1. Introducción:

Este trabajo tiene como objetivo principal crear un modelo de Machine Learning que sea capaz de ayudar en labores de Scouting dentro del fútbol a los equipos, consiguiendo encontrar jugadores que tengan un valor menor al que les correspondería por las estadísticas obtenidas de la pasada temporada.

Para ello, se han utilizado datasets descargados de la página fbref.com correspondientes a estadísticas de jugadores en la temporada 2018-2019 de 7 ligas diferentes:

- La Liga (España)
- Premier League (Inglaterra)
- Serie A (Italia)
- Bundesliga (Alemania)
- Ligue 1 (Francia)
- Liga NOS (Portugal)
- Eredivisie (Países Bajos)

Los valores de mercado de los jugadores de las ligas anteriormente mencionadas, se han extraído de Kaggle del dataset EuropeanRosters.csv, el cual se corresponde con una acción de webscrapping a la página web de Transfermarkt.

2. Datos utilizados:

Para la creación del modelo, se realiza una exhaustiva limpieza de datos. Se logran extraer 2012 datos referentes a jugadores de las 7 ligas mencionadas, de los cuales se destacarán:

Variables categóricas:

1. Equipo

Como necesitamos una variable numérica para la realización de modelos de Machine Learning, se han enumerado los 134 equipos a modo de ranking según el valor de mercado de sus jugadores (el que más valor total sume, recibirá el valor "1").

2. Liga

Al igual que los equipos, se han enumerado las 7 ligas a modo de ranking según el valor de mercado de sus jugadores (la que más valor total sume, recibirá el valor "1").

Liga	Valor de Mercado	Ranking
Premier League	9323925000	1
LaLiga	6179850000	2
Serie A	5310925000	3
Bundesliga	4712900000	4
Ligue 1	3588700000	5
Liga NOS	1134100000	6
Eredivisie	1016550000	7

3. Posición

Se han dividido en 4 categorías referenciando cada número lo alejada que está la posición del jugador de su propia portería, estableciendo:

1 → Portero
2 → Defensas y laterales
3 → Mediocentros e Interiores
4 → Atacantes

Variables numéricas generales:

1. Edad
2. Minutos jugados
3. Goles anotados
4. Asistencias anotadas
5. Goles anotados esperados

Variables numéricas específicas:

1. Goles encajados (Porteros)
2. Paradas (Porteros)
3. Porterías a cero (Porteros)
4. Goles encajados mientras juegan (Defensas y Laterales)
5. Goles encajados esperados (Defensas y Laterales)

3. Target utilizado

Se establecen un total de 4 rangos de valor, los cuales se obtienen de los cuartiles de los datos obtenidos previamente.

```
min      5.000000e+04
25%      1.750000e+06
50%      5.000000e+06
75%      1.500000e+07
max      2.000000e+08
Name: Valor de Mercado,
```

- El primer rango corresponderá a los jugadores que tienen valores de mercado entre 0 y el primer cuartil (C1) de los datos, es decir, 1.750.000€
- El segundo rango corresponderá a los jugadores que tienen valores de mercado entre C1 y C2.
- El tercer rango corresponderá a los jugadores que tienen valores de mercado entre C2 y C3
- El último rango corresponderá a los jugadores que tienen valores de mercado superiores a C3, es decir, 15 millones de euros.

4. Entrenamiento de modelos de Machine Learning

Se comienza el entrenamiento de modelos con una premisa muy clara:
“El modelo que será elegido se corresponderá con el que menos se aleje de la realidad”.

En términos coloquiales, como el modelo se basa en rangos de valor de mercado, es completamente normal que el modelo tenga dudas entre los valores “1” y “2”, “2” y “3”, “3” y “4”. Lo que nos otorgará información relevante serán los errores de predicción en los que:

- Si el valor del jugador es 1 (entre 0 y 1.750.000€), el valor predicho por el modelo sea 3 o 4 (más de 5.000.000€)
- Si el valor del jugador es 2 (entre 1.750.000€ y 5.000.000€), el valor predicho por el modelo sea 4 (más de 15.000.000€)

Por ello, no se tendrá en cuenta únicamente el score predictivo, sino que se pondrá el foco de atención en la matriz de confusión, verificando que no se produzca una gran desviación entre los valores del target y los predichos por el modelo.

Modelos:

1. *Árbol clasificador*

A pesar de cambiar la profundidad del árbol o el número de columnas a utilizar, los valores de score no terminan de superar el 0.63.

