Facultad de estudios estadísticos

Trabajo de Fin de Grado

El papel de la estadística en la detección del fraude bancario

María Teresa Villanueva Moreno

supervisado por

D. Enrique González Arangüena

Julio

Curso 2019-2020

Abstract

Este documento presenta un primer acercamiento a algunas técnicas estadísticas utilizadas para la detección del fraude bancario.

Aplicando modelos probabilísticos, se pretende etiquetar transacciones como fraudulentas o legítimas. Para este proceso de clasificación a menudo se intenta capturar patrones de fraude. Sin embargo, la constante innovación en las estrategias de engaño dificulta esta tarea. Por este motivo, se introducen también modelos de detección de cualquier comportamiento diferente, para examinarlos con detenimiento después.

Partiendo de una tabla que contiene información sobre miles de transacciones y de modelos de clasificación, ¿se podrá detectar ágilmente el fraude en futuras transacciones?

English Abstract

This document is an approach to some statistical techniques used to credit card fraud detection.

Using probabilistic models, transactions are supposed to be correctly classified into fraud transactions and non-fraud ones. This classification problem is often solved by capturing fraud patterns. However, the constant innovation in fraud strategies makes it difficult. Therefore, it is also introduced a model that proposes detecting any anomaly behaviour to check its nature later.

Taking thousands of detailed transactions and some classification models, will it be possible to detect credit card fraud quickly?

Índice

1. Introducción

La enorme cantidad de transacciones bancarias ha venido acompañada de un aumento significativo en el número de fraudes cometidos. Adicionalmente, las nuevas tecnologías han favorecido la constante innovación en las técnicas de estafa, provocando que muchos detectores de fraude queden pronto totalmente obsoletos. La profesionalización de los defraudadores y el incremento en el número de canales de acceso bancario han permitido el aumento de los tipos de fraude y del tamaño de las pérdidas que ocasionan. Este problema en expansión provoca pérdidas millonarias cada año y perjudica enormemente a los bancos. Al mismo tiempo, leyes europeas, como la normativa PSD2[1], obligan a las entidades bancarias a confirmar la autenticidad de una transacción antes de autorizarla, sancionándola si su sistema de detección no cumple ciertos estándares. Luego las entidades bancarias necesitan irremediablemente erradicar este problema y deben construir un sistema de detección de fraude fiable para ello.

Motivado por la necesidad mencionada, este documento propone varios modelos de clasificación[2] de Machine Learning como sistemas de detección del fraude bancario. El Machine Learning es una disciplina científica del ámbito de la inteligencia artificial, que crea sistemas que identifican patrones complejos en millones de datos a través de algoritmos. Se dice que son algoritmos que aprenden de forma autónoma con el tiempo, es decir, que se desarrollan y actualizan sin la intervención humana. Se distinguen fundamentalmente dos tipos de aprendizaje automático: el aprendizaje supervisado y el aprendizaje no supervisado.

Un modelo de clasificación de aprendizaje supervisado es aquel que se elabora con un grupo de datos ya clasificado. En este caso, se buscan patrones de comportamiento entre las acciones conocidas como fraudulentas para identificar otras en un futuro. Gracias al aprendizaje desarrollado con estos datos que sirven de ejemplo, se pueden hacer predicciones adecuadas de observaciones aún no etiquetadas.

Por otra parte, se encuentra el aprendizaje no supervisado, también llamado *Clustering*. En este caso, el conjunto de datos del que se dispone no está clasificado en categorías ni se conoce la estructura que posee. Se busca agrupar las observaciones en función de sus similitudes y perfilar así su forma.

Este documento presenta caminos alternativos desde el Machine Learning para distinguir el fraude en un conjunto de transacciones. Son modelos que alcanzan un gran resultado. No obstante, a menudo son catalogados como modelos de caja negra, es decir, modelos de alta complejidad en los que difícilmente se conoce con claridad qué hace el modelo. Pese a dicha limitación, los modelos de Machine Learning son altamente utilizados por las entidades bancarias. De hecho, aún continúan desarrollándose con esta finalidad.

[1] PSD2 (*Payment Services Directive*) es la segunda directiva europea que regula los servicios de pago realizados en Europa con la finalidad de impulsar la transparencia, la competencia y la innovación de los servicios de pago del sector financiero.

[2] Modelo estadístico en el que el resultado a predecir es una etiqueta discreta. En este caso, se recogen ciertos datos de una transacción y se pretende que el modelo la clasifique como legítima o como fraudulenta.

Para dar consistencia al estudio, se detalla cada uno de los modelos de clasificación antes de su aplicación a una tabla de datos. Los registros de la tabla de la que se dispone se corresponden con transacciones bancarias y una de las características recogidas indica si dicha transacción es fraudulenta o no. Este campo habilita el uso de modelos de clasificación basados en el aprendizaje supervisado, ya que etiqueta a los datos. La mayoría de los modelos que se van a construir siguen un aprendizaje supervisado.

Se plantea además un modelo de aprendizaje no supervisado válido para la detección del fraude. No pretende capturar un patrón de fraude, sino detectar cualquier anomalía. De ese modo, se clasifica como fraudulenta toda transacción que presenta un comportamiento muy distinto al de las demás transacciones.

La tabla en cuestión tiene casi 300.000 filas. Esto permite dividir la muestra en dos y diferenciar los datos con los que se crea el modelo de los datos con los que se valida. La submuestra de entrenamiento es la que se emplea para la elaboración del modelo. Por otro lado, está la submuestra test, que es menos numerosa y que queda apartada en todo el proceso de elaboración del modelo para poder probar con credibilidad su precisión después. Esta estrategia de división de la muestra es muy recurrente en la ciencia de los datos.

La tabla cuenta con la ventaja de estar ya limpia, es decir, no hay valores perdidos, duplicados o incorrectos. No obstante, sí será conveniente realizarle ciertas modificaciones para que algunos modelos consigan clasificar correctamente.

Los datos equilibrados son aquellos que cuentan con una buena representación de cada clase de la variable dependiente para la construcción del modelo de clasificación. En este caso, hay dos tipos de transacciones, las legítimas y las fraudulentas. Las acciones fraudulentas son mucho menos frecuentes que las legítimas, provocando que no estén lo suficientemente representadas. Luego la tabla no está equilibrada y, por consiguiente, la submuestra con la que se elabora el modelo tampoco lo estará. Este hecho puede ocasionar problemas de detección de la clase minoritaria en los modelos de aprendizaje supervisado. El algoritmo asume que las transacciones son legítimas en su mayoría y las acciones fraudulentas pueden no ser percibidas como tal. Una posible solución es tratar de garantizar el equilibrio entre las clases legítima y fraudulenta en la muestra de entrenamiento. Para ello, existen distintas propuestas de sobremuestreo y de submuestreo[3]. En concreto, es de gran interés la técnica SMOTE (Bowyer et al, 2002), que crea ejemplos sintéticos de la clase menos numerosa.

En este estudio, se comparan derivados del árbol de decisión, que es un modelo de Machine Learning que se expondrá detalladamente. Se prueba la eficacia de los modelos con la porción de la muestra que se mantenía apartada. Se clasifica la muestra test según cada uno y se elabora la matriz de confusión, que presenta las observaciones fraudulentas y legítimas clasificadas correcta e incorrectamente y es altamente ilustrativa. Además, existen medidas de rendimiento que facilitan la

[3]En el ámbito de la estadística, se refiere a un conjunto de técnicas utilizadas para aumentar o reducir el tamaño de una muestra.

comparación entre los modelos. Confrontados los modelos, se concluye cuál es el que ofrece un mejor resultado y, de esta manera, se propone una solución contrastada al problema del fraude bancario.

En definitiva, actualmente las entidades bancarias deben frenar el fraude si no quieren enfrentarse a pérdidas millonarias e incluso a problemas legales. Este trabajo quiere colaborar a la mejor detección del fraude bancario y, para ello, estudia posibles soluciones y hace una valoración de cuál de ellas sería la más efectiva.

2. Estado del arte

Desde 2006, se está ante una explosión del Machine Learning en la que numerosas empresas están transformando sus negocios hacia el dato e incorporando técnicas de inteligencia artificial en sus procesos, productos y servicios, para obtener ventajas sobre la competencia. Entre este conjunto de empresas, se encuentran las entidades bancarias siendo la construcción de modelos para la detección del fraude bancario una de las aplicaciones más populares entre ellas.

El uso del Machine Learning para la detección de acciones ilegítimas se remonta a más de dos décadas. Ya a finales del siglo XX, se publicaron trabajos que apoyaban el uso de las redes neuronales[4] para este cometido (Aleskerov, Freisleben y Rao, 1997). No obstante, no es hasta unos años más tarde, cuando se extiende el uso de la inteligencia artificial para la detección del fraude. Los estudiosos se ven incentivados por los bancos, que muestran gran interés en frenarlo a través de la detección automática. Han sido innumerables las técnicas aplicadas y los distintos enfoques empleados para la resolución del problema.

En su mayoría, los modelos siguen el denominado aprendizaje supervisado. Construyen sus algoritmos haciendo uso de transacciones bancarias de las que se conoce su condición legítima o fraudulenta, para tratar de acertar la naturaleza de las transacciones futuras. En cierto modo, estos modelos examinan el nuevo movimiento y deciden si se corresponde con una acción fraudulenta o no con base a las similitudes con transacciones bien conocidas.

En concreto, este documento desarrolla modelos construidos sobre los árboles de decisión, que siguen una técnica de aprendizaje supervisado que se desarrollará más adelante. Los árboles de decisión sobreajustan con facilidad. Para lidiar con este problema surge el random forest (Breiman, 2001), que construye varios árboles de decisión de manera independiente para su posterior puesta en común. Este modelo resultó ser muy eficaz y supuso una auténtica revolución para el Machine Learning. Por ello, se continuó estudiando y se propusieron extensiones de este. En este estudio se hace uso de la técnica extremely randomized trees (Ernst, Geurts y Wehenkel, 2006), basada en el random forest.

Estos modelos de detección a menudo se complementan. Se suele hablar de circunstancias en las que conviene aplicar un modelo frente a otro, y de las fortalezas y las debilidades de cada uno. Hay multitud de artículos publicados en este sentido. Por ejemplo, se compara la efectividad en la detección del fraude bancario de la conocida regresión logística con el innovador modelo random forest (Bhattacharyya et al, 2011). Los autores concluyen que el random forest presenta grandes ventajas frente a la regresión logística. Además, alertan de que la detección del fraude bancario por medio de estas técnicas se enfrenta a una limitación importante por el hecho de que los datos son no equilibrados.

Se trata de datos no equilibrados porque, con carácter general, las acciones

[4]Técnica de aprendizaje y procesamiento automático inspirado en la forma en la que funciona el sistema nervioso de las personas (Specht, 1997).

fraudulentas son poco frecuentes. Como se adelantaba, este hecho puede dificultar el proceso de aprendizaje del algoritmo. Por ello, es de utilidad tratar de alcanzar el equilibrio entre las clases. Para hacerlo, resulta de gran interés la técnica SMOTE (Bowyer et al, 2002).

Además del mencionado aprendizaje supervisado, también se usa el aprendizaje no supervisado, que parte de datos de los que no hay un conocimiento a priori sobre su naturaleza. Desde el comienzo, hay autores que apoyan su uso para la detección del fraude bancario. Su aplicación en este estudio se ve sostenida por el hecho de que las técnicas de fraude están innovando constantemente. Luego no basta con buscar similitudes con casos de fraude ya conocidos, sino que hay que ir más allá (Domingues et al, 2018). Así, se apuesta por la detección de anomalías en las transacciones bancarias frente a la búsqueda de patrones de comportamiento. Entre todos los algoritmos, destaca la técnica de detección de anomalías denominada isolation forest (Liu, Ting y Zhou, 2008), que se basa en el modelo random forest.

Como se adelantaba, es un tema muy actual, que tiene avances constantemente. Por lo tanto, interesa atender a artículos recientes. Khare y Sait (2018) apoyan el uso del aprendizaje supervisado para la detección de fraude. En concreto, resaltan las ventajas ofrecidas por el random forest. A su vez, se continua apoyando el uso del isolation forest para detectar cualquier comportamiento bancario extraño (Hyder y Sameena, 2019).

Por lo tanto, se aprecia como el Machine Learning ofrece distintos enfoques para solventar el problema del fraude en transacciones bancarias. Parece claro que los bancos apuestan por métodos estadísticos para frenar las pérdidas que el fraude bancario provoca. Es preciso mencionar que la gran mayoría de estudios se han realizado en los últimos años y que aún continúan en desarrollo.

3. Soporte teórico

El objetivo final de este documento es establecer una comparativa entre la aplicación de varios modelos estadísticos en la detección del fraude bancario. Para dar consistencia a este trabajo, en primer lugar, se presenta su soporte teórico. Los modelos corresponden al campo del Machine Learning. Por ello, se parte de una introducción a esta disciplina, que proporciona una visión global del algoritmo que siguen los modelos para su construcción.

3.1. Introducción al Machine Learning

Para realizar un modelo de clasificación, se toma un conjunto de datos. En este caso, es una tabla en la que cada registro corresponde a una transacción bancaria. Los campos son distintas características recogidas de cada transacción, como puede ser la cantidad de dinero implicada. Entre ellas, está la variable objetivo o *target Class*, que toma un valor u otro en función de si la transacción se corresponde con un fraude o no. Esta variable dicotómica es la que se quiere predecir en futuras transacciones.

Dicho conjunto de datos se emplea para crear un modelo de clasificación con el objetivo de catalogar una futura observación como legítima o fraudulenta, por medio de un aprendizaje supervisado o no supervisado.

En este trabajo, se introduce el aprendizaje no supervisado, cuya muestra inicial no cuenta con una clasificación conocida y agrupa la muestra en base a ciertas evidencias. En concreto, se presenta un modelo que separa la muestra en transacciones comunes y altamente atípicas, para etiquetar estas últimas como fraudulentas. Sin embargo, la mayoría de los modelos de clasificación de los que se hace uso en este documento sigue un aprendizaje supervisado. La creación del modelo de clasificación se establece con un conjunto de datos para los que se conoce la salida deseada. Estos modelos hacen un tratamiento particular de la muestra, que se presenta a continuación y que será detallado posteriormente.

Para la creación de modelos de clasificación es sumamente importante contar con una tabla de datos limpia y uniforme. No obstante, cuando se cuenta con una gran cantidad de datos ya etiquetados, antes de realizar cualquier manipulación, interesa dividir la muestra en dos: el conjunto de entrenamiento y el conjunto test. El conjunto de entrenamiento suele comprender del 60 al 80 % de la muestra inicial y se utiliza para la creación del modelo, que es probado después con el conjunto test. Nótese que este último conjunto es totalmente independiente a la creación del modelo, garantizando que se hace una correcta validación de este.

