Análisis del conjunto de datos de enfermedades crónicas renales

Fecha de actualización: 16/03/2019

1. Preprocesamiento de los datos

En esta sección se describe la base datos, así como se detallan los acciones de preprocesamiento realizadas.

1.1 Descripción de la base de datos

Hasta la fecha en el que se realizó el análisis, la base de datos está compuesta por 2047 pacientes, con un total de 7634 visitas registradas. Existe un promedio de 3.7 visitas por paciente, la cantidad máxima de visitas de un paciente es 13 y la cantidad mínima es 1. Cada paciente está identificado por un identificador numérico único (Id).

Los pacientes están descrito por variables demográficas y por variables relacionadas con cada una de las visitas y tratamientos. La relación entre las variables demográficas y los pacientes es de *1:1*, mientras que la relación entre paciente y variables correspondientes a visitas y tratamientos es de *1:m*.

La Figura 1 muestra las variables demográficas a considerar en el análisis de datos. Disponemos de las siguientes variables demográficas:

- Tres variables que almacenan fechas (Date)
- Ocho variables categóricas (factor). De las ocho variables, cuatro son variables que pueden ser consideradas binarias porque solo tienen dos posibles valores (Sexo, Dx_histológico, Patología_extrarrenal y Antecedentes_familiares), mientras que las otras cuatro variables son nominales (variables con más de dos valores).
- Dos variables numéricas reales relacionadas con el peso y las semanas de gestación al nacer.

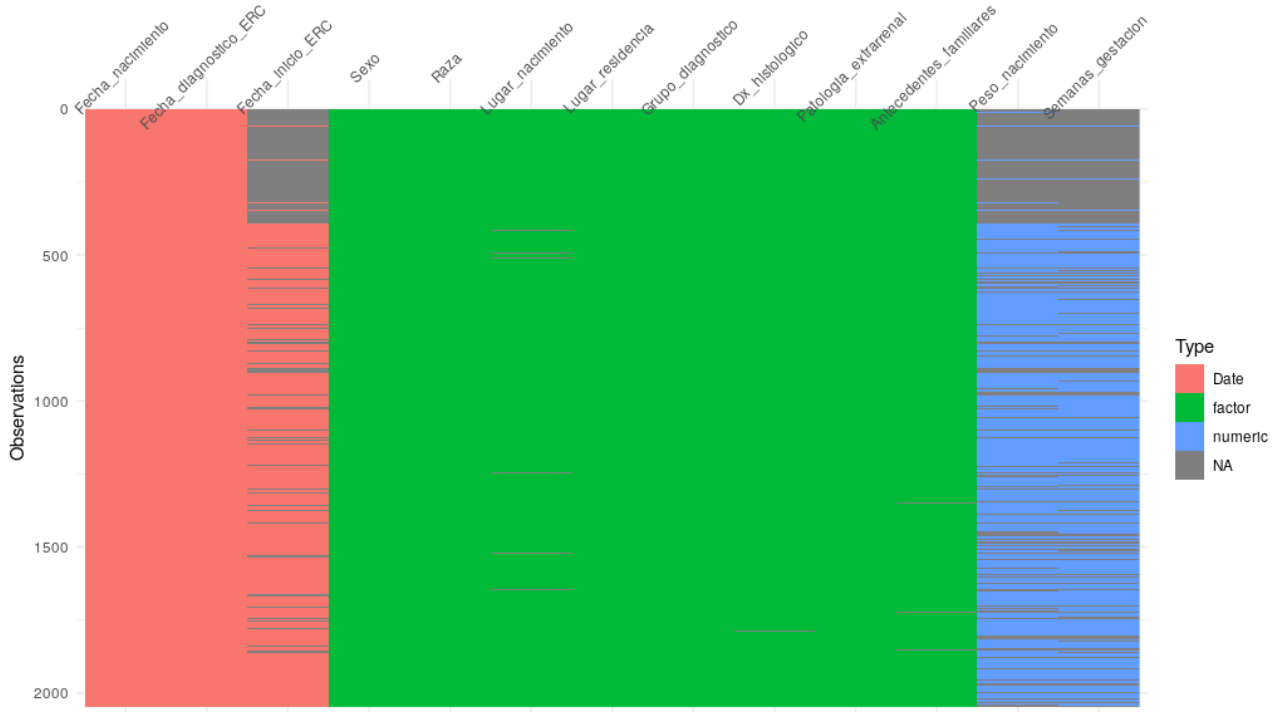


Figura 1: Tipo de variables demográficas consideradas en el análisis. NA *significa valores perdidos en las variables.*

La Figura 2 muestra las variables relacionadas con las visitas. En este caso, disponemos de las siguientes variables:

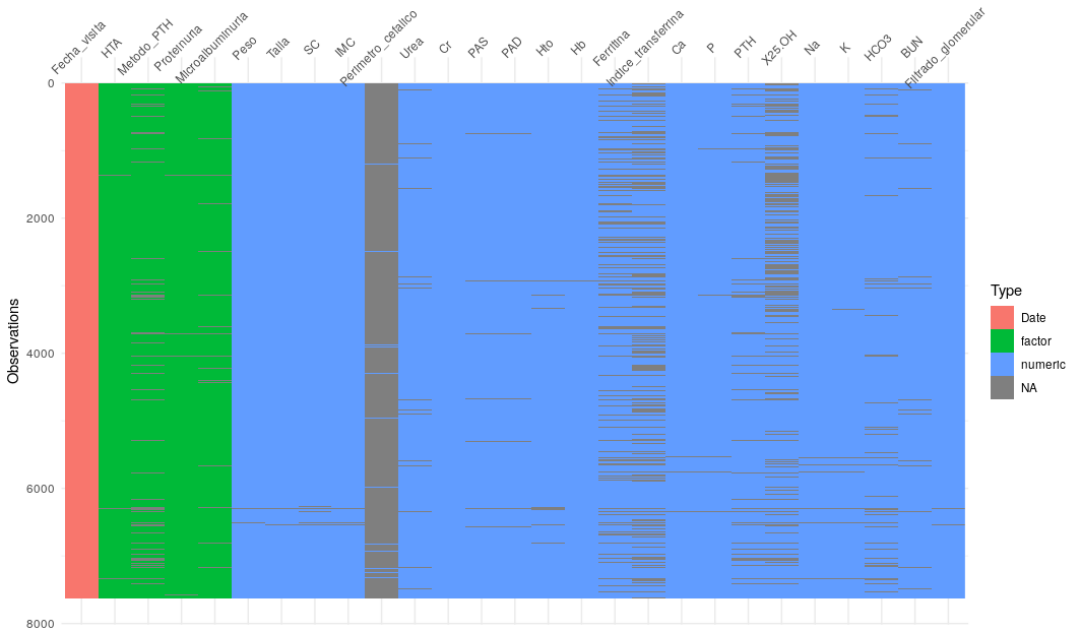


Figura 2: Tipo de variables de visitas consideradas en el análisis. NA *significa valores perdidos en las variables.*

- Una variable que representa la fecha de la visita.
- Cuatro variables categóricas. Dos de estas variables son binarias (HTA y Metodo_PTH), mientras que las otras dos son nominales.
- 22 variables numéricas reales.

La Figura 3 muestra las variables relacionadas con los tratamientos. En este caso tenemos 26 variables binarias que representan si un tratamiento es aplicado o no.

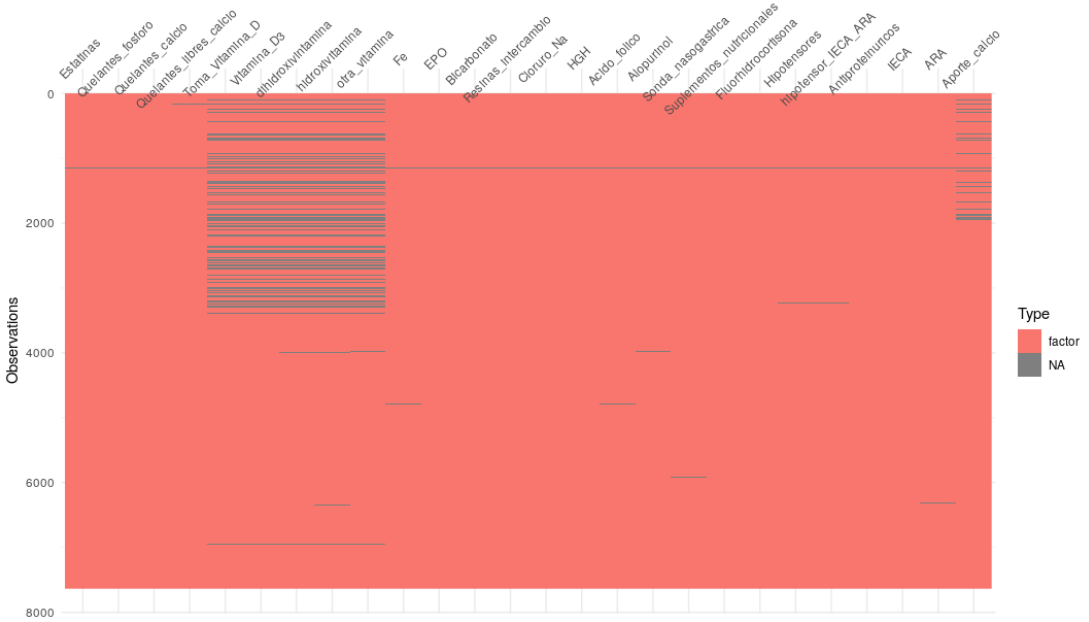


Figura 3: Tipo de variables de tratamientos consideradas en el análisis. NA significa valores perdidos en las variables.

