Resolución de la Hoja de Ejercicio del Tema 6

Sergio Mínguez Cruces

29 de noviembre de 2024

Índice

1. Introducción **3**

2. Explicación del código utilizado para la resolución del problema **3**
2.1. Cargar librerías y datos 3
2.2. Exploración de los datos 4
2.3. Preprocesamiento 5
2.4. Reducción de dimensionalidad 6
2.5. Entrenamiento y evaluación del modelo 7

3. Análisis Exploratorio **12**
3.1. Distribución de las Clases (Fraude/No Fraude) 12
3.2. Principales Observaciones sobre las variables 12
3.2.1. Variables Anonimas 12
3.2.2. Escala de las variables 12
3.2.3. Correlación entre variables 13

4. Metodología **13**
4.1. Frequent Directions: Descripción Breve 13
4.2. Aplicación al Conjunto de Datos 13

5. Resultados **14**
5.1. Explicación de los Resultados del Modelo Predictivo 14
5.2. Explicación de los Resultados del Modelo Predictivo con Frequent Directions 17

6. Conclusiones **18**
6.1. Reflexión sobre la Utilidad de las Matrices de Sketching en este Caso 18
6.2. Sugerencias para Mejorar el Enfoque 19

1. Introducción

En el contexto del análisis de datos masivos, la reducción de dimensionalidad es una técnica clave para optimizar el rendimiento de los modelos de aprendizaje automático y gestionar de manera eficiente grandes volúmenes de datos. Este trabajo se enfoca en aplicar *Frequent Directions*, un algoritmo de matrices de *sketching*, al problema de detección de fraudes en transacciones bancarias, utilizando un conjunto de datos público disponible en Kaggle: *Credit Card Fraud Detection*.

El conjunto de datos contiene variables tipo numéricas altamente correlacionadas, lo que lo convierte en una base muy útil para explorar el impacto de la reducción de dimensionalidad. Cada fila representa una transacción bancaria y está etiquetada como fraude o no fraude, formando un problema de clasificación supervisada binaria.

El objetivo principal de este trabajo es implementar el algoritmo *Frequent Directions* para reducir la dimensionalidad de los datos y comparar su impacto en el rendimiento de un modelo de clasificación, como la regresión logística. A través de este análisis, se busca evaluar cómo la reducción de dimensiones afecta métricas clave como la precisión, el *recall* y el F1-score, proporcionando información valiosa sobre la eficacia de las técnicas de *sketching* en aplicaciones prácticas.

2. Explicación del código utilizado para la resolución del problema

A continuación, se explicará el código junto con las gráficas resultantes de ejecutarlo. Cada parte del código irá acompañada de una breve explicación para indicar que hace cada parte y los resultados mostrados. Este apartado servirá como introducción al análisis, dejando un estudio más detallado y profundo para las siguientes secciones.

2.1. Cargar librerías y datos

El código comienza cargando las librerías necesarias: `numpy`, `pandas`, `sklearn`, `time` y `matplotlib`. Estas librerías serán utilizadas para manipular datos, aplicar normalización, calculo del tiempo transcurrido y entrenar modelos.

```
1 import numpy as np
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.metrics import classification_report,
    accuracy_score
```

```
from sklearn.utils.extmath import randomized_svd
import matplotlib.pyplot as plt
import time

np.random.seed(42)
data = pd.read_csv("ruta de acceso")
```

2.2. Exploración de los datos

Se realiza una exploración inicial del conjunto de datos:

- Se calcula la frecuencia absoluta y relativa de transacciones fraudulentas (`Class=1`) y no fraudulentas (`Class=0`).
- Se genera una tabla y un gráfico de barras con la distribución, al estar tan desbalaceadas se usará una escala logarítmica para así poder comparar mejor las frecuencias de cada clase.

```
# Frecuencias absolutas y relativas
frequencies_absolute = data['Class'].value_counts()
frequencies_relative = data['Class'].value_counts(normalize=
    True) * 100

# Tabla de frecuencias
print("Clase | Frecuencia Absoluta | Frecuencia Relativa (%)
    ")
for clase in frequencies_absolute.index:
    print(f"{clase:^5} | {frequencies_absolute[clase]:^19} |
        {frequencies_relative[clase]:^23.2f}")

# Gr fico de barras
plt.figure(figsize=(8, 6))
clases = ['No Fraude (0)', 'Fraude (1)']
plt.bar(clases, frequencies_absolute, color=['skyblue', '
    orange'], edgecolor='k', log=True)
plt.title("Frecuencia Absoluta (Escala Logar tmica)")
plt.ylabel("Frecuencia Absoluta (Log)")
plt.xlabel("Clase")
plt.show()
```

```
Tabla de Frecuencias de la Variable 'Class':
------------------------------------------------
Clase | Frecuencia Absoluta | Frecuencia Relativa (%)
------------------------------------------------
  0   |       284315        |         99.83
  1   |         492         |         0.17
------------------------------------------------
```

