Universidad Interamericana de Puerto Rico

Recinto de Bayamón

Escuela de Ingeniería
Departamento de Ingeniería Eléctrica y de Computadoras

COEN 4450 – Ciencia de Datos

Proyecto-2

Nombre de estudiante	:	Mireya Vázquez Robles
Nombre de proyecto	:	Clasificación
Porcentaje de tareas completadas (0 - 100 %)	:	100%
Fecha	:	5/12/2022

1) Introducción

Breve descripción del Proyecto y mostrar el resumen en la siguiente tabla:

Este proyecto consiste en el diseño e implementación de un clasificador de decisión Bayes. El programa tiene dos archivos. El primero tiene un menú que le permite al usuario elegir entre varias opciones. El segundo, es el que tiene todas las funciones que serán llamadas. Se crearon funciones propias para calcular la validación cruzada, media, varianza, probabilidad de Bayes, densidad de probabilidad, matriz de confusión, exactitud, precisión, sensibilidad y el f-score. Se utilizó la validación cruzada "10 fold" y el método PCA (Principle Component Analysis) para poder realizar una comparación y analizar los resultados. El propósito de este proyecto es aplicar conceptos aprendidos en clase sobre visualización, manejo de datos y clasificación.

Resumen	**Porcentaje (0-100%)**
Informe tiene todos los puntos completos – Indique porciento	100%
¿Programa corre al 100% con todas las tareas? – indique porciento	100%

2) Análisis del conjunto de datos

a) Llene la siguiente tabla con la información solicitada:

Tabla 1: conjunto de datos

Dataset	**Dimensiones*** [*filas x columnas*]	**Numero de Clases**
data_1	150 x 4	3
data_2	10,000 x 4	2
data_3	178 x 13	3
data_4	100 x 3	2

*sin contar la columna que indica a que clase pertenece (etiqueta)

b) Realice una clasificación de dificultad (alta, media, baja) de los datos que usted va a clasificar con Bayes según los datos observados en la tabla-1. Presenta una justificación de esa clasificación (¿Cuál fue la razón de colocar ese nivel de dificultad?)

He decidido clasificar el dataset data_1 como de dificultad baja, el datataset data_2 como de dificultad alta, el dataset data_3 como de dificultad media y el dataset data_4 como de dificultad baja. La razón por la cual le he colocado al dataset data_1 un nivel de dificultad bajo es porque tiene solo 4 columnas de características continuas y 150 datos a clasificar. Es muy fácil y rápido de clasificar. El dataset data_2 tiene un número muy alto de datos a clasificar, por lo que le he colocado un nivel de dificultad alto. Mientras más datos se deban clasificar, más tiempo tarda el programa en correr y más difícil es comprobar que los resultados son los correctos ya que realizar los cómputos manualmente tomaría demasiado tiempo. La razón por la cual le he colocado al dataset data_3 un nivel de dificultad media es porque tiene 14 columnas de características continuas. Esto me forzó a crear un programa que funcionara independientemente de la cantidad de columnas. Adicionalmente, tiene un número de datos que no es divisible entre 10, lo que significa que durante la validación cruzada uno de los grupos tendrá menos datos. Le coloqué un nivel de dificultad bajo al último dataset (data_4) porque a pesar de tener una combinación de características continuas y discretas es el que menor número de clases, filas y columnas tiene por lo que es el más fácil y rápido de clasificar y comprobar.

c) Calcule el porciento de cada clase y concluya si las clases están balanceadas o desbalanceadas.'

Las clases en el conjunto de datos data_1 están balanceadas con 33% en cada clase.

Tabla 2: porciento de cada clase del Dataset data_1

Dataset	Clase 1	Clase 2	Clase 3
data_1	33%	33%	33%

Las clases en el conjunto de datos data_2 están perfectamente balanceadas con 50% en cada clase.

Tabla 3: porciento de cada clase del Dataset data_2

Dataset	Clase 1	Clase 2
data_2	50%	50%

Las clases en el conjunto de datos data_3 están desbalanceadas. La mayoría de los datos pertenecen a la clase 2, mientras que la minoría pertenece a la clase 3. Aunque la diferencia no es muy drástica, esto afectará la medida de exactitud.

Tabla 4: porciento de cada clase del Dataset data_3

Dataset	Clase 1	Clase 2	Clase 3
data_3	33%	40%	27%

Las clases en el conjunto de datos data_4 están sumamente desbalanceadas. El 70% de los datos pertenecen a la clase 2. Esto significa que el clasificador estará predispuesto a clasificar los datos como perteneciente a la clase 2 y los resultados de exactitud estarán sesgados.

Tabla 5: porciento de cada clase del Dataset data_4

Dataset	Clase 1	Clase 2
data_4	30%	70%

3) Resultados

a) Corra el clasificador Bayes y muestre sus resultados en la Tabla-6:

En la tabla-6 se presentan los valores promedios de la exactitud, precisión, sensibilidad y f-score. En base a estos resultados, podemos ver que el clasificador tiene una alta exactitud, precisión y sensibilidad.

Tabla 6: Resultados con todas sus características

Dataset	Exactitud	Precisión	Sensibilidad	F
data_1	0.9689	0.9472	0.9524	0.9460
data_2	0.9356	0.9316	0.9402	0.9359
data_3	0.9811	0.9708	0.9721	0.9683
data_4	1.0	1.0	1.0	1.0

b) Después de llenar los datos en la Tabla-6, realiza un análisis individual para cada conjunto de datos según los valores obtenidos, es decir forme una opinión fundamentada y escriba en sus palabras esa opinión.

El conjunto de datos_1 tiene 150 datos con 4 características distribuidos balanceadamente en 3 clases. En base a los resultados obtenidos utilizando la validación cruzada, podemos observar que el clasificador tiene alta exactitud, precisión y sensibilidad en la mayoría de las iteraciones, lo cual significa que maneja bien las clases. La octava corrida es la que menor precisión y sensibilidad tuvo. Esto ocurrió debido a que falló en predecir correctamente los datos pertenecientes a la clase 2 y clase 3. En esta corrida el conjunto de prueba contenía 15 datos, 7 de ellos pertenecieron a la clase 1, 2 a la clase 2 y 6 a la clase 3. Pudo predecir correctamente todos los datos pertenecientes a la clase 1, pero clasificó uno de los datos de la clase 2 como perteneciente a la clase 3 y uno de los datos pertenecientes a la clase 3 como perteneciente a la clase 2. Esto posiblemente ocurrió porque estos datos tienen características similares. Por ejemplo, si vemos la columna 4 en el conjunto de datos, aquellos datos pertenecientes a la clase 1 tienen valores entre 0.1 y 0.5, los datos pertenecientes a la clase 2 tienen valores entre 1 y 1.8, y los datos pertenecientes a la clase 3 tienen valores entre 1.5 y 2.5. Los datos de las clases 2 y 3 comparten el mismo rasgo o valor en un rango de 1.5 a 1.8. Lo mismo ocurre con las otras columnas, causando así un poco de confusión.

