# Caso de estudio: Modelo de detección de fraude

En este artículo contamos el caso de uso de un modelo de detección de fraude de identidad o de solicitud. Solución planteada, desarrollo del modelo, resultados del mismo y posibles mejoras.

## El fraude de identidad o de solicitud

Según la AEECF (Asociación Española de Empresas Contra el Fraude), el fraude online está creciendo de forma exponencial debido a la proliferación de dispositivos electrónicos. Los ataques de phishing (técnica por la que se pretende engañar al usuario para que facilite información confidencial: datos personales, número de tarjetas de crédito, claves o contraseñas) han experimentado un crecimiento del 87% en los últimos años.

Entre los fraudes on-line más habituales, la AEECF considera que el fraude de identidad, aquel en el que el defraudador se hace pasar por un tercero para obtener un beneficio personal, es el ataque más habitual en sus organizaciones. En segundo lugar se sitúa el fraude de solicitud, que puede ser fraude de primera persona o fraude de tercera persona. El fraude de primera persona lo comete el infractor mediante una alteración de sus datos para engañar a la entidad o perjudicar la decisión de crédito, mientras que en el fraude de tercera persona asume la suplantación de identidad o el uso de identidades ficticias.

Generalmente, los fraudes de identidad y de solicitud quedan ocultos en las pérdidas de malos pagadores, es decir, los clientes que se retrasan en el pago. Por esa causa, los analistas no lo identifican como fraude (salvo en contados casos) y la empresa nunca se recupera de las pérdidas.

Este contexto junto con la disponibilidad creciente de datos sobre clientes, su interacción con las compañías, sus hábitos de comportamiento, etc., sí como el avance de la tecnología, ha hecho que la aplicación de técnicas de inteligencia artificial sea clave para la detección y lucha contra el fraude.

## Solución planteada para luchar contra el fraude

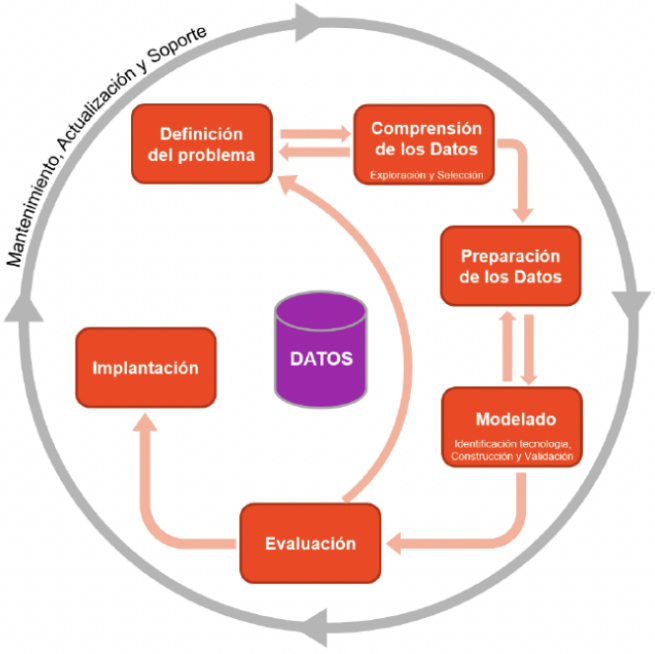
Hasta ahora, las compañías disponían de sistemas de gestión de fraude convencionales, basados en motores de reglas. Dichos sistemas entienden los fraudes conocidos y tienen reglas construidas para detectarlos, que pueden estar basadas en impagos, deuda existente en listas de morosos, listas de antiguos malos clientes, etc. Pero estos sistemas no son ágiles y siempre van

un paso por detrás de las constantes innovaciones en técnicas de fraude. Aun así, dichos sistemas aportan una información valiosa que se puede aprovechar con las nuevas tecnologías basadas en inteligencia artificial. De esta manera no se pierde el conocimiento adquirido por las compañías durante años.

