Aplicación de modelos de aprendizaje automático sobre la base de datos FIFA 2022

Introducción al aprendizaje automático - 2022 - FAMAF UNC

María Florencia Molina

La base de datos y el objetivo

Base de datos

Se trabaja con la base de datos del juego FIFA 2022, que cuenta con los datos para más de 19000 jugadores de fútbol con más de 100 atributos, entre los cuales se encuentran datos personales de los jugadores, habilidades en el juego y sus posiciones en los clubes donde juegan y seleccionados nacionales.

Las posiciones serán agrupadas globalmente como:

- Arqueros (GK)
- Defensores (CB, LCB, RCB, LB , RB)
- Mediocampistas (CM, LDM, LAM, RDM, CDM, CAM, LM, RM)
- Atacantes (ST, CF, LW, RW)

Objetivo

Predecir la posición general en el campo de juego de un jugador a partir de sus características generales utilizando un modelo de aprendizaje automático

Análisis exploratorio y curación

Eliminación de datos faltantes

Se encuentran ocho atributos que tienen más del 50% de datos faltantes o nulos, por lo que se desechan estas columnas de la base de datos. Los datos faltantes que restan (que contabilizan no más del 11% de los valores de las columnas en el peor de los casos) se reemplazan con los valores promedio de los datos no nulos.

```
# data.isna() devuelve una matriz de booleanos en la que cada posicion indica si ese valor esta perdido o no. Hacer .sum() suma sobre los valores
# True de esa matriz y haciendo ascending=False le pedimos que nos de las columnas con mas valores perdidos
nans = data.isna().sum().sort_values(ascending=False)
print(nans[nans!=0]/data.shape[0]) #Aca nos da la proporcion de valores perdidos de cada columna
```

```
nation_jersey_number        0.960549
nation_logo_url             0.960549
nation_position             0.960549
nation_team_id              0.960549
club_loaned_from            0.942721
player_tags                 0.925100
goalkeeping_speed           0.889183
player_traits               0.511513
dribbling                   0.110817
shooting                    0.110817
passing                     0.110817
physic                      0.110817
defending                   0.110817
pace                        0.110817
release_clause_eur          0.061126
club_joined                 0.060450
value_eur                   0.003846
club_contract_valid_until   0.003171
league_level                0.003171
club_jersey_number          0.003171
club_position               0.003171
league_name                 0.003171
club_name                   0.003171
wage_eur                    0.003171
club_logo_url               0.003171
club_flag_url               0.003171
club_team_id                0.003171
```

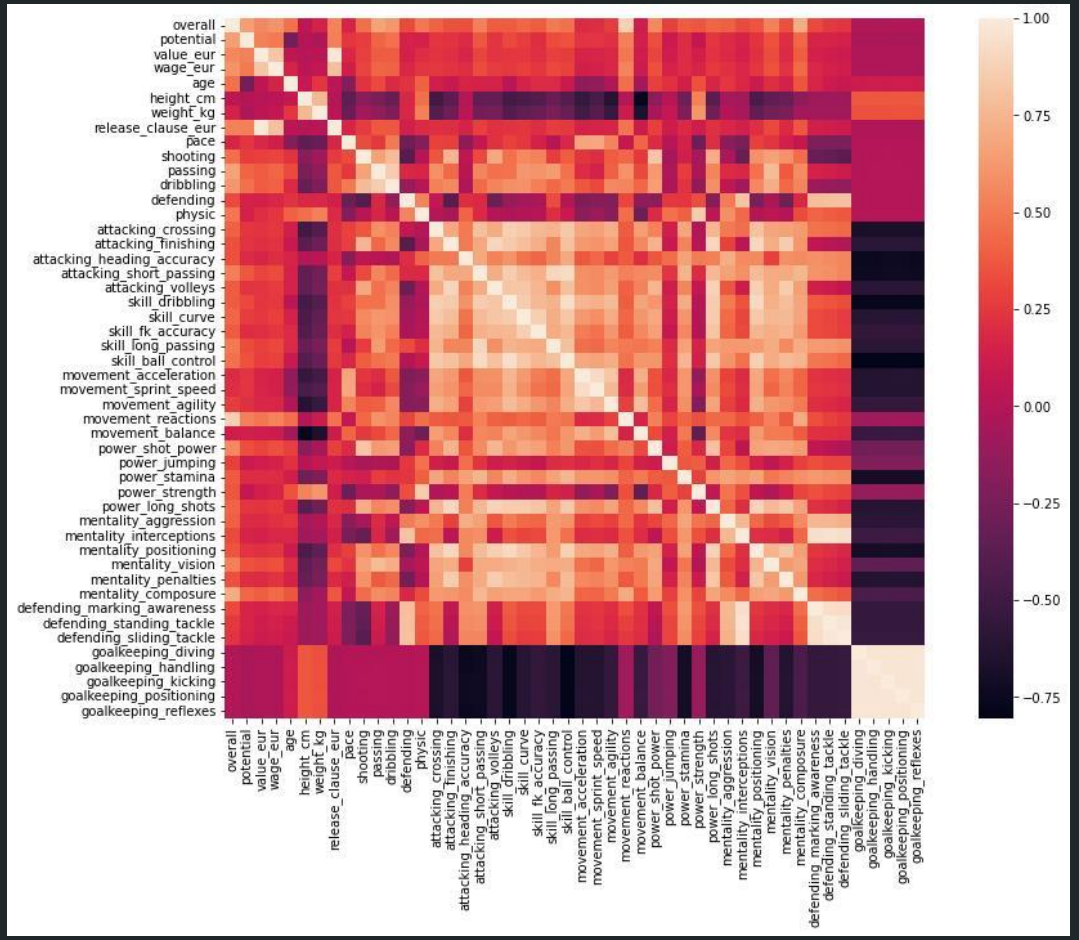
Elección de variables para el problema de clasificación

Distinguimos variables categóricas que consideramos no servirán para el problema de clasificación (por ejemplo la url del sitio web del jugador de fútbol) y nos quedamos con las variables numéricas que aluden a las características físicas del jugador y sus habilidades en el juego. Usaremos estas últimas para nuestro modelo.

Análisis exploratorio

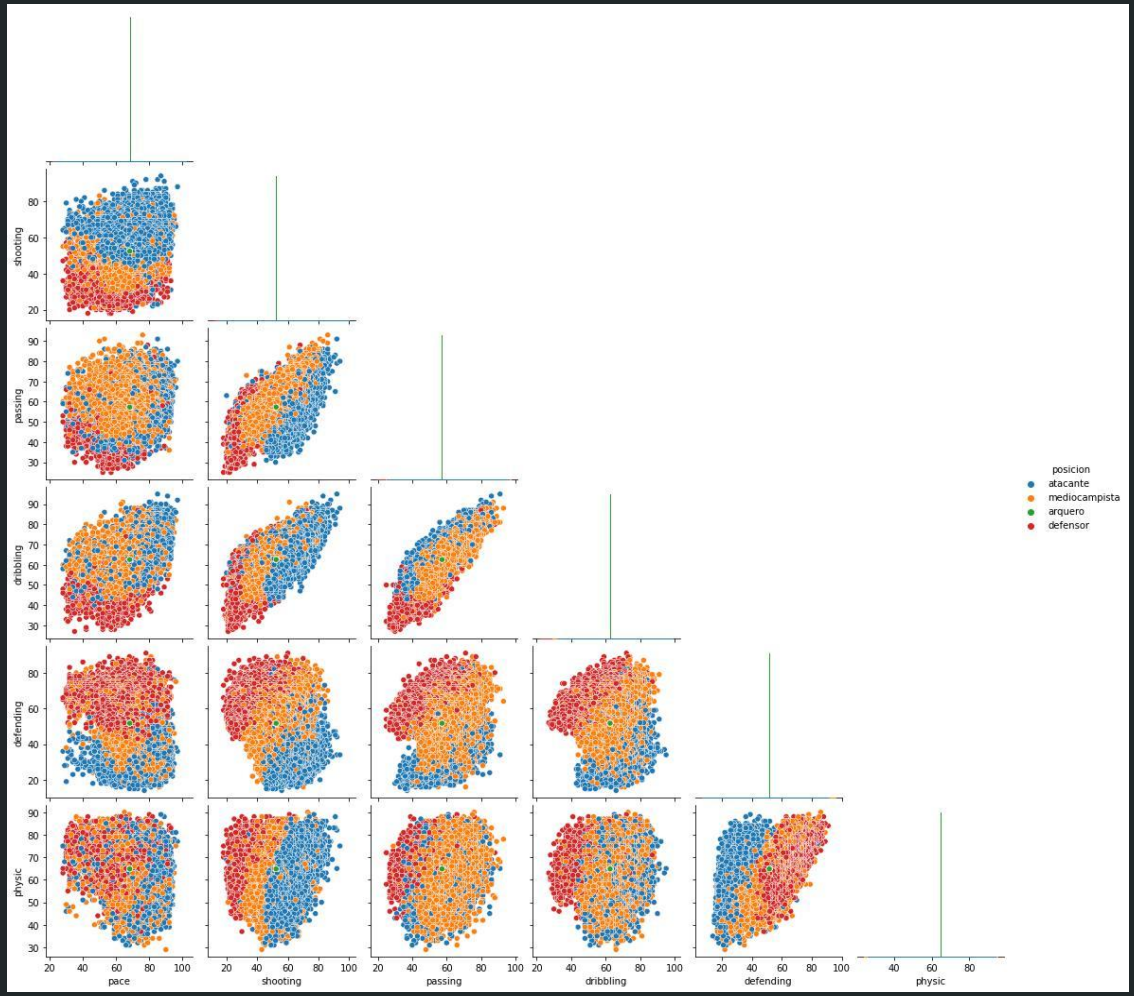
Observamos las distribuciones y los estadísticos de las variables distinguiendo por clases

