UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA
INF 354 – INTELIGENCIA ARTIFICIAL

“Predict Students' Dropout and Academic Success”

SIGLA	MATERIA	Fecha:	00-06-2025
CI	APELLIDOS Y NOMBRES	MATERIA	NOTA
9909789	ALIAGA YUJRA EDWIN	INF 354	
8302380	JURADO EVER EMERSON	INF 354	
13986139	TICONA LAURA YOEL	INF 354	

Docente: PhD. Moises Martin Silva Choque

Fecha: 11/06/2025

La Paz - Bolivia

ÍNDICE

1. Descripción del Dataset y Objetivo de Investigación

Descripción del Dataset

El dataset utilizado en este proyecto se titula "Predict Students' Dropout and Academic Success" y fue obtenido desde el repositorio oficial UCI Machine Learning Repository.

Este conjunto de datos fue creado en el marco de un proyecto cuyo objetivo principal es contribuir a la reducción de la deserción académica y al fracaso en la educación superior, mediante el uso de técnicas de aprendizaje automático que permitan identificar a los estudiantes en riesgo en una etapa temprana de su trayectoria académica, con el fin de implementar estrategias de apoyo oportunas.

Cada instancia del conjunto de datos representa a un estudiante e incluye información disponible al momento de la matrícula, además de datos sobre su trayectoria académica, perfil demográfico y situación socioeconómica. El problema se plantea como una tarea de clasificación en tres categorías posibles según el estado final del estudiante al concluir el curso:

- Dropout (abandono)
- Enrolled (matriculado)
- Graduate (graduado)

Este dataset fue desarrollado con el respaldo del programa SATDAP – Capacitação da Administração Pública, bajo la subvención POCI-05-5762-FSE-000191, contiene **4,424 registros** de estudiantes universitarios con **37 variables** (36 predictoras + 1 objetivo).

De acuerdo a la documentación original, se aplicó un preprocesamiento riguroso para tratar valores anómalos, casos atípicos y valores faltantes, asegurando así la calidad de los datos. Además, se recomienda una división de los datos en un 80% para entrenamiento y 20% para prueba durante el desarrollo del modelo predictivo.

Contenido del dataset:

- Variables académicas (calificaciones por semestre, número de materias, tipo de curso, modalidad, etc.)
- Factores personales (estado civil, edad, género, nacionalidad)
- Información familiar y económica (educación y ocupación de los padres, becas, deudas, matrícula al día)
- Factores macroeconómicos (tasa de desempleo, inflación, PIB)
- **Variable objetivo**: Target
    - dropout: El estudiante abandonó los estudios.
    - enrolled: El estudiante continúa cursando.
    - graduate: El estudiante completó el programa exitosamente.

Características Técnicas del Dataset

- **Dataset Characteristics**: Tabular
- **Subject Area**: Social Science
- **Associated Tasks**: Classification
- **Feature Type**: Real, Categorical, Integer
- **# Instances**: 4424
- **# Features**: 36

Objetivo de Investigación

El objetivo general de esta investigación es identificar tempranamente a los estudiantes en riesgo de deserción en la educación superior, utilizando modelos de aprendizaje automático y basándose en información disponible desde la etapa inicial de su vida universitaria.

El propósito es facilitar estrategias de apoyo personalizadas para mejorar la retención estudiantil y el rendimiento académico global de las instituciones, tomando en cuenta tanto variables académicas como personales, económicas y sociales.

2. Proceso Básico de Análisis de Datos

a) Preprocesamiento de Datos

Paso 1.- Se imputaron los valores faltantes en las variables numéricas utilizando la media, ya que el dataset está compuesto principalmente por datos numéricos. Este paso asegura que no se pierda información útil por filas incompletas, manteniendo la consistencia del conjunto de datos.

Paso 2.- Se aplicó codificación de variables categóricas usando **LabelEncoder**, específicamente en la variable objetivo (**Target**), que contiene tres categorías: *Dropout*, *Enrolled* y *Graduate*. Esto permite representar las clases de forma numérica, lo cual es necesario para que los algoritmos de clasificación puedan procesarlas.

Paso 3.- Se normalizaron las características numéricas utilizando **MinMaxScaler**, transformándolas al rango [0, 1]. Esto permite que todas las variables tengan el mismo peso durante el entrenamiento de los modelos supervisados.

Sin embargo, para el **PCA**, se aplicó **StandardScaler** (media 0, desviación estándar 1), ya que este método es sensible a la escala de los datos y requiere estandarización para representar correctamente la varianza.

