EVALUACION: UNIDAD II

Apellidos y Nombres: Paricela Yana Jimena Yessica

1. **¿Cuáles son las principales diferencias entre la optimización de funciones convexas y no convexas, y cómo influyen en la elección de métodos?**

   Rpta:
   Las funciones convexas garantizan un único mínimo global, lo que permite usar métodos como el descenso de gradiente con alta eficiencia. En cambio, las funciones no convexas pueden tener múltiples mínimos locales, lo que complica la optimización. Por eso, se requieren métodos más robustos, como algoritmos evolutivos o técnicas con reinicio. La forma de la función determina directamente la elección del método.

2. **¿Cuándo resulta más efectivo usar variantes del descenso de gradiente y cuáles son los factores clave en su selección?**

   Rpta:
   Las variantes del descenso de gradiente son útiles cuando el descenso clásico no logra una convergencia estable o rápida, especialmente en problemas con datos ruidosos o funciones no convexas. Usar algoritmos como Adam o Adagrad permite ajustar automáticamente la tasa de aprendizaje y mejora el rendimiento en tareas de deep learning. La elección depende del comportamiento del modelo, la complejidad del problema y la sensibilidad del entrenamiento.

3. **¿Cómo se aplican los criterios de optimalidad en un método de optimización?**

   Rpta:
   Los criterios de optimalidad permiten verificar si una solución encontrada realmente es válida dentro del problema. En la práctica, se usan para medir si el gradiente es suficientemente pequeño o si se cumple una condición de parada. Esto no solo garantiza que el algoritmo no siga iterando sin sentido, sino que también ayuda a controlar la calidad del resultado obtenido.

4. **¿De qué manera facilitan las herramientas de optimización automática (p.ej., Optuna) la búsqueda de configuraciones optimas en aprendizaje automático?**

   Rpta:
   Optuna es muy útil porque simplifica un proceso que normalmente es lento y manual. Al automatizar la búsqueda de hiperparámetros, permite enfocarse más en el diseño del modelo y menos en el ajuste. Su capacidad para explorar de forma inteligente el espacio de búsqueda mejora los resultados sin necesidad de intervención constante, lo cual es ideal en proyectos complejos de aprendizaje automático.

5. **¿Cómo influye la selección de estrategias de regularización en el desempeño y la generalización de un modelo?**

   Rpta:
   La regularización es clave para lograr un equilibrio entre precisión y generalización. Al penalizar la complejidad del modelo, se evitan predicciones demasiado ajustadas a los datos de entrenamiento. Elegir bien la técnica de regularización depende del

tipo de modelo y del problema, pero siempre tiene un impacto directo en su desempeño real frente a datos no vistos.

6. **¿Cómo se pueden integrar rutinas de validación y corrección de código en Optuna para mejorar la calidad y eficacia del proceso de optimización de hiperparámetros?**

   Rpta:
   Al añadir validación y control de errores en Optuna es fundamental para obtener resultados útiles. Esto permite detectar combinaciones de hiperparámetros que fallan o no generalizan bien, evitando perder tiempo en pruebas ineficaces. Además, mejora la eficiencia del proceso, ya que se filtran automáticamente las configuraciones no viables antes de continuar con la optimización.

7. **¿Cómo podemos utilizar las métricas provistas por scikit-learn para evaluar un modelo de clasificación, y qué información adicional brindan además de la exactitud? Completa y Responde:**

```
from ........

y_true = [0, 1, 2, 2, 2]
y_pred = [0, 2, 2, 2, 1]

print("Accuracy:", accuracy_score(y_true, y_pred))
print("Precision:", precision_score(y_true, y_pred, average='macro'))
print("Recall:", recall_score(y_true, y_pred, average='macro'))
```

   Rpta:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score

y_true = [0, 1, 2, 2, 2]
y_pred = [0, 2, 2, 2, 1]

print("Accuracy:", accuracy_score(y_true, y_pred))
print("Precision:", precision_score(y_true, y_pred, average='macro'))
print("Recall:", recall_score(y_true, y_pred, average='macro'))
```

   SALIDA:
   Accuracy: 0.4
   Precision: 0.38888888888888884
   Recall: 0.4444444444444444

   - **accuracy_score** muestra que el 40% de las predicciones fueron correctas.
   - **precision_score (macro)** indica el promedio de la precisión en cada clase sin considerar su frecuencia, es decir, cuán precisas son las predicciones por clase.
   - **recall_score (macro)** refleja la capacidad del modelo para recuperar correctamente los verdaderos positivos por clase.

   Estas métricas son fundamentales cuando se trabaja con clases desbalanceadas. Mientras el accuracy puede ser engañoso si una clase domina, la precisión y el recall permiten ver cómo el modelo se comporta en cada clase individual, brindando una evaluación más justa y completa.