James McGowan

Registration number 100277997

2023

Using Time Series Classification with a smartphone to identify a basketball players shot location and type

Supervised by Dr Jason Lines

University of East Anglia
Faculty of Science
School of Computing Sciences

Abstract

With the ever-growing demand for health tracking technology and increased growth of basketball within the UK, it presents a question: can time series classification be used to encourage more players to join the sport?
The main aim of this report is to discover if it is possible to accurately predict a player's shot type and location on a basketball court using just gyrometer and accelerometer data recorded from a smartphone. During this experiment, standard classification models as well as Time Series specific solutions were implemented to better compare what the best method for solving this problem would be. These models were compared using common metrics, such as accuracy, balanced accuracy, run time, etc. At the end of the experiments with the context of all results, it was found that this problem is possible and can be done to a good level using a state-of-the-art classifier.

Acknowledgements

I would like to thank this projects supervisor Dr Jason Lines for his invaluable insight and guidance over the year. I would also like to thank Sophia Walker for her unwavering encouragement and support.

Contents

List of Figures

List of Tables

1. Introduction

The UK suffers from a physical inactivity epidemic, it is responsible for 1 in 6 deaths and is estimated to cost the UK government £7.4 billion annually (GOV.UK, 2021). Therefore, it is an essential task for the UK government to encourage exercise within the country. With modern day technology being more accessible than ever before it has never been so easy to track your health and well being. As of March 2020 87% of people in the UK own a smartphone [1] and in 2023 almost all smartphones come with some form of built in health tracker that uses a combination of accelerometer, gyrometer and GPS data to track step counts (Guiry JJ, 2014). With movement tracking becoming so accessible it proves to be a suitable method for tracking exercise progress and by extent encouraging exercise by seeking further improvement.

A popular method for getting regular exercise is to engage in some form of sport. One possibility is basketball which is an ever growing sport in the UK and a recent study by the Department of Culture and Media Sport found basketball to be the second most played sport by children aged 11 to 15 years in the UK [2]. For players looking to improve their game adopting a way of tracking their shots could yield positive improvement to their game and further their passion to continue playing.

1.1. Aims

The goal of my project is to utilize a combination of experimental and observational elements, focusing on time series classification (TSC). The objective is to effectively model and predict the types of shots taken on a basketball court, along with their corresponding locations, using gyrometer and accelerometer data.

For my data-set I aim to gather 250 unique shots from 5 different locations on the court, with a suitable collection of shots. It is important to prioritise this stage as without a suitable data-set this project can not be complete. I aim to develop a way of isolating a short window in which the main movement of each shot takes place. An example[3] of this shot motion can be seen in Figure 13 of Appendix A. Once each shot has been identified this period of time I will gather both the acceleration and gyration data for

[1] *https:// cybercrew.uk/ blog/ how-many-people-own-a-smartphone-in-the-uk/*

[2] *https:// www.bbldaily.com/ news/ why-british-basketball-is-on-the-rise*

[3] https://www.shotur.com/stephen-curry-shooting- form-training-season-2-test-4-video/

each direction. When all of this data is combined it will form the backbone of this experiment.

My next aim is to manipulate this data and form as many sub data-sets as possible to be used in the classification process to evaluate what information is essential and what is not. From there my aim is to train and test a range of classifiers against this data to discover if recording time series data when shooting can be an effective way of tracking a players shot location.

1.2. Objectives

There are key objectives that must be met for this project to be considered a success, but with limited time for development I have had to prioritize more essential requirements over others. To do this I have used the MoSCoW technique (Kravchenko et al., 2022).

Must

- Produce a usable large data-set with 50 shots from each location.
- Proof of concept, an experiment to show whether or not this problem is possible.
- Test the complete data-set against a range of classifiers and evaluate which classifier has the best performance.

Should

- Test whether data normalization yields more accurate results.
- Compare the performance of KNN with DTW and without.
- Experiment with shot locations to judge which combination is the most accurately classified.

Could

- Test whether the gyrometer or accelerometer values provide more accurate classification.
- Compare the range of classifiers with different time windows of shot readings to see how the results are effected (e.g 4 seconds vs 5 second windows).

- Experiment with a range of parameters to achieve optimum performance for each classifier.
- Compare axis to see which direction has the biggest influence on performance.

Wont

- Use a smart watch to record shot data at the same time as a smartphone to compare the accuracy of data collection.
- Introduce another player and compare how accurately their shots are classified against my own.
- Compare if using a left handed shooter vs a right handed shooter would effect results.
- Compare the range of classifiers with different frequency of readings (e.g taking a reading every 0.2 vs 0.5 seconds).

2. Background

For this experiment there are many possible factors to consider when undertaking this project. Therefore, it is essential that I am well read and understand each component of the project thoroughly. In this section it is my goal to clearly explain the key concepts that make up this project.

2.1. Machine Learning

IBM defines Machine learning (ML) to be a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way humans learn, gradually improving its accuracy [4]. Machine learning algorithms are powerful pattern recognition tools capable of seeing hidden relationships within data that would otherwise be impossible for a human to find. For movement tracking being able to find the slightest similarities between activities is crucial for correct classification.

[4] *https://www.ibm.com/topics/machine-learning*

2.2. Time Series Classification

Time Series Classification (TSC) can be defined as a sub-section of machine learning where problems involve training a classifier on a set of cases. Each one of these cases is a set of real value attributes with a class label (Bagnall et al., 2017). Time series as described by Lines et al. is a sequence of data that is typically recorded in temporal order at fixed intervals of time. In the context of this experiment this will be recordings of the movement (x, y, z) of a users arm during the shot motion followed by the class label referring to the position on the court that the movement took place.

2.3. Previous activity recognition experiments

Similar experiments have been previously attempted using time series classification to predict a range of different activities. It is important to look over past works and get an understanding of the process that goes into this type of experiment. By reviewing past works I can achieve a better understanding of the methods and approaches for developing this project.

For instance one experiment involved using a smartphone and smartwatch to collect time series data of light sport exercises and to predict the movement using a K-Nearest Neighbor with Dynamic Time Warping classifier (Nurwanto et al., 2016). The authors method of exercise tracking if functional would in theory help to achieve their goal of encouraging health and fitness without the need of extra equipment. The movements that the experiment were based on differ heavily from my own, the process of recording each movement with a sensor attached to the users wrist will be the same. There are short comings in the experiment by Nurwanto et al. as there is no great detail on the process of data recording, it is said they repeat the same exercise twice, once with time, once without. However, the sample size they use for this experiment is rather small using only 10 repetitions for each movement. Still, it is an interesting question to see if there is a difference between fixed time and unfixed recordings. For my experiment however I do not think I will replicate this method of recording data as the shooting motions speed depends more on the distance at which the user is shooting from rather than how fast they perform the movement, for example trying to fix the time frame of each shot may effect how accurately the ball is thrown. The results of this experiment were successful achieving an accuracy of 76.67% for push-ups, 80% for sit-ups, and 96.67% for squat jump. From the strength of these results it proves to me that KNN with DTW can be a

strong classifier for TSC and will be utilised in my own development.

In my initial progress report I found that it was easiest to isolate each shot by focusing on just the acceleration in the x direction. Therefore, I have found that (Casale et al., 2011) is a relevant piece of work to my own. In this study Casale et al. attempts to recognise everyday activities by using a Random Forest classifier on accelerometer data gathered by a smart watch. Similarly to Nurwanto et al. the movements are different to that of which my experiment focuses on. In addition, the method for data gathering is also different as the user wore a chest harness for extended periods of time while performing their activities, where as I will just be focusing on the motion of my arm during the shooting form. In the conclusion of (Casale et al., 2011) they discuss how they were able to achieve results of up to 94% accuracy using the Random Forest classifier. With results this promising it is a good indication that accelerometer data works well with this approach and it will be interesting to see how this approach benefits my own work.

