TRABAJO PRÁCTICO NRO1

Aprendizaje automático 2

FECHA DE ENTREGA: 26 DE SEPTIEMBRE DE 2024

PROFESOR: ING. JORGE CEFERINO VALDEZ

Autor: Fernanda Cader

Aplicaciones de las Redes Neuronales en el Mundo Real

1. **Reconocimiento de Imágenes**: Utilizadas en aplicaciones de visión por computadora, como la identificación de objetos en imágenes (por ejemplo, reconocimiento facial).
2. **Procesamiento del Lenguaje Natural (NLP)**: Empleadas en traducción automática, chatbots y análisis de sentimientos.
3. **Diagnóstico Médico**: Ayudan en la detección de enfermedades a partir de imágenes médicas o datos de pacientes.
4. **Finanzas**: Utilizadas para detectar fraudes en transacciones, análisis de riesgos y predicción de precios en mercados.
5. **Automóviles Autónomos**: Integradas en sistemas de percepción y toma de decisiones para vehículos sin conductor.
6. **Recomendaciones Personalizadas**: Usadas en plataformas de streaming y comercio electrónico para sugerir productos o contenidos a los usuarios.

Funciones de Activación

1. **ReLU (Rectified Linear Unit)**:

   Esta función es muy popular en capas ocultas de redes neuronales porque permite que la red aprenda de manera eficiente y evita problemas de desvanecimiento del gradiente.

   Uso: Principalmente en redes profundas para clasificación y regresión, pero no se utiliza en la capa de salida para problemas de clasificación multiclase.

2. **tanh (tangente hiperbólica)**:

   Uso: A menudo se utiliza en capas ocultas. Aunque es mejor que la sigmoide en cuanto a la normalización de los datos, todavía puede sufrir el problema de desvanecimiento del gradiente.

3. **Sigmoide (Logistic)**:

   Su rango es de 0 a 1.

   Uso: Comúnmente se utiliza en la capa de salida para problemas de clasificación binaria.

Para Clasificación Multiclase

- **Softmax** es la función de activación más adecuada para la capa de salida en problemas de clasificación multiclase, ya que normaliza las salidas en un rango de probabilidades que suman 1.

Resumen

- **ReLU** y **tanh** son funciones útiles para capas ocultas, mientras que **sigmoide** es apropiada para clasificación binaria. Para clasificación multiclase, se utiliza **softmax** en la capa de salida. La elección de la función de activación depende del tipo de problema y de la arquitectura de la red neuronal.

