Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Oscar Fernando López Barrios (20679)

Proyecto Final – Modelo de Detección de Transacciones Fraudulentas

Security Data Science (CC3094)

Catedrático: Jorge Andrés Yass Coy

Introducción

La detección de fraudes en transacciones con tarjeta de crédito representa un desafío crítico en la actualidad, requiriendo el empleo de técnicas avanzadas de Machine Learning y Deep Learning. Los modelos de aprendizaje automático desempeñan un papel crucial al identificar patrones anómalos que podrían indicar actividades fraudulentas. Sin embargo, la dinámica cambiante de los datos financieros exige métodos de entrenamiento que permitan a los modelos adaptarse continuamente a nuevas tendencias y comportamientos.

En este trabajo práctico, se investiga la viabilidad del entrenamiento incremental en modelos de Machine Learning y Deep Learning. El entrenamiento incremental, una técnica que actualiza los modelos existentes con nuevos datos sin necesidad de reentrenarlos completamente desde cero, resulta particularmente útil en escenarios donde los datos se generan de forma continua y donde un reentrenamiento completo puede ser costoso en tiempo y recursos computacionales.

El estudio se basa en un conjunto de datos simulado de transacciones con tarjeta de crédito, abarcando tanto transacciones legítimas como fraudulentas entre el 1 de enero de 2019 y el 31 de diciembre de 2020. Se analizarán y compararán tres algoritmos: LightGBM, XGBoost y Redes Neuronales Artificiales (ANN). El objetivo es evaluar las capacidades y limitaciones del entrenamiento incremental en cada uno de estos algoritmos, así como determinar las condiciones bajo las cuales es preferible un reentrenamiento total frente a uno incremental.

Además, se desarrollará una metodología basada en los resultados experimentales y en la revisión bibliográfica para decidir cuándo aplicar un reentrenamiento total o incremental. Esta metodología considerará factores como la variación en el rendimiento del modelo, el tiempo transcurrido desde el último reentrenamiento completo y la aparición de nuevas tendencias en los datos.

Resumen de la Investigación Teórica sobre Entrenamiento Incremental

El entrenamiento incremental, también conocido como aprendizaje continuo o aprendizaje en línea, es una técnica empleada en Machine Learning y Deep Learning que permite a los modelos actualizarse de manera continua con nuevos datos. Esta estrategia resulta especialmente útil en entornos dinámicos donde los datos evolucionan con el tiempo, permitiendo a los modelos adaptarse rápidamente a cambios en los patrones de los datos sin necesidad de ser reentrenados completamente desde cero.

Ventajas:

- Los modelos entrenados incrementalmente pueden ajustarse de manera continua a nuevos datos, lo que resulta crucial en entornos donde los datos cambian constantemente.
- Al actualizar los modelos únicamente con nuevos datos en lugar de reentrenarlos desde el principio, se reduce significativamente el costo computacional y el tiempo de entrenamiento.
- El entrenamiento incremental permite gestionar grandes volúmenes de datos que podrían ser impracticables de procesar en una sola iteración debido a limitaciones de memoria y tiempo.

Desventajas:

- Existe el riesgo de que el modelo se degrade si se le alimenta con datos que no son representativos del conjunto original o si los nuevos datos introducen ruido significativo.
- Es esencial desarrollar estrategias para garantizar que el modelo no pierda precisión con el tiempo, lo que puede incluir técnicas de regularización y validación continua.
- Si los datos incrementales están desequilibrados, el modelo puede volverse sesgado hacia las clases más frecuentes en los nuevos datos.

Adicionalmente, es importante considerar el monitoreo constante del rendimiento del modelo y la implementación de mecanismos de retroalimentación para corregir y actualizar su funcionamiento en función de los cambios en los datos y las condiciones del entorno.

Para determinar cuándo es preferible un reentrenamiento total frente a uno incremental, se recomienda:

- Monitoreo del Rendimiento: Evaluar continuamente métricas de rendimiento como ROC-AUC, precisión, recall y F1-score para identificar degradaciones significativas en la precisión del modelo.
- Intervalos de Tiempo: Considerar el tiempo desde el último reentrenamiento completo. En entornos con cambios rápidos en los datos, puede ser necesario reentrenar completamente más frecuentemente.
- Detección de Tendencias Nuevas: Implementar mecanismos para detectar cambios en los patrones de los datos que podrían justificar un reentrenamiento completo.

