Work Sheet

A detailed description of certain methodological topics, including problems and thoughts, will be presented in the following worksheet

A new way of measuring loading in team sports

Using locomotion categories to predict drill types

By

Nicki Lentz-Nielsen

Master in Sports Technology
Master Thesis
Delivery deadline: 01-06-2022

Contents

Acknowledgements

The R code

This section will go through some of the functions that have been made to clean, manage, manipulate and model the data for this project. Arguments will be made for why and how these functions have been used while presenting exemplatory results of how the functions have worked. Remember to read the comments inside the codeblocks to understand some of the choices that have been made. All of the code can be found on github: **INSERT LINK TO MY GITHUB WHEN IT IS READY**

R and specifically tidyverse (a collection of R packages for data science) has two important features that should be noted to easier read the codeblocks presented. Most functions are named as verbs so it should be easy to rationalize what it does. The other feature is the pipe operator; *%>%*. When ever a pipe is present, read it as *"and then do"*. It basically pipes the previous object into the next function.

Importing and cleaning the data

When FMP is exported from Catapults Vision Exporter tool it looks like table 1

Table 1: Example output from Catapults Vision Exporter. Start and end time are time periods in unix time, movement type is the predicted football movement profile and duration is the length in seconds before a transition to a new movement type.

Athlete	startTime	endTime	movementType	duration [s]
2Y8 Yellow-8	1645080999	1645081002	Low Intensity	3
2Y8 Yellow-8	1645081003	1645081004	Low Intensity	1
2Y8 Yellow-8	1645081004	1645081004	Dynamic Medium Intensity	1
2Y8 Yellow-8	1645081008	1645081010	Low Intensity	2
2Y8 Yellow-8	1645081013	1645081016	Low Intensity	3

The start and end time columns are the time period in unix time. The duration shows number of seconds the given FMP category has occured. For example, the first row indicates that *Low Intensity* occured for 3 s, then another 1 s before changing to *Dynamic Medium Intensity.* Another important observation is the jump in time between row 1 and row 2 where the data jumps 1 s and again between row 3 and row 4 where the data jumps 3 s. The reason for this is the lack of the *Very Low Intensity* category in the data set. Catapult chose to omit this category and use the duration column format to sparse the stored files in the cloud.

Therefore, a function was created to lengthen the data to a 1Hz data set, and also pad the time gabs with the *Very Low Intensity* category, as can be seen in following codeblock.

```r
cleanFMPdata <- function(data){
  #This function will pad with "very low intensity"
  #when there is a time gab between rows.
  #Furthermore it will expand / lengthen the data frame to 1hz data set.



  #Create an index for easy sorting purposes later on.
  #Calculate time difference between end_time on row [i]
  #and start_time on row [i+1] in unix time
  tableData <- data %>%
    select(athlete, start_time, end_time, movement_type, duration) %>%
    #Adding one hour to get the unix time to the correct time zone
    mutate(start_time = start_time + 3600,
           end_time = end_time + 3600,
           ind = seq_len(nrow(.)),
           timeDifference = floor(lead(start_time, 1) - end_time),
           #Last row won't be able to find a difference,
           #therefore replace NA with 0.
           timeDifference = replace_na(timeDifference, 0)) %>%
    arrange(start_time)



  #Padding with "very low intensity" if the unixtime gap between
  #two adjacent rows is >= 1, and adding a 0.1 to the idx
  #to correctly arrange the dataframe relative to the time.
  for (i in 1:nrow(tableData)){
    if (tableData[i]$timeDifference >= 1){
      tableData <- rbind(tableData, data.frame(
        athlete = tableData$athlete[i],
        start_time = tableData$start_time[i],
        end_time = tableData$end_time[i],
        movement_type = "Very Low Intensity",
        duration = tableData[i]$timeDifference,
        ind = i+0.1,
        timeDifference = tableData[i]$timeDifference
      ))
    }
  }

  #Create a nested vector with movement type times duration,
  #then unnest the vector to lengthen the dataframe to a pure 1hz dataset.
  tableData <- tableData %>%
    arrange(ind) %>%
    mutate(movement_type = map2(movement_type, duration, ~rep(.x, .y))) %>%
    unnest_longer(movement_type)

  #fixing the unixTime now that we have a 1hz dataframe.
  finishedTableData <- tableData %>%
    mutate(unixTime = seq(
      from = .$start_time[[1]],
      to = .$start_time[[1]] + nrow(tableData)-1,
      by = 1)) %>%
    select(-c(duration, ind, timeDifference, start_time, end_time))

  return(finishedTableData)
}
```

The resulting data will look like table 2

Table 2: Example output from the full size data set.

athlete	movement_type	unixTime
2Y8 Yellow-8	Low Intensity	1645084599
2Y8 Yellow-8	Low Intensity	1645084600
2Y8 Yellow-8	Low Intensity	1645084601
2Y8 Yellow-8	Very Low Intensity	1645084602
2Y8 Yellow-8	Low Intensity	1645084603
2Y8 Yellow-8	Dynamic Medium Intensity	1645084604
2Y8 Yellow-8	Very Low Intensity	1645084605
2Y8 Yellow-8	Very Low Intensity	1645084606
2Y8 Yellow-8	Very Low Intensity	1645084607
2Y8 Yellow-8	Low Intensity	1645084608
2Y8 Yellow-8	Low Intensity	1645084609
2Y8 Yellow-8	Very Low Intensity	1645084610
2Y8 Yellow-8	Very Low Intensity	1645084611
2Y8 Yellow-8	Very Low Intensity	1645084612
2Y8 Yellow-8	Low Intensity	1645084613
2Y8 Yellow-8	Low Intensity	1645084614
2Y8 Yellow-8	Low Intensity	1645084615

