Universidad Politécnica de Tulancingo
Secretaría Académica
Dirección de Posgrado
Maestría en Computación Óptica

Asignatura:	Inteligencia Artificial	Unidad de aprendizaje:	2
Cuatrimestre	3ro.	Grupo:	MCO 3
Periodo:	Mayo-Agosto 2022	Fecha:	08 Julio 2022
Nombre del Profesor/a:	Dra. Rosa María Ortega Mendoza		
Nombre del alumno/a:	Ana Teresa Vargas Barona		
Tipo de examen:	Parcial	Calificación:	

NSTRUCCIONES: Realice un sistema de clasificación en python sobre un conjunto de datos (dataset de su elección usado en el parcial 1). Los puntos a evaluar son:

1. (obtuve__1__ de **1** punto). Dividir el dataset en 2 particiones (entrenamiento y prueba)
2. (obtuve__1__ de **1** punto). Aplicación el algoritmo de clasificación regresión logística.
    a. Aplicación del algoritmo
    b. Evaluación del algoritmo (matriz de confusión y reporte de clasificación)
    c. Análisis de resultados
3. (obtuve__1__ de **1** punto). Normalización de datos.
4. (obtuve__1__ de **1** punto). Aplicar la técnica PCA con 3 diferentes números de componentes n=2,3,4.
    a. Imprimir un segmento del dataset por cada caso
5. (obtuve__1__ de **1** punto). Clasificar usando regresión logística.
    a. Clasificar usando PCA con n=2, después n=3 y finalmente n=4 en las mismas particiones de entrenamiento y prueba del inciso 1
    b. Evaluación en cada conjunto de PCA (matriz de confusión y reporte de clasificación)
6. (obtuve__1__ de **1** punto). Analizar y contrastar resultados con gráficas de barras (pueden ser en Excel)
    a. del uso de PCA con n=2,3,4 (punto anterior)
    b. del mejor (n=2,3 o 4) y el uso de todas las características (punto 2)
7. (obtuve__1__ de **1** punto). Graficar resultados con PCA (n=2)
8. (obtuve__1__ de **1** punto). Aplicar k-means para generar 3 grupos y graficar. Tratar de interpretar los grupos
9. (obtuve__1__ de **1** punto). Implementar el perceptrón simple de sklearn al conjunto de datos.
10. (obtuve__1__ de **1** punto). Calidad de su reporte y contenido**:** *objetivo, introducción, desarrollo (código y resultado del código (imagen)), conclusiones, y bibliografía.).*

EXAMEN SEGUNDO PARCIAL

INTRODUCCIÓN

La cantidad de algoritmos que se utilizan en Inteligencia artificial son diversos y lo cual dependiendo la problemática que se busca solucionar, es el algoritmo que se implementa. Pero, como se ha logrado observar estos algoritmos se clasifican en diferentes formas de aprendizaje, los cuales solo no enfocaremos en el supervisado y no supervisado.

Los algoritmos de aprendizaje supervisado basan su aprendizaje de un conjunto de datos de entrenamiento, junto con las clases o etiquetas. Esto le permitirá al algoritmo poder "aprender" una función, para ser capaz de predecir la clase para un conjunto de datos nuevos.

El aprendizaje no supervisado está dedicado a las tareas de agrupamiento, también llamadas clustering, donde su objetivo es encontrar grupos similares en el conjunto de datos. En el siguiente documento resolveremos una problemática implementando ambos tipos de aprendizajes para poder visualizar el funcionamiento de cada uno.

PLANTREAMIENTO DEL PROBLEMA

El juego comúnmente conocido como "Gato o tic-tac-toe", consiste en un tablero de nueve casillas, 3 filas y 3 columnas, cada una de estas casillas tienen tres posibles valores en su contenido se X, O y b, donde X y O representa al jugador 1 y jugador 2 respectivamente y b representa un espacio en blanco.

El juego consiste en alternar turnos entre los dos jugadores, colocando su respectiva ficha en la casilla elegida, el juego culmina cuando uno de los jugadores realiza un alineación de 3 de las mismas figuras, en caso de no conseguirlo ninguno de los jugadores, se considera un empate.

Al ser un juego simple y conocido de manera internacional es un excelente candidato para implementar algunos algoritmos de Inteligencia Artificial. El dataset implementado consiste en el juego de gato, donde el jugador que utilice "X" gane el mayor número de partidas, siendo el quien realice las aperturas, pero en caso de empatar o perder, la partida se mostrará como derrota. En el siguiente reporte se

mostrara la implementación de un algoritmo de aprendizaje supervisado y uno de no supervisado, siendo un problema de clasificación binaria.

