Universidad Católica San Pablo

Laboratorio: Regresion Lineal

Computer Science — Tópicos en Inteligencia Artificial

Harold Alejandro Villanueva Borda

1. Introducción

La regresión lineal es un método de inteligencia artificial que predice valores continuos al encontrar una relación lineal entre variables. Se usa para modelar y predecir tendencias en datos, como precios o comportamientos futuros. Este informe se implementa una solución para el análisis del dataset escogido para realizar la práctica de laboratorio, en donde se aplican diferentes conceptos de regresión lineal.

2. Dataset

El conjunto de datos utilizado es *Credit Card Fraud Detection Dataset 2023*, extraído del repositorio Kaggle [1]. Este conjunto de datos contiene transacciones con tarjeta de crédito realizadas por titulares de tarjetas europeos en el año 2023. Comprende más de 550.000 registros.

2.1. Características Clave

- id: Identificador único para cada transacción
- V1-V28: Características anónimas que representan varios atributos de transacción (por ejemplo, hora, ubicación, etc.)
- Cantidad: El monto de la transacción
- Clase: Etiqueta binaria que indica si la transacción es fraudulenta (1) o no (0)

3. Implementación

A continuación se muestran las principales funciones implementadas en Python usando Numpy y Pandas.

```
# Lectura de datos
def leer_datos(archivo):
    return pd.read_csv(archivo)

# Normalizaci n usando Min-Max
def normalizar_min_max(df):
    return (df - df.min()) / (df.max() - df.min())

# Divisi n de los datos
def dividir_datos(df, train_ratio=0.7):
    n = len(df)
    train_size = int(n * train_ratio)
    return df.iloc[:train_size], df.iloc[train_size:]

# Funci n de costo (Error Cuadr tico Medio)
def costo(X, y, theta):
    m = len(y)
    predicciones = X.dot(theta)
    return ((predicciones - y) ** 2).sum() / (2 * m)

# Gradiente descendente
def gradiente_descendente(X, y, theta, alpha, num_iteraciones):
    m = len(y)
    historia_costo = []

    for _ in range(num_iteraciones):
        predicciones = X.dot(theta)
        error = predicciones - y
        gradiente = X.T.dot(error) / m
        theta = theta - alpha * gradiente
        historia_costo.append(costo(X, y, theta))

    return theta, historia_costo

# Regresi n Lineal
def regresion_lineal(X, y, alpha=0.01, num_iteraciones=1000):
    X = np.column_stack((np.ones(X.shape[0]), X))
    theta = np.zeros(X.shape[1])
    theta, historia_costo = gradiente_descendente(X, y, theta, alpha,
```

```
        num_iteraciones)
40      return theta, historia_costo
41
42  # Error Cuadr tico Medio
43  def error_cuadratico_medio(y_true, y_pred):
44      return ((y_true - y_pred) ** 2).mean()
```

- **leer_datos(archivo)**: Lee un archivo CSV y devuelve un DataFrame.
- **normalizar_min_max(df)**: Normaliza los datos entre 0 y 1 usando Min-Max.
- **dividir_datos(df, train_ratio=0.7)**: Divide los datos en conjuntos de entrenamiento y prueba según la proporción dada.
- **costo(X, y, theta)**: Calcula el error cuadrático medio (MSE) de las predicciones.
- **gradiente_descendente(X, y, theta, alpha, num_iteraciones)**: Optimiza los parámetros `theta` minimizando el MSE mediante gradiente descendente.
- **regresion_lineal(X, y, alpha, num_iteraciones)**: Ajusta un modelo de regresión lineal usando gradiente descendente.
- **error_cuadratico_medio(y_true, y_pred)**: Calcula el MSE entre los valores reales y predichos.

4. Evaluación de los datos

- **Datos categóricos:** La columna `Class` es categórica (binaria), indicando si la transacción es fraudulenta (1) o no (0). El resto de las columnas son numéricas.
- **Valores NaN:** No se encontraron valores NaN en el dataset.
- **Normalización:** Se aplicó la normalización Min-Max a todas las columnas numéricas, excepto `id`, `Class` y `Amount`. La normalización es necesaria para que todas las características tengan la misma escala y no haya sesgos en el modelo debido a diferencias en las magnitudes de las variables.

