PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

MACHINE LEARNING APLICADO A DATOS FINANCIEROS

JAVIER SALAMANCA CÁCERES
PAULINA VELÁSQUEZ VELÁSQUEZ

INFORME DE AVANCE DEL PROYECTO
PARA OPTAR AL TITULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

DICIEMBRE, 2018

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

MACHINE LEARNING APLICADO A DATOS FINANCIEROS

JAVIER SALAMANCA CÁCERES
PAULINA VELÁSQUEZ VELÁSQUEZ

PROFESOR GUÍA: **HÉCTOR ALLENDE CID**

CARRERA: **INGENIERÍA DE EJECUCIÓN EN INFORMÁTICA**

DICIEMBRE, 2018

Índice

Resumen

La minería de datos y machine learning son herramientas altamente potenciales en la identificación de observaciones inusuales en tendencias de patrones, dado que son un conjunto de técnicas robustas que facilitan la toma de decisiones, el proceso knowledge discovery in databases, kdd por sus siglas en inglés, es un campo de la estadística y ciencias de la computación que emplea diversas técnicas y metodologías para el proceso de identificar patrones valiosos en la extracción de la información nueva, útil y novedosa; una de las etapas más importantes es el data mining (minería de datos), donde se realiza la estimación de los parámetros de los modelos probabilísticos como son las redes neuronales, random forest, naive bayes, máquinas de soporte vectorial, modelos lineales generalizados logit, probit y log log.

El fraude se define como la acción contraria a la verdad y a la rectitud, que perjudica a la persona o entidad contra quien se comete, esto conlleva a pérdidas económicas y problemas legales. Hay diferentes tipos de fraude, como son intruso a redes privadas, tarjeta de crédito, telecomunicaciones y lavado de activos.

***Palabras Claves**: Minería de datos, fraude con tarjetas de crédito, algoritmos de aprendizaje.*

Lista de Figuras

1. Introducción

La implementación de la tecnología se ha vuelto una parte esencial de nuestras vidas. Para entender por qué, solo basta con mirar a nuestro alrededor y ver que en todo momento y contexto estamos rodeados por ella; ya sea que estemos trabajando o descansando, siempre está presente para hacer nuestras vidas más sencillas.

Debido a su aplicación, nuestro nivel de vida ha mejorado, pues las necesidades se satisfacen con mayor facilidad. De manera general todas las industrias se ven beneficiadas por ella, ya sea la medicina, el turismo, la educación, el entretenimiento entre muchos otros. Además no sólo las empresas han crecido y se han hecho más eficientes, sino también el trabajo que hacemos día a día de manera independiente.

Las tarjetas de crédito hacen parte del crecimiento comercial global de las economías de los países emergentes y desarrollados por sus canales tradicionales y en línea con el mundo; su crecimiento exponencial de los sistemas de transferencia de dinero en línea ha contribuido en la expansión del comercio electrónico y a un número mayor de consumidores en compra y venta de productos de bienes y servicios en que Chile no es la excepción debido a su infraestructura tecnológica en redes móviles e Internet.

El fraude es tan antiguo como la humanidad y puede tomar una variedad de formas ilimitadas. Sin embargo, en años recientes, el desarrollo de las nuevas tecnologías ha proporcionado maneras más extensas en que los delincuentes pueden cometer fraude. Formas tradicionales como el lavado de activos se han puesto más fácil de perpetuar y se ha unido a nuevos tipos de fraude como: fraude en telecomunicaciones móviles, detección de intrusos en redes y fraudes en tarjetas de crédito.

La tarea de detección de fraude no es un tema fácil de resolver, teniendo en cuenta las múltiples modalidades y evolución rápida que este tema ha tenido en la actualidad. Las entidades financieras a nivel mundial utilizan la ciencia de la estadística con herramientas de la minería de datos y el machine learning para reconocimiento de patrones de comportamiento fraudulento. Para ello la mayoría de los sistemas de detección actuales ofrecen dos tipos de alerta: alerta por calificación probabilística y por cumplimiento de reglas, en el primer tipo de alerta casi siempre se utilizan modelos predictivos para una calificación "score". Para el segundo caso se emplean filtros basados en sentencias de comandos sql.

En esta oportunidad se pretenden aplicar las técnicas de machine learning para crear y comparar modelos que permitan la predicción de fraudes para las tarjetas bancarias.

Las técnicas de minería de datos y machine learning emplean modelos probabilísticos eficientes como: modelos de regresión generalizados, redes neuronales artificiales, arboles de decisión y redes de creencia bayesiana para determinar y predecir con una probabilidad o "score.[el] fraude. Utilizan un sistema de aprendizaje autónomo para el reconocimiento de patrones y tendencias basados en hechos históricos, se utilizan los datos de transacciones hechas por los clientes para determinar los patrones, estos permiten identificar rápidamente circunstancias ajenas al comportamiento cotidiano de un cliente que pueden ser indicios de fraude.

El documento comienza con la descripción del problema, planteamiento de los objetivos generales y específicos del proyecto. A continuación se especifica el plan de trabajo a través de la metodología a seguir y la carta Gantt. Marco teórico y conceptos fundamentales de los algoritmos de aprendizaje y estado del arte donde se describen las técnicas para la detección del fraude, realizando énfasis en las técnicas de minería de datos, ya que esto ayuda a comprender y contextualizar mejor el tema central.

2. Presentación del Tema

En esta sección se explicará la situación actual del problema, como el objetivo general y específicos que se pretenden lograr en este proyecto.

2.1. Descripción del Problema

Cuando hablamos de machine learning (ML) nos referimos a sistemas informáticos que aprenden automáticamente, es decir, sin intervención humana. El machine learning es una rama de la inteligencia artificial que a través de algoritmos y grandes volúmenes de datos identifican patrones complejos, procesándolos para predecir modelos de comportamiento. Las entidades financieras han sido las pioneras tradicionalmente en utilizar el Data Mining y ML. Y lo han aplicado principalmente en el núcleo de su negocio, la financiación. Cuando un cliente quiere solicitar un préstamo, el banco le solicita una información. Esta información es utilizada para analizar los datos históricos de los préstamos que tiene concedidos e intentar determinar la probabilidad de que un cliente con determinadas características no pague el préstamo (a través de modelos de ML). Este es el primer requisito que requiere una entidad financiera para conceder un préstamo a un cliente, que pase ese modelo, es decir, que no tenga una gran probabilidad de impago según el modelo estimado. Las entidades financieras emiten tarjetas de crédito para comprar bienes o servicios a través de operaciones bancarias que celebran varios contratos que están implicados entre sí tras una finalidad económica común.[2] El uso de las tarjetas de crédito ha ido adquiriendo una relevancia con el devenir del tiempo, que no es apresurado afirmar que quizá en unos pocos años pasen a ser la forma principal de pago, desplazando, si no por completo, al menos de manera muy ostensible, a los medios de pago tradicionales tales como el dinero. El fraude a establecimientos de comercio y personas es resultado del manejo inadecuado de su información corporativa o personal, existen varias modalidades y técnicas utilizadas por los cibercriminales, en una de ellas usurpan su información por medio de correos maliciosos llamados phishing donde un programa malicioso se apropia de la información personal y corporativa, obteniendo las claves para accesos a cuentas bancarias, ya que estos correos maliciosos recrean portales ficticios de entidades financieras en que el usuario ingresa sus datos y claves. [1] Es por esto que se ha decidido aplicar técnicas machine learning para crear y comparar modelos que permitan la predicción de fraudes para tarjetas bancarias utilizando un sistema de aprendizaje autónomo para el reconocimiento de patrones y tendencias basados en hechos históricos, datos de transacciones hechas por los

clientes para determinar los patrones y así identificar de manera oportuna circunstancias anormales en el uso de las tarjetas.

2.2. Objetivos del Proyecto

En el siguiente apartado se describen los objetivos generales y específicos del proyecto, evidenciando la finalidad y enfoque a la solución que se propone.

2.2.1. Objetivo General

Aprender y aplicar técnicas de machine learning a datos financieros para poder generar un aplicativo capaz de realizar predicciones de fraude de tarjetas bancarias a través del análisis de sets de datos entregados con información histórica de los clientes.

2.2.2. Objetivos Específicos

- Investigar sobre fraude de tarjetas bancarias y machine learning.
- Verificar modelos de análisis de fraude de tarjetas bancarias a través del uso de aprendizaje de máquina.
- Modelar algoritmo de predicción de fraude a través de análisis de data histórica de clientes.
- Estimar la eficiencia de modelos de machine learning para identificar patrones de fraude a tarjetas bancarias.

2.3. Plan de Trabajo

Para organizar las etapas del desarrollo del proyecto de la mejor manera posible, es necesario realizar la planificación correspondiente; considerando la metodología a utilizar y una carta Gantt con la organización de las actividades y responsables asignados.

2.3.1. Metodología

La metodología que se utilizará para el desarrollo del software será la del modelo iterativo incremental, ya que permite obtener un producto por cada iteración que puede ser probado y posteriormente mejorado, permitiendo agregar nuevas funcionalidades y arreglar posibles fallas encontradas permitiendo reducir riesgos e incertidumbre durante el desarrollo del proyecto. Por

cada iteración, se realizan los pasos de análisis, diseño, desarrollo y pruebas, lo que permite validar y re evaluar en cada iteración.

2.3.2. Carta Gantt

	Modo de	Nombre de tarea	Duración	Comienzo	Fin
1		**Plan de trabajo de Machine Learning aplicado a datos financieros**	**80 días**	**lun 20/08/18**	**vie 07/12/18**
2		**1° Iteración**	**30 días**	**lun 20/08/18**	**vie 28/09/18**
3		Buscar información de como las técnicas de machine learning afectan al sector financiero	4 días	lun 20/08/18	jue 23/08/18
4		Buscar información sobre algoritmos de machine learning	6 días	vie 24/08/18	vie 31/08/18
5		Decidir si continuar con opción	1 día	vie 31/08/18	vie 31/08/18
6		Realizar informe de avance	21 días	vie 31/08/18	vie 28/09/18
7		Entregar informe de avance	1 día	vie 28/09/18	vie 28/09/18
8		**2° Iteración**	**32 días**	**vie 28/09/18**	**lun 12/11/18**
9		Capturar Requerimientos	2 días	vie 28/09/18	lun 01/10/18
10		Analizar requerimientos	4 días	lun 01/10/18	jue 04/10/18
11		Analizar opciones para construcción de prototipos	3 días	jue 04/10/18	lun 08/10/18
12		Desarrollar prototipos	9 días	lun 08/10/18	jue 18/10/18
13		Implementación prototipo	18 días	jue 18/10/18	lun 12/11/18
14		**3° Iteración**	**20 días**	**lun 12/11/18**	**vie 07/12/18**
15		Diseño de mejoras del aplicativo	8 días	lun 12/11/18	mié 21/11/18
16		Validación de mejoras del aplicativo	4 días	mié 21/11/18	dom 25/11/18
17		Diseño de pruebas	3 días	dom 25/11/1	mar 27/11/18
18		Analizar resultados obtenidos	3 días	mar 27/11/18	jue 29/11/18
19		Entrega de Software terminado	1 día	vie 30/11/18	vie 30/11/18
20		Entrega informe final	1 día	vie 07/12/18	vie 07/12/18

Figura 1: Carta Gantt.

