MODELO PREDICTIVO DE DETECCIÓN DE FRAUDE EN TARJETAS BANCARIAS

Sergio Mañanas López

Juan José Hernández Villar

Proyecto final

presentado en el Bootcamp de Data Science

en Neoland, Madrid

Diciembre 2020

Introducción

El problema de fraude en tarjetas bancarias afecta tanto a bancos, comercios y clientes. Es una carrera entre las instituciones financieras y los defraudadores, los primeros desarrollando nuevas prácticas para evitar los pagos fraudulentos como, por ejemplo, la doble autentificación de las transacciones desarrollada en la directiva de pagos PSD2, Know Your Customer, o aplicaciones basadas en Machine Learning para detener el fraude en base a patrones de estas transacciones. Por parte de los segundos, nuevas formas de apropiarse de los datos de los clientes y patrones de transacciones más difíciles de detectar.

Este proyecto lo elegimos por nuestro interés en banca y los procesos de detección de fraude en pagos, blanqueo de capitales o financiación del terrorismo. Los datos de entrenamiento y prueba obtenidos nos permitían satisfacer este objetivo, además de ser un reto por sus características. Valoramos entonces usar la metodología de desarrollo de un proyecto de ciencia de datos eligiendo las fases de CRISP-DM que aplicaban para este proyecto final de dos semanas del Bootcamp de Data Science. Esta memoria se estructura en base a esa metodología.

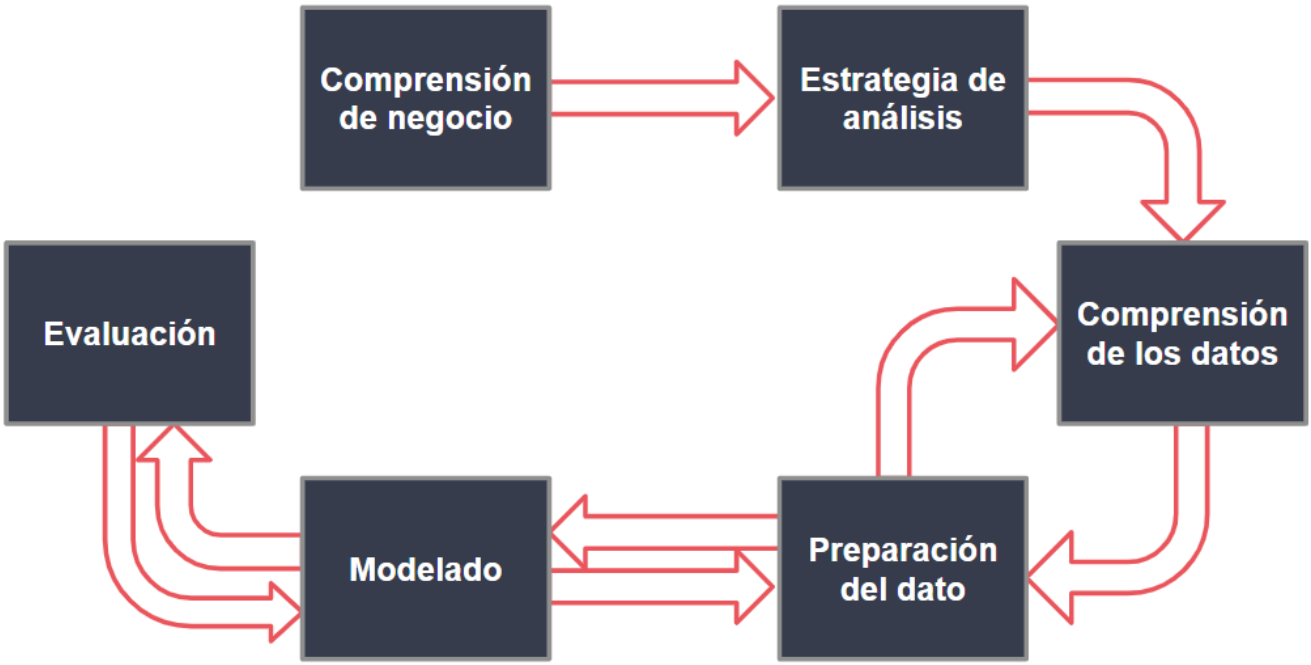


Gráfico 1. Metodología CRISP utilizada en el proyecto.

Tabla de Contenidos

1. Comprensión de Negocio

El **fraude en tarjetas** consiste en el uso no autorizado de una tarjeta de crédito o débito para adquirir productos o dinero. Existen varias formas de obtener los datos de una tarjeta:

- Perdida o robo de la tarjeta física.
- Robo del número de la tarjeta, pin y código de seguridad. Se puede cometer fraude sin tener acceso físico a la tarjeta. La entrada en sitios web poco seguros e ilegítimos, la respuesta a ofertas no solicitadas o incluso peticiones por correo electrónico y teléfono son fuentes de este tipo de robo de datos.
- Robo de la identidad del defraudado con lo que puede cometerse fraude haciendo uso de sus tarjetas llegando incluso a abrir nuevas cuentas o tarjetas. Estos datos pueden acabar siendo puestos a la venta en la *dark web*.
- "*Skimming*" o clonado de la tarjeta en cajeros o terminales de punto de venta (TPV) a través de la lectura de la banda magnética.[1]
- "*Shimming*" o clonado de la tarjeta a través de la lectura y grabado de los datos del chip, medida de seguridad adicional que tampoco está exenta de robo. No se clona el chip de la tarjeta, sino que se reproduce una tarjeta con banda magnética con los datos obtenidos o se utilizan los datos en compras online.[2]

El robo no presencial de la tarjeta es el más frecuente actualmente, dónde su venta y uso por terceros puede demorarse semanas o meses, lo que lo hace más difícil su detección inmediata, hecho que ocurre al empezar a verse transacciones no autorizadas.

Al igual que el medio para la obtención de datos de las tarjetas es más común a través de la tecnología en vez de la apropiación física, también el fraude cometido con éstos es ya más frecuente en transacciones online (no presencial), que en físicas (presencial).[3]

El fraude no presencial es actualmente un 81% mayor que en un terminal de punto de venta. Las pérdidas provocadas por el fraude presencial y no presencial en todo el mundo

[1] Poremba, S. (2019). How to Protect Yourself Against Card Skimmers at Gas Stations. Retrieved 30 December 2020, from https://www.experian.com/blogs/ask-experian/how-to-protect-yourself-against-credit-card-skimmers-at-gas-stations/

[2] Mangla, I. (2018). Shimming Is the Latest Credit Card Scam. Retrieved 30 December 2020, from https://www.experian.com/blogs/ask-experian/shimming-is-the-latest-credit-card-scam/

[3] Fraud and Security Practice Area. (2020). Retrieved 29 December 2020, from https://www.javelinstrategy.com/coverage-area/fraud-security

fueron de 24.260 millones de dólares en el mundo en 2018, un 18% más que el año anterior, de los cuales 88 millones solo en España.[4]

Nuestro proyecto es un Modelo Predictivo de Detección de Fraude que se enfoca en este grupo mayoritario de transacciones realizadas en plataformas de e-commerce. El modelo de aprendizaje automático debe aprender de aquellos patrones distintivos etiquetados como fraudulentos para que, una vez desplegado en producción, sea capaz de detectarlos y detenerlos evitando perjuicios a todos los participantes involucrados.

El coste del fraude

Un modelo de detección de fraude debe saber detectar y detener las transacciones fraudulentas. Esto permite tanto evitar que tanto el banco, el comercio o el cliente se haga cargo del fraude y poder reducir sus departamentos de gestión de este tipo de incidencias. Adicionalmente, debe evitar en lo posible los falsos positivos ya que estos causan un perjuicio o incomodidad al cliente, impactando en su satisfacción con la entidad.

Entidades financieras

- Las entidades cubren el 72% de las perdidas según las estadísticas de fraude en tarjetas de shiftprocessing.com. El resto los adquirientes de los cajeros y los comercios.
- Costes involucrados en la implantación de un sistema automático de detección de fraude.
- Costes de crear y mantener un equipo que cubra las incidencias que no pueda detectar el sistema anterior.
- Coste reputacional y de satisfacción de los clientes.

