Universidad Católica San Pablo

Laboratorio: Regresion Lineal Multivariada

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.

El objetivo es determinar el mejor modelo predictivo basado en el Error Cuadrático Medio (MSE) para cada configuración del experimento. Además, se incluye un análisis completo de los datos, el preprocesamiento y los resultados obtenidos.

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.

```python
# Funci n para normalizar usando Min-Max
def normalizar_min_max(df):
    return (df - df.min()) / (df.max() - df.min())

# Funci n para crear la matriz de correlaci n
def matriz_correlacion(df):
    return df.corr()

# Funci n para dividir los datos
def dividir_datos(X, y, train_ratio=0.7):
    n = X.shape[0]
    train_size = int(n * train_ratio)
    return X[:train_size], X[train_size:], y[:train_size], y[train_size
       :]

# Funci n usada para la regresi n lineal
def regresion_lineal(X, y):
    X = np.column_stack((np.ones(X.shape[0]), X))
    return np.linalg.inv(X.T @ X) @ X.T @ y

#Funsi+on para el MSE
def error_cuadratico_medio(y_true, y_pred):
    return np.mean((y_true - y_pred)**2)

#Funci n predictiva
def predecir(X, coef):
    X = np.column_stack((np.ones(X.shape[0]), X))
    return X @ coef
```

- **normalizar_min_max(df)**: Normaliza los valores de un *DataFrame* a un rango [0, 1].
- **matriz_correlacion(df)**: Calcula la matriz de correlación entre las columnas del *DataFrame*.
- **dividir_datos(X, y, train_ratio=0.7)**: Divide los datos en entrenamiento y prueba según un ratio.

- **regresion_lineal(X, y)**: Ajusta un modelo de regresión lineal usando la fórmula de mínimos cuadrados.
- **error_cuadratico_medio(y_true, y_pred)**: Calcula el MSE entre valores reales y predichos.
- **predecir(X, coef)**: Genera predicciones usando los coeficientes de regresión.

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. Esto facilita el preprocesamiento, evitando la necesidad de imputar o eliminar datos faltantes.
- **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.