Dividida la muestra, se toma el conjunto de entrenamiento y, en primer lugar, se realiza un análisis profundo de los datos. Este paso tiene encomendado garantizar la limpieza y la uniformidad de los datos de la tabla. Suele ser costoso, pero es imprescindible que el algoritmo reciba la información correctamente para que dé predicciones de alta calidad y confianza.

Posteriormente, se construye el modelo de clasificación. Para ello, se elige un modelo de Machine Learning considerado adecuado para el problema y se define el valor de los hiperparámetros que lo forman. Los hiperparámetros adaptan el modelo a las necesidades específicas que se demandan en cada caso. Pueden tomarse distintos valores para los hiperparámetros y, a través de la validación cruzada[5], elegir el modelo con los valores para los que se predice mejor.

Finalmente, se calcula la eficacia real del modelo con el conjunto test, que es ajeno al modelo creado. Se hace uso del modelo construido y, como se conoce la variable *Class*, se comprueba cómo de bien predice, comparando la predicción del modelo con la realidad.

Se pasa a detallar la división y la preparación de la muestra en modelos de aprendizaje supervisado. Se asume que la tabla está ya limpia, sin valores perdidos, duplicados ni incorrectos.

3.2. División y preparación de la muestra

Se divide la muestra en dos, la mayor parte se toma para la creación del modelo y se reservará otra parte más pequeña para su validación. Se suele emplear el 70 % de la muestra para la creación del modelo y el 30 % restante para testearlo.

Como se ha mencionado, el conjunto de creación del modelo es claramente no equilibrado, puesto que el número de acciones fraudulentas es mucho menor que el de las acciones legítimas con carácter general. Este hecho puede ocasionar grandes problemas en modelos de aprendizaje supervisado. El algoritmo aprende que las transacciones son legítimas en su mayoría y corre el peligro de no ser capaz de percibir cuando sí que está ante un caso de fraude bancario. Dado el coste que este error supondría, se prefiere la detección por exceso, aunque luego se descarte.

Por ello, resulta conveniente que el modelo crea que una transacción fraudulenta no es tan excepcional. Se consigue aumentando la proporción de ejemplares de este tipo en la muestra de entrenamiento.

Hay que utilizar técnicas que consigan que el número de transacciones fraudulentas sea más parecido al número de transacciones legítimas. Para ello, se hace uso de los métodos de remuestreo, que son de gran ayuda. Se introducen técnicas de submuestreo y de sobremuestreo.

- **Submuestreo *(undersampling)***

  En este caso, se reduce el número de observaciones de la muestra mayoritaria para que sea igual al correspondiente número de observaciones de la muestra minoritaria.

[5]La validación cruzada divide k veces la muestra en dos conjuntos complementarios de datos, el conjunto de entrenamiento y el conjunto test. Para cada una de las k divisiones, crea el modelo en cuestión con el conjunto de entrenamiento y lo prueba con el test. Estima cada una de las medidas de rendimiento para el modelo como el promedio de dichas medidas en los k modelos.

Uno de los métodos más clásicos se denomina *Random Under-Sampling* (RUS), que selecciona de manera aleatoria elementos de la clase mayoritaria para eliminarlos hasta que ambas clases queden equilibradas.

Cabe mencionar que, en la medida de lo posible, se intenta esquivar el submuestreo, ya que implica la renuncia de parte de la información de la que se dispone, y no conviene despreciar datos.

- **Sobremuestreo *(oversampling)***

  El sobremuestro, por el contrario, pretende aumentar el número de observaciones de la muestra minoritaria para que sea igual al correspondiente número de observaciones de la muestra mayoritaria.

  Una manera simple de equilibrar la muestra, propuesta por el método *Random Over-Sampling* (ROS), consiste en realizar copias de elementos de la clase minoritaria. Así se llegaría a alcanzar el equilibrio entre las clases. Sin embargo, contar con ejemplares que aparecen varias veces eleva el riesgo del sobreajuste, luego no convence del todo.

  Entre las estrategias más conocidas de sobremuestreo se encuentra SMOTE (*Synthetic Minority Over-Sampling Technique*). Este algoritmo se aplica cuando las variables predictoras son continuas y propone la creación de nuevos ejemplos como sigue.

  Operando en el espacio de atributos (*feature space*), crea ejemplos sintéticos a lo largo de los segmentos que unen a cada elemento de la clase minoritaria con los k vecinos más cercanos de su misma clase. Se suele tomar $k = 5$.

  El algoritmo de SMOTE realiza los siguientes pasos:

  1. Recibe como parámetro el porcentaje de elementos a sobremuestrear.
  2. Calcula el número de nuevos elementos que tiene que generar.
  3. Calcula los k vecinos más cercanos de los elementos de la clase minoritaria. Genera los nuevos elementos siguiendo este proceso:
     - Para cada elemento de la clase minoritaria, elige aleatoriamente uno de los k vecinos más próximos.
     - Calcula la diferencia entre el vector de atributos del elemento a sobremuestrear y el del vecino elegido.
     - Multiplica este vector diferencia por un número aleatorio entre 0 y 1.
     - Suma este vector al vector de atributos del elemento original.
     - Devuelve el nuevo ejemplo sintético.

  SMOTE es el algoritmo más utilizado para el sobremuestreo y por el que se opta en este documento.

3.3. Construcción del modelo

Los modelos que se van a desarrollar parten de los CART, también conocidos como árboles de decisión. Se presentan estos en primer lugar y se les irán haciendo modificaciones hasta llegar a los demás. Hay modelos más complejos que otros. Sin embargo, todo modelo presenta ventajas e inconvenientes, lo que indica que a menudo se complementan.

3.3.1. CART (árbol de decisión)

El algoritmo supervisado CART (*Classification And Regression Trees*) fue creado por Breiman, Friedman y Olshen (1984) y se utiliza principalmente en problemas de clasificación. Desarrolla un modelo que clasifica elementos en función de la combinación de valores que toman sus variables.

Para la elaboración del árbol de decisión, es necesario contar con observaciones ya clasificadas, operando con ellas del siguiente modo:

Inicialmente, todos los elementos se encuentran en un único grupo, que es la raíz del árbol. Después, ese grupo de elementos se divide en otros dos grupos, en función del valor de una de las variables predictoras. Los grupos obtenidos, también llamados nodos, son a su vez divididos iterativamente utilizando la variable y la condición adecuada. Por defecto, el algoritmo finaliza cuando los nodos terminales son homogéneos, en el sentido de que sus observaciones pertenecen a una sola clase.

La Fig. 1 proporciona un ejemplo en el que se tienen 2 variables predictoras, $X1$ y $X2$, y 6 clases para clasificar, $R1 - R6$. A la izquierda aparece la representación del árbol. A la derecha, la división del espacio de atributos en seis secciones, una sección para cada clase. Una nueva observación se clasificará en una clase o en otra en función del valor que toma para $X1$ y $X2$.

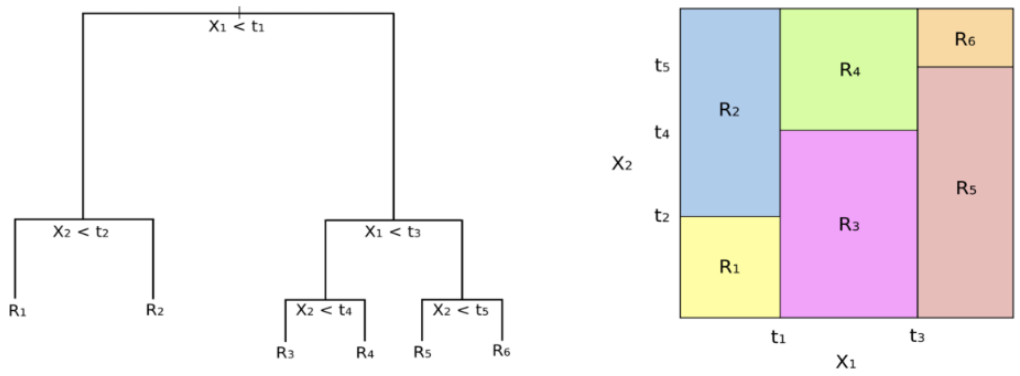


Figura 1: Árbol de decisión

Se ha presentado cómo funciona un árbol de decisión *grosso modo*. Se observa cómo se necesita un criterio que establezca qué condición se ha de poner en cada nodo para dividirlo.

Criterio de división

Para comenzar, se acuerda que cada condición involucrará siempre a una sola variable, siendo así más sencillo encontrar la condición óptima en cada paso. De otro modo, el problema de hallar la condición óptima sería demasiado complejo y ralentizaría demasiado el algoritmo.

Los datos de la tabla de la que se dispone cuentan con que todas sus variables son continuas, salvo la variable *target Class*, que no se utiliza para dividir. Por ello, se va a atajar la pregunta considerando variables predictoras continuas.

Para cada una de las variables continuas v, se necesita encontrar un valor v' de manera que se divida el nodo en elementos cuya variable v alcanza un valor superior a v' y en elementos con variable v inferior a v'. Dicho v' conviene tomarlo para que la división del nodo en los dos subnodos sea lo más conveniente posible para la clasificación. El parámetro v' puede tomar cualquier valor real. No obstante, sólo hay un número limitado de divisiones de los elementos posible. Por ello, basta con considerar un número finito de valores de la recta real.

Es decir, si se supone que la variable v toma los valores $v1 < v2 < v3 < v4 < v5$, sólo interesa comparar las divisiones tomando como v' un valor entre $v1$ y $v2$, un valor entre $v2$ y $v3$, un valor entre $v3$ y $v4$ y un valor entre $v4$ y $v5$. Por simplificar, se toma el punto medio de cada par de valores contiguos. Obviamente, con valores inferiores a $v1$ o superiores a $v5$ no se conseguiría dividir la muestra en dos, pues todos los individuos formarían parte del mismo subgrupo. Luego no se consideran.

Por tanto, si se cuenta con V valores diferentes para la variable v, se tienen V-1 condiciones posibles.

Una vez establecido el conjunto de condiciones posibles para cada variable, se necesita de un criterio de decisión para tomar una división frente a otra. Para ello, se hace uso de la llamada función de impureza.

Definición 3.1. Sean $c1$ y $c2$ las dos clases de una variable categórica C y t un nodo de un árbol de decisión. La **función de impureza** se corresponde con una función de la forma $i(t) = \phi(p(c1|t), p(c2|t))$, donde $p(c|t)$ es la probabilidad de encontrar un elemento de la clase c en el nodo t, tal que ϕ satisface las siguientes condiciones:

- ϕ alcanza su valor máximo para $p(c1|t) = p(c2|t) = 0{,}5$.
- ϕ alcanza su valor mínimo si y solo si $\exists c \mid p(c|t) = 1$.
- ϕ es una función simétrica sobre $p(c1|t)$ y $p(c2|t)$.

Existen varias funciones de impureza. A continuación, se presentan dos de ellas:

- **Error de clasificación**: $i(t) = 1 - \text{máx}_c\, p(c|t)$
- **Impureza de Gini**: $i(t) = p(c1|t)(1 - p(c1|t))$

Modo de aplicación de la función de impureza

Sea un nodo t dividido en dos subnodos t_+ y t_- por cierta condición q, como se observa en la Fig. 2.

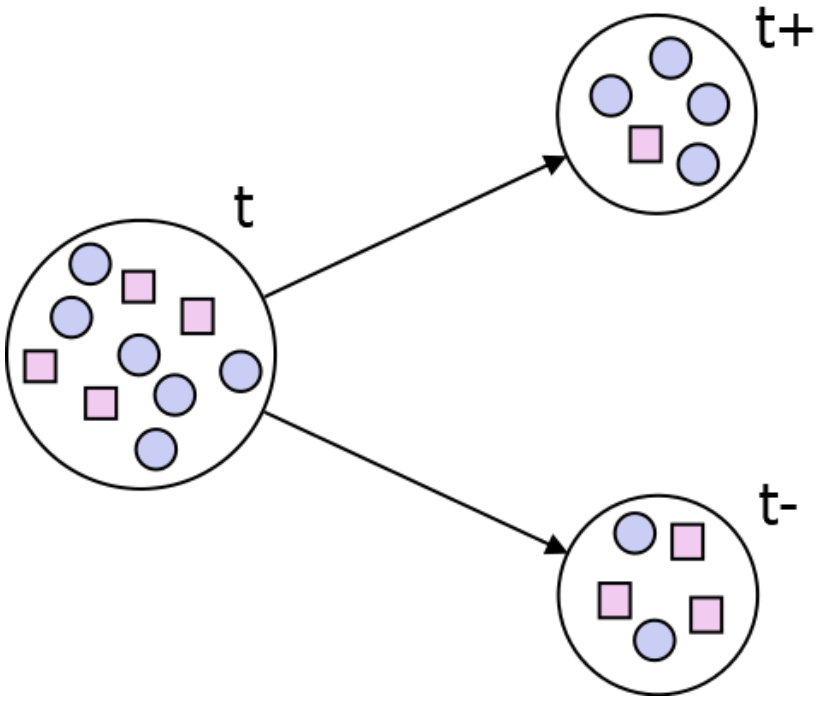


Figura 2: División de un nodo

La siguiente expresión mide cuánto disminuye la función de impureza si al nodo t se le aplica la condición q que divide la muestra t en t_+ y t_-:

$$\Delta i(t) = i(t) - p_+ i(t_+) - p_- i(t_-)$$

Donde p_+ es la proporción de elementos de t que pasa a formar parte de t_+, y p_- la proporción que pasa a formar parte de t_-.

La mejor condición q^* será aquella que minimice la función de impureza. Luego será aquella que consiga que el incremento $\Delta i(q,t)$ sea máximo:

$$q^* = \max_q \Delta i(q,t)$$

Por tanto, ante la posible división de un nodo, hay que tomar todas las divisiones que envuelven a una sola variable y compararlas utilizando alguna métrica, para tomar la que minimiza la función de impureza.

A menudo, se establece un criterio de parada para impedir que el árbol crezca hasta su máxima extensión. Es una forma de controlar el sobreajuste. Existen diferentes modos de frenar el crecimiento.

Criterios de parada

- **Profundidad máxima**: Se establece una profundidad máxima y no se dividen más los nodos que hayan alcanzado dicha profundidad.
- **Número mínimo de elementos**: Se establece el mínimo número de elementos a haber en un nodo para su división y no se realizan cortes sobre aquellos nodos con una cardinalidad inferior a dicho número.
- **Mejora mínima en la impureza**: El árbol deja de crecer si en algún nodo la mejor división posible disminuye la función de impureza demasiado poco. Es decir, se establece un β y se deja de dividir si $\Delta i(t) \leq \beta$. Si $\beta = 0$ el árbol no parará de crecer hasta que los nodos sean homogéneos.

Al impedir la extensión máxima de un árbol, pueden aparecer nodos terminales con elementos de distintas clases. ¿Pero qué clase se asigna a un nodo terminal si no es homogéneo? De nuevo, hay distintos criterios.

