Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey

Escuela de ingeniería y ciencias

Inteligencia Artificial Avanzada para la Ciencia de Datos I (Grupo 101)

Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución. (Portafolio Implementación)

Alumnos:

Amy Murakami Tsutsumi A01750185

Profesor:
Iván Mauricio Amaya Contreras

Fecha:
09/09/2022

Librería

La librería principal utilizada fue sklearn.neural_network- MLPClassifier para poder implementar el algoritmo de redes neuronales.

Otras librerías que se utilizarán son:

- pandas : para la creación y operaciones de dataframes.
- numpy : para la creación de vectores y matrices.
- sklearn.preprocessing - StandardScaler: para el escalamiento de datos.
- sklearn.model_selection - train_test_split : para la división de los datos en subconjuntos de entrenamiento y prueba.
- sklearn.metrics - confusion_matrix & classification_report : para la visualización de desempeño del algoritmo y las métricas de clasificación.
- matplotlib.pyplot: para la generación de gráficos.
- seaborn : basada en matplotlib para la graficación de datos estadísticos.

Dataset

Para este portafolio de implementación se utilizará el dataset de "Heart Failure Prediction Dataset" que se obtuvo de la siguiente liga: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

Este dataset contiene información con atributos que se utilizarán para determinar y predecir si una persona es propensa a tener un ataque cardiaco. Estos atributos son:

1. Age: edad del paciente
2. Sex: sexo del paciente [M: Hombre, F: Mujer]
3. ChestPainType: tipo de dolor en el pecho [TA: Angina típica, ATA: Angina atípica, NAP: Dolor no anginoso, ASY: Asintomático]
4. RestingBP: presión arterial en reposo [mm Hg]
5. Cholesterol: colesterol sérico [mm/dl]
6. FastingBS: glucemia en ayunas [1: si FastingBS > 120 mg/dl, 0: en caso contrario]
7. RestingECG: resultados del electrocardiograma [Normal: Normal, ST: con anormalidad de la onda ST-T (inversiones de la onda T y/o elevación o depresión del ST de > 0,05 mV), HVI: que muestra hipertrofia ventricular izquierda probable o definitiva según los criterios de Estes].
8. MaxHR: frecuencia cardíaca máxima alcanzada [Valor numérico entre 60 y 202]
9. ExerciseAngina: angina inducida por el ejercicio [Y: Sí, N: No]

10. Oldpeak: oldpeak = ST [Valor numérico medido en depresión]
11. ST_Slope: la pendiente del segmento ST máximo del ejercicio [Up: pendiente ascendente, Flat: plano, Down: pendiente descendente]
12. HeartDisease: clase de salida [1: enfermedad cardiaca, 0: Normal]

Este dataset cuenta con 918 registros de pacientes.

Modelo
Para los modelos se dividió el dataset: el 75% lo conforman los datos de entrenamiento y el 25% son los datos de prueba.

```
[87] #Modularización del data-set
     x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size = 0.25, train_size = 0.75, random_state= 2)
```

Métrica de desempeño (valor logrado sobre el subset de prueba)

Se realizaron 5 modelos con distintos valores en las capas para poder encontrar el mejor modelo.

Modelos de predicción

Ahora se realizarán 5 modelos diferentes de redes neuronales para encontrar el más eficiente.

```
modelo1 = MLPClassifier(random_state = 1,
                        hidden_layer_sizes = (5),
                        activation = "relu",
                        verbose = False,
                        solver = "adam",
                        learning_rate = "adaptive",
                        max_iter = 10000)

modelo2 = MLPClassifier(random_state = 1,
                        hidden_layer_sizes = (10, 9),
                        activation = "relu",
                        verbose = False,
                        solver = "adam",
                        learning_rate = "adaptive",
                        max_iter = 10000)
```

```
modelo3 = MLPClassifier(random_state = 1,
                                    hidden_layer_sizes = (10, 8, 10),
                                    activation = "relu",
                                    verbose = False,
                                    solver = "adam",
                                    learning_rate = "adaptive",
                                    max_iter = 10000)

modelo4 = MLPClassifier(random_state = 1,
                                    hidden_layer_sizes = (10, 15, 10, 4),
                                    activation = "relu",
                                    verbose = False,
                                    solver = "adam",
                                    learning_rate = "adaptive",
                                    max_iter = 10000)

modelo5 = MLPClassifier(random_state = 1,
                                    hidden_layer_sizes = (10, 9, 9, 10),
                                    activation = "relu",
                                    verbose = False,
                                    solver = "adam",
                                    learning_rate = "adaptive",
                                    max_iter = 10000)
```

```
modelo1.fit(x_train, y_train)
modelo2.fit(x_train, y_train)
modelo3.fit(x_train, y_train)
modelo4.fit(x_train, y_train)
modelo5.fit(x_train, y_train)

print("Training score modelo 1: ", modelo1.score(x_train,y_train))
print("Test score modelo 1: ", modelo1.score(x_test, y_test))
print("\nTraining score modelo 2: ", modelo2.score(x_train,y_train))
print("Test score modelo 2: ", modelo2.score(x_test, y_test))
print("\nTraining score modelo 3: ", modelo3.score(x_train,y_train))
print("Test score modelo 3: ", modelo3.score(x_test, y_test))
print("\nTraining score modelo 4: ", modelo4.score(x_train,y_train))
print("Test score modelo 4: ", modelo4.score(x_test, y_test))
print("\nTraining score modelo 5: ", modelo5.score(x_train,y_train))
print("Test score modelo 5: ", modelo5.score(x_test, y_test))
```

