Reporte Técnico - Módulo

1: Predicción de Demanda de Transporte (Series de Tiempo)

1.1 Descripción del problema

El Módulo 1 tiene como objetivo pronosticar la demanda de viajes por destino turístico utilizando series temporales, lo que permite optimizar la planificación de recursos de transporte (por ejemplo, asignación de vehículos o personal). Se emplea un modelo LSTM para predecir la cantidad de viajes diarios durante un horizonte de 30 días. El módulo se integra en una aplicación Streamlit para visualización interactiva, mostrando gráficas de demanda, descomposición aditiva, y métricas de evaluación.
Limitación: Los datos disponibles tienen solo 3 fechas únicas por destino, lo que requiere aumentación de datos (interpolación) para generar series temporales sintéticas, afectando la precisión de las predicciones.

1.2 Dataset

- **Archivos**:
    - Final_Updated_Expanded_UserHistory.csv: Contiene el historial de viajes con columnas VisitDate (fechas de visitas) y DestinationID (identificador del destino).
    - Expanded_Destinations.csv: Incluye detalles de los destinos (DestinationID, Name, Type, Popularity, BestTimeToVisit).
- **Fuente**: Travel Recommendation Dataset (Kaggle, adaptado).
- **Características**:
    - Los datos originales tienen solo 3 fechas únicas por destino, lo que limita la capacidad de modelado de series temporales.
    - Se generaron series sintéticas (~79 puntos por destino) mediante interpolación para permitir el análisis de estacionalidad y tendencias.
- **Ubicación**: Los archivos deben estar en la carpeta ./proyecto_streamlit/ en Visual Studio Code.

1.3 Preprocesamiento

El preprocesamiento se diseñó para transformar los datos crudos en series temporales aptas para el modelo LSTM y la descomposición aditiva:

- **Agrupación de viajes**: Se agruparon los datos de Final_Updated_Expanded_UserHistory.csv por VisitDate y DestinationID para contar los viajes diarios (user_hist.groupby(["VisitDate", "DestinationID"]).size()).
- **Pivot diario**: Se creó una tabla pivote con fechas diarias (pd.date_range) como índice y destinos como columnas, rellenando días sin viajes con ceros.
- **Escalado**: Los datos de viajes se escalaron con MinMaxScaler para normalizarlos en el rango [0, 1], adecuado para el entrenamiento del LSTM.
- **Secuencias para LSTM**: Se generaron secuencias de 7 días (seq_len=7) como entrada para predecir el siguiente día.

- **Interpolación para descomposición**: Debido a las 3 fechas únicas, se interpolaron los datos faltantes (ser.interpolate().fillna(method="bfill").fillna(method="ffill")) para la descomposición aditiva (seasonal_decompose).

Nota sobre aumentación: La interpolación genera series sintéticas, introduciendo estacionalidad artificial que limita la validez de las predicciones. Las visualizaciones (gráficas de demanda y descomposición) ayudan a interpretar estas series.

1.4 Diseño de modelos

- **Modelo**: Red neuronal LSTM con:
  - Una capa LSTM de 50 unidades (LSTM(50, input_shape=(seq_len, 1))).
  - Una capa densa de salida (Dense(1)).
- **Entrada**: Secuencias de 7 días de viajes escalados.
- **Salida**: Predicciones de viajes para los próximos 30 días (horizon_days=30).
- **Entrenamiento**:
  - Optimizador: adam.
  - Pérdida: Error cuadrático medio (mse).
  - Parámetros: 50 épocas, tamaño de lote 8.
- **Predicción**: Se usa un enfoque recursivo, donde las predicciones se alimentan como entrada para los siguientes días.
- **Visualizaciones**:
  - **Gráfica de demanda**: Muestra datos históricos y predicciones (línea roja, rejilla, títulos claros).
  - **Descomposición aditiva**: Muestra tendencia (azul), estacionalidad (verde), y residuo (morado) en subgráficas separadas.
- **Implementación en Visual Studio Code**: El código (modulo1_completo.py) genera archivos en ./proyecto_streamlit/ y está optimizado para un entorno local.

