Telefónica | EDUCACIÓN DIGITAL

Módulo 2:

Nuestros datos. ¿Están preparados para el Machine Learning?

Módulo 2

Machine Learning: ¿Están nuestros datos preparados?

La realidad es que los datos que se usan en el Machine Learning normalmente proceden de distintos orígenes. Nuestros datos se almacenan en bases de datos, logs de acceso, hojas de cálculo, sistemas CRM, etc. Pongamos un ejemplo. En el caso de tener datos sobre pacientes, algunos de ellos, como la edad y su peso, pueden estar almacenados en su historial, mientras que los resultados de analíticas pueden obtenerse de un registro externo y el diagnóstico o las observaciones pueden ser introducidas en una hoja de cálculo.

Incluso en el caso de que todos los datos disponibles sobre un tema estén almacenados en un solo entorno, como una base de datos, normalmente estarán separados en varias tablas relacionales[1]. Ese proceso de separación, llamado normalización, es una práctica conveniente para optimizar el almacenamiento de los datos y asegurar su mantenimiento, pero no es el adecuado como formato de entrada para los algoritmos de Machine Learning. En este módulo veremos qué estructura necesitamos dar a nuestros datos para que estén preparados para el Machine Learning y qué formatos se soportan.

2.1. Tipos de datos útiles y su formato.

El primer paso para abordar un problema de Machine Learning es, sin duda, hacer una definición clara que nos permita plantear una solución a partir de los datos.

¿Cómo definir un problema de Machine Learning?

Saber qué datos podemos usar para el aprendizaje y cómo prepararlos requiere antes que nada responder a dos preguntas.

¿Cuál es el sujeto del problema a resolver?

Supongamos que queremos saber qué usuarios de un servicio son susceptibles de querer darse de baja en el próximo mes. En este caso, el sujeto será el usuario.

Si en cambio queremos saber qué contratos pueden ser ampliados en el próximo mes, el sujeto de nuestro estudio será el contrato, dado que cada usuario puede tener más de un contrato y cada uno puede ser ampliado o no independientemente.

La segunda pregunta es:

¿Cuáles son las propiedades de ese sujeto que pensamos que pueden influir en la solución?

Analicemos el sector de los seguros de salud. Podemos pensar que los usuarios que amplían sus prestaciones tienen propiedades en común. Es fácil imaginar que para predecir quién puede estar interesado en una ampliación será útil disponer de datos como la edad de la persona, los antecedentes médicos, el número de consultas telefónicas realizadas los últimos meses, el número de visitas a especialistas, las enfermedades conocidas, si hace ejercicio regular, etc. En cambio, no nos interesarán otros datos como su música preferida o el número de libros comprados en los últimos meses.

Además, tendremos que determinar si disponemos de acceso a un histórico de dichos datos. Este histórico es el que proporcionaremos al servicio de Machine Learning para que pueda basar su aprendizaje en ellos.

[1] https://es.wikipedia.org/wiki/Base_de_datos_relacional

Si también queremos que nuestro aprendizaje se pueda repetir cada cierto tiempo para asegurar que el modelo se vaya ajustando a los posibles cambios de los datos, también deberemos asegurarnos de que los datos actualizados estén disponibles periódicamente.

Así pues, tendremos que seleccionar las propiedades posiblemente relevantes, asegurar la disponibilidad de datos históricos de dichas propiedades y la periodicidad en la actualización de dichos datos.

En principio, será interesante añadir todos aquellos datos que puedan tener una relación con lo que queremos averiguar. No obstante, habrá que tener en cuenta que cada nueva propiedad añadida tendrá un coste asociado a su adquisición, almacenaje y transformación. Por eso, cuando el modelo de Machine Learning nos informe de si esa propiedad es útil o no en el aprendizaje podremos replantearnos su uso en función del análisis de coste-beneficio[2].

En el *Módulo 3* veremos que para el caso específico de los problemas de aprendizaje supervisado[3], como la clasificación y la regresión, existe una propiedad de la cual conocemos el valor en algunos casos y queremos predecirla para los demás. Esta propiedad concreta será lo que llamaremos el *campo objetivo* u *objective field*. Los casos en que conocemos su valor serán usados como base de aprendizaje, por lo que es especialmente importante que la definición de esta propiedad sea la correcta. En algunas ocasiones puede pasar que esta propiedad no sea exactamente uno de los campos de nuestro fichero sino que tengamos que obtenerla mediante transformaciones sobre los datos existentes.

Pasar de la definición del problema a la estructura de los datos

Una vez tengamos la respuesta a estas dos preguntas podremos crear la estructura básica que necesitaremos para aplicar el Machine Learning a nuestro problema. Consistirá en una tabla, donde cada fila contendrá la información de uno de nuestros sujetos (una *instancia*) y cada columna los valores de una de sus propiedades, susceptibles de ser útiles en el aprendizaje. Por ejemplo, en el caso de ampliación o no de contratos de servicios, cada fila contendrá toda la información sobre un contrato determinado y cada columna uno de los atributos conocidos, como el número de consultas relacionadas con ese contrato, la categoría del contrato, si ha habido reclamaciones vinculadas, etc.