Criterios para la clasificación

- **Voto mayoritario**: Toma la clase c^* con más representantes en el nodo terminal t.

  $$c^* = arg\max_c p(c|t)$$

  Este criterio es el más intuitivo.
- **Minimización de costes**: Dado el coste por cada tipo de error, toma la clase c^* que disminuye el coste de los errores.

  $$c^* = arg\min_c \sum_k C(c|k)p(c|k)$$

  Donde $C(c|k)$ es el coste de clasificar un elemento de la clase k en la clase c.

Presentada la construcción del árbol de decisión al detalle, se percibe cómo el desarrollador del modelo tiene que tomar ciertas decisiones al crearlo. Por ejemplo, debe decidir el criterio de parada o el de división. Estas pequeñas opciones se conocen como los hiperparámetros del modelo y ayudan a ajustar el modelo en mayor medida a las necesidades que el desarrollador tiene. Todo modelo permite controlar el valor de los hiperparámetros que lo configuran.

A continuación, aparecen algunos posibles hiperparámetros a fijar en los árboles de decisión.

Hiperparámetros:

- Función de impureza.
- Mínimo número de observaciones para dividir un nodo.
- Máxima profundidad del árbol.
- Mínimo decrecimiento de la función impureza para dividir un nodo.

Dependiendo del problema, el valor óptimo para los hiperparámetros será uno u otro. Para llegar hasta ese conjunto óptimo puede utilizarse la validación cruzada. Por defecto, el software *Python* tiene definido ciertos valores y no siempre será necesario modificarlos todos.

Ventajas y limitaciones

Los árboles de decisión son fundamentales en Machine Learning y aparecen en muchos otros modelos. Gran número de problemas de Machine Learning pueden resolverse bien con algún desarrollo de su técnica. No obstante, es cierto que los árboles de decisión por sí solos no clasifican tan bien y caen con facilidad en el sobreajuste. Una forma de frenar el sobreajuste es estableciendo un criterio de parada que impida su máxima extensión.

Como todo modelo, los árboles de decisión tienen argumentos a favor de su uso y argumentos en contra. Una gran ventaja que presentan es su fácil interpretación. De hecho, la representación del árbol permite entender el modelo y justificarlo de cara a un potencial cliente. Aunque si el árbol fuera muy extenso, la representación no sería nada clara. Además, como cada subdivisión utiliza una variable diferente, puede capturar patrones no lineales y de gran complejidad. Es cierto que puede resultar poco óptimo para detectar patrones monótonos y que es sensible a la elección de sus hiperparámetros. Valores distintos en los hiperparámetros pueden dar como resultado clasificaciones demasiado diferentes.

No obstante, el mayor problema con el que lidian los árboles de decisión se corresponde con la falta de equilibrio entre el sesgo y la varianza.

Se recuerda que el sesgo cuantifica cuánto en promedio difieren los valores predecidos de los valores reales. Mientras que la varianza cuantifica cuánto de diferentes serán las predicciones de un modelo en un mismo punto si muestras distintas se tomaran de la misma población. Al construir un árbol pequeño se obtendrá un modelo con baja varianza, pero alto sesgo. Normalmente, si se incrementa la complejidad del árbol, disminuye el sesgo pero, el exceso de complejidad provoca el sobreajuste del modelo, que se verá reflejado en un incremento importante de la varianza.

El modelo óptimo debe mantener un balance entre estos dos tipos de errores. A esto se le conoce como el equilibrio entre los errores de sesgo y de varianza y es difícil de alcanzar en árboles de decisión, que generalmente tienen una alta varianza. El uso de agrupaciones de modelos es una forma de conseguir este equilibrio.

3.3.2. Bagging

El bagging es una técnica muy común usada para reducir la varianza a través de la combinación de varios clasificadores. Cada uno de ellos son modelados con diferentes conjuntos tomados aleatoriamente de la misma población.

Construcción del modelo

La Fig. 3 muestra la técnica bagging aplicada con árboles de decisión. Se ve cómo se toman diferentes subconjuntos de la muestra de entrenamiento con los que se forman distintos árboles.

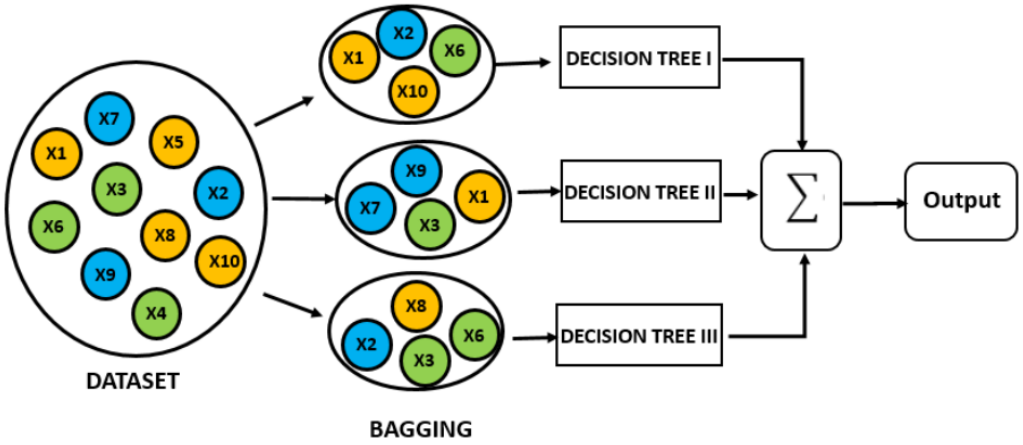


Figura 3: Bagging de árboles

Una vez construidos todos los árboles, se combinan su predicciones para dar una predicción final. En el caso de un problema de clasificación, como el planteado con la detección del fraude bancario, la transacción podría ser clasificada con la clase más votada entre todos los árboles.

Hiperparámetros

Los hiperparámetros del bagging de árboles en su mayoría son los hiperparámetros del CART. No obstante, el bagging tiene hiperparámetros propios.

- Número de árboles.
- Número de elementos con los que se construye cada árbol.
- Si las muestras son tomadas con reemplazamiento o no en cada árbol.

Ventajas y limitaciones

Generalmente, es más eficaz que el árbol de decisión, aunque es más costoso computacionalmente. Mejora la varianza al hacer varios árboles, pero corre el peligro de acabar elaborando árboles muy similares entre sí.

3.3.3. Random forest (bosque aleatorio)

El modelo random forest surge para proveer una mejora significativa a los CART, ya que sufren altos problemas de sesgo y varianza.

El random forest es una versión más compleja de la técnica bagging. De nuevo, se construyen varios árboles, de ahí su nombre (bosque aleatorio). La complejidad adicional del random forest reside en que en cada nodo se podrá eligir la condición de separación sólo con algunas de las variables. Concretamente, con un subconjunto de variables que se tomará aleatoriamente en cada nodo. Esto dificulta que los árboles del bosque sean muy parecidos entre sí. Así hay un contraste mayor entre las predicciones y se reduce aún más la varianza.

Construcción del modelo

Cada árbol de decisión se construye como sigue.

1. Se toma una muestra del conjunto de entrenamiento formado por N casos. Introduce aleatoriedad al algoritmo, ya que cada árbol se forma de una manera diferente.
2. Si existen M variables de entrada, un número $m < M$ se especifica tal que, para cada nodo, la condición de seperación del nodo se puede elegir sólo entre las m variables que se seleccionan aleatoriamente. El valor m se mantiene constante durante la generación de todo el bosque.
3. Cada árbol crece hasta su máxima extensión posible, si no se ha establecido un criterio de parada.

En la formación del bosque se introduce en gran medida la aleatoriedad, ya que cada árbol está construido con una muestra distinta y variables escogidas de manera diferente. Esta aleatoriedad consigue reducir la correlación entre los árboles.

Una vez construido el bosque, se utiliza para clasificar nuevas observaciones. Al igual que en la técnicas bagging, en problemas de clasificación, la predicción final puede ser la clase más votada entre todos los árboles.

Hiperparámetros

Los bosques aleatorios contienen a los hiperparámetros de los árboles de decisión y además poseen algunos propios. Se mencionan algunos de estos últimos.

- Número de árboles que forman el bosque.
- Número de variables que se seleccionan en cada nodo.
- Número de elementos con los que se crea cada árbol.
- Si en cada árbol las muestras son tomadas con reemplazamiento o no.

El error en las predicciones realizadas por el random forest está fuertemente relacionado con los dos primeros hiperparámetros nombrados.

Al reducir el número m de variables, se reduce la correlación entre los árboles, ya que cada nodo tiene menos posibilidades entre las que elegir. Sin embargo, al reducir m, también se reduce la precisión del árbol. Luego hay que llegar a un equilibrio. El valor recomendado para un problema de clasificación es tomar $m = \sqrt{M}$.

El número de árboles también tiene efecto en la precisión de la predicción. Como es lógico, a mayor número de árboles, mejor será la predicción. Sin embargo, existe un valor para el cual el error ya no disminuye apenas y aumenta considerablemente la complejidad del algoritmo.

Ventajas y limitaciones

El random forest consigue un buen equilibrio entre el sesgo y la varianza mejorando notablemente al CART. Además de realizar una buena clasificación de las observaciones, facilita información acerca de las varibles predictoras. Una de las salidas del modelo es la importacia de las variables, que de algún modo cuantifica cuánto aporta cada variable al modelo[6]. Es una información de gran utilidad.

Su mayor limitación es que se tiene poco control de lo que hace el modelo. Al tratarse de un bosque con decenas de árboles, se pierde la interpretación con la que se podía contar en los árboles pequeños.

3.3.4. Extremely randomized trees

El modelo extremely randomized trees (Ernst, Geurts y Wehenkel, 2006) es considerado un random forest que ha sufrido ciertas modificaciones.

Construcción del modelo

Se procede igual que en el random forest, salvo por el hecho de que los árboles se construyen usando todos los elementos y por la manera de tomar las condiciones en cada nodo.

En cada nodo, se elige aleatoriamente un subconjunto de m variables. Posteriormente, para cada variable, se elige una condición de manera aleatoria entre todas las posibles: $v1 < a1, v2 < a2, \ldots, vs < as$, para $a1, ..., as$ valores aleatorios en el rango de la variable. Finalmente, se toma la división que minimiza la función de impureza.

Ventajas y limitaciones

La aleatoriedad hace que los árboles difieran más entre sí y que el entrenamiento del modelo sea más rápido. No obstante, los árboles son más profundos y la velocidad de predicción es más lenta. Respecto al random forest, reduce aún más la varianza. Suele ofrecer buenos resultados.

[6]Más detalles al respecto pueden encontrarse, por ejemplo, en Archer y Kimes (2008)

3.3.5. Isolation forest (bosque de soledades)

El isolation forest no responde al aprendizaje supervisado como los modelos anteriores, sino al aprendizaje no supervisado. De hecho, no sirve para problemas de clasificación en general, sino para la detección de anomalías. Su aplicación en este caso consiste en tratar de detectar el fraude poniendo bajo sospecha de fraude cualquier movimiento anómalo.

Construcción del modelo

El isolation forest también se basa en el random forest. Como en el extremely randomized trees, en cada árbol se toman todos los elementos. Pero, en este caso, cada corte se realiza escogiendo una variable y un punto de corte de manera totalmente aleatoria. Se mantienen los árboles creciendo hasta que solo queda un elemento en cada nodo terminal.

Obviamente, los elementos anómalos tienen mayor probabilidad de acabar en nodos de poca profundidad, de ser aislados antes. Mientras que elementos muy parecidos a otros acabarán en nodos profundos.

Luego la idea es calcular cuánta profundidad en media necesita cada observación para quedar aislada. Así, cuantas menos divisiones haya necesitado el árbol para aislar a cierta observación, más anómala será.

Hiperparámetros

- Porcentaje de observaciones más anómalas que se quiere extraer de la muestra.

Ventajas y limitaciones

Detecta atípicos sin necesidad de medidas de distancia, similitud o densidad, que suele ser computacionalmente muy costoso. Tiene la capacidad de escalar en tablas de datos grandes y con muchas variables irrelevantes. Además, dado que las anomalías se encontrarán a poca profundidad, no interesa desarrollar árboles muy grandes y no será tan costoso computacionalmente. Como limitación principal está el tener que definir la proporción de atípicos a priori.

3.4. Medidas de rendimiento

Una vez obtenido el modelo de clasificación, se trata de probar su validez con la muestra test, que es totalmente independiente a la creación del modelo. Cuantificar el rendimiento permitiría comparar la eficacia de los distintos modelos.

Clasificada la muestra test según un modelo, se puede establecer la siguiente tabla donde Y es la variable dicotómica a predecir e $\hat{Y}$ la clasificación realizada. En este caso, el valor 1 se corresponde con los casos de fraude y el valor 0 con los casos legítimos.

Esta tabla es conocida como la matriz de confusión de un modelo de clasificación.

$\hat{Y}/Y$	1	0
1	VP (Verdadero Positivo)	FP (Falso Positivo)
0	FN (Falso Negativo)	VN (Verdadero Negativo)

Tabla 1: Matriz de confusión

En la medida de lo posible, se pretende que las observaciones sean clasificadas correctamente. Es decir, que la mayoría de valores se encuentren en VP (Verdadero Positivo) o en VN (Verdadero Negativo). En general, será difícil garantizar que la clasificación sea correcta al 100 %. Por desgracia, se asume que la clasificación probabilística comete errores. La labor del modelador será que dichos errores sean mínimos y supongan el menor coste posible. Para ver cómo de bien predice un modelo, se hace uso de las distintas medidas de rendimiento.

Una medida de rendimiento muy intuitiva es *accuracy*, que calcula la proporción de veces en la que se ha clasificado correctamente.

$$Accuracy = \frac{VP + VN}{VP + FP + FN + VN}$$

Sin embargo, no basta con tomar aquel modelo que haya acertado en un mayor número de ocasiones. De ser así, se podría optar por considerar que toda transacción es no fraudulenta. Se acertaría en la basta mayoría de los casos, pero parece obvio que no es un buen modelo de clasificación al no detectar ningún caso de fraude.

El objetivo de este trabajo es detectar el fraude bancario. Luego errar cuando Y=1 supone no detectar que una transacción es fraudulenta. Sería muy costoso y generaría gran desconfianza en el modelo. Así, conviene prestar una atención especial a este tipo de error.

Se encuentra la medida *recall*, que calcula el porcentaje de transacciones fraudulentas correctamente clasificadas.

$$Recall = \frac{VP}{VP + FN}$$

Otra medidas es *precision*, que calcula el porcentaje de transacciones realmente fraudulentas dentro de las clasificadas como tal.

$$Precision = \frac{VP}{VP + FP}$$

F1Score combina las medidas de *recall* y *precision* en una sola, haciendo una media ponderada de las dos. *F1Score* tiene en cuenta tanto los falsos negativos (FN) como los falsos positivos (FP), queriendo reducir ambos.

$$F1Score = 2 * \frac{Recall * Precision}{Recall + Precision}$$

Esta medida se considera más adecuada para el problema que *accuracy*, controlando la precisión en la clasificación de las transacciones fraudulentas. No obstante, se le prestará una especial atención a *recall*. Ante todo, es necesario detectar los casos de fraude.

