Machine Learning aplicado usando Python

1. Introducción

El *Machine Learning* o aprendizaje automático es una rama de la inteligencia artificial que permite a los sistemas aprender patrones a partir de los datos y realizar predicciones o tomar decisiones sin que cada regla haya sido programada manualmente. En lugar de definir todas las instrucciones de forma explícita, el modelo identifica regularidades en los datos de entrada y las utiliza para responder ante nuevos casos.

Python se ha convertido en uno de los lenguajes más utilizados en este ámbito por su sintaxis sencilla, su amplia comunidad y la existencia de bibliotecas especializadas como pandas, numpy, matplotlib y scikit-learn. Gracias a estas herramientas, es posible desarrollar soluciones de aprendizaje automático de forma relativamente accesible, tanto para fines académicos como para aplicaciones reales.

Desde una perspectiva aplicada, el aprendizaje automático puede emplearse para clasificar correos electrónicos, detectar fraudes, predecir precios, recomendar productos, analizar textos o identificar comportamientos anómalos en sistemas informáticos. Su utilidad reside en la capacidad para transformar grandes volúmenes de datos en conocimiento accionable.

---

2. Tipos principales de Machine Learning

2.1. Aprendizaje supervisado

En el aprendizaje supervisado, el modelo se entrena con datos etiquetados. Esto significa que cada ejemplo de entrada va acompañado de una respuesta correcta. El objetivo consiste en aprender la relación entre los datos de entrada y la salida esperada.

Los problemas más habituales dentro de esta categoría son:

- **Clasificación**: asignar una categoría.
- **Regresión**: predecir un valor numérico.

Ejemplo: predecir si un correo es spam o no spam a partir de su contenido.

2.2. Aprendizaje no supervisado

En este caso, los datos no incluyen etiquetas. El sistema trata de encontrar estructuras, agrupaciones o relaciones ocultas dentro del conjunto de datos.

Ejemplo: segmentar clientes en grupos según sus hábitos de compra.

2.3. Aprendizaje por refuerzo

Este enfoque se basa en un agente que interactúa con un entorno, recibe recompensas o penalizaciones y aprende estrategias para maximizar la recompensa acumulada.

Ejemplo: entrenar un sistema para jugar a un videojuego o para optimizar rutas.

3. Flujo general de un proyecto de Machine Learning

Un proyecto de aprendizaje automático suele seguir varias etapas encadenadas:

3.1. Recopilación de datos

El primer paso consiste en obtener los datos que se utilizarán para entrenar el modelo. Estos pueden proceder de bases de datos, sensores, archivos CSV, APIs o repositorios públicos.

3.2. Limpieza y preparación

Los datos rara vez están listos para usarse tal como se obtienen. Es habitual encontrar valores nulos, formatos inconsistentes, duplicados o variables irrelevantes.

3.3. Selección de características

No todas las variables tienen la misma utilidad. Elegir las características relevantes mejora el rendimiento y facilita la interpretación del modelo.

3.4. Entrenamiento del modelo

Durante esta fase, el algoritmo aprende a partir de los datos de entrenamiento.

3.5. Evaluación

Una vez entrenado, el modelo se evalúa con datos no vistos anteriormente para comprobar su capacidad de generalización.

3.6. Uso en nuevos casos

El modelo ya entrenado puede emplearse para hacer predicciones sobre datos nuevos.

---

4. Bibliotecas principales en Python

4.1. NumPy

Se usa para trabajar con arrays y operaciones matemáticas eficientes.

```
import numpy as npdatos = np.array([10, 20, 30, 40])
print(datos.mean())
```

4.2. Pandas

Permite manipular datos tabulares de manera cómoda.

```
import pandas as pddf = pd.DataFrame({
"edad": [25, 32, 47],
"salario": [18000, 24000, 35000]
})print(df)
```

4.3. Matplotlib

Se utiliza para visualizar datos.

```
import matplotlib.pyplot as pltx = [1, 2, 3, 4]
y = [10, 15, 13, 20]plt.plot(x, y)
plt.title("Ejemplo de gráfico")
plt.xlabel("Eje X")
plt.ylabel("Eje Y")
plt.show()
```

4.4. Scikit-learn

Es una de las bibliotecas más utilizadas para Machine Learning clásico en Python.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

---

5. Ejemplo 1: regresión lineal en Python

La regresión lineal se emplea para predecir valores numéricos. Imaginemos que queremos estimar el precio de una vivienda a partir de su tamaño.

