# Football Player Injury Predictor using Scraped Data from the Internet

Marc Zoel Arias Mitjà[1], Yeray Navarro Soler[2]
[1,2]Escola Tècnica Superior d'Enginyeria Industrial de Barcelona - ETSEIB, Barcelona, Spain
[1,2]Master's Degree in Industrial Engineering - MUEI
[1,2]Master's Degree in Automatic Control and Robotics - MUAR

***Abstract* - This paper presents an approach to football player injury prediction. The data used in this study consists on player statistics for each match played, for example: time played, goals, assists, passes and so on. The data acquisition have been performed using a web scraper to enable the gathering of a large dataset containing the information of all the matches on the big five European leagues from the previous months. The extracted data is a total of 98 features for each sample. To manage this dataset more easily and with less computational effort, it has been processed to reduce its dimensionality and obtain the most relevant features. The algorythms used for this matter have been random forest and PCA. With this data, two models have been trained and compared, a convolutional neural network and a support vector machine.**

***Index Terms*—Injury Prediction, Scraping, Data Handling, Random Forest, PCA, Convolutional Neural Network, Support Vector Machine**

## I. Introduction

FOOTBALL, also known as soccer, is the most popular sport in the world. This sport have 275 million players worldwide and an estimated 4 billion fan base, considering all youth and top leagues [1]. This sport generates substantial revenue, particularly for top teams and organizations. However, the physical demands of football often result in injuries.

The number of games played annually across various competitions is rising. As the time between games is being reduced, is nearly impossible for players to maintain peak physical levels for every game due to the insufficient recovery time [2]. Consequently, there has been a significant increase in the frequency of injuries across Europe's top divisions.

For instance, the 21/22 season witnessed 4,810 injuries, a 20% rise with respect to the previous season [3]. Studies indicates that football players have an injury risk rate of 12 injuries per 1000 hours, which increases to 36 injuries per 1000 hours during match play. Furthermore, about 3.7% of football injuries are due to overuse, and 34% of the injuries lead to an absence for more than 28 days for the player. Moreover, football players face a 65-95% chance of experiencing at least one injury throughout their career [4].

Other researches have established a statistically significant relationship between the number of days players are absent due to injuries and the difference between the team's actual and expected finish position in the league table [5].

All these facts lead up to the estimated cost of player injuries across the various leagues for the 21/22 season being over €600 million for the clubs according to European Football Injury Index. Given these substantial costs, professional teams have a strong incentive to invest in injury prevention and rehabilitation programs [5] [6].

The importance of player injuries and their financial consequences highlight the need for proactive measures to reduce these risks. Injury prediction and prevention is a decisive objective for football clubs to help players improve their performance through the years. The main goal is to inform of the personalized strategies to follow on a daily basis once a player is susceptible of being injured. Medical staff and coaches could implement changes on squad rotations, training intensity, recovery treatments and match planning to reduce the probability of injuries and improving the team's general performance.

Algorithms such as decision trees, support vector machines or neural networks can analyse data of different sources, like sport statistics websites, electronic health records, wearable devices, and so on, to find relationships and patterns to specific football injuries.

In this study we explore football player injury prediction with the potential of machine learning algorithms to develop a predictive model. It is focused only on match statistics, as training data is often restricted and not publicly available.

The structure of the paper is organized as follows. Section II describes the state of the art in this area of work and researches carried with different approaches. Section III explains the proposed approach and the methodology used to develop the predictive model. Section IV describes the experimentation done: data acquisition, its preprocessing and the models tested, which are CNN and SVM. Section V summarizes the work and the conclusions are discussed.

## II. State of the Art

Predicting injuries is a difficult challenge because of the different conditions and individual characteristics of each player, regarding biological and physical factors. The injury prediction in football has traditionally been determined by subjective evaluation, historical records and limited statistical analysis performed by the medical teams. However, with the current technological advancements, including large player performance data bases and better sensors, in combination with the rise of machine learning and pattern recognition techniques, new alternatives and methods are appearing. Advanced analytics can be used to identify risk factors by studying player statistics and biomechanical data to try and prevent injuries.