- Alta correlación entre variables de ataque, movimiento, energía, mentalidad y defensa entre ellas
- Fuerte correlación en las variables de ”goalkeeping” entre sí y correlación negativa de estas variables con otras



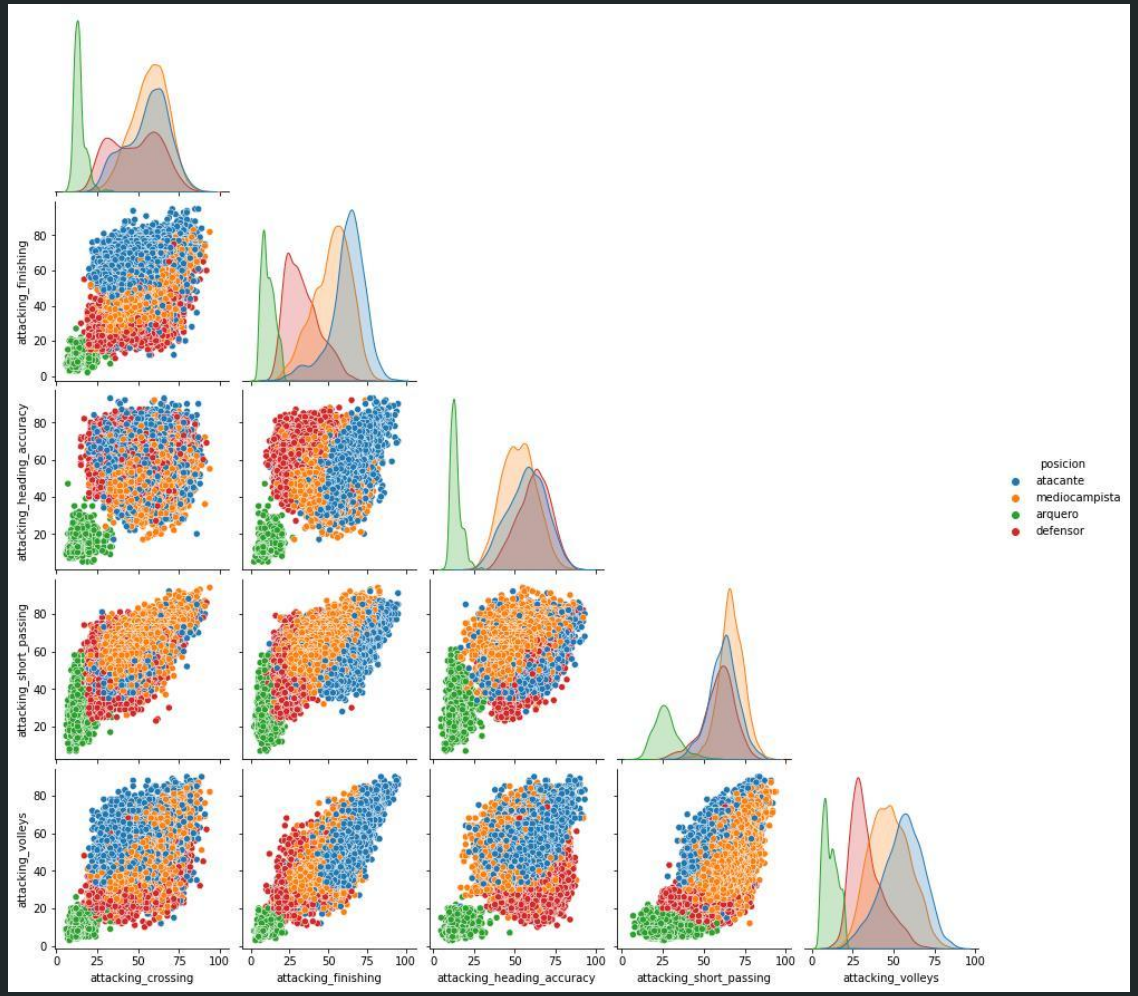
Habilidades generales:

- Correlación aproximadamente lineal entre passing, dribbling y shooting. No hay correlaciones apreciables en las demás variables
- En la mayoría se distinguen las clases claramente, estando esto más acentuado para los defensores
- Todos los arqueros toman un único valor en estas variables



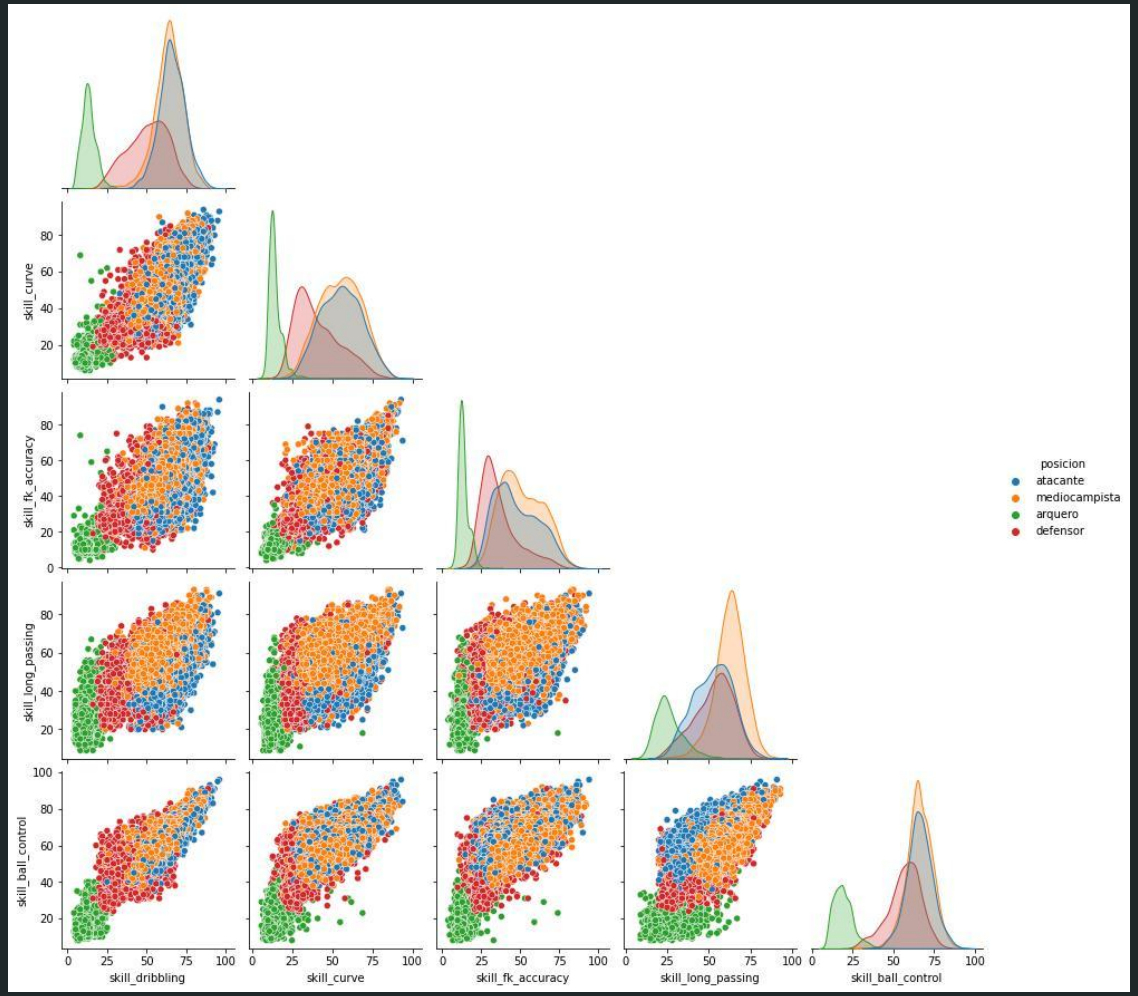
Habilidades de ataque:

- Los arqueros tienen distribuciones muy centradas y alejadas de las demás clases, presentando valores bajos en todas las variables
- Mediocampistas y atacantes poseen distribuciones similares y se agrupan en clusters cercanos
- Los defensores se distinguen más de atacantes y mediocampistas, aunque poseen distribuciones más cercanos a estos que con respecto a los arqueros



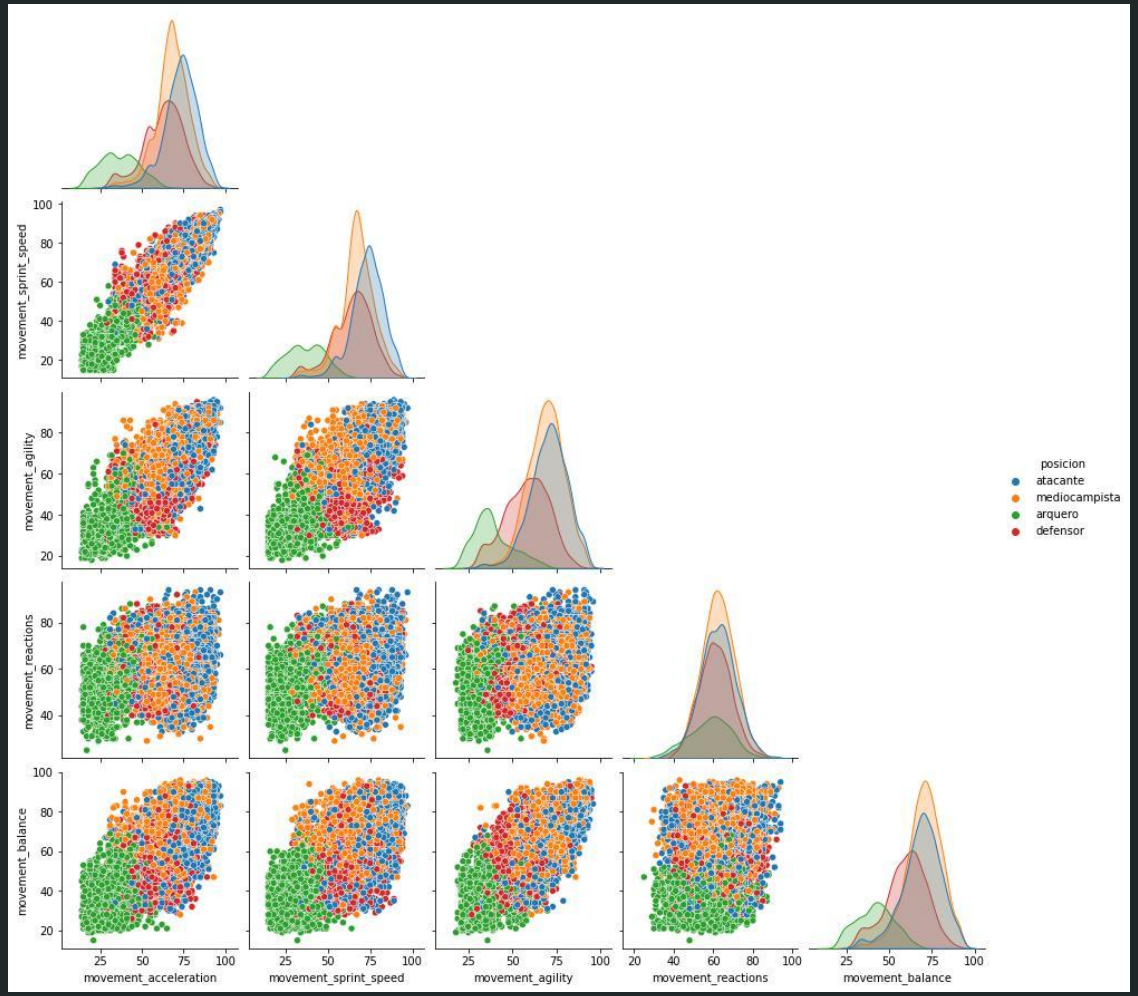
Habilidades de manejo de la pelota

- De nuevo se observan distribuciones similares para mediocampistas y atacantes, valores bajos para los arqueros y distribuciones más distinguidas para defensores
- La habilidad de dar pases largos es la única en la que mediocampistas se distinguen de defensores y atacantes (lo cual tiene lógica en el juego)



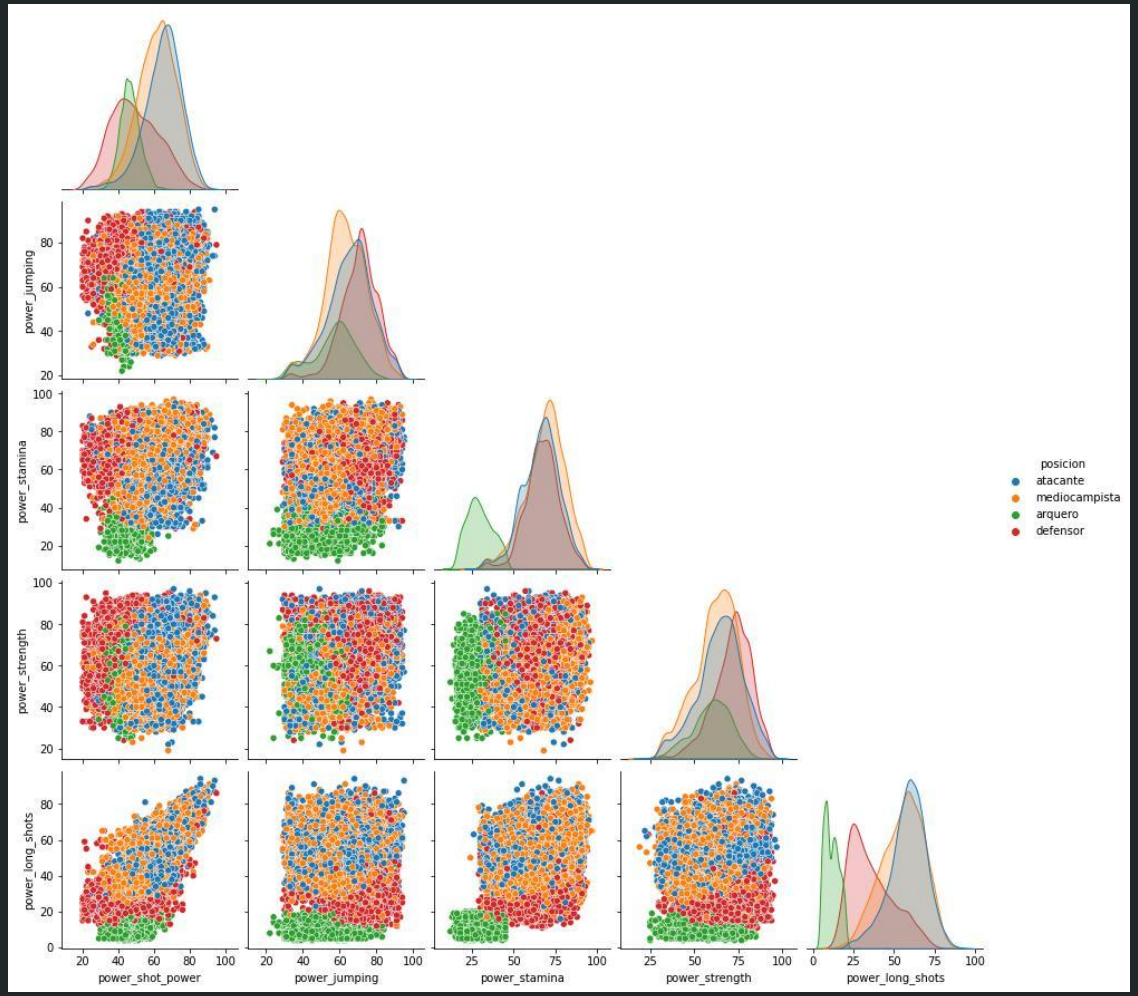
Habilidades de movimiento

- En reacción las distribuciones son muy parecidas entre todas las clases
- En las demás variables, no se observan diferencias notables entre las clases, a excepción de los arqueros
- No parece haber correlación entre estas variables, a excepción de la velocidad de pique y la aceleración, que parecen estar linealmente relacionadas



