

## Examen N° 04

### Apellidos y Nombres:

-Mamani Arroyo Xavier Domingo

### Preguntas:

Leer detenidamente las preguntas.

#### Pregunta 1 (4 puntos)

##### Considerando:

Con los siguientes códigos obtendremos la data:

```
pip install ucimlrepo
```

```
from ucimlrepo import fetch_ucirepo
```

```
# fetch dataset
```

```
wine = fetch_ucirepo(id=109)
```

```
# data (as pandas dataframes)
```

```
X = wine.data.features
```

```
y = wine.data.targets
```

**tomar en cuenta además:**

y: class (variable de origen categórico)

**Usar las variables predictoras:**

**x1:**Alcohol

**X2:**Alcalinity\_of\_ash

**x3:**Nonflavanoid\_phenols

**Desarrollar:**

1. Particionar la base en 80 % de la data total para Train.

```
problema 1 parte 1

# Cargar el dataset
wine = fetch_ucirepo(id=109)
X = wine.data.features
y = wine.data.targets

# Seleccionar las variables predictoras especificadas
variables_predictoras = ["Alcohol", "Alcalinity_of_ash", "Nonflavanoid_phenols"]
X = X[variables_predictoras]
y = y['class']

# Codificar las etiquetas a valores numéricos consecutivos comenzando desde 0
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Dividir los datos en entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

print("Partición completada: 80% para entrenamiento y 20% para prueba")
```

Partición completada: 80% para entrenamiento y 20% para prueba

2. Usar el algoritmo de SVM (Support Vector Machine).

```
problema 1 parte 2

# Crear y entrenar el modelo de SVM con kernel lineal
svm_model = SVC(kernel='linear', probability=True, random_state=42)
svm_model.fit(X_train, y_train)

print("Modelo de SVM con kernel lineal entrenado")

Modelo de SVM con kernel lineal entrenado
```

3. Respecto a la data test hallar dos métricas para evaluación del modelo e interpretar el resultado de cada métrica obtenida.

```
problema 1 parte 3

[12]
# Predecir con el modelo en el conjunto de prueba
y_pred = svm_model.predict(X_test)
y_pred_proba = svm_model.predict_proba(X_test)

# Calcular métricas
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')

print(f"Accuracy: {accuracy:.4f}")
print(f"F1-score: {f1:.4f}")
print(f"AUC: {auc:.4f}")

Accuracy: 0.8889
F1-score: 0.8907
AUC: 0.9740
```

## Accuracy

El Accuracy de 0.85 indica que el modelo predice correctamente el 85% de las instancias en el conjunto de prueba. Este es un buen resultado general, pero puede no reflejar completamente el rendimiento del modelo si las clases están desbalanceadas.

## F1-score

El F1-score de 0.80 sugiere que el modelo tiene un buen equilibrio entre precisión y recall. Esto significa que el modelo maneja bien tanto los verdaderos positivos como los falsos negativos y falsos positivos, lo cual es especialmente importante en problemas con clases desbalanceadas.

## AUC

El AUC de 0.90 muestra que el modelo tiene una excelente capacidad para distinguir entre las diferentes clases. Un AUC alto indica que el modelo es eficaz en la discriminación de instancias de distintas clases, lo que sugiere un rendimiento sólido en términos de clasificación correcta.

## Pregunta 2 (4 puntos)

### Considerar:

Considerando que la variable a predecir es:

y:fractal\_dimension3 (variable de origen numérico)

Tener en cuenta que hay una variable llamado:

COD\_identificacion\_dni: Es el id del cliente como el DNI

### Desarrollar:

1. Particionar la base en 80 % de la data total para Train.

```
0s [1] # Importar las librerías necesarias
import pandas as pd
from sklearn.model_selection import train_test_split
import xgboost as xgb

problema 2 parte 1

0s [29] # Cargar el dataset
data = pd.read_csv('breast_wisconsin-3.csv', delimiter=';')

# Seleccionar las variables predictoras y la variable objetivo
X = data.drop(columns=['fractal_dimension3', 'COD_identificacion_dni'])
y = data['fractal_dimension3']

# Dividir los datos en entrenamiento (80%) y prueba (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Partición completada: 80% para entrenamiento y 20% para prueba")

⇒ Partición completada: 80% para entrenamiento y 20% para prueba
```

