

Documentación del Proyecto: Red Neuronal para Predicción de Abandono Estudiantil

Objetivo del Proyecto

Diseñar, entrenar y probar una red neuronal artificial (RNA) que prediga si un estudiante universitario continuará, abandonará o se graduará, con base en datos reales o simulados.

Tecnologías Utilizadas

- Python 3.11
- TensorFlow / Keras
- Pandas, NumPy, Scikit-learn
- Jupyter o Visual Studio Code (opcional)

Estructura del Proyecto

RED-NEURONAL/

```
├── data/          # Dataset base
│   └── data.csv
├── models/        # Modelo entrenado
│   └── dropout_model.h5
├── src/           # Código fuente organizado
│   ├── model.py
│   ├── preprocessing.py
│   └── main.py
├── predict.py     # Script de predicción independiente
├── requirements.txt # Dependencias del proyecto
└── venv/          # Entorno virtual (generado localmente)
    └── pyvenv.cfg
```

DATASET:

https://archive.ics.uci.edu/dataset/697/predict%2Bstudents%2Bdropout%2Band%2Bacademic%2Bsuccess?utm_source=chatgpt.com

Pasos para Levantar el Proyecto

1. Instalar Python 3.11

Verifica tu versión:

```
python --version
```

Si tienes Python 3.11.x ya estás listo.

2. Crear un entorno virtual

Desde tu carpeta del proyecto red-neuronal, ejecuta:

```
python -m venv venv
```

3. Activar el entorno virtual

Windows CMD:

```
venv\Scripts\activate
```

PowerShell:

```
.\venv\Scripts\Activate.ps1
```

4. Instalar dependencias necesarias

```
pip install --upgrade pip
```

```
pip install tensorflow pandas numpy scikit-learn matplotlib
```

5. Colocar el modelo entrenado

Asegúrate de tener el archivo dropout_model.h5 dentro de la carpeta models/.

6. Probar el modelo con una entrada real

Usa el script predict.py con el siguiente contenido:

Código: predict.py

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import load_model

# Cargar modelo
model = load_model('models/dropout_model.h5')

# Cargar datos originales
df = pd.read_csv('data/data.csv', sep=';')

# Igual que en el entrenamiento
drop_cols = ['Course', 'Nacionality', "Daytime/evening attendance\t"]
df.drop(columns=drop_cols, inplace=True)

# Codificar variables categóricas
```

```

le = LabelEncoder()
for col in df.select_dtypes(include='object').columns:
    df[col] = le.fit_transform(df[col])

# Separar features y target
X = df.drop('Target', axis=1)
y = df['Target']

# Escalar
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#  Elegir una fila de prueba (por ejemplo, la número 10)
fila = X_scaled[9].reshape(1, -1)

# Predecir
prediccion = model.predict(fila)
clase = np.argmax(prediccion)
clases = {0: 'Continúa', 1: 'Abandona', 2: 'Se gradúa'}

# Mostrar resultado
print(f"Resultado: {clases[clase]} (probabilidades: {prediccion[0]})")

```

7. Ejecutar el script

Con el entorno activado:

`python predict.py`

Ejemplo de salida:

Resultado: Abandona (probabilidades: [0.02 0.93 0.05])

Aplicaciones en Ingeniería de Sistemas

- Sistemas de alerta temprana
- Dashboard de monitoreo
- Optimización académica
- Proyectos de Machine Learning aplicados

8. Conclusion

Este proyecto muestra cómo aplicar redes neuronales para resolver un problema social y educativo. Al predecir el abandono estudiantil, los sistemas informáticos pueden integrarse en plataformas web de gestión académica para emitir alertas tempranas. Así, un ingeniero de sistemas puede contribuir al bienestar educativo mediante IA, creando soluciones inteligentes

basadas en datos reales.

8. ANEXOS

DATA

	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	Father's occu	Admission	gra	Displaced	Educational	sj	Debtors	Tuition fees u	Gender	Scholarship h	Age at enroll	International	Curricular unit					
2	9	127.3	1	0	0	1	1	1	0	20	0	0	0	0	0	0	0	0
3	3	142.5	1	0	0	0	0	1	0	19	0	0	6	6	6	14	0	0
4	9	124.8	1	0	0	0	0	1	0	19	0	0	6	0	0	0	0	0
5	3	119.6	1	0	0	0	1	0	0	20	0	0	6	8	6	13.4285714	0	0
6	9	141.5	0	0	0	0	1	0	0	45	0	0	6	9	5	12.3333333	0	0
7	7	114.8	0	0	0	1	1	1	0	50	0	0	5	10	5	11.8571429	0	0
8	10	128.4	1	0	0	1	0	1	18	0	0	7	9	7	13.3	0	0	0
9	9	113.1	1	0	0	0	0	1	0	22	0	0	5	5	0	0	0	0
10	9	129.3	0	0	0	0	1	0	1	21	1	0	6	8	6	13.875	0	0
11	7	123	1	0	1	0	0	0	18	0	0	6	9	5	11.4	0	0	0
12	7	130.6	1	0	0	0	1	0	0	18	0	0	6	6	6	12.3333333	0	0
13	9	119.3	1	0	0	0	1	0	1	18	0	0	8	8	7	13.2142857	0	0
14	9	130.2	1	0	0	0	1	0	0	19	0	0	6	6	0	0	0	0
15	7	111.8	1	0	0	0	1	0	1	21	0	0	6	7	6	10.5714286	0	0
16	5	137.1	1	0	0	0	1	0	1	18	0	0	5	7	4	13.25	0	0
17	3	120.7	1	0	0	0	1	0	0	20	0	0	6	6	5	13.2	0	0
18	8	137.4	1	0	0	0	1	0	0	18	0	0	6	10	1	12	0	0
19	4	127.3	1	0	0	0	1	0	0	18	0	0	7	8	7	13.30625	0	0
20	5	136.3	1	0	0	0	1	0	0	20	0	0	5	8	4	12.5	1	0
21	7	124.6	1	0	0	0	1	0	0	18	0	0	7	7	6	11.6666667	0	0
22	8	120.3	0	0	0	0	1	0	1	21	0	0	0	0	0	0	0	0
23	7	121.8	1	0	0	0	1	0	0	20	-	-	7	14	7	11.4375	0	0
24	1	125.5	1	0	0	0	1	0	0	18	0	0	8	12	7	12.8571429	0	0
25	7	114.9	1	0	0	0	1	0	1	19	0	0	6	8	6	13.375	0	0
26	7	123.9	0	0	0	0	1	0	0	19	0	0	7	8	6	13.2966667	0	0
27	9	157	1	0	1	1	1	0	1	18	0	0	6	8	5	11.6	0	0

ENTRENANDO LA RED NEURONAL CON LA DATA

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows the project structure:
 - RED-NEURONAL**: Contains `qodo`, `data`, `models` (with `dropout_model.h5`), and `src` (with `_pycache_`, `model.py`, `preprocessing.py`).
 - venv**: Contains `Include`, `Lib`, `Scripts`, `share`, and `pyvenv.cfg`.
 - Documentación**: Contains `Red_Neuronal_Abandono.docx` and `Red_Neuronal_Abandono.docx`.
- Code Editor:** The main area displays Python code for a neural network model:

```
18 # predict
19 # Para Col. de tipo select_types(includes= object).columns:
20
21 # Separar features y target
22 X=df.drop('Target', axis=1)
23 y = df['Target']
24
25 # Escalar
26 scaler = StandardScaler()
27 X_scaled = scaler.fit_transform(X)
28
29 # Elegir una fila de prueba (por ejemplo, la numero 10)
30 fila = X_scaled[9].reshape(1,-1)
31
32 # Predecir
```
- Terminal:** Shows the output of a pip install command:

```
collecting h5py-3.11.0           12.0/12.0 MB 7.7 MB/s eta 0:00:00
  Downloading h5py-3.14.0-cp311-cp311-win_amd64.whl (2.9 MB) 2.9/2.9 MB 6.8 MB/s eta 0:00:00
collecting ml-dtypes<1.0.0,>=0.5.1
  Downloading ml_dtypes-0.5.1-cp311-cp311-win_amd64.whl (209 kB)
collecting tensorflow-io-gcs-filesystem>=0.23.1
  Downloading tensorflow_io_gcs_filesystem-0.23.1-cp311-cp311-win_amd64.whl (1.5 MB) 1.5/1.5 MB 7.3 MB/s eta 0:00:00
collecting wheel<1.0,>=23.0
  Downloading wheel-0.45.1-py3-none-any.whl (72 kB) 72.5/72.5 kB ? eta 0:00:00
Collecting rich
  Downloading rich-14.0.0-py3-none-any.whl (243 kB) 243.2/243.2 kB 14.6 MB/s eta 0:00:00
collecting namex
  Downloading namex-0.1.0-py3-none-any.whl (5.9 kB)
collecting optree
  Downloading optree-0.16.0-cp311-cp311-win_amd64.whl (314 kB) 314.2/314.2 kB 9.8 MB/s eta 0:00:00
collecting charset_normalizer<4,>=2
```
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (highlighted), and PORTS.

PRUEBAS

The screenshot shows a Jupyter Notebook interface with several tabs open in the top bar: File, Edit, Selection, View, Go, Run, Terminal, Help, and red-neuronal [Administrator]. The left sidebar displays a file tree for a project named 'RED-NEURONAL' containing subfolders like '.godo', 'data', 'models', 'src', 'venv', and files such as 'main.py', 'dropout_model.h5', 'model.py', 'preprocessing.py', 'pyenv.cfg', 'Documentacion_Red_Neuronal_Abandono.docx', and 'predict.py'. The main area contains code in a code editor and a terminal window.

Code Editor:

```
predict.py
18 for col in df.select_dtypes(include='object').columns:
19     # Separar features y target
20     X = df.drop('target', axis=1)
21     y = df['target']
22
23     # Escalar
24     scaler = StandardScaler()
25     X_scaled = scaler.fit_transform(X)
26
27     # Elegir una fila de prueba (por ejemplo, la número 10)
28     fila = X_scaled[9].reshape(1, -1)
29
30     # Predecir
31
32
```

Terminal:

```
1/1 0s 54ms/step
Resultado: Se gradúa (probabilidades: [0.00586969 0.07389123 0.9202391])
(wenv) D:\ciclo-7-2025-IVANALISIS MULTIVARIADO\UNIDAD 3\red-neuronal>python predict.py
2025-06-15 21:03:16.847686: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-06-15 21:03:16.847686: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-06-15 21:03:16.847686: I tensorflow/core/util/port.cc:153] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_WINT FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
1/1 0s 48ms/step
Resultado: Continúa (probabilidades: [8.4489036e-01 1.5509813e-01 1.1475574e-05])
```