UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

DESARROLLO DE UN MÉTODO PARA DETECCIÓN DE FRAUDES DE PAGOS EN LÍNEA UTILIZANDO APRENDIZAJE AUTOMÁTICO

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor (es):

Bach. Huaman Casas Junior Aldair

ORCID: https://orcid.org/0000-0002-3645-7761

Bach. Serrato Vilcherres Fernando Jose

ORCID: https://orcid.org/0000-0001-9884-572X

Asesor:

Mg. Tuesta Monteza Victor Alexci

ORCID: https://orcid.org/0000-0002-5913-990X

Línea de Investigación:

Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú 2022

APROBACIÓN DEL JURADO

DESARROLLO DE UN MÉTODO PARA DETECCIÓN DE FRAUDES DE PAGOS EN LÍNEA UTILIZANDO APRENDIZAJE AUTOMÁTICO.

_____

Huaman Casas Junior Aldair

Autor

_____

Serrato Vilcherres Fernando José

Autor

_____

Mg. Tuesta Monteza Victor Alexci

Asesor

_____

Mg. Bravo Ruiz Jaime Arturo

Presidente de Jurado

_____

Mg. Samillan Ayala Alberto Enrique

Secretario de Jurado

_____

Mg. Diaz Vidarte Miguel Orlando

Vocal de Jurado

Dedicatorias

Esta investigación está dedicada a Dios, sin Él no hubiera sido posible. Por brindarme su infinito amor, salud y fortaleza para recuperarme. A mis padres, Paulo César Huaman Velásquez y Giuliana Violeta Casas La Torre, por el enorme esfuerzo que hicieron para que pueda estudiar esta maravillosa carrera, por ser mi soporte, por ser pieza transcendental en mi formación y por aleccionarme siempre para mejorar a nivel personal y profesional. Quiero que sepan, todo valió la pena.

Huaman Casas Junior Aldair.

Agradecimientos.

Agradezco a Dios padre, por brindarme vida y salubridad en estos tiempos de pandemia. Contar con eso, me hace sentir afortunado. Agradezco a mis padres y hermanos por confiar en mí, por su inconmensurable apoyo, por testimoniar mi desarrollo en todo aspecto. Asimismo, mi más sincero agradecimiento a los docentes de la Escuela Académico Profesional de Ingeniería de Sistemas de la prestigiosa Universidad Señor de Sipán, por compartir sus conocimientos, en especial al Mg. Tuesta Monteza Víctor Alexci y al Mg. Bravo Ruiz Jaime Arturo por su vocación para enseñar y su enorme intelecto. ¡Muchas Gracias!

Huaman Casas Junior Aldair.

Resumen

Durante el transcurso del tiempo el fraude ha ido evolucionando, el avance de las tecnologías ha provocado que el mundo cambie drásticamente y con ello los estafadores han innovado en sus técnicas y formas de fraude. Los pagos en línea han adquirido popularidad, siendo una de las más usadas en la actualidad, por lo que a los delincuentes le ha abierto nuevas formas de fraude. Uno de los problemas más grandes que existen en la industria del comercio electrónico es el fraude de pagos en línea, dada la fluctuación de tráfico de personas realizando compras y el desconocimiento existente en este tipo de casos. La detección de estos fraudes, es una ardua tarea para los ingenieros de seguridad informática, debido a la falta de datos de la vida real, los datos con distribución sesgada, y a las transacciones legítimas que se ven exactamente igual a transacciones fraudulentas.

Por lo expuesto, se optó por desarrollar un método aplicando Decision Tree y Support Vector Machines para mejorar los niveles de eficiencia buscando detectar los fraudes de pagos en línea, puesto que, los modelos de fraudes evolucionan y se deben implementar formas que puedan afrontar este problema. En esa línea, se consideró complementariamente un aspecto comparativo que ayude a determinar si existe un alto o bajo grado de eficiencia, con la misma data, pero usando otros algoritmos como: Random Forest, Naive Bayes y Logistic Regression. Donde se concluyó que la inclusión del objeto svc dentro del algoritmo DT, permite una mayor correlación en sus variables, así obtuvo una precisión de 99,42%; exhaustividad de 98,73%; consumo de cpu de 24,3%, pero con una diferencia de tiempo de respuesta de 3s frente a Random Forest. En mención al consumo de memoria, se optimizó la combinación de los algoritmos trabajando con parámetros direccionales.

Palabras Clave: Detección de fraude, pago en línea, aprendizaje automático, comercio electrónico, fraude bancario.

Abstract

Over the course of time, fraud has been evolving, the advancement of technology has caused the world to change drastically and with it fraudsters have innovated their techniques and forms of fraud. Online payments have become one of the most popular and widely used forms of payment today, which has opened up new forms of fraud for criminals. One of the biggest problems in the e-commerce industry is online payment fraud, given the fluctuating traffic of people making purchases and the lack of awareness of such cases. Detecting these frauds is an arduous task for computer security engineers, due to the lack of real-life data, skewed data distribution, and legitimate transactions that look exactly like fraudulent transactions.

Therefore, it was decided to develop a method applying Decision Tree and Support Vector Machines to improve efficiency levels in order to detect online payment fraud, since fraud models are evolving and ways to deal with this problem must be implemented. In this line, a comparative aspect was also considered to help determine whether there is a high or low degree of efficiency, with the same data, but using other algorithms such as Random Forest, Naive Bayes and Logistic Regression. It was concluded that the inclusion of the svc object within the DT algorithm allows a greater correlation in its variables, thus obtaining an accuracy of 99.42%; completeness of 98.73%; cpu consumption of 24.3%, but with a difference in response time of 3s compared to Random Forest. In terms of memory consumption, the combination of the algorithms was optimised by working with directional parameters.

Keywords: Fraud detection, online payment, machine learning, e-commerce, bank fraud.

Índice

I. INTRODUCCIÓN

1.1. Realidad Problemática.

Los fraudes con pagos en línea se han fraccionado en dos tipos: fraude interno con tarjeta y fraude externo, mientras que se ha elaborado una clasificación más amplia en tres categorías, es decir, fraudes habituales ligados con tarjetas (petición, robo, suplantación de cuenta), estafas de comerciantes (complicidad y triangulación) y estafas en Internet (clonación de sitios, o websites falsas). Se informa que el número total de pérdidas por fraude de bancos y compañías alrededor del mundo alcanzó más de USD 16 mil millones en 2014 con un incremento de casi USD 2.5 mil millones en el año anterior. Registró pérdidas, lo que significa que, cada USD 100 tiene 5,6 centavos que fue fraudulento. (Awoyemi, Adetunmbi, & Oluwadare, 2017)

En relación al tema de detección de fraudes mediante el aprendizaje automático, existen varios problemas identificados:

Según (Rafalo, 2017) es la falta de datos de la vida real debido a los problemas de confidencialidad y privacidad de los datos por parte de las empresas.

(Zojaji, Atani, & Monadjemi, 2016) mencionan los datos de desequilibrio o distribución sesgada de datos. Asimismo, explican que la superposición de datos es otro problema importante en la fase de preparación de datos de transacciones de tarjetas de crédito.

En su investigación (Zareapoor, KR, & Alam, 2012) manifiestan que el dilema radica en que las transacciones legítimas se ven exactamente como transacciones fraudulentas. Además, se encuentra la dificultad de tratar con datos categóricos.

(West & Bhattacharya, 2016) en su estudio indican que el inconveniente es la elección de los algoritmos de detección y la selección de características, puesto que, la mayoría de los algoritmos de aprendizaje automático toman más tiempo para fines de entrenamiento que para predecir.

Según Thennakon et al. (2019), mencionan la importancia de aprender las tecnologías afiliadas en la captación de fraudes con tarjetas de crédito y tener un conocimiento claro de los diferentes tipos de fraude con tarjetas de crédito. Conforme pasaba el tiempo, los modelos de fraude evolucionaron y se incluyeron nuevas formas del mismo, lo que lo llevó a convertirse en un área de sumo interés para los investigadores.

Por otro lado, (Raghavan & El Gayar, 2019), mencionan que, el aprendizaje automático es uno de los temas más mencionados de esta década y un subconjunto de la inteligencia artificial. Cada vez más organizaciones quieren invertir en aprendizaje automático para perfeccionar sus servicios. El aprendizaje automático es una mezcla de varios algoritmos informáticos y modelos estadísticos para permitir que la computadora ejecute tareas sin codificación rígida.

El modelo obtenido estaría aprendiendo de los “datos de entrenamiento”. Se pueden hacer predicciones o acciones a partir del conocimiento experiencial almacenado. Los modelos de aprendizaje profundo son parte de las técnicas de aprendizaje automático que implican redes neuronales artificiales. Una NN correctamente entrenada tendría el potencial de capturar relaciones únicas en todo el conjunto de datos.

Thennakoon et al. (2019) en su investigación argumentan que las transacciones fraudulentas pueden ocurrir de diversas formas y pueden clasificarse en diferentes categorías. Se centran en 4 escenarios principales de fraude en transacciones del mundo real.

Asimismo, indican que el fraude se puede evitar de dos formas vitales: prevención y detección. La prevención evita los ataques de los estafadores actuando como una capa de protección. Mientras que la detección ocurre una vez que la prevención ha fallado. Por lo tanto, la segunda ayuda a identificar y alertar tan pronto como se active una transacción fraudulenta.

Aparte, hacen referencia algunos tipos de fraude como: fraude por quiebra, por robo, por falsificación, fraude de aplicaciones, y fraude conductual. Los cuáles serán explicados en la sección de teorías, más adelante para su mayor comprensión.

Avanzando en nuestro razonamiento, se detallarán los datos estudiados y los resultados conseguidos: Los 2 tipos de fraude identificados en la mayoría de casos, son fraudes con tarjeta presente (CP) y fraudes con tarjeta no presente (CNP). Según el estudio, los datos se analizan de dos formas principales: datos categóricos y numéricos.

El conjunto de datos se creó combinando dos fuentes de datos; el archivo de registro de transacciones de fraude y el archivo de registro de todas las transacciones. Debido al acuerdo de divulgación confidencial celebrado entre el banco y los autores del documento, el atributo del número de tarjeta fue codificado. Al evaluar el conjunto de datos combinado, la forma de los datos estaba muy sesgada debido al desequilibrio en el número de transacciones legítimas y ocurrencias fraudulentas.

El archivo con los casos de fraude tenía 200 registros, mientras que el archivo de registro de transacciones tenía 917781 registros. Los atributos de las dos fuentes de datos son los siguientes: Atributos del registro de transacciones originales (Tabla 1). Atributos del registro de transacciones con fraude (Tabla 2)

Tabla 1

Atributos del registro de transacciones originales

Nombre del campo	**Descripción**
CARD_NO	Número de tarjeta hash
DATE	Fecha de la transacción
TIME	Hora de la transacción
TRANSACTION_AMOUNT	Cantidad de transacción
MERCHANT_NAME	Nombre del comerciante
MERCHANT_CITY	Ciudad registrada del comerciante
MERCHANT_COUNTRY	País registrado del comerciante
RESPONSE_CODE	Código de respuesta
	ISO relacionado con la transacción

MERCHANT_CATEGORY_CODE	Código de categoría del comerciante
APPROVED/ NOT APPROVED	Estado de la transacción

Nota: Tomado de Thennakoon et al. (2019)

Tabla 2

Atributos del registro de transacciones fraudulentas

Nombre del campo	**Descripción**
CARD_NO	Número de tarjeta hash
DATE	Fecha de la transacción
TIME	Hora de la transacción
SEQ	Un número de secuencia único que se proporcionó para los fraudes.
NATURE OF THE FRAUD	Naturaleza de los fraudes como tarjeta presente o no presente.
MCC	Código de categoría del comerciante.
AMOUNT OF FRAUD	Cantidad de transacción
REVERSAL	Relacionado con el banco: Inversión.

Nota: Tomado de Thennakoon et al. (2019)

Dicho estudio analiza 4 patrones de fraude diferentes. Para analizar cada patrón, se ha reflejado el proceso que se describe en la Figura 1. Se priorizaron en este trabajo 4 algoritmos de aprendizaje automático: modelo Naive Bayes, K-nearest Neighbor, Support Vector Machines y Logistic Regression.

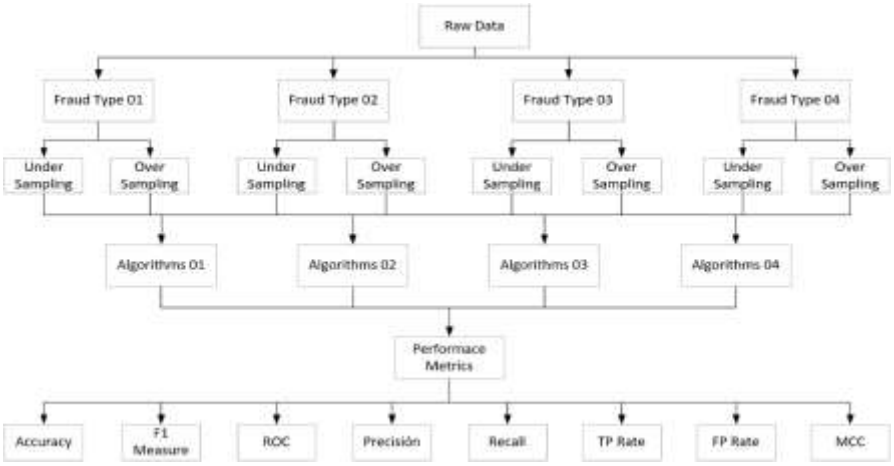


Figura 1. Selección del modelo. Fuente: Thennakoon et al. (2019)

Al seleccionar modelos de aprendizaje automático que pueden capturar cada fraude, se tuvieron en cuenta la precisión y el rendimiento de cada modelo. Los modelos óptimos se seleccionaron filtrándolos comparativamente con una matriz de desempeño apropiada (Tabla 3)

Tabla 3

Métricas de Rendimiento

Medida	**Fórmula**
Exactitud	TN + TP/TP + FP + FN + TN
Precisión	TP/TP + FP
Recall	TP / (TP + FP)
Tasa de verdaderos positivos	TP/TP + FN
Tasa de falsos positivos	FP/FP+TN
F1-medida	2 × (Precisión × Recall) / (Precisión + Recall)
ROC	La tasa TP frente a la tasa FP
MCC	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Nota: TP (Falso Positivo), TN (Falso Negativo), FP (Falso Positivo), FN (Falso Negativo). Tomado de Thennakoon et al. (2019)

Los siguientes gráficos muestran las tasas de precisión de los 4 tipos de fraude investigados cuando los clasificadores de Machine Learning se aplican a datos preprocesados y remuestreados.

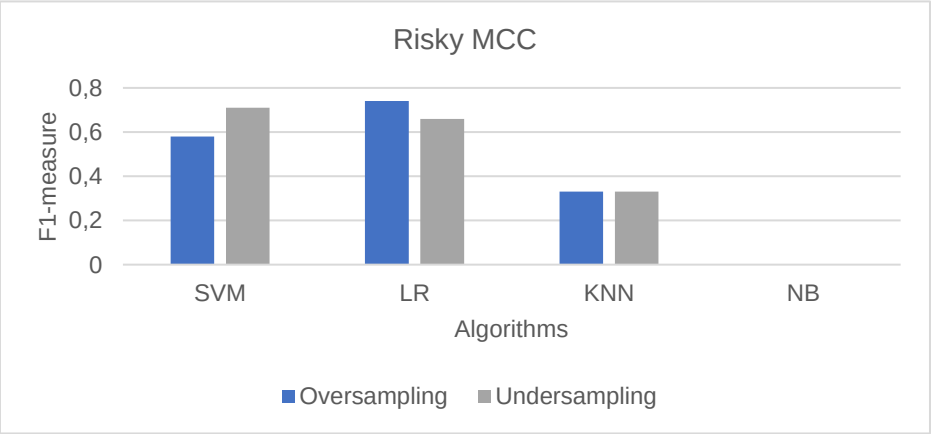


Figura 2. Resultados de transacciones de riesgo de MCC. Fuente: Thennakoon et al. *(2019)*

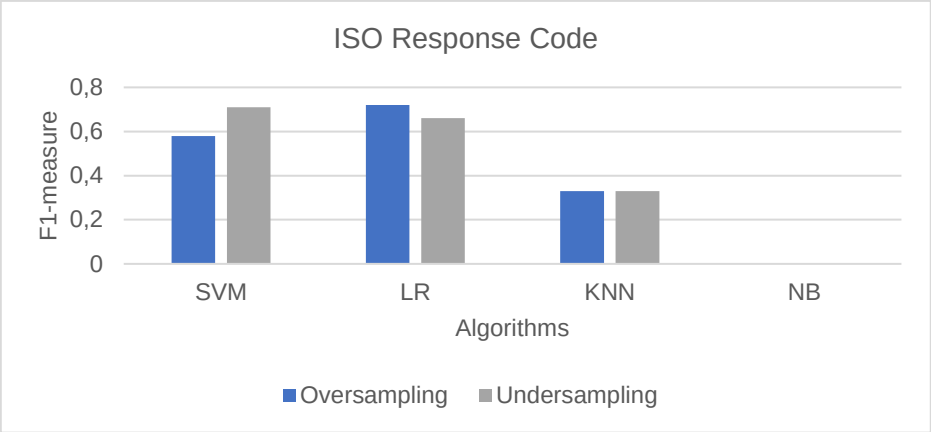


Figura 3. Resultados de transacciones con código de respuesta ISO. Fuente: Thennakoon et al. *(2019)*

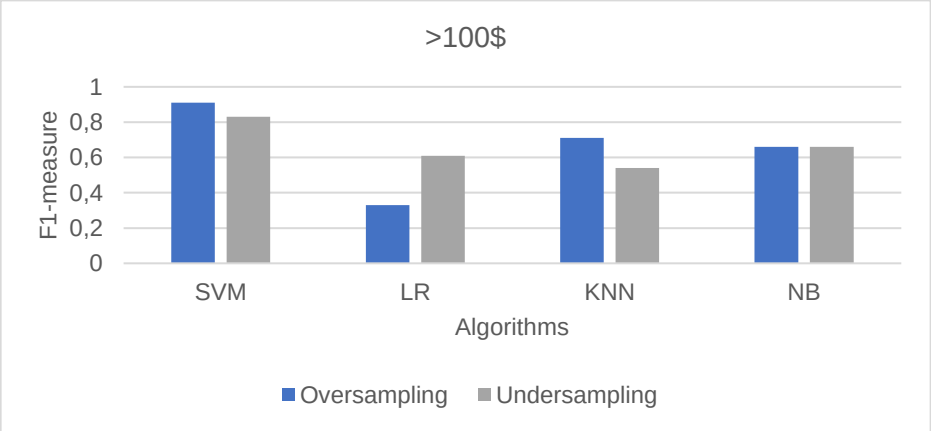


Figura 4. Resultados de transacciones >100$. Fuente: Thennakoon et al. *(2019)*

De esa forma, el sistema investigado es capaz de detectar fraudes en tiempo real. Se puede observar que se aplica en 4 escenarios de fraude diferentes haciendo uso de SVM, LR, KNN Y NB, los cuales presentaremos sus resultados más adelante. En referencia al fraude moderno se analiza la figura 5.

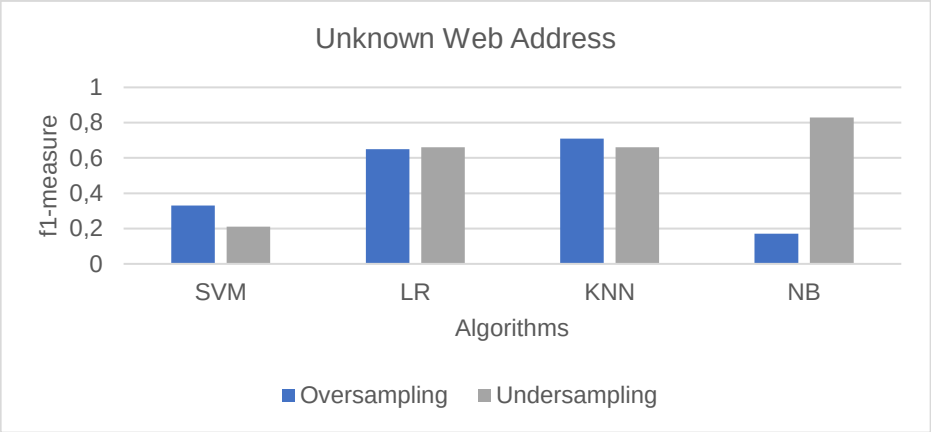


Figura 5. Resultados de transacciones en direcciones web desconocidas. Fuente: Thennakoon et al. *(2019)*

La detección en tiempo real de fraudes con tarjetas de crédito se puede afirmar como una de las principales contribuciones en este estudio. El sistema de detección de fraudes en tiempo real consta de 3 unidades principales: módulo API, modelos de detección de fraude y almacén de datos. Todos los componentes están involucrados en la detección del fraude simultáneamente.

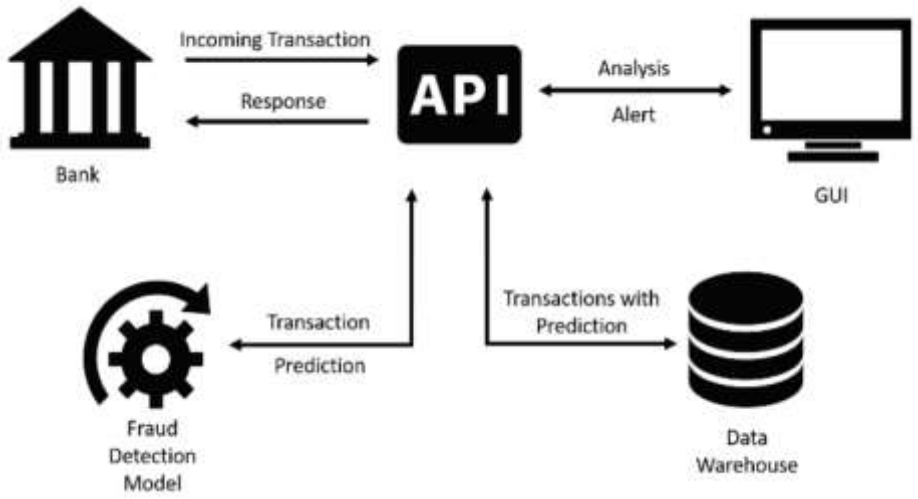


Figura 6. Diagrama del Sistema Propuesto en la Investigación para Detección de Fraudes Bancarios en Tiempo Real. Fuente: Thennakoon et al. *(2019)*

El módulo API es responsable de transferir transacciones en tiempo real entre el modelo de detección de fraude, la GUI y el almacén de datos. Se ha utilizado un almacén de datos para almacenar transacciones en vivo, los resultados previstos y otros datos importantes. El usuario puede interactuar con el sistema de detección de fraudes con GUI donde muestra las transacciones en tiempo real, alertas sobre fraudes y datos históricos sobre fraudes en una representación gráfica. La interacción del sistema se describe en la figura 6.