3. Methodology

3.1. Classification algorithms

Classification algorithms are a set of machine learning techniques designed to predict or classify data into predefined classes or categories based on their features or attributes (Priyam et al., 2013). For my project I have identified a range of classifiers that use various techniques to accurately predict a class based on the features provided. The majority of these classifiers are general algorithms and two are specifically intended for time series data.

3.1.1. Decision Trees

A Decision Tree Classifier (DTC) is a multi-stage approach to decision making. In any multistage approach, the fundamental concept is to divide a complicated decision into a combination of several simpler decisions, with the expectation that the resulting solution would closely resemble the desired outcome (Safavian and Landgrebe, 1991). Each tree is made up of decision nodes and leaf nodes, with decision nodes containing two or more branches that eventually conclude with a leaf node that represents a decision. The splitting criteria for branches is based upon a how pure each possible split will become.

The Gini index is a method of determining purity of a specific class when splitting upon a particular attribute (Tangirala, 2020). The formula for this algorithm can be seen below in Equation 1.

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} p_i^2 \tag{1}$$

Decision trees suffer from being sensitive to small variations in data, for my project this may effect results as the movements performed are not always precise and human error can result in small differences between the same types of shots. However, despite these limitations it will be useful to see the performance of this classifier to better evaluate approaches tailored for TSC.

3.1.2. Naive Bayes

The Naive Bayes classifier is a supervised machine learning algorithm and according to IBM it is part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs given a class or category [5]. The algorithm is based upon Bayes theorem with an assumption of independence among features. It bases predictions on the probability for each class as if each sample belongs to that class, then the class with the highest probability is assumed to be the correct class (Leung et al., 2007), this is demonstrated in Equation 2.

$$P(H|X) = \frac{P(X|H) \cdot P(H)}{P(X)} \tag{2}$$

The Naive Bayes classifier is good for this project as it performs training and predictions quickly and it is simple to implement. However, due to the nature of how Naive Bayes works it is not well suited for time series problems as each data feature is not independent of each other. I will still be implementing this algorithm however as it will be interesting to see how its performance compares to algorithms more suited to these problems.

[5] IBM Naive Bayes: *https://www.ibm.com/topics/naive-bayes*

3.1.3. Random Forest

The Random Forest classifier is an ensemble of tree classifiers where each tree is built using a random subset of training data. Each tree then casts a vote to for the most likely class to classify any input data (Pal, 2005).

Random Forest is an easy to use machine learning technique and is often effective without the need for hyper-parameter tuning as the classifier is less prone to overfitting. Random Forest is often used in time series problems and should prove to be one of the stronger general classifiers used during these experiments.

3.1.4. Support Vector Machines

Support Vector machines (SVM) aims to classify data based on an optimal "hyperplane". A "hyperplane" is a decision boundary that maximises the margin , which is the distance between the hyperplane and the closest data points of each class (Pisner and Schnyer, 2020).

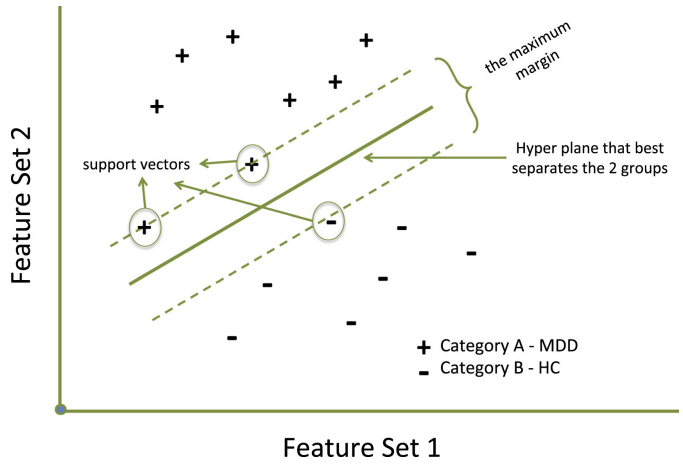

Figure 1: Example of an optimal SVM hyperplane as seen in (Schnyer et al., 2017).

SVM uses a technique called the kernel method which maps the input data into a higher-dimensional feature space so that it is easier to find linear separation. This allows the SVM to handle nonlinear relationships between data features. SVM's are widely used as they are effective at handling different types of data.

In this context the SVM may perform well as long as there is a clear distinction between the shot types allowing for the optimum "hyperplane" to be found.

3.1.5. ZeroR

In the experiment I will also be using the ZeroR classification method, this is purely for evaluation purposes as it will prove to be a good baseline to compare the results of the other classifiers to. ZeroR will perform the worst as it is a very simple classification technique and will choose the most common category all the time (Devasena et al., 2011).

3.2. Time Series specific algorithms

These final two algorithms should show the best performance in this experiment as they are both tailored to TSC.

3.2.1. Nearest Neighbor with DTW

The Nearest Neighbor algorithm is a commonly used classification and regression technique. The algorithm is best known for its simplicity and effectiveness (Bhatia et al., 2010). The algorithm predicts class based off of the distance between the test data and all of the training data. The algorithm finds the closest neighbours to the test data and uses a k value to identify how many of those nearest points vote on which class the test data belongs to [6]. The algorithm can calculate this distance between neighbours in different ways, the most common of which is to use Euclidean distance. The Euclidean distance is calculated as the square root of the sum of the squared differences between a test point and a training point, the formula for this can be seen below in Equation 3.

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3}$$

For TSC problems an alternative to Euclidean called Dynamic Time Warping (DTW) may be more suited. DTW is a an algorithm for measuring similarity between two temporal sequences, these sequences could differ in duration. An example[7] of the difference between Euclidean and Dynamic Time Warping can be seen below in Figure 2.

[6] *https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4*
[7] *https://www.databricks.com/blog/2019/04/30/understanding-dynamic-time-warping.html*

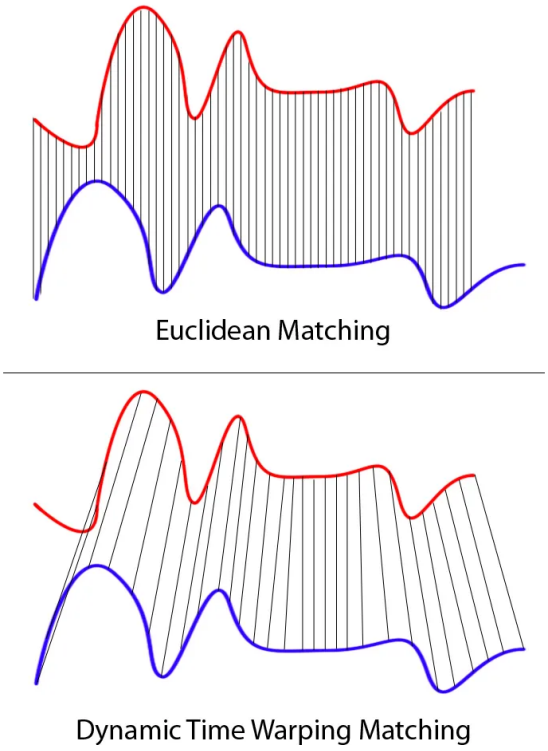

Figure 2: Euclidean distance vs DTW courtesy of databricks.com[7]

In the context of this project, having an algorithm that can help classify data of different lengths could be very beneficial as not all shots will take the same amount of time to record.

