RoboFEI 2022 Team Description Paper

Álvaro D. Neto, Bruno Bollos Correa, Gabriel M. Schiavetto, Gabriel S. L. Agune, Giovani S. Pereira, Guilherme W. Cardoso, Henrique B. Simões, Isabella R. Moscardo, João V. L. Aguiar, Leonardo da S. Costa, Luiz F. L. Baptistella, Wesley de S. Motta, Flavio Tonidandel, Plínio T. A. Junior, and Reinaldo A. C. Bianchi

Robotics and Artificial Intelligence Laboratory
Centro Universitário da FEI, São Bernardo do Campo, Brazil
{flaviot, rbianchi, plinio.aquino}@fei.edu.br

Abstract. This paper presents the current state of the RoboFEI Small Size League team as it stands for RoboCup International Small Size League competition 2022, in Bangkok, Thailand. The paper contains descriptions of the new robot decision to pass and shoot to the goal and the positioning of the attacker robots during normal game.

1 Introduction

For RoboCup 2022, the RoboFEI team intends to use mostly the same electronics and mechanical design that have been used over the last years.

Some significant advances were made in our software, the improvements were verified in the 2021 Small Size League RoboCup, which resulted in our best placement ever in the competition, getting us in the hall of fame of the league [1]. The objective now is to improve furthermore the software system and, according to our plans, we should be able to start replacing at least two robots per year from now on. Due to the COVID-19 pandemic, most of the development made in the past two years has been made in the software features, this TDP focuses mainly on those. With our researches, we hope to bring innovations and new ideas for the community.

2 Software

In the strategy software, we have have mostly improved the robot decision to pass and shoot to the goal, the offensive positioning of the robots and the dynamic between defenders and attackers during the game.

Additionally, we have started to open source some of our works, as we believe that sharing software is equivalent to sharing knowledge, which is important for educational purposes. Everything mentioned in this paper can be found at our GitLab page [2].

2.1 New log analyser tool

Inside the group of projects that were released in the past year, we have the LogAnalyserRoboFEI-SSL, which is a software that has the main purpose of dealing with log files from Small Size League RoboCup matches [2]. Its main features are: read and run a log file and send the messages from the referee and the vision over the network using the UDP protocol. It also contains its own graphical client, which contains the drawing of the field with many other informations about the match, sent by the game controller and the ssl-vision. A screenshot of the running software can be seen on Figure 1.

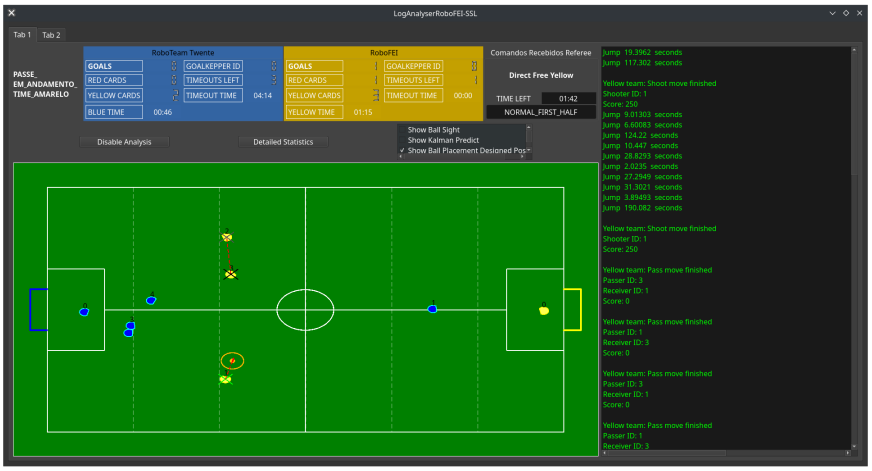

Fig. 1. Screenshot of the software LogAnalyserRoboFEI-SSL during its execution.

Alongside with this, the software is capable of detecting when a pass or a shoot to goal occurs and, once the move ends, evaluate its quality by giving it a score from 0 to 250. A zero score means that the pass or shoot decision was bad, while a score of 250 means that the move was very good.

The evaluation of a shoot move follows the decision tree [3] shown in Figure 2. It can be seen that the maximum score is given only if the move resulted on a goal and the minimum score occurs only if the ball did not even get to the opponent defense area.