Paso 4.- Se abordó el desbalance de clases aplicando la técnica de **oversampling** con **RandomOverSampler**, que replica aleatoriamente ejemplos de las clases minoritarias (*Dropout* y *Enrolled*) hasta igualar la cantidad de instancias con la clase mayoritaria (*Graduate*). También se consideró el uso de **undersampling** (RandomUnderSampler), que elimina ejemplos de la clase mayoritaria; sin embargo, se descartó para evitar pérdida de información.

b) Selección y Justificación del Clasificador

El problema planteado en este proyecto es de **aprendizaje supervisado**, ya que se dispone de una variable objetivo denominada **Target**, que clasifica a los estudiantes en tres categorías: **Dropout** (desertor), **Enrolled** (matriculado) y **Graduate** (graduado). La tarea consiste en predecir la categoría de un estudiante a partir de sus características académicas y socioeconómicas. Para llevar a cabo esta predicción, se seleccionó un **clasificador** adecuado que pueda procesar eficazmente las características del conjunto de datos.

Tras evaluar varias opciones, se decidió optar por el **Random Forest** como clasificador. Este modelo es un "metaestimador que ajusta una serie de árboles de decisión en diversas submuestras del conjunto de datos y utiliza el promedio de las predicciones para mejorar la precisión y controlar el sobreajuste" (Pedregosa et al., 2011). En términos sencillos, **Random Forest** emplea un conjunto de árboles de decisión entrenados sobre diferentes subconjuntos del dataset, lo que mejora su rendimiento general y reduce el riesgo de sobreajuste, algo común en los árboles de decisión individuales.

Razones para Elegir Random Forest

La elección de un clasificador depende de la naturaleza del problema y las características del conjunto de datos. **Random Forest** se eligió por su robustez y versatilidad, debido a las siguientes razones clave:

- **Manejo de Datos Mixtos**: El conjunto de datos utilizado en este proyecto contiene tanto variables numéricas como categóricas. **Random Forest** es capaz de manejar ambos tipos de datos de manera eficiente. Esta capacidad es crucial, ya que otros clasificadores, como la **regresión logística** o las **máquinas de soporte vectorial (SVM)**, podrían necesitar transformaciones adicionales para manejar adecuadamente las variables categóricas, como la codificación one-hot.

- **Robustez frente a Datos Ruidosos**: **Random Forest** es especialmente resistente a los valores atípicos o ruidosos en los datos. En problemas reales, como el análisis del rendimiento académico de los estudiantes, los datos pueden contener errores o inconsistencias. El modelo de **Random Forest** puede manejar este tipo de ruidos sin una pérdida significativa en la calidad del modelo.

- **No Requiere Escalado de Datos**: A diferencia de otros modelos como **SVM** o **k-NN**, **Random Forest** no depende de que los datos sean escalados o normalizados, lo que simplifica el proceso de preprocesamiento y mejora la eficiencia. Aunque el escalado puede ser útil en ciertos casos (por ejemplo, en la reducción de dimensionalidad con **PCA**), no es un requisito para el modelo en sí.

- **Capacidad para Capturar Relaciones No Lineales**: **Random Forest** es adecuado para datasets con relaciones complejas y no lineales entre las variables. En este proyecto, el

análisis del abandono o éxito académico de los estudiantes implica interacciones no lineales entre diversas características, lo cual es precisamente el tipo de problema que **Random Forest** puede modelar sin la necesidad de especificar explícitamente estas relaciones, a diferencia de modelos como la **regresión logística**, que asume relaciones lineales.

- **Importancia de las Características**: Según Breiman (2001), **Random Forest** tiene la capacidad de calcular la **importancia de las características**, lo que permite identificar qué variables son más relevantes para la predicción. Esta propiedad es especialmente útil en contextos académicos o de investigación, donde es importante comprender qué factores tienen un mayor impacto en el rendimiento de los estudiantes o en su probabilidad de deserción.

Comparación con Otros Clasificadores

Aunque existen otros clasificadores que podrían haber sido considerados para este problema, como la **regresión logística**, las **máquinas de soporte vectorial (SVM)** y **k-NN**, **Random Forest** fue seleccionado debido a sus ventajas específicas:

- **Regresión Logística**: Es un modelo simple y fácil de interpretar, pero es menos adecuado para problemas donde las relaciones entre las variables son no lineales, como es el caso en este proyecto. Además, la **regresión logística** no maneja bien las variables categóricas sin una transformación adicional, lo que la hace menos eficiente en este contexto.

- **Máquinas de Soporte Vectorial (SVM)**: **SVM** es eficaz para clasificación binaria y multiclase, pero requiere un ajuste cuidadoso de parámetros, como la selección del **kernel**, lo que puede hacerlo costoso computacionalmente. Además, **SVM** no maneja bien datasets grandes sin optimización adecuada, lo cual podría ser un desafío con el tamaño de este conjunto de datos.

- **k-NN (k-Nearest Neighbors)**: Aunque es un clasificador sencillo y efectivo, **k-NN** presenta dos grandes limitaciones: es computacionalmente intensivo, especialmente con conjuntos de datos grandes, y depende fuertemente del escalado de los datos. Además, **k-NN** no tiene un modelo explícito, lo que lo hace menos eficiente al manejar datasets complejos.

Ventajas Clave de Random Forest

- **Reducción del Sobreajuste**: Gracias a la combinación de múltiples árboles de decisión, **Random Forest** es menos susceptible al sobreajuste que un solo árbol de decisión. Al promediar las predicciones de varios árboles, el modelo mejora su capacidad de generalización.