La detección de fraudes con tarjetas de crédito ha sido un área de investigación de gran interés para los investigadores durante años y será un área de investigación intrigante en el futuro próximo. Esto sucede principalmente debido al cambio continuo de patrones en los fraudes. En conclusión, respecto al análisis del documento, los modelos de aprendizaje automático que capturaron los cuatro patrones de fraude (Riesgo de MCC, dirección web desconocida, código de respuesta ISO, transacción superior a 100 $) con las tasas de precisión más altas son LR (74%), NB (83%), LR (72%) y SVM (91%) respectivamente.

Finalmente, se puede decir que el pago con tarjeta se está convirtiendo en el método más amplio de pago tanto a través de Internet como en persona, el fraude con tarjetas de crédito tiende a acelerarse rápidamente. Distinguir transacciones fraudulentas utilizando métodos convencionales para la identificación manual es tedioso e inexacto, de esta manera el crecimiento de gran cantidad de información ha hecho que las estrategias de métodos convencionales sean cada vez más irreales. Las organizaciones corporativas han cambiado a métodos inteligentes para solucionar este problema, los cuales están basados en el aprendizaje automático. Estos métodos no están restringidos, pero incluyen: modelo de Red Neuronal Artificial, modelo Naive Bayes, Regresión logística, Support Vector Machines, Árbol de decisiones, Vecino K-más cercano, etc.

1.2. Trabajos previos.

Kumar Rai & Kumar Dwivedi, (2020), en su investigación, Fraud Detection in Credit Card Data using Unsupervised Machine Learning Based Scheme, en India. El meollo aquí, es la falsificación de la tarjeta de crédito. La cual es un uso no aprobado de los datos de la tarjeta de un cliente para crear compras o descartar fondos del registro del titular de la tarjeta. A menudo el propietario de la tarjeta, el agente que emite la tarjeta e incluso el garante de la tarjeta no sean informados del fraude hasta que el registro se utilice para crear compras. Propusieron un esquema para la detección de fraudes en datos de tarjetas de crédito que utilizó una técnica de aprendizaje no supervisado basada en redes neuronales (NN). El preprocesamiento se realizó para limpiar los datos y extraer las características. Luego entrenaron el modelo y aplicaron las pruebas con varios modelos de aprendizaje no supervisados. Luego, el rendimiento se midió en varias métricas. Para identificar los valores atípicos, utilizamos cinco algoritmos (NN, AE, IF, LOF, K-Means) uno por uno y evaluaron el rendimiento. El método propuesto superó los enfoques existentes de Auto Encoder (AE), Local Outlier Factor (LOF), Isolation Forest (IF) y agrupación de K-Means. El método propuesto de detección de fraude basado en NN funcionó con una precisión del 99,87%, mientras que los métodos existentes AE, IF, LOF y K Means dieron una precisión de 97%, 98%, 98% y 99,75% respectivamente. Se pudo observar que el enfoque basado en redes neuronales funcionó mejor que los esquemas existentes. La conclusión experimental muestra que el punto de vista basado en la red neuronal proporciona un 99,98% de precisión, mientras que las precisiones de Auto Encoder, Isolation Forest, Local Outlier Factor y K Means clustering son 97%, 98%, 98% y 99,97% respectivamente.

Kumar & Iqbal, (2019), en su investigación, Credit Card Fraud Identification Using Machine Learning Approaches, en India. Este estudio analizó el problema de viabilidad de un sistema, en la detección de fraudes con tarjetas de crédito en función de la minería de valores atípicos, aplicó la minería de detección de valores atípicos basado en la suma de distancia y lo propuso. Debido a ello, el método se basó en entrenar los datos en MLP y se emplearon algoritmos de aprendizaje automático para generar resultados precisos. Se obtuvieron

resultados prometedores utilizando datos normalizados. En este documento se han utilizado tanto aprendizajes supervisados como no supervisados. Isolation Forest ha detectado 73 errores, mientras que Local Outlier Factor ha detectado 97 errores junto con SVM detectando 8516 errores. Isolation Forest presenta un 99,74% adicional de corrección que LOF del 99,65% y SVM de 70,09. Al comparar la precisión y recuperación de errores para 3 modelos, el bosque de aislamiento se desempeñó mucho mejor que el LOF, ya que veremos que la detección de casos de fraude es de alrededor del 27%, la tasa de detección de preocupaciones versus LOF de simplemente 2% y SVM de cero. Entonces, la técnica general de Isolation Forest funcionó mucho mejor para decidir los casos de fraude que son alrededor de 30.

Awoyemi, et al. (2019), en su investigación, Comparative Evaluation of Credit Card Fraud Detection Using Machine Learning Techniques, en India. Clasificar correctamente las transacciones de tarjetas de crédito como fraudulentas o no fraudulentas siempre implica un problema. Para evaluar estos modelos de aprendizaje automático, se consideró dos métodos diferentes: a) Precisión de clasificación: que es la razón del número de predicción correcta al número de muestra de entrada. Pero esto es muy efectivo solo si hay el mismo número de muestras en cada clase. b) Matriz de confusión: esto da una matriz como salida y describe el rendimiento completo del modelo. Se utilizaron cuatro medidas esenciales en evaluar los análisis, para ser específico Verdadero Positivo Relación (TPR), Relación negativa verdadera (TNR), Falso Relación positiva (FPR) y relación falsa negativa (FNR) clasifica la métrica individualmente. En esta investigación se desarrollan cuatro sistemas de algoritmos que se basan en (Regresión logística, SVM, Naive Bayes y KNN) El 80% de esa misma muestra se utiliza para la preparación para evaluar el diseño, mientras que el 20% se reserva para experimentación. Para evaluar la implementación de los clasificadores se utilizan especificidad, precisión, exactitud y sensibilidad. Los resultados muestran una precisión ideal para (Regresión logística, Naive Bayes, K-nearest-neighbor y clasificadores de Support Vector Machine) son 99,07%, 95,98%, 96,91% y 97,53% respectivamente. Los resultados relativos demuestran que la regresión logística tiene un rendimiento superior a otros

algoritmos. Cuando se probó en condiciones realistas, la regresión logística fue la más precisa para detectar el fraude con tarjetas de crédito. Con base en esta exploración, una organización de tarjetas de crédito debería considerar la ejecución de un algoritmo de regresión logística que investiga el tiempo de compra para distinguir si una transacción con tarjeta de crédito es un fraude.

Kumar et al. (2019), en su investigación Credit Card Fraud Detection Using Random Forest Algorithm, en India. El fraude con tarjetas de crédito es el principal problema en el mundo tecnológico actual, que tiene un problema masivo en las transacciones bancarias. Por esta razón, propusieron recopilar el total de datos de las tarjetas de crédito y realizar un análisis del conjunto de datos recopilado. Después del análisis, se requirió la limpieza de este. Generalmente, en cualquier conjunto de datos habrá muchos valores duplicados y nulos, por lo que para eliminar todos esos valores duplicados y nulos se requiere un proceso de limpieza. Luego, se tuvo que dividir en dos categorías como conjunto de datos capacitado y de datos de prueba para comparar y analizar. Después de dividió, con ello se aplicó el algoritmo de bosque aleatorio, donde este algoritmo otorgó la mejor precisión sobre las transacciones fraudulentas con tarjetas de crédito. En este artículo, el uso del algoritmo de bosque aleatorio en la detección de fraudes con tarjetas de crédito puede brindarle una precisión de aproximadamente 90 a 95%. Se concluyó que, el algoritmo de bosque aleatorio proporciona una mayor precisión en los resultados.

Raghavan & El Gayar, (2019), en su investigación, Fraud Detection using Machine Learning and Deep Learning, en Emiratos Árabes Unidos. Los fraudes son dinámicos y no tienen patrones, por lo que no son fáciles de identificar. El método se basó en comparar el rendimiento de SVM, KNN y Random Forest con métodos de Deep Learning como AutoEncoders, RBM, DBN y CNN. Dado que se trabajó con tres conjuntos de datos diferentes, podríamos ver hasta qué punto estas técnicas de aprendizaje automático son válidas. También se consideraron métodos de ajuste fino como PCA, reducción de características, limpieza de datos, hiperparámetros, etc. para mejorar el rendimiento del

clasificador. RBM y AE tienen altos falsos positivos (tasa de falsas alarmas) y, por lo tanto, funcionan mal con respecto a MCC y costo. Random Forest tiene buenos valores de AUC y MCC. CNN, SVM y KNN tienen el mejor desempeño en términos de MCC y AUC. Podemos ver que SVM tiene el menor costo de falla, mientras que Autoencoders y RBM tienen el más alto. El bosque aleatorio, aunque produce buenos resultados, es pobre en términos de costo. Los tres modelos con mejor rendimiento para este conjunto de datos son SVM, KNN y CNN. Para detectar fraudes, los mejores métodos con conjuntos de datos más grandes serían usar SVM, potencialmente combinados con CNN para obtener un rendimiento más confiable. Para los conjuntos de datos más pequeños, los enfoques de conjunto de SVM, Random Forest y KNN pueden proporcionar buenas mejoras. Las redes neuronales convolucionales (CNN) generalmente superan a otros métodos de aprendizaje profundo como los codificadores automáticos, RBM y DBN.

Zhou et al. (2019), en su investigación, A Model Based on Siamese Neural Network for Online Transaction Fraud Detection, en Hungría. Existen 2 problemáticas: El problema del desequilibrio de la muestra en el conjunto de datos de transacciones de red y el problema del escaso tiempo de transacción de la red. Se diseña un modelo de detección de fraude en transacciones de red basado en la red neuronal siamesa, que se basa en la red neuronal de convolución y la red de memoria a corto plazo. Usando CNN para caracterizar el aprendizaje y LSTM para hacer la estructura de memoria de la red. El modelo presentado en este artículo se verifica en datos reales de transacciones B2C, y su precisión y recuperación alcanzan alrededor del 95% y 96%, respectivamente. La estructura de la red neuronal siamés diseñada en este artículo tiene una cierta mejora en la precisión y la memoria. En el índice de precisión, el aumento fue de aproximadamente el 20% y en el índice de recuperación, el aumento fue de aproximadamente el 5%.

Kanika & Singla, (2020), en su investigación, A Survey of Deep Learning based Online Transactions Fraud Detection Systems, en Inglaterra. La disponibilidad de conjuntos de datos transaccionales reales para los investigadores es uno de

los principales desafíos puesto que la privacidad genera mayor preocupación en las organizaciones financieras. El método se basa en detectar el fraude en las transacciones bancarias en línea y evaluar la disponibilidad de datos enfocándose en técnicas de Deep learning. Las técnicas de aprendizaje profundo en los sistemas de detección de fraude de transacciones en línea no se han explorado por completo, ya que todavía hay mucha investigación por hacer. Es evidente a partir de la revisión que los conjuntos de datos que han sido utilizados por los investigadores no están disponibles públicamente y en su mayoría están desequilibrados. Además, en algunos trabajos de investigación, los resultados publicados no están completos.

Puh & Brkic, (2019), en su investigación, Detecting Credit Card Fraud Using Selected Machine Learning Algorithms, en Croacia. Indican que el fraude en tarjetas de crédito es un problema global, sobre todo porque se trabaja siempre con datos sesgados. Propusieron la comparación del rendimiento de tres algoritmos de aprendizaje automático: bosque aleatorio, máquina de vectores de soporte y regresión logística en la detección de fraude en datos reales que contienen transacciones con tarjetas de crédito. De estos algoritmos se utilizaron algunos para mitigar los tamaños de clase desequilibrados, el método de muestreo SM. El problema de los patrones de fraude en constante cambio se considera al emplear el aprendizaje incremental de algoritmos ML seleccionados en experimentos. AUC score revela que los tres algoritmos tienen un rendimiento similar, y que SVM tiene un resultado ligeramente más bajo que los otros dos algoritmos. Los tres algoritmos tienen mejor AUC score para modelo estático que para uno incremental. A partir de los resultados presentados en papel, es evidente que SVM muestra el pobre desempeño tanto en configuración como en lo incremental.

Awoyemi et al. (2017), en su investigación, Credit card fraud detection using machine learning techniques: A comparative analysis, en Nigeria. Los perfiles de comportamientos normales y fraudulentos cambian constantemente y, en segundo lugar, los conjuntos de datos están muy sesgados, ello genera un gran inconveniente. El método se basa en tres clasificadores en estudio; Ingenuo

Bayes, kNearest Neighbour y técnicas de regresión logística. Las diferentes etapas involucradas en la generación de los clasificadores incluyen; recopilación, preprocesamiento y análisis de datos, entrenamiento del algoritmo clasificador y pruebas (evaluación). La exactitud, la sensibilidad, la especificidad, la precisión, el coeficiente de correlación de Matthews (MCC) y la tasa de clasificación equilibrada se utilizan para evaluar el rendimiento de los tres clasificadores. La precisión de los clasificadores para la distribución del conjunto de datos original de 0,172: 99,828, las distribuciones de muestra de 10:90 y 34:66 se presentan en tablas explicadas posteriormente. Una observación de las tablas métricas muestra que hay una mejora significativa de la distribución del conjunto de datos muestreados de 10:90 a 34:66 para la precisión, sensibilidad, especificidad, coeficiente de correlación de Matthews y tasa de clasificación equilibrada de los clasificadores. Se concluyó que, tres clasificadores basados en diferentes técnicas de aprendizaje automático (Naïve Bayes, kNearest Neighbour y Regresión logística) son entrenados en la vida real del crédito.

Pradheepan & Neamat, (2019), en su artículo científico, su objetivo se centró en comparar las diferentes técnicas de Aprendizaje Máquina (ML) y Aprendizaje Profundo (DL). Se aplicaron los 3 mejores modelos de rendimiento a 3 conjuntos de datos, los cuales fueron europeos, australianos y alemanes. En su estudio el conjunto de datos europeo tuvo 492 instancias de fraude de un total de 284 807, el conjunto de datos australiano contiene 383 casos normales y 307 casos de fraude y el conjunto de datos alemán tuvo un total de 1000 instancias de los cuales 700 son instancias normales y 300 de fraude. Dicho estudio investigó la efectividad de diferentes modelos de ML y DL utilizando conjunto de datos con diferentes tamaños y complejidad. Se comparó el rendimiento de SVM, KNN y Random Forest con métodos de Deep Learning como AutoEncoders, RBM, DBN y CNN. Para K-nearest neighbor se utilizó la validación cruzada sobre el conjunto de data de entrenamiento para determinar el mejor valor de K-nearest neighbor para cada conjunto de datos. Para Support Vector Machines y Random Forest se utilizó un enfoque de búsqueda basado en cuadrículas con el objetivo de encontrar el mejor parámetro para cada modelo. Los mejores parámetros se

utilizan para evaluar los modelos con todos los datos. Detrás de Autoencoders es que podría reconstruir su entrada. Se entrenó a los codificadores automáticos, mientras se ejecuta el experimento con los datos de prueba, se darían errores de reconstrucción para cada una de las instancias. La expectativa es que las transacciones normales produzcan menos errores de reconstrucción, mientras que las transacciones / instancias de fraude obtengan valores más altos. Se establece un cierto valor de umbral, de modo que, si el error de reconstrucción está por encima de este umbral, la instancia / transacción en particular es fraudulenta. Se obtuvo que RBM y AE tienen altos falsos positivos (tasa de falsas alarmas) y, por lo tanto, funcionan mal con respecto a MCC y costo. Random Forest tiene buenos valores de AUC y MCC. CNN, SVM y KNN tienen el mejor desempeño en términos de MCC y AUC. Podemos ver que Support Vector Machines (SVM) tiene el menor costo de falla, mientras que Autoencoders y la Restricted Boltzmann Machine tienen el más alto. Random Forest, aunque produce buenos resultados, es pobre en términos de costo. Los tres modelos con mejor rendimiento para este conjunto de datos son SVM, KNN y CNN. Llegaron a la conclusión que, para detectar fraudes, los mejores métodos con conjuntos de datos más grandes serían usar SVM, potencialmente combinados con CNN para obtener un rendimiento más confiable. Para los conjuntos de datos más pequeños, los enfoques de conjunto de SVM, Random Forest y KNN pueden proporcionar buenas mejoras. Las redes neuronales convolucionales (CNN) generalmente superan a otros métodos de aprendizaje profundo como los codificadores automáticos, RBM y DBN.

1.3. Teorías relacionadas al tema.

Machine Learning

Aprendizaje Automático

Adepoju et al. (2019) indica que el Aprendizaje automático es una extensión de la Inteligencia Artificial, en la que las computadoras son capaces de analizar e identificar diseños dentro de conjunto de datos masivos y tener la capacidad de mejorar los ejemplos de forma natural sin la intervención humana.

El procedimiento de entrenamiento incluye comenzar con un algoritmo básico de aprendizaje automático que forma datos de entrenamiento para diseccionar la relación de diferentes componentes con una estima objetiva. El valor objetivo se otorga de manera inequívoca al algoritmo de aprendizaje automático en la etapa de entrenamiento. Una vez entrenado, el modelo podría utilizarse para anticipar otras instancias de datos para pronosticar valores objetivo desconocidos. (Adepoju et al., 2019).

Naive Bayes.

Este término fue introducido por primera vez por (John & Langley, 1995), quienes argumentan que es un método de aprendizaje automático supervisado que hace uso de un conjunto de data de entrenamiento con clases objetivas conocidas para predecir la clase de instancias futuras. Dicho de otro modo, un método NB supone que la "presencia o ausencia" de un atributo particular de un conjunto no se enfoca en la existencia o inexistencia de otros atributos en el mismo conjunto. Los experimentos con el conjunto de datos del mundo real han demostrado que el algoritmo NB funciona de manera comparable. Empero, esta técnica recibe el nombre de "ingenua" porque asume de esa manera la independencia de los atributos dados a la clase (Zareapoor, KR, & Alam, 2012).

Se fundamenta en el teorema de Bayes, que selecciona la decisión basada en la probabilidad más alta. Estimaciones de probabilidad bayesiana a partir de valores conocidos y probabilidades conocidas.

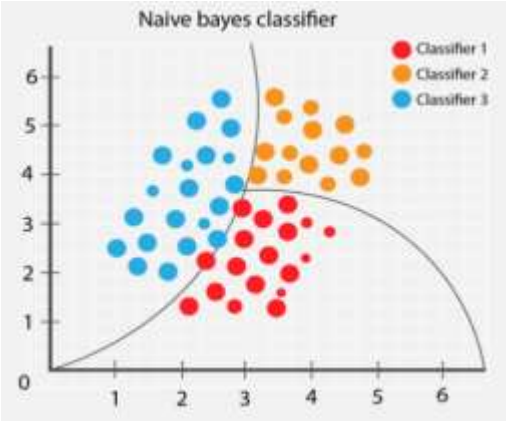


Figura 7. Clasificador Naive Bayes. Fuente: Towards Data Science Science (2019)

Red Neuronal ANN.

Es otra forma de método de detección de fraude bancario en línea que utiliza un marco informático flexible y resuelve problemas no lineales. El motivo principal detrás del desarrollo de ANN es simular la manera de aprendizaje del cerebro humano. Un perceptrón, representado como un nodo, es la unidad más pequeña de ANN. El sistema crea una estructura como el cerebro humano al conectar varios perceptrones en una red. Los nodos están en constante comunicación con otros nodos en las capas adyacentes creando una red bien conectada para la transferencia de información (Kataria & Tabrez, 2019).

El sistema toma información por medio de los nodos de entrada y los pasa a los nodos de salida a través de varias capas ocultas. Los nodos son de autoaprendizaje y pueden crear una salida más precisa al aprender y ajustar su comportamiento (Kataria & Tabrez, 2019).

La tarea de cada neurona se llama función de activación y se basa en su salida. Hay diferentes tipos de funciones de activación, como función escalonada, función lineal, función sigmoidea y función umbral. La función sigmoidea es la más popular para detectar fraudes bancarios en línea.

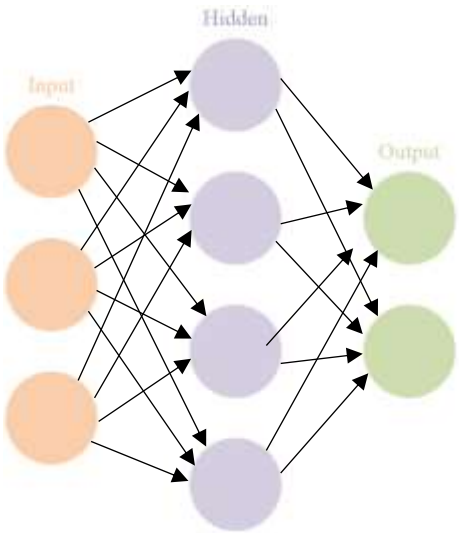


Figura 8. Estructura de la Red Neuronal. Fuente: Kataria & Tabrez *(2019)*

Árbol de decisiones.

Se pueden presentar diferentes resultados de situaciones únicas en un formato gráfico. El árbol de decisiones comienza con un nodo raíz que luego se divide en ramas separadas. Al conectar las ramas con otros nodos, se pueden calcular fácilmente varios resultados (Kataria & Tabrez, 2019).

Los nodos en un árbol de decisión representan la prueba y las ramas representan el posible resultado de un nodo. Un nodo hoja se etiqueta como clase. Un árbol de decisiones elimina la complejidad de un problema al dividir las partes complejas del problema en simples. Esta simplicidad permite que el árbol de decisiones marque una transacción como fraudulenta o legítima (Kataria & Tabrez, 2019).

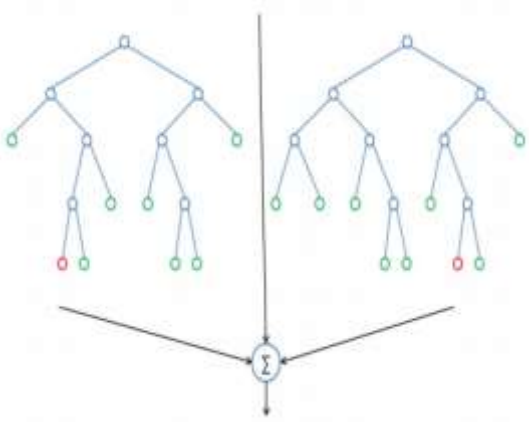


Figura 9. Representación de un Decision Tree. Fuente: Suresh et al. (2019)

K-Nearest Neighbor.

El algoritmo de vecino más cercano K es un algoritmo de clasificación que predice los atributos de un punto de información a otros puntos en función de su posición relativa. Su clasificación se basa en medidas de similitud como la distancia euclidiana, medida de la distancia de Manhattan. Se supone que el punto de datos en el conjunto de entrenamiento que tiene la distancia euclidiana más corta al punto de prueba tiene el mismo atributo desconocido que el punto de prueba. (Adepoju et al., 2019).

