Análisis de modelo de Machine Learning para la clasificación de partidos ganados cuando Messi metió gol en el primer tiempo

Instituto Tecnológico y de Estudios Superiores de Monterrey

Miguel Chávez Silva A01661109

10 de septiembre de 2023

1. Introducción

El fútbol es uno de los deportes más apasionantes del mundo, además de que influye mucho la gran accesibilidad que tiene para poder jugar. Cada partido de fútbol es único, y múltiples factores pueden influir en el resultado final. Un jugador puede modificar completamente los factores, y más cuando se trata de un jugador estrella que puede ser un elemento crucial dentro de un equipo capaz de determinar el destino del equipo dentro del campo.

Lionel Andrés Messi Cuccittini es el mejor jugador en la historia [Meadows, 2022] ha sido una figura icónica en el mundo del fútbol durante más de una década. Tiene 7 balones de oro, actualmente está nominado para su octavo semestre y con su último acto más reconocido, que fue ganar el mundial, es casi seguro que sí lo gane. En este contexto, surge una pregunta: ¿qué impacto tiene la participación de Messi en cada partido, específicamente cuando anota un gol en el primer tiempo, en el resultado final de un partido?

Este trabajo se centra en poder resolver esa pregunta mediante la implementación de la Inteligencia Artificial; con modelos de Machine Learning (ML). Se explorarán y se analizarán datos recavados a lo largo de la carrera profesional para determinar si la mera presencia de un gol de Messi en el primer tiempo puede ser un indicativo significativo para predecir si su equipo ganará el partido.

2. Limpieza de Base de datos

Se descargó una base de datos de la plataforma Kaggle. La base de datos está registrada como uso público, por lo que cualquier persona puede descargarla y hacer algún tipo de análisis de esta base de datos. Primer se cambió el formato de fecha, inicialmente estaba como 05-01/05 y se hizo

una columna para cada dato de la fecha. Además se limpio la columna que contiene la forma en que metió el gol, se separó en si lo metió en algún tipo de jugada o si fue a balón parado (penal o tiro libre) y se agregó una nueva columna con 0 para cuando no se anotó el gol de esa forma y 1 cuando sí.

Después se crea la columna "victoria" que es nuestro *target*, es decir, es lo que se quiere predecir, se debe crear porque no se cuenta explícitamente con una columna pero sí hay una con el resultado final del partido en formato 2:00, donde el primer número es el número de goles anotados por el equipo en el que jugaba Messi y el segundo son los goles anotados por el equipo contrario. Primer se separaron los números y se compararon para crear una columna llamada 'victoria' que tiene *"TRUE"* para cuando ganó y *"FALSE"* para cuando empató o perdió.

Además había una columna en la que venían las posiciones que jugó en los diferentes partidos. Para la separación de esta variable, se hicieron columnas nuevas para cada posición que tenía como valor *"TRUE"* y *"FALSE"*, entonces si estaba jugando en una posición aparece *"TRUE"* y *"FALSE"* en todas las demás. Después, separamos el resultado del partido en el momento del gol, fue algo similar a la columna "victoria".

Por último, se convierten a variables *dummies* las variables que indican si estaba de local o de visita y la posición que estaba jugando. Y para finalizar, se eliminan columnas que no usaremos para el análisis debido a que no se consideran de gran relevancia para el análisis como: persona que le dio la asistencia, temporada, partido de la temporada, competencia, entre otros.

3. Aplicación del Modelo

Para la selección del modelo, se realizaron 5 modelos para poder determinar cuál era el mejor entre ellos, seleccionar uno en especial y a partir de ahí, modificar cosas para poder mejorarlo. Los modelos que se usaron fueron: Random Tree, Decision Tree, HB Gradient Boosting, Adaboost y Multi-layer Perceptron. Además, para la selección de los hiperparámetros se usó: Random Search y Grid Search.

