Sapienza Università di Roma

STATISTICAL LEARNING

Basketball Shot Classification with Interpretable Machine Learning Algorithms

GROUP ID: **G-15**

Students

Vedat Katirci
Mert Yildiz
Ali Reza Seifi Mojaddar
Aybüke Hamide Ak
Gianmarco Ursini

Professor

Prof. Pierpaolo Brutti

Accademic Year 2020/2021

1 Abstract

Nowadays, machine learning algorithms applications area increases day by day. This study aim to classify basketball shots as goal vs. misses, using two smartphones to measure the acceleration of the 3 axes while the player throws the ball to the hoop. In this experiment one smartphone is attached to the right leg of the player and the other is attached to the right arm. Each phone records the data of 4 sensors. Three sensors stand for the acceleration and one is for the synchronization and separation of the shots. The player throws several free shots and after each throw, the details are saved including the label for each shot. To have a better insight of the importance of the features (acceleration of arm, acceleration of leg), different classification methods with the interpretation details have been applied and covered in this study.

2 Main research aim and framework

With the increasing capability to shrink powerful electronic devices to smaller dimensions, data collection became true even in all real-life situations where freedom of body movements is crucial to complete a particular task. In particular, in many sports disciplines this brand new type of dynamic data has allowed the chance of analyzing (often in real-time) many statistics, e.g players physical conditions (fatigue, beat rate, blood pressure), team strategies (player displacement over the field), betting quotes and much more. What comes after is a combined study of all those topics. Being able to classify the goodness of a basketball shot (strictly sticking to the player body dynamic) with a machine learning model, and most of all diving in the interpretation of our classifier, give rise to plenty of interesting matters of study: trying to get insights on our classifier in order to improve players training, real-time bettors performances improved are just a few of them. Evidently (being these dynamics almost chaotic that is with a strong dependence on boundary and initial system conditions) many of these topics require a huge amount of data to be precisely faced and are so far beyond the scope of our project. In this study, indeed the main focus will be on the interpretation of the model to try to answer the question: what makes a basketball shot good in the classifier's eyes?

3 Data Collection

The data has been collected by using two different smartphones that are attached to the player's arm and leg. One smartphone is attached to the right leg and the other to the right arm. Each smartphone measures the acceleration of the 3 axes to be considered for the classification of the shots as goal or miss while the player throws the ball to the basket. To synchronize and separate each shot's data between two smartphone sensors, a 4th sound intensity sensor has been added to crop the data later by using this sensor considering a whistle at the beginning and at the end of each throw. The player shoots several free throws and for each throw, the label has been recorded as goal or miss. After recording the acceleration data, each shot has been cropped and the CSV file has been imported. The CSV file contains the acceleration records for X, Y, and Z sensors with a relative time column in milliseconds.

relative_time	AccX	AccY	AccZ	DecibelSource
0	-1.143023071	9.069938965	-1.522184601	43.9537882
71	-0.295635223	8.864416351	-2.152374115	44.40436766
133	-0.936901703	9.392967224	-2.718047791	45.45900618
200	-0.777782593	9.645791473	-2.628084869	42.34972598
271	-0.440384216	8.430917816	-3.094065857	42.06205012
333	-0.953966217	9.19223465	-3.289858704	45.13124337
400	-0.074694672	9.780361633	-2.555186462	44.66734199
470	0.745150452	9.856553192	-1.873803406	44.31621048
533	0.846340027	9.379195862	-2.180515594	42.76806346
599	0.479602661	9.264384613	-1.94939621	42.11841096
669	0.31120285	9.416618042	-2.3038591	41.71755698
733	0.534837799	8.963061218	-2.048340454	39.5679252
799	0.238304443	9.275162201	-2.199226685	40.49954148
869	0.517773285	9.327104187	-2.143991547	45.07397058
933	0.035176849	9.48592392	-2.192490692	43.8257532
1000	0.219144287	9.364526367	-2.388433228	43.21494867
1071	0.398920441	9.641600189	-2.333946533	45.69139725
1133	0.400567017	9.568252716	-2.369722137	46.08707843

Data collection is performed through the recordings taken via the Arduino Science Journal app. The app lets the user record data from multiple mobile sensors in real-time. In this project, the programming language Python, and Scikit-Learn library which implements many Machine Learning algorithms efficiently has been used. Also, other regular libraries, modules, and frameworks include NumPy for matrix operation in Python. For the interpretability part Facets has been used which is a robust visualizations tool to aid in understanding and analyzing machine learning datasets.

4 Exploratory Data Analysis

In statistics and machine learning, exploratory data analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations. In this section, the basketball shots' data has been examined with the help of python tools. To have a better understanding, the distribution of normalized features data is shown on Figure 1. The scale of the data varies since different axes have different behaviour. Figure 2 shows the distribution of the labels. There are 62 goals that are labeled as 1 and 80 miss labeled as 0. Even if the dataset is not balanced with respect to the labels, the difference is not big and furthermore data has been homogeneously splitted in train and test via StratifiedShuffleSplit in order to preserve miss/goal rates over the two datasets. One of the important insights that needs to be analyzed is the correlation of the features. Figure 3 and Figure 4 represent the correlation pair plots with the data and the correlation matrix.