The relationship between training load and injury is a fast evolving field of study. Studies have demonstrated that training loads that are either too high or too low leads to a higher risk of injury. Nonetheless, there is no general opinion in which are the determinant factors to analyse. Realizing the possibilities that these studies can bring, football clubs are taking advantage of technological developments in data collection, such as multi-camera methods and electronic performance and tracking systems (EPTS), together with wearable sensors enabling the acquisition of more detailed data [7].

The vast majority of football injury prediction studies are focused on non-contact injuries, for instance [8], [9], [10] and [11]. These injuries can be predicted using models based on physical load data of the players. However, other studies don't discriminate between types of injuries, for example [12] and [13].

Most of the data used on these research is gathered using wearable sensors or EPTS during both training sessions and matches. Some studies, as [11] also included personal and psychological information.

The diversity of the models used for this topic is quite large, for example in [10] and [11] k-nearest neighbour classifiers were used, whereas in [8], [12] and[11] XGBoost algorithm was used. Decision trees were also applied by [9] and [11]. In addition, k-means classifier and support vector machines were also analysed in [10]. Supervised principal component analysis was performed in [13] in combination with Cox regression. Regarding the number of features used on each of the studies mentioned above, they vary between 18 [9] and 65 [10].

## III. Proposed approach

The main objective of this study is to develop a predictive model to determine whether a football player is susceptible to suffer an injury. The data to analyse is extracted from the player's statistics from previous matches played, as this is publicly available. To extract a notorious quantity of data from the matches of the top five European Leagues, a web scraper has been used in order to make this process more efficient. Once the data has been collected, a preprocessing has been performed in order to obtain all the sample's features with the needed form. After that, a dimensionality reduction procedure have been applied due to the large quantity of features gathered. Both Random Forest and PCA have been used to retain the most significant features. Then, two models were built and trained: Convolutional Neural Network and Support Vector Machine. Finally, these models are evaluated and compared against each other.

The proposed approach can be summarized as follows:

1) Scraping data from the internet.
2) Data handling and preprocessing.
3) Dimensionality reduction.
4) Creation and training of the models
   a) Convolutional Neural Network (CNN)
   b) Support Vector Machine (SVM)
5) Models evaluation and comparison.

## IV. Experimentation

In this section, we present the experimentation process carried out in our study, which involves data scraping, data handling, Convolutional Neural Networks (CNN) and Support Vector Machines (SVM).

### A. *Data acquisition: Scraping*

Here we provide an overview of the data scraping process done in our study to acquire the necessary data for the prediction algorithms. Data scraping refers to the automated extraction of data from various sources, such as websites, APIs, or databases. In our case, we are obtaining the data from a website known as *FBRef*. To do that, we installed a python library called *soccerdata*.

After exploring all the possibilities offered by the website, we have selected those tables that we considered most relevant and significant in terms of injury prediction. The scrap was made on a total of 4 tables which were called *df_summary* *df_passing*, *df_possession* and *df_defense*. All of them contain the same number of rows but not the same number of columns, as they contain different features. Each row of the data sets correspond to the statistics achieved by the player in a football match.

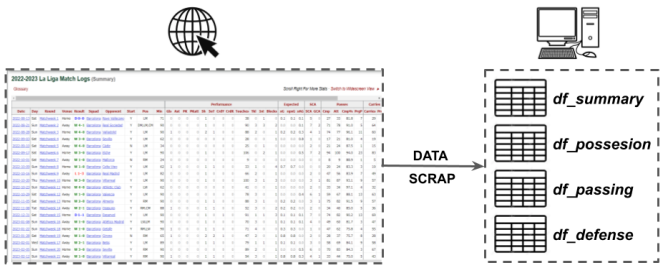


Fig. 1: Data scrap

It is important to know that the tables of the *FBRef* website are updated in real time -at the end of the day- with the data of the matches played that day. This implies that, given that the study is being carried out with current data, each time that we enter to modify the code we will have new data generated on that day. The fact of working with real time data motivated us and we thought it could be a good idea to semi-automate the scraping process to be able to update our data day by day in order to get the best possible results.

