ANEXOS:

FUNDAMENTOS DE MACHINE LEARNING CON SCIKIT-LEARN Y PYTORCH

SESION 1/ SEMANA 2

Anexo 01 : Ejemplo de modelo de Machine Learning de Detección de Diabetes

A continuación, para entender las etapas del entrenamiento y evaluación del modelo veremos el caso de predecir si un paciente esta con diabetes o no, basado en 10 características médicas.

Los conjuntos de datos de diabetes utilizado representan diferentes medidas médicas y biométricas de los pacientes. Estas características son variables numéricas que han sido normalizadas (escaladas) para que sus valores estén en una escala similar. Aquí están las características usadas:

1. **Edad (age):** Edad del paciente (normalizada).
2. **Sexo (sex):** Sexo del paciente (normalizado).
3. **Índice de Masa Corporal (IMC) (bmi):** Una medida de la grasa corporal en relación con la altura y el peso (normalizada).
4. **Presión Arterial Promedio (bp):** Valor promedio de la presión arterial (normalizado).
5. **Medida S1 (s1):** Una medida relacionada con el colesterol total en sangre (normalizada).
6. **Medida S2 (s2):** Una medida relacionada con la lipoproteína de baja densidad (LDL) en sangre (normalizada).
7. **Medida S3 (s3):** Una medida relacionada con la lipoproteína de alta densidad (HDL) en sangre (normalizada).
8. **Medida S4 (s4):** Una medida relacionada con los niveles de triglicéridos (normalizada).
9. **Medida S5 (s5):** Una medida relacionada con los niveles de azúcar en sangre (normalizada).
10. **Medida S6 (s6):** Una medida relacionada con el nivel de glucosa en sangre (normalizada).

Estas características son el resultado de transformaciones estadísticas y no representan directamente medidas fácilmente interpretables. Sin embargo, están relacionadas con aspectos clave de la salud metabólica, como la presión arterial, el nivel de glucosa y las lipoproteínas.

Las características han sido **normalizadas** para que tengan **media 0 y desviación estándar 1,** lo que es común en muchos conjuntos de datos de machine learning para facilitar la convergencia de los modelos de aprendizaje.

Este código utiliza un modelo de regresión logística para predecir si un paciente tiene diabetes basado en un conjunto de características médicas. Después de entrenar el modelo, se evalúa su precisión y se utilizan datos de prueba nuevos para hacer predicciones. A continuación, se presenta el código en Python:

```
#Paso 1: Importar las bibliotecas necesarias

from sklearn.datasets import load_diabetes

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score

import numpy as np

#Paso 2: Cargar el conjunto de datos de diabetes

diabetes = load_diabetes()

X = diabetes.data  # Características médicas

y = (diabetes.target > diabetes.target.mean()).astype(int)  # Etiqueta binaria (1 si el nivel de riesgo es alto, 0 si es bajo)

#Paso 3: Dividir los datos en conjunto de entrenamiento y prueba

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Paso 4: Escalar las características

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

#Paso 5: Crear y entrenar el modelo.Vamos a usar regresión logística, que es un modelo de clasificación lineal sencillo.

model = LogisticRegression()

# Entrenar el modelo

model.fit(X_train, y_train)

#Paso 6: Hacer predicciones sobre el conjunto de prueba
```

```
y_pred = model.predict(X_test)

#Paso 7: Evaluar la precisión del modelo

accuracy = accuracy_score(y_test, y_pred)

print(f"Precisión del modelo: {accuracy * 100:.2f}%")

#Paso 8: Predecir con nuevos datos

#Vamos a suponer que tenemos nuevos pacientes con características que queremos evaluar.

#Paso 9: Datos de prueba nuevos (características de nuevos pacientes)

X_new = np.array([

   [0.03807591,  0.05068012,  0.06169621,  0.02187235, -0.0442235, -0.03482076, -0.04340085, -0.00259226,  0.01990749, -0.01764613],

   [-0.00188202, -0.04464164,  0.05068012,  0.01103904,  0.01549073,  0.0123073, 0.02377494, -0.03949338, -0.06833339, -0.09220405],

   [0.08529891, -0.04464164,  0.04552903, -0.03734449,  0.07441156,  0.05822365, 0.03444344,  0.02377494, -0.01811827,  0.04448548]

])


# Escalar los nuevos datos utilizando el mismo escalador

X_new_scaled = scaler.transform(X_new)

# Hacer predicciones con los nuevos datos

predicciones = model.predict(X_new_scaled)

# Mostrar los resultados

for i, pred in enumerate(predicciones):

   resultado = "Diabetes" if pred == 1 else "No Diabetes"

   print(f"Paciente {i+1}: {resultado}")
```

Preguntas frecuentes:

¿Por qué es importante la normalización de las características?