Algoritmos Investigados

- LightGBM: Es un framework de boosting de árboles de decisión que es altamente eficiente y escalable. Soporta el entrenamiento incremental mediante la actualización de modelos existentes con nuevos datos.
- XGBoost: Es un algoritmo de boosting basado en árboles que también soporta el entrenamiento incremental. Es conocido por su eficiencia y rendimiento en competiciones de machine learning.
- Redes Neuronales Artificiales (ANN): Las ANN pueden ser adaptadas para el entrenamiento incremental utilizando técnicas como la optimización continua y el ajuste de pesos con mini-batches de nuevos datos.

Balanceo de Datos:

Funcionamiento de SMOTE

Selección de Vecinos: Para cada instancia de la clase minoritaria, SMOTE selecciona aleatoriamente uno de sus k vecinos más cercanos (generalmente utilizando la distancia euclidiana).

Repetición: Este proceso se repite hasta que se alcanza el número deseado de nuevas instancias sintéticas.

Ventajas de SMOTE

Mejora el Rendimiento del Modelo: Al equilibrar las clases, SMOTE ayuda a los clasificadores a aprender patrones significativos de la clase minoritaria, mejorando métricas como el recall y el F1 Score.
Reducción del Overfitting: A diferencia del sobremuestreo tradicional (replicación de instancias), SMOTE reduce el riesgo de overfitting al generar muestras nuevas y diversas.
Aplicabilidad General: SMOTE se puede aplicar en combinación con otros algoritmos de clasificación y técnicas de preprocesamiento.

Limitaciones de SMOTE

Riesgo de Ruido: SMOTE puede introducir ruido si se crean instancias sintéticas que no representen adecuadamente la distribución real de los datos.
Dependencia de la Distancia: La efectividad de SMOTE puede verse afectada por la métrica de distancia utilizada para seleccionar vecinos, particularmente en datos con alta dimensionalidad o características heterogéneas.

Descripción de la Práctica

Descripción de las Variables:

1. amt: Monto de la transacción.
2. is_fraud: Indicador de si la transacción es fraudulenta (1) o no (0).
3. amt_month_shopping_net_spend: Gasto neto mensual en compras.
4. count_month_shopping_net: Número de transacciones mensuales en compras.
5. first_time_at_merchant: Indicador de si es la primera transacción con el comerciante (True) o no (False).
6. hour: Hora de la transacción.
7. minute: Minuto de la transacción.
8. second: Segundo de la transacción.
9. day: Día del mes de la transacción.
10. month: Mes de la transacción.
11. year: Año de la transacción.
12. distance_to_previous: Distancia a la ubicación de la transacción anterior del mismo usuario.
13. distance_to_owner: Distancia a la ubicación registrada del propietario de la tarjeta.
14. misc_net: Número de transacciones anteriores en la categoría 'misc_net'.
15. gas_transport: Número de transacciones anteriores en la categoría 'gas_transport'.
16. kids_pets: Número de transacciones anteriores en la categoría 'kids_pets'.
17. home: Número de transacciones anteriores en la categoría 'home'.
18. shopping_net: Número de transacciones anteriores en la categoría 'shopping_net'.
19. food_dining: Número de transacciones anteriores en la categoría 'food_dining'.
20. personal_care: Número de transacciones anteriores en la categoría 'personal_care'.
21. grocery_pos: Número de transacciones anteriores en la categoría 'grocery_pos'.
22. entertainment: Número de transacciones anteriores en la categoría 'entertainment'.
23. shopping_pos: Número de transacciones anteriores en la categoría 'shopping_pos'.
24. misc_pos: Número de transacciones anteriores en la categoría 'misc_pos'.
25. travel: Número de transacciones anteriores en la categoría 'travel'.
26. health_fitness: Número de transacciones anteriores en la categoría 'health_fitness'.
27. grocery_net: Número de transacciones anteriores en la categoría 'grocery_net'.
28. is_above_20: Indicador de si el monto de la transacción es mayor a 20.
29. is_above_50: Indicador de si el monto de la transacción es mayor a 50.
30. is_above_100: Indicador de si el monto de la transacción es mayor a 100.
31. is_above_500: Indicador de si el monto de la transacción es mayor a 500.
32. is_above_1000: Indicador de si el monto de la transacción es mayor a 1000.
33. month_of_year: Mes del año en que se realizó la transacción.
34. day_of_month: Día del mes en que se realizó la transacción.
35. time_since_last_transaction: Tiempo transcurrido desde la última transacción.
36. average_monthly_spend: Gasto promedio mensual del usuario.
37. average_daily_spend: Gasto promedio diario del usuario.
38. monthly_spend_variance: Varianza del gasto mensual del usuario.
39. unique_merchants_per_month: Número de comerciantes únicos visitados por mes.
40. transactions_per_location: Número de transacciones en una misma ubicación.