3. Marco Teórico

El siguiente apartado considera una recopilación de antecedentes e investigaciones previas con el que se sustentará el proyecto.

3.1. Machine learning

Una de las mayores diferencias entre computadoras y humanos es la capacidad de estos últimos para mejorar la manera de afrontar problemas. Intentando adquirir información de errores anteriores(¿por qué fallo?) e intentando encontrar nuevas soluciones desde otros enfoques. El ML investiga el cómo las máquinas pueden aprender, lo que permite a las máquinas reconocer patrones complejos y tomar decisiones inteligentes basadas en la data que reciben y optimizar su proceso en el descubrimiento de nuevos patrones dado a su aprendizaje[10]. Los sistemas ML normalmente se clasifican por su estrategia de aprendizaje subyacente, existen dos roles cruciales, el profesor y el aprendiz. El profesor tiene el conocimiento para realizar una tarea y el aprendiz debe aprender ese conocimiento para realizar la tarea. Las estrategias de aprendizaje pueden ser distinguidas por la cantidad de inferencias realizadas por el aprendiz de la información entregada por el profesor[4].

3.2. Algoritmos de Aprendizaje

Dado al aprendizaje automático, es razonable suponer que hay un proceso oculto que explica los datos que se observan, aunque se conocen los detalles de este proceso, se sabe que no es completamente aleatorio. Esto representa la posibilidad de encontrar una aproximación buena y útil aunque no podamos identificar el proceso por completo; los modelos matemáticos definidos en los parámetros se pueden usar para esta tarea, la parte del aprendizaje y el método de ensamble de modelos consiste en elegir los parámetros no desconocidos que permitan optimizar un criterio de rendimiento con respecto a los datos observados. El aprendizaje puede llevarse a cabo de manera supervisada y no supervisada:

- Aprendizaje supervisado: El sistema recibe data sets (sets de datos), con diferentes parámetros de ejemplo, valores y clasificaciones de los cuales infiere una función matemática que mapea una señal de entrada a una de salida. Gracias a esto la máquina descubre que debe hacer.

- Aprendizaje no supervisado: El sistema hace cosas y observa la consecuencia de lo que hace sin tener referencias, por lo que solo se basa

en lo previamente ocurrido. Estos métodos tienen baja performance en un principio pero a medida de que van calibrado sus acciones mejora.

3.2.1. Naive bayes

Naive bayes es un clasificador estadístico supervisado.Está basado en el teorema de bayes en la teoría de hipótesis de independencia entre las variables X1, ..., Xp. El algoritmo naive clasificador, es altamente eficiente en rendimiento de precisión y velocidad al aplicarse en grandes bases de datos.

En términos bayesianos, X es considerado como evidencia o la observación de las variables independientes X1, ..., Xp, en que describe n atributos de un conjunto de datos dado a su aprendizaje supervisado. H expresa la hipótesis dado a su clase c. La probabilidad P(C | X) donde C es la hipótesis y X es la evidencia donde pertenece la clase, y es la probabilidad posteriori de C condicionada de X [3]. Es decir:

1. D es un set de entrenamiento de tuplas y sus clase de clasificación asociada. Cada tupla es representada por un vector de atributos n-dimensional, X = (x1,...,xn), que representan n medidas realizadas a la tupla de n atributos, respectivamente A1,...,An.

2. Suponiendo que hay m clases, C1,...,Cm. Dada una tupla, X, el clasificador va a predecir que X pertenece a la clase que tenga mayor probabilidad posterior, condicionado en X. Es decir, el clasificador predice que la tupla X pertenece a la clase Ci si y sólo si:

$$P(C_i|\boldsymbol{X}) > P(C_j|\boldsymbol{X}) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

   Por lo tanto, se maximiza P(Ci |X). La clase Ci para la cual P(Ci |X) esta maximizado es llamada la hipótesis máxima posteriori por el teorema de Bayes.

3. Como P(x) es constante para todas las clases, solo P(X|Ci)P(Ci) necesita ser maximizada. Si las probabilidades previas de la clase no se conocen, entonces se asume que las clases son igualmente probables, es decir, P(C1) = P(C2) = ··· = P(Cm), y por lo tanto se maximiza P(X|Ci). De otra manera, se maximiza P(X|Ci)P(Ci). Las probabilidades previas de clases pueden ser estimadas por P(Ci) = |Ci,D|/|D|,

donde |Ci,D| es el número de duplas de entrenamiento de la clase Ci en D.

4. Dados data sets con muchos atributos, sería extremadamente costoso, computacionalmente, calcular P(X|Ci). Para reducir el cálculo en la evaluación P(X|Ci), se hace la suposición "naïve" de independencia condicional de clase. esto supone que los valores de los atributos son condicionalmente independientes el uno del otro, dada la etiqueta de clase de la tupla (ej. que no existan dependencias entre los atributos). De esta manera:

$$P(\boldsymbol{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \qquad = P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i).$$

Es posible estimar las probabilidades P(x1|Ci), P(x2|Ci),..., P(xn|Ci) de las tuplas de entrenamiento. Se toma en cuenta que xk se refiere al valor del atributo Ak para la tupla X. Para cada atributo, analizamos si el atributo es categórico o de valor continuo. Por ejemplo, para calcular P (X | Ci), consideramos lo siguiente:

- Si Ak es categórico, entonces P(xK |Ci) es el número de tuplas de la clase Ci en D teniendo el valor xk para Ak, dividido por |Ci,D|, el numero de tuplas de la clase Ci en D.
- Si Ak es valor continuo, entonces se supone que un atributo de valor continuo tiene una distribución gaussiana con una media u y una desviación estándar o, definida por

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

Entonces:

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

Es necesario calcular uCi y oCi, que corresponden a las media y desviación estándar, respectivamente, de los valores de atributo Ak para tuplas de entrenamiento Ci. Luego se conectan las dos cantidades en la ecuación, ambas se conectan en conjunto con xk, para estimar P(ck |Ci).

5. Para predecir la etiqueta de clase de X, P(x|Ci)P(Ci) es evaluado por cada clase Ci. El clasificador predice que la etiqueta de clase de tupla X es la clase Ci si y sólo si

$$P(\boldsymbol{X}|C_i)P(C_i) > P(\boldsymbol{X}|C_j)P(C_j) \quad \text{for } 1 \leq j \leq m, j \neq i.$$

En otras palabras, la etiqueta de clase predicha es la clase Ci para la cual P(X|Ci)P(Ci) es el máximo.[3]

3.2.2. Neural Networks

La red neuronal artificial (RNA) es un modelo matemático inspirado en el comportamiento biológico de las neuronas y en cómo se organizan formando la estructura del cerebro, algunas neuronas se activan a través de sensores que perciben el entorno, otras neuronas se activan a través de conexiones ponderadas de neuronas previamente activas.

Las redes neuronales intentan aprender mediante ensayos repetidos como organizarse mejor a sí mismas para conseguir maximizar la predicción. Un modelo probabilístico de una red neuronal se compone de nodos que actúan como input, output o procesadores intermedios y cada nodo se conecta con el siguiente conjunto de nodos mediante una serie de trayectorias ponderadas. Basado en un paradigma de aprendizaje, el modelo toma el primer caso y toma inicial basada en las ponderaciones.[4]

Los principios básicos de las redes neuronales artificiales (RNA) fueron formulados por primera vez por McCulloch, y Pitts en 1943, en términos de cinco supuestos, de la siguiente manera:

1. La actividad de una neurona (RNA) es todo o nada.
2. Un cierto número fijo de sinapsis más grandes que 1 debe ser excitado dentro de un intervalo dado de adición neural para que una neurona sea excitada.
3. El único retraso significativo dentro del sistema neuronal es el retraso sináptico.

4. La actividad de cualquier sinapsis inhibidora impide absolutamente la excitación de la neurona en ese momento.

5. La estructura de la red de interconexión no cambia con el tiempo.

   a) Por la suposición anterior, la neurona es un elemento binario.

   b) Mientras que probablemente estos sean históricamente los principios sistemáticos más antiguos, no todos se aplican al estado actual del diseño de RNA de hoy en día.

6. La Ley de Aprendizaje de Hebbian (Regla de Hebbian) gracias a Donald Hebb (1949) también es un principio ampliamente aplicado. La Ley de Aprendizaje Hebbian establece que: Çuando un axón de la célula A es casi suficiente para excitar la célula B y cuando A participa repetida y persistentemente en el disparo de B, entonces se produce algún proceso de crecimiento o cambio metabólico en una o ambas células de tal manera que la eficiencia de la célula A se incrementa "(es decir, se aumenta el peso de la contribución de la salida de la celda A al disparo anterior de la celda B).

Sin embargo, el empleo de pesos a la entrada de cualquier neurona de una RNA y la variación de estos pesos de acuerdo con algún procedimiento, es común a todas las RNA. Se lleva a cabo en todas las neuronas biológicas.

3.2.3. Random Forest

Random forest implica elegir un conjunto de características al azar y crear un clasificador con una muestra inicial de los datos de entrenamiento. Se genera una gran cantidad de árboles (clasificadores) de esta forma y finalmente, la votación no ponderada se usa para asignar un valor desconocido a una clase.

Este clasificador consiste de una combinación de árboles clasificadores donde cada uno es generado utilizando un vector aleatorio muestreado independientemente del vector de entrada, cada árbol arroja una votación para la clase más popular para clasificar un vector de entrada. Diseñar un árbol de decisiones requiere de elegir una medida de selección de atributo y un método de poda. Hay muchos enfoques para la selección de atributos utilizados para la inducción del árbol de decisión y la mayoría de los enfoques asignan una medida de calidad directamente al atributo. El clasificador de bosque aleatorio usa el índice de Gini como una medida de selección de atributo, que mide la impureza de un atributo con respecto a las clases. Para un

conjunto de entrenamiento dado T, seleccionando un caso al azar y diciendo que pertenece a alguna clase Ci, el índice de Gini se puede escribir como:

$$\sum \sum_{j \neq i} (f(C_i, T)/|T|)(f(C_j, T)/|T|)$$

Cada vez que un árbol crece hasta la profundidad máxima en nuevos datos de entrenamiento usando una combinación de características. Estos árboles completamente desarrollados no están podados. Esta es una de las principales ventajas del clasificador de bosque aleatorio sobre otros métodos de árbol de decisión. Los estudios sugieren que la elección de los métodos de poda y no las medidas de selección de atributos afecta el rendimiento de los clasificadores basados en árboles. El número de características utilizadas en cada nodo para generar un árbol y el número de árboles que se cultivarán son dos parámetros definidos por el usuario necesarios para generar un clasificador de bosque aleatorio. En cada nodo, solo se buscan las mejores características para la mejor división.