Podemos visualizar que los modelos 1, 3 y 4 son los más precisos con los datos de entrenamiento. Sin embargo, el modelo más preciso con los datos de prueba es el modelo 4.

```
Training score modelo 1:  0.8662790697674418
Test score modelo 1:  0.8130434782608695

Training score modelo 2:  0.8619186046511628
Test score modelo 2:  0.8391304347826087

Training score modelo 3:  0.875
Test score modelo 3:  0.8347826086956521

Training score modelo 4:  0.8662790697674418
Test score modelo 4:  0.8521739130434782

Training score modelo 5:  0.8648255813953488
Test score modelo 5:  0.8304347826086956
```

Predicciones de prueba (entradas, valor esperado, valor obtenido)

Más adelante se realizaron las predicciones de los cinco modelos creados.

Predicciones

Ahora se realizarán las predicciones de los cinco modelos.

```
[66] pred1 = modelo1.predict(x_test)
     pred2 = modelo2.predict(x_test)
     pred3 = modelo3.predict(x_test)
     pred4 = modelo4.predict(x_test)
     pred5 = modelo5.predict(x_test)
```

```
#Modelo 1
print('Modelo 1: ')
print(modelo1.predict_proba([df_x.loc[0]]))
print('Heart Disease? Estimated: ', modelo1.predict([df_x.loc[0]]),
      'Real: ', df_y.loc[0])
#Modelo 2
print('\nModelo 2: ')
print(modelo2.predict_proba([df_x.loc[0]]))
print('Heart Disease? Estimated: ', modelo2.predict([df_x.loc[0]]),
      'Real: ', df_y.loc[0])
#Modelo 3
print('\nModelo 3: ')
print(modelo3.predict_proba([df_x.loc[0]]))
print('Heart Disease? Estimated: ', modelo3.predict([df_x.loc[0]]),
      'Real: ', df_y.loc[0])
#Modelo 4
print('\nModelo 4: ')
print(modelo4.predict_proba([df_x.loc[0]]))
print('Heart Disease? Estimated: ', modelo4.predict([df_x.loc[0]]),
      'Real: ', df_y.loc[0])
#Modelo 5
print('\nModelo 5: ')
print(modelo5.predict_proba([df_x.loc[0]]))
print('Heart Disease? Estimated: ', modelo5.predict([df_x.loc[0]]),
      'Real: ', df_y.loc[0])
```

```
Modelo 1:
[[0.96209784 0.03790216]]
Heart Disease? Estimated:  [0] Real:  0

Modelo 2:
[[0.9689947 0.0310053]]
Heart Disease? Estimated:  [0] Real:  0

Modelo 3:
[[0.96374434 0.03625566]]
Heart Disease? Estimated:  [0] Real:  0

Modelo 4:
[[0.94906654 0.05093346]]
Heart Disease? Estimated:  [0] Real:  0

Modelo 5:
[[0.96986661 0.03013339]]
Heart Disease? Estimated:  [0] Real:  0
```

Podemos observar que en todos los modelos se está realizando la predicción de manera correcta. Como primer valor muestra la probabilidad de que el HeartDisease tenga un valor de 0 y 1. Después se muestra el valor estimado con los modelos y el valor real. En todos los modelos coinciden que el valor estimado por cada modelo y