Figura 1: Enter Caption

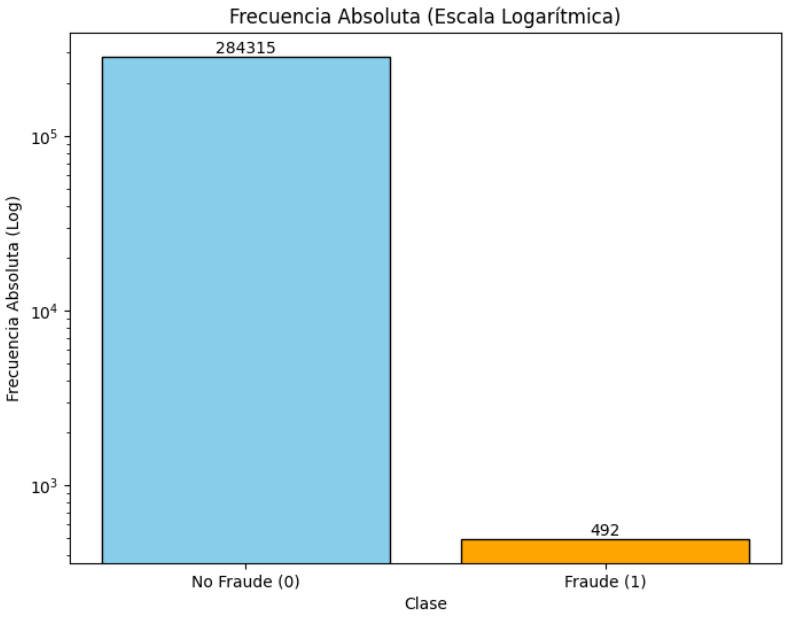

Figura 2: Enter Caption

2.3. Preprocesamiento

Las variables se normalizan usando `StandardScaler()` $z = \frac{x-\mu}{\sigma}$ y los datos se dividen en conjuntos de entrenamiento y prueba respetando la proporción original de las clases (80-20 %).

No se normaliza la variable `Class` porque al hacerlo emperoraban las métricas ya que a la regresión logistica se le dificulta la clasificación al ser valores muy próximos.

```
# Se separan la variable objetivo del resto de la base
X = data.drop(columns=['Class'])
y = data['Class']

# Normalizacion
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Divisi n en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y)
```

2.4. Reducción de dimensionalidad

Se utiliza la técnica *Frequent Directions*, implementada mediante SVD randomized, esta es una técnica de muestreo aleatorio para aproximar a una matriz de menor rango que conserva las propiedades más relevantes de la matriz original. Esto se logra proyectando la matriz en un subespacio de dimensiones reducidas antes de realizar la descomposición.

Esta técnica se usa para reducir las dimensiones de la matriz de 30 a 10 como se indicó.

```
# Aplicar SVD aleatorizado
n_components = 10
U, Sigma, VT = randomized_svd(X_train, n_components=
    n_components, random_state=42)

# Reducci n de dimensionalidad
X_train_sketch = np.dot(U, np.diag(Sigma))
X_test_sketch = np.dot(X_test, VT.T)

# Comparaci n de dimensionalidades
print(f"Dimensiones originales: {X_train.shape}")
print(f"Dimensiones reducidas: {X_train_sketch.shape}")
```

```
Dimensiones originales: (227845, 30)
Dimensiones reducidas: (227845, 10)
```

Figura 3: Enter Caption

Se hará un gráfico de dispersión para representar los datos transformados.

En este espacio reducido, los datos se proyectan sobre dos componentes principales (dos nuevas variables creadas durante el Frequent Directions. Estas variables resumen la información más importante de los datos originales, pero en un espacio reducido en este caso dos dimensiones), renombradas como la "Primera dirección principal"(eje X) y la "Segunda dirección principal"(eje Y).

Los puntos en el gráfico están coloreados según la variable objetivo (Class), que indica si una transacción es fraudulenta (rojo) o no (azul).