Por lo tanto, el clasificador de bosque aleatorio consiste en N árboles, donde N es la cantidad de árboles que se cultivarán, que puede ser cualquier valor definido por el usuario. Para clasificar un nuevo conjunto de datos, cada caso de los conjuntos de datos se pasa a cada uno de los N árboles. El bosque elige una clase que tenga el mayor número de N votos, para ese caso.[7]

3.3. ¿Qué significa que un conjunto de datos esté desequilibrado?

Un conjunto de datos desequilibrado es donde el número de observaciones pertenecientes a un grupo o clase es significativamente mayor que las pertenecientes a las otras clases.

Esto ocurre en casos como la detección de fraudes con tarjetas de crédito, donde puede haber solo 1000 casos de fraude en más de un millón de transacciones, lo que representa un escaso 0,1 % del conjunto de datos.

3.4. Métodos para manejar un conjunto de datos desequilibrado

Existen dos métodos principales para manejar los conjuntos de datos desequilibrados:

1. Oversampling: Este método implica reducir o eliminar el desequilibrio en el conjunto de datos al replicar o crear nuevas observaciones de la clase minoritaria. Hay cuatro tipos de técnicas de sobremuestreo:

    - Random Oversampling: En este caso, las nuevas instancias de la clase minoritaria se crean mediante la reproducción aleatoria de muestras existentes para aumentar el recuento minoritario en el conjunto de datos. Sin embargo, este método puede llevar a un ajuste excesivo, ya que simplemente replica las instancias ya existentes de la clase minoritaria.
    - Cluster-based oversampling: El algoritmo de K-medias se aplica por separado a las instancias mayoritarias y minoritarias. Esto ayuda a identificar los grupos en el conjunto de datos. Después de la identificación, cada grupo se sobreexplota de manera que todos los grupos tienen el mismo número de observaciones. Nuevamente, existe el riesgo de adaptar el modelo con este método.
    - Synthetic Oversampling: Este método ayuda a evitar el sobreajuste. En este método, se elige un pequeño subconjunto de minoría y se crean ejemplos sintéticos de este subconjunto para equilibrar el conjunto de datos general. Esto agrega nueva información al conjunto de datos y aumenta el número total de observaciones.
    - Modified Synthetic Oversampling: Este método es igual que el método de sobremuestreo sintético, sin embargo, este deja espacio para el ruido y las distribuciones inherentes de la clase minoritaria.

2. Under-sampling: En este método, el desequilibrio del conjunto de datos se reduce al centrarse en la clase mayoritaria. A continuación se explica la técnica más popular de under-sampling:

    - Random Undersampling: En este caso, las instancias existentes de la clase mayoritaria se eliminan aleatoriamente. Esta técnica no es la mejor porque puede eliminar información o puntos de datos que podrían ser útiles para el algoritmo de clasificación.

3.5. Medición del rendimiento del algoritmo en un conjunto de datos desequilibrado

Para medir el rendimiento primero se deben definir algunos términos.

- Falso Positivo (FP): Esto se utiliza para describir predicciones positivas que son realmente negativas en la vida real. Un ejemplo será predecir que una transacción con tarjeta de crédito es fraudulenta cuando en realidad no lo es.
- Verdadero positivo (TP): Se utiliza para describir predicciones positivas que son realmente positivas en la vida real.
- Falso negativo (FN): Se usa para describir predicciones negativas que son realmente positivas en la vida real. Un ejemplo será predecir que una transacción con tarjeta de crédito no es fraudulenta cuando en realidad lo es.
- Negativo verdadero (TN): Se usa para describir predicciones negativas que son realmente negativas en la vida real.

Algunas medidas e indicadores del desempeño del modelo incluyen:

- Precisión: Este es un indicador de la cantidad de elementos correctamente identificados como positivos del total de elementos identificados como positivos. La fórmula se da como: TP / (TP + FP).
- Recall / Sensitivity / True Positive Rate (TPR): Este es un indicador de la cantidad de elementos correctamente identificados como positivos del total de positivos reales. La fórmula se da como: TP / (TP + FN).
- Specificity / True Negative Rate (TNR) : Este es un indicador del número de elementos correctamente identificados como negativos del total de negativos reales. Fórmula se da como: TN / (TN + FP).
- F1 Score: Este es un puntaje de desempeño que combina precisión y destitución. Es una media armónica de estas dos variables. La fórmula se da como: 2*Precision*Recall / (Precision + Recall).
- Puntuación del coeficiente de correlación de Matthew: Es un puntaje de desempeño que toma en cuenta los positivos verdaderos y falsos, así como los negativos verdaderos y falsos. Este puntaje da una buena evaluación para el conjunto de datos desequilibrado. La fórmula se da como:

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \times (\mathrm{FN} + \mathrm{TN}) \times (\mathrm{TP} + \mathrm{FN}) \times (\mathrm{FP} + \mathrm{TN})}}$$

Figura 2: MCC.

- Área bajo la curva ROC: Una curva ROC (curva característica del receptor) es un gráfico que muestra el rendimiento de un algoritmo de clasificación en todos los umbrales de clasificación. Esta curva traza TPR vs. FPR.
- Área bajo la curva de recuperación de precisión: Una curva de recuperación de precisión es un gráfico que muestra el rendimiento al trazar la precisión contra la recuperación.
- Matriz de confusión: Esta es una representación gráfica de TP, FP, FN y TN. Una matriz de confusión generalizada se da a continuación:

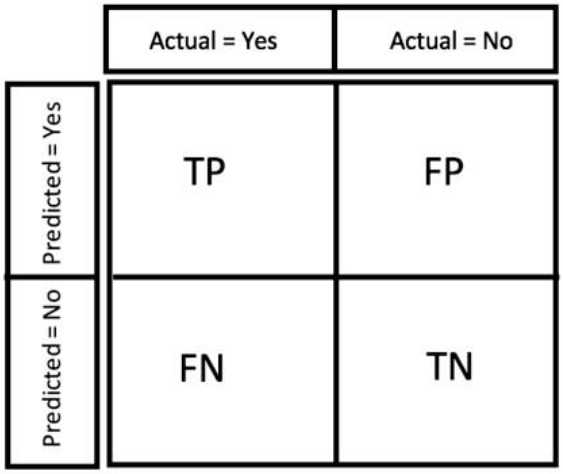

Figura 3: Matriz de Confusión.

En general, las mejores medidas para los conjuntos de datos desequilibrados son: la puntuación de correlación del coeficiente de Matthew, la puntuación de F1 y el área bajo la curva de recuperación de precisión.

3.6. Oversampling y Undersampling

El Oversampling y undersampling en el análisis de datos son técnicas utilizadas para ajustar la distribución de clases de un conjunto de datos (es decir, la relación entre las diferentes clases / categorías representadas).

El motivo habitual para el oversampling es corregir un sesgo en el conjunto de datos original. Un escenario donde es útil es cuando se entrena a un clasificador utilizando datos de entrenamiento etiquetados de una fuente sesgada, ya que los datos de entrenamiento etiquetados son valiosos pero a menudo provienen de fuentes no representativas.

Técnicas de Oversampling para problemas de clasificación:

- SMOTE: Hay una serie de métodos disponibles para remuestrear un conjunto de datos utilizado en un problema de clasificación típico. La técnica más común se conoce como SMOTE: técnica de sobre muestreo de minorías sintéticas. Para ilustrar cómo funciona esta técnica, se consideran algunos datos de entrenamiento que tienen muestras y características en el espacio de características de los datos. Se debe tener en cuenta que estas características, por simplicidad, son continuas.

- ADASYN: El enfoque de muestreo sintético adaptativo, o algoritmo ADASYN, se basa en la metodología de SMOTE, al cambiar la importancia del límite de clasificación a aquellas clases minoritarias que son difíciles. ADASYN utiliza una distribución ponderada para diferentes ejemplos de clases minoritarias según su nivel de dificultad en el aprendizaje, donde se generan más datos sintéticos para los ejemplos de clases minoritarias que son más difíciles de aprender.

3.7. Cross Validation

Este proceso de decidir si los resultados numéricos que cuantifican las relaciones hipotéticas entre variables, son aceptables como descripciones de los datos, se conoce como validación. Generalmente, una estimación de error para el modelo se realiza después del entrenamiento, mejor conocida como evaluación de residuos. En este proceso, se realiza una estimación numérica de la diferencia en las respuestas predichas y originales, también llamada error de entrenamiento. Sin embargo, esto solo nos da una idea de qué tan bien se desempeña nuestro modelo con los datos utilizados para entrenarlo. Ahora es posible que el modelo no se adapte a los datos o lo haga en exceso. Entonces, el problema con esta técnica de evaluación es que no da una indicación de qué tan bien el alumno generalizará a un conjunto de da-

tos independientes / invisibles. Conseguir esta idea sobre nuestro modelo se conoce como cross validation.

4. Estado del Arte

El contenido que se presenta a continuación describe una serie de estudios realizados sobre el tema.

4.1. El Problema de la Detección del Fraude

El problema en la detección de fraude, radica en el análisis de perfiles de usuario que permitan analizar el comportamiento de un cliente, con el fin de detectar anomalías. En CRM (Customer Resource Management), el análisis en la información de un usuario, implica una cadena de datos como se muestra en la figura 2.

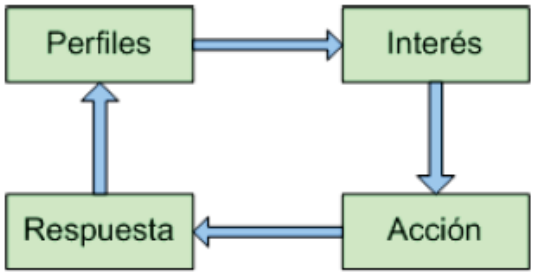

Figura 4: Cadena de datos CRM.

La cadena consiste en cuatro clases de datos [8]:

- Datos de perfil: Datos que representan información histórica del usuario tal como: nombre, profesión, edad, etc.
- Datos de Interés: Datos que representan las tendencias de interés del cliente en los productos de la compañía.
- Datos de Acción: Datos que representan las transacciones entre el cliente y la compañía.
- Datos de Respuesta: Datos que representan la información de servicio al cliente.

En la práctica de la construcción de perfiles de usuario, el procedimiento incluye cuatro pasos:

1. Limpieza de datos, para eliminar datos redundantes, con el fin de tener un análisis efectivo de detección de fraude.

2. Selección y Extracción de características, que permitan descubrir indicadores, correspondientes a cambios en comportamientos que indiquen fraude.

