Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Industrial

Tarea 1

Integrantes: Michael Clemans
Paula Marín
Bárbara Pérez
Sebastián Urbina
Profesor: Ángel Jiménez Molina
Auxiliar: Rafael De la Sotta Vargas
Fecha de entrega: 2 de octubre de 2021

Pregunta 1.

Para tener una visión general del problema e identificar algunas de las variables se buscan las variables que cuentan con ceros, se observan los productos más vendidos y con mayor inventario, se calcula la cantidad de productos que están activos, se observan los precios pagados por los usuarios y proveedores, se generan diagramas de caja y gráficos para observar las variables y obtener distintas distribuciones (Figura 3), tales como la distribución de inventario que distribuye parecido a una Chi-Cuadrado con distintos datos outliers (Figura 1), y finalmente se genera una matriz de correlación. En esta última (Figura 2) se observa que en general las variables no están muy fuertemente correlacionadas tanto de manera positiva como negativa, pues las correlaciones que destacan corresponden a 0,61 entre *SoldFlag* y *SoldCount*, 0,47 entre *New Release Flag* y *Release Number*, y -0,36 entre *StrengthFactor* y *ReleaseYear*.
Específicamente, se filtra en función de *SoldFlag* y se evidencia las mayores correlaciones entre esta y *SoldCount* (0,61), *ItemCount* (0,24) y *Marketing Type* (0,22), y de forma negativa se encuentran *LowNetPrice* (-0.014) y *StrengthFactor* (-0,14).

Luego de todo el análisis, se define que los productos que van a ser eliminados son aquellos que no se han vendido en los últimos 6 meses, es decir, con un $Soldflag = 0$, las variables que permitirán distinguir los productos a eliminar son las variables que afectan a *SoldFlag* de una manera u otra.

De forma inmediata se pueden eliminar del modelo las columnas correspondientes a *Order* y el *SKU number*, porque solo sirven para indexar los productos. Además, se hace una limpieza de valores de null, con lo que se identifican que hay 122.921 de estos para *SoldCount* y *SoldFlag*. Esto tiene sentido, pues corresponden a los valores activos que se buscan predecir y por tanto se dejan de lado, pasando a analizar la base de datos filtrada por los productos históricos.