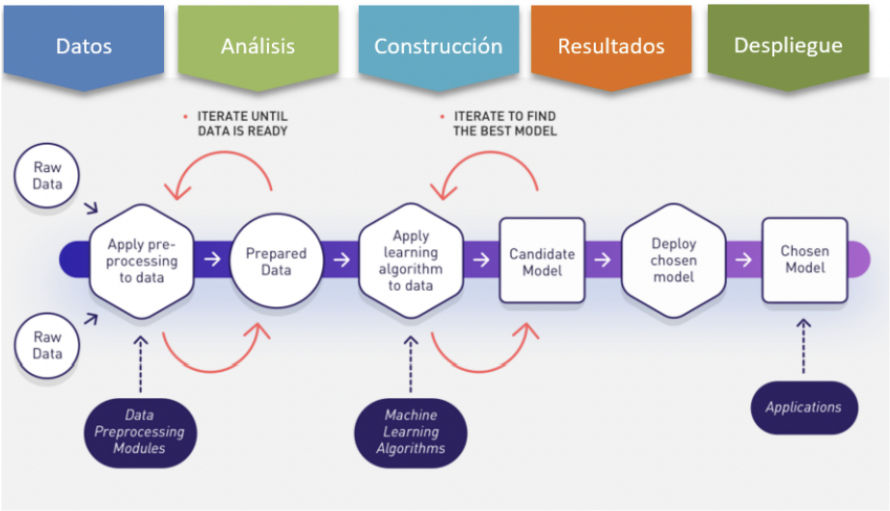
Por ello, una posible solución para la detección del fraude de identidad o de solicitud es aunar ambas técnicas: los motores de reglas convencionales y el aprendizaje automático, de manera que las decisiones provenientes de los motores de reglas se utilicen como variables de entrada para los algoritmos de aprendizaje automático.

Además, los resultados provenientes de los motores de reglas serán utilizados como referencia para evaluar posteriormente los resultados del algoritmo de aprendizaje automático y así, ver lo que aporta dicho algoritmo a la detección del fraude respecto a los motores de reglas convencionales.

Teniendo todo esto en cuenta y siguiendo la metodología habitual para proyectos de machine learning conocida como CRISP-DM, los pasos a seguir serán: comprensión del negocio/problema, comprensión de los datos, preparación de los datos, modelado, evaluación e implantación, tal y como se pueden ver en los siguientes diagramas:




Fuente: (Decide, 2019)




Fuente: (Decide, 2019)

## 1. Datos y Análisis Descriptivo

Lo primero que se debe hacer es un análisis descriptivo pormenorizado de los datos para comprenderlos, seleccionarlos, limpiarlos y transformarlos antes de alimentar ningún algoritmo con ellos.

Se parte de un problema de aprendizaje automático supervisado. En concreto es un problema de clasificación, es decir, se debe construir un modelo de detección de fraude que sea capaz de predecir una categoría (por ejemplo, verdadero o falso, o en este caso “fraude”/”no fraude”).

Por la naturaleza del problema, los datos que se tienen para abordarlo son datos con desbalanceo de clases, es decir, no se tiene la misma proporción de registros u observaciones fraudulentos como no fraudulentos. En este tipo de problemas, no es extraño que los datos tengan una proporción de 99% de casos de no fraude frente al 1% de casos de fraude. Lo que hará que sea más difícil para el algoritmo encontrar patrones de estos últimos. Por ello, se deben aplicar ciertas técnicas de balanceo de clases que se detallarán más adelante.

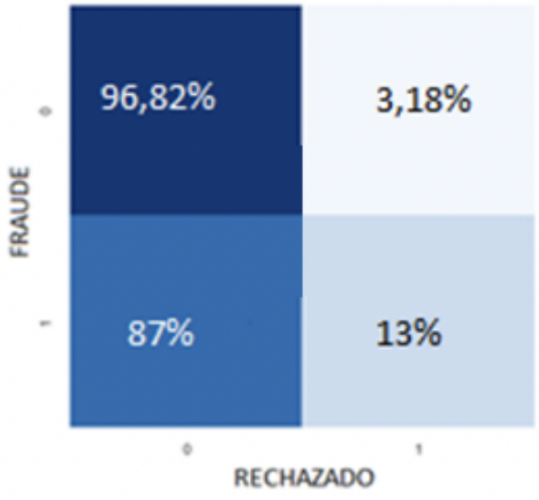
También se debe pensar que para que el algoritmo entregue valor real, éste debe detectar el mayor número de fraudes posible (verdaderos positivos), pero a la vez, no debe equivocarse demasiado (falsos positivos). Porque para las compañías, los falsos positivos suponen potenciales buenos clientes que el algoritmo va a descartar y, por consiguiente, son potenciales beneficios que van a dejar de ingresar. Lo que se sumará a las pérdidas provenientes de los individuos fraudulentos que el algoritmo no detecte como tal (falsos negativos). Por tanto, el algoritmo debe ser lo mejor posible tanto detectando los fraudes como los no fraudes.

