TRABAJO PRÁCTICO: APLICACIÓN DE IA

Introducción:

Carrera: Ingeniería en Informática

Materia: Inteligencia Artificial Aplicada

Ciclo Lectivo: Primer Cuatrimestre - 2024

Profesores: Dr. Ierache, Jorge - Dr. Becerra, Martín - Ing. Sanz, Diego

Grupo: N°3

Integrantes:

- Becerra, Diego Ezequiel
- Corrales, Mauro Exequiel
- Di Nicco, Luis Demetrio
- López Ferme, Nahuel Ezequiel
- Vivas, Pablo Ezequiel

Dominio Elegido:

El proyecto se centra en el ámbito educativo, específicamente en la Universidad Nacional de la Matanza (UNLaM).

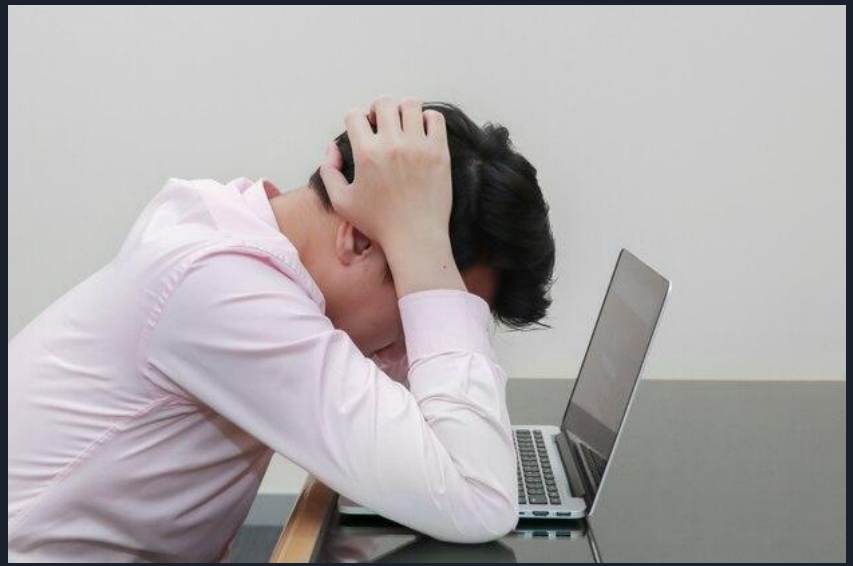
El objetivo es colaborar con una porción importante de la comunidad universitaria, particularmente con los estudiantes de la carrera de grado Ingeniería en Informática, buscando proporcionar una cursada más acorde a la disponibilidad horaria y a la probabilidad de éxito del alumnado.

Universidad Nacional de La Matanza

DIIT
Departamento de Ingeniería e Investigaciones Tecnológicas

Problema Identificado:

El problema que hemos identificado es la dificultad que tienen muchos estudiantes para organizar su cursada al inicio del cuatrimestre. Por inexperiencia o falta de tiempo, muchas veces terminan eligiendo sus materias en forma poco consciente, lo que puede llevar a que no obtengan el éxito esperado.



Solución Propuesta:

Por lo cual hemos desarrollado una aplicación con los conocimientos adquiridos en las materias de Inteligencia Artificial e Inteligencia Artificial Aplicada, que permitirá a los estudiantes de Ingeniería en Informática planificar sus horarios de manera más efectiva, teniendo en cuenta sus tiempos disponibles, trabajo y la probabilidad de éxito en cada materia.

Casos de Uso:

1) *Recomendar Cursada:*

El usuario ingresa la oferta de materias correspondiente al cuatrimestre actual, su historia académica, las materias pendientes de final que posee, su condición laboral y la cantidad de horas que tiene disponible para estudiar. El sistema genera una recomendación de materias en conjunto con el día y horario a cursar, en función a la probabilidad de éxito calculada y los datos proporcionados por el usuario.

2) *Calcular Probabilidad de Éxito de una Cursada:*

El usuario ingresa la cursada que le gustaría realizar, su condición laboral y la cantidad de horas que tiene disponible para estudiar. El sistema indica la probabilidad de éxito de la cursada ingresada en función de los datos proporcionados por el usuario.

1) Recomendar Cursada:

Tareas del Sistema:

1) Validar que los parámetros sean correctos.
2) Armar el mapa de las materias y sus correlativas de acuerdo al Plan 2023 de la carrera Ingeniería en Informática.
3) Procesar los archivos subidos por el usuario “Historia Académica” y “Oferta Académica”.
4) Procesar los datos ingresados por el usuario.
5) Calcular la situación académica del alumno.
6) Aplicar algoritmos genéticos para la generación y selección de cursadas.
7) Estimar probabilidad de éxito o fracaso de una cursada.
8) Mostrar la recomendación de cursada al usuario.

2) Calcular Probabilidad de Éxito de una Cursada:

Tareas del Sistema:

1) Validar que los parámetros sean correctos.
2) Procesar los datos ingresados por el usuario.
3) Calcular la situación académica del alumno.
4) Estimar probabilidad de éxito o fracaso de la cursada ingresada por el usuario.
5) Mostrar la probabilidad de éxito de la Cursada al usuario.

