La implementación del código tiene como objetivo poder hacer predicciones sobre resultados en los partidos de la Premier League con base en caracteírsticas y estadísticas del partido. A lo largo del código se tiene la intención de probar diferentes modelos de clasificación y mejorar el mejor con hiperparámetros para aumentar la precisión en las predicciones.

Para esto, se importan varios modelos de clasificación como lo son: Regresión Logistica, Random Forest Classifier, Supported Vector Machine, KNN Classifier, Decision Tree Classifier, Gradient Boosting Classifiery por último Ada Boost Classifier.

Empezando con la limpieza de datos, se eliminan columnas que no aportan ninguna información relevante para el desarrollo de un partido como lo son la fecha, el tiempo, reportes del partido, capitán. Además, se eliminan las columnas de '*goles a favor*' y '*goles en contra*', ya que estos en sí tienen el resultado de la predicción y el modelo no estaría aprendiendo. Después de quitar estos, con la librería missingno se hace una matriz para revisar que no haya datos faltantes, lo cual no hay, por lo que no se hará ninguna imputación de datos.

Para las columnas categóricas se hizo lo siguiente:

- Round: Esto indica la jornada (1 a 38) de la cuál es el partido, este dato puede tener mucha relevancia ya que es muy diferente un partido a comienzos de la temporada que a finales, sobretodo por la carga de partidos y el ritmo competitivo de los jugadores. Como los datos vienen de la siguiente manera: 'Matchweek 1', simplemente se quitó la palabra '*Matchweek* ' para dejar el número indicativo de la jornada.
- Día: De igual manera, el día de la semana que se juegue un partido puede cambiar el transcurso de este. Lo habitual es que las jornadas se den entre viernes y domingo, por lo que los partidos entre semana son más raros e indican una doble jornada, lo que puede suponer rotaciones en los equipos grandes y posibles cambios de resultados esperados.
- Venue: Indica si el partido es de local o de visitante, se le puso 1 a local y 0 a visitante ya que se tiene ventaja al ser local.
- Formation: La formación es una de las claves en el fútbol profesional, indica las posiciones de los jugadores y muchas veces si estos equipos serán ofensivos o defensivos. Se ordenó del más repetido al menos repetido y se le puso el número más grande al más repetido representando una ventaja contra los demás sistemas.
- Referee: El árbitro de igual manera tiene un gran peso en el partido, pues sus decisiones marcan el rumbo del partido, así como en las formaciones, se ordenó del más repetido al menos y así se numeró.
- Nombre de los equipos: Para esto, primero se homogenizó los nombres y se trató de hacer una numeración de los equipos que en los años recientes han sido mejores, a los cuales se les dió una mejor numeración contra los equipos con menos partidos.

Todas estas numeraciones en las variables tendrán mayor o menor peso para el modelo, es por eso que a lo mejor, o más repetido se les dió un mayor peso.

Una vez con los datos limpios y procesados se definió un diccionario con los modelos (inicializados con un random state), los cuales fueron entrenados y en un dataframe se guardaron los datos de su rendimiento para comparar los modelos.

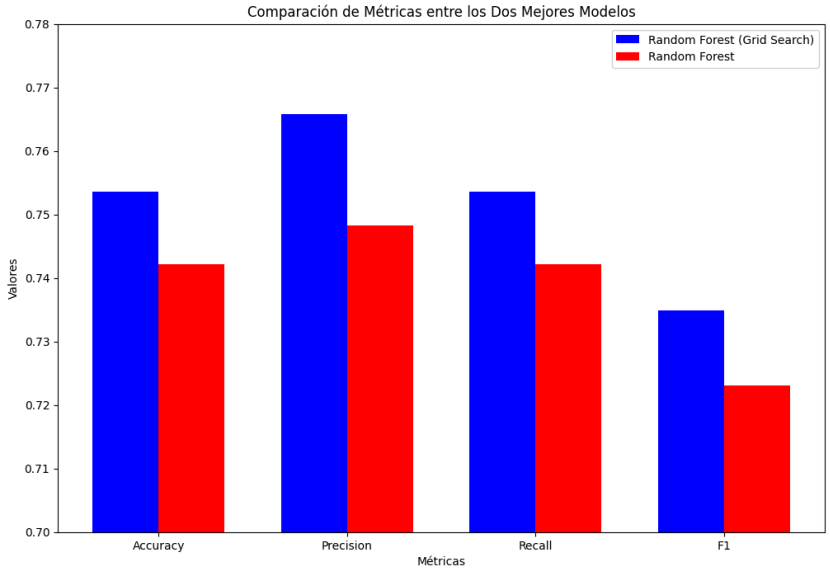
	Model	Accuracy	Precision	Recall	F1
1	Random Forest	0.742171	0.748335	0.742171	0.723025
4	Decision Tree	0.669102	0.664855	0.669102	0.666620
5	Gradient Boosting	0.644050	0.609807	0.644050	0.595175
6	AdaBoost	0.617954	0.557921	0.617954	0.565178
0	Logistic Regression	0.575157	0.505108	0.575157	0.510678
3	KNN	0.529228	0.525186	0.529228	0.516425
2	SVC	0.406054	0.164880	0.406054	0.234529

