TRATAMIENTO INTELIGENTE DE DATOS

DETECCIÓN DE FRAUDE BANCARIO

JAVIER CORREA MARICHAL
VIREN SAJJU DHANWANI DHANWANI
GABRIEL GARCÍA JAUBERT
ÁNGEL TORNERO HERNÁNDEZ

GRUPO 4

Índice

1. Introducción

El fraude con tarjeta de crédito es una forma de estafa común cometido a través de una tarjeta de pago, ya sea de crédito o de débito. Solo en 2018, se estima que a través de este tipo de operaciones no autorizadas se perdió un total de 844.8 millones de libras de todo Reino Unido. Sin embargo, ese mismo año, el trabajo de bancos y de proveedores de tarjetas de créditos permitió prevenir el tráfico de 1,660 millones de libras, identificando de forma precoz la realización de actividades sospechosas y deteniendo estas operaciones. En otras palabras: de cada £3 que intentaron ser robados a través de este fraude, £2 fueron detenidos con éxito por las entidades bancarias.

Puesto que este tipo de estafas se encuentra a la orden del día, se destaca la importancia de contar con un sistema que permita prevenir eficazmente transacciones similares, permitiendo así evitar que los clientes afectados sean cobrados por una compra que no realizaron. En esta práctica, se plantea crear un sistema capaz de reconocer y clasificar transacciones como fraudulentas, permitiendo detener desde una etapa temprana la perpetración de este tipo de estafa.

Para ello, nuestro grupo propone dar una respuesta a través de las técnicas y algoritmos estudiados en las clases teóricas de la asignatura a las siguientes preguntas:

- ¿Podemos predecir si una transacción es o no fraudulenta? ¿Con qué grado de certeza?

- ¿Es posible encontrar un conjunto de características comunes a todas las transacciones fraudulentas?

- ¿Es viable desplegar el sistema desarrollado en un entorno real?

2. Datos

2.1 Descripción de los datos

El conjunto de datos con los que se trabajará en este proyecto fue obtenido a través del registro de las transacciones a través de tarjeta de crédito realizadas en septiembre de 2013 por compradores de origen europeo. Durante el período de 2 días, se registraron 492 operaciones fraudulentas de entre un total de 284,807 transacciones.

Puesto que estas transacciones se corresponden con operaciones de clientes reales, el conjunto de datos ha sido ofuscado con el fin de evitar revelar información personal de los afectados. Las variables que componen el conjunto de datos son el resultado de una transformación PCA, por lo que el contenido está compuesto por entradas puramente numéricas. Las características del conjunto de datos han sido renombradas a través de etiquetas genéricas que ocupan el rango de 'V1' hasta 'V28'; con la excepción de las columnas 'Time' y 'Amount', que representan el tiempo relativo a la primera transacción registrada y la cantidad de dinero involucrada en la misma, respectivamente.

La característica 'Class' del conjunto de datos representa la variable de clase que se desea predecir. Esta variable categórica puede tener uno de dos posibles valores: 0, si la transacción fue clasificada como no fraudulenta; o 1, en caso contrario. Como fue comentado brevemente con anterioridad, el número de casos etiquetados como un intento de estafa supone un 0.172% del total de transacciones registradas.

Esta característica es una de las razones que hace que este conjunto de datos sea difícil de tratar: se encuentran registradas más entradas correspondientes con una variable de clase que con su contraria, por lo que mediante un entrenamiento tradicional de los modelos utilizados se podría caer en una tendencia a clasificar todas las entradas como transacciones no fraudulentas, pues esta clasificación sería exitosa un 99.83% de las veces. Este comportamiento no sería aceptable en un modelo donde se intenta detectar la máxima cantidad de fraudes cometidos posible, por lo que se ha de estudiar una forma de reducir la dimensionalidad de los datos del conjunto de trabajo de tal forma que se pueda obtener un resultado similar al que se obtendría utilizando el conjunto de datos original, pero sin caer en ajustar el modelo en exceso para la detección de una variable de clase concreta.

Por esta razón, a la hora de comparar los resultados obtenidos por los distintos clasificadores utilizados en este proyecto, se utiliza como métrica la exhaustividad (recall-score) antes que la precisión del modelo. Esto es: se desea que todos los casos fraudulentos sean etiquetados como tal, aún si ello implica etiquetar colateralmente como intento de estafa operaciones lícitas realizadas por los compradores. Se da prioridad a la minimización de los falsos negativos, afectando así al número de falsos positivos obtenidos como resultado de la clasificación.

# Time Number of seconds elapsed between this transaction and the first transaction in the dataset	# V1 may be result of a PCA Dimensionality reduction to protect user identities and sensitive features(v1-v28)	# V2	# V3	# V4
0 … 173k	-56.4 … 2.45	-72.7 … 22.1	-48.3 … 9.38	-5.68 … 16.9
0	-1.3598071336738	-0.0727811733098497	2.53634673796914	1.37815522427443
0	1.19185711131486	0.26615071205963	0.16648011335321	0.448154078460911
1	-1.35835406159823	-1.34016307473609	1.77320934263119	0.379779593034328
1	-0.966271711572087	-0.185226008082898	1.79299333957872	-0.863291275036453
2	-1.15823309349523	0.877736754848451	1.548717846511	0.403033933955121
2	-0.425965884412454	0.960523044882985	1.141109342232219	-0.168252079760302
4	1.22965763450793	0.141003507049326	0.0453707735899449	1.20261273673594
7	-0.644269442348146	1.41796354547385	1.0743803763556	-0.492199018495015

2.2 Preparación de los datos

Antes de comenzar a trabajar con los datos, debemos someterlos a una serie de procesos para facilitar su tratamiento.

Eliminación de duplicados

Consiste en la eliminación de entradas duplicadas en nuestra tabla. Dichos datos repetidos no aportan información útil y solo añaden tiempo de cómputo al estudio. Esta transformación logró reducir el conjunto de datos de 284,807 a 283,726 entradas, pero la proporción de las clases se mantuvo, así que seguimos teniendo un 99.83% de transacciones no fraudulentas.

Escalado

Debido a que la mayor parte de las variables son transformaciones numéricas de los valores reales, los atributos que conservan las cifras originales (*Time* y *Amount*) están en magnitudes diferentes al resto, por lo que necesitamos escalar los datos para que todos compartan las mismas proporciones. Para ello, utilizamos el método *RobustScaler* de la librería *sklearn*, el cual es robusto a elementos aislados (*outliers*).

Creación de subconjuntos de entrenamiento y pruebas

Tras estas modificaciones, se ha dividido el conjunto de datos aleatoriamente en 2 diferentes: un subconjunto de entrenamiento (70%) y uno de pruebas (30%). Es de vital importancia realizarlo antes de balancear los datos, puesto que queremos mantener el mayor número de entradas posibles para el conjunto de pruebas, y asegurarnos de que nuestro clasificador funciona correctamente. Las siguientes técnicas (balance y eliminación de *outliers*) se harán únicamente sobre el conjunto de entrenamiento.

