Sprint de CyberSecurity: Verificação de Fraudes

Problema

A proposta do Sprint de CyberSecurity se tratava de desenvolver uma solução com os recursos aprendidos para resolver alguns problemas enfrentados por um Banco. Entre esses problemas, encontrava-se: identificar acesso às contas de clientes falecidos, cancelamento de compras, denúncias de fraudes e identificação de pessoas fraudadores.

A ideia é, através da inteligência Artificial, ajudar os Bancos a melhorar o sistema de segurança dos códigos de identificação. Assim, independente do usuário ter ou não a senha e código de acesso da conta, caso esteja dentro de um dos grupos colocados, ele não possa acessar a conta e nem realizar transferências.

Recursos

Para a realização desse projeto, foram disponibilizadas algumas APIs, contendo o código de acesso de diferentes grupos do banco, os códigos dentro de cada grupo garante que não é permitido o acesso.

Ferramentas Utilizadas: Google Colab e Linguagem Python.

Método

Com os dados em mãos, o grupo deveria criar um sistema que possa receber os dados do usuário que está acessando o banco, e através desses, analisar se o código de acesso detecta alguma semelhança com as bases, permitindo ou não o acesso ao usuário.

Código

Para deixar de mais fácil compreensão, optamos por dividir o código em duas seções menores: código passo a passo e separado por etapa, e o código completo com o acesso à ‘webpage’.

Etapas do código:

- Instalar as bibliotecas necessárias;
- Realizar a importação das bibliotecas;
- Importas as APIs;
- Filtras os elementos das APIs;
- Relacionar o comando como um arquivo;
- Criando a WebPage;
- Analisar e Comparar os dados com as listas das APIs;
- Devolver o resultado para o usuário;
- Integração do código com a Web Page:

- Junção do código;

Instalação das Bibliotecas:

Aqui, faremos a instalação das bibliotecas que utilizaremos no código e em seguida, realizaremos a importação para as mesmas.

```
!pip install -q streamlit-ace
!pip install -q pyngrok
!pip install -q fuzzywuzzy[speedup]
!streamlit run app.py &>/dev/null&
```

Importação das Bibliotecas:

```
import requests as r
import json
from fuzzywuzzy import process
import re
```

Importar as APIs:

Aqui, criamos uma função para resgatar uma API, está localiza uma das listas de códigos de acessos de clientes que não podem acessar as contas. Puxamos a base, a transformando em um dicionário que tem como chave de busca o “id”.

```
def getFiltroCliente(codigoAcesso):
  # id = código de acesso do cliente
    id = r.get('https://emgabrielly.github.io/bases/falecidos.json')
  # Transformando em dicionário
    dict_id = id.json()['id']
```

Filtrando os elementos das APIs:

Aqui, consideramos que o cliente pode acabar digitando sim ou não traços e pontos, por isso é preciso criar esse filtro que despensa o que nós não utilizaremos.

```
# Verifica se está no formato desejado
    padrao = r"^(\d{5,5})$"
    match = re.search(padrao, codigoAcesso)
    print(match.group(1))
    for item in dict_id:
       for value in item.values():
          if str(match.group(1)) == str(value):
                    return True
```

Relacionando o comando como um arquivo:

Esse comando permite que você produza o código desenvolvido em um Notebook para um *Python* módulo. Note que nesta etapa, torna-se um comando dispensável se desenvolvido em outra plataforma diferente de jupyter ou colab, já que estamos colocando como um arquivo nomeado app.py.

```
%%writefile app.py
import streamlit as st
import requests
import json
from fuzzywuzzy import process
import re
```

Criando a Web Page:

Neste campo, chamamos uma função para criar nosso ambiente Web.
Assim como no HTML, os comandos são inseridos entre “<>”.

```
#Criando a webpage:
def main():
    html_temp = """
    <body style="background-color:black">
    <div style ="background-image:linear-gradient(to right,#33ccff,
pink);padding:13;border-radius:25px;">
    <h1 style ="color:white;text-align:center;">BANCO IAM</h1>
    </div>
    <br/>
    </body>
    """
    st.markdown(html_temp, unsafe_allow_html=True)
```