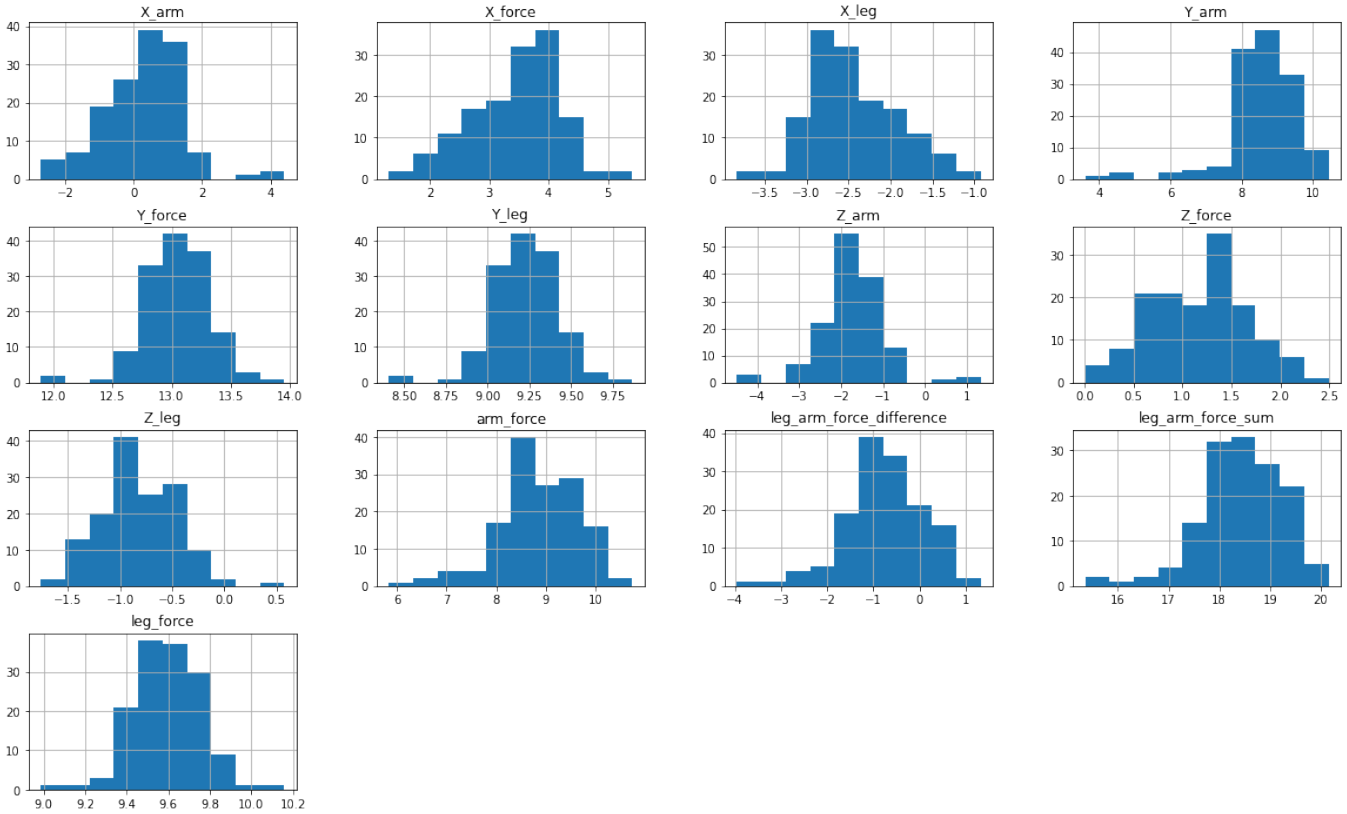

Figure 1

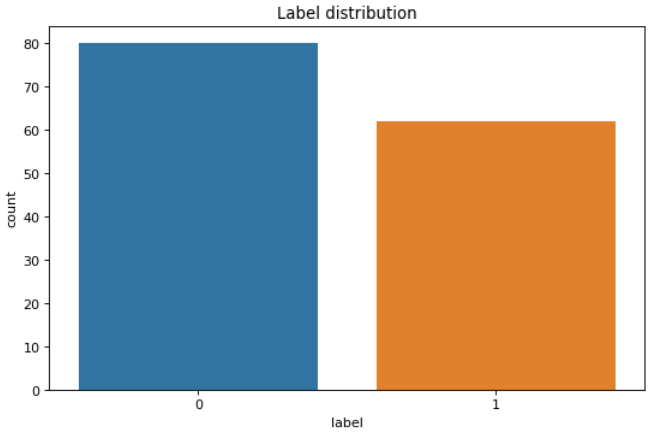

Figure 2

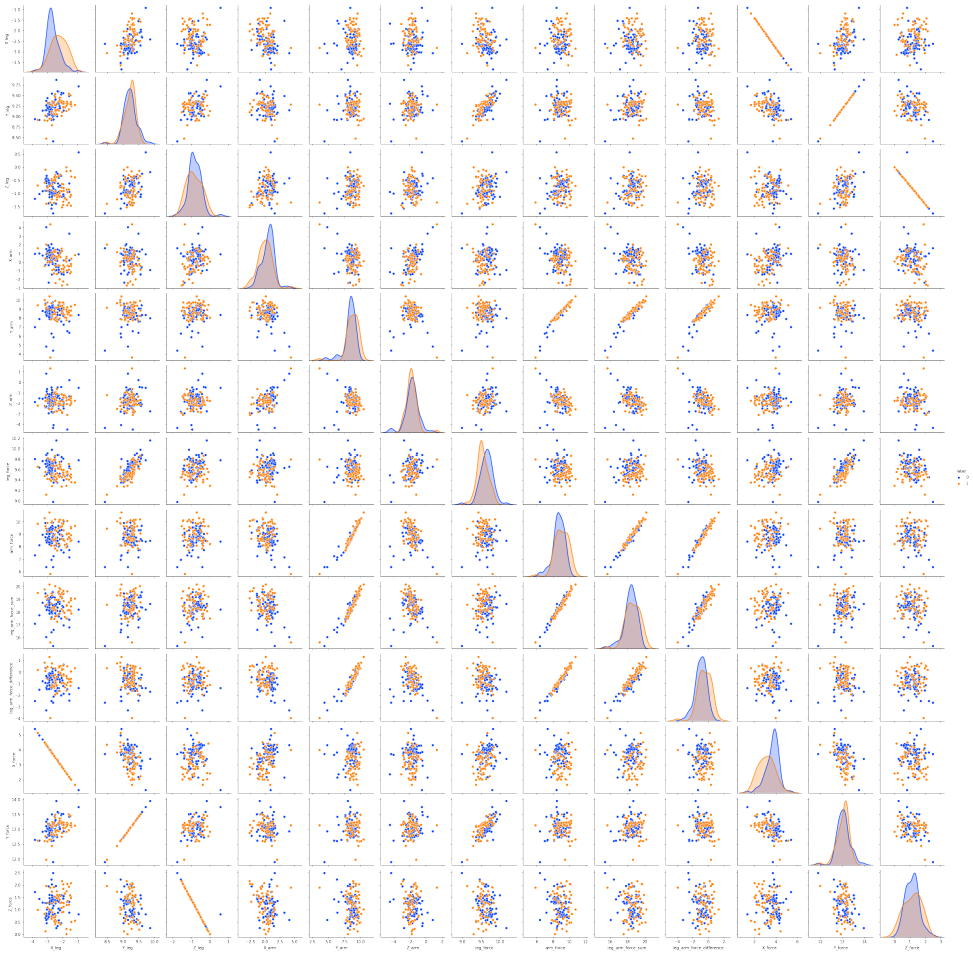

Figure 3

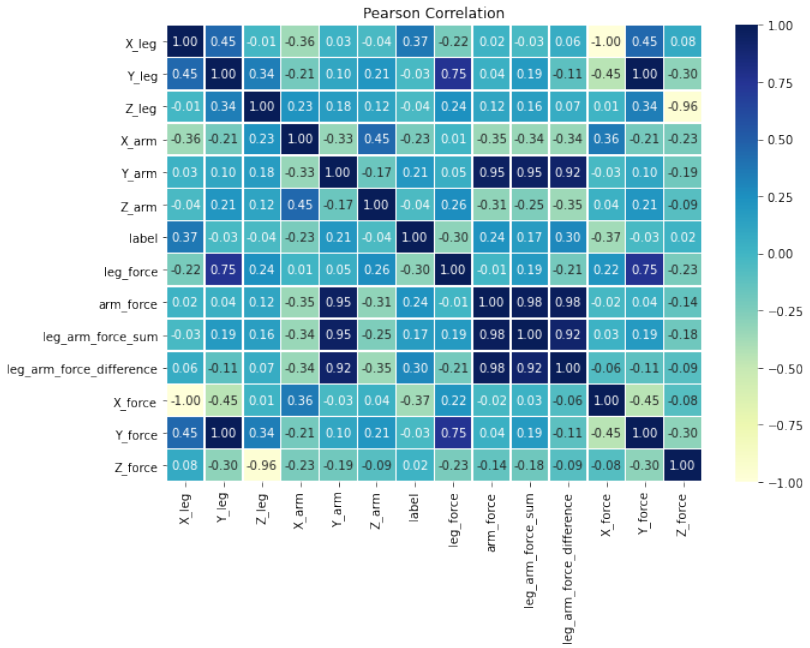

Figure 4

5 Preprocessing

Data gathering process has been explained in detail above. After gathering, the data needs to be preprocessed since data preprocessing is usually the most important phase of a machine learning project. Data preprocessing includes cleaning, instance selection, normalization, transformation, feature extraction and selection. As the first step, recorded accelerations have been cropped to clean the noise and synchronize the acceleration of arm and leg. The data has no missing values, so there was no need to work on this part. Usually most of the data has irrelevant and redundant information that needs to be cleaned. Since there was a limited available data, all features and all basketball shot's data has been taken into account. Each shot consists of thousands of rows for each shot that needs to be combined to be used on different methods, the mean of accelerations over X, Y, and Z sensors for each shot has been calculated and the data of leg and arm has been combined in a 6 columns dataframe. In the end, the data consists of one acceleration number for x_{arm} acceleration, y_{arm} acceleration, z_{arm} acceleration, and the same for leg for each shot. In the data frame below, there are 142 records, each row stands for a shot while the accelerations for each sensor and the label of shot, miss as 0, goal as 1 are represented in the columns.

	X_leg	Y_leg	Z_leg	X_arm	Y_arm	Z_arm	label
0	-2.520428	9.170563	-1.366449	0.704624	8.774168	-1.803058	0
1	-2.957112	9.322805	-0.901091	1.034031	8.972796	-1.737034	1
2	-2.220425	9.142323	-1.019355	0.617216	9.617884	-2.632403	1
3	-1.530770	9.486749	-0.625361	3.301970	6.364759	0.209094	0
4	-2.650597	8.994385	-0.964607	0.815463	8.357265	-1.919648	0
...	...	...	...	...	...	...	...
137	-2.727973	9.091102	-0.963056	2.074379	8.213622	-0.462881	0
138	-2.873861	9.326824	-0.730671	1.372348	8.129338	-1.301670	0
139	-3.317560	9.137745	-0.907511	0.796566	8.792056	-1.559713	0
140	-3.067694	9.264726	-0.530345	1.530033	9.037188	-2.258885	0
141	-2.864841	9.295853	-0.678086	-0.023854	8.535545	-1.144927	0

Figure 5

As the last step of the preprocessing, the features data has been normalized as follows:

$$x_{\text{norm}} = \frac{x - \mathbb{E}[x]}{x_{max} - x_{min}} \tag{1}$$

6 Feature engineering