	Random Search					Grid Search				
Columna1	Decision Tree	Random Forest	HB Gradient Boosting	AdaBoost	Multi-layer Perceptron	Decision Tree2	Random Forest2	HB Gradient Boosting2	AdaBoost2	Multi-layer Perceptron2
Puntuación Media	85%	84%	86%	84%	84%	78%	84%	82%	84%	84%
Accuracy	87%	91%	87%	90%	94%	78%	91%	88%	90%	91%
Precision	65%	95%	65%	45%	84%	63%	95%	62%	45%	77%

Figura 1: *Modelos con su Acurracy, Precision y Puntuación Media*

Como se puede observar en la figura 1, el modelo con mejores números es Random Forest, que incluso tiene los misma puntuación con Random Search y con Grid Search, es por eso que se eligió

ese modelo. Entre ambos, se eligió Random Search. Inicialmente, los parámetros a usar fueron:

```
param_dist = {
    'n_estimators': [50, 100, 150, 200],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2', None],
    'bootstrap': [True, False]
}
```

Se eligieron valores grandes para eventualmente ir recortando y finalmente encontrar los parámetros correctos; además, se añadieron y se quitaron parámetros para poder determinar si estaba *over, under o perfect fitting*. Para poder conocer eso, se graficó la curva de aprendizaje [Huet, 2023].

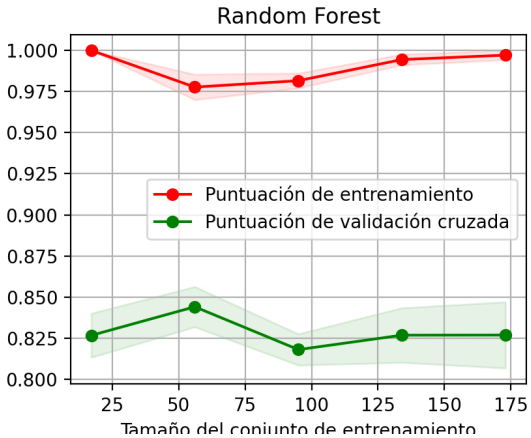


Figura 2: *Curva de Aprendizaje*

Como se puede observar en la figura 2, el hecho de que las líneas no convergen en ningún punto y además de que estén separadas, significa que hay *overffiting*, ya que el conjunto de entrenamiento se mantiene bien conforme aumentan los datos, y los datos de validación cruzada se mantienen pero nunca aumentan. Para ello, con base en los hiperparámetros anteriores, se imprimen para poder determinar alrededor de qué números se pueden encontrar los mejores. El primer resultado fue:

```
param_dist = {
    'n_estimators': 100,
    'min_samples_split': 2,
    'min_samples_leaf': 2,
```

```
    'max_features': 'log2',
    'max_depth': 50,
    'criterion': 'gini',
    'bootstrap': True
}
```

Con estos parámetros, el modelo logró obtener los resultados mencionados anteriormente, sin embargo, aún pueden mejorarse para evitar *overfitting*. Por ejemplo, para *'n_ estimators'*, el rango que existe entre 50 y 200, es muy grande, por lo que se puede definir un rango menor para poder encontrar la mejor combinación. Se hicieron varias repeticiones hasta llegar a esta lista de parámetros.

```
param_dist = {
    'n_estimators': 95,
    'min_samples_split': 8,
    'min_samples_leaf': 1,
    'max_features': 'log2',
    'max_depth': 50,
    'criterion':
    'entropy',
    'bootstrap': True
}
```

La gráfica para la curva de aprendizaje se ve de la siguiente manera:

