MODELO PREDICTIVO

El objetivo de este apartado es tomar el output de todas las transformaciones realizadas y explicadas anteriormente y obtener un modelo que sea capaz de predecir qué equipo va a ganar un partido. Para ello, se toma como input el csv input\_extendido que es el resultado de todas las transformaciones realizadas y explicadas anteriormente.

**FALTA UNA BUENA INTRODUCCION. LUEGO DE FINALIZAR LO HAGO**

El desarrollo del modelo se divide en cinco partes:

* **Parte I:** Exploración, limpieza, construcción/transformación de variables
* **Parte II:** Selección de variables.
* **Parte III:** Selección de Modelos.
* **Parte IV:** Construcción de modelos.
* **Parte V:** Evaluación del modelo.

**Parte I**

**Exploración, limpieza, construcción/transformación de variables**

En esta sección se realiza la preparación de los datos, limpieza del dataframe por valores nulos, renombrado de variables, cálculo de nuevas variables, descarte de variables que no forman parte de análisis, exploración gráfica y otros procesos. Haciendo uso de las librerías de *Pandas* y *Numpy* de *Python.*

En primer lugar, se importa el csv input extendido en un dataframe. Este dataframe inicial contempla 6220 filas y 180 columnas (variables). Después de un proceso de limpieza por valores nulos derivados del proceso de extracción y de cálculos de divisiones de denominadores iguales a 0, se ha obtenido finalmente 5809 (son menos hay que calcularlo si se considera necesario) filas para trabajar.

Seguidamente se pasa al **cálculo de variables compuestas** donde destaca, entre alguno de los cálculos, la creación de la variable ***“target”*** (objeto de estudio de valores binarios) a partir de la diferencia de los puntos marcados por el equipo local y el equipo visitante, de tal forma que, valores igual a 1 para indicar cuando el equipo local gana y 0 para el equipo visitante cuando pierde.

Entre las 180 variables que se han importado del .csv resaltan 2 variables categóricas que contienen la información de la división y la conferencia a la que pertenece el equipo local y el equipo visitante. Estas variables categóricas deben ser transformadas en variables dicotómicas (*dummy´s).* Esta transformación se ha realizado mediante el enfoque de *One Hot encoding* con funciones de la librería de *Pandas.* Dado que existen dos conferencias y seis divisiones en la NBA, han resultado 14 variables dicotómicas que sustituyen en cuanto a información a las dos variables originales.

Una vez identificadas, calculadas, transformadas, y organizadas todas las variables en diversas listas (variables continuas y dicotómicas) se decide filtrar las variables que *a priori* no aportan ningún valor al modelo, muchas derivan del proceso de extracción de datos y otras que, si bien tienen mucha información asociada, se han descartado por otras variables que en información ya las sustituye.

Básicamente un proceso de *“Garbage in, garbage out”* que nos permite trabajar específicamente con las variables de interés para el modelo.

Este proceso de selección se divide entre variables previas al partido y variables del partido se la siguiente forma:

**(se puede hacer una especie de cuadro o algo más visual)**

**Variables previas al partido:**

**REVISAR NOTEBOOK, YA TODO ESTO ESTA ESCRITO**

**Variables durante el partido:**

**REVISAR NOTEBOOK, YA TODO ESTO ESTA ESCRITO**

**Parte II**

**Selección de variables**

En esta apartado se realiza la selección definitiva de variables para el modelo desde distintos enfoques que dependen del tipo de variable. Inicialmente se han separado las variables en diferentes listas; continuas y dicotómicas, así como, la separación del conjunto de entrenamiento y test para la evaluación del modelo final.

Este proceso de separación de los datos para el proceso de selección de variables se realiza para evitar extraer información del total conjunto de datos, ya que, posteriormente una parte de estos datos se deben usar para evaluar el modelo. Comúnmente en aprendizaje automático a este fenómeno se le conoce como “fuga de datos” (*data leakege*).

Haciendo uso de la librería de ***sklearn*** de Python se ha desarrollado todo el proceso de aprendizaje automático. Sobre la función *train\_test\_split* se han separado los datos en un conjunto de entrenamiento (70% equivalente a 4.066 partidos) y otro para test del modelo (30% equivalente a 1.743 partidos), recordando que el rango de temporadas seleccionadas para el estudio es 2016-2020.