- **No Requiere Escalado**: Como ya se mencionó, **Random Forest** no depende del escalado de los datos, lo que hace que el proceso de preprocesamiento sea más simple y el modelo más flexible.

- **Manejo de Datos Mixtos**: **Random Forest** puede manejar tanto variables numéricas como categóricas sin necesidad de preprocesamiento complejo, lo que lo convierte en una opción ideal para conjuntos de datos heterogéneos.

- **Interpretabilidad**: Aunque **Random Forest** no es tan interpretable como un único árbol de decisión, ofrece información útil sobre la **importancia de las características**, lo que facilita la interpretación y comprensión de los resultados.

c) Primera Ejecución del Modelo

Se realizó una primera evaluación del modelo Random Forest utilizando una partición del 80% para entrenamiento y 20% para prueba, sobre el dataset balanceado y preprocesado.

A continuación, se presentan las métricas globales de desempeño obtenidas:

```
====== MÉTRICAS DEL MODELO ======
Accuracy:  0.8937
Precision: 0.8961
Recall:    0.8938
F1-score:  0.8938
```

- **Accuracy** indica que el modelo clasificó correctamente el 89.37% de los casos totales.
- **Precision** (macro promedio) muestra que, en promedio, el 89.61% de las predicciones positivas fueron correctas.
- **Recall** (macro promedio) indica que el modelo detectó correctamente el 89.38% de las instancias verdaderas.
- **F1-score** (macro promedio) refleja un balance armonioso entre precisión y recall con un valor de 0.8938.

El detalle por clase se muestra en el siguiente reporte de clasificación:

```
====== REPORTE DE CLASIFICACIÓN ======
              precision    recall  f1-score   support

     Dropout       0.95      0.89      0.92       444
    Enrolled       0.85      0.93      0.89       439
    Graduate       0.89      0.86      0.88       443
```

```
accuracy                           0.89      1326
macro avg       0.90      0.89      0.89      1326
weighted avg       0.90      0.89      0.89      1326
```

- La clase **Dropout** tiene una alta precisión (95%), lo que significa que casi todas las predicciones de abandono fueron correctas, aunque con un recall de 89%, indicando que algunos estudiantes en abandono no fueron detectados.
- La clase **Enrolled** presenta un recall más alto (93%), lo que indica que la mayoría de estudiantes activos fueron correctamente identificados, con una precisión del 85%.
- La clase **Graduate** muestra un equilibrio entre precision (89%) y recall (86%), con un buen desempeño general.

Finalmente, la matriz de confusión, que muestra la distribución de las predicciones respecto a las etiquetas reales, es la siguiente:

```
====== MATRIZ DE CONFUSIÓN ======
[[394  23  27]
 [ 10 410  19]
 [ 12  50 381]]
```

Interpretación de la matriz:

- De los 444 estudiantes que abandonaron (Dropout), 394 fueron correctamente clasificados, 23 fueron clasificados erróneamente como Enrolled y 27 como Graduate.
- De los 439 estudiantes activos (Enrolled), 410 fueron correctamente identificados, 10 fueron clasificados como Dropout y 19 como Graduate.
- De los 443 estudiantes graduados (Graduate), 381 fueron correctamente predichos, mientras que 12 fueron clasificados como Dropout y 50 como Enrolled.

d) Validación por Asignaciones (Splits)

Para evaluar la confiabilidad y robustez del modelo Random Forest, se realizaron 100 particiones (splits) aleatorias del dataset, considerando dos configuraciones distintas de división de los datos:

- **División 80% entrenamiento / 20% prueba**, utilizada con fines académicos.
- **División 50% entrenamiento / 50% prueba**, realizada con fines de investigación.

En cada una de las 100 particiones, se entrenó y evaluó el modelo calculando las métricas principales de desempeño: Accuracy, Precision, Recall y F1-score.

Los resultados obtenidos, expresados en términos de la mediana de cada métrica para las 100 ejecuciones, son los siguientes:

Validación 80/20 y Validación 50/50

```
Validación por Asignaciones (80/20) completada con 100 splits.
Mediana Accuracy : 0.9042
Mediana Precision: 0.9058
Mediana Recall   : 0.9043
Mediana F1-score : 0.9042

Validación por Asignaciones (50/50) completada con 100 splits.
Mediana Accuracy : 0.8771
Mediana Precision: 0.8803
Mediana Recall   : 0.8767
Mediana F1-score : 0.8775
```

Estos resultados muestran que el modelo mantiene un desempeño consistente y robusto frente a distintas particiones del dataset, con una ligera disminución en las métricas al reducir la proporción de datos para entrenamiento en la configuración 50/50, como es esperado.

e) Código Fuente

El código fuente completo utilizado para el análisis, preprocesamiento, entrenamiento y validación del modelo está disponible públicamente en el repositorio de GitHub:

https://github.com/I1vI/Proyecto-Final-IA.git

Este repositorio contiene todos los scripts y módulos necesarios para reproducir los experimentos descritos en este trabajo, facilitando la transparencia y la replicabilidad del estudio.