A menudo, si las propiedades de nuestros datos provienen de diferentes fuentes, será necesario aplicar ciertos procesos previos para lograr esta estructura. En concreto:

De-normalización

Las distintas propiedades de un mismo sujeto pueden estar guardadas en varias tablas relacionales. En ese caso deberemos deshacer el proceso de normalización[4] para unirlas de nuevo en una sola tabla.

En la Figura 2.1 podemos ver un ejemplo típico de la estructura de tablas relacionales que correspondería a una lista de reproducciones de canciones. Vemos que las propiedades de la canción están en una tabla separada, así como las de los álbumes y autores.

[2] https://es.wikipedia.org/wiki/An%C3%A1lisis_de_costo-beneficio
[3] https://es.wikipedia.org/wiki/Aprendizaje_supervisado
[4] https://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_bases_de_datos

Id. usuario	Id. canción	Reproducciones	Primera reproducción	Última reproducción
241	14	4	24/03/2013	2/11/2015
352	52	2	4/11/2016	5/11/2016
631	6	3	3/05/2015	25/01/2016

Id. canción	Título	Duración	Id. álbum	Id. autor
6	Ojalá	3:36	1	4
14	Lucía	3:13	7	2
52	Hello	4:24	9	15

Id. álbum	Título	Canciones	Año
1	Al final de este viaje	10	1978
14	Mediterráneo	10	1971
52	25	8	2015

Id. autor	Nombre
2	Joan Manel Serrat
4	Sílvio Rodríguez
15	Adele

Figura 2.1: Estructura en tablas relacionales de la información sobre reproducciones de canciones. La información está normalizada para evitar redundancias

En la Figura 2.2 Vemos como de-normalizar esa separación y reunir todos los atributos que nos interesan en una sola fila por usuario.

Id. usuario	Título canción	Título álbum	Nombre autor	Reproducciones	Primera reproducción	Última reproducción
241	Lucía	Mediterráneo	Joan Manel Serrat	4	24/03/2013	2/11/2015
352	Hello	25	Adele	2	4/11/2016	5/11/2016
631	Ojalá	Al final de este viaje	Sílvio Rodríguez	3	3/05/2015	25/01/2016

Figura 2.2: Estructura preparada para el aprendizaje de la información sobre reproducciones de canciones. La información ha sido de-normalizada hasta disponer en una sola fila todas las propiedades correspondientes a un solo usuario.

Agregación

Cuando los datos de que disponemos son muy detallados y presentan más de una instancia para cada sujeto de nuestro problema, como por ejemplo en los logs de un servicio web, podemos necesitar agruparlos usando funciones como *contar*, *mínimo*, *máximo*, *media*, etc. El fichero preparado para el aprendizaje contendrá entonces una sola fila por sujeto y sus propiedades serán dichos agregados. En la Figura 2.3 podemos ver un ejemplo de un fichero de datos que requeriría algún tipo de agregación.

Fecha	Id. usuario
24/03/2013	241
5/05/2013	241
18/12/2013	241
3/05/2015	631
2/11/2015	241
17/12/2015	631
25/01/2016	631
4/11/2016	352
5/11/2016	352

Figura 2.3: Lista detallada de reproducciones de canciones. El fichero contiene una fila por cada reproducción.

Usando las funciones de agregación, podremos construir el fichero que vemos en la Figura 2.4. En este fichero hay una fila por cada usuario y la información detallada se ha agrupado para formar propiedades, como el número de reproducciones y las primera y última fecha de reproducción. Esta estructura estará preparada para resolver problemas como predecir el número de reproducciones para un usuario.

Id. usuario	Reproducciones	Primera reproducción	Última reproducción
241	4	24/03/2013	2/11/2015
352	2	4/11/2016	5/11/2016
631	3	3/05/2015	25/01/2016

Figura 2.4: Lista de reproducciones de canciones por usuario. El fichero contiene una fila por usuario y el detalle de las reproducciones ha sido agrupado.

Pivoting

Similarmente a lo que vimos en el caso de la agregación, existen casos en que la información detallada incluye algún campo que nos interesaría usar también como propiedad. La Figura 2.5 muestra un ejemplo de lista de reproducciones de canciones donde el detalle incluye información sobre el soporte usado en la reproducción.

Fecha	Id. usuario	Soporte
24/03/2013	241	Tablet
5/05/2013	241	TV
18/12/2013	241	Tablet
3/05/2015	631	TV
2/11/2015	241	Smartphone
17/12/2015	631	TV
25/01/2016	631	Tablet
4/11/2016	352	Smartphone
5/11/2016	352	Smartphone

Figura 2.5: Lista detallada de reproducciones de canciones con la información sobre el aparato usado como soporte. El fichero contiene una fila por cada reproducción.

