SAPIENZA
UNIVERSITÀ DI ROMA

A Comparative Study of Machine Learning Models for Kinect-Based Data in Movement Classification

Faculty of Information Engineering, Computer Science and Statistics
Bachelor's Degree in Computer Science

Lucian Dorin Crainic
ID number 1938430

Advisor
Prof. Maurizio Mancini

Academic Year 2023/2024

Thesis not yet defended

A Comparative Study of Machine Learning Models for Kinect-Based Data in Movement Classification
Bachelor's Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: crainic.lucian@gmail.com

Abstract

This thesis conducts a detailed comparative study of several Machine Learning models, with a focus on their application to Kinect-based data for classifying human movements. The primary aim of this research is to evaluate these models to determine the most effective ones for accurately classifying movements recorded through Kinect sensors.

This study begins with a introduction to Kinect technology, highlighting its ability to capture detailed movement data. Following this, an examination of a range of Machine Learning models, such as Support Vector Machines, Random Forest, Linear Regression, and so on. Each model is tested to evaluate its accuracy, processing efficiency, and robustness in accurately classifying various movements.

The core of this comparative analysis is a diverse dataset consisting of several movements captured through a Microsoft Kinect. The research methodology involves several steps: processing the Kinect data, extracting key features that are characteristic of specific movements, and applying the selected models to this improved data. Performance evaluation of each model using standard metrics like accuracy, precision, recall, and the F1 score, which provide a complete picture of their effectiveness.

Over this study, valuable understandings are gained into the specific strengths and limitations of each model in the context of Kinect-based movement classification. The findings reveal that some models prove enhanced performance in certain situations, which is influenced by factors like the complexity of the captured movements and the characteristics of the dataset.

This thesis acts as a useful guide for researchers and professionals. It helps them pick the best models for similar work and sets the stage for more research in this area. This study contributes to the advancement of accurate and efficient Kinect-based data movement classification using Machine Learning methods, leading to more progress in this field.

Contents

Chapter 1

Introduction

1.1 Problem Statement

1.2 Literature Review

1.3 Dataset Overview

1.4 Aims and Objectives of the Study

Chapter 2

Dataset Analysis

2.1 Data Collection Methodology

2.1.1 Microsoft Kinect

2.1.2 Recording Setup

2.2 Data Structure and Attributes

2.3 Participants Characteristics

2.4 Movements Visualization

2.5 Data Processing

2.5.1 Data Cleaning

2.5.2 Data Normalization

2.5.3 Data Transformation

Chapter 3

Methodology

3.1 Overview of Models Analyzed

3.1.1 Scikit-Learn

3.1.2 Models Selection

3.2 Analysis of Top-Performing Models

3.2.1 Support Vector Machine

3.2.2 Random Forest

3.2.3 Gradient Boosting

3.2.4 Logistic Regression

3.2.5 Linear Discriminant Analysis

3.2.6 Multi-Layer Perceptron

3.3 Data Splitting Methods

3.3.1 Traditional Approach

3.3.2 Effective Approach

3.3.3 Sequential Data Approach

3.4 Feature Engineering

3.4.1 Overview

3.4.2 Calculation Methods

Chapter 4

Results and Discussion

This chapter presents the result obtained from the expirements conducted in the previous chapter. Classification models are evaluated using different approaches and metrics. The results are then discussed and compared to each other.

4.1 Models evaluation

Performed using the *Scikit-Learn* library [1]. It provides a wide range of validation methods and metrics to evaluate the performance of the models. The following sections will present the validation methods and metrics used in this thesis.

4.1.1 Validation

Validation is the process of evaluating the perfomance of the models. The goal of validation is to estimate the performance of the model on new data, not used during the training process. The following validation methods are used:

Hold-Out

This method is widely used for for its simplicity and speed. The dataset is split into two subsets. The Training set is used to train the model, Testing set is used to evaluate the performance of the model. Typically, a common split ratio is:

- **Training set**: 70% of the dataset.
- **Testing set**: 30% of the dataset.