Esquema de Pipeline

1) Descarga e Importación de Librerías

Como primer paso de este proyecto, se realiza la descarga e importación de todas las librerías que van a ser usadas durante el desarrollo de toda la solución.

Descargas

```
[ ]  #Libreria para leer los archivos PDF
     !pip install pdfplumber
```

```
Collecting pdfplumber
  Downloading pdfplumber-0.11.0-py3-none-any.whl (56 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.4/56.4 kB 1.3 MB/s eta 0:00:00
Collecting pdfminer.six==20231228 (from pdfplumber)
  Downloading pdfminer.six-20231228-py3-none-any.whl (5.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.6/5.6 MB 45.4 MB/s eta 0:00:00
Requirement already satisfied: Pillow>=9.1 in /usr/local/lib/python3.10/dist-packages (from pdfplumber) (9.4.0)
Collecting pypdfium2>=4.18.0 (from pdfplumber)
  Downloading pypdfium2-4.30.0-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 51.2 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from pdfminer.six==20231228->pdfplumber) (3.3.2)
Requirement already satisfied: cryptography>=36.0.0 in /usr/local/lib/python3.10/dist-packages (from pdfminer.six==20231228->pdfplumber) (42.0.7)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-packages (from cryptography>=36.0.0->pdfminer.six==20231228->pdfplumber) (1.16.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.12->cryptography>=36.0.0->pdfminer.six==20231228->pdfplumber) (2.22)
Installing collected packages: pypdfium2, pdfminer.six, pdfplumber
Successfully installed pdfminer.six-20231228 pdfplumber-0.11.0 pypdfium2-4.30.0
```

Imports

```
[ ]  import random                      # Para generar números aleatorios y realizar operaciones aleatorias.
     import networkx as nx              # Para la creación, manipulación y estudio de estructuras, grafos y funciones de redes complejas.
     import matplotlib.pyplot as plt    # Para crear gráficos y visualizaciones a partir de datos.
     import csv                         # Para leer y escribir archivos CSV.
     import pdfplumber                  # Para extraer texto, tablas e imágenes de archivos PDF.
     import os
     from copy import deepcopy          # Para realizar copias profundas de objetos, útil para evitar referencias no deseadas entre objetos.
     import numpy as np                 # Para realizar la normalizacion de la aptitud con la funcion sigmoide y para implementar la red neuronal
     import tensorflow as tf            # Para implementar la red neuronal
     import pandas as pd                # Para implementar la red neuronal
     from tensorflow.keras import layers   # Para implementar la red neuronal
```