3. Modelamiento, para determinar patrones de fraude por un clasificador [9].

4. Monitoreo y predicción de fraude, con el fin de emitir alarmas.

De los cuatro pasos anteriores, el modelamiento y predicción de fraude son los más importantes y tienen amplia discusión en el campo del aprendizaje de máquinas. Adicionalmente, una de las dificultades en la detección de fraude, es que típicamente la mayoría de los datos son legítimos.

4.2. Minería de datos y descubrimiento de conocimiento en bases de datos

Debido a los grandes volúmenes de datos que una organización puede llegar a tener, el obtener conocimiento a partir de estos no es una tarea fácil. Con este fin investigadores han dado origen a campos de investigación como el descubrimiento de conocimiento en bases de datos[10] (Knowledge Discovery in Database - KDD), y en especial, el proceso de minería de datos (Data Mining).

El término 'KDD' es empleado para describir el proceso total de descubrimiento y extracción de conocimiento nuevo, no obvio a partir de un conjunto de datos, el cual esta conformado por relaciones y patrones entre los elementos que conforman los datos [11]. El proceso de KDD abarca varias etapas en su realización, desde la selección de datos que pueden ser necesarios para descubrir conocimiento, hasta visualizar los resultados de dicho descubrimiento. El principal proceso dentro del KDD es la minería de datos 'Data Mining', que es la responsable de buscar, descubrir y extraer el conocimiento desde los datos [9].

A continuación se presenta la taxonómica general del proceso KDD, como se ve en la figura 3 (tomada de Jiawei Han[12]).

El proceso comienza con:

1. Preparación: Esta etapa consiste en determinar que datos de la base de datos, vamos a seleccionar para el proceso de extracción de conocimiento. Dentro de las tareas que se deben hacer en esta etapa tenemos: Selección de datos, Limpieza, Enriquecimiento y Codificación.

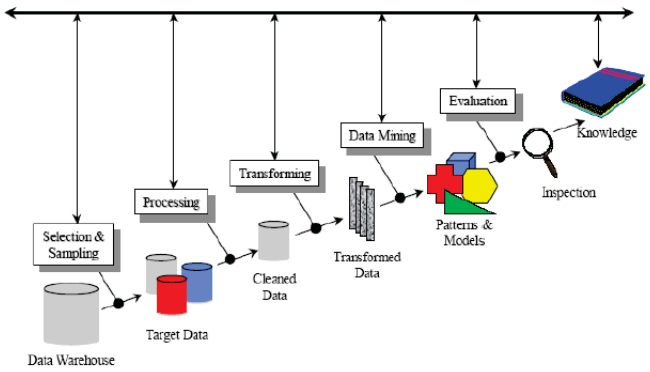

Figura 5: Etapas del proceso KDD.

2. Extracción o Minería: La minería de datos, es el proceso que pretende examinar la vasta cantidad de datos en una base de datos, en busca de patrones recurrentes, detectando tendencias y desenterrando hechos; intenta hallar conocimiento con una mínima o ninguna instrucción u orientación de analistas, todo ello en el menor tiempo posible. Con este conocimiento, el analista empresarial ejercita su habilidad y experiencia en la materia para separar los hechos útiles de los inútiles.

3. Presentación: En esta etapa se reporta los resultados obtenidos en el proceso de minería de datos. Muchas veces los usuarios se enamoran de una herramienta por los gráficos que despliegan. Las mejores gráficas que una herramienta puede mostrar son aquellas que el usuario entiende. Eso no quiere decir que las gráficas animadas y con mucho colorido no sean buenas, simplemente que los usuarios muchas veces no tienen los conocimientos necesarios sobre el tema al que realizaron minería, por lo que no pueden interpretar los resultados y no pueden definir si los resultados arrojados son buenos o son malos para la organización.

4.3. Técnicas para la detección de fraudes

La detección de fraude no es un tema trivial, las metodologías usadas por los "falsificadores" no son las mismas de hace algunos años; cuando las entidades identifican un patrón de comportamiento, los "falsificadores" ya están pensando en otras alternativas. Actualmente las herramientas para la detección de fraude se pueden clasificar en dos categorías:

- Técnicas tradicionales
- Técnicas de Minería de Datos

4.3.1. Técnicas Tradicionales

Los métodos tradicionales de detección de fraude consisten en una combinación de investigadores y herramientas que reportan alarmas de posibles sospechosos; para ello se utilizan técnicas como:

1. Identificación de los clientes que coinciden en listas de control como OFAC, DATACREDITO, etc, emitidas por entes internacionales o nacionales.

2. Sistemas basados en la aplicación de reglas que constan de sentencias SQL, definidas con la ayuda de expertos. Esta estructura puede detectar sumas acumulativas de dinero, ingresadas a una cuenta en un corto período de tiempo, como un día.

3. Métodos de clasificación estadísticos, como el análisis de regresión de datos para detectar comportamientos anómalos de cambio en una cuenta dada una serie de transacciones que efectúa un cliente en un lapso de tiempo.

4. Análisis de relaciones: Este análisis permite encontrar relaciones entre elementos de información como transacciones, cuentas y participantes. Esta técnica requiere un esquema supervisado.

4.3.2. Técnica de Minería de Datos

La minería de datos ofrece un rango de técnicas que permiten identificar casos sospechosos basados en modelos. Estos modelos se pueden clasificar en:

- Modelos de datos inusuales: Estos modelos pretenden detectar comportamientos raros en un dato respecto a su grupo de comparación o con el mismo, por ejemplo la consignación de altas sumas de dinero en efectivo. Para este caso, se puede emplear técnicas de análisis de Clustering, seguido de un análisis de detección de Outlier.

- Modelos de relaciones inexplicables: A través de este tipo de modelos se desea encontrar relaciones de registros que tienen iguales valores para determinados campos, resaltando el hecho de que la coincidencia

de valores debe ser auténticamente inesperado desechando similitudes obvias como el sexo, la nacionalidad. Por ejemplo la transferencia de fondos entre dos o más compañías con la misma dirección de envío. Para este caso se pueden aplicar técnicas de Clustering para encontrar grupos sospechosos y reglas de asociación.

- Modelos de características generales de fraude: Con estos modelos se pretende una vez detectado ciertos casos, hacer predicciones de futuros ingresos de transacciones sospechosas. Para estas predicciones usualmente se emplean técnicas de regresión, árboles de decisión y redes neuronales.

De igual forma, taxonómicamente la minería de datos se puede dividir en dos clases: descriptiva y predictiva [12] según como se presenta en la figura 4.

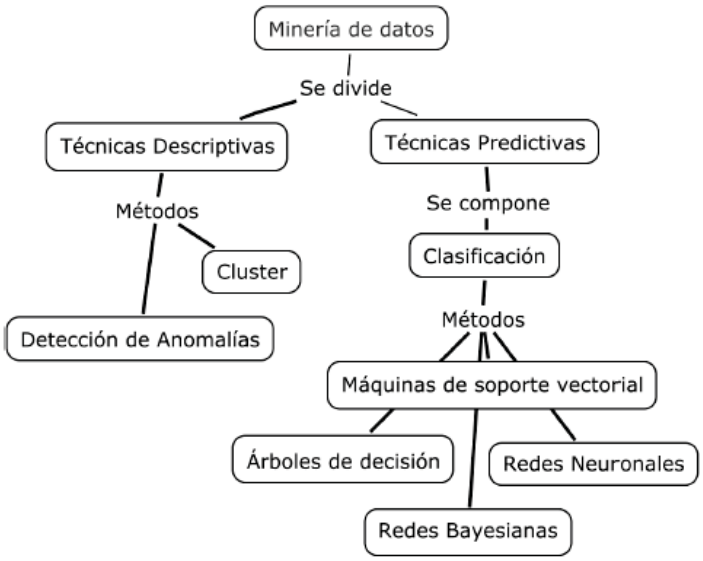

Figura 6: Minería de Datos

1. Técnicas de Minería Descriptiva: El objetivo de este tipo de minería, es encontrar patrones (correlaciones, tendencias, grupos, trayectorias y anomalías) que resuman relaciones en los datos. Dentro de las principales técnicas descriptivas encontramos:

a) Detección de Anomalías(Outlier): La meta principal en la detección de Anomalías, es encontrar objetos que sean diferentes de los demás. Frecuentemente estos objetos son conocidos como Outlier [13].

La detección de anomalías también es conocida como detección de desviaciones [16], porque objetos anómalos tienen valores de atributos con una desviación significativa respecto a los valores típicos esperados.

Aunque los Outlier son frecuentemente tratados como ruido o error en muchas operaciones, tales como clustering, para propósitos de detección de fraude, son una herramienta valiosa para encontrar comportamientos atípicos en las operaciones que un cliente realiza en una entidad financiera.

b) Clustering: El análisis de cluster es un proceso que divide un grupo de objetos, de tal forma que los miembros de cada grupo son similares de acuerdo a alguna métrica. El agrupamiento de acuerdo a la similitud, es una técnica muy poderosa, la clave para esto es trasladar alguna medida intuitiva de similitud dentro de una medida cuantitativa[14], como se ilustra en la figura 5.

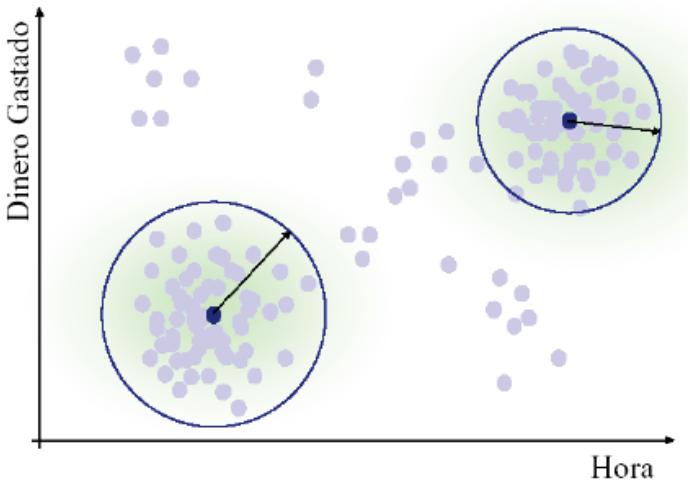

Figura 7: Clustering.

Las técnicas de clustering son utilizadas comúnmente para hacer segmentación y su gran aplicación está en estrategias de merca-

deo, mediante las cuales se determinan conjuntos de clientes que poseen el mismo comportamiento para hacer llegar ofertas especialmente diseñadas al perfil de dichos clientes. Las técnicas de segmentación permiten identificar claramente el comportamiento de un grupo de casos que difiere de otros grupos o conjuntos, sin embargo algunos autores plantean que por lo general, los cluster son resultados difíciles de entender.