Comercios

Los comercios, también tienen su responsabilidad en evitar fraude en las transacciones realizadas en sus locales o en sus páginas web y tienen costes asociados a ello:

- Deben pagar servicios de los emisores de tarjeta para evitar que se les imputen los cargos fraudulentos.
- Perseguible legalmente si no se toman las medidas adecuadas, exclusión del sistema de pago.

[4] Credit Card Fraud Statistics. (2020). Retrieved 30 December 2020, from https://shiftprocessing.com/credit-card-fraud-statistics/

- Costes reputacionales por la pérdida de confianza del consumidor.
- Se le pueden repercutir costes cómo comisiones o diferencia por el cambio de moneda.

Clientes

Por normal general, a no ser que se demuestre que el cliente ha tenido un comportamiento negligente grave, el fraude de tarjetas se imputa en su totalidad a las entidades financieras. Sin embargo, también existen perjuicios al cliente:

- En caso de robo o pérdida, la nueva normativa de pagos PSD2 limita el cargo al cliente de los primeros 50 euros defraudados antes de la comunicación de estos hechos. En caso de robo de los datos, la totalidad le corresponde a la entidad.[5]
- Aunque la carga de la prueba corresponde al banco, para anular el cargo es necesario seguir una serie de gestiones con el banco y policía.
- Inseguridad, frustración, estrés y desconfianza.
- No poder hacer uso del medio de pago hasta que llegue una nueva tarjeta.
- Indirectamente, y no menor, los comercios y las instituciones trasladan estos costes a los clientes incrementando sus precios, comisiones e intereses.

El coste del error en la detección del verdadero fraude

El falso positivo (en nuestro proyecto lo denominamos falso fraude) lleva a que la entidad financiera o el comercio pare una transacción cuyo patrón puede hacer sospechar de fraude. El cliente se ve perjudicado si no puede realizar una transacción legítima. Si esto es muy frecuente, puede llevar a la pérdida de confianza del cliente y a abandonar el comercio o la entidad financiera.

Para la entidad financiera, conlleva los costes de tener que dotar al equipo de gestión de incidencias para poder gestionar las llamadas de los clientes.

Un modelo predictivo debe tener en cuenta esta situación. Modelos deficientes pueden conllevar el que, para conseguir detectar un porcentaje alto de fraude, arrastre también un porcentaje alto de falsos positivos. En Estados Unidos en 2017, por ejemplo, 4 de cada 10 clientes recibieron alertas de transacciones fraudulentas cuando eran legítimas.[6]

[5] Psd2 Seguridad Pagos Todo lo que hay Saber Directiva Europea Psd2. (2020). Retrieved 30 December 2020, from https://bancaelectronica.net/que-es-psd2/

[6] Mecia, T. (2017). Credit card fraud alerts surge, false alarms still common. Retrieved 30 December 2020, from https://www.creditcards.com/credit-card-news/credit-card-fraud-alert-poll/

Para poder compaginar una alta detección de fraude minimizando los falsos positivos, en un conjunto de transacciones cuyas clases están fuertemente desbalanceadas se utilizarán modelos de clasificación de boosting. En los datos de prueba utilizados un 3,5% de los registros son fraude. Sin embargo, en datos reales, por ejemplo, en España, esta ratio podría llegar a ser hasta 100 veces inferior.

2. Estrategia de análisis

En el contexto de los problemas de clasificación, si una categoría tiene más representantes que otras categorías, significa que el conjunto de datos no está equilibrado (datos desbalanceados). En cuanto a la clasificación, este problema puede generar sesgos de aprendizaje perjudicando a los grupos minoritarios, que suelen contener los casos más interesantes para estudiar. El problema principal en la detección de fraude es el desbalanceo de clases: esta situación se conoce como el **Problema del Desequilibrio de Clases** (Class Imbalance Problem).

Es destacable que en los sistemas de detección de fraude están incrementando el número de falsos positivos, producto de los cambios en los perfiles de comportamiento de empleados y clientes. Todo ello afecta a la monitorización del fraude multicanal con perfil de comportamiento de clientes, a los modelos de monitorización transaccional de blanqueo de capitales por desviación del comportamiento histórico y a los de detección del abuso de mercado por el aprovechamiento ilícito de hechos relevantes sobre los emisores.[7]

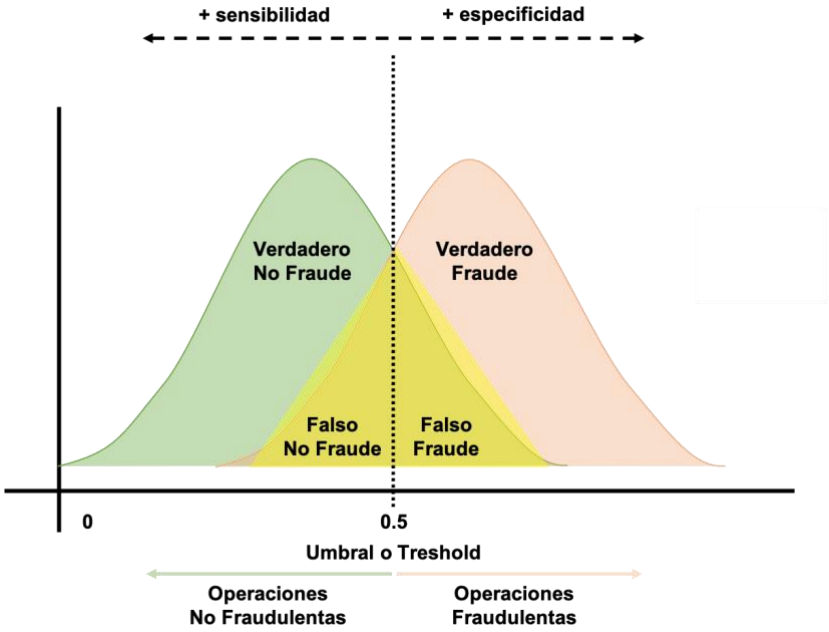
No hay que olvidar que **el fraude es un evento excepcional.** En los datos de nuestro problema, las transacciones consideradas fraudulentas (variable *isFraud* = 1) representan únicamente un 3.5% del total de transacciones. Para poder resolver este problema de clasificación se confía en la capacidad de los clasificadores con algoritmos de boosting por lo que no se plantean técnicas de oversampling.

Los **clasificadores de boosting** se basan en el hecho de que encontrar muchas reglas simples es mucho más fácil que encontrar una sola regla con alta precisión para la predicción. Para aplicar el método boosting, comenzaremos seleccionando un algoritmo para encontrar estas reglas simples. El algoritmo boosting entrena a estos clasificadores básicos o débiles repetidas veces, usando un conjunto de entrenamiento diferente cada vez. De esta manera, "El clasificador básico genera una nueva regla de predicción débil, y después de muchas iteraciones, el algoritmo boosting debe combinar estas reglas débiles en una única regla de predicción que, se espera, sean mucho más precisa que ninguna de las reglas débiles." (Basulto Santos & García del Hoyo, 2009)

[7] Faraco, F. (2020). El impacto de la pandemia en el fraude de las entidades financieras. Retrieved 27 December 2020, from https://cincodias.elpais.com/cincodias/2020/04/30/opinion/1588245588_987892.html

Tras un estudio previo de los distintos algoritmos de boosting existentes, nos decidimos a utilizar LightGBM y XGBoost. De sus ventajas e inconvenientes, así como del funcionamiento de estas se detallará en el apartado *Modelado* respectivo a cada clasificador. Para comprobar la eficacia para detectar fraude en nuestros modelos de boosting, **la métrica a utilizar es el área bajo la curva ROC (AUC).**

La definición de una curva ROC (Receiver Operating Characteristic o Característica Operativa del Receptor) es una representación gráfica de la sensibilidad frente a la especificidad para un sistema clasificador binario según se varía el umbral de discriminación. Es decir, la proporción de verdaderos positivos frente a la proporción de falsos positivos según se varía el umbral de discriminación o treshold (valor a partir del cual decidimos que un caso es un positivo).[8]



Grafica 2. Distribución de los resultados de los modelos.