Decission Tree Classifier

Random State	Max_depth	max_features	min_samples_leaf	SCORE TRAIN	SCORE TEST
42	5	12		0.66793	0.56858
42	6	12	5	0.71875	0.589403
42	7	14	9	0.73151	0.59553
42	7	9	5	0.7371	0.62034
42	7	12		0.773	0.592

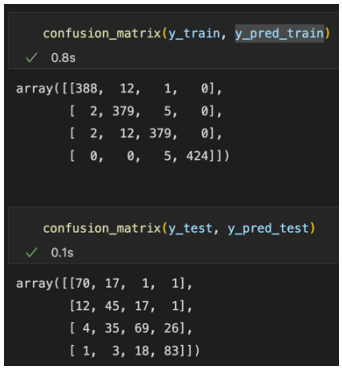
2. *Regresión Logística*

Cambiando parámetros, no se llega a lograr un score por encima de 0.55.

Logistic Regression

Penalty	C	SCORE TRAIN	SCORE TEST
L2	0.2	0.58349	0.52853
L2	0.1	0.59045	0.53349
L2	0.01	0.59045	0.53101
L2	0.001	0.58548	0.52605
L2	0.0001	0.57654	0.54094

3. *Random Forest*



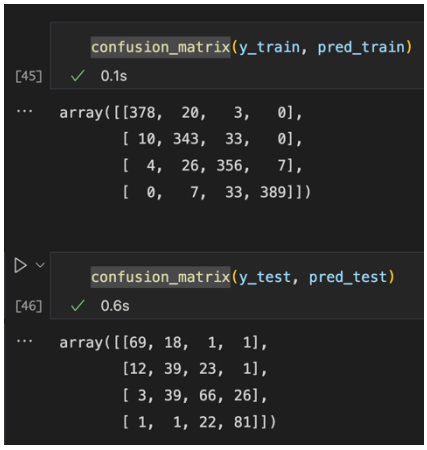
Se logran valores que llegan hasta un 0.66 en score, los cuales empiezan a ser notables debido a la poca dispersión que presenta la matriz de confusión.

Sin embargo, esta matriz denota un pequeño grado de Overfitting, debido a que el score entrenado es de 0.97576

Random Forest

Random State	n_estimators	Max_depth	Max features	SCORE TRAIN	SCORE TEST
42	75	5	10	0.75264	0.63027
42	80	7	10	0.83277	0.62779
42	75	7	12	0.83468	0.64268
42	82	10	12	0.94841	0.65260
42	80	9	12	0.92044	0.63523
42	80	11	12	0.97576	0.66253

4. *Grid Search CV → Logistic regression vs Random Forest*



Este modelo nos deja claro que, antes que una Regresión Logística, para predecir nuestros valores es mejor realizar un Random Forest.

Obtenemos valores de score que rondan el 0.65 y que presentan una matriz de confusión con muy pocos errores relevantes.

Grid Search CV --> Random Forest vs Logistic Regression

LR - Penalty	LR - C	RF - n_estimators	RF - max_features	RF - max_depth	CV	n_jobs	SCORE TRAIN	SCORE TEST
		96	9	9	20	-1	0.90926	0.65260
		78	9	7	20	-1	0.83778	0.64267
		120	9	7	20	-1	0.83903	0.63523
		120	10	7	15	-1	0.84214	0.62779
		75	9	7	20	-1	0.83343	0.64764

5. *Grid Search CV → SVC*

Vamos a probar un Grid Search Cross Validation para encontrar los parámetros óptimos en un vector soporte clasificador.

No logramos scores por encima del 0.60, por lo que no será el algoritmo óptimo.

Grid Search CV --> SVC

kernel	C	gamma	coef0	CV	n_jobs	SCORE TRAIN	SCORE TEST
rbf	720	scale	-100	10	-1	0.65872	0.59046
rbf	100	scale	-150	10	-1	0.62495	0.55865
rbf	500	scale	-150	15	-1	0.65479	0.58449
rbf	500	scale	-1000	20	-1	0.65342	0.58449

6. *KNN*

Este modelo entrenado nos otorga un score máximo de 0.74.
Como explicamos anteriormente, nuestro principal interés dentro de la predicción es que la dispersión de la matriz de confusión sea mínima.

```
confusion_matrix(y,pred)
✓ 0.5s

array([[435,  38,  15,   2],
       [ 86, 310,  57,   8],
       [ 60,  97, 320,  50],
       [ 14,  57,  54, 409]])
```

En la imagen se puede observar que la dispersión es muy alta comparada con la del Random Forest o el SVC, por lo que descartamos el KNN como candidato a ser el algoritmo óptimo.