Tabla 7: Resultados con todas sus características por cada pliegue de data_1

Dataset	Pliegue	Exactitud	Precisión	Sensibilidad	F
data_1	1	1.0	1.0	1.0	1.0
	2	0.9556	0.8889	0.9524	0.9077
	3	0.9556	0.9445	0.9524	0.9441
	4	1.0	1.0	1.0	1.0
	5	1.0	1.0	1.0	1.0
	6	1.0	1.0	1.0	1.0
	7	0.9556	0.9445	0.9524	0.9441
	8	0.9111	0.7778	0.7778	0.7780
	9	1.0	1.0	1.0	1.0
	10	0.9111	0.9167	0.8889	0.8857

El conjunto de datos_2 tiene 10,000 datos con 4 características distribuidos balanceadamente en 2 clases. En base a los resultados obtenidos utilizando la validación cruzada, podemos observar que el clasificador tiene alta precisión y sensibilidad en todas las iteraciones, lo cual significa que maneja bien las clases. El conjunto de prueba tenía 1000 datos en cada pliegue y falló en predecir correctamente la clasificación de entre 47 y 80 de ellos. Al igual que en el conjunto de datos anterior, estos datos tenían rasgos muy similares. Luego de reducir las

características y graficarlos (ver figura 2) podemos observar que muchos de los puntos quedan uno encima de otro.

Tabla 8: Resultados con todas sus características por cada pliegue de data_2

Dataset	Pliegue	Exactitud	Precisión	Sensibilidad	F
data_2	1	0.9530	0.9379	0.9638	0.9507
	2	0.9290	0.9261	0.9352	0.9306
	3	0.9440	0.9361	0.9577	0.9468
	4	0.9320	0.9302	0.9302	0.9302
	5	0.9450	0.9533	0.9363	0.9448
	6	0.9390	0.9373	0.9428	0.9400
	7	0.9230	0.9134	0.9271	0.9204
	8	0.9400	0.9492	0.9346	0.9419
	9	0.9200	0.9142	0.9253	0.9197
	10	0.9310	0.9184	0.9490	0.9335

El conjunto de datos_3 tiene 178 datos con 13 características distribuidos en 3 clases. En base a los resultados obtenidos utilizando la validación cruzada, podemos observar que el clasificador tiene alta exactitud, pero dado a que las clases están desbalanceadas no deberíamos confiar mucho en estos resultados. Sin embargo, podemos confiar en que maneja bien las clases debido a que tiene alta precisión y sensibilidad. En las iteraciones con resultados más bajos el clasificador confundió un dato perteneciente a la clase 1 como perteneciente a la clase 2 en los pliegues 5, 6 y 9, y un dato perteneciente a la clase 2 como pertenecientes a la clase 3 en los pliegues 8 y 10.

Tabla 9: Resultados con todas sus características por cada pliegue de data_3

Dataset	Pliegue	Exactitud	Precisión	Sensibilidad	F
data_3	1	1.0	1.0	1.0	1.0
	2	1.0	1.0	1.0	1.0
	3	1.0	1.0	1.0	1.0
	4	1.0	1.0	1.0	1.0
	5	0.9630	0.9444	0.9583	0.9475
	6	0.9630	0.9333	0.9630	0.9434

	7	1.0	1.0	1.0	1.0
	8	0.9630	0.9524	0.9583	0.9521
	9	0.9608	0.9333	0.9524	0.9373
	10	0.9608	0.9444	0.8889	0.9030

El conjunto de datos_4 tiene 100 datos con 3 características distribuidos en 2 clases. En base a los resultados obtenidos utilizando la validación cruzada, podemos observar que el clasificador tiene alta exactitud, precisión y sensibilidad en todas las iteraciones, lo cual significa que maneja bien las clases. Aunque el resultado de exactitud no es buena medida por tener clases desbalanceadas, en este caso el resultado si es correcto, al igual que los resultados de precisión y sensibilidad, ya que lo podemos comprobar utilizando la matriz de confusión. El clasificador pudo predecir a la perfección la clase a la que pertenece cada dato en todas las corridas a pesar de tener pocos datos y clases desbalanceadas. El modelo de Bayes es particularmente útil para conjuntos de datos pequeños como este.

Tabla 10: Resultados con todas sus características por cada pliegue de data_4

Dataset	Pliegue	Exactitud	Precisión	Sensibilidad	F
data_4	1	1.0	1.0	1.0	1.0
	2	1.0	1.0	1.0	1.0
	3	1.0	1.0	1.0	1.0
	4	1.0	1.0	1.0	1.0
	5	1.0	1.0	1.0	1.0
	6	1.0	1.0	1.0	1.0
	7	1.0	1.0	1.0	1.0
	8	1.0	1.0	1.0	1.0
	9	1.0	1.0	1.0	1.0
	10	1.0	1.0	1.0	1.0

4) Reducción de características (“features”)

Utilice el método PCA (Principle Component Analysis) para reducir las características (features) de los datos. Corra nuevamente Bayes y muestre los resultados en la tabla-11.

Tabla 11: Resultados con características reducidas

Dataset	Exactitud	Precisión	Sensibilidad	F
data_1	0.9289	0.8826	0.9060	0.8778
data_2	0.9303	0.9322	0.9279	0.9300
data_3	0.8273	0.7329	0.7275	0.8420
data_4	1.0	1.0	1.0	1.0

a) Realice un análisis individual para cada conjunto de datos comparando los resultados de la tabla-6 con la tabla-11. ¿Existen cambios? Si es así, ¿Cuál cree sea el motivo de esos cambios?, explique.

Al comparar los resultados de la tabla 6 con la tabla 11 del conjunto data_1, podemos observar que tanto la exactitud, precisión, sensibilidad y f-score disminuyeron. Esto podría ser debido a que como ya existía un poco de confusión al clasificar los datos, al reducir el número de características crea más confusión. Sin embargo, como solo se redujo de 4 columnas a 2 la diferencia no es muy drástica.

En el caso del conjunto de datos_2, la diferencia entre los resultados de la tabla 6 y la tabla 11 no es significativa. Al reducir las características disminuyó un poco la exactitud, precisión, sensibilidad y f-score, pero como solo se redujo de 4 a 2 columnas, el impacto no es mucho. Al igual que en el caso anterior, es posible que esto ocurrió debido a que ya existía un poco de confusión al clasificar los datos. Al reducir el número de características, se crea un poco más de confusión.

Donde hubo más cambios fue en el caso del conjunto de datos data_3, donde los resultados de exactitud, precisión, sensibilidad y f-score pasaron de ser los segundos más altos a los más bajos. El clasificador Bayes funciona muy bien con conjuntos de datos que tienen más de 3 características debido a que asume que cada característica es independiente de las demás. Por lo tanto, reducir las características es innecesario, es posible que al reducirlas se esté perdiendo información importante para la clasificación. El método PCA sería más útil al ser aplicado a clasificadores que funcionan mejor con menos dimensiones y en situaciones en que las características si están relacionadas unas a otras.

En cuanto al dataset data_4, aunque fue posible reemplazar las características discretas del conjunto de datos data_4 por un valor numérico convirtiéndolo así de tipo “string” a tipo “float”, el método PCA está diseñado para trabajar con características continuas, no discretas, por lo que no es muy confiable. Existen otros métodos para reducir las características de los datos que serían más apropiados en casos como este en donde tenemos una mezcla de características

continuas y discretas. Aun así, los resultados fueron los mismos en ambas instancias. A pesar de que el método PCA no es el ideal para este conjunto de datos, el clasificador pudo predecir a que clase pertenecía cada dato a la perfección.

b) Grafique cada conjunto de datos en dos dimensiones (es decir solo 2 columnas). Coloque un color a cada clase existente.