La métrica utilizada AUC (Area under the ROC Curve o área bajo la curva ROC) mide toda el área por debajo de la curva ROC completa. El AUC proporciona una medición agregada del rendimiento en todos los umbrales de clasificación posibles. El AUC es conveniente por las dos razones siguientes:

- El AUC es invariable con respecto a la escala. Mide qué tan bien se clasifican las predicciones, en lugar de sus valores absolutos.

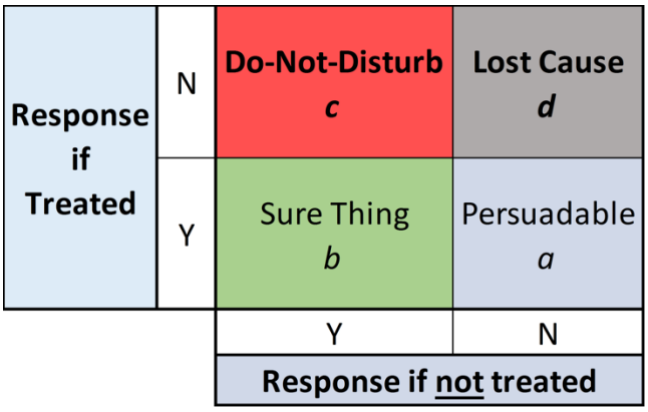
[8] Curva ROC. (2020). Retrieved 27 December 2020, from https://es.wikipedia.org/wiki/Curva_ROC

- El AUC es invariable con respecto al umbral de clasificación. Mide la calidad de las predicciones del modelo, sin tener en cuenta qué umbral de clasificación se elige.

La métrica AUC consideramos que es la más adecuada para comparar clasificadores en problemas de muestra desbalanceada. La elección de esta métrica se ampara en su uso frecuente ante casos de modelos de aprendizaje automático con clases desbalanceadas.[9]

Analizaremos los resultados a través de **Uplift modelling o modelo de elevación** para segmentar las transacciones y prescribir sobre qué cuantiles detener para maximizar la detección de fraude reduciendo los falsos positivos. Se define como una técnica de modelización predictiva que modela directamente el impacto incremental de un tratamiento en el comportamiento de un individuo respecto a la gestión de relaciones con los clientes.

Dado que el Upliftting se centra únicamente en las respuestas incrementales de manera que se puede medir el efecto de la agrupación, proporciona casos de gran rentabilidad de la inversión (ROI, Return On Investment o retorno de la inversión). Uno de los usos más eficaces, para nuestro caso en particular, es la eliminación de los efectos negativos de estas campañas de control de fraude. Tanto en la industria de las telecomunicaciones como en la de los servicios financieros, a menudo estas campañas pueden provocar que los clientes cancelen un contrato o una póliza. El Upliftting permite que estos clientes, pertenecientes al grupo "No Molestar" (los falsos positivos del modelo), sean eliminados de la campaña.



Grafica 3. Matriz de agrupación de los usuarios según el modelo de Uplift.

[9] Algunos de los artículos:

- Cruz-Reyes, H., Reyes-Nava, A., Rendón-Lara, E., & Alejo, R. (2018). *Estudio del desbalance de clases en bases de datos de microarrays de expresión genética mediante técnicas de Deep Learning.* Universidad Autónoma del Estado de México.
- Espinar Lara, R. (2018). *Modelos de Clasificación con datos no balanceados.* Universidad de Sevilla.

3. Comprensión de los datos

Los datos corresponden a un conjunto de transacciones realizadas por comercio electrónico reales proporcionadas por Vesta Corporation, durante aproximadamente **6 meses de datos de entrenamiento y 6 meses de prueba**. Estos datos forman parte de la competición de Kaggle IEEE-CIS Fraud Detection de la IEEE Computational Intelligence Society.

Los datos de entrenamiento suponen más de medio millón en los registros, siendo los de prueba en la misma proporción. De esto, se ha deducido que sólo se han seleccionado una muestra de tarjetas y sus transacciones de toda la población que puede haberse ejecutado en 12 meses.

Los registros representan transacciones a lo largo del tiempo. Aunque sí que tienen un carácter temporal, el problema no es de series temporales. Es un problema de cambio en los patrones de comportamiento en el uso de la tarjeta. Éste es el carácter temporal que debe incluirse en el registro de la transacción, para permitir detectar el fraude, bien a partir de las columnas V, bien reforzándolo creando nuevas variables por ingeniería (tiempos entre transacciones, frecuencia, dispersión en los importes, distancias, etc.).

Hay 435 columnas de características, con toda la información encriptada para proteger la identidad de cuentas y clientes. Esto también conlleva que no sólo el contenido de los registros esté enmascarado, sino también que muchas de las columnas tienen significado funcional oculto o directamente se han creado por ingeniería de variables a partir de otras. Lo que es lo mismo, la naturaleza de las columnas no se puede derivar de su contenido ni viceversa. Vesta sí que proporciona cual es el sentido funcional de los grupos de columnas a grandes rasgos.

Breve descripción de identity.csv

Un tercio de las transacciones del conjunto "*transaction*" tienen un registro de "*identity*". Además, en el EDA se observa que prácticamente todas las columnas de este conjunto tienen más de un 70% de nulos al unir los dos conjuntos. Por evitar exceso de imputación de valores NaNs, así como el límite de tiempo del proyecto, se decide descartar éste dataset y trabajar solo con el conjunto "*transaction*".

Breve descripción de transaction.csv

Columna / grupo de Columnas	Tipo	Descripción
TransactionID	Numérica	Identificador único de transacción
isFraud	Categórica	Es el target. Cabe decir que las transacciones se etiquetan al confirmar por parte de los clientes el fraude. Una vez una transacción se etiqueta como fraudulenta, todas las transacciones subsiguientes se considerarán como fraude.
TransactionDT	Numérica	Delta de tiempo (no fecha) referenciado al inicio del conjunto de datos de entrenamiento. Se observa que posiblemente las transacciones se inician el 1 de diciembre de 2017.
TransactionAMT	Flotante	Importe de la transacción.
ProductCD	Categórica	Código (categoría) de producto o servicio de la transacción. Enmascarado, categórica.
card1 - card6	Categórica	Información de la tarjeta como tipo, categoría, país, banco emisor. Son categóricas, están enmascaradas y se ve en el EDA que no identifican únicamente una tarjeta.
addr1 - addr2	Categórica	Código Zip y país de facturación de la tarjeta.
dist1 - dist2	Numérica	Distancias, que pueden significar distancia entre dirección de facturación, recepción, IP, área telefónica. Numérica.
P_emaildomain R_emaildomain	Categórica	Dominio del correo del comprador y del receptor del producto.
C1 - C14	Numérica	Cada columna puede indicar el número de direcciones asociadas a la tarjeta, o dispositivos, o direcciones IP. Está enmascarado, no se sabe a ciencia cierta.
D1 - D15	Numérica	Delta de tiempo. Entre ellas, parecen estar el tiempo contado en días de primera transacción o emisión (D1) y la transacción o de ésta con la anterior transacción de la misma tarjeta (D3).
M1 - M9	Categórica	Columnas de coincidencias, nombre en la tarjeta o dirección con la de la transacción, etc. Son categóricas.
V1 – V339	Numérica	337 columnas. Ingeniería de variables de Vesta. Contadores, rankings o relaciones con otras entidades, entre otras. Por ej. cuantas veces se ha asociado la tarjeta con una IP o dirección e-mail en las últimas 24 horas, aunque no están descritas tampoco una a una. Son numéricas.

EDA (Exploratory Data Analysis)

Exploración de las columnas y estrategia de reducción de características.

Todas las columnas del conjunto train_transaction.csv se analizaron, determinando su relación con el target fraude, su evolución en el tiempo, correlaciones entre ellas y número de nulos.

En la exploración, se propuso una reducción de columnas en base a tipo de columna (D, C, M, V). Se seleccionaron subconjuntos de éstas en base a poseer el mismo número de NaNs, bajo la hipótesis de que contenían información más relacionada por la ingeniería de variables por las que se crearon. Seguidamente, se obtuvo la correlación de Pearson de cada subconjunto, agrupando de nuevo aquellas variables que poseían una correlación superior al 0.7%. Para estos grupos, se seleccionaba únicamente aquella columna con mayor número de valores únicos asumiendo que a mayor número de valores únicos, una mayor explicatividad.