Validation

During the validation process of FMP the intensity part of the category was removed. This was done by using regular expresions (regex) to remove parts of the word and then mutate the categories down to four: No locomotion, Walking, Dynamic and Running

```
FMPTuesdayNIU2 <- data.table::fread("Data/ProcessedData/FMPTuesdayNIU2.csv") %>%
  #split movement type to get locomotion category and rename
  mutate(locomotion = sub(' .*', '', movement_type),
         locomotion = case_when(
           locomotion == "Low" ~ "Walking",
           locomotion == "Very" ~ "No locomotion",
           TRUE ~ as.character(locomotion)
         ))
```

After changing the categories, a function was created to create a truth-vector to verify the FMP with. This had to be done in respect to the right time periods and athletes for each of the controlled drills. It is important to note that "intensity" are used in the function but only used as a way of differentiating between the different drills and not related to the original FMP. Following codeblock shows the function used.

```
locomotionFMPtruth <- function(FMPdata, eventIntensity, eventType,
                               athleteID = NULL, groupID, startTime, endTime){

  #securing capital first letter
  eventIntensity <- tools::toTitleCase(eventIntensity)
  eventType <- tools::toTitleCase(eventType)

  #AthleteID was used during the non-linear locomotion drill as only one
  #Athlete attended at a time. During linear locomotion AthleteID were not used
  #as everyone attended at the same time.
  if (is.null(athleteID)) {
    athleteID = FMPdata$athlete
  }

  #Synchronizing the start and end time.
  syncFMPData <- FMPdata %>%
    filter(athlete == athleteID,
           unixTime >= startTime,
           unixTime <= endTime) %>%
    arrange(athlete, unixTime)
```

```r
  #Creating new vector for the "truth" and the event "intensity".
  FMPtruth <- syncFMPData %>%
    mutate(truth = eventType,
           intensity = eventIntensity,
           athlete = paste(groupID, athlete, sep = "_"))

  #Changing the character vectors to factors.
  FMPtruth$truth <- factor(FMPtruth$truth,
                           levels = c("Dynamic", "Walking",
                                      "Running", "No locomotion"))
  FMPtruth$locomotion <- factor(FMPtruth$locomotion,
                                levels = c("Dynamic", "Walking",
                                           "Running", "No locomotion"))

  return(FMPtruth)
}

#Example for Linear locomotion
Tuesday_NIU2_Walk2 <-
  locomotionFMPtruth(FMPTuesdayNIU2,
                     eventIntensity = "low",
                     eventType = "Walking",
                     groupID = "TuesdayNIU2",
                     startTime = "2022-02-15 08:31:10 UTC",
                     endTime = "2022-02-15 08:31:53 UTC")

#Example for non-linear locomotion
Tuesday_NIU2_high_Yellow2 <-
  locomotionFMPtruth(FMPTuesdayNIU2,
                     eventIntensity = "high",
                     eventType = "Dynamic",
                     athleteID = "Yellow-2",
                     groupID = "TuesdayNIU2",
                     startTime = "2022-02-15 09:06:55 UTC",
                     endTime = "2022-02-15 09:07:31 UTC")
```

After creating the truth-vector for all iterations of the controlled drills, each drill and its iterations would be concatenated into one dataframe before a percentage-agreement test was conducted. See the next codeblock

```
LinearLocomotion10 <- rbind(Tuesday_NIU2_Run10,
                            Tuesday_NIU3_Run10,
                            Wednesday_NIU1_Run10)


LinearLocomotion10Agree <- LinearLocomotion10 %>%
  select(truth, locomotion) %>%
  irr::agree(.)
```

Improvement to FMP

FMP is a proprietary algorithm within Catpault sports, which means that the source code is not openly available. Therefore, the changes that was applied to FMP would be performed directly on the FMP data and not on the raw acceleration data. Football consists of numerous changes of directions, accelerations and decelerations. Consequently, a linear locomotion can segue to a hard deceleration in an instant. Therefore, it was important to increase the window size of the FMP from 1 s to 0.5 s. A 0.5 s approach has been used before in the litteratur of human activity recognition, for example for daily tasks (Chavarriaga et al. 2013) or falls (Miller et al. 2022). Furthermore, Dehghani et al. (2019) found window sizes between 0.25-1.25 s to be optimal for human activity recognition when creating aggregated features for the feature engineering process. Consequently, the present project opted for a window size equal to 0.5 s.

To increase the frequency, each of the rows would be duplicated relative to each athlete for each group and set in correct order relative to time while also calculating a new time vector as can be seen in the next codeblock.

```
FMPTuesdayNIU2 <- FMPTuesdayNIU2 %>%
  group_by(athlete) %>%
  #Slice subset rows using their position, and rep() replicates values
  #So it takes the first row and replicates it once,
  #before jumping to the next row.
  slice(rep(1:n(), each = 2)) %>%
  mutate(time = 1:n()) %>%
  ungroup()
```

Remove inactive players

During the specific football drills, some players were inactive and sat on the sideline. Instead of manually removing the correct players during each drill a short function was applied that detects inactivity during the drill. This was done by removing players which had more than 90% of FMP to be *Very Low Intensity* and *Low Intensity*. After applying the function the results would be verified with the video material to see if the correct athletes was detected. It is therefore important to mention that the 90% threshold is specific and arbitrary to this data collection and is not a general threshold for activity recognition.

```
removeInactivePlayers <- function(data){

  #Create unique ID for each athlete, for each drill and its iteration
  data$ID <- paste0(data$athlete, data$drillType, data$drillCounter, sep = "_")

  InactivePlayer <- data
  #Recode the two categories Low intensity and very low intensity to one category
  InactivePlayer$movement_type <- recode(data$movement_type,
                                         "Low Intensity" = "Low Locomotion",
                                         "Very Low Intensity" = "Low Locomotion")

  #Calculate the percentage distribution of each category in drills of interest
  #Then filter
  findInactivePlayer <- InactivePlayer %>%
    filter(drillType != "downtime") %>%
    group_by(ID) %>%
    count(movement_type) %>%
    mutate(percent = n/sum(n)) %>%
    ungroup() %>%
    filter(percent >= 0.9 & movement_type == "Low Locomotion")

  data <- data %>%
    filter(ID %!in% findInactivePlayer$ID)



  return(data)

}
```

Technology Theory

This section will go through some of the existing technologies that have been used to catagorize physical activity and showcase their usecases and shortcommings. All the hardware technologies that have been used during this project were part of the Catapult Sports IMUs.