The evaluation of a pass move is more complex. If the receiver does not get the ball, the score will be zero, otherwise, the score will depend on what the receiver did with the ball. The analysis of the receiver move is called parallel move. The decision trees that evaluate the score of a pass move and the parallel move can be seen in Figure 3 and 4, respectively.

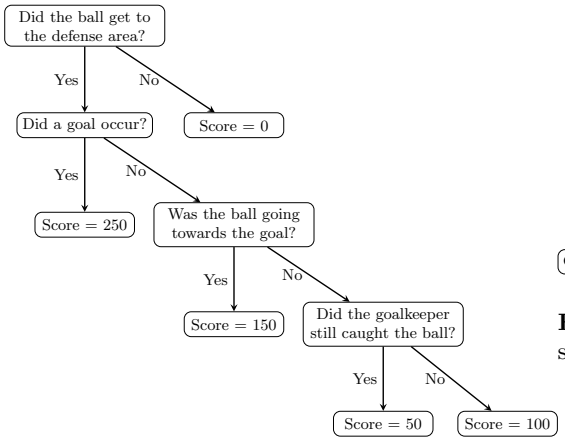

Fig. 2. Shoot move evaluation decision tree.

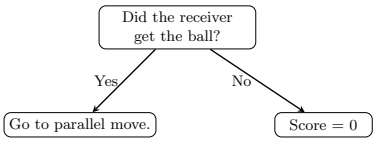

Fig. 3. Pass move evaluation decision tree.

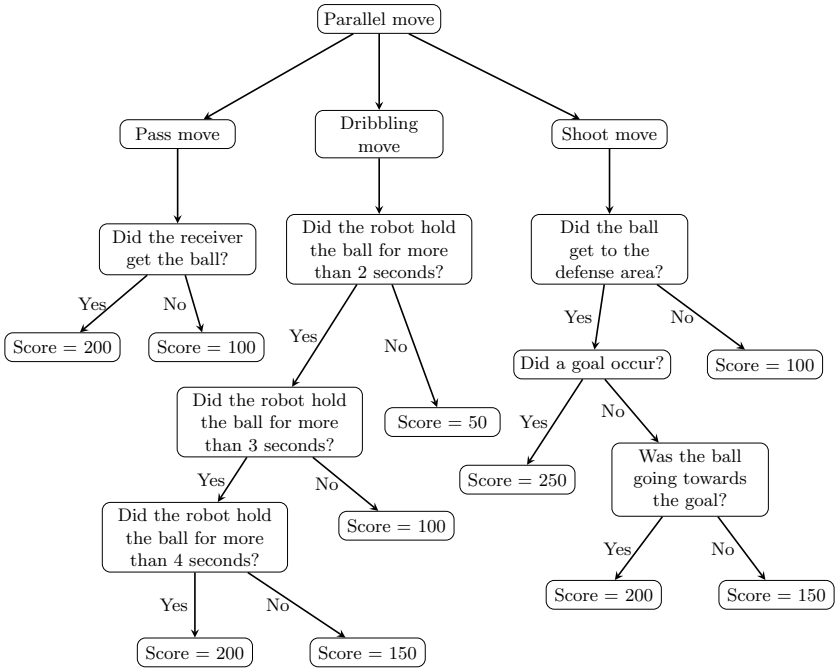

Fig. 4. Parallel move evaluation decision tree.

When a move is evaluated, some extra parameters are measured and written into a file alongside with the score of the move. That way, at the end of the process, this file has the information of all pass and shoot moves that occurred

during the analysed match. The parameters evaluated on a shoot to goal and on the pass move can be seen on Table 1.

Table 1. Shoot to goal and Pass move parameters.

