DataSynth: Documentação Técnica

Seu Nome

June 21, 2024

Abstract

DataSynth é uma ferramenta projetada para anonimizar dados financeiros sensíveis utilizando ruído branco gaussiano. Esta documentação técnica explica o funcionamento do DataSynth, incluindo a geração de ruído branco, a aplicação de anonimização e a recuperação de dados encriptados.

Contents

1 Introdução **2**

2 Importações e Bibliotecas **2**

3 Geração de Ruído Branco **3**
3.1 Transformação de Box-Muller 3
3.2 Ajuste da Intensidade do Ruído 3

4 Aplicação do Ruído Branco **3**
4.1 Exemplo de Código 3

5 Visualização do Ruído Branco **4**
5.1 Visualização da Distribuição do Ruído Branco 4
5.2 Visualização em Tempo Discreto 4
5.3 Autocorrelação do Ruído Branco 5

6 DataSynth e Tempo Discreto **5**
6.1 Exemplo com Dados Financeiros 5

7 Recuperação de Dados **5**

8 Considerações Finais **6**

1 Introdução

DataSynth é uma solução inovadora para a anonimização de dados financeiros. Utilizando ruído branco gaussiano, o DataSynth preserva as propriedades estatísticas dos dados enquanto protege informações sensíveis.

2 Importações e Bibliotecas

O código utiliza várias bibliotecas para manipulação de dados, visualização e criptografia:

Importações e Bibliotecas

```
import pandas as pd
import numpy as np
import streamlit as st
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
import base64
from io import BytesIO
from cryptography.fernet import Fernet
import hashlib
import random
import string
```

- **pandas** e **numpy** são usados para manipulação de dados.
- **streamlit** é utilizado para criar a interface web.
- **matplotlib** e **seaborn** são usados para visualização de dados.
- **PIL** é utilizado para manipulação de imagens.
- **base64** e **io** são usados para codificação e manipulação de fluxos de bytes.
- **cryptography.fernet** é utilizado para criptografia simétrica.
- **hashlib** e **string** são usados para derivar chaves e gerar senhas.

3 Geração de Ruído Branco

A geração de ruído branco no DataSynth é realizada utilizando a transformação de Box-Muller. Esta transformação gera variáveis aleatórias gaussianas (normais) a partir de variáveis uniformemente distribuídas.

3.1 Transformação de Box-Muller

A transformação de Box-Muller é utilizada para converter duas variáveis aleatórias uniformemente distribuídas U_1 e U_2 em duas variáveis gaussianas Z_0 e Z_1 com média zero e desvio padrão 1.

$$Z_0 = \sqrt{-2 \ln U_1} \cos(2\pi U_2) \tag{1}$$

$$Z_1 = \sqrt{-2 \ln U_1} \sin(2\pi U_2) \tag{2}$$

Essas variáveis gaussianas são então escaladas para ajustar a intensidade do ruído.

3.2 Ajuste da Intensidade do Ruído

O ruído gerado é ajustado por um fator de escala definido pelo usuário. Este fator de escala afeta diretamente o desvio padrão do ruído aplicado, enquanto a média permanece zero.

$$\textit{ruído} = Z_0 \times \text{ruído\_nível} \tag{3}$$

4 Aplicação do Ruído Branco

O ruído branco gaussiano é aplicado aos dados financeiros para anonimização. O nível de ruído pode ser ajustado para controlar a intensidade do ruído aplicado.

4.1 Exemplo de Código

```
def adicionar_ruido_branco_e_encriptar(dados, coluna, chave,
    ruido_nivel=1):
     tamanho = len(dados)
     Z0 = gerar_ruido_branco(tamanho)
     dados[coluna] = dados[coluna].apply(lambda x: float(str(x
         ).replace(',', '.')) if isinstance(x, str) else x)
     ruido = Z0 * ruido_nivel
     dados_sinteticos = dados.copy()
```

```
    dados_sinteticos[coluna] += ruido

    # Encriptar os dados originais
    fernet = Fernet(chave)
    dados['Encrypted_' + coluna] = dados[coluna].apply(lambda
        x: fernet.encrypt(str(x).encode()).decode())
    return dados_sinteticos, dados
```

5 Visualização do Ruído Branco

O DataSynth permite a visualização do ruído branco gerado e da autocorrelação do ruído. Essas visualizações ajudam a entender como o ruído é aplicado e como ele afeta os dados.