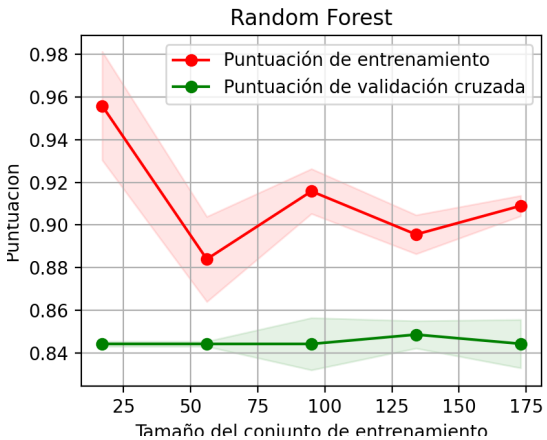


Figura 3: *Curva de Aprendizaje*

Como se puede observar en la fígura 3, las líneas empiezan a converger en un punto, por lo que significa que empieza a estar *perfect-fitting*. El punto es lograr encontrar que convergen en el

mismo punto y que además sigan estando en una posición elevada. Después de varias repeticiones, los mejores parámetros que se encontraron:

```
param_dist = {
    'n_estimators': 95,
    'min_samples_split': 6,
    'min_samples_leaf': 4,
    'max_features': 'log2',
    'max_depth': 51,
    'criterion': 'gini',
    'bootstrap': False
}
```

la gráfica se ve de la siguiente manera:

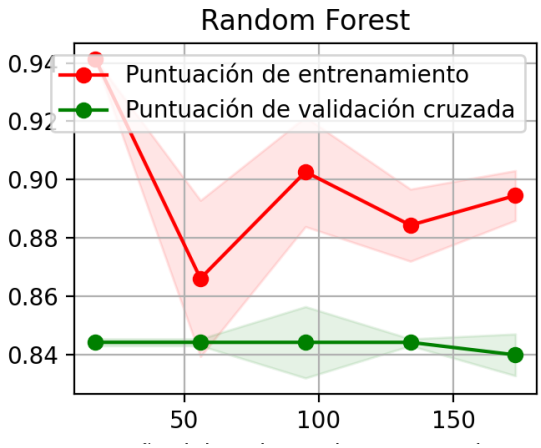


Figura 4: *Curva de Aprendizaje*

Se ve muy similar a la figura 3, sin embargo, se puede apreciar en la escala como está mucho más cerca, lo que indica que convergen más. Es importante mencionar que no se pierden los valores indicados en la tabla 1.

Al analizar varios parámetros, se concluyó que no se podía estabilizar como se deseaba, por lo que volvemos al mismo punto de partida en elección de modelos. Al observar las gráficas de cada uno de los modelos, la gráfica de de la curva de aprendizaje del modelo de *AdaBoost*, tenía un efecto muy parecido al deseado.

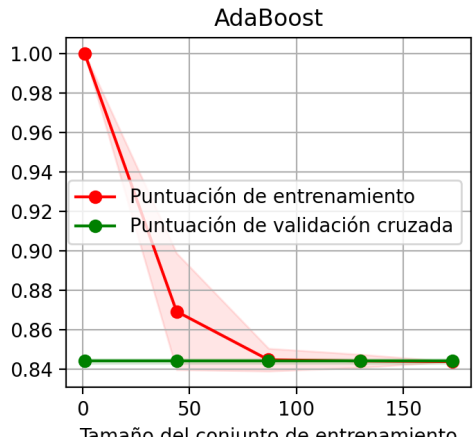


Figura 5: *Curva de Aprendizaje*

Como se puede observar en la figura 5, a medida que aumentan el tamaño de la base de datos, la puntuación de entrenamiento y la de validación cruzada, convergen en un mismo lugar, lo que significa que está *perfect fitted*, el único inconveniente es que se obtiene un *accuracy* de 0.89. Así se muestra la matriz de confusión.

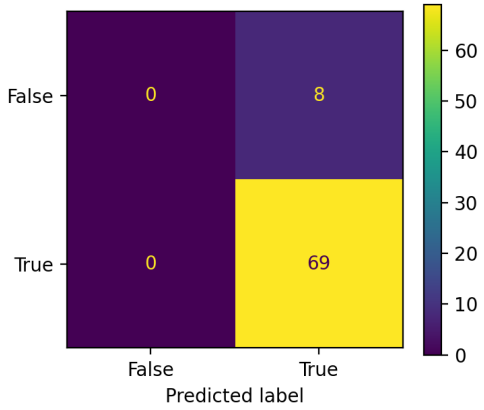


Figura 6: *Matriz de Confusión*

Como se puede observar en la figura 6, todas se clasificó como que ganó. Esto es un error debido al desbalance que hay entre las clases, es decir, se sabe de antemano que hay muchas más victorias que empates o derrotas, por lo que el modelo aprende que casi todo será victoria. El siguiente objetivo es poder balancear las clases para poder lograr que también se fije en la clase de no victoria.