Balance del conjunto de datos

Previamente se ha descrito que el conjunto de los datos está formado por 284,807 transacciones bancarias, de las cuales 492 son fraudulentas. A la hora de crear el subconjunto de entrenamiento, esta característica incurre en un problema de tendencia a clasificar todas las transacciones como la clase más frecuente. Con el fin de evitar esta situación, se ha procedido a balancear los datos, procurando que todas las clases aparezcan aproximadamente el mismo número de veces. Estas son algunas de las técnicas de balance que se ha empleado:

- Upsampling: consiste en aumentar el número de muestras de la clase menos frecuente, duplicando sus entradas hasta que no exista desbalance de datos. La desventaja de esta solución es el alto coste computacional, ya que en este caso estaríamos añadiendo aproximadamente 284,000 entradas a nuestros datos. No obstante, estamos manteniendo toda la información de la clase mayoritaria.

- Downsampling aleatorio: consiste en reducir de manera aleatoria el número de muestras de la clase predominante hasta que iguale a la menos frecuente. Este método es más asequible computacionalmente pero con una gran desventaja referida a la pérdida de información. Para este conjunto de datos se estaría perdiendo más del 99,9% de las entradas de transacciones de la clase 0.

- Downsampling usando centroides de clústeres: permite agrupar datos similares dentro de un mismo conjunto de datos. En este caso, se emplea el clustering como una técnica de reducción de la dimensionalidad (downsampling), ya que se pretende aislar los casos fraudulentos en un conjunto de clústeres y, dentro de los agrupamientos restantes, elegir un representante de clase de cada grupo; esto es, los centroides de cada uno de los mismos. Esto permite preservar las entradas fraudulentas mientras se reduce efectivamente y de forma representativa el número de casos contrario.

Eliminación de outliers (elementos aislados)

Después de aplicar el escalado, se procede a eliminar los elementos aislados u *outliers*. Esto es de vital importancia puesto que apreciamos que hay valores como la cantidad (*scaled_amount*) que tienen una media de 0.92 pero su valor máximo es de 356.96. Esto podría tener un impacto en nuestro estudio, ya que una persona que realizó una transacción de una cantidad considerable está afectando a la media, y la eliminación de estas transacciones especiales podría contribuir a obtener mejores resultados.

Selección de variables

Por último, probamos a eliminar aquellas variables (columnas) cuyo valor absoluto de la correlación respecto a la variable de clase sea menor a una determinada cantidad, escogiendo en este caso 0.1, puesto que pensamos que podrían ser variables poco relevantes. Cogiendo como ejemplo la base de datos a la que se aplicó downsampling aleatorio, pudimos eliminar 4 variables. En los análisis posteriores veremos el efecto de quitar estas variables a la hora de clasificar, puesto que se mantuvo la base de datos sin las variables eliminadas para comparar resultados y ver si realmente supone una mejora.

3. Clasificación

Como modelos para clasificar se usaron los siguientes:

- Árbol de decisión
- Clasificador de bosques aleatorios
- Regresión logística
- Clasificador bayesiano
- Perceptrón simple
- MLP (Perceptrón multicapa)
- SOM (Self-Organizing Map)

Todos los clasificadores, a excepción de los últimos 3, fueron entrenados con los siguientes conjuntos de datos:

- Base de datos escalada
- Base de datos con undersampling
- Base de datos con undersampling y eliminando variables
- Base de datos con upsampling
- Base de datos con upsampling y eliminando variables
- Base de datos con downsampling utilizando cluster centroids

Tras ser entrenados con estas bases de datos, las puntuaciones fueron obtenidas con el conjunto de test **original** sin balancear. En el caso de probar con un clasificador que fue entrenado con variables eliminadas, también se eliminan esas variables para el conjunto de pruebas para que así sean idénticos los conjuntos de datos respecto a la columnas.

3.1 Árbol de clasificación

Conjunto de datos	Resultado
Base de datos escalada	recall: 0.704 | precision: 0.83 | F1: 0.761
Undersampling	recall: 0.856 | precision: 0.023 | F1: 0.046
Undersampling y eliminando variables	recall: 0.848 | precision: 0.019 | F1: 0.037
Upsampling	recall: 0.776 | precision: 0.026 | F1: 0.05
Upsampling y eliminando variables	recall: 0.784 | precision: 0.026 | F1: 0.051
Downsampling usando Cluster Centroids	recall: 0.864 | precision: 0.004 | F1: 0.009

Como mencionamos en apartados anteriores, para comparar los resultados de los distintos clasificadores, nos centraremos en el recall-score (exhaustividad). Si observamos los resultados de los árboles de clasificación, el mayor recall lo tiene el 'Downsampling usando Cluster Centroids' pero la precisión en comparación con los otros árboles es demasiado baja; por ello, diremos que el árbol construido con 'Undersampling' da el mejor resultado. Cabe mencionar que ese mismo conjunto pero con variables eliminadas solamente pierde 0.01 puntos de recall así que, si buscamos trabajar con menos variables porque es más cómodo, podemos sacrificar esa puntuación de exhaustividad.

3.2 Clasificador de bosques aleatorios

Conjunto de datos	Resultado
Base de datos escalada	recall: 0.728 | precision: 0.928 | F1: 0.816
Undersampling	recall: 0.8 | precision: 0.041 | F1: 0.079
Undersampling y eliminando variables	recall: 0.848 | precision: 0.034 | F1: 0.066
Upsampling	recall: 0.72 | precision: 0.947 | F1: 0.818
Upsampling y eliminando variables	recall: 0.688 | precision: 0.905 | F1: 0.781
Downsampling usando Cluster Centroids	recall: 0.864 | precision: 0.007 | F1: 0.014

Este modelo debería ser una mejora del árbol de decisión. Esto es porque el clasificador de bosques aleatorios combina varios árboles y obtiene el resultado de la unión de estos; sin embargo, como vemos en los resultados, no se obtuvo ninguna mejora en comparación con los resultados del árbol de decisión. El conjunto de 'Undersampling' sigue siendo el mejor, pero esta vez eliminar variables poco relevantes da un claro mejor resultado. Aún así, al comparar con los resultados previos obtenemos que el 85.6% de exhaustividad del 'Undersampling' en el árbol de decisión sigue siendo el mejor resultado hasta el momento.