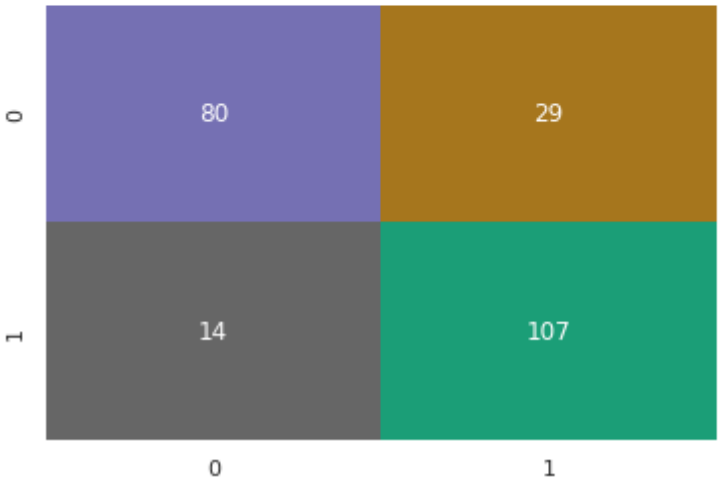
el valor real es 0, por lo que se puede confirmar que son modelos efectivos para la predicción de enfermedades cardiacas.

Validación

Para la etapa de validación se utilizarán matrices de confusión y reportes de clasificación.

```
#Modelo 1
print('Modelo 1: ')
sns.set()
f, ax = plt.subplots()
matriz1 = confusion_matrix(y_test,pred1)
print(classification_report(y_test,pred1))
print('\nMatriz de confusión: ')
sns.heatmap(matriz1, annot=True, ax=ax, cbar=False, fmt='g', cmap='Dark2_r')
```

```
Modelo 1:
              precision    recall  f1-score   support

           0       0.85      0.73      0.79       109
           1       0.79      0.88      0.83       121

    accuracy                           0.81       230
   macro avg       0.82      0.81      0.81       230
weighted avg       0.82      0.81      0.81       230


Matriz de confusión:
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1a3b3ce50>
```



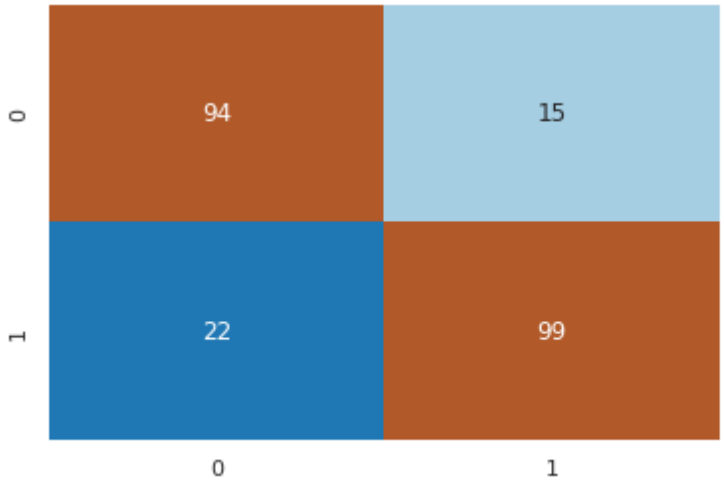
}

Para el primer modelo podemos observar que el valor de f1-score es 0.81 lo que indica que es un buen modelo de predicción. Por otro lado, en la matriz de confusión se tienen 80 valores true positive y 107 valores true negative; es decir, valores

predecidos que coinciden con el valor real. Además, se tienen 29 valores false positive y 14 false negative es decir valores erróneos.

```
#Modelo 2
print('Modelo 2: ')
sns.set()
f, ax = plt.subplots()
matriz2 = confusion_matrix(y_test,pred2)
print(classification_report(y_test,pred2))
print('\nMatriz de confusión: ')
sns.heatmap(matriz2, annot=True, ax=ax, cbar=False, fmt='g', cmap='Paired')
```