Los hiperparámetros que se le asignó al modelo fueron los siguientes:

```
param_dist = {
'n_estimators': [50, 100, 150, 200],
```

```
    'learning_rate': [1],
    'algorithm': ['SAMME', 'SAMME.R']
}
```

pero como se mencionó anteriormente, el modelo no estaba bien entrenado, por lo que se decidió añadirle un hiperparámetro más que ayude a poder balancear más las clases y se le pueda indicar al modelo que haga más esfuerzo por identificar los patrones para la clasificación de cuando no gana. El hiperparámetro usado fue:

```
class_weights = {0: 0.8, 1: 0.2}
```

Con ese hiperparámetro, lo que le indicamos al modelo es que ponga un esfuerzo del 80 % en los resultados que son 0, no victorias, y que ponga 20 % de esfuerzo en la identificación de las victorias; aumenta consideramblemente el modelo.

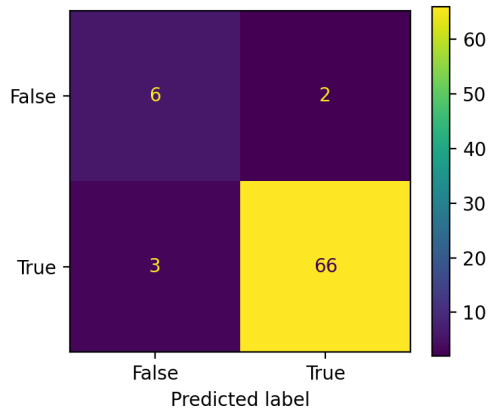
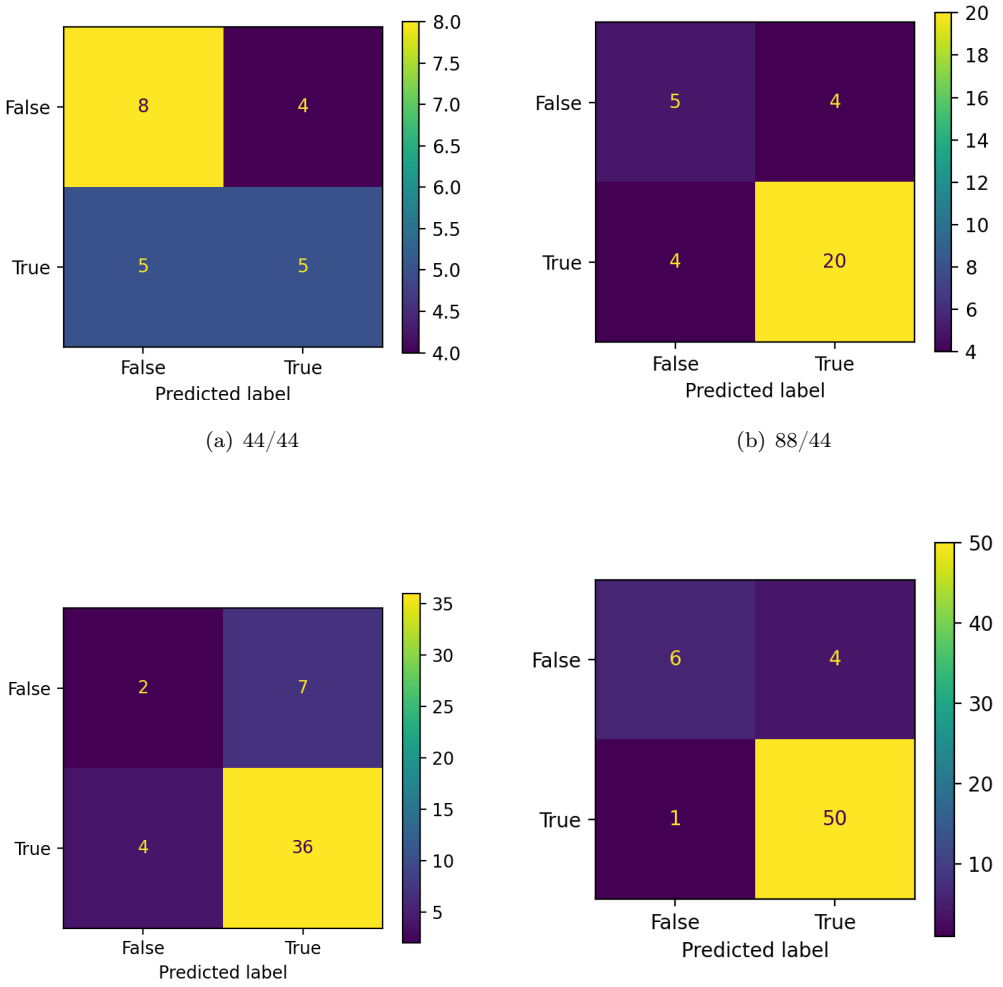