Cross-Validation

This method is used in the literature for its effectiveness and robustness. It can be time consuming for large datasets, but it is the best method to evaluate the performance of the models.

- **K-fold**: The data is divided into K folds, then K-*1* folds are used for training and the remaining fold is used for testing. This process is repeated K times,

with each fold being used exactly once for testing. The fig:kfold shows an example of a k-fold split. Fig 4.1 shows the KFold split.

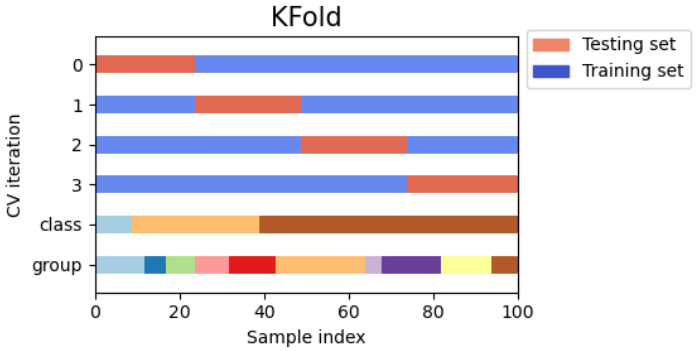

Figure 4.1. KFold Visualization from the scikit-learn documentation [2].

- **Group-k-fold**: Variation of k-fold designed for situations where the data has inherent groupings or dependecies that should be preserved in the train/test split. In this method, the data is divided into K folds, then an additional constraint is imposed to ensure that data point from the same group are in the same fold. Fig 4.2 shows the GroupKFold split.

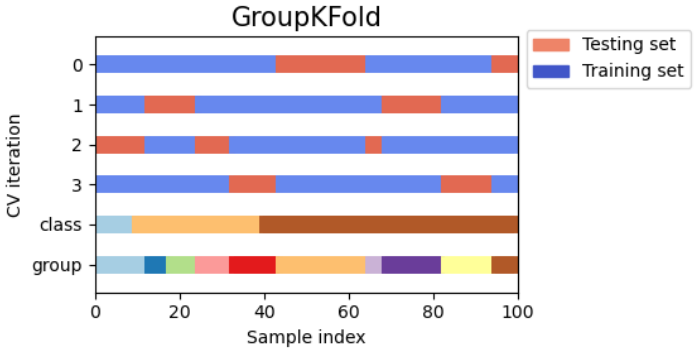

Figure 4.2. GroupKFold Visualization from the scikit-learn documentation [2].

The advantage of using *Cross-Validation* over *Hold-Out* is that all the samples are used for both training and testing, and each sample is used for testing exactly once. This method helps to reduce the variance of the estimated performance of the model, by averaging the results over a number of trials. The disadvantage of using Cross-Validation is that it is computationally expensive for very large datasets.

In this study, both methods are used to evaluate the performance of the models. The Hold-Out method is used to evaluate the perfomance of the models for the *Wrong approach* and *Sequence approach* datasets due to their large dimension. The Cross-Validation method is used with the Hold-Out method to evaluate the perfomance of the models for the *Correct approach* and *Feature Engineering approach*

dataset due to them scoring the best results and being the effective approaches. Confronting the results of the two methods will show the correctness of the evaluation, as the results should be similar.

4.1.2 Metrics

This section will report the metrics used to benchmark the different models used in this study.

Accuracy score

The *accuracy* is the proportion of correct predictions, considering both true positives and true negatives, among the total number of samples. The formula used to calculate the accuracy is the following:

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

where **TP** is the number of true positives, **TN** is the number of true negatives, **FP** is the number of false positives and **FN** is the number of false negatives.