1.2 Análisis de valores perdidos

Este análisis se realizó con el objetivo de conocer el porcentaje de valores perdidos que existe en la base de datos, sirviendo además como métrica de calidad. Es de destacar que, en la mayoría de los casos, la efectividad en el análisis de datos depende en gran medida de la calidad que tengan los mismos. Aunque se intentará trabajar con modelos predictivos que sean robustos ante datos perdidos, es importante recordar que muchos de los modelos de aprendizaje automático dependen de que los valores perdidos hayan sido imputados previamente.

1.2.1 Valores perdidos en variables demográficas

Como podemos observar en la Figura 4, las variables Semanas_gestacion, Peso_nacimiento y Fecha_inicio_ERC tienen niveles de valores perdidos que van desde 25% hasta 35%. La Figura 5 muestra el número de veces en que los valores perdidos de las variables ocurren simultáneamente. A partir de esta figura, se confirma que los valores perdidos en las tres variables mencionadas anteriormente aparecen frecuentemente al mismo tiempo.

La Figura 6 muestra que más de 400 pacientes tienen más del 20% de sus variables demográficas con valores perdidos, mientras que más de 1000 pacientes no tienen valores perdidos en sus variables demográficas. Por último, la Figura 7 muestra que en total existen aproximadamente 2000 valores perdidos en variables demográficas.

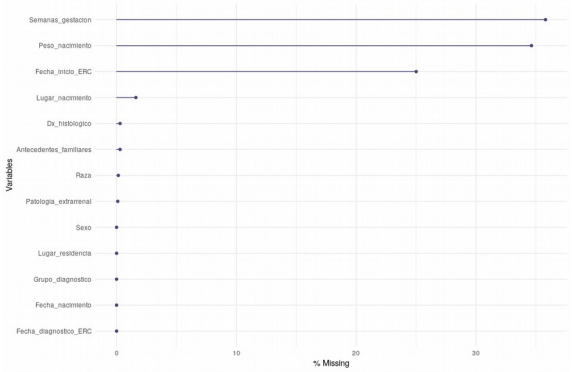


Figura 4: Porciento de valores perdidos en variables demográficas.

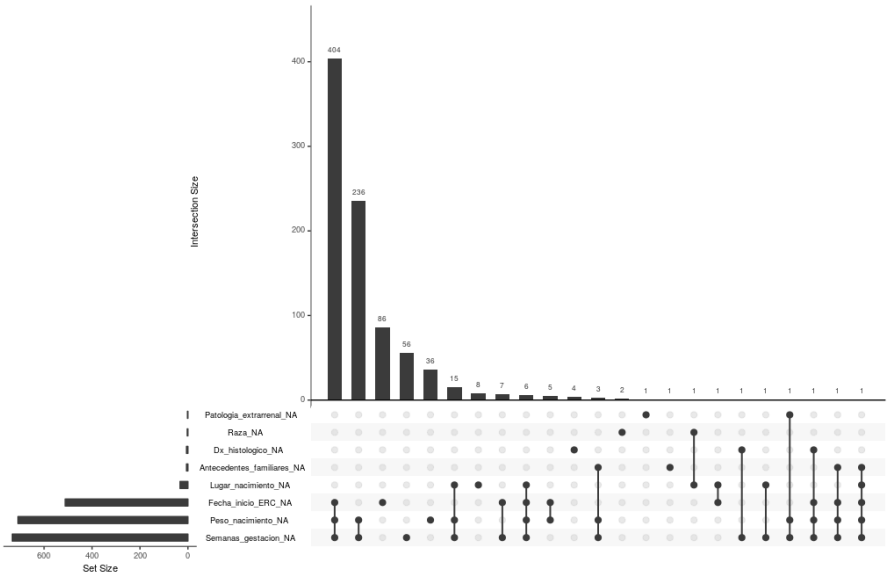


Figura 5: Combinación de valores perdidos en variables demográficas

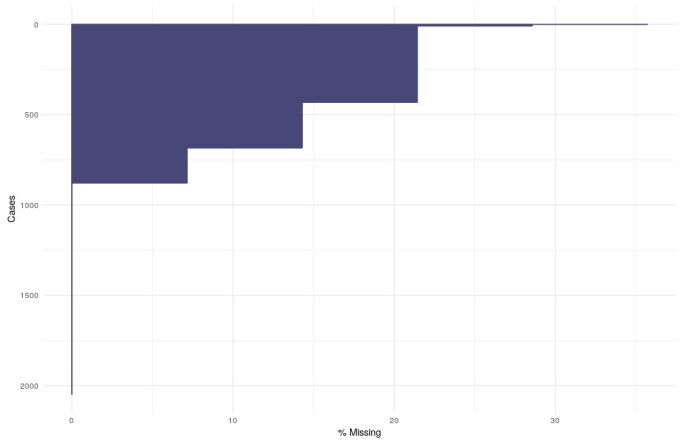


Figura 6: Porciento de valores perdidos en variables demográficas por pacientes.

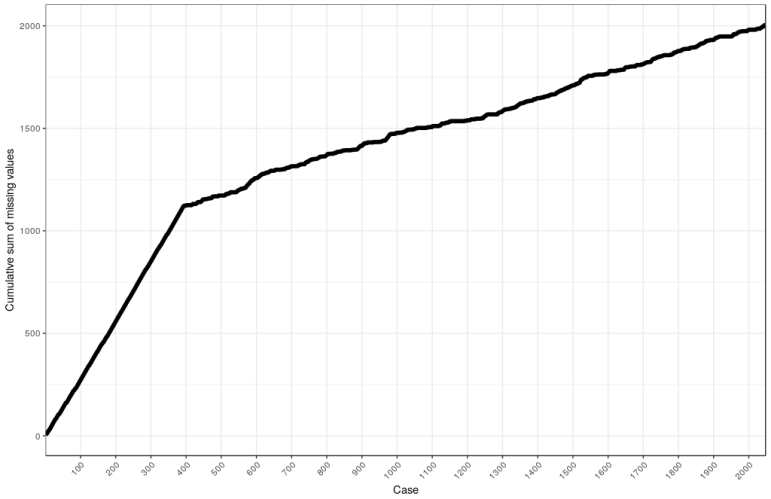


Figura 7: Suma acomulativa de valores perdidos en variables demográficas.

1.2.2 Valores perdidos en variables de visitas

La Figura 8 muestra los porcientos de valores perdidos en cada una de las variables relacionadas con las visitas. Como se puede observar, la variable Perímetro_cefálico tiene un enorme porciento de sus valores perdidos, lo cual tiene lógica ya que la medición de esta variable es de relevancia solamente en niños muy pequeños. Sin embargo, también podemos observar que la variable Indice_transferrina y 25-OH tienen niveles moderados de valores perdidos que superan al 30% del total de visitas.

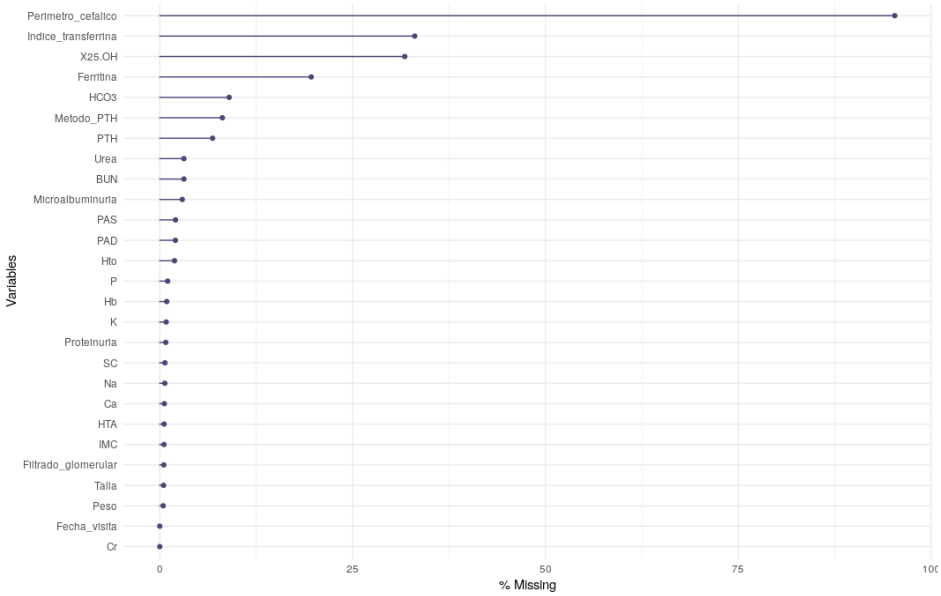


Figura 8: Porciento de valores perdidos en variables de visitas.