Na segunda aba de código, é onde está a “jogada”. Aqui, introduzimos a variável que armazena os dados fornecidos pelo usuário. A partir dela, efetuaremos a análise.

```
    codigo_id = st.text_input('Código de Acesso:')
```

Analisando os dados de entrada com as APIs:

As condicionais usadas apresentam a função de verificar cada vez que o botão “Acessar” é pressionado se os dados de entrada constam na API. Logo, é apresentado ao usuário o resultado do código.

```
    if st.button("Acessar"):
        if (getFiltroCliente(codigo_id)):
            return st.warning("Solicitação negada. O usuário não pode
acessar a conta")
        else:
            return st.success("Pode Acessar a conta")
```

Devolvendo o resultado para o usuário:

Aqui observamos que as condicionais if e else efetuam a seleção do que deve ou não ser apresentado ao usuário conforme o resultado dos dados.

Integração do código com um URL para acessar a Web Page:

O último código a ser adicionado é a URL. Aqui, fizemos a integração do código com um link que permite visualizá-lo como uma página ‘web’.

```
!streamlit run app.py & npx localtunnel --port 8501
```

Juntando o código:

Após entender cada etapa do código, a unimos em seções menores para o estruturar de maneira eficiente. Este deve ser o seu código:

```
!pip install -q streamlit
!pip install -q streamlit-ace
!pip install -q pyngrok
!pip install -q fuzzywuzzy[speedup]
!streamlit run app.py &>/dev/null&
```

```
%%writefile app.py
import streamlit as st
import requests
import json
from fuzzywuzzy import process
import re

#acessando a API
def getFiltroCliente(codigoAcesso):
  # id = código de acesso do cliente
    id = requests.get('https://emgabrielly.github.io/bases/falecidos.
json')
  # Transformando em dicionário
    dict_id = id.json()['id']

  # Verifica se está no formato desejado
    padrao = r"^(\d{5,5})$"
    match = re.search(padrao, codigoAcesso)
    for item in dict_id:
       for value in item.values():
          if str(match.group(1)) == str(value):
                    return True

#Criando a webpage:
def main():
    html_temp = """
    <body style="background-color:black">
    <div style ="background-image:linear-gradient(to right,#33ccff,
pink);padding:13;border-radius:25px;">
    <h1 style ="color:white;text-align:center;">BANCO IAM</h1>
    </div>
    <br/>
    </body>
    """
    st.markdown(html_temp, unsafe_allow_html=True)


    codigo_id = st.text_input('Código de Acesso:')
    if st.button("Acessar"):
        if (getFiltroCliente(codigo_id)):
            return st.warning("Solicitação negada. O usuário não pode
acessar a conta")
        else:
            return st.success("Pode Acessar a conta")

if __name__ == '__main__':
    main()
```

Visualizando o Arquivo no Colab:

Visualize como a formatação do código fica:

Colab-código-completo.

Testes

Realizando agora os testes do nosso código.

Teste Clientes Falecidos

Para testarmos a funcionalidade do código, testamos 44 códigos de acesso identificados como clientes falecidos, e este foi o resultado:

CÓDIGO DE ACESSO	Proibido?	Detectou como proibido?
20138	sim	sim
20824	sim	sim
22723	sim	sim
22439	sim	sim
21005	sim	sim
20605	sim	sim
21487	sim	sim
22828	sim	sim
22512	sim	sim
21206	sim	sim
22505	sim	sim
22835	sim	sim
21147	sim	sim
22701	sim	sim
20986	sim	sim
22680	sim	sim
20154	sim	sim
21096	sim	sim
20280	sim	sim
21566	sim	sim
20960	sim	sim
21333	sim	sim
22664	sim	sim
22099	sim	sim
21018	sim	sim
20869	sim	sim
21906	sim	sim

20501	sim	sim
20788	sim	sim
34445	não	não
34567	não	não
88888	não	não
90872	não	não
23456	não	não
23453	não	não
22222	não	não
33333	não	não
44444	não	não
55556	não	não
34567	não	não
98653	não	não
23456	não	não
12343	não	não
43457	não	sim

Resultado:

Vemos que o resultado foi quase perfeito, exceto por um código que ele interpretou como proibido sendo que, na verdade, não era. As métricas deram:

Precisão	F1-Score	Acurácia	Recall
0,97	0,98	0,97	1

Conclusão

Métricas

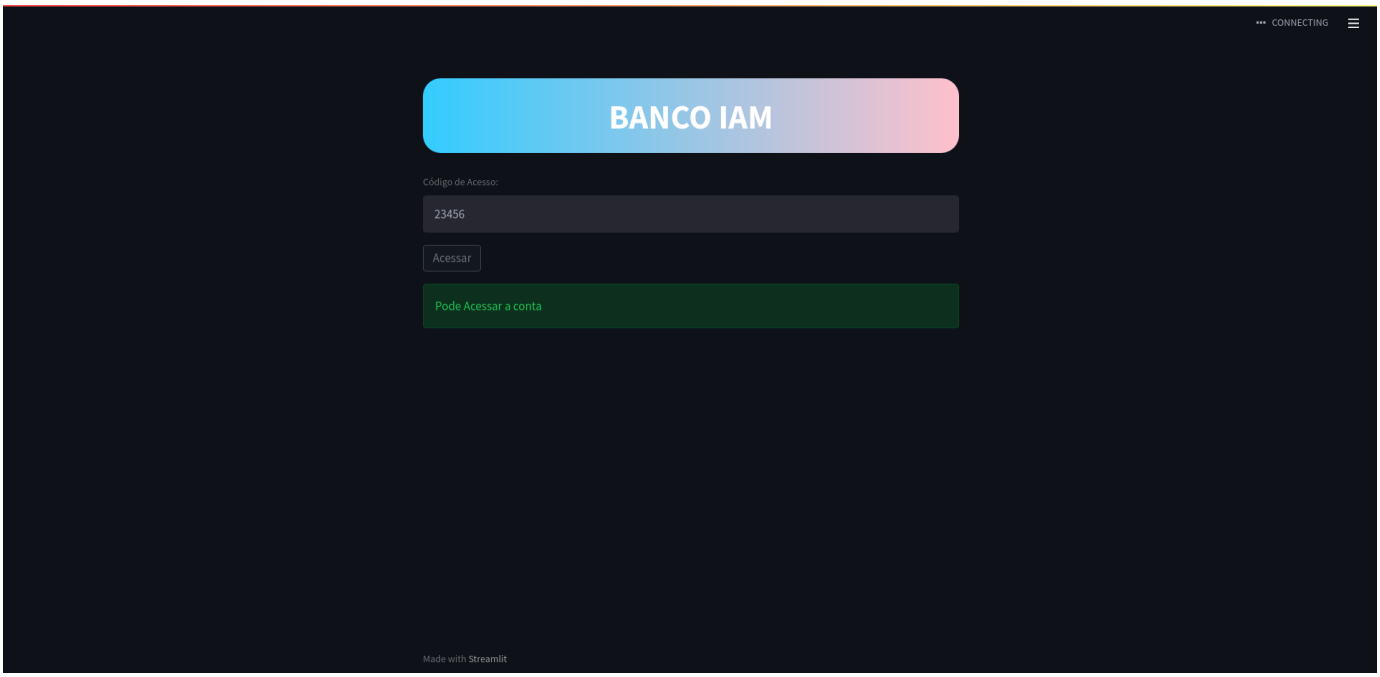
Para efetuarmos esse programa, escolhemos como métricas o F1-Score e o Recall.

Justificando:

- Recall: considerando que no ambiente de Bancos, primeiro se é avaliado a situação em que se encontra o usuário, e que nossa situação tratava-se de clientes falecidos, um dos processos que pode ocorrer ao acessar-se uma conta dessas é de imediato bloque-a-la e assim impedir seu uso indevido. O que pode acontecer com uma pessoa inocente, mas não é considerado uma medida extrema.
- F1-Score: considerado uma das medidas mais completas, melhor para se ter uma base de quão eficiente e bom é o desempenho do seu código.

Observações:

Os resultados parecem bons, porque estão de fato, mas devemos considerar que foi utilizado apenas uma API para testar o algoritmo e os dados não sofreram muitas alterações.


CONNECTING
BANCO IAM
Código de Acesso:
23456
Acessar
Pode Acessar a conta
Made with Streamlit