1.5 Evaluación

- **Métricas**:
  - **RMSE**: Raíz del error cuadrático medio, mide la magnitud del error en las predicciones.
  - **MAE**: Error absoluto medio, indica el error promedio en valor absoluto.
  - **R^2**: Coeficiente de determinación, mide la proporción de varianza explicada.
- **Método**: Back-test comparando predicciones con datos históricos (o sintéticos para los últimos 30 días si los datos son insuficientes).
- **Resultados**:
  - Las métricas se imprimen en la consola por cada ruta (por ejemplo, RMSE=10.23, MAE=8.45, R^2=0.12).
  - R^2 puede ser bajo o negativo debido a los datos sintéticos, lo que refleja la dificultad de modelar con solo 3 fechas únicas.
- **Visualizaciones**: Las gráficas de demanda y descomposición son esenciales para evaluar visualmente las tendencias y la estacionalidad, haciendo los resultados más comprensibles.

Solución al problema de imágenes:

- **Problema**: Las imágenes (demanda_<ruta>.png, descomposicion_<ruta>.png) no se mostraban en Streamlit.
- **Solución**:
    - Se ajustaron las rutas en modulo1_completo.py para guardar en ./proyecto_streamlit/.
    - Se mejoraron las visualizaciones (DPI=300, colores distintivos, rejilla).
    - En app.py, se verifican las rutas de los archivos y se muestran advertencias si no se encuentran.
    - Se aseguró que los nombres de los PNG coincidan con los CSV (predicciones_<ruta>.csv → demanda_<ruta>.png).

1.6 Resultados

- **Archivos generados** (en ./proyecto_streamlit/):
    - predicciones_<ruta>.csv: DataFrame con fechas (ds) y número de viajes (y), incluyendo datos históricos y predicciones.
    - demanda_<ruta>.png: Gráfica de demanda (línea roja, rejilla, títulos grandes) mostrando datos históricos y predicciones.
    - descomposicion_<ruta>.png: Descomposición aditiva con subgráficas de tendencia (azul), estacionalidad (verde), y residuo (morado).
- **Visualización en Streamlit**:
    - Pestaña interactiva con:
        - Selector de rutas (st.selectbox).
        - Gráfica interactiva de demanda (matplotlib).
        - Imágenes de demanda y descomposición en columnas (st.image, st.columns).
        - Tabla de métricas (placeholder: RMSE, MAE, R^2).
        - Botón para descargar el CSV de predicciones.
    - Las visualizaciones son claras y atractivas, facilitando la interpretación de las tendencias.
- **Integración en Visual Studio Code**:
    - El módulo se ejecuta con python modulo1_completo.py en un entorno virtual.
    - Streamlit se ejecuta con streamlit run app.py, mostrando los resultados en http://localhost:8501.
- **Limitación**: Las predicciones son menos confiables debido a los datos sintéticos, pero las visualizaciones destacan patrones útiles.

1.7 Conclusiones

- **Logros**:
    - El Módulo 1 predice la demanda de viajes usando LSTM, generando resultados visualmente comprensibles.
    - Las visualizaciones mejoradas (gráficas de demanda y descomposición) resuelven el problema de imágenes no mostradas, haciendo los resultados más atractivos y fáciles de interpretar.
    - La implementación en Visual Studio Code es robusta, con rutas claras (./proyecto_streamlit/) y manejo de errores.

- **Limitaciones**:
    - Los datos sintéticos (3 fechas únicas) generan series con estacionalidad artificial, reduciendo la precisión (R^2 bajo).
    - Las métricas son limitadas por la falta de datos reales para validación.
- **Trabajo futuro**:
    - Recolectar más datos históricos para mejorar la precisión del modelo.
    - Usar bibliotecas como Plotly para gráficas interactivas en Streamlit.
    - Guardar métricas en un archivo (por ejemplo, CSV) para mostrarlas dinámicamente en Streamlit.
- **Importancia de las visualizaciones**: Las gráficas y descomposiciones son clave para entender las predicciones, especialmente con datos sintéticos, y hacen que el módulo sea más accesible para usuarios no técnicos.

Referencias:

- Travel Recommendation
- Dataset: https://www.kaggle.com/datasets/amanmehra23/travel-recommendation-dataset
- TensorFlow: https://www.tensorflow.org/
- Statsmodels: https://www.statsmodels.org/
- Streamlit: https://streamlit.io/