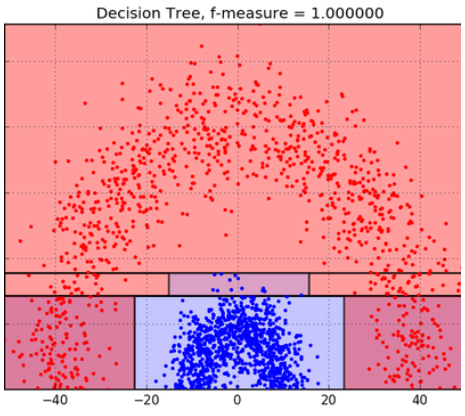
3.3 Regresión logística

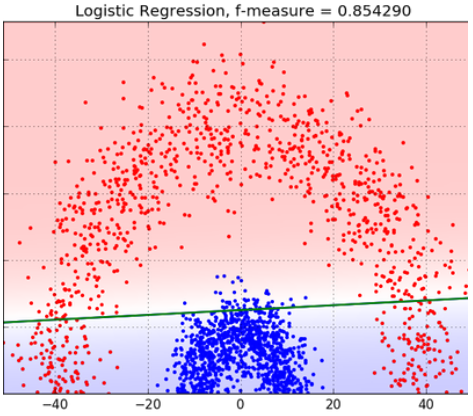
Conjunto de datos	Resultado
Base de datos escalada	recall: 0.608 | precision: 0.853 | F1: 0.71
Undersampling	recall: 0.864 | precision: 0.047 | F1: 0.089
Undersampling y eliminando variables	recall: 0.864 | precision: 0.028 | F1: 0.055
Upsampling	recall: 0.896 | precision: 0.053 | F1: 0.101
Upsampling y eliminando variables	recall: 0.872 | precision: 0.051 | F1: 0.096
Downsampling usando Cluster Centroids	recall: 0.88 | precision: 0.072| F1: 0.134

Este clasificador da los mejores resultados hasta el momento. Algo a destacar es que ha dado los valores más altos con el conjunto Upsampling, y no el de 'Downsampling', como los anteriores. Viendo la tabla comprobamos que cuando entrenamos el modelo con el mismo conjunto pero eliminando variables, se sigue perdiendo un ~2% de recall como se vio previamente en el árbol de decisión.

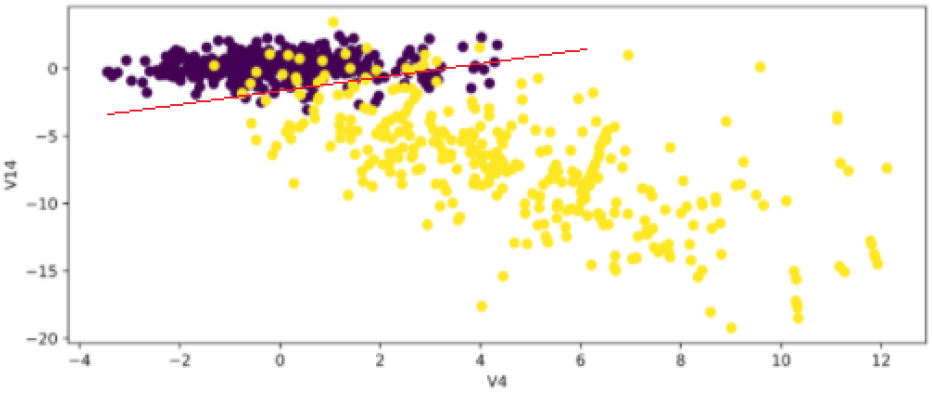
¿Podemos sacar alguna conclusión de estos resultados?

Los árboles de decisión tienen muy buen rendimiento si los valores de la variable de clase en la base de datos están bien separados cuando son representados en el espacio. Por el contrario, la regresión logística es mejor si no están bien separados; pondremos un ejemplo:



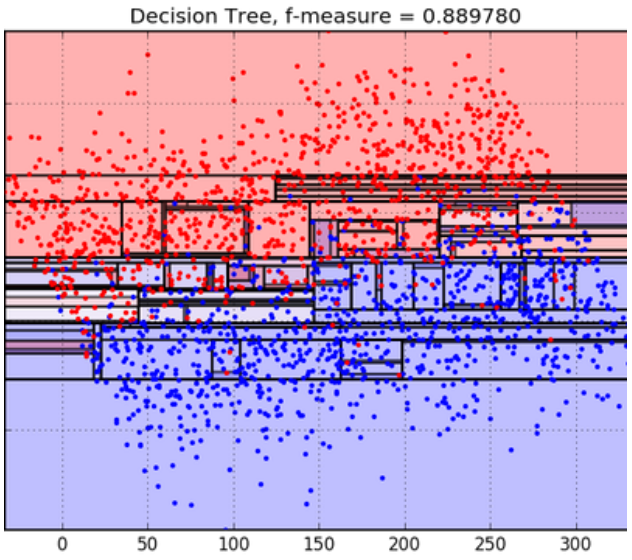


Aplicando estos conocimientos, vamos a ver cuánta separación hay entre los distintos valores de la variable de clase de nuestro dataset:



Según esta gráfica hay una sección donde los datos no están separados claramente, y es por ello que la regresión logística tiene un buen rendimiento, pero ¿por qué el árbol decisión no?

Esto se debe a un problema denominado *overfitting*. El árbol está realizando el aprendizaje con el conjunto de entrenamiento y está sobreajustando las clasificaciones al mismo; esto provoca que, cuando se recibe una transacción perteneciente al conjunto de prueba, no sea bien clasificada con una alta probabilidad, puesto que esta no se corresponde con ninguno de los grupos previamente creados.



3.4 Clasificador bayesiano

Conjunto de datos	Resultado
Base de datos escalada	recall: 0.76 | precision: 0.048 | F1: 0.09
Undersampling	recall: 0.808 | precision: 0.027 | F1: 0.052
Undersampling y eliminando variables	recall: 0.84 | precision: 0.019 | F1: 0.037
Upsampling	recall: 0.808 | precision: 0.032 | F1: 0.062
Upsampling y eliminando variables	recall: 0.808 | precision: 0.034 | F1: 0.065
Downsampling usando Cluster Centroids	recall: 0.768 | precision: 0.185 | F1: 0.298

El mejor conjunto para este modelo es el 'Undersampling' igual que en los anteriores, excepto en la regresión logística; pero, aún así, no obtenemos ninguna mejora.