La Figura 9 muestra el número de veces en que los valores perdidos de las variables de visitas ocurren simultáneamente. Como era de esperar las combinaciones de valores perdidos de mayor frecuencia están presente entre las variables Perímetro_cefálico, Indice_transferrina, 25-OH y Ferritina, que son a su vez las variables con mayor cantidad de valores perdidos.

A partir de la Figura 10 podemos apreciar que más de 2000 registros de visitas tienen valores perdidos en aproximadamente el 10% del total de variables, y la Figura 11 muestra que existen aproximadamente 17,000 valores perdidos en las variables de visitas.

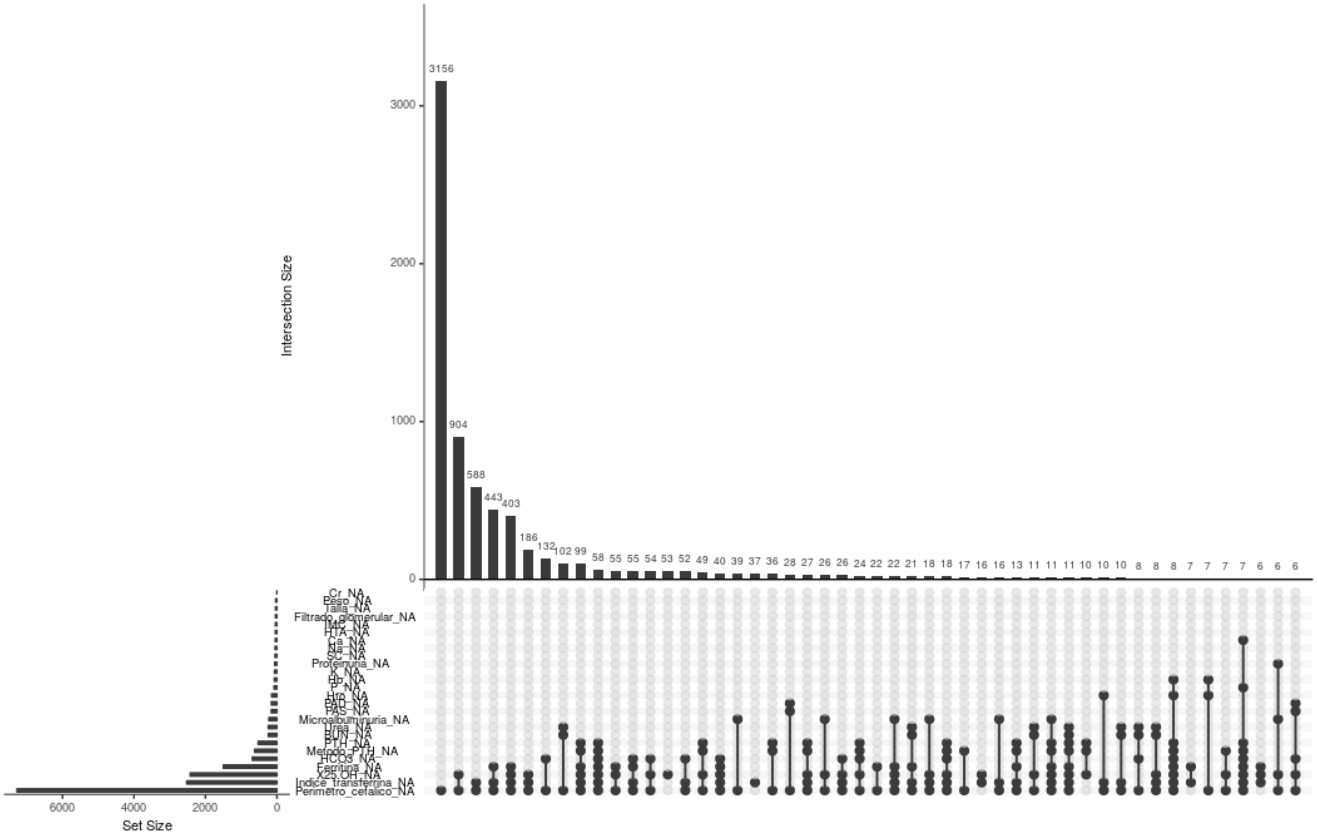


Figura 9: Combinación de valores perdidos en variables de visitas

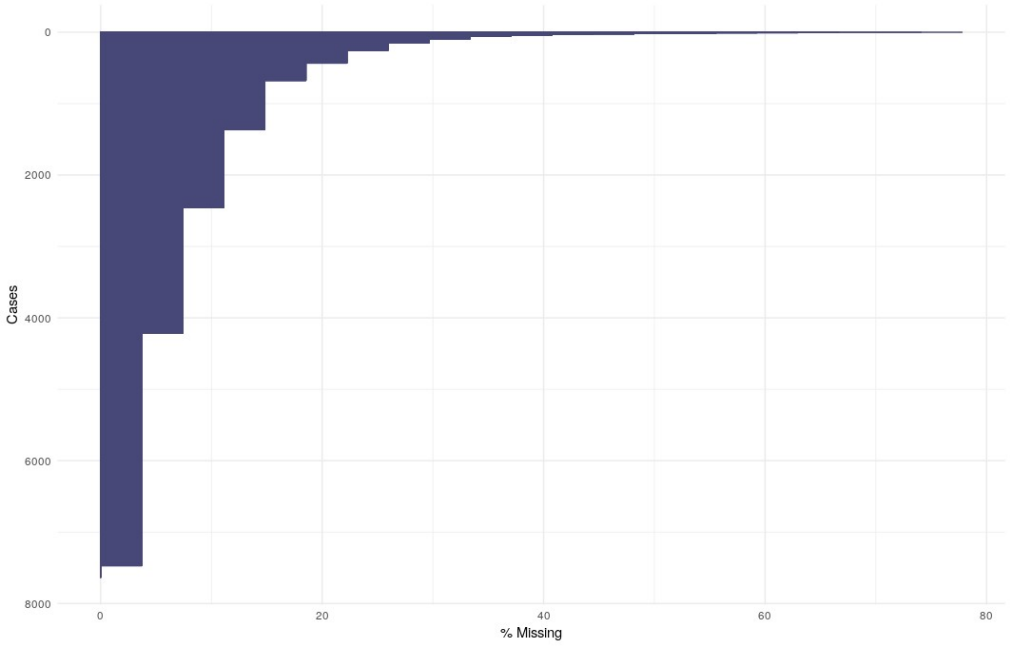


Figura 10: Porciento de valores perdidos en variables de visitas por pacientes.

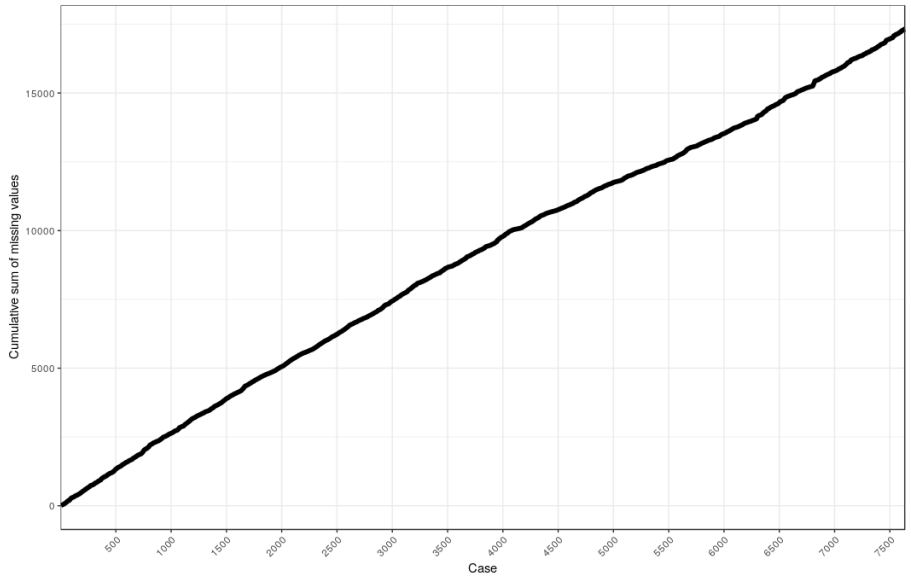


Figura 11: Suma acomulativa de valores perdidos en variables de visitas.