Figura 7: *Matriz de Aprendizaje*

Como se puede observar en la figura 7, ahora sí clasifica más derrotas y únicamente falla en 2 que deberían ser falsos y 3 que deberían ser positivos. Los puntajes obtenidos son: F1-score: 0.963, Accuracy = 0.935 y Precisión= 0.935. Se pueden considerar buenos puntajes, pero si se analizan los datos, se puede observar que en *accuracy* y la precisión son iguales, esto puede ser interpretado de varias maneras, pero de acuerdo al contexto del problema significa, como mencionamos, que caracteriza más como verdaderos sin importar los falsos, ya que es más fácil.

A continuación se trato de mejorar el modelo con *resampling*, el *resampling* consiste en recortar la base de datos para poder tener clases más igualitarias. Es decir, nuestra base de datos consiste en 264 *"TRUE"* y 44 *"FALSE"*, con el *downsampling*, es posible poder recortar la base de datos para poder encontrar un mejor balance. Se hizo con diferentes cantidades. Este paso se combinó con el balanceo del entrenamiento. Para combinaciones donde hay 44 *"TRUE"* y 44 *"FALSE"*, se hizo con

un balance de 50 % y 50 %, para un recorte al doble y al triple, es decir 88/44 y 150/44, se conservó el balance de 80 % y 20 %.



(a) 44/44

(b) 88/44

(c) 100/44

(d) 150/44

Figura 8: *Ciudades del mundo.*

Como se puede observar en la figura 8, para cuando se recortan mucho los datos, hay más errores para predecir las victorias, mientras conforme aumentan las victorias, disminuyen esos errores pero de igual forma hay menos errores en predecir cuando no es victoria. El mejor conjunto de datos fue el original, a pesar de que para cada conjunto de datos se encontraban diferentes hiperparámetros, el mejor conjunto de datos fue el que no estaba recortado. Es por eso que se decidió seguir con ese.

Se calculó el promedio del error logarítmico al cuadrado, el cual tiene un valor de 0.8311, el cual es bajo, lo que quiere decir que indica un mejor ajuste del modelo, ya que significa que los

logaritmos naturales de las predicciones se acercan más a los logaritmos naturales de los valores verdaderos en una escala logarítmica [Kapronczay, 2023].