Numerical Column Grouping, one of the feature engineering methods, was applied to preparing the proper input dataset, and improving the performance of models which are analyzed in this study. In order to increase the number of features and to gain insights over the shot's dynamics, new predictors with a physical meaning has been built over the original ones,e.g. taking the square root of the quadratic sum of accelerations and building a feature on it and built features proportional to the force applied to the ball. Forces were calculated in two different ways. First of all, `leg-force` and `arm-force` were calculated by using the accelerations recorded over the X, Y and Z axes for both leg and arm data.

$$leg_{force} = \sqrt{x_{leg}^2 + y_{leg}^2 + z_{leg}^2} \tag{2}$$

$$arm_{force} = \sqrt{x_{arm}^2 + y_{arm}^2 + z_{arm}^2} \tag{3}$$

Second, for each axis, `X-force` , `Y-force` and `Z-force` were calculated using the acceleration in the arm and leg recorded on that axis.

$$x_{force} = \sqrt{x_{leg}^2 + x_{arm}^2} \tag{4}$$

$$y_{force} = \sqrt{y_{leg}^2 + y_{arm}^2} \tag{5}$$

$$z_{force} = \sqrt{z_{leg}^2 + z_{arm}^2} \tag{6}$$

Finally, new features were created by taking the sum and difference of the calculated forces in the arm and leg.

$$leg - arm_{sum} = arm_{force} + leg_{force} \tag{7}$$

$$leg - arm_{diff} = arm_{force} - leg_{force} \tag{8}$$

	X_leg	Y_leg	Z_leg	X_arm	Y_arm	Z_arm	label	leg_force	arm_force	leg_arm_force_sum	leg_arm_force_difference	X_force
0	-2.520428	9.170563	-1.366449	0.704624	8.774168	-1.803058	0	9.608276	8.985184	18.593460	-0.623092	3.564424
1	-2.957112	9.322805	-0.901091	1.034031	8.972796	-1.737034	1	9.821974	9.197694	19.019668	-0.624280	4.181989
2	-2.220425	9.142323	-1.019355	0.617216	9.617884	-2.632403	1	9.463162	9.990705	19.453868	0.527543	3.140155
3	-1.530770	9.486749	-0.625361	3.301970	6.364759	0.209094	0	9.629784	7.173346	16.803129	-2.456438	2.164835
4	-2.650597	8.994385	-0.964607	0.815463	8.357265	-1.919648	0	9.426298	8.613588	18.039886	-0.812710	3.748510
...	...	...	...	...	...	...	...	...	...	...	...	...
137	-2.727973	9.091102	-0.963056	2.074379	8.213622	-0.462881	0	9.540306	8.484155	18.024462	-1.056151	3.857936
138	-2.873861	9.326824	-0.730671	1.372348	8.129338	-1.301670	0	9.786858	8.346485	18.133344	-1.440373	4.064253
139	-3.317560	9.137745	-0.907511	0.796566	8.792056	-1.559713	0	9.763614	8.964791	18.728405	-0.798823	4.691738
140	-3.067694	9.264726	-0.530345	1.530033	9.037188	-2.258885	0	9.773800	9.440039	19.213839	-0.333761	4.338374
141	-2.864841	9.295853	-0.678086	-0.023854	8.535545	-1.144927	0	9.750898	8.612024	18.362921	-1.138874	4.051497

Figure 6

7 Methods and Interpretability

Our main focus here is on the binary classification of the outcome of the shot. For this task, the linear classification was applied as a starting point. Then, a decision tree classifier and random forest were analyzed, and also a different approach was taken where we could apply the models using the whole data without calculation of means.

