**Splitting the data**

A new DataFrame is created which will have only the useful columns, these are shown below with the number of null values.

We have 0 null values in our DataFrame so we can proceed.

The data values will be assigned accordingly, predictors (X) and target (y), then it will be split into train and test.

* X will include all skills from players.
* y will include player position.

Inputs from train\_test\_split:

* test\_size=0.2 — Train data will be 80% and test will be 20%.
* random\_state=42 — Used to seed a new RandomState object.
* stratify=y — Makes the values splitted proportional.

Although the standardization of the data is an important pre-process, this will be executed in combination with each machine learning model and cross-validation through Pipeline to avoid leakage.

So what are these terms? Here is a brief explanation of each:

## Data leakage

When information from outside the training dataset is used to create the model.

## Standardization

Data standardization is the process of rescaling one or more attributes so that they have a mean value of 0 and a standard deviation of 1.

## Pipeline

According to scikit-learn documentation:  
Sequentially apply a list of transforms and a final estimator. Intermediate steps of the pipeline must be ‘transforms’, that is, they must implement fit and transform methods. The final estimator only needs to implement fit.

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.

## K-fold Cross-validation

Splitting the data into train and test is necessary for supervised learning, however, the results of the model will be based on this only partition. A solution to this problem is a procedure called k-fold cross-validation, where data is divided in k smaller sets. The training data will be k-1 folds, the remaining fold will be used for validation, average of the validation results across the k-folds will be the performance measure.

This procedure could be computationally expensive, but it helps to avoid overfitting or selection bias.

Gráfico, Diagrama

Descripción generada automáticamente

# **Player Classification**

Before jumping into different machine learning models to classify players position, let’s see the correlation matrix between player attributes.

By looking at this heat map, we can assume that models should easily identify Goalkeepers, let’s find out how they perform.

*Note: There will be a brief explanation of each classification algorithm, a deeper approach of each is out of the scope of this story. The presented explanation should be enough to understand the displayed code.*

## K-Nearest Neighbor

An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

The image below shows an example of how the algorithm works. The green dot is the test object. If k = 3 it will be classified as a red triangle since there are two red triangle against one blue square (circle area). If k = 5 it will now be classified as a blue square, since these “neighbors” now have the majority (dashed circle area).

Imagen que contiene Diagrama

Descripción generada automáticamente

The following code combines what we’ve read so far:

* Data standardization.
* Use of pipeline to link the different “pieces” (with their corresponding parameters) of the ML process.
* Cross-validation through the data with k-folds (k = 5) .

Texto

Descripción generada automáticamente

A useful tool to visualize the performance of an algorithm is the confusion matrix. This two dimension table shows the labeling of the players. The normalized shows the classification as percentage and the unnormalized as number of players. For instance:

* 510 (73%) Attackers were correctly classified.
* 180 (26%) Attackers were classified as Midfielders.
* 8 (1.1%) Attackers were classified as Defenders.
* 0 (0%) Attackers were classified as Goalkeepers.

And so on for each position.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Decision Tree Classifier

Structure in which each internal node represents a “test” on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. Composition of DTC:

Diagrama

Descripción generada automáticamente

Survival of passengers on the Titanic.

* **Decision Nodes:** As name implies, represent a decision point of a predictor variable that will lead to the target variable.  
  (example — gender)
* **Branches:** Connections between nodes, represented as arrows. Each branch represents a response.  
  (example — male/female)
* **Leaf Nodes:** Leaf nodes or terminating nodes represent the final class of the outcome.  
  (example — died/survived)

*Note: For our data, decision nodes will be the predictors, branches will be “≤*X*” and the leaf nodes will be the player position.  
For example, “skill\_dribbling” ≤ 55.0 → Defender.*

Again, we use pipeline to get the best parameters for the algorithm.

Can we be sure these are the best parameters? The code below shows a demonstration on the “behind the scenes” of how the best parameters are obtained in each GridSearchCV.

As we can see, out of the 24 different possibilities, (8, 'gini') has the best score (0.858638...). Now that we have the best parameters we can fit the model.

‘gini’ is the default parameter.

Take a look at the respective confusion matrixes.



Gráfico

Descripción generada automáticamente

Decision Tree Classifier Confusion Matrix

## Support Vector Machine

Diagrama

Descripción generada automáticamente

Example from [MathWorks](https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html).

