Creating an Expected goals (xG) model

Héctor Fabián Cuevas López

June 13, 2024

1 Project Overview

Expected goal (xG) model is a statistical method used to measure the quality of scoring opportunities in football (soccer). It assigns a probability value between 0 and 1 to each shot taken during a match, based on factors such as the location of the shot, the type of shot, and the situation leading up to the shot. The higher the xG value of a shot, the more likely it is to result in a goal.

An xG model uses historical information from thousands of shots with similar characteristics to estimate the likelihood of a goal. Using data from different competitions provided from Statsbomb, we managed to create our own xG model using Python and the library Scikit-Learn to train it.
Statsbomb has a big data collection of competitions availables for free, where we can find, among others the FIFA World Cup, but in this case we would like to obtain data from La Liga, the main soccer competition in Spain, to be exact from the seasons of 2010 to 2020, ten years of data is a big enough data to get a suitable model in order to predict the xG for players who have played in this competition.

The data provided d

2 Introduction

2.1 Main Features

The success or not success of a shot depends on several variables that changes every minute of a match like the pressure on the player given the minute or even the position of him, like if there is a defender who has the chance, because is well known that attackers and defenders have different qualities, however, there are some variables that never change. We based our model in two main parameters to consider when any type of shooting is performed.

- **Distance to goal :** The distance to goal is essential when a player is about to shoot, a larger distance translates in a smaller target, that´s why soccer teams seek to approach as much as they can to the goal. The data provided by Statsbomb includes the position of the ball at every moment of the game, this includes the position at the moment of the shot Using the x and y coordinates in the pitch we can define a function to calculate the distance. Knowing that the dimensions of the pitch are 120x80 is clear that the center of the goals are located on the point (120, 40) So we have two points which we can transform into a vector and simply calculate its norm using the math library of python.
- **Angle to goal:** As we can see in the figure, a closer location to the goal produces a bigger angle when a player shoot, but also we need to consider the zone where its performed. A shoot too close to the goal line could have a smaller angle if it is made far from the central zone.

3 Exploratory Analysis

3.1 Data Exploration

Once we obtained all the data events from the leagues from 2010 to 2020 of Spain we split the data event in the type shoot.

With all the information provided is easy to know when a shoot ends as a goal, we used this to find the areas where most of the goals are shot.

We also can make a statistical analysis on this data and this can make it easier for the reader to understand why we chose these parameters as the main ones.

3.2 Exploratory Visualization

Our main parameters are the angle and distance where the shots are made. Visualizing all the shots we can see a clear pattern.

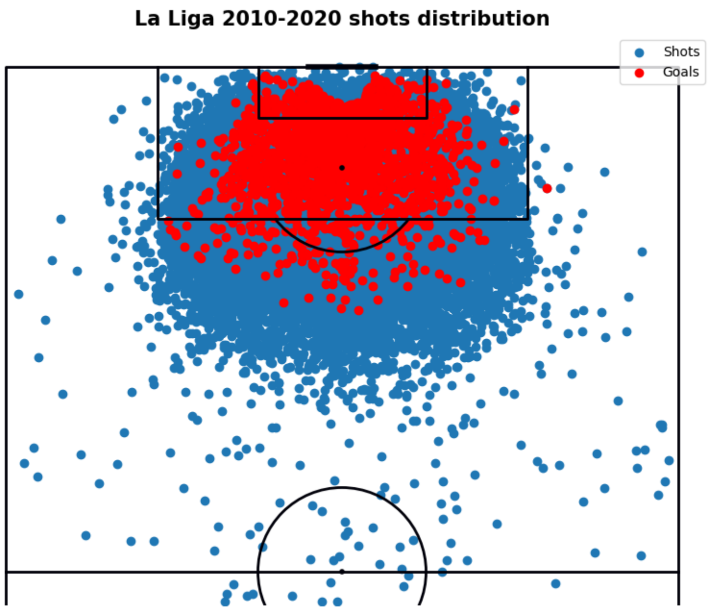


Figure 1: Shots distribution in La Liga seasons 2010 to 2020

Most of players shoot from inside the box and the surrounding areas without getting too close to the goal line because this reduces the angle and decrease the chances of getting a good shot.
For a better visualization of the shooting zone, we can use hexagons to plot the ideal zone. this is where most of the goals were made.