2. Técnicas De Minería Predictiva: El objetivo de este tipo de minería, es predecir el valor particular de un atributo basado en otros atributos. El atributo a predecir es comúnmente llamado ”clase” o variable dependiente, mientras que los atributos usados para hacer la predicción se llaman variables independientes [18]. Dentro de las principales técnicas predictivas encontramos:

   a) Árboles de Decisión: De las técnicas de aprendizaje, son el método más fácil de utilizar y entender. Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta sus hojas[10]. Se utilizan comúnmente cuando se necesitan detectar reglas del negocio que puedan ser fácilmente traducidas al lenguaje natural o SQL, o en la construcción de modelos predictivos.

   b) Redes Neuronales: Las redes neuronales consisten en "neuronas.º nodos interconectados que se organizan en capas. Por lo regular, los modelos neuronales constan de tres capas: de entrada, oculta y de salida. Cada neurona evalúa los valores de entrada, calcula el valor total de entrada, compara el total con el mecanismo de filtrado (valores de umbral), y en seguida determina su propio valor de salida. El comportamiento complejo se modela conectando un conjunto de neuronas. El aprendizaje o çapacitación.ºcurre modificando la "fuerza de conexión.º los parámetros que conectan las capas. Las redes neuronales se acondicionan con muestras adecuadas de la base de datos.

4.4. Trabajos Realizados

A continuación se presenta una revisión de los trabajos realizados, los cuales se pueden clasificar en dos categorías:

1. Las investigaciones realizadas en torno a la detección de fraude.

2. Las investigaciones de técnicas de Minería que pueden aplicarse al problema detección de Fraude.

4.4.1. Investigaciones en torno a la detección del fraude

El fraude es la actividad más antigua de la humanidad, y puede tomar una variedad de formas diferentes. Las áreas más vulnerables se centran en las tarjetas de crédito, el lavado de activos, el sector de las telecomunicaciones y el sector médico.

Referente al fraude con tarjetas de crédito, Bolton Richard y David Hand han desarrollado una técnica basada en modelos estadísticos concernientes a detectar el comportamiento de fraude a través del análisis longitudinal de los datos. Para ello emplean un método no supervisado que les permite detectar el cambio en el comportamiento de un objeto o detectar transacciones inusuales. El método propuesto por estos autores se llama PGA (Peer Group Analysis, 2001), es una nueva herramienta para monitorear el comportamiento individual de objetos respecto a diferentes objetos, que tiene previamente alguna característica similar. Cada objeto es seleccionado como una clase, y es comparado con todos los objetos en la base de datos, usando criterios de comparación internos y externos de patrones de comportamiento de cada objeto. Esta herramienta intenta ser parte de la minería de datos, en el sentido que tiene un ciclo que detecta objetos anómalos y trata de aislarlo de los demás. Igualmente Jon T.S Quah y M. Sringanesh (2007) desarrollaron una investigación sobre fraude con tarjetas de crédito por Internet [15], para ello emplearon el modelo de redes neuronales SOM(Selt Organizing Maps), la cual consta de tres capas:

- Una capa de inicial de autenticación PIN (Personal Identification Number).

- Una capa de análisis de comportamiento, la cual tiene un método de clasificación en cluster de los datos de entrada, seguido de la aplicación del algoritmo Feed-Forward de una red neuronal.

- Una capa de salida, que presenta si la transacción es sospechoso o no.

De otro lado, Efstathios Kirkos[17] presenta un estudio de métodos basados en técnicas de clasificación de Minería de datos para identificar firmas que emiten estamentos financieros fraudulentos conocido como FFS (Fraudulent Financial Statements). Esta técnica emplea modelos de minería de

datos como: Árboles de decisión (ID3), Redes Neuronales (Feed-Forward) y Redes de Creencia Bayesiana. Los métodos son comparados en términos de su exactitud de predicción.

Clifton Phua [20], presenta un compendio de los trabajos realizados en la aplicación de técnicas de minería para la detección de fraude, para ello dividió su estudio en dos ramas:

- Algoritmos predictivos con aprendizaje supervisado. En esta área se resaltan los trabajos realizados por Ghost y Reilly(1994) que proponen un modelo de redes neuronales usando tres capas con una función radial Feed-Forward denominado RBF(Function Basis Radial). Syeda (2002) propuso una red neuronal con lógica difusa. Ezawa y Norton (1996) desarrollaron un modelo de red Bayesiana de cuatro estados y dos parámetros para detección de fraude en telecomunicaciones.

- Algoritmos con aprendizaje no supervisado, en esta área se resalta los trabajos realizados por Williams y Hung (1997) aplicando tres pasos del algoritmo k-means para la detección de cluster. Brocket (1998) presenta un estudio basado en el modelo de red neuronal SOM (Self Organizing Maps) para la detección de cluster antes de lanzar el algoritmo Back-Propagation para la detección de fraude medico. S. Viaene (2005) presenta un estudio del aprendizaje Bayesiano para redes neuronales de Perceptrón Multicapa(MLP) con el fin de detectar fraude en las reclamaciones privadas de los accidentes que se produjeron en Massachusetts US en 1993 [19].

4.4.2. Trabajos Entorno a Técnicas de Minería de Datos

- Kaustav Das(2007)[21]. El autor aborda el problema de detectar anomalías en grandes conjuntos de datos categóricos. Para ello emplea redes Bayesianas que se basan en tener definido una estructura y un algoritmo de aprendizaje.

- Tianming Hu(2003)[6]. Este autor aborda el tema de la detección de Outlier a partir de la identificación de patrones obtenidos a través de técnicas de clustering.

- J. A. Fernandez Pierna(2001)[5]. Este autor hace un compendio de las principales técnicas utilizadas para la detección de Outlier entre las que se destacan: Método de la incertidumbre, Método de ”convex full”, Distancia de Mahalanobis, XResidual, Potencial Functions, RHM

(Resampling by the halfmeans method), SVM (Smallest half-volume method).

4.5. Evaluación de modelos

El objetivo de esta etapa es evaluar el desempeño del modelos probabilísticos. Para ello, se utiliza una métrica para seleccionar el modelo probabilístico que predice los mejores resultados en función de criterios como accuracy, precision, recall, área bajo la curva roc y f beta score.

1. Matriz de Confusión: Una matriz de confusión está dada por un número de m clases ordenadas en filas y columnas, es simétrica por contener las mismas categorías en filas y columnas, las columnas corresponden a los resultados arrojados por el modelo de pronóstico mientras las filas representan la clasificación real de los individuos, sobre las casillas de la diagonal se identifica a los individuos bien clasificados por el modelo. Para un problema de clasificación binaria, la diagonal principal muestra los verdaderos positivos (VP) y verdaderos negativos (VN), la clasificación errada está dada por falsos positivos (FP) y falsos negativos (FN).

    a) Verdaderos Positivos (V P): Cantidad de casos 'No' fraudulentos que fueron clasificados correctamente por el modelo.

    b) Verdaderos Negativos (V N): Cantidad de casos 'Sí' fraudulentos que fueron clasificados correctamente por el modelo.

    c) Falsos Positivos (F P): Cantidad de casos 'No' fraudulentos que fueron clasificados incorrectamente como 'Sí' fraudulentos.

    d) Falsos Negativos (F N): Cantidad de casos 'Sí' fraudulentos que fueron clasificados incorrectamente como no fraudulentos.

2. Exactitud: La exactitud es un indicador evalúa la capacidad del modelo de clasificar correctamente los casos positivos y negativos las categorías, resultados que parte de los valores clasificados en la matriz de confusión, se calcula como los valores clasificados correctamente de la diagonal principal o traza de la matriz verdaderos positivos y verdaderos negativos sobre el total de las categorías.

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

Figura 8: Exactitud.

3. Precisión: La precisión es la probabilidad promedio de recuperación relevante de información, esta estadística pretende proporcionar una indicación acerca de cuán relevantes son los resultados otorgados por el modelo, la alta precisión significa que un algoritmo arrojó resultados sustancialmente más relevantes que irrelevantes.

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

Figura 9: Precisión.

4. Curva ROC: Muestra a través de un gráfico la compensación entre la tasa de verdaderos positivos (TVP) y la tasa de falsos positivos (TFP).[3]

$$Sensibilidad = TVP = \frac{VP}{P} = \frac{VP}{(VP + FN)}$$

Figura 10: Sensibilidad.

$$TFP = \frac{FP}{N} = \frac{FP}{(FP + VN)}$$

Figura 11: Tasa de Falsos Positivos.

$$Especificidad = \frac{VN}{N} = \frac{VN}{(FP + VN)} = 1 - TFP$$

Figura 12: Especificidad.

El gráfico bidimensional representa en el eje vertical, la proporción de valores positivos de sensibilidad, y en el eje horizontal, la proporción de valores falsos positivos de especificidad, una recta de división desde el punto de origen hasta 1 en ambos ejes del plano; la curva ROC o también llamado AUC el área bajo la curva clásica estadísticamente bajo la hipótesis del mejor modelo que obtenga dado a su aprendizaje y entrenamiento, para poder interpretar estas señales se han establecido intervalos de calificación para los valores del AUC:

1. 0.50 - 0.60 Malo
2. 0.61 - 0.75 Regular
3. 0.76 - 0.90 Bueno
4. 0.91 - 0.97 Muy Bueno
5. 0.98 - 1.00 Excelente

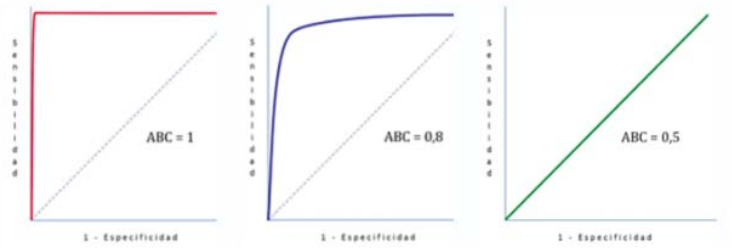

Figura 13: Gráfico Bidimensional.

5. Propuesta de Solución

En esta sección se pretende especificar la solución que se empleará en el proyecto, explicando las tecnologías, herramientas y métricas a utilizar.

5.1. Data Set para detección de fraude con tarjetas de crédito

Los conjuntos de datos contienen transacciones realizadas con tarjetas de crédito en septiembre de 2013 por titulares de tarjetas europeos. Este conjunto de datos presenta las transacciones que ocurrieron en dos días, donde tenemos 492 fraudes de 284,807 transacciones. El conjunto de datos es altamente desequilibrado, la clase positiva fraudes representa el 0.172 % de todas las transacciones. El conjunto de datos contiene variables de entrada numéricas que son el resultado de una transformación de análisis de componentes principales (ACP).

Desafortunadamente, debido a problemas de confidencialidad, no podemos proporcionar las características originales y más información de fondo sobre los datos. Las características V1, V2,..., V28 son los componentes principales obtenidos con ACP, las únicas características que no se han transformado con ACP son tiempo, clase o ítem y cantidad de dinero en euros. La característica tiempo contiene los segundos transcurridos entre cada transacción y la primera transacción en el conjunto de datos. La función Importe es la cantidad de la transacción de dinero en euros, esta función se puede utilizar para el aprendizaje dependiente del ejemplo. Característica Clase o ítem de fraude donde toma el valor (1) en caso de fraude y (0) en caso contrario.