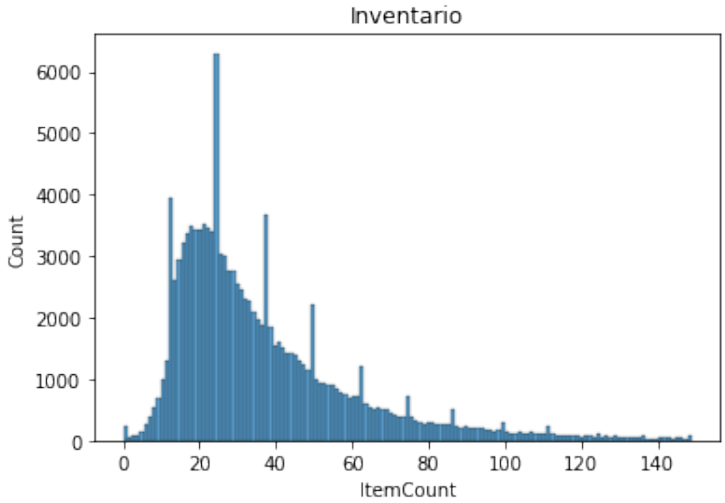

Figura 1: Histograma inventario

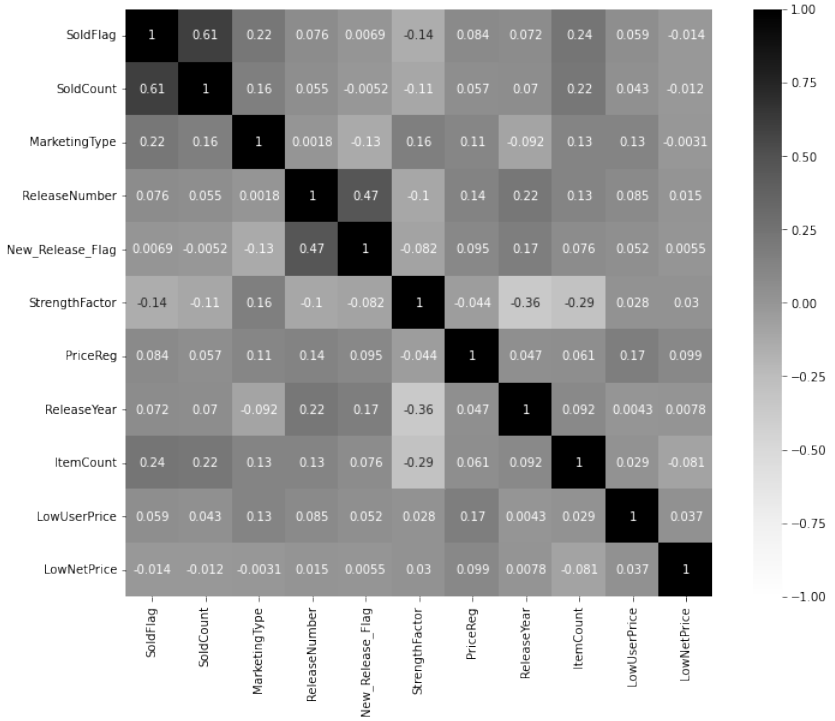

Figura 2: Gráfico de correlación entre las variables numéricas en los datos históricos