These tables have a column called *match_id* which contains a unique identifier for each football match. Therefore, a particular *match_id* will be repeated as many times as players have been on the pitch in that match. This identifier was very useful to compare the *match_id* of the most current table with the one that was last updated. A python file has been generated to compare these columns and keep only the new identifiers, which belong to the most recent matches. Therefore, we will scrap data filtering only by those identifiers to update our tables, saving us execution time and computational costs.

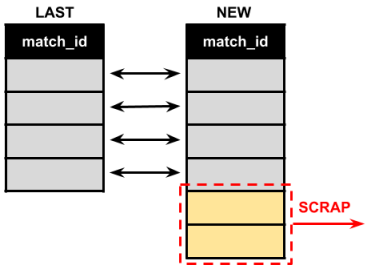


Fig. 2: Data frames update

In order to delimit the field of study, we have set some restrictions in order to filter the scraped data by:

1. Five best european leagues
2. Date from February to May

In the following section we will try to simplify and organise the data available to us.

### B. *Data Handling and Preprocessing*

Our main purpose is to select the most relevant features and reduce the 4 data frames to a single main table. But, how can we do that knowing that players are out of order in the different data frames?

It is necessary to find a field or a combination of fields that uniquely identifies each row in the table. This column is called primary key, which will let us join tables as we want.

A primary key would be the previously mentioned *match_id*, which is an identifier of the football duel. But, as many people play the same match, it will be repeated many times and will not be enough to characterise a row. We thought that the combination of the *match_id* with each player name could be unique for each row, and actually it is.

As you know, there is a large quantity of data to be joined -45.000rows x 98columns, approximately-. Instead of considering the players name -which is format string- as a primary key, we decided to generate a numerical identifier for each player, called *player_id*. We did also the same with each team, and we named it *team_id*. Numerical primary keys can result in more efficient storage and faster query performance.

Now, we can proceed to join the 4 tables to a single main table called *df_project* through the combination of primary keys *match_id* and *player_id*. It is important to note that we are applying an inner join, which returns only the rows that have matching values in both tables being joined. It excludes non-matching rows from the result set. So, it could happen that the main table *df_project* has less rows due to possible small irregularities in the data frames, which in this case, were negligible.

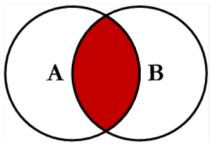


Fig. 3: Inner join

The inner join procedure using the combination of primary keys *match_id* and *player_id* is shown in the figure below.

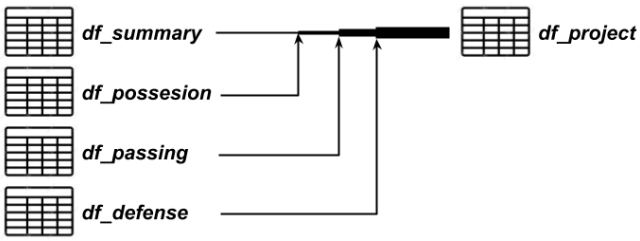


Fig. 4: Inner join procedure

Once we built the *df_project*, the next step was to delete several repeated columns. As it has a total of 98 columns, we also reorganised them as to be slightly distributed by groups and facilitate the task of finding them if necessary. The left region columns correspond to the player intrinsic features which are constant over time -such as name, age, nation, etc.-, and the right located ones are those that belong to a specific match and tend to vary throughout the matches.