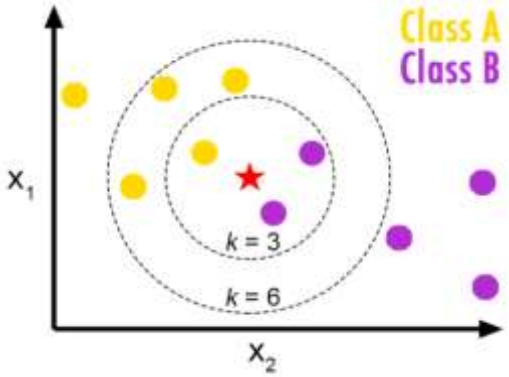


Figura 10. Procedimiento de clasificación empleado por Knearest Neighbors. Fuente: Adepoju et al. *(2019)*

Support Vector Machines.

Son ejemplares de algoritmos supervisados que se pueden emplear para solucionar problemas de clasificación o regresión. Una SVM determinará la mejor técnica de ajuste para clasificar la información. Una máquina de soporte (SVM) se define formalmente por un hiperplano separado como un clasificador discriminatorio. En otras palabras, dado el entrenamiento etiquetado, el algoritmo que categoriza nuevos ejemplos produce un hiperplano óptimo. El hiperplano subdivide un plano en espacios bidimensionales donde se halla a cada lado de cada clase.

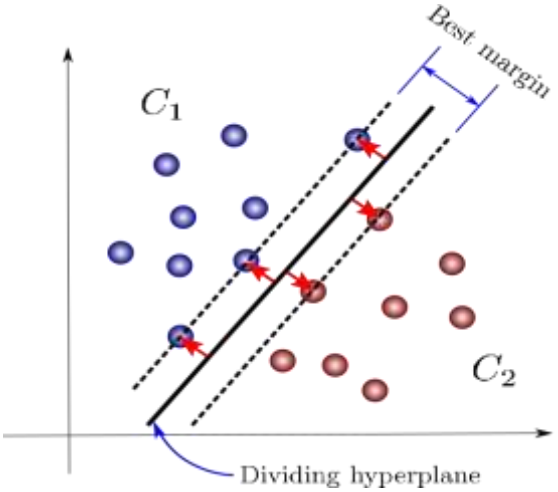


Figura 11. Representación y margen del conjunto de datos de Support Vector Machines. Fuente: Towards Data Science *(2019)*

Algoritmo Perceptrón Multicapa.

La red neuronal artificial es un algoritmo inspirado en la función de las neuronas humanas y muestra la mejor capacidad para aprender y enseñar. Para la detección de fraudes bancarios por Internet, la red neuronal perceptrón de tres capas también se puede utilizar para la detección de fraudes (Kataria & Tabrez, 2019).

La idea subyacente detrás de la red se basa en el funcionamiento de las funciones biológicas del sistema nervioso. El sistema biológico procesa nueva información y datos para aprender y crear nuevos conocimientos. Los elementos de procesamiento en el caso de este sistema son neuronas y la red es muy flexible ya que puede aprender, corregir errores y reaprender. (Kataria & Tabrez, 2019). El proceso de aprendizaje en estos ejemplos es comparativo. Por ejemplo, si hay nuevas entradas, el peso sináptico cambia y el sistema proporciona una respuesta correcta.

Red Neuronal Siamés.

Es una especie de estructura de red neuronal. El modelo de aprendizaje profundo basado en la red neuronal tiene las ventajas de abordar por completo la relación no lineal compleja arbitraria, fuerte robustez y tolerancia a fallas, alta velocidad para encontrar la capacidad de solución óptima, autoaprendizaje y capacidad de adaptación. Zhou et al. (2019).

Algoritmo Forestal Aleatorio.

También denominado Random Decision Forest (RFA), se utiliza para clasificación, regresión y otras tareas que se realizan mediante la construcción de varios árboles de decisión. Según Suresh et al. (2019), este algoritmo de bosque aleatorio se basa en el aprendizaje supervisado y la principal ventaja de este algoritmo es que se puede utilizar tanto para la clasificación como para la regresión. El algoritmo de bosque aleatorio le brinda una mejor precisión en comparación con todos los demás sistemas existentes y este es el algoritmo más comúnmente utilizado.

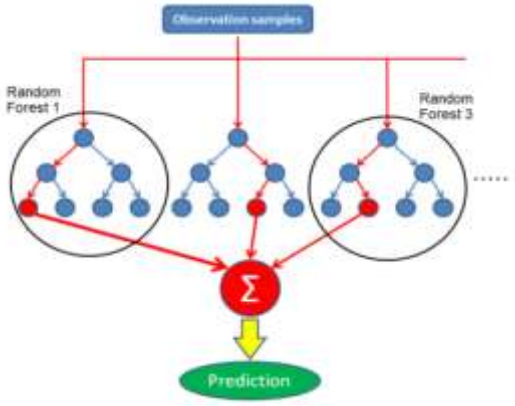


Figura 12. Selección Random Forest. Fuente: Takashi J (2015)

Modelo de Markov Oculto.

El sistema en este modelo de Markov está diseñado para comportarse como una cadena de Markov que tiene varias variables ocultas. Una propiedad distintiva de un HMM son los niveles de jerarquía que contienen una distribución de probabilidad integrada dual. Los importes de la transacción en un modelo oculto de Markov se determinan mediante la compra. El estado oculto está compuesto por los diferentes tipos de compras y el conjunto de líneas de negocio de los comerciantes. La idea básica detrás del modelo es detectar el fraude bancario en línea teniendo en cuenta los hábitos de gasto del usuario final (Kataria & Tabrez, 2019)

Redes Neuronales Convolucionales.

CNN es un tipo de sistema de aprendizaje profundo que es más adecuado para imágenes y gráficos visuales. Sin embargo, se puede utilizar para aplicar en datos textuales ya que es altamente adaptable. (Kanika & Singla, 2020)

Minería de Datos.

Es el proceso que se ejecuta a grandes conjuntos de datos con el fin de analizar, manipular datos e identificar tendencias, amalgamando diferentes campos de estudio como la estadística, la informática y el aprendizaje automático (Larose, 2014).

La clasificación es una función de la minería de datos que asigna elementos a categorías o clases de destino en una colección. El objetivo de la clasificación es predecir la clase objetivo con precisión en los datos de cada caso. Un trabajo de clasificación comienza con un conjunto de datos que conoce las asignaciones de clase que sirven como predictores de destino.

El tipo de problema de clasificación menos complejo es la clasificación binaria. En la clasificación binaria, el atributo objetivo tiene solo dos valores concebibles, por ejemplo, alto riesgo o, por otro lado, generalmente seguro para el fraude. Por tanto, el algoritmo más adecuado son los clasificadores binarios. (Larose, 2014)

Tipos de Fraude.

Phishing.

Conforme a la constante velocidad con la que evoluciona la red, del mismo modo la evolución persistente en la que está esté fenómeno llamada Phishing, es un factor que contribuye a no ofrecer una definición única de lo que verdaderamente es phishing.

El phishing es de las formas de estafa preferidas por indistintos delincuentes cibernéticos, lo que hace una de las formas de estafa más explotada actualmente para tratar de conseguir datos de suma importancia de los usuarios, como sus datos personales o información que puedas ser utilizada para generar algún lucro.

Según (ACENS, 2016) nos dice que: “tradicionalmente se le conoce al estafador con el alias de phisher, requiere de técnicas de ingeniería social, eludiendo ser una persona o empresa de una reputación intachable en la cual genera confianza y contactando a potenciales víctimas mediante un algún formato electrónico, usualmente un correo electrónico idéntico al que suelen remitir los contactos de confianza. De modo que, la persona que acepte estos correos no

tuviese cierto nivel de conocimiento acerca de ello, no logrará distinguir que se trata de un intento de fraude".

Pharming.

El pharming trata en la manipulación de direcciones DNS para engañar al usuario y cometer fraude. Según (INCIBE, 2015) nos indica que, es semejante al phishing. No obstante, lo que cambia es que la víctima no tiene que clickear un enlace fraudulento, más bien es redirigido a un sitio fraudulento al tratar de acceder al sitio legítimo. Esto se da debido a la manipulación de la información otorgada por los servidores DNS los cuales almacenan la información acerca de nombres de dominio y su correspondientes dirección IP.

Estafa nigeriana o Fraude 419.

Este fraude tiene muchos años de existir. Aun cuando resulta difícil caer en este engaño, este tipo de estafa reporta enormes pérdidas de dinero cada año. Se da principalmente a través de correos electrónicos no solicitados. Es más conocido bajo el alias de estafa nigeriana debido a que estos correos electrónicos provienen de Nigeria. Se le denomina fraude 419, puesto que, este fraude viola el artículo 419 del código penal nigeriano (Florentina, 2015).

La historia que más se cree verídica sobre la estafa se dio a saber en el año de 1990, en la cual el aparente beneficiario era parte de la realiza de Nigeria, o un burócrata del gobierno o gerente de una gran organización cuyos bienes monetarios está congelada por causas de la guerra interna, corrupción o problema políticos. Solo se tenía que hacer para trasladar esos recursos es una cuenta de banco segura para movilizar el dinero, y un depósito adelantado para pagar impuestos, comisiones o sobornos, todo era de acuerdo a la historia del mensaje (Perez, 2020).

Timo de la lotería.

Se da a través del envío copioso de correos electrónicos a variadas direcciones electrónicas, a la espera de que alguna victima caiga. A menudo estos correos

incluyen de manera no legítima logotipos del Organismo Nacional de Loterías y Apuestas del Estado (Florentina, 2015).

La Comisión Federal de Comercio, afirma que es más posible que, la mayor parte de las promociones de loterías que provienen del extranjero sean falsas. La mayoría de operadores de esta estafa ni siquiera compran los boletos de lotería prometidos a sus víctimas o hay otros que, si los compran, pero se dejan las ganancias para ellos. Así mismo, los estafadores que administran estas loterías usan las cuentas bancarias para realizar retiros de dinero no autorizados o utilizan las tarjetas de crédito para realizar cargos adicionales a las mismas (LA COMISIÓN FEDERAL DE COMERCIO, 2006).

1.4. Formulación del Problema.

¿El desarrollo de un método de detección basado en aprendizaje automático permitirá la detección de fraudes de pagos en línea?

1.5. Justificación e importancia del estudio.

Las entidades financieras tienen miles de millones de clientes y una concurrencia de miles de transacciones de usuarios por segundo. Teniendo como resultado bases de datos inmensas, frecuentemente repartidas en variados sistemas. La toma de decisiones se tiene que dar en tiempo real: un sistema de detección de fraude transaccional, es solo útil si logra identificar y detener una transacción fraudulenta de forma inmediata. Esto aplica rigurosas restricciones a la complejidad computacional de los algoritmos. Inclusive cuando la toma de decisión no es necesaria en tiempo real, los datos pueden tener cierto índice de cambio, por lo que es crucial extraer ciertas características que sinteticen el historial de los datos. El principal objetivo de los sistemas de detección es hallar patrones frecuentes de comportamiento sospechoso. Le prescripción de este dilema presenta un reto ya que los patrones evolucionan y son dinámicos a lo largo del tiempo para evitar los métodos de detección ya existentes. Los modelos son continuamente autentificados y acoplados a estas cambiantes reparticiones de patrones.

1.6. Hipótesis.

Con el desarrollo de un método aplicando Decision Tree y Support Vector Machines se podrá mejorar los niveles de eficiencia para detectar los fraudes de pagos en línea.

1.7. Objetivos.

1.7.1. Objetivo general.

Desarrollar un método para detección de fraudes de pagos en línea utilizando aprendizaje automático.

1.7.2. Objetivos específicos.

a) Analizar los algoritmos existentes.
b) Seleccionar algoritmos de machine learning.
c) Aplicar los mecanismos de detección en un dataset.
d) Evaluar mediante pruebas los resultados alcanzados.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación.

Tipo de Investigación

Tipo cuantitativa, puesto que, por su índole se deriva de criterio científico basado en temas de ciencias de la computación. Se precisa mencionar, que se utilizarán datos de tipo numérico para su posterior análisis o transformación en caso sea necesario. Los datos en mención estarán relacionados con las variables descritas en el apartado de Operacionalización. Asimismo, se busca medir valores como consumo de cpu, tiempo de respuesta, consumo de memoria, precisión, recall, f score y exactitud.

Diseño de Investigación

Diseño cuasi experimental. He aquí donde la variable independiente “Algoritmos de aprendizaje automático” se operará para detectar de una manera eficiente los fraudes de pagos en línea.

2.2. Población y muestra.

Población

Se hizo un análisis teórico de 12 algoritmos de aprendizaje automático, entre ellos algunos fusionados con técnicas de Deep Learning. Se precisa señalar que los mencionados, han sido empleados y ejecutados en diversas investigaciones previas, relacionadas al objeto de investigación y estos son:

Tabla 8

Población de Algoritmos de Machine Learning para la Investigación.

N°	Algoritmos de Machine Learning
1	K-Nearest Neighbor
2	Naive Bayes
3	Support Vector Machines
4	Artificial Neural Network
5	Siamese Neural Network
6	Decision Tree
7	Multilayer Perceptron Algorithm
8	Random Forest Algorithm
9	Convolucional Neural Network
10	Hidden Markov Model
11	Restricted Boltzmann machine
12	Neural Networks Long Short-Term memory

Nota: Elaboración propia.

Muestra

Es el grupo específico del que se recopilará datos. El porcentaje de la muestra es siempre menor que el de la población. Por conveniencia y guiados en sustento científico, tomando como referencia la métrica de precision, lo cual se denota en las tablas 17 y 21, para el desarrollo de la investigación se han tomado en cuenta dos algoritmos de Machine Learning: Decision Tree y Support Vector Machines. Por otro lado, para una comparación más amplia se usaron RF, LR, y NB, con el objeto de determinar si la hipótesis planteada es verdadera o no.

2.3. Variables, Operacionalización.

Variables	Dimensión	Indicador	Ítem	Técnica e instrumentos de recolección de datos
VARIABLE INDEPENDIENTE Algoritmos de aprendizaje automático	Consumo de recursos	Grado de consumo de CPU	$Ce = \sum_{j}^{n} \frac{Ce_j}{n}$	
		Grado de consumo de memoria	$c_m = \sum_{j}^{\eta} \frac{cm_j}{n}$	
		Promedio de tiempo de respuesta	$Tr = \sum_{j}^{n_f} \frac{tf_j - tf_i}{n}$	
VARIABLE DEPENDIENTE Detección de fraudes de pagos en línea	Rendimiento	Exactitud	$\mathrm{E} = \frac{\mathrm{VP} + \mathrm{VN}}{\mathrm{VP} + \mathrm{VN} + \mathrm{FP} + \mathrm{FN}}$	Registro electrónico Ficha de observación
		Precisión	$\mathrm{P} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}}$	
		Recall	$\mathrm{R} = \frac{VP}{VP + FN}$	
		F Score	$\mathrm{F} = 2 \cdot \frac{P \cdot E}{P + E}$	
		AUC	$AUC = \Sigma_{i(F+N)} \frac{(TPR_i + TPR_{i-1})(FPR_i + FPR_{\mathrm{i}-1})}{2}$	

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.

Técnicas de recolección de datos

Registro Electrónico

Será mediante esta técnica que se recopilará la información concerniente para el desarrollo de la investigación. En primer lugar, se identificó que el repositorio donde está almacenado el dataset sea seguro y fiable. En segundo lugar, se evaluó si cuenta con una licencia de uso, para que tenga mayor credibilidad y veracidad. En tercer lugar, se encontró que el dataset a utilizar había sido empleado en distintos proyectos de investigación, papers de revisión e informes de corte científico. Finalmente, se decidió trabajar con dicho registro de datos, en razón de que cumplía con todo lo mencionado anteriormente y porque es difícil conseguir información histórica sobre las transacciones en línea de las entidades bancarias debido a los acuerdos de confidencialidad y privacidad que celebran las empresas y sus clientes.

Instrumento de recolección de datos

Ficha de Observación

Conforme al proceso de desarrollo de la propuesta se gestionará cada cambio. Con el motivo de salvaguardar y generar un consolidado histórico de los pasos realizados para detectar si una transacción de pago en línea es legítima o fraudulenta según los algoritmos empleados.

2.5. Procedimiento de análisis de datos.

Cabe precisar que la obtención de datos de transacciones de pagos en línea de una entidad bancaria real, es muy difícil de lograr, dado que este tipo de empresas tienen un criterio de privacidad y confidencialidad respecto a sus cliente y movimientos bancarios. Es por ello que, la primera instancia para la implementación del método propuesto en la presente investigación, fue la adquisición del archivo “creditcard.csv”, el cual fue publicado por la unidad de investigación de Bélgica Machine Learning Group (MLG) y en escala del 1 a 10, tiene una usabilidad del 8.5. El dataset fue descargado del repositorio de Kaggle

en la dirección web: https://www.kaggle.com/mlg-ulb/creditcardfraud. Es importante mencionar que el mismo contiene 284,807 transacciones de tarjetas de crédito realizadas por personas de origen europeo durante un lapso de 2 días, en septiembre de 2013. La etiqueta denominada [Class] toma el valor 0 cuando la transacción es legítima y valor 1 cuando se trata de una transacción fraudulenta. Los pasos empleados se describen a continuación:

1. Se efectuó un amplio estudio teórico de los algoritmos de machine learning orientados a la detección de fraudes de pagos en línea; así seleccionar los mejores basados en términos de exactitud y precisión.

2. Se evaluó con ponderaciones los algoritmos previamente investigados en una matriz tomando factores de selección como codificación, complejidad, fiabilidad, simplicidad y en materia de hardware (consumo de recursos) para el algoritmo de machine learning.

3. Se escogió a los clasificadores en base a los resultados de la evaluación cuantitativa mencionada anteriormente. En este caso fueron: Support Vector Machines y Decision Tree, para realizar dicha combinación.

4. Se adquirió un dataset que cumpla con los requisitos de confiabilidad, veracidad, y validez; puesto que en esta investigación se prima estructuralmente un trabajo de calidad. Asimismo, se precisa denotar que, se encontraron 20 datasets, de los cuales sólo 3 cumplían con lo establecido. Por lo que, se seleccionó conforme el grado de usabilidad de los mismos.

5. Se concretó la adquisición del dataset “creditcard.csv” del grupo de investigación Machine Learning Group, el cual tiene una licencia.

6. Se hizo una limpieza del dataset y se prepararon las características V1, V2, hasta V28 con PCA. Recalcar que las únicas características que no se han transformado con PCA son Tiempo (número de segundos transcurridos

entre esta transacción y la primera transacción en el conjunto de datos) y Cantidad (Importe de la transacción).

7. Se trabajó el sistema en la herramienta de Python Jupyter Notebook y en la suite de Anaconda, se utilizó ambas debido a las sugerencias encontradas en los papers revisados.

8. Se importaron librerías como: Pandas para leer el dataset, Numpy para crear vectores o matrices, Matplotlib junto Seaborn para los gráficos, y finalmente, Plotfunctions para las funciones.

9. Se cargó el dataset con el método. read_csv, y se determinó la cantidad de registros que se trabajaran. Ello se hizo por filas y columnas, asimismo, se comprobó si faltan valores y tipos de datos de las columnas.

10. Se identificó la etiqueta de clasificación, en este caso “Class”.

11. Se estandarizaron los datos para que tengan media cero y varianza unitaria.

12. Se particionó el dataset, 70% (train_cf.csv) entrenamiento y 30% (test_cf.csv) prueba

13. Se aplicó el método SMOTE, dado que el dataset está desequilibrado y sirve para casos de sobremuestreo. El entrenamiento de conjuntos de datos desequilibrados con algoritmos de aprendizaje puede llevar a una clasificación errónea de la clase minoritaria. Por lo tanto, para compensar el desequilibrio, se usó SMOTE, diferenciándose así de otros artículos que trabajaron con ADASYN o calibración de características, este problema.

14. Se entrenaron complementariamente a la muestra, 03 algoritmos de machine learning; para tener un campo más amplio del comportamiento de otros algoritmos en un mismo escenario (dataset) y denotar si la fusión de SVM y DT era eficiente en la detección frente a RF, NB y LR.

15. Se obtuvieron indicadores de los modelos entrenados. Adicional a ello, se captó mediante código Python, el tiempo, CPU y memoria en uso.

16. Se realizaron pruebas para ver si los algoritmos de aprendizaje automático fueron entrenados correctamente.

17. Se produjeron indicadores de las pruebas realizadas.

18. Se obtuvo la matriz de confusión de los algoritmos: combinación de Support Vector Machine y Decision Tree, Logistic Regression, Random Forest, y Naive Bayes; para conocer cómo fue la detección de casos legítimos y fraudulentos.

19. Se ilustraron los resultados alcanzados en tablas, y gráficos.

20. Se discutieron los resultados. Por último, las fórmulas empleadas se describen a continuación:

Grado de consumo de CPU:

$$Ce = \sum_{j}^{n} \frac{Ce_j}{n}$$

Donde:

Ce: Es el grado de consumo de CPU

Ce_j: Es el grado de consumo de CPU en la prueba j

n: Es el total de pruebas

Grado de consumo de memoria:

$$Cm = \sum_{j}^{\eta} \frac{Cm_j}{n}$$

Donde:

Cm: Es el grado de consumo de memoria

Cm_j: Es el grado de consumo de memoria en la prueba j

n: Es el total de pruebas

Promedio de tiempo de respuesta:

$$Tr = \sum_{j}^{n_f} \frac{tf_j - tf_i}{n}$$

Donde:

Tr: Es el tiempo de respuesta

tf_j: Es el tiempo final de respuesta

tf_i: Es el tiempo inicial de respuesta

n: Es el total de pruebas

Exactitud

$$\mathrm{E} = \frac{\mathrm{VP} + \mathrm{VN}}{\mathrm{VP} + \mathrm{VN} + \mathrm{FP} + \mathrm{FN}}$$

Donde:

E: Es la exactitud

VP: Es el verdadero positivo

VN: Es el verdadero negativo

FP: Es el falso positivo

FN: Es el falso negativo

Precisión

$$\mathrm{P} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}}$$

Donde:

P: Es la precisión

VP: Es el verdadero positivos

FP: Es el falso positivo

Recall

$$\mathrm{R} = \frac{VP}{VP + FN}$$

Donde:

R: Es la recall o exhaustividad

VP: Es el verdadero positivo

FN: Es el falso negativo

F Score

$$\mathrm{F} = 2 \cdot \frac{P \cdot E}{P + E}$$

Donde:

F: Es el F -Score

P: Es la precisión

E: Es la exactitud

AUC

$$AUC = \Sigma_{i \in (P+N)} \frac{(TPR_i + TPR_{i-1})(FPR_i + FPR_{\mathrm{i}-1})}{2}$$

2.6. Criterios éticos.

Confiabilidad

Se deberá obtener la información de la presente investigación de manera legítima para así lograr cumplir lo que señala la ley N° 29733 y evitar daños a terceras personas en respecto a la protección de la información personal.