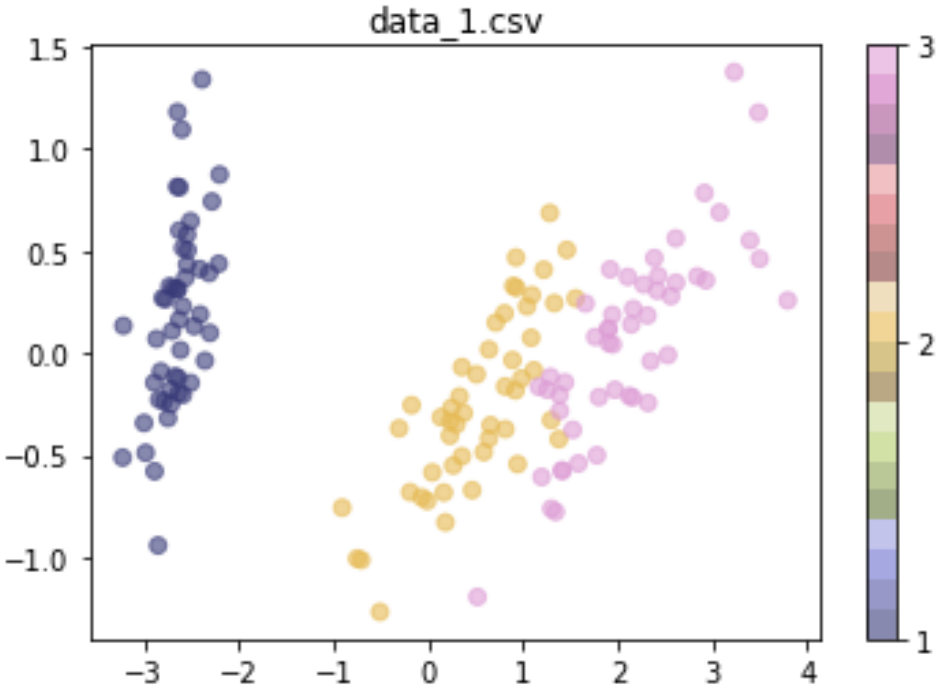

Fig. 1: Gráfica del conjunto de datos data_1

En la figura 1 podemos ver que los datos de la clase 2 y clase 3 están muy cercanos uno a otro. Esto significa que tienen características levemente similares.

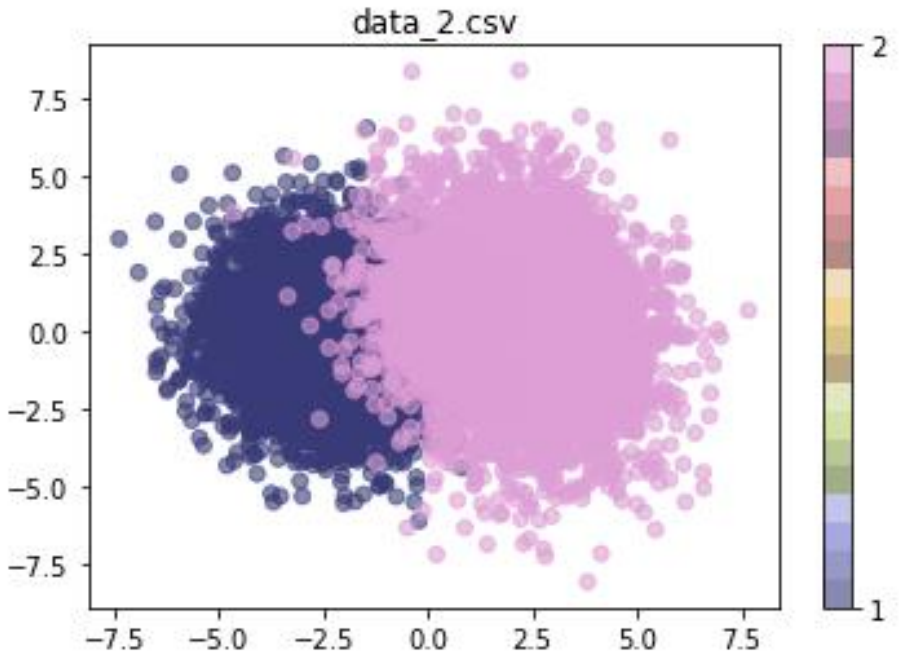

Fig. 2: Gráfica del conjunto de datos data_2

En la figura 2 podemos observar como muchos de los datos comparten las mismas características a pesar de pertenecer a clases distintas. No se puede apreciar mucho debido a la alta densidad de los grupos, pero los grupos están superpuestos en el centro. Es decir, los datos de ambas clases están ubicados uno encima de otro.

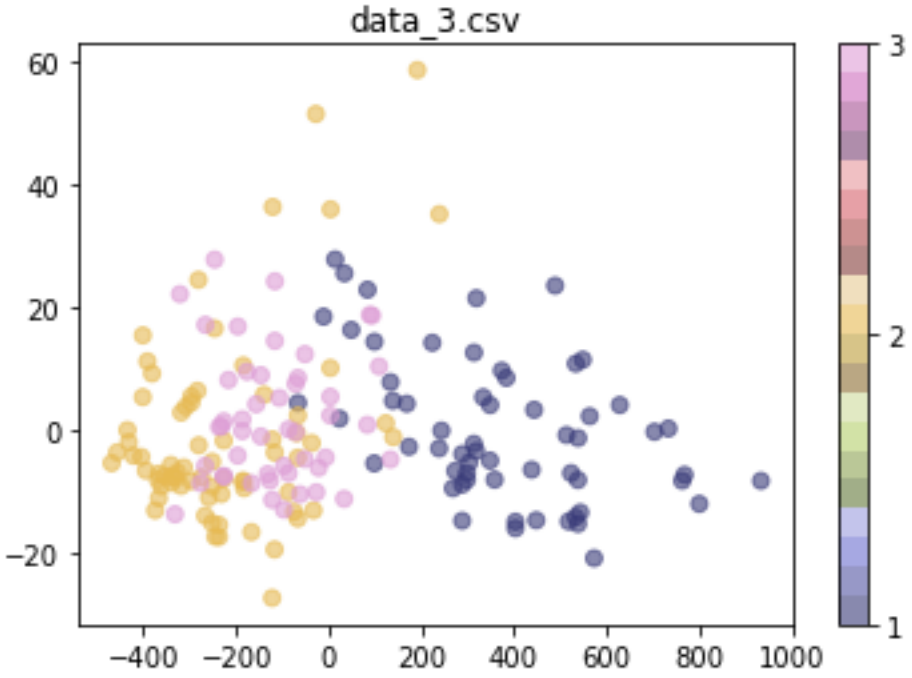

Fig. 3: Gráfica del conjunto de datos data_3

En la figura 3 podemos observar como los datos de las clases 2 y 3 se entrelazan. Esto indica que tienen características muy similares.