4. Aplicación

Se ha escogido una tabla de datos que recoge los detalles de muchas transacciones bancarias y su condición fraudulenta o legítima para probar, para ese caso particular, los distintos modelos analizados. En primer lugar, se debe mencionar que no es fácil disponer de una tabla de datos que muestre la información completa de las transacciones. Por un lado, porque son datos protegidos y cada entidad bancaria tiene que cumplir con su compromiso de privacidad. Por otro, porque son datos valiosos que los bancos no están dispuestos a ofrecer gratuitamente.

Ante dicha dificultad, se ha optado por utilizar una tabla facilitada por la plataforma *kaggle*[7] cuyas variables corresponden en su mayoría con las componentes principales de un PCA *(Principal Component Analysis)*[8]. Se entiende que las variables están en PCA precisamente para ocultar información que pueda resultar sensible. No ha de ser un inconveniente para el posterior desarrollo del modelo, aunque sí que debilita la sección de análisis exploratorio.

Se procede a analizar los datos que hay en la tabla. Posteriormente, se deberán hacer ciertas manipulaciones para desarrollar algunos modelos correctamente. Elaborados los modelos, se prueba el rendimiento de cada uno y se comparan entre ellos.

4.1. Análisis de los datos

La tabla escogida cuenta con 284.807 registros y 31 campos. No hay ningún valor perdido, duplicado o incorrecto. Lo que es una gran ventaja y ahorra una inmesa cantidad de trabajo en depuración de datos.

Las variables recogidas por cada transacción son:

- *Time*: Número de segundos entre la primera transacción de la tabla de datos y la correspondiente.
- *V1-V28*: Componentes principales del análisis de componentes principales.
- *Amount*: Cantidad en dólares que envuelve la transacción.
- *Class*: Variable dicotómica: 1 si es un fraude, 0 en otro caso.

La interpretabilidad de las variables es escasa, puesto que no se conoce el significado de ninguna de ellas, salvo el de *Time*, *Amount* y *Class*. Como bien es sabido, cada una de las componentes principales corresponde a una combinación lineal de las variables originales, por lo que no revela información en sí.

El objetivo se encuentra en predecir la variable *Class* en futuras transacciones de las que se contará con el valor en el resto de variables. Es decir, el modelo recibirá todos los datos acerca de una transacción, salvo su condición, y tratará

[7]Comunidad en línea perteneciente a Google de científicos de datos y profesionales del aprendizaje automático.

[8]Más detalles al respecto pueden encontrarse, por ejemplo, en Hotelling (1933).

de clasificarla. Por tanto, la variable *Class* es la variable *target* en este problema e interesa especialmente.

Cabe mencionar que, ante todo, el modelo ha de clasificar bien a las acciones fraudulentas. De no ser así, el banco se enfrentaría con pérdidas innecesarias. En cambio, el error en la clasificación de las acciones legítimas no preocupa tanto. Clasificar como fraudulenta una transacción legítima no supondría ninguna pérdida a priori, tan solo habría que revisar la situación y corregir la clasificación, si fuera el caso.

A continuación, la Fig. 4 muestra la distribución de la variable *Class*.

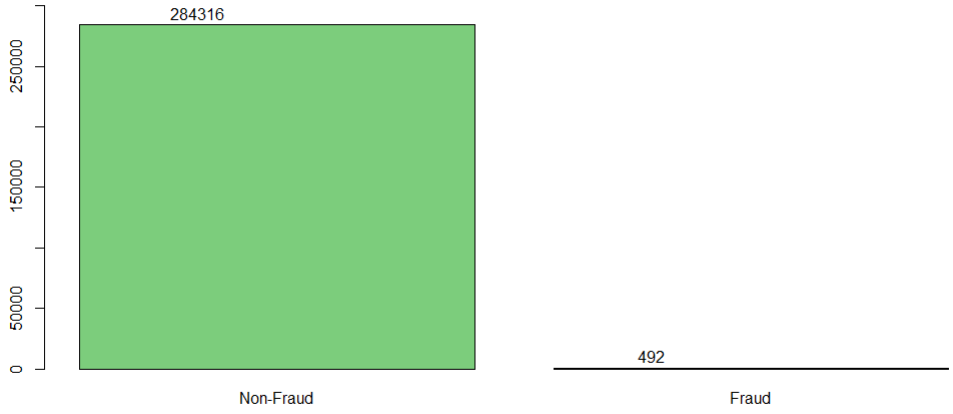


Figura 4: Cardinalidad de cada tipo de transacción

Hay 492 casos de fraude frente a 284.314 casos de no fraude. Las transacciones fraudulentas se corresponden con tan solo el 0,17 % de la muestra, que es un porcentaje muy bajo. La marginalidad del fraude es una buena noticia para los bancos, la mayoría de transacciones entran en el marco de la legalidad. Pero no lo es tanto para la elaboración de algunos modelo. Cuanto más ejemplos se tengan de casos de fraude, más sencillo resultará encontrar patrones de comportamiento entre ellos.

Además de observar la variable objetivo *Class*, se examinan el resto de variables. Se analiza la distribución que sigue cada una y se observa si difiere mucho en función del valor de *Class*. De ser así, se intuiría que ambas variables se encuentran relacionadas.