Es importante porque sin normalización, las características con valores más grandes pueden dominar la contribución a la predicción del modelo, lo que puede llevar a resultados **sesgados o inexactos**.

En cambio, se si usó la normalización estándar con la clase StandardScaler de scikit-learn, que escala las características de manera que tengan una media de 0 y una desviación estándar de 1.

Los algoritmos de machine learning que dependen de la distancia entre puntos de datos, que necesitan que sean normalizados sus características son:

- Regresión logística
- Máquinas de soporte vectorial (SVM)
- K-Nearest Neighbors (KNN)
- Redes neuronales

Esto significa que, después de la normalización con StandardScaler:

- La media de cada característica en los datos escalados es 0.
- La desviación estándar de cada característica es 1.

Anexo 02 : Tarea Aprendizaje 02

Alumno: [Juan Piero Vincha Loza]

1. ¿Cuál es la diferencia entre aprendizaje supervisado y no supervisado?

El aprendizaje supervisado utiliza datos etiquetados para entrenar modelos, donde conocemos las respuestas correctas y el modelo aprende a predecirlas. Por ejemplo, clasificar correos como spam o no spam. En cambio, el aprendizaje no supervisado trabaja con datos no etiquetados, buscando patrones o estructuras sin conocer las respuestas de antemano. Un ejemplo sería agrupar clientes por comportamiento de compra sin categorías predefinidas.

2. ¿Cuál es la diferencia entre regresión lineal y regresión logística?

La regresión lineal se usa para predecir valores continuos, como el precio de una casa basado en su tamaño. Produce una línea recta que mejor se ajusta a los datos. La regresión logística, por otro lado, se utiliza para problemas de clasificación binaria, prediciendo la probabilidad de que algo pertenezca a una de dos categorías. Por ejemplo, predecir si un estudiante aprobará o no un examen basado en sus horas de estudio.

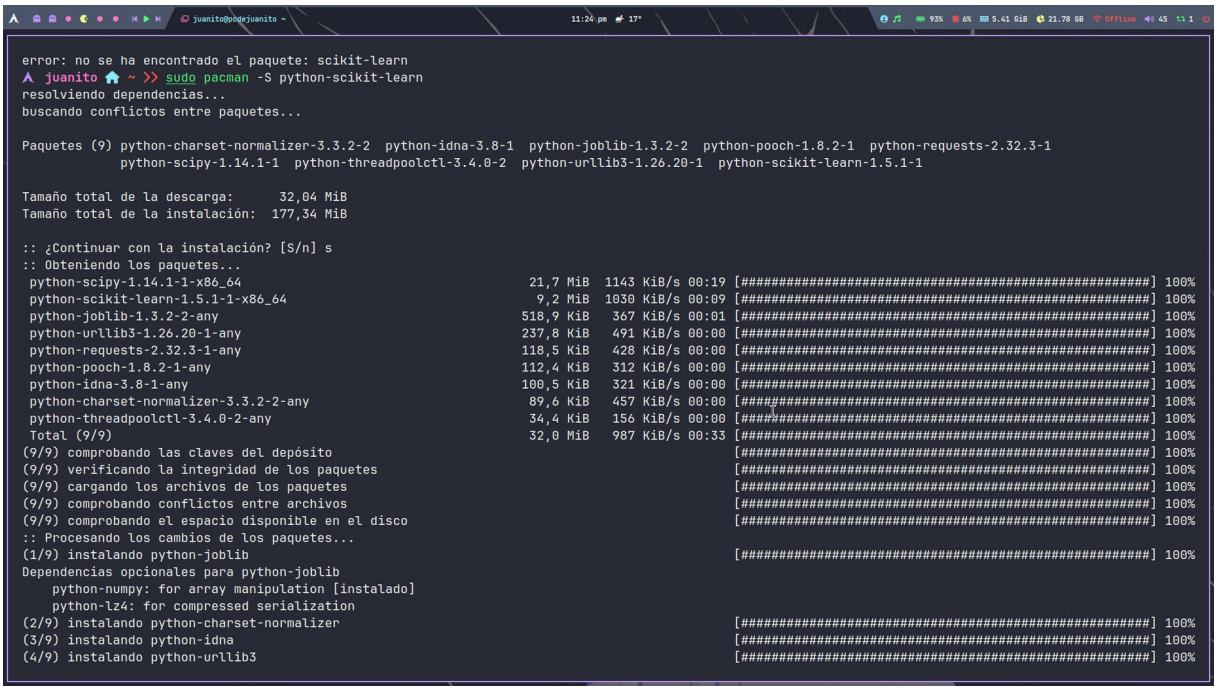
3. ¿Cuál es la diferencia entre métrica de exactitud y precisión y recall?