3.2.2. Rocket

ROCKET (RandOM Convolutional KErnal Transform)[8] is a state-of-the-art algorithm for TSC. Often algorithms with state-of-the-art accuracy have high computational complexity and are often unusable for large data-sets. However, ROCKET can achieve the same level of accuracy in just a fraction of the time[9]. The algorithm works by transforming the data using convolutional kernels and then passes this data to a linear classifier. The convolutional kernels are initialized with random length, weights, bias, dilation, and padding (Dempster et al., 2020). ROCKET uses a very large number of these ker-

[8] *https://www.sktime.net/en/stable/api_reference/auto_generated/sktime.classification.kernel_based.RocketClassifier.html*

[9] *https://pub.towardsai.net/rocket-fast-and-accurate-time-series-classification-f54923ad0ac9*

nels, with each kernel being convolved with each time series to produce a feature map. Each kernels feature map is used to produce two features per kernel: the proportion of positive values and the maximum value. The proportion of positive values decides how to weight a pattern captured by the kernel. This is the key element of ROCKET that contributes to its high accuracy. The Formula for the proportion of positive values (PPV) is shown below in Equation 4.

$$\text{PPV} = \frac{\sum_{i=0}^{n-1}(z_i > 0)}{n} \tag{4}$$

3.3. Data collection

For my experiment I have selected 5 positions on the court, where I will take 50 shots per location totalling 250 shots in my data-set. When choosing these locations I aimed to shoot from spots that I could confidently ensure I would make the shot or at least closely miss from, this would ensure that each shot showed a true representation of an in-game player doing the same movement. I also needed to ensure that these shots would not be too similar so that a classifier would be able to distinguish between them. The shots that I have selected have been mapped in Figure 12 of Appendix A. Each shots will follow the form demonstrated in Figure 13 of Appendix A, by following a consistent and proper form I can help ensure more consistent results. The data recording will take place indoors on a standard court using a regulation size 7 ball for all shots to try and negate factors such as the weight of the ball and wind speed from effecting the results.

For this experiment I will be using an iPhone 13 mini[10] to collect my movement data during the shooting process. Initially it was planned to record the movement data with a smartwatch, however due to limited access to suitable watches I have opted to record the data with a smartphone. In the future if development were to continue a smartwatch could be used to repeat the data collection process and test if that method of data collection yields any benefit. However it could be argued that due to the wide availability of smartphones it may be much more accessible for most audiences.

For recording the movement data, I plan to do each location one at a time. I will setup on the spot of where the shot takes place and continuously shoot the ball 50 times, with the help of an assistant to gather the rebound of the ball and pass it back to me so

[10] *https://www.apple.com/uk/shop/buy-iphone/iphone-13*

that I do not have to move position and effect the results. Both the starting position of each shot and end of each shots form with the tracking software running is shown in Figures 14 and 15 of Appendix A. Originally the tracking software that I planned to use was 'Sensor Data Recorder'[11] however, the software itself set a limit of up to 45 seconds per recording and this made the process of recording data very tedious along with complicating file management. The results from those initial recording sessions were discarded and the tracking software that I have chosen to use is the 'Physics Toolbox Sensor Suite'[12] as it allows for multiple sensor recordings to take place simultaneously. This is ideal as I want to obtain as much information about each shot as I can and in this case that would be the gyrometer and accelerometer values in all directions (x, y, z). The orientation and axis of the phones recordings[13] can be seen below in Figure 3.

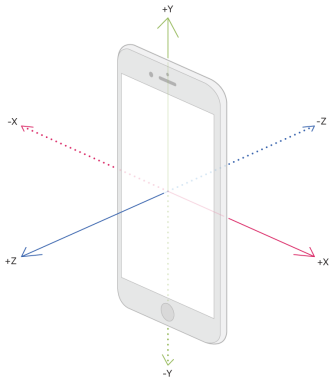

Figure 3: An example of the three axis recorded for an iPhone as shown in the Apple developer documentation .

By following through with this data gathering plan I have obtained 5 CSV files, one for each shot location. Each shot location contains a timestamp, acceleration values (x, y, z) and gyrometer values (x, y, z). An example of this raw data can be seen below in Figure 4 which shows the first initial readings for the layup location.

[11] *https://apps.apple.com/us/app/sensor-data-recorder/id1438400138*
[12] *https://apps.apple.com/us/app/physics-toolbox-sensor-suite/id1128914250*
[13] *https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events*

	time	ax	ay	az	wx	wy	wz
0	2023-02-08 13:49:32.3240	-0.17	0.48	0.39	0.04	-0.65	0.21
1	2023-02-08 13:49:32.3300	1.32	0.83	0.71	0.02	-0.15	0.35
2	2023-02-08 13:49:32.3400	1.32	0.83	0.71	0.03	0.30	0.32
3	2023-02-08 13:49:32.3500	1.81	0.82	0.77	0.04	0.53	0.25
4	2023-02-08 13:49:32.3600	1.18	0.74	0.61	0.05	0.55	0.26

Figure 4: A small sample of the initial raw Layup data.

Now that the data has been recorded it must go through pre-processing to extract features ready for the classifiers to train on.

3.4. Data processing

The method which I used for recording my data means that I have also recorded a large amount of inactivity in between shots while waiting for the rebound which needs to be removed. Initially during feature extraction I manually went through the data and found a few initial shots to use for example tests. From this process I found that acceleration in the x direction gave the most defined spikes for each shot and therefore making it the easiest metric to distinguish the movements. The best way to visualise the difference between shots was to graph them using the matplot library[14] along with pandas for the data-frames[15]. An example of these shots can be seen in Figure 5. The graph clearly shows a distinction that can be made with the human eye alone providing me with confidence to move forward with this data.

[14] *https:// matplotlib.org/ stable/ index.html*
[15] https://pandas.pydata.org/docs/

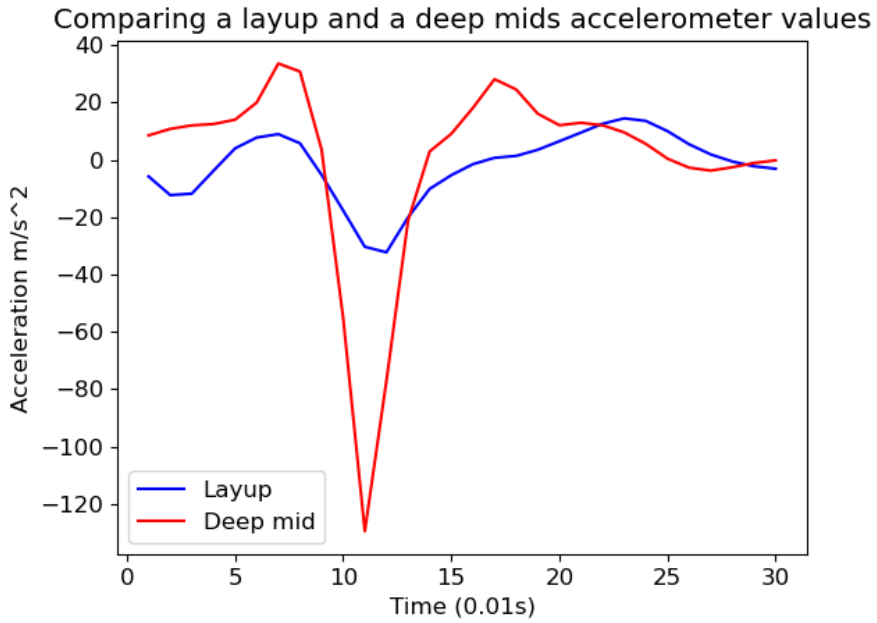

Figure 5: A graph comparing a layup and deep mid shot, made using matplotlib.

This method of isolating shots worked for a period of time, however it became very tedious and could result in inconsistencies if I continued using that method. To counter this I developed a function that would better sift through the data and isolate the shots. The pseudo-code for this function can be seen below in Algorithm 1 and works by taking in the shot type currently being sifted (i.e layup or deep mid), the threshold that all 50 shots in the specific data-set cross and the direction of shot that is being isolated. The function then isolates a window of 30 readings where the main movement of the shot takes place. When the threshold is crossed the function takes the 10 readings prior and the next 20 readings to make up the shot motion. A sample of a data-set collecting only the x acceleration and assigning the class can be seen below in Figure 6 with the 'shot_type' class ranging from 0 to 4.

	ax_1	ax_2	ax_3	ax_4	ax_5	...	ax_27	ax_28	ax_29	ax_30	Shot_type
0	4.65	4.65	3.70	-0.16	-5.66	...	1.97	1.11	-0.26	-1.98	0
1	9.97	7.27	3.87	1.02	-1.88	...	0.96	1.59	3.55	4.98	0
2	7.57	9.16	9.75	9.53	8.30	...	6.71	5.94	3.34	1.16	0
3	2.67	1.57	1.72	2.29	0.87	...	-4.47	-5.72	-6.74	-6.88	0
4	-14.91	-14.69	-8.39	-1.61	3.22	...	0.33	-1.36	-2.19	-2.59	0

Figure 6: An example of the processed data for acceleration only (x direction).