Para aprovechar esta información en el aprendizaje, deberemos incluir el detalle de cada tipo de soporte usándolo como una propiedad más en el fichero que usaremos para aprender. El proceso a realizar es transformar grupos de filas en columnas. Así, además de la columna que almacena el total de reproducciones, dispondremos de otras columnas que nos informarán del total de reproducciones por tipo de soporte.

Id. usuario	Reproducciones	Primera reproducción	Última reproducción	Tablet	TV	Smartphone
241	4	24/03/2013	2/11/2015	2	1	1
352	2	4/11/2016	5/11/2016	0	0	2
631	3	3/05/2015	25/01/2016	1	2	0

Figura 2.6: Lista de reproducciones de canciones por usuario. El fichero contiene una fila por usuario y el detalle de las reproducciones ha sido agrupado añadiendo la información del tipo de soporte como nuevas propiedades.

Ventanas temporales

Cuando el problema de Machine Learning puede tener dependencias en la evolución temporal de nuestros datos, necesitaremos que esa información temporal se convierta también en propiedades de nuestro fichero. La manera de convertir una evolución temporal en propiedades de una tabla es crear ventanas temporales. Creamos una ventana temporal cuando resumimos en un período de tiempo el valor de nuestras propiedades. Dependiendo del intervalo temporal en que se muevan los datos, podremos definir ventanas con diferentes periodicidades: semanal, mensual, anual, etc.

Usando como ejemplo el mismo fichero representado en la Figura 2.5, podríamos crear propiedades como el total de reproducciones por año. El resultado final sería el que vemos en la Figura 2.7, donde se ha añadido una nueva propiedad por cada año documentado en nuestro fichero.

Id. usuario	Reproducciones	Tablet	TV	Smartphone	Reproducciones 2013	Reproducciones 2015	Reproducciones 2016
241	4	2	1	1	3	1	0
352	2	0	0	2	0	0	2
631	3	1	2	0	0	2	1

Figura 2.7: Lista de reproducciones de canciones por usuario. El fichero contiene una fila por usuario y el detalle de las reproducciones ha sido agrupado en ventanas temporales de periodicidad anual.

Cualquiera de estos procesos es perfectamente factible con las herramientas de sistema y las que proporcionan las bases de datos actuales. Ejemplos de comandos útiles a tal fin son: *join*, *cut*, *awk*, *sed*, *sort* y *uniq*, disponibles en los sistemas Unix[5].

Exportación a CSV

Una vez diseñada esta estructura en forma de tabla, procederemos a exportar los datos para generar un fichero CSV[6] que la replique. El formato CSV (Comma Separated Values) es un formato común de exportación que contiene solamente filas cuyos valores están separados por comas. Dicho formato no tiene información sobre el origen de esos datos o los tipos de valores que contiene cada columna. En el siguiente apartado veremos cómo esa información se puede inferir en la mayor parte de casos y podrá ser modificada cuando convenga.

El formato de los datos

Partiendo de nuestros datos en un fichero CSV podemos empezar a usar un servicio de Machine Learning para analizarlos. El primer paso, pues, será cargar nuestros datos en el servicio que hayamos escogido. En nuestro caso usaremos la interfaz web de BigML[7].

Para cargar un fichero desde nuestro ordenador basta con entrar con nuestras credenciales, previamente registradas, en BigML y arrastrar el fichero a la pantalla de listado de recursos. También podemos cargar datos desde URLs públicas, escribirlos en un editor en linea, o subirlos desde repositorios como Google Drive, Google Storage, Dropbox o MS Azure. Una vez cargados los datos veremos que se ha creado un objeto nuevo en la interfaz con el nombre del fichero subido. Llamaremos *Source* a este tipo de objetos, que almacenan las características necesarias para interpretar:

- **Los campos que contiene nuestro fichero de datos.** La primera fila del fichero es analizada para determinar si contiene los nombres de los campos. De no ser así, se asignan nombres automáticamente.
- **El tipo de cada campo.** Al lado de cada nombre vemos una imagen que nos indica el tipo de valores que contiene ese campo.