El producto de esto fue un conjunto reducido de 133 columnas a informar al modelo. En paralelo, este conjunto se confirmó o modificó con los resultados de la importancia de características obtenidas del entrenamiento del modelo.

Problema de la detección de los campos cliente y/o tarjeta

Desde el inicio del análisis de los datos, se nos pone de manifiesto la dificultad de definir dentro del juego de datos cuales de estos componen la tarjeta o cliente. Quizá el principal reto que nos encontramos al inicio del proyecto. Para ello hicimos durante la fase de análisis exploratorio de los datos una exploración manual de todos los campos susceptibles de identificar individualmente la cuenta o tarjeta, pasando más tarde a un estudio más estructurado basado en la Validación Adversaria (Adversarial Validation).

En un primer término, debido al tiempo limitado para realizar un estudio más exhaustivo, se tomaron dos de las propuestas de otros participantes en la competición:

```
train['uids'] = train['card1'].astype('str')   + '_' + train['addr1'].astype('str')  + '_' \
              + train['D1achr'].astype('str')
```

Imagen 1. Líneas de código Python sobre la creación de la primera identificación de tarjeta.

```
train['uid1'] = train['card1'].astype('str') + '_' + train['card2'].astype('str')     + '_' \
              + train['card3'].astype('str') + '_' + train['card4'].astype('str')     + '_' \
              + train['card5'].astype('str') + '_' + train['card6'].astype('str')     + '_' \
              + train['addr1'].astype('str') + '_' + train['addr2'].astype('str')     + '_' \
              + train['ProductCD'].astype('str') + '_' + train['D1achr'].astype('str')
```

Imagen 2. Líneas de código Python sobre la creación de la segunda identificación de tarjeta.

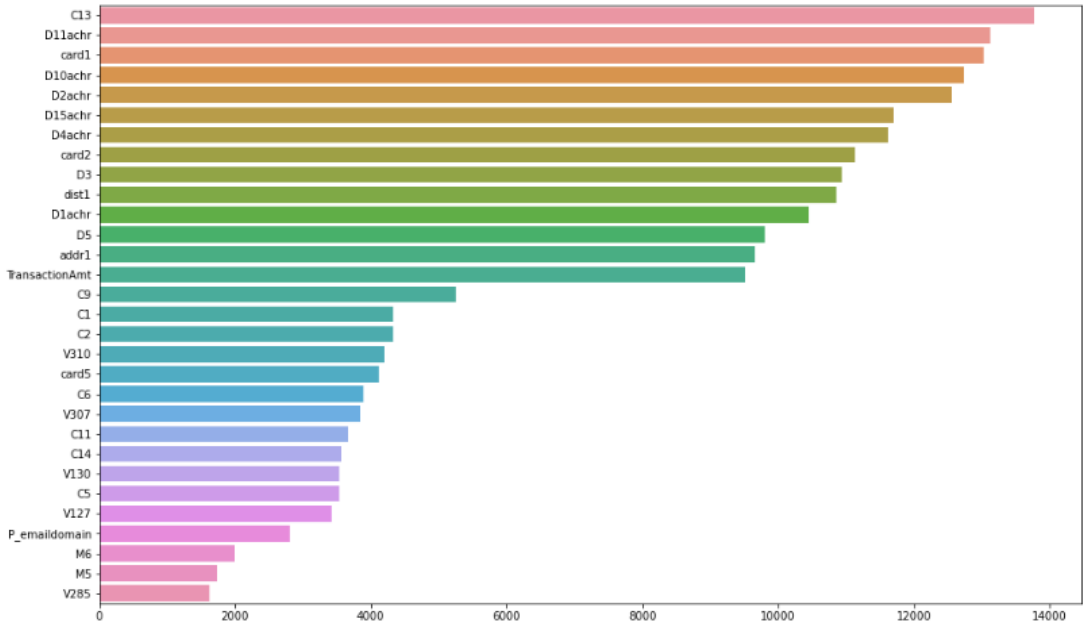
D1 se identifica con los días pasado entre la transacción y la primera transacción o fecha de emisión de la tarjeta. Se transformó restando TransactionDelta (en segundos), convertido a días con la formula:

$$D1achr = (TransactionDelta / (60*60*24)) - D1$$

Esta variable es ya constante para todas las transacciones de una misma tarjeta. Igualmente hicimos esta transformación para otras columnas D con las mismas características.

La Adversarial Validation o Validación Adversaria es una herramienta de detección de fuentes de sobreentrenamiento. Consiste en seleccionar las características que introduciremos en el modelo, unir el conjunto de datos de entrenamiento y de prueba y etiquetar el resultado con un nuevo target (es prueba, no es prueba). Si generamos un modelo y este es capaz de diferenciar ambos conjuntos, tenemos una fuente de sobreentrenamiento ya que hay características diferenciadas entre los dataset.

Nuestra hipótesis, basada en el estudio de otros participantes de la competición, es que las tarjetas del conjunto de entrenamiento no son las mismas que las de prueba. Si excluimos al target (isFraud), y las columnas únicas por transacción (TransactionID, TransationDT), las características que participen más en la detección del nuevo target se consideran campos pertenecientes o relacionados con la tarjeta. En esta validación, elegimos entre dos modelos de boosting, seleccionando LightGBM. Obtuvimos la siguiente importancia de características:



Gráfica 4. Feature Importance del Adversarial Validation.

A partir de estos resultados creamos una tercera definición de tarjeta:

```
train['uid2'] = train['card1'].astype('str')   + '_' + train['D2achr'].astype('str')  + '_' \
              + train['C13'].astype('str')     + '_' + train['D11achr'].astype('str') + '_' \
              + train['D10achr'].astype('str') + '_' + train['D15achr'].astype('str') + '_' \
              + train['D4achr'].astype('str')
```

Imagen 3. Línea de código Python sobre la creación de la tercera identificación de tarjeta.

Creamos tres variables auxiliares uids, uid1 y uid2 que en preprocesamiento e ingeniería de variables utilizamos para completar nulos o crear nuevas variables agregadas por tarjeta. Seleccionamos una u otra en función de los resultados del modelo, hasta obtener el conjunto de datos que nos dio el mejor AUC y el menor sobreentrenamiento.

4. Preparación de los datos

Se partió de un preprocesamiento e ingeniería de variables único para ambos modelos a probar (LightGBM y XGBoosting), viéndose después en iteraciones posteriores que cada uno mejoraba su resultado con el mismo preprocesamiento base, pero ingeniería distinta.

LightGBM y XGBoost

- Transformación de las columnas objeto a numéricas con un diccionario, dejando los nulos tal y como estaban. No utilizamos LabelEncoder ya que nos obliga a imputar los nulos antes, el LabelEncoder puede codificar entonces un número distinto al antiguo nulo en cada columna. Posteriormente, dimos un mismo valor -1 a todos los nulos del dataset.
- Conversión de valores de campos de tarjeta (cardx), direcciones (addrx) al valor más común no nulo de card1. Los nulos restantes a -999.
- Conversión de los valores de las columnas de concordancias (Mx) y dominios correo (emaildomain). al valor más común no nulo de uid1. Los nulos restantes a -999.
- El resto de los nulos se quedan tal cual, tanto LightGBM como XGBoost pueden trabajar con ellos al contrario de otros clasificadores, o se les puede dar un valor numérico antes.

XGBBoost

- En XGBoost se observó que si mejoraba el AUC cambiando el valor del número por nulo dependiendo de la columna. La mejor combinación fue -1 para todas las columnas excepto las Dachrs (contienen valores muy negativos), dejando estos valores a -9999.
- Ingeniería de variables utilizando un frequency encoder: significa que, para cada valor en una columna, se crea otra variable con el número de ocurrencias. Separa los valores más frecuentes de los esporádicos.
- Ingeniería de variables utilizando estadísticas (agregados): Se intenta estadísticas por tarjeta, en ningún caso se ve una gran mejora, y aumenta el sobreentrenamiento.
- El procesamiento fue un trabajo retroalimentado por los resultados de las pruebas de ambos modelos, requiriendo cambios en preprocesamiento, ingeniería de variables o eliminación de características después de obtener la importancia de características del modelo.