5.1. Datos de ejemplo

```
import pandas as pddatos = pd.DataFrame({
"metros": [50, 60, 80, 100, 120, 150],
"precio": [120000, 140000, 180000, 210000, 250000, 310000]
})print(datos)
```

5.2. Entrenamiento del modelo

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegressionX = datos[["metros"]]
y = datos["precio"]X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = train_test_split(
X, y, test_size=0.2, random_state=42
)modelo = LinearRegression()
modelo.fit(X_entrenamiento, y_entrenamiento)
```

5.3. Predicción

```
prediccion = modelo.predict([[90]])
print("Precio estimado para 90 metros:", prediccion[0])
```

5.4. Interpretación

El modelo intenta ajustar una línea que relacione los metros cuadrados con el precio. Después, utiliza esa relación para estimar el valor de una vivienda de tamaño no visto durante el entrenamiento.

---

6. Ejemplo 2: clasificación con árbol de decisión

Ahora supongamos que queremos clasificar si un estudiante aprobará o suspenderá según las horas de estudio y la asistencia.

6.1. Datos de ejemplo

```
import pandas as pddatos = pd.DataFrame({
"horas_estudio": [1, 2, 3, 4, 5, 6, 7, 8],
"asistencia": [50, 60, 65, 70, 75, 80, 90, 95],
"aprueba": [0, 0, 0, 1, 1, 1, 1, 1]
})print(datos)
```

6.2. Entrenamiento

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifierX = datos[["horas_estudio", "asistencia"]]
y = datos["aprueba"]X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = train_test_split(
X, y, test_size=0.25, random_state=42
)modelo = DecisionTreeClassifier(random_state=42)
modelo.fit(X_entrenamiento, y_entrenamiento)
```

6.3. Predicción

```
nuevo_estudiante = [[4, 72]]
```

```
resultado = modelo.predict(nuevo_estudiante)if resultado[0] == 1:
print("Predicción: aprueba")
else:
print("Predicción: suspende")
```

6.4. Explicación

El árbol de decisión aprende reglas a partir de los datos, por ejemplo, si el número de horas de estudio supera cierto valor o si la asistencia es suficientemente alta.

---

7. Ejemplo 3: agrupamiento con K-Means

El algoritmo K-Means sirve para agrupar elementos similares sin necesidad de etiquetas previas.

7.1. Datos de ejemplo

```
import pandas as pdclientes = pd.DataFrame({
"gasto_anual": [1000, 1200, 1500, 8000, 8500, 9000],
"visitas_mes": [1, 2, 2, 10, 12, 11]
})print(clientes)
```

7.2. Aplicación de K-Means

```
from sklearn.cluster import KMeansmodelo = KMeans(n_clusters=2, random_state=42, n_init=10)
clientes["grupo"] = modelo.fit_predict(clientes)print(clientes)
```

7.3. Interpretación

El algoritmo genera dos grupos de clientes con características similares. Esto podría utilizarse, por ejemplo, para campañas

comerciales diferenciadas.

---

8. Evaluación de modelos

La evaluación es una parte esencial porque no basta con entrenar un modelo: hay que comprobar si realmente funciona bien.

8.1. Métricas en clasificación

Algunas métricas frecuentes son:

- **Accuracy**: porcentaje de aciertos.
- **Precisión**: proporción de predicciones positivas correctas.
- **Recall**: capacidad para detectar los positivos reales.
- **F1-score**: equilibrio entre precisión y recall.

```
from sklearn.metrics import accuracy_score, classification_reportpredicciones = modelo.predict(X_prueba)
print("Accuracy:", accuracy_score(y_prueba, predicciones))
print(classification_report(y_prueba, predicciones))
```

8.2. Métricas en regresión

Entre las más utilizadas destacan:

- **MAE**: error absoluto medio.
- **MSE**: error cuadrático medio.
- **R²**: proporción de variabilidad explicada por el modelo.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_scorepredicciones = modelo.predict(X_prueba)print("MAE:", mean_absolute_error(y_prueba, predicciones))
```

```
print("MSE:", mean_squared_error(y_prueba, predicciones))
print("R2:", r2_score(y_prueba, predicciones))
```

---

9. Importancia del preprocesamiento

La calidad del modelo depende en gran medida de la calidad de los datos. Algunas tareas habituales de preprocesamiento son:

- eliminar valores nulos
- convertir texto en valores numéricos
- escalar variables
- normalizar datos
- codificar variables categóricas

Ejemplo de normalización