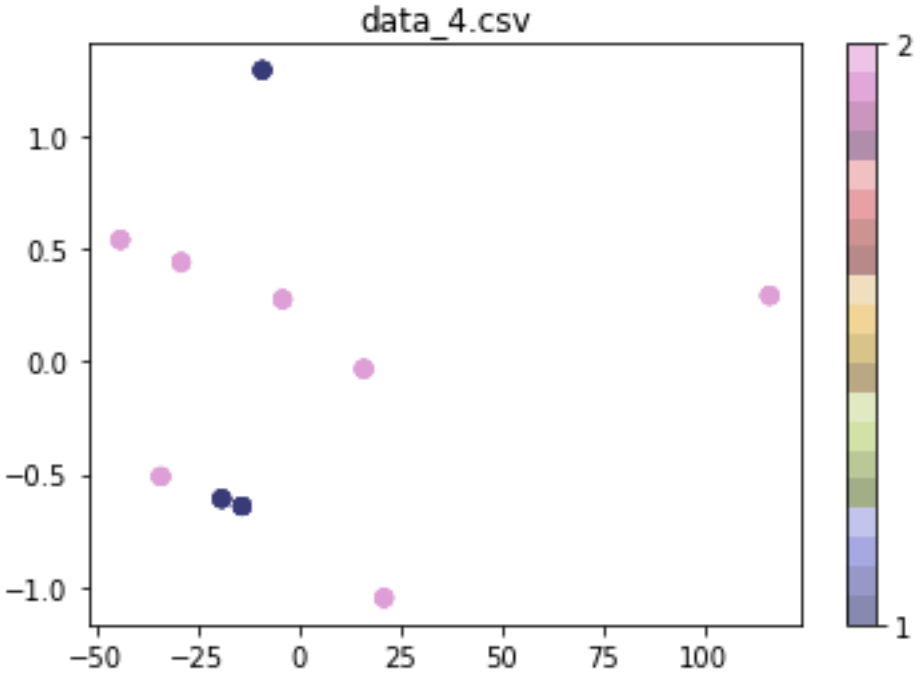

Fig. 4: Gráfica del conjunto de datos data_4

En la figura 4 podemos ver como los datos de las distintas clases están muy separados. Esto indica que sus características son muy muy distintas. En esta gráfica hay 100 datos, pero solo aparentan haber 10 ya que los datos de una misma clase están perfectamente uno encima de otro. Esto nos indica que muchos de los datos de una misma clase son exactamente iguales.

5) Conclusión

a) Describa todos los problemas principales que encontró y como fue solucionado.

El problema principal que encontré fue cómo lograr que el programa identificara cuales columnas tenían valores discretos y cuales tenían valores continuos, ya que esto determina cual será la manera en que se calculará la probabilidad de que ese dato pertenezca a cada clase. La forma en que solucioné este problema fue utilizando un if statement que evalúa si el elemento es de tipo “string”, es decir una palabra y no un número. Esto funcionó debido a que en los conjuntos de datos el único conjunto con características discretas utiliza palabras. El otro problema que encontré fue cómo evaluar los resultados, ya que debía obtener los valores

necesarios para los cálculos de la matriz de confusión y el tamaño de la misma depende del número de clases. La forma en que solucioné este problema fue utilizando un if statement que evalúa si el número de clases es igual a dos, y un elif statement que evalúa si el número de clases es igual a tres. Dentro de cada if statement está el procedimiento para evaluar los resultados. Otro problema encontrado fue que para poder aplicar el PCA al conjunto de datos data_4 era necesario que los valores fueran tipo “float”. La forma en que solucione este problema fue reemplazando los valores tipo “string” por valores numéricos tipo “float”. Es decir, le asigne un número a cada categoría. Reemplace “yes” por 1, “no” por 2, “single” por 1, “married” por 2 y “divorced” por 3.

b) Cosas que probaste y que no funcionaron.

Probé evaluar los resultados utilizando una misma serie de instrucciones que funcionara independientemente del tamaño de la matriz, pero esto no fue posible debido a que para evaluar los resultados de un conjunto de datos con una matriz de confusión de 3x3 es necesario realizar los cálculos con los resultados de cada clase y luego obtener el promedio, mientras que para un conjunto de datos con una matriz de confusión de 2x2 es necesario realizar los cálculos una sola vez.

c) Analice su diseño en términos de cómo los resultados cumplieron con sus expectativas. ¿Obtuviste lo que esperabas? ¿ Hubo alguna sorpresa?

En términos de cómo los resultados cumplieron con mis expectativas, considero que es un buen diseño debido a que cumple con lo requerido. Sin embargo, considero que el diseño podría ser mejorado si se le añaden otros métodos de análisis que sean apropiados para analizar conjuntos de datos con datos mixtos y si se cambian algunas partes del código para que pueda evaluar conjuntos de datos con más de 3 clases. Algunos de los resultados fueron inesperados, pero luego de analizar los datos y los resultados todo hizo sentido. Por ejemplo, una sorpresa fue el que para poder aplicar la reducción PCA fuera necesario convertir los valores de tipo “string” a tipo “float” en el caso del conjunto de datos data_4. Luego de investigar porqué esto ocurría pude entender que la razón es que la reducción PCA está diseñado para trabajar con valores numéricos y no es la ideal para analizar conjuntos de datos con características mixtas ni discretas. Aunque fue posible reemplazar los valores y asignarle un número a cada categoría logrando así que funcionara la función de PCA, un método como el de FAMD (Factor Analysis of Mixed Data) sería más apropiado para este conjunto de datos ya que está diseñado para reducir conjuntos de datos con características tanto categóricas como numéricas.

d) Haz una lista de todas las cosas que aprendiste.

- Cómo aplicar la validación cruzada
- Cómo separar los datos en conjunto de prueba y conjunto de entrenamiento
- Cómo realizar una matriz de confusión
- Cómo crear un clasificador de Bayes
- Cómo clasificar datos con características discretas y continuas
- Cómo evaluar los resultados utilizando las distintas métricas
- Cómo aplicar la técnica de suavizado de Laplace

