Instalación y primero pasos para usar be-GReaT

I. Instalación

Para realizar la instalación, usaremos una distribución de GNU/Linux, en este caso Linux Mint.

Primero, deberemos poseer la versión 3.9 de Python. En el caso de no poseerla, realizar las siguientes instrucciones en la consola:

1. sudo apt install software-properties-common
2. sudo add-apt-repository ppa:deadsnakes/ppa
3. sudo apt-get update
4. sudo apt install python3.9
5. (opcional) python3.9 –version (para comprobar que se instaló correctamente)
6. pip install be_great

Con esto habríamos instalado correctamente be-GReaT para nuestra distribución.

Segundo, vamos a hacer uso de algunos datasets (conjunto de datos tabulados) que nos proporciona la librería SKlearn. Para instalarla debemos realizar las siguientes instrucciones en la consola:

1. sudo apt-get update
2. pip3 install -U scikit-learn

Con todo esto configurado, solo haría falta crear un directorio y los archivos de ejemplo, para aprender la sintaxis y poder observar los resultados proporcionados por be-GReaT.

II. Primer ejemplo: Diabetes

Para este primer ejemplo crearemos un archivo vacío de extensión .py (EjemploDiabetes.py).

El objetivo es el siguiente: A partir de los datos reales brindados por el dataset de SKlearn, crear unos datos sintéticos usando be-GReaT que nos permitan anonimizar los anteriores y realizar una serie de comprobaciones que nos permitan ver si estos son prácticamente iguales para el estudio.

Aquí dejo una imagen con el código fuente:

```python
from be_great import GReaT
from sklearn import datasets
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


def main(dataset):

    dataset = dataset.sample(10) #obtenemos un ejemplo de 10

    dataset.columns = ["age","sex","bmi","bp","s1","s2","s3","s4","s5",
"s6","DiseaseProg"] # Asignamos los nombres de las columnas
    #print(dataset.head()) # Vista previa del dataset (se muesta 5 rows)

    # Creamos un modelo Great
    great = GReaT("distilgpt2",     # Nombre del lenguaje de modelo usado
    epochs=10,                  # Numero de las epocas a entrenar
    save_steps=2,               # Guardar el modelo cada 2000 pasos
    logging_steps=5,
# Realiza un log de la perdida y el aprendizaje cada 500 pasos
    experiment_dir="trainer_diabetes",
# Nombre del directorio para guardar los pasos intermedios
    batch_size=16,                  # Tamaño del batch
    #lr_schedulet_type="constant"
    #learning_rate=5e-5
    )


# Una vez esta creado el modelo, comenzamos el entrenamiento del dataset
    trainer = great.fit(dataset)

    # Cuando termine el entrenamiento


#Mostrar grafica de como el crecimiento disminuye con el paso del entrena
miento
    loss_hist = trainer.state.log_history.copy()
    loss_hist.pop()

    loss = [x["loss"] for x in loss_hist]
    epochs = [x["epoch"] for x in loss_hist]

    lt.plot(epochs,loss)
    plt.show()

    great.save("diabetes")# Guardar el modelo
    great = GReaT.load_from_dir("diabetes") # Cargamos el modelo

    # Recogemos la muestra de datos sintéticos
    n_samples = 10
    g_sample = great.sample(n_samples,k=50)

    print(dataset.head()) # print de los datos originales
    print(g_sample.head()) # print de los datos sintéticos


#Apartir de aqui podriamos realizar un estudio de regresion para comparar
los datos obtenidos:


if __name__ == '__main__':
    #llamada al main
    dataset = datasets.load_diabetes(as_frame=True,scaled=False).frame
# Cargamos el dataset quitando el escalado para tener los datos reales
    main(dataset)
```

Figura 1: Código fuente

III. Ejemplos de regresión proporcionados por be-GReaT

El siguiente ejemplo está disponible en el GitHub que contiene la librería be-GReaT. En este, se nos sitúa en un escenario donde poseemos un dataset con información acerca de casas de california. A continuación, se muestra dos muestras (Una con los datos originales y otra con los datos sintéticos):

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Figura 2: Muestra de datos originales

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	6.0990	34.0	6.242424	1.000000	1758.0	2.856757	33.68	-117.88	4.09088
1	4.1875	31.0	5.879051	1.041030	1127.0	2.446928	38.63	-121.35	1.53564
2	3.6000	16.0	6.837838	1.000000	693.0	2.790062	36.64	-120.84	1.10397
3	4.8934	23.0	5.137055	0.945205	984.0	3.160714	38.00	-121.45	1.64981
4	2.4861	52.0	4.694915	1.015965	867.0	2.251497	37.31	-122.46	1.09459

