**Diapositiva 1: Título**

* **Título:** Entrenamiento de Modelos de Regresión Lineal y Regresión Logística
* **Subtítulo:** Limpieza de datos, modelado, métricas y evolución a clasificación
* **Notas para el presentador:** Introducción al tema, importancia de entender el proceso completo desde la limpieza hasta la evaluación.

**Diapositiva 2: Agenda / Índice**

* Limpieza y preparación de datos
* División del dataset
* Entrenamiento y validación de regresión lineal
* Interpretación de coeficientes
* Ejemplo práctico de interpretación
* Funciones sklearn para regresión lineal
* Evolución a regresión logística
* Métricas y matriz de confusión para clasificación
* Conclusiones y recomendaciones

**Diapositiva 3: Limpieza de datos – Exploración previa**

* **Puntos clave:**
  + Importancia de entender la calidad y estructura del dataset antes de modelar
  + Análisis descriptivo: medias, desviaciones, outliers
  + Visualización: mapas de calor para correlaciones y valores faltantes
* **Ejemplo visual:**
  + Mapa de calor con seaborn.heatmap() mostrando correlaciones
* **Código ejemplo:**

python

CopiarEditar

import seaborn as sns

import matplotlib.pyplot as plt

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.show()

**Diapositiva 4: Limpieza de datos – Tratamiento valores faltantes**

* **Puntos clave:**
  + Identificación de valores faltantes (df.isnull().sum())
  + Métodos para tratamiento:
    - Eliminación de filas o columnas
    - Imputación con media, mediana o moda
    - Técnicas avanzadas: modelos predictivos para imputar
* **Código ejemplo:**

python

CopiarEditar

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')

df[['col1', 'col2']] = imputer.fit\_transform(df[['col1', 'col2']])

**Diapositiva 5: Limpieza de datos – Normalización y one-hot encoding**

* **Puntos clave:**
  + Normalización para escalar variables numéricas y mejorar convergencia del modelo
  + Técnicas:
    - StandardScaler (media 0, desviación estándar 1)
    - MinMaxScaler (rango [0,1])
  + Variables categóricas transformadas con one-hot encoding para evitar orden implícito
* **Código ejemplo:**

python

CopiarEditar

from sklearn.preprocessing import StandardScaler, OneHotEncoder

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(df[['num\_feature']])

encoder = OneHotEncoder(sparse=False)

X\_encoded = encoder.fit\_transform(df[['categorical\_feature']])

**Diapositiva 6: División del dataset**

* **Puntos clave:**
  + Separar datos para entrenamiento, validación y prueba para evaluar desempeño real
  + Proporciones típicas: 70% entrenamiento, 15% validación, 15% prueba
  + Uso de train\_test\_split() para dividir datos de forma reproducible
* **Código ejemplo:**

python

CopiarEditar

from sklearn.model\_selection import train\_test\_split

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, random\_state=42)

**Diapositiva 7: Entrenamiento de regresión lineal múltiple**

* **Puntos clave:**
  + Modelo de regresión lineal para predecir variable continua en función de varias independientes
  + Ajuste del modelo con datos de entrenamiento
  + Evaluación preliminar con datos de validación
* **Código ejemplo:**

python

CopiarEditar

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred\_val = model.predict(X\_val)

**Diapositiva 8: Métricas para regresión lineal**

* **Puntos clave:**
  + R²: proporción de varianza explicada
  + RMSE: error cuadrático medio, indica magnitud de error
  + MAE: error absoluto medio, más robusto a outliers
* **Código ejemplo:**

python

CopiarEditar

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

import numpy as np

rmse = np.sqrt(mean\_squared\_error(y\_val, y\_pred\_val))

r2 = r2\_score(y\_val, y\_pred\_val)

mae = mean\_absolute\_error(y\_val, y\_pred\_val)

**Diapositiva 9: Interpretación de coeficientes en regresión lineal**

* **Puntos clave:**
  + Cada coeficiente representa el cambio esperado en la variable dependiente por unidad de cambio en la independiente, manteniendo otras constantes
  + Signo positivo o negativo indica dirección del efecto
  + Magnitud indica impacto relativo

**Diapositiva 9bis: Ejemplo práctico de interpretación de coeficientes**

* **Suposición:** Modelo predice precio vivienda
* **Variables:** tamaño (m²), habitaciones, edad propiedad
* **Coeficientes:**

| **Variable** | **Coeficiente** | **Interpretación** |
| --- | --- | --- |
| Intercepto | 5,000 | Precio base cuando variables = 0 |
| Tamaño (size) | 300 | Cada m² adicional incrementa precio en 300 unidades |
| Habitaciones | 10,000 | Cada habitación aumenta precio en 10,000 unidades |
| Edad (age) | -1,500 | Cada año reduce precio en 1,500 unidades |

* **Conclusión:** Tamaño y habitaciones tienen impacto positivo; edad reduce valor (depreciación).

**Diapositiva 10: Pipeline sklearn para regresión lineal**

* **Puntos clave:**
  + Pipeline encadena preprocesamiento y modelo, facilita reproducibilidad y evita fugas de datos
* **Código ejemplo:**

python

CopiarEditar

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

pipeline = Pipeline([

('scaler', StandardScaler()),

('regressor', LinearRegression())

])

pipeline.fit(X\_train, y\_train)

y\_pred\_val = pipeline.predict(X\_val)

**Diapositiva 11: Evolución a regresión logística – Concepto**

* **Puntos clave:**
  + Regresión logística para problemas de clasificación binaria
  + Modela probabilidad de pertenecer a clase positiva
  + Función sigmoide convierte combinación lineal a probabilidad

**Diapositiva 12: Pipeline sklearn para regresión logística**

* **Puntos clave:**
  + Pipeline similar a regresión lineal, con modelo LogisticRegression
* **Código ejemplo:**

python

CopiarEditar

from sklearn.linear\_model import LogisticRegression

pipeline\_log = Pipeline([

('scaler', StandardScaler()),

('classifier', LogisticRegression())

])

pipeline\_log.fit(X\_train, y\_train)

y\_pred\_val = pipeline\_log.predict(X\_val)

**Diapositiva 13: Métricas para regresión logística**

* **Puntos clave:**
  + Accuracy: porcentaje de predicciones correctas
  + Precision: proporción de verdaderos positivos sobre predicciones positivas
  + Recall: proporción de verdaderos positivos sobre total de positivos reales
  + F1-score: balance entre precisión y recall
  + AUC-ROC: área bajo curva para clasificación binaria
* **Código ejemplo:**

python

CopiarEditar

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

acc = accuracy\_score(y\_val, y\_pred\_val)

prec = precision\_score(y\_val, y\_pred\_val)

rec = recall\_score(y\_val, y\_pred\_val)

f1 = f1\_score(y\_val, y\_pred\_val)

**Diapositiva 14: Matriz de confusión**

* **Puntos clave:**
  + Matriz 2x2 que muestra TP, FP, TN, FN
  + Ayuda a entender errores del modelo
* **Código ejemplo:**

python

CopiarEditar

from sklearn.metrics import confusion\_matrix

import seaborn as sns

cm = confusion\_matrix(y\_val, y\_pred\_val)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

**Diapositiva 15: Conclusiones y recomendaciones**

* Limpieza y preparación son clave para modelos robustos
* Uso de pipelines para facilitar mantenimiento y evitar errores
* Interpretar coeficientes para insights de negocio
* Evaluar con métricas adecuadas según el problema (regresión vs clasificación)
* Aplicación directa en toma de decisiones y análisis predictivo