6) Apéndices

I. Matriz de confusión de cada conjunto de datos

data_1

pliegue 1			
	predicción		
	5	0	0
	0	6	0
actual	0	0	4

pliegue 2			
	predicción		
	6	0	0
	0	6	1
actual	0	0	2

pliegue 3			
	predicción		
	3	0	0
	0	5	0
actual	0	1	6

pliegue 4			
	predicción		
	5	0	0
	0	3	0
actual	0	0	7

pliegue 5			
	predicción		
	5	0	0
	0	3	0
actual	0	0	7

pliegue 6			
	predicción		
	6	0	0
	0	4	0
actual	0	0	5

pliegue 7			
	predicción		
	3	0	0
	0	6	1
actual	0	0	5

pliegue 8			
	predicción		
	7	0	0
	0	1	1
actual	0	1	5

pliegue 9			
	predicción		
	7	0	0
	0	7	0
actual	0	0	1

pliegue 10			
	predicción		
	3	0	0
	0	6	0
actual	0	2	4

data_2

pliegue 1		
	predicción	
	453	17
actual	30	500

pliegue 2		
	predicción	
	476	33
actual	38	453

pliegue 3		
	predicción	
	498	22
actual	34	446

pliegue 4		
	predicción	
	453	34
actual	34	479

pliegue 5		
	predicción	
	470	32
actual	23	475

pliegue 6		
	predicción	
	478	29
actual	32	461

pliegue 7		
	predicción	
	445	35
actual	42	478

pliegue 8		
	predicción	
	486	34
actual	26	454

pliegue 9		
	predicción	
	458	37
actual	43	462

pliegue 10		
	predicción	
	484	26
actual	43	447

data_3

pliegue 1			
	predicción		
	4	0	0
	0	7	0
actual	0	0	7

pliegue 2			
	predicción		
	5	0	0
	0	6	0
actual	0	0	7

pliegue 3			
	predicción		
	3	0	0
	0	6	0
actual	0	0	9

pliegue 4			
	predicción		
	5	0	0
	0	8	0
actual	0	0	5

pliegue 5			
	predicción		
	5	0	0
	1	7	0
actual	0	0	5

pliegue 6			
	predicción		
	4	0	0
	1	8	0
actual	0	0	5

pliegue 7			
	predicción		
	5	0	0
	0	9	0
actual	0	0	4

pliegue 8			
	predicción		
	4	0	0
	0	6	0
actual	0	1	7

pliegue 9			
	predicción		
	4	0	0
	1	6	0
actual	0	0	6

pliegue 10			
	predicción		
	9	0	0
	0	5	0
actual	0	1	2

data_4

pliegue 1		
	predicción	
	7	0
actual	0	3

pliegue 2		
	predicción	
	8	0
actual	0	2

pliegue 3		
	predicción	
	6	0
actual	0	4

pliegue 4		
	predicción	
	5	0
actual	0	5

pliegue 5		
	predicción	
	9	0
actual	0	1

pliegue 6		
	predicción	
	5	0
actual	0	5

pliegue 7		
	predicción	
	8	0
actual	0	2

pliegue 8		
	predicción	
	7	0
actual	0	3

pliegue 9		
	predicción	
	9	0
actual	0	1

pliegue 10		
	predicción	
	6	0
actual	0	4

II. Tabla con todas las funciones creadas en su programa

nombre	descripción	entrada	salida
print_menu()	imprime el menú		imprime el menú
print_menu_datos()	imprime el submenú de datos		imprime el submenú de datos
reducir()	reduce las características a dos columnas por medio del método PCA	df, datos	df
graficar()	traza un gráfico de dispersión	df, clases, datos	muestra la gráfica

separar_clases()	separa las clases en un diccionario	df,df_shuffled	label, clases_separadas, clases
laplace()	aplica la técnica de suavizado Laplace que ayuda solucionar el problema de la probabilidad cero	PXC, count_all_when, count_all, test	PXC
calcular_media()	calcula la media	train_column_seleccion	media
calcular_varianza()	calcula la varianza	media, train_column_seleccion	varianza
densidad_probabilidad()	calcula la densidad de probabilidad para características continuas y discretas	x, test, train, train_column, key, column, count_all, PXC_list, media_list, varianza_list	PXC
bayes()	calcula la probabilidad de Bayes	PXC_list, count_all, train	probabilidad
pertenencia()	determina a cuál clase pertenece en base a la probabilidad	probabilidad_list ,clases	probabilidad_max, prediccion
exactitud_precision_sensibilidad_fscore()	calcula la exactitud, precisión, sensibilidad y f-score	TP,FN,FP,TN	exactitud, precisión, sensibilidad, fscore
matriz_confusion()	calcula e imprime la matriz de confusión	clases, real_list,rediccion_list	matriz
indices()	obtiene los valores de la matriz de confusión para poder calcular la exactitud, precisión, sensibilidad y f-score y luego imprime los resultados	clases, matriz	exactitud, precision, sensibilidad, fscore, promedio_exactitud, promedio_precision, promedio_fscore
resultado_final()	calcula los promedios de exactitud, precisión, sensibilidad y fscore e imprime la matriz de confusión final	matriz_masterlist,exactitud_masterlist,precision_masterlist,sensibilidad_masterlist,fscore_masterlist	matriz_total,exactitud_total,precision_total, sensibilidad_total,fscore_total
validación_cruzada()	divide aleatoriamente el conjunto de entrenamiento en 10 grupos de aproximadamente el	df	

	mismo tamaño y realiza los cálculos para cada grupo		

III. Vectores de media y varianza de cada conjunto de datos

Dataset	Pliegue	Media	Varianza
data_1	1	[6.534782608695653, 2.947826086956521 6, 5.532608695652173, 2.0130434782608697]	[0.3805293005671078, 0.09379962192816638, 0.3008931947069944, 0.07678638941398863]
	2	[6.58125, 2.9791666666666665, 5.554166666666667, 2.033333333333333]	[0.4031901041666668, 0.10039930555555555, 0.30039930555555555, 0.07555555555555556]
	3	[6.597674418604651, 3.011627906976744, 5.5558139534883715, 2.030232558139535]	[0.40534342888047603, 0.09637641968631687, 0.32665224445646307, 0.07187669010275824]
	4	[6.5, 2.974418604651163, 5.469767441860464, 2.0209302325581397]	[0.3483720930232559, 0.09678745267712274, 0.2100162249864792, 0.0802595997836668]
	5	[6.64186046511628, 2.976744186046511, 5.6139534883720925, 2.027906976744186]	[0.4233639805300163, 0.1078312601406166, 0.2965494862087616, 0.07643050297458082]
	6	[6.613333333333332, 2.9644444444444447, 5.555555555555555, 2.026666666666667]	[0.4260444444444446, 0.10673580246913576, 0.3206913580246916, 0.07439999999999997]
	7	[6.54, 2.9533333333333336, 5.515555555555555, 2.0044444444444447]	[0.38862222222222254, 0.09937777777777773, 0.2893135802469137, 0.0728691358024691]
	8	[6.670454545454544, 2.9795454545454545, 5.6, 2.0227272727272725]	[0.3393543388429752, 0.10299070247933885, 0.2918181818181818, 0.07630165289256197]
	9	[6.591836734693878, 2.9693877551020402, 5.557142857142857, 2.020408163265306]	[0.40360683048729706, 0.10294044148271554, 0.3032653061224491, 0.07386922115785088]
	10	[6.611363636363635, 2.986363636363636, 5.5659090909090905, 2.061363636363636]	[0.4187345041322317, 0.1093595041322314, 0.32815599173553717, 0.058734504132231385]

Dataset	Pliegue	Media	Varianza
data_2	1	[0.023758791946308824, 1.0083493736017883, -0.036604474272930725, -0.987915480984336]	[1.973319338646423, 1.032816274638311, 3.0571354688882684, 1.0364782149057548]
	2	[0.020198957640275125, 0.9951739410068744, -0.027267664670658637, -0.9888806387225513]	[1.9721008939288331, 1.0243770552734692, 3.041998754132967, 1.0421378037036497]
	3	[0.01835438053097356, 1.0004836725663706, -0.0280111725663716, -0.9877198893805272]	[1.95835257082152, 1.0401860106847405, 3.0329370004791554, 1.0427035510269775]
	4	[0.01615103632716746, 0.9944668598172492, -0.0220894138622688, -0.9894635168263836]	[1.9639286067422956, 1.0175536753648418, 3.07646390203414, 1.0301747232488738]
	5	[0.016488316303865033, 1.0051648600621943, -0.02814462461128381, -0.9998202354509065]	[1.9775682286729102, 1.02668631614857, 3.002064520574177, 1.0317382344750246]
	6	[0.02352895495895283, 0.9912857554914568, -0.03136296871533171, -1.000730996228086]	[1.9500296693746217, 1.0235159015765483, 3.0602022503637603, 1.0358441281384125]
	7	[0.01582502232142864, 0.9992184598214282, -0.017432232142857104, -0.9946677901785701]	[1.9447342991484398, 1.02881187521504, 3.064128593920915, 1.0311177866567232]
	8	[0.01059570796460177, 0.9978096017699095, -0.035300309734513334, -0.9987381415929172]	[1.9375988195435245, 1.0333971308237349, 3.0340345967122895, 1.0226318465938913]
	9	[0.035772769744160236, 0.9981687875417126, -0.030036662958843188, -0.9904043381535024]	[1.960371369972635, 1.0241272371058758, 3.0651605639016553, 1.0344486903326833]
	10	[0.025262461197339377, 0.9941952549889156, -0.034184589800443387, -1.010174168514412]	[1.957434748043167, 1.0179458596848028, 3.0480350810419004, 1.0279695976498098]