3. Reducción de Dimensionalidad con PCA

a) Análisis de Componentes Principales (PCA)

El análisis de componentes principales, o PCA, reduce el número de dimensiones en grandes conjuntos de datos a componentes principales que conservan la mayor parte de la información original. Para ello, transforma las variables potencialmente correlacionadas en un conjunto más pequeño de variables, denominadas componentes principales.

El PCA se utiliza comúnmente para el preprocesamiento de datos para su uso con algoritmos de machine learning. Puede extraer las características más informativas de grandes conjuntos de datos al tiempo que conserva la información más relevante del conjunto de datos inicial. Esto reduce la complejidad del modelo, ya que la adición de cada nueva característica repercute negativamente en el rendimiento del modelo, lo que también se conoce comúnmente como la "maldición de la dimensionalidad".

La covarianza (cov) mide la fuerza de correlación de dos o más variables. La matriz de covarianzas resume las covarianzas asociadas a todas las combinaciones de pares de las variables iniciales del conjunto de datos. Calcular la matriz de covarianza ayuda a identificar las relaciones entre las variables, es decir, cómo las variables varían de la media entre sí. Esta matriz de datos es una matriz simétrica, lo que significa que las combinaciones de variables pueden representarse como d × d, siendo d el número de dimensiones. Por ejemplo, para un conjunto de datos tridimensional, habría 3 × 3 o 9 combinaciones de variables en la matriz de covarianza.

El signo de las variables de la matriz nos dice si las combinaciones están correlacionadas:

- Positivo (las variables están correlacionadas y aumentan o disminuyen al mismo tiempo)
- Negativo (las variables no están correlacionadas, lo que significa que una disminuye y la otra aumenta)
- Cero (las variables no están relacionadas entre sí)

Para reducir la dimensionalidad del dataset y facilitar el análisis, se aplicó el **Análisis de Componentes Principales (PCA)**. Previamente, los datos fueron estandarizados utilizando **StandardScaler**, ya que el PCA es sensible a la escala de los datos. La estandarización asegura que todas las variables tengan media 0 y desviación estándar 1, permitiendo que el PCA identifique correctamente las direcciones de mayor varianza.

Se seleccionaron los **10 primeros componentes principales**, que representan nuevas variables construidas como combinaciones lineales de las variables originales. A continuación se presenta una muestra del dataset transformado tras aplicar PCA:

```
=== Análisis de Componentes Principales (PCA) con 10 componentes===
          PC1       PC2       PC3       PC4  ...       PC7       PC8       PC9      PC10
0   -6.062406 -0.773853 -0.329544 -1.074159  ... -1.377528 -1.872446 -1.070326 -1.107411
1   -0.133997 -1.313900 -0.645377 -1.490190  ...  1.045517  0.724783  1.225191 -0.202988
2   -3.989897  0.118819 -0.030606 -0.182097  ... -2.192649 -0.622003  1.169341  0.660392
3    0.550911 -1.081313 -0.159752  1.441998  ... -0.327282 -0.111888  0.314063  0.407051
4    0.555002  2.821617 -0.407978  2.924284  ... -0.491476 -0.027011 -0.620863 -1.080874
...       ...       ...       ...       ...  ...       ...       ...       ...       ...
6622  0.229716 -0.816836  0.200971  1.094675  ... -0.551118  0.627511 -0.079006 -0.261476
6623 -0.725175 -0.450420  1.454871 -4.315579  ... -3.169689  0.968459  0.582193 -0.815070
6624  1.580546 -1.622708 -0.291538  0.541549  ... -0.398350  0.116387 -0.683570  1.028436
6625  1.139144  1.189871 -0.395035  0.270603  ...  1.979053  0.306448  0.969967  0.386164
6626  1.490188  0.832304 -0.471235 -1.735745  ... -0.740684 -1.408912 -0.913266 -0.711452

[6627 rows x 10 columns]
```

En cuanto a la **varianza explicada por cada componente principal**, los valores obtenidos fueron:

```
Varianza explicada por componente: [0.16177652 0.09453631 0.06906959 0.05785592 0.05354558 0.04592502
 0.04471776 0.04315065 0.03739322 0.03420326]
```

Esto indica que el primer componente principal (PC1) captura aproximadamente el 16.18% de la varianza total del conjunto de datos, seguido por el segundo componente (PC2) que explica un 9.45% adicional y así sucesivamente.

La varianza acumulada para los 10 componentes es:

```
Varianza explicada acumulada: [0.16177652 0.25631283 0.32538242 0.38323834 0.43678392 0.48270894
 0.5274267  0.57057735 0.60797057 0.64217384]
```

Esto significa que los 10 primeros componentes capturan aproximadamente el **64.22%** de la varianza total del dataset, lo cual permite reducir significativamente la dimensionalidad manteniendo buena parte de la información original.