41. prev_lat: Latitud de la ubicación anterior del usuario.
42. prev_long: Longitud de la ubicación anterior del usuario.
43. prev_time: Tiempo de la transacción anterior.
44. distance_traveled: Distancia recorrida desde la última transacción.
45. time_traveled: Tiempo transcurrido desde la última transacción.
46. average_speed: Velocidad promedio de desplazamiento entre transacciones.
47. category_change: Indicador de cambio de categoría entre transacciones consecutivas.
48. part_of_day_evening: Indicador de si la transacción ocurrió en la tarde.
49. part_of_day_morning: Indicador de si la transacción ocurrió en la mañana.
50. part_of_day_night: Indicador de si la transacción ocurrió en la noche.
51. category_entertainment: Indicador de si la categoría predominante es 'entertainment'.
52. category_food_dining: Indicador de si la categoría predominante es 'food_dining'.
53. category_gas_transport: Indicador de si la categoría predominante es 'gas_transport'.
54. category_grocery_net: Indicador de si la categoría predominante es 'grocery_net'.
55. category_grocery_pos: Indicador de si la categoría predominante es 'grocery_pos'.
56. category_health_fitness: Indicador de si la categoría predominante es 'health_fitness'.
57. category_home: Indicador de si la categoría predominante es 'home'.
58. category_kids_pets: Indicador de si la categoría predominante es 'kids_pets'.
59. category_misc_net: Indicador de si la categoría predominante es 'misc_net'.
60. category_misc_pos: Indicador de si la categoría predominante es 'misc_pos'.
61. category_personal_care: Indicador de si la categoría predominante es 'personal_care'.
62. category_shopping_net: Indicador de si la categoría predominante es 'shopping_net'.
63. category_shopping_pos: Indicador de si la categoría predominante es 'shopping_pos'.
64. category_travel: Indicador de si la categoría predominante es 'travel'.

LightGBM:

Descripción de la implementación:

- Se utilizó la biblioteca LightGBM para entrenar un modelo de Gradient Boosting Decision Trees.
- Se implementó un bucle para entrenar el modelo de forma incremental por cada período de tiempo (en este caso, cada 3 meses).
- En cada iteración del bucle, se filtraron los datos para el período de tiempo actual y se entrenó el modelo con esos datos.
- Se usó la función train_lightgbm para entrenar el modelo LightGBM, la cual incluye la creación del dataset de LightGBM, la definición de parámetros del modelo y el entrenamiento con los datos proporcionados.
- Se actualizaron las métricas acumuladas después de cada iteración del bucle.
- Al final, se calcularon las métricas finales promedio y se imprimieron.

Ajustes para habilitar el entrenamiento incremental:

- En cada iteración del bucle, se pasó el modelo entrenado anteriormente como argumento a la función train_lightgbm. Esto permitió que el modelo se actualizara con los nuevos datos en cada período de tiempo, lo que habilitó el entrenamiento incremental.

XGBoost:

Descripción de la implementación:

- Se utilizó la biblioteca XGBoost para entrenar un modelo de Gradient Boosting Decision Trees.
- Se implementó un bucle similar al de LightGBM para entrenar el modelo de forma incremental por cada período de tiempo.
- En cada iteración del bucle, se filtraron los datos para el período de tiempo actual y se entrenó el modelo con esos datos.
- Se utilizó la función train_xgboost para entrenar el modelo XGBoost, la cual incluye la definición de parámetros del modelo y el entrenamiento con los datos proporcionados.
- Se actualizó el modelo en cada iteración del bucle.
- Al final, se calcularon las métricas finales promedio y se imprimieron.