2) Procesamiento del Plan de Carrera

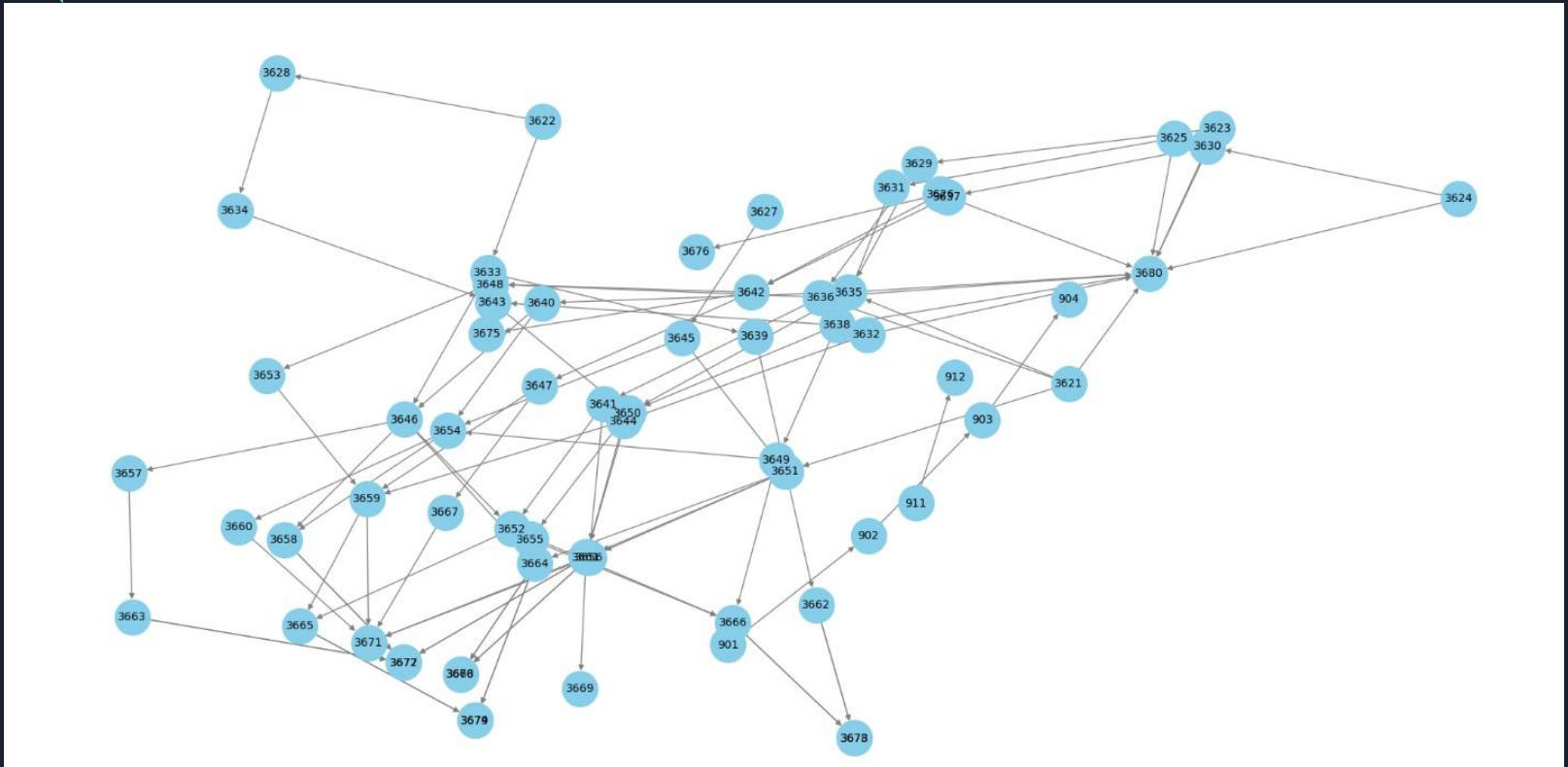
Aquí se crea una estructura para procesar el Plan 2023 de la Carrera de Ingeniería Informática, incluyendo algunos datos adicionales como la dificultad de cada materia y las horas de clase, estudio y práctica que requieren.

```
[ ]  #Clase Materia
     class Materia:
         def __init__(self, codigo, nombre, dificultad, horas_clase, horas_estudio, horas_practica, rama, año):
             self.codigo = codigo
             self.nombre = nombre
             self.dificultad = dificultad
             self.horas_clase = horas_clase
             self.horas_estudio = horas_estudio
             self.horas_practica = horas_practica
             self.rama = rama
             self.año = año
```