Esta transformación permite simplificar el dataset para posteriores análisis y modelados, manteniendo la mayor cantidad posible de información original y facilitando la detección de patrones y relaciones en los datos.

b) 5 ejecuciones con distintas cantidades de componentes y la cantidad óptima de componentes.

Para determinar la cantidad óptima de componentes principales necesarias para mantener o mejorar el rendimiento del modelo, se realizaron múltiples ejecuciones del modelo Random Forest utilizando datasets transformados con diferentes cantidades de componentes PCA: 14, 12, 11, 10, 9, 5 y 3.

Para cada cantidad de componentes, se realizaron 50 iteraciones de validación por asignaciones (splits) con una división 80% entrenamiento y 20% prueba, y se calcularon las métricas medianas de desempeño: Accuracy, Precision, Recall y F1-score.

Los resultados obtenidos fueron los siguientes:

```
=== Evaluación con distintas cantidades de componentes PCA ===

Probando con 14 componentes principales:
Mediana Accuracy : 0.8839
Mediana Precision: 0.8852
Mediana Recall   : 0.8844
Mediana F1-score : 0.8841

Probando con 12 componentes principales:
Mediana Accuracy : 0.8808
Mediana Precision: 0.8822
Mediana Recall   : 0.8809
Mediana F1-score : 0.8808
```

```
Probando con 10 componentes principales:
Mediana Accuracy : 0.8771
Mediana Precision: 0.8790
Mediana Recall   : 0.8772
Mediana F1-score : 0.8767

Probando con 11 componentes principales:
Mediana Accuracy : 0.8786
Mediana Precision: 0.8793
Mediana Recall   : 0.8783
Mediana F1-score : 0.8781

Probando con 9 componentes principales:
Mediana Accuracy : 0.8752
Mediana Precision: 0.8768
Mediana Recall   : 0.8753
Mediana F1-score : 0.8754

Probando con 5 componentes principales:
Mediana Accuracy : 0.8680
Mediana Precision: 0.8706
Mediana Recall   : 0.8686
Mediana F1-score : 0.8678

Probando con 3 componentes principales:
Mediana Accuracy : 0.8330
Mediana Precision: 0.8333
Mediana Recall   : 0.8332
Mediana F1-score : 0.8319

=== Mejor configuración encontrada ===
Componentes principales: 14
Mediana Accuracy : 0.8839
Mediana Precision: 0.8852
Mediana Recall   : 0.8844
Mediana F1-score : 0.8841
```

La mejor configuración se obtuvo utilizando **14 componentes principales**, alcanzando una mediana de Accuracy de **0.8839** y un F1-score de **0.8841**, lo que refleja un buen equilibrio entre precisión (0.8852) y recall (0.8844).

Si bien configuraciones con 9, 10, 11 y 12 componentes mantienen métricas relativamente cercanas, ninguna supera de forma consistente el rendimiento alcanzado con 14 componentes.

Por otro lado, al reducir excesivamente la cantidad de componentes (por ejemplo, a 5 o 3), se observa una caída importante en el desempeño del modelo, indicando que una

reducción demasiado agresiva de la dimensionalidad conduce a la pérdida de información relevante para la predicción.

c) Explicación técnica de cómo funciona PCA desde el punto de vista del álgebra lineal (vectores, matrices, autovalores, etc.)

El Análisis de Componentes Principales (PCA, por sus siglas en inglés) es una técnica fundamental en estadística y aprendizaje automático para reducir la dimensionalidad de un conjunto de datos, preservando la mayor cantidad posible de varianza. Su base teórica se encuentra en el álgebra lineal, especialmente en los conceptos de vectores, matrices, autovalores y autovectores, para ello principalmente son los pasos:

1. Presentación de los datos: Supongamos que tienes un conjunto de datos

   X X con n observaciones y d variables (dimensiones). Podemos representar esto como una matriz $X \in \Re^{n \times d}$, donde cada fila es una observación y cada columna es una variable.

   Supongamos que tienes el siguiente conjunto de datos con dos características (dimensiones):

$$X = \begin{bmatrix} 2 & 3 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \\ 6 & 6 \end{bmatrix}$$

   Estas son 5 observaciones con 2 características. Queremos reducir la dimensionalidad de estos datos y obtener las componentes principales.