Ajustes para habilitar el entrenamiento incremental:

- Similar al caso de LightGBM, en cada iteración del bucle, se pasó el modelo entrenado anteriormente como argumento a la función train_xgboost. Esto permitió que el modelo se actualizara con los nuevos datos en cada período de tiempo, habilitando así el entrenamiento incremental.

Red Neuronal (Neural Network):

Descripción de la implementación:

- Se utilizó la biblioteca TensorFlow y Keras para construir y entrenar una red neuronal.
- Se implementó un bucle similar a los anteriores para entrenar el modelo de forma incremental por cada período de tiempo.
- En cada iteración del bucle, se filtraron los datos para el período de tiempo actual y se entrenó el modelo con esos datos.
- Se utilizó la función train_neural_network para entrenar la red neuronal, la cual incluye la definición de la arquitectura del modelo, la compilación del modelo con el optimizador y la función de pérdida adecuados, así como el entrenamiento del modelo con los datos proporcionados.
- Se actualizó el modelo en cada iteración del bucle.
- Al final, se calcularon las métricas finales promedio y se imprimieron.

Ajustes para habilitar el entrenamiento incremental:

- En este caso, como se utiliza una red neuronal, no hay una función específica para habilitar el entrenamiento incremental como en el caso de LightGBM y XGBoost. Sin embargo, el modelo se actualiza en cada iteración del bucle, lo que permite que se adapte a los nuevos datos en cada período de tiempo, lo que es esencialmente un entrenamiento incremental.

Análisis de los Resultados de la Evaluación: LightGBM en la Detección de Transacciones Fraudulentas

En este análisis, se comparan los resultados de LightGBM utilizando entrenamiento incremental y entrenamiento total en la detección de transacciones fraudulentas. A continuación, se presenta una evaluación detallada de las métricas de rendimiento y su interpretación en el contexto de la detección de fraudes.

Resultados del Entrenamiento Incremental:

- Accuracy: 0.9891
- Precision: 0.4701
- Recall: 0.8538
- F1 Score: 0.5574
- ROC AUC Score: 0.9804
- Confusion Matrix:
  [[4377888 45072]
  [3333 19459]]

Resultados del Entrenamiento Total:

- Accuracy: 0.9951
- Precision: 0.5137
- Recall: 0.9800
- F1 Score: 0.6741
- ROC AUC Score: 0.9989
- PR AUC Score: 0.9573
- Confusion Matrix:
  [[550227 2643]
  [57 2792]]

Análisis Comparativo:

Precisión (Precision):
Entrenamiento Incremental: 0.4701
Entrenamiento Total: 0.5137
Interpretación: La precisión es la proporción de verdaderos positivos entre los positivos predichos. El entrenamiento total tiene una precisión ligeramente mayor, lo que indica que es más efectivo en minimizar los falsos positivos en comparación con el entrenamiento incremental.

Recall:
Entrenamiento Incremental: 0.8538
Entrenamiento Total: 0.9800
Interpretación: El recall es la proporción de verdaderos positivos entre todos los positivos reales. El entrenamiento total tiene un recall significativamente mayor, lo que indica que es mucho más efectivo en la detección de transacciones fraudulentas (minimiza los falsos negativos) en comparación con el entrenamiento incremental.

F1 Score:
Entrenamiento Incremental: 0.5574
Entrenamiento Total: 0.6741
Interpretación: El F1 Score es la media armónica de la precisión y el recall. El entrenamiento total tiene un F1 Score significativamente mayor, lo que indica un mejor equilibrio entre precisión y recall en la detección de fraudes.

ROC AUC Score:
Entrenamiento Incremental: 0.9804
Entrenamiento Total: 0.9989
Interpretación: El ROC AUC Score mide la capacidad del modelo para distinguir entre clases. Ambos métodos tienen un alto ROC AUC Score, pero el entrenamiento total tiene un valor cercano a 1, lo que indica un rendimiento casi perfecto en la diferenciación entre transacciones legítimas y fraudulentas.