Conformabilidad

Se producirán resultados en base al fundamento y desarrollo de la investigación, los cuales deberán ser correctamente aprobados y legitimados por expertos en la materia. Dichas personas, como criterios tendrán la subjetividad y veracidad en las declaraciones dadas y expresadas.

Integridad

Se deberá poner en práctica el criterio ético de integridad, al tratarse de información, se tendrá que ser cuidadoso para no tergiversar ni manipular la

misma, lo mencionado servirá para mostrar datos o resultados verídicos y generar un ambiente de seguridad. Se citarán debidamente las fuentes bibliográficas y la proveniencia de las mismas, con el objeto de no incurrir en adulteración de la información recolectada, codificada y analizada en el proceso del estudio.

2.7. Criterios de Rigor Científico.

Originalidad

Se especificará de donde provienen las fuentes bibliográficas correctamente analizadas y consultadas, con el fin de ser ponderados en fundamento al porcentaje resultante del programa antiplagio.

Fiabilidad

Se respetarán los procedimientos y las políticas para efectuar un debido desarrollo e implementación tomando como base estándares, puesto que, se busca que la presente investigación cumpla con los objetivos planteados.

Validez

Se evaluarán los datos alcanzados durante el transcurso de la investigación, tomando como fuentes de validez, el criterio de especialistas e ingenieros calificados. Todo ello, será para fundamentar su veracidad y autenticidad.

Consistencia

Se abordará mediante la propuesta contenido fiable, consistente y certificado por las normas, políticas y estándares establecidos por la comunidad científica.

III. RESULTADOS.

3.1. Resultados en Tablas y Figuras.

Se muestran los resultados de la fusión de los algoritmos SVM y DT, así como la parte complementaria de algoritmos (RF, NB y LR) para fines comparativos y conocer el grado de predicción de los modelos para detectar casos de fraude. Se afianza lo denotado, en la dimensión rendimiento, tal como lo sustenta la matriz 2.3 de Variables, estos fueron Exactitud, Precision, Recall, FScore y AUC.

Tabla 09

Medidas de rendimiento de algoritmos de aprendizaje de máquina.

Algoritmo Clasificador	**Exactitud**	**Precisión**	**Recall**	**F-Score**	**AUC**
Support Vector Machine + Decision Tree	99,39%	99,42%	98,73%	99,18%	99,07%
Random Forest	97,1%	97,3%	97,0%	97,0%	97,0%
Logistic Regression	97,2%	98,2%	96,0%	97,0%	97,0%
Naive Bayes	95,8%	97,6%	94,0%	96,0%	95,7%

Nota: Las medidas de rendimiento fueron calculadas usando la base de datos de prueba. Elaboración propia.

En el siguiente acápite, se consideraron estadísticas gráficas para determinar a mejor detalle el efecto de los resultados alcanzados durante las pruebas de los algoritmos. Se trabajó las dimensiones e indicadores. Dimensión Rendimiento con: exactitud, precisión, recall, f1 score y auc. Dimensión Consumo de Recursos con: grado de consumo de CPU, grado de consumo de memoria y promedio de tiempo de respuesta del algoritmo. Dado los resultados, Exactitud de los algoritmos, se genera la figura 13.

Figura 13. Evaluación de los modelos de ML basado en Exactitud.
Fuente: Elaboración propia.

Los datos muestran que, en términos de Exactitud, existe una mejor predicción en la fusión de algoritmos de Support Vector Machines y Decision Tree (99,39) y frente a Logistic Regression (97,2). Random Forest (97,1), y Naive Bayes (95,8). Se observó que el modelo combinado obtuvo el más alto porcentaje.

Dado los resultados, Precision de los algoritmos, se genera la figura 14.

Figura 14. Evaluación de los modelos de ML basado en Precisión.
Fuente: Elaboración propia.

Los datos muestran que, en términos de Precisión, existe una mejor predicción en la fusión de algoritmos de Support Vector Machines y Decision Tree (99,42). Y los clasificadores comparados fueron Logistic Regression (98,2), Naive Bayes (97,6), y Random Forest (97,3). Se observó que el modelo propuesto alcanzó una mejor performance en cuanto a precisión.

Dado los resultados, Recall de los algoritmos, se genera la figura 15.

Figura 15. Evaluación de los modelos de ML basado en Recall. Fuente: Elaboración propia.

Los datos muestran que, en términos de Recall, existe una mejor predicción en la fusión de algoritmos de Support Vector Machines y Decision Tree (98,73), frente a Random Forest (97,0), Logistic Regression (96,0) y Naive Bayes (94,0). Se observó que el modelo propuesto alcanzó una alta ponderación en aspectos de exhaustividad.

Dado los resultados F Score de los algoritmos, se genera la figura 16.

Figura 16. Evaluación de los modelos de machine learning basado en F Score. Fuente: Elaboración propia.

Los datos muestran que, en términos de F Score, existe una mejor predicción de la fusión de algoritmos de Support Vector Machines y Decision Tree (99,18). Luego de los cuales, están los modelos Random Forest (97,0), Logistic Regression (97,1) y Naive Bayes (96,0). Dado los resultados, AUC Score de los algoritmos, se genera la figura 17.

Figura 17. Evaluación de los modelos de machine learning basado en AUC Score. Fuente: Elaboración propia.

Los datos muestran que, en términos de AUC Score, el mejor porcentaje fue la combinación de los algoritmos de Support Vector Machines y Decision Tree (99,07). A continuación, se muestran los resultados de los algoritmos de **M**achine **L**earning. Se señalan los indicadores de la dimensión consumo de recursos, como lo sustenta la tabla de operacionalización 2.3, estos fueron grado de consumo de CPU, grado de consumo de memoria y promedio de tiempo de respuesta.

Tabla 10

Medidas de consumo de recursos de algoritmos de aprendizaje de máquina.

Algoritmo Clasificador	CPU	Memoria	Tiempo de Respuesta
Support Vector Machine + Decision Tree	24,3	68,5%	0,0112
Random Forest	26,1	68,7%	0,0109
Logistic Regression	25,9	69,2%	0,0173
Naive Bayes	28,9	68,4%	0,0172

Nota: Las medidas de consumo de recursos fueron calculadas usando la base de datos de prueba. Elaboración propia.

Dado los resultados, uso CPU de los algoritmos, se genera la figura 18.

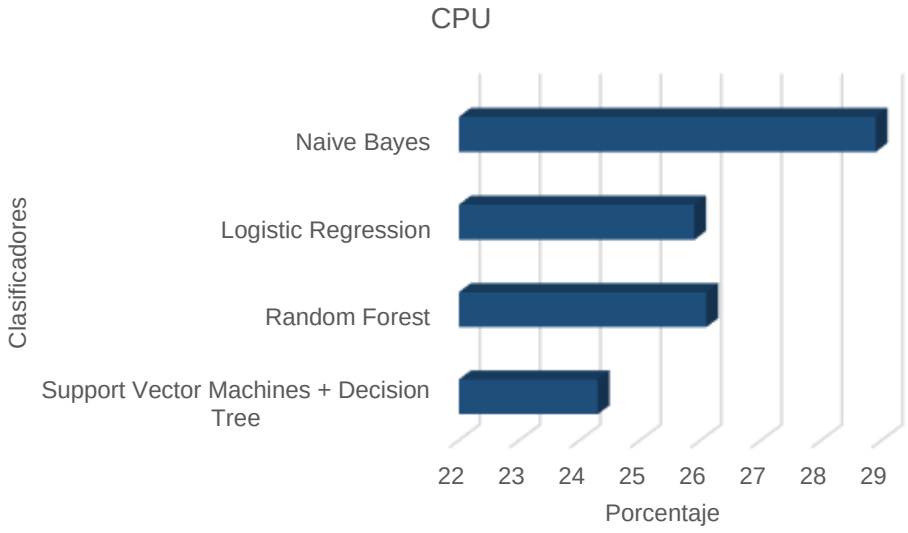


Figura 18. Evaluación de los modelos de machine learning basado en grado de consumo de CPU. Fuente: Elaboración propia.

Los datos muestran que, en términos de CPU existe un menor consumo en la fusión de algoritmos de Support Vector Machines y Decision Tree (24,3); respecto a Random Forest (26,1), Logistic Regression (25,9) y Naive Bayes (28,9). Asimismo, se debe mencionar que las pruebas se realizaron en 2 tipos de PC´s diferentes, empero sólo se añadió los valores sacados del computador con las siguientes características (8GB ram, 1000 GB rom, SSD 256, 4 núcleos, 10th generación, marca ACER, i3) debido a que el otro dispositivo era muy limitado para procesar la información y las respuestas tardaban demasiado.

Dado los resultados del uso de Memoria de los algoritmos, se genera la figura 19.

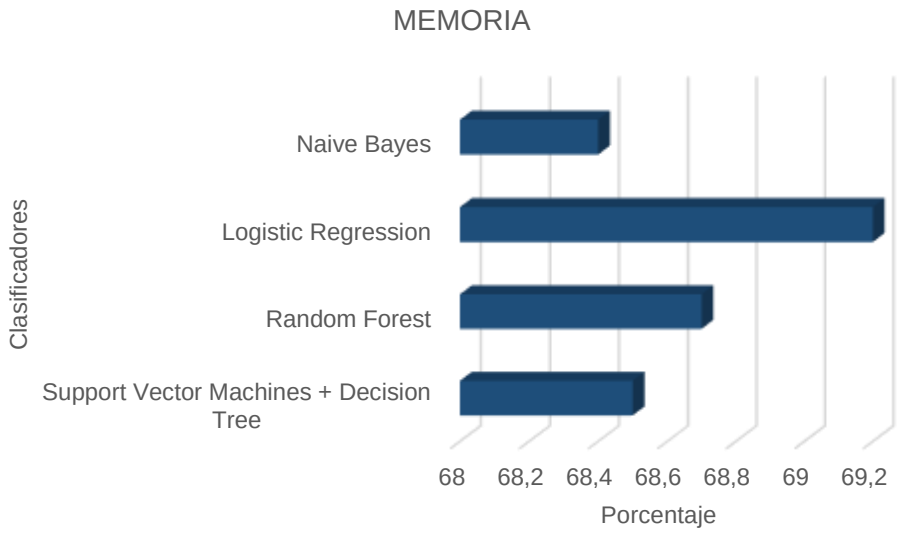


Figura 19. Evaluación de los modelos de machine learning basado en grado de consumo de Memoria. Fuente: Elaboración propia.

Los datos muestran que, en términos de MEMORIA existe un decremento del consumo cuando refiere a la fusión propuesta de SVM y DT (68,5%) en contraposición con Logistic Regression (69,2%), Random Forest (68,7%) y Naive Bayes (68,4%). Dado los resultados del Tiempo de Ejecución de los algoritmos, se genera la figura 20. Un acápite importante dado que vemos que la optimización en consumo de memoria está presente.

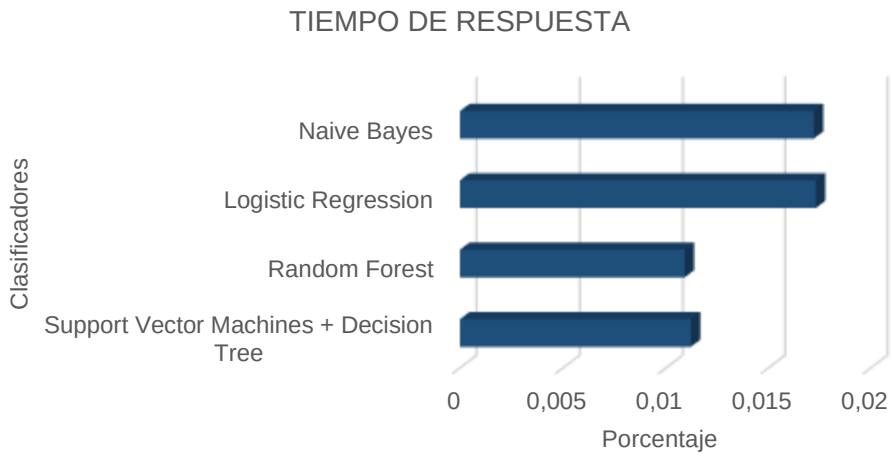


Figura 20. Evaluación de los modelos de machine learning basado en grado de consumo de Memoria. Fuente: Elaboración propia.

Los datos muestran que, en términos de TIEMPO existe un mayor consumo en el algoritmo Logistic Regression (0,0173), y Naive Bayes (0,0172); y en segundo escalón están la fusión de SVM y DT (0,0112) y Random Forest (0,0109) respectivamente. Es decir, un bajo consumo por parte de SVM, DT, y RF.

Habiendo expuesto los resultados anteriores de los indicadores, según la dimensión donde pertenecen. El último punto que se consideró necesario detallar en este apartado, fue la matriz de confusión de cada algoritmo ejecutado en el programa. Por ello, en la tabla 11, se observa la matriz del algoritmo Support Vector Machines + Decision Tree

Tabla 11

Matriz de Confusión del clasificador Decision Tree + Support Vector Machines en el dataset de prueba.

		Clasificación	
		Legítimo	Fraudulento
Reales	Legítimo	626	13
	Fraudulento	24	537

Fuente: Elaboración Propia.

De un total de 1200 registros en el conjunto de datos, la fusión de Decision Tree y Support Vector Machines detectó 650 transacciones como legítimas y 550 como fraudulentas. Por lo tanto, de las transacciones legítimas detectadas se logró predecir 626 transacciones correctamente, y 24 transacciones fraudulentas debido a que el modelo los predijo como transacciones legítimas cuando en realidad eran fraudulentas.

Por otro lado, de las transacciones fraudulentas detectadas se logró predecir 537 registros correctamente, y 13 consultas con grado de error debido a que el modelo los predijo como fraudulentas, pero en realidad eran legítimas.

Tabla 12

Matriz de Confusión del clasificador Random Forest en el dataset de prueba.

		Clasificación	
		Legítimo	Fraudulento
Reales	Legítimo	632	16
	Fraudulento	19	542

Nota: Elaboración Propia.

De un total de 1200 registros en el conjunto de datos, el algoritmo Random Forest detectó 651 transacciones como legítimas y 558 como fraudulentas.

Por lo tanto, de las transacciones legítimas detectadas se logró predecir 632 transacciones correctamente, y 19 transacciones fraudulentas debido a que el modelo los predijo como transacciones legítimas cuando en realidad eran fraudulentas.

Por otro lado, de las transacciones fraudulentas detectadas se logró predecir 542 registros correctamente, y 16 consultas con grado de error debido a que el modelo los predijo como fraudulentas, pero en realidad eran legítimas.

Tabla 13

Matriz de Confusión del clasificador Logistic Regression en el dataset de prueba.

		Clasificación	
		Legítimo	Fraudulento
Reales	Legítimo	630	9
	Fraudulento	25	536

Nota: Elaboración Propia.

De un total de 1200 registros en el conjunto de datos, el algoritmo Logistic Regression detectó 655 transacciones como legítimas y 545 como fraudulentas.

Por lo tanto, de las transacciones legítimas detectadas se logró predecir 630 transacciones correctamente, y 25 transacciones fraudulentas debido a que el modelo los predijo como transacciones legítimas cuando en realidad eran fraudulentas.

Por otro lado, de las transacciones fraudulentas detectadas se logró predecir 536 registros correctamente, y 9 consultas con grado de error debido a que el modelo los predijo como fraudulentas, pero en realidad eran legítimas.

Tabla 14

Matriz de Confusión del clasificador Naive Bayes en el dataset de prueba.

		Clasificación	
		Legítimo	Fraudulento
Reales	Legítimo	625	14
	Fraudulento	35	526

Nota: Elaboración Propia.

De un total de 1200 registros en el conjunto de datos, el algoritmo Naive Bayes detectó 660 transacciones como legítimas y 540 como fraudulentas.

Por lo tanto, de las transacciones legítimas detectadas se logró predecir 625 transacciones correctamente, y 35 transacciones fraudulentas debido a que el modelo los predijo como transacciones legítimas cuando en realidad eran fraudulentas.

Por otro lado, de las transacciones fraudulentas detectadas se logró predecir 526 registros correctamente, y 14 consultas con grado de error debido a que el modelo los predijo como fraudulentas, pero en realidad eran legítimas.

3.2. Discusión de resultados.

En base a los resultados alcanzados, se observó que, en el indicador exactitud, el mejor porcentaje lo obtuvo la fusión de Support Vector Machines y Decision Tree con 99,39%, seguido de Logistic Regression con 97,2% Después en un segundo nivel, se situaron los algoritmos de Random Forest con 97,1%. En última instancia estuvo el algoritmo Naive Bayes con 95,8% siendo este el de menor puntuación dentro de todos los clasificadores. Mencionar que, el indicador es la razón entre el número de predicciones de fraudes correctas y el número de total de predicciones.

Asimismo, se observó que, en el indicador precision, el mejor porcentaje lo obtuvo el modelo propuesto donde se combina Support Vector Machines y Decision Tree con 99,42%. En relación a ello, se mostraron las gráficas en los resultados conseguidos. Aunque muchas veces, se confunde este término con exactitud, se tiene que precisar que no es lo mismo, dado que la precision es la relación de todos los elementos clasificados correctamente de entre todos los que hemos clasificado.

Por otro lado, se observó que, en el indicador recall, el cual es la fracción de las instancias positivas que el clasificador identifica correctamente como positiva, obtuvieron porcentajes como 98,73%, 97%, 96%, y 94%. Siendo así, la distribución, el primer caso con los modelos Support Vector Machines fusionado con Decision Tree, el segundo caso fue Random Forest, tercero Logistic Regression y finalmente el clasificador Naive Bayes. Si visualizamos los valores de precision y recall, veremos que el margen es mínimo, lo cual nos demostró un equilibrio de los valores obtenidos de los algoritmos de machine learning.

De igual manera, se observó que, en el indicador f1 score, el cual se consideró debido que es la representación del promedio de las métricas precision y recall, conocido como media armónica, mencionar que este indicador toma en cuenta los falsos positivos y los falsos negativos. Dicho esto, el mejor porcentaje lo obtuvo la combinación de Support Vector Machines y Decision Tree con 99,18%; en segunda instancia se encontraron 3 algoritmos con la misma ponderación, estos son Random Forest con 97%, Logistic Regression con 97,1% y Naive Bayes con 96%. Adicional a ello, se observó que, en el indicador auc score, el mejor porcentaje lo obtuvo la fusión de SVM y DT con 99,07%.

Finalmente, en razón de la mejor eficiencia de los modelos, se encontraron que todos los algoritmos en su media armónica no superaron la precision de los mismos, lo cual se evidencia en la Tabla 09. Empero, se mostró en base a los resultados conseguidos en la sección 3.1, que los algoritmos con mejores valores fue la combinación de Support Vector Machines y Decision Tree propuesta. Para concluir, esta discusión se trató también el aspecto de consumo de cpu y memoria, donde SVM + DT obtuvieron de (CPU 24,3; MEMORIA 68,5) respectivamente. Siendo los valores más bajos frente a Random Forest (26,1; 68,7), Logistic Regression (25,9; 69,2), y Naive Bayes (28,4; 68,4).

3.3. Aporte práctico.

Gráfica del método

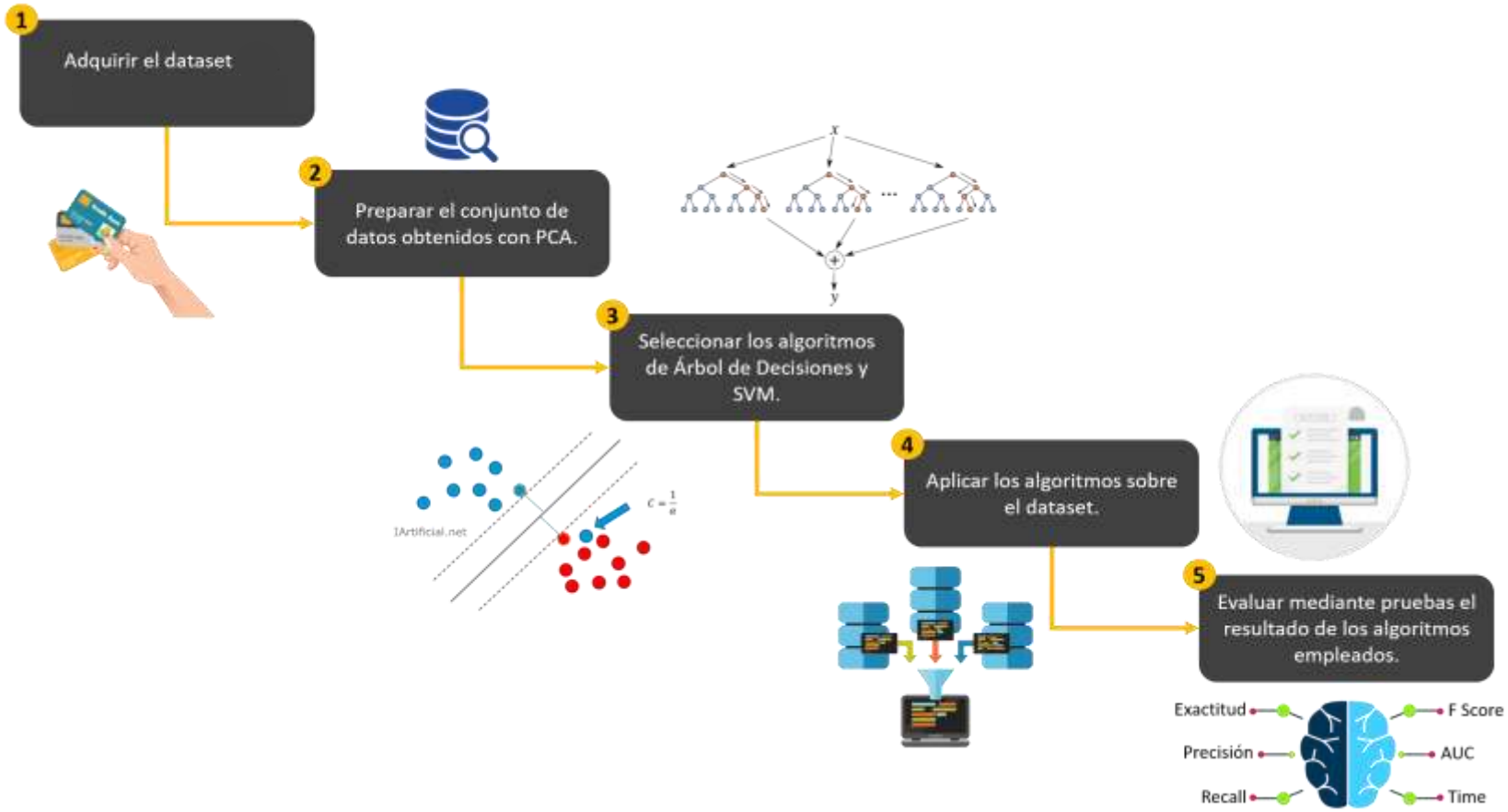


Figura 21. Representación del Método Propuesto a nivel general. Fuente: Elaboración propia.