Es importante resaltar que, durante la fase de análisis exploratorio (Parte I) se realizaron matrices de correlación sobre las variables continuas previas y durante el partido (por separado) sobre *“target”* para evaluar el posible grado de relación lineal que tienen las variables entre sí y la variable objetivo.

En este análisis de correlación no se han descartado variables porque en este estudio no se pretende realizar un análisis de causalidad entre variables explicativas y la variable objetivo. Este proceso de selección de variables tiene sentido en esos casos porque se pretende cuantificar la atribución de una variable respecto a otra y de esta forma el nivel de importancia.

En el caso de modelajes predictivos es un error excluir variables por presencia de altos niveles de correlación entre variables explicativas (excepto perfecta colinealidad entre independientes) porque se contribuye al deterioro de la precisión del modelo, esto se debe porque las variables en conjunto se complementan y comprenden más información que cada una de estas por separado.

Del mismo modo, el descarte de variables por escasez de correlación es debatible, depende en muchos casos el modelo a utilizar y de la relación establecida a priori entre la variable y la objetivo. El coeficiente de correlación de Pearson, por ejemplo, es una medida de la relación "lineal" entre dos variables, en este contexto, se puede decir que no necesariamente las variables tienen que estar relacionadas linealmente.

Asimismo, en el apartado anterior (Parte I) se han descartado variables que pretenden aportar la misma información en el modelo. En cualquier caso, de requerirse para modelos futuros, se ha dejado en el notebook **funciones\_modelo.ipynb** la función construida para Factor Inflación de la Varianza (VIF) que permite de forma automática, descartar variables explicativas dependiendo del nivel de correlación establecido.

A continuación, separamos el proceso de selección de variables como se ha comentado en el inicio de este apartado:

1. **Selección de Variables Dicotómicas:**

Después del proceso de transformación de variables categóricas comentado en la primera sección y sumado a las variables dicotómicas generadas en previos procesos del estudio, se han obtenido en total 26 variables dummy´s que se reparten en partes iguales entre el equipo local y visitante.

|  |  |
| --- | --- |
| played\_previous\_date | Conf\_Este |
| played\_two\_days\_ago | Div\_Atlantic |
| played\_three\_days\_ago | Div\_Central |
| played\_prorrogue\_previous\_date | Div\_Northwest |
| played\_prorrogue\_two\_days\_ago | Div\_Pacific |
| played\_prorrogue\_three\_days\_ago | Div\_Southeast |
|  | Div\_Southwest |

Estas variables se han calculado para enriquecer la complejidad del modelo, sin embargo, como es posible sospechar, muchas de estas pueden que no sean relevantes para predecir el resultado de un juego, y su inclusión en el modelo solo contribuyen a la generación de “ruido” (perturbaciones sobre las predicciones).

Por lo tanto, es necesario conocer cuáles de estas variables son las que más información aportan o afectan el comportamiento de la variable objetivo. Entre las distintas técnicas y conceptos estadísticos que existen se ha decidido utilizar la **prueba de independencia de chi-cuadrado** y la medida de **información mutua.**

La **prueba de independencia de ji-cuadrado** (chi-cuadrado) para variables categóricas contrasta la hipótesis nula de que las variables dentro de la prueba son independientes a través de la distribución que presenta una frente a la otra mediante tablas de contingencia.

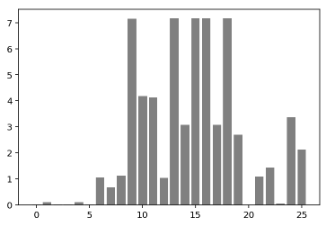
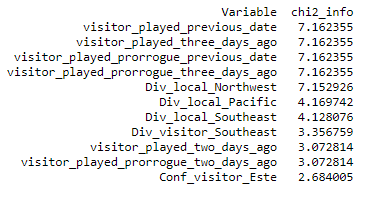
Mientras que, la **información mutua** (*mutual information*) es una cantidad que mide la relación entre dos variables aleatorias que se muestrean simultáneamente. En particular, mide cuánta información se comunica, en promedio, en una variable aleatoria sobre otra.