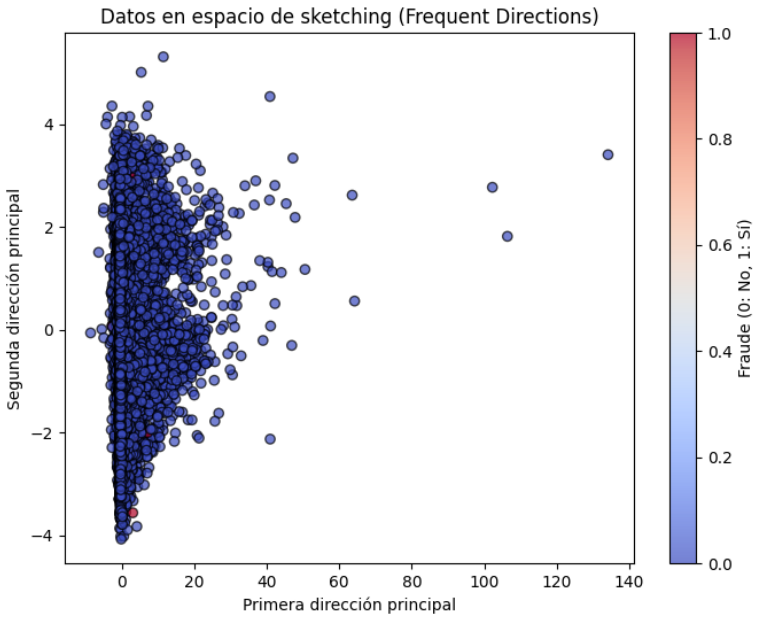

Figura 4: Enter Caption

Como se puede apreciar en el gráfico el desbalanceo de las clases es evidentes, concentrandose en los valores bajos del eje x y aproximandose a 0 en eje y según aumenta la x, como se puede apreciar en la nube; los valores fuera de la nube podrian considerarse en su mayoría datos atípicos por lo que no se deberían tener en cuenta.

Se puede observar que las pocas operaciones fraudulentas se situan en los valores inferiores a x igual a 10 pero independientes sobre el eje y.

2.5. Entrenamiento y evaluación del modelo

Se entrenan dos modelos de regresión logística, el primero con las variables originales y el segundo con las variables reducidas.

Se escoge este tipo de modelo ya que está específicamente diseñado para clasificar en variables objetivo binaria, como es la variable Class, sumadoa que es un modelo lineal fácil de implementar y entrenar, con un bajo costo computacional.

Se evalúan ambos modelos mediante métricas como *accuracy*, *recall* y *F1-score*, además de medir el tiempo de entrenamiento.

```python
# Modelo con datos originales
start_time = time.time()
model_original = LogisticRegression(random_state=42)
model_original.fit(X_train, y_train)
y_pred_original = model_original.predict(X_test)
end_time = time.time()

# Reporte y tiempo
print("Resultados con datos originales:")
print(classification_report(y_test, y_pred_original))
print(f"Tiempo de entrenamiento y predicci n: {end_time -
    start_time:.4f} segundos")
```

```
Resultados con datos originales:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.83      0.64      0.72        98

    accuracy                           1.00     56962
   macro avg       0.91      0.82      0.86     56962
weighted avg       1.00      1.00      1.00     56962

Tiempo de entrenamiento y predicción con datos originales: 0.4086 segundos
```

Figura 5: Enter Caption

```python
# Modelo con datos reducidos
start_time2 = time.time()
model_sketch = LogisticRegression(random_state=42)
model_sketch.fit(X_train_sketch, y_train)
y_pred_sketch = model_sketch.predict(X_test_sketch)
end_time2 = time.time()

# Reporte y tiempo
print("Resultados con Frequent Directions:")
print(classification_report(y_test, y_pred_sketch))
print(f"Tiempo de entrenamiento y predicci n: {end_time2 -
    start_time2:.4f} segundos")
```

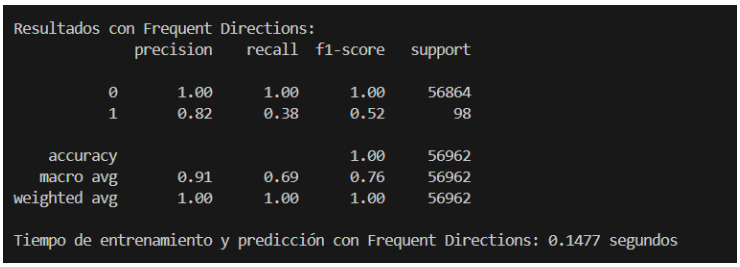

Figura 6: Enter Caption

Posteriormente se generará genera un gráfico de barras que compara el rendimiento de los dos modelos con las métricas más importantes, en este caso solo se tendrá encuenta *accuracy* (precisión general del modelo), la *precision*, *recall* y *f1-score* de la clase 1 ya que los valores de la clase 0 son todos 1.