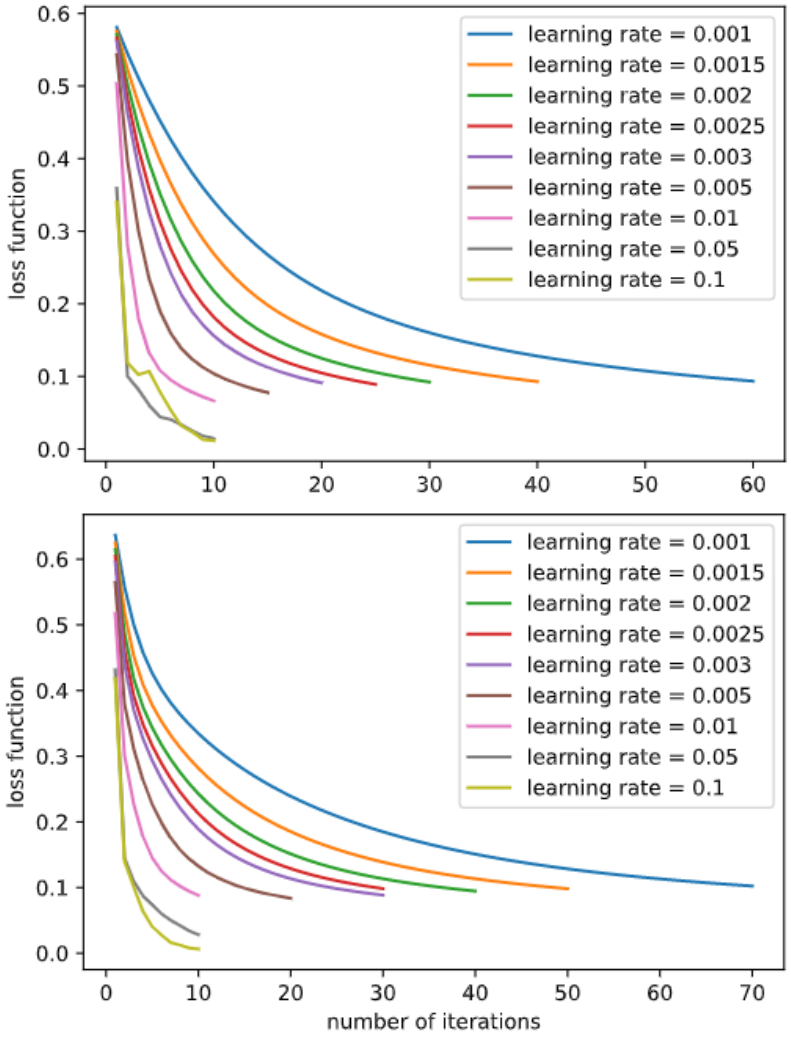
3.5 Perceptrón Multicapa (MLP)

Para entrenar los modelos pertenecientes a redes neuronales, creamos otro subconjunto que deriva del conjunto de entrenamiento ya creado, al que denominaremos conjunto de validación. En este caso concreto, usaremos el conjunto de datos 'Downsampling y con variables eliminadas', ya que por temas de complejidad computacional era más viable trabajar con un conjunto de aproximadamente 800 entradas en vez de 400,000 entradas.

Antes de comenzar con la construcción de un perceptrón multicapa creamos un perceptrón simple, el cual es entrenado con un límite de iteraciones máximas de 30. Este límite fue escogido de manera arbitraria. Tras este entrenamiento obtenemos un recall de 0.82 puntos.

Una vez creado el perceptrón simple, el siguiente paso es crear el Perceptrón Multicapa (MLP). Un MLP es una red neuronal más compleja que el perceptrón simple porque está formada por varias capas. El primer MLP tiene un número de capas ocultas de 60, número escogido arbitrariamente. Tras introducir el conjunto de prueba a este modelo obtenemos un 0.86 de recall, es una mejora pero el modelo todavía no está optimizado.

Para llevar a cabo esta mejora, se probó con dos funciones de activación: 'relu' y 'tangente hiperbólica'. El primer paso para comenzar a mejorar nuestro MLP fue probar cual es el mejor ratio de aprendizaje (*learning rate*) según la función de activación que queramos probar. Estos fueron los resultados para ambas funciones:



relu

tanh

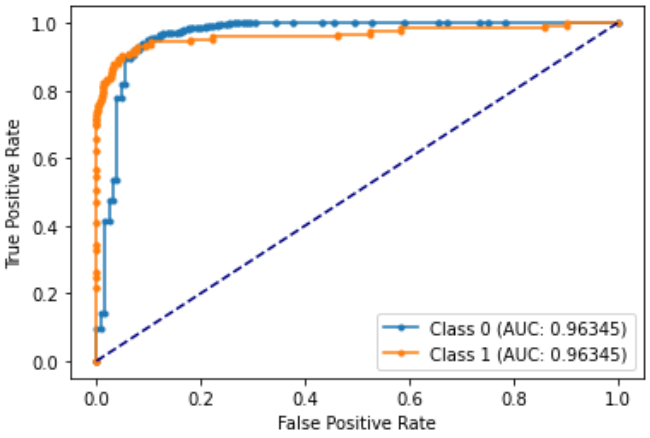
Los datos que nos proporcionan estas gráficas son que el mejor ratio de aprendizaje para la función de activación ‘relu’ es de 0.01 y para la función ‘tanh’ es 0.05.

Una vez hayamos obtenido el ratio de aprendizaje óptimo, buscaremos cuál es el número de capas ocultas que mejor se ajuste a nuestro conjunto de datos. Para ello iteramos sobre un vector con los números de capas ocultas que queramos probar; en este estudio, trabajamos sobre los siguientes valores: [5, 10, 20, 30, 50, 70, 90, 110, 130, 150, 170, 190]. Iremos probando con todos los números del vector y comparando resultados hasta obtener el mejor. El número de capas ocultas óptimo para la función ‘relu’ fue de 10 capas ocultas y para la función ‘tanh’ fue de 90 capas ocultas. Los resultados de ambas funciones son los siguientes:

- Relu:
    - Número de capas ocultas: 10
    - Learning rate: 0.01
    - Resultados:
        - Precision: 0.03
        - Recall: 0.89
        - F1: 0.04
- Tanh:
    - Número de capas ocultas: 90
    - Learning rate: 0.05
    - Resultados:
        - Precision: 0.02
        - Recall: 0.90
        - F1: 0.06

Analizando estos resultados, concluimos que hemos obtenido un modelo de más del 0.9 de recall, el cual rompe la barrera entre 0.8 y 0.89 en la que se encuentran el resto de modelos.

Por último, para visualizar los resultados de nuestro MLP, crearemos su curva ROC correspondiente.



En esta gráfica vemos que para el valor 1 de variable de clase, la curva se acerca bastante a la esquina superior izquierda por lo que concluimos con que da unos buenos resultados.

3.6 Mapa autoorganizado (SOM)

Aunque los mapas autoorganizados son utilizados frecuentemente para la realización de tareas de agrupamiento, en este proyecto se han utilizado a modo de clasificador. Con el fin de simular una red Counterpropagation, se ha concatenado la salida de la clasificación realizada por el SOM a un Perceptrón simple, permitiendo obtener resultados similares a los de otras redes más complejas con un menor tiempo de cómputo.