5.1 Visualização da Distribuição do Ruído Branco

O histograma do ruído branco gerado, juntamente com a curva de densidade estimada, pode ser visualizado para verificar se o ruído segue uma distribuição normal.

```
def plot_ruido_branco(ruido, ruido_nivel):
    media_ruido = np.mean(ruido)
    desvio_padrao_ruido = np.std(ruido)

    fig, ax = plt.subplots(figsize=(10, 4))
    sns.histplot(ruido, bins=30, kde=True, ax=ax)
    ax.axvline(media_ruido, color='r', linestyle='--', label=
       f'M dia do Ru do: {media_ruido:.2f}')
    ax.axvline(desvio_padrao_ruido, color='g', linestyle='--'
       , label=f'Desvio Padr o do Ru do: {
       desvio_padrao_ruido:.2f}')
    ax.set_xlabel('Valor')
    ax.set_ylabel('Frequ ncia')
    ax.set_title(f'Ru do Branco Gaussiano Gerado (N vel de
       Ru do = {ruido_nivel})')
    ax.legend()
    st.pyplot(fig)
```

5.2 Visualização em Tempo Discreto

```
def plot_ruido_branco_tempo_discreto(ruido):
    fig, ax = plt.subplots(figsize=(10, 4))
    ax.plot(ruido, label='Ru do Branco Gaussiano')
    ax.set_xlabel('Amostras')
```

```
    ax.set_ylabel('Valor')
    ax.set_title('Ru do Branco Gaussiano em Tempo Discreto')
    ax.legend()
    st.pyplot(fig)
```

5.3 Autocorrelação do Ruído Branco

A autocorrelação é uma medida da correlação de uma série temporal com uma cópia defasada dela mesma. Para um ruído branco ideal, a autocorrelação deve ser zero para qualquer defasagem diferente de zero. Matematicamente, a autocorrelação $R(\tau)$ de uma série temporal $X(t)$ é definida como:

$$R(\tau) = \frac{1}{N} \sum_{t=1}^{N-\tau} X(t)X(t+\tau) \tag{4}$$

Onde N é o número total de amostras e τ é a defasagem.

6 DataSynth e Tempo Discreto

O DataSynth lida com dados financeiros que são amostras discretas, como transações financeiras registradas em momentos específicos. Portanto, o ruído branco aplicado aos dados financeiros no DataSynth é gerado e analisado em um contexto de tempo discreto.

6.1 Exemplo com Dados Financeiros

Considere um banco de dados financeiro contendo transações com as seguintes colunas: ‘Nome_Cliente‘, ‘Numero_Cartao‘, ‘Email_Cliente‘, ‘Telefone_Cliente‘, ‘Endereco_IP‘, e ‘Valor_Transacao‘.

```
ruido_nivel = st.slider("N vel de Ru do", min_value=0,
    max_value=1000, value=1, step=1)
banco_dados_sinteticos, banco_dados_encriptados =
    adicionar_ruido_branco_e_encriptar(st.session_state['
    banco_dados_anonimizados'], 'Valor_Transacao', st.
    session_state.chave, ruido_nivel)
```

7 Recuperação de Dados

O DataSynth permite a recuperação dos dados originais utilizando uma chave de decriptação. Esta funcionalidade garante que os dados possam ser revertidos para sua forma original de maneira segura, se necessário.

```
def decriptar_dados(dados, chave, colunas_texto):
    fernet = Fernet(chave)
    for coluna in dados.columns:
        if coluna.startswith('Encrypted_'):
            original_coluna = coluna.replace('Encrypted_', ''
                )
            if original_coluna in colunas_texto:
                try:
                    dados[original_coluna] = dados[coluna].
                        apply(lambda x: fernet.decrypt(x.
                        encode()).decode())
                except Exception as e:
                    st.write(f"Erro ao decriptar a coluna {
                        coluna}: {e}")
            else:
                try:
                    dados[original_coluna] = dados[coluna].
                        apply(lambda x: float(fernet.decrypt(x
                        .encode()).decode()))
                except Exception as e:
                    st.write(f"Erro ao decriptar a coluna {
                        coluna}: {e}")
    # Remover colunas encriptadas ap s decripta  o
    colunas_encriptadas = [col for col in dados.columns if
        col.startswith('Encrypted_')]
    dados.drop(columns=colunas_encriptadas, inplace=True)
    return dados
```

8 Considerações Finais

DataSynth é uma ferramenta poderosa para anonimização de dados financeiros, garantindo privacidade e conformidade com as regulamentações. Com a visualização das propriedades estatísticas do ruído branco aplicado, os usuários podem ajustar os níveis de anonimização conforme necessário para proteger as informações sensíveis sem comprometer a utilidade dos dados.