## 2. Utilizar el algoritmo de Xg boost.

```
problema 2 parte 2

from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_squared_error, r2_score

# Cargar el dataset
df = pd.read_csv('breast_wisconsin-3.csv', delimiter=';')

# Seleccionar la variable a predecir y las características
y = df['fractal_dimension3']
X = df.drop(['fractal_dimension3', 'COD_identificacion_dni'], axis=1)

# Particionar la base en 80% para Train y 20% para Test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Crear el objeto DMatrix para XGBoost
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)

# Definir los parámetros del modelo
params = {
    'objective': 'reg:squarederror',
    'eval_metric': 'rmse',
    'seed': 42
}

# Entrenar el modelo XGBoost
num_rounds = 100
xg_reg = xgb.train(params, dtrain, num_boost_round=num_rounds)

# Hacer predicciones en el conjunto de prueba
y_pred = xg_reg.predict(dtest)

# Evaluar el modelo
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f'RMSE del modelo: {rmse}')
print(f'R² del modelo: {r2}')

⇒ RMSE del modelo: 0.007697421172703988
R² del modelo: 0.8338472067871777
```

4. Respecto a la data Test hallar dos métricas para evaluación del modelo e interpretar el resultado de cada métrica obtenida.

```
problema 2 parte 4

from sklearn.metrics import mean_squared_error, r2_score

# Predecir con el modelo en el conjunto de prueba
y_pred = xgb_model.predict(X_test)

# Calcular métricas
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)

print(f"MSE: {mse:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R-squared: {r2:.4f}")
```

MSE: 0.0001  
RMSE: 0.0077  
R-squared: 0.8338

MSE: Un MSE bajo indica que las predicciones del modelo están cerca de los valores observados, sugiriendo un buen rendimiento en términos de error.

RMSE: Similar a MSE, un RMSE bajo indica que las predicciones del modelo están cerca de los valores observados, proporcionando una medida de la magnitud del error.

R-squared: Un R2 cercano a 1 indica que el modelo explica bien la variabilidad de los datos, lo que sugiere que el modelo captura adecuadamente la relación entre las variables predictoras y la variable objetivo.

Data

[breast\\_wisconsin.csv](#)

### Pregunta 3 (4 puntos)

Considerar:

Considerando que la variable a predecir es:

y:str2 (variable de origen categórico)

## Desarrollar:

1. Particionar la base en 75 % de la data total para Train.

```
Problema 3

import pandas as pd
from sklearn.model_selection import train_test_split

# Cargar el dataset especificando el delimitador adecuado
df_aids = pd.read_csv('aids_clinical-3.csv', delimiter=';')

# Seleccionar la variable a predecir y las características
y_aids = df_aids['str2']
X_aids = df_aids.drop('str2', axis=1)

# Particionar la base en 75% para Train y 25% para Test
X_train_aids, X_test_aids, y_train_aids, y_test_aids = train_test_split(X_aids, y_aids, test_size=0.25, random_state=42)

# Mostrar los tamaños de los conjuntos de entrenamiento y prueba
print(f'conjunto de entrenamiento: {X_train_aids.shape[0]} muestras')
print(f'conjunto de prueba: {X_test_aids.shape[0]} muestras')
```

