In [1]:
```
import pandas as pd
import numpy as np
import hashlib
import uuid
```

In [2]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.impute import KNNImputer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, roc_auc_score
```

1. CARGA DE DATOS

In [3]:
```
df = pd.read_csv("datos_moodle.csv")

print("Datos cargados correctamente")
print(df.head())
```

```
Datos cargados correctamente
   codigo_estudiante  edad sexo periodo_academico  nota_primer_parcial  \
0           20248998    21    M           2025-II                    7
1           20248134    19    M           2025-II                    5
2           20254712    20    F           2025-II                   12
3           20257253    17    M           2025-II                   14
4           20256938    20    M           2025-II                   19

   nota_segundo_parcial  nota_tercer_parcial  promedio_previo  \
0                    17                   20              7.5
1                    18                   15             10.9
2                    13                   16              7.7
3                    11                   15             14.2
4                    19                   16             15.2

   total_accesos_lms  tiempo_total_conexion_min  tareas_entregadas  \
0                 81                        623                  7
1                273                       1999                  8
2                177                       1247                 10
3                380                       2611                  7
4                135                        761                  9

   tareas_totales  participaciones_foro  asistencias  clases_totales  \
0              10                     6           29              30
1              10                     9           18              30
2              10                     8           21              30
3              10                     7           25              30
4              10                    12           25              30

   retroalimentaciones_docente  nota_final
0                           17          15
1                           15          13
2                           12          14
3                           20          13
4                           11          18
```

2. ANONIMIZACION (Ley 29733)

In [4]:
```
def anonymize_id(value, salt):
    return hashlib.sha256(f"{value}{salt}".encode()).hexdigest()

salt = uuid.uuid4().hex
df["id_anonimo"] = df["codigo_estudiante"].apply(lambda x: anonymize_id(x, salt))
df.drop(columns=["codigo_estudiante"], inplace=True)

print("Anonimización aplicada")
```

```
Anonimización aplicada
```

3. VARIABLE OBJETIVO (RIESGO ACADEMICO)

In [5]:
```
df["riesgo_academico"] = (df["nota_final"] < 11).astype(int)
```

4. INGENIERIA DE CARACTERISTICAS

In [6]:
```
df["ratio_tareas"] = df["tareas_entregadas"] / df["tareas_totales"]
df["ratio_asistencia"] = df["asistencias"] / df["clases_totales"]

df["engagement_score"] = (
    df["total_accesos_lms"] * 0.4 +
    df["participaciones_foro"] * 0.3 +
    df["tiempo_total_conexion_min"] * 0.3
)
```

5. SELECCION DE VARIABLES

In [7]:
```
features = [
    "edad",
    "nota_primer_parcial",
    "nota_segundo_parcial",
    "promedio_previo",
    "total_accesos_lms",
    "tiempo_total_conexion_min",
    "participaciones_foro",
    "ratio_tareas",
    "ratio_asistencia",
    "engagement_score",
    "retroalimentaciones_docente"
]

X = df[features]
y = df["riesgo_academico"]
```

6. PREPROCESAMIENTO

In [9]:
```
imputer = KNNImputer(n_neighbors=5)
scaler = StandardScaler()
```

```
X = imputer.fit_transform(X)
X = scaler.fit_transform(X)
```

7. DIVISION TRAIN / TEST

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)
```

8. ENTRENAMIENTO DE MODELOS

In [11]:
```
models = {
    "Regresion_Logistica": LogisticRegression(max_iter=1000),
    "Random_Forest": RandomForestClassifier(n_estimators=200, random_state=42),
    "Red_Neuronal": MLPClassifier(hidden_layer_sizes=(50,50), max_iter=1000)
}

results = []

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]

    auc = roc_auc_score(y_test, y_prob)

    print(f"\nModelo: {name}")
    print(classification_report(y_test, y_pred))

    results.append({
        "Modelo": name,
        "ROC_AUC": auc
    })
```

```
Modelo: Regresion_Logistica
              precision    recall  f1-score   support

           0       0.96      0.95      0.96       105
           1       0.64      0.69      0.67        13

    accuracy                           0.92       118
   macro avg       0.80      0.82      0.81       118
weighted avg       0.93      0.92      0.92       118


Modelo: Random_Forest
              precision    recall  f1-score   support

           0       0.95      1.00      0.98       105
           1       1.00      0.62      0.76        13

    accuracy                           0.96       118
   macro avg       0.98      0.81      0.87       118
weighted avg       0.96      0.96      0.95       118


Modelo: Red_Neuronal
              precision    recall  f1-score   support

           0       0.97      0.92      0.95       105
           1       0.56      0.77      0.65        13

    accuracy                           0.91       118
   macro avg       0.76      0.85      0.80       118
weighted avg       0.92      0.91      0.91       118
```

9. RESULTADOS COMPARATIVOS

In [12]:
```
results_df = pd.DataFrame(results)
results_df.to_csv("resultados_modelos.csv", index=False)

print("\nResultados comparativos guardados")
```

```
Resultados comparativos guardados
```

10. DATASET FINAL PARA POWER BI

In [13]:
```
df_powerbi = df.copy()
df_powerbi["probabilidad_riesgo"] = models["Random_Forest"].predict_proba(X)[:,

df_powerbi.to_csv("dataset_powerbi.csv", index=False)

print("Dataset para Power BI generado correctamente")
```

```
Dataset para Power BI generado correctamente
```

In [ ]:
```

```