```
from sklearn.preprocessing import StandardScaler
import pandas as pddatos = pd.DataFrame({
"edad": [20, 25, 30, 35],
"salario": [1000, 1500, 2000, 3000]
})escalador = StandardScaler()
datos_escalados                                                =
escalador.fit_transform(datos)print(datos_escalados)
```

---

10. Ejemplo completo: detección simple de spam

Este ejemplo muestra una aplicación básica de clasificación de texto.

10.1. Datos

```
import pandas as pdmensajes = pd.DataFrame({
"texto": [
"Has ganado un premio",
"Reunión mañana a las diez",
"Oferta exclusiva solo hoy",
"Adjunto el informe solicitado",
"Haz clic aquí para reclamar tu premio"
],
"spam": [1, 0, 1, 0, 1]
})
```

10.2. Vectorización y entrenamiento

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNBvectorizador = CountVectorizer()
X = vectorizador.fit_transform(mensajes["texto"])
y = mensajes["spam"]modelo = MultinomialNB()
modelo.fit(X, y)
```

10.3. Predicción de un nuevo mensaje

```
nuevo_texto = ["Premio exclusivo para ti"]
nuevo_vector = vectorizador.transform(nuevo_texto)prediccion = modelo.predict(nuevo_vector)if prediccion[0] == 1:
print("El mensaje parece spam")
else:
print("El mensaje no parece spam")
```

---

11. Ventajas del uso de Python en Machine Learning

Python presenta numerosas ventajas para el desarrollo de soluciones de aprendizaje automático:

- sintaxis clara y fácil de aprender
- abundancia de bibliotecas especializadas
- gran cantidad de documentación y ejemplos
- integración sencilla con análisis de datos y visualización
- amplia adopción en entornos académicos y profesionales

Estas características facilitan tanto el aprendizaje inicial como el desarrollo de proyectos más complejos.

---

12. Limitaciones y precauciones

Aunque el aprendizaje automático ofrece muchas posibilidades, también presenta limitaciones que deben tenerse en cuenta. Un modelo puede producir resultados erróneos si los datos están sesgados, si el volumen de ejemplos es insuficiente o si las variables elegidas no representan adecuadamente el problema. También existe el riesgo de sobreajuste, es decir, que el modelo funcione muy bien con los datos de entrenamiento pero falle con nuevos casos.

Además, conviene recordar que un modelo no “entiende” la realidad como una persona, sino que detecta regularidades estadísticas. Por ello, sus resultados deben interpretarse de manera crítica y, en contextos sensibles, acompañarse de supervisión humana.

---

13. Conclusión

El Machine Learning aplicado usando Python constituye una herramienta muy valiosa para transformar datos en predicciones, clasificaciones o agrupaciones útiles. Su

importancia ha crecido en múltiples sectores debido a la facilidad con la que permite automatizar tareas, extraer patrones y apoyar procesos de decisión. Python, por su parte, proporciona un entorno accesible y potente para poner en práctica estos modelos gracias a su ecosistema de bibliotecas y a la claridad de su sintaxis.

Desde una perspectiva formativa, comprender los fundamentos del aprendizaje automático y saber implementarlos con ejemplos sencillos en Python permite construir una base sólida para avanzar hacia técnicas más complejas, como redes neuronales, procesamiento del lenguaje natural o visión por computador.

---

14. Propuesta de ejercicios prácticos

Ejercicio 1

Crear un modelo de regresión lineal que prediga el salario de una persona en función de sus años de experiencia.

Ejercicio 2

Diseñar un clasificador que prediga si un cliente comprará un producto a partir de su edad y número de visitas a una web.

Ejercicio 3

Aplicar K-Means para agrupar alumnos según horas de estudio y calificaciones medias.

Ejercicio 4

Construir un detector básico de mensajes spam usando textos cortos en español.

15. Código integrado de ejemplo sencillo

Este ejemplo reúne un caso mínimo de clasificación:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pandas as pddatos = pd.DataFrame({
"horas_estudio": [1, 2, 3, 4, 5, 6, 7, 8],
"asistencia": [50, 55, 60, 65, 70, 80, 90, 95],
"aprueba": [0, 0, 0, 0, 1, 1, 1, 1]
})X = datos[["horas_estudio", "asistencia"]]
y = datos["aprueba"]X_entrenamiento, X_prueba, y_entrenamiento, y_prueba = train_test_split(
X, y, test_size=0.25, random_state=42
)modelo = DecisionTreeClassifier(random_state=42)
modelo.fit(X_entrenamiento, y_entrenamiento)predicciones = modelo.predict(X_prueba)print("Accuracy:", accuracy_score(y_prueba, predicciones))nuevo_caso = [[5, 75]]
resultado = modelo.predict(nuevo_caso)print("Predicción para nuevo caso:", resultado[0])
```