En primer lugar, se analiza la variable *Amount*, que revela la cantidad de dinero implicada. ¿Tendrá un comportamiento diferente en función del tipo de transacción? Se muestran los detalles de la variable por clase.

```
Non-Fraud transaction statistics
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
Fraud transaction statistics
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Figura 5: Detalles de *Amount* por *Class*

Para presentar de una manera más ilustrativa estos datos, se realiza un gráfico box-plot. Así, se podrán sacar conclusiones con mayor claridad. Además del gráfico box-plot de *Amount* por *Class*, se muestra también el correspondiente con la transformación logarítmica de *Amount*. Esta transformación permite una mejor visualización de la variable. Muchas transacciones toman valores muy elevados y distorsionan el gráfico.

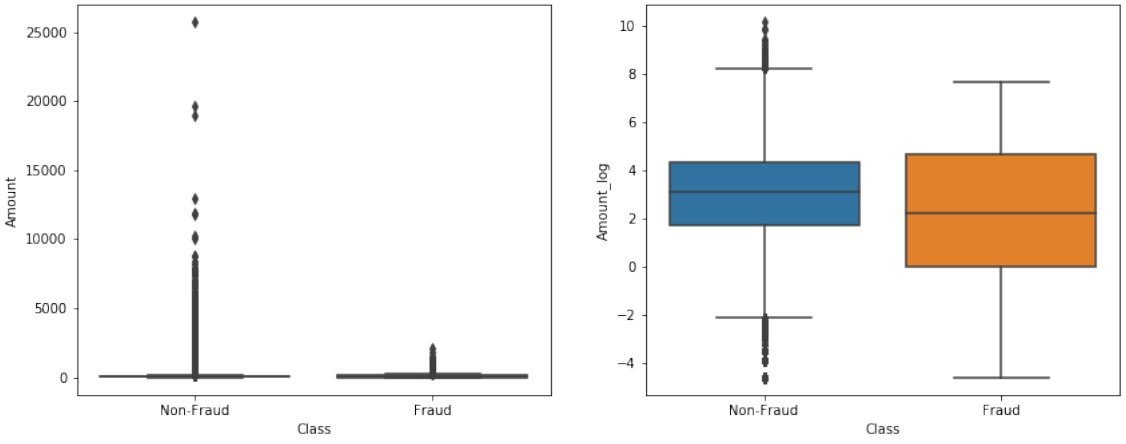


Figura 6: Gráficos box-plot

En la Fig. 6, se observa cómo efectivamente hay muchas transacciones no fraudulentas que abarcan gran cantidad de dinero. De hecho, hay 66.042 transacciones de este tipo que superan los 5.000 dólares, mientras que la media se encuentra tan solo en torno a los 250. No ocurre lo mismo con las transacciones fraudulentas, donde la máxima cantidad es de 2.125 dólares, distando mucho de 5.000. Quizás la motivación de esta diferencia sea una estrategia para poder camuflar mejor la acción ilegítima. Lógicamente, las transacciones voluminosas captan más la atención de las entidades bancarias.

En el gráfico box-plot de la transformación logarítmica, se percibe que el rango intercuatílico en las acciones fraudulentas es mayor. Pero también que el importe suele ser inferior al de las transacciones legítimas. Además, estas últimas transacciones cuentan con un gran número de outliers.

Por tanto, las transacciones fraudulentas en promedio comprenden menos cantidad de dinero que las legítimas y nunca cantidades excesivamente altas (superiores a 5.000 dólares).

También se encuentra la variable *Time*, que indica la diferencia en segundos entre la primera transacción de la tabla de datos y la transacción correspondiente. No se conoce la hora en la que se dió la primera transacción, por lo que el factor tiempo se tiene que tratar en términos relativos.

Para un manejo más sencillo, se convierten los segundos de *Time* en horas y minutos y se agrupan las transacciones en función de su hora módulo 24, por las 24 horas del día. De esa manera, se comprueba si hay alguna relación entre la ocurrencia de las transacciones fraudulentas y la hora del día. No se conoce el punto de partida, luego no se podrá concluir qué horas exactamente son más vulnerables a la estafa.

A continuación, en la Fig. 7, se presenta la función de densidad del número de transacciones por horas de diferencia con respecto a la hora en la que se produce la primera transacción, distinguiendo por clase. El color rojo alude a las acciones fraudulentas y el verde a las legítimas.

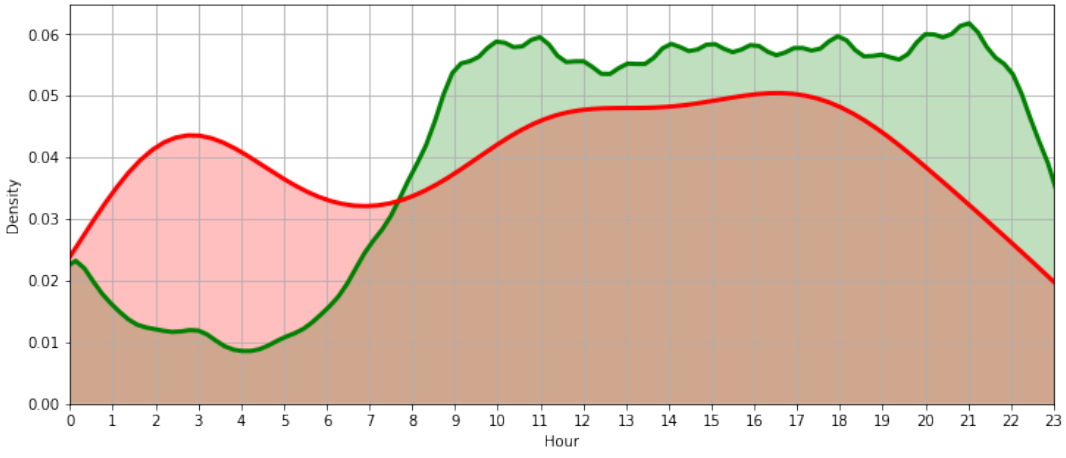
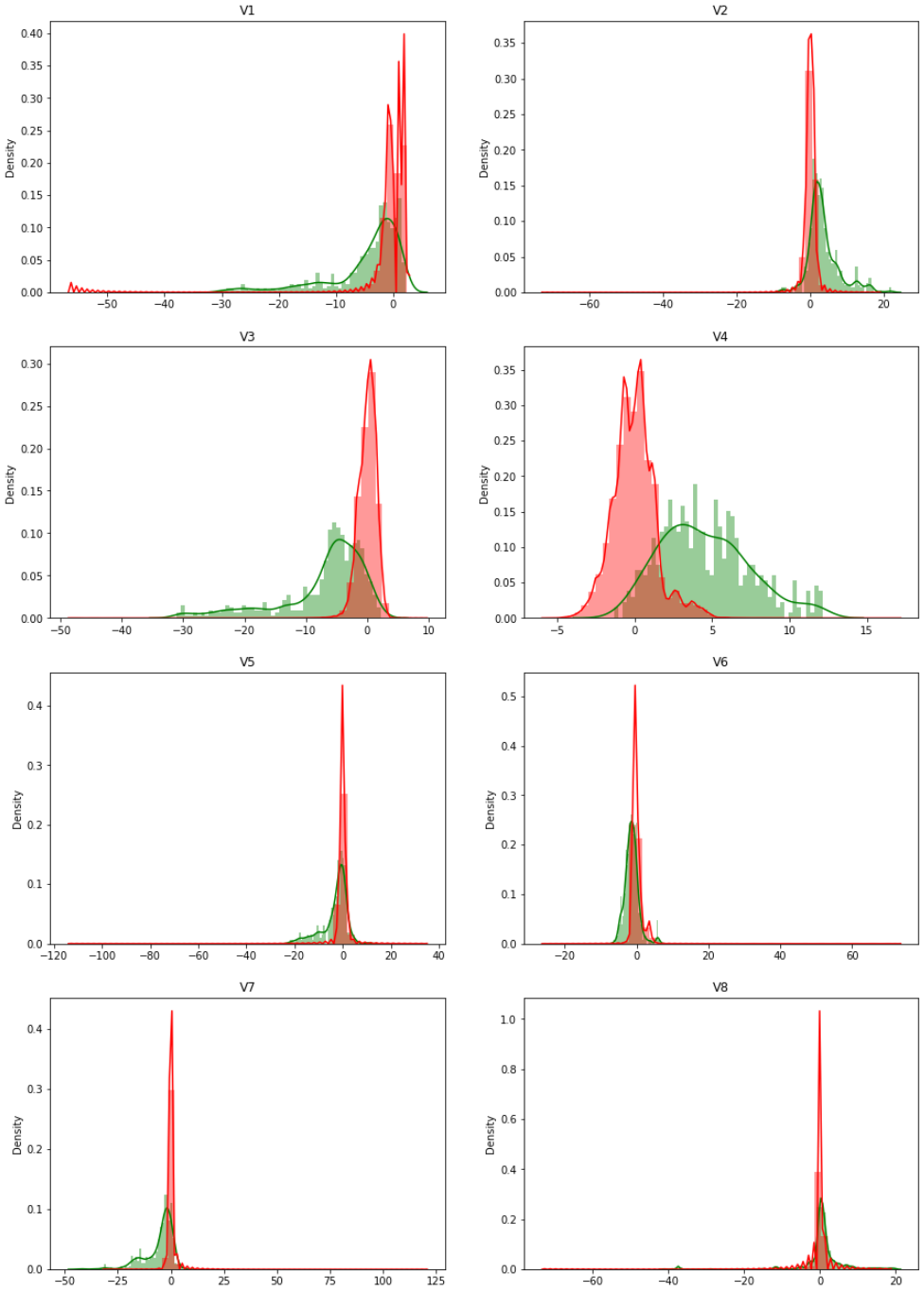
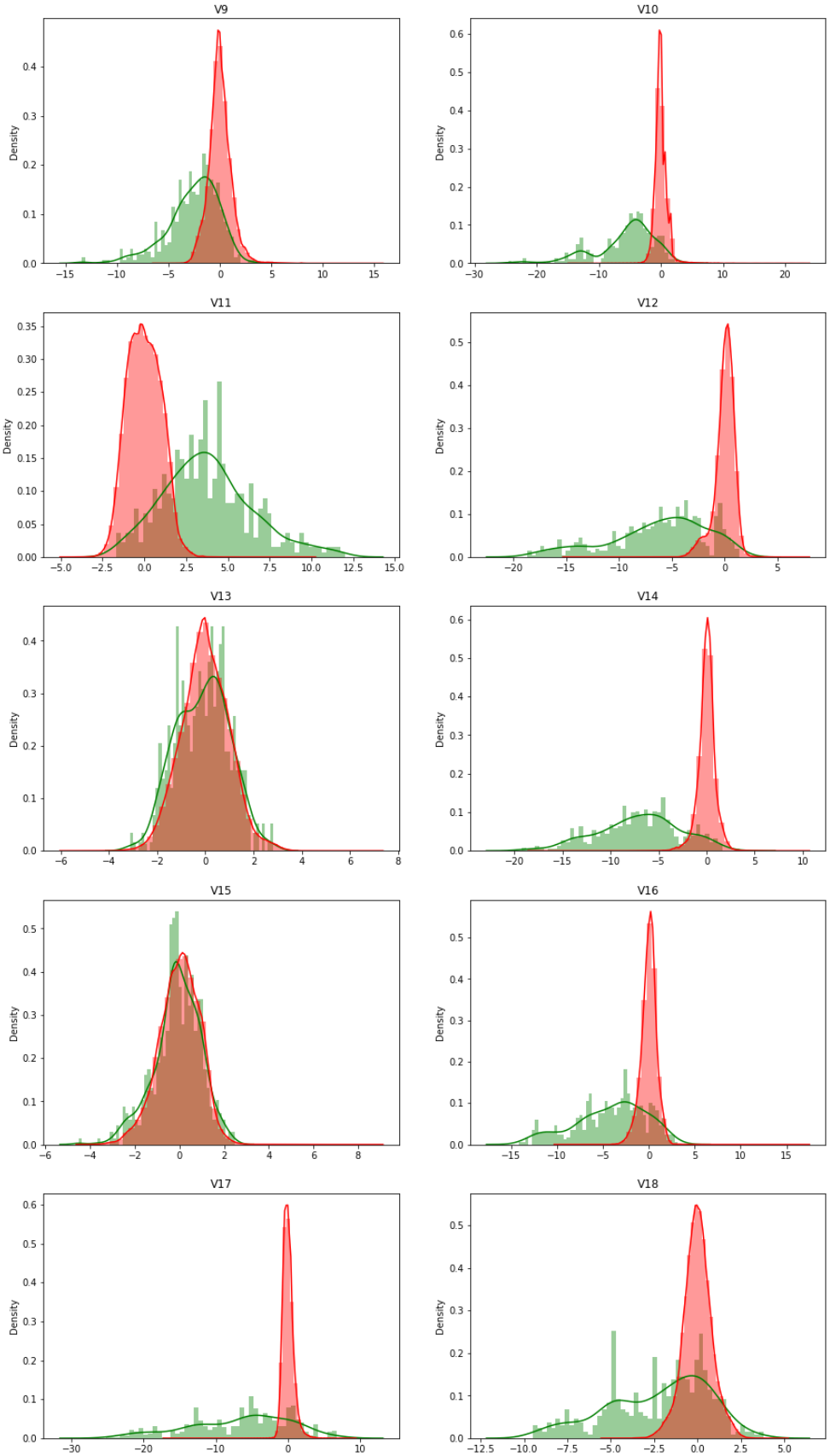


Figura 7: Función de densidad de número de transacciones por hora

Destaca el intervalo de 1 a 6 horas de diferencia. Parece claro que las acciones fraudulentas son especialmente frecuentes durante esas horas para la cantidad de transacciones que se realizan. Luego se puede concluir que sí que existe cierta correlación entre el momento del día y la probabilidad de encontrar fraude bancario.

Como se adelantaba, las otras variables son componentes principales de un PCA, luego no revelan mucha información. Sin embargo, se representan sus funciones de densidad por tipo de transacción. Será interesante observar si presentan diferencias en función de la variable *Class*. De nuevo, el color rojo alude a las acciones fraudulentas y el verde a las legítimas.




V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
Density

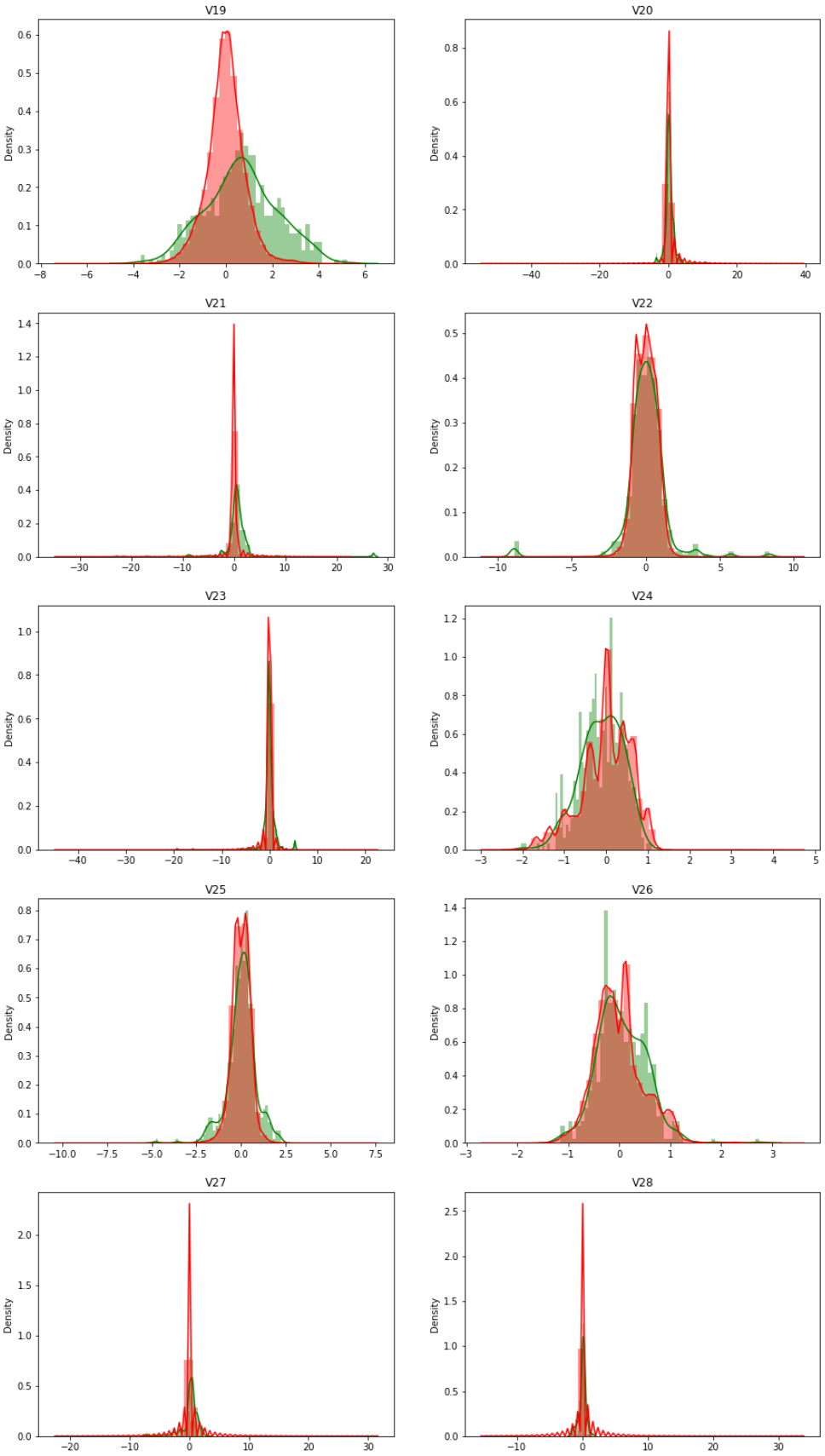


Figura 8: Función de densidad por componente y por tipo de transacción

Destacaría el comportamiento de las funciones de densidad en las variables V3, V4, V9, V10, V11, V12, V14, V16, V17, V18 y V19. En ellas, se observa claramente la diferencia de distribución entre fraude y no fraude.

La manera más directa de estudiar si existe alguna relación entre las variables es a través de la matriz de correlaciones, que presenta el coeficiente de correlación por cada par de variables. En este caso, la mayoría de las variables no están relacionadas, puesto que $V1 - V28$ son componentes principales provenientes de un PCA, que precisamente son escogidas de manera que tengan covarianza cero entre ellas. No obstante, sí que es interesante ver cómo se relacionan con la variable dicotómica *Class*.

Hay una gran cantidad de variables (31). Para que la matriz de correlaciones quede más despejada, se van a tomar sólo aquellas componentes del PCA mencionadas anteriormente, entendiéndose que serán las que tengan un índice de correlación con *Class* más lejano a cero[9].

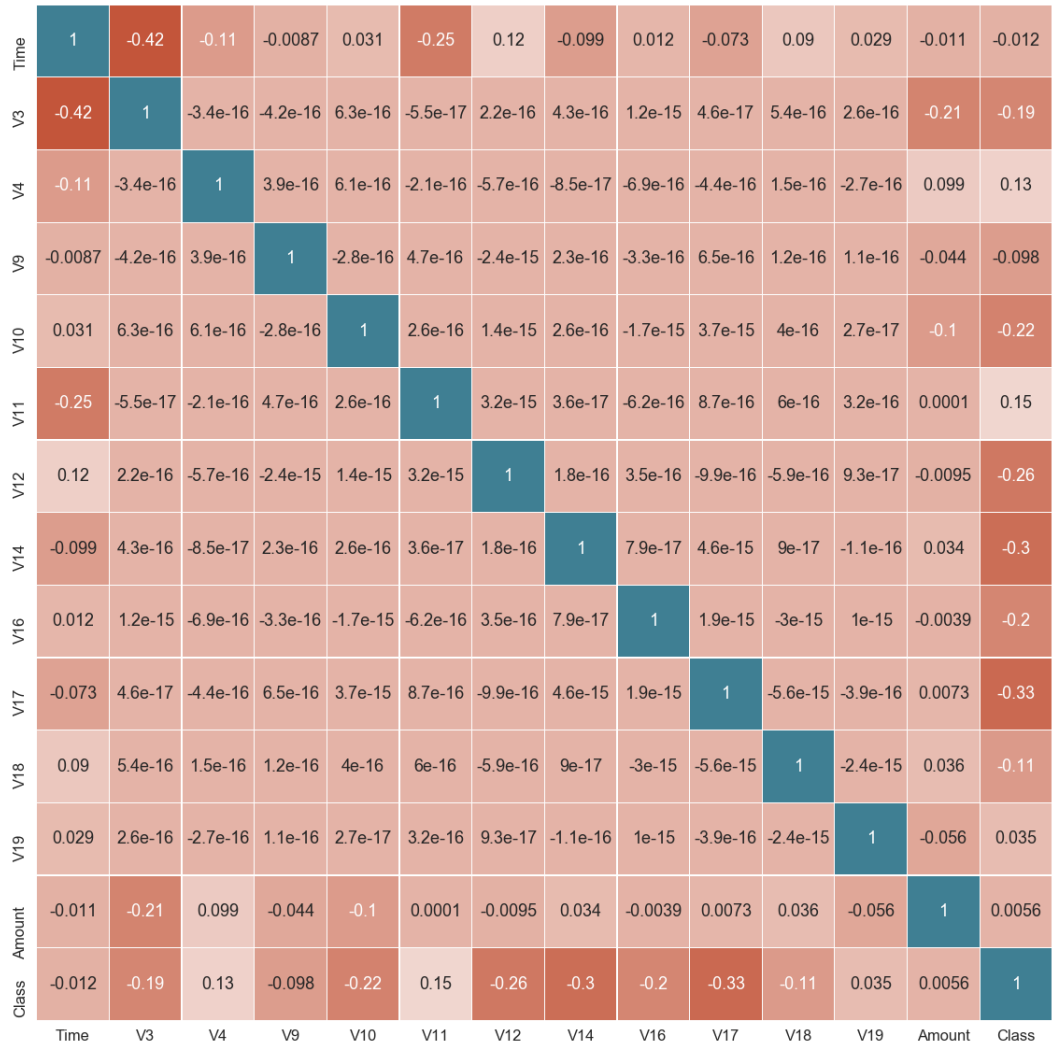


Figura 9: Correlograma

[9]En el anexo se encuentra la matriz de correlaciones con el resto de componentes principales.

Atendiendo a la variable *target Class*, se observa que tiene una correlación negativa con V12, V14 y V17. La correlación con el resto de variables es despreciable. Parece que estas variables cobrarán un peso especial en la realización de los modelos.

4.2. Preparación de los datos de la muestra

Para crear y posteriormente validar los modelos de clasificación, se divide la muestra en dos, un 70 % para la creación del modelo y un 30 % para validarlo. Luego la muestra de entrenamiento cuenta con 227.845 transacciones y la muestra test con 56.962 transacciones. Esta división es completamente aleatoria.

Una vez divida la muestra, se aparta por completo la muestra test. Sólo la muestra de entrenamiento se utiliza para crear el modelo. En la Fig. 10, se observa la cardinalidad de cada tipo de transacción en ella.

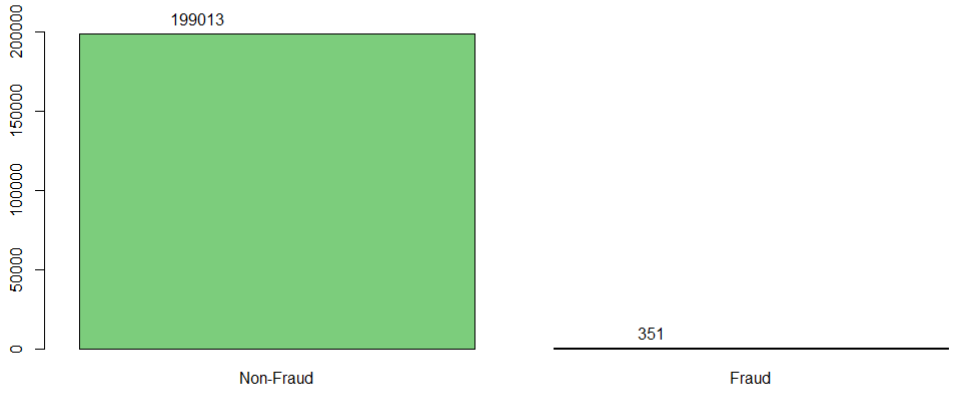


Figura 10: Distribución de cada clase

Hay 199.013 casos de no fraude frente a 351 casos de fraude. Se trata de una muestra no equilibrada, donde la mayor parte de las transacciones son legítimas.

Ante esto, en modelos de aprendizaje supervisado, se tiene que elegir entre preservar los datos de manera no equilibrada u optar por métodos de reajuste de datos, como el sobremuestreo o el submuestreo. Es posible que, si se mantienen los datos igual, el algoritmo asuma que la mayoría de las transacciones son legítimas. Sin embargo, se pretende que el algoritmo sepa detectar cualquier movimiento con posibilidad de fraude. Luego no interesa esa asunción y será conveniente considerar el remuestreo.

Atendiendo a la teoría, se opta por aplicar la técnica SMOTE, que equilibra totalmente la muestra creando transacciones fraudulentas sintéticas.

Elaborada la muestra con la técnica SMOTE, se consigue equiparar el número de transacciones legítimas y fraudulentas, como se aprecia en la Fig. 11.

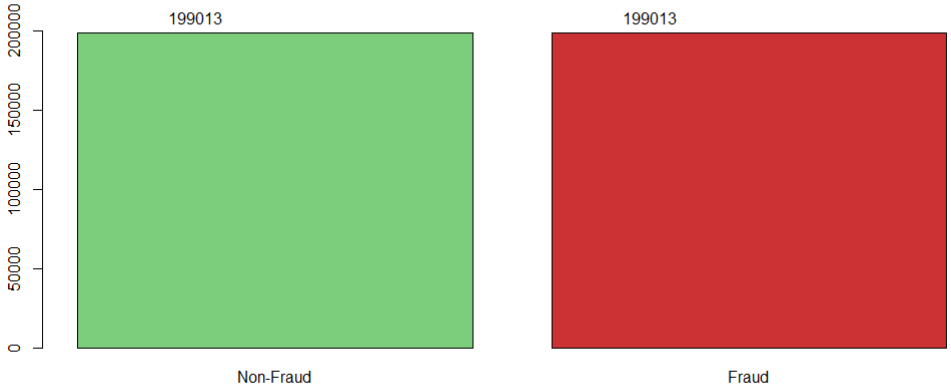


Figura 11: Distribución de cada clase tras SMOTE

Esta nueva muestra de entrenamiento se va a utilizar para la elaboración de los modelos de aprendizaje supervisado que se compararán con los modelos creados utilizando la muestra original.

4.3. Construcción de los modelos

En primer lugar, se construyen los modelos de aprendizaje supervisado.

Como se ha mencionado, los modelos son creados con la muestra de entrenamiento y cada uno requiere la definición de ciertos hiperparámetros. El software *Python* define por defecto el valor de dichos hiperparámetros, pero pueden ser alterados por el desarrollador del modelo. Se pretende tomar los valores que, entre todos los posibles, optimicen cierta medida de rendimiento. Se puede usar la validación cruzada con esta finalidad. No obstante, probar todos los conjuntos de valores posibles para ver cuál es el que ofrece mejor resultado ralentizaría enormemente la creación del modelo. Por ello, los modelos se apoyarán en la validación cruzada, pero solo de manera limitada.

Evidentemente, el modelo de clasificación tratará de acertar en la clasificiación de la muestra de entrenamiento, ya que conoce los valores que toma. Lo que no implica que la predicción sea buena para muestras diferentes. De hecho, una adecuación excesiva a la muestra de entrenamiento conduce al sobreajuste y a la mala predicción de observaciones futuras. Por ello, la clasificación de la muestra de entrenamiento no es concluyente, pudiendo confundir al desarrollador haciéndole creer que el modelo clasifica mejor de lo que lo hace realmente. Se usa entonces la muestra test, para validarlo a través de su matriz de confusión y de ciertas medidas de rendimiento.

Además, se compara si existen diferencias entre tomar la muestra de entrenamiento no equilibrada y la equilibrada con ejemplos sintéticos, y en qué consiste el cambio.

Para la comparación, se emplea exactamente el mismo modelo en ambos casos.

Los modelos de aprendizaje supervisado que se prueban son: el árbol de decisión, el bagging, el random forest y el extremely randomized trees.

4.3.1. CART

El árbol de decisión es el modelo más básico que se desarrolla en este documento. Es previsible que no ofrezca el mejor resultado. Aún así, se pretende adaptarlo para que clasifique lo mejor posible.