Explicación del Método

1. **Adquirir el dataset**

   El inicio de esta propuesta empezó con la adquisición del archivo "creditcard.csv", el cual fue realizado por Machine Learning Group - ULB. Los datos fueron descargados del repositorio de Kaggle en su página web https://www.kaggle.com/mlg-ulb/creditcardfraud. El dataset contiene el registro de 284.807 transacciones con tarjetas de crédito realizadas por titulares de tarjetas europeos durante el lapso de 2 días en septiembre del 2013. La característica '**Clase**' es la variable de respuesta y toma el valor **1** en caso de fraude y **0** en caso contrario.

2. **Preparar el conjunto de datos obtenidos con PCA.**

   La clase positiva (fraudes) representa el 0,172% (492 fraudes de 284.807 transacciones) de todas las transacciones. Las características V1, V2, hasta V28 son los componentes primordiales obtenidos con PCA, las únicas características que no se han transformado con PCA son Tiempo (número de segundos transcurridos entre esta y la primera transacción en el conjunto de datos) y Cantidad (Importe de la transacción). La clase positiva (fraudes) simboliza el 0,172% (492 fraudes de 284.807 transacciones) de todas las transacciones.

3. **Seleccionar los algoritmos de Árbol de decisiones y Support Vector Machines.**

   Posterior a realizar una voluble y extensa revisión literaria sobre aquellos algoritmos de Machine Learning, ya sea híbridos o no, pero que han sido aplicados al tema sobre detección de fraude de pagos en línea, se optó por conveniencia y ponderación con base científica a elegir los algoritmos de DT y SVM, para obtener mejores niveles de eficiencia para la detección del mismo. Dicha explicación acerca del análisis se encuentra descrita en la sección del primer objetivo y referido a la selección, se observa el proceso en el segundo objetivo.

4. **Aplicar los algoritmos sobre el dataset.**

   El lenguaje de programación que se utilizó es Python. El proceso empezó importando librerías especializadas en el manejo y análisis de estructuras de datos (Pandas), en el cálculo numérico de vectores (NumPy), y en la generación de gráficos (Matplotlib). Asimismo, se requirieron funciones gráficas (plot_functions), un kit de herramientas (Scikits) y por supuesto las configuraciones básicas. Consecutivamente, se procedió a cargar los datos del dataset “creditcard”, donde se determinó la cantidad de registros a usar, por ello se precisa que existen 284.807 transacciones y 31 columnas. En relación a esto, se comprobaron los valores perdidos y los tipos de datos, claro previo se realizó la preparación con PCA, se limpiaron los datos. Así pues, se sondeó la clase etiqueta, donde encontramos los siguientes valores: a) Recuento de transacciones normales: 284315 y b) Recuento de transacciones fraudulentas: 492. Separamos los datos de las etiquetas, mediante la función (.iloc), que nos permitió seleccionar una celda en particular. El siguiente paso fue estandarizar los datos con el método (StandarScaler) para que tengan media cero y varianza unitaria. Una vez hecho lo anterior, se particionó el dataset para el entrenamiento y las pruebas correspondientes de los algoritmos. Como se señaló anteriormente, el dataset está desequilibrado. El entrenamiento de conjuntos de datos desequilibrados con algoritmos de aprendizaje puede llevar a una clasificación errónea de la clase minoritaria. Por lo tanto, para compensar el desequilibrio, usaremos el método de sobre muestreo SMOTE como se implementó en el paquete desequilibrado-aprendizaje para volver a muestrear el conjunto de datos. Adicionalmente, se entrenaron 3 algoritmos de Machine Learning: Random Forest, Logistic Regression, Naive Bayes, a manera comparativa, empleando los datos de características procesados.

5. **Evaluar mediante pruebas el resultado de los algoritmos empleados.**

   Por último, se evaluaron los algoritmos, en base a los indicadores de exactitud, precisión, Recall, F2 o FScore y AUC o Curva ROC. Asimismo, con grado de CPU, consumo de memoria y el tiempo de respuesta estimado.

Arquitectura del Método

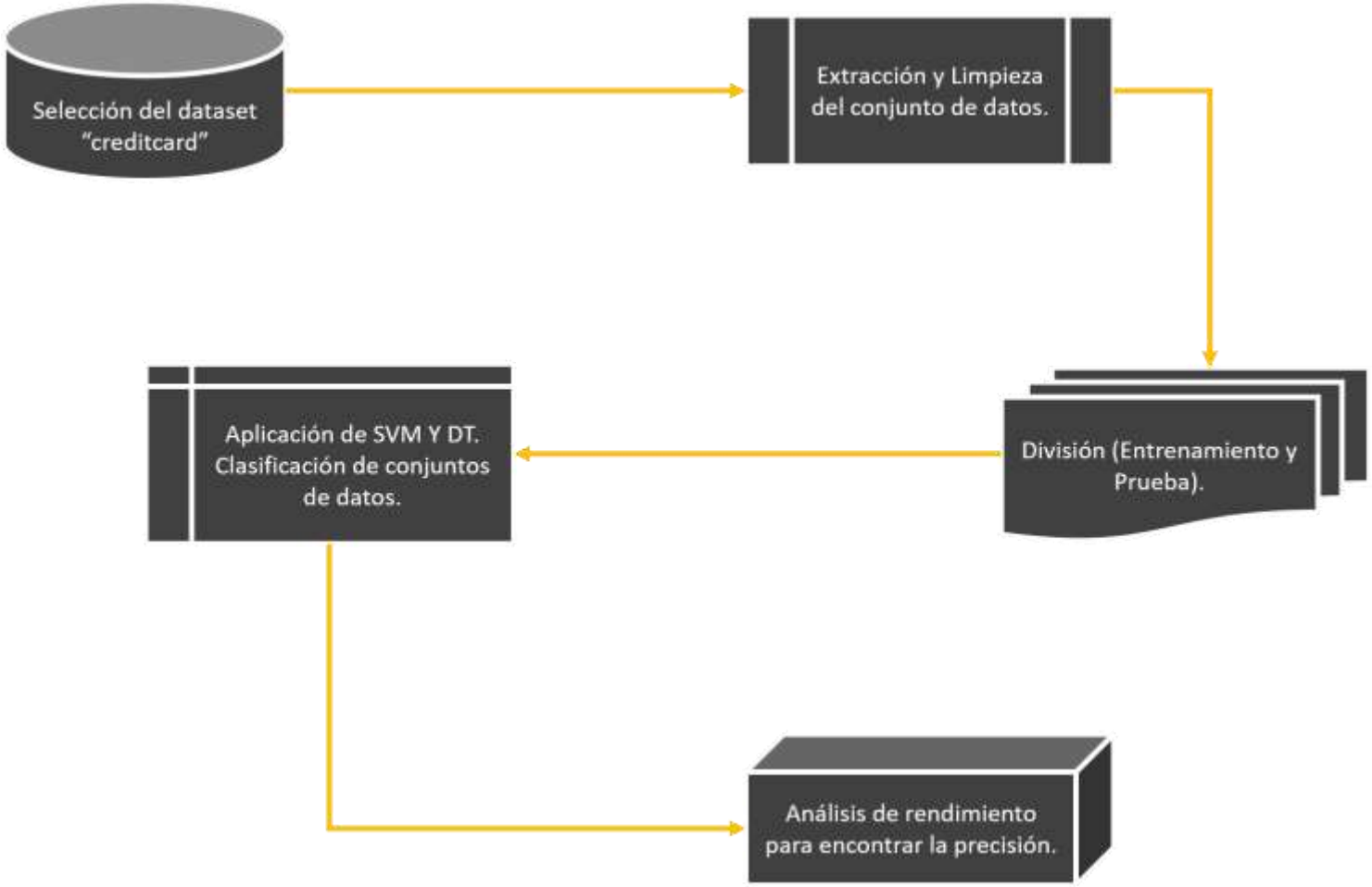


Figura 22. Representación de la arquitectura del método para la aplicación de Algoritmos. Fuente: Elaboración propia

Desarrollo de Objetivos Específicos

A) Analizar los algoritmos existentes.

Para llevar a cabo, el primer punto se ha realizado una matriz con los papers orientados al tema principal: detección de fraudes en línea. Dicho esquema contiene los campos (Año, Algoritmo, Objetivo, Método de los autores para detectar o predecir mediante clasificadores las transacciones fraudulentas o ilegítimas, Indicadores como Precisión y Exactitud, y finalmente la fuente de información para validar los campos descritos). A continuación, se detalla lo mencionado en la Tabla 16.

Tabla 16

Matriz de Algoritmos Investigados en el Tema de Detección de Fraudes en Línea.

Año	Algoritmo de ML	Objetivo	Método	Precisión	Exactitud	Referencia
2019	K-Nearest Neigbor	Proponer la aplicación de un modelo de clasificación predictiva mediante la implementación de algoritmos de aprendizaje automático (KNN, ELM, RF, MLP y BC) para conocer la precisión y otros parámetros de rendimiento para	a) Efectuaron una revisión sistemática y completa de los artículos existentes sobre la aplicación de algoritmos de clasificación de aprendizaje automático a CCFD. b) Seleccionaron 5 algoritmos de clasificación de machine learning para trabajar en Jupyter en la versión 3.6 de Python. c) Seleccionaron un dataset para aplicar los algoritmos, en	0.8852	0.8143	(Prusti & Kumar Rath, 2019)

		identificar la transacción fraudulenta.	este caso constó de 30.000 filas. 80% para entrenamiento y 20% para pruebas. d) Se entrenaron los algoritmos por separado en primera instancia y se obtuvo la precisión individual. e) Se realizó un modelo de clasificación hibridado para reducir la varianza y el sesgo. Así mejorar la precisión de la predicción en transacciones fraudulentas o legítimas. f) Evaluaron parámetros de rendimiento como exactitud, sensibilidad, especificidad, precisión, puntuación F1, coeficiente de correlación de Matthews (MCC). Utilizaron una matriz de confusión y explicaron su importancia para el desarrollo del modelo.			
2019	Naive Bayes	Analizar el desempeño de los algoritmos de Naïve Bayes, Regresión logística, J48 y AdaBoost en la detección de fraudes en línea.	a) Realizaron una revisión literaria sobre Algoritmos de Machine Learning aplicados a la detección de fraudes de tarjetas de crédito. b) Preseleccionaron los algoritmos a trabajar mediante matriz de ventajas y desventajas de los mismos.	0.83	0.83	(Naik & Kanikar, 2019)

			c) Seleccionaron los mejores algoritmos de ML para detectar fraudes en base al indicador exactitud. d) Adquirieron un conjunto de datos en línea. e) Implementaron los Algoritmos de Machine Learning en Python. f) Evaluaron los resultados.			
2019	Support Vector Machines	Realizar un análisis comparativo de identificación de actividad fraudulenta en tarjetas de crédito utilizando la máquina de vectores de soporte, la técnica del vecino más cercano k, bayes ingenuos y técnicas de regresión logística valiéndose de información sesgada.	a) Realizaron una Revisión de literatura, donde escogieron SVM, NB, KNN y LR. b) Seleccionaron el conjunto de datos del repositorio Kaggle, el cual presenta 3075 transacciones. c) Efectuaron un procesamiento de datos para el tipo categórico a numérico. d) Aplicaron los 4 algoritmos de machine learning seleccionados. e) Evaluaron los modelos de MA con métricas de precisión de clasificación, matriz de confusión, exactitud, sensibilidad, especificidad.	0.851	0.9753	(Adepoju, Wosowei, lawte, & Jaiman, 2019)
2021	Artificial Neural Network	Predecir la ocurrencia del fraude en tarjetas de crédito a partir de la	En este documento, las implementaciones comienzan con la carga del conjunto de	0.81	0.99	(RB & Kumar, 2021)

		utilización de SupportVectorMachines, KNN y Artificial Neural Network.	datos. Que el preprocesamiento de datos realizado que incluye la limpieza y normalización de datos. El conjunto de datos se divide en dos conjuntos de datos como datos de entrenamiento y datos de prueba, el modelo se entrena y se prueba. Finalmente, el sistema predice si la transacción es fraudulenta o no. El lenguaje usado fue Python. Las librerías son Numpy, Pandas, Keras. El sistema gestor bd fue MYSQL. Para el aspecto gráfico se trabajó con Tkinter.			
2019	Siamese Neural Network	Diseñar un modelo de detección de fraude en transacciones en línea basado en la red neuronal siamesa, donde CNN se usará para caracterizar el aprendizaje y LSTM para hacer la estructura de memoria la red.	a) Crearon el Framework donde se implementará el modelo siamés. b) Mapearon el modelo de la estructura de la red neuronal siamesa. c) Investigaron y diseñaron la función de memoria de red utilizando estructura siamesa LSTM. d) Adquirieron el dataset de transacciones B2C de un	0.95	0.97	(Zhou, Zhang, Wang, & Wang, 2019)

			banco comercial nacional, donde existían 5 millones de registros. e) Efectuaron una limpieza y transformación para garantizar la consistencia de los datos. f) Diseñaron la red neuronal convolucional para abordar por completo la relación no lineal compleja arbitraria, fuerte robustez y tolerancia a fallas, alta velocidad para encontrar la capacidad de solución óptima, autoaprendizaje y capacidad de adaptación. g) Evaluaron mediante la precisión, Recall y exactitud, tanto la red CNN, LSTM como la red neuronal siamesa.			
2019	Decision Tree	Desarrollar un método de detección de fraude para Streaming Transaction Data en transacciones de crédito europeas, con el objetivo de analizar los detalles de las transacciones pasadas de los clientes y extraer los patrones de comportamiento.	a) Agruparon a los clientes (titulares de tarjeta) en función de los montos de sus transacciones, es decir, alto, medio y bajo utilizando la partición de rango. b) Extrajeron patrones de comportamiento según el perfil de cada titular. c) Aplicaron los clasificadores en tres grupos distintos.	0.9814	0.9708	(Dornadulaa & S, 2019)

			d) Generaron puntajes de calificación para cada tipo de clasificador. e) Se apoyaron de un mecanismo de feedback para resolver el problema de deriva de concepto. f) Trabajaron el coeficiente CCM, el cual fue el mejor parámetro para tratar el problema del desequilibrio. g) Asimismo, aplicaron SMOTE, donde obtuvieron mejor funcionamiento de los clasificadores haciendo uso del mismo. h) Evaluaron a los clasificadores en base a las pruebas realizadas y los resultados conseguidos según los indicadores planteados.			
2020	Multilayer Perceptron	Detectar transacciones fraudulentas utilizando dos clasificadores de redes neuronales artificiales: MLP Y ELM.	a) Utilizaron la técnica del selector de funciones. b) Procesaron los datos para hallar las características más importantes, que ayuden a resolver el desequilibrio del dataset. c) Utilizaron SMOTE para conjuntos de datos desequilibrados.	0.9932	0.9784	El hlouli et al., (2020)

			d) Compararon MLP Y ELM en indicadores de precisión, rendimiento de clasificación y velocidad de aprendizaje.			
2019	Random Forest	Utilizar el algoritmo de bosque aleatorio (RFA) para encontrar las transacciones fraudulentas y la precisión de esas transacciones.	a) Recopilaron el dataset de la tarjeta de crédito y lo analizaron. b) Limpiaron los datos desequilibrados, valores duplicados o nulos. c) Efectuaron una división en el dataset de dos categorías: entrenamiento y test. d) Aplicaron el algoritmo Bosque Aleatorio para mejorar la precisión. e) Clasificaron el RFA en una matriz de confusión de 4 categorías. f) Evaluaron mediante gráficas el análisis del rendimiento del algoritmo para obtener valores en cuanto a la precisión.	0.90	0.90	Suresh et al., (2019)
2020	Convolucional Neural Network	Proponer un modelo híbrido entre CNN y KNN para la detección de fraude basado en transacciones europeas.	a) Importaron los datos de entrada. b) Efectuaron un procesamiento previo de los datos y eliminaron los datos falsos. c) Usaron la técnica CNN seleccionada.	0.81	0.8812	Nancy et al., (2020)

			d) Aplicaron el algoritmo KNN en el dataset, con método Fold. e) Crearon el Modelo Híbrido de predicción			
2017	Hidden Markov Model	Calcular la probabilidad de fraude basado en el registro de eventos a partir de la información de actividad registrada, haciendo uso del Márkov Oculto.	a) Seleccionaron el dataset con extensión csv. b) Usaron los datos de de eventos de registro de las transacciones bancarias. c) Calcularon el algoritmo mediante una matriz de probabilidad. d) Verificaron los valores de probabilidad obtenidos para saber si existe fraude o no. e) Generaron el flujo de trabajo algorítmico del modelo enfocado en el flujo del crédito bancario. f) Evaluaron resultados del algoritmo basado en eventos.	0.89	0.94	Rahmawati et al., (2017)
2018	Restricted Boltzmann Machine	Determinar cómo los algoritmos de aprendizaje profundo (EDT, SAE y RBM) pueden ser benévolos para detectar transacciones fraudulentas con alta precisión.	a) Seleccionaron datos y características del dataset. b) Efectuaron un preprocesamiento de los datos. c) Establecieron medidas de evaluación para los indicadores. d) Realizaron una comparación respecto a las	0.81	0.9153	(Mubalaike & Adali, 2018)

			precisiones y los resultados de los modelos DL, y RBM.			
2018	Neural Networks Long Short-Term Memory	Evaluar una subsección de topologías de aprendizaje profundo con componentes de tiempo y memoria integrados (LSTM), para identificar su eficacia en la detección de fraudes en un dataset de casi 80 millones de transacciones con tarjeta de crédito.	a) Utilizaron un entorno de computación en la nube distribuido de alto rendimiento para superar los problemas de desequilibrio de clases y la escalabilidad. b) Realizaron un procesamiento previo de datos, para identificar la clase etiqueta. c) Emplearon la ingeniería de funciones sobre el algoritmo LSTM. d) Aplicaron submuestreo para estimar la eficacia del algoritmo en las transacciones con tarjeta de crédito.	0.90	0.912	Roy et al., (2018)

Nota: Elaboración Propia.

B) Seleccionar Algoritmos de Machine Learning

En segunda instancia, el proceso de selección se ejecutó siguiendo criterios, los cuales se exponen en la Tabla 17. Aquí se explican las métricas para evaluar algoritmos, es importante mencionar que los indicadores exhibidos en la Tabla 21, no fueron alterados, sino que tal cual se encontraron, se anexaron a dicha matriz. Buscando la integridad de los datos, es que se trabajó de tal forma. En la Tabla 18 y Tabla 19, se definen los niveles para posteriormente efectuar las ponderaciones con fundamento en el desempeño de cada criterio establecido. Por último, se precisa que, para el Hardware, se consideró necesaria una matriz distinta debido que está enfocada al consumo de recursos que pueda requerir el computador para la aplicación del algoritmo. Como sabemos cada clasificador, tiene una complejidad distinta.

Tabla 17

Criterios a Utilizar.

N°	Criterio	Descripción
1	Facilidad de Codificación	Resulta sencillo programar el algoritmo, ya sea sistemas individuales o híbridos que se ajusten al tipo de problema para el que está orientado.
2	Complejidad	El grado de complicación en aspecto como configuración, anidación y flujo del algoritmo es mínimo.
3	Fiabilidad	El grado del algoritmo para conservar su nivel de desempeño en cualquier condición.
4	Simplicidad	El grado de comprensibilidad del algoritmo es mínimo. Involucra la comprensión de componentes básicos y criterios de combinación para generar las estructuras de control y de datos del algoritmo.
5	Hardware	Involucra la cantidad de recursos que requiere el algoritmo para obtener un desempeño adecuado cuando se ejecute el entrenamiento o prueba.

Nota: Elaboración propia.

Tabla 18

Clasificación Machine Learning.

N°	Desempeño	Descripción
1	Deficiente	El algoritmo de machine learning no cubre el criterio.
2	Insuficiente	El algoritmo de machine learning cubre el criterio inadecuadamente.
3	Aceptable	El algoritmo de machine learning cubre el criterio bien, pero quedan vacíos ciertos contenidos.
4	Excelente	El algoritmo de machine learning cubre el criterio satisfactoriamente.

Nota: Elaboración propia.

Tabla 19

Clasificación DB.*

N°	Desempeño	Descripción	Valor
A	Bajo	El algoritmo de machine learning requiere un procesador de gama baja.	1
B	Medio	El algoritmo de machine learning requiere un procesador de gama media.	2
C	Alto	El algoritmo de machine learning requiere un procesador de gama alta.	3

Nota: Elaboración propia.

En la siguiente sección, se visualizan los resultados obtenidos de las ponderaciones de los 12 algoritmos: KNN, NB, SVM, ANN, SNN, DT, MP, RD, CNN, HMM, RBM, LSTM. Asimismo, se agregan los indicadores extraídos de las investigaciones que aplicaron estos algoritmos en sus proyectos.

Tabla 20

Evaluación Cuantitativa de los Criterios según Algoritmo de Machine Learning.

Algoritmo / Criterio	KNN	NB	SVM	ANN	SNN	DT	MP	RD	CNN	HMM	RBM	LSTM
Facilidad de Codificación	3	3	4	2	2	4	3	3	2	2	2	2
Complejidad	3	3	4	3	3	4	3	3	3	2	2	3
Fiabilidad	4	3	4	3	3	4	4	3	4	4	3	4
Simplicidad	3	3	3	3	3	4	3	2	3	2	3	2
Hardware*	3	3	2	3	3	2	3	3	3	3	3	3
Ponderación	16	15	**17**	14	14	**18**	16	14	15	13	13	14

Nota: Elaboración propia.

*Consumo de recursos para aplicar el algoritmo en el dataset.

Tabla 21

Indicadores según Algoritmo de Machine Learning.

Algoritmo / Indicador	KNN	NB	SVM	ANN	SNN	DT	MP	RD	CNN	HMM	RBM	LSTM
Exactitud	0.8143	0.83	0.9753	0.99	0.97	0.9708	0.9784	0.90	0.8812	0.94	0.9153	0.91
Precisión	0.8852	0.83	0.851	0.81	0.95	0.9814	0.9932	0.90	0.81	0.89	0.81	0.90
Recall	0.8916	-	0.9756	0.76	0.96	-	0.9635	-	1.00	1.00	-	-
F Score	0.8884	-	-	-	-	-	-	-	-	0.94	-	-
AUC	-	-	0.90	-	-	-	-	-	-	-	0.8183	0.33

MCC*	0.3360	-	-	-	-	0.9420	-	-	-	-	-	-

Nota: Elaboración propia.

*En ciertos casos, figura el valor del indicador MCC en lugar de AUC.