Los datos de los que se dispone se componen de una serie de variables que podemos llamar “originales” o “naturales”, que son en su mayoría los datos que el solicitante rellena al solicitar el alta en el servicio y/o producto. Dichos datos son, como se ha comentado anteriormente, susceptibles de ser falsos o pertenecer a un tercero y no a la persona que rellena la solicitud. Dichos casos junto con los potenciales fraudes detectados por los motores de reglas, son los que debe detectar el algoritmo. Por ello, dichos datos deberían tener un tratamiento especial que se detallará más adelante.

Además de estas variables “originales”, los datos también se componen de un conjunto de “variables sintéticas”. Algunas creadas a partir de las variables originales, y otras que son las generadas por los motores de reglas convencionales. Estas últimas por tanto, son las que le darán un extra de conocimiento al algoritmo.

Las variables generadas por los motores de reglas, además de poder utilizarlas como variables de entrada para el algoritmo de aprendizaje automático, también se pueden utilizar para cuantificar los resultados de los motores de reglas y poder tener una “tasa de rechazo vs fraude”. De esta manera se tendrá una referencia para evaluar lo que aportará el algoritmo de aprendizaje automático a la solución del problema.

Se pueden ver dichos resultados de los motores de reglas en el siguiente esquema:




Fuente: (Decide, 2019)

La explicación del esquema anterior, sería la siguiente:

- 96.82% de registros no rechazados (0) y que no son fraude (0).
- 3.18% de registros rechazados (1) y que no son fraude (0). Hay que tener en cuenta también que los motores de reglas pueden rechazarlos por otras causas.
- 87% de registros no rechazados (0) y que sí son fraude (1). Registros fraudulentos que no son rechazados por los motores de reglas convencionales.
- 13% de registros rechazados (1) y que sí son fraude (1). Registros fraudulentos que sí son rechazados por los motores de reglas convencionales.

El objetivo de aplicar algoritmos de aprendizaje automático a este caso, es ese 87% de casos de fraude que los motores de reglas no son capaces de detectar.

## 2. Construcción del Modelo

### 2.1 Enriquecimiento de datos y creación de variables sintéticas

El enriquecimiento de los datos mediante fuentes de datos externas y la creación de otras variables (variables sintéticas) a partir de variables dadas, son técnicas habituales en la construcción de modelos de aprendizaje automático.

Es habitual en problemas de este tipo que, entre los datos de que se disponen, existan muchas variables que no aportan información en bruto a un modelo. Pero pueden utilizarse para obtener datos de más valor de fuentes externas o crear otras variables a partir de ellas que sí aporten información válida.

Para este caso de uso, es evidente que una información muy importante sería poder contrastar si los datos que introduce el solicitante al rellenar la solicitud pueden ser falsos o no. Dichos datos son datos personales del tipo: nombre, DNI, dirección, email, teléfono, etc.

Para ello, se cruzan dichos datos con fuentes de datos externas como:

- Base de datos de catastro.
- Validación de datos del individuo (nombre y DNI) mediante aparición en páginas webs públicas. (NOTA: esta validación no está implementada en el modelo de detección de fraude que se explica en este artículo, pero sí se ha evaluado su potencial y se considera muy interesante de cara a proponerlo como mejora al modelo aquí descrito.)

Así mismo, se utilizan diferentes técnicas para validar y cuantificar la veracidad y/o fiabilidad de variables como la dirección de email o el número de teléfono y se crean variables sintéticas con los resultados de dichas validaciones.

Además se utilizan técnicas de aprendizaje no supervisado (clusterización) para crear otra variable sintética a partir de una segmentación de los registros en base a diferentes variables. De esta manera, se crean grupos de solicitudes con características similares que ayudarán al algoritmo de aprendizaje supervisado a detectar las solicitudes fraudulentas de las no fraudulentas.

