN(LSTM) is a better model to use for our dataset since it is useful in time series prediction and the features could remember previous input. The random forest model, on the other hand, does not consider the time relationship among the instances and is, therefore, less accurate. However, it even shows better performance than the benchmark for the dedicated selection of features.

8 Conclusion

In this assignment, we performed a machine learning for quantified self circle on our collected sensory data, we pre-process the collected data including assigning the labels, detecting outliers, imputing missing values and transformating then. Then we performed feature engineering on the processed data to convert into useful features. Using RNN and random forest models are used to predict activities and we evaluate performance of the models. We also found that self-quantified datasets are not commonly available online, but that the data and information collected by sensors can predict many interesting things.

References

1. Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.
2. Breiman, Leo. "Bagging predictors." Machine learning 24.2 (1996): 123-140.
3. Olah, Christopher. "Understanding lstm networks." (2015).
4. Hoogendoorn, Mark, and Burkhardt Funk. "Machine learning for the quantified self." On the art of learning from sensory data (2018).