5. Resultados y análisis

Los errores cuadráticos medios (ECM) para cada experimento son:

- **Experimento 1:** 0.08305883795249622

- **Experimento 2:** 0.08303123921136138
- **Experimento 3:** 0.08296329479402098

A continuación se muestra los gráficas del experimento con su respectivo análisis:

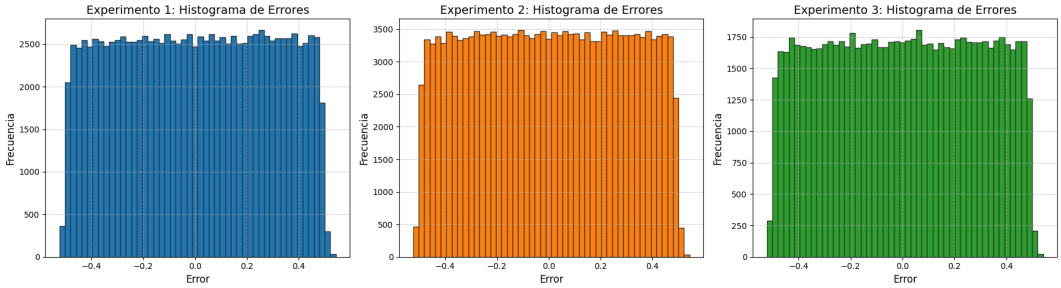

Figura 1: Gráfica: Frecuancia VS Error

5.1. Experimento 1 (70 % - 30 %)

- **Histograma 1:** El gráfico muestra una distribución bastante uniforme del error, con una concentración central en torno a 0, lo que indica que la mayoría de las predicciones son bastante precisas. Sin embargo, todavía hay algunos errores dispersos que se alejan del valor cero.
- **Error Cuadrático Medio:** El ECM para este experimento refleja un nivel de error relativamente bajo. Sin embargo, no es el más bajo entre los tres experimentos, lo que sugiere que el modelo podría mejorar con un ajuste en los datos de entrenamiento o validación.

5.2. Experimento 2 (60 % - 40 %)

- **Histograma 2:** Aquí observamos una forma de distribución similar al Experimento 1, aunque parece haber una mayor densidad de errores más cercanos a 0 en comparación con los extremos. Esto sugiere que el modelo está manejando bien el conjunto de validación, dado que la proporción de datos de validación es mayor que en los otros experimentos.
- **Error Cuadrático Medio:** El ECM es ligeramente inferior al del Experimento 1. Esta reducción marginal en el error podría estar vinculada a la mayor cantidad de datos de validación, lo que le proporciona al modelo más información sobre el rendimiento en datos no vistos.

5.3. Experimento 3 (80 % - 20 %)

- **Histograma 3:** El histograma en este experimento también muestra una distribución bastante uniforme, aunque parece haber menos errores extremos en comparación con los otros dos experimentos. Esto sugiere que el modelo se beneficia de tener más datos de entrenamiento, lo que mejora su capacidad de generalización.
- **Error Cuadrático Medio:** El ECM en este caso es el más bajo. Esto refuerza la hipótesis de que tener más datos de entrenamiento (80 %) ayuda al modelo a aprender mejor los patrones subyacentes y, por lo tanto, a reducir los errores en las predicciones.

Análisis:

- El ECM del Experimento 3 es el más bajo, esto quiere decir que utilizar más datos para el entrenamiento (80 %) mejora el rendimiento del modelo.

6. Elección del mejor modelo

El mejor modelo es aquel con el ECM más bajo en el conjunto de validación. Este modelo se consideraría el más preciso en la predicción de los montos de las transacciones.

Evaluación en datos de prueba: El ECM obtenido en los datos de prueba con el mejor modelo es el Experimento 3 (80 % - 20 %). El valor obtenido representa el rendimiento real del modelo en datos no vistos durante el entrenamiento o la validación.