DESARROLLO

El desarrollo e implementación de estos algoritmos requirió que algunos puntos fueran realizados de diferente orden, esto se debe al instante de utilizar algunos procedimientos con los cuales dependen otros métodos.

Como primer implementación con respecto a la codificación, es importante el tener listas las librerías a utilizar.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import *
from sklearn.linear_model import *
from sklearn.metrics import *
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.linear_model import Perceptron
import warnings
import seaborn as sns
```

Una vez importadas las librerías, se procede a leer el dataset que se necesita

```
warnings.filterwarnings('ignore')
# -------- Leemos el dataset
instancias = ['top-left-square', 'top-middle-square', 'top-right-square',
'middle-left-square', 'middle-middle-square',
              'middle-right-square', 'bottom-left-square', 'bottom-
middle-square', 'bottom-right-square', 'Class']
df = pd.read_csv('tic-tac-toe.csv', names=instancias)  # Unimos el vector
de las instancias con las caracteristicas

# -------- Separamos las caracteristicas y la clase
features = df[['top-left-square', 'top-middle-square', 'top-right-
square', 'middle-left-square', 'middle-middle-square',
               'middle-right-square', 'bottom-left-square', 'bottom-
middle-square', 'bottom-right-square']]
clase = df['Class']

print('Información sobre el dataframe: ')
print(df.info())
print('Dimensión: ')
print(df.shape)
pd.set_option('display.max_columns', None)
print('Head')
print(df.head())
```

```
Información sobre el dataframe:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 958 entries, 0 to 957
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   top-left-square       958 non-null    object
 1   top-middle-square     958 non-null    object
 2   top-right-square      958 non-null    object
 3   middle-left-square    958 non-null    object
 4   middle-middle-square  958 non-null    object
 5   middle-right-square   958 non-null    object
 6   bottom-left-square    958 non-null    object
 7   bottom-middle-square  958 non-null    object
 8   bottom-right-square   958 non-null    object
 9   Class                 958 non-null    object
dtypes: object(10)
memory usage: 75.0+ KB
None
Dimensión:
(958, 10)
```

Figura 1. Información sobre
El dataset y su dimensión.

```
Head
  top-left-square top-middle-square top-right-square middle-left-square  \
0               x                 x                x                   x
1               x                 x                x                   x
2               x                 x                x                   x
3               x                 x                x                   x
4               x                 x                x                   x

  middle-middle-square middle-right-square bottom-left-square  \
0                    o                   o                 x
1                    o                   o                 o
2                    o                   o                 o
3                    o                   o                 o
4                    o                   o                 b

  bottom-middle-square bottom-right-square     Class
0                    o                   o  positive
1                    x                   o  positive
2                    o                   x  positive
3                    b                   b  positive
4                    o                   b  positive
```

Figura 2. Contenido de los
Primeros 5 datos del dataset.

Como se puede ver, la primera línea nos ayuda a ignorar los mensajes “warnings” que se llegue a mostrar, con la finalidad de brindar una mejor presentación al instante de visualizar los resultados. Mandamos a imprimir la información sobre el dataset, su dimensión y que nos los primeros 5 datos que contiene.

En la figura 1 podemos observar las instancias y la cantidad de atributos que tiene cada una, así mismo el tipo de dato, después la dimensión del dataset que es de (958, 10), lo cual son 958 atributos, 9 instancias y 1 clase. En la Figura 2, logramos observar los primeros 5 atributos con su clase a la que corresponden.

3. Normalización de datos.

```
# ---------------------------------- PUNTO 3 Normalización de los datos

# Atributos
new_instances = pd.get_dummies(features)
pd.set_option('display.max_columns', None)
print('Instancias normalizadas: ')
print(new_instances)

# Clases
labeEncor = LabelEncoder()
clase2 = labeEncor.fit_transform(clase)
print('Clases normalizadas: ')
print(clase2)
```

Como se mencionó anterior mente algunos puntos se cambiaron de orden de implementación debido al uso que se les brinda, en este caso el punto 3 que consiste en la normalización de datos es muy importante, ya que debemos tener

todos los valores en una misma escala, esto es implementado en atributos y clases lo cual se muestra en las figuras 3 y 4. Como el dataset es de clasificación binaria al implementar la normalización tenemos los valores de 1 que representa positivo y 0 que representa negativo, lo cual en el juego sería victoria o derrota.