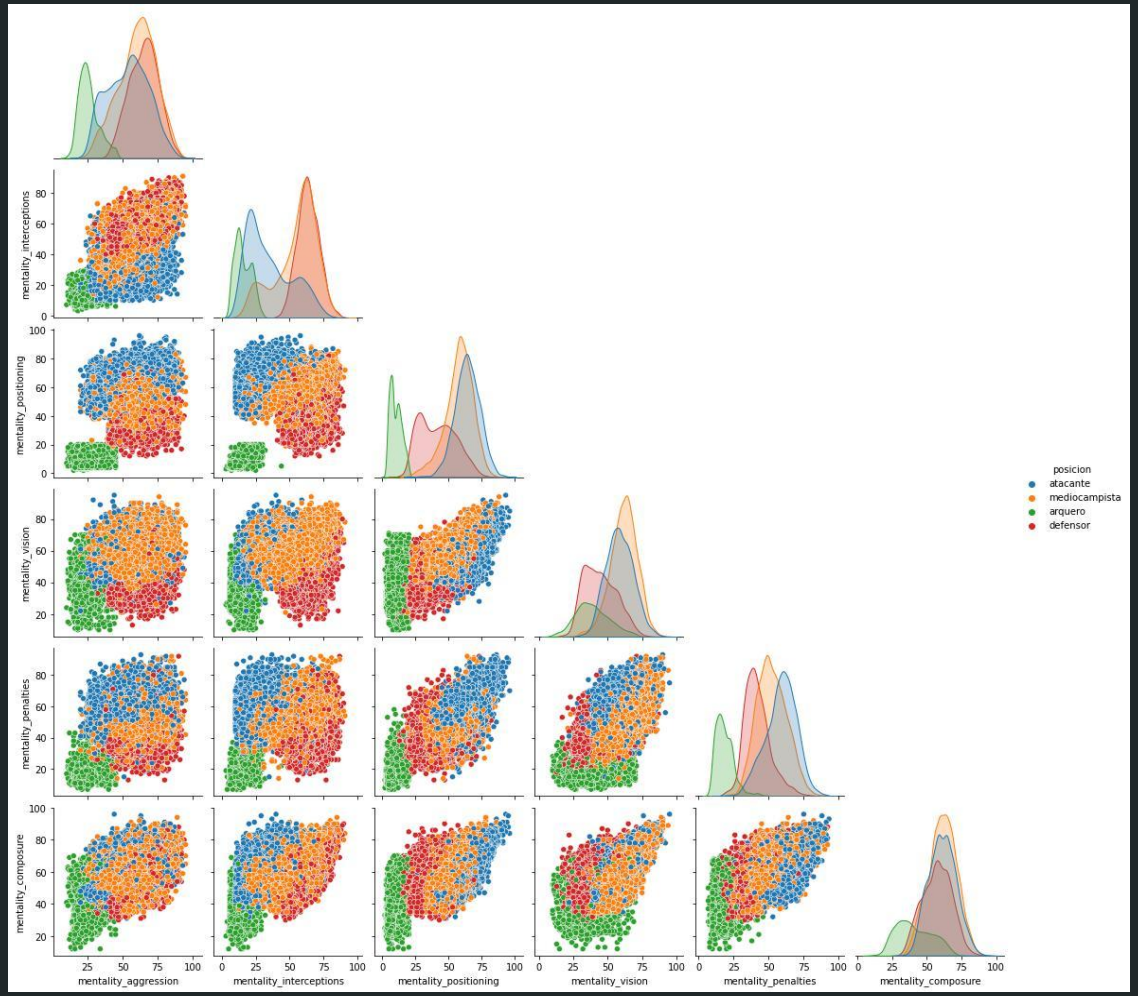
Habilidades de potencia

- Las variables son prácticamente indistinguibles entre atacantes y mediocampistas
- Para ‘power_jumping’ no hay distinción casi entre las clases
- Defensores y arqueros se acercan más en estas variables
- La resistencia (stamina) es muy baja en los arqueros.
- No hay correlaciones apreciables entre estas variables



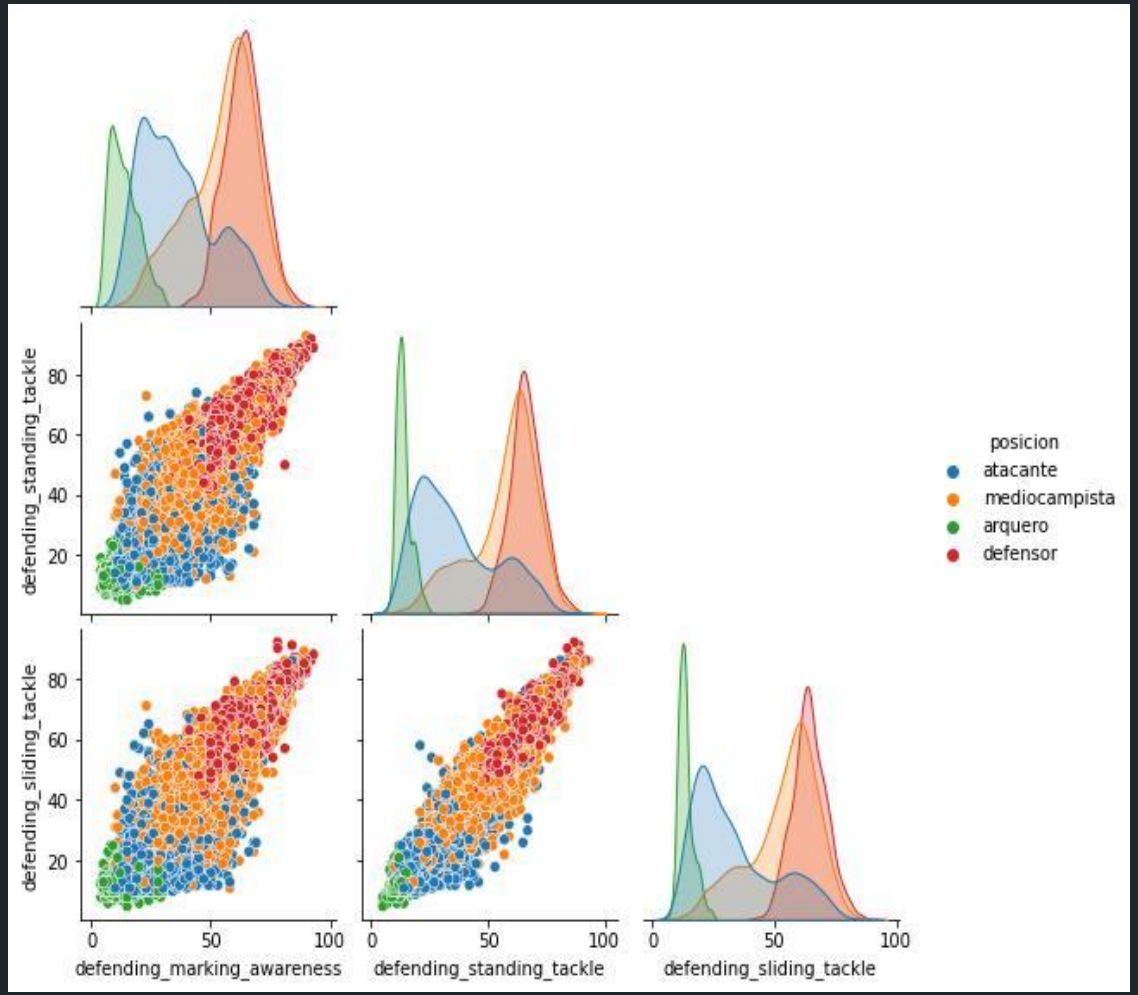
Habilidades mentales

- Intercepción muy baja en arqueros y atacantes y idénticas entre defensores y mediocampistas (con mas dispersión en estos ultimos)
- Todas estas habilidades son muy bajas en arqueros.
- Separaciones entre las otras clases no tan notorias
- No se observan correlaciones apreciables entre variables