Algorithm 1 find_Shots

Require: shot_type, threshold, direction

```
i ← 0
shots ← {}
while i < len(shot_type['ax']) do
    current_shot ← {}
    if shot_type['ax'][i] < threshold then
        for j ∈ {0,1,...,29} do
            current_shot.append(shot_type[direction][i + j − 10])
        end for
        i ← i + 20
        shots.append(current_shot)
    else
        i ← i + 1
    end if
end while
return shots
```

3.4.1. Data normalization

As aforementioned in the objectives section I wanted to experiment with normalization in this project. To implement this I normalized the acceleration (x direction) data-set before training the classifiers. For this process I used the 'min max' scale from the 'sklearn' pre-processing package[16]. The algorithm works by reducing the current data point by the minimum value in the column then dividing by the difference between the max and min values in the column, the formula showing this is demonstrated in Equation 5.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{5}$$

[16]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.minmax_scale.html

3.4.2. Experimental data-sets

After the data collection and pre-processing stages are complete it leaves me with 13 data-sets ready for experimentation to begin, descriptions of each data-set can be seen in Table 1.

Data-set characteristics									
Data-sets	X gyro	Y gyro	Z gyro	X acc	Y acc	Z acc	Attributes	Instances	Classes
Ax				X			30	250	5
Ax_lay_deep				X			30	100	2
Gyro_only	X	X	X				90	250	5
Acc_only				X	X	X	90	250	5
Ax_no_free				X			30	200	4
Ax_no_key				X			30	200	4
Ax_no_key_free				X			30	150	3
Complete_data	X	X	X	X	X	X	180	250	5
Complete_data_normalized	X	X	X	X	X	X	180	250	5
acc_gyro_x	X			X			60	250	5
acc_gyro_y		X			X		60	250	5
acc_gyro_z			X			X	60	250	5
all_15_readings	X	X	X	X	X	X	90	250	5

Table 1: Details of the data-sets formed for experimentation.

3.5. Experimental plan

The ultimate aim of these experiments is to discover if tracking a basketball players shot location is even possible, and if so what is the most accurate method of doing so. The result is currently unknown and therefore I need a set of experiments that can prove or disprove this method with a high level of certainty. To achieve this level of certainty I need to thoroughly test as many different ways as possible requiring a variety of different comparisons. Although, there are up to 63 different combinations possible for the data-sets and testing them all would be very time consuming and out of the scope of this project meaning I will be selective of which data-set combinations I choose going forward. In addition, there are also a large variety of methods for evaluating the results and these will also have to be chosen with consideration.

3.5.1. Evaluation metrics

When necessary classifiers may use a 5-fold cross-validation as a method of parameter tuning based on the training data, before being exposed to the test data. The randomness

of folds will be consistent across the classifiers as they have all been seeded to 1. For evaluating each classifiers test performance I have selected a range of metrics as seen below:

- **Accuracy -** This is the most common evaluation metric for classification models. It is calculated by dividing the total number of correct predictions by the total number of predictions made. Accuracy can be a weak evaluation metric for unbalanced data, however all of my data-sets are balanced and should avoid this problem.
- **Balanced accuracy -** For balanced data such as mine, its usefulness diminishes compared with unbalanced data. It works by calculating the average accuracy per class where each class is weighted equally. This will aid in understanding the performance across the classes as it may highlight any disparities in the data.
- **Area Under the Receiver Operator Characteristic Curve (AUROC) -** Is a valuable metric that will help me to comprehensively evaluate my classifier's performance. It works by measuring the ability to differentiate between positive and negative results across a variety of classification thresholds.
- **Critical Difference Diagram -** For evaluating the performance of models on my data-sets a CD can be very useful. It will aid in informing my decision for selecting the best model based on overall performance.
- **Confusion matrix -** A confusion matrix will help provide a detailed breakdown of the classifier's predictions. It works by showing the counts of true positives, true negatives, false positives, and false negatives. With the aid of a confusion matrix other metrics such as precision and recall can be derived. It provides a more comprehensive understanding of the classifier's performance.
- **Computational Time -** For these experiments it will be useful to know the run time for classification. This will be useful for real-life implementation, for example if a player wishes to generate a shot-chart of their performance during half-time it will need to be able to work quickly so that some benefit can be provided before play starts again.

3.5.2. Experiments planned

Before undergoing these experiments I want to clearly outline each one and discuss the purpose of the experiment along with what is likely to be found. The experiments that will be performed can be seen below:

- **Proof of concept** - It is imperative that I prove the basis of this experiment is even possible, for this I am isolating just the two shot locations with the furthest distance between them. As initially I found that the acceleration (axis: x) values provided the most distinct shape for each shot I will be using these readings to compare a layup against a deep mid shot.
- **Complete data-set** - It will be beneficial to compare the performance of the classifiers when trained upon all the data recordings that I have. This includes both accelerometer and gyrometer data (all axis) with all the shot locations present. In theory, this approach should yield the highest accuracy since the classifiers are provided with the most training data, which can enhance their performance.
- **Normalization vs raw data** - Because of the nature of my data I believe that normalization may harm the results by decreasing the detail in each movement. However, it will prove to be an interesting test as normalization is a common technique in data pre-processing. For this experiment I will be using the complete_data and complete_data_normalized data-sets for comparison as they are my most complete data-sets.
- **Accelerometer vs gyrometer** - Because they will have already been tested together it will be interesting to see which of the recording tools performs the highest when isolated.
- **Shot selection** - When recording the data some of the shot locations are more similar than others, for example free throw and key corner are very close together and may alter the accuracy of these experiments. It will be beneficial to see the difference to the results when trying different shot location combinations. Will experimenting with different combination of shot locations yield different results
- **Comparing different time windows** - Will isolating the peak of the shot motion by cutting down the length of each shot recording from 30 to 15 features help improve the accuracy of the classifiers?

- **Axis comparison** - A test to compare which of the directional readings of both gyrometer and accelerometer provides the best performance when isolated. This test will help indicate which parts of the shooting form are the most distinct.

4. Results and Analysis

The results are based upon the experiments proposed by my experimental plan in section 3.5.2. The experiments overarching goal was to find the best way of using the data collected with the classifiers selected to discover the optimal method for data-collection and classification. To ensure fairness in all tests, the data-sets were divided into training and testing sets using an 80/20 split, with a seed of 1.

All experiments were run on the same machine to ensure that a fair comparison was performed.

4.1. Experiments performed

4.1.1. Proof of concept

For my initial proof of concept I wanted to see if the proposed problem was even possible. For my project going forwards it is essential to know that development is not wasted and there is some possibility for my project aims to be fulfilled. For this experiment I selected to compare just the acceleration values (x axis) of layups and deep mid shots only. I chose these locations as they had the greatest difference between them and in theory would require completely different levels of force to move the ball in order to score. I chose to use the acceleration values as during my pre-processing stage I found that particular data produced the most distinct shape when plotted and theorised it would be the clearest indication of which shot was taken. I had already plotted an example of a layup and a deep mid shot on the same graph, proving there was a visual difference that even a human could pick out upon quick inspection, this was shown in Figure 5 of section 3.4.