Un teorema importante de la teoría de la información dice que la información mutua entre dos variables es 0 si y solo si las dos variables son estadísticamente independientes.

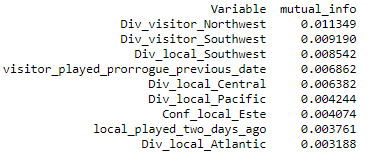
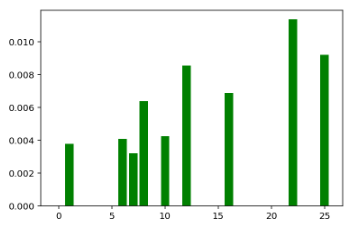
En efecto, el mismo resultado que evaluamos cuando se realiza la hipótesis nula de una prueba de chi-cuadrado y se espera rechazar cuando se realiza selección de variables categóricas, por ejemplo.

Más allá del criterio empírico del proceso estadístico, luego de evaluar los resultados de ambos enfoques, se decide utilizar las variables seleccionadas por el cálculo de la información mutua, a priori, parece que las variables seleccionadas por este método tienen mayor importancia en cuanto al criterio del deporte del baloncesto respecta.

Las variables dummy´s seleccionadas por chi-cuadrado son las siguientes:



Las variables *dummy´s* seleccionadas por *información mutua* son las siguientes:

1. **Selección de Variables Continuas:**

Derivado del proceso de filtrado de *“Garbage in, garbage out”* en el apartado de análisis exploratorio (Parte I) se obtuvieron 53 variables continuas. Al igual que el proceso de selección anterior, se utilizaron diversos métodos para descartar las variables no relevantes para predecir el resultado de un juego, entre las que destacan: *Genetic Algorithms, Permutation Importance y Random Forest Feature Importance.*

En entornos de aprendizaje automático podemos citar a Joaquín Amat Rodrigo para definir que los algoritmos genéticos son “métodos de optimización heurística que, entre otras aplicaciones, pueden emplearse para encontrar la combinación de variables que consigue maximizar la capacidad predictiva de un modelo”.

El primer intento de selección de variables se basó en este método, utilizando básicamente adaptaciones de códigos suministrados en una asignatura durante el Máster y aplicaciones técnicas encontradas en la web. La maximización de la capacidad predictiva del modelo se fundamentó en la medida del índice de impureza de Gini.

Según O’Relly Machine Learning Book etc etc **(cita).**“Dada una muestra, la impureza de Gini mide la probabilidad de una clasificación errónea si una etiqueta se elige aleatoriamente usando la distribución de probabilidad de la rama. El índice alcanza su mínimo (0.0) cuando todas las muestras de un nodo se clasifican en una sola categoría”.

El segundo proceso de selección de variables fue el *Random Forest Feature Importance* usando adaptaciones del código propuesto por la librería de *sklearn* en la web. Según la web este algoritmo es un “meta-estimador que ajusta un número de clasificadores de árbol de decisión en varias submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión predictiva y controlar el sobreajuste”.

El criterio de selección de importancia de variables por defecto de la librería de *sklearn* es basado en la medida del índice de impureza de Gini (*Gini impurity index*).

La importancia de las variables basadas en impurezas se fundamenta en el criterio de “cuanto más alto el índice, más importante es la variable”. Y esta importancia se calcula como la “reducción total (normalizada) del índice de cada variable”.

Finalizado el proceso de selección anterior, se decide realizar un último proceso de selección llamado “*Permutation Importance*” derivado de las recomendaciones de la propia librería de *sklearn* por las carencias que tiene el algoritmo anterior sobre el criterio de selección basado en la medida de importancia de Gini; en la web indican que este proceso de selección puede ser “engañoso” para variables de alta cardinalidad (sobrestimación de su importancia), asimismo, demuestran cómo el algoritmo tiende a sobreestimar la importancia de variables que no tienen relación alguna con la variable objetivo.

La “importancia de permutación” de una variable se calcula de la siguiente manera:

Primero, se evalúa una métrica de línea base (baseline metric), definida por una puntuación (scoring) en un conjunto de datos (diferente a la línea base). A continuación, se permuta una columna de una variable del conjunto de validación y se evalúa nuevamente la métrica. La importancia de la permutación de cada variable se define como la diferencia entre la métrica de línea base y la métrica obtenida de permutar la columna de la variable.