Precision score

The *precision* is the ability of the classifier not to label as positive a sample that is negative. The formula used to calculate the precision is the following:

$$\frac{TP}{TP + FP} \tag{4.2}$$

Recall score

The *recall* is the ability of the classifier to find all the positive samples. The formula used to calculate the recall is the following:

$$\frac{TP}{TP + FN} \tag{4.3}$$

F1 score

The *F1 score* is the harmonic mean of the precision and recall. The formula used to calculate the F1 score is the following:

$$\frac{2 \times (precision \times recall)}{precision + recall} \tag{4.4}$$

Matthews correlation coefficent

The *Matthews correlation coefficent* (or φ coefficient) takes into account true and false positives and negatives and is regarded as a balanced measure which can be

used even if the classes are of very different sizes. The formula used to calculate the φ coefficient is as follows:

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.5}$$

These metrics will be used to show the effectiveness of the approaches proposed in this thesis.

4.2 Evaluation results

Combining the validation methods and metrics presented above, the following tables will show the results obtained from the expirements conducted. The results are divided into two categories: **Exploratory** shows two approaches that were tested but did not obtain good results due to wrong implementation or loss of information. **Effective** shows two approaches that obtained good results and are suitable for this task. The results are presented in the following order: *Traditional Approach*, *Sequence Approach*, *Effective Approach* and *Feature Engineering Approach*.

4.2.1 Exploratory

The following approaches are included because they a starting point in this thesis, and show the how different approaches can affect the accuracy of the models.

Traditional Approach

This approach was the first one to be tested and it got surprisingly good results. Such a simple approach and yet high accuracy raised doubts about the validity of the results, after further investigation it was discovered that the dataset was not properly split into training and testing sets as stated in Chapter 3.3.1.

Table 4.1. Evaluation results using **Hold-Out** validation method.

Model	Accuracy	F1	Recall	Precision	MCC
Random Forest	**0.9958**	**0.9948**	**0.9950**	**0.9947**	**0.9953**
K-Nearest Neighbor	0.9847	0.9811	0.9814	0.9809	0.9826
Decision Tree	0.9651	0.9596	0.9599	0.9594	0.9605
Support-Vector Machines	0.8770	0.8605	0.8586	0.8667	0.8610
Logistic Regression	0.8264	0.8048	0.8022	0.8097	0.8035

In Table 4.1 the results obtained from the Hold-Out method are shown. High values are obtained for all the metrics, with **Random Forest** obtaining the highest values with a score of **0.9958** for accuracy. This confirmed the doubts about the validity of the results, a patient is both present in the training and testing set. This led to the models overfitting the data and obtaining high accuracy.

Sequence Approach

This approach got the lowest results of all the approaches. It was tested to see if concatenating the frames of a movement into a sequence would help the models differentiate between movements as stated in Chapter 3.3.3.

Table 4.2. Evaluation results using **Hold-Out** validation method.

Model	Accuracy	F1	Recall	Precision	MCC
K-Nearest Neighbor	**0.5659**	**0.5414**	**0.5474**	**0.5497**	**0.5158**
Random Forest	0.5463	0.5289	0.5272	0.5550	0.4932
Support-Vector Machines	0.5268	0.4716	0.4924	0.4677	0.4727
LogisticRegression	0.4439	0.4156	0.4274	0.4352	0.3812
Decision Tree	0.4390	0.4137	0.4190	0.4262	0.3749

In Table 4.2 the results obtained from the Hold-Out method are shown. Low values are obtained for all the metrics, with **K-Nearest Neighbor** obtaining the highest values with a score of **0.5659** for accuracy. This results are considered low based on other approaches, however in the context of randomly guessing the movement of a patient the accuracy would be **0.1** as there are 10 movements. This means that the models are able to differentiate between movements, but not with a high accuracy. This approach was not used in the other approaches as it was not able to obtain a high accuracy.

4.2.2 Effective

The following approaches are the ones that obtained the best results and are suitable for this task. The main difference between the two approaches is the data used to train the models. The *Correct Approach* uses the data as it is from the Kinect sensor, while the *Feature Engineering Approach* uses the data after applying feature engineering techniques.