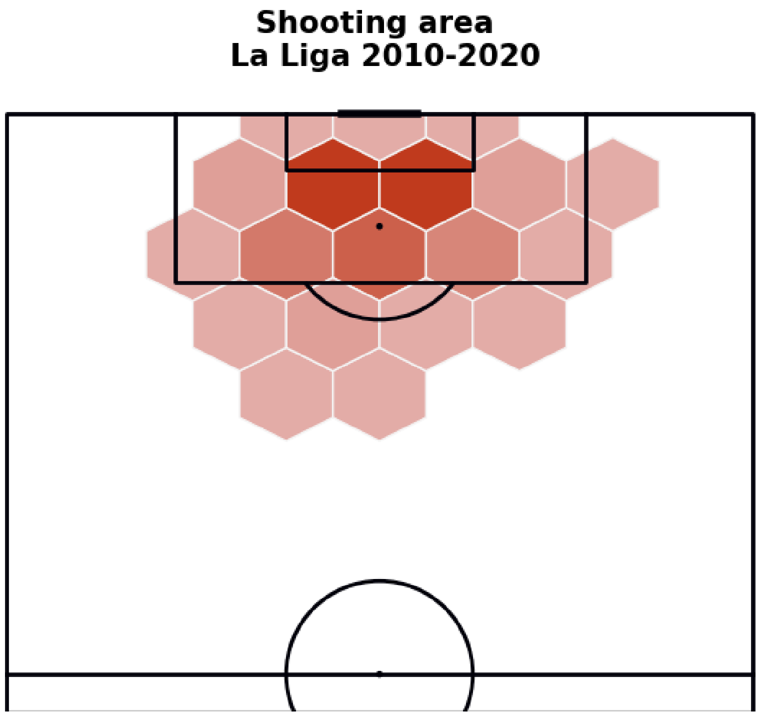


Figure 2: Preferred areas to shot to goal

The distribution of the shots suggests there is a normal distribution which can be graphed, first let´s see it with a QQPlot.

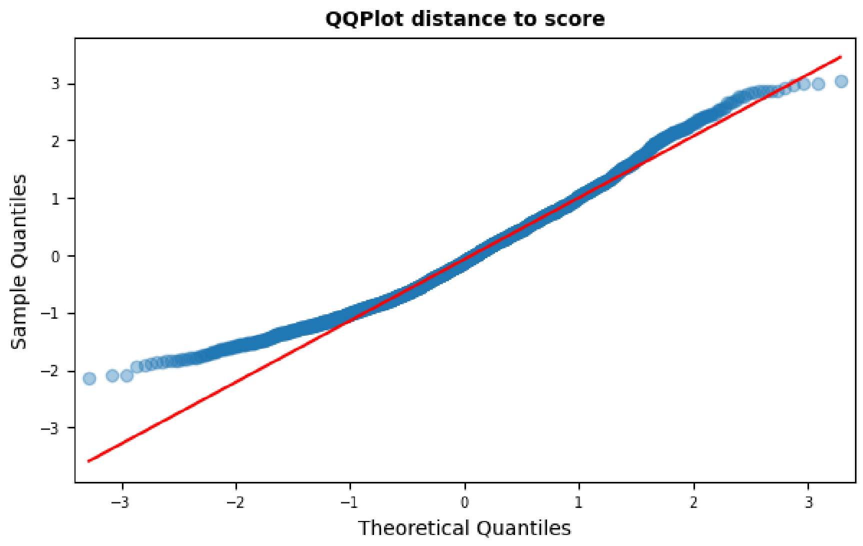


Figure 3: QQPlot distance to score

The shape of the points following the line almost completely suggests that indeed the distance has a normal distribution. To verify, we can plot the histogram of the values of distances.