The Catapult Vector X7 consists of five different technologies; accelerometer, gyroscope, magnetometer, global positioning system (GPS) and a local positioning system (LPS). The first three technologies when put together are also called an Inertial Measurement Unit (IMU). These are responsible for measuring linear accelerations and angular rotations, where the magnetometer is used to compute the heading of the sensor unit. The data from the different parts of the IMU is often sensor-fused to increase the validity of the measured data. This is due to some inherent shortcomings within the accelerometer and gyroscope when they are alone. The accelerometer has a lot of noisy measurements, but has no drift in the data as it is reliant on the earth's center of gravity. The gyroscope on the other hand has low noise, but it will drift over time. The sensor-fusion is a way to combine these measurements to make it more reliable with no drift and higher signal-to-noise ratio, while also calculating the unit's orientation (Kelly & Sukhatme 2010, Li et al. 2019). The filters for applying these fusions is most commonly the Kalman filter or the complementary filter (Valenti et al. 2015, Roell et al. 2018).

In addition, it is generally adviced to use a detrend algorithm to remove the trends (eg: drift) that can occur in the accelerometer and gyroscope data (Li et al. 2019). Though, it is not known if Catapult Sports does this prior to calculating the FMP categories. However, an earlier semester thesis investigated the impact of a detrend algorithm on the raw data, and as can be seen in figure 1 the detrend had very low impact, which suggests the proprietary kalman filter used, might have taken care of the drift (Lentz-Nielsen, 2021).

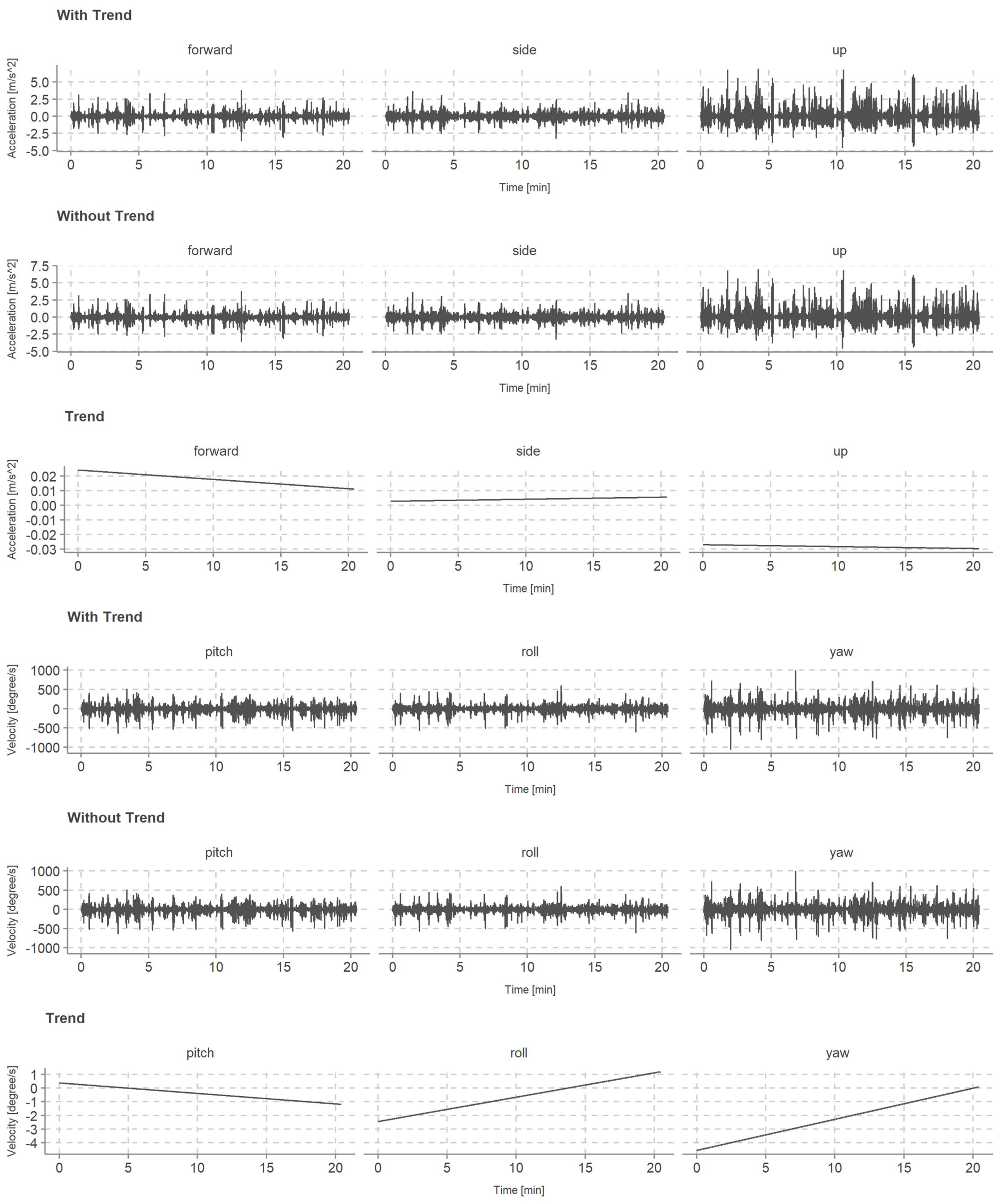

Figure 1: First tree plots shows the accelerometer data in forward direction, first shows the raw data, second plot shows the data when when the trend is removed, and third shows the differentiation between them. Same goes for the last 3 plots which represents the gyroscope data (Lentz-Nielsen, 2021).

