Manual de Usuario: Análisis de Datos Climáticos con TabNet

Grupo de Control y Procesamiento Digital de Señales
Universidad Nacional de Colombia
CHEC

April 28, 2025

Contents

1 Descripción General

Este script de Python está diseñado para realizar análisis de datos climáticos utilizando el modelo TabNet, una arquitectura de aprendizaje profundo especializada en datos tabulares. El objetivo principal es predecir y analizar métricas de calidad del servicio eléctrico, como el SAIFI (System Average Interruption Frequency Index) y el SAIDI (System Average Interruption Duration Index), basándose en variables climáticas y otras características relevantes. El script incluye funcionalidades para la carga de datos, preprocesamiento, entrenamiento del modelo, evaluación y visualización de resultados.

2 Funcionalidades Principales

- **Carga y Preprocesamiento de Datos:** Lectura de datos desde archivos CSV, limpieza de datos faltantes, codificación de variables categóricas y normalización de variables numéricas.
- **Entrenamiento del Modelo TabNet:** Utilización del modelo TabNetRegressor para entrenar un modelo de regresión que predice las métricas SAIFI y SAIDI.
- **Evaluación del Modelo:** Cálculo de métricas de rendimiento como el MAE (Mean Absolute Error) y el R^2 (coeficiente de determinación) y visualización de resultados a través de gráficos de dispersión.
- **Visualización de Resultados:** Generación de gráficos de dispersión, violin plots y mapas de calor para visualizar la relevancia de las características y la correlación entre variables.
- **Guardado de Resultados:** Guardado de máscaras y matrices de explicación en archivos para su posterior análisis.

3 Bibliotecas Necesarias

Para ejecutar este script, es necesario tener instaladas las siguientes bibliotecas de Python:

- `pytorch-tabnet`
- `torch`
- `sklearn`
- `pandas`
- `numpy`
- `matplotlib`
- `seaborn`
- `scipy`
- `openTSNE`
- `wget`

Estas bibliotecas pueden ser instaladas utilizando el gestor de paquetes `pip` de Python. Por ejemplo:

```
pip install pytorch-tabnet torch sklearn pandas numpy matplotlib seaborn scipy openTSNE wget
```

4 Archivos Necesarios

Para que el script funcione correctamente, se deben tener disponibles los siguientes archivos:

- `super_tabla_clima_v0.csv`: Archivo CSV que contiene los datos climáticos y de eventos de interrupción del servicio eléctrico. Este archivo debe estar ubicado en la ruta: `/content/drive/Shareddrives/CHEC/data_chec/datos/Data_Total/Clima/`.
- `mask.npy`: Archivo que contiene una máscara utilizada para la visualización de resultados. Este archivo debe estar ubicado en la ruta `/content/drive/Shareddrives/CURSO CHEC/Proyecto CHEC/Tabnet/`.

5 Instrucciones de Uso

5.1 Configuración Inicial

1. **Montar Google Drive:** Asegúrese de que su entorno de Google Colab esté configurado para acceder a Google Drive. Esto puede hacerse ejecutando la siguiente celda de código:

```
from google.colab import drive
drive.mount('/content/drive')
```

2. **Instalar Bibliotecas:** Instale las bibliotecas necesarias utilizando el gestor de paquetes `pip`. Esto puede hacerse ejecutando las siguientes celdas de código:

```
!pip install pytorch-tabnet --quiet
!pip install wget --quiet
!pip install openTSNE
```

5.2 Carga y Preprocesamiento de Datos

1. **Cargar Datos:** El script carga los datos desde el archivo `super_tabla_clima_v0.csv` y realiza un filtrado inicial para excluir eventos con una duración mayor a 100 horas.

```
Xdata = pd.read_csv('/content/drive/Shareddrives/CHEC/data_chec/datos/Data_Total/Clima
/super_tabla_clima_v0.csv')
Xdata = Xdata[Xdata['duracion_h'] <= 100]
```

2. **Preprocesamiento:** El script elimina columnas no utilizadas, codifica variables categóricas y normaliza variables numéricas. Asegúrese de que las columnas especificadas en el script existan en su archivo de datos.

5.3 Entrenamiento del Modelo

1. **Definir Parámetros:** El script define un conjunto de parámetros para el modelo TabNet. Estos parámetros pueden ser ajustados según sea necesario.

```
best_params = {'n_d': 144, 'n_a': 144, 'n_steps': 10, 'gamma': 94.66997047890686, ...}
```

2. **Entrenar Modelo:** El modelo se entrena utilizando los datos de entrenamiento y validación. El script utiliza una técnica de aumento de datos (RegressionSMOTE) y un optimizador (RMSprop) para mejorar el rendimiento del modelo.

```
clf.fit(
    X_train=X_train,
    y_train=y_train[:,0:1],
    eval_set=[(X_train, y_train[:,0:1]), (X_valid, y_valid[:,0:1])],
    eval_name=['train', 'valid'],
    eval_metric=['mae'],
    loss_fn=my_r2_score_fn,
    max_epochs=150,
    patience=60,
    batch_size=batch_size,
    virtual_batch_size=virtual_batch_size,
    num_workers=0,
    drop_last=False,
    augmentations=aug,
)
```

5.4 Evaluación del Modelo

1. **Generar Predicciones:** El script genera predicciones utilizando el modelo entrenado y evalúa el rendimiento utilizando el R^2 y el MAE.

```
y_pred = clf.predict(X_test)
```

2. **Visualizar Resultados:** El script genera gráficos de dispersión y violin plots para visualizar la relevancia de las características y la correlación entre variables.

```
fig, axs = plt.subplots(1, 3, figsize=(18, 6))
axs[0].scatter(y_test[:, 0], y_pred[:, 0], alpha=0.5)
axs[0].plot([y_test[:, 0].min(), y_test[:, 0].max()], [y_test[:, 0].min(), y_test[:, 0].max()],
'r-', label=f'R² = {r2_1:.2f}')
```

5.5 Guardado de Resultados

1. **Guardar Máscaras y Matrices de Explicación:** El script guarda las máscaras y matrices de explicación en archivos para su posterior análisis.

```
np.save(f"{savepath}masks_iteration_{iteration+1}.npy", masks, allow_pickle=True)
```

6 Consideraciones Adicionales

- **Ruta de Archivos:** Asegúrese de que las rutas de los archivos de datos y resultados sean correctas y accesibles desde su entorno de ejecución.
- **Parámetros del Modelo:** Los parámetros del modelo pueden ser ajustados para mejorar el rendimiento. Se recomienda realizar un análisis de sensibilidad para determinar los valores óptimos.
- **Visualización de Resultados:** Las visualizaciones generadas pueden ser personalizadas según las necesidades específicas del análisis.