5. Modelado

Se partió del **uso de modelos de boosting,** dónde se asume la disponibilidad de un algoritmo de aprendizaje base o débil que, dado ejemplos de entrenamiento ya etiquetados, produce un clasificador base o débil. El objetivo de Boosting es el de mejorar el rendimiento del algoritmo de aprendizaje al tratarlo como una "caja negra". La idea fundamental detrás de Boosting es elegir conjuntos de entrenamiento para el algoritmo de aprendizaje base de tal manera que lo obligue a inferir algo nuevo sobre los datos cada vez que se lo llame. Para ello, hemos utilizados dos algoritmos de boosting: LightGBM y XGBoost.

5.1 Modelado con LightGBM

El modelo de LightGBM es un algoritmo de refuerzo de gradientes (gradient boosting) basado en modelos de árboles de decisión.

Este framework utiliza la técnica Gradient Boosting, con este método los árboles se construyen de manera secuencial y cada uno que se agrega aporta para refinar la predicción anterior. Es decir, se comienza con un valor constante y cada árbol nuevo se entrena para predecir el error en la suma de todas las predicciones de los árboles anteriores. Una vez terminado el proceso, las predicciones se calculan sumando los resultados de todos los árboles que se construyeron. El efecto que tiene esto es que cada vez que se agrega un árbol nuevo se le presta atención a las muestras en las que el modelo está funcionando peor y se trabaja para mejorar ese aspecto.

Las principales ventajas del uso de LightGBM son:

- Un ritmo de preparación más rápido y una mayor productividad: LightGBM usa algoritmos basados en histogramas (es decir, que agrupa los valores de atributos continuos en bins discretos) para agilizar el entrenamiento y reducir el uso de memoria.
- Menor uso de la memoria: está optimizado para que el árbol crezca en la dirección de los mejores nodos (ayudando a una mejor administración de memoria).
- Precisión preferible a algún otro cálculo de impulso: crea árboles siguiendo un enfoque de división de hojas en vez de un enfoque de nivelación que es el factor fundamental para lograr una mayor exactitud.

- Soporte de aprendizaje paralelo y soporte para GPUs: los algoritmos de Machine Learning conllevan una gran carga computacional de ahí que sea tan importante contar con GPU para poder entrenar modelos de manera más fluida.
- Capacidad para manejar datos a gran escala: el funcionamiento interno de LightGBM y la forma en la que construye los árboles están diseñados para soportar enormes volúmenes de datos y grandes cantidades de variables.

Para entrenar el modelo, hemos utilizado un optimizador de hiperparametros: Bayesian Optimization. La optimización bayesiana es una estrategia de diseño secuencial para la optimización global de funciones de caja negra que no asume ninguna forma funcional.

La optimización bayesiana de hiperparámetros consiste en crear un modelo probabilístico en el que el valor de la función objetivo es la métrica de validación del modelo (en este caso, AUC). Con esta estrategia, se consigue que la búsqueda se vaya redirigiendo en cada iteración hacia las regiones de mayor interés. El objetivo final es reducir el número de combinaciones de hiperparámetros con las que se evalúa el modelo, eligiendo únicamente los mejores candidatos. Esto significa que se maximiza cuando el espacio de búsqueda es muy amplio o la evaluación del modelo es muy lenta.

De esta manera, obtenemos diferentes combinaciones en función del rango de valores asignado a cada uno de los hiperparámetros:

#	learning_rate	num_leaves	max_depth	colsample_bytree	subsample	AUC
1	0.03215518473989215	294	14	0.4	0.8	0.9194457912401148
2	0.04	278	13	0.4	0.9	0.9164233793906151
3	0.027832622205538907	268	15	0.3740990054742914	0.589171882228726	0.9188929389573968
4	0.029827755272285032	262	15	0.7823490596970961	0.8631490082348747	0.92719808846762

Imagen 4. Las cuatro mejores combinaciones según la optimización bayesiana de hiperparametros.

```
##### LightGBM with #3

parameters = {'objective': 'binary',
              'boosting_type': 'gbdt',
              'metric': 'auc',
              'n_jobs': -1,
              'tree_learner': 'serial',
              'learning_rate': 0.027832622205538907,
              'num_leaves': 268,
              'max_depth': 15,
              'colsample_bytree': 0.3740990054742914,
              'subsample': 0.589171882228726,
              'verbosity': -1,
              'random_state': 27}

categorical_columns = ['ProductCD', 'card4', 'card6', 'P_emaildomain', 'M1', 'M3', 'M4', 'M
5', 'M6', 'M8', 'M9']

train_data = lgb.Dataset(X_train, label = y_train, categorical_feature = categorical_column
s)
val_data   = lgb.Dataset(X_val,   label = y_val, categorical_feature = categorical_columns)

clf = lgb.train(parameters, train_data , valid_sets = [train_data, val_data], num_boost_rou
nd = 200, categorical_feature = categorical_columns,
                          early_stopping_rounds = 200, verbose_eval = 100)
```

Imagen 5. Líneas de código de Python sobre la creación y entrenamiento del modelo LightGBM.

5.2 Modelado con XGBoost

GBoost Extreme Gradient Boosting, es una implementación de gradient boosting a partir de árboles de decisión consecutivos permite resolver casos de clasificación y regresión. La idea es tener un conjunto de arboles de decisión más simples (estimador débiles) consecutivos, y que cada uno corrija los errores del anterior.

Los modelos se ajustan usando alguna función de pérdida diferenciable y el algoritmo de optimización de gradiente descendiente. Se llama Gradient Boosting porque el gradiente de pérdida se va minimizando según el modelo se corrige, similar a lo que ocurre en una red neuronal. Como en cualquier otra implementación de boosting, el principio es unir estimadores débiles formando un estimador fuerte.

XGBoost se diseño para ser computacionalmente eficiente, ejecuta muy rápidamente, y ofrece un modelo de alto rendimiento que hace que sea uno de los preferidos en las competiciones de Kaggle. Entre otras características:

- Velocidad y rendimiento: Escrito originalmente en C++, comparativamente más rápido que otros clasificadores ensamblados.
- El algoritmo del core es paralelizable. Puede aprovechar la potencia de computadoras de varios núcleos, en la nube y también paralelizable en GPU. Puede manejar grandes cantidades de datos.
- Generalmente supera otros algoritmos, según los benchmarks realizados.
- Gran cantidad de hiperparámetros (validación cruzada, regularización, funciones objetivo definidas por el usuario, valores nulos, parámetros específicos de los árboles, compatible con el API de scikit-learn)
- Uso de los algoritmos weighted quantile sketch para encontrar el punto óptimo de partición de un nodo.

```
##### XGBoosting

clf = xgb.XGBClassifier(
                objective = 'binary:logistic',
                base_score = 0.5,
                booster = 'gbtree',
                colsample_bylevel = 1,
                colsample_bynode = 1,
                colsample_bytree = 0.4,     #percentage of columns by tree, overfitting parameter
                gamma = 0,
                gpu_id = -1,
                importance_type = 'gain',
                interaction_constraints = '',
                learning_rate = 0.02,
                max_delta_step = 0,
                max_depth = 12,             #max tree depth, overfitting parameter
                min_child_weight = 1,
                missing = -1,               #missing value is the value to replace NaN. Default, np.nan
                monotone_constraints =(),
                n_estimators = 2000,        #number of trees.
                n_jobs = 0,
                num_parallel_tree = 1,
                random_state = 0,
                reg_alpha = 0,
                reg_lambda = 1,
                scale_pos_weight = 1,
                subsample = 0.8,            #percentage of rows by tree, overfitting parameter
                tree_method = 'gpu_hist',   #use gpu to speed the training/fit of the model.
                validate_parameters = 1,
                verbosity = None,           #verbosity (0-3) 0 is silent, and 3 debug. It looks this does not impact in verbose.
                eval_metric = 'auc')

clf.fit(df_train[feature_list], df_train[target],
eval_set=[(df_train[feature_list],   df_train[target]), (df_val[feature_list],   df_val[target])],
        verbose=100, early_stopping_rounds=100)
```

Imagen 6. Líneas de código de Python sobre la creación y entrenamiento del modelo XGBoost.