1.2.3 Valores perdidos en variables de tratamientos

La figura 12 muestra que las variables Toma_vitamina_D, Vitamina_D3, dihidroxivitamina, hidroxivitamina y otra_vitamina tienen aproximadamente el 14% de sus valores perdidos, mientras que la variable Aporte_calcio tiene aproximadamente el 5% de valores perdidos.

La Figura 13 muestra el número de veces en que los valores perdidos de las variables de tratamientos ocurren simultáneamente. Como era de esperar las combinaciones de valores perdidos de mayor frecuencia están presente entre las cinco variables anteriormente mencionadas. Existen 12 registros de tratamientos que tienen todas sus variables perdidas.

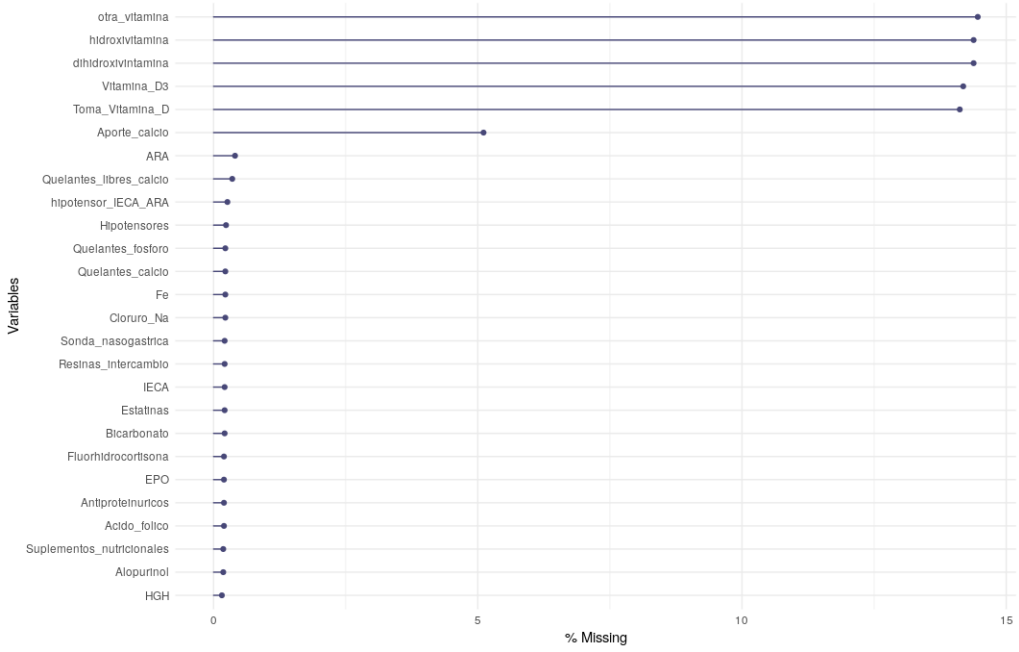


Figura 12: Porciento de valores perdidos en variables de tratamientos.

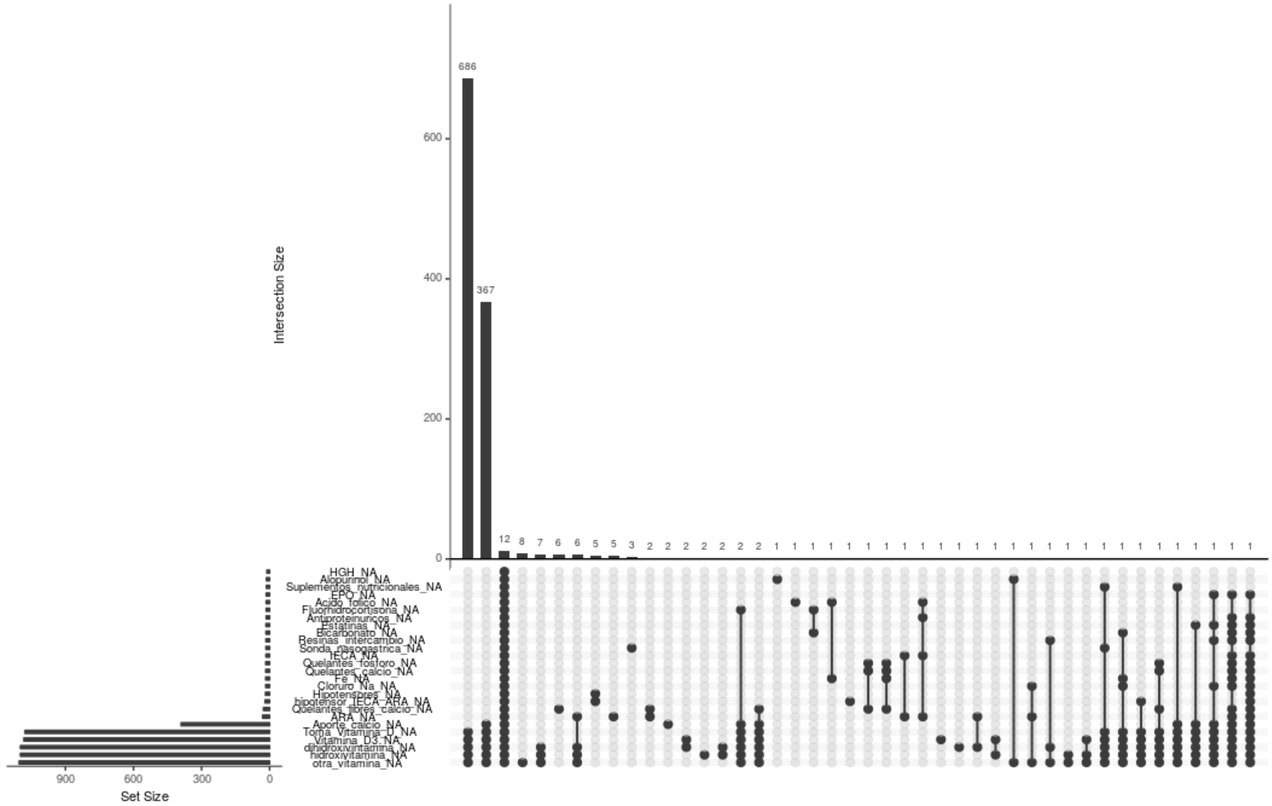


Figura 13: Combinación de valores perdidos en variables de tratamientos.

A partir de la Figura 14 podemos apreciar que 1000 registros de tratamientos tienen valores perdidos en alrededor del 20% del total de variables de tratamientos, y la Figura 15 muestra que existen aproximadamente un total de más de 6,000 valores perdidos en las variables de tratamientos.

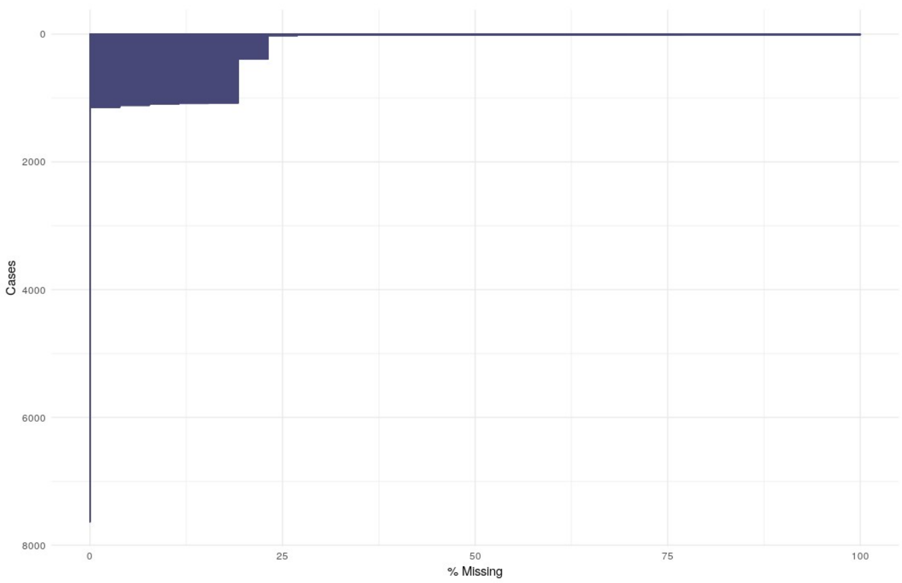


Figura 14: Porciento de valores perdidos en variables de tratmientos por pacientes

