Introducción a la inteligencia artificial - UdeA

PROYECTO: ENTREGA Nº2

NATALIA ANDREA GARCÍA RÍOS
MATEO VELÁSQUEZ RODRÍGUEZ

0. FUNCIONES CREADAS

En esta sección, explicaremos las funciones creadas y utilizadas en los notebooks.

get_corr_matrix: Toma dos argumentos: group, dataset. El parámetro *group* es una lista de columnas de algún *dataset*. Esta función se encarga de hallar y retornar la matriz de correlación entre dichas columnas. Además, se encarga de graficar la matriz de correlación.

reduce_columns: Toma dos argumentos: corr_matrix, dataset. El parámetro *corr_matrix* se refiere a la matriz de correlación de algún grupo de columnas del *dataset*. La función genera pares de columnas, representados en tuplas, que tienen una correlación mayor a 0.7, un nivel moderado de correlación. Se da la claridad de que en el caso de que una columna tenga ese o un mayor nivel de correlación con varias columnas, se empareja con la columna con la cual mayor nivel de correlación tenga. A partir de dichos pares, la función evalúa cuál columna tiene mayor número de valores únicos y lo agrega a las columnas que vamos a conservar, por el contrario, la columna con menor número de valores únicos se enlista en las columnas a eliminar. Finalmente, imprime las columnas a conservar y a eliminar y retorna una lista con las columnas que se eliminarán.

1. CARGA DE DATASETS

Para comenzar, cargamos los conjuntos de datos *train_transaction.csv*, *train_identity.csv*, *test_transaction.csv* y *test_identity.csv*. Posteriormente, unimos los datos de entrenamiento *train_transaction.csv* y *train_identity.csv*, así como los de prueba *test_transaction.csv* y *test_identity.csv*. Utilizamos la función merge de la librería pandas para combinar los conjuntos de datos de entrenamiento y prueba. Optamos por esta opción debido a que al emplear el parámetro how='left' en la función merge y pasar, por ejemplo, *train_transaction.csv* como primer conjunto de datos y luego *train_identity.csv*, podemos descartar los clientes que no hayan realizado ninguna transacción y así evitar la presencia de datos vacíos en el análisis.

Por otra parte, se renombraron las columnas del dataset *test_identity.csv* con los nombres de las columnas *train_identity.csv*, puesto que los nombres de las columnas en los archivos de identidad de entrenamiento y prueba son diferentes, pero representan la misma información.

Finalmente, guardamos en la variable train_y la columna *isFraud* del conjunto de datos de entrenamiento train, el cual posee 590540 filas y 433 columnas.

2. EXPLORACIÓN DE DATOS

A continuación se describirán los pasos seguidos realizados para llevar a cabo la exploración de datos en el dataset de entrenamiento: train.

2.1. Inicialmente, identificamos el tipo de dato que hay en cada columna.

2.2. Posteriormente calculamos el número datos faltantes (NaN) por columna en el conjunto de datos train utilizando la función isna , seguida de la función sum para sumar la cantidad de valores faltantes por columna. El resultado se almacena en la variable nan; todo esto con el fin de identificar cuáles columnas podemos descartar.

2.3. Graficamos para obtener una visión más dinámica de todos estos datos faltantes utilizando la instrucción train.isna().values.T, la cual identifica los valores faltantes en el conjunto de datos train y los convierte en una matriz de valores booleanos; la función plt.imshow() muestra la imagen en escala de grises de los valores faltantes en el conjunto de datos.

2.4. Calculamos la proporción de datos faltantes para todas las columnas y encontramos que el mayor porcentaje de datos vacíos que se encuentra es de 99.19%.

2.5. Calculamos la proporción de datos faltantes solo para las columnas *V*, esto dado que al momento de graficar en el paso 2.3, se visualizamos que la mayoría de columnas de este tipo tienen mayor proporción de datos vacíos en comparación a las demás; encontrando que la columna con el porcentaje máximo de datos faltantes es aproximadamente 86.12%.

2.6. En la inspección de columnas numéricas. Procedimos a realizar la matriz de correlación, utilizando la función get_corr_matrix explicada anteriormente, de varios grupos de columnas que podrían estar brindando información redundante, en este caso las columnas numéricas de tipo *C*, *D*, *V* e *id*.

Al finalizar la exploración de datos pudimos sacar varias conclusiones. Descubrimos una gran cantidad de columnas con su mayoría de filas vacías. Lo cual nos llevó a tomar la decisión de hacer una limpieza de datos, en este caso de columnas, y, de esta forma, eliminar el ruido, aquellas columnas que no proporcionaban información relevante.

Al realizar el paso 2.6. observamos alta correlación entre las columnas numéricas de tipo *C*, *D*, *V*, a excepción de las del tipo *id*, lo cual denotaba una baja calidad en la información registrada en el dataset y lo que confirmaba el hecho de que muchas de esas columnas podrían ser innecesarias, es decir, proporcionaban información redundante. Se debía hacer una limpieza de datos.

3. LIMPIEZA DE DATOS

Basados en los procedimientos descritos en *EDA for columns V and ID* y *XGB Fraud with magic - [0.9600]* en la plataforma Kaggle. los pasos seguidos para realizar una limpieza de datos en los datasets y, así, asegurar la calidad de los mismos.

Después de realizar el análisis de correlación entre las columnas del conjunto de datos, se observaron celdas oscuras en los gráficos que indican una alta correlación entre las variables. Esta observación hizo evidente la necesidad de reorganizar o limpiar las columnas sin perder información importante. Para lograr este objetivo, se tomó la decisión de eliminar aquellas columnas que presentaban la misma cantidad de valores faltantes y que mostraban una alta correlación entre sí. Esta elección se basó en el hecho de que las columnas que tenían la misma cantidad de datos faltantes y un grado de correlación alto o moderado, proporcionaban prácticamente la misma información.

3.1. Agrupamos las columnas con el mismo número de datos vacíos y, por cada grupo, calculamos y graficamos la matriz de correlación, implementando la función get_corr_matrix.

*Se hace claridad en el hecho de que el proceso que se está describiendo se hizo en su mayoría con columnas del tipo *V*, ya que no se encontró un valor igual de datos faltantes que nos permitiera agrupar las columnas de tipo *C* y *D*, en las cuales también notamos un gran nivel de correlación.

3.4. Una vez obtenida la matriz de correlación, llamamos la función reduce_columns, la cual nos retornará una lista con las columnas a eliminar, dichas columnas, como es de predecir, son eliminadas del dataset.

3.5. Identificamos cuántas columnas se eliminaron: 214 columnas, y el tamaño resultante del dataset: 590540 filas y 219 columnas.

3.6. En última instancia, se volvió a generar la matriz de correlación de las columnas *V*.

Desafortunadamente, la limpieza de datos no cumplió con nuestras expectativas, ya que las columnas *V* todavía presentan una alta correlación, es decir, se encuentra una gran cantidad de información redundante. Por lo tanto, debemos investigar más métodos para reducir el número de columnas y disminuir la correlación existente.

4. REFERENCIAS

[1] Cdeotte, C. (n.d.). EDA for Columns V and ID. Kaggle. Recuperado el año 2023, de https://www.kaggle.com/code/cdeotte/eda-for-columns-v-and-id/notebook

[2] Cdeotte, C. (n.d.). XGB Fraud with Magic 0.9600. Kaggle. Recuperado el año 2023, de https://www.kaggle.com/code/cdeotte/xgb-fraud-with-magic-0-9600/notebook