I did not perform parameter tuning on the models as this is not meant to be a lengthy test as it is just to prove that the experiment can move forward as planned as well as this the results are already very strong with just the default parameters. The results of the classifiers performance based upon some of the metrics outlined in section 3.5.1 can be

seen below in Table 2.

Classifier	Accuracy (%)	Balanced Accuracy (%)	Run Time (s)	AUROC
Decision Tree	100.0	100.0	0.005	1
Naive Bayes	100.0	100.0	0.005	1
Random Forest	100.0	100.0	0.233	1
Support Vector machine	100.0	100.0	0.004	1
KNN (Euclidean)	100.0	100.0	0.192	1
KNN (DTW)	95.0	94.4	1.482	0.96
ROCKET	90.0	90.0	22.517	0.9
ZeroR	50.0	50.0	NA	0.5

Table 2: Experiment results for the 'Ax_lay_deep' data-set.

The experiment results are highly robust, as expected due to the simplicity of the test. All standard algorithms demonstrated excellent performance, surpassing the TSC-specific models. The superior performance of the standard classifiers can be attributed to the simplicity of the data-set, which featured clearly distinguishable classes. However, it is possible that KNN (DTW) and ROCKET, which excel at detecting sequence patterns, faced challenges in performance because the initial readings in the data were quite similar, reflecting the early stages of the shot.

The overall performance of the classifiers is highly promising, demonstrating their capability to distinguish between shooting a basketball from different locations on the court. However, it is important to note that this experiment's real world capability is limited at this point. If this technique were to be directly applied in its current form to real-world scenarios, players would only have the ability to shoot from two extremely different court locations.

4.1.2. Complete data-set

This experiment aims to test which classifier performs the best when given the largest quantity of test data. All models in this experiment were again ran with default parameters, I chose to avoid parameter tuning due to the models overfitting during tuning and making it difficult to find the optimum parameters before introducing the test data. However, in future work if I were to continue this project and long computational times were not an issue then I would perform parameter tuning for all models using a 5-fold

cross validation score. This would involve training and predicting on the train set to simulate real-world application when being exposed to the test set.

Initially I expected this experiment to yield strong results as it was provided with the most amount of training data. The results for this experiment were positive, however the performance of TSC algorithms were worse than originally anticipated. The results of this experiment can be seen below in Table 3.

Classifier	Accuracy (%)	Balanced Accuracy (%)	Run Time (s)
Decision Tree	64.0	64.1	0.021
Naive Bayes	66.0	67.0	0.007
Random Forest	74.0	68.5	0.228
Support Vector machine	80.0	74.7	0.010
KNN (Euclidean)	70.0	73.1	0.145
KNN (DTW)	68.0	71.2	2.847
ROCKET	66.0	69.5	34.126
ZeroR	20.0	20.0	NA

Table 3: Experiment results for the complete data-set comparison.

The results show that are generated are very promising. As shown, all of the classifiers perform far superior to ZeroR proving that implementing this project practically is possible even with all shots. It is surprising to see that the SVM performed so strongly outperforming every classifier in both accuracy and balanced accuracy, while only being slower in computing time than Naive Bayes. I believe this could be because of the high number of features in this data-set, as SVMs are known for being effective classifiers when working with complex data that has many variables. Another reason for the superior performance could be that SVMs are less sensitive to outliers due to the nature of the 'hyperplane'. This means that if the shooting form wasn't always consistent it may reduce the other classifiers more than the SVM.

The TSC models have been disappointing in this experiment as I initially expected them to perform superior to the standard classifiers to achieve a higher accuracy. However, they still performed comparably well to the standard classifiers, especially when focusing on the balanced accuracy. Although for practical implementation the run time of these algorithms may prove to be too slow as ROCKET with default parameters performed by far the worst in that metric.

Because of the ROCKET models poor comparative results I tested the classifier again with some parameter tuning. I found that when tuning on the training set, the model may be suffering from overfitting and to counter this reducing the number of kernels from its default 10000 to just 170 improved its results across the three chosen metrics. The new updated results for this classifier can be seen below in Table 4

Classifier	Accuracy (%)	Balanced Accuracy (%)	Run Time (s)
ROCKET	72.0	75.3	1.470
ZeroR	20.0	20.0	NA

Table 4: Updated results of complete data-set with a tuned ROCKET model.

After tuning ROCKET has proven to be one of the superior classifiers as expected boasting the highest balanced accuracy of all models, with a 0.5% improvement on the next best model (SVM). along with respectable run times meaning that it may be a viable option for future implementation. For higher accuracy's than ROCKET a viable option could be Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE), however for this work having a short computational time is essential and the run time of this solution would be too slow for practical use (Lines et al., 2018).

4.1.3. Normalization vs raw data

Normalization is a standard practice in machine learning and it will be beneficial to see if my project can benefit from its implementation. Initially I hypothesised that it would be detrimental as it would remove some of the detail from the movements that helps differentiate them. For this experiment I kept the parameter tuning that was performed on ROCKET so that I could directly compare the best performance available across the two data-sets.

I believe my initial hypothesis to be correct as I have found that across all data-sets except for ROCKET the performance has decreased as seen below in Table 5 when compared with the un-normalized data shown in Table 3.

Classifier	Accuracy (%)	Balanced Accuracy (%)	Run Time (s)
Decision Tree	60.0	62.6	0.041
Naive Bayes	40.0	26.5	0.014
Random Forest	62.0	58.6	0.369
Support Vector Machine	70.0	67.9	0.058
KNN (Euclidean)	62.0	59.7	0.207
KNN (DTW)	58.0	54.7	4.459
ROCKET	72.0	75.3	1.250
ZeroR	20.0	20.0	NA

Table 5: Normalized data results.

Across each classifier the performance has decreased, I believe this is because the data has lost temporal information as normalizing may have disrupted the order of the data. Another explanation could be that outliers in the data such as a badly missed shot may have effected the normalization as min-max scaling can be sensitive to this and may have amplified their influence. However ROCKET has performed identically to how it did in the previous test once tuned. This can be attributed to ROCKETs invariance to scaling as the classifier uses random convolutional kernels to captures different scales and patterns in the time series data.

If there was to be future implementation I do not plan to keep normalization as a technique used in the pre-processing stage as it has (excluding ROCKET) decreased the balanced accuracy of all classifiers on average by 14.77%.

4.1.4. Accelerometer vs gyrometer

The main two data recording tools were the accelerometer (axis: x,y,z) and gyrometer (axis: x,y,z). If there was to be future implementation of this software reducing file storage would be beneficial and therefore makes testing the necessity of different recording metrics useful. For this experiment I am training and testing the models on the same shots, but isolating the gyrometer and accelerometer recordings to directly compare which is the more essential metric or if they rely on working in combination to produce strong results. I initially hypothesised that the accelerometer would prove to be the stronger metric as it was what I used during the pre-processing stage for feature

extraction as it gave the clearest indication of when a shot occurred when graphed, see Figure 5 for reference.

All models will be running on the same machine with default parameters as I just wanted to compare the data-type rather than any other factor in this experiment.

	Gyrometer		
Classifiers	Accuracy (%)	Balanced accuracy (%)	Run time (s)
Decision tree	68.0	66.3	0.012
Naive Bayes	54.0	55.9	0.006
Random Forest	82.0	79.7	0.289
Support Vector Machine	66.0	60.7	0.018
KNN (Euclidean)	62.0	56.9	0.189
KNN (DTW)	56.0	56.4	2.594
ROCKET	64.0	66.7	37.926
ZeroR	20.0	20.0	NA

Table 6: Gyrometer only data-set results.

	Accelerometer		
Classifiers	Accuracy (%)	Balanced accuracy (%)	Run time (s)
Decision tree	64.0	62.6	0.012
Naive Bayes	68.0	67.2	0.006
Random Forest	72.0	68.6	0.277
Support Vector Machine	72.0	66.0	0.012
KNN (Euclidean)	70.0	73.1	0.178
KNN (DTW)	72.0	75.0	2.542
ROCKET	62.0	65.4	35.807
ZeroR	20.0	20.0	NA

Table 7: Accelerometer only data-set results.