KNN Classifier

n_neighbors	leaf_size	p	n_jobs	SCORE
3	11	1	-1	0.74155
3	15	2	-1	0.73260
3	30	1	-1	0.74105
3	30	2	-1	0.73260
5	30	1	-1	0.69085
5	30	2	-1	0.67693
3	5	1	-1	0.74105

7. *XG Boost Clasificador*

Con este algoritmo alcanzamos el score más alto en la predicción: 0.662.
Como este score tampoco tiene nada que envidiar a los “0.65” por los que se mueven los Random Forest, verificamos visualizando la matriz de confusión.

```
confusion_matrix(y_train, pred_train)
✓ 0.4s

array([[374,  25,   2,   0],
       [ 25, 321,  39,   1],
       [  8,  32, 331,  22],
       [  0,   4,  29, 396]])
```

```
confusion_matrix(y_val, pred)
✓ 0.4s

array([[71, 14,  4,  0],
       [14, 41, 20,  0],
       [ 5, 30, 71, 28],
       [ 1,  2, 18, 84]])
```

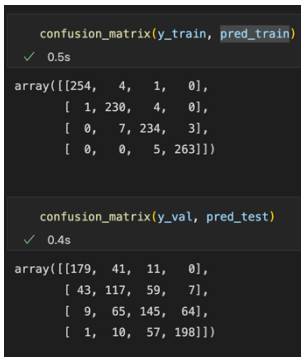
Podemos comprobar que tiene 4 errores relevantes

XG Boost Classifier

Random state	n_estimators	max_depth	early stopping rounds	n_jobs	SCORE TRAIN	SCORE VAL
42	80	6	5	-1	0.94460	0.64735
42	80	7	5	-1	0.96590	0.62582
42	100	7	5	-1	0.94460	0.64735
42	85	7	5	-1	0.98210	0.65019
42	85	6	5	-1	0.96322	0.65012
42	80	5	5	-1	0.93339	0.65009
42	80	4	8	-1	0.88377	0.66253
42	100	7	3	-1	0.93252	0.63741

8. Gradient Boosting Classifier

Con este algoritmo el score más alto que alcanzamos en la predicción es de 0.635, lo cual no está mal, pero no supera al XG Boost mencionado anteriormente.



Por si acaso, comprobamos la matriz de confusión para ver cuantos errores relevantes tendríamos en este modelo. Vemos que tiene 18 errores relevantes, por lo que no será el algoritmo a utilizar.

Gradient Boosting Classifier

Random state	max_depth	n_estimators	learning rate	SCORE TRAIN	SCORE VAL
42	7	6	1.0	0.99734	0.58449
42	5	10	0.5	0.93836	0.61630
42	4	7	0.5	0.84691	0.61232
42	4	7	1.0	0.88667	0.60835
42	5	4	1.0	0.88601	0.61431
42	5	4	0.5	0.85345	0.61033
42	5	5	0.7	0.90125	0.63021
42	5	5	0.71	0.89860	0.62624
42	4	9	1.0	0.94731	0.63021
42	5	6	0.95	0.97515	0.63518

5. Modelo de Machine Learning

Nos decantaremos por utilizar XG Boost Classifier para predecir el rango de valor de mercado de los jugadores, por ciertos detalles que lo diferencian del Random Classifier.

- Aprendiendo menos del train, es capaz de generalizar más, consiguiendo el mismo score que con un random forest cuyo accuracy en train es de 0.97.
- Además, engloba mejor los valores, no otorgando a los valores entre 0 – 1.750.000, valores superiores a 15 millones. Esto es algo que pasa en el Random Forest con únicamente un caso.
- Acierta más valores altos, lo que quiere decir que los comprende algo mejor y que va a sobrevalorar menos las predicciones.

	Importancia	Columna
1	0.224122	Equipo
3	0.117040	Minutos jugados
2	0.099675	Edad
12	0.091991	Liga
4	0.081996	Goles anotados
6	0.081865	Goles anotados esperados
8	0.054263	Paradas (Porteros)
10	0.051108	Goles encajados jugando
5	0.050116	Asistencias anotadas
11	0.048761	Goles encajados esperados
0	0.041602	Posición
7	0.031440	Goles encajados (Porteros)
9	0.026021	Porterías a cero (Porteros)

Es importante destacar la importancia de las columnas en el algoritmo, siendo las principales el equipo en el que juega, los minutos jugados y la edad

6. Conclusión

Hoy en día es muy complicado predecir el valor de mercado que debería tener un jugador, y muchísimo más difícil es predecir su proyección.

Con este proyecto mi intención es crear una ayuda al Scouting en fútbol, tratando de poder fijar la atención en unos pocos jugadores que el algoritmo considera “más valiosos” de lo que el mercado dicta.