Vemos un ejemplo de *Source* en la Figura 2.8

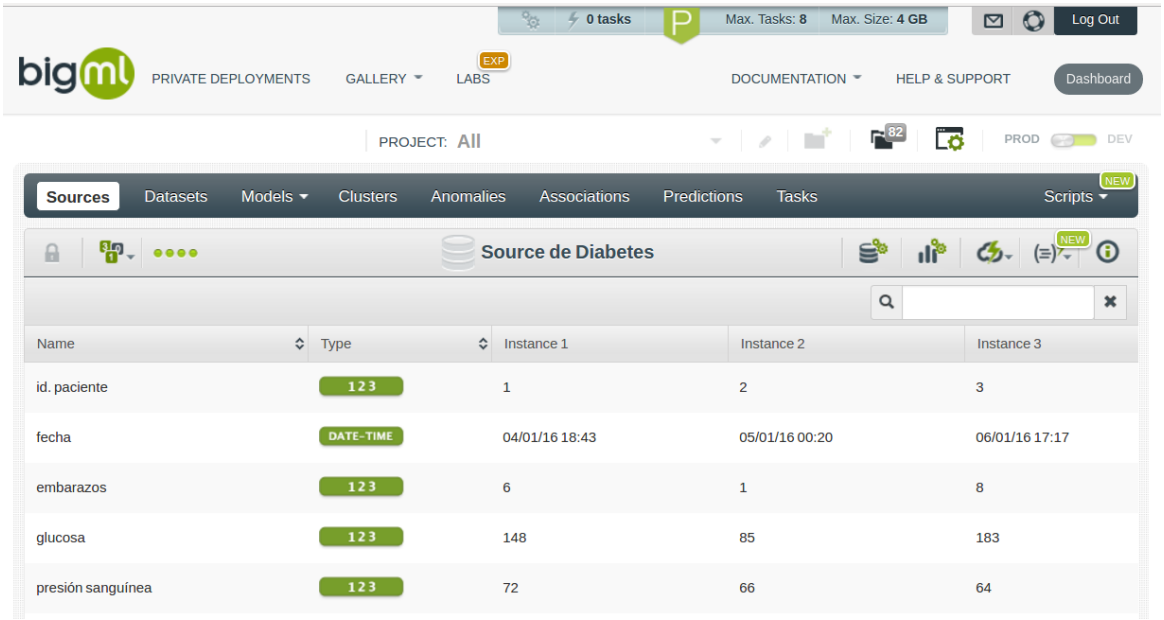

Figura 2.8: Vista del objeto *Source*. En esta pantalla se muestran los campos detectados en nuestro fichero, sus tipos y el contenido de las primeras filas.

Los tipos de los distintos campos se infieren a partir de los valores detectados en las primeras filas de nuestro fichero, y pueden ser modificados si es necesario. El tipo puede ser:

[5] https://www.ibm.com/developerworks/ssa/aix/library/au-unixtext/
[6] https://es.wikipedia.org/wiki/CSV
[7] https://bigml.com

- **Numérico:** Se considera numérico aquél campo cuyos valores sólo contienen números.
- **Categórico:** Se consideran categóricos los campos con un número limitado de valores de tipo texto que se repiten en varias instancias.
- **Fecha/hora:** Se identifican como campos de fecha/hora aquellos campos cuyo contenido encaja con alguno de los formatos de fecha/hora[8] más usuales. Los campos de este tipo son en realidad campos compuestos. Sus componentes (año, mes, día, hora, minutos, segundos) son los que pueden ser de utilidad en el aprendizaje y se añaden automáticamente al objeto *Source*. Este comportamiento puede ser cambiado en la pantalla de configuración de *Source* (Figura 2.9).
- **Items:** Se interpreta que un campo es de tipo *items* cuando contiene textos separados por algún carácter separador. La coma es el separador por defecto, con lo que los campos cuyos valores sean textos separados por comas (i.e.: *pan, jamón serrano, tomate*) serán considerados de tipo *items* automáticamente. Cada uno de estos textos entre comas se usará como una categoría o etiqueta independiente en el aprendizaje. El separador usado puede ser configurado globalmente para todos los campos items o bien particularmente para cada campo de este tipo en la pantalla de configuración de *Source*.
- **Texto:** Se interpreta que un campo es de este tipo cuando su contenido es de texto libre, es decir, que no está limitado a un subconjunto de etiquetas, o bien cuando el número de etiquetas distintas supera las 1.000. Estos campos se tratan al estilo de *bag of words*, separando las palabras y usando las frecuencias de las más significativas como nuevas propiedades para el aprendizaje. Existen diferentes configuraciones que pueden alterar este procedimiento: podemos elegir que no se separen las palabras, seleccionar el idioma del texto, que no se usen las *palabras vacías*, que se igualen mayúsculas y minúsculas, o que se use la lematización[9]. Todas estas configuraciones se pueden aplicar globalmente a todos los campos de tipo texto o individualmente usando la pantalla de configuración de *Source*.