Move	Parameter	Description
Shoot to goal	X_0	Angle of the free path (without opponents) from the ball to the opponent goal
	X_1	Distance from the ball to the opponent goal
	X_2	Opponent's marking (distance of the closer opponent to the ball)
Pass	X_0	Angle of the free path from the ball towards the receiver
	X_1	Distance from the ball towards the receiver
	X_2	Opponent's marking over the receiver (distance of the closer opponent to the receiver)
	X_3	Angle between the line from the ball to the receiver and the line from the receiver to the opponent goal
	X_4	Angle of the free path from the receiver towards the goal
	X_5	Distance from the receiver to the opponent goal
	X_6	Opponent's marking over the ball (distance of the closer opponent to the ball)
	X_7	A parameter that mesures if the ball is getting closer or farthest from the opponent goal

At the end, the parameters of the pass and shoot moves are normalized into a scale from 0 to 250 of only integers numbers. The normalization of all parameters is done according to the curve shown in Figure 5, where y is the normalized parameter and x is the parameter before the normalization, therefore, $y_0 = 0$ and $y_1 = 250$.

For the distance parameters, $x_0 = 0$ and x_1 is one half of the width of the field (disregarding the offset), i.e. $x_1 = 6\ m$ for division A rules, and $x_1 = 4.5\ m$ for division B rules. For the parameters that measure the angle of the free path, $x_0 = 0$ and $x_1 = 45°$.

Database of passes and shoots from RoboCup matches With the files generated from the analysis of the match, it is possible to create a database

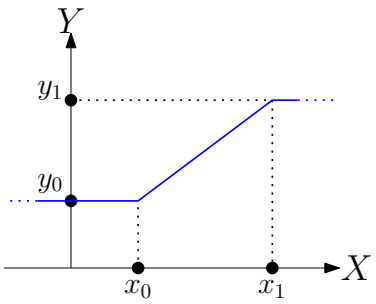

Fig. 5. Normalization curve.

containing the information of all passes and shoots from older matches of the league using its log files. This was done for the most outstanding teams from RoboCup 2017, RoboCup 2019 (Division A only) and RoboCup 2021 (online); the results were uploaded into a GitHub Repository [4].

2.2 Pass and shoot success prediction

Until RoboCup 2019, our team did not have an elaborate strategy when the ball is in play, the robot was programmed to always kick towards the opponent goal when it had possession of the ball. As the league evolved, teams have increasingly valued the passing strategy in their software; the team's behavior in rolling ball situations is of supreme importance and essential for any team.

Knowing this, we decided to implement a way for the robots to decide, on the most efficient way, when to pass the ball to other robots and shoot to the goal. For that purpose, a supervised machine learning regression model was designed [5] using the database mentioned in the last section. As result, given the input parameters from the field, the model is able to predict which move would be better in the given circumstances, by finding the biggest score of all possible moves, which is to pass to all alies or shoot towards the goal.

In order to create the machine learning model, the scikit-learn [6] python module was used. There are currently 5 types of Machine Learning (ML) models implemented in the strategy software:

- Linear Regression: a simple, popular and fast ML model that learns linear bias of the input features in order to generate its output [5];
- k-Nearest Neighbor (KNN): another simple ML model that make its predictions based on the input closest points in the dataset [5];
- Random Forest: a bootstrap ensemble learning algorithm that combines multiple weak learners for the purpose of make a strongest learner that increase its prediction power [5],
- AdaBoost: a boosting ensemble learning algorithm that combines multiple weak learners into a weighted sum way, making the best learners more important in the final prediction [7];

– Gradient Boosting: another boosting ensemble learning algorithm on which the residuals are used as input on the next consecutive learner, hence building an additive ML model [5].

Pass and shoot to the goal prediction analysis For the analysis of the shoot move, data from matches of ER-Force, KIKS, RoboFEI and TIGERs Mannheim from Robocup 2021 were used, arranging a dataset of 250 samples. On the other hand, for the analysis of the pass move, data from matches of ER-Force, KIKS, RôboCIn, RoboFEI and TIGERs Mannheim from Robocup 2021 were used, making a dataset of 751 samples.

For the analysis of the machine learning models, in order to determinate the R^2 score from training and testing dataset and the importance of each feature on the prediction of a new data, the following process was performed multiple times:

– Randomly separate 80% of the dataset for model training and the remaining 20% for the testing dataset;
– Create the model with the training dataset;
– Calculate and save the importance of each feature of the model;
– Make model predictions with the test and training datasets and save the value of the prediction R^2 score, which is a value between 0 and 1 that measures the quality of the predictions for a dataset.

This process was repeated 500 times. The final prediction score is the average of the scores obtained on each iteration, the final importance of each feature is determined the same way.