2. Primero, necesitamos centrar los datos restando la media de cada columna. Calculamos la media para cada dimensión (columna):
   - Media de la primera columna: $\mu_1 = \frac{2+3+4+5+6}{5} = 4$
   - Media de la segunda columna: $\mu_2 = \frac{3+3+4+5+6}{5} = 4.2$

   Entonces, restamos la media de cada columna:

$$X_{centrado} = \begin{bmatrix} 2-4 & 3-4.2 \\ 3-4 & 3-4.2 \\ 4-4 & 4-4.2 \\ 5-4 & 5-4.2 \\ 6-4 & 6-4.2 \end{bmatrix} = \begin{bmatrix} -2 & -1.2 \\ -1 & -1.2 \\ 0 & -0.2 \\ 1 & 0.8 \\ 2 & 1.8 \end{bmatrix}$$

3. Calcular la matriz de covarianza

   La matriz de covarianza nos dice cómo varían las características en relación entre sí. Se calcula con la siguiente fórmula:

$$\sum = \frac{1}{n-1} X_{centrado} X_{centrado}$$

$$X_{centrado}X_{centrado} = \begin{bmatrix} (-2)^2+(-1)^2+0^2+1^2+2^2 & (-2)(-1.2)+(-1)(-1.2)+0(-.02)+1(0.8)+2(1.8) \\ (-2)(-1.2)+(-1)(-1.2)+0(-.02)+1(0.8)+2(1.8) & (-1.2)^2+(-1.2)^2+(-0.2)^2+0.8^2+1.8^2 \end{bmatrix}$$

$$X_{centrado}X_{centrado} = \begin{bmatrix} 10 & 7.6 \\ 7.6 & 5.2 \end{bmatrix}$$

4. Calcular los autovalores y autovectores

   Ahora, necesitamos encontrar los autovalores λ y autovectores v de la matriz de covarianza Σ Esto se hace resolviendo el siguiente sistema:

$$det(\sum - \lambda I) = 0$$

$$\begin{bmatrix} 2.5-\lambda & 1.9 \\ 1.9 & 1.3-\lambda \end{bmatrix} = 0$$

$$det(\sum) = (2.5-\lambda)(1.3-\lambda) + (1.9)(1.9)$$

$$det(\sum) = (2.5)(1.3) - 2.5*\lambda - 1.3*\lambda - \lambda^2$$

$$det(\sum) = (2.5*1.3 - 2.5*\lambda - \lambda - 1.3 + \lambda^2) - (1.9^2)$$

$$det(\sum) = 3.25 - 2.5\lambda - 1.3\lambda + \lambda^2 - 3.61$$

$$det(\sum) = \lambda^2 - 3.8\lambda - 0.36$$

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\lambda = \frac{-(-3.8) \pm \sqrt{(-3.8)^2 - 4(1)(-3.6)}}{21}$$

$$\lambda = \frac{3.8 \pm 3.99}{2}$$

$$\lambda_1 = \frac{3.8 + 3.99}{2} = 3.895 \quad , \quad \lambda_2 = \frac{3.8 - 3.99}{2} = -0.095$$

para el calculo del los auto vectores se usa: $(\sum \; - \; \lambda I) \; * \; v = \; 0$

donde λ se reemplaza por los valores que hallamos

$$\begin{pmatrix} 2.5 - 3.895 & 1.9 \\ 1.9 & 1.3 - 3.895 \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix} = 0$$

$$\begin{pmatrix} -1.395 & 1.9 \\ 1.9 & -2.595 \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix} = 0$$

$$\begin{Bmatrix} -1.395x + 1.9y = 0 \\ 1.9x - 2.595y = 0 \end{Bmatrix}$$

$$0.505x - 0.695y = 0$$

$$0.695y = 0.505x$$

para la segunda variable

$$\begin{pmatrix} 2.5 - (-0.095) & 1.9 \\ 1.9 & 1.3 - (-0.095) \end{pmatrix} = 0$$

$$\begin{pmatrix} 2.595 & 1.9 \\ 1.9 & 1.395 \end{pmatrix} = 0$$

$$\begin{Bmatrix} 2.595x + 1.9y = 0 \\ 1.9x + 1.395y = 0 \end{Bmatrix}$$

$$4.495x + 3.295y = 0$$

$$3.295y = 4.495x$$

$$v_1 = \begin{pmatrix} 0.695 \\ 0.505 \end{pmatrix}, v_2 = \begin{pmatrix} 3.295 \\ -4.495 \end{pmatrix}$$

5. Proyección de los datos Finalmente, proyectamos los datos centrados XcentradoX_{\text{centrado}}Xcentrado sobre los autovectores seleccionados. En este caso, seleccionamos el autovector correspondiente al autovalor más grande λ1, ya que queremos la mayor varianza explicada.

   El nuevo conjunto de datos proyectado Z es:

$$Z = X_{centrado} v_1$$

$$Z = \begin{bmatrix} -2 & -1.2 \\ -1 & -1.2 \\ 0 & -0.2 \\ 1 & 0.8 \\ 2 & 1.8 \end{bmatrix} * \begin{pmatrix} 0.695 \\ 0.505 \end{pmatrix}$$

$$Z = \begin{bmatrix} -1.996 \\ -1.301 \\ -0.101 \\ 1.099 \\ 2.299 \end{bmatrix}$$

4. Aprendizaje No Supervisado

El **aprendizaje no supervisado** se refiere a un tipo de aprendizaje en el que el sistema debe identificar patrones o estructuras subyacentes en los datos sin contar con etiquetas previamente definidas (es decir, sin la variable objetivo, en nuestro caso el **Target**). En este proceso, el modelo toma decisiones basándose únicamente en las características del conjunto de datos, agrupando los elementos según su similitud y generando, así, clases (clusters) a partir de patrones detectados en los datos.