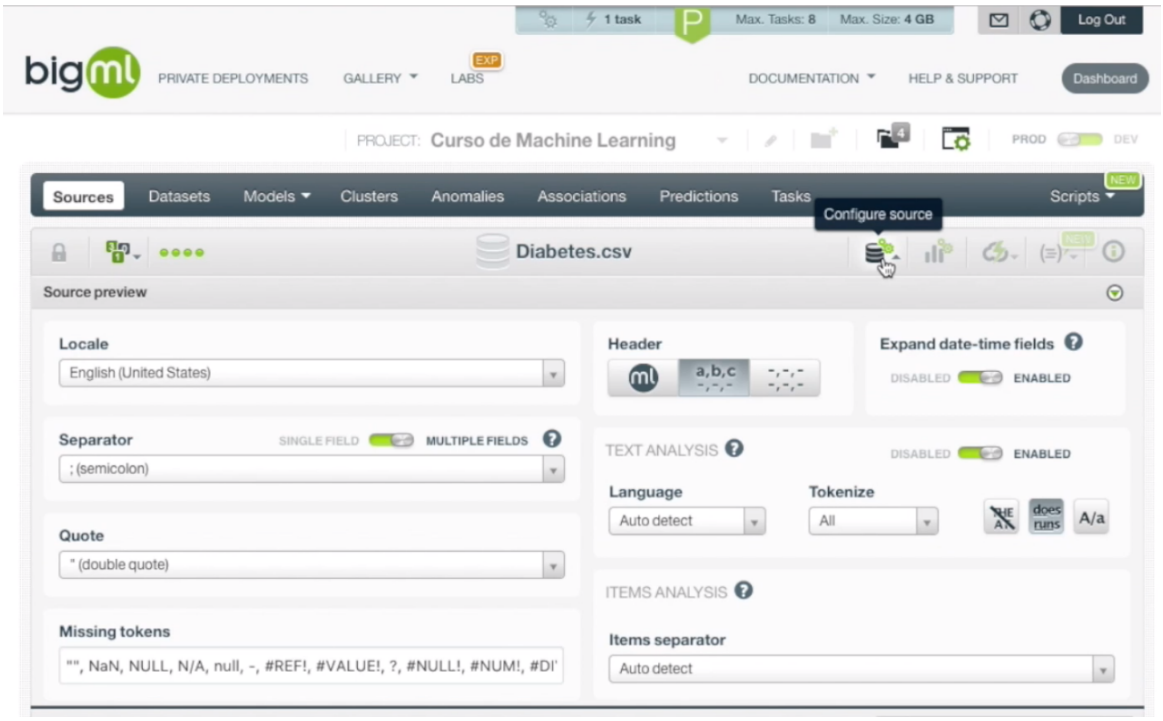

Figura 2.9: Pantalla de configuración del *Source*. Esta pantalla es la interfaz para poder cambiar cualquier atributo relacionado con el formato de los datos y los campos de nuestro fichero.

Cada uno de los campos detectados tendrá un identificador único, que podemos ver al pasar el ratón por encima de la etiqueta que muestra su tipo. Dicho identificador será el que se usará para hacer referencia al

[8] https://support.bigml.com/hc/en-us/articles/207423645-Which-date-time-formats-does-BigML-accept-
[9] https://es.wikipedia.org/wiki/Lematizaci%C3%B3n

campo si queremos modificar cualquiera de sus propiedades, como su nombre, tipo, descripción, etc. Se pueden añadir etiquetas y descripciones a los campos para hacerlos más comprensibles.

Finalmente, la pantalla de configuración de *Source* permite modificar algunas propiedades que determinan la correcta interpretación de los valores que contiene cada campo. El *locale* determinará el carácter decimal que se usa en los campos numéricos. El separador usado en el fichero CSV cambia la estructura de columnas. El carácter que se usa como comillas determina el contenido textual. También permite definir los textos que hay que interpretar como valores ausentes y la presencia o no de la fila de cabecera con los nombres de los campos, más las opciones ya comentadas en la sección de tipos de campos.

Aunque los tipos de los campos se infieren de su contenido, existen algunos casos en que puede ser necesario cambiarlos. En algunas ocasiones los números enteros pueden ser códigos asociados a una categoría. En este caso deberíamos cambiar el tipo del campo de *numérico* a *categórico*. Otras veces podemos tener secuencias de etiquetas separadas por un carácter poco usual. Si no se ha detectado dicho separador automáticamente el campo puede aparecer como *categórico* y deberá ser cambiado a *items*, tal como aparece en la Figura 2.10. Otro caso usual es el de textos que no deben ser tratados como palabras por separado, como los nombres de países. En este caso, el cambio a realizar es de configuración del campo de texto: el valor de *tokenize* deberá ser *full terms only*.

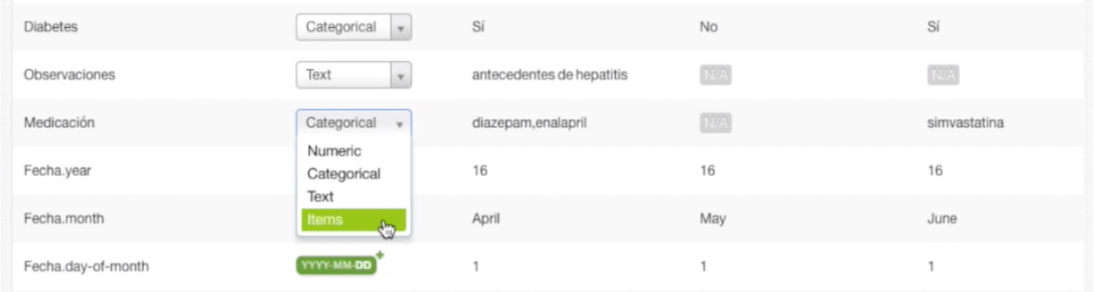

Figura 2.10: Pantalla de configuración del Source. Ejemplo de como cambiar un campo categórico a tipo items.

Una vez veamos que nuestros datos están correctamente estructurados y formateados, pasaremos al análisis de los valores que contiene cada uno de los campos creando un *Dataset*, que describiremos en la próxima sección.