We also considered it appropriate to split some columns into two, since they have double information separated by a hyphen or a space. These were:

- Field *game* split to *date* and *game*, which were in a single column separated by a space
- Field *league* split to *country* and *league*, which were in a single column separated by a hyphen

- Field *age* split to *age* and *age_days*, which were in a single column separated by a hyphen

Next, all rows with a null cell were removed. Any fields containing percentages or ratios between columns have been also eliminated in order to avoid future problems such as, for example, the generation of infinite values caused by division by zero.

As we are seeing, we have a large amount of information but, what about the *injury* column? This field, which in fact was the label for supervised methods implemented in next steps, is not provided in this data set. That meant that we had to manually built our own data frame with injuries events. This consisted of looking up the injuries for the months from February to May on the *BeSoccer* website, and writing down each injury with its date and the corresponding *player_id* in a .csv file. We recollected a total of 446 injuries and stored them in a data frame called *df_inj*. It is important to note that we have only considered the non-contact injuries -the muscular ones- and excluded the contact-injuried, which tend to be random.

What we did after that was to apply a left join between *df_project* and *df_inj* in order add the column *inj* and automatically label the injured samples with an integer 1. As it is a left join, it will return all the rows from the left table and the matching rows from the right table. If there is no match, it includes NULL values for the columns of the right table. Those NULL values correspond to the not-injured samples, so we replace them by the integer 0.

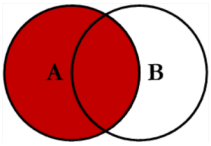


Fig. 5: Left join

The left join procedure using the combination of primary keys *date* and *player_id* is shown in the figure below.

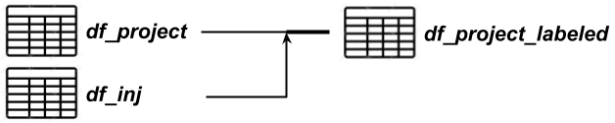


Fig. 6: Left join procedure

Now we dispose of the main complete data frame *df_project_labeled* which, as its name says, its samples are currently labeled.

We can affirm that we have built our own data set by means of scraping tabular information from the Internet and a tough manual labeling task. But this is not finished yet: as you know, in the future CNN and SVM algorithms will be applied and compared. That means that, from now on, we will carry out the last adjustments according to CNN and SVM separately, as they require of different data structures and formats to be inputted.

### *1) CNN: Random Forest feature selection and temporary windows arrangement*

As we know, a Convolutional Neural Network (CNN) works mainly with structured data in the form of images or matrices. At this point, we disposed of a huge data frame with approximately 45.000rows x 98columns size.

Our idea was to conserve the temporary sense of data to be executed in a CNN. We thought that this temporary sense could be very interesting and useful in order to predict an injury: at least, the five last games must be considered to check the player's historical data and also its statistical evolution over the matches. That is the reason why we focused on building data portions of 5 consecutive games for each player, which we call temporary windows.

First, we carried out a feature selection process through Random Forest model in order to reduce the number of fields. We were sure that many of them were not relevant for the study and could be removed to reduce noise, time and computational costs. The columns were reduced from 98 to 24, on which we added the *player_id* and the label *inj*. That results in a total of 26 columns.

After that, we scaled the data to avoid bias, improve model performance, and ensure equal importance of features. Of course, the fields *player_id* and *inj* were excluded from the scaling, as they are the identifiers and the labels, respectively.