And to see more clearly the distribution we added the normal line

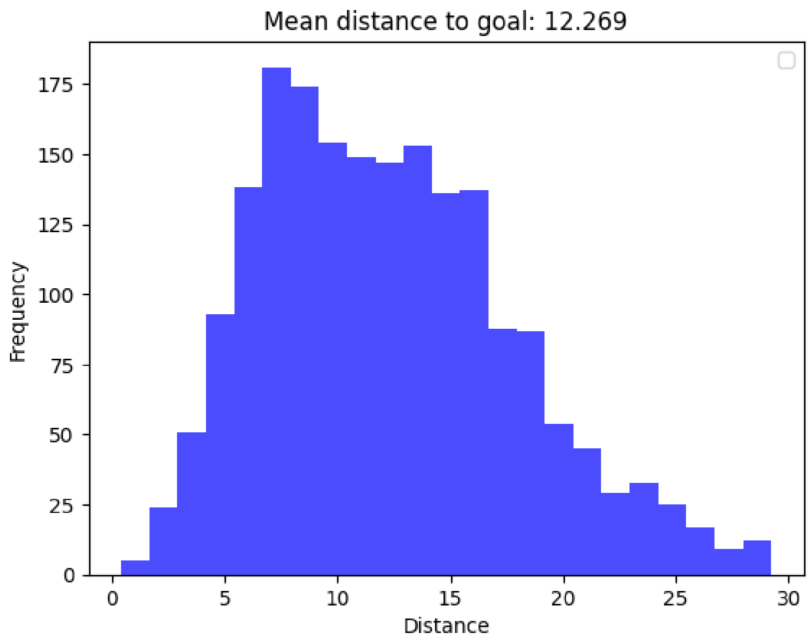


Figure 4: Histogram of the distances of goals

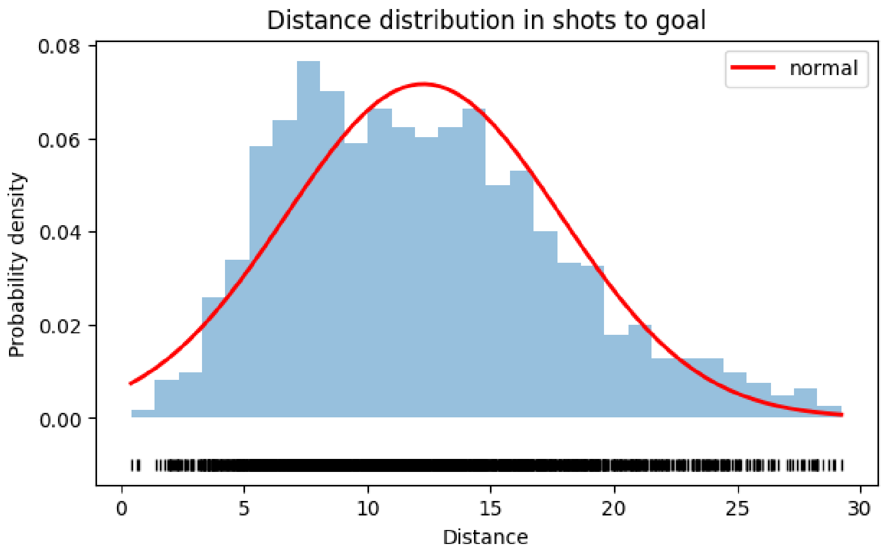


Figure 5: Distance distribution

4 Implementation

The angle and distance to goal are not features included in the data, but we have the position at the moment of the shoot given in coordinates x and y.

To calculate these we defined two functions in order to apply them to the entire dataset.

4.1 Distance

In order to calculate the distance to the center of the goal we need to know what are the coordinates of the goal. The library mplsoccer facilitates the creation of a pitch with the exact dimensions of a real one. Knowing the pitch dimensions we can find the posts of the goal located in the points (120, 36) and (120, 44), it is easy to see that the middle point between these two is the point (120, 40) where we want to know the distance. Drawing a vector from the shot point to this one we can easily find the distance with its norm.

$$\|\vec{AB}\| = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2} \tag{1}$$

Fortunately Python includes a function to calculate norm without writing the entire expression.

4.2 Angle

The procedure is quite similar but in this case, the objective is create two vectors with origin in the point where the shot is made. Drawing a vector to each post, the angle between these two is easily calculated with the dot product. Let A the vector from the shot to the post 1 and B the vector from the shot zone to the post 2, the angle can be calculated with

$$\vec{A} \cdot \vec{B} = ABCos\theta \tag{2}$$

By clearing θ we have

$$\theta = Cos^{-1}(\frac{\vec{A} \cdot \vec{B}}{AB}) \tag{3}$$

With our two functions defined it's easy to apply them to the shots dataframe.