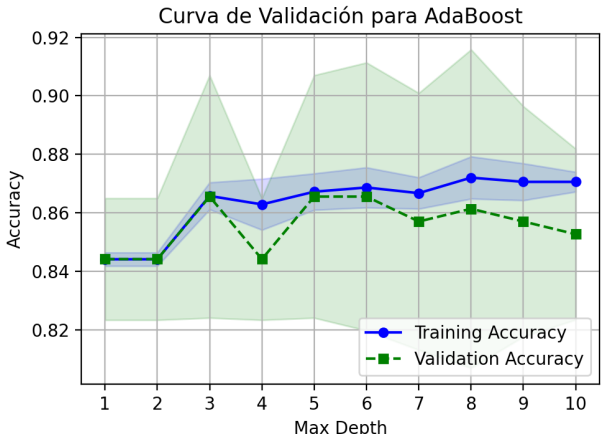


Figura 9: *Curva de Validación*

En la figura 9, se puede observar el *accuracy* conforme aumenta la profundidad dentro de los hiperparámetros, se puede observar que el *set de training* y el *set de validation* se puede considerar que van juntos, esto significa que en todo momento la validación del modelo mejora y que no importa la complicación de este.

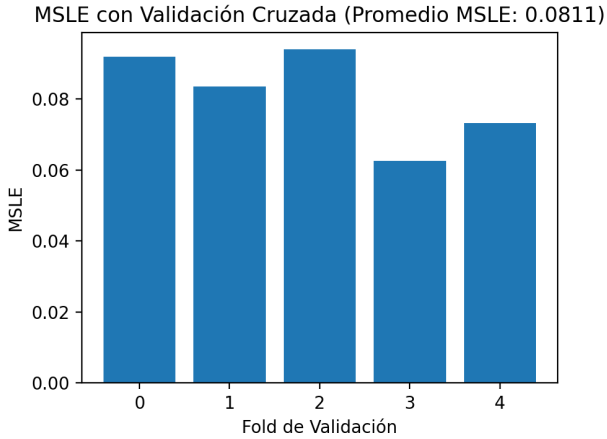


Figura 10: *MSLE*

Después se calcula el error logarítmico al cuadrado, pero con la validación cruzada para poder observar como cambia con el *fold* de validación, como se puede observar, no cambia mucho, además de ser un valor bajo, por lo que se considera que no hay un error considerable para el modelo.

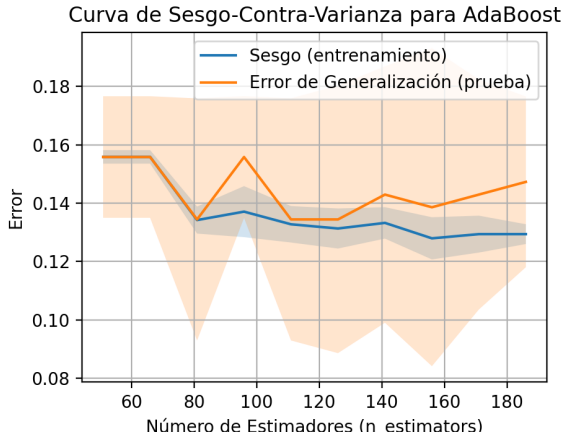


Figura 11: *Error vs n_ estimators*

En la figura 12, se puede observar como el sesgo disminuye a medida que aumentas el número de estimadores. Esto significa que el modelo *AdaBoost* se vuelve más flexible y puede ajustarse mejor a los datos de entrenamiento a medida que se agregan más estimadores débiles. Menor sesgo indica que el modelo se adapta mejor a los datos de entrenamiento y tiene una capacidad mejorada para capturar patrones complejos en los datos. Mientras que la varianza aumenta a medida que aumenta *'n_ estimators'*. Esto sugiere que el modelo *AdaBoost* se vuelve más propenso al sobreajuste a medida que se vuelve más complejo con más estimadores. Aunque el modelo puede adaptarse bien a los datos de entrenamiento, también puede volverse más sensible al ruido y menos generalizable a datos nuevos o de prueba. Cabe mencionar que en el primer enfoque se utiliza *'learning_ curve'* para calcular automáticamente las puntuaciones de entrenamiento y prueba [Singh, 2018].