From the results it is shown that my initial hypothesis was correct with an average accuracy of 68.6% for the accelerometer across the classifier, compared with an average of 64.6% for gyrometer. However, this metric is misleading due to the outlier performance by Random Forest on the gyrometer data. Random Forest may be performing so well because it is well suited to nonlinear relationships that are present between the features and shot type. Gyrometer data can often exhibit complex and nonlinear patterns, as it measures the angular velocity of the phone. Random Forests ability to capture nonlinear relationships allows it to effectively utilize the intercity of gyrometer data.

For comparison I believe median accuracy is a better metric as it ignores outliers in the accuracy results. The median accuracy for the gyrometer data was 64% compared

with accelerometers median accuracy which was 70%. Taking this information into consideration it is clear to see that the accelerometer data is more crucial to this experiment than gyrometer.

KNN with DTW returned the highest balanced accuracy of the accelerometer results. Again this matched my expectations as by using DTW, KNN can effectively compare and match accelerometer data sequences, even when they have different lengths or temporal variations. This ability to handle temporal variations is shown as DTW is a strong choice for the accelerometer data, where the timing and duration of shots can vary as the shooting form is not always consistent.

To investigate further I have produced the following confusion matrix as shown in Figure 7

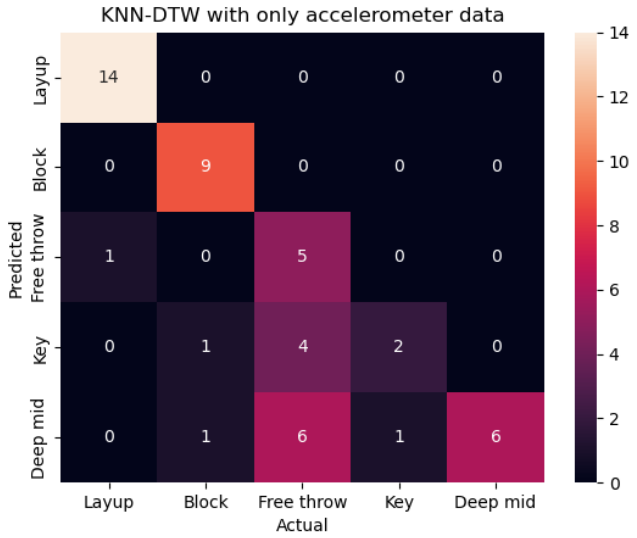

Figure 7: A confusion matrix of KNN-DTW based on accelerometer data.

What this figure tells me is that the accelerometer data for Layup and Block are very distinct, however the same can not be said for the rest of the shot locations. I will investigate this further in my next experiment. Although, with more accurate recordings and more advanced parameter tuning, KNN-DTW could prove to be a viable option when combined with accelerometer data.

4.1.5. Shot selection

After performing my last experiment it has already been seen that some shot locations have been more difficult to classify than others. For example, if we take the highest performing classifier of the last experiment which was Random Forest with gyrometer data and examine its confusion matrix as seen in Figure 8.

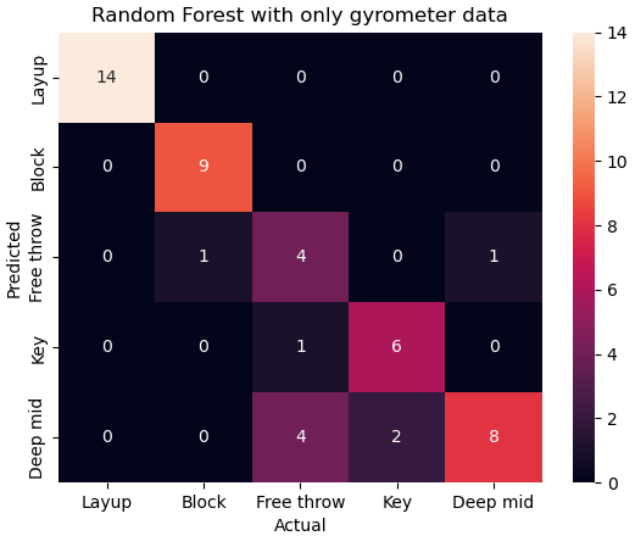

Figure 8: A confusion matrix of Random Forest based on gyrometer data.

There are many similarities between Figure 7 and Figure 8 as they both show that predicting layups and block shots are clearly distinguishable from the rest of the locations. However, as the distance gets larger between the player and the basket it becomes more and more difficult to accurately predict which type of shot it is.

For this experiment I will be testing a the accelerometer only data-set again, but removing different shot types each time to experiment and see the effect of removing shots that are too similar. I will be looking at the confusion matrices and balanced accuracy results for the experiments to see which class in particular is the most detrimental to the results. To begin, I will remove free throws and key corner shots from the data-set to see the reflection on the results of having three distinct shots remain. Then, I will re-run the experiment but add them back in one at a time and compare the performances.

The resulting confusion matrix when both free throws and key corner shots are removed can be seen below in Figure 9.

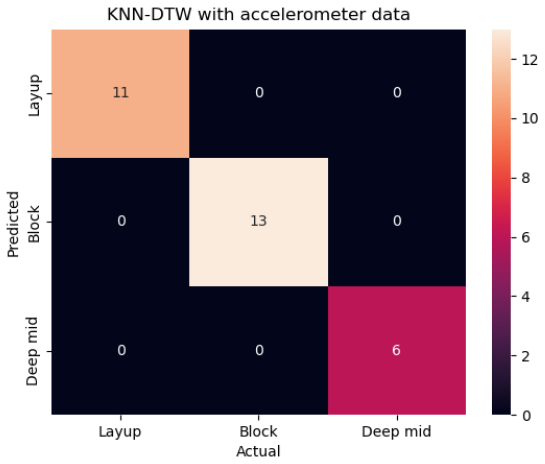

Figure 9: A confusion matrix of KNN-DTW with both free throws and key corner shots removed from the accelerometer (axis: x) data.

As expected, when left with only 3 distinct shot types KNN-DTW is able to classify them perfectly. This can be attributed to the fact that all three shots are taken with wildly different levels of force when shooting the ball.

Next, When free throws are added back into the experiment the following matrix in Figure 10 is produced.

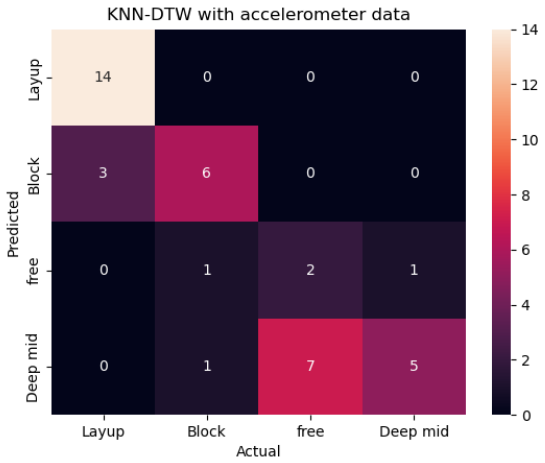

Figure 10: A confusion matrix of KNN-DTW with only key corner shots removed from the accelerometer (axis: x) data.

With free throws being added back into the training data it brings down the overall balanced accuracy as free throws are similar to both block and deep mid shots. This was to be expected, however I still thought free throws and deep mid shots would be more distinguishable than they are given the difference in court positioning they are taken from. However, upon seeing these results the distance from the basket may not be as drastic a difference as originally anticipated.

Finally, I removed free throws again and added in key corner shots to produce the following matrix in Figure 11.