Mapa de Materias:

```
[ ]  #Mapa de Materias
     mapa_materias = {
         "3621": {"nombre": "Matemática Discreta", "dificultad": 4, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 2, "rama": "Ciencias Básicas", "año": "1"},
         "3622": {"nombre": "Análisis Matemático I", "dificultad": 8, "horas_clase": 4, "horas_estudio": 4, "horas_practica": 5, "rama": "Ciencias Básicas", "año": "1"},
         "3623": {"nombre": "Programación Inicial", "dificultad": 6, "horas_clase": 4, "horas_estudio": 1, "horas_practica": 4, "rama": "Programación", "año": "1"},
         "3624": {"nombre": "Introducción a los Sistemas de Información", "dificultad": 4, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 2, "rama": "Desarrollo de SW", "año": "1"},
         "3625": {"nombre": "Sistemas de Numeración", "dificultad": 5, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 2, "rama": "Infraestructura", "año": "1"},
         "3626": {"nombre": "Principios de Calidad de Software", "dificultad": 2, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 1, "rama": "Calidad y Seguridad de la Información", "año": "
         "3627": {"nombre": "Álgebra y Geometría Analítica I", "dificultad": 7, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 3, "rama": "Ciencias Básicas", "año": "1"},
         "3628": {"nombre": "Física I", "dificultad": 8, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 4, "rama": "Ciencias Básicas", "año": "1"},
         "3629": {"nombre": "Programación Estructurada Básica", "dificultad": 6, "horas_clase": 4, "horas_estudio": 1, "horas_practica": 4, "rama": "Programación", "año": "1"},
         "3630": {"nombre": "Introducción a la Gestión de Requisitos", "dificultad": 5, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 2, "rama": "Desarrollo de SW", "año": "1"},
         "3631": {"nombre": "Fundamentos de Sistemas Embebidos", "dificultad": 5, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 2, "rama": "Infraestructura", "año": "1"},
         "3632": {"nombre": "Introducción a los Proyectos Informáticos", "dificultad": 4, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 1, "rama": "Gestión y Complementarias", "año": "1"},
         "3633": {"nombre": "Análisis Matemático II", "dificultad": 9, "horas_clase": 4, "horas_estudio": 4, "horas_practica": 5, "rama": "Ciencias Básicas", "año": "2"},
         "3634": {"nombre": "Física II", "dificultad": 9, "horas_clase": 4, "horas_estudio": 4, "horas_practica": 4, "rama": "Ciencias Básicas", "año": "2"},
         "3635": {"nombre": "Tópicos de Programación", "dificultad": 7, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 4, "rama": "Programación", "año": "2"},
         "3636": {"nombre": "Bases de Datos", "dificultad": 7, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 4, "rama": "Programación", "año": "2"},
         "3637": {"nombre": "Análisis de Sistemas", "dificultad": 6, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 3, "rama": "Desarrollo de SW", "año": "2"},
         "3638": {"nombre": "Arquitectura de Computadoras", "dificultad": 6, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 3, "rama": "Infraestructura", "año": "2"},
         "3676": {"nombre": "Responsabilidad Social Universitaria", "dificultad": 2, "horas_clase": 4, "horas_estudio": 1, "horas_practica": 1, "rama": "Otras", "año": "2"},
         "3639": {"nombre": "Análisis Matemático III", "dificultad": 6, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 3, "rama": "Ciencias Básicas", "año": "2"},
         "3640": {"nombre": "Algoritmos y Estructuras de Datos", "dificultad": 8, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 4, "rama": "Programación", "año": "2"},
         "3641": {"nombre": "Bases de Datos Aplicadas", "dificultad": 7, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 4, "rama": "Programación", "año": "2"},
         "3642": {"nombre": "Principios de Diseño de Sistemas", "dificultad": 7, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 3, "rama": "Desarrollo de SW", "año": "2"},
         "3643": {"nombre": "Redes de Computadoras", "dificultad": 7, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 3, "rama": "Infraestructura", "año": "2"},
         "3644": {"nombre": "Gestión de las Organizaciones", "dificultad": 4, "horas_clase": 4, "horas_estudio": 6, "horas_practica": 1, "rama": "Gestión y Complementarias", "año": "2"},
         "3680": {"nombre": "Taller de Integración", "dificultad": 4, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 2, "rama": "Otras", "año": "2"},
         "3645": {"nombre": "Álgebra y Geometría Analítica II", "dificultad": 6, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 3, "rama": "Ciencias Básicas", "año": "3"},
         "3646": {"nombre": "Paradigmas de Programación", "dificultad": 8, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 5, "rama": "Programación", "año": "3"},
         "3647": {"nombre": "Requisitos Avanzados", "dificultad": 5, "horas_clase": 4, "horas_estudio": 1, "horas_practica": 4, "rama": "Desarrollo de SW", "año": "3"},
         "3648": {"nombre": "Diseño de Software", "dificultad": 9, "horas_clase": 4, "horas_estudio": 4, "horas_practica": 4, "rama": "Desarrollo de SW", "año": "3"},
         "3649": {"nombre": "Sistemas Operativos", "dificultad": 8, "horas_clase": 4, "horas_estudio": 4, "horas_practica": 3, "rama": "Infraestructura", "año": "3"},
         "3650": {"nombre": "Seguridad de la Información", "dificultad": 4, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 1, "rama": "Calidad y Seguridad de la Información", "año": "3"},
         "3675": {"nombre": "Práctica Profesional Supervisada", "dificultad": 1, "horas_clase": 4, "horas_estudio": 1, "horas_practica": 1, "rama": "Otras", "año": "3"},
         "3651": {"nombre": "Probabilidad y Estadística", "dificultad": 7, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 3, "rama": "Ciencias Básicas", "año": "3"},
         "3652": {"nombre": "Programación Avanzada", "dificultad": 8, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 4, "rama": "Programación", "año": "3"},
         "3653": {"nombre": "Arquitectura de Sistemas Software", "dificultad": 6, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 3, "rama": "Desarrollo de SW", "año": "3"},
         "3654": {"nombre": "Virtualización de Hardware", "dificultad": 9, "horas_clase": 4, "horas_estudio": 2, "horas_practica": 6, "rama": "Infraestructura", "año": "3"},
         "3655": {"nombre": "Auditoria y Legislación", "dificultad": 4, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 1, "rama": "Calidad y Seguridad de la Información", "año": "3"},
         "3656": {"nombre": "Estadística Aplicada", "dificultad": 7, "horas_clase": 4, "horas_estudio": 3, "horas_practica": 3, "rama": "Ciencias Básicas", "año": "4"},
```

***Grafo de Correlatividad*:**



3) Procesamiento de la situación del usuario

En esta sección, se procesa la información del usuario para hacerle una recomendación personalizada. Para lo cual se le solicitan los siguientes datos:

1) Cuántas horas semanales va a dedicar a la cursada en el cuatrimestre
2) Si trabaja actualmente
3) Un archivo pdf con la oferta de materias
4) Un archivo pdf con su historia académica
5) Las materias que tiene pendiente de final

4) Creación del Algoritmo Genético

Para realizar la recomendación al usuario, se utilizó algoritmos genéticos. En este contexto, los individuos serán las diferentes opciones de cursada, las cuales se representarán como un conjunto de materias.

A cada cursada se le agrega una cantidad aleatoria de materias. Solo se añaden materias que el usuario está en condición de cursar, respetando los días y turnos de la oferta de materias para evitar superposición de horarios.

Se utiliza la función de Aptitud para evaluar a los individuos. Se considera que cuanta mayor Aptitud tenga, mejor es el individuo.

