Momento de Retroalimentación: Módulo 2 Implementación de una técnica de aprendizaje máquina sin el uso de un framework. (Portafolio Implementación)

Guadalupe Paulina López Cuevas A01701095@tec.mx

Resumen—

En este proyecto se usó el dataset Parkinson's Telemonitoring, que contiene mediciones biomédicas de la voz de pacientes con Parkinson en etapas tempranas, para predecir sus puntuaciones clínicas (motor_UPDRS y total_UPDRS). Se realizaron diferentes etapas de ETL como el preprocesamiento, incluyendo la limpieza de datos, reducción de features y análisis de correlación, para tener el dataset en las mejores condiciones para el entrenamiento del modelo. Inicialmente se implementó una regresión lineal simple, donde los resultados no fueron satisfactorios debido a la baja correlación lineal entre las variables. Posteriormente se aplicó una transformación polinomial de 2do grado que mejoró de forma considerable el rendimiento del modelo, reduciendo el error y aumentando la capacidad de predicción. Los resultados muestran que, aunque la predicción no es perfecta, el modelo puede capturar parte de la relación entre las variables.

I. Introducción

El estudio de la enfermedad de Parkinson a través de señales de voz representa una herramienta importante para el diagnóstico y seguimiento de los pacientes, ya que la voz refleja directamente los cambios motores característicos de esta condición. El dataset utilizado en este proyecto recopila más de cinco mil grabaciones de voz obtenidas con telemonitoreo, permitiendo un registro continuo y no invasivo del progreso de la enfermedad. Con estas grabaciones se obtienen diversos features como jitter, shimmer, NHR/HNR y métricas no lineales que miden irregularidades en la voz.

El objetivo principal fue analizar el dataset, realizar el proceso de limpieza y selección de variables relevantes, y finalmente implementar un modelo de regresión que permita predecir la severidad de los síntomas del Parkinson. Se evaluaron tanto la regresión lineal simple como una regresión polinomial, con el objetivo de determinar qué tan bien estos enfoques pueden ajustarse a la complejidad de los datos.

II. Información sobre dataset

El dataset "Parkingsons telemonitoring" es un data set proporcionado por Oxford en el que la información corresponde a un estudio que se hizo sobre la progresión de la enfermedad de Parkinson en etapas tempranas. Se recopilaron un total de 5,875 grabaciones de voz de 42 pacientes, a los cuales se les realizo un telemonitoreo desde sus casas por seis meses. El objetivo principal de todos estos datos fue predecir las puntuaciones clínicas de la enfermedad (motor_UPDRS y total_UPDRS) basándose en las medidas biomédicas extraídas de la voz.

El dataset cuenta con variables relacionadas a características acústicas como variaciones de frecuencia, amplitud, complejidad no lineal y la relación entre componentes tonales y de ruido. Algunos conceptos importantes de saber sobre los atributos que se midieron:

- **Jitter (Estabilidad del tono):** analizan cambios en la frecuencia fundamental (pitch) de la voz. Si hay mucha variación, la voz suena temblorosa o inestable.
- **Shimmer (Estabilidad del volumen):** mide que tan estable es el volumen de la voz. Una persona con Parkinson suena "temblorosa" en intensidad de la voz.
- **NHR/HNR:** Proporción de ruido vs claridad en la voz.
- **RPDE, DFA, PPE**: métricas más avanzadas que miden la complejidad e irregularidad de la voz.

Este dataset cuenta con los siguientes 22 atributos:

subject# (sin unidad)	Integer	ID único de cada paciente
age (medida en años)	Integer	Edad del paciente
sex (sin unidades)	Binary	Genero (0 = masculino, 1= femenino)
test_time (medida en días)	Continuous	Tiempo transcurrido desde el inicio de la prueba (en días).
Jitter(%) (voces normales < 1%)	Continuous	Variación % de la frecuencia de la voz. (mide que tan estable es la voz).
Jitter(Abs) (segundos(s) o milisegundos (ms), valores normales ~10-100 ms)	Continuous	Variación absoluta (en segundos) de la frecuencia de la voz.
Jitter:RAP (%) (valores altos = voz inestable)	Continuous	Mide la irregularidad de la voz promediando las variaciones entre ciclos consecutivos.

