Práctica Calificada 3

Machine Learning para CCSS y Gestión Pública

21 de noviembre de 2025

Instrucciones Generales

- **Grupos:** El trabajo será presentado por grupos de 4 integrantes.
- **Entregable:** Un único Jupyter Notebook (.ipynb), que contenga todo el código, los resultados (gráficos y salidas de celdas) y las respuestas escritas a las preguntas de interpretación.
- **Puntaje Total:** 20 puntos. **Se recomienda desarrollar el ejercicio en Colab y deepues descargar en formato Jupyter Notebook**
- **Fecha límite:** La fecha límite para la entrega es el **jueves 28 de noviembre hasta las 11:59 pm**. Se debe subir su solución a la actividad correspondiente creada en la plataforma de Paideia.
- **Cualquier duda:** Por favor, no duden en escribirme por correo a zdelacruzl@pucp.edu.pe.

Práctica Calificada 3: Aplicaciones de Redes Neuronales (20 puntos)

En este ejercicio, diseñarán, implementarán y evaluarán tres arquitecturas de redes neuronales para resolver problemas en diferentes dominios. Se espera que justifiquen sus decisiones de diseño y analicen críticamente los resultados.

Parte 1: Red Neuronal para Predicción de Precios de Viviendas en California (6 puntos)

Utilizaremos el dataset **California Housing**. Este conjunto de datos contiene información del censo de 1990.

0.1.1. Preparación de Datos (2 puntos)

a) **Carga:** Importe la función necesaria y cargue los datos:

```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
housing = fetch_california_housing()
X, y = housing.data, housing.target
```

b) **División:** Divida los datos en conjuntos de entrenamiento (75 %) y prueba (25 %) utilizando `train_test_split` y un `random_state` para reproducibilidad.

c) **Escalamiento:** Las redes neuronales son sensibles a la escala de los datos .

- Utilice `StandardScaler` de Scikit-Learn.
- Ajuste el escalador (`.fit`) **solo** con los datos de entrenamiento.
- Transforme (`.transform`) tanto los datos de entrenamiento como los de prueba.

0.1.2. Implementación y Comparación de Modelos (4 puntos)

a) **Modelo Base (Regresión Lineal):** Entrene una Regresión Lineal con los datos de entrenamiento escalados. Evalúela en el conjunto de prueba calculando el **RMSE** (Raíz del Error Cuadrático Medio).

b) **Red Neuronal (MLP):** Diseñe una arquitectura capaz de capturar no-linealidades. Se sugiere:

- Capa de entrada adecuada a la dimensión de los datos.
- **Dos capas ocultas** con función de activación `relu`.

c) **Entrenamiento:** Compile el modelo usando `loss='mse'` y un optimizador como `Adam`. Entrene por al menos 20-30 épocas. Use un `validation_split` del 20 % para monitorear que no haya sobreajuste.

d) **Evaluación:** Calcule el RMSE de la Red Neuronal en el conjunto de prueba y compárelo con el de la Regresión Lineal.

e) **Preguntas de Interpretación:**

ı) Reporte los RMSE de ambos modelos. ¿Hubo una mejora significativa al usar la Red Neuronal?

Parte 2: Red Neuronal Convolucional para Clasificación de Moda (7 puntos)

Utilizarán el dataset Fashion-MNIST, que contiene imágenes en escala de grises de 10 categorías de ropa. El desafío es construir una CNN para clasificar correctamente estas imágenes.

0.2.1. Carga y Preparación de Datos (2 puntos)

a) Cargue el dataset Fashion-MNIST directamente desde Keras: `from tensorflow.keras.datasets import fashion_mnist`.

b) Preprocese las imágenes: normalice los valores de los pixeles y asegúrese de que tengan la forma correcta para ser procesadas por una capa convolucional.

c) Convierta las etiquetas de clase a un formato categórico (one-hot encoding).

0.2.2. Diseño y Evaluación de la CNN (5 puntos)

a) Diseñe e implemente una Red Neuronal Convolucional (CNN). Su modelo debe incluir al menos dos capas convolucionales y al menos una capa de pooling.

b) Compile y entrene su modelo. Monitoree el rendimiento tanto en el conjunto de entrenamiento como en un conjunto de validación durante el entrenamiento.

c) Evalúe el modelo final sobre el **conjunto de prueba**. Reporte el *accuracy* global.

d) **Pregunta de Interpretación:** Explique brevemente el rol de las capas convolucionales y de pooling en su arquitectura. ¿Hubo alguna categoría de ropa con la que su modelo tuvo más dificultades? ¿A qué cree que se deba? (Puede usar la matriz de confusión para responder).

Parte 3: Red Neuronal Recurrente (LSTM) para Clasificación de Noticias (7 puntos)

En esta sección, implementarán una red recurrente para clasificar noticias cortas en 46 temas diferentes, utilizando el dataset Reuters.

0.3.1. Carga y Preparación de Datos (3 puntos)

a) Cargue el dataset Reuters directamente desde Keras: `from tensorflow.keras.datasets import reuters`. Limite el vocabulario a las 10,000 palabras más frecuentes (`num_words=10000`).

b) Las secuencias de texto (noticias) tienen diferentes longitudes. Estandarice su longitud usando `pad_sequences` para que todas tengan un tamaño uniforme. Elija una longitud máxima que capture la mayor parte de la información sin ser excesiva.

c) Dado que este es un problema de clasificación multi-clase, convierta las etiquetas de los temas (enteros) a un formato categórico (one-hot encoding) usando `to_categorical`.

0.3.2. Diseño y Evaluación de la LSTM (4 puntos)

a) Diseñe y construya un modelo secuencial basado en capas recurrentes para clasificar las noticias. Su arquitectura debe incluir una capa `Embedding` al inicio.

b) Compile su modelo. Asegúrese de seleccionar una función de pérdida y una función de activación en la capa de salida que sean apropiadas para una clasificación multi-clase.

c) Entrene el modelo con sus datos de entrenamiento, utilizando una porción de estos para validación (`validation_split`).

d) Evalúe el rendimiento del modelo final sobre el **conjunto de prueba** y reporte el *accuracy*.