```
Instancias normalizadas:
     top-left-square_b  top-left-square_o  top-left-square_x  \
0                    0                  0                  1
1                    0                  0                  1
2                    0                  0                  1
3                    0                  0                  1
4                    0                  0                  1
..                 ...                ...                ...
953                  0                  1                  0
954                  0                  1                  0
955                  0                  1                  0
956                  0                  1                  0
957                  0                  1                  0

     top-middle-square_b  top-middle-square_o  top-middle-square_x  \
0                      0                    0                    1
1                      0                    0                    1
2                      0                    0                    1
3                      0                    0                    1
4                      0                    0                    1
..                   ...                  ...                  ...
953                    0                    0                    1
954                    0                    0                    1
955                    0                    0                    1
956                    0                    0                    1
957                    0                    1                    0
```

Figura 3. Normalización de los atributos.

```
Clases normalizadas:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Figura 4. Normalización de las clases.

<u>1. Dividir el dataset en 2 particiones (entrenamiento y prueba)</u>

Una vez realizada la normalización, procedemos a dividir el dataset en dos particiones, la de entrenamiento y prueba, lo cual corresponde al primer punto solicitado.

```
# ------------ PUNTO 1 Dividir el dataset, 80% entrenamiento y 20% test

x_train, x_test, y_train, y_test = train_test_split(new_instances,
clase2, test_size=0.20)
```

Ya teniendo las particiones de entrenamiento y prueba, se procede a implementar el punto dos, donde se solicita implementar regresión logística, el cual es un algoritmo de aprendizaje supervisado y mostrar su matriz de confusión y reporte de clasificación.

2. Aplicación el algoritmo de clasificación regresión logística.

```
# ---------------------------------- PUNTO 2 Regresión logística

lg = LogisticRegression()
RL = lg.fit(x_train, y_train)
c3 = RL.predict(x_test)
c4 = RL.predict(x train)

print('Accuracy prueba: ', accuracy_score(y_test, c3))
print('Accuracy entrenamiento: ', accuracy_score(y_train, c4))

# ----- Datos de prueba
# a) Matríz de Confusión
print('Matriz de Confusión Prueba: ')
cm2 = confusion_matrix(y_test, c3)
print(cm2)

# b) Reporte de Clasificación
print('Reporte de Clasificación Prueba:')
cr2 = classification_report(y_test, c3)
print(cr2)

# ----- Datos de entrenamiento
# a) Matríz de Confusión
print('Matriz de Confusión Entrenamiento: ')
cm3 = confusion_matrix(y_train, c4)
print(cm3)

# b) Reporte de Clasificación
print('Reporte de Clasificación Entrenamiento:')
cr3 = classification_report(y_train, c4)
print(cr3)
```

Como primer punto, tenemos que generar una instancia para poder invocar la regresión logística, una vez implementada obtenemos el accuracy de los datos de entrenamiento y de prueba, como lo muestra la figura 5, logramos observar que tienen un valor muy bueno, e incluso notamos que el accuracy de entrenamiento es ligeramente mayor comparado con el de prueba, sin embargo ambos se encuentran dentro del rango considerado aceptable.

```
Accuracy prueba:  0.9791666666666666
Accuracy entrenamiento:  0.9830287206266318
```

Figura 5. Accuracy prueba y entrenamiento.

Procede a mostrar la matriz de confusión y reporte de clasificación de prueba y entrenamiento.

```
Matriz de Confusión Prueba:
[[ 61   3]
 [  1 127]]
Reporte de Clasificación Prueba:
              precision    recall  f1-score   support

           0       0.98      0.95      0.97        64
           1       0.98      0.99      0.98       128

    accuracy                           0.98       192
   macro avg       0.98      0.97      0.98       192
weighted avg       0.98      0.98      0.98       192

Matriz de Confusión Entrenamiento:
[[255  13]
 [  0 498]]
Reporte de Clasificación Entrenamiento:
              precision    recall  f1-score   support

           0       1.00      0.95      0.98       268
           1       0.97      1.00      0.99       498

    accuracy                           0.98       766
   macro avg       0.99      0.98      0.98       766
weighted avg       0.98      0.98      0.98       766
```

Figura 6. Matriz de confusión y reporte de clasificación (prueba y entrenamiento)

La figura 6 nos muestra la matriz de confusión de los datos de prueba y entrenamiento en la cual si recordamos almacenamos los valores que son verdaderos positivos, falsos positivos, falsos negativos y verdadero negativo.