Dataset	Pliegue	Media	Varianza
data_3	1	[13.739038461538463, 1.999807692307692, 2.4436538461538464,	[0.22087792159763306, 0.49232496301775164, 0.05208472633136096,

		17.06923076923077, 106.28846153846153, 2.8388461538461534, 2.9776923076923065, 0.2919230769230769, 1.9001923076923075, 5.527307692307692, 1.0623076923076922, 3.158461538461538, 1115.9615384615386]	6.064437869822484, 112.9360207100592, 0.11169482248520711, 0.15321005917159766, 0.0044963017751479305, 0.1627672707100592, 1.5841696745562126, 0.013598520710059172, 0.13177455621301776, 44402.921597633154]
	2	[13.722115384615385, 2.049423076923077, 2.4636538461538464, 17.028846153846153, 106.3076923076923, 2.8325, 2.966153846153845, 0.2921153846153846, 1.8734615384615383, 5.4575, 1.063653846153846, 3.158846153846153, 1105.8076923076924]	[0.22453590976331353, 0.5151246671597632, 0.05183857248520711, 6.860898668639054, 111.0207100591716, 0.11773413461538461, 0.1582659763313609, 0.005012832840236685, 0.18061109467455622, 1.6059533653846154, 0.01380780325443787, 0.12797174556213015, 46040.96301775148]
	3	[13.7238, 2.024, 2.464600000000001, 17.095999999999997, 105.26, 2.8153999999999995, 2.96, 0.29079999999999995, 1.8971999999999998, 5.422000000000001, 1.0687999999999998, 3.1715999999999998, 1111.44]	[0.22209556000000005, 0.44953199999999993, 0.054512840000000014, 6.9911840000000005, 92.35240000000002, 0.11584483999999998, 0.15290800000000007, 0.004263359999999998, 0.18286415999999994, 1.5002040000000003, 0.011718560000000005, 0.13461344000000003, 44699.806400000016]
	4	[13.722222222222225, 1.9953703703703705, 2.4405555555555565, 17.037037037037038, 106.29629629629629, 2.835555555555555, 2.973518518518518, 0.2827777777777778, 1.91037037037037, 5.552407407407408, 1.0568518518518517, 3.175925925925926, 1123.5555555555557]	[0.20172469135802473, 0.42489523319615896, 0.049683024691358026, 6.001591220850484, 111.98628257887519, 0.12061728395061726, 0.1533820644718793, 0.004464506172839506, 0.15278504801097392, 1.5884627229080932, 0.012454903978052122, 0.12255377229080933, 50820.76543209875]

	5	[13.753148148148147, 1.9844444444444447, 2.448518518518519, 16.987037037037034, 106.14814814814815, 2.8537037037037036, 2.999074074074074, 0.2922222222222222, 1.8620370370370365, 5.53425925925926, 1.0659259259259257, 3.1312962962962962, 1118.462962962963]	[0.197980829903978, 0.4434802469135801, 0.053812620027434845, 6.500757887517146, 105.8669410150892, 0.11732702331961596, 0.1573009945130315, 0.005080246913580245, 0.1569828875171468, 1.6163429698216734, 0.014353772290809324, 0.11945202331961591, 49786.13751714677]
	6	[13.744444444444445, 1.9953703703703707, 2.4664814814814826, 17.027777777777775, 106.57407407407408, 2.8555555555555547, 2.9927777777777775, 0.28944444444444445, 1.9001851851851848, 5.6255555555555565, 1.0529629629629629, 3.1716666666666664, 1113.8333333333333]	[0.2100395061728395, 0.4561248628257887, 0.051078360768175596, 6.811265432098766, 116.31858710562412, 0.1085283950617284, 0.1592904320987654, 0.0048570987654321, 0.16889811385459536, 1.482658024691358, 0.013020850480109734, 0.12815833333333335, 49306.0277777778]
	7	[13.769090909090908, 2.0390909090909095, 2.4590909090909094, 17.04, 106.81818181818181, 2.852, 3.0061818181818176, 0.292, 1.9261818181818182, 5.52309090909091, 1.059272727272727, 3.1541818181818186, 1112.2181818181818]	[0.20598280991735546, 0.4858809917355373, 0.05080826446280993, 6.806763636363637, 109.05785123966945, 0.11478327272727272, 0.14721269421487604, 0.005056000000000002, 0.16582723966942148, 1.4168686280991738, 0.01334128925619835, 0.12345705785123968, 50253.1523966942]
	8	[13.7556862745098, 1.9666666666666666, 2.457843137254902, 16.903921568627453, 107.03921568627452, 2.8135294117647054, 2.96235294117647, 0.2888235294117647, 1.9078431372549014, 5.48549019607843, 1.0727450980392157,	[0.22338923490965013, 0.385586928104575, 0.05302868127643215, 6.3317493271818535, 111.92003075740098, 0.0825679354094579, 0.14849642445213385, 0.005418223760092274, 0.17668750480584397, 1.225907112648981, 0.01201207227989235,

		3.1519607843137254, 1114.8431372549019]	0.12944321414840448, 46715.15186466745]
	9	[13.750943396226415, 2.0283018867924527, 2.4694339622641515, 17.29433962264151, 106.81132075471699, 2.852641509433961, 2.9873584905660384, 0.29245283018867924, 1.9111320754716976, 5.557358490566037, 1.0598113207547168, 3.145849056603773, 1112.867924528302]	[0.1909406194375222, 0.5126065503737985, 0.04803175507297973, 5.796005695977216, 109.17194731221078, 0.12202321110715561, 0.16482321110715556, 0.005052474190103241, 0.14922135991456034, 1.5334269846920612, 0.014726379494482017, 0.11668465646137412, 52316.190103239576]
	10	[13.763928571428574, 2.023214285714286, 2.443392857142858, 16.89642857142857, 105.82142857142857, 2.848035714285714, 2.994285714285714, 0.2876785714285714, 1.9035714285714282, 5.58375, 1.059285714285714, 3.15875, 1127.0]	[0.20180599489795917, 0.48593609693877554, 0.042154559948979584, 5.527487244897958, 100.18239795918365, 0.11497292729591833, 0.1557923469387755, 0.004453539540816327, 0.17138724489795917, 1.4885484375, 0.013888775510204076, 0.11908593749999999, 47224.28571428572]

Dataset	Pliegue	Media	Varianza
data_4	1	[90.0]	[16.666666666666668]
	2	[89.82142857142857]	[16.93239795918367]
	3	[90.1923076923077]	[16.30917159763314]
	4	[89.8]	[16.96]
	5	[90.17241379310344]	[16.349583828775266]
	6	[89.6]	[15.840000000000005]
	7	[90.17857142857143]	[16.93239795918367]
	8	[90.18518518518519]	[15.706447187928669]
	9	[90.0]	[17.24137931034483]
	10	[90.0]	[17.307692307692307]