Informe sobre el Rendimiento de Modelos MLPClassifier

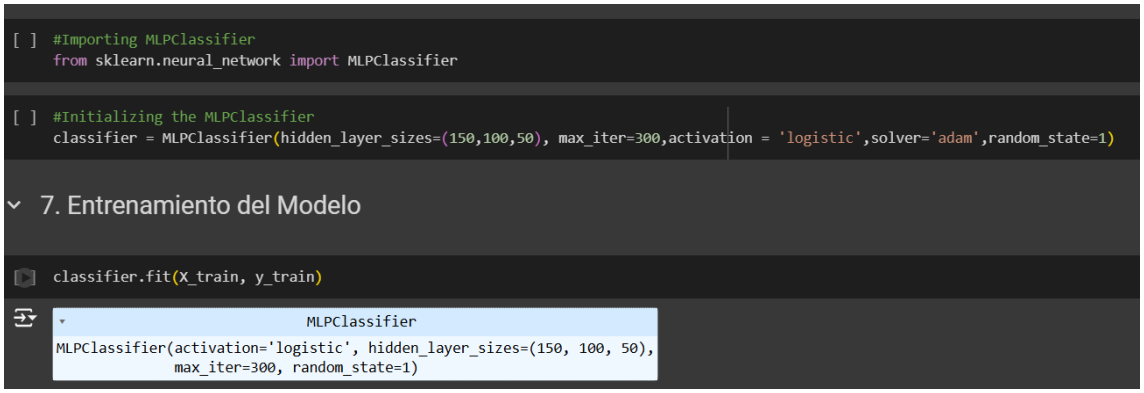
Modificaciones Realizadas

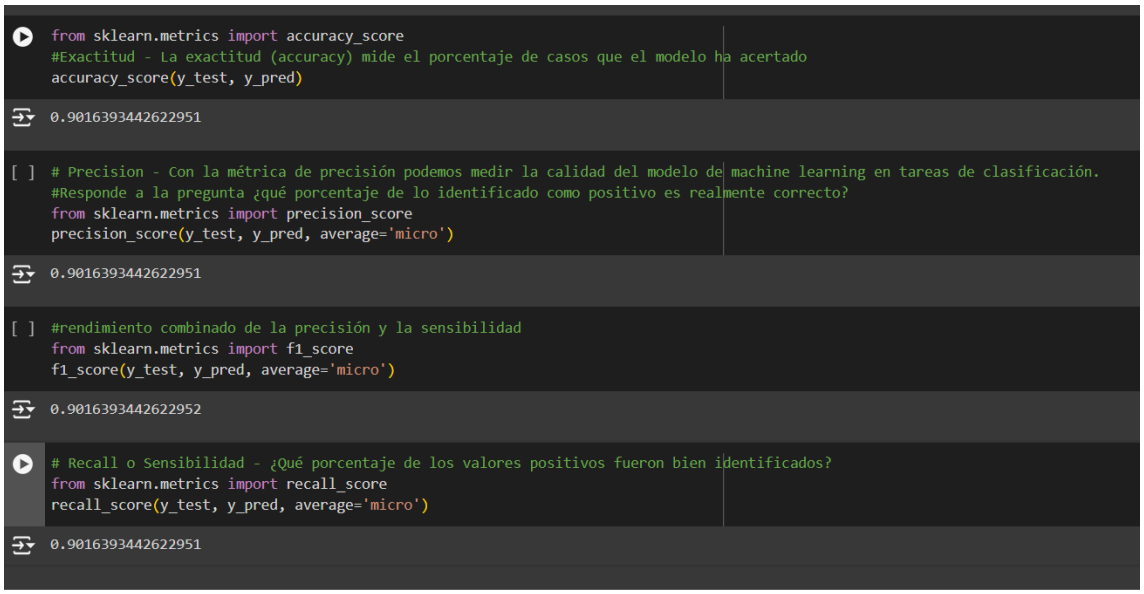
Se realizaron cambios en la configuración del modelo MLPClassifier de la siguiente manera:

1. **Estructura de Capas Ocultas**:
    - **Modelo Original**: hidden_layer_sizes=(150, 100, 50) con función de activación logistic.
    - **Modelo Modificado**: hidden_layer_sizes=(50, 30, 10) con función de activación relu.
2. **Número de Iteraciones**:
    - El número máximo de iteraciones se redujo de **300** a **100** en el modelo modificado.

Resultados y Métricas de Evaluación

Métrica	Modelo Original	Modelo Modificado
Exactitud	0.9016	0.8525
Precisión	0.9016	0.8525
F1 Score	0.9016	0.8525
Recall	0.9016	0.8525





```
from sklearn.metrics import accuracy_score
#Exactitud - La exactitud (accuracy) mide el porcentaje de casos que el modelo ha acertado
accuracy_score(y_test, y_pred)
```

```
0.9016393442622951
```

```
# Precision - Con la métrica de precisión podemos medir la calidad del modelo de machine learning en tareas de clasificación.
#Responde a la pregunta ¿qué porcentaje de lo identificado como positivo es realmente correcto?
from sklearn.metrics import precision_score
precision_score(y_test, y_pred, average='micro')
```

```
0.9016393442622951
```

```
#rendimiento combinado de la precisión y la sensibilidad
from sklearn.metrics import f1_score
f1_score(y_test, y_pred, average='micro')
```

```
0.9016393442622952
```

```
# Recall o Sensibilidad - ¿Qué porcentaje de los valores positivos fueron bien identificados?
from sklearn.metrics import recall_score
recall_score(y_test, y_pred, average='micro')
```

```
0.9016393442622951
```

```
[36] #Importing MLPClassifier
     from sklearn.neural_network import MLPClassifier
```