6. Evaluación

6.1 Resultados

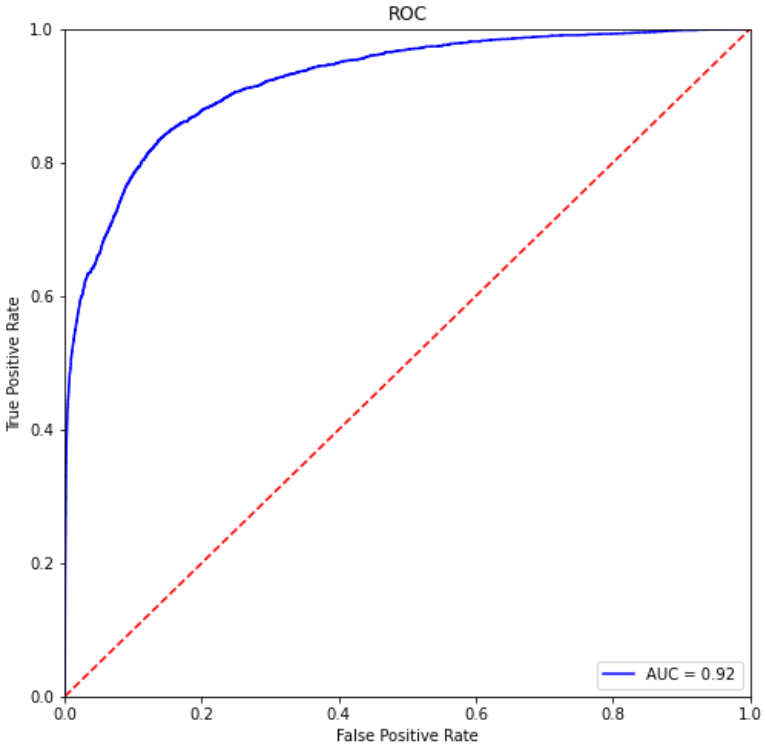
LightGBM

Nuestra elección final fue la combinación 3, de la que obtenemos la siguiente curva ROC y la importancia de las características para el modelo:

```
##### AUC

auc = roc_auc_score(y_val, clf.predict(X_val))
print('AUC:',auc)
```

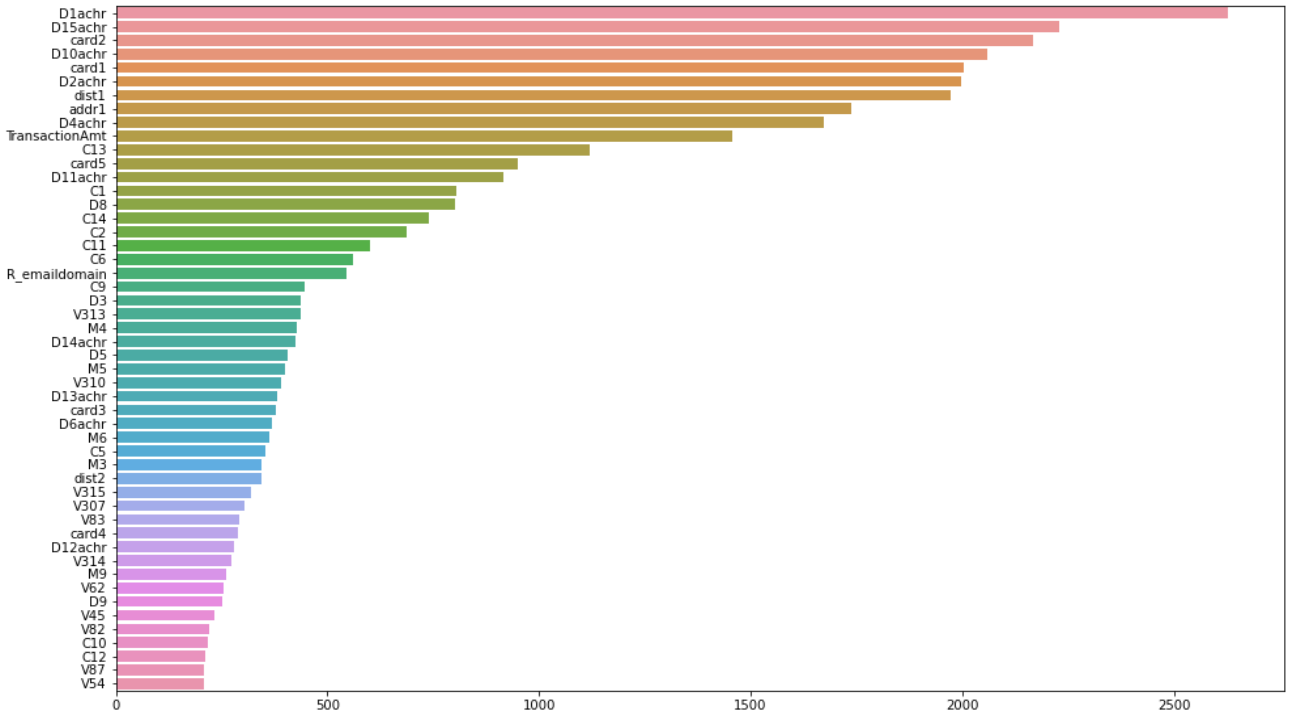
```
AUC: 0.9235192329572796
```

Imagen 7. Líneas de código de Python sobre el cálculo de la métrica AUC para LightGBM.



Gráfica 5. Curva ROC para LightGBM.

Gráfica 6. Feature Importance para LightGBM.

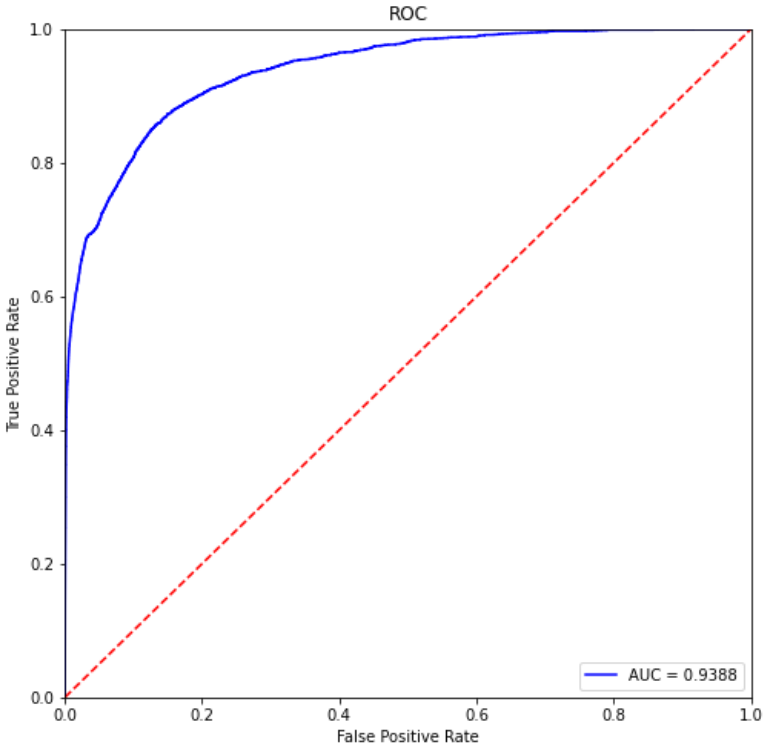


XGBoost

Nuestra elección final fue la elección de hiperparametros y sus respectivos valores a través prueba y error. De esa manera, obtenemos la siguiente curva ROC y la importancia de las características para el modelo:

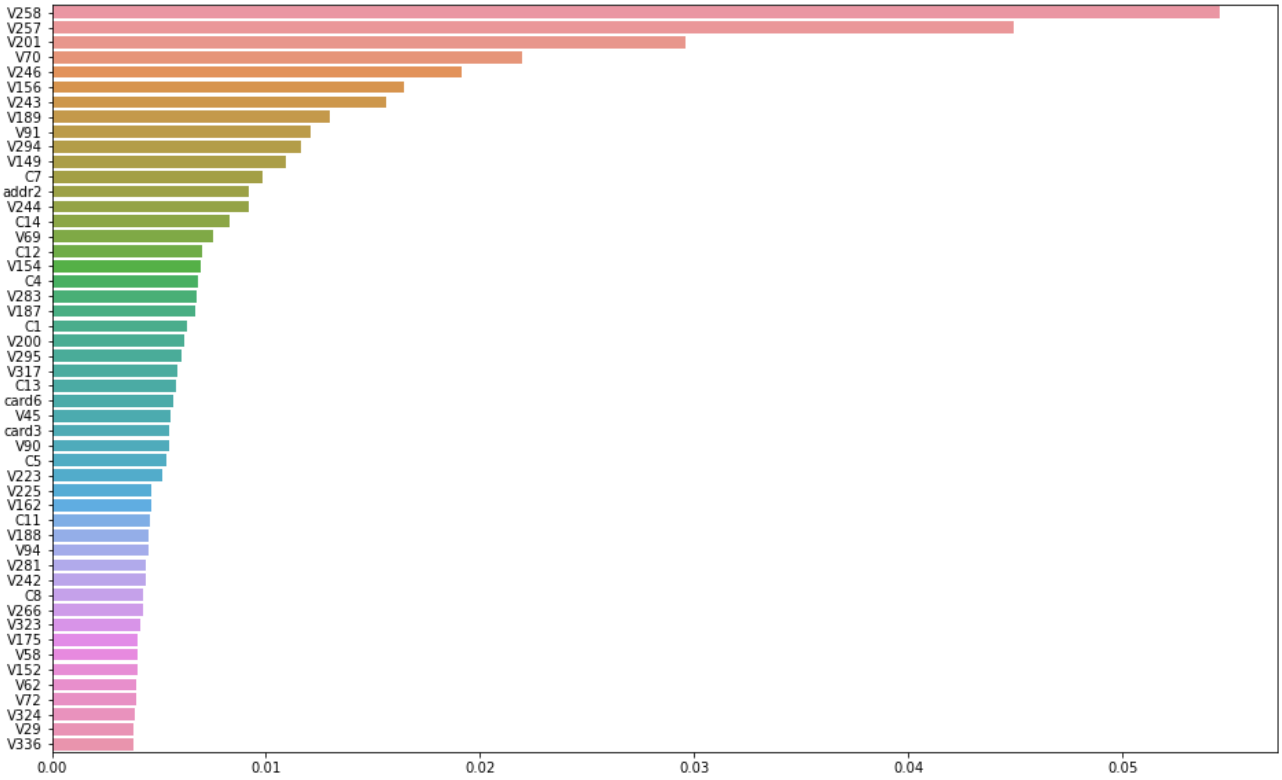
```
##### AUC

auc = clf.best_score
print('AUC:',auc)

AUC: 0.938792
```

Imagen 8. Líneas de código de Python sobre el cálculo de la métrica AUC para XGBoost.



Gráfica 7. Curva ROC para XGBoost.

Gráfica 8. Feature Importance para XGBoost.



Uplift modelling

Para poder obtener el AUC para el test, realizamos entregas a la competición a través de Kaggle y nos devolvía dos resultados: Private Score (calculado con el 80% de los datos) y Public Score (calculado con el 20% de los datos). De esa manera calculábamos el overfitting a través de la diferencia entre el AUC frente a validación y el AUC frente a test. Este overfitting podría ser provocado porque la parte de validación se había considerado como parte del train y podría estar algo contaminada con sus datos.

La elección de los modelos se realizó eligiendo los 3 mejores resultados del Private Score para ambos modelos y sobre ese ranking, elegíamos aquel que poseía menos overfitting. De esa manera obteníamos el máximo score a la vez que minimizamos el overfitting. Estos fueron los modelos seleccionados:

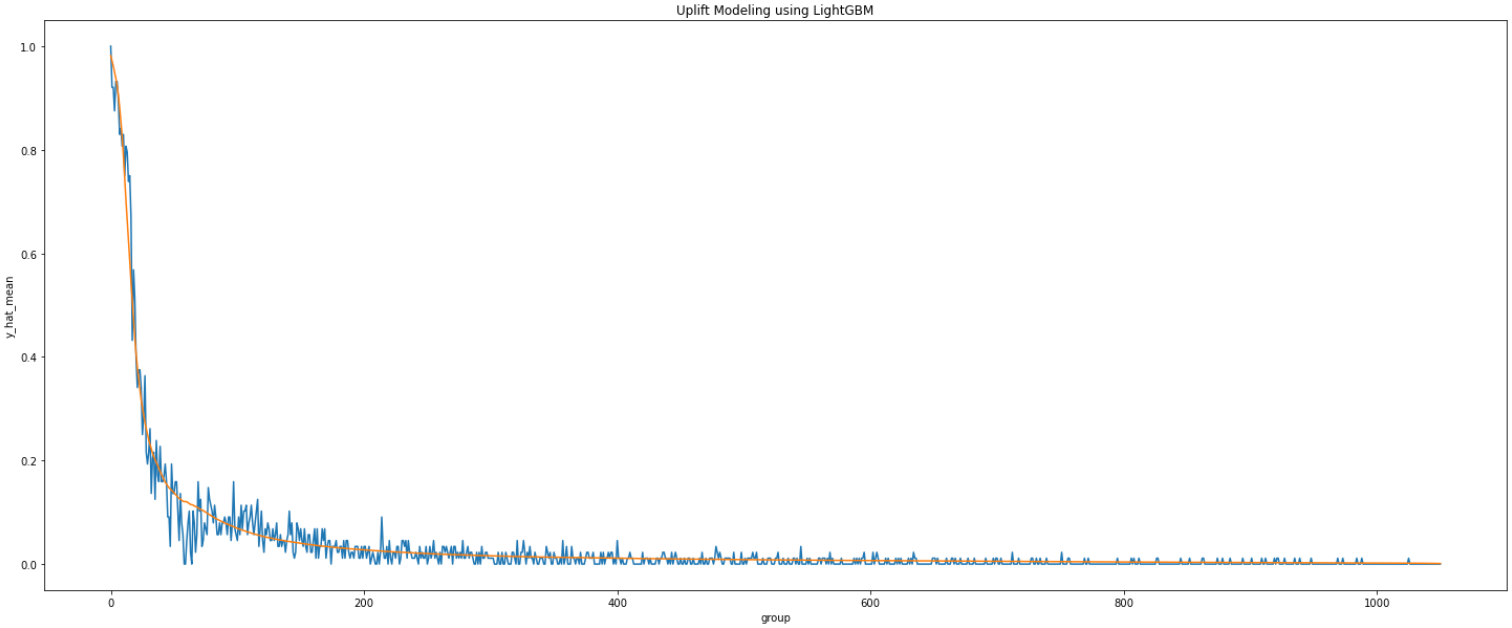
Model	Local Score	Private Score	Public Score	Overfitting with Local
LightGBM	0.923519	0.905291	0.927122	1.97%
XGBoost	0.938792	0.905261	0.929851	3.42%

Imagen 9. Tabla con los modelos elegidos.