Cada uno de los métodos mencionados anteriormente arrojaron resultados diferentes, sin embargo, a priori parece existir un patrón entre el número optimo de variables necesarias para maximizar la métrica con la que se estuviese evaluando cada método. Este hallazgo es importante, y es la razón por el cual se ha decidido contrastar diferentes metodologías para la selección de variables. Si bien cada método le dio más o menos importancia a algún grupo de variables en concreto, todas resultaron arrojar entre 22 y 27 variables esencialmente importantes para maximizar el desempeño del modelo.

Se decidió elegir finalmente las variables por el criterio de “importancia de permutación”; específicamente por mayor conocimiento del cálculo de la métrica de selección (comparado con el complejo proceso del proceso del Algoritmo Genérito) y porque este proceso es recomendado por la librería de *sklearn* por encima del proceso del *Random Forest Feature Importance.*

Las variables continuas más importantes seleccionadas por cada uno de los métodos seleccionados anteriormente son las siguientes:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**\*Nota 1:**

El proceso de selección de variables forma parte del proceso de optimización de parámetros durante la construcción y evaluación del modelo, en este contexto, no existen garantías de uso exclusivo de las variables seleccionadas en esta sección. La totalidad de estas variables se utilizan como base para la construcción del modelo, luego durante el proceso de optimización de parámetros y evaluación del modelo se determina si la inclusión (exclusión) de variables mejora o no el desempeño del modelo.

Una vez realizado el diagnostico de las curvas de aprendizaje de los modelos y de las métricas de evaluación, se encontró (derivado de muchas pruebas de ensayo y error) que la inclusión de la totalidad de las variables del conjunto de entrenamiento contribuiría a la mejora de la complejidad del modelo sin incidir en el detrimento de las métricas de evaluación.

Según el costo de oportunidad de “alto sesgo” y “baja varianza” *(high bias and low variance)* analizado en las curvas de aprendizaje, se prefirió incrementar la complejidad de los modelos (mayor varianza) y en efecto, reducir el sesgo.

Cuanto menos sesgado sea un modelo (magnitud del error), mayor será su capacidad para ajustar bien los datos (cuidando el riesgo del *overfitting*). Cuanto mayor es la capacidad del modelo para ajustarse bien a los datos, mayor es la varianza (medida de dispersión entre modelos respecto al modelo poblacional).

Finalmente, con esta decisión hemos optimizado dos modelos muy diferentes. El primero con un alto sesgo (baja varianza), y el segundo con alta varianza (bajo sesgo). Por lo que el primero tiene una mayor capacidad de generalización (replicar el comportamiento aprendido en el conjunto de entrenamiento en el conjunto de validación) y el segundo, un modelo más complejo que se ajusta al patrón observado en el conjunto de entrenamiento.

**Parte III**

**Selección de Modelos**

En esta sección se definen los algoritmos de aprendizaje automático que se utilizan para crear los modelos que mejor se ajustan a los datos extraídos, de esta forma tratar de encontrar el modelo con mayor capacidad predictiva sobre los resultados de los partidos de la NBA.

Sobre la división de los datos en el conjunto de entrenamiento y test realizado en la sección anterior se debe usar exclusivamente el conjunto de entrenamiento para preparar el algoritmo. Sin embargo, ¿es este conjunto de entrenamiento una muestra lo suficientemente representativa para que un modelo tenga la capacidad de aprender el patrón de datos que define el ganador de un partido? No lo sabemos, a priori se debe afirmar esta hipótesis de forma imperativa y en cualquier caso, de ser necesario, mejorar las condiciones de los datos de tal forma que lo que observe el algoritmo en el patrón de los datos sea lo más semejante a la realidad de los datos “poblacionales”.

Cuando el conjunto de datos no representa todas las clases de datos por igual, por ejemplo, el modelo podría ajustarse en exceso a la clase que está más representada en los datos que está observando (en nuestro caso como es de esperarse mayores juegos ganados por parte del local) y pasar por alto la existencia de la clase minoritaria (los juegos perdidos del local), lo que conlleva a mermar la capacidad de generalización del modelo sobre predicciones de datos no observados.