Resultados de la Evaluación Cuantitativa

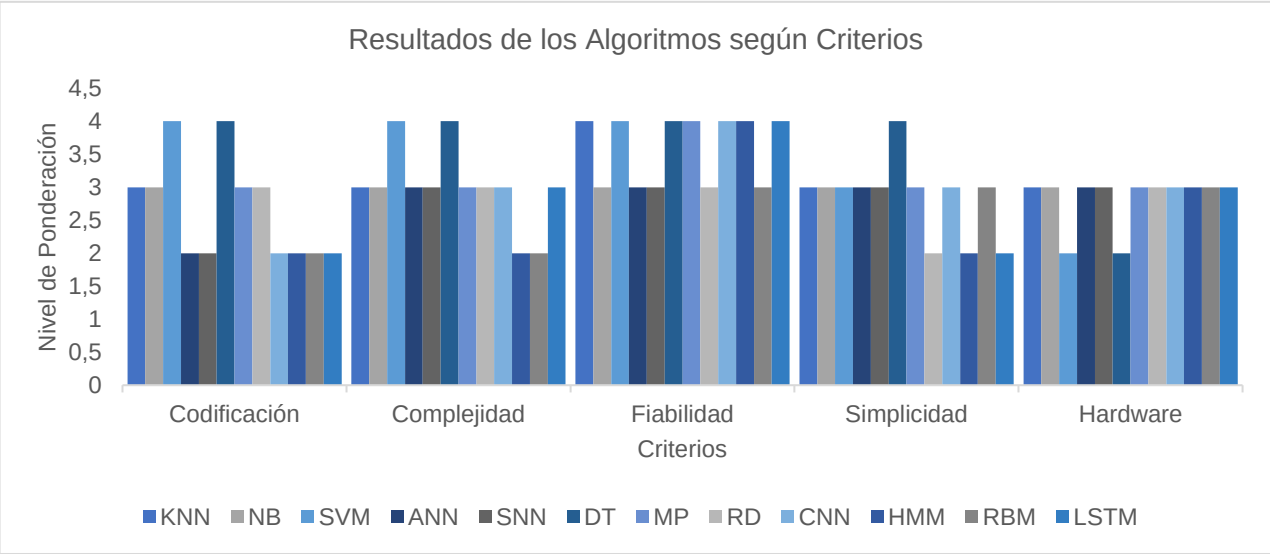


Figura 23. Evaluación del grado de cumplimiento de los algoritmos de machine learning en función de los criterios de selección. Fuente: Elaboración propia.

Podemos observar en la figura 23, que los algoritmos que obtuvieron mayor puntuación en criterios, fueron SVM y DT. De los cuales, fue por ellos por quienes optamos para la propuesta de la implementación: SVM (4,4,4,3,2) Y DT (4,4,4,4,2).

C) Aplicar los mecanismos de detección en un dataset

El desarrollo de este objetivo en esencia significa la implementación de los algoritmos. A continuación, se explica la manera como se realizó el proceso en su completitud.

La primera parte refiere al dataset, el cual se denomina “creditcard.csv”, alojado en el repositorio de Kaggle, publicado por el grupo de investigación Machine Learning Group ULB, y que obtiene una puntuación en escala del 1 a 10, de 8.5 de usabilidad

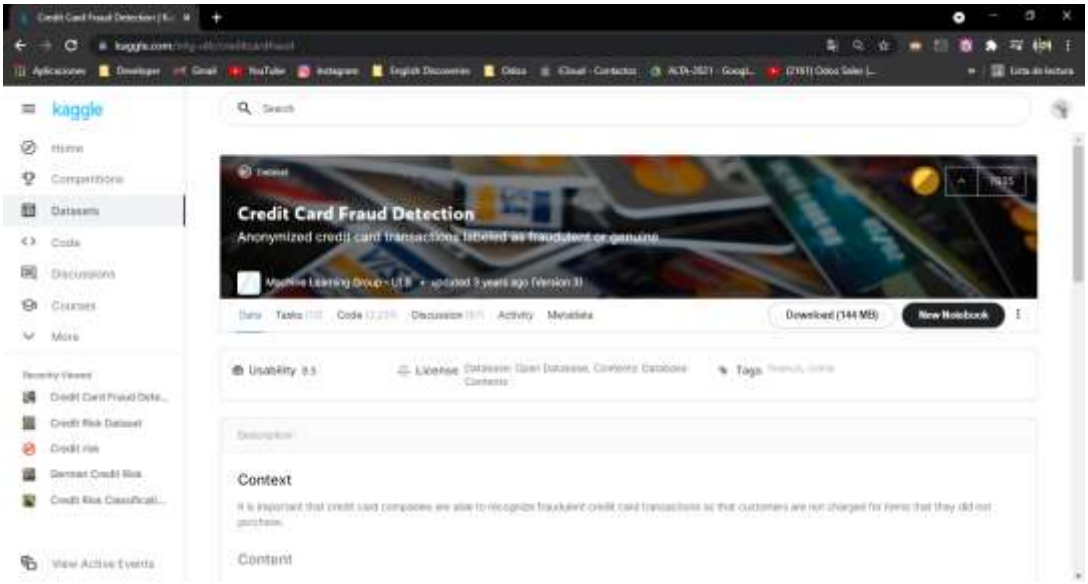


Figura 24. Credit Card Fraud Detection Dataset del repositorio Kaggle. Fuente: (Kaagle, 2021).

La figura 24, muestra la fuente del dataset, la licencia del mismo, lo cual otorga validez y veracidad. Asimismo, se observa la usabilidad y la cantidad de trabajos en los que se utilizó. Dicho esto, el peso del archivo es 143.84 MB. Posterior a ello, a conseguir la fuente se descargó el file, y tal como se visualiza en las figuras 25 y 26, se guardó en carpetas para luego ser preprocesado y procesado.

Figura 26. Ruta de descarga del Conjunto de datos. Fuente: Elaboración propia.

Figura 25. Ruta del archivo creditcard.csv. Fuente: Elaboración propia.

Se hizo una limpieza del dataset y se prepararon las características (Serie de tarjeta, número de serial, producto comprado, producto vendido, comprador, vendedor, tipo de movimiento (deposito o retiro), descripción de transacción, propiedad, valor del monto arancelario con impuesto añadiendo descuento, impuesto, tipo de impuesto, plazo, cancelaria, descuento de vta, descuento de compra, servicio, etc) entre otras características [V1, V2, hasta V28] con PCA. Recalcar que las únicas características que no se han transformado con PCA son Tiempo (número de segundos transcurridos entre esta transacción y la primera transacción en el conjunto de datos) y Monto (Importe de la transacción).

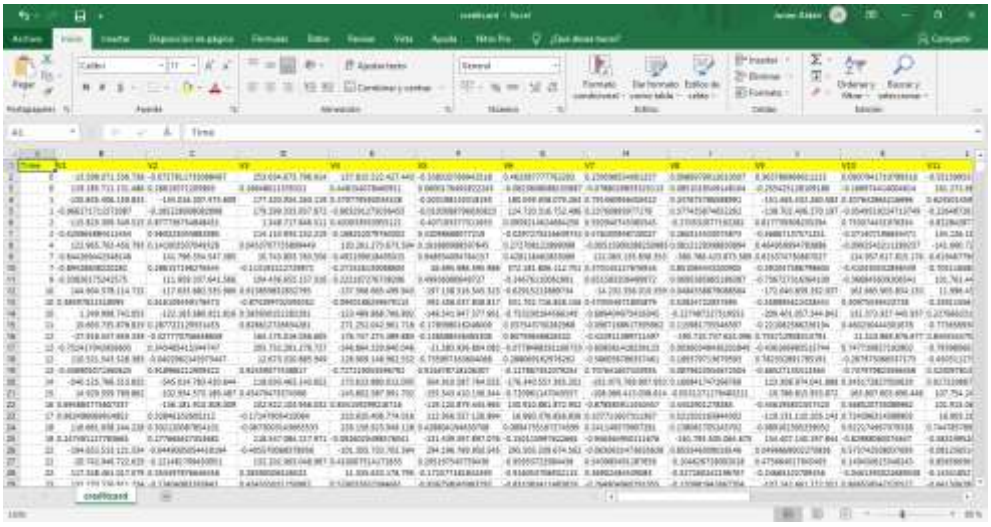


Figura 27. Dataset Creditcard primera parte.

Como se aprecia, el dataset contiene 284,807 transacciones de tarjetas de crédito realizadas por personas de origen europeo durante un lapso de 2 días, en septiembre de 2013. Puesto que la primera línea, son las características, Time, V1 …. V28, y Amount respectivamente. La preparación con PCA fue una actualización dada por el mismo repositorio el año 2021, ello se evidencia en dicha página web. La etiqueta denominada [Class] toma el valor 0 cuando la transacción es legítima y valor 1 cuando se trata de una transacción fraudulenta. Una vez que se efectuó lo anterior, se procede con la instalación de herramientas para la implementación del método con los algoritmos Support Vector Machines y Decision Tree. En primera instancia vamos a requerir la suite de Anaconda y Jupyter Notebook.

Figura 28. Versión de Anaconda instalada. Fuente: Elaboración propia.

Es importante saber que se puede ejecutar esto de dos formas, una descarga del archivo.exe y otra, mediante comandos por consola. Por consiguiente, de

manera rápida se muestran las líneas mencionadas. La línea [pip install jupyter] funciona como si descargáramos el ejecutable, la línea [conda -V] nos muestra la versión de Anaconda, la línea [where Jupyter] donde está instalado en nuestro pc Jupyter, y finalmente [python -V] la versión de Python en la que estamos trabajando.

En consecuencia, luego de instalar necesitamos ejecutarlo por ello, mediante la línea: C:\Users\ACER>jupyter notebook. Se nos cargará el localhost:8888, el cual es por default. La siguiente imagen, muestra a Jupyter en ejecución por consola. Una vez, ya instaladas correctamente las distribuciones anteriores, se procedió a la programación de líneas de código en Jupyter Notebook 6.3.0. Para la creación de nuestra carpeta en esta herramienta, se muestran las siguientes imágenes, donde primero se creó el folder Tesis Python, donde estará todo nuestro proyecto. Dentro del mismo, se guardó el dataset, con la ruta: C:\Users\ACER\Documents\Tesis Python\data. En la carpeta data, se anexó dicho creditcard.csv.

En el transcurso de este apartado se explica la programación de algoritmos, la extracción de características, limpieza de dataset, la aplicación del método SMOTE para corregir el desequilibrio de datos, para que no exista sesgo. Asimismo, se detalla la división de porcentajes para entrenamiento y prueba de los algoritmos de machine learning, todo ello mediante bloques de código fuente, para mayor entendimiento. Para la ejecución del presente sistema se requirieron las siguientes librerías:

- Línea 1 hasta la 7: leer y escribir datos (Pandas), vectores (Numpy), gráficos elegantes (Matplotlib y Seaborn), algoritmos de ML y funciones (Sklearn), números aleatorios (Random).

```
import pandas as pd
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import random
from sklearn.utils import shuffle
```

- Línea 8 hasta la 13: clasificar datos desbalanceados (Imblearn), equilibrar datos con técnica de sobremuestreo (SMOTE), funciones complejas de números (Math), preprocesamiento de datos (sklearn.preprocessing), transformar y estandarizar data (StandardScaler) y técnica para componentes principales (PCA).

```
import imblearn
from imblearn.over_sampling import  SMOTE
import math
import sklearn.preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

- Línea 14 hasta la 20: particionar el dataset mediante % para entrenamiento y prueba (sklearn.model_selection), indicadores de los algoritmos de machine learning (sklearn.metrics), biblioteca de colores (termcolor), seleccionar valores de parámetros (GridSearchCV), y matriz de confusión (sklearn.m y mlxtend.plotting).

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, precision_recall_curve, f1_score, roc_auc_score
from sklearn import metrics
from termcolor import colored as cl
from sklearn.model_selection import GridSearchCV
from mlxtend.plotting import plot_confusion_matrix
```

- Línea 21 hasta la 27: implementar los clasificadores de machine learning.

```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
```

Posterior se utilizó lectura para el dataset seleccionado con la siguiente línea: data=pd.read_csv("data/creditcard.csv"). Se visualizaron los siguientes gráficos, la figura 29 para representar la distribución del atributo Amount y la figura 30 para el atributo Time, el primero un diagrama de densidad y el segundo, un histograma. Ambos codificados en las líneas: sns.distplot(data['Amount']) y sns.distplot(data['Time'] respectivamente.

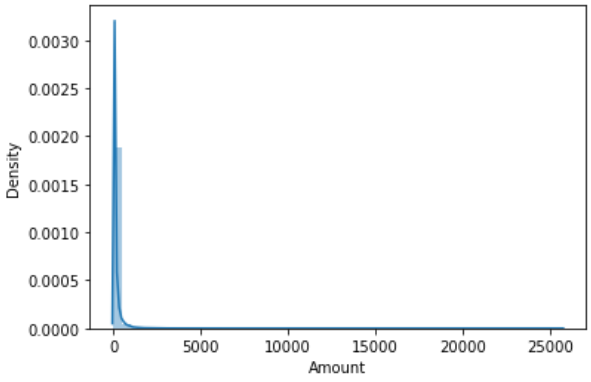


Figura 29. Diagrama del atributo Amount en Python. Fuente: Elaboración propia.

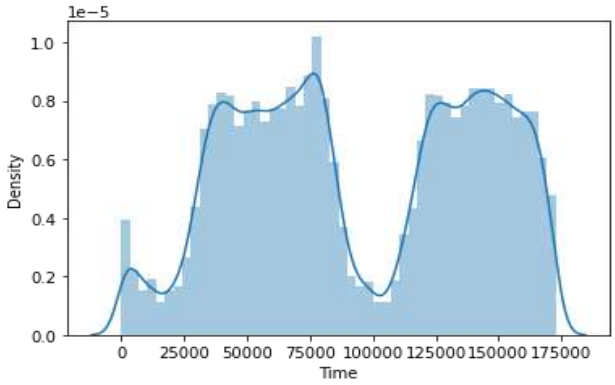


Figura 30. Histograma del atributo Time en Python. Fuente: Elaboración propia.

Asimismo, se efectuó lo anterior con los demás atributos y la clase etiqueta denominada [Class]. Evidencia de los atributos: Time, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, V28, Amount y Class, en la figura 31. La descripción de algunas de estas características es: código de serie, tarjeta, número de serial europeo, serie del producto comprado, serie del producto vendido, identificador de comprador, identificador de vendedor, tipo de movimiento (deposito o retiro), descripción de transacción, propiedad, valor arancelario, descuento, impuesto, tipo de impuesto, plazo, cancelaria, descuento de vta, descuento de compra, servicio, id de tipo de venta, etc

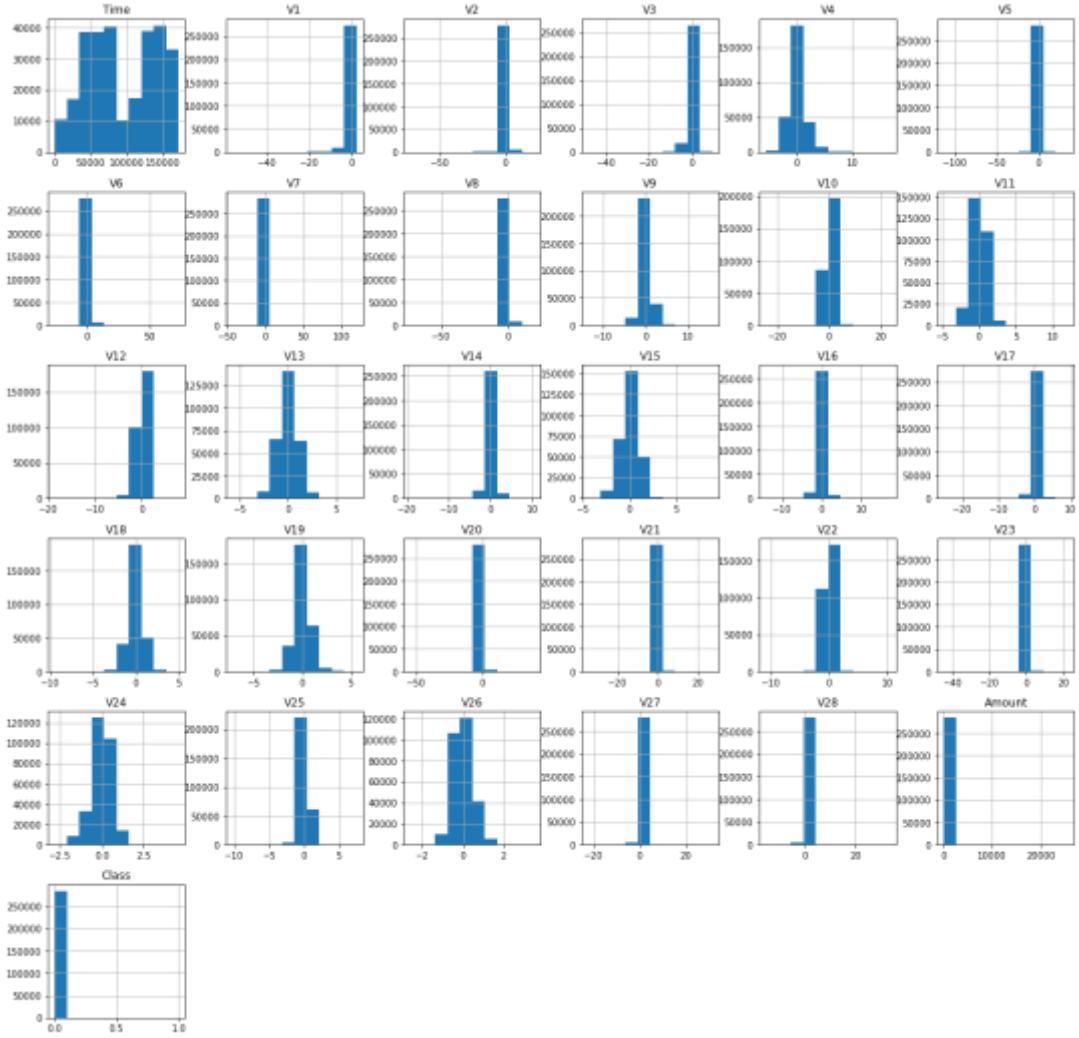


Figura 31. Histograma de 31 atributos en Python. Fuente: Elaboración propia

La figura 32, surge debido que se consideró necesario saber la relación del atributo Amount y Time. Por ello, se codificó para visualizar un diagrama de dispersión entre estas variables. En el eje x, se encuentra Time; en el eje y, Amount.

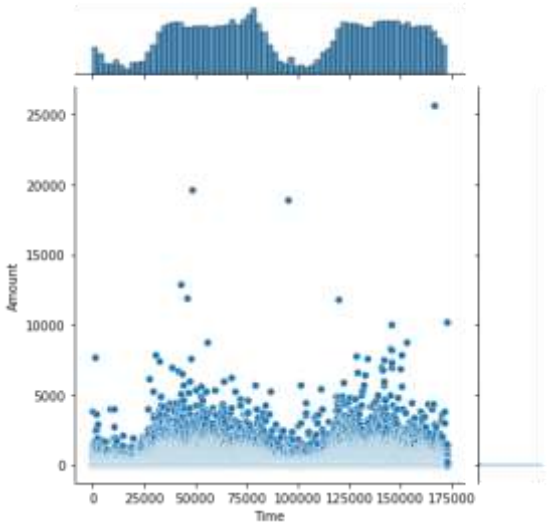


Figura 32. Diagrama de dispersión de la correlación de Time y Amount en Python. Fuente: Elaboración propia.

Para mostrar lo anterior: sns.jointplot(x= 'Time', y= 'Amount', data= d) , d=data , class0 = d[d['Class']==0]

Debido que el dataset tiene mucho sesgo, se empezó el proceso de extracción y limpieza de la data, para que los algoritmos trabajen sobre datos en equilibrio, para lograr mejores resultados en los indicadores. En el siguiente bloque, se devolvió los caracteres de class0 y class1, luego se aplica la función shuffle para cambiar el orden de los datos de manera aleatoria, posterior se seleccionó 2000 filas al azar, se creó un archivo temporal y finalmente se guardó en un archivo csv.

len(class0)
class1 = d[d['Class']==1]

```
len(class1)
class0
temp = shuffle(class0)
d1 = temp.iloc[:2000,:]
frames = [d1, class1]
df_temp = pd.concat(frames)
df_temp.info()
df= shuffle(df_temp)
df.to_csv('creditcard_ssampling.csv')
```

La primera gráfica observada referente al desequilibrio era exorbitantemente notable, sin embargo, con este 'creditcard_ssampling.csv' creado se busca obtener una mejora. A demostración de lo expuesto, la línea: sns.countplot('Class', data=df)

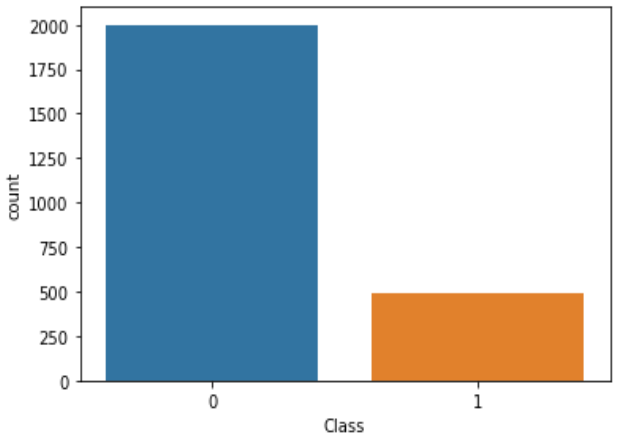


Figura 33. Representación de desequilibrio del dataset de la etiqueta clase en Python. Fuente: Elaboración propia.

Como se observó, el desequilibrio aún persiste, por lo que usaremos la técnica SMOTE para obtener mejoras en el desequilibrio de nivel considerable.

```
import imblearn
from imblearn.over_sampling import  SMOTE
oversample=SMOTE()
X=df.iloc[ : ,:-1]
```

```
Y=df.iloc[: , -1]
X,Y=oversample.fit_resample(X,Y)
(4000, 30)

Y=pd.DataFrame(Y)
Y.head()
names=['Time','V1','V2','V3','V4','V5','V6','V7','V8','V9','V10','V11','V12','V13','V1
4','V15','V16','V17','V18','V19','V20','V21','V22','V23','V24','V25','V26','V27','V28'
,'Amount','Class']
data=pd.concat([X,Y],axis=1)
d=data.values
data=pd.DataFrame(d,columns=names)
sns.countplot('Class', data=data)
```

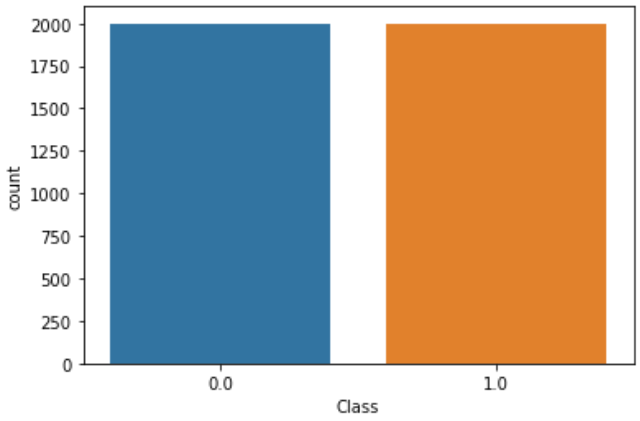


Figura 34. Representación de la aplicación de la técnica SMOTE en el dataset de la etiqueta clase en Python. Fuente: Elaboración propia.

