Documentação Técnica do **DataSynth** e **DataSynthDecripter**

Viviane Viana Sofiste de Abreu

June 21, 2024

Contents

1 Introdução

O DataSynth é uma ferramenta projetada para anonimizar dados sensíveis de transações financeiras utilizando técnicas de ruído branco e criptografia. O DataSynthDecripter é uma ferramenta complementar que permite a recuperação dos dados originais a partir dos dados encriptados. Este documento detalha cada trecho do código, explicando suas funcionalidades e conceitos subjacentes, incluindo a transformação de Box-Muller para gerar ruído branco gaussiano.

2 Importações e Bibliotecas

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

```
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 Funções de Geração de Ruído Branco

3.1 Transformação de Box-Muller

A transformação de Box-Muller é uma técnica para gerar variáveis aleatórias com distribuição normal (gaussiana) a partir de variáveis aleatórias uniformemente distribuídas. A função `gerar_ruido_branco` implementa essa técnica:

```python
def gerar_ruido_branco(tamanho):
    U1 = np.random.uniform(0, 1, tamanho)
    U2 = np.random.uniform(0, 1, tamanho)
    Z0 = np.sqrt(-2 * np.log(U1)) * np.cos(2 * np.pi * U2)
    return Z0
```

Explicação:

- `U1` e `U2` são variáveis aleatórias uniformemente distribuídas entre 0 e 1.
- `Z0` é uma variável aleatória com distribuição normal gerada pela transformação de Box-Muller.
- A transformação utiliza funções logarítmica e trigonométricas para converter as variáveis uniformes em variáveis normais.

4 Funções de Criptografia

4.1 Derivar Chave a partir da Senha

A função `derivar_chave` gera uma chave de criptografia a partir de uma senha usando PBKDF2 com HMAC-SHA256:

```python
def derivar_chave(senha):
    kdf_salt = b'some_salt_'  # Isso deve ser armazenado de maneira segura
    kdf_iterations = 100000
    chave = base64.urlsafe_b64encode(hashlib.pbkdf2_hmac('sha256', senha.encode(), kdf_salt, kdf_iterations, dklen=32))
    return chave
```

Explicação:

- `senha` é a senha fornecida pelo usuário.
- `kdf_salt` é o salt usado na derivação da chave.
- `kdf_iterations` é o número de iterações para a função PBKDF2.
- `hashlib.pbkdf2_hmac` deriva uma chave segura de 256 bits.
- `base64.urlsafe_b64encode` codifica a chave derivada em um formato seguro para URLs.

4.2 Adicionar Ruído Branco e Encriptar Dados

A função `adicionar_ruido_branco_e_encriptar` adiciona ruído branco a uma coluna de dados e encripta os dados originais:

```python
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(',', '.')))
    ruido = Z0 * ruido_nivel * dados[coluna].std()
    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
```

Explicação:

- `dados` é o DataFrame contendo os dados.
- `coluna` é a coluna onde o ruído será adicionado.
- `chave` é a chave de criptografia.
- `ruido_nivel` define a intensidade do ruído adicionado.
- `dados_sinteticos` contém os dados com ruído adicionado.
- Os dados originais são encriptados usando `Fernet`.

5 Funções de Anonimização

5.1 Anonimizar Dados Textuais com Ruído Branco

A função `anonimizar_texto_com_ruido` anonimiza dados textuais adicionando ruído:

```
def anonimizar_texto_com_ruido(dados, colunas_texto, chave):
    dados_anonimizados = dados.copy()
    fernet = Fernet(chave)
    for coluna in colunas_texto:
        dados_anonimizados['Encrypted_' + coluna] =
            dados_anonimizados[coluna].apply(lambda x: fernet.
            encrypt(str(x).encode()).decode())
        dados_anonimizados[coluna] = gerar_imagem_ruido()
    return dados_anonimizados
```

Explicação:

- `dados` é o DataFrame original.
- `colunas_texto` são as colunas textuais a serem anonimizadas.
- `dados_anonimizados` contém os dados anonimizados.
- Os valores textuais originais são encriptados e substituídos por imagens de ruído.

5.2 Gerar Imagem de Ruído Branco

A função `gerar_imagem_ruido` cria uma imagem de ruído branco:

```
def gerar_imagem_ruido(branco=True):
    largura, altura = 50, 50
    array_ruido = np.random.randint(0, 255, (altura, largura),
       dtype=np.uint8)
    img_ruido = Image.fromarray(array_ruido)
    buffer = BytesIO()
    img_ruido.save(buffer, format="PNG")
    img_b64 = base64.b64encode(buffer.getvalue()).decode()
    return f'<img src="{img_b64}" width="50"
        height="50">'
```

Explicação:

- `array_ruido` gera uma matriz de valores aleatórios.
- `Image.fromarray` cria uma imagem a partir da matriz de ruído.
- A imagem é codificada em base64 para ser exibida no HTML.

6 Funções de Interface do Streamlit

6.1 Ajustar Estilo CSS

A função `ajustar_estilo` aplica estilos CSS à interface:

```
def ajustar_estilo():
    st.markdown(
        """
        <style>
        .reportview-container .main .block-container {
            max-width: 80%;
            margin: auto;
            padding: 2rem;
        }
        .dataframe {
            width: 100%;
            overflow-x: auto;
            display: block;
            white-space: nowrap;
        }
        </style>
        """,
        unsafe_allow_html=True
    )
```

Explicação:

- Ajusta o layout da interface Streamlit para melhor visualização.

6.2 Formatar Colunas do DataFrame

A função `formatar_colunas` formata colunas específicas do DataFrame:

```
def formatar_colunas(dados):
    dados['Codigo_Autorizacao'] = dados['Codigo_Autorizacao'].
       astype(int)
    dados['Valor_Transacao'] = dados['Valor_Transacao'].apply(
       lambda x: float(str(x).replace(',', '.')))
    dados['Valor_Transacao'] = dados['Valor_Transacao'].apply(
       lambda x: f'{x:,.2f}'.replace(',', 'v').replace('.', ',').
       replace('v', '.'))
    return dados
```

Explicação:

- Converte colunas para tipos específicos e formata valores para melhor exibição.

6.3 Gerar Senha Aleatória

A função `gerar_senha_aleatoria` gera uma senha aleatória:

```
def gerar_senha_aleatoria(tamanho=12):
    caracteres = string.ascii_letters + string.digits
    senha = ''.join(random.choice(caracteres) for i in range(
       tamanho))
    return senha
```

Explicação:

- Gera uma senha aleatória de comprimento especificado, combinando letras e dígitos.

6.4 Decriptar Dados

A função `decriptar_dados` decripta os dados encriptados:

```
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
```

Explicação:

- Decripta as colunas que foram encriptadas, convertendo-as de volta para seus valores originais.

7 Fluxo Principal do DataSynth

7.1 Interface do Streamlit

A interface principal do Streamlit é configurada para carregar um arquivo CSV, aplicar anonimização, adicionar ruído branco, encriptar os dados e permitir a decriptação dos dados encriptados:

```
ajustar_estilo()

st.title("DataSynth - Demonstra  o de Anonimiza  o de Dados Financeiros")
st.write("Veja como os dados sens veis  s  o anonimizados para proteger a privacidade dos clientes.")

# Inicializa a senha e chave usando o estado do Streamlit
if 'senha_aleatoria' not in st.session_state:
    st.session_state.senha_aleatoria = None

if 'chave' not in st.session_state:
    st.session_state.chave = None

if 'uploaded_file_buffer' not in st.session_state:
    st.session_state.uploaded_file_buffer = None

# Bot o para gerar nova sess o
def gerar_nova_sessao():
    st.session_state.senha_aleatoria = None
    st.session_state.chave = None
    st.session_state.uploaded_file_buffer = None
    st.experimental_rerun()

# Carregar banco de dados de um arquivo CSV
uploaded_file = st.file_uploader("Escolha um arquivo CSV", type="csv")
if uploaded_file is not None:
    # Verifica se o arquivo carregado    novo
```