Volvemos a calcular las métricas para cada modelo usando `classification_report` con la opción `output_dict=True`, que devuelve los resultados como un diccionario, esto se hace porque en forma de diccionario es más fácil la obtención de los datos y ya que los valores de las métricas no cambian.

```
clasificacion_sketch = classification_report(y_test,
    y_pred_sketch, output_dict=True)
clasificacion_original = classification_report(y_test,
    y_pred_original, output_dict=True)
```

Se define una lista con las métricas principales y se extraen los valores correspondientes de los diccionarios generados previamente.

```
metrics = ['accuracy', 'precision', 'recall', 'f1-score']
original_scores = [
    clasificacion_original['accuracy'],
    clasificacion_original['1']['precision'],
    clasificacion_original['1']['recall'],
    clasificacion_original['1']['f1-score']
]
sketch_scores = [
    clasificacion_sketch['accuracy'],
    clasificacion_sketch['1']['precision'],
    clasificacion_sketch['1']['recall'],
    clasificacion_sketch['1']['f1-score']
]
```

- `metrics`: Lista las métricas a evaluar.
- `original_scores`: Es una lista que contiene los valores de las métricas del modelo entrenado con características originales.

- `sketch_scores`: Es una lista que contiene los valores de las métricas del modelo entrenado con características reducidas.

Se grafican las barras para los valores de las métricas de ambos modelos.

```
fig, ax = plt.subplots(figsize=(10, 6))
bars1 = ax.bar(x - width/2, original_scores, width, label='
    Original', color='blue')
bars2 = ax.bar(x + width/2, sketch_scores, width, label='
    Frequent Directions', color='orange')
```

Para que se puedan observar las dos métricas en el mismo gráfico se utiliza:

- `x - width/2`: Desplaza las barras del modelo original hacia la izquierda.
- `x + width/2`: Desplaza las barras del modelo reducido hacia la derecha.

Se añaden etiquetas encima de cada barra para mostrar los valores exactos de cada métrica.

```
for bar in bars1:
    ax.text(bar.get_x() + bar.get_width()/2, bar.get_height
        (), f'{bar.get_height():.2f}', ha='center', va='
        bottom')
for bar in bars2:
    ax.text(bar.get_x() + bar.get_width()/2, bar.get_height
        (), f'{bar.get_height():.2f}', ha='center', va='
        bottom')
```

Se usa un bucle para recorrer cada barra, calculando la posición del texto basado en la posición y altura de la barra para que quede encima de la barra correspondiente y se redondea a dos decimales (`:.2f`).

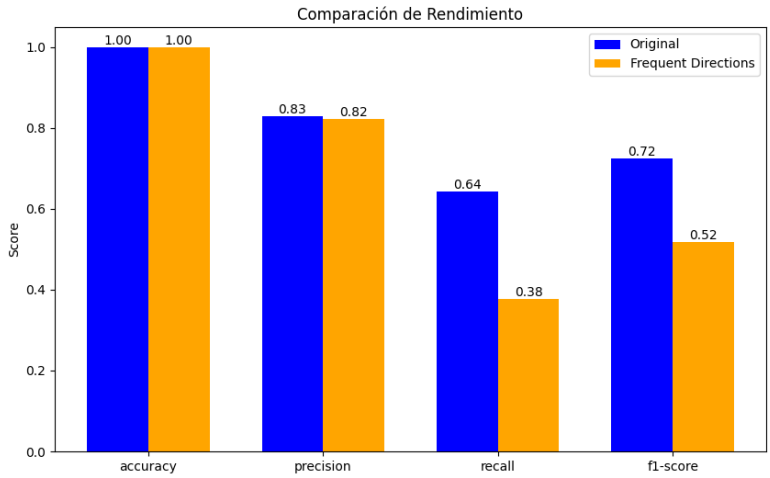

Figura 7: Enter Caption

El gráfico permite comparar visualmente las métricas principales entre ambos modelos. Las barras muestran diferencias en *precision*, *recall* y *f1-score*, proporcionando una visión clara del impacto de *Frequent Directions*.

Para hacer la tabla comparativa se extraen los valores del diccionario correspondiente junto con el tiempo, para poder comparar el rendimiento métrico junto con el tiempo necesario para el entrenamiento.

```
# Tabla comparativa
print("\nTabla Comparativa de Resultados:")
print("-------------------------------------------------")
print(f"{'Modelo':^25} | {'Tiempo de Ejecuci n (s)':^20} |
    {'Accuracy':^10} | {'Precision':^10} | {'Recall':^10} |
    {'F1-Score':^10}")
print("-------------------------------------------------")

# Mostrar los resultados para el modelo original
print(f"{'Original':^25} | {end_time - start_time:^23.4f} |
    {original_scores[0]:^10.4f} | {original_scores[1]:^10.4f}
     | {original_scores[2]:^10.4f} | {original_scores
    [3]:^10.4f}")
# Mostrar los resultados para el modelo reducido (Frequent
    Directions)
print(f"{'Frequent Directions':^25} | {end_time2 -
    start_time2:^23.4f} | {sketch_scores[0]:^10.4f} | {
    sketch_scores[1]:^10.4f} | {sketch_scores[2]:^10.4f} | {
    sketch_scores[3]:^10.4f}")
print("-------------------------------------------------")
```