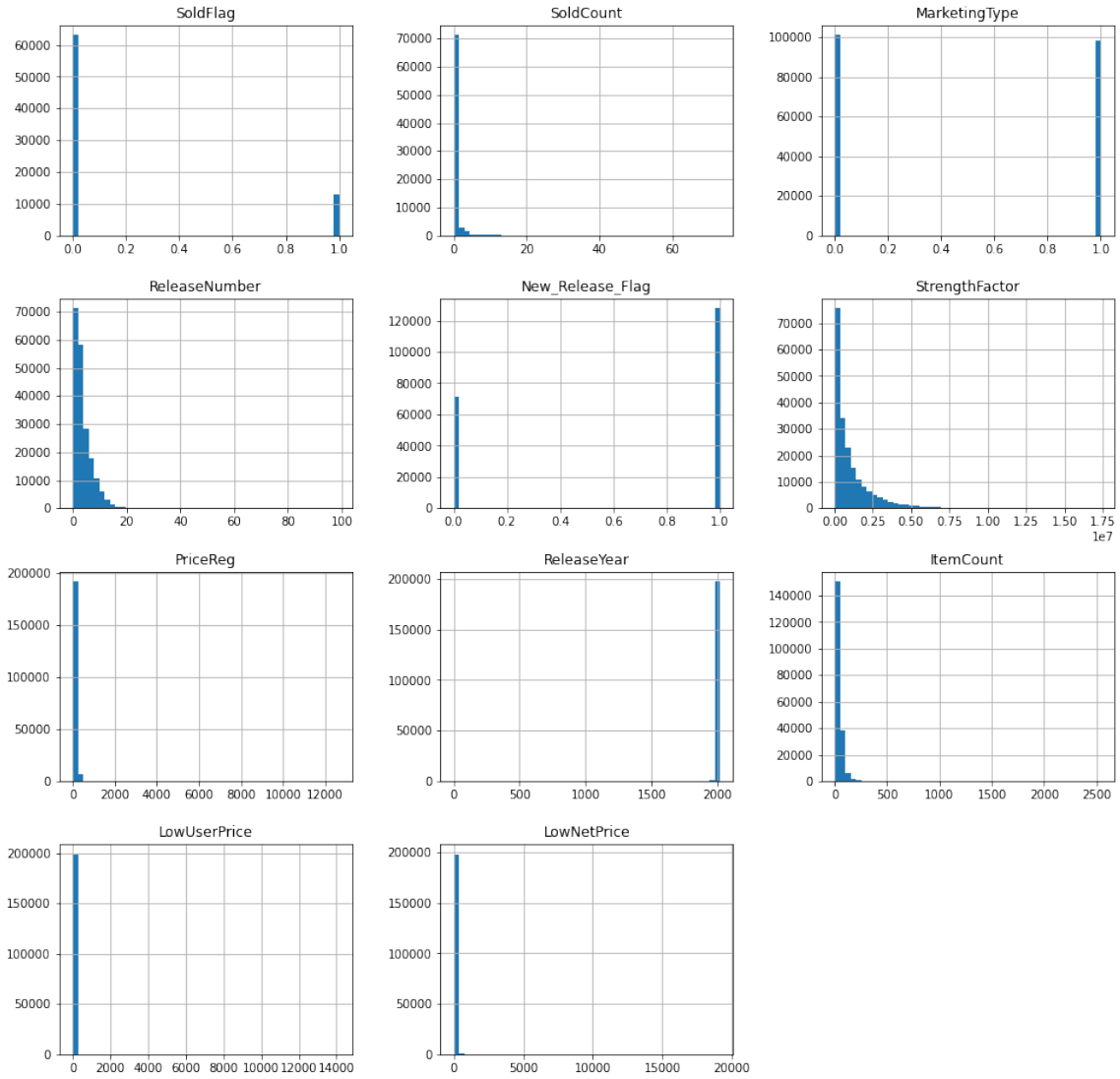

Figura 3: Histograma variables todos los datos

Pregunta 2.

Si es plausible definir un modelo que logre predecir el nivel de ventas que tendrá cada producto. En base a los datos que tenemos, se puede identificar rápidamente que la variable objetivo corresponde a la variable *SoldCount*, porque esta entrega la cantidad de ventas de los últimos 6 meses de cada producto, que después se proyectarían para un período en el futuro.

En cuanto al modelo de pronóstico como tal, se podría armar un regresión lineal, seleccionado *SoldCount* como la variable dependiente y las variables explicativas como aquellas que tengan alguna correlación significativa (en función del análisis descriptivo) con *SoldCount*, descartándose el resto.

Pregunta 3.

Primero que todo, para entrenar una red neuronal se deben identificar las variables de entrada y la variable objetivo. Como se quiere determinar que productos "activos" retirar del inventario, se separará la variables *Filetype* en dos *DataFrame* correspondientes a activos e históricos, y así, utilizar esta información para entrenar una red neuronal, donde, la variable objetivo será *SoldFlag*, ya que esta nos indica si un producto se vendió en los últimos 6 meses, por lo que podemos suponer que prediciendo esta variable, tendremos una buena idea de qué productos inventariar o no.

Asimismo, se considerarán como variables de entrada, todas las disponibles, excepto:

- Order: Pues, no aporta información respecto a la variable a predecir, sólo sirve como un identificador.
- FileType: Pues, se utilizará sólo para filtrar los datos históricos, con los cuales se entrenará el modelo.
- SKU number: Al igual que Order, sólo sirve como un identificador y no aporta información sobre la variable a predecir.
- Soldcount: Esta variable está estrictamente correlacionada con la variable a predecir, pues si se vendió un producto en los últimos 6 meses(1),
- SoldCount debe ser mayor a 0. Y si no se vendió un producto en los últimos 6 meses(0) SoldCount debe ser 0.

Adicionalmente, se deberá transformar *MarketingType* de una variable categórica a numérica, reemplazando los D por 1 y los S por 0.

Se utilizará un 70 % de los datos para entrenar, un 20 % para validar y un 10 % para testear.

Para generar estos conjuntos, se utilizará la librería de *sklearn*, específicamente la función *train_test_split*, con la cual se dividirá el conjunto de datos en la proporción mencionada, compuesto por las variables:

- MarketingType
- ReleaseNumber
- New_Release_Flag
- StrengthFactor
- PriceReg
- ReleaseYear
- ItemCount
- LowUserPrice
- LowNetPrice

Posteriormente se utilizará el parámetro del método *compile* de *keras* para definir el *validation split* en 0.1.

De esta manera, se tendrán divididos los datos en la proporción mencionada.

Pregunta 4.

Debido a que existe bastante desbalance en las clases a predecir, se utilizará el método de *Synthetic Minority Oversampling Technique(SMOTE)*[1] para realizar *oversampling* e igualar las clases para así, buscar un modelo más robusto.

Adicionalmente, se normalizarán los datos, vale decir, a cada *feature* restarle su media y dividirlo por la desviación estándar.

$$Normalization(\vec{x}) = \frac{\vec{x} - mean(\vec{x})}{std(\vec{x})}$$

Para la construcción del modelo se utilizarán 4 capas, ajustándose la cantidad de neuronas en función de los resultados de *accuracy*, *precision*, *recall* y *loss*. Se replicará el mismo procedimiento para los parámetros del *epoch* y *batch*. La metodología se basará esencialmente en prueba y error.

