Actividad: Predicción de Rotación de Empleados en una Empresa (Employee Churn)

Objetivo:

Desarrollar una aplicación web que prediga la probabilidad de que un empleado **abandone la empresa**, utilizando un modelo de red neuronal entrenado en Python (Colab) y exportado a **TensorFlow.js** para una interfaz de análisis predictivo.

Esta información es útil para **gerentes de RR.HH.**, **directores de operaciones** o **analistas organizacionales**.

Dataset: HR Employee Attrition Dataset (IBM)

- Archivo: WA_Fn-UseC_-HR-Employee-Attrition.csv El archivo CSV se encuentra en el siguiente enlace: https://www.kaggle.com/datasets/patelprashant/employee-attrition?resource=download

Variables más relevantes:

Variable	Descripción	Tipo
Age	Edad del empleado	Numérica
BusinessTravel	Frecuencia de viajes laborales	Categórica
Department	Departamento del empleado	Categórica
DistanceFromHome	Distancia desde su domicilio	Numérica
Education	Nivel educativo (1 a 5)	Numérica
JobRole	Cargo actual del empleado	Categórica
MonthlyIncome	Ingreso mensual	Numérica
NumCompaniesWorked	Nº de empresas donde ha trabajado antes	Numérica
OverTime	¿Hace horas extra? (Yes/No)	Categórica
TotalWorkingYears	Años totales de experiencia laboral	Numérica
YearsAtCompany	Años en la empresa	Numérica
Attrition	Variable objetivo: 1 = se fue, 0 = sigue	Binaria

Parte técnica que deben implementar los estudiantes

1. Preprocesamiento

- Convertir variables categóricas a one-hot encoding: BusinessTravel, Department, JobRole, OverTime
- Variables numéricas se pueden usar tal como están
- Variable objetivo: Attrition (0 o 1) -> Ya está lista para ser usada como objetivo

- Dividir en X (features) y y (Attrition)
- Dividir en train/test con train_test_split().

Código para generar el preprocesamiento:

import tensorflow as tf

import pandas as pd from sklearn.model_selection

import train_test_split

La siguiente linea guarda en un dataframe los datos del archivo CSV

```
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

La siguiente línea tiene la función get_dummies:

- Esta función convierte automáticamente todas las **variables categóricas** (como `JobRole`, `OverTime`, `Department`, etc.) en columnas numéricas utilizando **one-hot encoding**.
- ¿Qué hace *one-hot encoding*?
- Crea una nueva columna para cada categoría.
- El resultado es que todas las columnas ahora son **números**, lo que es necesario para alimentar una red neuronal.

```
df = pd.get_dummies(df, drop_first=True)
```

En la siguiente línea se eliminan del DataFrame la columna Attrition_Yes, que es la variable que queremos predecir.

Todo lo que queda en X son las características de entrada (features) que se usarán para entrenar el modelo.

```
X = df.drop("Attrition_Yes", axis=1)
```

En la siguiente línea se elimina el atributo del dataframe llamado Attrition_Yes, esto se hace porque es la variable que se va a predecir.

```
y = df["Attrition_Yes"]
```

En la siguiente línea dividimos el dataset en dos partes:

80% para entrenamiento (X_train, y_train)

20% para prueba (evaluación) (X_test, y_test)

El parámetro random_state=42 asegura que la división siempre sea la misma (para reproducibilidad).

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

2. Estructura de la red neuronal

Descripción textual:

El modelo debe ser una red neuronal **binaria** que predice si un empleado abandonará la empresa o no. Tendrá dos capas ocultas con funciones de activación ReLU, y una salida con activación Sigmoid.

Descripción técnica de la red neuronal

Estructura de capas:

1. **Capa de entrada**:
    - Tipo: Dense
    - Neuronas: 64
    - Función de activación: 'relu'
    - input_shape: número de columnas (features) después del one-hot encoding
2. **Capa oculta intermedia**:
    - Tipo: Dense
    - Neuronas: 32
    - Función de activación: 'relu'
3. **Capa de salida**:
    - Tipo: Dense
    - Neuronas: 1 (una sola salida) ya que el problema es binario: predecir 0 (no se va) o 1 (se va)
    - Función de activación: 'sigmoid'

Compilación del modelo:

- **Optimizador**: 'adam'

Es eficiente y se adapta bien para problemas clasificatorios

- **Función de pérdida (loss)**: 'binary_crossentropy'

Se usa para clasificación binaria

- **Métrica**: 'accuracy'

Para evaluar cuántas predicciones son correctas

Parámetros de entrenamiento:

- **Épocas**: 50
- **Tamaño de lote (batch_size)**: 16

- **Validación**: validation_split=0.1

Parte 2: Web App (Gerencial)

¿Qué debe incluir?

- Un **formulario web** para ingresar datos de un empleado:
    - Edad
    - Ingreso mensual
    - Cargo
    - Años de experiencia
    - Años en la empresa
    - ¿Hace horas extra?
    - ¿Viaja por trabajo?
- Botón: **"¿Este empleado está en riesgo de irse?"**
- Resultado:
    - "Riesgo alto de rotación" o "Probabilidad baja de salida"
    - Valor numérico con probabilidad (%)

Entregables:

- index.html
- script.js
- Carpeta del modelo: modelo_attrition_web/
- README.md con:
    - Explicación del problema
    - Cómo fue entrenado el modelo
    - Capturas de la app
    - Conclusiones