Función de Aptitud:

La función de Aptitud tiene en cuenta 6 parámetros:

1) La Probabilidad de Éxito de la cursada (valor entre 0 a 100)
2) La dificultad de la cursada (valor entero entre 1 a 10)
3) La cantidad requerida de horas semanales (valor entero entre 4 y 168)
4) La cantidad de materias que puede llegar a desbloquear instantáneamente (valor entero positivo)
5) La cantidad de materias que puede llegar a desbloquear hasta el final de la carrera (valor entero positivo)
6) La cantidad de Materias de los primeros años para el título intermedio

El resultado de la función surge de la siguiente operación:

Aptitud = Probabilidad de Éxito - Penalización de Horas - Penalización de Dificultad + Premio Materias Instantáneas Desbloqueadas + Premio Materias Totales Desbloqueadas + Premio materias primeros años

A) Selección de los mejores individuos:

En esta sección del algoritmo genético se utiliza un método de selección por torneo, en el cual se escoge de forma aleatoria dos individuos de la población, y el que posea mayor aptitud se selecciona para que se reproduzca y también para que pase a la siguiente generación. El perdedor es descartado.

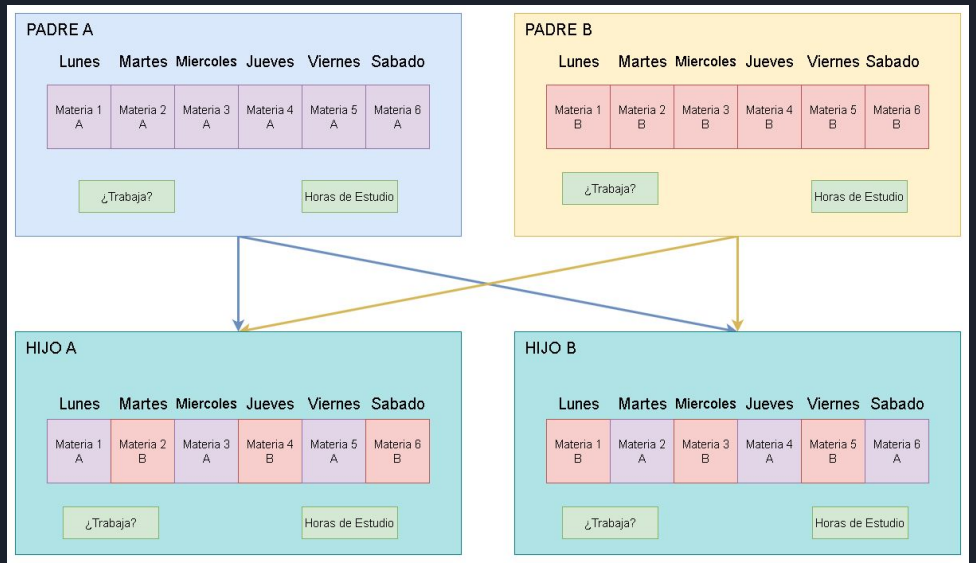
De esta forma se logra mantener cierto grado de diversidad en la población y se evita que el algoritmo converja rápidamente.

En esta primera selección, se elige al 50% que ganó su torneo para que pase a la siguiente generación. El 50% restante surgirá como resultado de la cruza de los individuos vencedores en el torneo.

B) Cruza de Individuos:

Se utiliza una cruza multipunto para conseguir una mayor diversidad entre padres e hijos y así poder explorar todas las combinaciones de cursadas rápidamente.

Hay 5 puntos de cruza, uno por cada día luego del lunes, de esta manera, las materias de los padres quedarían intercaladas en base a los días.



C) Mutación de Individuos:

Se le realiza una mutación simple a la población conformada por los padres e hijos del paso anterior.

Se generan distintos tipos de mutaciones para aumentar la diversidad de la población:

1) No realizar ninguna mutación (Probabilidad: 70%)
2) Agregar una materia al individuo (Probabilidad: 10%)
3) Intercambiar horarios entre materias del mismo individuo (Probabilidad: 10%)
4) Eliminar una materia al individuo (Probabilidad: 10%)

En total hay un 30% de probabilidades que el individuo sufra alguna mutación y un 70% de probabilidades de que no se le realice ningún cambio.

D) Evaluación de la Población Resultante:

La técnica de reemplazo que se utiliza es “Gap Generacional”. Se evalúa a la población resultante luego de realizar la mutación y realizamos una selección por torneo para quedarnos con el 50% de la población.

Este 50% seleccionado se combina con el 50% seleccionado en el primer proceso de selección, para pasar a la siguiente generación.

Al finalizar evaluamos la aptitud general de la población y continuamos iterando a partir del primer proceso de selección.

Se definen dos condiciones de corte:

1) Se alcanza límite máximo de 20 generaciones (iteraciones)
2) Se alcanza un límite de 5 generaciones sin mejora de aptitud poblacional, es decir, se produce un estancamiento.

Una vez obtenida la población final, se realiza una selección elitista donde se elige al cromosoma (cursada) de mayor aptitud. Esa es la cursada que se le recomendará al usuario.