2.2. Ingeniería de atributos antes del modelado

Usando la opción *1-click dataset* del menú de nuestro *Source* se crea un objeto *Dataset*. Éste es siempre el paso previo al aprendizaje, ya que en él se analiza el contenido de cada uno de los campos y se serializan los valores encontrados.

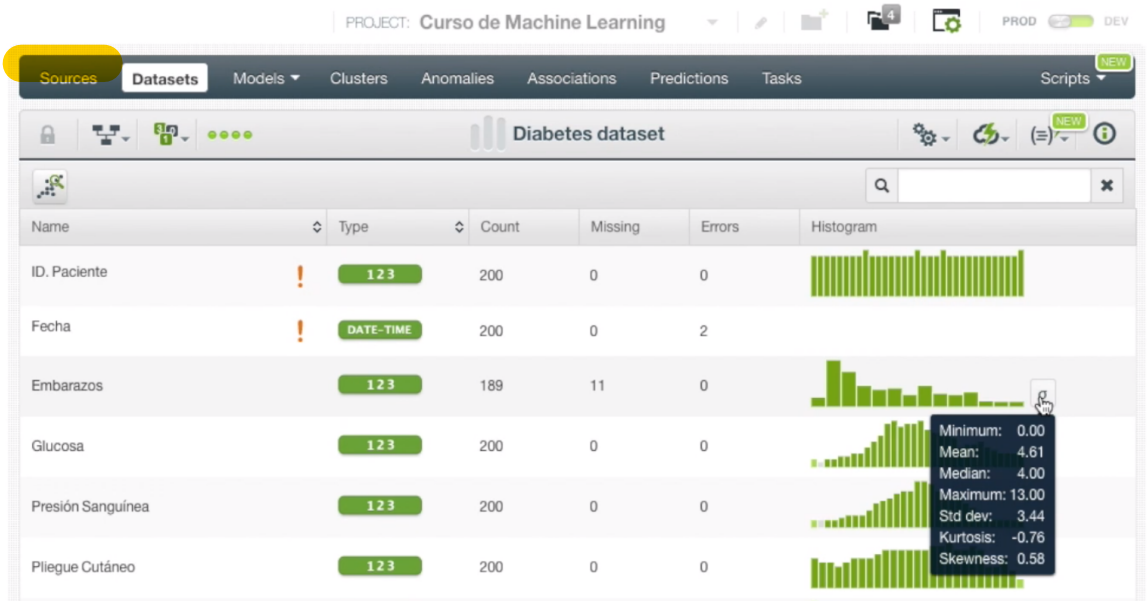

Figura 2.11: Ejemplo de pantalla de *Dataset* para el fichero de Diabetes. La pantalla de *Dataset* muestra la lista de campos y el resumen de sus valores.

En la Figura 2.11 vemos la pantalla que nos muestra la información de nuestro *Dataset*. Cada fila contiene un resumen de la información encontrada en cada uno de los campos que fueron detectados en el *Source*. Las columnas contienen:

- el número de valores encontrados
- el número de filas del fichero donde el campo estaba vacío
- el número de errores (como detectar textos en un campo numérico)
- la distribución de dichos valores, representada gráficamente en un histograma.

Cada tipo de campo presenta un histograma[10] adaptado a su naturaleza. En los campos numéricos se representa el número de instancias cuyos valores están en un rango determinado. En los categóricos, el número de instancias que tienen una categoría asociada. En los campos de tipo *items*, la frecuencia de aparición de cada ítem. En los campos de tipo *texto*, la frecuencia de aparición de cada palabra.

Junto al histograma algunos de los campo presentan un desplegable. Para los campos numéricos el desplegable muestra las estadísticas básicas, como el mínimo, el máximo, la media y la desviación estándar. En los campos de tipo *items* o *texto* aparecen los términos más usados en un formato gráfico de nube de palabras[11] (o *tag cloud*), donde el tamaño de las palabras es proporcional a la frecuencia con la que aparecen.

Algunos campos pueden contener valores que no son útiles para el aprendizaje. Supongamos el caso de un campo donde se guarda el identificador de usuario. Los valores que contiene son enteros completamente distintos entre sí, y por lo tanto no habrá más de una instancia de cada valor. Este tipo de campos no será útil para el aprendizaje (*non-preferred*), y eso es lo que señala la exclamación que aparece en el primer campo de la Figura 2.11. Otro caso en que el campo aparecerá como *non-preferred* son los campos de fecha/hora, aunque en este caso el motivo es que se trata de campos compuestos. Sus componentes, como el año, mes, día, etc., son generados automáticamente y sí serán tenidos en cuenta. Lo mismo ocurrirá para campos que sólo tengan un valor constante o para campos de texto con muy pocos valores. Aunque un campo se haya marcado como *non-preferred*, se puede cambiar esta decisión usando el menú de edición que hay junto al nombre del campo. Todos aquellos campos que no estén marcados como *non-preferred* serán usados en la creación de modelos de aprendizaje.