Confusion Matrix:
Entrenamiento Incremental:
[[4377888 45072]
[3333 19459]]
Entrenamiento Total:
[[550227 2643]
[57 2792]]

Interpretación:
Entrenamiento Incremental: Tiene un número mayor de falsos positivos (45072) y falsos negativos (3333) en comparación con el entrenamiento total.
Entrenamiento Total: Tiene un número significativamente menor de falsos positivos (2643) y falsos negativos (57), lo que demuestra su mayor efectividad en la detección precisa de fraudes.

Conclusión:
El análisis de los resultados de LightGBM muestra que el entrenamiento total supera al entrenamiento incremental en todas las métricas clave de rendimiento para la detección de transacciones fraudulentas. El entrenamiento total ofrece una mayor precisión, recall, F1 Score y ROC AUC Score, y, además, presenta una menor cantidad de falsos positivos y falsos negativos en comparación con el entrenamiento incremental. Esto sugiere que, aunque el entrenamiento incremental puede ser útil en ciertos escenarios por su eficiencia computacional, el entrenamiento total es preferible cuando la precisión en la detección de fraudes es crítica.

Análisis de los Resultados de la Evaluación: XGBoost en la Detección de Transacciones Fraudulentas

En este análisis, se comparan los resultados de XGBoost utilizando entrenamiento incremental y entrenamiento total en la detección de transacciones fraudulentas. A continuación, se presenta una evaluación detallada de las métricas de rendimiento y su interpretación en el contexto de la detección de fraudes.

Resultados del Entrenamiento Incremental:

- Accuracy: 0.9983
- Precision: 0.8186
- Recall: 0.8736
- F1 Score: 0.8439
- ROC AUC Score: 0.9974
- Confusion Matrix:
  [[4418479 4481]
  [2882 19910]]

Resultados del Entrenamiento Total:

- Accuracy: 0.9952
- Precision: 0.5163
- Recall: 0.9807
- F1 Score: 0.6764
- ROC AUC Score: 0.9988
- PR AUC Score: 0.9600
- Confusion Matrix:
  [[550252 2618]
  [55 2794]]

Análisis Comparativo

Precisión (Precision):
Entrenamiento Incremental: 0.8186
Entrenamiento Total: 0.5163
Interpretación: La precisión es la proporción de verdaderos positivos entre los positivos predichos. El entrenamiento incremental muestra una precisión significativamente mayor, lo que indica que es más efectivo en minimizar los falsos positivos en comparación con el entrenamiento total.

Recall:
Entrenamiento Incremental: 0.8736
Entrenamiento Total: 0.9807
Interpretación: El recall es la proporción de verdaderos positivos entre todos los positivos reales. El entrenamiento total tiene un recall mayor, lo que indica que es más efectivo en la detección de transacciones fraudulentas (minimiza los falsos negativos) en comparación con el entrenamiento incremental.

F1 Score:
Entrenamiento Incremental: 0.8439
Entrenamiento Total: 0.6764
Interpretación: El F1 Score es la media armónica de la precisión y el recall. El entrenamiento incremental tiene un F1 Score más alto, lo que sugiere un mejor equilibrio entre precisión y recall en la detección de fraudes.

ROC AUC Score:
Entrenamiento Incremental: 0.9974
Entrenamiento Total: 0.9988
Interpretación: El ROC AUC Score mide la capacidad del modelo para distinguir entre clases. Ambos métodos tienen un alto ROC AUC Score, pero el entrenamiento total tiene un valor ligeramente superior, lo que indica un rendimiento marginalmente mejor en la diferenciación entre transacciones legítimas y fraudulentas.

Confusion Matrix:
Entrenamiento Incremental:
[[4418479 4481]
[2882 19910]]
Entrenamiento Total:
[[550252 2618]
[55 2794]]

Interpretación:
Entrenamiento Incremental: Tiene un número mayor de falsos negativos (2882) pero menos falsos positivos (4481) en comparación con el entrenamiento total.
Entrenamiento Total: Tiene un número significativamente menor de falsos negativos (55) pero más falsos positivos (2618), lo que demuestra su mayor efectividad en la detección precisa de fraudes en términos de recall.

Conclusión
El análisis de los resultados de XGBoost muestra que el entrenamiento incremental y el entrenamiento total tienen ventajas en diferentes aspectos para la detección de transacciones fraudulentas.