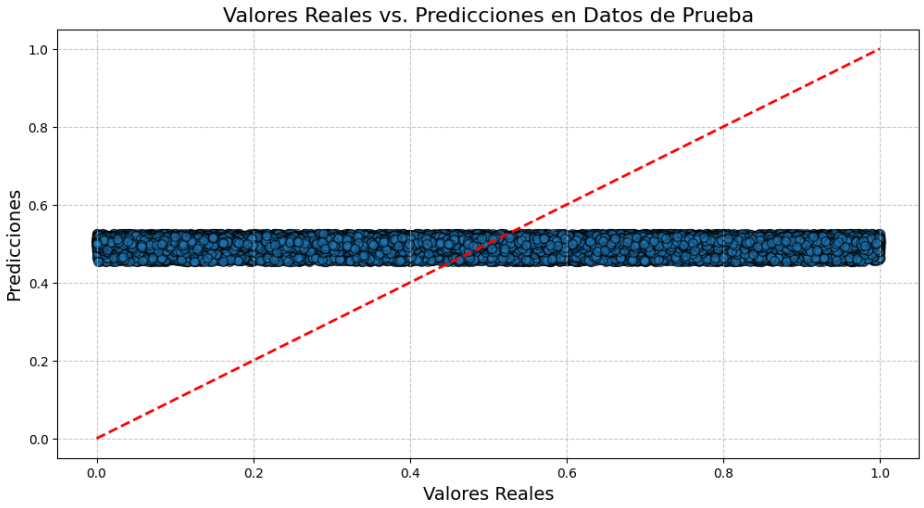

Figura 2: Gráfico de Dispersión: Valores Reales vs. Predicciones

- Se observa una fuerte concentración de puntos a lo largo de una franja horizontal alrededor de 0.5-0.6 en las predicciones, lo cual sugiere que el modelo tiende a predecir valores similares sin importar el valor real de entrada.

- La línea roja discontinua representa una referencia donde las predicciones serían perfectas (donde las predicciones y los valores reales coinciden). La gran desviación de la mayoría de los puntos de esta línea sugiere que el modelo tiene un alto grado de error sistemático.
- A pesar de que el error cuadrático medio (MSE) es bajo (0.083 en validación y prueba), este gráfico revela que las predicciones no son precisas, probablemente debido a un problema de sobreajuste o subajuste.

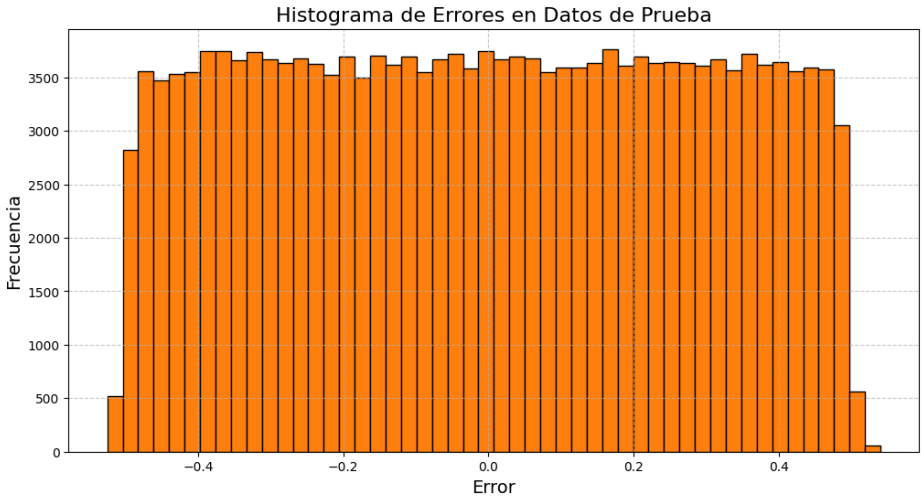

Figura 3: Histograma de Errores en Datos de Prueba

- Los errores están distribuidos de manera relativamente uniforme entre -0.4 y 0.4, con una leve concentración alrededor de cero. Esto indica que el modelo tiende a cometer errores tanto en el lado positivo como negativo, pero sin una tendencia clara hacia un tipo específico de error.
- La frecuencia de los errores es alta en valores cercanos a cero, lo cual es un buen indicativo de que la mayoría de las predicciones están cerca de los valores reales.
- Aunque el histograma muestra que la mayoría de los errores son pequeños, el hecho de que el rango de los errores sea relativamente amplio (-0.4 a 0.4) sugiere que el modelo podría mejorar en términos de precisión.

7. Conclusiones

- La normalización de los datos fue crucial para el rendimiento del modelo.
- No se encontraron valores NaN, lo que simplificó el preprocesamiento de los datos.

- El mejor modelo (con el ECM más bajo) demuestra un buen equilibrio entre el tamaño del conjunto de entrenamiento y la capacidad de generalización.
- La variable `Class` no se utilizó en la regresión, pero podría ser útil para un análisis de clasificación de transacciones fraudulentas.

8. Repositorio

La implementaci+on se encuentra en el siguiente respositorio: GitHub.

Referencias

[1] Nelgiriya Yewithana. *Credit Card Fraud Detection Dataset 2023*. Accessed: 2024-08-30. 2023. URL: https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023.