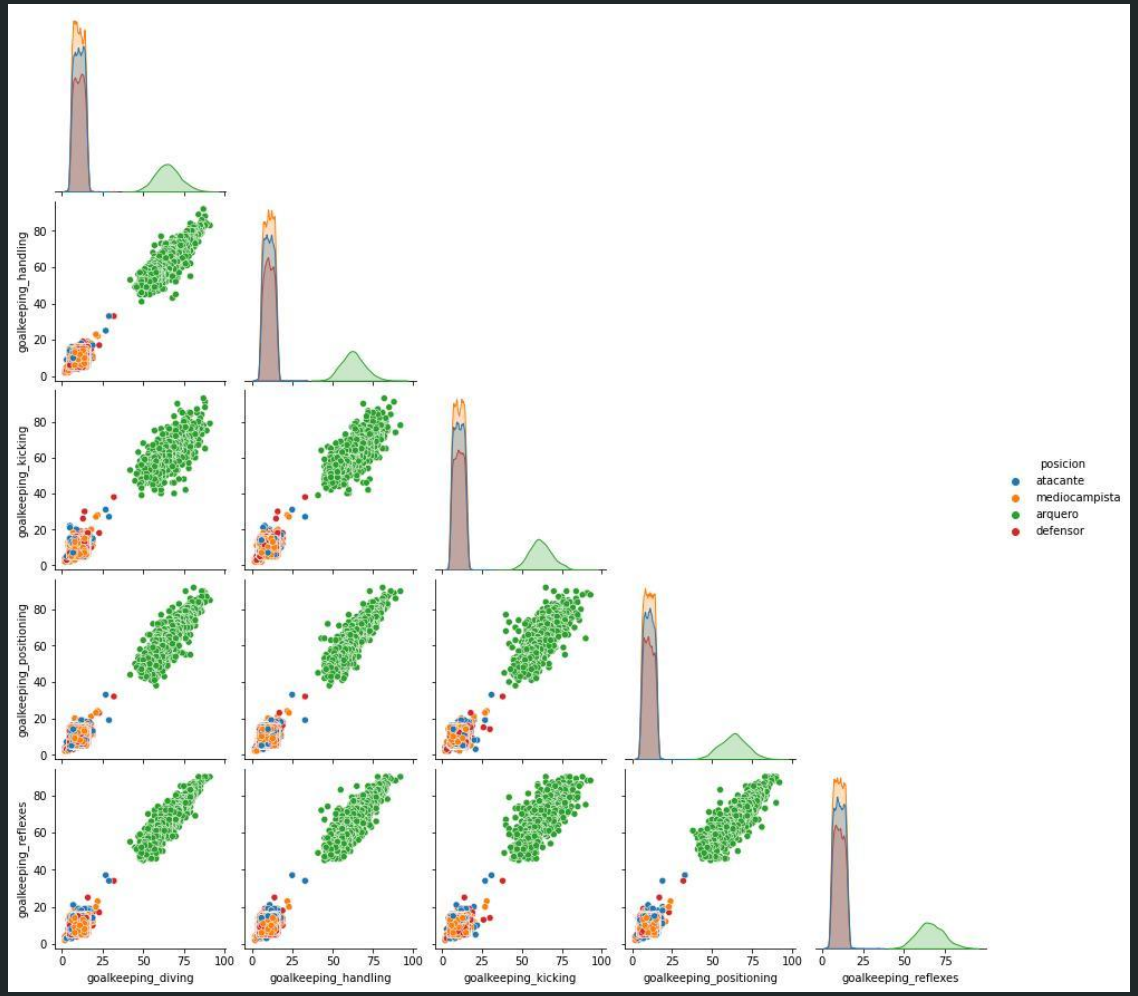
Habilidades de defensa

- Muy bajas en arqueros, distribuciones bimodales en atacantes con valores cercanos a los arqueros y valores elevados y similares para mediocampistas y defensores (con mayor varianza en mediocampistas)
- Correlación lineal entre las tres variables



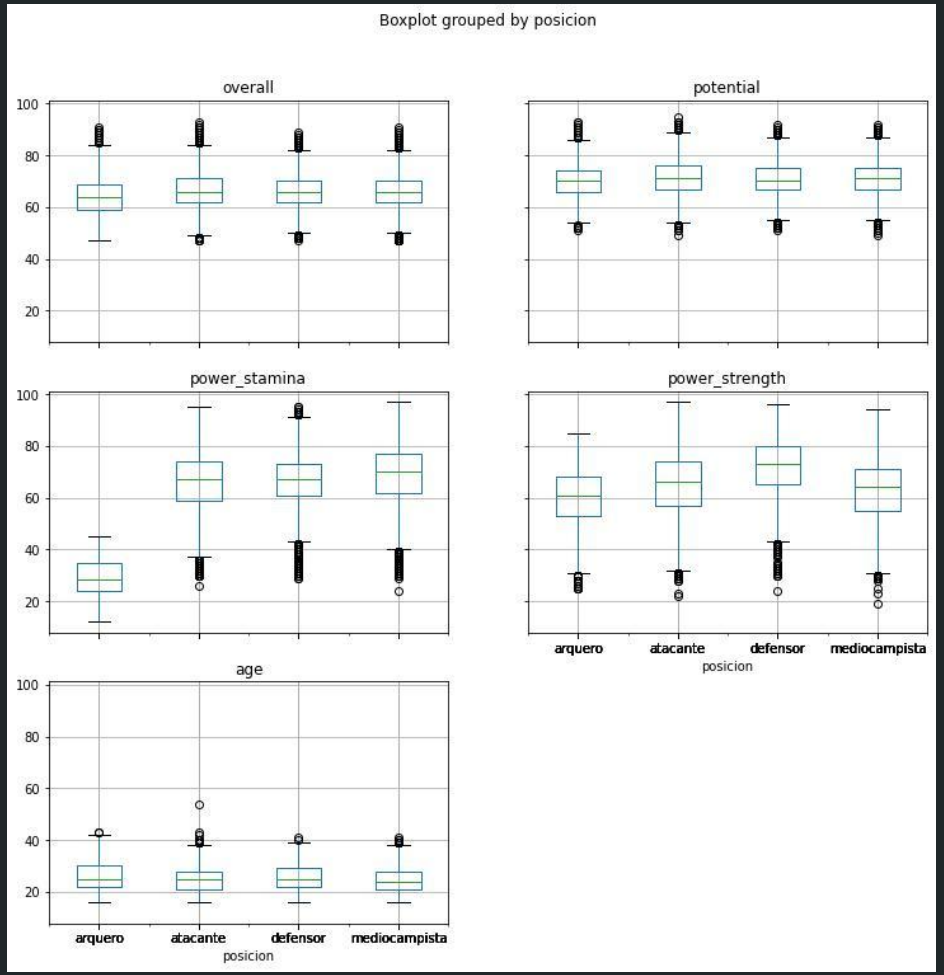
Habilidades de arquero

- Valores elevados y con varianza apreciable para arqueros, se observa correlación lineal entre las variables
- Distribuciones y valores idénticos para las tres clases restantes, no hay separabilidad