5.2. Diseño de la solución

Se realizará un análisis en busca de la eficiencia óptima del modelo probabilístico con el algoritmo de aprendizaje redes neuronales artificiales, el cual será aplicado a través de la librería “scikit-learn” en python. Para el marco probabilístico del conjunto de datos por fraude con tarjetas de crédito se estratifica por el ítem de fraude y segmentación por el valor en dinero en pesos. Se procede a realizar en dos fases la estimación y predicción: fase 1 un conjunto de datos de entrenamiento “train” realizará la estimación de los modelos probabilísticos de sus parámetros, y fase 2 se procede a realizar las predicciones con un conjunto de datos de prueba "test". El conjunto de datos a utilizar para el “training” y “test” ha sido generado manualmente utilizando las clases de tiempo transcurrido entre transacciones por cliente (tiempo),

número correspondiente a la tarjeta de crédito (numtarjeta) y rubro de la compañía en donde se realizó la compra (rubro).

5.3. Tecnología a utilizar

En el siguiente párrafo se detallan las tecnologías a utilizar para el desarrollo del aplicativo.

5.3.1. Jupyter Notebook

Jupyter Notebook, es un entorno de trabajo interactivo que permite desarrollar código en Python por defecto de manera dinámica, a la vez que integrar en un mismo documento tanto bloques de código como texto, gráficas o imágenes. Es utilizado ampliamente en análisis numérico, estadística y machine learning, entre otros campos de la informática y las matemáticas.

5.3.2. Python

Python es un lenguaje de programación interpretado de alto nivel para programación de propósito general. Cuenta con un sistema de tipo dinámico y gestión automática de memoria. Es compatible con múltiples paradigmas de programación, incluyendo orientado a objetos, imperativo, funcional y de procedimiento, y tiene una biblioteca estándar grande y completa. Los intérpretes de Python están disponibles para muchos sistemas operativos. CPython, la implementación de Python, es un open source software y tiene un modelo de desarrollo basado en la comunidad, al igual que casi todas las otras implementaciones de Python.

5.3.3. scikit-learn

Es una biblioteca para aprendizaje de máquina de software libre para el lenguaje de programación Python. Incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, Gradient boosting, K-means y DBSCAN. Está diseñada para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy.

También posee librerías para escalar la data y seleccionar los mejores parámetros para un estimador.

6. Data y Tratamiento

A continuación se explica la manera en que se desarrollaron los algoritmos de predicción.

6.1. Cargar y evaluar la data

Se obtiene el dataset desde el archivo 'entrenamiento.csv' donde se cuentan cuantas entradas hay para cada clase, obteniendo los siguientes resultados:

- Ejemplos de no fraude: 274.229
- Ejemplos de fraude: 476
- Sin fraudes 99.83 % del conjunto de datos
- Fraudes 0.17 % del conjunto de datos

A continuación se muestra un gráfico con la comparación entre fraude y no fraude:

Figura 14: Comparación fraude y no fraude.

6.2. Escalado de Data

A continuación se escala la data para estandarizar los valores obtenidos, ya que en la descripción del dataset a utilizar se explica que los valores 'V' han sido transformados con PCA por lo que se escaló el monto y el tiempo para poder llevarlos a valores comparables.

6.3. Obtencion de features y targets

Se asignan los features a la variable 'X'. Estos corresponden al tiempo y monto escalados. Los 'V', serán los valores que el modelo utilizará para asociarlos a los target contenidos en la variable 'y' y así aprender a reconocer cuando los registros son fraudes y cuando no.

6.4. Entrenamiento y pruebas

Se separan los datos en entrenamiento y pruebas, donde el entrenamiento tiene un 100 % de los datos ya que posteriormente se utilizará cross validation para generar distintos sets de entrenamiento y pruebas.

6.5. Over-sampling SMOTE

Ya que la data esta muy desbalanceada en cuanto a cantidad de fraudes por no fraudes (0.17 % y 99,83 % respectivamente) el modelo tiende a caer en overfitting, ya que nuestro modelo asumirá que la mayoría de los casos son "no fraude". Es por esto que se aplica la técnica de balanceo .ºver-sampling"la cual nos permite incrementar la cantidad de data de la clase mas baja para igualarse a la mas alta y así el modelo no tienda hacia un lado.

7. Resultados

A continuación se detallan los resultados obtenidos de la ejecución de los algoritmos.

7.1. Estimación de datos originales sin procesamiento

Se realizó una predicción con los estimadores de Redes Neuronales y Naive Bayes Bernoulli sobre los datos originales sin procesamiento. Los resultados fueron los siguientes:

- Redes Neuronales

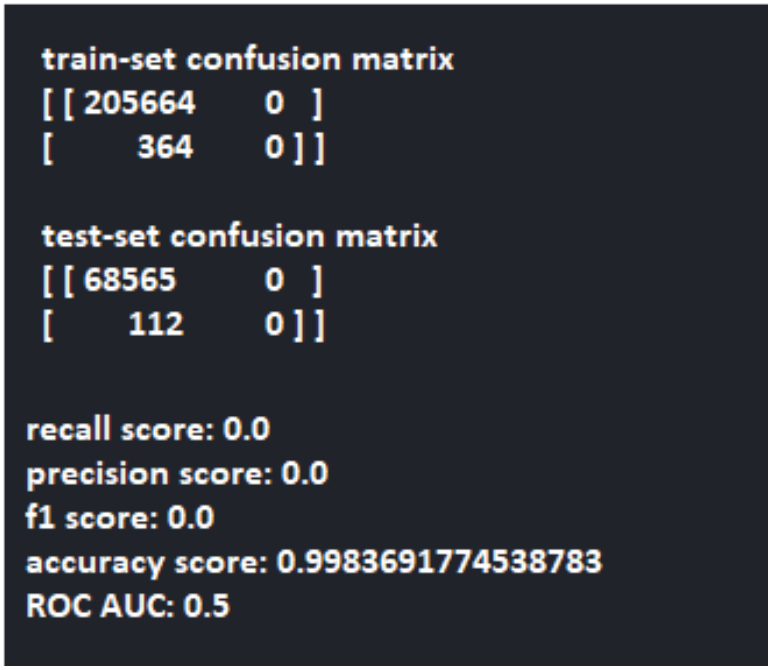

Figura 15: Matriz de Confusión RN.

Se puede observar que Neuronal Networks clasificó bien los datos no fraudes pero no encontró ninguna que sea fraude, ya que el clasificador asume que todos los datos son no fraude. Por ende, clasificó mal los que correspondían a fraude.

- Gaussian Naive bayes

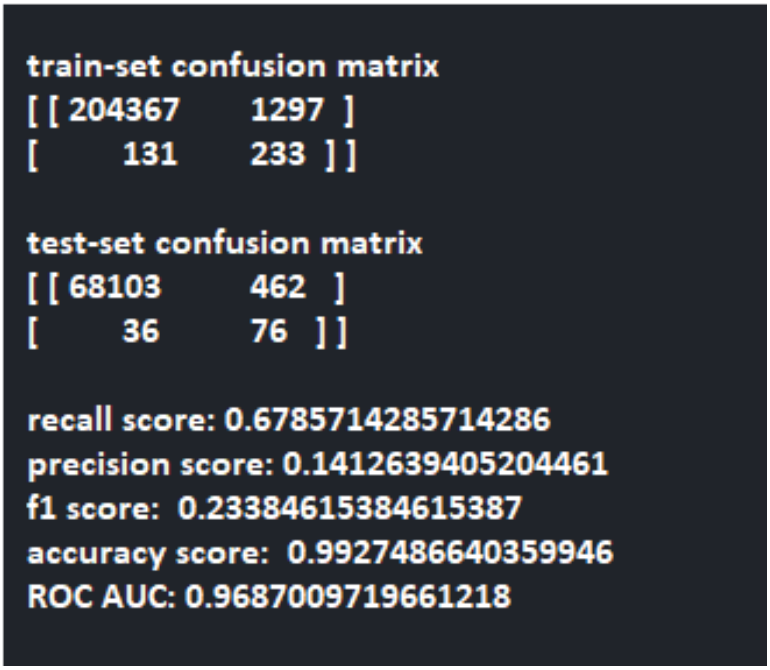

Figura 16: Matriz de Confusión NB.

Se puede observar que los modelos tienen un bajo rendimiento ya que obtienen datos sin procesar.

Según las métricas, ambos modelos tienen un pobre rendimiento, ya que presentan una baja presición, es decir, significa que se arrojaron resultados sustancialmente más irrelevantes que relevantes.

En cuanto a "recall score" Naive Bayes Bernoulli tuvo un mejor rendimiento ya que tuvo como indicador un 0.67 correspondiente a la cantidad de elementos correctamente identificados como positivos del total de positivos reales.

Para estas métricas se concluye que Naive Bayes Bernoulli obtuvo mejor desempeño que el de redes Neuronales.

7.2. Estimación de datos con procesamiento

Se realizó predicción con los estimadores de Redes Neuronales y Naive Bayes sobre los datos procesados. Los resultados fueron los siguientes:

- Redes Neuronales

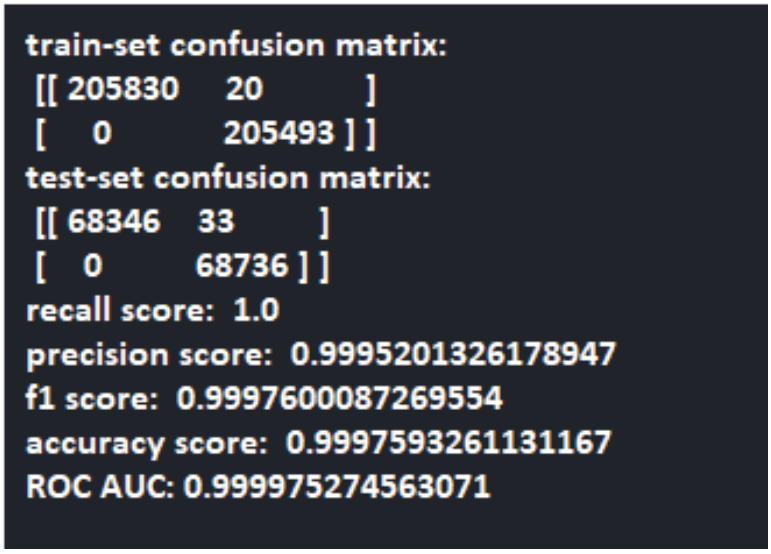

Figura 17: Matriz de Confusión RN.

Se observa que redes neuronales clasificó bien todos los fraudes y sólo se equivoco en clasificar los no fraude como fraude.