```
27      uploaded_file_buffer = uploaded_file.getvalue()
28      if uploaded_file_buffer != st.session_state.
           uploaded_file_buffer:
29          # Limpa o estado atual
30          st.session_state.uploaded_file_buffer =
               uploaded_file_buffer
31
32          # Gera  o da senha aleat ria ap s o upload do CSV
33          st.session_state.senha_aleatoria = gerar_senha_aleatoria()
34
35          # Gera  o da chave secreta ap s o upload do CSV
36          st.session_state.chave = derivar_chave(st.session_state.
               senha_aleatoria)
37
38          # Limpa as tabelas e entradas de senha
39          st.session_state['banco_dados_transacoes'] = None
40          st.session_state['banco_dados_anonimizados'] = None
41          st.session_state['banco_dados_sinteticos'] = None
42          st.session_state['banco_dados_encriptados'] = None
43          st.session_state['senha_input'] = None
44
45      fernet = Fernet(st.session_state.chave)
46
47      st.write(f"Senha para decriptar: {st.session_state.
           senha_aleatoria}")
48
49      banco_dados_transacoes = pd.read_csv(BytesIO(st.session_state.
           uploaded_file_buffer))
50      banco_dados_transacoes['Valor_Transacao'] =
           banco_dados_transacoes['Valor_Transacao'].apply(lambda x:
           float(str(x).replace(',', '.')))
51      banco_dados_transacoes = formatar_colunas(
           banco_dados_transacoes)
52
53      # Armazenar os dados no estado
54      st.session_state['banco_dados_transacoes'] =
           banco_dados_transacoes
55
56      # Adiciona o bot o "Gerar Nova Sess o" logo abaixo do upload
           do CSV
57      st.button("Gerar Nova Sess o", on_click=gerar_nova_sessao)
58
59      # Exibir dados originais
60      st.write("### Dados Originais")
61      st.write(st.session_state['banco_dados_transacoes'].head())
62
63      # Anonimizar dados textuais com ru do branco
64      colunas_texto = ['Nome_Cliente', 'Numero_Cartao', '
           Email_Cliente', 'Telefone_Cliente', 'Endereco_IP']
65      banco_dados_anonimizados = anonimizar_texto_com_ruido(st.
           session_state['banco_dados_transacoes'], colunas_texto, st.
```

```
        session_state.chave)
    banco_dados_anonimizados['Valor_Transacao'] =
        banco_dados_anonimizados['Valor_Transacao'].apply(lambda x:
         float(str(x).replace(',', '.')))
    banco_dados_anonimizados = formatar_colunas(
        banco_dados_anonimizados)

    # Armazenar os dados anonimizados no estado
    st.session_state['banco_dados_anonimizados'] =
        banco_dados_anonimizados

    # Aplicar ru do branco e encriptar os dados originais
    ruido_nivel = st.slider("N vel de Ru do", min_value=0,
        max_value=1000, value=50, step=10)
    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)
    banco_dados_sinteticos = formatar_colunas(
        banco_dados_sinteticos)

    # Armazenar os dados sinteticos e encriptados no estado
    st.session_state['banco_dados_sinteticos'] =
        banco_dados_sinteticos
    st.session_state['banco_dados_encriptados'] =
        banco_dados_encriptados

    # Remover colunas encriptadas dos dados sint ticos
    colunas_encriptadas = [col for col in st.session_state['
        banco_dados_sinteticos'].columns if col.startswith('
        Encrypted_')]
    banco_dados_sinteticos_sem_encriptacao = st.session_state['
        banco_dados_sinteticos'].drop(columns=colunas_encriptadas)

    # Exibir dados anonimizados com ru do branco aplicado com
        controle deslizante horizontal
    st.write("### Dados Anonimizados com Ru do Branco Aplicado")
    st.write('<div style="overflow-x: auto;">' +
        banco_dados_sinteticos_sem_encriptacao.head(5).to_html(
        escape=False, index=False) + '</div>', unsafe_allow_html=
        True)

    # Bot o para exportar dados encriptados
    csv_encriptado = banco_dados_encriptados.to_csv(index=False).
        encode('utf-8')
    st.download_button(
         label="Exportar Dados Encriptados",
         data=csv_encriptado,
         file_name='dados_encriptados.csv',
         mime='text/csv'
    )
```