Jitter:PPQ5 (%) (valores altos = voz inestable)	Continuous	Mide variaciones de tono, pero considerando promedios en ciclos de 5 vibraciones.
Jitter:DDP (%) (valores altos = voz inestable)	Continuous	3 veces el valor de RAP, otra forma de medir la irregularidad del tono.
Shimmer (%) (valores normales< 3-4%)	Continuous	Variación relativa en la intensidad de la voz. Mide que tan uniforme es la voz.
Shimmer(dB)	Continuous	Shimmer expresado en decibeles.
Shimmer:APQ3 (%) (valores normales< 3-4%)	Continuous	Variación de amplitud promedio en 3 ciclos consecutivos.
Shimmer:APQ5 (%) (valores normales< 3-4%)	Continuous	Variación de amplitud promedio en 5 ciclos consecutivos.
Shimmer:APQ11 (%) (valores normales< 3-4%)	Continuous	Variación de amplitud promedio en 11 ciclos consecutivos.
Shimmer:DDA (%) (valores normales< 3-4%)	Continuous	Similar a APQ3, pero calculado de otra manera.
NHR (Noise-to-Harmonics Ratio) (Unidad: proporción (adimensional), valores normales > 0.2)	Continuous	Relación entre el ruido y los componentes tonales de la voz. Un valor alto significa mas ruido en la voz.
HNR (Harmonics-to-Noise Ratio) (Unidad: decibeles, valores normales > 20dB)	Continuous	Relación entre los componentes tonales y el ruido. Es inverso a NHR: valores altos significa una voz más clara.
RPDE (Recurrence Period Density Entropy) (Unidad: adimensional (0-1))	Continuous	Mide la complejidad de la señal de voz. Detecta irregularidades en los patrones de vibración de las cuerdas vocales.
DFA (Detrended Fluctuation Analysis) (Unidad: adimensional, valores típicos 0.5 y 1.5)	Continuous	Exponente que mide el componente fractal de la voz (que tan caótica o regular es).
PPE (Pitch Period Entropy) (Unidad: adimensional, valores altos = mayor irregularidad)	Continuous	Mide la variabilidad no lineal del tono. Cuanto más alto, mas irregular es la frecuencia.
motor_UPDRS (Escala numérica clínica: Indice)	Continuous (Target)	Puntuación clínica que mide la severidad de los síntomas motores del Parkinson (rigidez, temblores, lentitud de movimiento)
total_UPDRS (Escala numérica clínica: menos revero 0 – 108 más severo)	Continuous (Target)	Puntuación clínica total que evalúa tanto síntomas motores como no motores.

III. ENTORNO PARA TRABAJAR

Lo primero de queremos hacer es crear un entorno en donde podamos trabajar todo nuestro proyecto.

```
# Crear environment para el proyecto
python -m venv yenv

# Activar environment (Windows)
yenv\Scripts\activate
```