A nivel de ejes se generó un mapa de calor(heatmap) para delimitar por colores los atributos y representarlo de manera bidimensional.

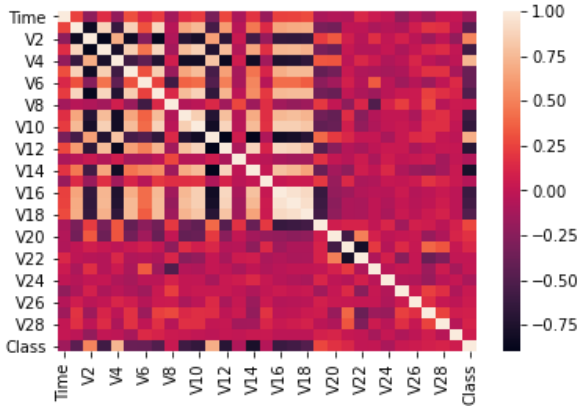


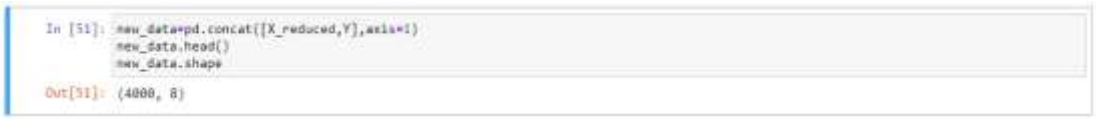
Figura 35. Mapa de calor de los atributos del dataset creditcard_saampling en Python. Fuente: Elaboración propia.

Se consideró necesario efectuar un preprocesamiento de datos con la transformación StandarScaler () para evitar los valores negativos, es decir una media 0 y una desviación estándar de 1. Asimismo, se aplicó el método PCA, para la extracción de características y condensar la información en pocos componentes, que serían los principales. Los cuáles fueron: names=['Time','Amount','Transaction Method','Transaction Id','Location','Type of Card','Bank']'. En español, las características fueron: Tiempo, Importe, Método de transacción, Codigo de transacción, Ubicación, Tipo de tarjeta, Banco. A continuación, se delimita por código fuente lo alegado.

```
import math
import sklearn.preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Feature Scaling
cols= ['V22', 'V24', 'V25', 'V26', 'V27', 'V28']
scaler = StandardScaler()
frames= ['Time', 'Amount']
x= data[frames]
d_temp = data.drop(frames, axis=1)
temp_col=scaler.fit_transform(x)
```

```
scaled_col = pd.DataFrame(temp_col, columns=frames)
scaled_col.head()
d_scaled = pd.concat([scaled_col, d_temp], axis =1)
# Dimensionality Reduction"""
from sklearn.decomposition import PCA
pca = PCA(n_components=7)
X_temp_reduced = pca.fit_transform(d_scaled)
pca.explained_variance_ratio_
pca.explained_variance_
names=['Time','Amount','Transaction Method','Transaction Id','Location','Type of Card','Bank']'
X_reduced= pd.DataFrame(X_temp_reduced,columns=names)
Y=d_scaled['Class']
new_data=pd.concat([X_reduced,Y],axis=1)
new_data.head()
new_data.shape
```

```
(4000, 8)
```



Recordar que tanto la extracción, limpieza, y transformación de la data fue para conseguir un dataset óptimo para poder trabajar y aplicar los algoritmos, donde se consiguieron buenas ponderaciones en la detección de fraudes. Como se observa, ahora el nuevo conjunto de datos tiene 4000 registros, cuyos datos están más equilibrados.

Finalmente, luego de las técnicas y métodos utilizados para el desbalance de datos, se generó un archivo denominado ‘creditfinal_ddata.csv’, el cual se particionó en 70% para entrenamiento y 30% para prueba.

```
new_data.to_csv('creditfinal_ddata.csv')
X_train, X_test, y_train, y_test= train_test_split(X_reduced, d_scaled['Class'], test_size = 0.30, random_state = 42)
```

X_train.shape, X_test.shape

```
((2800, 7), (1200, 7))
```

Terminada la etapa de la preparación del dataset, se procedió a la implementación de los clasificadores de machine learning, se detalla mediante código fuente como fue su aplicación.

El primer trabajo estuvo en la fusión de los algoritmos de SVM y DT, donde en el primer clasificador, se usó el kernel rbf (función de base radial), el cual se explicó con la fórmula siguiente:

$$\mathrm{K}(x, x') = e^{-\gamma \|x - x'\|^2}$$

Donde gamma debe ser 0 y manualmente se configura. En Python su valor por default de SVM de sklearn es:

$$\gamma = \frac{1}{n\ features * \sigma^2}$$

La primera es la distancia euclidiana x^2 entre dos vectores de características (2pt). Por otro lado, Gamma es un escalar que define cuánta influencia tiene un solo ejemplo de entrenamiento (1pt). Adicional a ello, explicar que también se creó el dataset para entrenamiento con el nombre 'train_cf.csv'. Con la selección del grid, se buscó encontrar los mejores parámetros para clasificar de manera sistemática en función de validación cruzada.

```
# Support Vector Machine
from sklearn.svm import SVC
svc=SVC (kernel='rbf', probability=True)
svc.fit(X_train,y_train)
y_pred_svc1=svc.predict(X_train)
y_pred_svc1

type(X_train)
```

```
X_train.to_csv('train_cf.csv')
from sklearn.model_selection import GridSearchCV
parameters = [ {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma': [0.1, 1, 0.01, 0.0001 ,0.001]}]
grid_search = GridSearchCV(estimator = svc,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           n_jobs = -1)
grid_search = grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print (''Best Accuracy: {:.2f} %”. format(best_accuracy*100))
print (''Best Parameters:'', best_parameters)
```

```
Best Accuracy: 97.14 %
Best Parameters: {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf
'}
```

Luego que se halló los mejores parámetros, se trabajaron en base a ello, consiguiendo buenos puntajes en: Accuracy, Precision, Recall, F1 Score, AUC.

```
svc_param=SVC (kernel='rbf’, gamma=0.01, C=100)
svc_param.fit(X_train,y_train)
y_pred_svc2=svc_param.predict(X_train)
con_mat = confusion_matrix(y_train, y_pred_svc2)
cls_report = classification_report(y_train, y_pred_svc2)
accuracy = metrics.accuracy_score(y_train,y_pred_svc2)
auc = metrics.roc_auc_score(y_train, y_pred_svc2)
print (cl ('================ ENTRENAMIENTO DE SVM ================', attrs = ['bold'], color = 'cyan'))
print (cl ('1. EXACTITUD: {}%'.format(np.round(accuracy, 3) * 100), attrs = ['bold'], color = 'grey'))
print (cl ('2. MATRIZ DE CONFUSION:', attrs = ['bold'], color = 'grey'))
```

```
print(con_mat)
print (cl ('3. REPORTE DE INDICADORES:', attrs = ['bold'], color = 'grey'))
print(cls_report)
print (cl ('4. AUC: {}'. format(roc_auc_score(y_train, y_pred_svc2)), attrs = ['bold'], color = 'grey'))
print(cl('5. REPRESENTACION GRAFICA DE MATRIZ DE CONFUSION:', attrs = ['bold'], color = 'grey'))
cm_svm= pd.DataFrame(con_mat, index=['normal', 'fraudulenta'], columns=['normal', 'fraudulenta'])
sns.heatmap(cm_svm, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("The Confusion Matrix"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
plt.show()
print(cl('================ ENTRENAMIENTO DE SVM ================', attrs = ['bold'], color = 'cyan'))
```

Por su parte la forma en que se empleó el clasificador DT fue, con los siguientes parámetros: entropy (por la aleatoriedad de la data), gini (divisiones binarias), max_depth(profundidad del árbol, en este caso se creó una lista de enteros con 3 argumentos, el cual inicia en 2 y termina en 4), min_samples(cantidad mínima de muestras en la hoja, aquí se creó una lista de enteros, inicia en 5 y termina en 7),

$$\text{Entropy} = -\rho\ log_2\ \rho - q\ log_2\ q$$

Donde $\rho\ y\ q$ son la probabilidad de éxito y fracaso de cada nodo, respectivamente. Mientras más baja sea la entropía, será mejor. Por consiguiente, luego de la explicación acerca del clasificador árbol de decisiones, se delimita la codificación.

```
# Support VM + Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier(svc)
dtree.fit(X_train,y_train)
```

```
y_pred_dtree1=dtree.predict(X_train)
d_tree_param=DecisionTreeClassifier(svc)
tree_parameters={'criterion':['gini','entropy'],'max_depth':list(range(2,4,1)),
                 'min_samples_leaf':list(range(5,7,1))}
grid_tree=GridSearchCV(d_tree_param,tree_parameters, svc)
grid_tree.fit(X_train,y_train)
y_pred_dtree2=grid_tree.predict(X_train)

con_mat = confusion_matrix(y_train,y_pred_dtree2)
cls_report = classification_report(y_train,y_pred_dtree2)
accuracy = metrics.accuracy_score(y_train,y_pred_dtree2)
auc = metrics.roc_auc_score(y_train,y_pred_dtree2)
print (cl('================ ENTRENAMIENTO DE DECISION TREE
================', attrs = ['bold'], color = 'cyan'))
print (cl('1. EXACTITUD: {}%'.format(np.round(accuracy, 3) * 100), attrs =
['bold'], color = 'grey'))
print (cl('2. MATRIZ DE CONFUSION:', attrs = ['bold'], color = 'grey'))
print(con_mat)
print (cl('3. REPORTE DE INDICADORES:', attrs = ['bold'], color = 'grey'))
print(cls_report)
print (cl ('4. AUC: {}'.format(roc_auc_score(y_train, y_pred_dtree2)), attrs =
['bold'], color = 'grey'))
print(cl('5. REPRESENTACION GRAFICA DE MATRIZ DE CONFUSION:',
attrs = ['bold'], color = 'grey'))
cm_dtree = pd.DataFrame(con_mat, index=['normal', 'fraudulenta'],
columns=['normal', 'fraudulenta'])
sns.heatmap(cm_dtree, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("The Confusion Matrix"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
plt.show()
print(cl('================ ENTRENAMIENTO DE DECISION TREE
================', attrs = ['bold'], color = 'cyan'))
```

La Tabla 22, expresa los valores (métricas) que obtuvo la combinación de SVM + DT en el entrenamiento realizado.

Tabla 22

Reporte de Indicadores del algoritmo SVM + DT en entrenamiento.

SUPPORT VECTOR MACHINES + DECISION TREE					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.972	0.972	0.973	0.972	0.976
Fraud	0.972	0.972	0.973	0.972	0.976

Nota: Elaboración Propia.

El segundo algoritmo fue RF, al ser un **tipo** de Árbol de decisión, reutilizaremos parte del código para aplicar este clasificador. Se inicializó Random Forest y se indicó la cantidad de árboles que va incluir el modelo, se muestra con el parámetro (n_estimators=2), en esta oportunidad no se utilizó oob_score, debido que se buscó mejorar el tiempo de respuesta del algoritmo. Tampoco, un max_depth, porque se probó en diferentes escenarios y los resultados fueron mejores, sin uso de ello.

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
randomforest=RandomForestClassifier(n_estimators=2)
randomforest.fit(X_train,y_train)
y_pred_rf2=randomforest.predict(X_train)
con_mat = confusion_matrix(y_train,y_pred_rf2)
cls_report = classification_report(y_train,y_pred_rf2)
accuracy = metrics.accuracy_score(y_train,y_pred_rf2)
auc = metrics.roc_auc_score(y_train,y_pred_rf2)
print(cl('================ ENTRENAMIENTO DE RANDOM FOREST
================', attrs = ['bold'], color = 'cyan'))
print(cl('1. EXACTITUD: {}%'.format(np.round(accuracy, 3) * 100), attrs =
['bold'], color = 'grey'))
```

```
print(cl('2. MATRIZ DE CONFUSION:', attrs = ['bold'], color = 'grey'))
print(con_mat)
print(cl('3. REPORTE DE INDICADORES:', attrs = ['bold'], color = 'grey'))
print(cls_report)
print(cl ('4. AUC: {}'.format(roc_auc_score(y_train, y_pred_rf2)), attrs = ['bold'],
color = 'grey'))
print(cl('5. REPRESENTACION GRAFICA DE MATRIZ DE CONFUSION:',
attrs = ['bold'], color = 'grey'))
cm_rafo = pd.DataFrame(con_mat, index=['normal', 'fraudulenta'],
columns=['normal', 'fraudulenta'])
sns.heatmap(cm_rafo, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("The Confusion Matrix"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
plt.show()
print(cl('================ ENTRENAMIENTO DE RANDOM FOREST
================', attrs = ['bold'], color = 'cyan'))
```

La Tabla 23, expresa los valores (métricas) que obtuvo RF en entrenamiento.

Tabla 23

Reporte de Indicadores del algoritmo RF en entrenamiento.

RANDOM FOREST					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.982	0.97	1.00	0.98	0.9829
Fraud	0.982	1.00	0.97	0.98	0.9829

Nota: Elaboración Propia.

El tercer algoritmo fue LR, el cual es para aprendizaje supervisado de clasificación. Se creó una instancia de Regresión Logística (línea 2), con la función fit(línea 3) se ajustó a las entradas X_train & y_train. En este caso, como es entrenamiento, se usó el modelo entrenado para predecir lr.predict(X_train). Posterior, se siguió con el reporte de indicadores y gráfica de la matriz de confusión, como se hizo con la totalidad de algoritmos.

Asimismo, se adjunta la fórmula de este clasificador.

$$f(x) = \frac{1}{1 + e^{-x}}$$

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
y_pred_lr2=lr.predict(X_train)
y_pred_lr2
con_mat = confusion_matrix(y_train,y_pred_lr2)
cls_report = classification_report(y_train,y_pred_lr2)
accuracy = metrics.accuracy_score(y_train,y_pred_lr2)
auc = metrics.roc_auc_score(y_train,y_pred_lr2)
print(cl('================ ENTRENAMIENTO DE REGRESION
LOGISTICA ================', attrs = ['bold'], color = 'cyan'))
print(cl('1. EXACTITUD: {}%'.format(np.round(accuracy, 3) * 100), attrs =
['bold'], color = 'grey'))
print(cl('2. MATRIZ DE CONFUSION:', attrs = ['bold'], color = 'grey'))
print(con_mat)
print(cl('3. REPORTE DE INDICADORES:', attrs = ['bold'], color = 'grey'))
print(cls_report)
print(cl ('4. AUC: {}'.format(roc_auc_score(y_train, y_pred_lr2)), attrs = ['bold'],
color = 'grey'))
print(cl('5. REPRESENTACION GRAFICA DE MATRIZ DE CONFUSION:',
attrs = ['bold'], color = 'grey'))
cm_lr = pd.DataFrame(con_mat, index=['normal', 'fraudulenta'],
columns=['normal', 'fraudulenta'])
sns.heatmap(cm_lr, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("The Confusion Matrix"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
```

```
plt.show()
print(cl('================ ENTRENAMIENTO DE REGRESION
LOGISTICA ================', attrs = ['bold'], color = 'cyan'))
```

La Tabla 24, expresa los valores (métricas) que obtuvo LR en entrenamiento.

Tabla 24

Reporte de Indicadores del algoritmo LR en entrenamiento.

LOGISTIC REGRESSION					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.968	0.95	0.99	0.97	0.9690
Fraud	0.968	0.99	0.95	0.97	0.9690

Nota: Elaboración Propia.

El último algoritmo fue NB, como ventaja de Naive Bayes está su simpleza en implementación en Python. Su simplicidad no sólo radica ahí, sino en genera. Este algoritmo funciona correctamente desde el principio, y rara vez es necesario ajustar sus parámetros. Otro concepto pro, que maneja es que en el entrenamiento su proceso de predicción fue muy rápido ya sea con el dataset sesgado como con el dataset estandarizado y transformado, con ambos se ejecutó previo a seleccionar las mejores condiciones. En razón a ello, en esta ocasión, no fue imperioso realizar modificaciones particulares dentro del mismo. A su vez, se expuso la fórmula matemática conocida para este clasificador y se indicó la programación por código.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Donde:

P(h): probabilidad de h. P(D): probabilidad de D.

(h|D): probabilidad posterior que se dé h dado D.

P(D|h): probabilidad que se dé D dado h.

```
# Naive Bayes
from sklearn.naive_bayes import BernoulliNB
naivebayes=BernoulliNB()
naivebayes.fit(X_train,y_train)
y_pred_nv2=naivebayes.predict(X_train)
con_mat = confusion_matrix(y_train,y_pred_nv2)
cls_report = classification_report(y_train,y_pred_nv2)
accuracy = metrics.accuracy_score(y_train,y_pred_nv2)
auc = metrics.roc_auc_score(y_train,y_pred_nv2)
print(cl('================ ENTRENAMIENTO DE NAIVE BAYES
================', attrs = ['bold'], color = 'cyan'))
print(cl('1. EXACTITUD: {}%'.format(np.round(accuracy, 3) * 100), attrs =
['bold'], color = 'grey'))
print(cl('2. MATRIZ DE CONFUSION:', attrs = ['bold'], color = 'grey'))
print(con_mat)
print(cl('3. REPORTE DE INDICADORES:', attrs = ['bold'], color = 'grey'))
print(cls_report)
print(cl ('4. AUC: {}'.format(roc_auc_score(y_train, y_pred_nv2)), attrs = ['bold'],
color = 'grey'))
print(cl('5. REPRESENTACION GRAFICA DE MATRIZ DE CONFUSION:',
attrs = ['bold'], color = 'grey'))
cm_nB= pd.DataFrame(con_mat, index=['normal', 'fraudulenta'],
columns=['normal', 'fraudulenta'])
sns.heatmap(cm_nB, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("The Confusion Matrix"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
plt.show()
print(cl('================ ENTRENAMIENTO DE NAIVE BAYES
================', attrs = ['bold'], color = 'cyan'))
```

La Tabla 25, expresa los valores (métricas) que obtuvo NB en entrenamiento.

Tabla 25

Reporte de Indicadores del algoritmo NB en entrenamiento.

NAIVE BAYES					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.951	0.92	0.99	0.95	0.953
Fraud	0.951	0.99	0.92	0.95	0.953

Nota: Elaboración Propia.

d) Evaluar mediante pruebas los resultados alcanzados.

He aquí nuestro objetivo final. Para dar cabida al proceso realizado en este apartado, se menciona que se particionó el dataset, para entrenamiento (train_cf.csv), y para prueba (tes_cf.csv). En temas porcentuales, el primero tuvo el 70% y el segundo el 30%. Surge para ver la correlación de los algoritmos, y evaluar mediante pruebas si los clasificadores fueron entrenados correctamente. En el mismo orden que se presentaron los algoritmos de machine learning anteriores, se ejecutaron las pruebas. La línea de código: X_train.shape, X_test.shape. Nos dio información sobre las transacciones o registros en cada archivo csv. Tanto para train como test. Se observó (2800, 7) y (1200, 7) respectivamente. Entonces luego de que tenemos conocimiento del dataset de test, se procedió a programar los algoritmos para que se ejecuten en el mismo. En esta ocasión, no se delimitó el código fuente utilizado para las pruebas dado que es la réplica del hecho para entrenamiento, donde sólo varían ciertos parámetros y configuraciones de menor relevancia. Pero, el proyecto final codificado en su completitud se incorporó en https://drive.google.com/drive/folders/1F1PlhDprT5Xd2cIPgJQQ7K3fDYEo19v1?usp=sharing. Asimismo, se realizaron matrices para comparar los números alcanzados de entrenamiento con prueba. El primer elemento que se sometió al test fue la combinación de Support Vector Machine + Decision Tree propuesta.

Dado los resultados de prueba de SVM + DT, se generó la tabla 26.

Tabla 26

Reporte de Indicadores del algoritmo SVM + DT en prueba.

SUPPORT VECTOR MACHINES + DECISION TREE					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.993	0.994	0.987	0.991	0.990
Fraud	0.993	0.994	0.987	0.991	0.990

Nota: Elaboración Propia.

Se observó que el modelo mejoró: en exactitud un (0,021), en precisión se mantuvo igual, en exhaustividad un (0,014), en f1 score un (0,019) también y en auc score incrementó un (0,014). Por lo que se concluye, que el método planteado entre SVM + DT, fue entrenado correctamente.

Dado los resultados de prueba de RF, se generaron la tabla 27.

Tabla 27

Reporte de Indicadores del algoritmo RF en prueba.

RANDOM FOREST					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.971	0.97	0.97	0.97	0.9705
Fraud	0.971	0.97	0.97	0.97	0.9705

Nota: Elaboración Propia.

Se observó que el modelo en exactitud se diferencia un (0,01), en precisión disminuyó un (0,03), en exhaustividad se conservó igual y en f1 score son casi similares por milésimas Por lo que se concluye, que el modelo RF fue entrenado correctamente. El cuarto algoritmo que se sometió al test fue Logistic Regression.

Dado los resultados de prueba de LR, se generaron la tabla 28.

Tabla 28

Reporte de Indicadores del algoritmo LR en prueba.

LOGISTIC REGRESSION					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.972	0.96	0.99	0.97	0.9706
Fraud	0.972	0.98	0.96	0.97	0.9706

Nota: Elaboración Propia.

Se observó que el modelo mejoró: en exactitud un (0,004), en precisión disminuyó un (0,01), en exhaustividad incrementó un (0,01), en f1 se conservó igual y en auc score mejoró un (0,0016). Por lo que se concluye, que el modelo LR fue entrenado correctamente. El quinto y último algoritmo que se sometió al test fue Naive Bayes.

Dado los resultados de prueba de NB, se generaron la tabla 29.

Tabla 29

Reporte de Indicadores del algoritmo NB en prueba.

NAIVE BAYES					
Class	Accuracy	Precision	Recall	F1 Score	AUC
Not Fraud	0.9589	0.95	0.98	0.96	0.9578
Fraud	0.9589	0.97	0.94	0.96	0.9578

Nota: Elaboración Propia.

Se observó que el modelo mejoró: en exactitud un (0,0079), en precisión disminuyó un (0,02), en exhaustividad incrementó un (0,02), en f1 aumentó un (0,01) y en auc score mejoró un (0,0048). Por lo que se concluye, que el modelo NB fue entrenado correctamente.

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones.