Para su creación se mantienen los hiperparámetros que están por defecto, como el criterio de división de Gini o el no establecimiento del número mínimo de elementos en un nodo para su división. No obstante, sí que se empleará tiempo en buscar el mejor valor para la profundidad máxima del árbol.

Como se ha mencionado anteriormente, los árboles de decisión sobreajustan con facilidad. Simplificar el modelo ayuda a que este no se adecue en exceso a los datos de los que dispone. Una manera de restar complejidad es fijando un criterio de parada del árbol que impida su máxima extensión. En este caso, se trabaja con la profundidad máxima. A priori, resulta díficil saber cuál es la profundidad máxima adecuada. Por ello, se prueba con todas las profundidades del 1 al 20 y, por validación cruzada, se coge la que ofrezca un mayor *recall* por este método. Se recuerda que *recall* es una medida de rendimiento y que cuanto mayor es su valor, más acciones fraudulentas se detectan.

Se prueba el modelo en la muestra original y en la muestra elaborada a través de SMOTE. Para la muestra original, la profundidad que ofrece un mejor resultado por validación cruzada es 4. Para la muestra equilibrada, es 19.

CART sin SMOTE

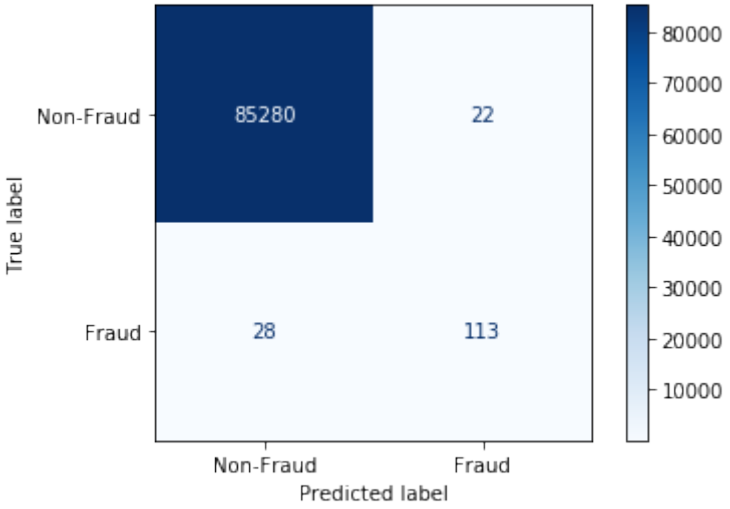


Figura 12: Matriz de confusión del árbol de decisión sin SMOTE

CART con SMOTE

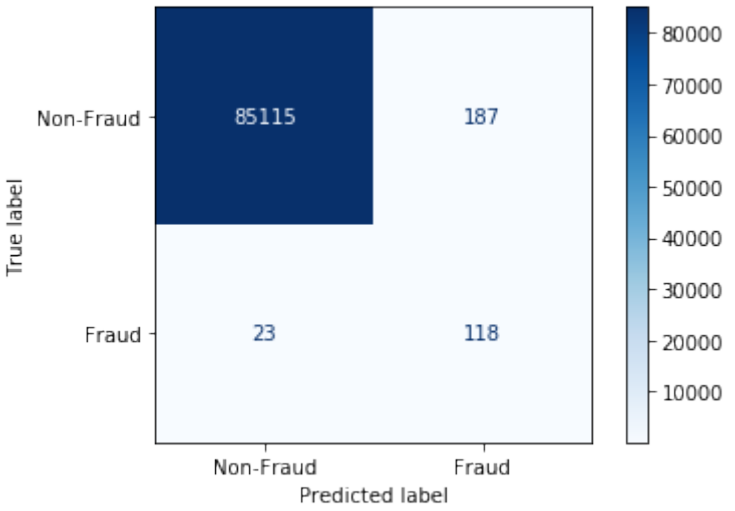


Figura 13: Matriz de confusión del árbol de decisión con SMOTE

Comparación de las medidas de rendimiento

	con SMOTE	sin SMOTE
Precision	0.39	0.84
Recall	0.84	0.80
F1Score	0.53	0.82

Tabla 2: Medidas de rendimiento del árbol de decisión

El árbol de decisión con la muestra equilibrada detecta el 84 % de los casos de fraude, mientras que sin la aplicación de la técnica SMOTE, solo el 80 %. No obstante, cabe mencionar que el modelo aplicando SMOTE yerra mucho más en los casos legítimos. De hecho, tan solo el 39 % de los casos detectados como fraudulentos realmente lo son.

4.3.2. Bagging

Pese a que es generalmente aceptado que el rendimiento del bagging es peor que el del random forest, considero útil su aportación a este trabajo. El random forest es una extensión del bagging, así se cuantifica cuánto, en la práctica, lo mejora.

La técnica bagging es muy costosa computacionalmente, ya que en ella se realizan muchos árboles de decisión. No resulta conveniente construir demasiados árboles. En primer lugar, porque añadiría más complejidad al algoritmo. En segundo lugar, porque pronto los árboles comienzan a ser muy parecidos entre sí y no suman precisión al modelo. Se recomienda utilizar en torno a la decena de árboles.

En el modelo desarrollado, se construyen cada uno de los 10 árboles con una muestra aleatoria tomada con reemplazamiento de la muestra de entrenamiento y con su misma cardinalidad.

Bagging sin SMOTE

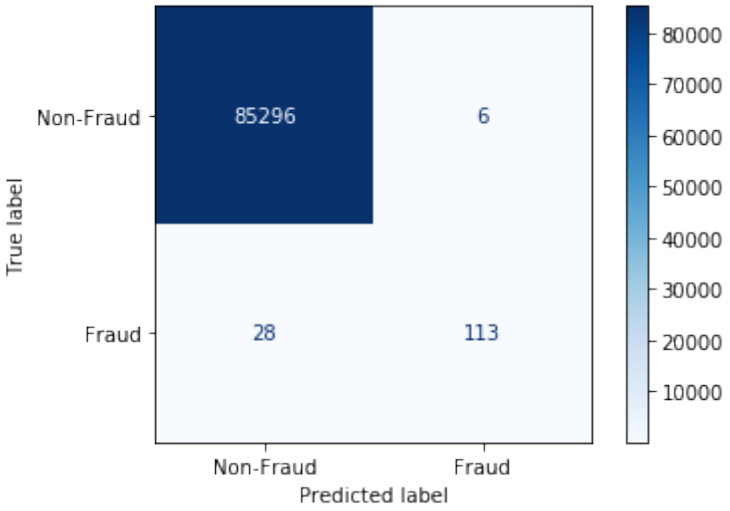


Figura 14: Matriz de confusión del bagging de árboles sin SMOTE

Bagging con SMOTE

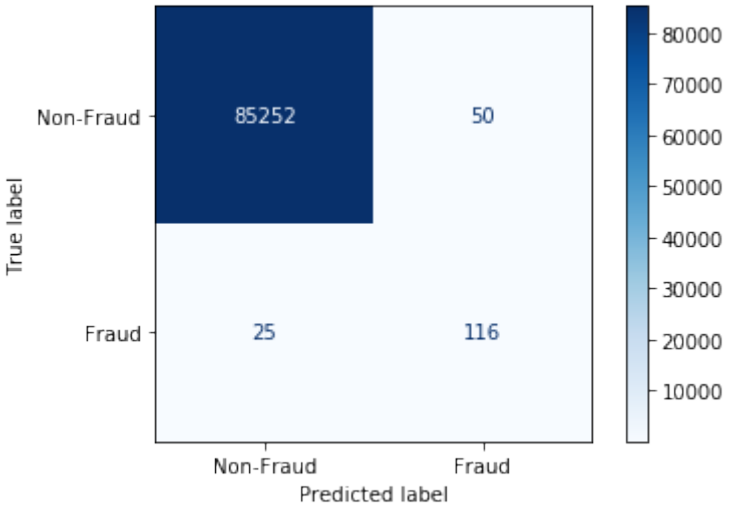


Figura 15: Matriz de confusión del bagging de árboles con SMOTE

Comparación de las medidas de rendimiento

	con SMOTE	sin SMOTE
Precision	0.70	0.95
Recall	0.82	0.80
F1Score	0.76	0.87

Tabla 3: Medidas de rendimiento del bagging

De nuevo, usando los datos equilibrados se detectan más casos de fraude, aunque se pierde precisión en la clasificación de las transacciones no fraudulentas, del 95 % al 70 %. Este modelo es costoso computacionalmente. La muestra equilibrada es más numerosa y aumenta el tiempo de creación del algoritmo, luego habría que considerar si realmente interesa. En este caso, tomar la muestra equilibrada ayuda a detectar más casos de fraude, pero aumentan en mayor medida los falsos positivos.

Respecto al árbol de decisión, es un modelo más elaborado, por lo que multiplica el tiempo de compilación. Es cierto que corrige notablemente el número de falsos positivos, siendo ahora mucho menor. No obstante, no mejora nada la clasificación de las transacciones fraudulentas. De hecho, la empeora.

La técnica bagging no parece haber aportado mucho al árbol de decisión por sí sola. Sin embargo, con pequeñas modificaciones que introducen extensiones del modelo se consigue mejorar considerablemente el rendimiento.

4.3.3. Random forest

El algoritmo del random forest construido es igual al del bagging, salvando que se toma una muestra de $\sqrt{M}$ variables[10] a elegir en cada nodo, donde M es el número total de variables predictoras.

Crear muchos árboles es costoso, pero este cambio reduce mucho el tiempo de compilación respecto al modelo anterior. En cada nodo, se busca la condición óptima entre $\sqrt{M}$ variables, no M. Ayuda a no sobreajustar, por eso ahora se deja crecer cada árbol hasta su extensión máxima. Además, provoca mayores diferencias entre lo árboles, por eso se contruyen 100, y no 10 como en el bagging. Este número de árboles es el que se estima necesario para reducir el error considerablemente sin ralentizar demasiado el algoritmo.

De nuevo, como en el bagging, cada árbol está elaborado con una muestra tomada con reemplazamiento de la muestra de entrenamiento y con su misma cardinalidad.

[10]Más detalles al respecto pueden encontrarse, por ejemplo, en Friedman, Hastie y Tibshirani (2009).

Random forest sin SMOTE

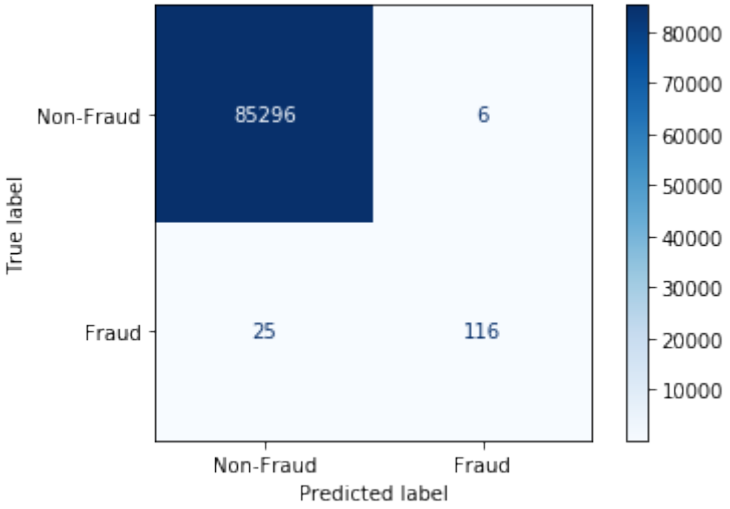


Figura 16: Matriz de confusión del random forest sin SMOTE

Random forest con SMOTE

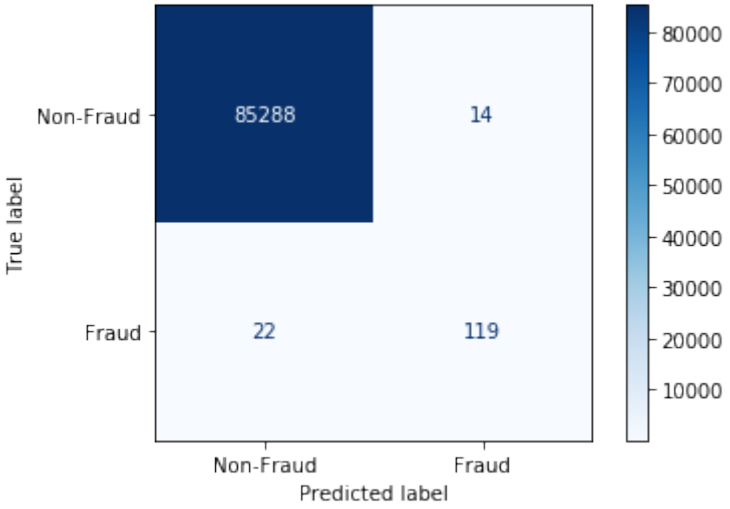


Figura 17: Matriz de confusión del random forest con SMOTE

Comparación de las medidas de rendimiento

	con SMOTE	sin SMOTE
Precision	0.89	0.97
Recall	0.84	0.83
F1Score	0.87	0.89

Tabla 4: Medidas de rendimiento del random forest

Otra vez, se sigue el patrón encontrado en los modelos anteriores. Utilizando la muestra con valores sintéticos, el modelo detecta un mayor número de acciones fraudulentas, pero también aumentan los falsos positivos.

Tomando el modelo elaborado con la muestra equilibrada, se tiene que el 84 % de los casos de fraude está correctamente clasificado y el 89 % de los casos detectados como fraudulentos realmente lo son. Esta última medida mejora claramente respecto a la correspondiente del bagging, donde apenas alcanzaba el 70 % para la muestra equilibrada. En efecto, el random forest supera al modelo bagging.

El modelo random forest elabora una salida muy valiosa que se corresponde con la importancia de las variables en la creación del modelo. Se extrae para el random forest elaborado con la muestra de entrenamiento equilibrada.

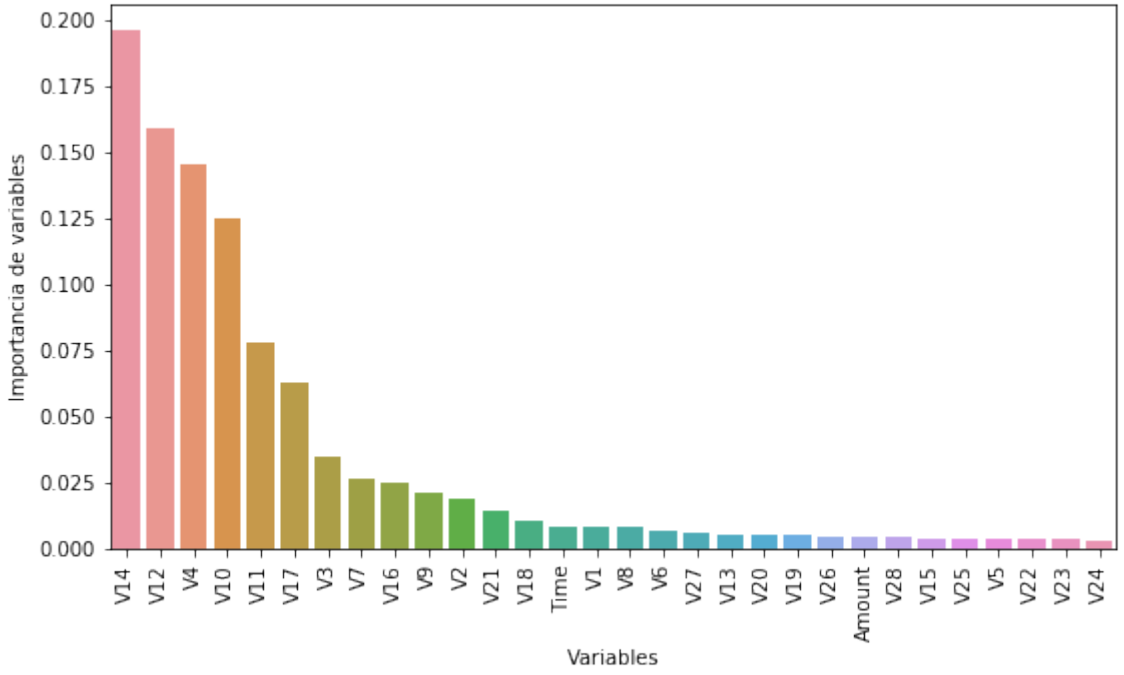


Figura 18: Importancia de variables para random forest con SMOTE

Las variables que cobran mayor importancia en el modelo son, por este orden, V14, V12, V4 y V10. Se recuerda que todas ellas fueron resaltadas al estudiar su función de densidad por tipo de transacción. Además, tanto V14 como V12 tenían un índice de correlación negativo con *Class* destacable frente a los demás.

4.3.4. Extremely randomized trees

Este modelo no induce aleatoriedad en la muestra con la que se crea cada árbol, pues cada uno de ellos se construye con la muestra de entrenamiento al completo.

Sin embargo, sí que toma de manera cuasi-aleatoria las condiciones de separación de los nodos. Esto provoca que los árboles no sobreajusten con tanta facilidad. Por ello, también se deja que crezcan hasta su máxima extensión posible.

Se elaboran 100 árboles y, como en el random forest, sólo se puede tomar la condición de sepación entre $\sqrt{M}$ variables.

Extremely randomized trees sin SMOTE

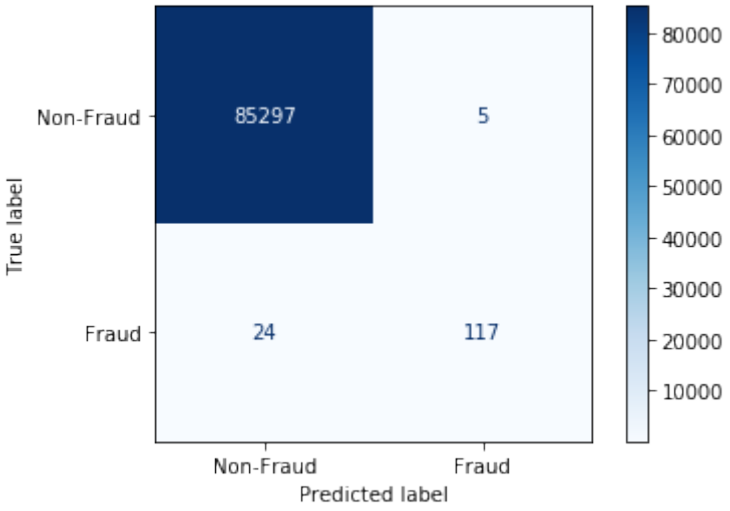


Figura 19: Matriz de confusión del extremely randomized trees sin SMOTE

Extremely randomized trees con SMOTE

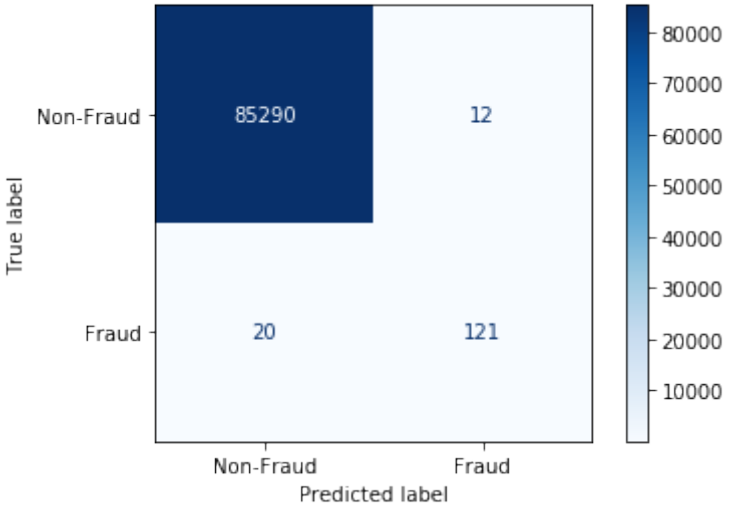


Figura 20: Matriz de confusión del extremely randomized trees con SMOTE

	con SMOTE	sin SMOTE
Precision	0.91	0.96
Recall	0.86	0.83
F1Score	0.88	0.89

Tabla 5: Medidas de rendimiento del extremely randomized trees

Este modelo es el que ofrece una mejor clasificación. Alcanza el 86 % de *recall* con la muestra equilibrada y disminuye aún más los falsos positivos. Además, su aleatoriedad permite que el modelo se ejecute rápido.

De nuevo, la muestra equilibrada detecta más casos de fraude a cargo de clasificar erróneamente transacciones legítimas como fraudulentas.

Se pasa a probar el modelo de aprendizaje no supervisado isolation forest. Ahora se prescinde de la información que indica de qué tipo es cada transacción para la elaboración del modelo. Sí que se hará uso de dicha información al probar su rendimiento con la muestra test, que además permitirá la comparación con el resto de modelos.

4.3.5. Isolation forest

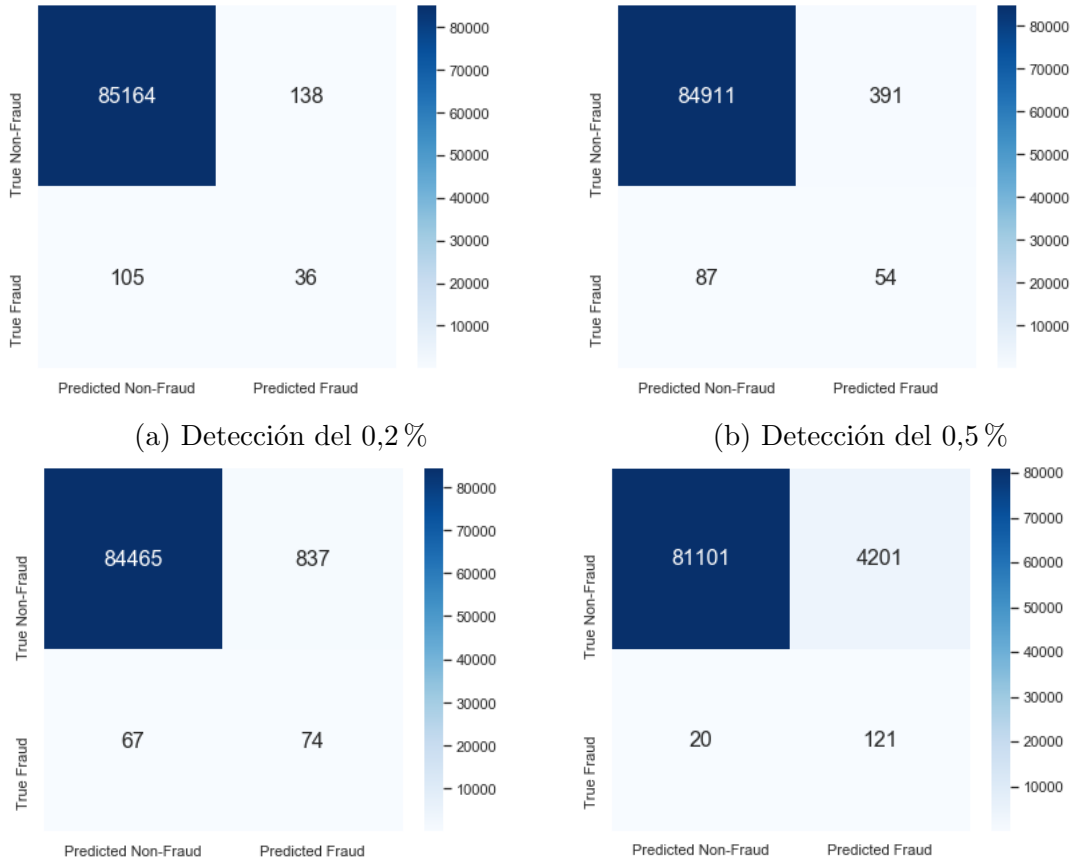
El isolation forest propone una forma alternativa de detectar de fraude. Asume que hay una relación directa entre las transacciones fraudulentas y los comportamientos anómalos. Por ello, plantea la detección de anomalías como equivalente a la detección del fraude.

Al igual que con los modelos anteriores, se va a elaborar con el conjunto de entrenamiento y se va a probar con el conjunto test. Sin embargo, ahora no se necesita de una muestra equilibrada. De hecho, en principio, es bueno que haya pocas transacciones fraudulentas, así serán filtradas como anómalas con mayor facilidad.

