

## ✓ Trabajo Práctico Final - Segunda Entrega

### Grupo 11

Integrantes:

- Luca Caunedo:
- Agustin Sumay:
- Lucas Gervasi: [gervasilucas22@gmail.com](mailto:gervasilucas22@gmail.com)

### Predicción de goles mediante eventos de remates en el fútbol

Enlace al conjunto de datos original:

2) El objetivo de este trabajo es desarrollar un modelo de clasificación, utilizando regresión logística para predecir la probabilidad de gol a partir de remates registrados en partidos de fútbol. Utilizando un conjunto de datos de eventos de remates, se analiza información relevante como la distancia al arco, el ángulo del remate, el tipo de remate y la posición del arquero. En esta primera entrega, logramos limpiar y explorar el dataset que brinda statsbomb, identificando variables claves y generando visualizaciones preliminares que servirán de base para el desarrollo del modelo predictivo en las siguientes etapas.

[+ Código](#)
[+ Texto](#)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
tiros = pd.read_csv('tiros.csv', on_bad_lines='skip')
```

on\_bad\_lines='skip' se agrega porque ante el error 'Error tokenizing data. C error: EOF inside string starting at row 2895' la linea está encerrada entre comillas dobles pero dentro tiene comillas simples ('), lo cual es inusual pero válido en Python, no en CSV estándar.

3)

```
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score, f1_score, roc_auc_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Variables
X = tiros[['distance', 'angle']]
y = tiros['goal']

# División train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Escalado
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```


4) Se aplicó un escalado estándar (StandardScaler) a las dos variables predictoras (distance y angle) para normalizar sus valores con media 0 y desviación estándar 1.

```
modelo_lineal = LinearRegression()
modelo_lineal.fit(X_train_scaled, y_train)
y_pred_lineal = modelo_lineal.predict(X_test_scaled)

modelo_log = LogisticRegression(max_iter=1000)
modelo_log.fit(X_train_scaled, y_train)

y_proba_log = modelo_log.predict_proba(X_test_scaled)[:, 1]
threshold = 0.3
y_pred_custom = (y_proba_log > threshold).astype(int)

print("F1 Score con threshold 0.3:", f1_score(y_test, y_pred_custom))
print("ROC AUC:", roc_auc_score(y_test, y_proba_log))
```

 F1 Score con threshold 0.3: 0.22437137330754353  
ROC AUC: 0.7528410358387986

5) Se utilizó un umbral de decisión personalizado (threshold = 0.3) en lugar del clásico 0.5, con el objetivo de mejorar la sensibilidad del modelo y detectar más goles, aun a costa de aumentar los falsos positivos.

F1 Score  $\approx$  0.22: Este valor indica que el modelo logra un balance moderado entre precisión y recall. Si bien no es un desempeño alto, para un modelo inicial basado únicamente en dos atributos (distancia y ángulo) resulta aceptable como benchmark. Este resultado sugiere que, aunque las variables seleccionadas son relevantes, existe espacio para mejorar el rendimiento.

ROC AUC  $\approx$  0.75: El área bajo la curva ROC muestra que el modelo discrimina significativamente mejor que una clasificación aleatoria (valor esperado de 0.5). Un AUC de 0.75 representa un buen punto de partida para un modelo simple y confirma que los atributos elegidos aportan información útil para diferenciar entre goles y no goles.

## Nuevo Modelo

```
shot_aerial_won_encoded = pd.get_dummies(tiros['shot_aerial_won'], prefix='shot_aerial')
ingoal_encoded = pd.get_dummies(tiros['goalkeeper_ingoal'], prefix='ingoal')
shot_body_part_encoded = pd.get_dummies(tiros['shot_body_part'], prefix='bodypart')

tiros = pd.concat([tiros, ingoal_encoded], axis=1)
tiros = pd.concat([tiros, shot_aerial_won_encoded], axis=1)
tiros = pd.concat([tiros, shot_body_part_encoded], axis=1)

# Seleccionar las características (X) y la variable objetivo (y)
X = tiros[['distance', 'angle'] + list(ingoal_encoded) + list(shot_aerial_won_encoded) + list(shot_body_part_encoded)]
y = tiros['goal']

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Escalado
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

modelo_lineal = LinearRegression()
modelo_lineal.fit(X_train_scaled, y_train)
y_pred_lineal = modelo_lineal.predict(X_test_scaled)

modelo_log = LogisticRegression(max_iter=1000)
modelo_log.fit(X_train_scaled, y_train)

y_proba_log = modelo_log.predict_proba(X_test_scaled)[: , 1]
threshold = 0.26
y_pred_custom = (y_proba_log > threshold).astype(int)

print("F1 Score con threshold 0.3:", f1_score(y_test, y_pred_custom))
print("ROC AUC:", roc_auc_score(y_test, y_proba_log))
```


 F1 Score con threshold 0.3: 0.4049159120310479  
ROC AUC: 0.7992743647106063

Tras ajustar el modelo con las 4 variables con un umbral de 0.3, el F1 Score mejoró a 0.404, mostrando un mejor balance entre precisión y recall. Además, el ROC AUC alcanzó 0.799, indicando una mayor capacidad para discriminar entre tiros que terminan en gol y los que no. Esto evidencia que el modelo no solo es más preciso sino también más robusto para clasificar correctamente.

```
from sklearn.model_selection import cross_val_score, cross_validate


# Usando roc_auc y f1 (con threshold 0.5 para predicción interna)
scores = cross_validate(modelo_log, X, y, cv=5,
                        scoring=['roc_auc', 'f1'],
                        return_train_score=False)

print("Validación cruzada ROC AUC:", scores['test_roc_auc'].mean())
print("Validación cruzada F1:", scores['test_f1'].mean())
```

 Validación cruzada ROC AUC: 0.7932446679622481  
Validación cruzada F1: 0.10690819779594969

```
import pandas as pd
coef_df = pd.DataFrame({
    'feature': X.columns,
    'coeficiente': modelo_log.coef_[0]
}).sort_values(by='coeficiente', key=abs, ascending=False)

print(coef_df)
```



	feature	coeficiente
0	distance	-1.641954
1	angle	0.253340
6	bodypart_Head	-0.247177
2	ingoal_False	0.172495
3	ingoal_True	-0.172495
9	bodypart_Right Foot	0.124998
5	shotaerial_True	-0.085035
4	shotaerial_False	0.085035
7	bodypart_Left Foot	0.056378
8	bodypart_Other	0.001210

- distance -1.64 A mayor distancia, mucho menos chance de gol. Impacto fuerte y negativo. Tiene mucho sentido.
- angle +0.25 Ángulo más amplio aumenta probabilidad de gol. Impacto positivo moderado.
- bodypart\_Head -0.25 Disparos de cabeza reducen la probabilidad de gol respecto a la categoría base (por ejemplo, pie).
- bodypart\_Right Foot +0.12 Disparos con pie derecho aumentan chances de gol.
- bodypart\_Left Foot +0.056 Disparos con pie izquierdo tienen un leve aumento en probabilidad.
- bodypart\_Other +0.001 Casi sin impacto.
- ingoal\_False +0.17 Cuando el arquero no está en la línea, aumenta la probabilidad (comparado con la base "True").
- ingoal\_True -0.17 Lo opuesto del anterior, disminuye la probabilidad.
- shotaerial\_True -0.085 Cuando el disparo es aéreo, baja un poco la probabilidad.
- shotaerial\_False +0.085 Lo opuesto, disparos no aéreos suben la probabilidad.

Haz doble clic (o pulsa Intro) para editar

```
!pip install mplsoccer
```