- Se realizó un estudio teórico-práctico de 30 investigaciones diferentes, donde se analizaron 12 algoritmos que estuvieron orientados al tema de detección de fraude de pagos en línea, estos fueron K Nearest Neighbors, Naive Bayes, Support Vector Machines, Artificial Neural Network, Siamese Neural Network, Decision Tree, Multilayer Perceptron Algorithm, Random Forest, Convolucional Neural Network, Hidden Markov Model, Restricted Bolztmann Machines, y Neural Networks Long Short-Term Memory. La evidencia de ello, se especificó en una matriz.

- Se seleccionó en base a criterios descrito en la tabla 17, los algoritmos y a una previa evaluación en donde se sometieron los 12 algoritmos analizados anteriormente, y finalmente se decidió optar por Support Vector Machines y Decision Tree, debido que en investigaciones previas alcanzaron los mejores porcentajes y el ambiente en donde se desempeñaron no requería mayor inconveniente.

- Se aplicó el método propuesto, siendo necesario preparar, estandarizar y transformar el conjunto de datos debido que inicialmente estuvo muy sesgado y los datos se encontraban en desequilibrio. Todo ello se hizo para evitar el oversampling, logrando así conseguir mejores resultados en la detección de fraudes. Como evidencia de este punto, está el desarrollo secuencial determinado de este objetivo.

- Se evaluaron los algoritmos mediante un dataset de prueba, la evidencia de este acápite fue la matriz de confusión de cada clasificador para conocer su desempeño en la detección de fraudes de pagos en línea.

4.2. Recomendaciones.

- Se recomienda seguir investigando acerca de los nuevos algoritmos que son aplicados a la detección del fraude de pagos en línea, una ventaja de este trabajo de investigación fue que se realizaron distintas transformaciones a la data para equilibrarla haciendo uso del método SMOTE, en anteriores artículos los autores habían trabajado este problema enfocado en detección de fraudes con ADASYN, CA. En evidencia de lo comentado, están los trabajos previos y las búsquedas efectuadas en bases científicas como IEEE XPLORE, ScienceDirect y SCOPUS, las cuales se sugiere revisar debido a su nivel de credibilidad y prestigio a nivel mundial.

- Se recomienda seguir indagando en los nuevos métodos o técnicas para trabajar datasets con sesgo o desequilibrados, dado que, mejores serán los resultados si conseguimos trabajar con un conjunto de datos estandarizado. En esta oportunidad, se aplicó la técnica SMOTE, pero seguro surgirán nuevos procedimientos en el transcurso del tiempo. Por ello, se sugiere mantenerse actualizado en conceptos tan importantes como el machine learning.

- Se recomienda aprender acerca del lenguaje Python, ya que es el lenguaje del futuro, en temas como la inteligencia artificial está siendo usado a nivel mayoritario y su tendencia sigue en auge, debido a su simplificación para codificar, su versatilidad, legibilidad y flexibilidad. Asimismo, es excelente ahorrando tiempo y recursos.

REFERENCIAS.

ACENS. (2016). WHITEPAPER: QUÉ ES EL PHISHING Y CÓMO PROTEGERSE. *QUÉ ES EL PHISHING Y CÓMO PROTEGERSE*. Recuperado el 30 de octubre de 2020, de https://www.acens.com/wp-content/images/2014/10/wp-phising-acens.pdf

Adepoju, O., Wosowei, J., lawte, S., & Jaiman, H. (2019). Comparative Evaluation of Credit Card Fraud Detection Using Machine Learning Techniques. Bangalore, India. Recuperado el 19 de 10 de 2020, de https://ieeexplore.ieee.org/document/8978372/

Adepoju, O., Wosowei, J., lawte, S., & Jaiman, H. (2019). Comparative Evaluation of Credit Card Fraud Detection Using Machine Learning Techniques. India. doi:10.1109/GCAT47503.2019.8978372

Awoyemi, J. O., Adetunmbi, A. O., & Oluwadare, S. A. (2017, diciembre 01). Credit card fraud detection using machine learning techniques: A comparative analysis. *2017 International Conference on Computing Networking and Informatics (ICCNI)*, 9. Lagos, Nigeria: IEEE. doi:10.1109 / ICCNI.2017.8123782

Dornadulaa, V., & S, G. (2019). Credit Card Fraud Detection using Machine Learning Algorithms. India. doi:10.1016/j.procs.2020.01.057

Florentina, C. (2015). Fraudes en Internet. Obtenido de http://repositori.uji.es/xmlui/bitstream/handle/10234/161252/TFG_2016_DincaClaudia.pdf?sequence=1

INCIBE. (2015). Fraude y gestión de la identidad online. Obtenido de https://www.incibe.es/sites/default/files/contenidos/dosieres/metad_fraude_y_gestion_de_la_identidad_online.pdf

John, G. H., & Langley, P. (1995). *Estimating continuous distributions in Bayesian classifiers*, 338-345. doi:10.5555/2074158.2074196

Kanika, & Singla, J. (2020). A Survey of Deep Learning based Online Transactions Fraud Detection Systems. Inglaterra. Recuperado el 19 de 10 de 2020, de https://ieeexplore.ieee.org/document/9160200

Kataria, S., & Tabrez, N. M. (2019). Internet Banking Fraud Detection Using Deep. Nueva Delhi, India. Recuperado el 19 de 10 de 2020, de https://ieeexplore.ieee.org/document/8991389

Kumar Rai, A., & Kumar Dwivedi, R. (julio de 2020). Fraud Detection in Credit Card Data using Unsupervised Machine Learning Based Scheme. Coimbatore, India. doi:10.1109/ICESC48915.2020.9155615

Kumar, P., & Iqbal, F. (abril de 2019). Credit Card Fraud Identification Using Machine Learning Approaches. Chennai, India. doi:10.1109/ICIICT1.2019.8741490

LA COMISIÓN FEDERAL DE COMERCIO. (2006). Estafas de loterías internacionales. Obtenido de https://www.consumidor.ftc.gov/articulos/s0086-estafas-de-loterias-internacionales

Larose, D. T. (2014). *Discovering knowledge in data: an introduction to data mining.* Hoboken, Nueva Jersey: John Wiley & Sons.

Mubalaike, A., & Adali, E. (2018). Deep Learning Approach for Intelligent Financial Fraud Detection System. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. Bosnia y Herzegovina. doi:10.1109/UBMK.2018.8566574

Naik, H., & Kanikar, P. (2019). Credit card Fraud Detection based on Machine Learning Algorithms. India. doi:10.5120/ijca2019918521

Nancy, A. M., Kumar, G. S., Veena, S., Vinoth, S., & Bandyopadhyay, M. (2020). Fraud detection in credit card transaction. *Department of software engineering, srm institute of science and technology*. India. doi:https://doi.org/10.1063/5.0025561

Perez, N. (2020). Análisis de Amenazas Cibernéticas. *El Fraude del príncipe nigeriano o phishing 419*. Obtenido de https://www.csirt.gob.cl/media/2020/05/AN2-2020-03.pdf

Prusti, D., & Kumar Rath, S. (2019). Fraudulent Transaction Detection in Credit Card by Applying Ensemble Machine Learning techniques. India. doi:10.1109/ICCCNT45670.2019.8944867

Puh, M., & Brkić, L. (mayo de 2019). Detecting Credit Card Fraud Using Selected Machine Learning Algorithms. Opatija, Croacia. doi:10.23919/MIPRO.2019.8757212

Rafalo, M. (2017). Real-Time fraud detection in credit card transactions. Varsovia, Polonia. Recuperado el 02 de noviembre de 2020

Raghavan, P., & El Gayar, N. (2019). Fraud Detection using Machine Learning and Deep Learning. *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*. Obtenido de https://ieeexplore.ieee.org/document/9004231

Rahmawati, D., Sarno, R., Fatichah, C., & Sunaryono, D. (2017). Fraud detection on event log of bank financial credit business process using Hidden Markov Model algorithm. *2017 3rd International Conference on Science in Information Technology (ICSITech)*. Indonesia. doi:10.1109/ICSITech.2017.8257082

RB, A., & Kumar, S. (2021). Credit card fraud detection using artificial neural network. India. doi:https://doi.org/10.1016/j.gltp.2021.01.006

Riffi, J., Adnane Mahraz, M., Tairi, H., El Yahyaouy, A., & El hlouli, F. Z. (9 de 6 de 2020). Credit Card Fraud Detection Based on Multilayer Perceptron and Extreme Learning Machine Architectures. *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. Marruecos. doi:10.1109 / ISCV49265.2020.9204185

Roy, A., Sun, J., Mahoney, R., Alonzi, L., Adams, S., & Beling, P. (2018). Deep learning detecting fraud in credit card transactions. *2018 Systems and Information Engineering Design Symposium (SIEDS)*. doi:10.1109/SIEDS.2018.8374722

Suresh Kumar, M., Soundarya, V., Kavitha, S., E.S. Keerthika, E. S., & Aswini, E. (2019). Credit Card Fraud Detection Using Random Forest Algorithm. *2019 3rd International Conference on Computing and Communications Technologies (ICCCT)*. India. doi:10.1109/ICCCT2.2019.8824930

Suresh Kumar, M., Soundarya, V., Kavitha, S., Keerthika, E., & Aswini, E. (febrero de 2019). Credit Card Fraud Detection Using Random Forest Algorithm. Chennai, India. doi:10.1109/ICCCT2.2019.8824930

Suresh, M. K., Soundarya, V., Kavitha, S., Keerthika, E., & Aswini, E. (2019). Credit Card Fraud Detection Using Random Forest Algorithm. Chennai, India. Recuperado el 19 de 10 de 2020, de https://ieeexplore.ieee.org/document/8824930

Thennakoon, A., Bhagyani, C., Premadasa, S., Mihiranga, S., & Kuruwitaarachchi, N. (2019). Real-time Credit Card Fraud Detection Using Machine Learning. *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 6. IEEE. Recuperado el 29 de octubre de 2020, de https://ieeexplore.ieee.org/document/8776942

Towards Data Science. (2019). *Support Vector Machines for Classification*. Obtenido de https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3

West, J., & Bhattacharya, M. (2016). An Investigation on Experimental Issues in Financial Fraud Mining. *80*, 1734-1744. Procedia Comput. Recuperado el 03 de noviembre de 2020

Zareapoor, M., KR, S., & Alam, A. (2012). Analysis on Credit Card Fraud Detection Techniques: Based on Certain Design Criteria. *52*, 35-42. doi:10.5120 / 8184-1538

Zhou, X., Zhang, Z., Wang, L., & Wang, P. (2019). A Model Based on Siamese Neural Network for Online Transaction Fraud Detection. Hungría. doi:10.1109/IJCNN.2019.8852295

Zhou, X., Zhang, Z., Wang, L., & Wang, P. (2019). Un modelo basado en la red neuronal siamesa para la detección de fraudes en transacciones en línea. *A Model Based on Siamese Neural Network for Online Transaction Fraud Detection*. Hungría. Recuperado el 19 de 10 de 2020, de https://ieeexplore.ieee.org/document/8852295/

Zojaji, Z., Atani, R. E., & Monadjemi, A. (2016). A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective. 1-26. Recuperado el 02 de noviembre de 2020

ANEXOS.

Anexo 01. Resolución de aprobación del proyecto de investigación.

USS | UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

RESOLUCIÓN N°0451-2021/FIAU-USS

Pimentel, 28 de mayo de 2021

VISTO:
El Acta de reunión N°1305-2021 del Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS remitida mediante oficio N°0227-2021/FIAU-IS-USS de fecha 19 de mayo de 2021, y;

CONSIDERANDO:
Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48° que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y tesis son aprobados por el Comité de Investigación y derivados a la Facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El periodo de vigencia de los mismos será de dos años, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24° señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; es individual o en pares para obtener un título profesional. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, según documentos de Vistos el Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS acuerdan aprobar los temas de las Tesis a cargo de los estudiantes del curso de Investigación II que se detallan en el anexo de la presente Resolución.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO 1°: APROBAR, el tema de la Tesis perteneciente a la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y MEDIO AMBIENTE, a cargo de los estudiantes del Programa de estudios de INGENIERÍA DE SISTEMAS según se detalla en el anexo de la presente Resolución.

ARTÍCULO 2°: ESTABLECER, que la inscripción del Tema de la Tesis se realice a partir de emitida la presente resolución y tendrá una vigencia de dos (02) años.

ARTÍCULO 3°: DEJAR SIN EFECTO, toda Resolución emitida por la Facultad que se oponga a la presente Resolución.

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Cc: Interesado, Archivo

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

RESOLUCIÓN N°0451-2021/FIAU-USS

Pimentel, 28 de mayo de 2021

ANEXO

N°	AUTOR (ES)	TEMA DE TESIS
1	BECERRA SANCHEZ MIGUEL ANGEL	EVALUCION DE RENDIMIENTO DE PROTOCOLOS DE COMUNICACIÓN DE IOT EN EL MONITOREO DE LA CALIDAD DEL AIRE
2	BOCANEGRA PINCHI YAN CARLOS	DESARROLLO DE UNA METODOLOGÍA PARA LA ADQUISICIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN ORIENTADA A PEQUEÑAS ORGANIZACIONES BASADA EN ESTÁNDARES
3	BRENIS LLAGUENTO JULIO ANTONIO	COMPARACIÓN DE PROTOCOLOS DE AUTENTICACIÓN EN CONEXIONES DE REDES PRIVADAS VIRTUALES PARA USO EN TRABAJO REMOTO. CASO DE ESTUDIO: MUNICIPALIDAD PROVINCIAL DE FERREÑAFE
4	- CARREÑO CASTILLO JORGE LUIS - SALAZAR AGUILAR LUIS	EVALUACIÓN DE ALGORITMOS PARA MEDIR EFICIENCIA EN EL TRAFICO OCULTO DE VOZ IP
5	CARREÑO GUERRERO SANTIAGO ANIBAL	MODELO PREDICTIVO DEL PROCESO DE VENTAS UTILIZANDO INTELIGENCIA DE NEGOCIOS Y DATA ANALITICS EN LA EMPRESA CENTRO TEXTIL DE LA MATTA S.A.C.
6	CARRERA SANCHEZ JOSE ANTONIO	EVALUACIÓN DE MARCOS DE TRABAJO PHP PARA EL DESARROLLO DE APLICACIONES WEB, BAJO LA NORMA ISO/IEC 25010, ENFOCADA A LA CALIDAD EN USO DEL PRODUCTO
7	CASTILLO CARDENAS JOSE LEONARDO	ANÁLISIS COMPARATIVO DE TÉCNICAS DE EVALUACIÓN EN USABILIDAD PARA MEDIR CALIDAD EN USO DE APLICACIONES WEB EN PEQUEÑAS EMPRESAS
8	CHIMPEN SERQUÉN MERLINA JESSICA	DISEÑO DE UNA MESA DE AYUDA BASADO EN BPMN E ITIL V3 PARA EL AREA DE TI DE LA MUNICIPALIDAD PROVINCIAL DE LAMBAYEQUE
9	CORONEL CAJAN ERICK ARTURO	RECONOCIMIENTO DE EXPRESIONES FACIALES DE TRISTEZA UTILIZANDO APRENDIZAJE PROFUNDO
10	- CORTEZ BURGOS JOHANDER ENRIQUE - MEDRANO MORI JOSE LUIS	DESARROLLO DE UN MODELO DE PROCESOS PARA UNA FÁBRICA PERUANA DE SOFTWARE BASADA EN METODOLOGÍAS ÁGILES CASO DE ESTUDIO CONASTEC S.R.L.
11	CRUZADO PEREZ SANDRA MARIANA	EVALUACIÓN DE EFECTIVIDAD DE UNA METODOLOGÍA ÁGIL DE ARQUITECTURA EMPRESARIAL PARA ALINEAR LOS SERVICIOS DE TI CON LOS OBJETIVOS DEL NEGOCIO EN UNA MICRO EMPRESA PERUANA DEL RUBRO DE MARKETING
12	FLORES ALEJANDRIA RONY ASUNCION	MODELO DE CALIDAD BASADO EN ESTANDARES PARA LOS PROCESOS DE DESARROLLO DE SOFTWARE EN PEQUEÑAS ORGANIZACIONES. Caso de estudio: Soluciones Virtuales VAIXS
13	- HUAMAN CASAS JUNIOR ALDAIR - SERRATO VILCHERRES FERNANDO JOSE	DESARROLLO DE UN MÉTODO PARA DETECCIÓN DE FRAUDES DE PAGOS EN LÍNEA UTILIZANDO APRENDIZAJE AUTOMÁTICO
14	MARTINEZ MONTENEGRO FERNANDO DANIEL	EVALUACIÓN DE ARQUITECTURA BLOCKCHAIN MA-ABS PARA LA GESTIÓN DE HISTORIAS CLÍNICAS ELECTRÓNICAS PERUANAS
15	PEREZ AVELLANEDA FRANKLIN	EVALUACIÓN DE ARQUITECTURA BLOCKCHAIN BASADO EN GUANG YANG PARA LA GESTIÓN DE HISTORIAS CLINICAS ELECTRONICAS PERUANAS
16	MENDOZA RENGIFO GENARO	DESARROLLO DE UNA METODOLOGÍA ÁGIL AD HOC PARA LA CREACIÓN DE APLICACIONES WEB EN PEQUEÑAS EMPRESAS. CASO DE ESTUDIO: SOLTI S.A.C
17	- MONTALVO SANDOVAL JOSE LUIS - RUBIO OTERO DANIEL	DESARROLLO DE UN MÉTODO DE IDENTIFICACIÓN DE DEFECTOS EXTERNOS DEL MANGIFERA INDICA L USANDO PROCESAMIENTO DE IMÁGENES DIGITALES Y APRENDIZAJE DE MÁQUINA
18	OLIVOS JULCA CHARLES	COMPARACIÓN DE TÉCNICAS DE RECONOCIMIENTO FACIAL PARA CONTROLAR LA ASISTENCIA DE LOS

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

RESOLUCIÓN N°0451-2021/FIAU-USS

Pimentel, 28 de mayo de 2021

N°	AUTOR (ES)	TEMA DE TESIS
		ESTUDIANTES DE NIVEL SECUNDARIO DE UNA INSTITUCIÓN EDUCATIVA
19	ORTEGA PUENTE OSCAR FERNANDO	DESARROLLO DE UNA APLICACIÓN MÓVIL DE REPORTE CIUDADANO UTILIZANDO IMÁGENES Y GEOLOCALIZACIÓN
20	PISFIL JAUREGUI JOSE LAURENT	SISTEMA WEB BASADO EN MICROSERVICIOS PARA MEJORAR LA INTEROPERABILIDAD DEL PROCESO DE CONSULTAS DE ARBITRIOS TRIBUTARIOS EN MUNICIPALIDADES
21	RAVELO RUIZ ALLEN MARCEL	EVALUACIÓN DE ALGORITMOS DE PREDICCIÓN APLICADOS AL MANTENIMIENTO PREDICTIVO DE MÁQUINAS ELÉCTRICAS ROTATIVAS
22	SUCLUPE CHAPOÑAN MANUELA DEL ROSARIO	SISTEMA DE INFORMACIÓN LOGÍSTICA PARA MEJORAR LA GESTIÓN Y CONTROL DEL PROCESO DE ALMACENAMIENTO DE LA CONSTRUCTORA YAVARI S.R.L. - PROVINCIA DE MAYNAS-LORETO.
23	VILLEGAS CASTAÑEDA CESAR RUDECINDO	ANÁLISIS COMPARATIVO DE SISTEMAS GESTORES DE BASE DE DATOS RELACIONAL Y NO RELACIONAL EN EL CONTEXTO DEL MANEJO DE INFORMACIÓN DE GRUPOS DE RESCATE INTERNACIONAL EN DESASTRES

Anexo 02. Instrumentos de recolección de datos, con su respectiva validación de los instrumentos.

CONSUMO DE RECURSOS			
CPU			
Algoritmo ML	Fecha	Hora	Valor
SVM + DT	12-10-2022	13:50:18	24.2
RF	12-10-2022	13:51:31	26.1
LR	12-10-2022	13:52:40	25.9
NB	12-10-2022	13:53:50	28.9
MEMORY			
Algoritmo ML	Fecha	Hora	Valor
SVM + DT	12-10-2022	13:50:18	68.5%
RF	12-10-2022	13:51:31	68.7%
LR	12-10-2022	13:52:40	69.2%
NB	12-10-2022	13:53:50	68.4%
RESPONSE TIME			
Algoritmo ML	Fecha	Hora	Valor
SVM + DT	12-10-2022	13:50:18	0.0112
RF	12-10-2022	13:51:31	0.0109
LR	12-10-2022	13:52:40	0.0173
NB	12-10-2022	13:53:50	0.0172

RENDIMIENTO			
ACCURACY SCORE			
Algoritmo ML	Fecha	Hora	Valor
SVM +	12-10-2022	13:50:18	0.993

DT			
RF	12-10-2022	13:51:31	0.971
LR	12-10-2022	13:52:40	0.972
NB	12-10-2022	13:53:50	0.958
PRECISION SCORE			
Algoritmo ML	Fecha	Hora	Valor
SVM + DT	12-10-2022	13:50:18	0.994
RF	12-10-2022	13:51:31	0.973
LR	12-10-2022	13:52:40	0.982
NB	12-10-2022	13:53:50	0.976
RECALL SCORE			
Algoritmo ML	Fecha	Hora	Valor
SVM + DT	12-10-2022	13:50:18	0.987
RF	12-10-2022	13:51:31	0.970
LR	12-10-2022	13:52:40	0.960
NB	12-10-2022	13:53:50	0.940
F1 SCORE			
Algoritmo ML	Fecha	Hora	Valor
SVM + DT	12-10-2022	13:50:18	0.991
RF	12-10-2022	13:51:31	0.970
LR	12-10-2022	13:52:40	0.970
NB	12-10-2022	13:53:50	0.960
AUC SCORE			
Algoritmo ML	Fecha	Hora	Valor
SVM + DT	12-10-2022	13:50:18	0.990
RF	12-10-2022	13:51:31	0.970

LR	12-10-2022	13:52:40	0.970
NB	12-10-2022	13:53:50	0.957

Modelos en Entrenamiento

ALGORITHM	CPU	MEMORY	TIME	ACCURACY	PRECISION	RECALL	F1	AUC
SVM + DT	23,3	67,2%	0.0150	0.972	0.972	0.973	0.972	0.976
RF	25,9	68,6%	0.0139	0.982	1.00	0.97	0.98	0.982
LR	27,2	67,1%	0.0140	0.968	0.99	0.95	0.97	0.969
NB	23,6	66,7%	0.0327	0.951	0.99	0.92	0.95	0.953

Modelos en Prueba

ALGORITHM	CPU	MEMORY	TIME	ACCURACY	PRECISION	RECALL	F1	AUC
SVM + DT	24,3	68,5%	0.0112	0.993	0.994	0.987	0.991	0.990
RF	26,1	68,7%	0.0109	0.971	0.97	0.97	0.97	0.970
LR	25,9	69,2%	0.0173	0.972	0.98	0.96	0.97	0.970
NB	28,9	68,4%	0.0172	0.958	0.97	0.94	0.96	0.957