### 2.2 Desbalanceo de clases

Como se comentó anteriormente, el desbalanceo de clases es una característica especial de este tipo de problemas por su naturaleza, ya que por suerte, el porcentaje de defraudadores siempre será una minoría. En este caso, el 1% de los registros. Por lo tanto, antes de empezar a entrenar modelos/algoritmos, se debe realizar un tratamiento previo, porque es imposible que ningún algoritmo pueda detectar patrones en un 1% del total de registros.

Si no se hiciese nada, el algoritmo sería muy bueno en detectar los “no fraudes” (99% del total) puesto que tiene muchos registros para encontrar patrones. Mientras que no detectaría prácticamente ningún “fraude” (1% del total), por lo que, finalmente, nuestro modelo de “detección de fraude” sería pésimo.

Para evitarlo, es necesario hacer un tratamiento previo, ya sea a nivel de datos o a nivel de algoritmo o ambos. El primer caso se basa en balancear las clases, disminuyendo la clase mayoritaria o aumentando la minoritaria. A nivel de algoritmo las soluciones no suelen ser tan genéricas y dependen del algoritmo que se utilice.

Las técnicas más típicas son:

- Modificando los datos (métodos de remuestreo):
- Under-sampling: consiste básicamente en eliminar datos aleatoriamente de la clase mayoritaria, con la consiguiente pérdida de información.
- Over-sampling: consiste en aumentar el número de casos de la clase minoritaria. Existen diferentes métodos.

Para que el algoritmo sea ágil adaptándose a nuevos comportamientos, es recomendable realizar de manera inteligente este over-sampling. En este caso, el replicar en mayor medida las muestras más recientes de fraude provoca que

el algoritmo sea capaz de tenerlas en cuenta en un corto plazo desde que se comienzan a realizar.

Modificando el algoritmo:

- Si el método lo permite, es posible usar en cada iteración una muestra de los datos originales pero igualando la proporción de las clases. Es común en arboles de decisión.
- Penalización del error: se trata de dar un mayor peso a los errores cometidos al clasificar mal un caso positivo. Al ser una solución genérica puede aplicarse a cualquier algoritmo.
- Método de contrapesos: consiste en parametrizar las clases para ajustarlas al desbalanceo.
- Si el output del algoritmo es probabilístico, es práctica común ajustar el umbral de decisión entre la clase positiva y la negativa.

### 2.3 Entrenamiento y evaluación de diferentes modelos

Una vez que se tienen todos los datos preparados, ya se puede empezar a entrenar modelos.

Se realizan diferentes entrenamientos con distintos algoritmos y parámetros de los mismos. Evaluando los resultados, se obtiene la siguiente información:

Saber qué algoritmos y con qué parámetros se obtienen mejores resultados para trabajar más en ellos.

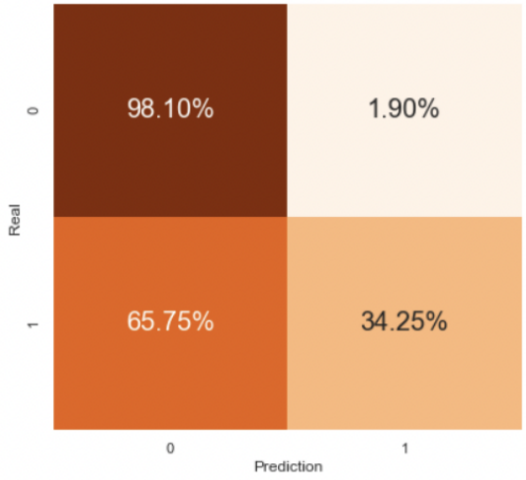
Constatar que el uso de algunas de las variables sintéticas resultantes de los motores de reglas, potenciaban bastante los resultados de los algoritmos.

## Los resultados del modelo de detección de fraude

Finalmente, el modelo para la detección de fraude ganador fue un Random Forest. Random Forest es un algoritmo de aprendizaje supervisado que, como se puede ver en su nombre, crea un “bosque” (conjunto de árboles de decisión) y lo hace de alguna manera aleatorio. Sin entrar en detalles técnicos, Random Forest crea múltiples árboles de decisión y los combina para obtener una predicción más precisa y estable.

Los resultados del modelo podemos verlos en una matriz de confusión que es una herramienta que permite la visualización del desempeño de un algoritmo de aprendizaje supervisado. Es una manera de “medir” lo que acertamos y lo que nos equivocamos de cada una de las clases a predecir.