Se ha evidenciado presencia de datos desbalanceados desde el inicio del proyecto, sin embargo, esta ha disminuido a medida que hemos podido ingresar y enriquecer más datos de diferentes fuentes.

A inicios del proyecto se evidenciaban niveles de 70%-30% (sobre la variable target), y se ha podido reducir esta diferencia hasta un 60-40%.

Con esta distribución no parece tener problemas la construcción del modelo, en cierto sentido, el modelo debe tener una ligera preferencia (como lo es en la realidad) porque gane el local.

En cualquier caso, de no poderse obtener mayores datos: ¿cuál es la forma correcta de afrontar este problema?

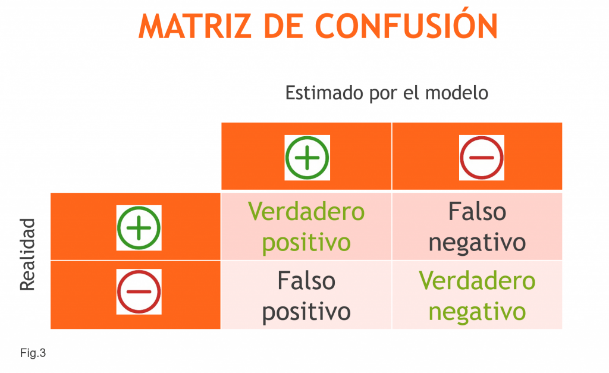
En teoría se debería realizar una operación de sobremuestreo al azar (*Random Oversampling*) ¿por qué? (colocar cita o viñeta que indique información del libro + links) + definición

Una vez identificado y contrastado que no existe presencia de datos desbalanceados. Nos preguntamos ¿Qué métricas se utilizan para evaluar el desempeño de un modelo? Y ¿Cómo se mide la capacidad de generalización del modelo?

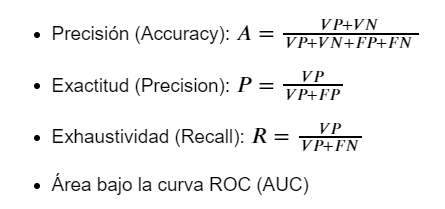
En modelos supervisados de clasificación (problemas de clasificación de clase binario), en la que los resultados se etiquetan positivos (P) o negativos (N), hay cuatro posibles resultados que se pueden obtener:

* Verdaderos Positivos (VP)
* Verdaderos Negativos (VN)
* Falsos Positivos (FP) o Error tipo I
* Falsos Negativos (FN) o Error de tipo II

Para facilitar el análisis, los resultados se pueden organizar en una matriz de confusión:



A partir de estos resultados se pueden construir diferentes métricas para evaluar la calidad de los modelos construidos, algunas de las más utilizadas son:



Cuadro

En scikit-learn todas estas métricas se encuentran en la librería ***metrics***

Entre las métricas más utilizadas, usaremos la Precision (*Accuracy*) y Área Bajo la Curva ROC (*AUC*) para definir cual algoritmo de aprendizaje automático elegir entre los distintos que se pretenden examinar.

Este proceso de selección se explicará más adelante, por ahora centraremos la atención en explicar brevemente las dos métricas que utilizaremos para este proceso.

Empezando por la más sencilla, la precisión (*Accuracy*). A partir de ahora nos referimos a esta métrica con su nombre en inglés para evitar confusiones. Informalmente, el *Accuracy* es la fracción de predicciones que el modelo realizó correctamente, mientras que, formalmente se calcula **como se muestra en el cuadro 1.**

En aprendizaje automático es ampliamente conocido evaluar modelos a través del cálculo del área bajo la curva (AUC por sus siglas en inglés). AUC significa "área bajo la curva ROC". Esto significa que el AUC mide toda el área bidimensional por debajo de la curva ROC desde los puntos (0,0) hasta (1,1).

El área bajo la curva es una métrica de evaluación de modelos que considera todos los umbrales de clasificación posibles. Entendiéndose umbrales de clasificación como; criterio de valor escalar que se aplica a la predicción de un modelo para separar la clase positiva de la negativa (tal como hemos observado anteriormente en la matriz de confusión).