It was also measured the time for the obtained model to predict 10000 random input datas, making it possible to calculate the average time for the model to predict each output. Since all tests were made on the same condition, it became possible to compare their results and select which one of them are better for implementing on the strategy software.

The results obtained for the shoot to goal and pass prediction are shown in Tables 2 and 3, respectively. All tests mentioned were made for five different ML models: Linear Regression (LR), AdaBoost (AB), Gradient Boosting (GB), Random Forest (RF) and k-Nearest Neighbor (KNN). The test and train R^2 scores are multiplied by 100, thus being in a scale between 0 and 100.

The results seen on Table 2 show that the fastest model to predict the evaluation of a shoot move is the Linear Regression model with an average prediction time of 0.04 μs, while the slowest one is the KNN model, with a prediction time of 3 μs. However, even the KNN prediction time is too small to be taken into consideration, since the function to predict the shoot and pass move is called, on the worst possible scenario, every 25 ms, this way, that the worst prediction time would need to be more than 100 times longer in order to slow down the software performance. Therefore, this parameter can be ignored at the moment.

By analyzing the test and train scores, the Gradient Boosting model seems to be the better choice for the shoot to the goal prediction on an environment

Table 2. Shoot to the goal prediction results.

	LR	AB	GB	RF	KNN
X_0 importance [%]	14	13	13	7	16
X_1 importance [%]	54	68	82	88	71
X_2 importance [%]	32	20	4	5	13
Train R^2 score (0–100)	19.7	26.2	24.8	21.8	23.4
Test R^2 score (0–100)	13.9	12.4	15.6	12.8	12.9
Prediction time [μs]	0.04	1.46	0.17	0.42	3.02

Table 3. Pass prediction results.

	LR	AB	GB	RF	KNN
X_0 importance [%]	26	48	51	67	79
X_1 importance [%]	13	7	8	4	3
X_2 importance [%]	26	11	10	8	6
X_3 importance [%]	1	6	5	2	1
X_4 importance [%]	4	5	3	2	2
X_5 importance [%]	6	6	7	7	4
X_6 importance [%]	18	13	12	8	3
X_7 importance [%]	4	4	4	2	2
Train R^2 score (0–100)	23.2	31.2	39.1	28.1	24.1
Test R^2 score (0–100)	20.4	20.3	20.8	21.1	20.8
Prediction time [μs]	0.06	2.86	0.67	1.31	16.84

with three features (X_0, X_1 and X_2) with a dataset of 250 samples. However, the other 4 models have done good predictions as well and the difference of the prediction quality is not too significant. In practice, any of the five models would do a good work, with a small difference between them.

It can also be seen that the X_1 feature, which is the distance between the ball to the goal, is way more important than the other two features in all models that were tested. In summary, according to the Robocup 2021 matches, the success of a shoot to the goal depends primordially on the distance of the ball towards the goal.

Seeing the Table 3, it can be seen that the slowest model, which is KNN once again, takes 17 μs on each prediction, but again, it is not slow enough in order to be taken into consideration.

The results shown in Table 3 show that the test score is very similar to all five models, but KNN and linear regression presented both the lowest train score, which allows the conclusion that those are the most generic models.

Analysing the importance of each feature, it can be seen that X_0 is by far the most relevant of them. Bellow X_0, the second and third most relevant parameters are, respectively, X_2 and X_6. So, according to the matches analysed, in order to make a good pass, it is important that the receiver must be as distant as possible from the opponents and that the path between the ball and the receiver must not have too many obstacles.

Pass and shoot success prediction in the strategy software With these new changes, when the robot approaches the ball, if the ball is not being possessed by the opponent team, the software evaluates the score of the move to pass to other robots on the field and to shoot on the opponent goal. Robots too close to our defense area and too close to the ball are ignored. At the end, if the shoot score is greater than 150, the robot shoots to the goal, otherwise, it chooses the move that resulted in the greater score.

Since the strategy software is written in C++ and the machine learning models used scikit-learn with python, a python interpreter had to be embedded into the C++ application. This has been accomplished with the help of the *pybind11* library [8], which made possible to create a python object via C++ code, call python methods and getting its types properly converted into C++ types.