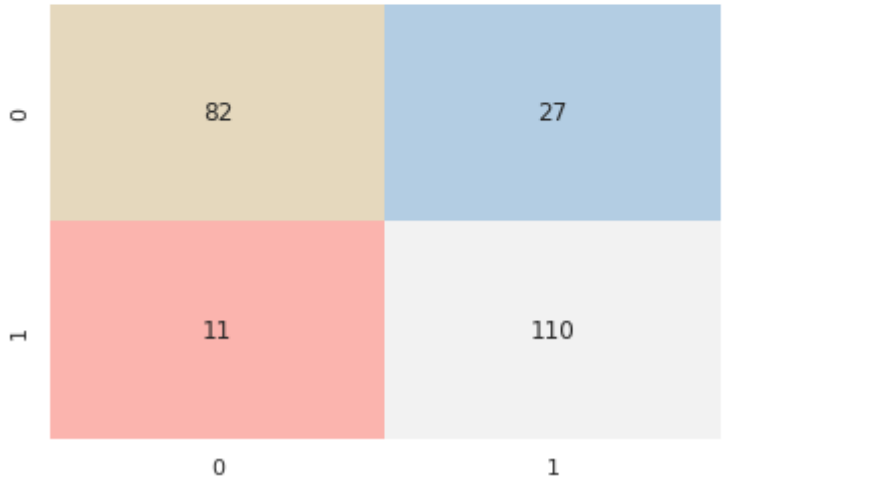
```
Modelo 2:
              precision    recall  f1-score   support

           0       0.81      0.86      0.84       109
           1       0.87      0.82      0.84       121

    accuracy                           0.84       230
   macro avg       0.84      0.84      0.84       230
weighted avg       0.84      0.84      0.84       230


Matriz de confusión:
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1a3ae4c90>
```



Para el segundo modelo podemos observar que el valor de f1-score es 0.84 lo que indica que es un mejor modelo de predicción que el primero. Por otro lado, en la matriz de confusión se tienen 94 valores true positive y 99 valores true negative; es decir, valores predecidos que coinciden con el valor real. Además, se tienen 15 valores false positive y 22 false negative es decir valores erróneos.

```
#Modelo 3
print('Modelo 3: ')
sns.set()
f, ax = plt.subplots()
matriz3 = confusion_matrix(y_test,pred3)
print(classification_report(y_test,pred3))
print('\nMatriz de confusión: ')
sns.heatmap(matriz3, annot=True, ax=ax, cbar=False, fmt='g', cmap='Pastel1')
```

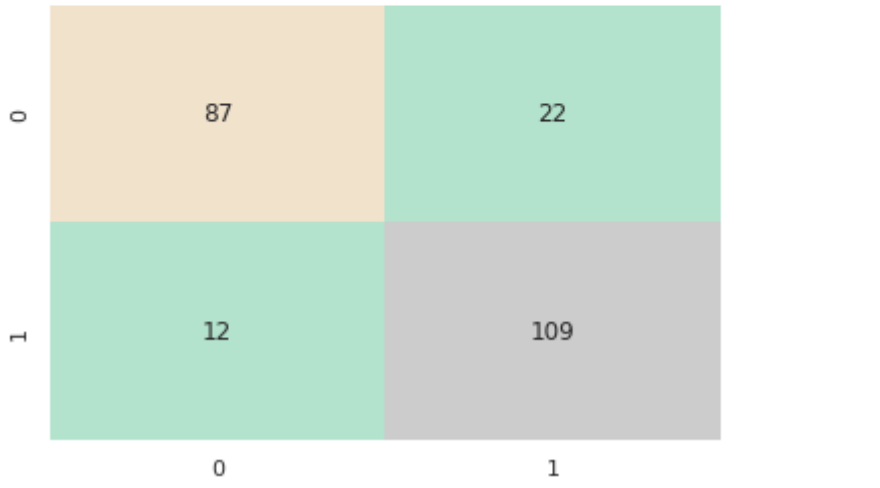
```
Modelo 3:
              precision    recall  f1-score   support

           0       0.88      0.75      0.81       109
           1       0.80      0.91      0.85       121

    accuracy                           0.83       230
   macro avg       0.84      0.83      0.83       230
weighted avg       0.84      0.83      0.83       230


Matriz de confusión:
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1a3a7b510>
```



Para el tercer modelo podemos observar que el valor de f1-score es 0.83 lo que indica que es un buen modelo, pero no tiene la misma precisión que el segundo. Por otro lado, en la matriz de confusión se tienen 82 valores true positive y 110 valores true negative; es decir, valores predecidos que coinciden con el valor real. Además, se tienen 27 valores false positive y 11 false negative es decir valores erróneos.

```python
#Modelo 4
print('Modelo 4: ')
sns.set()
f, ax = plt.subplots()
matriz4 = confusion_matrix(y_test,pred4)
print(classification_report(y_test,pred4))
print('\nMatriz de confusión: ')
sns.heatmap(matriz4, annot=True, ax=ax, cbar=False, fmt='g', cmap='Pastel2')
```