```
97
98      # Verifica  o de senha e decripta  o de dados
99      senha_input = st.text_input("Insira a senha para decriptar os
            dados", type="password")
100     decriptar_btn = st.button("Decriptar Dados", key="
            decriptar_btn")
101     if decriptar_btn and senha_input == st.session_state.
            senha_aleatoria:
102         banco_dados_decriptados = decriptar_dados(st.session_state
                ['banco_dados_encriptados'], st.session_state.chave,
                colunas_texto)
103         st.write("### Dados Decriptados")
104         st.write('<div style="overflow-x: auto;">' +
                banco_dados_decriptados.head(5).to_html(escape=False,
                index=False) + '</div>', unsafe_allow_html=True)
105     elif decriptar_btn:
106         st.write("Senha incorreta. Por favor, tente novamente.")
107
108     # Visualizar os dados originais e sint ticos
109     fig, ax = plt.subplots(1, 2, figsize=(12, 6))
110     ax[0].hist(st.session_state['banco_dados_transacoes']['
            Valor_Transacao'], bins=30, alpha=0.6, color='blue', label=
            'Original')
111     ax[0].set_title('Dados Originais')
112     ax[0].set_xlabel('Valor Transa  o')
113     ax[0].set_ylabel('Frequ ncia')
114
115     ax[1].hist(st.session_state['banco_dados_sinteticos']['
            Valor_Transacao'].apply(lambda x: float(str(x).replace('.',
            '').replace(',', '.'))), bins=30, alpha=0.6, color='green'
            , label='Sint tico')
116     ax[1].set_title(f'Dados Sint ticos (N vel de Ru do = {
            ruido_nivel})')
117     ax[1].set_xlabel('Valor Transa  o')
118     ax[1].set_ylabel('Frequ ncia')
119
120     st.pyplot(fig)
121
122     # Gerar e visualizar a PDF do ru do branco
123     ruido_branco = gerar_ruido_branco(1000)
124     fig, ax = plt.subplots()
125     sns.histplot(ruido_branco, bins=30, kde=True, ax=ax)
126     mu, std = np.mean(ruido_branco), np.std(ruido_branco)
127     xmin, xmax = ax.get_xlim()
128     x = np.linspace(xmin, xmax, 100)
129     p = (1/(np.sqrt(2*np.pi)*std)) * np.exp(-0.5*((x-mu)/std)**2)
130     ax.plot(x, p, 'k', linewidth=2)
131     ax.set_title("Histograma e Curva da Distribui  o Normal")
132     ax.set_xlabel("Valor")
133     ax.set_ylabel("Densidade")
134     st.pyplot(fig)
```

```
    st.write("Histograma e Curva da Distribui  o Normal para o Ru do Branco")
else:
    st.write("Por favor, carregue um arquivo CSV para continuar.")
```

8 Fluxo Principal do DataSynthDecripter

8.1 Interface do Streamlit

O DataSynthDecripter é uma ferramenta complementar que permite a recuperação dos dados originais a partir dos dados encriptados. A interface principal do Streamlit é configurada para carregar um arquivo CSV encriptado, solicitar a senha de decriptação, e exibir os dados decriptados:

```
import pandas as pd
import streamlit as st
from cryptography.fernet import Fernet
import base64
import hashlib
from io import BytesIO

# Fun  o para derivar chave a partir da senha
def derivar_chave(senha):
    kdf_salt = b'some_salt_'  # Isso deve ser armazenado de maneira segura
    kdf_iterations = 100000
    chave = base64.urlsafe_b64encode(hashlib.pbkdf2_hmac('sha256', senha.encode(), kdf_salt, kdf_iterations, dklen=32))
    return chave

# Fun  o para decriptar os dados
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:
```

```
30                     st.write(f"Erro ao decriptar a coluna {coluna
                           }: {e}")
31     # Remover colunas encriptadas ap s decripta  o
32     colunas_encriptadas = [col for col in dados.columns if col.
           startswith('Encrypted_')]
33     dados.drop(columns=colunas_encriptadas, inplace=True)
34     return dados
35
36 # Interface do Streamlit
37 st.title("DataSynthDecripter - Desencripta  o de Dados")
38
39 # Carregar banco de dados encriptado de um arquivo CSV
40 uploaded_file = st.file_uploader("Escolha um arquivo CSV
       encriptado", type="csv")
41 if uploaded_file is not None:
42     banco_dados_encriptados = pd.read_csv(uploaded_file)
43
44     # Verifica  o de senha e decripta  o de dados
45     senha_input = st.text_input("Insira a senha para decriptar os
           dados", type="password")
46     if st.button("Decriptar Dados"):
47         chave = derivar_chave(senha_input)
48         try:
49             banco_dados_decriptados = decriptar_dados(
                   banco_dados_encriptados, chave, ['Nome_Cliente', '
                   Numero_Cartao', 'Email_Cliente', 'Telefone_Cliente'
                   , 'Endereco_IP', 'Valor_Transacao'])
50             st.write("### Dados Decriptados")
51             st.write(banco_dados_decriptados.head())
52         except Exception as e:
53             st.write(f"Erro na decripta  o: {e}")
```

Explicação:

- **Carregar Dados Encriptados:** O arquivo CSV encriptado é carregado através do `st.file_uploader`.
- **Solicitar Senha:** A senha de decriptação é solicitada ao usuário através do `st.text_input`.
- **Derivar Chave:** A chave de decriptação é derivada da senha usando a função `derivar_chave`.
- **Decriptar Dados:** Os dados encriptados são decriptados usando a função `decriptar_dados`, e os dados decriptados são exibidos na interface.