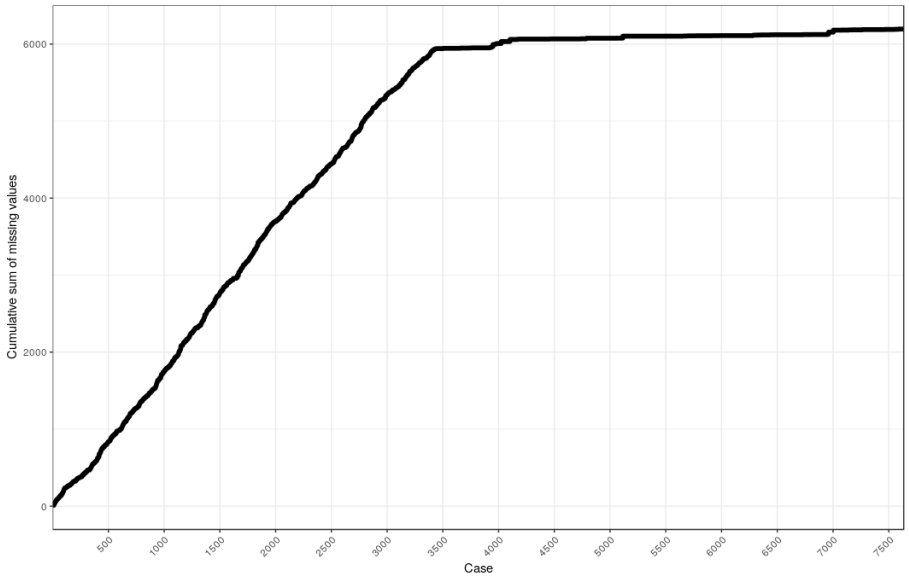


Figura 15: Suma acomulativa de valores perdidos en variables de tratamientos.

1.3.4 Estimación de valores perdidos

Para la imputación de valores perdidos inicialmente consideramos usar modelos de *Joint Modeling* o *Multiple Imputation by Chained Equations* para datos longitudinales, ya que estos métodos son los que más garantías ofrecen de estimar correctamente valores perdidos en este tipo de problemas [End16]. Sin embargo, la aplicación de estos modelos en los datos disponibles no fue posible, ya que para estimar los valores perdidos de una variable, estos modelos requieren que las variables predictoras no tengan valores perdidos, y en nuestro dataset esta condición no se cumple en ningún caso.

Debido a esta situación se procedió a imputar los valores perdidos de cada paciente mediante métodos clásicos para datos longitudinales. Se utilizaron los cinco métodos siguientes [Eng03]:

1- *Previous row* (PR): Substituye los valores perdidos por el promedio, mediana o moda de los valores previos conocidos del paciente.

2- *Before and after* (BA): Substituye los valores perdidos por el promedio, mediana o moda de los valores conocidos del paciente; a diferencia de PR no solo se consideran los valores previos antes de ocurrir el valor perdido, sino que se consediran además los valores posteriores.

3- Last observation carried foward (LOCF): Se substituye el valor perdido por el último valor conocido del paciente antes del valor perdido.

4- Next observation carried backward (NOCB): Se substituye el valor perdido por el siguiente valor conocido del paciente.

5- Last and next known values (LN): Promedio, mediana o moda del último valor conocido antes del valor perdido y el siguiente valor conocido después del valor perdido.

Es de destacar que los cinco métodos antes expuestos solo tienen sentido en variables longitudinales (que en nuestro caso son las variables asociadas a las visitas y tratamientos de los pacientes), y no sobre la variable globales (que en nuestro caso son las variables demográficas de los pacientes). En

el caso de las variables demográficas se puede utilizar cualquier método clásico de imputación, por ejemplo la media, mediana o moda.

Además, es importante aclarar que los métodos de imputación solo se utilizaron sobre variables independientes. Los valores perdidos de las variables dependientes SC, IMC, BUN y Filtrado_glomerular se calcularon mediantes sus correspondientes fórmulas una vez que las variables independientes relacionadas fueron estimadas.

Como nuestro dataset tiene tanto variables numéricas como nominales, además de tener tanto variables globales como longitudinales, consideramos las siguientes combinaciones para la imputación de valores perdidos: PR-mean, PR-median, BA-mean, BA-median, LOCF-mean, LOCF-median, NOCB-mean, NOCB-median, LN-mean, LN-median. Esto da como resultado que a partir del conjunto de datos original se obtengan 10 conjuntos de datos diferentes, cada uno de ellos se obtiene a partir de la aplicación del método de imputación correspondiente.

1.3 Eliminación de valores y registros atípicos

En la etapa de preprocesamiento de datos además se han eliminado los valores que estaban fuera de los rangos permitidos, y que pudieron haber sido insertados producto a error humano. El proceso de detección de outliers en los datos es de gran importancia, ya que este tipo de valores puede afectar significativamente el rendimiento predictivo de los modelos y el análisis de los datos en general.

En esta etapa, se han realizado las siguientes operaciones siguiendo el criterio de los expertos:

1- Se eliminaron 6 valores atípicos en las variables demográfica Peso_nacimiento. Se consideró como outlier todo valor que estuviera fuera del rango [0.5, 5] kg.

2- Se eliminaron 11 valores atípicos en la variable demográfica Semanas_gestación. Se consideró como outlier todo valor fuera del rango [25, 42] semanas de gestación.

En cuanto a los valores atípicos en las variables de visitas, se realizaron las siguientes acciones:

3- Se eliminaron 17 valores atípicos en la variable Peso. Se consideró como outlier todo valor fuera del rango [3, 110] kg.

4- Se eliminaron 12 valores atípicos en la variable Talla. Se consideró como outlier todo valor fuera del rango [45, 200] cm.

5- Se eliminaron 9 valores atípicos en la variable Perímetro_cefálico. Se consideró como outlier todo valor fuera del rango [30, 60] cm.

6- Se eliminaron 9 valores atípicos en la variable PAS. Se consideró como outlier todo valor fuera del rango [60, 200] mm Hg.

7- Se eliminaron 8 valores atípicos en la variable PAD. Se consideró como outlier todo valor fuera del rango [30, 120] mm Hg.

8- Se eliminaron 14 valores atípicos en la variable Urea. Se consideró como outlier todo valor fuera del rango [10, 321].

9- Se eliminaron 3 valores atípicos en la variable Na. Se consideró como outlier todo valor fuera del rango [100, 160] mEq/l.

10- Se eliminaron 5 valores atípicos en la variable K. Se consideró como outlier todo valor fuera del rango [2, 7] mEq/l.

11- Se eliminaron 5 valores atípicos en la variable HCO3. Se consideró como outlier todo valor fuera del rango [5, 40] mMol/l.

12- Se eliminaron 52 valores atípicos en la variable Hb. Se consideró como outlier todo valor fuera del rango [5, 20] g/dl.

13- Se eliminaron 46 valores atípicos en la variable Hto. Se consideró como outlier todo valor fuera del rango [15, 60] %.

14- Se eliminaron 220 (aprox. el 2.8% del total) valores atípicos en la variable Ferritina. Se consideró como outlier todo valor fuera del rango [10, 400] ng/ml.

15- Se eliminaron 487 (aprox. el 6% del total) valores atípicos en la variable Indice_transferina. Se consideró como outlier todo valor fuera del rango [5, 40] %.

16- Se eliminaron 7 valores atípicos en la variable Calcio. Se consideró como outlier todo valor fuera del rango [6, 12] mg/dl.

17 - Se eliminaron 7 valores atípicos en la variable P. Se consideró como outlier todo valor fuera del rango [2, 12] ng/dl.

18 - Se eliminaron 157 (aprox. el 2%) valores atípicos en la variable PTH. Se consideró como outlier todo valor fuera del rango [10, 1000] pg/ml.

19 - Se eliminaron 17 valores atípicos en la variable 25-OH. Se consideró como outlier todo valor menor de 5 ng/ml.

20- Se eliminaron 8 valores atípicos en la variable Cr. Se consideró como outlier todo valor menor fuera del rango [0.3, 20].

En el caso de la variable Metodo_PTH, la mayoría de los registros fueron calculados por el método Intacta, habiendo solo 44 valores calculados por BioPTH de un total de 7015 valores. Como los valores de PTH varían mucho de un método a otro, y no existe un método exacto que permita llevar el valor de PTH por BioPTH a valor Intacta, se decidió eliminar los valores de PTH que fueron calculados por BioPTH. Además, en lo adelante se asume que los valores perdidos de PTH también fueron calculados por el método Intacta, al ser esta la moda en la población.