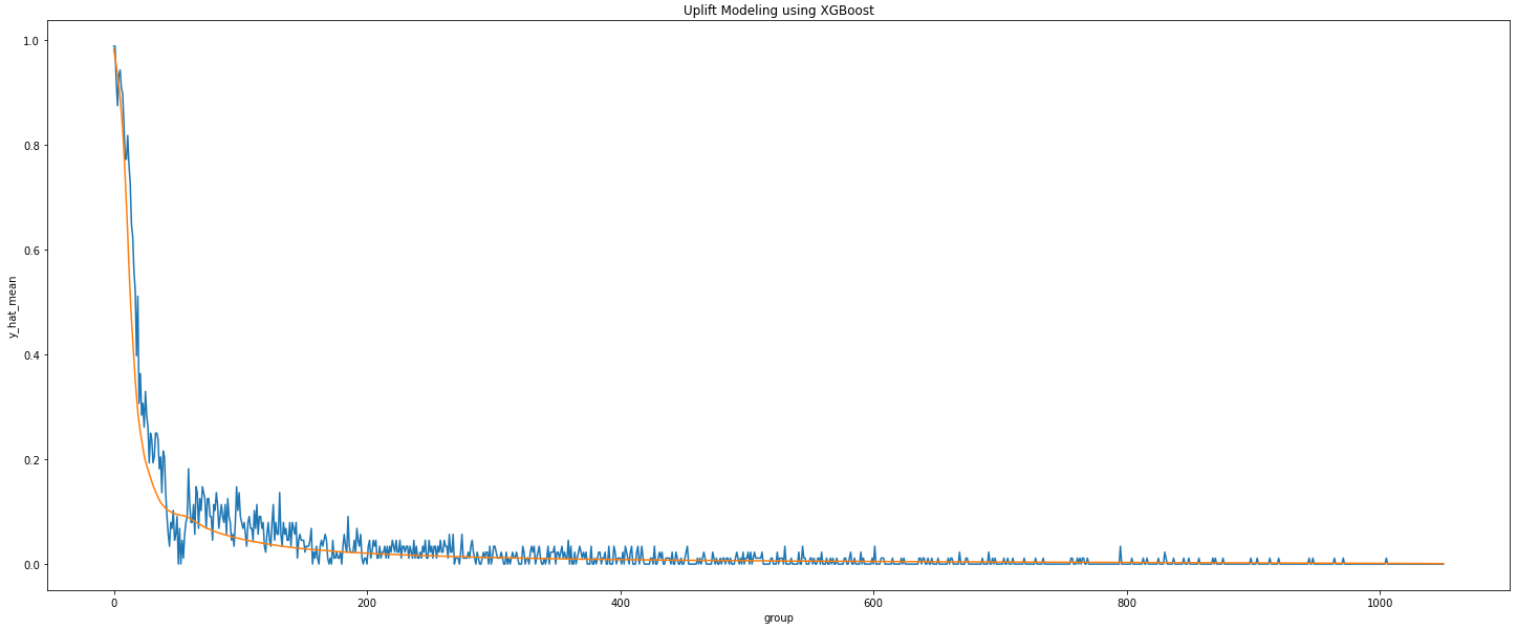
A través de estos modelos, desarrollamos la idea del Uplift modelling. En primer lugar, elegimos el data de validación y sobre él extraemos el ID de cada transacción, el valor de fraude real y el valor de fraude predicho según nuestro modelo. Para poder agrupar a las transacciones por fraude ordenamos, en orden ascendente, las transacciones según la probabilidad que determinaba el modelo de que fueran transacción fraudulenta.

Después agrupábamos por percentiles de 0.05% dando lugar a 1050 grupos con 88 transacciones en cada uno. De esta manera, obteníamos grupos de transacciones con fraude muy parecido y así podríamos ver sobre que grupos focalizar la atención.

Finalmente, sobre cada grupo observábamos el número de transacciones fraudulentas y no fraudulentas y extraemos el ratio de fraude = $\frac{\text{transacciones fraudulentas}}{\text{total de transacciones del grupo}}$ y lo comparamos con la probabilidad fraude media según nuestro modelo. Lo graficamos y vemos los siguientes resultados: la línea azul representa el ratio de fraude real por grupo y la línea naranja representa la probabilidad de fraude media según el modelo por grupo.



Gráfica 9. Uplift modelling usando LightGBM



Gráfica 10. Uplift modelling usando XGBoost

6.2 Conclusiones y Trabajo a Futuro

A través de estos modelos de Machine Learning desarrollamos la idea de poder clasificar transacciones fraudulentas de tarjetas de crédito. En base a los resultados de los modelos, obtenemos un área bajo la curva ROC de 0.92351 para LightGBM y 0.93879 para XGBoost lo que indica que obtenemos mas de un 92% de media de predicciones correctas para un par de transacciones, una fraudulenta y otra no, seleccionadas al azar.

Además, los modelos poseen un overfitting menor al 5%, lo cual indica que el modelo carece de ese sobreentrenamiento que perjudicaría ante la predicción de datos completamente nuevos para el algoritmo.

Sin embargo, el tiempo empleado para el desarrollo del proyecto ha sido de 15 días y no se ha podido explorar todas las líneas de posibilidades de mejora del algoritmo. Es por ello, que detallamos algunos de los cambios que meteríamos en un trabajo a futuro:

- Evaluar nuevas variables generadas por Feature Engineering: maximizando el área bajo la curva ROC y minimizando el overfitting de los algoritmos para poder reducir el número de falsos positivos. De esta manera, se evitaría molestar a clientes que no están sufriendo fraude y que acaben por darse de baja del servicio.

- Comparar resultados con otros algoritmos: utilizar Deep Learning a través de redes neuronales para ver si es posible mejorar los resultados y maximizar las métricas utilizadas.

- Generar datos sintéticos para fraude: el desbalance de clases debido al bajo número de transacciones fraudulentas complica el aprendizaje del algoritmo. A través de técnicas como SMOTE, intentar mejorar este desbalance.

- Aplicar optimizadores de hiperparámetros en XGBoost: gracias a la optimización de hiperparámetros se encuentra una tupla de hiperparámetros que produce un modelo óptimo que minimiza la función de pérdida. Aplicar nuevas técnicas de optimización de hiperparametros podrían mejorar estos resultados.

Bibliografía

- A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning (2020). Retrieved 30 December 2020, from https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f
- A Quick Uplift Modeling Introduction. (2020). Retrieved 30 December 2020, from https://towardsdatascience.com/a-quick-uplift-modeling-introduction-6e14de32bfe0
- Basulto Santos, J., & García del Hoyo, J. (2009). *Historia de la probabilidad y la estadística (IV)* (p. Capitulo 31). [Huelva]: Servicio de Publicaciones, Universidad de Huelva.
- Credit Card Fraud Statistics. (2020). Retrieved 30 December 2020, from https://shiftprocessing.com/credit-card-fraud-statistics/
- Cruz-Reyes, H., Reyes-Nava, A., Rendón-Lara, E., & Alejo, R. (2018). *Estudio del desbalance de clases en bases de datos de microarrays de expresión genética mediante técnicas de Deep Learning.* Universidad Autónoma del Estado de México.
- Curva ROC. (2020). Retrieved 27 December 2020, from https://es.wikipedia.org/wiki/Curva_ROC
- Espinar Lara, R. (2018). *Modelos de Clasificación con datos no balanceados.* Universidad de Sevilla.
- Exploring XG-Boost: Extreme Gradient Boosting. (2019). Retrieved 30 December 2020, from https://www.fromthegenesis.com/exploring-xg-boost-extreme-gradient-boosting/
- Faraco, F. (2020). El impacto de la pandemia en el fraude de las entidades financieras. Retrieved 27 December 2020, from https://cincodias.elpais.com/cincodias/2020/04/30/opinion/1588245588_987892.html
- Fraud and Security Practice Area. (2020). Retrieved 29 December 2020, from https://www.javelinstrategy.com/coverage-area/fraud-security
- Mangla, I. (2018). Shimming Is the Latest Credit Card Scam. Retrieved 30 December 2020, from https://www.experian.com/blogs/ask-experian/shimming-is-the-latest-credit-card-scam/
- Mecia, T. (2017). Credit card fraud alerts surge, false alarms still common. Retrieved 30 December 2020, from https://www.creditcards.com/credit-card-news/credit-card-fraud-alert-poll/

- Phua, C., Lee, V., Smith, K., & Gayler, R. *A Comprehensive Survey of Data Mining-based Fraud Detection Research*. School of Business Systems, Faculty of Information Technology, Monash University.
- Poremba, S. (2019). How to Protect Yourself Against Card Skimmers at Gas Stations. Retrieved 30 December 2020, from https://www.experian.com/blogs/ask-experian/how-to-protect-yourself-against-credit-card-skimmers-at-gas-stations/
- Psd2 Seguridad Pagos Todo lo que hay Saber Directiva Europea Psd2. (2020). Retrieved 30 December 2020, from https://bancaelectronica.net/que-es-psd2/
- Temple, S. (2020). A Quick Uplift Modeling Introduction. Retrieved 29 December 2020, from https://towardsdatascience.com/a-quick-uplift-modeling-introduction-6e14de32bfe0
- Uplift Modeling. Introduction. (2020). Retrieved 30 December 2020, from https://towardsdatascience.com/uplift-modeling-e38f96b1ef60
- Uplift modelling. (2020). Retrieved 28 December 2020, from https://en.wikipedia.org/wiki/Uplift_modelling