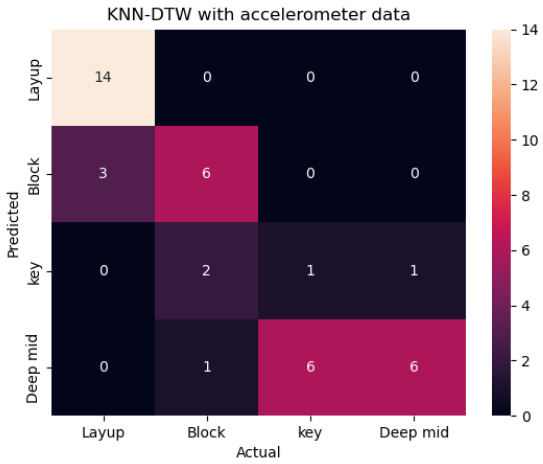

Figure 11: A confusion matrix of KNN-DTW with only free throws removed from the accelerometer (axis: x) data.

Finally with this final matrix we can see a similar problem to Figure 10 where both free throw and key corner shots appear to be too similar to the deep mid shots to be accurately distinguished. A complete table of these results can be seen in Appendix A Figure 10 which shows that ROCKET held perfect consistency across the different shot locations. SVM and Random Forest also performed well across the experiment, with Random Forest being more effected by having the free throw location present than key corner. But, overall removing just one or the other did not give a significant gain across all classifiers and may be because both key and free throw are too similar to deep mid for there to be any significant difference without removing both.

This experiment has highlighted key points that this project if re-done or continued may benefit from the spacing in-between shot locations being greater.

4.1.6. Comparing different time windows

For this experiment I wanted to see the impact of isolating the feature window to 15 features instead of the 30 I had been using previously. To do so I updated my 'find_shots' algorithm to allow for smaller windows to be captured. Previously I noticed that I was capturing unnecessary data before and after the main spike of the data. To cut this out I used the updated function shown in Algorithm 2.

Algorithm 2 find_Shots_15

```
Require: shot_type, threshold, direction
  i ← 0
  shots ← {}
  while i < len(shot_type['ax']) do
      current_shot ← {}
      if shot_type['ax'][i] < threshold then
          for j ∈ {0,1,...,14} do
              current_shot.append(shot_type[direction][i + j − 5])
          end for
          i ← i + 10
          shots.append(current_shot)
      else
          i ← i + 1
      end if
  end while
  return shots
```

The function works the same, however the number of values of which are taken before the threshold has been broken was halved and will only take the next 10 values after the threshold has been crossed. This results in just the peak of each movement being captured. The resulting data-set called 'all_15_readings' holds all of the same shots and reading metrics as 'Complete_data', however the window has just been halved resulting in half the amount of attributes (see Table 1 for details). I was initially unsure of the effect this change to the data would represent, however from the results gathered in Table 8. It is evident that the change is not substantial as the average balanced accuracy across all classifiers for the short data 65.93% compared with 69.73% for the larger time window. There was also only a 0.2% difference between the top performing classifiers when looking at the balanced accuracy's.

Classifiers	Accuracy (%)	Balanced Accuracy (%)	Run time (s)
Decision tree	50.0	54.6	0.019
Naive Bayes	60.0	60.0	0.008
Random Forest	74.0	68.5	0.316
Support Vector Machine	76.0	68.8	0.023
KNN (Euclidean)	76.0	74.5	0.272
KNN (DTW)	70.0	72.7	3.636
ROCKET	62.0	62.4	44.165
ZeroR	20.0	20.0	NA

Table 8: Shorter time window results (15 readings per window).

Overall I do not believe it to be worth while to shorten the time window for the shot duration as it may result in removing valuable data that can aid in differentiating between shot locations.

4.2. Analysis

A summary comparison of the results found throughout my tests can be seen below in Table 9 The results are very interesting and reflect what was to be expected. TSC algorithms such as ROCKET and KNN-DTW performed well, however not as accurate as originally expected. This could be because of the data-set losing too much detail during pre-processing or because of a lack of parameter tuning. If this experiment were to repeat or continue, I would like to test additional tuning methods as well as other ways of recording data that may be more accurate and better take advantage of the TSC algorithms.

	Balanced accuracy (%) across models						
Data-sets	Decision Tree	Naive Bayes	Random Forest	SVM	KNN	KNN-DTW	ROCKET
Ax_lay_deep	100.0	100.0	100.0	100.0	100.0	94.4	90.0
Gyro_only	66.3	55.9	79.7	60.7	56.9	56.4	66.7
Acc_only	62.6	67.2	68.6	66.0	73.1	75.0	65.4
Ax_no_free	58.4	61.9	77.8	74.2	62.8	62.3	76.7
Ax_no_key	64.5	65.8	69.7	73.9	63.5	65.7	76.7
Ax_no_key_no_free	86.7	80.7	91.9	85.4	92.1	100.0	76.7
Complete_data	64.1	67.0	68.5	74.7	73.1	71.2	75.3
Complete_data_normalized	62.6	26.5	58.6	67.9	59.7	54.7	75.3
all_15_readings	54.6	60.0	68.5	68.8	74.5	72.7	62.4
Average	68.9	65.0	75.9	74.6	72.9	72.5	73.9

Table 9: Final results of all models against all data-set tested on, showing balanced accuracy).

When looking at the results ROCKET appears to be very robust scoring consistent accuracy's across the tests showing that it is not effected by changes in the data as much as the other models. This benefit makes it an ideal model for future implementation, it also benefits from fast run times when using lower numbers of kernels.

I wanted to compare these results to a benchmark. However, I have not seen another activity recognition experiment that focuses on basketball making this difficult to do so. As previously discussed in section 2.3 The closest experiment would by Nurwanto et al. involving light sport activity recognition in the gym. I believe their experiment to be simpler in scope, because they only used three exercises compared to my 5 shot locations and each exercise has a distinct movement whereas my shooting form stays consistent. Still, even with these considerations many of my results are comparable with their experiment.

I believe this experiment could have been improved with the use of a more accurate data recorder, such as a smart watch. An additional data recorder could also benefit this experiment, such as GPS tracking for location on the court, however this could be argued requiring more tools limits its ability to encourage exercise. A larger spread of shot locations may also improve results as it was shown across the confusion matrices it is difficult to distinguish between free throw, key corner and deep mid as they were all similar in origin.

5. Conclusion

This project aimed to successfully model and predict the types of shots taken on a basketball court along with their locations, using accelerometer and gyrometer data. I believe that this goal has been achieved throughout my various experiments, while investigating the optimal way of doing so. It was proven to be possible in my first objective and then again with the complete data-set. Although, the accuracy did diminish it stayed within a respectable range. I found that with the complete data-set it can be shown that after slight parameter tuning ROCKET displays the best performance in balanced accuracy, while also having respectable run time which would be useful for real-world application. When compared to my initial objectives I succeeded in producing a large data-set with 50 shots from each location, I was even able to pre-process this data so that it was easily implemented in my experiments. This meant that I succeeded in completing the objectives outlined in my 'Must' section.

I was also able to complete all of my 'should' objectives outlined in my MoSCoW in section 1.2.

Short-comings arise when looking at my 'could' section as I was only able to experiment with gyrometer and accelerometer values, finding accelerometer to be the more important metric of the two as it provided more consistent results across the two metrics. Given more development time I would still be eager to pursue the rest of my objectives outlined, however they unfortunately landed beyond the scope of this project. I was able to complete almost all of the experiments in my experimental plan. I did not complete the axis comparison, because as implementation carried forward I found that time would be better spent improving other experiments. If I was to continue however, this still would be an interesting test to see which axis provides the greatest detail to a basketball shot.

In conclusion, the overall results were weaker than I initially anticipated, but I believe this can be attributed to trying to perform activity recognition on a sport not yet commonly explored with these methods. Basketball requires very precise movements that can be difficult for even TSC algorithms to detect.

If development were to continue I believe that the ROCKET classifier would prove to be the optimal choice going forward because of the robustness it has displayed across all experiments meaning that it would be easier to implement in the future.