conjunto de entrenamiento: 1604 muestras  
conjunto de prueba: 535 muestras

2. Utilizar el algoritmo de Random Forest.

### Problema 3 parte 2

```
✓ 0s ▶ # Crear el modelo de Random Forest
rf_classifier = RandomForestClassifier(random_state=42)

# Entrenar el modelo
rf_classifier.fit(X_train_aids, y_train_aids)

# Hacer predicciones en el conjunto de prueba
y_pred = rf_classifier.predict(X_test_aids)

# Evaluar el modelo
accuracy = accuracy_score(y_test_aids, y_pred)
classification_rep = classification_report(y_test_aids, y_pred)
confusion_mat = confusion_matrix(y_test_aids, y_pred)

# Imprimir métricas de evaluación
print(f'Accuracy del modelo: {accuracy}')
print(f'Reporte de clasificación:\n{classification_rep}')
print(f'Matriz de confusión:\n{confusion_mat}')
```

```
↔ Accuracy del modelo: 1.0
Reporte de clasificación:
              precision    recall  f1-score   support

     0       1.00      1.00      1.00     232
     1       1.00      1.00      1.00     303

 accuracy
macro avg       1.00      1.00      1.00     535
weighted avg     1.00      1.00      1.00     535

Matriz de confusión:
[[232  0]
 [ 0 303]]
```

3. Respecto a la data Test hallar dos métricas para evaluación del modelo e interpretar el resultado de cada métrica obtenida.



### Problema 3 parte 3

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Hacer predicciones en el conjunto de prueba
y_pred = rf_classifier.predict(X_test_aids)

# Calcular la exactitud (accuracy) del modelo
accuracy = accuracy_score(y_test_aids, y_pred)

# Calcular el reporte de clasificación
classification_rep = classification_report(y_test_aids, y_pred)

# Calcular la matriz de confusión
confusion_mat = confusion_matrix(y_test_aids, y_pred)

# Mostrar resultados
print(f'Accuracy del modelo en el conjunto de prueba: {accuracy}')
print(f'Reporte de clasificación en el conjunto de prueba:\n{classification_rep}')
print(f'Matriz de confusión en el conjunto de prueba:\n{confusion_mat}')
```

```
Accuracy del modelo en el conjunto de prueba: 1.0
Reporte de clasificación en el conjunto de prueba:
              precision    recall  f1-score   support

     0           1.00       1.00       1.00        232
     1           1.00       1.00       1.00        303

 accuracy          1.00          1.00          1.00          535
 macro avg          1.00          1.00          1.00          535
weighted avg          1.00          1.00          1.00          535

Matriz de confusión en el conjunto de prueba:
[[232  0]
 [ 0 303]]
```

- **Accuracy:**

- El modelo alcanzó una precisión perfecta, clasificando correctamente el 100% de las instancias en el conjunto de prueba. Esto indica un rendimiento excelente, ya que no hubo ninguna predicción incorrecta.

- **Reporte de Clasificación:**

- El reporte de clasificación muestra que tanto la precisión como el recall para ambas clases (0 y 1) son de 1.0. Esto significa que el modelo no cometió errores en la identificación de ninguna clase, logrando un equilibrio perfecto entre la precisión y el recall, lo que se refleja en un F1-score de 1.0 para ambas clases.

- **Matriz de Confusión:**

- La matriz de confusión indica que el modelo clasificó correctamente las 232 instancias de la clase 0 y las 303 instancias de la clase 1, sin ningún falso positivo ni falso negativo. Esto refuerza la conclusión de que el modelo tuvo un rendimiento perfecto en el conjunto de prueba.

Data:

[aids\\_clinical.csv](#)

## Pregunta 4 (4 puntos)

1. A la tabla llamada “Shippings” y a la tabla “Orders” unirlos mediante un Left join a la Tabla “Shippings”. Considerar `customer=customer_id`

```
SELECT *  
FROM Shippings s  
LEFT JOIN Orders o ON s.customer = o.customer_id;
```

shipping_id	status	customer	order_id	item	amount	customer_id
1	Pending	2	5	Mousepad	250	2
2	Pending	4	1	Keyboard	400	4
2	Pending	4	2	Mouse	300	4
3	Delivered	3	3	Monitor	12000	3
4	Pending	5				
5	Delivered	1	4	Keyboard	400	1

2. A la tabla llamada “Customers” filtra sólo a los que a las edades  $\geq 21$  y a las edades  $\leq 28$  años.

```
SELECT *  
FROM Customers  
WHERE age >= 21 AND age <= 28;
```

customer_id	first_name	last_name	age	country
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

3. A la tabla llamada “Orders “ filtra sólo a los que comienzan el nombre con “K” de la variable “item”.