Según la Real Academia Española (RAE), el término **cluster** se define como un "grupo de elementos similares o cercanos, o agrupados en función de alguna característica o variable" (RAE, s.f.). En el contexto de nuestro estudio, podemos entender un **cluster** como un conjunto

de datos agrupados en función de características comunes. Por lo tanto, el objetivo del **clustering** es particionar un conjunto de datos en **subclases significativas**, donde cada grupo o **cluster** contiene datos que son similares entre sí, pero distintos a los de otros clusters.

En el aprendizaje no supervisado, el objetivo es predecir las etiquetas de los datos (en nuestro caso, las clases **Dropout**, **Enrolled**, **Graduate**) sin tener información previa sobre a qué clase pertenece cada punto de datos. Esta tarea puede realizarse con métodos como el **clustering difuso** (fuzzy clustering), que permite que un punto pertenezca a varios clusters con diferentes **grados de pertenencia**. Es decir, los datos no tienen que ser separables de manera estricta en grupos, lo cual es útil cuando hay solapamiento entre las categorías. Sin embargo, en nuestro caso, dado que las clases están claramente definidas y separables, optamos por la técnica de **K-means**, que es adecuada para particionar los datos en **grupos bien definidos**.

Para implementar el **K-means**, es indispensable que los datos estén previamente escalados. Esto se debe a que el algoritmo de **K-means** se basa en la medida de distancias (usualmente la distancia Euclidiana) entre los puntos de datos, y si las características tienen diferentes escalas, algunas podrían dominar el proceso de agrupamiento. Por esta razón, en el preprocesamiento de los datos, hemos utilizado **StandardScaler** para estandarizar las características y asegurarnos de que todas tengan la misma importancia en el proceso de clustering.

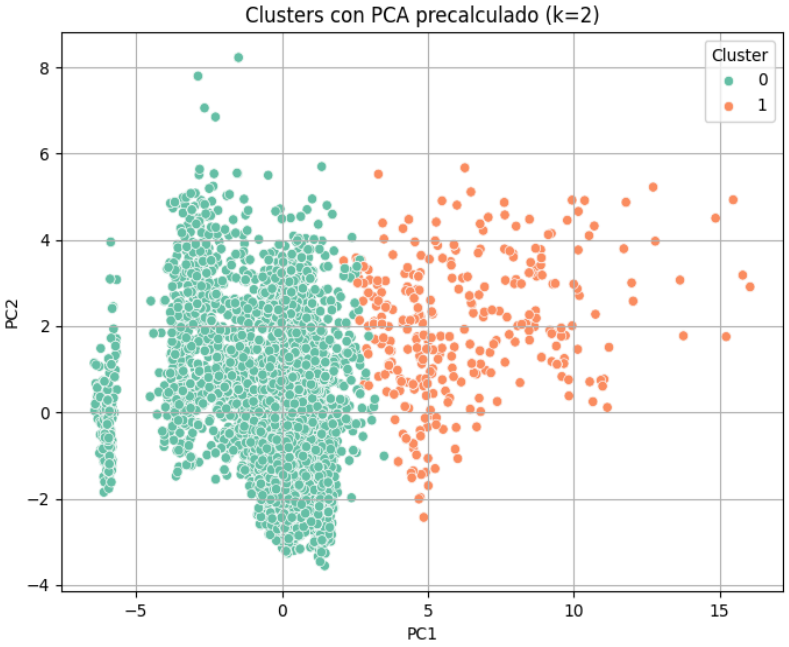
Una de las principales dificultades al aplicar **K-means** es determinar el número óptimo de clusters, **n_clusters**. Aunque algunas veces se puede hacer una suposición inicial sobre el número de clusters basándose en el conocimiento del dominio o en el número de características del dataset, como regla general, el número de clusters no debería depender directamente de la cantidad de variables (o características), sin embargo, existen varias técnicas para abordar este problema:

1. **Silhouette Score**: Este índice mide qué tan bien se agrupan los puntos dentro de un mismo cluster y cuán distintos están entre clusters. Su valor oscila entre -1 y +1, donde valores cercanos a +1 indican que los puntos están bien agrupados y alejados de otros clusters. El **Silhouette Score** es útil para encontrar el número de clusters que maximiza la cohesión interna y la separación entre los grupos.

2. **Elbow Method (Método del Codo)**: Este método identifica el punto en el que añadir más clusters ya no mejora significativamente la calidad del modelo. El **codo** en la gráfica de **WCSS (within-cluster sum of squares)** indica el número de clusters en el que la mejora comienza a estabilizarse.

Para este proyecto, hemos elegido el **Silhouette Score** como criterio para determinar el número óptimo de clusters, ya que esta métrica proporciona una evaluación más directa de la calidad del clustering en función de la cohesión y separación de los grupos durante las iteraciones.