The GPS is among the most used system in sports tracking (Aughey 2011). Catapult Sports GPS, samples at 10Hz and determines distance from positional differentiation and velocity from the Doppler shift technique. It has shown to have great validity for outdoor team sports with a sampling frequency of minimum 10Hz (Scott et al. 2016, Johnston et al. 2014). Furthermore, Petersen et al. (2009) also found that the reliability of the GPS was independent on the time of day, which is essential as sports teams train at different times in outdoor conditions. The biggest drawbacks for the GPS system is its reliance on satellite connection which means that indoor sports, or closed off stadiums will not be able to utilize the technology properly. An alternative is the LPS, which works in the same manner as a GPS system, except it connects to locally positioned satellites (see figure 2). The Catapult Sports Clearsky T6 have been validated and shows low errors in optimal settings relative to a motion capture system. Though, the measurements for instantaneous speeds is not valid (Luteberget et al. 2018). The other drawback for the GPS which is also found in the LPS is the lack of eucledian vectors when measuring velocity. Consequently, the velocity vectors does not have a direction, which linear accelerometer data from accelerometers does.

Figure 2: Catapult Sports Clearsky LPS satellite.

Signal filtering

The FMP is based on a propitiatory algorithm within Catapult Sports, which is not openly available. Consequently, the signal processing procedure of the IMU data is not known and can not be altered. This is an academic issue, as it makes it impossible to replicate or even verify the decisions that have been made in the algorithm. Even in the academic literature cut-off frequencies for signal filtering are often arbitrarily selected which may cause issues as each data set is highly dependent on the data collection method and the participating subjects as these can effect the artefacts showing in the data. Furthermore, there are several different ways of calculating a frequency cut-off which all results in different values (Fazlali et al. 2020).

Wundersitz et al. (2015) found the Minimax S4 (early version of Catapult Sports IMU product) positioned on the trunk to be valid in team sports when sampling at 100Hz and using a zero-lag low-pass butterworth filter with a cut-off of 12Hz. Where, Miller et al. (2022) investigated postural perturbations when walking on a treadmill, also using an IMU on the trunk. They found that a cut-off frequency of 35 ś 10 Hz for linear accelerations and 26 ś 7 Hz for angular velocity to be the best when basing it on the 99% energy spectrum analysis. These differences in chosen cut-offs indicates a need to investigate the frequency domain for each data collection as it can depend on athletes, practices activity and hardware.

Figure 3 clearly shows the implications of different cut-off frequencies. These filters have been made with a 2nd Order zero-lag butterworth filter with 35-, 12-, and 6Hz filters to showcase the difference in the time domain. Because the filters are not openly available for propitiatory software it is important to always declare the software and firmware version used in data collections, as the approach can change without the user knowing.