Por último se utilizará el método *early stopping*, para detener al modelo en caso de que exista un sobre ajuste, para así poder regularizarlo. Se definirá una tolerancia máxima de 5 respecto a la función de pérdida *loss*, es decir, en cada epóca, si el modelo no reduce su *loss* se detendrá el entrenamiento.

A continuación, se puede ver la arquitectura de la red neuronal definida, el cual tiene como *Input Layer* 9 nodos, luego 32 y 16 nodos en *Hidden Layer* y finalmente 1 nodo en *Ouput Layer*.

[1] https://arxiv.org/abs/1106.1813

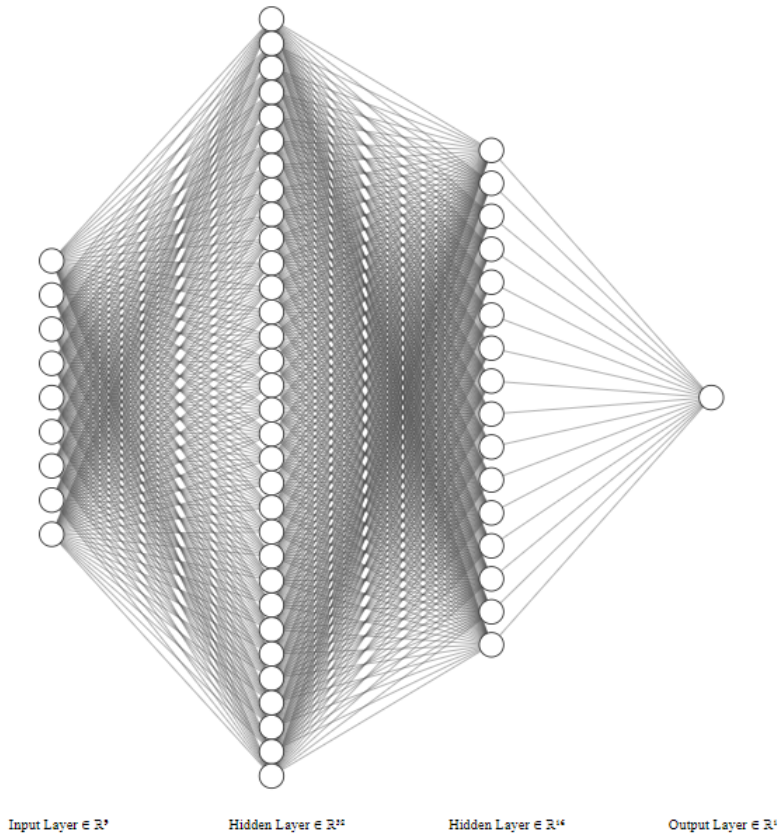

Figura 4: Diagrama de la red neuronal

- Función de activación Hidden Layer 1: ReLu
- Función de activación Hidden Layer 2: ReLu
- Función de activación Output Layer: Sigmoid

Además, entre las capas *Input Layer*, *Hidden Layer 1* y *Hidden Layer 1*, *Hidden Layer 2* se agregó *DropOut*.

Luego de ajustar prueba y error se definieron los siguientes hiper parámetros:

- $Batch\_size = 64$
- $n\_epochs = 100$

Finalmente, para ajustar el *DropOut* se utilizó la librería de *optune* para buscar la mejor probabilidad de *DropOut*, con lo que se llegó a un valor de $DropOut = 0.1$.

En base a estos resultados se entrenó el modelo final:

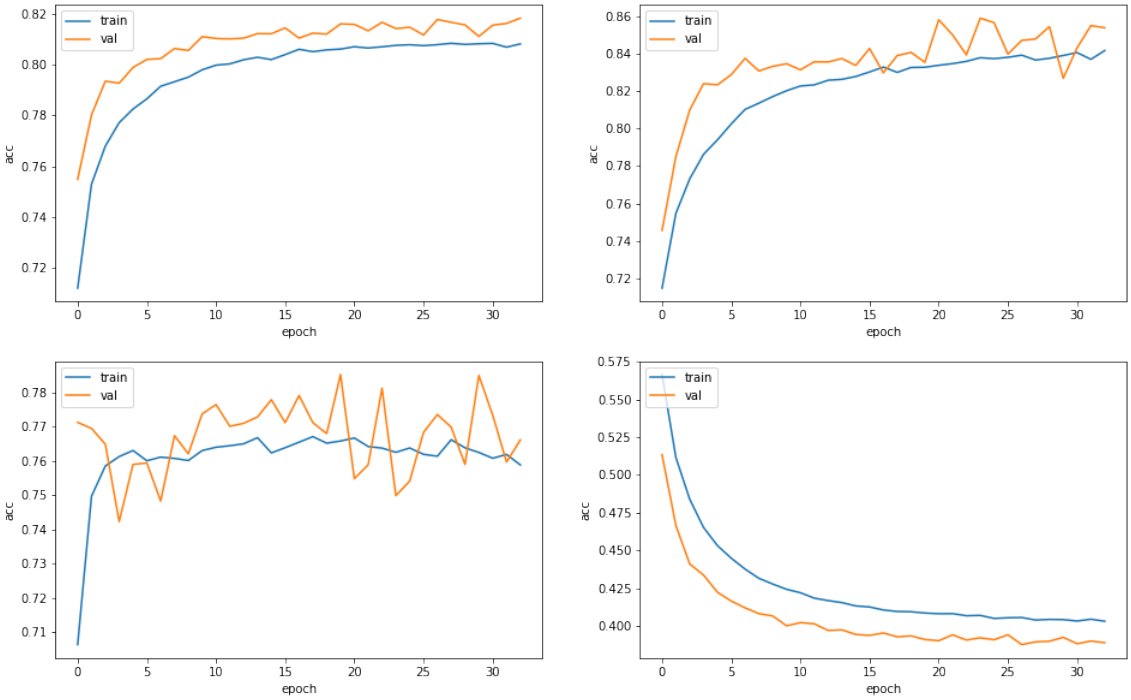

Figura 5: Metricas entrenamiento

Pregunta 5.

En efecto, se puede crear un *ranking* para los productos, utilizándose como criterio el numero de ventas históricas de aquellos productos activos que se predigan vender por una red neuronal. Más en especifico, lo que se puede hacer es tomar el *SKU_number* o el identificador de cada producto activo que el modelo predijo que se venderá (a forma de indexar los productos), buscar la información histórica del producto o la cantidad de ventas que tuvo en el pasado y en función de estas asignarle un lugar o *ranking*.

Cabe añadir que *ranking* no tendrá el detalle de los productos "nuevos" o aquellos productos de los que no se tienen datos históricos, al ser imposible jerarquizar 0 ventas.

Tabla 1: Ranking en función de sus ventas históricas[a b]

Ranking	**SKU number**	**Ventas**
1	665269	73
2	613864	69
3	767846	36
4	55769	36
5	141824	33
6	747765	30
7	254192	28
8	860747	27
9	540449	27
10	52194	26

[a] Las ventas corresponden a los datos históricos del producto
[b] Aquellos productos que no tienen datos históricos, no están considerados.

Se adjunta el archivo *.csv* que contiene el ranking completo.

Pregunta 6.

Para enfrentar este problema, es útil usar mecanismos de *deep learning* en comparación a los de *machine learning*, dado que en este problema se hace necesario predecir y clasificar, con el fin de mejorar la toma de decisiones, tales como, que producto eliminar o mantener. El *deep learning* precisamente, se utiliza para la toma de decisiones inteligentes de una manera mas avanzada que el *machine learning*, en donde se procesan y analizan datos con una estructura lógica similar a la de un cerebro humano.

Además, con un modelo de *deep learning* un algoritmo puede determinar por sí mismo si una predicción es precisa o no a través de los pesos que se minimizan, es decir, aprende a través método de *backpropagation*, teniendo un nivel de autonomía superior al de *machine learning*. Por otro lado, en arquitecturas de *deep learning*, los modelos pueden encontrar patrones imperceptibles para un humano, dado que los preceptrones reflejan el comportamiento de neuronas, tienen muchas conexiones y permiten establecer relaciones particulares, sin embargo, los modelos actúan como cajas negras, a diferencia de modelos de *machine learning*, donde se sabe que ocurre en cada momento con los datos.