- Naive Bayes

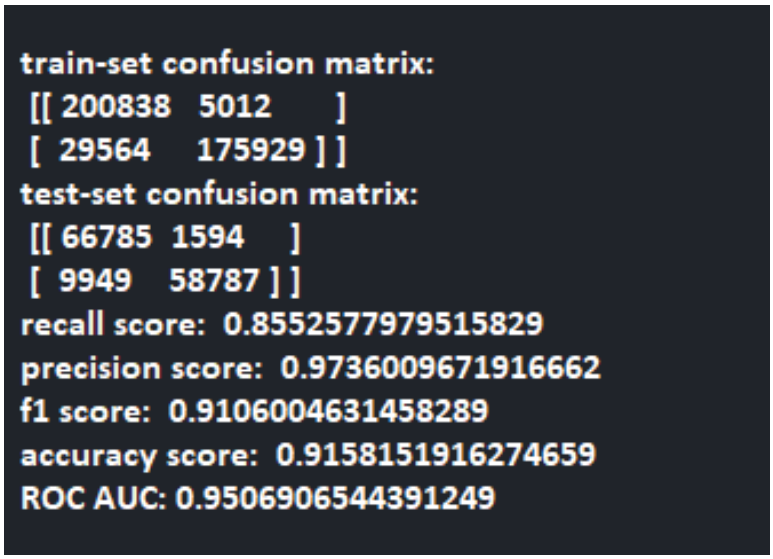

Figura 18: Matriz de Confusión NB.

7.3. Estimación de datos procesados

Para estimar los datos ya procesados, primero se realizo grid search para obtener los hyper parametros óptimos para entrenar nuestro modelo, los cuales se miden a través de la métrica de accuracy para ambos modelos, de manera que primero se crea un diccionario en donde se asignan los parámetros a buscar con sus distintos valores posibles para cada estimador.

Para ambos estimadores el grid search realizará 5 cross validations.

Se realiza entrenamiento para encontrar los parámetros óptimos de cada estimador los cuales son:

```
mlp_best_parameters = mlp.best_params_
print(mlp_best_parameters)
```

```
{'activation': 'relu', 'hidden_layer_sizes': (90, 2), 'solver': 'adam'}
```

Figura 19: Mejores Parámetros.

A partir de esto se obtiene el mejor puntaje obtenido por el mejor esti-

mador con los mejores parámetros.

```
In [28]: mlp_best_result = mlp.best_score_
         print(mlp_best_result)

         0.9997325735179711
```

Figura 20: Mejor Puntaje.

Se hace cross validation utilizando los parámetros obtenidos de grid search, para la exactitud promedio de cada modelo a través de 10 iteraciones de entrenamiento.

```
In [33]: # Cross validation to get best accuracy mean from best Multi-layer Perceptron classifier estimator obtained from grid search
         mlp_all_accuracies = cross_val_score(estimator=MLPClassifier(solver=mlp.best_estimator_.solver,
                                                                      activation = mlp.best_estimator_.activation,
                                                                      hidden_layer_sizes=mlp.best_estimator_.hidden_layer_sizes
                                                                      ),
                                              X=X_smote,
                                              y=y_smote,
                                              cv=5,
                                              scoring='accuracy'
                                             )

In [34]: print(mlp_all_accuracies.mean())
         0.9996888126790285

In [35]: # Cross validation to get best accuracy mean from best  bernoulli naive bayes classifier estimator obtained from grid search
         bnb_all_accuracies = cross_val_score(estimator=BernoulliNB(alpha=bnb.best_estimator_.alpha),
                                              X=X_smote,
                                              y=y_smote,
                                              cv=5
                                             )

In [36]: print(bnb_all_accuracies.mean())
         0.9061527602865125

In [37]: y_pred_mlp = mlp.predict(X_test)
         y_pred_prob_mlp = mlp.predict_proba(X_test)
         train_pred_mlp = mlp.predict(X_smote)
```

Figura 21: Cross Validation.

Los mejores resultados los tiene el de Redes Neuronales, ya que fue el más aproximado.

Se realizaron las predicciones del mejor modelo obtenido con grid search utilizando Redes Neuronales.

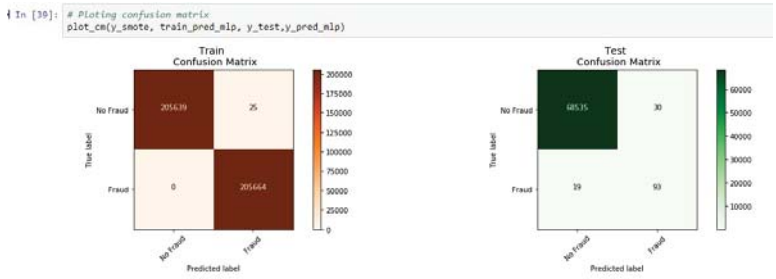

Figura 22: Matriz Confusion RN.

Predicciones del mejor modelo obtenido con grid search utilizando Naive Bayes Bernoulli.

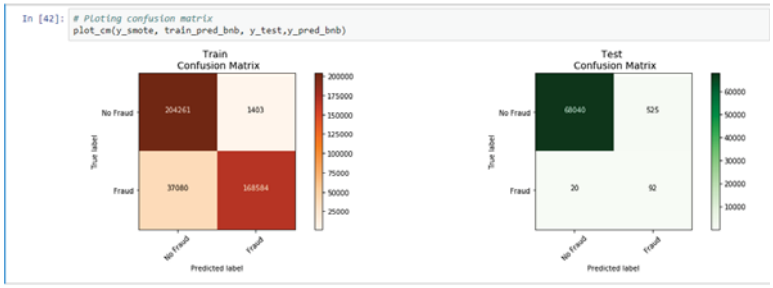

Figura 23: Matriz Confusion NBB.

8. Conclusión

Como se comentó en la introducción, la detección de fraude es una estrategia aplicada después que la prevención a fallado, para ello en la actualidad se usan herramientas tradicionales como la estadística y técnicas de Minería de datos, en especial las referentes a Redes Neuronales, Redes de Creencia Bayesiana y Árboles de Decisión, que han ayudado a encontrar modelos más complejos que las herramientas tradicionales.

Dado que los patrones de Fraude cambian frecuentemente, es importante contar con la participación de expertos en la formulación de reglas; los analistas que están diariamente monitoreando posibles comportamientos de fraude encuentran casos a diario. Dado lo anterior, se debe tener en cuenta que los modelos que proveen las técnicas de Minería de Datos deben ser reentrenados con cierta frecuencia, con el fin de actualizar los modelos con los nuevos datos.

La Minería de datos aporta diferentes tecnologías en la identificación de operaciones fraudulentas. Por lo general es necesario el uso de varias de estas tecnologías con el fin tener un mejor éxito en la solución del problema. La elección exacta y la combinación de estas tecnologías, depende en gran medida de las características de los datos disponibles.

De acuerdo a los resultados obtenidos, podemos conlcuir que el modelo de redes neuronales (multi-layer preceptron) logra predecir de mejor manera las transacciones fraudulentas y no fraudulentas obteniendo mejores resultados que Naive Bayes.

Para que funcionen mejor muchos algoritmos de Machine Learning, hay que normalizar las variables de entrada al algoritmo. Normalizar significa, en este caso, comprimir o extender los valores de la variable para que estén en un rango definido. Sin embargo, una mala aplicación de la normalización, o una elección descuidada del método de normalización puede arruinar los datos, y con ello el análisis. No existe un método ideal de normalización que funcione para todas las formas de variables. Es importante conocer cómo se distribuyen los datos, saber si existen anomalías, comprobar rangos, etc. Con este conocimiento, se puede seleccionar la mejor técnica para no distorsionar los datos.

9. Referencias

[1] BBC,Cómo se producen los fraudes con tarjetas de crédito y las reglas de oro para evitarlos, 2017, Disponible en: https://www.bbc.com/mundo/vert-cap-40638275

[2] Sandoval Ricardo (1991). Tarjeta de Crédito Bancaria. Editorial Jurídica de Chile.

[3] Han Jiawei, Kamber Micheline and Pei Jian (2014) Data Mining Concepts and Techniques Third Edition, Elsevier Science.

[4] F. Camastra and A. Vinciarelli, Machine Learning for Audio, Image and Video Analysis - Theory and Applications, Second Edition. Advanced Information and Knowledge Processing, Springer, 2015.

[5]J. A. Fernandez Pierna and F. Wahl and O. E. de Noord and D. L. Massart. Methods for outlier detection in prediction. Chemometrics and Intelligent Laboratory Systems, Vol 63, pp 27-39, Aug 2002.

[6]Tianming Hu and Sam Y. Sung. Detecting pattern-based outliers. Pattern Recognition Letters, Vol 24, pp3059-3068,Dec 2003.

[7] M. Pal (2005): Random forest classifier for remote sensing classification, International Journal of Remote Sensing, 26:1, 217-222.

[8] S. N. Pang and D. Kim and S. Y. Bang. Fraud detection using support vector machine ensemble. Pohang University of Science and Technology (POSTECH), 2001.

[9] Hand, David J and Blunt, Gordon and Kelly, Mark G and Adams, Niall M. Data Mining for Fun and Profit. Data Mining for Fun and Profit, Vol 15, pp 111-126, May 2000.

[10] Zhao, Q. and Bhowmick, S. S. Association Rule Mining: A Survey. Nanyang Technological University, Singapore, 2006.

[11] U Fayyad, R Uthurusamy. From Data Mining to Knowledge Discovery in Databases.ACM ,1996.

[12] Jiawei Han. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2006.

[13] Zengyou He and Xiaofei Xu and Joshua Zhexue Huang and Shengchun Deng. Mining class outliers: concepts, algorithms and applications in CRM. Expert Systems with Applications, Vol 27, pp 681-697, Nov 2004.

[14] Zengyou He and Xiaofei Xu and Shengchun Deng. Data Mining for Actionable Knowledge: A Survey. Computer Science, 2001.

[15] Jon T.S. Quah and M. Sriganesh. Real-time credit card fraud detection using computational intelligence. Expert Systems with Applications, 2007.

[16] Jian-Xin Pan and Wing-Kam Fung and Kai-Tai Fang. Multiple outlier detection in multivariate data using projection pursuit techniques. Journal of Statistical Planning and Inference, Vol 83, pp 153-167, 2000.

[17] Efstathios Kirkos and Charalambos Spathis and Yannis Manolopoulos. Data Mining techniques for the detection of fraudulent financial statements. Expert Systems with Applications, Vol 32, pp 995-1003, May 2007.

[18] Pang-Ning Tan, Michael Steinbach, Vipin Kumar. Introduction to Data Mining. Addison Wesley,2005.

[19] S. Viaene,G. Dedene and R.A. Derrig. Auto claim fraud detection using Bayesian learning neural networks. Expert Systems with Applications, Vol 29,pp 653-666, 2005.

[20] Clifton Phua, Vincent Lee, Kate Smith and Ross Gayler. A Comprehensive Survey of Data Mining-based Fraud Detection Research. School of Business Systems, Monash University, 2005.