Now, we are able to obtain the injured and not injured temporary windows data frames.

| | id_player | age | age_days | pos | min | touches | pass_succ | pass_att | pass_progg | carries | ... | inj |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4512 | 200.0 | 0.56 | 0.521978 | 0.916667 | 0.000000 | 0.006135 | 0.006897 | 0.006452 | 0.035714 | 0.007194 | ... | 0.0 |
| 7136 | 200.0 | 0.56 | 0.557692 | 0.666667 | 0.146067 | 0.049080 | 0.041379 | 0.051613 | 0.035714 | 0.043165 | ... | 0.0 |
| 8416 | 200.0 | 0.56 | 0.576923 | 0.916667 | 0.067416 | 0.006135 | 0.000000 | 0.006452 | 0.000000 | 0.007194 | ... | 0.0 |
| 12437 | 200.0 | 0.56 | 0.640110 | 0.916667 | 0.056180 | 0.018405 | 0.006897 | 0.019355 | 0.000000 | 0.007194 | ... | 0.0 |
| 18829 | 200.0 | 0.56 | 0.714286 | 1.000000 | 0.000000 | 0.030675 | 0.013793 | 0.025806 | 0.035714 | 0.014388 | ... | 0.0 |

a) Temporary window *inj=0*

| | id_player | age | age_days | pos | min | touches | pass_succ | pass_att | pass_progg | carries | ... | inj |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13357 | 100.0 | 0.76 | 0.508242 | 0.416667 | 1.000000 | 0.668712 | 0.627586 | 0.658065 | 0.571429 | 0.575540 | ... | 0.0 |
| 14880 | 100.0 | 0.76 | 0.527473 | 0.416667 | 1.000000 | 0.392638 | 0.296552 | 0.361290 | 0.071429 | 0.251799 | ... | 0.0 |
| 16365 | 100.0 | 0.76 | 0.546703 | 0.416667 | 1.000000 | 0.337423 | 0.289655 | 0.316129 | 0.321429 | 0.280576 | ... | 0.0 |
| 17158 | 100.0 | 0.76 | 0.554945 | 0.416667 | 1.000000 | 0.490798 | 0.448276 | 0.477419 | 0.178571 | 0.424460 | ... | 0.0 |
| 18791 | 100.0 | 0.76 | 0.568681 | 0.416667 | 0.213483 | 0.036810 | 0.034483 | 0.032258 | 0.035714 | 0.028777 | ... | 1.0 |

b) Temporary window *inj=1*

Fig. 7: Temporary windows

We obtained a total of 10.202 windows with *inj*=0 and 252 windows with *inj*=1. This fact was expected since, in real life, its usual that a very reduced number of players get non-contact injuries. In our data frame, only the 17,35% of the players had suffered an injury, but we dispose a total of 19,59% injured samples. That means that, some of them, had

suffered an injury more than one time. We can affirm, as it was expected, that we were working with unbalanced data. We had to maintain this ratio of unbalancing also with the temporary windows. That is why we randomly reduced the number of not injured windows from 10.202 to 1.260.

The next step was to convert these data frame temporary windows to .jgp image format to be inserted in the CNN and also have a visual perception of data in a gray scaled image.

a) Window image *inj=0*

b) Window image *inj=1*

Fig. 8: Window images

As it can be seen, their dimensions are 5x24 since we removed the *player_id* and *inj* fields, because identifiers should not be considered as a feature and injuries were the labels.

#### *2) SVM: Cumulative data frame arrangement and Principal Component Analysis*

SVMs operate on feature vectors, which are numerical representations of the input data. However, SVMs are not inherently designed to handle tabular data or image data directly. That means that we can't use the previous windows data frames.

That means that we have to find a way of conserve that temporary sense in a single vector of features. The best idea was to modify our main data frame *df_project_labeled* to obtain another one, in which for each row, we were taking into account the whole history player through the cumulative. Of course, we have excluded from the cumulative the fields *age*, *age_days*, *pos* and the label *inj*, as they did not make sense to add them up. This time we considered a maximum of 1 injury per player, which corresponds to the first one, as we set the accumulation to stop once the *inj* label was equal to 1.