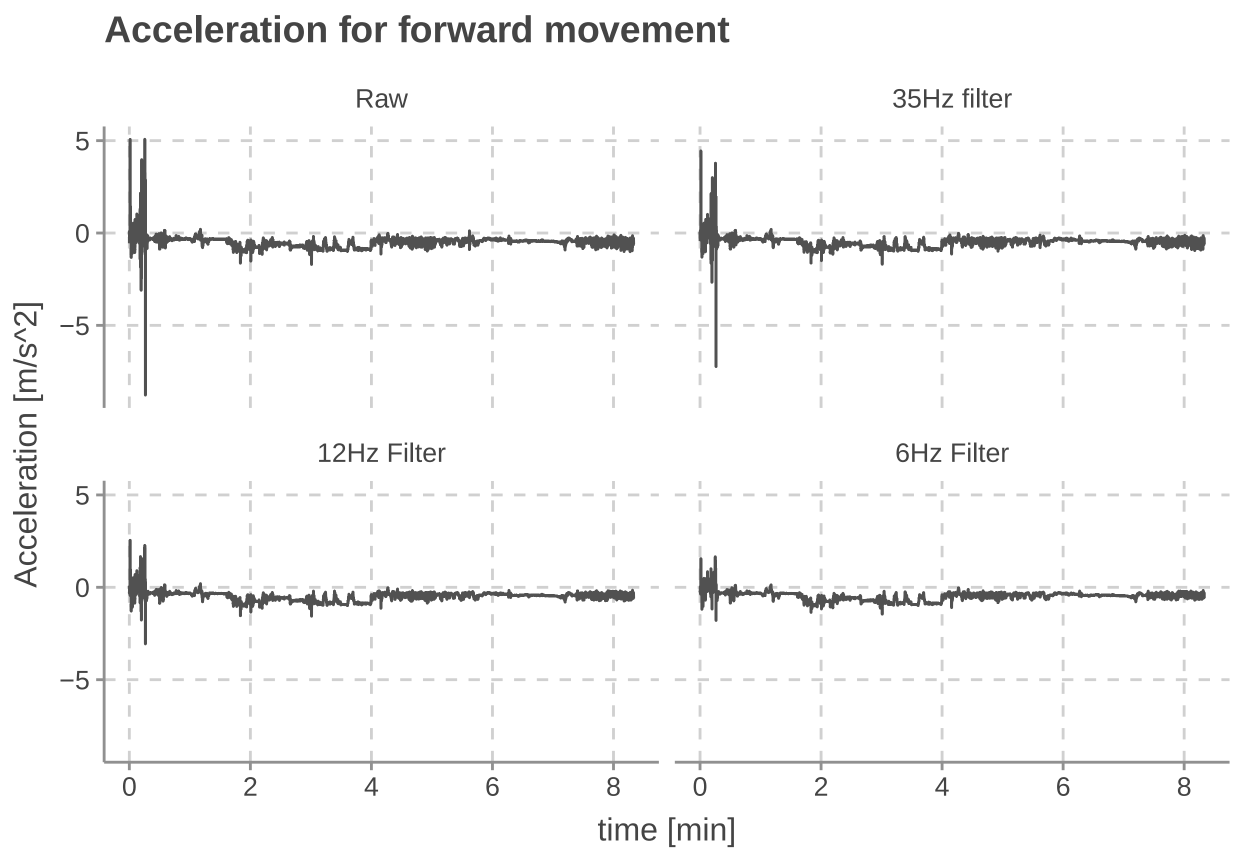

Figure 3: Example of the linear acceleration data for sagital movement. Showing the results of a zero-lag 2nd order butterworth filter with three different cut-offs relative to the raw data.

Interrater reliability

To measure the agreement between the FMP and the executed drills an interrater reliability test is needed. One of the more simple tests for this, is the overall percentage agreement test:

$$100 \ \frac{Agreement}{Total}$$

Though, the percentage agreement test does not take into account the chance for random agreements occuring. For this reason cohen's kappa is the more commonly used test for categorical variables in social and medical sciences (Chicco et al. 2021).

$$p_e = \frac{1}{N^2} \; n_{k1} n_{k2}$$

$$= \frac{p_o \;\; p_e}{1 \;\; p_e}$$

p_o is the observed agreement, p_e is the hypothetical probability for the chance that agreement occurs, where N is observations and n_{nki} is the number of times a rater predicted k.

However, according to Delgado & Tibau (2019) and Chicco et al. (2021), Kappa has issues with imbalanced datasets in machine learning classifications and suffers from the *prevalence paradox* and the *bias paradox*. The *prevalence paradox* derives when p_e among raters is high and even high values of the relative observed agreement results in low Kappa values. The *bias paradox* derives if the marginal distribution of the confusion matrix is imbalanced which produces higher Kappa values. **Therefore it has been recommended to start using Matthews Correlation Coefficient as an addition to Cohen's Kappa as performance metrics for classification problems within machine learning. These recommendation is based on how these metrics are calculated based on a confusion matrix and are not directed at interrater agreement tests.** Nonetheless, the formula for Cohen's kappa for interrater agreement and for 2 x 2 confusion matrix predictions produces the same results, as can be seen in the following example:

		True	True
		yes	no
Predict	yes	25	10
Predict	no	15	20

This can also be seen as:

		True	True
		yes	no
Predict	yes	True Positive (TP)	False Positive (FP)
Predict	no	False Negative (FN)	True Negative (TN)

The *True* and *predict* can be seen as two seperate raters, though for the controlled drills, from this thesis, we know the objective truth. If we calculate the agreement test for Kappa based on the aforementioned formula we get the following:

$$\begin{aligned}
p_o &= (25 + 20)/70 \\
p_{yes} &= ((25 + 10/70) \;\; ((25 + 15)/70) \\
p_{no} &= ((15 + 20)/70) \;\; ((10 + 20)/70) \\
p_e &= p_{yes} + p_{no} \\
&= \frac{p_o \;\; p_e}{1 \;\; p_e} \\
&= 0.285714
\end{aligned}$$

The formula for Kappa in binary classifications for confusion matrices:

$$\begin{aligned}
&= \frac{2 \;\; (TP \;\; TN \;\; FN \;\; FP)}{(TP + FP) \;\; (FP + TN) + (TP + FN) \;\; (FN + TN)} \\
&= \frac{2 \;\; (25 \;\; 20 \;\; 15 \;\; 10)}{(25 + 10) \;\; (10 + 20) + (25 + 15) \;\; (15 + 20)} \\
&= 0.285714
\end{aligned}$$

This was just an example of the calculations, a simulation is created that shows it is always the same to the fourth digit in sub-section *Simulation example of the two kappas.*

As stated previously, in machine learning both Cohen's Kappa and Matthews Correlation Coefficient (MCC) has been used in the literature as performance metrics. Both of them are reliant on a confusion matrix and even though (to the best of the authors knowledge) the MCC has never been used for interrater reliability test, it seems transferable to interrater agreement tests, as the only difference between MCC and Cohen's Kappa is their use of either the geometric or the harmonic mean, respectively.

$$MCC = \frac{TP\ TN\ FN\ FP}{(TP + FP)\ (FP + TN) + (TP + FN)\ (FN + TN)}$$

Therefore, it was the intention to use the MCC for the agreement test in this study. The original plan was to have three trained raters to label the controlled and specific drills for validation purposes. However, because of unforeseen circumstances this was not possible. As a result only the controlled drills were used for validation testing where each drill only had one objective truth. Because of this, each controlled drill would only have one category, eg: Linear locomotion or non-linear locomotion, which would result in a zero in atleast one of the four quadrants in the confusion matrix and the MCC and kappa would not be computed. Therefore, the percentage-agreement test was used as a substitute.