[21] Kaustav Das and Jeff Schneider. Detecting anomalous records in categorical datasets. Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 220-229, New York 2007.

10. Anexo

```
# # Fraud detection usin SMOTE

# In[1]:


import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import os
import itertools

# Scale data
from sklearn.preprocessing import StandardScaler
# Over-sample data
from imblearn.over_sampling import SMOTE

# Import clasifiers
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import BernoulliNB

# Cross-Validation
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection  import cross_val_score

# Split data in test and train
from sklearn.model_selection import train_test_split

```

```
# Get metrics
from sklearn.metrics import accuracy_score, f1_score, precision_score,roc_auc_score,recall_score, confusion_matrix, make_scorer


# Lets start defining some procedures.

# In[2]:


def print_scores(y_test,y_pred,y_pred_prob):
    print('test-set confusion matrix:\n', confusion_matrix(y_test,y_pred))
    print("recall score: ", recall_score(y_test,y_pred))
    print("precision score: ", precision_score(y_test,y_pred))
    print("f1 score: ", f1_score(y_test,y_pred))
    print("accuracy score: ", accuracy_score(y_test,y_pred))
    print("ROC AUC: {}".format(roc_auc_score(y_test, y_pred_prob[:,1])))


# In[3]:


def predict_estimator(clf, X_test, X_train, y_train):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    y_pred_prob = clf.predict_proba(X_test)
    train_pred = clf.predict(X_train)
    return clf, y_pred, y_pred_prob, train_pred

```

```
# In[4]:


def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title, fontsize=14)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


# In[5]:
```

```
def plot_cm(y_train, train_pred, y_test,y_pred):
    # Ploting confusion matrix
    train_cm = confusion_matrix(y_train,train_pred)
    test_cm = confusion_matrix(y_test,y_pred)
    labels = ['No Fraud', 'Fraud']

    fig = plt.figure(figsize=(16,8))

    fig.add_subplot(221)
    plot_confusion_matrix(train_cm, labels, title="Train \n Confusion Matrix", cmap=plt.cm.Oranges)

    fig.add_subplot(222)
    plot_confusion_matrix(test_cm, labels, title="Test \n Confusion Matrix", cmap=plt.cm.Greens)


# In[6]:


def predict_new(clf,data):
    # Normalize and drop amount old feature
    data['normalized_amount']=StandardScaler().fit_transform(data['Amount'].values.reshape(-1,1))
    data=data.drop(['Amount'],axis=1)

    # Normalize and drop Time old feature
    data['normalized_time']=StandardScaler().fit_transform(data['Time'].values.reshape(-1,1))
    data=data.drop(['Time'],axis=1)
```

```
    # Get data and predict data
    X = data.values
    y_pred = clf.predict(X)
    y_pred_prob = clf.predict_proba(X)

    # Return fraud list from predicted data
    fraud = list()
    for i in range(len(y_pred)):
      if y_pred[i] == 1:
        fraud.append(i)
    return fraud,y_pred, y_pred_prob


# ## Get and evaluate the data

# In[7]:


# Get dataset from csv file
dat = pd.read_csv('entrenamiento.csv')
dat.head()


# In[8]:


# Count how many entry there are for every class
classes_count = pd.value_counts(dat['Class'])

print("{} Non-fraud example\n{} Fraud examples".format(classes_count[0], classes_count[1]))
```

```
# The classes are heavily skewed we need to solve this issue later.
print('No Frauds', round(dat['Class'].value_counts()[0]/len(dat) * 100,2), '% of the dataset')
print('Frauds', round(dat['Class'].value_counts()[1]/len(dat) * 100,2), '% of the dataset')


# In[9]:


# Show comparison between fraud and not fraud
classes_count.plot(kind = 'bar')
plt.xlabel('Classes')
plt.ylabel('Frequencies')
plt.title('Fraud Class distribution')


# In[10]:


# To see the actual distribution of data
#sns.pairplot(dat, hue = 'Class', vars = ['Time','V1', 'V2', 'V3', 'V15', 'V18','Amount'] )


# In[11]:


fig, ax = plt.subplots(1, 1, figsize=(18,4))

time_val = dat['Time'].values
```

```
sns.distplot(time_val, ax=ax, color='b')
ax.set_title('Distribution of Transaction Time', fontsize=14)
ax.set_xlim([min(time_val), max(time_val)])

plt.show()


# ## Estimating original data without any data processing

# In[12]:


# Copy dataset into original to test data before processing
original = dat


# In[13]:


# Store feature matrix in "X"
X = original.drop(["Class"], axis=1).values

# Store response vector in "y"
y = original["Class"].values


# In[14]:

```

```
# Splitting data of original dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.25)


# In[15]:


# Train Neural Networks (Multi-layer Perceptron classifier) estimator and make predictions with original data
ori_mlp, y_pred, y_pred_prob, train_pred = predict_estimator(MLPClassifier(solver='adam',
                                                                           hidden_layer_sizes=(130, 2),
                                                                           random_state=1),
                                                             X_test,
                                                             X_train,
                                                             y_train)


# In[16]:


# Print training set's confusion matrix
print('train-set confusion matrix:\n', confusion_matrix(y_train,train_pred))

# Print test set's confusion matrix
print_scores(y_test,y_pred,y_pred_prob)


# In[17]:

```

```
# Ploting confusion matrix
plot_cm(y_train, train_pred, y_test,y_pred)


# In[18]:


# Train Gaussian Naive bayes estimator and make predictions
ori_gbn, y_pred, y_pred_prob, train_pred = predict_estimator(BernoulliNB(alpha = 1.0),
                                                             X_test,
                                                             X_train,
                                                             y_train)


# In[19]:


# Print training set's confusion matrix
print('train-set confusion matrix:\n', confusion_matrix(y_train,train_pred))

# Print test set's confusion matrix
print_scores(y_test,y_pred,y_pred_prob)


# In[20]:


# Ploting confusion matrix
plot_cm(y_train, train_pred, y_test,y_pred)
```

```
# We can see that the models performs poorly in because they get raw data

# ## Processing data

# In[21]:


# Normalize and drop old amount feature
dat['normalized_amount']=StandardScaler().fit_transform(dat['Amount'].values.reshape(-1,1))
dat=dat.drop(['Amount'],axis=1)

# Normalize and drop old time feature
dat['normalized_time']=StandardScaler().fit_transform(dat['Time'].values.reshape(-1,1))
dat=dat.drop(['Time'],axis=1)


# In[22]:


# Store feature matrix in "X"
X = dat.drop(["Class"], axis=1).values

# Store response vector in "y"
y = dat["Class"].values


# In[23]:
```

```
# Splitting data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.25)


# In[24]:


# Apply over-sampling
X_smote, y_smote = SMOTE().fit_sample(X_train, y_train)


# ## Estimating processed data

# In[25]:


# Set the parameters by cross-validation
tuned_parameters = [{'hidden_layer_sizes': [(90,2),(100,2),(110,2)]}]

mlp =  GridSearchCV(MLPClassifier(solver='adam', activation = 'relu'),
                    tuned_parameters,
                    cv=5,
                    scoring='accuracy',
                    return_train_score=True,
                    n_jobs=-1
                   )


# In[26]:
```

```
# Get best hyper parameters for Neural Networks (Multi-layer Perceptron classifier) estimator
mlp.fit(X_smote, y_smote)
results = mlp.cv_results_


# In[27]:


mlp_best_parameters = mlp.best_params_
print(mlp_best_parameters)


# In[28]:


mlp_best_result = mlp.best_score_
print(mlp_best_result)


# In[29]:


# Set the parameters by cross-validation
tuned_parameters = [{'alpha': [1, 10, 20, 30]}]
```

```
bnb =  GridSearchCV(BernoulliNB(),
                    tuned_parameters,
                    cv=5,
                    scoring='accuracy',
                    return_train_score=True,
                    n_jobs=-1
                   )


# In[30]:


# Get best hyper parameters for bernoulli naive bayes estimator
bnb.fit(X_smote, y_smote)
resultsbnb = bnb.cv_results_


# In[31]:


bnb_best_parameters = bnb.best_params_
print(bnb_best_parameters)


# In[32]:
```

```
bnb_best_result = bnb.best_score_
print(bnb_best_result)


# In[ ]:


# Cross validation to get best accuracy mean from best Multi-layer Perceptron classifier estimator obtained from grid search
mlp_all_accuracies = cross_val_score(estimator=MLPClassifier(solver=mlp.best_estimator_.solver,
                                                             activation = mlp.best_estimator_.activation,
                                                             hidden_layer_sizes=mlp.best_estimator_.hidden_layer_sizes
                                                            ),
                                     X=X_smote,
                                     y=y_smote,
                                     cv=10,
                                     scoring='accuracy'
                                    )


# In[ ]:


print(mlp_all_accuracies.mean())


# In[ ]:
```

```
# Cross validation to get best accuracy mean from best  bernoulli naive bayes classifier estimator obtained from grid search
bnb_all_accuracies = cross_val_score(estimator=BernoulliNB(alpha=bnb.best_estimator_.alpha),
                                     X=X_smote,
                                     y=y_smote,
                                     cv=10
                                    )


# In[ ]:


print(bnb_all_accuracies.mean())


# In[37]:


y_pred_mlp = mlp.predict(X_test)
y_pred_prob_mlp = mlp.predict_proba(X_test)
train_pred_mlp = mlp.predict(X_smote)


# In[38]:


# Print training set's confusion matrix
print('train-set confusion matrix:\n', confusion_matrix(y_smote,train_pred_mlp))
```

```
# Print test set's confusion matrix
print_scores(y_test,y_pred_mlp,y_pred_prob_mlp)


# In[39]:


# Ploting confusion matrix
plot_cm(y_smote, train_pred_mlp, y_test,y_pred_mlp)


# In[40]:


y_pred_bnb = bnb.predict(X_test)
y_pred_prob_bnb = bnb.predict_proba(X_test)
train_pred_bnb = bnb.predict(X_smote)


# In[41]:


# Print training set's confusion matrix
print('train-set confusion matrix:\n', confusion_matrix(y_smote,train_pred_bnb))

# Print test set's confusion matrix
print_scores(y_test,y_pred_bnb,y_pred_prob_bnb)
```

```
# In[42]:


# Ploting confusion matrix
plot_cm(y_smote, train_pred_bnb, y_test,y_pred_bnb)


# ## Real testing

# To make a real tert to our model we have took a part from the original data into a new excell (this data its not in "entrenamiento.csv")
# In "validacion.csv" is the feature "Class" for validations.

# In[43]:


# Get dataset from csv file
data = pd.read_csv('evaluacion.csv')
data.head()


# In[ ]:


# Prediction caller
fraud, pred, pred_prob = predict_new(mlp.best_estimator_,data)
print(fraud)
print(pred)
print(pred_prob)
```