Figura 7. Estructura de la matriz de confusión.

Como lo muestra la figura 7, base a esta representación podemos ver que el entrenamiento tiene 0 valores en falsos negativos a diferencia de los valores de la matriz de datos de prueba, más sin embargo la matriz de los datos de entrenamiento tiene más valores de falsos positivos.

En el caso de los reportes de clasificación, en la situación de los datos de prueba, se puede observar que las clases negativas y positivas tuvieron una precisión de 0.98, pero el Recall y f1 los valores de las clases positivas fueron más altos.

En la situación del reporte de clasificación de los datos de entrenamiento, tenemos que la precisión de la clase negativa es mayos que al de la positiva, más sin embargo en el Recall y el f1 es invertido, pues los valores de la clase positiva son mayores que al de la negativa.

4. Aplicar la técnica PCA con 3 diferentes números de componentes n=2, 3 y 4.

```
# ---------------------------------- PUNTO 4 Implementación del PCA

pca2 = PCA(n_components=2)
pca_x2 = pca2.fit_transform(x_train)
print('Dimensión de PCA 2: ', pca_x2.shape)
print('Primeros valores de PCA 2: ', pca_x2[0,:])
```

```
pca3 = PCA(n_components=3)
pca_x3 = pca3.fit_transform(x_train)
print('Dimensión de PCA 3: ', pca_x3.shape)
print('Primeros valores de PCA 3: ', pca_x3[0,:])

pca4 = PCA(n_components=4)
pca_x4 = pca4.fit_transform(x_train)
print('Dimensión de PCA 4: ', pca_x4.shape)
print('Primeros valores de PCA 4: ', pca_x4[0,:])
```

El siguiente punto nos indica que implementemos el PCA con diferentes números de componentes, esto nos permite realizar una reducción con respecto aquellas instancias que no sean necesarias o de gran relevancia, dando como resultado lo que se muestra en la figura 8.

```
Dimensión de PCA 2:  (766, 2)
Primeros valores de PCA 2:  [1.0076075  0.07344342]
Dimensión de PCA 3:  (766, 3)
Primeros valores de PCA 3:  [1.01262277 0.05353398 0.49826151]
Dimensión de PCA 4:  (766, 4)
Primeros valores de PCA 4:  [-1.01709054  0.07430883  0.45745383  1.22222707]
```

Figura 8. Dimensiones y primeros valores del PCA implementado.

5. Clasificar usando regresión logística.

a. Clasificar usando PCA con n=2, después n=3 y finalmente n=4 en las mismas particiones de entrenamiento y prueba del inciso 1

```
# ------------------------ PUNTO 5 APARTADO A PCA y regresión logística

#PARA n = 2
lg2 = LogisticRegression()

t2 = pca2.transform(x_test)
RL2 = lg2.fit(pca_x2, y_train)
pre2 = lg2.predict(t2)

#PARA n = 3
lg3 = LogisticRegression()

t3 = pca3.transform(x_test)
RL3 = lg3.fit(pca_x3, y_train)
pre3 = lg3.predict(t3)

#PARA n = 4
```

```
lg4 = LogisticRegression()

t4 = pca4.transform(x_test)
RL4 = lg4.fit(pca_x4, y_train)
pre4 = lg4.predict(t4)
```

Como nos indica el punto 5 e inciso a, a los valores que obtuvimos de implementar PCA para diferentes números de componentes, debemos implementar regresión logística para cada uno de ellos.

b. Evaluación en cada conjunto de PCA (matriz de confusión y reporte de clasificación)

```
# ------------------------------ PUNTO 5 APARTADO B Evaluación del PCA

# CON 2 COMPONENTES

# a) Matríz de Confusión
print('Matriz de Confusión PCA2: ')
cm22 = confusion_matrix(y_test, pre2)
print(cm22)

# b) Reporte de Clasificación
print('Reporte de Clasificación PCA2:')
cr22 = classification_report(y_test, pre2)
print(cr22)

# CON 3 COMPONENTES

# a) Matríz de Confusión
print('Matriz de Confusión PCA3: ')
cm33 = confusion_matrix(y_test, pre3)
print(cm33)

# b) Reporte de Clasificación
print('Reporte de Clasificación PCA3:')
cr33 = classification_report(y_test, pre3)
print(cr33)

# CON 4 COMPONENTES

# a) Matríz de Confusión
print('Matriz de Confusión PCA4: ')
cm44 = confusion_matrix(y_test, pre4)
print(cm44)

# b) Reporte de Clasificación
print('Reporte de Clasificación PCA4:')
cr44 = classification_report(y_test, pre4)
print(cr44)
```