Simulation example of the two kappas

```r
kappaAgreement <- function(TP, FP, TN, FN){
  #agreement calculation based on the interrater-agreement
  #formula for cohens kappa
  summ = TP+FP+TN+FN

  po = (TP+TN) / summ
  pyes = ((TP+FP)/summ) * ((TP+FN)/summ)
  pno = ((FN+TN)/summ) * ((FP+TN)/summ)
  pe = pyes + pno
  k = (po - pe) / (1-pe)

  return(k)
}
```

```r
kappaClassification <- function(TP,FP,TN,FN){
  #binary classification formula for cohens kappa
  ktop = 2*(TP*TN - FN * FP)
  kbot = (TP+FP)*(FP+TN)+(TP+FN)*(FN+TN)
  k = ktop / kbot

  return(k)
}

#creating a dataframe
resultDf <- tibble(agreement = NA,
                   classification = NA)

#100 iterations with different TP,FP,TN,FN values
for (i in 1:101){
#Setting a seed to make it replicateable, but still changing the seed
#for each iteration, so it won't produce identical results

set.seed(123+i)
#random number generation
TP = as.integer(runif(1, 10, 50))
FP = as.integer(runif(1, 10, 50))
TN = as.integer(runif(1, 10, 50))
FN = as.integer(runif(1, 10, 50))

result1 <- round(kappaAgreement(TP,FP,TN,FN), 4)
result2 <- round(kappaClassification(TP,FP,TN,FN),4)

resultDf <- rbind(resultDf, result1, result2)
}

#removing the first NA values, and removing duplicate rows
```

```r
resultDf <- resultDf %>%
  distinct() %>%
  na.omit()

#Boolean test, if TRUE it means the values are the same.
resultDf$agreement == resultDf$classification
```

```
##   [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [76] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##  [91] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Machine learning

This section will go through some of the machine learning techniques that have been used in the project. Primarily; Feature Engineering, Feature Selection and Modelling procedures.

Feature Engineering

The only new features that were made was based on an N-gram approach. N-gram is a method within natural language processing that pairs adjacent words into categories, where N is the number of words of interest (Silge & Robinson 2017). For example, given the following sentence: "I bought a white house", using a bigram (2-gram) approach would result in the following four bigrams: "I bought", "bought a", "a white", "white house". This can be extended to trigram (3-gram), quadgram (4-gram) and so on.

The relative distribution of FMP, bigram, trigram and quadgram were used for each athlete in respect to each drill and its iteration. These distributions was used to investigate differences between drills while also being used for the predictive modelling. Function to calculate the FMP can be seen in following the codeblock.

```r
calculatengram <- function(dataframe, ngram){

  #Creating an empty dataframe
  allGrams <- tibble()

  #Iterating through the athletes in the dataframe, and selecting
  #one athlete at a time. Also creating new variable the can identify
  #the correct athlete to the drill and the iteration of that drill
  for (i in unique(dataframe$athlete)){
    subject <- dataframe %>%
      filter(athlete == i) %>%
      mutate(drillID = paste(drillType, drillCounter, sep="_"))



    #Filter based on the newly created drill ID and calculate the n-gram
    for (j in unique(subject$drillID)){
      subject_drill <- subject %>%
        filter(drillID == j)

      #Create new dataframe with the n-grams.
      tempTibble <- tibble(
        longNewFMP = paste(subject_drill$newFMP, collapse = " ")) %>%
        tidytext::unnest_tokens(gram, longNewFMP, token = "ngrams", n = ngram)

      #Insert NA at ngram-1 rows to have correct length
      naValues <- tibble(naTest = rep(NA, ngram-1))
      tempTibble <- bind_rows(naValues, tempTibble) %>%
        select(gram)
      tempTibble$athlete = i