Como se ha mencionado con anterioridad, es necesario indicar el porcentaje de observaciones anómalas que se quiere detectar. En este caso, se parte con ventaja, al disponer de la muestra ya etiquetada. En el análisis de los datos, se determinó que las transacciones fraudulentas conformaban el 0,17 % de la muestra total. Luego, tiene sentido tomar esta cantidad como porcentaje de observaciones más anómalas a detectar. Normalmente no se dispone de este dato.

Se va a comparar el rendimiento del modelo con distintos valores para el hiperparámetro mencionado. En concreto, se va a comparar para el 0,2 %, el 0,5 %, el 1 % y el 5 %.

A continuación, aparecen las matrices de confusión para cada uno de los modelos desarrollados.



(a) Detección del 0,2 %

(b) Detección del 0,5 %

(c) Detección del 1 %

(d) Detección del 5 %

Figura 21: Matriz de correlaciones de isolation forest

A medida que aumenta el porcentaje de anomalías que se retiene, aumenta el número de casos de fraude que se detectan. No obstante, aumenta también el número de falsos positivos, siendo este demasiado elevado.

4.4. Comparación de los modelos

Los cinco modelos utilizan árboles de decisión. Sin embargo, se observa cómo la efectividad alcanzada por cada uno es diferente.

El modelo de aprendizaje no supervisado isolation forest presenta una clara desventaja frente a los demás. Para alcanzar un número similar a los otros modelos de casos de fraude detectados correctamente, clasifica erróneamente muchas transacciones legítimas. Por lo tanto, este modelo queda descartado.

Los otros modelos corresponden al aprendizaje supervisado y la siguiente tabla presenta el rendimiento de cada modelo para cada una de las dos muestras empleadas.

Muestra	Modelo	*Precision*	*Recall*	*F1Score*
Sin SMOTE	CART	0.84	0.80	0.82
	Bagging	0.95	0.80	0.87
	Random forest	0.97	0.83	0.89
	Extremely randomized trees	0.96	0.83	0.89
Con SMOTE	CART	0.39	0.84	0.53
	Bagging	0.70	0.82	0.76
	Random forest	0.89	0.84	0.87
	Extremely randomized trees	0.91	0.86	0.88

Tabla 6: Comparación de modelos

Se recuerda que *recall* es la proporción de transaccion fraudulentas que se encuentran bien clasificadas y que *precision* es la proporción de transacciones realmente fraudulentas dentro de las catalogadas como tal. *F1Score* es una media ponderada de las dos.

Ante todo, se pretende frenar el fraude. Por supuesto, es posible que al tratar de filtrar las transacciones fradulentas, se retengan también transacciones completamente legales. No se desea que esto suceda, pero esta confusión podría ser fácilmente comprobada y aclarada sin mayores consecuencias. En cambio, cada caso de fraude no detectado va acompañado de pérdidas y de generación de desconfianza. Por eso, se presta especial atención a *recall*, queriendo que se maximice todo lo posible.

Se observa cómo, con carácter general, *recall* es mayor en los modelos creados con la muestra equilibrada que en los modelos creados con la muestra no equilibrada. Luego, como se intuía, ciertamente, si se hace creer al algoritmo que la proporción de acciones fraudulentas es superior, el modelo tenderá a clasificar más transacciones como tal. No obstante, *precision* disminuye con la muestra equilibrada. Esto no resulta un gran incoveniente en la mayoría de los casos, aunque considero que contar con un *precision* de 0.39 no es tolerable. Este resultado se tiene en el árbol de decisión para la muestra equilibrada. Implicaría que sólo el 39 % de las transacciones clasificadas como fraudulentas realmente lo son.

Destacaría que los modelos bagging, random forest y extremely randomized trees consiguen controlar el número de falsos positivos. Alcanzan valores de *precision* muy altos, siendo superiores en todos ellos para la muestra no equilibrada.

Otro dato de interés es que, en ninguna de las dos muestras, la puesta en común de varios árboles en el modelo bagging consigue detectar más casos de fraude que un sólo árbol de decisión con profundidad máxima. Sí que consigue mejorar la detección el random forest y superar reseñablemente el valor de *F1Score*. En el modelo bagging no se añade profundidad máxima, así puede cuantificarse cuánto añade la selección aleatoria de un conjunto de variables en cada nodo que se da en el random forest.

No obstante, el modelo que más casos de fraude detecta es el extremely randomized trees elaborado con la muestra equilibrada. Clasifica correctamente el 86 % de las transacciones fraudulentas y prácticamente el 100 % de las legítimas. Además, al ser un modelo que introduce mucha aleatoriedad controla en gran medida la varianza y es relativamente rápido de compilar. Luego, para este ejemplo, el uso del modelo extremely randomized trees equilibrando la muestra con SMOTE ha destacado sobre los demás.

5. Conclusiones

Las cifras demuestran que es necesario frenar el fraude bancario, pues ocasiona pérdidas muy elevadas a las entidades financieras. Aunque el problema existe desde hace varias décadas, ha cobrado peso en los últimos años debido a que el progreso tecnológico ha facilitado la innovación en las estrategias de engaño.

Por ello, es preciso actualizar periódicamente los sistemas de detección del fraude, que quedan obsoletos rápidamente. Ante esta realidad, se opta por la inteligencia artificial, que construye modelos capaces de ir adaptándose a las circunstancias cambiantes.

En el análisis previo de los datos, se ha podido concluir que, tanto el importe de la transacción, como el momento del día en el que se realiza impactan en la probabilidad de fraude.

Se ha observado que, las transacciones fraudulentas suelen ser de importes no muy elevados, de manera que pueden pasar desapercibidas. Aunque pueda llamar la atención, las transacciones fraudulentas tienden a no superar los 5.000 dólares.

Por otro lado, se han extraído conclusiones claras sobre la correlación entre la probabilidad de fraude y el momento del día en que se realiza la transacción. Hay un intervalo de cinco horas diarias en el que, si bien apenas se producen transacciones bancarias, el fraude se encuentra presente con la misma frecuencia que en otros momentos del día. Es probable que este intervalo de tiempo se corresponda con la noche.

Los modelos de inteligencia artificial elaborados son costosos computacionalmente y de alta complejidad. No es sencillo explicar los detalles del modelo, ya que se pierde el control de lo que hace internamente. Por ello, para ofrecerlo como producto, habría que basarse en su matriz de confusión.

Teniendo en cuenta el resultado ofrecido por el modelo de aprendizaje no supervisado, cabe mencionar que el fraude no necesariamente se corresponde con observaciones altamente extrañas. Por tanto, conviene enfocar el problema desde la búsqueda de patrones en las transacciones que se saben fraudulentas.

En el presente trabajo se han construido modelos de aprendizaje supervisado con notable rendimiento. El mejor modelo desarrollado clasifica correctamente el 86 % de las transacciones fraudulentas y prácticamente el 100 % de las legítimas. Para aumentar la fiabilidad, sería convenientes archivar un mayor número de transacciones fraudulentas, de modo de que se capture mejor su comportamiento e incremente por tanto la eficacia del modelo.

6. Bibliografía

Aleskerov, E., Freisleben, B. y Rao, B. (1997) Cardwatch: A Neural Network Based Database Mining System for Credit Card Fraud Detection. Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr). DOI: 10.1109/CIFER.1997.618940.

Archer K.J. y Kimes, R.V. (2008) "Empirical characterization of random forest variable importance measures" Computational Statistics and Data Analysis, 52, 2249–2260.

Bhattacharyya, S., Jha, S., Tharakunnel, K. y Westland, C. (2011) "Data mining for credit card fraud: A comparative study" Decision Support Systems, 50, 602-613.

Bowyer, K.W., Chawla, N.V., Hall, L.O. y Kegelmeyer, W.P. (2002) "SMOTE: Synthetic Minority Over-sampling Technique" Journal Of Artificial Intelligence Research, 16, 321-357.

Breiman, L., Friedman, J. H. y Olshen, R. (1984) Classification and Regression Trees, Wadsworth International Group, Belmont.

Breiman, L. (2001) "Random forests" Machine Learning, 45, 5–32.

Domingues, R., Filippone, M., Michiardi, P. y Zouaoui, J. (2018) "A comparative evaluation of outlier detection algorithms: Experiments and analyses" Pattern Recognition, 74, 406-421.

Ernst, D., Geurts, P. y Wehenkel, L. (2006) "Extremely randomized trees" Machine Learning, 63, 3–42.

Friedman, J., Hastie, T. y Tibshirani, R. (2009) The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer, Stanford.

Hotelling, H. (1933) "Analysis of a complex of statistical variables into principal components" Journal of Educational Psychology, 24, 417–520.

Hyder, J. y Sameena, N. (2019) "Credit card fraud detection using local outlier factor and isolation forest" International Journal of Computational Science and Engineering, 7, 1060-1064.

Khare, N. y Sait, S.Y. (2018) "Credit card fraud detection using Machine Learning models and collating Machine Learning models" International Journal of Pure and

Applied Mathematics, 118, 825–838.

Liu, F. T., Ting, K. M., y Zhou, Z. H. (2008) "Isolation forest" Proceedings of IEEE International Conference on Data Mining, 8, 413-422.

Specht, D. (1997) "A General Regression Neural Network" IEEE Transactions on neural networks, 2, 568-576.

7. Anexo

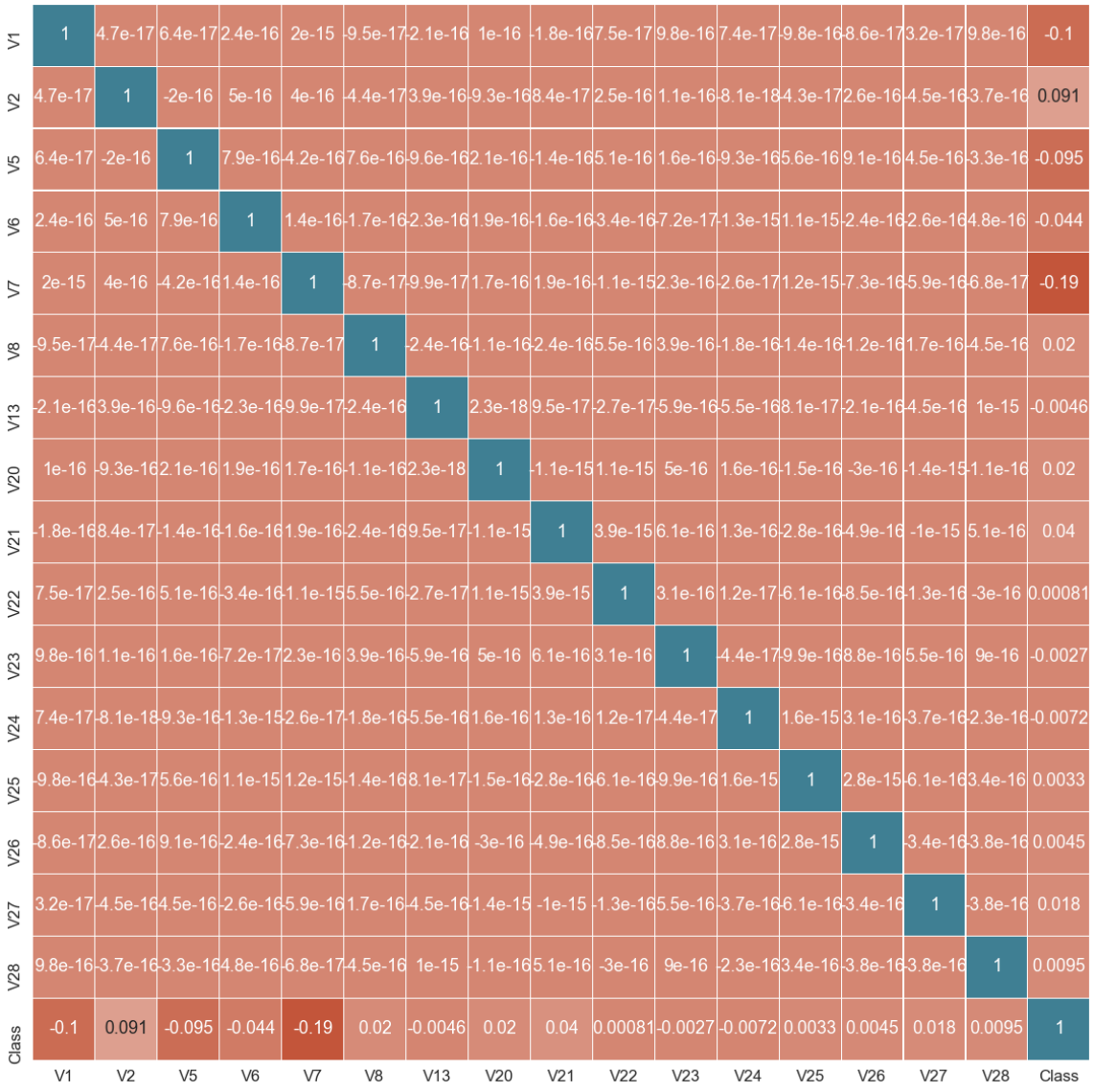


Figura 22: Matriz de correlaciones

Código Python para la elaboración de los modelos.