Al igual como se implementó con el caso de regresión logística, tenemos que mostrar la matriz de confusión y el reporte de clasificación, pero en este caso es implementado para cada uno de los componentes del PCA.

```
Matriz de Confusión PCA2:
[[  2  62]
 [  0 128]]
Reporte de Clasificación PCA2:
              precision    recall  f1-score   support

           0       1.00      0.03      0.06        64
           1       0.67      1.00      0.81       128

    accuracy                           0.68       192
   macro avg       0.84      0.52      0.43       192
weighted avg       0.78      0.68      0.56       192
```

Figura 9. Matriz de confusión y reporte de clasificación para PCA con n=2.

Como podemos observar la matriz de confusión del PCA2 no tiene valores falsos negativos, más sin embargo los valores falsos positivos son demasiados, con el reporte de clasificación vemos que la precisión de las clases negativas es muy alta, comparada con la positiva, pero su Recall y f1 son excesivamente bajos.

```
Matriz de Confusión PCA3:
[[  1  63]
 [  0 128]]
Reporte de Clasificación PCA3:
              precision    recall  f1-score   support

           0       1.00      0.02      0.03        64
           1       0.67      1.00      0.80       128

    accuracy                           0.67       192
   macro avg       0.84      0.51      0.42       192
weighted avg       0.78      0.67      0.55       192
```

Figura 10. Matriz de confusión y reporte de clasificación para PCA con n=3.

Pero con el PCA3 sucede lo mismo, a diferencia que sus valores verdaderos positivos son excesivamente muy bajos, en cambio en el reporte de clasificación vemos que sucede lo mismo que en el caso del PCA2.

```
Matriz de Confusión PCA4:
[[  4  60]
 [  4 124]]
Reporte de Clasificación PCA4:
              precision    recall  f1-score   support

           0       0.50      0.06      0.11        64
           1       0.67      0.97      0.79       128

    accuracy                           0.67       192
   macro avg       0.59      0.52      0.45       192
weighted avg       0.62      0.67      0.57       192
```

Figura 11. Matriz de confusión y reporte de clasificación para PCA con n=4.

En el caso del PCA4 tenemos que tenemos 4 valores falsos negativos, aparentemente esta mejor equilibrada e implementada, más sin embargo el reporte de clasificación a pesar de estar más equilibrado en la precisión, los valores son bajos y en el caso del Recall y el f1 los valores positivos son excesivamente más altos que los negativos.

6. Analizar y contrastar resultados con gráficas de barras (pueden ser en Excel)

```
# ---------------------------------- PUNTO 6 Analizar y contrastar
resultados con gráficas de barras

# a) del uso de PCA con n=2,3,4

plt.subplot(2, 3, 1)
sns.countplot(pre2, data=new_instances)
plt.title('PCA2')
plt.xlabel('Clases')
plt.ylabel('Instancias')

plt.subplot(2, 3, 2)
sns.countplot(pre3, data=new_instances)
plt.title('PCA3')
plt.xlabel('Clases')
plt.ylabel('Instancias')

plt.subplot(2, 3, 3)
sns.countplot(pre4, data=new_instances)
plt.title('PCA4')
```

```
plt.xlabel('Clases')
plt.ylabel('Instancias')


# b) del mejor (n=2,3 o 4) y el uso de todas las características

plt.subplot(2, 3, 4)
sns.countplot(c3, data=new_instances)
plt.title('Regresión Logistica')
plt.xlabel('Clases')
plt.ylabel('Instancias')
```

En este caso se realizaron las gráficas de barras en el mismo programa donde comparamos los la regresión implementada a los PCA y la primera clasificación a todos los valores de prueba, como resultado nos da lo de la figura 12.