This algorithm is based on the idea of finding a hyperplane that best divides the data. The goal is to find a hyperplane with the greatest possible margin (distance) between an observation (support vector) and the hyperplane, giving a greater chance of correctly identifying new data. If the number of input features is two, then the hyperplane is just a line, if the number of input features is three, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds three. The idea is that the data will continue to be mapped into higher and higher dimensions until a hyperplane can be formed to segregate it.

The multi-class problem is broken down to multiple binary classification cases, which is also called one-vs-one. Which splits the dataset into one dataset for each class versus every other class.

* **Binary Classification Problem 1**: ATT vs. MID
* **Binary Classification Problem 2**: ATT vs. DEF
* **Binary Classification Problem 3**: ATT vs. GK
* **Binary Classification Problem 4**: MID vs. DEF
* **Binary Classification Problem 5**: MID vs. GK
* **Binary Classification Problem 6**: DEF vs. GK

Each binary classification model may predict one class label and the model with the most predictions or votes is predicted by the one-vs-one strategy.

*Note: To get a better grasp of the algorithm,*[*notes*](http://cs229.stanford.edu/notes/cs229-notes3.pdf)*by Andrew Ng are fantastic.*

Once again, the best parameters are obtained and the model is fitted.

In this case, the best parameters where the default values, so no need to type them.

Corresponding confusion matrixes.



Interfaz de usuario gráfica, Gráfico

Descripción generada automáticamente

## Logistic Regression

This model is used to predict a binary outcome given a set of features (predictors). The classifier output (target) can be 1 or 0. We are looking for the probability that an observation is part of the class (1) or not (0).

In a nutshell, it takes the features, multiplies each by a weight, sums them, and passes the sum through a sigmoid function to generate a probability and assign a class. The weights (vector w and bias b) are learned from a labeled training set via a loss function (cross-entropy loss).

Our data has more than two targets, so we use the multinomial logistic regression which uses a generalization of the sigmoid, called the softmax function.

*Note: Again, this is a very simplistic approach of the algorithm, find more*[*here*](https://web.stanford.edu/~jurafsky/slp3/5.pdf)*.*

Same old, same old, use pipeline to get the best parameters and fit the model.

*Note: The parameter*multi\_class*is set by default to “auto” which will select ‘“multinomial” if data isn’t binary. If “multinomial” is selected, the softmax function is used to find the predicted probability of each class to assign the target.*

Last confusion matrixes.



Gráfico

Descripción generada automáticamente

Logistic Regression Confusion Matrix

# **Results**

As expected, all different models could easily identify Goalkeepers, the majority of skills they have seem to be negatively correlated with the rest, and strongly correlated within each other.

We’ve seen the different confusion matrixes, let’s take a final look at some of the main classification metrics. These are obtained through classification\_report and accuracy\_score from the sklearn.metrics library. For simplicity, all reports are placed together in one image.

Una captura de pantalla de un celular

Descripción generada automáticamente

Algorithms Classification Reports and Accuracy Score

SVM seems to be the most accurate algorithm to determine players position. However, there are other important metrics. Placing an Attacker as a Defender may not be the end of the world, but certainly, it won’t perform as good. Let’s take a look at what these useful metrics bring to the table.

## Accuracy

How close is the predicted value to the actual target.

## Precision

Fraction of relevant instances among the retrieved instances. (Diagonal values from the normalized confusion matrixes).  
How many selected items are relevant?

## Recall

Fraction of the total amount of relevant instances that were actually retrieved.  
How many relevant items are selected?

## F-score

Harmonic mean of precision and recall.

# **Conclusions**

There are several supervised algorithms, this story showed a glance at some of them. A correct pre-process of the data along with the correct selection of parameters is crucial for every algorithm.

When it comes to choosing which algorithm performed better it’s important to remember:

* Precision is a good model metric when the costs of False Positive is high.
* Recall is a good model metric when there is a high cost associated with False Negative.
* F-score might be a better model metric if we need to seek a balance between precision and recall.

Another important metric which sometimes is overlooked is time! Yes, you want an accurate model, but some models take more time for a small gain in accuracy. To exemplify, SVM had a 0.95% better accuracy score than Logistic Regression, however, finding the best parameters for LogReg was 6.6 times faster than SVM (both originally designed for binary classification).

Time in seconds.

In some scenarios 1% could be a significant difference yet sometimes it may just be okay to lose that percent to get a faster result. Choosing the best algorithm depends on the attributes of your data.