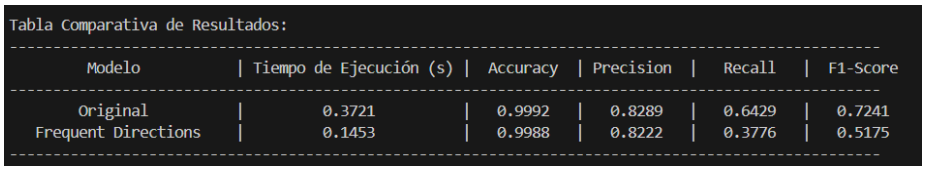

Tabla Comparativa de Resultados:

Modelo	Tiempo de Ejecución (s)	Accuracy	Precision	Recall	F1-Score
Original	0.3721	0.9992	0.8289	0.6429	0.7241
Frequent Directions	0.1453	0.9988	0.8222	0.3776	0.5175

Figura 8: Enter Caption

3. Análisis Exploratorio

3.1. Distribución de las Clases (Fraude/No Fraude)

La variable `Class` está altamente desbalanceada, con una clara mayoría de transacciones clasificadas como *No Fraude* (clase 0) y una minoría significativa de transacciones marcadas como *Fraude* (clase 1):

- **Clase 0 (No Fraude):** 284,315 observaciones, representando el 99.83 % de los datos.
- **Clase 1 (Fraude):** 492 observaciones, apenas el 0.17 % del total.

Esta desproporción sugiere que el problema requiere técnicas especializadas para manejar el desbalanceo, tales como:

- Submuestreo de la clase mayoritaria.
- Sobremuestreo de la clase minoritaria (e.g., SMOTE).
- Uso de métricas como el F1-Score, que son más adecuadas para problemas con clases desbalanceadas.

3.2. Principales Observaciones sobre las variables

3.2.1. Variables Anonimas

La basede datos contiene 30 variables numéricas anonimas (`Time`, `V1`, `V2`, ..., `V28` y `Amount`), además de la etiqueta `Class`. Estas variables han sido transformadas mediante técnicas de PCA (Análisis Componentes Principales) como se indica en la descripción de la pagina Kaggle.

3.2.2. Escala de las variables

Las variables parecen estar en diferentes rangos (principalmente `Time` y `Amount`), lo que sugiere la necesidad de una normalización o estandarización de los datos antes de aplicar modelos de clasificación, especialmente aquellos sensibles a la escala, como la regresión logística ya que una variable con valor elevado podria afectar al rendimiento del modelo.

3.2.3. Correlación entre variables

Según la descripción inicial, las variables presentan alta correlación entre sí, lo que refuerza la necesidad de técnicas de reducción de dimensionalidad, como el método de Frequent Directions, para eliminar redundancias y así reducir el tiempo computacional.

4. Metodología

4.1. Frequent Directions: Descripción Breve

Frequent Directions es un algoritmo eficiente para la reducción de dimensionalidad en matrices de datos de alta dimensión. Este método se basa en proyectar los datos en un espacio de menor dimensión mientras preserva las propiedades clave de la matriz original, como la estructura de las relaciones entre las variables. Frequent Directions utiliza técnicas de descomposición matricial, similares al análisis de componentes principales (PCA), pero se destaca por ser computacionalmente más eficiente y escalable.

La esencia del algoritmo es reducir la dimensión de una matriz $A \in \mathbb{R}^{n \times d}$ (con n observaciones y d variables) a una matriz comprimida $B \in \mathbb{R}^{n \times \ell}$, donde ℓ es mucho menor que d. Esto se logra manteniendo los vectores singulares más importantes de A, minimizando así la pérdida de información.

4.2. Aplicación al Conjunto de Datos

El algoritmo Frequent Directions se aplicó al conjunto de datos de detección de fraude en transacciones bancarias con el objetivo de reducir la dimensionalidad de las variables anonimizadas. El proceso se llevó a cabo en las siguientes etapas:

1. **Preprocesamiento de los datos:**
   Antes de aplicar Frequent Directions, se normalizaron las variables para garantizar que todas estuvieran en la misma escala.