8 Linear classifier

8.1 Coefficient's sizes approach

As well known, linear models are often used and implemented because of their simplicity and their innate capability to express the importance of each predictor in regression or classification (as in our framework) tasks: the importance of each predictor is indeed expressed by the magnitude of the coefficient associated to each single predictor, i.e. its absolute value. A linear model is so fitted over the normalized training data with stochastic gradient descent learning, and its predictive power is assessed over the test set via the accuracy metric, being equal to 0.86 (see accuracy table in the conclusion section).

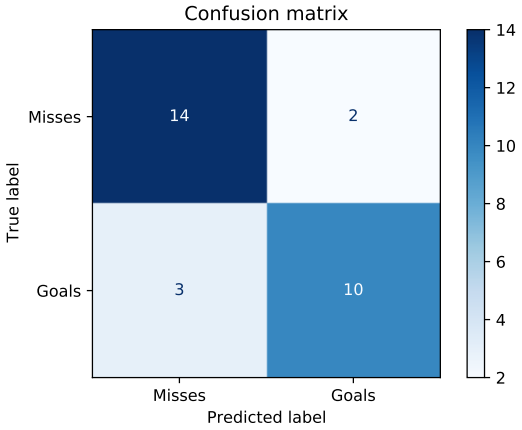

Figure 7

After the training procedure, a first dive in the linear model interpretation can be performed assessing the predictor's coefficient sizes in order to understand which are the features the classifier is weighing the most while choosing the label for each shot. The hierarchy of the predictors can be so visualized in the following plot:

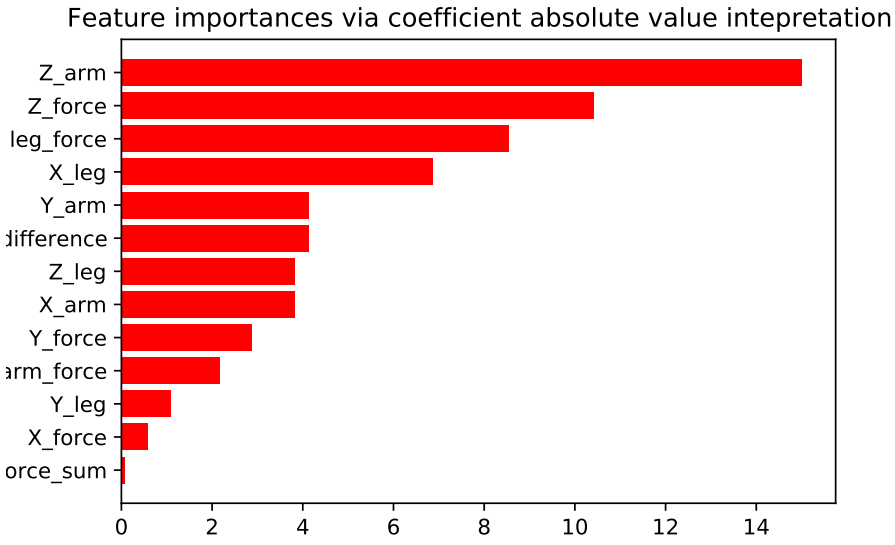

Figure 8

8.2 A geometric interpretation

From the linear model coefficient absolute values interpretation, `Z-arm`, `Z-force` and `leg-force` looks to be the most representative features of the shot's dynamic outcome. As well clear, the fitted linear model is defining a flat hyperplane splitting the 13 dimensional feature space in two different decisional spaces. A such high dimensional hyperplane is difficult to visualize in its original space, but makes sense to try to visualize it in a space spanned by the most representative features. The space generated by `Z-arm` and `Z-force` is so chosen, and the decisional hyperplane can be visualized through an intersection with the 0.5 level plane: said $x_1 =$ `Z-arm` and $x_2 =$ `Z-force` and α, β their respective coefficients together with the model intercept c, the decisional hyperplane is defined as follows:

$$\pi_{\text{decision}} = c + \alpha x_1 + \beta x_2 \tag{9}$$

An (x_1^*, x_2^*) data point will be labeled as 1 if $\pi(x_1^*, x_2^*) > 0.5$, or 0 otherwise. Taking the intersection between the two planes:

$$\pi_{\text{decision}} = c + \alpha x_1 + \beta x_2 \tag{10}$$

$$\pi_2 = 0.5 \tag{11}$$

The linear hyperplane can be visualized via the function:

$$x_1 = \frac{0.5 - c - \beta x_2}{\alpha} \tag{12}$$

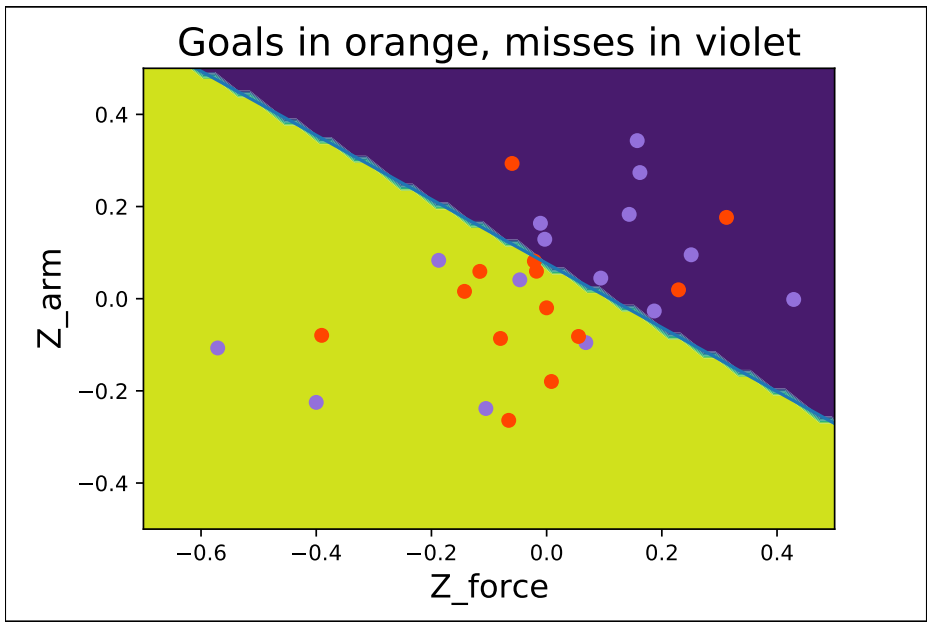

Figure 9

Of course, belonging the original data to a 13 dimensional space, even if the 2 selected features for plotting purposes are the most important ones, the above plot is just a raw representation of the decision boundary, justifying the presence of several misclassified data points. Following here a 3D visualization of the same double feature representation, being the z axis associated with the data point label value.