Effective Approach

This approach is considered effective because the raw kinect data is able to obtain a high accuracy. The data is not modified in any way, beside the removal of rotational data and pre-processing the data to remove noise as described in Chapter 3.3.2.

In Table 4.3 the results obtain from the Hold-Out method are shown. **Random Forest** obtains the highest values for all the metrics, with a score of **0.7461** for accuracy. Other models such as *Gradient Boosting*, *Linear Discriminant Analysis*, *Support-Vector Machines*, *K-Nearest Neighbors* obtained great results as well with a score greater than **0.70** for accuracy. This confirms that the data obtained from the Kinect sensor is suitable for the task of movement classification without any major tweaks.

Table 4.3. Evaluation results using **Hold-Out** validation method.

Model	Accuracy	F1	Recall	Precision	MCC
Random Forest	**0.7461**	**0.7322**	**0.7386**	**0.7302**	**0.7124**
Gradient Boosting	0.7344	0.7230	0.7260	0.7248	0.6989
Linear Discriminant Analysis	0.7232	0.7157	0.7111	0.7421	0.6887
Support-Vector Machines	0.7187	0.7182	0.7155	0.7274	0.6807
KNN	0.7101	0.6971	0.6980	0.7031	0.6716
Logistic Regression	0.6688	0.6467	0.6488	0.6468	0.6244
Decision Tree	0.6553	0.6328	0.6468	0.6317	0.6109
Naive Bayes	0.6333	0.6077	0.6105	0.6226	0.5840
Multi-Layer Perceptron	0.6278	0.5962	0.6241	0.6131	0.5840
AdaBoost	0.3563	0.2209	0.2888	0.2405	0.3205

In Table 4.3 **Hold-Out** validation method was used for all the models, while in Table 4.4 **Cross-Validation** was used with only 3 models to compare the two validation methods and show that there is no major difference between them. The results obtained from the two methods are similar, with the Cross-Validation method obtaining slightly lower values for all the metrics. This is due to the fact that the Cross-Validation method is more robust and the results are averaged over a number of trials.

Another observation is that the Hold-Out method was used for it's speed, with **10 minutes** of training time while cross-validation was run for hours without finishing. This is due to the fact that this approaches uses the raw Kinect data, which contains over **59000** rows and **100** columns.

Table 4.4. CV = Cross-Validation, HO = Hold-Out

Model	Accuracy	F1	Recall	Precision	MCC
LDA (CV)	0.6362	0.6169	0.6025	0.6732	0.5942
LDA (HO)	0.6362	0.6156	0.5987	0.6541	0.5905
KNN (CV)	0.6127	0.5832	0.5919	0.5917	0.5632
KNN (HO)	0.6292	0.5941	0.5971	0.5974	0.5788
Naive Bayes (CV)	0.5884	0.5551	0.5621	0.6176	0.5414
Naive Bayes (HO)	0.5747	0.5576	0.5538	0.5980	0.5180

Model	Accuracy	F1 Score	Recall	Precision	MCC
Gradient Boosting	**0.7137**	**0.7059**	**0.7133**	**0.7048**	**0.6711**
SVM	0.7024	0.6860	0.6976	0.6909	0.6594
LDA	0.6933	0.6829	0.6725	0.7233	0.6513

Table 4.5. Table 1

Model	Accuracy	F1 Score	Recall	Precision	MCC
SVM	**0.7073**	**0.6908**	**0.6984**	**0.6942**	**0.6630**
LDA	0.6890	0.6732	0.6651	0.7039	0.6452
Gradient Boosting	0.6887	0.6798	0.6866	0.6817	0.6421

Table 4.6. Table 2

Feature Engineering Approach

4.3 Discussion

Chapter 5

Conclusions

This chapter presents the key findings of the thesis, highligthing its limitations and review potential approaches for future research to improve the results and develop more effective models.