Figura 3: Muestra de datos sintéticos

Al tratar de realizar un estudio con los datos, por ejemplo, realizar un estudio sobre la regresión (entre la longitud y latitud) de cada dataset, podemos observar como la regresión es prácticamente idéntica en ambos casos; mostrándose así, la gran funcionalidad que nos proporciona la generación de datos sintéticos. La capacidad de crear datos nuevos que no pertenecen a nadie (por lo que no tienen la necesidad de ser protegidos), pero a su vez, sirven de la misma manera que los originales para estudiarlos:

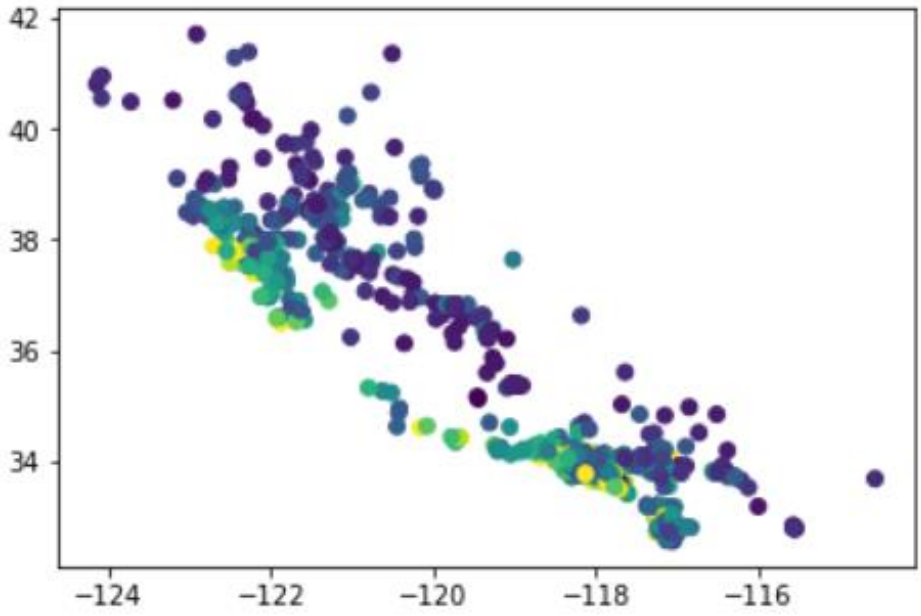

Figura 4: Regresión de los datos originales

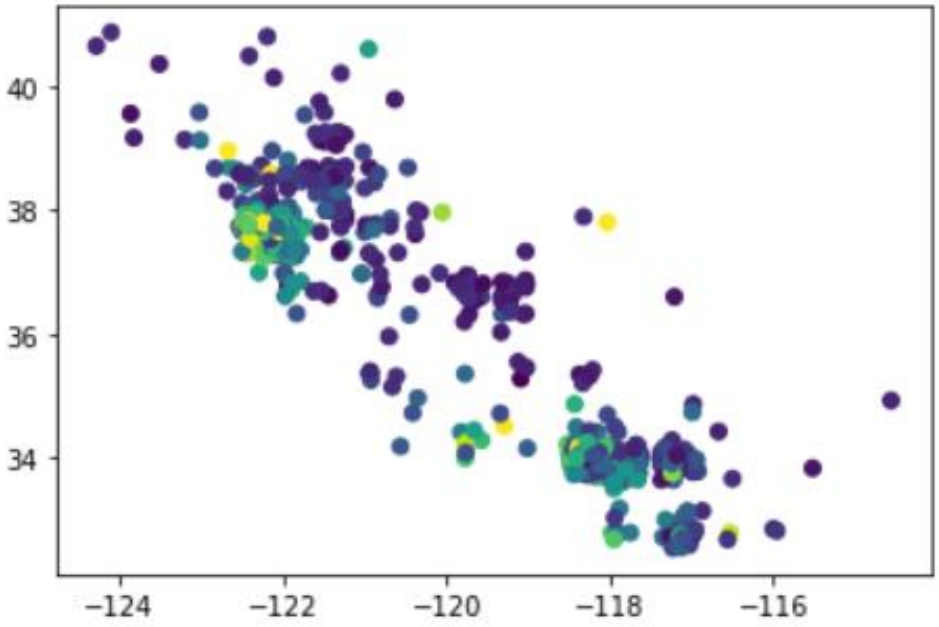

Figura 5: Regresión de los datos sintéticos

IV. Conclusión:

La generación de datos sintéticos es un gran paso en una buena dirección hacia la anonimización de los datos. Sin embargo, la calidad de estos depende fundamentalmente de que “tan entrenado” está el modelo que estemos usando por lo que es importante entrenar al modelo constantemente con distintos datasets, que le permitan ampliar su conocimiento para así exprimir su potencial al máximo.

Para consultar mas información acerca del proyecto, véase:

- https://github.com/kathrinse/be_great/
- https://kathrinse.github.io/be_great/
- https://arxiv.org/abs/2210.06280