This modified data frame was called *df_project_cum*. It contained a total of 10.202 not injured samples and 223 injured samples. The number of injured samples was reduced as we are only considering the first injury for each player, as it was said before. We know that SVM can work with unbalanced data, but we wanted to maintain the 20% injured - 80% not injured samples ratio. It is not recommended to use big amounts of data with SVMs. This is because they are sensitive to outliers, which can affect the accuracy of the model. After discussing with professionals in the field, we concluded that the quantity of samples should be around 1.000. So, *df_project_cum* is a data frame with cumulative samples, which 777 of them are not injured and 223 are injured.

The current samples present too much features to be inserted in a SVM: as the number of features grows, it can become more difficult to find an optimal hyperplane due to the phenomenon known as the "curse of dimensionality". To solve that, we scaled the data and applied PCA in order to capture as many features as necessary to cover the 95% of variability. The columns *player_id*, *age_days* and *inj* were previously removed. Once we projected the data in the PCA space, we observed that with 7 components was enough to represent that 95% of variability. So, the final *df_project_cum* presented a total of 1.000 cumulative samples with 7 features for each of them.

### *C. Convolutional Neural Network - CNN*

Convolutional Neural Networks (CNNs) work better with square images because they preserve translation invariant, allow for symmetric network architectures, and ensure consistent pooling operations, resulting in efficient feature extraction and improved performance. That is the reason why they have been resized to 24x24 dimensions. The data has been split 80% train and 20% test.

After a long process of experimentation and modification of the CNN architecture in order to obtain the best possible results, it has been concluded that the best option is as follows:

- Conv2D layer with 8 filters of 3x3 size and ReLU activation
- MaxPooling2D layer with a pool size of 2x2 for down-sampling
- Dropout 20% layer for regularization
- Flatten layer
- Dense layer with 16 neurons and ReLU activation
- Dense output layer with 2 neurons and SoftMax activation
- Dropout 30% layer for regularization

The model was compiled with the BinaryCrossentropy loss function, as we were working in a binary classification problem. We also used Adam optimizer in which we had to decrease the learning rate to 0.00001 in order to avoid abrupt slope changes between epochs.

We thought that F1_score was the best choice as it is the harmonic mean of precision and recall, providing a balanced measure of a model's accuracy. It helps to evaluate the model's ability to achieve a good trade-off between correctly identifying positive instances and minimizing false positives and false negatives. Our purpose was to get right as many test samples as possible, but bearing in mind avoiding the false negative results as we were working with an injury

prevention problem.

As our data was unbalanced, we had to set weights in order to assign different importance or emphasis to the binary classes during the training of the CNN model. By assigning higher weights to the minority class (in this case, injured one), the model is encouraged to pay more attention to correctly predicting instances of that class. In contrast, the majority class (not injured one) is assigned a lower weight, reflecting its abundance in the data set. After modifying these weights manually, we realised that they were very sensitive: if we increased the injured class weights, the model was paying much more attention to this minority class and it predicted always injured class. By contrast, by decreasing the weights of the injured class, the model always predicted 0. So we had to precisely find the correct value without biasing the model. Finally, weights were set according to the percentage of samples of its class. So, the weights of injured class samples were multiplied by 5 to equalize the number of not injured samples and resemble a balanced model.

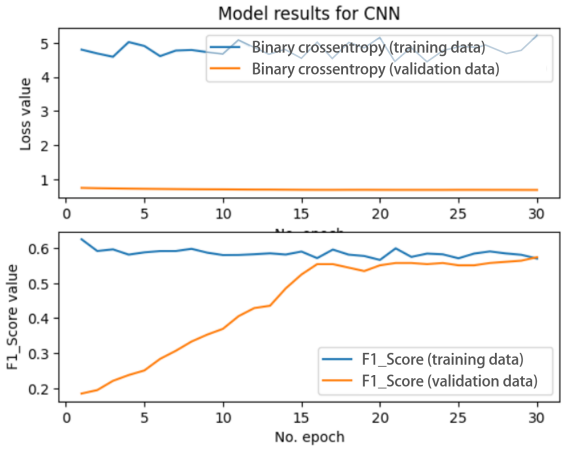


Fig. 9: Loss and F1_score values

We obtained a F1_score value of 57%. Its evolution over the epochs is shown above. As it can be observed, the loss value presents a very small decrease from 0,75 to 0,68.