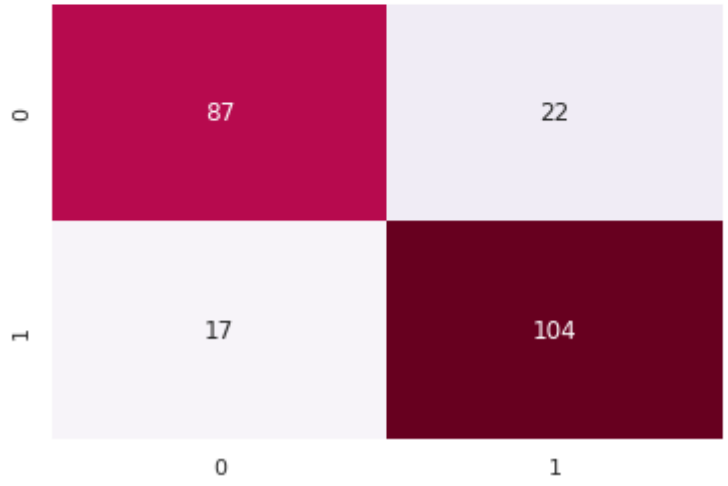
```
Modelo 4:
              precision    recall  f1-score   support

           0       0.88      0.80      0.84       109
           1       0.83      0.90      0.87       121

    accuracy                           0.85       230
   macro avg       0.86      0.85      0.85       230
weighted avg       0.85      0.85      0.85       230


Matriz de confusión:
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1a3a3dd90>
```



Para el cuarto modelo podemos observar que el valor de f1-score es 0.85 lo que indica que es el mejor modelo hasta ahora. Por otro lado, en la matriz de confusión se tienen 87 valores true positive y 109 valores true negative; es decir, valores predecidos que coinciden con el valor real. Además, se tienen 22 valores false positive y 12 false negative es decir valores erróneos.

```
#Modelo 5
print('Modelo 5: ')
sns.set()
f, ax = plt.subplots()
matriz5 = confusion_matrix(y_test,pred5)
print(classification_report(y_test,pred5))
print('\nMatriz de confusión: ')
sns.heatmap(matriz5, annot=True, ax=ax, cbar=False, fmt='g', cmap='PuRd')
```

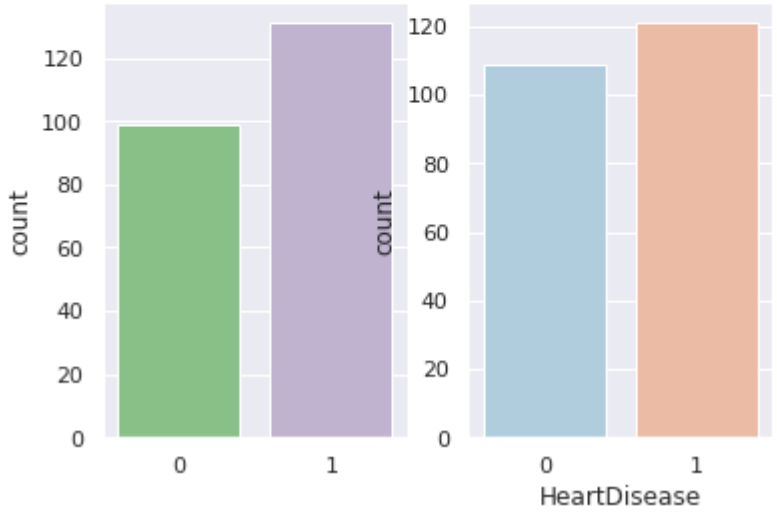
```
Modelo 5:
              precision    recall  f1-score   support

           0       0.84      0.80      0.82       109
           1       0.83      0.86      0.84       121

    accuracy                           0.83       230
   macro avg       0.83      0.83      0.83       230
weighted avg       0.83      0.83      0.83       230


Matriz de confusión:
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1aa809f50>
```



Por último, para el cuarto modelo podemos observar que el valor de f1-score es 0.83 lo que indica que es un buen modelo, pero no el mejor. Por otro lado, en la matriz de confusión se tienen 87 valores true positive y 104 valores true negative; es decir, valores predecidos que coinciden con el valor real. Además, se tienen 22 valores false positive y 17 false negative es decir valores erróneos.

```
fig, ax =plt.subplots(1,2)
sns.countplot(pred4, ax=ax[0], palette="Accent")
sns.countplot(y_test, ax=ax[1], palette="RdBu_r")
fig.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.
  FutureWarning
```



En la gráfica anterior se muestra del lado izquierdo los número de valores predecidos con el modelo cuatro y del lado derecho los valores reales. Se puede observar que los datos predecidos se acercan bastante a los valores reales.

Nombre del archivo a revisar

MomRetroM2Framework.py
MomentoRetroM2Framework.pdf