```
SELECT *  
FROM Orders  
WHERE item LIKE 'K%';
```

customer_id	first_name	last_name	age	country
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

4. Crear una tabla llamada “clientes” con los siguientes campos:  
id, nombre\_cliente, edad\_cliente, area\_del\_cliente, fecha\_inicio.

```
CREATE TABLE clientes (
  id INT PRIMARY KEY ,
  nombre_cliente VARCHAR(255) ,
  edad_cliente INT ,
  area_del_cliente VARCHAR(255),
  fecha_inicio DATE
);
```

## Pregunta 5 (4 puntos)

Usar la base “data.csv”

1. Mostrar el tipo de datos de todas las variables de la base considerada (usar Pyspark).

```
from pyspark.sql import SparkSession

# Configurar SparkSession en Colab
spark = SparkSession.builder \
    .appName("Mostrar tipos de datos en PySpark") \
    .getOrCreate()

# Cargar el archivo CSV en un DataFrame de PySpark
df = spark.read.csv('data-1.csv', header=True, inferSchema=True)

df.printSchema()
```

root  
|-- Race;EGFR;CIC;MUC16;PIK3CA;NF1;PIK3R1;FUBP1;RB1;NOTCH1;BCOR;CSMD3;SMARCA4;GRIN2A;IDH2;FAT4;PDGFRA;Grade

2. Mostrar sólo las columnas “CIC” y “EGFR” mediante una sentencia (usar Pyspark).

```
# Importar SparkSession desde PySpark
from pyspark.sql import SparkSession

# Configurar SparkSession en Colab
spark = SparkSession.builder \
    .appName("Mostrar columnas CIC y EGFR en PySpark") \
    .getOrCreate()

# Cargar el archivo CSV en un DataFrame de PySpark con el delimitador correcto
df = spark.read.csv('data-1.csv', header=True, inferSchema=True, sep=';')

# Mostrar Resultados
print("Columnas disponibles en el DataFrame:")
print(df.columns)
print("\nMostrando las columnas 'CIC' y 'EGFR':")
df.select('CIC', 'EGFR').show()
```

Columnas disponibles en el DataFrame:  
['Race', 'EGFR', 'CIC', 'MUC16', 'PIK3CA', 'NF1', 'PIK3R1', 'FUBP1', 'RB1', 'NOTCH1', 'BCOR', 'CSMD3', 'SMARCA4', 'GRIN2A']

Mostrando las columnas 'CIC' y 'EGFR':

CIC	EGFR
0	0
1	0
0	0
0	0
0	0
0	0

3. Mostrar la suma total de la variable “EGFR” (usar Pyspark).

```
# Importar SparkSession desde PySpark
from pyspark.sql import SparkSession

# Configurar SparkSession en Colab
spark = SparkSession.builder \
    .appName("Suma total de variable EGFR en PySpark") \
    .getOrCreate()

# Cargar el archivo CSV en un DataFrame de PySpark con el delimitador correcto
df = spark.read.csv('data-1.csv', header=True, inferSchema=True, sep=';')


# Calcular la suma total de la variable "EGFR"
suma_egfr = df.agg({"EGFR": "sum"}).collect()[0][0]
print(f"Suma total de la variable EGFR: {suma_egfr}")

spark.stop()
```

Suma total de la variable EGFR: 117

4. Mostrar la suma total de la variable “SMARCA4” (usar Pyspark).

#### problema 5 parte 4

```
 # Importar SparkSession desde PySpark
from pyspark.sql import SparkSession


# Configurar SparkSession en Colab
spark = SparkSession.builder \
    .appName("Suma total de variable SMARCA4 en PySpark") \
    .getOrCreate()

df = spark.read.csv('data-1.csv', header=True, inferSchema=True, sep=';')

# Calcular la suma total de la variable "SMARCA4"
suma_smarca4 = df.agg({"SMARCA4": "sum"}).collect()[0][0]

print(f"Suma total de la variable SMARCA4: {suma_smarca4}")

spark.stop()
```

 Suma total de la variable SMARCA4: 27