To conclude, we plotted the corresponding confusion matrix which is very important to check the false negatives obtained value and if the model is always predicting the same class -due to really sensitive weights problem, which was explained before-.

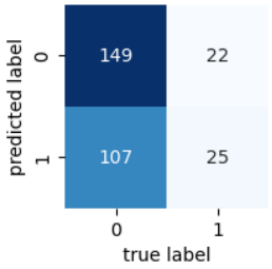


Fig. 10: CNN confusion matrix

It can be observed that 57,42% of the samples were well classified while, from the ones that were missclassified, only the 17% correspond to false negatives. In other words, it is a 7,3% of false negatives over the total of samples.

### *D. Support Vector Machine - SVM*

Again, data has been split 80% train and 20% test.

We have used the Support Vector Classifier (SVC) which is a specific implementation of SVM for binary or multi-class classification from the scikit-learn library for the binary classification problem. As we are dealing with an unbalanced data set, we have defined class weights again as a dictionary. The not injured class samples had a weight of 1 while the injured class samples had assigned a value which was approximately 3,5 which can slightly vary depending on the number of samples of each class in the train set.

We used the sigmoid kernel as it is often suitable for binary classification tasks in Support Vector Machines (SVM) due to its ability to model non-linear decision boundaries. The sigmoid function transforms the input data into a range between 0 and 1, allowing the SVM to capture complex relationships between the features and the target variable.

The cost parameter C had been set to a very high value to impose a strict penalty for misclassification errors on the training data. This means that the SVM aimed to achieve a smaller training error by allowing a smaller margin or even potential misclassifications. It was fixed to C = 1E10.

```
SVC
SVC(C=10000000000.0, class_weight={0: 1.0, 1: 3.6242774566473988},
    kernel='sigmoid')
```

Fig. 11: SVM model

After training the model, we obtained the following confusion matrix according to the test samples.

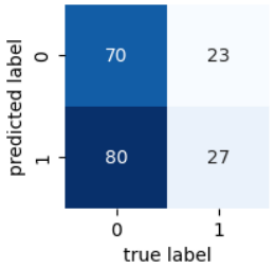


Fig. 12: SVM confusion matrix

The F1_score is 34,39%. We can also observe how the 48,5% of the test samples were well classified. According to the false negatives, we have obtained a total of 22,33% percentage over the missclassified test samples, which is the same as 11,5% of false negatives over the total of samples.

## V. Conclusion

This paper presented an approach to football injury prediction. The data was acquired from the *FBref* website by using a scraper. This data was in 4 different tables that were joined to create the data set, later on it was processed to eliminate rows with null values and split some columns into more understandable ones. The labelling had to be a manual process, extracting the injury dates from *BeSoccer*, to complete the labelled data set. In this data set, each sample had 98 features. After that, this data set had to be manipulated in order to reduce its dimensionality and to adapt it to the model used (CNN or SVM in our case).

For the CNN, Random Forest feature selection was applied and then, temporary windows were created to pass them as images. After that, the model was created and modified following and iterative process and trained, obtaining a F1_score value of 57%, accuracy of 57.42% but only 17% of false negatives over the samples that were missclassified, which is the same as 7,3% of false negatives over the total of samples –which is really important to maintain a low percentage as we are working with injury prevention. For the SVM, first, the cumulative sum of the rows had to be computed and then, PCA was applied to reduce the dimensionality to just 7 features – which represented 95% of the variability. After that, the model was created and modified following an iterative process and trained, leading to 48.5% accuracy and 22.33% of false negatives over the missclassified samples. That means a 11,5% of false negatives over the total of samples.