Código. 1. Crear un entorno de trabajo

De igual forma debemos instalar (de ser necesario) e importar las librerías necesarias para poder trabajar con nuestro dataset, operaciones, transformaciones, funciones, graficas, etc:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import math
```

Código. 2. Librerías para el proyecto

En este caso se usara Visual Code como herramienta de trabajo donde tendremos la terminal tanto para correr el programa como para tener el proyecto en GitHub.

IV. EXTRACCIÓN

Para poder utilizar un dataset que nos funcione para un modelo de técnicas de aprendizaje de maquina tenemos que hacer varios pasos antes de poner empezar a entrenar nuestro modelo, en este caso es la extracción y limpieza de dichos datos. Al extraer los datos, ya sea de un Excel, .csv, .data o cualquier tipo de archivo, los ponemos en nuestro espacio de trabajo para poder visualizarlos y poder manipularlos para nuestra limpieza del dataset. De igual forma una vez extraídos es importante conseguir la información del dataset ya que esto nos da todos los atributos dentro del dataset, la cantidad de datos que hay (que nos servirá saberlo posteriormente), si hay valores nulos, y tipo de dato.

V. LIMPIEZA DE DATOS

Para la limpieza del dataset se hicieron varias transformaciones para asegurar que los datos que hay, features, son los mejores.

V.I Null count

Primero se revisa si el dataset tiene algún valor nulo que al momento de manipular los datos provoque problemas.

V.II Remove null

En caso de que haya valores nulos, estos se deben remover. En caso de este dataset no existieron valores nulos, pero de igual forma se eliminó cualquier valor que pudiera ser nulo.

V.III Remove duplicate

De igual forma se tiene que revisar si el dataset cuenta con valores duplicados que puedan afectar al momento de hacer el entrenamiento del modelo.

V.IV Remove columns (subject#, test_time)

Una vez que se limpia el dataset, se deben quitar las columnas (features) que no serán relevantes para el entrenamiento ni para las predicciones. En este caso las columnas que fueron removidas fueron las siguientes:

- subject#: Se remueve porque es el ID del paciente (valor unico).
- test_time: Se remueve porque es el tiempo transcurrido desde que inicio la prueba (valor no relevante en la prueba).

V.V Quitar outliers

Muchas veces en los datsets existen valores que son comportamientos fuera de los patrones o errores de medición, etc. Por estos valores que se encuentran muy lejos de demás dataset es necesario quitarlos. La manera en la que se quitaron los outliers(valores fuera del patrón normal), por medio de calcular los percentiles entre el 25% y el 75%, los demás fueron despreciados, ya sea que se encontraran en el extremo izquierdo (valores muy negativos) o en el extremo derecho (valores muy positivos).

V.VI Hacer shuffle de datos

Finalmente se realiza un shufla pruebaaset ya que en este caso este contiene varias pruebas sobre las mismas 42 personas y es necesario mezclar para que el modelo no entrene a partir de algunos pacientes y al hacer la pruebas sea con un paciente diferente, sino que el modelo pueda ver a todos los pacientes, pero no todas sus pruebas.

Una ves finalizada la limpieza de los datos, en este caso el dataset termina con las siguientes dimensiones:

4511 rows x 20 columns

Ecuación. 1. Dimensiones del dataset después de limpieza

VI. Correlación entre features y targets

Para este dataset la distribución es de la siguiente manera:

- Features: ['age', 'sex', 'Jitter(%)', 'Jitter(Abs)', 'Jitter:RAP', 'JitteR:PPQ5', 'Jitter:DDP', 'Shimmer', 'Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5', 'Shimmer:APQ11', 'NHR', 'HNR', 'RPDE', 'DFA', 'PPE']
-
- Targets:['motor_UPDRS', 'total_UPDRS',]

El siguiente paso, después de la limpieza de datos, es buscar cual es la correlación entre los features(X) y el target(Y). Para este dataset la matriz de correlación se vio de la siguiente manera:

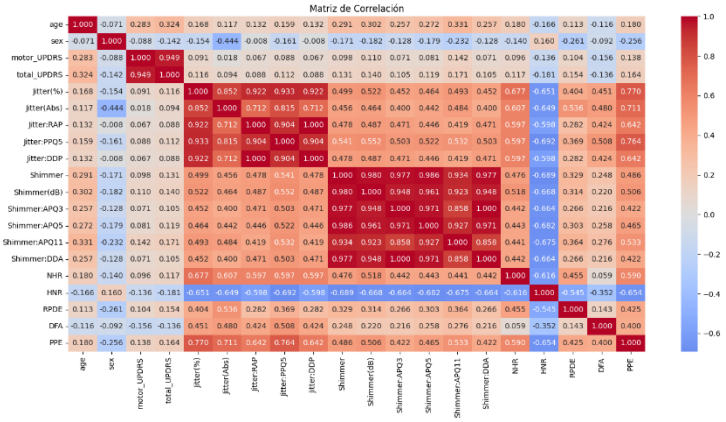

Fig. 1. Matriz de correlación (18 features, 2 targets)

Como se puede observar en la Fig.1, en la tercera y cuarta columna y fila se encuentran nuestros targets. La manera de interpretar esta matriz es observando los recuadros de estas columnas y filas y recorrerlos completos para ver cuales features están con un color muy oscuro (rojo o azul), esto nos dice que tanta correlación existe. Para este dataset se pude ver que si hay correlación entre los features y los targets, pero esta relación es muy baja lo cual puede perjudicar el modelo de entrenamiento.