Por tanto, los resultados fueron los siguientes:

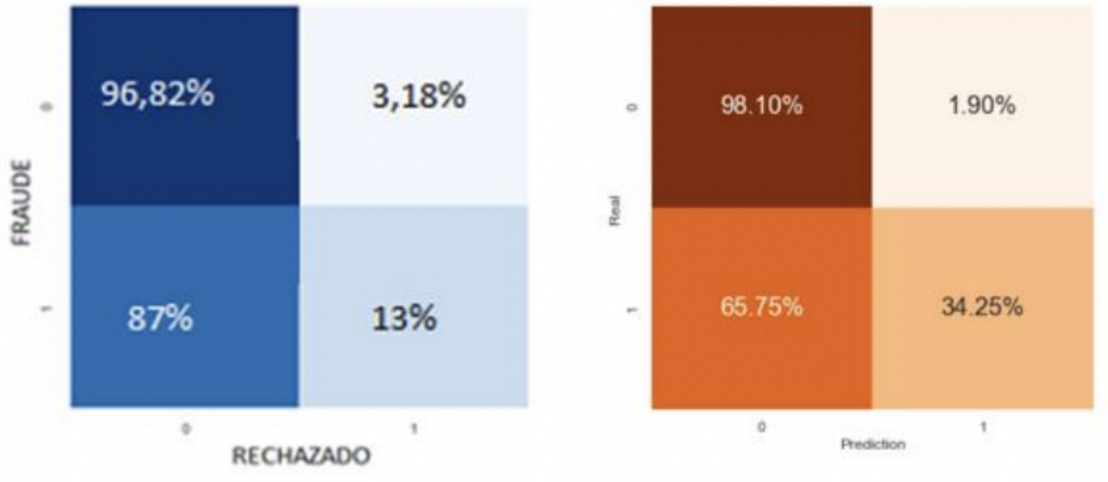



Fuente: (Decide, 2019)

La manera de interpretar los resultados de dicha matriz es la siguiente:

- 98.10% de “no fraudes” predichos correctamente (verdaderos negativos).
- 1.90% de “no fraudes” predichos incorrectamente (falsos positivos).
- 65.75% de “fraudes” predichos incorrectamente (falsos negativos).
- 34.25% de “fraudes” predichos correctamente (verdaderos positivos).

Si comparamos estos resultados con los resultados de los motores de reglas convencionales:




Fuente: (Decide, 2019)

- Se puede ver que el modelo reduce en un 1.28% los falsos positivos, algo que puede ser crítico para el negocio, ya que éstos son los potenciales buenos clientes que se descartan/rechazan, provocando pérdidas.
- Por otro lado, el modelo aumenta en un 21.25% los casos de fraude correctamente detectados (verdaderos positivos) con respecto a los casos que se rechazan con los motores de reglas.

Como se puede ver, los resultados de aunar los motores de reglas convencionales y el aprendizaje automático, pueden dar muy buenos resultados para la detección del fraude de solicitud.

## Posibles mejoras del algoritmo de detección de fraude

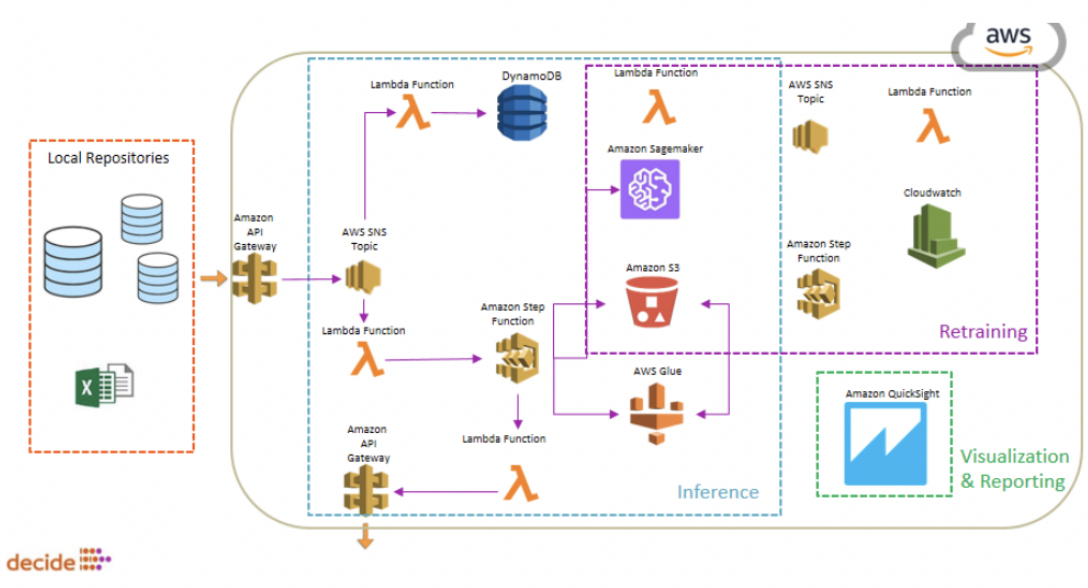
Para una segunda revisión del algoritmo, se proponen dos posibles mejoras ya comentadas en este artículo:

- Validación del individuo (nombre y DNI) si aparece o no en diferentes páginas webs públicas, ya que eso lo hace susceptible de que los estafadores usurpen su identidad para realizar la solicitud de servicio. Esta validación daría algunas variables sintéticas adicionales con las que alimentar el modelo de detección de fraude.
- Posible mejora de los motores de reglas, para ver cuáles pudiesen estar obsoletas y cuales se pueden mejorar para así poder alimentar mejor al modelo.

## Despliegue y explotación del modelo de detección de fraude

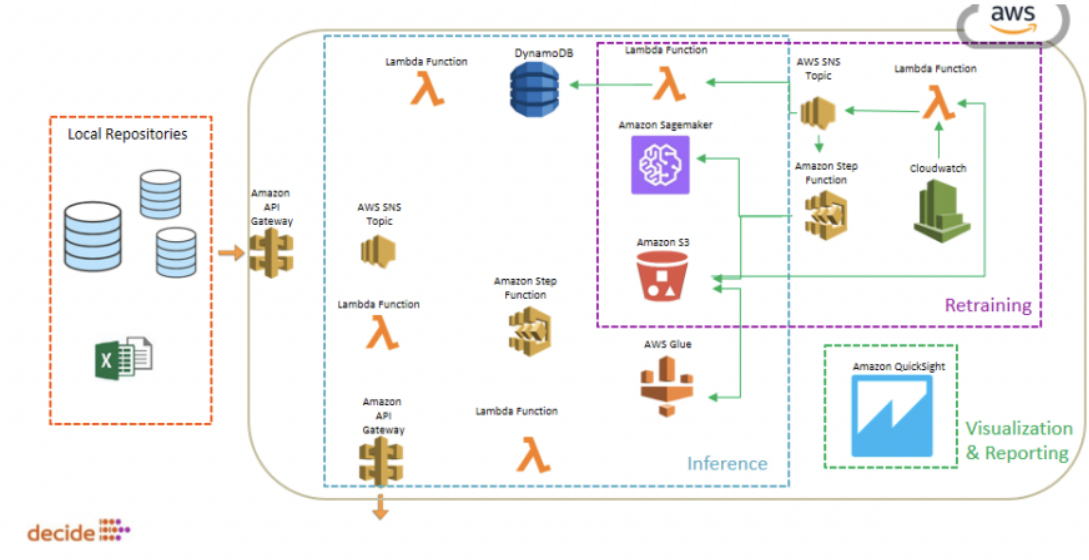
Por último, hay que tener en cuenta que no es suficiente con desarrollar el algoritmo de predicción de fraude, sino que hay que automatizar su uso para obtener las predicciones de los nuevos registros que tengamos (inferencia) e integrarlo en una arquitectura que, en este caso será AWS, que permitirá monitorizar periódicamente nuestro modelo y reentrenarlo con los nuevos datos en caso de que pierda precisión debido a la aparición de nuevos comportamientos fraudulentos.

Esquemáticamente, el flujo de trabajo y los diferentes componentes que entrarían en juego durante el proceso de inferencia sería el siguiente (área con línea discontinua de color azul):




Fuente: (Decide, 2019)

Así mismo, el esquema del flujo de trabajo y los componentes que entran en juego durante el proceso de reentrenamiento, es el siguiente (área con línea discontinua de color fucsia):




Fuente: (Decide, 2019)

## Referencias


- (Decide, 2019) Caso de uso – Modelo de detección de fraude. Recuperado desde: https://decidesoluciones.es/modelo-de-deteccion-de-fraude/