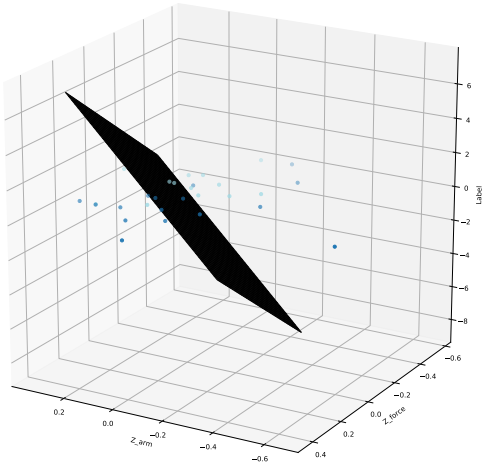

Figure 10

The only realistic representation of the decision boundary has to be performed in the 13 dimensional space. After collecting the predicted labels over the test set, a non linear mapping to a 2 dimensional space can be performed through the `TSNE()` package. Even if interpretability is lost via this embedding (being TSNE a non deterministic and randomized nonlinear transformation which has the power to preserve "neighbourhood metrics"), an accurate representation of our 2 different decision space is given in the following plot via the color background, and it can be retrieved fitting a KNN model with `n-neighbours=1`:

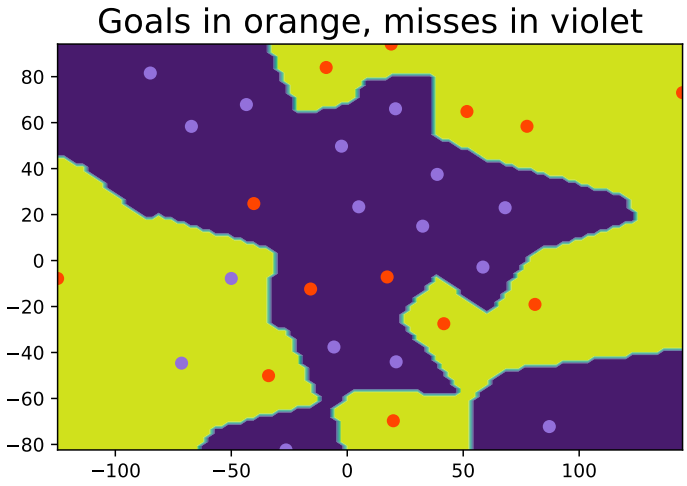

Figure 11

Being the color of the single datapoint associated with its original label, even if the projection of our 13-D hyperplane doesn't look linear in 2-D, it is possible to appreciate how the confusion matrix values are preserved in this embedded space: we have respectively 3 misclassified goals, 2 misclassified misses.

8.3 Feature permutation algorithm

Other than assessing linear coefficient's magnitudes, another analytic method can be implemented from scratch in order to retrieve the *importance* associate to each predictor: the **feature permutation algorithm**. It is reasonable to obtain feature weight results which are consistent with the ones obtained through their straight interpretation via the linear model coefficients. First step of the algorithm consists in saving the model accuracy s over the normalized and original test set. Then, K different random shuffling will be performed over the values of a single predictor (identified by letter j) at a time and the predictive accuracy of the original model is assessed over the shuffled test set and saved in the $s_{j,k}$ variable. The importance of each predictor is then computed as follows:

$$i_j = s - \frac{1}{K}\sum_{k=1}^{k=K} s_{j,k} \tag{13}$$

K repeated shufflings per each predictor are implemented to remove random noise in the estimate of $\overline{s_{j,k}}$). Basically, a epredictor's importance will be great if the shuffling of its values resulted in a severe reduction of the predictive accuracy.

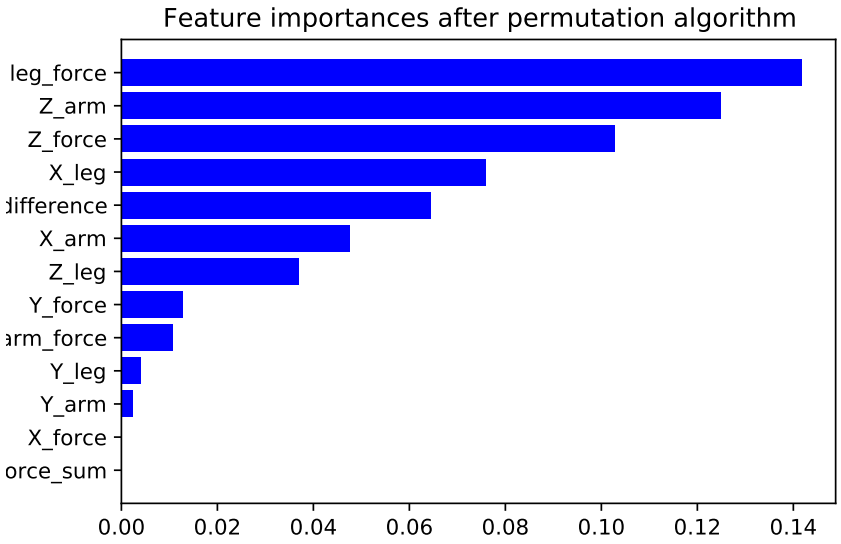

Figure 12

As it can be clearly seen, the results are consistent with the simple interpretation of the linear model coefficients, letting us appreciate even more the innate interpretability power of linear models. Infact, once again, `leg-force`, `Z-arm` and `Z-force` are on the podio of the 3 most weighted predictors in our classification task.

9 Decision Tree Classifier

9.1 Theoretical description of decision tree classifiers

The decision tree uses Gini impurity values to classify the features of the record in a dataset node by node. The nodes at the top of the tree contain highest values of Gini impurity. If Gini value is less than or equal to 0.491 for X_{leg} then the child node on the left filters the true values, also the child node on the right is false for the X_{leg} condition. let k represent the probability of a data point being incorrectly classified. Let X represent the dataset we apply the decision tree on. Then Gini impurity calculates the probability of each feature occurring and multiplies the result by 1, that is, the probability of occurring on the remaining values. $G(k) = \sum_{i=1}^{n} p_i * (1 - p_i)$ The decision tree is built on the gain of information on the features that contain the highest Gini impurity value. As the decision tree classifier calculates the Gini impurity at each node and creates child nodes, the decision tree's depth increases as depicted in the following plot.