The ML models that seemed to work better were Gradient boosting for the shoot to goal prediction and AdaBoost for the pass prediction. Linear regression was also a good choice, especially for the shoot to the goal prediction and it has the advantage to be an algorithm very easy to be implemented. Random forest also worked well on the pass to the goal prediction, but not as good as when compared with AdaBoost.

2.3 Mines positioning

With the pass strategy implemented, it was also needed that some of the robots in the field could get into a good position to receive a pass close to the opponent goal and, preferably, free of marking. For that purpose, the Mines Positioning algorithm was developed.

The Mines positioning is a simple algorithm used to determinate the point, inside a matrix of n_rows rows and n_lines lines, further away from the objects called mines and closest to the object called fortune.

It works by giving a score from 0 to 9 to each point of the matrix. The mines acts decreasing the score of its closest points, while the fortune decreases the score of its furthest points.

In the software, this is used to position the robots with offensive behaviour during normal start and all robots during opponent ball placement. In order to use it, the matrix must be proportional to the dimensions of the field, this can be seen on Equation (1).

$$\frac{n\_rows}{n\_lines} \propto \frac{field\_width}{field\_height} \tag{1}$$

That way, each element inside the matrix represents one square field area. The area of each square is given by Equation (2).

$$square\_area = \frac{field\_width \cdot field\_height}{n\_rows \cdot n\_lines} \tag{2}$$

The inverse of $square\_area$ is the scale of the matrix, which is currently set as 0.01 mm^{-2}. For better results, it is important that the $square\_area$ must not be much larger than the area occupied by a robot on the field.

In order for the algorithm to be adequate for different situations, there are a few attributes that can be defined each time it is used. They are:

- $additional\_mines\_positions$: A list with the coordinates of mines to be added beside the robots;
- $consider\_ally\_mine$: A boolean variable that defines if ally robots should be considered as mines;
- $begin\_area$ and end_area: The points on which the destination should be. This way, only points $P(x, y)$ on which $begin\_area.x < x < end\_area.x$ and $begin\_area.y > y > end\_area.y$ are considered by the algorithm;
- $fortune\_point$: The position of the fortune point;
- $minimum\_distance\_fortune$ and $maximum\_distance\_fortune$: Are the distance intervals from the $fortune\_point$ on which the fortune acts. This means that the score is always zero for points on which the distance from $fortune\_point$ is larger than $maximum\_distance\_fortune$, while for points on which this distance is less than $minimum\_distance\_fortune$, the score is unchanged by the fortune. For the other points, the score will be affected by the distance from the fortune point;
- $mine\_radius$: The radius of each mine;
- $mine\_reach\_length$: The maximum distance from the border of the mine on which the mine is still effective.

Every position of the matrix between the points $begin\_area$ and end_area is evaluated, depending on the position of the mines and of its distance to the

point of fortune, with a score from 0 to 9, where, the bigger the score the better is the position.

At the end of the evaluation, the chosen point will be in the center of the region of points with the maximum score.

Mines positioning on normal start During the normal start, the attackers must be as close as possible of the opponent defense area. For that purpose, the point of fortune is located on the center of the opponent goal. The matrix of the algorithm can be visualized with a heat map, thus, a game situation can be seen in Figure 6, the ally robots are represented by a black circle, opponents are represented by a white circle and the robot whose destination is being calculated by the algorithm is being represented by a black square. The points with higher scores are more green, while the points with lower scores are more red and the points with medium score are yellow, the location of the mines are represented by dots.

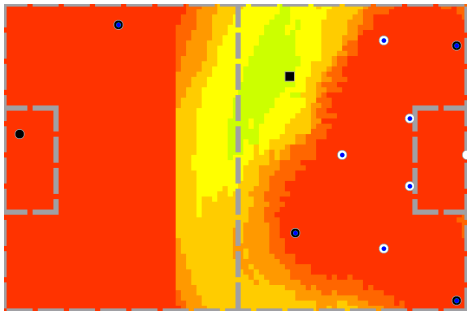

Fig. 6. Mines position during normal start - Situation 1.

In Figure 6, it can be seen that one mine was added on the position of each ally and opponent robots, that way, the attacker will get into a position as distant as possible from them. Mines are added on the destination of the ally robots as well, to avoid that the algorithm sets the same destination to two different robots. In the situation shown in Figure 6, the destiny of the robot would be on the center of the most green area. Other scenarios are shown in Figures 7 and 8.