Si queremos excluir alguno de los campos existentes en el dataset para que no se use en el proceso de modelado, podemos hacerlo desde la pantalla de configuración del modelo en cuestión. Sólo deberemos desmarcar la casilla del campo correspondiente antes de crear el modelo, tal como vemos en la figura Figura 2.12.

[10] https://es.wikipedia.org/wiki/Histograma
[11] https://es.wikipedia.org/wiki/Nube_de_palabras

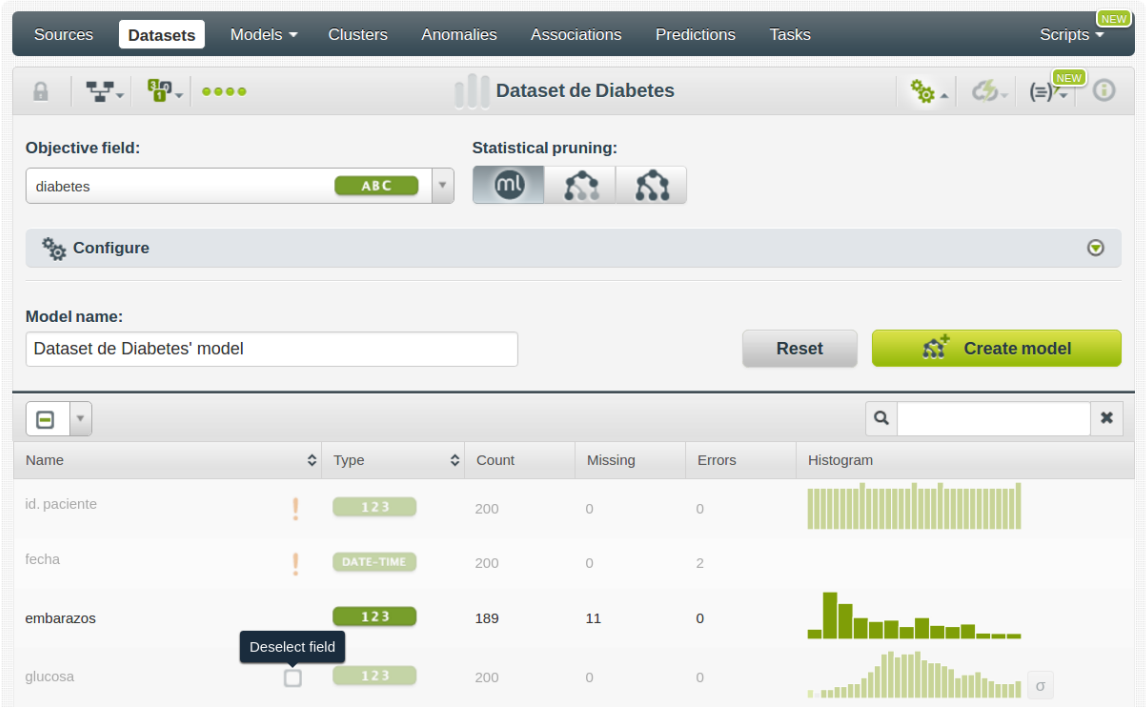

Figura 2.12: Pantalla de configuración del *Modelo*. Esta pantalla es la interfície para poder definir cualquier parámetro necesario en la creación del *Modelo*. En este caso, el ejemplo muestra como excluir un campo del aprendizaje.

Feature engineering para el aprendizaje

Los algoritmos que usa el Machine Learning para aprender no tienen ningún conocimiento sobre el ámbito al que pertenecen los datos ni contexto para utilizarlos. Por ejemplo, cuando un campo es numérico sólo saben que los números que contiene siguen un orden. No saben cuándo un número es par o impar, o primo, o si está dentro o fuera de un rango determinado. El algoritmo recibe el valor de un campo tal como está, y le aplica sus procedimientos para intentar aprender de él sin más información.

No obstante, las informaciones como que para agrupar un campo *edad* podría ser interesante crear tres grupos: jóvenes, adultos y mayores, o que en un campo *número* se puede distinguir entre pares y nones y eso indicará la acera de la calle donde se ubica un edificio pueden ser importantes para que el modelo funcione.

Por lo tanto, en estos casos nosotros deberemos introducir esa información para ayudar al algoritmo. La forma de hacerlo es crear nuevos campos transformados a partir de los que tenemos en nuestro *Dataset* y usar el nuevo *Dataset* extendido para entrenar el modelo.