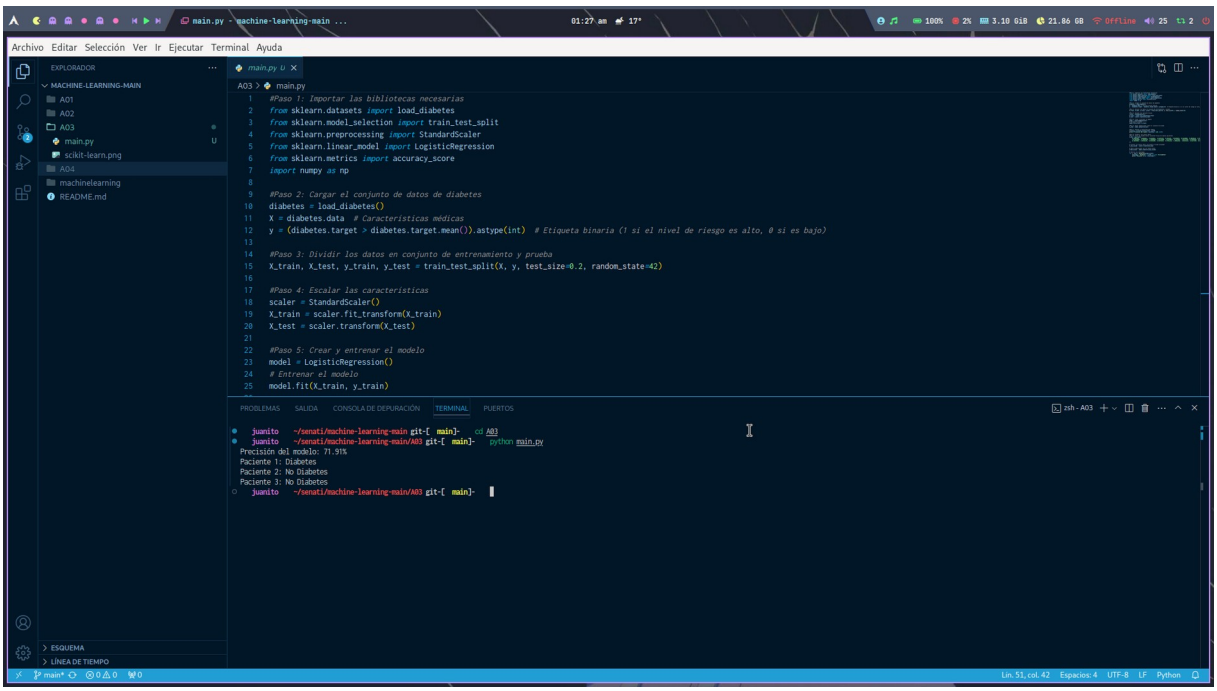
La exactitud (accuracy) es la proporción de predicciones correctas sobre el total. La precisión es la proporción de predicciones positivas correctas sobre todas las predicciones positivas, mientras que el recall es la proporción de predicciones positivas correctas sobre todos los casos realmente positivos. La precisión se enfoca en la calidad de las predicciones positivas, mientras que el recall se centra en no perder casos positivos reales.

4. ¿Cuál es la diferencia entre paquete scikit-learn y pytorch?

Scikit-learn es una biblioteca de Python para aprendizaje automático tradicional, con algoritmos como regresión, clasificación y clustering. Es fácil de usar y buena para proyectos más pequeños o para empezar en machine learning. PyTorch, por otro lado, es un framework de aprendizaje profundo, especializado en redes neuronales y computación en GPU. Es más flexible y potente, pero también más complejo. Se usa mucho en investigación y para proyectos de deep learning a gran escala.

5. De acuerdo al anexo 01, codificar el programa en Python, capturando la pantalla de las predicciones nuevas (12 PTOS)





Código ejecutado a través de la terminal de VS Codium.