5.1 Discoveries

Presented below are the key findings of this thesis:

1. *Preprocessed raw data* obtained from the Kinect sensor is suitable for the task of movement classification. However, it alone does not provide enough information for the models to obtain a high accuracy.

2. *Feature engineering* is a crucial process of creating new features from the raw data with the goal of improving the accuracy and training time of the models.

3. *Linear Discriminant Analysis* is the best perfoming model for this task, with a score of **0.82** using 10 movements and **0.93** using 9 movements after removing one of the similar movements and using a feature engineering approach.

4. *Data splitting* tecniques are an essential step in the process of training the models, an incorrect split can lead to overfitting and an incorrect evaluation. Using the "Patient-ID" as a split criteria is the best approach to avoid any data leakage between the training and testing sets.

5. *Sequence of frames* is not a good approach to take for this type of data, due to every movement having a variable number of frames and Machine Learning models need a fixed length input. Transforming the data into fixed length sequences will lead to a loss of information and a decrease in accuracy.

6. *3D visualization* of the movements allow to visually identify and label them. It was found that "Mat-Walk" and "Hoop-Walk" movements are very similar, with the only difference being the object that the patient is walking over. This led to the models struggling to differentiate between these two movements and by removing one of them from the dataset the accuracy of the models improved.

5.2 Limitations

Limitations encountered in this work will be presented, along with an exploration of their impact and the strategies used to overcome them.

A list of 10 movements names was provided with the dataset, however the movements were not labeled accordingly to the list and only a unique ID has been assigned to each one. This led to the need of updating the labels after visually identifying them with the help of the 3D visualization.
While the data was collected an unknown number of movements have not been performed correctly by the patients. It was not possible to develop a tecnique that would identify and remove them, so they have been kept in the dataset. This limitation may have affected the accuracy of the models due to the noise introduced.

The dataset dimensions is relatively small, with only 10 movements and 21 patients. This led to only using the Training and Testing sets for the evaluation of the models, as the dataset was too small to split it into Training/Validation/Testing sets. A larger dataset is needed to split it into these sets and evaluate the models better.
Features calculated in the feature engineering step are not accurate to literature due to only using positional data from the Kinect sensor. However, they still provide enough information for the models to obtain a high accuracy.

5.3 Future work

Kinect-based data is suitable for the task of movement classification, this leads to the possibility of implementing new tecniques and approaches to improve the accuracy of the models.

Feature engineering approach obtains a high accuracy and reduce the training time of the models by reducing the dimension of the original dataset. It is recommended to use it if the dataset is going to be scaled up to include more movements and patients, as the training time will increase exponentially.
The number of features have been reduced but it was not possible to tell which ones contribute the most to the accuracy of the models. In future work it is recommended to calculate the importance of each feature and remove the ones that do not contribute to the accuracy of the models. With the help of domain experts it is possible to calculate new and more meaningful features that can help the models differentiate between movements.
As stated before two movements (*Mat-Walk* and *Hoop-Walk*) are very similar. It is suggested to remove one of them from the dataset to obtain a realistic evaluation of the models, this will help to differentiate between movements that are very similar.

This thesis only used Machine Learning models from the *Scikit-Learn* library. It is possible to implement new models from the *TensorFlow* library, such as *Convolutional Neural Networks* and *Long Short-Term Memory* networks. These models

are more complex and require a larger dataset to train on, but they can obtain a higher accuracy than the models used in this thesis with a correct implementation. It is crucial to acquire new data from the Kinect sensor, add new movements and patients. This will allow to scale up the dataset and study how the models perform on more classes and patients.

The possibles approaches for future work on this task are endless, above are only a few suggestions that can be implemented. The goal of this thesis was to study the feasibility of using Kinect-based data for movement classification, and provide a baseline for future research with this type of data.

Bibliography

[1] Buitinck, L., et al. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122 (2013).

[2] Pedregosa, F., et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12** (2011), 2825.