El modelo que mejor accuracy tuvo (con diferencia de 0.5), mejor precisión (diferencia de 0.8), mejor recall (diferencia de 0.8) y mejor F1 (diferencia de 0.7) fue el modelo Random Forest Classifier.

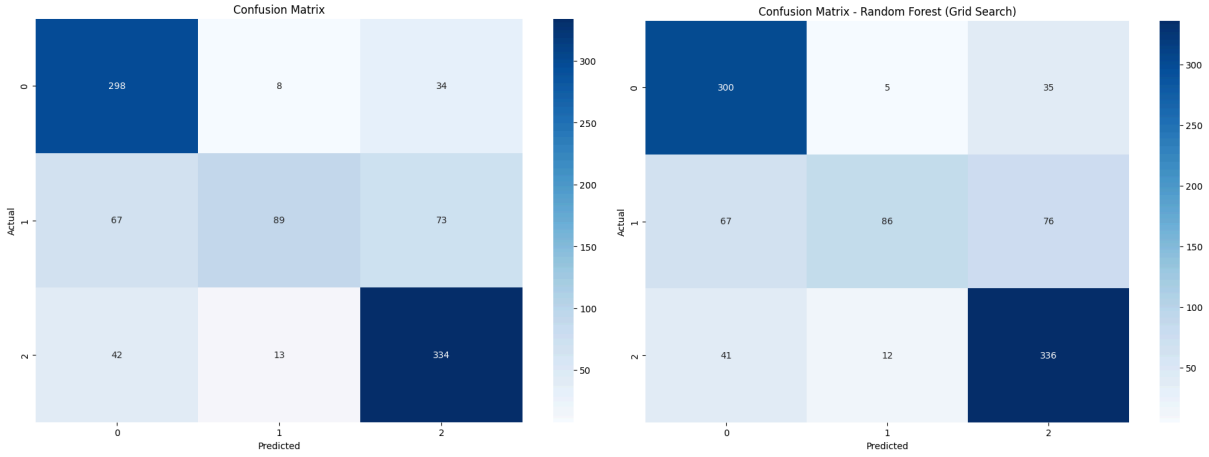
Una vez seleccionado este modelo, se realizó un grid search para encontrar los mejores hiperparámetros con los cuáles el modelo puede mejorar. Entre estos está el número de estimadores, la profundidad máxima de los árboles y la partición y hojas de los samples. Después de correr el código se obtuvo que los mejores resultados llegarían con:

- Max depth: 40
- min samples leaf: 1
- min samples split: 2
- estimators: 400

Se volvió a entrenar el modelo con estas mejoras y el modelo con sus parámetros de medición se agregaron al dataframe anterior. Y comparando solo los dos modelos de Random Forest se obtuvo la siguiente diferencia:



Lo que indica que en los 4 parámetros, el modelo de Random Forest con los hiperparámetros mejoró. De igual manera, se imprimeron las matrices de confusión para comparar rendimientos:



El nuevo modelo obtuvo mejores resultados en predecir correctamente las derrotas (0) y las victorias (2). Mientras que en los empates (1) tuvo peor rendimiento Aunque, los 2 modelos en general se parecen mucho y fallan específicamente en los empates, que los califican casi de igual manera como derrotas o victorias.