```
# Se importa el fichero
df = pd.read_csv("creditcard.csv")

# Se divide la muestra en muestra de entrenamiento y muestra test
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3, random_state=10)

# Uso de SMOTE para equilibrar la muestra
sm = SMOTE(random_state=10)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())

##### Modelos
```

```
## CART
tree_params = {"max_depth": list(range(1,20))}
grid_tree = GridSearchCV(DecisionTreeClassifier(), tree_params,
    scoring = 'recall')

# sin SMOTE
grid_tree.fit(X_train, y_train)
m_best1 = grid_tree.best_estimator_
predictions_cart1 = m_best1.predict(X_test)

# con SMOTE
grid_tree.fit(X_train_res, y_train_res)
m_best2 = grid_tree2.best_estimator_
predictions_cart2 = m_best2.predict(X_test)

## BAGGING
bagging = BaggingClassifier(
    DecisionTreeClassifier(max_depth = 10,
                           criterion = 'gini'),
    n_estimators = 10,
    random_state = 10)

# sin SMOTE
model_bag1 = bagging.fit(X_train, y_train)
predictions_bag1 = model_bag1.predict(X_test)

# con SMOTE
model_bag2 = bagging.fit(X_train_res, y_train_res)
predictions_bag2 = model_bag2.predict(X_test)

## RANDOM FOREST
rf = RandomForestClassifier(n_estimators = 100,
    max_features = 'sqrt',
    random_state = 10)

# sin SMOTE
model_rf1 = rf.fit(X_train, y_train)
predictions_rf1 = model_rf1.predict(X_test)

# con SMOTE
model_rf2 = rf.fit(X_train_res, y_train_res)
predictions_rf2 = model_rf2.predict(X_test)

## EXTREMELY RANDOMIZED TREES
extra = ExtraTreesClassifier(n_estimators = 100,
    max_features = 'sqrt',
    criterion = 'gini',
    random_state = 10)

# sin SMOTE
model_extra1 = extra.fit(X_train, y_train)
predictions_extra1 = model_extra1.predict(X_test)

# con SMOTE
```

```
model_extra2 = extra.fit(X_train_res, y_train_res)
predictions_extra2 = model_extra2.predict(X_test)

## ISOLATION FOREST
isf1 = IsolationForest(n_estimators=100, contamination=0.002,
    random_state=10, verbose=0)
isf2 = IsolationForest(n_estimators=100, contamination=0.005,
    random_state=10, verbose=0)
isf3 = IsolationForest(n_estimators=100, contamination=0.01,
    random_state=10, verbose=0)
isf4 = IsolationForest(n_estimators=100, contamination=0.05,
    random_state=10, verbose=0)
y_pred_isf1 = model_isf1.predict(X_test)
y_pred_isf2 = model_isf2.predict(X_test)
y_pred_isf3 = model_isf3.predict(X_test)
y_pred_isf4 = model_isf4.predict(X_test)
```