5) Aumentación de Datos

En esta sección se crea un dataset artificial para simular un historial de cursadas de alumnos, con el fin de poder entrenar la red neuronal que predecirá la probabilidad de éxito de una cursada. Se utiliza un segundo algoritmo genético para generar y seleccionar los individuos más “realistas”.

Se comienza ingresando el historial de cursadas de los integrantes del grupo, las cuales se utilizan como punto de partida para inicializar la población y también como referencia para plantear la función de aptitud.

Luego, se define el segundo algoritmo genético para generar los datos de entrenamiento de la Red Neuronal.

Una vez finalizada toda esta sección, se obtiene un dataset final de 100.000 tuplas para entrenar la Red Neuronal.

6) Creación de la Red Neuronal

El objetivo de la red Neuronal es poder estimar la probabilidad de éxito o fracaso de una cursada, la cual es necesaria para calcular la aptitud de un individuo en el algoritmo genético que realizará la recomendación final al usuario.

La probabilidad de éxito se estima a partir de las características de una cursada (como su dificultad y cantidad de horas de clase y estudio), así como los datos ingresados por el usuario (horas semanales que va a dedicar al estudio y si trabaja).

Dicha Red Neuronal está implementada en este Sistema por medio de las librerías TensorFlow y Keras.

Keras

A) Generación de los DataSets:

Antes de crear y entrenar la Red Neuronal, se generan 3 datasets a partir del dataset original de 100.000 tuplas (obtenido en la Sección de Aumentación de Datos):

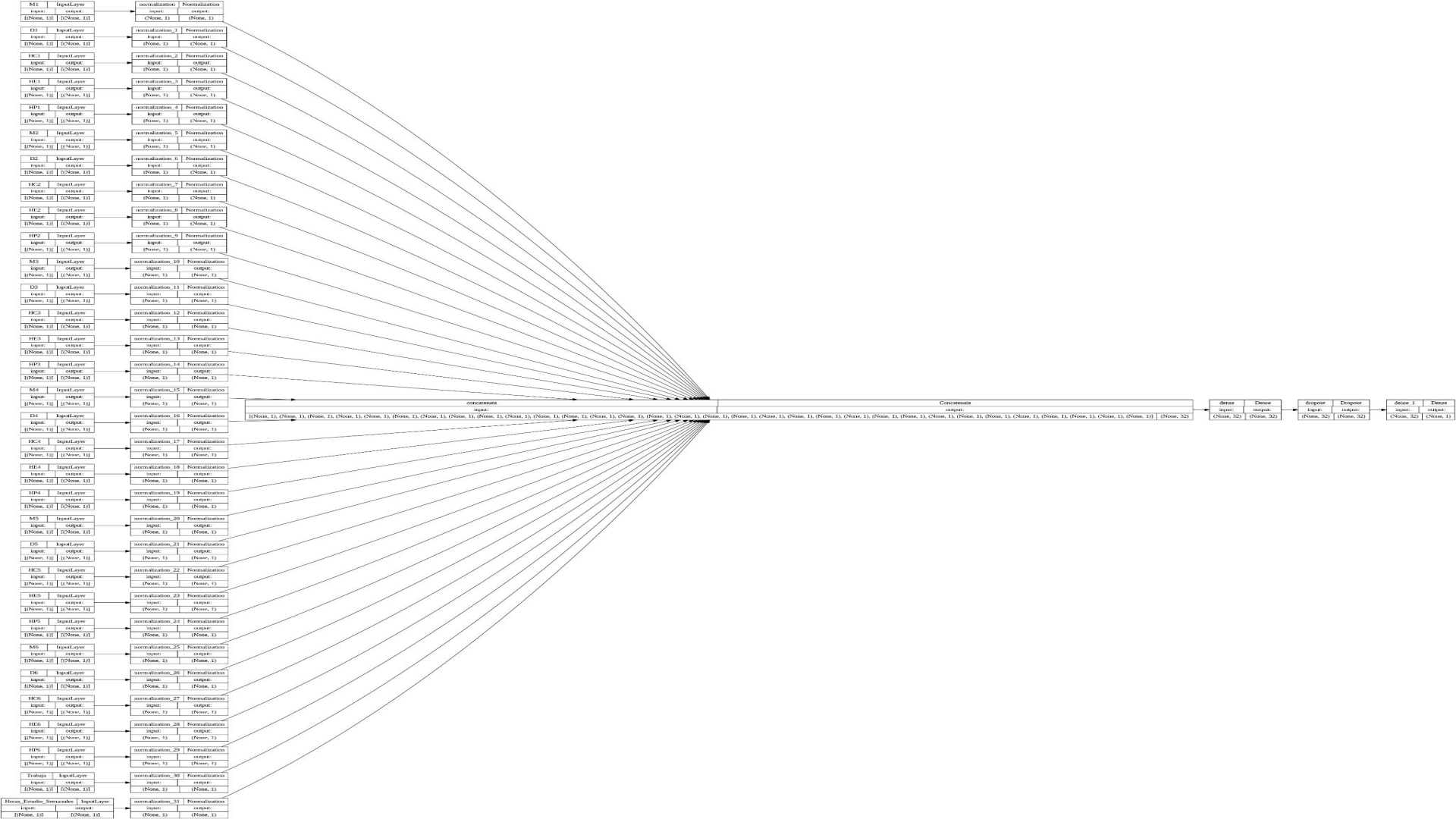
1) train_ds: Dataset de entrenamiento que consta de 80.000 tuplas
2) val_ds: Dataset de validación que consta de 10.000 tuplas
3) test_ds: Dataset de prueba que consta de 10.000 tuplas