2. **Construcción de la matriz de datos:**
   Se organizó el conjunto de datos como una matriz $A \in \mathbb{R}^{n \times d}$, donde $n = 284807$ corresponde al número de transacciones, y $d = 30$ es el número de variables originales.

3. **Reducción de dimensionalidad con Frequent Directions:**
   Frequent Directions se aplicó a la matriz A, seleccionando un parámetro ℓ (dimensión comprimida en este caso 10) adecuado para conservar una proporción significativa de la información original.

4. **Transformación de los datos:**
   Tras ejecutar el algoritmo, se generó una nueva matriz $B \in \mathbb{R}^{n \times \ell}$, donde

$\ell \ll d$. Esta matriz comprimida sirvió como base para el entrenamiento del modelo de clasificación supervisada.

5. **Evaluación del impacto:**
   Se entrenaron modelos de clasificación (regresión logística en este caso) tanto con el conjunto de datos original como con la versión comprimida. Posteriormente, se compararon las métricas de rendimiento (precisión, recall, F1-score) para evaluar el impacto de la reducción de dimensionalidad en la capacidad predictiva del modelo.

Al aplicar Frequent Directions se espera reducir la dimensionalidad del conjunto de datos manteniendo las relaciones clave entre las variables. Esto nsolo mejorará la eficiencia computacional del modelo de clasificación, sino que también destacará las variables más relevantes para la detección de fraudes sin perder prácticamente precisión.

5. Resultados

5.1. Explicación de los Resultados del Modelo Predictivo

1. Métricas principales

Para evaluar el desempeño del modelo, se han utilizado tres métricas principales: **precisión**, **recall** y **F1-score**, que se calculan para cada clase del problema (en este caso, las clases 0 y 1).

Precisión: La precisión indica la fracción de predicciones positivas correctas en relación con todas las predicciones positivas realizadas. Es decir, entre todas las veces que el modelo predijo la clase i, ¿cuántas veces fue correcta?

$$\text{Precisión}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}$$

Donde:

- TP_i = Verdaderos positivos para la clase i
- FP_i = Falsos positivos para la clase i
- Para la clase 0, la precisión es **1.00**, lo que significa que todas las predicciones de clase 0 fueron correctas (no hubo falsos positivos).
- Para la clase 1, la precisión es **0.83**, lo que indica que, de todas las predicciones de clase 1, el 83 % fueron correctas.

Siendo ambos valores óptimos.

Recall: El recall, también conocido como sensibilidad o tasa de verdaderos positivos, mide la fracción de casos positivos reales que el modelo fue capaz de identificar correctamente.

$$\text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}$$

Donde:

- FN_i = Falsos negativos para la clase i
- Para la clase 0, el recall es **1.00**, lo que significa que el modelo identificó correctamente todos los casos de la clase 0 (sin falsos negativos).
- Para la clase 1, el recall es **0.64**, lo que indica que el modelo fue capaz de identificar solo el 64 % de las instancias reales de la clase 1. Esto sugiere que el modelo no está identificando correctamente todos los casos de la clase minoritaria.

F1-Score: El F1-Score es la métrica que utiliza la precisión y el recall. Es una métrica que equilibra estos dos valores y es útil cuando las clases están desequilibradas como es esta base de datos.

$$\text{F1}_i = 2 \cdot \frac{\text{Precisión}_i \cdot \text{Recall}_i}{\text{Precisión}_i + \text{Recall}_i}$$

- Para la clase 0, el F1-Score es **1.00**, lo que refleja el excelente desempeño del modelo en la predicción de la clase mayoritaria.
- Para la clase 1, el F1-Score es **0.72**, lo que es razonable, pero no ideal, debido al bajo recall observado. Esto indica que, aunque el modelo tiene una buena precisión al predecir la clase 1, debería mejorar en identificar más instancias de esta clase.

2. Métricas Promedio

Macro Promedio: El promedio macro calcula el promedio de las métricas de precisión, recall y F1-Score de manera independientemente del peso de las clases. En este caso, se obtiene un **precision promedio de 0.91**, **recall promedio de 0.82** y **F1-Score promedio de 0.86**. Esto muestra un buen rendimiento global del modelo, pero al tener unas clases tan desbalanceadas no podemos obviar esas diferencia de peso, por lo que no sería recomendable usar está métrica para validar el modelo.

$$\text{Precisión Macro} = \frac{1}{N} \sum_{i=1}^{N} \text{Precisión}_i$$

$$\text{Recall Macro} = \frac{1}{N} \sum_{i=1}^{N} \text{Recall}_i$$

$$\text{F1 Macro} = \frac{1}{N} \sum_{i=1}^{N} \text{F1}_i$$

Donde N es el número de clases.