4.3 Under Pressure

When a player faces the goal without any obstruction is easier for him to shot to goal, likewise if he is pressed by an opponent. Soccer is a sport of contact. defenders will do anything to prevent you from getting a clear shot.

Thinking of it we added the feature Under Pressure where the data says if the player is being pressed or not. In this case we have a number when the player is under pressure and a null value when he is not. Changing the null values for zeros we can now assemble our model with all features represented with numbers.

4.4 Algorithms

Determining if a shot is or not a goal is a classification problem because the goal is to predict when a shot is or not a goal. In this sense, is correct to use algorithms such as Logistical Regression, Decision Trees, Naive-Bayes, among others.

- Logistic Regression: Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false
- Decision Trees: A decision tree is a decision support hierarchical model that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.
- Random Forest Classifier: This method combines the output of multiple decision trees to reach a single result.
- Naive-Bayes: Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

4.5 Choosing the best model

All the algorithms are suitable for our model, but it is necessary to know which can give us the best results. By using different scores we can compare the models.

- Accuracy: The difference between a measured value and the true value.
- Precision: The dispersion of the set of values obtained from repeated measurements of a magnitude.
- Recall score: Also known as the true positive ratio, is used to find out how many positive values are correctly classified.
- F1 Score: Measures a model's accuracy. It combines the precision and recall scores of a model.

Also we plotted graphic representations as the Confusion Matrix. In machine learning, to measure the performance of the classification model, we use the confusion matrix.

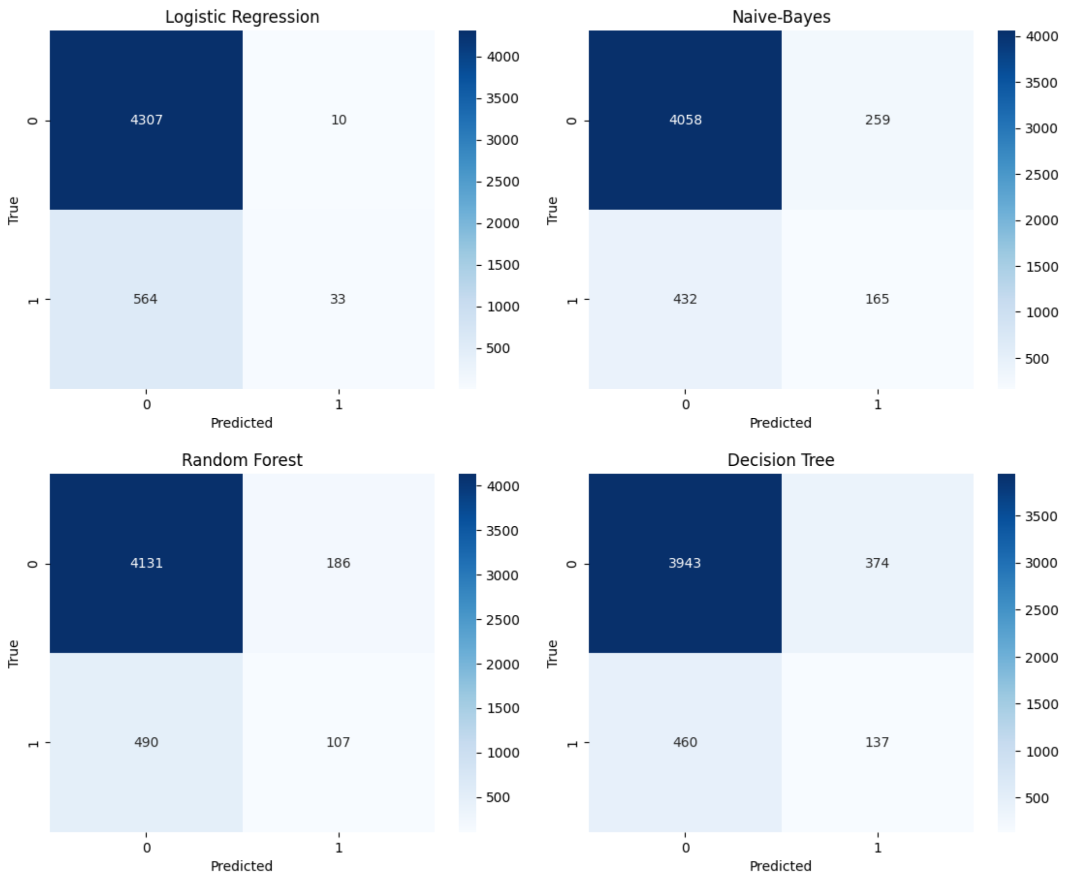


Figure 6: Confusion matrices from all methods used to test the data

The matrix displays the number of instances produced by the model on the test data.