Entrenamiento Incremental: Destaca en precisión y F1 Score, lo que indica un mejor equilibrio general y una menor tasa de falsos positivos. Esto puede ser ventajoso en escenarios donde los falsos positivos tienen un alto costo.
Entrenamiento Total: Sobresale en recall y ROC AUC Score, indicando una mayor capacidad para identificar todas las transacciones fraudulentas con menos falsos negativos. Esto es crucial en contextos donde es esencial capturar la mayor cantidad de fraudes posible.
En general, la elección entre entrenamiento incremental y entrenamiento total puede depender de las prioridades específicas del sistema de detección de fraudes: si se prefiere minimizar los falsos positivos o maximizar la captura de fraudes.

Análisis de los Resultados de la Evaluación: Redes Neuronales Artificiales (ANN) en la Detección de Transacciones Fraudulentas

En este análisis, se presentan los resultados del modelo de Redes Neuronales Artificiales (ANN) utilizando entrenamiento incremental en la detección de transacciones fraudulentas. A continuación, se ofrece una evaluación detallada de las métricas de rendimiento y su interpretación en el contexto de la detección de fraudes.

Resultados del Entrenamiento Incremental:

- Accuracy: 0.9975
- Precision: 0.8818
- Recall: 0.6006
- F1 Score: 0.7026
- ROC AUC Score: 0.9218
- Confusion Matrix:
  [[4420995 1965]
  [9103 13689]]

Descripción de las Métricas:

Precisión (Precision):
Valor: 0.8818
Interpretación: La precisión es la proporción de verdaderos positivos (transacciones fraudulentas correctamente identificadas) sobre el total de predicciones positivas. Un valor de 0.8818 indica que el 88.18% de las transacciones clasificadas como fraudulentas por el modelo son efectivamente fraudulentas. Esta alta precisión sugiere que el modelo es eficaz en minimizar los falsos positivos, lo cual es crucial en la práctica para evitar alertas innecesarias y costos asociados a la revisión manual de transacciones legítimas marcadas erróneamente como fraudulentas.

Recall:
Valor: 0.6006
Interpretación: El recall es la proporción de verdaderos positivos entre el total de transacciones fraudulentas reales. Un valor de 0.6006 indica que el modelo identifica correctamente el 60.06% de todas las transacciones fraudulentas. Aunque este valor muestra que el modelo captura una cantidad considerable de fraudes, todavía hay un 39.94% de transacciones fraudulentas que no se detectan (falsos negativos), lo cual es un aspecto crítico para mejorar en un sistema de detección de fraudes.

F1 Score:
Valor: 0.7026
Interpretación: El F1 Score es la media armónica de la precisión y el recall. Un valor de 0.7026 refleja un equilibrio entre la precisión y el recall del modelo, proporcionando una visión global de su rendimiento. Este valor sugiere que, aunque la precisión es alta, el compromiso con un recall relativamente menor hace que el F1 Score no sea tan alto como podría ser si ambos valores estuvieran equilibrados.

ROC AUC Score:
Valor: 0.9218
Interpretación: El ROC AUC Score mide la capacidad del modelo para distinguir entre clases (transacciones legítimas y fraudulentas). Un valor de 0.9218 es bastante alto, indicando que el modelo es eficaz en la clasificación general de las transacciones, aunque existe espacio para mejoras en la precisión de la detección de fraudes.

Confusion Matrix:
Valores:
[[4420995 1965]
[9103 13689]]

Interpretación:
Verdaderos Negativos (4420995): Transacciones legítimas correctamente identificadas.
Falsos Positivos (1965): Transacciones legítimas incorrectamente clasificadas como fraudulentas.
Falsos Negativos (9103): Transacciones fraudulentas que no fueron detectadas por el modelo.
Verdaderos Positivos (13689): Transacciones fraudulentas correctamente identificadas.
La matriz muestra que el modelo tiene un buen rendimiento en identificar transacciones legítimas y fraudulentas, pero aún hay un número significativo de transacciones fraudulentas que no se detectan, lo cual es crucial para mejorar.