El tamaño de lote de los 3 datasets es de 256

B) Definición del Modelo:

La red neuronal se estructura de la siguiente forma:

- **32 Capas de Entrada,** que representan cada parámetro del DataSet (6 códigos de materias, 6 dificultades correspondientes a cada materia, 6 horas de clase, 6 horas de estudio y 6 horas de práctica correspondiente a cada materia, si trabaja y la cantidad de horas semanales de estudio).
- **32 Capas de Normalización de datos**, una para capa de entrada, a fin de estabilizar el entrenamiento de la red y mejorar la convergencia del algoritmo.
- **1 Capa de Concatenación**, que se encarga de concatenar todas las features normalizadas en un único tensor, para luego ser pasado a través de capas posteriores del modelo.
- **1 Capa Densa de 32 neuronas**, que recibe el tensor de la capa de Concatenación y realiza una función de activación ReLu en cada una de sus neuronas.
- **1 Capa DropOut**, con una tasa de abandono del 50%, es decir que la mitad de los valores provenientes de la capa Densa se establecerán en cero utilizando un patrón de selección aleatorio. Por lo cual la salida de esta capa tendrá 16 valores en 0 y 16 que conservarán su valor original de la capa anterior. Esto se realiza principalmente para prevenir el overfitting, generando que el modelo sea más robusto y menos propenso a memorizar el ruido en los datos de entrenamiento.
- **1 Capa Densa de salida con una sola neurona** y sin ninguna función de activación. La salida de esta última capa es la que se utiliza para realizar las predicciones.


M1 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization Normalization input: output: (None, 1) (None, 1)
D1 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_1 Normalization input: output: (None, 1) (None, 1)
HC1 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_2 Normalization input: output: (None, 1) (None, 1)
HE1 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_3 Normalization input: output: (None, 1) (None, 1)
HP1 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_4 Normalization input: output: (None, 1) (None, 1)
M2 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_5 Normalization input: output: (None, 1) (None, 1)
D2 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_6 Normalization input: output: (None, 1) (None, 1)
HC2 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_7 Normalization input: output: (None, 1) (None, 1)
HE2 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_8 Normalization input: output: (None, 1) (None, 1)
HP2 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_9 Normalization input: output: (None, 1) (None, 1)
M3 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_10 Normalization input: output: (None, 1) (None, 1)
D3 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_11 Normalization input: output: (None, 1) (None, 1)
HC3 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_12 Normalization input: output: (None, 1) (None, 1)
HE3 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_13 Normalization input: output: (None, 1) (None, 1)
HP3 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_14 Normalization input: output: (None, 1) (None, 1)
M4 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_15 Normalization input: output: (None, 1) (None, 1)
D4 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_16 Normalization input: output: (None, 1) (None, 1)
HC4 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_17 Normalization input: output: (None, 1) (None, 1)
HE4 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_18 Normalization input: output: (None, 1) (None, 1)
HP4 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_19 Normalization input: output: (None, 1) (None, 1)
M5 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_20 Normalization input: output: (None, 1) (None, 1)
D5 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_21 Normalization input: output: (None, 1) (None, 1)
HC5 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_22 Normalization input: output: (None, 1) (None, 1)
HE5 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_23 Normalization input: output: (None, 1) (None, 1)
HP5 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_24 Normalization input: output: (None, 1) (None, 1)
M6 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_25 Normalization input: output: (None, 1) (None, 1)
D6 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_26 Normalization input: output: (None, 1) (None, 1)
HC6 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_27 Normalization input: output: (None, 1) (None, 1)
HE6 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_28 Normalization input: output: (None, 1) (None, 1)
HP6 InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_29 Normalization input: output: (None, 1) (None, 1)
Trabaja InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_30 Normalization input: output: (None, 1) (None, 1)
Horas_Estudio_Semanales InputLayer input: output: [(None, 1)] [(None, 1)]
normalization_31 Normalization input: output: (None, 1) (None, 1)
concatenate input: [(None, 1), (None, 1)] output: (None, 32)
Concatenate input: [(None, 1), (None, 1)] output: (None, 32)
dense Dense input: (None, 32) output: (None, 32)
dropout Dropout input: (None, 32) output: (None, 32)
dense_1 Dense input: (None, 32) output: (None, 1)

C) Entrenamiento del Modelo:

Se entrena al modelo con 50 Epochs de 313 iteraciones cada una, debido a que el dataset de entrenamiento posee 80000 tuplas y lotes de 256.

```
[ ] # Definimos el callback EarlyStopping
    early_stopping = EarlyStopping(
        monitor='val_accuracy',     # monitoreamos la precisión en el conjunto de validación
        patience=15,                # número de epochs con no mejora antes de detener
        restore_best_weights=True   # restaura los mejores pesos al finalizar
    )

    #Entrenamos con 50 Epochs de 313 iteraciones c/u (Batches de 256)
    model.fit(train_ds, epochs=50, validation_data=val_ds, callbacks=[early_stopping])
```

```
Epoch 23/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2139 - accuracy: 0.8954 - val_loss: 0.2032 - val_accuracy: 0.9006
Epoch 24/50
313/313 [==============================] - 5s 12ms/step - loss: 0.2132 - accuracy: 0.8965 - val_loss: 0.2035 - val_accuracy: 0.9002
Epoch 25/50
313/313 [==============================] - 4s 9ms/step - loss: 0.2126 - accuracy: 0.8964 - val_loss: 0.2032 - val_accuracy: 0.8989
Epoch 26/50
313/313 [==============================] - 4s 10ms/step - loss: 0.2115 - accuracy: 0.8963 - val_loss: 0.2023 - val_accuracy: 0.8998
Epoch 27/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2113 - accuracy: 0.8972 - val_loss: 0.2022 - val_accuracy: 0.9015
Epoch 28/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2106 - accuracy: 0.8969 - val_loss: 0.2020 - val_accuracy: 0.9016
Epoch 29/50
313/313 [==============================] - 5s 12ms/step - loss: 0.2107 - accuracy: 0.8973 - val_loss: 0.2013 - val_accuracy: 0.9005
Epoch 30/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2102 - accuracy: 0.8965 - val_loss: 0.2015 - val_accuracy: 0.8986
Epoch 31/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2103 - accuracy: 0.8976 - val_loss: 0.2016 - val_accuracy: 0.9021
Epoch 32/50
313/313 [==============================] - 5s 13ms/step - loss: 0.2094 - accuracy: 0.8974 - val_loss: 0.2013 - val_accuracy: 0.9015
Epoch 33/50
313/313 [==============================] - 4s 9ms/step - loss: 0.2092 - accuracy: 0.8982 - val_loss: 0.2006 - val_accuracy: 0.9017
Epoch 34/50
313/313 [==============================] - 4s 9ms/step - loss: 0.2089 - accuracy: 0.8977 - val_loss: 0.2005 - val_accuracy: 0.9004
Epoch 35/50
313/313 [==============================] - 5s 12ms/step - loss: 0.2082 - accuracy: 0.8974 - val_loss: 0.2000 - val_accuracy: 0.9021
Epoch 36/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2078 - accuracy: 0.8983 - val_loss: 0.1998 - val_accuracy: 0.9039
Epoch 37/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2077 - accuracy: 0.8978 - val_loss: 0.1999 - val_accuracy: 0.9022
Epoch 38/50
313/313 [==============================] - 5s 10ms/step - loss: 0.2086 - accuracy: 0.8981 - val_loss: 0.1995 - val_accuracy: 0.9026
Epoch 39/50
313/313 [==============================] - 4s 8ms/step - loss: 0.2078 - accuracy: 0.8986 - val_loss: 0.2001 - val_accuracy: 0.9026
Epoch 40/50
```

D) Evaluación del Modelo:

Utilizando el DataSet de Prueba, se determina el Accuracy y el Loss que posee el Modelo.

Paso 5.F) Evaluacion

```
[ ]  #Evaluamos Accuracy
     accuracy = model.evaluate(test_ds)[1]
     print("Accuracy", accuracy)
```

```
40/40 [==============================] - 0s 9ms/step - loss: 0.2012 - accuracy: 0.9004
Accuracy 0.9003999829292297
```

```
[ ]  #Evaluacion Loss
     loss = model.evaluate(test_ds)[0]
     print("Loss", loss)
```

```
40/40 [==============================] - 0s 8ms/step - loss: 0.2012 - accuracy: 0.9004
Loss 0.20118089020252228
```

7) Recomendación

En esta sección final se pone en marcha el algoritmo genético que obtendrá la cursada con mayor aptitud para ser recomendada al usuario. Dicho algoritmo trabajará con poblaciones de 200 individuos, y obtenida la población final se procederá a realizar una selección elitista para quedarse con el individuo más apto. Una vez obtenida la cursada ideal para el usuario, se procederá a mostrar la misma en forma de un calendario semanal, para que el usuario pueda visualizar de forma más amigable la recomendación de cursada.

A lo largo de todo el proyecto se utilizó el entorno de desarrollo colaborativo en línea “Google Colab”. Para acceder al archivo del proyecto y ver el código de esta implementación utilice el siguiente link: https://colab.research.google.com/drive/1pFh9c7hmZTvyRx3zXRiVC7QLZVLXdbik?usp=drive_link