- True positives (TP): occur when the model accurately predicts a positive data point.
- True negatives (TN): occur when the model accurately predicts a negative data point.
- False positives (FP): occur when the model predicts a positive data point incorrectly.
- False negatives (FN): occur when the model mispredicts a negative data point.

Finally we would like to see how the model behaves at the great quantity of data that we used. To see this more clearly we used the Learning Curves. Learning curves are plots used to show a model's performance as the training set size increases.

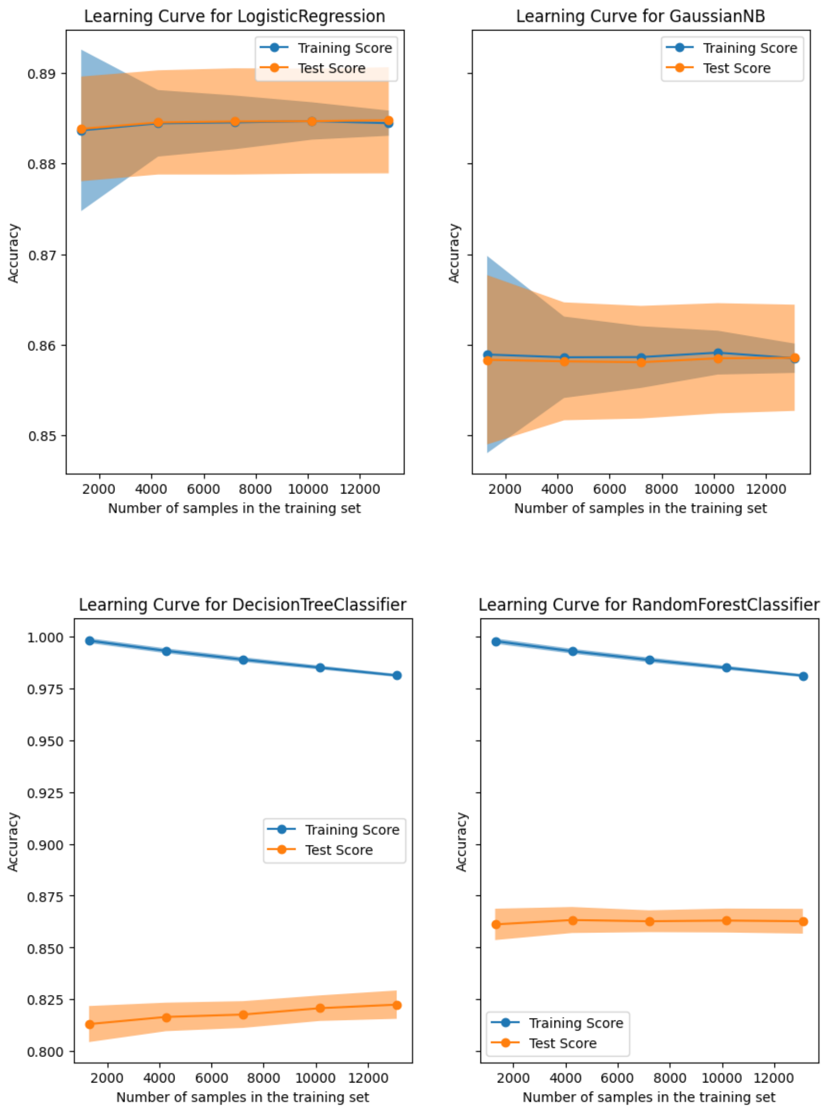


Figure 7: Learning curves for the algorithms used to test the data

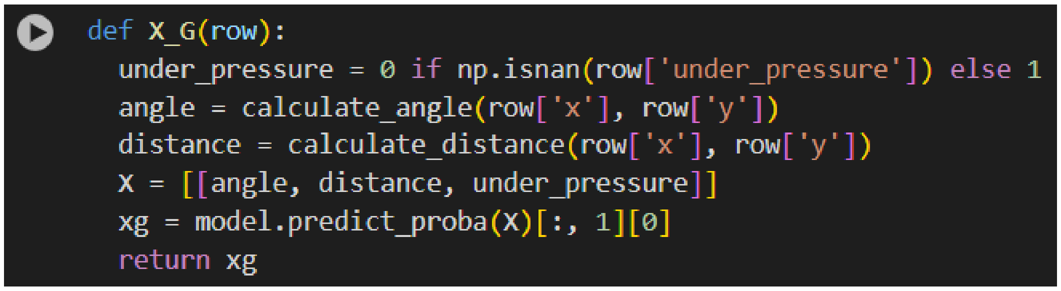
The Decision Tree and Random Forest models makes very few mistakes when it's required to predict instances it's seen during training, but performs terribly on new instances it hasn't been exposed to. We can observe this behavior by noticing how large the generalization error is between the training curve and the validation curve. For that reason, we will discard them, but not completely. We can notice that in the Decision Tree Classifier model, the curves are pretty symmetric, it may have a bigger accuracy if we add more data. So this model may be used further.

The model that seems to have a better performance are the Logistic and Gaussian NB, but the Logistic has a bigger accuracy as we can see in the table.

For this reason, our model was trained with the Logistic Regression algorithm.

5 Deploying the model

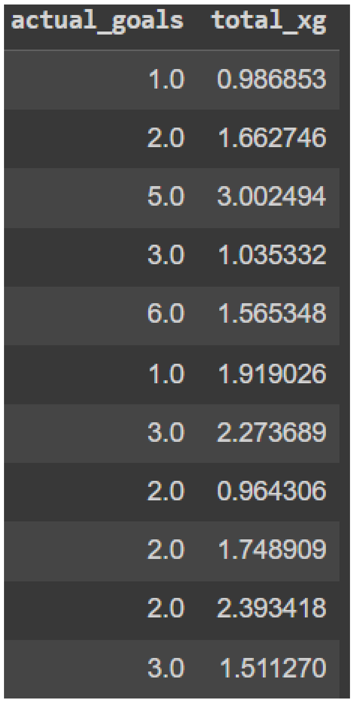
once our data is ready, our features are defined and we have chosen our algorithm, we defined a function to calculate our xG, using the Logistical Regression algorithm, and easily apply it to the entire dataset.