Existe otra grafica que puede ayudar a visualizar de mejor manera si los datos de los features tienen una correlación lineal con los targets. Para este dataset el plot de pares se vio de la siguiente manera:

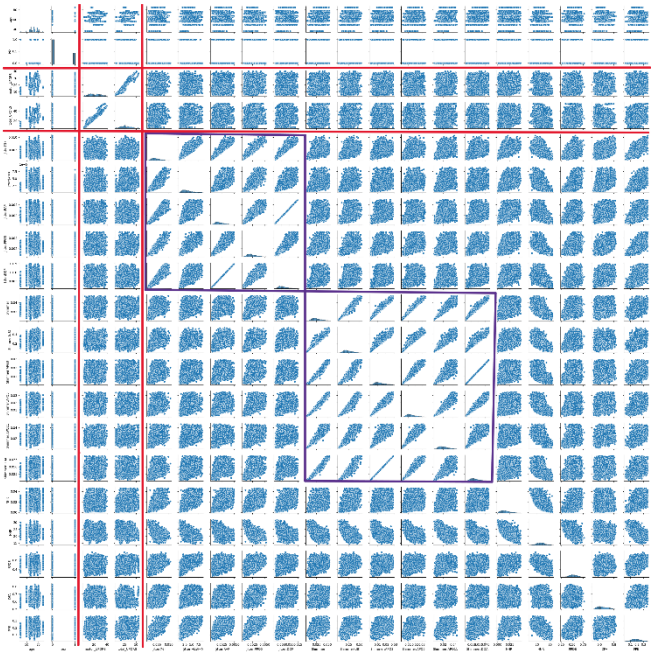

Fig. 2. Gráfica de pares (18 features, 2 targets)

Dentro de la Fig.2 se puede ver entre las líneas rojas la correlación que tienen cada uno de los features con los targets. Se puede ver que el comportamiento de cada feature con respecto a cada target NO es lineal, esto puede ser debido a que los valores tienen mucha variabilidad entre ellos, es por eso que en la matriz de correlación los valores son muy pequeños.

La manera ideal como se debería de ver la correlación entre los features y los targets es como se puede ver en los recuadros morados, esas graficas donde los valores "siguen" una diagonal significa que tienen una correlación lineal.

En este dataset, no existe una relación lineal entre features y targets, pero si entre algunos features entre sí. Para el entrenamiento de un modelo, que los features tengan una correlación lineal tan grande mientras que no exista mucha correlación lineal entre features y targets se vuelve un problema llamado multicolinealidad, así que lo que se procederá a hacer, que podría ser una recomendación, quitar algunos de los features que tengan relación entre si para disminuir esta correlación y que no perjudique el modelo.

VII. REDUCCIÓN DE FEATURES

Para la selección de los features hay que basarse en la matriz de correlación para seleccionar las que tienen una correlación mas alta con los targets. No es necesario que todas estén fuertemente correlacionadas, pero aquellas con mayor correlación positiva o negativa suelen ser las mas relevantes para el modelo. Hacemos un nuevo dataset con los features de mayor correlación y volvemos a sacar la matriz de correlación como se muestra a continuación:

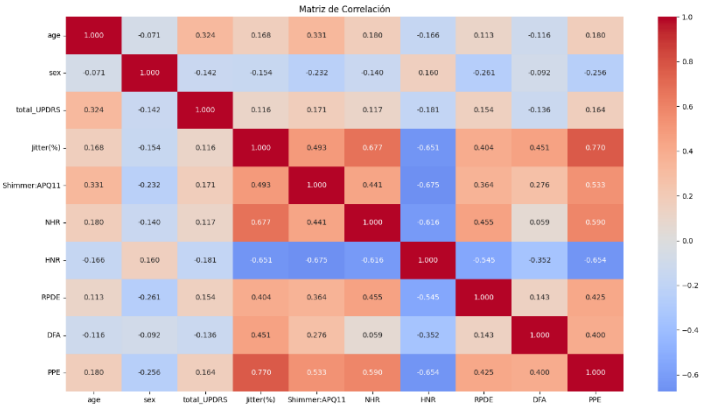

Fig. 3. Matriz de correlación (9 features,1 target)

Como se observa en la Fig.3, se redujeron considerablemente los features y targets (en este caso solo se usará un target). Sigue sin haber una correlación lineal muy fuerte entre estos features pero son los que tienen la mayor correlación. El dataset quedo reestructurado de la siguiente manera:

- Features: ['age', 'sex', 'Jitter(%)', 'Shimmer:APQ11', 'NHR', 'HNR', 'RPDE', 'DFA', 'PPE']
- Target: ['total_UPDRS']

4511 rows x 10 columns

Ecuación. 2. Dimensiones del dataset reducido

De igual forma se usará otra grafica que ayudará a visualizar de mejor manera la correlación lineal entre los features reducidos y el target. Para este dataset reducido el plot de pares se vio de la siguiente manera:

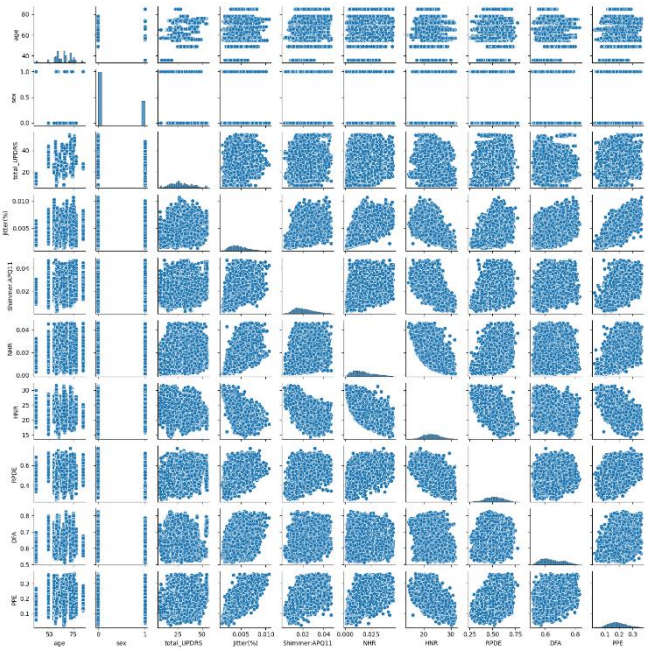

Fig. 4. Gráfica de pares (9 features,1 target)

En la Fig.4 se ve la correlación lineal que anteriormente se vio en el dataset completo, solo que ahora se puede ver que en

general no existe más esa correlación lineal tan fuerte entre los features. Teniendo el dataset listo se puede empezar a usar para el aprendizaje del modelo que se implementara.

VIII. MODELO DE ENTRENAMIENTO

Antes de poner el dataset dentro del modelo se debe separar el dataset en train y test. Esto debido a que para poder hacer un buen entrenamiento del modelo es necesario que al entrenarlo no se utilicen todos los datos del dataset para que el modelo no "aprenda" los valores y que al momento de hacer el test salga que tuvo un buen rendimiento cuando simplemente se terminó ajustando a los valores de entrenamiento.

Para este dataset se hizo una división en una proporción de 80/20 donde habrá 3608 muestras para entrenamiento (80%) y 903 muestras para prueba (20%), para que nuestro split siempre sea el mismo utilizamos una semilla de "random_state = 42".

$$3608 \times 903 = 4511$$

Ecuación. 3. Tamaño del dataset a trabajar

Una vez teniendo los datasets de entrenamiento y prueba, el siguiente paso es hacer una última transformación a los datos para poder meterlos al modelo.

VIII.I Escalamiento

Para los features, los valores independientes, es necesario escalarlos debido a que muchas veces los features cuentan con escalas diferentes, pueden ser valores muy grandes o pequeños, y de esta manera ningún feature va a tener mayor o menor peso que influya en el target. En este caso para el escalamiento se uso la siguiente función:

```
scaler = StandardScaler()
df_x_train = scaler.fit_transform(x_train)
df_x_test = scaler.transform(x_test)
```

Código. 3. Función para escalamiento

Esta función hace una transformación matemática, que tiene su media = 0, una desviación estándar = 1 y hace que los datos sigan una distribución normal estándar, donde cada columna se transforma usando la siguiente ecuación:

$$z = (x - \mu) / \sigma$$

x: Valor original del dato
μ: Media (promedio) de la característica
σ: Desviación estándar de la característica

Ecuación. 4. Ecuación de escalamiento

VIII.II Cambio de dimensiones

Para el target, valor dependiente, es necesario cambiarle las dimensiones debido a que como esta en una lista y no en una matriz, debemos convertir una serie de pandas a un array puro, cambiando las dimensiones de la siguiente manera:

$$(n, 1) => (n,)$$

Ecuación. 5. Cambio de 2D a 1D

Ahora si podemos empezar con el modelo de entrenamiento. Este dataset es un problema de regresión por lo que para poder resolverlo es necesario implementar una regresión lineal que es conocida por la siguiente formula:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n + b$$

$\hat{\mathbf{y}}$ = Predicciones (vector de shape (m,))
X = Matrix de features (shape (m, n))
θ = Parámetros del modelo (vector de weights, shape (n,))
b = Término de bias (escalar)

Ecuación. 6. Función de regresión lineal

Teniendo nuestra función de hipótesis podemos crear nuestro modelo de entrenamiento con las siguientes funciones:

VIII.IV Función de Hipótesis

hyp_theta(): Calcula la hipótesis de la regresión lineal, genera la predicción lineal para cada instancia de datos del modelo, haciendo el producto punto entre los features y los parámetros (θ o w) más el término bias (b), durante el proceso de entrenamiento.

$$hyp\_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

Ecuación. 7. Función de hipótesis

VIII.V Función de Mean Square Error (MSE)

MSE(): Calcula el Error Cuadrático Medio que cuantifica el costo (error) del modelo. Mide la diferencia al cuadrado entre la predicción (ŷ) y los valores reales (y).

$$J(\theta) = (1/2m) * \Sigma(hyp\_\theta(x^{(i)}) - y^{(i)})^2$$

Ecuación. 8. Función de mean square error

VIII.VI Función de Gradient Descent (GD)

update(): Implementa el algoritmo de Gradiente Descendiente. Actualiza de manera iterativa los parámetros theta (θ) y bias (b) para ajustar sus valores durante el entrenamiento.

$$\theta_j := \theta_j - \alpha * (1/m) * \Sigma(hyp\_\theta(x^{(i)}) - y^{(i)}) * x_j^{(i)}$$

$$b := b - \alpha * (1/m) * \Sigma(hyp\_\theta(x^{(i)}) - y^{(i)})$$

Ecuación. 9. Funciones de gradient descent para theta (θ) y bias (b).

VIII.VII Función de R^2 Score

Calcula el coeficiente de determinación R^2 que mide la proporción de la varianza en los features. Indica que porcentaje de la variabilidad de los datos es aprendida por el modelo.

Ya teniendo las funciones del modelo, es momento de definir los parámetros iniciales para el entrenamiento:

```
theta_W = np.zeros(len(df_x_train.shape[1]))
error = []      # Error
b = 100         # Bias
alfa = 0.01     # Learning rate
epoch = 100     # Epocas
```

Código. 4. Inicialización de parámetros

Como se ve en el Código.4, tanto los parámetros de theta_w como b no importa cual sea su valor inicial debido a que durante el entrenamiento estos se irán ajustando. El arreglo vacío de los errores para irlo llenando conforme se vaya entrenando el modelo. Para el learning rate, alfa = 0.01, se puso ese valor debido a que la ser un hiperparametro, este controla el tamaño de los pasos que da el algoritmo durante el entrenamiento. Las épocas van a empezar en 100 para ver el comportamiento del modelo, posteriormente de ser necesario se aumentará.

IX. Entrenamiento del modelo

Después del entrenamiento del modelo estos fueron los resultados:

```
Epoch 100: MSE = 399.034822, R-squared = -5.688124

Resultados finales después de 100 épocas:
Error final: 399.034822
Theta final: [ 1.86225932 -0.64207698  0.16735363
  0.50818107 -1.23674468  0.5286094 ]
b final: 54.79
```

Fig. 5. Resultados de entrenamiento

En la Fig.5 se puede ver claramente que el modelo no esta entrenando bien y se encuentra demasiado alejado de los valores a predecir, además de que la R^2 es muy negativa indicando que el modelo no saber predecir los valores porque no existe una correlación lineal. Esta prueba de igual manera se intentó con epoch = 400, y el resultado no bajo de manera significativa.

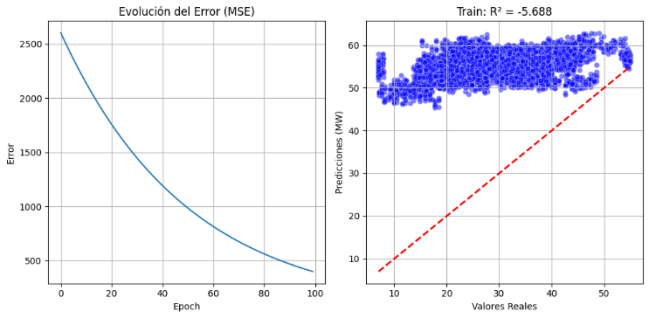

Fig. 6. Gráficas del error y Predicción vs Valor Real en entrenamiento.

En la Fig.6 se puede observar que, aunque el error esta disminuyendo, el modelo no se acerca para nada a los valores. Este es uno de los riesgos al saber que los features no cuentan con una correlación lineal fuerte con respecto al target.

X. Transformación a términos polinomiales

Debido a que, como se mencionó anteriormente, la correlación lineal entre los features y el target no es lineal una solución o propuesta que se puede hacer es hacer nuevamente una transformación a los features pero en este caso convertirlos a términos polinomiales para intentar ver si así el modelo puede acercarse de manera mas precisa a los valores reales del target.

Lo que se procederá hacer es usar el dataset reducido que hicimos anteriormente y crear un nuevo dataframe con los polimonios (2do grado) también, solo los features deben ser transformados y posteriormente se vuelve a agregar dentro del dataframe el target, teniendo como resultado un dataset con las siguientes dimensiones:

4511 rows x 55 columns

Ecuación. 10. Dimensiones del dataset después de limpieza

La razón por la que en este caso solo se transforman a un polinomio de 2do orden es porque tampoco se quiere un overfitting por parte del modelo y que se sobreajuste a los datos.

Nuevamente volvemos hacer una división en una proporción de 80/20 donde habrá 3608 muestras para entrenamiento (80%) y 903 muestras para prueba (20%), para que nuestro split siempre sea el mismo utilizamos una semilla de “random_state = 42”.

Hacemos nuevamente el escalamiento (features) y el cambio de dimensión (target), como se hizo previamente en el primer dataset reducido, para ajustar los valores y las dimensiones necesarias para reentrenar el modelo.

Volvemos a definir los parámetros para el entrenamiento:

```
theta_W = np.zeros(len(df_x_train.shape[1]))
error = []      # Error
b = 100         # Bias
alfa = 0.01     # Learning rate
epoch = 3000    # Epocas
```

Código. 5. Inicialización de parámetros nuevos

En este caso el único parámetro que se cambio fue las épocas que se llevaran a cabo. Para las pruebas, además de cambiar el número de épocas, también se cambió el learning rate a valores como alfa = 0.1, alfa = 0.001 pero en el modelo el MSE se disparaba y tenia un error gigantesco o los valores de predicción se volvían demasiado grandes, por lo que al final alfa = 0.01 se quedó como el hiperparametro.

XI. Resultado de entrenamiento

Después retransformar los datos, ahora si se ponen a reentrenar de nuevo.

```
Resultados finales después de 3000 épocas:
Error final: 45.431583
Theta final: [ 0.          1.33190677 -0.39978794 -0.22883103 -1.00941014 -0.46871557
 -0.1010638  -0.36379086 -0.46313589 -1.05931952 -0.26288686 -0.13143429
  0.22885868  0.72297313 -0.98394226  0.80455531 -0.89337272  0.76427322
  2.57892779 -0.39978794  3.55360007 -0.39408682  0.13161395 -0.66792416
 -0.46886766  0.01310097 -2.08377691 -1.56515547  1.01082934  1.44186299
  0.72495957 -0.11391781 -1.07646514 -0.00662405  2.72168958 -2.43192521
 -1.11231747 -3.074252   -0.30960558  3.21597689  0.13719722  2.97887796
 -1.57306278 -1.28360502 -0.23372395 -1.98541999  1.74278482 -1.89328004
 -1.65495065  0.07536588  1.77644243 -0.98745843 -1.20336672 -0.43835139
  2.34268348]
b final: 28.78
```

Fig. 7. Resultados de reentrenamiento

Como se puede ver en la Fig.7 el MSE bajo de manera considerable de 399.03 a 45.43, lo que significa que el modelo mejoro muchísimo tanto con los valores polinomiales como con la cantidad de épocas a ejecutar. De igual forma se puede ver que la R^2 ahora si muestra que el modelo si esta haciendo predicciones y que si puede existir una correlación lineal baja, pero si existe.

```
Epochs =  3000
Epoch 100: MSE = 395.875661, R-squared = -5.664770
Epoch 200: MSE = 95.203698, R-squared = -0.602803
Epoch 300: MSE = 54.714801, R-squared = 0.078848
Epoch 400: MSE = 49.097830, R-squared = 0.173413
Epoch 500: MSE = 48.169078, R-squared = 0.189049
```

Fig. 8. Entrenamientos primeros 500 Epochs

En la Fig.8 se observan los resultados de los primeros 500 epochs donde se puede ver como desde las 200 epochs el modelo mejora considerablemente y disminuye mucho el MSE y el R^2 se vuelve positivo.

```
Epoch 2500: MSE = 45.743141, R-squared = 0.229891
Epoch 2600: MSE = 45.675880, R-squared = 0.231023
Epoch 2700: MSE = 45.611241, R-squared = 0.232111
Epoch 2800: MSE = 45.549072, R-squared = 0.233158
Epoch 2900: MSE = 45.489230, R-squared = 0.234165
Epoch 3000: MSE = 45.431583, R-squared = 0.235136
```

Fig. 9. Entrenamientos últimos 500 Epochs

En la Fig.9 se observan los resultados de los últimos 500 epochs de 3000 epochs, en este caso se puede ver como ya en estas ultimas iteraciones el MSE y el R^2 no cambian mucho, siguen cambiando, dando a entender que el modelo sigue entrenando lo mejor que puede para ser lo más preciso, pero ya no hay mucho cambio.

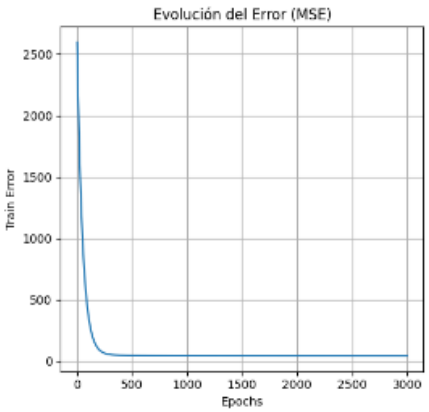

Fig. 10. Evaluación del error en el entrenamiento

Como se puede ver en la Fig.10 la gráfica del error esta bajando de manera más considerable a comparación de la del primer entrenamiento.

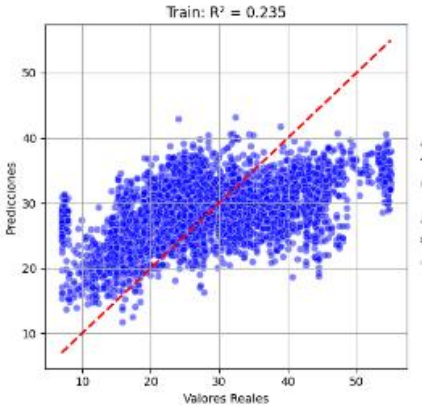

Fig. 11. Grafica Predicción vs Valor real en entrenamiento

Como se puede ver en la Fig.11 la línea roja ahora ya se encuentra encima de los valores de nuestro dataset, a pesar de que no todos los valores se encuentran cerca de la línea, muchos de ellos si están cerca de ella.

XII. PROBAR MODELO CON DATASET DE TEST

Ahora es momento de hacer el test con la otra parte del dataset para saber cómo se comportará nuestro modelo ya entrenado.