También es útil, dado que en el paradigma de *machine learning* se requiere más trabajo de preprocesamiento y la realización de nuevos *features*, cosa que se simplifica bastante al utilizar *deep learning*, pues, se entregan los datos, se define una arquitectura y el modelo puede lograr resultados muy buenos en poco tiempo.

Anexo

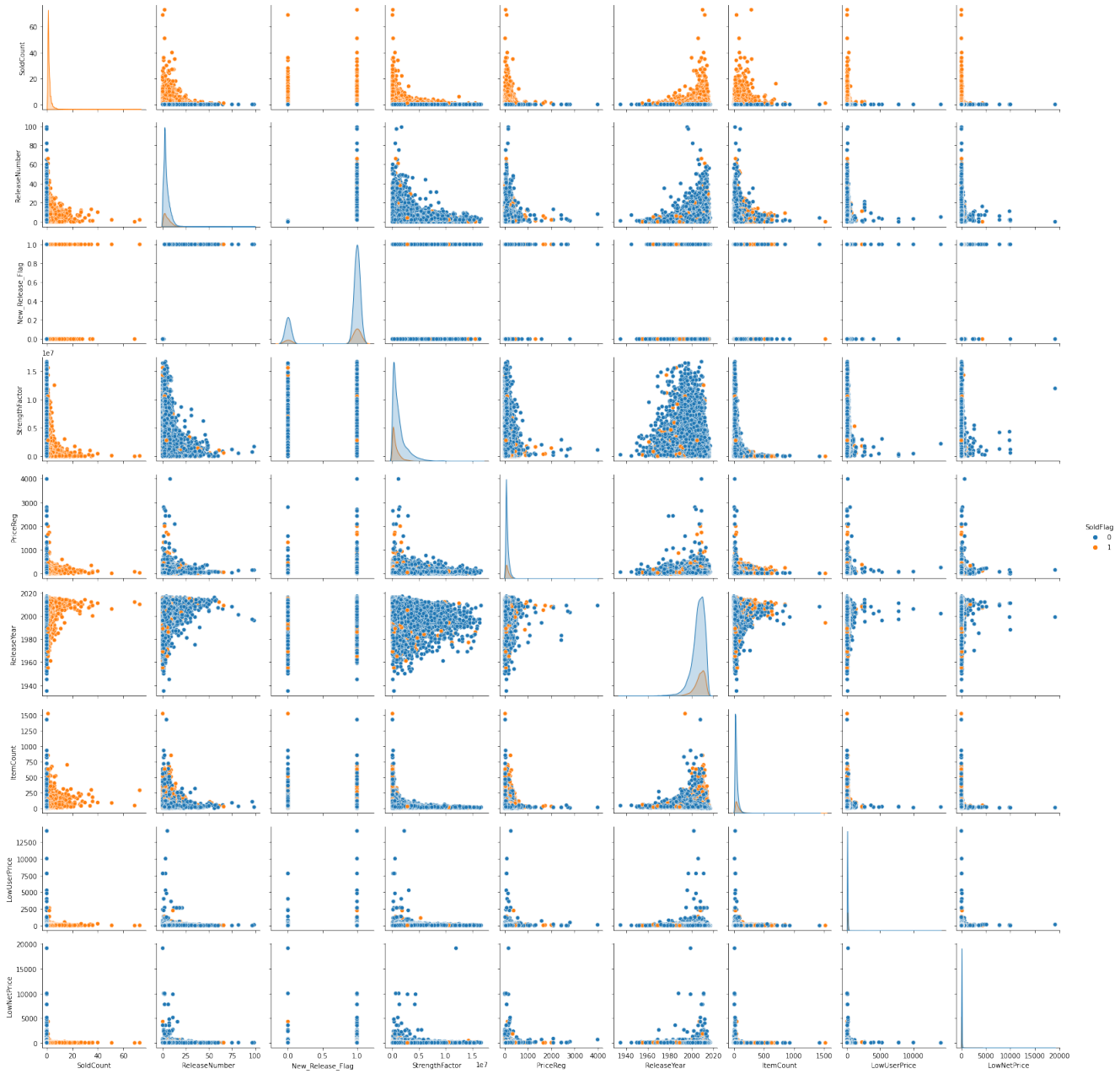

Figura 6: *pairplot* entre las variables numéricas de los datos históricos