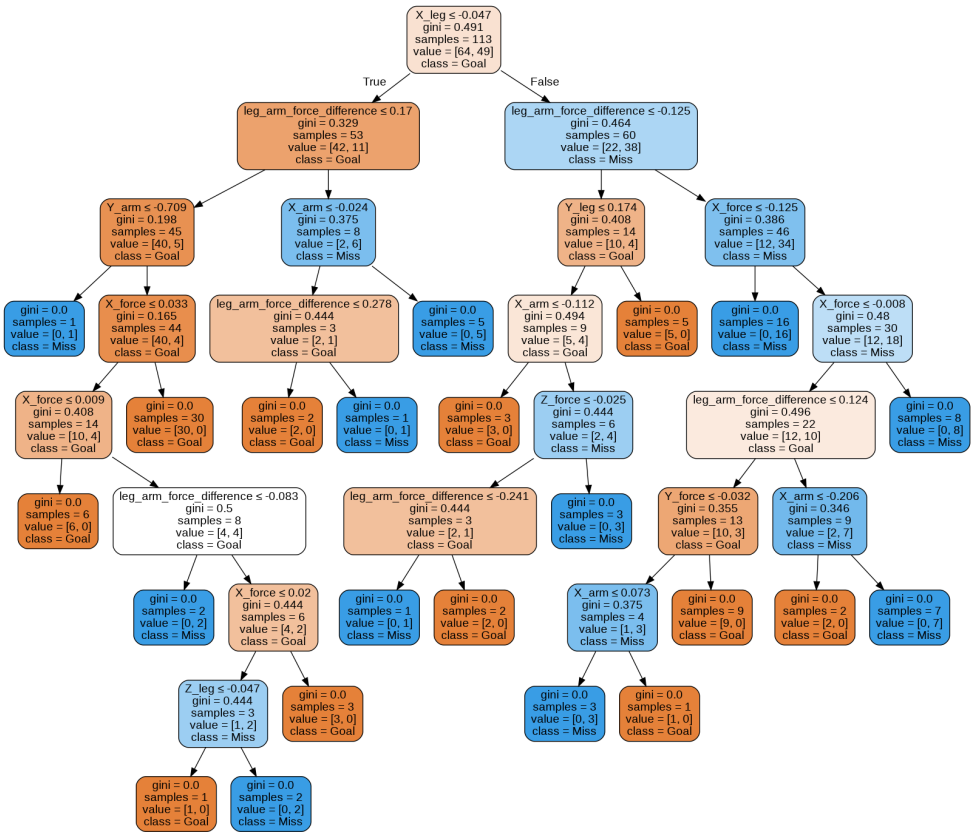

Figure 13

9.2 Fine tuning the model

In Scikit-learn optimization of decision tree classifiers performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning. In the following, the plot shows a decision tree on the same data with max-depth=6, besides pre-pruning parameters other attribute selection measures such as entropy can be applied. By applying this, the classification rate is increased, which depicts better accuracy compared to the previous model. This pruned model is less complex, explainable, and easy to understand than the previous decision tree plot.

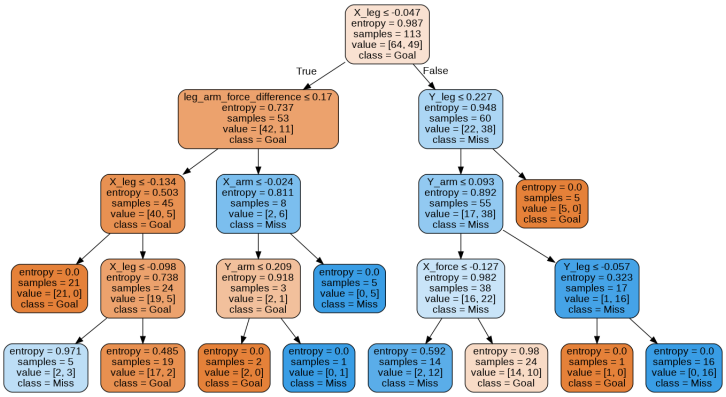

Figure 14

The following plot is a representation of the decision tree boundaries. As it's clear, overlapping between decision areas is occurring. Once again, the boundaries have been plotted over a 2 dimensional space: what is happening is that decisional areas are projected over a single plane, while (for example over a 3-D space) they would set themself at different z heights.

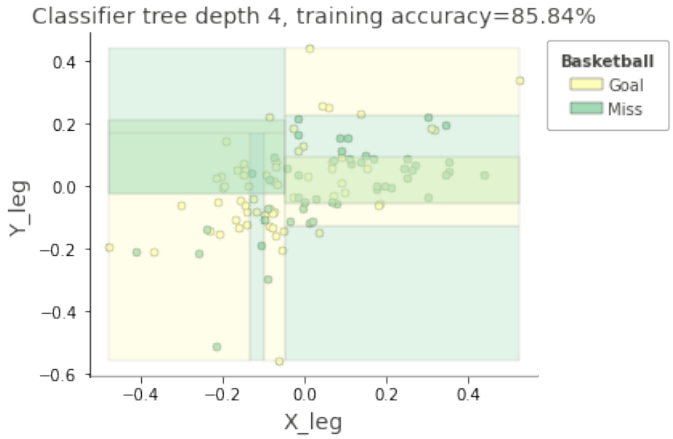

Figure 15

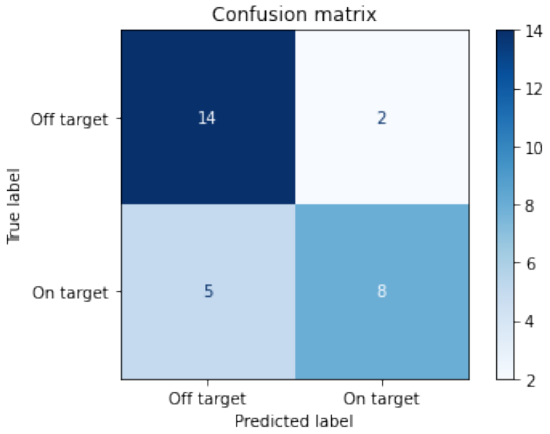

Figure 16

10 Random forest

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them. Internal estimates monitor error, strength, and correlation and these are used to show the response to increasing the number of features used in the splitting. Internal estimates are also used to measure variable importance

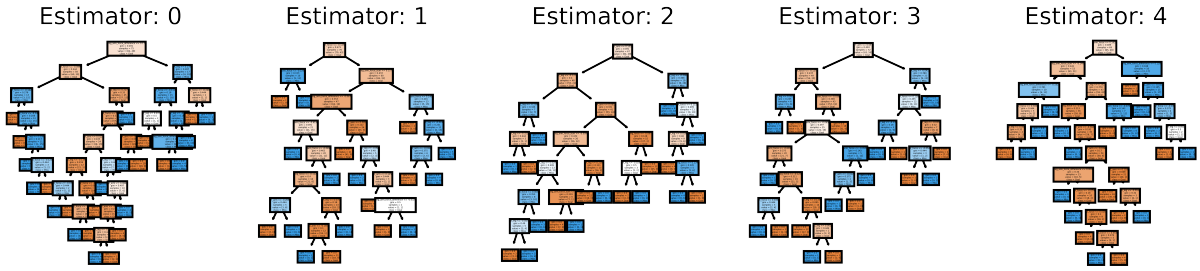

Figure 17

10.1 Feature importance in random forest

Feature importance attribute outputs an array containing a value between 0 and 100 for each feature representing how useful the model found each feature in trying to predict the target. This gives the opportunity to analyse what contributed to the accuracy of the model and what features were just noise. Among the Features it is clear that X_{leg}, Y_{leg}, and leg-arm-differences are contributing more.