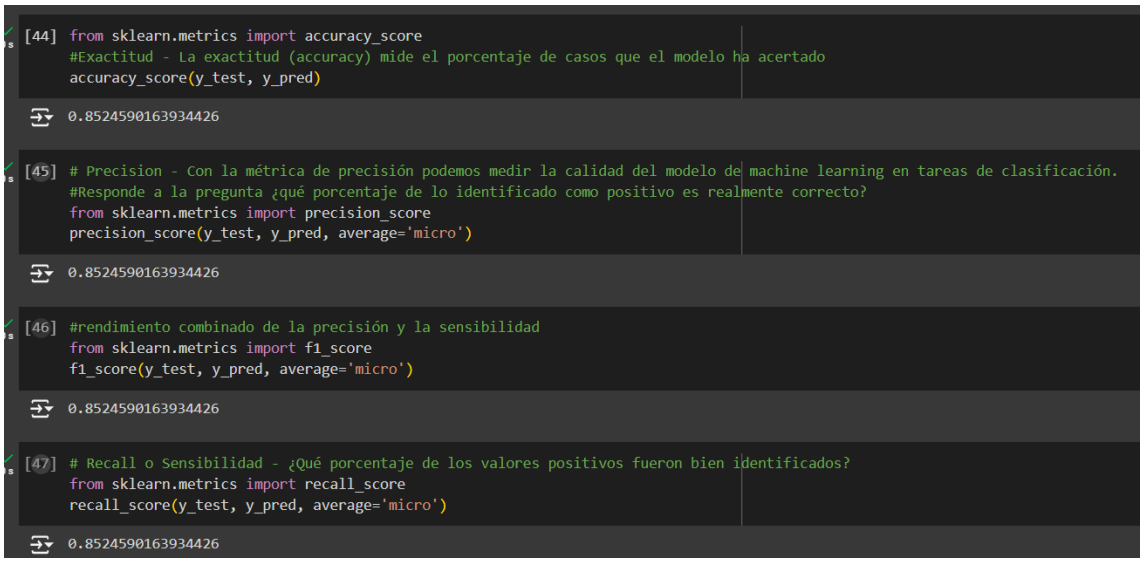
```
[37] #Initializing the MLPClassifier
     classifier = MLPClassifier(hidden_layer_sizes=(50,30,10), max_iter=100,activation = 'relu',solver='adam',random_state=1)
```

7. Entrenamiento del Modelo

+ Código + Texto

```
[38] classifier.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:690: ConvergenceWarning: Stochastic Optimizer: Maximum
  warnings.warn(
MLPClassifier
MLPClassifier(hidden_layer_sizes=(50, 30, 10), max_iter=100, random_state=1)
```



```
[44] from sklearn.metrics import accuracy_score
     #Exactitud - La exactitud (accuracy) mide el porcentaje de casos que el modelo ha acertado
     accuracy_score(y_test, y_pred)
```

```
0.8524590163934426
```

```
[45] # Precision - Con la métrica de precisión podemos medir la calidad del modelo de machine learning en tareas de clasificación.
     #Responde a la pregunta ¿qué porcentaje de lo identificado como positivo es realmente correcto?
     from sklearn.metrics import precision_score
     precision_score(y_test, y_pred, average='micro')
```

```
0.8524590163934426
```

```
[46] #rendimiento combinado de la precisión y la sensibilidad
     from sklearn.metrics import f1_score
     f1_score(y_test, y_pred, average='micro')
```

```
0.8524590163934426
```

```
[47] # Recall o Sensibilidad - ¿Qué porcentaje de los valores positivos fueron bien identificados?
     from sklearn.metrics import recall_score
     recall_score(y_test, y_pred, average='micro')
```

```
0.8524590163934426
```

Análisis de Resultados

- **Modelo Original**:
    - Mostró un rendimiento alto con métricas consistentes (exactitud, precisión, F1 score y recall) alrededor del 90.16%. Esto sugiere que el

modelo es efectivo para clasificar correctamente los casos de enfermedad cardíaca.

- **Modelo Modificado**:
    - Se observó una disminución notable en todas las métricas, con resultados alrededor del 85.25%. Esto indica que el modelo tiene un menor rendimiento en la clasificación y puede estar clasificando incorrectamente más casos de enfermedad cardíaca.

Conclusiones sobre el Impacto de los Cambios

1. **Impacto de la Estructura de Capas**:
    - Reducir el tamaño de las capas ocultas (de 150, 100, 50 a 50, 30, 10) probablemente limitó la capacidad del modelo para aprender patrones complejos en los datos, resultando en un rendimiento inferior.
2. **Función de Activación**:
    - Cambiar de la función de activación logistic a relu podría haber contribuido a un mejor aprendizaje en el modelo original, dado que relu permite una convergencia más rápida en redes profundas. Sin embargo, el cambio de arquitectura podría haber contrarrestado esta ventaja.
3. **Número de Iteraciones**:
    - La reducción de iteraciones de 300 a 100 limitó el tiempo de entrenamiento del modelo modificado, lo que puede haber impedido que el modelo alcanzara un mejor ajuste a los datos.

En resumen, los cambios realizados llevaron a un deterioro significativo en el rendimiento del modelo. Para futuras optimizaciones, sería recomendable ajustar los hiperparámetros, explorar diferentes configuraciones de capas y aumentar el número de iteraciones, manteniendo una función de activación adecuada que maximice el aprendizaje.