3.6.1 Modelo sin optimizar

Como primera toma de contacto al trabajo con este tipo de redes neuronales, se ha entrenado un modelo con parámetros de entrenamiento ajustados arbitrariamente. Puesto que estos ajustes, como el tamaño de la malla a utilizar, tienen un impacto directo en la efectividad del modelo obtenido tras el entrenamiento; se espera que esta primera aproximación presente un peor comportamiento dentro de las métricas de evaluación establecidas respecto a otros modelos mostrados hasta el momento en esta memoria.

- Resultados de la clasificación del conjunto de entrenamiento:

	Precision	**Recall**	**F1-score**
0	0.79	1.00	0.88
1	1.00	**0.65**	0.79

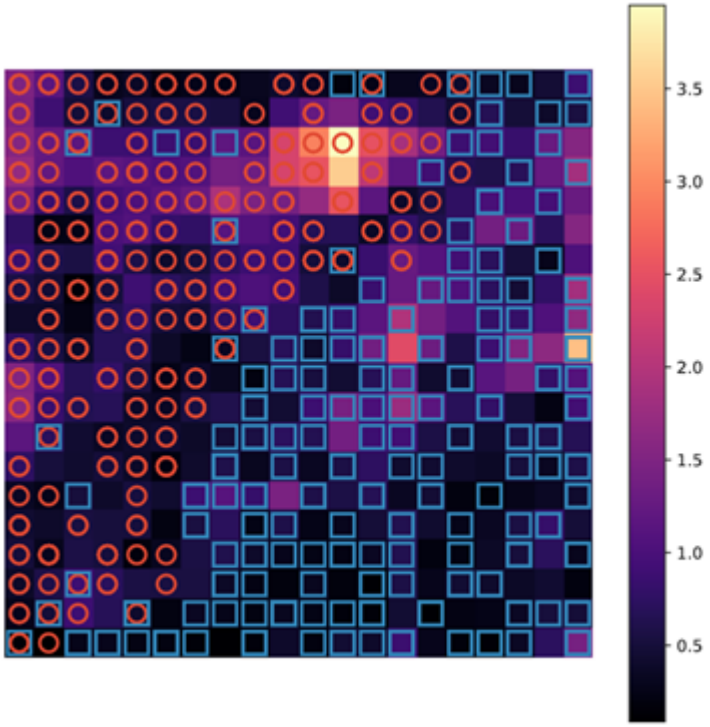
- Resultados de la clasificación del conjunto de validación:

	Precision	**Recall**	**F1-score**
0	0.71	0.99	0.83
1	0.98	**0.50**	0.66

Vemos que, tal y como se esperaba, este primer acercamiento al uso de este modelo nos ha proporcionado resultados bastantes alejados de los que se han obtenido previamente con modelos más sencillos, como árboles de clasificación. Un clasificador que descarta la mitad de los casos fraudulentos no es viable para los fines de este proyecto, pues se seguiría perdiendo una cantidad considerable de dinero debido a estafas con tarjetas de crédito.

Para la representación visual de los resultados obtenidos, se ha utilizado un mapa de calor o *heatmap* donde se representa cada una de las neuronas que forman la red utilizada a través de la distancia media a sus vecinas. Dentro de cada uno de los microclústeres producidos, se muestra la variable de clase más común dentro de la agrupación: los círculos representan casos no fraudulentos; los cuadrados, casos fraudulentos. Algunas casillas se encuentran vacías, indicando que existen separaciones entre los puntos de información; otras, se encuentran marcadas por ambos símbolos, representando que en estos agrupamientos ambas variables de clase se encuentran altamente presentes.

Se puede observar en la siguiente gráfica el trabajo realizado por el mapa autoorganizado para la separación de las variables de clase, las cuales prácticamente podrían ser separadas por una diagonal que atravesase el mapa:



3.6.2 Modelo optimizado

Con el fin de mejorar los resultados ilustrados en el anterior apartado, se ha llevado a cabo un proceso de optimización de la arquitectura: al ajustar el proceso de aprendizaje al problema que se está abordando, cambiando parámetros de entrenamiento como el tamaño de la rejilla, se espera obtener un modelo mejor adaptado a los datos del problema y más representativo del mismo.

Las pruebas realizadas comprueban la puntuación obtenida sobre el conjunto de validación utilizando distintos tamaños de rejillas, a elegir entre 2x2, 3x3, 4x4, 5x5, 6x6, 7x7, 10x10, 15x15 y 20x20. La mejor ejecución obtenida corresponde con una malla de tamaño

10x10, con una fidelidad del 95% sobre el conjunto de entrenamiento y de 93.8% sobre el conjunto de validación:

- Resultados de la clasificación del conjunto de entrenamiento:

	Precision	**Recall**	**F1-score**
0	0.95	0.98	0.96
1	0.98	**0.93**	0.95

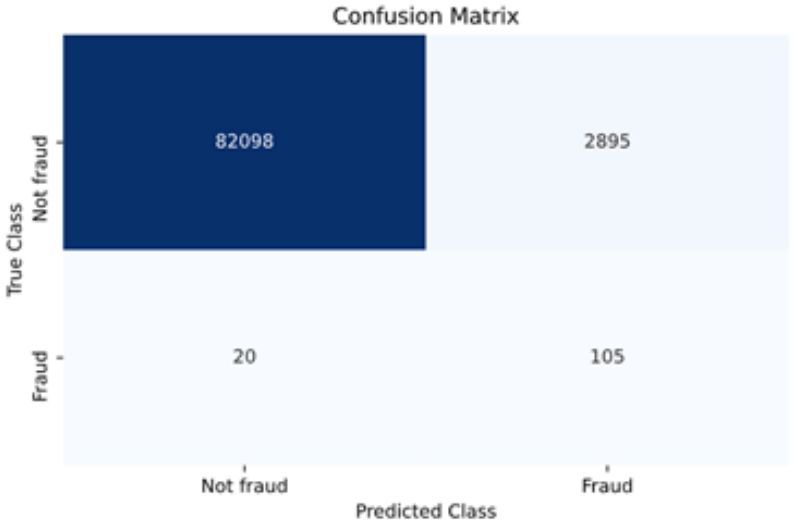
- Resultados de la clasificación del conjunto de validación:

	Precision	**Recall**	**F1-score**
0	0.91	0.9	0.95
1	0.98	**0.88**	0.93

- Resultados de la clasificación del conjunto de pruebas:

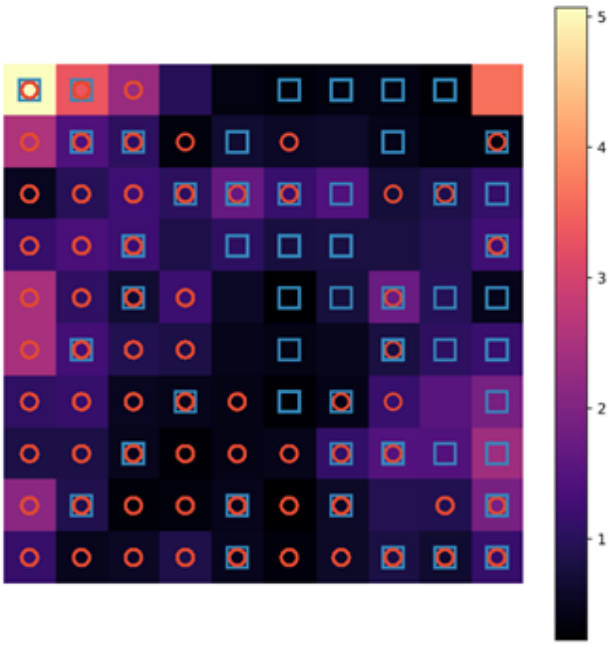
	Precision	**Recall**	**F1-score**
0	1.00	0.97	0.98
1	0.04	**0.84**	0.07

Vemos que los resultados obtenidos por este modelo son mucho mejores que los que se había computado originalmente sin la realización de optimizaciones; si bien no corresponde con la mejor puntuación de exhaustividad vista de entre todos los modelos de clasificación estudiados. Por otra parte, cabe destacar una gran bajada en la precisión del clasificador, produciendo que un considerable número de casos no fraudulentos sean etiquetados erróneamente como fraudulentos:



Como se ha explicado anteriormente, este es también un comportamiento esperado dentro del conjunto de datos con el que se está trabajando; puesto la minimización del número de falsos negativos predichos por el modelo conlleva el aumento de falsos positivos.

Podemos nuevamente visualizar los agrupamientos realizados por la red SOM utilizando un heatmap. Vemos que, al reducir el tamaño de la malla, la división entre las variables de clase es algo más confusa; si bien el modelo tiende a separar ambas en distintas regiones del espacio:



4. Agrupamiento

4.1 Agrupamiento basado en prototipos

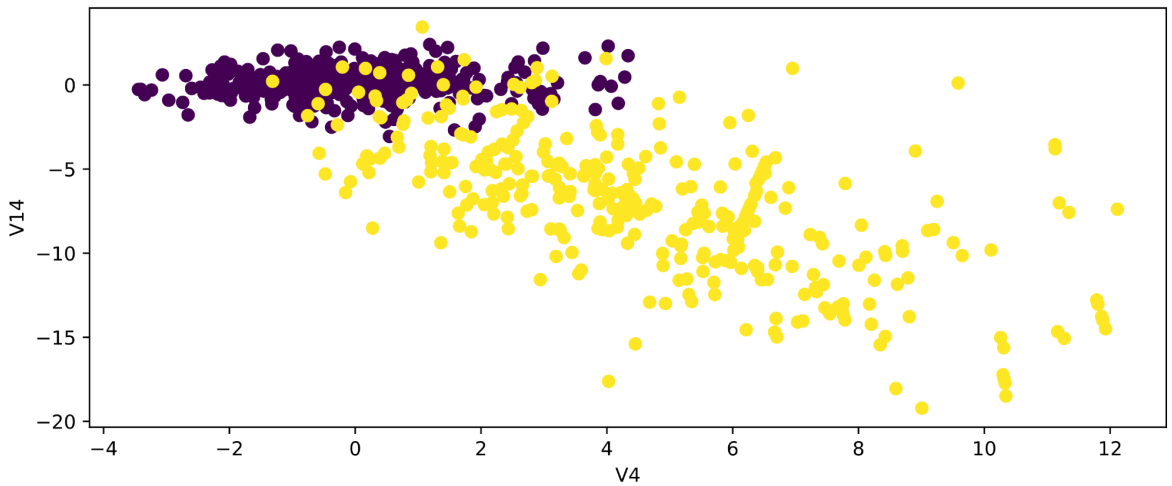
Una de las preguntas que nos hicimos para abordar el agrupamiento era si, tras agrupar el conjunto de datos en distintos clústeres, había algún grupo predominantemente fraudulento. Es por ello que se aplicó el algoritmo K-medias a la base de datos sin la variable de clase y, tras obtener un determinado número de grupos, comprobamos el porcentaje de elementos pertenecientes a la clase 1 (fraude). Probando con un número máximo de 20 clústeres vimos que una buena elección podría ser escoger 16 clústeres, así que procedimos a realizar la experimentación con ese número de grupos y obtuvimos los siguientes porcentajes:

Grupo	Porcentaje de fraude
0	0.04 %
1	0.31 %
2	0.10 %
3	0.29 %
4	0.00 %
5	80.49 %
6	0.02 %
7	0.05 %
8	0.01 %
9	0.18 %

10	0.00 %
11	0.68 %
12	0.42 %
13	0.24 %
14	0.00 %
15	0.17 %

Observamos que, efectivamente, el grupo 5 tiene concentrado un 80.49% de los fraudes, por lo que podemos concluir que, al agrupar los datos, hay un grupo predominantemente fraudulento.

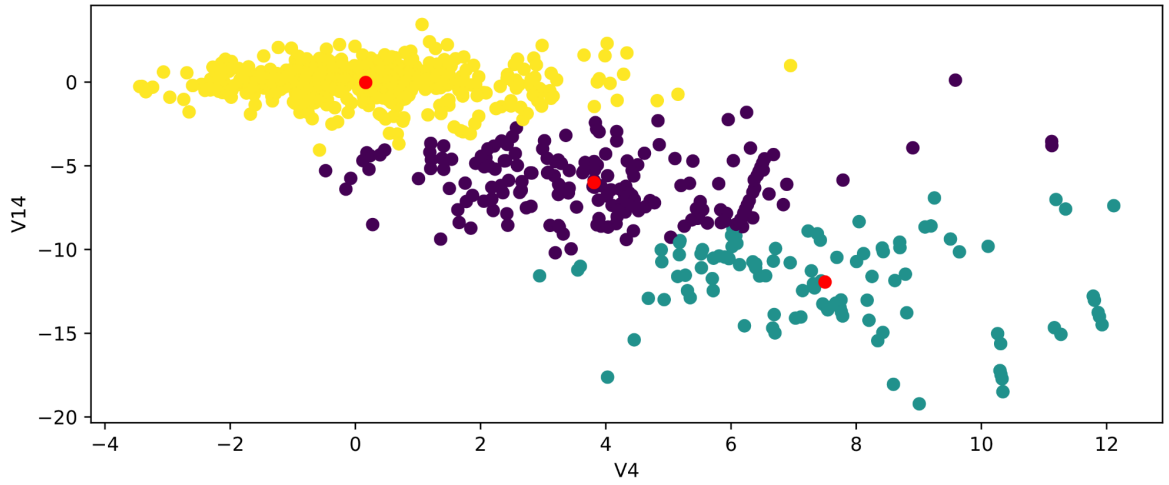
Con el fin de hacer un agrupamiento más visual y fácil de interpretar, decidimos escoger las dos variables cuyo valor absoluto de correlación respecto a la variable de clase sea mayor, y los representamos en la siguiente gráfica:



Amarillo: fraude Morado: no fraude

Cabe destacar que el conjunto de datos utilizado es el de entrenamiento con downsampling aleatorio aplicado; puesto que no era muy gráfico representar las 284,807 entradas originales. Vemos que hay cierta diferencia entre las clases, aunque quizás el resto de variables podrían ayudar a identificar aún mejor la diferencia entre los elementos que se solapan en una misma zona.