4.1. Matriz de Correlación

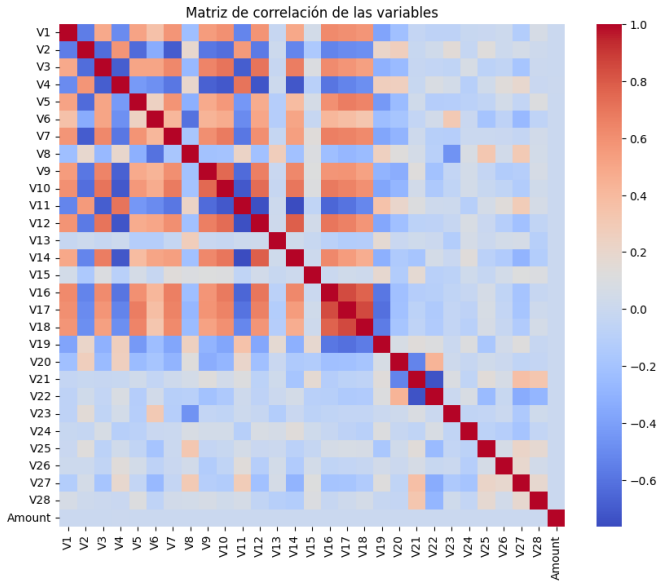

Figura 1: Matriz de correlación del dataset

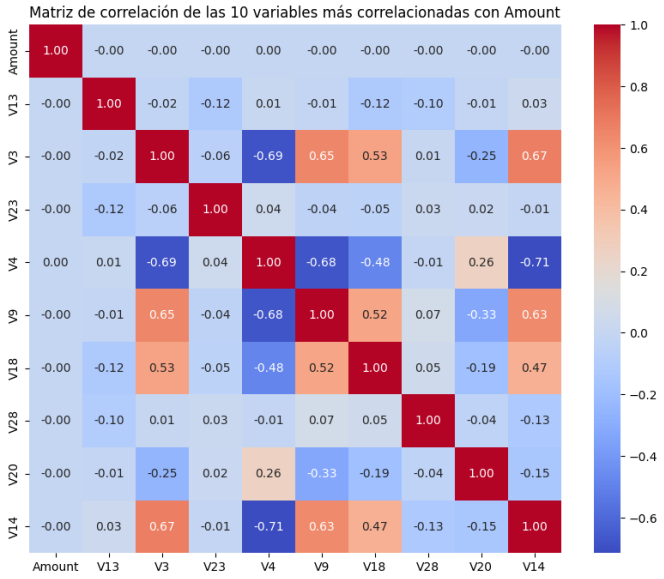

Figura 2: Matriz de correlación de las 10 variables más correlacionadas con Amount

- La primera matriz 1 muestra una baja correlación entre la mayoría de las variables, con pocas relaciones significativas. Las áreas rojas indican correlaciones positivas y las azules, negativas, pero en general, las relaciones son débiles.
- La segunda matriz 2, centrada en las 10 variables más correlacionadas con Amount, muestra que ninguna variable tiene una correlación fuerte con el monto. V13, V3 y V9 tienen correlaciones muy débiles, lo que indica que Amount no está fuertemente relacionado con ninguna variable individual en el dataset.

5. Experimentos y Resultados

Se dividió el conjunto de entrenamiento en proporciones de 70 %-30 %, 60 %-40 % y 80 %-20 %, y se evaluaron los modelos usando Regresión Lineal. El objetivo fue minimizar el Error Cuadrático Medio (MSE).

5.1. Experimento 1: División 70 %-30 %

- MSE Train: 0.08339941246733613
- MSE Test: 0.08296762046364167
- Diferencia: 0.0004317920036944617

Este experimento mostró un MSE bajo, pero no el más bajo. Esto indica que el modelo tiene un buen rendimiento, aunque podría mejorarse al ajustar la división de los datos de entrenamiento y prueba.

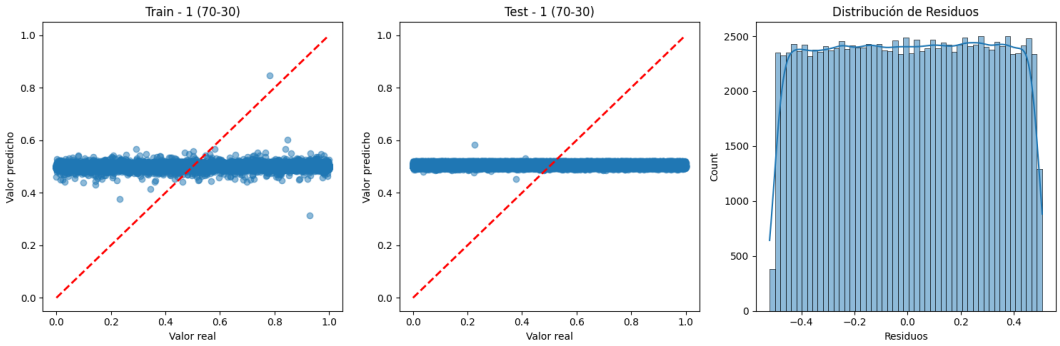

Figura 3: Valor predicho VS Valor real

- Gráfico de entrenamiento: Hay una dispersión considerable entre los valores reales y los valores predichos, aunque los valores se concentran en el rango intermedio. El modelo no parece ajustarse bien a los datos, con predicciones bastante planas.
- Gráfico de prueba: Similar al gráfico de entrenamiento, muestra que el modelo tiene dificultades para predecir correctamente. La dispersión es aún mayor que en el conjunto de entrenamiento.
- Distribución de residuos: Los residuos tienen una distribución uniforme, lo que indica que los errores están distribuidos de manera consistente, sin sesgo aparente.