```python
def X_G(row):
  under_pressure = 0 if np.isnan(row['under_pressure']) else 1
  angle = calculate_angle(row['x'], row['y'])
  distance = calculate_distance(row['x'], row['y'])
  X = [[angle, distance, under_pressure]]
  xg = model.predict_proba(X)[:, 1][0]
  return xg
```

Figure 8: The function XG calculates the probability for a shot to end as a goal

Let´s see how it works comparing the expected goals (our model) and the actual goals scored grouping the events by match using the match id available on the dataset.



actual_goals	total_xg
1.0	0.986853
2.0	1.662746
5.0	3.002494
3.0	1.035332
6.0	1.565348
1.0	1.919026
3.0	2.273689
2.0	0.964306
2.0	1.748909
2.0	2.393418
3.0	1.511270

Figure 9: Real and predicted goals

As we can see, our model is very effective when it comes to few goals, we have very good approach like in the first and second rows, but when there were scored many goals our model fails and let us with a very low expected goals.

6 Conclusions

Our model is very simple and only considers two main parameters, a football match is really complex and many variables can change the course of it.

In order to make a best model, we think in add more features, starting by the position of the players, which can represent an obstacle when the shot is made, moreover, it would be good to use the Decision Tree Classifier method to train the data, as we said before, to use it we need way more data, and a good addition to the model would be the data from the five big leagues of Europe and International competitions.

7 References

- https://es.wikipedia.org/wiki/Precisi%C3%B3n_y_exactitud
- https://www.v7labs.com/blog/f1-score-guide
- https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
- https://www.datacamp.com/tutorial/tutorial-learning-curves
- https://www.ibm.com/topics/random-forest