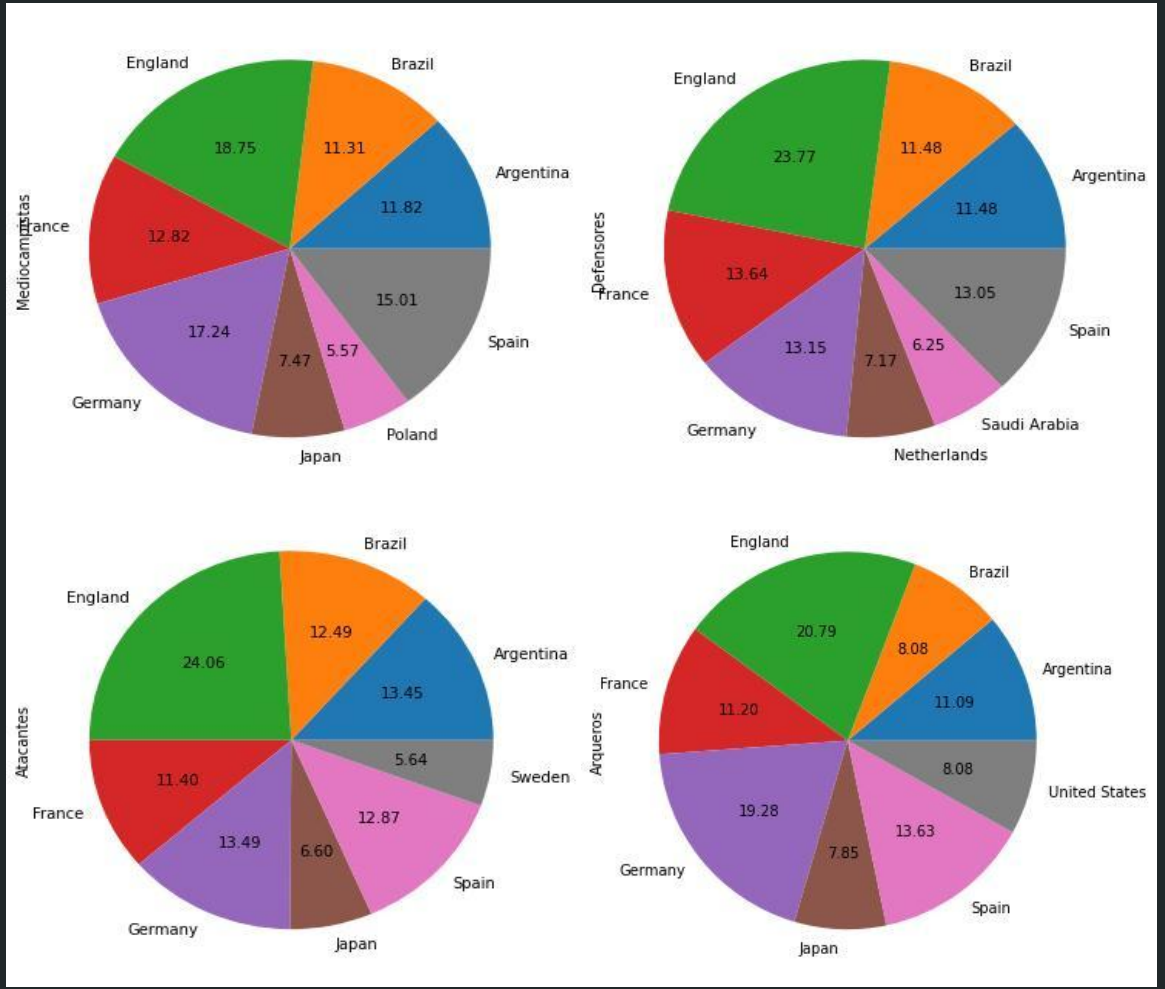


Boxplots en variables seleccionadas

No se observan distribuciones apreciablemente distintas entre clases excepto para la variable ‘power_stamina’, que es muy baja en arqueros. Se observa un número importante de outliers en casi todas las clases, y en todas las variables.



Países más frecuentes por clase



Luego de la exploración de la base de datos, observamos que los arqueros tienen características muy particulares y son fácilmente distinguibles de las otras clases

Por lo tanto, tomamos la decisión de eliminarlos de la base de datos junto con los atributos propios de arqueros (goalkeeping abilities)

Entrenamiento de modelos de aprendizaje

Modelos elegidos

- AdaBoostClassifier()
- BaggingClassifier(bootstrap_features=True)
- ExtraTreesClassifier(criterion='entropy')
- GradientBoostingClassifier()
- RandomForestClassifier()
- LogisticRegressionCV(solver='sag',max_iter=3000)
- SGDClassifier()
- KNeighborsClassifier(n_neighbors=10)
- svm.SVC(probability=True),
- svm.NuSVC(probability=True),
- svm.LinearSVC(max_iter=50000)
- DecisionTreeClassifier(criterion='entropy'),
- ExtraTreeClassifier()
- LinearDiscriminantAnalysis()
- QuadraticDiscriminantAnalysis()
- XGBClassifier()

La base de datos se separa en conjuntos de entrenamiento y validación con una proporción de 0.8 a 0.2

Para cada clasificador se computó la accuracy del mismo junto con el tiempo de cómputo del proceso de entrenamiento y validación. Se observa que SVC es el que obtuvo una accuracy más elevada, pero XGB obtuvo una accuracy apenas 0.005% menor en un tiempo de cómputo alrededor de 12 veces menor. Elegimos por lo tanto a XGB como mejor clasificador de los entrenados.

	Accuracy	Time [s]
SVC	0.808591	51.112328
XGB	0.802747	4.429037
Random Forest	0.800117	3.499199
Logistic Regression CV	0.798364	171.478245
Linear SVC	0.798071	162.897435
Gradient Boosting	0.795733	19.672372
Linear Discriminant Analysis	0.791058	0.267970
Extra Trees Classifier	0.790473	1.920984
SGD	0.787843	0.615901
Bagging Classifier	0.786967	2.098328
Quadratic Discriminant Analysis	0.786967	0.270047
K-Neighbors	0.779661	1.908481
NuSVC	0.777908	111.039436
Decision Tree Classifier	0.711572	0.401830
AdaBoost	0.700760	2.218109
Extra Tree Classifier	0.665400	0.040329

Optimización de hiperparámetros del mejor clasificador

Hiperparametros elegidos

Queremos mejorar la accuracy de XGB, y para conseguir esto elegimos un conjunto de valores para los hiperparámetros de XGB, entrenando los datos con estos y observando cual arroja una mejor accuracy. Los parámetros que elegimos iterar son:

- Número de estimadores: (50, 150, 200, 250, 300)
- Profundidad máxima del árbol: (1,3,5,7,9)
- Tasa de aprendizaje: (0.0001, 0.001, 0.01, 0.1, 0.2)

Observemos que tenemos en total 125 modelos distintos, y utilizando validación cruzada K-fold con K=10, esto nos da un total de 1250 entrenamientos y validaciones. Utilizamos el algoritmo HalvingGridSearchCV (en vez de GridSearchCV o RandomizedSearchCV) como estimador, que es un algoritmo todavía experimental de Scikit-Learn.

¿Por qué elegimos HalvingGridSearchCV?

A diferencia de GridSearchCV que entrena toda la base de datos sobre cada una de las combinaciones de hiperparametros y elige la mejor combinacion de todas, HalvingGridSearch por default realiza lo siguiente:

1. Entrena primero todas las combinaciones sobre una fracción reducida del dataset.
2. Se queda con la tercera parte de los candidatos, eligiendo los que mejor se desempeñaron.
3. Vuelve a entrenar estos sobre una fracción mayor del dataset
4. Vuelve a desechar dos tercios de los candidatos

El proceso sigue hasta quedar con solo un candidato

¿Por qué elegimos HalvingGridSearchCV?

Si bien GridSearchCV consigue una accuracy un poco más alta que HalvingGridSearchCV, lo hace a un costo en tiempo de cómputo que aumenta exponencialmente. HalvingGrid llega a ser hasta 23 veces más rápido que GridSearch sin un costo significativo en la precisión del modelo.

# of Components	Execution Time (seconds)		Cross-Validation AUC-ROC Score		Test AUC-ROC Score	
	GS	HGS	GS	HGS	GS	HGS
1	5	6	0.9832	0.9450	0.9846	0.9450
32	87	31	0.9849	0.9445	0.9524	0.9450
243	764	85	0.9852	0.9445	0.9861	0.9470
1024	3480	164	0.9861	0.9438	0.9779	0.9523
3125	10856	465	0.9864	0.9545	0.9816	0.9467

Fuente: ‘20x times faster Grid Search Cross-Validation’, Satyam Kumar. Publicado en TowardsDataScience

Finalmente, obtenemos que el mejor modelo es XGB con:

- Número de estimadores: 250
- Profundidad máxima: 7
- Tasa de aprendizaje: 0.2

```
{'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 250}
```

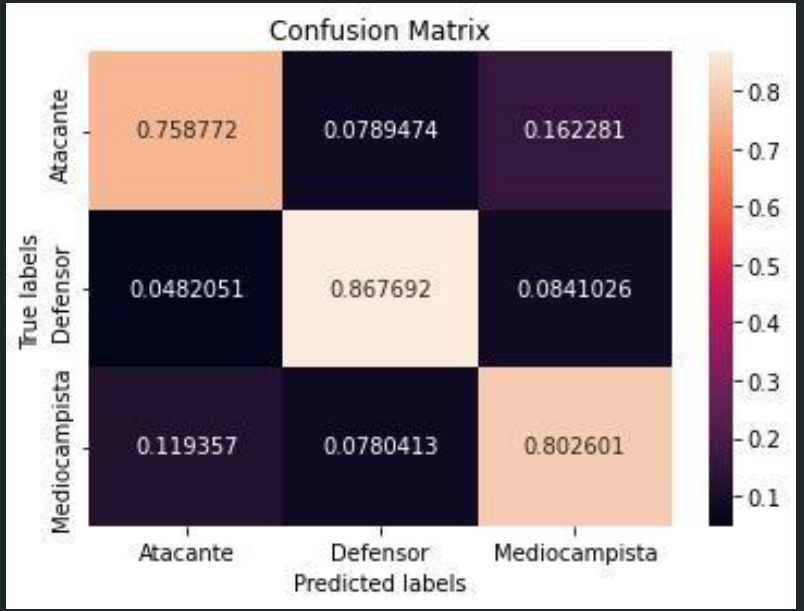
La accuracy conseguida entrenando el modelo optimizado para todo el dataset es 0.80655, lo cual no es apreciablemente mayor a la conseguida con los parámetros de default