IV. Todo el código fuente

#main.py

```
from program import *



menu = {
    1:'manual',
    2:'automático',
    3:'salir'
}

menu_datos = {
    1:'data_1.csv',
    2:'data_2.csv',
    3:'data_3.csv',
    4:'data_4.csv'
}

def print_menu():
    """imprime el menú"""

    print('\nMenu:')
    for key in menu.keys():
        print (key, '--', menu[key])

def print_menu_datos():
    """imprime el submenú de datos"""

    print('\nSubmenu:')
```

```
    for key in menu_datos.keys():
        print (key, '--', menu_datos[key])



#Seleccionar y actuar
flag = True

while flag == True:

    print_menu()
    opcion = int(input('Seleccione una opción: '))

    #Manual
    if opcion == 1:

        #selección de datos
        print_menu_datos()
        opcion_datos = int(input('Seleccione una opción: '))

        if opcion_datos > 4:
            print(('Opción inválida. Reiniciando...'))
            continue

        else:
            reduccion = input('reducir características? si/no: ')
            if reduccion == 'si':
                grafica = input('graficar? si/no: ')
            datos = menu_datos[opcion_datos]
            df = pd.read_csv(datos,header=None)
```

```
        if reduccion == 'si':
            df = reducir(df,datos)
            if grafica == 'si':
                label,clases_separadas,clases = separar_clases(df,df)
                graficar(df,clases,datos)
        print("\n.....",datos,".....")

matriz_masterlist,exactitud_masterlist,precision_masterlist,sensibilidad_masterlist,fscore_masterlist = validacion_cruzada(df)
        matriz_total,exactitud_total,precision_total,sensibilidad_total,fscore_total = resultado_final(matriz_masterlist,exactitud_masterlist,precision_masterlist,sensibilidad_masterlist,fscore_masterlist)



    #Automático
    elif opcion == 2:
        reduccion = input('reducir características? si/no: ')
        if reduccion == 'si':
            grafica = input('graficar? si/no: ')

        for key in menu_datos.keys():
            print("\n.....",menu_datos[key],".....")
            df = pd.read_csv(menu_datos[key],header=None)

            if reduccion == 'si':
                df = reducir(df,menu_datos[key])

                if grafica == 'si':
                    label,clases_separadas,clases = separar_clases(df,df)
                    graficar(df,clases,menu_datos[key])
```

```
matriz_masterlist,exactitud_masterlist,precision_masterlist,sensibilidad_masterlist,fscore_masterlist = validacion_cruzada(df)

        matriz_total,exactitud_total,precision_total,sensibilidad_total,fscore_total = resultado_final(matriz_masterlist,exactitud_masterlist,precision_masterlist,sensibilidad_masterlist,fscore_masterlist)


    #Salir
    elif opcion == 3:
        print('\nSaliendo...\n')
        flag = False


    else:
        print('Opción inválida. Ingrese un número entre 1 y 3.')
```