5.2. Experimento 2: División 60 %-40 %

- MSE Train: 0.08350210820271121
- MSE Test: 0.0829228864983752
- Diferencia: 0.0005792217043360109

Al aumentar el conjunto de validación, se observó una mejora marginal en el MSE. Esto sugiere que proporcionar más datos de validación ayuda al modelo a entender mejor el comportamiento en datos no vistos.

- Gráfico de entrenamiento: Al igual que en el experimento anterior, las predicciones son bastante planas, con algunos puntos que se desvían significativamente de la línea roja (perfecta predicción).
- Gráfico de prueba: La predicción sigue siendo plana con valores que no logran ajustarse a los valores reales, lo que sugiere un mal desempeño del modelo en la generalización.
- Distribución de residuos: Los residuos nuevamente muestran una distribución bastante uniforme, lo que es un buen indicador de que no hay sobreajuste o sesgo en la predicción de errores.

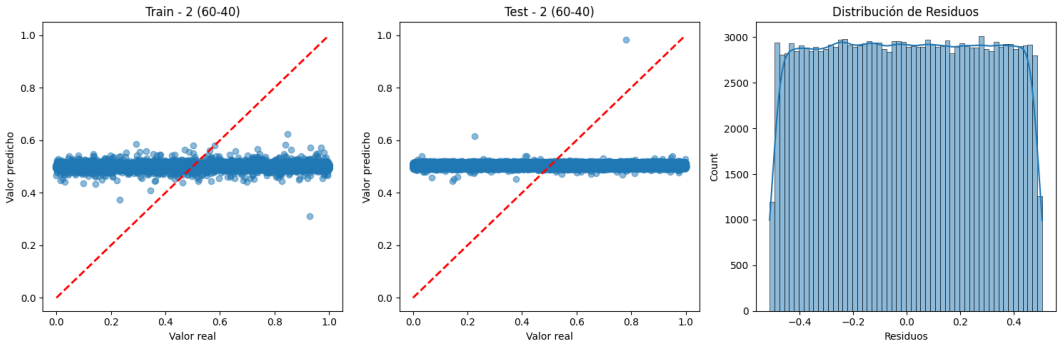

Figura 4: Valor predicho VS Valor real

5.3. Experimento 3: División 80 %-20 %

- MSE Train: 0.08336685409305604
- MSE Test: 0.08287106345311296
- Diferencia: 0.0004957906399430806

Este fue el mejor resultado. Utilizar más datos de entrenamiento permitió que el modelo aprendiera mejor los patrones subyacentes, lo que resultó en una mejor capacidad de generalización.

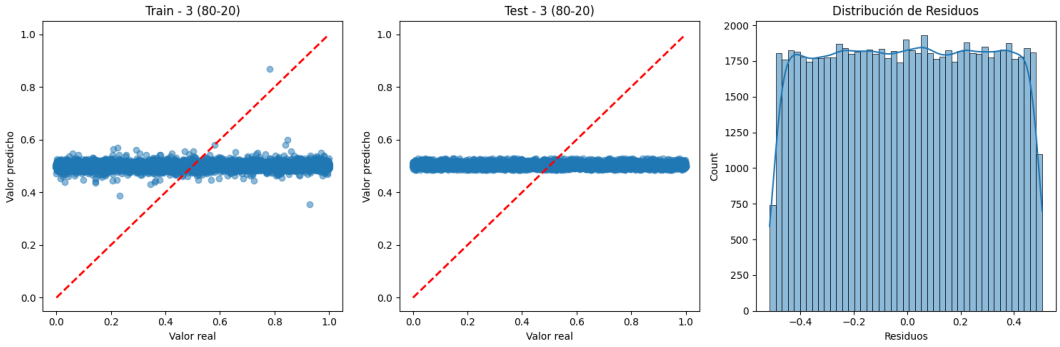

Figura 5: Valor predicho VS Valor real

- Gráfico de entrenamiento: Muestra una dispersión algo similar a los anteriores, con valores predichos concentrados en torno a un rango estrecho, lo que implica una falta de capacidad predictiva.
- Gráfico de prueba: Refleja lo mismo, con predicciones planas y muchos puntos alejados de la línea ideal.
- Distribución de residuos: Al igual que en los otros experimentos, la distribución de los residuos es bastante uniforme.