Promedio Ponderado: El promedio ponderado toma en cuenta la cantidad de muestras de cada clase y calcula la media ponderada de las métricas. Aquí, el promedio ponderado de **precisión**, **recall** y **F1-Score** son todos **1.00**, lo que indica que el modelo tiene un desempeño excelente cuando se considera el desequilibrio de clases.

$$\text{Precisión Ponderada} = \frac{\sum_{i=1}^{N}(\text{Precisión}_i \cdot \text{N}_i)}{\sum_{i=1}^{N}\text{N}_i}$$

$$\text{Recall Ponderado} = \frac{\sum_{i=1}^{N}(\text{Recall}_i \cdot \text{N}_i)}{\sum_{i=1}^{N}\text{N}_i}$$

$$\text{F1 Ponderado} = \frac{\sum_{i=1}^{N}(\text{F1}_i \cdot \text{N}_i)}{\sum_{i=1}^{N}\text{N}_i}$$

Donde:

- N_i es el número de instancias de la clase i.

Esto se debe a que clasifica tan bien la clase 0 y es tan abrumadoramente mayoritaria que al incluir el peso de las clases hace que sea indiferente los valores de la clase 1.

3. Precisión general

La **accuracy** del modelo es **1.00**. Esto significa que, en conjunto, el modelo hizo predicciones correctas en casi todos los casos. Sin embargo, sabiendo lo desbalanceadas que están las clases este dato no arrojaría mucha información ya que al tener un recall de 1 en la clase mayoritaria (0) provoca que el accuracy sea 1 independientemente del recall de la clase minoritaria (1).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TTP}}$$

Donde:

- TP = Verdaderos positivos
- TN = Verdaderos negativos
- TTP = Total de predicciones

4. Tiempo de Entrenamiento y Predicción

El tiempo de entrenamiento y predicción del modelo fue de **0.4086 segundos**. Este tiempo es relativamente corto, lo que indica que el modelo se entrenó y realizó las predicciones rápidamente.

5. Observaciones Finales

El modelo tiene un excelente desempeño en la clase mayoritaria (0), pero la clase minoritaria (1) está siendo menos identificada, como se refleja en el recall y F1-Score de la clase 1 debido a la distribución desbalanceada de las clases.

5.2. Explicación de los Resultados del Modelo Predictivo con Frequent Directions

Dado que ya se ha explicado el significado de cada variable, no se repetirá la explicación.

1. Métricas principales

Precisión:

- Para la clase 0, la precisión es **1.00**, lo que significa que todas las predicciones de clase 0 fueron correctas (no hubo falsos positivos).
- Para la clase 1, la precisión es **0.82**, lo que indica que, de todas las predicciones de clase 1, el 82 % fueron correctas.

Ambos valores son razonablemente buenos.

Recall:

- Para la clase 0, el recall es **1.00**, lo que significa que el modelo identificó correctamente todos los casos de la clase 0 (sin falsos negativos).
- Para la clase 1, el recall es **0.38**, lo que indica que el modelo solo identificó el 38 % de las instancias reales de la clase 1. Esto sugiere que el modelo tiene grandes dificultades para detectar correctamente la clase minoritaria, lo cual es un claro impedimento a la hora de valorar positivamente el modelo.

F1-Score:

- Para la clase 0, el F1-Score es **1.00**, lo que refleja el excelente desempeño del modelo en la predicción de la clase mayoritaria.
- Para la clase 1, el F1-Score es **0.52**, lo que indica un rendimiento moderado bajo. Aunque la precisión para la clase 1 es alta, el recall bajo sugiere que el modelo debería mejorar en la identificación de más instancias de esta clase.

2. Métricas Promedio

Macro Promedio: El **precision promedio** es **0.91**, el **recall promedio** es **0.69**, y el **F1-Score promedio** es **0.76**. Esto muestra un buen rendimiento global del modelo, pero el recall promedio indica que el modelo tiene dificultades

para identificar correctamente las instancias de la clase minoritaria, debido al desbalance de clases.

Promedio Ponderado: El promedio ponderado de **precisión**, **recall** y **F1-Score** son todos **1.00**, lo que sugiere que, cuando se tiene en cuenta la distribución de clases, el modelo tiene un desempeño excelente en general, especialmente para la clase mayoritaria (0).

3. Precisión general

La **accuracy** del modelo es **1.00**. Este valor refleja que el modelo hizo predicciones correctas en casi todos los casos. Sin embargo, dado el fuerte desbalance de clases (la clase 0 representa más del 99 % de los datos), la alta accuracy se debe en gran parte a que el modelo predice correctamente la clase mayoritaria en casi todos los casos. Esto hace que la accuracy no sea una métrica suficiente para evaluar el rendimiento del modelo, ya que no refleja el desempeño en la clase minoritaria.