#program.py

```
import pandas as pd
import numpy as np
from math import sqrt, pi, e
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt



def reducir(df, datos):
    """reduce las caracteristicas a dos columnas por medio del metodo PCA"""


    if datos == 'data_4.csv':
        df = df.replace({'yes': 1, 'no': 2, 'single': 1,
                         'married': 2, 'divorced': 3})
```

```
    label = df.iloc[:, -1]
    label = pd.DataFrame(label)
    data = df.iloc[:, :-1]
    data = pd.DataFrame(data)

    p = PCA(2)
    pca_columns = p.fit_transform(data)
    pca_columns = pd.DataFrame(pca_columns)
    df = pca_columns.join(label)
    df.columns = range(df.columns.size)

    return df


def graficar(df, clases, datos):
    """traza un grafico de dispersión"""
    scatter = plt.scatter(
        df.iloc[:, 0], df.iloc[:, 1], c=df.iloc[:, 2], cmap='tab20b', alpha=0.6)
    cbar = plt.colorbar(scatter)
    cbar.set_ticks(clases)
    plt.title(datos)
    plt.show()


def separar_clases(df, df_shuffled):
    """separa las clases en un diccionario"""

    label = df_shuffled.iloc[:, -1]
```

```
    clases_separadas = dict()
    for i in range(len(df)-1):
        clase = label[i]
        if clase not in clases_separadas:
            clases_separadas[clase] = list()
            clases_separadas[clase].append(i)
        else:
            clases_separadas[clase].append(i)

    clases = list(clases_separadas.keys())

    return label, clases_separadas, clases



def laplace(PXC, count_all_when, count_all, test):
    """aplica la técnica de suavizado Laplace que ayuda solucionar el problema de la probabilidad cero"""

    alpha = 1
    if PXC == 0.0:
        PXC = (count_all_when + alpha) / (count_all + alpha * test.columns[-1])

    return PXC



def calcular_media(train_column_seleccion):
    """calcula la media"""

    number_list = []
    for i in range(len(train_column_seleccion)):
```

```
        number = train_column_seleccion.iloc[i]
        number_list.append(number)

    media = sum(number_list) / len(number_list)

    return media


def calcular_varianza(media, train_column_seleccion):
    """calcula la varianza"""

    desviacion_list = []
    for i in train_column_seleccion:
        desviacion = (i - media) ** 2
        desviacion_list.append(desviacion)

    varianza = sum(desviacion_list) / len(train_column_seleccion)

    return varianza


def densidad_probabilidad(x, test, train, train_column, key, column, count_all, PXC_list,
media_list, varianza_list):
    """calcula la densidad de probabilidad para caracteristicas continuas y discretas"""

    if type(x) == str:

        count_all_when = len(train.loc[(train[train.columns[-1]] == key) &
                                       (train_column == x)])
```

```
            PXC = count_all_when / count_all

            PXC = laplace(PXC, count_all_when, count_all, test)

            PXC_list.append(PXC)

        else:

            train_column_seleccion = train.loc[(train[train.columns[-1]] == key)]
            train_column_seleccion = train_column_seleccion.iloc[:, column+1]

            media = calcular_media(train_column_seleccion)
            media_list.append(media)

            varianza = calcular_varianza(media, train_column_seleccion)
            varianza_list.append(varianza)

            PXC = (1 / (sqrt(2 * pi * varianza))) * \
                ((e) ** ((-(x - media)**2)/(2 * varianza)))

        return PXC


def bayes(PXC_list, count_all, train):
    """calcula la probabilidad de Bayes"""

    PXC_result = 1.0
    for j in PXC_list:

        PXC_result = PXC_result * j
```

```
    PC = count_all / len(train)

    probabilidad = PXC_result * PC

    return probabilidad


def pertenencia(probabilidad_list, clases):
    """determina a cuál clase pertenece en base a la probabilidad"""

    probabilidad_max = 0
    for i in range(len(probabilidad_list)):

        number = probabilidad_list[i]
        if number > probabilidad_max:
            probabilidad_max = number

    if probabilidad_max == probabilidad_list[0]:
        prediccion = clases[0]

    elif probabilidad_max == probabilidad_list[1]:
        prediccion = clases[1]

    else:
        prediccion = clases[2]

    return probabilidad_max, prediccion
```

```
def exactitud_precision_sensibilidad_fscore(TP, FN, FP, TN):
    """calcula la exactitud, precision, sensibilidad y f-score"""

    exactitud = (TP + TN) / (TP + FN + FP + TN)

    precision = TP / (TP + FP)

    sensibilidad = TP / (TP + FN)

    fscore = (2 * sensibilidad * precision) / (sensibilidad + precision)

    return exactitud, precision, sensibilidad, fscore



def matriz_confusion(clases, real_list, prediccion_list):
    """calcula e imprime la matriz de confusión"""

    matriz = [[sum([(real_list[i] == clase_real) and (prediccion_list[i] == clase_pred)
                    for i in range(len(real_list))])
               for clase_pred in clases]
              for clase_real in clases]
    matriz = np.stack(matriz)

    print('\nmatriz de confusion =\n', matriz)
    return matriz



def indices(clases, matriz):
    """obtiene los valores de la matriz de confusión para poder calcular la exactitud, precisión, sensibilidad y f-score y luego imprime los resultados"""
```

```
exactitud_list = []
precision_list = []
sensibilidad_list = []
fscore_list = []
if len(clases) == 2:
    TP = matriz[0, 0]
    TN = matriz[1, 1]
    FP = matriz[1, 0]
    FN = matriz[0, 1]

    exactitud, precision, sensibilidad, fscore = exactitud_precision_sensibilidad_fscore(
        TP, FN, FP, TN)

    print('\nexactitud = ', exactitud)
    print('\nprecision = ', precision)
    print('\nsensibilidad = ', sensibilidad)
    print('\nfscore = ', fscore)

elif len(clases) == 3:

    exactitud_parcial_list = []
    precision_parcial_list = []
    sensibilidad_parcial_list = []
    fscore_parcial_list = []

    a = matriz[0, 0]
    b = matriz[0, 1]
    c = matriz[0, 2]
    d = matriz[1, 0]
    e = matriz[1, 1]
```

```
f = matriz[1, 2]
g = matriz[2, 0]
h = matriz[2, 1]
i = matriz[2, 2]

C1_TP = a
C1_FN = b + c
C1_FP = d + g
C1_TN = e + f + h + i

exactitud, precision, sensibilidad, fscore = exactitud_precision_sensibilidad_fscore(
    C1_TP, C1_FN, C1_FP, C1_TN)

exactitud_parcial_list.append(exactitud)
precision_parcial_list.append(precision)
sensibilidad_parcial_list.append(sensibilidad)
fscore_parcial_list.append(fscore)

C2_TP = e
C2_FN = d + f
C2_FP = b + h
C2_TN = a + c + g + i

exactitud, precision, sensibilidad, fscore = exactitud_precision_sensibilidad_fscore(
    C2_TP, C2_FN, C2_FP, C2_TN)

exactitud_parcial_list.append(exactitud)
precision_parcial_list.append(precision)
sensibilidad_parcial_list.append(sensibilidad)
fscore_parcial_list.append(fscore)
```

```
    C3_TP = i
    C3_FN = g + h
    C3_FP = c + f
    C3_TN = a + b + d + e

    exactitud, precision, sensibilidad, fscore = exactitud_precision_sensibilidad_fscore(
        C3_TP, C3_FN, C3_FP, C3_TN)

    exactitud_parcial_list.append(exactitud)
    precision_parcial_list.append(precision)
    sensibilidad_parcial_list.append(sensibilidad)
    fscore_parcial_list.append(fscore)

    print('\nexactitud = ', exactitud_parcial_list)
    print('\nprecision = ', precision_parcial_list)
    print('\nsensibilidad = ', sensibilidad_parcial_list)
    print('\nfscore = ', fscore_parcial_list)

    exactitud = sum(exactitud_parcial_list) / len(exactitud_parcial_list)
    precision = sum(precision_parcial_list) / len(precision_parcial_list)
    sensibilidad = sum(sensibilidad_parcial_list) / len(sensibilidad_parcial_list)
    fscore = sum(fscore_parcial_list) / len(fscore_parcial_list)

    print('\nexactitud promedio = ', exactitud)
    print('\nprecision promedio = ', precision)
    print('\nsensibilidad promedio = ', sensibilidad)
    print('\nfscore promedio = ', fscore)

exactitud_list.append(exactitud)
```

```
    precision_list.append(precision)
    sensibilidad_list.append(sensibilidad)
    fscore_list.append(fscore)

    return exactitud_list, precision_list, sensibilidad_list, fscore_list

def
resultado_final(matriz_masterlist,exactitud_masterlist,precision_masterlist,sensibilidad_masterli
st,fscore_masterlist):
    print('\n.....')
    print('\n..... resultado final .....')

    matriz_total = sum(matriz_masterlist)
    print('\n matriz de confusión final=\n',matriz_total)

    exactitud_masterlist = sum(exactitud_masterlist, [])
    exactitud_total = sum(exactitud_masterlist) / len(exactitud_masterlist)
    print('\n exactitud final=',exactitud_total)

    precision_masterlist = sum(precision_masterlist, [])
    precision_total = sum(precision_masterlist) / len(precision_masterlist)
    print('\n precision final=',precision_total)

    sensibilidad_masterlist = sum(sensibilidad_masterlist, [])
    sensibilidad_total = sum(sensibilidad_masterlist) / len(sensibilidad_masterlist)
    print('\n sensibilidad final=',sensibilidad_total)

    fscore_masterlist = sum(fscore_masterlist, [])
    fscore_total = sum(fscore_masterlist) / len(fscore_masterlist)
    print('\n f-score final=',fscore_total)
    print('\n.....')
```

```
    print('\n.....................................................')
    return matriz_total,exactitud_total,precision_total,sensibilidad_total,fscore_total


def validacion_cruzada(df):
    """'divide aleatoriamente el conjunto de entrenamiento en 10 grupos de aproximadamente el mismo tamaño y realiza los cálculos para cada grupo"""

    fold = 10

    df_shuffled = df.sample(frac=1, random_state=1)
    df_shuffled = df_shuffled.reset_index()

    folds = np.array_split(df_shuffled, 10)

    prediccion_masterlist = []
    real_masterlist = []
    media_masterlist = []
    varianza_masterlist = []
    matriz_masterlist = []
    exactitud_masterlist = []
    precision_masterlist = []
    sensibilidad_masterlist = []
    fscore_masterlist = []

    for f in range(fold):
        print('\n..................... fold', f+1, '.....................')
        test = pd.concat(folds[f:f+1])
        train = pd.concat(folds[0:f]+folds[f+1::])

        label, clases_separadas, clases = separar_clases(df, df_shuffled)
```

```
    prediccion_list = []
    real_list = test.iloc[:, -1].values.tolist()
    real_masterlist.append(real_list)

    for row in range(len(test)):

        probabilidad_list = []

        for key in clases:

            PXC_list = []
            media_list = []
            varianza_list = []

            for column in range(test.columns[-1]):

                x = test.iloc[row][column]
                train_column = train.iloc[:, column+1]

                count_all = len(train[(train[train.columns[-1]] == key)])

                PXC = densidad_probabilidad(
                    x, test, train, train_column, key, column, count_all, PXC_list, media_list, varianza_list)
                PXC_list.append(PXC)

            probabilidad = bayes(PXC_list, count_all, train)
            probabilidad_list.append(probabilidad)
```

```
            probabilidad_max, prediccion = pertenencia(
                probabilidad_list, clases)

            prediccion_list.append(prediccion)

        media_masterlist.append(media_list)
        varianza_masterlist.append(varianza_list)
        prediccion_masterlist.append(prediccion_list)

        print('\nmedia =\n', media_list)
        print('\nvarianza =\n', varianza_list)

        matriz = matriz_confusion(clases, real_list, prediccion_list)

        matriz_masterlist.append(matriz)

        exactitud_list, precision_list, sensibilidad_list, fscore_list = indices(clases, matriz)

        exactitud_masterlist.append(exactitud_list)
        precision_masterlist.append(precision_list)
        sensibilidad_masterlist.append(sensibilidad_list)
        fscore_masterlist.append(fscore_list)

    return
matriz_masterlist,exactitud_masterlist,precision_masterlist,sensibilidad_masterlist,fscore_maste
rlist
```

7) Referencias

“Factor analysis of mixed data,” *Wikipedia*, 22-Mar-2022. [Online]. Available: https://en.wikipedia.org/wiki/Factor_analysis_of_mixed_data. [Accessed: 09-May-2022].