      allGrams <- rbind(allGrams, tempTibble)
    }
  }
  return(allGrams)
}
```

Feature selection

Calculating the relative distribution of FMP and three different n-grams results in a large feature space (Number of features > number of observations). A reduction of the feature space was needed to remove the noisy features and keep the once that assisted in prediction. “Variable Selection Using Random Forest” (VSURF) utilizes random forests to choose which features to keep or omit (Genuer et al. 2010, 2015). By removing the noisy features not only does it increase the prediction accuracy, but it will also decrease the final computational time of the modelling algorithm.

VSURF has several steps to classify the most important features. The first step orders the features from the most important to the least important. Importance are based on permutation instead of gini impurity because of higher reliability (Genuer et al. 2010, Hooker et al. 2021). Hereafter, the standard deviation is computed for the variable importance and a “Classification and Regression Tree” (CART) model calculates the minimum prediction value which are used as a threshold to eliminate features.

The second step select features that are highly related to the response variable. The remaining features from the first step, was selected in order of importance based on a forward step-wise method, which means it adds one feature at a time. Then 25 random forests was computed to calculate the out-of-bag error and the nested models with the lowest error was selected. The last step selected variables based on a good parsimonious prediction of the response feature. Like before a forward step-wise method was used to select features above a threshold. For this step the threshold was the error decrease needed to be significantly greater than the averaged variation from noisy variables (Genuer et al. 2010, 2015).

Data resampling

Resampling is a technique used to equalize the distribution between the classes of interest prior to training a machine learning algorithm., this can be an important tool since a skewed distribution can bias the trained algorithm. Two of the most simple ways of resampling is simply to upsample the classes with fewer occurences to the one with the most, or downsample the classes with more occurences to the one with the least. Up- and downsampling randomly dublicates or removes data in the dataset (Burkov 2019).

A more complex upsampling method was used during this study. He et al. (2008) describes an *Adaptive Synthetic Sampling Approach* (ADASYN) to upsample data which is based on Chawla et al. (2002) work on a *Synthetic Minority Over-sampling Technique* (SMOTE). The basic idea behind SMOTE is to synthesize the minority classes using an K-nearest neighbor algorithm and insert the new synthesized classes along a specific line segment for that class (see figure 4).

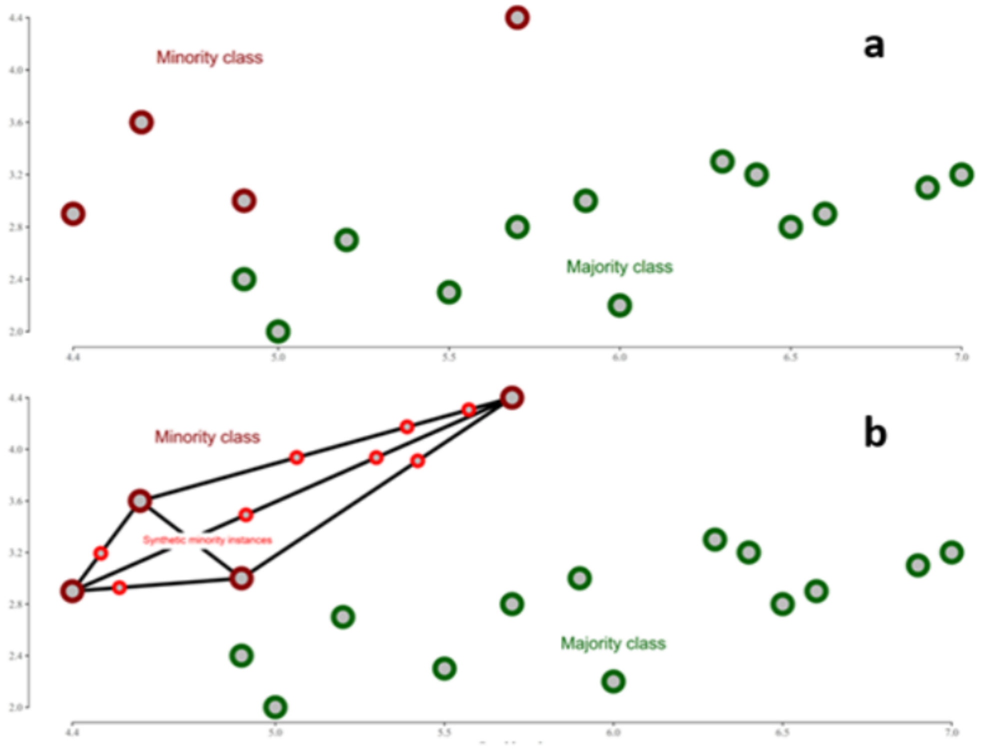

Figure 4: Concept of SMOTE showing how the synthesized data points is placed along the line segment.

ADASYN is expanding on this by focusing on a weighted distribution for each data point from the minority class, relative to how difficult that data point is distinguishable from the other classes. ADASYN then upsamples the data points with the highest weighting, eg. hardest to distiguish (see figure 5)

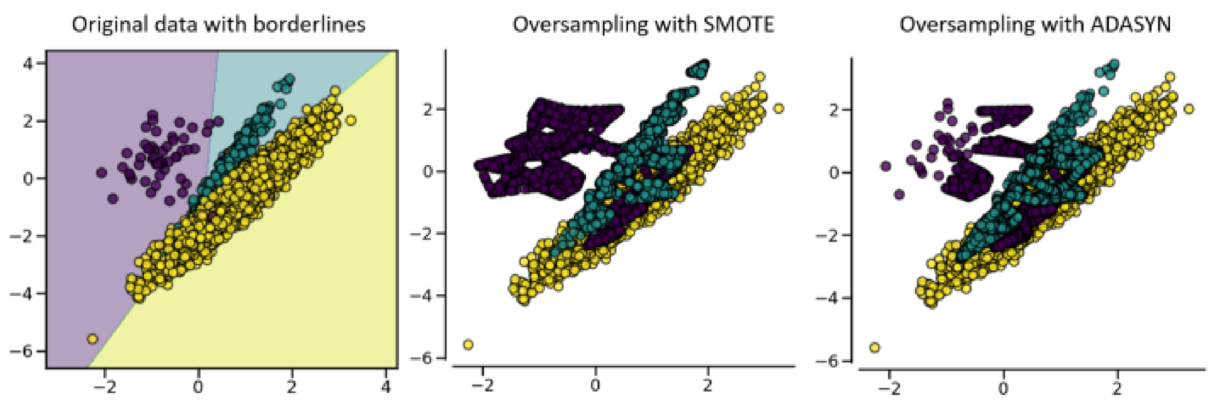

Figure 5: Difference between SMOTE and ADASYN upsampling. Notice how the purple class in between the green and yellow is upsampled more for ADASYN.

Hyperparameter tuning

Hyperparameter tuning is the tuning of the learning parameters in a machine learning model to make it as good as possible for the given training data without overfitting. Examples of hyperparameters could be, number of trees or depth of each tree in a random forest model, or the polynomial degrees in a support vector machine. To tune the hyperparameters a k-fold cross validation method was chosen for its small bias (Kuhn 2014). K-fold cross-validation splits the data into K-folds where the model will be trained on the k-1 folds and validated with the fold that was left out. Hereafter, the folds rotate so a model will be trained ten times and validated to ten different folds (Santos et al. 2018).

After the data is split up in k-folds, normally a grid search strategy would be used to identify the best hyperparameters. However, this method has computational shortcomings as it treats each parameter solution equally before attempting to do the next combination. Therefore, a futility analysis approach was conducted with repeated-measures ANOVA statistics to eliminate parameter combinations that would be unlikely to yield the best result. The benefit of this approach is decreased computational time (Kuhn 2014).

Random forest

Random forest models consist of an ensemble of trained decision trees. To do this a technique called Bootstrap aggregation (bagging) is used. This results in several decision trees being trained based on a bootstrapped (a random resampling) set of the data set. The number of trees trained depends on the hyperparameter setting as mentioned earlier. All the classifications from the trees is then aggregated based on the total number of class predictions across the trees (Burkov 2019, Géron 2019)