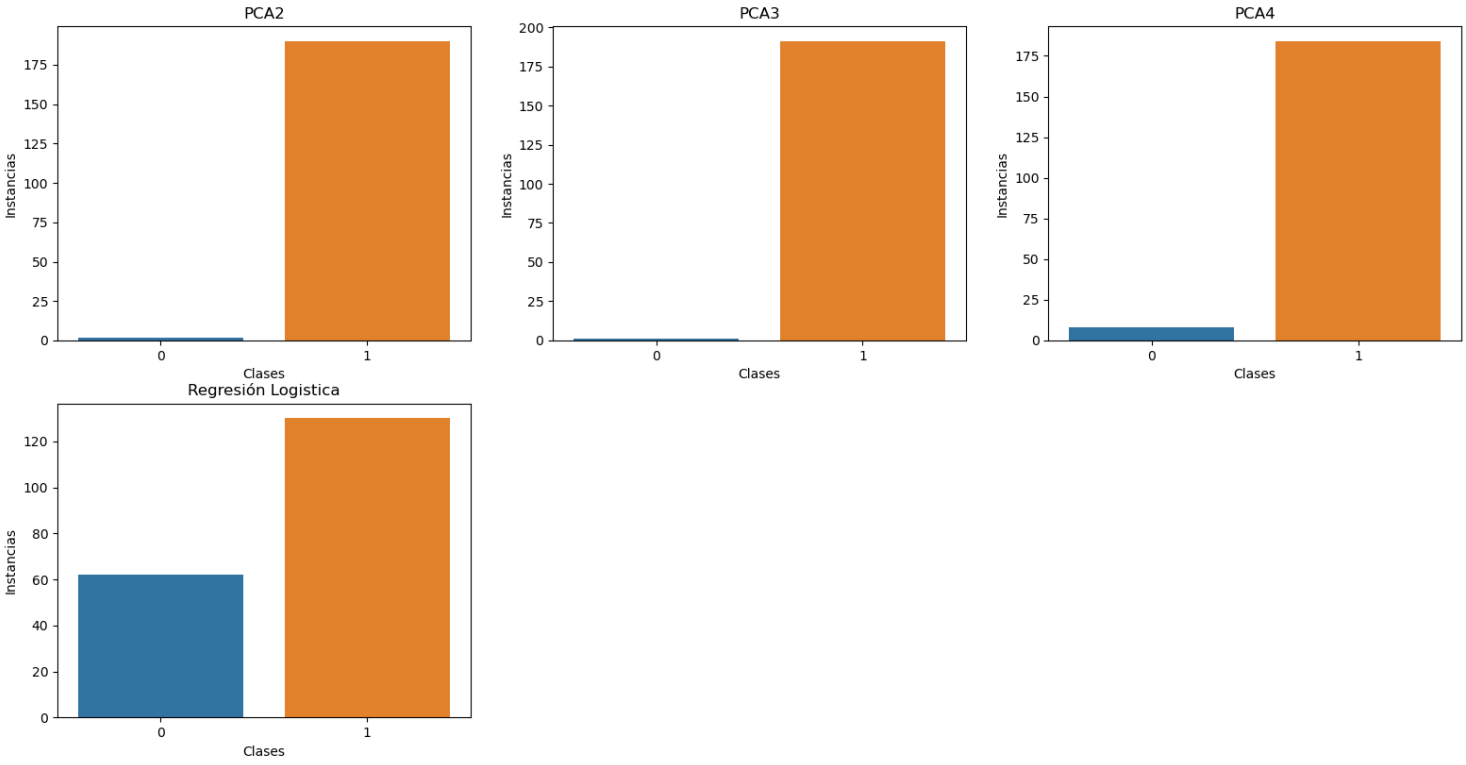

Figura 12. Grafica de resultados de la implementación PCA y Regresión logística.

Como podemos observar la implementación del PCA en este dataset no fue adecuada ya que como podemos observar la comparación con regresión logística, hubo mucha perdida de información, con la cual se pudo haber implementado para tener una clasificación más certera, e inclusive el PCA4 que fue quien obtuvo resultados un poco más adecuados es muy baja comparada con la implementación de todos los valores del dataset.