Mines positioning during opponent ball placement During the ball placement of the opposite team, our robots must be keep a distance from the line between the designed position and the current ball position in order to respect the game rules. Besides, it is preferable that our robots stay close to our defense area, since the ball placement will most likely be proceeded with a free kick for the opponent team, hence, our robots must be prepared to defend.

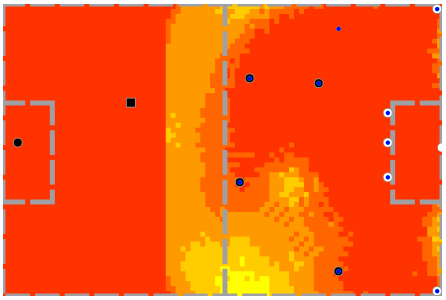

Fig. 7. Normal start - Situation 2.

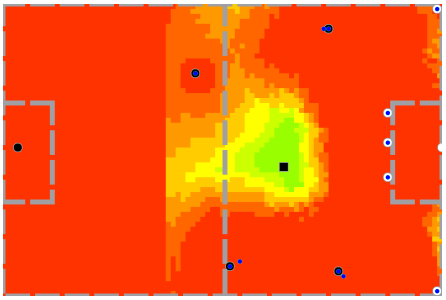

Fig. 8. Normal start - Situation 3.

For that purpose, the Mines positioning algorithm was used with no fortune points and 5 to 10 mines on the line that begins on the current ball position and ends on its designated position, and, one mine on each ally robot. Beyond that, points that are far from the defended goal are not considered.

That way, the robots tend to avoid the ball placement area and stay on the defense area, far from each other, as it can be seen on the different scenarios shown in Figures 9, 10 and 11.

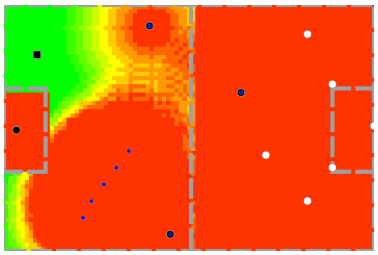

Fig. 9. Defensive ball placement.

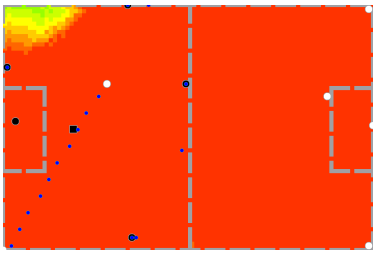

Fig. 10. Blue team with mines positioning on yellow ball placement.

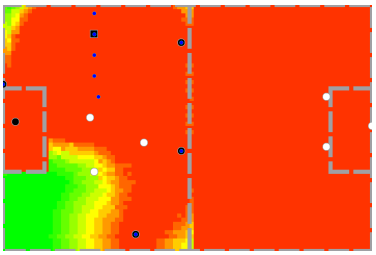

Fig. 11. Blue team with mines positioning on yellow ball placement.

3 Acknowledgements

We would like to thank, in advance, the Small Size League Committee, for the consideration of our material. We would also like to immensely thank the staff of Centro Universitário FEI, for all the help we always received from them.

References

1. RoboCup. Hall of fame. `https://ssl.robocup.org/hall-of-fame/`, 2021. [Online; accessed 23-Jan-2022].
2. RoboFEI-SSL. Robofei - small size league - gitlab. `https://gitlab.com/robofei/ssl`, 2021. [Online; accessed 23-Jan-2022].
3. Wikipedia contributors. Decision tree. `https://en.wikipedia.org/wiki/Decision_tree`, 2021. [Online; accessed 29-July-2021].
4. github/Bollos00. Bollos00 - databaseforkicksandpassesrobocup - github. `https://github.com/Bollos00/DatabaseForKicksAndPassesRobocup`, 2020. [Online; accessed 23-Jan-2022].
5. Andriy Burkov. *The hundred-page machine learning book*, volume 1. Andriy Burkov Canada, 2019.
6. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
7. Wikipedia contributors. Adaboost. `https://en.wikipedia.org/wiki/AdaBoost`, 2021. [Online; accessed 29-July-2021].
8. Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. `https://github.com/pybind/pybind11`.