Prozone and intensity zones

Maybe present the litterature on the velocity intensity zones, to increase the strength of my argument that it is "arbitraried" picked in the litterature.

explain ngram, pernetage test, mcc, kappa, and other features and test parameters that have been used or considered used.

alot of plots and features and their impact

Explain the procedure of creating features model it and see how different features impacts the predictions.

Bibliography

Aughey, R. J. (2011), 'Applications of gps technologies to field sports', *International journal of sports physiology and performance* **6**(3), 295–310.

Burkov, A. (2019), *The hundred-page machine learning book*, Vol. 1, Andriy Burkov Quebec City, QC, Canada.

Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T., Tröster, G., del R. Millán, J. & Roggen, D. (2013), 'The opportunity challenge: A benchmark database for on-body sensor-based activity recognition', *Pattern Recognition Letters* **34**(15), 2033–2042.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'Smote: synthetic minority over-sampling technique', *Journal of artificial intelligence research* **16**, 321–357.

Chicco, D., Warrens, M. J. & Jurman, G. (2021), 'The matthews correlation coefficient (MCC) is more informative than cohen's kappa and brier score in binary classification assessment', *IEEE Access* **9**, 78368–78381.

Dehghani, A., Glatard, T. & Shihab, E. (2019), 'Subject cross validation in human activity recognition'.

Delgado, R. & Tibau, X.-A. (2019), 'Why cohen's kappa should be avoided as performance measure in classification', *PLOS ONE* **14**(9), e0222916.

Fazlali, H., Sadeghi, H., Sadeghi, S., Ojaghi, M. & Allard, P. (2020), 'Comparison of four methods for determining the cut-off frequency of accelerometer signals in able-bodied individuals and ACL ruptured subjects', *Gait & Posture* **80**, 217–222.

Genuer, R., Poggi, J.-M. & Tuleau-Malot, C. (2010), 'Variable selection using random forests', *Pattern Recognition Letters* **31**(14), 2225–2236.

Genuer, R., Poggi, J.-M. & Tuleau-Malot, C. (2015), 'Vsurf: an r package for variable selection using random forests', *The R Journal* **7**(2), 19–33.

Géron, A. (2019), *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, " O'Reilly Media, Inc.".

He, H., Bai, Y., Garcia, E. A. & Li, S. (2008), ADASYN: Adaptive synthetic sampling approach for imbalanced learning, *in* '2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)', IEEE.

Hooker, G., Mentch, L. & Zhou, S. (2021), 'Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance', *Statistics and Computing* **31**(6).

Johnston, R. J., Watsford, M. L., Kelly, S. J., Pine, M. J. & Spurrs, R. W. (2014), 'Validity and interunit reliability of 10 hz and 15 hz GPS units for assessing athlete movement demands', *Journal of Strength and Conditioning Research* **28**(6), 1649–1655.

Kelly, J. & Sukhatme, G. S. (2010), 'Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration', *The International Journal of Robotics Research* **30**(1), 56–79.

Kuhn, M. (2014), ‘Futility analysis in the cross-validation of machine learning models’, *arXiv preprint arXiv:1405.6974* .

Li, S., Gao, Y., Meng, G., Wang, G. & Guan, L. (2019), ‘Accelerometer-based gyroscope drift compensation approach in a dual-axial stabilization platform’, *Electronics* **8**(5), 594.

Luteberget, L. S., Spencer, M. & Gilgien, M. (2018), ‘Validity of the catapult ClearSky t6 local positioning system for team sports specific drills, in indoor conditions’, *Frontiers in Physiology* **9**.

Miller, E. J., Sheehan, R. C. & Kaufman, K. R. (2022), ‘IMU filter settings for high intensity activities’, *Gait &amp; Posture* **91**, 26–29.

Petersen, C., Pyne, D., Portus, M. & Dawson, B. (2009), ‘Validity and reliability of GPS units to monitor cricket-specific movement patterns’, *International Journal of Sports Physiology and Performance* **4**(3), 381–393.

Roell, M., Roecker, K., Gehring, D., Mahler, H. & Gollhofer, A. (2018), ‘Player monitoring in indoor team sports: Concurrent validity of inertial measurement units to quantify average and peak acceleration values’, *Frontiers in Physiology* **9**.

Santos, M. S., Soares, J. P., Abreu, P. H., Araujo, H. & Santos, J. (2018), ‘Cross-validation for imbalanced datasets: avoiding overoptimistic and overfitting approaches [research frontier]’, *ieee ComputatioNal iNtelligeNCe magaziNe* **13**(4), 59–76.

Scott, M. T., Scott, T. J. & Kelly, V. G. (2016), ‘The validity and reliability of global positioning systems in team sport’, *Journal of Strength and Conditioning Research* **30**(5), 1470–1490.

Silge, J. & Robinson, D. (2017), *Text mining with R: A tidy approach*, " O’Reilly Media, Inc.".

Valenti, R., Dryanovski, I. & Xiao, J. (2015), ‘Keeping a good attitude: A quaternion-based orientation filter for IMUs and MARGs’, *Sensors* **15**(8), 19302–19330.

Wundersitz, D., Gastin, P., Robertson, S., Davey, P. & Netto, K. (2015), ‘Validation of a trunk-mounted accelerometer to measure peak impacts during team sport movements’, *International Journal of Sports Medicine* **36**(09), 742–746.