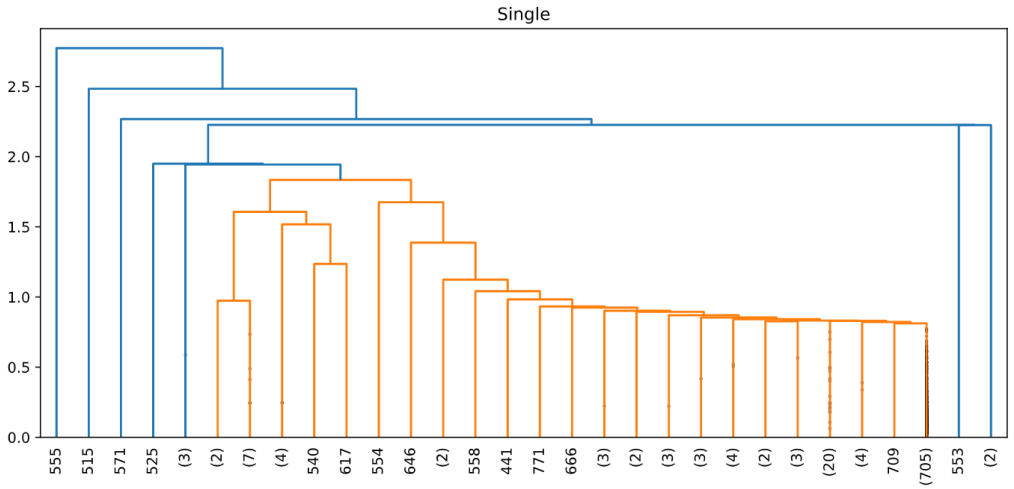
Si realizamos el agrupamiento basado en prototipos de K-medias obtenemos los siguientes clústeres:

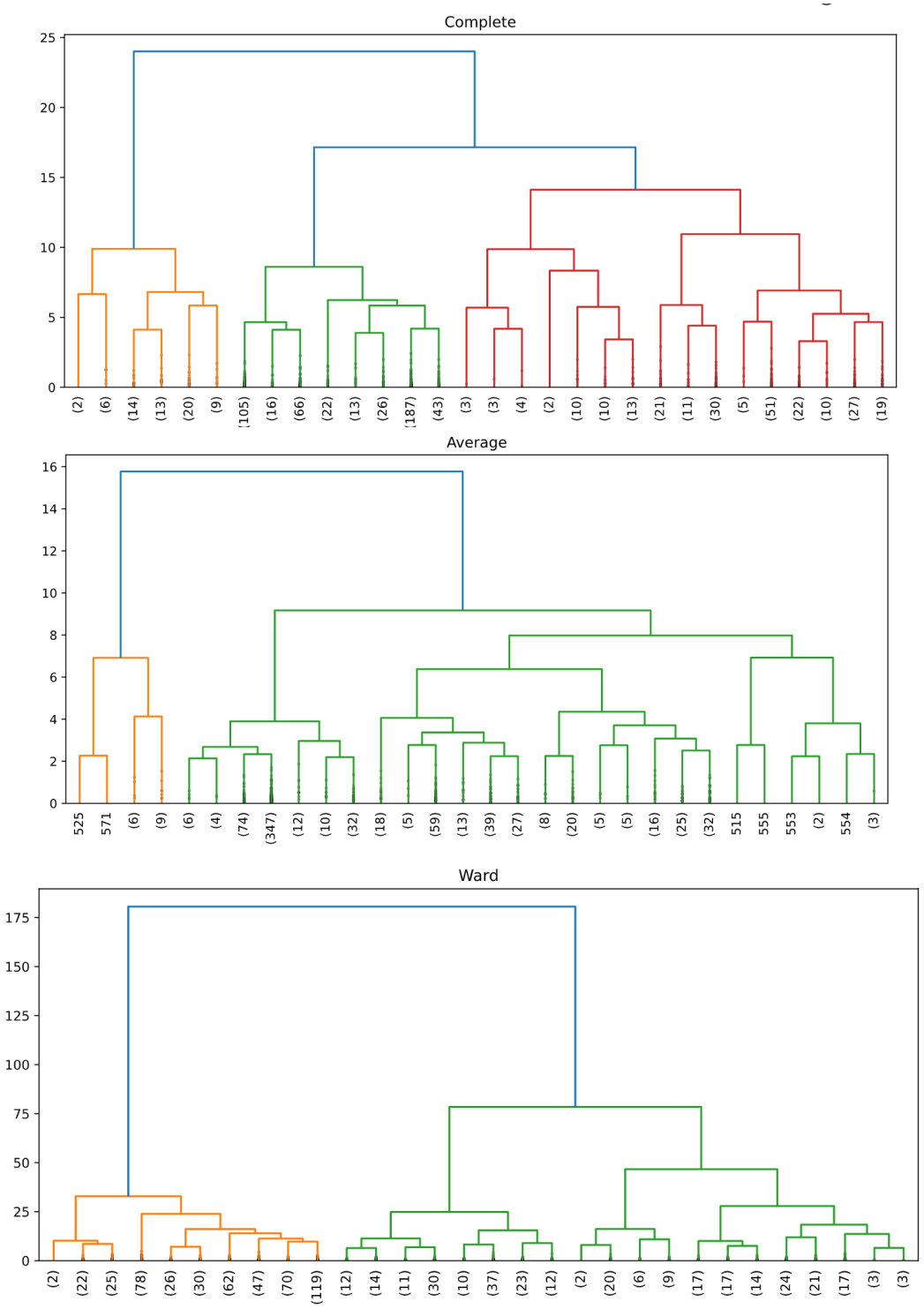


Podemos intuir que los agrupamientos azul y morado pertenecen a las transacciones fraudulentas, y el agrupamiento amarillo tiene una gran mayoría de transacciones no fraudulentas; por lo que teniendo en cuenta únicamente las variables V4 y V14, podríamos clasificar una transacción como fraudulenta.

