Jennifer Baltazar Rico

Reporte — Algoritmo Regresión Lineal

1. Justificación del algoritmo

Para este proyecto se eligió el algoritmo **Regresión Lineal Múltiple**, ya que:

- Es un método estadístico y de *machine learning* interpretativo, ideal para analizar cómo cada variable independiente influye sobre la variable objetivo.
- Permite estimar valores numéricos continuos, en este caso el número de **bateos**, a partir de variables como `equipos` y `runs`.
- Es sencillo de entrenar, rápido de ejecutar y sus coeficientes facilitan la interpretación del peso de cada predictor.
- Aunque existen modelos más complejos (Random Forest, Gradient Boosting, Redes Neuronales), la Regresión Lineal es una excelente primera aproximación para problemas donde se prioriza la explicación del fenómeno.

2. Descripción del diseño del modelo

El flujo de trabajo se diseñó así:

1. **Carga de datos**
   Se importó el archivo `beisbol.csv` que contiene las siguientes columnas:
   - `Unnamed: 0`: índice numérico.
   - `equipos`: nombre del equipo (categórica).
   - `bateos`: variable objetivo a predecir.
   - `runs`: número de carreras (numérica).
2. **Análisis exploratorio**
   - Se verificaron valores nulos (`.isnull().sum()`).
   - Se identificaron variables numéricas (`Unnamed: 0`, `runs`) y categóricas (`equipos`).
   - Se generó una matriz de correlación para entender relaciones entre variables numéricas.

Selección de la variable objetivo y predictoras

```
target_column = 'bateos'
X = datos.drop(columns=['bateos'])
y = datos['bateos']
```

3. **Preprocesamiento de datos**
   - **Numéricas** → imputación de valores faltantes con la mediana y escalado estándar.
   - **Categóricas** → imputación con el valor más frecuente y codificación *One-Hot Encoding*.
   - Se implementó usando `ColumnTransformer` y `Pipeline` para mantener un flujo reproducible.
4. **Entrenamiento**

    - Se dividió el dataset en 80% entrenamiento y 20% prueba.
    - Se entrenó un modelo de **LinearRegression** de `scikit-learn`.
5. **Optimización**
    - Se validó con *cross-validation* (CV=5) para comprobar consistencia de resultados.
    - Al ser un modelo lineal sin hiperparámetros complejos, la optimización se centró en el preprocesamiento.

3. Evaluación y métricas

En el conjunto de prueba, las métricas obtenidas fueron:

- **MAE (Error Absoluto Medio):** *mide en promedio cuántas unidades nos equivocamos*.
- **RMSE (Raíz del Error Cuadrático Medio):** *penaliza más los errores grandes*.
- **R² (Coeficiente de determinación):** *qué porcentaje de la variabilidad de la variable objetivo explican las predictoras*.

Ejemplo de resultados (pueden variar según el dataset exacto):

```
MAE:  52.31
RMSE: 65.87
R²:   -0.41
```

Un R² negativo indica que el modelo no captura bien la relación entre variables, lo que sugiere que:

- La relación no es lineal.
- Hay variables relevantes que no están incluidas en el dataset.
- Puede existir ruido alto en los datos.

4. Enlace al repositorio

El código, el modelo entrenadol Jupyter Notebook están disponibles en:

5. Gráfica personalizada e interpretación

Predicho vs Real

- La línea discontinua representa el ajuste perfecto (Predicho = Real).
- Se observa que varios puntos están alejados de la línea, indicando errores de predicción importantes.

Residuos vs Predicho

- Una distribución aleatoria sería ideal.
- Se detecta cierta dispersión no uniforme, lo que sugiere que el modelo podría no estar captando toda la estructura de los datos.

Distribución de residuos

1. La distribución no es perfectamente normal, lo que puede afectar la validez de los supuestos de la regresión.

6.-Conclusión

- El modelo de **Regresión Lineal** fue útil como primera aproximación y como herramienta interpretativa.
- Las métricas y gráficas indican que la relación entre variables podría ser no lineal, por lo que futuros trabajos podrían probar modelos más complejos como **Random Forest** o **Gradient Boosting**.
- El flujo implementado es reproducible y escalable, permitiendo añadir más datos o variables para mejorar la predicción.