```
Epoch 5: MSE = 47.953018, R-squared = 0.240714

Error Cuadrático Medio (MSE) en el conjunto de prueba: 47.953018
```

Fig. 12. Resultados de prueba

Como se ve en la Fig.12 los resultados de la prueba son bastante similares a los resultados del entrenamiento, a pesar de que no es un error muy pequeño o una R^2 cerca de 1, se mantienen dentro de los valores que se esperaban para un modelo entrenado para predecir features y tergets con baja correlación lineal.

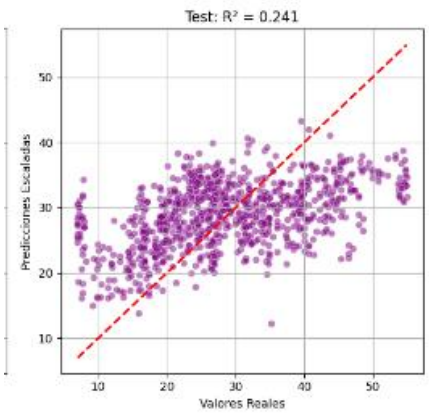

Fig. 13. Grafica Predicción vs Valor real en prueba

Como se observa en la Fig.13 el resultado es muy similar al del entrenamiento, a pesar de que la línea roja no representa por completo el comportamiento de todo el data set logra predecir algunos valores.

XIII. Conclusiones

Durante el desarrollo de este proyecto me permitió comprender de manera práctica las etapas necesarias para llevar a cabo el proceso de ETLs para el aprendizaje automático: desde la exploración y limpieza del dataset hasta la implementación y evaluación de modelos predictivos. Los resultados obtenidos para este dataset muestran que la regresión lineal simple no es suficiente para capturar la relación entre las variables acústicas y las puntuaciones clínicas del Parkinson, ya que la correlación lineal es baja entre los features y el target. Sin embargo, al transformar los datos en términos polinomiales, el modelo logró una mejora considerable en el error y en la capacidad de hacer predicciones, lo cual confirma que para la resolución del problema se requieren si o si métodos que consideren relaciones no lineales.

Sabiendo que el modelo polinomial aún presenta limitaciones, con este trabajo puedo darme cuenta de que efectivamente en problemas reales de análisis y aprendizaje de datasets complejos, relaciones no lineales, son necesarias las redes neuronales. Aunque en este trabajo no se han implementado, se puede ver el potencial de la precisión de este tipo de datasets con redes neuronales y de igual manera entrenamientos multihilos y por minibatches para hacerlos más rápidas, eficientes y precisas las predicciones. Con esto se puede ver la importancia del preprocesamiento de datos y la selección adecuada del modelo para abordar problemas reales de predicción de datos complejos.

XIV. Referencias

- Ruder, S. (2017). *An overview of gradient descent optimization algorithms*. arXiv:1609.04747. https://arxiv.org/abs/1609.04747
- Brownlee, J. (2021). *Linear Regression for Machine Learning*. Machine Learning Mastery. https://machinelearningmastery.com/linear-regression-for-machine-learning/
- Scikit-learn developers. (2023). StandardScaler documentation. *Scikit-learn: Machine Learning in Python*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
- Frost, J. (2023). *How to Interpret R-squared in Regression Analysis*. Statistics By Jim. https://statisticsbyjim.com/regression/interpret-r-squared-regression/