En el caso de las variables SC, IMC, BUN y Filtrado_glomerular, no se eliminaron valores atípicos directamente sobre ellas, ya que el valor de estas variables es calculado a partir de Talla, Peso, Cr y Urea.

Por otra parte, los expertos establecieron eliminar del análisis los registros completos de visitas donde el Filtrado_glomerular fuera mayor o igual de 90, ya que en estos casos no se cumple con los valores regulados y los pacientes causan baja del sistema por recuperación de las funciones renales.

Finalmente, se eliminaron del análisis, para la construcción del modelo predictivo, aquellos pacientes que tienen solo una visita, o que quedaron con una visita tras eliminar los registros atípicos. Esto se debe a que el estudio a realizar es longitudinal y una persona con una sola visita

registrada no aportaría información al modelo para la predicción de valores futuros de Filtrado_glomerular. Tras realizar esta acción, para la construcción del modelo quedaron 2004 pacientes con un total de 6671 visitas; la cantidad máxima y mínima de visitas por pacientes son dos y 13, respectivamente.

1.4 Selección y construcción de variables

En esta etapa de preprocesamiento de los datos, se detectaron y eliminaron variables que son redundantes, así como se detectaron aquellas variables no aportan ningún tipo de información. A continuación se listan las acciones realizadas:

1- Teniendo en cuenta la explicación mostrada en la Sección 1.3 respecto a los valores PTH, la variable Metodo_PTH carece de sentido, ya que tiene solo un posible valor, y por lo tanto, esta variable fue eliminada del análisis.

2- En cuanto a las variables relacionadas con Microalbuminuria y Proteinuria, se utilizaron las variables categóricas (es decir las que tienen los valores Normal, Patológica y No realizada) en vez de los valores numéricos; las variables numéricas asociadas a Microalbuminuria y Proteinuria tienen un considerable número de valores perdidos.

3- La variable Perímetro_cefálico se eliminó debido a que tiene demasiados valores perdidos, y su imputación conllevaría un sesgo significativo en el análisis de datos.

4- Las variables IECA y ARA dependen del valor de la variable Antiproteinuricos, por lo que esta última variable se ha eliminado al no aportar nueva información.

5- La variable Quelantes_fosforo se eliminó ya que ésta determina a las variables Quelantes_calcio y Quelantes_libre_calcio.

6- Las variables Vitamina_D3, hidroxivitamina, dihidroxivintamina y otra_vitamina dependen de la variable Toma_Vitamina_D, por lo que esta última variable se eliminó.

7- La variable Hipotensores determina la variable hipotensor_IECA_ARA, por lo que la primera se ha eliminado.

En cuanto a la construcción de nuevas variables, se realizaron las siguientes acciones:

1- Se creó la variable demográfica Edad_diagnostico a partir de las variables Fecha_nacimiento y Fecha_diagnostico_ERC. Esta variable no tiene valores perdidos ya que ninguna de las dos originales tienen valores perdidos.

2- Se creó la variable de visita Tiempo_evolucion a partir de las fechas de visitas y la Fecha_diagnostico_ERC. A partir de esta variable podemos analizar el tiempo que transcurre de una visita a otra, así como el tiempo de evolución de la enfermedad.

3- Se creó la variable de visita Edad_visita a partir de las fechas de visitas y la fecha de nacimiento. A partir de esta variable podemos tomar en cuenta la información que puede aportar la edad de los pacientes a lo largo del tiempo.

Luego de la creación de las variables antes mencionadas, todas las variables de tipo fecha fueron eliminadas del análisis.

2. Construcción del modelo predictivo

Una vez finalizada la etapa de preprocesamiento de los datos, la siguiente tarea se enfocó en modelar y estimar la variable Filtrado_glomerular. Esta es la variable de mayor interés para los especialistas, ya que a partir de ella se evalúa si la enfermedad renal del paciente mejora o empeora, y por consiguiente es el factor que determina si el paciente sigue dado de alta o causa baja del sistema.

El objetivo en esta etapa fue crear un modelo h: $\mathbf{X} \rightarrow \mathbf{Y}$, que a partir del registro de visitas **X** de un paciente, sea capas de predecir los futuros valores de Filtrado_glomerular en ese paciente (**Y**). Este modelo serviría como herramienta de pronóstico para los especialistas. Es de destacar que el modelo diseñado debe tener en cuenta que cada paciente presenta datos agrupados en el tiempo, donde las visitas de un paciente son dependientes entre sí, y son independientes de las visitas de otros pacientes. Esta características de los datos disponibles no permite la aplicación directa de modelos predictivos clásicos. A continuación detallamos el modelo diseñado.

2.1 Diseño del modelo

Los datos longitudinales disponibles en el conjunto de datos pueden ser analizados como datos secuenciales multi-variantes. En otras palabras, un paciente tiene un registro que está compuesto por *n* visitas-tratamientos, y cada uno de estos registros representa la visita-tratamiento del paciente en el instance de tiempo *t*. Además, se considera multi-variante porque cada registro visita-tratamiento de un paciente está descrito por *m* características; en nuestro caso este vector estaría compuesto por todas las variables de visitas y tratamientos. Es de destacar que, en el caso de las variables demográficas, al ser estas variables globales, la aproximación más común es repetir sus valores tantas veces registros de visitas-tratamientos existen para un paciente.

En consecuencia, digamos que el registro de un paciente en el instance de tiempo *t* está descrito por *m* variables (la suma de variables demográficas, de visitas y tratamientos), por lo tanto el conjunto de datos puede ser representado de manera general como un tensor 3D con forma (*nsamples*, *timesteps*, *m*), donde *nsamples* denota la cantidad total de pacientes en la base de datos, *timesteps* representa la cantidad máxima de visitas que tiene un paciente, y *m* la cantidad de variables que describen al paciente en cada instance de tiempo *t.* El objetivo fue entonces construir un modelo predictivo, que a partir de los datos de un paciente, sea capaz de predecir los valores futuros de Filtrado_glomerular de cualquier paciente.

Las condiciones del problema antes descrito corresponden a un problema de aprendizaje *sequence-to-sequence*, donde tenemos un paciente con datos longitudinales multi-variantes y se quiere predecir una secuencia de valores que corresponden a los valores correspondientes de Filtrado_glomerular que tendrá el paciente. A esta dificultad habría que añadirle que puede existir una cantidad diferentes de visitas por pacientes, y además que la cantidad de valores de Filtrado_glomerular a predecir en el tiempo puede ser variable. De aquí que el problema a resolver sea un *canonical sequence-to-sequence*, ya que tanto las secuencias de entradas como las de salidas por cada paciente pueden tener longitud variable.

Para resolver el problema mencionado, en este trabajo hemos hecho uso de la técnica de aprendizaje profundo Long Short-Term Memory (LSTM), la cual permite crear un modelo predictivo a partir de datos secuenciales como los descritos anteriormente. La Figura 16 muestra un modelo básico LSTM; este es un tipo de red neuronal recurrente que tiene como entrada un vector diferente $\mathbf{X_t}$ en cada instante de tiempo *t.* La Figura 17 muestra cómo está compuesta cada unidad de LSTM internamente.

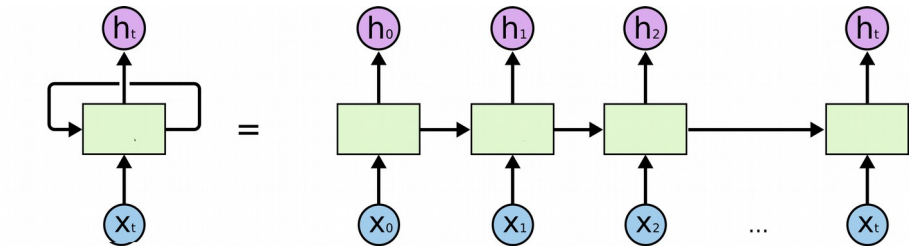
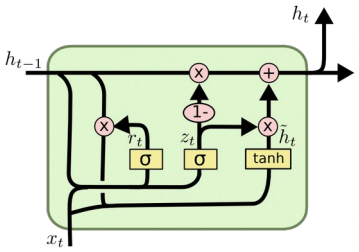


Figura 16: Modelo LSTM estándar (imagen tomada de [Und15]).



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figura 17: Unidad LSTM *(imagen tomada de [Und15])*.

Siguiendo la idea de arquitectura LSTM, la Figura 18 muestra de manera general el modelo diseñado para predecir los valores de Filtrado_glomerular. El modelo diseñado sigue un enfoque Encoder-Decoder, el cual es conocido ampliamente en el dominio de traducción de textos.