Dado de que HalvingGridSearchCV se desempeña muy bien en la selección de modelos, y que se han elegido una cantidad importante de hiperparámetros a variar, no podemos decir preliminarmente que este aumento tan insignificante en el desempeño se deba a un error en la programación

Matriz de confusión y curvas ROC

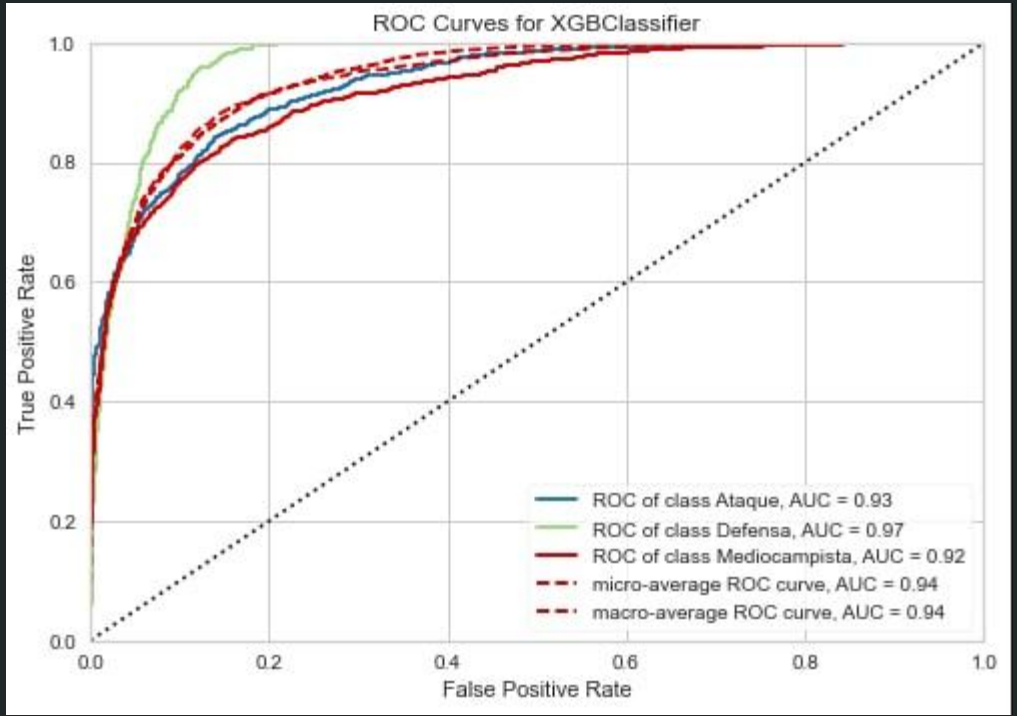
Vemos que la clase mejor clasificada (con menos de 1% de falsos negativos y falsos positivos) es la de defensores, lo cual es consistente con el análisis exploratorio realizado.

Por otro lado, alrededor del 12% de mediocampistas fueron clasificados como atacantes, y 16% de atacantes fueron clasificados como mediocampistas. La precisión para estas clases es significativamente más baja que para defensores



Se observa en las curvas ROC lo mismo que se observaba en la matriz de confusión: los defensores son los que poseen mejor desempeño AUC con un índice de 0.97.

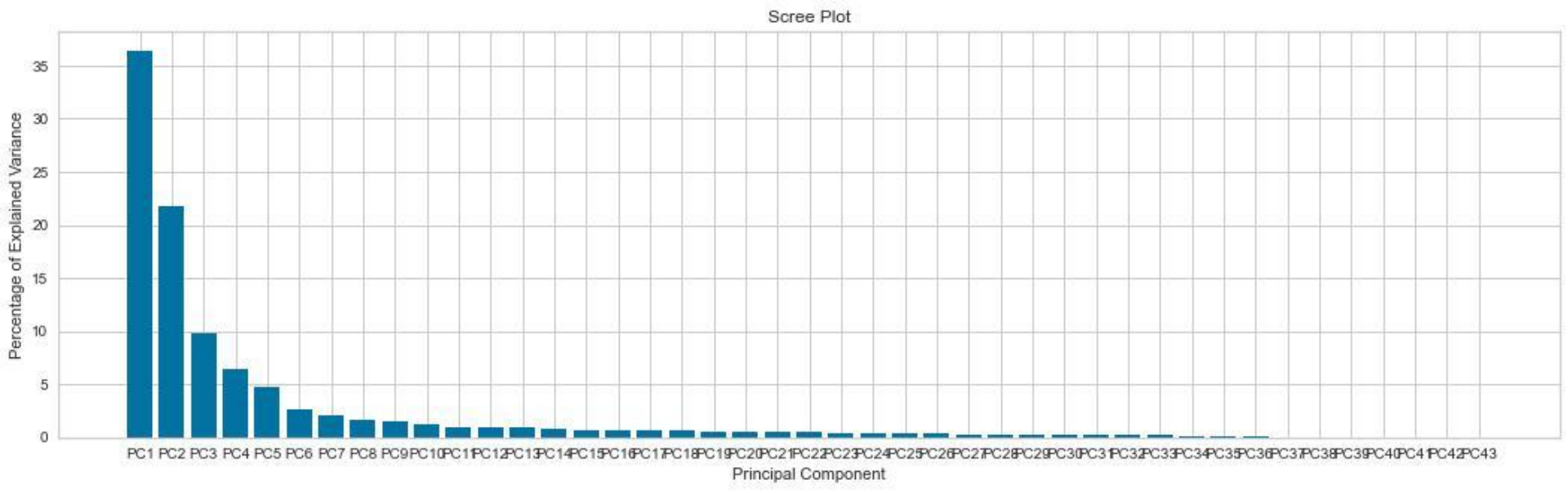
Mediocampistas y atacantes poseen desempeños AUC similares, aunque para la misma cantidad de falsos positivos, los atacantes tienen una mayor proporción de positivos correctamente clasificados.



Podemos decir que es la similaridad en los atributos entre mediocampistas y atacantes lo que hace que el desempeño del modelo no sea tan elevado como desearíamos, ya que la clasificación de defensores tiene un desempeño de casi el 90%

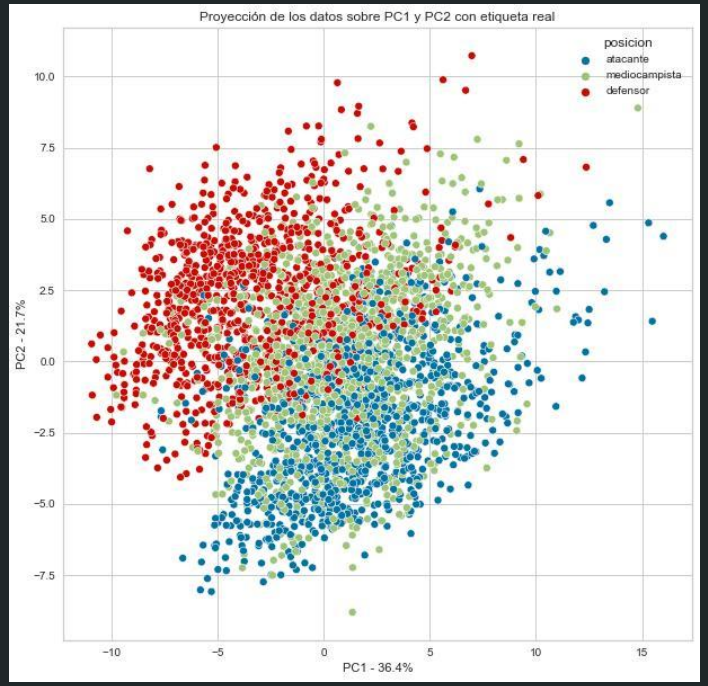
Análisis de componentes principales (PCA)