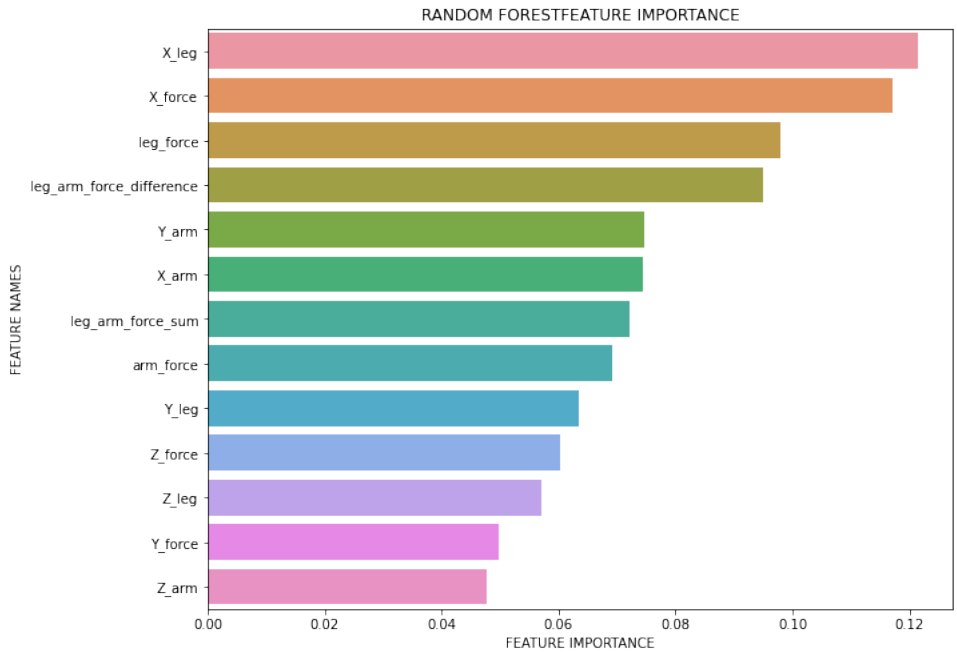

Figure 18

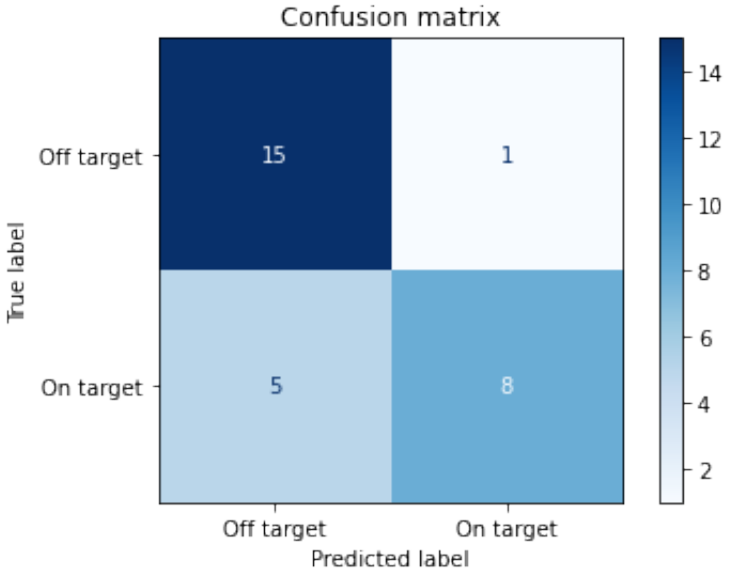

Figure 19

10.2 IML with SHAP

SHAP (SHApley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions.

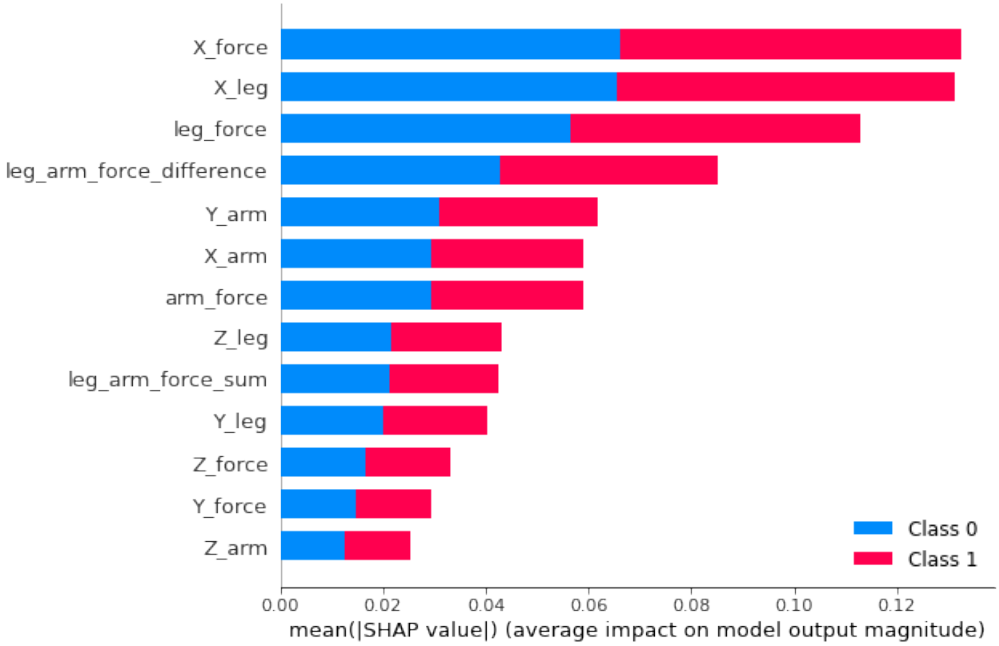

Figure 20

11 Further Analysis

We decided to approach our classification problem from a different point of view. We have time series data, which contains our sensor data with respect to the time frame that we have for each shot. Up to now we were preprocessing data by aggregating all the related time frames by finding the best method within median,mean and sum. What we have realized is when we are aggregating data we are losing information related to the time. That's why we decided to flatten out data and as time length we decided to choose minimum length sensor data that we have within each individual shot by assuming minimum corresponds to data when the ball leaves the hands. This minimum length is 30 data points. This process is illustrated below:

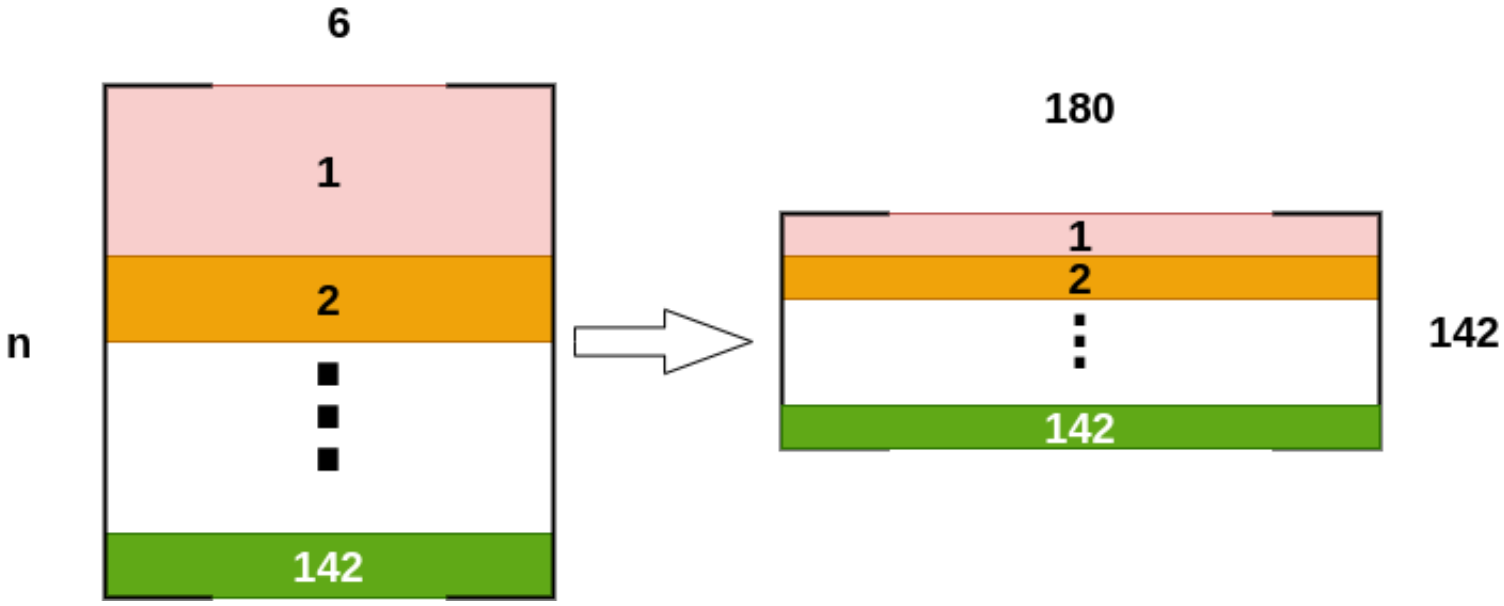

Figure 21

Furthermore, we decided to choose the first 30 data points rather than the last 30 ones due to immediate shooting after the whistle. After preparing data in this way we have normalized them by using the MinmaxScaler function.Then we have applied Random Forest,Support Vector Classifier,Logistic Regression and their corresponding accuracies are collected in the conclusion section table.

12 Conclusion

Model	Accuracy
Linear model (Averaged data)	0.86
Decision tree (Averaged data)	0.72
Pruned ecision tree (Averaged data)	0.75
Random forest (Averaged data)	0.79
Logistic regression (Full time series)	0.83
Decision tree (Full time series)	0.65
Random forest (Full time series)	0.89
Support Vector Classifier (Full time series)	0.86

Even if other qualitative measures can be used to assess model's goodness, accuracy metric has been the one used in this classification framework. Looking at the model accuracy, and assessing in particular how the linear model outperforms random decision tree and random forest classifiers in the averaged data approach, it is reasonable to assume that the outcome of the shot's dynamic (i.e. the selection of our label) has a linear dependence over the original features. Indeed, the motion of the ball in the space can be fully described just expressing the forces impressed on it over the 3 axes, which are proportional to our original features up to a proportionality constant (i.e. the mass measure).
Diving in the interpretability, linear classifier tended to weight the most the Z axis features, i.e. the one orthogonal to the plane to which the parabolic shot motion belongs to. And this is something that could have been expected, being indeed most of the misses due to an overshooting on the left or the right side of the basketball ring.
Looking at the decision tree and random forest classifier's feature importances, leg acceleration information is the most weighted one, in particular the one over the X axis, i.e. the one belonging the plane to which the parabolic shot lies on and passing through the player and the base of the basket. It basically means that the amount of the player motion towards the basket while shooting is a crucial parameter for those 2 classifiers while labeling the data points.
Even if the last approach only focused on the improvement of the predictive accuracy without dealing with model's interpretability, it is somehow possible to assess the importance attributed by those 3 classifiers to the time feature, which again (looking at the severe grow of the time-series fitted model's predictive power) can be considered as a crucial information while expressing the outcome of the shot dynamic. Further studies may indeed quantify its importance w.r.t. the acceleration's original features.
It is finally clear that through ML techniques strong and accurate predictive power can be easily achieved with relatively simple implementations. The crucial point is *why* we should believe to this prediction if we have to rely on them for serious and important decision. Once again, knowing the particular experimental framework dynamic and being capable to extract the parameters on which the model is deciding on, makes both the professional data scientist and the common person more relaxed and confident about the model predictions. Would you bet on the classifier prediction now?

13 References

[1] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.

[2] Du, Mengnan, Ninghao Liu, and Xia Hu. "Techniques for interpretable machine learning." Communications of the ACM 63.1 (2019): 68-77.

[3] Xiangyi Meng, Rui Xu, Xuantong Chen, Lingxiang Zheng, Ao Peng, Hai Lu , Haibin Shi, Biyu Tang, Huiru Zheng "Human Action Classification in Basketball: A Single Inertial Sensor Based Framework", 1st edn., Singapore: Springer (2018).

[4] James, G., Witten, D., Hastie, T., amp; Tibshirani, R. (2021). An introduction to statistical learning: with applications in R. Springer.

[5] Dataman, D. (2021, May 2). Explain Your Model with the SHAP Values. Medium. https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d.

[6]Elshawi, R., Al-Mallah, M.H. Sakr, S. On the interpretability of machine learning-based model for predicting hypertension. BMC Med Inform Decis Mak 19, 146 (2019). https://doi.org/10.1186/s12911-019-0874-0

[7]Tamas Madl,Plotting high-dimensional decision boundaries, (2019), GitHub repository, https://github.com/tmadl/highdimensional-decision-boundary-plot