Conclusión
El análisis de los resultados del modelo ANN utilizando entrenamiento incremental en la detección de transacciones fraudulentas muestra:

- Alta Precisión: Con una precisión de 0.8818, el modelo es muy eficaz en minimizar los falsos positivos, lo que es beneficioso para reducir las alertas falsas y los costos operativos asociados.
- Moderado Recall: Con un recall de 0.6006, el modelo detecta un 60.06% de las transacciones fraudulentas. Aunque es una cantidad significativa, hay un margen considerable para mejorar la detección de fraudes no identificados.
- Equilibrio Representado por F1 Score: El F1 Score de 0.7026 indica un equilibrio razonable entre precisión y recall, pero sugiere la necesidad de mejorar el recall para un rendimiento más robusto.
- Buena Capacidad de Clasificación (ROC AUC Score): El valor de 0.9218 muestra que el modelo tiene una alta capacidad para diferenciar entre transacciones legítimas y fraudulentas en general.

En resumen, el modelo ANN con entrenamiento incremental tiene un rendimiento prometedor, especialmente en términos de precisión, pero requiere mejoras en el recall para ser más eficaz en la detección completa de fraudes. Esto podría involucrar técnicas adicionales de ajuste de hiperparámetros, mayor entrenamiento, o la integración de estrategias complementarias como el uso de ensamblajes de modelos para mejorar la detección de fraudes.

Conclusiones

Este proyecto ha explorado la viabilidad del entrenamiento incremental en modelos de aprendizaje automático y profundo, con un enfoque en la detección de transacciones fraudulentas en un dataset de transacciones de tarjeta de crédito. Los modelos investigados incluyeron LightGBM, XGBoost y Redes Neuronales Artificiales (ANN).

Los resultados indican que LightGBM y XGBoost muestran un rendimiento considerablemente mejor con el entrenamiento total en comparación con el incremental, especialmente en términos de precisión, recall y F1 Score. Sin embargo, XGBoost demostró un alto rendimiento tanto en el entrenamiento incremental como en el total, con una ligera ventaja en términos de recall y F1 Score para el entrenamiento total. En contraste, el modelo ANN entrenado de manera incremental mostró una alta precisión, pero su recall fue moderado, indicando que el modelo fue eficaz en minimizar falsos positivos, pero no tan eficiente en capturar todas las instancias de fraude.

Se observó que el entrenamiento total generalmente proporciona un mejor rendimiento debido a que los modelos pueden captar mejor la distribución completa de los datos. El entrenamiento incremental, aunque útil para escenarios con limitaciones de tiempo y recursos, puede beneficiarse de ajustes y técnicas adicionales para acercarse al rendimiento del entrenamiento total. La creación de nuevas características relevantes y el balanceo adecuado de las clases mediante técnicas como SMOTE son componentes críticos para mejorar la detección de fraudes. El análisis exploratorio de datos (EDA) y la normalización de características también jugaron un papel importante en el rendimiento de los modelos.

En conclusión, este proyecto ha demostrado que el entrenamiento incremental puede ser una opción viable para la detección de fraudes en transacciones de tarjeta de crédito, aunque con ciertas limitaciones en comparación con el entrenamiento total. La aplicación de técnicas de balanceo como SMOTE fue fundamental para abordar el desafío de los datos desequilibrados y mejorar el rendimiento general de los modelos. En resumen, la elección entre entrenamiento incremental y total debe basarse en las necesidades específicas del contexto, considerando factores como el tiempo, los recursos y la variabilidad de los datos.

Bibliografía

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

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from Imbalanced Data Sets. Springer.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, 18, 1-5.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. ACM Computing Surveys (CSUR), 46(4), 1-37.

Žliobaitė, I., Bifet, A., Pfahringer, B., & Holmes, G. (2011). Active learning with drifting streaming data. IEEE Transactions on Neural Networks and Learning Systems, 23(1), 77-90.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. Machine Learning, 23(1), 69-101.

Luo, P., Zhang, L., & Yu, P. S. (2016). Real-time learning from data streams. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1961-1970).

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. Progress in Artificial Intelligence, 5(4), 221-232.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). The Elements of Statistical Learning. Springer Series in Statistics. Springer, New York, NY.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems (pp. 3146-3154).

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Bifet, A., Holmes, G., Pfahringer, B., & Gavaldà, R. (2011). Detecting sentiment change in twitter streaming data. Journal of Machine Learning Research, 12(2011), 1589-1608.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering, 31(12), 2346-2363.