Matemáticamente la curva ROC es la **tasa de verdaderos positivos** frente a la **tasa de falsos positivos** en diferentes umbrales de clasificación.



Una de las bondades de utilizar el área bajo la curva ROC es que está estrechamente relacionada con el coeficiente de Gini que en muchos casos se utiliza como una medida alternativa. Si recordamos en el apartado anterior se ha comentado que el índice de Gini en la librería de ***sklearn*** es la métrica utilizada por defecto.

Comúnmente se conoce el índice de Gini como el doble del área entre la curva ROC y la diagonal (línea de 45 grados que pasa por el origen).



Finalizado esta breve explicación de las métricas a considerar para evaluar el modelo, se continua con el proceso de selección de estos.

Siguiendo la extensa literatura sobre los modelos de aprendizaje automático que plantean problemas de clasificación de clase binario para la predicción del ganador de un juego del baloncesto podemos darnos cuenta de que muchas de las conclusiones identifican que los modelos basados en estimaciones lineales son los que mejores resultados arrojan.

Por citar algunos, por ejemplo, *"Prediction of NBA games based on Machine Learning Methods"* [Torres 2013] en base a sus resultados utilizando regresiones lineales concluye que en comparación con otros estudios pilares del conocimiento en el área como lo es *"NBA Oracle"* [Beckler et. al. 2008] han podido lograr resultados muy satisfactorios y cercanos a estos, entorno a un accuracy del 60% y 70%, mientras que *"NBA Oracle"* [Beckler et. al. 2008]entre todos sus modelos complejos de aprendizaje automático el mejor resultado ha sido un 73% de accuracy.

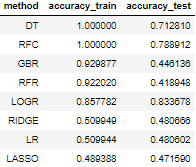
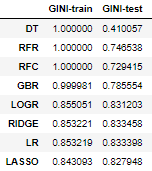
En *"NBA Oracle"* han podido experimentar con modelos de *Logistic Regression, Support Vector Machines, Logistic Regression, y Artificial Neural Networks*. Concluyendo brevemente al igual que [Torres 2013] que los modelos con mejores resultados en el baloncesto son los más sencillos, es decir, las regresiones lineales.

Finalmente, es importante comentar el estudio de “*Nba Game Prediction Based On Historical Data And Injuries” [Dionny et. al. 2016]* en el que destaca entre todos sus modelos de aprendizaje automático la regresión lineal con un accuracy del 68%.

Más allá de seguir los descubrimientos de la literatura de aprendizaje automático aplicado al entorno del deporte, este proyecto no pretende sesgar el enfoque al descubrimiento que han realizado otros, es por esto que desde una perspectiva neutra se han evaluado 8 algoritmos en su estructura de parámetros “más sencilla” con las métricas comentadas anteriormente para decidir al menos 2 para desarrollar con mayor profundidad.

Utilizando un conjunto de funciones, procesos y conocimientos adquiridos durante el Máster se ha desarrollado un proceso automático de selección de modelo basado principalmente en la comparación de métricas como el accuracy y el índice de Gini en un resumido dataframe **(para mayor detalle revisar el notebook del modelo)**

En resumen, los 8 modelos seleccionados con parámetros por defecto (según la sklearn) para la comparación de métricas son los siguientes:



Donde:

* DT: Decision Tree
* RFR: Random Forest Regression
* RFC: Random Forest Classifier
* GBR: Gradient Boosting Regression
* LOGR: Logistic Regression
* Ridge
* LR: Linear Regression
* Lasso

Tal como se ha comentado en el inicio de este apartado (Parte III), el objetivo es determinar el modelo de aprendizaje automático que mejor se ajusta a los datos extraídos.

En este contexto, solo se deben considerar los resultados en entrenamiento e ignorar así los resultados en test; este último nos sirve de referencia para determinar el porcentaje de caída de las métricas de evaluación o de manera informal determinar la capacidad de generalización del modelo (sabiendo que los parámetros de todos los modelos vienen por defecto y no se ha realizado una selección de variables importantes).