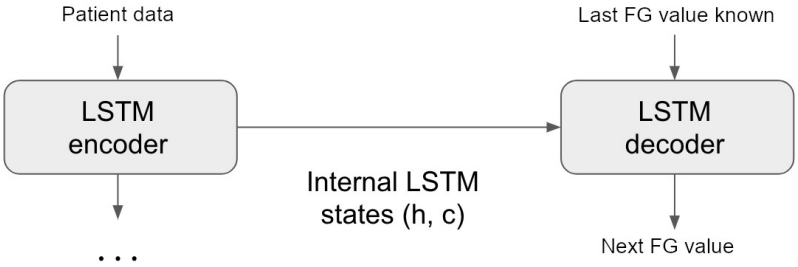


Figura 18: Modelo predictivo diseñado.

El modelo se compone de dos capas LSMT. La primer capa, llamada Encoder recibe como entrada los datos de un paciente en forma de un tensor de tamaño (1, *timesteps*, *m*), donde *timesteps* es igual a la cantidad de visitas que ha tenido el paciente, y *m* es el número total de variables describiendo cada instante de tiempo. Encoder procesa la secuencia de entrada y retorna su estado interno final; en este caso la salida del Encoder se desecha. Este estado interno representa el contexto abstracto, o condiciones, que servirá como punto de partida de la capa siguiente que compone el modelo. La capa LSTM siguiente se denomina Decoder, la cual recibe como entrada el último valor de Filtrado_glomerular conocido y comienza con un estado interno igual al Encoder. Decoder predice los siguientes valores de Filtrado_glomerular. En otra palabras, Decoder aprende a generar el siguiente valor de Filtrado_glomerular en el *timestep* t+1 dado el valor conocido en el *timestep* t y

condicionado por el conocimiento aprendido (contexto) a partir de todas las variables almacenadas a lo largo de las visitas.

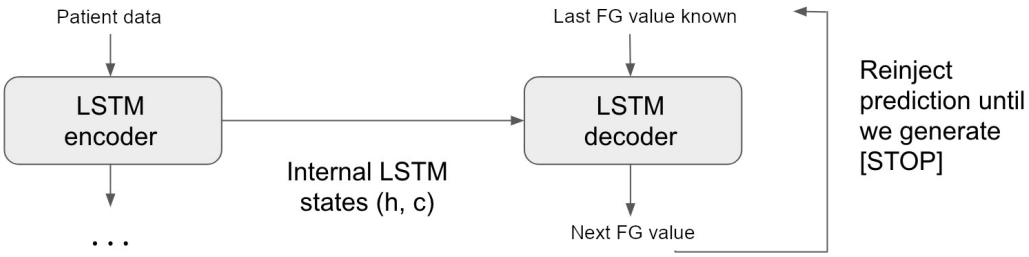


Figura 19: Esquema de inferencia utilizado para predecir secuencias de valores de Filtrado_glomerular.

La figura 19 representa el esquema de inferencia que sigue el modelo para predecir secuencias de valores de Filtrado_glomerular a partir de los datos de un paciente nunca antes visto. Los pasos que se ejecutan son los siguientes:

1- Codificar la secuencia de entrada (secuencia de vectores de variables asociados a las visitas del paciente) en un vector de estado.

2- Inicializar el Decoder con el estado interno del Encoder y el último valor conocido de Filtrado_glomerular del paciente.

3- Predecir un valor de Filtrado_glomerular e insertarlo como entrada nuevamente al Decoder para producir el siguiente valor. Este proceso se realizará iterativamente hasta alcanzar una cantidad de valores de Filtrado_glomerular deseados. Es de destacar que a medida que se predicen más valores futuros de Filtrado_glomerular se estaría acarreando un mayor error en la estimación.

Para entrenar el modelo diseñado, establecemos una función de error que permitirá ir ajustando gradualmente los valores de pesos de las neuronas mediante el algoritmo *back-propagation*. En este trabajo, hemos utilizado la función de error *log-cosh*, la cual ha demostrado ser adecuada para la optimización de modelos para problemas de regresión. Dado un paciente *i* con el último valor de Filtrado_glomerular registrado igual a $y_{it,}$ el error cometido en la predicción es igual a:

$$\mathbf{L_i}(z_{it}\ \mathbf{,}\ y_{it})\mathbf{=log(cosh(}z_{it} - y_{it}\mathbf{))},$$

siendo z_{it} el valor de Filtrado_glomerular predicho por el modelo para el paciente *i*. La función **L** es convexa, más suave en el punto de mínimo (esto hace que disminuya el gradiente) y menos sensitiva a outliers que otras funciones de pérdida populares como el error cuadrático. Además, **L** es diferenciable dos veces, y no necesita ningún ajuste de parámetros [Neu98]. Por lo tanto, la función de costo de la red neuronal se define como:

$$\mathbf{J(W) = 1/nsamples * sum(L_i(}z_{it}\ \mathbf{,}\ y_{it}\mathbf{))},$$

la cual promedia los errores cometidos en la predicción de cada paciente del conjunto de entrenamiento, y **W** representa el conjunto de pesos de la red. El objetivo entonces es encontrar el conjunto de pesos **W** que miniza **J**. Para encontrar dicho conjunto de pesos, hemos utilizado el algoritmo de optimización llamado *Adaptive Moment Estimation* (ADAM) [Kin14]. ADAM ha mostrado ser realmente efectivo en la optimización de una gran variedad de redes neuronales, ya que calcula el *exponential moving averages* del gradiente y su cuadrado. En cada iteración, ADAM actualiza los pesos de la red basado en el gradiente de la función **J,** el cual se calcula sobre un subconjunto de pacientes que es seleccionado aleatoriamente. El subconjunto de pacientes

seleccionado aleatoriamente se denomina *mini-batch,* y se considera que un *epoch* de la etapa de entrenamiento del modelo ha culminado cuando se ha muestreado completamente el conjunto de datos. Por lo tanto, el modelo se entrena durante *nepochs,* lo cual implica que el conjunto de datos es pasado completamente por la red *nepochs* veces.

2.2 Evaluación del modelo

En esta sección se describe la evaluación del modelo, con el objetivo de saber si realmente el modelo es efectivo para predecir los valores futuros de Filtrado_glomerular.

2.2.1 Configuración del experimento

Antes de poder utilizar el modelo sobre el conjunto de datos disponible, fue necesario convertir los datos a un formato entendible por la red neuronal; esto requiere que todas las variables que componen el conjunto de datos sean numéricas. Todas las variables categóricas del conjunto de datos fueron transformadas a variables *dummy* (conjunto de variables binarias), siguiendo la premisa de que una variable nominal con *n* posibles estados puede ser representada por *n* variables binarias.

En cada experimento, se utilizó una validación de tipo Walk Forward Cross-Validation (WFCV), que es el tipo de validación más adecuada para nuestro problema. Un proceso de WFCV realiza los siguientes pasos: (I) se entrena el modelo con los datos conocidos hasta el timestep t y se predice el valor de Filtrado_glomerular en el timestep $t+1$; (II) se calcula el error cometido en la predicción; (III) se comienza de nuevo en el paso (I) haciendo $t=t+1$. Este proceso iterativo de validación comienza con $t=1$. Además, en cada partición de la validación se realiza un proceso de normalización de las variables numéricas que tiene el conjunto de datos, evitando de esta manera que variables con escalas diferentes tengan un mayor efecto en la predicción; se utilizó el método Z-score para la normalización, donde a cada valor se le resta la media de la variable, y se divide por la desviación estándar.

Por otra parte, se llevó a cabo un proceso de ajuste de los hiperparámetros principales que controlan la red diseñada. En este caso, se utilizó un proceso de optimización bayesiana [Snoe12] para encontrar la mejor configuración de parámetros a partir de las siguientes posibilidades:

1- Métodos de imputación: Se consideraron las 10 versiones de métodos de imputación que se comentarin anteriormente en la sección 1.3.4.
2- La cantidad de unidades ocultas (dimensión latente a generar) utilizadas en cada capa LSTM fueron 32, 64, 128 y 256.
3- El modelo se entrenó con un número de *nepochs* igual a 30, 50 y 100.
4- El tamaño de batch fue de 1, 4, 8, 16 y 32.

Finalmente, para la implementación del modelopropuesto se utilizó el framework tensorflow [Aba15] mediante su API en lenguaje Python.

2.2.2 Resultados

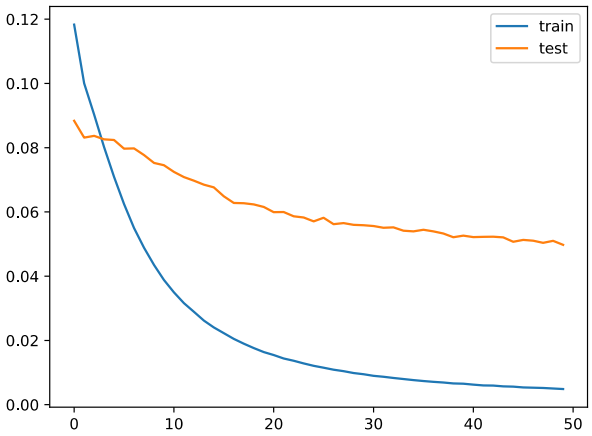


Figura 20: Errores de estimación promedio sobre los conjuntos de entrenamiento y prueba.