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.

Comparación del MSE en Diferentes Configuraciones A lo largo de los experimentos, se observó que el Experimento 3 fue el más eficiente con un MSE de 0.08336685409305604. Esto indica que tener más datos de entrenamiento mejora el rendimiento del modelo, disminuyendo la probabilidad de sobreajuste.

Evaluación en el 30 % Restante El modelo seleccionado se evaluó con el 30 % de los datos que no fueron utilizados durante el entrenamiento. El MSE en estos datos fue de 0.08304872469732205, lo cual es muy cercano al resultado del Experimento 3. Esto demuestra que el modelo generaliza bien a datos nuevos.

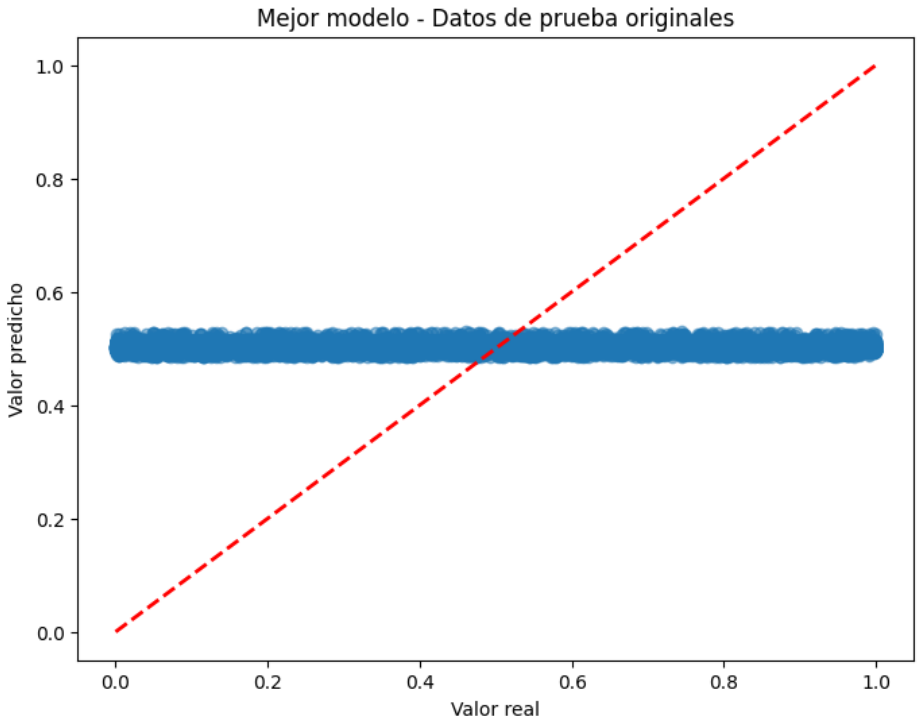

Figura 6: Mejor modelo VS Datos de pruebas originales

El gráfico 6 muestra la comparación entre los valores reales y los valores predichos por el mejor

modelo en los datos de prueba. La línea roja punteada representa la línea ideal donde los valores predichos serían iguales a los valores reales (ajuste perfecto).

Sin embargo, los puntos azules, que representan las predicciones del modelo, están muy alejados de esta línea, concentrándose en torno a un valor cercano a 0.5. Esto sugiere que el modelo tiene dificultades para predecir correctamente los valores y que las predicciones tienden a ser casi constantes, lo cual indica un posible problema de subajuste (underfitting) o que el modelo no está capturando bien la relación entre las variables.

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.