4.2 Agrupamiento jerárquico

En el caso del agrupamiento jerárquico, de nuevo escogimos las variables V4 y V14 del conjunto de entrenamiento con el downsampling aplicado; y probamos distintos métodos para comprobar si podíamos llegar a unas conclusiones satisfactorias. Se utilizó un método aglomerativo y obtuvimos los siguientes dendrogramas, empleando distintas medidas de proximidad (enlace simple, enlace completo, promedio del grupo y método de Ward) y la distancia euclídea entre puntos:

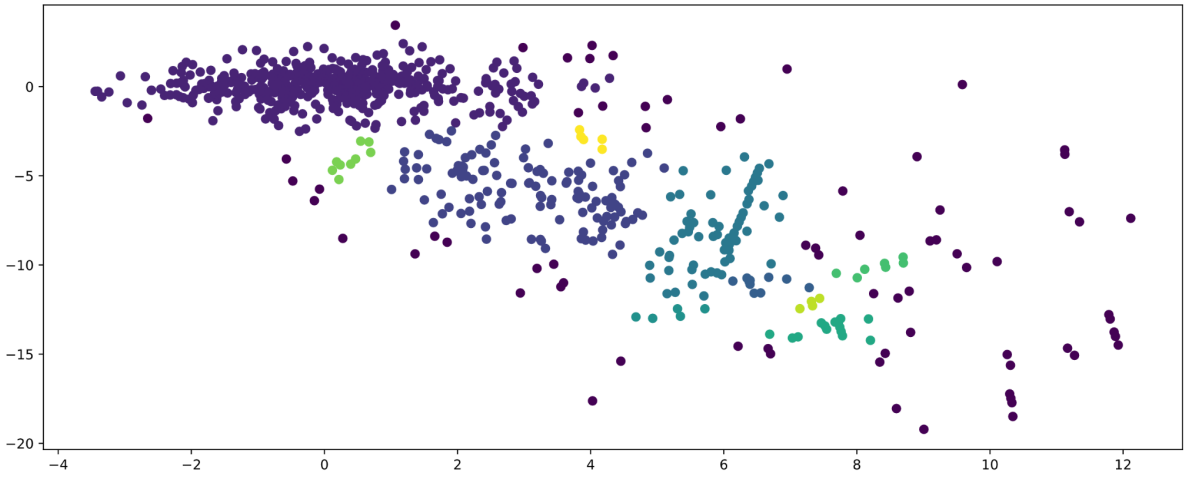




Además, se ha aplicado un truncamiento para obtener como máximo 30 clústeres iniciales con el fin de obtener dendrogramas legibles. Con el método de enlace completo obtenemos tres clústeres (igual que con el algoritmo K-medias), pero no consideramos que un agrupamiento jerárquico nos aporte información relevante para el análisis de esta base de datos, ya que la relación de proximidad entre los elementos no es relevante para este análisis.

4.3 Agrupamiento basado en densidad

Por último, aplicamos el algoritmo DBSCAN para realizar un agrupamiento basado en densidad; pero consideramos que en este ejemplo concreto no nos ayuda a diferenciar las clases, especialmente por la dispersión de los puntos de la clase fraudulenta. Además, previamente vimos que la densidad no parecía ser un factor importante a la hora de diferenciar las clases, ya que el clúster más denso (morado) contenía puntos tanto de una clase como de otra.



5. Conclusiones

Como hemos visto, sí que es posible predecir si una transacción es o no fraudulenta, con un elevado grado de certeza. También se ha conseguido encontrar características comunes a todos los fraudes conocidos, permitiéndonos usar dicho conocimiento para nuestras clasificaciones.

Podemos ver cómo conocer el conjunto de datos con el que se está trabajando es de vital importancia durante la realización de un análisis como el que se ha realizado en este proyecto. En este caso, el conjunto altamente desbalanceado de transacciones fraudulentas registradas dificulta el proceso de aprendizaje; puesto que algunos modelos aquí estudiados tienden a clasificar todas las entradas como la variable de clase más frecuente; en este caso, clasificaría todas las transacciones como no fraudulentas. Para contrarrestar este inconveniente, como hemos empleado en este proyecto, se utilizan técnicas como el oversampling o el downsampling; bajo el inconveniente de perder detalle y precisión en datos que no se encuentran incluidos dentro del conjunto seleccionado para el entrenamiento del modelo.

La respuesta a la pregunta de si es viable desplegar el sistema desarrollado en un entorno real depende en gran parte del protocolo de actuación ante una posible transacción fraudulenta. Supongamos que se alerta a los usuarios con un SMS informando de que se ha detectado un uso sospechoso en su tarjeta. En este caso, nuestro sistema podría ser completamente viable. Los modelos que mejores resultados nos han dado son el Perceptrón Multicapa, destacando que tiene un 0.9 de exhaustividad y 0.02 de precisión, y la Regresión Logística con un poco menos de exhaustividad, pero considerablemente más precisión. En un supuesto donde se reciben alertas por SMS, el Perceptrón Multicapa sería una mejor elección puesto que tiene mejor exhaustividad que la Regresión Logística y, para prevenir más fraudes, nos interesa que nuestro modelo clasifique más transacciones como fraudulentas; ya que no es tan grave mandar un SMS a un usuario erróneo, como ignorar a un usuario que ha sido víctima de una falsificación.

Finalmente, se ha realizado una comparación entre los resultados obtenidos con otros publicados en la plataforma Kaggle utilizando el mismo conjunto de datos. En general, un gran porcentaje de autores consigue obtener una exhaustividad similar o mejor a la que se ha obtenido con nuestro mejor modelo; como es el caso del investigador Jose Parreño *(https://www.kaggle.com/joparga3/in-depth-skewed-data-classif-93-recall-acc-now)*. Otros modelos, sin embargo, se encuentran considerablemente por debajo de los resultados obtenidos y descritos en este proyecto; como es el caso del usuario Renjith Madhavan *(https://www.kaggle.com/renjithmadhavan/credit-card-fraud-detection-using-python/comments#1071529)*, con una exhaustividad del 0.58 en su modelo de regresión logística.

6. Material consultado

- Shevchuk, Yurii. “Self-Organizing Maps and Applications.” NeuPy. Accessed June 1, 2021. http://neupy.com/2017/12/09/sofm_applications.html.

- “Credit Card Fraud Detection.” Kaggle, March 23, 2018. https://www.kaggle.com/mlg-ulb/creditcardfraud.

- Itdxer. “Itdxer/Neupy.” GitHub, February 5, 2019. https://github.com/itdxer/neupy/blob/master/examples/competitive/sofm_heatmap_visualization.py.

- “Sklearn.” scikit. June 5, 2021. https://scikit-learn.org/stable/