La configuración de los hiper-parámetros realizada mediante optimización bayesiana indicó que el modelo logra disminuir en promedio el error tanto en el conjunto de entrenamiento (*training*) como en el conjunto de prueba (*test*) a medida que pasan los *epochs*. En los 10 casos, el modelo obtuvo *bias* y *variance* bajos, lo cual indica que el modelo no solo logra ajustar los pesos correctamente sobre los datos de entrenamiento, sino además que tiene un rendimiento aceptable en el conjunto de prueba, indicando que el modelo puede predecir con un error aceptable los futuros valores de Filtrado_glomerular.

La configuración que mejor resultados promedios arrojó se obtuvo cuando se utilizó como método de imputación BA-mean, 64 unidades ocultas, 50 epochs, y tamaño de batch igual a 1. La figura 20 muestra el gráfico de error de esta configuración a medida que incrementaban los epochs.

2.2.3 Estimación de la importancia de las variables en la predicción

Para estimar la importancia global que tiene cada variable del conjunto de datos en la predicción del Filtrado_glomerular, en este trabajo seguimos el procedimiento *permutation importance.* Este método permite calcular la importancia de las variables en cualquier modelo predictivo, midiendo cuánto cambia el rendimiento del modelo si una variable deja de estar disponible.

La solución más directa para ejecutar este procedimiento es eliminar la variable del conjunto de datos y reintrenar el modelo. Sin embargo, este proceso puede ser computacionalmente bastante costoso. En vez de eliminar una variable, también es posible substituirla con ruido aleatorio, de esta manera la variable no contendrá información útil y nos permite a su vez conocer el impacto que tiene en la calidad de la predicción del modelo. Para que funcione bien esta última solución, es de destacar que el ruido tiene que ser muestreado a partir de la misma distribución original de la variable. La manera más simple de hacer este procedimiento es desordenar los valores de una variable, de ahí el nombre de *permutation importance*. Como en nuestro caso los datos están relacionados a las visitas de los pacientes, lo que hacemos es permutar los valores locales de una variable en todos los pacientes y finalmente calculamos cuánto se afecta el rendimiento del modelo.

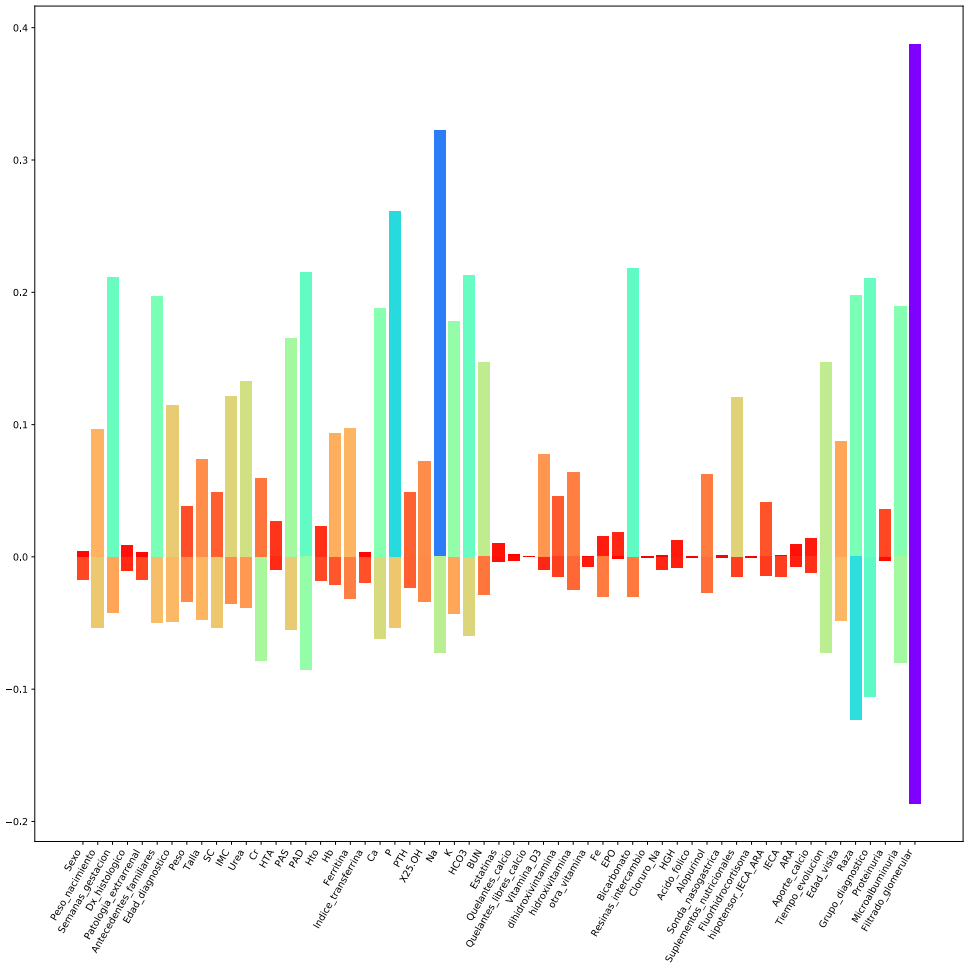


Figure 21: Importancia de las variables en la predicción.

La figura 21 muestra la importancia que tienen en la predicción de futuros valores de Filtrado_glomerular cada una de las variables que participan en el modelo. Esta gráfica se compone de valores positivos y negativos para cada variable, indicando si la variable tiene un efecto de sobreestimación o estimación por debajo de los valores reales de Filtrado_glomerular.

Como es lógico, los valores históricos de Filtrado_glomerular tiene la mayor importancia para predecir los valores futuros, y además se espera que los valores correspondientes a visitas más recientes del paciente tengan un mayor impacto en la predicción. Los valores históricos de Filtrado_glomerular tienen un efecto con una tendencia un poco mayor a sobreestimar los valores futuros que a predecir valores más bajos que los reales.

Obviando la variable Filtrado_glomerular, al introducir ruido en las variables relacionadas con visitas y tratamientos, las siguientes variables fueron las que más afectaron el rendimiento del modelo, lo cual muestra su importancia en la predicción (sumando su efecto de sobreestimación y estimación por debajo): Na, P, PAD, HCO3, Microalbuminuria, Ca, Bicarbonato, K, PAS, Tiempo_evolución, BUN, Urea, IMC, Cr, Suplementos_nutricionales, Ferritina, Talla, Hb, X25-OH, Peso, PTH, etc.

En cuanto a las variables basales demográficas, las dos variables más importantes (es decir, que al introducirles ruido empeoran significativamente el modelo) fueron Grupo_diagnóstico y Raza. En el caso Peso_nacimiento y Semanas_gestación hay que considerar que estas dos variables tenían más del 30% de valores perdidos en el conjunto de datos inicial, por lo que la importancia calculada por el método empleado puede estar sobre estimada.

Al observar la gráfica podemos ver que en su mayoría las covariables que utiliza el modelo tienen un mayor efecto en la sobreestimación del Filtrado_glomerular que en producir valores más bajos que los reales. Aunque es destacar que existen casos particulares de covariables que al introducir ruido tienen un mayor efecto en la producción de valores promedios por debajo de los reales, por ejemplo: Sexo, Dx_histológico, Patología_extrarrenal, SC, Cr, Indice_transferrina y Fe.

Referencias

[Aba15] M. Abadi et al., TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from https://www.tensorflow.org, 2015.

[End16] C. K. Enders, S. A. Mistler, & B. T. Keller. Multilevel multiple imputation: A review and evaluation of joint modeling and chained equations imputation. *Psychological Methods*, *21*(2), 222, 2016.

[Eng03] J. M. Engels & P. Diehr. Imputation of missing longitudinal data: a comparison of methods. *Journal of clinical epidemiology*, *56*(10), 968-976, 2003.

[Glo10] X. Glorot & Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics,* pp. 249-256, 2010.

[Kin14] D. P. Kingma and J. Ba. ADAM: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.

[Neu98] R. Neuneier & H. Zimmermann. How to train neural networks. In *Neural networks: tricks of the trade* (pp. 373-423). Springer, Berlin, Heidelberg, 1998.

[Snoe12] J. Snoek et al. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems,* pp. 2951-2959, 2012.

[Und15] Understanding LSTM Networks. http://http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 27/08/2015.