6. Future Work

As discussed throughout the report, there were short-comings and potential improvements outside of the scope of this project. However, I believe that if data recording was more accurate and user friendly, with run times kept low this project could be fleshed out into an app for a smartwatch that avid basketball fans could use to passively and accurately track their game.

To get to this eventual goal however, objectives outlined within my 'Wont' section would ideally be met. As it stands all shooting data has come from me alone, meaning that these models are tailored to the way I shoot and if a new player were to try use the models the results could be drastically different. Therefore, in future work it would be beneficial to obtain more players shot data to better train the system and allow for new players.

Again, due to just using my shots I have no left-handed data and that may prove to be discriminatory against some users who would wish to adopt this work. Again, this would require a left handed player to also record data and train the models to accommodate for this.

Finally, I did not test different frequencies of recordings as I could not increase the frequency of my recordings due to hardware limitations. Additionally, the movement of a shot lasts only approximately 0.3 seconds and decreasing the frequency may lose out on crucial data that distinguishes certain shots from others. If future work were to be done, then possibly using a software or improved hardware that could increase the frequency of recordings may be beneficial.

References

Bagnall, A., Lines, J., Bostrom, A., Large, J., and Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31:606–660.

Bhatia, N. et al. (2010). Survey of nearest neighbor techniques. *arXiv preprint arXiv:1007.0085*.

Casale, P., Pujol, O., and Radeva, P. (2011). Human activity recognition from accelerometer data using a wearable device. In *Pattern Recognition and Image Analysis: 5th Iberian Conference, IbPRIA 2011, Las Palmas de Gran Canaria, Spain, June 8-10, 2011. Proceedings 5*, pages 289–296. Springer.

Dempster, A., Petitjean, F., and Webb, G. I. (2020). Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495.

Devasena, C. L., Sumathi, T., Gomathi, V., and Hemalatha, M. (2011). Effectiveness evaluation of rule based classifiers for the classification of iris data set. *Bonfring International Journal of Man Machine Interface*, 1(Special Issue Inaugural Special Issue):05–09.

GOV.UK (2021). Physical activity: Applying all our health.

Guiry JJ, van de Ven P, N. J. (2014). Multi-sensor fusion for enhanced contextual awareness of everyday activities with ubiquitous devices.

Kravchenko, T., Bogdanova, T., and Shevgunov, T. (2022). Ranking requirements using moscow methodology in practice. In *Cybernetics Perspectives in Systems: Proceedings of 11th Computer Science On-line Conference 2022, Vol. 3*, pages 188–199. Springer.

Leung, K. M. et al. (2007). Naive bayesian classifier. *Polytechnic University Department of Computer Science/Finance and Risk Engineering*, 2007:123–156.

Lines, J., Davis, L. M., Hills, J., and Bagnall, A. (2012). A shapelet transform for time series classification. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–297.

Lines, J., Taylor, S., and Bagnall, A. (2018). Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM transactions on knowledge discovery from data*, 12(5).

Nurwanto, F., Ardiyanto, I., and Wibirama, S. (2016). Light sport exercise detection based on smartwatch and smartphone using k-nearest neighbor and dynamic time warping algorithm. In *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 1–5.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International journal of remote sensing*, 26(1):217–222.

Pisner, D. A. and Schnyer, D. M. (2020). Support vector machine. In *Machine learning*, pages 101–121. Elsevier.

Priyam, A., Abhijeeta, G., Rathee, A., and Srivastava, S. (2013). Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology*, 3(2):334–337.

Safavian, S. R. and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674.

Schnyer, D. M., Clasen, P. C., Gonzalez, C., and Beevers, C. G. (2017). Evaluating the diagnostic utility of applying a machine learning algorithm to diffusion tensor mri measures in individuals with major depressive disorder. *Psychiatry Research: Neuroimaging*, 264:1–9.

Tangirala, S. (2020). Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm. *International Journal of Advanced Computer Science and Applications*, 11(2):612–619.

A. Appendix

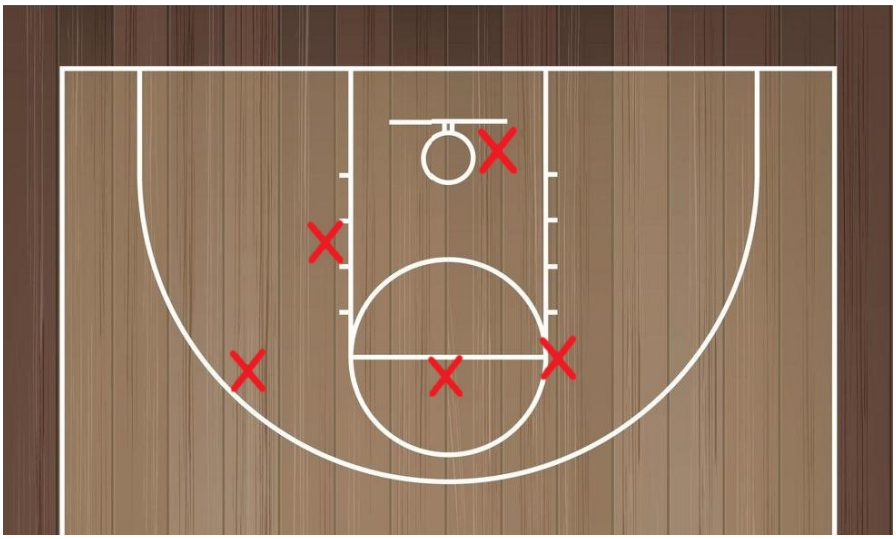

Figure 12: Basketball court with shot locations each marked by a red X.

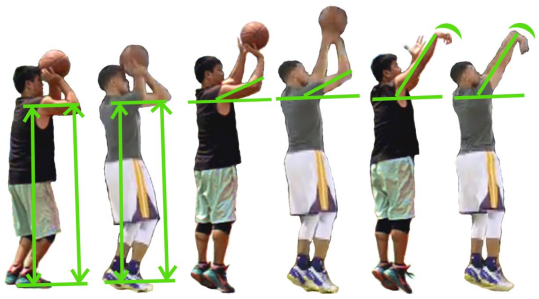

Figure 13: An example of the shooting form used in this experiment as demonstrated by Watertree[3]

Figure 14: An example of the beginning of the shot motion with the tracking software open.

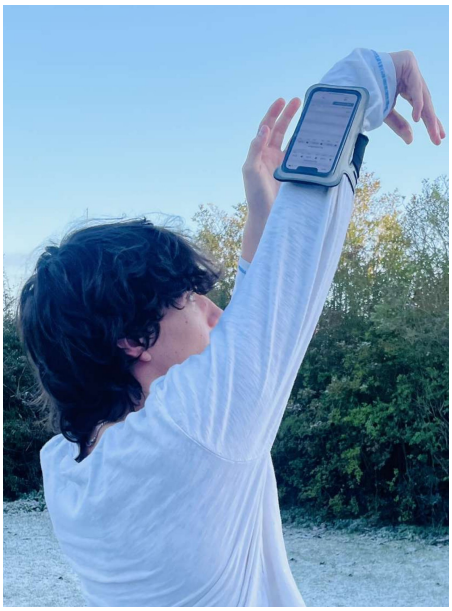

Figure 15: An example of the end of the shot motion with the tracking software open.

	Balanced Accuracy (%)		
Classifiers	No_key	No_free_no_key	No_free
Decision tree	64.5	86.7	58.4
Naive Bayes	65.8	80.7	61.9
Random Forest	69.7	91.9	77.8
Support Vector Machine	73.9	85.4	74.2
KNN (Euclidean)	63.5	92.1	62.8
KNN (DTW)	65.7	100.0	62.3
ROCKET	76.7	76.7	76.7
ZeroR	25.0	33.3	25.0

Table 10: Complete results of shot selection experiment, showing balanced accuracy for all classifiers across different data-set with altered shot locations.