Cada nuevo campo almacenará la información sobre un predicado (por ejemplo, a qué rango de edad pertenece un individuo) o un valor calculado a partir de los datos disponibles. Es lo que se conoce como *feature engineering* o ingeniería de atributos. Siguiendo el ejemplo propuesto de los datos de diversos pacientes de diabetes, podemos ver que, para once instancias, el campo *embarazos* no tiene un valor asignado. Eso podría ocurrir por varios motivos. El más sencillo es que no se haya podido obtener dicha información, y en ese caso sería bueno rellenar estos casos con el valor más frecuente, o el valor medio. También podría ser que la ausencia de valor para el campo embarazos fuese una señal de que esas filas corresponden a pacientes masculinos, para los cuales el campo no tiene sentido. En cualquiera de estos dos casos, deberíamos transformar nuesto dataset usando la opción *add fields to dataset* del menú de *Datasets*. En el primer supuesto, añadiendo un valor por defecto que fuese razonable. En el segundo, creando un nuevo campo que contenga información sobre si el paciente es un hombre o no.

En la Figura 2.13 se muestran algunas de las posibles transformaciones que podemos aplicar a los campos.

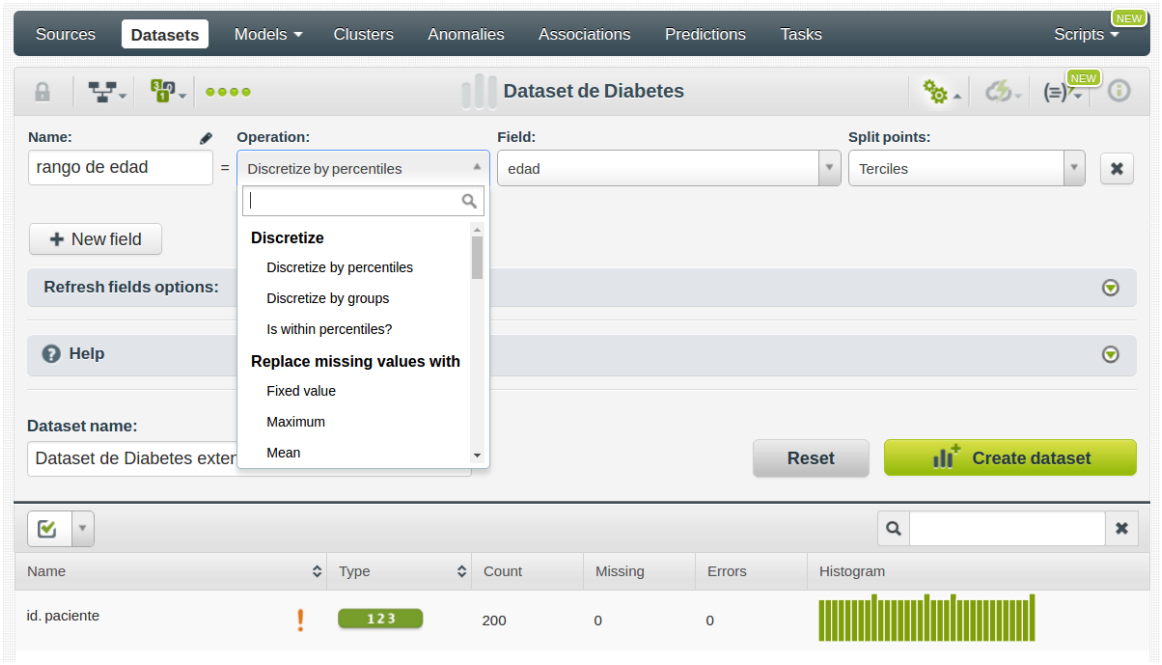

Figura 2.13: Pantalla para añadir nuevos campos a un *Dataset*. El ejemplo muestra como transformar un campo numérico a categórico usando percentiles.

Hay transformaciones usuales, como la discretización de los valores de un campo, reemplazar los valores ausentes por algún valor fijo, normalizar, o utilizar valores elegidos al azar. Así pues, transformar el campo *edad* en un campo categórico con tres rangos de edad se puede hacer usando una discretización del campo por terciles. Normalmente, una vez construidos estos campos derivados, eliminaremos el campo del cual proceden de nuestro dataset, o lo excluiremos del conjunto de campos a usar en el aprendizaje. Tener dos campos directamente dependientes como candidatos a predictores para un modelo podría afectar negativamente, ya que ambos están aportando la misma información y eso podría confundir al modelo.

En general, cualquier transformación es posible usando una *s-expression*. Existe un lenguaje para construirlas llamado flatline[12]. A modo de ejemplo, podríamos construir un nuevo campo *hombre* que contuviera un *Sí* o un *No* según si el campo *embarazos* está vacío. La expresión requerida sería:

```
(if (missing? "embarazos") "Sí" "No")
```

Las expresiones pueden ser también construidas en formato JSON:

```
["if", ["missing?", "embarazos"], "Sí", "No"]
```

Hay que recordar que el *Dataset* es la base de nuestro aprendizaje. Una buena ingeniería de datos puede ser, pues, determinante en la solución de nuestro problema. Para ello, es indudablemente necesario un buen conocimiento del dominio en que vamos a aplicar el Machine Learning. Dado el gran número de transformaciones posibles, no existen todavía soluciones automatizadas que permitan determinar cuál es la mejor combinación de nuestros datos de cara al aprendizaje, aunque se está empezando a avanzar en esta dirección.

[12] https://github.com/bigmlcom/flatline