8. Aplicar k-means para generar 3 grupos y graficar. Tratar de interpretar los grupos

En este caso decidí implementar primero el k-means para poder trabajarlo con la gráfica del PCA2, donde nos retorna 3 valores, 0, 1 y 2. (Figura 13).

```
# ---------------------------------- PUNTO 8 Implementación de kmeans

kmeans = KMeans(n_clusters=3)
km = kmeans.fit_predict(x_train)
print('Kmeans: ', km)
```

```
Kmeans:  [2 1 0 1 1 2 1 0 1 0 0 1 2 2 0 2 0 0 2 1 2 2 0 0 1 2 2 1 2 2 1 0 0 0 0 2 0
 2 1 2 0 1 0 1 2 2 0 1 1 1 1 0 2 0 0 2 0 0 1 1 0 0 1 2 0 1 1 1 2 2 0 1 1 2
 1 2 0 0 2 0 2 0 1 1 0 1 2 2 0 2 1 1 0 0 0 1 1 0 2 2 2 2 1 1 0 0 2 0 1 1 2
 2 1 0 1 2 0 0 1 1 1 2 1 1 0 1 2 0 0 1 2 1 2 2 0 0 2 0 2 0 2 2 2 0 0 2 0 0
 2 2 1 0 1 0 2 2 0 1 2 2 2 1 1 0 1 2 0 0 1 1 0 1 2 1 0 0 0 0 0 0 1 2 0 1 2
 1 2 1 0 2 1 0 0 2 0 2 0 2 2 1 2 1 0 1 2 0 0 0 0 0 0 2 1 2 2 1 2 1 0 0 0 0
 2 1 1 0 2 2 0 0 1 0 0 2 1 2 2 0 2 0 1 0 2 1 1 0 0 1 0 2 2 0 0 2 0 2 2 1 2
 1 2 2 2 0 1 1 1 0 1 2 0 1 2 1 0 2 2 2 2 2 1 2 2 1 2 1 2 0 0 1 1 1 0 1 2 1
 2 1 2 1 1 0 2 0 2 2 2 2 0 2 0 0 2 0 0 2 0 0 2 2 0 2 0 2 2 2 2 1 2 1 2 2 2
 1 0 0 2 0 1 1 1 1 2 0 0 0 0 2 0 0 1 1 0 0 2 1 0 0 0 1 1 0 2 2 0 0 1 0 1 0
 2 2 2 2 0 1 0 0 0 0 1 0 1 0 2 2 1 1 0 2 0 2 0 1 1 1 0 2 2 2 0 0 1 1 1 2 0
 1 2 2 1 1 0 0 0 2 0 0 0 1 1 0 2 2 1 1 1 2 2 0 0 2 1 2 0 0 2 2 1 0 2 0 1 2
 1 2 0 1 0 1 1 1 2 1 2 2 0 0 0 2 2 0 0 2 2 2 0 0 1 2 1 1 0 0 2 2 2 2 0 0 0
 0 2 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 2 1 0 0 0 0 1 0 2 1 1 0 1 1 2 2 2 1 1 0
 2 1 1 0 2 0 1 2 0 0 2 1 1 2 2 1 1 0 2 0 1 1 2 1 1 1 2 1 1 1 1 1 2 0 2 1 1
 2 1 2 2 0 1 1 2 1 2 2 1 2 0 2 1 0 0 2 0 0 0 2 0 2 1 1 2 1 0 0 1 2 1 2 1 0
 1 1 1 2 2 0 2 0 1 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 0 0 1 0 2 0 2 0 2 0 1 2 2
 0 2 1 0 1 2 2 1 1 1 2 2 2 1 0 2 0 2 0 2 0 1 1 2 0 1 0 2 1 1 1 0 1 2 2 0 2
 2 0 2 2 1 1 1 2 2 0 2 1 1 2 2 1 1 2 2 2 1 2 0 2 1 1 1 0 2 2 1 1 0 2 0 1 0
 1 1 0 1 0 2 0 0 0 1 0 0 0 1 2 0 0 2 1 0 1 0 1 2 1 0 0 2 0 2 1 2 2 2 1 0 2
 2 2 0 1 2 0 0 2 2 2 0 2 1 1 1 1 1 1 1 0 0 0 1 0 1 1]
```

Figura 13. K-means resultados.

En este punto realizamos la implementación de un algoritmo no supervisado, el cual sus funciones son basadas al clustering o agrupamiento, trabajamos con solo los atributos sin etiquetar, al saber esto podemos deducir que su clasificación implementada en este dataset es base a sus atributos, que en este caso solo pueden tener 3 valores, X, O y b, con los cuales se realiza el agrupamiento y nos da una clasificación mediante estos.

7. Graficar resultados con PCA (n=2)

Una vez implementado el K-means observamos la combinación con el PCA2, la cual se muestra en la siguiente figura.