Con este enfoque, obtuvimos los siguientes resultados:




```
=== Análisis de Componentes Principales (PCA) con 14 componentes ===
Varianza explicada por componente: [0.16177652 0.09453631 0.06906959 0.05785592 0.05354558
0.04592502
 0.04471776 0.04315065 0.03739322 0.03420326 0.02916098 0.02859319
 0.02803587 0.02635924]
Varianza explicada acumulada: [0.16177652 0.25631283 0.32538242 0.38323834 0.43678392
0.48270894
 0.5274267  0.57057735 0.60797057 0.64217384 0.67133482 0.69992802
 0.72796389 0.75432313]
Mejor número de clusters encontrado: k = 2

Distribución de clases reales por cluster (proporción por fila):
Target     0     1     2
Cluster
0       0.34  0.34  0.33
1       0.28  0.28  0.44
```

Como parte de la preparación de los datos, aplicamos la técnica de **Análisis de Componentes Principales (PCA)** para reducir la dimensionalidad. Tras evaluar el número óptimo de componentes, se seleccionaron **14 componentes principales**, los cuales explican un **75.43%** de la varianza acumulada en los datos. Esto significa que el **75.43%** de la información útil de los

datos originales se conserva en estas 14 dimensiones, lo que permite representar los datos de manera eficiente sin perder gran parte de la variabilidad original.

Posteriormente, se utilizó el algoritmo **K-means** para realizar el clustering de los datos, y mediante la evaluación del **Silhouette Score**, se determinó que el **número óptimo de clusters es 2**. Esto indica que los datos se agrupan de manera más efectiva en dos grupos distintos.

Los resultados del clustering, obtenidos con **K-means**, muestran la distribución de las clases reales (0: Dropout, 1: Enrolled, 2: Graduate) dentro de los dos clusters formados:

- **Cluster 0 (Dropout)**:
    - **34%** de los estudiantes son de clase **0** (Dropout)
    - **34%** de los estudiantes son de clase **1** (Enrolled)
    - **33%** de los estudiantes son de clase **2** (Graduate)
- **Cluster 1 (Enrolled)**:
    - **28%** de los estudiantes son de clase **0** (Dropout)
    - **28%** de los estudiantes son de clase **1** (Enrolled)
    - **44%** de los estudiantes son de clase **2** (Graduate)

Estos resultados indican que el algoritmo ha agrupado de forma mixta a los estudiantes en cada cluster, con una distribución razonable entre las tres clases. Sin embargo, se observa un solapamiento entre las clases, especialmente en el **Cluster 1 (Enrolled)**, que contiene una mayor proporción de estudiantes clasificados como **Graduate**. Esto sugiere que, aunque el algoritmo de clustering ha identificado dos grupos principales, aún hay cierta superposición entre las clases reales en los clusters generados.

Referencias

[x] Tello, J. C., & Informáticos, S. (2007). Reconocimiento de patrones y el aprendizaje no supervisado. Universidad de Alcalá, Madrid.

[x] Real Academia Española. (s.f.). **Clúster**. *Diccionario de la lengua española* (23.ª ed.). Recuperado de RAE

[x] Balbachan, F., & Dell'Era, D. (2008). **Técnicas de clustering para inducción de categorías sintácticas en un corpus de español**. INFOSUR Nro 2-Agosto 2008, 95.

5. Entrega de Artículos Académicos

Toda la documentación de este proyecto se presentará en **dos artículos académicos**: el primero, centrado en **PCA y Aprendizaje No Supervisado**, y el segundo, en el **proyecto completo**, que abarca tanto el **aprendizaje supervisado como no supervisado**. Ambos artículos se estructuran en formato académico.

Además, todo el proceso de implementación del proyecto se ha subido a un **repositorio de GitHub**, donde se utilizó **Python** dentro de un entorno virtual. El repositorio incluye el **código fuente**, las **librerías utilizadas** y un documento **README**, que proporciona una guía detallada para ejecutar el proyecto en un entorno local.

Proyecto en GitHub: Proyecto de GitHub en IA	
Artículo 1: Enfocado en PCA + Aprendizaje No Supervisado	https://docs.google.com/document/d/1y_uxRW-VA6bki8HUIJgiHNGJUlCwnggA/edit?usp=drive_link&ouid=109564262664899956335&rtpof=true&sd=true
Artículo 2: Proyecto completo (supervisado + no supervisado)	

5. Referencias:

- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (2nd ed.). Springer.

- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Random Forest Classifier. *scikit-learn 0.24.1 documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- Tello, J. C., & Informáticos, S. (2007). "Reconocimiento de patrones y el aprendizaje no supervisado." Universidad de Alcalá, Madrid .

- Real Academia Española (RAE). (s.f.). "Clúster." *Diccionario de la lengua española* (23.ª ed.). Recuperado de RAE .

- Balbachan, F., & Dell'Era, D. (2008). "Técnicas de clustering para inducción de categorías sintácticas en un corpus de español." INFOSUR Nro 2-Agosto 2008 .
-