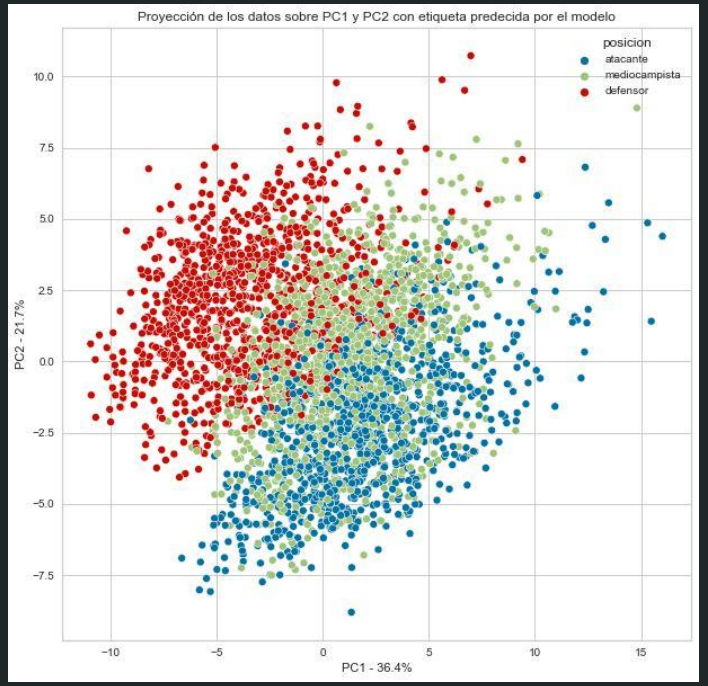
Realizamos la descomposición en componentes principales sobre los datos de prueba de la base de datos. Observamos el porcentaje de varianza explicada de cada componente.



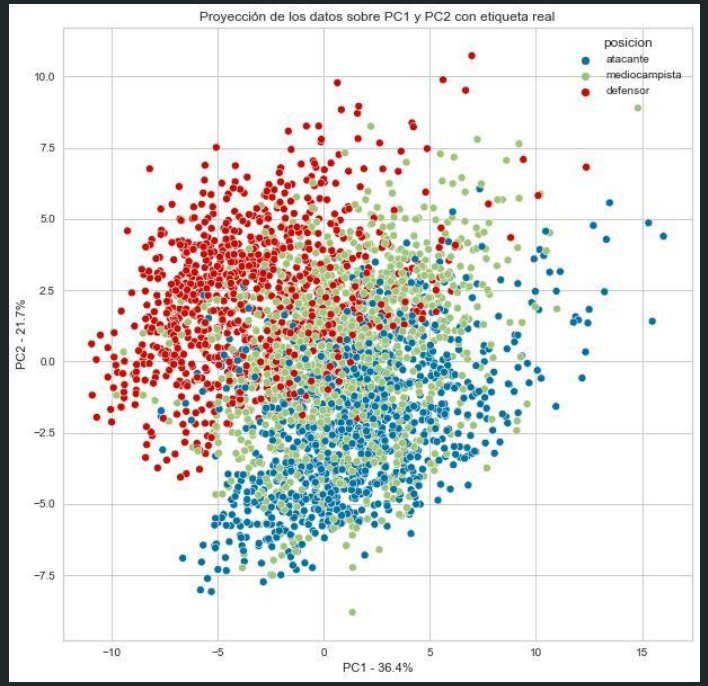
Observemos que las primeras dos componentes acumulan aproximadamente el 55% de la varianza explicada

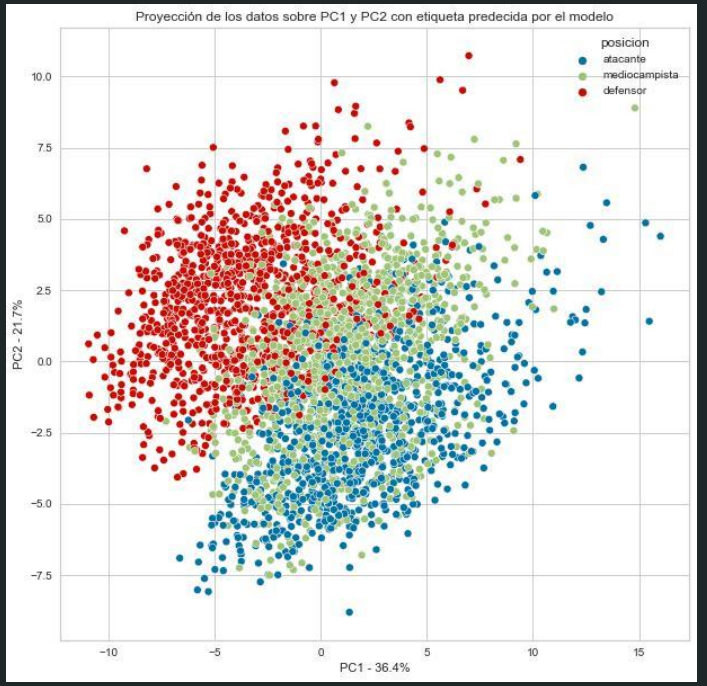
Proyectamos los datos de test sobre las primeras dos componentes, coloreando por etiqueta real y la etiqueta predecida por el modelo



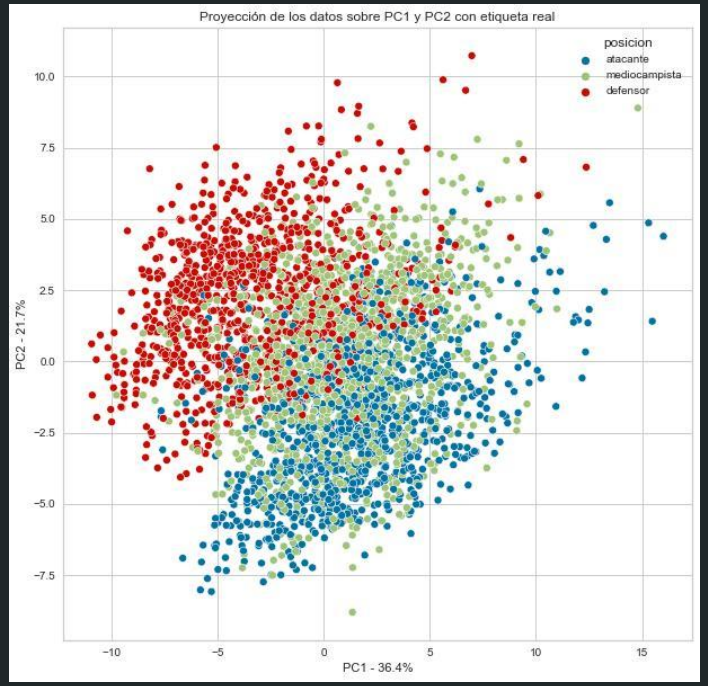


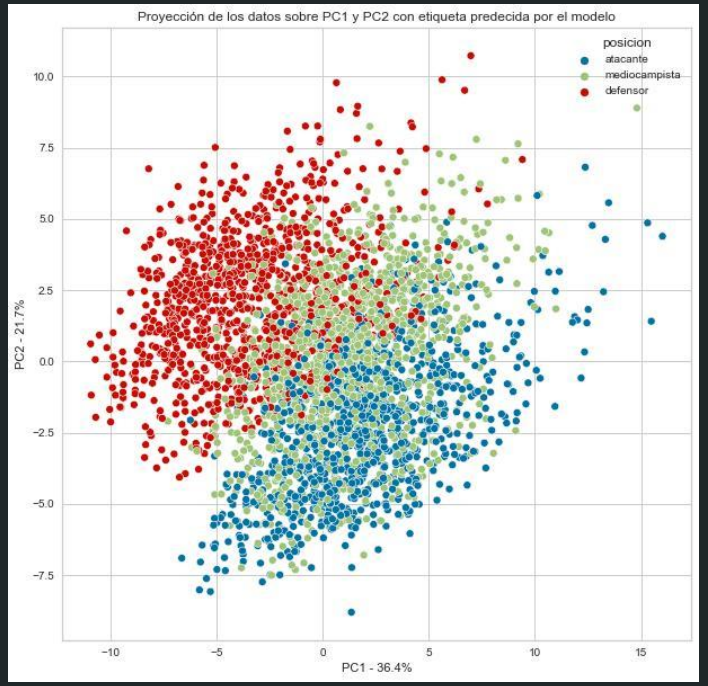
Observamos que los datos se separan en clusters, especialmente los defensores de los mediocampistas y atacantes, que están más ‘mezclados’. Observemos también que el modelo pudo reproducir la tendencia general de los datos reales





También podemos ver (intuitivamente) que el motivo por el que los mediocampistas tenían mayor proporción de falsos positivos es porque se encuentran entre los atacantes y los defensores: hay más probabilidades de que un punto clasificado como mediocampista sea en realidad de cualquiera de las otras dos clases. En cambio, si una muestra es clasificada como atacante, es muy raro que en realidad haya sido defensor, y viceversa.





Fin