Sobre el dataframe de resultados de Gini; se observan muy buenos resultados en general, entre otras cosas, los modelos de árboles de decisión sobre ajustados (comportamiento usual en este tipo de algoritmos complejos) y los modelos lineales con mayor capacidad de generalización. (según el criterio del porcentaje de caída de las métricas en el conjunto de test).

Sobre el dataframe de resultados de Accuracy; destacan los modelos de árboles de decisión con elevados resultados en entrenamiento y elevada perdida de capacidad de generalización. De igual forma destaca el comportamiento del Logistic Regression entre los modelos lineales en cuanto a la capacidad de generalización.

Posterior al análisis, se decide elegir el *Random Forest Classifier* y el *Logistic Regression* para el desarrollo de los modelos, proceso que conlleva tanto la optimización de hiperparámetros (hyperparameter optimization) como la selección de variables óptimas para cada uno.

**Si existen dudas sobre el proceso de selección de variables optimas leer Nota 1.** Aun cuando en el apartado anterior (Parte II) se han determinado las variables más importantes, estas se han utilizado como variables “base” para el proceso de selección del modelo (Parte III).

**Parte IV**

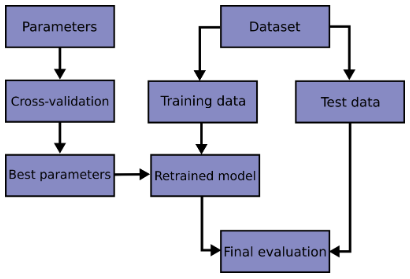
Construcción de modelos

“Aprender los parámetros de una función de predicción y probarlo con los mismos datos es un error metodológico: un modelo que simplemente repita las etiquetas de las muestras que acaba de ver tendría una puntuación perfecta pero no podría predecir nada útil todavía en datos que no ha visto”.

Aun cuando se pruebe el modelo en datos que no ha visto (conjunto de test) y se observen buenos resultados (como se comentó en la sección anterior en el proceso de selección de modelos). ¿Son estos resultados lo suficientemente robustos? O ¿es posible que un componente aleatorio influya en los resultados? Dependiendo cómo dividamos los datos en estos dos conjuntos (entrenamiento y test) tendremos una estimación diferente del error de generalización. En unos casos pensaremos que el modelo generaliza mejor y en otros peor.

Para incrementar la robustez de los resultados (minimizando el componente aleatorio) y evitar el sobreajuste de los modelos existe la validación cruzada (*cross-validation*). Una práctica común de su uso es para determinar los parámetros óptimos de un modelo.

Aquí hay un diagrama de flujo de trabajo típico de validación cruzada en el entrenamiento de modelos (extraído directamente de la web de sklearn).



Los mejores parámetros pueden determinarse mediante diversas técnicas, en sklearn existen dos enfoques: *RandomizedSearchCV y Grid Search.*

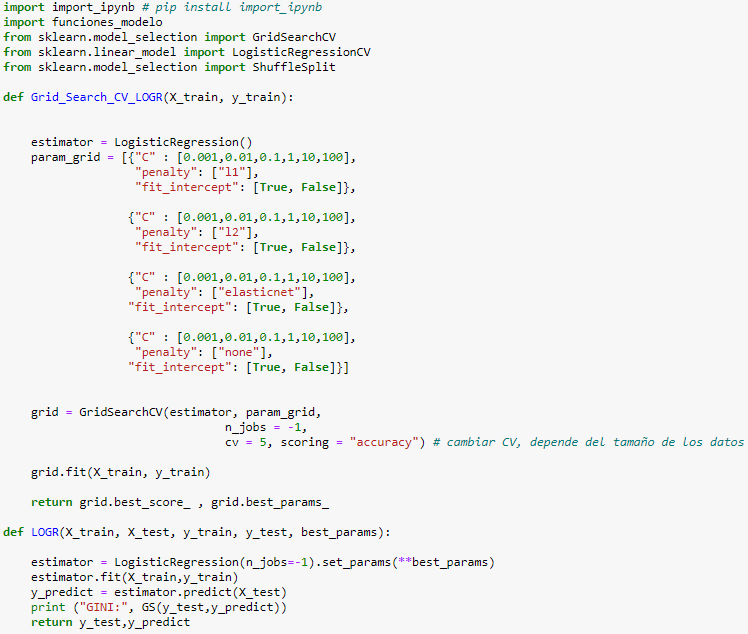
En este estudio se ha decidido usar la función de *Grid Search* aun cuando su costo computacional y tiempo es más elevado, su desempeño en la búsqueda es mejor; el algoritmo utiliza exhaustivamente todas las combinaciones posibles de los parámetros especificados.