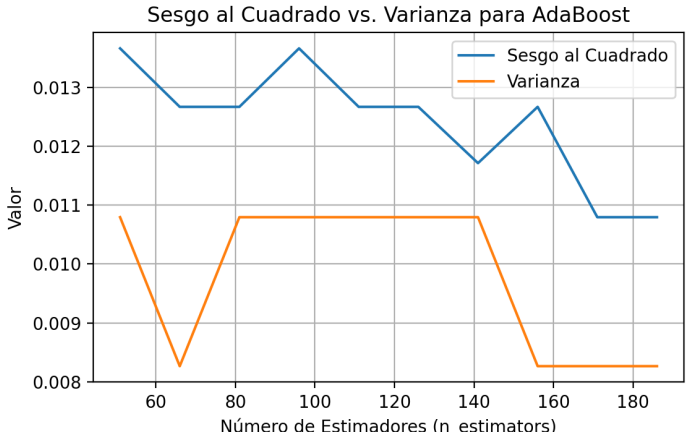


Figura 12: *Valor del bias y sesgo vs n_ estimators*

Ahora se obtuvo el $bias^2$ y la varianza pero en cada instancia de *'n_ estimator'* y además no se

calculan con una función. Como se mencionó anteriormente, el sesgo también se ve que disminuye, por lo que significa que el modelo *AdaBoost* está mejorando su capacidad para ajustarse a los datos de entrenamiento. A medida que se agregan más estimadores, el modelo se vuelve más flexible y puede capturar patrones más complejos en los datos y el modelo está aprendiendo de manera efectiva las relaciones en los datos. Por parte de la varianza, se observa que se mantiene estable e incluso disminuya al final, por lo que sugiere que el modelo *AdaBoost* está evitando el sobreajuste a pesar de volverse más complejo. A medida que el modelo se vuelve más flexible con más estimadores, logra mantener un buen equilibrio entre ajustarse a los datos de entrenamiento y mantener una capacidad de generalización adecuada en datos de prueba.

4. Conclusión

Los modelos para poder predecir cosas han sido una herramienta excepcional en la caracterización de los datos. Hablando del contexto en el que se predice si un equipo ganará un partido o no considerando el factor de que Messi haya anotado un gol en el primer tiempo podría ser un factor que decida el futuro de un equipo. Los diferentes modelos que se pueden implementar serán mejor que otros dependiendo de la naturaleza de los datos, en nuestro caso, el mejor modelo adaptado fue el *AdaBoost*. En las diferentes gráficas se puede observar como el error no es alto y que además el sesgo baja y la varianza, o no aumenta mucho, o disminuye con la complejidad del modelo. Al tratarse de un set de datos en el que hay demasiada inclinación hacía la clasificación de que sí ganó un partido, puede volverse un poco complicado la predicción; además se habla de fútbol, un deporte caracterizado por nunca ser algo seguro, incluso teniendo al mejor jugador de la historia jugando para ti. Pueden haber similitudes en los partidos donde Messi haya anotado gol en un minuto específico con un cierto marcador específico, y tener 2 resultados completamente diferentes, esto se puede observar en las casa de apuestas, como a pesar de que sucedan cosas así y la probabilidad sea mínima, tienen que asegurarse que no sea una apuesta segura.

Referencias

[Huet, 2023] Huet, P. (2023). Cómo entrenar un modelo de machine learning con scikit-learn. `https://openwebinars.net/blog/como-entrenar-un-modelo-de-machine-learning-con-scikit-learn/#:~:text=La%20curva%20de%20aprendizaje%20es,tama%C3%B1o%20del%20conjunto%20de%20entrenamiento.` [Accessed 10-09-2023].

[Kapronczay, 2023] Kapronczay, M. (2023). Mean squared error (mse) vs. mean squared logarithmic error (msle): A guide. https://builtin.com/data-science/msle-vs-mse. [Accessed 10-09-2023].

[Meadows, 2022] Meadows, M. (2022). Messi es ahora el mejor futbolista de todos los tiempos. https://www.dw.com/es/messi-es-ahora-el-mejor-futbolista-de-todos-los-tiempos/a-64154013. [Accessed 07-09-2023].

[Singh, 2018] Singh, S. (2018). Understanding the bias-variance tradeoff. https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229. [Accessed 10-09-2023].