Both models seem similar, but the CNN obtains a better ratio of false negatives and accuracy, which is the same as a better F1_score. It is important to highlight that both models present a large percentage of false positives, but is a lowest percentage in the case of the CNN. As last comment, we think that the data used for this study was not the best fitted in order to determine the injury risk of players, because it leaves a lot of aspects behind, such as training physical load, illness suffered and so on.

## References


[1] S. Editors, "Top-10 most popular sports in the world 2023," Jan 2023. [Online]. Available: https://sportytell.com/sports/most-popular-sports-world/

[2] "The rising volume of games per year and the correlation with increasing injuries to football/soccer players," Feb 2023. [Online]. Available: https://erkutsogut.com/blog/2023/02/27/the-rising-volume-of-games-per-year-and-the-correlation-with-increasing-injuries-to-football-soccer-players/

[3] "Player injuries cost europe's clubs over £500m," Sep 2022. [Online]. Available: https://fcbusiness.co.uk/news/player-injuries-cost-europes-clubs-over-500m/

[4] Gitnux, "The most surprising soccer injuries statistics and trends in 2023," Apr 2023. [Online]. Available: https://blog.gitnux.com/soccer-injuries-statistics/

[5] E. Eliakim, E. Morgulev, R. Lidor, and Y. Meckel, "Estimation of injury costs: Financial damage of english premier league teams' underachievement due to injuries," May 2020. [Online]. Available: https://bmjopensem.bmj.com/content/6/1/e000675

[6] "Howden's european football injury index reveals record injury cost of over £500m for 2021/22 season," Sep 2022. [Online]. Available: https://www.howdengroup.com/news-and-insights/howdens-european-footbal-Injury-Index-reveals-record-injury-cost-of-over-500m-for-2021-22-season

[7] A. Majumdar, R. Bakirov, D. Hodges, S. Scott, and T. Rees, "Machine learning for understanding and predicting injuries in football - sports medicine - open," Jun 2022. [Online]. Available: https://sportsmedicine-open.springeropen.com/articles/10.1186/s40798-022-00465-4

[8] T. Piłka, B. Grzelak, A. Sadurska, T. Górecki, and K. Dyczkowski, "Predicting injuries in football based on data collected from gps-based wearable sensors," Jan 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1227

[9] A. Rossi, L. Pappalardo, P. Cintia, F. M. Iaia, J. Fernàndez, and D. Medina, "Effective injury forecasting in soccer with gps training data and machine learning," Jul 2018. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0201264

[10] A. Naglah, F. Khalifa, A. Mahmoud, M. Ghazal, P. Jones, T. Murray, A. S. Elmaghraby, and A. El-baz, "Athlete-customized injury prediction using training load statistical records and machine learning," in *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2018, pp. 459–464.

[11] E. Vallance, N. Sutton-Charani, A. Imoussaten, J. Montmain, and S. Perrey, "Combining internal- and external-training-loads to predict non-contact injuries in soccer," *Applied Sciences*, vol. 10, no. 15, p. 5261, Jul 2020. [Online]. Available: http://dx.doi.org/10.3390/app10155261

[12] Rommers, Nikki and Rössler, Roland and Verhagen, Evert and Vandecasteele, Florian and Verstockt, Steven and Vaeyens, Roel and Lenoir, Matthieu and D'Hondt, Eva and Witvrouw, Erik, "A machine learning approach to assess injury risk in elite youth football players," *MEDICINE AND SCIENCE IN SPORTS AND EXERCISE*, vol. 52, no. 8, pp. 1745–1751, 2020. [Online]. Available: http://dx.doi.org/10.1249/mss.0000000000002305

[13] M. Venturelli, F. Schena, L. Zanolla, and D. Bishop, "Injury risk factors in young soccer players detected by a multivariate survival model," *Journal of Science and Medicine in Sport*, vol. 14, no. 4, pp. 293–298, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1440244011000442