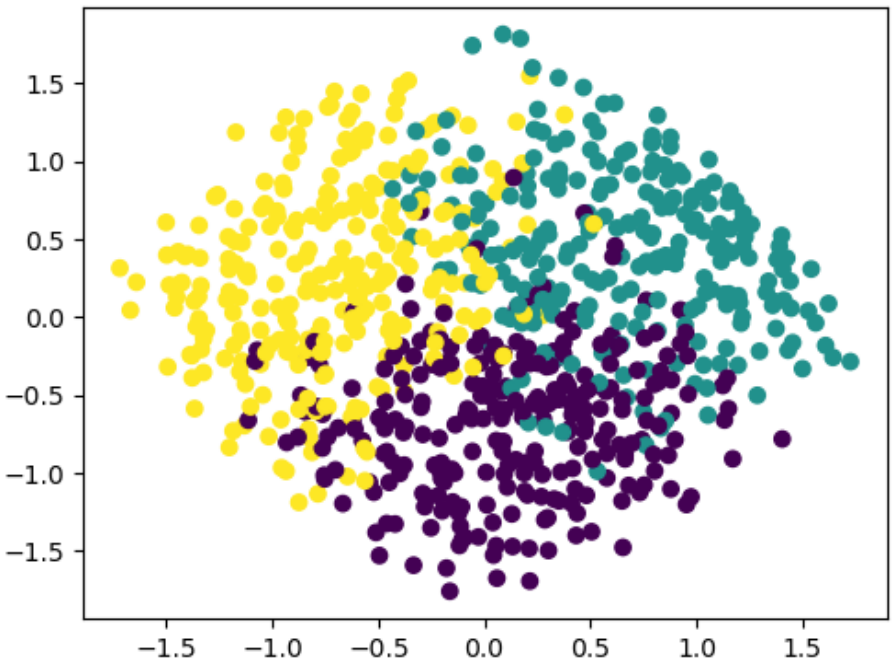

Figura 14. K-means y PCA 2 resultados.

Al observar los resultados, podemos deducir que la clasificación fue base a los atributos donde algunos pueden pertenecer a las victorias, a las derrotas e incluso empates.

9. Implementar el perceptrón simple de sklearn al conjunto de datos.

```
# ---------------------------------- PUNTO 9 Implementación del
perceptron simple al conjunto de datos

clf = Perceptron()
perc = clf.fit(x_train, y_train)
predicc = clf.predict(x_test)
predicc2 = clf.predict(x_train)
```

```
print('Puntaje del perceptron:
{:.2f}'.format(clf.score(x_train,y_train)))
print('Accuracy: ', accuracy_score(y_test, predicc))
```

El perceptrón simple consta de consta una red con una capa de salida de n neuronas y otra de salida con m neuronas. En este caso solo mostro el accuracy y el score implementado al conjunto de datos en el dataset. Dando unos resultados bastante buenos.

```
Puntaje del perceptron: 0.98
Accuracy:  0.984375
```

Figura 15. Implementación del perceptrón simple.

CONCLUSIONES

Mediante esta evaluación se mostró los conocimientos adquiridos en el segundo parcial y reforzando los obtenidos en el primer parcial.

La demostración de que cada problema tiene un algoritmo de Inteligencia Artificial adecuado para solucionarlo, ya que en este caso la implementación del PCA no fue el adecuado ya que los resultados brindados fueron demasiado bajos, en cambio con los algoritmos de regresión logística y el perceptrón fueron adecuados.

REFERENCIAS

1. Rueda, J. (2019). Aprendizaje supervisado y no supervisado - healthdataminer.com. From https://healthdataminer.com/data mining/aprendizaje-supervisado-y-no supervisado/#:~: text=El%20aprendizaje%20supervisado%20supone%20que, de%20datos%20no%20etiquetados%20previamente.

2. samples, U., & Higgins, O. (2017). UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples. From https://stackoverflow.com/questions/43162506/undefinedmetricwarning-f-score-is-ill-defined-and-being-set-to-0-0-in-labels-wi

3. sklearn.linear_model.Perceptron. (2022). From https://scikit learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html

4. Rueda, J. (2019). Aprendizaje supervisado y no supervisado - healthdataminer.com. From https://healthdataminer.com/data-mining/aprendizaje-supervisado-y-no-supervisado/#:~:text=El%20aprendizaje%20supervisado%20supone%20que,de%20datos%20no%20etiquetados%20previamente.

5. K-Means con Python paso a paso | Aprende Machine Learning. (2018). From https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/

6. PERCEPTR&#xFFFD;N SIMPLE. (2022). Retrieved 8 July 2022, from http://avellano.fis.usal.es/~lalonso/RNA/introMLP.htm