4. Tiempo de Entrenamiento y Predicción

El tiempo de entrenamiento y predicción del modelo fue de **0.1477 segundos**, lo que indica que el modelo se entrenó y realizó las predicciones rápidamente, siendo una mejora significativa respecto al tiempo anterior. Esto sugiere que el uso de *Frequent Directions* ha optimizado el proceso de entrenamiento.

5. Observaciones Finales

El modelo presenta un excelente desempeño en la clase mayoritaria (0), con una precisión, recall y F1-score perfectos. Sin embargo, la clase minoritaria (1) no se identifica adecuadamente , como se refleja en el bajo recall. Aunque la precisión para la clase 1 es razonablemente alta (82 %), el modelo tiene dificultades para identificar instancias de la clase minoritaria, lo que puede ser problemático. Las métricas ponderadas de **1.00** reflejan el excelente rendimiento general debido a la gran diferencia de pesos, pero se debe considerar mejorar el recall y el F1-score de la clase minoritaria para obtener un modelo más robusto en situaciones con un desbalanceo de clases menor.

6. Conclusiones

6.1. Reflexión sobre la Utilidad de las Matrices de Sketching en este Caso

El uso de la técnica de reducción de dimensionalidad basada en matrices de sketching, como *Frequent Directions*, ha demostrado ser útil en términos de reducción del tiempo de ejecución del modelo. Al aplicar *Frequent Directions* sobre los datos, se logró una reducción significativa en el tiempo de entrenamiento

y predicción, pasando de 0.4086 segundos con los datos originales a 0.1477 segundos.

Sin embargo, aunque el modelo entrenado con *Frequent Directions* mantuvo una precisión similar a la de los datos originales (con una precisión de 0.82 frente a 0.83 para la clase 1, fraude), hubo una disminución notable en el recall (de 0.64 a 0.38) y el F1-Score (de 0.72 a 0.52). Esto sugiere que la reducción de dimensionalidad, aunque mejora la eficiencia computacional, tiene un impacto negativo en la capacidad del modelo para identificar correctamente las transacciones fraudulentas. La baja tasa de recall con *Frequent Directions* indica que el modelo podría estar perdiendo una cantidad significativa de transacciones fraudulentas, lo cual es el aspecto más importante cuando se trata de la detección de fraudes.

Por lo tanto, si bien las matrices de sketching como *Frequent Directions* pueden ser útiles para acelerar el procesamiento, su efectividad en escenarios de alta importancia como la detección de fraudes, donde la precisión y el recall son fundamentales, debe ser cuidadosamente evaluada y mejorada.

6.2. Sugerencias para Mejorar el Enfoque

A continuación, se presentan algunas sugerencias para mejorar el rendimiento del modelo en este tipo de problemas:

- **Sobremuestreo de la Clase Minoritaria (Fraude):** Dado que el conjunto de datos es altamente desbalanceado, con una gran mayoría de transacciones no fraudulentas y un número reducido de fraudes, se recomienda aplicar técnicas de sobremuestreo, por estudios visto el más utilizado es SMOTE (Synthetic Minority Over-sampling Technique), para equilibrar las clases y mejorar el recall. Esto podría ayudar al modelo a identificar mejor las transacciones fraudulentas.
- **Uso de Modelos Específicos para Datos Desbalanceados:** En este estudio se ha utilizado una regresión logística, se recomienda utilizar modelos que estén diseñados para manejar clases desbalanceadas, como el *Random Forest*, que permiten ajustar pesos en las clases para evitar que el modelo se sesgue hacia la clase mayoritaria (No Fraude).
- **Optimización de Hiperparámetros:** Si se continua con la regresión logística se recominda realizar una optimización de hiperparámetros esto podría ayudar a mejorar tanto el recall como el F1-Score, las dos métricas más bajas.
- **Mejora de la Reducción de Dimensionalidad:** Si bien *Frequent Directions* muestra ventajas en términos de tiempo, la pérdida de rendimiento en la detección de fraudes sugiere que se debería aumentar la dimensión para obtener una mayor capacidad para distinguir entre las clases.

- **Normalización de las Variables:** Si bien en este estudio se ha utilizado StandardScaler() para normalizar, utilizar otro método de normalización (Min-Max, Robust Scaling o Quantile Transformation) podría resultar beneficioso a la hora de que el modelo detecte los fraudes.

En resumen, aunque la reducción de dimensionalidad mediante *Frequent Directions* permite una mejora sustancial en el tiempo de ejecución, es necesario seguir mejorando el modelo para garantizar que la capacidad de detectar fraudes no se vea comprometida.