Para el **Random Forest Classifier** el tiempo de ejecución del proceso fue aproximadamente 45 minutos, mientras que, el **Logistic Regression** fue un total de 5 segundos. Ambos en un equipo promedio con una memoria RAM de 8 gigabytes.

En resumen, se puede entender con la comparación de ambos tiempos de ejecución la magnitud de la diferencia de complejidad de ambos algoritmos, aun cuando en ambos procesos existen diferencias en parámetros y el número de iteraciones necesarias para encontrar el parámetro óptimo.

A continuación, se pueden observar las funciones creadas para ambos procesos. Posteriormente mediante *loops* los resultados fueron arrojados las veces que se fuese necesaria, para ambos casos se obtuvieron 5 resultados. (Revisar notebook para mayor detalle).





En las funciones anteriores no se presentaron la totalidad de parámetros para cada modelo, a medida que se introducen mayores parámetros incrementa el tiempo de ejecución del código, por lo que se han establecido solo los prioritarios para conocer los umbrales de los parámetros numéricos, por ejemplo, o al menos determinar un patrón de criterios que mayor peso tienen respecto a las métricas de evaluación.

Asimismo, es importante destacar que el proceso de optimización de parámetros no finaliza con los procesos establecidos anteriormente, se debe recordar que estos parámetros solo garantizan resultados robustos en cuanto al conjunto de datos de entrenamiento, por lo que se deben probar en el conjunto de test. En muchas ocasiones la combinación de parámetros se debe ajustar mediante ensayo y error de forma manual en el conjunto de datos de test.

En el siguiente apartado se proporciona más información acerca de la función y de la importancia de cada parámetro, así como el ajuste final de cada uno para garantizar los resultados mas cercanos posibles a los encontrados durante la validación cruzada.

**Parte V**

Evaluación del modelo

Derivado del proceso automático de selección de parámetros explicados en la sección anterior y del proceso de ensayo y error de forma manual durante la evaluación del modelo en el conjunto de test, se han optimizado los modelos evitando el sobreajuste y priorizando el máximo desempeño posible sobre las métricas de *accuracy* y el índice de Gini.

Para el *Random Forest Classifier* se han optimizado los siguientes parámetros (para mayor detalle revisar el modelo):

* ***Bootstrap:*** booleano que determina el proceso de entrenamiento de cada árbol de decisión (muestreo aleatorio con reemplazamiento o todos los datos de la muestra).
* ***Criterion:*** función para medir la calidad de la división de los árboles de decisión (impureza de Gini o entropía para *information gain)*
* ***max\_features:*** la cantidad de variables a considerar cuando se busca la mejor división.
* ***min\_samples\_split:***  representa el número mínimo de muestra necesaria para dividir un nodo interno. Incrementar este parámetro limita el crecimiento de cada árbol, ya que cada *Split* debe considerar mas muestras en cada nodo.
* ***n\_estimators:*** representa el número de árboles en el bosque. Comúnmente a mayor número el algoritmo aprende mejor los datos (incrementa el coste computacional)
* ***max\_depth:***  representa la profundidad de cada árbol en el bosque. A mayor profundidad, mayor divisiones y en efecto, captura más información. (debe controlarse para evitar la complejidad del modelo y el coste computacional).

Para el *Logistic Regression* se han optimizado los siguientes parámetros (para mayor detalle revisar el modelo):

* **fit\_intercept:** booleano que especifica si se debe incorporar una constante o no en la función de decisión.
* **Penalty:** especificar la norma utilizada en la penalización de la regresión. Permite controlar la complejidad del modelo.
* **C:** la inversa de la fuerza de regularización. A menores valores, mayor fuerte la regularización. Permite controlar la complejidad del modelo.

ANEXOS

